Crime Zone -

Can Zhong, Amin Phakhruddin

**Grand Vision** :

**Motivation -**

When I (Can Zhong) went out west for my internship, I was alone with only technology as a tool to organize my trip and life. It was both at time consuming and difficult to plan a week ahead, let alone a day and I never really stuck to my plan. I got to the point where creating a plan was not worth the time involved because I was going to end up trashing it with the spontaneity that naturally occurs with life. Therefore, we set out to create a product which will allow Users to plan spontaneously.

**Use Cases -**

To answer this problem, we are trying to create a social network that connects users with Activities or Events which they are interested in based on their real-time GPS positions. The Events would be categorized, and the User would determine which type of Events they would like to be notified of. This would also allow Users to notified of nearby Users of event’s which they are hosting. The ideal scenario would involve a User being made notified of an Event (outreach programs, concerts, or crime) when their zone overlaps the Event’s zone. The app could be used for many purposes like leisure or security.

If a user wants to travel to a foreign city, they may not be familiar with the good and bad areas pertaining to criminal activities. They may also not be aware of extracurricular activities around them. Although there are resources which could plan out a User’s itinerary during a day, there is no current app or service which allows an individual to choose their activities on the fly, or spontaneously. Users will be known of fun activities near and to allow them to be at ease when it comes to planning events This would be beneficial for lone Users who have the natural trait of exploring and wandering without a plan or without another companion. The unknown dangers of traveling alone can be hard enough when it is planned ahead, but to be absolutely spontaneous can spell danger for the User.

**Goal -**

We strive to create a service which will allow these Users to safely traverse their paths by making the User more mindful of high-danger zones. Additionally, but most importantly when it comes to avoiding high-danger zones. Our most important feature would be to rapidly upload a criminal activity. This would put power in the hands of the Users when it comes to defending themselves since the first defense should always be to avoid high-danger zones. With the warning system, the police can then focus their resources on more serious crimes. Not only will the police be able to reallocate their resources, but there arises an opportunity for the police to utilize the data generated by criminal events to analyze and predict future events.

**Semester Goals -**

Given the current constraints of time and resources, our goal for the semester is to develop a minimum viable product that exhibits the basic functionality (key features) of our grand vision.

As stated above our grand vision for the project is to build a social network that allows people to be aware of the events happening in the user’s particular zone. Making users aware of all the events that are of interest is a major task, so we decided to simplify our events to events of danger, harm, and urgency. These events would surely be of interest to everyone.

Our goal for the semester is to alert users in real-time to dangerous activities/events within the user’s zone. We haven’t decided what a zone is exactly for a user, but a zone might be a 1-mile radius, or it could be the neighborhood, depending on what’s most feasible and important to the user. If we define a zone, to be within the neighborhood that the user is currently in then the user should be notified of all dangerous or harmful events within the neighborhood the user is currently in.

We imagine that a user should be able to open the mobile app on his phone and be taken to a user map. Once the user has launched the app, he should be able to discover all the dangerous events that have taken place within a 24 hour period. This is akin to how the Google Maps app automatically displays to the user, the current traffic situation.

The user then has the ability to enter a destination for his walk or travel. He/she would be presented a route along with the events that are happening or have happened within the vicinity of the user’s route.

Users should also have the ability to update the map with their own events that they are aware of. For example, if a user found out or comes across a crime, then he should be able to update the map of that event so that other users are aware as well. Other users will be notified of the event in real-time.

The primary purpose of this app is to empower individuals to be aware and take control of the events within their environment.

**Technologies -**

**Operating Systems:**

We are going to be using Android Studio to create a Java-based app using Firebase as our backend server for authentication, databases, and persistence storage.

Android Studio - We decided to implement our App to be run on the Android OS because Android is the most widely used mobile operating system in usage today. Our app depends on high activity from many users so we wanted to create an app which will be accessible to the largest group of people.

Other: iOS, web - We did not choose these because iOS is an OS which we do not have a lot of experience with as a team. Therefore, not only would we have to design and learn a new framework, but also Swift the language used in iOS. The web is not a preferable mode because of the use cases of our app. This would be a real-time app which would have to be used on a mobile device and so the interfaces would have to be integrated with the native device due to the GPS real-time location being needed. The problem with the browser is, it cannot efficiently fetch the necessary data we will need. APIs for the native code is also easier and faster to implement than using a web-based front end. We could have used an alternative which would make the most sense, but due to time constraints we have decided not to use it and that is Flutter. Flutter is a hybrid app ecosystem which allows development in one area to be separate from deployment and compilation to native code. This would allow us to deploy to all three Android, iOS, and the web.

**Front - End:**

Java XML - We chose to implement our project in Java with XML because we have the most experience with this language as it is what we learned the past four years at UWM. Also, given the time frame, it is not preferred to learn a whole new language and framework. This will allow us to provide a product that is supported by our years of using Java versus other languages.

Other: Kotlin - Kotlin is Google’s advanced version of Java. We do not have any experience with Kotlin, but because it is object-oriented and shares many common paradigms with Java, learning it would not be that difficult. However, to provide the best product, we decided to use what we are best with so we focus on the engineering aspect.

