Hochschule Neu-Ulm
University of Applied Sciences

Bachelorarbeit

im Bachelorstudiengang Information Management & Automotive

an der Hochschule für angewandte Wissenschaften Neu-Ulm

Konzeption und Entwicklung einer gamifizierten App zur Unterstützung der Therapie von myofunktionellen Störung bei Kindern

Erstkorrektor:	Professor Dr. Johannes Schobel
Zweitkorrektor:	Professor Manfred A. Plechaty
Betreuer:	Maximilian Karthan

Verfasser: Kisanthan Nagarasa, Matrikelnummer: 287576

Thema erhalten:	01.06.2022
Arbeit abgegeben:	04.10.2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Code Verzeichnis

Abkürzungsverzeichnis

UX	-	User Experience
UI	-	User Interface

Abstrakt

Die stetige Entwicklung der Technologien und Digitalisierungen bereichern uns tagtäglich das Leben. In der Gesundheitsbranche wird dieser Umstand genutzt, um neue Therapiemöglichkeiten auf Basis mobiler Anwendungen zu entwickeln. Solche Anwendungen sollen unterstützend zur Behandlung bestimmter Krankheiten angewendet werden. Insbesondere für die Unterstützung der Therapie von Kindern liefert diese Technologie neue Möglichkeiten durch die Integration spielerischer Elemente die Motivation der Patienten hoch zu halten. Im Rahmen dieser Arbeit wird eine Anwendung entwickelt, welche Kinder bei der Therapie einer myofunktionellen Störung spielerisch unterstützt. Mit der Bereitstellung einer Anwendung wird dem Patienten ein Spiel zur Verfügung gestellt, dass durch das Pusten in einer Trillerpfeife die beeinträchtigten mundmotorischen Muskulaturen fördert. Das Spiel wird nicht über einen klassischen Touchscreen gesteuert, sondern über den Ton einer Pfeife. Das Grundgerüst ist mit der Entwicklung dieser App gegeben. Allerdings stellt die Applikation nicht für alle Symptome der myofunktionellen Störung eine Therapiemöglichkeit zur Verfügung. Diese Arbeit stellt die Grundlage, auf der aufbauend weitere Therapiemöglichkeiten angeboten werden können.

1 Einleitung

Die Fortschritte der Technologien und Digitalisierungen bereichern uns tagtäglich das Leben. Vor einigen Jahren hätte nie jemand gedacht, dass ein Roboter im Haushalt behilflich seien wird oder dass mit Sprachfunktionen das Licht ein- und ausgeschaltet werden kann. Es werden jegliche Apps entwickelt, damit unser gesamter Alltag über unser Smartphone gesteuert werden kann. Nur mit dem Smartphone können wir in Läden bezahlen, mit unseren Freunden im Ausland in Kontakt bleiben, unser Haus mit Smart-Home steuern und vieles mehr. Selbst für unsere Gesundheit ist das Smartphone bereits eine Bereicherung. Das Smartphone wird beispielsweise genutzt, um seine Fitnessdaten im Auge zu behalten, in dem z.B. die täglichen Schritte tracken können oder die Trainingsfortschritte überwachen können. Frauen können sogar mit einer App ihren Menstruationszyklus im Auge behalten, die obendrein zur Unterstützung während einer Schwangerschaft genutzt werden kann. In Kombination mit Smartwatches können überdies die Herzfrequenz beobachtet werden, bei Unregelmäßigkeiten erscheint obendrein eine Benachrichtigung zur Kenntnisnahme bzw. zur Warnung. Herzkranke Patienten können solche Funktionen zu Nutzen machen, um ihre Werte im Auge zu behalten. Krankenkassen stellen bereits eigene Apps zur Verfügung, in denen Krankmeldungen, Belege etc. hochgeladen werden können.

Dennoch werden die ganzen Vorteile solcher Apps in der Gesundheitsbranche noch nicht ausgiebig genutzt. Gewisse Krankheiten könnten anhand solcher Apps erkannt und sogar unterstützend zur Behandlung genutzt werden. Vor allem Kinderkrankheiten können mit spielerischen App-Funktionen erkannt und behandelt werden.

Im Rahmen dieser Forschungsarbeit gehen wir vor allem auf die Krankheit myofunktionelle Störung ein.

1.1 Problemstellung

Eine myofunktionelle Störung ist ein Ungleichgewicht der Muskeln im Mund- und Gesichtsbereich. Aufgrund dieser Störung ist zu dem auch die Wahrnehmung von Berührungen und Bewegungen (die taktil-kinästhetische Wahrnehmung) beeinträchtigt. Die myofunktionelle Störung wird in der Regel erst ab dem 4. Lebensjahr festgestellt und behandelt.

Neben eine Therapie können unterstützend spielerische mundmotorische Übungen helfen. Solche Übungen sollen dem Kind dabei helfen, die Muskulatur und die Berührungs- und Bewegungswahrnehmung zu fördern. Zu den spielerischen mundmotorischen Übungen gehören z.B. Puste-, Blas- und Ansaugspiele, Seifenblasen oder Grimassen-Schneiden (Elfert, 2022).

1.2 Zielsetzung

Zur Unterstützung der Therapie von einer myofunktionellen Störung können auch Apps eingesetzt werden. Mundmotorische Übungen können in eine App integriert werden. Da Kinder ohnehin viele Apps auf den Tablets / Smartphone von ihren Eltern nutzen, um z.B. Lernspiele zu spielen, kann das Integrieren von mundmotorischen Übungen ein großer Vorteil sein.

In diesem Sinne wird anhand der Entwicklung einer Spiel-App, zur Unterstützung der Therapie von einer myofunktionellen Störung, das Nutzen aufgezeigt und erläutert.

1.3 Aufbau der Arbeit

Die verbleibenden Kapitel sind folgt aufgebaut. In Kapitel 2 werden grundlegende Informationen aufgegriffen, die relevant für diese Arbeit sind. Kapitel 3 befasst sich mit vergleichbaren Applikationen. Das vierte Kapitel zeigt die Entwürfe, Methoden und verwendete Technologien dieser Arbeit auf. Im Kapitel 5 werden die essenziellen Codestellen erläutert, welche das Ergebnis dieser Arbeit veranschaulicht. Anschließend setzen wir uns im Kapitel 6 in einer Diskussion mit dem Ergebnis dieser Arbeit auseinander. In Kapitel 7 wird abschließend die gesamte Arbeit erläutert und mit einem Ausblick beendet.

2 Grundlagen

In diesem Kapitel erläutern wir die grundlegenden Begriffe dieser Arbeit, um ein besseres Verständnis in die Thematik zu erhalten.

2.1 Myofunktionelle Störung

Bei einer myofunktionellen Störung handelt es sich um eine Fehlfunktion im Mund- und Gesichtsbereich. Diese Fehlfunktion wird über eine fehlerhafte Schluckweise definiert. Die lässt sich erkennen, wenn z.B. die Zunge gegen die Zähne stößt oder sich zwischen den Zähnen zeigt. Die falsche Schluckweise wirkt sich negativ auf den orofazialen Bereich (Mund- und Gesichtsbereich) und auf die Gebissentwicklung aus. Ebenso spiegelt sie sich wider in der Haltung, Wahrnehmung oder im Gleichgewicht, z.B. durch einen ständig geöffneten Mund (Ruben & Wittich, 2014).
Die myofunktionelle Störung lässt sich in den folgenden Bereichen diagnostizieren, in diesen die Anomalien erkennbar sind:

- Lippen
- Kinnmuskel
- Kieferstellung
- Atmung
- Zunge
- Zungenbändchen
- Mimik
- Artikulation
- Körperhaltung
- Kiefergelenkschmerzen
- Zähneknirschen.

Es gibt mehrere Therapiemöglichkeiten. Der Fokus in dieser Arbeit wird vor allem auf die Therapie der Mundmotorik sein (Kittel & Förster , 2016).

2.2 Gamification

Die Gamification wird in Bereichen angewendet in den nicht-spielerische Methoden in Spielmechanismen umgeformt werden. Durch diese soll eine höhere Produktivität mit einer gleichzeitig steigenden Nutzerzufriedenheit entstehen. Mithilfe der Gamification wird die Motivation unterstützt und ein Flow-Erlebnis vermittelt. Daraus entstehen Fortschritte im Bereich Leistungssteigerung, positive Verhaltensänderung und Erfolgserlebnis. Diese Methode unterstützt vor allem Lernprozesse, um Problemstellungen zu bewältigen, die einen Haupteinsatzbereich der Gamification darstellt (Blohm & Leimeister, 2013).

2.3 UX-Design

Das Ziel im UX-Design ist es nicht nur, dass die Nutzung einer Website oder einer App dem User leichtfällt, sondern dass es ihn auch begeistert und somit einen Mehrwert für ihn schafft. In erster Linie werden Situationen betrachtet, wie das Produkt mit dem User interagieren könnte und somit das Produkt userorientiert definiert wird. Im zweiten Schritt wird die Nutzbarkeit, sowohl als auch die Interaktionen, der Anwendungsfälle erarbeitet. Der dritte Schritt ist eine detaillierte Implementierung als graphische Darstellung, welches auch bekannt ist als UI-Design. Im letzten Schritt wird das erarbeitete Design-Endprodukt durch User bzw. Testakteuren analysiert. Nachdem der letzte Schritt absolviert wurde, fängt es wieder, wie in einem Kreislauf, mit dem zweiten Schritt, welcher zur Optimierung des Designs dient, an. Die Zufriedenheit des Users, in Betracht auf das Produkt, wird erhöht, wenn die Nutzbarkeit, Zugänglichkeit und Anwendungsfreunde stetig optimiert wird (Güb, 2010).

