A Standalone Crypto Currency Seed Analyser

(suggested by the South Wales Police)

One Semester Individual Project – 40 Credits

Final Report

Author: Andre Mansley (1821796)

Supervisor: Michael Daley

Moderator: Stefano Zappala

Report Contents

1 Introduction 4
1.1 Project Scope and Problem 4
1.2 Aims and Objectives..... 4
2 Background Research 5
2.1 Crypto Currencies..... 5
2.2 Blockchain Technology..... 5
2.3 Wallets and Recovery Seeds..... 6
2.4 Existing Tools 7
3 Project Plan 8
3.1 Development Methodology 8
4 Specification and Design..... 9
4.1 Overview: 9
4.2 Functional Requirements 9
4.2.1 User Story..... 9
4.2.2 The Functional Requirements 9
4.2.3 Non-Functional Requirements . 11
4.3 System Design and Architecture.....12
4.3.1 User Interface Design 12
4.3.2 Website Flow 13
4.3.3 Blockchain Analysis..... 14
4.3.4 Code Structure 14
4.3.5 Pseudo Code 16
5 Implementation 17
5.1 Technologies and Tools Used 17
5.2 Final Implemented Design.....17
5.3 Blockchain Analysis & Code.....20
5.3.1 Changes to the Implementation20
5.3.2 Functionality Implementation.. 20
6 Testing.....26
7 Future Work 30
7.1 Future Implementation Idea 1: 30
7.2 Future Implementation Idea 2: 30
7.3 Feature Implementation Idea 3:.....30
7.4 Feature Implementation Idea 4:.....30
7.5 Summary.....30
8 Conclusion.....31
9 Reflection & Evaluation.....32
9.1 Overview 32
9.2 Programming Skills 32
9.3 Time Plan & Organisation 32
9.4 Challenges 32
9.5 Supervisor meetings 32
9.6 Future Projects 32
10 Glossary 33
11 References: 34

Table of Figures

Figure 1: CoinMarketCap Cryptocurrency Rankings 5
Figure 2: Example of Chained Blocks..... 5
Figure 3: Blockchain.com Search Result 6
Figure 4: Ian Coleman Screenshot 7
Figure 5: Development Time Plan 8
Figure 6: Actor and Use Cases..... 10
Figure 7: Initial Web Application Design..... 12
Figure 8: Colour Pallet Used in the Application 13
Figure 9: Flowchart of the Web Application 13
Figure 10: Blockchain Analysis Flowchart..... 14
Figure 11: Methods and Variables to be Used..... 15
Figure 12: Initial Web Application Design..... 17
Figure 13: Final Web Application Design 17
Figure 14: Initial Header Design..... 18
Figure 15: Final Header Design 18
Figure 16: Initial Derivation Settings Design..... 18
Figure 17: Final Derivation Settings Design 18
Figure 18: Initial Main Body Content Design..... 19
Figure 19: Final Main Body Content Design 19
Figure 20: Output of a Successful Wallet Analysis 19
Figure 21: Output of No Results Found 19
Figure 22: Output of a PDF Exportation 19
Figure 23: A List of Derived Wallet Addresses..... 27
Figure 24: Blockchain Class Being Initialised..... 27
Figure 25: Bitcoin Blockchain Analysis Results 28
Figure 26: Ethereum Blockchain Analysis Results .. 28
Figure 27: Possible UI Outputs from the Blockchain Analysis..... 28
Figure 28: Example Output of a Successful Wallet Analysis..... 29
Figure 29: Times Taken: Bitcoin Blockchain (Wallets with a Balance)..... 29
Figure 30: Times Taken: Bitcoin Blockchain (Wallets with No Balances) 29
Figure 31: Times Taken: Ethereum Blockchain (Wallets with No Balances)..... 29

Abstract

Cryptocurrencies are assets/currencies that run on a cryptographically secured blockchain that can be used as investments or as a currency to send, exchange or store value. Many people who invest in cryptocurrencies do so legally, however, some people may use cryptocurrencies for illegal activities. The project I am undertaking will be to develop a web application that enables investigators to analyse a retrieved cryptocurrency wallet recovery seed and determine if any associated wallet addresses contain any cryptocurrency value that can be recovered.

This report provides technical documentation of the design and development of the Standalone Crypto Currency Seed Analyser that I have created. The purpose of the cryptocurrency seed analyser application is to help Police Forensic/Digital investigators save a lot of time whilst they analyse a retrieved wallet recovery seed from an investigation. This project will be working towards the provided brief which states, “A standalone cryptocurrency seed analyser. There are tools like iancoleman.io – but what would be of significant value would be a tool that would analyse a recovery seed from a crypto wallet to output known addresses that it would use. Then a means of exporting this information to cross-reference the blockchain. So, in essence, if we have a recovery seed, the investigator wants to as quickly as possible understand if there is any value in recovering the wallet because cryptocurrency may be present.“.

This report will discuss the functionalities that I have implemented into the web application such as blockchain analysis and cross-referencing that is used to analyse potential cryptocurrency wallet addresses associated with an entered mnemonic wallet recovery seed to determine whether any of the wallet addresses contain any cryptocurrency that can be recovered.

I will also be outlining the system architecture via diagrams to help illustrate the step-by-step processes involved. The code that has been created to implement the functionality of the application has also been documented to help provide a clear understanding of how each process of the application functions. I will then proceed to perform tests against the specified requirements to determine whether the application meets expectations and then outline further future work and functionality that could be implemented.

Acknowledgments

I would like to thank my supervisor Michael Daley for all the support and advice that he has provided me throughout this project. I wish you the best and I hope you have a great retirement!

1 Introduction

Cryptocurrencies are digital currencies and the first and most well-known cryptocurrency is Bitcoin which was developed by someone under the alias name of 'Satoshi Nakamoto' back in 2008/2009 during the last economic recession.

These types of currencies run on a technology called a Blockchain which is a decentralised publicly distributed ledger that uses cryptography to ensure that all transactions made on the Blockchain are secure, anonymous, and immutable.

Cryptocurrencies are stored on addresses, also known as Crypto Wallets. These addresses which are comprised of a long string of numbers and letters can be used to send and receive cryptocurrency across the blockchain, eliminating the need for an intermediary such as Banks.

The project I am undertaking will be to develop a tool that can analyse a recovery seed of a wallet that will have been obtained during an investigation to determine how much cryptocurrency is being held within the wallet and whether it is worth recovering the funds held within it.

1.1 Project Scope and Problem

Cryptocurrency transactions on the Blockchain are anonymous and cannot be linked back to an individual's identity (unless trading on an exchange where KYC (Know Your Customer) is mandatory). The anonymity of cryptocurrency usage can lead to a rise in digital crime and therefore a lot of cryptocurrencies may be stored in wallets that may be retrieved during an investigation.

To help the police investigate cybercrime involving digital currencies I will be developing a web application that will take in a wallet recovery seed as an input and will output known addresses as well as the value of the cryptocurrency being held within the wallet. This will enable forensic police investigators to quickly and efficiently determine if there is any cryptocurrency within the wallet being investigated and whether it is worth recovering the funds. During my initial contact with the police to gather and clarify project requirements they stated, "To seize cryptocurrency, it requires quite a lengthy application to get the authorisation. This is done prior to recovering a wallet due to time restrictions. If we could rely on a tool to check seeds beforehand it would save a lot of time!". Therefore, to assist them and speed up investigations, the application I will be developing will aim to provide them results containing known/used wallet addresses and the balances of the wallets associated with the recovery seed they have retrieved.

1.2 Aims and Objectives

The overall aim of this project is to develop an easy-to-use web application that will analyse a wallet recovery seed to determine known addresses and then cross-reference the blockchain to output the value of the cryptocurrency present inside the wallet.

Aim 1: The first aim of this project will be to develop an easy-to-use interface that will allow the user to enter a wallet recovery seed as a string.

Objectives: To achieve this aim, the following objectives must be met:

a. Design a text box input that takes in a wallet recovery seed as a string *(Approx. 12 to 24 random English words. i.e. (hammer, pyramid, donkey…)*
b. Design an 'Analyse' button that will start the analysing process of the entered recovery seed
c. Design an area of the application that will be used to output the returned results of the seed analysis

Aim 2: The second aim will be to analyse the entered wallet recovery seed and determine a list of output known addresses

Objectives:

a. Recover the wallet associated with the provided recovery key
b. Extract the wallet address (a long string of numbers and letters) that can then be used to cross-reference the blockchain

Aim 3: The third aim will be to cross-reference the Blockchain and output the value of cryptocurrency being held within the wallet.

Objectives:

a. To display the amount of cryptocurrency inside the wallet being analysed (returned as values in the format £0.00
b. Retrieve and display the results within 5 seconds

2 Background Research

Before starting the development of the Cryptocurrency Seed Analyser application. I will be researching various topics that will expand my knowledge and facilitate development. The research areas involved will include cryptocurrencies, blockchain technology, wallets and recovery seeds, and then other applications that provide similar functionality to the proposed application that I will be developing.

2.1 Crypto Currencies

Cryptocurrencies, as previously stated are a form of digital currency which uses Blockchain technology and cryptography to function. Many of the cryptocurrencies on the market are projects which aim to bring real use cases to the world. For example, the new and current trend is Defi (Decentralised Finance) which uses blockchain smart contracts to verify and create contracts without the need for central governance such as Banks and Exchanges *(Eng-Tuck Cheah 2021).* Other types of fast-growing projects include borrowing/lending, IoT-related projects, synthetic asset creation of real-world assets, gaming, and many more.

There are currently 8514 cryptocurrencies in the market as of today according to CoinMarketCap *(Cryptocurrency Prices, Charts And Market Capitalizations | CoinMarketCap. 2021)* which is one of the most popular websites that lists cryptocurrencies, their rankings, as well as other useful information such as their descriptions and which exchanges the coin can be traded on.

Figure 1: CoinMarketCap Cryptocurrency Rankings

The total cryptocurrency market capitalisation, which is the value of all cryptocurrencies put together is currently sitting at approximately $1.6Trillion after the recent surge in prices across the entire market. Especially for Bitcoin which has risen from approximately $4,000 to $58,000 within a year and is ranked as the number 1 cryptocurrency and the 8th largest asset in the world in terms of market capitalisation making it worth more than Facebook, Tesla, and Visa according to Companies Market Cap *(Assets ranked by Market Cap - CompaniesMarketCap.com. 2021).*

Whilst many people who invest and trade in cryptocurrency and Bitcoin are doing so legally, there will be some that will use cryptocurrency for cybercrime and other criminal activities due to its anonymity and the market still being unregulated.

Some privacy-focused coins in the market prioritise anonymous transactions. These types of coins hide the identities of the user as well as the amount of cryptocurrency being held within a wallet. This can make it very difficult to track a wallet and its value. Some of the most used privacy coins include Monero, ZCash, Dash, Verge, Horizen, and Beam *(SETH 2021).* Due to the anonymity of these privacy coins, it would be very difficult to analyse a wallet and retrieve the value of cryptocurrency being held in these types of cryptocurrency wallets. Therefore, the application I will be developing will use the Bitcoin Blockchain and Ethereum Blockchain to retrieve the value of cryptocurrency held within Bitcoin and Ethereum wallets because these are the two most popular cryptocurrencies.

2.2 Blockchain Technology

A Blockchain is a decentralised and distributed database consisting of chained blocks that have been secured cryptographically to provide transparency, immutability, and most importantly, security. Blockchains work as a peer-peer network with no central authority involved and rely on computers to act as nodes to validate transactions. Once all nodes on the network agree to a consensus, the transactions are added to a block and then chained to the previous block.

