Development of a Functional Blockchain

Kacper Grzenda

B.Sc.(Hons) in Software Development

April 10, 2022

Final Year Project

Project Repository: https://github.com/kacpergrzenda/Java-Blockchain-Applied-Project

Advised by: Martin Hynes

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

About this project

Project Repository: https://github.com/kacpergrzenda/Java-Blockchain-Applied-Project

Abstract The work presented in this thesis describes the design and development of a robust functional coin-oriented blockchain application accessible through the browser. The main objective is to investigate the current state of blockchain technology and create a robust financial system that operates using cryptocurrency while following the main protocols of a blockchain system.

Blockchain is payment infrastructures in a digital reality which allow for building, making, and selling without any sort of physical assets. This new blockchain economy is thriving and being utilised in the Software industry due to its ability to break past barriers that have long held back their ability from sharing data and changing the way the financial system operates. Blockchains are known for holding data in blocks which are immutable once created making them almost impossible to hack and change. The design of this project is a developed blockchain from scratch that acts as a digital currency or also known as cryptocurrency. The Blockchain holds immutable blocks that contain block information and hold blockchain transactions which can be sent from user to user on the blockchain network without a third party. The Blockchains digital currency works just as currency just like the Euro or Dollar but in this blockchain the currency is a cryptocurrency. Transactions between blockchain participants are able to be created. Transaction's information is stored in blocks that are then mined, signed with a digital signature, and added to the blockchain. The Blockchain follows a strict consensus and validation protocol, making it tamper proof.

The Blockchain is accessible through the browser by users. The front-end visually represents the blockchain functionality and is designed to provide users with good user-experience with a simple learning curve. The front-end allows participants to mine new blocks, send transactions and view blocks that are on the blockchain. The blockchain and front-end are connected by

Control Container that acts as a bridge, that handles requests to and from the front-end and blockchain.

This project demonstrates a clear example of a robust blockchain that acts as a financial system and provides the ability of mining immutable blocks following a Proof of Work algorithm while being secure and tamper proof from outside sources by containing multiple validations algorithm to guarantee immutability. While also allowing the user to visually experience all the blockchain functionality through their very own browser on their computer.

Authors The Author of the project is Kacper Grzenda. Software Developer student at Galway-Mayo Institute of Technology.

Chapter 1

Introduction

Blockchain are tamper evident and tamper resistant digital ledgers implemented in a distributed fashion and usually without a central authority like a government. Bitcoin was the first blockchain, developed in 2008 by Satoshi Nakamoto. Since bitcoin there exist multiple blockchains such as Ethereum, Ripple and Solana, all with their own unique features and functionality. Blockchains are constantly being adapted by multiple industries worldwide for different uses such as security, currency, proof-of existence, and the tokenization of assets. All these different uses can be done in various ways as there is increase of blockchains circulating which allows easy adaptation of blockchains as industries don't need to wait in line to get onto a network it can be done in seconds.

The most know use for a blockchain is coin-oriented blockchains mainly knows as cryptocurrency intended for digital transactions secured by cryptography to anywhere in the world without a central authority controlling the system. [1] [2] Cryptocurrency enables for transparent secure payment this is due to various encryption algorithms and cryptographic techniques that eliminate the trust factor of a third party like a bank making it easier and faster to transfer funds directly between two parties. There is also blockchains that focus on creating smart contracts the most knowing blockchain for this is Ethereum, which allows to create decentralized applications that interact with contracts and offer features and services to users. [3] Smart contracts are programs stored on a blockchain and run when fixed conditions are met typically used to automate execution of agreements, then storing the data on the blockchain that hosts it.[4]. With the different uses this project is focuses on the development of a cryptocurrency blockchain allowing it to operate as a cryptocurrency.

However, no matter what use the blockchains has it needs to follow certain protocols to work and allow trust to the participants. These protocols are: 1) Consensus algorithm, to achieve agreement on data value among distributed systems. The two most used consensus algorithm are proof of work and proof of stake. 2) Effective Cryptography is a most to providing a secure environment for the participants. This ranges from wallets, keys, and digital signatures. A commonly used algorithm is the SHA-256 which handles the encryption of data. 3) Immutability: This is one of the key features of blockchain technology. The blockchain consists of blocks that are linked to each other by their hashes, this allows for a validation system where every block validates its previous block to make a change making it almost impossible to tamper with. 4) Decentralization: Most Blockchains are decentralized meaning they are not controlled by a single person controlling the system. Rather distributing the system onto many machines which work together also known as nodes, making blockchains decentralized. This makes blockchains transparent as it creates a transparent profile of every participant, with every transaction visible to anyone making it corruption proof.

1.1 Project Context

This Project is an example of a fully implemented functional Blockchain coded using Java programming language the blockchain is developed following the main blockchain protocols. The Blockchain in this case resembles financial system that uses cryptocurrency and allows a user to visually see how the blockchain operates through a front-end giving the user the ability to be a participant of the blockchain accessing all its functionality. The currency in this project works like the Euro or Dollar but in this case it is a digital currency, the cryptocurrency allows participants on the blockchain to make payments between them without going through a bank, instead the blockchain is a system of decentralised trust less verification based on cryptography. The Blockchain stores the payment data under a transaction in blocks which are all chained in a chronological order through cryptography. Each Block on the chain, contains transactions that follow a Merkle tree data structure to encode transaction data more efficiently and securely, strengthening the blockchain integrity. Participants have control of their wallets where their cryptocurrency is tracked, and the wallet allows a participant to send and receive the coin. Wallets are controlled by private and public keys which allow for the transactions between users, the private key must be kept safe and secure as its the authentication for the wallet. The wallets keys work under an elliptical curve digital signature algorithm that is used in the signing and verification process of the digital signature. The Blockchain acts as a ledger where in each block transactions are recorded keeping account of the payments made between the participants. The Blockchain operates under a strict decentralized consensus mechanism called the ”Proof of work” algorithm, this algorithm validates the blockchain transactions and broadcasts new blocks to the blockchain, first introduced by Bitcoin [5].The SHA-256 hash algorithm is used to generate hash numbers, it moderates the creation and management of addresses, wallet key pairs and is also used for transaction verification, this algorithm is used for the encrypting of data in the blockchain into a secure format.

Additionally, the Blockchain is accessible through a front-end. The Angular framework is used for the front-end functionally and building the application using HTML, CSS and Typescript. Additional Angular libraries are used to design the front-end this allows for a good user-experience using the front-end. The front-end simplifies the complexity of the Blockchain into an easy-to-understand process, where each page displays different functional features of the blockchain that a user can access. A user is able to generate their wallet, create transactions, mine blocks and see new blocks that are added to the blockchain on the front-end, this will allow for new users to visually learn and have a better understanding of how the blockchain main concepts work. The blockchain will be displayed on the main page showing the information of each block and each blocks transactions, allowing users to track their transactions and other transaction making the blockchain transparent to everyone.

The data transferred between the front-end and blockchain is through Spring-Boot a service framework that will handle requests between the user and the blockchain. When a user makes a request, the front-end this will send a request to the service and notify the blockchain to process this, which then the service gives feedback to the user with a change or message notifying them. The service objective is to handle validation, informing the user of incorrect requests when dealing with the wallet and creating transaction. Spring-Boot will allow for the front-end to run on its own while using the blockchain functionality. The service will be connected to a MySQL database this is for backing up the blockchain blocks, these blocks are identical to the original blockchain blocks in the blockchain. The blocks being saved to the MySQL database will be saved after they have been mined and passed validation checks for verification purposes. The need for the Sprint-Boot service is creating a bridge between the back end and front-end allowing both to function without incorrect requests being received.

Heroku a cloud platform service is used to deploy the Angular front-end and Spring-Boot service making the blockchain available to end users. The front-end of the blockchain is accessible through the browser under this URL: ”https://spring-angular-blockchain.herokuapp.com/” and the spring-boot service which handles the requests for the front-end and the blockchain is under this URL: ”https://spring-blockchain-server.herokuapp.com/”. The Spring-serve is deployed securely as it stores the blockchain data, and changes made to the repository of the spring-server will automatically update and deploy the new version. The deployed front-end is connected and receives requests from the spring deployed server allowing both to communicate and word constantly allowing full time access by the user.

1.2 Objectives

All objectives are designed and thought through to achieve the completion of a robust functional blockchain. Project objectives are set out under their own sections, with some sections having connecting objectives. Essentially meaning if an objective is split into two parts in two separate sections for the full objective to be complete both parts need to be complete. Objectives are clearly set out and showcase the metrics by which success and failure is measured.

