Coinz - Project Report

Szymon Filipiak s1651272

December 2018

Chapter 1

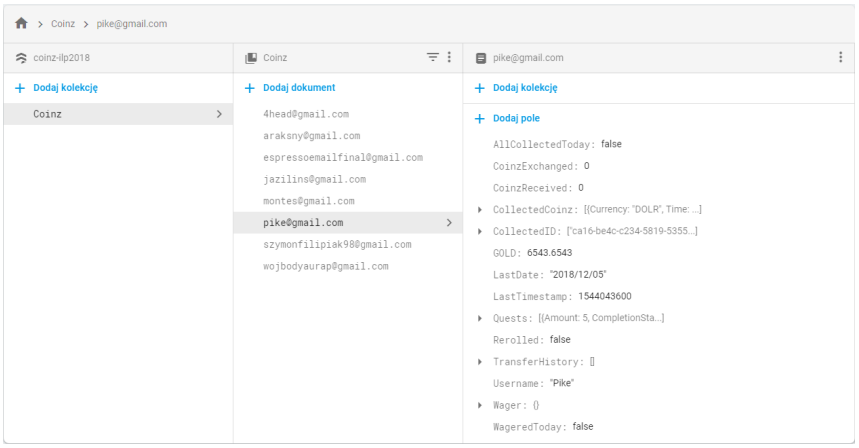
Core Algorithms and Data Structures

1.1 Downloading and Parsing of the Geo-JSON map

For downloading the map, I have used an asynchronous task as presented in the lectures. In order to parse the map to Kotlin data structures I have used mapbox's geojson library as well as json's JSONObject. I had to use these two, because mapbox's geojson package was conveniently extracting all the necessary marker's information but I was unable to retrieve the Coinz's exchange rates with it. That's what I have used json's JSONObject for. I have also decided to store the downloaded map as a string on a user's device, and only download the new one if the date has changed. This approach has two benefits. The game will load faster as well as it will use less data from user's mobile data plan.

1.2 Storing User's Data

I have decided to not store any of the user's data on the actual device, except for the above. All of the necessary information is stored in Cloud Firestore in a convenient format. It means that the game requires internet connection in order to function properly, but I think that isn't that big of a deal since it's 21st century as well as it is a multiplayer game. After user creates their account, the app creates a default document in Cloud Firestore. The corresponding document is identified by their e-mail address. The app is also downloading and uploading user's data to the database on almost every action of the user such as collecting exchanging, transferring a coin, as well as when user decides to open any other activity, for example: Ranking Screen or Transfer History. All of it to keep track the users' progress as well as to always provide them current information.



- AllCollectedToday - A Boolean. It represents whether a player has collected all Coinz on a given day. This value isn't necessary because it can be evaluated using CollectedID (see below). I have decided to use it anyway because this way the code looks nicer and more readable. Resets to false on a new day.
- CoinzExchanged - An Int. It represents the number of Coinz player has exchanged for gold on a given day. Resets to 0 on a new day.
- CoinzReceived - An Int. It represents the number of Coinz player has received from other players on a given day. Resets to 0 on a new day.
- CollectedCoinz - A List of HashMaps. Each HashMap contains information about the value and currency of a given coin as well as time when coin was collected, in form of a server's Timestamp. I am using this collection to display collected coinz in the Bank.

- CollectedID - A List. It contains the IDs of collected Coinz on a given day. I am using this to correctly display all the uncollected coin markers on the map. This List is independent of CollectedCoinz because it resets to an empty list on a new day.
- GOLD - A Double. It represent the amount of GOLD a player has collected throughout the entire course of the game.
- LastDate - A String. It represents the last recorded date when the player's document was updated in YYYY/MM/DD format. I am using it to tell whether some of the user's information should be reset.
- LastTimestamp - A Long. Similiar to LastDate, it represent the last recorded server Timestamp when the player's document was updated. I am also using it to tell whether some of the user's information should be reset.
- Quests - A List of HashMaps. Each HashMap contatins information about a quest. Each quest consists of:
  - Amount - Amount of Coinz player needs to collect in order to complete the quest.
  - CompletionStage - Amount of Coinz player has already collected.
  - Currency - Currency of Coinz player needs to collect.
  - Reward - Reward a player will receive for completing the quest
- Rerolled - A Boolean. It represents whether a player has already rerolled one of their quests on a given day. Resets to false on a new day.
- Transfer History - A List of HashMaps. Each HashMap contains information about amount of GOLD a player has received from other players and from whom they received it.
- Username - A String. It represents a player's display name, they have chosen when creating an account.
- Wager - A HashMap. It contains all the necessary information about player's bet against the clock in race mode, such as:
  - Amount - Amount of Coinz player needs to collect in a certain amount of time.
  - CompletionStage - Amount of Coinz player has already collected.
  - Time - Amount of time player has to complete the challenge.
  - Start - Server Timestamp representing when the wager has started. I am using it to calculate the remaining time.
  - Reward - Amount of GOLD player will win or lose depending on the outcome of the wager.