2.4 Figma

Das Tool Figma erstellt UX-Designs. Dieses Tool kann über eine Website aufgerufen werden, dass ermöglicht eine geräteübergreifende Nutzbarkeit. Mit Figma können einfache Modellierungen wie z.B. ein Aktivitätsdiagramm, User Flow, Mind-Map usw. erstellt werden. Ebenso können eigene Designs wie z.B. bei Adobe Illustrator kreiert werden. Mittels Figma werden Mockups erstellt. Zudem können weitere externe Software durch die Figma-Community angeboten werden, damit die Arbeit effizienter gestaltet wird. Icons, vorgefertigte Designs, Bilder etc. können durch die Plug-Ins kostenfrei und ohne Beschränkung verwendet werden. Des Weiteren besteht die Möglichkeit, dass die erstellten Mockups als interagierende Prototypen auf einem webbasierten grafischen Smartphone oder auch auf einem realen Smartphone

simuliert werden können, damit die Interaktion zwischen dem Design und dem User genaustens analysiert werden kann (Figma , 2022). In dieser Arbeit wird Figma als wesentliches Werkzeug zum Entwerfen von UI-Designs benutzt.

2.5 Ionic

Das Ionic ist ein Open-Source-UI-Tool, welches zum Entwickeln von Mobil- und Desktopapps genutzt wird. Die Entwicklung erfolgt durch Webtechnologien, wie HTML, CSS, JavaScript und eine Auswahl zwischen den Frameworks Angular, React und Vue. Als Cross-Plattform bietet Ionic die Möglichkeit, mit einem Code, Betriebssystem übergreifend, wie z.B. für iOS oder Android, zu entwickeln. Bereits vorhanden sind die vorgefertigten UI-Komponenten als Code und die Native APIs, beide sind in der Dokumentation mitaufgeführt. In dieser Arbeit wird das Framework Angular angewandt (Ionic, 2022).

2.5.1 Node.js

Node.js gilt als Voraussetzung für Ionic, da Ionic ein npm-Modul ist. Das Node.js ist eine Laufzeitkonstrukt für JavaScript, womit Daten schnell verarbeitet und einfach skaliert werden (Nodejs, 2022).

2.5.2 Angular

Angular ist ein clientseitiges Framework, welches als Programmiersprache das TypeScript nutzt. Angular ist ein komponentenbasiertes Framework, das skalierbare Web-Anwendungen erstellt, integrierte Bibliothek von Funktionen abdeckt und einige Entwicklungstools bereitstellt (Angular, 2022).

2.5.3 Capacitor

Capacitor Plug-ins ermöglichen Javascript eine unkomplizierte Interaktion mit nativen APIs. Durch die Plug-ins können vorgefertigte und native Funktionen abgerufen werden (Capacitor, 2022).

2.6 Nx Monorepo

Das Tool NX Monorepo vereinfacht das Kombinieren von verschiedenen Entwicklungstechnologien. Innerhalb der Monorepos können unterschiedliche Projekte, Frameworks und

projektübergreifende Codes verwendet werden, wie z.B. Frontend mit Ionic und Backend mit Firebase. Dadurch kann Zeit gespart und die Anzahl der Codezeilen verringert werden. Ebenso werden eventuelle Fehler bei Duplikationen vermieden. Durch eine eigene Konfiguration lassen sich Zugriffe auf Anwendungen bzw. Code leicht gestalten, die die Übersicht und Nutzung verbessern (Nx, 2022).

2.7 Phaser

Phaser ist ein Framework, welches Open-Source ist und für Entwicklungen von JavaScript-2D-Spielen eine effiziente Lösung anbietet. Hierfür wird Canvas und ein WebGL-Renderer benutzt, um das Wechseln zwischen Browsern einfach und automatisch zu ermöglichen, dass ebenfalls ein schnelles Rendern über Desktop und Mobile ermöglicht. Phaser verwendet die Programmiersprache JavaScript und beinhaltet bereits vorgefertigte Spiele mit den entsprechenden Dokumentationen (Phaser , 2022).

3 Vergleichbare Applikationen

Im Folgenden werden vergleichbare Applikationen vorgestellt, die ähnliche Ziele und Therapiemethoden vorweisen.

3.1 Leons Dschungelspaß

Die App Leons Dschungelspaß kann auf einem Android- und iOS-Smartphone heruntergeladen werden. Die App ist eine für Vorschulkinder geeignete Spielapplikation. Sie dient zur Unterstützung spielerisch den Körper und seine Sinne wahrzunehmen und vor allem die Mundmotorik zu verbessern. Im Insgesamten werden mehrere Minispiele zur Verfügung gestellt, die gleichzeitig verschiedene Übungen für die Entwicklung des Kindes darstellen.
Um die gesamte Leistung dieser App zu erhalten, muss eine einmalige Zahlung von 4,99€ getätigt werden. Im aktuellen Zustand wird diese App als eine Spiele-App und nicht als eine Therapie-App vermarktet (InnerMe GmbH, 2022).

3.2 PhonoLo

Die PhonoLo App ist im AppStore und im GooglePlay erhältlich. Die Zielgruppe der PhonoLo App sind Kinder zwischen 3 und 7 Jahren mit einer Störung in ihrer Artikulation. Somit soll die App die Aussprache der Kinder mit spielerisch dargestellten Übungen fachlich fördern. Die App wurde 2017 von Logopäden entwickelt und bereits bei logopädischen Praxen und Hochschulen erprobt. Um die volle Leistung dieser App zu erhalten, werden Kosten in Höhe von 9,99€ pro Lautpaar verlangt. Die App wird zur Therapiezwecken genutzt, die von Logopäden gekoppelt werden kann (PhonoLo, 2022).

3.3 Myolino Projekt

Das Myolino Projekt ist eine Webseite mit dem Fokus speziell auf myofunktionelle Störung. Von Kindes- bis Erwachsenenalter, jeder hat die Möglichkeit die Webseite zur Unterstützung seiner Therapie zu nutzen. Die Webseite ermöglicht den Nutzer eine sprachtherapeutische Behandlung durch das Zurverfügungstellen von Übungsvideos und therapeutischen Spiele. Das Myolino Projekt ermöglicht einen kostenlosen Zugang und stellt informationsreiche Inhalte über die myofunktionelle Störung zur Verfügung (Myolino, 2022).

4 Material und Methoden

Die ersten Schritte beginnen mit der Planung der ersten Entwürfe für die App. Die zeitliche Planung und Umsetzung erfolgten nach dem Scrum-Verfahren. Bei der Planung werden optimale Komponenten ausgewählt, um die App zu produzieren. Es werden optische Templates erstellt, um eine detaillierte Vorstellung der App zu erhalten.

4.1 Anforderungsanalyse

Die Anforderungen dieser Arbeit werden in funktionale und nicht-funktionale Anforderungen definiert. Die funktionalen Anforderungen zeigen die Hauptmerkmale eines Systems und das Verhalten unter festgelegten Bedingungen auf. Bei der nicht-funktionalen Anforderung werden Qualität und Randbedingungen beschrieben, dass das gesamte System betrifft (Balzert, 2011).

4.1.1 Funktionale Anforderung

Um einen von vielen Symptome der Krankheit myofunktionellen Störung entgegenzuwirken, soll diese Arbeit den Fokus auf die Entwicklung einer App legen, welche die notwendigen Funktionen hat, um eine spielerische Therapiemethode zur Verfügung zu stellen. Da bei der myofunktionellen Störung die Muskeln der Mundmotorik geschwächt bzw. beeinträchtigt sind, soll durch das implementieren von Pusteübungen in dieser App, eine erweiterte Therapie jederzeit ermöglichen. Folgende funktionale Anforderungen sollen dafür gegeben sein.

- Das Pusten in einer Trillerpfeife soll die Steuerung des Vogels ermöglichen
- Der Vogel wird je nach starkem bzw. schwachem Pusten höher oder niedriger getriggert

4.1.2 Nicht-Funktionale Anforderung

Neben der Hauptfunktion „Erkennung der Frequenzen durch das Pusten zur Bewegung des Flappy-Birds“, ist es ebenfalls wichtig, dass die App im Allgemeinen für den Anwender attraktiv bleibt. Das Interesse soll durch folgende Aspekte aufrechterhalten werden.

- Erstellen eines UX/UI Designs
- Einführung eines Rankingsystems
- Vernetzungen zwischen App Teilnehmer

- Weiterleiten von Spieldaten an Fachkräfte wie z.B. an den Arzt

Diese erwähnten Aspekte sowie andere essenzielle Randbedingen werden im Kapitel 4.2 UX-Design Entwurf detaillierter erläutert (Neureiter, 2019).

4.2 UX-Design Entwurf

Im UX-Design Entwurf wird die App konzipiert. Beim Erstellen des Konzepts wird ein großer Fokus auf die Punkte Analyse, Nutzerpsychologie, Nutzbarkeit, Inhalte und Ästhetik gesetzt. Im Folgenden wird die App aus Rohinformationen aufbauend zum detaillierten Endprodukt entworfen. Dennoch ist dieses Konzept nicht finalisiert und stellt im Moment die wichtigsten Aspekte und Struktur dar (Yablonski, Jon, 2022).

