



National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

Blockchain Data Analytics Tool

College of Engineering and Informatics

Bachelor of Science (Computer Science & Information
Technology)

Project Report

Author:

Jakub Wojtkowicz (17451684)

Academic Supervisor:

Dr Matthias Nickles

Table of Contents

Table of Contents	1
List of Figures	3
Chapter 2. Research.....	3
Chapter 3. Development.....	3
Chapter 4. The Application – CryptoChain	4
List of Tables	5
Chapter 3. Development.....	5
Chapter 1. Introduction	6
1.1 Project Overview.....	6
1.2 Document Structure.....	6
Chapter 2. Research	7
2.1 Introduction	7
2.2 Research Origins	7
2.3 Literature Review	9
2.4 Existing Software	13
2.5 Innovative Approach	14
2.6 Conclusion.....	15
Chapter 3. Development	16
3.1 Introduction	16
3.2 Planning.....	16
3.3 Tools.....	20
3.4 Frameworks.....	23
3.5 APIs.....	27
3.6 Languages.....	31
3.7 Libraries	33
3.8 Error Handling.....	35
3.9 Conclusion.....	38
Chapter 4. The Application – CryptoChain	39
4.1 Introduction	39
4.2 Application Architecture	39
4.3 Price Charts	42
4.4 Trend Correlations.....	47

4.5	Price Predictions.....	49
4.6	Investment Tools.....	52
4.7	Conclusion.....	55
Chapter 5. Results		56
5.1	Introduction	56
5.2	Evaluation.....	56
5.3	Deliverables.....	57
5.4	Future Work	57
5.5	Conclusion.....	58
References.....		59

List of Figures

Chapter 2. Research

Figure 2.1. Covid Movement [40].

Figure 2.2. Bitcoin Movement [2].

Figure 2.3. Blocks linked cryptographically through hashes.

Figure 2.4. Cryptocurrency Architecture.

Figure 2.5. Cryptocurrency market cap leaders, visualized data from Slikcharts [41].

Figure 2.6. Existing approach – Coindesk [3].

Figure 2.7. Metrics provided by Coindesk [3].

Chapter 3. Development

Figure 3.1. Project milestones Gantt chart.

Figure 3.2. Envisioned user interface.

Figure 3.3. Architecture flow diagram.

Figure 3.4. Maven utilisation.

Figure 3.5. Task Management Tool.

Figure 3.6. Git commit regularity.

Figure 3.7. Application running locally in a Docker container.

Figure 3.8. IDE version control integration.

Figure 3.9. Git Bash used for updating remote repository.

Figure 3.10. Spring initializer skeleton application.

Figure 3.11. Vaadin chart generation.

Figure 3.12. Vaadin components as Java objects.

Figure 3.13. Vaadin underlying library CSS.

Figure 3.14. Standard chart CSS.

Figure 3.15. Modified CSS.

Figure 3.16. JUnit test.

Figure 3.17. JUnit test completion.

Figure 3.18. Mockito @Mock annotation.

Figure 3.19. Mockito when, then return flow.

Figure 3.20. API request flow.

Figure 3.21. Send API request.

Figure 3.22. API request generation.

Figure 3.23. Postman API development.

Figure 3.24. Unmarshalling JSON.

Figure 3.25. JSON mapper.

Figure 3.26. CSS component assignment.

Figure 3.27. CSS component definition.

Figure 3.28. Building Project with Bash.

Figure 3.29. Lombok annotation used for getters and setters.

Figure 3.30. Joda Time date creation and use.

Figure 3.31. Machine learning autonomous CSV generation.

Figure 3.32. Exception source caught and console log printed.

Figure 3.33. Resulting downstream error due to lack of adequate return object triggers UI error alert.

Figure 3.34. UI error displayed.

Figure 3.35. Invalid dates excluded as Bitcoin data cannot be fetched for dates before Bitcoin existed.

Figure 3.36. Alert generation.

Chapter 4. The Application – CryptoChain

Figure 4.1. Application architecture.

Figure 4.2. View Generation.

Figure 4.3. Constructor generates components.

Figure 4.4. component generator.

Figure 4.5. Price Charts Page.

Figure 4.6. Price Charts page – chart style.

Figure 4.7. Price Charts page – history candlestick chart.

Figure 4.8. Price Charts page – history spline chart.

Figure 4.9. Price Charts page – technical analysis.

Figure 4.10. Price Charts page – moving average spline.

Figure 4.11. Price Charts page – set date scroll.

Figure 4.12. Price Charts page – set date button.

Figure 4.13. Price Charts page – set date input.

Figure 4.14. Price Charts page – live price positive.

Figure 4.15. Price Charts page – live price negative.

Figure 4.16. Price Charts page – trading volume.

Figure 4.17. Trend Correlations page.

Figure 4.18. Trend Correlations page – settings.

Figure 4.19. Price Predictions page.

Figure 4.20. Price Predictions page – testing downtrend.

Figure 4.21. Price Predictions page – testing uptrend.

Figure 4.22. Price Predictions page – settings.

Figure 4.23. Investment Tools page.

Figure 4.24. Investment Tools page – predicted highs & lows.

Figure 4.25. Investment Tools page – investment calculator.

Figure 4.26. Investment Tools page – relative strength index.

List of Tables

Chapter 3. Development

Table 3.1. Planned steps.

Table 3.2. APIs Integrated into application.

Chapter 1. Introduction

1.1 Project Overview

This Report was created to record the processes that were completed to construct a final year project application, these processes include research, planning, developing testing and an evaluation. The Below will outline all processes from inception to completion, the project's origins are in the provided Brief as follows.

“Blockchain Data Analytics: In this project, the student should develop a tool for the analysis of data or meta-data in a Blockchain (based on some existing Blockchain framework) using appropriate data analytics software, such as Apache Spark. An example would be the analysis of cryptocurrency transaction data. Requirement: Good programming skills (Python, Java or Scala).”

The purpose of this project is to create a provocative, useful, and innovative data analytics tool. The tool aims to visually present the cryptocurrency Bitcoin [8], its current transactions and prices then superimpose other commodities or trends on to it. Machine learning [56] will be employed to predict price movements based on user inputs and indicate ideal investing times. This design aims to discover the correlations between Bitcoin and other Trends. The discovered corelations will be indicate visually given the user imputed time frame and trend.

1.2 Document Structure

This Report is divided into 5 Chapters. The introduction followed by the research completed of literature and technical aspects of existing solutions. The development aspect which includes goals and requirements that have been set out to achieve the end result and technical details of tools used etc, the penultimate chapter entails the description and features of the completed application, and the final chapter discusses the results and evaluation of the developer and the users.

Chapter 2. Research

2.1 Introduction

This chapter describes the origins behind the research, literature review, and existing approaches. The literature review describes the garnered information on the topic and the technical review details the existing solutions and how they operate. The topic relevance will be covered to give the project some business context.

2.2 Research Origins

The research involved prior to beginning the project included reading many informative articles and explanations of what is and how blockchain works such as the one on Investopedia.com [7] and in the journal “A blockchain future for internet of things security” (*Banerjee, Lee and Choo, 2018*) [53]. These in dept explanations provided valuable information on how blockchain works, what it is used for and its perks and pitfalls.

This Material was very provocative and inspired more research into the cryptocurrency aspect of blockchain. Bitcoin, the most popular of the current cryptocurrencies was the most prevalent in the research conducted. Great websites exist on explaining Bitcoin and its innerworkings such as bitcoin.org [8] and journals such as “Bitcoin: Economics, Technology, and Governance” (*Böhme, Christin, Edelman and Moore, 2015*) [54] provide a myriad of examples and use cases of Bitcoin.

Investigations were conducted into price fluctuations of Bitcoin at a high level and a lot of movement was discovered before and during the current pandemic. This led to a belief that there must be some correlation between the two and that more research at a low level needs to be done.

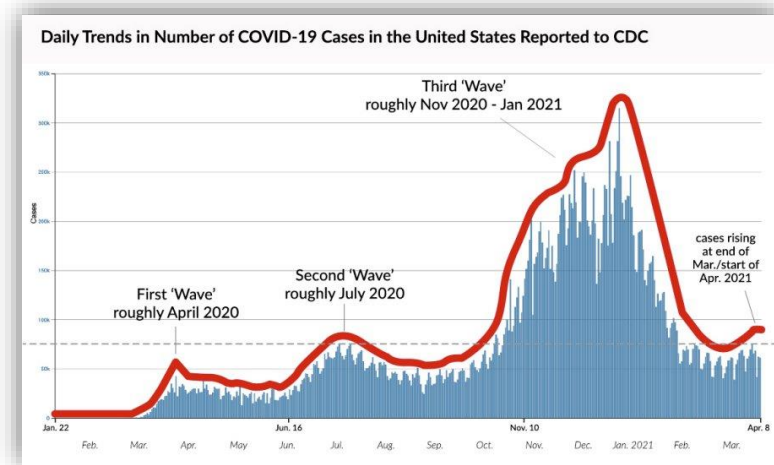


Figure 2.1. Covid Movement [40].



Figure 2.2. Bitcoin Movement [2].

After this interesting revelation, machine learning that could build a model based on previous trends and identify the correlations was explored. This could then allow the model to predict and advise on future investment. This model could also possibly inversely predict pandemic data such as incoming lockdowns as a result of a Bitcoin trend.

From research of machine learning and trend analysis, it was discovered that linear regression or neural networks would be suitable to use for trend prediction. These approaches were explained on sites such as upgrad.com [9]. The journal “Linear trend analysis: a comparison of methods” (Hess, Iyer and Malm, 2001) [55] evaluates statistical approaches of trend detection informing the author of other solutions available. Long Short-Term Memory Networks [17] also have become a potential candidate for model generation.

2.3 Literature Review

Blockchain

“A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in single list, called a chain. Blockchains are used for recording transactions made with cryptocurrencies, such as Bitcoin, and have many other applications.” (*Christensson (2018)*) [16]

The Blockchain is stored on a peer-to-peer network of independent computers. Each computer has a copy of the transaction ledger, and each transaction is verified using a cryptographic hash which means falsification is almost impossible [7]. This allows for a safe way to deal with currency and that is why Bitcoin and other cryptocurrencies were created. In the journal “Do you Need a Blockchain?” (*Wust and Gervais, 2018*) [57] the security measures are outlined of a centrally managed database that enables this ledger methodology.

When an individual record, or block is created it is linked to other blocks which results in a blockchain. This record is validated by many independent computers before it is added to the chain. The blocks are connected using a cryptographic hash created from the content of the adjacent blocks. This makes is very difficult to alter any blocks as all subsequent blocks must be altered too [13].

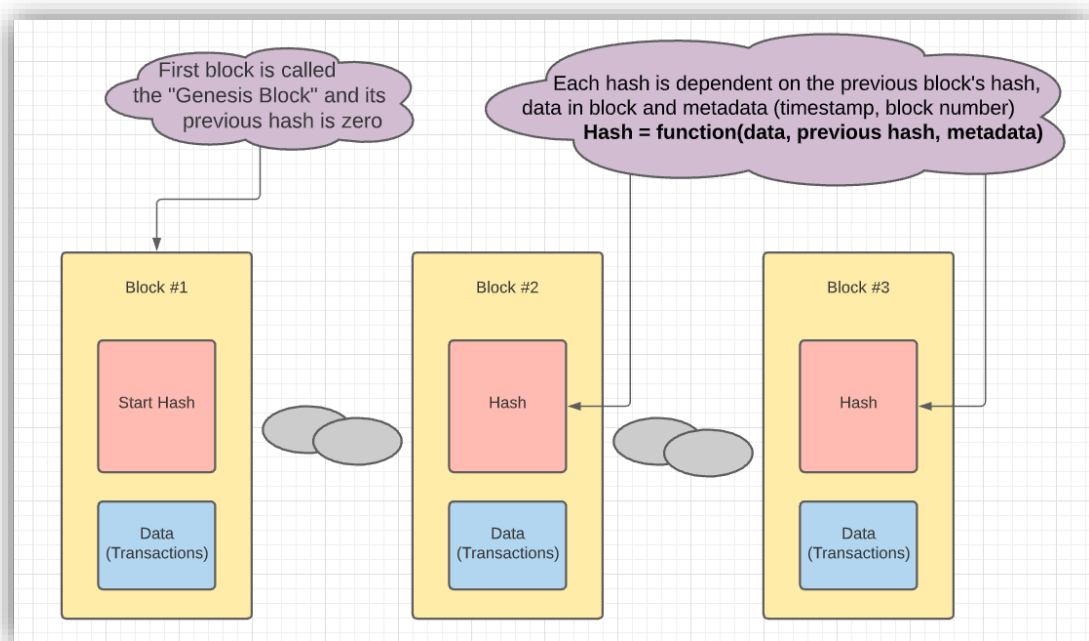


Figure 2.3. Blocks linked cryptographically through hashes.

To add a block to a chain you must go through a process called mining. This involves doing complex mathematics to find a suitable hash that begins with a specific number of zeros. Once you find the correct hash that block is considered forever signed unless

it is re-mined. The nodes or computers on the network accept this change in the ledger and update their ledgers, the miner then receives a financial reward [14].

If a change needs to be made to a block in the middle of a chain that block, and all subsequent blocks need to be re-mined to be approved by the network. This is very difficult to do as each hash has about four billion possibilities and requires very powerful computers.

Blockchain analytics involves the analysis of these transactions and how the ledgers are used. The analysis aims to discover useful information about who is using them and why. It is possible to map the spending history specific entities on the blockchain [15].

