Sheffield Hallam University

College of Business, Technology and Engineering

Department of Computing

Project (Technical Computing)

[55-604708]

2020/21

Author:	**Declan Stanley**
Student ID:	**27039772**
Year Submitted:	**2021**
Supervisor:	**Dr Jing Wang**
Second Marker:	**Dr Alessandro Di Nuovo**
Degree Course:	**Computer Science**
Title of Project:	**FOREX Predictor – Automating Forex Trading using Machine Learning**

Confidentiality Required?

NO

Acknowledgements

I would like to thank my family, housemates and close friends for encouraging me to complete this project to the best possible standard that I could. Secondly, I would like to thank my project supervisor Dr Jing Wang for his motivation and continuous support, going out of his way whenever he could to ensure the success of this project.

Abstract

People are always looking to obtain more money. Many turn to investing in stocks, however the investments are often only fruitful after a lengthy period of time. The Foreign Exchange (Forex) Market can be used to gain money over a shorter period of time, however learning how to trade Forex is normally a long process, and usually comes with many losses before wins.

This project aims to develop an application to predict Forex currency prices, that can be used by anybody regardless of their knowledge of Forex trading.

This report discusses the research, design, development and testing of the application that has been produced.

Table of Contents

1 INTRODUCTION 1

1.1 PROJECT BACKGROUND AND MOTIVATIONS 1

2 RESEARCH 2

2.1 EXPERT ANALYSIS METHODS 2

2.1.1 Momentum 2

2.1.2 Technical Analysis 2

2.3 SOFTWARE DEVELOPMENT LIFE CYCLE METHODOLOGIES 3

2.3.1 Waterfall 3

2.3.2 Agile 4

2.3.3 Kanban 5

2.4 DESIGN TOOLS 6

2.4.1 Low Level Fidelity 6

2.4.2 High Level Fidelity 7

2.5 DEVELOPMENT TECHNOLOGIES 8

2.5.1 Deployment Platform 8

2.5.2 UI Frameworks 9

2.5.4 Push Notifications 11

2.6 MACHINE LEARNING ALGORITHMS 12

2.6.1 Convolutional Neural Networks (CNNs) 12

2.6.2 Recurrent Neural Networks (RNNs) 14

2.6.3 Long Short-Term Memory Cells (LSTMs) 15

2.7 EXISTING OPEN-SOURCE SOLUTIONS 16

2.7.1 pankush9096/Stock-Prediction-using-LSTM 16

2.7.2 hayatoy/ml-forex-prediction 17

2.8 SERVER 17

2.8.1 Flask 17

2.8.2 Django 18

2.9 DATABASE 19

2.9.1 PostgreSQL 19

2.8.2 Firebase 20

2.9 PROJECT MANAGEMENT 21

2.9.1 Work Management 21

2.9.2 Version Control System (VCS) 22

3 DESIGN 23

3.1 PROCESS 23

3.2 PLATFORM 23

3.3 COLOUR SCHEME 23

3.4 LOGIN AND SIGNUP SCREEN 24

3.5 HOME SCREEN 25

3.6 PREDICTION SCREEN 26

3.7 PUSH NOTIFICATIONS 27

4 DEVELOPMENT 29

4.1 PROJECT SET UP 29

4.2 MACHINE LEARNING ALGORITHM 29

4.1.1 Data Collection 29

4.2.2 Data Preparation 29

4.2.3 LSTM 30

4.2.4 CNN & LSTM 31

4.2.5 Comparing the Results 32

4.3 SERVER AND DATABASE 33

4.3.1 Prediction View 35

4.3.2 Authentication ... 36
4.3.3 URLS ... 37
4.3.4 CORS ... 38
4.4 THE REACT APP ... 39
4.4.1 Base Application and Main Screen ... 39
4.4.2 Login Screen ... 39
4.4.3 Home Screen ... 45
4.4.4 Prediction Screen ... 46
4.4.1 Information Dialog ... 47

5 TESTING AND EVALUATION ... 48

5.1 API TESTING ... 48
5.2 SOFTWARE TESTING ... 50
5.3 USER EVALUATION ... 51

6 CRITICAL REFLECTION ... 53

6.1 DELIVERABLE SUCCESS ... 53
6.2 DELIVERABLE LIMITATIONS ... 54
6.3 FUTURE WORK ... 55
6.4 PROJECT MANAGEMENT ... 56
6.5 ETHICAL ISSUES AND CONSIDERATIONS ... 57
6.6 PERSONAL DEVELOPMENT ... 58

7 APPENDICES ... 59

7.1 APPENDIX A - PROJECT SPECIFICATION ... 59
7.2 APPENDIX B – PARTICIPANT CONSENT FORM ... 62
7.3 APPENDIX C – CHAD WRIGHT ... 64
7.3.1 Persona ... 64
7.3.2 Scenario ... 64
7.4 APPENDIX D – KEN AGBANOBI ... 66
7.4.1 Persona ... 66
7.4.2 Scenario ... 66
7.5 APPENDIX E – KANBAN BOARD AT END OF PROJECT ... 68
7.6 APPENDIX F – USABILITY SURVEY ... 69

BIBLIOGRAPHY ... 72

List of Figures

Figure 1 – The Waterfall Methodology (Jones & Waddell, 2019) 3
Figure 2 – Example of a Kanban Board (Ionos, 2019) 5
Figure 3 – Example of Paper Prototyping (Reidy, 2017) 6
Figure 4 – Examples of High Fidelity Wireframes (Babic & Tonheim, 2018) 7
Figure 5 – How Native, PWA and Hybrid Applications are Related (Infoways, 2020) 9
Figure 6 – How the Expo Push Notification Service Works (Expo, 2020) 12
Figure 7 Using a CNN for Time Series Data (Sayyad, 2020) 13
Figure 8 – Structure of an RNN (Banys & Kobran, Recurrent Neural Network (RNN), 2020) 14
Figure 9 – An Example of a Chain of LSTM Cells (Olah, 2015) 15
Figure 10 – Low and High Fidelity designs for the Login and Sign Up screen 24
Figure 11 – Low and High Fidelity Designs for the Home Screen 25
Figure 12- Low and High Fidelity Designs for the Prediction Screen 26
Figure 13 – Percentage split of if users find push notifications useful (VWO Engine) 27
Figure 14 – When users like to receive push notifications (VWO Engage) 28
Figure 15 – Summary of the multi-layer LSTM Model 30
Figure 16 – Predicted vs Real Prices for the multi-layer LSTM Model 31
Figure 17 – Summary for the CNN/LSTM Model 31
Figure 18 – Predicted vs Real Prices for the CNN/LSTM Model 32
Figure 19 – The final Login Screen 40
Figure 20 – The Redux Philosophy (Stevanoski, 2019) 41
Figure 21 – Redux Store wrapping the Current Screen 42
Figure 22 - Redux Store Changing 43
Figure 23 – PersistGate in between Store and the Current Screen 44
Figure 24 – The Final Home Screen 45
Figure 25 – The final Prediction Screen 46
Figure 26 – The Final Information Window 47
Figure 27 – The results of sending a request to 127.0.0.1:8000/api/auth/login 48
Figure 28 – The results of sending a request to 127.0.0.1:8000/api/auth/logout 49
Figure 29 – The results of sending a request to 127.0.0.1:8000/api/prediction/getData/ 50
Figure 30 – Chart showing acceptable a usability score is (Smyk, 2020) 52

Glossary

Base	-	The first currency in a pair
Quote	-	The second currency in a pair
UI	-	User Interface
API	-	Application Programming Interface
Dispatch	-	The process of starting an Action
Action	-	Describes something that happened in the app
Reducer	-	Makes changes to the state object in the store
Store	-	the object that brings the actions, reducers and the state objects together

1 Introduction

1.1 Project Background and Motivations

The Forex market is the largest financial market in the world, with an average of $6.59 trillion being traded every single day (BIS, 2019). For comparison, in 2020 the New York Stock Exchange (NYSE) had an average daily trading value of $200 billion (TradingHours.com, 2020). This means that the Forex market's daily trading value is almost 33 times larger than the NYSE.

However, a well-known figure in the world of Forex is that 90% of traders fail (Bennett, 2019). This could be down to greed, not knowing when to take a loss, or even lack of knowledge and experience. Many companies have developed their own algorithms for trading both stocks and Forex, to help eliminate human error. This is known as algorithmic trading. These companies keep their algorithms secret, as they do not want competitors to use them. A study in 2011 concluded that profits from companies using algorithmic trading was at the expense of other traders (Baron, Brogaard, & Kirilenko, 2014). This puts an unfair advantage on the regular person who wants to try and invest their money.

The aim of this project is to develop an application that can assist anyone to trade Forex, regardless of knowledge and technical ability. The full project aims and specification can be found in Appendix A.

2 Research

2.1 Expert Analysis Methods

2.1.1 Momentum

Experienced Forex traders use a variety of different indicators to help them determine whether they believe if a price is going to go up or down in value: the most affirmative indicator being momentum. In a study from 1993, it is suggested that individual stocks, and as an extension Forex prices, have momentum (Jegadeesh & Titman, 1993). In the study they found that stocks that gain in value over a few months have a higher chance of continuing in this trend and vice versa.

Machine learning algorithms can be used to detect momentum by using algorithms with a high tolerance to noise, which will help them to identify the trends.

2.1.2 Technical Analysis

Technical analysis is performed by traders when trying to analyse short-term changes (Chen, 2021). Technical analysis often takes months of learning and practice before getting it right. Even expert analysts get things wrong, as the patterns may not be obvious to the human eye (Pecherer, 2021). Machine learning may be able to spot these deeper patterns from when similar things have happened before.

2.3 Software Development Life Cycle Methodologies

2.3.1 Waterfall

The waterfall method is where there are 6 stages of development, and you do not start the next one until the previous one is complete. These six stages are: Analysis, Requirements, Design, Coding, Testing, and Deployment. Due to the linear progression through the stages, lots of prior research is conducted and every step is heavily documented. When working with a team this can be very useful, as if one member leaves, a replacement can be found easily as they can follow the development plan (Barker, 2002). A big disadvantage of using the waterfall methodology is that changes are difficult to make later on in the process and looping back to the requirements stage is almost certain to cause a delay in deployment (Royce, 1970).

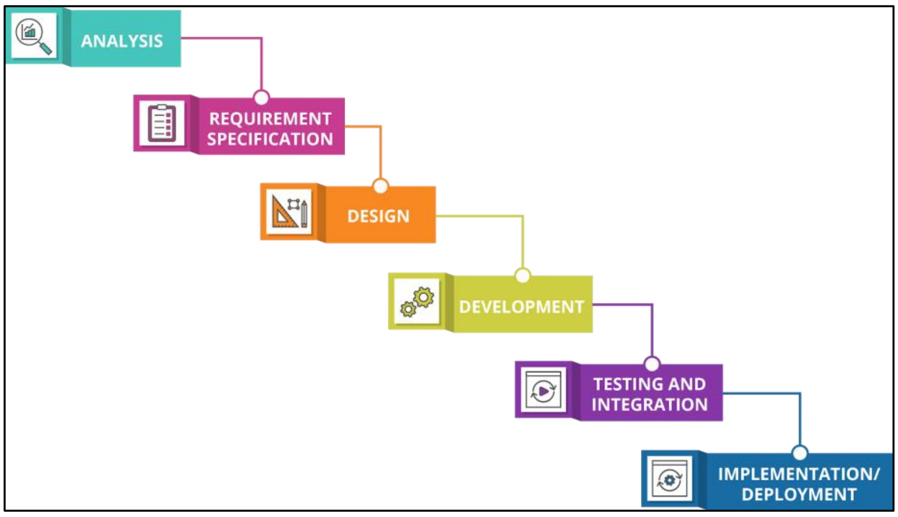


Figure 1 – The Waterfall Methodology (Jones & Waddell, 2019)

2.3.2 Agile

Agile development is the name given to a family of methodologies that focus on creating software often in short cycles called sprints, where each cycle is an improvement on the last. One of the four major principles set out in the Agile Manifesto in 2001 is *"working software over comprehensive documentation"* (Sacolick, 2020). This, in contrast to the waterfall cycle, highlights the emphasis on creating the software over planning every detail beforehand.

In 2020, 95% of people surveyed responded that the company that they work for practice Agile Development (digital.ai, 2020). The top reason for this was the ability to manage changing priorities. This shows that requirements and designs can be changed and implemented easily, often at the start of a sprint. Due to the iterative nature of Agile development, usable software is produced early on in development, so that it can be tested and refined.

The main problem with Agile development is that it can be difficult to plan how long something will take to complete (Cagle, 2019). This means that any underestimations for tasks in the current sprint can cause a growing backlog. This can result in time restrictions towards the end of the project meaning that some features may need to be missed out and work may not be completed to a high standard.

2.3.3 Kanban

Kanban is a type of Agile development designed to manage workflow with an emphasis on continuous improvement (Siderova, 2018). It involves the use of a board to visualise the current progress of tasks. The boards normally have four sections: Backlog, To Do, Doing, and Done.

At the beginning of the project, all tasks are put into the backlog. Then the idea is for all the tasks to eventually progress through each section until they are all in the 'Done' section. At any point in the project new tasks can be added to the backlog as this allows for continuous development and improvement.

Kanban does not have set sprints, as workflow is restricted: a new task is undertaken only when a developer is ready (Mesh, 2020). This could be seen as a disadvantage, as no set deadlines are made. However, it means that no time is spent planning and documenting features that may not end up existing. The flexibility of Kanban will be perfect for this project because I currently do not know how long it will take to complete many of the tasks required.

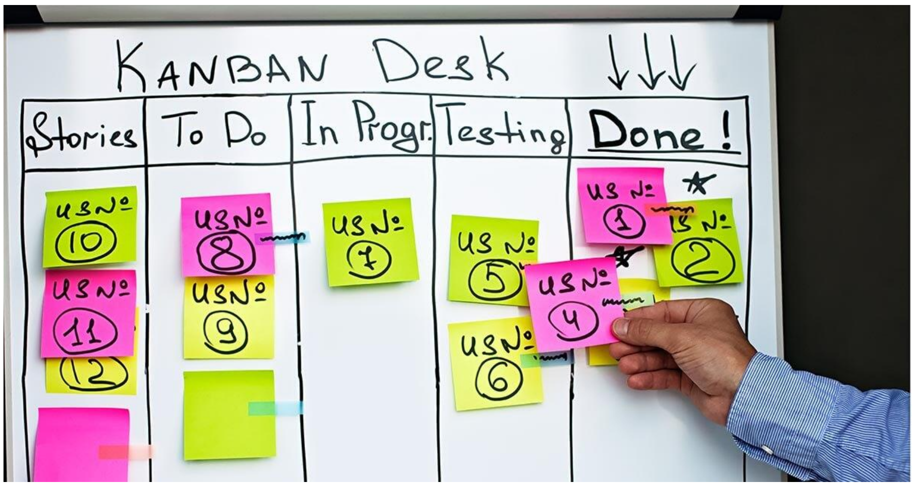


Figure 2 – Example of a Kanban Board (Ionos, 2019)

2.4 Design Tools

2.4.1 Low Level Fidelity

Wireframes are basic diagrams that reveal the functionality of each UI screen (Wood, 2014). They should not be fully complete designs, only consisting of lines, boxes and basic text. This is what is known as a low fidelity design (Babich, 2017). These designs can be sketched out on paper, as they do not require lots of detail. Furthermore, designs can be changed easily by creating different parts of the screen on different pieces of paper. These pieces of paper can be moved about and changed to find the best design. Once a final design is reached, the different sections can be stuck together to create a final low fidelity design. This is known as paper prototyping (Snyder, 2003).

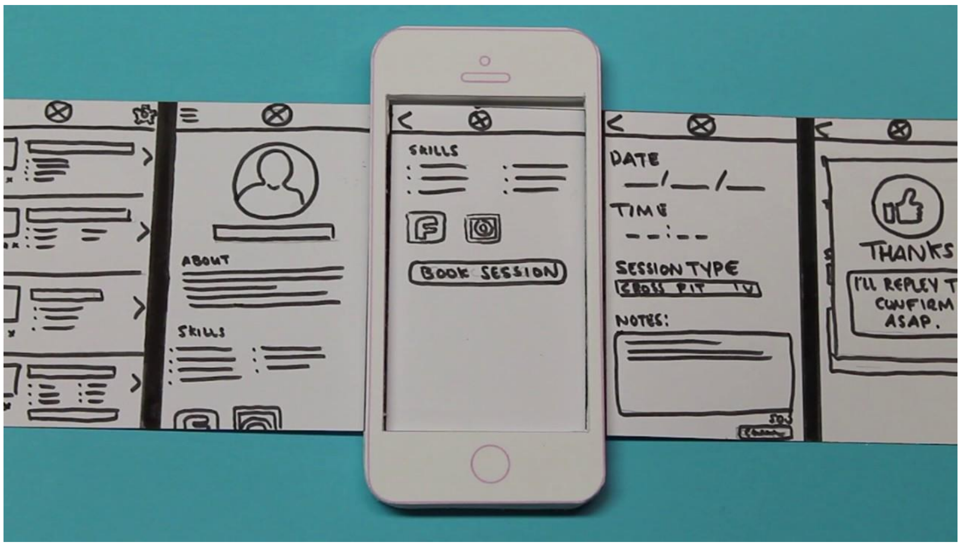


Figure 3 – Example of Paper Prototyping (Reidy, 2017)

2.4.2 High Level Fidelity

Once the low fidelity designs are complete, high fidelity designs will need to be created. These designs reflect how the UI will actually look. Graphics, spacing and UI layout are very important for high fidelity designs (Osman, 2020). Lucidchart have a wireframe generation tool with a vast amount of premade UI shapes (Lucidchart, 2021). It also has the ability to make the diagrams interactive by creating clickable hotspots that takes the user to the appropriate next wireframe. However, Lucidchart requires a subscription fee of £8 a month to use this service. Cacoo (Cacoo, 2021) is a free to use diagram creation website that also includes a vast amount of premade UI shapes. It doesn't allow interactivity in the diagrams, however since my app is only going to be 4 pages, interactivity is not necessary.