- WageredToday - A Boolean. It represents whether a player has already wagered their gold in the race mode on a given day. Resets to false on a new day.

1.3 Nearby coin detection

For this problem, I have decided to calculate the distance between a player and each displayed coin marker on each location change. To calculate the distance I have used Haversine Formula which is oftenly used in navigation:

$$d = 2Rarcsin(\sqrt{hav(\theta_2 - \theta_1) + cos(\theta_1)cos(\theta_2)hav(\phi_2 - \phi_1)})$$

where:

d - Distance between two points.
R - Radius of the earth
θ_1, θ_2 - Latitude of point 1 and 2 respectively
ϕ_1, ϕ_2 - Longitude of point 1 and 2 respectively
$hav(\theta) = \frac{1-cos(\theta)}{2}$

For evaluation of mathematical functions such as *arcsin* and *cos* I have used standard java.lang.Math library.

This setup may seem like it isn't the most optimal, because I am calculating distance between every displayed coin and player on each change of location. This sounds a little bit redundant, but on the other hand I am performing at most 50 simple calculations which is a matter of milliseconds and uses very little to none resources. Therefore I believe it just isn't worth the effort, to come up with a more sophisticated solution.

1.4 Transferring Coinz

At first, I wanted users to be able to transfer Coinz to the other players by typing in their username. But at the same time, I didn't want to enforce the uniqueness of them, because there is nothing worse than registering to a new game and seeing that your favourite nickname is already taken. I tried to come up with a good solution, but eventually when I found one, I was unable to efficiently implement it using Firebase. That's why I decided to use player's email instead for transferring coinz. That's also the major reason why I am using e-mails as identifiers of documents in Firestore.

1.5 Other functionalities

For all of the other features I have used provided libraries, either by Kotlin, Google or mapbox. Also I haven't used any special algorithms or data structures - other than already mentioned - to run my app.

Chapter 2

Parts of the design that have not been realised

2.1 Game Difficulty

At first the idea of various game difficulties seemed great, but the more I thought about it, the more questions it was raising. For example: What if a player decides to change their game's difficulty level? Should they be penalized for changing from easier to harder difficulties? How big of an advantage would they gain if they were not penalized? What to do about the ranking system? And many, many more. That's why I have decided to abandon this idea, and maybe the future development team will decide to implement it. The good news are that all the necessary tools are already given in my code.

2.2 Sign Up with Facebook

For some reason I was having problems with getting the Facebook developer's API key. Because of that I have decided to swap Facebook with Google, and create an option for Signing up with Google accounts.

2.3 Sharing Location with other players

I have decided to abandon this idea because I was afraid how secure it would be. Nobody would want their location to be shared with somehow they don't know. I have also had some troubles coding it in a way that would make it consistent. At first glance my implementation seemed very good but on emulators the game was lagging a lot, and I am not sure whether it was because I was running two emulators at the same time on my machine or was it something wrong with my code.

Chapter 3

Additions and changes to the initial design

3.1 Transfer History

I have added this feature because I needed a way to nicely display to a user from whom and how much gold he received. I have not planned it at first but I think it's a very natural thing to do. I believe it is a very convenient addition many players will appreciate

3.2 Drawing routes to coinz

I have not planned to add this feature at first because it was my first time working with Mapbox SDK. But then I have realised how easy it is and I would be angry at myself if I have not implemented it. So now, when you tap on a coinz marker you get a nice drawing of a route to your very desired coinz.

