Fakultät Design, Medien und Information

Department Medientechnik

Bachelorarbeit

Konzeption und Entwicklung einer gamifizierten Lernhilfe-App für neurodivergente Menschen

Vorgelegt von:

Samantha Ofosu-Asiamah

Matrikelnummer: [redacted]

Erstprüfer:

Prof. Dr. Larissa Putzar

Zweitprüfer:

Simon Dewert

Abgabedatum:

22. September 2025

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Finkenau 35
20081 Hamburg

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit der Entwicklung einer gamifizierten Lernhilfe-App für neurodivergente Menschen. Der Anstoß ergab sich aus der Beobachtung, dass Kinder mit ADHS oder Autismus-Spektrum-Störungen in herkömmlichen Bildungsumgebungen oft erhebliche Schwierigkeiten haben. Diese Problematik wird durch mangelnde Anpassung der Lernstrukturen sowie das Fehlen geeigneter digitaler Tools zusätzlich verschärft. Der Fokus liegt darauf, eine fundierte und dennoch praxistaugliche mobile Anwendung zu entwickeln. Die Gamification-Elemente sollen strukturiertes und motivierendes Lernen ermöglichen. Die theoretische Basis bildeten Forschungserkenntnisse zur Neurodiversität, bewährte UX/UI-Designprinzipien sowie gängige Gamification-Konzepte. Das methodische Vorgehen beinhaltet zunächst eine fundierte theoretische Grundlage über Neurodiversität, gefolgt von einer detaillierten Anforderungsanalyse mit spezifischen Personas für die Zielgruppe. Anschließend entsteht die Konzeption einer serviceorientierten Systemarchitektur unter Verwendung des Flutter-Frameworks mit Supabase-Backend. Der resultierende Prototyp verfügt über vollständige Aufgabenverwaltung, anpassbare Timer-Funktionen mit Pomodoro-Technik, ein XP-basiertes Belohnungssystem und Stimmungserfassung. Dabei wird besonders auf neurodivergenzspezifische Designaspekte wie sensorische Adaptierbarkeit und kognitive Entlastung geachtet.

Abstract

This bachelor thesis deals with the development of a gamified learning aid app for neurodivergent people. The impetus for this came from the observation that children with ADHD or autism spectrum disorders often have considerable difficulties in conventional educational environments. This problem is exacerbated by a lack of adaptation of learning structures and the absence of suitable digital tools. The focus is on developing a well-founded yet practical mobile application. The gamification elements are intended to enable structured and motivating learning. The theoretical basis was formed by research findings on neurodiversity, proven UX/UI design principles, and common gamification concepts. The methodological approach first involves establishing a sound theoretical basis on neurodiversity, followed by a detailed requirements analysis with specific personas for the target group. This is followed by the design of a service-oriented system architecture using the Flutter framework with Supabase backend. The resulting prototype features complete task management, customizable timer functions with Pomodoro technique, an XP-based reward system, and mood tracking. Particular attention is paid to neurodiversity-specific design aspects such as sensory adaptability and cognitive relief.

Inhaltsverzeichnis

Abkürzungsverzeichnis . . . . . **IV**

Abbildungsverzeichnis . . . . . **V**

Tabellenverzeichnis . . . . . **VI**

Listingverzeichnis . . . . . **VII**

1. Einleitung **1**

1.1. Problemstellung und Relevanz 1

1.2. Zielsetzung der Arbeit 2

2. Theoretische Grundlagen **2**

2.1. Neurodiversität und digitale Inklusion 3

2.1.1. Konzept und Definition der Neurodiversität 3

2.1.2. ADHS und Autismus-Spektrum: Kognitive Profile 5

2.2. UX/UI-Design 7

2.2.1. Barrierefreiheit vs. universelles Design 8

2.2.2. Designprinzipien und Gestaltungsaspekte für neurodivergente Menschen 9

2.3. Gamification in Lernanwendungen 11

2.3.1. Gamification-Mechanismen und ihre Wirkung 12

2.3.2. Gamification-Anpassungen für neurodivergente Lernende 14

2.4. Mobile Anwendungen für neurodivergente Nutzer 16

3. Anforderungsanalyse **18**

3.1. Zielgruppenanalyse 18

3.1.1. Personas 18

3.1.2. Nutzerkontext und Szenarien 20

3.2. Funktionale Anforderungen 23

3.2.1. FA01 - Registrierung 23

3.2.2. FA02 - Anmeldung 24

3.2.3. FA03 - Aufgabenverwaltung 25

3.2.4. FA04 - Timer 26

3.2.5. FA05 - Fortschrittsanzeige 27

3.2.6. FA06 - Belohnungssystem . . . 28
3.2.7. FA07 - Stimmungstracker . . . 29
3.2.8. FA08 - Einstellungen . . . 30
3.3. Nicht-Funktionale Anforderungen . . . 30
3.3.1. NF01 - Cross-Plattform Kompatibilität . . . 30
3.3.2. NF02 - Performance und Responsivität . . . 31
3.3.3. NF03 - Offline-Funktionalität . . . 31

4. Konzeption . . . 32
4.1. Architektur . . . 32
4.1.1. Architekturmuster . . . 32
4.1.2. Datenmodellierung & ER-Diagramm . . . 45
4.2. UML-Diagramme . . . 48
4.2.1. Aktivitätsdiagramm . . . 48
4.2.2. Sequenzdiagramm . . . 50
4.3. UI/UX-Design . . . 54
4.3.1. Farbkonzept . . . 54
4.3.2. Navigation und Informationsarchitektur . . . 56
4.3.3. Neurodivergenz-spezifische Gestaltungselemente . . . 57
4.4. Gamification–Konzept . . . 58
4.4.1. XP-System und Level-Progression . . . 59
4.4.2. Belohnungsmechanismen . . . 60

5. Implementierung . . . 61
5.1. Technologie-Stack . . . 62
5.2. Screen-Implementierung . . . 63
5.2.1. Onboarding . . . 64
5.2.2. Login/Registrierung . . . 65
5.2.3. Home . . . 66
5.2.4. Aufgaben . . . 67
5.2.5. Aufgaben erstellen . . . 68
5.2.6. Timer . . . 70
5.2.7. Stimmungstracker . . . 71
5.2.8. Einstellungen . . . 72
5.3. State-Management . . . 73

6. Evaluation **74**
6.1. Test-Strategien 75
6.2. Usability-Testing 75
6.2.1. Durchführung 76
6.2.2. Testszenarien für neurodivergente Nutzer 77
6.2.3. Ergebnisse und Erkenntnisse 79
6.3. Erfüllung der Anforderungen 87
6.3.1. Funktionale Anforderungen 87
6.3.2. Nicht-funktionale Anforderungen 96

7. Fazit **98**
7.1. Zusammenfassung 99
7.2. Ausblick 101

Literaturverzeichnis **105**

A. Anhang . . . . . **i**

B. Eigenständigkeitserklärung . . . . . **vi**

Abkürzungsverzeichnis

ADHS	Aufmerksamkeitsdefizit-/Hyperaktivitätsstörung
API	Application Programming Interface
ASS	Autismus-Spektrum-Störung
BLoC	Business Logic Component
CRUD	Create, Read, Update, Delete
EdTech	Educational Technology
EF	Executive Funktionen
ER	Entity-Relationship
FA	Funktionale Anforderungen
FAB	Floating Action Button
IA	Informationsarchitektur
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVP	Minimum Viable Product
NFA	Nicht-Funktionale Anforderungen
REST	Representational State Transfer
RLS	Row-Level Security
SQL	Structured Query Language
SSOT	Single Source of Truth
UI	User Interface
UX	User Experience
WCAG	Web Content Accessibility Guidelines

Abbildungsverzeichnis

1. Realtime-Funktion - Authentifizierungsprozess . . . 39
2. ER-Diagramm . . . 46
3. Aktivitätsdiagramm - Stimmungstracker . . . 49
4. Sequenzdiagramm Achievement-Trigger . . . 51
5. Sequenzdiagramm Perfect Day Achievement . . . 53
6. Modi . . . 56
7. Tab-Bar für die Navigation . . . 57
8. Barrierefreiheitseinstellungen . . . 58
9. Fortschrittsanzeige . . . 60
10. Achievement-Benachrichtigung und Login-Streak . . . 61
11. Onboarding Screen . . . 64
12. Login & Registrierung Screen . . . 65
13. Home Screen . . . 66
14. Aufgaben Screen . . . 67
15. Aufgaben erstellen . . . 68
16. Timer Screen . . . 70
17. Stimmungstracker Screen . . . 71
18. Einstellung Screen . . . 72
19. Verteilung der häufigsten Lernherausforderungen in der Befragung . . . 80
20. Unterstützungsfunktionen bei Überforderung . . . 81
21. Unterstützungsfunktionen bei Überforderung . . . 82
22. iOS- und Android-Installation . . . 82
23. Perfomance-Bewertung . . . 83
24. Bewertung technischer Qualität . . . 84

Tabellenverzeichnis

1. FA01 - Registrierung 24
2. FA02 - Anmeldung 24
3. FA03 - Aufgabenverwaltung 26
4. FA04 - Timer 27
5. FA05 - Fortschrittsanzeige 27
6. FA06 - Belohnungssystem 28
7. FA07 - Stimmungstracker 29
8. FA08 - Einstellungen 30

9. Test Szenario - Aufgabe erstellen 78
10. Test Szenario - Einstellungen anpassen 78
11. Test Szenario - Überfoderung 79

Listings

1. Präsentationsschicht - TaskScreen 33
2. Logikschicht - TaskService 33
3. Update-Methode in der Datenbank 35
4. TaskStatus-Enum 36
5. Client Implementierung 37
6. Server Implementierung 38
7. Service Kapselung 41
8. Enge Kopplung 42
9. Service Contracts 43
10. Autonomie und Wiederverwendbarkeit 44
11. Provider Pattern 73
12. StreamController 74

1 Einleitung

1.1 Problemstellung und Relevanz

Viele Kinder und Jugendliche mit neurodivergenten Eigenschaften stehen im Alltag oft vor Herausforderungen, die häufig ihre schulische, persönliche und soziale Entwicklung beeinträchtigen. Gerade bei Menschen mit Autismus-Spektrum-Störungen (ASS) oder Aufmerksamkeitsdefizit-/Hyperaktivitätsstörungen (ADHS) zeigen sich Schwierigkeiten. Sei es im Umgang mit sozialen Situationen, bei der Konzentration oder bei der Selbstregulation. Diese Denk- und Verhaltensweisen führen dann häufig zu Problemen wie schlechten Schulnoten oder geringem Selbstvertrauen. Dabei sind das keine einfachen Schwächen, sondern sie beruhen auf Unterschieden im Gehirn, die individuell berücksichtigt werden sollten (Roddy & O'Neill, 2019). Dennoch schaffen es Bildungssysteme oft nicht, diesen Besonderheiten gerecht zu werden. Die Lernumgebungen sind meistens auf einheitliche Leistungsmuster und Verhaltensmuster ausgerichtet, was neurodivergente Schüler automatisch benachteiligt. Zusätzlich gibt es einen Mangel an geschultem pädagogischem Personal, was die Situation erschwert. Viele therapeutische Hilfsmittel sind schwer zu erhalten, nicht ausreichend verfügbar oder setzen hohe Vorbereitungen voraus (Purnama et al., 2021). Auch digitale Anwendungen, die unterstützend wirken könnten, sind oft nicht ausreichend gestaltet oder entsprechen nicht den konkreten Bedürfnissen von neurodivergenten Personen. Neben den strukturellen Herausforderungen sind auch hohe finanzielle Kosten zu berücksichtigen. Für Familien, die auf Betreuung angewiesen sind, können hohe Zusatzkosten entstehen, sei es durch private Therapeuten, Zeitverlust im Beruf oder zusätzliche Pflegeleistungen (Roddy & O'Neill, 2019). Obwohl es zwar staatliche Hilfen gibt, reichen diese in vielen Fällen nicht aus. Besonders Familien mit geringem Bildungshintergrund oder knappen finanziellen Mitteln haben kaum Möglichkeiten, private Absicherungen vorzunehmen. Die Folge ist ein erheblicher Druck auf das Umfeld der Kinder, während gleichzeitig die notwendigen Förderungen oft ausbleiben, die die Kinder benötigen.

Darüber hinaus wirken die mangelnden Lernangebote negativ auf die Motivation der Kinder (Kakoura et al., 2024). Zudem zeigt sich, dass digitale Lernangebote längst nicht allen gleichermaßen zur Verfügung gestellt werden. Zwar sind die Angebote vielerorts eingeführt, doch gerade in ländlichen Regionen oder in einkommensschwachen Familien fehlen häufig der Zugang zu leistungsfähigen Geräten oder stabiles Internet. Das erschwert die digitale Teilhabe zusätzlich. Dabei bieten mobile Apps und assistive Technologien viele Möglichkeiten. Sie können flexibel auf die Lernbedürfnisse jedes Einzelnen eingehen, Unterstützung in den Alltag einbinden und digitale Bildungsräume insgesamt inklusiver gestalten (Kakoura et al., 2024). Der Einsatz spielerischer Elemente hat sich als sehr motivierend erwiesen. Rück-

meldungen, kleine Belohnungssysteme oder interaktive Inhalte stärken nicht nur die Ausdauer, sondern unterstützen auch Lernprozesse auf lange Sicht. Ziel ist es nicht, das Lernen in ein Spiel zu verwandeln, sondern Erkenntnisse aus der Motivationsforschung gezielt zu nutzen. Studien deuten zudem darauf hin, dass gerade klare Strukturen und visuelle Reize für Personen mit ASS oder ADHS besonders hilfreich sein können. In vielen Fällen entlasten digitale Lernhilfen Eltern oder Betreuungspersonen, indem sie pädagogische Inhalte in leicht zugänglicher Form bereitstellen (Purnama et al., 2021). Der konkrete Anstoß für die Entwicklung dieser Arbeit ergab sich schließlich aus einem Gespräch mit einer Bekannten, die im sozialen Bereich tätig ist. Dabei wurde deutlich, wie schwierig es ist, geeignete digitale Angebote für diese Zielgruppe zu finden. Um diesen Herausforderungen zu begegnen, braucht es also neue und vor allem praxisnahe Ansätze. Eine durchdachte, gamifizierte Lernhilfe-App könnte eine sinnvolle Ergänzung darstellen und helfen, bestehende Lücken zu schließen und den Zugang zur Bildung für alle gerechter zu gestalten.

1.2 Zielsetzung der Arbeit

In dieser Arbeit wird die Entwicklung einer mobilen Lernhilfe-App (MindSpark) behandelt, die sich gezielt an neurodivergente Menschen richtet, vor allem an Menschen mit ADHS oder ASS. Dabei stehen die technischen Anforderungen im Zentrum sowie die kognitiven Besonderheiten und alltagsrelevanten Herausforderungen dieser Zielgruppe, um mehr Transparenz zu schaffen. Die App soll wissenschaftlich fundiert sein und gleichzeitig einen praktischen Nutzen haben. Das Ziel dieser Arbeit ist es, eine funktionale App zu entwickeln und zu konzipieren, die Gamification-Mechanismen einbezieht. Ein wesentlicher Aspekt dabei ist, dass der Ansatz nicht die Schwächen betont, sondern die Stärken. Die App wird als funktionsfähiger Prototyp mit Backend entwickelt, der die wichtigsten Kernfunktionen abdeckt, jedoch nicht alle möglichen Features implementiert. Der Prototyp veranschaulicht, wie eine solche App grundsätzlich funktioniert, und dient als Grundlage für weitere Forschung und technische Fortschritte für vergleichbare Vorhaben. Dabei liegt der Schwerpunkt auf der Entwicklung für Android-Geräte, da diese eine breitere Gerätebasis bieten und die Entwicklungsumgebung zugänglicher ist. Diese Arbeit umfasst keine umfassenden Nutzertests. Die Evaluation erfolgt mit einer begrenzten Anzahl an Personen, sowohl neurodivergenten als auch neurotypischen Teilnehmern. Marktforschung oder kommerzielle Verwaltung sind ebenfalls nicht Teil dieser Arbeit. Der Fokus liegt dabei auf der Gestaltung und den Funktionen im Rahmen eines Bachelorprojekts.

2 Theoretische Grundlagen

2.1 Neurodiversität und digitale Inklusion

Die Entwicklung einer effektiven gamifizierten Lernhilfe für neurodivergente Menschen zeigt schnell, dass es sich nicht nur um technische Aspekte handelt. Das Verständnis ihrer besonderen Denk- und Verarbeitungsweise bildet die Grundlage für die Zugänglichkeit, die Menschen mit unterschiedlichen kognitiven Fähigkeiten miteinschließt. Das Konzept der Neurodiversität orientiert sich an einem wichtigen gesellschaftlichen Rahmen. Dort wird eine Perspektive gelegt, die neurologische Unterschiede nicht als Einschränkung betrachtet, sondern als natürliche Variation würdigt. Diese Betrachtungsweise bestimmt, wie gut individuelle Lernprozesse gestaltet werden können, und diese Vermutungen lassen sich in den neuropsychologischen Studien zeigen (Kofler et al., 2019). Denn genau solche Erkenntnisse ermöglichen die Entwicklung digitaler Lösungen, die sowohl benutzerfreundlich als auch gezielt gestaltet werden. Die eigentliche Schwierigkeit liegt jedoch darin, etwas in der Mitte zu finden, das einerseits die spezifischen kognitiven Profile berücksichtigt und andererseits funktional zugänglich bleiben soll. Das Konzept gliedert sich daher in die Auseinandersetzung mit ADHS und ASS ein. Diese Profile haben einen Einfluss darauf, wie Informationen verarbeitet, strukturiert und erinnert werden. Sie sind daher entscheidend für die Gestaltung individueller Lernprozesse (Kofler et al., 2019). Ihre Entwicklung basiert so auf verlässlichen Daten statt bloßen Annahmen, hat einen Einfluss auf die Benutzerfreundlichkeit und gleichzeitig bleibt die Funktionalität auch bei verschiedenen kognitiven Anforderungen gewährleistet.

In den folgenden Abschnitten wird zunächst das konzeptionelle Verständnis von Neurodiversität dargelegt. Danach werden die spezifischen kognitiven Merkmale behandelt, die bei der Gestaltung von digitalen Lernanwendungen berücksichtigt werden sollten.

2.1.1 Konzept und Definition der Neurodiversität

Der Begriff *Neurodiversität* wird in der allgemeinen Diskussion häufig verwendet, bleibt jedoch für viele Menschen unverständlich. Das Konzept der Neurodiversität stellt einen grundlegenden Perspektivwechsel in neurologischen Unterschieden dar. Anstatt diese als Abweichungen von der Norm zu sehen, wird darauf hingewiesen, dass es sich um eine natürliche menschliche Vielfalt handelt (Baron-Cohen, 2017). In den letzten Jahrzehnten war das medizinische Modell die einzige Art, neurologische Besonderheiten zu betrachten. Jedoch tauchten in den 1990er immer mehr Stimmen auf, die dieses Verständnis ziemlich grundsätzlich in Frage stellten (Chapman, 2021).

Jaarsma und Welin (2012) beschreiben es als ein Konzept, das davon ausgeht, dass Autismus oder ADHS keine Störungen im klassischen Sinne sind, sondern einfach andere Arten, wie das Gehirn funktionieren kann. Hier stoßen viele auf Kritik bei der Verwendung des Begriffs *Störung*. Das Wort legt nahe, dass etwas in der Entwicklung bzw. in der Ordnung *schiefgelaufen* ist (Chapman, 2021). Die Neurodiversitätsbewegung stellt genau das infrage, da dadurch Autismus als *Störung* in der Gesellschaft angesehen wird und damit wird schnell die Idee von Behandlung oder Heilung mitgegeben (Jaarsma Welin, 2012). Diese Perspektive steht im Gegensatz zu der Ansicht, dass das Gehirn auf eine *normale* Art arbeiten soll. Der Begriff ist in diesem Sinne ähnlich wie Konzepte wie Biodiversität oder kulturelle Vielfalt und weist darauf hin, dass es keine normale oder richtige Art gibt, wie ein Gehirn zu funktionieren hat. Die Bewegung vertritt demgegenüber die Meinung, dass es erforderlich sei, dass sich Gehirne in ihrer Struktur unterscheiden. Es gibt strukturelle und neuronale Unterschiede wie vergrößerte oder verkleinerte Gehirnregionen, erhöhte Neuronenzahlen oder andere neuronale Verbindungen. Jedoch tragen diese Unterschiede dazu bei, dass kulturelle Vielfalt in der heutigen Gesellschaft vertretbar ist (Baron-Cohen, 2017).

Neurodiversität lässt sich in zwei zentrale Aspekte unterteilen. Zum einen wird Autismus als eine neurologische Entwicklungsvariante beschrieben, die mit Besonderheiten in der Wahrnehmung, aber auch in anderen Aspekten einhergehen kann. Dadurch werden Menschen mit neurodiversen Eigenschaften nicht allzu sehr von neurotypischen Menschen unterschieden. Und zum anderen geht es um die gesellschaftliche Anerkennung. Für die Gesellschaft ist es daher wichtig, nicht nur die Rechte zu wahren, sondern auch den Wert neurodiverser Zustände anzuerkennen und zu respektieren (Jaarsma & Welin, 2012).

Die Entstehung des Begriffs *Neurodiversität* wurde vor allem durch die Arbeit von Judy Singer bekannt. Judy Singer, eine Soziologin mit Asperger-Syndrom, gilt als Gründerin des Begriffs (Jaarsma Welin, 2012). Erschienen ist der Begriff zum ersten Mal im Jahre 1998 in einem Artikel von Harvey Blume und entwickelte sich in einem Online-Forum heraus (Baron-Cohen, 2017). In den 1990ern sah die Gesellschaft und vor allem die Eltern neurodivergente Menschen als Krankheit an und suchten nach Wegen, Autismus und andere neurologische Besonderheiten zu heilen. Als Reaktion auf die Sichtweise der Elternorganisation entstand eine Online-Community aus autistischen Menschen (Chapman, 2021). Die Bewegung entwickelte sich nicht nur im Internet weiter, sondern es bot neurodivergenten Menschen die Möglichkeit, alltägliche Hürden in der Kommunikation zu umgehen. Das Internet schafft Raum, der frei von vielen Zwängen ist, die in neurotypischen Kommunikationsformen mitwirken, wie etwa ständigem Blickkontakt, schnellen Gesprächswechseln oder Körpersignalen. Daher dient die Nutzung von digitalen Medien als eine Art Lösung für bestimmte Herausforderungen. Neurodivergente Menschen finden nicht nur online Austausch oder Unterstützung, sondern auch Gleichgesinnte, mit denen sie zusammen über

das Konzept der Neurodiversität vernetzen können (Jaarsma Welin, 2012).

Der Ansatz, der im Grunde etwas gemeinsam hat, ist jedoch eine Kritik an der Art, wie bisher über neurologische Unterschiede gedacht wurde. Die Perspektive der Neurodiversität unterscheidet sich nämlich von den traditionellen medizinischen Modellen. Während die Medizin sich hauptsächlich darauf konzentriert, was bei Menschen nicht stimmt und wie man sie am besten heilen kann, sieht Neurodiversität diese Unterschiede als normale, wertvolle Variationen (Chapman, 2021). Es geht also nicht darum, das zu heilen, sondern eher darum, das als etwas zu betrachten, das einfach anders ist.

2.1.2 ADHS und Autismus-Spektrum: Kognitive Profile

Die kognitiven Profile von Menschen mit ADHS oder ASS zeigen nicht nur bestimmte Herausforderungen, sondern bringen ebenfalls ganz eigene Stärken mit sich, die für die Entwicklung digitaler Lernangebote von Bedeutung sind. Jedoch ist es wichtig, zu verstehen, dass die beiden Nutzergruppen nicht einheitlich oder gleich funktionierend zu betrachten sind. Bei ADHS sowie bei Autismus stehen vor allem Schwierigkeiten mit exekutiven Funktionen (EF) im Fokus. Die höheren kognitiven Prozesse, die hauptsächlich von den Frontallappen gesteuert werden, sind von Bedeutung, um mit den täglichen Herausforderungen auszukommen. Dazu gehören Fähigkeiten wie Planung, Arbeitsgedächtnis, kognitive Flexibilität und Impulskontrolle (Faraone et al., 2021).

Bei ADHS spricht man von einer Störung der Selbstregulation, d. h., Menschen mit ADHS haben es schwer, ihr Verhalten bewusst zu steuern. Dieses Merkmal wirkt sich auf mehrere exekutive Funktionen aus (Faraone et al., 2021). Dazu gehören unter anderem nonverbales und verbales Arbeitsgedächtnis, die Fähigkeit, Emotionen, Motivation und Erregung zu steuern, sowie das Zusammensetzen von Informationen. Die stärksten Schwierigkeiten zeigen sich im Bereich Reaktionshemmung, bei der Daueraufmerksamkeit oder beim Arbeitsgedächtnis (Faraone et al., 2021). Jedoch sind exekutive Dysfunktionen nicht bei allen Menschen mit ADHS vorhanden. Eine Untersuchung zeigt, dass bis zu 89 % der Kinder mit ADHS in mindestens einer EF beeinträchtigt sind. Diese Verschiedenartigkeit zeigt, dass personalisierte Lösungen in digitalen Lernanwendungen unbedingt mitbedacht werden müssen (Kofler et al., 2019).

Eine Schwierigkeit für Menschen mit ADHS zeigt sich in Lernsituationen. Im Umgang mit komplexen Aufgaben zeigt sich, dass diese Nutzergruppe oft Probleme aufweist, komplexe Aufgaben in kleinere Schritte zu unterteilen und zu strukturieren. Dies wirkt sich dann auf die Aufmerksamkeitsspanne desjenigen aus (Faraone et al., 2021). Klassische Tests für das Arbeitsgedächtnis können manchmal die Herausforderungen nicht erfassen. Insbesondere ist das der Fall, wenn Wiederholungen ausschließlich zu gering sind oder Aufgaben mit geringer kognitiver Belastung durchgeführt werden, bei denen keine

Anstrengungen erforderlich sind, um sich Inhalte zu merken (Mavrides, 2016). Dies kann dazu führen, dass Betroffene bei wiederholten Verfahren bessere Ergebnisse erzielen können, als es tatsächlich in ihrem Alltag der Fall ist. Das wiederum birgt die Gefahr, dass ihre Lernschwierigkeiten nicht erkannt und dementsprechend übersehen werden. Letztendlich kann genau das zu Bildungsunterleistung führen. Und nicht Intelligenz oder Interesse stellen die Barriere dar, sondern vielmehr die Interaktion zwischen Aufmerksamkeit, Planung und der Fähigkeit, mehrere Informationen gleichzeitig zu verarbeiten. Trotz der Herausforderungen zeigen Studien, dass etwa 10 % der Kinder mit ADHS in den USA keine Probleme in den drei primären exekutiven Funktionen haben (Kofler et al., 2019). Es ist deutlich, dass man ADHS nicht als einheitliches Störungsbild betrachten sollte, da es Untergruppen gibt, bei denen EF weitgehend intakt sind. Außerdem kann kognitives Training das verbale Arbeitsgedächtnis verbessern und gewisse Fähigkeiten stärken, wenn die Ansätze stimmen (Faraone et al., 2021).

Auch Menschen mit ASS haben Schwierigkeiten mit EF. Allerdings zeigen Forschungsergebnisse in diesem Bereich Unstimmigkeiten und es bestehen Zweifel, ob EF-Dysfunktionen tatsächlich als Marker genutzt werden können. Jedoch, wenn es ums Planen geht, sei es nun das Planen, um Probleme zu lösen, oder das Vorausdenken, was Handlungen bewirken können, zeigt sich dort für Menschen mit ASS das Problem. Besonders auffällig ist die mentale Flexibilität, auch *Shifting* genannt. Für Menschen mit ASS zeigen sich oft dort große Hürden, da sie gedanklich sehr lange bei einem Thema bleiben, selbst wenn sich die Situation geändert hat (Faraone et al., 2021). Es fällt ihnen häufig schwer, das eigene Verhalten spontan an neue oder ungewohnte Situationen anzupassen. Die Forschung über das Arbeitsgedächtnis liefert unterschiedliche Ergebnisse. Jedoch scheint der grundlegende Speicherteil des Arbeitsgedächtnisses betroffen zu sein. Besonders bei der Verarbeitung visueller und räumlicher Informationen zeigen sich deutlich Schwierigkeiten, die wesentlich gut dokumentiert sind. Während verbale Einschränkungen bei komplexeren und sprachlichen Aufgaben auftreten können (Schuh et al., 2012). Diese Schwächen im Arbeitsgedächtnis können im schulischen sowie im akademischen Kontext herausfordernd sein.

Eine der zentralen Herausforderungen bei Menschen mit ASS zeigt sich deutlich bei sozialen Interaktionen. In der Kommunikation sind sie oft beeinträchtigt und auch bei der sogenannten *Theory of Mind* sind Menschen mit ASS häufig eingeschränkt. Sie haben oft nicht die Fähigkeit, sich in andere hineinzuversetzen und ihre Gedanken oder Gefühle zu erkennen (Philip et al., 2012). Dennoch haben Menschen mit ASS häufig die Fähigkeit, Reize zu verarbeiten und Wahrnehmungen besser und präziser zu definieren. Sie setzen dabei eine visuelle Verarbeitungsstrategie ein, d. h., sie nehmen Informationen stärker über das Sehen auf und strukturieren so ihr Denken (Philip et al., 2012). Die visuell Betroffenen, vor allem jene ohne kognitive Beeinträchtigung, haben zusätzlich noch überdurchschnittliche Leistungen bei praxis-konstruktiven Aufgaben. Ein Beispiel wäre das Blockdesign, bei dem Formen nachgebaut oder

zusammengesetzt werden müssen. Gerade bei solchen Aufgaben haben Menschen mit ASS ein Talent dafür (Faraone et al., 2021). Durch diese visuellen Fähigkeiten können Lernanwendungen in diesem Bereich ansetzen und bewusst Strategien verwenden, um Inhalte besser zugänglich zu gestalten (Philip et al., 2012). Vor diesem Hintergrund stellt sich die Frage, wie eine App konkret auf diese individuellen Profile eingehen kann. Ein sinnvoller Ansatz für die App-Entwicklung für Menschen mit ASS ist eine personalisierte Einstiegsphase. Ein kurz gestaltetes Screening könnte hilfreich sein, da es ermöglicht, individuelle Stärken und Schwächen zu erkennen. Auf dieser Basis kann die App gezielte Aufgaben bereitstellen, die diese Bereiche trainieren und dabei weder über- noch unterfordern (Kofler et al., 2019).

Darüber hinaus sind klare und direkte Rückmeldungen für beide Nutzergruppen von großer Bedeutung. Die App sollte deutlich zeigen und kommunizieren, ob eine Aufgabe korrekt gelöst wurde, was noch fehlt oder wie der aktuelle Fortschritt aussieht. Dies schafft Sicherheit und Orientierung im Lernprozess (Mavrides, 2016). Außerdem sollten Aufgaben klar strukturiert sein und in kleine, verständliche Schritte unterteilt werden, sodass es die kognitiven Überforderungen reduziert und es Nutzern hilft, die Planungsschwächen haben (Faraone et al., 2021). Für Menschen mit ADHS sollte die App auf typische Herausforderungen im Bereich der EF eingehen. Die App sollte Aufgaben bieten, die Reaktionshemmung, Arbeitsgedächtnis, Konzentration und Planung trainieren. Insbesondere beim Training des Arbeitsgedächtnisses empfiehlt es sich, sogenannte Recall-Aufgaben einzusetzen, d. h. Aufgaben, bei denen aktiv erinnert werden muss. Viele Menschen mit ADHS haben meistens Schwierigkeiten mit Organisation und Struktur im Alltag. Die App sollte dazu beitragen, Ordnung zu schaffen, indem einfache To-do-Listen, Kalenderfunktionen oder strukturierte Lerneinheiten Hilfestellungen bieten. Eine App für neurodivergente Menschen muss nicht alle Anforderungen gleichzeitig erfüllen, jedoch sollte sie flexibel auf individuelle Unterschiede reagieren können sowie klar kommunizieren und dabei motivieren. An dieser Schnittstelle von Kognition, Alltagsunterstützung und Gamification liegt das Potenzial für digitale Inklusion.

2.2 UX/UI-Design

Die Gestaltung von Benutzeroberflächen und digitalen Nutzererfahrungen für neurodivergente Menschen stellt die Prinzipien des Interface-Designs in Frage. Herkömmliche Ansätze in den Bereichen UI (User Interface) und UX (User Experience) basieren auf einem durchschnittlichen Nutzer. In der Praxis zeigt sich jedoch, dass diese Annahmen für neurodivergente Menschen oft nicht zutreffen und sogar hinderlich sein können (Keil & Bleisinger, 2023). Neurodivergente Nutzer verarbeiten Informationen unterschiedlich und haben eigene Erwartungen an digitale Apps. Diese Erwartungen ergeben sich meistens aus den individuellen Lebensrealitäten und den persönlichen Bewältigungsstrategien der Betroffenen im

Alltag. Daher ist es notwendig, das Verständnis von Nutzerzentrierung und digitaler Verfügbarkeit neu zu überdenken. Ein zentraler Aspekt ist, dass die Inklusion nicht durch Anpassungen erreicht werden kann, sie muss von Anfang an Teil des Designs sein (Keil & Bleisinger, 2023). Dies betrifft sowohl die Werte, auf denen das Design basiert, als auch die konkreten Entscheidungen während der Umsetzung. Das Ziel besteht darin, digitale Lösungen zu entwickeln, die weder überfordern noch unterfordern.

In den folgenden Abschnitten wird untersucht, wie sich diese theoretischen Überlegungen in der Praxis umsetzen lassen. Zudem werden Prinzipien vorgestellt, die dazu beitragen können, digitale Anwendungen inklusiver zu gestalten.

2.2.1 Barrierefreiheit vs. universelles Design

Wenn man digitale Anwendungen für neurodivergente Menschen entwickeln möchte, muss man sich grundsätzlich mit den Konzepten von Barrierefreiheit und universellem Design auseinandersetzen. Beide Konzepte verfolgen das Ziel, digitale Produkte und Dienstleistungen für alle Menschen zugänglich zu machen. Dennoch unterscheiden sie sich in ihrer Herangehensweise und ihren Schwerpunkten.

Barrierefreiheit bedeutet, dass Produkte, Services oder digitale Umgebungen von Menschen mit besonderen Bedürfnissen genutzt werden können. Dazu zählen insbesondere Menschen mit Behinderungen sowie ältere Menschen (Probiesch, 2013). Im Kontext vom Internet bedeutet es, dass Menschen mit Einschränkungen in der Lage sind, Inhalte wahrzunehmen, zu verstehen, sich zurechtzufinden und damit interagieren zu können. Darüber hinaus haben ebenso Menschen ohne Einschränkungen einen Vorteil, z. B. wenn sie vorübergehend eingeschränkt sind oder unter schwierigen Nutzerbedingungen leiden (Probiesch, 2013). Heutzutage wird Barrierefreiheit nicht nur als technische Anforderung betrachtet, sondern hat sich auch im Bereich Human-Computer-Interaktion entwickelt, wenn es um Inklusion und gleichberechtigten Zugang geht (Stephanidis et al., 2025).

Universelles Design, auch bekannt als *Design for All* oder *Inclusive Design*, zielt darauf ab, Produkte und Umgebungen so zu gestalten, dass sie von möglichst vielen Menschen ohne Anpassungen genutzt werden können. Dabei spielen das Alter, die Fähigkeiten, die sie haben, oder wie sie diese einsetzen werden, keine Rolle. Universelles Design basiert auf Eigenschaften wie Flexibilität, einfacher Bedienung, Fehlertoleranz und geringem physischem Aufwand (Chen, 2025). Der Fokus liegt hierbei nicht nur auf der Barrierefreiheit, die häufig auf spezielle Einschränkungen reagiert, sondern vielmehr werden alle Nutzer berücksichtigt, auch solche, die vorübergehend eingeschränkt sind.

Barrierefreiheit und universelles Design haben zwar unterschiedliche Ausgangspunkte, verfolgen jedoch

im Grunde ein gemeinsames Ziel. Sie wollen die digitale Teilhabe für alle Menschen ermöglichen. Während Barrierefreiheit nachträgliche Anpassungen existierender Produkte mit Hilfe assistiver Technologien entwickelt, verfolgt das universelle Design einen anderen Weg. Schon während des Designprozesses einer digitalen Anwendung wird Barrierefreiheit von Anfang an mitgedacht (Stephanidis et al., 2025). Veränderungen, die nachträglich gemacht werden, sind nicht nur teuer, sondern widersprechen dem Grundgedanken des inklusiven Designs, der keine Sonderlösungen oder getrennte Versionen zulässt (Probiesch, 2013). Allerdings ist die Vorstellung, dass man ein Produkt so gestalten kann, dass es für alle Menschen gleichermaßen nutzbar ist, in der Praxis kaum umzusetzen (Prinzellner, 2022). Prinzellner et al. (2022) schlagen anstelle des universellen Designs das Konzept des *User Sensitive Inclusive Design* vor. Anstatt sich auf ein bestimmtes Design zu verlassen, das eine kleine repräsentative Teilmenge berücksichtigt, geht das *User Sensitive Inclusive Design* auf die Diversität individueller Fähigkeiten ein. Ziel ist es, dabei auf Unterschiede in der Wahrnehmung, Kognition und Motorik zu reagieren und dadurch inklusivere Lösungen zu entwickeln.

Viele Webseiten und Apps erfüllen nicht grundlegende Standards zur Barrierefreiheit, was sich unter anderem in der steigenden Zahl an Rechtsstreitigkeiten zeigt. Zum einen sind die Richtlinien oft komplex und lassen sich nicht immer einfach auf bestimmte Fälle anwenden (Prinzellner, 2022). Und zum anderen fehlt es oft an der richtigen Ausbildung, den Werkzeugen und der nötigen Aufklärung, vor allem bei Entwicklern. Wirtschaftliche Zwänge sowie der Wunsch nach schnellen Lösungen erschweren die Umsetzung von Barrierefreiheit (Stephanidis et al., 2025). Hinzu kommen spezifische Herausforderungen im Designprozess. Eine der größten Herausforderungen ist, die vielen verschiedenen Fähigkeiten, Bedürfnisse und Anwendungsbereiche angemessen zu berücksichtigen. Demnach ist es oft schwierig, passende Nutzergruppen für Tests zu finden. Außerdem können sich die Anforderungen verschiedener Nutzergruppen widersprechen. Was für eine bestimmte Gruppe hilfreich ist, kann für die andere Gruppe hinderlich sein (Prinzellner, 2022).

2.2.2 Designprinzipien und Gestaltungsaspekte für neurodivergente Menschen

Die Gestaltung digitaler Benutzeroberflächen für neurodivergente Menschen verlangt einen umfassenden und nutzerzentrierten Ansatz. Hierbei müssen Designprinzipien sowie sensorische und kognitive Besonderheiten einbezogen werden. Insbesondere bei Menschen mit ASS und ADHS zeigt sich, dass Designentscheidungen nicht nur technische oder designbezogene Aspekte betreffen, sondern auch direkte Auswirkungen auf Nutzbarkeit, Verständnis und Motivation haben. Ein zentrales Prinzip in der Gestaltung ist das nutzerzentrierte Design. Bei neurodivergenten Menschen spielt dieser Ansatz besonders eine Rolle, denn standardisierte Usability-Tests allein sind nicht ausreichend. Vielmehr ist es notwendig, die

Zielgruppe frühzeitig und aktiv in den Designprozess einzubeziehen (Maaß & Rink, 2018). Studien und Befragungen helfen dabei, die Bedürfnisse, Vorlieben sowie Herausforderungen dieser Zielgruppe besser zu verstehen.

Gerade neurodivergente Menschen sehen sich in gesellschaftlichen Aspekten häufig bevormundenden Strukturen gegenüber, was sich in digitalen Räumen fortsetzen kann, wenn Designentscheidungen über ihre Bedürfnisse hinweg getroffen werden. Die Informationsverarbeitung hängt meistens von der Gestaltung der Benutzeroberfläche ab. Der Verstehensprozess kann dadurch beeinflusst werden, wenn Nutzer ihr Vorwissen, Erwartungen und individuelle Lernstrategien mitbringen (Maaß & Rink, 2018). Bei neurodivergenten Menschen können diese Strukturen sehr unterschiedlich ausgeprägt sein, was zu anderen Verarbeitungsstrategien führen kann. Deshalb sollte die Informationsgestaltung möglichst so gestaltet werden, dass sie für alle Nutzergruppen zugänglich ist.

Kommunikationsbarrieren können sich auf verschiedenen Ebenen zeigen. Zum einen fachlich, sprachlich oder medial. Eine einfache Sprache sowie klare Satzstrukturen können hilfreich wirken. Metaphern oder Abkürzungen sollten vermieden werden und eine handlungsorientierte Vorgehensweise ist vom Vorteil (Maaß & Rink, 2018). Dies gilt insbesondere für Menschen mit ADHS, die leicht ablenkbar werden können. Auch Menschen mit ASS nehmen metaphorische Sprache oft wörtlich. Lange Worte wie *Zusammenarbeit* lassen sich durch Bindestriche strukturieren, um das Lesen zu erleichtern. Neben der Sprache spielt die Gliederung eine wesentliche Rolle. Die Inhalte sollten klar strukturiert, visuell ansprechend und inhaltlich aufeinander abgestimmt sein. Sogenannte *Advance Organizers*, kurze Einführungen vor komplexen Themen, Signalwörter sowie sichtbare Gliederungen durch z. B. Überschriften oder Aufzählungen helfen dabei, die Informationen besser zu verarbeiten (Maaß & Rink, 2018). Zentrale Informationen sollten relativ am Anfang stehen. Dies ermöglicht den Nutzern, einen besseren Überblick zu haben. Komplexe Inhalte sollten durch Visualisierungen, wie z. B. Bilder oder einfachere Beispiele, ergänzt werden (Maaß & Rink, 2018). Obwohl Vereinfachungen nützlich sind, muss man gleichzeitig auf Stigmatisierung achten. Die übermäßige Übersetzung in leichte Sprache könnte als abwertend wahrgenommen werden. Daher sind individualisierbare Elemente wie anpassbare Schriftgröße oder Kontraste sinnvoll.

Viele neurodivergente Menschen reagieren verstärkt empfindlich auf ihre Umwelt. Für Menschen mit ADHS können intensive Farben, Bewegungen oder Geräusche überfordernd wirken, während Menschen mit ASS häufig auf visuelle Struktur und Konsistenz angewiesen sind. Ein einheitliches Layout mit klarer Ordnung und ausreichend Kontrasten unterstützt dabei, das Arbeitsgedächtnis zu entlasten (Maaß & Rink, 2018). Der gezielte Einsatz von Bildern sollte jedoch immer mit Bedacht eingesetzt werden. Abbildungen können helfen, Begriffe zu veranschaulichen oder in einen Kontext zu setzen, aber sie soll-

ten nicht lediglich zur Dekoration dienen (Maaß & Rink, 2018). Die Abbildungen sollten zum Inhalt das Verständnis fördern und zentrale Konzepte visuell übersetzen. Text-Bild-Kombinationen, bei denen Bild und Text unterschiedliche, aber sich ergänzende Informationen liefern, sind besonders effektiv. Ebenso sind alternative Ansätze wichtig. Dabei zählen Dinge wie Vorlesefunktionen, Grafiken sowie Gebärdensprache. Für blinde oder sehbehinderte Nutzer spielen Alternativtexte eine wichtige Rolle. Daher sollten Alternativtexte sachlich den Bildinhalt beschreiben, ohne neue Begriffe einzuführen oder zu überfordern. Diese multimodale Herangehensweise ermöglicht es, dass Nutzer mit unterschiedlichen sensorischen Voraussetzungen Inhalte gleichwertig erfassen können (Maaß & Rink, 2018).

