Anu Faboro, Alan Simon, Zikora Anyaoku

Keheler

CMSC436

December 5, 2021

# Free Eats:

As a college student, one of life's greatest pleasures is finding free food on campus. Saved from having to purchase food from one of the university's multitude of overpriced restaurants, free food is the holy grail of finds on campus. There's no shortage of free food on campus, especially when departments and student organizations have meetings with leftover food that inevitably end up in the trash due to a lack of demand. Since we were tasked to create an app, we thought an app that would directly support students, whether they commuted or they lived on campus, would be beneficial. Thus, in order to reduce food waste and silence the rumbling in students' bellies, we developed an app to help organizations reduce food waste and help students find free food on the College Park campus called Free Eats. This report will detail our process of developing Free Eats, an account of the challenges we experienced and lastly ideas on improving Free Eats.

Before we envisioned Free Eats, we knew we wanted to develop an app that would be beneficial to students in a meaningful way. So we crafted a proposal for an app that would help students keep track of their sleeping cycles, in a hopefully more accurate manner, than the current Health app would. However, our proposal was rejected and we had to go back to the drawing board. We brainstormed Free Eats and drafted a second proposal, describing an app

that would enable users, consisting of students, student organizations, and university departments, to create accounts, be able to view a map of locations where food is available for students, and add their own food finds for others to find. Free Eats was successfully approved and we began constructing the application.

As we put our proposal together we identified three goals for our minimum viable product. Firstly, users should be able to create accounts and login to Free Eats. Secondly, users should be able to create and submit alerts whenever food is available. Thirdly, users should be able to view the highlighted locations of any submissions on a map. And lastly, users should be able to get notifications whenever submissions are created. We then identified additional stretch goals, such as being able to login using Google or Apple, creating a rewards system based on how active a user was on the app, and creating location-based notifications based on users' preferences. And with these goals in mind, we split our tasks amongst ourselves and began working on the project.

When users first open Free Eats, they have the option of creating an account or signing into an existing account, using a valid email account and an alphanumeric password as shown below in Figures 1 - 3. The login system is powered by Firebase Authentication, which enables login by storing user credentials within its servers and provides authentication services. Additionally, using Firebase Authentication also enabled us to have the option for adding additional authentication methods, such as Sign In with Google, etc.