Cryptocurrency

A crypto currency is a form of digital asset based on a blockchain network. Given the decentralised nature of blockchain, cryptocurrencies exist with no governing body such as governments or banks. These currencies operate by using virtual tokens which are part of the ledger entries in a blockchain. Blockchain's complex mathematical verification system of entries protects these tokens from duplication or manipulation. Many different cryptocurrencies exist, many of them have different functions and specifications for a particular use case [24].

Cryptocurrency is considered to be like Fiat money which indicates that it does not have any intrinsic value and its value is relative to how effective of a medium of exchange it is [25]. The journal "Risks and Returns of Cryptocurrency" (*Liu and Tsyvinski, 2020*) [58] explores how this type of asset is view by the market and how and why its price fluctuates.

A Bitcoin wallet [41] is a software program for holding and trading Bitcoin, a wallet contains a private key which corresponds to the wallet address. Wallets enable the receiving and sending of Bitcoin and can be accessed on desktop, mobile, web and hardware.

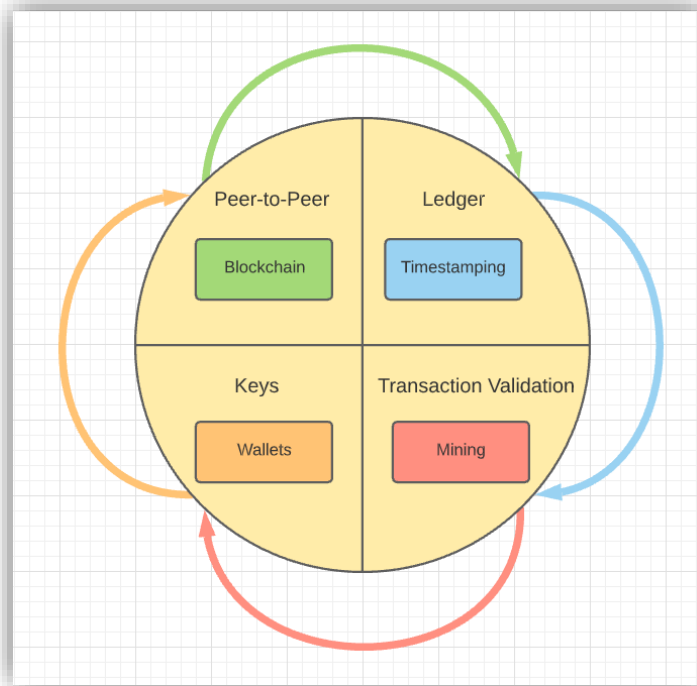


Figure 2.4. Cryptocurrency Architecture.

Some advantages of cryptocurrency are no restrictions on payment, anonymity, fast exchanges and free or cheap transactions. Disadvantages include value volatility, not accepted everywhere, complex operations, criminal activity, currency exchange issues and no payment reversals or recovery [26].

Bitcoin

The pioneer of cryptocurrency was Bitcoin, it was the first successful attempt out of hundreds to create virtual money using cryptography. Bitcoin's main purpose is to store value and enable a distributed payment system, it is primarily a list that stores transactions that contain entries such as Person A sent X bitcoin to person B. To maintain the transaction ledger, mining is used as seen above in the blockchain part of the literature review [27].

Bitcoin is the current leader of cryptocurrencies for a few reasons, it has been around for 8 years without failure thus making it a good store of value and it is the most accessible cryptocurrency with the most merchants accepting it and the most exchanges trading it making it the most liquid. Bitcoin also has the largest developer ecosystem with the most software and implementations available to interact with it [28].

The viability of Bitcoin as a retail currency is discussed in the journal "Is Bitcoin a Real Currency? An Economic Appraisal" (Yermack, 2015) [59]. It discusses issues

such as short term volatility and lack of insurance implementation which impacts the currencies functions and how it is viewed by the world.

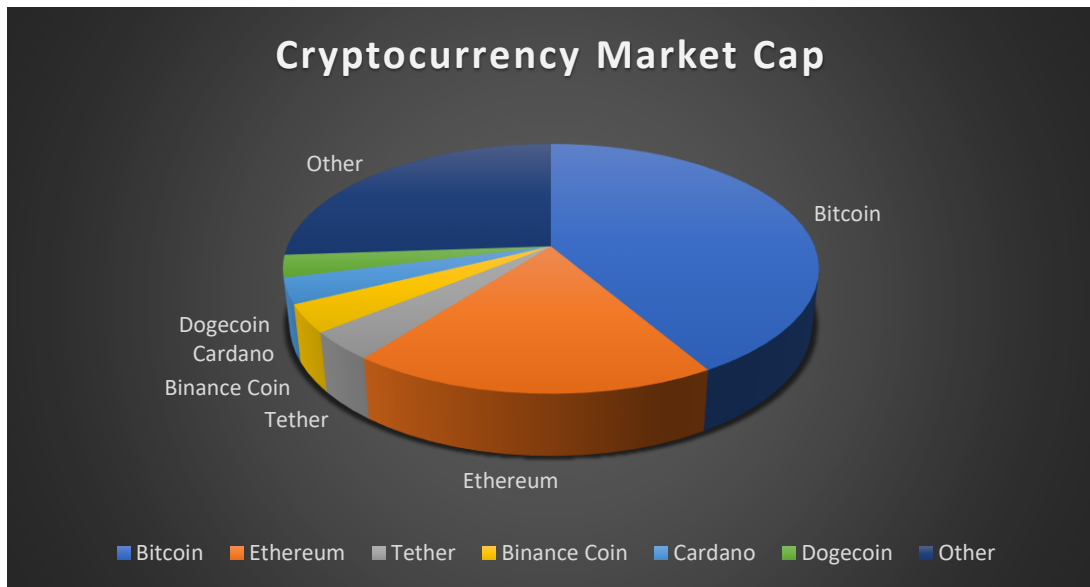


Figure 2.5. Cryptocurrency market cap leaders, visualized data from Slikcharts [42].

Project Topic Relevance

Blockchain is a very relevant topic as its expansion has some promising benefits but also some worrying issues. The increase in the use of blockchain has created a vast decentralised peer-to-peer network which has in turn spawned a liberated currency in its cryptocurrencies as no central authority controls the flow of them. This is a very secure and transparent system but allows anonymity when used in a certain way which has sparked issues with criminal activity [20].

With the current Covid-19 pandemic crippling the world and its markets [21], It would be quite interesting to see how the virtual world and its currencies are reacting to the changes currently taking place such as inflation and stock market crashes in the real world. Since Bitcoin is not affected by inflation or any governing bodies the movement of its prices is quite interesting.

Correlating Bitcoin with other trends and commodities would prove to be quite useful as this can help people make rational decisions in relation to using Bitcoin and investing in it. With tech giants such as Tesla and MasterCard introducing their industries to the commercial use of cryptocurrency [29], this format of exchange is inevitable to become mainstream in the future.

Whether it is to stay safe while using cryptocurrency or ensure your business is prepared for a payment revolution, the analysis of blockchain and Bitcoin is invaluable for anyone.

2.4 Existing Software

Existing Approaches

Bitcoin Analysis tools such as Coindesk [3], Kitco [22] and Business Insider [23] are existing approaches to analyse blockchain transactions. They share some features such as graph representation of price of the currency over a selected period, customisation of the data dashboard and currency snapshot displayed. The data is live and provides current up to date information.

These applications allow for fine tuning of the data displayed such as trends over a specific period such as day, week, or month. A metric section sits below the graph and gives detailed information such as value transacted, transaction count and average transaction fee.

The tools do not indicate any patterns present, predict, or suggest prime investing times. They also do not support any superimposing of other data onto the graph. These tools provide basic transaction data and prices but lack any information that can aid in interpreting it.

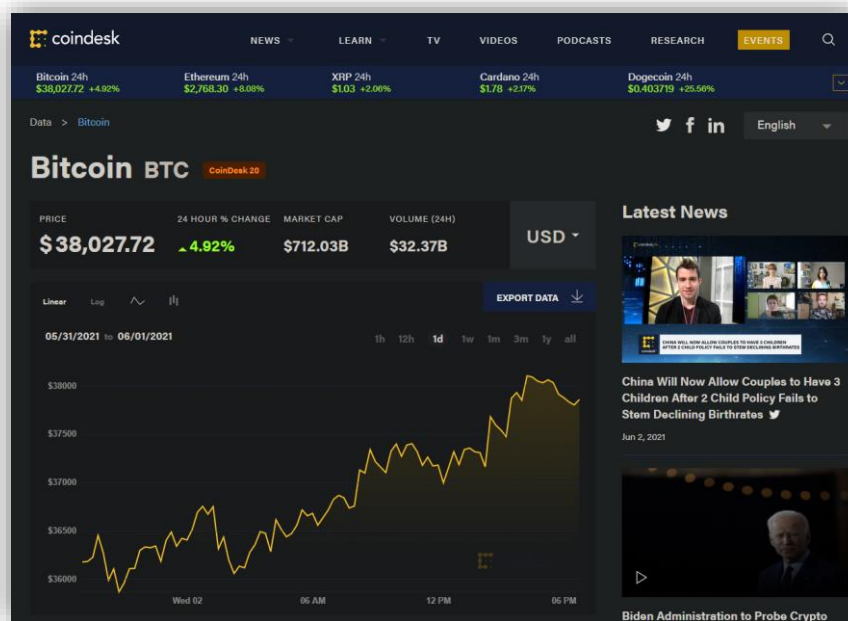


Figure 2.6. Existing approach – Coindesk [3].

Architecture

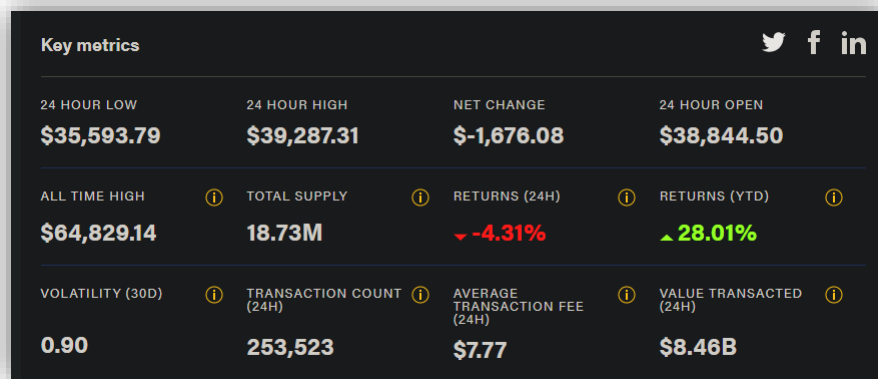
From analysis of these existing approaches, the author has garnered some basic architectural information of these applications. The applications all share these features but may implement them differently. The application are all web apps that consist of a user interface, a backend that makes requests for information such as

prices between dates and some form of a frontend service that can dynamically convert chart types client side and complete small tasks without making any requests.

Design

The design of existing approaches is quite similar. The landing page is dominated by the history chart of the selected currency, the user can toggle between different ways of displaying the data. On top of the chart rests a bar which displays the current price and the current percentage change in price over a certain period.

The colour palate consists of dull dark background colours contrasted by vibrant colours displaying the information such as yellow or red. The Design is not cluttered and simply displays the content with tabs linking other content that the user can navigate to. Colour is also used to indicate positive and negative movements making the design even more concise and effective in conveying the information to the user.



Key metrics				Twitter Facebook LinkedIn	
24 HOUR LOW	24 HOUR HIGH	NET CHANGE	24 HOUR OPEN		
\$35,593.79	\$39,287.31	\$-1,676.08	\$38,844.50		
ALL TIME HIGH	TOTAL SUPPLY	RETURNS (24H)	RETURNS (YTD)		
\$64,829.14	18.73M	▼ -4.31%	▲ 28.01%		
VOLATILITY (30D)	TRANSACTION COUNT (24H)	AVERAGE TRANSACTION FEE (24H)	VALUE TRANSACTED (24H)		
0.90	253,523	\$7.77	\$8.46B		

Figure 2.7. Metrics provided by Coindesk [3].

2.5 Innovative Approach

The project aims at building on current functionality of common Bitcoin analysis sites. The added functionality would include an overlay of trends to discover any potential correlations and the introduction of machine learning to build a model to predict trends in the world of Bitcoin.

Values such as total transactions and total value of transactions would be important to focus on along with the price of Bitcoin. These values would be juxtaposed with values such as Covid-19 cases. The algorithm would build a model with the data selected by the user and predict future trends based on current ones.

2.6 Conclusion

From the research conducted by the author, they garnered valuable information about the brief and existing approaches of the data analytics too that will be constructed. The literature review described the way a blockchain works and how it makes cryptocurrencies possible. This was followed by an analysis of already existing approaches and their features and how they are constructed.

Given this information an innovative design was created that took guidance on how to construct the application's architecture from already existing approaches. The feature set revolved around new innovative ideas that are relevant to the current uses of cryptocurrencies.

Chapter 3. Development

3.1 Introduction

This chapter will cover the steps taken to set out requirements, create a development plan, the modules that the application is made up of, unit testing and what tools, frameworks and libraries were utilized. Code snippets or images may be used to fully illustrate what each component is responsible for and how it is implemented.

3.2 Planning

Primary Goals

Primary goals of this project will be any vital functionality needed to create a viable innovative data analytics tool. This will be divided as follows.

