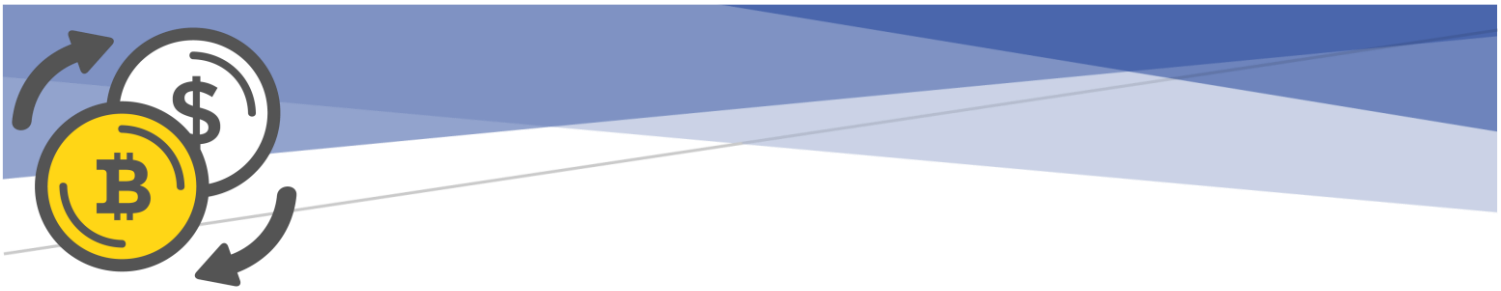


BRAIKOUT – FUNCTIONAL SPECIFICATION

CA400

Project Title:

Braikout – A Machine Learning Cryptocurrency and Forex Trading Platform

- Student Name: Ross Franey
- Student Number: 14302851
- Date of Final Submission: 24/11/2017

Ross Franey
Ross.franey3@mail.dcu.ie

Contents

1. Introduction

Overview

Braikout is a cryptocurrency and forex trading platform which makes use of machine learning and automated candlestick chart analysis, designed with the intent of making the job of traders easier, less time consuming, and more profitable.

Idea Generation:
The idea behind the development of this application, as is the case with many innovative systems, stems from an unsatisfied need in the current sphere of currency trading.
There is currently no example of a platform through which a user can:

1) Trade both cryptocurrencies as well as traditional currencies.
2) Gauge sentiment of multiple markets they are trading in real time (or at all for that matter!)
3) Make use of machine learning algorithms to predict price action. While this is something that large corporate investors make use of, the source code, of course, is kept secret and as such, the average trader is at an immediate disadvantage.
4) Automatically generate chart analysis for the markets they are watching, saving hours of manual plotting and never missing a technical breakout (either up or down) of a currency.

Trading is known to be not only particularly time consuming, but also require a lot of space both in terms of screen real-estate, and mental capacity in order to ensure all potentially profitable markets are being monitored to a sufficient degree. This application aims to reduce the time, physical space, and mental strain required, all while improving the user's trading potential.

Functionality:

- Cryptocurrency and Forex Trading
- Machine Learning Price Prediction
- Real-Time Sentiment Analysis
- Automatic Chart Analysis
- User Alert System
- Data Visualisation of markets, performance, and evaluation of the project

Interaction with other Systems:
This application will make use of the Bitstamp API for trading cryptocurrencies, as well as the Oanda API for trading Forex.

1.1 Business Context

The nature of this application makes it an ideal solution both for personal traders who wish to negate any disadvantage they may otherwise suffer for having a regular job in conjunction with their trading endeavours, or for large asset management companies such as Fidelity investments, many of whom have for a long time been

involved with Forex trading, but have just recently become involved with cryptocurrency trading.

This application will make it easier for such companies to manage large trade decisions across multiple markets automatically. Decisions will consider current sentiment, previous price, technical analysis, and have a prediction from a machine learning module reaffirming them.

Using this application, the company will also be afforded the ability to gauge the health of each individual portfolio using the data visualisation functionality, to immediately conceptualise the data rather than having to keep a log of performance metrics themselves, which would be an unnecessary waste, both of employee time, and of storage space.

1.2 Glossary

- **Long** – This refers to a trading position which was taken by buying a certain currency.
- **Short** – This refers to a trading position which was taken by borrowing money from an exchange to buy a currency, only to immediately sell it. It is essentially betting on the decrease of a currency.
- **Margin Position** – A position taken using borrowed money from an exchange.
- **RSI** – Relative Strength Index, a metric used to gauge whether a currency is oversold or overbought.
- **MACD** – "Moving Average Convergence Divergence is a trend-following momentum indicator which shows the relationship between the two moving averages of prices". - Investopedia
- **Candlestick Chart** – A candlestick chart an alternative representation to a line graph in trading. Candles appear on defined time intervals, with the highest point/wick of the candle being the highest price of that time, and the lowest point being the cheapest.
- **Technical Breakout** – A technical breakout refers to a currency that has been in a recognised pattern breaking that pattern in one direction, which typically is indicative of imminent price action in that direction.
- **Technical Analysis (TA)** – The act of analysing charts in order to gather metrics which help to make a more informed trade.
- **Stop-Loss** – This refers to a trading position being opened with a defined point at which the trader wishes to automatically close the trade if it goes poorly.
- **SALT** – In cryptography, this is random data which is used as an input to a one-way function that hashes a password.

1.3 Technologies Used

Below is a list of technologies new to the developer which will be used in the creation of this application, highlighting some of the technological learning challenges associated with the project.

- Kibana – Data visualisation
- Elasticsearch – Lightweight database solution which can be used to index Kibana data visualisation pages.
- Python – The backend of the web application
- Django – Front end of the web application
- Scrapy – Web scraper used in sentiment analysis
- NLTK (Natural Language Toolkit) – Sentiment analysis classifier
- SQL – Internal database
- Apache web server – Web app Host
- NumPy – Machine learning library
- SciPy - Machine learning library
- Plotly – Graph visualisation library

2. General Description

2.1 System Functions

Cryptocurrency and Forex Trading:
Users will have the ability to trade real markets, and have access to all the options that would be expected, such as buying, selling, and opening a margin long or short position.

Machine Learning Price Action Algorithm:
This algorithm will look at various indicators such as a dataset of the past price of a market, the current sentiment of the market, the technical indicators of the market such as trading volume, relative strength index (RSI), moving average convergence divergence (MACD) and Fibonacci levels, in order to make a prediction on the daily price closing either above or below the previous day.

Real Time Sentiment Analysis:
As well as being an integral part of the machine learning algorithm itself, a standalone visualisation of market sentiment will be available to users. This will include sentiment from major news sites such as Bloomberg and Coindesk, as well as twitter sentiment.

Automated Chart Analysis:
Time-series analysis will be carried out on charts in order to give the user key information about the market they are trading. This will include the identification of support and resistance areas for the price, triangle/wedge patterns that may be forming, and a notification of markets the user is subscribed to becoming oversold or overbought, or indeed breaking out of one of the patterns mentioned which would indicate imminent price action depending on what side of the pattern the price broke through.

Data Visualisation:
A data visualisation section will also be present on the site. This section will link a user to their “Data Dashboard”. Here, the user will find relevant information regarding the currencies they are interested in, including charts and graphs representing everything from their own portfolios, to the actual evaluation of the price prediction algorithm itself. This will be displayed in the form of a chart representing the expected/predicted chart, alongside the actual realised price action that occurred.

2.2 User Characteristics and Objectives

The typical user of this system, be it an employee at an investment company or simply a hobbyist, would be expected to have a basic understanding of how trading works in terms of the various pairs that are being traded against, and how to read candlestick charts.

As well as this, the ideal user would have at least a basic understanding of the various terms used in the trading scene such as long, short and margin trading, however, to satisfy the design goal of accessibility and continual learning, information modals containing an explanation of such terms will exist on the application in order to ensure new users can quickly learn all the terms used in the trading world.

From the perspective of the user, the goal of this system is essentially to do everything that they could do themselves in order to be a better trader, and to be frank, make money, in a faster, more informed, and more accurate manner.

