**CA326 Year 3 Project**

**Technical Specification**

**Dylan Sloyan - 16511419**

**Samuel Wood - 16476066**

**Life Logging Application**

**Table of contents**

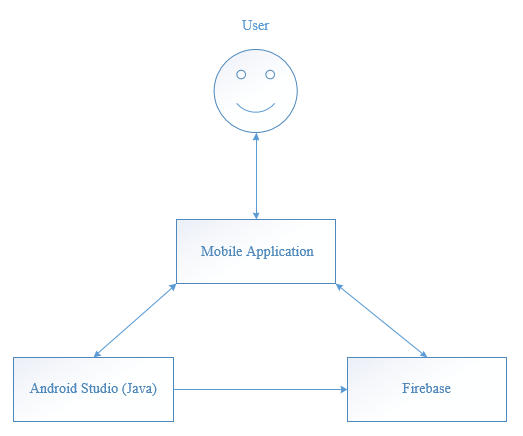
1. **Introduction**

1.1 **Overview**

Our system is an Android app designed for people who are interested in keeping a life log and also to help people that suffer from deficiencies such as short term memory loss, dementia or Alzheimer’s. Our project allows users to record and view their day to day activities. The app operates through three main features. The first of which is the devices camera where the user may take photos and add a description. The users photos can then be viewed within their designated gallery. The second attribute of the app is the Location feature. This allows the user to view their location by a blue dot and can be returned to at any time while using this feature. The user can also search for a location of their choosing and acquire directions to this location by being given the option to switch to google maps,. There is also a Step Counter implemented in the application. When the user logs in the Step Counter is automatically activated. The number of steps taken by the user can be viewed by selecting the Step Counter button from the home screen. The app holder can use this feature as a simple means of knowing if they’re desired goal for steps taken each day is met. Also, the number of steps resets to 0 at midnight every night so the user can review their steps everyday. The user is given the option to make an account with an Email address and a password of their choosing. If the user is logged in, they will not be asked to enter their details again until they sign out from the app.

2. **System Architecture**

2.1 **High level overview:**

****

As illustrated in the figure above our application can be decomposed into three key platforms.

**2.2 Android Mobile Device**

This is simply the user interface, also known as the front end of the application. It allows users to interact with the features the application has to provide. The page layouts of the interface are consistent and strategically use text and colour for ease of use.

The mobile device must have an active internet connection in order for the application to work. The user must also accept permission requests from the application allowing it to access their device’s camera and location services.

**2.3 Android Studio**

This core component works as the brains of the overall system by dealing with inputs and outputs of data. It works in conjunction with both the mobile device and the firebase. It uses Firebases dependencies and methods to manage data appropriately between each area within the Firebase console.

**2.4 Firebase**

Firebase essentially works as a development platform for the application. A specific console is created within Firebase. The console contains all the necessary data handling features required by the application, also allowing the developers to monitor and manage user activity.

Firebase authentication deals with all the emails which were used to register an account. A unique user ID is created for each email as a reference. The date of when the email was registered and also last used to sign in to the application can be viewed here.

Firebase storage simply acts as a folder to store images which are uploaded by the user. The image is given a time and date of when it was uploaded along with a Url.

The Firebase Real-time database creates a folder with a list of keys. Each unique key contains a description and its corresponding image url which is used to be retrieved and shown within the application on the user’s mobile device.

3. **High-Level Design**

Using Visio, we created the following diagrams to cover the system architecture, describing the relation between functions, modules and the database within the system:

*Context Diagram (3.1)*

In this diagram the application is viewed as one big process. All external entities that interact with the application are included and can be seen as squares around the process. The arrows pointing to and from the circle are the inputs and outputs of data/instructions between the process and its external entities.

