**Looking for Global Warming in the UK**

CONN09

Academic: Richard Connor

**Sean McMahon**

Matriculation Number - 201312433

# Abstract

Table of Contents

Abstract 2

Introduction 6

1.1 Outcome 6

1.2 Report Structure 6

Background Study of Related Work 7

2.1 Justification for Related Work 7

2.2 Simple Graph Maker 7

2.2.1 Overview and Functionality 7

2.2.2 Strengths 8

2.2.3 Weaknesses 8

2.2.4 Similarities and Differences Compared to AnalysApp 9

2.3 Graph 10

2.3.1 Overview and Functionality 10

2.3.2 Strengths 11

2.3.3 Weaknesses 11

2.3.4 Similarities and Differences Compared to AnalysApp 11

2.4 Stocktradamus 13

2.4.1 Overview and Functionality 13

2.4.2 Strengths 14

2.4.3 Weaknesses 14

2.4.4 Similarities and Differences Compared to AnalysApp 14

2.5 Google Analytics 15

2.5.1 Overview and Functionality 15

2.5.2 Strengths 15

2.5.3 Weaknesses 16

2.5.4 Similarities and Differences Compared to AnalysApp 16

Problem Description and Specification 17

3.1 Problem Description 17

3.2 Requirements Analysis 17

3.3 Feature Detailing and Initial Specification and Requirements 18

3.4 Updated Final Specification and Requirements 18

3.5 Approach to Solving the Problem 18

3.6 Design Methodology 19

System Design 20

4.1 Why a Mobile Application? 20

4.2 Kivy 20

4.3 Python and KV Lang 21

4.4 Integrated Development Environment (IDE) 21

4.5 Buildozer 22

4.6 Python For Android 22

4.6 Kivy-iOS 23

4.7 Kivy App Lifecycle 23

4.7.1 build() 24

4.7.2 on\_start() 24

4.7.3 on\_pause() 24

4.7.4 on\_resume() 24

4.7.5 on\_stop() 24

4.8 UI Design 24

4.9 Good Practices 24

4.10 Application Permissions 24

4.11 Pandas 24

4.12 MatPlotLib 25

4.13 Design Patterns 25

4.14 How the System Design Relates to the Use Cases 25

Detailed Design and Implementation 26

Verification and Validation 27

Results and Evaluation 28

7.1 Overview 28

7.2 User Evaluation 28

7.2.1 Survey Evaluation 28

7.2.2 Client Evaluation (Richard? Section perhaps not required) 28

7.3 Project Evaluation 28

7.3.1 Bugs 28

7.3.2 Project Methodology Success 28

7.4 Final Results 28

Summary and Conclusions 29

8.1 Summary 29

8.2 Future Work 29

8.2.1 Bugs (If any are known to still exist) 29

8.2.2 Expand Data Coverage (Web API for American/worldwide historical data) 29

8.2.3 UI 29

8.3 Conclusion 29

Chapter 1

# Introduction

## Outcome

## 1.2 Report Structure

Chapter 2

# Background Study of Related Work

## 2.1 Justification for Related Work

Before work could commence on the application, a background study of several apps that provide similar functionality to “AnalysApp” had to be carried out. After searching the Google Play Store and iOS App Store, a number of applications were found to provide visualisation, forecasting and analytics functionality.

The applications listed below were used in the background study:

1. Simple Graph Maker (Android)
2. Graph (iOS)
3. Stocktradamus (iOS)
4. Google Analytics (iOS and Android)

## 2.2 Simple Graph Maker

### 2.2.1 Overview and Functionality

“Simple Graph Maker” is an application available to download from the Google Play Store. The app has over 100,000 downloads, and a rating of 3.4/5 from over 700 reviews (as of Dec 2017). The app appears in the top 3 results when searching with the term “Graph”. It is described as a “simple graphing tool” that allows users to “create a graph of a flat design easily”. It a wrapper for the Chart.js tool to provide Android compatibility with the framework. “Simple Graph Maker” is designed to allow users to create a number of different graphs with their own data, and to label the graphs and axes. There is also the ability to save graphs to file and then open them at a later date. It is free to download; however the application uses adverts as a means to make profit.

The application’s home screen contains five buttons, each with different functionality. The first three buttons are to create each graph type: pie chart, line chart and bar chart. The fourth button is to open graphs the user has previously saved and the final option is to view further information about the app itself.

Clicking on the “create a pie chart!” takes the user to a menu that allows them to input the graph title along with the data series’ names and accompanying values. Users can also set custom colours for each data series in the pie chart. Initially, the menu only shows inputs for three data series. In order to add extra series, users can select the “+ add” button at the bottom of the screen to keep adding data series as required.

Once the user is happy with their input data, clicking “CREATE THE CHART!” at the top-right of the menu takes the application to the display screen. This displays the pie chart with the labels of each section stored within them. There is also a legend in the bottom-left corner that shows the percentage of area that each section fills the chart. By clicking the “MENU” button at the top-right of the display, three options appear for the graph: the user can either save the graph within the app, save it to their phone’s saved images or share the image through another app such as Gmail or Dropbox.

The bar and line chart menus both have the same menu for setting graph and data series labels as that of the pie chart option, though this screen doesn’t have inputs for values for these chart types. There is an extra menu between the labelling screen and display screen, and this is where users input the y values/coordinates for each x value. Selecting the “CREATE THE CHART!” button brings the user to the graph display screen again like with the pie chart menus.

Selecting the “Open Saved Graphs” menu brings up a display showing the visualisations in descending order from the oldest saved graph. Finally, the “About” screen shows information about the developer and their other apps, along with the licenses of the open source frameworks used.

### 2.2.2 Strengths

* The UI for “Simple Graph Maker” is simple and easy to understand and grasp. The button’s use rounded rectangles, which gives a softer look to the application by not having any sharp corners in the design.
* User’s can go from the home screen to viewing a graph within three screens. This allows users to see instant results, rather than be lost in endless amounts of menus and inputs.
* The application is robust and coped well with various inputs, and no crashes occurred when creating multiple graphs for each graph type. It is visible from this that the developer has extensively tested the application, and that the third party Javascript libraries have also been tested and validated to the same extent.

### 2.2.3 Weaknesses

* Elements of the UI are too simple and grammar on the buttons and labels is poor also: there is little use of capital letters throughout the app, preventing the application from having a professional feel. This amateur look could put discourage users from downloading the application in favour of one that looks like it has been produced by a full development team.
* The app displays the adverts continuously, even when entering values to input boxes. This can means that almost half of the screen can be taken away from the main screens of the application, and can hinder navigation. As understandable as it is for the need for adverts in mobile applications to generate profit, the placement of some of them can irritate users after a period of time.

### 2.2.4 Similarities and Differences Compared to AnalysApp

“Simple Graph Maker” shares some common functionality with the AnalysApp application. Both aim to make the creation of graphs as quick and easy as possible for the user, and strive to keep the UI simple so that people of all age ranges can use them. The application also allows users to save graphs, which aligns with one of the advanced features of AnalysApp.

The applications differ in the chart types they offer: Simple Graph Maker allows users to produce three types of graph whereas AnalysApp will only produce line graphs. Simple graph maker also allows users to label graphs and axes with custom text, which was not considered for the development of this application, however this feature could be included in the advanced additions. Simple Graph Maker gives multiple options for the saving/sharing of graphs outwith the app itself, but this is beyond the scope of the ten-week development period for AnalysApp however could be implemented if the project was continued after the assessment period.

