LinkUp

Social-Media-App

Ein Bild, das Multimedia, Software, Screenshot, Computer enthält.

Automatisch generierte Beschreibung

|  |  |
| --- | --- |
| Autor | Noris, Simon, Sasa, Navin und Nicolas |
| Erstelldatum | **22.06.2023** |
| Abgabedatum |  |

Inhaltsverzeichnis

[Änderungstabelle 4](#_Toc146873931)

[1. Anforderungen 5](#_Toc146873932)

[1.1 Idee 5](#_Toc146873933)

[1.2 User Stories 5](#_Toc146873934)

[2. Technologien 7](#_Toc146873935)

[2.1 Technologien 7](#_Toc146873936)

[3. SCRUM Umsetzung 8](#_Toc146873937)

[3.1 Team 8](#_Toc146873938)

[3.2 Tool 8](#_Toc146873939)

[3.3 9](#_Toc146873940)

[4. Code Planung 9](#_Toc146873941)

[4.1 Wireframe/UX, UI-Design 9](#_Toc146873942)

[4.2 Klassendiagramm 10](#_Toc146873943)

[4.3 Name & Logo 11](#_Toc146873944)

[4. Code Umsetzung 12](#_Toc146873945)

[4.1 Backend 12](#_Toc146873946)

[4.2 Datenbank 13](#_Toc146873947)

[4.3 Frontend 14](#_Toc146873948)

[5. Kontrollieren 16](#_Toc146873949)

[5.1 Testprotokoll Backend 16](#_Toc146873950)

[5.2 Testprotokoll Frontend 18](#_Toc146873951)

[6. Factsheet 21](#_Toc146873952)

[6.1 Factsheet 21](#_Toc146873953)

[7. Inbetriebnahme 22](#_Toc146873954)

[7.1 Voraussetzung: 22](#_Toc146873955)

[7.2 Installation 22](#_Toc146873956)

[8. Auswerten 23](#_Toc146873957)

[8.1 Reflexion 23](#_Toc146873958)

[9. Quellen 24](#_Toc146873959)

[9.1 Quellen 24](#_Toc146873960)

[9.2 Abbildungsverzeichnis 25](#_Toc146873961)

# Änderungstabelle

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Wer | Was | Checked |
| 22.09.2023 | Simon Streuli | Initial Dokument erstellt |  |
| 29.09.2023 | Simon Streuli |  |  |
| 15.10.2023 | Noris Girolimetto |  |  |
|  |  |  |  |

# 1. Anforderungen

### 1.1 Idee

Das Konzept besteht darin, eine soziale Medienplattform zu entwickeln. Die Anwendung soll es Benutzern ermöglichen, sich zu registrieren, Inhalte zu teilen, Inhalte zu kommentieren, anderen Benutzern zu folgen. Die App soll eine benutzerfreundliche Oberfläche besitzen und es Benutzern ermöglichen, sich zu vernetzten. Sie soll sowie auf Desktopgeräten und auch Mobilgeräten verfügbar sein.

