IMPLEMENTATION

Informatics Large Practical

Abstract

A document containing implementation details about Coinz. Coinz is a map based game where players have to collect cryptocurrency coins.

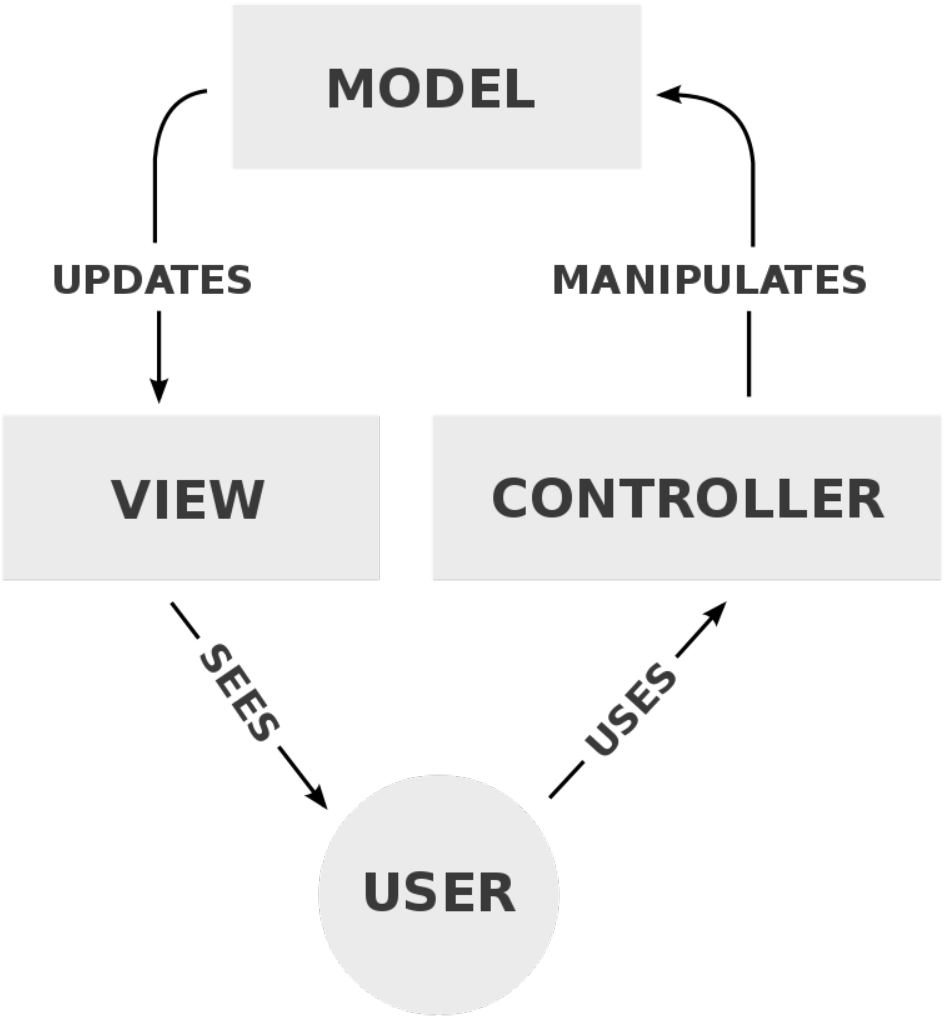
Raees Aamir

Matriculation number: s1617910

Table of Contents

Algorithms/Data Structures and Architectural Details

Coinz makes use of the MVC paradigm. MVC is an acronym for model-view-controller; it is an architectural pattern commonly used for developing user interfaces that divides the application into three connected parts. [2]



The views are XML layout files, the controllers are the activity class files named with a 'controller' suffix and the models are plain java classes that represent features of the game.

The code makes use of fragments in all the subsections apart from login and registration. The reason why I decided to use fragments is because it allows for easy navigation between sections of the app. A bottom navigation bar is used to hold the sections that you can transition to.

Any subsections of the app that require an internet connection will show the user an error if there is no internet connection available. They get presented with a dialog telling them that no connection is available and that they should come back later.

Share

The share system uses the Facebook Sharing API to let players share their achievements with their friends on social media.

