Crypt Predict
Design Document

BSc (Hons) in Software Development

Name: Adam Eaton

Student ID: C00179859

Year: 4th Year

Supervisor: Lei Shi

Due-Date: 18 - 04 - 2018

Table of Contents

1. Abstract

The purpose of this document is to describe the overall architecture and the system design of the application. It will outline both the user interface as well as the core functionality.

2. Introduction

This design document will outline the required architecture to implement the functionality described within the Functional Specification. This will include a series of Use Cases, System Sequence Diagrams, details regarding the User Interface as well as information related to the formatting of the stored Data.

3. Brief Use Cases

3.1. Price Prediction Method

Name: Price Prediction Method

Actors: User, Application, Datastore

Description: This use case begins when the user submits a request to the application to run the Price Prediction method. For each of the selected cryptocurrencies the associated data is selected, this data is then validated to assure there are no missing values. The validated data is then ran through the prediction generation function, the returned values are then used to generate a list of datetime items. The method choice, predicted data and the datetime list are then passed to the graph generating method, this returns HTML code for a div containing the generated graph. This returned HTML is then set as a the Flask variable for the cryptocurrencies graph which will appear once the results page is loaded.

3.2. Test Accuracy Method

Name: Test Accuracy Method

Actors: User, Application, Datastore

Description: This use case begins when the user submits a request to the application to run the Test Accuracy method. For each of the selected cryptocurrencies the associated data is selected, this data is then validated to assure there are no missing values. The validated data is then ran through the test generation method, this returns two values, for both the actual values and the predicted values. Both the actual values and the predicted values are then added to the datastore for accuracy data. The predicted values for the chosen cryptocurrency are then passed to the date prediction function, which returns a list of generated datetime items. Following this the method choice, the actual values, the predicted values and the time values are passed to the graph generating method which returns HTML code for a div containing the generated graph. This values is then set as the Flask variable for the cryptocurrencies graph which will appear once the results page is loaded.

3.3. Send Error Notification

Name: Send Error Notification

Actors: Application, Slack API

Description: This use case begins when an error occurs during runtime within the application. When the error is thrown the send notification function is called with a predefined message passed as a parameter, this parameter is a string denoting the file the error occured in and the function that was running while the error occured. The send notification function then creates an instance of the SlackClient with the stored slack token as a parameter. Following this it retrieves the most recent error raised on the stack and appends this to the end of the message passed. The Slack API is then called with the selected channel and the message text passed as parameters, resulting in the message being posted into the chosen channel.

3.4. Calculate Accuracy

Name: Calculate Accuracy

Actors: User, Application, Datastore

Description: This use case begins when the index page of the Application is requested. There are then three flask variables, one denoting each of the cryptocurrencies. For each of these values the calculate accuracy method is called with a string of a shortened version of the associated cryptocurrency name passed as a parameter. Within the calculate accuracy method the file containing the data regarding accuracy is opened and both the predicted values and the accuracy values for the given cryptocurrency are retrieved and stored in seperate arrays. Using these values the overall accuracy of the given cryptocurrency is calculated and returned to be displayed when the index page is rendered.

3.5. Run Data Collection

Name: Run Data Collection

Actors: User, Data Collection Script, Datastore, OKCoin API

Description: This use case begins when the User runs the Data Collection Script. The script will infinitely run a series of queries related to the cryptocurrencies every twenty seconds. For each cryptocurrency the script makes two queries to the OkCoin API, one request for the Ticker Data and one for Depth Data. The JSON response of these queries are then parsed for the information regarding current time in epoch, current price, volume of bids and volume of ask prices. These elements are stored within a list and formatted as a single string. These strings are then written into the Price related datastore for the associated Cryptocurrency.