Figure 2

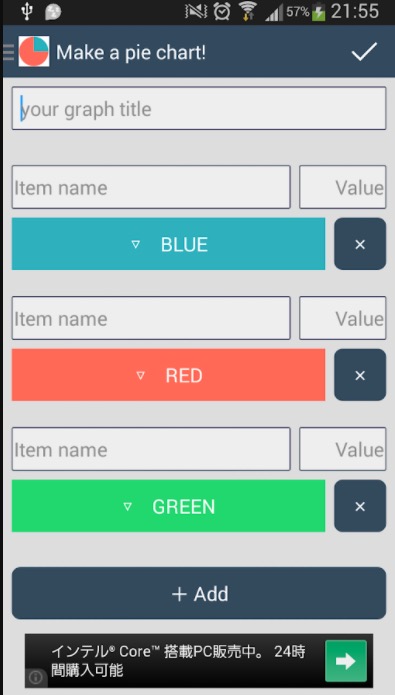
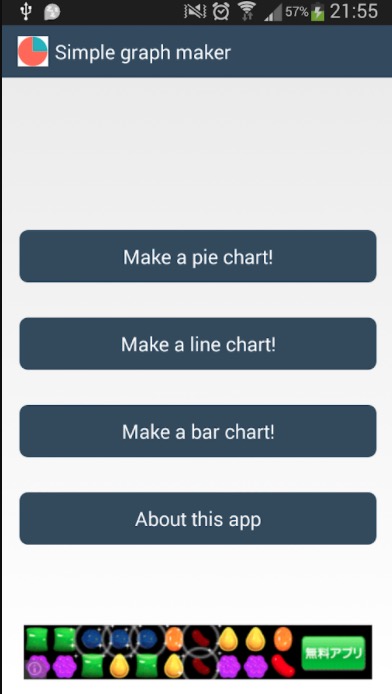
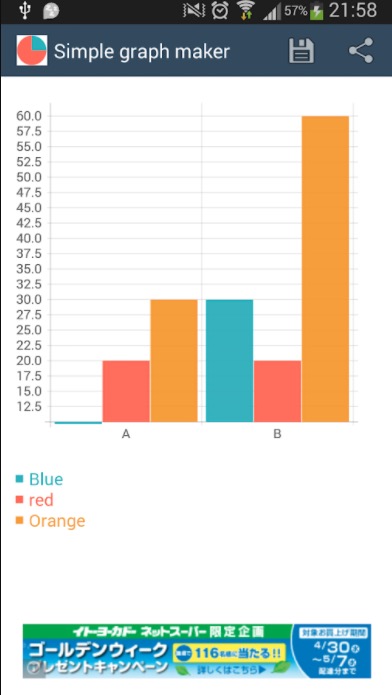


Figure 3

Figure 1

## 2.3 Graph

### 2.3.1 Overview and Functionality

“Graph” is an app available for free on the iOS App Store. It has been available to download for six years and has an average rating of 1.5/5. As Apple does not include download figures in the App Store, it is unknown how many devices the application has been installed on. The application’s description on the App Store explains, “Graph is straightforward” and allows users to create 1D, 2D and 3D graphs from numbers and functions.

After launching, the home screen for the application presents the user with a scrolling list of the various graph types that can be created, along with the extra options that can be unlocked for an in-app purchase of £5.99: for example, error bar graphs and graphs for monitoring health and weight. App settings can also be configured at the bottom of this scrolling menu.

Once the type of graph has been selected, the application transitions to the graph settings screen. This screen presents the multiple options tabs for the graph to be created, and these options are split into three headings: “Textual”, “Graphical” and “General” options.

The first option, “Table Editor”, appears under the “Textual” heading and when selected brings up an input screen, which contains a simple spreadsheet for the values to be entered into. Using the top and bottom slider buttons scroll the spreadsheet horizontally and vertically, respectively. Labels for each ‘x’ value can also be inserted and the options buttons at the top-right of the screen allows the user to import/export data or clear the spreadsheet.

The second option of the graph settings, which appears under the “Graphical” heading, displays the graph and its labels. The Edit button allows the user to select different views of the graph, and from here they can send an email to the developers of the application to suggest new view options for the graphs.

The “General” heading contains the final three options from the graph settings menu. The “Graph Preferences” option brings up the input screen for setting the labels of the graph and its axes, and also for specifying the minimum and maximum values for the domain and range of the graph. The second option under the General heading lets the user switch between the data being presented in a bar, column or pie chart and whether to animate the state of the graph. Finally, the “Manual” option brings up a guide for the user on how to input data and configure the graph.

These options are the same for all the different graph type on the home screen of the Graph app. The differences occur in the data input screen, as some graphs require functions rather than numbers as input.

### 2.3.2 Strengths

* The application offers a comprehensive range of graphs to produce, and the use of spreadsheets to input the data series allows users to easily enter large amounts of data quickly, especially with the import data functionality.
* Having one spreadsheet for each type of graph, i.e. one for 1-Dimensional graphs, one for 2-Dimensional graphs etc. makes it very simple for users to quickly alter the graph that is used to display the values. For example, a user need only enter values once in the 1-Dimensional graphs screen and can then tab between bar, column and pie chart without having to enter any more values.

### 2.3.3 Weaknesses

* There is very little assistance or explanation within the app on how to format the values/functions users input into the application. If a value is entered in an invalid format, the cell will remain empty and no error or warning message displays to explain why the input was incorrect.
* There are many options and graph types for the user to select and this is very confusing when using the application for the first few times. The UI and ads displayed seem very homemade and prevent the application looking professionally made.
* The lack of explanation or help buttons in the app makes it very difficult for those with little understanding of graph functions and 3D models to pick up this app and use it immediately. As one review on the App Store said, “It is clear this app is only good if you are an expert with this type of thing”.
* Although the health and weight monitoring features are useful and bring value to the app, the in-app £5.99 purchase to unlock these contents seems very expensive and off-putting to most users.

### 2.3.4 Similarities and Differences Compared to AnalysApp

This application, like Simple Graph Maker, aims to allow users to produce graphs using their own values and data. The app takes the data and puts it into graph form and, like the initial project specification for AnalysApp, users are unable to customise the colours etc. of any data series in the graphs produced.

Like Simple Graph Maker, the Graph produces different graph types than AnalysApp would offer in the original project specification. An advanced feature of the application, therefore, could be to allow users to produce various graph types for data that could incorporate this feature.