**Back - End:**

Firebase - We are using Firebase because it is heavily integrated with android development. Firebase is really easy to use and powerful. An alternative could have been AWS. We didn’t choose it because AWS is not as integrated as Firebase.

**App User Interface Layout:**

**Terminology:**

As mentioned above, we decided to implement our mobile app, Crime Zone, as an Android app. For this reason, we would like to explain some of the terminology specific to building Android applications. One term that we will use often throughout this report is “activity.” An Android activity consists of the java file as well as the xml layout file. So, the term activity encompasses design as well as the logic in order to make that design functional. For example, the “Start Activity” encompasses both the java file, StartActivity.java, and the layout resource file, Activity\_start.xml.

**Design Overview:**

When the app launches for the first time, the user will be taken to a start activity. At the start activity, the user is given the choice of either signing up or logging in. If the user already has an account, he can choose to log in and will be taken to the login activity. If the user does not already have an account, then the user should choose to sign up (create account) and will be taken to the sign-up activity.

Whichever route the user has chosen, the user should fill out the necessary fields and credentials required based on the activity. The login activity only requires an email and a password, while the sign-up activity requires some additional fields. After filling out these fields, the user will then be authenticated by the Firebase authentication service. These activities will be discussed in detail in the following subsections.

Once the user has been authenticated, the user will be taken to the main activity. The main activity is the home of our app, Crime Zone. The previous activities are a necessary process for a user to reach the main activity, if the user is not already logged in (details to be discussed in the main activity section).

As of now the main activity is blank and doesn’t display any information. This will soon be the home of the map that shows the events have or is happening around the user, or within the user’s zone. Within the main activity there is a menu, that allows the user to navigate to the other activities. Crime Zone’s functionality can be split into two areas, the activities revolved around the user and the activities revolved around the events.

The activities that are revolved around the user are:

* Profile Activity
* Profile Settings Activity

The activities that are revolved around the events are:

* Event Activity (The profile of a specific event)
* Register Event Activity (Add the event)
* Event Settings Activity (The details and settings of the event)
* All Events Activity (A list of all the events)

Diagram of the login and sign up process:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**A close up of a logo

Description automatically generated**A screenshot of a cell phone

Description automatically generated**A screenshot of a cell phone

Description automatically generatedA picture containing electronics

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedDifferent activities available to the user:**

**Activities In-Depth:**

**Start Activity:**

If there is not a currently logged in user, this is the first activity that a user will be taken to. By implementation and design the app is designed to launch the Main Activity first. Within the MainActivity.java file, there is a check for whether there is a currently logged in user. If the user is not currently logged in, then the Main Activity tells the app to take the user to the Start Activity.

A screenshot of a cell phone

Description automatically generated

As seen in the code, we are using the methods from the Firebase libraries that we imported into our project. Firebase, has multiple libraries, depending on the functionality that we want to include in our application. In this specific case we are using the Firebase authentication library, which includes the log in and sign up methods, along with authentication checks. Using this library makes it easy to check whether a user is logged in or not.

A screenshot of a cell phone

Description automatically generatedSo, when a user is logged, the Firebase user will not be null. If there is no user currently logged in, then there will be a return value. If null is returned there is no user logged in, and an intent is created to take the user to the start activity. An intent is essentially Android’s way of moving from one activity to another activity.

As shown in the screenshot, there is a line of text to indicate to the user application that has been started. Within this activity there are two buttons. One button will navigate a user to the sign-up activity and the other button will navigate a user to the log in activity. These two buttons design components are defined in the xml layout file, and the actual logic behind each button is defined in the StartActivity.java file.

**Login Activity:**

A screenshot of a cell phone

Description automatically generated

This activity includes two input fields. One is simply a text input field while the other is a password input field. The main difference is that the input in the password input field is hidden, while the other is not.

The first field is meant for the user to input their email address, and the second field is input the associated password with their Crime Zone account. Both fields are required to be inputted for Firebase to authenticate the user. So, the app will take the input and pass it to a Firebase method that authenticates the credentials. Based on the credentials, Firebase will try to match the credentials to an existing user. If no user exists, the result of the authentication process will return unsuccessful. If the result of the authentication process is successful, then an intent will be created to take the user to the Main Activity.

Once a user has been authenticated and does not log out, Firebase will recognize that user is logged in. So, if the user closes the app and relaunches the app, the user will be taken straight to the main activity.

Although, we only used email authentication, Firebase also offers other modes of authentication. For example, there are authentication processes for Google Accounts, Facebook Accounts, and even by phone number. Using the other authentication processes also open some other possibilities, such as access to setting up google calendar invites, etc.

**Sign-up Activity:**

A screenshot of a cell phone

Description automatically generated

This activity includes a couple input fields. The app will use the values taken from the input fields and pass it to a Firebase method that creates the user. Once created, the user’s credentials can now be seen under the list of users in Firebase. The email address is shown, but not the password.