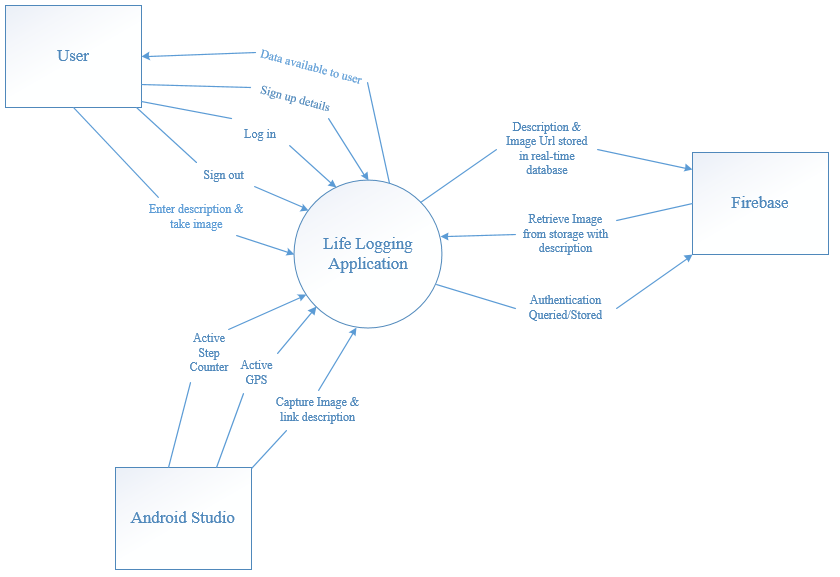
*Data Flow Diagram (3.2)*

This is a depiction of the flow of data through our intended software system. Creating this diagram helped to provide an overview of the data and functionality within the system. External entities are rectangle boxes, processes can be seen as ellipses, data stores are the rectangles with missing sides and data flow is conveyed through arrows.

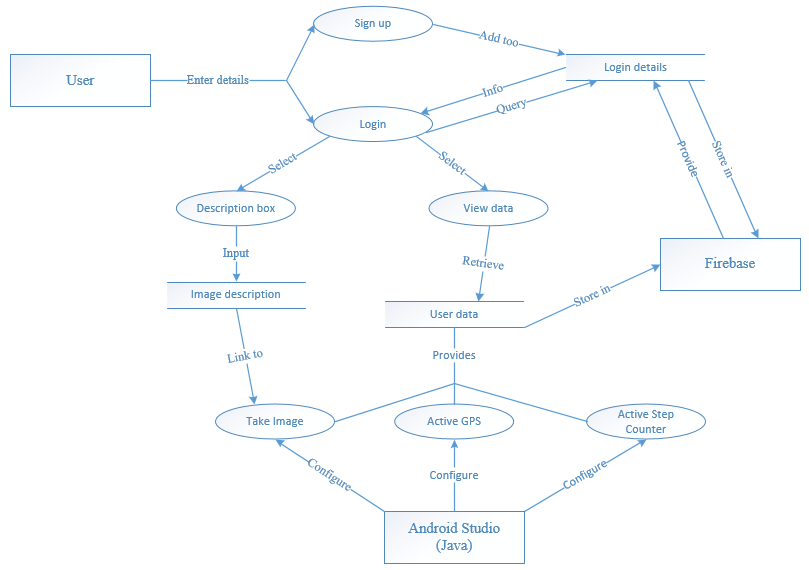
*Sequence Diagram (3.3)*

The sequence diagram portrays the order of interactions that take place between the user and the subsystems within the overall system. The subsystems are square boxes, the dotted lines represent the timeline of events and arrows are the passing of information between subsystems.

