Analysis

**By Benson Gao, Nick McDaniel,
Michael Ly, and Julian Tran**

Use Case #1: Account

User logins and account registration will be handled through Firebase. Firebase's Authentication service is one the simplest to implement platforms with numerous useful features for login including authentication using passwords, phone-numbers, and social login providers such as Facebook, Twitter, and Google. Firebase Authentication doesn't have any limitations or pricing restrictions upon number of registered. There are some limitations on the rate of different operations but the free tier is generous enough for our needs (10,000 email address verification links/day, 10,000 phone sign-ins/month, etc.). If our application grows to a point where we would exceed these numbers, the Blaze Plan allows for dynamic pricing based upon usage, allowing for effortless and affordable scalability. Security would also be taken care of natively, ensuring we meet one of our non-functional requirements.

Use Case #2: Modify Profile

Since we are using Firebase for user logins and account registration, we have access to Firebase Realtime Database. The data is stored as JSON and synchronized in real time to every connected client. Everytime data changes, such as the user changing their preferences or settings, any connected device receives updates within milliseconds. The free tier that Firebase offers allows for 1GB of storage and 10GB of downloads. However, it only allows for 100 simultaneous connections which will likely be exceeded very quickly. With the Blaze plan, as with the other use cases, we will be able to not only pay dynamically for the amount of storage we need but we will also be able to split our data across multiple database instances in the same Firebase project. We will be able to control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

Use Case #3: Discovery

The way that we plan on handling discovery of new locations is going to be based off of using user selected preferences and comparing them with categories of locations/businesses tracked with the Yelp API to create a curated list of things that the user is more likely to be interested in and actually visit or want to talk to someone about. For discovering guides, we will focus on a different style of recommendation system. This one will be based off of the collaborative ratings that the guide has gotten from past people that they have interacted with. When a user looks at a location that they are interested in, or checks the discover guide page to see guides from the local area, they will see ones that have interests that closely align with theirs as well as have a higher overall rating. The current proposed algorithm is:

$$guide\ match\ score = \frac{matched\ interests}{total\ interests} * (0.5 * average\ rating + 5 * (1 - e^{\frac{-number\ of\ ratings}{Q}}))$$

This current iteration of our guide rating algorithm gives weight to those that have a lot of ratings while also making sure that those who have large number of ratings but have lower scores or lower interest compatibility aren't promoted. The variable Q is a constant weight that we can select to determine how much weight the number of reviews should have. This can be modified so that guides with fewer reviews still have chances to compete with established guides. This way, the people with the most relevant and useful information for that particular

user are the ones that show up at the top of the list of people recommended for them to talk and get information from.

The necessary information will be stored in two seperate locations. The data for businesses/locations will be stored within Yelp's own databases, as we will be pulling the information from their own API. When it comes to guide recommendations, that will be implemented with the Firebase database that we will be integrating in our app. On the database, we will store the user preferences that each user inputs. Then when we need to make guide recommendations, all we need to do is pull the information for guides of that location and compare it to the information of the current user to make our list of relevant guides.

Use Case #4: User Interactions

As with other use cases, User Interactions will also be handled through Firebase. Firebase Cloud Messaging (FCM) allows for messages to be sent from clients through our server to other clients. FCM also allows for optional data payloads attached to messages which we will be using to attach dynamic links with location data to allow for the recommendation of specific locations. FCM also allows for messages to go to the notification tray when the application is in the background. FCM also handles when the recipient's device is turned off, not connected to a network by storing and delivering the message when possible. There are no limitations or restrictions on FCM in terms of pricing so when scaled to large numbers of users, there would be no changes in budget or changes that need to be made to scale the project up.

Use Case #5: Post-User Interactions

Post-User Interactions will be stored in firebase database. The rating of a guide will be an average of the total user ratings. The ratings will be attached to the guide's profile and stored in JSON. The ability to block a user will be handled in our user model, if userIsBlocked = true; then the client will not receive messages from Firebase Cloud Messaging from that specific user. The report function will create and store a message containing information about the type of report, the user reporting, the user being reported, and any additional information the user can provide (A paragraph, screenshot, or image). After a certain number of reports have been filed on a guide, the guide will be flagged and temporarily suspended until an administrator manually review the case. After reviewing the case, the administrator will lift the suspension or permanently suspend the account.