Technischer Zugang allein reicht jedoch nicht aus. Die inhaltliche Verständlichkeit ist entscheidend, insbesondere für neurodivergente Menschen. Selbst bei optimaler Zugänglichkeit kann ein Angebot scheitern, wenn die Inhalte zu komplex oder unklar sind. Die Verständlichkeitsforschung zeigt, dass es kein Verständlichkeitsoptimum gibt (Maaß & Rink, 2018). Besonders bei geringem Vorwissen oder kognitiven Einschränkungen wirkt eine verständliche Gestaltung positiv auf den Lernerfolg aus. Für die Praxis bedeutet es, dass Systeme so gestaltet werden sollen, dass sie sowohl einen einfachen Einstieg bieten als auch Raum für komplexere Vertiefung lassen können. Gestaltung muss über die Sprache hinausdenken. Interaktive Elemente wie Icons, Farbcodes oder einfache Handlungsschritte helfen dabei, Inhalte zugänglich zu machen. Die Gestaltung für neurodivergente Menschen verlangt nach einem ausgeglichenen Design. Das Design muss sowohl strukturelle Klarheit als auch die Möglichkeit für individuelle Anpassungen bieten. Die technischen, sensorischen und kognitiven Aspekte stehen eng miteinander in Beziehung. Sie müssen die Nutzer nicht überfordern, sondern ihnen die Orientierung geben und ihre Motivation fördern. Es ist wichtig, neurodivergente Nutzer so früh wie möglich in die Entwicklung einzubinden, denn nur so lässt sich eine inklusive, nutzerfreundliche Gestaltung umsetzen.

2.3 Gamification in Lernanwendungen

Der Einsatz spielerischer Elemente im Lernen gilt als ein vielversprechender Ansatz, um Motivation und Beteiligung der Lernenden zu fördern. Klassische Unterrichtsmethoden zeigen oft nur begrenzte Wirkung, vor allem dann, wenn die Aufmerksamkeit nachlässt oder Inhalte wenig ansprechend wirken. In diesem Zusammenhang kann Gamification dazu beitragen, Lernprozesse abwechslungsreicher und langfristig wirksamer zu gestalten. Der Erfolg dieses Ansatzes knüpft an die Beobachtung an, dass Menschen von Natur aus gerne spielen und dabei wichtige Fähigkeiten entwickeln (Nicholson, 2015). Dabei geht es nicht nur darum, Inhalte spielerisch zu verpacken, sondern auch die psychologischen Prozesse zu verstehen, die spielerisches Handeln wirksam machen. Gerade für neurodivergente Lernende hat Gamification eine besondere Bedeutung (Hosseini et al., 2022). Ihre Lern- und Denkweisen unterscheiden

sich häufig von traditionellen Ansätzen, wodurch klassische Methoden oft nicht ausreichen. Gamifizierte Ansätze eröffnen hier mehr Flexibilität, da sie individuelle Stärken hervorheben und zugleich bei spezifischen Herausforderungen unterstützen können. Die theoretische Grundlage solcher Methoden liegt vor allem in der Motivations- und Neuropsychologie (Purnama et al., 2021). Eine sorgfältige und durchdachte Gestaltung ist daher notwendig, damit Gamification nicht nur motivierend wirkt, sondern auch pädagogisch sinnvoll bleibt (Purnama et al., 2021).

2.3.1 Gamification-Mechanismen und ihre Wirkung

Gamification bezieht sich im Wesentlichen auf die Integration spieltypischer Elemente in nicht-spielerische Kontexte. Das Ziel besteht dabei nicht darin, ein Spiel zu entwickeln, sondern vielmehr darin, Spielprinzipien anzuwenden, um alltägliche Aufgaben interessanter und motivierender zu gestalten (Hosseini et al., 2022). Gamification kann in verschiedenen Formen auftreten. Die reichen von einfachen Punkte- und Abzeichensystemen bis hin zu komplexen Konzepten. Die komplexen Konzepte zielen darauf ab, emotionale und psychologische Bedürfnisse der Nutzer anzusprechen. Hier spricht man von *Meaningful Gamification* (Nicholsen, 2015). Das sorgt dafür, dass Nutzer eine persönliche Verbindung zu den Tätigkeiten herstellen und somit langfristig motiviert bleiben, anstelle sich nur auf kurzfristige Belohnungen zu konzentrieren.

Die drei Komponenten (Punkte, Abzeichen und Bestenliste) sind die Grundlagen für viele Gamification-Systeme. Daneben gibt es weitere bedeutende Elemente wie Fortschrittsbalken, die den Fortschritt einer Aufgabe anzeigen können (Mazarakis & Bräuer, 2023). Solche visuellen Darstellungen haben einen großen Einfluss auf die Motivation der Nutzer, denn sie verdeutlichen, was bereits erreicht wurde und was noch aussteht. Außerdem können Geschichte oder Erzählungen einer Aktivität eine tiefere Ebene vermitteln als nur die Aufgabe selbst, sofern sie gut umgesetzt sind. Das steigert sowohl die Motivation als auch das emotionale Interesse (Mazarakis & Bräuer, 2023). Feedback ist ein wesentliches Element von gamifizierten Systemen. Auch Avatare spielen eine Rolle. Sie dienen nämlich als visuelle Stellvertreter der Nutzer und helfen ihnen, sich mit der Anwendung zu identifizieren. Team-Elemente fördern soziale Interaktionen und unterstützen die Zusammenarbeit. Die unterschiedlichen Ansätze lassen sich kombinieren, jedoch ist nicht jede Kombination sinnvoll, denn jedes dieser Elemente spricht verschiedene psychologische Bedürfnisse an.

Die Selbstbestimmungstheorie (engl. Self-Determination Theory) ist eine zentrale theoretische Grundlage zur Erklärung der Wirkungsweise von Gamification. Nach Deci und Ryan besagt die Theorie, dass Menschen besonders motivierend sind, wenn drei grundlegende psychologische Bedürfnisse erfüllt sind.

Das Bedürfnis nach Kompetenz, Autonomie und sozialer Verbundenheit (Hosseini et al., 2022). Das Bedürfnis nach Kompetenz beschreibt das Verlangen, effektiv und erfolgreich zu sein bzw. sich danach zu fühlen. Das Bedürfnis nach Autonomie bezieht sich auf den Wunsch, Entscheidungen über das eigene Handeln zu treffen und dieses selbst zu kontrollieren. In Hinblick auf die Konzeption von Gamification-Designs ist es von Bedeutung, Auswahlmöglichkeiten zu bieten und die Anzahl der Lösungswege zu erhöhen bzw. den Handlungsspielraum zu erweitern. Wichtig hierbei ist es, sicherzustellen, dass die Aufgaben und Belohnungen nicht manipulativ wirken, sondern freiwillig und fair empfunden werden. Das Bedürfnis nach sozialer Verbundenheit umfasst die Aspekte des Empfindens von Zugehörigkeit sowie von sozialer Anerkennung und Unterstützung. Ein weiterer Aspekt zur Erklärung der Wirkungsweise von Gamification ist die Art der Motivation. Intrinsische Motivation entsteht aus dem Inneren. Das Handeln erfolgt aus Spaß oder als Herausforderung (Hosseini et al., 2022). In solchen Fällen hat sich Gamification häufig als effektive Methode für langfristige Verhaltensänderungen erwiesen. Im Gegensatz zur intrinsischen Motivation stammt extrinsische Motivation von äußeren Anreizen. Man handelt beispielsweise, um Belohnungen zu erhalten oder Bestrafungen zu vermeiden. Für eine kurzfristige Zeit können externe Belohnungen wirksam sein, jedoch besteht in manchen Fällen die Gefahr, dass extrinsische Motivation intrinsische Motivation untergraben kann (Nicholson, 2015). Eine Kritik an Gamification richtet sich meistens gegen Systeme, die ausschließlich auf extrinsischen Belohnungen basieren, denn dieser Anreiz kann lediglich vorübergehende Effekte erzielen. Effektive Gamification sollte sowohl intrinsische als auch extrinsische Motivationen berücksichtigen und versuchen, extrinsische Anreize so zu gestalten, dass sie sich wie intrinsische anfühlen (Hosseini et al., 2022).

Die Wirkung der Gamification hängt stark vom Kontext sowie von den spezifischen Eigenschaften der Zielgruppe ab. Die in einem Lernkontext erfolgreiche Methode könnte sich in einem Gesundheits- oder Arbeitsumfeld eher als unzureichend erweisen. Nicht jede Person reagiert auf Belohnungen oder Wettbewerbselemente. Einige Nutzer bevorzugen kooperative Ansätze, während andere Nutzer Rankings bevorzugen. Daher ist es wichtig, die Implementierung von Gamification nicht oberflächlich und wahllos einzuführen, sondern sorgfältig an die jeweiligen Nutzerbedürfnisse anzupassen. Ein nutzerzentrierter Designprozess, bei dem systematisch mit den Bedürfnissen, Zielen und der Motivation der Zielgruppe gearbeitet wird, spielt hier eine zentrale Rolle (Mazarakis & Bräuer, 2023). Des Weiteren sollte eine transparente Kommunikation hinsichtlich der Gründe für die Belohnung sowie der Zielsetzung ermöglicht werden. Dies stärkt das Gefühl der Kompetenz des Nutzers und trägt zur Nutzerakzeptanz bei.

2.3.2 Gamification-Anpassungen für neurodivergente Lernende

Die in Abschnitt 2.3.1 beschriebenen allgemeinen Gamification-Mechanismen können effektiv sein, aber neurodivergente Menschen haben spezifische neurologische und motivationale Profile. Diese erfordern besondere Anpassungen innerhalb von Lernumgebungen (Dovis et al., 2012). Ein Einheitsansatz (One-Size-Fits-All) ist häufig nicht effektiv in der Motivationsförderung und kann schädlich wirken, da er individuelle Reaktionen nicht berücksichtigt (Orji et al., 2014). Es zeigt sich, dass die bekannten Gamification-Elemente grundsätzlich funktionieren, aber bei neurodivergenten Lernenden müssen sie häufig anders eingesetzt werden.

Kinder mit ADHS zeigen ausgeprägte Probleme im Bereich des Arbeitsgedächtnisses sowie bei der Impulskontrolle. Dies kann dazu führen, dass sie Aufgaben oder Ziele nur schwer behalten können. Ihre Aufmerksamkeitsspanne ist insbesondere bei monotonen Aufgaben oder in unbeaufsichtigten Situationen stark begrenzt (Prins et al., 2011). Im Vergleich zu neurotypischen Lernenden benötigen sie demnach stärkere und häufigere Anreize, um motiviert zu bleiben und langfristig eine optimale Leistung zu zeigen. Darüber hinaus konnte beobachtet werden, dass die Leistungsfähigkeit über längere Zeiträume hinweg häufig abnahm. Insbesondere, wenn die Aufgaben als wenig anspruchsvoll empfunden wurden oder eine zu geringe Struktur aufwiesen (Prins et al., 2011). Ein zentraler Aspekt ist dabei die Bedeutung von Anerkennung und Belohnung. Kinder mit ADHS zeigen in der Regel eine erhöhte Reaktion auf motivierende Rückmeldungen. Belohnungssysteme wie Geldbeträge sind oft nicht ausreichend. Stattdessen sind interaktive Reize, wie z. B. Computerspiele oder spielerisch gestaltete Lernsysteme, deutlich effizienter, um Motivation, Ausdauer und Leistung langfristig zu fördern (Prins et al., 2011).

Ein weiterer Aspekt ist die adaptive Schwierigkeit. Der Schwierigkeitsgrad einer Aufgabe sollte an die Fähigkeit des Lernenden angepasst werden. Das dient dazu, Frustration zu vermeiden und die Motivation aufrechtzuerhalten. Gamification kann die Fähigkeit, neue Dinge zu lernen, um bis zu 40 % verbessern (Prins et al., 2011). Empirische Studien belegen, dass der Einsatz von mobilen Gamification-Systemen zu besseren Lernleistungen führen kann als traditioneller Unterricht. Die Verbesserungen der Problemlösungsfähigkeit, Kreativität und des kritischen Denkens werden durch die Methode ermöglicht (Peijnenborgh et al., 2016). Die Entwicklung des Spiels *Timo's Adventure* zielte darauf ab, Aufmerksamkeits-, Planungs-, Arbeitsgedächtnis-, Zeitwahrnehmungs- und Motivationsdefizite bei jungen Kindern im Alter von vier bis acht Jahren zu identifizieren. Es konnte eine klare Trennung zwischen Kindern mit ADHS und anderen Gruppen festgestellt werden. Dabei zeigte sich eine Vorliebe für kurzfristige Belohnungen seitens der Kinder mit ADHS. Das positive Feedback der Kinder belegt zudem die Wirksamkeit von solchen Formaten (Peijnenborgh et al., 2016). Selbst eine kurze spielerische Übung zur Stärkung des Arbeitsgedächtnisses zeigt einen positiven Einfluss auf EF sowie auf ADHS-bezogene Verhaltenswei-

sen.

Dahingegen zeigen Menschen mit ASS oft ein starkes Interesse am computergestützten Lernen. Aufgaben sind klar definiert und entsprechend gestaltet, was Ablenkungen reduziert. Häufig treten Schwierigkeiten in der sozialen Kommunikation auf, insbesondere hinsichtlich der sprachlichen Ausdrucksweise, die nur schwer veränderbar ist. Dies führt dazu, dass die Sprecher unbeabsichtigt falsche Emotionen vermitteln. Darüber hinaus fällt es Menschen mit ASS schwer, ihre eigene sprachliche Ausdrucksweise wahrzunehmen (Boyd et al., 2016). Technologische Vermittlungsformen spielen eine große Rolle, da soziale Interaktionen oft herausfordernd sind. Interaktive Roboter wie *KiliRo* finden bei Kindern mit ADHS hohen Zuspruch, denn die Kinder zeigen wenig Angst und interagieren mit dem Roboter (Bharatharaj et al., 2017). Technologien wie *SayWat* wurden entwickelt, um bei der sprachlichen Ausdrucksweise bei Erwachsenen mit ASS unterstützend zu wirken. Sie zielen darauf ab, das Bewusstsein für das eigene Verhalten zu fördern, anstelle direkte Anweisungen zu erteilen. Dadurch wird Nutzern ermöglicht, ihre eigene Verhaltensweise besser wahrzunehmen oder auszuprobieren (Boyd et al., 2016).

Über die grundlegenden Elemente aus Abschnitt 2.3.1 hinaus werden im Bildungsbereich zusätzliche Elemente für neurodivergente Lernende eingesetzt. Erzählungen oder Geschichten können zum Lernprozess beitragen, indem sie tiefere Bedeutung schaffen und Nutzer in die spielerischen Welten entführen. Im Fall von *Timo's Adventure* wurde beispielsweise eine Märchengeschichte verwendet, um die Motivation zu steigern (Peijnenborgh et al., 2016). Die bereits erwähnte adaptive Schwierigkeit passt sich dynamisch an das Können des Lernenden an, was ein optimales Maß an Herausforderungen sowie nachhaltiges Engagement zur Folge hat. Die Anpassung kann systemgesteuert erfolgen oder erfordert vom Lernenden, sich dem Level des Spiels anzupassen. Es besteht ein wesentlicher Unterschied zwischen Personalisierung, die systemgesteuert erfolgt, und Customization, die nutzergesteuert erfolgt (Orji et al., 2014). Die Personalisierung stellt eine wirksame Methode zur Steigerung von Motivation und Engagement dar. Das System passt hier die Inhalte sowie Funktionen individuell an die Bedürfnisse des Lernenden an. Customization ermöglicht dem Lernenden, eigene Inhalte sowie Funktionen nach persönlichen Vorlieben anzupassen (Orji et al., 2014).

Gamification sollte bewusst in einen breiteren Bildungsrahmen integriert werden. Dabei darf die Gamification nicht den eigentlichen Bildungswert übersteigen. Darüber hinaus ist eine umfassende Schulung der Lehrkräfte erforderlich, denn nur so können die Lehrkräfte sowohl die Technik beherrschen als auch flexibel auf Veränderungen reagieren. Damit solche Konzepte wirksam umgesetzt werden können, braucht es jedoch nicht nur pädagogisches Wissen, sondern ebenso entsprechende technische Voraussetzungen. Die Umsetzung erfordert eine starke Infrastruktur sowie eine Integration mit bestehenden Bildungstechnologien. Finanzielle Mittel müssen bereitgestellt werden, sowohl für Investitionen als auch

für den Zeitaufwand für Lehrschulungen (Boyd et al., 2016). Es muss dafür gesorgt werden, dass alle Studierenden Zugang zu den nötigen technologischen Ressourcen haben, sonst besteht die Gefahr, dass die Unterschiede in der Bildung zu groß werden.

2.4 Mobile Anwendungen für neurodivergente Nutzer

Die fortschreitende Digitalisierung eröffnet neue Möglichkeiten, um Menschen mit neurodivergenten Bedürfnissen zu unterstützen. Mobile Apps haben sich als nützliche Begleiter im Alltag von Personen mit ADHS oder anderen neurodivergenten Profilen erwiesen. Aktuelle Forschungsergebnisse zeigen jedoch, dass viele dieser Technologien nicht ausreichend auf die Bedürfnisse der Nutzer zugeschnitten sind. Viele der vorhandenen mobilen Apps gehen von einem einseitigen Ansatz aus. Sie versuchen, Verhaltensweisen zu korrigieren, die von der neurotypischen Norm abweichen (Spiel et al. 2022). Statt neurodivergente Erfahrungen als eigenständige Tatsache anzuerkennen, wollen viele Apps Anpassungen an gesellschaftliche Standards vornehmen. Gerade die Flexibilität digitaler Technologien hat großes Potenzial. Mobilen Geräten ermöglichen es, Informationen auf viele Arten darzustellen, sei es mit Bildern, visuellen Symbolen, Sprachen oder Berührungen (Miesenberger, 2018).

Digitale Apps sind für neurodivergente Menschen in den Bereichen Lernen, Zeitmanagement und Aufgabenstrukturierung besonders relevant. In der Bildung können EdTech-Lösungen (Educational Technology) mit interaktiven Lernspielen oder visuell unterstützten Aufgaben dabei helfen, individuelle Lernwege zu gestalten. Im frühen Schulalter werden solche Technologien kaum eingesetzt, aufgrund von Unsicherheiten bei Lehrkräften im Umgang mit digitalen Hilfen oder weil diese nicht verfügbar sind (Miesenberger, 2018). Auch beim Zeitmanagement können Apps unterstützend wirken. Wie in Abschnitt 2.1.2 genannt, haben Menschen mit ADHS häufig Schwierigkeiten bei der Zeiteinschätzung oder dem rechtzeitigen Beginn einer Aufgabe. Digitale Timer könnten hilfreich sein, sofern sie nutzerzentriert entwickelt werden. Mobile Apps für neurodivergente Menschen lassen sich in verschiedene Kategorien unterteilen. Drei zentrale Bereiche sind dabei besonders relevant. Assistierende Technologien, sprachbezogene Technologien sowie dialogbasierte Systeme. Jede Kategorie ist auf bestimmte Bedürfnisse ausgerichtet, je nachdem, welche Barrieren im Alltag auftreten.

Assistierende Technologien sorgen dafür, dass der Zugang zu digitalen Geräten und Informationen erleichtert wird. Sie helfen Menschen mit Einschränkungen dabei, alltägliche Aufgaben zu bewältigen. Dazu gehören Apps mit einfachen Oberflächen oder Programme, mit denen man Aufgaben strukturieren kann. Für Menschen mit ADHS oder Autismus sind solche Tools besonders hilfreich. Sie reduzieren den Reiz und erleichtern die Navigation sowie das Strukturieren von Routinen. Beispielsweise kön-

nen Symbole anstelle von Text verwendet werden, um Informationen schneller erfassbar zu machen. Auch reduzierte Farbschemata und klare Layouts können helfen, die Nutzer nicht zu überfordern und die Nutzung zu vereinfachen (Miesenberger, 2018). Sprachtechnologien und Sprachverarbeitung hingegen umfassen Anwendungen für die Interaktion mit gesprochener Sprache. Das kann Spracherkennung oder Sprachsynthese sein. Diese Technologien sind besonders nützlich für Menschen mit Leseschwierigkeiten oder Aufmerksamkeitsproblemen. Eine App mit Text-zu-Sprache-Funktion kann Kindern mit Autismus helfen, schriftliche Anweisungen in gesprochene Sprache umzuwandeln. In den letzten Jahren haben interaktive Technologien an Bedeutung gewonnen. Dazu gehören Dialogsysteme wie Avatare oder Chatbots zur Kommunikation. Solche Systeme werden oft in der Therapie benutzt, vor allem bei autistischen Personen, um sozialen Interaktionen zu üben. Bestimmte Apps erlauben den Nutzern, mithilfe eines Avatars Alltagssituationen zu simulieren. Solche Programme bieten kontrollierbare Umgebungen, in denen Nutzer soziale Regeln üben können, ohne dass es zu viel Druck oder Reizüberflutung kommt (Miesenberger, 2018).

Apps für neurodivergente Menschen müssen in technischer als auch in konzeptioneller Hinsicht sensibel gestaltet werden. Dies betrifft nicht nur die Barrierefreiheit, sondern auch Aspekte wie Wahrnehmung, Informationsverarbeitung und emotionale Belastbarkeit. Viele Produkte sind so gestaltet, dass sie den typischen Denkweisen und Verhaltensmustern von Menschen ohne Behinderungen entsprechen. Ein weiteres Problem besteht darin, dass Apps selten auf die tatsächlichen Alltagsrealitäten und Bedürfnisse der Zielgruppen abgestimmt sind. Die betroffenen Nutzer werden nur selten aktiv in den Entwicklungsprozess einbezogen. Obwohl viele Entwickler von einer nutzungszentrierten Gestaltung sprechen, bleibt dies in der Praxis oft theoretisch. Entscheidungen über Funktionen, Inhalte und Design werden in der Regel von Fachexperten, Therapeuten oder sogar Angehörigen getroffen und nicht von denjenigen, die die App tatsächlich nutzen sollen. Des Weiteren wird Neurodivergenz in der technischen Entwicklung oft aus einer defizitorientierten Perspektive betrachtet. Neurodivergentes Denken und Verhalten wird häufig als Störung, Verletzung oder Abweichung betrachtet und genau dies reduziert die Relevanz. Technologien werden nicht entwickelt, um Vielfalt zu unterstützen, sondern um Abweichungen zu korrigieren (Alper et al., 2024).

Trotz der genannten Herausforderungen bieten mobile Apps ein großes Potenzial, um echte Veränderungen im Alltag zu ermöglichen. Dies setzt jedoch voraus, dass die App nicht nur für, sondern gemeinsam mit den Betroffenen entwickelt wird. Anstatt sich ausschließlich auf die Schwächen zu fokussieren, sollte der Blick auf die individuellen Stärken und Potenziale gerichtet werden. Ein theoretischer und praktischer Rahmen für diesen Ansatz ist die *Crip Technoscience*. Dieser Begriff beschreibt eine technologische Praxis, die Menschen mit Behinderung nicht als Objekt von Fürsorge betrachtet, sondern als

aktive Wissensträger mit eigenständigen Sichtweisen (Spiel et al., 2022). Wird dieses Wissen anerkannt, eröffnet sich eine neue Perspektive in der Technologieentwicklung. Der Fokus liegt auf Zugänglichkeit und sozialer Gerechtigkeit. In der Praxis bedeutet es, dass neurodivergente Menschen frühzeitig und kontinuierlich in alle Phasen der Entwicklung eingebunden werden. Ein Beispiel ist das Projekt *Tangiplan*, bei dem Kindern mit ADHS aktiv an der Gestaltung einer interaktiven Planungsapp mitgewirkt haben (Spiel et al., 2022). Diese genannten Beispiele verdeutlichen, dass partizipative Entwicklung zu funktional besseren Lösungen führt. Mobile Apps bieten ein Potenzial, insbesondere in Bezug auf die Förderung von Selbstbestimmung, Flexibilität und Wahlmöglichkeiten. Jedoch muss auf die Stimmen neurodivergenter Nutzer gehört werden. Die Entwicklung digitaler Anwendungen für neurodivergente ist noch relativ am Anfang ihrer Möglichkeiten.

3 Anforderungsanalyse

3.1 Zielgruppenanalyse

Die theoretischen Grundlagen, die in Kapitel 2 vorgestellt wurden, ermöglichen es, die Anforderungen von neurodivergenten Nutzern zu verstehen. Zudem zeigen sie das Potenzial von Gamification in Lernumgebungen. Durch die Analyse einiger Konzepte und wissenschaftlicher Erkenntnisse lassen sich konkrete Anforderungen für die Lernhilfe-App ableiten. Im Mittelpunkt stehen die besonderen Bedürfnisse und Nutzungskontexte der Zielgruppe. Die Anforderungsanalyse beginnt daher mit einer genauen Beschreibung der Zielgruppe, die sich auf Nutzergruppen stützt. Durch den Einsatz von Personas und Szenarien wird ein realistisches Bild der Nutzer erstellt.

3.1.1 Personas

Eine sorgfältige Analyse der kognitiven Profile und individuellen Bedürfnisse ist von Vorteil, wenn es um die Erstellung von Personas für neurodiversitätsorientierte Anwendungen geht. Personas dienen als semi-fiktive Vertreter der Zielgruppe. Um bessere Lösungen und Angebote zu entwickeln, dienen diese Personas dazu, die Bedürfnisse und Wünsche der Zielgruppen besser zu verstehen. Außerdem helfen sie dabei, designorientierte Entscheidungen zu stützen. Im Gegensatz zu Methoden, die nur Zahlen verwenden, ermöglichen Personas einen ganzheitlichen Blick auf die Nutzererfahrung (Salminen et al., 2022). Sie berücksichtigen Verhaltensweisen, Erfahrungen und emotionale Bedürfnisse, die gerade bei neurodivergenten Nutzern von entscheidender Bedeutung sind. Die Methode erlaubt es, die Komplexität

individueller Herausforderungen in nachvollziehbare Nutzungsszenarien zu übersetzen, ohne dabei in übermäßige Verallgemeinerung zu verfallen. Für die Entwicklung barrierefreier Apps bieten Personas mehrere methodische Vorteile. Sie schaffen eine gemeinsame Verständigungsbasis zwischen allen Projektbeteiligten und verhindern, dass Entwicklungsentscheidungen nicht auf vagen Annahmen basieren. Es geht mehr um konkrete Nutzungssituationen und spezifische Bedürfnisse. Dies ist besonders relevant, weil Nutzer, die anders als die Norm sind, oft übersehen werden. Die Personas dienen als ständige Erinnerungen daran, was die Zielgruppe wirklich benötigt (Lundqvist, 2024).

Im Rahmen dieser Arbeit wurden drei zentrale Personas entwickelt, die sowohl neurodivergente als auch neurotypische Nutzer mit Herausforderungen im Umgang mit digitalen Lernhilfen darstellen. Diese Personas basieren auf realitätsnahen Profilen und spiegeln unterschiedliche neurokognitive Eigenschaften wider.

Anna repräsentiert junge Erwachsene mit diagnostizierter ADHS, die sich im akademischen Umfeld bewegen. Als 19-jährige Informatikstudentin im ersten Jahr wohnt sie in einer WG und gehört zur technisch affinen Nutzergruppe, die spezifischen Herausforderungen des ADHS-Spektrums ausgesetzt ist. Annas Hauptschwierigkeiten liegen in der Priorisierung der Aufgaben, einem typischen Symptom von exekutiven Funktionsstörungen bei ADHS. Ebenso wie das ständige Wechselspiel zwischen Fokus und Ablenkbarkeit beeinflussen die Nutzererfahrung maßgeblich. In spezifischen Phasen ist Anna in der Lage, sich mit hoher Konzentration einer Aufgabe zu widmen, während es in anderen Momenten Schwierigkeiten gibt, die Aufmerksamkeit aufrechtzuerhalten. Insbesondere bei Aufgaben, die sie als uninteressant oder wenig lohnend findet. In solchen Situationen tritt bei Anna oft Prokrastination auf. Sie schiebt Aufgaben auf und entwickelt dadurch Schuldgefühle, wobei sie daraus dann noch mehr Aufgaben schiebt. Darüber hinaus kämpft Anna mit der Umsetzung von Aufgaben. Es fällt ihr deutlich schwer, die Aufgabe zu beginnen, und ihr Arbeitsgedächtnis funktioniert nicht optimal, weshalb sie zwischen Arbeitsschritten wichtige Informationen vergisst. Sie zeigt außerdem eine hohe technische Affinität. Auf ihrem Smartphone befinden sich etwa zehn Produktivitäts-Apps, keine jedoch wird davon konsequent genutzt. Sie ist noch auf der Suche nach dem perfekten Tool und zeigt eine hohe Bereitschaft, neue Apps auszuprobieren.

Marcus steht für die Jugendlichen mit ASS, die ein starkes Bedürfnis nach Struktur und Routine aufweisen. Als 16-jähriger Schüler der 11. Klasse nutzt Marcus digitale Tools hauptsächlich zur Bewältigung schulischer Anforderungen. Seine Interessen liegen in Mathematik und Geschichte. Er lebt bei seinen Eltern in einer Kleinstadt und hat seine Diagnose mit 12 Jahren erhalten. Jeden Tag steht er um exakt 07:15 Uhr auf und Abweichungen von dieser Routine stressen ihn erheblich. Die Überforderung durch visuelle Reize ist ein zentrales Thema bei der Gestaltung. Marcus zeigt eine ausgeprägte Lichtempfindlichkeit und reagiert auf grelle Lichter und blinkende Displays mit Kopfschmerzen. Seine auditive Wahrneh-

mung ist so ausgeprägt, dass die Schulklingel für ihn körperlich schmerzhaft ist. Marcus hat Probleme mit Veränderungen, die er nicht vorhersehen kann. Sein Gehirn ist darauf programmiert, Muster und Routinen zu erkennen und sich darauf zu verlassen. Wenn sich sein Stundenplan ändert, löst das bei ihm große Unsicherheiten aus und er wird gestresst und nervös. Marcus kann stundenlang an Mathe-Aufgaben arbeiten, jedoch kann er sich nur auf wenige Dinge gleichzeitig konzentrieren. Marcus zeigt eine mittlere technische Affinität und bevorzugt einfache und konsistente Interfaces. Er verwendet seit Jahren dieselbe App und passt die Einstellungen sofort an seine Bedürfnisse an.

Lisa vertritt neurotypische Individuen mit ausgeprägter digitaler Ablenkbarkeit. Als Realschülerin der 8. Klasse lebt sie bei ihren Eltern in einer Großstadt und steht im ständigen Vergleich mit ihrer älteren, erfolgreichen Schwester. Lisas Hauptprobleme sind Prokrastination durch digitale Ablenkungen, insbesondere Social Media und Gaming. Sie benutzt Instagram und TikTok in Intervallen von fünf bis zehn Minuten und hat für alle Apps Benachrichtigungen aktiviert. Zeitmanagement und die Überschätzung verfügbarer Zeit für Aufgaben spiegeln ihre exekutiven Funktionsdefizite wider. Aufgrund ihrer hohen technischen Affinität und Gewöhnung an gamifizierte Anwendungen bringt Lisa eine optimale Voraussetzung für Produktivitäts-Apps mit.

3.1.2 Nutzerkontext und Szenarien

Um digitale Anwendungen zielgerecht zu entwickeln, ist nicht nur die Beschreibung der Zielgruppe von Bedeutung. Ein Verständnis des Nutzerkontexts spielt ebenfalls eine große Rolle. Es braucht das Verständnis dafür, wie und wann die Anwendung tatsächlich genutzt wird. Gerade bei neurodivergenten Menschen ist dieser Kontext komplex. Faktoren wie Tagesform, kognitive Belastung und Umgebungsreize beeinflussen das Nutzungserlebnis sehr stark. Der Einsatz von Anwendungsszenarien, auch als *Use Case* bezeichnet, bietet Einblicke in konkrete Abläufe und unterstützt die praxisnahe Auswertung von Gestaltungsideen (Floch et al., 2020).

Use Case 1: Aufgabenerstellung

Ein konkretes Beispiel lässt sich anhand der Persona Anna darstellen. Im vorliegenden Fall plant Anna, eine Hausarbeit über SQL-Datenbanken zu erstellen. Nach dem Mittagessen möchte sie sich in den Arbeitsmodus versetzen und öffnet die App. Der Einstieg erfolgt über einen klar gekennzeichneten +-Button, um eine neue Aufgabe zu erstellen. Das Interface ermöglicht es ihr direkt, den Titel *Datenbanken-Hausarbeit* einzutragen. Eine Funktion zur Priorisierung erlaubt es ihr, die Aufgaben visuell als hoch einzustufen. Eine sehr hilfreiche Funktion ist, dass sie große Aufgaben in kleinere Teilaufgaben unter-

teilen kann. Anna fügt *Literaturrecherche* (*2 h*), *Gliederung erstellen* (*30 min*), *Einleitung schreiben* (*1 h*) und *Hauptteil Kapitel 1* (*2 h*) hinzu. Die App berechnet automatisch die Gesamtarbeitszeit. Das Erfolgsszenario dabei ist, dass Anna eine überwältigende Aufgabe in Teilaufgaben unterteilt hat. Die App identifiziert zudem potenzielle Fehlerquellen. Vergisst Anna, die benötigte Zeit für eine Aufgabe oder Teilaufgabe anzugeben, nutzt die App die vorgegebenen Standardwerte.

Use Case 2: Routine-Aufgaben mit sensorischen Anpassungen

Aus der Perspektive von Marcus lässt sich ein weiterer Nutzungskontext ableiten. In diesem Anwendungsszenario öffnet Marcus wie gewohnt jeden Tag um Punkt 15 Uhr die App. Die Benutzeroberfläche der App ist bewusst reizarm gestaltet. Die App verwendet zurückhaltende Farben, verzichtet auf blinkende Elemente und laute Animationen. Für Marcus ist es wichtig, dass die Benutzeroberfläche in Struktur und Farbe konsistent bleibt. Über das Menüelement *Aufgaben* kommt Marcus direkt zu dem Bereich, in dem er seine Aufgabe pflegt. Über den +-Button, erstellt er eine neue Aufgabe mit dem Titel *Literatur lesen*. Da diese Aufgabe regelmäßig Teil seines Lernplans ist, legt er eine Wiederholung fest. Die App ermöglicht es ihm, die Wiederholung individuell anzupassen. Er wählt eine benutzerdefinierte Einstellung, bei der die Aufgaben an bestimmten Wochentagen erscheinen. Für ihn sind es die Tage Montag, Dienstag, Freitag und Sonntag. Für die vorgesehene Bearbeitungszeit schätzt Marcus 20 Minuten. Dies ist lang genug, um in die Aufgabe einzutauchen, und gleichzeitig kurz genug, um keine Reizüberflutung durch Dauerbelastung auszulösen. Beim Speichern der Aufgabe reagiert die App mit einer kleinen Bestätigung, ohne dass Pop-ups, Geräusche oder Animationen erscheinen. In den Einstellungen kann Marcus den Erinnerungstyp anpassen. Anstelle von Benachrichtigung mit Sound setzt Marcus die Erinnerung auf Vibration, aufgrund seiner Empfindlichkeit gegenüber Tönen. Das Erfolgsszenario ist dabei, dass die Interaktion reibungslos verläuft. Marcus hat eine stressfreie, routinierte Aufgabe erstellt.

Use Case 3: Timer-Session mit Ablenkungsmanagement

Das Nutzungskontext beginnt mit einem Moment, wo Lisa beschließt, für zwei Stunden konzentriert für ihre bevorstehende Klausur zu lernen. Ihre Lernmaterialien liegen bereit sowie ihr Smartphone. Aus Routine öffnet sie die App und die Benutzeroberfläche begrüßt sie freundlich und ansprechend. Direkt auf der Startseite wählt sie die Aufgabe *Mathe Kapitel 4-6 wiederholen* aus und startet direkt eine Session. Mit dem Start des Timers wechselt die App in den Fokus-Modus. Der Bildschirm zeigt nun einen beruhigenden Hintergrund sowie einen sichtbaren Timer. Push-Nachrichten anderer Apps werden automatisch stummgeschaltet, was besonders wichtig für Lisa ist, da sie häufig durch Benachrichtigungen aus ihrer Konzentration gerissen wird. Nach zehn Minuten wird Lisa unruhig und spielt mit den Gedanken, ihr Socal Media zu kontrollieren, jedoch sieht sie die kleine Nachricht auf dem Bildschirm. Diese

Motivationssprüche reichen für Lisa aus, sich nicht ablenken zu lassen und weiter zu lernen. Nach 25 Minuten ertönt ein leiser Ton, welcher das Ende bzw. die Pause der Fokuszeit signalisiert. Mit dem Ton taucht eine Nachricht auf, dass Lisa es geschafft hat. Die fünfminütige Pause startet automatisch und in dieser Zeit kann Lisa machen, wonach es ihr beliebt. Sobald die Pause endet, ertönt ein Ton mit der Nachricht, ob sie bereit für die nächste Runde sei. Jedoch hat Lisa die Wahl und sie kann den Timer abbrechen. Im Idealfall schafft es Lisa an diesem Tag vier solcher Pomodoro-Einheiten und erreicht somit ihre zwei Stunden konzentriertes Lernen. Selbst bei Unterbrechungen durch ein Ereignis wird die App automatisch gestoppt und fragt später Lisa, ob sie weitermachen möchte, bzw. neustarten möchte.

Use Case 4: Pomodoro-Technik

Ein weiteres Anwendungsszenario lässt sich anhand der Persona Anna veranschaulichen. Nachdem Anna ihre Hausarbeit über SQL-Datenbanken erfolgreich in Teilaufgaben untergliedert hat, steht sie vor der Herausforderung, die zeitintensive Teilaufgabe *Literaturrecherche (2h)* zu bewältigen. Anna navigiert durch Antippen zur entsprechenden Teilaufgabe, wodurch sie automatisch in die Timer-Ansicht der App gelangt. Das Interface bietet ihr die Möglichkeit, die Pomodoro-Technik zu aktivieren. Dieser Ansatz zur Zeitverwaltung ermöglicht ihr, längere Arbeitsperioden in Intervallen zu unterteilen. Die App bietet standardmäßig das klassische Pomodoro-System von 25 Minuten Arbeitszeit, gefolgt von 5 Minuten Pause. Anna nutzt jedoch die Einstellung von Pomodoro und passt die Parameter ihren individuellen Bedürfnissen an. Sie stellt 30-minütige Arbeitsintervalle bei 5-minütigen Pausenzeiten ein. Die App berechnet automatisch die erforderliche Anzahl der Pomodoro-Zyklen basierend auf der geschätzten Aufgabendauer. Für die 90-minütige Literaturrecherche werden drei Arbeitszyklen à 30 Minuten benötigt, unterbrochen von zwei 5-minütigen Pausen zwischen den Zyklen. Die App visualisiert den Fortschritt durch eine Zyklusanzeige, die Anna über ihren aktuellen Status informiert. Während der Arbeitsphase läuft der Timer im Vordergrund und bietet Anna eine klare zeitliche Orientierung. Nach Abschluss jedes Zyklus erhält sie eine visuelle Benachrichtigung über das Ende der Arbeits- bzw. Pausenzeit. Nach erfolgreichem Abschluss aller drei Pomodoro-Zyklen und damit der gesamten Teilaufgabe Literaturrecherche zeigt die App Anna eine motivierende Erfolgsmeldung. Anna hat durch die Anwendung der Pomodoro-Technik eine zeitintensive Teilaufgabe erfolgreich bewältigt, ohne von der Gesamtdauer überwältigt zu werden.

Use Case 5: Streak-Management

In diesem Szenario hat Lisa bereits sechs Tage in Folge Aufgaben erledigt. Heute wäre der siebte Tag und somit hätte sie eine Belohnung erhalten. Jedoch fühlt sich Lisa überfordert und müde und möchte heute nichts erledigen. Dennoch öffnet sie die App und die App zeigt ihr den aktuellen Streak-Status und dass sie nur noch eine Aufgabe erledigen muss. Dann hat sie den 7-Tages-Streak erreicht. Die App zeigt ihr

zwei Möglichkeiten. Eine davon ist es, dass sie eine kleine Aufgabe erledigt, die nur fünf Minuten dauert. Und die zweite Möglichkeit wäre die *Freeze-Day*-Funktion. Der Streak bleibt bei sechs Tagen, wird nicht unterbrochen und am nächsten Tag würde der Streak weiterlaufen und Lisa hätte ihren 7-Tages-Streak erreicht. SIe verliert dadurch keine XP, stattdessen bekommt sie eine motivierende Rückmeldung.

3.2 Funktionale Anforderungen

Funktionale Anforderungen beschreiben die spezifischen Eigenschaften und Leistungen, die das System bereitstellen muss, um die Bedürfnisse und Erwartungen der Nutzerzielgruppe zu erfüllen (Al-Msie'deen et al., 2021). Im Kontext dieser Lernhilfe-App stehen vor allem Funktionen im Fokus, die auf die Herausforderungen und Anforderungen neurodivergenter Personen ausgerichtet sind. Dazu zählen Aspekte wie Aufgabenorganisation, Zeitstrukturierung und Motivationserhalt. Die Anforderungen wurden basierend auf der theoretischen Analyse sowie den entwickelten Personas in den Abschnitt 3.1.1 sowie Abschnitt 3.1.2 formuliert. Im Rahmen dieser Entwicklung werden sowohl Basisfunktionen (MUSS-Anforderungen) als auch erweiterte Funktionen (SOLL/KANN-Anforderungen) berücksichtigt, um eine realistische Implementierung während der Entwicklung zu gewährleisten. Jede funktionale Anforderung ist klar benannt und umfasst eine kurze Beschreibung sowie eindeutige Akzeptanzkriterien.

3.2.1 FA01 - Registrierung

Die Authentifizierung sichert die Identifikation der Nutzer und bildet die Grundlage für personalisierte Funktionen sowie Datenschutz. Die App nutzt eine E-Mail-basierte Registrierung zusammen mit Passwort-Authentifizierung.

Nr.	**Bezeichnung**	**Beschreibung**	**Priorität**
FA01a	Registrierung	Nutzer kann ein neues Konto erstellen mit folgenden Daten: • Benutzername (optional) • E-Mail-Adresse • Passwort	MUSS
FA01b	E-Mail-Validierung	• App validiert die E-Mail-Adresse • Format und Eindeutigkeitsprüfung • Nutzer erhält E-Mail-Bestätigung zur Verifizierung	MUSS

FA01c	Passwort-Sicherheit	• Nutzer kann Passwort Anzeigen/Verstecken • Nutzer muss ein Passwort von einer Länge von mindestens 8 Zeichen eingeben	MUSS

Tabelle 1: FA01 - Registrierung

Die Anforderung gilt als erfüllt, wenn ein Benutzerkonto erfolgreich in der Datenbank angelegt wurde und der Nutzer anschließend zur Startseite der App weitergeleitet wird. Die Registrierung muss dabei innerhalb von zwei Minuten erfolgen und darf nur mit einer zuvor noch nicht registrierten E-Mail-Adresse erfolgen.