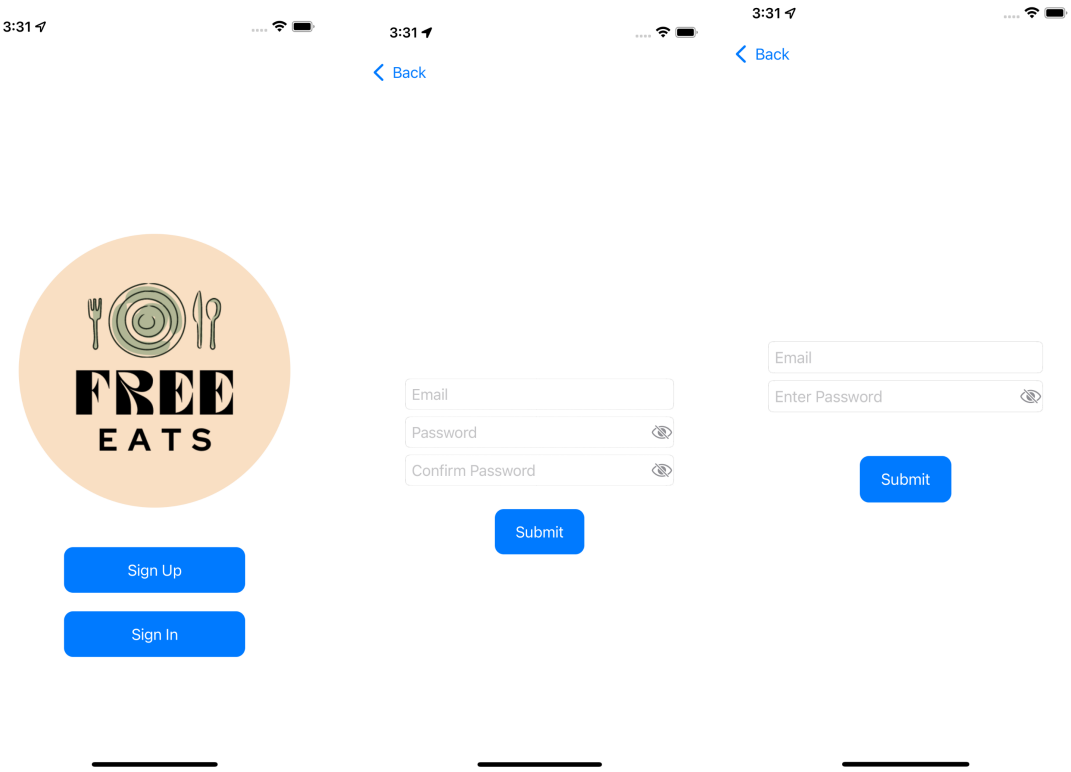


*Figure 1 – 3.* App Initialization and Login Flow

Once the user has successfully created an account, the user is then led to a registration screen, where they fill in their information, such as creating a username, adding their names and selecting their favorite types of food as seen in Figure 4.

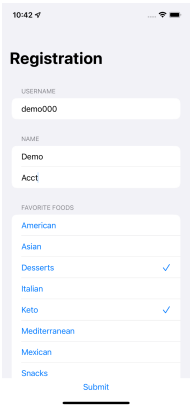


*Figure 4.* Registration Flow

Once users have successfully logged in, they are redirected to the Home View, where they have two options. They can find or submit alerts for free food or they take additional actions with their account such as signing out of their account. As below in Figure 5, the tab Find Eats details the user submitted available eats and where they are located. They can also click each submission to find out more details such as who submitted the alert, how much food is available and additional info, provided by the submitter, if available.

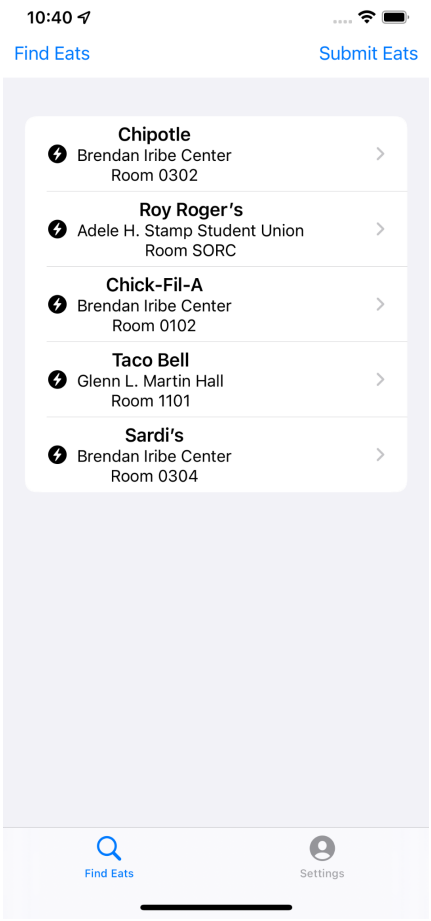


*Figure 5.* Find Eats

When they select the Find Eats buttons, ideally they can use the map to find locations nearby them where there is food available. When users select the Submit Eats button, they are

led to a form (in Figure 6) where they can provide more details about the food that's available for the taking.

