FOM Hochschule für Ökonomie und Management

Hochschulzentrum München

Seminararbeit

Im Rahmen des Moduls

Anwendungsprojekt

Über das Thema

STH - Konzept und Implementierung der plattformunabhängigen SportTalentHub App für den Austausch zwischen Spieler und Sportmanager

von

Jonas Waigel, Fitim Makolli, Hasan Deveci und Vatsegkan Zournatsidis

Gutachter: Prof. Dr. Joachim Berlak
Matrikelnummer: 590367, 590827, 587470 und 585568
Abgabedatum: 9. Juni 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

STH-App *Sport Talent Hub App*

IDE *Integrierte Entwicklungsumgebung*

iOS *Internetwork Operating System*

LaTeX *Lamport TeX*

MVP *Minimal Viable Product*

1 Vorwort

Diese Projektdokumentation behandelt das Konzept und die Implementierung der plattformunabhängigen SportTalentHub App (kurz STH) für den Austausch zwischen Spieler und Sportmanager. Dabei werden zunächst die wichtigsten Voraussetzungen wie die Teambildung, die Projektinitiierung und die Projektskizze beschrieben. Das Projektteam besteht aus vier Projektmitgliedern (Jonas Waigel, Fitim Makolli, Hasan Deveci und Vatsegkan Zournatsidis). Zusätzlich hat das Team Jonas Waigel als Projektleiter gewählt. Durch die einzigartigen Fähigkeiten, Erfahrungen, die gemeinsame Vision, eine Smartphone-App zu gestalten und die Leidenschaft für Sport gab dem Team den notwendigen Spirit und die Dynamik, um ein tolles Produkt zu erschaffen.
Viele Recherchen, Überlegungen und die Begeisterung für Sport hat das Projektteam auf die Idee gebracht, eine plattformunabhängige Smartphone-App zu implementieren, die ähnlich zur Social Media App Instagram den Austausch und das Anwerben zwischen Spielern und Sportmanagern ermöglicht.
Dabei war es dem Team stets wichtig, dass zunächst anstehende Aufgaben detailliert besprochen werden und ein gemeinsames Einverständnis über Aufgaben und Themen herrschte. Das Konzept und die Implementierung eines MVP der App soll im Zeitrahmen von 10.03.2024 bis 06.07.2024 entstehen und in dieser Dokumentation beschrieben werden. Das Projekt soll agil in einer auf das Projekt abgewandelten Form von Scrum durchgeführt werden.

2 Ziele

In diesem Kapitel werden die Muss-, Soll-, Kann- und Nicht-Ziele für das Projekt STH-App aufgezeigt.

Muss-Ziele

- Die STH-App muss als plattformunabhängige App mit einen Source-Code implementiert werden, die auf verschiedenen Betriebssystemen und Geräten implementiert (vorzugsweise macOS und Windows) und ausgeführt (vorzugsweise iOS und Android) werden kann.
- Die Chat-Funktion in der STH App muss es ermöglichen, eine sichere Kommunikation zwischen den App-Nutzern aufbauen zu können.
- Die Projektdokumentation muss bis zum 21.06.2024 vom Projektteam komplett fertiggestellt und durch die anderen Projektmitglieder Korrektur gelesen sein.
- Das Projektteam muss bis 15.05.2024 einen fertigen MVP incl. aller Screens in der App bereitstellen.
- Die STH-App für jeden Screen ein einheitliches Design aufweisen und bis zur Präsentation Design-Anpassungen und Verbesserungen durch das Projektteam durchgeführt zu haben.

Soll-Ziele

- Die STH-App soll bis zur Abgabe der Projektarbeit am 06.07.2024 auf verschiedenen Test-Geräten und Simulatoren / Emulatoren getestet werden und Fehler als neue Aufgaben eingestellt werden.
- Gefundene Fehler bei App-Tests sollen bis zum 06.07.2024 durch das Projektteam gelöst werden, um das Projekt gut und mit einer funktionierenden App präsentieren zu können.
- Für die Veröffentlichung sollen die datenschutzrechtliche Themen und Datensicherheit durch weitere Implementierungen oder das Ausweisen wichtiger Dokumente durch das Projektteam gewährleistet sein.

Kann-Ziele

- Nach Abschluss aller Implementierungsschritte und ausreichenden Tests kann die STH App im Apple Appstore oder Google Play Store veröffentlicht werden.

- Bis zu einer offiziellen App Veröffentlichung kann eine Login- und Registrierungsfunktion implementiert werden, die es ermöglicht, nutzerspezifische Inhalte zu speichern, abrufen und filtern zu können.
- Bei regelmäßigen Updates der App kann eine weitere Funktion für ein Hilfe- und Supportcenter angeboten werden.

Nicht-Ziele

- Auf dem Homescreen darf es keine Möglichkeit geben, Posts oder Feeds zu kommentieren.
- Stories und Reels ähnlich wie bei der Social Media App Instagram sollen nicht möglich sein.
- Die STH App soll nicht dazu dienen, Sportmanager bewerten zu können oder Transaktionen bzw. Verträge abzuschließen sondern als Kommunikationsplattform zwischen Spieler und Sportmanager.
- Es dürfen keine Kosten für Spieler oder Sportmanager entstehen, um die App mit allen Funktionen nutzen zu können.
- Es dürfen keine laufenden Kosten wie Lizenzkosten für das Projektteam entstehen.

3 Stakeholderanalyse

Für die STH App sind Stakeholder aus verschiedenen Bereichen vorhanden. Die Applikation erstrebt einen großen Einfluss auf die Sportindustrie und das speziell auf den Prozess des Anwerbens von neuen Fußballspielern durch Vereine in unterschiedlichen Größen.

Deshalb ist eine Stakeholderanalyse besonders wichtig, um die verschiedenen Gruppen an Interessenten zu identifizieren und zufriedenstellen zu können. Für die Analyse werden folgende Gruppen an Stakeholdern betrachtet: Interne-, Externe-, Community- und Technische Stakeholder.

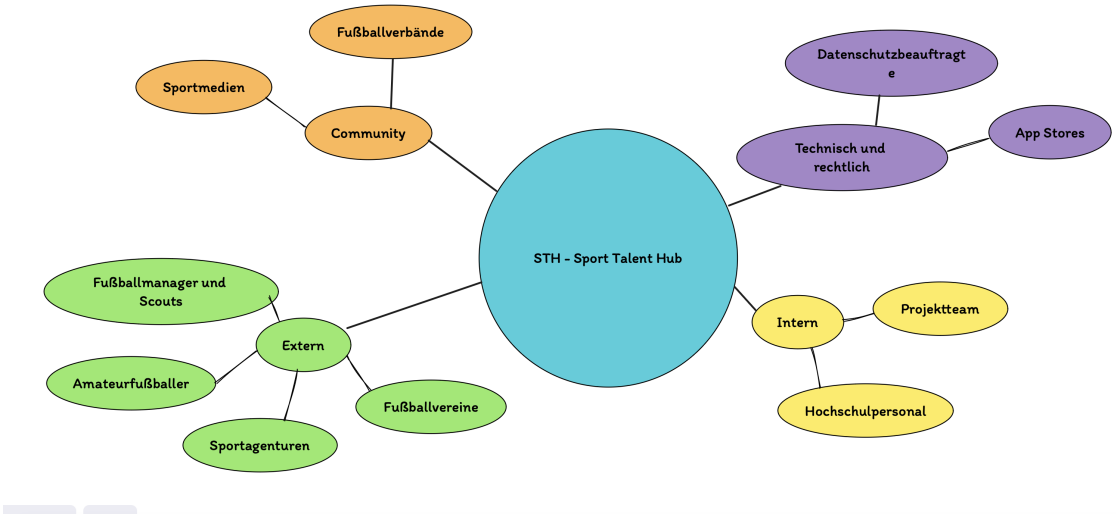
3.1 Vorgehen Stakeholderanalyse

Die Stakeholderanalyse wurde mit einem konkreten Vorgehen erstellt. Zuerst wurde erneut die Größe und Auswirkung der Applikation betrachtet. Dabei ist es auch wichtig, die Zielsysteme nicht außer Acht zu lassen, um konkrete Benutzergruppen definieren zu können.

3.2 Grafische Darstellung Stakeholderanalyse

In der folgenden Abbildung sind die Stakeholdergruppen und deren Abhängigkeiten aufgezeichnet.

Bild 1: Stakeholderanalyse



Quelle: Quelle Stakeholderanalyse

4 Risikoanalyse

Die Risikoanalyse dient dazu, kritische Einflüsse innerhalb und außerhalb des Projektes zu identifizieren. Für die STH App gilt vergleichbar zu allen anderen Projekten, dass der Projekterfolg nur mit begleitenden Risiken ermöglicht werden kann. Für die Klassifizierung und Einschätzungen der Risiken dient eine Risikomatrix. Hierbei wird aufgezeigt, welche Risiken mit welcher Eintrittswahrscheinlichkeit und zugehöriger Auswirkung eintreten können. Anhanddessen kann bewertet werden, welche Risiken besonders laufend beobachtet werden müssen und ob es Risiken gibt, die den Projekterfolg maßgeblich gefährden.