A screenshot of a cell phone

Description automatically generated

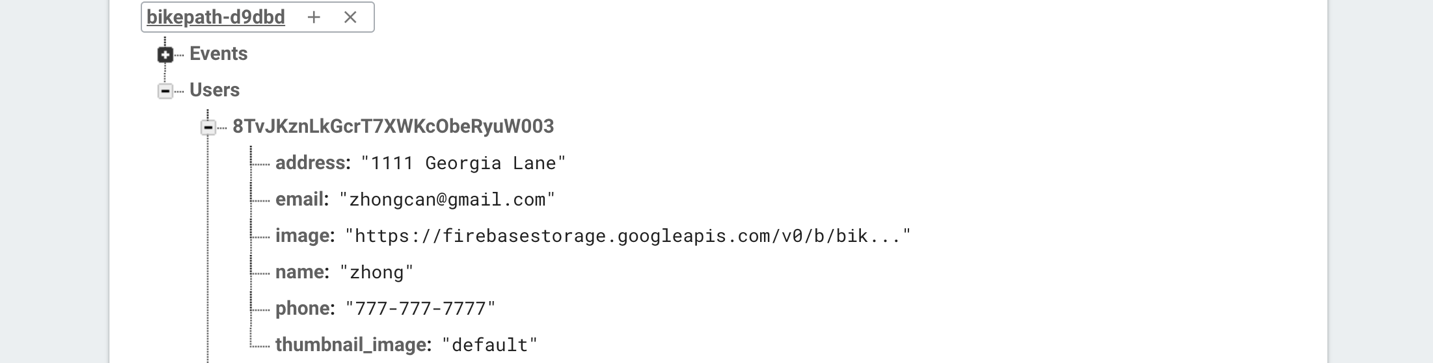
One difference between this activity and the login activity is that we call a Firebase create account method instead of authentication method. Another difference is that we also create a record of the user’s information within the Firebase database. This is different then just creating a user. In order to do this, we included the Firebase dependencies for the real-time database. After including the dependencies, we now have the ability to connect the real-time database to store the details of the user, such as address, phone number, and name.

So, we create a pointer that points to the Firebase real-time database. Then we create a pointer that points to a child titled “Users.” This is where the application will store all the user objects. After pointing to the “Users” child, we create a user object with the key being the current user’s id, which is specified by Firebase. Under that key, the application stores all the information regarding the user.

A screenshot of a cell phone

Description automatically generated

In the code snippet above, a hash map is used to store the information. This was easier and simpler to follow than create multiple lines of code that would place each key and its corresponding value into the database. Instead, a hash map stores the key and its corresponding value. This hash map is then passed on to the Firebase method that will update the real-time database.



Note: Amin used an existing Firebase project Bike Path, but Crime Zone is a sub application of the project Bike Path. This is why, is says bike path at the top of the image.

**Logout:**

There is no specific activity for logging out, but there is a menu option in the main activity for logging out. Logging out is extremely simple with Firebase and is part of the methods included with the Firebase authentication methods.

Once a user presses the logout option and click event is fired, and the logout process is called. By doing so, Firebase logs the user out, and the current user is now null. The user will then be taken to the start activity. If the user were to close the app and launch it again, the Start Activity is launched.

**Event Activity:**

**A picture containing electronics

Description automatically generated**

The event activity is essentially a profile activity for the event that is created. The event activity contains a picture for the event, in our case, possibly the scene of the crime. Along with an image, the name of the event is mentioned and the address as well. All the information displayed in this activity is retrieved from the Firebase database.

A close up of a logo

Description automatically generated

In order to do this, a pointer is created to point to the event’s id which is passed to EventActivity.java from the AllEventsActivity.java file. Using the value store under the variable “eventId”, we create a pointer to point to the event id in the real-time database under the “events” section of the database.

A screenshot of a cell phone

Description automatically generated

**Event Settings Activity:**

**A screenshot of a cell phone

Description automatically generated**

This profile event activity is where the user can edit the info of the event. As of now the only information that can be changed is the address of the event. The information that prepopulates the input fields is not retrieved from the database rather it passed in from the event activity. This prevents unnecessary calls to the real-time database. There are many more instances where we could probably decrease the amount of calls, we make to the database, which would decrease the costs, especially if we had more users. As of now, everything is free since the amount of calls this application makes is small.

Once the user enters in the updated info and presses the button, a call is made to push the new information to the child whose event id matches the event id passed from the previous intent.

**Register Event Activity:**

**A screenshot of a cell phone

Description automatically generated**

This activity is where the user will create the activity by inputting the necessary information specified by the fields shown in the activity. So, a name and address are required. We felt that this was all that was necessary at the moment. In the future, we may update events to include more information. For the moment, the most important information that will be needed is the addresses, so that we can determine whether or not an event is within the user’s zone.

Currently the event is just an arbitrary default string, but we will soon include Google’s location services so that every event has an address or set of coordinates that Google maps can point to.