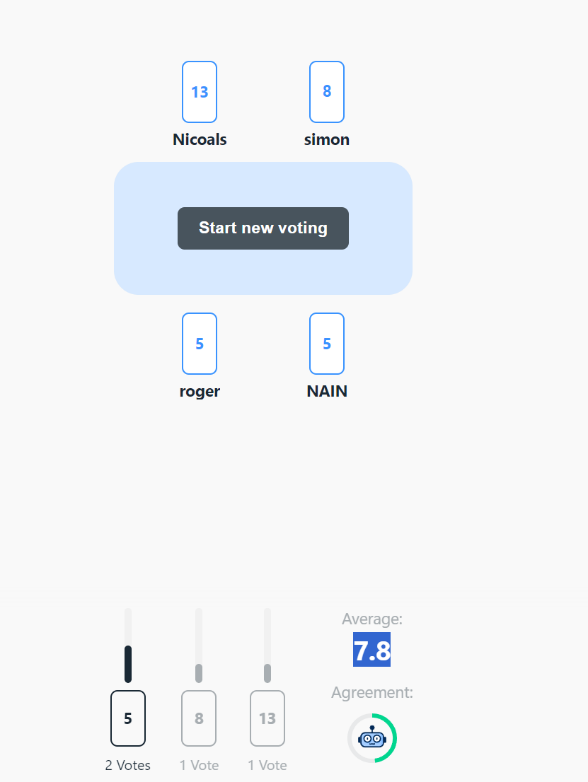
### 1.2 User Stories

|  |  |  |
| --- | --- | --- |
| User Story | Prior.1 | Poker AVG. |
| * Als Benutzer möchte ich die Möglichkeit haben, mich über mein vorhandenes Social-Media-Konto anzumelden, um Zeit zu sparen. | M | 5.6 |
| * Als neuer Benutzer möchte ich mich schnell und einfach mit meiner E-Mail-Adresse und einem sicheren Passwort registrieren können, um die App nutzen zu können. | M | 7.8 |
| * Als Benutzer möchte ich mein Profilbild und meine persönlichen Informationen bearbeiten können, um meine Identität darzustellen. | S | 7.9 |
| * Als Benutzer möchte ich meine Datenschutzeinstellungen anpassen können, um die Sichtbarkeit meiner Informationen zu steuern. | S | 13 |
| * Als Benutzer möchte ich meinen personalisierten Newsfeed durchsuchen können, um aktuelle Beiträge und Aktivitäten in meiner Community zu sehen. | M | 11.6 |
| * Als Benutzer möchte ich Beiträge erstellen, bearbeiten und löschen können, um meine Gedanken und Erfahrungen zu teilen. | M | 9.4 |
| * Als Benutzer möchte Empfehlungen von Beiträgen erhalten, um neue Inhalte und Benutzer zu entdecken. | S | 11.2 |
| * Als Benutzer möchte ich Beiträge liken und kommentieren können, um meine Unterstützung oder Meinung auszudrücken | S | 6.8 |
| * Als Benutzer möchte ich Benachrichtigungen über Aktivitäten in meinen Beiträgen und Kommentaren erhalten, um auf dem Laufenden zu bleiben. | C | 13.1 |
| * Als Benutzer möchte ich Premium-Funktionen abonnieren können, um ein werbefreies Erlebnis zu erhalten. | C | 18.7 |

|  |  |  |
| --- | --- | --- |
| 1) M = Must have | S = Should Have C = Could Have | W = Won’t have |
|  |  |  |

Die Schwierigkeit haben wir mit **Planning Poker** eingeschätzt.

Planning Poker ist eine Technik in Scrum, um Aufwandsschätzungen für Aufgaben oder User Stories zu machen. Teammitglieder verwenden Kartendecks mit Zahlen, um den Aufwand auszudrücken. Der Product Owner stellt die Aufgabe vor, und das Team wählt gleichzeitig eine Karte aus, um ihre Schätzung anzuzeigen. Dies fördert Teamzusammenarbeit und führt zu realistischen Schätzungen.

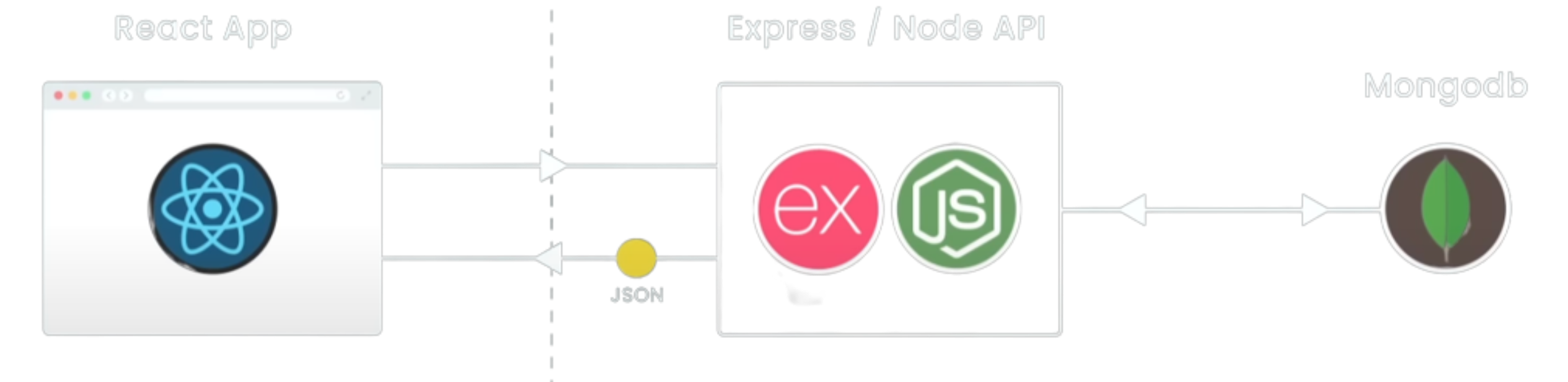


# 2. Technologien

### 2.1 Technologien

**MERN-Stack** ist eine beliebte Technologiekombination für die Entwicklung von Webanwendungen, die aus MongoDB, Express.js, React und Node.js besteht. Wir haben uns entschieden, das Projekt mit dem MERN-Stack zu realisieren, da er eine Reihe von Vorteilen bietet.