4.1 Vorgehen Risikoanalyse

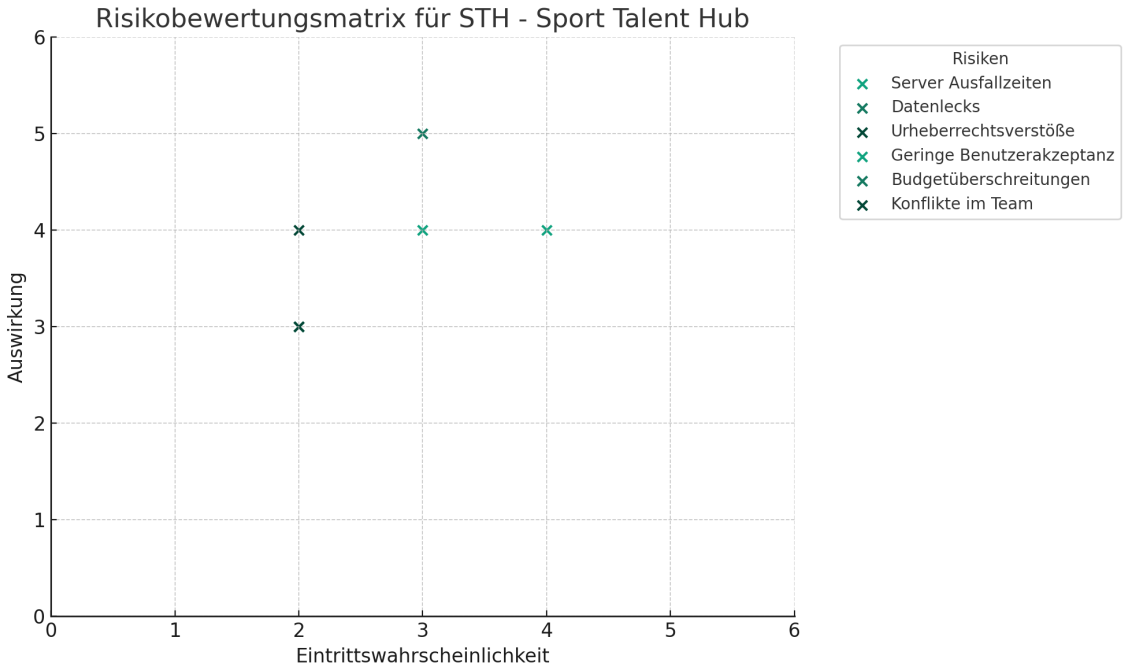
Um eine Risikobewertungsmatrix erstellen und veranschaulichen zu können, benötigt es folgende Schritte. Zunächst müssen die Risiken erkannt werden, die im Projekt auftreten können. Dabei werden sowohl interne als auch externe Faktoren beleuchtet. Zu den internen Risiken gehören z.B. Konflikte im Team und zu den externen Server Ausfallzeiten beim externen Dienstleister. Hierbei ist zu beachten, dass mit jedem vorangegangenen Projektfortschritt auch neue Risiken hinzukommen können. Daraufhin werden die Risiken nach ihrer Eintrittswahrscheinlichkeit klassifiziert. Die Einordnung hilft dabei Risiken zu priorisieren. Dabei werden Risiken mit hohen Eintrittswahrscheinlichkeiten genauer betrachtet und in Zukunft im Blick behalten. Zuletzt werden die Risiken nach ihrer Auswirkung eingestuft. Risiken mit hoher Eintrittswahrscheinlichkeit und hohen Auswirkungen können hierbei zum Scheitern des Projektes bzwżum Nichterfolg führen. Deshalb ist es besonders wichtig diese Art von Risiken nicht nur zu beobachten, sondern kontinuierlich zu messen, welche Folgen der Eintritt haben kann.

4.2 Grafische Darstellung Risikobewertungsmatrix

Die Risikobewertungsmatrix für das Projekt STH App ist in der folgenden Abbildung (Nr einfügen) zu sehen.

Für eine App mit diversen online Funktionen ist ein Server Ausfall fatal. Deshalb wurde dieses Risiko als eines der mit am höchsten verbunden Auswirkungen klassifiziert. Die Eintrittswahrscheinlichkeit bewegt sich hierbei im mittleren Bereich, da nach dem aktuellen Stand der Technik und der Rahmenverträge mit Dienstleistern bzwṘechenzentren bei einem Ausfall meist auf eine redundante Serverlandschaft ausgewichen werden kann. Ein Datenleck, welches durch einen Angriff auf die Backendsysteme entstehen kann, ist für den Erfolg und gleichzeitig für die Auswirkungen einer Smartphone-App schwerwiegend. Innerhalb der STH App können sensible Daten eingegeben und abgespeichert werden, welche nicht an außenstehende gelangen dürfen. Zudem sind die immensen Kosten

Bild 1: Risikomatrix



Quelle: Quelle Risikomatrix

bezüglich der Vertragsstrafen bei der Nichteinhaltung der DSGVO ein großes finanzielles Risiko.
Ein weiteres Risiko für die Applikation sind Urheberrechtsverstöße. Diese können auftreten, indem Nutzer Urheberrechtsgeschützte Inhalte veröffentlichen, welche nicht Ihnen gehören. Eine Begünstigung der Urheberrechtsverstöße könnte auf die Applikation selbst zurückzuführen sein und damit finanziell intensive Vertragsstrafen auslösen.
Eine geringe Benutzerakzeptanz kann durch ein vorher schlecht ausgearbeitetes UI/UX Konzept ausgelöst werden. Die Eintrittswahrscheinlichkeit ist hierbei im mittleren Bereich, da es nicht einfach ist, den Nutzern ein qualitativ hochwertiges und gut durchdachtes User-Interface zu liefern. Die Auswirkung liegt hierbei ebenso im mittleren Bereich, da das Design schnell angepasst werden kann.
Die Budgetüberschreitung ist wie in fast jedem Projekt ein potenzielles Risiko, welches durchaus durch eine schlechte Planung auftreten kann. Sobald die finanziellen Ressourcen ausgeschöpft sind, kann nicht mehr an der Applikation gearbeitet und weiterentwickelt werden. Das könnte unter Umständen zu einem frühzeitigen Scheitern des Projektes führen.
In jedem Projektteam ist immer ein gewisses Konfliktpotenzial vorhanden. Dieses Risiko kann jederzeit und vor allem in Hochphasen wie z.B. kurz vor dem Start der Veröffentlichung der Applikation, auftreten. Allerdings ist das Risiko durch eine gute Projektleitung gut einschätzbar und präventiv vermeidbar bzwżu lösen.

5 Produktanforderungen und technische Architektur

In diesem Kapitel werfen wir einen umfassenden Blick auf die Produktanforderungen und die technische Architektur der STH-App. Die STH-App bietet weit mehr als nur das Teilen von Inhalten. Sie fungiert als Social-Media-Plattform, auf der Sportmanager und Sportler miteinander kommunizieren und Fähigkeiten sowie personenbezogene Profildaten teilen können. Daher ist es von entscheidender Bedeutung, dass die Produktanforderungen einzelner Funktionen und Features separat zur Implementierung bereitgestellt werden, um den unterschiedlichen Anforderungen und Erwartungen beider Gruppen gerecht zu werden. Insbesondere mit der großen Anzahl an Funktionen ermöglicht die STH-App den Nutzern, ihre sportliche Karriere zu teilen und sich dabei mit anderen Benutzern vernetzen können. Der Home-Screen der STH-App bietet einen umfassenden Überblick über die sportlichen Aktivitäten und Leistungen durch Posts der jeweiligen Benutzer. Auf diese Weise können Manager beispielsweise die Profile von Athleten verfolgen. Gleichzeitig ermöglicht er den Sportlern, ihre Fähigkeiten zu veröffentlichen und sich mit anderen Athleten und Sportmanagern anderer Sportvereine zu vernetzen. Der Chat-Screen hingegen bietet für Sportmanager und Athleten einen zentralen Bereich für den direkten Austausch von Textnachrichten über Chatkanäle. Dies ermöglicht eine reibungslose Kommunikation und Zusammenarbeit, indem wichtige Informationen schnell und effizient ausgetauscht werden können, um die gemeinsamen Vereinbarungen und Ziele zu erreichen. Auf dem Search-Screen können Benutzer nach spezifischen personenbezogenen Daten oder Hashtags suchen. So wird beispielsweise das Auffinden neuer Benutzer oder das Suchen nach individuellen Informationen erleichtert, die den eigenen Interessen entsprechen. Beim Profile-Screen können Benutzer die digitale Visitenkarte eines jeden Sportlers oder Managers einsehen. Diese enthält personenbezogene Daten sowie eine Beschreibung der Person, Bilder, Videos und Qualifikationen und ermöglicht es den Benutzern, sich ein umfassendes Bild von anderen Nutzern zu machen und potenzielle Verbindungen zu knüpfen oder Zusammenarbeit zu erleichtern. Die Leistung der STH-App spielt eine ebenso wichtige Rolle, um sicherzustellen, dass die Benutzer ein reibungsloses Erlebnis genießen können. Dabei erfordert es eine effiziente Programmierung und Optimierung sowohl auf dem Frontend als auch auf dem Backend der App. Der Loading-Screen unterstützt dabei, sämtliche Initialisierungsfeatures der STH-App aufzurufen und damit alle verbundenen Services zu starten und zu Initialisieren. Neben den Produktanforderungen ist auch die technische Architektur der App von entscheidender Bedeutung. Hierbei spielt die technische Architektur einer App eine entscheidende Rolle, da sie die Grundlage für Effizienz und Skalierbarkeit bildet. Ein sorgfältig ausgewähltes Set von Technologien und Plattformen ist daher unerlässlich, um sicherzustellen, dass die Anwendung nicht nur reibungslos funktioniert, sondern auch die sich ständig wandelnden Anforderungen der Benutzer erfüllen kann. Als Entwicklungsplattform dient Visual

