Nate Ackermann, Nadia Asif, Polly Bainbridge, Calvin Lin, Sarem Rashid

Professor Densmore

EC327 Introduction to Software Engineering

December 11, 2017

**Walk With Wakki**

*Developing the Future of Personalized Health Care*

***Section 1: Motivation***

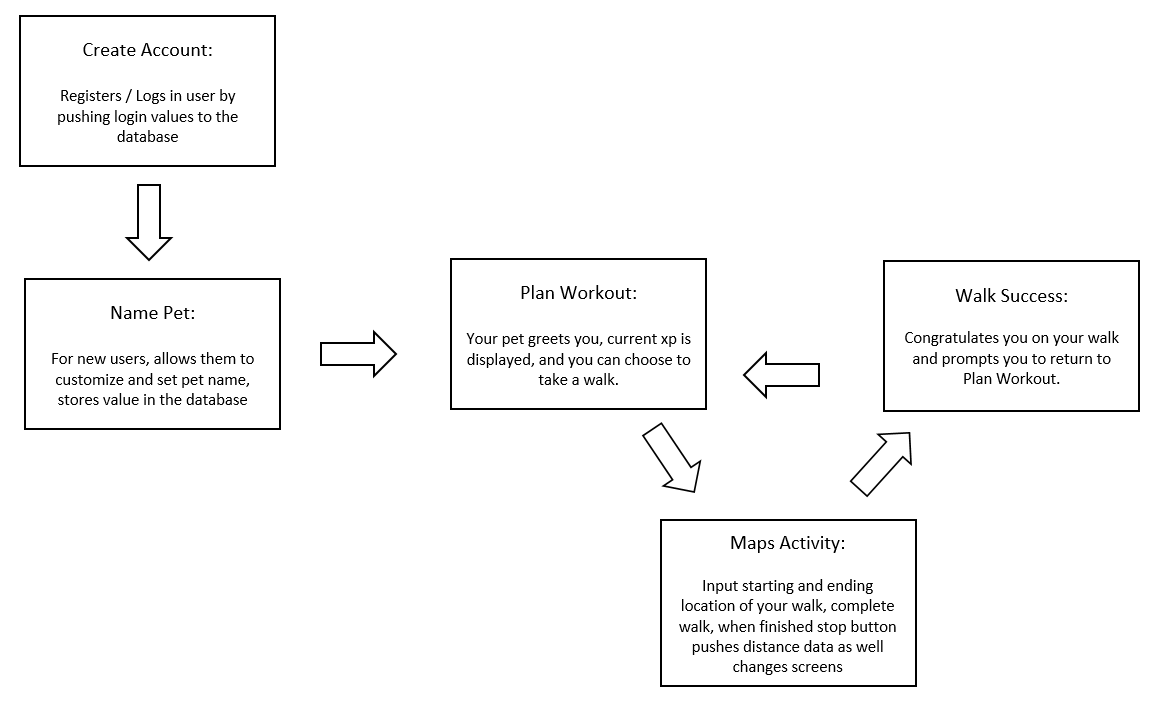
With advances in health care across the United States, the average lifespan of the general population has been steadily increasing. While longevity may seem like a benign win for society, it however leads to an unsuspecting, systemic healthcare crisis. It has been reported that “obesity rates among older adults have been increasing, standing at about 40 percent of 65-to-74-year-olds in 2009-2012”.[[1]](#footnote-0) Furthermore, within the next few years, the average age of the population is predicted to increase dramatically in size. It has been predicted that the number of Americans aged 65 and older is to “more than double from 46 million today to over 98 million by 2060, and the 65-and-older age group’s share of the total population will rise to nearly 24 percent from 15 percent”.[[2]](#footnote-1) In order for the United States to withstand this shift, as well as a higher proclivity for obesity, we must develop a way to ensure this population is improving their general health. The ideal method for changing an individual’s lifestyle is through the integration of small improvements in day-to-day life. With the goal to improve the general health and well-being of this increasing population, our group settled upon a Tamagotchi-inspired application that could be utilized to improve the quantity of exercise the user achieves each day. We were able to finalize the structure of our design approach and visualize “*Walk with Wakki*” - a Google Maps integrated application that encourages users to both walk greater distances in their day-to-day life and take care of their virtual pet, Wakki.

