***EC327 A1: Nick Awtry, Veronica Herzog, Michaela Moynihan, Tiffany Wu***

**Java the Hut:**

**Project Documentation**

**for**

**Fun Finder App**



**Fun Finder Overview:**

The Fun Finder app is a social media application designed to provide users information about different events around them. The user would able to either create their own event in detail, or look for other users’ events in the surrounding area. To view others’ events, the user could either look at a map with markers to all of the events, or look at list ordered in proximity to their location. Once they find an event, they can check it out and have the time of their life!

**Front-End Process**

For the home page, we had our logo and app title at the top. We customized the color scheme in the .xml files to be fluent throughout the entire application. In addition, we have three buttons: Map, List, and Create an Event. Each button is connected to the back end portion of our code and together, they start the intents that link the user to each activity page.

For the Create an Event page, we have multiple Text Edit layout, allowing the user to type in various information about their event (ex. name, location, time, etc.). When the user taps the ‘Create Event’ button at the bottom of the page, all of the inputted event information would potentially be sent to and stored in the application database. Each stored event would then be accessed in both the List and Map pages.

When a user opens the Map page, the map automatically zooms in on the location where you last had it open. The map would then accesses the database, and add each event as a marker on the Map. The user would have the ability to tap each marker event, and a small box would pop up with only the Event Name and Event Location. To find more information about the event, the user would click the ‘More Information’ button, and would taken to the Event Information page of our application. This page would include all of the information stored in the database for the event.

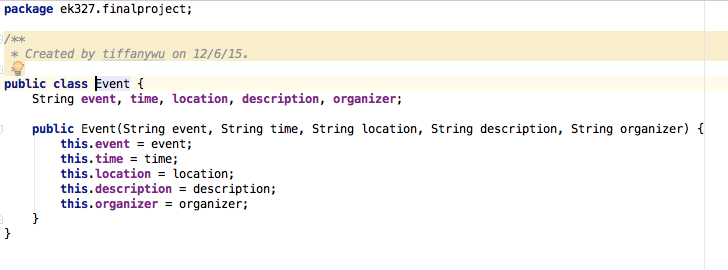
In the List page, the application would automatically accesses their current location as well as the data base. The page would list each event in order of proximity to their location. In the List view, the user would only see the Event Name and Event Location. When they click on the event, they would again be taken to the Event Info page with a full view of the event’s information.

The code behind all of these intents, connections and implementations will be explained in the Back-End Process of this report.

**Back-End Process:**

For the home page, we set up the buttons that connect to the front end buttons. We then made functions for all of the intents (we imported android.content.Intent). Basically, when the user clicks a button, it calls the corresponding function and starts the intent for the matching activity.

For the create an event page, the text edit portions are connected to variables here that saves this information. This page then connects with our server in order to connect with our database. We have php files for both create an event and get event data that theoretically should have respectively made a new entry in our database and show all the entries. The code below shows how we saved all the edit texts into variables and then called the Event class to make the event when we click the button. The Event class has a default constructor that puts fields into the event object including the name of the event, the time, the location, etc..



We had two php files. The first one, create\_new\_event connected to mySQL and used the data sent to the server to create a new database entry. It used the POST method to put the data onto the database. It then prepared, binded, and executed the information.



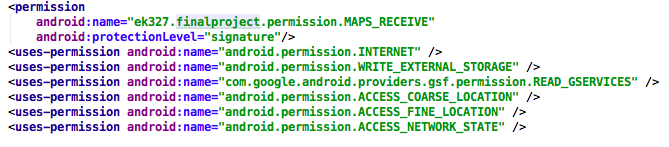
The second one, get\_event\_data connected to mySQL and would take the data from the database. It used fetch to get the information. We also had a class called ServerRequests that connects Android Studio to our server. However, due to other complications, we were not able to test it fully to see if it successfully connects to our server.

For eventLocalStore, we used this class to locally store the Event() with all the parameters but did not get a chance to call it yet since our php for our database was not working.

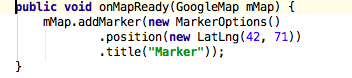
For List, we wanted this class to display all the events. Theoretically, it should use the POST method to get the database information from our server and then display all events. Then, we would have added a scrolling feature in order for the user to see all the events

For information, we were going to use this class to get the database information from the server and if the user would have clicked on the marker in map, the event’s information should have popped up.

In order to allow the application access to the user’s current location, access must be granted by the user. This is initialized in the <AndroidManifest.xml>, as shown below:



When the Map activity is opened from the Home page, the Google map fragment should add a marker to the latitude longitude coordinates (42,71), as shown below:



However, we had trouble successfully implementing the marker feature. If we had found a way to correctly configure this, we could have created Latitude and Longitude variables, accessed the positions of each Event in the database, and created new markers for each Event based off of these variables.

In order to implement a Map into our application, we needed to obtain a Google API Key. We created an account and obtained a key through the Google Developers Console. After enabling the Google Maps Android API, we added our key to the <AndroidManifest.xml> file in our Android App project.



This allowed us to build a customizable Map, with access to Google Map databases.

**Challenges Faced**

Our group faced multiple challenges when creating this app. To start, seeing as the app is for android phones and the language is in java, we all had initial issues with exactly how the language worked, and how to make code of our own in this language. Once we got past these issues and started coding for the app we intended to make, we inevitably came across errors within our code; some errors being harder to solve than the others. The biggest challenges we had in our app is accessing google maps and the code behind it, as well as getting a database to work.

For several days we struggled to make the map work. Moreover, whenever we thought we made improvements to our map code, and there were no errors when we ran it, we couldn’t check if the code worked since google play doesn’t run on the android emulator. We eventually were able to get the map to work, but not to its best ability. We intended for the app to have certain qualities such as markers of the events, current location, and other aspects. In some respects, the map failed to match these goals of ours due to compile errors. In other words, we had errors that we could not solve, and thus had to ditch some of the qualities we wanted our map to have at the very least a functioning map.

The biggest challenge we faced and were never able to completely solve was our apps usage of the database. Perhaps our idea was too ambitious to get the database working because we had little knowledge in using databases. Our first issue we came across when trying to access a database was finding a server that didn’t cost money and was functional. After this long hunt, we started having to write php files to connect to the database. This was and is our biggest issue to date because we were unable to get the php files working. The php files are in a different language, which was tough, and gave us many errors that we were unable to comprehend or solve. Moreover, another challenge we faced was getting android studio to connect to the server/ php files. This was a challenge itself, and since the php files weren’t functional anyways, the connection between android studio and the database failed. Since this connection failed, we were unable to send events that a person created to the database or retrieve the events from the database either.

Though our app was not entirely functional, this project was rewarding and taught us a lot about android studio, java, and coding in general. Perhaps the biggest takeaway from this project is to not take databases lightly. We certainly learned the hard way but it is good to know for the future that databases take extensive prior knowledge that we should teach ourselves. This project also allowed us to improve our github skills extremely and knowledge for github in general. Not only did we learn more about github, but we also learned a lot about android studio, and how to approach creating an app.