Studio Code, eine leistungsstarke IDE, die speziell für die Entwicklung von Flutter-Apps optimiert ist. Visual Studio Code bietet eine Fülle von Funktionen und Erweiterungen, die den Entwicklungsprozess beschleunigen und vereinfachen. Die klare Definition der Produktanforderungen sowie die Festlegung der Arbeitspakete über GitHub-Commits, in denen die Aufgabenstellung detailliert beschrieben wird, bilden zusammen mit der soliden technischen Architektur das Fundament für eine effektive Zusammenarbeit in der Entwicklung und Dokumentation des Projekts. Flutter selbst fungiert als Cross-Platform-Framework für die Entwicklung des Frontends als auch der Schnittstelle zum Backend der STH-App. Die Verwendung von Flutter bietet eine Reihe von Vorteilen, darunter eine hohe Leistung, schnelle Entwicklung und einfache Wartung. Für das Backend der STH-App wird Firebase genutzt, eine Plattform von Google, die eine breite Palette von Diensten für die Entwicklung von Web- und Mobile-Apps bietet. Mit Funktionen wie Echtzeitdatenbanken, Authentifizierungsdiensten und Cloud-Speichern bietet Firebase eine robuste Lösung für die Anforderungen der STH-App.

6 Teilprojekte und Arbeitspakete

In diesem Kapitel liegt der Fokus auf den Arbeitspaketen, die für die Entwicklung der STH-App von entscheidender Bedeutung sind. Es wird zunächst erläutert, welche Aufgabenstellung jedes Arbeitspaket initialisiert, wobei sowohl die Verantwortlichen des Arbeitspakets als auch die damit verbundenen Ressourcen betrachtet werden.

Definition der Arbeitspakete:

Arbeitspaket 1 vom 08.03.2024 bis 11.03.2024

- Bezeichnung: Initial Setup Flutter
- Verantwortliche/r: Das Team
- Ressourcen: VS Code, Flutter SDK, Android Emulator / iOS Simulator
- Aufgabenstellung: Einrichtung der Entwicklungsumgebung für Flutter

Arbeitspaket 2 vom 08.03.2024 bis 11.03.2024

- Bezeichnung: Initial Setup LaTeX
- Verantwortliche/r: Das Team
- Ressourcen: VS Code inclLaTeX-Plugins, Container-Plattform (Docker)
- Aufgabenstellung: Einrichtung der Entwicklungsumgebung für LaTeX

Arbeitspaket 3 vom 08.03.2024 bis 11.03.2024

- Bezeichnung: Initial Setup GitHub
- Verantwortliche/r: Das Team
- Ressourcen: Git-Client, GitHub-Konto
- Aufgabenstellung: Erstellung des Repositories auf GitHub

Arbeitspaket 4 vom 13.03.2024 bis 14.03.2024

- Bezeichnung: Project Roadmap
- Verantwortliche/r: Das Team
- Ressourcen: Brainstorming
- Aufgabenstellung: Definition der App Funktionalitäten

Arbeitspaket 5 vom 14.03.2024 bis 15.03.2024

- Bezeichnung: Project Structure
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Erstellung der ersten Projektstruktur in Visual Studio Code / Flutter

Arbeitspaket 6 vom 16.03.2024 bis 23.03.2024

- Bezeichnung: Home Screen
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung des Home-Screens inclMenüleiste (Bottom Navigationbar) am unteren Bildschirmrand und App Bar am oberen Bildschirmrand

Arbeitspaket 7 vom 18.03.2024 bis 01.04.2024

- Bezeichnung: Chat Screen
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung des Chat-Screens

Arbeitspaket 8 vom 21.03.2024 bis 04.04.2024

- Bezeichnung: Account Profile
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung des Account-Profile-Screens inklusive Avatar und personenbezogene Daten

Arbeitspaket 9 vom 23.03.2024 bis 06.04.2024

- Bezeichnung: Search Screen
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: ChatGPT 4.0
- Aufgabenstellung: Implementierung des Search-Screens

Arbeitspaket 10 vom 23.03.2024 bis 31.03.2024

- Bezeichnung: Global Menu Bar
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer globalen Menüleiste am unteren Bildschirmrand für alle Pages

Arbeitspaket 11 vom 24.03.2024 bis 07.04.2024

- Bezeichnung: Initial Setup Backend
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework, Backend-Plattform (Firebase)
- Aufgabenstellung: Backend-Initialisierung (Firebase)

Arbeitspaket 12 vom 24.03.2024 bis 31.03.2024

- Bezeichnung: Profile Edit Button
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines Bearbeitungsbuttons für den Account-Profile-Screen

Arbeitspaket 13 vom 24.03.2024 bis 31.03.2024

- Bezeichnung: Logo Design
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: ChatGPT 4.0
- Aufgabenstellung: Gestaltung eines Logos für die STH-APP

Arbeitspaket 14 vom 25.03.2024 bis 04.04.2024

- Bezeichnung: Loading Screen
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer Ladebildschirm-Funktion beim Start der STH-App

Arbeitspaket 15 vom 27.03.2024 bis 06.04.2024

- Bezeichnung: GetStream Chat
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung von GetStream-Chat

Arbeitspaket 16 vom 27.03.2024 bis 28.03.2024

- Bezeichnung: Chat App Bar
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Anpassung der App-Leiste auf dem Chat-Bildschirm

Arbeitspaket 17 vom 29.03.2024 bis 04.04.2024

- Bezeichnung: Logo Loading Screen
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines Ladebildschirms mit einem Logo

Arbeitspaket 18 vom 31.03.2024 bis 07.04.2024

- Bezeichnung: Backend Config
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework, Backend-Plattform (Firebase)
- Aufgabenstellung: Implementierung der Backend-Konfiguration (Firebase), Authentifizierung und deren Methoden

Arbeitspaket 19 vom 01.04.2024 bis 07.04.2024

- Bezeichnung: Custom Page Routing
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer benutzerdefinierten Seitenroute beim Wechsel von Pages

Arbeitspaket 20 vom 02.04.2024 bis 08.04.2024

- Bezeichnung: Chat UI Adjustments
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Anpassung des Chat-Screens und der benutzerdefinierte App-Leiste

Arbeitspaket 21 vom 02.04.2024 bis 08.04.2024

- Bezeichnung: Channel Adjustment
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Anpassung des Chanels im Chat-Screen

Arbeitspaket 22 vom 02.04.2024 bis 03.04.2024

- Bezeichnung: Latex Docs Update
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code
- Aufgabenstellung: Aufgabenstellung: Dokumentationsstruktur in LaTeX aktualisieren

Arbeitspaket 23 vom 03.04.2024 bis 10.04.2024

- Bezeichnung: Local Storage Setup
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines lokalen Speichers für den Account-Profile-Screen

Arbeitspaket 24 vom 03.04.2024 bis 10.04.2024

- Bezeichnung: Save/Load Functions
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer Funktion zum Speichern/Laden für den Account-Profile-Screen

Arbeitspaket 25 vom 03.04.2024 bis 10.04.2024

- Bezeichnung: RegEx Guidelines
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung von RegEx-Richtlinien in Bezug auf personenbezogene Daten

Arbeitspaket 26 vom 03.04.2024 bis 15.04.2024

- Bezeichnung: Hovering Function
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer Hovering-Funktion für personenbezogene Daten

Arbeitspaket 27 vom 03.04.2024 bis 15.04.2024

- Bezeichnung: Profile UI
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung des Profile-Screens mit Avatar, Bilder und Mediathek

Arbeitspaket 28 vom 10.04.2024 bis 18.04.2024

- Bezeichnung: Media Upload
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung von Funktionen zum Auswählen und Hochladen von Bilder und Videos

Arbeitspaket 29 vom 10.04.2024 bis 18.04.2024

- Bezeichnung: Profile Local Storage
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines lokalen Speichers für den Profile-Screen

Arbeitspaket 30 vom 10.04.2024 bis 18.04.2024

- Bezeichnung: Profile Save/Load
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer Funktion zum Speichern/ Laden für den Profile-Screen

Arbeitspaket 31 vom 10.04.2024 bis 11.04.2024

- Bezeichnung: App Bar Navigation
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Anpassung der App-Leiste für die Navigation und das Verhalten beim Seitenwechsel

Arbeitspaket 32 vom 13.04.2024 bis 19.04.2024

- Bezeichnung: Shared Preferences
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework, Backend-Plattform (Firebase)
- Aufgabenstellung: Verwendung von Shared Preferences zusammen mit Firebase-Testdaten

Arbeitspaket 33 vom 14.04.2024 bis 15.04.2024

- Bezeichnung: Page Linking
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Verlinkung der Pages von Profile-Screen und Account-Profile-Screen