***Section 2: App Functionality and Metrics***

***Section 2.1: Overview***

Initially, *Walk with Wakki* was developed with the scope to allow the user to set a profile, including a username, age, weight, and daily exercise goal, as well as display a default "pet" page, where a simple picture/animation of the pet could be displayed (main page for the app), allow the user to start a "workout," to be a tracked walk/run via the Google Maps API. Additionally, we aimed to prompt the user to walk a set percentage longer after a certain period of time had passed. On the back end, this percent increase would cause the app to generate a longer route via Google Maps while on the front end the motivation for the longer walk would be the user's pet wanting to "play somewhere new" or "try a new food." Finally, in order to make the app function as a monitor for activity over a long period of time it should save the user's daily exercise data for the past week, month or year as well output notifications encouraging the user to return to the app if it has not been used for over a week. However, soon after beginning the process of developing the app we became aware that these specifications were too complicated to achieve in the given time period. For this reason we developed a decreased vision with smaller scope and correspondingly smaller metrics. Now the app will allow the user to set a profile, which includes a username, email and password, display a default "pet" page, where a picture of the pet is displayed and prompts the user to rename the “pet” as well as store this information within a database for later use. Once logged in the user is moved to a default plan workout screen where their pet says hello and score displayed (score is based on distances walked) from here they can choose to begin a workout. Once workout is started the user is linked to Maps API where they input location and distance and are given a route to walk. The final distance is recorded, stored in the database and used to calculate a score. This score will now function as the motivation for the user to walk longer distances more often.

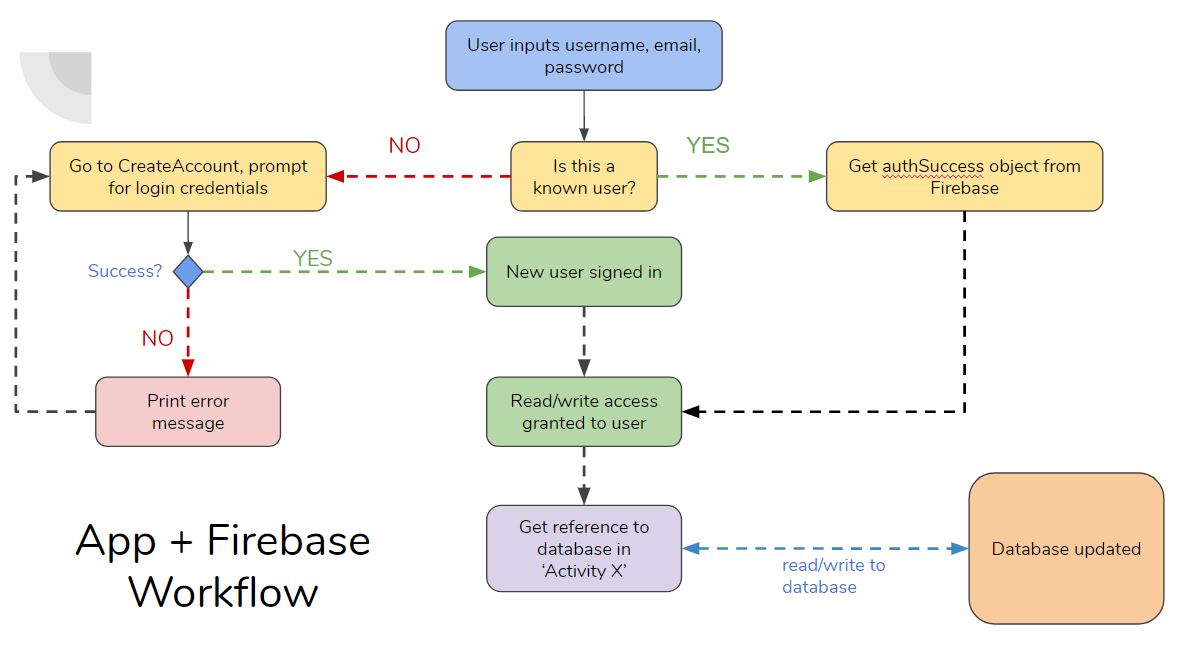
*Figure 1: Flow Diagram of App Functionality*