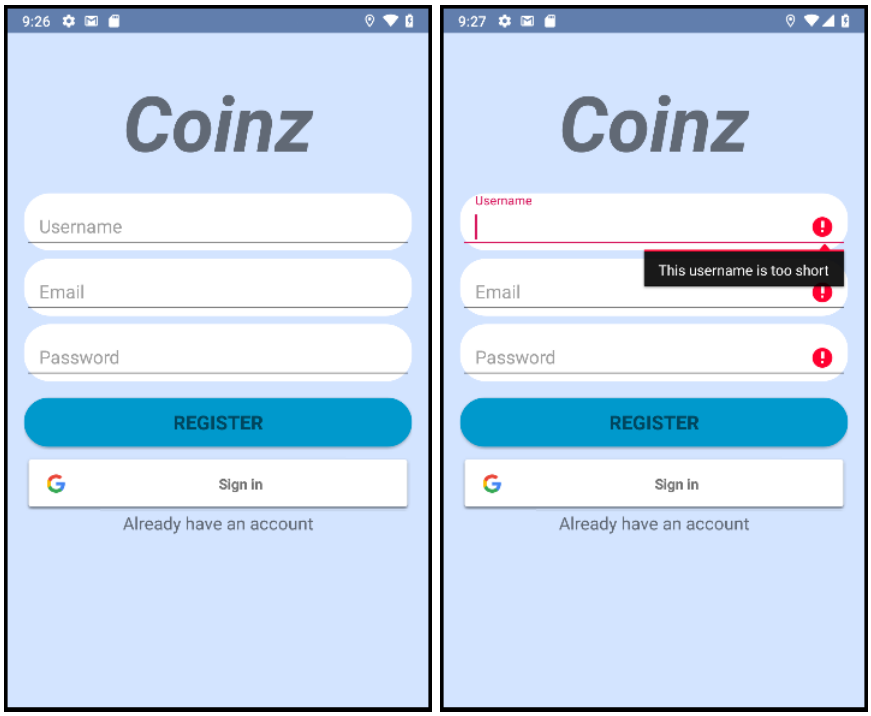
3.3 User Interface

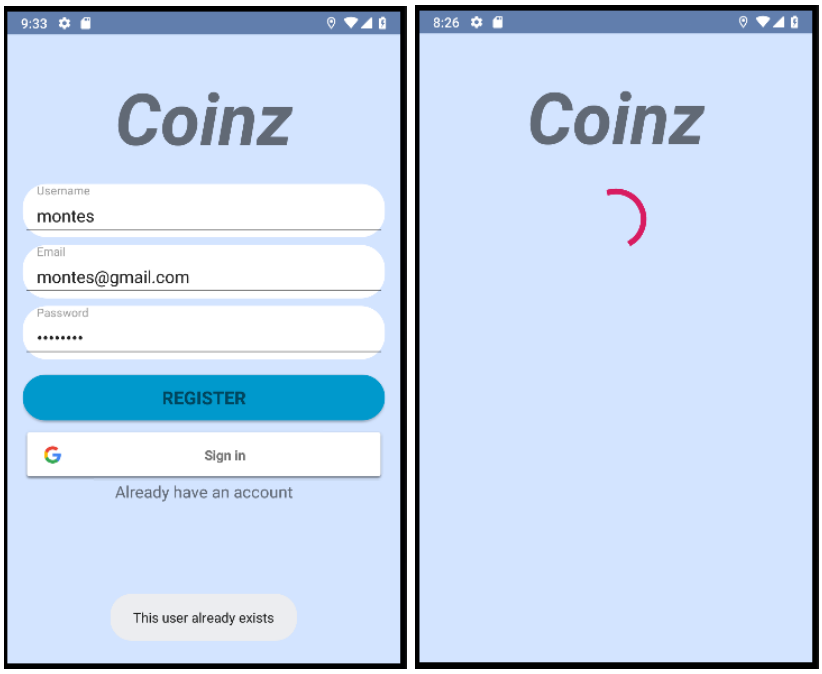
I have decided to completely change my user interface design to something even more minimalistic but yet cleaner and nicer. Now the entire screen is used by the map except for a small floating action button at the bottom. When pressed it shows other buttons that will direct player to another screens. I believe it looks great and overall I am very happy with the result.