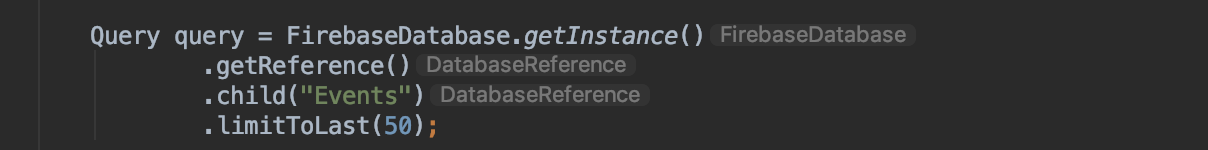
The logic behind this activity is similar to the register user activity. A hash map of the values is created to put the data under the child “events” in the real-time database, with the “event id” being the unique identifier for the registered event.

**All Events Activity:**

**A screenshot of a cell phone

Description automatically generated**

This activity is a display of all of the events in the Firebase database. It does this be creating a query to retrieve the information from the database. After querying the data, the activity now uses an event class to store and create each event. So, we create an adapter that allows us to retrieve the data and display it in a list. The adapter we use is a specific version of the normal recycler adapter.



A screenshot of a cell phone

Description automatically generated

Developing this activity was slightly more difficult than the other activities, due to the fact that were a couple of other files and processes we had to create. For example, creating a layout resource file for the adapter to use to display the information for each event. Along with that, the information retrieved in this activity would have to be kept and passed around to other event activities, so that they had access to the event information as well.

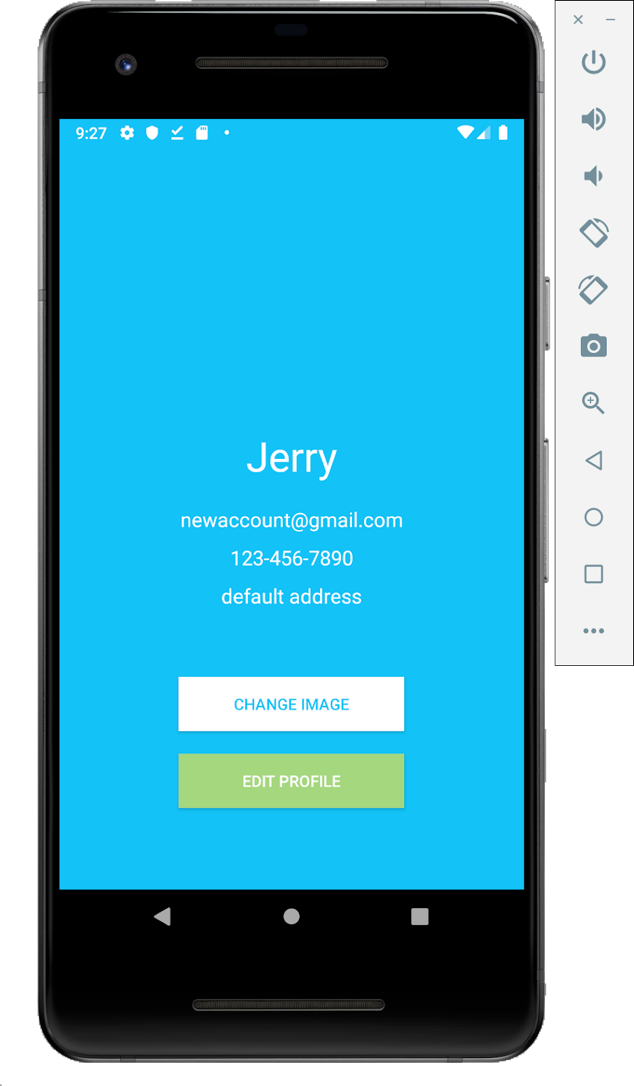
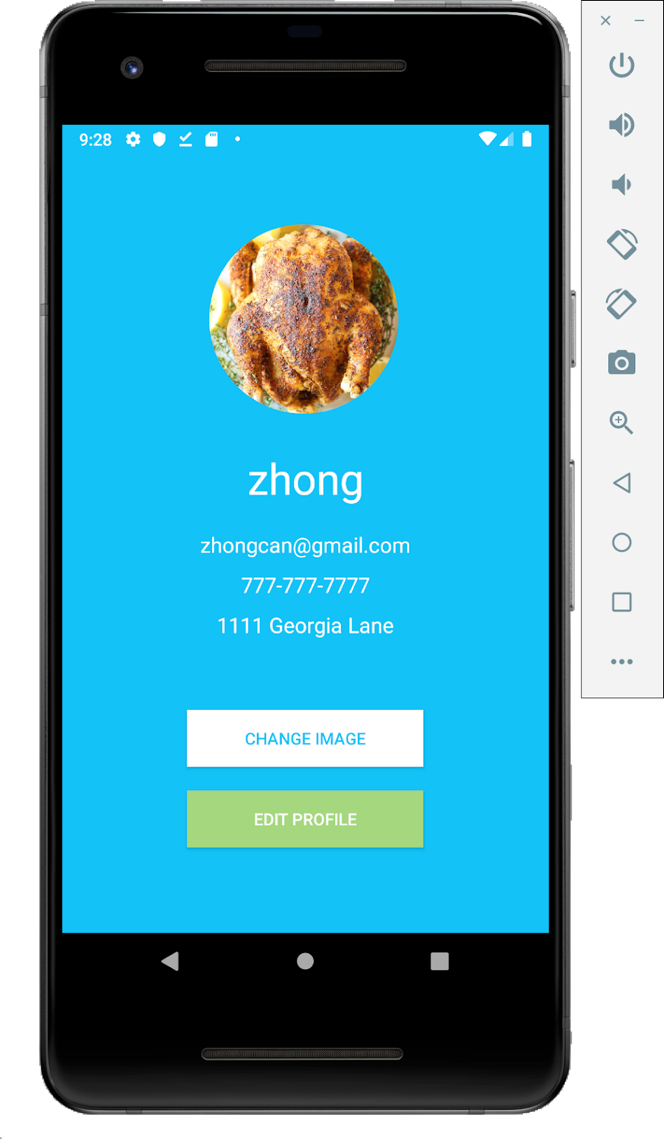
**Profile Activity:**