Arbeitspaket 34 vom 16.04.2024 bis 17.04.2024

- Bezeichnung: Loading Screen Update
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Aktualisierung des Loadingscreens

Arbeitspaket 35 vom 16.04.2024 bis 25.04.2024

- Bezeichnung: Start Page Logo
- Verantwortliche/r: Herr Vatsegkan Zournatsidis
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Hinzufügen vom Logo auf die Startseite

Arbeitspaket 36 vom 21.04.2024 bis 28.04.2024

- Bezeichnung: Media Button
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines Buttons für das Hochladen von Bilder und Videos

Arbeitspaket 37 vom 21.04.2024 bis 01.05.2024

- Bezeichnung: Avatar Transmission
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework, Backend-Plattform (Firebase)
- Aufgabenstellung: Implementierung von Funktionen zur Übertragung vom Avatar zum Backend

Arbeitspaket 38 vom 25.04.2024 bis 05.05.2024

- Bezeichnung: Media Transmission
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework, Backend-Plattform (Firebase)
- Aufgabenstellung: Implementierung von Funktionen zur Übertragung von Bilder und Videos an das Backend

Arbeitspaket 39 vom 27.04.2024 bis 28.04.2024

- Bezeichnung: Page Navigation
- Verantwortliche/r: Herr Jonas Waigel
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Anpassung der Seiten-Navigation und Entfernung von Animationen beim Wechsel

Arbeitspaket 40 vom 28.04.2024 bis 29.04.2024

- Bezeichnung: Latex Docs Update
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code
- Aufgabenstellung: Dokumentationsstruktur in Latex aktualisiert

Arbeitspaket 41 vom 29.04.2024 bis 05.04.2024

- Bezeichnung: Chat Routing
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung der Seitenroute vom Chat zum Chanelscreen

Arbeitspaket 42 vom 29.04.2024 bis 05.04.2024

- Bezeichnung: Profile Navigation
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung der Navigationsroute vom Chat-Screen zum Profile-Screen

Arbeitspaket 43 vom 30.04.2024 bis 10.05.2024

- Bezeichnung: Stream Headers
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung des Stream-Channel-Headers

Arbeitspaket 44 vom 30.04.2024 bis 10.05.2024

- Bezeichnung: Username Update
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung der Aktualisierung des Benutzernamens auf dem Profile-Screen

Arbeitspaket 45 vom 30.04.2024 bis 07.05.2024

- Bezeichnung: Action Button
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung einer Aktionsschaltfläche mit Benutzerintegration

Arbeitspaket 46 vom 30.04.2024 bis 07.05.2024

- Bezeichnung: Test User Addition
- Verantwortliche/r: Herr Hasan Deveci
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung neuer Chat-User als Testdaten

Arbeitspaket 47 vom 05.05.2024 bis 15.05.2024

- Bezeichnung: Hashtag Implementation
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung von Hashtags im Profile-Screen

Arbeitspaket 48 vom 06.05.2024 bis 15.05.2024

- Bezeichnung: Hashtag Storage
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung eines lokalen Speichers für die Hashtags

Arbeitspaket 49 vom 06.05.2024 bis 15.05.2024

- Bezeichnung: Hashtag Transmission
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Implementierung von Funktionen zur Übertragung von Hashtags an das Backend

Arbeitspaket 50 vom 07.05.2024 bis 20.05.2024

- Bezeichnung: Major Bug Fixes
- Verantwortliche/r: Herr Fitim Makolli
- Ressourcen: VS Code, Flutter Framework
- Aufgabenstellung: Fehlerbehebungen für den Profile-Screen und Account-Profile-Screen

7 Meilensteinplanung

Im Rahmen dieses Projekts war es erforderlich, eine detaillierte Planung und Organisation der Arbeitspakete vorzunehmen. Um dies zu erreichen, haben wir die Tabelle mit den Arbeitspaketen analysiert und ein Gantt-Diagramm erstellt. Das Gantt-Diagramm bietet eine visuelle Darstellung der zeitlichen Abfolge und Dauer der verschiedenen Aufgaben, was die Projektplanung und -kontrolle erleichtert.

7.1 Vorgehensweise zur Erstellung des Gantt-Diagramms

Zunächst haben wir die Daten aus der Tabelle extrahiert und aufbereitet. Die Tabelle enthielt drei wesentliche Spalten: Arbeitspaket vom, Beschreibung und Dauer. Diese Daten wurden in ein geeignetes Format gebracht, um sie in das Gantt-Diagramm-Tool einzugeben. Zur Erstellung des Gantt-Diagramms wurde die Website `onlinegantt.com` genutzt. Diese Plattform bietet eine benutzerfreundliche Oberfläche zur Eingabe und Visualisierung von Projektplänen. Für jedes Arbeitspaket wurde das Startdatum entsprechend der Tabelle eingegeben. Die Beschreibung der Aufgaben wurde als Titel für die jeweiligen Arbeitspakete verwendet. Die Dauer der Aufgaben wurde in Tagen oder Wochen entsprechend der Tabelle angegeben. Nachdem alle Daten eingegeben wurden, generierte die Plattform das Gantt-Diagramm. Dieses Diagramm ermöglicht es, die zeitliche Abfolge der Aufgaben zu visualisieren und Überschneidungen oder Abhängigkeiten zwischen den Aufgaben zu erkennen.

7.2 Analyse der Zeitplanung

7.2.1 Gesamtübersicht

Die Erstellung des Gantt-Diagramms zeigte eine klare Struktur des Projekts, beginnend am 08. März 2024 und endend am 30. April 2024. Die Aufgaben sind logisch und sequenziell angeordnet, was die Nachverfolgung und Kontrolle erleichtert.

7.2.2 Dauer und Überlappungen

Einige Aufgaben, wie die Einrichtung des Repositories auf GitHub oder die Erstellung der Projektstruktur, hatten eine kurze Dauer von nur einem Tag. Andere Aufgaben, wie die Implementierung des Chat-Screens oder die Anpassung der Seiten-Navigation, dauerten mehrere Wochen. Diese längeren Aufgaben wurden parallel zu kürzeren Aufgaben geplant, um die Gesamtzeit des Projekts effizient zu nutzen.

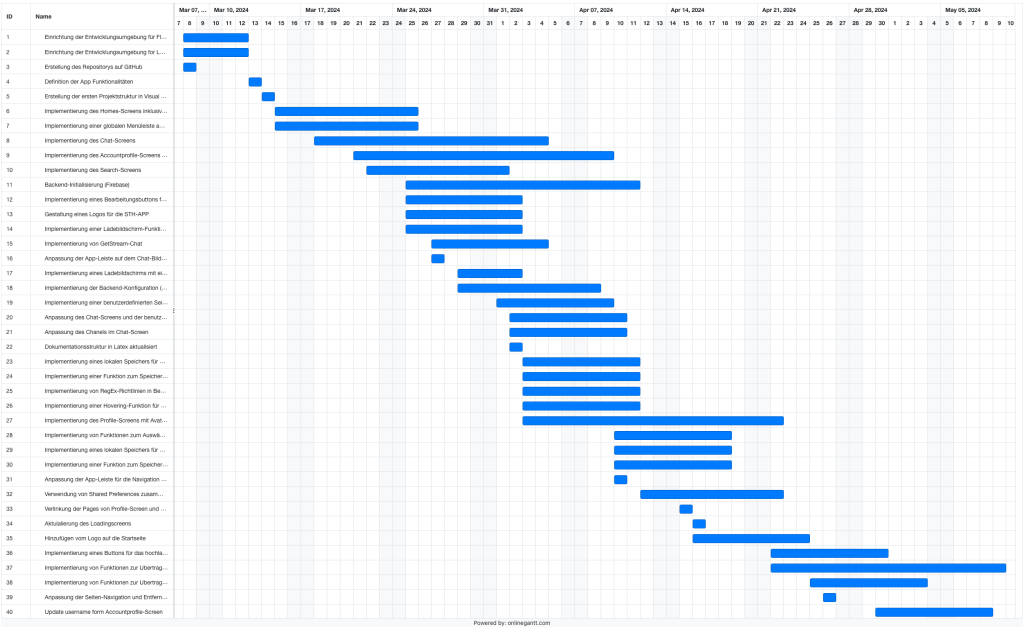
7.2.3 Wichtige Meilensteine

Einige wesentliche Meilensteine des Projekts sind: Die ersten beiden Wochen des Projekts waren der Einrichtung der Entwicklungsumgebungen und der Definition der App-Funktionalitäten gewidmet. Der Großteil des Projekts, von Mitte März bis Ende April, bestand aus der Implementierung verschiedener Funktionen und der Backend-Integration. Die letzten Tage des Projekts wurden für Feinanpassungen und Updates, wie die Aktualisierung des Loadingscreens und das Update des Usernames auf dem Accountprofile-Screen, verwendet.

7.2.4 Risikobewertung

Durch das Gantt-Diagramm konnten potenzielle Engpässe und Risiken identifiziert werden. Aufgaben, die kritische Pfade darstellen, wurden besonders beachtet, um Verzögerungen zu vermeiden.

Bild 1: Meilensteinplanung



Quelle: https://www.onlinegantt.com

8 Initiale Einrichtung der Tools für das STH-Projekt

Dieses Kapitel beschreibt die initiale Einrichtung des Flutter-Code-Repositories, die Nutzung von Google Firebase und die Verwendung vorhandener Microsoft Teams Add-On Tools für die Projektorganisation.