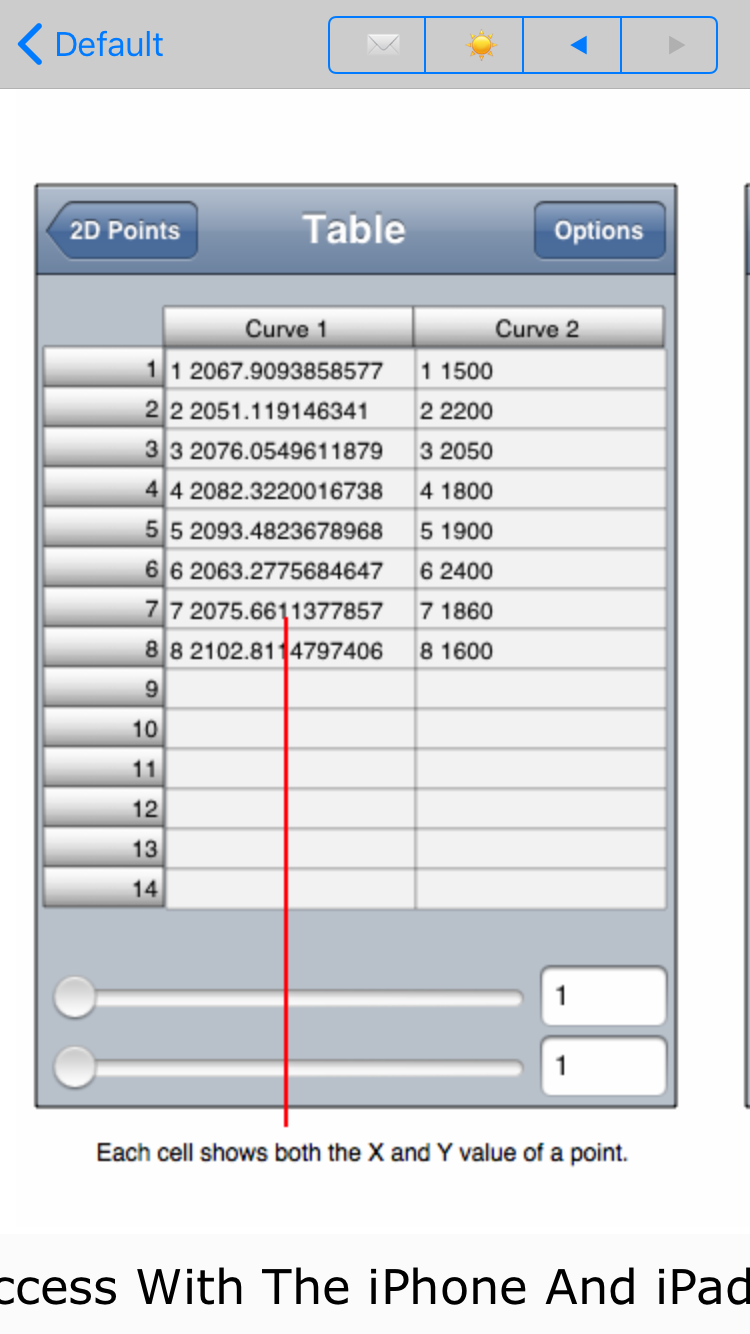
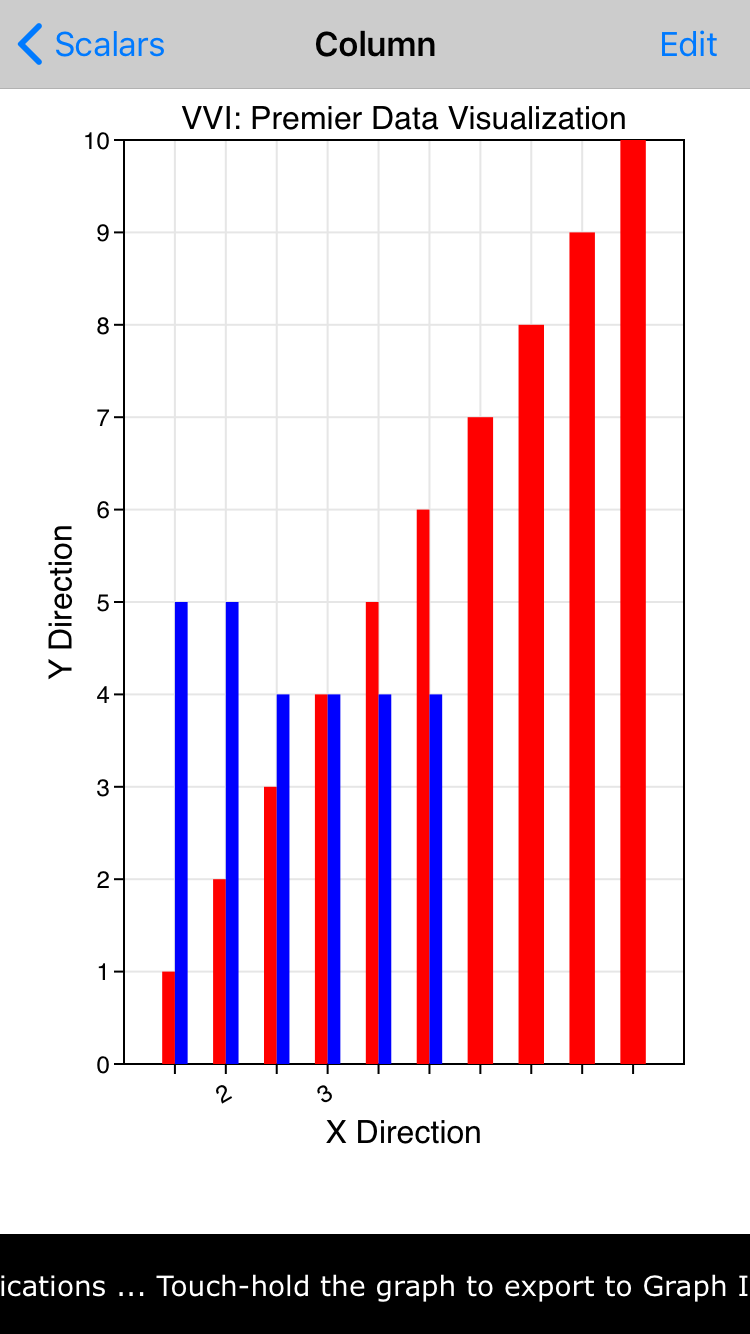
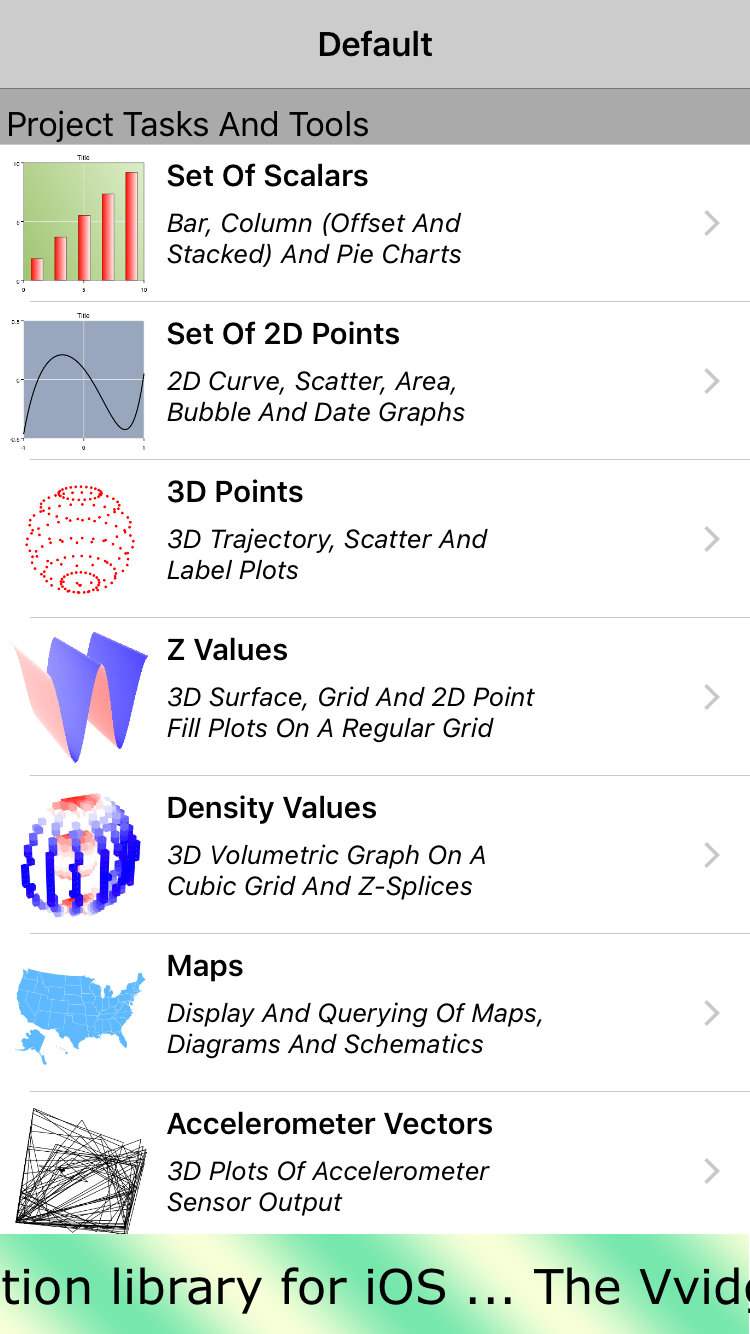


Figure 7

Figure 9

Figure 8

## 2.4 Stocktradamus

### 2.4.1 Overview and Functionality

Stocktradamus is an app available to download on the iOS App Store for a purchase of £1.99. Again, due to Apple not releasing download figures it is unknown how many users have installed the app on their devices. There are no reviews for this application.

The applications purpose is to predict the future value of stock that have at least five years worth of previous stock value data. The user enters a stock ticker name, and selects a time frame for the prediction, and the app will create an Auto Regressive Integrated Moving Averages (ARIMA) model of the existing data and use this to predict the future values. The application’s description on the App Store heavily emphasises that it should be used purely for educational purposes and not as an accurate tool for making investment decisions in stocks.

The home screen for the application is the main input screen and has a simple black background, with labels and inputs using white and cyan coloured text. The first input asks for a stock ticker symbol and the second is a Scrollview for selecting the prediction window. The options in this Scrollview range are 1-9 months or a year. There are three buttons beneath the input areas: two for restoring in-app purchase and view subscriptions terms, and one to start the ARIMA prediction. If the user has entered an invalid stock ticker or a ticker for a stock with less than five years then an error message is displayed to the user.

Clicking the “Done” button with valid inputs will display a temporary loading screen while the model calculates the future values. This screen shows the date of the prediction, the stock ticker to predict and the prediction window selected on the previous screen. Once complete, the app transitions to the results screen. This screen also has the stock ticker and prediction window, and displays the result in the form “Up/Down by x%”. If the model predicts the stock will be higher in value then the background of this label is green, or red for a lower prediction.

If the users wishes to view advanced details on the ARIMA model prediction, they can select the “View forecast details” button at the bottom of the results screen. However, to view these details, an in-app purchase of £1.99 must be made, and this must be paid every time the user wishes to view these advanced figures for the prediction model. Alternatively, users can sign up to a monthly £7.49 subscription that allows them to view unlimited advanced figures for results models.