Chapter 4

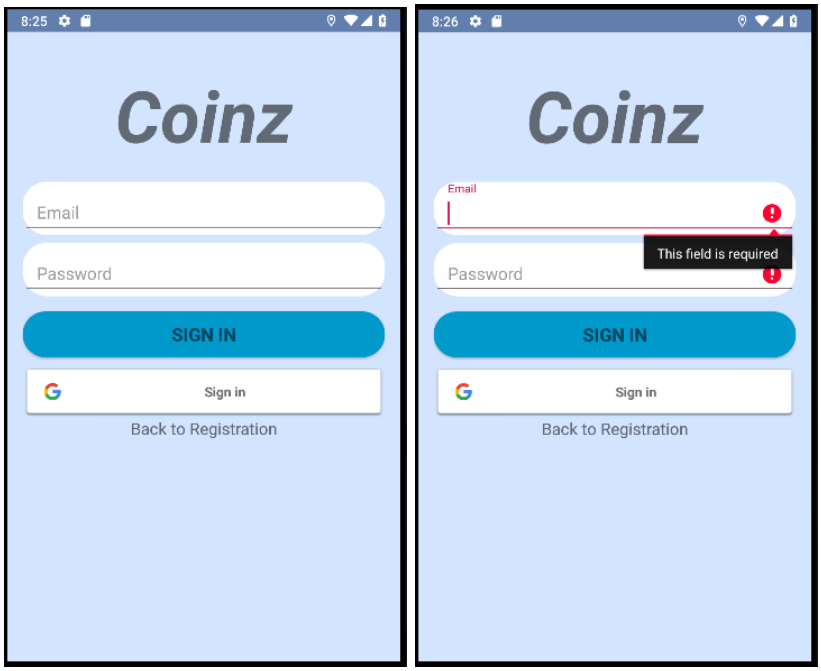
Screenshots

4.1 Sign Up Screen

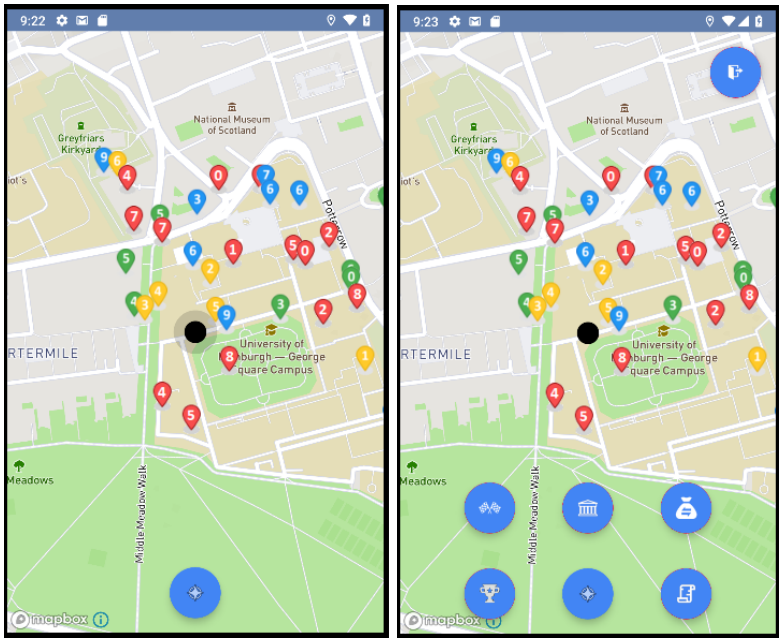




4.2 Login Screen

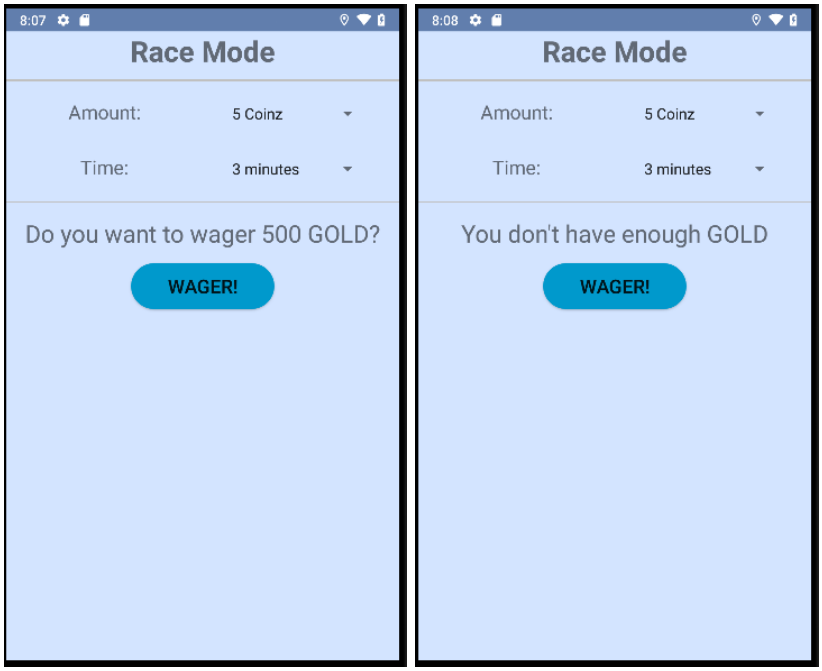