3.2.2 FA02 - Anmeldung

Die Anmeldefunktion erlaubt registrierten Nutzern einen sicheren Zugriff auf ihre personalisierten Daten und Funktionen.

Nr.	**Bezeichnung**	**Beschreibung**	**Priorität**
FA02a	Login	Nutzer kann sich mit bestehenden Zugangsdaten anmelden: • E-Mail-Adresse • Passwort	MUSS
FA02b	Passwort-Wiederherstellung	Nutzer kann Passwörter zurücksetzen	KANN
FA02c	Angemeldet bleiben	Nutzer kann sich entscheiden, ob er angemeldet bleiben oder sich bei App-Start jedes Mal neuanmelden möchte	KANN

Tabelle 2: FA02 - Anmeldung

Die Anforderung ist erfüllt, wenn sich der Nutzer erfolgreich authentifizieren kann, eine gültige Session erzeugt und gespeichert wird. Die Hauptanwendung muss fehlerfrei geladen werden und alle benutzerspezifischen Inhalte sind verfügbar. Beim Auftreten eines Fehlers reagiert die App mit einer Meldung, die auf Sicherheitsbedenken hinweist. Beispielsweise werden Zugangsdaten, die nicht korrekt eingegeben werden, mit einer entsprechenden Fehlermeldung angezeigt. Die Anmeldung muss innerhalb von 20

Sekunden erfolgen und Rückmeldungen sind verständlich sowie nutzerfreundlich formuliert.

3.2.3 FA03 - Aufgabenverwaltung

Die Aufgabenverwaltung stellt eine der Hauptfunktionen der App dar und ermöglicht Nutzern eine systematische Organisation ihrer To-do-Listen für Aufgaben. Die App führt vollständige CRUD-Operationen (Create, Read, Update, Delete) durch und bietet zusätzliche Funktionen für wiederkehrende Aufgaben sowie flexible Priorisierungen.

Nr.	Bezeichnung	Beschreibung	Priorität
FA03a	Aufgabenerstellung	Nutzer kann neue Aufgaben mit folgenden Attributen erstellen: • Titel (Pflichtfeld) • Beschreibung (optional) • Priorität (optional) • Geschätzte Bearbeitungszeit (optional) • Fälligkeitsdatum und -zeit (optional) • Wiederkehrende Aufgaben konfigurieren (optional) – Täglich – Wöchentlich – Benutzerdefiniert (spezifische Wochentage)	MUSS
FA03b	Aufgabe speichern	Nutzer kann Aufgaben speichern mit nur einem Pflichtfeld (Titel)	MUSS
FA03c	Status-Management	Nutzer kann Aufgaben-Status ändern für mehr Organisation: • Offen • In Bearbeitung • Abgeschlossen • Archivieren	SOLL
FA03d	Aufgaben bearbeiten	Nutzer kann die Bearbeitung aller Aufgabenattribute vornehmen	SOLL

FA03e	Aufgaben löschen	• Nutzer kann Aufgaben entfernen • Nutzer bekommt ein Bestätigungsdialog bei endgültiger Löschung	SOLL

Tabelle 3: FA03 - Aufgabenverwaltung

Die Anforderung wird als erfüllt betrachtet, wenn neue Aufgaben vollständig erstellt, korrekt in der Datenbank gespeichert und innerhalb von 200 Millisekunden in der Aufgabenliste sichtbar sind. Die App gewährleistet zudem, dass wiederkehrende Aufgaben automatisch rechtzeitig generiert werden. Neue Aufgaben kann man innerhalb von 20 Sekunden erstellen, da mindestens die Titel-Eingabe erforderlich ist. Die Änderungen von bearbeiteten und gelöschten Aufgaben sind sofort in der App sichtbar. Im Falle fehlerhafter Eingaben gibt die App klare, benutzerfreundliche Rückmeldungen.

3.2.4 FA04 - Timer

Der Timer stellt mit der Aufgabenverwaltung zusammen die Hauptfunktionen der App dar. Es dient als Produktivitätsunterstützung und ermöglicht strukturierte Arbeitseinheiten mit flexiblen Zeitintervallen. Die App implementiert sowohl einfache Timer als auch komplexere Pomodoro-Zyklen mit Fokus-Unterstützung.

Nr.	**Bezeichnung**	**Beschreibung**	**Priorität**
FA04a	Basis-Timer	• Nutzer kann direkt mit einem Klick auf die Aufgabe den Timer starten • Nutzer kann die Timer-Längen anpassen	MUSS
FA04b	Pomodoro-System	• Nutzer kann strukturierte Arbeits-/Pausenzyklen nutzen – Standard: 25 Min. lernen/ 5 Min Pause • Nutzer erhält Benachrichtigung bei automatischem Wechsel • Nutzer kann die Intervalle anpassen	SOLL
FA04c	Fokus-Modus	Nutzer erhält Fokus-Unterstützung während der Session und Motivationsanzeigen	SOLL

FA04d	Visueller Fortschritt	Nutzer kann seinen Countdown visuell in seinen Ringdiagramm sehen	MUSS

Tabelle 4: FA04 - Timer

Die Anforderung ist erfüllt, wenn der Timer nach Nutzerinteraktion binnen 200 ms startet und während seiner gesamten Zeit stabil funktioniert. Diese Response-Time-Grenze orientiert sich an aktuellen HCI-Standards, die zeigen, dass Interaktionszeiten unter 200 ms als unmittelbar responsiv wahrgenommen werden (Pushpakumar et al., 2023). Der Fortschrittsring aktualisiert sich flüssig und liefert mindestens einmal pro Sekunde visuelles Feedback, indem sich der Ring mit Farbe füllt. Der Übergang zwischen Arbeits- und Pausenphasen erfolgt automatisch und die Intervalle sind anpassbar. Der Zykluszähler (z. B. „2 von 4“) bleibt jederzeit sichtbar, sodass Nutzer ihren Fortschritt im Pomodoro-System nachvollziehen können. Zudem wird nach Ablauf des Timers der Aufgabenstatus automatisch aktualisiert und die Nutzer kehren zur Startseite zurück.

3.2.5 FA05 - Fortschrittsanzeige

Die Fortschrittsanzeige bietet Nutzern transparente Einblicke in ihre Produktivitätsentwicklung durch dynamische Charts und motivierende Nachrichten ohne Leistungsdruck.

Nr.	**Bezeichnung**	**Beschreibung**	**Priorität**
FA05a	Progress-Bar	Nutzer kann seine aktuelle Bearbeitungsfortschritt sehen: • Prozentuale Fortschritssanzeige der erledigten Aufgabe	MUSS
FA05b	Fortschritts-Nachrichten	Nutzer erhält positive Verstärkung bei Meilensteinen	KANN

Tabelle 5: FA05 - Fortschrittsanzeige

Die Anforderung gilt als erfüllt, wenn Fortschrittsdaten visuell klar und binnen 500 Millisekunden nach Aufgabenstatusänderung dargestellt werden, ohne dabei Leistungsdruck zu erzeugen. Die Progress-Bar enthält eine Prozentangabe, wodurch der aktuelle Bearbeitungsprozess nachvollziehbar ist und sie sich automatisch bei jedem Aufgabenabschluss oder jeder Aufgabenänderung aktualisiert. Zusätzlich wird ein Wochenfortschritt durch grafische Elemente wie ein Balkendiagramm dargestellt, das täglich um Mitternacht neu berechnet wird. Um eine Abwechslung zu garantieren, sind mindestens fünf verschiedene Texte pro Fortschrittsbereich hinterlegt. Diese Vielfalt trägt zu einer anhaltenden Motivation bei.

Alle Fortschrittsinformationen sind transparent, gut verständlich und jederzeit aufrufbar.

3.2.6 FA06 - Belohnungssystem

Das Belohnungssystem beinhaltet spielerische Elemente zur Förderung intrinsischer Motivation (siehe Abschnitt 2.3.1) durch Erfahrungspunkte, Level-Progressionen, Challenges und Achievement-Mechanismen.

Nr.	Bezeichnung	Beschreibung	Priorität
FA06a	XP-Vergabe	Nutzer erhält XP für abgeschlossene Aufgaben: • XP nach Aufgabendauer: – < 15 Min. → 10XP – 15 – 40 Min. → 20XP – > 40 Min. → 40 XP	MUSS
FA06b	XP-Anzeige	Nutzer kann seine aktuelle XP einsehen	KANN
FA06c	Levelsystem	• Nutzer kann durch XP-Sammlung Leveln aufsteigen • Nutzer erhält eine visuelle Level-Up Rückmeldung • Nutzer kann sein aktuelles Level einsehen	MUSS
FA06d	Achievements	• Nutzer kann besondere Leistungen freischalten • Nutzer kann an wöchentliche Challenges teilnehmen	SOLL
FA06e	Flexible Regelung	Nutzer erhält Unterstützung bei schwierigen Aufgaben: • *Freeze-Days* - Nutzung	KANN

Tabelle 6: FA06 - Belohnungssystem

Die Anforderung gilt als erfüllt, wenn die Vergabe von XP sowie der Fortschritt im Level-System nachvollziehbar und motivierend gestaltet sind. XP werden direkt nach Abschluss einer Aufgabe gutgeschrieben und die Berechnung dafür erfolgt clientseitig, um eine unterbrechungsfreie Nutzererfahrung zu gewährleisten. Ein Levelaufstieg wird direkt ausgelöst, sobald die entsprechenden XP gesammelt wurden. Dies wird durch eine visuelle Rückmeldung unterstützt. Die XP-Leiste zeigt den Fortschritt deutlich

an und wird durch flüssige Animation ergänzt. Der aktuelle Level-Status wird sowohl numerisch als auch sprachlich dargestellt, z. B. *Level 5 – Lernender*. Um unterschiedlichen sensorischen Bedürfnissen gerecht zu werden, kann die Level-Up-Animation in den Einstellungen deaktiviert werden. Es werden wöchentliche Challenges angeboten, die jeden Montag neu generiert werden. Zudem gibt es einmalige Achievements mit Benachrichtigungen. Im Kontext von Streaks stehen Unterstützungsfunktionen zur Verfügung, wie etwa Freeze-Days oder Mini-Aufgaben zur Streak-Erhaltung. Bei einem Streak-Verlust soll ein Wiedereinstieg ermöglicht werden.

3.2.7 FA07 - Stimmungstracker

Das Stimmungstracking-System ermöglicht es den Nutzern, ihre Stimmung optional zu dokumentieren.

Nr.	Bezeichnung	Beschreibung	Priorität
FA07a	Schnelle Stimmungserfassung	Nutzer kann seine aktuelle Stimmung auf der Startseite schnell dokumentieren	MUSS
FA07b	Detaillierte Stimmungserfassung	Nutzer kann spezifische Aspekte bewerten basierend auf seiner Stimmung (optional) • Energielevel (0 – 100 • Fokuslevel (0 -100 • Stresslevel (0 – 100 • Notizen	SOLL
FA07c	Stimmungsverlauf	Nutzer kann seinen wöchentlichen/monatlichen Stimmungsverlauf einsehen	KANN

Tabelle 7: FA07 - Stimmungstracker

Die Funktion gilt als erfolgreich implementiert, wenn Nutzer ihre Stimmung zeiteffizient erfassen können. Die Auswahl erfolgt durch ein einfaches Tippen auf ein Emoji auf der Startseite. Die Daten werden daraufhin gespeichert und durch ein visuelles Feedback bestätigt. Die Interaktion mit den optionalen Zusatzfunktionen, wie den Kategorien für *Energie*, *Fokus* und *Stress* sowie dem Hinzufügen von Notizen, sollte innerhalb von einer Minute erfolgen. Alle Bedienelemente, insbesondere die Slider, müssen verzögerungsfrei auf Touch-Eingaben reagieren. Der Stimmungsverlauf soll in einem übersichtlichen Liniendiagramm mit Emoji-Markierung eingesehen werden.

3.2.8 FA08 - Einstellungen

Die Einstellungen ermöglichen eine umfassende Anpassung an individuelle Bedürfnisse und Vorlieben der Nutzer. Die App bietet Konfigurationsmöglichkeiten in den Bereichen Barrierefreiheit und Benachrichtigung.

Nr.	Bezeichnung	Beschreibung	Priorität
FA08a	Barrierefreiheit	Nutzer kann visuelle sowie sensorische Anpassungen vornehmen: • Reduzierte/ Keine Animationen • Größere Schrift • Vibration/Sound • Dark Mode	MUSS
FA08b	Benachrichtigung	Nutzer kann Benachrichtigungen anpassen: • Tägliche Erinnerung für Aufgaben • Streak Erinnerung	SOLL
FA08c	Abmeldung	Nutzer kann sich von der App abmelden	MUSS

Tabelle 8: FA08 - Einstellungen

Die Anforderung gilt als erfüllt, wenn alle gewählten Präferenzen korrekt übernommen und gespeichert werden. Die Änderungen müssen nach 200 ms sichtbar sein, ohne einen Neustart der App zu erfordern.

3.3 Nicht-Funktionale Anforderungen

Die nicht-funktionalen Anforderungen legen die Merkmale und Rahmenbedingungen einer Anwendung fest, die über die grundlegenden Funktionalitäten hinausgehen. Sie umfassen Aspekte wie Leistung, Benutzerfreundlichkeit, Sicherheit und Kompatibilität. Diese Aspekte sind entscheidend für eine erfolgreiche Implementierung sowie für die Akzeptanz durch die Nutzer.

3.3.1 NF01 - Cross-Plattform Kompatibilität

Die technische Grundlage der App muss eine umfassende Geräte- und Betriebssystemkompatibilität gewährleisten. Das Ziel besteht darin, eine maximale Zugänglichkeit für die Zielgruppe zu erreichen. Hierfür wird das Flutter-Framework eingesetzt, welches eine einheitliche Entwicklung ermöglicht. Dadurch entsteht eine gemeinsame Codebasis für iOS und Android. Dies garantiert native Performance durch

die Dart-Kompilierung. Gleichzeitig werden plattformspezifische UI-Anpassungen gemäß der jeweiligen Designrichtlinien ermöglicht. Flutter bietet mit seinem Material-Design-System für Android und dem Cupertino-System für iOS zwei unterschiedliche Designsprachen, die den jeweiligen Plattform-Guidelines entsprechen. Demgegenüber orientiert sich das Cupertino-System an Apples Human Interface Guidelines und zeichnet sich durch minimalistische Designelemente, subtile Farbverläufe und die charakteristische iOS-Typografie aus (Flutter, 2025). Daher wird darauf geachtet, das Betriebssystem von Android über der Version 8.0+ sowie bei iOS der Version 12.0+ zu halten. Dadurch wird garantiert, dass aktuelle mobile Geräte mit abgedeckt sind. Alle Kernfunktionen müssen plattformübergreifend gleich funktionieren, wobei Performance-Unterschiede in einem niedrigen Bereich liegen sollten.

3.3.2 NF02 - Performance und Responsivität

Die Anforderungen an die Performance stellen sicher, dass die Benutzererfahrung responsiv und flüssig ist. Spürbare Verzögerungen könnten insbesondere bei neurodivergenten Nutzern Ablenkung oder Frustration verursachen. Beim Start der App wird die Verbindung zur Datenbank im Hintergrund (asynchron) initialisiert, sodass die Benutzeroberfläche dabei nicht blockiert wird. Aufwendige Aufgaben, die mehr Rechenleistung benötigen, werden in getrennten Bereichen ausgeführt. So bleibt die Hauptoberfläche stets reaktionsfähig. Die Benutzeroberfläche reagiert direkt auf jede Interaktion, was zu einer flüssigen und angenehmeren Nutzungserfahrung führt. Um dieses reaktionsschnelle Verhalten sicherzustellen, erfolgen die visuellen Aktualisierungen im Haupt-Thread. Dieser Teil der App ist für die Darstellung zuständig. Ziel ist es dabei, dass diese Abläufe weniger als 16 Millisekunden in Anspruch nehmen. Der Hintergrund hierfür liegt darin, dass Bildschirme mit einer Bildwiederholrate von 60 Hz arbeiten, was einer Bilddarstellung von ca. 16 ms entspricht. Werden diese Berechnungen schneller abgeschlossen, kann jedes Frame rechtzeitig dargestellt werden, sodass ein gleichmäßiger Ablauf mit 60 Bildern pro Sekunde möglich bleibt. Überschreitet die Operation diese Schwelle, kommt es zu Verzögerungen oder Aussetzern, die Nutzer als störend wahrnehmen (Liu et al., 2025). Falls eine Aktion doch etwas länger dauert, informieren gezielte Ladeanzeigen die Nutzer darüber, dass im Hintergrund Verarbeitungen stattfinden.

3.3.3 NF03 - Offline-Funktionalität

Die App bietet umfassende Offline-Funktionen, um kontinuierliche Nutzbarkeit unabhängig vom Internet sicherzustellen. Daten wie Aufgaben, den Timer, XP/Level-Daten und Benutzereinstellungen werden lokal gespeichert und bleiben offline verfügbar. Diese Offline-Funktionalität wurde allerdings als

KANN-Anforderung gekennzeichnet, da das primäre Ziel der App nicht die offline-basierte Nutzung ist. Die Implementierung erfolgte vielmehr als zusätzliche Komfortfunktion für Situationen mit instabiler Internetverbindung. Die Kernfunktion der Offline-Nutzung garantiert, dass alle wesentlichen Funktionalitäten ohne Internetzugang zur Verfügung stehen. Ein visueller Offline-Indikator informiert die Nutzer über den aktuellen Verbindungsstatus. Die Akzeptanzkriterien definieren dementsprechend einen App-Start ohne Internet innerhalb von 600 ms sowie die Verfügbarkeit aller Kernfunktionen im Offline-Modus. Die exklusiven Funktionen, die nur online zur Verfügung gestellt werden können, werden bei fehlender Verbindung ausgeblendet. Da die App primär als online-basierte Lernhilfe-Anwendung konzipiert wurde, stellt die Offline-Funktionalität eine ergänzende Funktion dar, die den Nutzungskomfort erhöht, aber nicht zwingend erforderlich ist.

4 Konzeption

4.1 Architektur

Für die MindSpark-App ist es wichtig, eine durchdachte Architekturstrategie zu entwickeln, die sowohl technische Anforderungen als auch spezifische Bedürfnisse der Zielgruppe berücksichtigt. Die Architektur der App bildet die Grundlage und entscheidet über spätere Wartbarkeit und Erweitbarkeit der App. In diesem Zusammenhang wurden gleich mehrere Architekturmuster gewählt, da ein einzelnes Muster nicht alle Aspekte einer cloudbasierten Anwendung abdecken kann.

4.1.1 Architekturmuster

Im Folgenden werden drei Ansätze vorgestellt, die für die Konzeption besonders bedeutsam sind. Dazu zählen die Schichtenarchitektur, das Client-Server-Modell sowie die serviceorientierte Architektur. Jedes dieser Muster hat bestimmte Eigenschaften, die für die Entwicklung einer mobilen Anwendung eine wichtige Grundlage darstellen.

Die Schichtenarchitektur bildet das strukturelle Gerüst der MindSpark-App dar. Sie gliedert die Anwendungslogik in drei definierte Schichten. Die Präsentationsschicht, die Logikschicht und die Datenschicht. Die Aufgabe der Präsentationsschicht besteht in der Gestaltung der Benutzeroberfläche sowie der Interaktion mit dem Nutzer. Die Logikschicht implementiert die Geschäftsregeln, während die Datenschicht den Zugriff auf externe Datenquellen verarbeitet. Diese Trennung fördert eine wartbare und erweiterbare Struktur für die auf ADHS und ASS fokussierte Produktivitäts-App. Jede Schicht hat bestimmte Verant-

wortlichkeiten und kommuniziert ausschließlich mit den darunterliegenden Schichten. Im Folgenden wird der Datenfluss beim Ändern des Status einer Aufgabe durch alle drei Schichten dargestellt.

Präsentationsschicht

Die Präsentationsschicht wird durch die genannte `TaskScreen`-Klasse repräsentiert, welche auf Benutzerinteraktionen reagiert und Änderungen des Aufgaben-Status verarbeitet.

```dart
Future<void> handleToggleCompletion(String taskId) async {
    final result = await taskService.toggleTaskCompletion(taskId);

    if (result.isSuccess) {
        showSuccessSnackBar(result.message ?? 'Status geaendert');
    } else {
        showErrorSnackBar(result.error ?? 'Fehler');
    }
}
```

Listing 1: Präsentationsschicht - TaskScreen

Die Methode zeigt, wie die Präsentationsschicht sich bei Benutzerinteraktionen typischerweise verhält. Sie wird aktiviert durch ein Long-Tap auf die entsprechende Aufgabe vom Nutzer, um dessen Status zwischen offen und abgeschlossen zu wechseln. Hierbei wird nur die `taskId` als Parameter übergeben. Alle Geschäftslogiken werden an die darunterliegende Logikschicht delegiert, sobald man `taskService.toggleTaskCompletion` aufruft. Nach einem asynchronen Aufruf erhält diese Methode ein strukturiertes `TaskServiceResult`-Objekt zurück. Dieses Objekt dient der Anzeige eines erfolgreichen Abschlusses der Ausführung oder eines aufgetretenen Fehlers. Gleichzeitig dient es als eine Art Zwischenebene, über die Informationen einheitlich zwischen der Logikschicht und der Benutzeroberfläche ausgetauscht werden können. Daraufhin reagiert die Benutzeroberfläche, wie in Listing 1 zu erkennen, mit SnackBar-Nachrichten, ohne Details zu Datenbanktabellen oder SQL-Abfragen zu kennen. Der Fokus der Schicht liegt ausschließlich auf der Interaktion mit den Nutzern und dem visuellen Feedback. Diese klare Trennung reduziert unnötige Abhängigkeiten und ermöglicht eine flexible Wartbarkeit der Oberfläche.

Logikschicht

Die Logikschicht wird durch die `TaskService`-Klasse repräsentiert. Der `TaskService` implementiert die Methode `toggleTaskCompletion()`, welche von der Präsentationsschicht aufgerufen wird, um Geschäftslogik für Statusänderungen zu koordinieren.

```dart
Future<TaskServiceResult> toggleTaskCompletion(String taskId) async {
    final task = _allTasks.firstWhere((t) => t.id == taskId);
    final isCurrentlyCompleted = _isCompleted[taskId] ?? false;

  // Geschaeftslogik: Aufgaben mit Subtask oder Hauptaufgaben duerfen nicht einfach
  // umgeschaltet werden
    if (task.parentTaskId != null || await hasSubTasks(taskId)) {
        return TaskServiceResult.error('Der Abschluss dieser Aufgabe wird über den
    Timer oder die Teilaufgaben verwaltet.',
    );
    }

  // Status umschalten und sofort UI aktualisieren
  _isCompleted[taskId] = !isCurrentlyCompleted;
  final newStatus = _isCurrentlyCompleted ? TaskStatus.open : TaskStatus.completed;

    try {
        final result = isCurrentlyCompleted
            ? await updateTaskStatus(taskId, TaskStatus.open)
            : await completeTask(taskId);
        return result.isSuccess
            ? TaskServiceResult.success('Status erfolgreich geändert')
            : TaskServiceResult.error(result.error ?? 'Fehler beim Aktualisieren');
  } catch (e) {
    return TaskServiceResult.error('Fehler beim Aktualisieren: $e');
  }
}
```

Listing 2: Logikschicht - TaskService

Zunächst wird anhand der übergebenden `taskId` die zugehörige Aufgabe aus dem lokalen Zwischenspeicher (`allTask`, Zeile 2) gesucht. Anschließend erfolgt seitens des Services eine Prüfung der `taskCompletion`-Map, um zu ermitteln, ob die Aufgabe als abgeschlossen oder noch offen gilt. Hier greift die Geschäftslogik zu. Sobald eine Aufgabe entweder eine übergeordnete Aufgabe (`parentTaskId`) aufweist oder selbst eine Unteraufgabe besitzt, darf ihr Status nicht direkt geändert werden. Stattdessen muss der Abschluss über den Timer oder die Logik der Teilaufgabe erfolgen (siehe Zeile 7 - 10). In diesem Fall wird sofort ein entsprechender Fehler zurückgegeben. Wenn keine Abhängigkeiten bestehen, wird der Status lokal von *offen* auf *abgeschlossen* umgeschaltet. Dabei greift er auf den `TaskStatus`-Enum. Die Benutzeroberfläche kann diese Veränderungen unmittelbar visuell darstellen. Die tatsächliche Speicherung des neuen Status in der Datenbank wird dann an die Methode `updateTaskStatus()` oder bei Abschluss einer Aufgabe an `completeTask()` übergeben, die beide zur Datenschicht weiterleiten. Ein zentrales

Merkmal in dieser Schicht ist die strukturierte Fehlerbehandlung. Alle möglichen Fehler, die bei der Statusänderung auftreten können, wie z. B. Datenbankprobleme oder Regelverletzung, werden abgefangen und als `TaskServiceResult` zurückgegeben.

Datenschicht

Die Datenschicht wird durch die Klasse `MySupabaseClient` dargestellt und veranschaulicht die technische Komplexität der Datenbankverbindung. Die Implementierung kapselt sämtliche Anfragen an die Supabase-Datenbank und extrahiert dadurch die Kommunikation mit dem Backend vollständig aus den anderen Architekturschichten.

```dart
Future<List<Map<String, dynamic>>> update({
    required String table,
    required Map<String, dynamic> data,
    required String where,
    required dynamic whereValue,
}) async {
    final response = await _client.from(table).update(data).eq(where, whereValue).
    select();
    return List<Map<String, dynamic>>.from(response);
}
```

Listing 3: Update-Methode in der Datenbank

Die Datenschicht bietet eine Schnittstelle für die verschiedenen CRUD-Operationen. Die zentrale Methode `update()` akzeptiert vier Parameter. Diese umfassen den Namen der Zieltabelle als String, ein Map-Objekt mit den zu aktualisierenden Werten, ein weiteres Map zur Definition der WHERE-Kriterien als Filter sowie den entsprechenden Vergleichswert. Mithilfe des Supabase-Query-Builders wird aus diesen Informationen eine SQL-Update-Anweisung erstellt und ausgeführt. Das Ergebnis dieser Operation, und zwar die aktualisierten Datenzeilen, wird zurückgegeben. Ebenso, wenn ein Fehler in dieser Schicht auftritt, wird dieser erkannt und an die Geschäftslogik weitergereicht. Ein weiterer Vorteil dieser Schicht ist, dass sie durch die klare Trennung zwischen Anwendungslogik und Datenspeicherung die Wartbarkeit und Skalierbarkeit der App verbessert. `MySupabaseClient` verarbeitet ausschließlich Map-Strukturen und führt Datenbankoperationen aus. Durch diese Trennung kann ein Wechsel der Datenbanktechnologie oder die Einführung zusätzlicher Anforderungen mit minimalem Anpassungsaufwand vorgenommen werden, ohne Änderungen an der Geschäftslogik zu erzwingen. Beispielsweise arbeitet die Schicht nicht direkt mit dem `TaskStatus`-Enum der Logikschicht zusammen, sondern wandelt diesen in einfache Strings um, wie etwa von `TaskStatus.completed` zu *completed*, um Kompatibilität mit der Datenbank sicherzustellen. Die Architektur sorgt nicht nur für eine höhere Sicherheit durch definierte Zugriffs-

punkte, sondern man kann sie ebenfalls an die besonderen Bedürfnisse neurodivergenter Nutzergruppen anpassen. Durch die flexible Gestaltung der Datenverarbeitung lassen sich individuelle Anforderungen einfacher integrieren und bei Bedarf erweitern.

Das `TaskStatus`-Enum ergänzt die Architektur durch ein zentrales Model, das alle gültigen Zustände einer Aufgabe beschreibt. Die Logikschicht nutzt das Enum für Geschäftsentscheidungen, z. B. für das Ändern des Aufgabenstatus. Wie in Listing 4 dargestellt, wird bei der Eingabe einzelner Aufgaben der aktuelle Status aus der Datenbank abgerufen. Dieser Status wird dann von der Methode `TaskModel.fromJson` in ein typisiertes Objekt umgewandelt. Anschließend erfolgt die Interpretation mithilfe des Enums. Dadurch kann sofort festgestellt werden, ob die Aufgabe als abgeschlossen gilt. Diese Information wird dann entsprechend in der lokalen Status-Map `taskCompletion` vermerkt. Die Datenschicht hingegen verwendet die String-Werte zum Speichern in der Datenbank und die Präsentationsschicht wiederum profitiert von dem Enum, da sie sich auf die geprüften Statuswerte verlassen kann. Dadurch wird vermieden, dass ungültige oder uneinheitliche Zustände im System auftreten. Diese gemeinsame Nutzung durch alle Schichten sorgt für Typsicherheit, Verständlichkeit und saubere Trennung der Zuständigkeiten.

```
enum TaskStatus {
    open('open'),
    completed('completed'),

    const TaskStatus(this.value);
    final String value;

    static TaskStatus fromString(String value) {
        return TaskStatus.values.firstWhere((e) => e.value == value);
  }
}
Future<TaskModel?> getTaskById(String taskId) async {
    try {
       final data = await database.selectWithWhere('tasks', 'id', taskId);
          if (data.isNotEmpty) {
            final task = TaskModel.fromJson(data.first);
            final index = _allTasks.indexWhere((t) => t.id == task.id);
            if (index == -1) {
              _allTasks.add(task);
              _taskCompletion[task.id] = task.status == TaskStatus.completed;
              _tasksStreamController.add(_allTasks);
            }
            return task;
          }
```

```
    }
}
```

Listing 4: TaskStatus-Enum

Die Schichtenarchitektur bildet das Grundgerüst der MindSpark-App. Sie definiert die Organisation der Anwendungslogik, berücksichtigt jedoch nicht die Kommunikationsmuster und die Verteilung der Systemkomponenten. Für eine umfassende Betrachtung der Architektur ist es notwendig, diese Aspekte zu analysieren. Die Schichtenarchitektur regelt die Trennung innerhalb der Client-Anwendung. Darüber hinaus ergänzen weitere Architekturmuster die Struktur.

Client-Server-Architektur

Die Client-Server-Architektur fokussiert sich auf die Kommunikation zwischen der Flutter-App und dem Supabase-Backend. Es stellt ein grundlegendes Architekturmuster dar, das die Systemkomponenten in zwei Rollen unterteilt: den Client und den Server. Der Client übernimmt die Funktion einer Präsentations- und Interaktionsschicht, während der Server für die zentrale Datenverarbeitung und Datenspeicherung zuständig ist. Diese Architektur ermöglicht es, Anwendungslogik und Datenmanagement auf Systeme zu verteilen. Dies fördert Skalierbarkeit, Wartbarkeit und Unterstützung für verschiedene Endgeräte (Nyabuto et al., 2024). In der MindSpark-App wird das beschriebene Modell durch eine Flutter-Client-Anwendung realisiert. Der Flutter-Client stellt die Benutzeroberfläche bereit und verwaltet lokale Zustandsinformationen. Das Supabase-Backend dient als zentraler Datenspeicher, welcher mit einer PostgreSQL-Datenbank sowie Authentifizierungsdiensten und Realtime-Funktionalitäten ausgestattet ist. Die Trennung ermöglicht den Nutzern die Verwendung der MindSpark-App auf unterschiedlichen Geräten, während eine zentrale Synchronisierung aller Daten erfolgt.

```dart
class MySupabaseClient {
    static MySupabaseClient? _instance;
    static MySupabaseClient get instance => _instance ??= MySupabaseClient._();
    MySupabaseClient._();
    static const String supabaseUrl = 'supabaseURL';
    static const String supabaseAnonKey = 'supabasekey';
    SupabaseClient get client => Supabase.instance.client;

  /// Initialisiert Supabase
    static Future<void> initialize() async {
        await Supabase.initialize(
        url: supabaseUrl,
        anonKey: supabaseAnonKey,
```

```
    );
  }
}
```

Listing 5: Client Implementierung

Der `SupabaseClient` dient als zentrale Schnittstelle zur Kommunikation und kapselt alle serverbezogenen Operationen. Durch die Singleton-Implementierung in Listing 5 (ab Zeile 10) wird sichergestellt, dass während der gesamten Laufzeit der App eine konsistente Verbindung zum Server besteht. Die Methode `initialize()` baut beim Start der App eine Verbindung zum Supabase-Server auf, indem sie die konfigurierte URL sowie den anonymen API-Schlüssel nutzt. Der Client kümmert sich um Netzwerkkommunikation sowie um lokale Zwischenspeicherung von Daten für Offline-Nutzung. Aufgrund dieser Trennung bleibt jede andere Komponente unabhängig von spezifischen Implementierungsdetails des Servers. Das Supabase-Backend bildet die Serverseite des Client-Server-Modells ab und übernimmt die zentrale Datenspeicherung und Datenverarbeitung.

```dart
Future<AuthResult> signUp({
    required String email,
    required String password,
    required String username,
  }) async {
    try {
      final response = await supabase.auth.signUp(
            email: email,
            password: password,
            data: {'username': username},
      );
      final userModel = UserModel.create(username: username, email: email);

      await database.insert(
            table: 'users',
            data: {
              'id': response.user!.id,
              'email': email,
              'username': username.isEmpty ? 'User' : username,
              //...
            },
      );
      return AuthResult.success(response.user!);
```

Listing 6: Server Implementierung

Wie in Listing 5 zu sehen, verwaltet der Authentifizierungsdienst Benutzerregistrierungen und -anmeldungen über REST-API-Endpunkte. Während des Registrierungsprozesses führt der Server sowohl die Benutzerauthentifizierung als auch das Anlegen eines zugehörigen Benutzerprofils in der PostgreSQL-Datenbank durch. Im konkreten Ablauf wird dabei zunächst die `signUp()-Methode` von Supabase Auth aufgerufen, um einen neuen Account mit E-Mail und Passwort zu registrieren. Ergänzende Profildaten wie der Benutzername werden bereits in diesem Schritt mitgegeben. Anschließend erfolgt eine separate Eintragung in die Nutzertabelle der Datenbank. Neben der vom Auth-System generierten User-ID werden dabei auch individuelle Benutzereinstellungen wie Barrierefreiheits-Optionen oder bevorzugte Benachrichtigungsoptionen gespeichert. Die Serverarchitektur trennt klar zwischen Authentifizierung über Supabase Auth und Datenmanagement über PostgreSQL.

Das vorliegende Beispiel zeigt die Interaktion zwischen Client und Server im Authentifizierungsprozess.

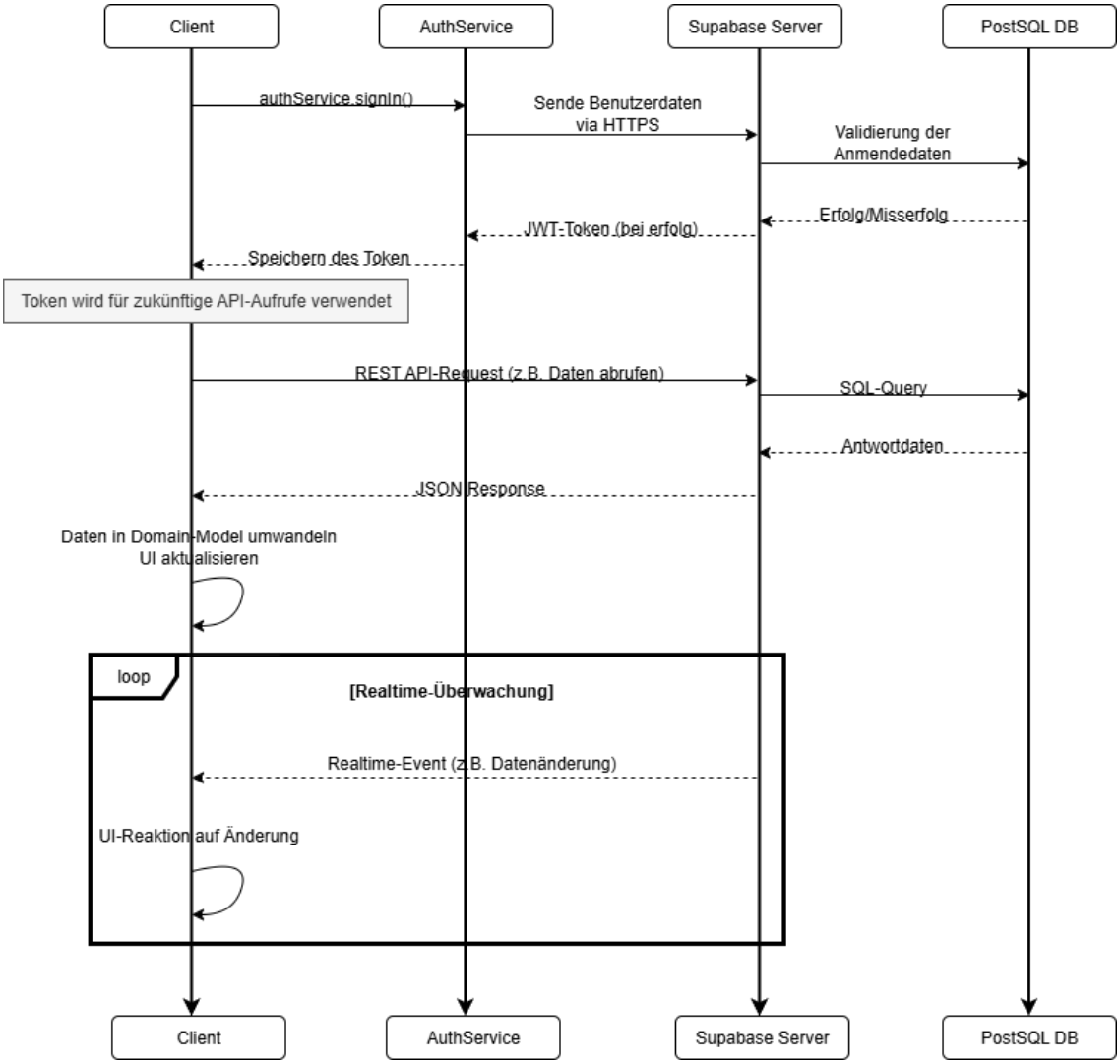

Abbildung 1: Realtime-Funktion - Authentifizierungsprozess

Der Flutter-Client imitiert eine Anmeldeanfrage mittels der Methode `authService.signIn()`, wobei

er Benutzerdaten über HTTPS an den Supabase-Autorisierungsserver sendet. Nach erfolgreicher Validierung der Anmeldedaten generiert der Server bei erfolgreicher Authentifizierung einen JWT-Token und sendet diesen zurück an den Client als Antwort. Der Client speichert das Token lokal ab und verwendet es für alle nachfolgenden API-Aufrufe zur Autorisierung. Bei weiteren Datenbankoperationen folgt die Kommunikation einem ähnlichen Muster. Der Client sendet REST-API-Anfragen an den Server, welcher sie in SQL-Abfragen gegen die PostSQL-Datenbank übersetzt und Ergebnisse als JSON-Antwort zurückgibt. Währenddessen wandelt der Client diese JSON-Daten in lokale Domänen-Modelle zur Aktualisierung seiner Benutzerfläche um. Der Client überwacht kontinuierlich Realtime-Ereignisse vom Server und kann so sofort auf Änderungen anderer Clients reagieren. Diese Architektur gestattet es der MindSpark-App, eine zentrale Datenverwaltung sicherzustellen, während gleichzeitig eine responsive Benutzererfahrung bereitgestellt wird.

Die Client-Server-Architektur bildet die Grundlage für die Kommunikation zwischen der Flutter-App und dem Supabase-Backend. Sie dient als Grundgerüst für die Verwaltung und Synchronisation von Daten. Während diese Architektur die externe Kommunikation regelt, ist eine interne Struktur für die Geschäftslogik auf der Client-Seite erforderlich. Die verschiedenen Funktionsbereiche der MindSpark-App, wie z. B. die Aufgabenverwaltung, die Timer-Funktion und die Benutzerauthentifizierung, verlangen nach wartbarer Anwendungslogik. An dieser Stelle erweitert die Service-Oriented Architecture (SOA) die bestehenden Architekturmustern.

Serviceorientierte Architektur

Die serviceorientierte Architektur (SOA) ist ein Architekturmuster mit dem Ansatz für den Aufbau von Software, der große Anwendungen in kleinere und klar definierte Einheiten aufteilt. Jeder dieser Services erfüllt eine spezifische Aufgabe innerhalb der Geschäftslogik und bietet seine Funktionen über Schnittstellen an. Diese Struktur führt zu einem flexiblen und erweiterbaren System. Ein zentrales Prinzip von SOA ist die Kapselung. Jeder Service verwaltet seine eigenen Prozesse und Daten. Das macht die Verantwortlichkeiten übersichtlicher und reduziert die Komplexität der einzelnen Teile, was wiederum die Wartbarkeit verbessert. Die Art der Kopplung zwischen Services kann dabei variieren und unterschiedliche architektonische Eigenschaften mit sich bringen. Ein weiterer Aspekt ist die Wiederverwendbarkeit. Ein Service kann von mehreren Anwendungsteilen genutzt werden, was den Entwicklungsaufwand reduziert und die Effizienz steigert (Chaves & De Freitas, 2019). In der MindSpark-App wird dieses Architekturprinzip durchgängig umgesetzt. Die Geschäftslogik ist in Services unterteilt, die als Vermittler zwischen der Benutzeroberfläche und dem Backend dienen. Damit bilden sie die Logikschicht innerhalb der dreischichtigen Architektur und gewährleisten eine klare Trennung von Darstellung, Logik und Daten. In diesem Kontext rückt die Art der Service-Kapselung in den Fokus der Architektur.

```dart
class CreateTaskService {
    final database = MySupabaseClient.instance;
    final authService = AuthService.instance;

    Future<TaskResult> createTaskFromUI(CreateTaskRequest request) async {
        try {
          final userId = authService.userId;
          if (userId == null) {
            return TaskResult.error('Benutzer nicht eingeloggt');
          }
        }
    }

   Future<TaskResult> createTask({
        required String userId,
        required String title,
        Priority priority = Priority.medium,
  }) async {
        try {
          final task = TaskModel.create(userId: userId, title: title.trim(),
    priority: priority);
          final taskData = task.toJson();

          final result = await database.insert(table: 'tasks', data: taskData);
            if (result.isEmpty) {
                return TaskResult.error(
                  'Aufgabe konnte nicht in der Datenbank gespeichert werden',
                );
          }
          return TaskResult.success(savedTask, 'Aufgabe erfolgreich erstellt');
    }
}
```

Listing 7: Service Kapselung

Die Kapselung in der App orientiert sich am Single-Responsibility-Prinzip, was bedeutet, dass jeder Service eine klar definierte Geschäftsdomäne verwaltet. Der `CreateTaskService` nutzt die Kapselung beispielsweise mithilfe des Singleton-Patterns (`AuthService.instance`) (siehe Listing 7, Zeile 5). Er fasst alle aufgabenbezogenen Operationen, wie `createTask()`, `updateTaskStatus()` etc., zusammen. Die technische Realisierung dieser Kapselung erfolgt durch private Instanzvariablen sowie öffentliche Methoden. So bleiben interne Implementierungsdetails vor anderen Services verborgen. Der `CreateTaskService` stellt eine Schnittstelle bereit, welche sowohl synchrone (lokale Datenverarbei-

tung) als auch asynchrone (Datenabfrage ans Backend) Datenbank-Interaktionen über `MySupabaseClient.instance` integriert. Diese Architektur ermöglicht es, weitere Services wie `AuthService` unabhängig zu entwickeln und zu testen, ohne negative Auswirkungen auf die Gesamtarchitektur.

```dart
class TimerService extends ChangeNotifier {
    final ProgressService _progressService = ProgressService.instance;
    final TaskService _taskService = TaskService.instance;

    Future<void> _handleTaskCompletionLogic()
        async {
            if (xpAmount > 0) {
                final xpResult = await _progressService.addXP(
                  xpAmount,
                  'Hauptaufgabe abgeschlossen',
                  taskId: parentTask.id,
                );
            }
        }
}
```

Listing 8: Enge Kopplung

Neben der Kapselung spielt auch die direkte Kommunikation zwischen den einzelnen Services eine entscheidende Rolle für die Architektur. Dieses Prinzip wird durch eine klare Trennung von Verantwortlichkeiten zwischen den Services umgesetzt, auch wenn die Services eng miteinander gekoppelt sind. So arbeitet, wie in Listing 8 dargestellt, der `TimerService` direkt mit anderen Services wie dem `ProgressService` zusammen. Diese werden über deren Singleton-Instanzen initialisiert und über ihre öffentlichen Methoden angesprochen. Durch dieses Vorgehen entstehen direkte Abhängigkeiten zwischen den Services, was eine enge Kopplung zur Folge hat. Änderungen an einem Service können somit durchaus Auswirkungen auf andere Komponenten der App haben. Ein weiteres Beispiel für die enge Kopplung ist in der Methode `handleTaskCompletionLogic()`. Hier ruft der `TimerService` asynchron Funktionen wie `await _progressService.addXP()` auf und ist dabei direkt auf die Verfügbarkeit des `ProgressService` angewiesen. Falls im `TaskService` ein Fehler auftreten sollte, kann dies die Timer-Logik beeinträchtigen, da eine direkte Abhängigkeit besteht. Diese Art der Programmierung sorgt zwar für eine straightforward Implementierung, macht aber die App anfällig für Fehlerfortpflanzung zwischen den Services. Die Kommunikation erfolgt über direkte Methodenaufrufe und gemeinsame Instanzen. Direkte Zugriffe auf andere Services sind Bestandteil der Architektur. Das sorgt für eine eng gekoppelte, aber funktionale serviceorientierte Struktur.