4.2.1 User-Story

Mit der User-Story wird bei der Konzeption einer App das Produkt aus Kundensicht bzw. mögliche Kundenanforderungen im Fokus gehalten. Ebenso soll ein gemeinsames Verständnis für alle Akteure der Entwicklung erschaffen werden. Die User-Story wird durch eine bestimmte Anforderungsart definiert und wie folgt formuliert (Kusay-Merkle, 2021).

- Als User möchte ich die Möglichkeit haben mich zu registrieren.
- Als User möchte ich die Möglichkeit haben mich anmelden zu können.
- Als User möchte ich die Möglichkeit haben mein Passwort ändern zu können.
- Als User möchte ich persönliche Datenänderungen und Einstellungen vornehmen.
- Als User möchte ich mein Profil teilen können.
- Als User möchte ich mich mit Freunden und Bekannten vernetzen.
- Als User möchte ich ein Rankingsystem mit Freunden und Bekannten einsehen.
- Als User möchte ich die Möglichkeit haben unterschiedliche Spiele auszuwählen.
- Als User möchte ich das Spiel Flappy-Bird starten.
- Als User möchte ich das Spiel Flappy-Bird durch höhere Frequenzen bzw. einer Trillerpfeife steuern.

- Als User möchte ich meine Spielleistungen in Punkte beobachten und messen können.
- Als User möchte ich die Fortschritte meiner Spielleistung in einem Diagramm sehen.
- Als User möchte ich meine Daten und Spielleistungen teilen können z.B. an meinen Hausarzt oder an meine Freunde.

4.2.2 Use-Case-Diagramm

Mit dem Use-Case-Diagramm werden die Funktionalitäten der App aus Sicht des Users beschrieben. Damit sollen die Anwendungsfälle des Users und daran beteiligten Akteuren beschrieben werden.

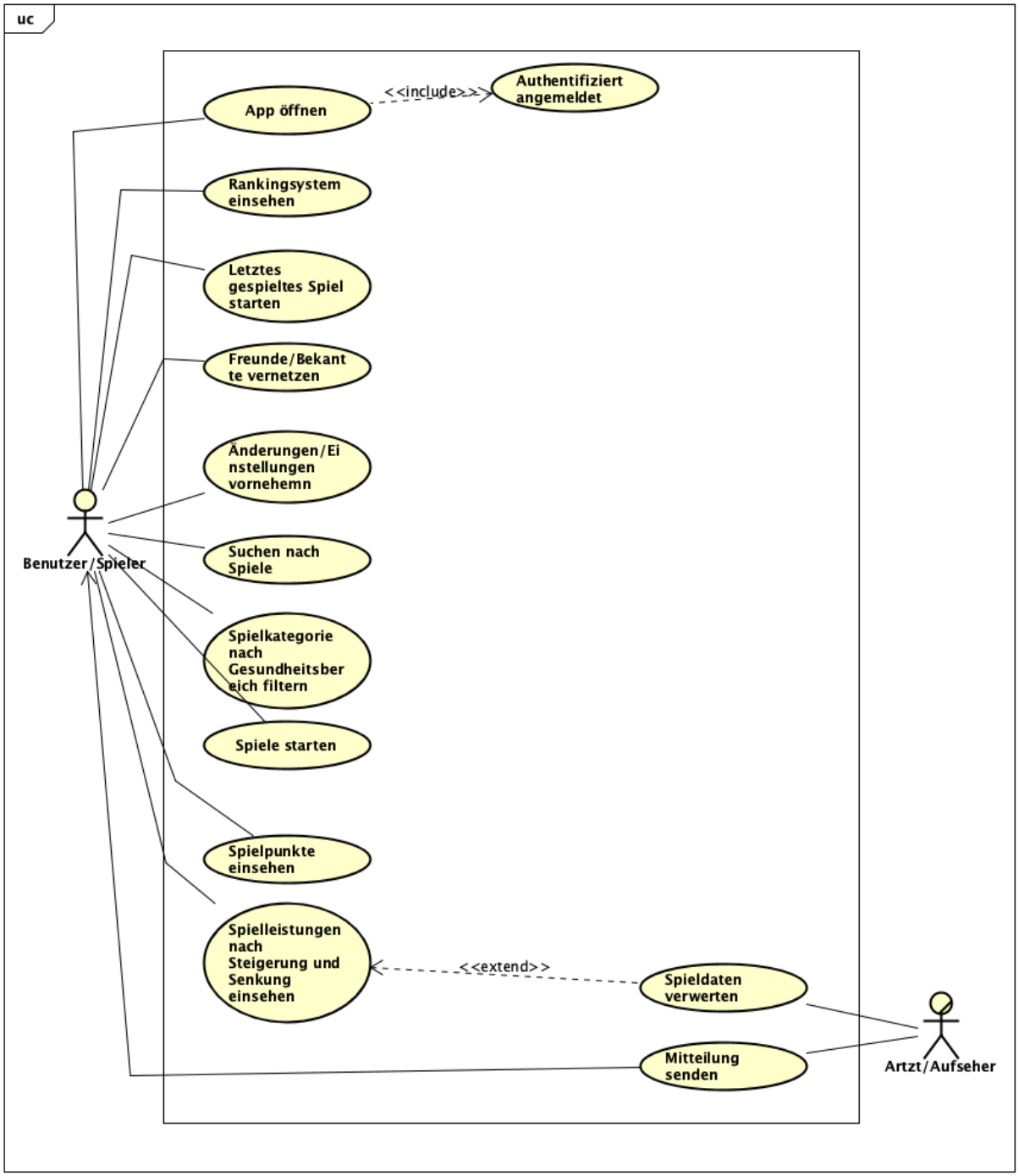

Abbildung 1: Use-Case-Diagramm

Quelle: Eigene Darstellung

App öffnen: Die App kann jederzeit geöffnet und benutzt werden, solange sich der User seine Anmeldung als Voraussetzung durchgeführt hat.

Rankingsystem einsehen: Der User kann jederzeit im Rankingsystem seine Platzierung und die Platzierung aller anderen vernetzten Teilnehmer sehen.

Zuletzt gespieltes Spiel starten: Mit dieser Anzeige kann der User das zuletzt gespielte Spiel weiterspielen.

Freunde/Bekannte vernetzen: Der User kann sich mit Freunden oder Bekannten vernetzen, die ebenfalls diese App benutzen. User können durch eine gezielte Suche nach einem eindeutigen Namen gesucht werden.

Änderungen/Einstellungen vornehmen: Für den User soll die Möglichkeit bestehen seine Daten zu ändern und App Einstellung durchzuführen.

Suche nach Spielen: Nach einem bestimmten Spiel kann der User suchen.

Spielkategorie nach Gesundheitsbereich filtern: Der User kann für verschiede Gesundheitsthemen Spiele filtern und aufrufen.

Spiele starten: Nach Auswahl eines Spiels kann der User das Spiel starten bzw. spielen.

Spielpunkte einsehen: Der User kann nach jedem gespielten Spiel und neuen Leistungen seine neuerzielten Spielpunkte einsehen.

Spielleistungen nach Fortschritten: Für den User sollen seine Spielleistungen für alle Zeiträume detailliert ersichtlich sein, heißt es ist für ihn erkennbar, wann er welche Leistung erbracht hat. Zudem soll es die Option für den User geben, bei Freigabe die Spieldaten an einer Fachkraft z.B. Arzt weiterzuleiten, der die Daten auswerten kann.

Mitteilung senden: Nachdem der Arzt oder Aufseher die Daten ausgewertet hat, kann der User eine Mitteilung empfangen.

4.2.3 UI – Mockups

In diesem Abschnitt gehen wir genauer auf die UI-Mockups ein, die im UX-Design als Essenz gilt und das ganze Konzept detailliert darstellt. Für den User soll die Benutzung vereinfacht und die Visualisierung ansprechend sein. Dies kann umgesetzt werden, in dem die App strukturell, einheitlich, intuitiv und minimalistisch nach Laws of UX aufgebaut wird. Das Laws of UX beinhaltet Gesetze von psychologischen Heuristiken, Phänomen, Prinzipien und Effekte, die als Leitlinie verwendet werden (Yablonski, 2020). Farben stellen ein wichtiges Merkmal als Gestaltungsmittel eines Designs. Durch Farben lassen sich beim Benutzer verschiedene

Emotionen wecken. Farben in Verbindung mit dem UI haben ein Aufteilungsverhältnis von 60%+30%+10%, die als stimmige Farbbalance und Best Practice gilt. Die Primärfarbe definiert die Hauptfarbe mit 60% und die zwei anderen Farben gelten als unterstützend sowie akzentuierend (Müller, 2022). Die Beschaffenheit der App liegt mit einem großen Anteil im Gesundheitsbereich, worin sich sehr gut die Farbe Grün als Primärfarbe widerspiegeln lässt. Die grüne Farbe assoziiert Leben, Wachstum, Glück und weitere Emotionen. Allerdings müssen Teilnehmer mit einer Rot-Grün-Schwäche mit einkalkuliert werden, für die, die die Farben benachteiligend ist und unattraktiv macht. Mit dem Fokus auf die Gamification bietet die Farbe Blau mit den Eigenschaften Sicherheit, Stärke, Besonnenheit und weiteren Assoziationen ein gutes Gesamtpaket für diese App an. Als Akzentfarbe soll die Farbe Orange mitaufgeführt werden da hier, Werte wie Lebensfreude, Optimismus und weitere positive Emotionen aufgegriffen werden (Hahn, 2022). Dadurch entsteht folgende UI-Farbkombination.

Abbildung 2: UI-Farbkombinaton

Quelle: Eigene Darstellung

Wie in der folgenden Abbildung 3 zeigt diese Farbkombination auch eine gute kontrastreiche Visualisierung der App in der hellen sowie in der dunklen Darstellung auf.