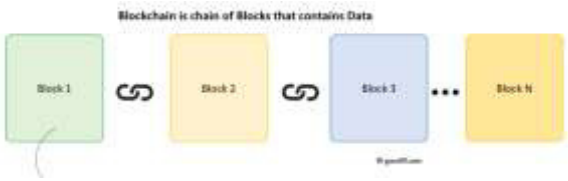


Figure 2: Example of Chained Blocks

On the Bitcoin blockchain, each block is approximately 1MB in size which contains many transactions, a timestamp for each transaction, as well as a cryptographic hash of the previous block which is used to link all the blocks together. An example of a cryptographic hash may look like this:

1395cf5e4ec872a1036ad408b8294d49384c59813324437ee54afb16b1cc7c4f (This is a cryptographic hash.)

The process of transacting on the Bitcoin blockchain is as follows *(Blockchain Tutorial: Learn Blockchain Technology (Examples). 2021):*

1. Someone makes a transaction, from one wallet to another
2. The transaction is broadcasted to the blockchain nodes
3. Blockchain nodes (computers) validate the transaction
4. Transactions are stored in blocks and then the block is appended to the Blockchain permanently

The first blockchain was developed over 10 years ago by Satoshi Nakamoto, the creator of Bitcoin. However, it was not till recently that Blockchain has started to become recognised as an emerging technology that can provide a whole range of benefits in the technological world, and for businesses. Some of the advantages of Blockchains according to IBM include greater transparency, enhanced security, improved traceability, an increase in speed and efficiency, and lastly, lower costs *(Hooper 2021).*

Blockchains can be public or private. The Bitcoin blockchain and Ethereum Blockchain that I will be using for my project are public. This means that all transactions sent and received from wallets, and the value of cryptocurrency stored in these wallets are easily viewable to all (no personal identity is attached). There are websites such as Blockchain.com *(Blockchain.com Explorer | BTC | ETH | BCH. 2021)* which enable people to easily view all this information and search for wallet addresses.

From this website, Blockchain addresses, block numbers, and transactions can be searched. The output from a search on this platform would result in the following format being received. See Figure 3 below.

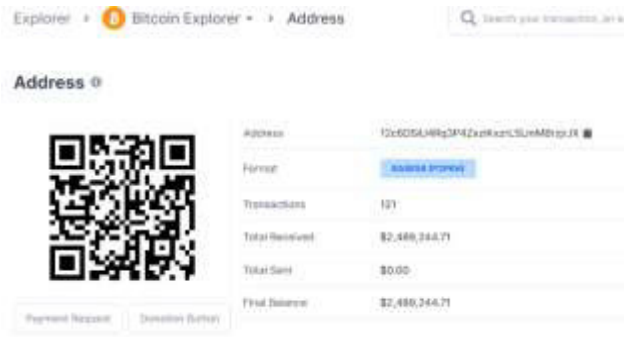


Figure 3: Blockchain.com Search Result

From searching that blockchain address, we can see the following information:

- The Bitcoin wallet address
- Number of transactions made from/to wallet
- The total value of money received
- The total value of money sent
- The balance/value of the wallet currently

The project I am undertaking will require me to search blockchain to retrieve the amount of cryptocurrency being held within a wallet address that will be derived from the entered wallet recovery seed. To search the blockchain, I will be using the API provided by Blockchain.com. This will enable me to pass an xPub wallet address as a parameter, and retrieve the information listed above. An xPub wallet address is the main wallet public address that can hold multiple wallet addresses within it.

Searching the blockchain with an xPub address will return a list of associated addresses that have been used, the balances within each, and all the transactions that have been made from/to each address.

2.3 Wallets and Recovery Seeds

Cryptocurrency wallets are addresses on the Blockchain which people can use to send and receive crypto. A wallet address is comprised of a long string of numbers and letters. An example of a Bitcoin wallet address can be seen below:

1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

Cryptocurrency can only be sent across its own blockchain. For example, a Bitcoin address can only send Bitcoin to another Bitcoin address.

A Bitcoin address cannot send Bitcoin to an Ethereum address etc. otherwise the funds are lost and cannot be retrieved.

A user can create a new wallet at any time in which upon creating they are provided with a public key and a private key. The private key is presented to the user as a list of 12 to 24 randomised English words which are selected from a predefined BIP39 dictionary list with 2048 words. These are known as BIP39 mnemonic recovery seeds which can be used to recover/access the wallet if needed. This makes it very important to keep private keys secure and safe as others will be able to get access to the wallet if they obtain the recovery seeds.

An example of a BIP39 Mnemonic recovery seed:

rookie tiger abstract patient solar opera cattle debate shrimp finish flower party

The web application that I am developing will take in a BIP39 mnemonic recovery as an input such as the one above and will output known addresses, and then cross-reference the Blockchain to determine the value of cryptocurrency being held within the wallet.

2.4 Existing Tools

An existing tool that provides similar functionality is Ianocoleman.io *(BIP39 - Mnemonic Code. 2021).* This tool takes in a set of BIP39 Mnemonic words and derives the BIP39 private seed equivalent and a BIP32 root key which is known as the 'master private key'. This master private key is then used to derive a list of all the potential wallet addresses that can be used from the entered mnemonics.

Figure 4: Ian Coleman Screenshot

This web application allows for different types of BIP derivations. BIP stands for Bitcoin Improvement Protocol. There are currently 5 protocols which include BIP32, BIP44, BIP49, BIP84, BIP141. These different BIP variants result in different derivations of wallet addresses. Most wallet addresses use the BIP32 and BIP44 derivations.

Example of Ian Coleman Mnemonic Code Converter:

Using the BIP39 mnemonic words previously mentioned, the following outputs will be derived and shown:

Bip39 Seed:
0ebfe3c2aa3af5aa2741a64e36d6ce8a442d8971c8898ac6be2f07bb48a7756baf0d68a930b04868a915a012d07303011726a1cec401c930e2f4e472b6c77883

BIP Root Key:

xprv9s21ZrQH143K2FMSgzJ3ECYgJ7PazVcLggT5jbGwpftgsyBVT8zMBCcgEXUeKyfBQ4G5syrXKhJFpTE4BR8TRamRy34ca3r3sWyfR8BTdBT

Using a BIP44 wallet derivation method, the following xPubKey will be derived:

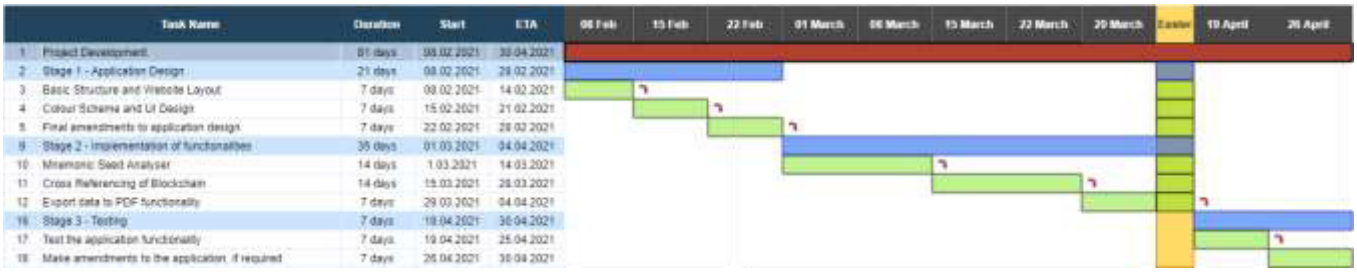
Account Extended Public Key:

xpub6DWFzqfdFEknaazWRRZVmFWCK2nQn8jmCB5tpe1eSzoJFgKziEtpY1E7aQ62DB4G64odGsrGHBzozkpeBPJtNNXmLyeawiD2HMoGhTgfDLP

The account extended public key which I will reference as the xPubKey throughout this report will be the value that my application will use to cross-reference against the blockchain to retrieve used wallet addresses, and balances.

The source code provided by this tool is under the MIT license and is free for anyone to use, copy, modify, publish, and sell. I will be using the source code from this existing tool *(iancoleman/bip39. 2021)* to convert the entered mnemonic words into an xPubKey. I will be redesigning the user interface as well as implementing new extended functionality that will meet the requirements set out in the project brief and requirements.

3 Project Plan



	Task Name	Duration	Start	ETA
1	Project Development	81 days	08.02.2021	30.04.2021
2	Stage 1 - Application Design	21 days	08.02.2021	28.02.2021
3	Basic Structure and Website Layout	7 days	08.02.2021	14.02.2021
4	Colour Scheme and UI Design	7 days	15.02.2021	21.02.2021
5	Final amendments to application design	7 days	22.02.2021	28.02.2021
9	Stage 2 - Implementation of functionalities	35 days	01.03.2021	04.04.2021
10	Mnemonic Seed Analyser	14 days	1.03.2021	14.03.2021
11	Cross Referencing of Blockchain	14 days	15.03.2021	28.03.2021
12	Export data to PDF functionality	7 days	29.03.2021	04.04.2021
16	Stage 3 - Testing	7 days	18.04.2021	30.04.2021
17	Test the application functionality	7 days	19.04.2021	25.04.2021
18	Make amendments to the application, if required	7 days	26.04.2021	30.04.2021

Figure 5: Development Time Plan

3.1 Development Methodology

The development methodology that I will be using for my project is the waterfall method. The advantage of using the waterfall methodology for this project is that it enables me to gather and understand all the requirements before starting on the development of the application. Even though the maximum project timeframe is 15 weeks long, this still provides enough time to develop a fully functional web application using the waterfall methodology.

The stages of the waterfall methodology include requirement gathering, designing, implementation, testing of the system, and then the final stage being deployment.

As I start the development, I will firstly be designing the application which includes the website structure and the visual user interface that the user would interact with.

After the design stage, I will move to the implementation stage. This is where I will be implementing the main functionality and features of the application which includes the seed analyser and cross-referencing of the blockchain.

Lastly, after the main application features and technical functionality have been implemented, I will move to the testing stage where I will test the application thoroughly to ensure that all features have been implemented and are functional and that all technical issues, if any, are eliminated.

4 Specification and Design

4.1 Overview:

This section will describe the system specifications and designs of this project including the functional and non-functional requirements.

The functional and non-functional requirements are important to identify before the start of the project as these describe how the system should function, the features and functionality expected from the software, and how this functionality will be achieved.

The user interface design and system architecture design will be detailed and described along with diagrams to provide a graphical insight into the functionality of the cryptocurrency seed analyser.

4.2 Functional Requirements

The functional requirements detailed below are the features and functionality that must be developed into the proposed web application to ensure that the requirements outlined in the project brief are met.

4.2.1 User Story

"As a cybercrime investigator, I want to be able to analyse a cryptocurrency wallet recovery seed so that I can quickly and efficiently determine if any associated wallets hold any cryptocurrency that can be recovered."

I have acquired a cryptocurrency wallet seed in the form of a BIP39 mnemonic seed. I would like to be able to analyse this seed to determine if there is any cryptocurrency being held and how much is in the wallet.

- As a user, I can enter the wallet recovery seed into the application.
- As a user, I can derive the public key and private key from the recovery seed I have entered.
- As a user, I can quickly view wallet addresses associated with the recovery seed.
- As a user, I can view the total balance of cryptocurrency being held within the wallets and the balance within each wallet.
- As a user, I can export all the information derived and retrieved into a downloadable PDF.

4.2.2 The Functional Requirements

FR1: The application must allow for the input of a BIP39 mnemonic seed

The BIP39 mnemonic seed will have been recovered by the police during an investigation. To analyse the recovered seed, they must be able to enter the mnemonic seed into the application.