***Section 2.2: Back End Functionality***

***Section 2.2.1: Firebase***

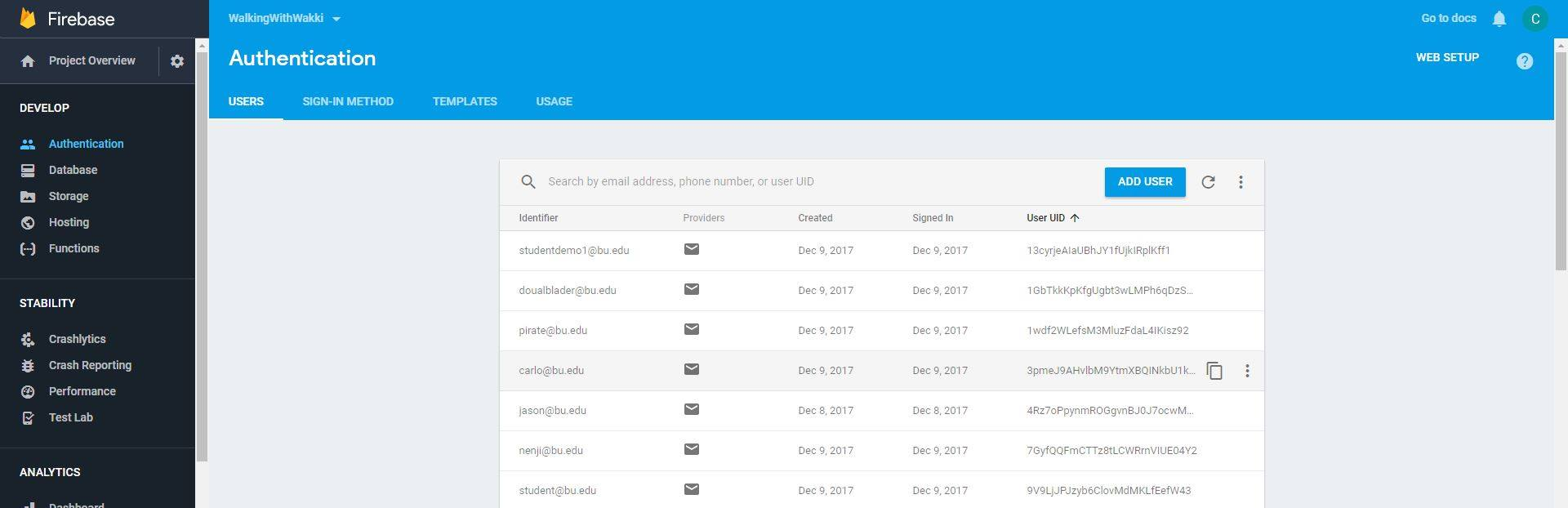
*Figure 2: Flow Diagram of Firebase Functionality*

******

When the user enters username, email, and password and clicks the “Create Account” button, a Firebase attempts to create a user with those credentials. If the user creation process is successful, an authentication value is set to ‘true,’ and the user is allowed to upload data to the database. A database entry is then created with the user’s username. Several other fields, “DistanceTraveled,” “PetName,” “XP,” “Email,” and “Password” are created as children of the username entry in the database.

When a new activity is started, the authentication status is checked, and if ‘true,’ the user’s username is retrieved and used to access fields within the database for each of the corresponding activities. In the same manner as before, values are retrieved and rewritten from the database for each individual activity.

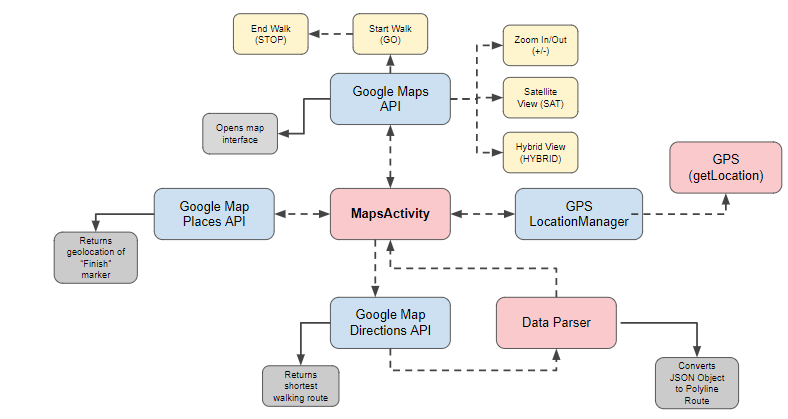
*Figure 3: User table generated in Firebase*



***Section 2.2.2: Maps Activity***

Walking is currently the only XP-boosting functionality available in the current app build. The diagram below depicts the flow of information within the different map coding sections: a central MapsActivity file draws information from multiple. Red denotes classes, blue denotes the APIs that MapsActivity utilizes in its implementation, yellow denotes a button, and gray denotes a function or return-value. The arrows indicate the relationship between the elements of the MapsActivity: the dotted lines denotes an implicit sense of relationship (ex: the class GPS is found in the GPS LocationManager API; The button for Satellite View implements a feature included in the Maps API), while the solid lines denote an explicit input/output relationship, depending on the direction of the arrows (i.e. the arrow points to the output).