A webapp that displays Bitcoin data as a graph and by other means will be the first primary goal. The webapp at this point needs to visually display the data and have options of displaying data between different time periods.

The second primary goal will be to allow the overlay of Covid-19 data on the original Bitcoin graph. It will be updated to show data between a specific time period similarly to the Bitcoin data.

The third primary goal will be the implementation of machine learning to indicate Bitcoin trend behaviours and predict the Bitcoin data. This will allow the program to advise the user on ideal investing and selling times.

Secondary Goals

The first secondary goal will be to allow the overlay of other data on the graph such as the US Dollar. This will allow the tool to compare the Bitcoin and trends with different values and movement patterns.

The secondary goal is to make the machine learning component more dynamic so that the user can select the data to be used to train a model to predict trends.

The third secondary goal will be to allow the user to input their past investments into the program and it will show its predicted price at a future time of their choosing. It will also show the percentage gain or loss of their investment.

#	Planned Steps
1.	Research Blockchain
2.	Decide on project
3.	Identify Tools and Technologies
4.	Create Project Definition Document
5.	Set up GitHub Repository
6.	Create Webapp Skeleton
7.	Create UI Skeleton
8.	Connect Bitcoin API
9.	Connect Covid-19 API
10.	Display Basic Graph and data
11.	Connect Machine Learning API
12.	Set up Machine Learning Algorithm
13.	Format data and send to Machine Learning API
14.	Add Investment Calculator/Predictor UI
15.	Add Ideal time to Buy/Sell Indicator
16.	Add Customisation Currency/data displayed
17.	Adjust functionality to take in the new data

Table 3.1. Planned steps

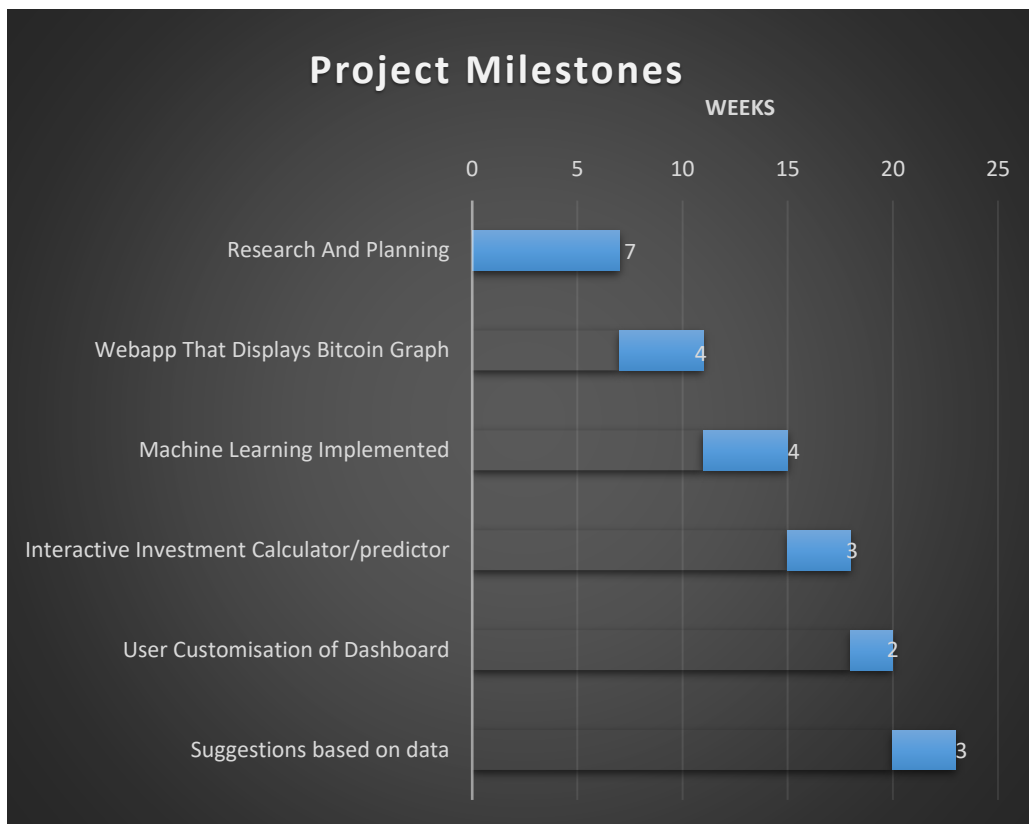


Figure 3.1. Project milestone Gantt chart.

Envisioned Design – User interface

The envisioned design for the user interface includes a large history graph with the current price displayed below along with the price action indicated with colour. A section is allocated to suggest to the user what the best investment move would be and a calculator to help the user evaluate the current price of their previous investments. The graph is multi axis allowing differing valued commodities to be compared accurately in terms of percentage movement.

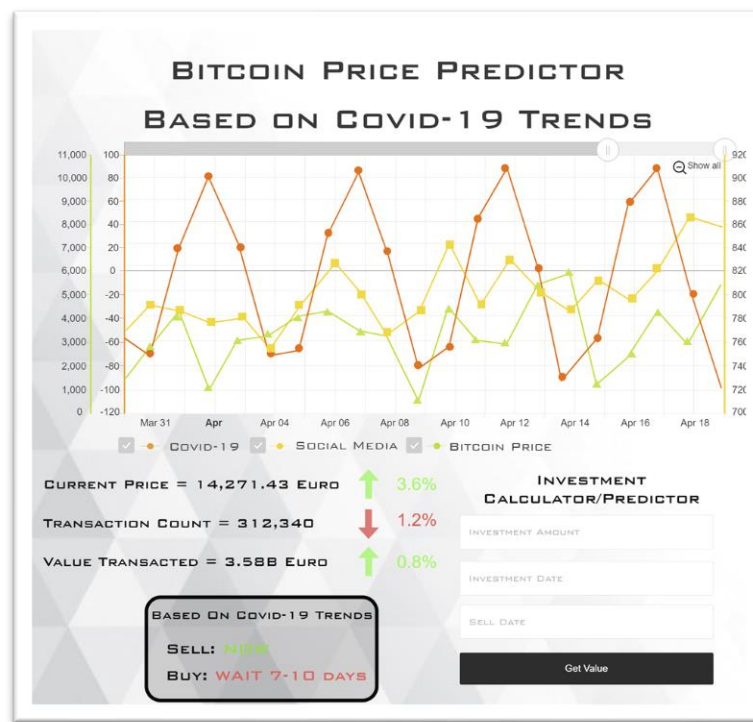


Figure 3.2. Envisioned user interface.

Planned Frontend and UI software

The UI will be built using the Vaadin [6] framework which allows the building of frontend components using Java [31]. This will allow for a cleaner and faster webapp as one language can be used.

The UI will consist of a central graph with components such as buttons that support interaction. The UI will allow the user to also view data in a numerical way. An input area will be available for the user to input past investments to the show the current or future value of it along with the profit or loss experienced.

Custom CSS [30] or cascading style sheet scripts will be used to refine the design colours and to access the different libraries Vaadin wraps to enable creation of elements in Java. This will allow the details of Vaadin created user interface components to be changed.

Planned Backend Software

The first component to complete will be a Java webapp that will serve as the base of the project. It will be the hub of the project as all the APIs [32] and the user interface will be accessed from here. A Maven [10] webapp will be implemented that employs the Spring Boot [5] framework.

The logic of the webapp will be based here, this will include API input conversion for machine learning purposes, preparation of data to be displayed for the frontend and any computation required to calculate values such as future value of specific investments. The backend will also connect to any APIs necessary to run the app such as the Bitcoin API [2].

Architecture Overview

The Application will be run in a Docker [33] container that will interact with the internet to make requests to any necessary APIs included in the final design. The Spring Framework will manage the front end and backend components also interacting with any external packages present.

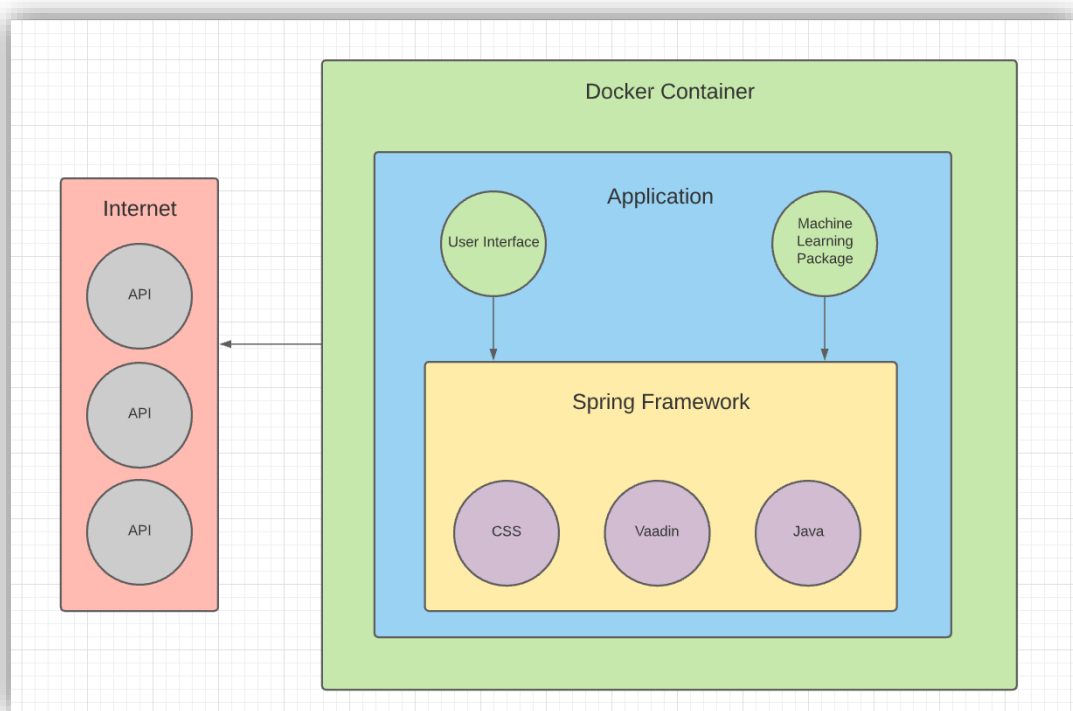


Figure 3.3. Architecture flow diagram.

3.3 Tools

Build Automation – Maven

Maven [10], a software project management and comprehension tool that is used to build the project, manage its dependencies and documentation. It is based on the project object model (POM) which uses XML files that contain the project's configuration, plugins, dependencies etc [34]. Below is an example of the output Maven produces when using a Maven command to build the project with the production configuration “mvn clean package -Pproduction”.

```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 46.762 s
[INFO] Finished at: 2021-06-03T15:29:44+01:00
[INFO] -----
```

Figure 3.4. Maven utilisation.

Task Management – Asana

A task managing tool called Asana [12] is used to set out tasks with due dates. This will keep the project on track and ensure that all primary goals and as many secondary ones as possible are reached. It facilitates the use of many workflows such as Kanban, which aims to visualise the work and maximise efficiency. It is a very prevalent technique used in agile workflows [35] and is used during the development of this project.

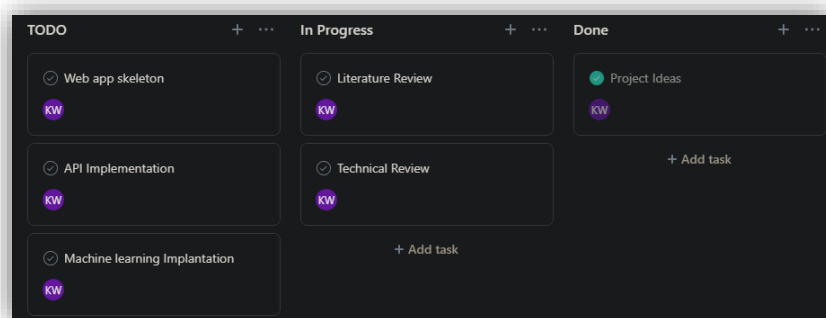


Figure 3.5. Task Management Tool.

Version Control – GIT

The project will utilise GitHub [11] to backup all project files and to keep a record of all code base changes. This will also aid in the recording of goals and milestones as they are reached. It allows the branching of code to enable changes to the codebase while having a copy of the code unchanged, if corruption occurs of your branch, you can delete it and create a new one with the saved version. Version control also enables the rollback of code to a previous state if changes negatively affect the project. Commits are made to the remote repository onto the develop branch regularly to ensure no progress is lost.

```
commit 5326c527204b6466c53a290022f3c7482eb4c9bd
Author: Jakub Wojtkowicz <jakub.wojtkowicz@gmail.com>
Date: Sat May 8 01:48:55 2021 +0100

    layout adjustments

commit 096658e0a56d5a3c6cdd21688751e7adce24e44c
Author: Jakub Wojtkowicz <jakub.wojtkowicz@gmail.com>
Date: Fri May 7 19:56:33 2021 +0100

    added trends, changed data names to show unit
```

Figure 3.6. Git commit regularity.

Containerisation – Docker

Docker [33] is a platform that enables the building and deploying of containerised applications. These applications are independent once containerized and can be run on many systems easily. A Docker image is created containing the application and any necessary components it needs to operate and can be distributed to others wanting to run the application. The user can then use a Docker container to run this image.

Docker was used in this project to enable hosting of the application easily and remotely. A server or local user can run this image with ease making the application rapidly available. Kubernetes [36] which is a system for automating deployment, can be utilised to scale this application. An Image of the application can be used to run multiple instances of the application to scale the service to linearly improve the request speeds.