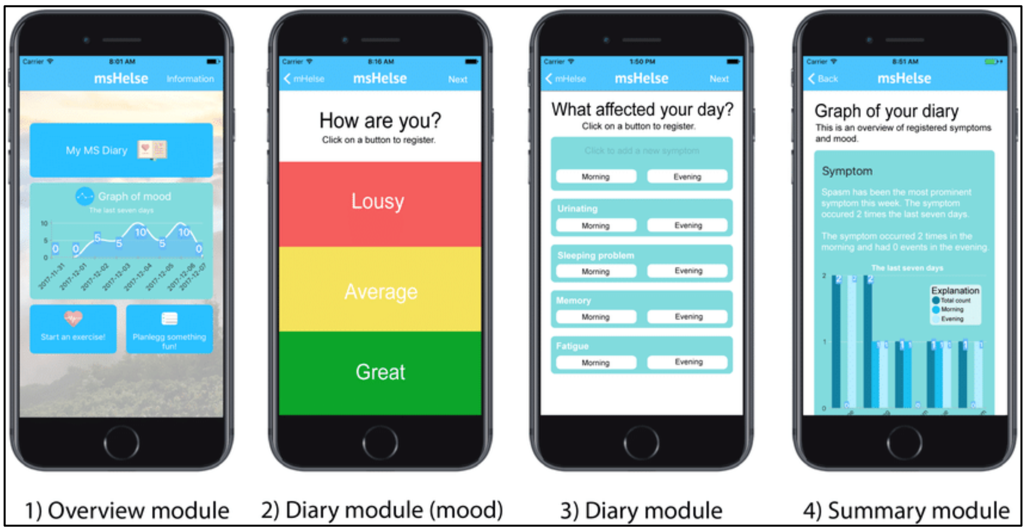


Figure 4 – Examples of High Fidelity Wireframes (Babic & Tonheim, 2018)

2.5 Development Technologies

2.5.1 Deployment Platform

Progressive Web Applications (PWAs) are websites that have lots of the benefits of a mobile application (Scandiweb, 2021). They are used via a browser and often liked as nothing needs to be downloaded for it to be used. PWAs do come with some drawbacks though. They can often seem slower and less seamless than their app store equivalent (Sheppard, 2017). Furthermore, some features are unavailable to use when creating a PWA, for example push notifications.

Native development is the creation of software that run on specific devices and platforms (Monus, 2019). They are written in languages specifically designed for that platform, for example Java for Android and Swift for iOS. They often run very quickly because the code is designed for that specific platform, however if trying to make the application available to users on different platforms multiple programs need to be written.

Hybrid applications are a combination of PWAs and native mobile apps. They can be accessed via browser or by downloading an app (Existek, 2019). They are often written in HTML5 and JavaScript and wrapped inside a native container to deploy as a mobile application. Despite making development and deployment much quicker, lots of hybrid apps still feel slow due to most the data being loaded as the user navigates through the app (YML, 2020). However, recently new frameworks have been developed that will create a web application and also true native applications. I will need to use one of these frameworks if I am to create a PWA and also a native application.

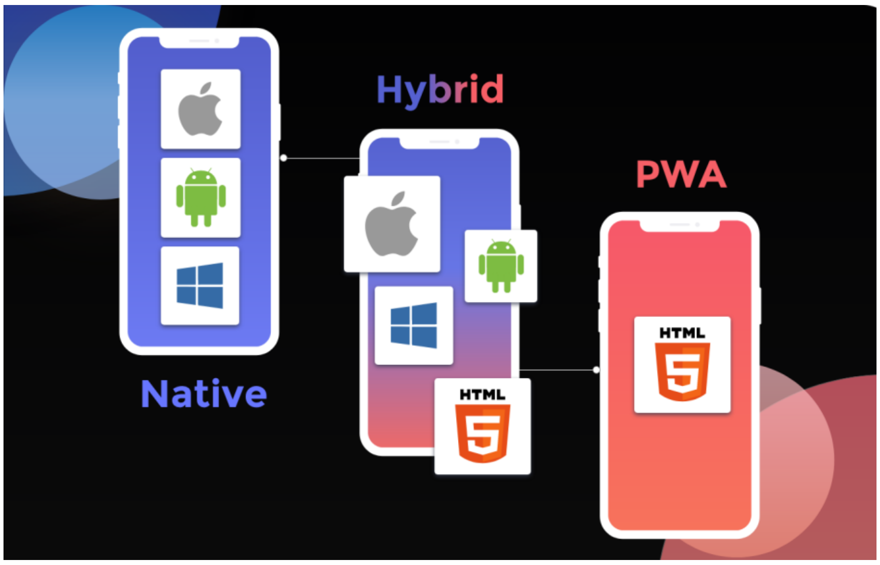
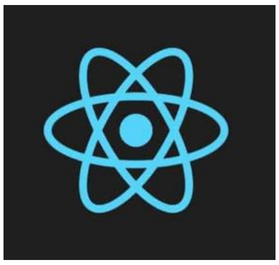


Figure 5 – How Native, PWA and Hybrid Applications are Related (Infoways, 2020)

2.5.2 UI Frameworks

There are two different frameworks available to create hybrid apps that generate true native applications.



- **React Native** (Facebook, 2021) is a framework built upon ReactJS. It is written using JavaScript and then compiled into native views (Lithios, 2021). This allows React Native applications to be used as native applications but also via browser. It often requires using lots of additional 3rd party libraries made by the community, and therefore some of these may have limited documentation. However due to already knowing how to program in JavaScript, React Native would be easy to learn. Furthermore, a survey in 2020 found that 42% of all cross-platform developers use React-Native (Liu, 2020), this being the highest percentage of all cross-platform languages. This highlights the size of the React Native community, and therefore the amount of reference material that is available for the standard packages.

- **Flutter** (Google, Flutter, 2021) was found to be the second highest used language in the survey at 39%. Whilst React Native requires the code to go through a JavaScript bridge before compiling, Flutter is compiled Ahead-of-Time (AOT) into native code when in release mode (Rozwadowski, 2020). This results in faster start-up times and overall performance. However, unlike React Native, Flutter does not use JavaScript as the base language. Instead, Flutter applications are built using Google's programming language Dart (Google, Dart, 2021). This would mean a steeper learning curve than React Native. Due to both Flutter and Dart being relatively immature (Rozwadowski, 2020), there is also less customisability than React Native, as many libraries are still in the pre-alpha stage.

In conclusion, I have decided that I will use React Native to create my application. I came to this decision because React Native is a more mature language than Flutter. Combining this with the fact that React Native uses JavaScript, results in there being more reference material and libraries available to use to aid me in the development of FOREX Predictor.

2.5.4 Push Notifications

A push notification is a message that pops up on a mobile device to convey a message (Airship, 2021). They are used by app developers to increase engagement by users. There are several companies that provide services for sending push notifications, and they all work in different ways.

Apple Push Notification service (APN) can be used to send notifications to iOS devices. APN uses high level encryption when establishing an initial connection so that nobody can intercept the device token and impersonate the app. However, FOREX Predictor is being developed for both iOS and Android devices, so a second service would also need to be used.

Firebase Cloud Messaging (FCM) can be used to send notifications to Android devices. It handles lots of the logic for the developer, so that they only need to create code for creating the notifications to send and handling the notifications on the app (Google, FCM, 2020). The same problem arises as APN, a second service would be required.

Expo is a platform used for creating hybrid applications, including using React Native (Expo). They provide a Push Notification service that can be implemented easily into a React-Native project. As seen in Figure 6, the Expo backend identifies what operating system the device is using, and then communicates with the appropriate provider to send a notification to the user. This means that only one service will need to be set up, as Expo will configure FCM and APN. It is for this reason that Expo Notification Service will be used for FOREX Predictor.

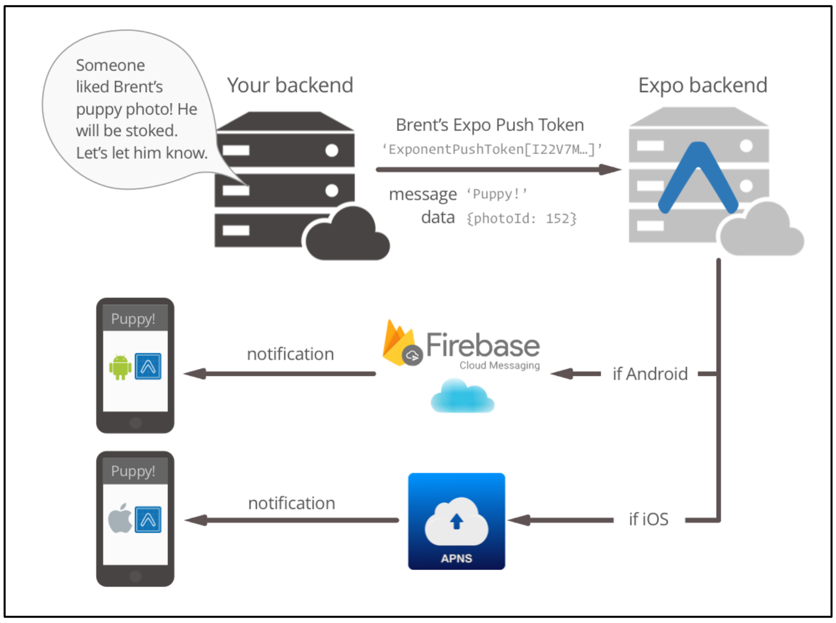


Figure 6 – How the Expo Push Notification Service Works (Expo, 2020)

2.6 Machine Learning Algorithms

As discussed in section 2.1.1, momentum is a huge factor involved. For a machine to calculate whether the momentum is causing a price to go up or down, it needs to look at several previous prices, not just the current one. This can be achieved by training a model using a time series. A time series is a sequence of data points in chronological order (Scott, 2021). There are several different machine learning algorithms that can be used to process time series data.

2.6.1 Convolutional Neural Networks (CNNs)

CNN's were originally designed for two-dimensional image data; however, they can also be used for time series forecasting (Brownlee, How to Develop Convolutional Neural Network Models for Time Series Forecasting, 2018).

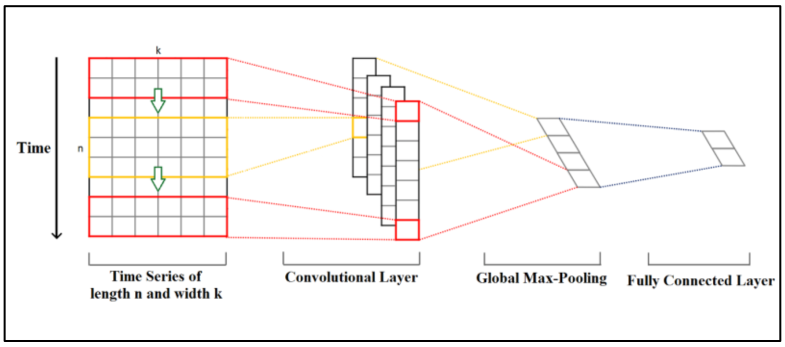


Figure 7 - Using a CNN for Time Series Data (Sayyad, 2020)

As seen in Figure 7, a basic CNN consists of 3 types of layers. The convolutional layer acts as the input layer. It has a kernel parameter, which is how many points to analyse at one time. The kernel should always be the same size as the width of the time series data, so that it always moves in one direction, performing the convolution (Sayyad, 2020). After the calculations the max pooling layer takes the largest value from each window and adds these to a new vector. This new vector is then used as the input for a regular connected layer.

Research shows that CNNs are useful for time series data because they have a high tolerance for noise and are good at extracting deep features (Sayyad, 2020). This would be good for FOREX Predictor because currency pair prices fluctuate lots over a period of time, and general trends need to be found. However, they do however have a weakness. They have no memory of what has happened in the past, and therefore when trying to look at patterns over long periods of time they start to become less accurate.

2.6.2 Recurrent Neural Networks (RNNs)

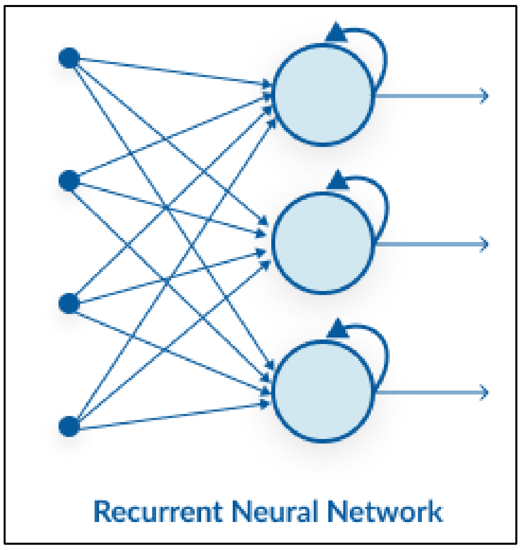


Figure 8 – Structure of an RNN (Banys & Kobran, Recurrent Neural Network (RNN), 2020)

RNNs utilise a feedback loop, as seen in figure 8. This acts as a type of short-term memory. This short-term memory is great for time series, as the algorithm does not only consider the current window, but also what has happened in previous windows. This means that two identical inputs may give different outputs, depending on what has happened previously (Banys & Kobran, Recurrent Neural Network (RNN), 2020). This would be useful for FOREX Predictor, as it would be able to analyse what has happened with previous trends and apply this to the current trend.

The main weakness of using RNNs is their inability to store memory over long periods of time (Shih, Sun, & Lee, 2019). This is due to what is known as the vanishing gradient problem. This is where the input from the feedback loop exponentially decreases for a specific data point the further away it is (Brownlee, How to Fix the Vanishing Gradients Problem Using the ReLU, 2019). For example, if data point A took place a year later than data point B, then the RNN will barely take into account what happened at point B.

2.6.3 Long Short-Term Memory Cells (LSTMs)

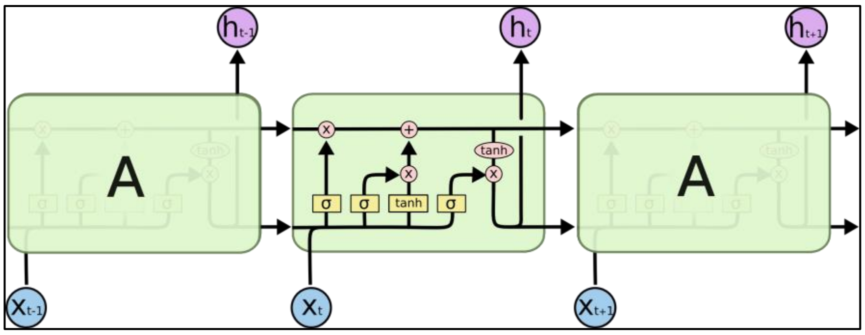


Figure 9 – An Example of a Chain of LSTM Cells (Olah, 2015)

LSTMs were developed to overcome one of the main limitations of regular RNNs. As seen in Figure 9, they utilise four activation functions to save relevant information to use for later stages of training (Banys & Kobran, Long Short-Term Memory (LSTM), 2020). These activation functions are used to create three gates:

- **Forget Gate:** Decides what to remove from memory
- **Input Gate:** Controls what data is added to memory from the input
- **Output Gate:** Decides what to output from the memory

Deep LSTMs can be created by using several layers of LSTM cells. Providing sufficient data is provided for training, deep LSTMs perform significantly better than single layer versions (laddad, Basic understanding of LSTM , 2019).

Overall LSTMs seem to be the best option to use for FOREX Predictor, as over time it should still keep its accuracy due to its long-term memory. A model using both a CNN and an LSTM will also be tried, in order to see whether the CNN can be used for trend analysis and the LSTM for remembering these trends for long time periods.

2.7 Existing Open-Source Solutions

No public software solutions exist, most likely due to being company secrets, so the focus of this section will be on trained models found on GitHub.

2.7.1 pankush9096/Stock-Prediction-using-LSTM

(Kukreja, 2019)

This solution utilises four layers of LSTMs. Dropout layers are also used to help prevent overfitting. Dropout layers temporarily disable a certain proportion of all of the cells in one layer. This means that for that pass any weight updates are not applied to that cell, and they do not contribute to any weight updates downstream (Brownlee, Dropout Regularization in Deep Learning Models With Keras, 2016). This helps neighbouring cells to not become too reliant on one another and cause overfitting.

The solution uses windows of 60 days as inputs for the algorithm. The result is a set of predictions that follow the trend fairly accurately, however the predicted prices tend to vary by a fair amount to the actual prices.

2.7.2 hayatoy/ml-forex-prediction

(hayatoy, 2017)

This solution treats the problem as a binary classification problem. For each datapoint, a label is given to indicate whether the price increased or decreased the next day. These labels are then used as part of the time series for training. A Gradient Boosting Classifier was used, and this model resulted in an accuracy of 58%. This resulted in a profit when back tested, however it does not forecast the prices, only the trend.

2.8 Server

Once trained, the machine learning models will need to be hosted on a python server so that they can be used by the React application.

2.8.1 Flask

Flask (Flask, 2010) is a lightweight framework used to create web-based applications and servers in Python. Flask requires little code to get a simple app running and can be seen as more 'Pythonic' than other web frameworks (Full Stack Python, 2021). This means that providing prior knowledge of Python, it is relatively easy to learn when compared to other frameworks. However, the flask community is smaller than that of the giants of the industry, and therefore there is less reference material and support available for it.

2.8.2 Django

Django (Django, 2021) is a high-level web framework for Python. They follow Python’s “batteries included” philosophy (Sidorenko, 2017). This means that they aim to deliver as many features as possible, minimising the amount of work that the user needs to do. This encourages rapid development and clean design (Django, 2021).

Django also handles lots of aspects of security for the user, helping to protect against a variety of different attacks including Cross Site Scripting (XSS) attacks and Cross Site Request Forgery (CSRF) attacks. This is great for complying with GDPR as it helps to ensure that user data is secure.