4. Detailed Use Cases

4.1. Price Prediction Method

Name: Price Prediction Method

Actors: User, Application, Datastore

Description: This use case begins when the user submits a request to the application to run the Price Prediction method. For each of the selected cryptocurrencies the associated data is selected, this data is then validated to assure there are no missing values. The validated data is then ran through the prediction generation function, the returned values are then used to generate a list of datetime items. The method choice, predicted data and the datetime list are then passed to the graph generating method, this returns HTML code for a div containing the generated graph. This returned HTML is then set as a the Flask variable for the cryptocurrencies graph which will appear once the results page is loaded.

Main Success Scenario:

1. The user opens the application.
2. The user selects their chosen parameters, selecting the Price Prediction method.
3. The user submits the request to the application.
4. For each of the chosen Cryptocurrencies, their price data is read in.
5. This price data is validated successfully.
6. A series of predictions are generated using this validated data.
7. A list of datetime items are generated using the predictions.
8. The method, predicted values, time values are passed to generate the HTML code for a graph based on the provided data.
9. The flask variable for the associated cryptocurrencies graph is set as this HTML code.
10. The results page is rendered and the graph is displayed.

4.2. Test Accuracy Method

Name: Test Accuracy Method

Actors: User, Application, Datastore

Description: This use case begins when the user submits a request to the application to run the Test Accuracy method. For each of the selected cryptocurrencies the associated data is selected, this data is then validated to assure there are no missing values. The validated data is then ran through the test generation method, this returns two values, for both the actual values and the predicted values. Both the actual values and the predicted values are then added to the datastore for accuracy data. The predicted values for the chosen cryptocurrency are then passed to the date prediction function, which returns a list of generated datetime items. Following this the method choice, the actual values, the predicted values and the time values are passed to the graph generating method which returns HTML code for a div containing the generated graph. This values is then set as the Flask variable for the cryptocurrencies graph which will appear once the results page is loaded.

Main Success Scenario:

1. The user opens the application.
2. The user selects their chosen parameters, selecting the Test Accuracy method.
3. The user submits the request to the application.
4. For each of the chosen Cryptocurrencies, their price data is read in.
5. This price data is validated successfully.
6. This validated data is used to generate two lists, actual values and predicted values.
7. Both the actual values and predicted values are added to the accuracy datastore.
8. A series of datetime items are generated using the predicted values.
9. The method, actual values, predicted values, time values are passed to generate the HTML code for a graph based on the provided data.
10. The flask variable for the associated cryptocurrencies graph is set as the HTML code.
11. The results page is rendered and the graph is displayed.

4.3. Send Error Notification

Name: Send Error Notification

Actors: Application, Slack API

Description: This use case begins when an error occurs during runtime within the application. When the error is thrown the send notification function is called with a predefined message passed as a parameter, this parameter is a string denoting the file the error occured in and the function that was running while the error occured. The send notification function then creates an instance of the SlackClient with the stored slack token as a parameter. Following this it retrieves the most recent error raised on the stack and appends this to the end of the message passed. The Slack API is then called with the selected channel and the message text passed as parameters, resulting in the message being posted into the chosen channel.

Main Success Scenario:

1. An error occurs during application runtime.
2. An exception is raised.
3. The message is passed to the notification function.
4. An instance of SlackClient is created using the provided token.
5. This most recent value on the stack is retrieved.
6. This value is appended to the end of the message provided.
7. The Slack API is called, passing the channel and the message as parameters.
8. The message is posted in the Slack channel, thus notifying the developer of the error.

4.4. Calculate Accuracy

Name: Calculate Accuracy

Actors: User, Application, Datastore

Description: This use case begins when the index page of the Application is requested. There are then three flask variables, one denoting each of the cryptocurrencies. For each of these values the calculate accuracy method is called with a string of a shortened version of the associated cryptocurrency name passed as a parameter. Within the calculate accuracy method the file containing the data regarding accuracy is opened and both the predicted values and the accuracy values for the given cryptocurrency are retrieved and stored in seperate arrays. Using these values the overall accuracy of the given cryptocurrency is calculated and returned to be displayed when the index page is rendered.