Once on the advanced figures screen, the user can view the parameters used for the ARIMA model, test statistics for stationarity and how well the model fits the actual values. As well as these test statistics, there is a graph showing the price chart for the predicted stock. This graph displays the historical data for the stock and the ARIMA model’s historical and predicted values. Users have the option to save these forecast details either to the device or via email. If emailing the results, the application uses the iOS Mail app to send a screenshot of the results screen to the given email address.

### 2.4.2 Strengths

* The app is extremely simple to use. The user only has to input two values into the application for it to create a complete prediction model, and can go from the home screen to viewing a stock value prediction in seconds.
* The advanced results give users a greater insight into how the model was created and it is useful to be able to save these results or forward to someone else for viewing.

### 2.4.3 Weaknesses

* The main weakness of this app is its’ pricing. £1.99 is expensive for an app of this size and more than double the common price of £0.79 for iOS apps. Charging users per individual viewing of advanced results is also very off-putting and expensive. If the cost for this feature was a one off fee, then it would perhaps encourage more users make the purchase.
* Once a user is on the basic results screen, they have no way to return to the home screen and create a new prediction. The only way to do this is to close the app completely and restart it.

### 2.4.4 Similarities and Differences Compared to AnalysApp

Stocktradamus is similar to AnalysApp for using an ARIMA model to make its predictions. It also tests a number of different parameter combinations for the ARIMA to see which gives the best accuracy to use as the final model.

However, both applications differ in how they retrieve their data. Stocktradamus sources the stock data via a web server whereas AnalysApp stores the data locally in text files. Storing the data on a server reduces the size of the application by a small fraction, but also means that users cannot predict data when they have no Internet access.

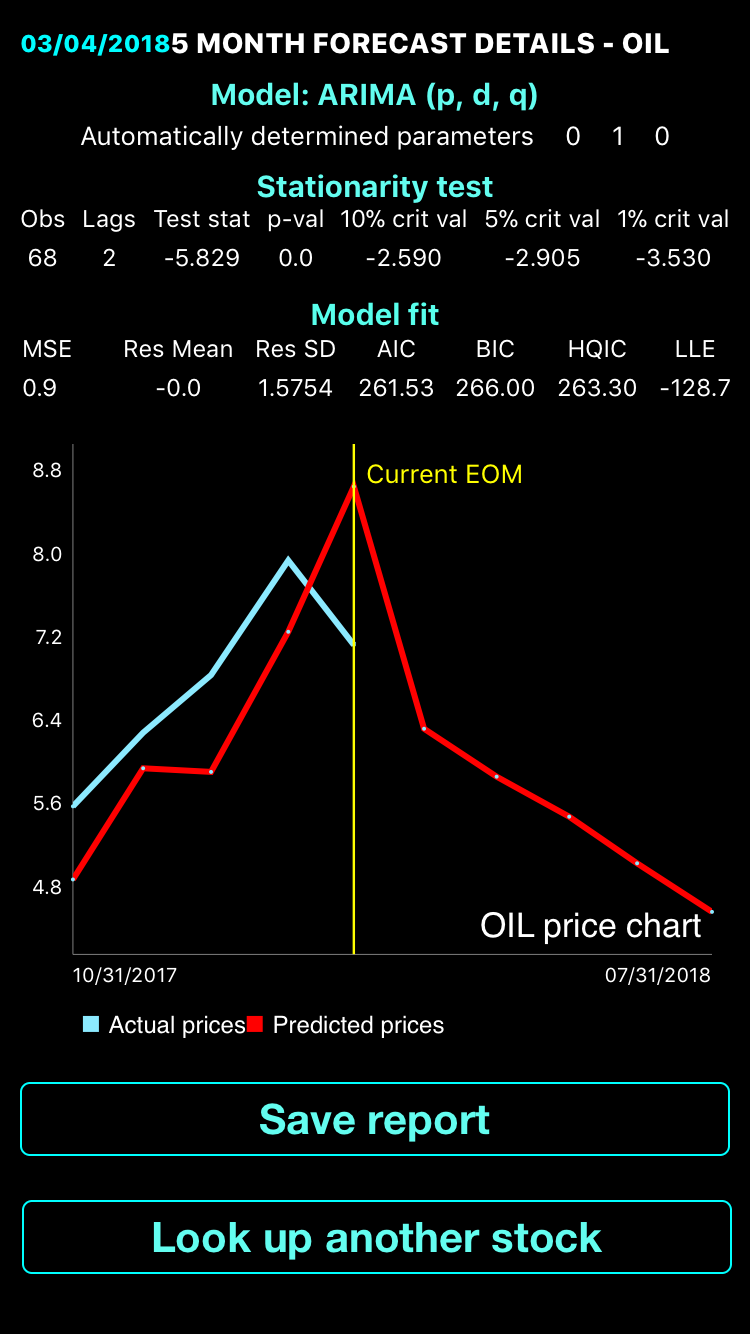
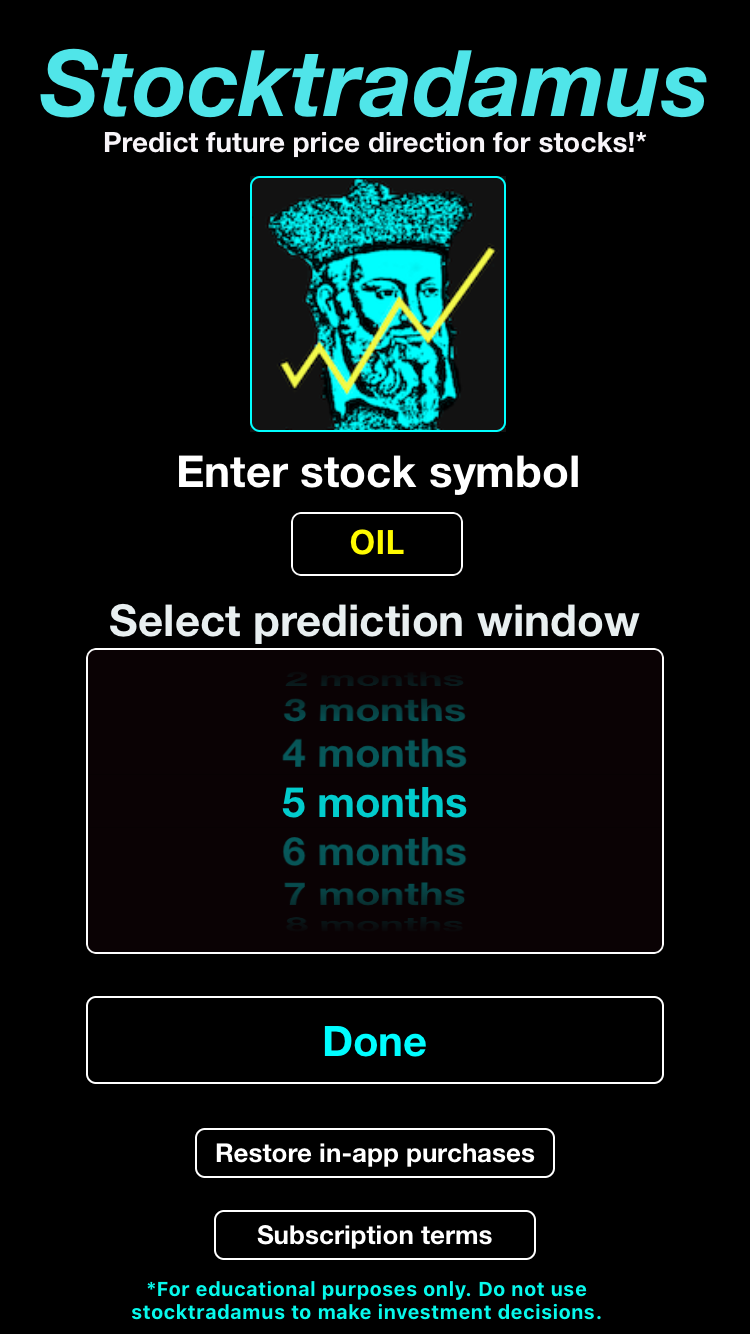


Figure 7

Figure 8

Figure 9

## 2.5 Google Analytics

### 2.5.1 Overview and Functionality

Google Analytics is an app available to download for free from both the App Store and the Google Play Store. Users can also access the Google Analytics tool via its website on their browser (<http://www.google.com/analytics>). The tool aims to give users greater insight and statistics on their website in order to analyse information such as number of users in a given time period or which pages in the site are the most viewed. It has over one million downloads for Android alone, with a rating of 4.5/5. On iOS, the application has an even better rating of 4.8/5 but the 276 reviews from iOS users is miniscule compared to over 60,000 reviews on the Play Store.