1.2.1 Blockchain Objectives

Blockchains are known for their security because of their cryptography, decentralization, and consensus, this is what keeps the trust of the blockchain and its blocks and this is the main goal of this project. The blockchains objective in this project is to ensure that the blockchain follows security and validation principles and verifying that the blockchain is tamper-proof by-passing chain validation test guaranteeing its immutability and also passing validation checks in other areas. The Proof of work consensus algorithm must pass validation and produce blocks onto the blockchain. Each block must pass a validation, the validation checks for immutability, if the block is unique and is mined under the correct process and that the block is part of the chain in the correct position. Each block most also have a unique Merkle root id, this is achieved when transaction are passed through a Merkle root algorithm. Transactions that are part of the block are verified using evidence from previous transaction input and output and also by checking transaction signature, validating that transaction if it passes the process. The transaction also is checked for correct signature, this is checked by an algorithm to verify that the transaction is legit and signed by a correct private key once verified true, check is complete. Once the blockchain validation in all parts of the blockchain are complete and passed correctly the object is complete.

1.2.2 Front-End Objectives

The front-end objective is to be reactive and responsive allowing for quick navigation and better user experience while using the front-end. When using the Front-end the user should understand how the blockchain functionality is connected together this should be achieved by creating a simple layout to understand the blockchain. Wallet accessing and generation should be easy to access and should display the wallet information, wallet information should be the users private key, public key, and the wallet balance. The Wallet information is to be stored in local storage when the user returns to the site, they can quickly access their information. Mining page should display the current transaction that is ready to be mined with the information of the transaction displayed. Transaction page allows the user to send funds to another user's wallet, the transaction page is in the form of a form that the user needs to fill out for the transaction to be completed. All other necessary blockchain functionality should be also correctly display on their significant pages. The front-end is also to be deployed on the Heroku cloud platform to allow the blockchain to be accessible by any user from their browser. Once this is complete the front-end objective is complete.

1.2.3 Server Objectives

The objective of the server is complete when the front-end and blockchain can communicate together through a local environment or through a production build environment on a deployed site. The server can handle requests from both sides and in situations where the request is incorrect the user should be notified with a validation message. The server should be able to access the blockchain functionality and call methods that are public to the server. Blocks from the blockchain should be handled by the server and saved into a MySQL database for backup purposes, these blocks should have the exact information as the blocks in the original blockchain which have already been verified. These objectives must be complete for the server to be complete.

1.2.4 Deployment Objectives

The Blockchain deployment on Heroku should work exactly the same as locally featuring all the same functionality. The spring-boot server deployed on Heroku should respond to all requests from the server controller, these requests should be tested using postman. The deployed front-end should display each page exactly as on the local server. Both deployments should be built successfully this should be tracked and followed using Heroku logs.

The MySQL database has to be switched to the database that Heroku supports and needs to be able to save the blocks in the format that they are saved locally with the correct data. The deployed nodes need to be able to communicate and respond to each other once this is achieved the deployment object is complete.

1.3 Repository Main Elements

The project repository (https://github.com/kacpergrzenda/Java-Blockchain-Applied-Project) contains a lot of code and functionality for the project to work, this section gives a good understanding of the elements in the repository and what they are used for and the features they contain.

The repository to this project contains the front-end, back-end and the Sprint-Boot service. The blockchain functionality is under the ”blockchain” folder, it is also exported as a Java jar file ”blockchainExtension.jar” for the service to be able to execute the blockchain methods. The ”blockchain” folder holds the main components of the blockchain, these include the cryptography, consensus algorithm, validation to all components, Merkle root algorithm, wallet.class, transaction.class, block.class. This is the main folder of the entire project that holds the main functionality and make the entire blockchain functionality work. The java jar file ”bcprov-jdk15on-170.jar” is a library [6] containing the Elliptical Curve Digital Signature Algorithm (ECDSA) used for the key creation and verification process. The front-end code is the ”blockchainapp” folder, the main files in this folder are the four pages which visually represent the blockchain features and functionality and give the user access to navigate through. The code for the deployment of the front-end is in the ”server.js” allowing the front-end to be deployed on Heroku and connecting to the server. The technology in this folder is Angular, Typescript and node.js which are all used for the front-end to function. The Spring-boot service folder is ”blockchainmanager” this folder contains the service file which handles the requests from the front-end and blockchain creating a bridge between the two. The folder contains the requests for the front-end and back-end to make calls to each other. The MySQL database is a dependency of this folder and is connected through the server saving the blockchain contents.

1.4 Description

This section gives a brief description of each chapter and its contents, summarising what can be expected in this thesis.

1.4.1 Methodology

Methodology chapter gives an overview of the approach that was taking to research and develop the project. Details of the research methodology and development are listed and explained, giving a descriptive and concise understanding of the type of research done. The planning of the project, explanation of agile methodology Scrum, approach for testing and validation the blockchain, development tools and empirical approach is all explained in this chapter.

1.4.2 Technology Review

This chapter conducts a literature review on the technology used to complete the blockchain project. The technologies used are described and explained on a conceptual level. The project is split into three sections Front-End, Server and blockchain, the technologies that were used for each section are explained.

1.4.3 System Design

The system design chapter focuses on the architecture of the blockchain how it is designed to be robust, the front-end design and the back-end Spring-Boot design. Diagrams are used to showcase the design of the project.

1.4.4 System Evaluation

This chapter evaluates the project against the objectives set out in the introduction. The design and architecture of the project is discussed and proved to be robust. The testing and validating that was done during the Agile cycle is also discussed in this chapter.

1.4.5 Conclusion

A brief summary of the context and objectives that were first set in the beginning. The outcomes of the project, insights that were made during the development and process of the project and opportunities identified for future investigation are discusses in this chapter.

Chapter 2

Methodology

In this chapter the approach to the development of the blockchain is explained, describing the stages of the research process and the development process and how each stage was conducted making sure the approach to the problem was valid. Details of project planning are individually explained making sure the requirements for the project are confirmed and follow the main protocols needed for the completion of the blockchain to its full standards. The requirements are designed to ensure the full efficiency later on in the development cycle to produce a valid result that addresses the research aims and display the completion of all objectives. The agile methodology approach is addressed, explaining the steps that were took to produce a successful project following all correct steps that need to be taken in an agile development cycle, these steps consist of meetings, structure, checks, summarising development, and feedback. Validation and testing are an important research section in this chapter as it is one of the main protocols that needs to be followed, the research was analysed to a great extend to assure not gap was left in the validation that could be exploited. Research conducted before development was on a theory level and empirical research, analysing previous projects and existing blockchains, gaining knowledge of what is needed to produce the most appropriate blockchain for the type of blockchain project that this blockchain is going to be used for, cryptocurrency. All the details mentioned in this chapter are connected to the previous chapter of the context of the project explaining the approach that is followed.

2.1 Approach to Development

Blockchains take a long time to develop and need to be perfectly designed with the functioning complex algorithms working correctly, as one small problem could cause the blockchain to no longer being secure. Research was undertaking first, as time management and security were the most important factors to keep in mind the most fitting cryptography and consensus algorithms were being analysed and thought through deeply to make sure the functionality is perfect with no holes in the project while also keeping the scope of the project in mind. Then the technology that was going to be used was researched and the best software for this type of project was chosen to benefit workflow and also using something that is popular in the blockchain software industry and is known to work.

Once the software research was complete a research methodology was chosen that would successfully help the idea come to life. This is why an agile approach was taking to design and develop the project, following a scrum agile methodology. The agile approach was chosen as it is considered a successful project management plan in the software industry with a good percentage of projects succeeding following the agile cycle. The Agile approach allowed for focusing on coding right away which allowed the blockchain validation functionality to be robust, also as the blockchain is a complex design if future problems came up responding to change would be easier to do while keeping on track with the deadline and prioritising the security of the blockchain. The use of Scrum made sure the blockchain was being validation and tested frequently making sure the design is tamper proof, which was the main goal of the plan. Scrum would also help keeping on top of the blockchain protocols and making sure the requirements for the blockchain to function were met. GitHub was decided as the code hosting platform, it is a great collaboration tool and allows the supervisor to keep track of the project. GitHub Project tool was also decided on to keep track of the sprints and keep everything in order, it is also great for the agile methodology approach.

