CS 4720 - F17 - Final Project Documentation

Device Name: Cortland Platform: iOS

Name: Michael Chang Computing ID: mhc6kp

App Name: UVA Dining

Project Description:

The UVA Dining App is an information hub for people who are unfamiliar with UVA's dining experience (i.e. mostly first years and late transfers). The list of restaurants that will be listed will be limited to whatever is on-grounds or is eligible for the use of meal exchange swipes. It will provide information on how many meal swipes and plus dollars you have left, today's menu for the three dining halls. When an individual restaurant is selected, it will display their menu items, its operating hours, and its meal exchange active hours. There will also be a part of the app that allows you to see the meal exchange restaurants in order of promixity to your current location. There is also an random meal exchange restaurant chooser, for those who are unable to make up their minds on where to go, or what to get.

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

- The system shall allow a student to login in through Netbadge to and extract their current balance of plus dollars, CavAdvantage and meal exchange swipes. The user will have to log in to Netbadge every time they want to update the status with the pin they defined, but the app will use UserDefaults to store and load its status to its latest update and display the date of the latest update.
- The system shall display the statuses of the three main dining halls to see if they are currently open or not in the main screen.
- The system shall dynamically read from CampusDish to extract the current menu of each dining hall.
- The system shall also have a list of the closest meal exchange restaurants to your current location if you are just looking for a quick bite to eat.

- The system shall give the option of randomization: if you aren't sure of what you want to eat in a particular moment in time in terms of a restaurant or a restaurant's specific menu item, there will be an option to shake the device to select a restaurant or a menu item for you.
- The system will utilize the Firebase webservice to provide information on the non-dining hall menu items and operating hours, with information on times of when meal exchange would be active for each respective restaurant.

I plan to incorporate the following features:

- GPS/Location Awareness – The app will get the current location of the user and will reload the current list of restaurants in the order of proximity to your current location
- Device Shake – The app will randomly select a restaurant or menu item for you depending on whether the restaurant was confirmed or not.
- Consume a pre-built web service – The app will parse through the Netbadge accessible web service that allows you to display the user's current balance of plus dollars and swipes. It will also parse through Campus Dish's website to extract today's dining hall menu and the Hours of Operation for each Dining Hall.
- Data storage using key/value pair storage – The app will store the student's computing ID and Netbadge password and will utilize Keychain to create a pin for that login information.
- Build and consume your own web service using a third-party platform – Use Firebase to get data on non-dining hall menu items and restaurants (list of menu items, the times at which meal exchange would be active, and its current operating hours).

Wireframe Description:

After the launch screen appears, there are labels displaying the user's current balance of plus dollars and meal swipes, and 8 screens to possibly go to. Under the current balance box, there is a reload icon, which, when pressed, updates to user's balance by logging into Netbadge and extracting the data from the respective webservice. It is required to have your login information saved if you want to see your plus dollars and meal swipe balance. Users have the choice to have

their login information saved, along with a pin when loading that information with the update button. Under each dining hall screen, it lists if the respective dining hall is currently open, what day it is, and what time the dining hall operates on that certain day. It also lists today's menu of that dining hall, and the various entrees they might serve throughout the day. When tapping on the meal exchange button, it returns a list of relevant restaurants in a tableview. When an individual cell is tapped on, it moves on to another screen and pulls up the list of relevant menu items in another tableview and information about the restaurant, including what times the meal exchanges are available. When tapping on the "Restaurants near me" button, it returns a list of relevant restaurants in a tableview, which has an "update" button on the right of the navigation bar. What this "update" button does is that it grabs the user's current location, calculates the distance between all the meal exchange restaurants and your current location, a returns the list of restaurants in order of proximity to your current location and the distance. Note that you will not be able to update again until you reload the view controller. The last screen is for people who are indecisive on what they want to get. The user will shake to choose a restaurant, and, if they decide they want to go there, they have to option to confirm the restaurant that was given to them. The user can shake the phone to also find a new random restaurant if they haven't confirmed them. They can also shake the phone to pick a random item from the restaurant they have confirmed, or they can go to the restaurant's page to look at the menu with the button that says "Go to Restaurant Page".