Activity is the portion of our app responsible for displaying the current logged in User’s profile information. In order to access the Profile Activity, the user must click the three dots in top left corner of the app and select Account Settings. This will change the view of the app to the profile. There are two possible views.

1. It is a **New User**, so there should be no profile image set and the default fields are used as a filler until the user is ready to change.

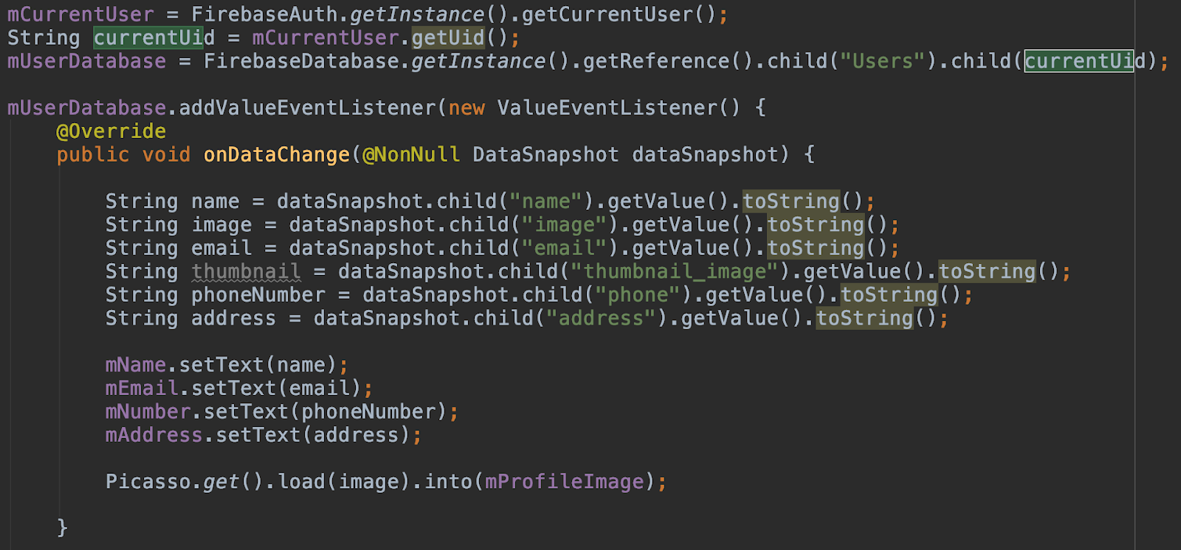
* The pre-filled fields for phone number is “123-456-7890”.
* The pre-filled field for address is “default address”.

1. It is a **Registered User**, and thus the app fetches data from the Firebase database and then displays the information received from the server.



**New User Registered User**

**Firebase Data Retrieval:**



The app connects to the Firebase database and attempts to get the User object associated with the currently authenticated user. When it receives the User from firebase, it adds a Listener that sets the fields of the various texts when Firebase changes the data. The fields set are Name, Email, Number, Address, and the Image.

**Picasso Image:**

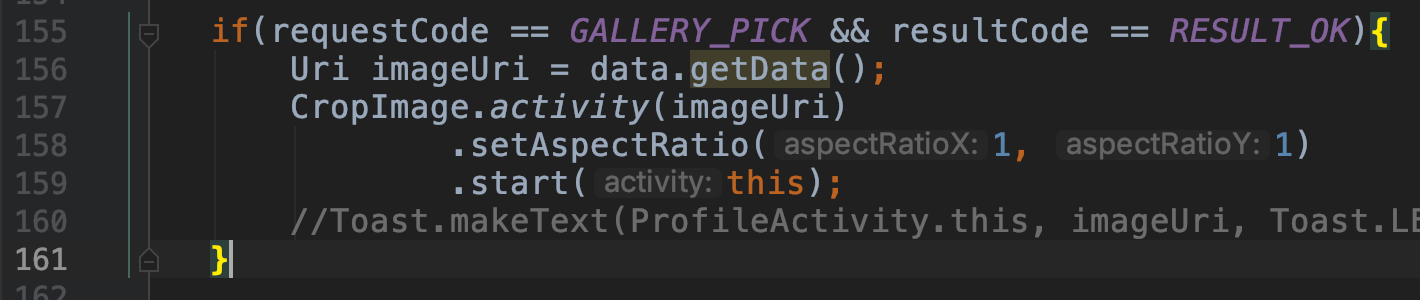
Picasso is a powerful open source library used for downloading and caching images for Android apps. We have decided to use Picasso for our project because of the support and the ease of use it provides us. In most scenarios, an image can be displayed with one line of code!



