Was wird benötigt:

- Intellij für die Entwicklung in React

- Pycharm damit man den Login Server, Registration Server und den Settings Server laufen lassen kann (nur python installieren und damit ausführen würde auch gehen aber Pycharm ist so wie Intellij sehr bequem wenn es um das nachinstallieren von Libraries geht)

- Node.js muss installiert sein: <https://nodejs.org/en/download/>

Im ordner Frontend ist das React Projekt. Dieses kann in Intellij geöffnet werden. Nachdem ca. 100 MB Pakete nachinstalliert wurden (die habe ich für das Hochladen auf git weggelassen), kann man das Projekt starten, entweder über Play-Button in Intellij, oder, wenn das nicht funktioniert, in der Konsole in Intellij den Befehl **npm start**eingeben. Dann sollte sich automatisch der Browser auf localhost:3000 öffnen und man die Startseite sehen. Von da aus kann man schon die Sachen ausprobieren, sobald man einen Login oder Registrierung ausführen will, bekommt man aber einen Netzwerk Fehler, weil die Python server noch nicht laufen.

Python Server starten: Die Ordner loginServer, signupServer und settingsServer jeweils als eigenes Projekt in Pycharm öffnen. Es sollte sich alles selbst konfigurieren. Eventuell muss Flask, das Paket mit dem ich die REST Sachen gemacht habe nachinstalliert werden. Dazu einfach in die Konsole in Pycharm den Befehl  
**pip install (name des fehlenden Pakets)**eingeben.

Wenn alle drei Server laufen, sollten am Frontend alle Sachen funktionieren. (Zum Testen ob die Python Server laufen, habe ich im loginServer eine HTTP GET und eine POST Mapping zum testen gemacht. Wenn also der loginSever läuft, sollte man im Browser wenn man **localhost:5001/get\_test**eingiebt die ausgabe "hallo get" bekommen, sonst läuft der Server nicht richtig.

Erklärung des React Proekts:

Die einzelnen Seiten sind alle im "src" Ordner und haben eine .js - Endung, in einer solchen Datei ist immer Logik und Design in einem Drinnen, oben der Java Script Code für die Logik und unten der quasi HTML Code für das Design. Die Hauptseite ist in App.js, die ganzen anderen Seiten sind im Ordner "src/pages".

Von einer React Seite aus kann man REST Requests machen, ich habe dafür Axios verwendet und nicht die fetch Methoden von React selbst. Man baut sich also einfach einen Request zusammen, sagt welcher Typ (GET oder POST), welche Daten mitgeschickt werden und so, und erhält eine Response zurück, von der man dan die Daten auslesen kann. Zum Beispiel für den Login funktioniert das konkret so, dass man auf der Weboberfläche seine Daten (user name, password) eingibt, diese werden als state (quasi lokale Variablen der loginseite) zwischengespeichert. Wird dann der Button "Login" gedrückt, wird eine Methode aufgerufen, die auf das Button Event reagiert ("handleSubmit"). In dieser Methode wird dann die REST anfrage gesendet. Es wird also ein Request aufgebaut und an den Server gesendet. Der Server (Python programm login\_server) nimmt diese Anfrage, verbindet sich mit der MongoDB und schaut ob user name und password gültig sind und sendet zurück ob es erfolgreich war oder nicht (alles verpackt im json Format). Wenn der Login erfolgreich war, wird das lokal im Browser gespeichert. Im Browser wird also mit **localStorage.setItem('user', username)** der username gespeichert. Damit bleibt der User eingeloggt und wir können auf anderen Seiten personalisierte inhalte anzeigen. Welcher user gerade eingeloggt ist kann man nämlich einfach über   
**localStorage.getItem('user')**abgefragt werden. Zusätzlich habe ich auch noch das Passwort im local Storage gespeichert, weils für manche services eventuell praktisch ist, für das abfragen der Persönlichen Daten auf der Profil seite habe ich das verwendet. Damit man wieder ausgeloggt wird, kann man einfach den localStorage löschen, mit dem Befehl  
localStorage.clear()  
Das wird zum Beispiel auf der Logout Seite gemacht wenn man auf den Logout Button klickt, es wird aber auch immer beim neu laden der Hauptseite gemacht  
**componentDidMount() {  
 localStorage.clear()  
}**Um Seiten zu entwickeln auf die man nur kommt wenn man eingeloggt ist, ist es vl sinnvol das auszukommentieren, sonst muss man sich ständig neu einloggen.

Kurz zusammengefasst:

- Login funktioniert so dass lokal im Browser im lokal Storage der username gespeichert wird. Dadurch können wir auch von jeder anderen Seite aus abfragen wer gerade eingeloggt ist und die persönlichen Inhalte anzeigen.

- Von den einzelnen Pages aus kann man über Axios relativ einfach HTTP GET und POST requests an einen REST Server machen.

- Eine Zwischenschicht Api habe ich nicht gemacht, weil ich den Sinn davon nicht wirklich verstehe, es macht alles noch komplexer aber ich sehe nicht wirklich einen Nutzen darin, weil man von den einzelnen Pages aus eh direkt REST Requests an die Server machen kann. In den Online Tutorials habe ich es eigentlich überall so gesehen, dass direkt von React auf die REST server angefragt wird.

- Das Backend der einzelnen Mikroservices, müsste also so aufbereitet werden, dass es Anfragen erhält, als GET und/oder POST Mappings zur Verfügung stellt. In diesen Mappfings wird die Kommunikation mit den entsprechenden Datenbanken durchgeführt (z.B. neuer User eingefügt, User Daten abgefragt, usw...) und dann die entsprechenden Daten an den Anfrager zurückgesendet. Auf der Reactseite erhält man also die Rückgabewerte der GET/POST Mappings im json Format, aus denen man sich dann die Daten herausholt, die man am Frontend braucht.

Tipp: Fürs Debugging in React kann man mit  
**console.log(...)**  
Sachen testaufgaben auf der Konsole machen. Diese Konsole ist aber nicht die Konsole in Intellij, die Konsole im Browser. Zu der kommt man, indem man Rechtsklick auf dem Webfrontend macht und dann "Untersuchen" auswählt und dort auf den Reiter "Konsole" (ist bei jedem Browser etwas anders).