Main Success Scenario:

1. The user request the index page of the application.
2. For each of the flask variables associated with each of the cryptocurrencies, the calculate accuracy method is ran with a shortened version of each cryptocurrency name passed as a variable.
3. For each cryptocurrency the accuracy datastore is opened and all data for past predicted values and accuracy values for the associated cryptocurrency are retrieved and stored in seperate arrays.
4. Using these values the overall accuracy of the given cryptocurrency is calculated.
5. The overall accuracy score is returned.
6. The index page is rendered with the accuracy score for each cryptocurrency being displayed.

4.5. Run Data Collection

Name: Run Data Collection

Actors: User, Data Collection Script, Datastore, OKCoin API

Description: This use case begins when the User runs the Data Collection Script. The script will infinitely run a series of queries related to the cryptocurrencies every twenty seconds. For each cryptocurrency the script makes two queries to the OkCoin API, one request for the Ticker Data and one for Depth Data. The JSON response of these queries are then parsed for the information regarding current time in epoch, current price, volume of bids and volume of ask prices. These elements are stored within a list and formatted as a single string. These strings are then written into the Price related datastore for the associated Cryptocurrency.

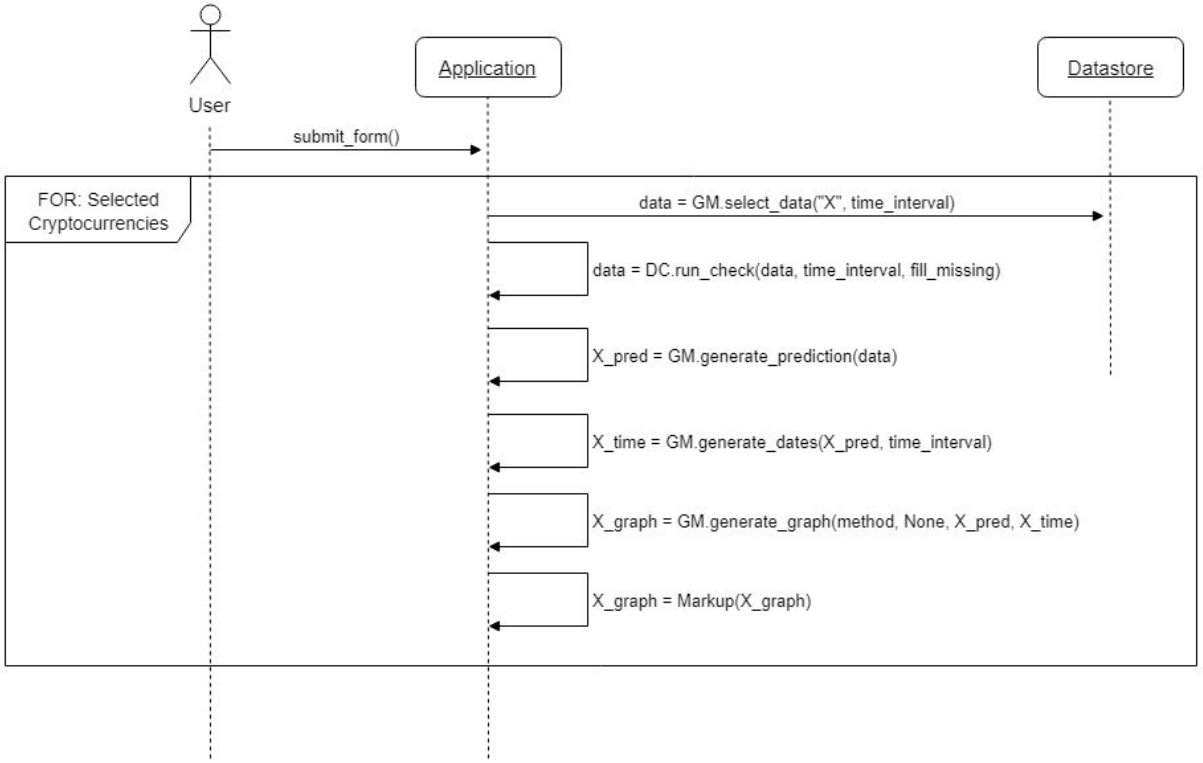
Main Success Scenario:

1. The user runs the Data Collection Script.
2. The Script calls the function to run the series of queries.
3. For each cryptocurrency a query for both Ticker data and Depth data is made.
4. The JSON response to these queries is parsed to retrieve data regarding current time in epoch, current price, volume of bids, volume of ask prices.
5. These values are added to a list.
6. This list is converted to a single string of comma separated values.
7. This string is written to price related datastore for the associated cryptocurrency.
8. The script sleeps for 20 seconds.
9. Steps 2 to 8 are repeated until the script is stopped.

5. System Sequence Diagrams

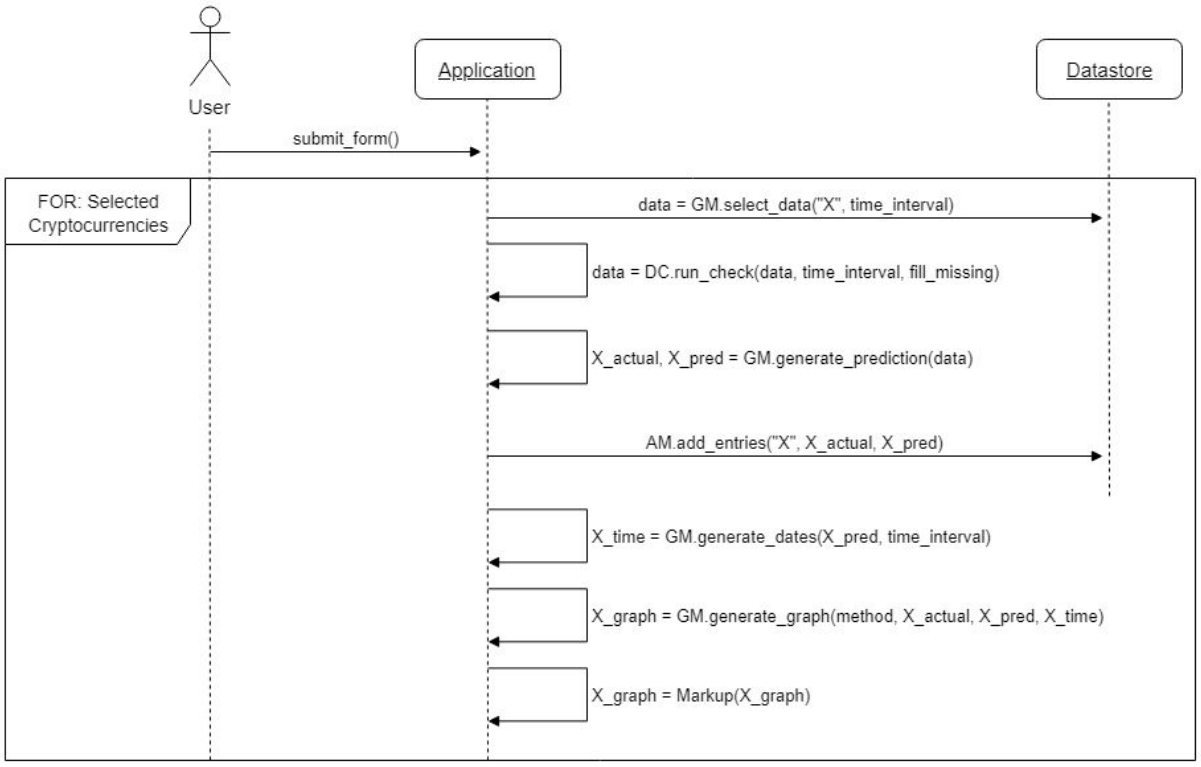
5.1. Price Prediction Method

Note that in the diagram below X would be replaced with the associated Cryptocurrencies shortened name during runtime. Those being; Bitcoin = btc, Ethereum = eth, Litecoin = ltc.



