Roommate Tasks
Implementation and Challenges

The roommate task is a web application to get the tasks of a group of people sharing an apartment or living space in sync and to maintain the daily chores of household. It has a simple user interface to list done the task and then mark it as complete when it is done. The user will have to register in order to use the application. Only signed in users will be able to interact with the task list.

To achieve this web application, we divided the team in the 2 groups for back-end and front-end. The back-end team used node and express mainly to serve the client. The database is in mongodb and the backend can manipulate it as required. The front-end team worked on html, css, react.js and redux. They made the functional components and made it talk to the backend using axios library. The web application doesn't have a lot of user interface, but it has what it needs and can still be called a full-stack application.

There were many challenges faced during the formulation of back-end. One of the teammates who was working on gateway was not able to receive the data when the request was coming through gateway and hitting the end point. The axios post request was going correctly without gateway (directly to the endpoint) but it was having trouble when sent via gateway. It was solved during one of the group discussions. There was another challenge when we were trying to configure the Redis server. We used cookieparser to get the cookies but somehow it was not able to read it. It turned out that we were not sending the withCredentials as true with the request. It was a small thing, but it was needed, and it was discussed in the class as well.

The front end was laid smooth initially as there were only html elements inside the return statement of React's functional components. But as we started going further and started using effect then it was a little rough patch. The redux was taught well during the class so there was no difficulty in configuring reducers, actions and then dispatching them when required. The css was integrated in last few weeks but that was done with few hiccups.

The components which were hard to figure out were kafka, web-socket and docker. The web-socket was still easy as we were able to get the count of how many users are connected easily but then sending the data over web socket was something else. It interacts with many files, so it was necessary to follow the naming style. Kafka was being used in the same service in the beginning and then I had a talk with professor who led me the right way. The producer and consumer are supposed to be used separately and consumer must be initialized only once. We learnt it late, but it was good knowledge and then with few doubts, were able to execute it.

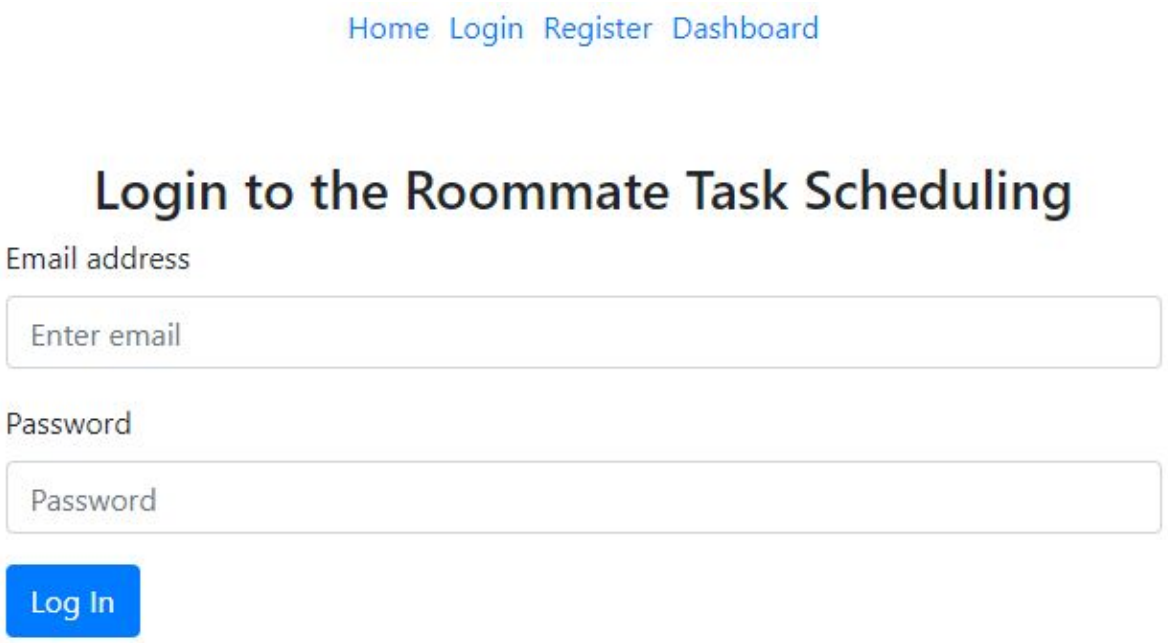
Docker looked easy at first sight but there is much more to it than what comes to eye. It was discussed with another team and they explained us better to compile the docker file.

Overall, the project was fun to work on, not only project but all lectures. We learnt many things throughout the process. It's different in that there were few bad days, but we were happy that we got to learn in the end.