It does however come with some disadvantages. Django projects are rather large in size and can feel overkill for some projects (Nader, 2020). Furthermore, it can be described as monolithic, meaning that there is a popular way of doing things and much of the community believe that is the way it should be done. If a problem does occur it could become problematic finding other solutions than the main one.

Due to its rapid development process and security features, Django will be used to host the machine learning models.

2.9 Database

FOREX Predictor will require a database to store user data for logging in. The database needs to be secure to be GDPR compliant. There are several different open-source database management systems (DBMSs) that could be used.

2.9.1 PostgreSQL

PostgreSQL, also known as Postgres, is an open-source DBMS that provides enterprise-class performance (Bitnine, 2016). It is ACID secure, meaning that data validity is guaranteed despite power cuts or other errors (Saračević & Mašović, 2013). Postgres can be used in conjunction with many programming languages, including Python. This would be useful for FOREX Predictor, as this means that all of the authentication processes can be handled by the server. Furthermore, Postgres emphasises on extensibility.

If FOREX Predictor was to be deployed, Postgres would be able to handle large amounts of users with ease (Babeni, 2020). Installation can be quite difficult for a beginner (Pedamkar), so this would need to be taken into consideration.

2.8.2 Firebase

Firebase (Google, Firebase, 2012) is known as a Backend-as-a-Service (BaaS). It provides many of the services that developers normally need to build themselves, saving time to allow them to focus on the app functionality (Stevenson, 2018). Some of these features include databases and authentication.

A Firebase database is hosted in the cloud by Google, meaning that they are extremely secure and would be GDPR compliant. However, since Firebase uses non-relational databases, deep querying of the data stored can prove extremely difficult (Croos, 2018). For FOREX Predictor’s initial prototype this wouldn’t be an issue as no querying is necessary. However, if the application was to be deployed and expanded upon, it is most likely that more data would be required to be stored. This would result in large custom query functions needing to be written that would perform slower than SQL (Dearmer, 2020).

PostgreSQL’s advantage of scalability makes it the best choice for FOREX Predictor, as this means that the project can be expanded upon easily with little changes necessary. It would also not hinder on performance when the database grows and becomes more complex.

2.9 Project Management

2.9.1 Work Management

Trello (Atlassian, 2011) is a project-planning tool designed to help manage time and track progress. It can be used as a Kanban board as lists and cards can be made, and then cards can be dragged and dropped into different sections. Each card can store a variety of information, including pictures, documents and text. They can also be assigned tags to group cards together.

Kanboard is an alternative to Trello, boasting many of the same features. It has a minimalistic design to help focus on key information. Kanboard needs to be downloaded to the user's system, but this has the benefit of being able to access it offline (Kanboard).

Due to Covid-19, offline capabilities are not necessary as all development will be completed from home. Therefore, Trello will be used to manage the project.

2.9.2 Version Control System (VCS)

In order to manage code and track changes within it, version control will be used. This has many benefits, one of which being troubleshooting. If a bug is found, the code can be compared to a previous known working version (Azarian, 2013). There is a plethora of different version control software available for use. Two of the main ones are Git and SVN.

SVN uses a central server to store all files historical versions (Backlog, 2020). Moreover, it allows the user to checkout subdirectories to work on, instead of the entire repository. However, the central server becomes problematic if an error occurs, as it could result in all builds being destroyed (Azarian, 2013).

Git is a distributed VCS system; meaning that the complete codebase and its full history are stored on each developer's computer (Gehman, 2018). There are several services that allow Git repositories to be hosted online: the most popular being GitHub (Github Inc, 2008). One of the main benefits of using GitHub as a single developer is that if for any reason the project becomes corrupted locally, the developer can easily recover a backup from GitHub.

Since having previously used GitHub and having experience in it, GitHub will be the VCS used for FOREX Predictor.

3 Design

3.1 Process

I initially created personas and scenarios for the application. These describe potential users of the application and how they would interact with it. The wireframes that I created took into account these personas. Example designs were also used to make the screens look familiar to users.

3.2 Platform

Due to some users wanting to use a smartphone, whilst others wanting to use a website, the app needed to be designed to work on several platforms. Based on the research conducted in Section 2.5.1, FOREX Predictor will be deployed as a cross-platform application, using React Native.

3.3 Colour Scheme

Colour schemes are important for designing applications because different colours convey different messages (Morioka & Stone, 2008). Red is used to represent power whilst blue is used to represent knowledge. I want the application to convey these messages to the user, and hence have decided to use these colours for my colour scheme. Furthermore, I want the background to be dark so that it is easy to look at even when tired, for people like Chad (Appendix C).

3.4 Login and Signup Screen

This is the screen that will be shown when a user first downloads and open FOREX Predictor. The design is inspired by a concept app designed by UI designer Liliya Kizlaitis, called 'wwater' (Hannah, 2020). All of the text is large so that even people like Ken (Appendix D) can easily read it and use the app. Contrasting colours are also used to make sure everything is easily visible.

A large 'i' icon can be seen in the top left of the screen, which can be clicked to give information about the app and how it works. This allows the user to gain an insight about the app and its use, before creating an account. This will increase the amount of people that use the app, as many people are reluctant signing up for things that they do not know about (anthony, 2012).

The final designs can be seen in Figure 10.

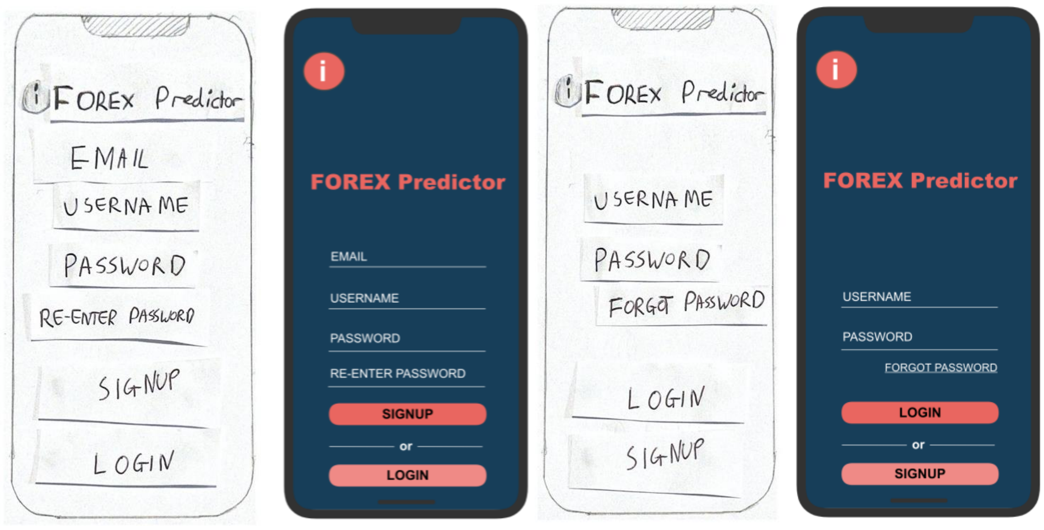


Figure 10 – Low and High Fidelity designs for the Login and Sign Up screen

3.5 Home Screen

The home screen is designed to be as simple as possible, so that even people who struggle with technology, like Ken (Appendix D), can use it easily. The information button remains in the top left corner so even after login users can still get information about the application.

Three currency pairs are clearly displayed in the centre of the screen as buttons. These buttons can be clicked to take the user to the prediction screen. They are in a different shade to the logout button, to emphasise that they have a different function.

The final designs can be seen in Figure 11.

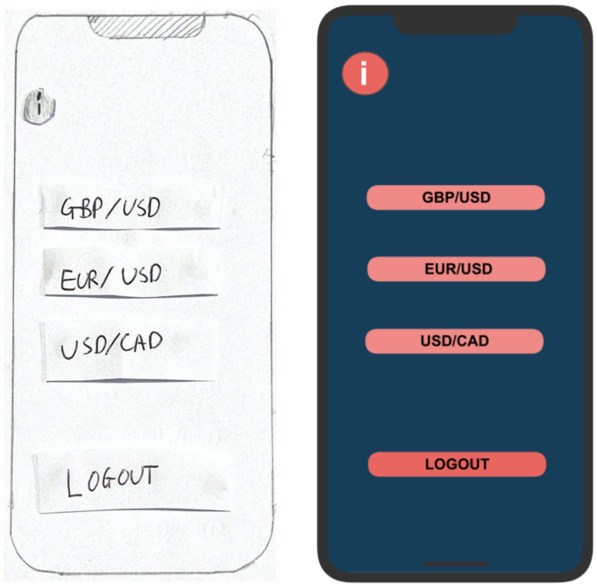


Figure 11 – Low and High Fidelity Designs for the Home Screen

3.6 Prediction Screen

The prediction screen features a large title at the top indicating to the user which currency pair they have selected to view. Below that is an interactive graph for the historical prices of that currency pair. When hovering over a point, the price for the day corresponding to that point is displayed. This price is large in size so that it is easily read.

Below the graph is where the user can see the predicted price for tomorrow. It is clearly labelled so that it is obvious what the number means. It is then followed by the difference compared to today’s price. The difference changes colour depending on whether it is a positive of negative difference. Green is used for a positive difference, and red for negative. This highlights to the user if the change is positive or negative, as these are the colours that people associate with them (Mammarella, Domenico, Palumbo, & Fairfield, 2016).

The final designs can be seen in Figure 12.

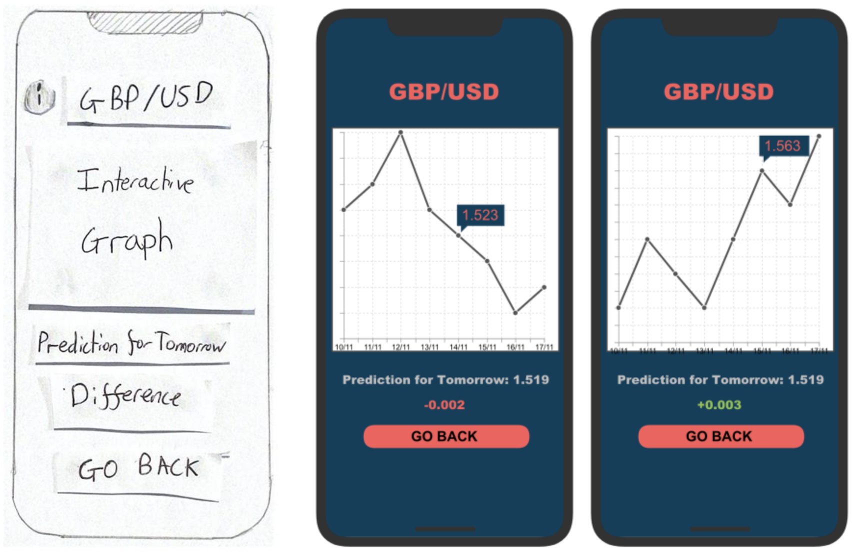


Figure 12 - Low and High Fidelity Designs for the Prediction Screen

3.7 Push Notifications

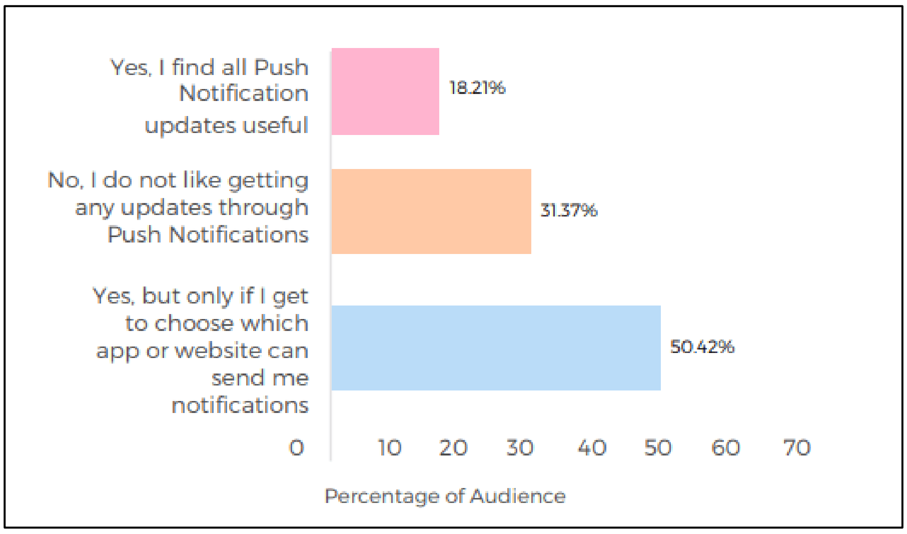


Figure 13 – Percentage split of if users find push notifications useful (VWO Engine)

As seen in Figure 13, over 50% of people only find push notifications useful when they choose which app sends the notifications. This means that making the notifications engaging is very important in order to get user's opting in for them. There are several ways of making push notifications engaging. Using emojis increases reaction rates by 20% (Business of Apps, 2021). FOREX Predictor's notifications will include the use of emojis to gain more user interaction.

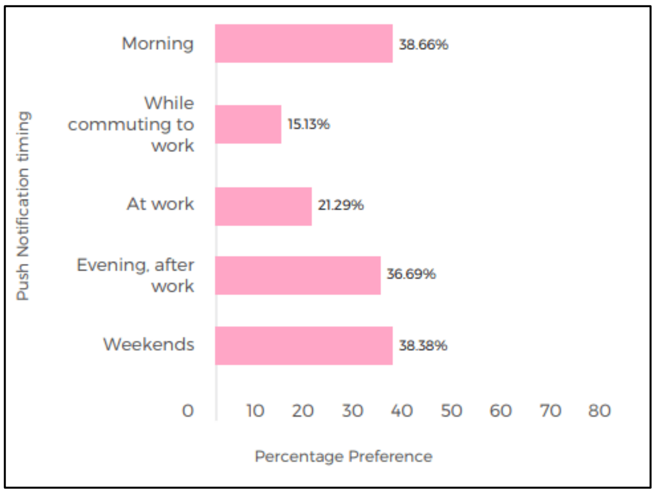


Figure 14 – When users like to receive push notifications (VWO Engage)

As seen in figure 14, users like to receive notifications least whilst commuting to work and whilst at work. To create maximum engagement, FOREX Predictor’s push notifications will be sent outside of standard working hours, including commute time: Monday-Friday 8:00-18:00.

4 Development

4.1 Project Set Up

A Trello board was created to use as a Kanban board, which was used to keep track of all tasks required for the project. A GitHub repository was also set up to version control the project. This meant that code could be backed up regularly, and any mistakes could easily be undone.

4.2 Machine Learning Algorithm

4.1.1 Data Collection

Data for both training and use in the app is retrieved from Yahoo Finance. This data is open-source and can be obtained easily using the python package *'yfinance'*. This package can retrieve historical data from the site for any Forex currency pair between any two dates. The data is returned as a Pandas DataFrame ready for preparation.

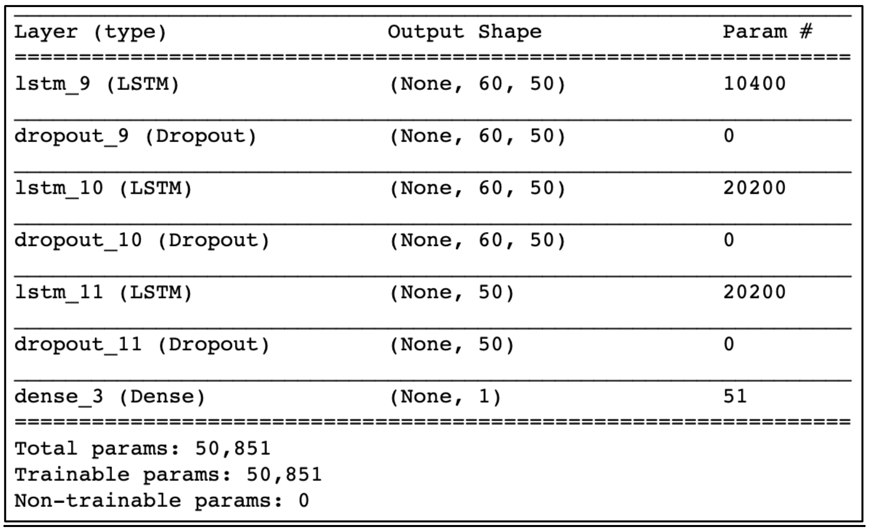
4.2.2 Data Preparation

The data preparation for both models is the same. The data retrieved is first split into a training set and a test set. It is then scaled using Sci-kit Learn's *MinMaxScaler* to normalise it. The final step was to create the windows to be used as inputs. A window size of 60 was chosen to get a good balance between finding trends over a longer period of time and keeping the predictions close to the actual values.

4.2.3 LSTM

For each model a 0.2 dropout was used after each layer, to help prevent overfitting (tf.keras.layers.Dropout, 2021). This means that on each pass 20% of cells on each layer have no effect on the weights they are connected to.

The first model tried utilised LSTMs. After experimenting with different hyperparameters, it was decided that this model was going to use 3 dense layers containing 50 LSTM units in each layer. The summary of the model can be seen in Figure 15.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 60, 50)            10400
_________________________________________________________________
dropout_9 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_10 (LSTM)               (None, 60, 50)            20200
_________________________________________________________________
dropout_10 (Dropout)         (None, 60, 50)            0
_________________________________________________________________
lstm_11 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_11 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

Figure 15 – Summary of the multi-layer LSTM Model

Once trained, the model was used for the test data set that was created earlier. The results of the predicted prices vs the actual prices can be seen in Figure 16.