Die MindSpark-App verwendet nicht nur instanzbasierte Kommunikation, sondern auch klar definierte

Schnittstellen, auch Service-Contracts genannt, um eine flexible und unabhängige Verbindung zwischen den Komponenten sicherzustellen (Chaves & De Freitas, 2019). Die klar definierten Schnittstellen ermöglichen eine zuverlässige Kommunikation zwischen den verschiedenen Diensten. In diesem Kontext dienen Service-Contracts als formale Vereinbarungen. Sie definieren die Rahmenbedingungen für die Kommunikation, indem sie u. a. die Input-/Output-Parameter und Datenstrukturen spezifizieren. Anstelle eines unstrukturierten Datenaustauschs kommen definierte Klassen zum Einsatz. Diese Klassen kapseln sowohl Eingaben als auch Ausgaben und gewährleisten Klarheit und Nachvollziehbarkeit (Chaves & De Freitas, 2019). Wie in Listing 9 dargestellt, implementiert der `CreateTaskService` zwei zentrale Datenklassen. Die erste dieser Datenklassen ist die `CreateTaskRequest`. Diese Klasse erhält alle erforderlichen Eingabefelder, etwa Titel und Prioritäten, sowie optionale Angaben. Die zweite Klasse `TaskResult` regelt die Rückgabe, entweder mit einem erfolgreichen Ergebnis und der erstellten Aufgabe oder mit einer Fehlermeldung (Zeile 22 & 23). Diese Struktur trägt nicht nur zur besseren Verständlichkeit bei, sondern erleichtert auch das frühzeitige Erkennen von Fehlern. Die Service-Contracts bieten den Vorteil der Typsicherheit. Durch den Einsatz stark typisierter Klassen können Fehler bereits zur Compile-Zeit identifiziert werden.

```
1  class CreateTaskRequest {
2      final String title;
3      final String? description;
4      final String priority;
5      //...
6
7        CreateTaskRequest({
8          required this.title,
9          this.description,
10         required this.priority,
11         //...
12       });
13 }
14
15 class TaskResult {
16     final bool isSuccess;
17     final TaskModel? task;
18     final String? message;
19     final String? error;
20
21     TaskResult.success(this.task, [this.message]): isSuccess = true, error = null;
22     TaskResult.error(this.error) : isSuccess = false, task = null, message = null;
```

```
}
```

Listing 9: Service Contracts

Während Service-Contracts bereits strukturierte Kommunikation ermöglichen, geht das Design hinter MindSpark darüber hinaus. Autonome Dienste zeichnen sich dadurch aus, dass sie ihre Geschäftslogik eigenständig verwalten und Entscheidungen über interne Prozesse treffen können. Dies zeigt sich beim `AuthService`, welcher eigenständig die Authentifizierungslogik verwaltet und autonom entscheidet, welche Folgeaktionen nach einer erfolgreichen Anmeldung ausgeführt werden, wie etwa die Erstellung des Benutzerprofils. Andere Services müssen diese internen Entscheidungen nicht kennen. Dies ermöglicht es, diesen Dienst flexibel weiterzuentwickeln, ohne dass an anderer Stelle Anpassungen erforderlich werden. Dies gewährleistet, dass der Code sowohl modular als auch verständlich und wartbar bleibt (Blal et al., 2025). Außerdem stellt der `AuthService` nützliche Utility-Methoden bereit, wie `isLoggedIn` und `userId`, die von anderen Services verwendet werden können. Diese Getter integrieren die Authentifizierungslogik und bieten eine benutzerfreundliche Schnittstelle für andere Services. Der `AuthService` agiert als *Single Point of Truth* für den Anmeldestatus. Dadurch werden Widersprüche vermieden und es wird sichergestellt, dass alle Services dieselbe Authentifizierungslogik anwenden.

```dart
class AuthService {
  // Wiederverndbare utility-Methoden
  bool get isLoggedIn => currentUser != null;
  String? get userId => currentUser?.id;

  // Autonomer Service mit eigenständiger Dateneinheit
  Future<AuthResult> signIn({
    required String email,
    required String password,
  }) async {
    try {
      // Service verwaltet seine eigene Authentifizierungslogik
      final response = await supabase.auth.signInWithPassword(
        email: email,
        password: password,
      );

      if (response.user == null) {
        // Autonome Entscheidung über zusätzliche Aktionen
        return AuthResult.error('Login hat nicht geklappt');
      }
      return AuthResult.success(response.user!);
    } catch (e) {
```

```
24        return AuthResult.error('Login Fehler: $e');
25      }
26    }
27  }
```

Listing 10: Autonomie und Wiederverwendbarkeit

Die konsequente Implementierung der SOA in der MindSpark-App verdeutlicht, dass durch die gezielte Anwendung von Service-Kapselung, loser Kopplung, Service-Contracts und autonomen Services eine wartbare sowie erweiterbare Anwendungsarchitektur entsteht. Diese Herangehensweise ermöglicht es, komplexe Geschäftslogik in überschaubare und unabhängige Einheiten zu gliedern. Damit bildet sie die Grundlage für eine skalierbare und zukunftsfähige Softwarearchitektur.

4.1.2 Datenmodellierung & ER-Diagramm

Für die dargestellte Architektur ist eine entsprechend strukturierte Datenbasis erforderlich, die die komplexe Geschäftslogik und deren Beziehungen abbildet. Das zugrundeliegende Datenmodell (siehe Abbildung 2) muss dabei sowohl technische Anforderungen als auch die spezifischen Bedürfnisse der neurodivergenten Zielgruppe berücksichtigen. Es strukturiert die wesentlichen Geschäftsobjekte und deren Beziehung zueinander, um eine konsistente und erweiterbare Datenbasis zu schaffen. Eine saubere Datenstruktur ist für neurodivergente Zielgruppen von besonderer Bedeutung, da sie Nachvollziehbarkeit und Konsistenz gewährleisten. Die Erfassung von Stimmungsdaten, Fortschrittsinformationen etc. unterstützt die spezifischen Bedürfnisse der Zielgruppen nach Übersichtlichkeit und individueller Anpassbarkeit. Um diese Anforderungen zu erfüllen, wurden neun Tabellen identifiziert und modelliert, die alle wesentlichen Aspekte der App abdecken.

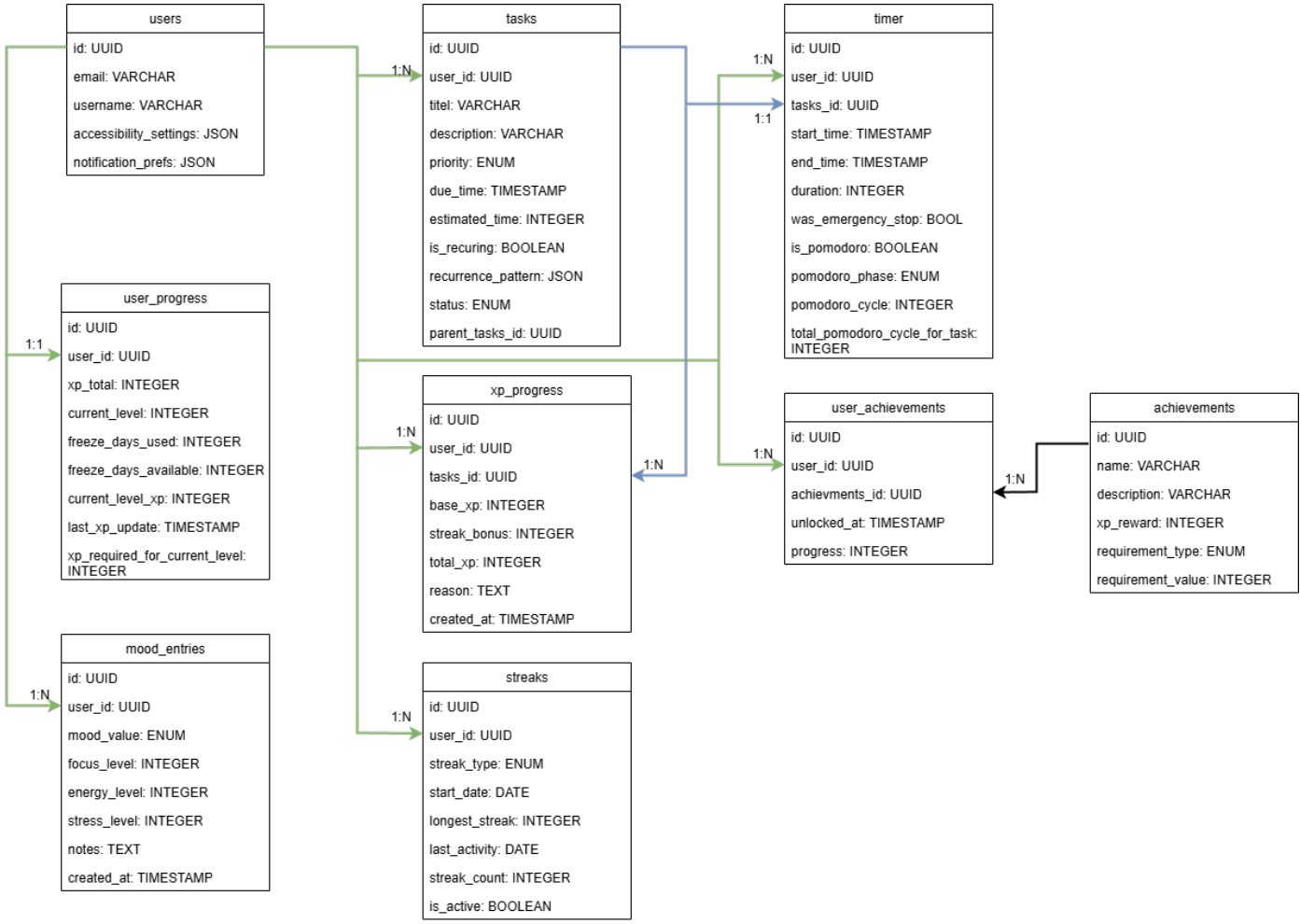

Abbildung 2: ER-Diagramm

Die Tabelle *users* speichert grundlegende Informationen über registrierte Personen, darunter eine eindeutige ID, die E-Mail-Adresse sowie Metadaten wie Nutzertyp und Registrierungszeitpunkt. Das Passwort wird aus Datenschutzgründen nicht in die Datenbank gespeichert. Sie dient als Ausgangspunkt für fast alle anderen Datenbank-Tabellen, da viele Tabellen über die `user_id` mit ihr verbunden sind. Dadurch ist sichergestellt, dass sämtliche Inhalte eindeutig einer bestimmten Person zugeordnet werden können.

Die *task*-Tabelle verwaltet die zu erledigenden Aufgaben. Jede Aufgabe enthält Angaben wie Titel, Beschreibung, etc. Zusätzlich gibt es Felder für optionale Informationen wie Fälligkeitszeit oder verknüpfte Unteraufgaben. Die Aufgaben sind über die `user_id` mit dem jeweiligen Konto verbunden. Durch den zusätzlichen Fremdschlüssel `parent_task_id` können Aufgaben zudem hierarchisch gegliedert werden, was die Umsetzung von Teilaufgaben-Strukturen ermöglicht.

Die Tabelle *user_progress* speichert pro Benutzer die zentrale Fortschrittsanzeige des Gamification-Systems. Sie enthält die Gesamtsumme der XP-Punkte, das aktuelle Level sowie Informationen zum Streak-System mit verfügbaren und bereits genutzten Freeze-Days. Zudem werden levelbezogene Daten wie die für das nächste Level benötigten XP und der Zeitpunkt der letzten XP-Aktualisierung ge-

speichert. Ergänzend dazu dokumentiert die *xp_progress* alle einzelnen XP-Gewinne mit Zeitstempel, Begründung und optionaler Verknüpfung zu einer spezifischen Aufgabe, wodurch eine detaillierte Nachverfolgung des Fortschritts ermöglicht wird. Die *timer*-Tabelle erfasst alle Fokus- und Arbeitssessions der Benutzer. Jeder Eintrag enthält Start- und Endzeit, die Gesamtdauer etc. Besondere Felder wie `was_emergency_stop` und `is_pomodoro` dokumentieren die Art der Session-Beendigung und den verwendeten Timer-Modus. Zusätzliche Informationen über Pomodoro-Phasen und Zyklen werden in entsprechenden Feldern gespeichert, um verschiedene Zeitmanagement-Techniken zu unterstützen.

Eine weitere Tabelle ist die *mood_entries*, die das emotionale Befinden der Nutzer über die Zeit hinweg speichert. Sie erlaubt tägliche Einträge mit Stimmungsauswahl, optionalem Kommentar und Zeitstempel. Auch diese Einträge sind an die `user_id` gebunden und bieten eine Möglichkeit zur Reflexion und individuellen Auswertungen.

Die *streaks*-Tabelle dokumentiert verschiedene Arten von Aktivitätenserien für jeden Benutzer. Sie enthält den Streak-Typ, Start- und Enddatum, die aktuelle und längste Streak-Serie etc. Das Achievement-System wird durch zwei Tabellen realisiert. Die *achievements*-Tabelle definiert verfügbare Errungenschaften mit Namen, Beschreibung, etc. Die Verknüpfungstabelle *user_achievements* dokumentiert, welche Benutzer welche Achievements zu welchem Zeitpunkt freigeschaltet haben und speichert den aktuellen Fortschritt `progress` bei mehrstufigen Errungenschaften. Die beiden Tabellen haben eine m:n-Beziehung.

Das Modell basiert auf klaren Beziehungen zwischen den Tabellen, wie beispielsweise einer 1:n-Beziehung zwischen Nutzer und Aufgaben, Timer oder Fortschrittsdaten. Diese Struktur ermöglicht es der App, flexibel, nachvollziehbar und konsistent mit den Daten zu arbeiten. Für neurodivergente Nutzer, die besonders Wert auf Übersichtlichkeit und Transparenz legen, stellt dieses Modell eine solide Grundlage dar. Nutzer erfahren einen Fortschritt nicht nur durch abgeschlossene Aufgaben, sondern auch durch die Streaks und Errungenschaften. Die Integration von JSON-Feldern, wie z. B. bei `accessibility_settings` oder `recurrence_pattern`, verleiht dem Modell zusätzliche Flexibilität. Nutzer haben die Möglichkeit, eigene Vorlieben oder komplexe Wiederholungsmuster zu definieren, ohne dass separate Tabellen hierfür erforderlich wären. Die Datenhaltung wurde in einer Datenbank auf Supabase realisiert. Supabase verwendet PostgreSQL als Datenbankbasis und ermöglicht somit die direkte Umsetzung des ER-Modells in tabellarischer Form. Die Beziehungen zwischen den Tabellen wurden mithilfe von Fremdschlüsseln explizit definiert und durch Constraints abgesichert. Dies schließt Mehrdeutigkeiten aus und gewährleistet die Datenintegrität. Zudem wurden Row-Level-Security-Polices (RLS) implementiert. Diese stellen sicher, dass Nutzer ausschließlich auf ihre eigenen Daten zugreifen können. Das ist ein wesentlicher Aspekt im Umgang mit sensiblen persönlichen Informationen. Damit legt die Datenstruktur nicht nur

die Grundlage für die aktuellen Funktionen, sondern erlaubt auch eine langfristige Weiterentwicklung der App, technisch stabil und inhaltlich anpassbar an individuellen Bedürfnissen.

4.2 UML-Diagramme

Um komplexe Softwarearchitekturen sichtbar zu machen, eignen sich UML-Diagramme, die eine klare Darstellung der verschiedenen Abläufe in der App bieten. Um eine der wichtigsten Funktionen und deren Zusammenhang darzustellen, wurden für die MindSpark-App zwei unterschiedliche Diagrammarten ausgewählt. Das Aktivitätsdiagramm konzentriert sich auf den Ablauf der Stimmungserfassung, während das Sequenzdiagramm die technischen Vorgänge des Belohnungssystems detailliert veranschaulicht. Beide Bereiche sind zentrale Bestandteile der App und spiegeln verschiedene Aspekte der Nutzererfahrung wider.

4.2.1 Aktivitätsdiagramm

Das Aktivitätsdiagramm zeigt den systematischen Ablauf der Stimmungserfassung in der Mindspark-App. Es zeigt die Abfolge von Aktionen und stellt die drei Ebenen dar, wie Benutzer, App und Backend zusammenarbeiten, um einen Stimmungseintrag zu ermöglichen.

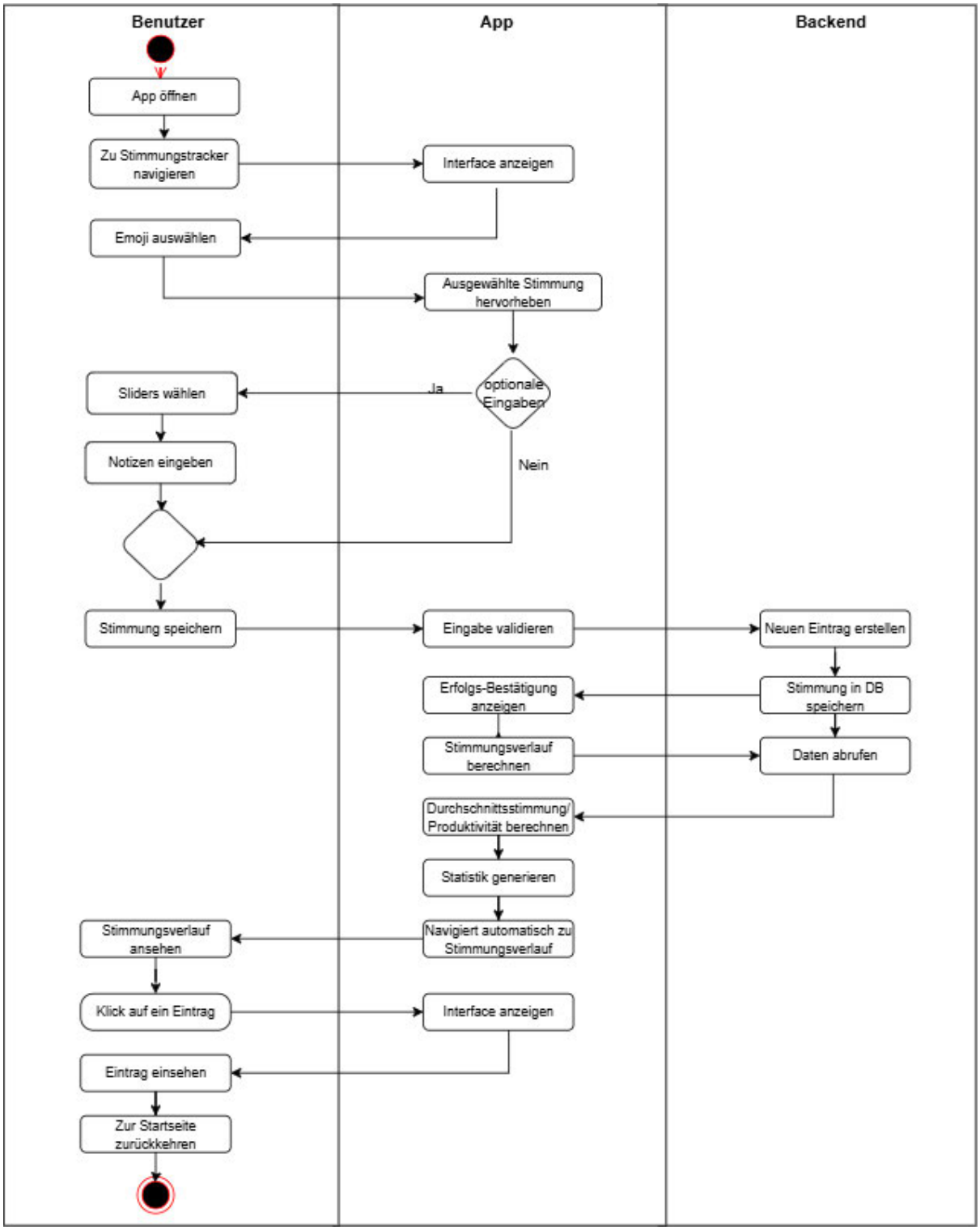

Abbildung 3: Aktivitätsdiagramm - Stimmungstracker

Der Prozess beginnt damit, dass der Benutzer die App öffnet. Daraufhin navigiert der Benutzer direkt zum *Stimmungstracker*. Die App lädt das Interface hoch, sodass der Benutzer die Seite sofort erkennt. Nun beginnt der Hauptprozess mit dem Schritt *Emoji auswählen*. Die App hebt die ausgewählte Stimmung hervor, indem der Hintergrund des Emojis farblich hervorgehoben wird. Der Benutzer kann die Stimmungserfassung beenden, indem er auf *Stimmung speichern* klickt. Optional kann er zusätzliche

Parameter wie Energielevel, Fokus und Stresslevel in der Form eines Sliders auswählen sowie eine Notiz hinterlassen für die Dokumentation. Die Eingabephase endet ebenfalls mit einem Klick auf *Stimmung speichern*. Die App führt anschließend die Eingabevalidierung durch und übergibt die Daten an das Backend, das daraufhin einen neuen Eintrag in der Datenbank erstellt und speichert. Die App zeigt im Anschluss eine Erfolgsbestätigung, dass die Stimmung erfolgreich gespeichert worden ist. Parallel beginnt die Datenverarbeitung und es werden die Durchschnittsstimmung sowie die Produktivität berechnet. Diese Berechnungen werden für die automatische Statistikgenerierung verwendet. Sobald die Daten verarbeitet wurden sind und der Benutzer auf *Stimmung speichern* geklickt hat, navigiert die App automatisch zum Stimmungsverlauf und zeigt dem Benutzer den aktuellen Verlauf der letzten sieben oder dreißig Tage. Der Benutzer hat noch die Möglichkeit, bei einem Klick auf einen Eintrag die Details einzusehen. Das Aktivitätsdiagramm endet damit, dass der Benutzer zurück zur Startseite kehrt. Das dargestellte Diagramm zeigt eine Hauptfunktion der App mit strategischen Entscheidungspunkten. Diese sorgen für eine unkomplizierte Benutzerführung ohne kognitive Überlastung für den Benutzer.

4.2.2 Sequenzdiagramm

Das Sequenzdiagramm zeigt die ungefähre zeitliche Abfolge der Freischaltung von Achievements in der Mindsapark-App. Es zeigt, wie vier Systemkomponenten (*TimerService, ProgressService, AchievementManager* und die Datenbank) zusammenarbeiten, angefangen bei der Nutzeraktion bis hin zur Datenspeicherung.

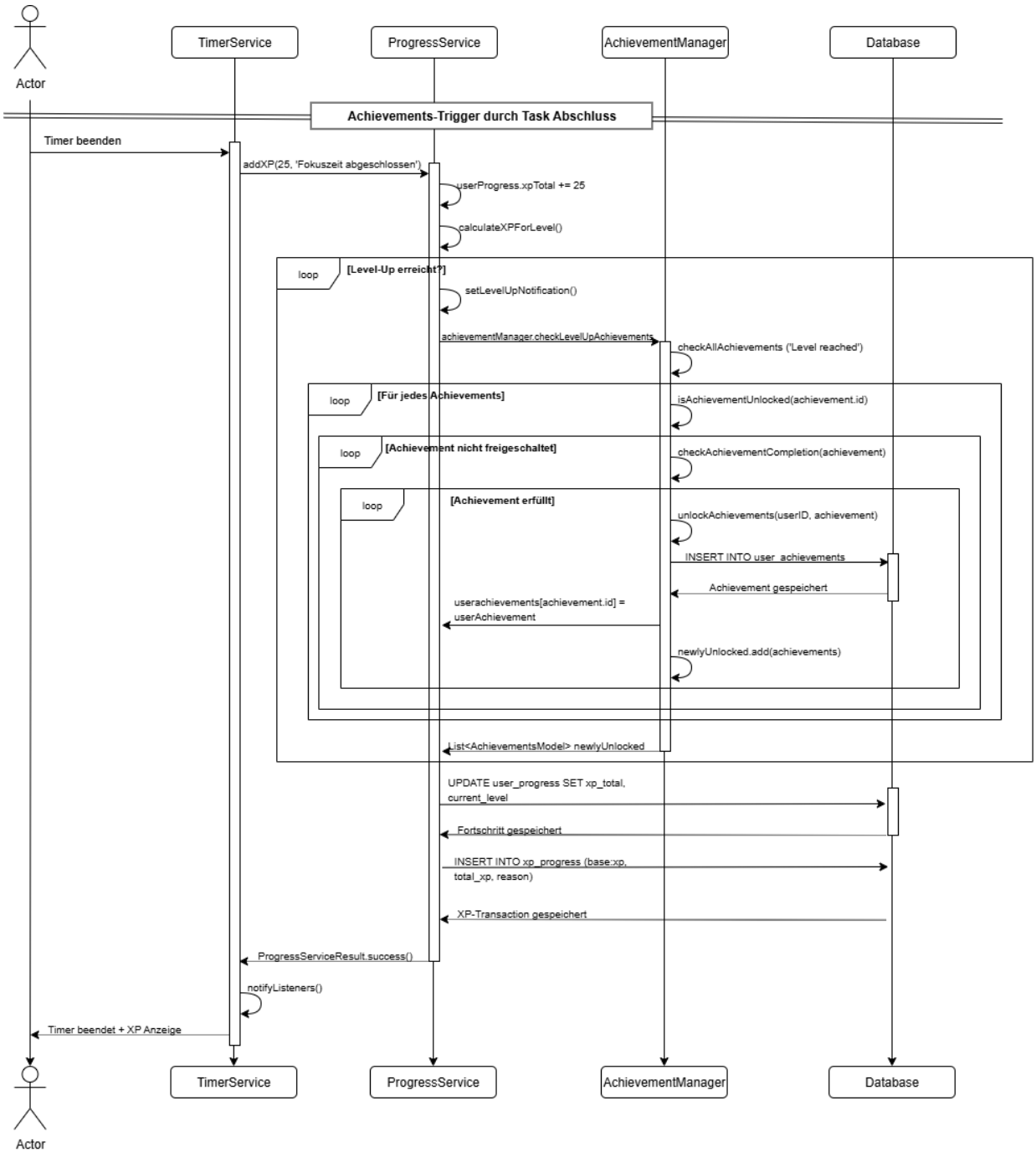

Abbildung 4: Sequenzdiagramm Achievement-Trigger

Der Prozess startet, sobald der Benutzer den Timer beendet oder seine Session abgeschlossen hat. Das Diagramm stellt sowohl die Reihenfolge dieser Abläufe dar als auch die Verzweigung, die dabei beachtet werden muss. Zuerst ruft der `TimerService` die Funktion `addXP` im `ProgressService` auf, der die vorgesehene XP sowie die Art, für was XP vergeben wird, übergibt. Der `ProgressService` macht daraufhin Berechnungen, wie die Addition der Punkte zum bereits bestehenden Gesamtwert der XP, und berechnet das neue Level mit der `calculateXPForLevel()`-Funktion. Hier taucht die erste Verzweigung auf. Es wird überprüft, ob der Nutzer ein neues Level erreicht hat. Wenn dem so ist, werden zusätzliche

Schritte eingeleitet. Der `ProgressService` schickt eine Benachrichtigung über das Level-Up und startet den Achievement-Check durch den Funktionsaufruf im `AchievementManager`, der alle Achievements prüft. Die App kontrolliert zuerst, ob ein Achievement schon freigeschaltet ist oder nicht, und innerhalb dieser Schleife gibt es wieder eine Verzweigung. Wenn ein Achievement noch nicht freigeschaltet worden ist, prüft die `checkAchievementCompletion()`, ob alle nötigen Voraussetzungen erfüllt sind. Wenn dem so ist, erfolgt die Prüfung des Typs `RequirementType.levelReached`. Sobald ein Achievement freigeschaltet wird, wird dieses in die Datenbank-Tabelle *user_achievement* hinzugefügt und gespeichert. Der lokale Cache wird zugleich mit aktualisiert (`userAchievements[achievement.id = userAchievement]`), indem das `userAchievement` unter der Achievement-ID in der Map gespeichert wird. Das neue Achievement wird in die Liste `newlyUnlocked` hinzugefügt und aktualisiert. Der `ProgressService` aktualisiert die XP- und Level-Werte in der Datenbank und erstellt einen neuen Eintrag in der XP-Transaction. Der `TimerService` erhält vom `ProgressService` eine erfolgreiche Benachrichtigung und führt den `notifyListener()` aus, um Änderungen in der UI sofort sichtbar für den Benutzer zu machen. Zum Schluss wird eine Benachrichtigung an den Benutzer rausgeschickt mit der XP-Anzeige und dem Level-Up.

Des Weiteren veranschaulicht das Sequenzdiagramm die Funktionsweise des *Perfect-Day-Achievements*. Im Gegensatz zu einem allgemeinen Achievement-Check konzentriert sich dieses Diagramm auf die Logik der Freischaltung des perfekten Tages. Dieses Achievement wird nur freigeschaltet, wenn alle Aufgaben für einen Tag erledigt worden sind.

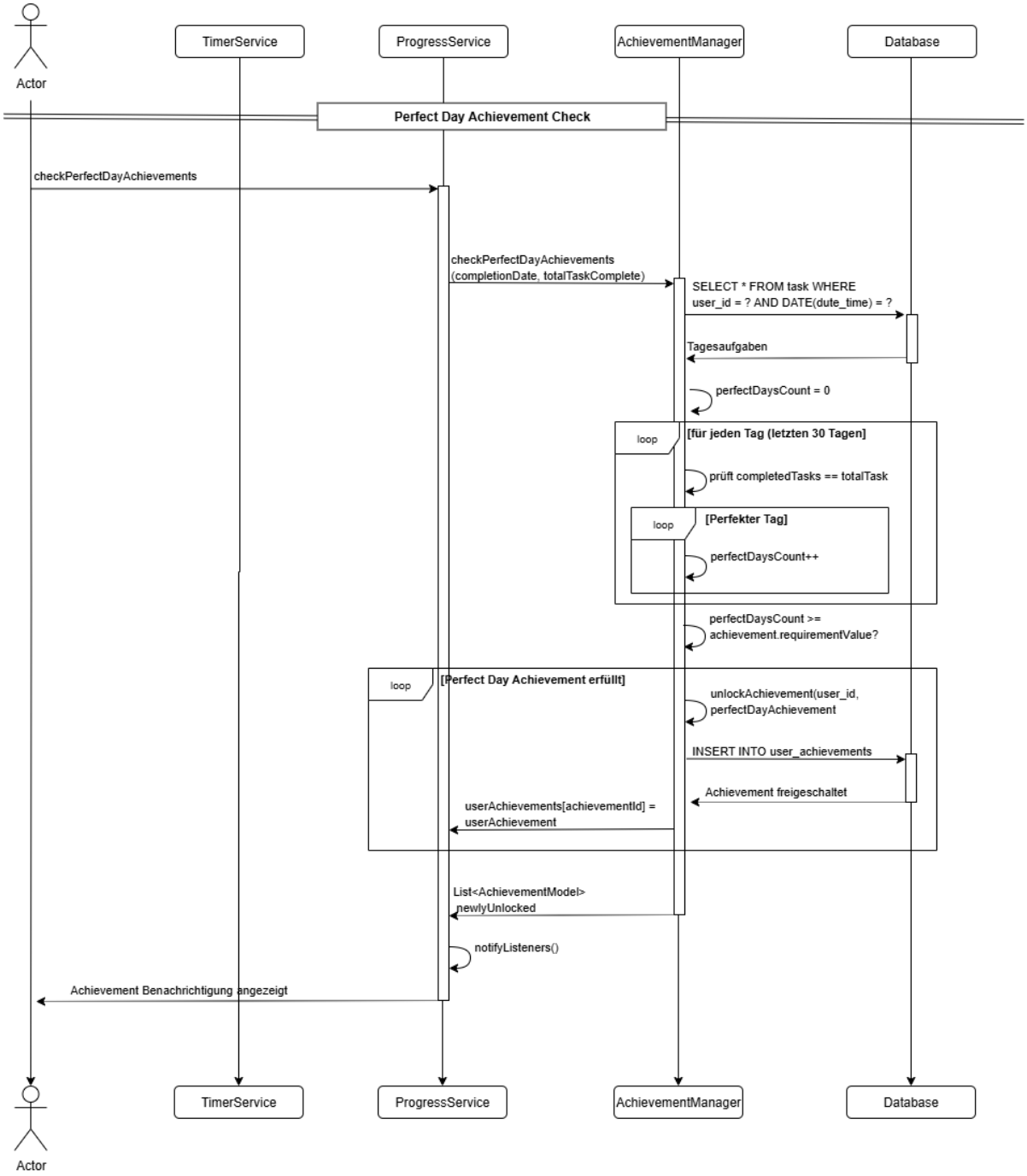

Abbildung 5: Sequenzdiagramm Perfect Day Achievement

Der Prozess beginnt mit der Funktion `checkPerfectDayAchievements` durch den Benutzer, der seine Session beendet hat. Diese Funktion wird vom `ProgressService` direkt an den `Achievementmanager` weitergeleitet, wobei wichtige Parameter wie das Abschlussdatum (`completionDate`) und die Anzahl der erledigten Aufgaben (`totalTaskCompleted`) übergeben werden. Der `AchievementManager` führt zunächst eine Datenbankabfrage mittels eines `SELECTs` durch. Diese SQL-Abfrage beschafft alle Aufgaben eines Benutzers, die an einem bestimmten Datum fällig sind. Die Datenbank reagiert darauf und liefert die relevanten Tagesaufgaben zurück. Nach der Abfrage zählt der `AchievementManager` einen Zäh-

ler hoch mit dem Wert `perfectDaysCount = 0`. Dieser Zähler verfolgt die Anzahl der perfekten Tage. Nun iteriert der `AchievementManager` über einen Zeitraum von 30 Tagen und innerhalb dieser Hauptschleife prüft er für jeden Tag, ob alle Aufgaben abgeschlossen sind (`completedTask == totalTask`). Wenn dem so ist, wird der `perfectDaysCount`-Zähler um eins erhöht (`perfectDaysCount ++`). Nach der Beendigung der Schleife vergleicht der `AchievementManager` den gezählten Wert für perfekte Tage mit dem erforderlichen Ziel des Achievements (`perfectDaysCount` $\geq$ `achievement.requirementValue?`). Diese Bedingung entscheidet über die Freischaltung des *Perfect-Day-Achievements*. Wenn die Voraussetzungen erfüllt sind, wird eine weitere bedingte Ausführung durchgeführt, in der die Funktion `unlockAchievement` aufgerufen wird. Diese Funktion benötigt zwei Parameter: `user_id` und das `perfectDaysAchievement`-Objekt. Dieses Achievement wird mit einem `INSERT` in die Datenbank hinzugefügt und gespeichert. Die Datenbank bestätigt die erfolgreiche Speicherung, daraufhin aktualisiert sie den lokalen Cache durch die Zuweisung `userAchievements[achievementId] = userAchievement`. Der `AchievementManager` erstellt eine Liste der neu freigeschalteten Achievements und gibt diese an den `ProgressService` zurück. Der `ProgressService` führt den `notifyListener()` aus, um alle UI-Komponenten über die Änderungen zu informieren. Diese Herangehensweise ermöglicht es der App, Benutzer für ihre Produktivität über einen längeren Zeitraum hinweg zu belohnen. Dies kann sehr motivierend für neurodivergente Benutzer sein, da es ihnen hilft, langfristige Erfolge sichtbar zu machen.

4.3 UI/UX-Design

In diesem Kapitel handelt es sich um die Benutzeroberflächen und Entscheidungen, die dafür sorgen, dass die App zugänglich, einfach bedienbar und verständlich bleibt. Im Fokus stehen die Bereiche, die das Nutzererlebnis definieren. Dazu gehören das Farbkonzept, das dafür sorgt, dass Inhalte gut sichtbar und lesbar sind, die Organisation der Navigation und spezielle Elemente, die gezielt auf die Bedürfnisse neurodivergenter Nutzer abgestimmt sind und wie diese dazu beitragen, eine benutzerfreundliche App zu gestalten.

4.3.1 Farbkonzept

Bei der Farbgestaltung der MindSpark-App zeigt sich relativ schnell, dass es sich nicht nur um das Ästhetische handelt. Die Entwicklung eines barrierefreien Farbkonzepts bildet die Grundlage für die Zugänglichkeit, die Menschen mit unterschiedlichen visuellen Fähigkeiten miteinschließt. Das Farbkonzept orientiert sich an den Web Content Accessibility Guidelines (WCAG). Dort sind Richtlinien festgelegt,

wie z. B. die Festlegung der Größe einer Schrift. Für normalen Text wird ein Verhältnis von 4,5 : 1 verlangt, während für größere Schrift ein Wert von 3 : 1 genügt (W3C, 2025). Diese Verhältnisse bestimmen, wie gut Inhalte lesbar sind, und diese Vermutungen lassen sich in einer Studie zeigen. Denn genau solche Anpassungen verbessern die Zugänglichkeiten von Interfaces (Reinecke et al., 2018). Die eigentliche Schwierigkeit liegt jedoch darin, etwas in der Mitte zu finden, das einerseits das Farbschema ästhetisch wirken lässt und andererseits funktional zugänglich bleiben soll. Das Konzept gliedert sich daher in drei Modi ein. Ein Standard-Hellmodus für normale Lichtverhältnisse. Einen Dunkelmodus zur Entlastung der Augen in dunkleren Umgebungen und einen Kontrastmodus für Nutzer mit deutlichen Sehbeeinträchtigungen (siehe Abbildung 6). Diese drei Modi bieten genau die Flexibilität, die eine barrierefreie App benötigt, da laut einer Studie rund 8 % der Männer und 0,5 % der Frauen von einer Farbsehschwäche betroffen sind (Birch, 2012). Der Dunkelmodus ist so aufgebaut, dass zentrale Elemente in helleren Farben dargestellt werden. Die Hauptfarbe, ein helles Lila (0xFF818CF8), hebt sich klar vom dunklen Hintergrund (0xFF1F2937) ab. Dieser Kontrast, hell auf dunkel oder dunkel auf hell, hat einen Einfluss auf die Lesegeschwindigkeit eines Nutzers und gleichzeitig bleibt die Sichtbarkeit auch bei gedimmten Bildschirmen gewährleistet (Pedersen et al., 2020).

Trotz dass viele Menschen an einer Rot-Grün-Schwäche leiden, wurde für die Darstellung der Prioritätsstufen eine Rot-Gelb-Grün-Logik gewählt. Einer der Gründe für die Auswahl ist dessen Vertrautheit. Das Muster ist vielen Menschen geläufig und wird sofort verstanden, ohne große Überlegungen. Jedoch wurde eine Lösung gefunden, um auch Menschen mit einer Rot-Grün-Schwäche das Erkennen zu ermöglichen. Farben alleine können leicht missverstanden werden, aber wenn sie mit Text kombiniert werden, wird die Information eindeutiger. Zudem hilft der Text nicht nur neurodivergenten Menschen und sehbeeinträchtigten Menschen, sondern auch diesen, deren Umgebung nicht profitabel ist. Z. B. bei Sonnenlicht. Im Kontrastmodus werden die Farben fast komplett weggelassen und es bleiben nur Schwarz-Weiß-Kontraste. Laut Stone (2016) sollen Farben nie das einzige Mittel sein, um Informationen zu vermitteln. Die technische Umsetzung erfolgt über eine zentrale `AppColors`-Klasse, die als Single Source of Truth (SSOT) fast alle Farbdefinitionen bereitstellt. Mithilfe der Methode `createTheme` wird ein vollständiges Material-Design-Theme erzeugt, das sich an den Nutzereinstellungen orientiert.