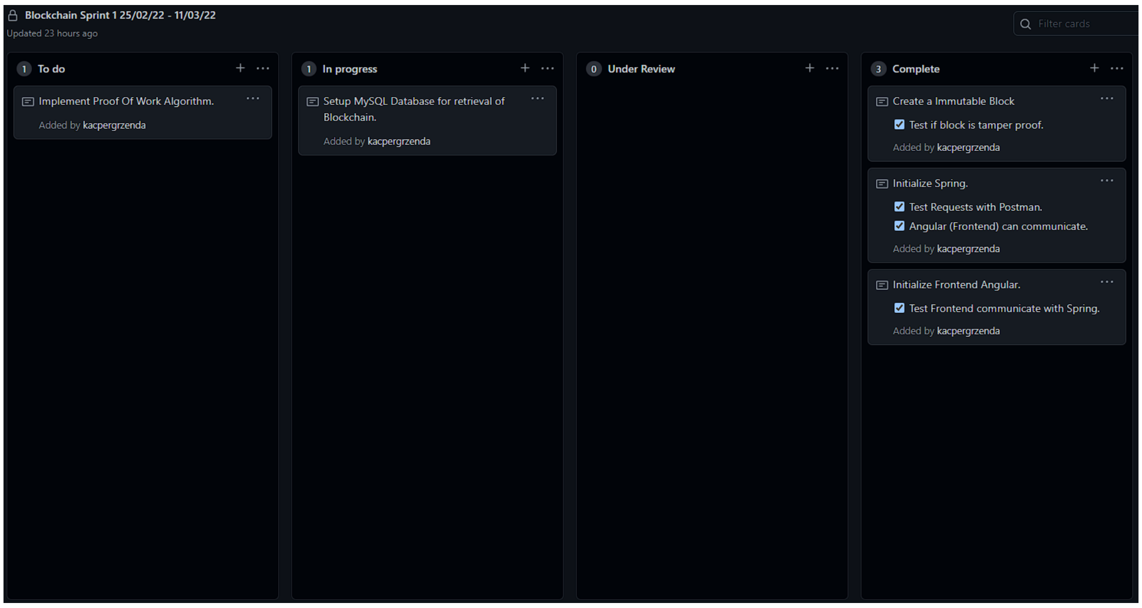
2.2 Planning

The project idea is a complex structure so the architecture of the blockchain is designed and thought through thoroughly making sure each part of the blockchain will work together and ticking of the required functionality needed to pass validation and make blockchain work. The main important parts of the blockchain are split into sections, these sections are analysed making sure no changes need to be made. Each sections functionality is transformed into user stories. The user stories are made small, addressing the who, what, why pieces of information and making sure the main blockchain protocols are being followed and complete. The advantage of keeping the user stories small helped with errors and bug being addressed early which was crucial for the security of the blockchain.

Stage two of planning is the first sprint creation and organising how the process will go. The most important user stories were selected and turned into tasks that are broke down even more into specific functionality that needed to be dealt with early in-case of errors, an example of these were block plan, mining, and proof of work algorithm. These tasks created were then analysed and a plan was created how to complete them. Each sprint is planned to be max duration of two weeks, during these two weeks as much of the tasks as can be completed, if not completed the uncompleted tasks are moved into the second sprint. A stand-up meeting is held once a week using Microsoft teams between the developer and the project supervisor, discussing what has been complete, problems faced and changes that might need to be made. This allows to keep track of project and sticking to the plan of the sprint.

Stage three is the start of the developing process in this stage, a Kanban board was created using GitHub Project as showcase in Figure 2.1 The Kanban board displays details of the sprint such as start-date and end-date and the chosen tasks are placed on the Kanban board that are planned to be completed. The Kanban board had the following columns ”To Do”, ”In Progress”, ”Under Review”, ”Complete”, this helped organize the tasks and keep track of the stage a task was at in the development process. The chosen tasks that are most important are then transferred onto the GitHub Kanban board and placed onto the ”To do” section. These tasks are the tasks that will be worked on in this two-week period. The Kanban board also helped prioritise the main tasks and allowed for flexibility of how the tasks were handled and created this helped simplify the sprint process by easily keeping track of everything.

In stage four the tasks that are ”Under Review” section on the Kanban board are tested and need to pass validation. This is the most important part as the blockchain needs to pass all security measures as its acts as a financial system. Specific validation tests are created for each completed task, once a task passes validation and system works correctly it is marked a fully completed and moved to the ”complete” section of the Kanban board. Postman was used for API testing this made it easy to test requests from the front-end and blockchain, this made testing more efficient as tests were less tedious work, allowing to keep on track of time for the sprint. Selenium was used to test the front-end, this was done by running selenium in the browser and testing navigation and important buttons that output data to the screen, also visual tests were done to test the layout and design of the front-end. These visualisation tests were compared to the designed wireframes and once agreed on satisfaction they were marked complete.



Blockchain Sprint 1 25/02/22 - 11/03/22
Updated 23 hours ago

1 To do
- Implement Proof Of Work Algorithm.
  Added by kacpergrzenda

1 In progress
- Setup MySQL Database for retrieval of Blockchain.
  Added by kacpergrzenda

0 Under Review

3 Complete
- Create a Immutable Block
  - ✓ Test if block is tamper proof.
  Added by kacpergrzenda
- Initialize Spring.
  - ✓ Test Requests with Postman.
  - ✓ Angular (Frontend) can communicate.
  Added by kacpergrzenda
- Initialize Frontend Angular.
  - ✓ Test Frontend communicate with Spring.
  Added by kacpergrzenda

Figure 2.1: Example of Kanban board used for sprint1.

After the sprint is finished one week planning stage occurs after each sprint discussing what went well and what can be improved in the next sprint. The incomplete tasks from the previous sprint are moved into the current sprint to be complete. The knowledge of the previous sprint helps avoid the problems faced in the current sprints and everything is discussed with the project supervisor. New tasks are added to the Kanban board to replace the complete tasks, also making sure that the two-week schedule suits all objectives.

Chapter 3

Technology Review

In this chapter, literature review has been conducted on the technology used to build the fair and open financial system blockchain, while also reviewing the technologies used for developing the front-end of the blockchain, back-end service and cloud platform for deployment. The details of the technologies used have been provided on conceptual level. The blockchain system is an advanced financial network that consists of multiple different functionality that have tremendous advantages, behind these functionalities consist various different technologies that allows the blockchain to work consistently without any disruptions as one minor problem could negatively affect the entire blockchain. When it comes to choosing the technologies to develop a functional blockchain it comes down to the most researched, commonly used, technologies that best support the functionality, secure technologies that allow the blockchain to work consistently to its full potential without and flows in the technology being used and most importantly choosing the right technologies based on your own specific requirements. This is especially true with the number of different platforms and technologies available, making an informed choice is key to a successful blockchain.[7] The objectives of the blockchain are tightly considered when choosing the right technology, it comes down to which technology is most suitable for the objectives to be complete and will it allow for the progression of the blockchain. The correct blockchain protocols are followed along with the requirements ticking of both boxes of the process. As the blockchain is based on a financial system and also deployed as a front-end to visually represent the functionality of the blockchain each tier of the project is analysed and split into sections to prioritise efficiency during research. The main sections of this chapter are Programming language, Consensus Algorithm, Cryptography, Immutability, Front-end technology, Back-end Technology

There are many uses for blockchains and also different ways they can function, with there currently being 1500 cryptocurrencies that are classified under different categories.[1] As the most important requirement of this project is a blockchain that is coin-oriented, research is made under this specific category. Research of pre-existing blockchains under this category, blockchains that have succeeded, are still widely used and blockchains that follow strict validation protocols with no breaches. Cryptocurrencies that are similar to this project's idea are analysed one of these cryptocurrencies is Bitcoin, the first blockchain which is well known around the world. It serves as money for large transactions because of its large transaction cost and investment due to its massive increase of value, it also cannot be used for anything other than means of payment.[5] [1]

This functionality that makes the blockchain popular, immutability, distributed control, enhanced security, and decentralization, are also the sources for the blockchains greatest weaknesses if the technology backing the blockchain is not correctly researched and developed.[8] These blockchain technology advantages are looked at by many as a great new tool to add to the economy sector, number one being cryptocurrencies decentralized system. However, these advantages are scoped by hackers that are eagerly looking to find gaps and holes in the technology to exploit the blockchain. Blockchain breaches are unfortunately a common resource as between 2011 and 2018, 72 public breaches were reported ranging in losses from \$12,000 to \$600 million and in total the reported losses of cyber-attacks exceed \$1 billion. [9] It's no surprise why blockchains would be targeted which these massive amounts of money being stole and blockchain being a new technology that is still be learnt and adapted into the financial market leaves many vulnerabilities that are overlooked by the developers and designers of the blockchains.

Each section of this chapter is deeply research under its specific technology making sure of it use and how it will impact the blockchain. This technology research work also aims to indicate the prospective use of the blockchain as a cryptocurrency, to demonstrate the challenges and possible different ways blockchains can operate.

3.0.1 Java Blockchain Programming Language

Almost all popular programming languages are used in the blockchain industry, with new programming languages behind developed to make blockchain development more efficient and make the programming language more adaptable for the specific blockchain. One of these being Solidity, which is a contract-oriented Turing complete language for the Ethereum blockchain. [10] [11] However when it comes to picking a programming language, the development needs to be considered for the type of blockchain that is being developed.