<https://github.com/square/picasso>

**Gallery Cropper:**

We decided to use a circular avatar image for app because of the pleasing aesthetics that it provides to the User. Many social media applications are starting to trend to this type of image like Facebook and Instagram. Our application first attempts to retrieve the image and then crops the image if it has received a successful image.

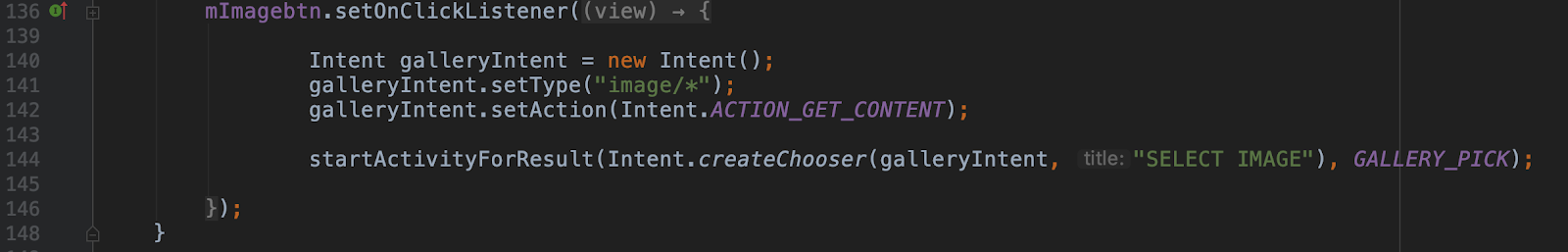


<https://github.com/hdodenhof/CircleImageView>

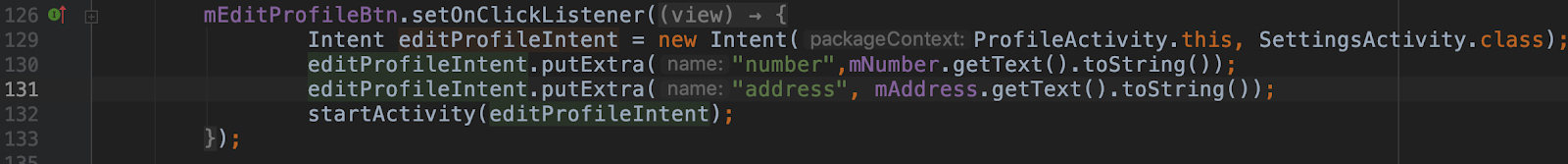
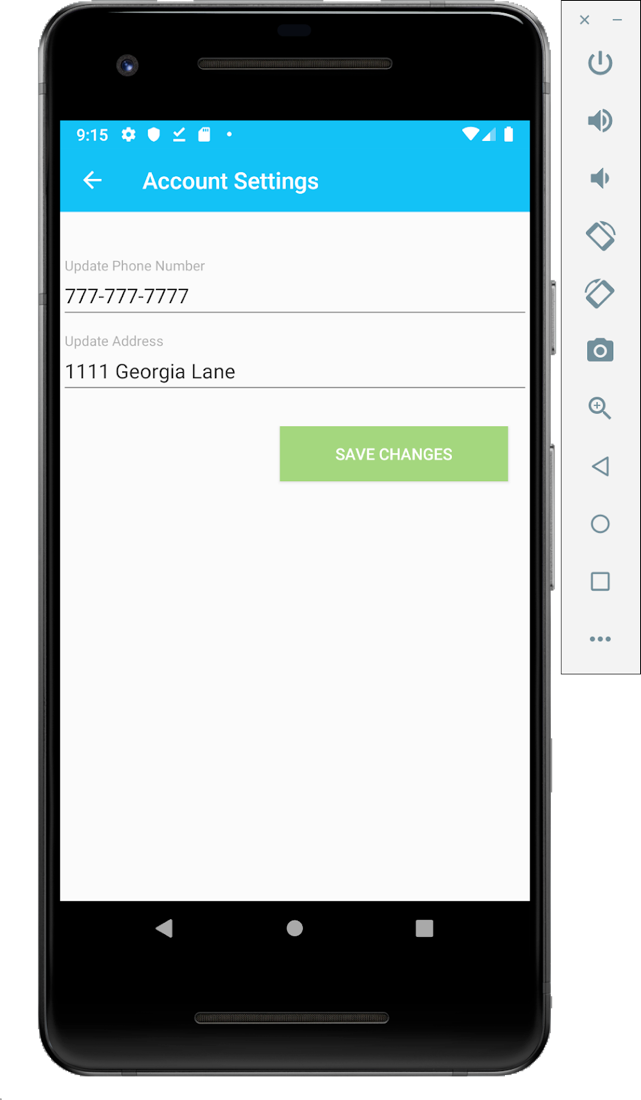
**Profile Navigation Buttons :**