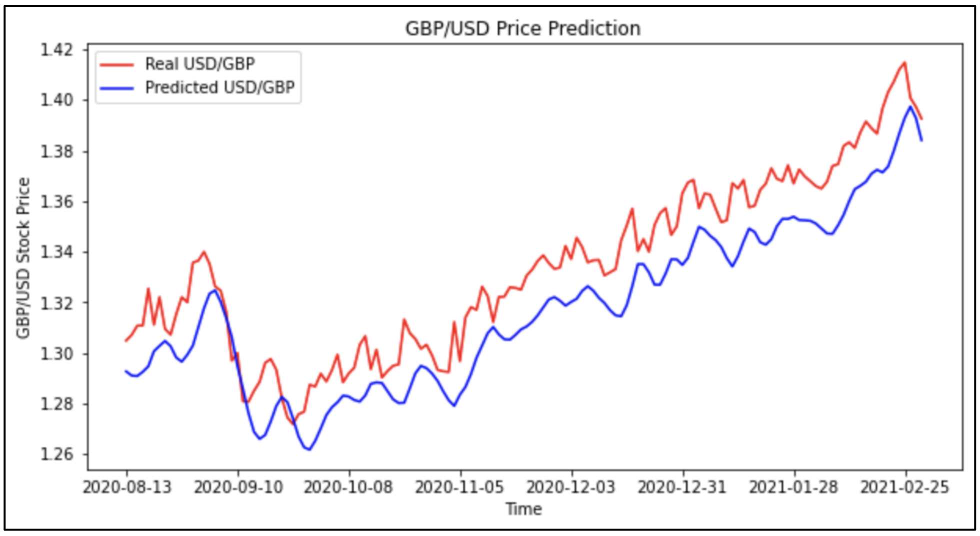


Figure 16 – Predicted vs Real Prices for the multi-layer LSTM Model

4.2.4 CNN & LSTM

The other model trained incorporated a CNN layer to identify trends and an LSTM layer to act as a memory and remember the trends. As discussed in section 2.6.1, the CNN's kernel size is the same as the size of a time window. After experimenting with different numbers of units for the LSTM layer, it was decided that 50 was the optimal amount. The summary of the model can be found in Figure 17.

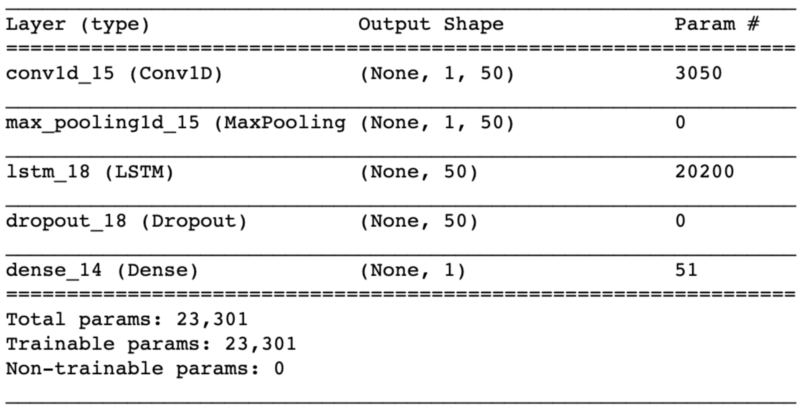

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_15 (Conv1D)           (None, 1, 50)             3050
_________________________________________________________________
max_pooling1d_15 (MaxPooling (None, 1, 50)             0
_________________________________________________________________
lstm_18 (LSTM)               (None, 50)                20200
_________________________________________________________________
dropout_18 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 51
=================================================================
Total params: 23,301
Trainable params: 23,301
Non-trainable params: 0
_________________________________________________________________
```

Figure 17 – Summary for the CNN/LSTM Model

The results comparing the predicted prices using the model vs the actual prices can be seen in Figure 18.

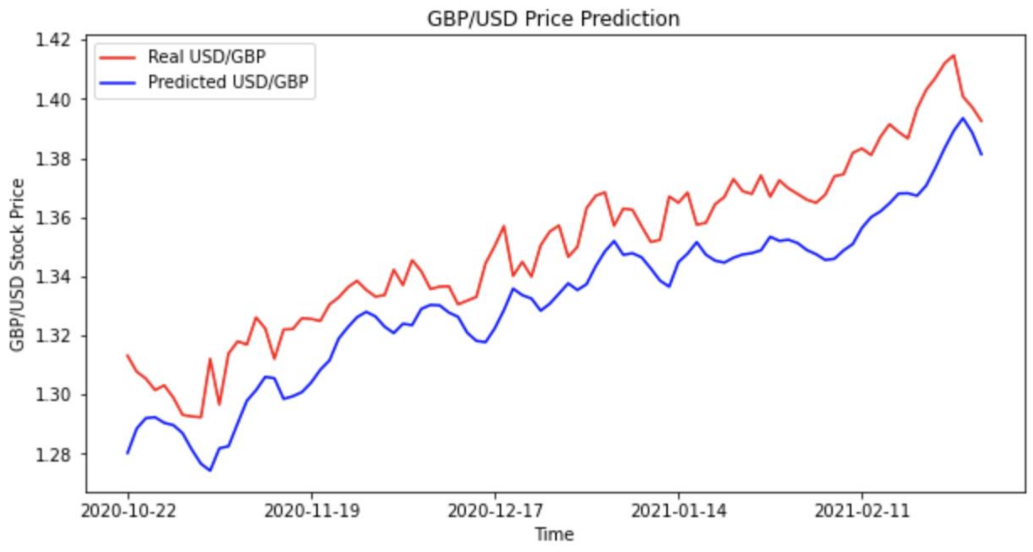


Figure 18 – Predicted vs Real Prices for the CNN/LSTM Model

4.2.5 Comparing the Results

Both models' predictions sit below the real price, however they do follow the trends well. The model composed of multiple LSTM layers seems to follow the real price more precisely than the CNN/LSTM model. It is because of this reason that FOREX Predictor will utilise the first model. The EUR/USD and CAD/USD datasets will be used to train this model also, resulting in a total of three models.

4.3 Server and Database

The server and database were created following a tutorial found on *Towards Data Science* (Gaurav, 2020). The server uses a local python environment to ensure that all packages that are required are installed, and versions can be set to solve any compatibility issues. A requirements file was made so that these packages can be tracked and updated easily.

The server's skeleton code is created by running a series of Django commands in the terminal. The first command is:

```
django-admin startproject mainapp
```

This command creates a new Django project called *'mainapp'*. It also creates a new app within this project also called *'mainapp'* which was then renamed to *'django_app'*

mainapp serves as the Django application that is used (Gaurav, 2020), however any functionality that needed to be added was made in other custom apps that the main app uses. Therefore, the next step was to create an app within the *Django-app* project called 'prediction'. This was made using the terminal command:

```
django-admin startapp prediction
```

As the name suggests this is where the machine learning algorithm will be housed and utilised. This app also needed to be added to the 'INSTALLED_APPS' section of the main app so that it can be used.

Next, a database called '*predictiondb*' was created for the app to use, using the default application for managing PostgreSQL databases: pgAdmin 4. The database connection information needed to be added to a local settings file for the server.

The next step was to migrate the app schema. This simply means synchronising the database and the server so that all pre-existing models are accounted for in both the database and the server. This is extremely important as the next step involved modifying the database and would have huge problems if the server models are not in sync with the tables in the database (Gaurav, 2020). This migration is completed by running the command:

```
python manage.py migrate
```

Finally, an admin account was created for the server. This admin account has access to a dashboard that allows them to add and remove users, as well as delete authentication tokens which became extremely useful when debugging the application. This was achieved by running the command:

```
python manage.py createsuperuser
```

This then prompts for a username and password. Once complete, the admin dashboard can be accessed by visiting h*ttp://127.0.0.1:8000/admin/* whilst the server is running.

4.3.1 Prediction View

First the model needed to be put onto the server, as well as the scaler that the training data used. The scaler needs to be stored as it is fitted to the training data, so therefore when new predictions are to be made, they need to be scaled the same as the training data was. Joblib is a package that is used to run python functions as pipeline jobs (Joblib, 2021). It is optimised for *NumPy* arrays and as a result, creates smaller files that run more efficiently than other packages, for example *Pickle*.

The models were loaded in the *app.py* file for the prediction app. This reduces overhead by only loading the model once when a connection has been established (Gaurav, 2020). This would be extremely useful for if the project was to be extended so that the data that is displayed is updated every second, as opposed to daily as the current dataset allows. If this was the case it would mean that the model would still only be loaded once and could still be used multiple times in one session, making the application seem more seamless.

The next step was to write the API that the application uses to get the historical financial data to display and to use the required prediction algorithm to predict the next day’s value. This was achieved by using the Django Rest Framework. By inheriting from the *APIView* class, APIs can be easily constructed (Django Rest Framework Community, 2021).

For the predict function it retrieves the past 3 months data for the selected currency pair and scales each point using the saved scaler. Then it creates windows of 60 days to be passed into the algorithm to calculate the prediction for the next day. Next it puts this prediction and the historical data into a dictionary which is returned through a Response object. This means that it can then be used by the React application.

4.3.2 Authentication

Without adding authentication, anybody could send requests to the API. This could be used maliciously to easily take down the server through a DDoS attack, as an attacker would not need separate authenticated logins for it to process a request (Gilling, 2020).

For authentication, *Django Rest Auth* was used. The library was added to the '*INSTALLED_APPS'* section of the main app, and then migrated again. This added a table to store all of the authentication tokens that were in use. These tokens are also viewable from the admin dashboard. This was extremely useful when writing the login/logout functions as it could easily be seen if a token had been created or removed. It also meant that if a token needed to be removed for testing purposes it could be done easily.

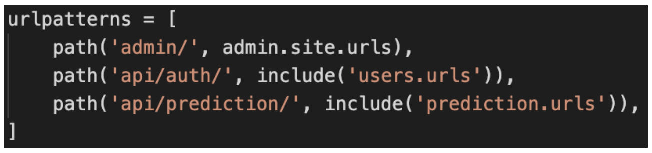
The next step was to create a 'users' app using Django. The terminal command for this was:

```
django-admin startapp users
```

Once this app was created, the relevant Django libraries were imported and the API classes for login and logout were written. Login uses the Rest libraries *'LoginView'* as it is. Logout inherits from *'LogoutView'* and adds a simple check to ensure that the only account that can call the logout is the one that is logged in and has clicked logout. It does this by sending its own authentication token (Django Rest Framework Community, 2021).

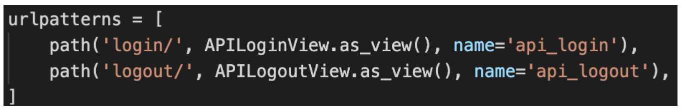
4.3.3 URLS

Now that all of the API functions had been written, they needed to be given URLs on the server so that they can be accessed by the React Application. This was achieved by adding two paths to the *urlpatterns* list that is stored in *mainapp.urls*:



```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/auth/', include('users.urls')),
    path('api/prediction/', include('prediction.urls')),
]
```

The admin path is the one that takes you to the dashboard (Gaurav, 2020). *'api/auth/'* is used for the authentication APIs and *'api/prediction'* is used for the prediction API. These work almost like folders, so within the *urls.py* file for the users app and prediction app, more patterns need to be written for the individual API calls. *users.urls* can be seen below.


```
urlpatterns = [
    path('login/', APILoginView.as_view(), name='api_login'),
    path('logout/', APILogoutView.as_view(), name='api_logout'),
]
```

This links the API functions that were written earlier to a final path. This means that if a login request needs to be sent, it is sent to *127.0.0.1:8000/api/auth/login*.

4.3.4 CORS

CORS stands for Cross-Origin Resource Sharing. It is used when a browser on a device tries to access a different domain or port. CORS is defaulted to be disabled for security reasons, as there are less times that this kind of access is required than more (Hobbs, 2019). However, this app requires CORS to be enabled, as the backend and frontend are on different servers.

CORS was added to the Django server by adding *'corsheaders'* to the main app and then adding my react app development server to its whitelist. This means that CORS is only enabled between the Django server and the React server, so it maintains its security (Kat, 2020).

4.4 The React App

4.4.1 Base Application and Main Screen

The base application was created using Node and Expo. This creates all the necessary files and directories needed for a React-Native application, including a blank screen which could then be customised for my application. This is achieved by running the terminal command:

```
expo init react-app
```

The Login Screen was the first screen to be developed.

4.4.2 Login Screen

The screen was developed to look like the wireframe seen in Section 3.4. Two hooks were implemented that the screen uses to track the change of state of the text inputs. This means that when the form is submitted, the values for the username and password are easily accessible.

The text inputs were put into a *View* (a container, similar to a HTML *div*) so that separate styling can be applied to them easily (View, 2021). Each entry has a placeholder text so that the user knows what they are supposed to input in that area. The password entry is made as a *secureTextEntry* so that it does not display what characters the user has entered.

The login button consists of a *Text* component wrapped within a *TouchableOpacity* component. This *TouchableOpacity* makes the *Text* component clickable, and by setting the height and width of the button this makes the clickable area larger than just the text (Habchi, 2019).

Finally, a stylesheet was created to contain all of the styling for the screen. By putting it all in a stylesheet which is then referenced when required, it keeps the code looking neat and easily readable. It also meant that this stylesheet could be used for the other screens. This gets rid of the need to copy and paste the same styling. Furthermore, if in the future it was decided to change the styling of the inputs, it would only have to be changed in one place (Dua, 2018). At this stage, all of the components for the Login Screen had been placed and styled. The final version can be seen in Figure 19.

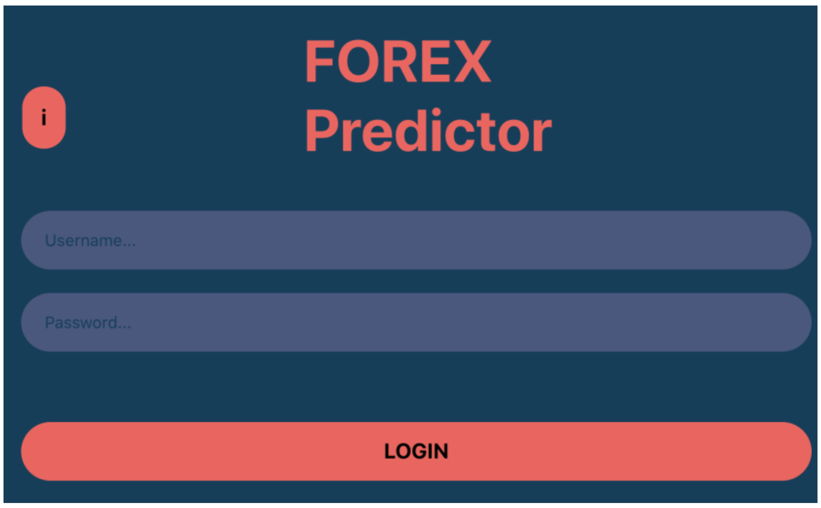


Figure 19 – The final Login Screen

After making all of the components, the next step was to add the login functionality.

4.4.2.1 Redux

In order to keep track of whether a user is logged in, a Redux store needs to be set up to keep track of the authentication token between screens. Redux is a package used to create containers that store information to be used between screens (Ighodaro, 2020). Once this was set up, a simple check can be run to see if there is an authentication token stored when loading the screen, and if not, the application would re-route the user to the login page.

A Redux store is comprised of three basic concepts: Actions, Reducers and Stores. The philosophy of Redux in React is as follows: "Whenever an Action is dispatched, the Reducer makes changes to the state objects in the Store. The Store is the object that brings the actions, reducers and the state objects together." (Gaurav, 2020).

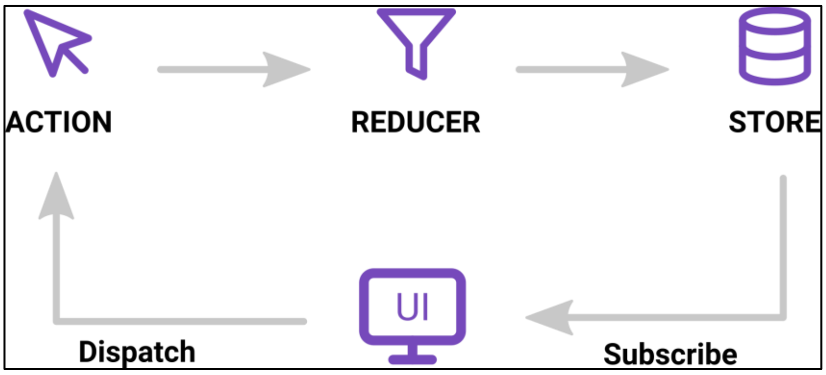
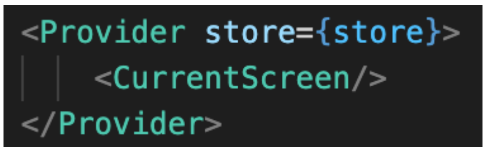


Figure 20 – The Redux Philosophy (Stevanoski, 2019)

The Redux Actions were the first to be developed. One of the most important actions is the *authLogin* action. This takes in a username and password and sends these to the Django server via an API post request. If this request is successful, it dispatches another action called *authSuccess* which saves the authentication token to the local store. If the request fails, an error message is displayed and *authFail* is dispatched, which saves the error to the store. The *authLogout* action was written next which sends the authentication token to the logout API so that it can be removed from the database, and then it removes the token from the store.

The reducers for these actions were written next, which handle changing states within the store (Gaurav, 2020). Each reducer is stored as a separate function, and then a main reducer accesses the correct one via a switch statement. This makes the code more readable and makes adding more actions easier.

The final stage was to wrap the current screen with the store, so that it is accessible from any of the screens. This is extremely important as this is how the application will know if a user is logged in. This can be seen in Figure 21.



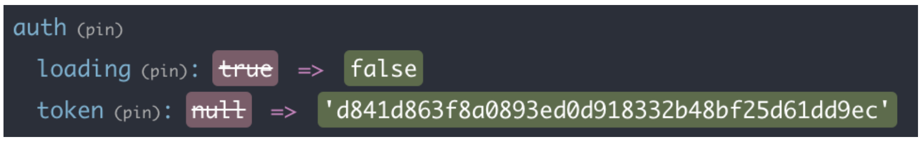
```
<Provider store={store}>
    <CurrentScreen/>
</Provider>
```

Figure 21 – Redux Store wrapping the Current Screen

Two final functions needed to be written for each screen that is going to use the store. A *mapStateToProps* function is required when wanting to access a value stored in the store, and a *mapDispatchToProps* function is required when wanting to use a dispatch hook to call an action (Gaurav, 2020).

4.4.2.2 Adding the Login Functionality

After setting up the store, a dispatch for *authLogin* was added to the *onclick* for the login button. This was then tested this by using the Redux Dev Tools browser extension for Chrome (Diordiev, 2019).