Microsoft Teams Tools für die Organisation des Projektes und der Aufgaben für das Projektteam

Im Projektstart-Meeting hat das Projektteam zusätzlich zur Ideenfindung und Projektteam-Aufteilung entschieden, dass das STH-Projekt mit agilen Projektmanagement-Methoden angelehnt an Scrum durchgeführt werden soll. Zunächst wurde dafür ein Tool gesucht, das sowohl die Dokumentation von besprochenen Themen als auch die Planung von Aufgaben und die Kommunikation im Projektteam ermöglicht. Nach einigen Recherechen und Diskussionen hat sich das Projektteam für Microsoft Teams entschieden. Teams bietet mit seinen Add-On Tools die benötigten Möglichkeiten:
Ergebnisse aus Diskussionen und Weekly-Standup-Meetings können mit Microsoft OneNote dokumentiert werden. Aufgaben für das Projektteam können mit dem Add-On Microsoft Planner in einem Kanban Board übersichtlich dargestellt werden und der jeweilige Aufgabenstatus kann von jedem Projektmitglied eingesehen und bearbeitet werden. Zusätzlich wurde ein Team in Microsoft Teams erstellt, in dem die Kommunikation im Projektteam stattgefunden hat und Neuerungen oder Probleme bei der Entwicklung besprochen werden konnten.

Einrichtung des Flutter-Code-Repositories für das STH-Projekt

Im nächsten Schritt hat das Projektteam im Projektstart-Meeting ein Framework bzwėine Programmiersprache für das STH-Projekt gesucht, die sowohl eine gute Dokumentation, eine leicht verständliche Programmiersprache, Plattformunabhängigkeit für iOS/Android und die Kompatibilität zum Kompilieren der Applikation für die Betriebssysteme Windows und MacOS bietet. Durch bereits vorhandenes Know-How wurde das Google-Framework Flutter gewählt.
Flutter ist ein Open-Source-Framework von Google, das eine Frontend-Entwicklung von Benutzeroberflächen für Apps bietet. Dabei gilt das Prinzip One-Codebase, welches bedeutet, dass Anwendungen für die unterschiedlichen Betriebssysteme iOS und Android aus einem Sourcecode kompiliert werden.

Für die Einrichtung des Repositories wird zunächst das Github Projekt "anwendungsprojekt_fom" erstellt und das Projektteam freigegeben. Im nächsten Schritt wurde auf den Entwicklungsrechnern der Projektmitglieder die Entwicklungsumgebung Visual Studio Code für die Code-Entwicklung und das Framework Flutter der Version 3.19.3 installiert und eingerichtet. "Flutter" bietet außerdem die Möglichkeit mithilfe des "flutter create" Terminal Commands ein initiales Flutter-Projekt erstellen zu lassen. Dieses initiale Setup beinhaltet eine voll funktionsfähige Beispiel-App, die für den ersten Test des Setups erfolgreich für Android und iOS kompiliert werden konnte. Für einen erfolgreichen iOS-Build wird zudem ein Apple-Gerät mit dem Betriebssystem macOS und die Apple-Software xCode benötigt, um iOS-spezifische Einheiten der App kompilieren zu können. Für einen sogenannten "Clean Code", der für alle Projektmitglieder gut lesbar und verständlich ist, wurde festgelegt, dass für jede Änderung im Code der "Flutter Formatter (Terminal Command: dart format -l 120)ünd der Dart Analyzer (Terminal Command: dart analyze && dart fix –apply)" ausgeführt werden muss.

Google Firebase Tools als Backend

Firebase ist ein Cloud Service, der von Google bereitgestellt wird. Dieser bietet folgende wichtige Funktionen, die für die STH App essentiell sind:
- Der Cloud Firestore ist eine Cloud-Datenbank, in der verschiedene Werte von Variablen gespeichert werden können.
- Der Cloud Storage ist ein Cloud-Speicher, in dem Dateien unterschiedlicher Dateiformate hinterlegt werden können.
- Die Chat- und Kommunikationsfähigkeit
Über den TerminalCommand "flutter pub add" inclden jeweiligen Flutter packages / dependencies $firebase\_auth, firebase\_core, firebase\_storage, cloud\_firestore$ wurden Flutter Packages dem Projekt hinzugefügt und können verwendet werden.

Zentrale Funktionen für Firebase, Shared Preferences, Flutter Page Routing, Flutter Bottom Navigationbar und Flutter App Bar

Zusätzlich wurden Funktionen, die an anderen Stellen des Codes wiederverwendet werden, zentralisiert. Dazu wurde im Repository ein Ordner "technical" angelegt und dieser in die jeweilige Grundfunktionalitäten unterteilt:

Firebase

- Authentifizierungsmethoden (Login / Register)
- Datei-Upload und -Download Funktionen
- Upload und Download von Variablenwerten

Shared Preferences

- Lokale Speicherung von Daten nach Download aus Firebase

- Update Funktion von editierten Daten

Custom Page Route

- Einmalige zentrale Konfiguration des Routing-Verhaltens zwischen Screens
- Flutter PageRouteBuilder inclġenerateRoute Funktion und neuer Screen als Übergabeparameter

Custom App Bar

- Einmalige zentrale Konfiguration der Flutter App Bar für ein einheitliches Design
- Boolean-Übergabeparameter für die Anzeige von Buttons/Icons
- String-Übergabeparameter für den Seitentitel

Custom Bottom Navigationbar

- Einmalige zentrale Konfiguration der Flutter Bottom Navigationbar für ein einheitliches Design
- Hervorhebung der aktuell ausgewählten Seite in der App
- Routing bei Auswahl eines Buttons in der Navigationbar

9 Ladebildschirm

Der Ladebildschirm ist ein essenzieller Bestandteil der Benutzererfahrung unserer STH-App (SportTalentHub). Er erscheint, bevor der Nutzer die Hauptfunktionen der App nutzen kann, und dient als Zwischenbildschirm, um dem Nutzer zu signalisieren, dass die App lädt.

Ziele des Ladebildschirms

Der Ladebildschirm hat mehrere wichtige Funktionen. Er zeigt dem Nutzer visuelles Feedback, dass die App aktiv ist und lädt, wodurch eine bessere Benutzererfahrung gewährleistet wird. Der Ladebildschirm wurde so gestaltet, dass er genau drei Sekunden lang angezeigt wird. Dies gibt der App ausreichend Zeit, um die notwendigen Daten und Ressourcen im Hintergrund zu laden.
Es ist entscheidend, dass nach Ablauf der vier Sekunden die Benutzer zur Startseite (Homepage) der App navigiert werden und nicht zu anderen Seiten wie der Profil- oder Chatseite.

Gestaltung des Ladebildschirms

Für die Gestaltung des Ladebildschirms waren mehrere Schritte notwendig. Das Logo wurde mit Canva erstellt und musste den Charakter und die Zielgruppe der App widerspiegeln.
Es wurde darauf geachtet, dass das Design für eine SportTalentHub-App geeignet ist. Die Inspiration für das Logo wurde aus verschiedenen Quellen, wie der NFL, gezogen. Die Farben und das Design sollten sportlich und ansprechend sein.
Nach der Erstellung wurde das Logo transparent gemacht, um es optimal in den Ladebildschirm integrieren zu können.

Bevor das Logo endgültig in die App integriert wurde, wurde es im Rahmen eines wöchentlichen Meetings präsentiert. Das positive Feedback der Teammitglieder bestätigte die Eignung des Logos, sodass es anschließend in den Ladebildschirm eingefügt wurde.

Das Logo wurde im Ladebildschirm implementiert und ein animiertes Symbol hinzugefügt, das die Ladebewegung anzeigt.
Es wurde darauf geachtet, dass nach dem Ablauf der vier Sekunden der Nutzer zur Startseite navigiert wird. Dies wurde durch entsprechende Programmierung in Flutter sichergestellt.

Wichtige Funktionen des Ladebildschirms

Der Ladebildschirm wurde in Flutter programmiert und enthält folgende wichtige Funktionen:
- Die ‘initState‘-Methode startet einen Timer, der nach einer Dauer von drei Sekunden den Nutzer zur Startseite (Homescreen) weiterleitet:

```
@override
void initState() {
  super.initState();
  Timer(const Duration(seconds: 3), () {
    Navigator.of(context).pushReplacementNamed('/homescreen');
  });
}
```

- Der Ladebildschirm verwendet ein ‘Scaffold‘-Widget mit einem weißen Hintergrund und einem zentrierten ‘Column‘-Widget, das das Logo und einen ‘CircularProgressIndicator‘ anzeigt:

```
@override
Widget build(BuildContext context) {
  return const Scaffold(
    backgroundColor: Colors.white,
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Image(image: AssetImage('assets/images/FinalLogoSTHOriginal.png')),
          SizedBox(height: 20),
          CircularProgressIndicator(),
        ],
      ),
    ),
  );
}
```

Änderungen am App-Logo

Neben der Erstellung und Implementierung des Logos für den Ladebildschirm war es auch notwendig, das App-Logo selbst zu ändern. Das Ändern des App-Logos wurde in der ‘pubspec.yaml‘-Datei wie folgt konfiguriert:

```
flutter_launcher_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/Images/FinalLogoSTHOriginal.png"
  min_sdk_android: 21
```

Hierbei wurde das neue Logo in den entsprechenden Bereich des App-Projekts eingefügt. Wir entschieden uns außerdem, das Logo auch auf der Startseite der App anzuzeigen. Hierzu wurde ein Code in die Startseite geschrieben, der diese Funktion realisiert.