Acceptance Criteria:

- Textbox must be positioned in a location that is easily accessible and modifiable
- Textbox input must be of a string data type
- Textbox placeholder must state what input is required
- Input must conform to the BIP39 word list

FR2: The application must output an xPubKey for the entered mnemonic seed

Acceptance Criteria:

- **T**he system must derive the correct and valid xPubKey from the entered mnemonic seed

---

FR3: The application must cross-reference the Blockchain to output known used wallet addresses

To find known used wallet addresses, the system will use a Blockchain API to find and retrieve wallets associated with the xPubKey.

Acceptance Criteria:

- The Blockchain API must return all known addresses associated with the xPubKey
- Wallet addresses returned via API must be displayed to the user in an easy-to-read format

---

FR4: The application must state the balances of all addresses found

The system will use the Blockchain API to retrieve and display wallet balances.

Acceptance Criteria:

- The Blockchain API must return the sum balance of all wallets found
- The Blockchain API must display the balance of each wallet individually
- Wallet balances must be displayed in a clear readable format such as £0,000,000.00

FR5: The application should provide an option to export all the retrieved information in PDF format

The application will enable the user to quickly and efficiently export data retrieved from the mnemonic seed analysis as a PDF.

Acceptance Criteria:

- The export to PDF button must be easily accessible and provide the user with a one-click export
- The PDF report must contain the mnemonic seed, the xPubKey, the retrieved wallet addresses, and the wallet balances.

4.2.2.1 Use Cases

The use case diagram shown below depicts the functionality of the system that the user can interact with. In this situation, the user of the system will be the forensic investigator who wants to analyse a cryptocurrency wallet recovery seed.

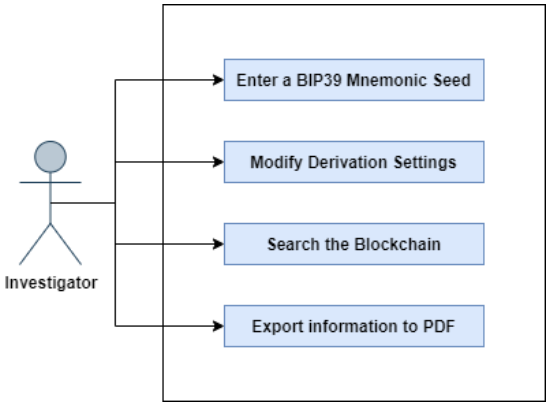


Figure 6: Actor and Use Cases

The use cases listed below detail the conditions of the application's main features and functionality.

ID: Use Case 1
Title: Bip39 Mnemonic Seed
Description: The user can enter a mnemonic seed
Primary Actor: User (Investigator)
Preconditions: The application has loaded
Main Flow:

1. The user has loaded the application
2. The user enters the BIP39 mnemonic seed that was gathered during an investigation
3. The application checks whether the entered seed is valid
4. The application derives a wallet public key and a wallet private key

Alternative Flow:
Post Conditions: A wallet public key and wallet private has been derived (with default derivation settings)

ID: Use Case 2
Title: Derivation Settings
Description: The user can modify the derivation settings
Primary Actor: User (Investigator)
Preconditions: A BIP39 mnemonic seed has been entered
Main Flow:

1. The user selects a derivation tab (BIP32, BIP44, BIP49, BIP84, BIP144
2. The user modifies the derivation settings i
3. A new wallet public key and a wallet private key are derived by using the derivation settings and mnemonic seed

Alternative Flow:
Post Conditions: A wallet public key and wallet private has been derived (using their entered derivation settings)

ID: Use Case 3
Title: Blockchain Search.
Description: The user can analyse the blockchain.
Primary Actor: User (Investigator).
Preconditions: A BIP39 mnemonic seed has been entered, derivation settings have been set, and a cryptocurrency has been selected for analysis.

Main Flow:

1. The user has clicked the 'Search Blockchain' button
2. The application fetches all form data entered by the user, and the results from derivation (Cryptocurrency selected, the derivation type selected, the wallet public key (xPubKey)
3. The application determines which Blockchain is required to be used for the analysis of the xPubKey
4. The application passes in the xPubKey into the appropriate Blockchain API.
5. The application fetches and retrieves all data associated with the xPubKey and sorts through the returned data
6. The application displays the wallet balances, and wallet addresses to the user

Alternative Flow:

Post Conditions: The user can view all wallet balances and wallet addresses that are associated with the wallet public Key that was derived from the BIP39 mnemonic seed and derivation settings that they had entered.

ID: Use Case 4

Title: Exportation of derived information

Description: The user can export the information that was retrieved during the analysis of the mnemonic seed

Primary Actor: User (Investigator)

Preconditions: The user has retrieved results from the blockchain analysis

Main Flow:

1. The user selects clicks the export button
2. The application compiles all the information used throughout the application (derivation settings, mnemonic seed, public and private keys, wallet address, balances, and transactions)
3. Information is formatted into a downloadable PDF

Alternative Flow:

Post Conditions: A downloadable PDF is created

4.2.3 Non-Functional Requirements

These non-functional requirements state how the system should behave what the user should expect from using the

1. The application should be available and functional 99% of the time.
2. The blockchain API should retrieve and display data to the user within 5 seconds.
3. The application text must be easy to read and understandable.
4. The application must gather all information and create a PDF for exportation within 10 seconds of the user clicking the export button.

4.3 System Design and Architecture

4.3.1 User Interface Design

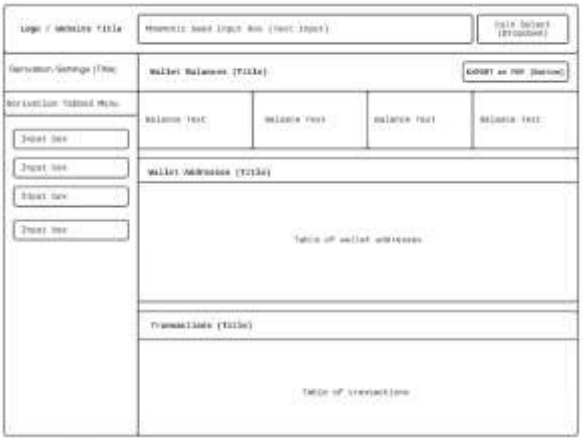


Figure 7: Initial Web Application Design

Overview

The wireframe picture above illustrates the layout and structure of the user interface that I will be designing for the cryptocurrency seed analyser application.

The application interface will be split into 3 main sections; the application header which will allow for seed input, a left column for derivation settings, and a central content area which will provide the output of the derivations and blockchain results.

This section of the report will describe these 3 layout areas, the content that they will contain, and the colour scheme that will be applied.

The Header

The website header will consist of a logo/website title, a text input box for the mnemonic seed, and a dropdown option to the side of it that will enable the user to select the coin/blockchain that will be analysed. The mnemonic seed input box has been placed at the top and inside the header so that it is always accessible. To ensure the input will always be accessible, the header will be fixed to the top of the page, so the page scroll does not affect its position.

Derivation Settings

The wallet address derivation settings have been placed in a column on the left side of the page. This allows for easy access and requires no page scroll, unlike the Ian Coleman website where the user is required to do a lot of scrolling. This section will contain a tabbed menu for the different BIPs (BIP32, BIP44, BIP49, BIP84, BIP141) and each tabbed content will provide the user with text input boxes that they can modify for the derivation process.

Main Website Content

The main website content area will consist of 3 sections. The top section which sits just under the website header will display the number of wallet addresses found, the total balance value of all these wallet addresses, the number of transactions made, and the money outflow/inflow. There will also be an export button in this section which upon clicking, will compile all the information derived from the seed analysis and blockchain process into a PDF file that can be downloaded by the user.

This section has been placed at the top under the header because it will display the most important and relevant information that the investigator will want to see first upon the seed analysis completion.

The second section in the main website body will list the wallet addresses that have been found through cross-referencing of the blockchain. The results from using a Blockchain API will be displayed as a table in this section with the columns, 'Wallet Address', 'Transactions Made', 'Money In', 'Money Out', and 'Total Wallet Balance'. The number of rows in this table will be determined by the number of unique wallet addresses retrieved.

The third section will be displayed just under the wallet address table. This section will contain another table that will display all the transactions made to and from all the wallet addresses that will have been found.

All this information will be displayed inside the report upon exportation as mentioned previously.

Colour Scheme

The website will be designed in a dark and light colour scheme. This will consist of a dark coloured background with white text in the left column and white background with black text in the main

website content area which will be used to display wallet addresses and balances.

I have chosen to develop the web application in these colours because they provide contrast, and it ensures that text and outputs are easily readable.

I have decided to use an orange colour within the application because the Bitcoin logo is orange and Bitcoin is the largest and most well-known cryptocurrency in the market.

I will be developing with the colour palette shown below:

Figure 8: Colour Pallet Used in the Application

4.3.2 Website Flow

The flowchart to the right describes the flow of actions that are performed when using the application from start to finish.

Once the web application had loaded, the user will be able to enter a BIP39 mnemonic seed and modify the derivation settings which will affect the output of the BIP39 seed and xPubKey that is derived.

The BIP39 mnemonic recovery seed must conform to the standardised BIP39 word dictionary else the application will not continue with the process until all entered words are valid.

If both the mnemonic seed and derivation settings they have specified are valid, the system will then proceed to the next stage in the process where a private seed and xPubKey are derived.

Once these values have been derived, the application will then use a Blockchain API and pass in the xPubKey as a search parameter. If the parameter is valid and the wallet address is not empty, then the application will calculate the total balance of all associated wallet addresses and then display out all information to the user including all the wallet addresses, the balance inside each wallet address, the number of transactions that have been made, and all the transactions that have been made to and from the wallets.

The user will then be provided a button to export all the information that has been derived and

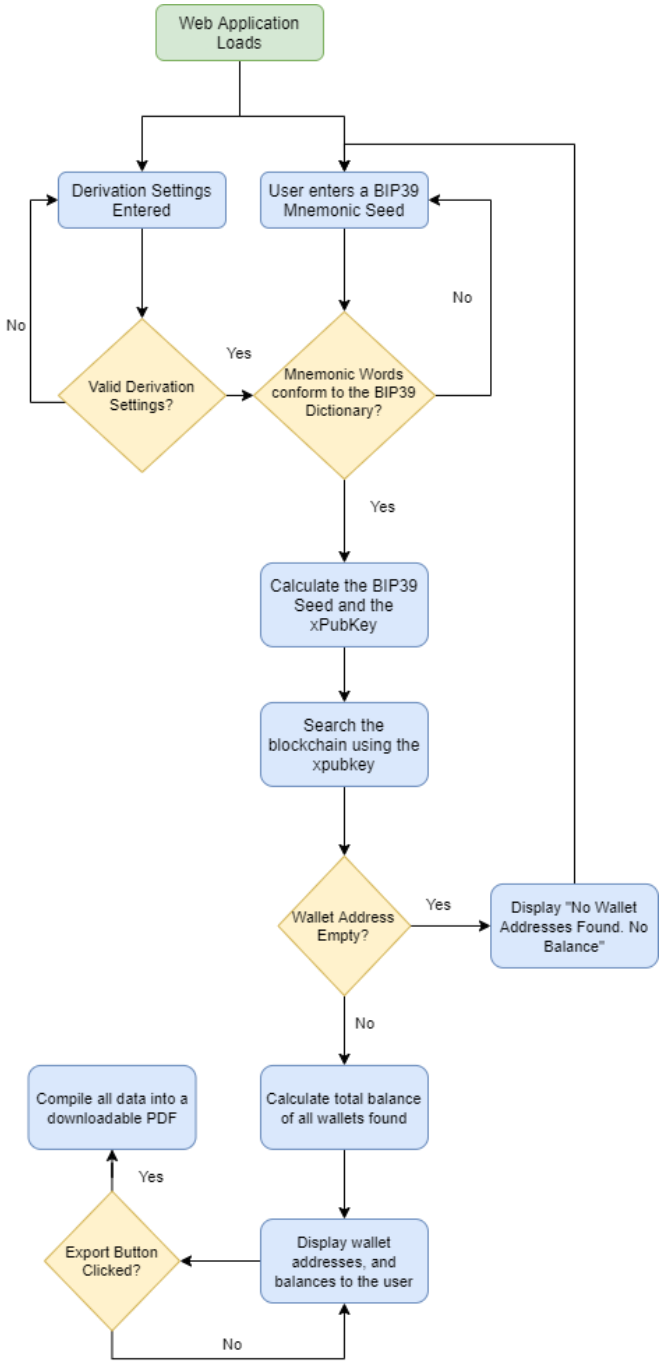


Figure 9: Flowchart of the Web Application

retrieved during the seed analysis process. This information will be compiled into a downloadable PDF which they will be able to access at any time without them being required to go through the entire process again.