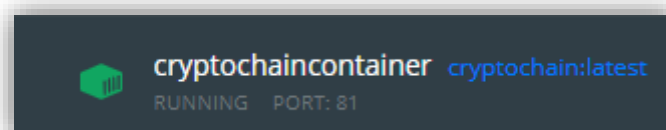


Figure 3.7. Application running locally in a Docker container.

IDE – IntelliJ

IntelliJ [37] is an integrated development environment (IDE) which is used for developing software. It enables the editing and compiling of code along with many other features.

This IDE is used for this project as it is integrated with GitHub which enables version control directly in the editor. This helps with tasks such as comparing code changes to the parent branch and resolving merge conflicts, it can also directly commit, pull, and push changes to the remote repository.

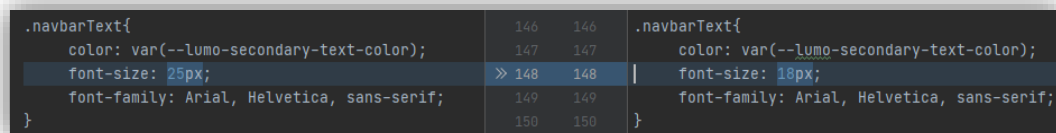


Figure 3.8. IDE version control integration.

Terminal Application - Git Bash

Git Bash [38] is an application for windows that emulates the original git version control system that was built for Unix environments. It supports the necessary Git command line commands to use the git version control system. Git Bash supports the Bash shell and necessary Bash utilities.

Git Bash is used in this project for version control, all repository manipulation is done using this application including branch creation, branch merging, committing changes, pushing changes, pulling repository updates etc.

```
jakub@DESKTOP-4N0ICVL MINGW64 ~/Documents/Final-Year-Project/blockchain-data-analytics-tool (develop)
$ git add .

jakub@DESKTOP-4N0ICVL MINGW64 ~/Documents/Final-Year-Project/blockchain-data-analytics-tool (develop)
$ git commit -m"UI font change and initial trend display name st"
[develop 768c494] UI font change and initial trend display name st
2 files changed, 2 insertions(+), 4 deletions(-)

jakub@DESKTOP-4N0ICVL MINGW64 ~/Documents/Final-Year-Project/blockchain-data-analytics-tool (develop)
$ git push
Enumerating objects: 42, done.
Counting objects: 100% (42/42), done.
Delta compression using up to 16 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (27/27), 1.81 KiB | 617.00 KiB/s, done.
Total 27 (delta 14), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (14/14), completed with 9 local objects.
To https://github.com/Wojtkowicz/Final-Year-Project.git
5e2c87d..768c494 develop -> develop

jakub@DESKTOP-4N0ICVL MINGW64 ~/Documents/Final-Year-Project/blockchain-data-analytics-tool (develop)
$ git status
On branch develop
Your branch is up to date with 'origin/develop'.
```

Figure 3.9. Git Bash used for updating remote repository.

3.4 Frameworks

Application Development – Spring

The Spring framework is an application development framework, it provides comprehensive infrastructure support for developing Java applications. Spring Boot simplifies the task of creating stand-alone quality applications.

Spring Initializr [39] is used to create a skeleton application. It creates all the necessary modules needed to have a runnable application. It enables the user to input package and group names, select dependencies and indicate what type of project it is, in this case Maven was selected.

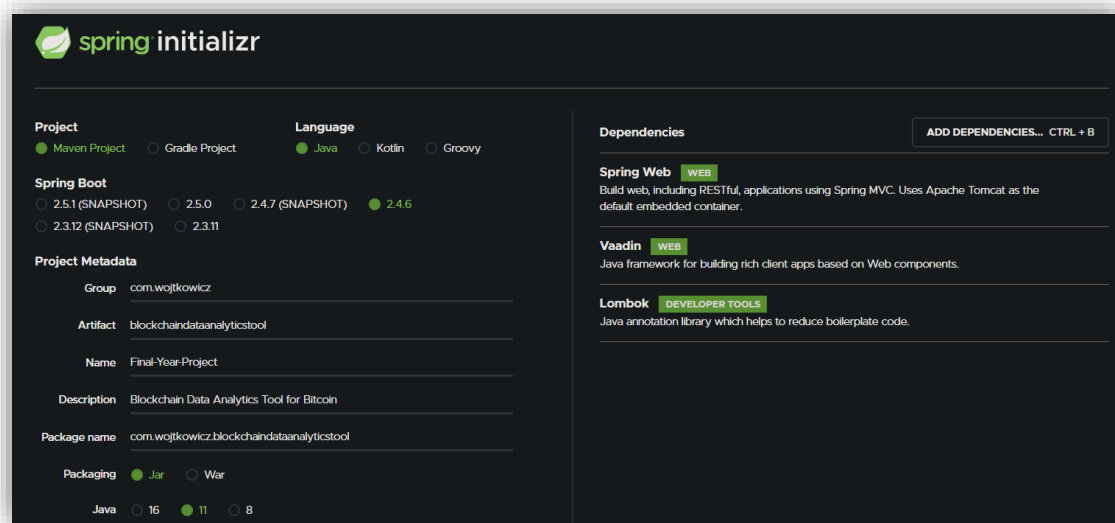
The screenshot shows the Spring Initializr web application interface. It has a dark theme. At the top left is the 'spring initializr' logo. The interface is divided into several sections: 'Project' with radio buttons for 'Maven Project' (selected) and 'Gradle Project'; 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for versions 2.5.1 (SNAPSHOT), 2.5.0, 2.4.7 (SNAPSHOT), 2.4.6 (selected), 2.3.12 (SNAPSHOT), and 2.3.11; 'Project Metadata' with input fields for 'Group' (com.wojtkowicz), 'Artifact' (blockchaindataanalyticstool), 'Name' (Final-Year-Project), 'Description' (Blockchain Data Analytics Tool for Bitcoin), and 'Package name' (com.wojtkowicz.blockchaindataanalyticstool); 'Packaging' with radio buttons for 'Jar' (selected) and 'War'; and 'Java' with radio buttons for versions 16, 11 (selected), and 8. On the right side, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... CTRL + B'. Below this, there are three dependency cards: 'Spring Web' (WEB) with a description 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'; 'Vaadin' (WEB) with a description 'Java framework for building rich client apps based on Web components.'; and 'Lombok' (DEVELOPER TOOLS) with a description 'Java annotation library which helps to reduce boilerplate code.'

Figure 3.10. Spring initializer skeleton application.

User Interface Development – Vaadin

The Vaadin [6] framework provides user interface components constructed from HTML5 that are represented as Java objects. This allows the use of the Java programming language to manipulate and create full user interfaces. Vaadin provides a wrapper around JavaScript libraries such as Highcharts [43] that create charts and graphs.

Since the charts or any other UI elements are now objects, the author can create, populate, manipulate, and remove these elements using Java. This gives great options when creating or manipulating elements dynamically given an end user triggered event.

The ability to create your own components enables modularity of the user interface in turn making cleaner and more reliable code when generating views. Graph generation can be assigned to functions that can run dynamically with arguments to indicate user made changes.


```

public static Chart generateHistoryBitcoinGraphCandleStick(){
    final Chart chart = new Chart(ChartType.CANDLESTICK);
    chart.setTimeline(true);

    Configuration configuration = chart.getConfiguration();
    configuration.getTitle().setText("Bitcoin BTC Price");
    configuration.setSubTitle("Candlestick Chart Showing Daily Trading Data");
    configuration.getTooltip().setEnabled(true);

    PlotOptionsCandlestick plotOptionsCandlestick = new PlotOptionsCandlestick();
    DataGrouping grouping = new DataGrouping();
    grouping.addUnit(new TimeUnitMultiples(TimeUnit.WEEK, ...allowedMultiples: 1));
    grouping.addUnit(new TimeUnitMultiples(TimeUnit.MONTH, ...allowedMultiples: 1, 2, 3, 4, 6));
    plotOptionsCandlestick.setDataGrouping(grouping);
    bitcoinDetailedSeries.setPlotOptions(plotOptionsCandlestick);

    configuration.setSeries(bitcoinDetailedSeries);

    RangeSelector rangeSelector = new RangeSelector();
    rangeSelector.setSelected(4);
    configuration.setRangeSelector(rangeSelector);

    chart.setTimeline(true);
    chart.setHeight("780px");

    return chart;
}

```

Figure 3.11. Vaadin chart generation.

```

candlestickButton.addClickListener(click -> {
    history.removeAll();
    history.add(historyGraphCandlestick);
    techAnalysisButtonsArray.stream().forEach(b -> b.setEnabled(true));
    click.getSource().setEnabled(false);
});

```

Figure 3.12. Vaadin components as Java objects.

Vaadin only provides a wrapper for JavaScript and HTML5 libraries and components, this approach negates the ability to customise the CSS of the created components and leaves the colours and look of the component much to be desired. To remedy this the author directly accessed the CSS of the underlying library components that Vaadin has wrapped. This enabled fine grained editing of the look and feel of each component individually.

```

).highcharts-candlestick-series .highcharts-point {
  stroke: #ff6666;
  fill: #ff6666;
}

```

Figure 3.13. Vaadin underlying library CSS.



Figure 3.14. Standard chart CSS.



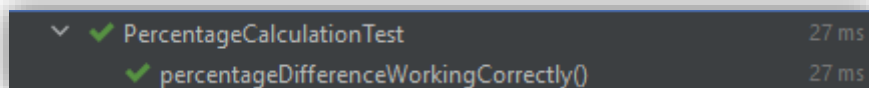
Figure 3.15. Modified CSS.

Developer Side Unit Testing – JUnit

JUnit [44] is a unit testing framework for the Java programming language. JUnit is integrated with the project via the project POM as a dependency. Tests can be run in the IDE and by Maven when the project is being built. These tests use static values so that if a change negatively effects the function of the application the result will change, and the test will fail alerting whoever is maintaining the application that the change needs to be rolled back and investigated into why it had a negative effect on the application's processes.

```
@Test
public void percentageDifferenceWorkingCorrectly(){
    // Setup
    Double pastPrice = 123.0;
    Double currentPrice = 140.0;
    Double expectedResult = 13.82;
    // Execute
    Double result = calculatePercentageDifference(pastPrice, currentPrice);
    // Verify
    assertEquals(expectedResult, result);
}
```

Figure 3.16. JUnit test.

A screenshot of a JUnit test completion window. It shows a tree view with a green checkmark next to 'PercentageCalculationTest' and a sub-entry 'percentageDifferenceWorkingCorrectly()' also with a green checkmark. The execution time for both is listed as '27 ms' on the right.

✓ PercentageCalculationTest	27 ms
✓ percentageDifferenceWorkingCorrectly()	27 ms

Figure 3.17. JUnit test completion

Mocking – Mockito

Mockito [45] is a mocking framework for unit tests in Java. It enables the duplication of objects into “mock” objects that are not the real object and can be manipulated in the testing environment. In this project Mockito was used for various tests, in the example below the API connections are being tested.

A mock RestTemplate is created so that when used in a test it will not make a request to the API but return a pre-defined result that can be verified at the end of the test. This ensures the flow of the method is verified correctly using static information, eliminating any API issues from the test.

```
@Mock
private RestTemplate restTemplateMock;
```

Figure 3.18. Mockito @Mock annotation.

```
Mockito.when(restTemplateMock.exchange(
    eq( value: "https://api.coindesk.com/v1/bpi/historical/close.json?start=2021-04-05&end=2021-06-05"),
    eq(HttpMethod.GET),
    any(),
    eq(String.class)))
    .thenReturn(apiResponse);
```

Figure 3.19. Mockito when, then return flow.

3.5 APIs

An API [32] is an acronym for Application Programming Interface, it is a software intermediary that allows communication between applications [46]. In this project APIs were used for data retrieval. Many API endpoints and sources were used to fully enable this application to produce quality results and content.

Requests

The application has a connection class that deals with all these connections. Each connection has a function that is called when the application needs data, it assembles the URL needed with data from the constants class such as keys and URIs or uniform resource identifier for that request and uses a municipal function that takes the created URL or site address and makes the request. The response is then returned by this utility function and then sent to be mapped. The mapped response is returned to the main API function and then to the application from where it was originally requested.

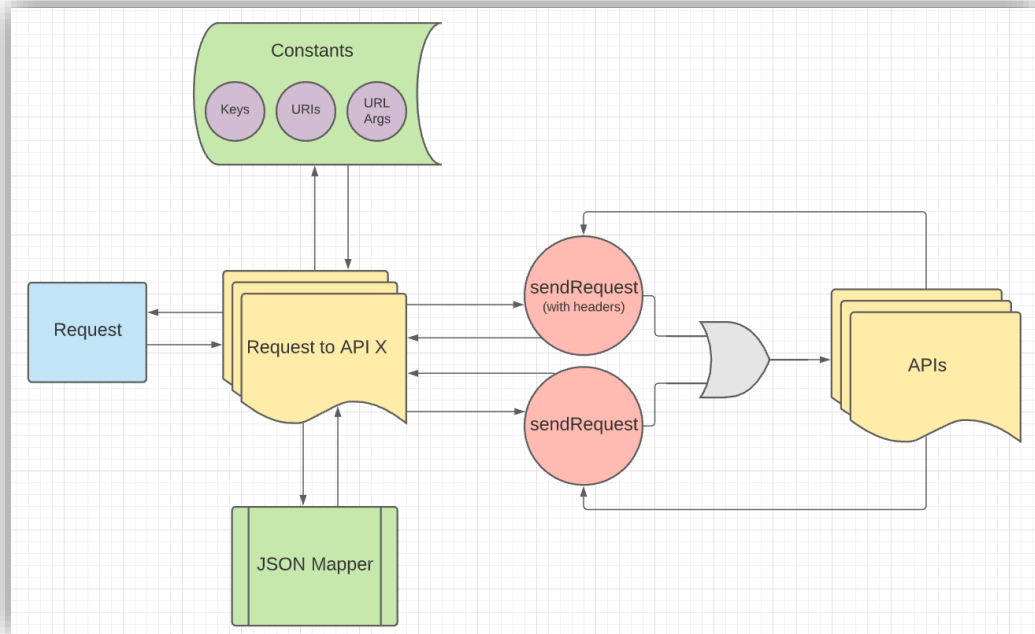


Figure 3.20. API request flow.

```

public static String sendRequest(String url) {
    final HttpEntity<String> entity = new HttpEntity<>(" ");
    RestTemplate restTemplate = new RestTemplate();
    ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.GET, entity, String.class);
    return response.getBody();
}

```

Figure 3.21. Send API request.