*Figure 4: MapsActivity functionality with Maps API*



***Section 2.2.2.1: Maps API***

*Walk with Wakki* implements Google Maps API to launch an interactive maps interface with built-in pan and zoom gestures and the ability to toggle between a hybrid (HYBRID) and satellite (SAT) view of the map for easy map viewing. Additionally, the Maps API provides the the geo-coordinates (longitude and latitude) of any location that the user inputs. *Walk with Wakki* uses these coordinated to demarcate the “Start” position of the user’s route. This position is denoted by a blue marker in the app, which is wille be referred to as the “Start” Marker.

***Section 2.2.2.2: Places API***

The application makes use of the Google Places API for Android in order to find the geolocation of our “Finish” marker. As this API functions to build location-aware applications it is able to locate each place through an input from the user and not only locate the business on the map, but also provide information such as *“the name of the place and its address, geographical location, place ID, phone number, place type, website URL, and more.”[[3]](#footnote-2)* The user inputted string in the text box is passed to the API server, which in turn provides the coordinates of the corresponding place (marked as the “Finish” marker).

***Section 2.2.2.3: Directions API***

Walk with Wakki also uses the Directions API in order to compute and display the shortest route between the “Start” and “Finish” markers. This API was also manipulated in order to ensure that all directions are delivered for walking rather than any other mode of transport. Additionally, this API utilizes latitude/longitude coordinates within our application to identify locations. Initially, start and destination coordinates are integrated into a URL query that is passed to the Directions API server. The app then retrieves walking route data in the form of an JSON object. An additional class, DataParser, converts the JSON object into a sequential set of map movements that can be overlayed onto the map interfaced (called the polyline object). The final polyline object represents the optimal walking route from the start marker (retrieved by GPS) to the finish marker (retrieved by Google Map Places API).

***Section 2.2.2.4: GPS LocationManager***

LocationManager is one of Android’s many API Classes. It “provides access to the system location services [which] allow applications to obtain periodic updates of the device's geographical location.”[[4]](#footnote-3) *Walk with Wakki* uses the getLocation member function found in the LocationManager Class. This function does exactly what is says is does-- it gets the current location of the user. This value is marked as the “Start” marker.

***Section 2.2.3: Distance Calculation***

The distance between the “Start” and “Finish” markers is computed by the an implementation of the Haversine Formula as suggested by Yugandhar Babu, a coder we found on StackOverflow. Although this source code may not have been sourced from the most credible source, instead from codecodex, we verified that it works and outputs the desired values.[[5]](#footnote-4)

***Section 2.2.4: XP Calculation***

To calculate the XP the user gains upon the completion of their workout, we use this formula that is a function of the distance travelled by the user:

XP = round(Distance(km) \* 1.5 + 12

This value is then added to the user’s *current* XP, which is a value stored in the FireBase database.