4.3.3 Blockchain Analysis

The web application will require an algorithm that detects the blockchain that the user is requiring to be analysed and then retrieve results based on the derived wallet address. The blockchain analysis algorithm will utilise API connections and will pass in the wallet public key as a parameter and retrieve wallet balances and transactions. A flowchart diagram of the blockchain analysis process is structured below.

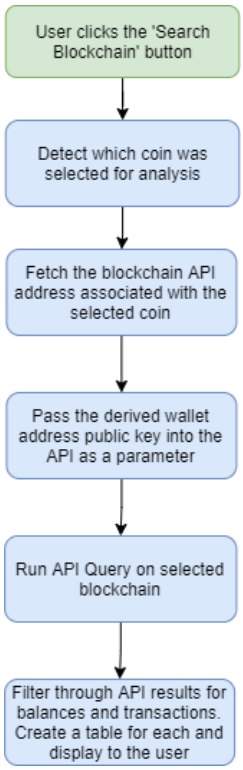


Figure 10: Blockchain Analysis Flowchart

Upon submission of the mnemonic seed and the coin to be analysed, the code I will be developing will store each of these values inside a variable that can be used and referenced when required. The cryptocurrency coin that the user has chosen to analyse against the entered mnemonic seed will be passed through a function that will determine which blockchain is required to be used in the next stage of the process. For example, a user selecting Bitcoin will return the blockchain.com API address from the function.

The derived wallet public key address will then be appended to the API address as a parameter before connecting to the API and executing the query.

The API query will then return any data associated with the passed-in wallet address public key parameter as JSON data. The section of JSON data that lists the wallet balances will be passed to a function that will process the balances and the wallet addresses will be passed through to another function. These 2 functions will then call another function whilst passing the JSON data as a parameter along with API headers, which will create visually neat tables that can be displayed to the user on the application.

4.3.4 Code Structure

The code I will be developing will be written into one class called 'Blockchain'. This class will be called upon user submission of the mnemonic recovery seed. The purpose of the Blockchain class is to process all user input and then find the appropriate cryptocurrency blockchain to perform the analysis on.

This analysis will include using an API to find the addresses, balances, and transactions of a specified wallet address. This class will then proceed to extract the relevant information from returned results and display the information out to the user in a clear and readable format which will be in the format of a table.

The function names, variables, data types, and purpose of each function are described below.

Function: blockchainNetwork
Parameters: None
Functionality:
- Detects which Blockchain API to use
- Returns an array containing the API address and API result headers

---

Function:
- processData

Parameters:
- blockchainData (Data Type: JSON Array)
- apiDetails (Data Type: Array)

Functionality:
- calls showBalances()
- calls showAddresses()
- calls showTransactions()

Function: showBalances

Parameters:
- balances (Data Type: JSON Array)

Functionality:
- Read balances
- Convert balances to a GBP format
- Display balances to the user

Function Name: showAddresses
Parameters:
- addresses (Data Type: JSON Array)
- apiAddressHeaders (Data Type: Array)

Functionality:
- calls createTable()

Function: showTransactions
Parameters:
- transactions (Data Type: JSON Array)

Functionality:
- calls createTable()

Function: createTable
Parameters:
- tableData (JSON Array)
- headers (Array)

Functionality:
- creates and populates a table with the data passed through to the function

Function: blockchainAPIConnection
Parameters: apiDetails (Array)
Functionality:
- Read array which contains the API website URL and API result headers which are required later
- Asynchronous API call function with results returned as JSON format
- Returns JSON format from function to be processed

Function: blockchainSearch
Parameters: None
Functionality:

- The first function that is called upon user submission
- Detects which derivation, and cryptocurrency coin was selected for analysis
- Creates and initialises a new Blockchain Class and passes in the wallet address public key and the select cryptocurrency network

Class Diagram

The following class diagram illustrates the functions and variables that will be used within the Blockchain class.

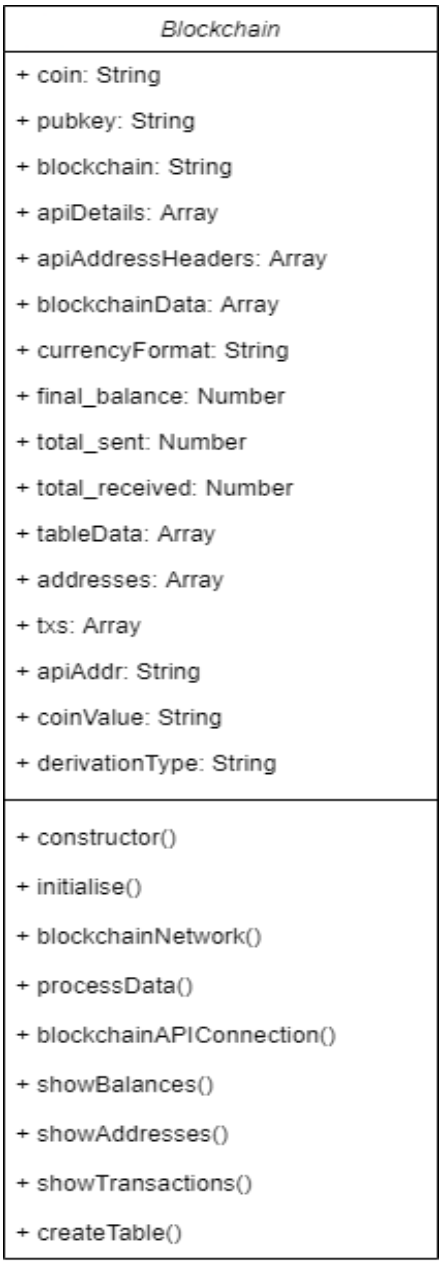


Figure 11: Methods and Variables to be Used

4.3.5 Pseudo Code

The following pseudo-code describes how the blockchain analysis will function. This code will be responsible for extracting details from the form submission, determine which blockchain is required for analysis, use API queries to search the selected blockchain, and then lastly, process the data and display the results to the user.

CLASS BLOCKCHAIN

Function: constructor

Declare variable: coin (String)
Declare Variable pubkey (String)

Function: initialise

Call function blockchainNetwork
Call function blockchainAPIConnection

Function: blockchainNetwork

Declare variable: blockchain (string)
Declare variable: apiDetails (Array)
Declare variable: apiAddressHeaders (Array)
Switch: Coin
Determine which blockchain is required
Set variable blockchain to a blockchain API address
Set variable apiAddressHeaders to the JSON keys that will be received from API results
Set variable apiDetails to blockchain and apiAddressHeaders
Return the apiDetails

Function processData(Array: blockchainData, Array: apiDetails)

Call function showBalances(JSON: wallet values)
Call function showAddresses(JSON: walletAddress)

Function showBalances(JSON Array: balances)

Read JSON Data: final wallet balance
Read JSON Data: total money sent
Read JSON Data: total money received

Convert balances into a currency format
Display balances to the user

Function create Table(Array: tableData, Array: tableHeaders)
Create element table
Create element table header
Create element table body

FOR header in tableHeaders:
    Create table header column with tableHeaders value
FOR the number of results in tableData:
    Loop through Array and create a row
    FOR data in tableData:
        Create a column inside a row
Append row to table body
Append table body to the table element
Return the table to the function that initiated

Function showAddresses(Array: addresses, Array: apiAddressHeaders)
Call function createTable(Array: addresses, Array: apiAddressHeaders)
Display table with addresses to the user

Function blockchainAPIConnection(Array: apiDetails)
Declare variable: apiAddr (String)
Declare variable: blockchainData (JSON Array)
Set variable apiAddr to API Blockchain URL
Run a query that connects to the blockchain API
Set variable blockchainData to API JSON results
Call function processData(JSON Array: blockchainData, Array: apiDetails)

END CLASS

Function blockchainSearch()
Declare variable: pubkey (String)
Declare variable: derivationType (String)
Declare variable coinValue (String)
Declare variable blockchainConnection (String)
Get derivation method selected by the user and store in variable derivationType
Get the value of the public key from chosen derivation method and store it in variable pubkey
Set variable coinValue to the user-selected cryptocurrency
Call and Create a new Blockchain Class(String: coinValue, String: pubkey)

5 Implementation

This section will outline the implementation process that was used to develop the cryptocurrency seed analyser application including the tools, technologies, and software that have been used, the final user interface design of the application, and explanations of the code that provide the functionality I have developed.

5.1 Technologies and Tools Used

The web application that I have developed was built using HTML, CSS, and JavaScript. The HTML and CSS were used for front-end design whilst the JavaScript was used for the backend functionality. This required an Integrated Development Environment, commonly known as an IDE. The IDE software that was used to code the application was 'Visual Studio Code' which is a code editor made and designed by Microsoft.

Existing software and code have been implemented into the application to provide the functionality which analyses the mnemonic private seed entered by the user to derive a list of potential wallet addresses. These wallet addresses are then used for the blockchain search & analysis that I have developed. This existing functionality has been implemented into the application because the Police (the client) stated that they are essentially requiring an expansion of the current existing seed analyser application that they are using *(BIP39 - Mnemonic Code. 2021).*

Before implementing this code from Ian Coleman, I investigated the licensing which has been listed under the MIT license *(The MIT License | Open Source Initiative. 2021)*. This license states that permission has been granted to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the software. After implementation of this functionality, I proceeded to develop and implement additional features on top such as analysing user input, cross-referencing and searching on the appropriate blockchain, retrieval and output of balances, and detecting known wallet addresses that have been used. Another functionality I implemented was the exportation of all the information that is extracted from the blockchain to a downloadable PDF. The code behind some of these functionalities I have discussed in Section 6.3, Blockchain Analysis & Code.

5.2 Final Implemented Design

The final design that I have created resembles a similar structure as drawn and detailed in section 5.3.1, User Interface Design. Figure 12 below depicts the original proposed design for the application.

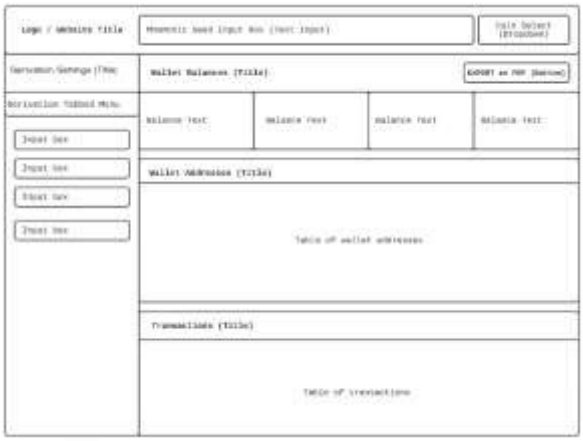


Figure 12: Initial Web Application Design

Throughout the development of the user interface, some structural modifications were made. Details of the modifications made to the design will be described under their respective headings below. Figure 13 is the result of the application user interface design.