There are no interesting data structures in this section. However, the installation of the Facebook Sharing API requires the use of a security algorithm so I will go into some detail about the installation procedure. The first involved adding the dependencies to the module Gradle build file:

```
implementation 'com.facebook.android:facebook-share:[4,5)'
```

The second step is adding a Facebook App ID to the Android Manifest. [1]

The third step requires a unique key hash. I did this by generating a SHA1 key for the app's signature. Java has a very useful inbuilt security framework which lets me do this without having to manually implement the SHA1 key hashing algorithm.

```
final MessageDigest md = MessageDigest.getInstance("SHA");
md.update(signature.toByteArray());
final String hashKey = new String(Base64.encode(md.digest(), 0));
Log.i("AppLog", "key:" + hashKey + "=");
```

The final step involved creating an empty activity called FacebookActivity. This is needed for the Facebook share button to segue to Facebook.

Menu

Menu items are defined in an Enum called MenuItem. MenuItem takes a class reference to the controller that manages the navigation for the particular subsection of the app. Menu items are loaded into a list view that is referenced in MenuFragment. An ImmutableSet is used to make all the items from the Enum iterable. In particular, an ImmutableSet is used because we don't want to let the developer add/remove/edit/delete MenuItem instances outside of the MenuItem Enum.

Login

Both the login and registration controllers are subclasses of an abstract authentication controller. The authentication controller handles the UI code that is common to both the login and registration, e.g. the progress spinner, the error message alerts and the on click listeners.

Before the Firebase login is called, appropriate checks are done to make sure the email address and the password are valid. The criteria for determining a valid email address is whether the email contains an "@" and the criteria for determining a valid password is whether the password has atleast 6 characters. If the checks pass then the Firebase method signInWithEmailAndPassword is called.

(Note that firebaseAuth in the below code is an instance of FirebaseAuth)

```
firebaseAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener((@NonNull
Task<AuthResult> task) -> {
...
});
```

Registration

The registration form has four fields: display name, email, password and confirm passwords. Before the appropriate firebaseAuth method is called there are checks to make sure the password and confirm password fields are equal. If they are then the Firebase registration is carried out with the code below:

```
firebaseAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener((@NonNull Task<AuthResult> task) -> {
...
}
```

Game

The game section of the app is the most complex so I will divide the explanation into three sections.

Downloading & Parsing

Downloading

The dependency OkHttp is used to download files from the internet. I created an abstract class called DownloadFileTask which is a subclass of DownloadFileTask<String, Void, T>. T is a generic type which is defined in the class definition as anything that is an object. In Java that means everything apart from primitive types. DownloadFileTask has two methods:

```
protected abstract T readStream(String inputStream);
protected T doInBackground(String... params);
```

The method doInBackground downloads the file from the internet. The URL to the file is found at index 0 of params.

```
Request.Builder builder = new Request.Builder();
builder.url(params[0]);
Request request = builder.build();

try {
    Response response = client.newCall(request).execute();
    return readStream(Objects.requireNonNull(response.body()).string());
} catch (Exception e) {
    e.printStackTrace();
}
```

readStream is returned callback when the download is completed.

Parsing

A dependency called Gson is used to parse the JSON string into a Java object. A anonymous class of DownloadFileTask is created and it returns a one liner which handles the JSON decoding:

```
return new Gson().fromJson(json, FeatureCollection.class);
```

Persistent Storage

After parsing the feature collection JSON data, it is stored in the app's shared preferences and an instance of a Java class called FeatureCollection is used to manipulate the data. When a player collects a coin from the map, the coin is removed from today's instance of the player's feature collection.

```
Feature[] features = featureCollection.getFeatures();
features[indexOfFeature] = null;
```

The FeatureCollection Java class is serialized back into JSON data using the Gson dependency.

```
String jSONDocument = gson.toJson(featureCollection);
String key = mUser.getUid() + "/" + dateFormatted;
preferences.edit().putString(key, jSONDocument).commit();
```

The persistent storage of the player's coins in the wallet is discussed further down in the Wallet section.

Detection of Coins

Every time a player moves a check is carried out to see if the player is within a 25m radius of a coin or a group of coins.

```
if (featureLatLng.distanceTo(playerLatLng) <= 25) {
   featureMap.put(i, feature);
}
```

where featureLatLng is the latitude and longitude of the marker and playerLatLng is the latitude and longitude of the player's current location.