```
auth (pin)
  loading (pin): ~~true~~ => false
  token (pin): ~~null~~ => 'd841d863f8a0893ed0d918332b48bf25d61dd9ec'
```

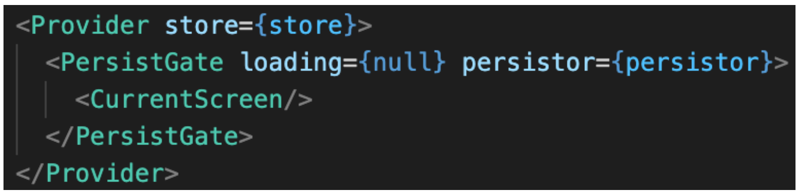
Figure 22 - Redux Store Changing

As seen in Figure 22, when the user logs in, loading is set to true until the process is complete and then changed back to false. The authentication token replaces the null value for token in the store, meaning that it was a success.

4.4.2.3 Persisting the State

The next problem was keeping the state for the user between uses. Currently at this stage, if the user closed the application, the store would be lost and therefore they would have to login every time they opened it. The solution to this is to use the 'redux persist' library. This saves the redux store on every state change to the asynchronous storage on the device. This means that when the app is closed down, the data can be retrieved from local storage and put back into the store. Since the authentication tokens are stored in the store, they will be loaded in from local storage when the app is opened, therefore removing the need to login again.

The first stage is to wrap the current screen with a *PersistGate* component, which links a *persistor* to the application. It is important that this component goes inside of the store, as it needs to make changes to the store. This can be seen in Figure 23.



```
<Provider store={store}>
  <PersistGate loading={null} persistor={persistor}>
    <CurrentScreen/>
  </PersistGate>
</Provider>
```

Figure 23 – PersistGate in between Store and the Current Screen

Since the user may not be logged into the app upon opening, a check needs to be made as to whether to return login data or null if there isn't any. Once this check is made, if there is login data then it needs to be set within the store. To do this, two more functions were written. One for trying to retrieve the data, and one for setting it to the asynchronous storage. Since the processes are asynchronous a promise is returned. An await is then used so that it is certain that the promise has been fulfilled before continuing (mozilla, 2021).

4.4.3 Home Screen

After adding the login functionality, the home screen was the next to be added. The same formatting as the login button was used for all of the buttons on this screen to keep the design synonymous.

The currency buttons were made to be their own type of component. The base and quote currencies are passed into the button and it adds the text name. Then when a button is clicked, these currencies are passed through to the prediction page so that the application can load up the correct history and use the correct model.

The final version can be seen in Figure 24.

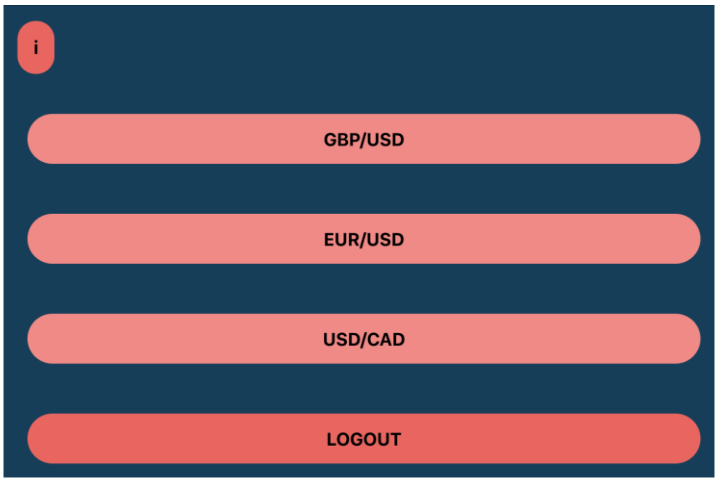
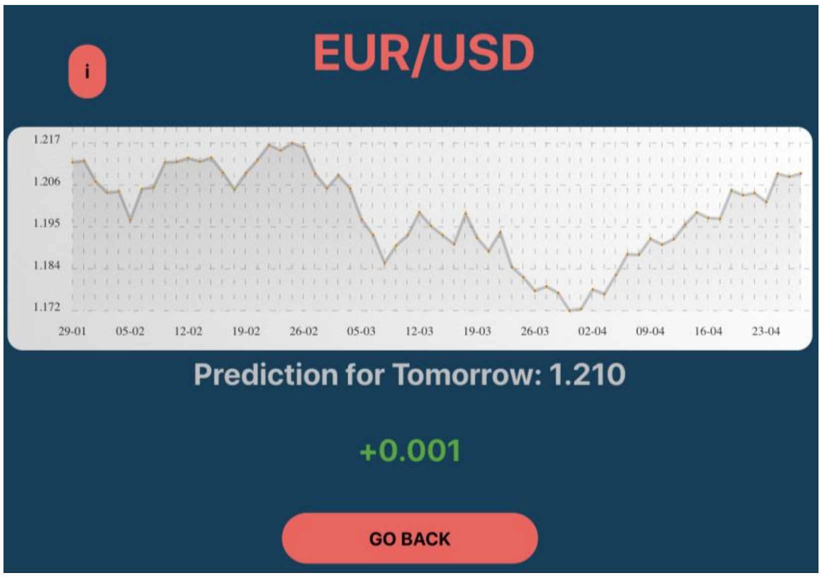


Figure 24 – The Final Home Screen

4.4.4 Prediction Screen

The prediction screen uses a value passed to it containing the name of the button that was clicked to access the screen. This value is used to label the title correctly, but most importantly it is used as the body of the API call to the Django server to receive the historical values and the prediction. The historical values are then used to populate the line chart. The final version of the screen can be seen in Figure 25.



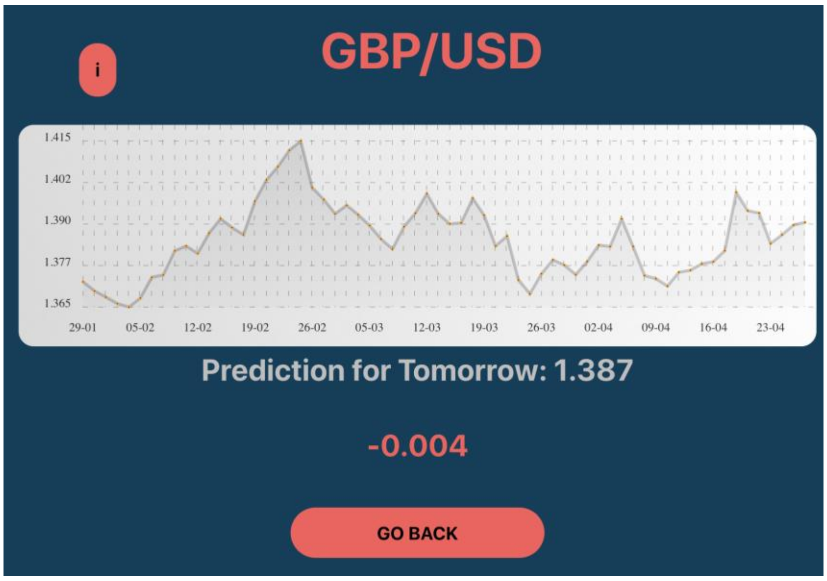
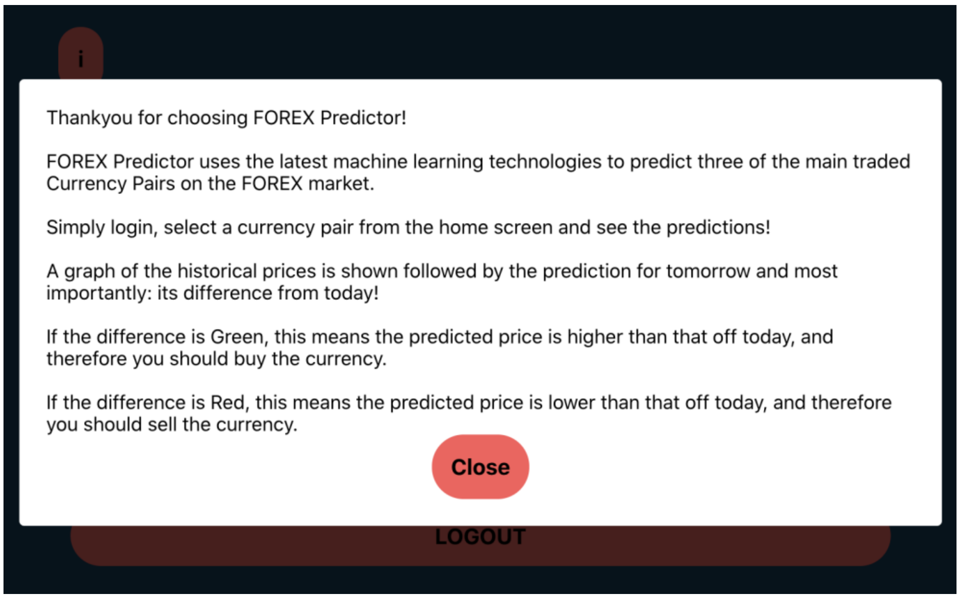


Figure 25 – The final Prediction Screen

4.4.1 Information Dialog

The information dialog was developed in its own file, so that it could be used on every screen without the need of copying and pasting the code. It takes in a parameter called *startState*. This indicates whether on screen load it should be showing or not. The login screen is set to true so that when anybody opens the app or website for the first time, they are greeted by the dialog and given information about the application. The dialog can be seen in Figure 26.


Thankyou for choosing FOREX Predictor!

FOREX Predictor uses the latest machine learning technologies to predict three of the main traded Currency Pairs on the FOREX market.

Simply login, select a currency pair from the home screen and see the predictions!

A graph of the historical prices is shown followed by the prediction for tomorrow and most importantly: its difference from today!

If the difference is Green, this means the predicted price is higher than that off today, and therefore you should buy the currency.

If the difference is Red, this means the predicted price is lower than that off today, and therefore you should sell the currency.

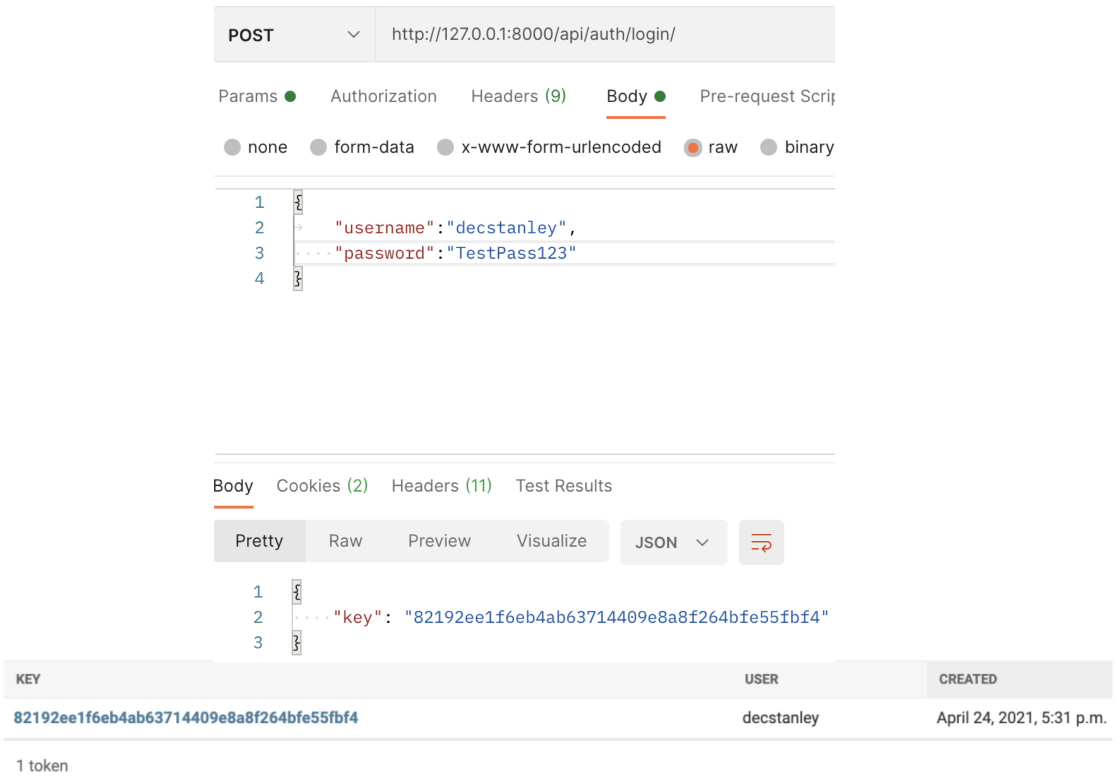
Figure 26 – The Final Information Window

5 Testing and Evaluation

5.1 API Testing

Once developed, the APIs were tested to make sure that they functioned correctly. This was achieved by the use of Postman (Postman Inc, 2014).

First, a mock request was sent to the login API containing login details. This then returned back an authentication token, which could be seen both in Postman and on the admin dashboard, as seen in Figure 27.



```
{
    "username":"decstanley",
    "password":"TestPass123"
}
```

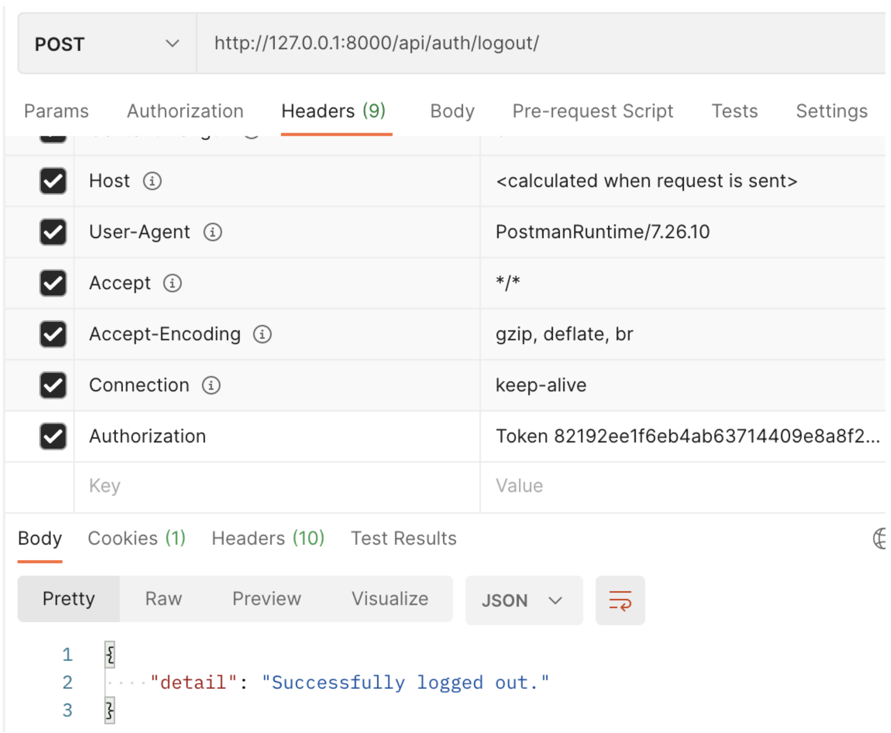
```
{
    "key": "82192ee1f6eb4ab63714409e8a8f264bfe55fbf4"
}
```

KEY	USER	CREATED
82192ee1f6eb4ab63714409e8a8f264bfe55fbf4	decstanley	April 24, 2021, 5:31 p.m.

1 token

Figure 27 – The results of sending a request to 127.0.0.1:8000/api/auth/login

Logout was the next API to be tested. The authorisation token needed to be sent as a header and as this is the only thing required for logout, there was no actual body for the request. As seen in the response, logout was successful, and this can be seen on the admin dashboard as the token has been removed, see Figure 28.



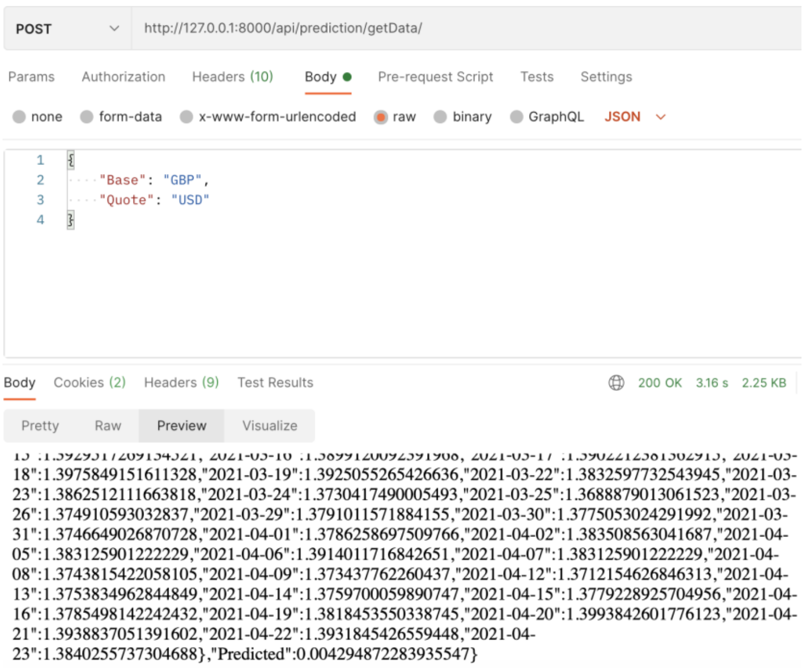
POST http://127.0.0.1:8000/api/auth/logout/

Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.26.10
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Token 82192ee1f6eb4ab63714409e8a8f2...
Key	Value

