Crypt Predict
Functional Specification

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 set out both the core functional and nonfunctional requirements for this project. It will cover the aforementioned requirements as well as topics including the target market of the project, the metrics by which we can assess the success of the project and the planned work breakdown for each iterative stage within the project.

2. Introduction

This project is intended to act as a Cryptocurrency price prediction platform which can be used to assist investors in making more educated decisions when decided to buy / sell. The project was ultimately designed with the idea of making it easy to implement additional cryptocurrencies as well as making it simple to use for any given user.

The project allows for a user to select their chosen cryptocurrency(s) between Bitcoin, Ethereum and Litecoin, they then choose whether they want to generate a future price prediction or to test the accuracy of the model, and finally they select the period of time between each prediction interval. After the user selects their chosen parameters, the application will run the selected method and forward them to a page displaying the graphs generated from their chosen parameters.

The project requires a significant amount of data to run it's different methods, each method requires a minimum of 27000 entries. This data is collected through querying the OkCoin API every 20 seconds to retrieve information regarding the cryptocurrencies, the returned JSON is then parsed and the chosen information is stored in separate CSV files for each cryptocurrency.

3. Target Market

This project is a tool that is created with existing cryptocurrency investors and potential investors in mind. The hope is that these users can utilise the project to assist them in making better educated decisions when trading cryptocurrency.

4. Metrics

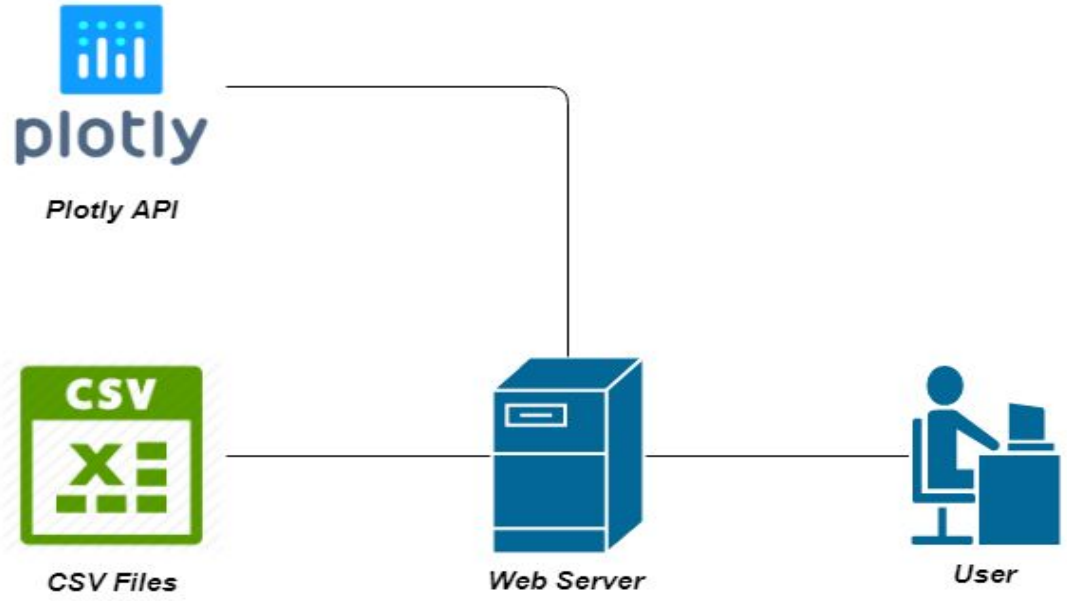
As the nature of this project is the prediction of future prices there are several metrics readily available to assess the success of the project. Naturally the a key metric will be whether or not the planned functionality has been successfully implemented, additionally the overall ease of using the application and the accuracy of the predictive algorithm.

5. System Architecture

This application is primarily a web based application, however it can also be ran on a local machine, additionally the data collection process should be ran from a local machine. Within this section we will break each of these different architectures down into their own sub-sections as well as an overview of the technologies used within the overall architecture.