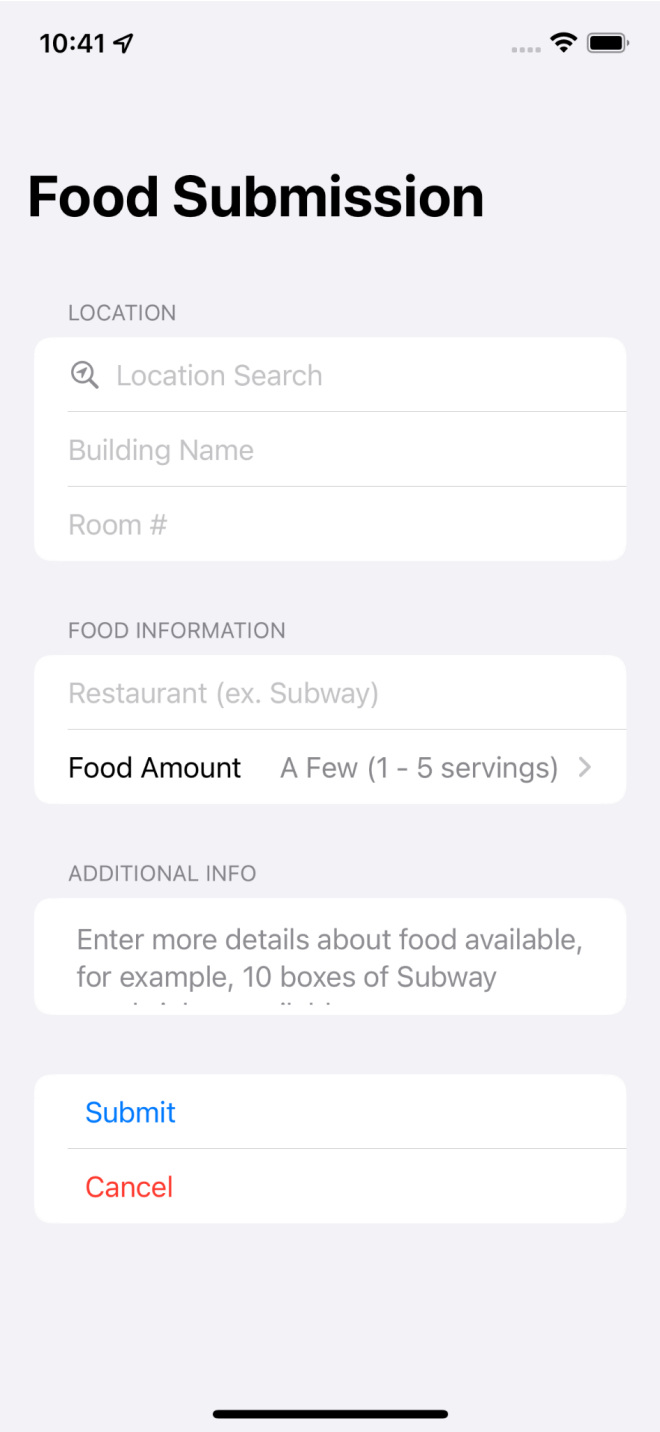


*Figure 6.* Food Submission

After discussing how Free Eats operates, now we will detail some of the APIs we used to build the application. Firebase provides developers with a multitude of options that assist in their development needs. Developers can choose between Firestore - a new real-time NoSQL database - or Firebase's traditional real-time database. It is likely that since Firestore was created not too long ago, there are less available resources and tutorials on its use available online compared to Firebase real-time database. This posed somewhat of an issue at times, which we will describe in the coming paragraphs. We opted to go with Firestore over Firebase real-time database because Google documentation encourages developers who are working on new projects to do so. In addition, we wanted to work with the latest and greatest technology wherever possible. We found that it was quite easy to get Firestore up and running. Within a matter of minutes, we were able to get it going with mock data.

The language used to describe the organization of data in Firestore differs greatly from traditional database systems. Firestore works off the assumption of two core components: Documents and Collections. A document is akin to a table in traditional databases. It may have several user-defined fields where each field has an associated value. Each document has an identifier which can be used to look-up the document directly if desired. Values in a document can be a multitude of different data types such as String, Integer, Array, etc. The ability to store data structures such as arrays allows for a ton of flexibility when designing a database. Documents reside in collections. A collection is simply several documents grouped together. A collection has a user-defined name.