Specifically, however, this goal can be broken down into **four predominant objectives**:

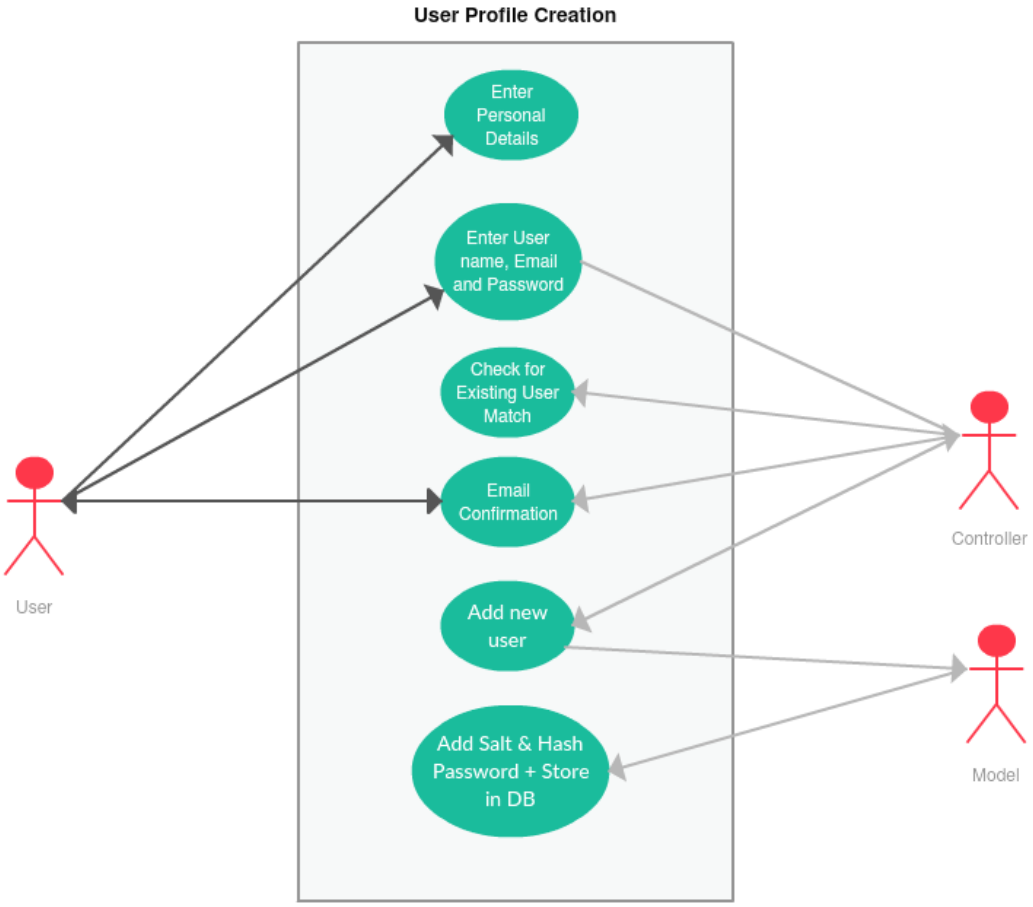
1) Perform technical analysis (TA) to a standard which is *at least equal* to that which they would perform manually, saving hours upon hours of charting, and allowing for quick decision making rather than having to spend time performing manual TA before entering a trade.
2) Alert the user when there is a potential trading opportunity. These alerts will ensure that a user knows each time any market is oversold for example (which always leads to a short-term price increase), overbought, or when a pattern that has been identified (such as a wedge or triangle) is broken by the price action.
3) Predict, to an accuracy that starts reasonably greater than random, and is ever improving, the price direction for each market. This will involve determining whether a market will close at a higher or lower price than the previous day's closing price.
4) Evaluation of performance is another objective from the perspective of the user, which is key to any trader's success. This application will automatically provide the user with this summarization, without any additional work required from him/her, in the form of data visualisation.

2.3 Operational Scenarios

There are three main operational scenarios that the user will undertake while using this application, which are represented as use cases in this section. Modules are listed as secondary actors, which corresponds to the concept of a model in the MVC design pattern that has been used to design this application (explained in section 4 of this document).

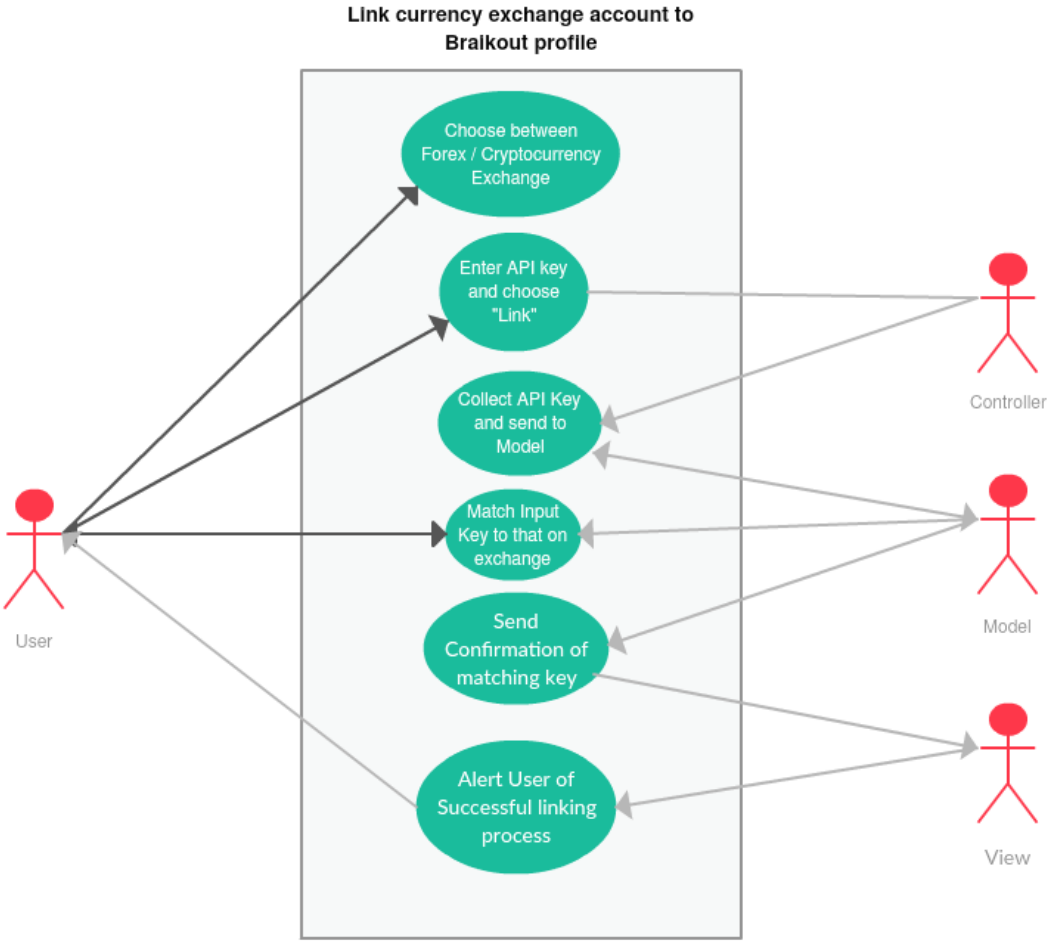
Use Case 1 – User Profile Creation

Use Case 1	User Objective: Braikout User Profile Creation	
Goal in Context	User wishes to create a profile for the application, so that they may link API keys from 3rd party exchanges to this app, and make use of Braikout's features.	
Scope & Level	Account & Settings Module (Client & Server)	
Preconditions	User has a registered email address which is not already associated with a profile	
Success End Condition	User has successfully created a profile with username ID and password, and can link API keys to connect this profile with their exchange trading account(s).	
Failed End Condition	A profile has failed to be established	
Primary Actors **Secondary**	User Controller, Model	
Trigger	User chooses "Create Profile"	
DESCRIPTION	**Step**	**Action**
	1	User enters personal details
	2	User chooses username, email and password
Extended	3	Controller checks with the Model to ensure the username or email does not already exist
	4	Controller sends email to the user's specified address
Extended	5	User Confirms account creation process via email link
	6	Controller passes the new profile information to the model
	7	Model adds SALT, hashes password and stores in database.
EXTENSIONS	**Step**	**Branching Condition**
	3.1	Username already exists
	5.1	User does not receive email
EXTENSIONS	**Step**	**Branching Action**
	3.1	View asks user to choose a different username or email
	5.1	Confirmation link times out and user must contact support


User Profile Creation
Enter Personal Details
Enter User name, Email and Password
Check for Existing User Match
Email Confirmation
Add new user
Add Salt & Hash Password + Store in DB
User
Controller
Model