***Section 2.3: Front End Functionality***

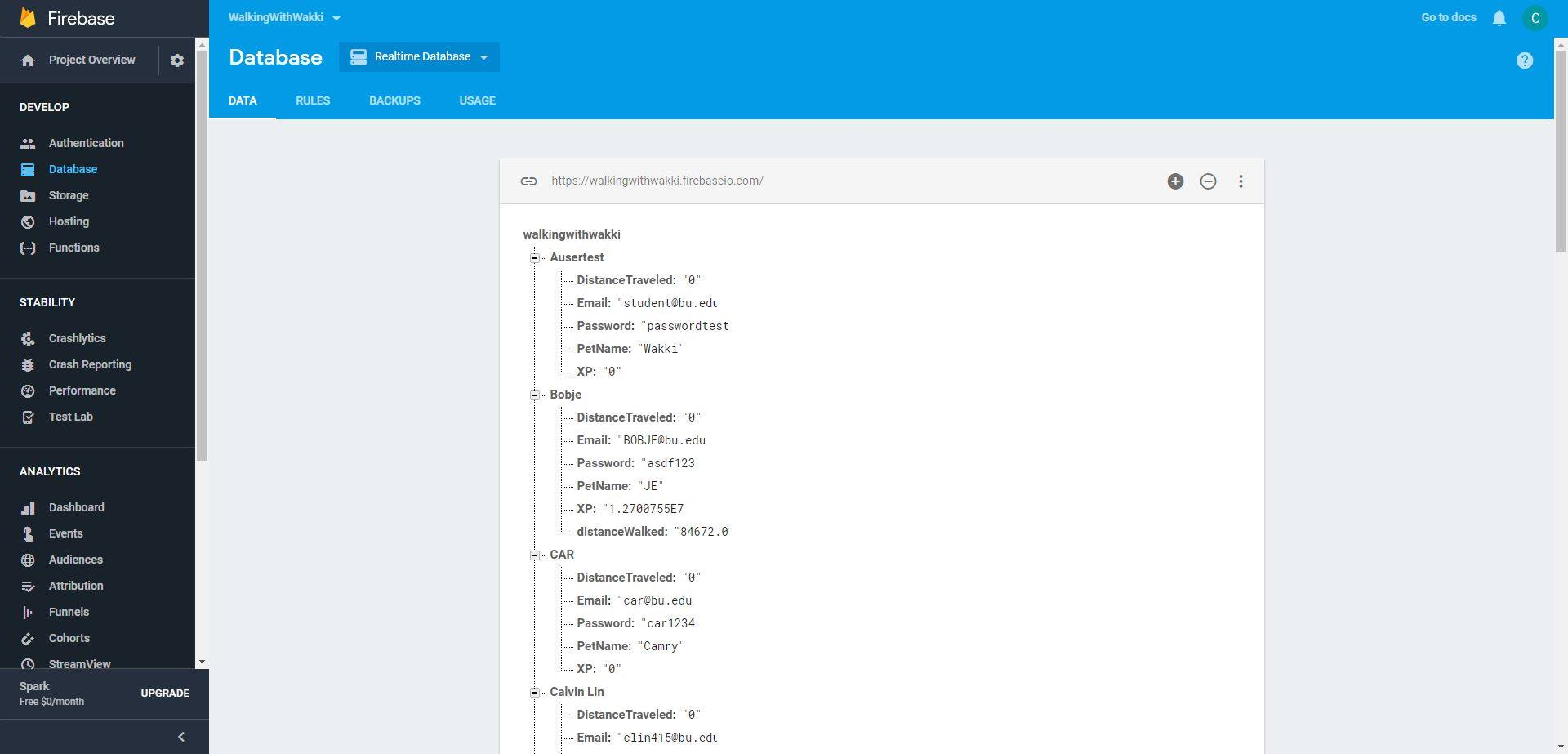
On the front end, this application is designed to be simple, intuitive, and easy to use. These design choices are integral to the ability of elderly to be able to read, write, and interact with the app and focus on utilizing a clean white background, black writing, clear and concise instructions within each text entry box as well as well labeled buttons that clearly suggest what each will do. The first screen visible to the user is the CreateAccount page; on this screen the *Walk with Wakki* logo is displayed and below this, textEdit boxes prompting for email, username, and password are shown. At the bottom of the screen a button labelled Create Account is shown. Upon clicking this button, the user is redirected to the NamePet screen where they are prompted to input their pet’s name, and upon clicking the home button, are redirected to the PlanWorkout. The PlanWorkout screen has a very simple design, merely showing the logo, the pet, a greeting from the pet and the score. At the bottom of the screen is a button showing Begin Workout which, upon clicking, redirects the user to the MapsActivity. This screen permits the user to input the destination they wish to reach and upon pressing the button GO they will be given a route to walk. At the end of his/her walk the user presses the button STOP and the user is redirected to the Walk Success screen where the logo and pet image are shown and the user is congratulated on the completion of the walk and instructed to press the Home button to return to the PlanWorkout screen. On return to the PlanWorkout screen the user will be shown a ‘hello’ message from their pet and their updated score, or XP, will now be shown. At this point the user has the option to go for another walk and repeat the process, or they can minimize the app and return later to walk again.

***Section 3: Testing Mechanisms***

This app is highly reliant on the functionality and integration of its database, which stores all the different user profiles and their account specifics like XP, pet name etc. An Android Virtual Device was used to test the functionality between Firebase and the Google navigation APIs. In addition, a Samsung S8 was used to test functionality of the app in-between screens.