Some of the APIs required an API key and had restricted access. An Account was set up that monitored the number of requests the author was making. This caused the application's live updating price and graph to have its update time extended to two minutes from a few seconds as the application quickly surpassed the request limit and lost its steady information stream.

To assemble a URL, a URL builder is employed. It takes an input of the site URI and uses the function "addParameters" to add value pairs to be appended to the URL to indicate any API parameters for the requested data such as currency and the API key.

```

public static HashMap<String, String> getDogeHistoryData() {
    try {
        // Create parameters for URL
        List<NameValuePair> parameters = new ArrayList<>();
        parameters.add(new BasicNameValuePair( name: "fsym", value: "DOGE"));
        parameters.add(new BasicNameValuePair( name: "tsym", value: "USD"));
        parameters.add(new BasicNameValuePair( name: "allData", value: "true"));
        parameters.add(new BasicNameValuePair( name: "api_key", CRYPTO_COMPARE_API_KEY));
        // Create URL with URI and parameters
        URIBuilder url = new URIBuilder(CRYPTO_COMPARE_HISTORY_PRICE_URI);
        url.addParameters(parameters);
        // Create a rest template and send request with url
        String apiResponse = sendRequest(url.toString());
        // Return mapped output with dates and prices
        return Mapper.jsonEthereumPastDataToMap(apiResponse);
    } catch (Exception e) {
        System.out.println("ERROR: cryptocompare API error, getting doge history data");
    }
    // Return empty data if error occurs
    return new HashMap<String, String>();
}

```

Figure 3.22. API request generation.

Integrated APIs

#	Location	Data	Request Limit	Key
1	Coindesk	Bitcoin data between two dates	NO	NO
2	Coindesk	Current Bitcoin data	NO	NO
3	Crypto Compare	Bitcoin candlestick history data	YES	YES
4	Crypto Compare	Bitcoin candlestick current data	YES	YES
5	Crypto Compare	Ethereum history data	YES	YES
6	Crypto Compare	XRP history data	YES	YES
7	Crypto Compare	Doge history data	YES	YES
8	Crypto Compare	GBP history data	YES	YES
9	Crypto Compare	EUR history data	YES	YES
10	Rapid API	Covid-19	YES	YES
11	Rapid API	S&P500	YES	YES

Table 3.2. APIs Integrated into application.

URL Testing – Postman

To receive the necessary data from an API a URL needs to be constructed correctly, to easily test these requests with different parameters Postman [47] an API

development tool is utilised. It allows for the verification of specific URLs and displays the received payload from the API.

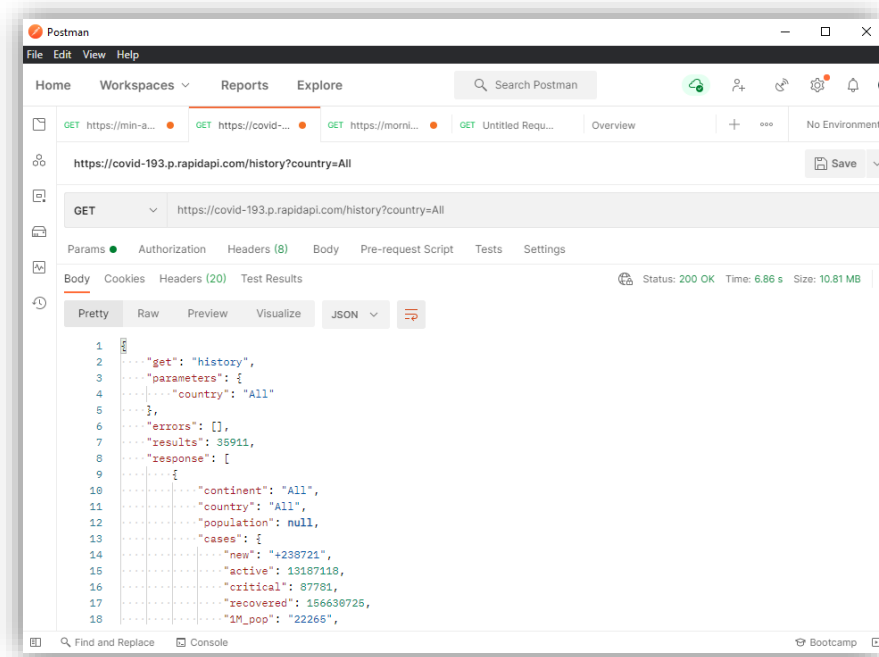


Figure 3.23. Postman API development.

JSON Mappers

The data received in the body of the API response is in the format of JSON [48] or JavaScript Object Notation. This format is commonly used for transmitting data in web applications. This data is unmarshalled and turned into a Java object. In the case of the APIs used in the project, the data was turned in a linked hash map because the data was sequential, and the type of data was time series, so a date was affixed and pointed to a price.

A mapper class is used in this application with unique methods to map each type of data to a uniquely built linked hash map. The JSON is first turned into a hash map that is then iterated through using Java streams to sequentially construct a linked hash map of defined structure. Some of the more complex data sets such as the candlestick data had many values linked to a single date which required more complex mappers which had nested linked hash maps within them.

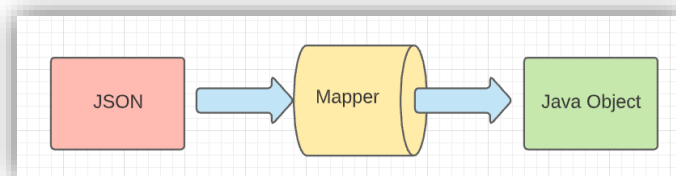


Figure 3.24. Unmarshalling JSON.

```

public static LinkedHashMap<String, String> jsonBitcoinPastDataToMap(String jsonApiResponse) {
    try {
        // Use mapper to turn json string to hashmap
        ObjectMapper mapper = new ObjectMapper();
        LinkedHashMap<String, LinkedHashMap<String, Object>> inputMap = mapper.readValue(jsonApiResponse, LinkedHashMap.class);
        // Create an output hashmap of key String and value String that represent the date and value
        LinkedHashMap<String, String> outputMap = new LinkedHashMap<>();
        // Iterate over each key and value in the bpi hashmap to dates and prices
        inputMap.get("bpi").keySet().stream().forEach(k -> {
            String key = k;
            String value = inputMap.get("bpi").get(k).toString();
            outputMap.put(key, value);
        });
        return outputMap;
    } catch (Exception e) {
        System.out.println("ERROR: mapper error, bitcoin history data");
    }
    // Return empty data if error occurs
    return null;
}

```

Figure 3.25. JSON mapper.

3.6 Languages

Frontend – CSS

The Frontend of the application was constructed using Vaadin’s component library, to edit the visual aesthetics of these components CSS [30] scripts were utilised. As mentioned earlier in the chapter, the underlying libraries Vaadin uses were altered via CSS that did not affect the operation of the components.

The Views that are displayed are built around a main layout that imports any necessary imports such as the CSS files and icons. This main view navigates to the views that are traversed to by the user. All the views inherit these imports and allow the two CSS files and imports to be shared among all displayed pages.

The CSS is responsible for things such as page padding, element position, background colour, element colour and size, text colour and size, graph colour and size etc. Dynamic components that require to change colour given a condition have multiple CSS components that are used and alternated by assigning the element to be coloured to a different CSS name given an event.


```

if(percentage > 0) {
    priceMovement.removeAll();
    priceMovement.add(new Text("+" + percentage + "%"));
    priceMovement.setClassName("priceMovementGreen");
}else{
    priceMovement.removeAll();
    priceMovement.add(new Text(percentage + "%"));
    priceMovement.setClassName("priceMovementRed");
}

```

Figure 3.26. CSS component assignment.

```

.priceMovementGreen {
    color: #99ff99;
    font-size: 35px;
    font-family: Arial, Helvetica, sans-serif;
}

.priceMovementRed {
    color: #ff6666;
    font-size: 35px;
    font-family: Arial, Helvetica, sans-serif;
}

```

Figure 3.27. CSS component definition.

Backend – Java

Java [31] is a programming language that is high-level, class-based and object oriented. It is used for most of this application as the backend of the application is built using this language and a Java accessible UI solution is implemented for the frontend. Java is quite universal and deals with all API connections and connection mappers in this application.

Command Line - Bash

Bash [49] is a Unix Shell and command language that is a command processor. In this project Bash was used for building the project, running project tests and version control etc.

```

[INFO] Building jar: C:\Users\jakub\Documents\Final-Year-Project\blockchain-da
ta-analytics-tool\target\blockchain-data-analytics-tool-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot-maven-plugin:2.4.2:repackage (repackage) @ blockchain-d
ata-analytics-tool ---
[INFO] Replacing main artifact with repackaged archive
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 49.521 s
[INFO] Finished at: 2021-06-06T07:22:25+01:00
[INFO] -----
jakub@DESKTOP-4N0ICVL MINGW64 ~/Documents/Final-Year-Project/blockchain-data-a
nalytics-tool (develop)
$

```

Figure 3.28. Building Project with Bash.

3.7 Libraries

Code Optimization - Lombok

The Lombok [50] library is a Java library that removes boiler plate code that would be repeated throughout the project such as getters and setters and replaces them with annotations. It reduces the amount of infrastructural code making Lombok generate this in the background when needed.

```

@Data
public class Alerts {

```

Figure 3.29. Lombok annotation used for getters and setters.

Time - Joda Time

The Joda Time [51] library provides date and time processing. It enables this project to easily access the current date time or get a previous or future date using its intuitive methods. In this project date and time are used when selecting dates by the user to select bounds for data to be displayed and for the predictions that require the time frame to which from extrapolate training data.

```
LocalDate now = LocalDate.now();  
LocalDate lastMonth = LocalDate.now().minusMonths(1);  
datePickerFrom.setValue(lastMonth);  
datePickerTo.setValue(now.minusDays(1));
```

Figure 3.30. Joda Time date creation and use.

Machine Learning – Weka

Weka [52] is machine learning software and consists of a collection of machine learning algorithms for data mining tasks. In this project Weka is imported as a dependency and uses a Time forecasting algorithm to predict time series data that is used in the project to represent prices on a given day.

This algorithm takes a CSV file as input with the necessary annotations, each column represents a different type of data such as highest price or volume of a given day. Each row represents a specific day and is identified by its date in the date column.

This file is generated every time a prediction request is made and will consist of the date and price as columns and then any other columns of data indicated by the user. The length of the file is also user defined as the user indicates how many days or rows to use and at what date the data should start and end. The file is deleted after use to ensure no reduction in speed of the service.

The output is the predicted-on price that points to its date, this means that if the data selected to train on is from the 5th of May to the 20th of May and the prediction is for five days, the date assigned to the first output will be the 21st of May and increment for the other four subsequent days.

```

private static File generateDataFile(Integer startDateIndex,
                                     Integer endDateIndex,
                                     ArrayList<ArrayList<String>> data,
                                     CheckboxGroup<String> selectedVariables) throws IOException {

    File file = new File( pathname: "inputData");
    FileWriter fw = new FileWriter(file, append: false);
    BufferedWriter bw = new BufferedWriter(fw);
    fw.write( str: "");
    bw.write( str: "@relation PricePrediction\n" + "\n");
    bw.write( str: "@attribute Date date 'yyyy-MM-dd'\n");
    bw.write( str: "@attribute close numeric\n");

    for (String s : selectedVariables.getSelectedItems()) {
        bw.write( str: "@attribute " + s + " numeric\n");
    }

    bw.write( str: "\n" + "@data");

    for(int i=startDateIndex; i<=endDateIndex; i++){
        bw.newLine();
        String dataLine = "";
        dataLine += stringToVariableReference( name: "date", data, i);
        dataLine += COMMA;
        dataLine += stringToVariableReference( name: "close", data, i);
        dataLine += COMMA;

        for (String s : selectedVariables.getSelectedItems()) {
            dataLine += stringToVariableReference(s, data, i);
            dataLine += COMMA;
        }
        bw.write(dataLine);
    }
    bw.close();
    return file;
}

```

Figure 3.31. Machine learning autonomous CSV generation.

3.8 Error Handling

Identification

While developing the application, the author had discovered many situations and edge cases where an error could be thrown. These were caused by such things as API connection issues, Machine learning algorithm failure, JSON mapping errors, UI component generation issues, CSS reference issues etc.