Figure 13: Final Web Application Design

Colour Scheme

The colours used throughout the application have remained the same as initially described.
The left column background has used the hexadecimal colour #1F2025. I have made this section darker than the rest of the application to provide some contrast to the white colour used and to attract the user's attention to the derivation settings that they can modify.
The rest of the application has used a white colour (#FFFFFF) for simplicity and readability. Some heading titles and the information submitted by the user have been displayed in an orange colour (#F7931A) to ensure they stand out to the user as the values associated with them are the important values returned by the blockchain search.

The Header

The header has had the biggest modification out of all sections compared to the original design. Originally, the header, shown in Figure 14 below, would have taken up the full width of the application and be positioned along the top.

Figure 14: Initial Header Design

However, whilst designing the application, I decided to remove the header and move the content initially in this section into a smaller section that is now situated at the top of the main body content.
The final 'header' content can be seen in Figure 15 below which consists of a title, an input box for the mnemonic wallet recovery seed, a dropdown menu for the cryptocurrency network they would like to be analysed, a button that upon clicking will connect to a blockchain and fetch details associated with the derived xPubKey, and lastly, an export button which will extract all information into a PDF that the user can download.

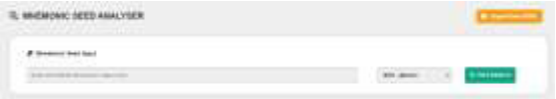


Figure 15: Final Header Design

Derivation settings

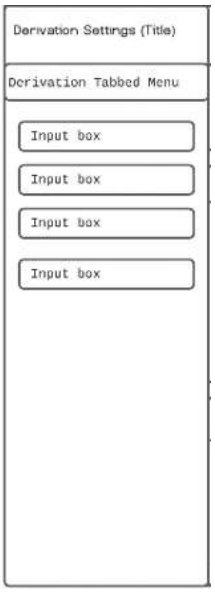


Figure 16: Initial Derivation Settings Design

The design of the derivation settings has not changed much compared to the original prototype which can be seen in Figure 16.
The derivation settings section consists of a tabbed menu row. These tabs provide the user with different derivation methods. BIP32, BIP44, BIP49, BIP84, and BIP141. Below the tabbed menu, there is a BIP32 Derivation Path which details the derivation method being used.
The input boxes below allow for modification of derivations. The 'purpose' is the BIP number (Bitcoin Improvement Proposal), the 'coin' input is auto-filled with a value based on the cryptocurrency that the user selects (positioned next to the mnemonic seed input). The 'Account' is the starting index of the wallet addresses that is used upon derivation and the External/Internal is whether an internal chain/ external chain is to be used. Internal are wallet addresses that are not visible outside of a wallet, and external are wallet addresses that are meant to be visible outside of the wallet.
These four inputs, along with the entered mnemonic seed are used to derive the wallet xPubKey. This functionality comes from the existing code that has been implemented.
Upon derivation, the Account Extended Private Key, and the Account Extended Public Key (referenced as xPubKey throughout the report) are displayed below.

Figure 17: Final Derivation Settings Design

Main Body Content

The main body content is where the wallet balance totals and a list of used wallet addresses with value inside them are displayed out to the user. The image below illustrates the layout of the initial design for the application.

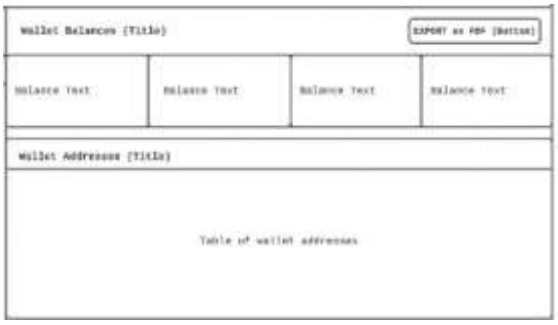


Figure 18: Initial Main Body Content Design

The final design of the main body content that I have implemented can be seen below in Figure 19. This design is displayed to the user before they have entered the mnemonic seed and started the blockchain analysis process. The final design has stayed very accurate to the initial drawing. Both contain a row that outputs the balances and another section underneath which lists all associated wallet addresses.

Figure 19: Final Main Body Content Design

One new section that I have added here is the 'Search Results for'. This contains all the information and settings that the user-specified such as the mnemonic seed, the derivation path, and the cryptocurrency blockchain that they would like to be analysed.

The 'Export as PDF' button has been moved from this section into the header, next to the application title.

Upon clicking the 'Search Blockchain' button which can be seen in Figure 15 in the section 'The Header', the application will fill the content with the wallet address, and its balance as well as the total money received and sent to each address.

See Figure 20 for an example of the final output that is shown to the user.

Figure 20: Output of a Successful Wallet Analysis

If no addresses from the blockchain results contain a balance, an output stating 'No Results Found' will be displayed to the user. This will indicate that the wallet addresses associated with the mnemonic seed entered are empty and contain no value/cryptocurrency.

Figure 21: Output of No Results Found

Exportation of Results:

After the analysis process, the user will be able to export all the information returned from the blockchain analysis results as a downloadable PDF. The PDF that is created upon clicking the 'Export' button, will essentially be a snapshot of the main body content where all the information used for derivation, the total calculated balances of the analysed wallet, and the wallet addresses a. re-listed. An example of the PDF output can be seen in Figure 22.

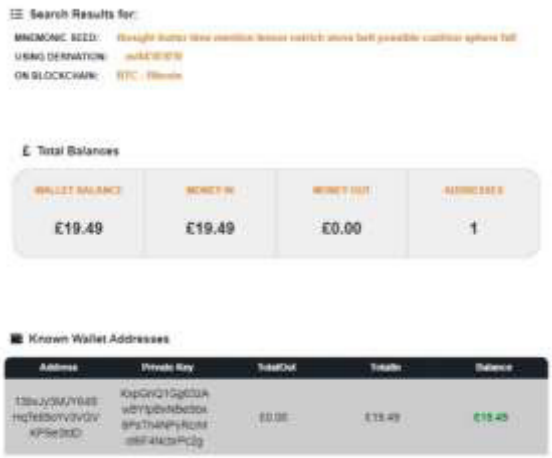


Figure 22: Output of a PDF Exportation

5.3 Blockchain Analysis & Code

Overview

This section will describe how the functionality of the application has been implemented as well as the code to some of the functionality which includes extracting information from the blockchain, processing and outputting the relevant information to the user and the exportation of the information as a PDF.

5.3.1 Changes to the Implementation

During the design and specification stage of the project, it was initially planned to use the derived xPubKey (Account Extended Public Key) to search the blockchain and retrieve all known/used wallet addresses. However, when it came to the implementation stage, I realised this would not be possible due to many blockchain API's not allowing an xPubKey to be entered, let alone a yPubKey, and zPubKey (derived using a BIP49, and BIP84 derivation Path respectively).

It was then decided that the most appropriate way of implementing the blockchain analysis would be to use a list of individual wallet addresses that are derived upon entering a mnemonic seed and the derivation settings. The derivation of these wallet addresses is created from the Ian Coleman code. Upon the derivation of these addresses, I created an array that would then store these addresses globally so that they would be accessible from within my code.

I have implemented the blockchain search functionality for two of the most common cryptocurrency blockchains which are Bitcoin and Ethereum. Initially, the plan was to include all possible cryptocurrencies that are available on the Ian Coleman website (which the Police have currently been using to derive wallet addresses). However, there are over 150+ cryptocurrencies listed on Ian Coleman and therefore the implementation of all the different Blockchain APIs required to achieve this would take a very long time to add.

The two Blockchains APIs that I have implemented are Blockchain.com (for Bitcoin), and Ethplorer-API (for Ethereum) *(EverexIO/Ethplorer. 2021).* I have chosen to use the Blockchain.com API because it is one of the most common and most well-known blockchain search websites for Bitcoin. Although Blockchain.com allows users to search Ethereum addresses, their API documentation did not support the Ethereum blockchain. Therefore, I was required to search elsewhere for an Ethereum API. After extensive research, Ethplorer-API appeared to be the best option to use for Ethereum wallet address analysis. Many other blockchain APIs that I came across required a paid subscription or did not provide the details that I required from a search.

Both of these APIs allow for a wallet address to be passed through as a parameter which will then return a set of results in a JSON format. This returned JSON data is then processed by reading through the JSON data set and extracting the relevant and important information that is required for processing and output to the user.

The blockchain.com API allows for an xPubKey to be passed through, as initially planned. However, it does not allow for a yPubKey, and a zPubKey to be passed through. The Ethplorer API does not allow for any xPub, yPub, or zPub to be passed in as a parameter. Therefore, the decision was made to individually pass in each potential wallet address derived and retrieve the results via that method.

5.3.2 Functionality Implementation

In this section, I will be describing the step-by-step process of the application's functionalities implemented along with the code that I have created. Due to the client (the Police) suggesting that a Web Application would be most suitable for them, I decided to use the programming/scripting language called JavaScript. JavaScript is a very common and popular client-side scripting language that is used amongst many websites to provide functionality and interaction.

The code I have created is stored within a class object called 'Blockchain' which is instantiated and initialised upon the user clicking the 'Search Blockchain' button. A list of all the methods and variables with their data types that have been used within my code and the 'Blockchain' class can be seen in the table below.

Methods and Variables:

Method	Variables
Constructor()	this.coin(String) this.walletAddresses (Array) this.devType (String) this.network (String)
Initialise()	--
Bip44Blockchains()	var blockchain44 (String) var apiDetails44 (Array) var ext (String) var type (String) var delimiter (String)
Bip49Blockchains()	var blockchain49 (String) var apiDetails49 (Array) var ext (String) var type (String) var delimiter (String)
Bip84Blockchains()	var blockchain84 var apiDetails84 (Array) var ext (String) var type (String) var delimiter (String)
BlockchainNetwork()	var apiDetails (Array)
processData()	var balance_totals (Object) var addresses_balances (Array) var temp_address_list (Object) var balance_all (Number) var total_out (Number) var total_in (Number) var ethw (Number) var btcw (Number) var blockchainData (Object)
crypto2gbp()	var gbpvalue (Number) var conversion (Number) var result (Number)
showBalances()	var balances (Object) var final_balance (Number) var total_sent (Number) var total_received (Number)
showAddresses()	var addresses (Object)
createTable()	var table (Element) var tableHead (Element) var tableBody (Element) var addrCount (Number) var a (String) var i (Number) var tableData (Object) var vals (String) var x (String) var y (String) var z (String) var privateKey (String)
blockchainAPIConnection()	var apiAddr (String) var apiExt (String) var apiDelimiter (String) var apiValues (String) var apiURL (String) var apiSingleAddresses (Array) var blockchainResponse () var blockchainData (Object) apiLimitInterval (Function) var c (Number) var j (Number)
fetchGBPValues()	var gbpquery () var priceData (Object)
blockchainSearch()	var derivationPath (String) var walletSeed (String) var derivationType (String) var derivationPath (String) var coinValue (Number) var networkName (String)

Main Code:

Once the user has entered the mnemonic seed and set the derivation settings and has clicked the 'Search Blockchain' button, the function 'blockchainSearch is called and executed. This following set of code declares the variables and retrieves the mnemonic seed the user has entered along with the type of derivation that they have specified.

```
function blockchainSearch() {
var derivationPath = "";
var walletSeed = DOM.WS.value;
var derivationType = document.querySelector(".derivation-type > li.active").id;
```

The switch statement below then detects the type of wallet derivation used and retrieves the used derivation path. For example, m/0'/0'/0'.

```
switch (derivationType) {
case "bip32-tab":
derivationPath = DOM.DP32;
break;
case "bip44-tab":
derivationPath = DOM.DP44;
break;
case "bip49-tab":
derivationPath = DOM.DP49;
break;
case "bip84-tab":
derivationPath = DOM.DP84;
break;
case "bip141-tab":
derivationPath = DOM.DP141;
break;
}
```

The selected cryptocurrency and the blockchain required for the analysis are then extracted.

```
// get coin
var coinSelect = DOM.NP;
var coinValue = coinSelect.value;
var networkName = coinSelect.options
    [coinSelect.selectedIndex].text;
```

These values are then outputted into the main body content section under 'Search Results For:'

At the bottom of the blockchainSearch function, a new class called 'Blockchain' is instantiated with parameters passed in such as the cryptocurrency selected, the list of derived addresses (derived from Ian Coleman code), and the type of derivation that the user had selected.

```
DOM.LD.style.visibility = "visible";
DOM.MS.innerHTML = walletSeed;
DOM.DU.innerHTML = derivationPath;
DOM.BU.innerHTML = networkName;
DOM.AO.innerHTML = "";

// create new blockchain connection
var blockchainConnection = new Block
    chain(coinValue, networkName, wallet
    AddressList, derivationType);

// initiate the blockchain process
blockchainConnection.initialise();
```

The Blockchain Class

The Blockchain class contains all the information and methods required to build a Blockchain API query, execute the query, and then process all the data from the query and display it out to the user in a tidy format.

