UXQCC

Release Version 1.0 DE

Juli 2026

UXQCC - GenAI Certified Professional in Software Usability with GenAI Foundation Level

Usability, User Experience und Accessibility mit GenAI gestalten, umsetzen und prüfen

UX QUALITY CERTIFICATION CENTER

Impressum, Urheberrecht und Nutzungshinweise

© 2026 Robert Pucher

Alle Inhalte dieses Werks, insbesondere Texte und Grafiken, sind urheberrechtlich geschützt. Urheber und alleiniger Rechteinhaber ist Robert Pucher. Herausgeber ist UXQCC; UXQCC ist berechtigt, dieses Werk im Rahmen des Zertifizierungsprogramms zu nutzen, zu pflegen, zu bearbeiten, zu übersetzen und an akkreditierte Trainingsprovider sowie Prüfungsprovider (z. B. GASQ) zu lizenzieren. Akkreditierte Trainings- und Prüfungsprovider sind berechtigt, dieses Werk im Rahmen ihrer Tätigkeit zu nutzen und zu veröffentlichen, etwa auf ihrer Website. Die Vervielfältigung oder Reproduktion des Werks, auch auszugsweise, ohne Zustimmung des Rechteinhabers ist im Übrigen untersagt. Bei Zuwiderhandlung behält sich der Rechteinhaber zivil- und strafrechtliche Schritte vor.

Dokumentinformation

Bezeichnung
Syllabus: UXQCC Certified Professional in Software Usability with GenAI – Foundation Level

Untertitel
Usability, User Experience und Accessibility mit GenAI gestalten, umsetzen und prüfen

Kurzform
Syllabus: UXQCC GenAI - FL

Herausgeber
UXQCC – User Experience Quality Certification Center
International Board for Usability and User Experience Qualification

Autor
FH-Prof. Dipl.-Ing. Dr. techn. Robert Pucher

Öffentliche Syllabus-Fassung und Provider-Fassung

Dieser Syllabus ist die öffentliche Fassung des Lehrplans zum UXQCC Certified Professional in Software Usability with GenAI – Foundation Level. Er beschreibt den fachlichen Rahmen, die Business Outcomes, Learning Objectives, zentralen Begriffe, Prüfungsabgrenzung und grundlegenden Inhalte der Zertifizierung.

Akkreditierte Trainingsprovider erhalten zusätzlich eine ausführlichere Provider-Fassung. Diese enthält vertiefende Erläuterungen, didaktische Hinweise, umfangreichere fachliche Ausführungen, Beispiele, Abgrenzungen und Hinweise zur Trainingsgestaltung. Sie dient der Vorbereitung, Durchführung und Qualitätssicherung offizieller Trainings.

Die öffentliche Syllabus-Fassung ist daher alleine nicht als vollständiges Trainingsmaterial zu verstehen. Sie legt fest, welche Inhalte und Lernziele für die Zertifizierung maßgeblich sind, ersetzt aber nicht die offiziellen Trainingsunterlagen akkreditierter Trainingsprovider.

Danksagung

Mein besonderer Dank gilt den vielen Personen, insbesondere Frau Dr. Verena Seibert-Giller, für ihre wertvollen fachlichen Beiträge, kritischen Rückmeldungen und konstruktiven Anregungen bei der Entwicklung dieses Syllabus.

Ihre unterschiedlichen Perspektiven aus Praxis, Lehre, Softwareentwicklung, Qualitätssicherung, Usability und User Experience haben wesentlich dazu beigetragen, die Inhalte fachlich zu schärfen, praxisnah auszurichten und für die Zielgruppen verständlich aufzubereiten.

Ebenso danke ich allen weiteren Personen, die durch Diskussionen, Hinweise, Reviews oder fachliche Impulse zur Weiterentwicklung dieses Dokuments beigetragen haben. Ihre Beiträge haben geholfen, den Syllabus nicht nur als fachlichen Rahmen, sondern auch als anschlussfähige Grundlage für Training, Zertifizierung und praktische Anwendung in Softwareentwicklungsteams zu gestalten.

Versionen

Version	Datum	Anmerkung
1.0	14.07.2026	Intern geprüftes Dokument

Aktuelle Version 1.0 (Build 1.013)

Disclaimer und Nutzungshinweis

Dieses Dokument dient der allgemeinen Information über die Verbesserung von Software-Usability, User Experience und Interaktionsqualität mit Unterstützung von Generativer Künstlicher Intelligenz (GenAI) unter Berücksichtigung verfügbarer ergonomischer Normen und Qualitätsstandards, insbesondere ISO 9241 und ISO/IEC 25010.

Die Autoren und UXQCC übernehmen keine Gewähr für Vollständigkeit, Richtigkeit oder Aktualität der Inhalte. Das Dokument ersetzt keine individuelle UX-Beratung und keine verbindliche Konformitätsprüfung, etwa im Sinne gesetzlicher Anforderungen oder des European Accessibility Act.

Da GenAI-Systeme Halluzinationen erzeugen und den spezifischen Domänenkontext eines Produkts nicht eigenständig erfassen können, dient dieser Syllabus u.a. als Leitfaden zur kritischen Beurteilung von AI-Ergebnissen. Die Verantwortung für Softwarequalität, Software-Usability, Barrierefreiheit und rechtliche Konformität bleibt stets bei den verantwortlichen Personen und Organisationen.

Für verbindliche Umsetzungen sind die jeweils aktuellen gesetzlichen Bestimmungen, offiziellen Standards, insbesondere WCAG 2.2, sowie bei Bedarf fachkundige Beratung heranzuziehen.

Hinweis zur beschreibenden Verwendung von Scrum

Scrum wird in diesem Syllabus ausschließlich beschreibend als ein verbreiteter agiler Prozessrahmen verwendet. Der Syllabus ist unabhängig und steht in keiner Verbindung zu Scrum.org, Scrum Alliance oder Scrum Inc. Er wird von diesen Organisationen weder herausgegeben, geprüft, unterstützt, autorisiert noch gesponsert.

Hinweis zur Verwendung von Artificial Intelligence (AI)

Dieser Syllabus wurde teilweise mit Unterstützung von AI-Werkzeugen erarbeitet und redaktionell geprüft.

Inhalt

Impressum, Urheberrecht und Nutzungshinweise 1

Inhalt 4

0 Präambel 7

0.1 Zweck des Syllabus 7
0.2 Zielgruppen für den Syllabus 8
0.3 Prüfungsrelevante Lernziele und kognitive Wissensstufen 11
0.4 Die Prüfung 12
0.5 Akkreditierung 13
0.6 Detaillierungsgrad 13
0.7 Struktur dieses Syllabus 14
0.8 Begleitende Prompt-Bibliothek und Praxisbeispiele 15
0.9 Business Outcomes (BOs) 17
0.10 Verwendete Kürzel und Akronyme 22

1 Usability mit AI: Warum das Thema relevant ist 28

1.1 Leitgedanke des Kapitels 28
1.2 Beitrag zu den Business Outcomes 28
1.3 Learning Objectives 28
1.4 Wichtige Begriffe 29
1.5 Software-Usability, UX und Interaction Capability als Bestandteile von Softwarequalität 29
1.6 Usability in Entwicklungsprozessen 30
1.7 GenAI als Unterstützung entlang agiler Aktivitäten 31
1.8 Bekannte Muster, unbekannter Kontext und AI-Output 31
1.9 Verantwortungsvoller AI-Einsatz im Usability-Kontext 32

2 Mit AI Usability-Basiswissen aufbauen 33

2.1 Leitgedanke des Kapitels 33
2.2 Beitrag zu den Business Outcomes 33
2.3 Learning Objectives 33
2.4 Wichtige Begriffe 34
2.5 Basiswissen zu Usability als Voraussetzung für AI-gestützte Arbeit 35
2.6 Menschliche Faktoren für UX 36
2.7 GenAI als Lernassistent für Usability-Grundlagen 36
2.8 Prompting als Lern- und Reflexionstechnik 37
2.9 AI-Output kritisch beurteilen 37
2.10 Bekannte Muster und unbekannter Kontext 38
2.11 AI-gestützte Usability-Reflexion und menschliche Prüfung 39

3 Discovery: Nutzer, Kontext, Probleme und Hypothesen verstehen 40

3.1 Leitgedanke des Kapitels 40
3.2 Beitrag zu den Business Outcomes 40
3.3 Learning Objectives 41
3.4 Wichtige Begriffe 41
3.5 Discovery im Softwareentwicklungsprozess 42
3.6 Nutzer, Nutzerziel, Aufgabe und Nutzungskontext 42
3.7 Typische Informationsquellen für Usability-Discovery 43
3.8 GenAI zur Strukturierung vorhandener Informationen 44
3.9 Von Informationen zu Problemfeldern für Usability 45
3.10 Hypothesen statt Gewissheiten 46
3.11 Personas und Nutzungsszenarien mit Vorsicht einsetzen 47
3.12 Wann reale Nutzer, Domänenexperten oder zusätzliche Research-Aktivitäten erforderlich sind 48
3.13 Discovery-Erkenntnisse in anschlussfähige Artefakte überführen 48

3.14 AI-gestützte Discovery und menschliche Prüfung 49

4 Backlog Refinement und Usability 51

4.1 Leitgedanke des Kapitels 51
4.2 Beitrag zu den Business Outcomes 51
4.3 Learning Objectives 51
4.4 Wichtige Begriffe 52
4.5 Backlog Refinement als Übersetzungspunkt 52
4.6 Usability-Requirements als Bestandteil professioneller Requirements-Arbeit 53
4.7 Implizite Usability-Anforderungen sichtbar machen 54
4.8 User Stories aus Nutzungsperspektive prüfen 54
4.9 Backlog Items mit GenAI prüfen 55
4.10 Akzeptanzkriterien mit Usability-Aspekten ergänzen 56
4.11 Definition of Ready und Definition of Done mit Usability-Bezug 57
4.12 AI-generierte Anforderungen fachlich einordnen 57
4.13 AI-gestütztes Refinement und menschliche Prüfung 58

5 Planung und Teamorganisation 60

5.1 Leitgedanke des Kapitels 60
5.2 Beitrag zu den Business Outcomes 60
5.3 Learning Objectives 60
5.4 Wichtige Begriffe 61
5.5 Planung als Übergang von Backlog zu Teamarbeit 61
5.6 Usability-Aufgaben in der Planung sichtbar machen 62
5.7 Rollen und Verantwortung im Softwareentwicklungsteam 63
5.8 Einfache Usability-Aktivitäten organisatorisch einplanen 64
5.9 Wo Teams selbst handeln können und wo Expertise notwendig ist 65
5.10 Design Critiques als strukturiertes Format 65
5.11 GenAI zur Vorbereitung und Strukturierung von Teamaktivitäten 66
5.12 Dokumentation des AI-Einsatzes 66
5.13 AI-gestützte Teamorganisation und menschliche Prüfung 67

6 UI, Informationsarchitektur, States und AI-generierte Entwürfe 69

6.1 Leitgedanke des Kapitels 69
6.2 Beitrag zu den Business Outcomes 69
6.3 Learning Objectives 70
6.4 Wichtige Begriffe 70
6.5 Vom Backlog Item zur sichtbaren Lösung 71
6.6 UI-Design-Grundlagen zur Beurteilung von Entwürfen 72
6.7 Informationsarchitektur, Navigation und Auffindbarkeit 73
6.8 Prototyp-Arten und deren Zweck 74
6.9 AI-gestütztes Rapid Prototyping in der Umsetzung 75
6.10 AI-generierte UI-Entwürfe beurteilen 76
6.11 Bekannte UI-Muster als Stärke und Risiko von AI 77
6.12 States, Edge Cases und alternative Pfade 77
6.13 UI-Patterns, Onboarding und Hilfen 78
6.14 Prototypen als Grundlage für Kommunikation, Lernen, Evaluation und Validierung 79
6.15 AI-gestütztes Design, Prototyping und menschliche Prüfung 80

7 Code-Erstellung, Komponenten, Accessibility und Handoff 81

7.1 Leitgedanke des Kapitels 81
7.2 Beitrag zu den Business Outcomes 81
7.3 Learning Objectives 82
7.4 Wichtige Begriffe 82
7.5 Von Usability-Anforderungen zur implementierten UI 83
7.6 AI-gestützte Code-Erstellung für Usability und Frontend 84

7.7 Usability in der Implementierung 85
7.8 Komponenten, Designsysteme und Design Tokens 86
7.9 AI zur Ableitung von UI-Mustern und Designsystem-Bausteinen 87
7.10 Handoff zwischen Design, AI und Entwicklung 88
7.11 Accessibility-Basics in der Umsetzung 88
7.12 AI für Dokumentation, Tests und Reviews 90
7.13 Risiken AI-generierter Implementierung 91
7.14 Prüfmaßnahmen für AI-generierten Code 92
7.15 AI-gestützte Umsetzung und menschliche Prüfung 93

8 Evaluation, Testing und Validierung 95

8.1 Leitgedanke des Kapitels 95
8.2 Beitrag zu den Business Outcomes 96
8.3 Learning Objectives 96
8.4 Wichtige Begriffe 97
8.5 Evaluation, Testing und Validierung im Softwareentwicklungsprozess 97
8.6 Usability in Review-Formaten sichtbar machen 98
8.7 Heuristische Evaluation mit AI-Unterstützung 99
8.8 Cognitive Walkthrough 100
8.9 Usability-Tests planen 100
8.10 Testaufgaben formulieren 102
8.11 Moderation, Think-Aloud und strukturierte Beobachtung 102
8.12 Beobachtungen, Findings und Berichtsentwürfe strukturieren 103
8.13 Probleme der Usability priorisieren 104
8.14 Aus AI-Vorschlägen testbare Annahmen ableiten 105
8.15 AI-Simulationen und synthetische Nutzer 106
8.16 AI-gestützte Auswertung und menschliche Prüfung 107

9 Release, Monitoring und Verbesserungen 109

9.1 Leitgedanke des Kapitels 109
9.2 Beitrag zu den Business Outcomes 109
9.3 Learning Objectives 110
9.4 Wichtige Begriffe 110
9.5 Usability nach dem Release beobachten 111
9.6 Quellen für Erkenntnisse zu Usability nach dem Release 112
9.7 Warum Usability nach dem Release weiter verbessert werden muss 112
9.8 UX-Metriken und HEART-Framework 113
9.9 GenAI zur Strukturierung von Feedback, Tickets und Metriken 115
9.10 Muster, Korrelation, Ursache und Priorität unterscheiden 116
9.11 UX-Debt erkennen 117
9.12 Erkenntnisse in Backlog Items und Verbesserungsmaßnahmen überführen 118
9.13 Kontinuierliche Verbesserung im Team verankern 119
9.14 Externe Unterstützung und Grenzen teaminterner AI-Nutzung 120
9.15 AI-gestütztes Monitoring und menschliche Prüfung 120

10 Literatur 122

11 Anhang 1 – Kurzglossar zentraler Begriffe 124

0 Präambel

0.1 Zweck des Syllabus

Dieser Syllabus beschreibt die angestrebten Geschäftsergebnisse (Business Outcomes), Lernziele (Learning Objectives) und grundlegenden Konzepte für den Syllabus *UXQCC Certified Professional in Software Usability with GenAI – Foundation Level.*

Der Syllabus legt fest, welche fachlichen Inputs in einem Training von etwa 16 Stunden vermittelt werden sollen. Er richtet sich an Softwareentwicklungsteams, die Software-Usability und User Experience mit Unterstützung generativer künstlicher Intelligenz früher, pragmatischer und systematischer in ihren Entwicklungsprozess integrieren möchten.

Dieser Syllabus verwendet die Begriffe Software-Usability, User Experience und Interaction Capability in ihrer jeweiligen Bedeutung für Software Engineering. Usability bezeichnet die Gebrauchstauglichkeit von Software in einem bestimmten Nutzungskontext. Im Mittelpunkt steht die Frage, ob bestimmte Nutzer ihre Ziele mit einem System effektiv, effizient und zufriedenstellend erreichen können. User Experience, kurz UX, ist der breitere Rahmen und umfasst zusätzlich Erwartungen, Wahrnehmungen und Reaktionen vor, während und nach der Nutzung eines Produkts oder Systems. Interaction Capability bezeichnet in ISO/IEC 25010 die produktbezogene Qualität der Mensch-System-Interaktion im Softwarequalitätsmodell. Im Syllabus wird UX daher nicht als nachgelagerte optische Gestaltung verstanden, sondern als Bestandteil von Software-Usability, Softwarequalität und überprüfbarer Interaktionsqualität.

Zur besseren Lesbarkeit wird im weiteren Text überwiegend der Begriff Usability verwendet. Er steht stellvertretend für Software-Usability im oben definierten Sinn und schließt die in diesem Syllabus behandelten UX-Aspekte mit ein. UX wird dort ausdrücklich genannt, wo der breitere Erlebnisrahmen gemeint ist oder wo feststehende Fachbegriffe wie UX-Qualität, UX-Debt, UX-Metriken oder Rollenbezeichnungen wie UX-Designerin und UX-Designer verwendet werden.

Im Mittelpunkt steht der Aufbau einer belastbaren Grundkompetenz für Usability in Softwareentwicklungsteams. GenAI wird dabei als Lern-, Strukturierungs-, Entwurfs-, Prüf- und Unterstützungswerkzeug betrachtet. Gleichzeitig betont der Syllabus die Grenzen von AI: AI kann bekannte Muster nutzen, Hypothesen erzeugen und Vorschläge formulieren, ersetzt aber keine menschliche Bewertung, kein Domänenwissen, keine reale Nutzerbeobachtung und keine fachliche Verantwortung.

Der Syllabus orientiert sich bewusst an einem typischen agilen Softwareentwicklungsprozess und verwendet Scrum dabei beschreibend als verbreiteten Referenzrahmen. Die Inhalte folgen daher nicht einer klassischen HCI-Systematik, sondern den Arbeitsschritten, in denen Usability in agilen Softwareentwicklungsteams entsteht, beeinflusst, geprüft oder verbessert wird: Orientierung, Kompetenzaufbau, Discovery, Backlog Refinement, Sprint Planning, Design und Prototyping, Umsetzung, Review-Formate, Evaluation und Testing sowie Release, Monitoring und kontinuierliche Verbesserung.

0.2 Zielgruppen für den Syllabus

Dieser Lehrplan richtet sich in erster Linie an Softwareentwicklerinnen und Softwareentwickler sowie an Testerinnen, Tester und Qualitätssicherungsteams in Softwareentwicklungsteams. Sie setzen heute mit AI-Unterstützung in kurzer Zeit Anforderungen, UI-Komponenten, Code und Tests um und entscheiden damit faktisch über die Usability der entstehenden Software, häufig ohne dass eine UX-Spezialrolle im Team verfügbar ist.

Ebenso angesprochen sind alle weiteren Rollen, die an diesem Prozess beteiligt sind: Product Owner, Requirements Engineers und Business Analysts, UI-Designerinnen und -Designer, UX-Spezialistinnen und UX-Spezialisten, Scrum Master und agile Coaches sowie Führungskräfte, die Rahmenbedingungen für den verantwortungsvollen AI-Einsatz schaffen.

UX-Spezialistinnen und UX-Spezialisten finden im Syllabus die gemeinsame Sprache und die Prozessanker, über die ihr Fachwissen für Entwicklungsteams anschlussfähig wird und können AI zielgerichtet einsetzen, um ihre Arbeit effizienter zu machen.

Besonders relevant ist der Syllabus für Organisationen, in denen AI-Werkzeuge bereits für Anforderungen, Design, Code, Tests, Dokumentation oder Analyse eingesetzt werden, Usability-Arbeit aber noch nicht ausreichend systematisch im Entwicklungsprozess verankert ist. Dies betrifft insbesondere Teams, in denen keine dauerhaft verfügbare UX-Spezialrolle vorhanden ist oder Usability-Aspekte bisher zu spät, zu unsystematisch oder nur punktuell berücksichtigt werden.

Ziel ist nicht die Ausbildung von UX-Spezialistinnen und UX-Spezialisten, sondern der Aufbau einer belastbaren Grundkompetenz für Usability im Team mit Unterstützung von AI. AI wird dabei zur Beschleunigung von Entwicklungsarbeit genutzt. AI-Ergebnisse sind jedoch aus Nutzungs- und Qualitätsperspektive kritisch zu prüfen, um Risiken für die Usability zu erkennen und Usability als Bestandteil professioneller Softwarequalität in der täglichen Teamarbeit zu verankern.

Der Syllabus ist insbesondere für Teams und Personen relevant,

- die Software schneller mit AI entwickeln möchten und besonders Wert darauf legen, Usability, Verständlichkeit und Nutzbarkeit zu integrieren,
- die keine dauerhaft verfügbare UX-Spezialrolle im Team haben,
- die User Stories, Akzeptanzkriterien, UI-Entwürfe, Prototypen, Code, Tests oder Feedbackdaten mit GenAI unterstützen wollen,
- die AI-generierte Ergebnisse nicht nur übernehmen, sondern fachlich, methodisch und aus Nutzungsperspektive bewerten möchten,
- die Usability, Accessibility-Basics und UX-Qualität früher im agilen Entwicklungsprozess sichtbar machen möchten,
- die ihre Entwicklungs-, Test- und Produktentscheidungen stärker an Nutzerzielen, Nutzungskontext und überprüfbaren Qualitätskriterien ausrichten wollen,
- die ihre UX Expertise effizient in die Softwareentwicklung einbringen wollen.

Im Einzelnen richtet sich der Lehrplan an folgende Zielgruppen:

Softwareentwicklerinnen und Softwareentwickler

- setzen UI-nahe Anforderungen, Komponenten, Zustände, Validierung, Fehlermeldungen und Accessibility-Basics um
- nutzen AI-gestützte Coding-Werkzeuge für Frontend, Komponenten, Tests, Dokumentation und Refactoring
- lernen, AI-generierten Code nicht nur technisch, sondern auch aus Usability-Sicht zu prüfen
- erkennen, dass schnelle Code-Erstellung nur dann Wert schafft, wenn die resultierende Software von den vorgesehenen Nutzerinnen und Nutzern im jeweiligen Nutzungskontext effektiv, effizient und zufriedenstellend genutzt werden kann und dabei verständlich, bedienbar und robust umgesetzt ist

Testerinnen, Tester und Qualitätssicherungsteams

- prüfen nicht nur funktionale Korrektheit, sondern auch Verständlichkeit, Zustände, Fehlerfälle, Tastaturbedienbarkeit und grundlegende Accessibility
- nutzen GenAI zur Vorbereitung von Testfällen, Review-Checklisten, Beobachtungsbögen und Findings
- unterscheiden AI-generierte Hinweise, heuristische Einschätzungen und reale Testbeobachtungen
- tragen dazu bei, dass Usability als prüfbares Qualitätsmerkmal in Reviews, Tests und Abnahmen sichtbar wird

Product Owner, Produktverantwortliche und Product Manager

- formulieren Nutzerziele, Produktannahmen, Backlog Items und Akzeptanzkriterien mit Usability-Bezug
- priorisieren Usability-Probleme, UX-Debt und Verbesserungsmaßnahmen
- nutzen GenAI zur Strukturierung von Feedback, Hypothesen, Anforderungen und Entscheidungsoptionen
- stellen sicher, dass AI-gestützte Produktentscheidungen nicht nur plausibel klingen, sondern zum tatsächlichen Nutzungskontext passen

Requirements Engineers, Business Analysts und Fachkonzepterinnen bzw. Fachkonzepter

- übersetzen Nutzerprobleme, Nutzungskontexte und Produktannahmen in Backlog Items und prüfbare Anforderungen
- nutzen AI zur Strukturierung von Feedback, Supporttickets, Interviews und Fachinformationen
- prüfen AI-generierte Anforderungen auf fachliche Richtigkeit, Kontextpassung, implizite Annahmen und Validierungsbedarf
- machen sichtbar, wo funktionale Anforderungen durch Usability- und Accessibility-Kriterien ergänzt werden müssen

UX-Spezialistinnen und UX-Spezialisten

- übersetzen ihr Wissen über Nutzer, Nutzungskontexte, Research und Evaluationsmethoden in die Arbeitsformen der Softwareentwicklung, etwa in Backlog Items, Akzeptanzkriterien, Definition of Ready und Definition of Done, Review-Formate und Tests
- verankern Usability-Kriterien dort, wo Entwicklungsteams tatsächlich entscheiden: im Refinement, in der Planung, im Code Review und in der Evaluation
- nutzen GenAI, um ihre Expertise im Team zu skalieren, etwa durch wiederverwendbare Prüffragen, Checklisten, Heuristik-Reviews und die Vorbereitung von Usability-Tests
- prüfen AI-generierte Anforderungen, Entwürfe und Auswertungen fachlich und methodisch und befähigen das Team, diese Prüfung zunehmend selbst vorzunehmen

UX- und UI-Designerinnen und Designer

- integrieren GenAI in Design, Prototyping, Variantenbildung und Design Critiques
- beurteilen AI-generierte Entwürfe anhand von Nutzerziel, visueller Hierarchie, mentalen Modellen, kognitiver Belastung, Konsistenz und Accessibility
- unterstützen agile Teams beim Aufbau gemeinsamer Usability-Kriterien

- profitieren von einem gemeinsamen Verständnis, das die Zusammenarbeit mit Entwicklung, QA, Product Ownern und Stakeholdern erleichtert

Scrum Master, agile Coaches und Personen mit Verantwortung für Teamprozesse

- unterstützen die Verankerung von Usability-Aktivitäten in agilen Entwicklungsprozessen
- machen Usability-Aufgaben, Verantwortlichkeiten, Review-Formate und Lernschleifen im Team sichtbar
- fördern die reflektierte Nutzung von AI im Team, ohne Verantwortung an AI zu delegieren
- helfen dabei, dass UX nicht als Zusatzaufgabe am Ende, sondern als Teil der laufenden Teamarbeit verstanden wird

Führungskräfte, Team Leads und Entscheidungsverantwortliche

- verstehen Usability mit GenAI als Teil von Softwarequalität, Produktqualität und AI Governance
- schaffen organisatorische Rahmenbedingungen für verantwortungsvollen AI-Einsatz, Datenschutz, Nachvollziehbarkeit und menschliche Kontrolle
- erkennen, wann Teams einfache Usability-Aktivitäten selbst übernehmen können und wann zusätzliche Usability-, Research-, Accessibility- oder Domänenexpertise erforderlich ist
- können die Schulung als pragmatische Maßnahme nutzen, um AI-Einsatz, Softwarequalität und Kunden- bzw. Nutzerzufriedenheit besser miteinander zu verbinden

Externe Dienstleister, Agenturen, Trainerinnen und Trainer

- können den Syllabus als Grundlage für Trainings, Workshops oder Qualifizierungsmaßnahmen verwenden
- ergänzen eigenständig passende Übungen, Fallbeispiele und Tool-Demonstrationen
- achten darauf, dass AI-gestützte Usability-Arbeit als verantwortete Teamkompetenz und nicht als bloße Tool-Bedienung vermittelt wird
- unterstützen Organisationen dabei, GenAI nicht nur als Effizienzwerkzeug, sondern als Anlass für bessere Usability- und Qualitätsprozesse zu nutzen

0.3 Prüfungsrelevante Lernziele und kognitive Wissensstufen

Die Lernziele (Learning Objectives) unterstützen die Geschäftsergebnisse (Business Outcomes) und dienen als Grundlage für die Strukturierung der Trainingsinhalte und gegebenenfalls für die Erstellung einer Prüfung.

Von den Lernenden kann erwartet werden, dass sie zentrale Inhalte des Syllabus wiedergeben, Zusammenhänge erklären und Wissen in typischen praxisnahen Situationen anwenden können. Der Fokus liegt auf einem Foundation-Level: Die Teilnehmenden sollen GenAI im Kontext von Usability nicht nur bedienen, sondern AI-Ergebnisse einordnen, kritisch prüfen und auf typische Artefakte der Softwareentwicklung übertragen können.

Die Wissensstufen (Knowledge Levels) der jeweiligen Learning Objectives werden zu Beginn jedes Kapitels angegeben und wie folgt klassifiziert:

K1 – Erinnern:
Fakten, Begriffe und Informationen gezielt abrufen und wiedergeben.

K2 – Verstehen:
Bedeutungen erfassen, Zusammenhänge erklären und Konzepte einordnen.

K3 – Anwenden:
Wissen in konkreten Situationen oder auf typische Artefakte anwenden, zum Beispiel auf User Stories, Akzeptanzkriterien, UI-Entwürfe, Prototypen, Code, Tests, Feedbackdaten oder UX-Debt.

Die höheren Stufen K4 bis K6 der revidierten Bloom'schen Taxonomie werden in diesem Syllabus nicht als eigenständige Prüfungsstufen verwendet. Sie können jedoch in vertiefenden Trainings, praktischen Übungen, Projektarbeiten oder weiterführenden Zertifizierungen eine Rolle spielen.

0.4 Die Prüfung

Die Prüfung zum *UXQCC Certified Professional in Software Usability with GenAI – Foundation Level* basiert auf den Inhalten dieses Syllabus. Antworten auf Prüfungsfragen können die Nutzung von Inhalten aus mehreren Abschnitten dieses Syllabus erfordern. Alle Kapitel mit fachlichem Inhalt sind prüfungsrelevant, mit Ausnahme der Präambel und möglicher Anhänge, sofern diese nicht ausdrücklich als prüfungsrelevant gekennzeichnet werden.

Standards, Bücher, Artikel, Online-Ressourcen, rechtliche Vorschriften und Tool-Dokumentationen können als Referenzen herangezogen werden. Ihre Inhalte sind jedoch nur insoweit prüfungsrelevant, wie sie in diesem Syllabus zusammengefasst oder ausdrücklich behandelt werden.

Die Prüfung besteht aus **40 Multiple-Choice-Fragen**. Jede richtige Antwort wird mit einem Punkt bewertet. Zum Bestehen der Prüfung ist eine Mindestpunktzahl von **65 %** erforderlich, also mindestens **26 korrekt beantwortete Fragen**. Für die Bearbeitung der Prüfung stehen **60 Minuten** zur Verfügung. Ist die Muttersprache der geprüften Person nicht die Sprache der Prüfung, kann eine zusätzliche

Prüfungszeit von **25 %**, also **15 Minuten**, gewährt werden.

0.5 Akkreditierung

Schulungsanbieter, die Trainings zum *UXQCC Certified Professional in Software Usability with GenAI – Foundation Level* anbieten möchten, müssen akkreditiert sein und die offiziellen Schulungsunterlagen verwenden. Sie müssen sicherstellen, dass ihre Beispiele, Tool-Demonstrationen und Übungen mit den Business Outcomes und Learning Objectives dieses Syllabus übereinstimmen.

Trainingsanbieter sollten insbesondere darauf achten, dass die Grenzen von AI, Datenschutz, menschliche Prüfung, Governance, Usability, Accessibility, Code-Qualität und Validierung nicht zugunsten reiner Tool-Demonstrationen vernachlässigt werden.

Für akkreditierte Trainingsanbieter stehen ergänzend zur öffentlichen Syllabus-Fassung zusätzliche Materialien zur Verfügung. Dazu gehören insbesondere ein ausführlicher, didaktisch aufbereiteter Provider-Syllabus, ein Basis-Slide-Set für offizielle Trainings sowie zahlreiche Übungsbeispiele mit begleitenden Beispielprompts zur praktischen Vertiefung der Inhalte. Diese Materialien dienen der Vorbereitung, Durchführung und Qualitätssicherung akkreditierter Trainings und sind ausschließlich im Rahmen der jeweils geltenden Lizenzbedingungen zu verwenden.

0.6 Detaillierungsgrad

Diese öffentliche Fassung ist bewusst kompakter gehalten als die ausführliche Provider-Fassung. Sie beschreibt die prüfungsrelevanten Business Outcomes, Learning Objectives, zentralen Begriffe und fachlichen Inhalte in einem rahmensetzenden Format.

Akkreditierte Trainingsprovider erhalten eine umfassendere Fassung als Trainingsgrundlage. Diese enthält zusätzliche Erläuterungen, didaktische Hinweise, Beispiele, Abgrenzungen und vertiefende fachliche Ausführungen zur Gestaltung offizieller Trainings.

Die Kürze dieser öffentlichen Fassung bedeutet daher nicht, dass Trainings auf diesen Textumfang beschränkt sind. Offizielle Trainings können und sollen die Inhalte didaktisch ausarbeiten, mit Übungen, Beispielen, Tool-Demonstrationen und Praxisfällen ergänzen und an die Zielgruppen anpassen, solange sie mit den Business Outcomes und Learning Objectives dieses Syllabus übereinstimmen.

Dieser Syllabus besteht aus:

- Business Outcomes, die festlegen, was nach der Lerneinheit in der beruflichen Praxis geleistet werden kann, um Usability mit GenAI in Softwareentwicklungsteams sinnvoll, verantwortungsvoll und wirksam einzusetzen
- allgemeinen Lernzielen, die die Zielsetzung des Syllabus beschreiben
- kapitelbezogenen Learning Objectives mit zugeordneten kognitiven Wissensstufen
- wichtigen Begriffen je Kapitel, die im jeweiligen Kontext verstanden und verwendet werden sollen
- Beschreibungen zentraler Konzepte, Vorgehensweisen und Grenzen
- relevanter Fachliteratur, Standards und Online-Ressourcen

Der Inhalt des Syllabus stellt keine vollständige Darstellung der gesamten Wissensgebiete Software-Usability, User Experience, Human-Computer Interaction, Accessibility, Software Engineering, Scrum oder Generative AI dar. Er beschreibt den Detaillierungsgrad, der in einem kompakten Foundation-Level-Training von etwa 16 Stunden behandelt werden kann.

Der Syllabus legt fest, **welche** Inhalte, Business Outcomes und Learning Objectives behandelt werden. Er enthält jedoch keine verbindlichen Vorgaben dazu, **wie** diese Inhalte didaktisch umgesetzt werden, welche konkreten Übungen verwendet oder welche Tools eingesetzt werden. Die Ausgestaltung praktischer Übungsanteile liegt daher bei den Trainingsanbietern.

Für die praktische Umsetzung werden daher ergänzende **empfohlene Praxisunterlagen** bereitgestellt. Dazu zählen insbesondere strukturierte Prompt-Beispiele und eine begleitende Prompt-Bibliothek. Diese Materialien zeigen exemplarisch, wie Inhalte des Syllabus mit GenAI praktisch erarbeitet, vertieft und kritisch geprüft werden können. Die Nutzung der Prompt-Bibliothek erfolgt im Rahmen der jeweils geltenden Lizenzbedingungen.

Als fachliche Vertiefung kann ergänzend das Buch **Pucher/Simandl, *UX Design mit AI*** herangezogen werden. Es ist inhaltlich auf den Syllabus abgestimmt und so aufgebaut, dass es das Lernen, Wiederholen und praktische Anwenden der Inhalte gezielt unterstützt. Maßgeblich für Business Outcomes, Learning Objectives und Prüfungsabgrenzung bleibt jedoch dieser Syllabus.

0.7 Struktur dieses Syllabus

Der Syllabus ist nicht entlang einer klassischen HCI-Systematik aufgebaut. Die Struktur folgt bewusst einem typischen Softwareentwicklungsprozess mit agilen Arbeitsschritten, damit die Inhalte für Softwareentwicklungsteams direkt anschlussfähig sind. Scrum wird dabei beschreibend als verbreiteter Referenzrahmen verwendet.

Es gibt neun Kapitel mit fachlich prüfungsrelevantem Inhalt. Die Überschrift jedes Kapitels gibt die vorgesehene Unterrichtszeit an; innerhalb der Kapitel erfolgt keine weitere verbindliche Zeitangabe auf Unterebene. Für den Trainingskurs schlägt der Lehrplan insgesamt etwa **960 Minuten**, also **16 Stunden**, Unterricht vor.

Die tatsächlich benötigten Unterrichtszeiten können abhängig sein von Vorwissen der Teilnehmenden, eingesetzten Beispielen, gewählten Tools, Diskussionsanteilen, praktischen Übungen, Branchenkontext und didaktischer Gestaltung durch den Trainingsanbieter.

Unterrichtsstruktur und Vorschlag zur Zeitplanung

Kapitel	Titel	Dauer in Minuten (ca.)	Stunden (ca.)
1	Usability mit AI: Warum das Thema relevant ist	90	1,5
2	Mit AI Usability-Basiswissen aufbauen	120	2
3	Discovery: Nutzer, Kontext, Probleme und Hypothesen verstehen	120	2
4	Backlog Refinement und Usability	120	2
5	Planung und Teamorganisation	90	1,5
6	UI, Informationsarchitektur, States und AI-generierte Entwürfe	135	2,25
7	Code-Erstellung, Komponenten, Accessibility und Handoff	135	2,25
8	Evaluation, Testing und Validierung	150	2,5
9	Release, Monitoring und Verbesserungen	90	1,5
	Summenzeiten	**960**	**16**

0.8 Begleitende Prompt-Bibliothek und Praxisbeispiele

Der Syllabus beschreibt, welche Inhalte, Business Outcomes und Learning Objectives in einem Training behandelt werden sollen. Damit legt er den fachlichen Rahmen fest, beschreibt jedoch nicht im Detail, wie einzelne Inhalte didaktisch erarbeitet, mit GenAI praktisch umgesetzt oder in konkreten Übungen vertieft werden. Um auch diese Anwendungsebene sichtbar zu machen, wird ergänzend zum Syllabus eine strukturierte Prompt-Bibliothek mit begleitenden Praxisbeispielen bereitgestellt.

Diese Materialien richten sich an Teilnehmende, Trainingsanbieter und Selbstlernende. Sie zeigen nicht nur, welche Inhalte im Sinne des Syllabus behandelt werden sollen, sondern auch, wie GenAI in typischen Situationen konkret angeleitet, befragt, geprüft und iterativ genutzt werden kann. Die Prompt-Bibliothek enthält daher keine bloße Sammlung einzelner Eingaben, sondern erläutert den Zweck, den Einsatzkontext, die erwartete Art des AI-Outputs, typische Grenzen sowie notwendige menschliche Prüfschritte.

Da sich GenAI-Werkzeuge, verfügbare Funktionen, Einsatzmöglichkeiten und typische Tool-Workflows sehr schnell weiterentwickeln, werden diese Praxisunterlagen vorzugsweise online bereitgestellt und laufend gepflegt. Dadurch können Prompt-Beispiele, Tool-Hinweise und Anwendungsszenarien aktualisiert werden, ohne den stabilen fachlichen Rahmen des Syllabus verändern zu müssen.

Damit entsteht eine Brücke zwischen dem fachlichen Rahmen des Syllabus und der praktischen Anwendung von Usability mit GenAI in Trainings, Workshops, Selbststudium und projektbezogenen Lernsettings.

0.9 Business Outcomes (BOs)

Generelle Business Outcomes

BO-G 1

Usability-Arbeit mit Hilfe von GenAI früher, pragmatischer und systematischer in Softwareentwicklungsteams verankern.

BO-G 2

AI-gestützte Usability-Ergebnisse kritisch bewerten und zwischen erkannten Mustern, plausiblen Hypothesen, AI-generierten Vorschlägen und tatsächlich validierten Erkenntnissen unterscheiden.

BO-G 3

Grundlegende Kriterien für Usability einschließlich Wahrnehmung, Verständlichkeit, kognitiver Belastung, Accessibility und erwartbarem Verhalten in Discovery, Requirements, Backlog Items, UI-Entwürfen, Prototypen, Code, Review-Formaten, Evaluation, Testing und Monitoring berücksichtigen.

BO-G 4

Den Nutzen und die Grenzen von AI im Usability-Kontext einschätzen, insbesondere bei bekannten Mustern, unbekanntem Nutzungskontext, Domänenwissen, Nutzerkontakt und Validierungsbedarf.

BO-G 5

Verantwortung für Usability, Datenschutz, Accessibility, Code-Qualität, AI-Einsatz und fachliche Entscheidungen im Softwareentwicklungsteam verorten und geeignete Prüf- und Governance-Maßnahmen ableiten.

BO-G 6

Erkenntnisse aus AI-gestützter Analyse, Evaluation, realer Nutzung, Nutzer- und Kundenfeedback sowie Supportdaten in konkrete Verbesserungen, Backlog Items und kontinuierliche Usability-Verbesserung überführen.

Kapitel 1

BO 1.1

Usability als Bestandteil professioneller Softwarequalität in Entwicklungsprozessen berücksichtigen und begründen.

BO 1.2
GenAI als niederschwelligen Einstieg in konkrete Usability-Arbeit nutzen, insbesondere in Teams ohne eigene UX-Spezialisten, ausreichendes Budget, Zeit oder methodisches Vorwissen.

BO 1.3
Aus dem Prinzip „AI ist stark bei bekannten Mustern, aber schwach bei unbekanntem Kontext“ ableiten, wo AI im Usability-Prozess sinnvoll unterstützt und wo menschliche Bewertung, Domänenwissen oder reale Nutzerbeobachtung notwendig bleiben.

BO 1.4
Grundlegende Regeln für verantwortungsvollen AI-Einsatz im Usability-Kontext anwenden, insbesondere menschliche Kontrolle, Datenschutz, Transparenz, Nachvollziehbarkeit und Governance.

Kapitel 2

BO 2.1
GenAI gezielt nutzen, um zentrale Usability-Grundlagen einschließlich menschlicher Wahrnehmung, mentaler Modelle, kognitiver Belastung, Verständlichkeit und Accessibility zu erlernen, zu wiederholen und auf eigene Entwicklungsartefakte anzuwenden.

BO 2.2
Ein gemeinsames Basisverständnis für Usability im Softwareentwicklungsteam aufbauen, indem zentrale Begriffe, Qualitätskriterien und typische Usability-Probleme mit Hilfe von AI erklärt, verglichen und auf eigene Entwicklungsartefakte bezogen werden.

BO 2.3
AI-Ausgaben kritisch prüfen und zwischen plausibel wirkenden Vorschlägen, brauchbaren Hypothesen und belastbaren Erkenntnissen unterscheiden.

BO 2.4
AI-Unterstützung bei bekannten Usability-Mustern nutzen und bei unbekanntem Kontext, impliziten Anforderungen und domänenspezifischen Fragestellungen gezielt überprüfen.

Kapitel 3

BO 3.1
Vorhandene Informationen wie Supporttickets, Kundenfeedback, Analytics, Interviews oder Fachwissen nutzen, um Usability-relevante Problemfelder zu identifizieren.

BO 3.2
GenAI einsetzen, um vorhandenes Nutzer- und Kundenfeedback, Supporthinweise, Beobachtungen und Daten zu strukturieren, zusammenzufassen und daraus überprüfbare Usability-Hypothesen abzuleiten.

BO 3.3
Aus AI-gestützt strukturierten Informationen konkrete Nutzerprobleme, Nutzungshypothesen und prüfbare Produktannahmen ableiten, die als Grundlage für Backlog Items, Prototypen oder Tests dienen.

BO 3.4
Erkennen, wann AI-gestützte Hypothesen nicht ausreichen und reale Nutzer, Domänenexperten oder zusätzliche Research-Aktivitäten einbezogen werden müssen.

Kapitel 4

BO 4.1
Usability-Anforderungen als Bestandteil professioneller Requirements-Arbeit in User Stories, Akzeptanzkriterien und Definition-of-Done-Kriterien integrieren.

BO 4.2
GenAI nutzen, um Backlog Items auf fehlende Usability-Aspekte, unklare Formulierungen, Happy-Path-Fixierung, fehlende Zustände und mögliche Edge Cases zu prüfen.

BO 4.3
Usability-relevante Kriterien in Definition of Ready und Definition of Done integrieren, damit Nutzerziel, Nutzungskontext, Zustände, Fehlerfälle, Accessibility-Basics und Validierungsbedarf überprüfbar werden.