Use Case 2 – Link currency exchange account to Braikout profile

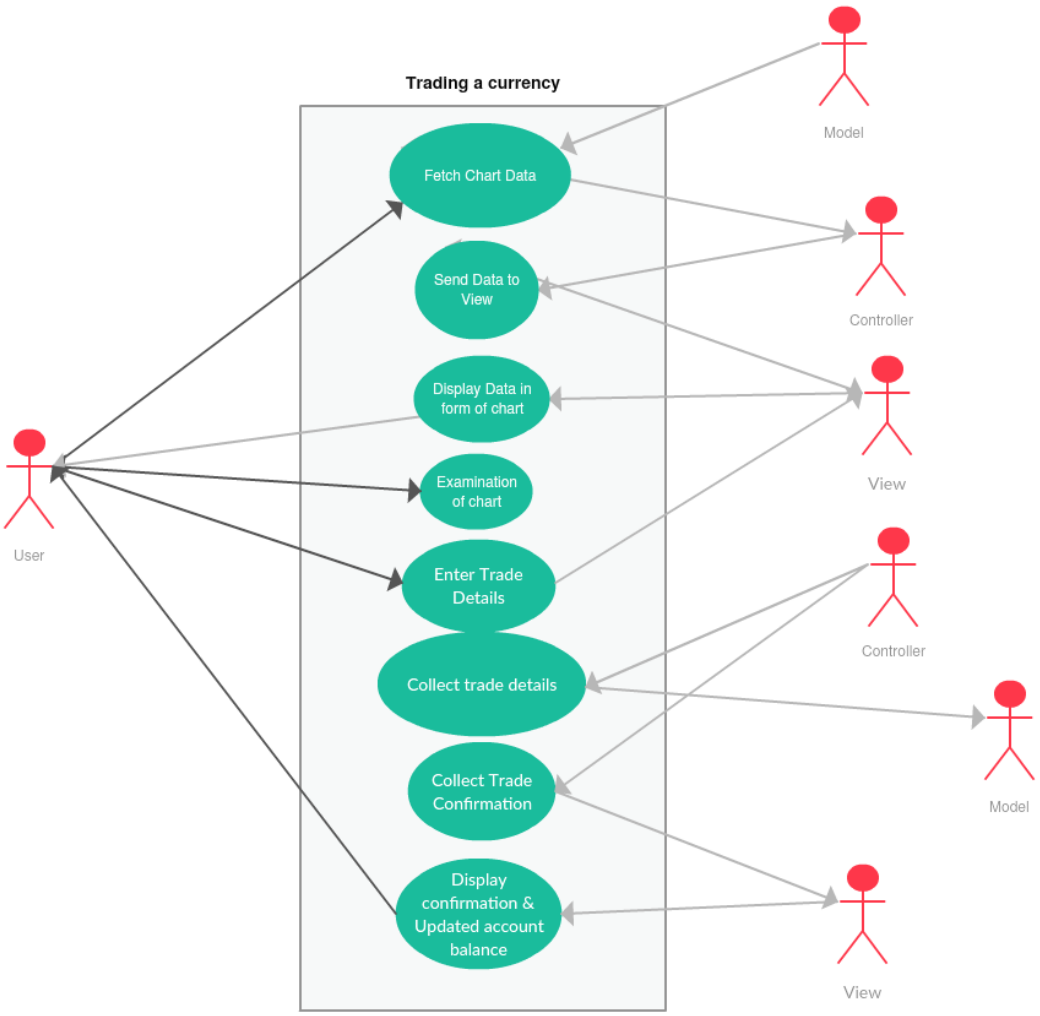
Use Case 2	User Objective: Link currency exchange account to Braikout profile	
Goal in Context	User wishes to access a currency exchanges services via Braikout so that they may trade from within this application, making use of its additional features which aren't found on a standard exchange	
Scope & Level	Account & Settings Module (Client & Server)	
Preconditions	User has already a registered account with either a forex or cryptocurrency exchange, and has generated a unique API key on the respective exchange User has an established Braikout profile and is logged in	
Success End Condition	User has successfully linked a valid API key from either a cryptocurrency or forex exchange to their Braikout profile	
Failed End Condition	An API key is incorrect or the user is unable to establish a link	
Primary Actors **Secondary**	User Controller, View, Model	
Trigger	User opens "Link API" tab	
DESCRIPTION	**Step**	**Action**
	1	User chooses between forex or cryptocurrency exchange
	2	User Enters API key and clicks "link"
	3	Controller collects API key from the view and sends it to the model
Extended	4	Model matches input key to key stored on exchange
	5	Controller sends a confirmation of the success to the view
	6	View alerts user of the successful linking of API key
EXTENSIONS	**Step**	**Branching Condition**
	4.1	API key is not a match
EXTENSIONS	**Step**	**Branching Action**
	4.1	Controller sends information of failed match to the view
	4.2	View Alerts user of the failed match and asks user to try again


Link currency exchange account to Braikout profile
Choose between Forex / Cryptocurrency Exchange
Enter API key and choose "Link"
Collect API Key and send to Model
Match Input Key to that on exchange
Send Confirmation of matching key
Alert User of Successful linking process
User
Controller
Model
View

Use Case 3 – Trading a Currency

Use Case 3	User Objective: Trading a currency	
Goal in Context	User wishes to trade a currency pair using this application	
Scope & Level	Trading Module (Client & Server)	
Preconditions	User has already a registered Braikout profile with successfully linked API keys and is logged in.	
Success End Condition	User carries out a trade (either buy/sell/long/short) of their chosen currency	
Failed End Condition	User is unable to make the trade which they intend	
Primary Actors **Secondary**	User Controller, View, Model	
Trigger	User chooses a currency pair from the dropdown menu	
DESCRIPTION	**Step**	**Action**
	1	Controller fetches chart data from the model and sends to view
	2	The View displays the data in the form of a chart to the user
Extended	3	User examines chart, making use of the various tools this application offers to offer to do so
	4	User enters the amount in which they want to trade
	5	Controller collects this data from view and passes it to model
Extended	6	Controller collects confirmation of trade and updated account balance information and passes it to View
	7	View displays confirmation of trade to the user along with the updated account balance
EXTENSIONS	**Step**	**Branching Condition**
	6.1	Model cannot confirm trade
EXTENSIONS	**Step**	**Branching Action**
	3.1	User examines the machine learning price prediction result
	3.2	User runs a sentiment analysis check for the market in which they are interested
	3.3	User examines the technical analysis indicators generated from the chart analysis module
	6.1	Controller collects reason for invalid trade from model

	6.2	Controller sends reason to View to be displayed to user



2.4 Constraints

1. One constraint which was instrumental in driving the direction of the design of this application was the need to have a license to afford the service of buying and selling currencies. To solve this problem, the application makes use of two very different APIs from a cryptocurrency and forex exchange respectively. Despite this solution, one constraint necessarily persists, being the funding for each exchange. Since technically two different user accounts will be used to connect to each API, it is not possible for the user to have a shared wallet with which they can fund the purchase of both crypto and forex currencies. As there is no resolution to this constraint, the application will be designed such that the user will see a “forex” balance and “cryptocurrency” balance separately.
2. The prediction of price direction is also constrained by the availability of forex market data. Unfortunately, historical intraday data is not available past a certain point, however, daily “close price” data is. This means that there is no possibility of predicting the price using the machine learning algorithm on any

time frame other than the daily. Although the same constraint does not exist on cryptocurrency data (likely because of its relatively short history), the daily time frame will be the only time frame predicted here also, to maintain consistency. A large goal of the design of this application is to make the integration of both forex and cryptocurrencies seamless. Considering this, the aforementioned approach has been deemed most suitable.

3. Another constraint on the application is the lack of continuity in cryptocurrency prices across different exchanges/regions. This is because unlike forex trading, cryptocurrency prices are different depending on the exchange they are being traded on. To account for this, the price prediction will simply use price history of the exchange being utilised in this application, Bitstamp. Market average historical data was considered and rejected, as this would skew the data with outlier situations (such as the unrest in Venezuela, for example, which lead to an almost doubling of bitcoin price).
4. Finally, a constraint has been placed on the sentiment analysis module of this application, specifically the portion of sentiment which is sourced from Twitter. This is because the Twitter API restricts the amount of data a user can extract in short time frames. The solution to this constraint will simply be to adopt the search-technology approach of increasing the "precision at recall" score, by targeting known important twitter hashtags and accounts, essentially being more specific about the information that will be useful to the algorithm.

3. Functional Requirements

3.1. Trading

Description: The most fundamental aspect of the application is the buying and selling of currency. A user can buy, sell, long or short currency pairs BTC/USD, LTC/USD, ETH/USD, EUR/USD, GBP/USD, and CNY/USD.

Criticality: This is an essential feature of the application, given the fact that the inspiration for its development was the notion of a single location to manage all the complexities of a trader's job. If the trading itself was omitted, the application would not truly reflect this inspiration.

Technical Issues: A large design challenge associated with allowing trading of both traditional and cryptocurrencies, is the fact that currently no exchange offers the trading of both, meaning two separate APIs must be learned and utilized in this application. Essentially, though the code may look very different, both Oanda's Forex trading API and Bitstamp's cryptocurrency API will be utilized and presented to the user as if they were one back-end.
Another technical issue is the aforementioned fact that two APIs means that a user's funding will be restricted to each individual account. IE, the user will have a forex wallet and a cryptocurrency wallet, between which funding is not immediately transferable.

Dependencies: Despite being critical to the application from a conceptual perspective, the function of trading is in fact, entirely separate from the other elements of the application which focus more on helping make good trading decisions and visualising trading performance. From a technical perspective, however, the remaining functions of the application could be packaged into a standalone application which is focused on price prediction, chart analysis, sentiment

analysis and data visualisation, which would still be a useful application. It would not, however, fulfil the design goal for the system, which is to be the **only** piece of software a trader will ever need to perform maximally.

3.2. Sentiment Analysis

Description: This trading platform will offer users real time sentiment analysis from different sources such as social media and news websites. An example may be a user wishing to check what the current sentiment on bitcoin is amongst twitter users, in order to ensure that there isn't a large amount of negativity in the marketplace before they make a trade.

Criticality: This feature of the application, while not necessarily critical as a standalone function (it is nonetheless a very useful one), is a critical element of the machine learning module which will be utilised by the system to predict daily price action.