Zunächst ermöglicht der MERN-Stack eine effiziente und nahtlose Entwicklung. Mit MongoDB als NoSQL-Datenbank bietet er Flexibilität bei der Speicherung und Abfrage von Daten. Express.js dient als leichtgewichtiger Webframework für die Serverseite und ermöglicht eine einfache Handhabung von HTTP-Anfragen. React, eine leistungsstarke JavaScript-Bibliothek, ermöglicht eine schnelle und reaktive Benutzeroberfläche auf der Clientseite. Node.js, eine JavaScript-Laufzeitumgebung, ermöglicht die Ausführung des Servers und die Kommunikation zwischen Frontend und Backend.



9 MERN Stack

# 3. SCRUM Umsetzung

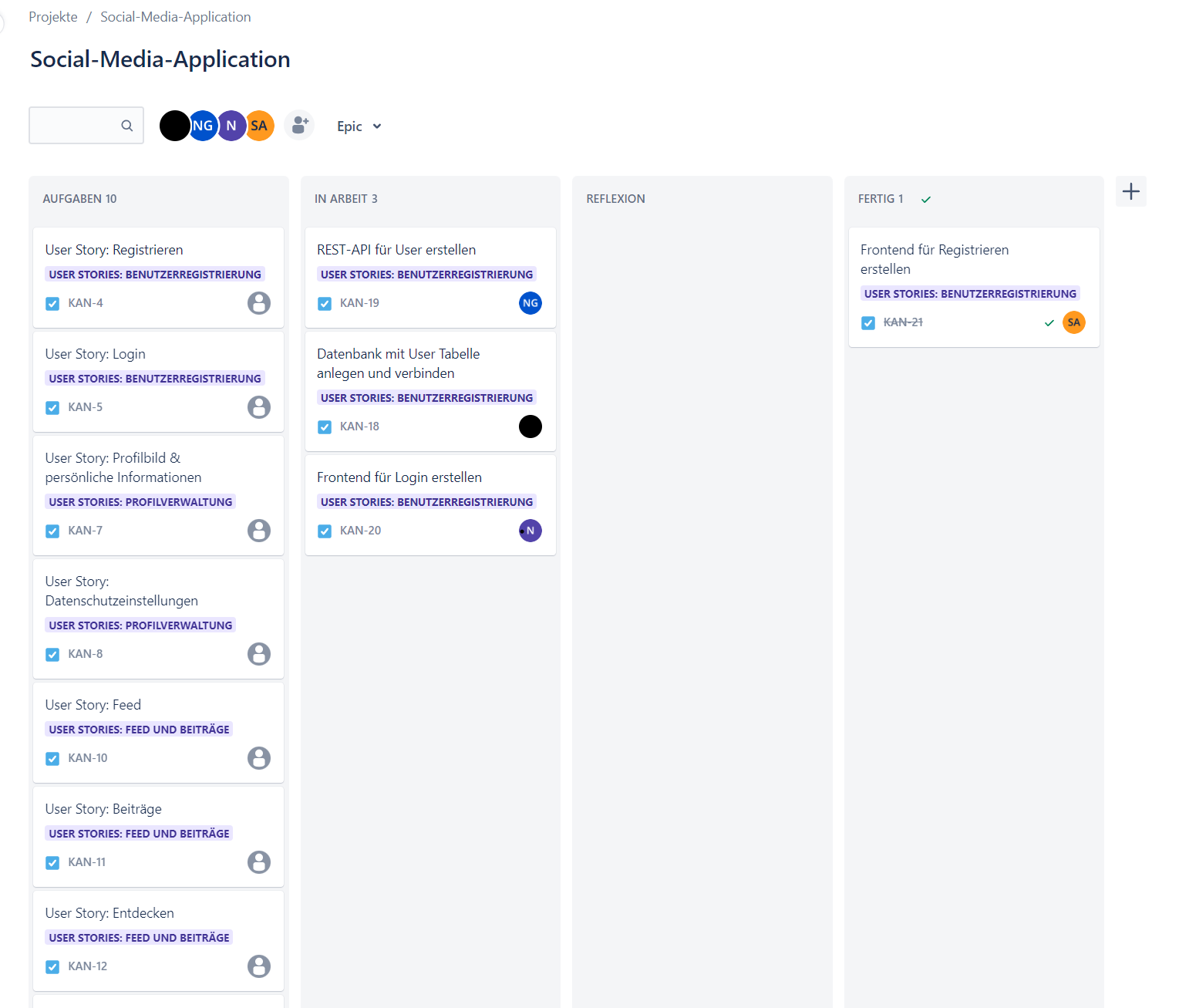
### 3.1 Team

|  |  |  |
| --- | --- | --- |
| Product Owner | Scrum Master | DEV-Team |
| Nicolas Spiess | Sasa Acimovic | Simon Streuli |
|  |  | Noris Girolimetto |
|  |  | Navin Shrestha |

### 3.2 Tool

Wir haben uns für das Scrum-Tool **Jira** entschieden, da es ein tiefgründiges Scrum-Tool ist und einige von uns in ihrem Betrieb ebenfalls Jira verwenden. Es bietet erweiterte Anpassungsmöglichkeiten und Funktionalitäten, die unseren Bedürfnissen am besten entsprechen.