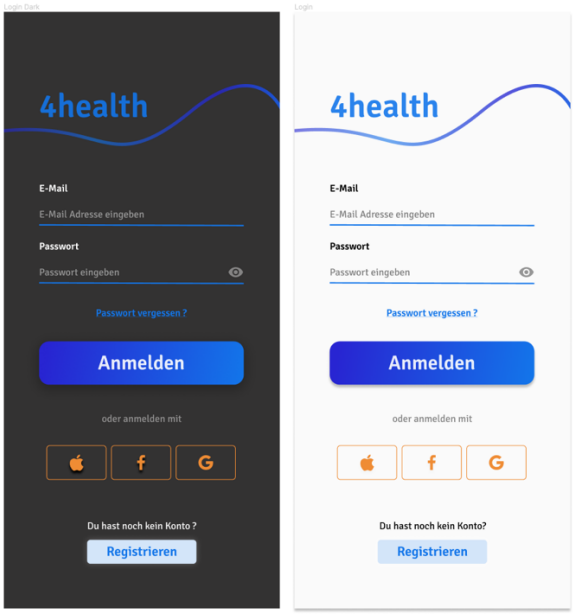

Abbildung 3: Farbkombination in heller & dunkler Darstellung

Quelle: Eigene Darstellung

Im Folgenden werden alle Screens dieser App vorgestellt.

- **Mockup – Login, Register & ForgetPassword Screen**

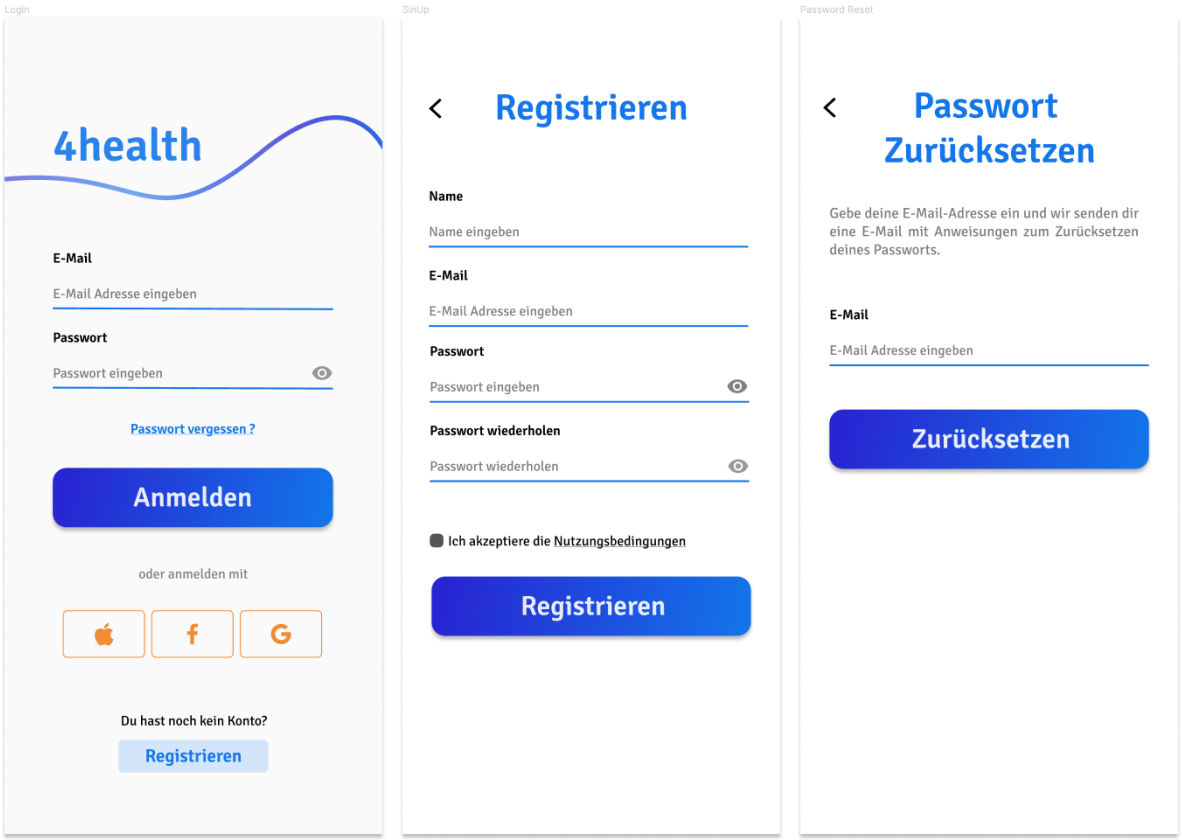

Abbildung 4: Mockup - Login, Register & ForgetPassword

Quelle: Eigene Darstellung

Mit der Primären Farbe Blau soll die Gestaltung dezent gehalten werden. In diesen Farben sollen einfache Buttons wie das Anmelden, Registrieren und Passwort zurücksetzen gestaltet werden.

Unter Login (siehe Abbildung 4: Login Screen) soll es die Möglichkeit geben die Anmeldung durch die Eingabe von E-Mail Adresse und Passwort durchzuführen. Außerdem kann mit einem Sozialen-Netzwerk Zugang die Anmeldung ebenfalls angeboten werden.

Bei Erstanmeldung muss vorher eine Registrierung erfolgen. Die Registrierung erfolgt unter den Button „Registrieren“. Mit Eingabe des Namens und einer noch nicht registrierten E-Mail Adresse kann ein Konto erstellt werden, siehe Abbildung 4 Register-Screen. Voraussetzung für die Registrierung ist auch die Festlegung eines Passworts. Das Passwort wird wiederholend eingegeben, um Fehler bei der Eingabe zu vermeiden. Mit dem Obsecure Icon Button, welcher

neben den Eingabefelder der Passwörter dargestellt wird, können die Passwörter abgeglichen werden.

Im Falle dessen, dass das Passwort vergessen werden sollte, kann über die E-Mail Adresse das Passwort zurückgesetzt werden, siehe Abbildung 4 PasswordForget Screen.

- **Mockup – Home Screen**

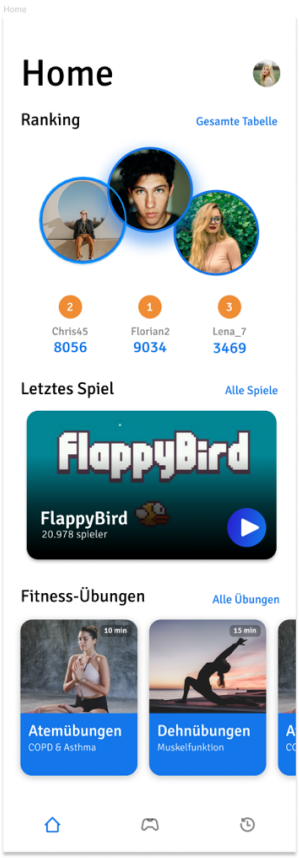

Abbildung 5: Mockup – Home

Quelle: Eigene Darstellung

Nach jeder Anmeldung wird der in Abbildung 5: Mockup-Home aufgezeigte Home-Screen als Startseite angezeigt. Durch das Klicken, oben rechts auf das Profilbild, lässt sich die Seitenleiste öffnen. Der Home Screen ist vertikal gerichtet. Im oberen Abschnitt des Home-Screens wird ein Rankingsystem zwischen Freunde und Bekannte aufgezeigt. Die besten drei Teilnehmer werden direkt angezeigt. Das oben rechts liegende Textbutton „gesamte Tabelle“ öffnet einen neuen Screen, welcher alle Teilnehmer in einem Listenformat nach Rang sortiert auflistet.

In der Mitte des Home-Screens kann das zuletzt gespielte Spiel geöffnet bzw. weitergespielt werden. Wie auch im oberen Abschnitt kann durch das Textbutton „alle Spiele“, alle die in der Applikation integrierten Spiele, in einem separaten Screen angezeigt werden.
Der untere Abschnitt des Home-Screens zeigt ein zusätzliches Feature auf, der den Teilnehmer weitere entsprechende Übungen, zum Therapieren ihrer gesundheitlichen Beschwerden, vorschlägt. Die unterschiedlichen Übungen sollen in Kacheln horizontal aufgeführt werden. Auch hier wird durch das Textbutton „alle Übungen“ in einem neuen Screen alle vorhandenen Übungen angezeigt.