Technical Issues: One issue with sentiment analysis, particularly when analysing tweets, will be the format in which people articulate their sentiment. It will be a difficult task taking individual tweets, and processing them to remove any "incorrect" English, or strange characters such as "@" symbols or Emoticons, so that they are in a readable format to be analysed by the classifier on the sentiment analysis code. Another issue is the various terms specific to trading, which would indicate the sentiment of a market, that a typical library would not consider/recognise. This includes extremely positive terms such as "bull" (optimist), or negative terms such as "bear" (pessimist). To combat this, Braikout will undertake the difficult task of training its own classifier, using learning files manually constructed with currency trading in mind.

Dependencies: The sentiment analysis tool is "standalone" as it were, and not dependent on any other requirement. It is, however, a dependency of the machine learning tool boasted by this application.

3.3. Machine Learning

Description: The application will utilize a regression solving Support Vector Machine (SVM) to predict whether the daily price of a currency will close above or below the previous day. The SVM has been chosen for its ability to create a linear, binary classifier given two sets of data belonging to two different categories (in our case, days preceding a price increase and those preceding a price decrease.
Various factors that will be considered as part of this machine learning are the previous price, using datasets from Kaggle and other sources such as the respective exchanges.
Criticality: This function is critical to the vision for this application. There are many factors that go into making a successful trade, and indeed a successful trader. Intra-day charting is certainly one of them, but larger time frames such as the daily time frame cannot be ignored, and is always indicative of overall trends in the market. This feature will allow the user to confidently enter trades, knowing that even if the smaller time frames don't convey some uncertainty, there is a "back-up" or reassurance coming from an ever-learning algorithm which is predicting price action on a larger time frame.

Technical Issues: A technical issue associated with the machine learning module will be normalising results for cryptocurrency and forex trading. It may well be the case that different factors influence the price action of each market to a different degree, and as such, the algorithm will need to be general enough to cover the spectrums of both. One solution is to simply use two different algorithms and predict the markets independently. This may be a wise path to traverse, as it would also allow for the collection and visualisation of data regarding factors that influence each market separately, which would be of great interest to both myself, and any potential user of this platform.
Another issue considered is how often the algorithm should run and make a new prediction. Of course, once daily may not be optimal as factors such as intraday indicators or market sentiment may change drastically throughout the day, rendering an initial prediction obsolete. The ideal scenario would be to have a real-time continuous solution, however a design goal for this application is maximum efficiency, so that it can be run on as large a range of machines as possible. To accommodate this, a good solution would be to run the algorithm periodically every 4 hours, and allow the option for it to be ran manually from within the app also, ensuring minimal downtime, while also maintaining relative efficiency.

Dependencies:
The machine learning tool will be dependent on the various other indication tools; sentiment analysis, and chart analysis.

3.4. Chart Analysis

Description: Chart analysis comprises many hours of a trader's day. It could be defined as the studying of candlestick charts in order to determine key pieces of information, crucial to making an informed trade which has high as possible probability of resulting in profit. This is a very complex and wide-reaching field, so this application will focus on the fundamentals; RSI levels, MACD crossings, trading volume, support / resistance levels, and trend lines.

Criticality: This aspect of the application is considered critical for the same reason as the machine learning module. The application's design goal is centred around giving traders as much information as possible, without requiring the user to exert any effort other than making the trade. Chart analysis is the primary functionality through which this application will save the trader large amounts of time, and as such is a functional necessity.

Technical Issues:
A technical issue associated with the chart analysis module will be identifying trend lines in the charts. As the charts are simply a dataset of numbers, I will need to perform time series analysis in order to identify areas which are forming as either support or resistance for any given currency.

Dependencies: The chart analysis tool is not dependent on any other aspect of the application. It is, however, a dependency for the Alert functionality.

3.5. Alerts

Description: The user will be alerted when coins they are subscribed to exhibits useful information that they may need in order to handle a trade optimally. This will

include oversold/overbought conditions (using RSI), breaking out of a trend line, or particularly positive/negative market sentiment.

Criticality: This is not a critical feature of the application, but a useful one that aptly ties together the various components of the system and allows users to utilise its benefits even when not at their computers.

Technical Issues: One issue with alerts will be ensuring that they are only given to the user for markets to which they are subscribed.
Furthermore, a second will be automatically running the various tests for when a market indicator is relevant, even when the user is not running the application. This will likely have to be at a fixed time increment, such as every 15 mins.

Dependencies: This functionality will depend on the various modules of the application that provide the indicative information, such as the technical analysis module, sentiment analyser, and machine learning algorithm.

3.6. Data Visualisation

Description: The data visualisation section will take the user to a Kibana dashboard which will convey, in the form of various charts and graphs, key information such as the performance of the machine learning module over time, a visualisation of sentiment analysis, and the user's current portfolio (percentage of various currencies).

Criticality: This feature is not critical, but rather acts as a convenient way for users to evaluate both their own performance, and that of the application itself. It also provides a means of proving/evaluating the effectiveness of the system as a project.

Technical Issues: One difficulty will be ensuring that the Kibana dashboard can extract data sufficiently from the SQLite database. Typically, elasticsearch is the easiest back end to integrate with this type of dashboard, so attempting to achieve an equally functional result with SQL will be challenging.
Another difficulty associated with a data visualisation page will be developing the metrics by which the periodic evaluation of the various components of the system will be carried out. For example, figuring out how the system will evaluate the effectiveness or "weighting" of sentiment analysis, in comparison to the effectiveness of chart analysis, in the overall price prediction.

Dependencies: This data visualisation functionality will depend on the same data as the alert function, it will however store this data rather than simply alert the user of its presence.

4. System Architecture

Braikout is a web application and as such, its design is based on the **Model-View-Controller design pattern (MVC).** This design pattern is, with seemingly little contention from within the community, typically the most preferable design a web application can adopt.

4.1. What is MVC?

As the name suggests, there are three main components which comprise this design pattern:

Model: This is a representation of data. Not to be confused with the data itself, the model can interact with, while ignoring the intricacies of, the underlying database. The model may also act as an abstraction layer between the data and the database, meaning it can be general to any of the various types of databases in question.

View: The view is the presentation layer for the model. Essentially, it is what the user will see in their browser while using the web application. Importantly, the view also boasts an interface which can accept user input.

Controller: Following the trend of rather aptly named components, the controller controls the flow of information between the model and the view (the "front-end" and the "back-end"). The controller uses programmed logic to interact with the database, and to collect information through the view.

4.2. MVC as a Client/Server Architecture in Django

Braikout has been designed in Python Django, and as such follows the MVC design pattern by structure. In Django, however, it is often referred to as the "Model-Template-View" or "MTV" development pattern.

The difference is entirely superficial, with the model translating directly to a MVC model, the "template" translating to a MVC view, and the View translating to a MVC controller.

MVC	**MTV (Django)**	**Client/Server**
Model	Model	Server
View	Template	Client
Controller	View	Server

Considering its well-established precedence, this document will describe the design of Braikout in terms of the traditional MVC pattern, despite the technical disparity between the names of components.

4.3. MVC in Braikout

Structure of the application:
Braikout is an application which, in its design, is broken up into multiple different modules working together to form the larger system.
These modules are largely based on the functional requirements.

1) The Trading module

2) Sentiment module
3) Machine Learning module
4) Chart Analysis module
5) Data Visualisation module

Each of these modules contain their own implementation of the MVC design pattern, meaning there are five separate models, views, and controllers in the system.

Trading Module:

The trading module represents the “back-bone” of the system. It is here that a user will find the homepage or “dashboard” of the system. From here, the user can also carry out currency trading on the various currency pairs they are interested in.

Model:
The trading model will store information about the market in real time. As the model can act as an abstraction on multiple different databases, the SQLLite database of Braikout will pull data from online databases, and update itself with real-time prices for coins. This model will also contain the user’s wallet balances.

View:
The view of the trading module will be the dashboard/home page itself, as well as the various currency pair pages, which will allow the user to see a candlestick chart of the chosen market, as well as offer the ability to trade the market. The view’s interface for input data will take in the specified trades of the user, which will include details such as amount, price and stop loss specification for the trade.

Controller:
The controller for the trading module will pass the trade specification from the view to the model, which will in turn interact with the exchange database and carry out the trade. Further, the controller will then pull an updated version of user data from the model, which will contain new user balance, and send it to the view.

Sentiment Module

This module will gather sentiment analysis both from twitter, and from various news sources such as Bloomberg, compile the result, and determine whether a market is currently “bullish” or “bearish”

Model:
The model for this module will represent the data that is fetched from social media and news websites, which will be used by internal program logic to determine the market sentiment.

View:
The view for this module will be a web page in which a user may visualise the sentiment of all markets available. This view can be implemented in other modules of the system such as the trading application, which will display market sentiment in terms of percentages for that currency.

Controller:
The controller of the sentiment module will ensure that the sentiment result is passed from the model to the page on which it can be displayed.

Machine Learning Module

The machine learning module is designed to make predictions on the daily time frame, regarding the direction of future price action.