BO 4.4
AI-generierte Anforderungen und Akzeptanzkriterien fachlich einordnen und entscheiden, welche Vorschläge im konkreten Produkt- und Nutzungskontext relevant sind.

Kapitel 5

BO 5.1
Usability-Aufgaben in der Planung sichtbar machen und grundlegende Verantwortlichkeiten für UX im Softwareentwicklungsteam klären.

BO 5.2

Einfache Usability-Aktivitäten wie Design Critiques, heuristische Prüfungen anhand definierter Kriterien, Formulartests, Accessibility-Basic-Checks und Systemzustandsprüfungen im Team organisatorisch einplanen.

BO 5.3

GenAI zur Vorbereitung, Strukturierung und Dokumentation von Usability-bezogenen Teamaktivitäten einsetzen, ohne Verantwortung und Entscheidungsbefugnis an AI zu delegieren.

BO 5.4

Den Einsatz von AI in Usability-relevanten Teamaktivitäten nachvollziehbar dokumentieren, insbesondere Zweck, verwendete Eingaben, übernommene Ergebnisse und menschliche Prüfung.

Kapitel 6

BO 6.1

AI-gestützte UI-Entwürfe, Wireframes, Prototypen und Frontend-Vorschläge anhand grundlegender Usability- und Human-Factors-Kriterien beurteilen, insbesondere Wahrnehmbarkeit, Verständlichkeit, kognitive Belastung, Konsistenz, Accessibility und erwartbares Verhalten.

BO 6.2

Typische Schwächen AI-generierter Entwürfe erkennen, insbesondere fehlende States, Edge Cases, Fehlerpfade, unklare Navigation, mangelnde Accessibility und zu starke Happy-Path-Orientierung.

BO 6.3

AI-generierte Prototypen als Kommunikations-, Lern- und Validierungsgrundlage nutzen, ohne sie vorschnell als fachlich richtige oder produktionsreife Lösung zu behandeln.

BO 6.4

AI-generierte UI-Varianten anhand von Nutzerziel, visueller Hierarchie, Verständlichkeit, Konsistenz, Accessibility-Basics und erwartbarem Verhalten vergleichen und begründet auswählen.

Kapitel 7

BO 7.1

AI-gestützte Coding-Werkzeuge für UI-nahe Entwicklungsaufgaben, Komponenten, Validierung, Dokumentation und Tests fachlich einordnen und gezielt einsetzen.

BO 7.2

Grundlegende Usability-Qualitätsaspekte in der Implementierung berücksichtigen, insbesondere

konsistente Komponenten, Zustände, semantisches HTML, Tastaturbedienbarkeit, Fokusführung, Labels, Fehlermeldungen und Accessibility-Basics.

BO 7.3
AI-generierte UI-Implementierungen auf ihre Übereinstimmung mit Usability-Requirements, Akzeptanzkriterien, Zuständen, Komponentenregeln und Tests prüfen.

BO 7.4
Risiken AI-generierter Implementierung erkennen und geeignete Prüfmaßnahmen wie Code Review, Tests, Security Review, Accessibility-Prüfung und fachliche Abnahme ableiten.

Kapitel 8

BO 8.1
Grundlegende Review- und Evaluationsmethoden für Usability wie heuristische Evaluation, Cognitive Walkthrough und Usability-Test in Entwicklungsprozessen einordnen und zweckmäßig einsetzen.

BO 8.2
GenAI zur Vorbereitung, Strukturierung und Dokumentation von Usability-Reviews und Usability-Tests nutzen, insbesondere für Testaufgaben, Beobachtungsnotizen, Findings und Berichtsentwürfe.

BO 8.3
Erkennen, warum AI-Simulationen und heuristische AI-Analysen keine Validierung mit realen Nutzern ersetzen und wann echte Tests notwendig sind.

BO 8.4
Usability-Probleme nach Schweregrad, Häufigkeit, Nutzerwirkung, Risiko, Umsetzungsaufwand und Business-Relevanz priorisieren und AI-Vorschläge dazu kritisch einordnen.

BO 8.5
Aus AI-generierten Usability-Vorschlägen testbare Annahmen und Fragestellungen ableiten, die durch heuristische Reviews, Usability-Tests oder Nutzungsdaten überprüft werden können.

Kapitel 9

BO 9.1
Usability nach dem Release anhand von Feedback, Supporttickets, Nutzungsdaten und einfachen UX-Metriken beobachten und Verbesserungspotenziale identifizieren.

BO 9.2

GenAI einsetzen, um Nutzer- und Kundenfeedback, Tickets, Metriken und UX-Debt zu strukturieren, Muster sichtbar zu machen und Hypothesen für Verbesserungen abzuleiten.

BO 9.3

Grenzen daten- und AI-gestützter Auswertung erkennen, insbesondere den Unterschied zwischen Muster, Korrelation, Ursache, Priorität und belastbarer Produktentscheidung.

BO 9.4

Erkenntnisse aus Feedback, Metriken, Supportdaten und AI-gestützter Analyse in priorisierte Backlog Items, UX-Debt-Maßnahmen oder neue Discovery-Fragen überführen.

0.10 Verwendete Kürzel und Akronyme

Abkürzungsverzeichnis

Abkürzung	Voller Name	Erklärung
AI	Artificial Intelligence	Englische Bezeichnung für künstliche Intelligenz. Im Syllabus wird AI als Oberbegriff für Systeme verwendet, die Aufgaben wie Textgenerierung, Analyse, Mustererkennung, Code-Unterstützung oder Entscheidungsunterstützung leisten.
API	Application Programming Interface	Programmierschnittstelle, über die Softwarekomponenten oder Systeme miteinander kommunizieren. Im Usability-Kontext relevant, wenn Frontend, Daten, Validierung und Systemreaktionen zusammenspielen.
ARIA	Accessible Rich Internet Applications	W3C-Spezifikation zur Ergänzung von semantischen Informationen für assistive Technologien. ARIA sollte nur eingesetzt werden, wenn native HTML-Elemente nicht ausreichen.
BO	Business Outcome	Ergebnisorientierte Beschreibung dessen, was Teilnehmende nach der Schulung in der beruflichen Praxis leisten können sollen.
CI/CD	Continuous Integration / Continuous Delivery	Entwicklungs- und Auslieferungspraktiken, bei denen Code regelmäßig integriert, getestet und bereitgestellt wird. Relevant für Qualitätssicherung, Tests und kontinuierliche Verbesserung.

Abkürzung	Voller Name	Erklärung
CSS	Cascading Style Sheets	Stylesheet-Sprache zur Gestaltung von Weboberflächen, etwa Layout, Farben, Typografie, Abstände und responsive Darstellung.
DoD	Definition of Done	Gemeinsame Teamvereinbarung, wann ein Backlog Item oder Inkrement als fertig gilt. Im Syllabus wird betont, dass Usability-, Accessibility-, Test- und Qualitätskriterien darin sichtbar werden können.
DoR	Definition of Ready	Gemeinsame Teamvereinbarung, wann ein Backlog Item ausreichend vorbereitet ist, um in die Umsetzung beziehungsweise in einen Sprint aufgenommen zu werden. Usability-relevant sind etwa Nutzerziel, Kontext, Zustände, Fehlerfälle und offene Annahmen.
DSGVO	Datenschutz-Grundverordnung	Europäische Datenschutzverordnung zum Schutz personenbezogener Daten. Relevant, wenn AI mit Nutzerfeedback, Research-Daten, Supporttickets oder personenbezogenen Informationen eingesetzt wird.
EAA	European Accessibility Act	Europäische Richtlinie zu Barrierefreiheitsanforderungen für bestimmte Produkte und Dienstleistungen. Im Syllabus relevant als rechtlicher Rahmen für digitale Barrierefreiheit.
EN	European Norm	Europäische Norm. Im Syllabus besonders relevant im Zusammenhang mit EN 301 549 für Barrierefreiheitsanforderungen an ICT-Produkte und -Dienstleistungen.
EU	Europäische Union	Politischer und rechtlicher Rahmen, insbesondere relevant für Datenschutz, AI-Regulierung und digitale Barrierefreiheit.
FL	Foundation Level	Grundlagenstufe der Zertifizierung. Der Foundation Level beschreibt ein grundlegendes, praxisnahes Qualifikationsniveau, bei dem zentrale Begriffe, Konzepte und typische Anwendungen verstanden und auf einfache berufliche Situationen angewendet werden können.
GenAI	Generative Artificial Intelligence	Generative künstliche Intelligenz. Systeme, die neue Inhalte wie Texte, Bilder, UI-Entwürfe, Code, Zusammenfassungen oder Testfälle erzeugen können.

Abkürzung	Voller Name	Erklärung
GDPR	General Data Protection Regulation	Englische Bezeichnung für die Datenschutz-Grundverordnung. Siehe DSGVO.
HCI	Human-Computer Interaction	Forschungs- und Praxisfeld zur Interaktion zwischen Menschen und Computersystemen. Im Syllabus wird die Struktur jedoch nicht klassisch entlang HCI aufgebaut, sondern entlang eines typischen agilen Software-entwicklungsprozesses.
HEART	Happiness, Engagement, Adoption, Retention, Task Success	UX-Metrik-Framework von Google zur strukturierten Betrachtung von Nutzerzufriedenheit, Engagement, Adoption, Retention und Zielerreichung.
HTML	HyperText Markup Language	Auszeichnungssprache für Webinhalte. Semantisches HTML ist eine Grundlage für Accessibility, robuste Struktur und assistive Technologien.
ICT	Information and Communication Technology	Informations- und Kommunikationstechnologie. Der Begriff wird etwa in Standards zur Barrierefreiheit digitaler Produkte und Dienstleistungen verwendet.
IDE	Integrated Development Environment	Integrierte Entwicklungsumgebung, z. B. für Codebearbeitung, Debugging, Tests und AI-gestützte Entwicklungsunterstützung.
IEC	International Electrotechnical Commission	Internationale Normungsorganisation für Elektrotechnik und verwandte Technologien. Relevant in Kombination mit ISO, etwa bei ISO/IEC-Standards.
ISO	International Organization for Standardization	Internationale Normungsorganisation. Im Syllabus relevant für Usability, Human-Centered Design und Softwarequalität, z. B. ISO 9241 und ISO/IEC 25010.
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission	Gemeinsame Normen von ISO und IEC, besonders relevant für Software- und Systemqualität.
K1	Knowledge Level 1 – Erinnern	Lernzielstufe nach der revidierten Bloom’schen Taxonomie. Teilnehmende können Begriffe, Fakten oder Grundprinzipien wiedergeben.

Abkürzung	Voller Name	Erklärung
K2	Knowledge Level 2 – Verstehen	Lernzielstufe nach der revidierten Bloom'schen Taxonomie. Teilnehmende können Zusammenhänge erklären, Unterschiede beschreiben und Konzepte einordnen.
K3	Knowledge Level 3 – Anwenden	Lernzielstufe nach der revidierten Bloom'schen Taxonomie. Teilnehmende können Wissen auf typische Aufgaben, Artefakte oder Situationen anwenden.
K4	Knowledge Level 4 – Analysieren	Höhere Lernzielstufe nach Bloom. Wird im Foundation-Level-Syllabus nicht als eigene Prüfungsstufe verwendet.
K5	Knowledge Level 5 – Evaluieren / Bewerten	Höhere Lernzielstufe nach Bloom. Wird im Foundation-Level-Syllabus nicht als eigene Prüfungsstufe verwendet.
K6	Knowledge Level 6 – Erstellen / kreatives Gestalten	Höhere Lernzielstufe nach Bloom. Wird im Foundation-Level-Syllabus nicht als eigene Prüfungsstufe verwendet.
KPI	Key Performance Indicator	Kennzahl zur Bewertung von Leistung oder Zielerreichung. Im Usability-Kontext relevant, wenn UX-Metriken mit Produkt- oder Business-Zielen verbunden werden.
LO	Learning Objective	Lernziel. Beschreibt konkret, was Teilnehmende nach einem Kapitel wissen, verstehen oder anwenden können sollen.
LLM	Large Language Model	Großes Sprachmodell, das auf umfangreichen Textdaten trainiert wurde und Texte analysieren, generieren, zusammenfassen oder transformieren kann. Grundlage vieler GenAI-Systeme.
MVP	Minimum Viable Product	Minimal funktionsfähige Produktversion, mit der zentrale Annahmen geprüft oder ein erster Nutzen bereitgestellt werden kann.
NIST	National Institute of Standards and Technology	US-amerikanische Institution für Standards und Technologie. Im Syllabus relevant durch das AI Risk Management Framework.
OECD	Organisation for Economic Co-operation and Development	Internationale Organisation für wirtschaftliche Zusammenarbeit und Entwicklung. Relevant durch AI Principles und Governance-Orientierung.
PO	Product Owner	Rolle in Scrum und in vielen agilen Produktteams, verantwortlich für Product Backlog, Priorisierung und

Abkürzung	Voller Name	Erklärung
		Produktwert. Im Usability-Kontext wichtig für Nutzerziel, Akzeptanzkriterien und Priorisierung von Usability-Problemen.
RMF	Risk Management Framework	Rahmenwerk für Risikomanagement. Im Syllabus besonders relevant als Teil des NIST AI Risk Management Framework.
Scrum	Kein Akronym	Rahmenwerk für agile Produktentwicklung. Der Begriff ist keine Abkürzung, wird aber häufig wie ein Fachkürzel verwendet.
SQuaRE	Systems and software Quality Requirements and Evaluation	ISO/IEC-Normenreihe zur Qualität von Systemen und Software. Relevant im Zusammenhang mit ISO/IEC 25010, Softwareproduktqualität, Interaction Capability und Quality in Use.
SUS	System Usability Scale	Standardisierter Fragebogen zur Messung wahrgenommener Usability. Relevant für Usability-Evaluation und UX-Metriken.
UI	User Interface	Benutzeroberfläche eines Systems, also sichtbare und bedienbare Elemente wie Screens, Formulare, Navigation, Controls und Interaktionen.
UNESCO	United Nations Educational, Scientific and Cultural Organization	Organisation der Vereinten Nationen für Bildung, Wissenschaft und Kultur. Relevant durch Empfehlungen zur Ethik künstlicher Intelligenz.
UX	User Experience	Gesamte Nutzungserfahrung eines Produkts oder Systems. UX umfasst Erwartungen, Wahrnehmungen und Reaktionen vor, während und nach der Nutzung. Im Syllabus wird UX als breiterer Rahmen verstanden, der Software-Usability einschließt, aber über diese hinausgeht.
UXQCC	User Experience Quality Certification Center	Internationale Zertifizierungsorganisation für User Experience und Usability. UXQCC entwickelt und pflegt Zertifizierungen für UX- und Usability-Fachkompetenz und orientiert sich dabei an international anerkannten Standards, insbesondere der ISO-9241-Reihe
W3C	World Wide Web Consortium	Internationale Organisation zur Entwicklung von Webstandards. Relevant für HTML, ARIA, WCAG und Web Accessibility.

Abkürzung	Voller Name	Erklärung
WAI	Web Accessibility Initiative	Initiative des W3C zur Förderung digitaler Barrierefreiheit. Entwickelt Ressourcen und Standards wie WCAG-Unterlagen.
WCAG	Web Content Accessibility Guidelines	Richtlinien des W3C für barrierefreie Webinhalte. Im Syllabus relevant für Accessibility-Basics und digitale Barrierefreiheit.

1 Usability mit AI: Warum das Thema relevant ist

Unterrichtszeit: ca. 90 Minuten

1.1 Leitgedanke des Kapitels

Dieses Kapitel führt in die Grundlogik des Syllabus ein. Software-Usability und User Experience werden als Bestandteile professioneller Softwarequalität verstanden. Für Softwareentwicklungsteams ist relevant, dass diese Qualität bereits bei Problemverständnis, Anforderungen, User Stories, Akzeptanzkriterien, Zuständen, Fehlermeldungen, Komponenten, Tests, Reviews und Monitoring entsteht.

GenAI kann einen niederschwelligen Einstieg in Usability-Arbeit ermöglichen, indem sie Informationen strukturiert, Begriffe erklärt, Varianten erzeugt, Artefakte prüft oder Auswertungen vorbereitet. AI-Ergebnisse müssen jedoch durch Menschen geprüft, mit Produkt- und Nutzungskontext verbunden und in den Entwicklungsprozess eingeordnet werden. Ein zentrales Grundprinzip lautet:

AI ist stark bei bekannten Mustern, aber schwach bei unbekanntem Kontext.

1.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 1.1 bis BO 1.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

1.3 Learning Objectives

LO	Learning Objective	K
LO 1.1	Die Begriffe Software-Usability, User Experience und UX-Qualität im Kontext der Softwareentwicklung wiedergeben können.	K1
LO 1.2	Erklären können, warum Usability Bestandteil professioneller Softwarequalität ist.	K2
LO 1.3	Erklären können, warum agile Prozesse gute UX nicht automatisch sicherstellen.	K2
LO 1.4	Beschreiben können, wie GenAI einen niederschwelligen Einstieg in Usability-Arbeit ermöglicht.	K2

LO	Learning Objective	K
LO 1.5	Die Grundregel „AI ist stark bei bekannten Mustern, aber schwach bei unbekanntem Kontext“ erläutern können.	K2
LO 1.6	Typische Einsatzfelder von GenAI entlang typischer agiler Entwicklungsaktivitäten benennen können.	K1
LO 1.7	AI-Einsatzmöglichkeiten in agilen Entwicklungsprozessen anhand ihres Nutzens und ihrer Grenzen einordnen können.	K3
LO 1.8	Grundlegende Regeln für verantwortungsvollen AI-Einsatz im Usability-Kontext erklären können.	K2

1.4 Wichtige Begriffe

Software-Usability, User Experience, Usability, UX-Qualität, Softwarequalität, Interaction Capability, Quality in Use, ISO/IEC 25010, agile Entwicklung, Scrum, Backlog, Discovery, Delivery, GenAI, AI-Output, Mustererkennung, Halluzination, Nutzungskontext, menschliche Kontrolle, Datenschutz, Transparenz, AI Governance.

1.5 Software-Usability, UX und Interaction Capability als Bestandteile von Softwarequalität

Software-Usability und User Experience sind Bestandteile der Qualität interaktiver Software. Software-Usability beschreibt, in welchem Ausmaß bestimmte Nutzerinnen und Nutzer ein System in einem bestimmten Nutzungskontext nutzen können, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.

User Experience, kurz UX, ist der breitere Begriff. UX umfasst neben der eigentlichen Nutzung auch Erwartungen, Wahrnehmungen und Reaktionen vor, während und nach der Nutzung eines Produkts oder Systems. UX schließt Software-Usability ein, geht aber darüber hinaus.

In der Softwarequalität wird die produktbezogene Qualität der Mensch-System-Interaktion unter anderem über den Begriff Interaction Capability beschrieben. Dazu gehören Eigenschaften wie Erlernbarkeit, Bedienbarkeit, Fehlerschutz, Unterstützung der Nutzer, Selbsterklärungsfähigkeit und Zugänglichkeit. Für Softwareentwicklungsteams ist dies relevant, weil solche Eigenschaften in Anforderungen, Akzeptanzkriterien, Reviews, Tests, Definition-of-Done-Kriterien und kontinuierlicher Verbesserung berücksichtigt werden können.

Software ist daher nicht schon dann gut, wenn sie technisch funktioniert. Sie muss auch so nutzbar sein, dass Menschen ihre Aufgaben verstehen, relevante Informationen finden, Entscheidungen treffen und Fehler vermeiden oder beheben können. Jede Funktion hat neben einer technischen auch eine Nutzungsperspektive.

UX ist nicht mit visueller Gestaltung gleichzusetzen. Eine visuell ansprechende Oberfläche kann schwer verständlich oder ineffizient sein. Umgekehrt kann eine einfache Oberfläche eine hohe UX-Qualität haben, wenn sie Nutzerziele klar unterstützt und die Interaktion nachvollziehbar macht.

Für agile Softwareentwicklungsteams ist wichtig, dass UX-Qualität nicht erst am Ende eines Projekts entsteht. Sie zeigt sich bereits in Backlog Items, Akzeptanzkriterien, Prototypen, Komponenten, Tests, Reviews und im Umgang mit Feedback nach dem Release.

1.6 Usability in Entwicklungsprozessen

Agile Prozesse helfen Softwareentwicklungsteams, Arbeit iterativ zu planen, umzusetzen und zu überprüfen. Scrum wird in diesem Syllabus beschreibend als verbreiteter agiler Referenzrahmen verwendet. Weder Scrum noch andere agile Vorgehensweisen stellen jedoch automatisch sicher, dass reale Nutzerbedürfnisse verstanden, Usability-Anforderungen formuliert oder Nutzungskontexte ausreichend berücksichtigt werden.

Usability-relevante Entscheidungen entstehen häufig bereits, bevor ein Interface sichtbar wird: wenn ein Problem beschrieben, eine Nutzergruppe angenommen, eine User Story formuliert, ein Akzeptanzkriterium festgelegt oder ein technischer Lösungsweg gewählt wird. Auch während der Umsetzung entstehen Usability-Wirkungen, etwa durch Benennungen, Zustände, Fehlermeldungen, Validierungen, Komponenten, Navigationslogik, Fokusführung oder Ladeverhalten.

Ein Product Backlog kann vollständig gepflegt sein und dennoch wesentliche Usability-Aspekte auslassen. Eine User Story kann funktional korrekt beschrieben sein, aber keine Hinweise auf Verständlichkeit, Fehlerfälle, Zustände, Accessibility oder Nutzungskontext enthalten. Ein Sprint Review kann zeigen, dass eine Funktion umgesetzt wurde, ohne zu prüfen, ob Nutzerinnen und Nutzer diese Funktion tatsächlich verstehen oder erfolgreich verwenden können.

UX kann in agilen Entwicklungsprozessen unter anderem in Discovery-Aktivitäten, Backlog Refinements, Akzeptanzkriterien, Definition of Ready, Definition of Done, Design Critiques, heuristischen Reviews, Usability-Tests und Monitoring nach dem Release sichtbar gemacht werden.

1.7 GenAI als Unterstützung entlang agiler Aktivitäten

GenAI kann Usability-Arbeit in agilen Softwareentwicklungsteams früher zugänglich und leichter anschlussfähig machen, insbesondere wenn keine dauerhaft verfügbare UX-Spezialrolle vorhanden ist oder wenig methodische UX-Erfahrung besteht.

In der Discovery kann AI vorhandenes Feedback strukturieren, Supporttickets zusammenfassen, Interviewnotizen ordnen oder erste Hypothesen formulieren. Im Backlog Refinement kann AI User Stories prüfen, Akzeptanzkriterien ergänzen, unklare Formulierungen markieren, fehlende Zustände aufdecken oder eine zu starke Happy-Path-Orientierung reduzieren. In der agilen Planung kann AI helfen, Usability-Aufgaben zu strukturieren, Review-Fragen vorzubereiten oder Checklisten zu erstellen.

Im Design und Prototyping kann AI UI-Varianten, Navigationsmodelle, Wireframes oder klickbare Prototypen erzeugen. In der Umsetzung kann AI bei UI-Komponenten, Code, Tests, Dokumentation und Accessibility-Hinweisen unterstützen. In Review, Evaluation und Testing kann AI heuristische Bewertungen vorbereiten, Testaufgaben formulieren, Beobachtungsnotizen strukturieren oder Findings zusammenfassen. Nach dem Release kann AI Feedback, Tickets oder Metriken auswerten und Hinweise auf UX-Debt liefern.

Diese Einsatzmöglichkeiten ersetzen keine professionelle Usability-Arbeit, können aber eine erste, pragmatische Auseinandersetzung mit UX ermöglichen. Sie sind nach Nutzen und Grenzen einzuordnen, insbesondere danach, ob AI vor allem Strukturierung, Variantenbildung, Formulierung, technische Unterstützung oder erste Analyse leistet und welche menschliche Prüfung erforderlich bleibt.

1.8 Bekannte Muster, unbekannter Kontext und AI-Output

Generative AI denkt nicht im menschlichen Sinn. Sie besitzt keine eigene Wahrnehmung, keine eigene Nutzungserfahrung und kein echtes Verständnis für konkrete Nutzerinnen und Nutzer. Sie erzeugt Ausgaben auf Basis gelernter Muster und Wahrscheinlichkeiten.

AI kann bei Aufgaben hilfreich sein, die häufig vorkommen, gut dokumentiert sind und auf bekannten Mustern beruhen. Dazu zählen zum Beispiel typische Login-Flows, Standardformulare, einfache Navigationsmuster, Fehlermeldungen, Button-Texte, Onboarding-Abläufe, einfache Accessibility-Hinweise oder heuristische Erstbewertungen.

AI wird unsicherer, sobald der konkrete Nutzungskontext entscheidend ist. Dazu zählen reale Nutzerziele, spezielle Fachdomänen, organisatorische Rahmenbedingungen, implizite Anforderungen, seltene Ausnahmefälle, rechtliche Fragen, sicherheitskritische Anwendungen oder

Priorisierungsentscheidungen. Gute UX hängt davon ab, wer ein System nutzt, in welcher Situation, mit welchem Vorwissen, unter welchen Einschränkungen und mit welchem Ziel.

AI-Ausgaben können sprachlich oder visuell überzeugend wirken, obwohl sie fachlich unzutreffend, unvollständig oder nicht zum konkreten Kontext passend sind. Plausibilität ist daher kein Qualitätsnachweis. AI-Output ist im Usability-Kontext zunächst als Hypothese, Variante oder Strukturierungsvorschlag zu behandeln.

1.9 Verantwortungsvoller AI-Einsatz im Usability-Kontext

Der Einsatz von AI im Usability-Kontext erfordert menschliche Kontrolle. AI-Ergebnisse sind zu prüfen, einzuordnen und bei Bedarf zu verwerfen oder anzupassen. Dies gilt für Texte, User Stories, Akzeptanzkriterien, UI-Entwürfe, Prototypen, Code, Testfälle und Auswertungen.

AI ist als Entwurfspartner zu verstehen, nicht als Entscheider. Sie kann Varianten erzeugen, Hypothesen formulieren, Strukturen vorschlagen und Alternativen aufzeigen. Entscheidungen über Produkt, Nutzer, Qualität, Risiko und Umsetzung bleiben beim Team beziehungsweise bei den verantwortlichen Personen und Organisationen.

Datenschutz und Vertraulichkeit sind vor der Nutzung eines AI-Werkzeugs zu berücksichtigen. Personenbezogene Daten, vertrauliche Projektdaten oder sensible Research-Daten dürfen nicht unreflektiert in externe AI-Dienste eingegeben werden. Wo möglich, sind Daten zu anonymisieren, zu pseudonymisieren oder durch synthetische Beispiele zu ersetzen.

Transparenz und Nachvollziehbarkeit sind erforderlich, wenn AI-Ausgaben in Requirements, Designs, Code, Tests oder Produktentscheidungen einfließen. Es sollte nachvollziehbar bleiben, wo AI eingesetzt wurde, mit welchem Ziel, welche Ergebnisse übernommen wurden und welche menschliche Prüfung erfolgt ist.

AI Governance schafft den organisatorischen Rahmen für verantwortungsvollen AI-Einsatz. Dazu zählen Regeln zu erlaubten Werkzeugen, Datenschutz, Dokumentation, menschlicher Prüfung, Qualitätssicherung und Verantwortlichkeiten.

AI kann Usability-Probleme vermuten, aber reale Nutzung nicht ersetzen. Wo zentrale Annahmen unsicher sind, müssen Nutzerfeedback, Domänenexpertise, Usability-Tests oder reale Nutzungsdaten herangezogen werden.

2 Mit AI Usability-Basiswissen aufbauen

Unterrichtszeit: ca. 120 Minuten

2.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt GenAI nicht primär als Werkzeug zur Erstellung von Usability-Artefakten, sondern als Lern- und Reflexionswerkzeug für Softwareentwicklungsteams. Teams benötigen ein gemeinsames Basisverständnis für Usability, um AI-generierte Vorschläge, UI-Entwürfe, Requirements, Fehlermeldungen, Prototypen oder Code fachlich beurteilen zu können.

Basiswissen zu Usability umfasst zentrale Begriffe wie Nutzerziel, Nutzungskontext, mentales Modell, Erwartungskonformität, kognitive Last, visuelle Hierarchie und Accessibility. Dazu gehören auch grundlegende menschliche Faktoren wie Wahrnehmung, Aufmerksamkeit, Gedächtnis, motorische Fähigkeiten, situative Einschränkungen und kognitive Belastung.

AI kann diese Grundlagen erklären, mit Beispielen verbinden und auf eigene Entwicklungsartefakte anwenden helfen. AI-Ausgaben sind jedoch nicht als gesicherte Erkenntnisse zu behandeln. Sie müssen anhand grundlegender Usability-Kriterien, des Nutzungskontexts und menschlicher Prüfung kritisch eingeordnet werden.

2.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 2.1 bis BO 2.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

2.3 Learning Objectives

LO	Learning Objective	K
LO 2.1	Zentrale Usability-Basisbegriffe wie Nutzerziel, Nutzungskontext, mentales Modell, Erwartungskonformität, kognitive Last, visuelle Hierarchie und Accessibility wiedergeben können.	K1
LO 2.2	Grundlegende menschliche Faktoren für UX benennen können, insbesondere Wahrnehmung, Aufmerksamkeit, Gedächtnis, kognitive Belastung, motorische Fähigkeiten und situative Einschränkungen.	K1

LO	Learning Objective	K
LO 2.3	Erklären können, warum menschliche Wahrnehmung selektiv, begrenzt und kontextabhängig ist.	K2
LO 2.4	Erklären können, warum mentale Modelle, Erwartungskonformität und kognitive Last für die Beurteilung von Benutzeroberflächen zentral sind.	K2
LO 2.5	Erklären können, warum Softwareentwicklungsteams ein gemeinsames Basisverständnis für Usability benötigen, um AI-Ergebnisse sinnvoll beurteilen zu können.	K2
LO 2.6	Beschreiben können, wie GenAI beim Lernen und Wiederholen von Usability-Grundlagen unterstützen kann.	K2
LO 2.7	Eigene Entwicklungsartefakte wie Screens, User Stories, Fehlermeldungen, Prototypen oder Codebeispiele mit AI-Unterstützung zur Usability-Reflexion verwenden können.	K3
LO 2.8	Prompting-Techniken für Usability-Lernen beschreiben können, insbesondere Erklären, Vergleichen, Gegenbeispiele erzeugen, Annahmen prüfen und Verbesserungsvorschläge begründen lassen.	K2
LO 2.9	AI-Ausgaben als Hypothesen und nicht als gesicherte Erkenntnisse einordnen können.	K2
LO 2.10	Plausible, aber potenziell falsche oder unvollständige AI-Antworten erkennen und kritisch hinterfragen können.	K3
LO 2.11	AI-gestützte Usability-Erklärungen und Verbesserungsvorschläge mit grundlegenden Kriterien menschlicher Wahrnehmung, Verständlichkeit, Erwartungskonformität und Accessibility abgleichen können.	K3

2.4 Wichtige Begriffe

Nutzer, Nutzerziel, Aufgabe, Nutzungskontext, Software-Usability, Usability, User Experience, UX-Qualität, Basiswissen zu Usability, Human-Centered Design, Wahrnehmung, Aufmerksamkeit, Gedächtnis, kognitive Last, mentales Modell, Erwartungskonformität, Verständlichkeit, visuelle Hierarchie, motorische Fähigkeiten, situative Einschränkungen, Accessibility, AI-Output, Prompting, Hypothese, Plausibilität, Halluzination, Scheinplausibilität, Mustererkennung, unbekannter Kontext, kritische Prüfung, menschliche Kontrolle

Hinweis zu den wichtigen Begriffen

Die in diesem Abschnitt genannten Begriffe bilden die zentrale Begriffsbasis für Kapitel 2. Sie benennen jene Konzepte, die für das Verständnis von Usability-Basiswissen, menschlichen Faktoren und der kritischen Beurteilung von AI-Output besonders wichtig sind.

Das Kurzglossar in Anhang 1 fasst die wichtigsten Begriffe des gesamten Syllabus knapp zusammen und dient als Orientierung für ein gemeinsames Grundverständnis von UX, agiler Entwicklung, Scrum und GenAI im Kontext dieses Lehrplans.

Ergänzend steht ein separates erweitertes Glossar-Dokument zur Verfügung. Dieses enthält zusätzliche Begriffe, Übersetzungen, Synonyme, Beispiele, Abgrenzungen sowie normative und fachliche Bezüge. Es dient als Lern- und Nachschlagehilfe, aber stellt keinen eigenständigen zusätzlichen Prüfungsstoff dar. Prüfungsrelevant sind Begriffe nur insoweit, wie sie in den Kapiteln, Business Outcomes und Learning Objectives dieses Syllabus fachlich behandelt werden.

2.5 Basiswissen zu Usability als Voraussetzung für AI-gestützte Arbeit

Softwareentwicklungsteams benötigen ein gemeinsames Basisverständnis für Usability, um AI-Ergebnisse sinnvoll beurteilen zu können. Ohne gemeinsame Begriffe bleiben Aussagen zur Nutzung oft ungenau, etwa wenn ein Interface lediglich als „nicht intuitiv“, „unübersichtlich“ oder „nicht ansprechend“ beschrieben wird. Für professionelle Produktentwicklung müssen solche Eindrücke in überprüfbare Qualitätsfragen übersetzt werden.

Zentrale Begriffe wie Nutzerziel, Aufgabe, Nutzungskontext, Software-Usability, User Experience, mentales Modell, Erwartungskonformität, kognitive Last, visuelle Hierarchie und Accessibility helfen dabei, Probleme der Nutzung fachlich zu beschreiben. Sie ermöglichen es, AI-generierte Vorschläge nicht nur nach Geschmack oder Plausibilität, sondern anhand nachvollziehbarer Kriterien zu prüfen.

GenAI kann sehr schnell Texte, Screens, Requirements, Akzeptanzkriterien, Fehlermeldungen, Prototypen oder Codevorschläge erzeugen. Dadurch verschiebt sich der Engpass von der Erzeugung zur Beurteilung. Basiswissen zu Usability wird damit zu einer Prüffähigkeit: Teams müssen beurteilen können, ob AI-Ausgaben verständlich, erwartungskonform, konsistent, zugänglich und für den Nutzungskontext plausibel sind.

Diese Fähigkeit ist eine Teamkompetenz. Product Owner, Entwicklerinnen und Entwickler, Testerinnen und Tester, Requirements Engineers, Usability- und UI-Rollen sowie Personen mit Prozessverantwortung beeinflussen gemeinsam Anforderungen, Umsetzung, Prüfung und Priorisierung. Ein gemeinsames Usability-Basisverständnis unterstützt die Zusammenarbeit zwischen diesen Rollen.

2.6 Menschliche Faktoren für UX

Menschliche Wahrnehmung ist selektiv, begrenzt und kontextabhängig. Menschen nehmen nicht alle Informationen vollständig und objektiv wahr, sondern richten Aufmerksamkeit auf das, was für ihre aktuelle Aufgabe, Erwartung oder Situation relevant erscheint. Für UI-Design bedeutet dies, dass Sichtbarkeit allein nicht ausreicht. Ein Element muss im richtigen Moment, an der richtigen Stelle, mit ausreichender Klarheit und im passenden Zusammenhang wahrnehmbar sein.

Aufmerksamkeit und Gedächtnis sind begrenzt. Benutzeroberflächen, die zu viele Optionen, unklare Prioritäten, widersprüchliche Signale oder unnötige Informationen enthalten, erhöhen die kognitive Belastung. Dies ist besonders relevant bei Formularen, komplexen Dashboards, mehrstufigen Prozessen, Fehlermeldungen, Sicherheitsabfragen und fachlich anspruchsvollen Systemen.

Mentale Modelle beschreiben, wie Nutzerinnen und Nutzer glauben, dass ein System funktioniert oder funktionieren sollte. Sie entstehen aus bisherigen Erfahrungen, bekannten Mustern, Fachwissen und Nutzungskontext. Erwartungskonformität bedeutet, dass ein System sich so verhält, wie Nutzerinnen und Nutzer es aufgrund ihrer Erfahrung und Aufgabe erwarten. Wenn ein Button, eine Navigation oder ein Prozess anders reagiert als erwartet, können Fehler, Unsicherheit oder Vertrauensverlust entstehen.

Nutzerinnen und Nutzer unterscheiden sich in visuellen, motorischen, kognitiven, sprachlichen und technischen Fähigkeiten. Zusätzlich können situative Einschränkungen auftreten, etwa schlechte Lichtverhältnisse, mobile Nutzung, Zeitdruck, Stress, Ablenkung, einhändige Bedienung oder laute Umgebung.

Accessibility ist ebenfalls ein grundlegender Bestandteil guter UX. AI kann Hinweise auf Accessibility-Probleme geben, aber keine vollständige Accessibility-Prüfung oder Konformität garantieren.

2.7 GenAI als Lernassistent für Usability-Grundlagen

GenAI kann Teams beim Lernen, Wiederholen und Strukturieren von Usability-Grundlagen unterstützen. Sie kann Begriffe erklären, Unterschiede zwischen Konzepten herausarbeiten, Beispiele erzeugen, typische Fehler beschreiben oder Usability-Kriterien auf Artefakte aus dem Entwicklungsalltag beziehen.

Besonders relevant ist der Einsatz eigener Entwicklungsartefakte. Screens, User Stories, Fehlermeldungen, Prototypen, Komponenten, Codebeispiele, Testaufgaben oder Nutzerfeedback können mit AI-Unterstützung zur Usability-Reflexion verwendet werden. Ein Team kann sich beispielsweise erklären lassen, warum eine Fehlermeldung unverständlich wirkt, warum ein Formular

kognitiv belastend sein könnte oder welche Annahmen in einer User Story über Nutzerziele enthalten sind.

Teilnehmende sollen eigene Entwicklungsartefakte mit AI-Unterstützung analysieren, die erzeugten Hinweise anhand grundlegender Usability-Kriterien prüfen und daraus Fragen für die weitere fachliche Klärung ableiten können.

AI-gestütztes Lernen ist jedoch nicht mit gesichertem Fachwissen gleichzusetzen. AI-Erklärungen können unvollständig, zu allgemein oder falsch sein. Teilnehmende sollen AI daher als Lernhilfe und Reflexionswerkzeug einsetzen, nicht als Autorität. Der Nutzen liegt darin, dass AI Fragen, Varianten, Begründungen und Gegenbeispiele erzeugen kann, die fachlich geprüft und im Team diskutiert werden.

2.8 Prompting als Lern- und Reflexionstechnik

Prompting kann im Usability-Kontext als Lern- und Reflexionstechnik verwendet werden. Dabei geht es nicht um das Auswendiglernen einzelner Prompt-Formulierungen, sondern um die Fähigkeit, AI gezielt zur Erklärung, zum Vergleich, zur Prüfung von Annahmen und zur Begründung von Verbesserungsvorschlägen einzusetzen.

Prompts zur Erklärung können genutzt werden, um Usability-Begriffe in Bezug auf Softwareentwicklung zu klären. Dabei sollte nicht nur nach Definitionen gefragt werden, sondern nach Bedeutung, typischen Beispielen, Risiken und Anwendung auf konkrete Artefakte. Ein Begriff wie kognitive Last kann etwa im Zusammenhang mit Formularen, Dashboards oder mehrstufigen Prozessen betrachtet werden.

Prompts zum Vergleichen und Begründen können helfen, Varianten von Button-Texten, Fehlermeldungen, Formularstrukturen, Onboarding-Schritten oder Navigationslösungen zu diskutieren. Dabei sollen Kriterien wie Verständlichkeit, Nutzerziel, Erwartungskonformität, kognitive Belastung, Accessibility, Fehlervermeidung und Konsistenz berücksichtigt werden.

Prompts für Gegenbeispiele und kritische Prüfung unterstützen die Reflexion eigener Annahmen. Teams können AI auffordern, mögliche Probleme einer Lösung zu benennen, implizite Annahmen offenzulegen oder zu beschreiben, in welchen Nutzungskontexten ein Vorschlag ungeeignet sein könnte. Die Bewertung der Ergebnisse bleibt Aufgabe des Teams.

2.9 AI-Output kritisch beurteilen

AI-Ausgaben können vollständig, professionell und überzeugend wirken. Im Usability-Kontext kann dies dazu führen, dass Vorschläge zu schnell akzeptiert werden. Eine klare Formulierung oder ein visuell

ansprechender Entwurf bedeutet jedoch nicht, dass der Vorschlag fachlich korrekt, vollständig oder für den Nutzungskontext geeignet ist.

Teilnehmende sollen zwischen Plausibilität, Hypothese, Vorschlag und belastbarer Erkenntnis unterscheiden können. Eine plausible AI-Antwort kann ein nützlicher Ausgangspunkt sein, ist aber keine validierte Erkenntnis. Belastbarkeit entsteht erst durch geeignete Prüfung, etwa durch Domänenwissen, Abgleich mit Anforderungen, Nutzerfeedback, Evaluation, Usability-Tests oder reale Nutzungsdaten.

Eine häufige Schwäche von AI besteht darin, Kontextlücken mit plausiblen Annahmen zu füllen. Wenn keine Informationen über Nutzergruppe, Aufgabe, Domäne, Risiken oder Rahmenbedingungen vorhanden sind, erzeugt AI dennoch eine Antwort. Diese Antwort kann sinnvoll wirken, beruht aber möglicherweise auf generischen Mustern.

Teilnehmende sollen deshalb Kontextlücken möglichst früh erkennen. Wichtige Prüffragen sind, welche Informationen fehlen, welche Nutzergruppe implizit unterstellt wird, welche fachlichen Regeln relevant sind, welche Nutzungssituationen beschrieben werden müssen und welche Risiken sonst in frühen Anforderungen, UI-Entwürfen oder Prototypen unsichtbar mitgeführt werden.

2.10 Bekannte Muster und unbekannter Kontext

AI ist besonders hilfreich bei bekannten, häufig vorkommenden und gut dokumentierten Usability-Mustern. Dazu gehören Standardformulare, Login-Flows, Suchfunktionen, Fehlermeldungen, Call-to-Action-Texte, einfache Dashboards, Onboarding-Schritte oder grundlegende UI-Patterns. In solchen Bereichen kann AI typische Varianten vorschlagen, bekannte Schwachstellen benennen oder Verbesserungen formulieren.

Je stärker eine Usability-Frage vom konkreten Nutzungskontext abhängt, desto vorsichtiger müssen AI-Ausgaben behandelt werden. Dies betrifft etwa medizinische Anwendungen, industrielle Steuerungen, Finanzsysteme, sicherheitskritische Prozesse, interne Fachanwendungen oder Systeme mit spezialisierten Nutzergruppen.

In solchen Fällen reichen generische Usability-Muster nicht aus. Entscheidend ist, wie konkrete Nutzerinnen und Nutzer unter realen Bedingungen arbeiten, welche Fachsprache sie verwenden, welche Fehler schwerwiegend sind und welche Erwartungen aus ihrer Praxis entstehen. Fachliteratur, Normen, Guidelines, Human-Factors-Analysen, Domänenwissen und reale Nutzerbeobachtung können erforderlich sein, um AI-Ausgaben angemessen einzuordnen.

Für agile Softwareentwicklungsteams ergibt sich daraus eine **einfache Prüflogik**: AI kann bei bekannten Mustern schnell helfen. Wenn jedoch konkrete Nutzer, Domänenwissen, Arbeitsabläufe,

Risiken oder rechtliche Anforderungen entscheidend sind, müssen AI-Vorschläge durch menschliches Fachwissen, Nutzerfeedback oder Tests geprüft werden.