The code below creates a class called 'Blockchain'. The constructor processes the parameters passed into the class and stores them in variables for ease of access within the code later. The initialise function, once called, calls the first method, 'Blockchain Network', and then passes the return value into the method 'Blockchain API Connection'.

```
class Blockchain {

constructor(coin, networkName, walle
    tAddresses, derivationType) {

this.coin = coin;
this.walletAddresses = walletAddress
    es;
this.devType = derivationType;
this.network = networkName;
}

initialise() {
this.blockchainAPIConnection(this.bl
    ockchainNetwork());
}

```

Method: blockchainNetwork

The blockchainNetwork method detects which derivation type was chosen by the user and then calls the appropriate function to fetch the details required for the API connection.

```
blockchainNetwork() {
var apiDetails = [];
if (this.devType === "bip32-tab") {
apiDetails = this.bip44Blockchains()
}
else if (this.devType === "bip44-
    tab") {
apiDetails = this.bip44Blockchains()
}
else if (this.devType === "bip49-
    tab") {
apiDetails = this.bip49Blockchains()
}
else if (this.devType === "bip84-
    tab") {
apiDetails = this.bip84Blockchains()
}

return apiDetails;
}
```

Bip44Blockchains method()

An example of one of the Blockchain methods for a derivation selected can be seen below. If a BIP44 derivation type was selected by the user then this method would be called and executed.

This method detects which cryptocurrency was selected and then assigns the necessary details required to build a full URL for the API query to variables such as the base API URL, whether the addresses can be appended into one query (multi), or individual requests will need to be sent for each wallet address (single). These variables are then stored inside an array which is returned and passed through to another method called 'blockchainAPIConnection'.

The two cryptocurrencies which have been implemented, as previously mentioned, are Bitcoin and Ethereum.

```
bip44Blockchains() {

var blockchain44 = "";
var apiDetails44 = [];
var ext = "";
var type = "";
var delimiter = "";

switch (this.network) {

case "ETH - Ethereum":
blockchain44 = "https://api.ethplorer.io/getAddressInfo/";
ext = "?showETHTotals=true&apiKey=EK-2VyLa-DhJ5UYC-qd5bU";
type = "single";
delimiter = "";
break;

case "BTC - Bitcoin":
blockchain44 = "https://blockchain.info/multiaddr?active=";
ext = "";
type = "multi";
delimiter = "|";
break;
default:
blockchain44 = "None";
}

apiDetails44.push(headers, blockchain44, ext, type, delimiter);
return apiDetails44;
}

```

Method: blockchainAPIConnection

The purpose of the method 'blockchainAPIConnection' is to build a full API URL that can be used within a POST request to fetch data from the blockchain API servers.

The code below defines all the variables for this section and reads data from the array created previously.

```
blockchainAPIConnection(apiDetails) {
var apiAddr = apiDetails[1];
var apiExt = apiDetails[2];
var apiType = apiDetails[3];
var apiDelimiter = apiDetails[4];
var apiValues = "";
var apiURL = "";
var apiSingleAddresses = [];
```

The code below is an asynchronous function that makes an HTTP request to the appropriate Blockchain API. The full API URL is used here. The results from this request are returned in a JSON format that is then processed in the next method.

```
async function blockchainQuery() {
let blockchainResponse = await fetch(apiURL);
let blockchainData = await blockchainResponse.json();
return blockchainData;
```

If the Blockchain API has to run with 1 wallet address at a time, then the code below is used. For this code, I have implemented a setInterval function. This means the code is run every 0.2seconds to prevent exceeding API request limits set by the API providers.

The code here is also used to create a full API URL that includes the base URL (the domain), and the address to be searched and then the asynchronous blockchainQuery function above is called which uses the full API URL just created. The JSON data return is then stored inside an array that is then sent through to the 'processData' method.

```
if (apiType == "single") {
var c = 0;
var apiLimitInterval = setInterval(function() {

if (c >= walletAddressList.length) {
clearInterval(apiLimitInterval);
this.processData(apiSingleAddresses, apiDetails, apiType);
}
else {
apiURL = apiAddr + walletAddressList[c]["address"] + apiExt;
blockchainQuery().then((blockchainData) =>
apiSingleAddresses.push(blockchainData)
);
c += 1;
}
}.bind(this), 200);
}
```

The code below is similar to the one described above, however, this code below is executed if the Blockchain API provider allows for multiple addresses to be sent at once and therefore the query only needs to be run once. In this code, a delimiter is appended at the end of each wallet address as it is concatenated against the previous address.

Therefore, this code builds one long API URL that can be sent and processed in one go, rather than through many iterations as is required in the "apiType == 'single'" code.

```
if (apiType == "multi") {

for (var j = 0; j < this.walletAddresses.length; j++) {
apiValues += this.walletAddresses[j]['address'] + apiDelimiter;
}
apiURL = apiAddr + apiValues + apiExt;
blockchainQuery(apiAddr).then((blockchainData) =>
this.processData(blockchainData, apiDetails, apiType)
);
}
```

Method: ProcessData

This method processes the JSON data received from the API request by reading, extracting, and storing the relevant data into a new array.

```
processData(blockchainData, apiDetails, apiType) {

var balance_totals = {};
var addresses_balances = [];
var temp_address_list = {};
var balance_all = 0;
var total_out = 0;
var total_in = 0;

```

This method also detects which blockchain network was selected. If the blockchain was Ethereum, then the code below is run. Similar code has been constructed for Bitcoin with a slight variant.

The code below will loop through the number of records received by the Blockchain API and will extract the balance, money in, and money out for each wallet address. The totals will be calculated for each of these and a separate array list for each wallet address and their balances are stored ready to be displayed out to the user in a table format.

```
if (this.network === "ETH - Ethereum") {

for (var ethw = 0; ethw < blockchainData.length; ethw++) {
balance_all += blockchainData[ethw]["ETH"]["balance"];
total_out += blockchainData[ethw]["ETH"]["totalOut"];
total_in += blockchainData[ethw]["ETH"]["totalIn"];

temp_address_list = {};
temp_address_list["address"] = blockchainData[ethw]["address"];
temp_address_list["totalOut"] = blockchainData[ethw]["ETH"]["totalOut"];
temp_address_list["totalIn"] = blockchainData[ethw]["ETH"]["totalIn"];
temp_address_list["balance"] = blockchainData[ethw]["ETH"]["balance"];
addresses_balances.push(temp_address_list);
}

balance_totals["final_balance"] = balance_all;
balance_totals["total_sent"] = total_out;
balance_totals["total_received"] = total_in;

this.showBalances(balance_totals);
this.showAddresses(addresses_balances, apiDetails[0]);
}
```

Method: createTable

This method is responsible for displaying all the information out to the user in table format. The information shown will include the total balance of the wallet, the total money in, the total out, the wallet address, the balance of the address, the money received and sent to/from the address, and its private key (which is used to access a wallet).

The code listed below creates the structure of the table. This includes iterating through a predefined list of table headers and then create a table header element for each before appending to the main table element.

```
createTable(tableData, headers) {
var table = document.createElement("table");
var tableHead = document.createElement("thead");
var tableBody = document.createElement("tbody");
var addrCount = 0;
table.appendChild(tableHead);

for (var i = 0; i < headers.length; i++) {
if (i == 1) {
tableHead
.appendChild(document.createElement("th"))
.appendChild(document.createTextNode("Private Key"));
}
tableHead
.appendChild(document.createElement("th"))
.appendChild(document.createTextNode(headers[i]));
}

```

```
table.setAttribute("class", "table known-addresses");
```

The next stage within this method is to loop through the table data and check which wallet addresses contain a current balance or have previously had some money within in it. If a wallet address matches this criterion then it will loop through a set of data that contains derived wallet addresses, and its public key for a match. If there is a match between the two wallet addresses, then the private key will be retrieved and stored ready for output.

```
for (i = 0; i < tableData.length; i++) {
if (tableData[i]['balance'] !== 0 || tableData[i]['totalIn'] !== 0) {
var row = document.createElement("tr");
for (var j = 0; j < this.walletAddresses.length; j++) {
var y = JSON.stringify(this.walletAddresses[j]["address"].toUpperCase());
var z = JSON.stringify(tableData[i]["address"].toUpperCase());

if (y == z) {
var address = this.walletAddresses[j]["address"];
var privateKey = this.walletAddresses[j]["privkey"];
}
}
addrCount += 1;
```

The processed JSON data retrieved from the Blockchain API is then looped through one dataset at a time. A table cell for each value in each dataset is created. The balance, totalIn, and totalOut fields are converted into a GBP value (from a Satoshi or Gwei value, which are the smallest units that make up a Bitcoin/Ethereum). The value is then converted into a readable GBP format, for example, 1000.00 to £1,000.00

```
for (var x = 0; x < headers.length; x++) {
if (x == 1) {
var cell = document.createElement("td");
cell.textContent = privateKey;
row.appendChild(cell);
}

var a = headers[x];
var vals = tableData[i][a];
var cell = document.createElement("td");

if (x >= 3) {
cell.textContent = currencyFormat.format(this.crypto2gbp(vals));

if (tableData[i]['balance'] !== 0) {
cell.setAttribute("style", "color: #009432; font-weight:bold");
} else if (tableData[i]['totalIn'] !== 0) {
cell.setAttribute("style", "color: red; font-weight:bold");
}
} else if (x > 0 && x < 3) {
cell.textContent = currencyFormat.format(this.crypto2gbp(vals));
} else {
cell.textContent = vals;
}
```

Once a dataset within the JSON array has been looped through and a table cell for each value has been created, the row will be appended to the table. Once all datasets within the JSON array have been processed, the table is displayed out to the user, as illustrated in the previous section.

```
row.appendChild(cell);
}
tableBody.appendChild(row);
}
}
table.appendChild(tableBody);
DOM.AC.innerHTML = addrCount;

return table;
}

```

6 Testing

The test cases described below will be tested on the application that I have developed to ensure that the application meets the requirements set out in the brief and the functional/non-functional requirements that were detailed in the 'Specification & Design section. These test cases will be tested as though the user is using the application and they are trying to derive balances, and wallet addresses from an entered mnemonic seed. The table below is a summary of all the tests that have been made and their outcome. Details and comments about the tests performed will be described underneath.