If the player is within a 25m radius of a coin or group of coins, then the coins are added to the player's wallet. If the player has more than 25 coins in their wallet, then the coins are added to a spare change wallet that can only be used for trading.

Wallet

The wallet stores the coins the player collects from walking around the map. The wallet uses Firestore for data persistence as opposed to the app's local storage. The reasoning behind this decision is that the player may decide to play the game on multiple devices at different

times. The app gets the wallet data from Firestore by making a call to the collection reference

```
wallets.get().addOnCompleteListener((@NonNull Task<QuerySnapshot> task) -> {
...
}
```

Practically this is not the most efficient solution because it can take some time to establish a connection to the database to get the data. I noticed this and I decided to cache the player's wallet so that the app doesn't need to connect to the database each time the player wants to see their wallet.

When the loadWallet method is called, it will check if there's a cached wallet and if there is it will not attempt to make a connection to the database.

```
if (Wallets.getWallet() != null) {
    System.out.println("[Wallet] not null!");
    listener.onComplete(Wallets.getWallet());
    return;
}
```

The same logic is used for the bank but I must point out that the bank is implemented in the same area as the wallet. To access the bank, go to the menu and click "Wallet", and you will see a section that says "Total gold". Total gold represents how much gold is deposited into the player's bank. If you want to deposit a coin into a bank, tap the coin you want to deposit and the gold value will update in the bank.

Unrealized parts of my design

Distance recording

The idea of distance recording was that a player who travels to a coin marker by following the shortest path will receive a bonus of 10% of the coin's value. The path is defined as the route the player takes from their previous coin marker to the next coin marker. If the player hasn't collected a coin before, then the path is defined as the distance between the coin marker and the area at which the player logged into.

I decided not to implement this feature because it doesn't make much sense given the way the coins are distributed on the daily map. Coins are very closely coupled together and it is very possible that a player could collect three coins at one particular location. When I came up with this idea my understanding was that coins would not be so close together on the map, i.e. the player would have to walk a somewhat non-trivial distance to get to the next coin. In that case it makes sense to reward the player for following the shortest path, but if the distance between two coins is literally a few footsteps then it's pointless!

Weather based events

The idea of the weather based events was that a player can collect a coin when the player is close to a coin, but when they're not exactly at it. For example, if there's a coin at the KFC then the player will be able to collect the coin at the Tesco near Starbucks. If the player collects the coin at Tesco, the shortest path reward will still apply, except the shortest path will be calculated to Tesco instead of KFC.

In practicality implementing this feature would mean increasing the distanceTo bound on a rainy day. But given that coins are in a very close proximity to each other, it isn't worth it as explained above.

Firebase database structure

Cloud Firestore

Collections

Name	Purpose	Document Structure
Users	Stores instances of identifying information for registered users. Used to enable user querying.	• displayName: String • email: String • uid: String
Banks	Stores the central bank for each user	• coins: List • uid: String
Wallets	Stores the wallets for each user	• coins: List • date: String • userUid: String • walletType: Int • walletUid: String

Rules

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /Users/{anything=**} {
     allow read, write: if true;
    }
    match /Banks/{anything=**} {
     allow read, write: if true;
    }
    match /Wallets/{anything=**} {
     allow read, write: if true;
    }
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

Realtime Database

Collections

Name	Purpose	Document Structure
Coinz-12df3	Stores chat messages	• messageFromUser: String • messageText:

		String • messageTime: Long • messageToUser: String

Rules

```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Additional features that were not described in my design

Messaging

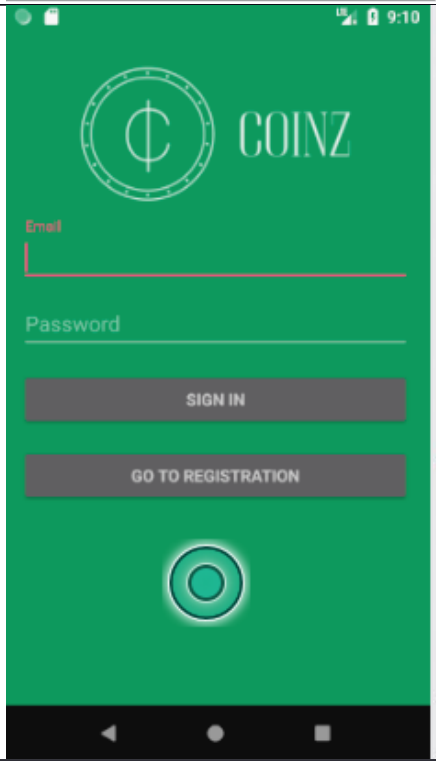
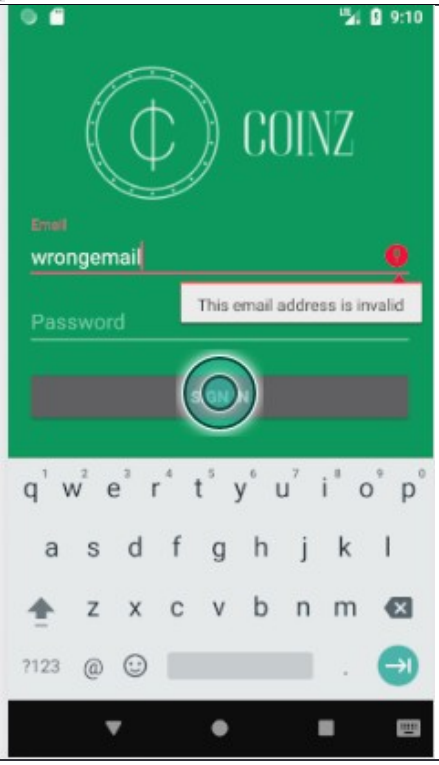
Trading was described in my design as a fundamental feature. I decided to expand on coin trading as a bonus feature. In doing this, I turned it into a fully fledged messaging system.

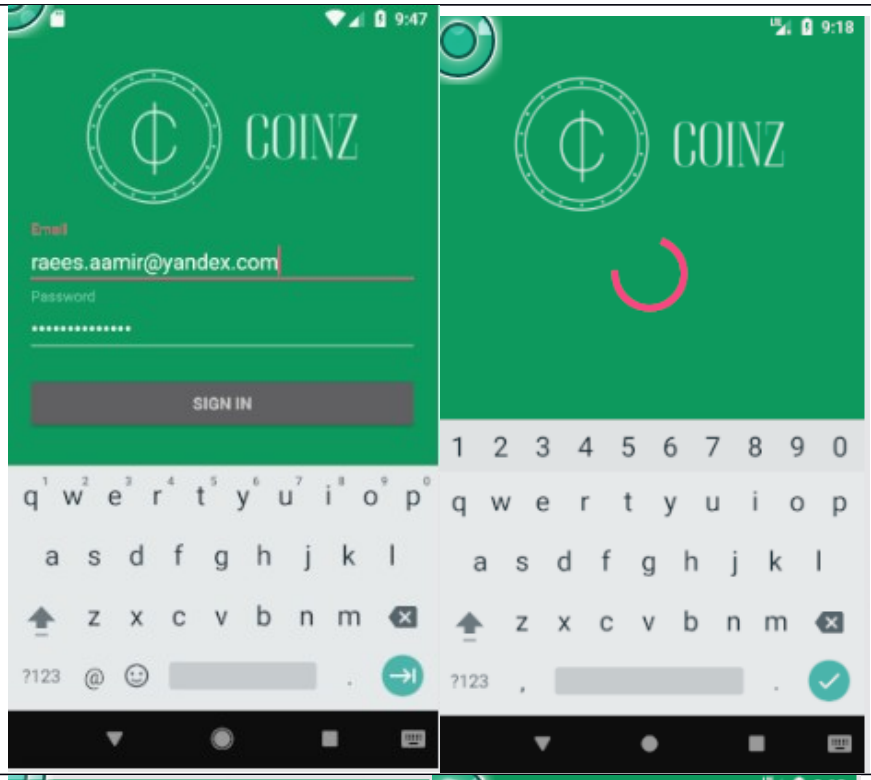
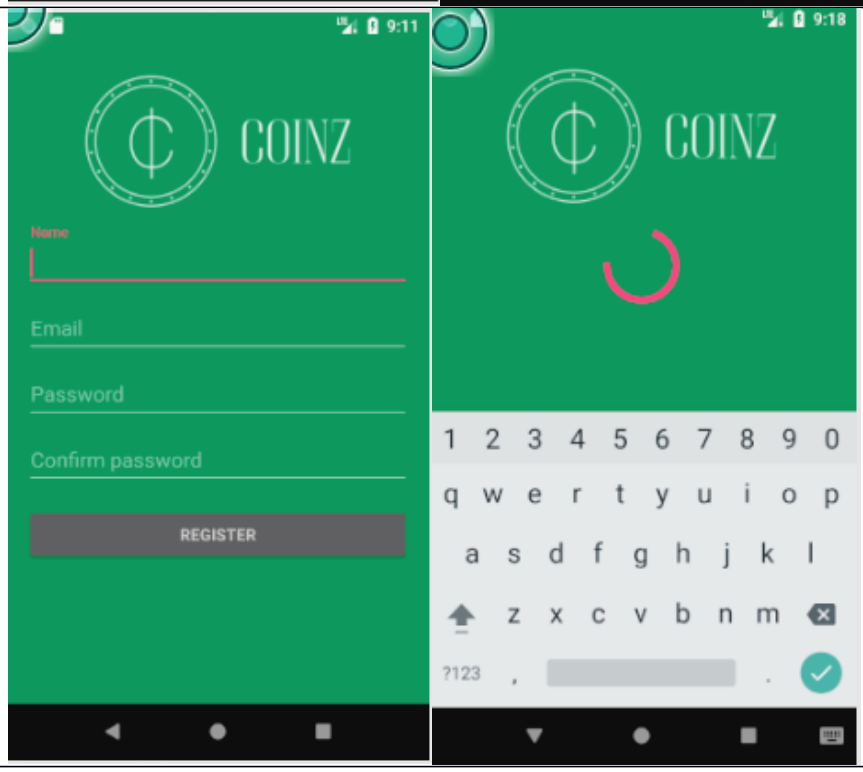
It is the case for most people that before you give them something, whether it be a virtual item or a real item, you will want to talk to them and have a discussion. For example, you might want to discuss what you want in return from the other player. The messaging system allows players to do this without having to use another app to have a conversation. It results in a more integrated gaming experience.

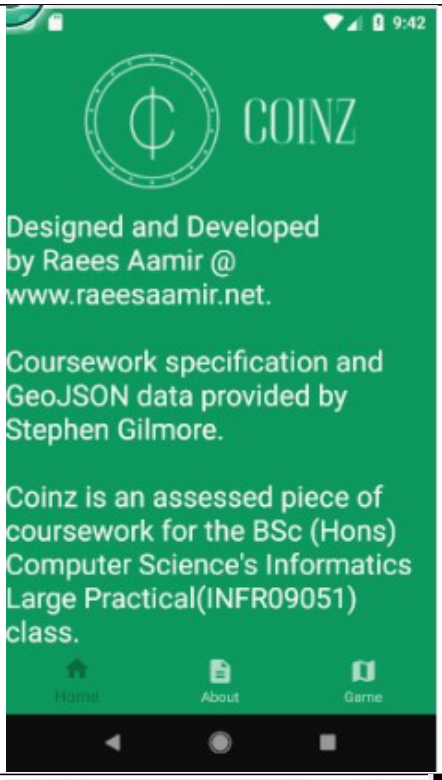
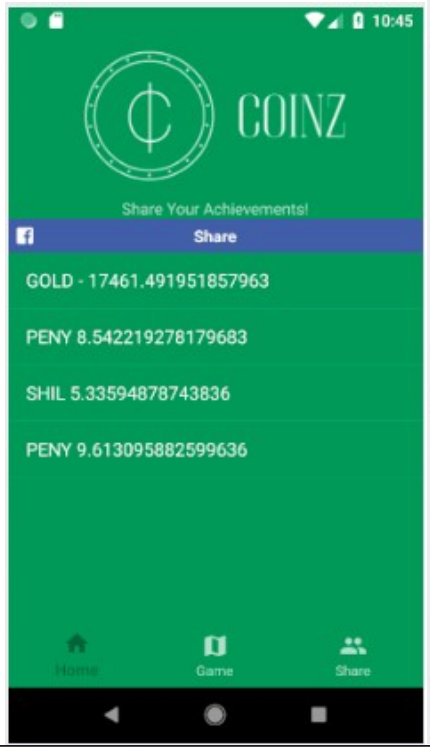
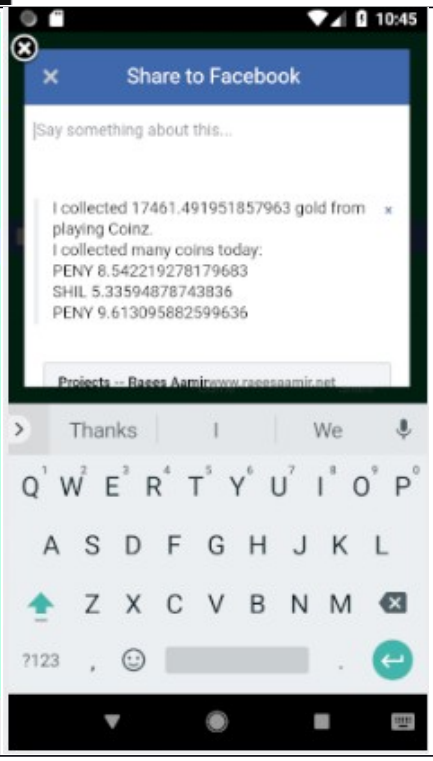
In the messaging environment, you can send message or you can transfer coins to the user you are talking to. If you have any coins in your spare change wallet, then you will be able to transfer them to the other user by pressing the ‘Trade’ button. Note that if you have not collected 25 coins you won’t have any spare change to trade.

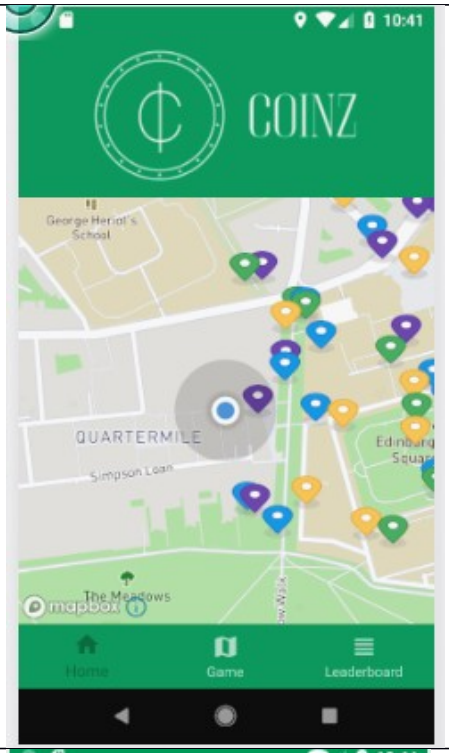