In order to use the tool, users need to create an account and install the tracking tool into their website. This is a block of simple Javascript code that is added to the html code of every page in the website. Once this has been added, the analytics information should begin to appear in the app as users visit the website.

When the user logs into the application, the home screen is displayed and users are presented with a weekly view of their analytics. This screen shows a summary of each of the categories that users can pick from the side menu. These graph displays are grouped into categories such as audience overview to see figures for new or returning visitors to the website, behaviour overview to view statistics on how long visitors stay on the website and time stats to view users by the time of day they viewed it. These sections also offer comparisons against the previous time period in order for users to analyse whether their site is performing better or worse than before. Users who have Ecommerce sites can also view these results within the app and examine revenue over time, the percentage of sales by device type, popular products and many more statistics.

The app has functionality for users to create reports based on the analytics data f or use in other analysis methods or perhaps present to other colleagues. The ability to add other users and set their access rights means that a full team can use the tool and team members can have individual roles or abilities assigned to them.

The application also allows users to ask questions by speaking into the devices microphone and the analytics tool will use Google’s speech to text to generate a query on the data. Users can then save these results to view at a later date.

### 2.5.2 Strengths

* It is clear the tool has been created by a team of professional developers, due to the slick, easy-to-use user interface and breadth of analysis on offer to users. The tool is obviously well trusted also due to the number of users who have integrated it with their confidential site data.
* The app allows users to set goals for their sites, for example to increase site traffic by 5%, and then to track these goals in real-time. This gives the ability to track goal progress at any given moment and make decisions on the spot, rather than wait until the goal time has passed before viewing the results.
* Users can switch seamlessly between viewing the mobile and web app, meaning that the analytics data can be viewed at anytime, in any location and on any device.

### 2.5.3 Weaknesses

* It is difficult to find weaknesses in this application, though one could be the sheer volume of data presented to the user. For new users, this could mean a steep learning curve before finding out how to get the most from the tool. Google has extensive documentation and tutorials, though, that could help users to quickly understand how to use the tool.

### 2.5.4 Similarities and Differences Compared to AnalysApp

Google Analytics is similar to AnalysApp by aiming to provide users with a greater insight of their respective datasets. Analytics also allows users to refine the dataset used in the analysis, i.e. by device or site page. This is similar to AnalysApp, which would allow to users refine the dataset by station, data type or date range.

It is obvious that Analytics offers much more functionality than AnalysApp could, but this is expected due to the sheer size and expertise of the developer. The app offers many more graph types than AnalysApp and has a full documentation site for help and Q+A sections. Although this is not possible for AnalysApp, a small help article can be added within the app itself to assist users with the application’s functions.

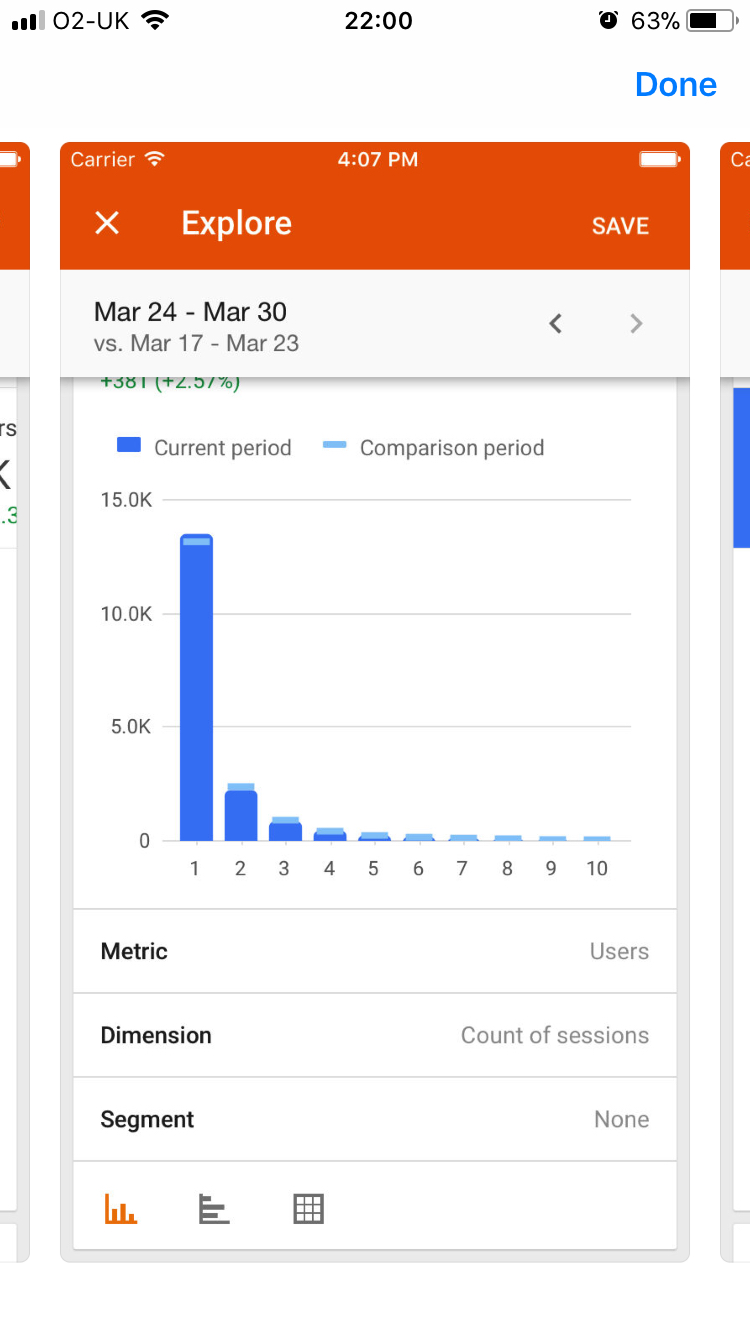
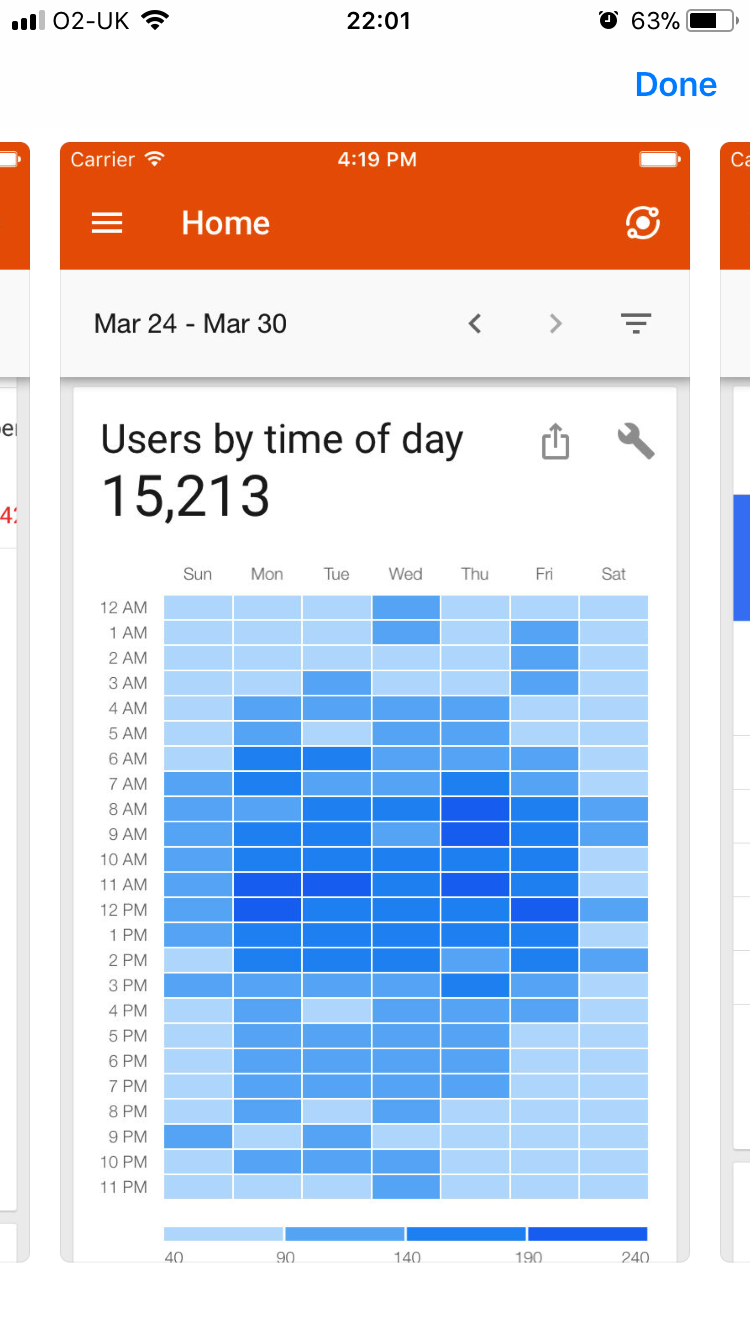
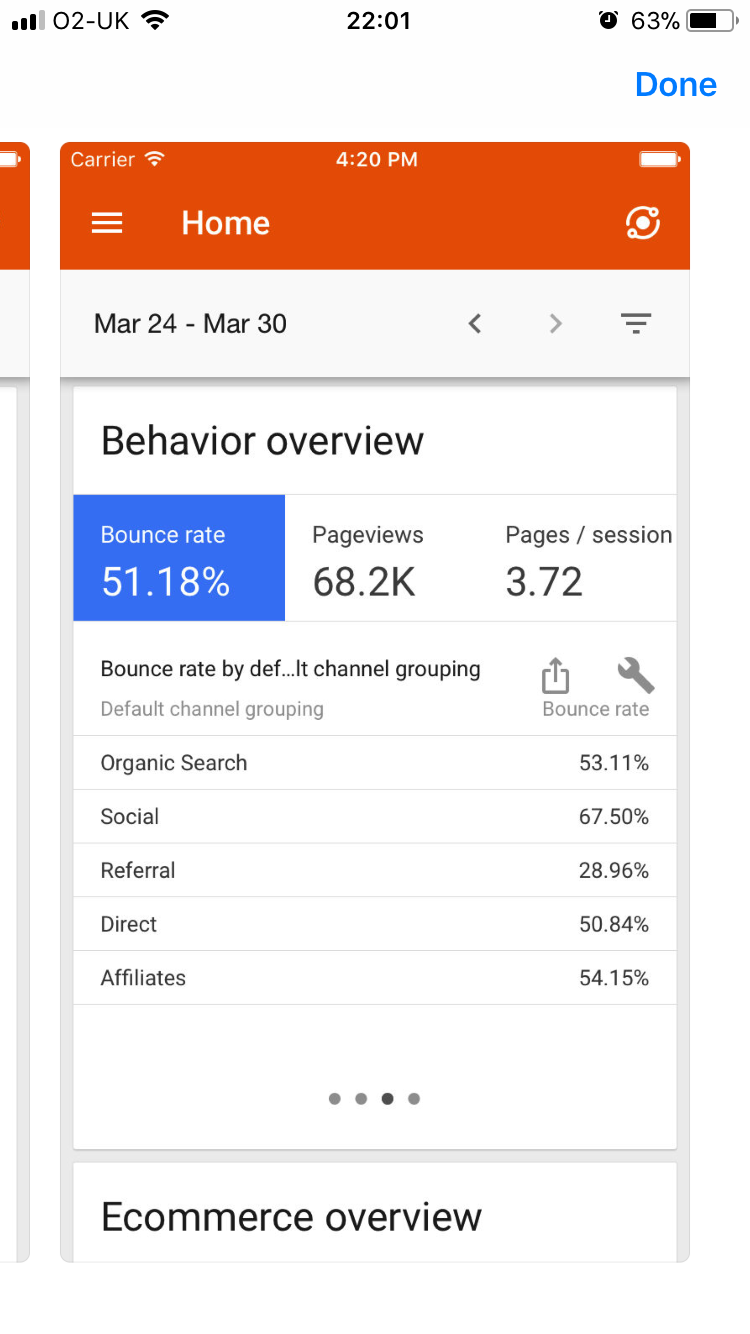