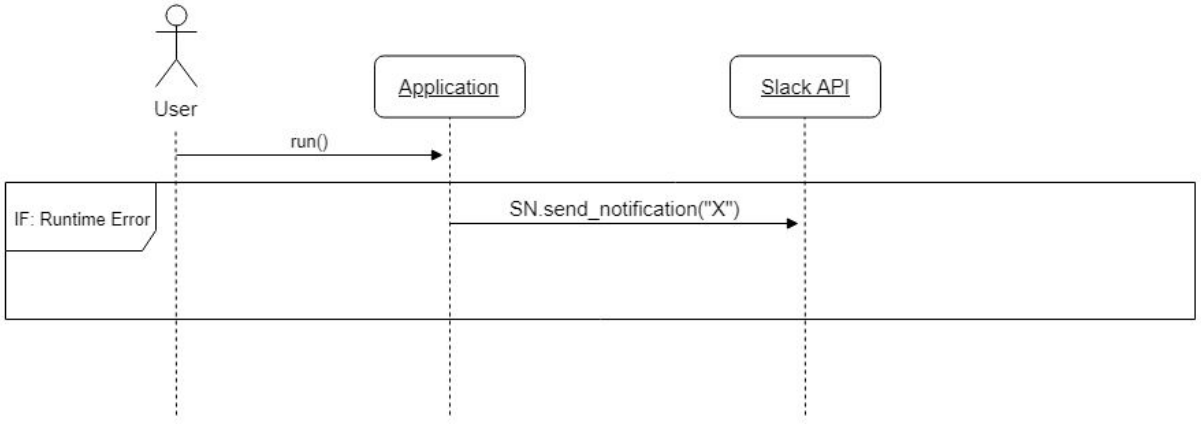
5.2. Test Accuracy Method

Note that in the diagram below X would be replaced with the associated Cryptocurrencies shortened name during runtime. Those being; Bitcoin = btc, Ethereum = eth, Litecoin = ltc.



