Informatics Large Practical Project Plan

Design Decisions

App screens

- **Login/Register screen**

This will be the first screen that users see when they start the app for the first time. This will take user's details to create an account for them if they haven't got an account already. If they do already have an account, then it will present them the option to login to their existing account. This will enable users to keep their account with the game even if they lose/change their phone.

The only data that we will take from the user will be their email address and a password to use for the app. This will allow a future development team to implement password recovery, and for us to announce new versions of the app.

Transitions - After the user is successfully registered or logged in, it will redirect to the main screen listing coins.

- **Main screen listing coins**

This will be the screen that the user spends the most time on. It will show users (through a text interface) the distance to the nearest 4 coins, the coins' values, and the conversion rates for that day.

Across this screen and the remaining screens, you will be able to navigate between screens by using a menu icon that will appear in the top left corner.

- **Upgrade/trap inventory screen**

This screen will show users what traps and upgrades they have purchased, and allow them to activate them.

- **Map screen**

This screen will replace the main screen if the user has activated the map visibility upgrade. It will show the user a map with all of the coins available, and plot their location on the map, which will update in real-time.

- **Shop screen**

This screen will allow users to purchase upgrades and traps. It will list all of the purchasable items alongside a button which allows you to purchase it. If the user doesn't have enough currency to purchase this item, then the button will be disabled, and the user will be notified through colour or text that they don't have enough GOLD.

- **Banking screen**

This screen will show users all of the coins they have collected that day, and allow them to bank them. It will also show users what coins they have already banked that day, and what the conversion rates are for that day.

There should be a sub-screen that shows users what currencies they have in the bank, and allow them to convert these currencies to GOLD.

- **Messaging and friends screen**

This screen will allow users to add friends, and show them the list of existing friends. The user should be able to send their friends unbanked coins through this screen.

- **Leaderboard screen**

This screen will show the users a leaderboard of how much GOLD each player has. This should be filterable to show them just how many coins their friends have. The user should be able to see themselves on this screen.

Ambiguities

- **Type distribution of coin**

The specification does not include how many of each type of coin to include on the map. My app will generate 50 coins and randomly select a currency for each. This should lead to a roughly equal amount of each currency each day.

- **Selection of value of coins**

The specification doesn't require any particular method for deciding the value of each coin, only that they are between 0 and 10 of their respective currency. My app will randomly select a value in this range. This value should be stored in a decimal type instead of a floating point type since we are dealing with money.

- **User knowledge of coin value**

The users are to be made aware of the coin values that are visible to them to 2 decimal places.

- **User knowledge of conversion rates**

Whilst the specification says that the coins fluctuate in value compared to the reference GOLD currency, it doesn't say whether or not the user knows these conversion rates. The user should be able to find out these conversion rates, both from the main screen and the banking screen.

- **Currency fluctuation**

The specification ignores the implementation of currency fluctuations. To give an extra aspect to gameplay, I would like the currency conversions to fluctuate with a random factor up and down, but also take into account momentum. This will allow players to get an intuition for when the currency is likely to be worth more but also will mirror real markets better. It would also be interesting to have random drop or spike events, also to model more closely real markets.

- **Time of conversion to GOLD**

My app will convert currencies to GOLD at whatever point in time the player chooses after banking the coins. This adds in a gameplay element of attempting to follow the market and guessing when the best time to cash in on investments are. The trade-off for this will be that coins in their raw currencies are worth nothing, and you cannot purchase any in-game item with them, only with GOLD.

- **Expiration of coins**

Coins collected that haven't been paid into the bank or messaged to anyone else will expire at the end of the day. This will encourage players to collaborate and include friends in the game, as they then get greater rewards. If you allow players to pay in the coins at a future date then it somewhat removes the point of the 25 coin pay-in limit.

- **Receiving a spare coin that has already been paid in**

Players will not be allowed to receive coins from other players that have already been paid in that day by themselves. This will lead to more interesting strategising from players as they talk to their friends about which coins each of them should collect to maximise their earnings.

Bonus features

- **Blind vs Map mode**

Seeing the coins on a map so you know where to go to get them doesn't feel particularly challenging as a game mode. My version of Coinz will by default have a text only screen that tells you how close you are to the nearest 4 coins. This means that it will be a challenge just to locate the coins.

As part of this feature, I will allow people to use their saved up coins to buy a "Map mode". This will allow them to see all the coins on the map, which is a big advantage. The price for this should be set appropriately, potentially based on some kind of market, where the demand for these increases the price.

- **Upgrades and traps**

In the specification, once users have converted their coins to GOLD they just sit there in the bank and do nothing. Personally, I really enjoy games where I can use my accumulated rewards to increase my ability to get more rewards, so my app will allow the user to purchase upgrades and traps.

Upgrades will be items that give the user an advantage when it comes to collecting coins. One example would be the map mode upgrade, as described above.

Traps will be items that harm other users when collecting coins. These will add a competitive element for players that want to see themselves climb the leaderboard. They should only be able to be used on friends to prevent griefing (bullying in-game).

Both upgrades and traps should be time-limited to prevent them being overpowered.

Other examples:

Conversion rate improvement

This will be an upgrade purchasable with GOLD that will give you a better rate when converting a specific currency.

Halve value of other person's currency

This will be a trap that will allow users to target another user and halve the value of their collected coins for a day.

These aren't the only traps/upgrades that will be included. I have scheduled in a week for coming up with some more. I will aim for around 3 of each to make strategising for the game slightly more challenging and keep users interested.

- **Leaderboard**

The app will allow the user to see a leaderboard of the top users of the app. This will be sortable to either just the user's friends, or to see the top users overall.

Reasons for Choosing Kotlin

Concise code

My experience with Java on Android is that you end up writing 100 lines for what would otherwise take 10 in another language. When you come to maintain code like that, it can be very hard to decipher the intended purpose of the code due to all of the boilerplating that is needed. If this project is intended to be handed off to a team of developers, a conciser language should make the code easier to read, and help reduce the amount of documentation needed since the code is more likely to document itself.

Language features

Kotlin has much of the same functionality as Java, but has extra features that make your code safer and more readable for free. Here are some examples:

- **Null Checking** - You have to explicitly state a variable to be nullable, meaning that it's impossible to accidentally cause a NullReferenceException.
- **String Interpolation** - This is a feature that is in most modern languages, but Java still uses `String.format()`, a far less readable way of achieving the same thing.
- **Better equality checking** - In Java `==` checks for referential equality, meaning that it will check whether two objects are the same segment of memory. This is unintuitive, leading to lots of strange bugs when new to the

language. Kotlin has swapped `==` for structural equality, which follows the principle of least surprise.

Support for Kotlin

Google now officially supports Kotlin on Android. This is on top of the support offered by JetBrains, the creators of the language and creators of a mature IDE that natively supports it. The indicators are pointing towards Kotlin being the future of Android development, which makes it a natural choice for new projects that will be supported in future.

Timetable

Week	Milestones	Work done
2		Report started
3		Language researched and chosen
4	Create repo, finish report	
5	Login Screen and users table	
6	Start implementing main screen in blind mode	
7	Finish main screen in blind mode	
8	Come up with traps and upgrades. Implement the shop screen.	
9	Implement the banking screen including conversions	
10	Start implementing friend system	
11	Finish friend system and enable targeting of traps	
12	Leaderboard screen + bug swatting	
13	Bug swatting	