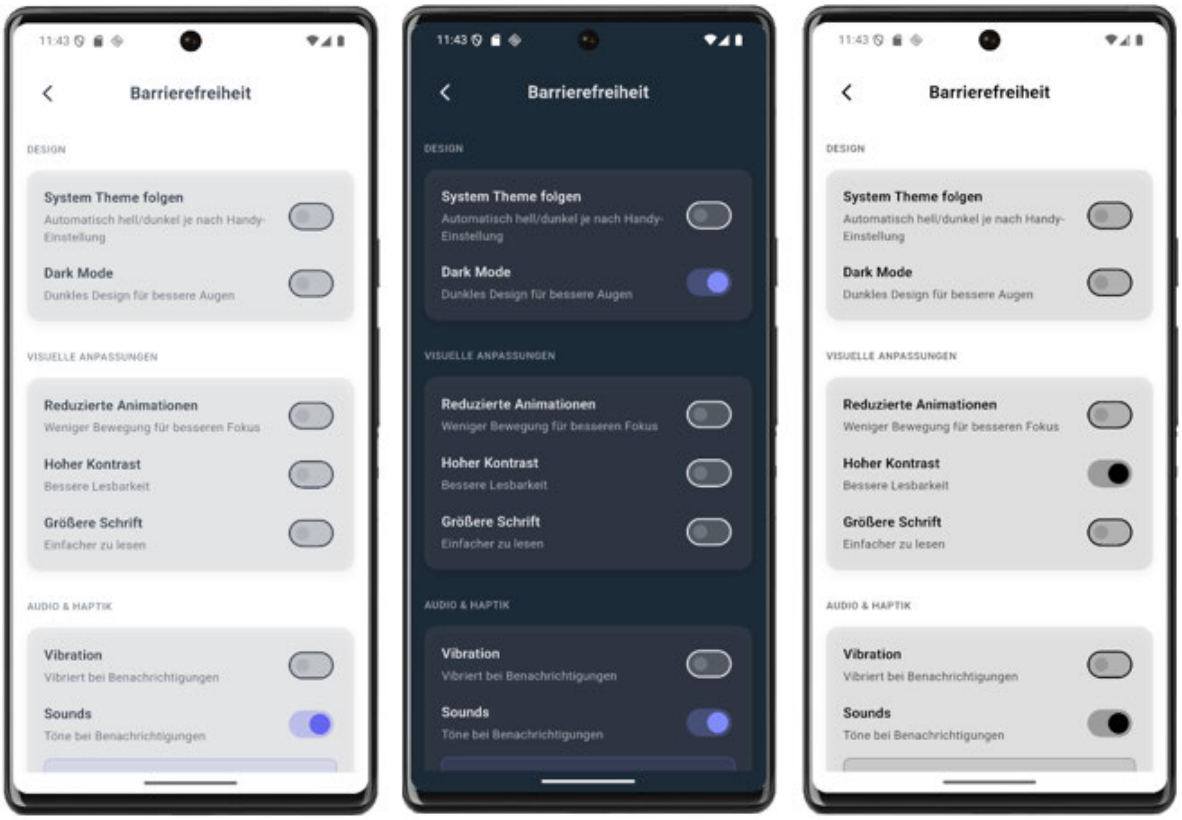

Abbildung 6: Modi

4.3.2 Navigation und Informationsarchitektur

Die Navigation durch die MindSpark-App erfolgt über eine Bottom-Navigation als primäres Navigationselement. Bei einer einhändigen Smartphone-Nutzung muss die Erreichbarkeit mit dem primären Interaktionswerkzeug, dem Daumen, gewährleistet sein (Babich, 2016). Die Platzierung für mobile Anwendungen muss optimal erfolgen, denn oft haben neurodivergente Menschen, vor allem Menschen mit ASS, motorische Koordinationsschwierigkeiten. In der App sind genau vier Hauptnavigationspunkte (Home, Aufgaben, Timer und Stimmung) implementiert, um kognitive Überlastung zu vermeiden (siehe Abbildung 7). Die Navigation bleibt konstant präsent, um zu vermeiden, dass Nutzer aktiv nach Navigationsmöglichkeiten suchen. An den Stellen, wo eine verschachtelte Navigation erfolgt, wie z. B. vom Stimmungs-Interface zum Stimmungsverlauf-Interface oder vom Timer-Interface zu den Timer-Einstellungen, ist ein Zurück-Button konsistent oben auf der linken Seite eines Bildschirms platziert und visuell gestaltet. In der Navigation hat jeder Tab ein Icon mit einem passenden Label. Zum Beispiel hat der Timer-Tab einen Wecker als Icon. In einer Studie zu neurodivergenzfreundlichem Design zeigt sich, dass besonders autistische Nutzer von dieser doppelten Kodierung profitieren, da sie unterschiedliche Informationsverarbeitungspräferenzen bedienen (Maaß & Rink, 2018).

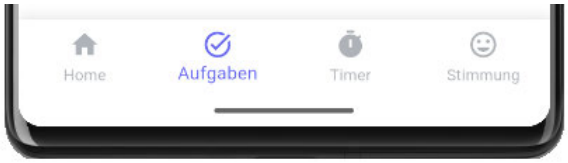

Abbildung 7: Tab-Bar für die Navigation

Die Informationsarchitektur (IA) folgt dem Prinzip der flachen Hierarchie. Maximal drei Navigationsebenen trennen den Nutzer von jeder Funktion. Dadurch minimiert sich die Anzahl der Entscheidungspunkte, was für Nutzer mit Aufmerksamkeitsdefiziten vorteilhaft ist. Menschen mit ADHS navigieren oft assoziativ statt hierarchisch. So wurde die strikte Kategorisierung manchmal durchbrochen. So ist der Timer sowohl als eigenständiger Tab als auch kontextuell innerhalb einzelner Aufgaben erreichbar, d. h., wenn ein Nutzer in der Home-Oberfläche auf eine Aufgabe klickt, wird dieser unmittelbar zu dem Timer-Interface navigiert. Diese bewusste Redundanz mag aus klassischer IA-Perspektive unvorteilhaft erscheinen, entspricht aber dem nichtlinearen Denkmuster vieler neurodivergenter Menschen. Die Screens folgen einem konsistenten Layout. Jeder Screen folgt dem gleichen strukturellen Aufbau mit einem Header, dem Hauptcontent und zum Schluss der Tab-Bar. Die technische Umsetzung nutzt Flutter mit `routes`, was eine deklarative Navigation ermöglicht. Jede Route hat einen semantischen Namen (`/home, /task, /timer, /mood` etc.), der die Wartbarkeit erhöht und Deep-Linking ermöglicht.

4.3.3 Neurodivergenz-spezifische Gestaltungselemente

Ein zentraler Baustein von neurodivergenz-spezifischen Gestaltungselementen liegt in der sensorischen Anpassung. Die App verfügt über ein mehrstufiges Audio-Management, das zwischen verschiedenen Feedback-Modalitäten differenziert. Nutzer können gezielt zwischen Vibrations- und Audio-Feedback wählen oder diese deaktivieren. Die Implementierung integriert sich in die Android-Systemhierarchie. Wenn ein Nutzer ihr Smartphone auf Vibrationsmodus stellt, respektiert die App diese Einstellung unabhängig von den app-internen Präferenzen. D. h., wenn der Nutzer sein Smartphone auf Stumm stellt und in der App die Vibration oder den Sound aktiviert, ist das Ergebnis eine stumme Benachrichtigung. Zudem gestaltet sich die Schriftvergrößerung als sehr praktisch. Die Textgröße wird nicht nur erhöht, sondern auch proportional an die Touch-Target-Größen angepasst. Diese Methode gewährleistet, dass motorische Herausforderungen, die häufig neurodivergente Menschen, aber auch neurotypische Menschen betreffen können, kompensiert werden.

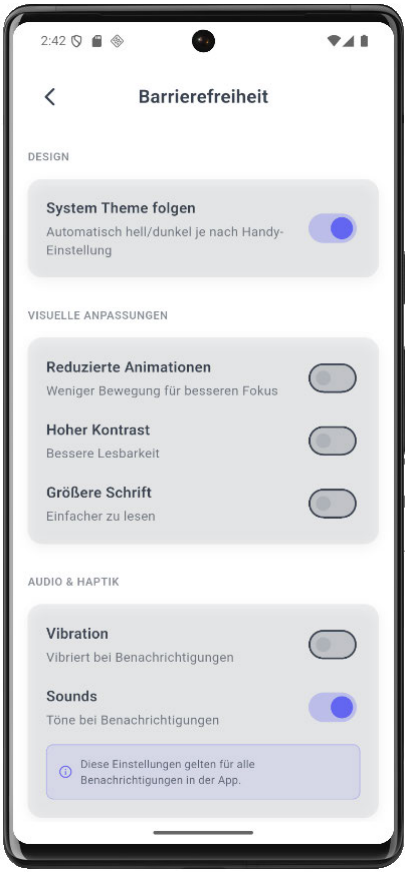

Abbildung 8: Barrierefreiheitseinstellungen

Die räumliche Konsistenz von Navigationselementen ist ein wichtiges Designprinzip der App. Der Floating Action Button (FAB) für das Hinzufügen neuer Aufgaben befindet sich an den Stellen, wo es benötigt wird, durchgängig in der rechten unteren Bildschirmecke. Bei allen verschachtelten Navigationsebenen befinden sich alle Zurück-Buttons an ihren festen Positionen, oben rechts in der Ecke eines Bildschirms (siehe Abbildung 8). Dieses Layout ist besonders für Menschen mit ASS wichtig, da unerwartete Änderungen in der Interface-Struktur oft zu Stress und Nervosität führen können. Der Notfall-Stop ist ein gut sichtbarer roter Button im Timer und ermöglicht das sofortige Beenden jeder Timer-Session ohne negative Konsequenzen.

4.4 Gamification–Konzept

Das Gamification-Konzept dieser App kombiniert verschiedene motivationspsychologische Ansätze, um langfristige Nutzung zu gewährleisten. Dabei stehen zwei zentrale Mechanismen im Fokus. Ein XP-System mit progressiver Level-Struktur und Belohnungsmechanismen durch Achievements. Die Gestaltung orientiert sich bewusst an den spezifischen Bedürfnissen von Personen mit ADHS und ASS.

4.4.1 XP-System und Level-Progression

Das Experience-Point-System (XP) stellt einen zentralen Baustein für eine gamifizierte Lernhilfe-App dar. Die grundlegende XP-Vergabe richtet sich hauptsächlich nach der Länge einer Aufgabe. Kurze Aufgaben unter 15 Minuten werden mit 10 XP belohnt, mittlere Aufgaben zwischen 15 und 40 Minuten erhalten 20 XP und für längere Aufgaben über 40 Minuten erhält der Nutzer 40 XP. Diese Staffelung erscheint zunächst linear, jedoch sorgen die längeren Aufgaben dafür, dass die Nutzer mehr mentale Anstrengungen für eine Aufgabe leisten. Gerade bei Menschen mit ADHS zeigen sich Schwierigkeiten, die Konzentration über einen längeren Zeitraum aufrechtzuhalten. Diese zusätzliche Belastung wird durch die höhere XP-Belohnung anerkannt. Ziel ist es dabei nicht, überzogene Erwartungen aufzubauen, sondern kleine Anreize zu setzen, die machbar wirken. Zudem brechen viele Menschen ihre Aufgaben, die einen hohen Zeitaufwand erfordern, eher ab, wenn keine Rückmeldungen bezüglich ihres Fortschritts erfolgen. Durch die höhere XP-Vergabe wird das Durchhalten belohnt und kann so Motivation zurückgeben. Es geht dementsprechend nicht nur um die investierte Zeit, sondern auch um die kognitive Energie, die in diesen Prozess einfließt.

Beim Levelsystem wurde bewusst eine progressive Struktur gewählt, die vom linearen Wachstum abweicht. In den ersten fünf Leveln benötigt der Nutzer jeweils 100 XP, um das nächste Level zu erreichen. Die geringe XP der ersten fünf Level sorgt für einen schnellen Fortschritt. Es verhindert, dass Nutzer schnell das Interesse an der App verlieren. Ab dem sechsten Level steigt der Bedarf auf 180 XP und ab dem elften Level steigt der Bedarf sogar auf 250 XP pro Level an. Der zugrundeliegende Gedanke ist, dass zunächst die kleinen Erfolgserlebnisse eine entscheidende Rolle spielen. Im späteren Zeitraum braucht der Nutzer mehr Herausforderungen, sonst wirkt die App zu eintönig.

Für Nutzer wurde besonders Wert auf ein sofortiges Feedback gelegt. XP werden direkt nach dem Abschluss einer Aufgabe gutgeschrieben, ohne spürbare Verzögerungen. Die Rückmeldung erfolgt in Form von animierten Pop-up-Benachrichtigungen, wodurch der Nutzer unmittelbar eine positive Verstärkung erfährt. Die App verzichtet auf versteckte Mechanismen oder überraschende Änderungen. So sind Level- und XP-Verluste grundsätzlich ausgeschlossen, was die Frustration eines Nutzers minimiert. Nutzer haben jederzeit die Möglichkeit, ihren aktuellen Fortschritt einzusehen und genau nachvollziehen, wie viele XP bis zum nächsten Level erforderlich sind und wie sein XP Verlauf ist (siehe Abbildung 9). Diese Möglichkeit bietet nicht nur Transparent, sondern fördert auch ein Gefühl von Selbstwirksamkeit

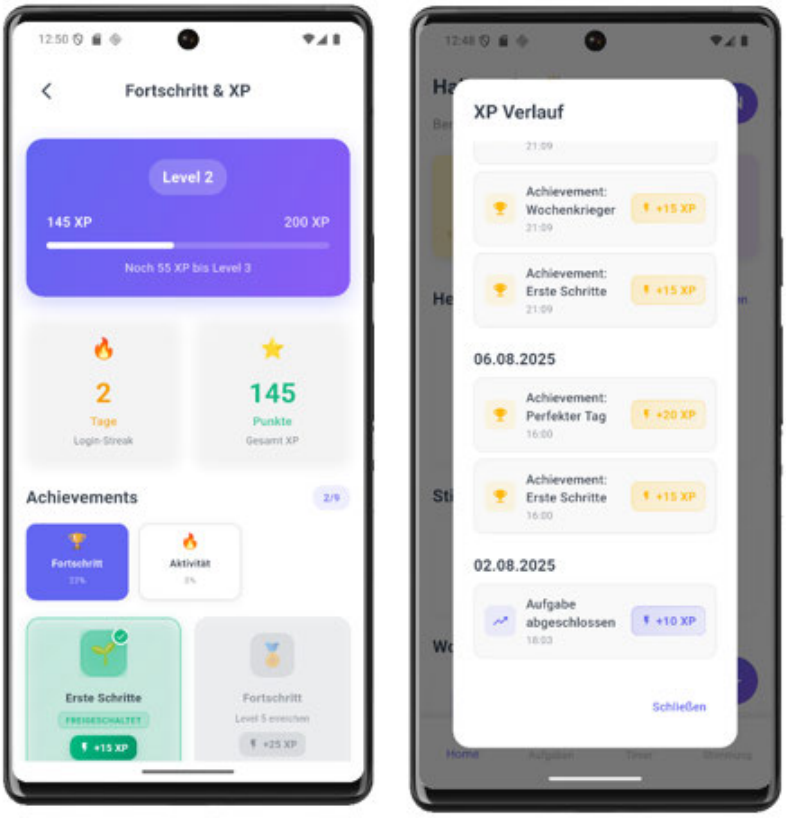

Abbildung 9: Fortschrittsanzeige

Wie bereits in Abschnitt 2.3.1 erläutert, orientiert sich das Design am Grundgedanken der Selbstbestimmungstheorie von Deci und Ryan (Hosseini et al., 2022). Das eingesetzte XP-System spricht dabei vor allem das Erleben von Kompetenz an. Nutzer können ihren Fortschritt einsehen. Autonomie wird durch die Möglichkeit gewährleistet, eigene Aufgaben zu definieren und flexibel zu gestalten. Und soziale Verbundenheit wird über Achievements gefördert, ohne dass Nutzer sich dazu gezwungen fühlen.

4.4.2 Belohnungsmechanismen

Neben der direkten XP-Vergabe durch Aufgabenabschluss implementiert die App ein Achievement-System. Diese zusätzlichen Belohnungsmechanismen sollen langfristiges Engagement fördern und verschiedene Nutzungsverhalten anerkennen. Die Belohnungen sind in *Fortschritt* und *Aktivität* unterteilt (siehe Abbildung 9). Unter der Kategorie *Fortschritt* werden mehrere Achievement-Typen vergeben. Dazu gehört zunächst das *Erste-Schritte*-Achievement. Dieses Achievement wird nach Beendigung der ersten Lerneinheit ausgelöst. Diese Belohnung wurde bewusst gewählt, um zu signalisieren, dass der Nutzer es geschafft hat, sich zu überwinden und seine erste Aufgabe beendet hat. Beim Level-Achievement erhält der Nutzer zusätzlich XP bei der Erreichung spezifischer Level. Hier gehören Level 5, 10, 15 und

20. Die Staffelung beginnt bei 25 XP für Level 5 und steigt in 5er-Schritten. Level 20 bildet eine Ausnahme mit 50 XP, da es den letzten Meilenstein darstellt. Die *Perfekten Tage*–Achievements belohnen die vollständige Erfüllung aller Tagesaufgaben an 1, 3, 7 oder 14 aufeinanderfolgenden Tagen. Dabei werden realistische Erwartungen berücksichtigt. Bereits ein perfekter Tag wird anerkannt und die Belohnung wird dem Nutzer gutgeschrieben. Wie in Abbildung 10 auf der Linken Seite dargestellt, erhält der Nutzer für jede Erfüllung eines Achievements eine visuelle Bestätigung mit positivem Feedback und weiteren Informationen.

Die Kategorie *Aktivität* fokussiert sich auf regelmäßige Nutzung der App. Das Login-Streak-System trackt durchgängig Appnutzung über 3, 7, 14 und 30 Tage. Jeder Meilenstein bringt zusätzlich XP. Regelmäßige Nutzung kann besonders bei Routine-orientierten Personen im Autismus-Spektrum positive Verstärkung schaffen. Die App dokumentiert nicht nur die Streak-Länge, sondern visualisiert den aktuellen Stand durch farbkodierte Pop-up-Nachrichten. Die Streak-Visualisierung verwendet ein zweifarbiges System. Bei kontinuierlichem Login erscheint die Nachricht in der Hauptfarbe Lila (0xFF8B5CF6). Ein Streak-Neustart wird durch Rot (0xFFEF4444) signalisiert (siehe Abbildung 10).

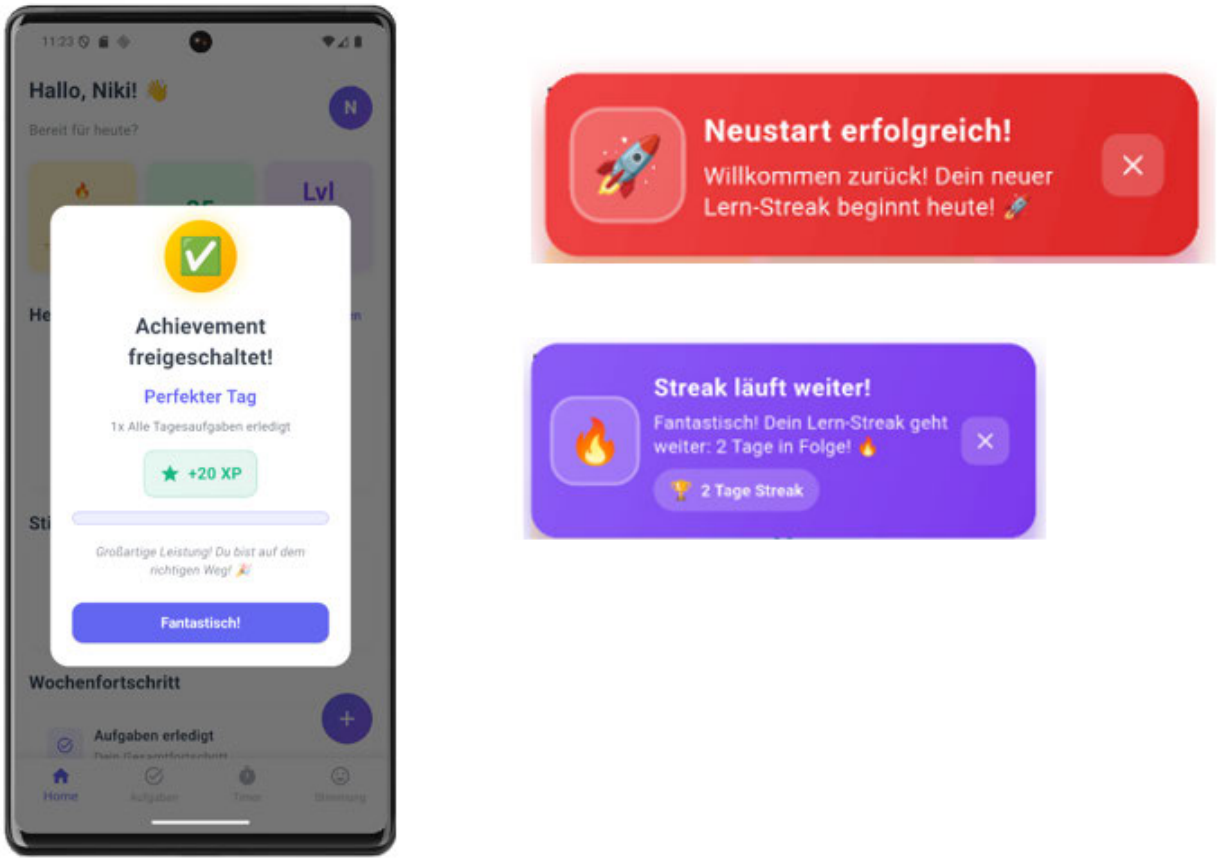

Abbildung 10: Achievement-Benachrichtigung und Login-Streak

5 Implementierung

5.1 Technologie-Stack

Für die technische Grundlage einer barrierefreien Produktivitäts-App ist eine solide Auswahl der Technologien erforderlich. Neben den funktionalen Anforderungen spielen auch Aspekte wie Zugänglichkeit, Performance und langfristige Wartbarkeit eine wichtige Rolle. Für neurodivergente Nutzer sind stabile Apps besonders wichtig. Technische Probleme können zu Frustration und Ablenkung führen, was die Produktivität beeinträchtigen kann. Aus diesem Grund wurde ein Technologie-Stack gewählt, der Lösungen kombiniert und dabei Flexibilität für zukünftige Erweiterungen bietet. Die Entscheidung für moderne Entwicklungswerkzeuge und cloud-basierte Dienste ermöglicht es, sich auf die Kernfunktionen zu konzentrieren, anstatt grundlegende Infrastruktur neu zu entwickeln. Gleichzeitig wird durch die Verwendung von standardisierten Technologien sichergestellt, dass die App langfristig unterstützt und weiterentwickelt werden kann.

Als Open-Source-Framework wurde Flutter gewählt. Für die App-Entwicklung bietet Flutter eine Menge Vorteile. Das Framework nutzt die Programmiersprache Dart und ermöglicht es somit, mit einer einzigen Codebasis sowohl Android- als auch iOS-Apps zu entwickeln. Während Flutter prinzipiell Cross-Platform-Entwicklung unterstützt, wurde bei der Entwicklung ein klarer Fokus auf Android-Geräte gelegt. Diese Entscheidung basiert auf mehreren praktischen Überlegungen. Android bietet deutlich niedrigere Einstiegshürden für die Entwicklung, da kein kostenpflichtiges Developer-Programm erforderlich ist. Dies war insbesondere für die zeitlich begrenzte Bachelorarbeit von Vorteil. Zudem steht für Android-Entwicklung ein vollwertiger Emulator zur Verfügung, der umfangreiche Tests ermöglicht, ohne auf physische Geräte angewiesen zu sein. Die meisten verfügbaren Testgeräte aus dem persönlichen Umfeld waren Android-Smartphones verschiedener Hersteller wie Samsung, Huawei und Google Pixel, was eine realitätsnahe Testumgebung gewährleistete. Dennoch bleibt durch die Flutter-Architektur die grundsätzliche Möglichkeit erhalten, die App später für iOS zu optimieren, ohne die Codebasis grundlegend umschreiben zu müssen. Die Architektur von Flutter basiert auf Widgets, die wiederverwendbar sind. Sie eignen sich für barrierefreie Anwendungen. Ein weiterer Vorteil ist die Performance von Flutter-Apps. Das Framework kompiliert direkt zu nativem Code, wodurch flüssige Animationen und schnelle Reaktionszeiten erreicht werden. Das ist von entscheidender Bedeutung für die Nutzer mit Aufmerksamkeitsstörungen, die von Verzögerungen oder ruckeligen Bewegungen schnell abgelenkt werden können. Die Entwicklung erfolgte mit dem Provider-Pattern für State-Management. Dieses Pattern trennt die UI-Logik von der Geschäftslogik, was den Code wartbar macht.

Supabase wurde als Backend gewählt, da es eine komplette Infrastruktur ohne eigene Server bietet. Das Modell, bei dem das Backend als Dienst bereitgestellt wird, verkürzt die Entwicklungszeit enorm und lässt mehr Raum für die grundlegenden Funktionen der App. Die PostgreSQL-Datenbank von Supabase sorgt dafür, dass Daten sicher gespeichert werden, und ermöglicht komplizierte Abfragen. Gemäß der Struktur der Datenbank-Tabellen aus Abbildung 2 wurden die zentralen Tabellen angelegt. Die Struktur der Datenbank ist so gestaltet, dass sie leicht erwartbar bleibt. Ein nützliches Feature von Supabase ist die Echtzeit-Funktion. Wenn Benutzer ihre Daten auf einem Gerät ändern, werden diese Änderungen sofort auf allen anderen Geräten aktualisiert, inklusive des eigenen Geräts, das im Moment in Nutzung ist. Zudem wird die gesamte Authentifizierung von Supabase übernommen. Das System erlaubt Anmeldungen mit E-Mail und Passwort und kann im späteren Zeitraum um weitere Methoden ergänzt werden.

In Bezug auf die Entwicklungsumgebung wurde Visual Studio Code als Haupteditor gewählt und wird mit Flutter- sowie Dart-Erweiterungen genutzt. Diese Kombination bietet nützliche Funktionen wie Syntaxhervorhebung, Debugging-Möglichkeiten und Hot-Reload für schnelle Entwicklungszyklen. Das Flutter-SDK verwendet zudem Abhängigkeiten. Dazu gehören `supabase_flutter` zur Backend-Integration, ∓Provider zum Verwalten des Zustands einer UI-Komponente sowie `Audioplayer` für akustisches Feedback und weitere Pakete für spezifische Funktionen. Diese Abhängigkeiten und vieles mehr werden in einer `pubspec.yaml`-Datei verwaltet.

Die Versionskontrolle erfolgt über Git. Damit lassen sich verschiedene Entwicklungsstände gut nachvollziehen. Neue Features werden in einem separaten Branch entwickelt und erst nach dem Nutzertesting in die main zusammengeführt.

5.2 Screen-Implementierung

In diesem Kapitel werden die implementierten Benutzeroberflächen der Mindspark-App anhand von Screenshots vorgestellt. Es zeigt, wie die in Abschnitt 4.3 und Abschnitt 4.4 behandelten Design-Prinzipien und Anforderungen in der finalen App umgesetzt wurden. Jeder Screen wird hinsichtlich seiner Funktionalitäten beschrieben.

5.2.1 Onboarding

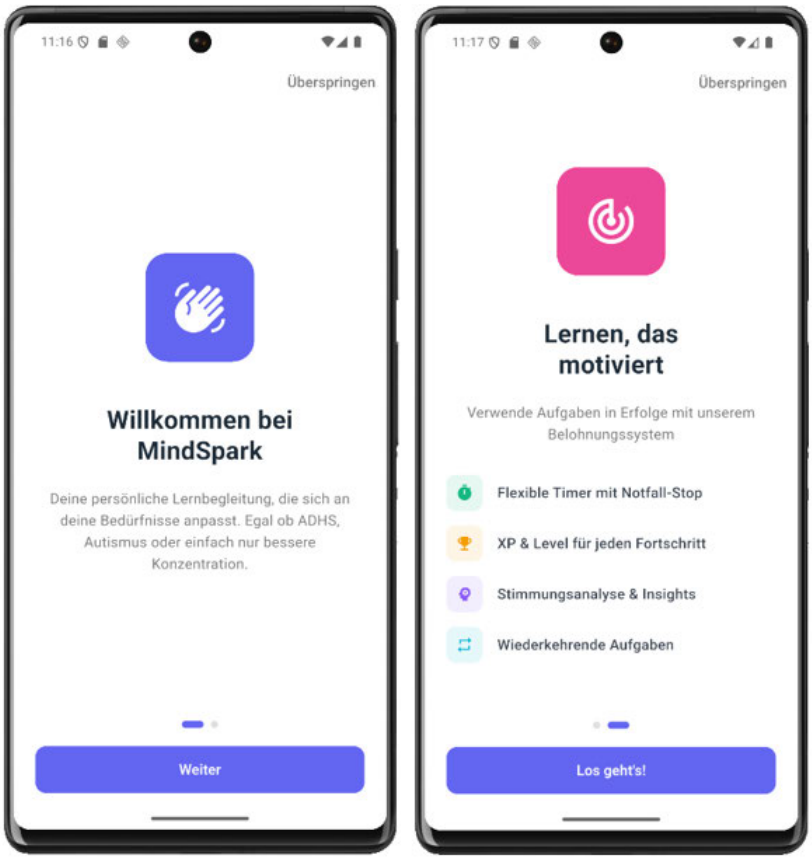

Abbildung 11: Onboarding Screen

Das Onboarding der MindSpark-App soll den ersten Eindruck eines Benutzers prägen und gleichzeitig die besonderen Bedürfnisse neurodivergenter Nutzer berücksichtigen. Dabei fängt es schon beim Onboarding an, potenzielle Barrieren bereits im ersten Moment abzubauen. Es bedeutet keine Überforderung durch zu viele Informationen. Dennoch muss das Onboarding ausreichend motivieren, um die App weiter zu nutzen. Das Design folgt einem durchdachten Ansatz. Die Farbwahl von Blau und Violett wirkt beruhigend und gleichzeitig motivierend. Die Farbtöne sind nachweislich weniger stimulierend als etwa grelle Rottöne. Laut Studien zeigt sich, dass die Verwendung von kühlen Farben wie Blau die Konzentration fördert (Herron, 2023).

5.2.2 Login/Registrierung

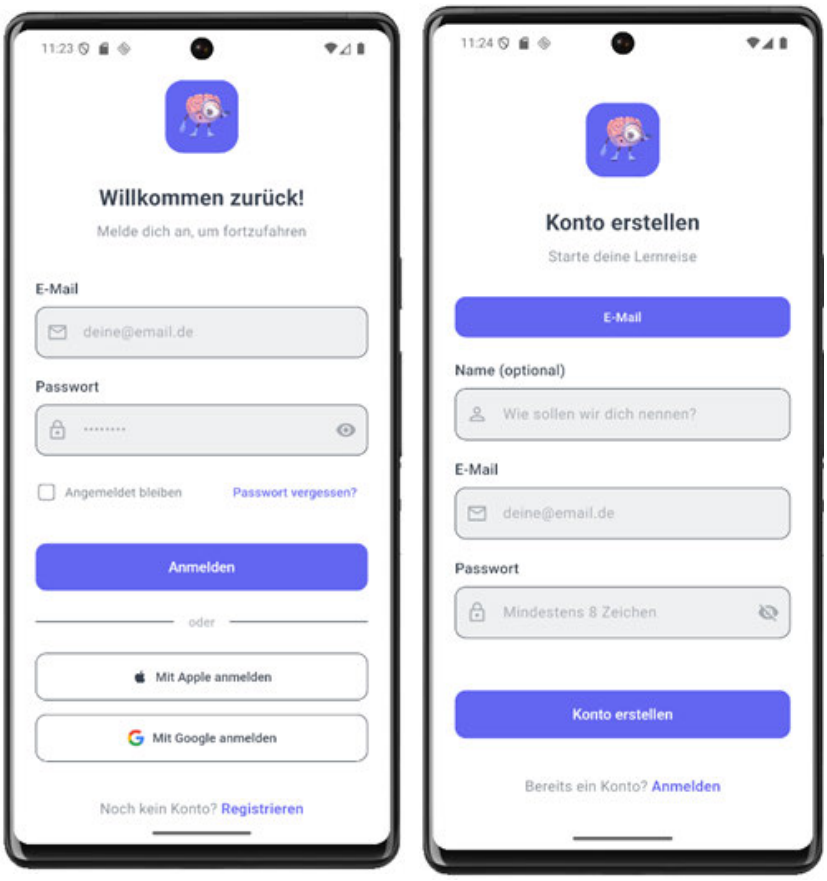

Abbildung 12: Login & Registrierung Screen

Nach Abschluss des Onboardings erfolgt die Weiterleitung zum Login-Bildschirm. Bei der ersten App-Nutzung existiert selbst verständlicherweise noch kein Benutzerkonto, weshalb eine Registrierung notwendig wird. Über den farblich lila abgesetzten Registrieren-Link unten auf dem Login-Interface startet der Registrierungsvorgang. Beide Bildschirme folgen einem ähnlichen Layout-Schema, was die Navigation konsistent hält. Optional kann der Nutzer einen Benutzernamen eingeben, während E-Mail und Passwort Pflichtfelder sind. Durch einen Klick auf *Konto erstellen* prüft die App die Eingaben und leitet bei Erfolg direkt zur Startseite weiter. Diese direkte Navigation verhindert überflüssige Zwischenschritte seitens des Nutzers. Das ist ein wichtiger Aspekt für Nutzer mit ADHS, da zusätzliche Unterbrechungen die Konzentration beeinträchtigen können. Nutzer, die bereits ein Benutzerkonto besitzen, durchlaufen einen einfachen Prozess mit E-Mail- und Passworteingabe. Das Augensymbol beim Passwortfeld ermöglicht die Sichtbarmachung der Eingabe, falls der Nutzer sich bei der Passworteingabe unsicher fühlt.

5.2.3 Home

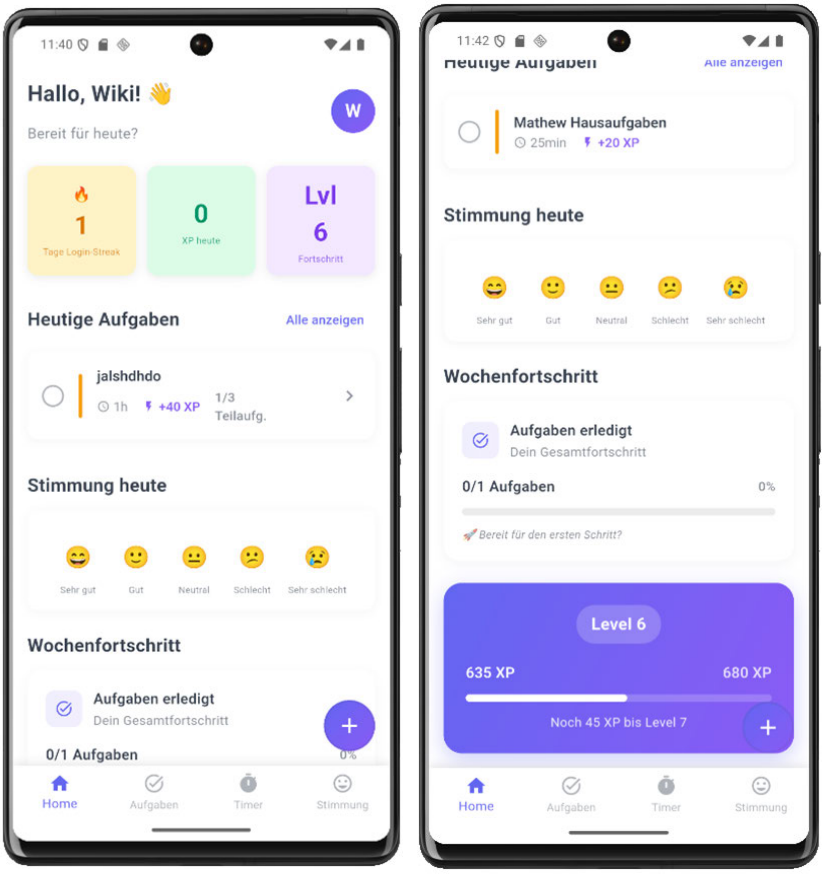

Abbildung 13: Home Screen

Die Startseite begrüßt den Nutzer mit seinem Benutzernamen, den er bei der Registrierung angegeben hat. Direkt neben der Begrüßung befindet sich ein Button mit dem jeweiligen Anfangsbuchstaben des Nutzernamens, welcher zu den Einstellungen führt. Drei farbige Stats-Boxes vermitteln dem Nutzer sofort wichtige Informationen. Links zeigt der Streak-Counter, wie viele Tage der Nutzer am Stück in der App war. Diese Streak-Anzeige ist besonders motivierend für neurodivergente Nutzer, da sie Routine und Fortschritt auf einen Blick erkennbar macht. Die mittlere Box zeigt die heutigen XP an. Durch Antippen öffnet sich eine detaillierte Ansicht aller XP-Vergaben, wann, wie viel und für welchen Typ der Nutzer XP erhalten hat (siehe Abbildung 9). So bleibt das Belohnungssystem völlig transparent und der Nutzer kann die Vergabe nachvollziehen. Die rechte Stats-Box zeigt das aktuelle Level an und dient gleichzeitig als visueller Anker für den Gamification-Fortschritt.

Der Bereich *Heutige Aufgaben* listet alle für den aktuellen Tag fälligen Tasks auf. Dies verschafft dem Nutzer beim App-Start einen sofortigen Überblick über anstehende To-dos, ohne dass der Nutzer erst suchen muss. Die farbigen Prioritätsbalken helfen bei der schnellen Einschätzung der Aufgabenwichtigkeit. Die Stimmungsauswahl auf der Startseite ermöglicht schnelle Check-ins. Ein einfacher Tap auf einem Emoji reicht aus und die Stimmung wird gespeichert. Die direkte Verfügbarkeit reduziert Hürden für

regelmäßiges Mood-Tracking. Beim weiteren Scrollen erscheint der Wochenfortschritt mit Prozentangaben für alle wöchentlichen Aufgaben. Diese Darstellung bietet Orientierung bezüglich der verbleibenden Wochenarbeitsleistung. Im unteren Bereich ist eine detaillierte Fortschrittsansicht mit dem aktuellen Level, Gesamt-XP und benötigten XP bis zum nächsten Level-Up. Der FAB bleibt konsistent an derselben Position und führt zum Hinzufügen neuer Aufgaben. Die untere Tab-Bar navigiert zu den vier Hauptbereichen, wobei die aktuelle Seite durch Lila hervorgehoben wird. Eine wichtige Orientierungshilfe, nicht nur für neurodivergente, sondern auch für neurotypische Nutzer.

5.2.4 Aufgaben

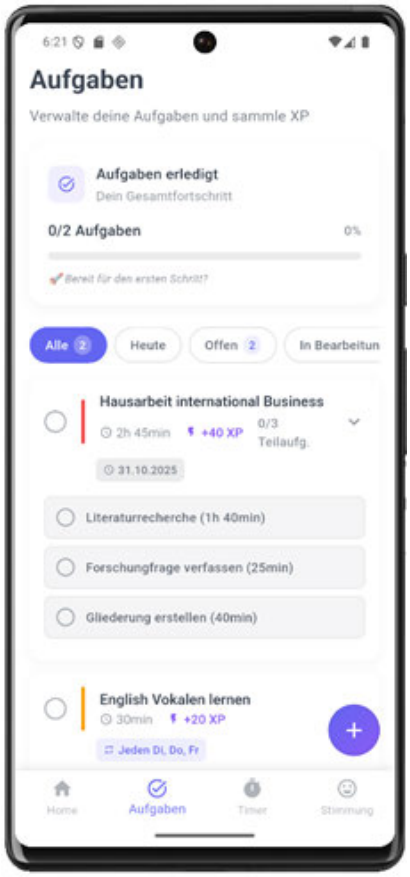

Abbildung 14: Aufgaben Screen

Über zwei Wege erreicht der Nutzer die Aufgabenliste. Entweder durch Klick auf *Aufgaben* in der Tab-Bar oder über *Alle anzeigen* bei den heutigen Aufgaben auf der Startseite. Diese doppelte Erreichbarkeit entspricht dem nicht-linearen Denkverhalten vieler ADHS-Nutzer. Im ersten Augenblick erhält der Nutzer eine Ansicht, wie viele Aufgaben er bereits abgeschlossen hat. Diese Zahl erscheint sowohl als Prozentangabe als auch als visueller Fortschrittsbalken. Motivierende Sprüche passen sich dem Fortschritt an und sorgen für positive Verstärkung, ohne aufdringlich zu wirken. In der Mitte des Bildschirmes sind verschiedene Filter-Kategorien dargestellt, die dem Nutzer helfen, seine Aufgabe zu sortieren. *Heute* für

tägliche Tasks, *Offen* für noch nicht begonnene Aufgaben, *In Bearbeitung* für laufende Tasks, *Erledigt* für abgeschlossene Aufgaben, *Überfällig* für verspätete und *Archiviert* für Aufgaben, die der Nutzer weder abschließen noch offenhalten möchte. Das Filtersystem reduziert kognitive Belastung, da Nutzer nur die relevanten Aufgaben sehen und nicht von einer langen, unstrukturierten Liste überfordert werden. Jeder Filter zeigt die entsprechende Anzahl an und führt bei Antippen zur gefilterten Aufgabenliste.

Die Aufgabenliste zeigt im Gegensatz zur Startseite alle Aufgaben an. Zudem zeigen die einzelnen Aufgaben, wie viele XP sie wert sind, sowie die eingeschätzte Zeit einer Aufgabenbearbeitung. Bei Aufgaben, die sich wiederholen, ist eine visuelle Ansicht für den Nutzer gegeben. Wie in Abbildung 14 zu sehen ist, ist die Aufgabe *Englisch Vokabeln lernen* für jeden Dienstag, Donnerstag und Freitag angesetzt. Durch ein Links-Swipe auf einer Aufgabe öffnet sich ein Aktionsmenü mit den Optionen *Löschen* und *Bearbeiten*. Diese Bewegung ist platzsparend und natürlich. Der Löschvorgang ist bewusst zweistufig. Eine Sicherheitsabfrage verhindert, dass der Nutzer versehentlich eine Aufgabe löscht, obwohl er eigentlich eine Aufgabe bearbeiten möchte. Die Bearbeiten-Option führt zum Aufgabenerstellungs-Interface, wo Anpassungen vorgenommen werden können. Wie auf der Startseite sitzt auch hier der FAB an seiner gewohnten Position. Dieser Button ist die einzige Stelle für das Hinzufügen neuer Aufgaben.

5.2.5 Aufgaben erstellen

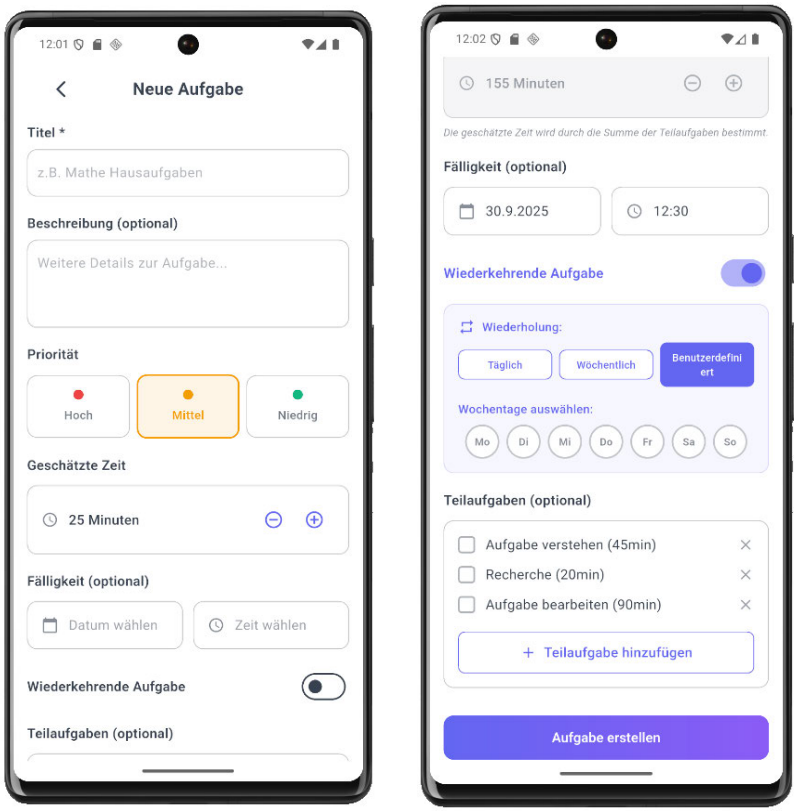

Abbildung 15: Aufgaben erstellen

Dieses Interface verzichtet bewusst auf die Navigation-Bar, da es sich um eine verschachtelte Navigation handelt. Für den Rückweg steht ein deutlich erkennbarer Pfeil-Button zur Verfügung, der zur vorherigen Seite zurückführt. Der Titel ist das einzige Pflichtfeld. Das erkennt der Nutzer durch das Stern-Symbol. Diese Minimierung ermöglicht eine schnelle Aufgabenerstellung, wenn es besonders eilt. Das Beschreibungsfeld erweitert die Aufgabe um Details für komplexere Tasks, die später mehr Erklärung brauchen. Das Prioritätsfeld hilft dem Nutzer beim Sortieren seiner Aufgaben nach Dringlichkeiten. Bei der Zeiteinschätzung ist in der App standardmäßig 25 Minuten eingestellt. Mit dem Plus- und Minus-Button kann der Nutzer die Zeit zwischen 5 und 240 Minuten anpassen. Diese Begrenzung verhindert zu lange Zeitblöcke, was gerade für ADHS-Nutzer eine Rolle spielt. Der Nutzer hat die Möglichkeit, ein Datum und eine Zeit einzugeben, zu der die Aufgabe abgeschlossen sein soll. Die Fälligkeit bietet dem Nutzer die Möglichkeit, sich aktiv mit der Aufgabe zu beschäftigen und sie nicht aufzuschieben.