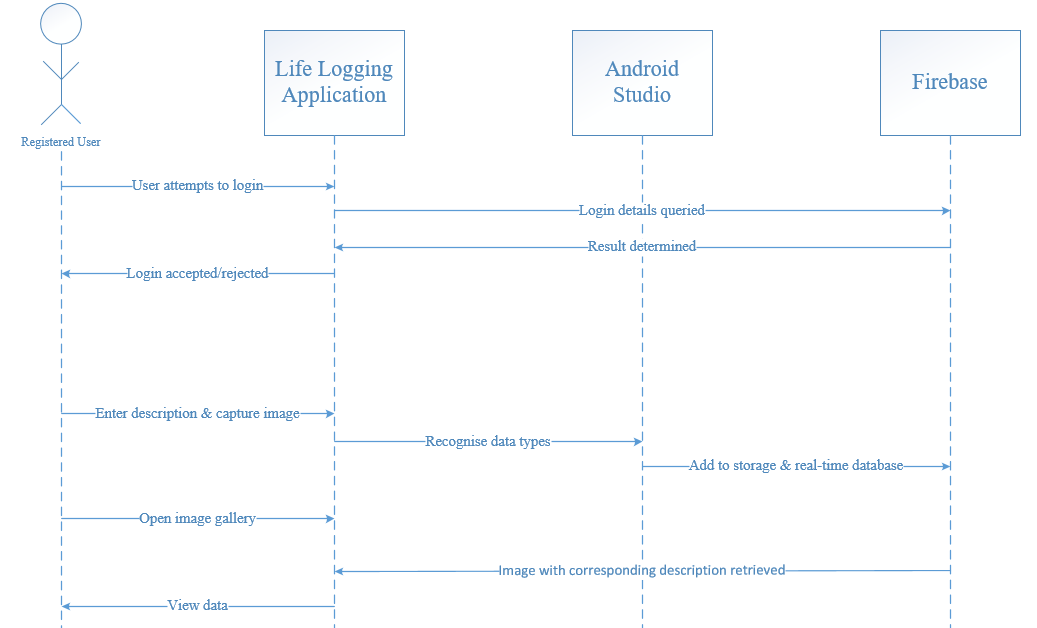
**3.1 Context Diagram**

****

**3.2 Data Flow Diagram (Level 1)**

****

**3.3 Sequence Diagram**

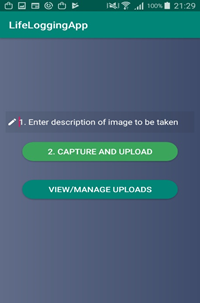
****

4. **Problems and resolution**

4.1 **Button to Capture Image**

Initially we sought for our app to capture images automatically at a chosen time interval set by the user. We intended this to done with the use of an alarm manager in Android studio. With other features that still required a substantial amount of work and with less and less success with this part of the project we needed a resolution. We used our incentive to implement the description feature in our project. This means the user can enter a description to go with their captured image.

*Fig 4.1 Camera activity screen*

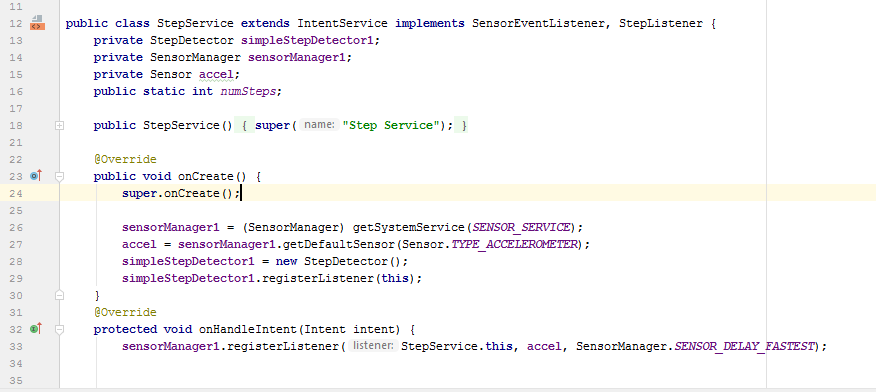


Once the image has been captured it can then be retrieved along with the users description. If the user does suffer from short term memory loss, this feature can act as a means to jog memory or to remind the user. For example, a picture of their medicine and a description of what time they should take it.

4.2 **Implementing a Background Service for the Step Counter**

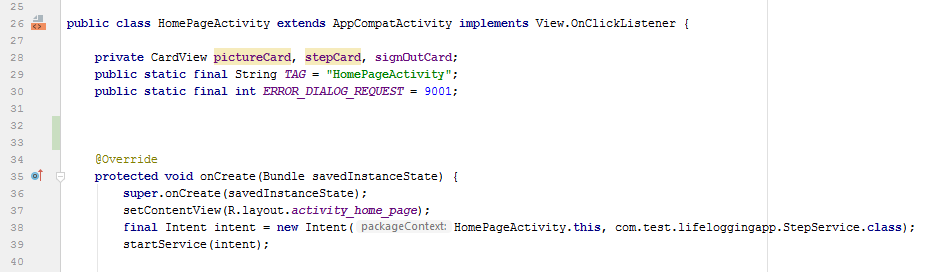
When the step counter was first implemented, it contained only a start and stop button and the number of steps would reset whenever the user would exit the Step activity. The user would have to open the app, go into the step activity and push the start button and not use the device for anything else until they no longer wished for their steps to be recorded. This was a big problem. This was occuring because the Step counter was being initialized in the ‘OnCreate’ method in the Step activity. To resolve the issue their needed to be a background service class and thus the ‘StepService’ class was made. In this class there is an intent that contains the line to initiate the “registerListener” for the Step counter, see lines 32 and 33 of the below diagram.

*Fig 4.2 StepService Class*



The call for this intent was then placed in the “OnCreate” for the Home Page activity, in other words when the user logs in and runs indefinitely. See lines 38 and 39 of the below diagram.

*Fig 4.3 Home page Activity class*

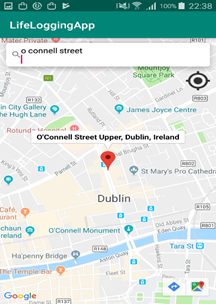


With the use of the Alarm Manager feature, we were able to reset the number of steps to 0 at midnight every evening. The user would then be able to view their steps without having to activate the feature themselves.