2.11 AI-gestützte Usability-Reflexion und menschliche Prüfung

AI kann in diesem Kapitel besonders gut als Lern- und Reflexionswerkzeug eingesetzt werden.

AI unterstützt gut bei	Begriffe erklären; Human-Factors-Kriterien erläutern; eigene Artefakte analysieren; Varianten vergleichen; Gegenbeispiele erzeugen; typische Usability-Probleme aufzeigen.
Grenzen und typische Risiken	Keine gesicherte fachliche Wahrheit: Begriffe können falsch gewichtet, Beispiele zu allgemein und Erklärungen überzeugend, aber unvollständig sein; domänenspezifische Besonderheiten werden übersehen; Wahrnehmung, mentale Modelle und reales Verhalten konkreter Nutzergruppen kennt AI nicht aus eigener Erfahrung.
Menschlich zu prüfen	Abgleich mit den Kriterien menschlicher Wahrnehmung, Verständlichkeit, Erwartungskonformität, kognitiver Belastung und Accessibility; Berücksichtigung des konkreten Nutzungskontexts; Klärung, welche Aussagen durch Nutzerfeedback, Usability-Tests, Nutzungsdaten oder Domänenwissen weiter zu prüfen sind.

Entscheidend für den Kompetenzaufbau ist, dass Begriffe nicht nur wiederholt, sondern verstanden und auf eigene Artefakte und Entscheidungssituationen angewendet werden.

3 Discovery: Nutzer, Kontext, Probleme und Hypothesen verstehen

Unterrichtszeit: ca. 120 Minuten

3.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt Discovery für Usability vor oder neben der konkreten Umsetzung. Im Mittelpunkt steht die Frage, wie Softwareentwicklungsteams mit AI-Unterstützung ein besseres Verständnis von Nutzern, Aufgaben, Nutzungskontexten, Problemen und Produktannahmen aufbauen können, bevor Lösungen konkret entworfen oder umgesetzt werden.

Discovery wird in diesem Syllabus nicht als umfangreiches Research-Programm verstanden, das zwingend eigene Spezialrollen, lange Vorlaufzeiten oder große Budgets erfordert. Es geht um einen pragmatischen Einstieg: vorhandene Informationen nutzen, Hinweise auf Nutzerprobleme erkennen, Annahmen offenlegen, Hypothesen formulieren und entscheiden, wo weitere Klärung notwendig ist.

GenAI kann diesen Prozess unterstützen, indem sie Supporttickets, Nutzer- und Kundenfeedback, Interviewnotizen, qualitative Rückmeldungen, Analytics-Daten, interne Beobachtungen oder fachliche Beschreibungen strukturiert und wiederkehrende Muster herausarbeitet. Daraus können erste Problemhypothesen, Research-Fragen, Nutzungsszenarien oder Produktannahmen abgeleitet werden.

AI-gestützte Discovery bleibt jedoch besonders kontextabhängig. AI kann vorhandene Informationen ordnen, kennt aber reale Nutzerinnen und Nutzer, deren Aufgaben und deren Nutzungskontext nicht aus eigener Erfahrung. AI-generierte Aussagen sind daher als prüfbare Annahmen und nicht als gesicherte Erkenntnisse zu behandeln.

3.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 3.1 bis BO 3.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

3.3 Learning Objectives

LO	Learning Objective	K
LO 3.1	Die Begriffe Nutzer, Nutzerziel, Aufgabe, Nutzungskontext, Problemhypothese und Produktannahme wiedergeben können.	K1
LO 3.2	Erklären können, warum Usability-Arbeit vor der Lösungsidee mit dem Verständnis von Nutzer, Aufgabe und Kontext beginnen muss.	K2
LO 3.3	Typische Informationsquellen für Discovery für Usability benennen können, etwa Supporttickets, Kundenfeedback, Analytics, Interviews, Fachwissen und bestehende Dokumentation.	K1
LO 3.4	Beschreiben können, wie GenAI vorhandenes Nutzer- und Kundenfeedback, Supporthinweise, Beobachtungen und Daten strukturieren und zusammenfassen kann.	K2
LO 3.5	Aus vorhandenen Informationen mit AI-Unterstützung erste Usability-relevante Problemfelder ableiten können.	K3
LO 3.6	AI-generierte Nutzerannahmen als Hypothesen formulieren und von gesicherten Erkenntnissen unterscheiden können.	K3
LO 3.7	Erklären können, warum AI-generierte Personas und Szenarien ohne belastbare Daten problematisch sein können.	K2
LO 3.8	Beurteilen können, wann reale Nutzer, Domänenexperten oder zusätzliche Research-Aktivitäten erforderlich sind.	K3
LO 3.9	Aus Discovery-Erkenntnissen prüfbare Produktannahmen für Backlog Items, Prototypen oder Tests ableiten können.	K3

3.4 Wichtige Begriffe

Nutzer, Nutzergruppe, Nutzerziel, Aufgabe, Nutzungskontext, Domänenwissen, Problemhypothese, Produktannahme, Research-Frage, Nutzerfeedback, Kundenfeedback, Supportticket, Supportdaten, Analytics, Nutzungsdaten, Interviewnotiz, Beobachtung, Persona, datenbasierte Persona, hypothetische Persona, Nutzungsszenario, AI-gestützte Strukturierung, Muster, Hypothese, Evidenz, Validierungsbedarf, Discovery, Discovery-Ergebnis, Backlog-Anschluss

3.5 Discovery im Softwareentwicklungsprozess

Discovery beschreibt Aktivitäten, die helfen, ein Problem, eine Zielgruppe, eine Aufgabe oder einen Nutzungskontext besser zu verstehen, bevor eine Lösung umgesetzt wird. In agilen Entwicklungsprozessen kann Discovery vor einer Umsetzungsiteration stattfinden oder parallel zur Umsetzung laufen, wenn offene Fragen für kommende Arbeitsschritte geklärt werden.

Für agile Teams ist Discovery besonders wichtig, weil Backlog Items häufig bereits als Lösung formuliert werden. Eine Story beschreibt dann eine neue Funktion, ohne dass klar ist, welches Nutzerproblem damit gelöst werden soll. Discovery hilft, einen Schritt zurückzugehen: Welches Problem liegt vor? Für wen ist es relevant? In welcher Situation tritt es auf? Welche Annahmen treffen wir? Welche Informationen fehlen?

Discovery ist nicht nur dann wertvoll, wenn umfangreiche Research-Aktivitäten möglich sind. Auch kleine Schritte können relevant sein: vorhandenes Feedback sichten, Supporttickets clustern, offene Fragen sammeln, Annahmen formulieren, Nutzerziele präzisieren oder Domänenwissen einholen. Entscheidend ist, zwischen Lösung, Problem, Annahme und Erkenntnis zu unterscheiden.

In agilen Teams können verschiedene Rollen zur Discovery beitragen. Product Owner können Kundenfeedback einbringen, Entwicklerinnen und Entwickler können Supportfälle oder technische Einschränkungen analysieren, Testerinnen und Tester können wiederkehrende Fehler und Irritationen dokumentieren, Stakeholder können Domänenwissen beitragen, und bei Bedarf können reale Nutzerinnen und Nutzer befragt oder beobachtet werden.

Discovery ist nur dann wirksam, wenn ihre Ergebnisse in die weitere Arbeit überführt werden. Aus einem Problemfeld kann ein Backlog Item entstehen. Aus einer unsicheren Annahme kann eine Testfrage entstehen. Aus einer unklaren Nutzergruppe kann eine Research-Frage entstehen. Aus einem vermuteten Problem kann ein Prototyp entstehen, mit dem eine Lösungsidee geprüft wird.

3.6 Nutzer, Nutzerziel, Aufgabe und Nutzungskontext

Zu Beginn jeder Usability-orientierten Discovery steht die Frage, für wen ein System oder eine Funktion gedacht ist. „Der Nutzer“ ist häufig zu ungenau. Unterschiedliche Nutzergruppen können unterschiedliche Ziele, Rollen, Berechtigungen, Vorerfahrungen, Arbeitsbedingungen und Erwartungen haben.

Nutzergruppen sollten daher nicht nur demografisch beschrieben werden. Entscheidend ist ihre Beziehung zur Aufgabe und zum System. Relevant sind etwa Rolle, Ziel, Erfahrung, Fachwissen,

technische Kompetenz, Nutzungshäufigkeit, Verantwortung, Entscheidungsspielraum und mögliche Einschränkungen.

Ebenso ist zwischen Systemfunktion und Nutzerziel zu unterscheiden. Eine Funktion wie „Export als CSV“ beschreibt, was das System leisten soll. Das zugrunde liegende Nutzerziel kann jedoch unterschiedlich sein: Daten an die Buchhaltung übergeben, einen Bericht vorbereiten, eine externe Analyse durchführen oder Nachweispflichten erfüllen. Usability-Arbeit beginnt deshalb vor der Lösungsidee mit Nutzer, Aufgabe und Kontext.

Eine Aufgabe beschreibt, was Nutzerinnen und Nutzer tun müssen, um ein Ziel zu erreichen. Für UX ist relevant, wie diese Aufgabe im Alltag eingebettet ist: Welche Schritte gehen voraus? Welche Informationen werden benötigt? Welche Entscheidungen müssen getroffen werden? Welche Fehler sind wahrscheinlich? Welche Folgen hat ein Fehler? Welche Unterbrechungen oder Zeitbedingungen gibt es?

Der Nutzungskontext umfasst die Bedingungen, unter denen ein System genutzt wird. Dazu gehören physische, technische, organisatorische, soziale und psychologische Rahmenbedingungen. Ein System wird unterschiedlich erlebt, je nachdem, ob es im Büro, unterwegs am Smartphone, unter Zeitdruck, im Callcenter, in einer Werkhalle, im Krankenhaus oder in einer Prüfungssituation genutzt wird.

GenAI kann helfen, aus funktionalen Beschreibungen mögliche Nutzerziele, Aufgaben oder Kontextdimensionen abzuleiten. Diese Ableitungen sind jedoch Hypothesen. Das Team muss prüfen, ob sie zur Produktrealität, zur Nutzergruppe, zur Domäne und zu den tatsächlichen Nutzungssituationen passen.

3.7 Typische Informationsquellen für Usability-Discovery

Typische Informationsquellen für Usability-Discovery sind:

- Supporttickets, Kundenfeedback, Reviews und Freitextkommentare,
- Analytics- und Nutzungsdaten,
- Interviews, Beobachtungsnotizen und Workshop-Protokolle,
- Fachkonzepte, Prozessbeschreibungen und bestehende Dokumentation,
- Domänenwissen im Team.

Supporttickets und Servicedesk-Daten enthalten Hinweise darauf, wo Nutzerinnen und Nutzer scheitern, welche Begriffe sie nicht verstehen, welche Abläufe wiederholt Probleme erzeugen oder welche Funktionen häufig erklärt werden müssen. GenAI kann solche Tickets nach Themen gruppieren,

wiederkehrende Probleme identifizieren, ähnliche Fälle zusammenfassen oder erste Problemhypothesen formulieren.

Kundenfeedback, App-Store-Reviews, Umfragekommentare oder Freitextfelder können Hinweise auf Frustration, Wünsche, Erwartungen oder wiederkehrende Irritationen liefern. AI kann solche qualitativen Rückmeldungen zusammenfassen, clustern, Tonalität beschreiben und wiederkehrende Themen herausarbeiten. Die Ergebnisse sind als erste Strukturierung zu verstehen und müssen fachlich geprüft werden.

Analytics- und Nutzungsdaten können zeigen, was Nutzerinnen und Nutzer tun: wo sie abbrechen, welche Seiten sie häufig besuchen, welche Funktionen selten genutzt werden oder wie lange bestimmte Abläufe dauern. Diese Daten erklären jedoch nicht automatisch, warum etwas passiert. Eine hohe Abbruchrate kann unterschiedliche Ursachen haben, etwa unklare Texte, technische Fehler, fehlendes Vertrauen, falsche Erwartungen oder hohe kognitive Belastung.

Interviews, Beobachtungen und bestehende Dokumentation enthalten oft wertvolle Informationen über Nutzer, Aufgaben und Kontext. AI kann helfen, solche Materialien zusammenzufassen, zentrale Themen zu extrahieren, offene Fragen zu markieren oder Widersprüche aufzudecken. Dabei ist zwischen tatsächlicher Beobachtung, menschlicher Interpretation und AI-generierter Ergänzung zu unterscheiden.

Domänenwissen im Team oder bei Stakeholdern ist besonders wichtig, weil AI spezifische Fachlogik nicht zuverlässig kennt. Fachpersonen, Support, Tester, Product Owner, Compliance-Verantwortliche oder erfahrene Nutzer können helfen, AI-generierte Hypothesen fachlich einzuordnen und blinde Flecken zu erkennen.

Alle genannten Quellen haben Grenzen. Supporttickets zeigen nur gemeldete Probleme. Analytics-Daten zeigen Verhalten, aber nicht automatisch Ursachen. Interviews zeigen Perspektiven einzelner Personen. Fachwissen kann wertvoll sein, enthält aber ebenfalls Annahmen. AI kann solche Quellen strukturieren, aber nicht automatisch prüfen, ob sie vollständig, repräsentativ oder korrekt interpretiert sind.

3.8 GenAI zur Strukturierung vorhandener Informationen

Damit GenAI vorhandene Informationen sinnvoll strukturieren kann, müssen diese Informationen vorbereitet werden. Dabei ist zu prüfen, welche Daten verwendet werden dürfen. Personenbezogene, vertrauliche oder sensible Daten dürfen nicht unreflektiert in AI-Systeme eingegeben werden. Je nach Kontext sind Daten zu anonymisieren, zu pseudonymisieren, zu kürzen oder durch synthetische Beispiele zu ersetzen.

AI kann vorhandenes Nutzerfeedback, Kundenfeedback, Supporthinweise, Interviewnotizen, Beobachtungen oder Nutzungsdaten zusammenfassen, clustern und in erste Hypothesen überführen. Die Qualität dieser Strukturierung hängt direkt von der Qualität, Vollständigkeit und Zulässigkeit der Eingaben ab. Schlechte, einseitige, veraltete, unklare oder unzulässige Daten führen zu entsprechend unsicheren Ergebnissen.

Beim Clustering können Themen wie Login-Probleme, Fehlermeldungen, Navigation, Performance, Verständnisprobleme oder fehlende Funktionen sichtbar werden. Solche Cluster sind jedoch nur eine Strukturierungshilfe. Die von AI vorgeschlagenen Kategorien müssen geprüft werden, weil sie zu grob, zu spezifisch oder fachlich unpassend sein können. Auch Minderheitenthemen können durch Clustering unsichtbar werden, obwohl sie fachlich, regulatorisch oder geschäftlich relevant sind.

Aus Clustern können erste Problemfelder abgeleitet werden. Dabei ist zu beachten, dass nicht jeder Cluster bereits ein präzises Usability-Problem beschreibt. Aus einem Cluster wie „viele Tickets zu Passwort“ muss erst genauer herausgearbeitet werden, ob es um Passwortregeln, Reset-Prozess, Fehlermeldung, E-Mail-Zustellung, Konto-Sperre oder Vertrauen geht.

AI kann außerdem wiederkehrende Formulierungen, emotionale Signale, Widersprüche oder offene Fragen aufzeigen. Ein wichtiger Nutzen liegt darin, nicht nur Antworten zu erzeugen, sondern fehlende Informationen, implizite Annahmen und weitere Klärungsbedarfe zu erkennen. Daraus können Research-Fragen, Klärungsfragen für Stakeholder oder Testfragen für spätere Usability-Tests entstehen.

3.9 Von Informationen zu Problemfeldern für Usability

Ein Problemfeld beschreibt einen Bereich, in dem Nutzerinnen und Nutzer wiederholt Schwierigkeiten, Unsicherheiten, Fehler oder Abbrüche erleben. Es sollte nicht vorschnell als Lösung formuliert werden. Eine Aussage wie „Wir brauchen einen neuen Wizard“ beschreibt bereits eine mögliche Lösung, aber noch nicht das zugrunde liegende Problem.

Besser ist eine problemorientierte Formulierung, etwa: „Nutzer verstehen im aktuellen Prozess nicht, welche Angaben im zweiten Schritt erforderlich sind.“ Diese Formulierung zeigt, wo die Schwierigkeit liegt, ohne bereits festzulegen, wie sie gelöst werden soll.

Ein gutes Problemfeld beschreibt nach Möglichkeit Nutzergruppe, Aufgabe, Situation, beobachtetes oder vermutetes Problem und mögliche Auswirkung. Dadurch wird es für Requirements, Prototyping und Testing anschlussfähig. Allgemeine Themen wie „Navigation“, „Passwort“ oder „Onboarding“ müssen daher weiter präzisiert werden, bevor sie als nutzungsbezogene Problemfelder bearbeitet werden können.

Aus vorhandenen Informationen wie Supporttickets, Feedback, Beobachtungen, Analytics-Daten oder Fachwissen können solche Problemfelder mit AI-Unterstützung abgeleitet werden. AI kann Einzelhinweise zusammenführen und Formulierungen vorschlagen. Das Team muss prüfen, ob die vorgeschlagenen Problemfelder durch vorhandene Hinweise gedeckt, ausreichend konkret und noch lösungsoffen formuliert sind.

In Discovery ist außerdem zwischen Symptomen und Ursachen zu unterscheiden. Ein Symptom kann sein, dass Nutzer einen Prozess abbrechen. Mögliche Ursachen können unklare Pflichtfelder, fehlendes Vertrauen, fachliche Unsicherheit, technische Fehler oder ungeeignete Systemrückmeldungen sein. AI kann mögliche Ursachen vorschlagen, diese bleiben jedoch Hypothesen.

Problemfelder können bereits grob eingeordnet werden. Kriterien dafür sind etwa Häufigkeit, Nutzerwirkung, Risiko, Business-Relevanz, strategische Bedeutung, regulatorische Relevanz oder Umsetzungsnähe. AI kann eine erste Sortierung unterstützen. Die tatsächliche Bewertung muss jedoch durch das Team erfolgen, weil AI Geschäftsziele, Risiken, regulatorische Anforderungen, technische Abhängigkeiten und organisatorische Zwänge nicht automatisch kennt.

3.10 Hypothesen statt Gewissheiten

Eine Hypothese ist eine prüfbare Annahme. Sie beschreibt etwas, das plausibel sein könnte, aber noch nicht ausreichend belegt ist. Discovery erzeugt häufig Hypothesen über Nutzerziele, Probleme, Ursachen, Erwartungen oder mögliche Lösungen.

AI-generierte Nutzerannahmen sind daher nicht als gesicherte Erkenntnisse zu behandeln. Eine AI-Aussage wie „Nutzer bevorzugen eine Schritt-für-Schritt-Führung“ ist ohne Daten keine Erkenntnis. Sie kann aber als Hypothese formuliert werden: „Wir nehmen an, dass Nutzer im aktuellen Prozess eine Schritt-für-Schritt-Führung benötigen, weil sie den Gesamtprozess nicht überblicken.“

Eine gute Hypothese ist konkret, überprüfbar und an eine Nutzergruppe, Aufgabe oder Situation gebunden. „Die Oberfläche ist schlecht“ ist keine brauchbare Hypothese. „Erstnutzer finden den Export nicht, weil die Funktion nur in einem Kontextmenü verfügbar ist“ ist besser prüfbar.

Teilnehmende sollen AI nutzen können, um unklare Annahmen zu schärfen: Wer ist betroffen? Welche Aufgabe ist relevant? In welcher Situation tritt das Problem auf? Welche Ursache wird vermutet? Welche Beobachtung oder welcher Test könnte die Annahme prüfen?

Hypothesen sind von Erkenntnissen zu unterscheiden. Eine Erkenntnis entsteht durch belastbare Beobachtung, Datenanalyse, Evaluation, Usability-Test oder fachliche Prüfung. Eine Hypothese ist dagegen ein möglicher Erklärungsansatz, der noch überprüft werden muss. Diese Unterscheidung ist

zentral, weil AI-Ausgaben häufig so formuliert sind, als wären Annahmen bereits gesicherte Erkenntnisse.

Hypothesen sollten dokumentiert werden, damit im weiteren Entwicklungsprozess nachvollziehbar bleibt, welche Annahmen einer Story, einem Prototyp oder einer Entscheidung zugrunde liegen. Eine Hypothese kann später bestätigt, verworfen oder angepasst werden.

3.11 Personas und Nutzungsszenarien mit Vorsicht einsetzen

Personas und Nutzungsszenarien können helfen, Nutzerperspektiven im Team greifbarer zu machen. Eine Persona beschreibt typisierte Eigenschaften, Ziele, Bedürfnisse und Rahmenbedingungen einer Nutzergruppe. Ein Szenario beschreibt eine konkrete Nutzungssituation oder Aufgabe.

Ihr Nutzen liegt darin, Diskussionen zu konkretisieren und Anforderungen aus Nutzerperspektive zu betrachten. Statt allgemein über „die Nutzer“ zu sprechen, kann ein Team präziser fragen, welche Rolle eine Nutzergruppe hat, welches Ziel sie verfolgt, welche Vorerfahrung sie mitbringt und in welcher Situation eine Funktion verwendet wird.

AI kann Personas und Szenarien schnell erzeugen. Ohne belastbare Daten sind solche Artefakte jedoch problematisch. Sie können professionell wirken, aber auf Annahmen, Stereotypen oder generischen Mustern beruhen. Deshalb sollten AI-generierte Personas und Szenarien nur als Hypothesen oder Diskussionsgrundlage verwendet werden, solange sie nicht durch reale Daten, Domänenwissen, Interviews, Beobachtungen oder andere belastbare Quellen gestützt sind.

Eine datenbasierte Persona stützt sich auf Interviews, Beobachtungen, Nutzungsdaten oder belastbares Fachwissen. Eine hypothetische Persona dient dagegen dazu, Annahmen offenzulegen. Beide Formen können nützlich sein, müssen aber klar unterschieden und gekennzeichnet werden.

Bei jeder Persona ist zu prüfen, welche Aussagen belegt sind und welche nur vermutet werden. Aussagen zu Zielen, Aufgaben, Problemen, Fachwissen, Nutzungshäufigkeit, Einschränkungen oder Erwartungen sollten möglichst auf nachvollziehbare Quellen zurückgeführt werden. Dadurch wird verhindert, dass AI-generierte Personas im Team als Fakten behandelt werden.

Auch AI-generierte Nutzungsszenarien sind als Hypothesen zu verstehen. Sie sind dann nützlich, wenn sie konkrete Fragen für Backlog, Prototyp oder Test erzeugen. Teilnehmende sollen Szenarien auf Plausibilität prüfen: Passt die Aufgabe zur Nutzergruppe? Ist der Kontext realistisch? Sind wichtige Einschränkungen berücksichtigt? Welche Annahmen müssten durch Nutzer, Domänenexperten oder Daten überprüft werden?

3.12 Wann reale Nutzer, Domänenexperten oder zusätzliche Research-Aktivitäten erforderlich sind

AI kann vorhandene Informationen strukturieren, zusammenfassen und erste Hypothesen ableiten. Sie verfügt jedoch über keine eigene Erfahrung damit, wie reale Nutzerinnen und Nutzer in einer konkreten Situation arbeiten, denken, zögern, Fehler machen oder Entscheidungen treffen. Wenn die zentrale Frage lautet, wie Nutzer tatsächlich handeln, reichen AI-gestützte Annahmen nicht aus.

Zusätzliche Research-Aktivitäten sind besonders wichtig bei unklaren Nutzerzielen, widersprüchlichem Feedback, hoher Unsicherheit, sicherheitskritischen Funktionen, neuen Zielgruppen, fachlich komplexen Prozessen oder Entscheidungen mit hoher Auswirkung. In solchen Fällen können reale Nutzer, Domänenexpertinnen und Domänenexperten, Beobachtungen, Interviews, Usability-Tests oder weitere Datenquellen erforderlich sein.

Domänenexperten können wichtige Hinweise zu fachlichen Regeln, Risiken, Arbeitsabläufen, Ausnahmesituationen oder organisatorischen Rahmenbedingungen liefern. Dazu zählen beispielsweise interne Fachpersonen, Supportmitarbeitende, Trainerinnen und Trainer, Compliance-Verantwortliche oder besonders erfahrene Anwenderinnen und Anwender. Ihre Expertise ist besonders relevant, wenn AI generische Annahmen erzeugt, Fachsprache unklar bleibt oder regulatorische Anforderungen betroffen sind.

Reale Nutzerinnen und Nutzer sind erforderlich, wenn Annahmen über Verhalten, Verständnis, Erwartungen, mentale Modelle oder tatsächliche Nutzungssituationen überprüft werden sollen. AI kann mögliche Nutzerreaktionen simulieren und dadurch Hypothesen erzeugen. Solche Simulationen sind jedoch keine Validierung, weil sie kein reales Verhalten, kein echtes Zögern, keine tatsächlichen Fehlhandlungen und keine situativen Einflüsse beobachten.

Zusätzliche Research-Aktivitäten müssen nicht immer groß angelegt sein. Möglich sind kurze Interviews, Beobachtungen, interne Domänenreviews, kurze Usability-Tests, Rückfragen an Support oder Analyse einzelner Nutzerpfade. GenAI kann solche Aktivitäten vorbereiten, etwa durch Interviewfragen, Beobachtungsleitfäden, Hypothesenlisten oder Testaufgaben. Durchführung, Bewertung und Verantwortung bleiben menschliche Aufgaben.

3.13 Discovery-Erkenntnisse in anschlussfähige Artefakte überführen

Discovery-Ergebnisse sollten so formuliert werden, dass sie im weiteren agilen Entwicklungsprozess nutzbar sind. Sie können in Problemhypothesen, Produktannahmen, Backlog Items, Prototyp-Fragen, Akzeptanzkriterien oder Testaufgaben überführt werden.

Aus einem Problemfeld kann ein Backlog Item entstehen, wenn erkennbar ist, welches Nutzerproblem bearbeitet werden soll, welche Annahme dahintersteht und welcher nächste Klärungs- oder Umsetzungsschritt sinnvoll ist. Dabei ist wichtig, Backlog Items nicht vorschnell als fertige Lösung zu formulieren. Wenn Ursache, Nutzerziel oder geeignete Lösung noch unsicher sind, kann ein Backlog Item zunächst auch eine Klärungs-, Prototyping- oder Testaufgabe sein.

Wenn unklar ist, welche Lösung geeignet ist, können Discovery-Erkenntnisse in Prototyp-Fragen übersetzt werden. Dabei ist zu klären, welche Annahme offengelegt oder geprüft werden soll und welche Nutzerreaktion relevant ist. AI kann helfen, solche Fragen aus Hypothesen abzuleiten. Das Team muss prüfen, ob der vorgeschlagene Prototyp tatsächlich die relevante Annahme adressiert oder nur eine oberflächliche Lösung visualisiert.

Unsichere Annahmen können außerdem in überprüfbare Testfragen überführt werden. Aus der Annahme „Nutzer verstehen den nächsten Schritt nicht“ kann zum Beispiel die Usability-Testfrage entstehen: „Erkennen Nutzer nach Abschluss von Schritt 1, was sie in Schritt 2 tun müssen?“ Damit wird eine vage Vermutung in eine konkrete Fragestellung übersetzt, die durch Beobachtung, Prototyping oder Usability-Tests geprüft werden kann.

Discovery-Ergebnisse sollten nachvollziehbar dokumentiert werden. Dazu gehört, welche Quellen verwendet wurden, welche AI-Unterstützung eingesetzt wurde, welche Hypothesen entstanden sind, welche Annahmen unsicher bleiben und welche nächsten Schritte vorgeschlagen werden. AI kann bei der Dokumentation unterstützen. Das Team muss jedoch prüfen, ob die Dokumentation fachlich korrekt ist und nicht mehr Sicherheit suggeriert, als tatsächlich vorhanden ist.

3.14 AI-gestützte Discovery und menschliche Prüfung

AI kann in der Discovery besonders gut vorhandene Informationen strukturieren und leichter zugänglich machen.

AI unterstützt gut bei	Supporttickets und Feedback clustern; Interviewnotizen zusammenfassen; wiederkehrende Themen aufzeigen; mögliche Nutzerprobleme formulieren; Hypothesen und Research-Fragen ableiten; Nutzungsszenarien als Diskussionsgrundlage erzeugen; Discovery-Ergebnisse für Backlog, Prototyping oder Tests strukturieren.
Grenzen und typische Risiken	Ersetzt keine realen Nutzer und kennt deren Ziele, Erwartungen und Barrieren nicht aus eigener Erfahrung; kann die Repräsentativität von Daten nicht sicher beurteilen und führt Verzerrungen einseitigen

	Feedbacks fort; problematisch wird es, wenn AI-Annahmen als gesicherte Erkenntnisse gelten, synthetische Personas als Research-Ergebnisse erscheinen, Symptome vorschnell als Ursachen gedeutet oder Lösungen vor dem Problemverständnis formuliert werden.
Menschlich zu prüfen	Welche Nutzergruppe tatsächlich gemeint ist; welches Nutzerziel oder welche Aufgabe im Mittelpunkt steht; in welchem Nutzungskontext das Problem auftritt; welche Daten die Annahme stützen; welche Aussagen belegt und welche hypothetisch sind; ob Domänenexperten, reale Nutzerinnen und Nutzer oder zusätzliche Research-Aktivitäten erforderlich sind.

AI-gestützte Discovery ist hilfreich, wenn sie Annahmen offenlegt, Fragen präzisiert und nächste Schritte anschlussfähig macht.

4 Backlog Refinement und Usability

Unterrichtszeit: ca. 120 Minuten

4.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt das Backlog Refinement als zentrale Stelle im Softwareentwicklungsprozess, an der Usability früh beeinflusst werden kann. Viele Probleme der Nutzung entstehen nicht erst im Design oder in der Implementierung, sondern bereits dort, wo Anforderungen zu allgemein, zu funktional oder zu stark aus Systemsicht beschrieben werden.

Backlog Items, User Stories und Akzeptanzkriterien bestimmen, worauf das Team in der Umsetzung achtet. Wenn dort nur die Funktion beschrieben wird, bleiben Nutzerziel, Nutzungskontext, Verständlichkeit, Systemzustände, Fehlerfälle, Accessibility, erwartbares Verhalten und Usability oft unsichtbar.

GenAI kann helfen, solche Lücken aufzudecken, etwa unklare Formulierungen, fehlende Usability-Aspekte, Happy-Path-Fixierung, fehlende States, Edge Cases oder Accessibility-Aspekte. AI-generierte Vorschläge sind jedoch keine verbindlichen Anforderungen. Sie müssen fachlich geprüft, priorisiert und an Produkt, Nutzergruppe und Nutzungskontext angepasst werden.

Ziel dieses Kapitels ist es, Usability im Backlog nicht als zusätzliche Designaufgabe zu behandeln, sondern als Bestandteil professioneller Requirements-Arbeit.

4.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 4.1 bis BO 4.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

4.3 Learning Objectives

LO	Learning Objective	K
LO 4.1	Die Begriffe Usability-Requirement, User Story, Akzeptanzkriterium, Definition of Ready und Definition of Done wiedergeben können.	K1
LO 4.2	Erklären können, warum Usability-Anforderungen Bestandteil professioneller Requirements-Arbeit sind.	K2

LO	Learning Objective	K
LO 4.3	Typische implizite Usability-Anforderungen beschreiben können, etwa Verständlichkeit, Fehlertoleranz, Systemfeedback, Systemzustände und Accessibility-Basics.	K2
LO 4.4	User Stories aus Nutzungsperspektive analysieren können, insbesondere im Hinblick auf Nutzerziel, Nutzungskontext und erwartetes Verhalten.	K3
LO 4.5	GenAI nutzen können, um Backlog Items auf unklare Formulierungen, fehlende Usability-Aspekte sowie Happy-Path-Fixierung zu prüfen.	K3
LO 4.6	Akzeptanzkriterien mit Usability-Aspekten wie Fehlermeldungen, Eingabevalidierung, Systemzuständen, Tastaturbedienbarkeit oder verständlichem Systemfeedback ergänzen können.	K3
LO 4.7	Usability-relevante Kriterien für Definition of Ready und Definition of Done erklären können.	K2
LO 4.8	AI-generierte Vorschläge für Requirements und Akzeptanzkriterien fachlich beurteilen und entscheiden können, ob sie übernommen, angepasst oder begründet ausgeschlossen werden müssen.	K3
LO 4.9	Erklären können, warum AI keine verbindliche Priorisierung oder fachliche Notwendigkeit im konkreten Produktkontext bestimmen kann.	K2

4.4 Wichtige Begriffe

Usability-Requirement, Nutzeranforderung, User Story, Backlog Item, Akzeptanzkriterium, Definition of Ready, Definition of Done, Nutzerziel, Nutzungskontext, implizite Anforderung, Qualitätskriterium, Fehlertoleranz, Systemfeedback, Eingabevalidierung, Validierungsbedarf, State / Systemzustand, Edge Case, Happy Path, Accessibility-Basics, erwartbares Verhalten, AI-gestütztes Refinement, Hypothese, Evidenz, fachliche Prüfung, Priorisierung.

4.5 Backlog Refinement als Übersetzungspunkt

Backlog Refinement bildet den Übergang zwischen Problemverständnis und Umsetzung. In dieser Phase werden Ideen, Anforderungen, Hypothesen und Produktentscheidungen so geschärft, dass sie in der Umsetzung bearbeitbar werden. Dabei entscheidet sich häufig, ob Usability und Nutzungserfahrung bewusst berücksichtigt werden oder ob Verständlichkeit, Zustände, Fehlerfälle und Nutzungskontext unbeachtet bleiben.

Wenn Discovery-Erkenntnisse nicht in klare Backlog Items überführt werden, bleiben sie unverbindlich. Wenn User Stories nur technische oder funktionale Aspekte enthalten, fehlen dem Team Hinweise

darauf, wie die Lösung für Nutzerinnen und Nutzer funktionieren soll. Wenn Akzeptanzkriterien nur den erfolgreichen Ablauf beschreiben, bleiben Fehlerfälle, alternative Zustände und Verständlichkeit offen.

Ein gutes Refinement klärt daher nicht nur, was gebaut werden soll, sondern auch, für wen, in welchem Nutzungskontext, mit welchem Ziel und unter welchen Qualitätsbedingungen. Usability-Anforderungen sind dabei keine Ergänzung außerhalb der Requirements, sondern beschreiben Qualitätsaspekte der Nutzung.

GenAI kann Backlog Items aus einer zusätzlichen Perspektive prüfen. Sie kann auf fehlende Nutzerziele, unklare Formulierungen, fehlende Akzeptanzkriterien, Happy-Path-Fixierung, nicht berücksichtigte Fehlerfälle oder fehlende Accessibility-Aspekte hinweisen. Die fachliche Entscheidung bleibt beim Team.

4.6 Usability-Requirements als Bestandteil professioneller Requirements-Arbeit

Funktionale Anforderungen beschreiben, was ein System leisten soll. Usability-Requirements beschreiben, unter welchen Bedingungen diese Leistung von bestimmten Nutzerinnen und Nutzern in einem bestimmten Nutzungskontext effektiv, effizient und zufriedenstellend genutzt werden kann. Dazu gehören Verständlichkeit, Erwartungskonformität, Fehlertoleranz, klares Systemfeedback, passende Systemzustände und Accessibility-Basics.

Funktionale Anforderungen und Usability-Requirements gehören zusammen. Bei einer Suche reicht die reine Funktion „Suchergebnisse anzeigen“ nicht aus. Nutzungsqualität umfasst auch verständliche Eingaben, sinnvolle Sortierung, Leerergebnisse, Ladezustände, Filterlogik, Tastaturbedienbarkeit und klares Systemfeedback.

Usability-Requirements entstehen aus Nutzerziel, Aufgabe und Nutzungskontext. Ein Nutzer, der in Ruhe ein persönliches Profil bearbeitet, benötigt andere Unterstützung als ein Nutzer, der unter Zeitdruck eine kritische Entscheidung trifft. Ein gelegentlicher Nutzer benötigt andere Orientierung als ein täglicher Fachanwender.

Backlog Items sollten daher nicht nur beschreiben, welche Funktion benötigt wird, sondern auch, welches Ziel damit erreicht werden soll. Aus einer Story wie „Als Sachbearbeiter möchte ich offene Fälle filtern, damit ich dringende Anträge zuerst bearbeiten kann“ ergeben sich Usability-Anforderungen an Sortierung, Prioritätsanzeige, Filterzustand, leere Ergebnisse und Rückmeldung bei geänderten Filtern.

Usability-Requirements sollten so formuliert werden, dass sie geprüft werden können. Allgemeine Aussagen wie „Die Oberfläche soll intuitiv sein“ sind zu unpräzise. Besser sind Kriterien, die verifiziert oder validiert werden können, etwa verständliche Fehlermeldungen, sichtbare Systemzustände, Tastaturbedienbarkeit, erkennbare Pflichtfelder oder klare Rückmeldung nach dem Speichern.

4.7 Implizite Usability-Anforderungen sichtbar machen

Viele Usability-Anforderungen werden nicht ausdrücklich formuliert, weil sie selbstverständlich erscheinen. Nutzerinnen und Nutzer erwarten verständliche Fehlermeldungen, klares Systemfeedback, nachvollziehbare Zustände, lesbare Texte und erwartbares Systemverhalten. Gerade weil diese Erwartungen selbstverständlich wirken, werden sie in Anforderungen oft nicht beschrieben.

Typische implizite Usability-Anforderungen betreffen Verständlichkeit, Systemfeedback, Fehlertoleranz, Orientierung, Konsistenz, Systemzustände und Accessibility-Basics. Dazu gehören etwa erkennbare Pflichtfelder, verständliche Validierung, Ladezustände, Empty States, Error States, eindeutige Bestätigungen, Tastaturbedienbarkeit und sichtbare Fokuszustände.

Wenn solche Anforderungen nicht sichtbar gemacht werden, werden sie während der Umsetzung unterschiedlich interpretiert oder übersehen. Das führt zu inkonsistenten Oberflächen, unklaren Abläufen, fehlenden Fehlerzuständen oder schlecht nutzbaren Komponenten.

GenAI kann helfen, implizite Anforderungen sichtbar zu machen. Ein Backlog Item kann gezielt darauf geprüft werden, welche Usability-Aspekte fehlen, welche Zustände berücksichtigt werden sollten, welche Fehlerfälle wahrscheinlich sind und welche Fragen vor der Umsetzung offen bleiben.

AI kann typische Lücken benennen, etwa: Was passiert bei ungültiger Eingabe? Was sieht der Nutzer während des Ladens? Was passiert, wenn keine Daten vorhanden sind? Wie wird Erfolg bestätigt? Wie wird ein Fehler behoben? Welche Information braucht der Nutzer vor, während und nach der Aktion? Solche Fragen unterstützen das Refinement, ersetzen aber nicht die fachliche Entscheidung, welche Anforderungen tatsächlich relevant sind.

4.8 User Stories aus Nutzungsperspektive prüfen

Eine User Story beschreibt üblicherweise Rolle, Ziel und Nutzen. Die bekannte Form „Als … möchte ich …, damit …“ kann hilfreich sein, wenn sie nicht rein formal verwendet wird. Entscheidend ist, dass die Story ein echtes Nutzerziel ausdrückt.

Eine schwache Story beschreibt häufig nur eine Systemfunktion, etwa: „Als Nutzer möchte ich einen Button haben, damit ich speichern kann.“ Aus Nutzungsperspektive ist relevanter, welches Ziel erreicht werden soll, etwa: „Als Nutzer möchte ich meine Änderungen speichern und eine eindeutige Bestätigung erhalten, damit ich sicher sein kann, dass meine Eingaben übernommen wurden.“

Beim Refinement sollte geprüft werden, ob das Nutzerziel klar ist. Wenn das Ziel fehlt, wird oft nur eine Systemfunktion umgesetzt. AI kann helfen, mögliche Nutzerziele aus einer Story abzuleiten. Das Team muss prüfen, ob diese Ziele zutreffen, insbesondere bei internen Fachanwendungen, regulatorischen Prozessen oder spezialisierten Arbeitsabläufen.

Auch der Nutzungskontext muss ausreichend geklärt sein. Relevante Fragen sind: Wie häufig wird die Funktion genutzt? Unter welchem Zeitdruck? Auf welchem Gerät? Mit welchem Vorwissen? Welche Fehler wären kritisch? Welche Daten oder Vorbedingungen gibt es? Diese Informationen müssen nicht immer vollständig in der Story stehen, sollten aber im Refinement bekannt oder als offene Frage dokumentiert sein.

User Stories sollten außerdem nicht nur den erfolgreichen Ablauf beschreiben. Erwartetes Verhalten umfasst auch Systemfeedback, Validierungen, Fehlermeldungen, Zustände und weitere relevante Systemreaktionen. Wenn Nutzer ein Formular absenden, muss nicht nur gespeichert werden. Das System sollte anzeigen, ob der Vorgang erfolgreich war, welche Eingaben ungültig sind, ob Daten noch verarbeitet werden oder ob später erneut versucht werden muss.

4.9 Backlog Items mit GenAI prüfen

GenAI kann genutzt werden, um Backlog Items auf unklare Formulierungen, fehlende Usability-Aspekte und Happy-Path-Fixierung zu prüfen. Besonders nützlich ist diese Unterstützung bei bekannten Mustern wie Formularen, Login, Suche, Upload, Tabellen, Filtern, Benachrichtigungen oder Checkout-Prozessen.

Backlog Items enthalten häufig unklare Begriffe wie „einfach“, „intuitiv“, „schnell“, „übersichtlich“ oder „benutzerfreundlich“. Solche Begriffe können sinnvoll sein, sind aber ohne Konkretisierung nicht prüfbar. AI kann unklare Formulierungen markieren und präzisere Alternativen vorschlagen. Das Team entscheidet, welche Kriterien relevant und realistisch sind.

AI kann außerdem prüfen, ob Nutzerziel, Kontext, Systemfeedback, Fehlerfälle, Zustände, Accessibility, Konsistenz und erwartbares Verhalten fehlen. Dabei sollte nicht allgemein gefragt werden, ob eine Story „gut“ ist, sondern gezielt, welche Usability-Anforderungen fehlen, welche Akzeptanzkriterien nur den Happy Path betreffen, welche Zustände berücksichtigt werden müssen und welche Accessibility-Basics relevant sind.

Viele Backlog Items beschreiben nur den erfolgreichen Ablauf. Usability-Probleme entstehen jedoch häufig in Abweichungen: ungültige Eingaben, fehlende Daten, Berechtigungsprobleme, Netzwerkfehler, langsame Antwortzeiten, doppelte Aktionen oder Abbruch eines Prozesses. AI kann typische Nicht-Happy-Path-Situationen vorschlagen. Diese müssen nicht alle umgesetzt werden, sollten aber bewusst geprüft werden.

Ein gutes Refinement deckt auch offene Fragen auf. Dabei geht es nicht darum, sofort alle Fragen zu beantworten, sondern Unsicherheiten bewusst zu dokumentieren. Offene Fragen können zu Klärungsaufgaben, Spike Stories, Prototypen oder Testaufgaben führen.

4.10 Akzeptanzkriterien mit Usability-Aspekten ergänzen

Akzeptanzkriterien beschreiben, wann ein Backlog Item als erfüllt gilt. Sie sind für Usability besonders wichtig, weil sie bestimmen, welche Aspekte in der Umsetzung überprüft werden. Wenn Akzeptanzkriterien nur technische Erfolgsbedingungen enthalten, wird Usability nicht systematisch geprüft.

