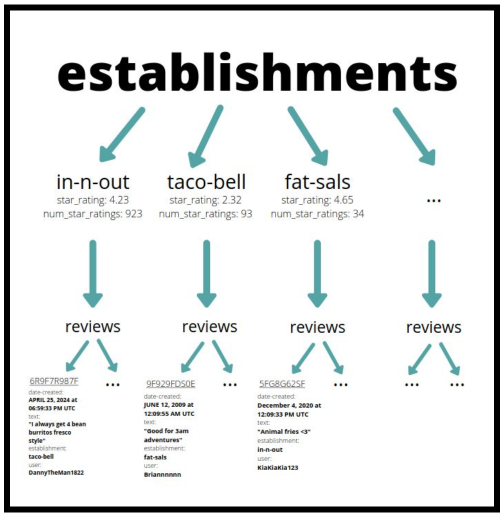
Charles Zhang
Discussion 1A

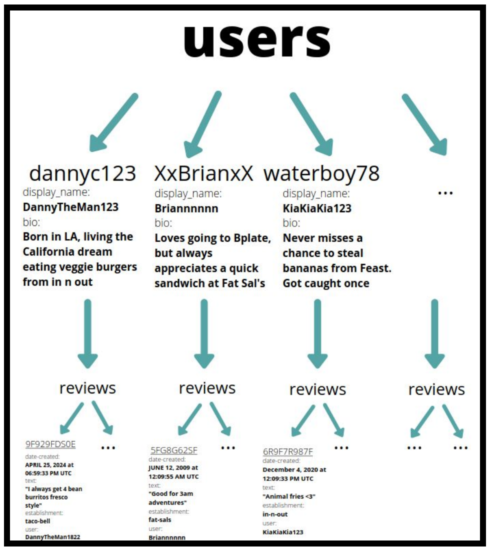
Purpose

The goal of BruinByte was to essentially create a specialized version of Yelp! for UCLA students by restricting our range to the Westwood area. After authenticating themselves and creating an account, users are allowed to write reviews and rate any of the 15 restaurants currently built into the page, all of which are restaurants within walking distance of the UCLA campus. From here, we handle the aggregate of all users' responses for each establishment, displaying all reviews and the average star rating on each restaurant's card. In addition to this, we also provide some extra information about each restaurant, including address, contact information, map location, and available delivery services. To add an extra degree of personalization, we allow users to maintain their own display name and bio, alongside displaying all of their past reviews on their private profile page. Their display name, bio, and total number of reviews are made public to all viewers of the website.

Architecture

Our app is built on React and the Material-UI framework. We combined this with CSS styling to create the front-end of our website. Our back-end is handled using Node.js as a bridge to our Google Firestore database. Our database recorded information on restaurants and users by the following structure:

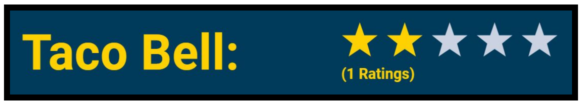




We outsource our user authentication needs to the Auth0 service, which allowed us easy access to a user framework and authentication through Google. We then link this database to our front-end through functions written in JavaScript. Our development was done locally using npm to test through localhost, and later published online through free services at Netlify.

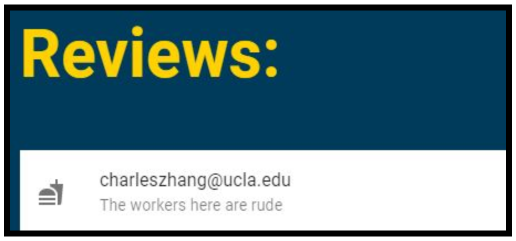
Features

Our website centers around a dynamic review system. Each of our reviews is split into 2 mandatory parts: the star rating and the text review. Both are handled within the Submission component found on each restaurant's information card. Whatever rating and review the user chooses to leave here is then written to our Firestore database and added to the appropriate collections. After the page is refreshed, the information they submitted will then be reflected in the main body of the restaurant card, using multiple reads from Firestore:



Taco Bell: ★★☆☆☆

(1 Ratings)

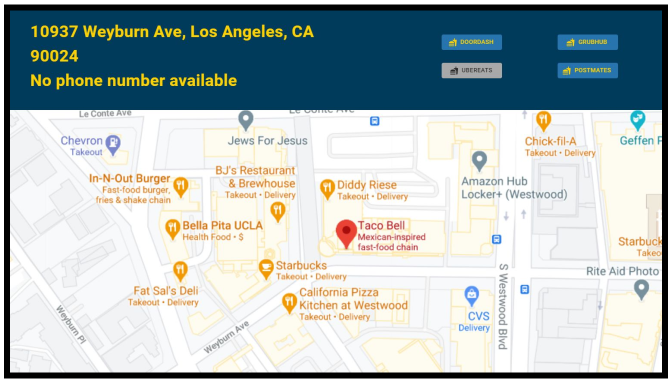


Reviews:

charleszhang@ucla.edu

The workers here are rude

In addition to this dynamic data, our restaurant cards also contain a lot of static information for the users' convenience:

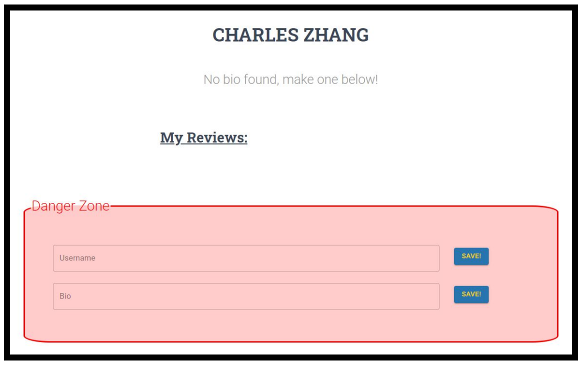


10937 Weyburn Ave, Los Angeles, CA 90024

No phone number available

Each establishment is statically associated with its address, phone number, map location, and delivery service availability. Due to a lack of foresight, this information is actually hard-coded into our website, rather than leveraging our database to access the information instead. The goal of this feature was to give our users tangible information on the restaurant beyond other users' opinions. It also ideally provides a degree of convenience if the user were to choose to order from the restaurant.

Our final feature is the maintenance of a user profile. Each user is given access to a private profile page:


CHARLES ZHANG

No bio found, make one below!

My Reviews:

Danger Zone

Here, the user can view their public display name and bio, alongside all their past reviews. At the bottom of the page, there is a "danger zone" where the user can modify any of this information using the username and bio text fields. When clicking the "Save!" button on the right of each field, the user initiates a write to our database, which accesses the user (indexed by a hidden username initialized when the user created their account), and modifies the appropriate field. The information at the top of the page is then created by reading from the user's fields in our database, and then displaying them.

Personal Contribution

Personally, I started the project by working on integrating user authentication into our application using Auth0. This involved setting up Auth0 itself, while also placing the useAuth0() hook into the correct places in our application to protect routes. Once I finished that, I set up the framework for our page navigation using React Router, while also designing the homepage and profile page. I then helped out our front-end team in developing the components we would need to make our application operational, specifically the components that handled review submission, star ratings, and any elements related to user authentication. Towards the end of the project, I worked on cleaning up our code and integrating the Firebase functions built by our back-end team into the necessary components. In between all of this work, I attempted to style our elements using CSS to the best of my ability.

Obstacles

I think that the biggest difficulty our team had was getting started. Learning enough React to get something down on the page was a huge obstacle, and resulted in a lot of badly written code that would get propagated down into later iterations of our application. For instance, we failed to learn how to generalize a component using props until much later in our development, resulting in us having a separate component for each restaurant. Although some of these oversights were fixed later in the project as we improved, we decided that others weren't worth the time. This resulted in a lot of inefficient code and headaches for us. Another big struggle for us was integrating Firebase. None of us had any experience dealing with asynchronous behavior or promise handling, so we were caught dealing with those issues for an extended period of time. Debugging took us a very long time and caused a lot of confusion between our teams. Thankfully, with the help of the back-end team, we ended up resolving those issues and got the database working in the end, although there are still some oversights that could've been fixed.

Possible Improvements

There are plenty of improvements that I would have liked to make given extra time. For one, the ability to add a restaurant to our current list would be pretty key to this application's usability. However, since we didn't link the front-end to our database until relatively late in our development, we had to deal with other issues and didn't have time to make this addition. Also, we struggled a lot with integrating our user information into the database. This resulted in some unforeseen changes, including the removal of profile pictures and a lack of a public profile page. This significantly takes away from user identity on our website, but was necessary for certain functionalities. I really would've liked to spend more time to get both these aspects working. Another huge issue with our current implementation is a lack of portability. Our entire layout is unusable on mobile devices and browsers other than Chrome. None of our CSS really accounts for resizing of screen size, resulting in a very messy layout on certain devices. With time, this could have been fixed, but it wasn't a priority as it didn't directly relate to the app's overall functionality. Finally, I probably would've decided to add a search bar and sorting feature to our restaurants page. As it stands, these are probably unnecessary since we only have 15 establishments represented, but both would be useful if that number were ever to expand. Even now, both could've been convenient for the user and made the site more interactive.