4.3 **Marker left where pictures were taken**

Another problem we ran into was recording the location where each picture was taken. Our goal was for the users map to have a marker where each image was taken. This was supposed to help users that have one of the deficiencies mentioned in our overview. The feature was intended to help them retrace their steps. With little progress we devised the resolution to provide the user with directions to their desired location.

*Fig 4.4 Searched location*



When the location is entered in the search bar it drops a marker that will serve as a means of direction. This could be a big help to someone with a poor memory should they become lost or if they need to keep these set of directions for future reference. The user’s current location is displayed by a blue dot and can be returned to at any time by pressing on the black icon underneath the search bar.

5**. Installation guide**

*Due to the nature of our application the step counter will not be functional on an emulator and neither will the emulators rear camera. The app will have to be installed on a real device to test these functions.*

5.1 **Requirements**

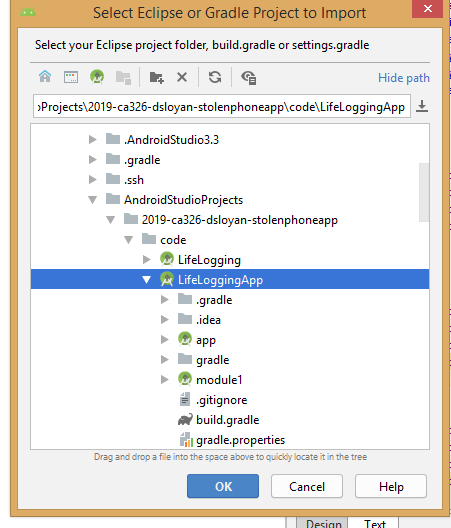
* Android Studio 3.3
* Java 1.8
* Android Device (5.0.1 or Higher)
* Internet Connection
* Access to camera and location services on device

The Life Logging application is not available on the google play store. Alternatively the app can be installed using a git repository.

5.2 **Instructions**

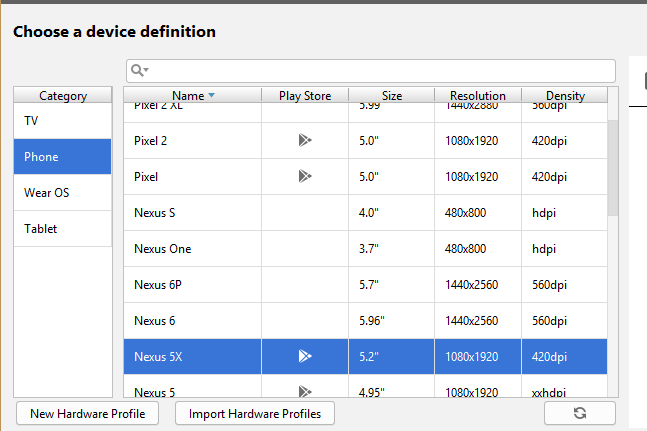
* Have a cloned git repository and change to your desired directory.
* Run the below git command.



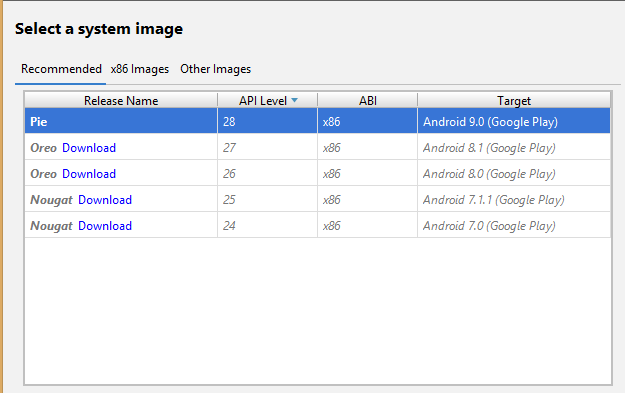
* Go to android studio and close any open Android Studio Projects and choose import project . Then navigate to the file in the above screenshot.
* Select LifeLogging as in the below image and then wait for the build to finish.

5.3 **Instructions for device**

* Go to tools and select AVD manager.
* If there is no usable device create a new one.
* Select Nexus 5X for the device



* Choose Pie for System Image, Android 9.0 is the Target.

****

* Select portrait for for the orientation and then Finish.
* You have now created a new device. Select the “Run App” option at the top of the platform under “Run” or the green arrow to the right of it and select your new device.
* Make sure you have internet connection.

**Physical device instructions**

* Go into settings on device and select “About phone” and go to “Developer options” and enable USB debugging.
* Always allow access on this device to the computer you are using.
* Select the “Run App” option on android studio and subsequently select “Run” and then choose user running device.
* Select your device.