Model:
The machine learning model will represent a dataset of previous prices on the various currencies. As well as this, the storage of other factors of the machine learning module such as sentiment and chart analysis will be represented by this model.

View:
The front-end of the machine learning module will be a web page which will allow the user to choose between the various currencies they are interested in, displaying the prediction for the direction of the price.

Controller:
The controller will ensure the result of the machine learning module's prediction, either up or down, as well as the corresponding currency in question, will be passed from the model to the view.

Data Visualisation Module:

The data visualisation model will allow the user to get an immediate conceptualisation of the performance of the system and their current state as a trader.

Model:
The model for this module represents the data which will be produced by the previously discussed sentiment, machine learning, and chart analysis modules.

View:
The view for this module will be a Kibana data visualisation front-end.

Controller:
The controller for this module will transport the visualised data from the model to the view, not before performing programmed logic to transform the data into the respective graphs and charts.

Chart analysis Module

The chart analysis module will highlight important areas of price, such as resistance, support, volume levels, and other technical indicators.

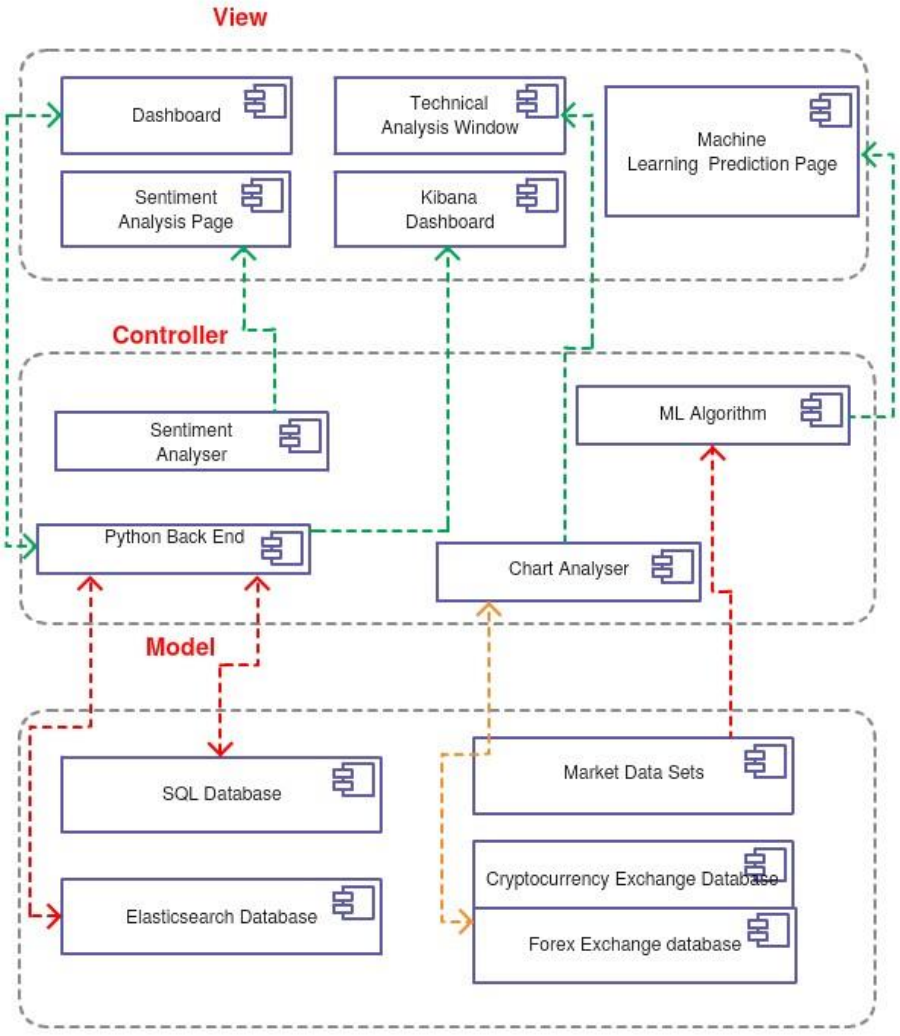
Model:
The chart analysis model will represent data coming from exchanges, which is stored in the local SQL database if it is relevant to the indicator. This will include the opening and closing price of candlesticks, as well as RSI and volume levels. The chart analysis module will also represent the storage of user-subscribed currencies, for which an alert/notification will be received.

View:
The view of the chart analysis module will be implemented in the trading module. This will provide the front end with a window detailing the current RSI, resistance, support, and volume levels. The view will also allow for the input of user data relating to specific currencies they want to subscribe to for alerts. The view also represents the possibilities of the various means by which a user can be alerted, such as email, desktop and SMS notification.

Controller:
The controller for the chart analysis module will handle the flow of data from the model, not just to the view window, but also to the preferred method of user alert.

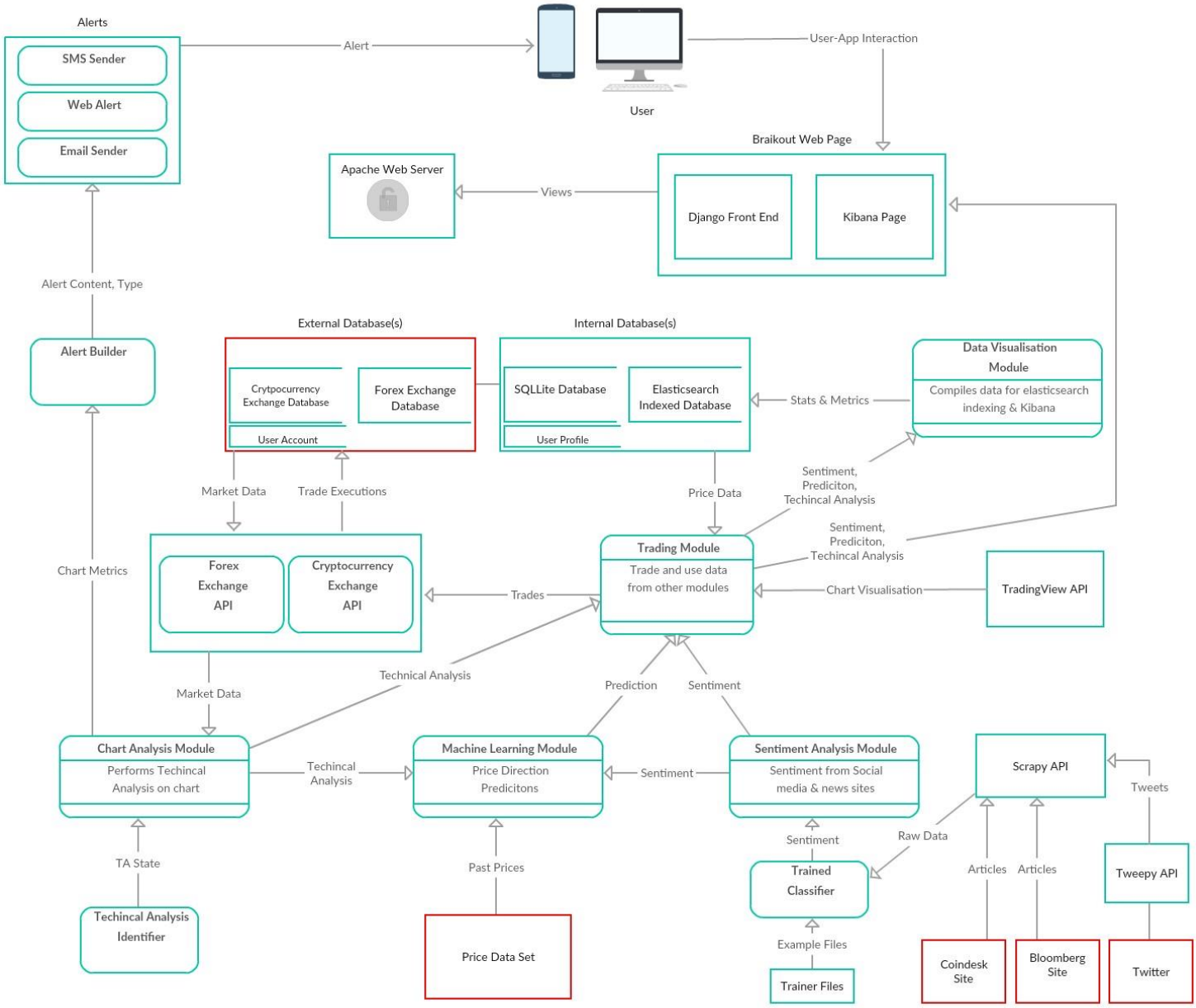
4.4. Component Diagram

Rather than drawing individual component diagrams for each of the above modules, a data flow diagram will delve into further detail in section 5.
The following is a component diagram for the wider system, meaning rather than focusing on the MVC design of each module, this diagram represents the resulting MVC design of the system as a whole.



5. High-Level Design

5.1. Braikout Data Flow Diagram (DFD)



Seen above is an example of a traditional “Gane and Sarson (GS)” data flow diagram. The only addition to the GS format are boxes which have a red border. In this DFD, these boxes represent external entities which the system relies upon, but exerts no control over.