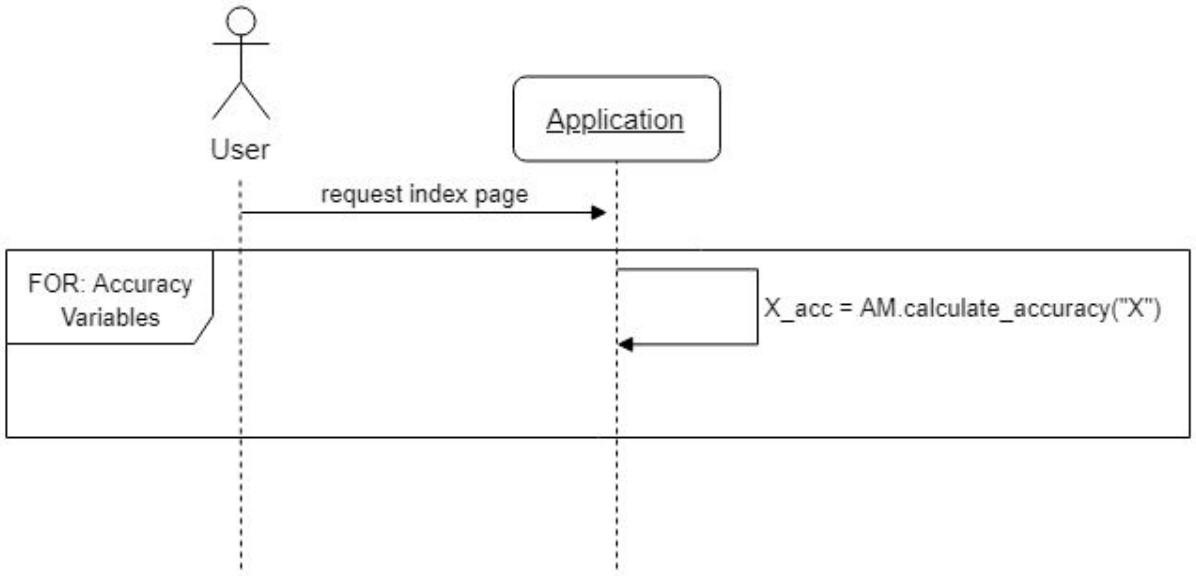
5.3. Send Error Notification

Note that in this case, X would be a string containing the name of the file and function in which the error occured.

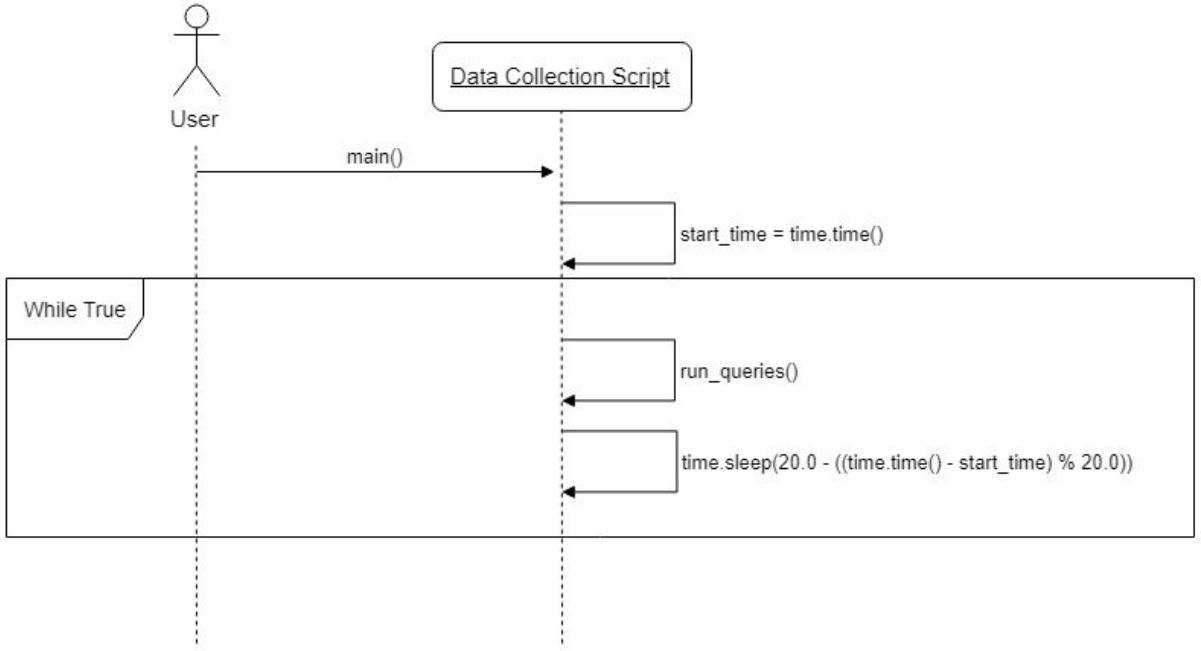


5.4. Calculate Accuracy

Note that in the diagram below X would be replaced with the associated Cryptocurrencies shortened name during runtime. Those being; Bitcoin = btc, Ethereum = eth, Litecoin = ltc.