These errors are caught at the source and a console log detailing the issue is printed. Downstream an alert will be triggered for the user displaying a simple message of the error and to try again.

```

} catch (Exception e) {
    System.out.println("ERROR: cryptocompare API error, getting ethereum history data");
}

```

Figure 3.32. Exception source caught and console log printed.

```

} catch (Exception e) {
    e.printStackTrace();
    ALERTS.getCorrelationErrorAlert().open();
}

```

Figure 3.33. Resulting downstream error due to lack of adequate return object triggers UI error alert.

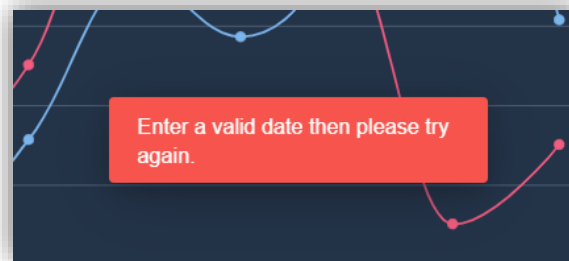


Figure 3.34. UI error displayed.

Prevention

To attempt to prevent errors, the data from inputs and other sources has bounds. This means that the dropdown options do not allow for date selection that is invalid, a prediction test data range too be too big or for the days to forecast into the future to be too big. Such bounds are coupled by impossible values such as negative day values or dates to compare trends before they existed. These edge cases have been identified and prevented.

For input values such as prices, maximum values are set so that a stack overflow does not occur, and the process will not run out of memory computing a result. Minimum values are set to zero.

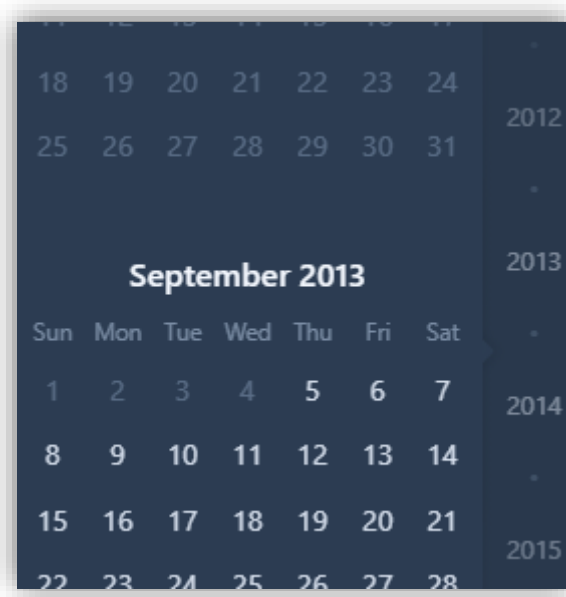


Figure 3.35. Invalid dates excluded as Bitcoin data can not be fetched for dates before Bitcoin existed.

Challenges

Some of the error prevention was challenging as it needed to be dynamic, for instance to compare two trends the minimum date will be set differently for each comparison as the shortest trend's minimum date will need to be used. This is because if the example of Bitcoin and Covid-19 is taken, Bitcoin's data will go back all the way to 2013 where as Covid-19 only has data since 2020. This means that the minimum date bounds of the user interface will need to be set to Covid-19's for this example.

Maximum date setting also followed this trend, but the opposite occurs, the lowest maximum date is selected for the bounds so that data of a day is not queried if it does not exist. This occurs because some APIs update the current day's data all day and the application has access to the current data whereas some are only set when the day is over resulting in a null value for the current day value.

This logic for setting bounds is also used for the selection of data from a trend for a comparison and will be covered in a later chapter.

Alerts

The user interface displayed alerts are Vaadin components. A class is created that generates all the alerts to be displayed on application start to reduce latency. A single method is used to create the alerts and it takes in a String containing the message to be displayed and returns a Notification type. As seen above, an Alert can be triggered to open for a few seconds using the open() extension then it closes automatically. The

alerts can be accessed anywhere from the application via the created static ALERTS instance that is located in the application constants.

```
public Alerts(){
    invalidDateInput = createAlert("Enter a valid date then please try again.");
    invalidDaysInput = createAlert("Enter a valid amount of days to forecast then please try again.");
    machineLearningPredictionErrorAlert = createAlert("Prediction error occurred, please try again.");
    invalidDateCombinationInput = createAlert("Start date must be before end date, please try again.");
    correlationErrorAlert = createAlert("Correlation error occurred, please try again.");
    invalidAmountInput = createAlert("Enter a valid amount then please try again.");
}

private Notification createAlert(String message){
    //Error notification
    Span content = new Span(message);
    Notification notification = new Notification(content);
    notification.addThemeVariants(NotificationVariant.LUMO_ERROR);
    content.setClassName("error");
    notification.setDuration(3000);
    notification.setPosition(Notification.Position.MIDDLE);
    return notification;
}
```

Figure 3.36. Alert generation.

3.9 Conclusion

This chapter covered the development aspects of the project such as what tools, libraries, languages etc were used during the completion of the project. As seen above many technologies were required to complete this project and the resulting application will be showcased in the next chapter.

Chapter 4. The Application – CryptoChain

4.1 Introduction

This chapter will cover the complete application, its features and how it functions. The Sections will individually focus on the application architecture, the routes or views the user can navigate to, their contents and the possible actions a user can take.

4.2 Application Architecture

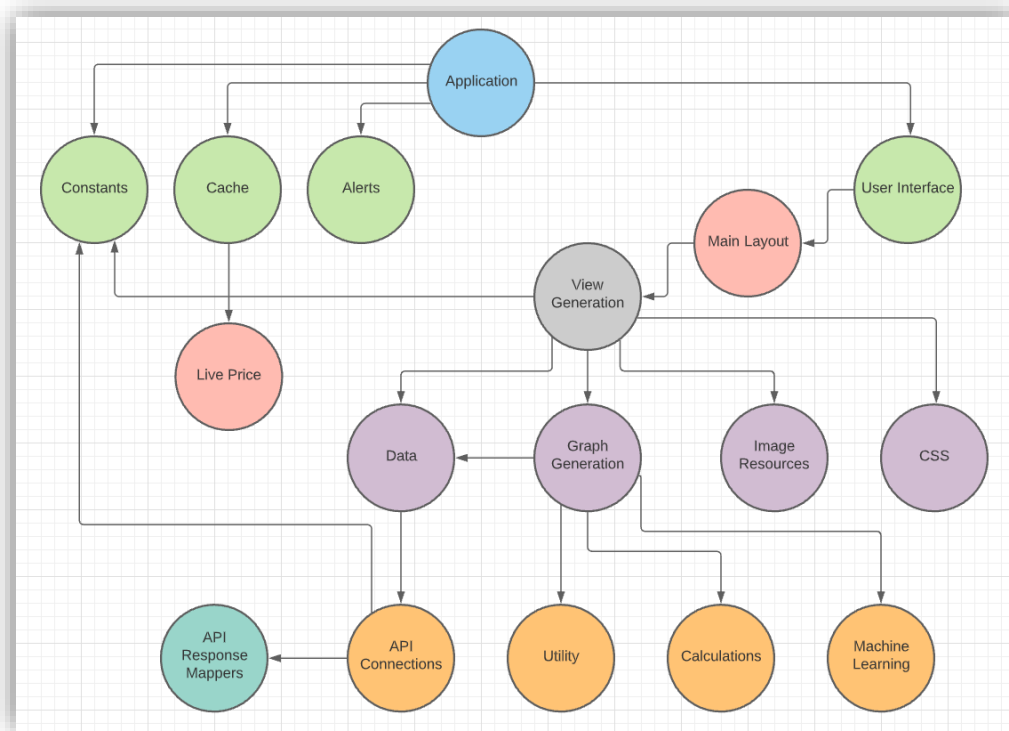


Figure 4.1. Application architecture.

Root

The Architecture of the application is shown in the figure above. The Application is divided using colours and levels to indicate finer grained access and function. The root of the application is the Application class that is annotated with “@SpringBootApplication”. The application is run from here, it initialises the static Cache and Alerts classes that are defined in the constants class so that they are accessible anywhere in the application.

Constants contain information such as API addresses, API keys, frequently used characters or strings etc. The Cache contains values that are changed frequently that need global access such as Bitcoin's live price that uses a constantly running thread to retrieve the price, update the value then sleep for a set amount of time. Alerts contain the generated error alerts that need to be accessed globally when an error occurs, they are displayed for a set amount of time then hidden again.

User Interface

The user interface is based on a main view layout. This is the parent that has all the necessary imports etc that the views inherit from. The main layout switches the displayed view when a user traverses to a different route on the page via the navigation tab. Each view is generated on request to enable multiple users to see the same page and have different interactions with it.

Each view is set of components such as graphs and displayed text which are in separate generation methods making the views modular, this modularisation enables the re-rendering of certain components given a user input. On View load, the constructor uses the methods to generate all components, components are added to a board object for positioning and grouping then the board object is added to the view. The view extends horizontal layout that enables the add() method to add elements to the view.

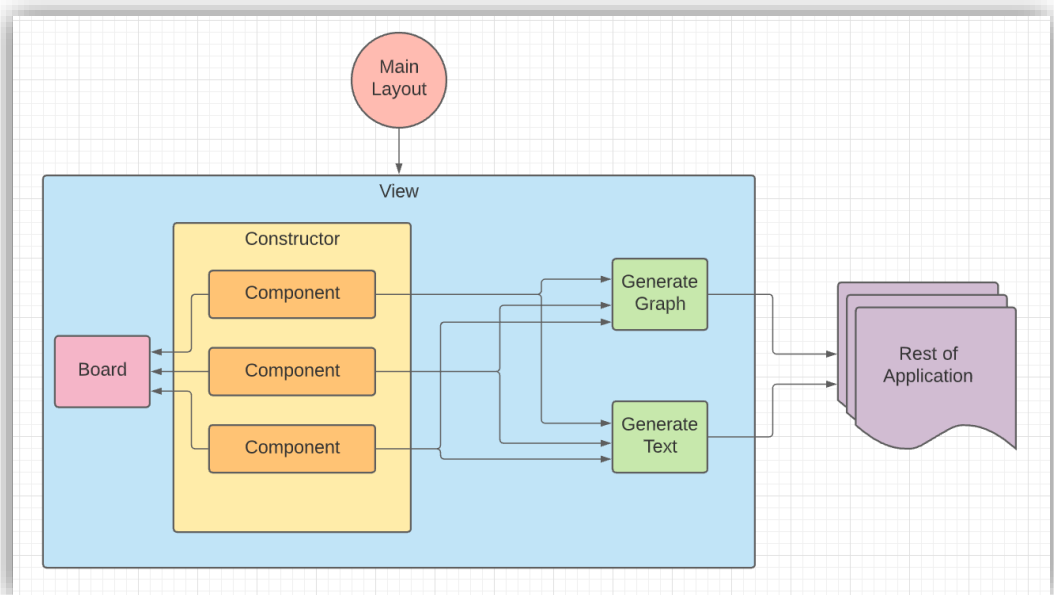


Figure 4.2. View Generation.

Resources & Data

Any resources the view requires can be retrieved from or calculated by the machine learning class, calculations module, utility module, the API connections class, or the

CSS scripts. The calculations module deals with calculations such as the correlation between two data sets and date conversion from a string to Epoch & Unix timestamp which is the date time in the form of a number which represents seconds since Jan 1st, 1970. This format is required for graphs to plot time series data.

The module also calculates the value of a current investment given a value and investment amount along with a calculator that determines the percentage change between two values which is used to show price action in the application.

The utility module consists of the Cache class, Constant class and Live price which were all covered previously in the report. The API connections, machine learning and CSS scripts were also covered in a previous chapter.

```
public PricePredictions(){
    bitcoinSeries = calculateBitcoinDataSeries();
    predictionComponent = createPredictionComponent();
    bitcoinDayData = calculateBitcoinDayData(bitcoinSeries);
    generatedContent = generateContent();
    generatedContent.setClassName("contentBoardPredictions");
    add(generatedContent);
}
```

Figure 4.3. Constructor generates components.

```
private CheckboxGroup<String> createRequiredPredictionCheckBoxes(){
    CheckboxGroup<String> checkboxGroup = new CheckboxGroup<>();
    checkboxGroup.setLabel("Required Variables for Forecasting");
    checkboxGroup.setItems("close", "date");
    checkboxGroup.addThemeVariants(CheckboxGroupVariant.LUMO_VERTICAL);
    add(checkboxGroup);
    checkboxGroup.select(...items: "close", "date");
    checkboxGroup.setHelperText(
        "close: Price at the end of a given day | " +
        "date: Date on a given day");
    checkboxGroup.setItemEnabledProvider(item -> item.contains(" "));
    return checkboxGroup;
}
```

Figure 4.4. component generator.

4.3 Price Charts



Figure 4.5. Price Charts page.

History Chart

The History chart provides past data of bitcoin price. It can be viewed in candlestick form which provides the day opening, close, high, and low price, and spline form. The user can hover over the chart at a point to see the date and data at that point in time. The different chart types can be navigated using the chart settings dropdown.

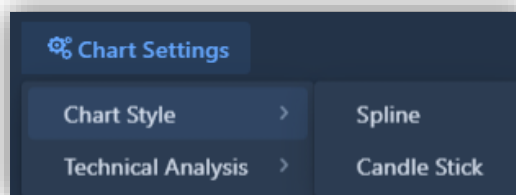


Figure 4.6. Price Charts page – chart style.