Figure 12

Figure 11

Figure 10

Chapter 3

# Problem Description and Specification

## 3.1 Problem Description

Initially, the project specification was simply to produce a piece of software that could do some interesting form of analysis on the historical weather data available from the Met Office. In order to be able to target more users with the final product, the decision was taken very early on to produce a mobile application. This could then be targeted at students in school/university or members of the general public who have an interest in weather or global warming, to gain a greater understanding of the UK’s weather systems.

“AnalysApp: UK Weather” aims to allow users to view, forecast and analyse historical weather data from thirty seven of the Met Office’s weather stations on a mobile or tablet device. The app tries to provide users with a greater insight into the UK’s weather and climate over the last hundred years and will allow users to look for signs of global warming within the country. To do this, the app has been split into three clear features:

1. Data Visualisation
2. Data Forecasting
3. Data Analytics

Each of these sections had to be researched, implemented and tested separately whilst still using the same data model. Precaution was taken to ensure the finished features brought value to the app and that they were simple and easy to use by the survey groups.

## 3.2 Requirements Analysis

Upon speaking with the project supervisor, further detail was received on the specification for the app. These were the three features bullet pointed above, and it was also said that the results of these features could be displayed through well presented graphs that explain to the user the steps that were taken in any calculations. With these details noted, the features needed to be fleshed out in terms of what they would do.

## 3.3 Feature Detailing and Initial Specification and Requirements

**Primary Features**

* **Data Visualisation**
  + View maximum/minimum/mean temperature, rainfall, sun hours and air frost days.
  + View data by station, country or Britain as a whole
* **Data Forecasting**
  + Predict future data
  + Predict past data
  + Predict future and past data in the same graph
* **Data Analysis**
  + Look for trends in the data for a particular station/data type
  + Any changes in mean value of data types
  + Any changes in standard deviation of data types

These primary features were deemed vital to include in an app that would work and perform well on a mobile device. For this reason, these were the first to be implemented in the app. This was decided on the belief that the ten-week development window would be sufficient to have all of these features in the application.

**Extra Features**

* Save last x amount of graphs created to quickly view again without recalculation
* Colour blind UI option
* View advanced details of forecasting model
* Speech to text input
* Ability to view weather graphs (maximum & minimum temperature plotted against rainfall)
* View multiple data series in a single graph

These extra features are not vital to the finished application, however if time remained at the end of the development process then some of these features would be implemented in order to give a more polished final product.

## 3.4 Updated Final Specification and Requirements

<To be written at once development complete>

## 3.5 Approach to Solving the Problem

In order for the application to be available to as many users as possible, it was decided that it should be cross-platform. For this reason, the Python framework Kivy was used for development, as the developer had experience programming in Python, and the Kivy framework had support for compiling to Android .apk files and iOS Xcode projects. As the developer had not used the Kivy framework before, some initial reading of the Kivy documentation and best practice was carried out to avoid teething problems during initial development.

Each feature was implemented separately, in the order bulleted under Primary Features in section 3.3. The project was developed with the intention of being a fully working mobile application, but not quite ready for deployment to the App Store or Google Play Store due to the developer’s inexperience in creating well-designed, fluid user interfaces (UI). Such prototype-like UIs could hinder the app’s success on the marketplace were it to be launched at the end of the ten-week period. For this reason, a simple prototype UI was used in development with the intention of creating a more appealing one if time permitted.

Research was also carried out to find and understand how to use any Python data modelling and visualisation packages, as these would be vital to the production of the application. The developer attended a Python data science event hosted by JP Morgan and Cambridge Spark at the firm’s Glasgow office, and this contributed greatly towards learning how to manipulate, work on, and visualise the weather data.

Throughout development unit testing was carried out using the Python Unittest module to reduce errors and bugs at run time. As each feature was implemented, it was extensively tested by the user and also by a small group of volunteers who completed surveys after testing each component.

Each feature was developed on top of the existing codebase, as opposed to developing them all in separate applications and combining them at the end of the project. This would also ensure all features worked with each other and the data model as any changes were made.