Akzeptanzkriterien sollten daher auch Nutzerverhalten, Verständlichkeit, Systemfeedback, Fehlerfälle, Systemzustände und Accessibility-Basics berücksichtigen. Beispiele sind eindeutige Bestätigung nach dem Speichern, verständliche Korrekturanweisung bei ungültiger Eingabe, Erhalt bereits eingegebener Daten nach einem Fehler, sichtbarer Ladezustand bei längerer Verarbeitung oder ein erklärender Empty State bei null Ergebnissen.

Systemzustände sind dabei nicht nur technische Details. Loading State, Empty State, Error State, Success State, Disabled State, fehlende Berechtigung, unvollständige Daten oder abgebrochene Aktionen beeinflussen direkt, ob Nutzerinnen und Nutzer einen Ablauf verstehen und Fehler korrigieren können.

Accessibility-Anforderungen sollten in Akzeptanzkriterien aufgenommen werden, wenn sie für das jeweilige Backlog Item relevant sind. Dazu gehören etwa Tastaturbedienbarkeit, sichtbarer Fokus, verständliche Labels, ausreichender Kontrast, semantische Struktur und verständliche Fehlermeldungen. Ob eine bestimmte WCAG-Konformitätsstufe gefordert ist, muss abhängig von Produkt, Zielgruppe, rechtlichem Rahmen und organisatorischen Vorgaben festgelegt werden.

AI kann solche Kriterien vorschlagen. Eine vollständige Accessibility-Konformität oder fachliche Richtigkeit kann daraus jedoch nicht abgeleitet werden. Die Kriterien müssen durch das Team geprüft und an Produkt, Nutzungskontext und Qualitätsanforderungen angepasst werden.

4.11 Definition of Ready und Definition of Done mit Usability-Bezug

Die Definition of Ready beschreibt, wann ein Backlog Item ausreichend vorbereitet ist, um in die Umsetzung beziehungsweise in einen Sprint aufgenommen zu werden. Kriterien für Usability können helfen, unklare oder zu früh formulierte Stories zu erkennen.

Mögliche Usability-Kriterien für Ready sind ein beschriebenes Nutzerziel, eine bekannte Nutzergruppe, ein ausreichend geklärter Nutzungskontext, dokumentierte offene Fragen, zentrale Zustände und Fehlerfälle, relevante Accessibility-Basics sowie geklärter Validierungsbedarf.

Die Definition of Done beschreibt, wann ein Inkrement als abgeschlossen betrachtet wird. Usability-relevante Kriterien können sicherstellen, dass nicht nur Funktionalität, sondern auch Nutzungsqualität berücksichtigt wurde. Dazu gehören erfüllte Akzeptanzkriterien inklusive Usability-Aspekten, relevante Systemzustände, verständliche Fehlermeldungen, geprüfte Tastaturbedienbarkeit, konsistente UI-Texte, grundlegende Accessibility-Prüfungen, menschlich geprüfte AI-Ergebnisse und dokumentierte offene Usability-Risiken.

Definition of Ready und Definition of Done sind Teamvereinbarungen. Für UX ist entscheidend, dass sie zur Produktart, Teamreife und Risikosituation passen. Ein kleines internes Tool benötigt andere Kriterien als ein öffentliches E-Commerce-System oder eine sicherheitskritische Fachanwendung.

AI kann Vorschläge für DoR- und DoD-Kriterien erzeugen. Die Auswahl muss jedoch zum konkreten Produkt, Nutzungskontext, Qualitätsanspruch und Risiko passen.

4.12 AI-generierte Anforderungen fachlich einordnen

AI-generierte Requirements und Akzeptanzkriterien sind zunächst Rohmaterial. Sie können nützlich sein, aber auch überzogen, generisch, unpassend oder fachlich falsch. Ein häufiger Fehler besteht darin, AI-Vorschläge vollständig zu übernehmen, weil sie formal gut klingen.

Teilnehmende sollen AI-Vorschläge anhand klarer Kriterien prüfen: Passt der Vorschlag zum Nutzerziel? Passt er zum Nutzungskontext? Ist er fachlich korrekt? Ist er prüfbar? Ist er für die Umsetzung relevant? Ist er notwendig oder nur wünschenswert? Erzeugt er unverhältnismäßigen Aufwand?

Relevanz und Priorität sind zu unterscheiden. Nicht jeder relevante Usability-Aspekt hat dieselbe Dringlichkeit. Ein fehlender sichtbarer Fokus kann für Tastaturnutzer kritisch sein. Ein leicht verbesserbarer Hilfetext kann wichtig, aber weniger dringend sein. Eine missverständliche Fehlermeldung in einem kritischen Prozess kann schwerwiegend sein.

AI kann Prioritäten vorschlagen, kennt aber nicht automatisch Business-Ziele, Risiken, Nutzerzahlen, regulatorische Anforderungen oder technische Abhängigkeiten. Sie kann auch keine verbindliche fachliche Notwendigkeit im konkreten Produktkontext bestimmen. Priorisierung und Verantwortung bleiben Aufgabe des Teams.

Fachliche Korrektheit ist besonders bei domänenspezifischen Prozessen, rechtlichen Anforderungen, medizinischen Anwendungen, Finanzsystemen oder interner Fachlogik kritisch zu prüfen. Ein Vorschlag, der in einer Standardanwendung sinnvoll wäre, kann in einem regulierten oder sicherheitskritischen Prozess unpassend sein.

AI neigt außerdem dazu, umfangreiche Kriterienlisten zu erzeugen. Diese können hilfreich sein, aber auch ein Backlog Item überfrachten. Gutes Refinement entscheidet, was in der nächsten Umsetzung erforderlich ist, was später bearbeitet werden kann und was bewusst nicht Teil des Items ist.

4.13 AI-gestütztes Refinement und menschliche Prüfung

AI kann im Backlog Refinement als Review- und Strukturierungswerkzeug eingesetzt werden.

AI unterstützt gut bei	User Stories auf Nutzerziel und Nutzungskontext prüfen; unklare Formulierungen markieren; fehlende Usability-Aspekte benennen; Akzeptanzkriterien vorschlagen; Zustände und Fehlerfälle ergänzen; Happy-Path-Fixierung aufdecken; Accessibility-Basics vorschlagen; offene Fragen formulieren; DoR- und DoD-Kriterien vorbereiten.
Grenzen und typische Risiken	Entscheidet nicht verbindlich, welche Anforderungen fachlich notwendig, geschäftlich priorisiert oder regulatorisch ausreichend sind; der Produktkontext ist nur so gut wie die eingegebenen Informationen, fehlende oder falsche Angaben führen dennoch zu plausiblen Vorschlägen; typische Risiken: ungeprüft übernommene Akzeptanzkriterien, überladene Backlog Items, fachlich falsche Annahmen, oberflächlich behandelte Accessibility, weiterhin unberücksichtigte Edge Cases, zu allgemeine DoR- oder DoD-Kriterien, nicht testbar formulierte Usability-Anforderungen.
Menschlich zu prüfen	Ist das Nutzerziel klar beschrieben; ist der Nutzungskontext ausreichend bekannt; sind wichtige Zustände und Fehlerfälle berücksichtigt; ist das Systemfeedback verständlich und handlungsleitend; sind Accessibility-Basics relevant und angemessen berücksichtigt; sind Akzeptanzkriterien prüfbar; sind AI-Vorschläge fachlich korrekt, priorisiert und umsetzbar.

Entscheidend bleibt, dass das Team AI-Vorschläge bewusst übernimmt, anpasst oder begründet ausschließt.

5 Planung und Teamorganisation

Unterrichtszeit: ca. 90 Minuten

5.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt die organisatorische Verankerung von Usability im Softwareentwicklungsteam. Nachdem entsprechende Aspekte im Backlog Refinement benannt wurden, müssen sie in der Umsetzung geplant, bearbeitet, geprüft und verantwortet werden.

Usability entsteht nicht allein dadurch, dass ein Backlog Item entsprechende Kriterien enthält. Diese Kriterien müssen in konkrete Aufgaben, Zuständigkeiten, Prüffragen und Review-Formate überführt werden. Dafür braucht es ein gemeinsames Verständnis, welche Usability-Aufgaben das Team selbst übernehmen kann und wann zusätzliche Expertise erforderlich ist.

GenAI kann die Teamorganisation unterstützen, indem sie Usability-Aufgaben vorschlägt, Checklisten erzeugt, Design Critiques vorbereitet, Review-Fragen formuliert, Meeting-Ergebnisse strukturiert oder Entscheidungsoptionen zusammenfasst. AI darf jedoch nicht zur verdeckten Entscheidungsinstanz werden. Verantwortung für UX-Qualität, Priorisierung, fachliche Bewertung und Freigabe bleibt beim Team beziehungsweise bei den zuständigen Rollen.

Ziel dieses Kapitels ist es, Usability in der Umsetzung nicht als diffuse Erwartung an „gutes Design“ zu behandeln, sondern als planbare, prüfbare und verantwortete Teamarbeit.

5.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 5.1 bis BO 5.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

5.3 Learning Objectives

LO	Learning Objective	K
LO 5.1	Typische Usability-Aufgaben in der Umsetzung benennen können, etwa Usability-Check, Design Critique, Systemzustandsprüfung, Formulartest, Accessibility-Basic-Check und Vorbereitung von Review-Formaten oder Prüffragen.	K1

LO	Learning Objective	K
LO 5.2	Erklären können, warum Usability-Aufgaben in der Planung sichtbar gemacht und nicht nur implizit erwartet werden sollten.	K2
LO 5.3	Die Beiträge typischer Rollen in agilen Softwareentwicklungsteams zur UX-Qualität beschreiben können, etwa Product Owner, Entwicklung, Testerinnen und Tester, Qualitätssicherung, Scrum Master, agile Coaches und gegebenenfalls UX-Spezialistinnen und UX-Spezialisten.	K2
LO 5.4	Einfache Usability-Aktivitäten in der Planung der Umsetzung organisatorisch einplanen können.	K3
LO 5.5	Erklären können, wo agile Softwareentwicklungsteams einfache Usability-Aufgaben selbst übernehmen können und wo zusätzliche Expertise erforderlich ist.	K2
LO 5.6	Design Critiques als strukturiertes Format zur Bewertung von Entwürfen beschreiben können.	K2
LO 5.7	GenAI zur Vorbereitung von Usability-Meetings, Prüffragen, Checklisten und Entscheidungsoptionen einsetzen können.	K3
LO 5.8	Den Einsatz von AI in Usability-relevanten Teamaktivitäten nachvollziehbar dokumentieren können, insbesondere Zweck, Eingaben, übernommene Ergebnisse und menschliche Prüfung.	K3

5.4 Wichtige Begriffe

Sprint Planning, Usability-Aufgabe, Teamverantwortung, Product Owner, Entwicklerinnen und Entwickler, Testerinnen und Tester, Qualitätssicherung, Scrum Master, agile Coach, UX-Spezialistin bzw. UX-Spezialist, Design Critique, Usability-Check, heuristische Prüfung, heuristische Evaluation, Formulartest, Systemzustandsprüfung, Accessibility-Basic-Check, Prüffrage, Review-Format, Entscheidungsoption, Facilitation, Verantwortung, AI-gestützte Vorbereitung, Dokumentation, menschliche Prüfung.

5.5 Planung als Übergang von Backlog zu Teamarbeit

Agile Planung übersetzt ausgewählte Backlog Items in konkrete Arbeit für die Umsetzung. Sprint Planning ist ein verbreitetes Format dafür. Wenn Usability-Aspekte im Backlog sichtbar wurden, müssen sie in Aufgaben, Zuständigkeiten und Prüfungen überführt werden. Andernfalls bleiben Usability-Kriterien formal vorhanden, werden aber in der Umsetzung nicht aktiv bearbeitet.

Agile Planung ist daher nicht nur Aufwandsschätzung und technische Zerlegung. Sie ist auch die Stelle, an der Usability-Arbeit sichtbar gemacht wird. Ein Akzeptanzkriterium zu verständlichen Fehlermeldungen sollte beispielsweise zu Aufgaben führen: Fehlermeldung formulieren, implementieren, testen und bei Bedarf aus Usability-Sicht prüfen.

UX-Qualität entsteht in agilen Teams durch mehrere Rollen und Tätigkeiten. Product Owner beeinflussen UX durch Zielklärung, Priorisierung und Akzeptanzkriterien. Entwicklerinnen und Entwickler beeinflussen UX durch Interaktionslogik, Zustände, Komponenten, Fehlermeldungen, Performance und Accessibility. Testerinnen, Tester und Qualitätssicherung machen Usability-Probleme durch Prüfung von Verhalten, Randfällen, Verständlichkeit und Fehlerzuständen sichtbar. Scrum Master und agile Coaches können helfen, Usability-Aktivitäten in Teamprozessen zu verankern. UX-Spezialistinnen und UX-Spezialisten, falls vorhanden, bringen methodisches, gestalterisches und evaluatives Fachwissen ein.

UX ist damit nicht nur Aufgabe einer einzelnen Rolle. Auch wenn spezialisierte UX-Kompetenz wertvoll ist, müssen alle Teammitglieder verstehen, wie ihre Entscheidungen die Nutzung beeinflussen.

5.6 Usability-Aufgaben in der Planung sichtbar machen

Usability-Aufgaben werden in vielen Teams implizit erwartet. Es wird angenommen, dass jemand „darauf achtet“, ob eine Oberfläche verständlich ist, ob Fehlermeldungen passen oder ob ein Formular gut bedienbar ist. Diese implizite Erwartung führt selten zu klarer Verantwortung.

Wenn Usability-Aufgaben nicht sichtbar geplant werden, entstehen typische Lücken: Fehlermeldungen werden spät formuliert, Empty States fehlen, Ladezustände werden nicht gestaltet, Tastaturbedienbarkeit wird nicht geprüft, Akzeptanzkriterien werden nur funktional getestet oder AI-generierte Entwürfe und Texte werden nicht kritisch beurteilt.

Typische Usability-Aufgaben in der agilen Umsetzung sind Usability-Checks, Design Critiques, heuristische Prüfungen, Formulartests, Systemzustandsprüfungen, Accessibility-Basic-Checks, Review-Vorbereitungen und die Prüfung von UI-Texten. Diese Aufgaben müssen nicht groß sein. Oft reichen kleine, klar beschriebene Tätigkeiten, um UX-Qualität deutlich zu verbessern.

Usability-Aufgaben sollten aus User Stories, Akzeptanzkriterien, Definition of Ready und Definition of Done abgeleitet werden. Wenn ein Akzeptanzkriterium einen verständlichen Fehlerzustand verlangt, können daraus eine Aufgabe zur Textformulierung, eine Implementierungsaufgabe, eine Testaufgabe und eine Prüffrage entstehen.

GenAI kann helfen, aus einem Backlog Item Usability-relevante Aufgaben abzuleiten. Sie kann vorschlagen, welche Usability-Prüfungen sinnvoll sind, welche Zustände berücksichtigt werden sollten

oder welche Prüffragen gestellt werden können. Die Vorschläge müssen jedoch an Sprintziel, Teamkapazität, technische Abhängigkeiten und Produktprioritäten angepasst werden.

5.7 Rollen und Verantwortung im Softwareentwicklungsteam

Usability ist nicht ausschließlich Aufgabe einer einzelnen Spezialrolle. Sie entsteht durch Entscheidungen in Product Ownership, Requirements, Design, Entwicklung, Testing, Qualitätssicherung und Betrieb. Eine UX-Spezialrolle kann wichtige Expertise einbringen, ersetzt aber nicht die Verantwortung des Teams für die konkrete Produktqualität.

Der Product Owner beeinflusst UX vor allem durch Zielklärung, Priorisierung und Akzeptanz. Backlog Items sollten nicht nur Funktionen beschreiben, sondern auch Nutzerziel, Kontext und relevante Qualitätskriterien benennen. Der Product Owner entscheidet mit, welche Usability-Probleme priorisiert und welche Risiken bewusst akzeptiert werden.

Entwicklerinnen und Entwickler beeinflussen UX durch die technische Umsetzung. Viele Usability-Eigenschaften entstehen unmittelbar im Code: Eingabevalidierung, Ladeverhalten, Fokusführung, Fehlermeldungen, Tastaturbedienbarkeit, Komponentenlogik, Responsiveness, Performance und Systemzustände. AI-gestützte Coding-Werkzeuge können unterstützen, müssen aber auf Usability-Anforderung, Architektur und Kontext geprüft werden.

Testerinnen, Tester und Qualitätssicherung können UX-Qualität nachweisbar machen, indem sie nicht nur funktionale Korrektheit prüfen, sondern auch Verständlichkeit, Systemzustände, Fehlerfälle, Eingabevalidierung, Tastaturbedienbarkeit und Nutzerführung betrachten. Ein Formular ist nicht nur dann zu prüfen, wenn ungültige Eingaben technisch abgewiesen werden, sondern auch danach, ob Fehlermeldung, Feldbezug, Korrekturhilfe und Erhalt bereits eingegebener Daten nutzbar sind.

Scrum Master und agile Coaches beeinflussen UX nicht durch fachliche Designentscheidungen, sondern durch die Arbeitsweise des Teams. Sie können darauf hinweisen, wenn Usability-Aufgaben regelmäßig ungeplant bleiben, Review-Formate zu meinungsbasiert verlaufen oder Lernschleifen aus Nutzerfeedback fehlen.

UX-Spezialistinnen und UX-Spezialisten, falls vorhanden, bringen methodische, gestalterische und evaluative Expertise ein. Sie können Discovery, Prototyping, Research, Evaluation, Usability-Tests, Accessibility oder Designsysteme fachlich vertiefen. In vielen Teams ist eine solche Rolle jedoch nicht dauerhaft verfügbar. Deshalb müssen agile Teams einfache Usability-Aktivitäten selbst übernehmen können und zugleich erkennen, wann zusätzliche Expertise erforderlich ist.

Im Zusammenhang mit GenAI wird Verantwortungsklärung besonders wichtig. AI kann Vorschläge erzeugen, Aufgaben strukturieren, Usability-Kriterien formulieren oder Entscheidungsgrundlagen vorbereiten. Sie übernimmt jedoch keine Verantwortung. Teams sollten daher klären, wer eine AI-gestützte Aufgabe operativ bearbeitet, wer fachlich entscheidet, wer einbezogen werden muss und wer informiert wird. Das RACI-Prinzip kann hierfür eine einfache Orientierung bieten; AI selbst übernimmt dabei keine Rolle in der Verantwortungsmatrix.

5.8 Einfache Usability-Aktivitäten organisatorisch einplanen

Entwicklungsteams können einfache Usability-Aktivitäten in Planung und Umsetzung integrieren, ohne daraus einen schwergewichtigen Zusatzprozess zu machen. Dazu gehören Design Critiques, heuristische Prüfungen anhand definierter Kriterien, Formulartests, Systemzustandsprüfungen, Accessibility-Basic-Checks und Review-Vorbereitungen.

Eine Design Critique ist ein strukturiertes Review-Format zur fachlichen Diskussion eines Entwurfs. Sie dient nicht dazu, persönliche Vorlieben zu sammeln, sondern einen Entwurf anhand nachvollziehbarer Usability-Kriterien zu prüfen, etwa Nutzerziel, Verständlichkeit, Erwartungskonformität, Konsistenz, relevante Zustände, Accessibility und Risiken.

Eine heuristische Prüfung betrachtet eine Oberfläche anhand definierter Usability-Prinzipien oder Gestaltungskriterien. Im Sprint kann eine vereinfachte heuristische Prüfung helfen, grobe Usability-Probleme früh aufzudecken. Sie ersetzt keine vollständige heuristische Evaluation und keinen Usability-Test mit realen Nutzerinnen und Nutzern.

Ein Formulartest kann prüfen, ob Labels verständlich sind, Pflichtfelder erkennbar sind, Validierung sinnvoll erfolgt, Fehlermeldungen helfen und Eingaben nach Fehlern erhalten bleiben. Eine Systemzustandsprüfung betrachtet relevante Zustände wie Loading, Empty, Error, Success, Disabled, fehlende Berechtigung, unvollständige Daten, Netzwerkfehler oder abgebrochene Aktionen.

Ein Accessibility-Basic-Check prüft ausgewählte Anforderungen, die für viele Interfaces relevant sind, etwa Tastaturbedienbarkeit, sichtbarer Fokus, programmatisch zuordenbare Labels, ausreichender Kontrast, semantische Struktur und verständliche Fehlermeldungen. Er ersetzt keine vollständige Prüfung nach WCAG oder EN 301 549, hilft aber, häufige Barrieren früh zu erkennen.

UX sollte auch in Review-Formaten nicht nur beiläufig erwähnt werden. Eine Review-Vorbereitung kann klären, welches Nutzerziel unterstützt wird, welche Zustände umgesetzt wurden, welche Fehlerfälle geprüft wurden, welche Annahmen offen bleiben und welche Nutzerreaktionen später validiert werden müssen.

5.9 Wo Teams selbst handeln können und wo Expertise notwendig ist

Agile Softwareentwicklungsteams können viele einfache Usability-Aufgaben selbst übernehmen, wenn sie grundlegende Kriterien kennen. Dazu gehören erste Usability-Checks, vereinfachte heuristische Prüfungen, Formulartests, verständliche Fehlermeldungen, Systemzustandsprüfungen, Accessibility-Basic-Checks, Design Critiques und die kritische Bewertung AI-generierter Vorschläge.

Diese Tätigkeiten erfordern keine vollständige UX-Spezialisierung. Sie erfordern aber ein gemeinsames Basisverständnis, klare Kriterien und die Bereitschaft, UX nicht als Geschmackssache zu behandeln.

Zusätzliche Expertise wird erforderlich, wenn Unsicherheit, Risiko oder Komplexität hoch sind. Das betrifft etwa umfangreiche Nutzerforschung, sicherheitskritische Systeme, komplexe Fachdomänen, rechtliche Fragen, tiefergehende Accessibility-Prüfungen, strategische Usability-Entscheidungen, Designsystem-Governance oder kritische Conversion-Optimierung.

Ob ein Team selbst handeln kann oder Expertise hinzuziehen sollte, hängt von mehreren Faktoren ab: möglichen Folgen einer Fehlbedienung, Spezialisierung der Domäne, Unsicherheit über Nutzerbedürfnisse, rechtlichen oder regulatorischen Anforderungen, Hinweisen auf schwerwiegende Nutzungsprobleme, Erfahrung des Teams mit der Methode und Bedarf an Validierung mit realen Nutzerinnen und Nutzern.

AI kann diese Entscheidung vorbereiten, etwa durch Checklisten oder Risikofragen. Die Entscheidung bleibt organisatorisch und fachlich beim Team beziehungsweise bei der Organisation.

5.10 Design Critiques als strukturiertes Format

Design Critiques helfen Teams, UI-Entwürfe, Prototypen oder umgesetzte Funktionen strukturiert aus Sicht von Usability, Accessibility und erwartbarem Verhalten zu bewerten, bevor sie umgesetzt oder weiterentwickelt werden. Sie sind besonders wichtig, wenn AI schnell mehrere Varianten erzeugt. Ohne Kriterien besteht die Gefahr, dass die visuell überzeugendste Variante gewählt wird, obwohl sie aus Nutzungsperspektive nicht die geeignetste ist.

Eine Design Critique fragt nicht nur, ob ein Entwurf gefällt. Sie prüft, ob der Entwurf das Nutzerziel verständlich, erwartungskonform, konsistent, fehlertolerant und zugänglich unterstützt.

Eine gute Design Critique verwendet klare Bewertungskriterien. Dazu gehören Nutzerziel, Nutzungskontext, visuelle Hierarchie, Verständlichkeit, Konsistenz, kognitive Belastung, relevante Zustände, Fehlervermeidung, Accessibility-Basics und erwartbares Verhalten.

In einer Design Critique können unterschiedliche Rollen beitragen. Product Owner achten auf Nutzerwert und Priorität. Entwicklerinnen und Entwickler prüfen technische Umsetzbarkeit und Komponentenlogik. Testerinnen und Tester achten auf prüfbare Zustände und Fehlerfälle. UX-Spezialistinnen und UX-Spezialisten, falls vorhanden, prüfen Gestaltung, Interaktion und Methode. Scrum Master, agile Coaches oder Personen mit Verantwortung für Teamprozesse können den Ablauf strukturieren.

Eine Design Critique sollte zu klaren Ergebnissen führen: erkannte Probleme, notwendige Änderungen, offene Annahmen, spätere Prüfpunkte und getroffene Entscheidungen. AI kann helfen, Kriterien, Review-Fragen oder Zusammenfassungen vorzubereiten. Bewertung und Entscheidung bleiben beim Team.

5.11 GenAI zur Vorbereitung und Strukturierung von Teamaktivitäten

GenAI kann Teamaktivitäten zu Usability vorbereiten und strukturieren, wenn Ziel, Kontext und angestrebtes Ergebnis klar definiert sind. Dazu gehören Design Critiques, Refinement-Sessions, Review-Vorbereitungen, Retrospektiven oder Entscheidungsworkshops.

AI kann Agenden, Leitfragen, Checklisten und Rollenhinweise vorschlagen. Aus einem Backlog Item oder Prototyp kann AI Review-Fragen generieren, etwa zu fehlenden Zuständen, ungültigen Eingaben, nächstem Schritt, Erfolgsmeldung, Tastaturbedienbarkeit oder offenen Annahmen. Solche Checklisten sollten kurz und kontextbezogen bleiben, weil zu lange AI-generierte Listen Meetings überfrachten können.

AI kann auch Entscheidungsoptionen strukturieren. Wenn mehrere UI-Varianten oder Lösungswege vorliegen, kann AI Vor- und Nachteile, Risiken, offene Fragen und mögliche Bewertungskriterien zusammenfassen. Sie liefert damit Struktur für eine Entscheidung, entscheidet aber nicht selbst.

AI kann außerdem Meeting-Ergebnisse, Findings oder Entscheidungen zusammenfassen. Diese Zusammenfassungen müssen geprüft werden. Besonders bei Entscheidungen ist wichtig, dass AI keine nicht getroffenen Beschlüsse ergänzt, keine Unsicherheiten verschweigt und keine Hypothesen als Fakten formuliert.

5.12 Dokumentation des AI-Einsatzes

Wenn AI in Usability-relevanten Teamaktivitäten eingesetzt wird, sollte nachvollziehbar bleiben, wo und wofür sie genutzt wurde. Dies ist wichtig, weil AI-Ergebnisse später in Anforderungen, Designs, Code, Tests oder Produktentscheidungen einfließen können.

Ohne Dokumentation kann unklar werden, ob eine Aussage aus Nutzerdaten, Fachwissen, Teamentscheidung oder einem AI-Vorschlag stammt. Dadurch steigt das Risiko, dass AI-generierte Inhalte später als gesicherte Grundlage behandelt werden.

Die Dokumentation sollte zweckmäßig und leichtgewichtig bleiben. Wichtig sind Zweck des AI-Einsatzes, verwendete Eingaben oder Datenquellen, wesentliche AI-Ergebnisse, übernommene Ergebnisse, verworfene oder angepasste Vorschläge, menschliche Prüfung sowie offene Annahmen oder Risiken.

Dokumentation unterstützt AI Governance. Sie zeigt, wie AI im Team verwendet wurde, welche Verantwortung beim Menschen blieb und wie Qualität geprüft wurde. Bei AI-gestützten Usability-Aktivitäten sollte außerdem festgehalten werden, wer die AI-Ausgabe erstellt oder vorbereitet hat, wer sie fachlich geprüft hat und wer die Entscheidung über Übernahme, Anpassung oder Ausschluss getroffen hat.

Für agile Teams ist dies besonders relevant, wenn AI-Ausgaben in Backlog Items, Akzeptanzkriterien, UI-Entwürfe, Tests oder Code einfließen. Je näher ein AI-Ergebnis an produktive Umsetzung oder fachliche Entscheidung gelangt, desto wichtiger wird Nachvollziehbarkeit.

5.13 AI-gestützte Teamorganisation und menschliche Prüfung

AI kann in der Teamorganisation besonders gut bei Vorbereitung, Strukturierung und Dokumentation unterstützen.

AI unterstützt gut bei	Usability-Aufgaben aus Backlog Items ableiten; Checklisten erstellen; Review-Fragen formulieren; Design-Critique-Kriterien strukturieren; Entscheidungsoptionen zusammenfassen; Meeting-Ergebnisse dokumentieren; AI-Einsatz nachvollziehbar protokollieren.
Grenzen und typische Risiken	Übernimmt keine Teamverantwortung und entscheidet nicht über Rollen, Prioritäten oder Konflikte; kennt Teamdynamik, Organisationskultur, Kapazitäten und politische Rahmenbedingungen nicht ausreichend; typische Risiken: ungeprüft übernommene Checklisten, überladene Planungs- oder Review-Formate, unklare Verantwortlichkeiten, ungenaue Zusammenfassungen, als Beschlüsse dokumentierte Hypothesen, nicht dokumentierter AI-Einsatz, zu spät hinzugezogene externe Expertise.
Menschlich zu prüfen	Welche Usability-Aufgaben sich aus den Backlog Items ergeben; ob diese Aufgaben in der agilen Planung eingeplant sind; wer welche

	Verantwortung übernimmt; welche Usability-Prüfungen für das Umsetzungsziel notwendig sind; welche AI-generierten Checklisten und Review-Fragen relevant sind; welche Entscheidungen tatsächlich getroffen wurden; ob zusätzliche Usability-, Accessibility-, Research- oder Domänenexpertise erforderlich ist.

AI kann Struktur liefern, aber keine Zuständigkeit, Priorität oder fachliche Entscheidung übernehmen.

6 UI, Informationsarchitektur, States und AI-generierte Entwürfe

Unterrichtszeit: ca. 135 Minuten

6.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt die Gestaltung und frühe Konkretisierung von Lösungen in der Umsetzung. Im Mittelpunkt stehen UI-Entwürfe, Informationsarchitektur, Navigation, Interaktionsabläufe, Systemzustände, Prototypen und AI-generierte Vorschläge für Benutzeroberflächen. Diese Artefakte werden im Syllabus nicht als reine Design-Showcases verstanden, sondern als Mittel, um Usability und Interaktionsqualität sichtbar, diskutierbar, prüfbar und bei Bedarf testbar zu machen.

Dabei ist begrifflich zu unterscheiden: Software-Usability bezeichnet die Gebrauchstauglichkeit von Software in einem bestimmten Nutzungskontext. UX bezeichnet die gesamte Nutzungserfahrung eines Produkts oder Systems. UI bezeichnet die konkrete Benutzerschnittstelle mit Screens, Formularen, Navigation, Controls, Texten und Interaktionselementen. UI-Design gestaltet diese Schnittstelle. Frontend bezeichnet die technische Umsetzung der sichtbaren und bedienbaren Teile eines digitalen Systems. Gute UI-Entwürfe sind daher nicht nur visuell zu beurteilen, sondern danach, ob sie Nutzerziele, Nutzungskontext, Verständlichkeit, erwartbares Verhalten, Accessibility und relevante Systemzustände unterstützen.

GenAI kann Wireframes, Mockups, Screen-Varianten, Interface-Texte, Navigationsmodelle, Informationsstrukturen, klickbare Prototypen oder frontend-nahe Strukturen erzeugen. Dadurch wird der Abstand zwischen Idee, Entwurf und testbarer Lösung kleiner. Gleichzeitig können AI-generierte Entwürfe professionell und vollständig wirken, obwohl zentrale Usability-Fragen ungeklärt bleiben.

Ziel dieses Kapitels ist es, AI-generierte UI-Entwürfe und Prototypen anhand grundlegender Usability- und Human-Factors-Kriterien kritisch zu beurteilen. Dazu gehören Nutzerziel, Nutzungskontext, menschliche Wahrnehmung, visuelle Hierarchie, mentale Modelle, Verständlichkeit, kognitive Belastung, Konsistenz, Accessibility und erwartbares Verhalten.

6.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 6.1 bis BO 6.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

6.3 Learning Objectives

LO	Learning Objective	K
LO 6.1	Grundlegende Begriffe des UI-Designs wie visuelle Hierarchie, Layout, Spacing, Typografie, Konsistenz, Interaktion, State und Control wiedergeben können.	K1
LO 6.2	Erklären können, warum UI-Design-Grundlagen und Human-Factors-Kriterien notwendig sind, um AI-generierte Entwürfe beurteilen zu können.	K2
LO 6.3	Die Bedeutung von Informationsarchitektur, Navigation, Benennungen, mentalen Modellen und Auffindbarkeit für Usability erklären können.	K2
LO 6.4	Typische Prototyp-Arten und deren Zweck benennen können, insbesondere Wireframe, Mockup, Low-Fidelity-Prototyp, High-Fidelity-Prototyp, klickbarer Prototyp, funktionsnaher Prototyp, horizontaler Prototyp, vertikaler Prototyp und Szenario-Prototyp.	K1
LO 6.5	Beschreiben können, wie AI-gestütztes Rapid Prototyping in agilen Entwicklungsprozessen eingesetzt werden kann.	K2
LO 6.6	AI-generierte UI-Entwürfe anhand von Nutzerziel, Nutzungskontext, menschlicher Wahrnehmung, mentalen Modellen, visueller Hierarchie, Verständlichkeit, kognitiver Last, Konsistenz, Accessibility und erwartbarem Verhalten beurteilen können.	K3
LO 6.7	Typische Stärken von AI bei bekannten UI-Mustern erklären können, etwa Login, Formular, Dashboard, Navigation, Suche, Filter oder Onboarding.	K2
LO 6.8	Typische Schwächen AI-generierter Entwürfe erkennen können, insbesondere fehlende States, Edge Cases, Fehlerpfade, Berechtigungen, alternative Nutzungssituationen und unvollständige Accessibility-Berücksichtigung.	K3
LO 6.9	AI-generierte UI-Varianten anhand definierter Usability-Kriterien vergleichen und begründet auswählen können	K3
LO 6.10	Prototypen als Grundlage für Kommunikation, Lernen, Evaluation und gegebenenfalls Validierung im Team einsetzen können.	K3

6.4 Wichtige Begriffe

Software-Usability, User Experience, UX-Qualität, Interaction Capability, Quality in Use, UI, User Interface, UI-Design, Interaction Design, visuelle Hierarchie, Layout, Spacing, Typografie, Lesbarkeit,

Konsistenz, Interaktion, Informationsarchitektur, Navigation, Benennung, Auffindbarkeit, Tree Testing, mentales Modell, Erwartungskonformität, kognitive Last, Gestaltgesetze, Wireframe, Mockup, Prototyp, Low-Fidelity-Prototyp, High-Fidelity-Prototyp, klickbarer Prototyp, funktionsnaher Prototyp, horizontaler Prototyp, vertikaler Prototyp, Szenario-Prototyp, Rapid Prototyping, State, Loading State, Empty State, Error State, Success State, Disabled State, Happy Path, Edge Case, alternativer Nutzungspfad, UI-Pattern, Onboarding, AI-generierter Entwurf, frontend-naher Vorschlag, Human-Factors-Kriterien, Accessibility, menschliche Prüfung.

6.5 Vom Backlog Item zur sichtbaren Lösung

Der Schritt vom Backlog Item zur sichtbaren Lösung ist zugleich ein Schritt von abstrakten Anforderungen zu konkreter Usability: Nutzerziele, Systemreaktionen, Benennungen, Zustände, Fehlerpfade und Bedienbarkeit werden erstmals als konkrete Interaktion sichtbar.

Nach Discovery, Refinement und agiler Planung liegen idealerweise ein klares Nutzerziel, ein beschriebener Nutzungskontext, relevante Akzeptanzkriterien und sichtbare Usability-Aufgaben vor. In der Umsetzung werden diese Grundlagen in konkrete Lösungsideen übersetzt. Dazu gehören UI-Entwürfe, Informationsstrukturen, Navigationsmodelle, Interaktionsabläufe, Systemzustände, Prototypen und erste technische Umsetzungsüberlegungen.

Design und Prototyping sind dabei nicht als isolierte Designphase zu verstehen. Sie dienen dazu, Annahmen offenzulegen. Ein Backlog Item kann textlich plausibel wirken, aber erst ein Entwurf zeigt, ob die Lösung verständlich, vollständig, auffindbar und erwartungskonform erscheint.

AI kann diesen Schritt beschleunigen, indem sie aus Anforderungen, User Stories, Akzeptanzkriterien oder Beschreibungen erste UI-Entwürfe und Prototypen erzeugt. Dadurch können Teams schneller diskutieren, vergleichen und prüfen. Ein AI-generierter Entwurf ist jedoch zunächst ein Vorschlag, der anhand von Nutzerziel, Nutzungskontext, Usability-Kriterien, Human-Factors-Kriterien und fachlichen Anforderungen geprüft werden muss.

Ein UI-Entwurf macht sichtbar, welche Informationen hervorgehoben werden, welche Aktionen angeboten werden, welche Begriffe verwendet werden, welche Reihenfolge ein Ablauf hat und wie Nutzerinnen und Nutzer durch eine Aufgabe geführt werden. Dadurch wird er zum Prüfobjekt für UX-Qualität.

Prototypen verbinden Teamdiskussion, Evaluation und gegebenenfalls Validierung. Sie können für interne Diskussionen, Design Critiques, Stakeholder-Abstimmung, Expert Reviews oder Usability-Tests genutzt werden. Eine Validierung entsteht jedoch erst dann, wenn eine konkrete Annahme methodisch

angemessen geprüft wird, etwa durch reale Nutzerinnen und Nutzer, Domänenexpertinnen und Domänenexperten, Usability-Tests, Nutzungsdaten oder andere belastbare Quellen.

6.6 UI-Design-Grundlagen zur Beurteilung von Entwürfen

UI-Design-Grundlagen werden in diesem Syllabus nicht als ästhetisches Spezialwissen verstanden, sondern als Grundlage zur Beurteilung von Usability und Interaktionsqualität. Visuelle Hierarchie, Layout, Spacing, Typografie, Konsistenz und Lesbarkeit beeinflussen, ob Nutzer Informationen wahrnehmen, Aufgaben verstehen, Aktionen korrekt ausführen und Fehler vermeiden können.

Visuelle Hierarchie beschreibt, welche Elemente einer Oberfläche zuerst wahrgenommen werden und welche Bedeutung ihnen zugeschrieben wird. Größe, Position, Kontrast, Abstand, Farbe, Typografie, Gruppierung und Bewegung beeinflussen, worauf Nutzerinnen und Nutzer achten. AI-generierte Entwürfe können visuell modern wirken, aber dennoch eine schlechte visuelle Hierarchie haben, etwa wenn mehrere Aktionen gleich stark hervorgehoben werden.

Layout und Spacing strukturieren Inhalte. Abstände zeigen, welche Elemente zusammengehören und welche voneinander getrennt sind. Ein konsistentes Spacing-System unterstützt Orientierung und reduziert visuelle Unruhe. Dabei sind grundlegende Gestaltgesetze relevant, etwa Nähe, Ähnlichkeit, gemeinsame Region, Geschlossenheit und gute Gestalt. Elemente, die nah beieinanderstehen oder ähnlich gestaltet sind, werden häufig als zusammengehörig wahrgenommen.

Typografie beeinflusst, ob Informationen schnell und sicher gelesen werden können. Relevante Aspekte sind Schriftgröße, Zeilenlänge, Zeilenabstand, Schriftgewicht, Kontrast, Informationsdichte und klare Unterscheidung unterschiedlicher Informationsebenen. Besonders bei Fehlermeldungen, rechtlichen Hinweisen, Tabellen, Dashboards oder komplexen Fachinformationen ist Lesbarkeit ein Qualitätskriterium.

Konsistenz bedeutet, dass ähnliche Dinge ähnlich aussehen und ähnlich funktionieren. Gleiche Aktionen sollten gleich benannt werden, gleiche Zustände gleich dargestellt werden und wiederkehrende Komponenten sich erwartbar verhalten. AI kann in einzelnen Screens gute Varianten erzeugen, aber über mehrere Screens hinweg Inkonsistenzen einführen.

Interaktion beschreibt, wie Nutzerinnen und Nutzer mit dem System handeln und wie das System reagiert. Dazu gehören Klicks, Eingaben, Auswahl, Navigation, Bestätigung, Abbruch, Systemfeedback und Fehlerbehandlung. Ein statischer UI-Entwurf zeigt oft nur, wie etwas aussieht. Gute UX erfordert zusätzlich, dass Interaktionsabläufe, Systemzustände und Rückmeldungen explizit betrachtet werden.

Controls bringen bekannte mentale Modelle mit. Radio Buttons stehen typischerweise für eine Auswahl aus gegenseitig ausschließenden Optionen, Checkboxes für mehrere unabhängige Auswahlmöglichkeiten, Toggle für unmittelbares Ein- oder Ausschalten, Buttons für Aktionen, Links für Navigation und Tabs für zusammengehörige Inhaltsbereiche. Wenn Controls anders verwendet werden, als Nutzerinnen und Nutzer es erwarten, entstehen Irritation, Fehlbedienung oder zusätzliche kognitive Belastung.

6.7 Informationsarchitektur, Navigation und Auffindbarkeit

Informationsarchitektur, Navigation und Auffindbarkeit sind zentrale Bestandteile von Usability. Sie bestimmen, ob Nutzer relevante Inhalte, Funktionen und nächste Schritte finden und ob die Struktur eines Systems zu ihren mentalen Modellen und Aufgaben passt.

Informationsarchitektur beschreibt, wie Inhalte, Funktionen und Objekte strukturiert werden. Sie beeinflusst, ob Nutzerinnen und Nutzer finden, was sie brauchen, und ob sie verstehen, wo sie sich im System befinden. Usability-Probleme entstehen häufig nicht durch einzelne schlechte Screens, sondern durch unklare Struktur, unerwartete Kategorien oder uneinheitliche Begriffe.

Ein mentales Modell beschreibt, wie Nutzerinnen und Nutzer glauben, dass ein System, ein Prozess oder ein fachlicher Gegenstand aufgebaut ist und funktioniert. Für Informationsarchitektur ist dieses Konzept zentral: Nutzer suchen Funktionen und Informationen dort, wo sie diese aufgrund ihres mentalen Modells erwarten. Wenn die Struktur einer Anwendung vor allem der internen Systemlogik folgt, kann sie für das Entwicklungsteam plausibel sein, für Nutzerinnen und Nutzer aber schwer verständlich wirken.

Navigation hilft Nutzerinnen und Nutzern, sich durch ein System zu bewegen. Gute Navigation beantwortet Fragen wie: Wo bin ich? Was kann ich hier tun? Wie komme ich zurück? Was ist der nächste sinnvolle Schritt? Welche Bereiche gehören zusammen? Welche Funktion gehört zu welcher Aufgabe?

Benennungen beeinflussen Auffindbarkeit und Verständnis. Ein Begriff, der im Entwicklungsteam selbstverständlich ist, kann für Nutzerinnen und Nutzer unklar sein. Umgekehrt kann Fachsprache in spezialisierten Domänen notwendig sein. AI kann alternative Benennungen vorschlagen und Begriffe vereinfachen. Das ist nützlich, aber riskant, wenn Fachsprache präzise gebraucht wird.

Auffindbarkeit beschreibt, ob Nutzerinnen und Nutzer relevante Funktionen, Informationen oder Objekte finden können. Sie hängt von Informationsarchitektur, Navigation, Suche, Filterung, Benennungen, visueller Hierarchie und Nutzungskontext ab. AI kann mögliche Probleme aufzeigen und Prüffragen

vorbereiten. Belastbar wird die Einschätzung erst, wenn sie mit Nutzerwissen, Domänenwissen oder empirischen Hinweisen abgeglichen wird.