- **Mockup – Drawer & Profile Screen**

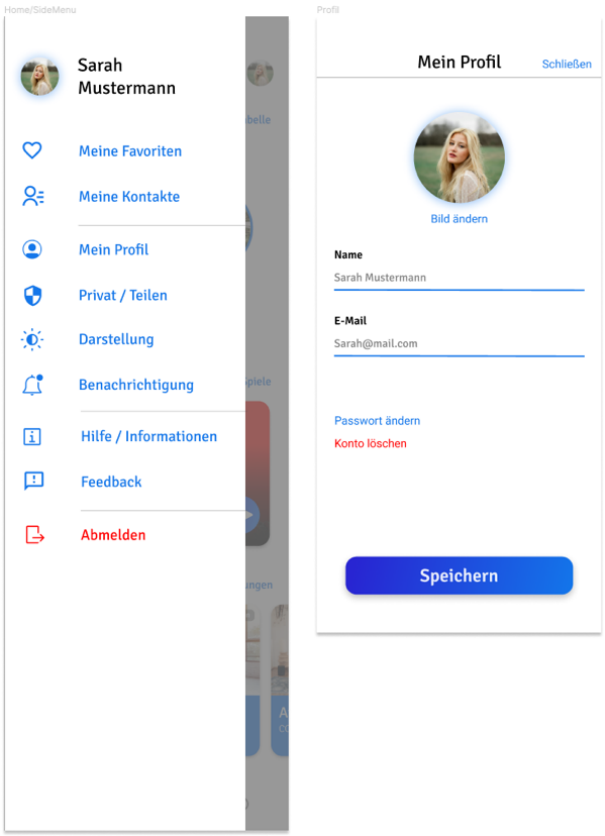

Abbildung 6: Mockup - Drawer & Profile

Quelle: Eigene Darstellung

Die Seitenleiste öffnet sich durch einen Klick auf das Profilbild im Home-Screen. Sie stellt die Möglichkeit zur Verfügung, dass Veränderungen in den Einstellungen der App vorgenommen werden können, z.B. Änderung persönlicher Daten. Bei einem Klick eines der Seitenleisten-Items öffnet sich animiert von unten nach oben ein Modal Screen. Dieser Modal Screen kann entweder durch das Button „schließen“ oben rechts geschlossen werden oder durch das

nachunten schieben. Alle Items werden auf dieser Seitenleiste geöffnet und führen zurück zur Seitenleiste, bis auf das Abmelde-Item, das führt zurück zum Login Screen.

- **Mockup – Games & FlappyBird Screen**

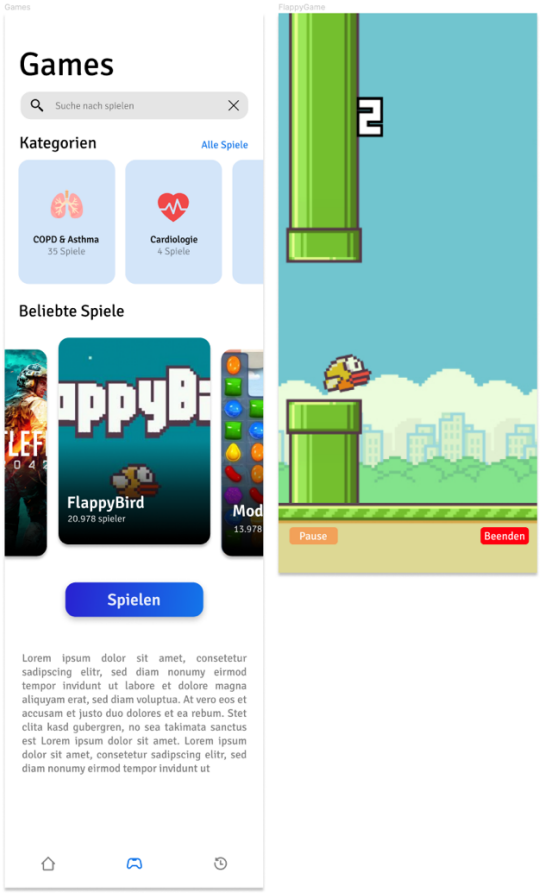

Abbildung 7: Mockup - Games & FlappyBird

Quelle: Eigene Darstellung

Im Games Screen sollen alle Spiele aufgeführt werden. Die obere Ebene zeigt ein Suchfeld auf mit dem nach einem bestimmten Spiel gesucht bzw. gefiltert werden kann. Die nächste Ebene stellt Spiele nach gesundheitlichen Kategorien dar, die in einem Kachel-Format horizontal gescrollt werden kann. Hier besteht erneut die Möglichkeit durch das Textbutton „alle Spiele“ in einem Listformat alle Spiele anzuzeigen. Die unterste Ebene fasst die aktuellen beliebtesten Spiele, mit einem Bild und mit einer kurzen Spielzusammenfassung, auf. Mit dem Button „Spielen“ kann das gewünschte Spiel gestartet werden. Als Beispiel wird hier das Flappy-Bird Spiel angezeigt, das durch einen Klick auf „Spielen“ gestartet werden kann.

- **Mockup – History Screen**

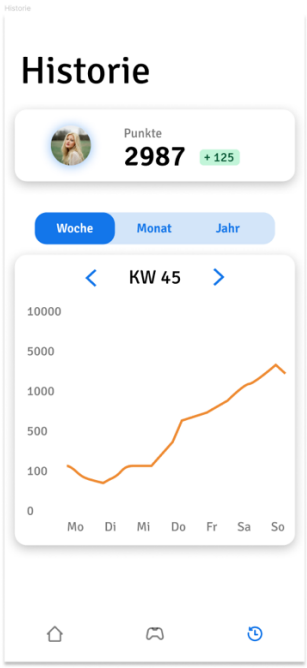

Abbildung 8: Mockup – History

Quelle: Eigene Darstellung

Der History Screen soll die persönlichen Aktivitäten darstellen. Hier wird dem Teilnehmer veranschaulicht, wie seine erbrachten Leistungen in einem von ihm bestimmten Zeitraum waren. Im oberen Abschnitt ist der beste Punktestand der aktuellen Woche ersichtlich. Direkt nebenan ist erkennbar, ob der Teilnehmer im Vergleich zur vergangenen Woche sich verbessert oder verschlechtert hat.

Im unteren Abschnitt wird ein detaillierter Ablauf der erbrachten Leistung in einem Diagramm veranschaulicht, welche nach Woche, Monat oder Jahr visualisierbar ist.

4.2.4 Wireframes – interaktiver Prototyp

Um eine genauere Vorstellung der App zu erhalten, wird hier ein interaktives Modell gebaut, welches durch das Programm Figma erstellt wurde. Die erstellten Mockups bzw. Screens werden hier verbunden.

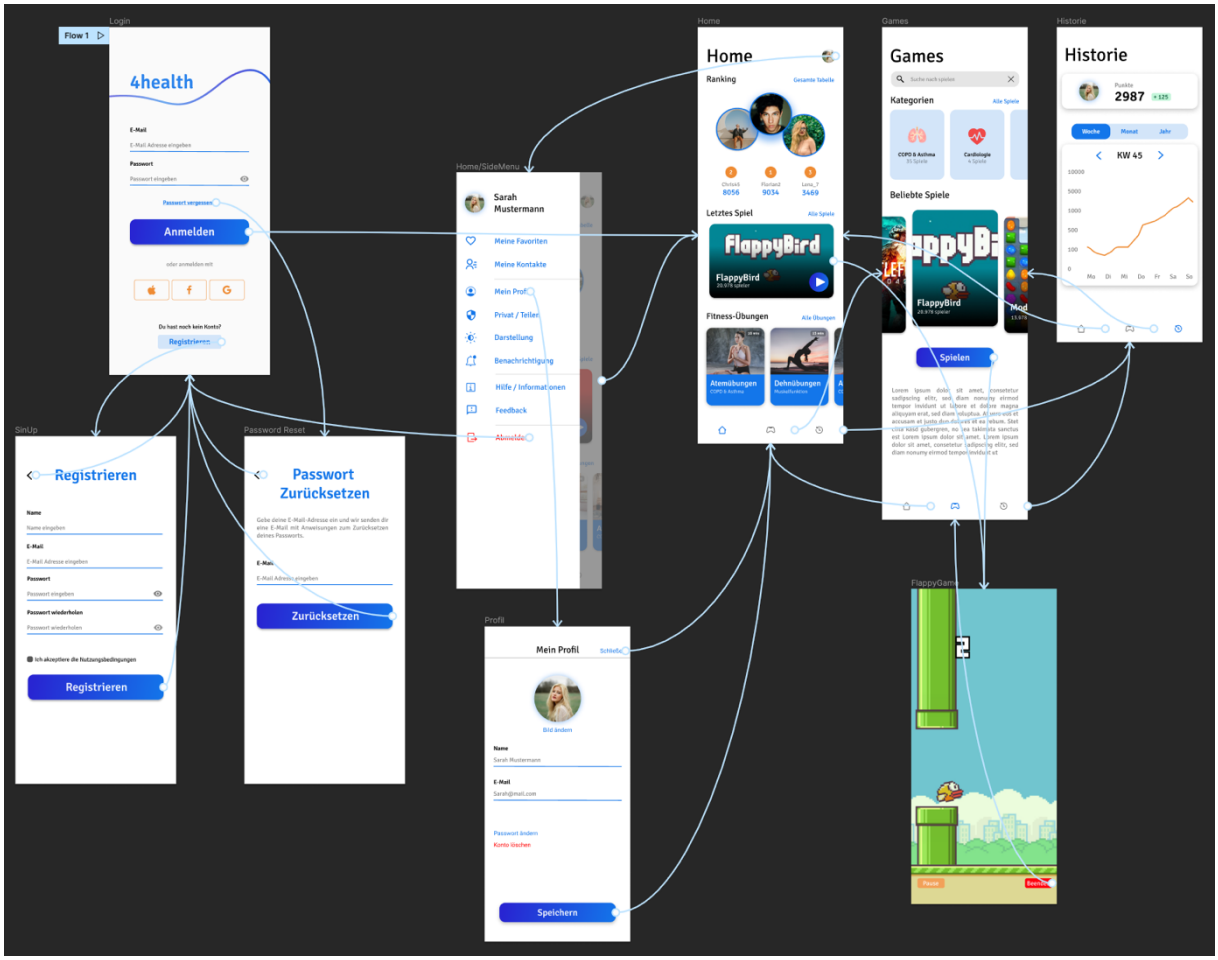

Abbildung 9: UI – Wireframe

Quelle: Eigene Darstellung

Das Mockup, welches durch die Bezeichnung „Flow 1“ links oben betitelt ist, ist der Startpunkt dieses interaktiven Prototyps. Anhand der hellblauen Pfeile sind die Verbindungen zwischen den Mockups ersichtlich. Der Pfeilanfang ist mit einem Kreis erkenntlich, der gleichzeitig auch den Ausgangspunkt definiert und die Navigationsrichtung zum nächsten Mockup zeigt. Dieser interaktive Prototyp kann durch ein Smartphone oder durch das Figma Programm simuliert werden. Damit kann die konzipierte App im Voraus Nutzungserfahrung sowohl als auch Verbesserungspunkte über die Struktur und Designelemente sammeln.