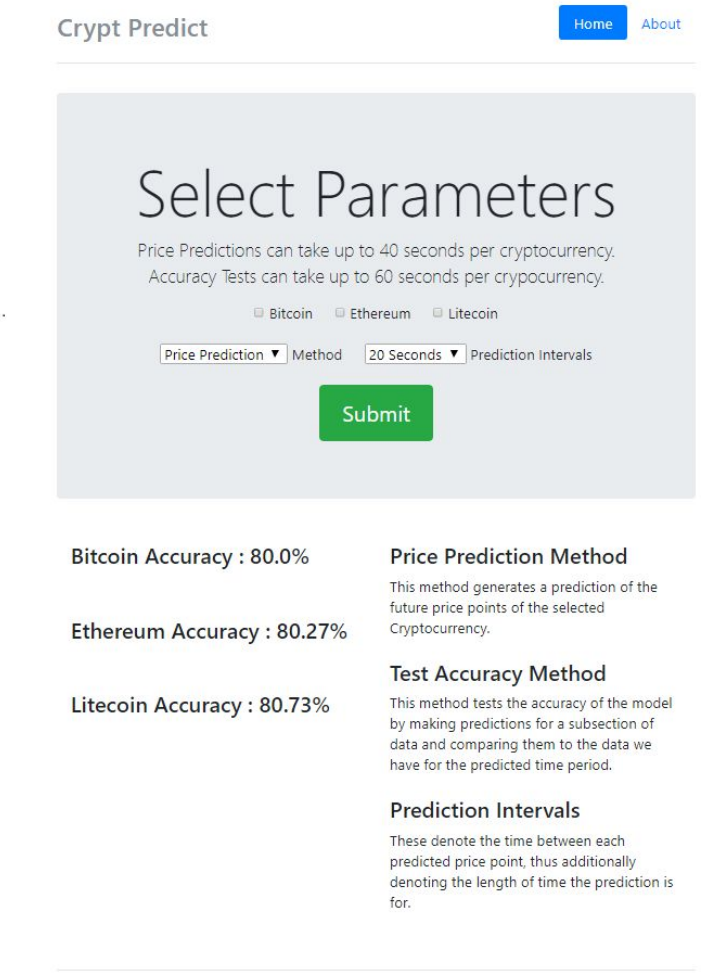
5.5. Run Data Collection



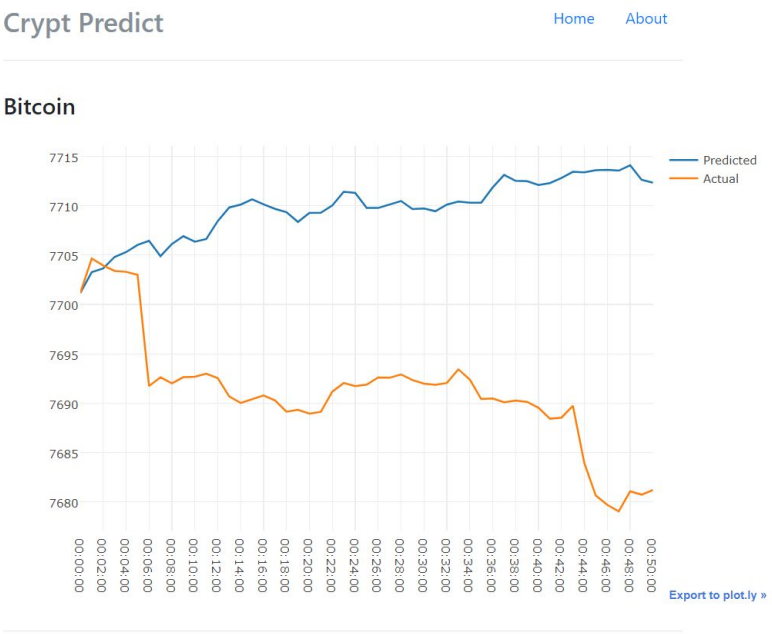
6. User Interface Design

The User Interface will be implemented using the readily available code from Bootstrap as the base HTML and CSS. In addition to this base HTML a form to allow for user parameters and an additional page for displaying results will be implemented.