Figure 4.7. Price Charts page – history candlestick chart



Figure 4.8. Price Charts page – history spline chart.

The spline chart also has two different moving average overlays that indicate the average of a previous number of days. These are used in pairs, one high and one low moving average and the occurrence of crossings indicate a change of momentum of the price movement.

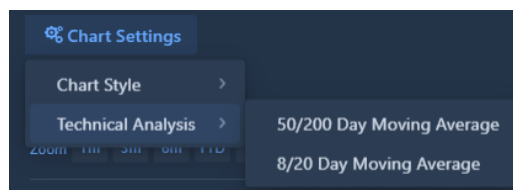


Figure 4.9. Price Charts page – technical analysis.



Figure 4.10. Price Charts page – moving average spline.

These chart's time period can be moved via adjusting the highlighted scroll bar below the chart, the buttons in the top left or dates directly input into the top right.

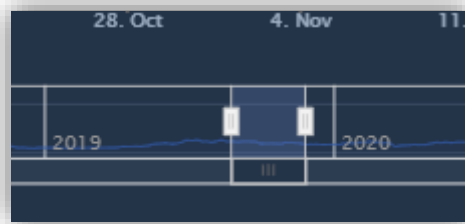


Figure 4.11. Price Charts page – set date scroll.

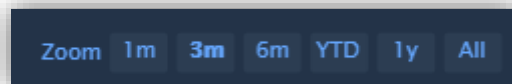


Figure 4.12. Price Charts page – set date button.

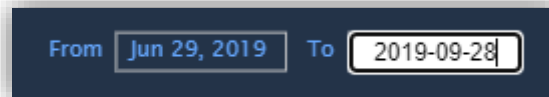


Figure 4.13. Price Charts page – set date input.

Live Price Chart

The live price chart updates every two minutes, it displays a live spline of the price. The colour and arrow change based on the previously received price, e.g., if the new price is higher than the price two minutes ago it will be green and red if its lower. The 24hr change indicates the change in price since the day before and will be green if positive and red if negative.

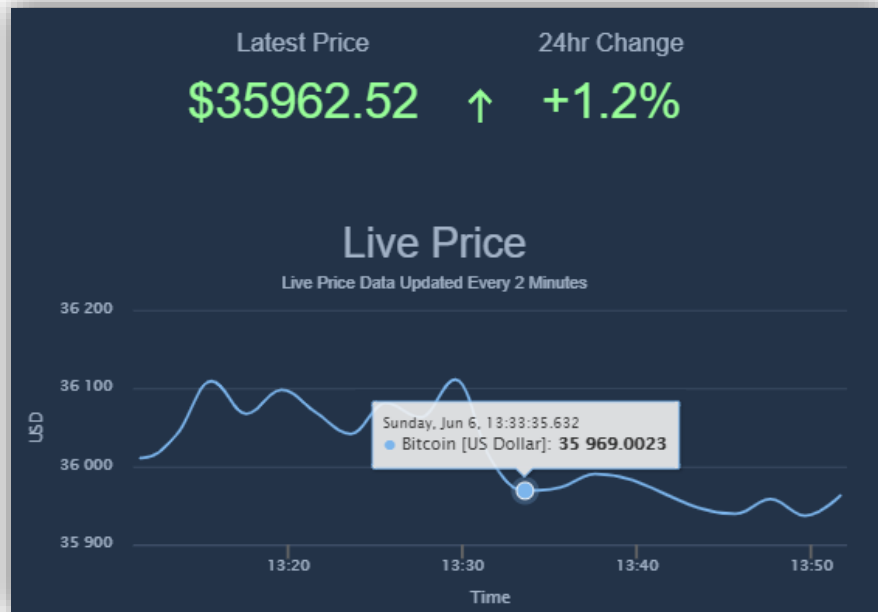


Figure 4.14. Price Charts page – live price positive.

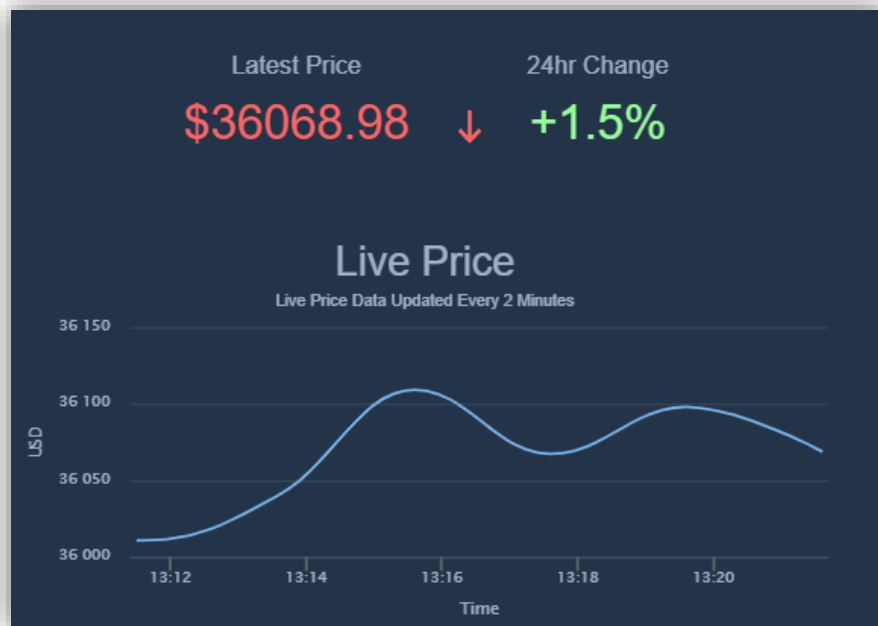


Figure 4.15. Price Charts page – live price negative.

Volume Chart

The volume chart indicates the highest trading volume in a day based on the sum of the transactions on a given moment of a day. This indicates large interest and selling / buying of the currency.



Figure 4.16. Price Charts page – trading volume.

4.4 Trend Correlations

The trend correlations page displays the visual representation of two sets of data and performs Pearson's product moment coefficient calculation to determine the strength of the correlation between the two sets of data. The chart displays each day and its value in relation to the other. This graph has dual axis so that the values are compared to in relation to percentage change rather than the raw value which is arbitrary when it comes to correlations and movement comparisons.



Figure 4.17. Trend Correlations page.

The user can select a myriad of trends that they can use to compare Bitcoin to including other cryptocurrencies, Covid-19, and stock market indexes such as the S&P 500. The user can also specify the period in which to make the comparison with the start and end date dropdown. The user can hover at any location on the graph to show the date and values at that point.

The gauge visually indicates the result of the correlation calculation and above it the current comparison trends are displayed with the percentage strength of the correlation. The correlation changes colour with the gauge which means if the gauge is pointing to an orange zone the text will be orange etc.

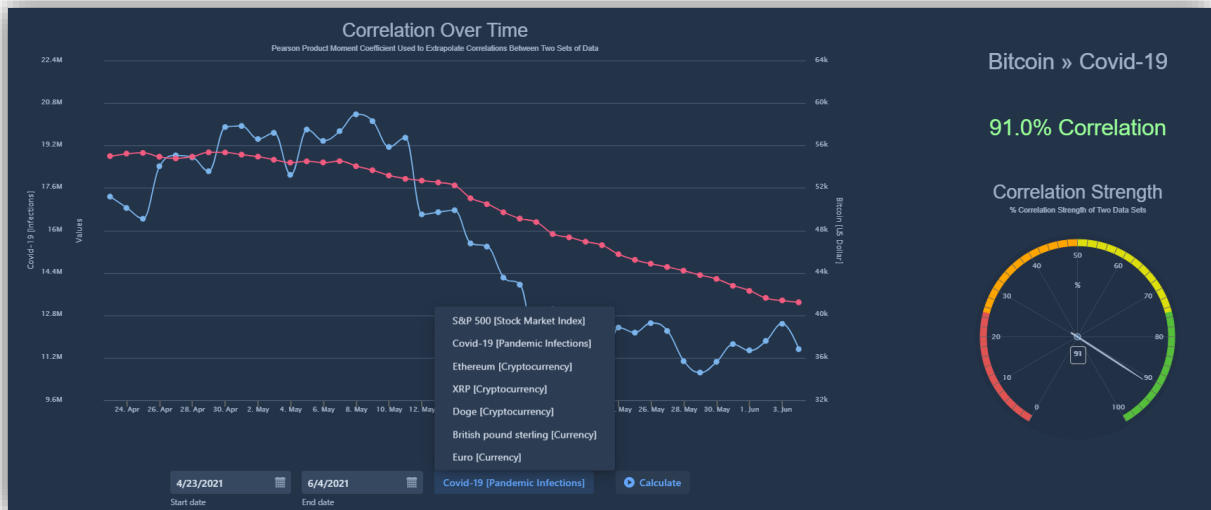


Figure 4.18. Trend Correlations page – settings.

4.5 Price Predictions



Figure 4.19. Price Predictions page.

The price predictions page uses machine learning to predict future time series data which was explained in the previous chapter. The user can specify what variables to include in the calculation, what time frame to use as training data and how many days to forecast into the future.

Giving the user the ability to change the time frame allows for the user's selected inputs to be tested and compared to what happened as if the user specifies for example to use four weeks of training data from last year and to forecast the subsequent week, the model will forecast a week that has already occurred, and it will show the prediction and actual occurrence on the graph.

Once the user feels that their settings are optimal for the period they wish to forecast, they can specify to forecast the currently unknown days and use it as a prediction. In the chart the green line represents the predicted price, and the blue represents the current price, if the user hovers over the line the type of price will be indicated.



Figure 4.20. Price Predictions page – testing downtrend.



Figure 4.21. Price Predictions page – testing uptrend.

5/6/2021

6/5/2021

Start of training data

End of training data

7

Days to forecast

Optional Variables for Forecasting

☒ RSI
 ☒ high
 ☐ low
 ☒ open
 ☐ volumeFrom
 ☐ volumeTo
 ☒ 8MA
 ☐ 20MA
 ☐ 50MA
 ☒ 200MA

RSI: Relative strength index indicating over/under valued | high: Highest price reached on a given day | low: Lowest price reached on a given day | open: Price at the start of a given day | volumeFrom: Lowest volume on a given day | volumeTo: Highest volume on a given day | 8/20/50/200MA: Day moving average

Required Variables for Forecasting

☒ close
 ☒ date

close: Price at the end of a given day | date: Date on a given day

Calculate

Figure 4.22. Price Predictions page – settings.

4.6 Investment Tools



Figure 4.23. Investment Tools page.

The investment tools page aims to use all the data computed by the application in useful suggestions components. They aim to serve as an investing aid in making informed decisions.

Expected Highs and Lows

Predicted highs and lows that are upcoming are displayed on a spider graph. It uses a green line to indicate the highs and a red one to indicate lows. This is intended to be used as an aid in timing an entry or exit into or out of a position of cryptocurrency as it informs the user of the probable lows and highs.

This will stop the user from buying or selling at suboptimal times. This chart indicates that if the red line price is reached within that time frame that it is an ideal buying opportunity as this is as low as the price is expected to get and if it reaches the green line value that this is an optimal selling opportunity as the price has peaked in the given time frame.



Figure 4.24. Investment Tools page – predicted highs & lows.

Investment Value Calculation

The investment calculator calculates the current value of an investment given the value invested and when it was invested, it also indicates the price action of the investment using either red or green percentage increase or decrease display.

Investment Value Calculator
Enter an investment amount and its date to calculate how much it is worth now

Amount Invested: \$ 1200 Date Invested: 6/5/2021 [Calculate](#)

\$1212.12 +1.01%

Figure 4.25. Investment Tools page – investment calculator.

Relative Strength Index

The relative strength index is calculated using the set of previous day's percentage movement. It is a good indicator as it replicates a commodities movement due to supply and demand, this means that if a commodity is oversold and drops rapidly then it will be quickly bought back up by others. This will provide a more accurate prediction when used also as it makes the chart fluctuate within a range rather than just falling or rising in value. If it is oversold, then it will be bought as it has fallen too quickly, and people sell it if its overbought as they know it will drop back down as its overpriced.

The RSI chart indicates something is oversold once it passes a value of 70 and undersold if it drops below 30. This is a good indicator for investing as if something is oversold it is a good time to buy and if it is overbought it is a good time to sell.

The dial below the chart indicates to the user if bitcoin is oversold and to buy, fair valued and to just hold your position or if its over bought and to sell. The dial will turn red to sell orange to hold and green to buy.



Figure 4.26. Investment Tools page – relative strength index.

4.7 Conclusion

The application's architecture and flow were described in this chapter, each page was displayed and explained. The application's first three pages are reflected in the final investment tools page where the user can use the suggestions provided by the application whereas the rest of the application provides a more experimental and analytical approach where the tools are presented, and the user can manipulate the inputs and come to their own conclusions.

Chapter 5. Results

5.1 Introduction

This chapter will investigate the results provided by this application and retrospectively analyse the work completed during the completion of the project. Difficulties will be discussed along with what future work could be completed to improve the application further.

5.2 Evaluation

The performance of the application was surprisingly effective in predicting price movements given the correct application settings. The author feels that this was quite a success in terms of experimentation of prediction of time series data using self-calculated data. The user can really experiment with this tool and create unique setting configurations that extract as much accuracy from the prediction as possible.

The charts and displays were meticulously tuned with CSS and the resulting effect of a dark base colour with vibrant pastel colours on top to contrast worked very well. The design and colour scheme are unique and quite elegant. The resulting content is very legible and pleasant to use.