To message a player, you have to open the messaging system by clicking ‘Messaging’ in the menu. When the list is loaded you will be presented with a list of all registered users. You can then open a messaging environment by pressing the name of the user you want to message. However, this is not the best design but given that this is a prototype of a work in progress, there will be opportunities to improve this in the future. A better way to show the users would be to have a friends list and each friend would be displayed in the list. Strangers would not show in the user list and the status of each friend would be shown so that the user would know if the friend is online or offline.

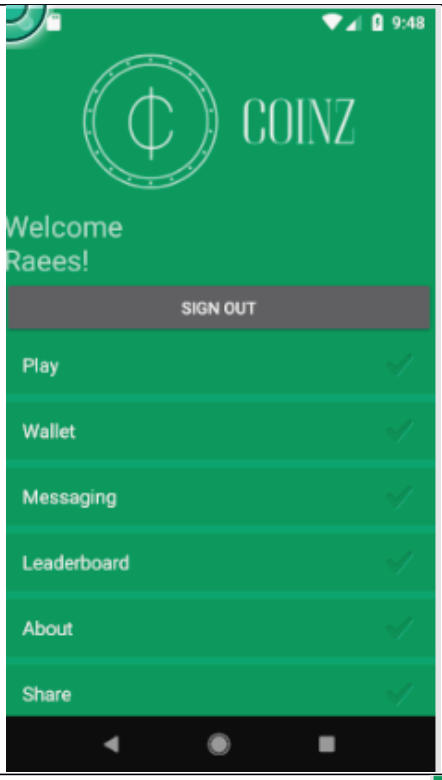
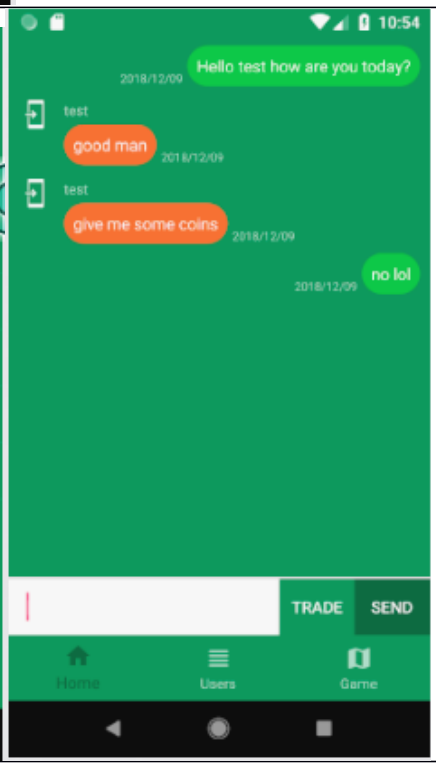
Screenshots

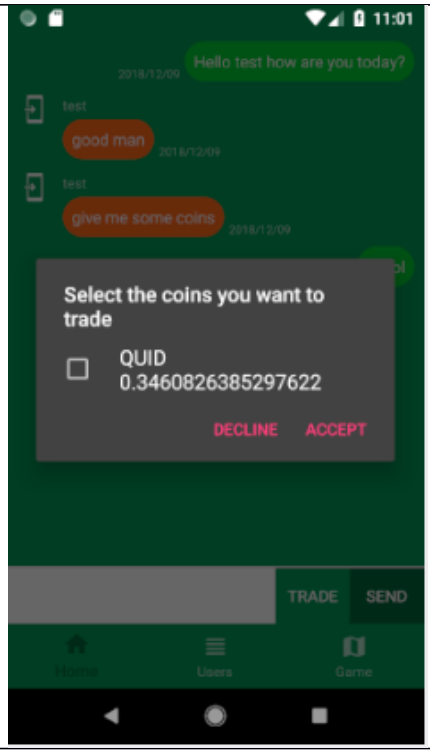
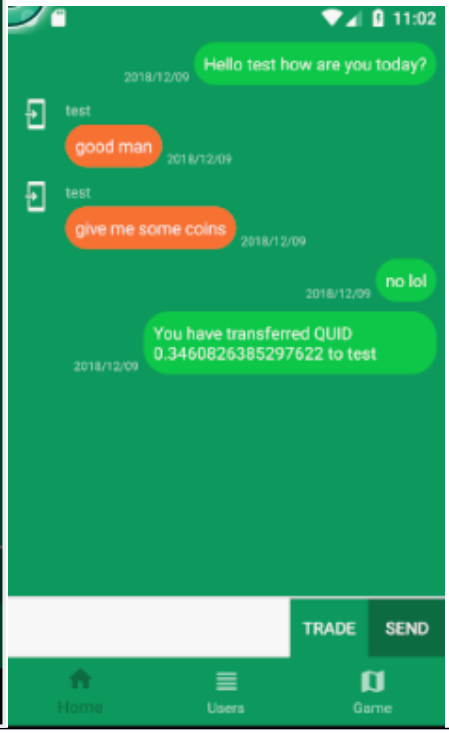
Subsection	Screenshot
Splash screen	
Login	 

More Login	
Registration	

About	 9:42 COINZ Designed and Developed by Raees Aamir @ www.raeesaamir.net. Coursework specification and GeoJSON data provided by Stephen Gilmore. Coinz is an assessed piece of coursework for the BSc (Hons) Computer Science's Informatics Large Practical(INFR09051) class. Home About Game
Share	 10:45 COINZ Share Your Achievements! Share GOLD - 17461.491951857963 PENY 8.542219278179683 SHIL 5.33594878743836 PENY 9.613095882599636 Home Game Share  10:45 Share to Facebook Say something about this... I collected 17461.491951857963 gold from playing Coinz. I collected many coins today: PENY 8.542219278179683 SHIL 5.33594878743836 PENY 9.613095882599636 Thanks I We