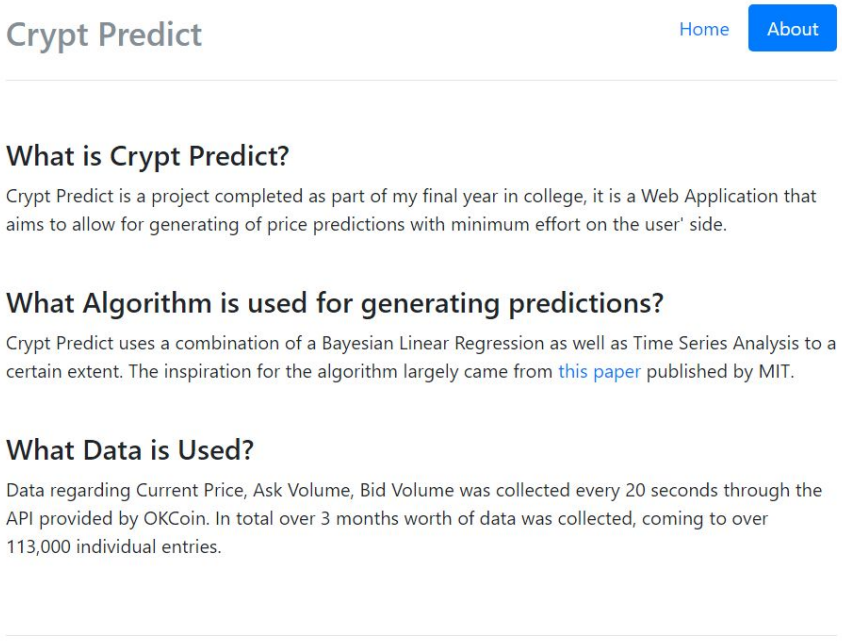
The first of three screens will be the index page, on this we will display the parameter form, the overall accuracy of the predictions made for each cryptocurrency as well as details to give a brief explanation of the key parameters. This screen is shown in the image below.



The next screen is the results page, this is quite simply a page retaining the nav bar from the Home screen, but on it each graph that was generated will be displayed. For the purpose of this screenshot i've generated just a single graph using the Test Accuracy method.



The final of the three screens is the About screen, it quite simply contains information regarding the project, acting as a FAQ of sorts.



7. Data Format

7.1. Price Data

Price related data has been broken down into individual files for each cryptocurrency. Within each of these files there are four columns of data stored, those being:

- date, which stores the date and time at which the data was retrieved in epoch format.
- price, which stores the price of the given cryptocurrency when the data was retrieved.
- vAsk, which stores the volume of Ask bids at the time the data was retrieved.
- vBid, which stores the volume of Bids at the time the data was retrieved.

Below is a subsection of data from the btc_data file which stores data related to Bitcoin.

date	***price***	***vAsk***	***vBid***
1522631419	7701.16	146.2303	29.0112
1522631440	7682.92	147.1853	30.6075
1522631459	7682.92	147.3303	25.7475

7.2. Accuracy Data

Accuracy Data is a datastore of the predicted values generated when the Test Accuracy method is ran. It stores both the predicted value, it's corresponding actual value as well as a variety of other features. Within the accuracy_data file it is broken down into the following five columns:

- crypto, this denotes which cryptocurrency the data is related to. Possible values and their corresponding cryptocurrency are:
    - btc - Bitcoin
    - eth - Ethereum
    - ltc - Litecoin
- predicted, this is one of the predicted values generated through the use of the Test Accuracy method.
- actual, this is the corresponding actual value to the predicted value generated through the Test Accuracy method.
- diff, this is the percentage difference between the predicted value and its corresponding actual value.
- accuracy, this is a value generated as part of the overall accuracy per cryptocurrency.

Below is a subsection of data from within the accuracy_data file.

crypto	***predicted***	***actual***	***diff***	***accuracy***
btc	7670.75	7673.99	3.24	0.42
btc	7671.06	7673.81	2.75	0.36
eth	422.88	422.74	-0.14	-0.33