4.3 Verwaltung des Entwicklungscodes mit Github

Der Entwicklungscode wird im Github abgespeichert und kann dadurch unabhängig abgerufen werden. Die Speicherung erfolgt in mehreren Branches mit verschiedenen fortgeschrittenen Entwicklungsständen. Bei den verschiedenen Branches wird das Ziel der Branch definiert und der Entwicklungsablauf bestimmt. Die Branches sind nach Reihenfolge nummeriert und beinhalten vor ihrer Nummer ein T mit der Bedeutung Task.

- **Main**

  Die Main Branch ist die Erste sowohl als auch die Hauptbranch. Diese Branch wird nur zur Veröffentlichung des Codes genutzt bzw. Versionsupdates
- **Dev**

  In der Dev Branch werden sämtliche entwickelnde Branchen zusammengeführt, nachdem das Entwicklungsziel der jeweiligen Branch erreicht wurde. Aus dieser Branch beziehen alle anderen Branchen den aktuellen Entwicklungsstand.
- **T1_Login**

  In diesem Branch soll das Login Screen mit dem entsprechenden Designvorlage erstellt werden.
- **T2_Register_ForgetPassword**

  Wie im vorherigen Branch soll auch hier mit den entsprechenden Designvorlage die Registrierung und das Screen „Passwort zurücksetzen“ erstellt werden.
- **T3_Splashscreen**

  Hier wird ein Screen erstellt, um Ladezyklen abzubilden und Hintergrundprozesse erkenntlich zu machen.
- **T4_Auth_Validation**

  In diesem Branch werden das Login, Register_ForgetPassword und Splashscreen Branchen bearbeitet. In erster Linie wird die Navigation per Klick zwischen den einzelnen Screens „Login“, „Registrierung“ und „Passwort zurücksetzen“ konfiguriert. Als nächster Schritt sollen für die Registrierung, sowie für das Login Verfahren, Authentifizierungs- und Validierungsfunktionalitäten mit einer entsprechenden vorläufigen Backend eingebracht werden. Im dritten Schritt wird das Splashscreen bei Ladevorgängen abgebildet.
- **T5_Drawer**

  Hier soll per Klick eine Seitenleiste abgebildet werden, um sämtliche Einstellungen wie z.B.

Profileigenschaften der App bearbeiten zu können. Die Seitenleiste soll in ein Listformat erstellt werden, um weitere Funktionen in der Zukunft hinzufügen zu können.

- **T6_Mic_Integration**

  Bei diesem Branch soll eine Aufnahmefunktion eingerichtet werden.

- **T7_Phaser**

  In diesem Branch wird das Phaser Paket zum Code implementiert.

- **T8_Game_Integration**

  Nun soll das Spiel „Flappy-Bird“ entwickelt werden, die den Vogel und Hindernisse abbildet. Dementsprechend werden Bewegungen für den Vogel in Richtung der y-Achse nach unten und mit einem Trigger Funktion nach oben eingestellt. Für die Hindernisse wird dasselbe in Richtung der X-Achse eingestellt. Nach den vorläufigen Einstellungen soll das Spiel durch die Aufnahme von hohen Frequenzen getriggert werden.

- **T9_Game_Configuration**

  Im letzten Abschnitt werden Anpassungen und Optimierungen für den Spielfluss umgesetzt.

4.4 Setup der Entwicklungsumgebung

Für die Entwicklung dieser App wird das Ionic-Framework verwendet. Um das Ionic zu nutzen, muss als Voraussetzung Node.js und Angular lokal installiert werden. Zusätzlich werden die Capacitor Komponenten installiert, um den Code nicht nur auf dem Browser widerzuspiegeln, sondern auch auf einem Android oder iOS Simulator bzw. Smartphone zu visualisieren. Für eine iOS Simulation gilt als Voraussetzung das Programm Xcode zu verwenden, welcher nur auf ein Apple Computer umsetzbar ist. Bei der Android Simulation wird als Voraussetzung das Programm Android Studio benötigt, der sich auf Windows, Linux, Chrome und Apple Computer installieren lässt. Anschließend, nachdem alle Ionic abhängigen Komponenten implementiert wurden wird das Projekt mit den NX-Monorepo kombiniert bzw. umhüllt, um weitere projektübergreifende Komponenten oder Codes problemlos einzugliedern. Das gesamte Ionic-Framework befindet sich innerhalb des NX unter dem Pfad „apps/mobile-app“.

5 Entwicklung und Ergebnisse

Die ersten Schritte der Entwicklung ist die Programmierung der Appstruktur, welches durch das Design vorgegeben ist. Im Codeskript werden alle aufgenommenen Spiele unter den Ordner „games“ aufgelistet, das als separater Sektor geführt wird. In dem untergliederten Ordner „flappy-game“ wird das Spiel Flappy-Bird und deren zusammenhängenden Funktionen untergliedert.

Im Folgenden werden aus den entwickelten Codes die Schlüsselfaktoren des Spiels Flappy-Bird und die Aufnahmefunktionen entnommen und genauer behandelt.

5.1 Entwicklung Aufnahmefunktion

Die Aufnahmefunktion wird in der separaten Klasse RecordingFunctions erstellt. Die Klasse wird in zwei Methoden startRecording() und stopRecording() aufgeteilt. Durch die Web API MediaDevice wird ein Zugriff zum Mikrofon des Endgeräts erstellt (MDN, 2022).
Durch die Methode startRecording() wird der mediaDevice API aufgerufen, um eine Schnittstelle zu erstellen, die konkret durch die Methode .getUserMedia() nur auf das Mikrofon zugreift. Daraufhin wird mit der Funktion .then() die Funktion .onSuccess() ausgeführt, die die Konfiguration und die Aufnahmemethoden von Frequenzen beinhaltet. Anschließend werden mit .catch() Fehler aufgefangen und in der Konsole ausgegeben. In der onSuccess() wird mit dem AudioContext() das Audioformat erstellt, welches die Knoten enthält zur Ausführung, Verarbeitung und Decodierung, das letztendlich den Audiostream darstellt. Die Variable frequencyArray beinhaltet die aktuellen Frequenzdaten, die in einem Array von 8-Bit-Ganzzahlen umformatiert wird. Durch die .frequencyBinCount werden überzählige Elemente gelöscht. Die Methode .getByteFrequencyData() kopiert und übergibt die Frequenzdaten von frequencyArray. Auf die Variable volumeInterval wird die Methode setInterval() definiert. Die darin enthaltene Funktion volumeCallback() wird in einem Zeitabstand von 25 Millisekunden aufgerufen. Zusammenfassend wird beim Starten der Aufnahmefunktion alle 25 Millisekunden die Frequenzen aufgenommen und an die Spielklasse „game-scene“ übermittelt. In der volumeCallback() Methode wird mit der Variable averageVolume die aktuelle bzw. momentane Frequenz aufgenommen, welche durch frequencyArray berechnet wird. Der erhaltene Wert calibrationValue wird aus der Seviceklasse LocalStorageService entnommen, dessen

Aufbau im Kapitel 5.3 näher erklärt wird. Die calibrationValue beinhaltet den maximalen Frequenzwert, der zur Kalibrierung aufgenommen wurde. Dieser Wert wird unterschiedlich multipliziert, um unterschiedliche Gewichtungen zu erzielen, die als Parameter im Nachhinein fungieren. Im Folgenden werden die aufgenommen volumeSum Werte mit den kalibrierten Parameterwerten in den if-Abfragen abgeglichen. Daraus resultierend wird die entsprechende Variable volumeValue mit der next() Methode ausgeführt, die mit der Klasse game-scene und der darin enthaltenen Methode birdMotion() interagiert. Bei dieser Interaktion wird je nach aufgenommener Frequenz der Wert mitgegeben, welcher den Vogel weniger höher bis stark höher fliegen bzw. triggern lässt (MDN, 2022).

```
startRecording() {
  navigator.mediaDevices
    .getUserMedia({ audio: true })
    .then(this.onSuccess.bind(this))
    .catch((err) => console.log('Error: ', err));}
private onSuccess(mediaStream: MediaStream) {
  const audioContext = new AudioContext();
  const source = audioContext.createMediaStreamSource(mediaStream);
  const analyser = audioContext.createAnalyser();
  analyser.fftSize = 512;
  analyser.minDecibels = -127;
  analyser.maxDecibels = 0;
  analyser.smoothingTimeConstant = 0.4;
  const frequencyArray = new Uint8Array(analyser.frequencyBinCount);
  const volumeCallback = async () => {
    analyser.getByteFrequencyData(frequencyArray);
    let volumeSum = 0;
    for (const volume of frequencyArray) {
      volumeSum += volume;}
    const averageVolume = volumeSum / frequencyArray.length;
    const calibrationValue = await this.localstorage.getData();
    const cvMIN = calibrationValue[0] * 0.1;
    const cvLowMID = calibrationValue[0] * 0.3;
    const cvHighMID = calibrationValue[0] * 0.5;
    const cvMAX = calibrationValue[0] * 0.8;
    let volumeValue = 0;
    if (averageVolume <= cvMIN) {
      volumeValue = 0;
      this.volume.next(volumeValue);
    } else if (averageVolume > cvMIN+1 && averageVolume <= cvLowMID) {
      volumeValue = 20;
      this.volume.next(volumeValue);
    } else if (averageVolume > cvLowMID+1 && averageVolume <= cvHighMID){
      volumeValue = 50;
      this.volume.next(volumeValue);
    } else if (averageVolume > cvHighMID+1 && averageVolume <= cvMAX) {
      volumeValue = 80;
      this.volume.next(volumeValue);
    } else {
      volumeValue = 120;
      this.volume.next(volumeValue);}
  source.connect(analyser);
  this.volumeInterval = setInterval(volumeCallback, 25);}
```

