Apurva Shah, Abishan Sutharshan, Ayesha Alvi, Mark Benliyan, Jesus Ortega
Professor Eggert
CS35L
03/15/2023

CS35L Project Report

Introduction

Bruin Subleasing is a web-based application designed to solve the common feat of subleasing or finding temporary living situations in the UCLA residential area. Students do not have a reliable platform to search and post sublets and currently use external unofficial platforms. Features included in this application are authentication for posting control, viewing other listings, and adding or removing posts accordingly. The back-end database and authentication is achieved using the Google app development platform Firebase. The front-end is built with Next.js (a React.js framework) and styled using Tailwind CSS and Material UI. Our main obstacles dealt with authentication, dynamic routing using slugs, and deleting posts.

The Problem

Subleasing is a common but unofficial practice college students deal with. The quarter system allows students to take job opportunities, study abroad, or remotely enroll for small windows of time, yet most of them cannot afford to maintain their current lease simultaneously. Finding a person to take over your lease is a cumbersome process that is scattered across several platforms such as Facebook Marketplace, where users are faced with endless outdated posts, bots, and advertisements. The current process entails posting, waiting days on end for a response, and answering redundant questions regarding apartment features and timeframes.

Features and User Navigation

The primary issue is the absence of a streamlined service with adequate features to deal with subleasing. The leaser must be able to upload and delete posts that are expired; posts should be unique to their own account, requiring authentication. On the lease searcher’s side, students should be able to easily search through listings, see another user’s other posts, and contact them easily. There is no specific template laid out for posting subleases on Facebook Marketplace, often leaving unanswered questions to answer in individual messages. We require specific input

fields such as number of roommates, amenities, and timeframe, and rent prices. This way, all posts have the necessary information for users to consider.

When users visit our application, a home screen displays further information regarding our mission and resources to consult about faulty posts. Users are prompted to make an account before they can access any of the other tabs. From there, they can go to the “Add Sublease” tab and fill out the proper input fields. The clear input feature at the bottom of the page allows users to start from scratch for undesirable postings. Users are able to quickly select amenities by highlighting the appropriate icons. The subleaser can view their own listings under their profile icon. On the other hand, if users are searching through posts, they can go to the “listings” tab after making an account and view other user’s postings. The search tab further facilitates browsing by street name.

Implementation and Architecture

Our project's frameworks consisted of using Next.js, Tailwind CSS, Material UI, Firestore, and FirebaseAuth. For the front-end framework, we used Next.js for its efficient server side rendering, allowing students of all socioeconomic backgrounds to access our platform on any device at the same speed. Tailwind CSS was used for styling since it allowed for rapid development with its predefined components/styles and was a relatively lightweight package implemented into Node.js well. Lastly, the Firebase SDK provided us with authentication options and Firestore, a NoSQL database that allowed for flexible documents to be stored.

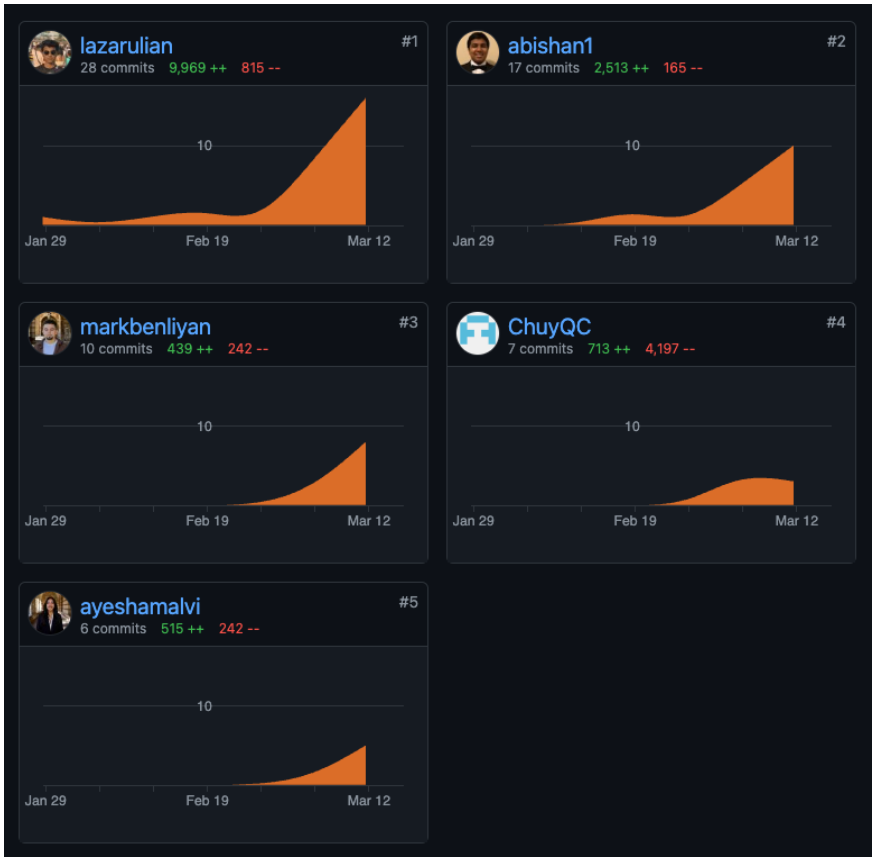
Starting with authentication, we used FirebaseAuth with email/password. On creation of a user, we also created a Firestore collection called Users that collected metadata such as Full Name, Profile Picture, and a unique hashed UID. So the registration form appeared to be one form but posted data to the authentication module and our “users” datastore. We then created a React context to maintain login information throughout the project and protect pages when not logged in. Moving onto post details, when a post is created, the logged-in context allows us to link the sublease-input form details with the current user’s unique UID. Thus, each post is linked to a user through this unique UID and can be queried using either the post-meta-data or a user’s unique UID. This relationship allows us to create dynamic routes for each user using their unique

