CS 4720 - F17 - Final Project Proposal

Device Name: Opal	Platform: iOS
Name: Jessica Ewing	Computing ID: jme4xg
Name: Chris Lee	Computing ID: csl7dk

App Name: Food Finder

Project Description:

Our app, Food Finder, will provide a way for students to get live-feed updates about the wait-time of popular eating locations on grounds. Far too many times, students have made a 20 minute trek to eat dumplings at the Got Dumplings cart, only to be faced with a line that curves around the amphitheater. With this app, students can look up specific food places on grounds and see the current estimated wait time for the business. This wait time is the most recent time inputted by a user on grounds. This means wait time displayed is always the most current update of the wait time at that location. They can also see the route they can walk from their current location to get to the eatery selected. There is an easily accessible link to pull up the restaurant's menu while they walk, and there is also a link for any app user to input wait times at whatever location they choose. Every time they submit a wait estimation, they receive 5 points on their "score," which they can check on their profile page. When the student reaches a 500 point level, they can show the points with a student ID at one of the restaurants to get a free drink! The restaurant will record the student ID number to enforce students only using this deal at point intervals of 500. Overall, this app will be a way for students to optimize the time they spend getting food they love from around grounds.

We created an app that delivers the following:

- The system shall allow a student to create a profile picture using pictures/camera.
- The system shall allow a student to see their currently updated points on their profile page and allow them to change their name to reflect their student ID name.
- The system shall allow a student to click on a given location on grounds and see the estimated wait time.
- The system shall allow a student to click on a link and pull up the location's menu.
- The system shall allow a student to input a wait time for a given location, and this wait time will be updated to the most recent wait time for all phones running the application.
- The system shall allow a student to see their location, the location of their chosen eatery, and the route that can be taken between them.

We have incorporated the following features:

- GPS - We use the GPS to find the user's current location, send it between views and use it in our Google Map APIs
- Consume a pre-built web service - We used APIs from Google Maps and Google Directions to create and display a route from the user's current location to a specified restaurant location
- Camera - A user can take a picture to customize their profile, and the picture will save on the profile page
- Build and consume your own web service using a third-party platform - We used Firebase Database to store the most recently submitted wait time. The app pulls from this database to display the most recent time, and overwrites the database when a time is submitted.
- Data Storage (Core Data) - Core Data is used to store the user's current points. We did not have a login because we assumed there would be one app per user on the phone. The amount of points a user has is saved in core data and updated every time they submit a wait time.

Wireframe Description (a lot has changed since our initial plan):

In our initial wireframe, we started with a login page. We decided not to have a login because we thought each user could download the app to their phone, and the points could just be stored with Core Data. The login page would have branched to a sign-in and sign-up page. The home page contains a profile button, the user's location, the time to class switch, and scrollable view of all 7 restaurants we decided to include. We are using a scrollable view because we want the restaurant buttons to be large and noticeable for the user to press, so we didn't want to squeeze them into one view. If the user presses the profile, the view controller has the saved profile picture, the user's point total, and the user's name which can be changed. The user can set a picture by tapping on the image view or tapping on the camera button. An about button leads to a view controller that describes the app. Back from the home view, if you press on a restaurant button, the next view contains a small map with directions from the user's location to the location of the restaurant. It also contains the wait time and the options for a user to look at the restaurant menu or upload a wait time. The link for the menu opens safari and brings the user to the menu website. The button to submit a wait takes the user to the submit wait view. This view has a picker for different restaurants, which we thought would be useful if the user was in a location such as the PAV (so they can submit more than one location wait time at once). It also has an input for the wait time estimation, and it will only accept numbers. This prevents the app from crashing when something other than a number tries to write to Firebase.