Kurzer Einblick in unser Jira Board:



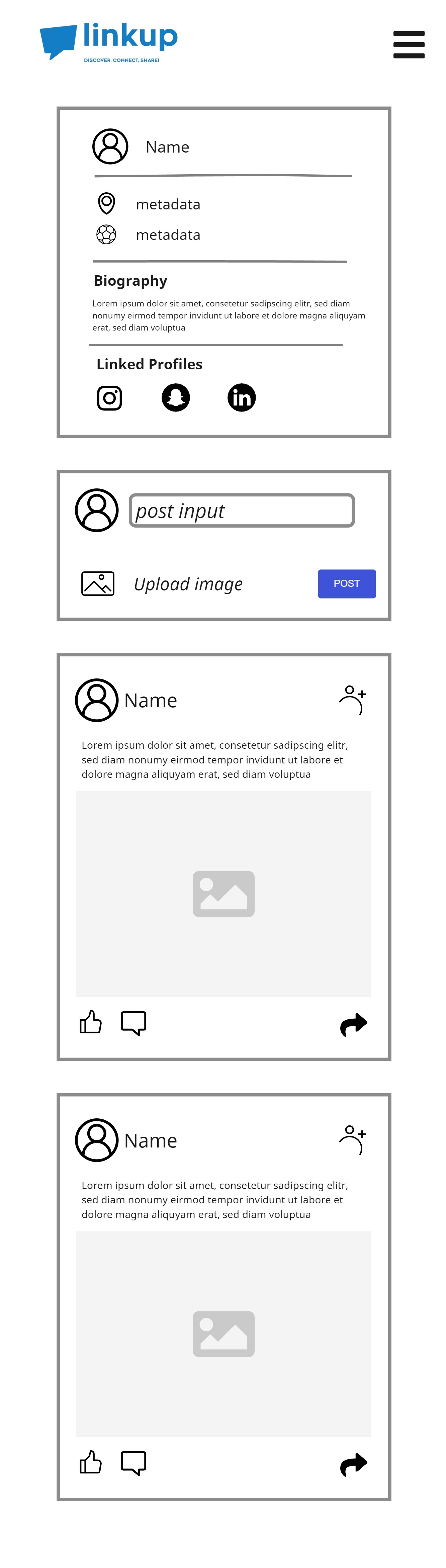
Jira Board

### 3.3

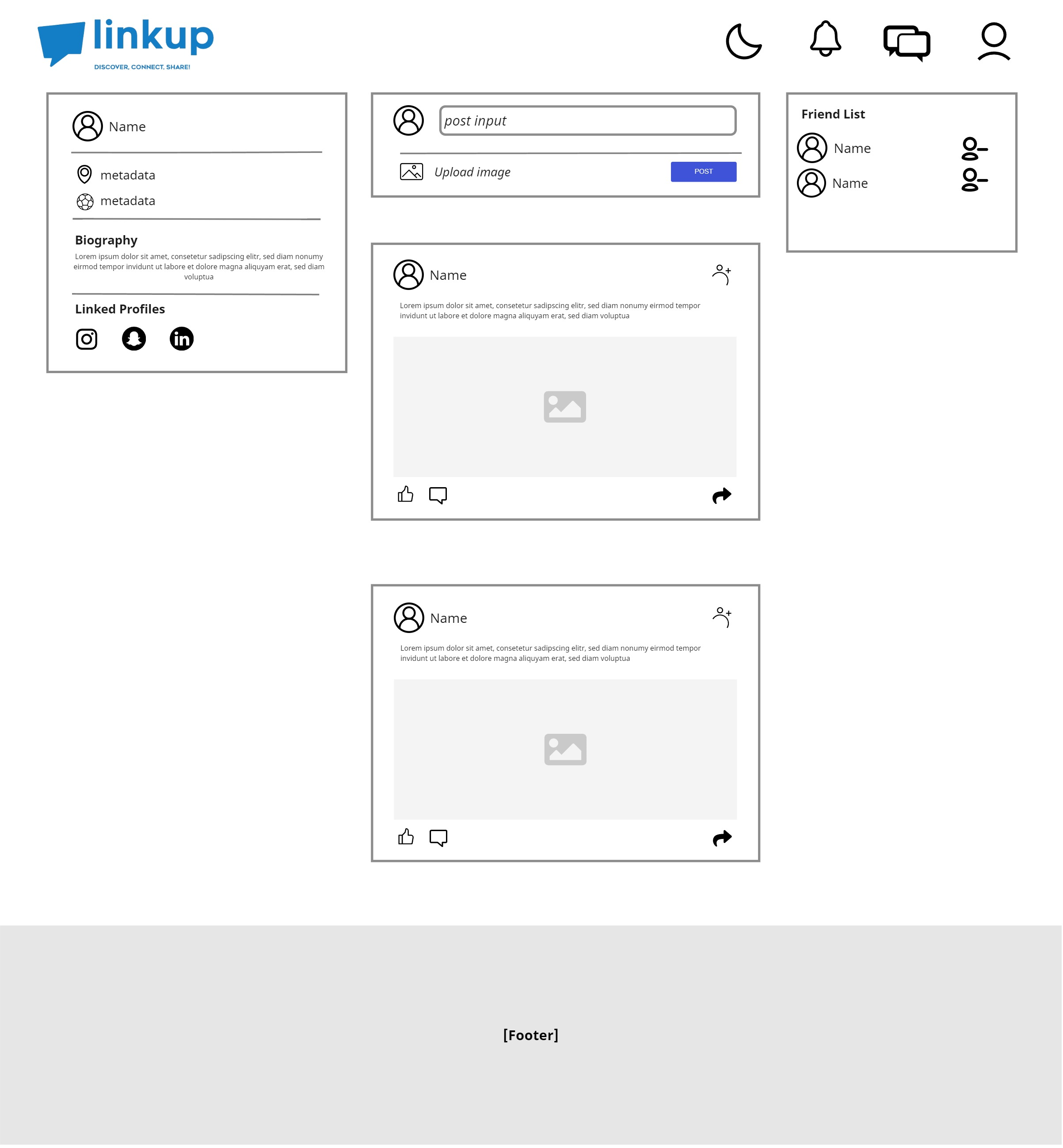
# 4. Code Planung

### 4.1 Wireframe/UX, UI-Design

Wireframes sind eine wichtige Komponente bei der Entwicklung einer Anwendung und spielen eine entscheidende Rolle in der Phase des UI/UX-Designs. Die Erstellung von Wireframes hat uns dabei geholfen, das grundlegende Layout, die Struktur und die Funktionalität unserer Social Media App zu visualisieren und zu planen.



3 Wireframe Startseite Handy



2 Wirefame Startseite Desktop

3 Wireframe Handy Desktop

Mobil-Verison Wireframe

### 4.2 Klassendiagramm

Ein Bild, das Text, Diagramm, Zahl, parallel enthält.

Automatisch generierte BeschreibungDas Klassendiagramm dient als visuelle Darstellung der Struktur und der Beziehungen zwischen den Klassen in deiner Social Media App. Es bietet einen Überblick über die wichtigsten Klassen und deren Attribute und Methoden.

5 Klassendiagramm

### 4.3 Name & Logo

Für den Namen der Social Media App habe ich zur Inspiration ChatGPT gefragt und habe mich für den Namen "**LinkUp**" entschieden.

Ich habe mich für den Namen "LinkUp" entschieden, da er verschiedene Aspekte meiner Social Media App widerspiegelt. Der Name setzt sich aus den beiden Wörtern "Link" und "Up" zusammen.

Ein Bild, das Text, Schrift, Logo, Grafiken enthält.

Automatisch generierte BeschreibungDas Logo habe ich mit Looka erstellt. Ich habe mich für ein schlichtes, blaues Design mit dem Slogan „Discover, Connect, Share!“ entschieden. Das Logo sieht so aus:

6 Logo

Ein Bild, das Grafiken, Design enthält.

Automatisch generierte BeschreibungAusserdem habe ich auch ein Icon für die Webseite erstellt. Das Icon sieht so aus:

7 Transparent Logo

8 Icon

Ein Bild, das Schrift, Logo, Grafiken, Text enthält.