Eine einfache Methode zur Prüfung von Informationsarchitektur und Navigation ist Tree Testing. Dabei wird keine fertige Oberfläche getestet, sondern eine reduzierte, textbasierte Navigationsstruktur. Testpersonen erhalten typische Aufgaben und geben an, wo sie die gesuchte Information oder Funktion erwarten würden. AI kann Aufgaben, alternative Navigationslabels oder erste Ergebnisstrukturen vorbereiten, ersetzt aber keine Prüfung mit geeigneten Personen.

6.8 Prototyp-Arten und deren Zweck

Prototypen dienen nicht dazu, möglichst früh eine scheinbar fertige Lösung zu zeigen. Sie helfen, Annahmen über Struktur, Ablauf, Verständlichkeit, Interaktion, Systemzustände und Nutzungskontext sichtbar und überprüfbar zu machen.

Ein Prototyp ist eine vorläufige Darstellung oder Umsetzung einer Lösung. Er dient dazu, Ideen sichtbar, diskutierbar, prüfbar oder testbar zu machen. Sein Wert hängt nicht davon ab, wie fertig er wirkt, sondern davon, ob er zur jeweiligen Fragestellung passt.

Wireframes zeigen Struktur und grundlegende Anordnung, ohne visuelle Details in den Vordergrund zu stellen. Sie eignen sich, um Informationsarchitektur, Inhalte, Reihenfolge und grundlegende Interaktion zu diskutieren.

Mockups zeigen ein visuell ausgearbeiteteres Bild einer Oberfläche. Sie enthalten meist Farben, Typografie, Abstände und konkrete UI-Elemente. Sie eignen sich, um visuelle Hierarchie, Konsistenz, Lesbarkeit und Verständlichkeit zu diskutieren.

Low-Fidelity-Prototypen haben einen geringen Detailgrad. Sie eignen sich für frühe Diskussionen, Strukturfragen, erste Varianten und grundlegende Abläufe. High-Fidelity-Prototypen ähneln stärker dem späteren Produkt und eignen sich, wenn konkrete Abläufe, visuelle Hierarchie, Verständlichkeit oder realitätsnähere Nutzungssituationen geprüft werden sollen.

Horizontale Prototypen zeigen viele Bereiche oder Screens eines Systems in der Breite, arbeiten einzelne Funktionen aber nur oberflächlich aus. Vertikale Prototypen arbeiten einen ausgewählten Funktionsbereich oder Ablauf tiefer aus. Szenario-Prototypen bilden einen konkreten Nutzungspfad oder eine typische Nutzungssituation ab.

Klickbare Prototypen machen Abläufe erlebbar. Nutzerinnen und Nutzer oder Teammitglieder können durch Screens navigieren und erste Interaktionspfade nachvollziehen. Funktionsnahe Prototypen

verhalten sich stärker wie echte Anwendungen und können Daten, einfache Logik, Formulare oder Interaktionen enthalten.

AI erleichtert es, schnell detaillierte oder funktionsnahe Prototypen zu erzeugen. Das bedeutet aber nicht, dass immer ein komplexer Prototyp sinnvoll ist. Der Detailgrad sollte vom Zweck abhängen: Erst klären, welche Frage beantwortet werden soll, dann den passenden Prototyp wählen.

6.9 AI-gestütztes Rapid Prototyping in der Umsetzung

AI-gestütztes Rapid Prototyping kann aus Anforderungen, User Stories, Akzeptanzkriterien, Skizzen oder Textbeschreibungen schnell erste UI-Entwürfe und Prototypen erzeugen. Dadurch können Teams früher sehen, wie eine Lösung aussehen und funktionieren könnte.

Je schneller AI von einer Idee zu einem scheinbar fertigen Prototyp führt, desto wichtiger wird die bewusste Prüfung durch das Team. Geschwindigkeit ersetzt keine Klärung von Nutzerziel, Nutzungskontext, Zuständen, Fehlerpfaden, Accessibility und fachlicher Richtigkeit.

Der Vorteil liegt nicht nur in der Geschwindigkeit. Entwürfe machen Annahmen sichtbar. Sie zeigen, welche Informationen benötigt werden, welche Schritte vorgesehen sind, welche Begriffe verwendet werden, welche Systemzustände berücksichtigt wurden und welche Zustände möglicherweise fehlen.

AI kann mehrere Varianten eines Screens oder Ablaufs erzeugen. Varianten können unterschiedliche Schwerpunkte setzen: mehr Führung, weniger Schritte, stärkerer Fokus auf Daten, andere Navigation, andere Sprache, andere Struktur oder andere Interaktionsmuster. Die Auswahl einer Variante sollte anhand definierter Kriterien erfolgen, etwa Nutzerziel, Nutzungskontext, Verständlichkeit, visuelle Hierarchie, kognitive Belastung, Konsistenz, Accessibility, erwartbares Verhalten und fachliche Korrektheit.

Prompt-to-UI-, Prompt-to-App- und Prompt-to-Frontend-Werkzeuge können UI-Entwürfe, frontend-nahe Strukturen oder funktionsnahe App-Prototypen erzeugen. Dadurch rückt Prototyping näher an Umsetzung und Code. Je näher ein AI-Werkzeug an lauffähigen Code heranrückt, desto wichtiger werden Requirements-Prüfung, Usability-Prüfung, Accessibility-Prüfung, Code Review, Security-Prüfung und fachliche Abnahme.

Rapid Prototyping ist besonders wertvoll, wenn klar ist, welche Hypothese oder Fragestellung geprüft werden soll. Ohne klare Fragestellung erzeugt AI zwar Varianten, aber nicht automatisch bessere Produktentscheidungen.

6.10 AI-generierte UI-Entwürfe beurteilen

AI-generierte UI-Entwürfe sind nicht danach zu beurteilen, ob sie professionell aussehen, sondern ob sie die beabsichtigte Nutzung unterstützen. Entscheidend ist, ob Nutzerziel, Nutzungskontext, Informationsstruktur, Verständlichkeit, erwartbares Verhalten, Accessibility, Systemzustände und fachliche Regeln ausreichend berücksichtigt sind.

Der wichtigste Prüfpunkt ist das Nutzerziel. Ein UI-Entwurf soll nicht nur Funktionen darstellen, sondern ein Ziel unterstützen. Wenn unklar ist, welches Ziel Nutzerinnen und Nutzer erreichen sollen, kann auch der Entwurf nicht sinnvoll bewertet werden. Ebenso ist zu prüfen, ob das angenommene Nutzerziel belegt oder nur vermutet ist.

Ein UI-Entwurf kann nur im Zusammenhang mit Nutzungskontext und Aufgabe beurteilt werden. Eine Oberfläche, die für gelegentliche Nutzer gut geeignet ist, kann für erfahrene Fachnutzer zu langsam oder zu erklärend sein. Eine geführte Schrittfolge kann Einsteiger unterstützen, aber in einer häufig wiederholten Fachaufgabe ineffizient sein.

AI-generierte Entwürfe müssen darauf geprüft werden, ob wichtige Informationen wahrnehmbar sind. Entscheidend sind Kontrast, Größe, Position, Gruppierung, Abstände, Reihenfolge und visuelle Gewichtung. Die Prüfung sollte fragen, was zuerst auffällt, ob dies das Wichtigste ist, ob zusammengehörige Elemente erkennbar gruppiert sind und ob die visuelle Reihenfolge der Aufgabe entspricht.

Ein Entwurf sollte Nutzerinnen und Nutzer nicht unnötig zum Nachdenken zwingen. Texte, Benennungen, Reihenfolge, Gruppierung, Interaktionslogik und Informationsdichte sollten verständlich sein. Kognitive Last entsteht durch zu viele Optionen, unklare Begriffe, fehlende Orientierung, ungewohnte Muster, Gedächtnisanforderungen, unklare Systemzustände oder widersprüchliche Signale.

Nutzerinnen und Nutzer erwarten, dass bekannte Elemente erwartbar funktionieren. Ein Link verhält sich anders als ein Button. Eine Suche sollte nachvollziehbare Suchergebnisse liefern. Ein Filter sollte sichtbar aktivierbar und zurücksetzbar sein. Ein Speichern-Button sollte nach erfolgreicher Aktion Rückmeldung geben.

AI-generierte Entwürfe müssen auch auf grundlegende Accessibility-Aspekte geprüft werden. Dazu gehören Kontrast, Fokus, Tastaturbedienbarkeit, verständliche Labels, semantische Struktur, ausreichende Schriftgröße, klare Fehlermeldungen und Informationen, die nicht allein über Farbe vermittelt werden. AI kann Hinweise liefern, aber keine vollständige Barrierefreiheit garantieren.

6.11 Bekannte UI-Muster als Stärke und Risiko von AI

Bekannte UI-Muster können Usability unterstützen, wenn sie zum Nutzungskontext passen. Sie können aber auch schaden, wenn sie nur vertraut wirken, fachliche Abläufe vereinfachen oder notwendige Ausnahmen verdecken.

AI ist besonders nützlich bei bekannten UI-Mustern, weil diese in vielen Beispielen dokumentiert sind. Login, Registrierung, Passwort-Reset, Formulare, Suche, Filter, Tabellen, Dashboards, Onboarding, Upload-Komponenten, Bestätigungsdialoge oder einfache Checkout-Prozesse folgen häufig wiederkehrenden Strukturen.

AI kann solche Muster schnell reproduzieren und Varianten erzeugen. Für Teams ist das hilfreich, weil sie nicht bei null beginnen müssen. Besonders in frühen Entwurfsphasen kann AI helfen, typische Elemente, mögliche Zustände, Standardabläufe und alternative Darstellungsformen aufzuzeigen.

Diese Stärke beruht jedoch auf Mustererkennung. AI erkennt häufige Lösungen und kombiniert sie zu plausiblen Vorschlägen. Daraus folgt nicht automatisch, dass ein vorgeschlagenes Muster für den konkreten Nutzungskontext geeignet ist.

Standardmuster können unpassend sein. Ein Onboarding-Flow kann für eine einfache App sinnvoll sein, aber in einer Fachanwendung mit erfahrenen Nutzern unnötig stören. Ein Wizard kann Einsteiger unterstützen, aber Expertinnen und Experten verlangsamen. Ein Dashboard kann attraktiv wirken, aber irrelevante Kennzahlen hervorheben.

Teams müssen daher prüfen, ob ein UI-Pattern zur Aufgabe, zur Nutzergruppe, zur Häufigkeit der Nutzung, zum Risiko, zur Accessibility und zum mentalen Modell der Nutzerinnen und Nutzer passt.

6.12 States, Edge Cases und alternative Pfade

States, Edge Cases und alternative Pfade sind zentrale Bestandteile von Usability. Sie entscheiden darüber, ob ein System auch außerhalb des idealen Happy Path verständlich, robust, fehlertolerant und vertrauenswürdig bleibt.

AI-generierte Entwürfe zeigen häufig den idealen Ablauf. Nutzerinnen und Nutzer geben korrekte Daten ein, das System reagiert schnell, Berechtigungen passen, Daten sind vorhanden und Aktionen gelingen. Dieser Happy Path ist wichtig, aber nicht ausreichend.

Viele Usability-Probleme entstehen in Nicht-Idealzuständen. Wenn solche Zustände fehlen, stehen Nutzerinnen und Nutzer ohne Orientierung da. Gerade Fehlerfälle, leere Datenlagen, Ladezeiten,

abgebrochene Prozesse oder fehlende Berechtigungen bestimmen häufig, ob ein System als verständlich und vertrauenswürdig erlebt wird.

Wichtige Systemzustände sind unter anderem:

- Loading, Empty, Error, Success und Disabled State,
- fehlende Berechtigung und unvollständige Daten,
- unterbrochene Verbindung und fehlgeschlagene Validierung,
- abgebrochene Aktion, aktive Auswahl und aktive Filter,
- nicht gespeicherte Bearbeitung.

Diese Zustände müssen nicht immer alle relevant sein, sollten aber bewusst geprüft werden.

Empty States sollten erklären, warum nichts angezeigt wird und welche nächste Aktion möglich ist. Error States sollten verständlich erklären, was passiert ist und was Nutzerinnen und Nutzer tun können. Berechtigungsfälle sollten erkennen lassen, warum eine Aktion nicht möglich ist und welche Alternative gegebenenfalls besteht.

Edge Cases sind besondere oder seltenere Situationen außerhalb des idealen Ablaufs. Dazu zählen etwa sehr lange Namen, ungewöhnliche Datenformate, fehlende Werte, doppelte Datensätze, abgelaufene Sessions, parallele Bearbeitung oder unerwartete Eingaben. AI kann Edge Cases vorschlagen, erkennt aber nicht zuverlässig, welche davon im konkreten Produkt relevant oder kritisch sind.

6.13 UI-Patterns, Onboarding und Hilfen

UI-Patterns, Onboarding und Hilfen sollen nicht als dekorative Ergänzungen verstanden werden. Sie unterstützen Usability, wenn sie Nutzer dabei helfen, Funktionen zu verstehen, Aufgaben sicher auszuführen und bei Bedarf Orientierung oder Unterstützung zu erhalten.

UI-Patterns sind bewährte Lösungen für wiederkehrende Interaktionsprobleme. Beispiele sind Tab-Navigation, Breadcrumbs, Suchfilter, Modal-Dialoge, Stepper, Cards, Toast-Meldungen oder Inline-Validierung. Sie helfen, konsistente und erwartbare Oberflächen zu gestalten.

AI kann UI-Patterns vorschlagen und erklären. Teams sollten jedoch prüfen, ob ein Pattern zum Nutzerziel, zur Aufgabe und zum Nutzungskontext passt. Ein bekanntes Pattern ist nicht automatisch geeignet. Ein Modal-Dialog kann für eine fokussierte Entscheidung hilfreich sein, aber stören, wenn Nutzer Informationen aus dem Hintergrund benötigen. Ein Stepper kann komplexe Prozesse strukturieren, aber erfahrene Nutzer verlangsamen.

Onboarding unterstützt Nutzerinnen und Nutzer beim Einstieg in ein System oder eine Funktion. AI kann Onboarding-Texte, Schrittfolgen, Tooltips oder Hilfekonzepte vorschlagen. Dabei ist zu prüfen, ob Onboarding tatsächlich notwendig ist oder ob das Interface selbst verständlicher gestaltet werden sollte.

Gutes Onboarding erklärt nicht nur Funktionen, sondern unterstützt Nutzerinnen und Nutzer beim Erreichen eines ersten sinnvollen Ziels. Wenn ein System nur durch ausführliche Erklärungen verständlich wird, kann dies ein Hinweis auf strukturelle Usability-Probleme sein.

AI schlägt häufig Hilfetexte oder Erklärungen vor. Diese können nützlich sein, aber zu viele Hinweise erhöhen die kognitive Belastung. Progressive Offenlegung bedeutet, Informationen dann zu zeigen, wenn sie gebraucht werden. Teams sollten prüfen, ob eine Erklärung notwendig ist oder ob Struktur, Benennung, visuelle Hierarchie oder Systemfeedback verbessert werden sollten.

6.14 Prototypen als Grundlage für Kommunikation, Lernen, Evaluation und Validierung

Prototypen legen Annahmen offen. Sie helfen Softwareentwicklungsteams, früh über Nutzerziel, Ablauf, Informationsstruktur, Zustände, Fehlerfälle und Umsetzbarkeit zu sprechen, bevor Entscheidungen teuer werden oder AI-generierte Vorschläge vorschnell in Code überführt werden.

Prototypen machen abstrakte Anforderungen greifbar. Sie helfen Missverständnisse zu erkennen, Varianten zu vergleichen und Annahmen über Struktur, Ablauf, Sprache oder Interaktion zu konkretisieren. Für die Teamkommunikation ist wichtig, dass ein Prototyp als Arbeitsmittel verstanden wird, nicht als endgültige Lösung.

Prototypen unterstützen auch Lernen. Wenn ein Prototyp geprüft wird, werden Fragen nach Nutzerziel, Verständlichkeit, Systemzuständen, kognitiver Last, mentalen Modellen und Accessibility konkret. Teams lernen nicht dadurch, dass AI eine Lösung erzeugt, sondern dadurch, dass sie AI-Ergebnisse anhand von Usability-Kriterien beurteilen und mit Nutzungskontext, Anforderungen und fachlichen Regeln abgleichen.

Prototypen können genutzt werden, um Annahmen zu prüfen. Dafür muss klar sein, welche Annahme oder Untersuchungsfrage geprüft werden soll. Ein Prototyp ohne Fragestellung erzeugt zwar Feedback, aber nicht zwingend verwertbare Erkenntnisse.

Der Begriff Validierung sollte sorgfältig verwendet werden. Ein Prototyp ist nicht selbst die Validierung. Er kann Validierung ermöglichen oder unterstützen, wenn eine Annahme methodisch angemessen

geprüft wird, etwa mit realen Nutzerinnen und Nutzern, Domänenexpertinnen und Domänenexperten, Usability-Tests oder realen Nutzungsdaten.

Ein Prototyp ist auch nicht automatisch eine vollständige Spezifikation. Er kann Details auslassen, vereinfachen oder nur bestimmte Pfade darstellen. Wenn Prototypen als Grundlage für Umsetzung dienen, müssen fehlende Systemzustände, Regeln, Texte, Interaktionen, Accessibility-Kriterien und fachliche Anforderungen ergänzt werden.

6.15 AI-gestütztes Design, Prototyping und menschliche Prüfung

AI kann in Design und Prototyping besonders gut zur schnellen Variantenbildung beitragen.

AI unterstützt gut bei	Aus Anforderungen, User Stories oder Akzeptanzkriterien erste UI-Entwürfe erzeugen; bekannte UI-Patterns vorschlagen; Informationsstrukturen entwerfen; Interface-Texte variieren; Prototypen vorbereiten; Prüffragen für Design Critiques und Teamdiskussionen erzeugen.
Grenzen und typische Risiken	Garantiert nicht, dass ein Entwurf zum tatsächlichen Nutzungskontext passt; reale Nutzer, Fachlogik, Risiken und implizite Anforderungen kennt AI nur, wenn sie korrekt eingebracht und anschließend geprüft werden; Vollständigkeit von Systemzuständen, Edge Cases und Accessibility ist nicht gesichert; typische Risiken: Entwürfe wirken fertiger, als sie sind, visuelle Attraktivität wird mit UX-Qualität verwechselt, nur der Happy Path wird berücksichtigt, generische UI-Patterns treffen auf unpassende Fachkontexte, Prototypen werden mit Spezifikationen verwechselt, frontend-nahe Ergebnisse wandern zu schnell in Richtung Umsetzung.
Menschlich zu prüfen	Welches Nutzerziel der Entwurf unterstützt; ob der Nutzungskontext ausreichend berücksichtigt ist; welche Annahmen im Entwurf enthalten sind; ob visuelle Hierarchie und Informationsstruktur sinnvoll sind; ob Begriffe verständlich und fachlich korrekt sind; ob relevante Systemzustände und Fehlerfälle berücksichtigt sind; ob Accessibility-Basics erfüllt sind; ob der Prototyp für die beabsichtigte Fragestellung geeignet ist.

AI-Ergebnisse bleiben Entwurfsvarianten und Hypothesen; sie werden durch Menschen fachlich, methodisch und aus Nutzungsperspektive geprüft.

7 Code-Erstellung, Komponenten, Accessibility und Handoff

Unterrichtszeit: ca. 135 Minuten

7.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt die Umsetzung von Usability-Anforderungen, UI-Entwürfen und Prototypen in Code. Im Mittelpunkt steht die Frage, wie Entwürfe in implementierte, konsistente, zugängliche und wartbare UI-Komponenten überführt werden.

Ein UI-Entwurf oder Prototyp macht eine Lösung sichtbar und diskutierbar. Die UI-Implementierung übersetzt diese Lösung in konkrete Komponenten, Code, Zustände, Interaktionslogik, semantische Struktur, Validierung, Fehlermeldungen, Accessibility-Eigenschaften und Tests. UX-Qualität entsteht daher nicht nur im Entwurf, sondern auch in vielen Umsetzungsentscheidungen.

AI-gestützte Coding-Werkzeuge können UI-Komponenten erzeugen, bestehende Codebasen analysieren, Tests vorschlagen, Validierungslogik vorbereiten, Dokumentation schreiben, Stories für einen Component Workshop erstellen oder Hinweise zu Accessibility geben. Dadurch wird der Weg von Usability-Anforderung zu implementierter UI kürzer.

AI-generierter Code ist jedoch nicht automatisch produktionsreif, sicher, wartbar, zugänglich oder fachlich korrekt. Entscheidend ist, dass implementierte UI-Lösungen mit Usability-Requirements, Akzeptanzkriterien, Systemzuständen, Komponentenregeln, Designsystem-Vorgaben, Accessibility-Basics und Tests übereinstimmen. Die Verantwortung für diese Prüfung bleibt beim Team.

7.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 7.1 bis BO 7.4. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

7.3 Learning Objectives

LO	Learning Objective	K
LO 7.1	Typische AI-gestützte Coding-Werkzeuge und Einsatzbereiche für UI-nahe Entwicklung benennen können.	K1
LO 7.2	Erklären können, wie AI-gestützte Code-Erstellung die Schnittstelle zwischen Usability-Anforderungen, UI-Design und Implementierung verändert.	K2
LO 7.3	Grundlegende Usability-Qualitätsaspekte in der Implementierung beschreiben können, insbesondere Systemzustände, Komponentenlogik, Fehlermeldungen, Labels, Fokusführung, Tastaturbedienbarkeit und erwartbares Verhalten.	K2
LO 7.4	Die Bedeutung konsistenter, wiederverwendbarer Komponenten für Usability, Wartbarkeit und Accessibility erklären können.	K2
LO 7.5	Die Funktion von Designsystemen und Design Tokens als Verbindung zwischen Design, UI-Implementierung und Code beschreiben können.	K2
LO 7.6	AI-generierte UI-Implementierungen mit Usability-Requirements, Akzeptanzkriterien, Systemzuständen, Komponentenregeln und Designsystem-Vorgaben abgleichen können.	K3
LO 7.7	AI nutzen können, um Testfälle, Dokumentation, Stories für einen Component Workshop, Prüffragen oder Accessibility-Hinweise vorzubereiten.	K3
LO 7.8	Grundlegende Accessibility-Basics in der Umsetzung benennen können, insbesondere semantisches HTML, Kontrast, Labels, Fokusführung, Tastaturbedienbarkeit und ARIA nur wo nötig.	K1
LO 7.9	Risiken AI-generierter Implementierung erklären können, insbesondere Sicherheitslücken, Accessibility-Probleme, Inkonsistenzen, Wartbarkeitsprobleme, Performance-Probleme und fachlich falsche Logik.	K2
LO 7.10	Geeignete Prüfmaßnahmen für AI-generierten Code ableiten können, insbesondere Code Review, funktionale Tests, UI-Tests, Security Review, Accessibility-Prüfung und fachliche Abnahme.	K3
LO 7.11	Erklären können, wie AI genutzt werden kann, um aus einer bestehenden Anwendung wiederkehrende UI-Muster, Inkonsistenzen und erste Designsystem-Bausteine abzuleiten.	K2

7.4 Wichtige Begriffe

AI-gestützte Code-Erstellung, Coding-Agent, UI-Implementierung, Frontend, UI-Komponente, Komponentenlogik, Komponentenbibliothek, Designsystem, Design Token, Minimal Viable Design

System, retroaktives Designsystem, UI-Inventar, Komponenten-Inventar, Inkonsistenzanalyse, Handoff, Traceability, Component Workshop, Story, semantisches HTML, Accessibility, Accessibility-Basics, Tastaturbedienbarkeit, Fokusführung, Label, ARIA, Validierung, Eingabevalidierung, Fehlermeldung, Systemzustand, Loading State, Empty State, Error State, Success State, Disabled State, erwartbares Verhalten, Code Review, Security Review, Accessibility-Prüfung, fachliche Abnahme, Testfall, Wartbarkeit, Performance, Produktionsreife.

7.5 Von Usability-Anforderungen zur implementierten UI

Usability entsteht in Entwicklungsprozessen nicht nur durch Anforderungen, Entwürfe und Prototypen, sondern auch durch deren konkrete Umsetzung. Eine gut formulierte User Story, ein sauberer Prototyp oder ein überzeugender UI-Entwurf erzeugen noch keine gut nutzbare Software, wenn die Implementierung zentrale Details auslässt.

In der Umsetzung werden viele Usability-relevante Entscheidungen konkret: Welche Komponente wird verwendet? Wie verhält sich ein Button im Disabled State? Wie wird Eingabevalidierung angezeigt? Bleiben Eingaben nach Fehlern erhalten? Ist der Fokus sichtbar? Kann der Ablauf ohne Maus bedient werden? Wird ein Ladezustand angezeigt? Sind Labels korrekt zugeordnet? Sind Fehlermeldungen verständlich und handlungsleitend?

AI-gestützte Coding-Werkzeuge können diese Arbeit beschleunigen. Sie können jedoch auch fehlerhafte oder unvollständige Umsetzungen erzeugen. Deshalb ist Implementierung als Usability-relevante Qualitätsarbeit zu verstehen. Die zentrale Frage lautet nicht nur, ob Code funktioniert, sondern ob die implementierte UI das Nutzerziel im Nutzungskontext verständlich, zugänglich, konsistent und erwartbar unterstützt.

Handoff ist in diesem Zusammenhang nicht nur eine Übergabe von Design an Entwicklung. Es ist ein laufender Abgleich zwischen Requirements, Entwurf, Komponentenregeln, Systemzuständen, Interaktionslogik, Accessibility-Anforderungen, Tests und Code. Besonders bei AI-generierten Entwürfen oder Implementierungen muss klar bleiben, welche Teile fachlich bestätigt sind und welche AI-Ergänzungen nur Annahmen darstellen.

Was in diesem Kapitel umgesetzt wird, wird in späteren Reviews, Tests und Evaluationen geprüft. Daher müssen Testfälle, Prüffragen, Accessibility-Prüfungen und fachliche Abnahmen bereits in der Umsetzung mitgedacht werden. Je stärker AI produktive Implementierung unterstützt, desto wichtiger werden nachvollziehbare Prüfmaßnahmen wie Code Review, funktionale Tests, UI-Tests, Accessibility-Prüfung, Security Review und fachliche Abnahme.

7.6 AI-gestützte Code-Erstellung für Usability und Frontend

AI-gestützte Coding-Werkzeuge unterstützen Softwareentwicklung direkt im Code. Dazu gehören IDE-Assistenzsysteme, Coding-Agenten, Code-Completion-Werkzeuge, Pull-Request-Assistenten, Prompt-to-Code-Werkzeuge und Systeme, die größere Änderungen in Codebasen vorschlagen oder ausführen können.

Typische Einsatzbereiche im Usability-nahen Frontend sind UI-Komponenten, Formulare, Eingabevalidierung, Systemzustände, Fehlermeldungen, Tests, Accessibility-Hinweise, Dokumentation, Stories für einen Component Workshop und Refactorings. Diese Werkzeuge können Entwicklung beschleunigen und helfen, Usability-relevante Details früher aufzudecken. Ihr Output bleibt jedoch ein Vorschlag, der anhand von Requirements, Codequalität, Accessibility, Security, Wartbarkeit und fachlicher Korrektheit geprüft werden muss.

AI-gestützte Code-Erstellung verändert die Schnittstelle zwischen UX-Entwurf, UI-Design und Implementierung. Aus Beschreibungen, Screens, Komponentenbibliotheken oder bestehenden Codebasen können direkt erste Implementierungen entstehen. Dadurch wird der Weg von der Idee zur implementierten UI kürzer. Gleichzeitig steigt das Risiko, dass unklare Anforderungen direkt in Code übersetzt werden.

Wenn Nutzerziel, Nutzungskontext, Systemzustände, Fehlerfälle, Berechtigungen oder Accessibility nicht beschrieben sind, ergänzt AI diese Lücken möglicherweise mit generischen Annahmen. Deshalb sollten Usability-Requirements, Akzeptanzkriterien, Komponentenregeln, Designsystem-Vorgaben und relevante Systemzustände vor der Code-Erstellung möglichst klar sein.

AI kann bessere Implementierungsvorschläge erzeugen, wenn der Kontext präzise bereitgestellt wird:

- verwendetes Framework und bestehende Komponentenbibliothek,
- Designsystem-Regeln und Design Tokens,
- Akzeptanzkriterien, Systemzustände und Accessibility-Anforderungen,
- Testanforderungen und fachliche Einschränkungen,
- Codekonventionen und Security-Anforderungen.

Je weniger Kontext bereitgestellt wird, desto stärker orientiert sich AI an generischen Mustern.

AI-generierter Code sollte als Vorschlag behandelt werden, nicht als fertige Lösung. Ein AI-generiertes Formular kann technisch Eingaben verarbeiten, aber unklare Labels, schlechte Fokusführung, fehlende Fehlermeldungen oder unzureichende Tastaturbedienbarkeit enthalten. AI-Code muss daher technisch, fachlich, aus Usability-Sicht, hinsichtlich Accessibility, Security und Wartbarkeit geprüft werden.

7.7 Usability in der Implementierung

Usability entsteht in der Implementierung durch konkrete Umsetzungsdetails: konsistente Komponenten, vollständige Systemzustände, semantisches HTML, Tastaturbedienbarkeit, sinnvolle Fokusführung, verständliche Labels, hilfreiche Fehlermeldungen, erwartbares Verhalten und grundlegende Accessibility.

Systemzustände sind ein zentraler Teil der Implementierung. Gute UX erfordert, dass Nutzerinnen und Nutzer verstehen, was gerade passiert und wie sie weiter handeln können. Dazu gehören Loading State, Empty State, Error State, Success State, Disabled State, fehlende Berechtigung und fehlgeschlagene Eingabevalidierung. Diese Zustände müssen nicht nur visuell vorhanden sein, sondern korrekt ausgelöst und verständlich kommuniziert werden.

Fehlermeldungen sind Usability-kritisch. Sie sollten verständlich, konkret und handlungsorientiert sein. Eine gute Fehlermeldung beschreibt, welches Problem aufgetreten ist, wo es aufgetreten ist und wie Nutzer es beheben können. In der Implementierung ist außerdem wichtig, dass Fehlermeldungen dem richtigen Feld zugeordnet sind, sichtbar bleiben, von assistiven Technologien erfasst werden können und Eingaben nicht unnötig verloren gehen.

Labels geben Eingabefeldern Bedeutung. Platzhaltertexte ersetzen keine Labels, weil sie beim Tippen verschwinden und oft nicht ausreichend zugänglich sind. Hilfetexte können zusätzliche Orientierung geben, sollten aber gezielt eingesetzt werden. AI kann Labels und Hilfetexte vorschlagen. Das Team muss prüfen, ob die Begriffe zur Fachsprache der Nutzer, zum mentalen Modell, zur Aufgabe und zum Nutzungskontext passen.

Fokusführung ist wichtig für Tastaturnutzung, Screenreader-Nutzung und effiziente Bedienung. Nutzer müssen erkennen können, welches Element aktiv ist. Die Tab-Reihenfolge sollte logisch sein. Dialoge, Menüs und modale Fenster müssen Fokus korrekt setzen und nach dem Schließen sinnvoll zurückgeben.

Tastaturbedienbarkeit bedeutet, dass interaktive Elemente ohne Maus erreichbar und bedienbar sind. Gerade bei eigenen Komponenten, Modals, Dropdowns, Tabs oder komplexen Formularen ist menschliche Prüfung notwendig, weil AI-generierte Komponenten Fokus-Management und Tastaturbedienbarkeit nicht immer korrekt berücksichtigen.

Erwartbares Verhalten verbindet UI-Design, Interaction Design und Implementierung. Ein Button soll eine Aktion auslösen, ein Link navigieren, ein Toggle einen Zustand ändern, ein Tab zwischen zusammengehörigen Inhalten wechseln, eine Suche nachvollziehbare Ergebnisse liefern und ein Filter sichtbar aktivierbar und zurücksetzbar sein. Wenn Control, Verhalten und Systemfeedback nicht zusammenpassen, entstehen Irritationen, Fehler oder zusätzlicher Erklärungsbedarf.

Performance und wahrgenommene Reaktionsfähigkeit beeinflussen ebenfalls UX. Lange Ladezeiten, fehlende Rückmeldung oder blockierte Interaktionen können Nutzer irritieren. In der Umsetzung bedeutet dies, Ladezustände anzuzeigen, blockierte Aktionen zu erklären, doppelte Ausführung zu verhindern und Fortschritt anzuzeigen, wenn Vorgänge länger dauern.

7.8 Komponenten, Designsysteme und Design Tokens

Komponenten, Designsysteme und Design Tokens sind nicht nur Mittel zur visuellen Konsistenz. Sie unterstützen Usability, Wartbarkeit und Accessibility, weil sie wiederkehrende Interaktionen, Zustände, Benennungen und Verhaltensweisen systematisch vereinheitlichen.

Wiederverwendbare Komponenten helfen, Konsistenz, Wartbarkeit und Accessibility sicherzustellen. Wenn Buttons, Formulare, Fehlermeldungen, Dialoge oder Tabellen immer wieder neu umgesetzt werden, entstehen Inkonsistenzen. Diese erhöhen kognitive Belastung und erschweren die Nutzung.

Eine Komponente besteht nicht nur aus visueller Darstellung. Sie enthält Verhalten, Zustände, Varianten, Accessibility-Eigenschaften und technische Schnittstellen. Eine Formularfeld-Komponente umfasst etwa Eingabefeld, Label, Hilfetext, Fehlermeldung, Pflichtfeldkennzeichnung, Validierungszustand, Fokusverhalten und programmatische Zuordnung für assistive Technologien.

Für AI-gestützte Entwicklung sind Komponenten besonders wichtig. Wenn AI bestehende Komponenten nutzt, bleibt die Anwendung konsistenter. Wenn AI dagegen für jeden Screen neue Einzellösungen erzeugt, entstehen schnell Varianten, die visuell ähnlich wirken, aber unterschiedlich funktionieren.

Designsysteme beschreiben wiederverwendbare UI-Komponenten, Gestaltungsregeln, Interaktionsmuster, Texte, Systemzustände und Qualitätskriterien. Sie verbinden Design, Code, Accessibility, Dokumentation und Governance. Gerade bei AI-gestützter Umsetzung helfen sie, AI-Ausgaben an bestehenden Regeln auszurichten.

Nicht jedes Team benötigt sofort ein umfangreiches Designsystem. Ein Minimal Viable Design System kann mit wenigen zentralen Elementen beginnen:

- Farben, Typografie und Spacing,
- Buttons und Formularfelder,
- Fehlermeldungen, Dialoge und Tabellen,
- Systemzustände und grundlegende Accessibility-Regeln.

Entscheidend ist nicht Vollständigkeit, sondern Verbindlichkeit für häufige und Usability-kritische Elemente.

Design Tokens sind benannte Werte für Gestaltungseigenschaften wie Farben, Abstände, Schriftgrößen, Border-Radius oder Schatten. Sie verbinden Design und Code. Tokens helfen, Konsistenz und Wartbarkeit zu verbessern, weil Änderungen zentral gesteuert werden können. Für AI-gestützte Umsetzung ist wichtig, dass AI bestehende Tokens und Designsystem-Regeln kennt. Sonst erzeugt sie möglicherweise abweichende Werte oder neue Varianten.

AI kann Designsystemarbeit unterstützen, etwa durch Dokumentation von Komponenten, Erkennen von Inkonsistenzen, Vorschläge für Tokens, Usage Guidelines, Stories für einen Component Workshop, Vergleich von Implementierung mit Designsystem-Regeln und Unterstützung beim Refactoring. Sie kann jedoch nicht allein entscheiden, welche Designsystem-Regeln gelten sollen. Governance, Freigabe und Pflege bleiben menschliche Aufgaben.

7.9 AI zur Ableitung von UI-Mustern und Designsystem-Bausteinen

Viele Teams beginnen nicht mit einem fertigen Designsystem, sondern mit einer gewachsenen Anwendung. Unterschiedliche Buttons, Formularfelder, Abstände, Farben, Fehlermeldungen, Tabellenvarianten oder Dialoge sind über mehrere Sprints oder Releases hinweg entstanden. Dadurch entstehen Inkonsistenzen, die für Nutzer verwirrend und für Teams schwer wartbar werden können.

AI kann helfen, aus einer bestehenden Anwendung erste Designsystem-Bausteine abzuleiten. Dazu kann AI Screenshots, Codeausschnitte, Komponenten, CSS-Regeln oder UI-Beschreibungen analysieren und wiederkehrende Muster auflisten: Welche Button-Varianten gibt es? Welche Formularfelder werden verwendet? Welche Farben und Abstände kommen vor? Welche Systemzustände sind implementiert? Wo gibt es Inkonsistenzen? Welche Komponenten könnten vereinheitlicht werden?

Der Nutzen liegt darin, dass ein Team nicht mit einem idealen Designsystem beginnen muss. Es kann aus dem vorhandenen Produkt heraus ein Minimal Viable Design System aufbauen: zunächst zentrale Farben, Typografie, Spacing, Buttons, Formularfelder, Fehlermeldungen, Tabellen, Dialoge und Systemzustände. Daraus entstehen schrittweise Komponentenregeln, Design Tokens, Dokumentation und Prüfkriterien.

AI kann diesen Prozess beschleunigen, aber nicht automatisch entscheiden, welche Variante künftig als Standard gelten soll. Diese Entscheidung muss anhand von Usability, Accessibility, technischer Wartbarkeit, Markenanforderungen, bestehender Nutzung und Teamkonventionen getroffen werden. Eine häufig vorkommende Variante ist nicht automatisch die beste Variante. Häufigkeit darf daher nicht mit Qualität verwechselt werden.

7.10 Handoff zwischen Design, AI und Entwicklung

Ein guter Handoff beschreibt nicht nur, wie etwas aussieht. Er beschreibt, wie eine UI funktioniert und welche Anforderungen an Usability, Accessibility, fachliche Richtigkeit und Tests erfüllt werden müssen. Dazu gehören:

- Nutzerziel und Nutzungskontext,
- Akzeptanzkriterien,
- Komponenten, Varianten und Systemzustände,
- Fehlerfälle, Texte und Eingabevalidierung,
- Accessibility-Anforderungen und Datenabhängigkeiten,
- Testhinweise und offene Annahmen.

Bei AI-generierten Entwürfen ist besonders zu prüfen, welche Annahmen enthalten sind, welche Systemzustände fehlen, welche Komponenten verwendet werden sollen, welche Texte Platzhalter sind, welche Interaktionen nicht spezifiziert sind, welche Eingabevalidierung vorgesehen ist und welche Accessibility-Kriterien gelten. Ein AI-generierter Entwurf sollte nicht ungeprüft in Code überführt werden.

AI kann helfen, aus einem Entwurf eine Handoff-Beschreibung zu erstellen. Sie kann Komponentenlisten, Systemzustände, offene Fragen, Testhinweise, Accessibility-Hinweise oder Akzeptanzkriterien vorschlagen. Solche Unterlagen müssen geprüft werden, weil AI Details ergänzen kann, die nicht im Entwurf enthalten waren. Diese Ergänzungen sind als Annahmen zu kennzeichnen, wenn sie nicht fachlich bestätigt sind.

Traceability ist bei AI-gestützter Umsetzung besonders wichtig. Es sollte erkennbar sein, welches Usability-Requirement durch welche Komponente, welchen Code, welchen Systemzustand oder welchen Test abgedeckt wird. Wenn nachvollziehbar ist, welche Anforderung durch welche Umsetzung adressiert wird, lassen sich Lücken, Missverständnisse und unbeabsichtigte Abweichungen leichter erkennen.

In agilen Teams ist Handoff kein einmaliger Übergabepunkt. Anforderungen, Entwürfe, Implementierung und Tests entwickeln sich iterativ weiter. Handoff sollte daher als laufender Abgleich verstanden werden: Was wurde entschieden? Was ist umgesetzt? Was ist offen? Was muss geprüft werden?

7.11 Accessibility-Basics in der Umsetzung

Accessibility-Basics sind ein Bestandteil von Usability und professioneller UI-Implementierung. Sie stellen sicher, dass zentrale Aufgaben auch mit unterschiedlichen Fähigkeiten, Hilfsmitteln, Geräten und Nutzungssituationen ausgeführt werden können.

Accessibility ist kein nachgelagertes Zusatzthema und kein rein formales Compliance-Thema. Sie betrifft die grundlegende Nutzbarkeit interaktiver Systeme für Menschen mit unterschiedlichen Fähigkeiten, Einschränkungen und Nutzungssituationen. Dazu gehören dauerhafte Einschränkungen ebenso wie situative Bedingungen, etwa mobile Nutzung, schlechte Lichtverhältnisse, Zeitdruck, Stress oder Bedienung ohne Maus.

Standards und Richtlinien wie WCAG, EN 301 549 und ISO/IEC 40500 bilden wichtige fachliche Bezugspunkte für digitale Barrierefreiheit. Im Foundation-Level-Kontext meint Accessibility-Basics eine gezielte Auswahl grundlegender Anforderungen, die agile Softwareentwicklungsteams in der Umsetzung kennen und berücksichtigen müssen. Dazu gehören insbesondere semantisches HTML, ausreichender Kontrast, verständliche Labels, Tastaturbedienbarkeit, sichtbare und sinnvolle Fokusführung, zugängliche Fehlermeldungen und der vorsichtige Einsatz von ARIA.

Diese Auswahl ersetzt kein eigenständiges Accessibility-Training und keine fachkundige Accessibility-Prüfung. Digitale Barrierefreiheit ist ein eigenes Fachgebiet mit eigenen Standards, Methoden, Prüfverfahren und Spezialisierungen. Dieser Syllabus behandelt Accessibility insoweit, wie sie für AI-gestützte Usability-Arbeit und UI-nahe Umsetzung auf Foundation Level unmittelbar relevant ist.

Semantisches HTML bedeutet, passende HTML-Elemente entsprechend ihrer Bedeutung und Funktion zu verwenden. Ein Button sollte eine Aktion auslösen, ein Link zu einem anderen Ziel navigieren, Überschriften sollten als Überschriften ausgezeichnet und Formularfelder mit Labels verbunden sein. AI-generierter Code verwendet nicht immer die semantisch passende Struktur, etwa wenn ein klickbares div statt eines echten Buttons verwendet wird.

Tastaturbedienbarkeit bedeutet, dass alle interaktiven Elemente mit der Tastatur erreichbar, bedienbar und in sinnvoller Reihenfolge fokussierbar sind. Besonders wichtig sind Formulare, Menüs, Dialoge, Dropdowns, Tabs und komplexe Komponenten. Custom Components, die optisch wie Standard-Controls aussehen, verhalten sich nicht automatisch wie native Bedienelemente.

Fokusführung zeigt, wo sich die aktuelle Interaktion befindet. Ein sichtbarer Fokus ist notwendig, damit Tastaturnutzer wissen, welches Element aktiv ist. Bei dynamischen Bereichen, Dialogen, Menüs oder modalen Fenstern muss der Fokus sinnvoll gesetzt und zurückgeführt werden. Fehler in der Fokusführung können eine Oberfläche für bestimmte Nutzergruppen schwer oder gar nicht bedienbar machen.

Labels, Hilfetexte und Fehlermeldungen müssen verständlich, fachlich korrekt und zugänglich sein. Platzhaltertexte ersetzen keine Labels. Fehlermeldungen sollten klar dem betroffenen Feld oder Bereich zugeordnet sein und für assistive Technologien programmatisch erkennbar sein. Gute Fehlermeldungen erklären, was passiert ist, wie Nutzer das Problem beheben können, und erhalten möglichst den bisherigen Arbeitsstand.

Ausreichender Kontrast ist eine Voraussetzung dafür, dass Texte, Icons, Controls und Zustände wahrgenommen werden können. Farbe darf nicht das einzige Mittel sein, um Bedeutung zu vermitteln. Ein Fehlerzustand sollte daher nicht nur rot markiert sein, sondern auch durch Text, Icon oder strukturelle Zuordnung verständlich werden.