5.1. Web Hosted Architecture

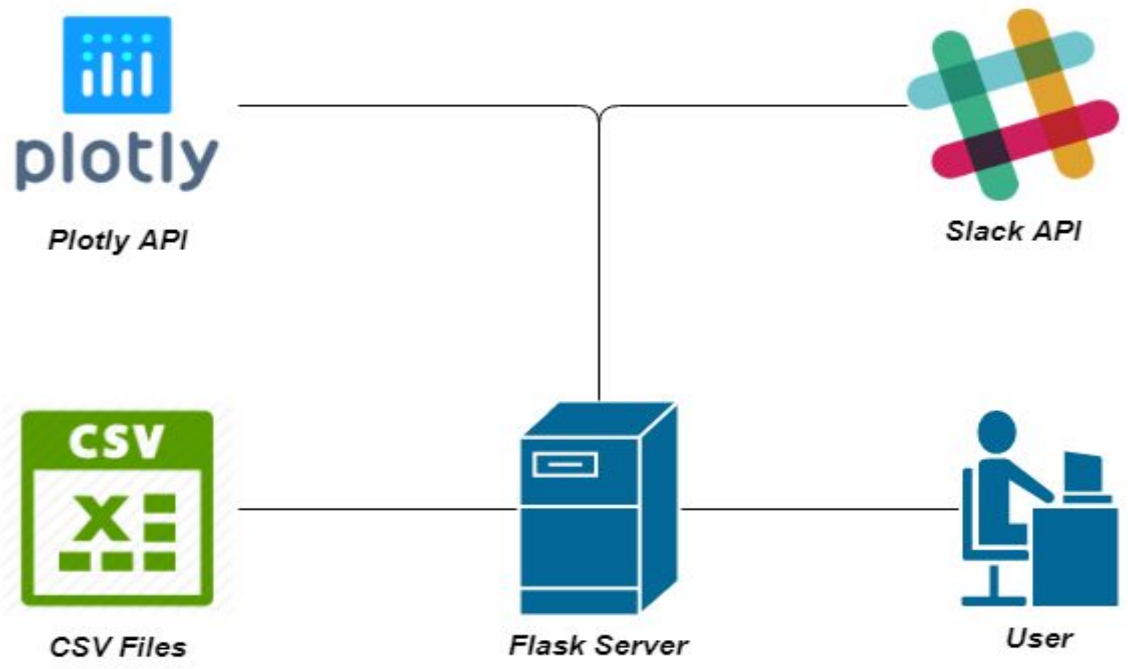
The web hosted application will retain the core functionality of the application, however due to restrictions enforced by the hosting service the data collection functionality and the error notification functionality could not be implemented. As such they will not be included in the architecture diagram.



The Web Server will act as the point of interaction between the User and the underlying code. The user will submit a request to the Web Server with their choice of parameters, and based on those parameters the predetermined code segments will be executed. Regardless of method chosen the Web Server is guaranteed to interact with both the Plotly API and the CSV Files when running. The CSV Files are used as a store for all data regarding each of the individual cryptocurrencies as well as the data regarding accuracy score of predictions made. The Plotly API is utilised whenever the Web Server needs to generate a graph, selected data is passed to the API which returns HTML code for a graph displayed the previously passed data, this graph code is then passed as a parameter to the Web Server to be displayed within the results page.

5.2. Locally Hosted Architecture

By locally hosting the application the user will be able to utilise the functionality which was restricted by the hosting service, those being the use of Slack to send notifications of runtime errors, as well as running the Data Collection script to gather new information on prices. For the purpose of this diagram I have excluded the Data Collection Architecture and will discuss that in the next section.



The architecture of hosting the application locally is extremely similar to the architecture of the web hosted alternative, the main difference between the two is the additional use of the Slack API. This API allows the user to generate and send slack messages to their chosen Workspace and even a specific channel within that workspace. The messages generated are based on a pre-written string detailing the file and the function in which the error occurred, additionally the most recent error on the stack is appended to the end of the message for more detailed information regarding exactly which error has occurred.

5.3. Data Collection Architecture

The Data Collection Architecture is relatively simple, it gives a clear idea of the process of the collection and the storage of the information. The Data Collection script should be ran as its own process however through the use of Multi-Threading and scheduling it could hypothetically be made an addition to the core application.

The architecture diagram is similar to the previously shown diagrams with the exception of the lack of a user, due to that the script should be ran as its own process, and the replacement of the Plotly API with the OKCoin API.

The Data Collection process involves the script running a series of queries to the OKCoin API every 20 seconds, specifically one for each of the Cryptocurrencies. The returned JSON from these queries is then parsed for the information regarding the current date in epoch value, the current price of the chosen cryptocurrency, the volume of bids and the volume of asking prices. These values are then converted to a comma separated string and stored in their designated CSV file and a ‘Tick’ message is printed to the console, acting as a means of recognising the script is still executing. If an error occurs during this process an error message is generated and sent to the provided Slack channel.