```
{
    "detail": "Successfully logged out."
}
```

Figure 28 – The results of sending a request to 127.0.0.1:8000/api/auth/logout

The final test is for the prediction API. This request contained the base and quote currencies so that the correct historical data can be retrieved and also so that it uses the correct machine learning algorithm. As stated previously, an authentication token also needed to be passed through the authorization header.


POST http://127.0.0.1:8000/api/prediction/getData/

```
{
    "Base": "GBP",
    "Quote": "USD"
}
```

200 OK 3.16 s 2.25 KB

```
15":1.3929317269134521,"2021-03-16":1.3899120092391968,"2021-03-17":1.3902212381362913,"2021-03-
18":1.397584915161 1328,"2021-03-19":1.3925055265426636,"2021-03-22":1.3832597732543945,"2021-03-
23":1.386251211 1663818,"2021-03-24":1.3730417490005493,"2021-03-25":1.3688879013061523,"2021-03-
26":1.374910593032837,"2021-03-29":1.3791011571884155,"2021-03-30":1.3775053024291992,"2021-03-
31":1.3746649026870728,"2021-04-01":1.3786258697509766,"2021-04-02":1.383508563041687,"2021-04-
05":1.383125901222229,"2021-04-06":1.3914011716842651,"2021-04-07":1.383125901222229,"2021-04-
08":1.3743815422058105,"2021-04-09":1.373437762260437,"2021-04-12":1.3712154626846313,"2021-04-
13":1.3753834962844849,"2021-04-14":1.3759700059890747,"2021-04-15":1.3779228925704956,"2021-04-
16":1.378549814224 2432,"2021-04-19":1.3818453550338745,"2021-04-20":1.3993842601776123,"2021-04-
21":1.3938837051391602,"2021-04-22":1.3931845426559448,"2021-04-
23":1.3840255737304688},"Predicted":0.004294872283935547}
```

Figure 29 – The results of sending a request to 127.0.0.1:8000/api/prediction/getData/

The response from this request as seen in Figure 29, shows the last 90 day's closing prices for the 'GBP/USD' pair, and also its predicted difference for tomorrow.

5.2 Software Testing

Regression testing was completed throughout development, to ensure that any functions written continued to work even after change. Upon completion of the prototype, thorough exploratory testing was also completed to ensure that no bugs were present before giving to users for evaluation.

5.3 User Evaluation

In order to evaluate the user experience, an anonymous survey (Appendix F) was completed by five individuals. The survey used the System Usability Scale (SUS) (Brooke, 2013) questions so that a score could be calculated. There are a set of ten statements, where the respondents answer on a scale from strongly disagree to strongly agree. These are then given values from 1 being strongly disagree to 5 being strongly agree.

To calculate the score: (Smyk, 2020)

- Add up the scores for all of the odd number questions and subtract 5. This becomes X
- Add up the scores for all of the even number questions and subtract that total by 25. This becomes Y.
- Finally add X and Y together and multiply by 2.5

Averaging each respondent's score gives an overall score out of 100 for the application.

For example: one respondent scored 2, 2, 5, 1, 4, 2, 5, 2, 5, 2.

$X = (2 + 5 + 4 + 5 + 5) - 5$
$= 21 - 5$
$= 16$

$Y = 25 - (2 + 1 + 2 + 2 + 2)$
$= 25 - 8$
$= 17$

$\text{Score} = (16 + 17) * 2.5$
$= 34 * 2.5$
$= 85$

- Respondent 2 Scored 62.5
- Respondent 3 Scored 95
- Respondent 4 Scored 77.5
- Respondent 5 Scored 90

This gives FOREX predictor a final average score of 82 out of 100. As seen in Figure 30, this is classed as good on the acceptability scale. There is room for improvement however, as respondent 2 scored low in comparison to the rest. This means that there must be some users that would find using the app difficult. More participants could be used in the future to see whether this user is an anomaly, or whether there is a group of people that do struggle using FOREX Predictor.

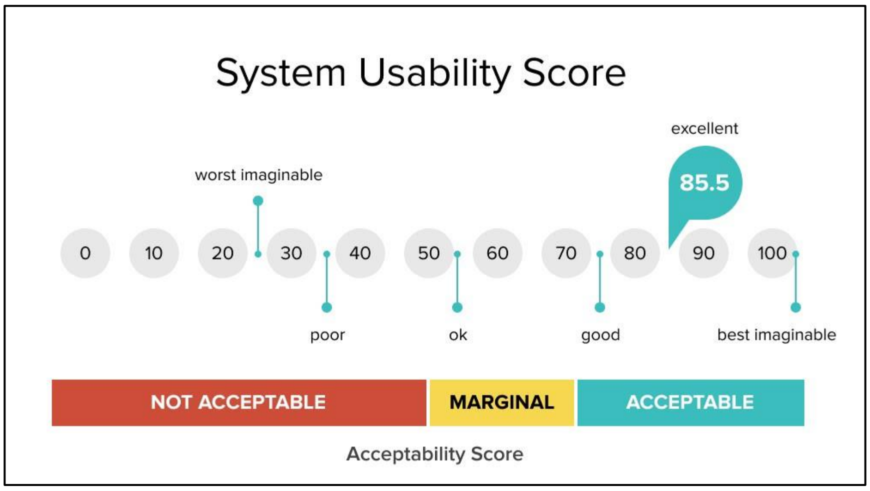


Figure 30 – Chart showing acceptable a usability score is (Smyk, 2020)

6 Critical Reflection

6.1 Deliverable Success

The deliverable of the project has met most of the project aims defined in the project specification (Appendix A) and received good praise in the user feedback survey. An open-source method for finding historical prices for currency pairs was found. This method is through the use of a library meaning that new data can be pulled automatically by the server.

Research was undertaken into frameworks that can be used to create native applications for both iOS and Android, resulting in the use of React Native. Research was also undertaken into how to generate push notifications for the deliverable to use.

The deliverable has been designed so that both Chad (Appendix C) and Ken (Appendix D) can use the application as set in their respective scenarios. Low-level and high-level fidelity designs were created using paper prototyping and software based wireframing, and the final deliverable follows these designs.

Two machine learning algorithms were trained to find the best model to use for the final deliverable. An application was developed that utilises the trained model and displays the historical information through the use of a graph and the predictions are displayed in large bold text so that they are easy to read and looks aesthetically pleasing.

A login system was also implemented, which is not part of the original aims. Version control and project management tools were successfully used to track progress.

6.2 Deliverable Limitations

Throughout the project several problems did arise and as a result there a few shortcomings. The first major problem that occurred arose with the login feature. The tutorial followed was designed for React, not React Native. This meant that changes needed to be made so that it would work for both native applications and accessing it through a browser and this took far longer than expected. As a result, the deliverable currently does not implement the sign-up feature. This means currently an admin needs to create a user account for the application. Following on from this, after many hours of debugging and result, it was discovered that Django does not allow other machines on the network to access the server when hosted locally on 127.0.0.1 (Django, django-admin and manage.py). Even after changing it to use the device IP, the problem persisted. It was then decided that further implementation would be tested only via web browser. Since the application could only be tested on web browsers, push notifications were never implemented.

Despite not being able to test it on mobile devices, the deliverable was still developed so that native applications could be used if the server was to be hosted publicly. Because of this, finding interactive graph libraries that would work for both native and web applications proved very difficult. The best option was to use a library called *CanvasJS*, however this library is designed for just pure React. React Native implementations were possible, however there was very little documentation in regard to it. Furthermore, it required putting the chart in a *WebView*, which defies the point of creating a native application.

Pushed for time due to the login issue, it was decided that FOREX Predictor would just utilise a normal graph instead of an interactive one.

6.3 Future Work

If the project was to be continued in the future, there are several features that could be implemented to improve the application, as well as fixing the limitations that have been highlighted above.

- **Additional Charts:** Expert traders use a variety of different charts to help them determine whether they should buy or sell a currency. These charts could be added to FOREX Predictor to help users learn how to make informed decisions themselves. This would also open up the app's audience to more experienced traders as they would have all of the tools that they need to make their decisions

- **Tutorial Videos and Pages:** In order to learn how to trade FOREX professionally, the app could contain informative videos and tutorial documents that users can watch/read in order to gain knowledge on the subject. These tutorials would range from beginner to advanced so that people of varying levels of knowledge would all be able to use them to improve their knowledge.

- **Training models using technical indicators:** As outlined in Section 2.1.2, experienced traders use several different technical indicators to help them determine whether to buy or sell. These technical indicators could be used as inputs for the machine learning models to see whether they improve the accuracy. If so, these models could then be saved and put on the server for use.

- **Implement Trading Directly within the App:** There are several available APIs that can be used to implement trading. One of these being FXCM (FXCM). This would encourage more people to use the app, as this would mean that everything that they would need to trade Forex would be included.

6.4 Project Management

Task management could have been better for this project. Other assignments took priority at several points over the year, resulting in tasks that were in the 'doing' section of the Kanban board being there for longer periods of time than necessary. This then meant that the backlog remaining large for a long period of time, resulting in some tickets not being started. The final board can be seen in Appendix E.

However, a development log was kept from the start, so that it could be used for reference later on when reflecting on the work that was accomplished. A list of websites that were used to aid in development was also kept from the start to be used for reflection. This list also became useful whenever a bug occurred, as a relevant website could be used from it to help find a solution.

6.5 Ethical Issues and Considerations

The current implementation only stores usernames and passwords; however these are still covered by GDPR and therefore need to be stored securely. The database and server libraries were selected with security in mind, as highlighted in Sections 2.8 and 2.9.

If the project was to be continued in the future, and trading was implemented, then much more personal data would be stored. This data would potentially include card details. Storage of these would need to meet the Payment Card Industry Data Security Standard (PCI-DSS). Alternatively, a third party specialised in handling payments could be used. An example of one of these is Stripe (Stripe, 2010).

Since the user evaluation questionnaires were anonymous, there is no GDPR issues with them. Participants were still provided with contact information for if they have any queries, and all had access to a virtual copy of a consent form (Appendix B) which they could choose to keep.

6.6 Personal Development

Throughout this project, I have learnt so much and developed a huge number of skills. These include time management, server development and hybrid application development.

I have always been interested in the use of machine learning for real world applications, and this project has allowed me to research into the latest technologies and how to deploy models to be used by applications. I had never developed a system that required coding in different programming languages and making them interact, and this project gave me the opportunity to learn how to do this. Now I feel confident enough to tackle new problems using all of the technologies and methods that I have learnt throughout this project.

Never before had I followed a software development methodology. I used to just like to get stuck in with the coding with little planning and thought. Completing this project has shown me why these methodologies exist and all of the benefits of following them. By taking time to research into different solutions for problems, it allows you to compare the benefits of them and in term makes the development process easier.

In hindsight I would've planned my time better so that I dedicated more time to this project. I have really enjoyed completing it and would love to see the application in its full form that was outlined in the project specification. In future projects, I will still use the Kanban methodology but still give myself certain deadlines for tickets in order to keep progression high.

7 Appendices

7.1 Appendix A - Project Specification

PROJECT SPECIFICATION - Project (Technical Computing) 2020/21

Student:	**Declan Stanley**
Date:	**22nd October 2020**
Supervisor:	**Dr Jing Wang**
Degree Course:	**Computer Science**
Title of Project:	**Automating FOREX Trading using Machine Learning**

Elaboration

FOREX (Foreign Exchange) trading is becoming more and more popular every day, however starting up can be extremely difficult; as to an unexperienced eye patterns and trends are almost impossible to spot. This project aims to remove this difficulty by showing predicted prices of currency pairs, and suggesting to the user when to buy or sell.

Predictions will be made through technical analysis of the history of the currency pair, which will include checking for specific indicators and following trends.

To further advance this project I aim to create a mobile application that can be used to monitor the current prices and predictions of currency pairs and send notifications to the user when the system thinks that the user should buy or sell.

Project Aims

- Investigate ways to obtain data containing the history of currency pairs in order to train the algorithm and as a result make predictions
- Research methods of developing mobile applications for both IOS and android
- Research how to generate push notifications for apps
- Produce an algorithm that predicts currency pair trends to a 55% accuracy
- Produce a mobile application that utilises the algorithm and displays information to the user in an aesthetically pleasing way
- Add push notifications to the application so the user knows when to check the app
