DESIGN

Informatics Large Practical

Abstract

A document containing design decisions about Coinz. A map based game where players have to collect cryptocurrency coins. Features of the app, choice of programming language and the timescale are all included.

Raees Aamir

Matriculation number: s1617910

Table of Contents

Software Development Timetable

Week	Objectives
2	• Setup private repository for Coinz development project and include Professor Gilmore as a contributor. Complete by Tuesday. • Have the word document for the design created and added to the Git repository by Tuesday. • Decide on programming language choice; Java vs Kotlin by Thursday. • List all objectives in the software development timetable by Saturday. • Start watching the relevant Firebase tutorial videos produced by TVAC Studios on Friday.
3	• Complete list of fundamental features to the Coinz app by Tuesday. • Complete list of four additional features to add to the Coinz app by Wednesday. • Start watching the video tutorials on Mockito and JUnit testing from Udemy. [3] • Finish watching half of the relevant Firebase tutorial videos produced by TVAC Studios on Friday. [4]
4	• Start making paper designs of the MVC views by Monday. For example, the login view, the map view, the wallet view, the settings view, and the bank view. • Perform final spelling, grammar and content checks on design document and submit by Tuesday afternoon. • Finish the paper designs of the MVC views by Thursday. • Finish watching all the relevant Firebase tutorial videos produced by TVAC Studio by Friday. [4]
5	• Add a commons dependency like Google Guava or Apache Commons to the Gradle project by Monday. • Make a document for listing the algorithms and data structures used in developing the MVC models of entities by Monday. • Start programming the MVC models of entities involved in the gameplay by Monday. For example, the player model, the coin model, the map model, the wallet model, and the bank model. Whilst programming the models, add descriptions of algorithms and data structures used. • Finish implementation of MVC models of entities by Friday.
6	• Start translating all paper designs of MVC views to Android XML activity views by Monday. • Finish translating all paper designs of views to Android XML activity views by Friday. • Finish watching the videos from the Udemy Mockito and JUnit tutorials by Friday. [3]
7	• Make a document for listing the algorithms and data structures used in developing the Firebase backend server by Monday.

	• Start writing the backend Firebase server by Monday. Whilst programming the backend Firebase server, add descriptions of algorithms and data structures used to the document. • Start writing automation testing scripts for the MVC models and the MVC views by Thursday. • Make a document for listing the algorithms and data structures used in developing the MVC controllers by Wednesday. • Start implementing the MVC controllers by Wednesday. Whilst programming the controllers, add descriptions of algorithms and data structures used to the document.
8	• Finish writing automation testing scripts for the MVC models and the MVC views by Wednesday. • Finish programming the backend Firebase server by Thursday. • Have the word document for the implementation report created and added to the repository by Friday. • Put all documents listing algorithm and data structure descriptions into the implementation report by Friday.
9	• Finish implementing the MVC controllers by Wednesday. • Add screenshots of the Coinz gameplay to the implementation report by Thursday. • Begin writing unit testing scripts for each MVC controller by Friday • Begin writing unit testing scripts for the backend Firebase server by Thursday.
10	• Finish writing unit testing scripts for each MVC controller by Friday. • Finish writing unit testing scripts for the backend Firebase server by Tuesday. • Start documenting the unit testing scripts for the backend Firebase server by Wednesday.
11	• Finish documenting the unit testing scripts for the backend Firebase server by Monday. • Start documenting each MVC controller by Tuesday. • Start documenting each MVC model by Wednesday. • Finish documenting the unit testing scripts for each MVC controller by Friday.
12	• Finish documenting each MVC controller by Monday. • Finish documenting each MVC model by Tuesday. • Start checking spelling, grammar and writing style of documentation and implementation report by Wednesday.
13	• Perform final checks of spelling, grammar and writing style on documentation and implementation report by Monday. • Perform final checks of the codebase and submit the project by Tuesday afternoon.

Fundamental Features to Coinz

Resolution to ambiguities

- Coins are converted to GOLD at the time they're paid into the bank. So a player could deposit 5 PENYs and 2 SHILLs and they would be changed into GOLD according the exchange rate. Once they are converted into GOLD, they cannot be converted back and withdrawn as PENYs and SHILs.
- If the coins aren't deposited into the bank by the end of the day, then they expire. During the day, unwanted coins can be traded to another player who needs them.
- If a player receives as spare change a coin which they have already deposited into the bank, they will be able to deposit the spare change as long as they have less than 25 coins in their bank. However, as per usual, the player will not be able to withdraw their spare change after depositing it into the bank.

Logo

- Since the application is based on collecting cryptocurrency coins, the logo will be different coloured coins stacked on top of each other.

Welcome

- On starting the application, a welcome splash screen will appear with the logo. Once the splash screen is complete, the application will redirect to the menu.

Menu

- The menu will include the logo at the header of the view with an appropriate background to create an aesthetic look.
- In the body of the view (below the header and above the footer), there will be three buttons: Play, View leader-board and Exit.
  - The play button will redirect to a login view if the user is not logged into the game, otherwise it will redirect to a map view where the player can play the game.
  - If the player is logged in, at the right of the header there will be a door to signify the option to logout of the game.
  - The view leader-board button will redirect to a view showing the top players and their total amount of GOLD.

Additional features of Coinz

Coin Trading

Players will be able to trade coins with other players, so someone might decide to swap a PENY for a QUID or vice versa. Players will be able to trade coins at any time in the game; there won't be a specific trading time. On the home screen there will be an additional option called "Trade" which will show a list of online players. Tapping the name of the player will open a trade screen showing the coins in the player's wallet and the coins in the other player's wallet.

Leader-board

In an effort to make the game more competitive, a leader-board will be included. On the home screen there will be an additional option to open a leader-board. Pressing the leader-board button will open a view showing a list of players and the total amount of GOLD in their wallet. There will be two types of leader-boards. The first type of leader-board will show every players total GOLD value since they started playing the game. The second type of leader-board will show the total amount of GOLD every player obtained by playing today's map. Personally, I believe having two types of leader-boards will distinguish between players who are just having a good day and players who are consistently skilled at the game.

Distance Recording

The 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.

When the player arrives at the coin marker, the application will be able to determine if the player's path to the coin is the shortest path by comparing the path with the path generated by A* OR Dijkstra's. If a path-finding algorithm is implemented in Mapbox, I will not implement A* or Dijkstra's algorithm.

Share on social media.

When the player signs out of the game, there will be an option to share on social media. Sharing on social media will make a post showing how many coins and what types of coins the player collected. If the player hasn't collected any coins, then the option to share on social media will not appear.

Programming Language Choice

Decision made: Java

Justification:

In this section, I'm going to discuss the advantages and disadvantages of choosing Kotlin over Java or by choosing Java over Kotlin. I will weigh up the advantages and disadvantages of each choice and I will reach a conclusion based on which language maximizes benefits and minimizes costs.

As of the release of Android Studio 3.0 in October last year, Google added Kotlin as an officially supported programming language for the Android SDK. When developing a specification for a project, this might cause analysts to ponder on which choice of language would be the best for a new Android app. There are many things to consider such as the supported libraries, the Java to Kotlin translator and the syntax and semantics. A conservative developer might want to stay with Java until Kotlin matures; whilst a more liberal developer would be open to going with Kotlin in its youth.

Since it is possible to use Java libraries in Kotlin and vice versa, simply looking at which language has better libraries is not a valid comparison. A more important factor is thinking about how *effective* the Java to Kotlin translator is. It seems to be fine at translating simple Java code to Kotlin code, but there are situations where it can struggle translating complex Java code to Kotlin. For example, the Stackoverflow question [1] is about a developer using a Java "CallbackWrapper" in Kotlin that works fine; however, when the Java "CallbackWrapper" is translated to Kotlin, the compiler throws cryptic errors. The developer who answered the question noticed that the problem was caused by the fact Kotlin "doesn't allow using lambdas to implement Kotlin functional interfaces, so your lambda isn't going to work (*it does allow implementing Java functional interfaces, because Java doesn't have proper function types*)" The Android Studio translator translated the Java lambda to a Kotlin lambda but it did not realise the Kotlin lambda would fail because the Java lambda implemented a functional interface. Unfortunately, a novice Kotlin developer would struggle to resolve a problem like this and it could cost the company lots of time and money to fix the problem.

One of Kotlin's biggest advantage over Java is null-safety by default. This is in contrast to Java which does not have null-safety. No null-safety is a major problem because it can cost developers a significant amount of time debugging silly errors. Tony Hoare called this the "billion-dollar mistake" because he thought debugging null references have cost companies a billion dollars (in terms of time) over the past 40 years. With the release of Java 8 came the introduction of java.util.Optional which forces you to not forget null checks. But as the name suggests, it's optional, in Kotlin it's mandatory. The optional class does not solve the lack of null safety in Java because lots of old frameworks and software were not written using Java 8 features, and thus don't use Optional.

So far there's a fair tie between Java and Kotlin; Kotlin wins when it comes to null-safety and syntax, but Java wins in the compatibility department. But we need to make a decision now

because if we choose the wrong language, it could lead to further problems down the line. For example, even though Google has an official page on migrating Java projects to Kotlin, there have been disputes about whether it's an effective way to move to Kotlin because of the problems it introduces. For example, in the migration article [2], Android Developer Paulina Sadowska talks about how following Google's guidance will introduce problems if the Java project uses certain libraries. She talks about how using Java with Lombok to generate getters and setters causes errors in Kotlin because "when the Kotlin compiler runs it uses javac as well but with no annotation processing" Therefore, if we want to use Kotlin we must decide now. It would be a bad idea to start in Java and then migrate to Kotlin later because using libraries that involve annotations will cause strange bugs in Kotlin.

In conclusion, I have decided to program the Coinz app in Java. Java wins when we consider code migration and translation problems. Kotlin wins when we consider null-safety and syntax. However, I won't be completing the Coinz app, I will just be making a prototype to pass onto a development team. There's a significant time cost if 5 or 10 people have to spend a week learning the intricacies of Kotlin to understand good Kotlin programming principles. We could just hire Kotlin developers but given the low supply and high demand, management will have to pay a higher salary.

Key phrases and notation used

- [i] means the ith reference. So [1] is the 1st reference.
- MVC means model-view-controller. A common architectural design pattern used in web and mobile applications development. Views are the Android XML files and Controllers are the Android Activity classes. Models are plain Java classes used to represent in-game processes and entities.

References

[1] – Mgebrishvilli, J. (2017). Converted Java class file to Kotlin makes compilation error. Stackoverflow. [online], p.1. Available at: https://stackoverflow.com/questions/43235423/converted-java-class-file-to-kotlin-makes-compilation-error
[Accessed 26 Sept. 2018]

[2] – Sadowska, P. (2018). How to f*** up Java to Kotlin migration in your existing Android app. *Usejournal.* [online], p. 1. Available at: https://blog.usejournal.com/how-to-fuck-up-java-to-kotlin-migration-in-your-existing-android-app-325b57c9ddbb
[Accessed 26 Sept. 2018]

[3] – in28Minutes, O. (2018). Mockito tutorial with JUnit examples. *Udemy.* [online], p.1. Available at: https://www.udemy.com/mockito-tutorial-with-junit-examples
[Accessed 29 Sept. 2018]

[4] – TVAC Studio. (2018). Android Studio – Firebase Backend Tutorial Full Course. *Youtube.* [online], p.1. Available at: https://www.youtube.com/playlist?list=PLGCjwl1RrtcTXrWuRTa59RyRmQ4OedWrt
[Accessed 28 Sept. 2018]