ARIA kann Accessibility verbessern, wenn native HTML-Elemente nicht ausreichen. Falsch eingesetztes ARIA kann Accessibility jedoch verschlechtern. Grundsätzlich sollten native HTML-Elemente bevorzugt werden, wenn sie die benötigte Semantik und Funktion bereits bieten. ARIA sollte nicht verwendet werden, um falsches HTML zu reparieren, wenn ein passendes natives Element verfügbar ist.

AI und automatisierte Tools können Accessibility-Probleme aufdecken, finden aber nur einen Teil der Probleme. Accessibility-Prüfung sollte daher automatisierte Checks, manuelle Prüfung, Tastaturtest, gegebenenfalls Screenreader-Test und bei relevanten Risiken spezialisierte Accessibility-Expertise umfassen. AI kann diesen Prozess unterstützen, aber nicht ersetzen.

7.12 AI für Dokumentation, Tests und Reviews

AI kann aus Akzeptanzkriterien, Code, Komponentenregeln oder Handoff-Unterlagen Testfälle vorschlagen. Besonders nützlich ist dies bei Systemzuständen, Fehlerfällen, Eingabevalidierung und UI-Verhalten. Solche Tests sollten nicht nur den Happy Path, sondern auch ungültige Eingaben, erhaltene Eingabedaten nach Fehlern, Loading States, Empty States, Disabled States, Tastaturbedienbarkeit, Fokusverhalten und Dialogverhalten berücksichtigen.

Ein Component Workshop ist eine Entwicklungsumgebung, in der einzelne Komponenten losgelöst vom Rest der Anwendung in unterschiedlichen Varianten und Systemzuständen dargestellt und geprüft werden. Jede dargestellte Ausprägung einer Komponente wird dort als Story bezeichnet. AI kann Stories für Komponenten erzeugen, etwa Button Primary, Button Disabled, Button Loading, Form Field Error oder Modal Open. Wichtig ist, dass nicht nur der Standardfall abgebildet wird, sondern auch Fehlerzustände, leere Zustände, lange Texte, deaktivierte Varianten, Tastaturverhalten und Accessibility-relevante Eigenschaften.

AI kann Prüffragen und Checklisten für UI-Code erzeugen. Diese können Komponentenverwendung, semantisches HTML, Tastaturbedienbarkeit, Fokusführung, Systemzustände, Fehlermeldungen, Konsistenz, Tests, Security-Risiken und fachliche Korrektheit betreffen. Checklisten sollten kontextbezogen bleiben. Zu allgemeine Listen werden schnell unpraktisch.

AI kann auch Komponentendokumentation, Usage Guidelines, Code-Kommentare oder Handoff-Texte formulieren. Diese Dokumentation ist hilfreich, wenn sie korrekt, knapp und nachvollziehbar ist. Besonders wichtig ist, dass AI keine Begründung erfindet, die im Team nie entschieden wurde. Dokumentation muss

tatsächliche Entscheidungen, geprüfte Anforderungen, bekannte Einschränkungen und offene Annahmen wiedergeben.

AI kann außerdem helfen, Code, Akzeptanzkriterien und Tests miteinander abzugleichen. Beispielsweise kann sie prüfen, ob für relevante Systemzustände Tests vorhanden sind oder ob eine Komponente die beschriebenen Varianten abdeckt. Dieser Abgleich unterstützt Traceability, ersetzt aber keine fachliche Entscheidung. Wenn AI keine Lücke findet, bedeutet das nicht, dass die Implementierung vollständig oder korrekt ist.

7.13 Risiken AI-generierter Implementierung

Risiken AI-generierter Implementierung betreffen nicht nur technische Korrektheit. Sie können auch Usability, Accessibility, Sicherheit, Wartbarkeit, Performance, fachliche Logik und Vertrauen in das System beeinträchtigen.

AI-generierter Code kann Sicherheitslücken enthalten. Dies betrifft unsichere Eingabeverarbeitung, fehlende oder falsche Eingabevalidierung, unsichere API-Nutzung, unzureichende Authentifizierung, fehlerhafte Berechtigungsprüfungen oder unsicheren Umgang mit Daten. Auch Frontend-Code kann sicherheitsrelevant sein, etwa wenn sensible Informationen angezeigt, Berechtigungen falsch angenommen oder Eingaben unzureichend behandelt werden.

AI kann visuell passende Komponenten erzeugen, die accessibility-technisch problematisch sind. Häufige Probleme sind fehlende Labels, falsche Semantik, unzureichende Fokusführung, nicht bedienbare Custom Components, zu geringer Kontrast, nur farbliche Bedeutungsmarkierung oder falscher ARIA-Einsatz. Diese Probleme sind nicht immer durch visuelle Prüfung erkennbar.

AI kann neue Varianten erzeugen, statt bestehende Komponenten zu nutzen. Dadurch entstehen Inkonsistenzen in Verhalten, Styling, Texten, Systemzuständen oder Interaktionsmustern. Inkonsistenzen sind nicht nur ästhetische Probleme; sie können dazu führen, dass Nutzer gleiche Funktionen unterschiedlich interpretieren oder gleiche Zustände unterschiedlich verstehen.

AI-generierter Code kann schwer wartbar sein, wenn er zu komplex, redundant oder nicht konform mit Projektstandards ist. Wartbarkeit ist auch Usability-relevant, weil schwer wartbare UI-Strukturen spätere Verbesserungen, Accessibility-Korrekturen oder Konsistenzarbeit erschweren.

AI kann fachliche Regeln falsch interpretieren. Dies ist besonders kritisch bei Eingabevalidierung, Berechtigungen, Workflow-Schritten, Statuslogik oder regulatorischen Anforderungen. Eine Oberfläche kann effizient wirken, aber zentrale Prozessregeln verletzen.

AI kann Dokumentation erzeugen, die überzeugend klingt, aber nicht den tatsächlichen Entscheidungen entspricht. Dadurch entsteht ein Schein von Nachvollziehbarkeit. Teams sollten daher prüfen, ob Dokumentation tatsächliche Entscheidungen, geprüfte Anforderungen und bekannte Einschränkungen wiedergibt.

Beim retroaktiven Aufbau eines Designsystems besteht zusätzlich das Risiko, dass AI vorhandene Muster nur beschreibt, aber nicht bewertet. Eine häufig vorkommende Button-Variante ist nicht automatisch die beste Variante. Eine bestehende Inkonsistenz wird nicht dadurch zum Standard, dass sie in der Anwendung mehrfach vorkommt.

7.14 Prüfmaßnahmen für AI-generierten Code

Prüfmaßnahmen für AI-generierten Code müssen technische Qualität und Nutzungsqualität gemeinsam betrachten. Dazu gehören Code Review, funktionale Tests, UI-Tests, Accessibility-Prüfung, Security Review, fachliche Abnahme und Prüfung der Usability im konkreten Nutzungskontext.

AI-generierter Code sollte daher wie anderer Code reviewed werden, häufig sogar sorgfältiger. Code Review sollte nicht nur Stil und Funktion prüfen, sondern auch Usability-relevante Aspekte: Systemzustände, Fehlermeldungen, Fokusführung, Komponentenverwendung, Accessibility, Konsistenz und erwartbares Verhalten.

Funktionale Tests und UI-Tests sollten nicht nur den erfolgreichen Ablauf prüfen. Sie sollten auch Fehlerfälle, alternative Pfade und Systemzustände berücksichtigen. Wichtige Testbereiche sind:

- Eingabevalidierung und Fehlerzustände,
- Ladezustände und leere Datenlagen,
- Berechtigungen,
- Tastaturbedienung und Fokusverhalten,
- Komponentenvarianten,
- Abbruch und Wiederaufnahme von Abläufen.

Accessibility sollte mit automatisierten Tools, manueller Prüfung und bei Bedarf spezialisierter Expertise geprüft werden. Manuell zu prüfen sind insbesondere Tastaturfluss, sichtbarer Fokus, Fokuslogik bei Dialogen und dynamischen Inhalten, Verständlichkeit von Fehlermeldungen, Zuordnung von Labels und Fehlermeldungen, Screenreader-Erfahrung, semantische Struktur und Aufgabenangemessenheit.

Security Review ist erforderlich, wenn AI Code erzeugt oder verändert. Dies betrifft Datenverarbeitung, Authentifizierung, Autorisierung, externe Bibliotheken, API-Zugriffe und Eingabevalidierung. Security ist auch aus Usability-Sicht relevant, weil Login, Session-Timeouts, Berechtigungen, Fehlermeldungen und

Bestätigungen sicher und zugleich verständlich umgesetzt werden müssen.

Fachliche Abnahme prüft, ob die Implementierung zur Domäne, zum Prozess und zum Nutzerziel passt. Ein Feature kann technisch funktionieren und dennoch fachlich falsch sein. Die fachliche Abnahme stellt sicher, dass Rollen, Berechtigungen, Fachbegriffe, rechtliche oder organisatorische Rahmenbedingungen und Prozesslogik korrekt berücksichtigt wurden.

Produktionsreife bedeutet, dass eine Implementierung nicht nur lokal funktioniert oder visuell passend aussieht, sondern für den produktiven Einsatz geeignet ist. Dazu gehören technische Stabilität, Security, Accessibility, Performance, Wartbarkeit, fachliche Korrektheit, Tests, Monitoring und nachvollziehbare Freigabe. AI kann dafür Hinweise, Testvorschläge und Dokumentation liefern, aber keine Freigabeverantwortung übernehmen.

7.15 AI-gestützte Umsetzung und menschliche Prüfung

AI kann in der Umsetzung besonders gut bei wiederkehrenden UI- und Entwicklungsaufgaben unterstützen; je klarer Usability-Requirements, Akzeptanzkriterien, Komponentenregeln und Designsystem-Vorgaben vorliegen, desto besser die Ergebnisse.

AI unterstützt gut bei	Komponenten generieren; Code erklären und refaktorieren; Systemzustände ergänzen; Formulare und Eingabevalidierung vorbereiten; Fehlermeldungen vorschlagen; Testfälle erzeugen; Stories für einen Component Workshop und Komponentendokumentation erstellen; Accessibility-Hinweise liefern; Inkonsistenzen in Code oder Komponenten aufdecken.
Grenzen und typische Risiken	Garantiert keine Produktionsreife: Sicherheit, Wartbarkeit, Zugänglichkeit, Performance und fachliche Korrektheit sind nicht gesichert; Projektstandards, Architekturentscheidungen, fachliche Regeln und regulatorische Anforderungen kennt AI nur, wenn sie ausreichend eingebracht und anschließend geprüft werden.
Menschlich zu prüfen	Welche Usability-Anforderung die Implementierung erfüllen soll; ob der Code die Akzeptanzkriterien erfüllt; ob relevante Systemzustände umgesetzt sind; ob Fehlermeldungen verständlich und korrekt zugeordnet sind; ob bestehende Komponenten, Design Tokens und Designsystem-Regeln verwendet werden; ob Tastaturbedienung und Fokusführung funktionieren; ob semantisches HTML korrekt und ARIA notwendig und korrekt eingesetzt ist; ob grundlegende WCAG-relevante Anforderungen

	berücksichtigt sind; ob Security-Risiken bestehen; ob die Lösung wartbar und performant ist; ob die fachliche Logik stimmt. Die Prüfung wird nachvollziehbar dokumentiert, insbesondere wenn AI wesentliche Teile von Code, Komponenten, Dokumentation oder Tests erzeugt hat.

Entscheidend ist nicht nur, dass etwas funktioniert, sondern dass es im Nutzungskontext richtig, verständlich, zugänglich, sicher und wartbar umgesetzt ist.

8 Evaluation, Testing und Validierung

Unterrichtszeit: ca. 150 Minuten

8.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt die strukturierte Überprüfung von Usability im Softwareentwicklungsprozess. Im Mittelpunkt steht die Frage, wie Teams erkennen, ob eine Lösung tatsächlich verständlich, nutzbar, erwartungskonform und wirksam ist.

Dafür müssen mehrere Begriffe unterschieden werden. Evaluation ist der Oberbegriff für die systematische Bewertung eines Entwurfs, Prototyps, Produkts oder Nutzungspfads mit Blick auf Usability, Nutzerziel und Nutzungskontext. Evaluation kann mit realen Nutzerinnen und Nutzern erfolgen, etwa durch Usability-Tests, oder expertenbasiert, etwa durch heuristische Evaluation oder Cognitive Walkthrough.

Review-Formate, etwa ein Sprint Review, sind Teamformate, in denen Arbeitsergebnisse gezeigt und besprochen werden. Sie können Fragen zu Usability sichtbar machen, sind aber nicht automatisch Evaluationsmethoden im engeren Sinn und ersetzen keine Validierung mit realen Nutzerinnen und Nutzern.

Testing kann im Softwarekontext Unterschiedliches bedeuten. Funktionale Tests prüfen, ob spezifizierte Funktionen korrekt arbeiten. Usability-Tests prüfen Usability nicht nur technisch, sondern durch Beobachtung realer oder repräsentativer Nutzerinnen und Nutzer bei konkreten Aufgaben. Dadurch werden Verständnisprobleme, Fehler, Suchbewegungen, Unsicherheiten und Hindernisse sichtbar.

Validierung bedeutet, Annahmen über Nutzer, Aufgaben, Verständlichkeit, Verhalten oder Nutzungskontext mit geeigneten Quellen zu überprüfen. Dazu können reale Nutzerbeobachtung, Usability-Tests, Domänenwissen, Nutzungsdaten oder andere belastbare Evidenz beitragen.

GenAI kann Evaluation, Testing und Validierung vorbereiten und strukturieren. Sie kann Prüffragen formulieren, heuristische Evaluationen vorbereiten, Cognitive-Walkthrough-Schritte strukturieren, Testaufgaben entwerfen, Beobachtungsnotizen ordnen, Findings clustern und Berichtsentwürfe erzeugen. AI ersetzt jedoch keine realen Nutzerinnen und Nutzer. AI-gestützte Analysen sind als Unterstützung, Hypothesenquelle oder Strukturierungshilfe zu verstehen, nicht als Ersatz für Evaluation oder Validierung.

Ziel dieses Kapitels ist es, Softwareentwicklungsteams zu befähigen, Usability mit einfachen Methoden strukturiert zu prüfen, AI sinnvoll zur Vorbereitung und Auswertung einzusetzen und die Grenzen

synthetischer oder heuristischer AI-Analysen klar zu erkennen.

8.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 8.1 bis BO 8.5. Die vollständige und verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

8.3 Learning Objectives

LO	Learning Objective	K
LO 8.1	Grundlegende Evaluationsmethoden für Usability sowie expertenbasierte Verfahren benennen können, insbesondere heuristische Evaluation, Cognitive Walkthrough und Usability-Test.	K1
LO 8.2	Den Zweck von Evaluation und Validierung in Softwareentwicklungsprozessen erklären können.	K2
LO 8.3	Erklären können, warum in Review-Formaten, etwa im Sprint Review, nicht nur Funktionalität, sondern auch Nutzerziel, Verständlichkeit, Systemzustände und Fehlertoleranz betrachtet werden sollten.	K2
LO 8.4	Eine heuristische Evaluation mit AI-Unterstützung vorbereiten und deren Ergebnisse kritisch einordnen können.	K3
LO 8.5	Den Grundgedanken des Cognitive Walkthrough als expertenbasiertes Verfahren erklären können.	K2
LO 8.6	Die zentralen Bestandteile eines einfachen Usability-Tests beschreiben können, insbesondere Untersuchungsfrage, Testpersonen, Testgegenstand, Testaufgaben, Moderation, Beobachtung und Auswertung.	K2
LO 8.7	Testaufgaben zielorientiert formulieren können, ohne den Lösungsweg vorzugeben.	K3
LO 8.8	Den Zweck und die Grenzen von Think-Aloud und strukturierter Beobachtung erklären können.	K2
LO 8.9	GenAI zur Strukturierung von Beobachtungsnotizen, Findings und Berichtsentwürfen einsetzen können.	K3
LO 8.10	Probleme der Usability nach Schweregrad, Häufigkeit, Nutzerwirkung, Risiko, Umsetzungsaufwand und Business-Relevanz priorisieren können.	K3

LO	Learning Objective	K
LO 8.11	Aus AI-generierten Vorschlägen zu Usability testbare Annahmen und Untersuchungsfragen ableiten können.	K3
LO 8.12	Erklären können, warum AI-Simulationen, AI-gestützte heuristische Vorprüfungen und synthetische Nutzer keine Validierung mit realen Nutzerinnen und Nutzern ersetzen.	K2

8.4 Wichtige Begriffe

Review-Format, Sprint Review, Evaluation, formative Evaluation, summative Evaluation, Validierung, Verifikation, Expert Review, expertenbasiertes Verfahren, heuristische Evaluation, Usability-Heuristik, Cognitive Walkthrough, Usability-Test, moderierter Usability-Test, unmoderierter Usability-Test, Remote Usability Test, formativer Usability-Test, summativer Usability-Test, Untersuchungsfrage, Testfrage, Hypothese, Testgegenstand, Testaufgabe, Testperson, Moderation, Beobachtung, Think-Aloud, Lautes Denken, Finding, Schweregrad, Severity, Häufigkeit, Nutzerwirkung, Risiko, Business-Relevanz, Umsetzungsaufwand, Berichtsentwurf, A/B-Test, AI-gestützte Auswertung, AI-Simulation, synthetischer Nutzer, menschliche Prüfung.

8.5 Evaluation, Testing und Validierung im Softwareentwicklungsprozess

In Softwareentwicklungsprozessen sind Review-Formate, etwa das Sprint Review, zentrale Momente, um Arbeitsergebnisse zu zeigen. Häufig wird dabei vor allem gezeigt, dass eine Funktion implementiert wurde. Für UX reicht das nicht aus. Eine Funktion kann technisch umgesetzt sein und dennoch aus Nutzersicht schwer verständlich, fehleranfällig, unvollständig oder nicht erwartungskonform sein.

Review-Formate können genutzt werden, um Usability-relevante Fragen sichtbar zu machen: Wird das Nutzerziel unterstützt? Ist der Ablauf verständlich? Sind relevante Systemzustände umgesetzt? Sind Fehlermeldungen, Systemfeedback und Interaktionen erwartbar? Sind Accessibility-Basics berücksichtigt? Gibt es offene Annahmen, die noch geprüft werden müssen?

Ein Review-Format ist jedoch nicht automatisch eine Usability-Evaluationsmethode im engeren Sinn. Es ist zunächst eine strukturierte Besprechung oder Prüfung durch Team, Stakeholder oder Fachpersonen. Es kann Hinweise auf mögliche Usability-Probleme liefern, ersetzt aber keine methodisch geeignete Evaluation und keine Validierung mit realen Nutzerinnen und Nutzern.

Evaluation bezeichnet die systematische Bewertung eines Entwurfs, Prototyps, Produkts oder

Nutzungspfads anhand von Kriterien, Methoden oder Beobachtungen. Evaluation kann expertenbasiert erfolgen oder reale Nutzerinnen und Nutzer einbeziehen. Beispiele sind heuristische Evaluation, Cognitive Walkthrough und Usability-Test.

Testing kann im Softwareentwicklungsprozess funktionale Tests, UI-Tests, Accessibility-Tests, technische Prüfungen oder Usability-Tests meinen. Im Usability-Kontext ist deshalb jeweils klar zu benennen, welche Art von Test gemeint ist.

Validierung bedeutet, zentrale Annahmen über Nutzer, Aufgaben, Verständlichkeit, Verhalten oder Nutzungskontext mit geeigneter Evidenz zu überprüfen. Eine AI-Analyse kann etwa die Hypothese erzeugen: „Nutzer könnten die Filterfunktion übersehen.“ Validierung bedeutet, diese Annahme durch geeignete Quellen zu prüfen, etwa durch Usability-Test, Beobachtung, Domänenexpertise, Nutzungsdaten oder andere belastbare Evidenz.

Ergebnisse aus Evaluation, Testing und Validierung müssen in Entscheidungen überführt werden. Sie können zu neuen Backlog Items, ergänzten Akzeptanzkriterien, Designanpassungen, technischen Aufgaben, weiteren Tests oder dokumentiertem UX-Debt führen. Dabei ist zwischen beobachteten Problemen, fachlichen Einschätzungen, AI-generierten Hinweisen und offenen Hypothesen zu unterscheiden.

8.6 Usability in Review-Formaten sichtbar machen

In Review-Formaten sollte nicht nur Funktionalität betrachtet werden. Funktionalität beschreibt, ob ein System eine bestimmte Leistung erbringt. Nutzbarkeit beschreibt, ob bestimmte Nutzerinnen und Nutzer diese Leistung in einem bestimmten Nutzungskontext effektiv, effizient und zufriedenstellend verwenden können.

Eine Exportfunktion ist technisch umgesetzt, wenn sie eine Datei erzeugt. Aus Usability-Sicht ist zusätzlich relevant, ob Nutzerinnen und Nutzer die Funktion finden, verstehen, welches Format erzeugt wird, erkennen, wann der Export abgeschlossen ist, und bei Fehlern eine verständliche Rückmeldung erhalten.

Ein Review-Format kann anhand einfacher Usability-Prüffragen strukturiert werden: Welches Nutzerziel unterstützt diese Lösung? Ist der nächste Schritt erkennbar? Sind wichtige Systemzustände umgesetzt? Sind Fehlermeldungen verständlich und handlungsleitend? Ist Systemfeedback sichtbar und erwartungskonform? Sind relevante Edge Cases und Accessibility-Basics berücksichtigt? Gibt es offene Annahmen über Nutzerverhalten?

GenAI kann Review-Formate vorbereiten, indem sie Prüffragen aus User Stories, Akzeptanzkriterien, Prototypen oder implementierten Funktionen ableitet. Diese Fragen müssen an Nutzungskontext,

Zielgruppe und fachliche Anforderungen angepasst werden.

Ergebnisse aus Review-Formaten sollten dokumentiert werden. Dabei ist zu unterscheiden zwischen bestätigtem Problem, offener Frage, Verbesserungsidee, AI-generiertem Hinweis, Hypothese für weitere Evaluation und daraus abgeleitetem Backlog Item. Ein Finding sollte auf Beobachtung, Prüfung oder fachlicher Bewertung beruhen. Eine AI-Vermutung bleibt eine Hypothese, bis sie geprüft wurde.

8.7 Heuristische Evaluation mit AI-Unterstützung

Eine heuristische Evaluation ist ein expertenbasiertes Verfahren. Dabei wird eine Benutzeroberfläche systematisch anhand definierter Usability-Heuristiken, Interaktionsprinzipien oder Gestaltungskriterien geprüft, um mögliche Probleme der Usability aufzudecken. Ohne solchen Bezugsrahmen handelt es sich nicht um eine heuristische Evaluation, sondern um eine allgemeine Einschätzung.

Mögliche Bezugspunkte sind Sichtbarkeit des Systemstatus, Übereinstimmung mit Nutzererwartungen, Konsistenz, Fehlervermeidung, Wiedererkennen statt Erinnern, Unterstützung bei Fehlerbehebung, Steuerbarkeit, Aufgabenangemessenheit oder verständliche Systemrückmeldungen.

Für agile Teams ist heuristische Evaluation nützlich, weil sie relativ schnell durchgeführt werden kann und keine aufwendige Testinfrastruktur erfordert. Sie kann grobe Probleme früh aufdecken und eignet sich als Vorprüfung von Entwürfen, Prototypen oder implementierten Screens. Sie ersetzt jedoch keine Usability-Tests. Eine heuristische Evaluation zeigt mögliche Probleme aus Expertensicht, aber nicht verlässlich, wie reale Nutzerinnen und Nutzer tatsächlich handeln.

GenAI kann eine heuristische Evaluation vorbereiten oder unterstützen, wenn klare Heuristiken oder Kriterien vorgegeben werden. AI kann eine Oberfläche, einen Screenshot, eine Beschreibung oder einen Prototyp anhand dieser Kriterien analysieren, mögliche Probleme benennen, Verbesserungsvorschläge formulieren und Prüffragen erzeugen.

Die Ergebnisse AI-gestützter heuristischer Bewertungen müssen kritisch eingeordnet werden. AI kann Probleme erkennen, die im Kontext nicht relevant sind, oder Probleme übersehen, die ohne Domänenwissen nicht sichtbar werden. Ergebnisse können daher als offensichtlich zutreffend, plausibel aber prüfbedürftig, unklar oder kontextabhängig, nicht relevant oder als Frage für weitere Evaluation eingeordnet werden.

Wichtig ist die Trennung zwischen heuristischem Hinweis und evidenzbasiertem Finding. Ein heuristischer Hinweis kann auf ein mögliches Problem hinweisen. Ein Finding sollte dagegen auf konkreter Prüfung, Beobachtung oder fachlich nachvollziehbarer Bewertung beruhen.

8.8 Cognitive Walkthrough

Ein Cognitive Walkthrough ist ein expertenbasiertes Verfahren. Dabei wird ein Nutzungspfad Schritt für Schritt aus Sicht einer Nutzerin oder eines Nutzers durchgegangen. Im Mittelpunkt steht die Frage, ob Nutzerinnen und Nutzer die notwendigen Schritte erkennen, verstehen und ausführen können, um ein bestimmtes Ziel zu erreichen.

Die Methode ist besonders nützlich für neue Nutzerinnen und Nutzer, selten genutzte Funktionen oder Abläufe, bei denen Orientierung, Erlernbarkeit und erste Verständlichkeit wichtig sind. Sie eignet sich auch, um früh zu prüfen, ob ein Entwurf oder Prototyp zu den angenommenen mentalen Modellen der Nutzer passt.

Das Verfahren arbeitet mit vier Leitfragen: Wird der Nutzer erkennen, was der nächste sinnvolle Schritt ist? Wird der Nutzer das richtige Element finden? Wird der Nutzer verstehen, dass dieses Element zur Aufgabe passt? Wird der Nutzer nach der Aktion erkennen, ob er erfolgreich war? In der Praxis werden häufig zwei weitere Fragen ergänzt: Wird der Nutzer die richtige Aktion ausführen können? Wird der Nutzer wissen, was bei einem Fehler zu tun ist? Diese beiden Fragen gehören nicht zum ursprünglichen Verfahren, sind in der Anwendung aber verbreitet. Daneben existiert eine gestraffte Fassung, der Streamlined Cognitive Walkthrough, der mit zwei Leitfragen arbeitet.

Diese Leitfragen sollten immer auf einen konkreten Nutzungspfad bezogen werden. Ein Cognitive Walkthrough bewertet nicht allgemein „die ganze Oberfläche“, sondern prüft schrittweise, ob eine bestimmte Aufgabe mit einem bestimmten Interface voraussichtlich bewältigt werden kann.

GenAI kann helfen, einen Ablauf in Schritte zu zerlegen, mögliche Verständlichkeitsprobleme je Schritt zu formulieren und aus einer User Story, einem Prototyp oder einer Prozessbeschreibung einen Walkthrough-Plan zu erstellen. Die eigentliche Prüfung erfordert jedoch Wissen über Nutzer, Aufgabe, Domäne und Interface.

Ein Cognitive Walkthrough bleibt eine Experten- oder Teamprüfung. Er zeigt mögliche Probleme, aber nicht, ob reale Nutzerinnen und Nutzer tatsächlich scheitern. Wenn zentrale Annahmen unsicher sind oder hohe Risiken bestehen, sollten reale Nutzerinnen und Nutzer einbezogen werden.

8.9 Usability-Tests planen

Usability-Tests sind eine zentrale Methode, um Usability mit realen oder repräsentativen Nutzerinnen und Nutzern zu überprüfen. Sie zeigen, ob Nutzer konkrete Aufgaben verstehen, passende Funktionen finden, Fehlersituationen bewältigen und ihre Ziele im Nutzungskontext erreichen können.

Usability-Tests sind in diesem Syllabus keine bloßen Testfälle mit bestanden/nicht bestanden. Sie sind eine nutzungsbezogene Evaluationsmethode, mit der Softwareentwicklungsteams prüfen, ob reale oder repräsentative Nutzerinnen und Nutzer konkrete Aufgaben verstehen, ausführen und abschließen können.

Testpersonen bearbeiten konkrete Aufgaben mit einem Produkt, Prototyp, Entwurf oder einer Anwendung. Beobachtet wird, wie sie vorgehen, wo sie suchen, zögern, Fehler machen, etwas missverstehen oder eine Aufgabe abbrechen.

Ziel eines Usability-Tests ist nicht, allgemeine Meinungen über ein Design zu sammeln. Im Mittelpunkt steht beobachtbares Verhalten im Zusammenhang mit konkreten Aufgaben. Dadurch werden Verständnisprobleme, Hindernisse, Fehlbedienungen und unerwartete Nutzungsmuster sichtbar.

Usability-Tests können moderiert oder unmoderiert, vor Ort oder remote, formativ oder summativ durchgeführt werden. Moderierte Tests eignen sich besonders, wenn das Team verstehen möchte, warum Nutzerinnen und Nutzer zögern, scheitern oder andere Wege wählen als erwartet. Unmoderierte Tests können schneller und mit mehr Personen durchgeführt werden, liefern aber weniger Kontext. Formative Tests dienen der Verbesserung während der Entwicklung, summative Tests eher der Bewertung einer weiter fortgeschrittenen oder fertigen Lösung.

Vor einem Usability-Test muss die Untersuchungsfrage geklärt werden. Eine Frage wie „Ist der neue Screen gut?“ ist zu allgemein. Besser ist etwa: „Erkennen neue Nutzerinnen und Nutzer, wie sie nach dem ersten Login ein Projekt anlegen können?“ Die Untersuchungsfrage bestimmt Testpersonen, Testgegenstand, Testaufgaben und relevante Beobachtungen.

Testpersonen sollten zur relevanten Nutzergruppe passen. Für erste pragmatische Tests müssen es nicht immer große Stichproben sein. Schon wenige passende Personen können wichtige Probleme aufdecken. Entscheidend ist, dass die Personen eine sinnvolle Nähe zur tatsächlichen Nutzung haben.

Der Testgegenstand muss zur Untersuchungsfrage passen. Ein Test kann mit Wireframe, klickbarem Prototyp, funktionsnahem Prototyp, Staging-Version, bestehender Anwendung oder einzelnem Ablauf durchgeführt werden. Ebenso muss der Systemzustand definiert sein, etwa leere Anwendung, vorhandene Daten, bestimmte Berechtigungen, ausgelöster Fehlerzustand oder teilweise abgeschlossener Prozess.

Ein einfacher Usability-Test benötigt Ziel und Untersuchungsfrage, passende Testpersonen, geeigneten Testgegenstand und definierten Systemzustand, konkrete Testaufgaben, kurze Einführung, Beobachtungsleitfaden, Regeln für Moderation, Dokumentation der Beobachtungen und Auswertung der Findings. AI kann bei Testskripten, Aufgabenentwürfen, Beobachtungsbögen und Auswertungsrastern unterstützen.

8.10 Testaufgaben formulieren

Gute Testaufgaben übersetzen Usability-Fragen in beobachtbare Aufgaben. Sie beschreiben ein Ziel aus Nutzersicht, ohne den Lösungsweg oder das relevante UI-Element vorzugeben.

Testaufgaben sollten aus der Untersuchungsfrage abgeleitet werden. Die Untersuchungsfrage beschreibt, was das Team herausfinden möchte. Die Testaufgabe schafft eine Situation, in der beobachtet werden kann, ob Nutzerinnen und Nutzer die relevante Aufgabe tatsächlich bewältigen können.

Eine gute Testaufgabe beschreibt ein Ziel aus Nutzersicht, ohne den Lösungsweg vorzugeben. Wenn die Aufgabe schon sagt, welchen Button Nutzerinnen und Nutzer klicken sollen, wird nicht mehr getestet, ob sie die Funktion selbst finden und verstehen.

Schwach wäre: „Klicken Sie auf den Button ‚Exportieren'." Besser ist: „Sie möchten die Daten für die Monatsabrechnung weitergeben. Finden Sie heraus, wie Sie diese Daten aus dem System erhalten können." Die zweite Aufgabe prüft, ob Nutzerinnen und Nutzer die Exportfunktion finden, verstehen und verwenden können.

Testaufgaben sollten realistische Situationen beschreiben. Eine Aufgabe wie „Verwenden Sie den Filter" ist weniger hilfreich als: „Sie möchten nur offene Anträge anzeigen, die seit mehr als sieben Tagen unbearbeitet sind. Finden Sie heraus, wie Sie diese Liste erstellen können."

Testaufgaben sollten keine Begriffe verwenden, die direkt auf UI-Elemente hinweisen, wenn genau deren Auffindbarkeit geprüft werden soll. Wenn getestet werden soll, ob Nutzerinnen und Nutzer den Tab „Archiv" finden, sollte die Aufgabe nicht lauten: „Öffnen Sie den Tab Archiv." Besser ist: „Sie möchten einen abgeschlossenen Antrag aus dem letzten Monat erneut ansehen."

Eine Testaufgabe sollte so formuliert sein, dass beobachtbar ist, ob sie erfolgreich bearbeitet wurde. Dafür muss klar sein, welches Ergebnis erwartet wird. Wenn kein beobachtbares Erfolgskriterium existiert, wird die Auswertung unscharf.

AI kann aus User Stories, Akzeptanzkriterien, Hypothesen oder Untersuchungsfragen Testaufgaben formulieren. Diese Aufgaben müssen geprüft werden: Leiten sie sich aus der Untersuchungsfrage ab? Beschreiben sie ein realistisches Nutzerziel? Verraten sie die Lösung? Passen sie zur Zielgruppe? Ist der Erfolg beobachtbar? Passen sie zum Testgegenstand und Systemzustand?

8.11 Moderation, Think-Aloud und strukturierte Beobachtung

Moderation in einem Usability-Test bedeutet, den Ablauf zu erklären, die Aufgabe zu geben, Beobachtung

zu ermöglichen und bei Bedarf neutral nachzufragen. Moderation bedeutet nicht, zu helfen, zu erklären oder Nutzerinnen und Nutzer zum richtigen Verhalten zu führen.

Wenn eine Testperson zögert, sollte nicht sofort geholfen werden. Gerade das Zögern ist eine wichtige Beobachtung. Eine gute Moderation schafft einen sicheren Rahmen. Testpersonen sollten verstehen, dass nicht sie getestet werden, sondern das System.

Neutrale Nachfragen helfen, Gedanken zu verbalisieren, ohne zu lenken. Beispiele sind: „Was erwarten Sie, was hier passiert?“, „Was suchen Sie gerade?“, „Was macht Sie unsicher?“ oder „Was würden Sie als Nächstes tun?“ Nicht neutral wären Hinweise wie: „Sehen Sie den Button oben rechts?“ oder „Würden Sie jetzt auf Export klicken?“

Think-Aloud bedeutet, dass Testpersonen während der Aufgabe laut aussprechen, was sie denken, erwarten, suchen oder verstehen. Dadurch können mentale Modelle, Unsicherheiten, Erwartungen und Missverständnisse sichtbar werden. Think-Aloud ersetzt jedoch nicht die Beobachtung tatsächlichen Verhaltens. Was Nutzer sagen und was sie tun, kann auseinanderfallen.

Think-Aloud hat Grenzen. Lautes Denken kann die Bearbeitung beeinflussen, die kognitive Belastung erhöhen, das Verhalten verlangsamen oder zu Rationalisierungen führen. Es ist daher als unterstützende Methode zu verstehen, nicht als vollständige Erklärung des Nutzerverhaltens.

Strukturierte Beobachtung sollte möglichst beobachtungsnah dokumentieren, wo Testpersonen zögern, was übersehen wird, welche Begriffe missverstanden werden, welche Fehlermeldungen helfen oder nicht helfen, welche falschen Schritte gewählt werden, ob Hilfe benötigt wird oder ob eine Aufgabe abgebrochen wird.

Für Softwareentwicklungsteams ist dabei wichtig, Beobachtungen nicht vorschnell als individuelles Nutzerproblem zu interpretieren. Zögern, Umwege, Nachfragen oder Fehler können viel häufiger Hinweise auf Probleme der Usability, Verständlichkeit oder Systemrückmeldung sein.

AI kann Moderationsleitfäden, Beobachtungsbögen und Protokollstrukturen vorbereiten. Nach einem Test kann AI Notizen ordnen und erste Muster herausarbeiten. Bei der Auswertung muss geprüft werden, ob AI tatsächliche Beobachtungen korrekt wiedergibt oder bereits interpretiert, zusammenfasst oder gewichtet.

8.12 Beobachtungen, Findings und Berichtsentwürfe strukturieren

Ein häufiges Problem in der Auswertung ist die Vermischung von Beobachtung und Interpretation. Eine Beobachtung beschreibt, was tatsächlich gesehen, gehört oder protokolliert wurde. Eine Interpretation

erklärt, was dies bedeuten könnte.

Beobachtung: „Drei von fünf Testpersonen suchten die Exportfunktion im Menü ‚Berichte' und nicht in der Tabellenansicht." Interpretation: „Die Exportfunktion ist wahrscheinlich nicht dort platziert, wo Nutzerinnen und Nutzer sie erwarten." Beide Aussagen sind nützlich, sollten aber getrennt bleiben.

Ein gutes Finding beschreibt, was beobachtet wurde, bei wem oder in welcher Aufgabe es auftrat, welche Auswirkung es hatte, warum es relevant ist und ob es Hinweise auf Ursache oder Lösung gibt. Ein Finding sollte konkret, nachvollziehbar und evidenzbasiert sein. Es sollte nicht nur eine Meinung oder eine AI-Vermutung wiedergeben.

Findings können aus unterschiedlichen Quellen entstehen: Usability-Tests, heuristische Evaluation, Cognitive Walkthrough, fachliche Prüfung, Supportdaten, Nutzungsdaten oder AI-gestützte Hinweise. Diese Quellen sollten klar gekennzeichnet werden. Ein beobachtetes Problem aus einem Usability-Test hat eine andere Aussagekraft als eine AI-Vermutung.

AI kann Beobachtungsnotizen clustern, ähnliche Probleme zusammenführen und erste Findings formulieren. Dies ist hilfreich, wenn viele Notizen oder mehrere Tests ausgewertet werden. Die AI-Auswertung muss jedoch geprüft werden: Wurden Beobachtungen korrekt wiedergegeben? Wurden mehrere Probleme unzulässig zusammengefasst? Wurden Minderheitenprobleme übersehen? Wurden Interpretationen als Fakten formuliert? Ist die Quelle des Findings klar erkennbar?

AI kann auch Berichtsentwürfe erzeugen. Solche Entwürfe sparen Zeit, dürfen aber nicht ungeprüft übernommen werden. Besonders wichtig ist, dass AI keine Evidenz ergänzt, die nicht beobachtet wurde. Wenn nur zwei Testpersonen beteiligt waren, darf daraus keine allgemeine Aussage über „die meisten Nutzer" werden.

8.13 Probleme der Usability priorisieren

Nicht jedes Problem der Usability hat dieselbe Bedeutung. Einige Probleme verhindern die Aufgabenerfüllung vollständig. Andere erzeugen kleine Irritationen. Einige treten häufig auf, andere selten, sind aber in kritischen Situationen schwerwiegend.

Priorisierung hilft, Findings in umsetzbare Entscheidungen zu überführen. Ohne Priorisierung entsteht häufig eine lange Liste von Problemen, aber kein klarer nächster Schritt. Priorisierung ist keine rein mechanische Bewertung. Sie verbindet Evidenz, Nutzerwirkung, Risiko, Häufigkeit, Business-Relevanz, Aufwand und Produktstrategie.

Usability-Probleme können nach Schweregrad, Häufigkeit, Nutzerwirkung, Risiko, Business-Relevanz,

Umsetzungsaufwand, regulatorischer Bedeutung und strategischer Bedeutung bewertet werden. Ein Problem ist nicht automatisch hoch priorisiert, nur weil es häufig auftritt. Ein seltenes Problem kann kritisch sein, wenn es zu Fehlentscheidungen, Datenverlust, Sicherheitsrisiken, rechtlichen Problemen oder erheblichem Vertrauensverlust führt.

AI kann Severity- oder Priorisierungsvorschläge machen. Diese können hilfreich sein, sind aber ohne Kontext unsicher. AI kennt nicht automatisch Business-Ziele, rechtliche Risiken, technische Abhängigkeiten, Nutzerzahlen, Kosten oder strategische Prioritäten.

Das Team muss daher prüfen, welche Evidenz vorliegt, aus welcher Quelle ein Finding stammt, wie schwer die Auswirkung für Nutzerinnen und Nutzer ist, wie häufig das Problem auftritt, welche Risiken entstehen, welche Business-Relevanz besteht, wie aufwendig die Behebung ist und was passiert, wenn das Problem nicht behoben wird.

Priorisierung ist erst dann wirksam, wenn daraus konkrete Maßnahmen entstehen. Findings können zu sofortiger Korrektur, neuem Backlog Item, Ergänzung von Akzeptanzkriterien, Anpassung eines Prototyps, technischer Aufgabe, Accessibility-Prüfung, weiterer Evaluation oder Dokumentation als UX-Debt führen.

8.14 Aus AI-Vorschlägen testbare Annahmen ableiten

AI kann aus einem Screen, Prototyp, Backlog Item oder einer implementierten Funktion mögliche Probleme der Usability benennen. Sie können auf blinde Flecken hinweisen, sind aber keine bestätigten Findings und keine validierten Erkenntnisse.

Ein AI-Vorschlag wie „Die Filterfunktion könnte schwer auffindbar sein“ kann in eine testbare Untersuchungsfrage überführt werden: „Finden Nutzerinnen und Nutzer die Filterfunktion, wenn sie eine Ergebnisliste auf offene Anträge einschränken möchten?“ Die Stärke von AI liegt hier nicht darin, das Problem zu beweisen, sondern darin, mögliche Annahmen offenzulegen.

Eine testbare Untersuchungsfrage beschreibt, was überprüft werden soll und unter welchen Bedingungen eine Beobachtung aussagekräftig wäre. Relevante Fragen sind: Welche Nutzergruppe ist betroffen? Welche Aufgabe soll bearbeitet werden? Welcher Screen, Prototyp oder Ablauf ist betroffen? Welches Verhalten soll beobachtet werden? Woran erkennt man Erfolg, Unsicherheit oder ein Problem? Welche Entscheidung soll durch die Prüfung unterstützt werden?

Nicht jede Frage benötigt einen Usability-Test. Manche Fragen können durch heuristische Evaluation, Cognitive Walkthrough, Expert Review, Tree Testing, Analytics, Supportdaten oder Domänenprüfung bearbeitet werden. Andere Fragen benötigen reale Nutzerinnen und Nutzer. Das Team entscheidet,

welche Methode zur Untersuchungsfrage, zum Risiko, zum Entwicklungsstand und zur verfügbaren Evidenz passt.

Offene Annahmen sollten dokumentiert werden. Wenn eine AI-Analyse ein Problem vermutet, aber keine Prüfung erfolgt, muss dies dokumentiert bleiben. Eine geeignete Dokumentation kann festhalten: AI-generierter Hinweis, daraus abgeleitete Hypothese, mögliche Untersuchungsfrage, vorgeschlagene Methode, aktueller Evidenzstatus, Entscheidung des Teams und nächster Schritt.

Nicht jede Hypothese führt sofort zu einem Backlog Item. Manche Hypothesen müssen zuerst geprüft werden. Andere können als offene Frage, Verbesserungsidee oder UX-Debt dokumentiert werden. AI kann diese Kategorien strukturieren und erste Formulierungen vorschlagen. Die Entscheidung bleibt beim Team.