UID (querying their posts) and dynamic routes for each headquarters using the user's document-reference hashed ID. This is accomplished using Next.js's dynamic routing.

Obstacles and Future Plans

Storage requires API calls which can eventually be costly. To combat this, our project limited API calls with autocomplete. We attempted to store images using AWS S3 buckets, but we would have to implement a Node server which has limited storage for free. For a more cost effective solution, we prompt the users to provide image URLs. Apurva worked on Authentication, but this required several considerations including react contexts to pass authentication requirements, edge cases for incorrect inputs, and data normalization. Another issue was linking UIDs to posts for addition and deletion. In the future, we hope to embed Google Maps APIs to each post, enable post editing, and insert a chat feature so students can directly communicate with potential subleasers.

Git Commit Usage



Individual Contributions

Jesus worked on styling the components for the Home and About page. One of the main features he added was the hamburger menu that appears in the navigation bar in a mobile view. Abishan worked on the Add Sublease page and the Search page, which connected the front-end to the back-end. He also worked on the Clear Input feature. Apurva worked on the Authentication for the website, which includes the contact card and the "My Listings" page. Ayesha worked on styling the individual postings after users input sublease data. Mark worked on the card components that render all of the listings on the platform and the grid that renders them. Mark also worked on adding the Contact page found in that navbar, which links with EmailJS and allows users to input their form information and send an email directly to us. Mark also helped with styling changes on the home page.

Ayesha Alvi

I worked on styling the individual postings and deciding what fields to include in the subleasing upload form. I have little experience styling with Tailwind CSS, but learned the best practices to make horizontal versus vertical grids, centering, and spacing. I was able to make my code more readable than my past projects by better utilizing components and separating the posting into the post itself and the amenities. I learned a lot about linking components together, but the biggest challenge was figuring out how to conditionally make properties given a certain input. It took a bit of searching, but I was able to solve the issue with simple syntax fixes. I was also able to make the props I passed into my post component a bit cleaner by placing all the amenities in a dictionary.

I also played with a few styling features like hover and expanding. I learned to work with flex boxes to make the pages more mobile friendly. It was cool to see how my team members linked the back-end database to this component I made. I went through several stylistic iterations of this page and realized styling without a plan is incredibly difficult. I designed some potential pages on Figma using inspiration from other listing websites, and was able to design the final iteration much faster. Using material UI components also made for easy manipulation.

Jesus Ortega

I worked on creating all the frontend behind the entire home page and the about page. I was tasked with creating the navbar component, which included buttons that link to the: Home page, sublease page, listings page, and search page. I spent a lot of my time reading the react and tailwind languages to get a feel on how to implement the bar. I learned how to implement the hamburger menu, when the screen was small or minimized I made it so a hamburger bar would show on the top right corner. I implemented every button to react to being hovered to, noticeably every button lights up once you hover your mouse over it. I decided to make the website sleek, simple and clean with a tech inspired dark blue and white look. I implemented both the support page and info tabs at the bottom of the home page. This included using React and Tailwind to style links to support pages in our website, in case the user needed to report any problems to ideally a support email that would be run between all of us. After creating the first couple of info tabs which can be seen as the dark blue tab at the bottom of our page, I started to become exponentially more comfortable in coding in react.