Der Nutzer hat die Wahl, ob die Aufgabe eine einmalige ist oder ob er die Aufgabe wiederholen möchte. Das Wiederholungssystem bietet drei Muster. Täglich, wöchentlich oder benutzerdefiniert. Beim Letzteren können spezifische Wochentage gewählt werden und die Aufgaben erscheinen jeweils an den fälligen Tagen. Diese Option ist ideal für unregelmäßige Routine oder Termine, die nur an bestimmten Tagen anfallen. Das erspart dem Nutzer das ständige Neuerstellen gleicher Aufgaben. Für Nutzer, die so große Aufgaben erstellen und nicht wissen, wo sie am besten anfangen sollen, bietet sich die Chunking-Technik an, in der große Aufgaben in kleinere Teilaufgaben unterteilt werden kann. Jede Teilaufgabe erhält eine eigene Zeiteinschätzung, die der Nutzer ebenfalls mit einem Plus- und Minus-Button einstellen kann, und die App berechnet automatisch die Gesamtdauer. So basiert die ursprüngliche Zeitschätzung auf der Summe der Teilaufgaben. Nachdem der Nutzer seine Aufgabe überprüft hat, kann er mit einem Klick auf *Aufgabe erstellen* die Aufgabe hinzufügen und er wird automatisch zurück zur Aufgabenliste navigiert. Diese automatische Navigation spart einen zusätzlichen Klick und bringt den Nutzer direkt dorthin, wo er das Ergebnis seiner Arbeit sehen kann.

5.2.6 Timer

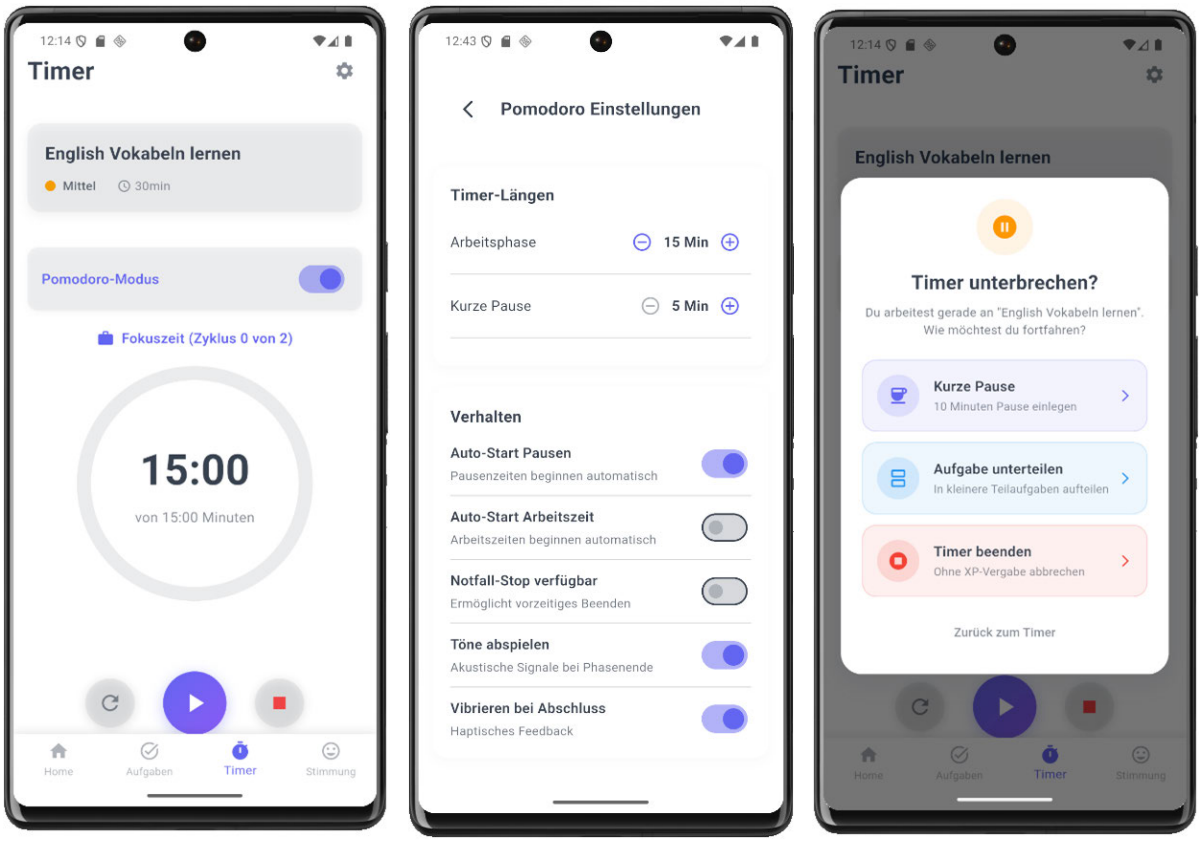

Abbildung 16: Timer Screen

Durch Klick auf eine beliebige Aufgabe landet der Nutzer direkt beim Timer. Eine bewusste Designentscheidung, die den Weg von der Planung zur Ausführung verkürzt. Der Nutzer hat einen klaren Überblick, welche Aufgabe ansteht, und kann die Priorität sowie die geschätzte Bearbeitungszeit einsehen. Der Nutzer kann frei wählen zwischen der Pomodoro-Technik oder dem Durcharbeiten der Aufgabe in einem Stück. Wenn er sich für die Pomodoro-Technik entscheidet, kann der Nutzer durch einen Klick auf das Zahnrad im oberen rechten Bereich des Bildschirmes in die Pomodoro-Einstellungen gelangen. Dort hat der Nutzer, wie in Abbildung 16 dargestellt, verschiedene Einstellungsmöglichkeiten. Er kann die Länge der Arbeits- und Pausenphasen beliebig nach seinen Präferenzen einstellen. Er entscheidet selbst, ob Übergänge automatisch ablaufen oder manuell bestätigt werden müssen sowie ob der Notfall-Stop-Button zur Verfügung gestellt wird. Das berücksichtigt verschiedene Arbeitstypen. Manche brauchen die strikte Automatik, während andere bewusste Entscheidungspunkte bevorzugen. Nach der Einstellung kann der Nutzer durch den Zurück-Pfeil wieder in das Timer-Interface gelangen. Der Timer zeigt nun die Fokuszeit mit den benötigten Zyklen an. Nach dem Abschluss einer Aufgabe folgt eine visuelle Erfolgsbestätigung. Bei Überforderung oder Blockade bietet der Notfall-Stop-Button drei Auswege für den Nutzer. Er kann eine zehnminütige Pause einlegen, die Aufgabe in Teilaufgaben zerlegen oder sie komplett beenden. Diese Optionen zeigen, dass mentales Wohlbefinden wichtiger ist als das Durchhalten

einer Aufgabe. Die drei Notfall-Optionen decken verschiedene Überforderungstypen ab.

5.2.7 Stimmungstracker

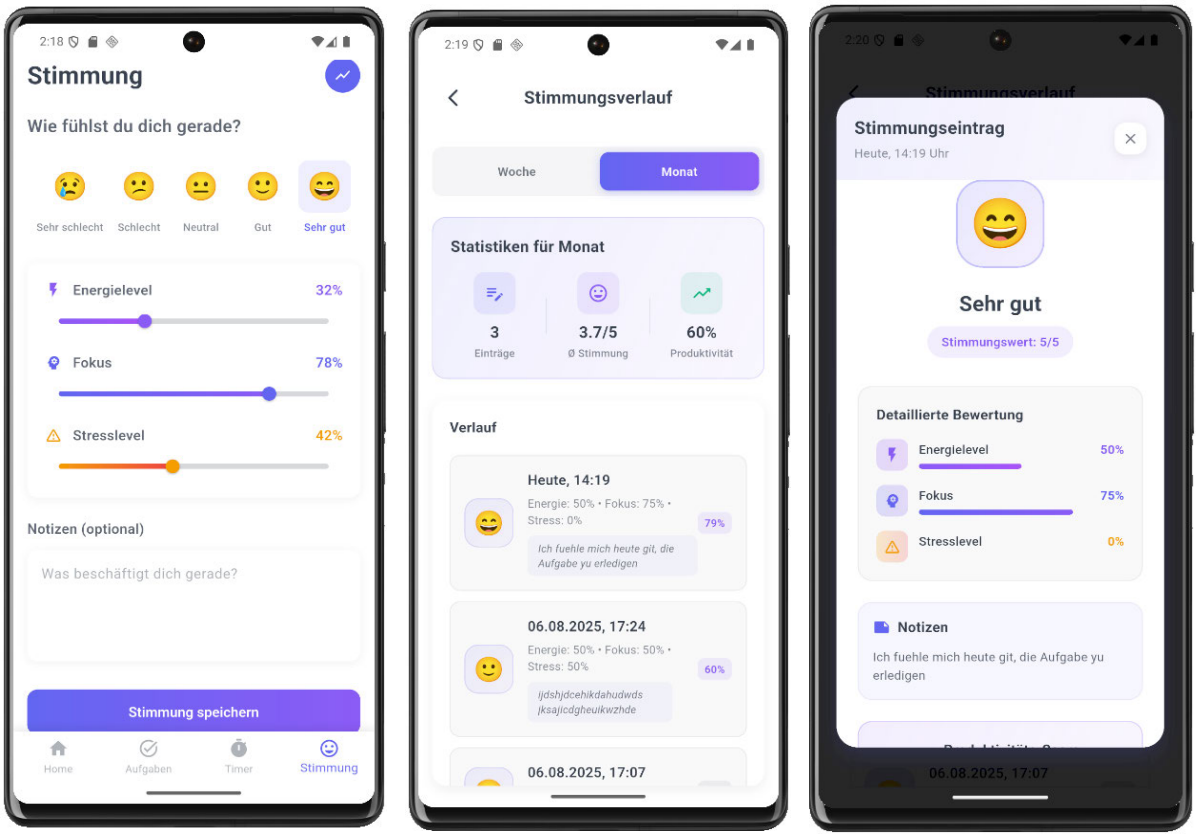

Abbildung 17: Stimmungstracker Screen

Der Nutzer kann zum Stimmungs-Interface navigieren, wenn er das Bedürfnis verspürt, nach einer Aufgabe seine Gefühle oder mentale Verfassung zu notieren. Er kann eines von fünf unterschiedlichen Emojis auswählen, die das Spektrum der Grundstimmung abdecken. Von sehr schlecht bis sehr gut. Das gewählte Emoji hebt sich farblich hervor und gibt ein klares visuelles Feedback zurück. Der Nutzer kann weitere drei Slider betätigen, die spezifische Felder abdecken. Er kann sein Energielevel, seinen Fokus und sein Stresslevel dadurch tracken. Diese Aufschlüsselung ermöglicht es, Zusammenhänge zwischen verschiedenen Zustandsparametern und Arbeitsleistung zu erkennen. Verspürt der Nutzer den Wunsch, seine Gefühle etwas mehr zu beschreiben, kann er das Notizenfeld nutzen. Nach dem Speichern erfolgt die automatische Navigation zum Stimmungsverlauf-Interface. Zunächst erfasst der Nutzer die Statistiksektion. Diese Sektion erfasst alle Einträge zusammen, von der Gesamtanzahl der erfassten Einträge über den berechneten Stimmungsdurchschnitt bis zum Produktivitätsdurchschnitt. Die Statistik hilft dabei, Veränderungen im emotionalen Wohlbefinden zu erkennen. Der Produktivitätsdurchschnitt wertet die Slider-Werte aus. Hohe Energie- und Fokuslevel steigern die Produktivität, während hoher Stress

sie mindert. Diese algorithmische Analyse macht unbewusste Zusammenhänge zwischen mentaler Verfassung und Leistungsfähigkeit sichtbar. Der Verlauf zeigt sowohl wöchentliche als auch monatliche Einträge in chronologischer Reihenfolge. Durch einen Klick auf einen beliebigen Eintrag öffnet sich eine Ansicht mit allen gespeicherten Parametern und dem berechneten Produktivitäts-Score für diesen Tag.

5.2.8 Einstellungen

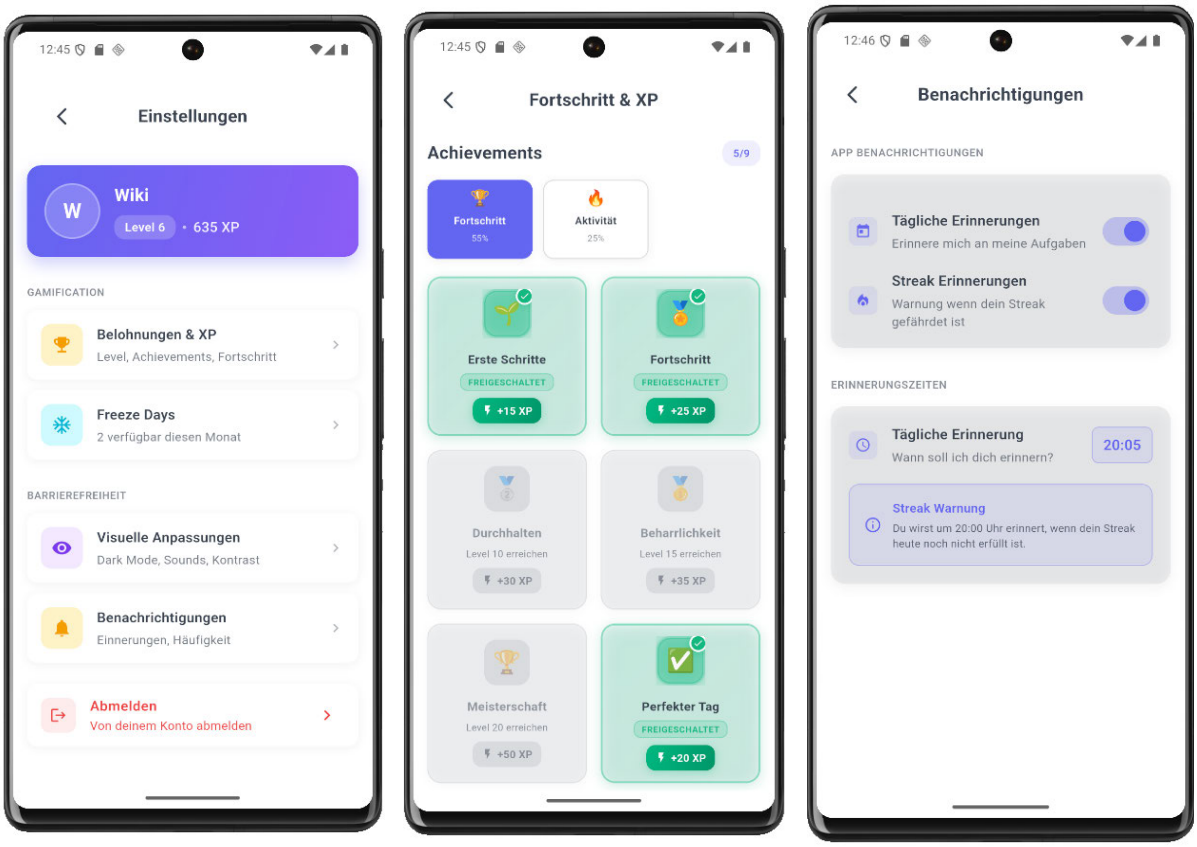

Abbildung 18: Einstellung Screen

Wenn der Nutzer zurück zur Startseite gelangt, erreicht er über den Nutzer-Button auf der rechten oberen Bildschirmseite die Einstellungen, die mit einer übersichtlichen Board startet. Diese zeigt das aktuelle Level und die gesammelten XP an und dient gleichzeitig als motivierender Einstieg in die Einstellungen. Die Einstellungen bieten direkten Zugang zu detaillierten Einsichten in Belohnungen und Fortschritt. Der Nutzer hat die Wahl, sich Belohnungen in der Kategorie *Fortschritt* detaillierter anzusehen oder in der Kategorie *Aktivitäten*. Beide Kategorien sprechen verschiedene Motivationstypen an. Wie in Abbildung 18 dargestellt sind in der Kategorie einige Achievements bereits freigeschaltet. Diese erscheinen in einem hellen Grün, während ausstehende Achievements in Grau ausgeblendet dargestellt werden. Diese Kodierung schafft sofort einen Überblick über den Fortschritt und motiviert zur Vervollständigung

der offenen Achievements, ohne zu überfordern oder gezwungen zu werden. Wenn der Nutzer zurück zu den Einstellungen gelangt, sieht er die Freeze-Day-Funktion. Diese erlaubt es Nutzern, ihren Streak einzufrieren, wenn sie sich nicht gut fühlen oder andere Umstände eine reguläre App-Nutzung verhindern. Am nächsten Tag läuft der Streak dann nahtlos weiter, ohne dass die aufgebaute Serie unterbrochen wird. Diese Optionen bieten neurodivergenten Nutzern Flexibilität und verhindern, dass schlechte Tage oder Überforderungen zu Frustration führen. Der Bereich *Visuelle Anpassungen* ermöglicht individuelle Theme-Modifikationen. Hier kann der Nutzer zwischen verschiedenen Farbschemata (Hell-/Dunkel-/Kontrastmodus), visuellen Anpassungen wie einer größeren Schrift und sensorischen Einstellungen wie Sound und Vibrationen wählen. Das Benachrichtigungssystem arbeitet zweistufig. Tägliche Erinnerungen helfen dabei, die App regelmäßig zu nutzen und geplante Aufgaben nicht zu vergessen. Nach Aktivierung der täglichen Erinnerungen kann der Nutzer seine Zeit eingeben, wann er die täglichen Erinnerungen erhalten möchte.

5.3 State-Management

In der App kommt das Provider-Pattern von Flutter zum Einsatz. Es handelt sich dabei um eine bewährte Methode, um Zustände effizient zu verwalten, ohne unnötige Ressourcen zu verbrauchen (Hallie & Osmani, o. J.). Dieses Pattern wurde bewusst gewählt, da er einfach bleibt und dabei flexibel genug ist, um komplexere Szenarien abzudecken. Beim Start der App sorgt ein `MultiProvider`-Setup in der `main.dart` dafür, dass alle benötigten `ChangeNotifier`-Services initialisiert und für die gesamte App bereitgestellt werden. Die Services laufen als Singleton-Instanzen, wodurch auf allen Screens derselbe Zustand vorliegt, was besonders relevant ist, wenn Nutzer empfindlich auf unerwartete Änderungen reagieren.

```

MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => TimerService.instance),
        ChangeNotifierProvider(create: (_) => ThemeProvider.instance),
        ChangeNotifierProvider(create: (_) => MoodService.instance),
        ChangeNotifierProvider(create: (_) => MoodHistoryService.instance),
        ChangeNotifierProvider(create: (_) => ProgressService.instance),
        ChangeNotifierProvider(create: (_) => HomeService.instance),
        ChangeNotifierProvider(create: (_) => AchievementManager.instance),
      ],
      child: const MyApp(), )
```

Listing 11: Provider Pattern

Alternativen wie BloC oder Riverpod bieten ebenfalls Lösungen an. BLoC trennt Logik und Darstellung sehr strikt und setzt stark auf Streams. Das macht den Ansatz sinnvoll, aber auch komplex und erfordert oft mehr Boilerplate-Code. Riverpod wiederum ist flexibel und vermeidet einige Einschränkungen des klassischen Providers, kann jedoch den Einstieg erschweren (Aswalekar & Vishwakarma, 2025). Für die geplante App war Einfachheit ein entscheidender Faktor. Das Provider-Pattern erfüllt die Punkte weniger kognitive Last, klare Strukturen und schnelle Umsetzung. Wenn Änderungen an einer Stelle sofort, aber kontrolliert in der Oberfläche sichtbar werden, entsteht ein Gefühl von Sicherheit. Gerade bei einer App für neurodivergente Nutzer zählt eine Architektur, die klare Datenflüsse und stabile Oberflächen garantiert. Dieses Ziel ist zwar auch erreicht, allerdings mit zusätzlicher Komplexität im Code, die das Risiko für unklare Zustände erhöht. Damit die Oberfläche nicht starr wirkt, sondern direkt auf Änderungen reagiert, kommt das `Consumer`-Widget ins Spiel. Es zeichnet nur die Bereiche neu, die von einer Änderung betroffen sind, und spart so Ressourcen. Für Echtzeitereignisse, wie das Ende eines Pomodoro-Timers, wird ein System auf Basis von `StreamController` eingesetzt. Es ermöglicht, dass Services wie der `TimerService` Nachrichten in einem Stream schreiben, auf den die UI hört.

```
1 // StreamController für In-App-Benachrichtigungen
2  final _notificationStreamController = StreamController<Map<String, dynamic>>.
     broadcast();
3  Stream<Map<String, dynamic>> get notificationStream =>
     _notificationStreamController.stream;
```

Listing 12: StreamController

Die Benachrichtigung erscheint bewusst nicht als passives Banner, das automatisch verschwindet, sondern bleibt als Overlay sichtbar, bis der Nutzer reagiert. Vor allem wenn man passive Meldungen leicht aus dem Blick verliert. Durch die manuelle Bestätigung bleibt die Kontrolle beim Nutzer. Da mobile Apps nicht immer eine stabile Verbindung haben, setzt die App auf Caching. Inhalte, die zuletzt geladen wurden, bleiben verfügbar, auch wenn die Verbindung unterbrochen wird. Fehler werden nicht einfach mit Meldungen angezeigt, sondern klar kommuniziert. Ein `Result`-Pattern sorgt dafür, dass Entwickler zwischen Erfolg und Fehlschlag unterscheiden können und die UI konsistent reagiert.

6 Evaluation

6.1 Test-Strategien

Das Testen mobiler Apps erfordert einen Ansatz, der nicht nur auf eine einzige Methode setzt, sondern mehrere miteinander kombiniert (Weichbroth, 2024). So lassen sich sowohl technische Anforderungen als auch die individuellen Erfahrungen der Nutzer berücksichtigen. Da mobile Geräte ihre eigenen Besonderheiten haben und neurodivergente Menschen andere Erwartungen mitbringen, stützt sich diese Arbeit auf zwei Ansätze, die sich gegenseitig ergänzen. Zum einen Online-Umfragen und zum anderen gezieltes Usability-Testing. Durch diese Kombination können Einschätzungen von Nutzern gesammelt werden und man gewinnt detaillierte Einblicke in ihr Verhalten während der Nutzung. Der Prozess ist in mehrere Stufen gegliedert und folgt einem wiederkehrenden Ablauf. Dabei werden erste Rückmeldungen direkt genutzt, um die Anwendung zu überarbeiten, bevor umfangreichere Tests durchgeführt werden (Weichbroth, 2024). Online-Befragungen bilden die Grundlage dieser Bewertungsstrategie. Sie sind schnell durchführbar und erreichen viele Menschen, ohne das Budget zu sprengen. Mit standardisierten Fragebögen lassen sich Antworten verschiedener Nutzergruppen gut vergleichen. Besonders bei der Entwicklung für spezielle Zielgruppen zahlt es sich aus, bereits früh im Prozess Feedback zu bekommen.

6.2 Usability-Testing

Das Usability-Testing mobiler Apps unterscheidet sich methodisch von herkömmlichen Desktop-Tests durch die Integration von verschiedenen Datenerhebungstechniken. Weichbroth (2024) nennt vier Hauptmethoden, die normalerweise in einer Usability-Test-Sitzung eingesetzt werden. Fragebögen, Beobachtung der Teilnehmer, lautes Denken (Think-Aloud) und Interviews. Diese Kombination ermöglicht es, sowohl beobachtbares Verhalten als auch subjektive Eindrücke der Nutzer festzuhalten. Die Bewertung von Anwendungen für neurodivergente Menschen bringt spezielle Anforderungen an die Testmethoden mit sich. Während übliche Usability-Metriken wie Erfolgsrate oder Bearbeitungsdauer wichtige Anhaltspunkte liefern können, muss man darüber hinaus ebenso Aspekte wie geistige Belastung, sensorische Verträglichkeit und emotionale Reaktionen im Auge behalten (Weichbroth, 2024). Das erfordert Anpassungen an den Testbedingungen und -verfahren, damit eine echte und stressfreie Testsituation entsteht. Die Entscheidung zwischen Laborstudien und Feldstudien gewinnt hier zusätzliche Bedeutung. Während Laborstudien kontrollierte Bedingungen bieten können, ermöglichen Feldstudien es wiederum, das Nutzungsverhalten in natürlichen Umgebungen zu beobachten. Feldstudien spielen eine wichtige Rolle,

da neurodivergente Menschen oft auf vertraute Umgebungen und Routine angewiesen sind (Weichbroth, 2024).

Die erstellte Teststrategie für diese Bachelorarbeit kombiniert die Vorteile beider Test-Methoden. Zu Beginn sammelt man über Online-Umfragen erste Nutzerrückmeldungen und Eindrücke zur Benutzeroberfläche. Mit diesen Informationen verbessert man die App, bevor tiefere Tests mit den tatsächlichen Nutzern durchgeführt werden. Laut einer Studie werden etwa 5 % aller mobilen Apps wirklich erfolgreich, während etwa 80 % bis 90 % der veröffentlichten mobilen Apps nach einmaliger Nutzung wieder verschwinden. Durch solche Tests wird sichergestellt, dass die entwickelte App den Bedürfnissen von neurodivergenten Nutzern gerecht wird und eine langfristige Nutzung ermöglicht (Weichbroth, 2024).

6.2.1 Durchführung

Die Untersuchung fand im Zeitraum des Bachelorprozessen statt und umfasste insgesamt 12 Teilnehmende im Alter zwischen 16 und 24 Jahren. Die Zusammensetzung der Gruppe war eine Mischung aus Menschen mit ADHS und ASS sowie neurotypische Teilnehmende. Die Rekrutierung erfolgte über persönliche Kontakte. Eine Bekannte, die in einer Einrichtung tätig ist, in der sich neurodivergente Menschen regelmäßig treffen und austauschen, stellte den Kontakt zu potenziellen Teilnehmenden her. Alle Beteiligten erklärten sich freiwillig zur Teilnahme bereit. Die Tests fanden in einer ruhigen Einrichtung statt. Der Raum bot eine entspannte Atmosphäre.

Für die Durchführung der Tests wurde eine bewusste gemischte Geräteauswahl bereitgestellt, um unterschiedliche Displaygrößen und Betriebssysteme einzubeziehen. Die Mehrheit der Teilnehmenden nutze eigene Android-Smartphones, überwiegend Geräte von Samsung. Zwei Personen testeten die App auf einem iPhone. Zusätzlich standen ein Google Pixel 7 sowie ein Tablet des Herstellers Huawei zur Verfügung, um die Bandbreite an Hardware zu erweitern. Für die Für die Aufzeichnung der Think-Aloud-Protokolle wurde das Mikrofon des persönlichen Smartphones verwendet. Die Audiodaten wurden ausschließlich für die Auswertung genutzt und auf Wunsch der Teilnehmenden im Anschluss gelöscht. Jede Testsitzung dauerte etwa 60 Minuten und folgte einem strukturierten, aber flexiblen Ablauf:

Phase 1: Begrüßung und Vorbereitung
Viele neurodivergente Teilnehmer benötigen etwas Zeit, um sich an die Situation zu gewöhnen. Ein kurzes Gespräch über ihre Erfahrung mit Apps half dabei, eine entspannte Atmosphäre zu schaffen. Das Einverständnis wurde ausführlich besprochen, wobei besonders auf die Freiwilligkeit und die Möglichkeit hingewiesen wurde, jederzeit abzubrechen.

Phase 2: App-Einführung

Die Teilnehmer erhielten einen kurzen Überblick über die App, ohne zu viele Details preiszugeben. Wobei sie durch die Umfrage eine Einschätzung der Benutzeroberfläche hatten. Wichtig war hier, Erwartungen zu setzen und zu erklären, dass es sich um einen Prototyp handelt.

Phase 4: Think-Aloud-Training

Viele Teilnehmer waren zunächst unsicher, aber eine kleine Übungsaufgabe mit einer bekannten App half dabei, sich an die Verbalisierung der Gedanken zu gewöhnen.

Phase 5: Haupttest

Der Kern der Evaluation umfasste fünf Hauptaufgaben und jeweils spezifische Szenarien für ADHS- und Autismus-Teilnehmer. Hier zeigte sich die Stärke des Think-Aloud-Protokolls und viele Erkenntnisse und Probleme wären ohne diese Verbalisierung nicht aufgefallen.

6.2.2 Testszenarien für neurodivergente Nutzer

Nach Recherche und Gesprächen mit Betroffenen wurden die Aufgaben in zwei Bereiche gegliedert: grundlegende Kernaufgaben für alle sowie separate Szenarien, die speziell auf Personen mit ADHS oder Autismus zugeschnitten waren. Die Abfolge der Testaufgaben wurde bewusst so gestaltet, dass sie den typischen Ablauf einer Nutzung nachahmt. Von der erstmaligen Installation bis hin zur regelmäßigen Anwendung. In jedem Szenario gab es eine feste Struktur. Eine praktische Aufgabe stellte den Bezug zum Alltag her, messbare Kriterien ermöglichen eine objektive Bewertung, ob die Teilnehmenden die Aufgabe erfolgreich gelöst haben, und zusätzliche Beobachtungspunkte gaben Hinweise, worauf bei der Durchführung geachtet werden sollte. Auf diese Weise ließen sich vergleichbare Daten für alle Teilnehmenden sammeln. Die fünf Kernaufgaben bilden das Grundgerüst jeder Testsitzung und decken die wichtigsten Funktionsbereiche der App ab. Sie beginnen mit dem ersten Kontakt zur App und führen schrittweise zu fortgeschritteneren Features.

In einem der Szenarien sollten die Teilnehmenden eine Aufgabe in der App anlegen. Die Teilnehmer erhielten folgende Aufgabe: *„Du hast morgen ein wichtiges Referat über Klimawandel zu halten und musst noch einiges dafür vorbereiten. Erstelle eine entsprechende Aufgabe in der App, damit du den Überblick behältst.“*

Dieses Szenario testete nicht nur die Funktionalität der Aufgabenerstellung, sondern auch das Verständnis für Prioritäten, Zeitschätzung und Fälligkeitsdaten. Um den Erfolg der Durchführung zu bewerten,

wurden sowohl Erfolgskriterien als auch Beobachtungspunkte definiert. Diese sind in der folgenden Tabelle zusammengefasst, um einen klaren Überblick zu geben und die Bewertung nachvollziehbar zu machen.

Bezeichnung	**Beschreibung**
Erfolgskriterien	• Aufgabe wird erfolgreich mit Titel und Details angelegt • Falligkeitsdatum wird korrekt gesetzt • Prioritätsstufe wird verstanden und verwendet
Beobachtungspunkte	• Intuitive Nutzung der Eingabefelder • Verständnis der Prioritäts-Farbkodierung • Reaktion auf Pflichtfelder und Validierung • Umgang mit der Zeitschätzungsfunktion

Tabelle 9: Test Szenario - Aufgabe erstellen

Ein weiteres wichtiges Szenario lautete: *„Nach längerem Arbeiten mit der App ist dir aufgefallen, dass die helle Oberfläche anstrengend für deine Augen ist und die kleinen Textgrößen das Lesen erschweren. Schaue, ob du die Einstellungen an deine Bedürfnisse anpassen kannst.“*

Ziel war es, die Einstellung so zu verändern, dass die Nutzung angenehmer und weniger belastend wirkt. Dieses Szenario spiegelt damit eine typische Alltags-Erfahrung wider, bei der Nutzer aktiv nach individuellen Anpassungen suchen, um die App langfristig komfortabel einsetzen zu können. Auch hier wurden zur Bewertung Erfolgskriterien und Beobachtungspunkte festgelegt.

Bezeichnung	**Beschreibung**
Erfolgskriterien	• Einstellungsbereich wird gefunden • Mindesten zwei Accessibility-Features werden erfolgreich aktiviert • Verbesserungen werden subjektiv wahrgenommen
Beobachtungspunkte	• Intuitive Navigation zu den Einstellungen • Verständnis der Accessibility-Optionen • Sofortige Wahrnehmung der Änderung • Zufriedenheit mit verfügbaren Optionen

Tabelle 10: Test Szenario - Einstellungen anpassen

Für ADHS-Teilnehmer wurden zusätzliche Szenarien entwickelt, die typische Herausforderungen dieser Zielgruppe adressieren. Der Fokus lag auf Situationen mit der Überforderung. In diesem Szenario sollten die Teilnehmenden prüfen, wie die App unterstützt, wenn ein Tag mit zu vielen Aufgaben überfordert. Die Aufgabenstellung lautet: *„Du hast dir heute fünf verschiedene Aufgaben vorgenommen: Referat vorbereiten, Hausaufgaben machen, Zimmer aufräumen, Sport treiben und mit Oma telefonieren. Du bist gerade dabei, die Hausaufgaben zu erledigen, merkst jedoch, dass das alles zu viel für einen Tag ist und du dich überfordert fühlst.“*

Dieses Szenario testet den Notfall-Stop der App und bewertet, wie gut die Anwendung Nutzer bei Überforderung unterstützt. Zur Auswertung des Szenarios wurden die wichtigsten Erfolgskriterien und Punkte zur Beobachtung bestimmt.

Bezeichnung	**Beschreibung**
Erfolgskriterien	• Aufgaben können verschoben werden • Aufgaben können in Teilaufgaben unterteilt werden • Positive Bestätigung bei Nutzung der Flexibilität
Beobachtungspunkte	• Intuitive Bedienung des Notfall-Stops • Verständnis der Flexibilität-Optionen • Leichtes Verschieben oder Aufteilen von Aufgaben • Subjektives Gefühl der Entlastung • Reaktion auf visuelles Feedback der App

Tabelle 11: Test Szenario - Überfoderung

Die drei dargestellten Szenarien zeigen, wie verschiedene Aufgaben und Alltagssituationen genutzt werden können, um die Funktionalität und Benutzerfreundlichkeit der App gezielt zu testen. Die Tests geben Einblicke in die Grundfunktionen und gleichzeitig zeigen sie, wie gut sich die App an individuelle Bedürfnisse anpasst. Auf Grundlage dieser Erkenntnisse können im folgenden Kapitel die Ergebnisse sowie die Beobachtungen im Detail vorgestellt werden.

6.2.3 Ergebnisse und Erkenntnisse

Zunächst werden die Ergebnisse der Online-Umfragen ausgewertet, die von den Teilnehmenden vor den praktischen Tests ausgefüllt wurden. Diese dienten als Vorabeinschätzung und ermöglichten eine fundierte Vorbereitung der Testszenarien. Anschließend folgt die Analyse der Beta-Testing- und

UI-Evaluation-Ergebnisse, die vor Ort durchgeführt wurden. Besonders für iOS-Nutzer stellte dies eine technische Herausforderung dar, da Apples Sicherheitsrichtlinien die direkte Installation von Beta-Versionen verhindern. iPhone-Nutzer mussten ihre Geräte direkt am Entwicklungslaptop anschließen, um über Xcode die Testversion zu installieren und die Evaluation durchführen zu können. Android-Nutzer konnten hingegen die APK-Datei direkt auf ihre Geräte laden.

An der Befragung nahmen insgesamt 14 Personen teil, wobei die Hälfte (50 %) sich als neurodivergent identifizierte. Ein Drittel der Teilnehmenden (35,7 %) gab an, neurotypisch zu sein, während 14,3 % unsicher waren oder keine Angabe machten. Diese Verteilung ermöglichte es, sowohl spezifische Bedürfnisse neurodivergenter Lernender als auch allgemeine Lernherausforderungen zu erfassen.

Diese prozentuale Verteilung bezieht sich auf die gesamte Stichprobe von 14 Personen und verdeutlicht, dass bestimmte Lernschwierigkeiten gruppenübergreifend auftreten, während sich bei der isolierten Betrachtung der neurodivergenten Teilgruppe teilweise noch ausgeprägtere Muster zeigten. Wie die Auswertung der ersten Umfrage zeigt (siehe Abbildung 19), erkennt diese die hauptsächlichen Schwierigkeiten beim Lernen. Die Teilnehmenden nannten dabei folgende Kernprobleme:

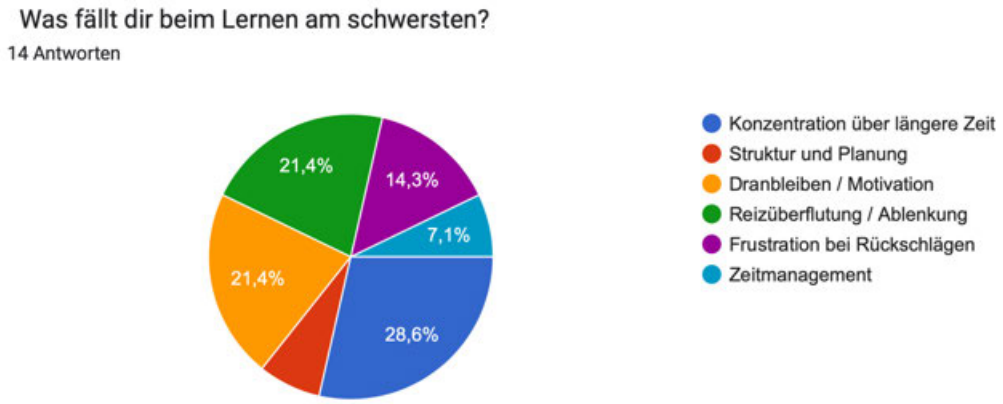

Abbildung 19: Verteilung der häufigsten Lernherausforderungen in der Befragung

- Konzentrationsprobleme und Ablenkbarkeit stellen für 78 % der befragten neurodivergenten Personen die größte Herausforderung dar. Besonders ADHS-Betroffene berichteten über Schwierigkeiten, über längere Zeiträume fokussiert zu bleiben. Diese Erkenntnis beeinflusste direkt die Entwicklung der Timer-Funktion mit flexiblen Arbeitsintervallen.
- Struktur- und Planungsdefizite wurden von 65 % der neurodivergente Teilnehmer benannt. Personen im Autismus-Spektrum betonten dabei besonders den Bedarf nach vorhersehbaren Routinen und klaren Strukturen. Dies führte zur Integration von konsistenten Navigationselementen und wiederkehrenden Layoutmustern.

- Motivationsprobleme beim Durchhalten betreffen 72 % der Befragten. Hier zeigte sich deutlich der Bedarf nach angepassten Gamification-Elementen, die weder überfordern noch zu kindlich wirken.

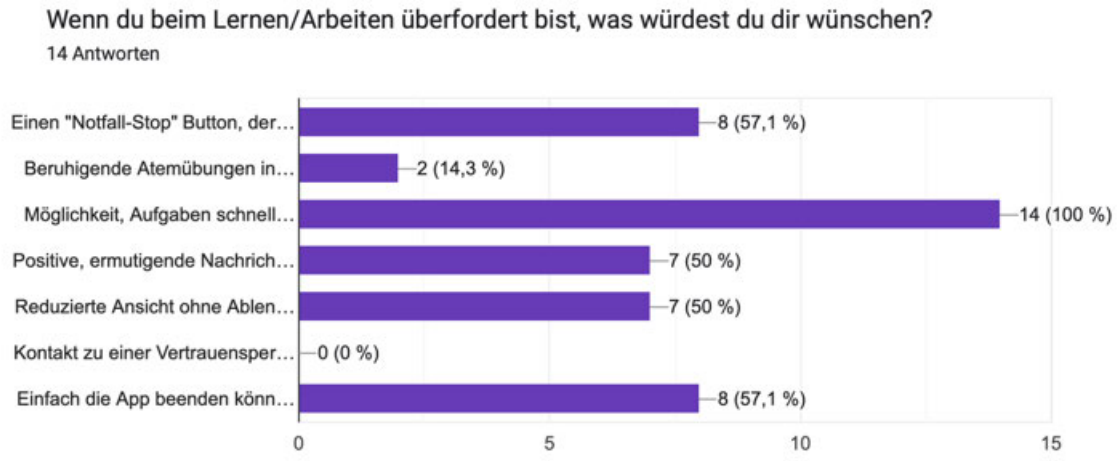

Abbildung 20: Unterstützungsfunktionen bei Überforderung

Die Bewertung gewünschter Unterstützungsfunktionen bei Überforderung lieferte aufschlussreiche Erkenntnisse über die spezifischen Bedürfnisse neurodivergenter Lernender in kritischen Momenten. Flexibilität bei Aufgabenverschiebung wurde von allen 14 Teilnehmenden (100 %) als wichtige Funktion bewertet. Diese einhellige Zustimmung unterstreicht die zentrale Bedeutung adaptiver Systeme, die sich an die schwankenden Kapazitäten neurodivergenter Nutzer anpassen können. Die Möglichkeit, Aufgaben spontan zu verschieben, ohne Bestrafung oder negative Konsequenzen zu erfahren, scheint ein grundlegendes Bedürfnis dieser Zielgruppe darzustellen. Notfall-Stop-Funktionalität fand bei 57,1 % der Befragten Zustimmung. Diese hohe Akzeptanz bestätigt die Relevanz von Ausstiegsmöglichkeiten in Lernanwendungen für neurodivergente Menschen. Die Implementierung eines solchen Features in MindSpark reagiert direkt auf das Bedürfnis nach Kontrolle und Selbstbestimmung in überwältigenden Situationen. Einfache App-Beendigung ohne negative Konsequenzen wurde ebenfalls von 57,1 % gewünscht. Diese Präferenz verstärkt die Bedeutung stressfreier Nutzungserfahrungen und unterstützt das Designprinzip, dass Nutzer jederzeit ohne schuldhafte Gefühle die Anwendung verlassen können. Positive, ermutigende Nachrichten und reduzierte Ansichten ohne Ablenkungen erhielten jeweils 50 % Zustimmung. Diese moderate, aber bedeutsame Resonanz zeigt, dass emotionale Unterstützung und sensorische Entlastung wichtige Aspekte der Überforderungsbehandlung darstellen. Beruhigende Atemübungen fanden nur bei 14,3 % Zustimmung statt, was darauf hindeutet, dass technische Unterstützungsmechanismen gegenüber therapeutischen Ansätzen bevorzugt werden. Bemerkenswert ist, dass Kontakt zu Vertrauenspersonen von keinem Teilnehmenden gewünscht wurde, was möglicherweise die Präferenz für autonome Bewältigungsstrategien widerspiegelt.