Code 1: Funktion – startRecording, startet die momentane Aufnahme

Die stopRecording() Methode annulliert die Aufnahmefunktion in dem die Variable volumeInterval durch die Funktion clearInterval() geleert bzw. auf undefined gesetzt wird.

```
stopRecording() {
    console.log('volume interval: ', this.volumeInterval);
    if (this.volumeInterval) {
      clearInterval(this.volumeInterval);
      this.volumeInterval = undefined;
    }
  }
```

Code 2: Funktion – stopRecording, stoppt die momentane Aufnahme

5.2 Entwicklung Game Setup

Für die Entwicklung des Spiels wird das Phaser-Framwork verwendet. Als Spiel wird Flappy-Bird entwickelt. Hierzu werden im Code nur auf die essenziellen Komponenten eingegangen. Die Klasse FlappyGamePage stellt die Ausführungsklasse dar. In dieser Klasse befinden sich die Konfigurationsdaten, um das Spiel zu initialisieren (Martinez, 2022). Das GameScene erstellt die Verbindung zur GameScene Klasse, worin die Spiellogik, Einstellungen und weitere Faktoren des Spiel Flappy-Bird definiert sind.

```
game: Phaser.Game = new Game();
  config: Phaser.Types.Core.GameConfig;
  constructor() {
    this.config = {
      type: Phaser.AUTO,
      width: 400,
      height: 600,
      scale: { mode: Phaser.Scale.ENVELOP },
      physics: {
        default: 'arcade',
      },
      render: { pixelArt: true },
      parent: 'game',
      scene: [GameScene],
    };
  }
  ngOnInit(): void {
    this.game = new Phaser.Game(this.config);
}
```

Code 3: Phaser Konfiguration, Einstellung der Spieloberfläche

Die Klasse GameScene beinhaltet alle Komponenten zum Flappy-Bird Spiel. Es gibt drei Basismethoden preload(), create() und update(). Die Methode preload() ladet alle Bilddateien im Voraus, damit diese später in der create() Methode ohne Ladezeit aufgerufen werden. Die create() Methode dient dazu, die geladenen und allgemeine Phaser Komponenten konkret zu definieren. Die Update Methode ist zuständig für die Aktualisierung und der Neuzeichnung der Spielkomponenten (Phaser, 2022).

Die Initialisierung und Steuerung des Vogels wird in der create() Methode unter der Funktion birdMotion() implementiert. Darin wird der Vogel als sprite() Format aufgerufen mit einer entsprechenden Größe. Mit der auf true gesetzten Funktion .setColliderWorldBounds() wird der Vogel auf alle Komponenten reagieren, bei dem die .collider() Funktion verknüpft ist, wie der Boden oder die Säulen. Die Funktion .setGravityY() setzt den Vogel in einem stetigen vertikal zum Boden ziehenden Zustand. Im nächsten Abschnitt stellen wir die Verknüpfung zur Klasse Recordingfunction und der Methode startRecording() her. Durch die Variable volume werden die aufgenommen Frequenzdaten entnommen und an die Methode .setVelocity() übergeben, die den Vogel vertikal in die Höhe triggert. Der Parameterwert this.birdGravity wird durch das Endgerät(Desktop, iOS, Android) bestimmt, das durch if-Abfragen gefiltert wird. Der definierte Wert von birdGravity lassen sich durch mehrere Testabläufe mit Anbetracht auf den verschiedenen Endgeräten und dessen Volontären volume Werte bestimmen.

```
birdMotion() {
    this.bird = this.physics.add
      .sprite(80, 200, 'bird')
      .setCollideWorldBounds(true)
      .setBodySize(35, 15)
      .setGravityY(this.birdGravity);
    this.bird.body.allowGravity = true;

    this.recorder.startRecording();
    this.recorder.volume$.subscribe((volume) => {
    this.bird.setVelocity(-volume);
    });
  }
```

Code 4: Funktion - birdMotion, Steuerung des Vogels

5.3 Mikrofonkalibrierung

Da in jedem Endgerät unterschiedliche Mikrofone eingebaut sind und dementsprechend unterschiedliche Frequenzwerte bei der Tonaufnahme entnommen werden, ist es nur möglich mit variablen Werten zu agieren. Die Lösung hierbei ist es eine maximale Frequenz aufzunehmen und diesen Frequenzwert für jedes individuelle Endgeräte als einen Standardwert zu definieren.

Mit der Serviceklasse LocalStorageService wird die Grundlage für einen lokalen Speicher erschaffen, welcher den kalibrierten Frequenzwert speichert. Dafür werden drei Methoden kreiert die zum Abrufen (getData()), Abspeichern (addData()) und zum Löschen (removeData()) dient. Hierzu wurde das Storage PlugIn verwendet, welches die Methoden vorkonfiguriert zur Verfügung stellt. Diese Methoden werden in der Klasse MicrophoneCalibrationComponent ausgeführt. Die Ausführung wird im nächsten Abschnitt ausführlich behandelt. (Ionic Storage, 2022)

```
async getData() {
    console.log('GET DATA');
    return this.storage.get(STORAGE_KEY) || [];
  }

  async addData(item: number) {
    const storedData = (await this.storage.get(STORAGE_KEY)) || [];
    storedData.push(item);
    this.isCalibrated = true;
    return this.storage.set(STORAGE_KEY, storedData);
  }

  async removeDate() {
    await this.storage.clear();
    this.isCalibrated = false;
    console.log('Deleted');
  }
```

Code 5: Klasse - LocalStorageService, abrufen, hinzufügen und löschen von Daten in den lokalen Speicher

In der Klasse MicrophoneCalibrationComponent wird die Aufnahme des Kalibrierungswert gestartet. Beim Starten der Kalibrierung werden die Funktionen startCalibration() und stopCalibration() parallel aktiviert. Der Unterschied liegt darin das die stopCalibration() durch die Funktion setTimeOut() erst nach fünf Sekunden ausgeführt wird. Die startCalibration() hat den selben Aufbau bzw. Ablauf der Funktion wie im Kapitel 5.1 Entwicklung Aufnahmefunktion, nur das hier der Wert averrageVolume in dem Array volumeList hinzugefügt wird. Nach fünf Sekunden wird die Aufnahme gestoppt, welche durch die Ausführung der Methode stopCalibration() geschieht. Daraufhin wird aus dem volumeList der maximale Frequenzwert entnommen und an die variable maxVolume definiert. In dem darauffolgendem Ablauf wird die Methode removeStorage() ausgeführt, die bei bereits vorhandenen Wert, diesen löscht und als nächstes mit der Methode pushStorage() den aktuellen aufgenommen maxVolume Wert in den lokalen Speicher hinzufügt. Dieser Wert kann dann global durch die Serviceklasse LocalStorageService abgerufen bzw. gesteuert werden, wie es in dieser MicrophoneCalibrationComponent Klasse umgesetzt wird.

```
startCalibration() {
.
.
  }
private onSuccess(mediaStream: MediaStream) {
.
.
    const volumeCallback = () => {
      analyser.getByteFrequencyData(frequencyArray);
      let volumeSum = 0;
      for (const volume of frequencyArray) {
        volumeSum += volume;
      }
      const averageVolume = volumeSum / frequencyArray.length;
      console.log('average volume: ', averageVolume);
      this.volumeList.push(averageVolume);
.
.
  }
  stopCalibration() {
    setTimeout(() => {
      if (this.volumeInterval) {
        clearInterval(this.volumeInterval);
        this.volumeInterval = undefined;
        console.log('list volume', this.volumeList.join());
        this.maxVolume = Math.max(...this.volumeList);
        console.log('Max volume; ', this.maxVolume);
        this.removeStorage();
        this.pushStorage(this.maxVolume);
      }
    }, 5000);
  }
  async loadStorage() {
    await this.localStorage.getData();
  }
  async pushStorage(item: number) {
    await this.localStorage.addData(item);
    this.loadStorage();
  }
  async removeStorage() {
    this.localStorage.removeData();
  }
```

Code 6: Klasse - MicrophoneCalibrationComponent, Aufnahme der Kalibrierungswert

6 Diskussion

Nun betrachten wir diese Arbeit kritisch auf Darstellung, Umsetzung und Ergebnis. Anschließend werden Problematiken erläutert, welche die Arbeit erschwert haben.

6.1 Anforderungsabgleich

Das Ziel dieser App ist eine weitere Therapiemöglichkeit zur Verfügung zu stellen, um die myofunktionelle Störung zu behandeln. Durch die Entwicklung dieser App wird dem Patienten eine Möglichkeit bereitgestellt, die entsprechende Symptome spielerisch zu therapieren. Das Spiel wird gesteuert durch das Pusten in einer Trillerpfeife, dadurch wird die beanspruchte Muskulatur gefördert. Somit stellt die App eine unterhaltende Therapiemöglichkeit zur Verfügung.