## 3.6 Design Methodology

It was decided very early on in the project that an Agile approach would be used in the development of the application. Fortnightly sprints would be allocated to each of the three main features, with the remaining two sprints being used for adding advancements and improvements to the existing features. These remain sprints could also be allocated to adding in some of the components listed in the extra features if all primary ones were completed. Each sprint was linked to Trello, a project management application, to allow easy monitoring of sprint progress and to easily copy any incomplete tasks over into the sprints allocated for advancements and improvements. Using this approach would prevent delaying the development of other sprints.

Chapter 4

# System Design

## 4.1 Why a Mobile Application?

The first decision to be made with the project was whether to create a mobile, web or desktop based application. With current statistics predicting there being 2.5 billion smartphone users in 2018, (<https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>) and these users having round the day access to their devices, a pure desktop application was ruled out. This left the choice between developing a web application or a mobile one, but due to the developer’s limited experience in web development, the preferred and final choice was to develop a mobile application.

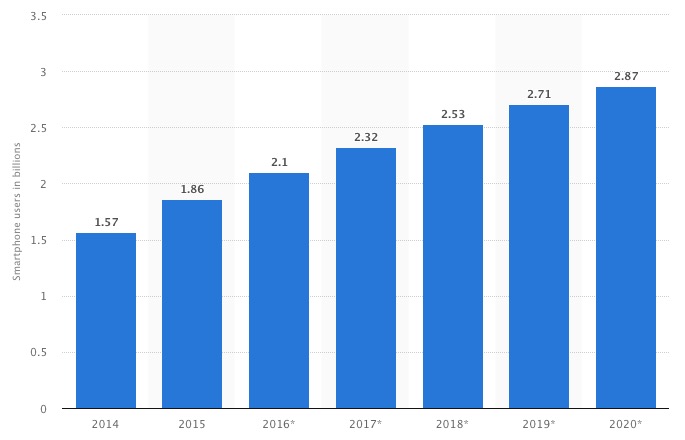


Figure 13 – Predicted Smartphone users from 2014-2020 in Billions

## 4.2 Kivy

The two development platforms considered for mobile were Android and iOS, as they accounted for 99.9% of mobile devices in the first quarter of 2017.

(<https://www.idc.com/promo/smartphone-market-share/os>)

Though Android held 85% of the market, one of the developer’s aims for the project was to target as many users as possible. For this reason, it was decided that the application should be cross platform and from a single code base, so this ruled out using the native Java and Swift languages.

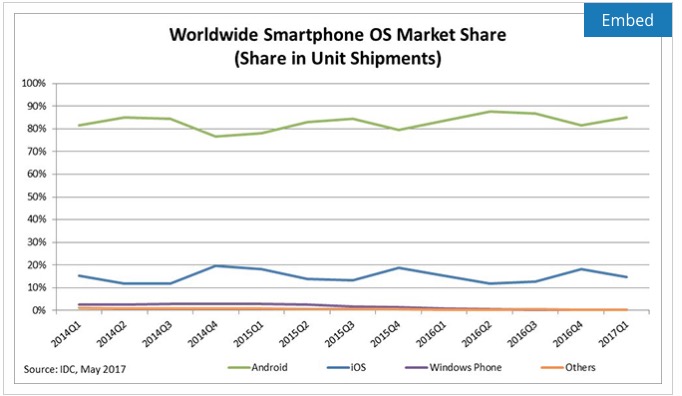
****

Figure 14 and 15 – Mobile Device Type By Share of Market

The next stage of the process was to research the possible cross-platform development tools that could be used for the implementation of the project. A number of tools were found, but the majority of these, such as PhoneGap and Appcelerator, use a Javascript/HTML codebase before compile to the desired platform. Due to the lack of development experience in these languages as mentioned in section 4.1, these tools were not considered.

The next possible development framework considered was the Kivy open source Python library. Kivy is a free to use library that runs on Linux, Windows, OS X, Android, iOS and Raspberry Pi. (https://kivy.org/#home) The framework provides widgets that support multiple touch, mouse and keyboard events and a graphics library using OpenGL ES 2. It also has its own intermediary language, known as KV lang, for designing custom User Interface widgets. This KV lang helps developers to follow a Model-View-Controller design as the KV lang files can be kept separate from the model. As the framework is mostly Python and not just a wrapper around a library written in another language, it makes it very easy for users to extend and customise widgets exactly how they want.

The framework also has many support channels for developers with any questions or issues with it. These include StackOverflow, Google Groups and many more. These channels allow developers to quickly get help from existing users as well as the Kivy core developers who can give expert advice and help.

Once the user has written their Kivy application, there are various tools for compiling the code for different platforms. These tools are discussed in later subsections.

## 4.3 Python and KV Lang

The design of a Kivy application is generally split into two sections. These are the

Functionality/model of the application and the User Interface and are written in Python and KV lang, respectively. The User Interface can be written in Python code in the same classes and the application’s model classes but separating this into KV lang files makes the code base easier to read and maintain. The decision to use a Python based framework was due to the developer having gained a year’s experience using the language while taking part in a fifteen month industrial placement in between their third and fourth year of studies. The developer felt more comfortable programming in a language they had previous experience with and only having to learn the about the modules/packages used, rather than having to learn both in the ten week development period. Also, as Python is a high-level language, the developer would not have to write any platform-specific code.

## 4.4 Integrated Development Environment (IDE)

Though Kivy and Python applications can be written through a command line terminal, the majority of these are written using an Integrated Development Environment (IDE). This is because using an IDE is less time consuming and includes error checking and built in debugging features. Some of the popular IDE’s used for Python development are PyCharm, IDLE and Spyder, however there are also plugins for use with Eclipse and Visual Studio. Following recommendations on the Kivy support channels, the PyDev plugin for Eclipse was the chosen IDE for this project. The benefits of this included not having to learn the layout of a new IDE as the developer had previously used Eclipse for development.

## 4.5 Buildozer

In order for the finished application to run on a mobile device, the code must be compiled to the respective file type. For Android this is an .apk file and for iOS the code must be converted to an Xcode project. There are instructions in the Kivy documentation on how to install tools and do this manually through the command line, but there are tools that greatly simplify this for developers. One of these tools is Buildozer, a Python module available through the Pip installer.

Buildozer works by creating a single application specifications file, named “buildozer.spec”. In this file, users specify the application title, icon, module dependencies etc. and then running the Buildozer command from the terminal will run the appropriate tool based on the platform to compile to. Using Buildozer makes compilation to each desired platform much simpler for the developer as they do not have to understand how each individual build tool works. Users simply have to enter the required values into the buildozer.spec file and leave Buildozer to fill the remaining options with default values before running the necessary commands.

## 4.6 Python For Android

Python For Android (P4A) is an open source tool originally developed for converting Kivy apps into standalone Android apks but now supports porting many Python packages to the file type. The tool runs from the command line terminal and includes a number of features:

* Create a debug version of the application for testing on a personal device.
* Create a signed application for uploading to the Google Play Store.
* Deploy the compiled apk to a connected physical device or emulator.
* Print the device or emulator’s log output to the terminal to aid in debugging the compiled application.

The tool takes the developers code and a list of dependencies to be included in the compiled apk. The script will then create a virtual environment for the project and install the required Pip modules to it. For Pip modules that are purely written in Python, the porting process is simple, but for modules that include low-level C code the task is more complex. These interoperable modules require a ‘Recipe’ – a collection of scripts that take care of compiling any compiled components of the Pip module. These must also be compiled for the correct architecture. A handful of Pip modules have already had recipes created for them by the Kivy community, though many modules that involve a large amount of compiled code still await recipes to be written.

## 4.6 Kivy-iOS

Kivy for iOS works the same way as Python for Android, but has some more dependencies in order to work. It can only run on the OS X operating system and requires Xcode and Cython to be installed on the host machine, and it includes the same features as Python for Android. However, instead of producing a file ready for installation on a device or simulator, the tool takes the Kivy project code and produces an Xcode project ready to be opened by the application. The app can then be run signed through the Xcode application.

## 4.7 Kivy App Lifecycle

Kivy applications have a number of different states during their lifecycle. Users opening, closing and returning to the application causes this state to change. As these state changes occur, various callback methods are invoked in order for the application to react accordingly to the user’s actions. The developer can use these callback methods to perform any necessary loading/saving of files or any other actions that may have to be carried out. There are five callback methods in the Kivy application lifecycle:

* build()
* on\_start()
* on\_pause()
* on\_resume()
* on\_stop()

### 4.7.1 build()

The build callback method initialises the application and is only called once in the entire lifecycle of the program. This is used for retrieving the widget tree and root for the application. AnalysApp will define all user interface widgets in the KV lang files and these will be loaded into the widget tree as the build method is called.