The diagram represents how the various components of the application communicate and transfer data between each other, so that they can fulfil the goal of the module they are associated with.

To ensure consistency throughout this document, the DFD has been carefully designed to tie together the various perspectives of the application given in the different sections of the document.

As such, each “module” in the diagram corresponds to the modules listed in section 4.3 in the Architectural Overview above, as well as the processes and entities on which these modules rely to carry out their functionality in the overall application.

A brief description of this functionality is provided in the beneath of the module names above, and again, corresponds with the functionality from section 2.1 of this document.

6. Preliminary Schedule

In-keeping with the consistency of this documents various perspectives on the project, each major task below will correspond to the development of the various modules seen in section 2.1 and section 5. The ordering of the tasks is such that each task that is dependent on another is scheduled for after its dependency, allowing the tasks to be carried out in a linear fashion.

To complete this project, Agile task management in the form of a Scrum framework will be utilised. As such, task completion will typically be split up into sprints which are one week in length. For many tasks defined below, sprint lengths are longer depending on the perceived difficulty of the task.

After each sprint, the degree to which the task has been completed will be evaluated, recorded, and the next task will be planned in detail.

In order to keep track of the development cycle, Atlassian software’s JIRA will be used, which is an interactive online project and software tracking toolkit, often utilised in industry.

6.1. Tasks

1. Proposal:
   The idea generation and proposal of this project involves developing a project proposal form, in which a brief description of the system and the technology that may be used to achieve its development would be outlined.
   a. Interdependencies: None.

2. User Interface Development:
   The development of the user interface is the next task to undertake. There are various sub-tasks involved under this heading.
   a. Familiarization with Django and Python:
      As no previous experience of either python or Django has been acquired, the development of the front-end of this application will first require gaining an understanding of these technologies.
   b. The next stage of development will be to integrate a back end into the user interface so that it may be populated with content. This will

involve creating a SQL database, as well as indexing this database with the various pieces of module data necessary.

c. Implement APIs: Implementing the various APIs involved in the development of this project is the next phase, and will involve learning the programmatic practices of not one, but two libraries (for both cryptocurrency and forex exchanges). The libraries of these exchanges must be studied to a degree which is sufficient for their integration into Braikout.

3. Sentiment Analysis:
   The third major task in the development of this application is building the sentiment analysis module. This will involve working with the scrapy API in python, with which I have no experience. As well as this, sentiment analysis is not something that has been covered in the course up to now, so research into the various methods of classifying sentiment will also be necessary, as well as study into the processes involved in building and training a classifier.

4. Chart Analysis:
   Chart analysis will involve developing mathematical algorithms which will take current market data, and identify key metrics. In order to achieve this, various internal mathematical processes will need to be developed in order to calculate things such as RSI and MACD. This is very different to anything previously encountered in the course and as such will require extensive research.

5. Machine Learning:
   The next major task involved in the development of this application is the development the machine learning module. Machine learning is an enormously extensive field of research, and as such this task will involve a large degree of prior research into the various types of machine learning, determining which is most appropriate for use in this application, and subsequently the various methods through which the selected variant of machine learning can be implemented.

6. Data visualisation:
   The task of data visualisation will involve indexing the internal database to an Elasticsearch (ES) database, a task with which the developer has no experience. The indexed ES database will then need to be established as a data source for a Kibana page. Research is necessary both into the establishment of an ES database (particularly the conversion from SQL to ES), as well as the process of creating a Kibana page.

7. Testing:
   The next and final process involved in the development of this project is testing. The testing phase will involve a variety of categories of tests in order to ensure a full coverage as possible. Testing will include:
   Code:
   - Continuous Ad-Hoc testing
   - Back-End testing
   - Unit Testing
   - Integration Testing (Testing External entities from section 5)
   - Boundary value testing (passwords, account details, etc.)

   System-Wide:

- Usability Testing
- Browser Compatibility testing
- User Acceptance Testing (UAT) – For which ethical approval will be acquired

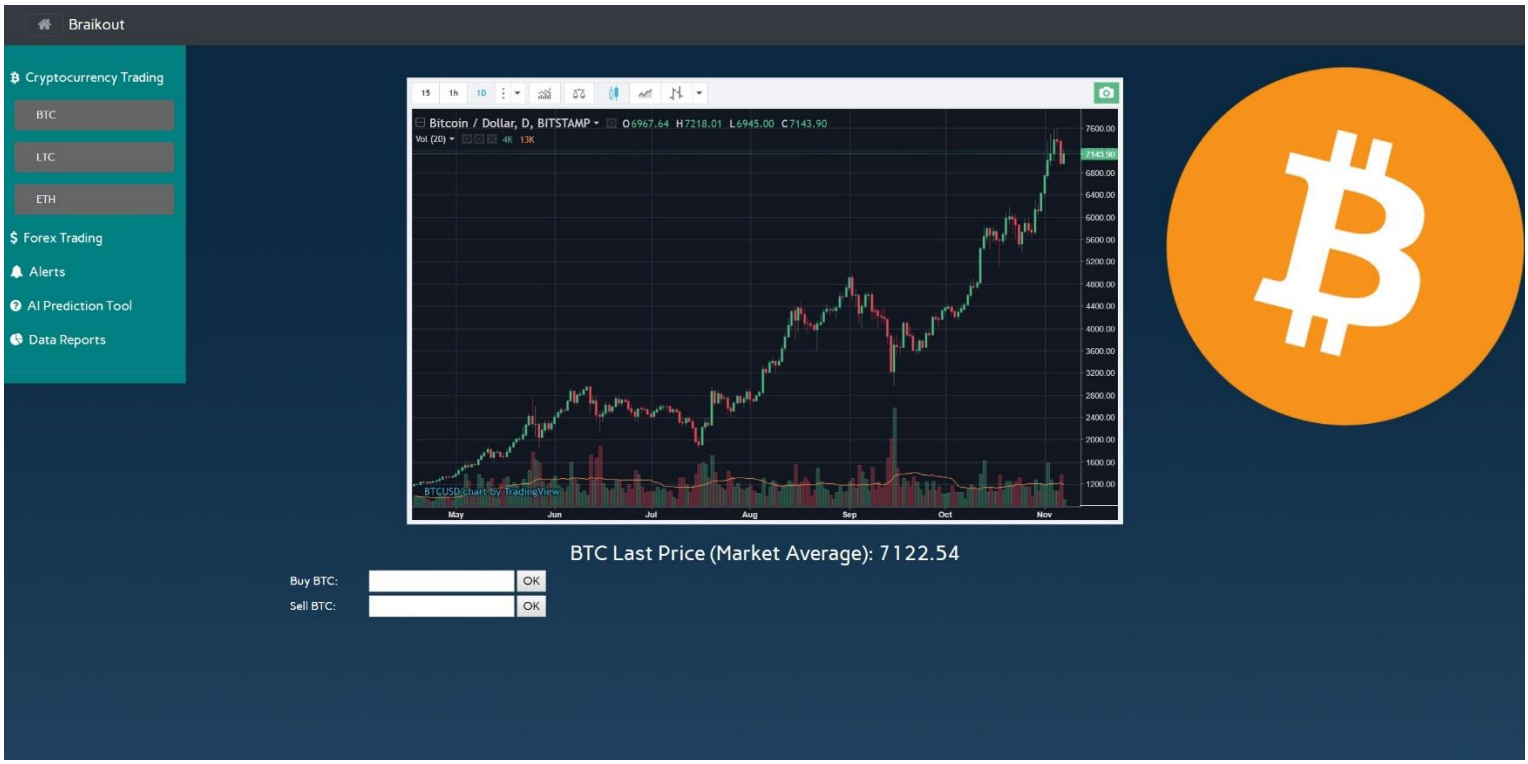
6.2. GANTT Chart

At Risk	Task Name	Status	Start Date	End Date
	Section 1 - Proposal			
	Sub-task 1 - Idea Generation & Project Proposal		09/01/17	09/26/17
	Section 2 - User Interface		**10/10/17**	**11/22/17**
	Django and Python Familiarisation		10/10/17	10/24/17
	Implement back end database		10/25/17	10/31/17
	Implement APIs		11/01/17	11/22/17
	Section 3 - Machine Learning		**11/23/17**	**01/27/18**
	Source previous market data		11/23/17	11/30/17
	Build model for chart theory Analysis		11/30/17	12/21/17
	Implement Sentiment Analysis		01/01/18	01/21/18
	Implement Chart Theory		01/22/18	02/14/18
	Build Machine Learning Algorithm		02/14/18	02/28/18
	Section 4 - Data Visualisation		02/28/18	03/01/18
	Implement Kibana Charts Page		02/28/18	03/07/18
	Analyze difference in influencing factors across the markets		03/07/18	03/10/18
	Section 5 - Testing		02/25/18	03/31/18
	Back-End Testing		03/12/18	03/19/18
	Unit Testing		03/19/18	03/26/18
	Integration Testing		03/26/18	04/02/18
	Boundary Value Testing		04/02/18	04/09/18
	UAT & Usability Testing		04/09/18	04/23/18