The chosen programming language for the blockchain being developed for this project, is Java. Java is object oriented, class-based and concurrent. It is designed in such a way that it has few implementation dependencies, it can also easily run on any computer that has the Java Runtime Environment (JRE) installed on it. [12] [13] Java was originally developed by James Gosling at Sun Microsystems and released in May 1995. Since its launch in 1995, Java has become one of the top 3 programming languages and rightly so with over 9 million developers. [11] The language is derived from the C-syntax and in blockchain development, Java is used for building interactive DApps because of its ease of memory cleaning and availability of ample libraries. [14] Java programming language is good to create simple and immutable blockchains, which this project is exactly based on. With this immutability, it is impossible for blockchain to be tampered with and change the data in the blocks. Java is preferred amongst the blockchain community because of its high portability, this allows all programs that are written in Java to be portable across all computational systems as they don't rely on system-specific architecture by using the Java Virtual Machine for execution. [12]

Java programming language is a high performance, as bytecode is the compiled format for java programs. While the performance of interpreted bytecodes is usually more than adequate, there are situations where higher performance is required. Bytecodes can be translated at runtime to machine code for the particular CPU the program is running on. Allowing Java program performances to be indistinguishable to other programming languages. java provides powerful addition to the tools that blockchain developers have at their disposal. java makes blockchain development easier because it is object-oriented and had the automatic garbage collection. In addition, because compiled Java code is architecture-neutral, Java applications are ideal for a diverse environment like the Internet. [13]

3.0.2 Angular Front-End Framework

The frontend of a software program is everything with which the user interacts. The frontend from a user's use is the User interface, which allows the user to control the software application, a good user interface provides a user-friendly experience. [15] From a developers standpoint, it is the interface design and the programming that makes the interface function. One of the primary goals of frontend development is to create a smooth, responsive, reactive use experience. [16] The frontend of an application should be intuitive and easy to use especially when it comes to blockchains where your dealing

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Figure 3.1: Angular component

with a financial system with multiple functionality, the learning curve for the user should be quick and simple. While this sounds like a simple goal, it can be surprisingly complex since not all users are the same.

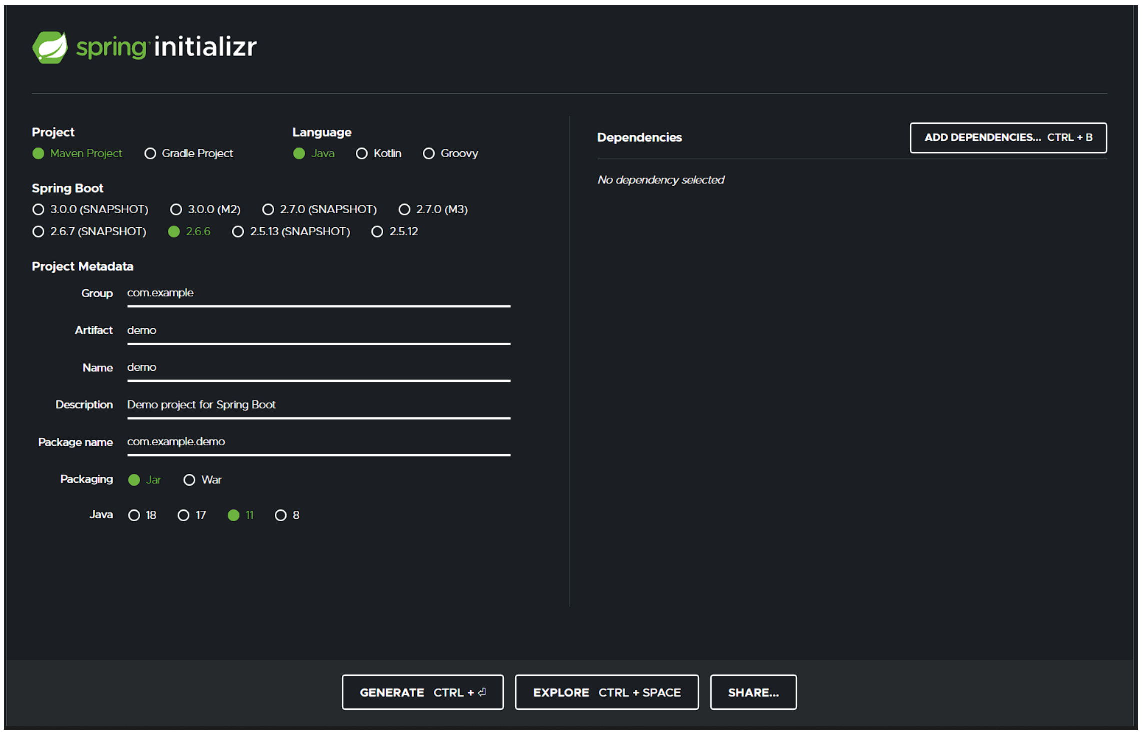
However there is frameworks that can help with the development of the frontend, Angular is a platform and framework for building single-page client applications using HTML and Typescript. Angular is written in TypeScript. It implements core and optional functionality as a set of Typescript libraries that you import into your applications. [17]. Angular is built on Typescript, Typescript is a primary language for Angular application development. It is a superset of JavaScript with design-time support for type safety and tooling. Browsers can't execute Typescript directly. Typescript must be "transpiled" into JavaScript using the tsc compiler, which requires some configuration. [18] Typescript provides an error handling strategy based on try/catch syntax, this helps eliminate unnecessary errors leading to security breaches and provides an efficient way of building the front-end, which is a primary objective of the blockchain.[19]

The architecture of Angular applications relies on certain fundamental concepts. The main building blocks of the Angular framework are Angular Components that are organised into modules. The components define views that compose and application, which are pages that combine HTML and styles with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display. [17] Figure 3.1 is an example of a minimal Angular component:

Angular Material is an implementation of Google's Material Design Specification (2014-2017). [20] Angular material provides a set of reusable, well tested, and accessible UI Components for building frontend applications. Angular material provides for development of high quality, versatile and frictionless UI components. This allows for designing and creating pages with simple user-experience and easy learning curves for users, by using angular material components the front-end of the blockchain can be designed for users to have a good understanding of the blockchain functionality.

3.1 Sprint Boot Back-End Service

The Blockchain and the front-end need a bridge between them to communicate, this service is needed to allow requests to be sent between both ends. The service allows the user to be a participant of the blockchain by being able to control all blockchain functionality that is available to them. Spring Boot is an open source, microservice-based Java web framework, that allows for the bridge to be connected between the back-end and front-end.[21] Spring is a framework that enables a big program to be decoupled, it makes switching components or implementations easier by using dependency-injection, also using Apache Maven along with spring manage the project.[22] The Spring platforms already contains various components to get the project initialised, especially helpful for web applications, offering web-application dependencies like the MySQL and PostgreSQL dependency as used in the project. Using the Spring Initializer allows for a quick way to choose the needed dependencies, programming language, maven project and version of the programming language.[23] Initially Spring does not generate code automatically and doesn't use XML configuration file, Spring uses internally pragmatically configuration done by spring boot developer that are provided by jar, this what the Spring Initializer does sets up the needed file and structures the project[24]. The Figure 3.2 showcases the Spring Initializer used for he build.


spring initializr

Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M2) 2.7.0 (SNAPSHOT) 2.7.0 (M3)
2.6.7 (SNAPSHOT) [✓ 2.6.6] 2.5.13 (SNAPSHOT) 2.5.12

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	18 17 [✓ 11] 8

Dependencies
No dependency selected

Figure 3.2: Sprint Initializr Used to Setup Sprint Service.

3.1.1 Maven

Maven is used for java-based projects, helping to download dependencies which refers to the libraries or JAR files and to automate the build of the Spring Boot application. Maven helps to get the right JAR files for each project. While using Maven doesn't eliminate the need to know about the underlying mechanisms. Maven is based on the Project Object Model, this allows projects to be maintainable, reusable and display a simple model for projects. Mavens project object model is an XML file that has all the information regarding project details. [25] The POM keeps the details of the Spring Boot service and the keeps the dependencies needed allowing for the development to be more efficient. Blockchain JAR file is a dependency in the Maven Spring project allowing all the necessary methods be used in the Spring surface. Maven simplifies the process of building the project, allowing the deployment of the Spring-Boot service on Heroku to be simple as maven manages the building of the environment.