8.15 AI-Simulationen und synthetische Nutzer

AI-Simulationen können für die Vorbereitung von Usability-Tests hilfreich sein, etwa um mögliche Fragestellungen, Risiken oder Testaufgaben zu sammeln. Sie sind jedoch alleine keine Evidenz für Usability und keine Validierung realer Nutzung.

AI kann simulieren, wie ein bestimmter Nutzertyp möglicherweise auf eine Oberfläche reagiert. Sie kann typische Fragen, Missverständnisse, Erwartungen oder mögliche Barrieren formulieren. Dies kann hilfreich sein, um Hypothesen zu erzeugen, blinde Flecken zu erkennen oder eine Evaluation vorzubereiten.

AI-Simulationen beruhen jedoch auf Mustern, nicht auf realem Verhalten konkreter Personen. Sie zeigen nicht, was reale Nutzerinnen und Nutzer tatsächlich tun. Sie erfassen keine echten Suchbewegungen, kein tatsächliches Zögern, keine nonverbalen Signale, keine reale Frustration, keine situativen Bedingungen und keine tatsächliche Fehlbedienung.

Synthetische Nutzer sind durch AI erzeugte Nutzerprofile, Rollen oder simulierte Reaktionen. Sie können helfen, Annahmen explizit zu machen oder unterschiedliche Perspektiven gedanklich durchzuspielen. Sie dürfen aber nicht mit echten Nutzerinnen und Nutzern verwechselt werden.

Ein synthetischer Nutzer basiert nicht auf eigener Erfahrung, realem Kontext oder tatsächlichem Verhalten. Er ist eine generierte Darstellung, die plausibel klingen kann, aber keine Evidenz für reale Nutzung liefert. Synthetische Nutzer können daher höchstens als Hilfsmittel zur Hypothesengenerierung dienen.

Ein besonderes Risiko besteht darin, dass AI-Simulationen Sicherheit erzeugen, obwohl keine reale

Prüfung stattgefunden hat. Wenn AI schreibt, dass „Nutzer den Ablauf wahrscheinlich verstehen“, kann das beruhigend wirken. Ohne echte Beobachtung bleibt es eine Vermutung.

Echte Tests sind besonders wichtig, wenn zentrale Nutzerannahmen unsicher sind, eine Funktion geschäftskritisch ist, Fehler hohe Folgen haben, Zielgruppen wenig bekannt sind, Fachsprache oder Domänenlogik komplex ist, mehrere plausible Lösungen existieren, Accessibility oder besondere Nutzungssituationen relevant sind oder ein Ergebnis Grundlage wichtiger Produktentscheidungen werden soll.

8.16 AI-gestützte Auswertung und menschliche Prüfung

AI kann in Evaluation, Testing und Validierung besonders gut bei Vorbereitung, Strukturierung und Dokumentation unterstützen.

AI unterstützt gut bei	Prüffragen aus Backlog Items ableiten; heuristische Kriterien strukturieren; Cognitive-Walkthrough-Schritte vorbereiten; Testaufgaben formulieren; Moderationsleitfäden und Beobachtungsbögen entwerfen; Beobachtungsnotizen clustern; Findings vorformulieren; Berichtsentwürfe und Priorisierungsvorschläge erstellen; AI-Vorschläge in testbare Hypothesen übersetzen.
Grenzen und typische Risiken	Validiert keine reale Nutzung: Verhalten, Zögern, Scheitern, nonverbale Signale, tatsächliche Frustration und reale Arbeitsumgebungen kann AI nicht beobachten; typische Risiken: AI-Analysen werden als Nutzertest missverstanden, synthetische Nutzer mit realen verwechselt, heuristische Hinweise als validierte Findings dokumentiert, Beobachtung und Interpretation vermischt, Minderheitenprobleme übersehen, Testaufgaben verraten die Lösung, Berichtsentwürfe suggerieren mehr Evidenz, als vorhanden ist.
Menschlich zu prüfen	Welche Untersuchungsfrage geprüft wird und welche Methode dazu passt; ob es sich um Review, Evaluation, Test oder Validierung handelt; auf welcher Quelle ein Finding beruht; ob Testaufgaben zielorientiert und ohne Lösungshinweis formuliert sind; ob Testpersonen zur Nutzergruppe passen; ob Beobachtung und Interpretation getrennt wurden; ob Findings konkret und evidenzbasiert sind; ob die Priorisierung nachvollziehbar ist; welche AI-Aussagen übernommen, angepasst oder verworfen wurden; was Evidenz und was Hypothese ist; was sofort behoben, was als UX-

	Debt dokumentiert und was in Backlog Items, Akzeptanzkriterien oder Tests überführt wird.

AI kann Usability-Evaluation pragmatischer und zugänglicher machen, darf aber nicht mit evidenzbasierter Evaluation oder Validierung verwechselt werden; Bewertung und Verantwortung bleiben beim Team beziehungsweise bei den zuständigen Rollen.

9 Release, Monitoring und Verbesserungen

Unterrichtszeit: ca. 90 Minuten

9.1 Leitgedanke des Kapitels

Dieses Kapitel behandelt Usability nach dem Release. Arbeit an Usability endet nicht mit der Umsetzung eines Inkrements und auch nicht mit einem Review-Format wie dem Sprint Review. Erst in der tatsächlichen Nutzung zeigt sich, ob Annahmen aus Discovery, Backlog Refinement, Design, Implementierung, Evaluation und Testing in der Praxis tragen.

Der Begriff Feedback wird in diesem Kapitel im Sinn von Nutzer-, Kunden-, Support- oder Betriebsfeedback verwendet. Er bezeichnet Rückmeldungen aus der tatsächlichen Nutzung oder aus dem produktiven Umfeld, etwa Hinweise, Beschwerden, Wünsche, Beobachtungen oder Erfahrungsberichte. Davon zu unterscheiden ist Systemfeedback, also die Rückmeldung des Systems an Nutzerinnen und Nutzer während der Interaktion, etwa Fehlermeldungen, Ladezustände, Erfolgsmeldungen oder Bestätigungen.

Nach dem Release entstehen neue Hinweise auf Usability-Probleme. Nutzerinnen und Nutzer geben Feedback, Supporttickets häufen sich zu bestimmten Themen, Nutzungsdaten zeigen Abbrüche oder selten genutzte Funktionen, qualitative Rückmeldungen machen Irritationen sichtbar, und Metriken zeigen, ob bestimmte Ziele erreicht werden.

GenAI kann helfen, solche Rückmeldungen und Daten zu strukturieren. AI kann Feedback clustern, Supporttickets zusammenfassen, Muster in Freitexten erkennen, UX-Debt-Listen vorbereiten, Metriken erklären oder Hypothesen für Verbesserungen der Usability formulieren. Gleichzeitig dürfen Muster in Daten nicht mit Ursachen verwechselt werden. Eine Abbruchrate zeigt, dass etwas passiert. Sie erklärt nicht automatisch, warum es passiert.

Ziel dieses Kapitels ist es, Softwareentwicklungsteams zu befähigen, Usability nach dem Release systematisch zu beobachten, AI-gestützte Auswertungen kritisch einzuordnen und Erkenntnisse in Backlog Items, UX-Debt-Maßnahmen oder neue Discovery-Fragen zu überführen.

9.2 Beitrag zu den Business Outcomes

Dieses Kapitel unterstützt insbesondere die Business Outcomes BO 9.1 bis BO 9.4. Die vollständige und

verbindliche Formulierung der Business Outcomes befindet sich in Abschnitt 0.9.

9.3 Learning Objectives

LO	Learning Objective	K
LO 9.1	Typische Quellen für Erkenntnisse zu Usability nach dem Release benennen können, insbesondere Nutzerfeedback, Kundenfeedback, Supporttickets, Nutzungsdaten, technische Signale und UX-Metriken.	K1
LO 9.2	Erklären können, warum Usability nach dem Release weiter beobachtet und verbessert werden muss.	K2
LO 9.3	Grundlegende Metriken für Usability wie Aufgabenerfolg, Fehlerrate, Abbruchrate, Bearbeitungszeit, Wiederkehrrate, Nutzung zentraler Funktionen und Supportaufkommen beschreiben können.	K2
LO 9.4	Den Grundgedanken des HEART-Frameworks erklären und die Dimensionen Happiness, Engagement, Adoption, Retention und Task Success als Strukturierungshilfe für UX-Ziele und UX-Metriken einordnen können.	K2
LO 9.5	GenAI einsetzen können, um Feedback, Tickets, Metriken und Hinweise auf Usability-Probleme oder UX-Debt zu strukturieren und Muster sichtbar zu machen.	K3
LO 9.6	Den Unterschied zwischen Muster, Korrelation, Ursache, Priorität und belastbarer Produktentscheidung erläutern können.	K2
LO 9.7	UX-Debt anhand typischer Beispiele mit Bezug zu Usability erkennen können, etwa unvollständige Systemzustände, inkonsistente Komponenten, unklare Texte, fehlende Accessibility oder ungelöste Usability-Probleme.	K3
LO 9.8	Erkenntnisse aus Feedback, Metriken, Supportdaten und AI-gestützter Analyse in Backlog Items, UX-Debt-Maßnahmen, Verbesserungen der Usability oder neue Discovery-Fragen überführen können.	K3

9.4 Wichtige Begriffe

Release, Monitoring, Feedback, Nutzerfeedback, Kundenfeedback, Supportfeedback, Betriebsfeedback, Systemfeedback, Supportticket, Supportdaten, Nutzungsdaten, Analytics, technische Signale, UX-Metrik, Aufgabenerfolg, Task Success, Fehlerrate, Abbruchrate, Bearbeitungszeit, Time on Task, Wiederkehrrate, Retention, Adoption, Nutzung zentraler Funktionen, Supportaufkommen, HEART-

Framework, Happiness, Engagement, Adoption, Retention, Task Success, Goals–Signals–Metrics, KPI, Muster, Korrelation, Ursache, Hypothese, Evidenz, Priorität, Produktentscheidung, UX-Debt, Technical Debt, Backlog Item, Verbesserungsmaßnahme, Discovery-Frage, kontinuierliche Verbesserung, AI-gestützte Analyse, AI-gestützte Auswertung, menschliche Prüfung, Datenschutz, Anonymisierung.

9.5 Usability nach dem Release beobachten

Nach dem Release zeigt sich Usability im realen Nutzungskontext. Teams erkennen dann, ob Nutzer Aufgaben tatsächlich abschließen, wo sie abbrechen, welche Funktionen selten genutzt werden, welche Fehlermeldungen zu Supportfällen führen und welche Annahmen aus Discovery, Backlog Refinement, Design, Implementierung oder Testing nicht tragen.

Review-Formate, Evaluation und Testing vor dem Release liefern wichtige Hinweise auf UX-Qualität. Sie zeigen jedoch nicht vollständig, wie eine Lösung in realer Nutzung funktioniert. Manche Usability-Probleme werden erst sichtbar, wenn Nutzerinnen und Nutzer mit echten Daten, echten Aufgaben, echten Geräten, Zeitdruck, Ablenkung, Berechtigungen, organisatorischen Rahmenbedingungen oder Fachkontext arbeiten.

Nach dem Release wird sichtbar, ob die Annahmen aus vorherigen Prozessschritten tragen. Discovery-Hypothesen können bestätigt, präzisiert oder widerlegt werden. Akzeptanzkriterien können sich als zu eng oder unvollständig erweisen. Prototypen können Probleme nicht gezeigt haben, die in produktiver Nutzung auftreten. Tests können Randfälle übersehen haben.

Monitoring umfasst die systematische Beobachtung von Nutzerfeedback, Kundenfeedback, Supportfällen, Nutzungsdaten, technischen Signalen und UX-Metriken. Es ist keine rein technische Aktivität. Auch technische Signale wie Ladezeiten, Fehlerraten, Timeouts oder Abbrüche können Usability-Relevanz haben, wenn sie die Zielerreichung beeinflussen.

Für agile Softwareentwicklungsteams ist Monitoring besonders nützlich, wenn Erkenntnisse wieder in Backlog Refinement, Discovery, Design, Umsetzung oder technische Verbesserung zurückgeführt werden. Andernfalls entstehen Berichte, aber keine Verbesserung.

UX-Debt entsteht, wenn Usability-Probleme bewusst oder unbewusst liegen bleiben. Nach dem Release wird UX-Debt häufig sichtbar: Nutzerinnen und Nutzer melden dieselben Probleme, bestimmte Abläufe erzeugen Abbrüche, Hilfetexte fehlen, Systemzustände sind unvollständig, Komponenten sind inkonsistent oder Accessibility-Probleme bleiben bestehen.

9.6 Quellen für Erkenntnisse zu Usability nach dem Release

Typische Quellen für Usability-Erkenntnisse nach dem Release sind Nutzerfeedback, Kundenfeedback, Supporttickets, Supportdaten, Nutzungsdaten, Analytics, technische Signale und UX-Metriken. Diese Quellen zeigen unterschiedliche Ausschnitte der tatsächlichen Nutzung und sollten nicht isoliert betrachtet werden.

Direktes Nutzerfeedback und Kundenfeedback können aus Feedbackformularen, Interviews, Kommentaren, Bewertungen, Supportgesprächen, internen Rückmeldungen, Kundenkommunikation oder Produktumfragen stammen. Feedback zeigt häufig, was Nutzerinnen und Nutzer als störend, unklar, hilfreich oder fehlend erleben. Es ist wertvoll, aber nicht automatisch repräsentativ.

Supporttickets zeigen, wo Nutzerinnen und Nutzer Hilfe benötigen oder scheitern. Häufen sich Tickets zu einer Funktion, kann das auf Verständlichkeitsprobleme, fehlendes Systemfeedback, technische Fehler, unklare Begriffe, fehlende Berechtigungsinformationen oder Lücken in der Dokumentation hinweisen. Supportdaten zeigen häufig Symptome. Die Ursache muss durch weitere Analyse eingegrenzt werden.

Nutzungsdaten und Analytics zeigen, was Nutzerinnen und Nutzer tatsächlich tun. Sie können Hinweise liefern auf Abbrüche, selten genutzte Funktionen, lange Bearbeitungszeiten, wiederholte Fehler, häufige Rücksprünge, nicht abgeschlossene Transaktionen oder ungewöhnliche Pfade. Analytics beantwortet vor allem die Frage, was passiert. Es beantwortet nicht automatisch, warum es passiert.

Qualitative Rückmeldungen wie App-Store-Reviews, Produktbewertungen, Umfragekommentare oder interne Stakeholder-Rückmeldungen enthalten oft Hinweise auf Frustration, Erwartungen oder wiederkehrende Missverständnisse. AI kann solche Rückmeldungen clustern und Tonalitäten beschreiben. Tonalität ist jedoch nicht dasselbe wie Priorität.

Auch technische Signale können Usability-relevant sein. Dazu gehören Ladezeiten, Fehlerraten, Timeouts, Crash Rates, Validierungsfehler, API-Latenzen, nicht abgeschlossene Transaktionen oder wiederholte Auslöseversuche derselben Aktion. Wenn Nutzerinnen und Nutzer Aktionen mehrfach auslösen, fehlt möglicherweise sichtbares Systemfeedback.

AI kann diese Quellen strukturieren, Muster sichtbar machen und Ursachenhypothesen formulieren. Die fachliche Interpretation bleibt Aufgabe des Teams.

9.7 Warum Usability nach dem Release weiter verbessert werden muss

Usability ist keine einmalig abgeschlossene Eigenschaft. Nach dem Release verändern sich Nutzung,

Erwartungen, technische Rahmenbedingungen, Supportaufkommen und Produktziele. Daher müssen Teams beobachten, welche Annahmen tragen, welche Probleme neu entstehen und welche Verbesserungen priorisiert werden sollten.

Tests, Evaluationen und Review-Formate vor dem Release sind wichtig, bilden reale Nutzung aber nicht vollständig ab. In echter Nutzung arbeiten Menschen mit eigenen Daten, echten Konsequenzen, Zeitdruck, Ablenkung, unterschiedlichen Geräten, individuellen Erwartungen und organisatorischen Rahmenbedingungen.

Manche Probleme entstehen erst durch reale Datenmengen, seltene Rollen, ungewöhnliche Berechtigungen, langsame Verbindungen, parallele Arbeit, mobile Nutzung oder komplexe Kombinationen von Funktionen. Deshalb ist Monitoring nach dem Release notwendig, um UX-Qualität weiterzuentwickeln.

Viele Entscheidungen im Entwicklungsprozess beruhen auf Annahmen. Das Team nimmt an, dass Nutzerinnen und Nutzer eine Funktion finden, einen Begriff verstehen, einen Prozess akzeptieren oder eine Fehlermeldung richtig interpretieren. Nach dem Release können diese Annahmen anhand von Feedback, Supportdaten, Nutzungsdaten, Metriken oder gezielter Evaluation überprüft werden.

UX ist keine einmalige Aktivität. Produkte verändern sich, Nutzergruppen verändern sich, Anforderungen wachsen, und neue Funktionen beeinflussen bestehende Abläufe. Eine Oberfläche, die ursprünglich gut funktioniert hat, kann durch spätere Erweiterungen komplex, inkonsistent oder schwer verständlich werden.

Kontinuierliche Verbesserung bedeutet, UX regelmäßig zu beobachten, Probleme zu priorisieren und gezielt zu verbessern. AI kann diese Lernschleife unterstützen, indem sie Muster in Feedback, Supporttickets, Nutzungsdaten und Metriken herausarbeitet und Verbesserungshypothesen vorbereitet. Die Verantwortung für Interpretation, Priorisierung und Entscheidung bleibt beim Team.

9.8 UX-Metriken und HEART-Framework

UX-Metriken können helfen, Usability und Nutzungserfahrung nach dem Release messbar zu machen. Sie ersetzen keine qualitative Bewertung, zeigen aber, ob Nutzer Aufgaben erfolgreich abschließen, wo sie abbrechen, wie lange sie für zentrale Abläufe benötigen, wie häufig Fehler auftreten und ob wichtige Funktionen angenommen werden. Metriken sollten mit Nutzerzielen und Produktzielen verbunden sein. Eine Metrik ist nur hilfreich, wenn klar ist, was sie aussagt und welche Entscheidung sie unterstützen soll.

Aufgabenerfolg beziehungsweise Task Success beschreibt, ob Nutzerinnen und Nutzer eine relevante Aufgabe erfolgreich abschließen können. Diese Metrik ist für UX besonders wichtig, weil sie unmittelbar mit der Zielerreichung verbunden ist. Eine Aufgabe kann jedoch formal abgeschlossen sein, obwohl

Nutzerinnen und Nutzer unnötige Umwege nehmen, falsche Annahmen treffen oder nur mit erheblichem Aufwand zum Ziel kommen.

Fehlerrate beschreibt, wie häufig Nutzerinnen und Nutzer bei der Bearbeitung einer Aufgabe Fehler machen. Dazu können falsche Eingaben, abgebrochene Aktionen, wiederholte Validierungsfehler, Fehlklicks, Missverständnisse oder nicht korrekt ausgeführte Schritte gehören. Nicht jeder Fehler hat dieselbe Bedeutung. Deshalb sollte die Fehlerrate mit Schweregrad, Kontext und Nutzerwirkung verbunden werden.

Abbruchrate beschreibt, an welchen Stellen Nutzerinnen und Nutzer einen Prozess, eine Aufgabe oder eine Interaktion verlassen, bevor das gewünschte Ziel erreicht wurde. Eine hohe Abbruchrate kann auf Usability-Probleme hinweisen, etwa unklare nächste Schritte, zu hohe kognitive Belastung, fehlendes Vertrauen, technische Probleme oder unverständliche Fehlermeldungen. Ein Abbruch ist jedoch nicht automatisch ein Usability-Problem.

Bearbeitungszeit beziehungsweise Time on Task beschreibt, wie lange Nutzerinnen und Nutzer benötigen, um eine bestimmte Aufgabe abzuschließen. Eine lange Bearbeitungszeit kann auf unnötige Komplexität, unklare Begriffe, schlechte Informationsarchitektur, zu viele Schritte oder fehlende Orientierung hinweisen. Kürzer ist jedoch nicht immer besser, etwa bei sicherheitskritischen oder rechtlich relevanten Entscheidungen.

Supportaufkommen beschreibt, wie häufig Nutzerinnen und Nutzer Hilfe benötigen, Rückfragen stellen oder Probleme melden. Diese Metrik ist besonders hilfreich, weil Usability-Probleme häufig nicht direkt als Usability-Probleme benannt werden. Wiederkehrende Supportanfragen können auf unklare Abläufe, missverständliche Begriffe, fehlendes Systemfeedback, schlechte Auffindbarkeit oder unzureichende Fehlertoleranz hinweisen.

Das HEART-Framework strukturiert UX-Metriken in fünf Dimensionen:

HEART-Dimension Bedeutung Beispielhafte Metriken
Happiness Subjektive Bewertung der Nutzungserfahrung, etwa Zufriedenheit, Vertrauen oder wahrgenommene Einfachheit. Zufriedenheitsrating, Umfragewert, SUS, qualitative Rückmeldungen, Beschwerdehäufigkeit
Engagement Intensität oder Häufigkeit der Nutzung eines Produkts oder einer Funktion. Aktive Sitzungen, Interaktionen pro Nutzung, Nutzungshäufigkeit, aktive Tage
Adoption Annahme eines neuen Produkts, einer neuen Funktion oder eines neuen Workflows. Erstnutzung einer Funktion, Aktivierungsrate, Nutzung nach Einführung
Retention Wiederkehr oder fortgesetzte Nutzung über einen bestimmten Zeitraum. Wiederkehrrate, erneute Nutzung nach Zeitraum, Churn-Rate

Task Success Erfolgreiche Erledigung relevanter Aufgaben im Nutzungskontext. Abschlussrate, Fehlerrate, Bearbeitungszeit, Abbruchrate, Supporttickets zu einer Aufgabe

Wichtig ist die Übersetzung in Goals, Signals und Metrics. Dieses Verfahren wird als Goals–Signals–Metrics bezeichnet und gehört zum HEART-Framework. Goals beschreiben, welches Nutzer- oder Produktziel verbessert werden soll. Signals sind beobachtbare Hinweise darauf, ob dieses Ziel erreicht wird. Metrics sind konkrete Messgrößen, mit denen diese Hinweise erfasst werden. Erst dadurch wird aus einer allgemeinen UX-Dimension eine Kennzahl, die eine Entscheidung stützen kann.

HEART ist kein starres Pflichtschema. Nicht jede Dimension ist für jedes Produkt gleich wichtig. Für eine interne Fachanwendung kann Task Success wichtiger sein als Engagement. Für eine Lernplattform können Retention und Task Success gemeinsam relevant sein. AI kann mögliche HEART-Dimensionen, Goals, Signals und Metrics vorschlagen. Das Team muss entscheiden, welche Ziele tatsächlich relevant und welche Metriken interpretierbar und entscheidungsrelevant sind.

9.9 GenAI zur Strukturierung von Feedback, Tickets und Metriken

AI kann besonders hilfreich sein, um große Mengen unstrukturierter Rückmeldungen in mögliche Problemfelder für Usability zu überführen.

Insbesondere kann AI große Mengen an Freitextfeedback, Supporttickets, Produktbewertungen, Umfrageantworten oder internen Rückmeldungen clustern. Mögliche Cluster sind Navigation, Performance, Fehlermeldungen, Suche, Onboarding, Verständlichkeit, Accessibility, fehlende Funktionen oder Supportbedarf. Solche Cluster sind ein erster Ordnungsschritt. Sie zeigen Themen, beweisen aber noch keine Ursache und keine Priorität.

AI kann wiederkehrende Begriffe, Beschwerden, Wünsche oder Irritationen herausfiltern. Aus Rückmeldungen wie „Ich finde meine alten Rechnungen nicht“, „Wo sind abgeschlossene Rechnungen?“ und „Archiv ist unklar“ kann AI das Muster „Auffindbarkeit archivierter Rechnungen“ ableiten. Das Team muss anschließend prüfen, ob dieses Muster tatsächlich ein Usability-Problem beschreibt und welche Ursache wahrscheinlich ist.

AI kann Metriken erklären und mögliche Interpretationen vorschlagen. Wenn eine Funktion geringe Adoption hat, kann AI mögliche Ursachen nennen, etwa fehlende Sichtbarkeit, unklarer Nutzen, schlechte Kommunikation, falsche Zielgruppe, technische Probleme oder mangelnde Relevanz. Diese Ursachen sind Hypothesen. Sie müssen mit Daten, Nutzerfeedback, Supportinformationen, Usability-Tests, Domänenwissen oder technischer Analyse abgeglichen werden.

AI kann aus Feedback, Tickets, Evaluationsergebnissen, bekannten Schwächen und bestehenden

Backlog Items eine UX-Debt-Liste vorbereiten. Eine solche Liste kann Probleme beschreiben, betroffene Bereiche nennen, mögliche Auswirkungen formulieren und erste Priorisierungsvorschläge machen. Sie ist jedoch ein Arbeitsartefakt, das vom Team geprüft, bereinigt und priorisiert werden muss.

Bei AI-gestützter Auswertung sind Datenschutz und Vertraulichkeit zu beachten. Feedback, Supporttickets, Chatprotokolle, Nutzungsdaten und Freitexte können personenbezogene oder vertrauliche Informationen enthalten. Je nach Kontext können Anonymisierung, Pseudonymisierung, Aggregation oder der Einsatz freigegebener Unternehmenssysteme notwendig sein.

9.10 Muster, Korrelation, Ursache und Priorität unterscheiden

Für Entscheidungen über Usability reicht es nicht aus, Muster in Daten zu finden. Teams müssen unterscheiden, ob ein Muster tatsächlich ein Problem zeigt, ob es nur mit einem anderen Ereignis korreliert, welche Ursache plausibel ist und ob daraus eine priorisierte Produktentscheidung folgen sollte.

Ein Muster ist eine wiederkehrende Beobachtung. Mehrere Nutzerinnen und Nutzer melden dasselbe Problem, viele Tickets betreffen dieselbe Funktion, oder Analytics zeigen wiederholte Abbrüche an einer bestimmten Stelle. Muster sind wichtig, weil sie Aufmerksamkeit lenken. Sie sind aber noch keine Erklärung.

Eine Korrelation beschreibt, dass zwei Dinge gemeinsam auftreten. Beispielsweise brechen Nutzerinnen und Nutzer häufiger auf mobilen Geräten ab als am Desktop. Das kann ein wichtiger Hinweis sein, aber noch keine sichere Ursache. Möglich sind unterschiedliche Ursachen, etwa schlechtes Layout auf kleinen Bildschirmen, langsamere Verbindung, andere Nutzungssituation oder ein technischer Fehler.

Eine Ursache erklärt, warum ein Problem entsteht. Ursachen zu erkennen ist schwieriger als Muster zu erkennen. Dafür braucht es oft zusätzliche Daten, Usability-Tests, Nutzerbeobachtung, Supportanalyse, technische Analyse oder Domänenwissen. AI kann Ursachenhypothesen formulieren, Ursachen aber nicht allein aus einem Muster beweisen.

Priorität beschreibt, was als Nächstes bearbeitet werden soll. Priorität ergibt sich nicht allein aus Häufigkeit. Sie hängt auch von Schweregrad, Nutzerwirkung, Risiko, Business-Relevanz, technischer Machbarkeit, regulatorischer Relevanz, Accessibility-Relevanz und strategischer Bedeutung ab. AI kann Priorisierungsvorschläge liefern, aber sie kennt den vollständigen Kontext nicht.

Eine belastbare Produktentscheidung kombiniert mehrere Informationsquellen: Nutzerfeedback, Supportdaten, Nutzungsdaten, Evaluationsergebnisse, Domänenwissen, technische Machbarkeit, Business-Ziele und Risiken. AI kann diese Informationen strukturieren, mögliche Zusammenhänge aufzeigen und Entscheidungsoptionen formulieren.

Der zentrale Lernpunkt lautet: Daten zeigen Hinweise. AI erkennt Muster. Menschen treffen verantwortete Entscheidungen.

9.11 UX-Debt erkennen

UX-Debt bezeichnet aufgeschobene oder unvollständig gelöste Probleme der Nutzungserfahrung. In Softwareentwicklungsteams zeigt sich UX-Debt häufig sehr konkret als Usability-Problem: unvollständige Systemzustände, inkonsistente Komponenten, unklare Texte, fehlende Accessibility-Basics, schwer auffindbare Funktionen, unnötige Arbeitsschritte oder ungelöste Usability-Probleme.

UX-Debt entsteht, wenn Usability-Probleme nicht gelöst, nur provisorisch behandelt oder durch schnelle Entscheidungen erzeugt werden. Ähnlich wie technischer Debt kann UX-Debt später höhere Kosten verursachen, weil Nutzerprobleme, Inkonsistenzen, Workarounds, Supportaufwand und Nachbesserungen zunehmen.

UX-Debt ist nicht jede Verbesserungsidee. Gemeint sind Schwächen, die Nutzbarkeit, Verständlichkeit, Konsistenz, Accessibility, Effizienz oder Zufriedenheit beeinträchtigen oder künftige Änderungen erschweren.

Typische Formen von UX-Debt sind:

- unvollständige Systemzustände und fehlendes Systemfeedback nach Aktionen,
- inkonsistente Komponenten, uneinheitliche Formulare und uneinheitliche Begriffe,
- schlechte Fehlermeldungen und unklare Navigation,
- fehlende oder unvollständige Accessibility,
- nicht geprüfte AI-generierte Entwürfe und veraltete Hilfetexte,
- zu hohe kognitive Belastung und ungelöste Usability-Probleme,
- nicht dokumentierte Workarounds und wiederkehrende Supportprobleme ohne nachhaltige Lösung.

In gewachsenen Anwendungen entsteht UX-Debt häufig über längere Zeit. Verschiedene Teams, unterschiedliche Frameworks, schnelle Features, wechselnde Prioritäten oder AI-generierte Einzellösungen können zu Inkonsistenzen führen. AI kann helfen, solche Inkonsistenzen zu identifizieren und als UX-Debt zu dokumentieren. Das Team muss entscheiden, welche Variante künftig gelten soll, welche Unterschiede fachlich notwendig sind und welche Inkonsistenzen abgebaut werden sollen.

UX-Debt sollte dokumentiert werden. Eine gute Dokumentation nennt betroffenen Bereich, beobachtetes Problem, Auswirkung auf Nutzerinnen und Nutzer, Evidenz oder Quelle, mögliche Maßnahme, Risiko bei Nichtbehebung, grobe Priorität und nächsten Schritt.

UX-Debt und Technical Debt können zusammenhängen, sind aber nicht identisch. Technical Debt betrifft vor allem technische Struktur, Wartbarkeit, Architektur oder Codequalität. UX-Debt betrifft die Nutzungserfahrung, etwa Verständlichkeit, Konsistenz, Auffindbarkeit, Accessibility oder Fehlertoleranz.

9.12 Erkenntnisse in Backlog Items und Verbesserungsmaßnahmen überführen

Erkenntnisse nach dem Release schaffen erst dann Wert, wenn sie in bearbeitbare Artefakte überführt werden. Für Softwareentwicklungsteams bedeutet das: Beobachtungen, Feedback, Metriken und AI-gestützte Hypothesen müssen in Backlog Items, Akzeptanzkriterien, UX-Debt-Maßnahmen, technische Aufgaben oder neue Discovery-Fragen übersetzt werden.

Ein Muster aus Feedback oder Daten wird erst dann handlungsfähig, wenn daraus ein Backlog Item, eine Discovery-Frage oder eine Verbesserungsmaßnahme entsteht.

Beispiel Muster:
„Viele Nutzerinnen und Nutzer melden, dass sie archivierte Dokumente nicht finden."

Mögliches Backlog Item:
„Als Fachnutzerin möchte ich archivierte Dokumente über Suche und Filter auffinden können, damit ich abgeschlossene Vorgänge bei Rückfragen schnell wiederfinden kann."

Mögliche Discovery-Frage:
„Welche Begriffe verwenden Nutzerinnen und Nutzer für archivierte Dokumente, und wo erwarten sie diese im System?"

AI kann solche Übersetzungen vorschlagen. Das Team muss prüfen, welche Form sinnvoll ist. Wenn Ursache und Nutzerziel klar genug sind, kann ein Backlog Item entstehen. Wenn Ursache oder Nutzerziel unklar sind, sollte zunächst eine Discovery-Frage formuliert werden.

UX-Debt-Maßnahmen sollten konkret sein. „UX verbessern" ist zu unpräzise. Besser sind Maßnahmen wie Fehlermeldungen im Registrierungsprozess vereinheitlichen, Empty States für Projektlisten ergänzen, Tabellenfilter konsolidieren, Fokusführung im Dialogsystem prüfen, Button-Benennungen vereinheitlichen oder Kontrastprobleme in sekundären Aktionen beheben.

Nicht jedes Problem sollte sofort in eine Lösung übersetzt werden. Wenn Ursache oder Nutzerziel unklar sind, ist eine neue Discovery-Frage sinnvoll. Wenn Analytics zeigt, dass eine Funktion selten verwendet wird, können mögliche Ursachen fehlender Bedarf, schlechte Auffindbarkeit, unklarer Nutzen, unpassender Zeitpunkt im Workflow, fehlende Berechtigung oder technische Probleme sein.

Verbesserungsmaßnahmen sollten anhand von Nutzerwirkung, Risiko, Häufigkeit, Aufwand, strategischer Bedeutung, Accessibility-Relevanz, regulatorischer Bedeutung und technischer Abhängigkeit priorisiert werden. AI kann eine erste Sortierung vorschlagen, aber Priorisierung bleibt eine Team- und Produktentscheidung.

Wenn eine Verbesserungsmaßnahme umgesetzt wurde, sollte ihre Wirkung beobachtet werden. Dazu können erneute Usability-Tests, Nutzungsdaten, Supportticket-Entwicklung, Feedback oder definierte UX-Metriken genutzt werden. Damit schließt sich die Lernschleife: Hinweise werden erkannt, Hypothesen formuliert, Maßnahmen umgesetzt und deren Wirkung erneut beobachtet.

9.13 Kontinuierliche Verbesserung im Team verankern

Kontinuierliche Verbesserung bedeutet, Usability regelmäßig in Teamarbeit, Priorisierung und Produktentscheidungen einzubeziehen. Feedback, Supportdaten, Metriken und UX-Debt sollten nicht nur gesammelt, sondern wiederkehrend bewertet, priorisiert und in die Produktentwicklung zurückgeführt werden.

Teams können Usability-Erkenntnisse regelmäßig reflektieren. Dies kann Teil einer Retrospektive, eines Review-Formats, eines Refinements oder eines eigenen kurzen Usability-Checkpoints sein. Entscheidend ist nicht das konkrete Meetingformat, sondern dass Usability-Erkenntnisse aus realer Nutzung, Feedback, Supportdaten, Metriken und Evaluationsergebnissen regelmäßig betrachtet werden.

Geeignete Fragen sind: Welche Usability-Probleme sind wiederholt aufgetreten? Welche Annahmen wurden bestätigt, widerlegt oder bleiben offen? Welche Feedbackthemen häufen sich? Welche Supporttickets deuten auf Verständlichkeits- oder Auffindbarkeitsprobleme hin? Welche UX-Debt-Punkte wachsen an? Welche Verbesserung sollte als Nächstes eingeplant werden?

UX-Debt sollte sichtbar bleiben, etwa als eigene Kategorie im Backlog, als Label, als Liste, als Qualitätsboard oder als Teil eines regelmäßigen Produktqualitätsreviews. Sichtbarkeit verhindert, dass kleine Probleme dauerhaft liegen bleiben und sich über mehrere Releases zu größeren Usability-Problemen verdichten.

Auch der AI-Einsatz selbst sollte reflektiert werden. Teams können beobachten, wo AI gute Vorschläge geliefert hat, wo Kontext fehlte, wo Halluzinationen auftraten und welche Prompts besonders hilfreich waren. Dadurch verbessert sich nicht nur Usability-Arbeit, sondern auch AI-Kompetenz im Team.

Die Erkenntnisse nach dem Release fließen zurück in Discovery, Backlog Refinement, Design, Umsetzung, Evaluation und Testing. Damit entsteht ein kontinuierlicher Kreislauf:

Feedback und Daten → Muster erkennen → Hypothesen formulieren → Ursachen prüfen → priorisieren → Backlog Items oder UX-Debt-Maßnahmen ableiten → umsetzen → testen → erneut beobachten.

AI kann diesen Kreislauf unterstützen, indem sie strukturiert, zusammenfasst, Hypothesen erzeugt und Dokumentation vorbereitet. Monitoring ohne Rückkopplung erzeugt Berichte; Monitoring mit Rückkopplung erzeugt Lernen.

9.14 Externe Unterstützung und Grenzen teaminterner AI-Nutzung

Externe Usability-, Research-, Accessibility- oder Fachexpertise ist besonders wichtig, wenn Probleme hohe Risiken haben. Das betrifft sicherheitskritische Prozesse, rechtliche Anforderungen, medizinische oder finanzielle Anwendungen, große Nutzergruppen, barrierefreie Nutzung, geschäftskritische Abläufe oder Entscheidungen mit erheblicher Auswirkung auf Nutzerinnen und Nutzer.

Externe Unterstützung kann auch sinnvoll sein, wenn ein Team nicht sicher ist, welche Methode geeignet ist. Das betrifft etwa komplexe Usability-Tests, quantitative UX-Messung, Accessibility-Audits, Research-Design, A/B-Tests, statistische Interpretation oder strategische Usability-Entscheidungen.

Wenn bestimmte Usability-Probleme über mehrere Releases bestehen bleiben, häufige Supportfälle verursachen oder durch interne Maßnahmen nicht gelöst werden, sollte zusätzliche Expertise erwogen werden. Wiederkehrende Probleme können darauf hinweisen, dass die Ursache tiefer liegt, etwa in Informationsarchitektur, mentalem Modell, Produktstrategie, Fachprozess, Accessibility, technischer Architektur oder organisatorischen Rahmenbedingungen.

AI-gestützte Selbsthilfe ist wertvoll, aber sie hat Grenzen. Wenn hohe Unsicherheit, hohe Auswirkung oder hohe Verantwortung besteht, braucht es fachlich belastbare Prüfung. AI kann solche Situationen nicht auflösen, sondern höchstens Hinweise liefern, dass weitere Prüfung notwendig ist.

Externe Usability-, Research-, Accessibility-, Datenanalyse- oder Domänenexpertise ist besonders dann notwendig, wenn Entscheidungen hohe Risiken haben, Nutzergruppen schwer erreichbar sind, Daten widersprüchlich sind, Accessibility-Konformität verbindlich geprüft werden muss oder AI-Auswertungen nicht ausreichend belastbar sind.

9.15 AI-gestütztes Monitoring und menschliche Prüfung

AI kann nach dem Release besonders gut große Mengen unstrukturierter Informationen strukturieren, UX-Debt aufdecken und erste Verbesserungshypothesen formulieren.

AI unterstützt gut bei	Feedback und Supporttickets clustern und zusammenfassen; wiederkehrende Themen aufzeigen; Freitexte auswerten; technische Signale mit Usability-Hypothesen verbinden; Metriken erklären; Ursachenhypothesen formulieren; UX-Debt-Listen vorbereiten; Verbesserungsmaßnahmen und Backlog Items vorschlagen; neue Discovery-Fragen ableiten; Reports für Reviews, Retrospektiven oder Produktentscheidungen vorbereiten.
Grenzen und typische Risiken	Beweist keine Ursachen und setzt keine verbindlichen Produktprioritäten; Business-Kontext, Risiko, Strategie, regulatorische Anforderungen und technische Abhängigkeiten fehlen häufig; typische Risiken: Muster werden als Ursachen und Korrelationen als Beweise interpretiert, Feedback wird als repräsentativ angenommen, lautes Feedback verdrängt stille, aber wichtige Probleme, AI-Priorisierung wird ungeprüft übernommen, UX-Debt wird dokumentiert, aber nicht bearbeitet, technische Metriken werden ohne Usability-Kontext interpretiert, Datenschutz und Vertraulichkeit werden bei Auswertungen vernachlässigt.
Menschlich zu prüfen	Aus welchen Quellen Hinweise stammen und ob die Daten repräsentativ oder verzerrt sind; ob personenbezogene oder vertrauliche Daten betroffen sind und ob Datenschutz, Anonymisierung und Berechtigungen berücksichtigt wurden; ob es sich um ein Muster, eine Korrelation oder eine mögliche Ursache handelt; welche alternativen Erklärungen möglich sind; welche Nutzergruppe betroffen und wie schwer die Auswirkung ist; welche Business-, Accessibility- oder regulatorische Relevanz besteht; ob Maßnahmen konkret genug für das Backlog sind.

AI-gestütztes Monitoring soll nicht nur Berichte erzeugen, sondern zu belastbaren Verbesserungsentscheidungen führen; Interpretation, Priorisierung und Entscheidung bleiben beim Team beziehungsweise bei den verantwortlichen Rollen.

10 Literatur

Die folgende Liste enthält zentrale Standards, fachliche Grundlagen und Referenzen, auf denen dieser Syllabus aufbaut. Sie dient der fachlichen Einordnung und Vertiefung.

Prüfungsrelevant sind externe Quellen nur insoweit, wie ihre Inhalte in diesem Syllabus ausdrücklich behandelt werden. Eine vollständige Beherrschung der genannten Standards, Bücher oder Online-Ressourcen ist für die Foundation-Level-Prüfung nicht erforderlich.

Weiterführende Literatur, Praxisbeispiele, Prompt-Beispiele, Tool-Hinweise und vertiefende Quellen werden in separaten Begleitmaterialien bereitgestellt.

Basisliteratur und zentrale Referenzen

Normen und Standards

[1] ISO 9241-11, Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts.

[2] ISO 9241-210, Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.

[3] ISO/IEC 25010, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

[4] W3C, Web Content Accessibility Guidelines (WCAG) 2.2.

[5] EN 301 549, Accessibility requirements for ICT products and services.

[6] ISO/IEC 40500, Information technology – W3C Web Content Accessibility Guidelines (WCAG) 2.0.

Usability, User Experience und Human-Centred Design

[7] J. Nielsen, “10 Usability Heuristics for User Interface Design,” Nielsen Norman Group, 1994.

[8] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1990.

[9] Nielsen Norman Group, Articles and resources on usability, user experience, usability testing, information architecture and interaction design.

Agile Entwicklung

[10] K. Schwaber and J. Sutherland, The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.

Artificial Intelligence und Governance

[11] European Union, Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act).

[12] NIST, Artificial Intelligence Risk Management Framework (AI RMF 1.0).

Begleitende Materialien

[13] R. Pucher and M. Simandl, UX Design mit AI. Begleitendes Lehr- und Praxisbuch zum Syllabus.

[14] UXQCC, Begleitende Prompt-Bibliothek und Praxisbeispiele zum UXQCC Certified Professional in Software Usability with GenAI – Foundation Level.

Hinweis für akkreditierte Trainingsprovider

Für akkreditierte Trainingsprovider stehen zusätzlich zur öffentlichen Syllabus-Fassung ausführlichere Trainings- und Begleitmaterialien zur Verfügung. Dazu können insbesondere eine Provider-Fassung des Syllabus, didaktische Hinweise, Praxisbeispiele, Prompt-Beispiele, Tool-Hinweise, Übungsvorschläge und weiterführende Literatur gehören.

Diese Materialien unterstützen die konkrete Trainingsgestaltung und Vertiefung. Prüfungsrelevant bleiben die in diesem Syllabus formulierten Business Outcomes, Learning Objectives und ausdrücklich behandelten Inhalte.

11 Anhang 1 – Kurzglossar zentraler Begriffe

Dieses Kurzglossar erläutert zentrale Begriffe, die für das Verständnis dieses Syllabus wesentlich sind. Es dient der einheitlichen Verwendung wichtiger Begriffe aus User Experience, Usability, Accessibility, agiler Softwareentwicklung und Generativer AI.