7. Appendices

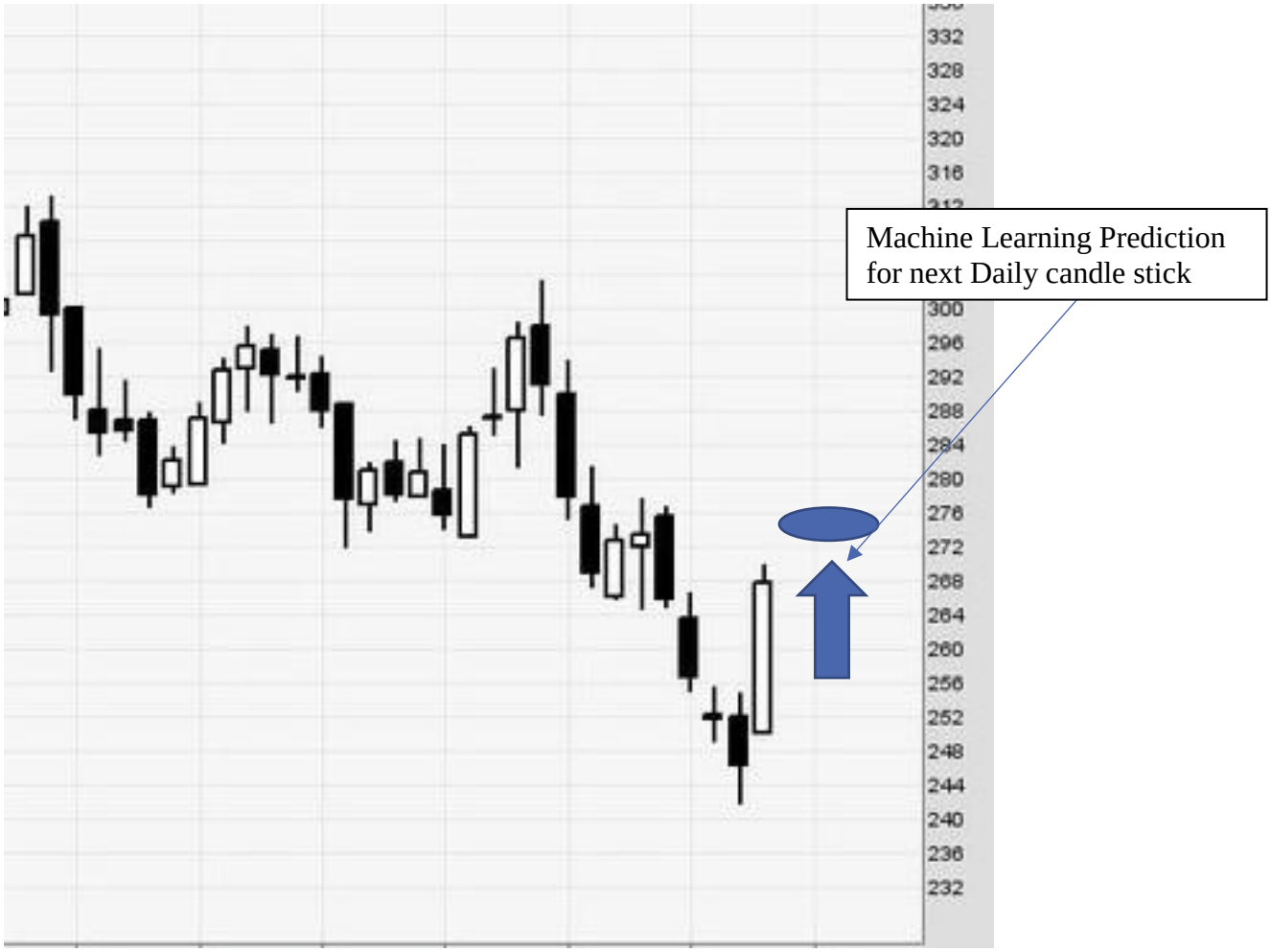
7.1. Trading

Below is a mock-up simplified image of the envisioned trading user interface (UI).



7.2. Machine Learning

While the rise of the internet has allowed for the average person to get involved with trading and investing, it has also resulted in a large difficulty in the competition. It is estimated that High frequency trading (HFT) comprises around 70% of wall street trades. HFT refers to a type of algorithmic trading characterized by high speeds and high turnover rates that make use of electronic prediction tools and previous financial data. These algorithms, of course, are not disclosed and as such, the average trader using solely his own judgement and chart analysis is at a severe disadvantage when compared with those participating in HFT.
The machine learning module in this application aims to level the playing field by making use of the same market history, and combining it with the traditional methods of technical analysis to develop a model which gives the user a much larger degree of predictive accuracy. The following is an example of what the user may see when they run the machine learning algorithm on a market they are interested in.



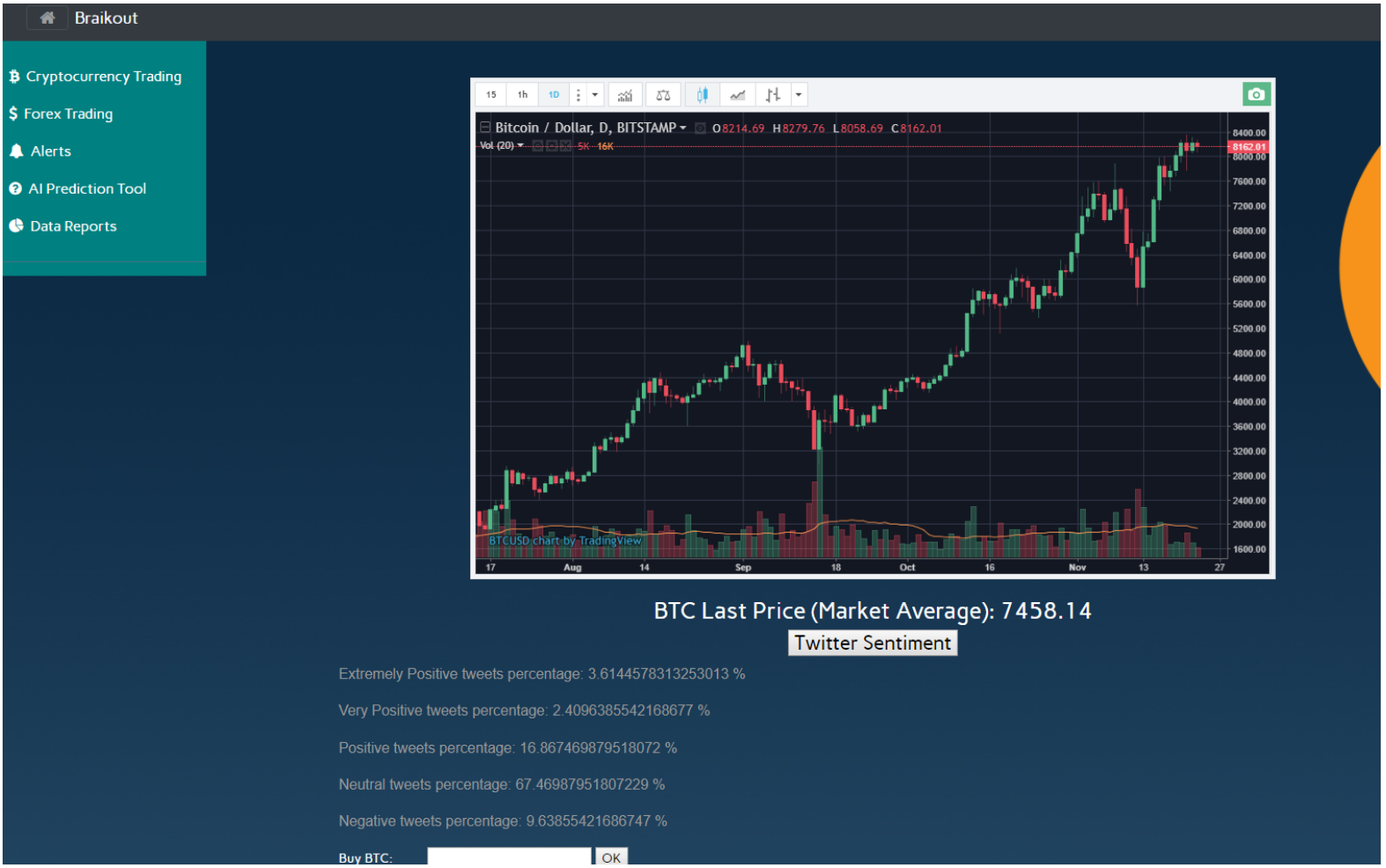
7.3. Technical/Chart Analysis

The basis of technical analysis, or TA, (As opposed to fundamental analysis) is that the previous price action and past trading activity are better indicators of the future of a security/currency than the perceived intrinsic value. There are many branches of technical analysis, and they all revolve around identifying key price levels as important areas of support (an area past which the price should not fall easily), and resistance (an area over which a price should have a difficult time moving). The manual act of examining charts and determining these values, regardless of the method one uses, is very time consuming. Add to this the fact that traders will want to watch multiple different currencies at once, and it becomes a near impossible task to achieve full market coverage without hours of time invested, and all of this while algorithms are instantly acting on these indicators in wall street without hesitation. The chart analysis module aims to automatically find these values for the trader so that they can instantly make trades without the otherwise necessary time investment. In order to illustrate the advantages of technical or chart analysis in this application, the below screenshot of a recent personal trading experience with the cryptocurrency “Litecoin” will be used. The blue circles on the image below represent the points at which the user of Braikout would have been alerted during this chart action.