Automatisch generierte Beschreibung

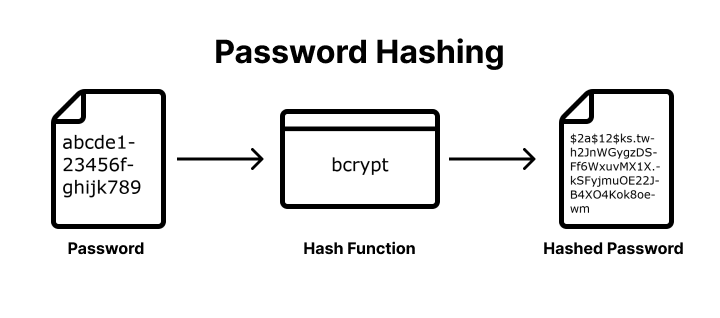
# 4. Code Umsetzung

### 4.1 Backend

Um das Backend zu realisieren, haben wir Express.js verwendet, das ein beliebtes Node.js-Framework ist. Express.js ermöglicht das Erstellen von APIs und das Handhaben von HTTP-Anfragen. Mit Express.js können Routen definiert werden, um verschiedene Endpunkte anzusprechen und die entsprechenden Aktionen auszuführen.

Ich habe Routes für User, Posts und die Authentifizierung erstellt. Bei den User und Posts Routes haben wir **CRUD-Methoden** implementiert, um User zu erstellen, User zu updaten und User zu löschen.

In unserem Projekt haben wir bcrypt verwendet, um die Sicherheit der Passwörter zu gewährleisten. bcrypt ist eine bewährte Methode zur Hashung und Überprüfung von Passwörtern. Durch die Verwendung von bcrypt werden die Passwörter meiner Benutzer sicher gehasht, sodass sie nicht im Klartext gespeichert werden.

Ein weiterer Vorteil von bcrypt ist die Verwendung von Salting. Beim Hashen wird automatisch ein zufälliger Salz-Wert generiert und mit dem Passwort kombiniert. Dadurch wird die Sicherheit weiter erhöht, da es schwieriger wird, die gehashten Passwörter durch vorberechnete Hash-Tabellen oder Brute-Force-Angriffe zu entschlüsseln.

11 Bcrypt

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungFür die Registrierung haben wir einen POST-Endpunkt erstellt. Ich das Passwort entgegen und hashe es mit bcrypt und dem Salt Wert. Danach wird ein neuer User mit den Informationen vom Body erstellt und der Benutzer wird in der Datenbank erstellt. Das Ganze habe ich in einen try und catch Block geschrieben, sodass ich Fehler abfangen kann. Im Fehlerfall wird eine Fehlermeldung mit Statuscode 500 zurückgegeben.

### 4.2 Datenbank

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungFür die Datenbank habe ich MongoDB verwendet. Ich bin auf die offizielle MongoDB Seite gegangen und habe ein Projekt erstellt, mit dem gratis Speicher. Danach habe ich geschaut, wie ich mich mit meiner Datenbank connecte, das sieht so aus:  
Die Datenbank URL und der Port habe ich in das. env File geschrieben. Da meine Datenbank nicht öffentlich sein soll und man den Port beliebig anpassen können soll.

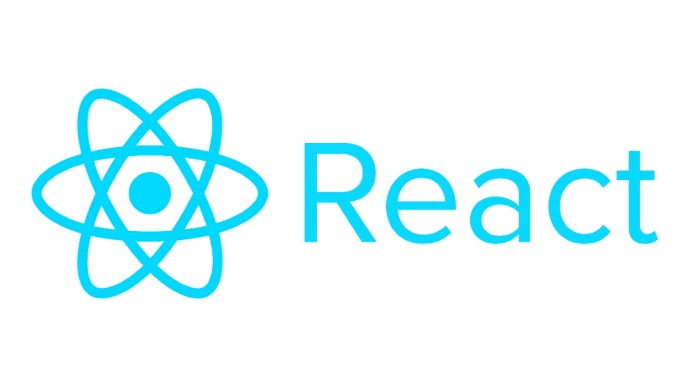
12 Datenbank Connection

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungAusserdem habe ich die Datenstruktur für User und Posts erstellt, mit Models. Das sieht in etwa so aus:

13 User Model

### 4.3 Frontend

Für die Entwicklung des Frontends in meinem Projekt habe ich moderne Webtechnologien verwendet. Ich habe mich für das React-Framework entschieden, da es eine leistungsstarke und flexible Lösung zur Erstellung von Benutzeroberflächen bietet.

14 React

Mit React konnte ich mein Frontend Komponenten aufteilen, um eine saubere und wiederverwendbare Codebasis zu schaffen. Ich habe JSX (JavaScript XML) verwendet, um die Komponenten zu erstellen und die Logik und das Design zu integrieren. Ich habe folgende Komponenten erstellt:

* Explorefeed
* Feed
* Navbar
* Post
* Share

Diese Komponenten habe ich in verschiedenen Pages verwendet. Ich habe folgende Pages:

* Home
* Explore
* Login
* Register
* Profile

Zur Verwaltung des Zustands meiner Anwendung habe ich den Context API von React genutzt. Dadurch konnte ich Daten zentral speichern und über verschiedene Komponenten hinweg darauf zugreifen. Die Verwendung des Context API hat mir geholfen, den Zustand meiner Anwendung effizient zu verwalten und die Kommunikation zwischen den Komponenten zu erleichtern.