An example of how we used Firestore can be seen in our organization of user data. Each time a new user registers their information, a document is generated specific to that user and is identified by their unique uid which is conveniently provided by Firebase authentication. A user's details include String fields for their username, first name, last name, etc. All of the documents corresponding to the users registered are then stored in a collection named users. The "users" collection can then be easily queried and filtered with typical "where" filters to narrow down the results to specific documents.

As we built Free Eats, we encountered several issues and challenges while developing. Firstly, we weren't able to meet one of our goals of providing notifications due to Apple limitations. Notifications require that we have access to the Apple Developer program and due to the scope of the class we decided to forgo creating that feature. Additionally, there were some hiccups in regards to the map feature. Unfortunately, it cannot show the locations of submissions on campus, despite multiple attempts to fix the issue. Since the Firebase API relies mostly on asynchronous functions, it was a challenge to get these functions to work properly to provide updates from the Firebase server to the app. As a result, whenever Free Eats is initialized, the Find Eats tab is unresponsive unless you click off to the Account tab and go back.

Our intention is to continue building upon the foundation laid down for this app in the future. As mentioned previously, we were not able to accomplish everything that we hoped to. There were several stretch goals we did not have time to implement but we plan to do so in the future. One of the features we hope to flesh out in the future is the process of claiming food events. The idea is that a user should be able to select any of the food events viewable to them

in their food feed of their find tab and then press a button which engages them in a “food claim” workflow. In simple terms, clicking the “Go!” button on a food event would bring up a map with an annotated line connecting the user’s location and the location of the food event, accounting for the fastest time to travel. The user would then be able to go into a navigation mode that directs them to the location of the food event and provides real-time directions in a fashion similar to the walking mode of Google Maps. This begs the question of whether or not it is possible to integrate Apple Maps walking directional mode into our app which is something we will have to look into.

To complete the “food claim” workflow that we described in the previous paragraph, we’d likely have a mechanism which allows food-goers to confirm with the app that they did in fact claim the event once they arrive there. Perhaps this could be implemented with some sort of scanning feature involving both parties - the food-goer and the people running the food-event. One thought might be to simply allow food-claimers to press a button in that app that says “Claim” and reduces the quantity of food reported for that event. This could work in a perfect world, but a trust-based system might not be optimal as it could be subject to abuse.

Another feature that we hope to implement would be a scoring system. Users would have incentive to submit food events because they would get points for doing so, assuming they submitted a valid event that is actually occurring. Every user starts with zero points when they register for the app and they can earn points either by submitting food events or by claiming them. These points would be stored in Firestore in the user’s associated document in the “users” collection. The scoring system would provide a means to implement a leaderboard

system. In the future, we would like to add a tabbed view for a leaderboard which provides visual feedback on who the top user contributors to the app are. The leaderboard could also allow users to compare their scores with fellow classmates and friends.

One final feature we'd like to mention the future integration of would be socialization/personalization. We would like to implement the ability for users to upload a profile picture viewable to them and all others that use the app. When a logged in user views a food event, they should be able to click on the reporter's name in the FoodEventView and then be brought to a page that describes details about the reporter's user profile. Ideally, the logged in user would be able to chat with the reporter and follow them to see their activity on the app on their personalized activity feed. The activity feed would be another tabbed view which allows for facebook-esque posts where users that a logged in user follows may describe their experience at a food event. Socialization and personalization would provide an optimal user experience and make the app feel fun rather than just having utility.

We predict that this app can help reduce food insecurity on campus by up to 70% because organizations have a lot of events on campus where a lot of food is being wasted. Using this modern day technology app, it makes it easier for students or organizations to communicate where and what food is available to be given out for free. With this being done, students will be able to improve their health as well as concentration in classes which can help aid their school performance as well be a benefit to the school by increasing the schools passing grade rate.