Summary of Test Cases		
Test #	**Test Title**	**Status**
1	Application Loads	PASS
2	Mnemonic Seed Input	PASS
3	xPubKey Derivation	PASS
4	Wallet Address Derivation	PASS
5	Search Button	PASS
6	Bitcoin Blockchain	PASS
7	Ethereum Blockchain	PASS
8	Used Wallet Address	PASS
9	Balance Calculations & Formatting	PASS
10	Timing of Results	PASS

To test the application, I deposited £20 worth of Bitcoin into a Bitcoin Wallet that I was able to retrieve a mnemonic seed for. The mnemonic seed that I will be using for testing is:

- *"thought butter time mention lemon ostrich stove belt possible cushion sphere fall"*

This mnemonic seed has been used throughout the testing to determine used/known addresses and if the balances that have been retrieved are calculated and displayed correctly. A randomly generated mnemonic seed with wallet addresses that have no value attached to them was also used within the testing process to check whether the application provides a valid response when no balances are returned.

The test cases outlined below describe the testing performed on each process of the application's functionality to ensure that they work as expected. Additional comments have also been provided under each test to describe what I had done to test the functionality or the values that have been used or received from the tests.

Test Case 1:
Title: Application Loads
Description: Testing of the application to ensure it loads without any errors
Expected: The web application should load all UI elements and be ready for use without any errors
Actual: Web application loads. No Errors. Criteria met.
Comments: None
Status: PASS

Test Case 2:
Title: Mnemonic Seed Input
Description: Testing of the mnemonic seed input. The user must be able to enter a valid mnemonic seed.
Expected: The user should be able to enter a wallet mnemonic seed into the application
Actual: A valid mnemonic seed was able to be entered into the application
Comments:
The Mnemonic seed entered:

- *thought butter time mention lemon ostrich stove belt possible cushion sphere fall*

Status: PASS

Test Case 3:
Title: xPubKey, or yPubKey, or zPubKey derivation (Bitcoin) and Ethereum
Description: A correct public key is derived from the entered mnemonic seed. In the format of either an xPubKey, yPubKey, or a zPubKey
Expected: The application should derive a public key from the mnemonic wallet recovery seed
Actual: The application successfully derived the three different types of public keys for both the Bitcoin and Ethereum Blockchain.
Results for Bitcoin:
Derived xPubKey:
xpub6C54TtTm2WQTcaG9QqaoWc8zN1qsLViZBjJyo78nb8tteZimzSKXpeth5o78hB4c5yVYzJh7LB4YYhG788Rijer1UbMXYFruLTnahKDWUEY

Derived yPubKey:
ypub6Xz6RPG8ov8jyY18vh3cBjo5Szc4VrCWVkyXtbdpxgb2nv1YcgB8fp9PYuiNGN5DePAF2PTJaJJSCu9igomxpeyPp5Cxe85rVmVpFkdP1zj
Derived zPubKey:
zpub6rSqYJiq2BKEb3Hdt61QMCqFDRpTgLYgskt9CpyJLeRhSYGsUA27Jg9aEDb2xUBhXwyrJ4xs7xjLreHnbfpVeXvL7g411VcJW9mh9tLv5Xe

Results for Ethereum:
Derived xPubKey:
xpub6CFA4ktgkBwAHJjugXPux1E2C4eAnRR9TFUka8zfpBZ472WyiBYoEFpCoBvvMb8KLrX3Bz19dwPWVTDeqxdvuMCjxCLLd9EozAGttZKTjeA
Derived yPubKey:
ypub6YEHtKDmWnWqaVxQA5AozP31z2E1ptvt7in6r1aJaSQFUzUGGLA2JxjLrv9dYw26itBTDxg5VRRNt4gDN4JTMKCkpHJ7ad7t1zu2AFNJvLH
Derived zPubKey:
zpub6r3i14RMmJzJ7zrdJqiBVsWtdSoqYQafdD7yQtNeixrKWoNKESMpTanQwYd3dXT2VLt2KFzSceWghM5TUs7bAUkCyVSM7yAVMsMxhRxJ3Ja
Comments:
- Used the same mnemonic seed as previously commented.
- The functionality of this test case was implemented from the Ian Coleman Code.
- These derivations are no longer used within the functionality I implemented due to the changes mentioned previously in the report.

Status: PASS

Test Case 4:
Title: Wallet Address Derivation
Description: Test that a list of potential wallet addresses and their private keys associated with the previously derived public key are derived.
Expected: Wallet addresses and private keys are derived successfully from the public key
Actual: Same as expected.
Screenshot of addresses and private keys generated:

Figure 23: A List of Derived Wallet Addresses

Comments:
- This functionality is implemented from the Ian Coleman Code.
- The wallet addresses used are stored within an array and are used later within the functionality I have created.
- The wallet addresses are created from the mnemonic seed previously mentioned.
- The array of wallet addresses, and private keys have been printed to the browser console log as shown in the screenshot above.

Status: PASS

Test Case 5:
Title: Search Button
Description: Test that the code I've created is called and ran as expected upon the user clicking the 'Search Blockchain' button.
Expected: The application should work as expected and the Blockchain Class should be successfully initialised upon the user clicking the 'Blockchain Search' Button.
Actual: The blockchain class code upon the user clicking the button successfully worked. The browse console log received outputs when the user clicked the search button.
Screenshot:

Figure 24: Blockchain Class Being Initialised

Comments:
- To test this functionality and check that the class object is initialised upon clicking the search button, I added temporary code that will print out the text to the browser console log when each stage of the code is executed.

Status: PASS

Test Case 6:
Title: Bitcoin Blockchain
Description: Test that the correct Bitcoin blockchain API works correctly and that data is returned from the API request.
Expected: The application should connect to the Bitcoin Blockchain API and successfully retrieve a set of results associated with the wallet addresses being analysed.

Actual: Bitcoin Blockchain API connection was successful. Data from the Bitcoin Blockchain was returned as a set of JSON data.

Screenshots:

Figure 25: Bitcoin Blockchain Analysis Results

Status: PASS
Comments:

- These wallet addresses and balances have been returned from the Bitcoin Blockchain API using the mnemonic seed mentioned previously.
- Data was retrieved successfully without any errors

Test Case 7:
Title: Ethereum Blockchain
Description: Test that the Ethereum blockchain API works correctly and that data is returned from the API request.
Expected: The application should connect to the Ethereum Blockchain API and successfully retrieve a set of results associated with the wallet addresses being analysed.
Actual: Ethereum Blockchain API connection was successful. Data from the Ethereum Blockchain was returned as a set of JSON data.
Screenshot:

Figure 26: Ethereum Blockchain Analysis Results

Comments:

- These wallet addresses and balances have been returned from the Bitcoin Blockchain API using the mnemonic seed mentioned previously.
- Data was retrieved successfully without any errors

Status: PASS

Test Case 8:
Title: Used Wallet Addresses
Description: List all wallet addresses that contain a valid balance and have been used.
Expected: Used wallet addresses should be displayed to the user. If no wallet addresses have been used, then a message stating that there were no results found should be displayed.
Actual: Wallet Addresses that have been used were shown to the user in a table format. 'No Results Found' was shown when no used wallet addresses were found.

Screenshots:

Figure 27: Possible UI Outputs from the Blockchain Analysis

Comments:

- The first screenshot is the output of the mnemonic seed used for testing (*thought butter time mention lemon ostrich stove belt possible cushion sphere fall)* returned an address that contains a balance (Success). This is a wallet address that I deposited some Bitcoin in.
- The second screenshot is the output of a randomly generated mnemonic seed that has no balances attached was used. No Results Found was returned (Success)

Status: PASS

Test Case 9:
Title: Balance Calculations & Formatting
Description: Balances should be converted from Satoshi value (units that make up a Bitcoin) or Gwei (units that make up an Ethereum) and displayed as a numeric and formatted GBP value
Expected: The application should display a converted Satoshi or Gwei value into a GBP format (£0,000.00)
Actual: Wallet Balances are converted and displayed successfully as a GBP formatted value.

Screenshots:

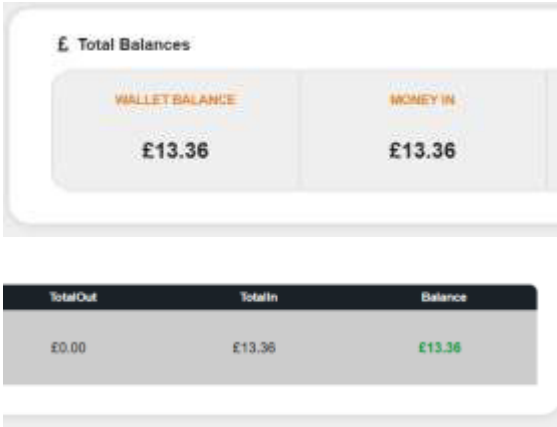


Figure 28: Example Output of a Successful Wallet Analysis

Comments:

- The Satoshi value received from the Bitcoin Blockchain API was 48598. This value is converted into a decimal Satoshi format (0.00048598) and then converted and formatted into the GBP value displayed above.

Status - PASS

Test Case 10:
Title: Timing of Results
Description: A list of used wallet addresses with their balances should be processed and displayed to the user within 5 seconds.
Expected: Wallet addresses and balances to be shown to the user within 5 seconds of the user clicking the 'Search Blockchain' button.
Actual: Results were retrieved, processed, and displayed within the 5-second objective.
Screenshots:

Bitcoin (Processing Wallet with a Balance)	
Test	Excution Time
1	112
2	131
3	107
4	384
5	293
6	209
7	113
8	118
9	100
10	117
Average:	168.4 ms

Figure 29: Times Taken: Bitcoin Blockchain (Wallets with a Balance)

Bitcoin (Processing Wallet with NO Balances)	
Test	Excution Time
1	94
2	71
3	100
4	144
5	90
6	117
7	77
8	118
9	96
10	106
Average:	101.3 ms

Figure 30: Times Taken: Bitcoin Blockchain (Wallets with No Balances)

Ethereum (Processing Wallet with NO Balances)	
Test	Excution Time
1	4232
2	4229
3	4237
4	4228
5	4236
6	4238
7	4232
8	4222
9	4236
10	4226
Average:	4231.6 ms

Figure 31: Times Taken: Ethereum Blockchain (Wallets with No Balances)

Comments:

- The results timings returned are in milliseconds which are counted from the user pressing the search button and the results being displayed out to the user
- An average of the timing results was created from 10 tests on each blockchain with and without known wallet addresses.
- The application took an average of 168ms when processing a wallet with a used wallet on the Bitcoin Blockchain
- The application took an average of 101ms on a Bitcoin wallet with NO used wallet addresses or balances.
- The application took an average of 4231ms on the Ethereum Blockchain with no wallet addresses. This is due to having to implement an API request limit on the Ethereum Blockchain request which runs every 0.2seconds and due to each Ethereum wallet address having to be processed individually.

Status: PASS

7 Future Work

The web application that has been developed has met all the requirements that were intended and outlined in the functional and non-functional requirements. However, there is still a lot of room for further expansion of the application which can enable a wider variety and more in-depth analysis, as well as providing more features and functionality. This section will discuss the potential future improvements and features that can be implemented into this web application.

7.1 Future Implementation Idea 1:

Due to time constraints, only a limited of cryptocurrency blockchain APIs could be implemented into the application. As of now, only Bitcoin and Ethereum blockchains have been added. Future work could include implementing more blockchain APIs to provide a wider range of cryptocurrency options for the analyst using the platform. The Ian Coleman website, which the developed functionality has been built over and improved on currently has 202 cryptocurrencies in a select dropdown that potential wallet addresses can be derived for.