Platform Justification:

A major benefit that I found while using iOS instead of android is that there are already pre-defined functions for many of the features, which made it much easier to implement this app. For this app specifically, I think that shake would've been much harder to implement for android, whereas for iOS, it was trivial and only a few lines of code. There was also the function of calculating distances using latitude and longitude, to which iOS had a function available for that. iOS also allows you to implement Keychain into your app, which allowed authentication if a user wanted to access certain information securely. Apple is also known for their simplistic design and focus on aesthetics, so my thought process was that it would be easier to design the UI with iOS instead of android, and that it would look better overall. Apple also doesn't have variance on what devices are being used, since they are the only manufacturer that uses iOS. This means I

would only have to factor in the sizes of the models that Apple has made, instead of all the possible sizes of each and every brand that you find using Android OS.

Major Features/Screens:

HomeView Screen – The Landing page of the app. On the right of the navigation bar is a button that allows you to login into netbadge and save that login information with a pin that you define. After saving this login information, it enables you to update your current balance of plus dollars and meal swipes if you input the previously defined pin. On reloading the app, it will save the balances from the previous update and the date and time that it was updated. Next, we have the three dining Hall icons, which will take you to screens that have relevant information on each respective dining hall, and their menu items. Then we have the "I'm Feeling Lucky" Button, which will take you to a screen that randomly displays a restaurant for you when you shake the device. The "Restaurants Near Me" button takes you to a screen that has a list of all the meal exchange eligible restaurants and gives you the option to see which ones are closest to you. The "Meal Exchange" icon allows you to select restaurants and see what menu items they have that are meal swipe eligible.

LoginView – To access this screen, you have to press the "Login" button on the right of the navigation bar. You then put in your netbadge computing ID, your netbadge password, and a user-defined pin. This first checks to see if the pin was a valid 4 digit code, and then checks if the Netbadge Login is valid. If it is valid, the LoginView will display an alert box saying that the login was successful, and then it will close the LoginView automatically. After the user information has been saved, the LoginView cannot be accessed again, because there really isn't a need for it anymore.

BalanceView – To access this screen, you must enter the pin that you previously defined in the LoginView. After you successfully enter your pin, it loads a webView that automatically logins you into Netbadge, and then takes you to the UVA webservice with your current plus dollar and meal swipe balances. It parses through that webView, and saves the current balances to the labels on your HomeView page and UserDefaults, so when you reload the app, it'll show the most recently updated values for your balances. After the webView gets to the balances page, it'll automatically close for you and go back to the HomeView. However, there is the possibility that it won't reach the balances page due to the asynchronous nature of these web requests. There

is the possibility of the App crashing if the webpage does not successfully load because it'll be searching for an HTML element that isn't there.

DiningHallViews – These Views are very similar to one another, so I'll explain how they work only once. The view displays the current status of the respective dining hall and displays the current day and the operating hours of that respective day. The current status is created by seeing if the current time is within the range of the operating hours of that dining hall. The DiningHallView also displays the menu items that are being served on the current day. However, these items default on the first type of meal period. There is a selector that allows the user to choose what meal period that they want to see on the current day. In other words, selecting "Dinner", will show you the menu items that are being served during dinner, and the default will be "Breakfast", which will show the menu items that are being served during dinner.

LuckyView – This view displays a random meal exchange restaurant on shake and allows you to go to the respective restaurant page. If the current restaurant is confirmed, subsequent shakes will display a random menu item that is meal exchange eligible from that restaurant. There is also a button to cancel the confirmation, which will remove the menu item currently being displayed, and allow the user to have another random restaurant being displayed.

MealExchangeView – This is a tableView of all the meal exchange eligible restaurants. When a cell is tapped, it takes you to the relevant RestaurantView.

CloseView – This is very similar to the MealExchangeView, except that it allows you to view the restaurants that are closest to your current location. To do this, you press on the "Update" button on the right of the navigation bar; this first asks you if it is okay to use Location Services on this app. You must then tap on the "Update" Button again; this takes in your current GPS coordinates, and goes through the latitudes and longitudes of each of the meal exchange eligible restaurants to calculate the distance, puts the restaurantView in order of proximity to the current location, and reloads the tableView with the distances. Note that this can only be updated once, and subsequent updates must be done by first closing the CloseView and reopening it.

RestaurantView – Very similar to the DiningHallView. Displays the operating hours, the current day, and meal swipe hours. It also displays the meal exchange eligible menu items.