3.1.2 MySQL

MySQL is an open-source relational database management system. A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. An item in a row can be marked as a unique identifier using foreign keys, the item can then be identified as a relation to a different item in a different table. [26] A MySQL database is a structural collection of data. This database can hold any type of data from small to big data of different types such as Integer, Float, Byte (etc.), because of this the database can store each block from the blockchain. The MySQL database allows the blocks to be backed up. The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces. [27] MySQL is the perfect database to use for this project as it fits all the requirements needed for this project and also it being fast, reliable and scalable is a bonus. In MySQL you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

3.2 Postman

Postman is a software used for API request testing. It is an HTTP client that tests requests (POST, GET, PUT DELETE, etc.). When testing APIs with Postman, it responds with different response codes, these code help with identifying what the status of the request is. The main codes are 200 series where client accepts request and it is processed successfully ”200 OK”, 400 series client error response ”400 Bad Request”, 500 Series server error response ”500 internal Server Error”. Postman gives the possibility to group different requests. This feature is known as ‘collections’ and helps organize tests. These collections are folders where requests are stored and can be structured in whichever way the team prefers. [28]

3.3 Heroku Cloud Platform

A cloud platform is an operating system and hardware of a server in an internet-based data center, this allows software and hardware products to co-exist remotely at a large scale. Cloud platforms can be used for software storage, databases, networking, analytics etc. Cloud computing services work for different type of platforms, infrastructure as a service (IaaS), platform as a service (PaaS), serverless, and software as a service (SaaS). [29] These are sometimes called the cloud computing ”stack” because they build on top of one another, and are used for different reasons, of what the application will be used for and how it’ll work. Heroku is a container-based cloud platform as a service supporting several programming languages. Heroku allows for the deployment and management of large-scale modern applications. It is a flexible and easy to use platform that always for efficient deployment of applications. [30] Heroku fully manages deployed applications, giving developers the freedom to operate on the core of their applications. The Heroku platform uses Git as the means for deploying applications, when a new application is created on Heroku, a new Git remote is associated with the application. Once the application is pushed to Heroku, it receives the application source and initialises the build of the source of the application. Heroku identifies the programming language. In the case of a Java application may fetch binary library dependencies using Maven, compile the source code together with those libraries, and produce a JAR file to execute. Heroku provides add-ons like databases, queuing and catching systems. Add-ons are provided as services by Heroku, Heroku treats these add-ons as attached resources and links them to the applications. [31] Heroku monitors and keeps logs of deployed application, keeping time stamps events and collates the streams of logs produces from all the processes. Heroku PostgreSQL add-on runs on the cloud for over 12 years, securing the data with compliance with key industry standards for data protection. [32]

3.3.1 Heroku PostgreSQL

PostgreSQL is an open-source relational database management system emphasizing extensibility and SQL compliance. PostgreSQL is combined with many features that safely store and scale the most complicated data workloads [33], which works ideally with the Blockchain data. PostgreSQL is aimed to help build applications to protect data integrity and build fault-tolerant environments, keeping the data security a most. It manages the data no matter how big or small the dataset. PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Heroku PostgreSQL is an add-on for deployed applications, it operates as an advanced secure scalable database that allows for building data-driven application while relying on Heroku’s expertise and fully managed platform to build, operate, secure, and validate compliance for their data stack. [32]

3.4 GitHub

GitHub is a cloud-based hosting service, a hosting software development platform that stores, tracks, and allows for the collaboration on software project. It allows for software developers to create remote, public repositories on the cloud. GitHub also serves as a social networking site in which developers can openly network collaborate and follow methodology cycles. [34] GitHub's enhanced collaboration allows all project members of a team to create branch, this stops members interfering with each other's works, a branch is a separate development area allowing more efficient work by the team. Once a functionality is complete then a pull request is made combining work created and once everything is lined up a merge is made between the team's branches. [34] GitHub version control allows developers track and manage project code changes, this lets developers safely work through the branching and merging process, allowing to also revert these changes if needed. [35] GitHub provides tools that are great for planning and building projects, GitHub Project feature allows for the creation of Kanban boards that teams can use for user stories when following Agile methodology.

Chapter 4

System Design

Blockchains most be robust and designed with no loose functionality that can cause gaps in the design, as the blockchain is a financial system dealing with user's cryptocurrency, a solid design of the blockchain is the primary goal. Strict programming principles are followed in the design of the blockchain these principles are, encapsulation, abstraction, inheritance, and polymorphism, these can be seen throughout the design of the system to keep the blockchain as secure as possible. Blockchains can be designed using multiple different types of technology and different algorithms depending on what the main use of the blockchain is, however all blockchains follow a similar set of protocols that are needed for a blockchain to work and are implemented in every blockchain design. This chapter gives a detailed system design of the blockchain architecture, informed by the researched technology in the previous chapter and how it is applied in the design of the blockchain to meet the requirements of the project. The system design of the blockchain is explained and all-important functionality of the blockchain is covered and explained using code and the research that was made prior in the planning stage of the project. Front-end that allows users to access the blockchains functionality is explained in this chapter, the architecture of the front-end, components, UML diagrams and connectivity to cloud platform is explained in detail providing all the information within depth research. The design of the back-end service that provides a connection between the blockchain and front-end is design to support the user with the needed services, the design makes sure all requests made by the user are validated and don't interfere with the blockchain functionality in the back-end. The system design is built on the research made prior to development, all system architecture is designed to meet the requirements of the project and to fulfil the required protocols of the blockchain while also providing a simple and functional design for front-end for a good user experience to be accomplished.

4.1 Blockchain Design

The system design of the blockchain is based for it to work as a financial system, as the goal of the project is to develop a coin-oriented blockchain that allows users to exchange cryptocurrency. The blockchain is designed following SOLID programming principle, keeping the blockchain secure and components coupled. The blockchain needs to follow the correct protocols to function to its full potential, the Java programming language will help the design be more robust. The blockchains main functionality are, blockchain blocks, consensus algorithm, wallet private and public keys, transaction, and cryptography. Figure ?? shows the design of the blockchain and how its components are linked together for the blockchain to function.

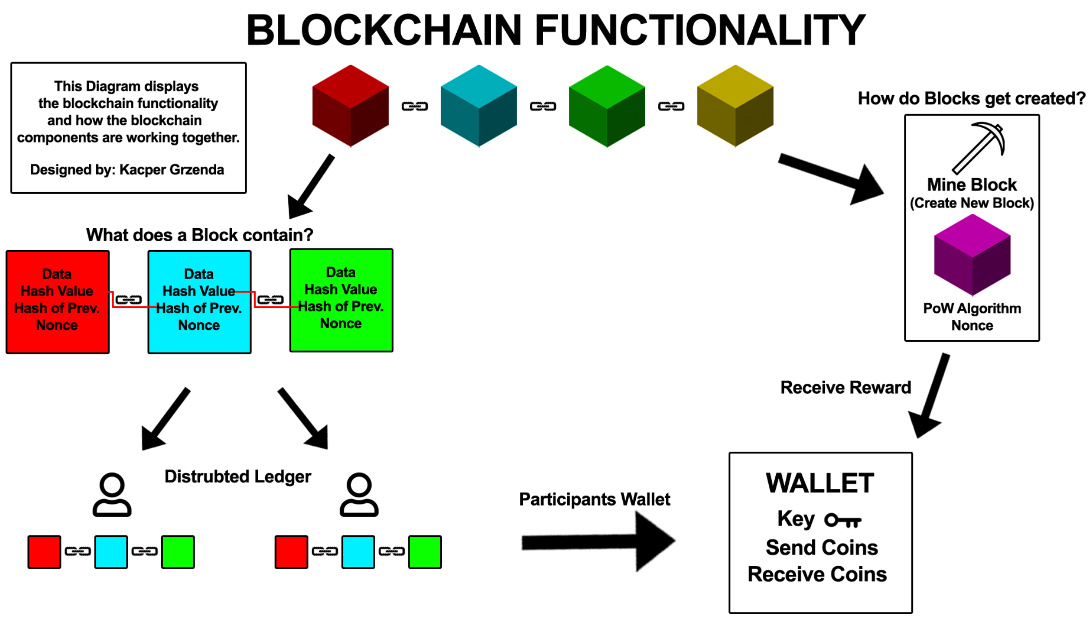

BLOCKCHAIN FUNCTIONALITY
This Diagram displays the blockchain functionality and how the blockchain components are working together.
Designed by: Kacper Grzenda
How do Blocks get created?
Mine Block
(Create New Block)
PoW Algorithm
Nonce
What does a Block contain?
Data
Hash Value
Hash of Prev.
Nonce
Receive Reward
Distrubted Ledger
Participants Wallet
WALLET
Key
Send Coins
Receive Coins

Figure 4.1: Blockchain System Design.