Apurva Shah

For the project, I was in charge of working on Authentication, Dynamic Routing, and Database Design. I started with authentication, one of the more challenging tasks that I have never worked on before. To implement authentication, I first read up on the Firebase Auth documentation and asked ChatGPT how authentication was implemented within a React project. The firebase documentation allowed me to understand and implement the functions that i would need to use to login, logout, and register a user. ChatGPT pointed me towards React documentation for their context hooks, which would allow for logged in parameters to be passed as a global “context” throughout the entire app.

This was challenging to implement as understanding the edge cases for registration was not something I thought to do initially. When implementing the registration form, I had to think about how we would store the data that we would collect for the users such as their personal information. It was clear that we would need to link the auth module to user data and store it in a separate firestore collection so that we could query the data on command based on the authentication modules unique hashed UID for each user. When creating the context, I allowed

for calling the UID using useAuth() so that we can query both posts and information at will when users are logged in.

Dynamic routing was implemented by next/router. To create the dynamic routes, I planned to pass the UID and the POST ID as parameters that would serve as the points to query in the components that would be called. This allowed for each user to display all of the posts that contained their UID and each post to have its own page under its UID. A lot of challenges in this portion had to do with database design. I had to redesign the database such that each post would store the document references and the uid of the poster.

Outside of this, I did do a lot of work in terms of linking the entire application together and catching a lot of the bug fixes that were happening. I worked on page styling, fixing issues with components, and doing general full-stack work. I also setup the project directory which is why it appears that I have so many lines of code written.

Mark Benliyan

During the Bruin Subleasing project, I focused on two core features: 1) implementing sublease listings as cards, and 2) creating the contact form. I designed the Card and CardGrid components to fetch data dynamically from Firebase and display sublease listings effectively.

To implement the sublease listings, I built the Card and CardGrid components using React and Tailwind CSS. This allowed for a visually appealing and user-friendly display of listings fetched from Firebase. The dynamic rendering of subleases made it easier for users to view and interact with available options.

For the contact form, I used EmailJS to configure the API call and managed React state to make the form functional. I then integrated the /contact route into the navbar to provide users with easy access to the form. Here I had to learn a ton of new things about how to manage React state and interpret the users typed input to the form. Additionally, I polished the UI by updating all our link CTA text, enhancing the spacing and layout of the info tab divs, and restyling homepage icons.

Throughout the project, I also focused on maintaining a clean and organized codebase by moving card components to a dedicated folder, removing stale code and imports, and fixing sublease route issues. I also wrote the guides in our README documentation to facilitate easier interaction with our project both by our own teammates and reviewers.

Overall, the Bruin Subleasing project allowed me to hone my skills in React, Tailwind CSS, and code organization. I'm confident that my contributions have led to a more intuitive and visually appealing user experience.

Abishan Sutharshan

I worked on the Sublease Input page, which adds apartment sublease listings to the database. This consisted of connecting the front-end to the back-end using API keys. Then, I had to import things from Firebase/firestore, like db, collection, doc, and addDoc. The collection I used in the Firebase to store the apartments was called “apartments,” and I retrieved the collection into the front-end. Then, I used addDoc to add a new listing to the site. I also completed the Clear Input feature on the Add Sublease page, which was one of our main features.

I also worked on the Search page, which queries for apartments that match the same street name in the search box. This allows users to meaningfully search through server-side data, and return results that are most relevant to their search. Finally, I worked on some finishing touches, like fixing styling issues on various pages (like the Sublease Input page and the About page) and linking buttons/text to their respective pages.

In conclusion, I learned a lot from this project, like how to query into the Firestore database with React. I also generally became better at using React, like following React’s stateness rules to create the web app. I also learned how to style elements with Tailwind CSS and Material UI. With these two styling tools, I added small animations and reactive elements to the things I worked on, which made the front-end visually pleasing. Overall, I am pleased with the outcome of our project.

Works Cited

Next.js by vercel - the REACT framework [Internet]. by Vercel - The React Framework. [cited 2023Mar17]. Available from: https://nextjs.org/

Firebase [Internet]. Google. Google; [cited 2023Mar17]. Available from: https://firebase.google.com/

React [Internet]. – The library for web and native user interfaces. [cited 2023Mar17]. Available from: https://react.dev/

Tailwind CSS. [cited 2023Mar17]. Available from: https://tailwindcss.com/

The React Component Library You always wanted [Internet]. MUI. [cited 2023Mar17]. Available from: https://mui.com/