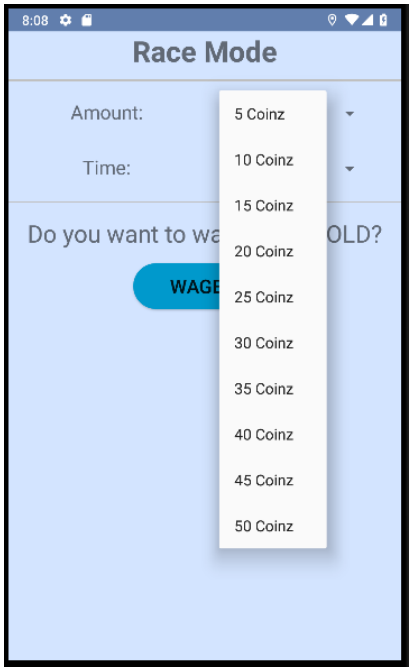
4.3 Main Screen

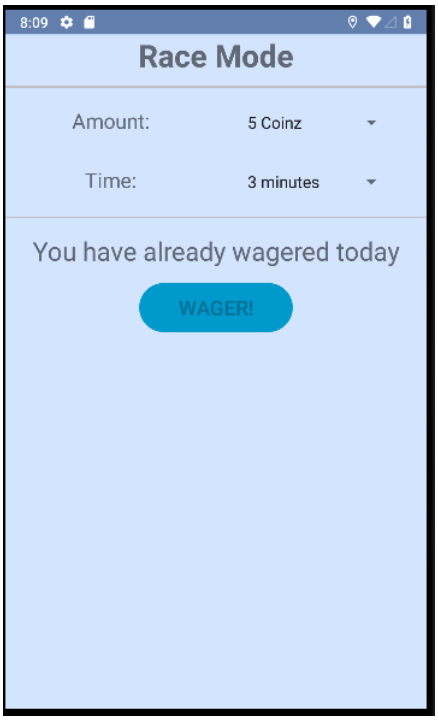


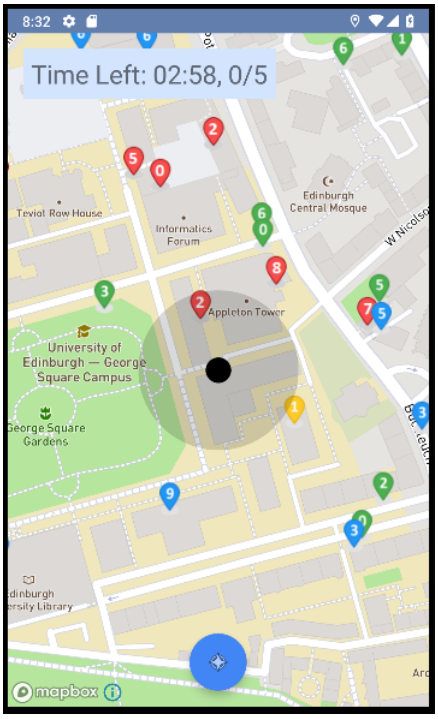
4.4 Ranking Screen

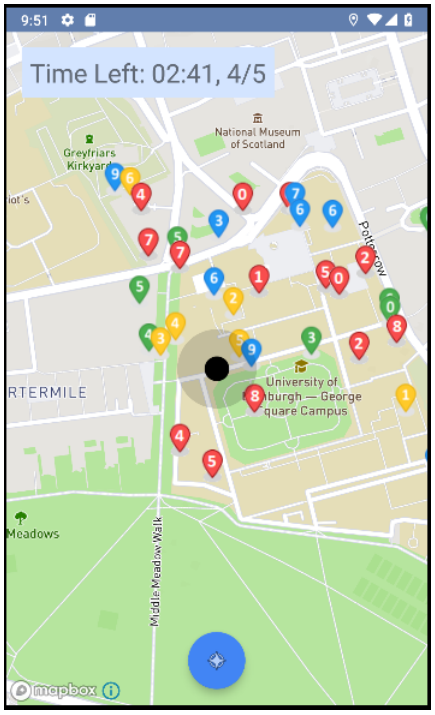