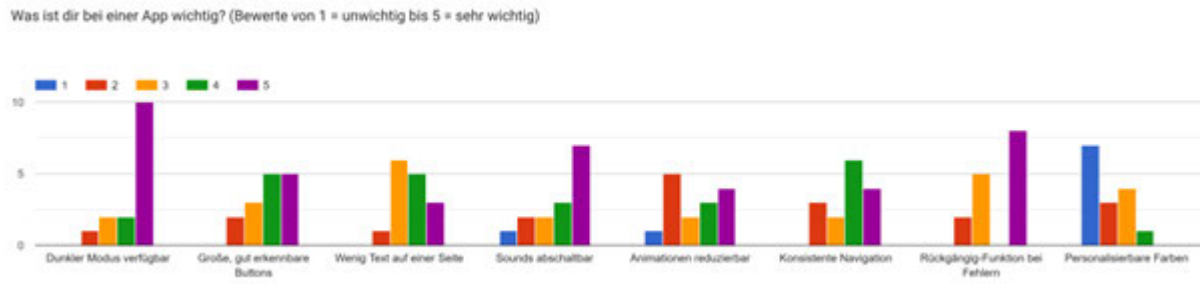

Abbildung 21: Unterstützungsfunktionen bei Überforderung

Die Bewertung verschiedener Barrierefreiheits-Features verdeutlichte die Bedürfnisse der Zielgruppe und bestätigte die Notwendigkeit anpassbarer Designelemente. Dunkler Modus erhielt die höchste Bewertung mit durchschnittlich 4,7 von 5 Punkten, wobei 10 von 14 Teilnehmenden (71,4 %) dieses Feature als sehr wichtig (Note 5) einstuften. Diese deutliche Präferenz bestätigt die sensorischen Bedürfnisse vieler neurodivergenter Personen und unterstützt die frühzeitige Implementierung des Dark Mode in MindSpark. Abschaltbare Sounds erreichten eine durchschnittliche Bewertung von 4,3 Punkten. Die starke Präferenz für Sound-Kontrolle unterstreicht die auditive Sensitivität vieler neurodivergenter Personen und bestätigt die Designentscheidung, alle akustischen Elemente optional zu gestalten. Personalisierbare Farben wurden mit durchschnittlich 3,9 Punkten bewertet, was eine moderate, aber relevante Präferenz darstellt. Die Bewertungsspanne zeigt, dass individuelle Anpassungsmöglichkeiten geschätzt werden, auch wenn sie nicht für alle Nutzer höchste Priorität haben. Reduzierbare Animationen erhielten nur 2,8 Punkte im Durchschnitt. Diese vergleichsweise niedrige Bewertung überrascht angesichts der zuvor geäußerten Bedenken bezüglich visueller Überstimulation und deutet möglicherweise darauf hin, dass Animationen weniger problematisch sind als zunächst angenommen, solange sie angemessen gestaltet werden.

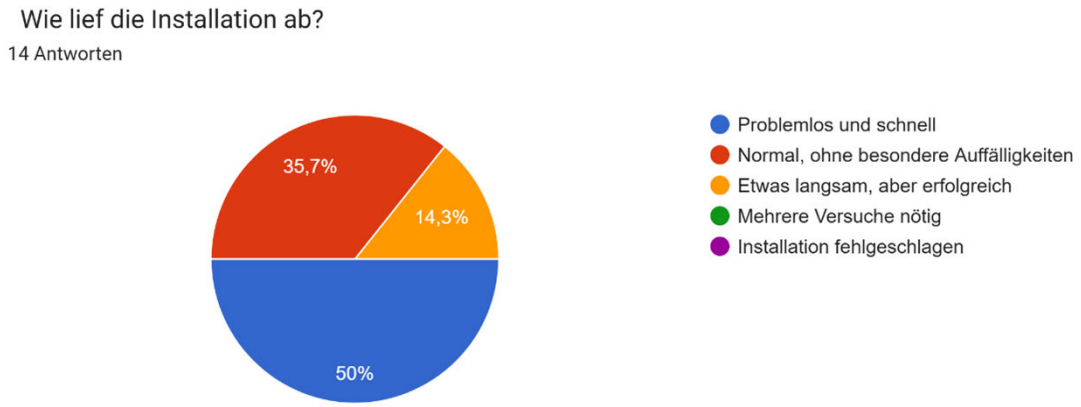

Abbildung 22: iOS- und Android-Installation

Die zweite Umfrage konzentrierte sich auf die technische Evaluation der MindSpark-Beta-Version. Die plattformübergreifende Entwicklung mit Flutter offenbarte dabei Unterschiede zwischen iOS- und Android-Implementierungen, die direkte Auswirkungen auf die Nutzererfahrung hatten. Die Betriebssystemverteilung zeigte eine klare Android-Dominanz mit 64,3 % der Teilnehmenden, während 35,7 % iOS-Geräte verwendeten (siehe Abbildung 22). Diese Verteilung spiegelt die praktischen Herausforderungen bei der iOS-Beta-Distribution wider. Die Installationsanalyse offenbarte deutliche plattformspezifische Unterschiede, die direkte Auswirkungen auf die Entwicklungsstrategie hatten. Während 50 % der Teilnehmenden von einer problemlosen und schnellen Installation berichteten, zeigten sich bei näherer Betrachtung deutliche Unterschiede zwischen den Betriebssystemen.

iOS-Installation stellte sich als erheblich komplexer dar als zunächst angenommen. Die 35,7 % der Teilnehmenden, die eine normale Installation ohne besondere Auffälligkeiten meldeten, verdeckten die tatsächlichen technischen Herausforderungen. iOS-Nutzer mussten ihre Geräte physisch mit dem Entwicklungslaptop verbinden, da Apples Sicherheitsarchitektur die direkte Installation von Beta-Versionen ohne App-Store-Distribution verhindert. Der Installationsprozess über Xcode erforderte mehrere Schritte: Gerätekopplung, Entwicklerzertifikat-Vertrauen und Build-Übertragung. Diese Komplexität führte zu längeren Wartezeiten, während der Code kompiliert und auf das Gerät übertragen wurde. Android-Installation verlief hingegen deutlich reibungsloser. Die direkte APK-Installation ermöglichte eine schnellere und autonomere Testumgebung. Die 14,3 % der Teilnehmenden, die eine langsame, aber erfolgreiche Installation meldeten, bezogen sich primär auf ältere Android-Geräte mit begrenztem Speicher oder langsameren Prozessoren (siehe Abbildung 22). Diese Erkenntnisse beeinflussten die Entwicklungspriorisierung derart, dass Android als primäre Entwicklungsplattform etabliert wurde, während iOS-spezifische Optimierungen für spätere Entwicklungsphasen zurückgestellt wurden.

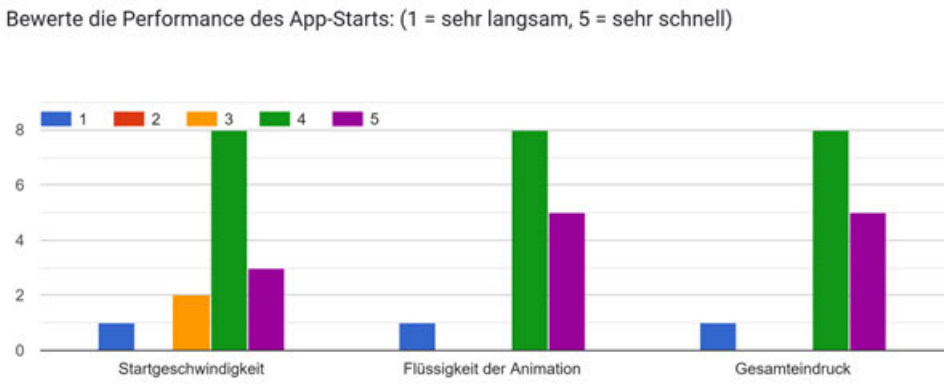

Abbildung 23: Perfomance-Bewertung

Die Performance-Bewertung des App-Starts ergab aufschlussreiche Erkenntnisse über die Flutter-Engine-

Optimierung auf verschiedenen Plattformen. Die Startgeschwindigkeit wurde mehrheitlich positiv bewertet, wobei die Bewertungen zwischen den Plattformen variierten. Startgeschwindigkeit erhielt überwiegend Bewertungen von 4–5 Punkten, was die erfolgreiche Implementierung der Flutter-Engine-Optimierungen bestätigt. Die clientseitige XP-Berechnung über `AppConstants. calculateTaskXP()` und die asynchrone Supabase-Datenbankverbindung trugen zur verzögerungsfreien App-Initialisierung bei. Flüssigkeit der Animation wurde ebenfalls überwiegend positiv bewertet. Die Implementierung des `CustomPainter` mit 60-FPS-Animationen für den Timer-Fortschrittsring erwies sich als erfolgreich. Die optimierten Berechnungen verhinderten Frame-Drops auch auf älteren Geräten. Gesamteindruck des App-Starts spiegelte die erfolgreiche Integration der verschiedenen Services wider. Die Stream-basierte Architektur für Echtzeit-Updates und die Provider-basierte State-Management-Koordination funktionierten ohne merkbare Verzögerungen.

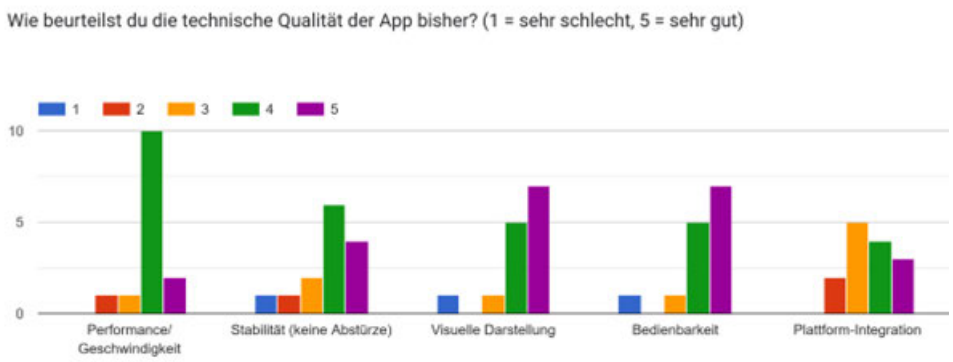

Abbildung 24: Bewertung technischer Qualität

Die umfassende Bewertung der technischen Qualität lieferte wichtige Erkenntnisse für die finale Implementierung verschiedener Systemkomponenten. Performance und Geschwindigkeit erhielten durchgehend hohe Bewertungen (4–5 Punkte), was die Architekturentscheidungen validierte. Die hierarchische Aufgaben-Datenstruktur mit optimiertem Cache-Mechanismus reduzierte Datenbankabfragen erheblich, während die asynchrone Verarbeitung den UI-Thread entlastete. Systemstabilität wurde ebenfalls positiv bewertet. Die Implementierung der Race-Condition-Vermeidung zwischen verschiedenen Screens durch explizite Textvergleiche (`if (notesController.text != moodService.notes)`) verhinderte erfolgreich Endlosschleifen und App-Abstürze. Die zweistufigen Sicherheitsmechanismen bei kritischen Aktionen wie Aufgabenlöschung trugen zur wahrgenommenen Stabilität bei. Visuelle Darstellung profitierte von der konsistenten Implementierung der Barrierefreiheits-Features. Der Dark Mode mit systemweiter Farbschema-Anpassung und die WCAG-2.1-konformen (W3C, 2025) Kontraste führten zu hohen Bewertungen in diesem Bereich. Bedienbarkeit wurde durch die Wiederverwendung bekannter UI-Patterns optimiert. Die Entscheidung, dieselbe Eingabemaske für Aufgabenerstel-

lung und -bearbeitung zu verwenden, reduzierte die kognitive Belastung und verbesserte die Bewertungen. Plattform-Integration zeigte erwartungsgemäß Unterschiede zwischen iOS und Android. Während Android-Nutzer die Integration durchgehend positiv bewerteten, zeigten iOS-Nutzer gemischte Reaktionen, was die plattformspezifischen Implementierungsherausforderungen widerspiegelt.

Diese Beta-Testing-Ergebnisse bestätigten die grundlegenden Architekturentscheidungen und identifizierten gleichzeitig spezifische Verbesserungsbereiche für die finale Implementierung. Die plattformspezifischen Unterschiede führten zu einer Fokussierung auf Android als primäre Zielplattform, während die allgemein positive Performance-Bewertung die technische Stabilität der Flutter-basierten Lösung validierte.

Auswertung des Usability-Testing

Die praktischen Tests mit 12 Teilnehmenden vor Ort ergänzten die Online-Umfragen durch direkte Beobachtung der Nutzerinteraktionen und Think-Aloud-Protokolle. Während die Kernfunktionalitäten der App erfolgreich validiert wurden, offenbarten sich durch die intensive Nutzung auch spezifische Problembereiche, die in der theoretischen Planung nicht vollständig berücksichtigt werden konnten. Dennoch sind bei der Testdurchführung auch einige Kritikpunkte, offene Wünsche und wertvolle Verbesserungsvorschläge festgestellt worden, die zusammengefasst und festgehalten wurden.

Der wohl häufigste technische Kritikpunkt betraf das responsive Design auf verschiedenen Gerätetypen. Sieben der zwölf Teilnehmer stellten fest, dass Textinhalte auf unterschiedlichen Bildschirmgrößen abgeschnitten wurden. Besonders betroffen waren längere Aufgabenbeschreibungen und Motivationsnachrichten, die auf kleineren Android-Geräten nicht vollständig dargestellt wurden. Diese Problematik trat sowohl bei älteren Samsung-Geräten als auch bei neueren Smartphones mit abweichenden Seitenverhältnissen auf. Die Flutter-Implementierung erwies sich als unzureichend optimiert für die Vielfalt der Android-Bildschirmgrößen, was die Benutzerfreundlichkeit erheblich beeinträchtigte.

Als zweithäufigste Rückmeldung kam von zwei iOS-Probanden die fehlende Funktionalität der täglichen Erinnerungen. Während Android-Nutzer problemlos Push-Benachrichtigungen erhielten, blieben diese bei iPhone-Nutzern vollständig aus. Die Ursache lag in den restriktiven Berechtigungsanforderungen von iOS für Beta-Versionen, die über Xcode installiert wurden. Da die App nicht über den App Store bezogen wurde, verweigerte das System automatisch Notification-Berechtigungen. Für eine produktive iOS-Version wäre eine ordnungsgemäße App-Store-Distribution erforderlich, um diese Kernfunktionalität zu gewährleisten.

Von vier Teilnehmern wurde die begrenzte Rückgängig-Funktionalität kritisiert. Während Aufgaben-

löschungen durch Bestätigungsdialoge geschützt waren, fehlten Rückgängig-Mechanismen bei anderen Aktionen wie Prioritätsänderungen, Timer-Abbrüchen oder Stimmungseinträgen. Besonders neurodivergente Nutzer empfanden dies als stressig, da sie befürchteten, durch impulsive Aktionen wichtige Daten zu verlieren. Die Implementierung einer umfassenden Undo-Funktionalität würde das Vertrauen in die App erheblich stärken.

Drei Probanden äußerten den Wunsch nach einer grafischen Visualisierung der Stimmungsverläufe. Die implementierte statistische Auswertung mit Durchschnittswerten wurde als zu abstrakt empfunden. Nutzer wünschten sich Liniendiagramme oder andere visuelle Darstellungen, um Muster und Trends in ihrer emotionalen Verfassung erkennen zu können. Die `MoodHistoryService`-Klasse bietet zwar die Datengrundlage für solche Visualisierungen, jedoch fehlt die entsprechende UI-Komponente für die grafische Aufbereitung.

Ebenfalls von drei Probanden wurde die fehlende Passwort-Wiederherstellung als problematisch eingestuft. Während dieser Aspekt bei der Entwicklungspriorisierung als weniger kritisch eingestuft wurde, zeigten die Tests, dass auch technikaffine Nutzer gelegentlich ihre Zugangsdaten vergessen. Die aktuelle Lösung, sich neu registrieren zu müssen, führt zum Verlust aller Nutzerdaten und wurde als inakzeptabel empfunden.

Fünf Teilnehmer kritisierten die begrenzten Kategorisierungsoptionen für Aufgaben. Das System bietet lediglich Prioritätsstufen (niedrig, mittel, hoch) und Statuskategorien (offen, in Bearbeitung, abgeschlossen, archiviert). Nutzer wünschten sich zusätzliche Klassifizierungen nach Kontext (Universität, Zuhause, Arbeit) oder thematischen Gruppierungen (Lernen, Haushalt, Freizeit). Die aktuelle Datenmodell-Struktur würde solche Erweiterungen prinzipiell unterstützen, jedoch fehlt die entsprechende UI-Implementierung.

Einige Probanden bemängelten die unzureichenden Such- und Filterfunktionen in der Aufgabenübersicht. Während die `TaskService`-Implementierung grundlegende Filterungen nach Status unterstützt, fehlen erweiterte Suchoptionen nach Titel, Beschreibung oder Erstellungsdatum. Bei einer größeren Anzahl von Aufgaben wird die Navigation in der Liste unübersichtlich, was die Produktivität beeinträchtigt.

Diese identifizierten Schwachstellen spiegeln typische Herausforderungen bei der Entwicklung von Prototypen mit begrenzten Ressourcen wider. Während die Kernfunktionalitäten erfolgreich implementiert wurden, zeigen die Kritikpunkte Bereiche auf, in denen weitere Entwicklungsiterationen notwendig wären, um eine vollständig produktionstaugliche Anwendung zu erreichen. Die Rückmeldungen bestätigen gleichzeitig die grundsätzliche Funktionsfähigkeit der App und liefern wertvolle Erkenntnisse für zukünftige Verbesserungen.

6.3 Erfüllung der Anforderungen

Dieses Kapitel prüft die Umsetzung der Anforderungen aus den Abschnitten 3.2 und 3.3. Die App muss bestimmte Funktionen bereitstellen und zeigt damit, inwieweit die geplanten Funktionen tatsächlich in der App realisiert wurden. Die funktionalen Anforderungen bilden den Kern der Anwendung. Daneben stehen die nicht-funktionalen Anforderungen, zu denen Leistung, Sicherheit und Bedienbarkeit gehören. Die Bewertung folgt klaren Kriterien, wobei der vorhandene Funktionsumfang mit den ursprünglichen Anforderungen verglichen wird. Einige der Funktionen wurden vollständig implementiert, während andere nur teilweise oder gar nicht implementiert wurden. An wenigen Stellen waren Kompromisse nötig, da die Stabilität und Nutzerfreundlichkeit Vorrang hatten.

6.3.1 Funktionale Anforderungen

Die Basisfunktion der Registrierung (siehe Tabelle 1) wurde vollständig umgesetzt. Nutzer können erfolgreich ein neues Konto erstellen mit allen geforderten Datenfeldern:

- **Benutzername**: Als optionales Feld implementiert, ermöglicht personalisierte Ansprache.
- **E-Mail-Adresse**: Dient als eindeutiger Identifier und Login-Kennung
- **Passwort**: Sichere Eingabe mit Validierung

Die Registrierung erfolgt innerhalb der geforderten zwei Minuten, und neue Benutzerkonten werden erfolgreich in der Supabase-Datenbank angelegt. Nach erfolgreicher Registrierung werden Nutzer automatisch zur Startseite weitergeleitet.

Die Anforderung FA01b wurde teilweise umgesetzt. Diese stellte während der Entwicklung eine besondere Herausforderung dar. Der ursprüngliche Ansatz mit Supabase Auth's eingebauter E-Mail-Bestätigung, bei der Nutzer ihre E-Mail-Adresse durch Klick auf einen zugesandten Link verifizieren mussten, führte zu mehreren Problemen. Die E-Mail-Zustellung verzögerte sich extrem oder es kam zu vollständigen Ausfällen der Bestätigungsnachrichten. Zusätzlich wurden viele der Mails von den Spam-Filtern abgefangen oder es kam zu einer fehlerhaften Token-Validierung bei der Bestätigung. Außerdem störte es die User Experience. Nutzer mussten zwischen App und E-Mail-Client wechseln, was den Registrierungsflow unterbrach. Nach mehreren Durchläufen wurde entschieden, die E-Mail-Bestätigung zu deaktivieren bzw. die Implementierung komplett zu löschen. Die aktuelle Lösung ist nun, dass die App weiterhin die E-Mail-Adresse auf Format und Eindeutigkeit validiert, jedoch auf den Bestätigungslink verzichtet. Dies stellt einen Kompromiss zwischen Benutzerfreundlichkeit und Sicherheit dar, der für den Anwendungskontext einer Lernhilfe-App vertretbar ist. Die Anforderung FA01c wurde vollständig

umgesetzt:

- **Passwort anzeigen/verstecken**: Toggle-Button mit Auge-Icon ermöglicht das temporäre Anzeigen des eingegebenen Passworts
- **Mindestlänge**: Validierung auf mindestens acht Zeichen

Alle Funktionen der Anforderung Anforderung FA01 sind implementiert und funktionsfähig. Die Anpassung bei der E-Mail-Verifizierung stellt einen bewussten Kompromiss dar, der die Benutzerfreundlichkeit priorisiert, ohne die Grundsicherheit zu gefährden.

FA02 – Anmeldung

Die Anmeldefunktion wurde mit dem Ziel entwickelt, einen schnellen und zugleich sicheren Zugriff auf die App zu gewährleisten. Die Anforderung FA02a wurde vollständig erfüllt. Nutzer können sich erfolgreich mit ihren Zugangsdaten anmelden. Nach erfolgreicher Authentifizierung wird eine gültige Session erzeugt und in der Datenbank gesichert. Die Startseite lädt fehlerfrei und alle benutzerspezifischen Inhalte werden korrekt aus der Datenbank abgerufen. Bei falschen Zugangsdaten erscheint eine klare Fehlermeldung. Die Anmeldung erfolgt stets innerhalb der geforderten 20 Sekunden, meist jedoch deutlich schneller (unter 10 Sekunden). Rückmeldungen sind so formuliert, dass sie verständlich und nutzerfreundlich wirken, um Frustration zu vermeiden.

Die Anforderung FA02b wurde nicht implementiert. Die Entscheidung beruht auf einer Priorisierung der Entwicklungsressourcen. Da die Zielgruppe primär aus Schülern und Studierenden besteht, die in der Regel mit digitalen Tools vertraut sind, wurde die Passwort-Wiederherstellung als weniger kritisch eingestuft. Nutzer, die ihr Passwort vergessen, müssen sich derzeit neu registrieren. Für eine zukünftige Version ist die Implementierung jedoch vorgesehen.

FA02c wurde bewusst nicht umgesetzt. Während der Usability-Tests wurde sie von mehreren Teilnehmern als unnötig empfunden. Stattdessen speichert die App die Session automatisch. Erst nach einer expliziten Abmeldung ist eine erneute Anmeldung erforderlich. Dieses Vorgehen reduziert die Komplexität der Einstellung, was insbesondere für neurodivergente Nutzer eine Erleichterung darstellt.

Insgesamt erfüllt die Anmeldefunktion alle MUSS-Anforderungen und gewährleistet einen sicheren sowie schnellen Zugang zur App. Die nicht umgesetzten KANN-Anforderungen beeinträchtigen die Kernfunktionalität nicht, sondern wurden basierend auf Nutzerfeedback und Entwicklungsprioritäten bewusst ausgelassen.

FA03 – Aufgabenerstellung

Die Aufgabenverwaltung bildet das zentrale Funktionselement der App und ist damit entscheidend für die gesamte Nutzererfahrung. Nach Entwicklungs- und Testphasen wurden alle in diesem Bereich definierten Anforderungen (FA03a bis FA03e) vollständig realisiert. Bei der Anforderung FA03a müssen Nutzer lediglich einen Titel angeben, um eine neue Aufgabe anzulegen. Alle weiteren Angaben wie Beschreibung, Prioritäten, geschätzte Bearbeitungszeit oder Fälligkeitsdatum sind optional und können bei Bedarf ergänzt werden. Dieser Ansatz senkt die Einstiegshürde deutlich und ermöglicht es, Aufgaben schnell und unkompliziert zu erfassen. Die automatische Generierung dieser Aufgaben erfolgt systemseitig und ohne zusätzlichen manuellen Aufwand, wodurch der Nutzerfluss nicht unterbrochen wird. Besonders hervorzuheben ist die Implementierung wiederkehrender Aufgaben, die flexibel auf individuelle Routinen eingeht:

- Tägliche Wiederholungen für Routinenaufgaben
- Wöchentliche Intervalle für wiederkehrende Lerneinheiten
- Benutzerdefinierte Konfiguration mit Auswahl bestimmter Wochentage

Auch die Anforderung FA03b orientiert sich am Prinzip der Einfachheit. Durch diese Architektur bleibt der Workflow stets flüssig und die App reagiert ohne Verzögerung. Der Ablauf gestaltet sich klar und effizient:

1. Die Aufgabe wird innerhalb von 20 Sekunden erstellt.
2. Sie erscheint sofort sichtbar in der Aufgabenliste.
3. Im Hintergrund synchronisiert sich die Supabase-Datenbank asynchron und ohne Unterbrechung der Nutzerinteraktion.

FA03c wurde als SOLL-Anforderung definiert und vollständig umgesetzt. Nutzer können ihre Aufgaben klar in vier Zustände einteilen: *offen, in Bearbeitung, abgeschlossen* oder *archiviert*. Der Wechsel erfolgt entweder manuell durch ein Long-Press oder automatisch durch die Timer-Funktion. Diese Struktur sorgt für Transparenz und macht den Fortschritt jederzeit sichtbar. Eine weitere Optimierung liegt in der Bearbeitung (FA03d). Statt eine gesonderte Bearbeitungsoberfläche zu entwickeln, wird die bereits für die Erstellung von Aufgaben genutzte Oberfläche verwendet. Diese können angepasst und anschließend gespeichert werden. Durch die Wiederverwendbarkeit der bekannten Oberfläche werden die Konsistenz gewahrt, die Bedienung vereinfacht und die kognitive Belastung reduziert, da keine neuen Interaktionsmuster gelernt werden müssen.

Die Anforderung FA03e wurde mit einem zweistufigen Sicherheitsmechanismus versehen. Zunächst löst der Nutzer die Aktion aus, anschließend erscheint ein Bestätigungsdialog. Erst nach dieser bewussten Rückmeldung wird die Aufgabe endgültig entfernt und die Datenbank synchron aktualisiert. Damit werden versehentliche Löschvorgänge zuverlässig vermieden.

Über die Kernanforderung hinaus ergaben sich während der Entwicklung zusätzliche Herausforderungen. Besonders die Teilaufgaben-Funktionalität spielte dabei eine wichtige Rolle. Viele Aufgaben setzten sich aus mehreren Teilschritten zusammen, weshalb ein hierarchisches Datenmodell eingeführt wurde. Hauptaufgaben speichern Referenzen auf ihre Teilaufgaben, während der `TaskService` einen optimierten Cache-Mechanismus nutzt. Dieser Ansatz reduziert Datenbankabfragen erheblich, da Teilaufgaben nur bei Änderungen neu geladen werden. Gleichzeitig wird die Konsistenz zwischen Haupt- und Teilaufgaben automatisch geprüft. Sind alle Teilaufgaben abgeschlossen, markiert die App auch die übergeordnete Aufgabe als erledigt.

Die erfolgreiche Umsetzung aller Kernfunktionen bestätigt den architektonischen Ansatz der App. Gleichzeitig zeigen die Erweiterungen wie die Teilaufgaben-Funktion, dass die Lösung flexibel auf reale Nutzungsszenarien eingeht. Für zukünftige Versionen wären zusätzliche Funktionen denkbar, etwa erweiterbare Filteroptionen. Schon jetzt bietet die Aufgabenverwaltung eine stabile, nutzerorientierte Grundlage, die sich an den Bedürfnissen der Zielgruppe orientiert.

FA04 - Timer

Die Anforderung FA04 wurde größtenteils erfolgreich umgesetzt. Sie gewährleistet strukturierte Arbeitseinheiten für neurodivergente sowie neurotypische Nutzer. Wie in Abschnitt 3.2.4 spezifiziert, umfasst die Timer-Funktion sowohl einfache Timer-Modi als auch komplexere Pomodoro-Zyklen.

Die Basisfunktion, Anforderung FA04a, wurde vollständig realisiert:

- **Basis-Timer**: Nutzer kann direkt mit einem Klick auf die Aufgabe den Timer starten
- **Timer-Anpassung**: Zeitintervalle lassen sich zwischen 5, 25, 45 und 60 Minuten auswählen.
- **Verzögerung**: Timer startet verzögerungsfrei und behält die Verbindung zur ursprünglichen Aufgabe bei
- **Speicherung**: Timer-Session wird beim Start in der Datenbank angelegt

Die Anforderung FA04b bietet erweiterte Konfigurationsmöglichkeiten. Die Standardintervalle von 25 Minuten Arbeitszeit und 5 Minuten Pause können in den Einstellungen individuell angepasst werden. Der Übergang zwischen Arbeits- und Pausenphasen wird durch In-App-Benachrichtigungen über jeden

Übergang informiert. Ein kritischer Aspekt bei der Pomodoro-Implementierung war die Zustandsverwaltung. Die App muss zwischen verschiedenen Time-Modi (einfacher Timer vs. Pomodoro), Phasen und Zyklen unterscheiden. Die Implementierung nutzt ein Enum-System für TimerPhase und verwaltet Zustände wie `currentPhase, pomodoroCycleCount` und `totalPomodoroCyclesForTask`. Besonders komplex war dabei die Behandlung des Übergangs zwischen verschiedenen Pomodoro-Zyklen und die Berechnung der XP-Vergabe basierend auf der tatsächlich geleisteten Arbeitszeit.

Die Anforderung FA04c konnte nur teilweise umgesetzt werden. Die ursprünglich geplante Implementierung sah vor, während aktiver Timer-Session alle externen App-Benachrichtigungen zu unterdrücken und ablenkende System-Elemente auszublenden. Nach intensiver Entwicklungsarbeit stellte sich heraus, dass die Funktion erheblich komplexer ist als zunächst angenommen. Die Integration mit den nativen Android- und iOS-Notification-Management-Systemen erforderte plattformspezifische Implementierungen mit tiefen Eingriffen in die Betriebssystemfunktionen. Aufgrund zeitlicher Beschränkungen und der Notwendigkeit zusätzlicher Berechtigungen wurde diese Funktion aus der aktuellen Version entfernt. Stattdessen wurde der Fokus auf motivierende Nachrichten und visuelle Hinweise gelegt, die den Nutzer während der Timer-Session bei der Konzentration unterstützen.

Der visuelle Fortschrittsring aus der Anforderung FA04d wurde erfolgreich implementiert und bietet eine intuitive Darstellung des Timerverlaufs. Der Ring füllt sich kontinuierlich mit einer Farbverlauf-Animation und aktualisiert sich sekündlich. Die Implementierung nutzt einen `CustomPainter` mit optimierten Berechnungen für flüssige Animationen bei 60 FPS.

Alle MUSS-Anforderungen wurden erfüllt. Die SOLL-Anforderungen FA04b und FA04d sind vollständig umgesetzt, während FA04c bewusst modifiziert wurde, um eine praktikable Lösung zu gewährleisten. Die Timer-Implementierung bietet eine stabile Grundlage für produktives Arbeiten und fügt sich in das Gesamtkonzept der neurodivergenzfreundlichen Lernhilfe-App ein.

FA05 – Fortschritt

Nach der Analyse der Fortschrittsanzeige-Implementierung wurde die Anforderung FA05 vollständig erfüllt. Die prozentuale Fortschrittsanzeige wurde vollständig umgesetzt und ist sowohl auf der Startseite als auch in der Aufgabenliste platziert. Die Progress-Bar zeigt den aktuellen Bearbeitungsfortschritt erledigter Aufgaben an. Die Implementierung nutzt dabei sowohl die `getCompletedTaskCount()`- als auch die `getTotalTasksCount`-Methoden des `TaskService`, um eine präzise Prozentangabe zu berechnen. Durch die Integration in das Widget-System aktualisiert sich die Anzeige automatisch bei jeder Statusänderung von Aufgaben. Die Anforderung FA05b bietet pro Fortschrittsbereich abwechslungsrei-

che Motivationsnachrichten in der Progress-Bar an.

Die Echtzeit-Datenaktualisierung stellte eine weitere Herausforderung dar. Diese implementiert ein Mixin, das zu der `HomeScreen`-Klasse hinzugefügt wurde, um ihr zusätzliche Funktionalitäten zu geben. Der `AutomaticKeepAliveClientMixin` wurde dem `HomeScreen` hinzugefügt, um den State auch bei Tab-Wechsel zu erhalten. Gleichzeitig muss die Fortschrittsanzeige jedoch auf Änderungen in anderen Screens reagieren. Die Lösung nutzt einen Stream-basierten Ansatz. Der `taskService.tasksStream.listen()` sorgt dafür, dass sich die UI automatisch neu aufbaut, sobald sich Aufgaben-Daten ändern, unabhängig davon, in welchem Screen die Änderung vorgenommen wurde. Zudem muss die Progress-Bar korrekt reagieren, wenn keine Aufgaben vorhanden sind (Division durch Null vermeiden) oder wenn alle Aufgaben bereits abgeschlossen sind. Die Implementierung prüft diese Edge Cases explizit und zeigt entsprechende Fallback-Inhalte an, wie das `buildEmptyTasksState()`-Widget, das eine nutzerfreundliche Nachricht mit motivierenden Elementen darstellt.

FA06 – Belohnungssystem

Die Anforderung FA06 wurde größtenteils erfolgreich realisiert. Die XP-Vergabe wurde vollständig nach den definierten Kriterien implementiert und sorgt für eine transparente, nachvollziehbare Belohnungsstruktur. Die Grundlage ist eine automatische Berechnung, die sich an der Dauer der jeweiligen Aufgabe orientiert:

- Kurze Aufgaben unter 15 Minuten: 10 XP
- Mittlere Aufgaben zwischen 15 und 40 Minuten: 20 XP
- Lange Aufgaben über 40 Minuten: 40 XP

Die Berechnung erfolgt clientseitig über `AppConstants.calculateTaskXP()` für eine verzögerungsfreie Nutzererfahrung. XP werden sowohl für Timer-Sessionen als auch für direkte Aufgabenabschlüsse vergeben. Alle XP werden sofort in der Datenbank gespeichert, sodass Fortschritte jederzeit konsistent nachvollzogen werden können.

Die Anforderungen FA06b und FA06c wurden vollständig umgesetzt, allerdings mit Anpassungen basierend auf Nutzerfeedback. Die ursprünglich geplanten sprachlichen Level-Bezeichnungen, wie *Level 5 – Lernender*, wurden nach dem Usability-Testing entfernt, da sie von den Testern als nicht notwendig empfunden wurden. Stattdessen wird nur die numerische Level-Anzeige verwendet. Die visuelle Level-Up-Rückmeldung wurde ebenfalls angepasst. Statt einer Animation wird nur noch eine Textbenachrichtigung angezeigt, da sowohl neurodivergente als auch neurotypische Tester die Animation als

überflüssig betrachten und ein einfacher Text genügend erscheint.

Die Anforderung FA06d wurde nur teilweise realisiert. Das grundlegende Achievement-System ist vollständig implementiert und bietet verschiedene Kategorien wie Level- und Perfect-Day-Achievements und spezielle Meilenstein-Belohnungen. Die wöchentliche Challenges wurde als SOLL-Anforderung aus zeitlichen Gründen nicht implementiert. Da sich diese Funktionalität als komplexer erwies als ursprünglich angenommen und die Entwicklungsprioritäten auf die Kernfunktionen fokussiert wurden.

Die Anforderung FA06e wurde nicht implementiert. Es waren mehrere Streak-Typen geplant (Aufgaben-Streak, Timer-Streak etc.), für die Freeze-Days als Unterstützungsmechanismus gedacht waren. Da letztendlich nur die Login-Streak-Logik implementiert wurde, die lediglich das tägliche Öffnen der App erfordert, wurde die Freeze-Days-Logik als nicht notwendig erachtet. Die UI-Komponenten für Freeze-Days sind technisch vorhanden, aber die Backend-Logik ist nicht aktiv. Diese Funktion bleibt für zukünftige Versionen vorgesehen, wenn komplexere Streak-Systeme eingeführt werden.

Ein komplexer Aspekt war die Achievement-Notification-Pipeline. Da mehrere Achievements gleichzeitig freigeschaltet werden können, war die ursprüngliche Variante eine Queue-basierte Lösung mit `pendingNotification` und `currentNotification`. Die Benachrichtigungen werden sequenziell mit Delays angezeigt (500 ms zwischen Achievement-Popups), um Überlappungen zu vermeiden. Jedoch kam es zu dem Problem, dass einige Achievement-Benachrichtigungen gar nicht angezeigt wurden. Eine einfache Lösung war es, dass bei mehreren Achievement-Benachrichtigungen nach Bestätigung des Nutzers die nächste Nachricht angezeigt wird. So hat der Nutzer die volle Kontrolle.

Alle MUSS-Anforderungen wurden erfüllt. Die SOLL-Anforderung FA06d wurde teilweise umgesetzt, während FA06e bewusst für zukünftige Versionen zurückgestellt wurde. Die Anpassungen basierend auf Usability-Testing-Feedback verbessern die Nutzererfahrung für die Zielgruppe neurodivergenter Personen und zeigen die Wichtigkeit nutzerorientierter Entwicklung.

FA07 - Stimmungstracker

Nach einer detaillierten Analyse lässt sich feststellen, dass die Anforderung FA07 größtenteils erfolgreich umgesetzt wurde. Die App ermöglicht Nutzern eine optionale und benutzerfreundliche Dokumentation ihrer emotionalen Verfassung, wie in Abschnitt 3.2.7 spezifiziert. Die Anforderung FA07a wurde vollständig implementiert und ist sowohl auf dem `MoodScreen` als auch über die `HomeScreen`-Integration verfügbar. Nutzer können ihre aktuelle Stimmung durch ein einfaches Tippen auf ein Emoji auswählen. Das System bietet fünf verschiedene Stimmungsoptionen mit entsprechenden Emoji-Darstellungen und klaren Bezeichnungen. Die Auswahl wird sofort visuell hervorgehoben und kann binnen weniger Sekun-

den abgeschlossen werden. Die Daten werden automatisch in der Supabase-Datenbank gespeichert und durch visuelles Feedback bestätigt.

Die Anforderung FA07b wurden vollständig umgesetzt und bieten umfassende Bewertungsmöglichkeiten. Das System implementiert drei Hauptkategorien:

- Energielevel
- Fokuslevel
- Stresslevel

Die `MoodService`-Logik konvertiert diese Prozentwerte automatisch in 5-stufige Skalen für die Datenbanksspeicherung über die Methoden `energyLevelForModel, focusLevelForModel` und `stressLevelForModel`. Zusätzlich steht ein Notizen-Textfeld zur Verfügung, das bis zu vier Zeilen Text für kontextuelle Informationen ermöglicht. Die Slider-Komponenten reagieren verzögerungsfrei auf Touch-Eingaben und bieten sowohl visuelle als auch textuelle Rückmeldungen über die aktuellen Werte.

Diese Anforderung FA07c wurde nur teilweise erfüllt. Anstatt der ursprünglich geplanten Liniendiagramm-Darstellung wurde ein statistisches Auswertungssystem implementiert. Der `MoodHistoryScreen` bietet umfassende Statistiken über die `MoodHistoryService`-Klasse, die Durchschnittswerte für alle erfassten Kategorien berechnet. Das System zeigt durchschnittliche Stimmung, Produktivität, Energie-, Fokus- und Stresslevel an. Die Daten können nach Wochen- oder Monatsperioden gefiltert werden, und die `MoodStatistics`-Klasse stellt aggregierte Informationen bereit, die als Kompromiss für die ursprünglich geplante grafische Darstellung dienen.

Eine technische Herausforderung lag in der Datenkonvertierung zwischen UI-Repräsentation und Datenbankmodell. Der Stimmungstracker verwendet eine hybride Skalierung. Die Benutzeroberfläche arbeitet mit intuitiven 0–100 %-Slidern, während die Datenbank eine 5-stufige Skala erwartet. Der `MoodService` implementiert komplexe Konvertierungslogik über `convertModelValueToPercentage()` und die entsprechenden `ForModel`-Getter. Diese Doppelkonvertierung war notwendig, um sowohl benutzerfreundliche Eingabemöglichkeiten als auch effiziente Datenbankspeicherung zu gewährleisten, führte jedoch zu erhöhter Komplexität bei der Datenvalidierung und -synchronisation.

Ein weiterer komplexer Aspekt war die State-Management-Koordination zwischen mehreren Screens. Der `MoodService` muss Zustandsänderungen zwischen `MoodScreen`, `HomeScreen` (für schnelle Stimmungserfassung) und `MoodHistoryScreen` synchronisieren. Die Implementierung nutzt Provider-basierte Listener mit `notifyListeners()`, aber die Herausforderung bestand darin, Race Conditions zu vermei-

den, wenn mehrere UI-Komponenten gleichzeitig auf Stimmungsdaten zugreifen. Eine Race Condition entsteht immer dann, wenn mehrere unabhängige Abläufe (Threads, Prozesse etc.) zeitgleich auf denselben Zustand zugreifen und dessen Wert lesen bzw. ändern (Zbarcea & Tudose, 2024). Wenn z. B. sowohl das `MoodScreen` als auch das `MoodHistoryScreen` über den gemeinsamen `MoodService` Daten wie Stimmungswert oder Notizen ändern, kann eine Race Condition entstehen. Aktualisieren zwei Screens gleichzeitig etwa den Text einer Notiz, so kann jeweils ein Rückruf das andere erneut triggern. Ohne Abgleich kann sich daraus eine Endlosschleife ergeben. Zur Vermeidung solcher Endlosschleifen wurde eine Abfrage eingebaut, die nur ein Update auslöst, wenn sich der Text tatsächlich geändert hat (`if (notesController.text != MoodService.notes)`).

NF08 – Einstellungen

Die Anforderung FA08 wurde größtenteils erfolgreich umgesetzt. Die Anforderung FA08a wurde teilweise implementiert. Das System bietet Optionen für reduzierte Animationen, größere Schrift, hohen Kontrast und Vibration-Modi. Die `AccessibilitySettings`-Klasse verwaltet fünf Hauptkategorien: `reducedAnimations, highContrast, largeText, onlyVibration` und `softSounds`. Die Implementierung über `AccessibilityService` ermöglicht sowohl Batch-Updates über `saveUserSettings` als auch einzelne Anpassungen über `updateSingleSetting`. Wie bereits bei FA06 erwähnt, wurden Animationen als überflüssig empfunden und daher standardmäßig deaktiviert. Die Textvergrößerung wird über die `main.dart` mit `TextScaler.linear(textScaleFactor)` umgesetzt, wobei ein Faktor von 1,3 für größere Schrift verwendet wird.

Das Benachrichtigungssystem (FA08b) wurde vollständig implementiert und bietet umfassende Konfigurationsmöglichkeiten. Der `NotificationService` verwaltet sowohl lokale Push-Benachrichtigungen über `flutter_local_notifications` als auch app-interne Einstellungen. Das System unterstützt Timer-Erinnerungen, tägliche Aufgaben-Reminder, Achievement-Benachrichtigungen und Streak-Erinnerungen. Die Implementierung nutzt timezone-basierte Planung für wiederkehrende Benachrichtigungen und erstellt separate Notification-Channels für verschiedene Kategorien. Ton- und Vibrationseinstellungen sind einzeln konfigurierbar und werden über die `AccessibilityService`-Integration verwaltet.