There are two buttons relating to editing the profile that is available upon opening up the Account Settings page. There is a listener attached to each one that will perform different actions. The two buttons are

1. **Change Image**



**2) Edit Profile**

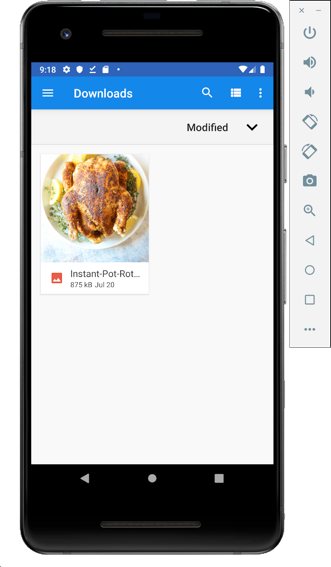
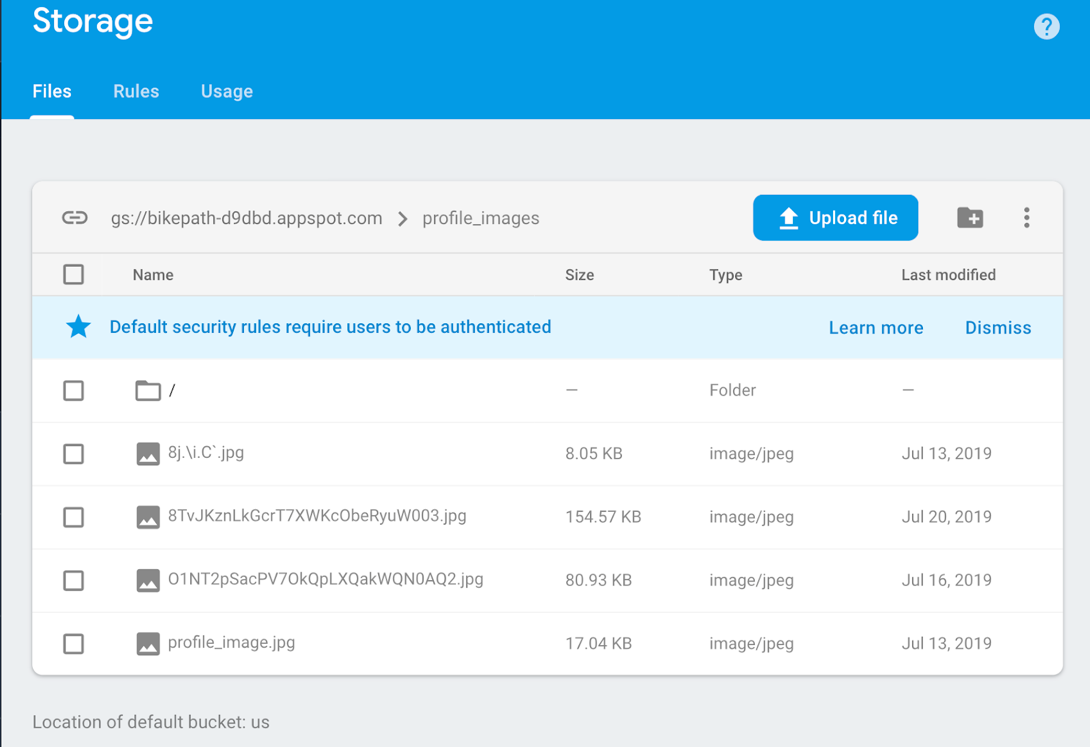
  
  
  
  
  


**Edit Profile Activity:**

When the Edit Profile button is clicked, the listener will send a signal to the current activity which will cause the current view to change to a form view which will allow information to be added to the current intent. Upon clicking “Save Changes”, the database will change the current value to the newer value specified. The back arrow must be clicked in order to change back to the previous page.