Platform Justification:

One of the biggest benefits to using iOS is that we can see the overall layout of our app on the main storyboard. It is really helpful to see how each view connects to another view, and how the app works together with all of its components. Also, it is really easy to connect view controllers using iOS, and it is easy to see how things interact with each other. Another benefit to iOS is that iOS is coded in Swift. Although at first we were not a fan of swift (due to being more familiar with Java), Swift is very straightforward, and the methods are incredibly easy to read. As partners, we were able to work separately and easily understand what the other was doing based solely on the method names and parameters. Another benefit to iOS is that designing the app layout is easier and looks a lot better. The components are more visually based, and the constraints are straightforward to add compared to Android where you need to manually code views. Finally, both of us have iPhones, so it was really awesome to be able to create an app and download it to a platform we both use. We also have more familiarity with the design of many iPhone apps, so we were able to think about design ideas with previous app interaction on iOS.

Major Features/Screens:

1. Home Screen → On this screen, a user can see their location in the form of coordinates at the top of the screen. This is for aesthetic purposes and is not the use for GPS. There is a minute to class switch label that tells the user the number of minutes they have before a UVA class usually ends, where walking traffic will be heightened and lines may be longer. We decided to have this switch every 15 minutes (a majority of class ending times fall on the 15 minute mark), as well as the 50 minute mark. During times earlier than 8 and later than 9, the label displays that no classes are in session. There is a scrollable view of 7 different locations the user can click on, which will bring them to a new view.
2. Location Screen → There is a different location screen for each of the 7 restaurants, but they all look the same. They all have a map view that presents the user's location, the location of the chosen restaurant, and the route that a user can take to move between the two destinations. There is also an estimated wait time, which is comprised of the most recently updated wait time submitted by users. There is an online menu button that takes the user to a safari page with the menu up. There is also a submit wait button that takes the user to the submit wait view.
3. Submit Wait Screen → On this screen, there is a picker wheel of all the restaurants that the user can choose, and there is an input box where the user can enter a wait time to submit. If the user enters nothing or something that is not a number, a dialogue box will pop up telling them to input a number. When they press submit, a box pops up confirming their wait has been submitted, and (1) their point total on the profile will increment by 5 and (2) the wait on the restaurant they chose to submit a wait for will update.
4. Profile Screen → On this screen, a user can choose a profile picture, set their profile name, and check to see their current point total. The profile picture will save, so it will still be there when the user re-opens the app. The current points are taken from core data and will also not change unless the user updates a wait time. Tap on the picture to pick a new profile picture from camera roll or click the camera button to use the camera.

Optional Features:

1. 15 points: GPS → We use the GPS to find the user's location current location, send it between views and use it in our Google Map APIs. You can see the location coordinates that we get at the top of the home screen. You can test to make sure this feature is working by clicking on one of the restaurant pictures and looking at the map. If there is a point where you are (a point at your current GPS location), then the GPS has gotten your location and translated it to a point on the map, so GPS is working.
2. 10 points: Consume a pre-built web service - We used APIs from Google Maps and Google Directions to create and display a route from the user's current location to a specified restaurant location. You can test it by clicking on a restaurant picture and looking at the map on the screen. If there is a correct GPS coordinate where the eatery is, and if there is a route appearing between the two points, then the API router is being called as well as the method to create the pointers. Thus, the pre-build web service is correctly being called.
3. 15 points: Build and consume your own web service using a third-party platform - We used Firebase Database to store the most recently submitted wait time. The app pulls from this database to display the most recent time, and overwrites the database when a time is submitted. To test this, click on a restaurant button and look at the wait time. Go to the submit wait screen (from the submit wait button) and submit a wait time for that restaurant. Go back to the previous screen. If the wait time is updated to the most recently submitted time, then that time was correctly written and read from the Firebase database.
4. 20 points: Data Storage (Core Data) - Core Data is used to store the user's current points. We did not have a login because we assumed there would be one app per user on the phone. The amount of points a user has is saved in core data and updated every time they submit a wait time. You can test this by going to the profile page and looking at the point total. Then, go back to the home page and click on a restaurant button, and then click on submit wait. Submit a proper wait time, and go back to profile page. If the point total has updated by adding 5, it has been properly stored and read from core data.
5. EXTRA: 15 points: Camera - A user can take a picture to customize their profile, and the picture will save on the profile page. To test this, go to the profile page and take a picture with the camera (or click on the picture and upload the picture to the camera roll). The picture will be saved whether you close the app or go back to the home page.