**Section 3.1: Firebase**

In order to ensure proper Firebase functionality, we ran the code both on an Android Virtual Device emulator and an Samsung S8 device and observed the addition of the information on the Firebase database online. Because Google Firebase includes real-time updates for the database, we were able to verify within seconds that 1) the proper user information was being stored in the proper fields of the database, and 2) that a User object with matching credentials within the Authentication tab of Firebase was created. On the device side, the pet name and score fields were displayed in the “Plan Workout” screen and we tested to make sure that these values returned non-null. Many users were populated through both devices to ensure functionality.



**Section 3.2: Maps**

**Section 3.2.1: GPS**

The GPS component of the application was developed after the Maps API was already established in the app. Originally the application required both initial and final locations to be entered; however, GPS capability was incorporated to prevent the user from having to enter their location to use the application. In order to error check the GPS component of the application, we walked around campus with the application on and pressed the homing button on the maps screen. The Maps API was able to correctly place the location marker at all locations, proving that the GPS component works.

**Section 3.2.2: Route Plotting**

In order to ensure that route plotting functions correctly within the app we ran multiple trials with different endpoints. After using the GPS to locate our start point we began to input a variety of local endpoints, checking that the route drawn by the application instructed the user to follow a route from the start point to endpoint correctly.

**Section 3.3: Distance Return and XP calculation**

The distance return and XP calculation were again tested through a similar trial and error method as the previous Maps section. After plotting routes, walking them and pressing stop on our application we checked the Firebase database to ensure that the distance, in meters, was recorded. For the score we multiplied this distance by 1.5 and added 10 before rounding to the nearest integer and then reported the value on the “Plan Workout” page. We checked that the reported number was indeed 1.5 times the distance to ensure that the equation is functioning correctly.

***Section 4: Future Iterations and Goals***

As this app has the potential to make a great difference in the day-to-day lifestyle of the growing elderly population within the United States and take a stride towards reducing obesity within this population. For this reason we believe that future iterations of this app can become more complex, permitting the user to gain more health benefits from each use. The first improvement we would implement before release would be to integrate the initial specifications we were not able to achieve during this semester's first stage of development. This would first allow the user to add additional information to their account including age, weight, and daily exercise goal; through the addition of this information the user will be able to monitor their improvements in weight and exercise as they get older. The second large change we would wish to implement would allow prompting of the user to walk a set percentage longer after a certain period of time had passed. This would, as mentioned previously, be reflected on the back end with the generation of longer routes via Google Maps to locations of interest in the user’s vicinity. On the front end, the motivation for the longer walk would be conveyed by having the GUI update to show the user's pet wanting to "play somewhere new" (route user to a park or museum) or "try a new food” (route user to a restaurant). Finally, we would also output notifications encouraging the user to return to the app if it has not been used for over a week.

In addition to these functionality changes, some design changes would be introduced. These changes would include the animation of Wakki to advance the Tamagotchi-like relationship by making the user’s pet feel more real. Additionally, the introduction of geotags and messages from Wakki would transform this application from the merely health conscious way in which it is presented now to a fun game that is sure to increase interest in its use. Furthermore, we could see further development through the partnership with companies such as Samsung and Fitbit. If this collaboration was achieved the data reported by Walk with Wakki could be integrated into Samsung’s health app as a third party activity monitoring contributor. Additionally, collaboration with Fitbit could permit the storage and presentation of further data regarding cardiovascular health; in particular heart rate could be reported throughout the walk to demonstrate to the user which areas challenge them the most. This information could be recorded over time in the database to allow the user to monitor improvements not only in their weight but also in their overall heart health.

1. Mather, Mark. “Fact Sheet: Aging in the United States.” PRB, Population Reference Bureau, Jan. 2016, www.prb.org/Publications/Media-Guides/2016/aging-unitedstates-fact-sheet.aspx. [↑](#footnote-ref-0)
2. Ibid [↑](#footnote-ref-1)
3. “Getting Started | Google Places API for Android | Google Developers.” *Google*, Google, developers.google.com/places/android-api/start. [↑](#footnote-ref-2)
4. “Android.location.LocationManager.” *Android Developers*, 4 Dec. 2017, developer.android.com/reference/android/location/LocationManager.html. [↑](#footnote-ref-3)
5. “Calculate Distance between Two Points on a Globe.” *CodeCodex,* www.codecodex.com/wiki/Calculate\_Distance\_Between\_Two\_Points\_on\_a\_Globe. [↑](#footnote-ref-4)