4.1.1 Block

Blocks are permanently stored inside the blockchain in chronological order, each block is a data structure that holds data relevant to the blockchain and transactions made on the blockchain. Everyone block is unique and the data inside a block is unable to be changed, if the data inside a block was to be changed the blockchain network would have to agree for this change to occur requiring immense amounts of computing power. So much computing

```
public ArrayList<Transaction> transactions = new ArrayList<Transaction>();

public Block(int index, String previousHash) {
this.index = index;
this.timestamp = dateFormatter.format(date);
this.previousHash = previousHash;
this.nonce = 0;
this.hash = calculateHash();
}
```

Figure 4.2: Java Code of a Block

power is required that modifying blocks is effectively impossible to do. A block contains an index this number specifies the position of the block in the blockchain, timestamp indicating when a block has been created, hash this is a 256-bit unique value that every block contains this value is unable to be changed, previous hash of the block previously mined, nonce is the number added to the hash to label the difficulty of the blockchain, each block contains a transaction list of each verified transaction. The Merkle-root algorithm is also used to create a unique ID, this ID is used in the generation of the hash to make each block even more secure, the Merkle root is explained in the cryptography section. The hash values in the block are created by the SHA-256 algorithm, this is explained in more detail in the ”Cryptography” section. Blocks store the most recent transactions that occur during there ”open” period of time, this can be from one transaction to one-thousand transactions. A block is ”closed” when it is mined by a user on the blockchain network, once the block is mined it is added onto the blockchain, this block is then permanently saved and cannot be tampered with as it is visible by the entire network, keeping it from being changed. Figure -4.2 showcases the Constructor and Array-List of transactions of every block created.

4.1.2 Proof Of Work Algorithm

Proof of work consensus algorithm is used to confirm transactions and produce a new block to the blockchain. Participants, mine a block to try and solve a puzzle, the puzzle is a hash that needs to be cracked by the computer, whoever solves the puzzle mines the block. The participant who mines the block first receives a cryptocurrency as the reward. The difficulty of the puzzle can be increased, making each block harder to mine, the difficulty is increased by making the hash being solved larger. In this design the difficulty

```
public void mineBlock(int difficulty) {
        System.out.println("Mining Block...");
        this.merkleRoot = BlockchainCryptography.getMerkleRoot(transactions);
        String target = new String(new char[difficulty]).replace('\0', '0');
        while(!this.hash.substring( 0, difficulty).equals(target)) {
                nonce ++;
                this.hash = calculateHash();
        }
        System.out.println("Block Mined!!! : " + hash);
}
```

Figure 4.3: Proof of Work Consensus Algorithm.

is set to a final value that doesn't change, this is to allow the participants mine blocks in a reasonable time with low amount of computational power. The mining of the block is a long process that involves multiple validations that involves the blocks transactions and hashes of each block to make sure the block isn't tampered with, once the validations are passed and the blockchain is confirmed to be valid the block is added to the blockchain and the PoW algorithm is complete. The proof of work algorithm uses components from the cryptography section to help solve the puzzle, the components used are Merkle-tree algorithm which is part of the transaction security and the SHA-256 algorithm that creates a hash. Figure -4.3 is part of the proof of work algorithm, this specific piece of code loops until the puzzle is solved.

4.1.3 Wallet Private & Public Keys

Participants of the blockchain network have wallets to access their cryptocurrency. Wallets are you used to send and receive cryptocurrency, each wallet is generated with a private and public key, these keys are used to create transactions. The private key acts as a password, the user needs to keep their private key safe, as this is used to sign signatures and connect your wallet. Public key is used as an address, when a transaction is being created a public key, most be provided, for example a user is sending funds to another user, the receiver needs to provided their public for the sender to send them cryptocurrency. When a wallet is generated a public key, private key and the wallet balance is stored in each wallet that is then saved onto the blockchain. Elliptic curve digital signature algorithm is used to create key pairs, the public and private key, the key pairs are created using a random 256-bit for security. Figure - 4.4 is the generation of the pair of keys algo-

```
public void generateKeyPair() {
if(Security.getProvider("BC") == null) {
      Security.addProvider(new BouncyCastleProvider());
}
try {
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("ECDSA","BC");
        SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
        ECGenParameterSpec ecSpec = new ECGenParameterSpec("prime192v1");
        // Initialize the key generator and generate a KeyPair
        keyGen.initialize(ecSpec, random);   //256 bytes provides an acceptable
            KeyPair keyPair = keyGen.generateKeyPair();
            // Set the public and private keys from the keyPair
            privateKey = keyPair.getPrivate();
            publicKey = keyPair.getPublic();
}catch(Exception e) {
        throw new RuntimeException(e);
}
}
```

Figure 4.4: Elliptic Curve Digital Signature.

rithm. The key pairs are locked together, the ECDSA is used in the signing and verification process of the digital signature, the signing process involves the private key to complete the signature, and to verify the signature the paired public key is needed to complete the verification.

4.1.4 Transactions

Blockchain Transaction follows a strict process, from creating the transaction to approving it. A transaction holds key information for it to be easily identified making each transaction as transparent as possible. Transactions have unique IDs that use the SHA-256 algorithm to create a 256-bit String that is the transaction ID. Transactions follow a verification process that is part of the blockchain security protocol, participants keys are used in the transaction verification process, private key is used to sign a transaction and the public is used to decrypt and verify it. When a transaction is created the verification process involves a check on the amount of cryptocurrency that is being sent, as the blockchain stores all transactions that have been previously been created, each one of the transactions is checked to see the total of the cryptocurrency that has been linked to the wallet and if the user has enough to send to the receiver, if true the transaction is complete. The output transaction funds, and the input transactions funds are saved in lists that are stored on the Blockchain for all participants to see and be used for the next transactions being created. The transaction input and output data are stored with unique IDs that can be linked to the transaction that it was created from. This process allows no cryptocurrency to lost and tampered with keeping track of all transaction records that occur. Figure -4.5 shows a visual example of the link between the inputs and outputs.

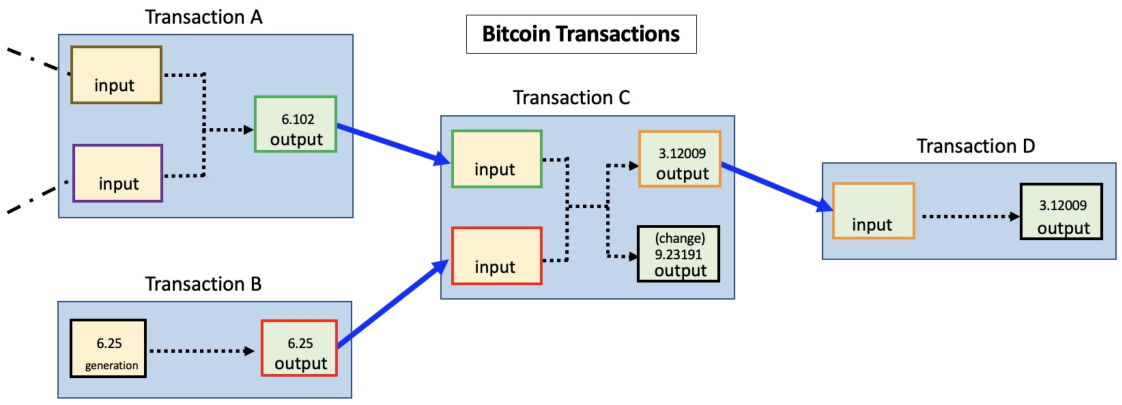


Figure 4.5: Transaction Output and Input Connection

4.1.5 Cryptography

Cryptography is the most important part of the blockchain as it is used in every part of the functionality to keep the blockchain secure and immutable. the blockchain consists of many elements that need to be encrypted and need to be kept unreadable unless there exists a private key to decrypt it. When an element is in its encrypted for, the data may be of unlimited size. The SHA-256 is a cryptographic has function that outputs a value that is 256 bits long. This hash function is used in the hashing of blocks and transaction to keep them secure, it is infeasible for an attacker to compute collisions, making the SHA-256 the perfect solution to securing the data inside the blocks and transactions. The hash function works by inputting a string that is there converted into bytes, each byte is then converted in a Hexadecimal and appended onto a new string sequence, creating a hashed string that can be used for as unique identifier. These unique hashed strings are used to identify blocks data, if someone would try change a piece of data in a block the hashed string would drastically change, this would then notify the

```java
public static String convertToHash(String convertToHash) {
    try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.digest(convertToHash.getBytes("UTF-8"));
            StringBuilder hexString = new StringBuilder();
            for (int i = 0; i < hash.length; i++) {
                    final String hex = Integer.toHexString(0xff & hash[i]);
                    if(hex.length() == 1)
                            hexString.append('0');
                    hexString.append(hex);
            }
            return hexString.toString();
    } catch(Exception ex){
            throw new RuntimeException(ex);
    }
}
```

Figure 4.6: SHA-256 Hash Algorithm.

blockchain of the change and alarm the system of a breach and no change would be made to the blockchain blocks. Figure -4.6 is the algorithm used in the blockchain to convert a data string into a hash value.