Das Kurzglossar ersetzt kein umfassendes Fachglossar zu UX, Accessibility, Software Engineering oder AI. Begriffe sind nur insoweit prüfungsrelevant, wie sie in den Business Outcomes, Learning Objectives und Kapitelinhalten dieses Syllabus behandelt werden.

Begriff	**Erklärung**
Accessibility / Barrierefreiheit	Accessibility bezeichnet die Gestaltung digitaler Systeme so, dass sie auch von Menschen mit unterschiedlichen Fähigkeiten, Einschränkungen und Nutzungssituationen genutzt werden können. Accessibility ist Bestandteil guter UX und professioneller Softwarequalität. Wichtige Bezugspunkte sind unter anderem WCAG 2.2, EN 301 549 und ISO/IEC 40500.
AI Governance	AI Governance bezeichnet Regeln, Zuständigkeiten und Prüfprozesse für den verantwortungsvollen Einsatz von AI in einer Organisation. Dazu gehören unter anderem Datenschutz, Transparenz, Dokumentation, Qualitätssicherung, menschliche Prüfung und Verantwortlichkeit.
AI-Output	AI-Output ist das Ergebnis, das von einem AI-System erzeugt wurde. Im Usability-Kontext ist AI-Output als Vorschlag, Hypothese, Entwurf oder Strukturierungshilfe zu behandeln, nicht als geprüfte Wahrheit.
AI-gestützte Analyse	AI-gestützte Analyse bezeichnet die Nutzung von AI zur Strukturierung, Zusammenfassung oder ersten Einschätzung vorhandener Informationen. Sie kann Hinweise und Hypothesen liefern, ersetzt aber keine fachliche Prüfung, Evaluation oder Validierung.
AI-gestützte Auswertung	AI-gestützte Auswertung bezeichnet die Nutzung von AI zur Ordnung, Clusterung, Zusammenfassung oder Aufbereitung von Daten, Feedback, Beobachtungen oder Findings. Die Ergebnisse müssen menschlich geprüft werden, insbesondere hinsichtlich Evidenz, Ursache, Priorität und fachlicher Relevanz.
AI-Simulation	AI-Simulation bezeichnet eine von AI erzeugte hypothetische Darstellung möglicher Nutzerreaktionen, Nutzungssituationen oder Perspektiven. Sie

Begriff	Erklärung
	kann Hypothesen erzeugen, ersetzt aber keine Beobachtung realer Nutzerinnen und Nutzer und keine Validierung.
Akzeptanzkriterien	Akzeptanzkriterien sind prüfbare Bedingungen, die erfüllt sein müssen, damit ein Backlog Item als umgesetzt gilt. Usability-relevante Akzeptanzkriterien können etwa Verständlichkeit, Systemzustände, Fehlerfälle, Systemfeedback, erwartbares Verhalten oder Accessibility-Basics betreffen.
Analytics	Analytics bezeichnet die Erfassung und Auswertung von Nutzungsdaten, etwa Klickpfaden, Abbrüchen, Nutzungshäufigkeit oder Prozessabschlüssen. Analytics zeigt vor allem, was passiert, erklärt aber nicht automatisch, warum es passiert.
Anonymisierung	Anonymisierung bezeichnet die Veränderung von Daten so, dass Personen nicht mehr identifizierbar sind. Sie ist besonders relevant, wenn Feedback, Supporttickets, Nutzungsdaten oder Freitexte mit AI ausgewertet werden.
ARIA	ARIA bezeichnet technische Attribute, mit denen zusätzliche Informationen für assistive Technologien bereitgestellt werden können. ARIA sollte nur eingesetzt werden, wenn native HTML-Elemente nicht ausreichen. Falsch eingesetztes ARIA kann Accessibility verschlechtern.
Aufgabe / Task	Eine Aufgabe ist eine Tätigkeit, die Nutzerinnen und Nutzer mit einem System ausführen, um ein Nutzerziel zu erreichen. Aufgaben stehen immer im Zusammenhang mit Nutzerziel und Nutzungskontext.
Aufmerksamkeit	Aufmerksamkeit bezeichnet die begrenzte Fähigkeit, bestimmte Informationen bewusst zu verarbeiten. Benutzeroberflächen müssen Aufmerksamkeit gezielt unterstützen und dürfen Nutzerinnen und Nutzer nicht durch unnötige Informationen, unklare Prioritäten oder widersprüchliche Signale überfordern.
Aufgabenerfolg / Task Success	Aufgabenerfolg beschreibt, ob Nutzerinnen und Nutzer eine relevante Aufgabe erfolgreich abschließen können. Task Success ist eine wichtige UX-Metrik und eng mit Effektivität verbunden.
Backlog Item	Ein Backlog Item ist ein Eintrag im Product Backlog oder in einer vergleichbaren Arbeitsliste. Es beschreibt eine Anforderung, Aufgabe, Verbesserung, Korrektur oder Klärung, die priorisiert und umgesetzt werden kann.

Begriff	Erklärung
Backlog Refinement	Backlog Refinement bezeichnet die laufende Verfeinerung von Backlog Items im Team. Dabei werden Einträge geklärt, zerlegt, priorisiert und mit Akzeptanzkriterien versehen. Für Usability ist das Refinement zentral, weil hier Nutzerziel, Nutzungskontext, Systemzustände, Fehlerfälle und Accessibility-Basics benannt werden können.
Bedienbarkeit	Bedienbarkeit bedeutet, dass ein System tatsächlich gesteuert und verwendet werden kann. Dazu gehören unter anderem Tastaturbedienbarkeit, sichtbarer Fokus, ausreichende Zielgrößen, vorhersehbare Interaktion und die Nutzbarkeit mit verschiedenen Eingabemethoden.
Beobachtung	Beobachtung bezeichnet das systematische Erfassen dessen, was Nutzerinnen und Nutzer tatsächlich tun, sagen, suchen, übersehen, missverstehen oder abbrechen. Im Usability-Test ist Beobachtung zentral und von Interpretation zu unterscheiden.
Business-Relevanz	Business-Relevanz beschreibt die Bedeutung eines Usability-Problems oder einer Verbesserung für Geschäftsziele, Produktstrategie, Kosten, Risiken, Nutzung, Kundenbindung oder Supportaufwand. Sie ist ein Kriterium für Priorisierung.
Checkbox	Eine Checkbox ist ein UI-Control für unabhängige Auswahlmöglichkeiten. Mehrere Checkboxes können gleichzeitig ausgewählt sein. Sie sollte nicht für gegenseitig ausschließende Optionen verwendet werden.
Code Review	Code Review bezeichnet die strukturierte Prüfung von Code durch andere Personen. Bei AI-generiertem Code sollte neben Funktionalität auch auf Usability-relevante Aspekte wie Systemzustände, Fehlermeldungen, Fokusführung, Accessibility, Security und Wartbarkeit geachtet werden.
Cognitive Walkthrough	Der Cognitive Walkthrough ist ein expertenbasiertes Verfahren, bei dem ein Nutzungspfad Schritt für Schritt aus Sicht einer Nutzerin oder eines Nutzers durchgegangen wird. Das Verfahren arbeitet mit vier Leitfragen: ob Nutzerinnen und Nutzer erkennen, was der nächste sinnvolle Schritt ist, ob sie das richtige Element finden, ob sie verstehen, dass dieses Element zur Aufgabe passt, und ob sie nach der Aktion erkennen, ob sie erfolgreich waren.
Component Workshop	Ein Component Workshop ist eine Entwicklungsumgebung, in der einzelne UI-Komponenten losgelöst vom Rest der Anwendung in

Begriff	Erklärung
	unterschiedlichen Varianten und Systemzuständen dargestellt, durchgeklickt und geprüft werden. Jede dargestellte Ausprägung einer Komponente wird als Story bezeichnet. Ein Component Workshop unterstützt Entwicklung, Handoff, Testing und Designsystemarbeit.
Control	Ein Control ist ein bedienbares UI-Element, etwa Button, Link, Checkbox, Radio Button, Dropdown, Toggle oder Tab. Controls sollten entsprechend ihrem erwarteten mentalen Modell verwendet werden.
Datenschutz	Datenschutz bezeichnet den Schutz personenbezogener Daten. Im Kontext von AI-gestützter Usability-Arbeit ist besonders zu prüfen, welche Daten in AI-Systeme eingegeben, gespeichert, verarbeitet oder weitergegeben werden dürfen.
Definition of Done / DoD	Die Definition of Done ist eine Teamvereinbarung, wann eine Arbeit als fertig gilt. Usability-Aspekte können darin ausdrücklich aufgenommen werden, etwa geprüfte Systemzustände, verständliche Fehlermeldungen, Accessibility-Basics oder erfüllte Usability-Akzeptanzkriterien.
Definition of Ready / DoR	Die Definition of Ready ist eine Teamvereinbarung, wann ein Backlog Item ausreichend vorbereitet ist, um in die Umsetzung beziehungsweise in einen Sprint aufgenommen zu werden. Usability-relevant sind etwa klares Nutzerziel, bekannter Nutzungskontext, offene Annahmen, Systemzustände, Fehlerfälle und Validierungsbedarf.
Design Critique	Eine Design Critique ist ein strukturiertes Review-Format zur fachlichen Bewertung von Entwürfen anhand nachvollziehbarer Kriterien wie Nutzerziel, Verständlichkeit, Erwartungskonformität, Konsistenz, Systemzustände und Accessibility. Sie unterscheidet sich von einer Geschmacksdiskussion dadurch, dass Kriterien und Ergebnisse dokumentiert werden.
Design Token	Design Tokens sind benannte Werte für Gestaltungseigenschaften wie Farben, Abstände, Schriftgrößen, Border-Radius oder Schatten. Sie verbinden Design und Code und unterstützen Konsistenz, Wartbarkeit und Designsysteme.
Designsystem	Ein Designsystem beschreibt wiederverwendbare UI-Komponenten, Gestaltungsregeln, Interaktionsmuster, Texte, Systemzustände und Qualitätskriterien. Es unterstützt konsistente, zugängliche und wartbare Benutzeroberflächen.

Begriff	Erklärung
Discovery	Discovery bezeichnet Aktivitäten, mit denen Nutzer, Nutzungskontext, Aufgaben, Probleme, Ziele, Risiken und Hypothesen vor oder während der Umsetzung besser verstanden werden.
Edge Case	Ein Edge Case ist ein seltener, aber möglicher Sonderfall. Edge Cases sind wichtig, weil sie in realer Nutzung zu Unsicherheit, Fehlern, Datenverlust oder Supportbedarf führen können.
Effektivität	Effektivität beschreibt, ob Nutzerinnen und Nutzer ihre Ziele mit einem System korrekt und vollständig erreichen können. Im Softwarekontext betrifft das zum Beispiel erfolgreiche Aufgabenerledigung, richtige Eingaben oder das Finden relevanter Funktionen.
Effizienz	Effizienz beschreibt den Aufwand, mit dem Nutzerinnen und Nutzer ihre Ziele erreichen. Dazu zählen unter anderem Zeit, Anzahl der Schritte, Suchaufwand, unnötige Wiederholungen und kognitive Belastung.
Empty State	Ein Empty State ist ein Systemzustand, in dem keine Daten oder Inhalte vorhanden sind. Gute Empty States erklären, warum nichts angezeigt wird und welche nächste Handlung möglich ist.
Error State	Ein Error State ist ein Systemzustand, in dem ein Fehler aufgetreten ist. Gute Error States erklären verständlich, was passiert ist, welche Auswirkung dies hat und wie Nutzerinnen und Nutzer weiter vorgehen können.
Evaluation	Evaluation bezeichnet die systematische Bewertung eines Entwurfs, Prototyps, Produkts oder Nutzungspfads mit Blick auf UX, Usability, Nutzerziel und Nutzungskontext. Evaluation kann mit realen Nutzerinnen und Nutzern erfolgen, etwa durch Usability-Tests, oder expertenbasiert, etwa durch heuristische Evaluation oder Cognitive Walkthrough.
Evidenz	Evidenz bezeichnet die Grundlage, auf der eine Aussage, ein Finding oder eine Entscheidung beruht. Evidenz kann aus Beobachtungen, Tests, Nutzungsdaten, Supportdaten, Domänenwissen oder fachlicher Prüfung stammen. AI-Ausgaben sind ohne Prüfung keine belastbare Evidenz.
Expert Review / expertenbasiertes Verfahren	Ein Expert Review ist eine strukturierte Prüfung durch Fachpersonen oder UX-Expertinnen und -Experten. Beispiele sind heuristische Evaluation oder Cognitive Walkthrough. Expert Reviews können mögliche Usability-Probleme aufspüren, ersetzen aber keine Usability-Tests mit realen Nutzerinnen und Nutzern.

Begriff	Erklärung
Erwartungskonformität	Erwartungskonformität bedeutet, dass ein System sich so verhält, wie Nutzerinnen und Nutzer es aufgrund ihrer Erfahrung, Aufgabe und Situation erwarten. Erwartungskonformität reduziert Lernaufwand, Irritation und Fehlbedienungen.
Feedback	Feedback bezeichnet in diesem Syllabus Rückmeldungen von Nutzerinnen und Nutzern, Kunden, Support, Betrieb oder anderen Stakeholdern zur tatsächlichen Nutzung oder zum produktiven Umfeld eines Systems. Davon zu unterscheiden ist Systemfeedback, also die Rückmeldung des Systems an Nutzerinnen und Nutzer während der Interaktion.
Fehlerrate	Die Fehlerrate beschreibt, wie häufig Nutzerinnen und Nutzer bei einer Aufgabe Fehler machen, etwa falsche Eingaben, wiederholte Validierungsfehler, Fehlklicks oder missverstandene Schritte. Sie muss im Zusammenhang mit Schweregrad, Kontext und Nutzerwirkung interpretiert werden.
Fehlermeldung	Eine Fehlermeldung informiert Nutzerinnen und Nutzer über ein Problem und sollte verständlich, konkret, handlungsleitend und dem betroffenen Bereich zugeordnet sein. Gute Fehlermeldungen unterstützen Fehlerbehebung und vermeiden unnötige Frustration.
Finding / UX-Finding	Ein Finding ist eine dokumentierte Erkenntnis aus Evaluation, Usability-Test, Expert Review, Supportanalyse oder anderer Prüfung. Ein gutes Finding beschreibt Beobachtung, Kontext, Auswirkung, Evidenz und gegebenenfalls Ursache oder Empfehlung.
Fokusführung	Fokusführung beschreibt, wie der Eingabefokus in einer Oberfläche sichtbar, logisch und erwartbar geführt wird. Sie ist besonders wichtig für Tastaturbedienbarkeit, Screenreader-Nutzung, Dialoge, Menüs und komplexe Formulare.
Formative und summative Evaluation	Formative Evaluation dient der Verbesserung während der Entwicklung, etwa durch frühe Usability-Tests an Prototypen. Summative Evaluation bewertet eine weiter fortgeschrittene oder fertige Lösung, etwa zur Abnahme oder zum Vergleich. Beide Formen können expertenbasiert oder mit realen Nutzerinnen und Nutzern erfolgen.
Frontend	Frontend bezeichnet den technisch implementierten Teil eines digitalen Systems, mit dem Nutzerinnen und Nutzer direkt interagieren. Dazu

Begriff	Erklärung
	gehören UI-Code, Komponenten, Interaktionslogik, Systemzustände, Darstellung und Accessibility-Eigenschaften.
GenAI	GenAI steht für Generative Artificial Intelligence, also generative künstliche Intelligenz. GenAI kann neue Inhalte erzeugen, etwa Texte, Code, UI-Vorschläge, Zusammenfassungen, Testideen, Analysevorschläge oder Prototypen.
Gestaltgesetze	Gestaltgesetze beschreiben Prinzipien der menschlichen Wahrnehmung, etwa Nähe, Ähnlichkeit, gemeinsame Region, Geschlossenheit oder gute Gestalt. Sie helfen zu verstehen, wie Nutzerinnen und Nutzer visuelle Elemente gruppieren und strukturieren.
Goals–Signals–Metrics	Goals–Signals–Metrics ist das Verfahren, mit dem aus einer allgemeinen UX-Dimension eine entscheidungsrelevante Kennzahl wird. Goals beschreiben, welches Nutzer- oder Produktziel verbessert werden soll. Signals sind beobachtbare Hinweise darauf, ob dieses Ziel erreicht wird. Metrics sind die konkreten Messgrößen, mit denen diese Hinweise erfasst werden. Das Verfahren gehört zum HEART-Framework.
Halluzination	Eine Halluzination ist eine plausibel wirkende, aber falsche oder nicht belegte Ausgabe eines AI-Systems. Halluzinationen sind besonders riskant, wenn sie unkritisch in Anforderungen, Designs, Tests, Berichte oder Code übernommen werden.
Handoff	Handoff bezeichnet den Abgleich und die Übergabe von Informationen zwischen Design, AI-Unterstützung und Entwicklung. Ein guter Handoff beschreibt nicht nur, wie etwas aussieht, sondern auch Nutzerziel, Komponenten, Zustände, Verhalten, Texte, Accessibility-Anforderungen, Akzeptanzkriterien und offene Annahmen.
Happy Path	Der Happy Path ist der ideale Ablauf, bei dem alles wie geplant funktioniert. Gute UX berücksichtigt zusätzlich Fehlerfälle, Abbrüche, leere Zustände, alternative Pfade, Berechtigungen und Sonderfälle.
HEART-Framework	Das HEART-Framework strukturiert UX-Metriken in die Dimensionen Happiness, Engagement, Adoption, Retention und Task Success. Es hilft, UX-Ziele, Signale und Metriken systematisch miteinander zu verbinden.
Heuristische Evaluation	Die heuristische Evaluation ist ein expertenbasiertes Verfahren, bei dem ein Interface anhand definierter Usability-Heuristiken oder Gestaltungsprinzipien geprüft wird. Sie liefert Hinweise auf mögliche

Begriff	Erklärung
	Usability-Probleme, ersetzt aber keinen Usability-Test mit realen Nutzerinnen und Nutzern.
High-Fidelity-Prototyp	Ein High-Fidelity-Prototyp hat einen hohen Detailgrad und ähnelt stärker dem späteren Produkt. Er kann visuelle Gestaltung und realistischere Interaktionen enthalten, kann aber auch vorschnell den Eindruck einer fertigen Lösung erzeugen.
Horizontaler Prototyp	Ein horizontaler Prototyp zeigt viele Bereiche oder Screens eines Systems in der Breite, arbeitet einzelne Funktionen aber nur oberflächlich aus. Er eignet sich besonders zur Diskussion von Struktur, Navigation und Umfang.
Human-Centered Design / Menschenzentrierte Gestaltung	Human-Centered Design ist ein Gestaltungsansatz, bei dem Nutzerinnen und Nutzer, ihre Ziele, Anforderungen, Fähigkeiten und Nutzungskontexte systematisch berücksichtigt werden. Ziel ist es, interaktive Systeme so zu gestalten, dass sie nutzbar, nützlich und für Menschen angemessen sind.
Human-Factors-Kriterien	Human-Factors-Kriterien beziehen menschliche Eigenschaften, Fähigkeiten und Grenzen in die Gestaltung ein. Dazu gehören Wahrnehmung, Aufmerksamkeit, Gedächtnis, mentale Modelle, kognitive Belastung, motorische Fähigkeiten und situative Einschränkungen.
Hypothese	Eine Hypothese ist eine überprüfbare Annahme über Nutzerinnen und Nutzer, Aufgaben, Probleme, Ursachen oder Lösungen. AI-generierte Usability-Vorschläge sind häufig Hypothesen und müssen durch fachliche Prüfung, Evaluation, Domänenwissen, Nutzungsdaten oder Usability-Tests geprüft werden.
Informationsarchitektur	Informationsarchitektur beschreibt, wie Inhalte, Funktionen und Objekte strukturiert, benannt und auffindbar gemacht werden. Sie beeinflusst, ob Nutzerinnen und Nutzer verstehen, wo sie sich befinden und wo sie relevante Informationen oder Funktionen erwarten.
Interaction Design / Interaktionsdesign	Interaction Design bezeichnet die Gestaltung der Interaktion zwischen Mensch und System. Dazu gehören Abläufe, Systemzustände, Controls, Systemfeedback, Fehlerbehandlung, Steuerbarkeit und erwartbares Verhalten.
Kognitive Last	Kognitive Last bezeichnet den geistigen Aufwand, den Nutzerinnen und Nutzer benötigen, um Informationen zu verstehen, Entscheidungen zu treffen und Aufgaben auszuführen. Gute UX reduziert unnötige kognitive

Begriff	Erklärung
	Last durch klare Struktur, verständliche Sprache, erkennbare Prioritäten und konsistente Abläufe.
Komponente / UI-Komponente	Eine UI-Komponente ist ein wiederverwendbarer Baustein einer Benutzeroberfläche, etwa Button, Formularfeld, Dialog, Tabelle oder Fehlermeldung. Sie umfasst nicht nur Darstellung, sondern auch Verhalten, Zustände, Accessibility-Eigenschaften und technische Schnittstellen.
Korrelation	Korrelation bedeutet, dass zwei Beobachtungen gemeinsam auftreten, etwa hohe Abbrüche auf mobilen Geräten. Eine Korrelation kann ein wichtiger Hinweis sein, beweist aber nicht automatisch eine Ursache.
KPI	KPI steht für Key Performance Indicator und bezeichnet eine Kennzahl zur Bewertung wichtiger Produkt-, Geschäfts- oder Qualitätsziele. UX-Metriken können mit KPIs verbunden sein, sind aber nicht automatisch dasselbe.
Label	Ein Label bezeichnet die Beschriftung eines Eingabefelds oder Controls. Labels sollten verständlich, fachlich korrekt und programmatisch mit dem jeweiligen Element verbunden sein. Platzhaltertexte ersetzen keine Labels.
Lesbarkeit	Lesbarkeit beschreibt, wie gut Texte visuell erfasst und gelesen werden können. Sie hängt unter anderem von Schriftgröße, Kontrast, Zeilenlänge, Zeilenabstand, Schriftgewicht und Informationsdichte ab.
Loading State	Ein Loading State ist ein Systemzustand, der anzeigt, dass Daten geladen oder eine Aktion verarbeitet wird. Er hilft Nutzerinnen und Nutzern zu verstehen, dass das System arbeitet und noch keine endgültige Antwort vorliegt.
Low-Fidelity-Prototyp	Ein Low-Fidelity-Prototyp hat einen geringen Detailgrad, etwa eine Skizze, ein einfacher Wireframe oder ein grober Klickablauf. Er eignet sich für frühe Diskussionen, Strukturfragen und Varianten.
Menschliche Kontrolle	Menschliche Kontrolle bedeutet, dass AI-Ergebnisse durch verantwortliche Personen bewusst geprüft, bewertet und freigegeben werden. AI kann unterstützen, aber fachliche Verantwortung nicht übernehmen.
Menschliche Prüfung	Menschliche Prüfung bezeichnet die fachliche, methodische und kontextbezogene Kontrolle von AI-Ausgaben, Entwürfen, Analysen oder

Begriff	Erklärung
	Code durch verantwortliche Personen. Sie ist notwendig, weil AI-Ergebnisse falsch, unvollständig oder kontextblind sein können.
Mentales Modell	Ein mentales Modell ist die Vorstellung, die Nutzerinnen und Nutzer davon haben, wie ein System, Prozess oder fachlicher Gegenstand funktioniert oder funktionieren sollte. Wenn ein System gegen dieses mentale Modell arbeitet, entstehen häufig Fehler, Unsicherheit oder Frustration.
Minimal Viable Design System	Ein Minimal Viable Design System ist eine bewusst kleine, pflegbare Anfangsversion eines Designsystems. Es enthält zentrale Elemente wie Farben, Typografie, Spacing, Buttons, Formularfelder, Fehlermeldungen, Systemzustände und grundlegende Accessibility-Regeln.
Mockup	Ein Mockup ist eine visuell ausgearbeitete Darstellung einer Oberfläche. Es zeigt Layout, Farben, Typografie und UI-Elemente, bildet aber nicht zwingend Interaktion oder Systemverhalten vollständig ab.
Monitoring	Monitoring bezeichnet die systematische Beobachtung von Feedback, Supportdaten, Nutzungsdaten, technischen Signalen und UX-Metriken nach dem Release. Es dient dazu, Hinweise auf Probleme, Muster, UX-Debt und Verbesserungsmöglichkeiten zu erkennen.
Muster	Ein Muster ist eine wiederkehrende Beobachtung, etwa häufige Abbrüche an derselben Stelle oder wiederkehrende Supporttickets zu einer Funktion. Muster lenken Aufmerksamkeit, erklären aber noch nicht automatisch Ursachen.
Mustererkennung	Mustererkennung bezeichnet die Fähigkeit von AI-Systemen, bekannte Muster in Texten, Interfaces, Daten oder Abläufen zu erkennen und daraus Vorschläge abzuleiten. AI ist besonders hilfreich bei gut bekannten, häufig vorkommenden Mustern.
Nutzer / User	Ein Nutzer ist eine Person, die ein Produkt, System oder eine Dienstleistung tatsächlich verwendet oder verwenden soll. Der Nutzer ist nicht zwingend identisch mit Käufer, Auftraggeber, Product Owner oder Stakeholder.
Nutzeranforderung / User Requirement	Eine Nutzeranforderung ist eine Anforderung, die sich aus Nutzerzielen, Aufgaben und Nutzungskontext ergibt. Sie beschreibt, was Nutzerinnen und Nutzer in einer bestimmten Situation benötigen, um ihr Ziel zu erreichen.

Begriff	Erklärung
Nutzerfeedback	Nutzerfeedback bezeichnet Rückmeldungen von Nutzerinnen und Nutzern zur tatsächlichen Nutzung, etwa Hinweise, Beschwerden, Wünsche, Bewertungen oder Erfahrungsberichte. Nutzerfeedback ist wertvoll, aber nicht automatisch repräsentativ.
Nutzerziel	Ein Nutzerziel ist das Ziel, das Nutzerinnen und Nutzer mit Hilfe eines Systems erreichen möchten. Gute UX unterstützt Nutzerinnen und Nutzer dabei, ihre Ziele wirksam, effizient und verständlich zu erreichen.
Nutzungskontext / Context of Use	Der Nutzungskontext beschreibt die Bedingungen, unter denen ein System genutzt wird. Dazu gehören Nutzerinnen und Nutzer, Ziele, Aufgaben, Geräte, Umgebung, Organisation, Fachdomäne, Risiken und situative Rahmenbedingungen.
Nutzungsdaten	Nutzungsdaten beschreiben, wie ein System tatsächlich verwendet wird, etwa Abbrüche, Nutzungshäufigkeit, Klickpfade, Bearbeitungszeiten oder wiederholte Fehler. Sie zeigen, was passiert, erklären aber nicht automatisch, warum es passiert.
Persona	Eine Persona ist eine typisierte Beschreibung einer Nutzergruppe mit Zielen, Aufgaben, Vorerfahrung und Nutzungskontext. Datenbasierte Personas beruhen auf Interviews, Beobachtungen, Nutzungsdaten oder belastbarem Domänenwissen. Hypothetische, auch AI-generierte Personas machen Annahmen explizit, sind aber keine Research-Ergebnisse und müssen als Hypothesen gekennzeichnet werden.
Priorität	Priorität beschreibt, was als Nächstes bearbeitet werden soll. Sie ergibt sich nicht allein aus Häufigkeit, sondern aus Schweregrad, Nutzerwirkung, Risiko, Business-Relevanz, Aufwand, Accessibility-Relevanz und strategischer Bedeutung.
Product Owner	Der Product Owner ist eine Rolle in Scrum und in vielen agilen Produktteams. Im Usability-Kontext ist diese Rolle wichtig, weil Nutzerziele, Priorisierung, Akzeptanzkriterien und UX-Debt stark durch Produktentscheidungen beeinflusst werden.
Produktentscheidung	Eine Produktentscheidung ist eine verantwortete Entscheidung über Richtung, Priorität, Umsetzung oder Änderung eines Produkts. Sie sollte auf geeigneter Evidenz, Nutzerwirkung, Business-Relevanz, Risiken und technischer Machbarkeit beruhen.

Begriff	Erklärung
Prompt	Ein Prompt ist eine Eingabe oder Aufgabenstellung an ein AI-System. Im Usability-Kontext kann ein Prompt zum Beispiel eine Oberfläche beschreiben, eine User Story prüfen lassen oder Verbesserungsvorschläge anfordern.
Prototyp	Ein Prototyp ist eine vorläufige Darstellung oder Umsetzung einer Lösung. Er dient dazu, Ideen sichtbar, diskutierbar, prüfbar oder testbar zu machen. Prototypen sind keine fertigen Produkte und keine automatische Validierung.
Radio Button	Ein Radio Button ist ein UI-Control für eine Auswahl aus mehreren gegenseitig ausschließenden Optionen. Er sollte nicht für Mehrfachauswahl verwendet werden.
Release	Ein Release bezeichnet die Bereitstellung eines Produkts, Inkrements oder Features für tatsächliche Nutzung. Nach dem Release können Feedback, Nutzungsdaten und Metriken Hinweise auf reale UX-Qualität liefern.
Remote Usability Test	Ein Remote Usability Test ist ein Usability-Test über Distanz. Er kann moderiert oder unmoderiert durchgeführt werden und ist nützlich, wenn Nutzerinnen und Nutzer geografisch verteilt sind oder im eigenen Nutzungskontext getestet werden sollen.
Retention / Wiederkehrrate	Retention beschreibt, ob Nutzerinnen und Nutzer nach einer ersten Nutzung wiederkehren oder ein Produkt beziehungsweise eine Funktion dauerhaft weiterverwenden. Sie ist eine mögliche UX- oder Produktmetrik.
Review-Format	Ein Review-Format ist eine strukturierte Besprechung oder Prüfung eines Arbeitsergebnisses, etwa im Team, mit Stakeholdern oder Fachpersonen. Review-Formate können Usability-relevante Fragen sichtbar machen, sind aber nicht automatisch Evaluationsmethoden und ersetzen keine Validierung mit realen Nutzerinnen und Nutzern.
Robustheit	Robustheit bedeutet, dass digitale Inhalte und Interaktionen zuverlässig mit unterschiedlichen Geräten, Browsern und assistiven Technologien funktionieren. Robustheit ist besonders wichtig für langfristig stabile und zugängliche Benutzeroberflächen.
Scrum	Scrum ist ein verbreiteter agiler Prozessrahmen für Produkt- und Softwareentwicklung. Im Syllabus wird Scrum beschreibend verwendet,

Begriff	Erklärung
	um typische agile Rollen, Artefakte und Ereignisse einzuordnen. Der Syllabus ist jedoch nicht auf Scrum beschränkt.
Security Review	Ein Security Review ist eine Prüfung sicherheitsrelevanter Aspekte einer Lösung oder Implementierung. Bei AI-generiertem Code ist es wichtig zu prüfen, ob Sicherheitsanforderungen, Berechtigungen, Datenverarbeitung und Eingabevalidierung korrekt umgesetzt sind.
Semantisches HTML	Semantisches HTML bedeutet, passende HTML-Elemente entsprechend ihrer Bedeutung und Funktion zu verwenden, etwa Button für Aktionen, Link für Navigation, Überschriften für Überschriften und Labels für Formularfelder. Es ist eine Grundlage für Accessibility und robuste UI-Implementierung.
Severity / Schweregrad	Severity oder Schweregrad beschreibt die Bedeutung eines Usability-Problems. Sie kann sich aus Nutzerwirkung, Häufigkeit, Risiko, Aufgabenrelevanz, Business-Relevanz und Behebbarkeit ergeben.
Sprint	Ein Sprint ist ein zeitlich begrenzter Arbeitszyklus in Scrum. Im Syllabus wird der Begriff verwendet, wenn konkrete Scrum-nahe Planungs-, Umsetzungs- oder Review-Situationen beschrieben werden. Allgemeiner spricht der Syllabus von agilen Entwicklungsprozessen, agiler Planung oder agiler Umsetzung.
Sprint Review	Das Sprint Review ist ein Scrum-Ereignis, in dem Arbeitsergebnisse gezeigt und mit Stakeholdern besprochen werden. Im Syllabus wird es beschreibend als Beispiel für ein agiles Review-Format verwendet, in dem auch Usability-relevante Fragen betrachtet werden können.
State / Systemzustand	Ein State ist ein konkreter Zustand eines Interfaces, etwa Standardzustand, Ladezustand, leerer Zustand, Fehlerzustand, Warnzustand, Erfolgszustand oder deaktivierter Zustand. Fehlende oder unklare States sind eine häufige Ursache schlechter UX.
Supportdaten	Supportdaten entstehen aus Supporttickets, Helpdesk-Anfragen, Chatprotokollen, internen Rückfragen oder wiederkehrenden Beschwerden. Sie können Hinweise auf Usability-Probleme geben, zeigen aber häufig zunächst Symptome und nicht automatisch Ursachen.
Synthetischer Nutzer	Ein synthetischer Nutzer ist ein durch AI erzeugtes Nutzerprofil oder eine simulierte Nutzerreaktion. Synthetische Nutzer können Perspektiven und

Begriff	Erklärung
	Hypothesen erzeugen, ersetzen aber keine realen Nutzerinnen und Nutzer, keine Research-Daten und keine Usability-Tests.
Systemfeedback	Systemfeedback bezeichnet Rückmeldungen des Systems an Nutzerinnen und Nutzer während der Interaktion, etwa Ladezustände, Fehlermeldungen, Erfolgsmeldungen, Bestätigungen oder Hinweise nach einer Aktion. Es ist von Nutzerfeedback oder Kundenfeedback zu unterscheiden.
Tastaturbedienbarkeit	Tastaturbedienbarkeit bedeutet, dass interaktive Elemente ohne Maus erreichbar und bedienbar sind. Sie ist eine grundlegende Voraussetzung für Accessibility und auch für effiziente Bedienung relevant.
Testaufgabe	Eine Testaufgabe beschreibt in einem Usability-Test ein Ziel oder eine Situation aus Nutzersicht, ohne den Lösungsweg vorzugeben. Sie sollte aus einer Untersuchungsfrage abgeleitet sein und einen beobachtbaren Erfolg ermöglichen.
Testfrage / Untersuchungsfrage	Eine Testfrage oder Untersuchungsfrage beschreibt, was durch eine Evaluation oder einen Usability-Test herausgefunden werden soll. Sie bestimmt Testpersonen, Testgegenstand, Testaufgaben und Beobachtungskriterien.
Testgegenstand	Der Testgegenstand ist das Artefakt oder System, das in einer Evaluation oder einem Usability-Test geprüft wird, etwa Wireframe, Prototyp, Staging-Version, bestehende Anwendung oder einzelner Ablauf.
Testperson	Eine Testperson ist eine Person, die in einem Usability-Test Aufgaben bearbeitet. Sie sollte möglichst gut zur relevanten Nutzergruppe oder zum zu prüfenden Nutzungskontext passen.
Think-Aloud / Lautes Denken	Think-Aloud ist eine unterstützende Methode, bei der Testpersonen während der Bearbeitung einer Aufgabe aussprechen, was sie denken, suchen, erwarten oder nicht verstehen. Die Methode kann mentale Modelle sichtbar machen, aber auch Verhalten beeinflussen und kognitive Belastung erhöhen.
Toggle	Ein Toggle ist ein UI-Control zum Ein- oder Ausschalten eines Zustands. Nutzerinnen und Nutzer erwarten häufig, dass ein Toggle unmittelbar wirkt oder klar zeigt, wann eine Änderung übernommen wird.
Traceability	Traceability bezeichnet die Nachvollziehbarkeit von Anforderungen, Entscheidungen, Implementierung, Tests und Ergebnissen. Im Usability-

Begriff	Erklärung
	Kontext hilft sie zu erkennen, welches Usability-Requirement durch welche Umsetzung und welche Prüfung abgedeckt wird.
Transparenz des AI-Einsatzes	Transparenz des AI-Einsatzes bedeutet, dass nachvollziehbar bleibt, wo AI eingesetzt wurde, mit welchem Ziel, welche Ergebnisse übernommen wurden und welche menschliche Prüfung erfolgt ist. Transparenz verhindert, dass AI-generierte Inhalte später fälschlich als gesicherte fachliche Grundlage behandelt werden.
Tree Testing	Tree Testing ist ein Verfahren zur Prüfung von Informationsarchitektur. Testpersonen erhalten Aufgaben und suchen passende Informationen oder Funktionen in einer reduzierten, textbasierten Navigationsstruktur.
UI / User Interface	Das User Interface ist die konkrete Benutzerschnittstelle eines Systems. Dazu gehören Screens, Formulare, Navigation, Controls, Texte, Interaktionselemente und sichtbare Systemzustände.
UI-Design	UI-Design bezeichnet die Gestaltung der konkreten Benutzerschnittstelle. Dazu gehören Layout, visuelle Hierarchie, Typografie, Controls, Texte, Zustände und Interaktionsmöglichkeiten.
UI-Pattern	Ein UI-Pattern ist eine bewährte Lösung für ein wiederkehrendes Interaktionsproblem, etwa Breadcrumb, Modal-Dialog, Tab-Navigation, Suchfilter oder Inline-Validierung. Patterns müssen zum Nutzerziel und Nutzungskontext passen.
Unbekannter Kontext	Unbekannter Kontext bezeichnet Informationen über Nutzerinnen und Nutzer, Domäne, Aufgaben, Risiken oder reale Nutzungssituationen, die AI nicht kennt. Bei unbekanntem Kontext sind menschliche Prüfung, Domänenwissen und gegebenenfalls reale Nutzerbeobachtung notwendig.
Ursache	Eine Ursache erklärt, warum ein beobachtetes Problem entsteht. Ursachen sind schwieriger zu bestimmen als Muster oder Korrelationen und benötigen häufig zusätzliche Daten, Tests, Nutzerbeobachtung, technische Analyse oder Domänenwissen.
Usability / Gebrauchstauglichkeit	Usability beschreibt das Ausmaß, in dem bestimmte Nutzerinnen und Nutzer ein Produkt in einem bestimmten Nutzungskontext verwenden können, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen. Usability ist damit keine Geschmacksfrage, sondern eine bewertbare Qualität der Nutzung.

Begriff	Erklärung
Usability-Heuristik	Eine Usability-Heuristik ist ein allgemeines Bewertungsprinzip oder eine Regel, mit der typische Usability-Probleme sichtbar gemacht werden können. Sie dient als Bezugsrahmen für heuristische Evaluation.
Usability-Test	Ein Usability-Test ist eine Evaluationsmethode, bei der reale oder repräsentative Nutzerinnen und Nutzer konkrete Aufgaben mit einem Produkt, Prototyp oder System bearbeiten. Beobachtet werden Verhalten, Verständnisprobleme, Fehler, Zögern, Suchbewegungen und Aufgabenerfolg.
User Experience / UX	User Experience umfasst die Wahrnehmungen und Reaktionen einer Person, die sich aus der Nutzung oder erwarteten Nutzung eines Produkts, Systems oder einer Dienstleistung ergeben. UX umfasst Erfahrungen vor, während und nach der Nutzung und geht damit über reine Usability hinaus.
User Story	Eine User Story ist eine kurze Beschreibung einer Anforderung aus Nutzer- oder Stakeholder-Perspektive. Im Usability-Kontext sollte eine User Story nicht nur Funktionalität beschreiben, sondern auch Nutzerziel, Nutzungskontext und erwartete Nutzung berücksichtigen.
UX-Debt	UX-Debt bezeichnet angesammelte oder bewusst aufgeschobene Schwächen in der Nutzungserfahrung eines Produkts. Dazu gehören etwa unvollständige Systemzustände, inkonsistente Komponenten, unklare Begriffe, schlechte Fehlermeldungen, fehlende Accessibility oder ungelöste Usability-Probleme. UX-Debt kann später zusätzlichen Aufwand, Supportbedarf, Fehlbedienung oder Vertrauensverlust verursachen.
UX-Metrik	Eine UX-Metrik ist eine Messgröße zur Bewertung von Nutzung und Nutzererfahrung. Beispiele sind Aufgabenerfolg, Fehlerrate, Bearbeitungszeit, Zufriedenheit, Abbruchrate, Wiederkehrrate oder Supportaufkommen. UX-Metriken zeigen Hinweise, aber nicht automatisch Ursachen.
UX-Qualität	UX-Qualität bezeichnet die Qualität der Nutzung und Nutzungserfahrung eines Systems. Dazu gehören unter anderem Verständlichkeit, Erwartungskonformität, Effizienz, Fehlervermeidung, Zufriedenheit, Accessibility und die Passung zum Nutzungskontext.

Begriff	Erklärung
Usability-Requirement	Ein Usability-Requirement ist eine Anforderung an die Nutzungsqualität eines Systems. Usability-Requirements betreffen zum Beispiel Verständlichkeit, Effizienz, Fehlervermeidung, Systemfeedback, Accessibility, erwartbares Verhalten, kognitive Entlastung oder Systemzustände.
Validierung	Validierung bedeutet, zu prüfen, ob Annahmen, Anforderungen oder Lösungen für die tatsächlichen Nutzerinnen und Nutzer, Aufgaben und Nutzungskontexte tragfähig sind. Im Usability-Kontext kann Validierung durch Usability-Tests, reale Nutzungsdaten, Domänenexpertise, gezielte Nutzerbeobachtung oder andere geeignete Evidenz unterstützt werden. AI-Simulationen und AI-Analysen liefern Hypothesen, aber keine Validierung.
Verifikation	Verifikation bedeutet zu prüfen, ob eine Lösung den spezifizierten Anforderungen entspricht. Sie beantwortet die Frage, ob etwas gemäß Vorgabe umgesetzt wurde. Validierung beantwortet dagegen, ob die Lösung für Nutzerinnen und Nutzer im Nutzungskontext geeignet ist.
Vertikaler Prototyp	Ein vertikaler Prototyp arbeitet einen ausgewählten Funktionsbereich oder Ablauf tiefer aus. Er eignet sich, wenn ein konkreter Ablauf, eine Funktion oder ein kritischer Interaktionspfad genauer geprüft werden soll.
Verständlichkeit	Verständlichkeit bedeutet, dass Nutzerinnen und Nutzer Inhalte, Begriffe, Zustände, Fehlermeldungen und nächste Schritte nachvollziehen können. Verständlichkeit betrifft Sprache, Struktur, Systemfeedback, Navigation und Systemverhalten.
Wahrnehmbarkeit	Wahrnehmbarkeit bedeutet, dass Informationen und Bedienelemente für Nutzerinnen und Nutzer erkennbar sein müssen. Dazu gehören etwa ausreichender Kontrast, klare Struktur, Textalternativen und Informationen, die nicht ausschließlich über Farbe vermittelt werden.
Wahrnehmung	Wahrnehmung ist der Prozess, mit dem Menschen Informationen aus ihrer Umgebung aufnehmen und verarbeiten. Wahrnehmung ist selektiv, begrenzt und kontextabhängig. Für UX bedeutet das: Sichtbarkeit allein reicht nicht; Informationen müssen im richtigen Moment verständlich wahrnehmbar sein.

Begriff	Erklärung
Wireframe	Ein Wireframe ist eine schematische Darstellung von Struktur, Inhalt, Navigation und grundlegender Funktion einer Oberfläche. Visuelle Details stehen dabei nicht im Vordergrund.
Zufriedenheit	Zufriedenheit beschreibt, wie Nutzerinnen und Nutzer die Verwendung eines Systems subjektiv erleben. Sie umfasst unter anderem Vertrauen, Frustration, Sicherheit, Angemessenheit und das Gefühl, eine Aufgabe gut bewältigen zu können.