As can be seen above, the initial spike in volume (indicated by the green bars on the floor) led to upward price action. The user of Braikout would have been alerted of this volume spike and had the opportunity to make a trade before the increase in price. After this, a triangular flag was formed, marked by the white lines above. As soon as this triangle broke upwards, this signalled a buy. At this point, the Braikout user would be notified again that a flag pattern has just broken upwards. The resulting price action after this occurred speaks for itself. Thanks to Braikout, the user would not need to perform the manual technical analysis and chart observation that was needed in the case above.

7.4. Sentiment Analysis

Another aspect of trading, and the opposite approach to chart analysis mentioned above, is fundamental analysis. This involves determining the intrinsic value of the asset being traded, and is undoubtedly an influencer on price action. A key example is the news of China banning cryptocurrency exchanges, which lead to a market drop of 40% within less than two weeks. For traditional currencies, a good example is the price effect on the dollar of the news of Donald Trump being elected president of the United States, or the initial effect of Brexit on the British Pound (GBP).
This sentiment analysis module aims to detect these large stories and general sentiment left in the market as a result, through scanning various news sites which are specific to financial information, as well as searching for tweets (which typically provides near instant news reaction), in order to identify large fundamental changes in a market that the trader may otherwise be unaware of. Below is an example of what the sentiment raw output would look like if the user chose to view sentiment from twitter on the trading page.



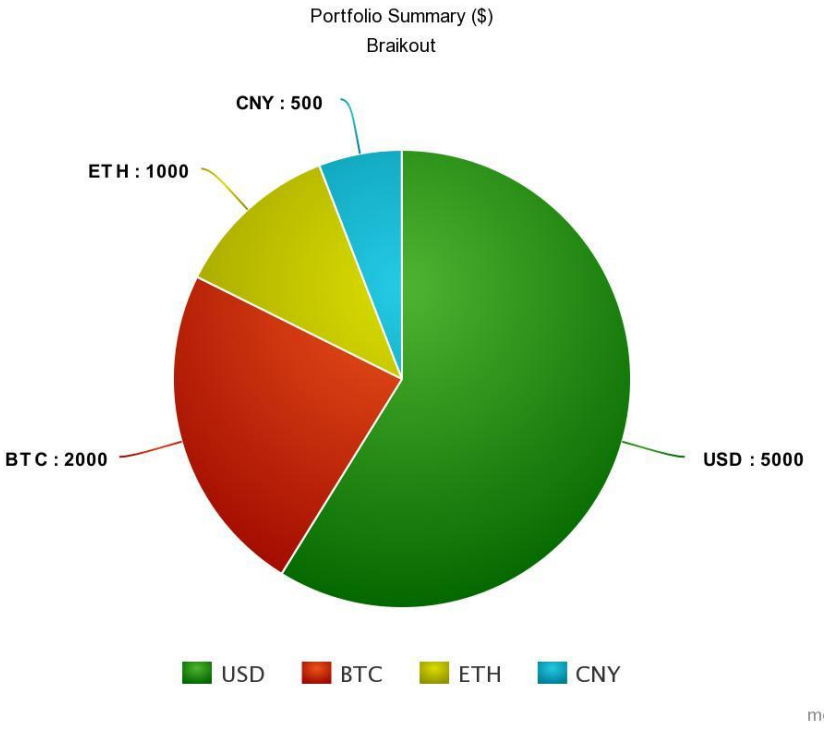
7.5. Alerts

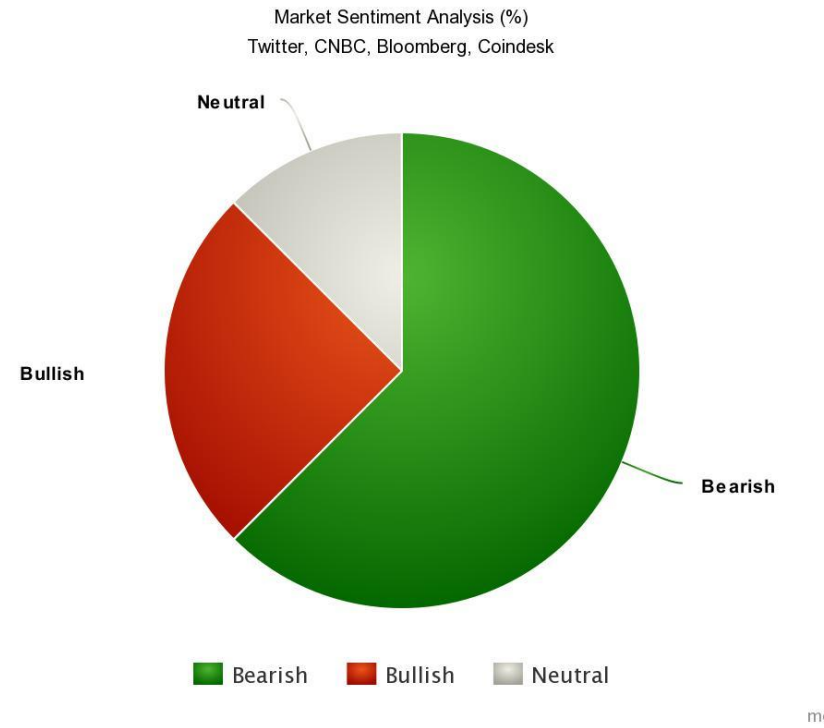
The alert functionality of this application affords the average person the ability to spend time away from their computer, while still not missing out on market opportunities. It is very difficult to maintain an effective trading protocol while also having a traditional job. However, much of the time spent trading will involve no active participation, rather analysing the market for opportunities. Thus, the user can now spend valuable time away from their computer, and simply be alerted of important technical or fundamental indicators relevant to the markets they choose to subscribe to.

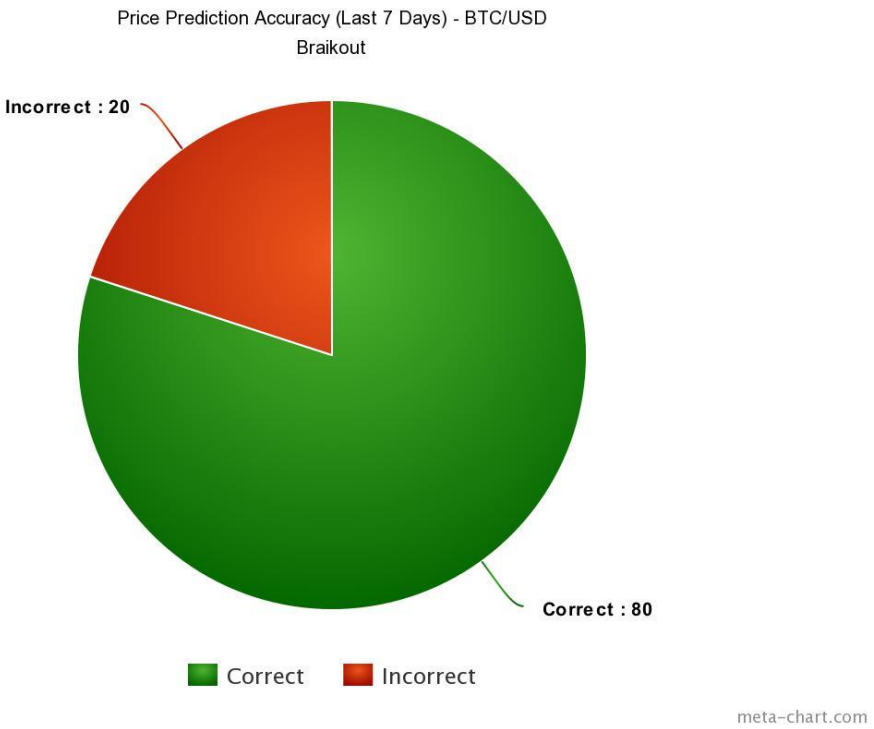
7.6. Data Visualisation

The data visualisation page will provide the user with important information regarding their trading history, but also an analysis of the system itself. This will act as an evaluation method of the various modules of the system. Particularly of interest is the effectiveness of the machine learning module in predicting correct price direction. Below is an example of the type of data visualisation a user will encounter. The first

is a summary of their current trading portfolio, with the next being an evaluation of the machine learning module's performance. Finally, an example of market sentiment data visualisation can be seen.







Of course, further data visualisation will be carried out, however the above is an indication of the various kinds of statistics that this application will store and analyse.

7.7. References

GANTT Chart – www.smartsheet.com
Architectural, component, use case diagrams – www.creately.com
Pie Charts – www.meta-chart.com