Merkle tree algorithm is used in the blockchain system to furthermore guarantee immutability of the blockchain. Merkle tree is a hash-based data structure that is generalisation of the transaction hash list. The Merkle tree encodes the current transactions and its data in to create on hashed string called the Merkle root. As every transaction has on the blockchain has a hash associated with it, and each transaction is stored in a tree-like structure as explained in the transaction section, the Merkle tree algorithm loops through each transaction to create a secure piece of data that is a hashed value that is the Merkle root, this Merkle root is then added to the current block to be then converted in to another hash value that is the block hash for the blockchain to store, this guarantees immutability and uniqueness of each block and also helps save the transaction data of all previous transaction created. Figure -4.7 is the Merkle tree algorithm used in the project to create a unique Merkle root for each block.

Elliptic Curve Digital Algorithm is cryptographic algorithm used to ensure that funds can be only spent by their rightful owner. The algorithm is dependent on the curve order and the hash function used, in this case being the SHA-256. The design of the algorithm consists of three main parts, the

```java
public static String getMerkleRoot(ArrayList<Transaction> transactions) {
int count = transactions.size();
ArrayList<String> previousTreeLayer = new ArrayList<String>();
for(Transaction transaction : transactions) {
        previousTreeLayer.add(transaction.transactionId);
}
ArrayList<String> treeLayer = previousTreeLayer;
while(count > 1) {
        treeLayer = new ArrayList<String>();
        for(int i=1; i < previousTreeLayer.size(); i++) {
                treeLayer.add(convertToHash(previousTreeLayer.get(i-1)
                                + previousTreeLayer.get(i)));
        }
        count = treeLayer.size();
        previousTreeLayer = treeLayer;
}
String merkleRoot = (treeLayer.size() == 1) ? treeLayer.get(0) : "";
return merkleRoot;
}
```

Figure 4.7: Merkle Tree Algorithm.

```
public static byte[] applySignature(PrivateKey privateKey, String input) {
        Signature dsa;
        byte[] output = new byte[0];
        try {
                dsa = Signature.getInstance("ECDSA", "BC");
                dsa.initSign(privateKey);
                byte[] strByte = input.getBytes();
                dsa.update(strByte);
                byte[] realSig = dsa.sign();
                output = realSig;
        } catch (Exception e) {
                throw new RuntimeException(e);
        }
        return output;
}
```

Figure 4.8: Create Signature using Private Key.

private key, public key, and signature. The private and public key are a pair that are randomly generated, the private key is private just like its name specifies. The public key does not need to be kept secret. A public key can be used to determine if a signature is genuine by using ECDSA to validate that a signature was generated by the private key. A signature is mathematically generated from a hash of the message being signed and a private key. With the public key, a mathematical algorithm can be used on the signature to determine that it was originally produced from the hash and the private key. This can be seen in Figure -4.9 as the ”verifySignature” methods parameter is a public key that is used to verify the signature and Figure -4.8 is the decryption, ”applySignature” methods parameter is a private key that is used to create the signature.

4.2 Front-End Design

Front-End is designed to be responsive, reactive, and simple to use to create a good user-experience. As blockchains can be a difficult to understand, the front-end needs to be simple and easy for the user to be able to use the necessary functionality available to them that the blockchain provides. Angular framework is used for this process to be as smooth as possible. Angular provides many features and the documentation is well detailed creating a good

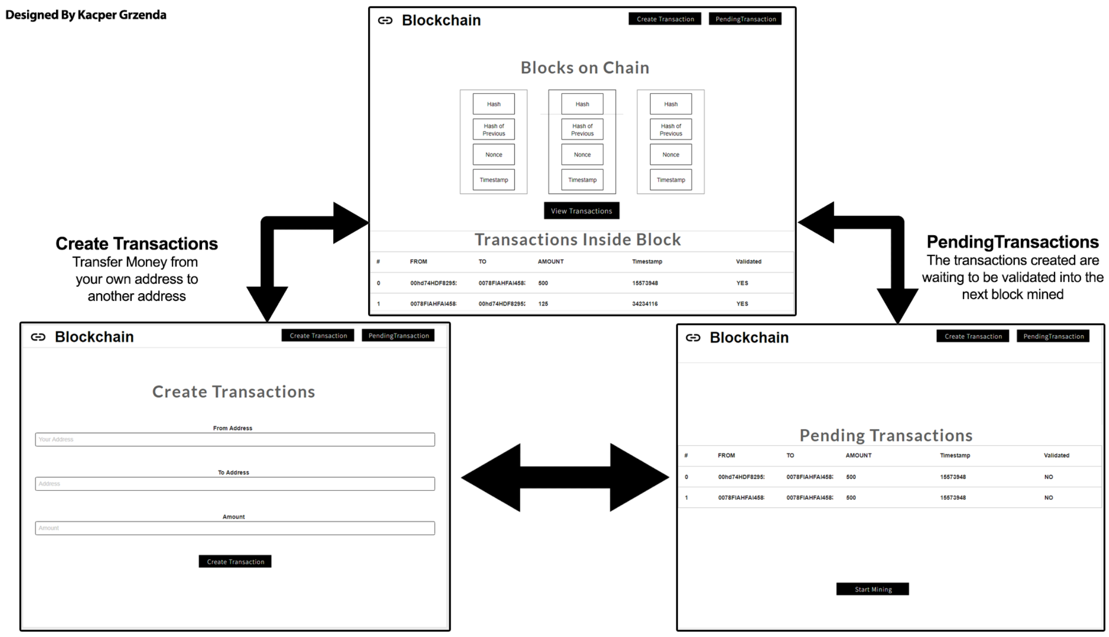
```
public static boolean verifySignature(PublicKey publicKey,
                                       String data,
                                       byte[] signature) {
        try {
                Signature ecdsaVerify = Signature.getInstance("ECDSA", "BC");
                ecdsaVerify.initVerify(publicKey);
                ecdsaVerify.update(data.getBytes());
                return ecdsaVerify.verify(signature);
        }catch(Exception e) {
                throw new RuntimeException(e);
        }
}
```

Figure 4.9: Verify Signature.

starting environment for the front-end. Angular material is used to make the front-end visually appealing and provide components the user can use to have a more reactive experience learning and using the blockchain functionality.

4.2.1 Page Description.

The front-end consists of three pages, Home page, Mining page and Transaction Page, Figure -4.10 is a wire frame that displays the design of what the three pages should look like. Each page has a navigation bar to allow the user to navigate through the three pages. Home page is designed to display the blockchain information, the is a list of blocks that are part of the blockchain. Each block displays its data, its ID, hash, timestamp, previous hash, and transactions that it contains. The transaction that are displayed are part of its current block, the transaction display ID, sender public key, recipient public key and amount of cryptocurrency that was sent. Mining Page allows the participant to mine to current block, this is done by clicking the ”Mine Block” button that sends a request to the server and processes the request. A user is rewarded with cryptocurrency to their wallet when they mine a block. The user can access their wallet through the navigation bar, this opens the user’s wallet information, displaying their unique public key, private key, and balance. The wallet information is very responsive and responds to any action that occurs on the blockchain. The transaction pages allow a user to send funds from their wallet to other wallets on the blockchain, this is done using keys. The transaction page has a form that a user fills out, in the form the user needs to provide their private key, receivers



#	FROM	TO	AMOUNT	Timestamp	Validated
0	00hd74HDF8295:	0078FIAHFAI458:	500	15573948	YES
1	0078FIAHFAI458:	00hd74HDF8295:	125	34234116	YES

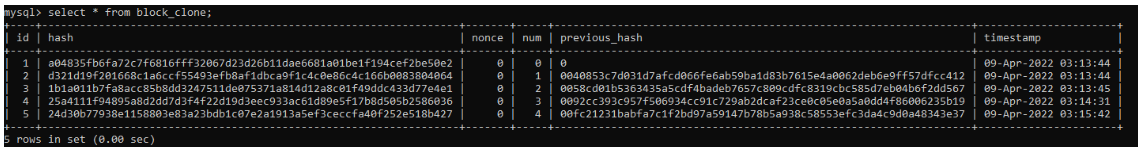
#	FROM	TO	AMOUNT	Timestamp	Validated
0	00hd74HDF8295:	0078FIAHFAI458:	500	15573948	NO
1	0078FIAHFAI458:	0078FIAHFAI458:	500	15573948	NO

Figure 4.10: Blockchain System Design.

public key and the amount of cryptocurrency they want to send. Once the form is filled out the users clicks the button at the bottom of the screen this transaction request is processed and handled by the blockchain.