4.5 Race Mode Screen

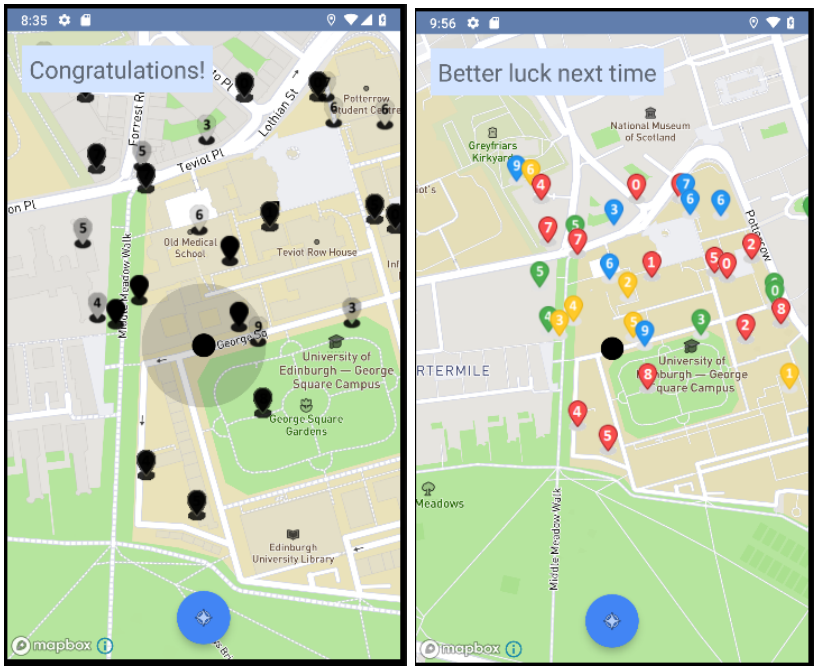




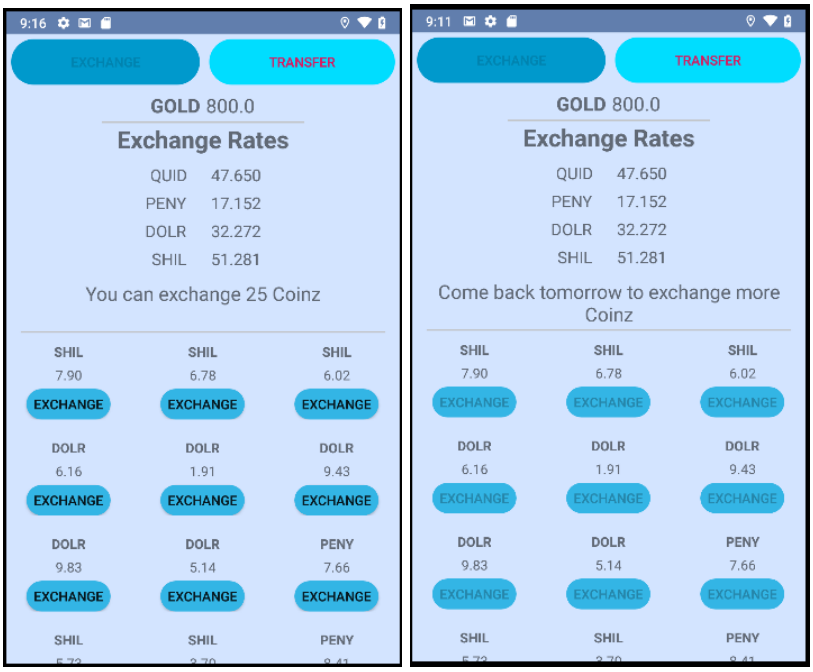

8:09
Race Mode
Amount:
5 Coinz
Time:
3 minutes
You have already wagered today
WAGER!

