Group 6: Colton Tucker, Tyler Gerth, Rylan McAlister, Dalynn Hatch, Brayden Alvarez, Landry Ishimwe Karangwa

UNIVERSITY OF ARKANSAS
College of Engineering

API Documentation

ReZerve

4/29/2021

Firebase

Firebase is used as our database and user authentication schema. We used Firestore within Firebase (a no-SQL database with JSON hierarchical collections) as the main storage for our application, and we designed it to be as close to normal SQL as we could (in terms of linking different collections together via document keys). We found that it would be the best to implement due to its scalability and flexibility. It was implemented over time and slowly added into the application, refactoring as necessary until we finished our pages. Testing was done with mock data. To bring Firebase into our website we created interface files whose only job was to keep all of the variables and data types consistent throughout. From there we imported the respective Firebase libraries and created development rules on how everyone should use Firebase consistently.

Our main database collections were for businesses, customers, and employees, and each of these might be associated with sets of messages and appointments. Then, to access this data, we used Firebase documentation to create queries for specific user data, rather than pulling everything into our application and sorting from there.

For the Authentication, we used Firebase Authentication to create user accounts for both employees and customers. This allowed them to login to our application and be presented with different user interface flows depending on their role. By using Firebase Authentication, we were able to set up persistent authentication where a user could close down their web browser or refresh the page and still be logged in, due to the fact that Firebase Authentication stores user login information in the local storage of the browser.

Google Maps

The Google Maps API was used so that we could access location data for the various places that a customer might search when looking for a barbershop/salon, as well as for encoding an address into lat/lng coordinates when a business initially signs up and creates an account. We used the Maps API,

the Places API, and the Geocoding API. The Maps and Places APIs were free (up to $200 worth of credit which we never reached), and the Geocoding API cost 0.005 USD per query (so $5.00 per 1000 requests).

Stripe

Stripe is used as our payment API to allow the customers to pay for their bookings. It was used because as compared to similar payment APIs we found that this would be the easiest to integrate due to the documentation available and compatibility with React and TypeScript. On the website during the checkout process the customer is given a prompt for card number, CVC, expiration date, and zip code. There was also another server needed (also hosted on Heroku) to fully implement Stripe. Essentially what it does is allow the website to communicate with the authentication servers that Stripe has in a safe, encrypted, way. During the talks with the server the correct amount of money is taken from the user's card, and split correctly between the business and our product champion (a small fee for providing the application infrastructure for users).

Heroku

Heroku used as our host for the application and Stripe payment server. It allowed us to have our application and server deployed in an "always-on" state that could be accessed any time. It also allowed us to do things like connect out Github repository into it for ease of deployment and store secret API keys in a place that wouldn't be accessible from the code. It was used because it was the hosting service we were all the most familiar with. For the duration of development and testing it was just used on the free plan. It served the purposes we needed for the time being. We spent a total of $7 a month on this specific hosting service once we were ready to go into a more "live" state.