### 4.7.2 on\_start()

The on\_start method is fired after the build callback but before the application has started running. This method can be used to perform any initial actions the application requires such as checking the platform it is running on in order to make any necessary configurations.

### 4.7.3 on\_pause()

Android and iOS devices mainly call the on\_pause callback method. It is not possible to control when this method is called, as it only occurs when the user closes the application without fully stopping it. This method should return true if the application can go into Pause mode, and false otherwise. If false is returned, then the application will be stopped. This method is also used for resizing.

### 4.7.4 on\_resume()

Figure 16 – Kivy Application Lifecycle

The on\_resume method is called when the application is about to exit Pause mode. This should be used to render any UI elements or OpenGL graphics that may have been freed by the application/operating system during Pause mode.

### 4.7.5 on\_stop()

The on\_stop callback is called when the application is about to be closed. This should be used to save any files and safely stop any resources that the application was using before they are freed.

## 4.8 UI Design

The end product for AnalysApp at the completion of the ten-week development period was to be a full feature demo application. This was so that the majority of development could be focused on feature functionality rather than on aiming to create the slickest user interface possible. Even so, it was still important to the developer to create a clear, concise user interface that would be simple for users to understand and make use of.

The plan for the development of the application’s interface was to create a prototype interface for use in initial development and testing. Then, once the main functionality was completed for the app, a cleaner, more professional looking interface would be developed. This second interface would bring the application closer to be ready for deployment to the App Store or Google Play store, and research on its design would be carried out by comparing existing web forms online for good and bad aspects.

## 4.9 Good Practices

While developing the AnalysApp application, a focus was placed on following and implementing the many practices and conventions of Python and Kivy developers. These conventions are recommended in order to provide cleaner, more readable code that is easier to maintain and to improve the overall quality of the code. The conventions followed throughout the development process are as follows:

### 4.9.1 Python Enhancement Proposals (PEP)

Python Enhancement Proposals are collection of documents that propose major new features, collect the Python community’s opinion on an issue and detail the design decisions that have been made for the Python language. There are three types of PEP documents:

1. Standard Track PEP – describes a new Python feature or implementation
2. Informational PEP – details Python design issues or suggests general guidelines to the development community. However it does not propose new features.
3. Process PEP – these are similar to Standard Track PEPs but differ in that they can apply to other topics than just the Python language. Usually these PEP documents are enforced and users do not have the choice to ignore them unlike with other PEP document types.

### 4.9.2 Kivy API

The Kivy API documents all widgets and components that are available to use in the framework. Within each entry are details on how to initialise the widget, both in Python code and in KV language, as well as the various callback events that methods can be bound to. Along with these API details, there are example code snippets that show the best way to use the widget.

## 4.10 Application Permissions

As the application does not require access to any device hardware like a camera, microphone or web browser, no app permissions will be required from the user after installation in order for the full functionality to be available.

## 4.11 Pandas

Before any of the main features of the application can be implemented, the data from each of the thirty-seven text files from the Met Office must be loaded in. While carrying out the initial research for the development of the application, the obvious choice for the in-memory storage of the data was Pandas.

### 4.11.1 What is Pandas?

Pandas is a Python library for data manipulation and analysis. As explained on the Pandas website:

***‘****Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. ‘*

There are two main data structures in Pandas: Series and Dataframes. Series are one-dimensional structures and Dataframes are two-dimensional. Dataframes are a collection of Series.

### 4.11.2 Reasons for using Pandas

Pandas can read data in from a text file and automatically create the appropriate data structure with column names and row indexes. The data structures can also be grouped by particular columns values and have operations performed on these grouped values such as max, min, mean or standard deviation. Pandas also handles missing data and provide many approaches to filling these rows with values. As well as this, there are simple methods for performing queries on the dataframe or series. All of these features make the package an ideal choice for use in the AnalysApp application.

## 4.12 MatPlotLib

The main feature of the application is the ability to produce graphs to present the data, forecasts and analysis results to the user. It was important to find a package that could provide this functionality early on in the research phase of the project as this tool would be used throughout the application’s codebase.

### 4.12.1 what is Matplotlib

Matplotlib is a Python two-dimensional plotting library for producing a great variety of graphs. The package describes itself as trying to “make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code”. The package allows users full customisation of the graphs that they produce, from setting axis labels and series titles to setting the location that the graph legend will be placed.

### 4.12.2 Reasons for using Matplotlib

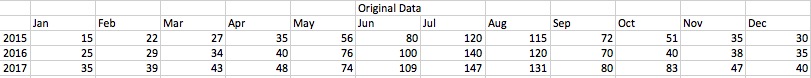
Matplotlib provides the ability to save graphs that are produced. As well as this, the package is well integrated with Pandas, which makes this an obvious choice for use in the application. As it is so popular within the Python community, there is a vast amount of documentation on how to use the package also, which would help greatly throughout the development cycle.

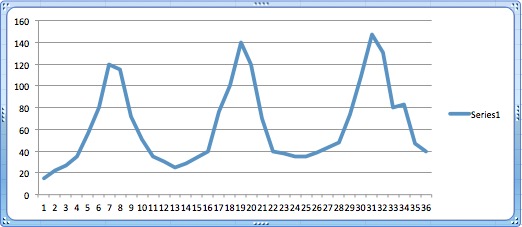
## 4.13 StatsModels ARIMA

StatsModels is a Python package for creating statistical models of datasets and statistical data exploration. It includes a number of methods for representing statistical data, but the one that suits the needs of the AnalysApp project is the ARIMA model. Before looking at the ARIMA model, it helps to understand the general approach to time series data forecasting.

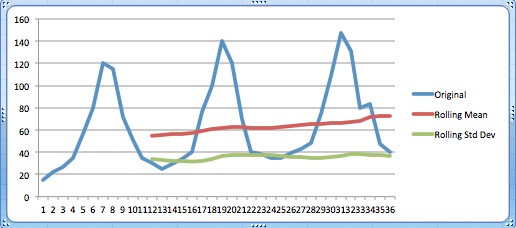
### 4.13.1 Forecasting Time Series Data

In order to forecast time series data, a number of techniques must be applied. The first thing that must be done is to ensure that the data is stationary. Stationary data means that things like mean and variance in the data remain constant over time. For example, given a dataset of sales of sunglasses, it helps to first view the data and plot the series in a graph.





We can then plot the rolling mean and the rolling standard deviation alongside the original data to check for stationarity. Rolling means that for point *x* in the data, we perform the given action on this point and the previous *y* values. For example, the rolling mean at index 3 if *y=2* is calculated as:



As can be seen from the graph, there is both seasonality and a positive trend in the data. The rolling standard deviation increase is minimal so this does not impact much, but the rolling mean gradient does. Before any forecasting can be performed, these aspects must be removed from the data set in order to make it stationary. It is important to note also that both the rolling mean and rolling standard deviation do not have values for the first eleven points in the data set. This is because we set a lag of twelve so we do not start get values until we reach index twelve.

In order to remove trend from a data series, we can apply transformation. Transformation is the technique of applying a function that penalises higher values. There are many functions that can do this for us, such as log, square root, cube root etc. For this example we will take the log of each value

### 4.13.2 What is ARIMA?

AutoRegressive Integrated Moving Average (ARIMA) models, in theory, are the most general type of model for forecasting a time series of data.

## 4.14 Design Patterns

State, Oberserver = kivy properties

## 4.15 How the System Design Relates to the Use Cases

Chapter 5

# Detailed Design and Implementation

Chapter 6

# Verification and Validation

Chapter 7

# Results and Evaluation

## 7.1 Overview

## 7.2 User Evaluation

### 7.2.1 Survey Evaluation

### 7.2.2 Client Evaluation (Richard? Section perhaps not required)

## 7.3 Project Evaluation

### 7.3.1 Bugs

### 7.3.2 Project Methodology Success

## 7.4 Final Results

Chapter 8

# Summary and Conclusions

## 8.1 Summary

## 8.2 Future Work

### 8.2.1 Bugs (If any are known to still exist)

### 8.2.2 Expand Data Coverage (Web API for American/worldwide historical data)

### 8.2.3 UI

## 8.3 Conclusion

Appendix

References

Trello

Kivy

Met Office data

Stackoverflow posts

Github issue posts google group posts