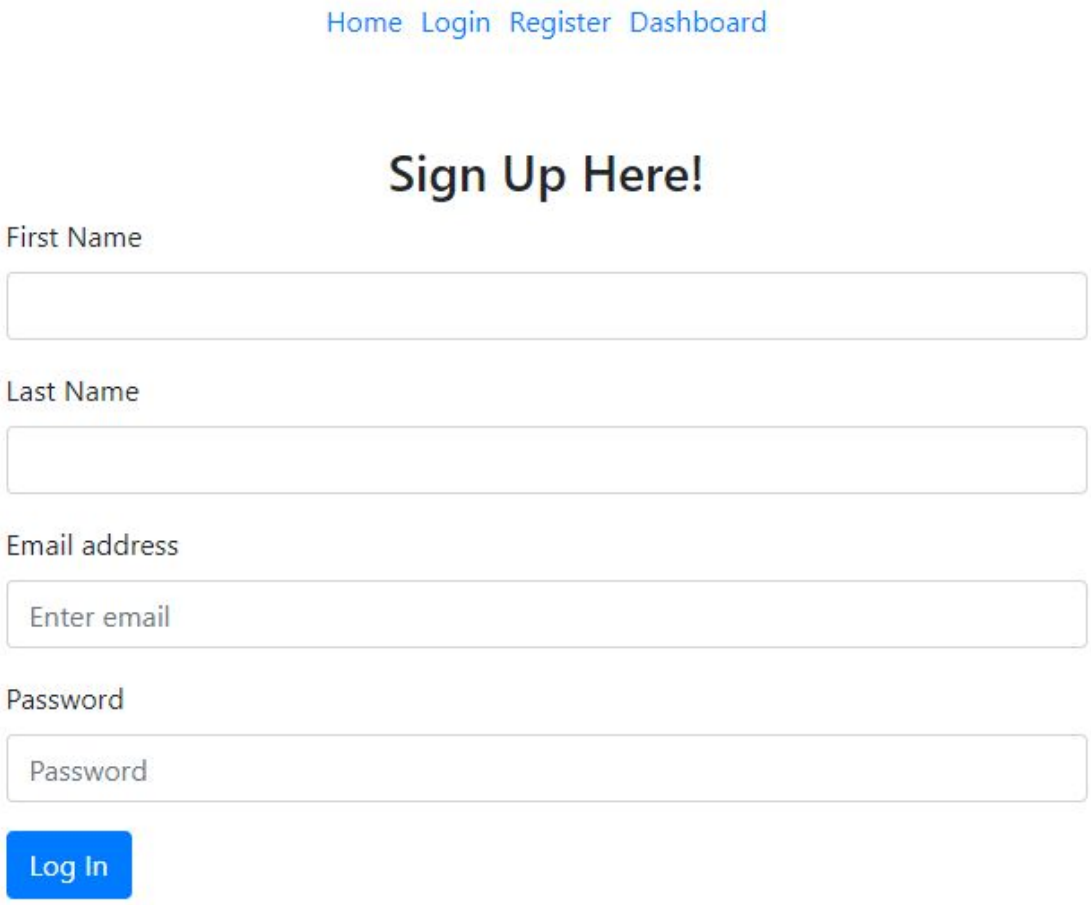
Thank you, Brian, – for making this course fun and project, a really learning experience.

Screenshots

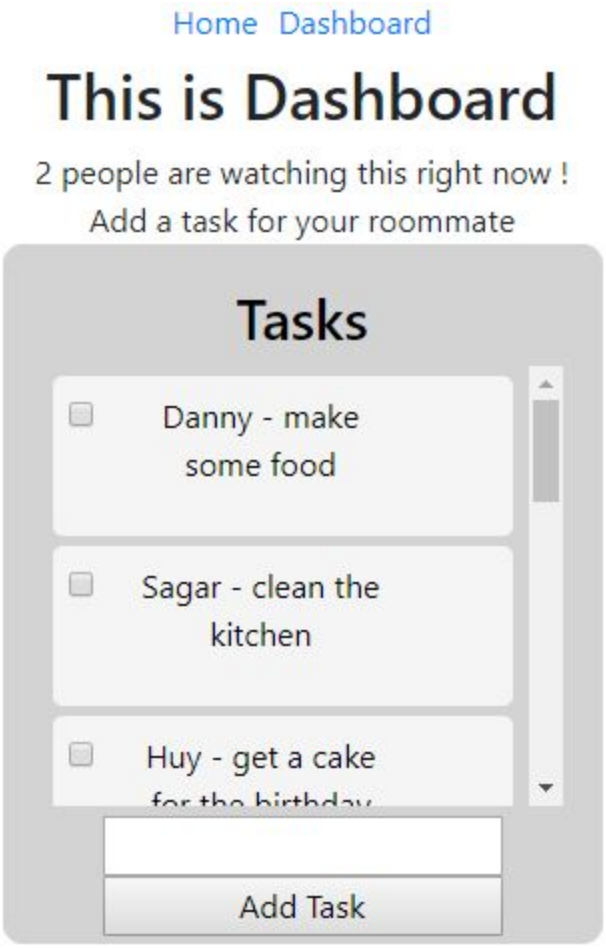
1. Login Screen



2. Register Screen



3. New Task List



4. Done Task List