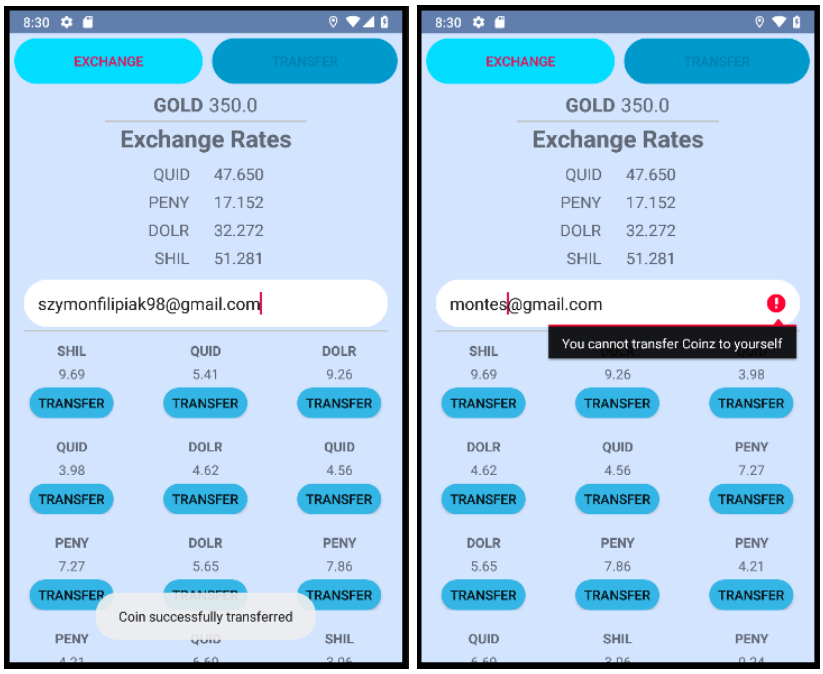

8:32
Time Left: 02:58, 0/5
Teviot Row House
Informatics Forum
Edinburgh Central Mosque
Appleton Tower
University of Edinburgh — George Square Campus
George Square Gardens
mapbox


9:51
Time Left: 02:41, 4/5
National Museum of Scotland
Greyfriars Kirkyard
Meadows
Middle Meadow Walk
mapbox

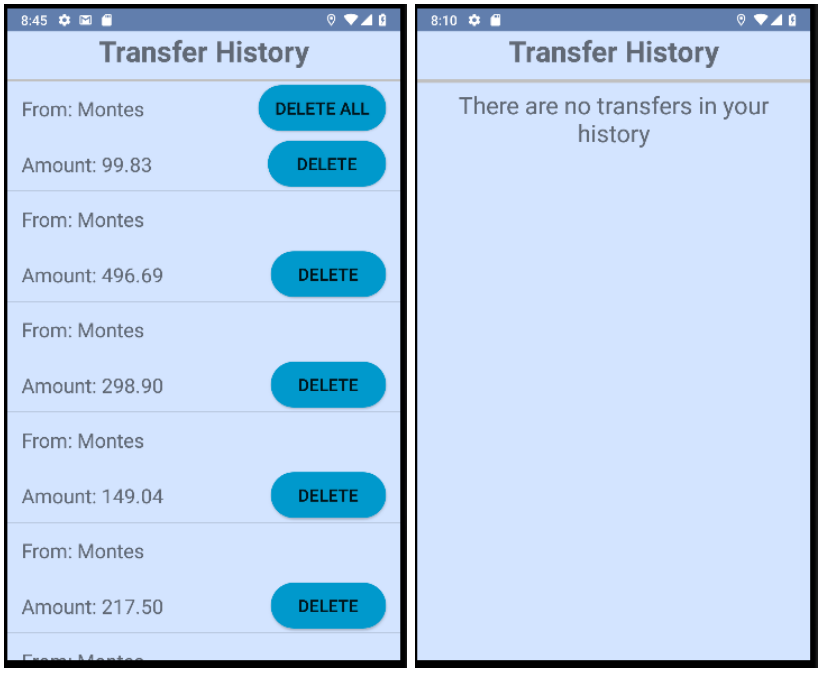

8:35
Congratulations!
Lothian St
Teviot Pl
Old Medical School
Teviot Row House
University of Edinburgh — George Square Campus
George Square Gardens
Edinburgh University Library
mapbox
9:56
Better luck next time
National Museum of Scotland
Greyfriars Kirkyard
Meadows
Middle Meadow Walk
mapbox