- Evaluate the effectiveness of the algorithm, suggesting methods of improving its accuracy
- Evaluate the mobile application and discuss any future improvements that can be made to it

Project deliverable(s)

I will deliver a prototype mobile application that will be on the Android and IOS platforms.
The end users of my application will be anybody that is interested in starting to trade FOREX, regardless of knowledge of FOREX trading and technical ability.
The prototype application will utilise a machine learning algorithm to predict the prices of FOREX currency pairs and display these to the user in an easy to interpret way. It will also show the history of the currency pair, and send the users notifications when it thinks that they should buy/sell.

Action plan

Task Deadlines

Task	Deadline Date
Find a data source for the currency pairs	Friday 25th October
Research languages for app development and experiment with them	Friday 30th October
Research into existing algorithms and products with similar functions	Friday 30th October
Research indicators and trends to use in the prediction algorithm	Friday 6th November
Learn how to create push notifications	Friday 13th November
Design initial machine learning algorithm	Friday 20th November
Design mobile application	Friday 27th November
Implement machine learning algorithm	Friday 18th December
Implement mobile application that displays current and recent currency pair prices aesthetically	Friday 15th January
Integrate mobile application and machine learning algorithm to show predictions	Friday 29th January
Add push notifications to the app suggesting to the user when to buy and sell	Friday 5th February
Test and refine machine learning algorithm	Friday 19th February
User testing	Friday 26th February

Milestone Deadlines

Task	Deadline Date
MS1: Initial machine learning algorithm developed and trained	Friday 18th December
MS2: First application prototype that shows predictions developed	Friday 29th December

MS3: Prototype containing all features developed	Friday 5th February

Module Deadlines

Task	Deadline Date
Project Specification and Ethics Form	Friday 23rd October
Information Review	Friday 4th December
Provisional Contents Page	Friday 19th February
Draft Critical Evaluation	Friday 19th March
Draft Project Report	Friday 19th March
Project Report and Deliverables Hand In	Thursday 15th April
Demonstration of Work	Agreed before 29th April

BCS Code of Conduct

I confirm that I have successfully completed the BCS code of conduct on-line test with a mark of 70% or above. This is a condition of completing the Project (Technical Computing) module.

Signature: [illegible]

Publication of Work

I confirm that I understand the "Guidance on Publication Procedures" as described on the Bb site for the module.

Signature: [illegible]

GDPR

I confirm that I will use the "Participant Information Sheet" as a basis for any survey, questionnaire or participant testing materials. This form is available on the Bb site for the module and as an appendix in the handbook.

Signature: [illegible]

7.2 Appendix B – Participant Consent Form

PARTICIPANT CONSENT FORM

Automating FOREX Trading using Machine Learning

Please answer the following questions by ticking the response that applies

	YES	NO
1. I have read the Information Sheet for this study and have had details of the study explained to me.	☐	☐
2. My questions about the study have been answered to my satisfaction and I understand that I may ask further questions at any point.	☐	☐
3. I understand that I am free to withdraw from the study within the time limits outlined in the Information Sheet, without giving a reason for my withdrawal or to decline to answer any particular questions in the study without any consequences to my future treatment by the researcher.	☐	☐
4. I agree to provide information to the researchers under the conditions of confidentiality set out in the Information Sheet.	☐	☐
5. I wish to participate in the study under the conditions set out in the Information Sheet.	☐	☐
6. I consent to the information collected for the purposes of this research study, once anonymised (so that I cannot be identified), to be used for any other research purposes.	☐	☐

Participant's Signature:
____ **Date:** _____________

Participant's Name (Printed):
____

Contact details:
____
____

____
____

Researcher's Name (Printed):
____

Researcher’s Signature:

____

Researcher's contact details:
Declan Stanley
128 Duchess Road
Sheffield
S2 4BL
07939139492
declan.stanley1999@outlook.com

Please keep your copy of the consent form and the information sheet together.

7.3 Appendix C – Chad Wright

7.3.1 Persona

Chad Wright is a 28-year-old male from Lincoln. He is in full time work as an equine veterinarian and because of this does not have a lot of free time. Chad would like to start investing some of his money that he earns, however does not know anything about trading stocks or currencies. He has grown up around technology his whole life and is therefore an avid smartphone user. He always tries to use his smartphone to make aspects of his daily life easier.

7.3.2 Scenario

It is a rainy Tuesday afternoon. The time is 20:30. Chad has just gotten back from a long day at work. He is very tired and decides to go relax by lying in bed. When scrolling through social media, he receives a message from a friend recommending an app to help him invest his money. He initially shrugs this off and tells his friend that he does not have the time to learn how to invest. His friend then tells him that the app does all the hard work for him and notifies him when he should buy or sell.

This sparks Chad’s interest, and he decides to download the app to find more. When he opens the app, he sees a login page with a signup button. Chad is very pleased with the darker colour scheme used by the app, as it does not irritate his tired eyes. He signs up and is greeted by the homepage. He then decides that he wants to go to sleep, and that he will explore the app the next day.

After waking up, Chad checks his phone and sees that he has a notification. It is from the prediction app he downloaded the night before and see’s that it is giving him a suggestion to sell USD/CAD. He likes that he has needed minimal interaction with the app and yet it is already trying to help him.

Chad is still slightly concerned about the accuracy of the predictions, so he decides to open the app and presses on the USD/CAD button. This takes him to a screen displaying a graph of the historical prices of the currency, and the predicted price for the next day, followed by the difference. He takes note of this difference and gets on with his day. He then checks the app every morning when he wakes up to see how accurate the predictions are. After two weeks he gains confidence, and starts to trade currencies using the predictions made by the app.

7.4 Appendix D – Ken Agbanobi

7.4.1 Persona

Ken Agbanobi is a 70-year-old grandfather from Manchester. He is retired and wanting to start investing some of his money to help pay for his grandchildren's university fees. Ken has never done any trading before and is not very comfortable with technology. He does not own a smartphone; however he does own a PC which he uses to check emails and keep in touch with family through social media. He also has poor eyesight due to his age, and therefore requires text to be bold and clear.

7.4.2 Scenario

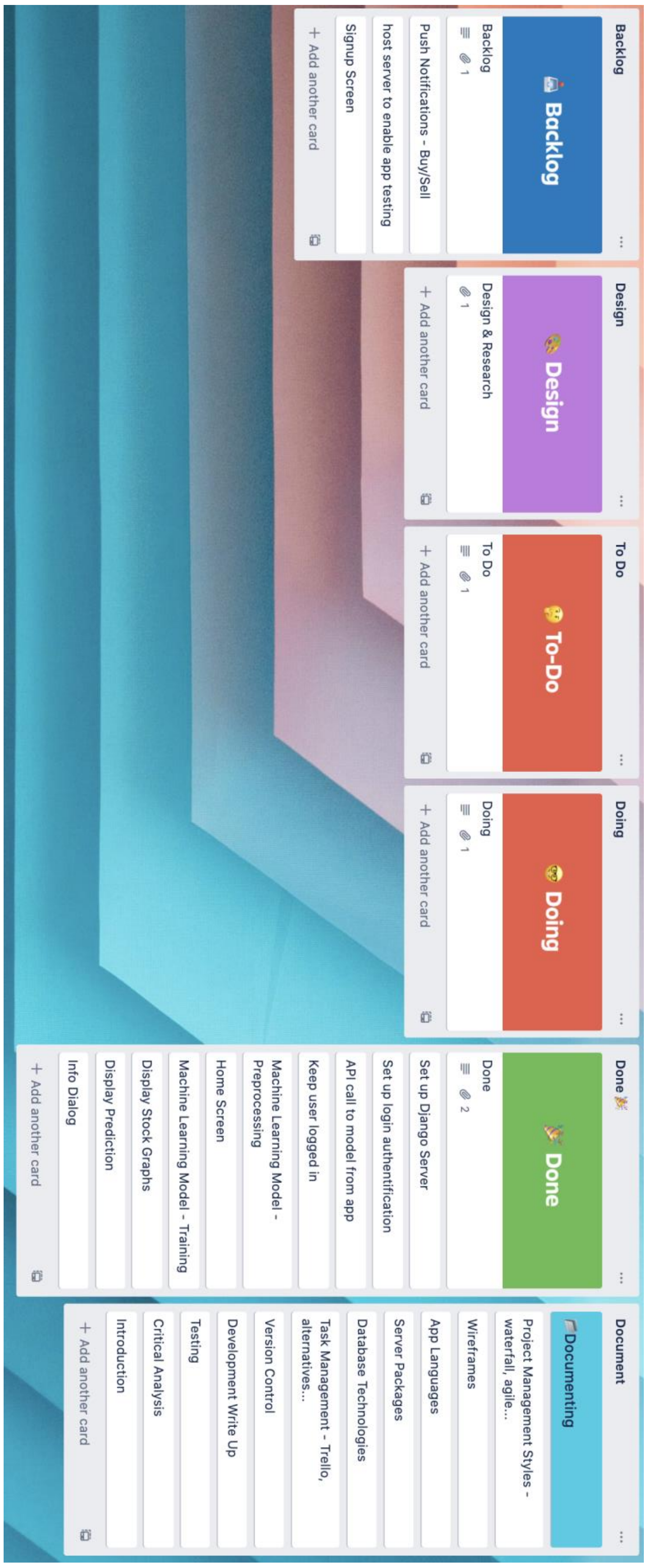
It is the 20th of April 2022 at 2pm. Ken has just finished eating his lunch and is scrolling through his social media. He sees an advertisement for a website that says it can be used to help people trade to make money, even with no knowledge on the subject. Ken clicks on the link to find out more.

He sees an information icon and clicks it. A box appears containing a description of what the website does and how to use it. After reading this, Ken feels at ease knowing that it isn't just a scam. He signs up and then explores the site to reinforce what he learnt from the information box.

Ken clicks on the EUR/USD button which takes him to a screen displaying a large graph. When hovering over the graph, he notices that prices start to appear next to the points that he hovers over. Ken then sees bold text under the graph clearly stating the prediction price for the next day and the difference compared to the current day. He likes that this information is displayed separately as without he would be put off by just seeing a graph that he doesn’t understand.

Ken decides to check the website daily after eating his lunch for a few days to gain confidence in the predictions before committing to start trading and following the predictions. After a week Ken trusts the app and starts to use the predictions to make trades, using the recommended site found in the information section.

7.5 Appendix E – Kanban Board at End of Project



Backlog

Backlog

Backlog

1

Push Notifications - Buy/Sell

host server to enable app testing

Signup Screen

+ Add another card

Design

Design

Design & Research

1

+ Add another card

To Do

To-Do

To Do

1

+ Add another card

Doing

Doing

Doing

1

+ Add another card

Done

Done

Done

2

Set up Django Server

Set up login authentification

API call to model from app

Keep user logged in

Machine Learning Model - Preprocessing

Home Screen

Machine Learning Model - Training

Display Stock Graphs

Display Prediction

Info Dialog

+ Add another card

Document

Documenting

Project Management Styles - waterfall, agile...

Wireframes

App Languages

Server Packages

Database Technologies

Task Management - Trello, alternatives...

Version Control

Development Write Up

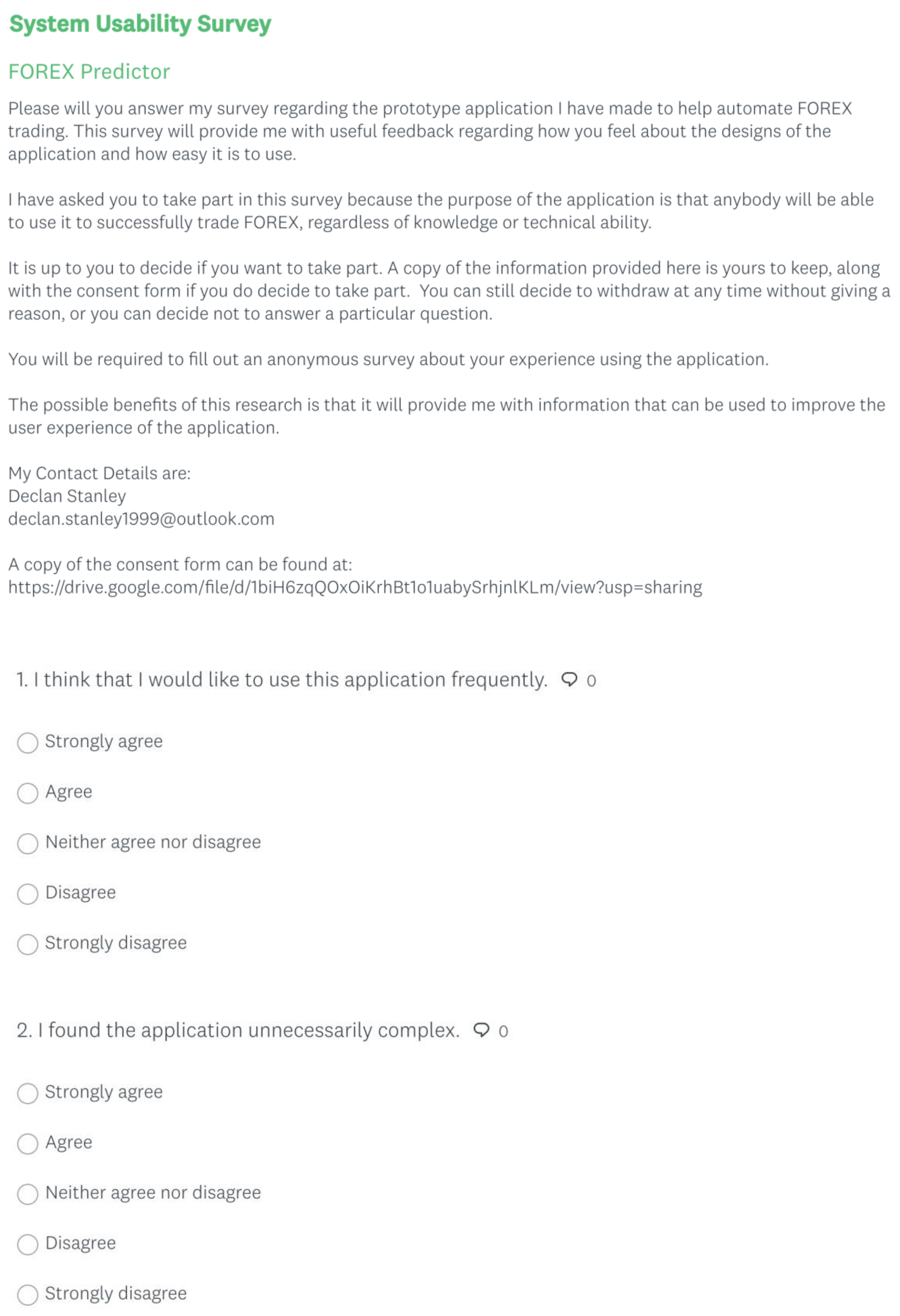
Testing

Critical Analysis

Introduction

+ Add another card

7.6 Appendix F – Usability Survey



System Usability Survey

FOREX Predictor

Please will you answer my survey regarding the prototype application I have made to help automate FOREX trading. This survey will provide me with useful feedback regarding how you feel about the designs of the application and how easy it is to use.

I have asked you to take part in this survey because the purpose of the application is that anybody will be able to use it to successfully trade FOREX, regardless of knowledge or technical ability.

It is up to you to decide if you want to take part. A copy of the information provided here is yours to keep, along with the consent form if you do decide to take part. You can still decide to withdraw at any time without giving a reason, or you can decide not to answer a particular question.

You will be required to fill out an anonymous survey about your experience using the application.

The possible benefits of this research is that it will provide me with information that can be used to improve the user experience of the application.

My Contact Details are:
Declan Stanley
declan.stanley1999@outlook.com

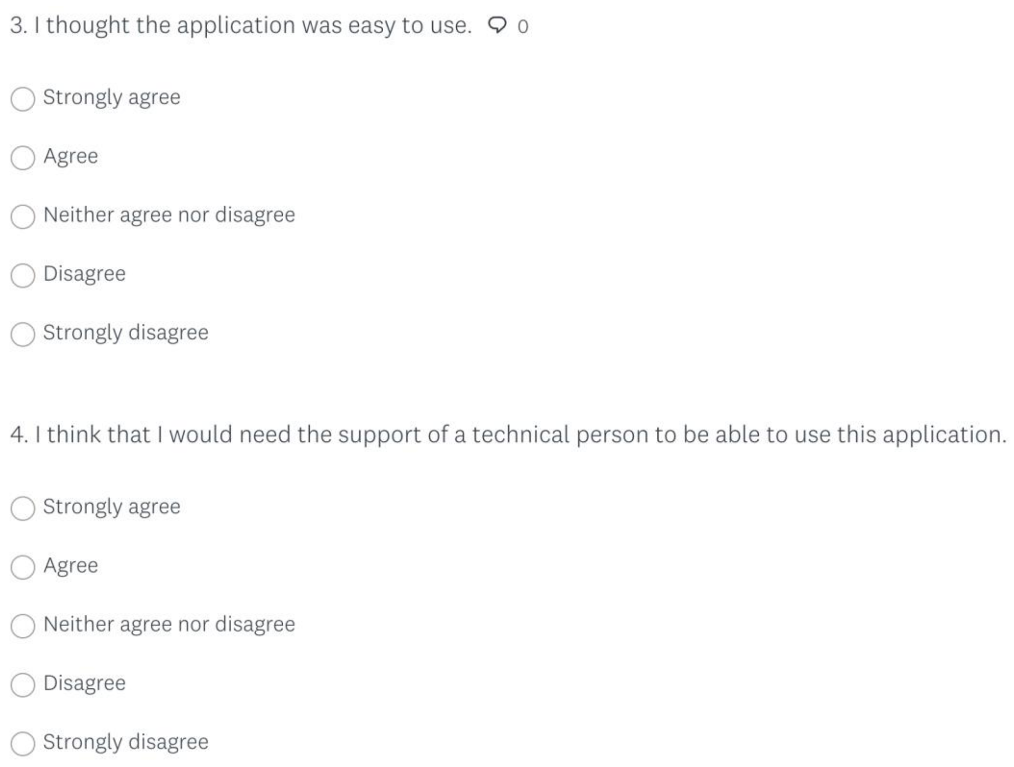
A copy of the consent form can be found at:
https://drive.google.com/file/d/1biH6zqQOxOiKrhBt1o1uabySrhjnlKLm/view?usp=sharing

1. I think that I would like to use this application frequently. 0

- Strongly agree
- Agree
- Neither agree nor disagree
- Disagree
- Strongly disagree

2. I found the application unnecessarily complex. 0

- Strongly agree
- Agree
- Neither agree nor disagree
- Disagree
- Strongly disagree

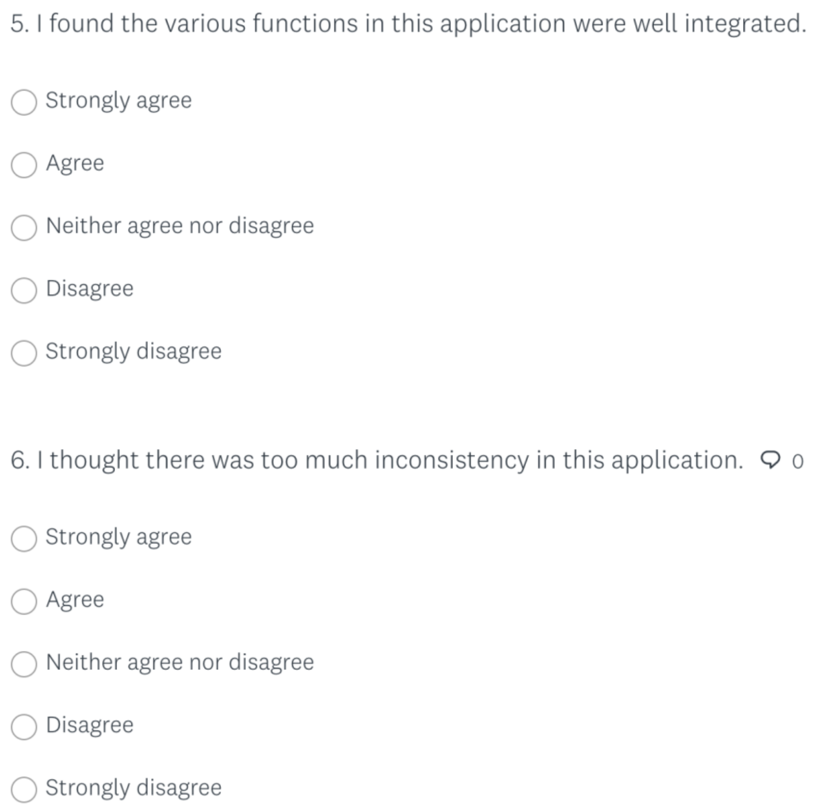


3. I thought the application was easy to use. 0

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree

4. I think that I would need the support of a technical person to be able to use this application.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree

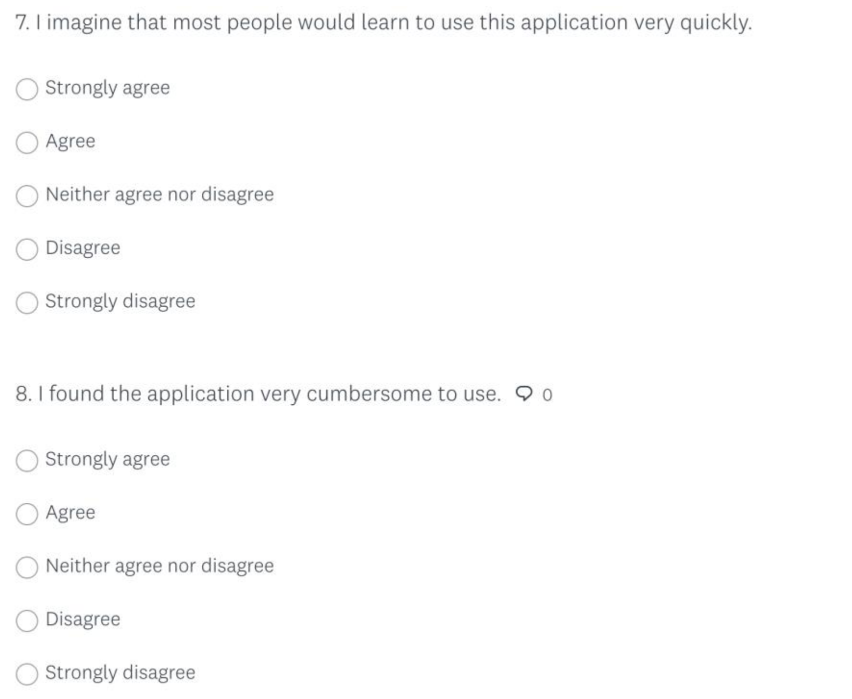


5. I found the various functions in this application were well integrated.

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree

6. I thought there was too much inconsistency in this application. 0

- ◯ Strongly agree
- ◯ Agree
- ◯ Neither agree nor disagree
- ◯ Disagree
- ◯ Strongly disagree



7. I imagine that most people would learn to use this application very quickly.

○ Strongly agree

○ Agree

○ Neither agree nor disagree

○ Disagree

○ Strongly disagree

8. I found the application very cumbersome to use. 0

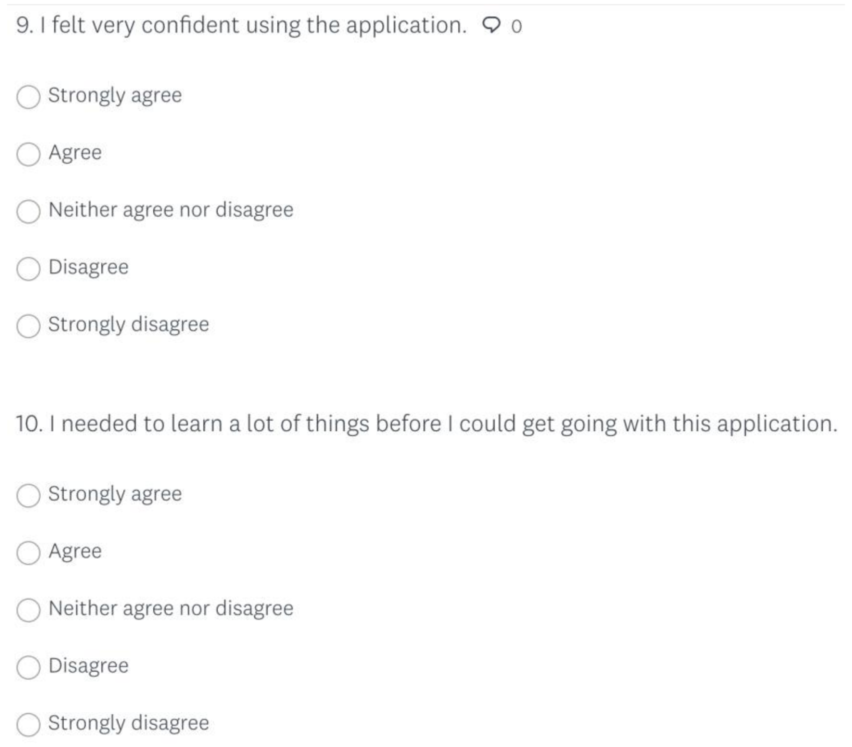
○ Strongly agree

○ Agree

○ Neither agree nor disagree

○ Disagree

○ Strongly disagree



9. I felt very confident using the application. 0

○ Strongly agree

○ Agree

○ Neither agree nor disagree

○ Disagree

○ Strongly disagree

10. I needed to learn a lot of things before I could get going with this application.

○ Strongly agree

○ Agree

○ Neither agree nor disagree

○ Disagree

○ Strongly disagree

Bibliography

Airship. (2021). *Push Notifications Explained.* Retrieved from Airship: https://www.airship.com/resources/explainer/push-notifications-explained/

anthony. (2012, April 5). *8 Reasons Users Don't Fill Out Sign Up Forms.* Retrieved from ux movement: https://uxmovement.com/forms/8-reasons-users-arent-filling-out-your-sign-up-form/

Atlassian. (2011). Retrieved from Trello: https://trello.com/en-GB

Azarian, I. (2013, June 14). *A Review of Software Version Control: Systems, Benefits, and Why it Matters.* Retrieved from Segue Technologies: https://www.seguetech.com/a-review-of-software-version-control-systems-benefits-and-why-it-matters/#:~:text=Version%20Control%20Benefits,development%2C%20QA%2C%20and%20production.

Babeni, S. (2020, January 24). *Most Popular Databases in 2020: Here's How They Stack Up.* Retrieved from Ormuco: https://ormuco.com/blog/most-popular-databases

Babic, A., & Tonheim, A. (2018). User Evaluation of a Multiple Sclerosis Self-Management Mobile Application. *Studies in Health Technology and Informatics 251*, 233-236. Retrieved from https://www.researchgate.net/figure/A-selection-of-four-wireframes-from-the-high-fidelity-prototype_fig1_326187997