Play	 10:41 COINZ George Heriot's School QUARTERMILE Simpson Loan The Meadows mapbox Home Game Leaderboard	
Wallet	 10:44 COINZ TOTAL GOLD: 17461.491951857963 PENY 8.542219278179683 SHIL 5.33594878743836 PENY 9.613095882599636 Home Users Wallet	

Menu		
Messaging (the screenshot on the right is a chat with a person called test)		

More Messaging	 

Acknowledgements

Bug fixes I used:

Stackoverflow. (2018). Android app crashes when switching Fragment after showing a keyboard that is set with nextFocusDown, *Stackoverflow*. [online]. Available at: https://stackoverflow.com/questions/49745418/android-app-crashes-when-switching-fragment-after-showing-a-keyboard-that-is-set
[Accessed 12 Dec. 2018]

Stackoverflow. (2018). Android studio please select android sdk, *Stackoverflow*. [online]. Available at: https://stackoverflow.com/questions/34353220/android-studio-please-select-android-sdk
[Accessed 12 Dec. 2018]

Stackoverflow. (2018). Installation failed with message invalid file, *Stackoverflow*. [online]. Available at: https://stackoverflow.com/questions/42219784/installation-failed-with-message-invalid-file
[Accessed 12 Dec. 2018]

Code acknowledgements:

Codepath. (2018). Using OkHttp | CodePath Android Cliffnotes, Codepath. [online]
Available at: https://guides.codepath.com/android/Using-OkHttp
[Accessed 12 Dec. 2018]