Um mit dem Backend zu kommunizieren und Daten von den APIs abzurufen, habe ich das Axios-Paket verwendet. Mit Axios konnte ich HTTP-Anfragen an meine Backend-Server senden und die entsprechenden Daten verarbeiten.  
  
Ein Bild, das Schrift, Grafiken, Logo, Text enthält.

Automatisch generierte Beschreibung

15 Axios

# 5. Testing

### 5.1 Testprotokoll Backend

|  |  |
| --- | --- |
| **Testfall Nr.** | **1** |
| Beschreibung | Im Backend mittels POST Request User registrieren. |
| Voraussetzungen | Backend gestartet und am Laufen. |
| Testschritte | * Öffne Postman und erstelle einen POST Request mit der URL <http://localhost:8800/api/auth/register> * Füge im Request-Body die folgenden Daten hinzu: * username: [Test-Benutzername] * email: [Test-E-Mail] * password: [Test-Passwort] * Request absenden |
| Erwartetes Ergebnis | Der Server gibt den Statuscode 200 zurück und sendet die Daten des erstellten Benutzers als Response. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **2** |
| Beschreibung | Im Backend mit POST Request einloggen. |
| Voraussetzungen | Backend gestartet und am Laufen. |
| Testschritte | * Öffne Postman und erstelle einen POST Request mit der URL <http://localhost:8800/api/auth/login> * Füge im Request-Body die folgenden Daten hinzu: * email: [Test-E-Mail] * password: [Test-Passwort] * Request absenden |
| Erwartetes Ergebnis | Der Server gibt den Statuscode 200 zurück und sendet die Benutzerdaten als Response. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **3** |
| Beschreibung | Im Backend mit einem PUT Request Benutzerdaten aktualisieren. |
| Voraussetzungen | Backend gestartet und am Laufen. |
| Testschritte | * Öffne Postman und erstelle einen PUT Request mit der URL [http://localhost:8800/api/users/{userID}](http://localhost:8800/api/users/%7buserID%7d) * Ersetze "{UserID}" in der URL durch die tatsächliche Benutzer-ID. * Füge im Request-Body die folgenden Daten hinzu: * Beschreibung: [Neue Benutzerbeschreibung] * Stadt: [Neue Stadt] * Ursprung: [Neuer Ursprung] * Beziehung: [Neue Beziehung] * Request absenden |
| Erwartetes Ergebnis | Der Server gibt den Statuscode 200 zurück und sendet die Meldung "Account updated" als Response. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **4** |
| Beschreibung | Im Backend mit einem PUT Request demselben Benutzer folgen. |
| Voraussetzungen | Backend gestartet und am Laufen. |
| Testschritte | * Öffne Postman und erstelle einen PUT Request mit der URL http://localhost:8800/api/users/{UserID}/follow * Ersetze "{UserID}" in der URL durch die tatsächliche Benutzer-ID * Füge im Request-Body die folgenden Daten hinzu: * UserId: [UserId] * Request absenden |
| Erwartetes Ergebnis | Der Server gibt Statuscode 403 zurück und sendet die Meldung „you cant follow yourself“ als Response |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **5** |
| Beschreibung | Im Backend mit einem POST Request Post erstellen. |
| Voraussetzungen | Backend gestartet und am Laufen. |
| Testschritte | * Öffne Postman und erstelle einen POST Request mit der URL <http://localhost:8800/api/posts> * Füge im Request-Body die folgenden Daten hinzu: * userId: [Benutzer-ID] * desc: [Beitragstext] * Request absenden |
| Erwartetes Ergebnis | Der Server gibt den Statuscode 200 zurück und sendet den erstellten Beitrag als Response. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