10 Homescreen

Der Homescreen ist der erste und damit wichtigste Anzeigebildschirm für die STH-App. Dieser wird direkt nach dem App Start angezeigt und ist somit sofort sichtbar.

Anforderungen an den Homescreen

Für die Funktionalitäten auf dem Homescreen wurden zunächst vergleichbare Apps wie Instagram verglichen und die wichtigsten Funktionen ausgearbeitet.
Folgende Anforderungen wurden dabei erstellt:
- Profile/Posts von Sportlern anzeigen
- Entry Point für die Chatfunktionalität

Erstellung der Flutter Widgets

Zunächst hat der Homescreen die zentralen Elemente CustomAppBar und CustomNavigationBar erhalten. Im nächsten Schritt wurde das sogenannte Post-Widget erstellt, das im Homescreen-Widget verwendet wird. Grund dafür ist, dass die unterschiedlichen Posts der Sportler vom Grund-Designkonzept gleich aufgebaut sein sollen. Die Posts sollen sich nur durch die inhaltlichen / persönlichen Informationen unterscheiden.
Aufgebaut ist ein Post mit einem Profilbild links oben, dem Profilnamen, den Bildern des Sportlerposts und einem Chat-Button, mit dem der Manager direkt zum jeweiligen Chat mit dem Sportler springt.

Daten aus Firebase für den Homescreen

Die Informationen (Profilname) und Dateien (Bilder und Profilbild) für die Sportlerposts auf dem Homescreen werden aus Firebase geladen, indem die UserID der jeweiligen Sportler übergeben wird. Aufgrund von Beispieldaten gibt es hier datenschutzrechtlich keine Einschränkungen, für einen möglichen Rollout der Funktionen in der App werden allerdings im Ausblick noch einige Verbesserungen / Möglichkeiten angesprochen.

11 Suchbildschirm

Der Suchbildschirm ist eine zentrale Funktion unserer STH-App (SportTalentHub). Er ermöglicht es den Nutzern, sowohl Sportlern als auch Sportmanagern, gezielt nach bestimmten Personen anhand von Hashtags oder Namen zu suchen.

Ziele des Suchbildschirms

Das Hauptziel des Suchbildschirms ist es, den Nutzern eine effiziente Möglichkeit zu bieten, Spieler oder Manager zu finden. Dies kann durch die Eingabe von Hashtags oder Namen erfolgen. Die Suchfunktion ist über das Suchsymbol auf dem Startbildschirm zugänglich.

Integration mit Firebase

Die Suchfunktion ist eng mit Firebase verbunden. Firebase spielt eine entscheidende Rolle, da dort die Profildaten der Nutzer gespeichert sind. Dadurch können Nutzer problemlos nach anderen Nutzern suchen. Wenn ein Suchbegriff, sei es ein Hashtag oder ein Name, eingegeben wird, erscheinen die gesuchten Personen auf dem Suchbildschirm. Bei Auswahl einer Person wird der Nutzer zu deren Profil weitergeleitet.

Wichtigkeit der Suchfunktion

Die Suchfunktion ist für unsere App von großer Bedeutung. Sie ermöglicht es Sportmanagern, gezielt nach bestimmten Merkmalen oder Fähigkeiten zu suchen und schnell und effektiv Ergebnisse zu erhalten. Dies verbessert die Benutzererfahrung und erhöht die Effizienz der App.

Wichtige Funktionen des Suchbildschirms

Der Suchbildschirm wurde in Flutter programmiert und enthält folgende wichtige Funktionen: - Ein ‘TextEditingController‘ wird verwendet, um die Eingabe des Suchbegriffs zu steuern und zu überwachen. - Eine Methode namens ‘fetchUserSearchResults‘ wird aufgerufen, wenn sich der Text im Suchfeld ändert. Diese Methode überprüft, ob der Suchbegriff nicht leer ist, und ruft dann die Suchergebnisse von Firebase ab. Die Suchergebnisse werden in einer Liste gespeichert und auf dem Bildschirm angezeigt. - Das Layout des Suchbildschirms besteht aus einem ‘Scaffold‘-Widget, das eine benutzerdefinierte App-Leiste (‘CustomAppBar‘) und eine Suchleiste enthält. Die Suchleiste ist ein

‘TextField‘-Widget mit einer Eingabeaufforderung und einem Suchsymbol. - Die Suchergebnisse werden in einer scrollbaren Liste (‘ListView.builder‘) angezeigt. Jede Ergebniszeile zeigt Informationen wie Name, E-Mail, Telefonnummer, Adresse und Hashtags des Nutzers an. Wenn ein Profilbild vorhanden ist, wird es ebenfalls angezeigt. - Ein ‘Card‘-Widget umgibt jede Ergebniszeile, um eine visuell ansprechende Darstellung zu gewährleisten. Beim Tippen auf eine Ergebniszeile wird der Nutzer zu weiteren Details des ausgewählten Profils weitergeleitet. - Am unteren Rand des Bildschirms befindet sich eine benutzerdefinierte Navigationsleiste (‘CustomBottomNavigationBar‘), die es den Nutzern ermöglicht, leicht zwischen verschiedenen Abschnitten der App zu navigieren.

Zukünftige Erweiterungen

Zukünftige Erweiterungen der Suchfunktion könnten die Speicherung früher gesuchter Personen im Suchbereich umfassen. Außerdem wäre es möglich, die Suche zu erweitern, sodass nicht nur nach Hashtags und Namen gesucht werden kann, sondern auch nach weiteren Informationen oder Merkmalen.

12 Profile-Screen & Accountprofile-Screen

In diesem Kapitel wird sowohl auf die Umsetzung der Flutter-Komponenten für den Profile-Screen als auch den Accountprofile-Screen des Projekts eingegangen. Dabei liegt der Fokus auf die konzeptionelle Gestaltung und Implementierung beider App-Seiten, die es beispielsweise Benutzern ermöglichen, ihr Profilbild sowie multimediale Inhalte wie Bilder und Videos zu verwalten, aber auch personenbezogene Daten sicher abzuspeichern. Diese Funktionalitäten werden dabei unter Einsatz verschiedener Flutter-Pakete und Kernfunktionen der Dart-Programmiersprache umgesetzt.
Beginnend mit dem Profile-Screen, wird eine StatefulWidget-Klasse verwendet, die Zustandsänderungen verfolgt und das Benutzerinterface entsprechend aktualisiert. Verschiedene Funktionen werden definiert, um auf Benutzerinteraktionen zu reagieren, beispielsweise das Öffnen von Galerieinhalten, das Laden von Bildern und Videos aus dem lokalen Speicher, das Hochladen von Medieninhalten auf Firebase sowie das Löschen von Bildern und Videos aus der Ansicht. Darüber hinaus werden Methoden implementiert, um die Anwendungszustände zu verwalten, Benutzereingaben zu verarbeiten und Multimedia-Inhalte wie Bilder und Videos sicher zwischen dem lokalen Speicher, der in der Regel durch "shared preferences" verwaltet wird, und dem Backend in Firebase zu übertragen. Dadurch werden die Benutzerdaten effizient synchronisiert und sicher in der Cloud gespeichert, wodurch Datenschutz und Datensicherheit gewährleistet werden. Bei Bedarf können diese Daten dann nahtlos abgerufen und aktualisiert werden. Dabei spielen bei der Implementierung dieser Features folgende Methoden eine entscheidende Rolle:
"build" ist die Methode, die das Benutzerinterface basierend auf dem aktuellen Zustand der StatefulWidget-Klasse aufbaut. Sie ist der Ausgangspunkt für den Aufbau der UI-Elemente und ermöglicht die dynamische Erstellung und Aktualisierung des Bildschirms. Die Methode "initState" spielt eine wichtige Rolle bei der Initialisierung des Zustands der StatefulWidget-Klasse. Hier werden unter anderem Informationen aus dem lokalen Speicher geladen und andere vorbereitende Maßnahmen getroffen, um das Benutzerinterface entsprechend anzupassen und eine konsistente Benutzererfahrung zu gewährleisten. Diese beiden Methoden bilden das Rückgrat des Profile-Screens und ermöglichen es, mithilfe von weiteren Methoden, die Benutzeroberfläche zu strukturieren und den Zustand der App zu verwalten. Zusätzliche Methoden wie "openGallery" und "openVideo" sind von entscheidender Bedeutung, um dem Benutzer die Möglichkeit zu geben, Bilder und Videos auszuwählen und anzuzeigen. Durch das Öffnen neuer Bildschirmfenster bieten sie eine intuitive Benutzeroberfläche und ermöglichen es dem Benutzer, Inhalte zu erkunden. Darüber hinaus bieten sie Funktionen zum Anzeigen, Schließen und Löschen von Inhalten, was die Interaktivität und Benutzerfreundlichkeit der STH-App verbessert. Methoden wie "loadAvatarImage" und "loadUserName" spielen eine essenzielle Rolle, um den Avatar des Benutzers und seinen Benutzernamen aus dem lokalen Speicher zu laden.