Die Anforderung FA08c wurde vollständig reallisiert und ist über die Einstellung verfügbar. Nutzer können sich sicher von der App abmelden, wobei alle lokalen Daten gelöscht und die Session beendet wird. Eine der technischen Herausforderungen lag in der Cross-Service-Koordination zwischen `Accessibility-` und `Notification-Service`. Barrierefreiheitseinstellungen beeinflussen die Art, wie Benachrichtigungen dargestellt werden (Sound/Vibration), was eine enge Integration zwischen beiden Services erforderte. Der `AccessibilityService` muss über `updateAudioSettings()` Änderungen an den

`NotificationService` weiterleiten, während gleichzeitig die Konsistenz zwischen Datenbankzustand und UI-Zustand gewährleistet werden muss. Die `triggerNotificationReminder()`-Methode koordiniert dabei verschiedene ReminderTypes (`dailyReminder, streakReminder`) mit den entsprechenden Accessibility-Präferenzen.

6.3.2 Nicht-funktionale Anforderungen

Nach der eingehenden Analyse der Cross-Platform-Kompatibilität lässt sich feststellen, dass die nichtfunktionale Anforderung NF01 erfolgreich erfüllt wurde. Die App wurde primär für Android-Geräte entwickelt, funktioniert jedoch auch auf iOS-Systemen, wie in Abschnitt 5.1 spezifiziert. Die Implementierung nutzt das Flutter-Framework mit einer gemeinsamen Dart-Codebasis für beide Plattformen. Dies gewährleistet eine einheitliche Entwicklung und native Performance durch Dart-Kompilierung. Die Unterstützung erstreckt sich auf Android 8.0+ und iOS 12.0+, womit aktuelle mobile Geräte abgedeckt werden. Der Fokus lag auf verschiedenen Android-Herstellern wie Samsung, Huawei und Google Pixel, da die meisten Testpersonen Android-Smartphones verwendeten.

Alle Kernfunktionen (Aufgabenverwaltung, Timer, Belohnungssystem, Stimmungstracking) funktionieren plattformübergreifend identisch. Eine Ausnahme bilden die Push-Benachrichtigungen auf iOS, die nicht vollständig funktionieren. Die iOS-Benachrichtigungen erfordern tiefgreifendere Berechtigungen über `DarwinInitializationSettings` mit `requestAlertPermission, requestBadgePermission` und `requestSoundPermission`, die komplexere Implementierungen benötigen als ursprünglich angenommen. Die größte technische Herausforderung lag in der plattformspezifischen Benachrichtigungsarchitektur. Der `NotificationService` implementiert sowohl Android- als auch iOS-spezifische Initialisierungspfade über `AndroidInitializationSettings` und `DarwinInitializationSettings`. Bei iOS erfordern Benachrichtigungen explizite Benutzerberechtigungen, die über komplexere Permission-Flows verwaltet werden müssen.

Die Performance-Unterschiede zwischen nativen UI-Elementen waren ebenfalls bemerkenswert. Flutters Rendering-Engine produziert konsistente 60-FPS-Performance auf beiden Plattformen, aber spezifische Widgets wie `Slider`, `TextField` oder `CustomPaint` verhalten sich auf unterschiedlichen GPU-Architekturen verschieden. Die GPU-Architektur spielt dabei eine entscheidende Rolle. Wie Studien zu nativen Anwendungen zeigen, führen unterschiedliche Hardware-Konfigurationen zu messbaren Performance-Variationen, insbesondere bei GPU-beschleunigten UI-Elementen. Android-Geräte mit ihrer Hardware zeigen dabei erwartungsgemäß größere Schwankungen als iOS-Geräte mit ihrer standardisierten Hardware-Basis (Xiao et al., 2024).

Die Implementierung spezieller Animationskomponenten wie Timer-Animationen oder Progress-Ring-Berechnungen verstärkt diese Effekte zusätzlich. Tests mit nativen Anwendungen bestätigen, dass Draw-Performance-Operationen, die mehr als 16 ms benötigen, die Flüssigkeit der Benutzeroberfläche beeinträchtigen können. Bei der App-Entwicklung wurde deshalb besondere Aufmerksamkeit auf die Optimierung dieser kritischen Komponenten gelegt. Die nicht-funktionale Anforderung NF01 wurde erfolgreich erfüllt, wobei der Schwerpunkt auf Android-Kompatibilität gelegt wurde. Die iOS-Funktionalität ist größtenteils gegeben, mit Ausnahme der Benachrichtigungsfunktionen. Die Flutter-basierte Architektur gewährleistet eine gemeinsame Codebasis mit plattformspezifischen Anpassungen, was die Wartbarkeit und Weiterentwicklung erheblich erleichtert.

NF02 - Perfomance und Responsivität

Die nicht-funktionale Anforderung NF02 wurde vollständig erfüllt. Das System gewährleistet eine flüssige und reaktionsschnelle Benutzererfahrung, die speziell auf die Bedürfnisse neurodivergenter Nutzer zugeschnitten ist. Die App implementiert eine nicht-blockierende Datenbankverbindung beim Start. Die Initialisierung erfolgt asynchron im Hintergrund, während die Benutzeroberfläche sofort verfügbar bleibt. Der `TaskService` nutzt `loadUserTasks` mit Future-basierten Operationen, die parallel zur UI-Darstellung ablaufen. Durch das Loading-State-Management mit `_isLoading`-Flags wird verhindert, dass Nutzer mit einer unresponsiven Oberfläche konfrontiert werden.

Bei Nutzern mit vielen Aufgaben wurde eine Performance-Optimierung durch Lazy Loading implementiert. Die Aufgabenliste wird nicht komplett geladen, sondern in manageable Chunks aufgeteilt. Der `TaskService` nutzt clientseitiges Caching mit separaten Variablen für Aufgabenzähler, die nur bei tatsächlichen Datenänderungen neu berechnet werden. Die Filter-Funktionalität arbeitet auf dem gecachten Datensatz, was sofortige Reaktionszeiten ermöglicht.

Häufige Datenbankzugriffe werden durch lokale Caches minimiert, wobei `_taskCompletion`-Maps als eine Art Nachschlagewerk für schnelle Lookup-Operationen dienen, sodass der aktuelle Aufgabenstatus ohne Umweg über die Datenbank abgefragt werden kann. Der `TaskService` implementiert dabei ein Write-Through-Caching, was bedeutet, dass Änderungen gleichzeitig im Cache und in der Datenbank gespeichert werden, damit beide stets synchron bleiben. Tritt beim Speichern in der Datenbank ein Fehler auf, wird der Cache automatisch auf den vorherigen Zustand zurückgesetzt, wodurch Datenkonsistenz gesichert und falsche Einträge vermieden werden.

NF03 - Offline-Funktionalität

Nach der Analyse der Offline-Funktionalität lässt sich feststellen, dass die nicht-funktionale Anforderung NF03 aus zeitlichen Gründen nicht umgesetzt wurde. Diese Anforderung war als KANN-Anforderung klassifiziert und wurde zugunsten der Kernfunktionalitäten zurückgestellt. Die Offline-Funktionalität war als zusätzliche Komfortfunktion für Situationen mit instabiler Internetverbindung vorgesehen. Das Konzept sah vor, dass Aufgaben, Timer-Daten, XP/Level-Informationen und Benutzereinstellungen lokal gespeichert und offline verfügbar bleiben. Ein visueller Offline-Indikator sollte Nutzer über den aktuellen Verbindungsstatus informieren, während online-exklusive Funktionen bei fehlender Verbindung ausgeblendet werden. Die aktuelle Architektur bietet bereits eine solide Basis für zukünftige Offline-Implementierungen. Die Service-Layer-Struktur mit lokalen Caching-Mechanismen, die Supabase-Integration mit clientseitiger Datenhaltung und die State-Management-Systeme könnten relativ nahtlos um Offline-Capabilities erweitert werden. Die vorhandenen Caching-Strategien im `TaskService` und `ProgressService` stellen bereits Grundbausteine für Offline-Datenverwaltung dar. Erweiterungen oder Komfortfunktionen, wie etwa ein Offline-Modus, werden in diesem Ansatz bewusst zurückgestellt und erst in späteren Iterationen ergänzt. Da die App primär als online-basierte Lernhilfe-Anwendung konzipiert wurde, war die Offline-Funktionalität nicht kritisch für den MVP. Unter einem MVP (Minimum Viable Product) versteht man die kleinstmögliche, funktionsfähige Version eines Produkts, die nur die zentralen Funktionen umfasst, um das eigentliche Nutzungsszenario zu validieren und frühes Feedback von Nutzern einzuholen (Keogh, 2022). Die Entwicklungszeit wurde stattdessen auf die vollständige Implementierung der Kernfunktionen und neurodivergenzspezifische Features fokussiert.

Die Analyse der Anforderungserfüllung zeigt, dass die entwickelte Lernhilfe-App die gesteckten Ziele größtenteils erreicht hat. Von den acht funktionalen Anforderungen (FA01–FA08) wurden zwei vollständig, fünft größtenteils und eine teilweise umgesetzt. Die zwei nicht-funktionalen Anforderungen konnten erfolgreich erfüllt werden, wobei der Fokus auf Android-Kompatibilität und Performance-Optimierung lag. Die Offline-Funktionalität bleibt für zukünftige Versionen relevant. Die bewussten Anpassungen basierend auf Usability-Testing-Feedback (Entfernung von Animationen, vereinfachte Level-Darstellung etc.) unterstreichen den nutzerorientierten Entwicklungsansatz. Kompromisse wie die vereinfachte Stimmungsverlauf-Darstellung oder die iOS-Benachrichtigungs-Einschränkungen zeigen realistische Priorisierung bei begrenzten Entwicklungsressourcen. Die App stellt eine solide technische Grundlage für eine neurodivergenzfreundliche Lernhilfe dar, die durch ihre Flutter-basierte Architektur und modulare Service-Struktur gut für zukünftige Erweiterungen positioniert ist.

7 Fazit

7.1 Zusammenfassung

Ziel dieser Arbeit war die Konzeption und Implementierung einer gamifizierten Lernhilfe-App, die speziell auf die Bedürfnisse neurodivergenter Nutzer eingeht. Der gewählte Entwicklungsansatz, der systematisch von der Anforderungsanalyse über die Konzeptentwicklung bis hin zur prototypischen Implementierung reichte, hat sich als geeignet erwiesen. Die Arbeit verbindet erfolgreich konzeptionelle Vorgaben mit einer lauffähigen technischen Umsetzung in Flutter. Dabei wurden insbesondere Gamification-Mechaniken wie das XP-System mit progressiver Level-Struktur, Achievement-Belohnungen und Streak-Mechanismen sowie neurodivergenzspezifische Designprinzipien wie sensorische Anpassbarkeit, kognitive Entlastung und flexible Interaktionsmuster integriert. Die konkrete Implementierung umfasst die zentralen Screens und deren Funktionalitäten, beginnend mit einem barrierefreien Onboarding, über sichere Login- und Registrierungsprozesse, eine informative Startseite mit Fortschrittsvisualisierung, eine umfassende Aufgabenverwaltung mit CRUD-Operationen, flexible Timer-Funktionen inklusive Pomodoro-Technik, ein optionales Mood-Tracking-System bis hin zu personalisierbaren Einstellungen für Barrierefreiheit. Die dazugehörigen Services, die als Logikschicht zwischen Präsentation und Datenhaltung fungieren, wurden entsprechend der serviceorientierten Architektur implementiert und in chapter 5 ausführlich dokumentiert.

Die Evaluation erfolgte anhand eines zweistufiges Testdesigns, das sowohl Online-Befragungen als auch Usability-Testing umfasste und dabei die besonderen methodischen Anforderungen für neurodivergente Nutzer berücksichtigte. Dieses zweistufige Vorgehen ermöglichte es, zunächst über Online-Umfragen mit 14 Teilnehmenden grundlegende Präferenzen und Bedürfnisse zu identifizieren, um anschließend durch gezieltes Usability-Testing mit 12 Personen in realistischen Nutzungsszenarien sowohl quantitative Bewertungen als auch qualitative Beobachtungen zu gewinnen. Die gewonnenen Erkenntnisse flossen unmittelbar in Designanpassungen zurück, was den iterativen und nutzerzentrierten Charakter des Entwicklungsprozesses unterstreicht. Das Usability-Testing lieferte dabei wertvolle Einsichten zu verschiedenen Dimensionen der Nutzererfahrung, insbesondere zur Bedienbarkeit komplexer Funktionen wie der Aufgabenerstellung und Timer-Steuerung, zur sensorischen Verträglichkeit verschiedener Interface-Elemente wie Farbschemata und Animationen sowie zur subjektiven Belastung in realitätsnahen Nutzungsszenarien, die typische Herausforderungen neurodivergenter Lernender widerspiegeln.

Bezüglich der Zielerreichung zeigen die Ergebnisse eine deutliche Tendenz zur erfolgreichen Erfüllung der definierten Kernziele. Von den acht systematisch abgeleiteten funktionalen Anforderungen, die sich

von der Benutzerauthentifizierung über die Aufgabenverwaltung bis hin zu Gamification-Mechanismen erstrecken, wurden zwei vollständig umgesetzt, fünf größtenteils, während eine Anforderung aufgrund bewusster Priorisierungsentscheidungen teilweise realisiert wurde. Die relevanten nicht-funktionalen Anforderungen konnten erfolgreich erreicht werden, wobei insbesondere die Cross-Plattform-Kompatibilität durch die Flutter-basierte Entwicklung mit einheitlicher Codebasis für Android und iOS sowie die Performance und Responsivität durch optimierte State-Management-Patterns und asynchrone Datenbankoperationen gewährleistet wurden. Diese systematische Zusammenfassung der Anforderungserfüllung, die in Abschnitt 6.3 detailliert dokumentiert ist, bestätigt eindeutig, dass das Projekt den angestrebten Scope eines funktionsfähigen MVP erfüllt hat, das als solide Grundlage für weiterführende Entwicklungsiterationen dienen kann.

Technisch ist die Umsetzung als robust zu bewerten. Die gewählte Architektur, bestehend aus einem Flutter-Frontend mit nativem Rendering, einem modularen Service-Layer nach dem Single-Responsibility-Prinzip, einer cloudbasierten Supabase-Integration mit PostgreSQL-Datenbank und Echtzeit-Synchronisation sowie einem Provider-basierten State-Management für optimierte Speicherverwaltung, ermöglichte sowohl eine performante App-Initialisierung als auch flüssige UI-Animationen mit konstanter Bildwiederholrate. Besonders hervorzuheben ist die Implementierung eines 60-FPS-CustomPainter für den Timer-Fortschrittsring, der auch auf älteren Geräten verzögerungsfreie Animationen gewährleistet, sowie die systematischen Maßnahmen zur Race-Condition-Vermeidung durch explizite Textvergleiche zwischen UI-Komponenten und Service-Layer, die potenzielle Endlosschleifen und App-Abstürze zuverlässig verhindern. Auch das durchdachte Caching-Konzept mit Write-Through-Mechanismen reduziert wiederholte Datenbankabfragen erheblich und trägt zur spürbaren Entlastung des UI-Threads bei kontinuierlicher Nutzung bei. Insgesamt bestätigen sowohl die technischen Messungen als auch die positiven Nutzerbewertungen eindeutig die Stabilität und Performance der entwickelten Lösung.

Gleichzeitig zeigen die Evaluationen klar identifizierbare Schwachstellen und Grenzen der aktuellen Version, die für zukünftige Entwicklungsiterationen relevant sind. Dazu gehören insbesondere plattformspezifische Besonderheiten wie der komplizierte iOS-Beta-Installationsweg über Xcode-Kopplung, der erhebliche technische Hürden für Endnutzer darstellt, sowie systematische Einschränkungen bei iOS-Benachrichtigungen aufgrund fehlender App-Store-Distribution und entsprechender Berechtigungsstrukturen. Weitere Limitationen umfassen vereinzelte nicht implementierte Komfortfunktionen wie die komplexere Freeze-Days-Logik für verschiedene Streak-Typen, die aufgrund der Fokussierung auf Login-Streaks nicht vollständig ausgearbeitet wurde, sowie noch nicht realisierte erweiterte Such- und Filteroptionen in der Aufgabenverwaltung, die bei größeren Aufgabenmengen die Übersichtlichkeit beeinträchtigen könnten. Diese identifizierten Punkte sind primär auf bewusste zeitliche und ressourcenbedingte

Priorisierungsentscheidungen zurückzuführen, die zugunsten der Implementierung und Stabilisierung der Kernfunktionen getroffen wurden.

Die zahlreichen nutzerorientierten Anpassungen, die direkt auf Usability-Feedback aus den Think-Aloud-Protokollen basieren, unterstreichen nachdrücklich die nutzerzentrierte Ausrichtung des gesamten Projekts und verbessern die Eignung der App für die Zielgruppe. Dazu gehören insbesondere vereinfachte Level-Darstellungen ohne überflüssige sprachliche Bezeichnungen sowie umfassende flexible Accessibility-Einstellungen wie einen WCAG-konformen Dark Mode mit optimierten Kontrastverhältnissen, stufenlose Textvergrößerung mit proportionaler Touch-Target-Anpassung und differenzierte Sound-Kontrolle mit Vibrations-Alternativen. Die durchweg positiven Nutzerbewertungen für diese Accessibility-Features mit Durchschnittswerten von 4,3 bis 4,7 von 5 Punkten bestätigen eindrucksvoll die Angemessenheit der getroffenen Designentscheidungen.

Zusammenfassend lässt sich mit hoher Evidenz festhalten, dass die entwickelte App die in der Konzeptionsphase formulierten Kernziele weitgehend erreicht und dabei sowohl technische als auch nutzerzentrierte Anforderungen erfüllt. Die MindSpark-Prototyp-Implementierung bietet eine umfassend funktionale, durchgängig performante und für die spezifische Zielgruppe neurodivergenter Lernender adaptierbare technische Basis, die sowohl konzeptionell durch die theoretisch fundierte Gamification-Strategie als auch technisch durch die modulare Flutter-Architektur langfristig tragfähig ist. Die strategische Priorisierung auf Kernfunktionen entsprechend dem MVP-Ansatz hat sich als besonders zielführend bewährt, weil dadurch eine nachweislich stabile und in realen Nutzungsszenarien erprobte Grundlage entstanden ist, auf der spätere systematische Erweiterungen wie umfassende Offline-Funktionalität mit lokaler Datensynchronisation, erweiterte Streak-Logiken für verschiedene Aktivitätstypen sowie vollständige iOS-Optimierungen mit nativer Benachrichtigungsintegration gezielt und ressourcenschonend aufgebaut werden können. Die vorliegende Arbeit liefert damit eine wissenschaftlich fundierte und praktisch validierte Ausgangsbasis für weiterführende Entwicklungs- und Evaluationsschritte im Bereich neurodivergenzspezifischer Bildungstechnologien.

7.2 Ausblick

Wenngleich die Testpersonen bei den Usability-Tests die MindSpark-App bereits als funktional umfassend und technisch stabil bewerteten, sind dennoch zahlreiche ergänzende Themenbereiche, innovative Ideen, konstruktive Anregungen sowie konkrete Verbesserungsvorschläge aufgekommen, die erhebliches weiteres Entwicklungspotenzial bieten. In der nachfolgenden strukturierten Auflistung sollen all diese sowie weitere wissenschaftlich fundierte Vorschläge aufgegriffen und detailliert beschrieben wer-

den. Damit werden zusätzliche praxisnahe Anregungen zu Implementierungsmöglichkeiten für die letztendliche produktive Umsetzung einer vollständigen Version der neurodivergenzspezifischen Lernhilfe-App gegeben.

Anbindung und Integration von Social-Media-Authentifizierung:
Dieser zentrale Punkt wurde beim finalen Usability-Testing wiederholt von verschiedenen Teilnehmenden genannt und umschließt die Integration von Single-Sign-On-Verfahren wie Google-ID und Apple-ID in den bestehenden Authentifizierungsworkflow der App. Dies beinhaltet die Integration in die bestehende Supabase-Authentifizierungsarchitektur, die bereits die erforderlichen Provider-Schnittstellen unterstützt. Eine solche Erweiterung würde insbesondere neurodivergenten Nutzern entgegenkommen, da sie komplexe Passwort-Eingabeprozesse umgeht und die kognitive Belastung bei der ersten App-Nutzung erheblich reduziert. Sofern dies datenschutzrechtlich vertretbar ist und den Privacy-by-Design-Prinzipien entspricht, wäre eine weitere strategische Option auch die Integration zusätzlicher Identity-Provider wie Microsoft-Konten für Bildungseinrichtungen.

Implementierung einer Passwort-Wiederherstellungsfunktion:
Nach der Evaluation wurde deutlich, dass selbst technikaffine Nutzer gelegentlich ihre Zugangsdaten vergessen, was derzeit zum vollständigen Verlust aller Nutzerdaten führt. Eine robuste Passwort-Reset-Funktionalität über E-Mail-basierte Token-Verfahren würde nicht nur die Benutzerfreundlichkeit erheblich steigern, sondern auch die langfristige Nutzerbindung stärken. Die technische Umsetzung könnte über Supabase's integrierte Auth-Reset-Funktionen mit zeitlich begrenzten Sicherheits-Token erfolgen, die über verschlüsselte E-Mail-Links aktiviert werden.

Implementierung einer Offline-Funktionalität:
Eine vollständige Offline-Capability der App wurde von der Mehrheit der Testpersonen als wünschenswert bezeichnet und stellt eine der prioritären Erweiterungen für zukünftige Versionen dar. Die technische Implementierung würde ein ausgeklügeltes Synchronisationskonzept mit lokaler SQLite-Datenbank und intelligenten Conflict-Resolution-Mechanismen erfordern. Alle Kernfunktionen wie Aufgabenverwaltung, Timer-Sessions, XP-Vergabe und Mood-Tracking sollten vollständig offline verfügbar bleiben, während eine automatische Synchronisation bei Wiederherstellung der Internetverbindung erfolgt. Ein visueller Offline-Indikator würde Nutzer stets über den aktuellen Verbindungsstatus informieren.

Erweiterte Kategorisierungs- und Organisationsoptionen für Aufgaben:
Die aktuell implementierte Prioritätsstufung (niedrig, mittel, hoch) erwies sich während der Tests als zu limitiert für komplexere Anwendungsszenarien. Eine Erweiterung um kontextuelle Kategorien wie Universität, Zuhause, Arbeit oder thematische Gruppierungen wie Lernen, Haushalt, Freizeit würde die Organisationsmöglichkeiten erheblich verbessern. Die Implementierung sollte direkt in die bestehende

Aufgabenerstellungs-UI integriert werden, wobei Nutzer bereits während der Erstellung entsprechende Kategorien auswählen können. Zusätzlich könnte ein Tagging-System mit frei definierbaren Labels noch flexiblere Organisationsmöglichkeiten bieten.

Implementierung erweiterter Such- und Filterfunktionen:
Bei einer größeren Anzahl von Aufgaben wird die Navigation in der aktuellen Liste unübersichtlich, was die Produktivität beeinträchtigt. Erweiterte Suchoptionen nach Titel, Beschreibung, Erstellungsdatum, Kategorie oder Tags würden die Auffindbarkeit spezifischer Aufgaben erheblich verbessern. Eine Implementierung mit Real-Time-Search und intelligenten Vorschlägen würde dabei die Benutzerfreundlichkeit optimieren.

Visualisierung der Stimmungsverläufe durch interaktive Diagramme:
Die derzeit implementierte statistische Auswertung mit Durchschnittswerten wurde von den Testpersonen als zu abstrakt empfunden. Nutzer wünschten sich explizit Liniendiagramme, Balkendiagramme oder andere visuelle Darstellungen, um Muster und Trends in ihrer emotionalen Verfassung intuitiv erkennen zu können. Die Implementierung könnte über Chart-Bibliotheken wie Charts.js oder D3.js erfolgen und sowohl wöchentliche als auch monatliche Trends mit interaktiven Zoom- und Filterfunktionen darstellen.

Erweiterte Freeze-Days-Anbindung und diversifizierte Streak-Systeme:
Das aktuell nur rudimentär implementierte Streak-System beschränkt sich auf Login-Aktivitäten und bietet noch nicht die geplante Freeze-Days-Funktionalität. Eine Erweiterung um verschiedene Streak-Typen wie Aufgaben-Streaks, Timer-Streaks oder Mood-Tracking-Streaks würde die Motivationsmechanismen erheblich steigern. Die Freeze-Days-Logik sollte dabei flexibel konfigurierbar sein und verschiedene Unterstützungsmechanismen für Nutzer in schwierigen Phasen bieten, ohne das Gamification-System zu untergraben.

Implementierung Achievement-Systeme und dynamischer Challenges:
Das bestehende Achievement-System könnte um zeitlich begrenzte Challenges oder personalisierte Ziele erweitert werden. Wöchentliche oder monatliche Herausforderungen wie Produktivitäts-Sprints, Konsistenz-Challenges oder collaborative Achievements würden zusätzliche Motivation schaffen. Die Implementierung sollte adaptive Algorithmen nutzen, die sich an individuelle Nutzungspattern anpassen und weder über- noch unterfordern.

Umfassende Rückgängig-Funktionalität für Benutzeraktionen:
Die begrenzte Undo-Funktionalität wurde von mehreren Teilnehmenden als stressverursachend empfunden, besonders bei impulsiven Aktionen. Eine systemweite Undo-Funktion für Aufgabenlöschungen, Prioritätsänderungen, Timer-Abbrüche und Stimmungseinträge würde das Vertrauen in die App erheb-

lich stärken. Die technische Umsetzung könnte über ein Command-Pattern mit Action-History erfolgen, das zeitlich begrenzte Rollback-Möglichkeiten bietet.

Vollständige iOS-Benachrichtigungsintegration:
Die derzeit eingeschränkte iOS-Benachrichtigungsfunktionalität erfordert eine komplette Überarbeitung der Notification-Architektur. Eine ordnungsgemäße App-Store-Distribution mit entsprechenden Berechtigungsstrukturen würde native iOS-Push-Notifications mit täglichen Erinnerungen, ermöglichen. Die Implementierung sollte dabei die spezifischen iOS-Design-Guidelines und Notification-Categories berücksichtigen.

Abschließend lässt sich festhalten, dass die vorgestellten Erweiterungen und Optimierungen nicht nur die Funktionalität und Benutzerfreundlichkeit der MindSpark-App steigern, sondern auch das Potenzial haben, die Anwendung noch gezielter auf die Bedürfnisse neurodivergenter Menschen abzustimmen. Durch die Einbindung von neurodivergenten Nutzern als Co-Designer und aktive Mitwirkende können authentische Lösungen entstehen, die ihre spezifischen Anforderungen direkt berücksichtigen. Die MindSpark-App demonstriert das Potenzial technologischer Lösungen für neurodivergente Lernende. Sie zeigt gleichzeitig die Notwendigkeit nutzerzentrierter Entwicklungsansätze und interdisziplinärer Zusammenarbeit auf. Die Arbeit trägt zur noch jungen Forschungslandschaft digitaler Inklusion bei und liefert praktische Erkenntnisse für die Entwicklung barrierefreier Bildungstechnologien.

Literaturverzeichnis

1) Al-Msie'deen, R. F., Blasi, A. H., & Alsuwaiket, M. A. (2021). Constructing a software requirements specification and design for electronic IT news magazine system. arXiv preprint arXiv:2111.01501.

2) Alper, M., Alcorn, A. M., Harrison, K., Manganello, J. A., & Romeo, R. R. (2024). Digital Media and Neurodevelopmental Differences. Handbook of Children and Screens: Digital Media, Development, and Well-Being from Birth Through Adolescence, 55-60.

3) Aswalekar, U. & Vishwakarma, A. (2025). State Management in Flutter: A Performance Comparison of GetX, Provider, Riverpod and Bloc.

4) Baron-Cohen, S. (2017). Editorial Perspective: Neurodiversity—a revolutionary concept for autism and psychiatry. Journal of child psychology and psychiatry, 58(6), 744-747.

5) Birch, J. (2012). Worldwide prevalence of red-green color deficiency. Journal of the Optical Society of America A, 29(3), 313-320.

6) Blal, R., Leshob, A., Mili, H., Benzarti, I., Hadaya, P., & Rab, R. (2025). SOA services identification and design methods from business models: A systematic literature review. IEEE Access.

7) Boyd, L. E., Ringland, K. E., Haimson, O. L., Fernandez, H., Bistarkey, M., & Hayes, G. R. (2016). SayWAT: Augmenting face-to-face conversations for adults with autism. Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, 4872-4883.

8) Chapman, R. (2021). Neurodiversity and the social ecology of mental functions. Perspectives on Psychological Science, 16(6), 1360-1372.

9) Chen, P. W. (2024). Accessibility in HCI: Analyzing Inclusive Design Principles and Techniques for Ensuring Accessibility to Interactive Systems for Users With Disabilities. Human-Computer Interaction Perspectives, 2(1), 1–10. Retrieved from https://thesciencebrigade.com/hcip/article/view/102

10) Dovis, S., Van der Oord, S., Wiers, R. W., & Prins, P. J. (2012). Can motivation normalize working memory and task persistence in children with attention-deficit/hyperactivity disorder? The effects of money and computer-gaming. Journal of abnormal child psychology, 40(5), 669-681.

11) Faraone, S. V., Banaschewski, T., Coghill, D., Zheng, Y., Biederman, J., Bellgrove, M. A., ... & Wang, Y. (2021). The world federation of ADHD international consensus statement: 208 evidence-based conclusions about the disorder. Neuroscience & biobehavioral reviews, 128, 789-818.

12) Floch, J., Vilarinho, T., Zettl, A., Ibanez-Sanchez, G., Calvo-Lerma, J., Stav, E., ... & Montón, J. L. B. (2020). Users' experiences of a mobile health self-management approach for the treatment of cystic fibrosis: mixed methods study. JMIR mHealth and uHealth, 8(7), e15896.

13) Flutter Team. (2025). Automatic platform adaptations. Flutter Documentation. [zuletzt besucht am 11.09.2025] https://docs.flutter.dev/ui/adaptive-responsive/platform-adaptations

14) Hallie, L. & Osmani, A., (o. J.). *Provider Pattern*. [zuletzt besucht am 28.09.2025] von https://www.patterns.dev/vanilla/provider-pattern/

15) Herron, K. (2023). From Comfort to Contempt: Human made environments, the effects of color and light on the human condition (Doctoral dissertation, Alfred University).

16) Hosseini, C., Humlung, O., Fagerström, A., & Haddara, M. (2022). An experimental study on the effects of gamification on task performance. Procedia Computer Science, 196, 999-1006.

17) Jaarsma, P., & Welin, S. (2012). Autism as a natural human variation: Reflections on the claims of the neurodiversity movement. Health care analysis, 20, 20-30.

18) Kakoura, E., Loukas, P., & Sideraki, A. (2024). A mobile app as a gamified early intervention for ADHD students. World Journal of Biology Pharmacy and Health Sciences, 18(1), 48-53.

19) Kapp, S. K., Gillespie-Lynch, K., Sherman, L. E., & Hutman, T. (2013). Deficit, difference, or both? Autism and neurodiversity. Developmental psychology, 49(1), 59.

20) Keil, M., & Bleisinger, O. (2023, November). Benutzeroberflächen von MBSE-Tools und deren Auswirkung auf neurodivergente Systemarchitekten. In Proceedings of Tag des Systems Engineering 2023 (pp. 9-15).

21) Keogh, C. (2022). Minimum Viable Product (MVP). Definitions. https://doi.org/10.32388/bzygt9.

22) Kofler, M. J., Irwin, L. N., Soto, E. F., Groves, N. B., Harmon, S. L., & Sarver, D. E. (2019). Executive functioning heterogeneity in pediatric ADHD. Journal of abnormal child psychology, 47, 273-286.

23) Liu, W., Guo, L., Heng, Y. W., Li, C., & Hassan, A. E. (2025). Screencast-Based Analysis of User-Perceived GUI Responsiveness. arXiv preprint arXiv:2508.01337.

24) Lundqvist, L. (2024). Optimizing Accessibility in UX Design through the Integration of Artificial Intelligence.

25) Maaß, C., & Rink, I. (2018). Handbuch Barrierefreie Kommunikation (p. 800). Frank & Timme.

26) Mavrides, N. (2016). Attention-Deficit Hyperactivity Disorder: A Handbook for Diagnosis and Treatment, Russell A. Barkley, Ed.(2014) New York: The Guilford Press. 898 pp.

27) Mazarakis, A., & Bräuer, P. (2023). Gamification is working, but which one exactly? Results from an experiment with four game design elements. International Journal of Human–Computer Interaction, 39(3), 612-627.

28) Miesenberger, K. (2018). Assistierende Technologien und digitale Barrierefreiheit. Sonderpädagogik in der digitalisierten Lernwelt, 11.

29) Nicholson, S. (2015). A recipe for meaningful gamification. Gamification in education and business, 1-20.

30) Nyabuto, M. G. M., Mony, V., & Mbugua, S. (2024). Architectural review of client-server models. International journal of scientific research and engineering trends, 10(1), 139-143.

31) Orji, R., Vassileva, J., & Mandryk, R. L. (2014). Modeling the efficacy of persuasive strategies for different gamer types in serious games for health. User Modeling and User-Adapted Interaction, 24(5), 453-498.

32) Pedersen, L. A., Einarsson, S. S., Rikheim, F. A., & Sandnes, F. E. (2020, July). User interfaces in dark mode during daytime—improved productivity or just cool-looking?. In International Conference on Human-Computer Interaction (pp. 178-187). Cham: Springer International Publishing.

33) Peijnenborgh, J. C., Hurks, P. M., Aldenkamp, A. P., Vles, J. S., & Hendriksen, J. G. (2016). Efficacy of working memory training in children and adolescents with learning disabilities: A review study and meta-analysis. Neuropsychological Rehabilitation, 26(5-6), 645-672.

34) Philip, R. C., Dauvermann, M. R., Whalley, H. C., Baynham, K., Lawrie, S. M., & Stanfield, A. C. (2012). A systematic review and meta-analysis of the fMRI investigation of autism spectrum disorders. Neuroscience & Biobehavioral Reviews, 36(2), 901-942.

35) Prins, P. J., Dovis, S., Ponsioen, A., ten Brink, E., & van der Oord, S. (2011). Does computerized working memory training with game elements enhance motivation and training efficacy in children with ADHD? Cyberpsychology, Behavior, and Social Networking, 14(3), 115-122.

36) Prinzellner, Y., Simon, A., Drachmann, D., Werner, K., Münter, L., Bulsink, V., ... & Schwaninger, I. (2022, December). ”The support needs to be part of the system: designing inclusive eHealth applications for older adults with low eHealth literacy. In Proceedings of the Conference on Computer Supported Cooperative Work (Vol. 418).

37) Probiesch, K. (2013). Barrierefreiheit im Web. RKW Kompetenzzentrum.

38) Purnama, Y., Herman, F. A., Hartono, J., Suryani, D., & Sanjaya, G. (2021). Educational software as assistive technologies for children with autism spectrum disorder. Procedia Computer Science, 179, 6-16.

39) Pushpakumar, R., Sanjaya, K., Rathika, S., Alawadi, A. H., Makhzuna, K., Venkatesh, S., & Rajalakshmi, B. (2023). Human-computer interaction: Enhancing user experience in interactive systems. In E3S Web of Conferences (Vol. 399, p. 04037). EDP Sciences.

40) Reinecke, K., Nguyen, M. K., Bernstein, A., Näf, M., & Gajos, K. Z. (2013, February). Doodle around the world: Online scheduling behavior reflects cultural differences in time perception and group decision-making. In Proceedings of the 2013 conference on Computer supported cooperative work (pp. 45-54).

41) Roddy, A., & O'Neill, C. (2019). The economic costs and its predictors for childhood autism spectrum disorders in Ireland: How is the burden distributed?. Autism, 23(5), 1106-1118.

42) Salminen, J., Wenyun Guan, K., Jung, S. G., & Jansen, B. (2022, April). Use cases for design personas: A systematic review and new frontiers. In Proceedings of the 2022 CHI Conference on human factors in computing systems (pp. 1-21).

43) Schuh, J. M., & Eigsti, I. M. (2012). Working memory, language skills, and autism symptomatology. Behavioral Sciences, 2(4), 207-218.

44) Spiel, K., Hornecker, E., Williams, R. M., & Good, J. (2022, April). ADHD and technology research—investigated by neurodivergent readers. In Proceedings of the 2022 CHI conference on human factors in computing systems (pp. 1-21).

45) Stephanidis, C., Salvendy, G., Antona, M., Duffy, V. G., Gao, Q., Karwowski, W., ... & Zhou, J. (2025). Seven HCI Grand Challenges Revisited: Five-Year Progress. International Journal of Human–Computer Interaction, 1-49.

46) Stone, M. (2016). A field guide to digital color. CRC press.

47) W3C. (2025). Web Content Accessibility Guidelines (WCAG) 2.1. World Wide Web Consortium. [zuletzt besucht am 28.08.2025] Web Content Accessibility Guidelines (WCAG) 2.1

48) Weichbroth, P. (2024). Usability testing of mobile applications: A methodological framework. Applied Sciences, 14(5), 1792.

49) Xiao, J., Huang, Q., Chen, X., & Tian, C. (2024). Understanding Large Language Models in Your Pockets: Performance Study on COTS Mobile Devices. arXiv preprint arXiv:2410.03613.

50) Zbarcea, A., & Tudose, C. (2024). Migrating from Developing Asynchronous Multi-Threading Programs to Reactive Programs in Java. Applied Sciences (2076-3417), 14(24).

A Anhang

Zusammenfassung Umfrage 1: Lernmotivation und App-Gestaltung

Kategorie	Fragestellung / Ergebnis	Anteil (%)
Demografie	Neurodivergente Teilnehmer	46,7
	Altersgruppe 18-25 Jahre	40,0
	Altersgruppe unter 18 Jahre	26,7
Lernprobleme	Konzentration über längere Zeit	26,7
	Dranbleiben / Motivation	26,7
	Ablenkung	20,0
	Frustration bei Rückschlägen	13,3
Zeitmanagement	Timer helfen beim Fokussieren	73,3
	Brauche feste Zeitblöcke	53,3
	Erinnerungen alle paar Minuten	46,7
Aufgaben-organisation	Nach Priorität organisieren	66,7
	Nach Fälligkeit/Deadline	60,0
	Nach Aufwand (kurz/mittel/lang)	46,7
Überforderung	Aufgaben schneller bearbeiten können	93,3
	Notfall-Stop Button	60,0
Störfaktoren	Zu viele Pop-ups oder Unterbrechungen	86,7
	Komplizierte Bedienung	80,0
	Ständige Werbung	66,7
Erinnerungen	In der App sichtbare Erinnerungen	60,0
	Motivierende Sprüche/Quotes	46,7

Zusammenfassung Umfrage 2: MindSpark Beta-Testing Feedback

Kategorie	Fragestellung / Ergebnis	Anteil (%)
Betriebssystem	iOS (iPhone)	64,3
	Android	35,7
Installation	Problemlos und schnell	50,0
	Normal, ohne Auffälligkeiten	35,7
	Etwas langsam, aber erfolgreich	14,3
Installations-probleme	Keine Probleme	71,4
	Sicherheitswarnung des Systems	14,3
	Berechtigungen nicht gewährt	7,1
App-Start	Ja, sofort beim ersten Versuch	92,9
	Ja, nach einem zweiten Versuch	7,1
Splash Screen	Angemessen - gute Balance	64,3
	Etwas zu lang - hat aufgehalten	28,6
Onboarding	Ja, vollständig durchlaufen	64,3
	Ja, teilweise durchlaufen	28,6
	Weiß nicht, was das ist	7,1
Onboarding-probleme	Keine Probleme	71,4
	Text war unleserlich/abgeschnitten	42,9
Login/ Registrierung	Alles funktionierte einwandfrei	92,9
	Verbindungsfehler	7,1
Gemeldete Bugs	Onboarding nach Neuinstallation fehlt	1 Nennung
	Keine Bugs aufgefallen	1 Nennung

Zusammenfassung Umfrage 3: MindSpark Beta-Test UI Evaluation

Kategorie	Fragestellung / Ergebnis	Anteil (%)
Getestete Bereiche	Aufgaben-Bereich	100,0
	Stimmung/Mood-Tracker	100,0
	Home-Screen	90,0
	Einstellungen	90,0
Navigation	Navigation intuitiv (Durchschnitt 4-5 Sterne)	Mehrheit
	Bereiche nicht gefunden	0,0
Home-Screen Design	angemessen strukturiert	80,0
	Neutral	10,0
	Sehr beruhigend und übersichtlich	10,0
Aufgaben-Darstellung	Größtenteils gut lesbar	60,0
	Alles sehr klar erkennbar	40,0
Timer-Bereich	Sehr beruhigend und fokussierend	50,0
	Neutral	40,0
	Eher beruhigend	10,0
Fortschritt	Ja, sehr motivierend	50,0
	Neutral - weder motivierend noch demotivierend	30,0
	Ja, etwas motivierend	20,0
Barrierefreiheit	Ja, leicht zu finden	80,0
	Ja, nach kurzem Suchen	20,0
Visuelle Aspekte (Top 3)	Ruhige, gedeckte Farben	60,0
	Klare Schriftarten (wichtig)	60,0
	Hoher Kontrast für bessere Lesbarkeit	50,0
Touch-Eingaben	Sehr reaktionsschnell	60,0

Kategorie	Fragestellung / Ergebnis	Anteil (%)
	Gut responsiv	40,0
UI-Probleme	Keine größeren Probleme	60,0
	Text ist abgeschnitten	40,0
	Farben zu grell/aufdringlich	10,0
Vorschläge	Sliders farblich anpassen (weniger monoton)	2 Nennung
Positive Aspekte	XP und Fortschrittsseite	1 Nennung
	Schnelle Stimmungserfassung mit Speicherung	1 Nennung

Zusammenfassung der Testszenarien für neurodivergente Nutzer

Typ	Szenario	Zielsetzung	Zeit
Kern-aufgaben	1: Erste Schritte und Onboarding	Bewertung der ersten Nutzererfahrung und Barrierenidentifikation	5 Min
	2: Erste Aufgabe erstellen	Test der Aufgabenerstellungs-Funktion bei komplexeren Eingaben	-
	3: Timer-Funktion erkunden	Bewertung der Pomodoro-Timer Implementierung	25 Min
	4: Fortschritt und Gamification	Test der Progress-Screens und Motivationswirkung	-
	5: Personalisierung und Accessibility	Bewertung der Accessibility-Features und Auffindbarkeit	-

Typ	Szenario	Zielsetzung	Zeit
	6: Stimmungstracker-Integration	Test der Stimmungstracker-Funktionalität	-
	7: Kollaborative Aufgaben verwalten	Test bei komplexeren, mehrteiligen Projekten	-
	8: Langzeit-Motivation und Streak-System	Bewertung der langfristigen Motivationswirkung	-
ADHS-spezifisch	A: Ablenkung und Wiedereinstieg	Test der Emergency-Stop-Funktion und Wiedereinstiegshilfen	-
	B: Überforderungssituation	Test der Freeze-Days-Funktion bei Überforderung	-
Autismus-spezifisch	C: Sensorische Anpassung	Test der sensorischen Anpassungsmöglichkeiten und reizarmen Modi	-
	D: Routine-Etablierung	Test der wiederkehrenden Aufgaben und Routine-Unterstützung	täglich

B Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Konzeption und Entwicklung einer gamifizierten Lernhilfe-App für neurodivergente Menschen

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

22. September 2025 [redacted]

Datum Unterschrift