Babich, N. (2017, November 29). *Prototyping 101: The Difference between Low-Fidelity and High-Fidelity Prototypes and When to Use Each.* Retrieved from Adobe Blog: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use.html#gs.z6ijqq

Backlog. (2020, June 23). *Git vs. SVN: Which version control system is right for you?* Retrieved from Backlog: https://backlog.com/blog/git-vs-svn-version-control-system/

Banys, D., & Kobran, D. (2020). *Long Short-Term Memory (LSTM).* Retrieved from AI Wiki: https://docs.paperspace.com/machine-learning/wiki/long-short-term-memory-lstm

Banys, D., & Kobran, D. (2020). *Recurrent Neural Network (RNN).* Retrieved from AI Wiki: https://docs.paperspace.com/machine-learning/wiki/recurrent-neural-network-rnn

Barker, T. T. (2002). Documentation for Software and IS Development. In H. Bidgoli, *The Encyclopedia of Information Systems.* Academic Press.

Baron, M., Brogaard, J., & Kirilenko, A. (2014). *Risk and Return in High Frequency Trading.*

Bennett, J. (2019, August 12). *9 Things You Didn't Know About Successful Forex Traders in 2020.* Retrieved from Daily Price Action: https://dailypriceaction.com/blog/successful-forex-traders/

BIS. (2019). *Triennial Central Bank Survey of Foreign Exchange and Over-the-counter (OTC) Derivatives Markets in 2019.*

Bitnine. (2016, June 10). *Advantages of PostgreSQL.* Retrieved from Bitnine: https://bitnine.net/blog-postgresql/advantages-of-postgresql/

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies Vol. 8 Issue 2*, 29-40.

Brownlee, J. (2016, June 20). *Dropout Regularization in Deep Learning Models With Keras.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

Brownlee, J. (2018, November 12). *How to Develop Convolutional Neural Network Models for Time Series Forecasting.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/

Brownlee, J. (2019). *How to Fix the Vanishing Gradients Problem Using the ReLU.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/#:~:text=Vanishing%20gradients%20is%20a%20particular,network%20that%20requires%20weight%20updates.

Business of Apps. (2021, April 27). *Push Notifications Statistics (2019).* Retrieved from Business of Apps: https://www.businessofapps.com/marketplace/push-notifications/research/push-notifications-statistics/#2

Cacoo. (2021). *Wireframe Software*. Retrieved from cacoo: https://cacoo.com/examples/wireframe-software

Cagle, K. (2019, August 3). *The End of Agile.* Retrieved from Forbes: https://www.forbes.com/sites/cognitiveworld/2019/08/23/the-end-of-agile/?sh=268c2cd82071

Chen, J. (2021, January 15). *Technical Indicator.* Retrieved from Investopedia: https://www.investopedia.com/terms/t/technicalindicator.asp#:~:text=Technical%20indicators%20are%20heuristic%20or,to%20predict%20future%20price%20movements.

Croos, P. d. (2018, January 4). *When You Should (and Shouldn't) Use Firebase.* Retrieved from codementor: https://www.codementor.io/@cultofmetatron/when-you-should-and-shouldn-t-use-firebase-f62bo3gxv

Dearmer, A. (2020, December 1). *Firebase vs. MySQL: Battle of the Databases.* Retrieved from Xplenty: https://www.xplenty.com/blog/firebase-vs-mysql/#conclusion

digital.ai. (2020). *The 14th Annual State of Agile Report.*

Diordiev, M. (2019, January 25). *redux-devtools-extension.* Retrieved from GitHub: https://github.com/zalmoxisus/redux-devtools-extension

Django. (2021). Retrieved from Django: https://www.djangoproject.com/

Django. (n.d.). *django-admin and manage.py.* Retrieved from Django: https://docs.djangoproject.com/en/dev/ref/django-admin/#django-admin-option---addrport

Django Rest Framework Community. (2021, April 20). *Class-based Views* . Retrieved from Django Rest Framework: https://www.django-rest-framework.org/api-guide/views/

Dua, Y. (2018, September 3). *Understanding styling in React Native.* Retrieved from MindOrks: https://medium.com/mindorks/everything-to-know-about-styling-in-react-native-7e30aed53ad

Existek. (2019, April 3). *What is the Difference between Native App and Hybrid App?* Retrieved from Existek: https://existek.com/blog/difference-between-native-app-and-hybrid-app/

Expo. (n.d.). Retrieved from Expo: https://expo.io/

Expo. (2020). *Sending Notifications with Expo's Push API.* Retrieved from Expo: https://docs.expo.io/push-notifications/sending-notifications/

Facebook. (2021, March 12). Retrieved from React Native: https://reactnative.dev/

Flask. (2010). Retrieved from Flask: https://flask.palletsprojects.com/en/1.1.x/

Full Stack Python. (2021). *Flask.* Retrieved from Full Stack Python: https://www.fullstackpython.com/flask.html#:~:text=Flask%20is%20considered%20more%20Pythonic,simple%20app%20up%20and%20running.

FXCM. (n.d.). *What is API Trading?* Retrieved from FXCM: https://www.fxcm.com/uk/algorithmic-trading/api-trading/

Gaurav, M. (2020, September 13). *Build a fully production ready machine learning app with Python Django, React, and Docker.* Retrieved from towards data science: https://towardsdatascience.com/build-a-fully-production-ready-machine-learning-app-with-python-django-react-and-docker-c4d938c251e5

Gehman, C. (2018, July 25). *What Is DVCS Anyway?* Retrieved from Perforce: https://www.perforce.com/blog/vcs/what-dvcs-anyway

Gilling, D. (2020, July 7). *Top 10 API Security Threats Every API Team Should Know.* Retrieved from moesiF Blog: https://www.moesif.com/blog/technical/api-security/API-Security-Threats-Every-API-Team-Should-Know/

Github Inc. (2008). Retrieved from GitHub: https://github.com/

Google. (2012). Retrieved from Firebase: https://firebase.google.com/

Google. (2020, October 9). *FCM.* Retrieved from Firebase: https://firebase.google.com/docs/cloud-messaging/

Google. (2021, April 2). Retrieved from Flutter: https://flutter.dev/?gclid=Cj0KCQjwyZmEBhCpARIsALIzmnLyqTy1ZdKVrJMQrnesPw9wyKJb5kwTGSL3eyD0Tg8KHa2c3XL6w2waAqMmEALw_wcB&gclsrc=aw.ds

Google. (2021, March 17). Retrieved from Dart: https://dart.dev/

Habchi, Y. E. (2019, November 17). *React Native Login Screen Tutorial.* Retrieved from React Native Master: https://reactnativemaster.com/react-native-login-screen-tutorial/

Hannah, J. (2020, July 22). *9 Of The Best Login Screen Examples* . Retrieved from CAREERFOUNDRY: https://careerfoundry.com/en/blog/ui-design/best-login-screen-examples/

hayatoy. (2017, February 13). *ml-forex-prediction.* Retrieved from GitHub: https://github.com/hayatoy/ml-forex-prediction

Hobbs, S. (2019, April 16). *CORS Tutorial: A Guide to Cross-Origin Resource Sharing.* Retrieved from auth0: https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/

Ighodaro, N. (2020, October 12). *Why use Redux? A tutorial with examples.* Retrieved from LogRocket: https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/

Infoways, A. (2020, June 8). *Quick Tips For Choosing The Right Mobile App: PWA Vs Hybrid App Vs Native App.* Retrieved from Agile Infoways: https://www.agileinfoways.com/blog/quick-tips-for-choosing-the-right-mobile-app-pwa-vs-hybrid-app-vs-native-app/

Ionos. (2019, August 1). Retrieved from Ionos: https://www.ionos.co.uk/digitalguide/websites/web-development/about-kanban/

Jegadeesh, N., & Titman, S. (1993). Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance Vol. 48, No. 1*, 65-91.

Joblib. (2021, April 2). *Running Python functions as pipeline jobs.* Retrieved from Joblib: https://joblib.readthedocs.io/en/latest/

Jones, & Waddell. (2019, April 23). *The Cascading Costs of Waterfall.* Retrieved from Medium: https://medium.com/@joneswaddell/the-cascading-costs-of-waterfall-5c3b1b8beaec

Kanboard. (n.d.). Retrieved from Kanboard: https://kanboard.org/

Kat, S. (2020, September 8). *How to Fix Django CORS Error.* Retrieved from DZone: https://dzone.com/articles/how-to-fix-django-cors-error

Kukreja, P. (2019, August 14). *Stock-Prediction-using-LSTM.* Retrieved from GitHub: https://github.com/pankush9096/Stock-Prediction-using-LSTM

laddad, A. (2019, March 13). *Basic understanding of LSTM.* Retrieved from Good Audience: https://blog.goodaudience.com/basic-understanding-of-lstm-539f3b013f1e

laddad, A. (2019, March 13). *Basic understanding of LSTM* . Retrieved from Good Audience: https://blog.goodaudience.com/basic-understanding-of-lstm-539f3b013f1e

Lithios. (2021). *Mobile App Development: Native, Hybrid, and React Native*. Retrieved from Lithios: https://lithiosapps.com/mobile-app-development-native-hybrid-and-react-native/#:~:text=React%20Native%20(RN)%2C%20built,has%20several%20%E2%80%9Cnative%E2%80%9D%20features.

Liu, S. (2020). *Cross-platform mobile frameworks used by developers worldwide 2019 and 2020.*

Lucidchart. (2021). *Online wireframe tool*. Retrieved from Lucidchart: https://www.lucidchart.com/pages/landing/wireframe-software?utm_source=google&utm_medium=cpc&utm_campaign=_en_tier1_mixed_search_brand_bmm_&km_CPC_CampaignId=1490375424&km_CPC_AdGroupID=55688906577&km_CPC_Keyword=%2Blucid%20chart%20%2Bwireframe&km_CPC_Mat

Mammarella, N., Domenico, A. D., Palumbo, R., & Fairfield, B. (2016). When Green Is Positive and Red Is Negative: Aging and the Influence of Color on Emotional Memories. *Psychology and Aging 31*, 914-926.

Mesh, J. (2020, February 17). *Kanban 101: How Any Team Can Be More Agile.* Retrieved from A blog for teams by Trello: https://blog.trello.com/kanban-101

Monus, A. (2019, March 19). *Understanding native app development - what you need to know in 2019.* Retrieved from RAYGUN: https://raygun.com/blog/native-app-development/

Morioka, A., & Stone, T. (2008). *Color Design Workbook.* Rockport Publishers.

mozilla. (2021, February 25). *await.* Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await

Nader, Y. (2020, November 1). *What is Django? Advantages and Disadvantages.* Retrieved from hackr.io: https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django

Olah, C. (2015, August 27). *Understanding LSTM Networks.* Retrieved from colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Osman, M. (2020, May 22). *The Differences in Wireframe Fidelity: From Low to High Fidelity Wireframes.* Retrieved from HubSpot: https://blog.hubspot.com/website/high-fidelity-wireframe

Pecherer, A. (2021, March 4). *How much time does it take to learn technical analysis in stock market?* Retrieved from Quora: https://www.quora.com/How-much-time-does-it-take-to-learn-technical-analysis-in-stock-market

Pedamkar, P. (n.d.). *What is PostgreSQL?* Retrieved from EDUCBA: https://www.educba.com/what-is-postgresql/

Postman Inc. (2014). Retrieved from Postman: https://www.postman.com/

Reidy, C. (Director). (2017). *Mobile Application Design : Paper Prototype Video* [Motion Picture].

Royce, D. W. (1970). Managing the development of large software systems. *Proceedings IEEE WESCON*.

Rozwadowski, W. (2020, June 2). *Pros & Cons of Flutter Mobile Development.* Retrieved from Future Mind: https://www.futuremind.com/blog/pros-cons-flutter-mobile-development

Sacolick, I. (2020, February 25). *What is agile methodology? Modern software development explained.* Retrieved from InfoWorld: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html

Saračević, M., & Mašović, S. (2013). Advantages of Acid Compliance in Application Development in Firebird Databases. *Thematic Fields,* 53-61.

Sayyad, R. A. (2020, June 13). *How to Use Convolutional Neural Networks for Time Series Classification.* Retrieved from Medium: https://medium.com/@Rehan_Sayyad/how-to-use-convolutional-neural-networks-for-time-series-classification-80575131a474

Scandiweb. (2021). *COMPLETE GUIDE TO PWA: DEFINITION, TECHNOLOGY, EXAMPLES*. Retrieved from Scandiweb: https://scandiweb.com/blog/learn-all-about-progressive-web-apps/

Scott, G. (2021, April 04). *What Is a Time Series?* Retrieved from Investopedia: https://www.investopedia.com/terms/t/timeseries.asp

Sheppard, D. (2017). *Beginning Progressive Web App Development.* Apress.

Shih, S.-Y., Sun, F.-K., & Lee, H.-Y. (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning 108*, 1421-1441.

Siderova, S. (2018). *The Kanban Method: The Ultimate Beginner's Guide!* Retrieved from nave: https://getnave.com/blog/what-is-the-kanban-method/

Sidorenko, V. (2017, May 8). *The Advantages And Disadvantages of Using Django.* Retrieved from Datafloq: https://datafloq.com/read/advantages-and-disadvantages-of-using-django/3050

Smyk, A. (2020, March 28). *The System Usability Scale & How it's Used in UX.* Retrieved from Medium: https://medium.com/thinking-design/the-system-usability-scale-how-its-used-in-ux-b823045270b7

Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* Morgan Kaufmann.

Stevanoski, H. (2019, August 30). *The only introduction to Redux (and React-Redux) you'll ever need.* Retrieved from Medium: https://javascript.plainenglish.io/the-only-introduction-to-redux-and-react-redux-youll-ever-need-8ce5da9e53c6

Stevenson, D. (2018, September 24). *What is Firebase? The complete story, abridged.* Retrieved from Medium: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

Stripe. (2010). Retrieved from Stripe: https://stripe.com/gb?utm_campaign=paid_brand-UK_en_Search_Brand_Stripe-2032860449&utm_medium=cpc&utm_source=google&ad_content=355351450259&utm_term=kwd-94834400&utm_matchtype=e&utm_adposition=&utm_device=c&gclid=CjwKCAjwj6SEBhAOEiwAvFRuKO-qHrC01BfkSX83qM

tf.keras.layers.Dropout. (2021, February 23). Retrieved from TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

TradingHours.com. (2020). *New York Stock Exchange.* Retrieved from TradingHours.com: tradinghours.com/markets/nyse

View. (2021, March 12). Retrieved from React Native: https://reactnative.dev/docs/view

Wood, D. (2014). *Basic Interactive Design: Interface Design: An Introduction to Visual Communication in UI Design.* Fairchild Books.

YML. (2020, March 30). *Native VS Hybrid Mobile Apps — Here's How To Choose.* Retrieved from UX Planet: https://uxplanet.org/native-vs-hybrid-mobile-apps-heres-how-to-choose-192ecbf04da8