Diese Informationen werden wiederum aus dem Accountprofile-Screen bezogen, was eine Integration und eine personalisierte Anpassung des Benutzerinterfaces ermöglicht. Durch das Laden dieser Daten können Benutzer ihre Profile individuell gestalten und ein konsistentes Benutzererlebnis über verschiedene Bildschirme hinweg gewährleisten. Im Kontext der Datenschutzsicherheit spielen Methoden wie "saveImagePathsToLocalStorage", "saveImage", "loadImagesFromStorage" und "uploadGalleryImages" eine entscheidende Rolle beim Speichern, Laden und Hochladen von Bildern. Diese Methoden ermöglichen es Benutzern, ihre Bilder sowohl lokal im lokalen Speicher über die SharedPreferences als auch in der Cloud über das Backend in Firebase sicher zu verwalten und zu teilen. Durch die Verwendung dieser Methoden wird die Integrität der Benutzerdaten gewahrt und gleichzeitig eine zuverlässige Speicherung und Übertragung von Bildinhalten gewährleistet. Ähnlich wichtig sind Methoden wie "saveVideoPathsToLocalStorage", "saveVideo", "loadVideosFromStorage" und "uploadVideos" für die Verwaltung von Videos. Sie ermöglichen es dem Benutzer, multimediale Inhalte nahtlos zu verwalten und zu teilen. Dabei werden die Videodaten sowohl lokal im lokalen Speicher gespeichert als auch über das Backend in Firebase hochgeladen, wo sie für weitere Verwendungszwecke verfügbar sind. Diese Methoden spielen somit eine essenzielle Rolle bei der Sicherung und Bereitstellung von Videoinhalten für die Benutzer der App. Um dem Benutzer die Kontrolle über seine Inhalte zu geben, dienen "deleteImage" und "deleteVideo" dazu, ausgewählte Bilder und Videos zu löschen. "generateThumbnail" ermöglicht es, Vorschaubilder für Videos anzuzeigen, um dem Benutzer eine Vorschau des Inhalts zu bieten. Die Methoden "saveHashtagsToLocalStorage", "loadHashtagsFromStorage" und "deleteHashtagFromLocalStorage" tragen zur Verwaltung von Hashtags bei und verbessern die Personalisierung und Relevanz der bereitgestellten Inhalte. Die Methode "showHashtagsModal" erlaubt es dem Benutzer, Hashtags auszuwählen und hinzuzufügen, um seine Interessen zu kennzeichnen. Sie fördert die Benutzerbeteiligung und ermöglicht die Personalisierung von Inhalten. Die Benutzeroberfläche des Profilbildschirms setzt sich aus einem Header-Bereich mit dem Benutzernamen und dem Profilbild sowie einem Abschnitt zur Anzeige und Verwaltung von Bild- und Videoinhalten zusammen. Die Anzeige kann zwischen Bildern und Videos umgeschaltet werden, und Benutzer haben die Möglichkeit, neue Medieninhalte auszuwählen und hochzuladen, welche von den Methoden pickMultiImage und pickVideo unterstützt werden. Für die Gestaltung der Benutzeroberfläche kommen verschiedene Flutter-Widgets wie Column, Stack, AppBar, CircleAvatar und GridView zum Einsatz, um eine ansprechende und benutzerfreundliche Erfahrung zu gewährleisten. Zusätzlich werden benutzerdefinierte Widgets wie CustomAppBar und CustomBottomNavigationBar verwendet, um die Navigation und Interaktion zu erleichtern und das Design konsistent zu halten.

Im Accountprofile-Screen hingegen erscheinen bearbeitbare personenbezogene Daten, wobei zur Validierung der Eingaben reguläre Ausdrücke (Regex) herangezogen werden. Den Nutzern wird die Gelegenheit geboten, ihre Profildaten anzupassen und zu sichern – darunter fallen der Name, die Telefonnummer, die Adresse und die E-Mail-Adresse. Beim Start des Bildschirms werden die aktuellen Profildaten aus den SharedPreferences geladen und in entsprechenden Textfeldern angezeigt. Nutzer können diese Informationen bearbeiten und die Änderungen anschließend speichern. Hier haben Nutzer die Freiheit, ein Profilbild zu wählen, das dann in einem kreisförmigen Avatar angezeigt wird. Das ausgesuchte Bild wird nicht nur lokal auf dem Gerät gespeichert, sondern auch im Backend

abgelegt. Bei Bedarf wird es von Firebase abgerufen, um anderen Seiten mit den entsprechenden Daten zu bereichern. Die Benutzeroberfläche des Accountprofilbildschirms besteht aus mehreren Textfeldern für die Bearbeitung von Profildaten sowie einem Abschnitt für das Profilbild. Im Bearbeitungsmodus werden die Textfelder kenntlich und sichtbar zur Bearbeitung hervorgehoben, sodass Nutzer ihre Daten ändern können. Fehlermeldungen erscheinen, falls ungültige Eingaben gemacht wurden, und ein Tooltip mit einem Hover-Effekt informiert die Nutzer über die Art des Fehlers. Auch hierbei wurden folgende Methode zur Implementierung benötigt, um unseren konzeptionellen Anforderungen gerecht zu werden:
Die Build-Methode erstellt die Benutzeroberfläche des Accountprofil-Screens. Sie umfasst Textfelder für die Bearbeitung von Profildaten sowie einen Bereich für das Profilbild. Die UI wird je nach Bearbeitungsstatus der Daten dynamisch angepasst. Die PickImage-Methode ermöglicht es Benutzern, ein Profilbild auszuwählen und es hochzuladen. Sie wird aktiviert, sobald der Benutzer auf die entsprechende Schaltfläche klickt. Dadurch wird die Bildauswahl gestartet, und das ausgewählte Bild wird für das Profil verwendet. Beim Start des Bildschirms ruft die LoadProfileData-Methode die aktuellen Profildaten aus den SharedPreferences ab. Diese Daten, einschließlich Name, Telefonnummer, Adresse und E-Mail, werden dann in den entsprechenden Textfeldern angezeigt, damit der Benutzer sie bearbeiten kann. Um das ausgewählte Profilbild lokal zu speichern und sicherzustellen, dass es für spätere Verwendungszwecke verfügbar ist, wird die Methode saveAvatarImage verwendet. Nachdem der Benutzer ein Bild ausgewählt hat, wird es lokal auf dem Gerät gespeichert. Zusätzlich dazu werden die aktualisierten Profildaten, einschließlich des Profilbildes, im Backend gespeichert, damit das Profilbild auch von anderen Seiten oder Anwendungen wiederverwendet werden kann. Die LoadAvatarImage-Methode wird aufgerufen, um das Profilbild aus der lokalen Speicherung zu laden. Dadurch wird das zuvor gespeicherte Profilbild beim Öffnen des Accountprofil-Screens geladen und im Kreisavatar angezeigt. Für die Validierung der personenbezogenen Daten werden unter Verwendung von regulären Ausdrücken (Regex) folgende Methoden verwendet. Diese dienen dazu, sicherzustellen, dass die eingegebenen Daten gültig sind, bevor sie gespeichert werden. Dabei werden sämtliche personenbezogenen Daten sowohl lokal auf dem Gerät als auch im Backend gespeichert, um die Integrität der Daten zu gewährleisten und sicherzustellen, dass sie für verschiedene Anwendungen verfügbar sind:

- Die ValidatePhone-Methode validiert die eingegebene Telefonnummer und akzeptiert nur Zahlen und optional ein Pluszeichen am Anfang
- Um die Gültigkeit der eingegebenen Adresse zu überprüfen, wird die ValidateAddress-Methode verwendet. Diese erwartet Buchstaben, Zahlen, Kommas, Punkte, Leerzeichen und Umlaute als gültige Zeichen
- Die ValidateEmail-Methode prüft, ob die eingegebene E-Mail-Adresse gültig ist und erwartet das übliche Format einer E-Mail-Adresse

Während des abschließenden Bearbeitungsvorgangs wird die Methode updateUserData aufgerufen. Diese Funktion dient dazu, die Profildaten in der Cloud Firestore-Datenbank zu aktualisieren. Dabei werden sämtliche neuen Informationen wie Name, Telefonnummer, Adresse und E-Mail in der Datenbank aktualisiert, um sicherzustellen, dass sie synchronisiert und auf dem neuesten Stand sind. Dies ermöglicht es, die aktualisierten Daten

für andere Zwecke zu beziehen und weiterzuverwenden. Sobald der Benutzer die Profildaten bearbeitet hat, werden diese mithilfe der SaveProfile-Methode in den SharedPreferences gespeichert, um sie beim nächsten Öffnen des Bildschirms wiederherzustellen. Die Interaktion zwischen dem Profilbildschirm und dem Accountprofilbildschirm sowie der Datenaustausch zwischen ihnen bestimmen die grundlegende Ausrichtung der appspezifischen Funktionen. Diese Funktionen erstrecken sich auf verschiedene Bereiche der Anwendung, wie beispielsweise den Home-Screen über das Backend, die dem Benutzer die Wiedergabe von Informationen in Form von Bildern, Videos und personenbezogenen Daten ermöglicht. Auch der Suchbildschirm profitiert von der Implementierung der Daten über das Backend, da dieser den Suchalgorithmus unterstützt und dem Benutzer dabei hilft, relevante Ergebnisse zu finden, die im Sinne einer Social-Media-App agieren.