The correlation of trends with Bitcoin tool exceeded expectations with many different types of trends available for the user to choose from along with the ability to choose the period for which to compare the two trends made this a very useful and powerful tool. The Visual gauge and colour changing display and names are a great addition to the initial plan.

The Investing tools that give suggestions became a larger part of that project than expected once the author realised how useful and powerful they are. They can help novice investors make sound decisions based on analytics and logic and the more seasoned investors can still enjoy the tool as the values calculated such as the relative strength index are universally used due to their relevance.

Some challenges encountered included architectural design. The original version of the application revolved more around the cache. All charts were generated and updated there and used around the app. The issue arose when a page was reloaded the application would crash. The cause turned out to be the fact that Vaadin does not support static components which meant that a cache containing graphs was not possible as each graph needed to be generated for each view individually every time that page was visited. The component then needed to be discarded once the user left the page. The cache is still used but only holds values such as the live price rather than Vaadin objects.

Another issue faced was the varied limit on API calls. For free API access there are usually call limits, there varied among the APIs used. Some functionality suffered such as the live price because it can only update every few minutes rather than every few seconds to ensure the API key does not pass the limit. There were other live price charts created also but had to be scrapped as the limits for those specific API were so low that this was no viable.

The difficulty in creating the correlation feature was quite substantial. This was because each set of data had a different amount of data within it. This was an issue as an algorithm had to be construed so that for each comparison two new sets of filtered data were constructed to be compared. The start and end date of the new sets needed to match. To complete this the algorithm identifies the latest start for a data set and the earliest end and extrapolates data between those time frames into the filtered sets of data.

5.3 Deliverables

This project delivered a fully functioning application that has a plethora of tools to analyse blockchain through Bitcoin. The tool provides experimental tools along side data displays and calculated suggestions, these aspects make the application well rounded.

The application is containerised in a docker container and can be deployed on any system locally to use by the user or can be hosted on a server that can scale this application and provide access to it to anyone in the world.

5.4 Future Work

Future work could be done on this app to make some improvements. The machine learning algorithm could be run autonomously by a program, and it would run many predictions with many different setting sets and use the best set of combinations on future predictions. This is a difficult task as there are a lot of variables such as what data to use, how much data to use and how far into the future can the accuracy of prediction be maintained.

Other trends that correlate highly in the correlation tool could also be used in the prediction tool. If something is highly correlated to Bitcoin, then this could be added to the model and expose the model to some unique data and movements that could assist in creating more effective models.

Another possible addition to this application would be a login system and a connection to a database so that users can save any settings made and keep a list of their investments and the application could keep track of the status, gain or loss of

each position. The addition of logins and profiles could enable users to send each other settings and results of calculations.

The ability to draw on the graphs could also increase the use cases for the display graphs. This would enable the drawing of trend lines and any other channels an investor may like to keep track of.

5.5 Conclusion

To conclude this project report, the author has learned a lot about blockchain, cryptocurrency and Bitcoin. The ability to extrapolate trends from sets of data, predict time series data and create suggestion tools has been quite interesting as the price of Bitcoin is constantly changing and any tools created or knowledge learned can be applied instantly to current data and the accuracy recorded.

The creation of an aesthetically pleasing application using Java and CSS has been quite rewarding for the author and has showcased that frontend and UI creation can be done by programmers who focus mainly on backend work.

The blockchain data analytics tool that was created has been named CryptoChain and the author hopes that it can be used in the future whether to just track prices or by using the investing tools, the application has been a success and will be utilised.

The initial idea for the project was to determine the correlation between Covid-19 and Bitcoin, the resulting application can compute this and do much more. The initial concept of the pandemic being correlated to the value of bitcoin has turned out to be true however this does not imply causality. With much more tools implemented this concept can be further explored using the created application, CryptoChain.

References

- [1] Covid-19 API. [Online] Available from <https://rapidapi.com/marketplace> [Accessed November 2020]
- [2] Bitcoin API (Coindesk). [Online] Available from <https://www.coindesk.com/coindesk-api> [Accessed November 2020]
- [3] Bitcoin Analysis Tool (Coindesk). [Online] Available from <https://www.coindesk.com/price/bitcoin> [Accessed November 2020]
- [4] Machine Learning API (Smile). [Online] Available from <https://haifengl.github.io/quickstart.html> [Accessed November 2020]
- [5] Micro Service Framework (Spring Boot). [Online] Available from <https://spring.io/projects/spring-boot> [Accessed November 2020]
- [6] Java UI (Vaadin). [Online] Available from <https://spring.io/blog/2020/07/14/building-web-applications-with-spring-boot-and-vaadin> [Accessed November 2020]
- [7] Blockchain Explained (Investopedia). [Online] Available from <https://www.investopedia.com/terms/b/blockchain.asp> [Accessed November 2020]
- [8] How does Bitcoin Work?. [Online] Available from <https://bitcoin.org/en/how-it-works> [Accessed November 2020]
- [9] Machine Learning vs Neural Networks. [Online] Available from <https://www.upgrad.com/blog/machine-learning-vs-neural-networks/#:~:text=Neural%20Networks%20are%20essentially%20a,in%20many%20fields%20of%20interest.> [Accessed November 2020]
- [10] Software Project Management (Maven). [Online] Available from <https://maven.apache.org/> [Accessed November 2020]
- [11] Version Control (Github). [Online] Available from <https://github.com/> [Accessed November 2020]
- [12] Task Management (Asana). [Online] Available from https://asana.com/campaign/fac/think?utm_campaign=Brand--UK--EN--Core--EX&utm_source=google&utm_medium=pd_cpc_br&gclid=Cj0KCQiAqdP9BRDVARIsAGSZ8An4W1ykUSQwC7rRpRdyKTU65eHcTMP6q4ekWi9gjylhQFokZz-gUDkaAkDJEALw_wcB&gclsrc=aw.ds [Accessed November 2020]
- [13] Blockchain Technical Details (MIT). [Online] Available from <http://blockchain.mit.edu/how-blockchain-works/> [Accessed November 2020]
- [14] Bitcoin Mining (Investopedia). [Online] Available from <https://www.investopedia.com/tech/how-does-bitcoin-mining-work/> [Accessed November 2020]

- [15] Blockchain Analytics (Fintech News). [Online] Available from <https://www.fintechnews.org/blockchain-analytics/> [Accessed November 2020]
- [16] Blockchain Definition (TechTerms). [Online] Available from <https://techterms.com/definition/blockchain> [Accessed November 2020]
- [17] Long Short-Term Memory Networks (Machine Learning Mastery). [Online] Available from <https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/> [Accessed November 2020]
- [18] Neural Networks (Wikipedia). [Online] Available from https://en.wikipedia.org/wiki/Artificial_neural_network [Accessed November 2020]
- [19] Linear Regression (Yale). [Online] Available from <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm> [Accessed November 2020]
- [20] Blockchain Pros and Cons (101 Blockchains). Available from <https://101blockchains.com/pros-and-cons-of-blockchain/> [Accessed November 2020]
- [21] Covid-19 and Stock Markets (Taylor & Francis Online). [Online] Available from <https://www.tandfonline.com/doi/full/10.1080/20954816.2020.1757570> [Accessed November 2020]
- [22] Bitcoin Price Tool (Kitco). [Online] Available from <https://www.kitco.com/bitcoin-price-charts-usd/> [Accessed November 2020]
- [23] Bitcoin Price Tool (Business Insider). [Online] Available from <https://www.kitco.com/bitcoin-price-charts-usd/> [Accessed November 2020]
- [24] Cryptocurrency (Investopedia). [Online] Available from <https://www.investopedia.com/terms/c/cryptocurrency.asp> [Accessed June 2021]
- [25] Fiat Money (bitpanda). [Online] Available from <https://www.bitpanda.com/academy/en/lessons/whats-the-difference-between-a-cryptocurrency-like-bitcoin-and-fiat-money/> [Accessed June 2021]
- [26] Cryptocurrency (HPC). [Online] Available from <https://honestproscons.com/advantages-and-disadvantages-of-cryptocurrency/> [Accessed June 2021]
- [27] Bitcoin (Investopedia). [Online] Available from <https://www.investopedia.com/news/how-bitcoin-works/> [Accessed June 2021]
- [28] Bitcoin (Jimmy Song). [Online] Available from <https://jimmysong.medium.com/why-bitcoin-is-different-than-other-cryptocurrencies-e16b17d48b94#:~:text=The%20main%20advantages%20of%20Bitcoin%20are%20network%20effect%20and%20proven%20security.&text=Further%2C%20Bitcoin%20is%20more%20accessible,larger%20volumes%20than%20every%20altcoin.> [Accessed June 2021]
- [29] Bitcoin popularity (The European Business Review). [Online] Available from <https://www.europeanbusinessreview.com/bitcoin-and-its-constantly-rising-popularity/> [Accessed June 2021]

- [30] CSS (Wikipedia). [Online] Available from <https://en.wikipedia.org/wiki/CSS> [Accessed June 2021]
- [31] Java (Wikipedia). [Online] Available from [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)) [Accessed June 2021]
- [32] API (Wikipedia). [Online] Available from <https://en.wikipedia.org/wiki/API> [Accessed June 2021]
- [33] Containerization (Docker). [Online] Available from <https://www.docker.com/> [Accessed June 2021]
- [34] Maven (GeeksforGeeks). [Online] Available from <https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/> [Accessed June 2021]
- [35] Kanban (Kanbanize). [Online] Available from <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban#:~:text=in%20a%20Nutshell-Kanban%20is%20a%20workflow%20management%20method%20for%20defining%2C%20managing%2C%20and,by%20Agile%20software%20development%20teams.> [Accessed June 2021]
- [36] Automated deployment (Kubernetes). [Online] Available from <https://kubernetes.io/> [accessed June 2021]
- [37] IDE (IntelliJ). [Online] Available from <https://www.jetbrains.com/idea/> [Accessed June 2021]
- [38] Terminal application (Git Bash). [Online] Available from <https://www.git-tower.com/learn/git/faq/git-bash/> [Accessed June 2021]
- [39] Spring project generation (Spring initializr). [Online] Available from <https://start.spring.io/> [Accessed June 2021]
- [40] Covid-19 waves (AARP). [Online] Available from <https://www.aarp.org/health/conditions-treatments/info-2021/covid-4th-wave.html> [Accessed June 2021]
- [41] Bitcoin wallet (Investopedia). [Online] Available from <https://www.investopedia.com/terms/b/bitcoin-wallet.asp> [Accessed June 2021]
- [42] Cryptocurrency market cap leaders (Slick charts). [Online] Available from https://www.google.com/search?q=cryptocurrency+market+cap+leaders&safe=strict&bih=977&biw=1920&hl=en&ei=KWa6YN_oEJiNhbIPL_q90Aw&oq=cryptocurrency+leaders+&gs_lcp=Cgdnd3Mtd2l6EAEYAzIGCAAQFhAeMgYIABAWEB4yBggAEBYQHjIGCAAQFhAeOgcIABBHELADUPEOWPEOYLklaAFwAngAgAFJiAGLAZIBATKYAQCgAQGqAQdnd3Mtd2l6yAEIwAEB&sclient=gws-wiz [Accessed June 2021]
- [43] JavaScript Library (High Charts). [Online] Available from <https://www.highcharts.com/> [Accessed June 2021]

- [44] Testing Framework (JUnit). [Online] Available from <https://junit.org/junit5/> [Accessed June 2021]
- [45] Mocking Framework (Mockito). [Online] Available from <https://site.mockito.org/> [Accessed June 2021]
- [46] API Description (MuleSoft). [Online] Available from <https://www.mulesoft.com/resources/api/what-is-an-api> [Accessed June 2021]
- [47] API Development (Postman). [Online] Available from <https://www.postman.com/> [Accessed June 2021]
- [48] JSON (developer.mozilla). [Online] Available from <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON> [Accessed June 2021]
- [49] Unix Shell command language (Bash). [Online] Available from [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell)) [Accessed June 2021]
- [50] Library (Lombok). [Online] Available from <https://projectlombok.org/> [Accessed June 2021]
- [51] Library (Joda-Time). [Online] Available from <https://www.joda.org/joda-time/> [Accessed June 2021]
- [52] Machine learning (Weka). [Online] Available from <https://www.cs.waikato.ac.nz/ml/weka/> [Accessed June 2021]
- [53] Banerjee, M., Lee, J. and Choo, K., 2018. A blockchain future for internet of things security: a position paper. *Digital Communications and Networks*, 4(3), pp.149-160.
- [54] Böhme, R., Christin, N., Edelman, B. and Moore, T., 2015. Bitcoin: Economics, Technology, and Governance. *Journal of Economic Perspectives*, 29(2), pp.213-238.
- [55] Hess, A., Iyer, H. and Malm, W., 2001. Linear trend analysis: a comparison of methods. *Atmospheric Environment*, 35(30), pp.5211-5222.
- [56] Machine Learning (SAS). [Online] Available from https://www.sas.com/en_ie/insights/analytics/machine-learning.html [Accessed June 2021]
- [57] Wust, K. and Gervais, A., 2018. Do you Need a Blockchain?. 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)
- [58] Liu, Y. and Tsyvinski, A., 2020. Risks and Returns of Cryptocurrency. *The Review of Financial Studies*, 34(6), pp.2689-2727.
- [59] Yermack, D., 2015. Is Bitcoin a Real Currency? An Economic Appraisal. *Handbook of Digital Currency*, pp.31-43.