Total points: 75 points. Sorry, we ended up going over to include all of the features we wanted. If you choose one not to grade, please do not grade the Camera.

Testing Methodologies:

To test the app, we ran it on the iPhone 5 simulator and on the iPod we rented. Here is what we did to test all of the features:

- On the home screen:
    - Is the minute to class switch correct based on the current time?
    - Is the location correct?
    - Can you scroll down the screen?
    - Can you click on all the pictures and pull up the correct views for each?
- Profile screen:
    - Can you click on the picture and pull up the camera roll?
    - Does the selected picture from the camera roll save in the photo view?
    - When you go back to home and back to the profile, does the picture stay the same/do the points stay the same?
- On each of the restaurant screens:
    - Is our location correct on the map?
    - Is the location of the restaurant correct on the map?
    - Is the route a valid route to walk to the location?
    - Does the menu button click to the proper website?
- On the submit wait screen:
    - Can you submit a blank box or a box with characters?
    - When you submit a wait time to each of the restaurants, is the wait time reflected back on the restaurant screen and are there 5 points added to your point total on the profile screen?

Usage:

- You need no additional information to run the app.
- Problem solving error → Just in case this comes up, we had difficulties getting the GoogleService-Info.plist to work from cloning our GitHub repository. If the GoogleService-Info.plist turns red and is unrecognizable, click on the GoogleService-Info.plist file in the left-hand pane. In the right pane, click on the paper button. You can see the 'full path' there. Above the full path, you can click on the file icon and specify the location where the plist file is located (which should just be the app folder). This should fix the issue. For some reason, it was specifying that it needed to be in downloads. We believe we fixed it, but just in case!

Lessons Learned:

- Screens are tedious, and we should think about how we can combine screens before we start making views. While we were making our app, we made a design decision to limit how many eateries to have in our app. We left off probably 5 or 6 places on Grounds that students could eat at, mainly because the amount of work to add the 3 extra screens for each establishment would have eaten into our time. While the code and layout of each extra place would be nearly identical to the the ones that were already made, it would have been a hassle and taken away time to finish our other features. Thinking back, we probably could have passed data from each button to only use one view controller as compared to 7. We used a single view controller for the submit wait button.
- Working with Firebase is not as difficult as we anticipated. There are a lot of great tutorials online, and the user interface is very friendly. It connects seamlessly with the app, and it was no hassle to implement. We will definitely be using it if we need to make a serious app in the future.
- Google APIs are fun to work with, but they can quickly get complicated. While getting a map to appear from the API and setting the points was easier, creating the route, attempting to get a route ETA from directions, making the map focus in on our specific route, and even just making the map smaller to fit into a screen-sized square quickly got difficult. We would need more time to fully understand the breadth of implementation we could achieve with the APIs.
- It is important to think about the space we are working with. We originally thought about putting all of the restaurant buttons on one view controller without a scroll, but after looking at the small screen of our practice iPod, we realized that we would need a scroll so there were large buttons the user could press on.
- Even though we were worried about how professional our app would look, we are so proud of our app. The amount of work we put in and the fact it does what we want it to do is more rewarding than we imagined.

Extra Notes:

- The free drink based on 500 points would heavily rely on the honor system, as we understand there are ways people could cheat the system in order to get their friends or themselves free drinks. However, we thought the student ID idea would at least deter a majority of people hoping to cheat the system.
- Using the most currently submitted wait time is dangerous, because if one user submits an outrageous number, there is nothing to skew the result. However, this is also running on the honor system of UVA students, and if one student decides to input a crazy number, another student may correct it quickly to maintain the accuracy of the app (we hope). The whole point of downloading it is to get an accurate read of restaurant wait times.
- Thank you for a great semester!!!