CS 35L Final project report

Team member names, preferred email addresses, and assigned section and TA

- Team Member Information:
  - Naman Modani, modani@g.ucla.edu
  - Eavan Jael Soriano, eavanjael@g.ucla.edu
  - Le'on Norfleet, leonn35@g.ucla.edu ← this is my submission
  - Kevin Chen, kechen8@g.ucla.edu
  - Sunehra Rahman, sunehra0714@g.ucla.edu

- Assigned Section: 1B

- Assigned TA: Shi, Z.

Project name or Team name, if you have one

- UCLAble

As a Math of Computation major, the courses I attend every day are a fair distance away from each other. A lot of the time I spend on campus involves walking, and with tight schedules I usually only have at most a 10 minute grace period between classes. Most of the time I am able to make it to the next class with 2-3 minutes before it starts, but if the shortest path is somehow blocked or inaccessible, I won't be able to make it in time. This quarter, a majority of my classes do not record lectures, so I can miss important information for being late. As I walk from class to class, I always see construction being done. I began to wonder why there is no announcement or warning that is easily accessible for students. I also thought about how this would affect students who have accessibility issues. If the main ramps and sloped paths would be closed, or an important elevator was closed on short notice, what would those students do if they depended on that route and did not memorize any other ones.

This was the main idea behind my group's final project, an app called UCLAble. The app's purpose is to serve as a message board and announcement system that students can use to inform each other about things ranging from out-of-order elevators, to cracked pavement, to poor lighting in hallways in order to help one another stay informed about possible obstacles while navigating on and near campus. The app was built with a MERN stack, including MongoDB,

Express, Node, and ReactJS. MongoDB was used as a ‘cloud storage’ for the data that could be uploaded and viewed by users. User accounts and the posts were stored in JSON documents for easy parsing and data manipulation. Express and Node were used to create the Node.js server that would handle calls made by the client(the app) in order to post a report, get account information, or like a post. The features were developed from scratch and the only libraries used were MongoDB for compatibility and express/cors to make sure data was being accessed securely. React constitutes the buld of the backend behavior of the app itself and its communication with the Node server. Significant libraries used for backend development were axios for http requests, the react-select library for an easily creatable dropdown, and google oauth for a simple and secure way to create and collect data needed to create accounts for the app. React and CSS styling were also utilized for the frontend aspect of the app.

The main thing I looked to inspiration for while developing the web app was reddit. Being able to post and upvote seemed like an obvious choice to include in a message board style application. However, one thing reddit lacks is the ability to search for specific posts easily so adding that was also important. In addition to being able to post reports for locations and UCLA and viewing those posts, users are able to upvote posts they believe were helpful, which can increase the change of it being viewed by others who may also think the same. The main ways for viewing the reports is by recency or by its number of votes. By recency is good if a user wants to be kept up to date with current events and by votes can keep the most helpful votes viewable without needing to do further searching. In addition, if there was a specific post and it gets lost in the large number of posts that could be made in a day, users can search by title. Or, if there is a specific location a user wants to be kept up to date on, they could also search by location. Dark mode is another feature that the app has, which can be used for anyone who finds the original color palette to be too bright. Finally, users can see their account information in the profile page, which also includes how many posts they have given an upvote.

I contributed significantly to the backend development for the app and server. For the server, I set up the initial environment and created all of the functionality that allows the app to interact with the MongoDB database. For the app itself, I created all of the components(except for Profile.js), the sorting functions, and implemented google authentication for user login and logout. I also contributed to the profile page and helped with bug fixing.

Over the course of development, I faced a few challenges while implementing the main features of the app. The main challenges were related to the view reports page. I wanted there to be an infinite-scroll-esque effect to where the user could keep scrolling to look through the reports without pressing a next page button, so I wanted a way to generatively create components for the reports without having to write a significant amount of code, where there would be a blueprint that is filled in with the post data as it is pulled from the database. I first had trouble implementing this because I was also thinking about how to combine it with voting. each button

should be corresponding to a specific post, and I didnt want it to have to search or do array iteration to keep it fast. I eventually solved this problem by making each button also mapped to the posts and created one single function they all call. That way, the button would use its own index information as the function argument and then each call would be different depending on the button pressed. An extension of this problem is when voting had to be fully implemented. I was struggling with finding a way for the button to update the vote count on both the database and the app dynamically without needing to reload the page. Eventually, I thought that the best way to do this was to make both behaviors dependent on each other through an http put request. That way, whenever the server updated the vote count in the database, it would send a response back to the app and then the array of components would dynamically update by re-mapping itself to show the new vote status. This also fixed the issue of a single user being able to vote on a post more than once by reloading page. Another challenge that the team faced was implementing account authentication. The task was originally given to another member, but it caused other features such as voting to be on standby until it was completed and other parts of the app to be non-functional, so we had to re-vert the repository to before it was implemented, along with a majority of the styling to reduce dependency bloat. After this I made sure all backend was fully functional so that styling could be re-added consistently and without any bugs. Kevin and Sunehra contributed to the profile feature but due to merge issues, some of their contributions are not present on the main branch. In addition, we had to do a git revert to a point a week prior to do bug fixing so the group's progress can also be seen on the profile2, prof-feature, and redo branches. I believe that the total contribution to the backend development can be split 75/15/10 between me, Sunehra, and Kevin respectively because they did alot of work on the profile page and even though it was refactored their prototype helped give an idea of how to streamline the application. They also almost fully implemented Profile.js on their own but they were using an un-mergable version of the branch so I had to enter their code to the main branch manually. Even though it was just one component, it was the part that took the most time to achieve and get working properly. This can be seen in the commit history.

If I had more time, I would improve the profile page to include the total number of posts, the top 5 most upvoted posts, and the ability to make a custom user name, where on the posts it would show the custom name and their email in parentheses, like: joebruinOnTop55(jbr559@g.ucla.edu). Also a comments section would be a cool feature to have, allowing the students to communicate about the issues without just looking at the static report. Another feature would be the ability to favorite posts to keep track of the ones most important to you without having to search for it. Im not sure about the feature to close/edit/delete posts because that could cause issues or be abused. If I could do this project over again, I would only change the profile page to include more personal information and add the previously mentioned additions.