Implementing more of these cryptocurrencies' blockchain APIs into the application would help for greater and wider analysis and help save the time of those investigating a mnemonic wallet seed to determine the blockchain used and the amount of cryptocurrency that has been stored within wallet addresses associated with the entered mnemonic wallet seed.

7.2 Future Implementation Idea 2:

Additional functionality such as an 'Auto-Analyse' feature could potentially save the investigator using the web application a lot of time. The idea behind this feature would be for the application to loop through each of the cryptocurrency blockchains and derivation settings automatically to determine the blockchain and wallet addresses that have been used that are associated with the mnemonic seed.

Currently, the investigator is required to manually select a blockchain and derivation settings before searching. If there were no results, they would have to manually change the cryptocurrency blockchain/derivation settings until they retrieved a result. Implementing the 'Auto-Analyse' feature would eliminate the need for the investigator to do this and therefore save them a lot of time and manual interaction.

7.3 Feature Implementation Idea 3:

Another feature that could be implemented into the web application is the option to map out the transactions sent to and from wallet addresses that have been returned positive containing a balance from the application. This would enable the investigator to interact with the application to visually see where money has been sent to/from so the addresses can be added into a 'watchlist' for further inspection and monitoring if necessary. Currently, the application just returns the balances of wallet addresses and the total money in and the total money out. Knowing where this money has been sent from or being sent to allows for further wallets to be tracked which leads to the next feature idea.

7.4 Feature Implementation Idea 4:

Implementing a feature that can automatically track the activity of wallet addresses of interest associated with the mnemonic seed would enable investigators to monitor where the money is being sent to and from and get real-time alerts of when the wallet is being active / when it was last active in receiving and sending funds. Wallet addresses could be added to a 'watchlist' feature on the web application that will automatically track the balances within them.

7.5 Summary

The four new implementation ideas mentioned above would expand the web application with more advanced functionality as well as save the investigator a lot of time with the seed analysis and provide a greater level of depth and insight into the used wallet addresses and the money that is being sent to and from them.

8 Conclusion

The aim of developing the cryptocurrency wallet seed and blockchain analyser web application is to assist investigators with their analysis on a cryptocurrency wallet as well as save them as much time as possible. Currently, the Police are using a web application called Ian Coleman which allows them to enter a cryptocurrency mnemonic wallet seed and derive a long list of potential wallet addresses.

The application that I have developed further enhances the functionality of the Ian Coleman web application to quickly analyse wallet addresses on the blockchain and determine which of the derived potential wallet addresses contains a balance. This results in a huge amount of time saved as otherwise the investigator would be required to manually analyse each address individually to determine if there is a balance within it.

This web application takes in an input of a mnemonic wallet recovery seed which then derives a list of potential wallet addresses. These wallet addresses are then analysed on the appropriate blockchain and the balances, if any, are calculated, converted, and displayed to the user. Resulting in the user quickly knowing (in approx. 2 seconds) the exact wallet addresses that have been used and the balances inside them.

Although the initial plan was to use an xPubKey to analyse the blockchain and retrieve a set of known wallet addresses, changes were made to manually analyse individual potential wallet addresses. This change was made due to many blockchain APIs not allowing for an xPubKey to be passed through, but wallet addresses were allowed. Although this change was made and differs slightly from originally described during the design and specification section, the application still works as intended.

During the testing stage of the web application, I tested the functionality of analysing a randomly generated mnemonic seed and a real mnemonic seed of a cryptocurrency wallet that I had created. As expected, the application outputs 'No Results Found' when analysing the randomly generated mnemonic seed (which has associated wallet addresses with no balances). The analysis from the mnemonic seed of a real wallet address that I had deposited money into, resulted in the application correctly retrieving the correct wallet address with a correctly formatted wallet balance that was displayed out neatly on the UI for the user to read.

Whilst the application functionality has met the requirements I had initially set out to achieve, there is plenty of room for further improvement and for new features to be implemented which have been discussed within the 'Future Work' section of this report. The web application as it is, provides a functional proof of concept that enables a user to quickly determine wallet addresses and their total balances associated with the entered mnemonic seed. Although only two of the most popular blockchains have been implemented (Bitcoin and Ethereum), with room for more to be implemented, I believe this is sufficient to provide a functional proof of concept.

Overall, I am pleased with the developed web application despite some of the challenges faced during the development. It has been an enjoyable project to work on with new and existing skills being learned and utilised.

9 Reflection & Evaluation

9.1 Overview

This crypto-currency-based project has enabled me to develop a web application in an area of fairly new technology. The idea of this project has kept me motivated even whilst issues arose during development. The application that I have developed has required the use of a wide range of skills including, programming knowledge, time management, organisational skills, self-discipline, and learning how to manage a large project effectively.

9.2 Programming Skills

I developed the web application for this project using the JavaScript language. This enabled me to further develop my existing knowledge and skills in JavaScript, especially with using classes, working with API requests, and extracting important information from a set of returned JSON data. Having worked with JavaScript before several years ago, this project enabled me to refresh my existing skills as well as learn new ones.

9.3 Time Plan & Organisation

This project has helped me with my time management and organisational skills, as well as managing to progress when setbacks and unseen/unplanned issues arose. Managing to effectively measure the length of time for each project stage before development and then work to the devised time plan and achieve the targets was a little challenging at times, but overall, it was kept well apart from a few setbacks.

The time plan that I had devised before the start of the application development was initially followed very well and tasks/milestones were completed by the date targets I had set out. However, some issues arose during the web application development that required me to reprogram a large section of the code which resulted in the date targets set being pushed back further.

9.4 Challenges

Throughout the project, I had encountered some challenges that resulted in some functionality being delayed past its estimated completion date. One of the main issues that I had encountered was the implementation of searching the blockchain. The initial plan was to use a derived wallet public address key (the xPubKey) to search for known and used wallet addresses. However, when it came to the implementation, I had realised that this was only possible on the Blockchain.com API and not on the Ethplorer-API. Although pressured for time, this resulted in me having to restructure and recode a lot of the functionality so that it would instead loop through and analyse a list of individual derived wallet addresses. This issue could have been prevented and saved me a lot of time and stress later if I have undergone more in depth-research into blockchain APIs at the start rather than making assumptions that all APIs would offer the feature to search by an xPubKey, yPubKey, and zPubKey.

COVID-19 has been one of the major challenges throughout this project as it had affected my mental health which resulted in me feeling very unmotivated, unwilling, and stressed. To overcome this, I created a daily work routine and tried to ensure I completed one small task at a time, and I aimed for a minimum of 2 to 4 hours a day.

9.5 Supervisor meetings

It was pre-arranged during the first week of the project that supervisor meetings would be held every two weeks. (except over the Easter break). I have had a total of 5 supervisor meetings throughout the project. These meetings took place on the video calling platform, 'Microsoft Teams'. These supervisor meetings enabled me to share the progress I have been making on the project and ask any further questions that I may have. Emails were also sent to/from the supervisor to receive feedback on report writing as well as answer any general queries that I had.

I found my supervisor to be extremely helpful throughout the project and has provided strong guidance and ideas that were used to further enhance my report and project overall. The support and feedback on the work I had completed had encouraged me to keep going when the project got tough.

9.6 Future Projects

In future projects, I will devise a time plan that allows for slightly more flexibility so that if issues were to occur it would not impact the deadlines and targets set for each milestone, which consequently has a knock-on effect otherwise. I will also not make assumptions and ensure that I investigate every little detail where required.

10 Glossary

BIP – BIP stands for Bitcoin Improvement Protocol. These protocols are used to propose improvements to the Bitcoin Protocol which can include new features, security, and information.

BIP 84 – BIP84 is used to derive native SegWit addresses. These addresses start with "bc1". *(Wallet 2021)*

Bip32 – BIP32 defines the rules used to derive wallet addresses.

BIP44 – BIP 44 is the most common BIP for deriving addresses that are non-SegWit. These types of addresses begin with the number 1. *(Wallet 2021)*

BIP49 – BIP49 is used to derive SegWit compatible addresses. These addresses begin with the number 3. *(Wallet 2021)*

Blockchain – A blockchain is a distributed peer-to-peer network of cryptographically secured blocks that hold information and transactions that are sent to the blockchain by those interacting with it.

Mnemonic Seed – A mnemonic seed is a list of words that act as a private key to recover a wallet. These usually consist of a string of 12 or 24 words. Knowing the mnemonic seed to the wallet enables access to the funds stored inside.

SegWit – "SegWit is the process by which the block size limit on a blockchain is increased by removing signature data from bitcoin transactions. When certain parts of a transaction are removed, this frees up space or capacity to add more transactions to the chain." *(Frankenfield 2021)*

xPubKey – An xPubKey is an account extended public key. These keys provide read-only access into wallets and balances held by a person. Created through derivation using BIP44.

yPubKey – An yPubKey is an account extended public key. These keys provide read-only access into wallets and balances held by a person. Created through derivation using BIP49.

zPubKey - An zPubKey is an account extended public key. These keys provide read-only access into wallets and balances held by a person. Created through derivation using BIP84.

11 References:

1. Assets ranked by Market Cap - CompaniesMarketCap.com. 2021. Available at: https://companiesmarketcap.com/assets-by-market-cap/ [Accessed: 18 February 2021].
2. BIP39 - Mnemonic Code. 2021. Available at: https://iancoleman.io/bip39/ [Accessed: 4 March 2021].
3. Blockchain Tutorial: Learn Blockchain Technology (Examples). 2021. Available at: https://www.guru99.com/blockchain-tutorial.html [Accessed: 24 February 2021].
4. Blockchain.com Explorer | BTC | ETH | BCH. 2021. Available at: https://www.blockchain.com/api [Accessed: 24 February 2021].
5. Cryptocurrency Prices, Charts, And Market Capitalizations | CoinMarketCap. 2021. Available at: https://coinmarketcap.com/ [Accessed: 18 February 2021].
6. Eng-Tuck Cheah, J. 2021. What is Defi and why is it the hottest ticket in cryptocurrencies?. Available at: https://theconversation.com/what-is-defi-and-why-is-it-the-hottest-ticket-in-cryptocurrencies-144883 [Accessed: 23 February 2021].
7. EverexIO/Ethplorer. 2021. Available at: https://github.com/EverexIO/Ethplorer/wiki/Ethplorer-API [Accessed: 6 May 2021].
8. Frankenfield, J. 2021. SegWit (Segregated Witness). Available at: https://www.investopedia.com/terms/s/segwit-segregated-witness.asp [Accessed: 21 May 2021].
9. Hooper, M. 2021. Top five blockchain benefits transforming your industry - Blockchain Pulse: IBM Blockchain Blog. Available at: https://www.ibm.com/blogs/blockchain/2018/02/top-five-blockchain-benefits-transforming-your-industry/ [Accessed: 23 February 2021].
10. https://www.guru99.com/blockchain-tutorial.html Blocks Image
11. iancoleman/bip39. 2021. Available at: https://github.com/iancoleman/bip39 [Accessed: 5 March 2021].
12. Project Allocation & Tracking System (PATS). 2021. Available at: https://pats.cs.cf.ac.uk/ [Accessed: 9 February 2021].
13. SETH, S. 2021. Six Private Cryptocurrencies. Available at: https://www.investopedia.com/tech/five-most-private-cryptocurrencies/ [Accessed: 22 February 2021].
14. The MIT License | Open Source Initiative. 2021. Available at: https://opensource.org/licenses/MIT [Accessed: 5 March 2021].
15. Wallet, S. 2021. BIP 44, BIP 49, and BIP84. Available at: https://support.samourai.io/article/65-bip-44-bip-49-and-bip84 [Accessed: 21 May 2021].