### 5.2 Testprotokoll Frontend

|  |  |
| --- | --- |
| **Testfall Nr.** | **6** |
| Beschreibung | Einen neuen Benutzer über das Registrierungsformular im Frontend registrieren. |
| Voraussetzungen | Frontend und Backend gestartet und am Laufen. |
| Testschritte | * Öffne im Browser den URL <http://localhost:3000/register> * Auf der Registrierungsseite das Registrierungsformular ausfüllen. * Mit «SignUp» Formular abschicken. |
| Erwartetes Ergebnis | Registrierung erfolgreich abgeschlossen, und der Benutzer wird auf die Login-Seite weitergeleitet. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **7** |
| Beschreibung | Einloggen mit einem registrierten Benutzer über das Login-Formular im Frontend. |
| Voraussetzungen | Frontend und Backend gestartet und am Laufen. |
| Testschritte | * Öffne im Browser den URL http://localhost:3000/login * Auf der Login Seite das Login Formular mit korrekten Daten ausfüllen. * Mit «Login» Formular abschicken. |
| Erwartetes Ergebnis | Registrierung erfolgreich abgeschlossen, und der Benutzer wird auf die Homeseite weitergeleitet. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **8** |
| Beschreibung | Beschreibung Ändern der Benutzerinformationen im Profil im Frontend. |
| Voraussetzungen | Frontend und Backend gestartet und am Laufen. |
| Testschritte | * Öffne im Browser den URL http://localhost:3000/profile/{username} * Ersetzte username mit dem wirklichen Benutzernamen. * Klicke auf den «Follow» Button |
| Erwartetes Ergebnis | Änderungen werden erfolgreich gespeichert und im Profil angezeigt. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **9** |
| Beschreibung | Einen anderen Benutzer im Frontend als Freund hinzufügen. |
| Voraussetzungen | Frontend und Backend gestartet, am Laufen und Benutzer eingeloggt. |
| Testschritte | * Öffne im Browser den URL http://localhost:3000/profile/{username} * Ersetzte username mit einem wirklichen Benutzernamen. * Füge im Request-Body die folgenden Daten hinzu: * userId: [Benutzer-ID] * desc: [Beitragstext] * Request absenden |
| Erwartetes Ergebnis | Benutzer wird erfolgreich gefolgt, Button ändert sich zu «unfollow» und die Followers Anzahl steigt beim gefolgten Benutzer. |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

|  |  |
| --- | --- |
| **Testfall Nr.** | **10** |
| Beschreibung | Einen neuen Beitrag über das Eingabeformular im Frontend erstellen |
| Voraussetzungen | Frontend und Backend gestartet, am Laufen und eingeloggt. |
| Testschritte | * Öffne im Browser den URL http://localhost:3000/ * Schreibe im «What’s going on right now» Input die Post Description rein und füge ein Bild hinzu. * Schicke den Post mit dem «Post» Button ab. |
| Erwartetes Ergebnis | Post wird erfolgreich erstellt, der Post kann auf der Seite gesehen werden und hat ein Bild. Der Post hat keine Likes und ist von «a few seconds ago» |
| Testergebnis | erfüllt |
| Testdatum | 03.11.2023 |
| Bemerkungen | - |

16 Factsheet

# 6. Inbetriebnahme

### 7.1 Voraussetzung:

* Node.js
* MongoDB
* Yarn oder NPM

### 7.2 Installation

Um LinkUp zu starten, muss man diesen Schritten folgen:

1. Das Repository klonen und zum Repository navigieren:

git clone https://github.com/simonstreuli/Social-Media-Application.git

cd Social-Media-Application

2. Die benötigten Dependencies herunterladen.

cd backend

npm install

3. Erstellen von. env File und hinzufügen von der Datenbank-Connection

cd backend

.env File erstellen

Kostenfreies Cluster bei MongoDB erstellen und den Connection String in. env File einfügen. Danach muss man nur noch das "Password" mit seinem Passwort ersetzen.

MONGO\_URL=mongodb+srv://<user>:<password>@cluster0.phhwfao.mongodb.net/?retryWrites=true&w=majority

4. Das Backend starten

npm start run

5. Navigiere zum Frontend und installiere die benötigten Dependencies

cd ../frontend

npm install

6. Starte das Frontend

npm run start

6. Öffne den Browser auf <http://localhost:3000>

# 8. Reflexion

### 8.1 Reflexion

# 9. Quellen

### 9.1 Quellen

|  |  |
| --- | --- |
| Quelle | Beschreibung |
| https://www.mongodb.com/docs/ | MongoDB Dokumentation |
| https://mui.com/material-ui/material-icons/ | React Icons |
| https://legacy.reactjs.org/docs/getting-started.html | React Dokumentation |
| https://www.youtube.com/watch?v=SqcY0GlETPk | React Tutorial |
| https://nodejs.org/en/docs | Node.js Dokumentation |
| https://www.youtube.com/playlist?list=PL4cUxeGkcC9iJ\_KkrkBZWZRHVwnzLIoUE | MERN Stack Tutorial |
| https://looka.com/ | Logo erstellen |
| https://www.youtube.com/watch?v=ldGl6L4Vktk&t | Social Media API |
| https://miro.com/ | Wireframes und Notizen |
| https://trello.com/ | Kanban Board und Gantt Diagramm |
| https://www.w3schools.com/ | Information für CSS |
| https://chat.openai.com/ | Hilfe bei Problemen ChatGPT |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |