



Cardiff University

School of Computer Science and Informatics

CM3203 - Individual Project Report

# Currency Exchange Notifier: Finding the right time to exchange currency

Supervisor: Dr Parisa Eslambolchilar

Moderator: Dr Sylwia Polberg

Author: Eun-Kyul Ko

12<sup>th</sup> May 2022

# Abstract

The foreign exchange market is very unpredictable; the fluctuations of the currency exchange rate makes it harder for people to know when to buy or sell currencies. These currency fluctuations can be caused by political affairs, economics and businesses especially the current world wide affairs such as COVID and Ukrainian crisis. The larger the amount of money exchanged, the greater the impact of the fluctuation. However, monitoring currency charts every day waiting for exchange values to drop is unnecessary and tedious.

In this project I will be presenting a solution that helps people to trade their currency at a better rate. Currency Exchange Notifier is an Android mobile application that sends out notifications when the user's desired currency is reached, meaning the user does not have to constantly monitor their currency of interest. This report documents the whole process of the project including the background research, approach, design, implementation, testing and evaluation of Currency Exchange Notifier.

# Acknowledgements

I would like to express my appreciation to my supervisor Dr Parisa Eslambolchilar for all her help and guidance through the project. The feedback she provided and the support she has shown have given me the confidence to finish the project.

I could not have undertaken this journey without my friends Sabeegah, Malaika and Kiya. The love and support they gave me throughout this project and the university life as a whole was invaluable.

# Table of Contents

Abstract	2
Acknowledgements	3
Table of Contents	4
1. Introduction	5
2. Background research	6
3. Approach	12
3.1. Methodology	12
3.2. Software choices	12
3.2.1. IDE	12
3.2.2. Currency API	13
4. System Design	14
4.1. Specification	14
4.1.1. Functional requirements	14
4.1.1.1. Must have	14
4.1.1.2. Should have	15
4.1.1.3. Could have	16
4.1.2. Non-functional requirements	17
4.1.2.1. Must have	17
4.1.2.2. Should have	17
4.2. Use Cases	17
4.3. User Interface Design	22
5. Implementation	32
5.1. Database creation	32
5.2. Overview of implementation structure	33
5.3. Code quality	42
6. Results and Evaluation	43
6.1. Test Cases	43
6.2. User Testing	44
7. Future work	45
8. Conclusions	47
9. Reflection on Learning	48
References	49
Appendices	50

# 1.Introduction

The aim of this project is to create a prototype of a mobile application that sends out notifications when the currency exchange rate drops below a certain point. This will help the user to know when the optimal time to trade their currency at the best rate is.

The three main objectives of the project are:

- To gain an understanding of the currency fluctuations and existing solutions in the market
- To implement prototype with features that help users to exchange currencies at the right time
- To create a UX design that is suitable for a currency exchange notifying mobile application

For the first objective I will begin with a series of background research, to gain an in-depth understanding of how currency fluctuation is affected by different factors. At the same time I will be carrying out market research to identify and review existing solutions and how they are integrated with other features of the service.

The second objective is the core part of this project, the implementation of the prototype. This prototype should allow the user to 1) select the currency they want to get the notification for and 2) enter the desired value of that currency relative to the user's default currency. The detailed specification and requirements of the app will be determined in the system design part of the project.

Lastly, maintaining a high level of UX design is important. Keeping your system usable is equally as important as the integration of main functions; even if an app has amazing features and high quality functions, if it has a bad UX design it will eventually fail to attract users to the system. To make sure that I have a user friendly interface I will be focusing on producing self-descriptive and consistent design, with a clear and conventional design of buttons and icons. Also the system must be easy to use regardless of the user's technical skills. In order to make sure I achieve this I will generate a high fidelity mockup prototype design.

## 2. Background research

Although I had a clear idea of the application I wanted to develop, it was essential to gain background knowledge of the existing solutions that are currently on the market. Before implementing the prototype I found that there are several mobile applications and online services that provide notifications when currency values drop.

The following section analyses existing solutions of currency exchange notifiers that are on the market.

<b>Name:</b> XE	<b>Type:</b> Mobile application
<b>Summary:</b> XE is an online foreign exchange tool that allows users to send money abroad within a one click process. It provides lots of additional features other than the international transfer features such as a currency chart, live exchange rates, rate alerts, a travel expenses calculator and daily currency updates.	
<b>Features:</b> <ul style="list-style-type: none"> <li>• Live exchange rates for global currencies</li> <li>• Quick quote &amp; send for international transfer</li> <li>• Money transfer tracking service</li> <li>• Currency chart with live rates and historical data for past 10 years</li> <li>• Rate alerts for desired currency rates</li> </ul>	
<b>Rating:</b> App Store: 4.1/5 (2.6k ratings) Google Play: 3.4/5 (106k ratings)	
<b>Reviews:</b> "A very practical app" "The rates are far from accurate" "The currency that I convert more often has a long name and so it bugs out and gets covered by numbers, making it difficult to read" "Perfect for doing quick and accurate currency calculations" "The charts are mostly unresponsive when you choose a different date range"	
<b>Evaluation:</b> The app has a simple and aesthetically pleasing UI design. The refresh cycle at the top of the screen suggests that the currency rate is updated every 60 seconds. The main function of the app is the exchange function, making the exchange notification feature complementary. It was very straightforward and easy to set up the notifications. However, when the actual notification is sent, it is received twice which is an error as I did not specify it to do so in the settings. Also the set up expires after one use, therefore if the user wants to be notified multiple times for the same rate they have to set up multiple notifications. Lastly, they calculate their own exchange rates which could be beneficial for customers if the rate is cheaper than the actual currency rates but lots of user reviews disapprove of it. In a review, one user said they lost approximately £80 because of the rates they were offering.	
<b>Available at:</b>  <a href="https://apps.apple.com/app/apple-store/id315241195?pt=211875&amp;ct=Website%20-%20QR%20Code&amp;mt=8">https://apps.apple.com/app/apple-store/id315241195?pt=211875&amp;ct=Website%20-%20QR%20Code&amp;mt=8</a>  <a href="https://play.google.com/store/apps/details?id=com.xe.currency&amp;referrer=utm_source%3DWebsite%26utm_medium%3DQR%2520Code%26utm_campaign%3DWebsite%2520%252F%2520QR%2520Code">https://play.google.com/store/apps/details?id=com.xe.currency&amp;referrer=utm_source%3DWebsite%26utm_medium%3DQR%2520Code%26utm_campaign%3DWebsite%2520%252F%2520QR%2520Code</a>	

<b>Name:</b> Wise (formerly Transferwise)	<b>Type:</b> Mobile application
<b>Summary:</b> Wise is a money transfer application that allows users to send money abroad. It almost works like a banking app, where users can easily make transactions abroad and convert money to different currencies.	
<b>Features:</b> <ul style="list-style-type: none"> <li>• Cheaper and faster overseas money transfers</li> <li>• Send money with any currency</li> <li>• Rate tracker</li> <li>• Hold 50+ currencies and can convert between them instantly</li> <li>• Historical currency data of up to 5 years</li> <li>• Currency chart with different time scales</li> </ul>	
<b>Rating:</b> App Store: 4.7/5 (42k ratings) Google Play: 4.5/5 (181k ratings)	
<b>Reviews:</b> “Efficiency and economy in financial transfers.” “Shockingly convenient and easy” “Transferwise is unreliable” “I made a transfer to my own account in Spain but the money was held for checks for 12 days with the excuse that it may be a sanctioned bank or a fraudulent transaction.”	
<b>Evaluation:</b> This application is more like a banking app, it is heavily focused on international money transfer features. Wise provides the cheapest transfer rate but only support 53 currencies for conversion - which is a lot less than most currency services offer. Wise state that they have a currency rate notification feature but I could not find it when using the app. There is a possibility they do not have it in the app, otherwise it is not immediately clear where to find it. They have their currency ‘rate tracker’ feature on their website where they send emails to users for 1) daily updates and 2) desired currency rate that the user sets. However this feature was not implemented sufficiently as I did not receive any email notifications.	
<b>Available at:</b>  <a href="https://apps.apple.com/gb/app/wise/id612261027">https://apps.apple.com/gb/app/wise/id612261027</a>  <a href="https://play.google.com/store/apps/details?id=com.transferwise.android">https://play.google.com/store/apps/details?id=com.transferwise.android</a>	



<b>Name:</b> OFX	<b>Type:</b> Website
<b>Summary:</b> OFX is an online service for international money transfers. They offer quick and zero fee transfer, with tracking features. They also have a target rate transfer service with a complementary 'market rate alert' feature.	
<b>Features:</b> <ul style="list-style-type: none"> <li>● Conversion of 50+ currencies to over 190 countries</li> <li>● Sending money overseas</li> <li>● Tracking international transfer</li> <li>● Market rate alerts</li> </ul>	
<b>Rating:</b> Trustpilot: 4.2/5 (4.8k ratings)	
<b>Reviews:</b> "Yet another exchange company looking to park your money for as long as possible. Scam" "have used this service for many years and found it very good" "Consistently straight-forward and efficient" "Deceptive currency conversion rate" "it comes with a really poor rate alert online system and it won't notified me when my target price reached, which was disappointed"	
<b>Evaluation:</b> OFX have a 'market rate alerts' function where they allow users to be notified through email once their currency pair meets the target rate that the user has set. Unfortunately, I did not receive any email notifications even though the currency dropped below the target rate multiple times. Several users also left reviews commenting on how the 'market rate alerts' feature did not work properly, saying how the rate was incorrect or that they did not receive a notification at all.	
<b>Available at:</b> <a href="https://www.ofx.com/en-gb/">https://www.ofx.com/en-gb/</a>	

<b>Name:</b> CurrencyFair	<b>Type:</b> Mobile application
<b>Summary:</b> This is another international transfer app that helps users to send money abroad. They offer both personal and business use of the transfer and online foreign exchange. The interesting feature of CurrencyFair is that when there is a user who wishes to get a funding in one currency, they find a user that has a corresponding need and match them together therefore they can literally 'exchange' the funding instead of sending the user's money to an account abroad.	
<b>Features:</b> <ul style="list-style-type: none"> <li>• Overseas money transfer</li> <li>• Multi currency accounts</li> <li>• Daily/weekly/custom rate alerts</li> <li>• Currency chart with different time scales</li> </ul>	
<b>Rating:</b> App Store: 4.7/5 (736 ratings) Google Play: 4.2/5 (719 ratings) Trustpilot: 4.6/5 (5.6k ratings)	
<b>Reviews:</b> "so easy to use and understand" "Efficient and secure system of converting and transferring funds" "The money has not been credited to my account and appears to have gone missing" "Strict process in setting up so it takes a while. But once in place the exchange is easy and good rates" "Generally good why the complicated log in all the time"	
<b>Evaluation:</b> The CurrencyFair mobile app focuses on currency exchange and sending money features, and actually did not include a rate notification feature at all. Instead the currency exchange rate alert feature was available on their website. They have daily, weekly or custom rate alert features through email and that work efficiently. I received notifications every time the desired value I set was reached, even several times a day, until I unsubscribed from the notification. However, they offer very limited availability for sending and receiving currencies with a total of 17 sending currencies and 22 receiving currencies. Also the app itself malfunctioned often, after 5 minutes of logging in, it informed me that the session timed out and that I would have to log back in again. The overall reviews of the service was the most positive out of all the ones I have reviewed.	
<b>Available at:</b>  <a href="https://play.google.com/store/apps/details?id=com.currencyfair.apps.android.currencyfair&amp;hl=en">https://play.google.com/store/apps/details?id=com.currencyfair.apps.android.currencyfair&amp;hl=en</a>  <a href="https://itunes.apple.com/ie/app/the-currencyfair-money-transfer-">https://itunes.apple.com/ie/app/the-currencyfair-money-transfer-</a>	

[app/id1020622852?mt=8](https://www.currencyfair.com/app/id1020622852?mt=8)

<https://www.currencyfair.com/> (website)

Overall, there are both web and mobile currency exchange applications that have currency rate alert features, however some of them do not work at all, or do not offer a wide range of currencies; even if they were working they are not as accurate as they should be, for example notification not coming through even though when the rate is achieved. Another finding was that there is no application that is only dedicated to currency notification features, it usually comes as a complementary feature to currency exchange apps or services. The prototype I aim to create will not have any transfer features but instead will be heavily focused on the currency exchange rate alerting feature with some other complementary features such as historical chart, relevant news and favouriting currencies.

I also found that there are similar apps and services that contain this feature, for example American Express also offers a foreign exchange rate alert service. However in order to test this feature you need to be an American Express customer, and therefore I was unable to test it. Similarly, with the mobile app Conotoxia - I was unsuccessful in testing the rate alert service as this required verification.

## 3.Approach

### 3.1. Methodology

In order to deliver this project successfully, I followed the mixture of two methodologies: Design thinking and Agile development methodology. Design thinking methodology is a user-focused research methodology that is widely used in solution based problems in the UX design industry. Prior to beginning this final year project, I brainstormed my ideas for developing a suitable solution in a module I previously completed 'CM3116 Design Thinking and Prototyping for User Experience'. I wanted to display the skills gained from this module by choosing a design thinking methodology. The benefit of this methodology is that it covers the 5 stages of the design thinking process: empathise, define, ideate, prototype and test which enabled me to seek different approaches and helped me come up with a potential solution.

When I began implementing the system, I followed the Agile software development methodology within the prototyping stage of the design thinking methodology. I broke up the project into several stages with clear milestones, which can be seen in my initial plan. I listed my weekly goal on my reminders app and synced to my laptop so I could regularly check my progress across all my personal devices. I also made sure I kept on track by having regular meetings with my supervisor to report the progress I had made each week. Due to my lack of experience in developing a mobile application, I anticipated a lot of complications to occur, therefore I allowed myself some flexibility by scheduling extra learning time to develop the application. Furthermore, I focused on implementing the basic functions before adding additional features or applying an aesthetically pleasing UI design.

### 3.2. Software choices

#### 3.2.1. IDE

In order to develop a mobile application for the Android platform, I used Android Studio which is an official integrated development environment for Android operating system. Although I have no prior experience on building a mobile application myself, there were lots of tutorials and documentation available for Android Studio which made me feel confident to learn. The programming languages that Android Studio offers are Kotlin, Java and C++; I decided to develop my prototype in Java alongside with XML since I was more comfortable working in Java with the prior experience I gained throughout this degree and XML was straightforward enough for me to familiarise myself with short amount of time. Although I

do not own any Android devices myself, the Android Emulator, which is a virtual device included in the Android Studio environment allowed me to run and debug the prototype while I was developing.

### 3.2.2. Currency API

Through extensive research, I managed to create a list of different currency APIs available. I listed them in a spreadsheet to compare each API. In this table I focused on comparing the formats they support, currency update cycle, number of currencies they are supporting and whether they distribute their APIs freely or not.

	A	B	C	D	E
1	API	formats	update cycle (for free plan)	no. of currencies	no. of calls available (for free plan)
2	fixer		1 hour	170	100 API calls/month
3	currencylayer	JSON	1 day	168	250 requests/month
4	Exchangerates	JSON	60 seconds		250 requests/month
5	CURRENCY API	XML, JSON	1 hour	170+	100 requests/day
6	Open Exchange Rates	JSON	1 hour	200+	1000 requests/month
7	ExchangeRate-API	JSON	1 hour	161	1500 API requests/month
8	XE	XML, JSON, CSV	1 hour	170+	free trial available
9	Xignite	XML, JSON, CSV	real time	170+	free 7 day trial
10	IBAN	XML, JSON	10 minutes	154	30 day fully featured trial

From the information I obtained, all of the listed APIs support JSON format with some of them also supporting XML. More than half of them have an update cycle of 1 hour, and the majority of them support more than 160 currencies which is sufficient, considering there are 180 official currencies in the world that are recognised by the UN.

Lastly, the number of API calls that are available was quite different for each API, from 100 requests per month to 100 requests per day. Some of them offer fully featured trials for certain periods of time but the time scale was not enough.

Ideally my prototype has to obtain the currency data in 'real time' because that was the whole point of the system - sending alerts to users as soon as the currency drops. However the only API that offers real time updates is Xignite and unfortunately they do not offer solutions for individual or personal use. Taking all these factors into account I found the CURRENCY API the most suitable one to implement. It has the most API call allowance for free plan, supports both XML and JSON formats and there are 170+ currencies available for this API. As well as the update cycle occurs on an hourly basis which is arguably the closest to real time. Their documentation offers example requests to follow, it was simple to use and easy to integrate.

## 4. System Design

### 4.1. Specification

#### 4.1.1. Functional requirements

##### 4.1.1.1. Must have

FR1. The system must allow users to sign in/login securely

Acceptance Criteria:

- The user can log in to the system with the username and password they set.
- If the user has no account registered, they will be able to create an account
- The sign in credentials are stored in a database
- The system sends an error message when invalid credentials are entered

FR2. The system must allow users to set their default currency

Acceptance Criteria:

- The user can select a currency from a dropdown list
- Once the currency is selected users can see the conversion for their selected currency

FR3. The system must allow users to set their target currency

Acceptance Criteria:

- The user can select their target currency (currency they want to see the conversion of) from a dropdown list
- Once the currency is selected users can see the conversion from their target currency

FR4. The system must do the conversion of the selected currencies in real time

Acceptance Criteria:

- The API converts currency values in real time
- The user can see the direct conversion of the currency

FR5. The system must let the user to set the notification for the currency pair they want to receive notifications for

Acceptance Criteria:

- The user is able to set up notification for the currency pair they selected
- The user is able to type in their desired rate of the chosen currency

FR6. The system must let the user change their default and target currencies

Acceptance Criteria:

- The user is able to change their default and target currency from the settings

4.1.1.2. Should have

FR7. The system should have an option for changing the refresh cycle i.e., how often the data will be pulled from the server

Acceptance Criteria:

- The user is able to select one of the radio buttons for different refresh cycle
- There is at least 3 options

FR8. The system should have a feature that limits the number of notifications per day

Acceptance Criteria:

- The user is able to set the maximum number of times they want to be notified for.

FR9. The system should allow users to favourite certain currencies

Acceptance Criteria:

- The user is able to favourite the currencies they visit frequently
- The user is able to browse the list of their favourite currencies

#### 4.1.1.3. Could have

FR10. The system could display daily updates/notifications of the favourite currencies

Acceptance Criteria:

- The user can get daily rate updates once a day on a selected time

FR11. The system could show a historical currency chart to see the trend of selected currency pairs

Acceptance Criteria:

- The user can see the historical data of a selected currency

FR12. The system could have a news feature that displays the information about a currency

Acceptance Criteria:

- When the user clicks the currency they can see the relevant news, which can be general information about the currency, or world wide affairs that is going on in the world currently which may affect the price of the currency

FR13. The system could allow the user to receive notifications via email

Acceptance Criteria:

- In the currency notification set up page, there is an option where the user can also receive the notification by email

FR14. The system could have multilingual features

Acceptance Criteria:

- The user can change the language of the app



### 4.1.2. Non-functional requirements

#### 4.1.2.1. Must have

1. The system must be intuitive and easy to use for the user

Acceptance Criteria:

- The system must be user-friendly, the user should be able to navigate the system easily without background knowledge
- The system must have conventional and consistent design throughout

2. The app must be compatible with most Android devices

Acceptance Criteria:

- The app must be runnable in most of Android devices that has Android version 10 or above

#### 4.1.2.2. Should have

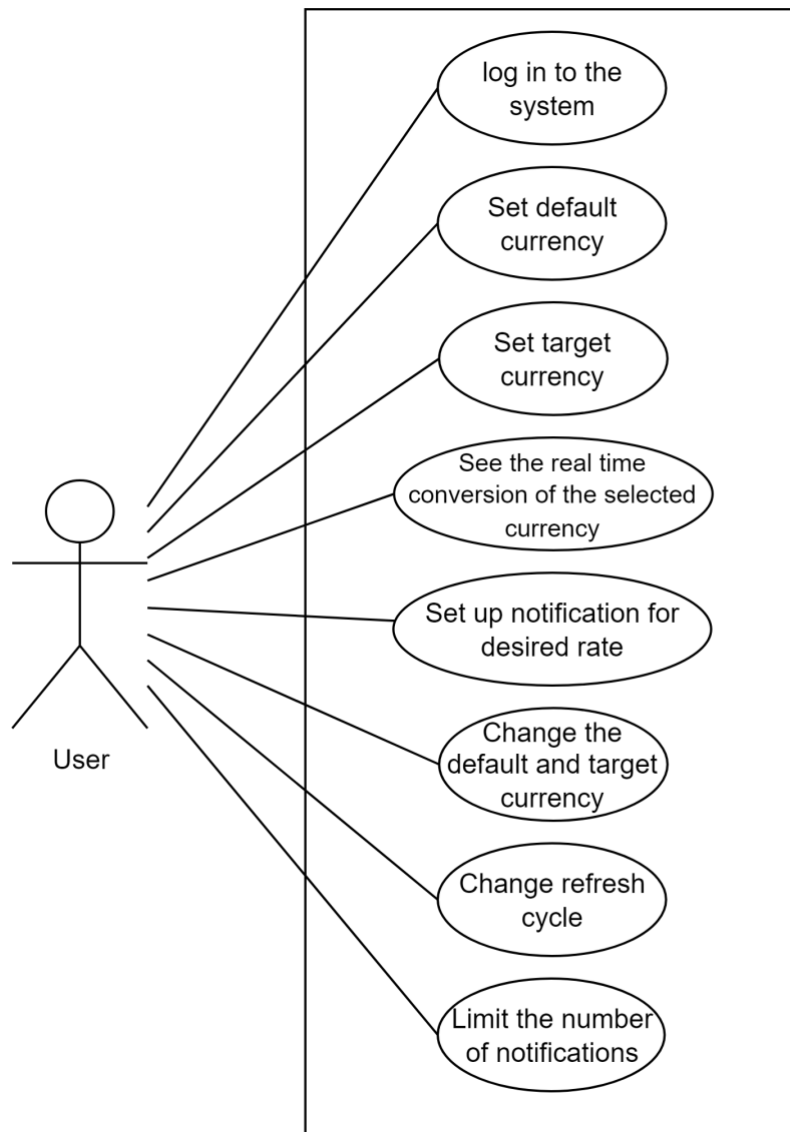
3. The app should have a professional and aesthetic design

Acceptance Criteria:

- The app has an aesthetically pleasing UI design

## 4.2. Use Cases

In this section I will discuss the use cases of the system, which are based on the 'must have' functional requirements and some of the 'should have' requirements. The use case diagram below illustrates a graphical outline showing the interaction between the actor User and the key functional requirements.



<b>Use Case No: 1</b>	<b>Use Case Name:</b> Sign up/log in to the system	<b>Type:</b> Must have
<b>Description:</b>	The user can create an account by registering to the system. If the user already has an account registered, they will be able to log in.	
<b>Pre-conditions:</b>	N/A	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. On the sign up page, the user enters their username and password and retype password.               <ol style="list-style-type: none"> <li>a. User selects the 'register' button</li> </ol> </li> </ol>	

	<ol style="list-style-type: none"> <li>2. User clicks the login button <ol style="list-style-type: none"> <li>a. User enters their username and password</li> </ol> </li> <li>3. User is directed to the default currency page</li> </ol>
<b>Alternative Flow:</b>	If the user enters the wrong credentials access will be denied.
<b>Related Use Case:</b>	All use cases require the user to be logged in.

<b>Use Case No.: 2</b>	<b>Use Case Name:</b> Set default currency	<b>Type:</b> Must have
<b>Description:</b>	User can select their default currency from a dropdown list	
<b>Pre-conditions:</b>	The user has logged into the system with their username and password	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks the arrow on dropdown list</li> <li>2. User scrolls through the list of currencies</li> <li>3. User selects their default currency</li> <li>4. User clicks the submit button</li> </ol>	
<b>Alternative Flow:</b>	N/A	
<b>Related Use Case:</b>	UC1 (User must be logged in), UC3 (User sets their target currency)	

<b>Use Case No.: 3</b>	<b>Use Case Name:</b> Set target currency	<b>Type:</b> Must have
<b>Description:</b>	User can select their target currency from a dropdown list	
<b>Pre-conditions:</b>	The user has logged into the system and selected their default currency	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks the arrow on dropdown list</li> <li>2. User scrolls through the list of currencies</li> <li>3. User selects their target currency</li> <li>4. User clicks the submit button</li> </ol>	
<b>Alternative Flow:</b>	N/A	
<b>Related Use Case:</b>	UC1 (User must be logged in), UC2 (User must have set their default currency)	

<b>Use Case No.:</b> 4	<b>Use Case Name:</b> See the real time conversion of selected currencies	<b>Type:</b> Must have
<b>Description:</b>	The user can see the direct conversion of the currencies they selected	
<b>Pre-conditions:</b>	The user has logged into the system and selected their default and target currency	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User enters the amount they want to convert</li> <li>2. User clicks the 'convert' button</li> </ol>	
<b>Alternative Flow:</b>	N/A	
<b>Related Use Case:</b>	UC1 (User must be logged in), UC2 (User must have set their default currency), UC3 (User must have set their target currency)	

<b>Use Case No.:</b> 5	<b>Use Case Name:</b> Set up notification for the desired rate	<b>Type:</b> Must have
<b>Description:</b>	The user can set up notification for the desired rate of the currency pair they chose	
<b>Pre-conditions:</b>	The user has logged into the system and selected their default and target currency	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks the notification icon</li> <li>2. User enters the value for the desired rate</li> <li>3. User clicks 'submit' button</li> </ol>	
<b>Alternative Flow:</b>	N/A	
<b>Related Use Case:</b>	UC1 (User must be logged in), UC2 (User must have set their default currency), UC3 (User must have set their target currency)	

<b>Use Case No.:</b> 6	<b>Use Case Name:</b> Change the default and target currency	<b>Type:</b> Must have
<b>Description:</b>	The user can change their default and target currency from the settings	
<b>Pre-conditions:</b>	The user has logged into the system and selected their default and target currency	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks settings</li> <li>2. User clicks 'change default and target currency'</li> </ol>	

	<ol style="list-style-type: none"> <li>a. User selects 'change default currency'</li> <li>b. User clicks the arrow on dropdown list</li> <li>c. User scrolls through the list of currencies</li> <li>d. User selects their target currency</li> <li>e. User clicks the submit button</li> </ol> <ol style="list-style-type: none"> <li>3. User selects 'change default currency'               <ol style="list-style-type: none"> <li>a. User clicks the arrow on dropdown list</li> <li>b. User scrolls through the list of currencies</li> <li>c. User selects their target currency</li> <li>d. User clicks the submit button</li> </ol> </li> </ol>
<b>Alternative Flow:</b>	User changes their mind and exits the page without changing the currencies
<b>Related Use Case:</b>	UC1 (User must be logged in), UC2 (User must have set their default currency), UC3 (User must have set their target currency)

<b>Use Case No.: 7</b>	<b>Use Case Name:</b> Change refresh cycle	<b>Type:</b> Should have
<b>Description:</b>	The user can change how often the app will refresh	
<b>Pre-conditions:</b>	The user has logged into the system	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks the settings</li> <li>2. User clicks 'general'</li> <li>3. User clicks 'refresh cycle'</li> <li>4. User selects one of the radio buttons for different refresh cycle</li> </ol>	
<b>Alternative Flow:</b>	User exits the page without changing anything	
<b>Related Use Case:</b>	UC1 (User must be logged in)	

<b>Use Case No.: 8</b>	<b>Use Case Name:</b> Limit the number of notifications	<b>Type:</b> Should have
<b>Description:</b>	The user can limit the number of notification they receive in a day	
<b>Pre-conditions:</b>	The user has logged into the system	
<b>Basic Flow:</b>	<ol style="list-style-type: none"> <li>1. User clicks settings</li> <li>2. User clicks 'notification'</li> </ol>	

	3. User selects the tick box for 'limit the number of daily notifications' 4. User enters the value
<b>Alternative Flow:</b>	User exits the page without changing anything
<b>Related Use Case:</b>	UC1 (User must be logged in)

### 4.3. User Interface Design

In order to make a mobile application that is user friendly, I created an initial mock up of the interfaces using a design tool called Figma. These mock-ups demonstrated how the key functions will be integrated and how each activity is related to each other. They also allowed me to receive user feedback on the design and provided me with a clear guideline on what I should be implementing to keep the system user focused. In this section I will detail my design justifications and how the initial mock-ups were improved based on the user feedback I received.

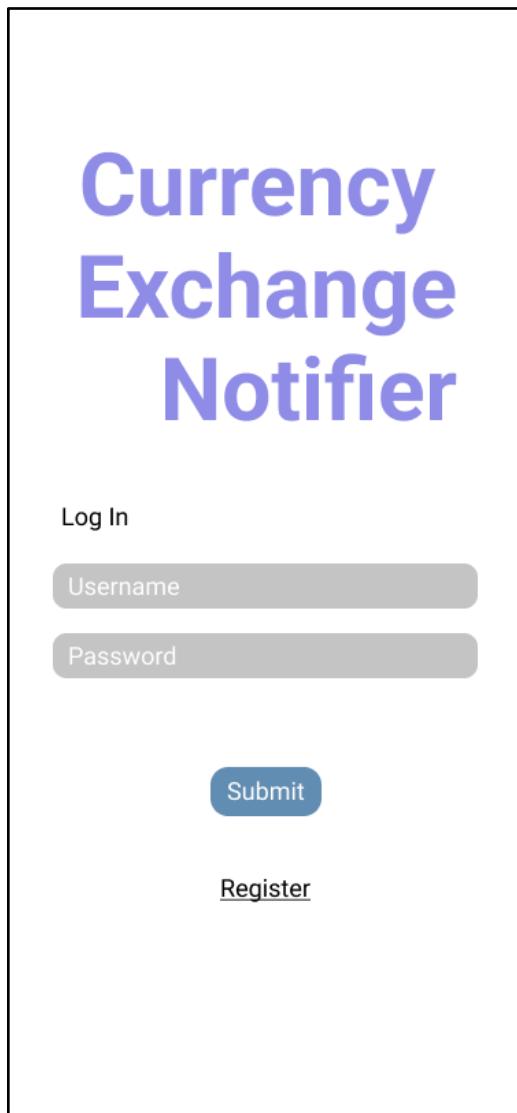
See Appendix A for the interaction design of the mock up.

**Feature:** Log in

**Description:** This is the initial page that the user sees when they open the application. It displays the logo of the system and the login form where users enter their username and password. The input boxes have a grey background colour to contrast against the background and are labelled with placeholders to inform the user of what to enter into each input field. Once the user inputs their login credentials they can proceed to the next stage by clicking the blue 'Submit' button. I tried not to use more than 3 colours and all of them were neutral so it is not too distracting. If the user does not have an account with the system already, they can click the 'Register' button which will redirect them to the sign up page. In the sign up page, the user can create a new account by entering a username, password and retype the password for confirmation. The colour scheme remains the same, and there is a back button on the top so the user can go back to the initial login page.

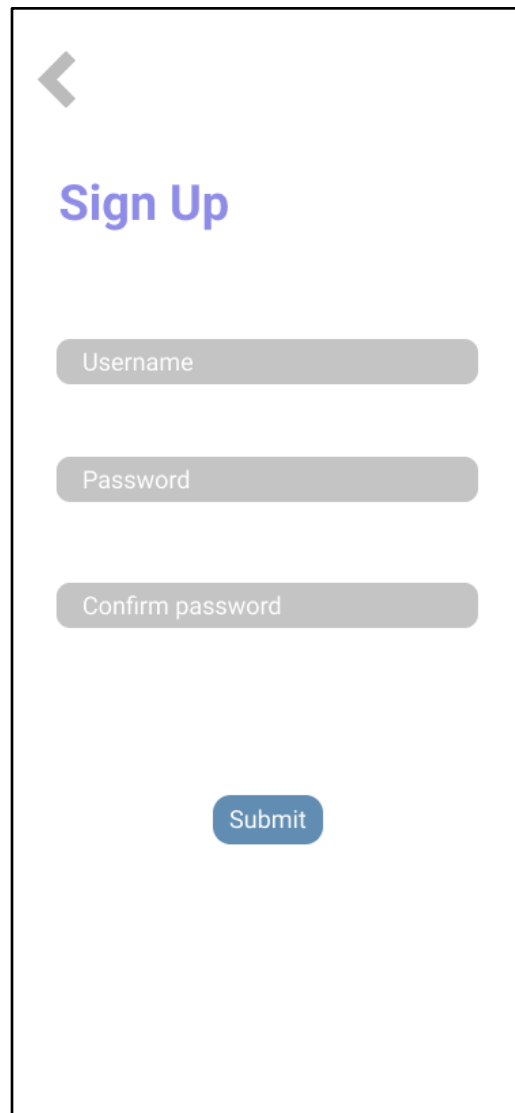
**Use Case(s):** UC1

**Screenshots:**



The login page design features a large, bold title "Currency Exchange Notifier" in purple at the top. Below the title, the text "Log In" is displayed. There are two input fields: "Username" and "Password", both with light gray backgrounds. A blue "Submit" button is positioned below the input fields. At the bottom, there is a link labeled "Register" in purple.

Figure 1: Design of the log in page



The sign up page design includes a back arrow icon in the top left corner. The title "Sign Up" is displayed in purple. There are three input fields: "Username", "Password", and "Confirm password", all with light gray backgrounds. A blue "Submit" button is located at the bottom of the form.

Figure 2: Design of the sign up page

## Feature: Set default/target currency

**Description:** This is the page where the user sets their default and target currency. The user can browse through a list of currencies and choose their default and target currency. You can see the initial design on the left and the improved design on the middle and right. Some users commented that they find it hard to scroll through the whole list of currencies, and multiple 'Select' buttons make the page look cluttered. So instead I changed the list of currencies to the dropdown list and it will display the list of currencies in alphabetical order when it comes to the actual implementation. Also regarding the flags, some currencies are used in more than one country so it is hard to decide which flag to display therefore I decided it may be better to not include them. I also added a caption informing the user that they can change the default currency at a later time via the settings. In the improved design, I tried to keep a simple design with minimum information as possible so the user can focus on the one feature they need to. The colour scheme remains the same throughout for consistency.

**Use case(s):** UC2, UC3

### Screenshots:

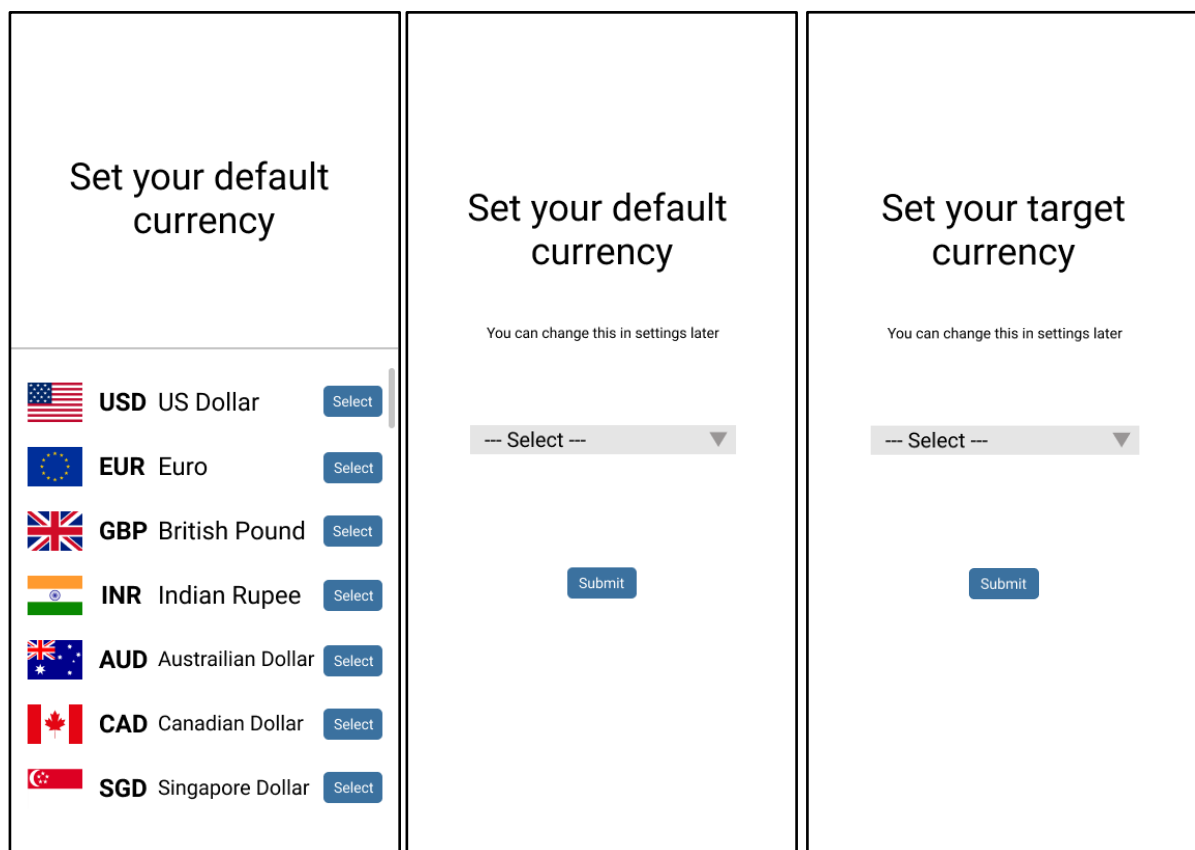


Figure 3: Initial design of the currency selecting page

Figure 4: Design of the default currency setting page

Figure 5: Design of the target currency setting page



**Feature:** Currency conversion

**Description:** This page allows the user to convert their target currency to their default currency in real time. The left screenshot shows where the user enters the value to see the conversion. I included the example on the right hand side, so in this case it shows the conversion between Pound Sterling (GBP) and US Dollar (USD). This page could further be improved by changing the colour of the bell icon because it is not very distinctive. However I could not change the colour of the icon in Figma due to the need of a premium account, but the colour of this icon will be changed during implementation.

**Use case(s):** UC4

**Screenshots:**

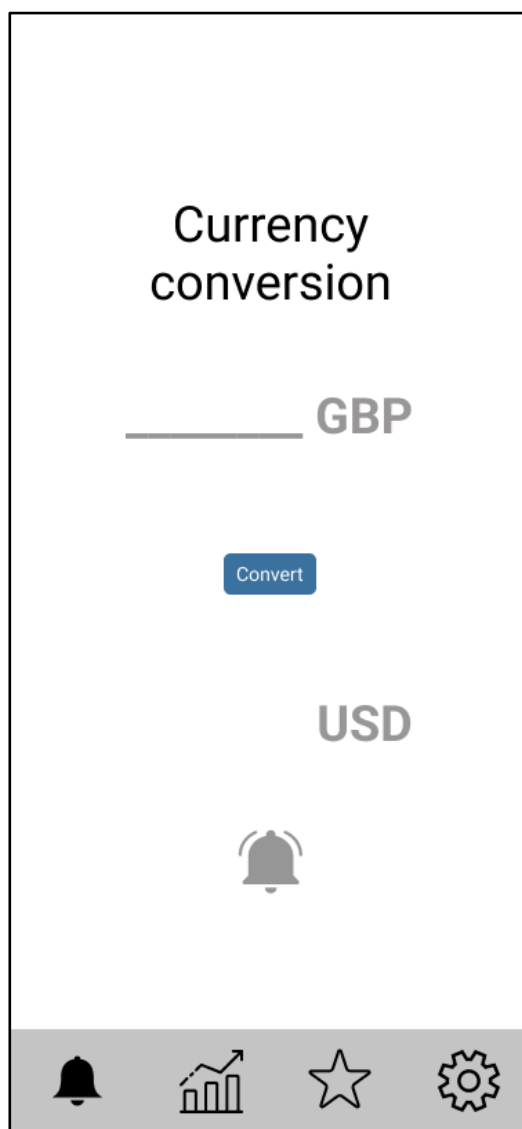


Figure 6: Design of the currency conversion page

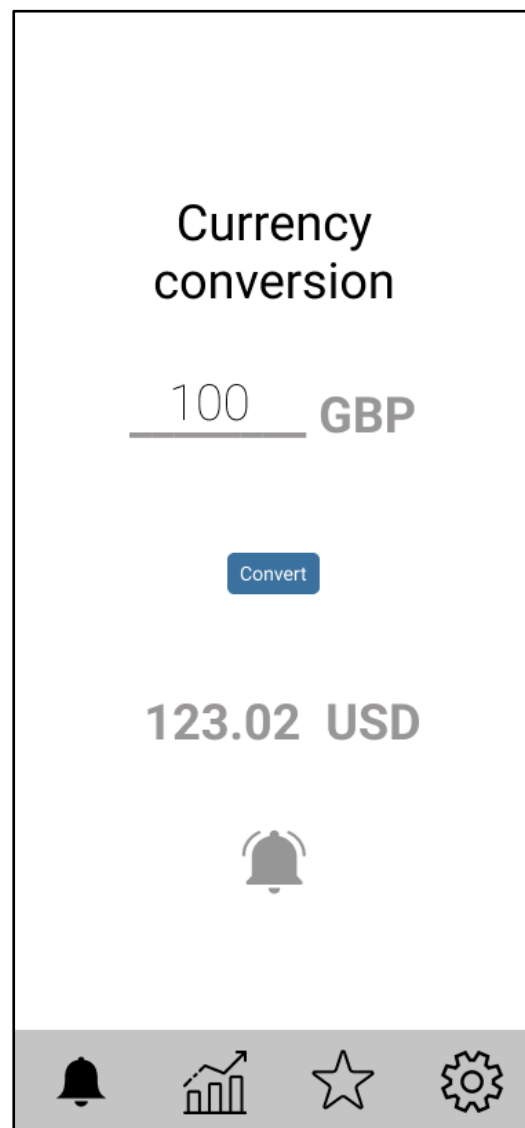


Figure 7: Design of the currency conversion page with an example

**Feature:** Set up the notification

**Description:** This is the main function of the application. On this page the user can enter their desired exchange rate and set up the notification. I highlighted the currencies and values in bold blue. To this page, I added number keypads so the user can enter their desired value easily. Although it is not included in the use case, I integrated one of my 'could have' functional requirements, which allows the user to receive email notifications when the currency is dropped. On the bottom there is a navigation bar in which the user can navigate through different features of the system.

**Use case(s):** UC5

**Screenshots:**

Send a notification  
when

**GBP £100.00**

=

**USD \$** \_\_\_\_\_

or below

☐ I would also like to receive email notifications

7	8	9
4	5	6
1	2	3
C	0	✓

Figure 8: Design of the notification set up page

**Feature:** Settings

**Description:** This is the settings page. The user can reach this page by clicking the settings icon in the navigation bar. There are different setting options including general, display, notification, languages, etc. you could say how the notification bar at the bottom allows the user to leave the page and talk about the design of this page a bit more

**Use case(s):** UC6, UC7, UC8

**Screenshots:**

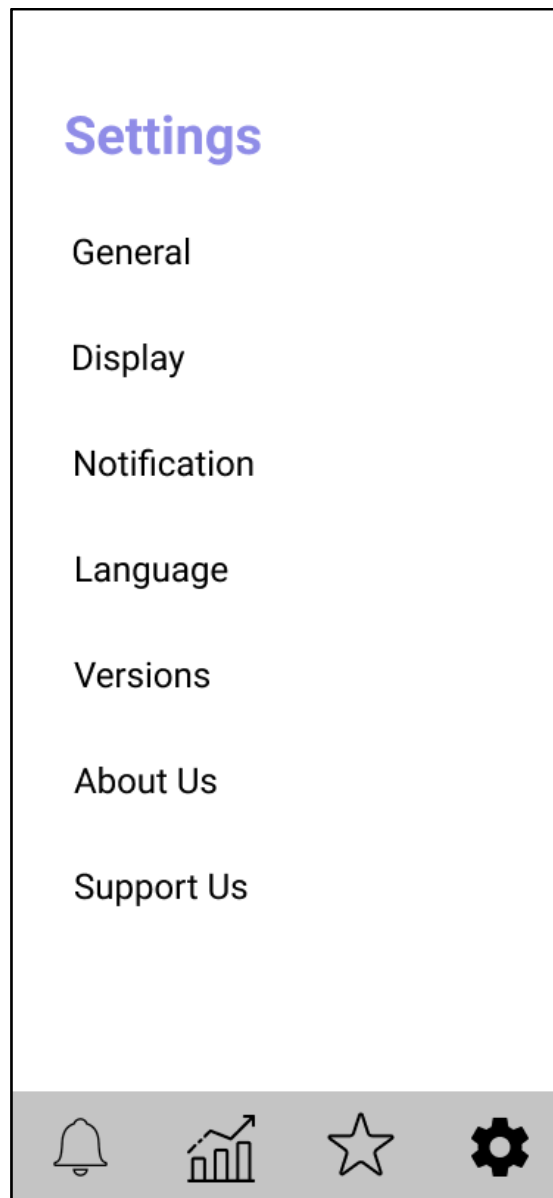


Figure 9: Design of the settings page

**Feature:** General settings

**Description:** This is the page where the user can control their refresh cycle and change their default and target currency. Since this page is a subordinate page, the navigation bar is unnecessary as the user can go back to the previous page by clicking the back button on the top left corner.

**Use case(s):** UC6, UC7

**Screenshots:**

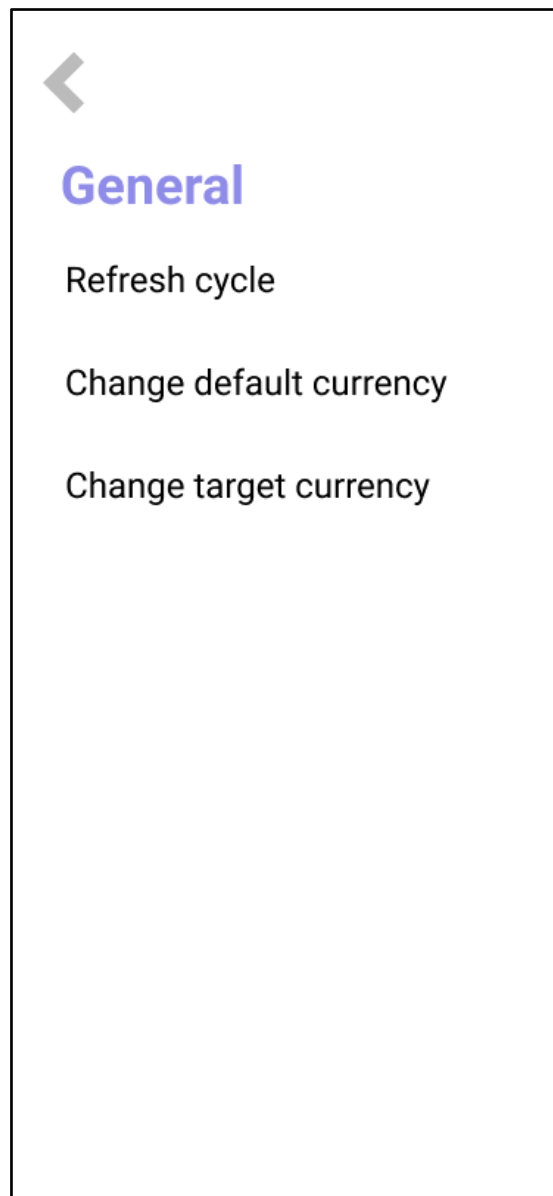


Figure 10: Design of the general settings page

**Feature:** Refresh cycle

**Description:** On this page the user can change the refresh cycle of the system. The default setting is 1 hour, this is because normally most currency conversion apps offer hourly updates and I do not want the user to receive a series of notifications in a short period of time unless they have chosen to do so. I added a little explanation below the title so the user knows what the refresh cycle is.

**Use case(s):** UC7

**Screenshots:**

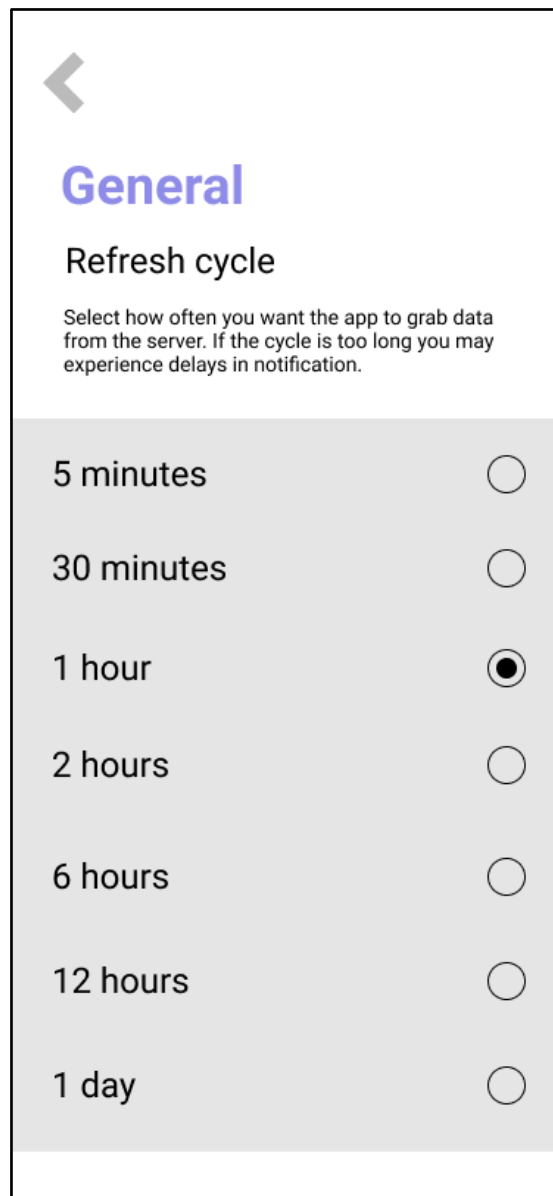


Figure 11: Example of the refresh cycle

**Feature:** Change default and target currencies

**Description:** On this page the user can change their default and target currencies. These pages have the same structure as the set default/target currency pages, so the user is familiar with the design. If the user changes their mind and wishes to go back to the previous settings page they can simply click the back button.

**Use case(s):** UC6

**Screenshots:**

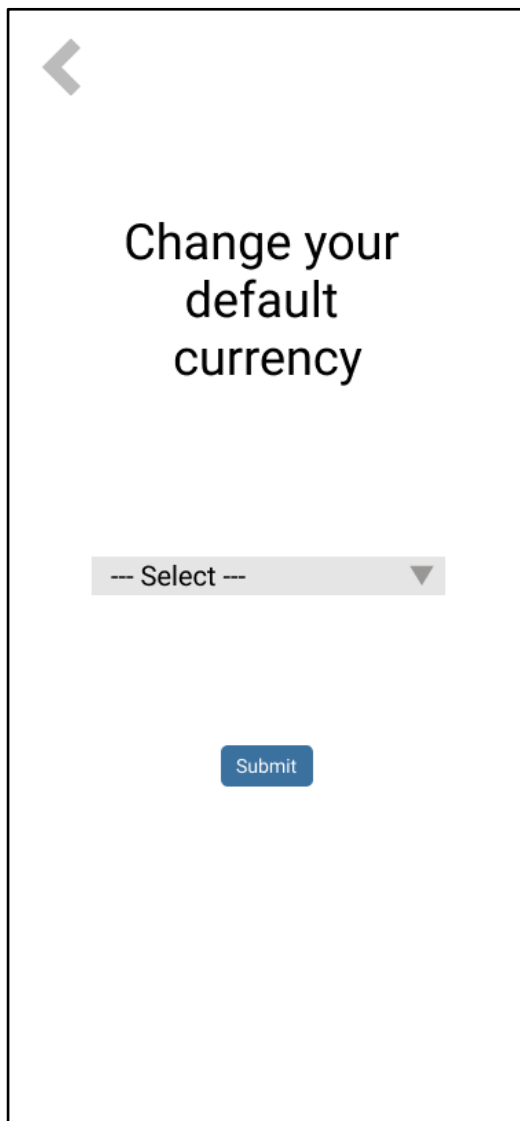
A mobile app screen with a white background. At the top left is a grey back arrow icon. In the center, the text "Change your default currency" is displayed in a large, black, sans-serif font. Below this text is a grey dropdown menu with the text "-- Select --" and a small downward-pointing triangle on the right. At the bottom center is a blue rectangular button with the word "Submit" in white text.

Figure 12: Changing default currency page

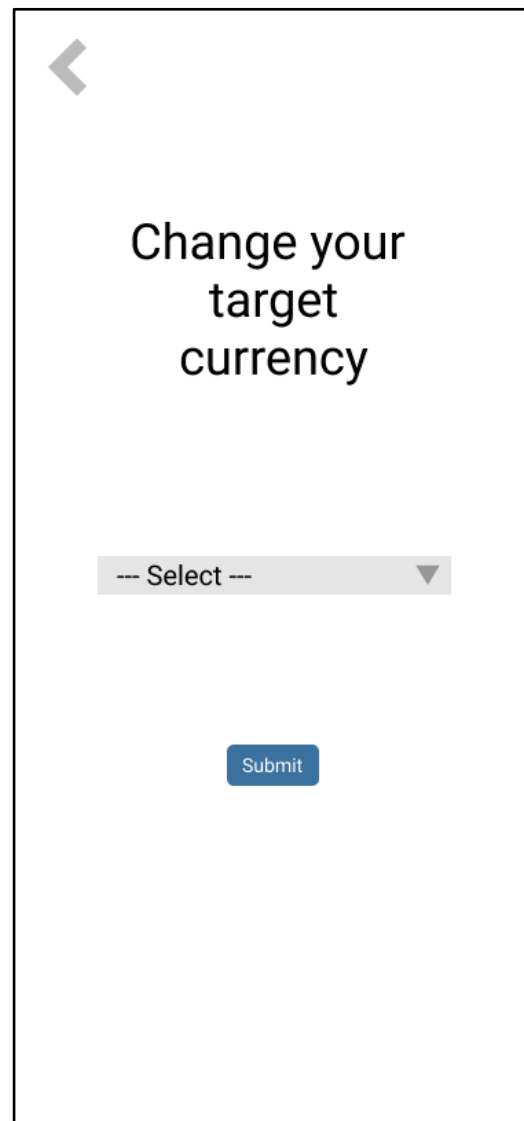
A mobile app screen with a white background. At the top left is a grey back arrow icon. In the center, the text "Change your target currency" is displayed in a large, black, sans-serif font. Below this text is a grey dropdown menu with the text "-- Select --" and a small downward-pointing triangle on the right. At the bottom center is a blue rectangular button with the word "Submit" in white text.

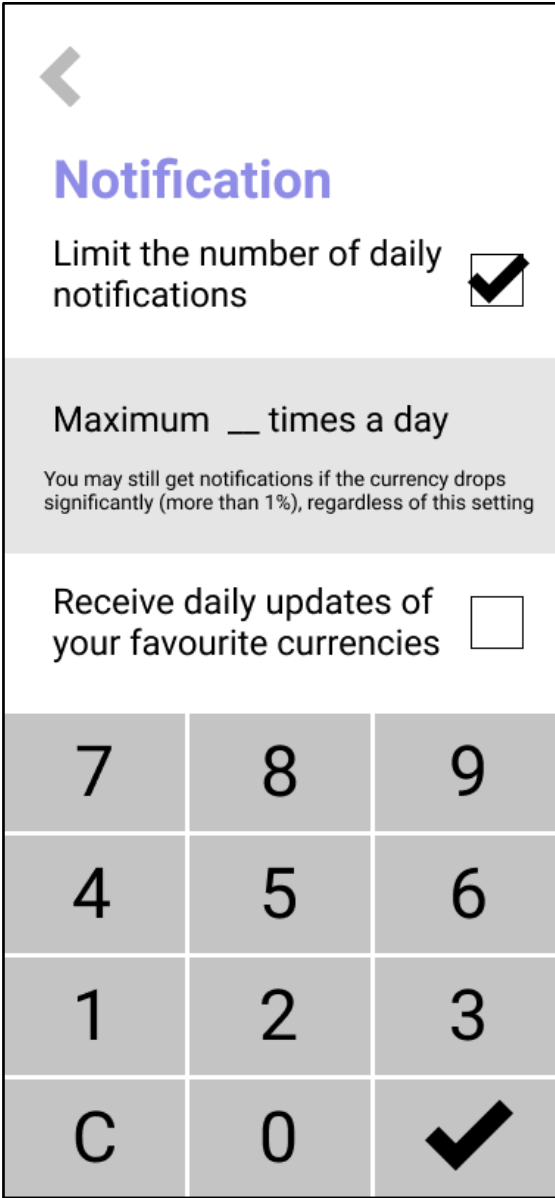
Figure 13: Changing the target currency page

**Feature:** Notification settings

Description: On this page the user can limit the number of notifications they receive per day. If the exchange rate reaches the user's desired rate multiple times in one day, the user may not want to receive notifications every time the rate is hit. Therefore this feature enables the user to control the number of notifications they receive by setting up the maximum number of the notification per day. The user can tick the tick box then enter their desired number of notifications using the keypad, and when they want to disable this feature they can simply untick the box.

**Use case(s):** UC8

**Screenshots:**



The screenshot shows a mobile app interface for notification settings. At the top is a back arrow icon. Below it is the title "Notification" in blue. The main section is titled "Limit the number of daily notifications" with a checked checkbox. Below this is a grey bar containing the text "Maximum \_\_ times a day" and a smaller note: "You may still get notifications if the currency drops significantly (more than 1%), regardless of this setting". Below the grey bar is the text "Receive daily updates of your favourite currencies" with an unchecked checkbox. At the bottom is a keypad with a 4x3 grid of buttons: the first three rows contain numbers 7-9, 4-6, and 1-3 respectively; the last row contains 'C', '0', and a checkmark icon.

7	8	9
4	5	6
1	2	3
C	0	✓

Figure 14: Design of the notification settings page

## 5. Implementation

As mentioned in my initial report, my prototype was developed in Java and XML using Android Studio. Since I have no prior experience on developing a mobile app or using Android Studio I followed tutorials on YouTube and Android Studio's official documentation. In this section, I will give an overview of the development process of the prototype of Currency Exchange Notifier and how the main functionality was developed.

### 5.1. Database creation

To store and organise my data I used SQLite database, which is a database engine that is compatible with Android Studio mobile applications. In the database the user's login credentials are stored. I created the login database in a Java class file called DB Helper. The following screenshot displays how this was achieved. If no user credentials exist, it automatically executes the table called 'users'. Then it populates the table with the data the user inputs. When the user tries to login, it checks the input in the table to see if the username is existing or the password matches with that username.

```
public class DBHelper extends SQLiteOpenHelper {

    public static final String DBNAME = "Login.db";

    public DBHelper(Context context) { super(context, name: "Login.db", factory: null, version: 1); }

    @Override
    public void onCreate(SQLiteDatabase MyDB) {
        MyDB.execSQL("create Table users(username TEXT primary key, password TEXT )");
    }

    @Override
    public void onUpgrade(SQLiteDatabase MyDB, int i, int i1) {
        MyDB.execSQL("drop Table if exists users");
    }

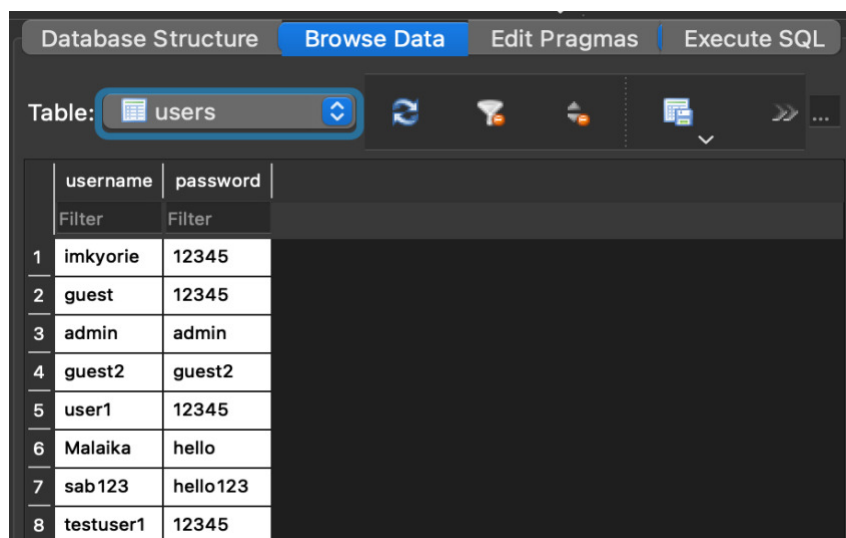
    public Boolean insertData(String username, String password) {
        SQLiteDatabase MyDB = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("username", username);
        contentValues.put("password", password);
        long results = MyDB.insert(table: "users", nullColumnHack: null, contentValues);
        if(results==-1) return false;
        else
            return true;
    }

    public Boolean checkusername(String username){
        SQLiteDatabase MyDB = this.getWritableDatabase();
        Cursor cursor = MyDB.rawQuery( sql: "Select * from users where username = ?", new String[] {username});
        if(cursor.getCount()>0)
            return true;
        else
            return false;
    }
}
```

Figure 15: Snippet of code showing database creation



Users' credentials can be browsed through the DB Browser.



The screenshot shows a database browser window with tabs for 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. The 'Browse Data' tab is active, and the 'users' table is selected. The table has two columns: 'username' and 'password'. Below the table, there are filter inputs for each column. The table contains 8 rows of data, numbered 1 through 8.

	username	password
	Filter	Filter
1	imkyorie	12345
2	guest	12345
3	admin	admin
4	guest2	guest2
5	user1	12345
6	Malaika	hello
7	sab123	hello123
8	testuser1	12345

Figure 16: Snippet of the database

## 5.2. Overview of implementation structure

The screenshot below (Figure 17) shows the overall structure of the directories and files. Within the main directory you can find folders java and res; the Java includes all the classes responsible for each activity. These activities are in relation to each other allowing the user to browse different activities back and forth. Each activity has a corresponding xml file, which is responsible for the interface design of the system. Lastly, the AndroidManifest.xml file is the file that contains the project source set. This file holds essential information about the project and delivers this information to Android devices, Google Play Store and Android developing tools such as Android Studio.

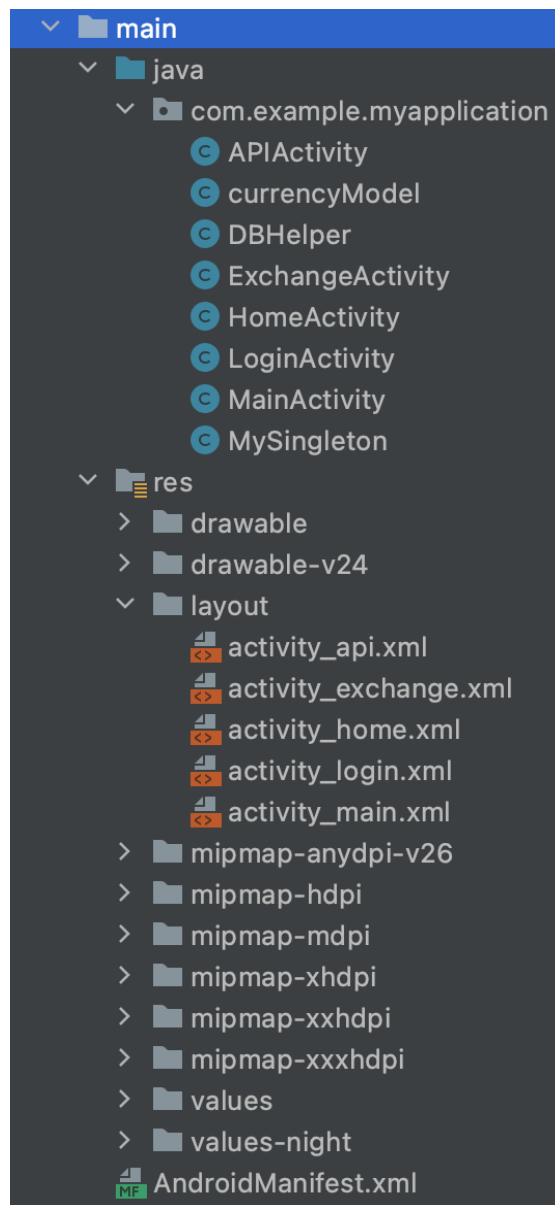


Figure 17: Snippet of the list of directories

The excerpt of code below is responsible for the sign up functionality of my application. Originally in my design mock up I set the login function to be the initial page, but upon implementation I decided to change it to the signup page. This is because the users will not have yet created an account when they install the application for the first time, and once they are logged in the app will remember the user's login credentials so they do not have to log back in every time they enter the system. Also the app does not require any personal details such as the user's name, nationality or phone number to sign up as it was unnecessary for the application and I did not want to collect users' personal data. The figures 18 and 19 show the code for my sign up and log in page. They follow the same structure, both operate with if statements to check if the input information matches with the data stored in the database, then it goes through a series of if statements to see if the user input is valid.

```
signup.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
  
        String user = username.getText().toString();           //stores the input to string  
        String pass = password.getText().toString();  
        String repass = repassword.getText().toString();  
  
        if(user.equals("")||pass.equals("")||repass.equals("")) //checking if any of the fields are empty  
            Toast.makeText( context: MainActivity.this, text: "Please enter all the fields", Toast.LENGTH_SHORT).show();  
        else{  
            if(pass.equals(repass)) { //checking if the passwords are matching  
                Boolean checkuser = DB.checkusername(user);  
                if(checkuser==false){  
                    Boolean insert = DB.insertData(user, pass);  
                    if(insert==true){  
                        Toast.makeText( context: MainActivity.this, text: "Registered Successfully", Toast.LENGTH_SHORT).show();  
                        Intent intent = new Intent(getApplicationContext(), HomeActivity.class);  
                        startActivity(intent);  
                    }else{  
                        Toast.makeText( context: MainActivity.this, text: "Registration failed", Toast.LENGTH_SHORT).show();  
                    }  
                }else{  
                    Toast.makeText( context: MainActivity.this, text: "User already exist! Please sign in", Toast.LENGTH_SHORT).show();  
                }  
            }else{  
                Toast.makeText( context: MainActivity.this, text: "Passwords not matching", Toast.LENGTH_SHORT).show();  
            }  
        }  
    }  
});
```

Figure 18: Snippet of the sign up function

```

btnlogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String user = username.getText().toString();
        String pass = password.getText().toString();

        if (user.equals("") || pass.equals(""))
            Toast.makeText( context: LoginActivity.this, text: "Please enter all the fields", Toast.LENGTH_SHORT).show();
        else{
            Boolean checkuserpass = DB.checkusernamepassword(user,pass);
            if(checkuserpass==true){
                Toast.makeText( context: LoginActivity.this, text: "Sign in successful", Toast.LENGTH_SHORT).show();
                Intent home = new Intent(getApplicationContext(),HomeActivity.class); //redirect to home activity
                startActivity(home);
            }else{
                Toast.makeText( context: LoginActivity.this, text: "Invalid credentials", Toast.LENGTH_SHORT).show();
            }
        }
    }
});

```

Figure 19: Snippet of the log in function

The figures 20 and 21 are examples of the XML code for the sign up page. TextView is a text that shows up on the screen, EditText is a text field that allows the user to enter the value and the button widget is an element that allows the user to perform an action.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/signup"
        android:text="Sign Up"
        android:textColor="@color/black"
        android:textSize="35dp"
        android:textStyle="bold"
        android:layout_margin="50dp"
        android:gravity="center"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/username"
        android:layout_below="@id/signup"
        android:background="#95C4C4"
        android:hint="Username"
        android:textColorHint="@color/black"
        android:textColor="@color/black"
        android:layout_margin="18dp"
        android:padding="20dp"
    />

```

Figure 20: Snippet of the sign up design – TextView and EditText

```

<Button
    android:id="@+id/btnsignup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/repassword"
    android:layout_centerHorizontal="true"
    android:layout_marginStart="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="20dp"
    android:layout_marginBottom="20dp"
    android:backgroundTint="#3B719F"
    android:text="REGISTER" />

```

Figure 21: Snippet of the sign up design - button

Figures 22 and 23 shows the implemented prototype of sign up and log in page. The log in button on Figure 22 directs the user to the log in page. A back button is not present on the designs because that function is included in the Android emulator. In Figure 23 you can see how the username and password is displayed, and an example of a toast message that appears if the user enters invalid credentials is shown on the bottom of the screen.

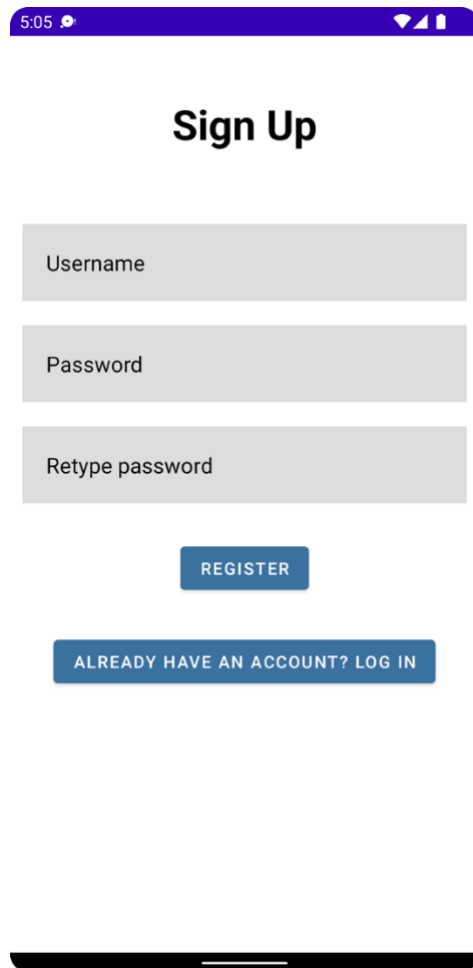


Figure 22: Screenshot of the sign up page

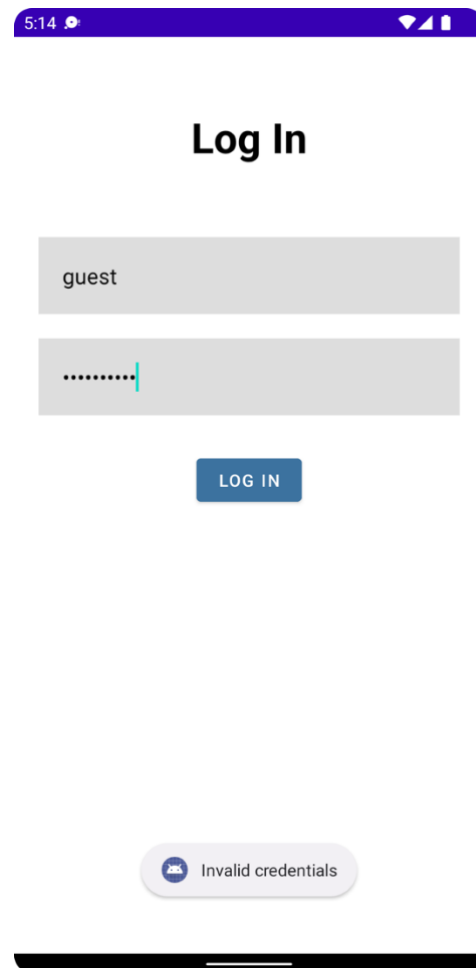


Figure 23: Screenshot of the log in page

The screenshot below shows the Java code for setting a default currency activity. In this method I implemented a spinner and populated it with more than 170 currencies. The same function structure is applied for setting the target currency activity because it is essentially doing the same execution.

```

public class HomeActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {

    Button submitbtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        Spinner spinner = findViewById(R.id.defaultCurrencySpinner);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(context, this, R.array.currencies, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemClickListener(this);

        submitbtn = (Button) findViewById(R.id.submitbtn);

        submitbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Intent exchange = new Intent(getApplicationContext(), ExchangeActivity.class); //Direct to exchange activity
                startActivity(exchange);

            }
        });
    }
}

```

Figure 24: Snippet of the default currency activity

This is an example code of a spinner in xml.

```

<Spinner
    android:id="@+id/defaultCurrencySpinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="300dp"
    android:layout_marginRight="20dp" />

```

Figure 25: Snippet of the spinner

In Figures 26 and 27 you can see how the spinner is displayed. I followed the convention 'currency code: currency name' as this was how they were displayed in the Currency API. I also sorted the currency codes in alphabetical order, as I mentioned earlier in the design section in page 24. The user can scroll through the list of currencies.

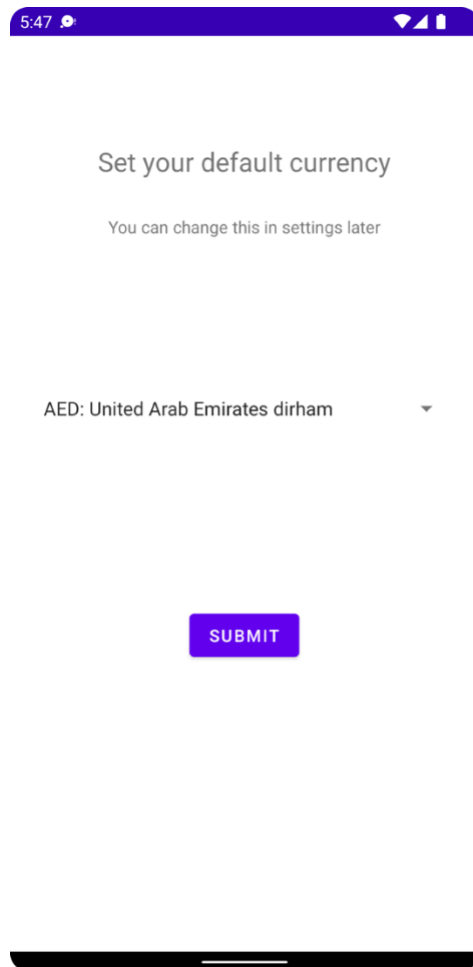


Figure 26: Screenshot of the default currency page

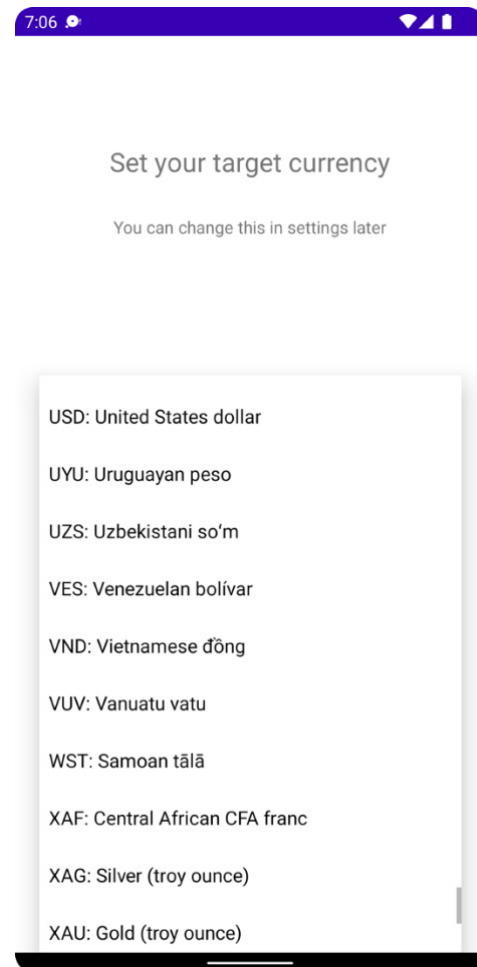


Figure 27: Screenshot of the target currency page

Figure 28 shows a conversion activity where the API retrieves the data to convert the currency in real time currency exchange rates. String URL makes the API request to the server with the default and target currency code that the user puts in. Figure 29 shows you the example JSON response from the server.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_api);

    String defaultCurrencyCode = HomeActivity.defaultText.substring(0,3);
    String targetCurrencyCode = ExchangeActivity.targetText.substring(0,3);
    amount = (EditText)findViewById(R.id.amount);
    convertbtn = (Button)findViewById(R.id.convertbtn);

    convertbtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String value = amount.getText().toString();

            // Instantiate the RequestQueue.
            String url = "https://api.getgeopapi.com/v2/currency/convert?api_key=bf208d8001d5c0bd49453387c7a8ec46a76a3e855f&from=" + targetCurrencyCode
            + "&to=" + defaultCurrencyCode + "&amount=" + value + "&format=json";

            JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null, new Response.Listener<JsonObject>() {
                @Override
                public void onResponse(JsonObject response) {

                    String defaultCurrency = null; //displaying default currency

                    try {
                        JsonObject rates = response.getJSONObject("rates");
                        currencyModel one = new currencyModel();

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                    Toast.makeText(context: APIActivity.this, response.toString(), Toast.LENGTH_SHORT).show(); //displaying the "base_currency_code" of json object
                }
            });
        }
    });
}

```

Figure 28: Snippet of code in the conversion activity

JSON	Raw Data	Headers
Save	Copy	Collapse All Expand All Filter JSON
base_currency_code:	"EUR"	
base_currency_name:	"Euro"	
amount:	"10.0000"	
updated_date:	"2022-05-11"	
▼ rates:		
▼ GBP:		
currency_name:	"Pound sterling"	
rate:	"0.8552"	
rate_for_amount:	"8.5522"	
status:	"success"	

Figure 29: Example JSON response from the server

You can see how this feature is integrated in the figure below. It takes in the user input and converts it to the real time currency exchange rate.



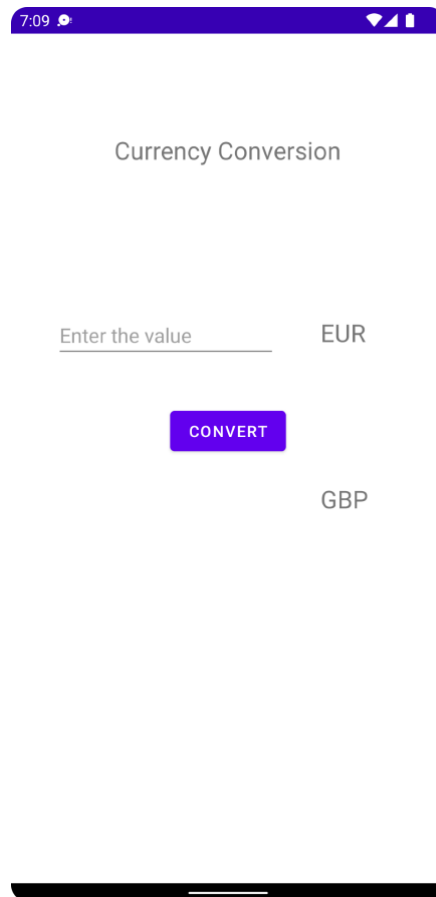


Figure 30: Screenshot of the currency conversion page

```
{"base_currency_code":"EUR","base_currency_name":"Euro","amount":"10.0000","updated_date":"2022-05-11","rates":{"GBP":{"currency_name":"Pound sterling","rate":"0.8552","rate_for_amount":"8.5522"}}, "status":"success"}
```

Figure 31: Screenshot of the JSON response

Although I failed to show the converted value for the amount that the user put in, I confirmed that the data was successfully retrieved from the server as shown in the figures below. However I struggled to pull out the 'rate\_for\_amount' value from the block of JSON code. This is as far as the implementation has reached. The detailed explanation of the remaining work can be found in the future work section.

### 5.3. Code quality

To produce a system that is efficient and accessible, I created a code in a simple way and refactor it to make it better for a more scalable program. One example for this is implementing a singleton design pattern which I learnt in 'CM2307 Object Orientation, Algorithms and Data Structures' module in 2nd year. Singleton class ensures that only one object gets created at a time, allowing the program to access the object directly without instantiating the whole object of the class.

I also maintained a consistent style of code writing, kept most of the code self-explanatory and documented the lines where it needs further explanation.

## 6.Results and Evaluation

### 6.1. Test Cases

The following table shows test results for the prototype of Currency Exchange Notifier.

For full test cases, see Appendix B.

Test ID	Related requirement	Pass/Fail
1	FR1	Pass
2	FR1	Pass
3	FR2	Pass
4	FR3	Pass
5	FR4	Partial pass
6	FR5	Fail
7	FR6	Fail
8	FR7	Fail
9	FR8	Fail
10	FR9	Fail
11	FR10	Fail
12	FR11	Fail
13	FR12	Fail
14	FR13	Fail
15	FR14	Fail

The first three test cases regarding sign up, setting default and target currencies have passed. These are the functions I implemented successfully, all of them meet the initial requirements and the acceptance criteria. Test case 4 was a partial pass because one of the acceptance criteria was not met.

The rest of the test cases have failed, this is due to time constraints which meant the related features were not implemented in time.

## 6.2. User Testing

The User testing consists of 2 parts - quantitative and qualitative user evaluation methods. I recruited 5 Cardiff University Computer Science students to test my prototype.

The qualitative research was performed whilst the test user was carrying out given tasks under my supervision. None of the participants made mistakes when performing the tasks, and it took them on average of 1m 3s to create an account and convert the currencies. All of the participants clearly understood the sign in page, and quickly noticed how the spinners and text fields work. Participants commented that the system is straightforward to use, and the design was nice and clear, 'making it pretty obvious what you have to do'. One participant said that scrolling through the spinner was inefficient; this feature could be improved by changing the spinner to a search bar. This improvement can also address the following issue that was spotted - some participants struggled to find some currencies because the spinner is sorted in alphabetical order of the currency code, if they do not know the currency code of the given currency it is time consuming to go through each currency. The last feedback I received was that the definition of 'default' and 'target' currency was confusing. This could be improved if the definitions of default and target currency were specified.

After the testing, the participants were given a questionnaire to complete. The questionnaire featured all questions that allowed me to calculate the SUS (System Usability Score) for the prototype. It consists of 10 questions with a Likert scale from strongly agree to strongly disagree. The average SUS for my prototype was calculated to be 83.75, considering the fact that the average SUS in the industry is 68 (Sauro, 2016), this proves that my prototype is highly usable. One thing to point out is that because the test users were all Computer Science students who are familiar with the structures of the conventions of applications like this, it might have affected the SUS value being high.

## 7. Future work

Although my implementation has not reached as far as I wished, integrating a few things could make the prototype minimum viable product.

### 1. Making the converted currency value to appear

This can be done by extracting the corresponding value from the JSON response. I tried to complete this by creating a separate class called `currencyModel` which saves data fields as an array list. However my approach was not very successful, there could be another way of achieving this. Once the converted value of the currency comes through functional requirement 4 will be implemented.

### 2. The main function for the notification feature

As this function was the motivation behind this app development, it should be prioritised in the further implementation once the implementation of the above function is achieved. As shown in Figures 6 and 7 in the design section of this report I would create a notification bell icon on the currency conversion page which acts as a button to direct the user to the notification set up page. This page will be a separate activity in Java class which will have the layout of Figure 8 shown in the design section, which will allow the user to input their desired value of currency exchange rate for the currency pair they selected. Then the user input is taken to compare the real time currency value. Once the rate is hit, a push notification will come through the emulator so that the user will be able to know immediately. By doing this the acceptance criteria for the functional requirement No.5 will be accomplished and the main notification functionality for Currency Exchange Notifier will be achieved.

### 3. Navigation bar on the bottom

One thing that could easily be done is the implementation of the navigation bar on the bottom of the displayed screen. This will help the users navigate through different activities of the system, also fulfilling one of the acceptance criteria of non-functional requirement No.1 - the system must be easy to navigate. I found various tutorials for executing this function but due to time limitations and the difficulties I faced trying to implement the functional requirements, I was unable to develop this function. .

### 4. Changing default and target currencies

A function that is responsible for changing the user's default and target currencies should be implemented in a similar way that they were set up in the first place. This can easily be completed within a separate class within the settings.

Once the above functions are executed the prototype will become a minimum viable product. The prototype can furthermore be improved by implementing several 'should have' and 'could have' functional requirements. For example, 'FR8: the system should have a feature that limits the number of notifications per day' could be implemented by setting a maximum value based on the user input and then ignoring calls that come through after, once the limit is reached. 'FR9: the system should allow the users to favourite certain currencies' - this can either be done by 1) making a list where the user can add and remove currencies from or by 2) creating a dedicated currency page for each currency so that the user can 'star' the currency. The second option will be slightly harder to implement but in this way 'FR12: news and information feature' can also be easily accomplished, since it has a dedicated currency information page it will be easier to display information specific to that currency. One of the 'could have' requirements in which the system could show a historical currency chart would be helpful for the user to see the general trend of the given currency pair. This could also be implemented as the Currency API offers historical currency data available from the server and Android Studio supports Graphview library. Last but not least, FR14 which specifies that the system could be supported in different languages would be extremely useful if implemented because the target audience for this application writes this better.

## 8. Conclusions

To determine whether the project went successful I revised the aims and objectives I set at the beginning of this project. There were three main objectives which were 1) to gain an understanding of the currency fluctuations and existing solutions 2) to implement features that help users to exchange currencies at the right time and 3) to create a suitable UX design.

The first objective was reached through the background research section of the project. From the point I submitted my project proposal last semester, I observed the currency fluctuations by regularly monitoring the currencies that are considered to be main currencies (US Dollar, Pound sterling, Euros) and the currency that is personally relevant to me (South Korean Won). I gained an in depth understanding of the general trend of these currencies such as how political affairs can impact currencies and how the fluctuation of one currency causes the fluctuation of the other currencies. I also analysed and reviewed the existing solutions in the market, which helped me figure out the problems and limitations and produce solutions of how to address them.

The second objective however, was not as successful as I intended. My lack of experience in development in Android Studios and time restrictions made it rather difficult for me to deliver a fully functioning prototype in the given time. According to 'the table that determines in which category your app falls in' (Jain, 2018), my app is classed as a large version app with the features I intended to implement including push notifications and multilingual properties. On average, a 'big sized app' can take up to 20-22 weeks solely to develop and test (Jain, 2018), which was almost double the time I had for this project. Due to the time constraints this aim failed to be achieved as I produced a partially functioning prototype.

The final objective of the project was creating a suitable UX design and this was achieved through the mock-up designs I produced in Figma. I focused on generating a high-fidelity design with intuitive UX design by improving my design several times based on the user feedback. This objective could have been further accomplished by applying these designs to the actual prototype.

## 9. Reflection on Learning

From the planning to the delivery of the product, the entire project has provided me insight in many ways. First of all, this project allowed me to gain experience in Android app development, which was a huge learning curve. I also learnt how to code in XML, and also improved my Java coding skills. I never used to be entirely confident about my coding skills but through this project I realised that coding is not just always hard, once an error is resolved it comes with a high degree of satisfaction. Also, the project required me to present various skills beyond academic abilities, such as planning, organising, time management and problem solving.

Although I planned and prepared for the expected difficulties due to my lack of experience, in mobile app development there were more obstacles in my learning than I had anticipated. My first experience with Android Studio taught me that the next time I ever get an opportunity to develop a prototype with a completely new IDE I will undertake a series of detailed tutorials in advance so I can be prepared for any unforeseen problems, and make sure I have more than enough time to deliver the product. The biggest issue with my project was time management, if I organised my time better I would have been able to complete my project to a higher standard.

As an international student I often find it difficult finding the right time to exchange currency due to the impact of the exchange rate fluctuation. Therefore the idea of Currency Exchange Notifier was purely based on my own problem, which made me more passionate about the project. Although I may have failed to deliver a fully functioning prototype I believe that I successfully presented the skills and techniques I gained through the 3 years of learning from my Computer Science degree. Considering I never coded before starting this course it is a huge improvement that I produced a partially functioning prototype of a mobile application by myself. This invaluable learning experience will be a solid foundation for any future work I do.



# References

AllCodingTutorials. 2020. *Login and Register Form using SQLite Database in Android Studio | login registration android studio*. Available at: <https://youtu.be/8obgNNlj3Eo> [Accessed: 7 March 2022]

Android Developers. 2020. *Android Developer Guide*. Available at: <https://developer.android.com/docs> [Accessed 15 February 2020].

Coding in Flow. 2017. *Text Spinner - Android Studio Tutorial*. Available at: [https://youtu.be/on\\_OrrX7Nw4](https://youtu.be/on_OrrX7Nw4) [Accessed: 29 March 2022]

CodingWithMitch. 2017. *Android Beginner Tutorial #2 - TextViews [Displaying information on the screen]*. Available at: <https://youtu.be/VkCeWHa4EH0> [Accessed: 12 April 2022].

Conotoxia.com. 2020. *How many currencies are there in the world? Famous and those not so well known*. Available at: <https://conotoxia.com/news/blog/worth-knowing/how-many-currencies-are-there-in-the-world-famous-and-those-not-so-well-known> [Accessed: 1 March 2022].

CurrencyFair. 2015. *CurrencyFair Money Transfer* [iOS app]. Version 6.3.0. CurrencyFair. [Accessed: 24 February 2022].

freeCodeCamp.org. 2020. *Java Android App using REST API - Network Data in Android Course*. Available at: <https://youtu.be/xPi-z3nOcn8> [Accessed: 7 April 2022].

Easy Tuto. 2021. *Simple Login App in Android Studio | 2022*. Available at: [https://youtu.be/sOJRJtM\\_iu0](https://youtu.be/sOJRJtM_iu0) [Accessed: 1 March 2022]

Jain, P. 2018. *How Long does It take to Build an App*. Available at: <https://www.startupgrind.com/blog/how-long-does-it-take-to-build-an-app/> [Accessed: 11 May 2022].

Sauro, J. 2016. *Measuring Usability With The System Usability Scale (SUS)*. Available at: <https://www.userfocus.co.uk/articles/measuring-usability-with-the-SUS.html> [Accessed: 11 May 2022].

Wise. 2014. *Wise* [iOS app]. Version 7.46.2. Wise Payments Ltd. [Accessed: 24 February 2022].

XE. 2009. *Xe Currency & Money Transfers* [iOS app]. Version 7.8.1. XE.com Inc. [Accessed: 24 February 2022].

# Appendices

## Appendix A: Interaction design of the mock up



## Appendix B: Test cases

<b>Test Case ID:</b> TC1		<b>Related Requirements:</b> FR1
<b>Test Description:</b>	Verify that the user can sign in to the system securely	
<b>Prerequisites:</b>	User is not registered to the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. Open application</li> <li>2. User enters the test data to the required fields</li> <li>3. Select 'register' button</li> </ol>	
<b>Test Data:</b>	<ol style="list-style-type: none"> <li>1. User details: Username: testuser1 Password: 12345 Retype password: 12345</li> <li>2. User details: Username: testuser2 Password: 12345 Password: 1234</li> </ol>	
<b>Expected Results:</b>	<ol style="list-style-type: none"> <li>1. The accounts will be created and stored in the database. The toast message 'Registered Successfully' will be displayed and the default currency page will be displayed.</li> <li>2. The toast message 'Passwords not matching' will be displayed.</li> </ol>	
<b>Actual Results:</b>	<ol style="list-style-type: none"> <li>1. As expected.</li> <li>2. As expected.</li> </ol>	
<b>Comments:</b>	N/A	
<b>Test Status:</b>	PASS	

<b>Test Case ID:</b> TC2		<b>Related Requirements:</b> FR1
<b>Test Description:</b>	Verify that the user can log in to the system	
<b>Prerequisites:</b>	User credentials are registered to the system User is not logged in	

<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. Open application</li> <li>2. User clicks 'Already have an account? Log in' button</li> <li>3. User enters the test data to the required fields</li> <li>4. Select log in button</li> </ol>
<b>Test Data:</b>	<ol style="list-style-type: none"> <li>1. User details: Username: testuser1 Password: 12345</li> <li>2. User details: Username: testuser1 Password: 1234</li> </ol>
<b>Expected Results:</b>	<ol style="list-style-type: none"> <li>1. The toast message 'Sign in successful' will be displayed and the user is directed to the default currency page.</li> <li>2. The toast message 'Invalid credentials' will be displayed.</li> </ol>
<b>Actual Results:</b>	<ol style="list-style-type: none"> <li>1. As expected.</li> <li>2. As expected.</li> </ol>
<b>Comments:</b>	N/A
<b>Test Status:</b>	PASS

<b>Test Case ID:</b> TC3	<b>Related Requirements:</b> FR2
<b>Test Description:</b>	Verify that the user can set default currency
<b>Prerequisites:</b>	User is logged into the system
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the spinner</li> <li>2. User selects a currency</li> <li>3. User clicks submit button</li> </ol>
<b>Test Data:</b>	GBP as a default currency
<b>Expected Results:</b>	Toast message 'GBP: Pound Sterling' will be displayed when currency is selected. Then the user will be directed to the target currency page when they click the submit button.
<b>Actual Results:</b>	As expected.

<b>Comments:</b>	N/A
<b>Test Status:</b>	PASS

<b>Test Case ID:</b> TC4		<b>Related Requirements:</b> FR3
<b>Test Description:</b>	Verify that the user can set target currency	
<b>Prerequisites:</b>	User is logged into the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the spinner</li> <li>2. User selects a currency</li> <li>3. User clicks submit button</li> </ol>	
<b>Test Data:</b>	EUR as a target currency	
<b>Expected Results:</b>	Toast message 'EUR: Euro' will be displayed when currency is selected. Then the user will be directed to the currency conversion page when they click the submit button.	
<b>Actual Results:</b>	As expected.	
<b>Comments:</b>	N/A	
<b>Test Status:</b>	PASS	

<b>Test Case ID:</b> TC5		<b>Related Requirements:</b> FR4
<b>Test Description:</b>	Verify that the currency is converted in real time	
<b>Prerequisites:</b>	User is logged into the system User has selected the default and target currencies	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User enters the amount</li> <li>2. User clicks convert button</li> </ol>	
<b>Test Data:</b>	100 as an amount	

<b>Expected Results:</b>	The converted value will be displayed
<b>Actual Results:</b>	The value is not displayed. Instead the whole JSON response appears.
<b>Comments:</b>	The JSON response does show the converted value which proves that the API request was successful, however it is not displayed as it should be.
<b>Test Status:</b>	PARTIAL PASS

<b>Test Case ID:</b> TC6		<b>Related Requirements:</b> FR5
<b>Test Description:</b>	Verify that the user can set the notification for the selected currency pair	
<b>Prerequisites:</b>	User is logged into the system User has selected the default and target currencies User is in the currency conversion page	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the notification icon</li> <li>2. User enters the desired value</li> <li>3. User clicks submit</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	Notification will come through once the currency value reaches the goal value	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	
<b>Test Status:</b>	FAIL	

<b>Test Case ID:</b> TC7		<b>Related Requirements:</b> FR6
<b>Test Description:</b>	Verify that the user can change their default currency	

<b>Prerequisites:</b>	User is logged into the system User has selected the default and target currencies
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks settings icon on the navigation bar</li> <li>2. User clicks 'general'</li> <li>3. User clicks 'change default currency'</li> <li>4. User clicks the spinner</li> <li>5. User selects a currency</li> <li>6. User clicks submit button</li> </ol>
<b>Test Data:</b>	N/A
<b>Expected Results:</b>	The user's default currency will be changed and the user will be redirected to the general settings page
<b>Actual Results:</b>	N/A
<b>Comments:</b>	This function was not implemented in time
<b>Test Status:</b>	FAIL

<b>Test Case ID:</b> TC8		<b>Related Requirements:</b> FR7
<b>Test Description:</b>	Verify that the user can change the refresh cycle	
<b>Prerequisites:</b>	User is logged into the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks settings icon on the navigation bar</li> <li>2. User clicks 'general'</li> <li>3. User clicks 'refresh cycle'</li> <li>4. User changes refresh cycle</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	Refresh cycle will be changed	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	

<b>Test Status:</b>	FAIL
---------------------	------

<b>Test Case ID:</b> TC9		<b>Related Requirements:</b> FR8
<b>Test Description:</b>	Verify that the user can restrict the number of notifications per day	
<b>Prerequisites:</b>	User is logged into the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks settings icon on the navigation bar</li> <li>2. User clicks 'notifications'</li> <li>3. User checks the tick box for 'limit the number of daily notifications' field</li> <li>4. User enters a value</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	The number of notifications the user receive will be restricted	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	
<b>Test Status:</b>	FAIL	

<b>Test Case ID:</b> TC10		<b>Related Requirements:</b> FR9
<b>Test Description:</b>	Verify that the user can favourite currencies	
<b>Prerequisites:</b>	User is logged into the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the star icon on the navigation bar</li> <li>2. User clicks the plus icon</li> <li>3. User selects a currency</li> </ol>	
<b>Test Data:</b>	N/A	



<b>Expected Results:</b>	The selected currency will be added to the favourite currency list
<b>Actual Results:</b>	N/A
<b>Comments:</b>	This function was not implemented in time
<b>Test Status:</b>	FAIL

<b>Test Case ID:</b> TC11		<b>Related Requirements:</b> FR10
<b>Test Description:</b>	Verify that the daily rate alerts come through	
<b>Prerequisites:</b>	User is logged into the system User has at least one currency in their favourite currency list	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the setting icon on the navigation bar</li> <li>2. User clicks 'notifications'</li> <li>3. User checks the tick box 'Receive daily updates of</li> <li>4. your favourite currencies'</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	Daily updates will come through	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	
<b>Test Status:</b>	FAIL	

<b>Test Case ID:</b> TC12		<b>Related Requirements:</b> FR11
<b>Test Description:</b>	Verify that the user can see the historical chart	
<b>Prerequisites:</b>	User is logged into the system User has selected the default and target currencies	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the chart icon on the navigation bar</li> </ol>	

	2. User selects a time scale
<b>Test Data:</b>	N/A
<b>Expected Results:</b>	Historical chart will be available to view
<b>Actual Results:</b>	N/A
<b>Comments:</b>	This function was not implemented in time
<b>Test Status:</b>	FAIL

<b>Test Case ID:</b> TC13		<b>Related Requirements:</b> FR12
<b>Test Description:</b>	Verify that there is a news feature integrated	
<b>Prerequisites:</b>	User is logged into the system User has at least one currency in their favourite currency list	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the star icon on the navigation bar</li> <li>2. User clicks a currency</li> <li>3. User clicks news</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	News that is relevant to the currency will be displayed	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	
<b>Test Status:</b>	FAIL	

<b>Test Case ID:</b> TC14	<b>Related Requirements:</b> FR13
---------------------------	-----------------------------------

<b>Test Description:</b>	Verify that the user can receive the notification by email
<b>Prerequisites:</b>	User is logged into the system User has set a notification for their currency pair
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the notification button</li> <li>2. User checks the tick box 'I would also like to receive email notifications'</li> </ol>
<b>Test Data:</b>	N/A
<b>Expected Results:</b>	The email notification will come through when the desired rate is reached
<b>Actual Results:</b>	N/A
<b>Comments:</b>	This function was not implemented in time
<b>Test Status:</b>	FAIL

<b>Test Case ID:</b> TC15		<b>Related Requirements:</b> FR14
<b>Test Description:</b>	Verify that the system has multilingual features	
<b>Prerequisites:</b>	User is logged into the system	
<b>Test Procedure:</b>	<ol style="list-style-type: none"> <li>1. User clicks the settings icon on the navigation bar</li> <li>2. User clicks 'languages'</li> <li>3. User selects a language</li> </ol>	
<b>Test Data:</b>	N/A	
<b>Expected Results:</b>	The app will be displayed in the selected language	
<b>Actual Results:</b>	N/A	
<b>Comments:</b>	This function was not implemented in time	
<b>Test Status:</b>	FAIL	