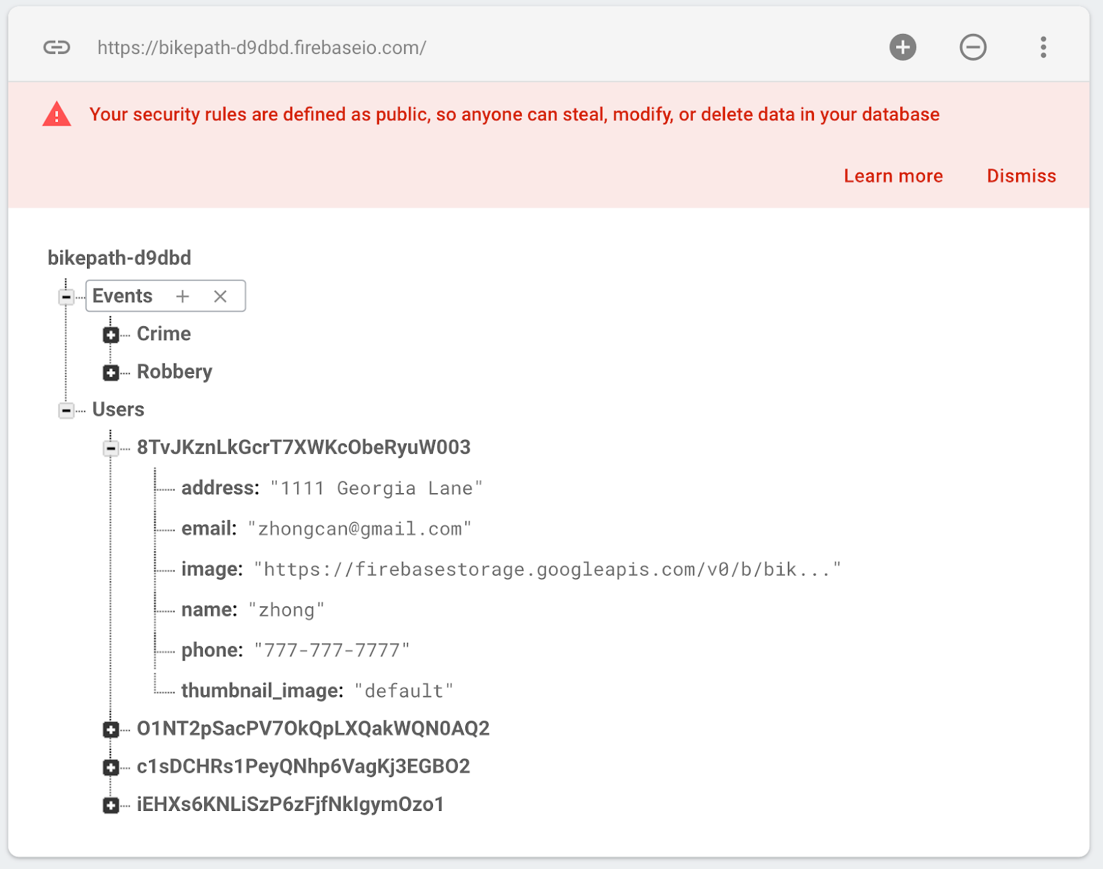


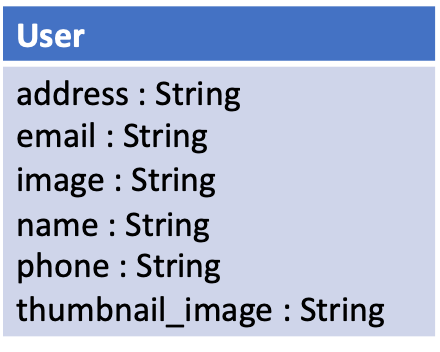
When the Change Image button is clicked, the listener will send a signal to the current activity which will cause the app to open the stock file path picker. An image can be picked that’s on the phone and when it does, it will also update the profile as well as uploading the image to the firebase database where User’s profile images are stored.



**UML Models:**

**User:**





**Conclusion:**

During this app developmental stage, we have designed the User Interface, implemented the core functionalities, and have integrated the app with the Firebase back-end.

After determining our core functionalities during planning, we immediately began drafting what the app would look like visually. We made several rough drafts of the views which we were considering and how buttons would change the view. We decided to stick with one that made the view containing the Event Activity our core page since that functionality is the backbone of our app and everything is built from that. We designed what the login view looked like as well as the account creation view. From there we decided how we were going to implement visually the Event Activity portion and we decided to use a list view as that is the view which would be easiest to implement. We planned on dedicating the core functionality of our app a whole developmental stage. We also designed the visuals of viewing profiles and editing them.

Once we drafted the design, we started to implement the core functionality in Android Studio. We used a test API before we integrated it with the Firebase backend to help speed up testing and development. We implemented and connected our app with the Firebase backend when the core functionalities were mainly coded and adjusted the code to work with Firebase. We implemented creating an account, logging in, logging out, sending and retrieving data from Firebase, editing a profile, creating events, and viewing them.

In the next stage, we plan on making our app more user friendly and interactive. This consists of changing the list view of events into an interactive map view of events. This will allow users to user their finger to scroll through an area. This kind of interaction is what we believe to be the most engaging and relative for the user. We believe a user who can visually see the distances of events and their location is more engaging than being given a number stating how far they are from the event. In order to create such an interactive feature, we first need to decide how large a zone is. This will be decided before we start the next stage. The overall goal is still the same, but we believe it is better to implement a feature we desired with excellence than having a bunch of small features that are mediocre. We really want to spend time on the map portion in order to make it as interactive and engaging as possible.