CS 4720 - F17 - Final Project Documentation

Device Name: Google Pixel
Platform: Android
Name: Cristopher Serrano
Computing ID: cas5tu
Name: David Clark
Computing ID: dc7hk
App Name: Bite Builder

Project Description:
Our app, Bite Builder, will provide a place to organize what a user will eat along with monitor what a user should buy in the grocery store in order to make the meals. Essentially, it is like a virtual cookbook. The app will have meal recipes that are what a user should shopped for at the grocery store so a user utilizes all the ingredients and does not have to let them go to waste. Along with planning your meals, it will give you the location of the nearest grocery store and creates a user account.

What we propose to do is create an app that will do the following:

- The system shall allow a user to make up a meal plan in order to organize themselves on what to buy for ingredients and what to eat
- The system shall allow a user to tell you what food you should buy in the grocery store
- The system shall allow a user to find the nearest grocery store around them
- The system shall allow a user to select recipes to cook and also deselect them
- The system shall allow a user to create an account with their personalized information

We plan to incorporate the following features:

- GPS / Location-awareness – to look for the nearest grocery store around the user.
- We will use SQLite as our backend to store the meals selected
- We will use Firebase to hold our meal recipes
- Data storage using key/value pair storage - We will store the user's id and their password for easy log in

Wireframe Description:
Our wireframe shows the basic layout we are envisioning for our app. After the launch screen appears, the user will start creating their own profile through a series of different activities asking them questions about what they want their diet to be with the recipes. Once the profile is created, each activity within the app will be blank at first and hold information on what each screen does along with point to where the user needs to go to begin creating their meal plan. Users, in order, to get started click the floating action button. Once they have clicked on the floating action button, they will have a selection of meals to choose from. Once the meals are chosen, the user will be directed back into the meal plan and each activity will be populated

with the information corresponding with the meals chosen. In the grocery activity page, there will be a list of ingredients need to make the meals. The GPS will direct users to the nears grocery store and then the sign out activity confirms your sign out.

Platform Justification:

When deciding which platform to use, the main consideration was accessibility. This app was designed to be quickly used after it was built. With Android we had the opportunity to download the app and deploy it on our phones quickly. Furthermore, we wanted this app to be aimed for massive adoption so having the platform be open source was more advantageous. Personal reasons, the app is in android focused on the code being written in java were the learning curve was much less compared to Objective-C.

Major Features/Screens:

The sign in screen is the first screen a user sees when entering the app. It takes in a username and password to allow a previous user to login back in. We set up SharedPreferences for it and it prepopulates the two edit text fields. The app then verifies the user's credentials with the firebase to verify if the user exits if not then it gives a toast message saying there is no user. The sign in also has the new user button to allow a user to create an account.

The next screen, which is the diet selection screen, allows the users to select a diet in order to know which meal plans to populate for them to choose from. The selected diet is then put in the firebase database for that user later on when the user creates their account.

The following screen is the allergy screen, which allows the users to select if they have an allergy to any food items. The selected allergies are then put in the firebase database for that user later on when the user creates their account.

The create account finally takes in what the user wants to be their username and password. It confirms the user typed in their password correctly if not then they cannot go through. It receives the previews data the user up in to create a profile which is then stored in the firebase. Then the user is finally directed in the app.

The meal plan activity is the first active the user sees when they have an account. It displays the meals a user added. It has an add button to add meals, which leads to the add meal activity, along with a reset button to clear all the meals. The meals are populated in the activity by the SQLite database and when you click on a meal it takes you to the actual meal activity with the meal's information such as the title, ingredients, and a large image.

In the add meal activity, it displays all the meals a user can chose from that are stored in the Firebase database. If a meal is selected it can also be deselected, but the meals that are selected are stored in the SQLite database so a user can still see their meals when they do not have internet. Finally, it has a done button on top to finish up the selection and go back to the other activities.

The grocery list activity, displays all the ingredients a user needs to buy in order to make the meals. It pulls the ingredients from the SQLite database.

The store picker activity, uses a GPS to track your location and find the nearest grocery store around you. It does this through a google API.

The logout confirms a user wants to log out by displaying a logout button that will redirect them to the sign in screen.

Optional Features:

The optional features that this app has are the web service using a third-party platform(Firebase), data storage using SQLite, data storage using key/value pair storage in SharedPreferences, and finally the GPS/ Location awareness.

The Firebase was used to create a backend data storage for all the meals the app has along with the user information. The Firebase also allows the user to login in by checking the username and password stored in it. It then connects to the SQLite to transfer all the meals the user has select to make. It can be tested/demoed by clicking add a meal and seeing all the meals prepopulated along with just login in after creating an account, to verify an account the app logins into Firebases and sees if the username and password are in there.

The SQLite then stores all the information the user has selected from the meals. It populates the grocery list ingredients, meals on the meal plan activity and the meal details if a user clicks on a meal. It was used so the if the user after selecting their meals does not have internet they can still use the app and have all their information on it. In order to test the SQLite, a user can see if their meals which were selected are displayed in the meal plan activity or they can see if ingredients are displayed in the grocery list activity. Also, if a user deselects a meal, the meal will no longer be in the SQLite.

The SharedPreferences storage system was used to allow for quick access in the app after a user creates an account. It is on the start activity when entering the app and in the create account activity to store the username and password created. In order to test, a user can create an account and once in the main part of the app, they can log out and then they will see in the start activity that their username and password are in the text fields ready for them to just press sign in.

Testing Methodologies:

The testing methodologies done were functionality tests where both partners went through all the functionalities of the app and made sure they worked, such as database checks, running through test, printing log statements. Some of the functionality tests done are described in the optional features section under each feature mentioned.

Another testing methodology was usability tests where a person, in this case a friend, was asked perform a functionality. The friend was asked to create an account and select the

meals they wanted to eat for the week and look up the ingredients they need to buy. This test revealed if our user interface was clear and easy to use.

The last testing methodology used was a compatibility test to make sure our app worked on different device versions. We used the Google Pixel emulator along with a Samsung Galaxy Tablet.

Lessons Learned:

One of the biggest lessons learned from this project was how to really operate Firebase. It has so much power and a good database, but specifically we learned how to insert into the Firebase database along with how to receive data from it that had many fields to it. Furthermore, we learned how to build a Firebase database and how to use Firebase cloud storage. In SQLite, we learned how to query the database with parameters and how to structure a database in SQLite. We also learned how to transfer data from Firebase to SQLite.

In this project, we saw our java skills further improve and there was a learning curve at first. We learned that we needed to set up global variables so that the data was saved throughout methods and carried over through all the methods we needed to have it pass through. Specifically, this was important for our Firebase reference where we looked at the Firebase to verify if the user had typed in the correct username and password. Another component we learned about in Java was the different of how to instantiate arraylist and array. This came into play when we had to package an array of information from one activity to another activity and we did this through an array as they had that option in putExtra method of intents.

Finally, we also learned from is the process of designing an application and how it might chance as development proceeds. We had to add more functionality and activities in the middle of the project to get the whole interface to connect better and more efficiently.