

Coinz

Infomatics Large Practical

Sean Mullan - s1627291

Coinz Game Specification

Coins are displayed on a map. Players move around the map (physically with their mobile phones), and if they move within 25 meters of a coin, the coin is collected. A new map is released every day, with each map having 50 coins. Each player has the chance to collect every coin. That is, if one player collects a coin, it is still available for all other players to collect on their devices.

Collectable coins are one of four (fictional) crypto-currencies: Penny, Shilling, Quid and Dollar (denoted PENY, SHIL, QUID and DOLR). They fluctuate daily with respect to a fifth cryptocurrency: GOLD. The collectable coins have a value of greater than zero and less than 10.

On the map, the coins are colour coded accordingly - PENY: Red, SHIL: Blue, QUID: Yellow, DOLR: Green. The approximation of the value of the coin is displayed on the coin marker (rounded down).

When coins are collected, they are saved to the players 'local wallet'. Their wallet might look something like this if they have collected four coins:

| Type | Value |
|------|---------|
| PENY | 9.34522 |
| DOLR | 1.12432 |
| DOLR | 8.91342 |
| QUID | 5.24221 |

Each player also has a bank account with the 'central bank'. Players can transfer a maximum of 25 collected coins into their account per day. The player can select which coins they want to transfer in. Once the coins are transferred into their bank account, they are automatically converted to GOLD. Players are encouraged to check the exchange rates for that day, and select a combination of coins which will give them the most amount of GOLD. Their bank account will show the history of coins paid in, along with the value (in GOLD) they were worth at the time. **The aim of the game is to maximise the amount of GOLD coins in your bank account.**

The remaining coins (i.e. 'spare change') will sit in a players local wallet until the end of the day. Until that time, the player is able to send their spare coins to their friends. Again, they can choose which coins they want to send. Once the selected coins are sent, they are removed from that players local wallet, and placed in their friends local wallet. Each players local wallet will be separated in two – spare change which they have collected, and spare change that they have received from friends. Players are allowed to transfer an unlimited amount of coins which they have received from friends into their bank account. This creates the opportunity to negotiate with fellow players, since **all** spare change will disappear at the end of the day.

Bonus Features

Race Against The Clock Bonus – Once a day, each player has the option to start a 120 second timer. Once the timer starts, all coins in the map double in value (only for that player!). The player then attempts to collect as many coins as possible during this period, after which the coins return to their normal value. The double value coins are collected and stored in the players local wallet in the exact same way as specified above. If the player decides to send the double value coins to a friend rather than transferring them to their own bank account, the double value is retained. This feature creates an exciting reward for the players, making the game much more interesting to play.

Leaderboard – A leaderboard will be displayed to show every player ranked in order of how many GOLD coins they have in their bank account - the player with the most amount of coins being at the top of the leaderboard. This adds competitiveness to the game, making it more interesting and encouraging players to play it more often or for longer sessions.

Implementation of Features

The above specification gives rise to a number of features. The features listed below can be viewed as the sub-tasks, or milestones, required to complete this project. For each feature, I have included a brief summary of how I intend to implement it. However, these features, along with their intended implementation, may be subject to change as the project progresses.

User accounts

The user will be able to create an account using their email address and a password, and log in with the account in the future. User emails and passwords will be stored on a Firebase Cloud Firestore database (under a parent node “Users”), and the application will make use of Firebase Authentication to ensure user data is securely stored and identified. The user will be able to achieve this through the applications Login Activity. When the user creates an account, they will be allocated a unique user ID in the database to ensure that they are uniquely identifiable.

Map data retrieval and rendering

When the app is booted up, it will check the current date, and date that is saved on the users internal storage. If the saved date is doesn't yet exist, then it writes the current date. If the current date is equal to the saved date, it retrieves map data from the users internal storage. Otherwise, it retrieves the map data from this Informatics server: <http://homepages.inf.ed.ac.uk/stg/coinz/>. The map data will then be saved to the devices internal storage (overwriting previously stored map data if it exists). As the user collects coins, they will be removed from the map data in the internal storage. This is to ensure that if the app is closed and reopened, the application does not lose information about which coins have been collected. The map will then be rendered with the map data stored on the internal storage using the MapBox API.

User location and coin collection

The application will use the Android LocationManager to determine the users position. The users position will be displayed on the map, along with the markers of the coins. The application will be able to determine if a user is within a 25 meter radius of any collectable coin, and subsequently collect the coin if this is the case. When a coin is collected, it will be removed from the map, and then placed in the 'collected' section of the users wallet.

Local wallet

The users local wallet will be divided in two – coins which the user has collected, and coins that they have received from friends. The collected coins will be stored on the devices internal storage, and the received coins will be stored on firebase. Upon bootup of the app, when the dates are being checked, the local wallet will be cleared if a new day has begun. That is, the collected coins in the internal storage will be deleted, and the received coins stored on firebase will be deleted. Otherwise, the application reads the data and the UI will then display the collected and received coins.

Friend adding and coin sending

A user can add a friend via their email address. When one user sends a friend request to another user via their email address, the other user will be prompted to accept or decline the request. This will be stored under a “Friends” node in the database, which will store all user ID’s, and each of the user ID’s will contain a list of friends that they have added, plus a list of friend requests. The UI can then display a users list of friends on their device, plus a list of friend requests. If a request is accepted, the requester will be added to the users friends list, and the user will be added to the requesters friends list.

Users can send coins to their friends – either coins that they have collected themselves or coins which they have received from another friend. When one user selects the coins they wish to send to a friend, these coins will be removed from the users local wallet (from either the collected section or the received section) and placed in the friends local wallet (in the received section).

Bank account

The database will contain a “Bank” node, under which all user ID’s will be stored. Each user ID will contain the value of GOLD in their account, plus a history of transactions. A transaction is defined as a collection of selected coins being transferred from a users local wallet to their bank account. Each transaction will detail when the transaction occurred, and the quantity of GOLD that the collection of coins added to the bank account.

This section of the application will handle the conversion of collectable coins to GOLD via the exchange rates, and will also display the exchange rates to the user. When a user wants to transfer coins from their local wallet into their bank account, the bank will first check the eligibility of the coins they have selected to transfer. If the coins are spare change received from a friend, then no further checks are needed – the user can transfer as much of these as they want. If the coins are spare change that have been collected by the user, is it at most 25 coins? If so, the bank will inform the user that they cannot transfer more than 25 coins.

Otherwise, the bank will take each of the selected coins in turn and convert them into GOLD using the corresponding exchange rate, and the quantity of GOLD coins will be updated in firebase. The coins will then be removed from the appropriate section of their local wallet. The bank will record the time and date of the transaction and the total value of GOLD that was added, and add this to the database.

There is no restriction on having the same coin being transferred into a users bank account twice. That is, in the case that a user collects a coin and transfers it into their bank account, meanwhile a friend collects the same coin and transfers it to the aforementioned user, that user is allowed to transfer the coin into their account.

Race against the clock

Upon bootup, when the dates are checked, if a new day has begun the app will enable a timer bonus button. The application will use a Timer class to measure the period of time (120 seconds) over which coins are doubled in value for. The game will be played as normal, but if a coin is collected while the Timer is active, its value will be doubled on the users device locally, and subsequently stored in the users local wallet. Once the timer is finished, coins return to their original value and the timer bonus button will be disabled (as the player can only use the bonus once per day).

Leaderboard

The leaderboard can display a 'global' leaderboard which will display every players score, and can also display a 'friends' leaderboard in which the user will only see their friends scores (score is just the amount of GOLD in a users bank account). The application will retrieve the data from the Bank section of the database, and subsequently display the appropriate list of users with the amount of GOLD coins they have in ranked order.

Language Choice

For this project, I have chosen to write the software in Java as opposed to Kotlin. Kotlin appears to be a great language that has gained a lot of support from the Android community recently. I think the main benefit in comparison to Java is its conciseness – Kotlin can reduce the amount of boilerplate code required in Java, which is relatively verbose compared to other languages. This decreases the opportunity for errors and bugs. Kotlin also has null in its type system, which mitigates the large null-unsafety problem with Java.

However, Kotlin is still relatively new in Android development. I believe it will grow to become very popular in the next couple of years, but for now the Kotlin development community is still small. Due to the fact that this project is a prototype which will be passed on to a team of developers who will maintain and develop it further, I think it is crucial to write this software in a language that the vast majority of development teams use – Java.

It would be difficult to find a team of Kotlin developers, and even more difficult to find a mentor for the team that is actually experienced with Kotlin. Most teams will be comprised of Java developers, and it would be a challenge to switch entire teams over to Kotlin. Furthermore, switching a team over to a new language increases the risk of errors and bugs appearing in code.

The Java development community is one of the largest in the world, boasting over 1.37 million questions on Stackoverflow. The Kotlin community pales in comparison, with around 8000 questions on Stackoverflow. I would like to have plenty of resources and a strong support network to ensure that my application is well-engineered, and I want the team of developers who will further develop my application to share this support.

Furthermore, applications built in Java are much more compact, as Kotlin has a bigger runtime size when compared to Java. Applications written in Java also benefit from accelerated assembly within Gradle, making the build process much quicker.

Timeline

Below is a brief timeline of when I hope to complete the above tasks. Throughout the development of this project, I will test the application continuously and thoroughly using J-Unit testing, and I will continuously document my progress in a report. I intend to consistently follow a coding style guide and continuously review and refactor my code.

Some tasks are larger than others, so I have tried to allocate a full week to larger tasks and a few days to smaller tasks. Each week is considered as the end of the week i.e. I want to complete each of these tasks by the end of the week specified.

| Week | Task |
|------|--|
| 2 | Create list of potential bonus features; begin dealing with underspecification; write out draft of Project Plan |
| 3 | Have complete specification of game design including two bonus features; complete Project Plan |
| 4 | Create required fields in firebase database. Implement login and sign up. Start implementation of map data retrieval and rendering |
| 5 | Complete map data retrieval and rendering, implement user location and coin collection |
| 6 | Implement UI and data retrieval for local wallet. Begin implementing friend adding functionality |
| 7 | Complete adding friends feature and implement ability to send coins between friends |
| 8 | Begin implementing functionality of central bank, including exchange rates, transferring from local wallet to bank account and storing transaction history |
| 9 | Complete full bank functionality and test thoroughly. Begin implementation of bonus features |
| 10 | Complete bonus features |
| 11 | Thoroughly test full application – write additional unit tests where required |
| 12 | Complete report |
| 13 | Review and submit |