Google. (2017). ImmutableCollectionsExplained google/guava Wiki, *Google*. [online]. Available at: https://github.com/google/guava/wiki/ImmutableCollectionsExplained
[Accessed 12 Dec. 2018]

Google. (2018). Perform Simple and Compound Firestore queries in Cloud Firestore, *Google*. [online]. Available at: https://firebase.google.com/docs/firestore/query-data/queries
[Accessed 12 Dec. 2018]

Coding In Flow. (2018). BottomNavigationView With Fragments – Android Studio Tutorial. *Youtube*. [online]. Available at: https://www.youtube.com/watch?v=tPV8xA7m-iw
[Accessed 12 Dec. 2018]

Stackoverflow. (2018). How to check Google Play services version, *Stackoverflow*. [online]. Available at: https://stackoverflow.com/questions/18737632/how-to-check-google-play-services-version. [Accessed 12 Dec. 2018]

Both bug fixes and code acknowledgements:

Stackoverflow. (2018). Android : A/libc: Fatal signal 11 (SIGSEGV), code 1, fault addr 0x8 in tid 18372, *Stackoverflow*. [online]. Available at: https://stackoverflow.com/questions/50602221/android-a-libc-fatal-signal-11-sigsegv-code-1-fault-addr-0x8-in-tid-18372
[Accessed 12 Dec. 2018]

Emulator Specification & Troubleshooting

Emulator

The device I'm using is Pixel; the original Google Pixel with the normal size (not the XL). When you click 'Create New Virtual Device', you will see a device definition named 'Pixel' and that is the one I'm talking about. The API version I'm using is Oreo, API 27.

Troubleshooting

Android Studio

If you get an error "Please select Android SDK" then go to File → Sync Project with Gradle Files.

If you get an error about Installation failed with message invalid file along the lines of "It is possible that this issue is resolved by uninstallating an existing version of the apk if it is presented, and then re-installing" then go to Build → Clean Project and after that Build → Build APK. After doing that the project should run fine.

Google Play Services

When logging in or registering, if you get an error message saying wait for Google Play Services to update then that means you need to try again after its updated. Google Play Services usually updates in the background but if the update doesn't work then you'll have to update Google Play Services manually.

Location dot not showing on map

If you're loading the app for the first time on a new emulator, you may notice that the location dot does not appear. That means you need to click the three dots on the bottom of the emulator's sidebar and you'll see that the location dot appears. This does not occur when the app is tested on a physical device.

References

[1] – Facebook. (2018). Getting Started Android SDK, Facebook. Available at: https://developers.facebook.com/docs/android/getting-started#app_id
[Accessed 12 Dec. 2018]

[2] – Wikipedia. (2018). Model-view-controller – Wikipedia. Available at: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [Accessed 12 Dec. 2018]