4.6 Bank screen

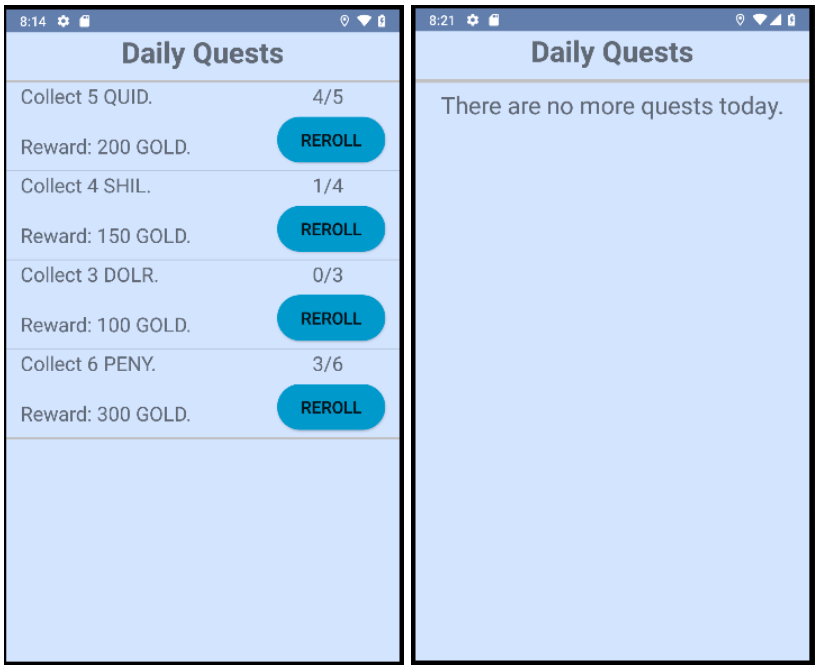


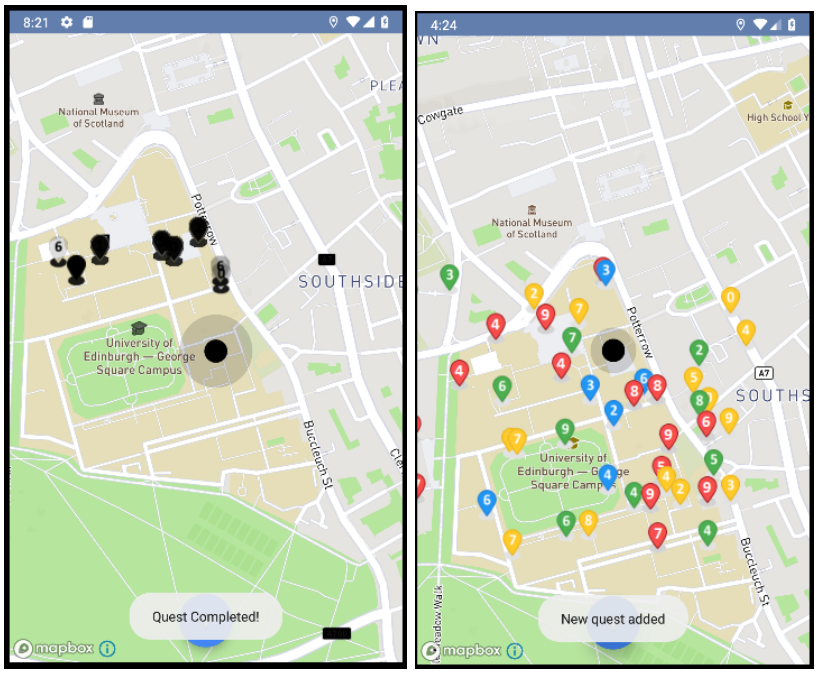


4.7 Transfer History Screen



4.8 Quests Screen





Chapter 5

Acknowledgements

- Login and Sign Up Activities - Some of the XML code and function that displays the red loading circle was automatically generated by Android Studio.
- Authentication with Google account - I have followed this guide on Google's developer site:
  - Google Sign-In for Android

  As well as this YouTube tutorial:
  - Firebase Google Authentication using Kotlin
- Mapbox Map and Routing - I have followed tutorials posted on YouTube from these two channels:
  - Devslopes
  - Programming Experts
- Mapbox Markers - I have followed guidelines given in mapbox SDK for Android:
  - Maps SDK for Android
- Creating User Interface in XML - I have followed tutorials posted on YouTube on this channel:
  - Lets Build That App
- FirebaseAuth and FirebaseFirestore - I have used very clear guidelines in Google Firebase documentation:
  - Firebase Authentication
  - Firebase Firestore

- A part of code in MainActivity that causes the phone to vibrate has been copy pasted from some source, but I am unable to tell where it came from.
- Icons and Markers - Icons used for floating action buttons as well as launcher's icon have been downloaded from Google Graphic's (unfortunately I wasn't able to track where they exactly came from) and edited to suit the design of the app. Markers icons are based on the initial mapbox's red marker, also edited by myself.
- Espresso Custom Matcher - A matcher that allows me to check the size of a ListView has been copy pasted from:
  - medium.com

5.1 Testing

I have tested my game on Pixel 2 with Android 9.0 API 28 Revision 6. Android Emulator 28.0.16, Android SDK Platform-tools 28.0.1, Android SDK Tools 26.1.1, Intel x86 Emulator Accelerator 7.3.2,