Demnach ist die Grundidee der App bereits vorhanden. Eine benutzerfreundliche UI/UX ist entwickelt, dass als interagierendes Mock-Up konzipiert wurde. Im nächsten Schritt wurde das Spiel Flappy-Bird implementiert. Als nächster Prozess wurden die momentane Aufnahmefunktion und Kalibrierungsfunktion entwickelt. Der letzte Schritt war die aufgenommenen Frequenzen auf die Spielsteuerung anzupassen, welches erfolgreich umgesetzt wurde. Für die Entwicklung und Umsetzung der App kann bestätigt werden, dass die funktionalen Anforderungen erfüllt sind und die nicht funktionalen Anforderungen beschränkt erfüllt sind.

6.2 Limitation

Auf Grund dessen, dass die App erst entwickelt wurde, bestehen noch keine grundlegenden Ergebnisse. Um eine ausführliche Interpretation zu geben, muss die App noch angewendet werden, um ausschlaggebende Kennzahlen zu erhalten.
Die Forschung beschränkt sich auf folgenden Punkt. Die App dient nicht zur Therapierung alle Symptome der myofunktionellen Störung, da durch das Pusten nicht alle Muskeln beansprucht werden können.

Als zusätzliches Feature zum Spiel, sollte ein Ranking-System erstellt werden und eine Vernetzungsmöglichkeit für die Teilnehmer. Die Vernetzung stellt die Möglichkeit zur Verfügung, dass die Anwender untereinander kommunizieren können. Außerdem wird durch das

Ranking-System ebenso der Ehrgeiz des Patienten angekurbelt. Das wiederum hat den positiven Effekt, dass sie häufiger die spielerische Übung ausführen. Allerdings wurde für diese Arbeit keine passende Datenbank gefunden. Dementsprechend war es nicht möglich das Ranking-System, die Vernetzung oder Authentifizierungsfunktionen einzuführen. Eine weitere technische Beschränkung bestand, da als Voraussetzung vorgegeben wurde, mit Ionic-Framework diese App zu entwickeln. Ionic basiert auf Angular, welches in der Regel für Websiteentwicklung genutzt wird. Eine vorteilhaftere Alternative wäre für die App-Entwicklung das Flutter-Framework, da das Flutter auf Mobile App-Entwicklung fokussiert ist. Durch vereinfachtere APIs und weniger erforderlichen Codes ist die Entwicklung effizienter als das Ionic-Framework.

7 Resümee

Abschließend wird die Arbeit zusammenfassend erläutert und mit einem Ausblick auf weitere Entwicklungsmöglichkeiten dargelegt.

7.1 Fazit

Die Forschungsarbeit basiert sich auf die Krankheit myofunktionelle Störung. Die Krankheit sorgt für ein Ungleichgewicht der Muskeln im Mund- und Gesichtsbereich. Durch mundmotorische Übungen wird die Krankheit therapiert z.B. durch Puste-, Blas-, und Aufsage-Übungen. Hierfür befasst sich die Forschungsarbeit mit der Entwicklung einer App, die eine Pusteübung spielerisch darstellt.

Als aller erstes wurde ein Design- und Appentwurf erstellt, der nach den Laws von UX aufgebaut wurde. Danach wurden Vorbereitungsmaßnahmen für die Entwicklungsumgebung der App durchgeführt. Hiermit waren die Voraussetzungen zur Entwicklung der App erfüllt.

Das Spiel „Flappy-Bird“ wurde zur Unterstützung zur Darstellung der spielerischen Therapiemöglichkeit genommen. Im nächsten Ablauf wurde bei der App das Spiel und die Aufnahmefunktion programmiert, weshalb durch das Pusten in einer Trillerpfeife die Bewegung des Vogels (Flappy-Bird) im Spiel ermöglicht wird. Damit wird eine mundmotorische Übung über eine App bereitgestellt.

7.2 Ausblick

Für die zukünftige Entwicklung soll auf jeden Fall die Implementierung der Datenbank mit vorhanden sein, damit dadurch die weiteren Features dieser App verwendet werden können. Ebenso sollen weitere Therapiemöglichkeiten für diverse Krankheiten zur Verfügung gestellt werden. Die App soll damit als eine Plattform fungieren, die eine weitgehende Behandlungsmöglichkeit unterstützen soll.

Der Erfolg der App kann erst nach ausgiebiger Anwendung gemessen werden. Die Grundidee der App ist bereits erfolgreich implementiert. Zusätzliche Feature wie die Vernetzung der Teilnehmer und das zur Verfügung stellen eines Ranking-Systems konnten auf Grund nicht passenden Datenbanken nicht durchgeführt werden, die aber in der zukünftigen Appentwicklung berücksichtigt werden sollen.

Literaturverzeichnis

Angular. (22. 06 2022). *Einführung in die Angular-Dokumentation*. Von Angular: https://angular.io/docs abgerufen

Balzert, H. (2011). *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb* (3 Ausg.). Heidelberg: Spektrum Akademischer Verlag.

Blohm, I., & Leimeister, J. M. (2013). *Gestaltung IT-basierter Zusatzdienstleistungen zur Motivationsunterstützung und Verhaltensänderung.* Wirtschaftsinformatik. Wiesbaden: Springer Fachmedien.

Capacitor. (22. 06 2022). *Cross-platform Native Runtime for Web Apps*. Von Capacitor Docs: https://capacitorjs.com/docs abgerufen

Elfert, U. (17. 06 2022). *Alltagsintegrierte Sprachbildung*. Von Myofunktionelle Störung : https://sprachbildung.net/myofunktionelle-stoerung/ abgerufen

Figma . (17. 06 2022). *Figma* . Von Designfeature: https://www.figma.com/de/design/ abgerufen

Güb, J. (05. 10 2010). *Smashingmagazine*. Abgerufen am 17. 06 2022 von Was ist User-Experience-Design? Überblick, Tools und Ressourcen: https://www.smashingmagazine.com/2010/10/what-is-user-experience-design-overview-tools-and-resources/

Hahn, M. (16. 06 2022). *WebDesign Journal*. Von Farbwirkung Wirkung und Bedeutung von Farben: https://www.webdesign-journal.de/farbwirkung/ abgerufen

Ionic. (20. 06 2022). *Introduction to Ionic*. Von Ionic Docs: https://ionicframework.com/docs abgerufen

Ionic Storage. (23. 09 2022). *ionic-team / ionic-storage*. Von github: https://github.com/ionic-team/ionic-storage abgerufen

Kittel, A., & Förster , N. (29. 04 2016). Myofunktionelle Diagnostik und Therapie. *Der Freie Zahnarzt*, S. 76-86.

Kusay-Merkle, U. (2021). *Themes, Epics, Features, User Stories, Tasks – To-dos in unterschiedlicher Granularität.* Berlin: Springer Gabler .

Müller, C. (15. 06 2022). *Exportarts*. Von Farben im Kontext von User Interface Design: https://www.exportarts.io/blog/farben-im-kontext-von-user-interface-design abgerufen

Martinez, A. (19. 08 2022). *Easy Peasy Setup of a Phaser 3 Project With Ionic*. Von Medium: https://medium.com/swlh/easy-peasy-setup-of-a-phaser-3-project-with-ionic-fb4f4dc01625 abgerufen

MDN. (19. 08 2022). *mdn web docs*. Von AnalyserNode.getByteFrequencyData: https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode/getByteFrequencyData abgerufen

MDN. (19. 08 2022). *MediaDevices*. Von mdn web docs: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices abgerufen

Myolino. (06. 09 2022). *Myolino*. Von Myolino: http://myolino.de/ abgerufen

Neureiter, A. (2019). Arbeitsmotivation und Einstellung gegenüber Gamification am Arbeitsplatz unter Berücksichtigung des Leistungsmotives. *Die Rangliste als zweischneidiges Schwert* (S. 92). Graz: Universitätsbiblithek Graz.

Nodejs. (20. 06 2022). *Über Node.js*. Von Nodejs: https://nodejs.org/en/about/ abgerufen

Nx. (01. 08 2022). *Nx*. Von Nx documentation: https://nx.dev abgerufen

Phaser . (18. 06 2022). *Phaser*. Von Lernen: https://phaser.io/phaser3 abgerufen

Phaser. (19. 08 2022). *Phaser API Documentation*. Von Phaser: https://newdocs.phaser.io/docs/3.55.2/namespaces abgerufen

PhonoLo. (06. 09 2022). *Phono.Lo.* Von phonolo: https://phonolo.de/downloads/phonolo_flyer_elternversion.pdf abgerufen

Ruben, L., & Wittich, C. (01. 01 2014). Evidenzbasierte Behandlung Myofunktioneller Störungen. *Forum Logopädie*, S. 22-29.

Yablonski, J. (2020). *Laws of UX.* Sebastopol: O´Reilly Media.

Yablonski, Jon. (21. 09 2022). *Law of UX*. Von Law of UX: https://lawsofux.com/ abgerufen

InnerMe GmbH. (06. 09 2022). *Leons Dschungelspaß*. Von GameLoop: https://www.gameloop.com/game/educational/de.logoleon.logoleonapp abgerufen

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Bachelorarbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe und die Überprüfung mittels Anti-Plagiatssoftware dulde.

Lindau, 04.10.2022

_____

Kisanthan Nagarasa