Optional Features:

- GPS/Location Awareness **(15 points)** – The app will get the current location of the user and will reload the current list of restaurants in the order of proximity to your current location. It does this by getting the current location of the user and then calculating the distances using the latitudes and longitudes of each respective restaurant. It then reorders the list of restaurants based on the distances from the current location.
- Device Shake **(10 points)** – The app will randomly select a restaurant or menu item for you depending on whether the restaurant was confirmed or not. It first displays a random restaurant on shake, and on subsequent shakes displays another random restaurant. The user has the option of confirming this restaurant, and if confirmed, on subsequent shakes, it displays random menu items. When cancelling, it clears the current random menu item, and will display another random restaurant on subsequent shakes.
- Consume a pre-built web service **(10 points)** – The app will parse through the Netbadge accessible web service that allows you to display the user's current balance of plus dollars and swipes. It will also parse through Campus Dish's website to extract today's dining hall menu and the Hours of Operation for each Dining Hall. Note that due to the asynchronous nature of web requests, this may not always work because some searches for certain elements, and if those elements are not there it will crash.
- Data storage using key/value pair storage (UserDefaults) **(10 points)** – The app will store the student's computing ID and Netbadge password and will utilize Keychain to create a pin for that login information. The app also stores the most recent balances and loads it in the HomeView when the app is loaded.
- Build and consume your own web service using a third-party platform **(15 points)** – Uses Firebase to get data on non-dining hall menu items and restaurants (list of menu items, the times at which meal exchange would be active, and its current operating hours).

15 + 10 + 10 + 10 + 15 = 60 points

Testing Methodologies:

The app was mostly tested on the simulator to test every feature before going to the next feature. After I was finished working on every feature (except the GPS feature), I started to test the app on the device itself. Due to asynchronous nature of web requests, throughout testing the balance

updater I noticed that I would have to have delays on the code, because the time it would take to load a webpage would be variable. I also went around testing if the restaurants near me feature actually worked by walking all around grounds and seeing if the distance values actually made sense. I noticed that the things that require web requests might not always work or may even crash because an element isn't there or the web page that was supposed to be fully loaded after the delay wasn't finished loading.

Usage:

Some assumptions:

The user has a good internet connection

The user has a valid Netbadge Login

The webpages successfully load within the webview of the update scene.

The user has a CavAdvantage balance.

The user has a plus dollars balance.

The user has a meal swipe balance.

If these assumptions aren't met, the app is likely to crash because grabbing the specific elements relies on the webpage successfully loading.

It takes a few seconds to load the firebase database (waits 3 seconds before loading on initial app loading), if it isn't loaded the MealExchangeView, CloseView, and LuckyView will display empty tables.

Loading the menu on the DiningHallViews will sometimes timeout after 8 seconds and won't display any items, but usually it loads successfully after 1-3 seconds

The user has Location Services enabled

Dependencies

AlamoFire

SwiftSoup

KeychainSwift

Firebase

Lessons Learned:

I learned that some web services, especially ones that haven't been updated to the most recent standards, won't play nice (I'm looking at you CampusDish) with parsers, so I ended up having to use regular expressions and parse through the webpages myself. I also learned about the asynchronous nature of web requests the hard way, because my app would keep crashing or would return an empty data set if the webpage wasn't completely done loading, because there wouldn't be the specific elements to grab if it wasn't finished loading. To solve this, I needed to add a delay to my code, which doesn't completely solve the issue because it still might not be long enough for the webpage to load. I also realized that not everything is going to be available online; my original intention was to have another screen for plus dollars restaurants and their respective menu items and their prices, but there weren't many resources available detailing what restaurants you could spend plus dollars on and the exact prices on the items you could spend plus dollars on. I thought database entry on firebase took the longest because there was a lot of room for clerical error, and there were also a lot of changes that I had to make. I really underestimated how much time it would take to figure out how to parse the CampusDish website What I learned was that you can't always have the features you want, whether it be a time constraint, or a space constraint, especially when it comes to mobile applications. I also learned that there aren't a lot of resources available for how to implement certain functions on Xcode. A lot of the resources were either outdated or didn't work as intended, but I definitely think the worst part about this was how some of the syntax was updated. I actually think Xcode was pretty good about correcting syntax, but there are definitely some improvements they can make about detecting errors on the main Storyboard with missing references and whatnot. Overall I thought this project really fun, yet really stressful, especially the last week when it started getting down to the final hour because I had to cut out many of the features that I intended to have on the app, such as the current status of the Meal Exchange eligibility, the current status of the restaurant, plus dollar restaurants, and a GPS view that would navigate to the location using the coordinates provided in the restaurant class.