13 Implementierung des Chat-Screens

Zu jeder modernen Community Applikation gehört auch die Chatfunktionalität immer dazu. Der Austausch zwischen zwei oder mehreren Menschen ist besonders in der STH App wichtig, um die Kommunikation zwischen Spielern und Managern zu ermöglichen. Die Anforderungen an den Chat-Screen standen schnell fest. Die Funktionalität muss mindestens den gleichen Interaktionsumfang ermöglichen, wie vergleichbare Apps wie z.B. Instagram, Snapchat oder X (ehemalig Twitter).

13.1 Anforderungen an den Chat-Screen

Die spezifischen Anforderungen für die Chat-Funktion in der STH App wurden im Projektteam gemeinsam ausgearbeitet. Dabei wurden zunächst vergleichbare Applikationen genauer beleuchtet und die Mindestanforderungen anhand der dort gesichteten Funktionen definiert. Das Chatten in der STH App muss mittels verschlüsselte eins zu eins Verbindungen funktionieren, in der zwei Menschen miteinander in einem Chat-Raum kommunizieren können. Zudem muss es auch die Möglichkeit geben Gruppenchats zu erstellen und mehrere Menschen dazu einzuladen. Auch hierbei müssen die Verbindungen stets verschlüsselt sein. Innerhalb eines Chats muss jeder Nutzer Nachrichten senden und empfangen können. Zu den Nachrichten gehören Textnachrichten, GIFs, Bilder und Videos. Jeder Nutzer muss gesendete und empfangene Nachrichten im Chat sehen können und die Chats müssen mit ihrem kompletten Verlauf beim ein- und ausloggen erhalten und neu geladen werden können.

13.2 Erstellung einer Flutter Widget-Seite

Nachdem die Anforderungen definiert wurden, kann nun die Erstellung der Chat-Screen Widget-Seite beginnen. Diese Seiten dienen als Vorbereitung dazu, Widgets platzieren und anzeigen zu können. Das Erstellen einer derartigen Seite kann in Flutter durch das Erstellen einer Klasse realisiert werden. Hierbei wird für die Klasse ein kontextbezogener Name eingesetzt und die Art der Widget-Seite definiert. Dabei wird zwischen Stateful und Stateless Widgets unterschieden. Bei den Stateful Widget Klassen können sich deren Komponenten verändern. Die Stateless Widgets verändern sich nicht und können sich nicht in der Ansicht aktualisieren. Für den Chat-Screen ist das Stateful Klassenwidget die richtige Wahl, da sich die Anzahl und Inhalte der angezeigten Chats stetig verändern kann.

13.3 Integration des Stream-Chat-Flutter Pakets

Die Implementierung einer Chat-Funktion kommt in Social-Media Apps sehr häufig bzwfast immer vor. Deshalb gibt es hierzu eine große Anzahl an Flutter Paketen, welche sich unter der Seite pub.dev genauer angeschaut werden können. Ein sehr mächtiges und umfängliches Paket nennt sich hierbei Stream-Chat-Flutter. Dieses Paket bringt eine Anzahl an vorbereiteten Widgets für die Anzeige und Erstellung von Chats mit sich. Darin beinhaltet sind App-Bars, Channel-Screens, Chat-Screens und Chatfunktionen. Die Einbindung dieses Pakets erfordert das Verändern der gesamten Struktur der Anwendung. Die Chat-Funktion stellt einen Hauptbestandteil der Anwendung dar und muss auf den Top of Widget-Tree implementiert werden. Das bedeutet, dass das Paket Stream-Chat-Flutter als Erstes in der Anwendung geladen werden muss, um es korrekt auszuführen. Nachdem die Struktur und das Laden der Pakete in der Anwendung angepasst wird, müssen Funktionen implementiert werden, die das Abrufen von Chats aus dem Stream-Chat-Client ermöglichen. Dabei greift die STH App auf das separate Backend des Paketes zu und authentifiziert den aktuellen Nutzer der Applikation. Nach erfolgreicher Authentifizierung werden die zugehörigen Chatdaten und Channels abgerufen, um dem Nutzer alle Chatverläufe korrekt anzeigen zu können.

13.4 Benutzerauthentifizierung

Um die Benutzerauthentifizierung durch die Anwendung zu ermöglichen, müssen Zugangsschlüssel bzw. Tokens bereitstehen. Diese Tokens werden durch den Aufruf durch Firebase generiert und der jeweilige Nutzer der Anwendung registriert. Nach erfolgreicher Übermittlung der Tokens kann die STH App nun sicherstellen, dass es sich um den angemeldeten Nutzer handelt und alle Chatverläufe inclpersönlicher Daten laden.

13.5 Implementierung der Chat-Widgets

Nach der erfolgreichen Benutzerauthentifizierung folgt der letzte Schritt. Die verschiedenen Chats des Nutzers und die Chatverläufe können mittels der Chat-Widgets aus dem Paket Stream-Chat-Flutter angezeigt werden. Das Paket beinhaltet einen vollständigen Channel-Screens. Das ist das Widget, welches zuerst beim Einstieg in die Chat-Funktionen angezeigt wird. Analog zu anderen bekannten Applikationen wird hierbei eine Übersicht über alle aktiven Chats des Nutzers angezeigt. Daraufhin folgt der Channel-Screens. Dieser stellt den einzelnen Chat, auf dem vom Channel-Screens aus geklickt wird, dar. Im Chat-Screen ist der gesamte Chatverlauf aus diesem Chat des Nutzers zu sehen. Zudem kann hier durch das Textfeld am unteren Ende der Seite neue Nachrichten durch den Nutzer eingegeben werden und an den Chatpartner versendet werden.
Die Implementierung des Channel-Screens erfolgt als Erstes. Hierfür wird eine eigene Klasse erstellt und das Chat-Screen Widget aus dem Paket wird auf dieser Seite platziert. Nun muss noch das richtige Routing definiert werden. Das stellt sicher, dass die verschiedenen Chats geladen werden können. Die Liste der verfügbaren Chats wird geladen und am oberen Ende der Seite wird die App-Bar mit dem Namen Chats angezeigt.

Nach der Integration des Channel-Screen-Widgets kann nun das Chat-Screen-Widget folgen. Für das Chat-Screen-Widget wird ebenso eine neue Klasse und Seite erstellt, worauf das Widget platziert werden kann. Nach der Platzierung wird durch das korrekte Routing sichergestellt, dass der richtige Chat vom angemeldeten User geöffnet wird. Durch die nahtlose Integration des Paketes sind alle Funktionen automatisch verfügbar. Das Versenden und Empfangen von Nachrichten ist bereits möglich und das Reagieren auf Nachrichten ebenso. Zu normalen Textnachrichten können auch Bilder, GIFs und Emojis versendet werden. Die Grundfunktionalitäten einer ausgereiften Social-Media App stehen nun mit der Chat-Funktion.
Nun ist der letzte Schritt die Anwendung auf die korrekte Wegweisung bzw. Routing anzulernen. Dabei wird in der main.dart, welches die Hauptdatei darstellt und dabei auch die Strukturen für die gesamte Anwendung festlegt, eingearbeitet. Die neuen Screens müssen den gesamten Routing Optionen der Anwendung hinzugefügt werden. Die dabei verwendeten Klassen werden dabei referenziert. Innerhalb der Datei CustomAppBar wird nun für den Chat-Button das korrekte Routing hinterlegt. Dabei wird auch festgelegt in welchen Bildschirmen der Chat-Button sichtbar ist und in welchen nicht. Innerhalb des Chat-Screens verschwindet der Chat-Button, da der Anwender sich bereits in der Chat-Funktionalität aufhält. In allen anderen Ansichten ist der Button sichtbar und das Routing erfolgt beim auslösen. Nun ist der Channel-Screen und der Chat-Screen in der Anwendung vollständig mit korrektem Routing implementiert.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die angemeldete Prüfungsleistung in allen Teilen eigenständig ohne Hilfe von Dritten anfertigen und keine anderen als die in der Prüfungsleistung angegebenen Quellen und zugelassenen Hilfsmittel verwenden werde. Sämtliche wörtlichen und sinngemäßen Übernahmen inklusive KI-generierter Inhalte werde ich kenntlich machen. Diese Prüfungsleistung hat zum Zeitpunkt der Abgabe weder in gleicher noch in ähnlicher Form, auch nicht auszugsweise, bereits einer Prüfungsbehörde zur Prüfung vorgelegen; hiervon ausgenommen sind Prüfungsleistungen, für die in der Modulbeschreibung ausdrücklich andere Regelungen festgelegt sind. Mir ist bekannt, dass die Zuwiderhandlung gegen den Inhalt dieser Erklärung einen Täuschungsversuch darstellt, der das Nichtbestehen der Prüfung zur Folge hat und darüber hinaus strafrechtlich gem. § 156 StGB verfolgt werden kann. Darüber hinaus ist mir bekannt, dass ich bei schwerwiegender Täuschung exmatrikuliert und mit einer Geldbuße bis zu 50.000 EUR nach der für mich gültigen Rahmenprüfungsordnung belegt werden kann. Ich erkläre mich damit einverstanden, dass diese Prüfungsleistung zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

München, den 9. Juni 2024

Fitim Makolli