5.4. Technology Overview

Below are the intended core technologies that will be used within the application, each will have been discussed in more detail in the research document.

- Python 3.6 as the core programming language.
- Flask as the Web Framework to handle the HTTP requests.
- Bootstrap to reduce the amount of time spent developing the front facing elements of the application.
- Plotly as the means to graph the generated data.
- Bitcoin, Ethereum and Litecoin as the chosen Cryptocurrencies.
- OKCoin' API to retrieve the price information for the chosen cryptocurrencies.
- CSV Files to act as datastores.
- Slack will act as a means of notifying the developer of runtime errors.
- PythonAnywhere as the hosting service.

6. Supplementary Specification

6.1. Functionality

- The APIs used within the project need to be available to interact with.
- An internet connection will be required whenever running the Data Collection script, however the core application should function without a connection.
- The application will send a Slack notification whenever a runtime error occurs.
- An overall accuracy score should be available, this score should be updated whenever new values are added to the accuracy datastore.

6.2. Usability

As previously mentioned, the application has been developed with ease of use on both the end user and the developers side. By making the UI simple to understand with tips on what individual parameters account for provided it aims to allow even the newest of users to understand the operation of the application. In addition, the process of adding a new Cryptocurrency to the application is intended to be a simple process, allowing future developers to improve on the selections available with ease.

- A user should be able to clearly understand the parameters that need to be selected in order to generate predictions or to test the accuracy of the model.
- The graph(s) generated should be easy to understand with clear labels to differentiate graphs if multiple have been generated simultaneously.
- If the application has been downloaded with the intention of running it locally, the process of setting up and beginning to use the application should be clearly laid out.

6.3. Reliability

- The Web Hosted Application should have absolutely minimum downtime.
- In the event of a runtime error the developer should be notified immediately via Slack notification, assuming there is an internet connection available.
- A user should be able to leave the Data Collection script run without issue outside of API side errors occuring.

6.4. Performance

- The generation of a prediction or accuracy test should be completed within a timely manner as to not lose the attention of the user.
- The GUI must be responsive aside from the aforementioned generations.
- The application should be capable of processing large datasets.

6.5. Supportability

- Additional Cryptocurrencies should be simple to implement.
- Additional Cryptocurrency data sources should be simple to implement.
- The application should be usable with all modern browsers.

7. Iteration Plan

Throughout this project an agile development approach will be applied, this will naturally involve several iterations within the development period. At the end of each of these iterations the work completed during it will be presented along with relevant information such as the plan for the next iteration and any problems that might have occurred during the previous iteration.

7.1. Iteration 1

This iteration is planned to run over the course of six weeks, from November 27th 2017 to January 8th 2018.

7.1.1. Planned Schedule of Work

- Create databases for storing Cryptocurrency' data.
- Develop the functionality to query the OKCoin API every 20 seconds, parse the response and store the information.
- Develop a basic version of the Web Application.
- Begin the development of the Algorithm.
- Source a suitable dataset to test the algorithm with while building our own datastore.

7.1.2. Changes during Iteration

- Switched from using MongoDB to using CSV files due to an issue of transferring data off of the computer designated to collect data.
- Put the development of the Web Application on hold and focused more heavily on the development of the Algorithm.

7.2. Iteration 2

This iteration is intended to run for five weeks, between January 15th 2018 to February 19th 2018.

7.2.1. Planned Schedule of Work

- Further develop the Algorithm, begin to use self collected data.
- Develop the graphing method using Plotly.
- Develop a means of checking that the data being used is suitable and not missing any fields.
- Develop method to determine accuracy of Algorithm.
- Develop a means of notifying the developer in the event of a runtime error.

7.2.2. Changes during Iteration

- Postponed the development of the means of determining accuracy in order to focus on finishing more core functionality.

7.3. Iteration 3

This final iteration is intended to run for a final six weeks, between February 26th 2018 until April 9th 2018.

7.3.1. Planned Schedule of Work

- Develop Web Application and implement existing functionality.
- Develop means of calculating accuracy of algorithm.
- Host the application online.

7.3.2. Changes during Iteration

- Improved error notifications by having it utilise the actual runtime error messages instead of predefined messages.
- Due to restrictions in place on PythonAnywhere the slack package could not be installed, thus that functionality was removed on the Web Application.