4.3 Back-End Service

The back-end service is the request handler and the bridge between the front-end and blockchain functionality. This service handles all requests from the user and sends them to the blockchain to be processed and validated. Spring-Boot does a great job providing this service with its online initialiser and the Maven project provides all the needed dependencies for the service to function correctly. The Blockchain service also is designed to save blockchain blocks into a MySQL database, the database saves the blocks data. This is server is designed to save blocks into a MySQL database locally and when hosted of the Heroku cloud platform the blocks are saved into a PostgreSQL database, this is because Heroku supports PostgreSQL and is a free database to use. The blocks are saved into the database after they have been validated this is to ensure the correct blocks are being saved. Figure -4.11 represents the block data that is been stored in the database, the blocks being saved are the current blocks that exists on the chain.



```
mysql> select * from block_clone;
+----+------------------------------------------------------------------+-------+-----+------------------------------------------------------------------+----------------------+
| id | hash                                                             | nonce | num | previous_hash                                                    | timestamp            |
+----+------------------------------------------------------------------+-------+-----+------------------------------------------------------------------+----------------------+
|  1 | a04835fb6fa72c7f6816fff32067d23d26b11dae6681a01be1f194cef2be50e2 |     0 |   0 | 0                                                                | 09-Apr-2022 03:13:44 |
|  2 | d321d19f201668c1a6ccf55493efb8af1dbca9f1c4c0e86c4c166b0083804064 |     0 |   1 | 0040853c7d031d7afcd066fe6ab59ba1d83b7615e4a0062deb6e9ff57dfcc412 | 09-Apr-2022 03:13:44 |
|  3 | 1b1a011b7fa8acc85b8dd3247511de075371a814d12a8c01f49ddc433d77e4e1 |     0 |   2 | 0058cd01b5363435a5cdf4badeb7657c809cdfc8319cbc585d7eb04b6f2dd567 | 09-Apr-2022 03:13:45 |
|  4 | 25a4111f94895a8d2dd7d3f4f22d19d3eec933ac61d89e5f17b8d505b2586036 |     0 |   3 | 0092cc393c957f506934cc91c729ab2dcaf23ce0c05e0a5a0dd4f86006235b19 | 09-Apr-2022 03:14:31 |
|  5 | 24d30b77938e1158803e83a23bdb1c07e2a1913a5ef3ceccfa40f252e518b427 |     0 |   4 | 00fc21231babfa7c1f2bd97a59147b78b5a938c58553efc3da4c9d0a48343e37 | 09-Apr-2022 03:15:42 |
+----+------------------------------------------------------------------+-------+-----+------------------------------------------------------------------+----------------------+
5 rows in set (0.00 sec)
```

Figure 4.11: MySQL Database Table.

Chapter 5

System Evaluation

The objective of the project was to create a robust blockchain that is accessible to anyone through the browser, this chapter evaluates the project by showcasing the robustness of the blockchain, functional user friendly front-end and spring-boot service that is accessible by both back-end and front-end. All parts of the project are tested in their own ways to evaluate their design and the objectives they were set with. The objects that were set at the beginning are checked and the system is evaluated on these results making sure the blockchain follows all necessary protocols while also having the correct functionality to work as a coin-oriented system.

5.1 Blockchain Functionality Evaluation

The main objective set at the beginning of the project was to create a secure immutable blockchain for anyone to use and eliminate any trust factors as the blockchain will handle all the security and validation functionality. The blockchain system features three main validation processes these processes are for the initial blockchain and block verification, transaction validation and cryptocurrency validation.

5.1.1 Blockchain All or Nothing Validation

The blockchain validation is the strictest process that features in the blockchain this is an all or nothing check to validate every detail in the blockchain. The process consists of looping through the entire blockchain making sure that each block belongs in the blockchain, checking the blocks hash and previous hash, then the blocks transactions are checked if they were tampered with this includes checking the transactions inputs and outputs. If any small detail is changed or added to the blockchain that should not exists the blockchain instantly fails and doesn't not allow a block to be added onto the chain, this allows for no outside source tampering with the blockchain. If the validation process passed the block can be added. The validation is in the "Blockchain.java" class in the GitHub repository here as it is to long to be displayed in the dissertation.

5.1.2 Transaction Validation

Transaction validation process exists to check if the new transaction created is not tampered with and followed the correct protocols before allowing it to be added into the current block. The transaction goes through a process that checks its signature, correct cryptocurrency amount and processes the transaction inputs and outputs adding them to the blockchains data. This process is used in the "Block.java" class before the transaction is added into the blocks data, this is mainly to not allow users to give themselves cryptocurrency into their wallets. The validation is passed if it returns "True" to the main method in the block class then a the transaction is added to the block, if the process gets interrupted it notifies the user with the reasoning of the invalid transaction by a System.Out.Println(); function. Figure -5.1 is the validation process that each transaction goes through before being added to the block transaction data list.

5.1.3 Signature Verification

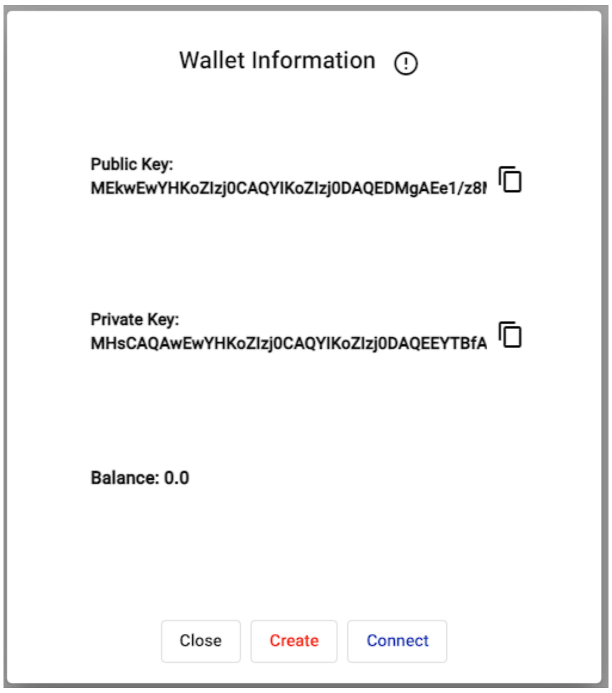
Signature verification is the process of using a digital signature algorithm and a public key to verify a digital signature on data that was signed with a private key. This verifies the correct owner of the data as the key pairs only work together, no one else can decrypt the data without having the paired public key. This validation process is part of the transaction validation and blockchain validation to make sure that the identity of the transaction is correct. If the signature verification is approved "True" is returned. If the process fails it fails the blockchain validation and also the transaction validation, the signature verification is an important validation process in the security of the blockchain it follows the elliptic curve digital signature algorithm.

```java
public boolean processTransaction() {

if(verifiySignature() == false) {
        System.out.println("Transaction Signature failed to verify");
        return false;
}

for(TransactionInput i : inputs) {
        i.unspentTXOutput = Blockchain.UTXOs.get(i.transactionOutputId);
}

//Checks if transaction is valid:
if(getInputsValue() < 0.1f) {
        System.out.println("Transaction Inputs too small: " + getInputsValue());
        System.out.println("Please enter the amount greater than " + 0.1f);
        return false;
}

//Generate transaction outputs:
float leftOver = getInputsValue() - amount;
//get value of inputs then the left over change:
transactionId = calculateTransactionHash();
//send value to recipient
outputs.add(new TransactionOutput( this.reciepient, amount,transactionId));
//send the left over 'change' back to sender
outputs.add(new TransactionOutput( this.sender, leftOver,transactionId));

//Add outputs to Unspent list
for(TransactionOutput o : outputs) {
        Blockchain.UTXOs.put(o.id , o);
}

//Remove transaction inputs from UTXO lists as spent:
for(TransactionInput i : inputs) {
    //if Transaction can't be found skip it
        if(i.unspentTXOutput == null) continue;
        Blockchain.UTXOs.remove(i.unspentTXOutput.id);
}

return true;
}
```

Figure 5.1: Verify Transaction.

5.2 Front-End Evaluation

Front-End evaluation is based on the responsiveness of the GUI elements and how they react to input. The main objective of the front-end was to create a good user experience that helps the user to understand, locate and use the available blockchain functionality. The layout was designed to be a simple learning curve for the user to quickly understand what they are looking at. This was done by reducing the amount of information that is displayed to the user and not over complicating the layout of the front-end. The GUI elements were tested by manually clicking and seeing the output of the input change, this can be seen in the wallet section Figure -5.2 where the wallet information responds instantly to any change made on the blockchain. Visual regression testing was conducted during the coding process, where checks were made when a code change had been made to the front-end, comparing the old version to the new version that was changed.


Wallet Information

Public Key:
MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEe1/z8!

Private Key:
MHsCAQAwEwYHKoZIzj0CAQYIKoZIzj0DAQEEYTBfA

Balance: 0.0

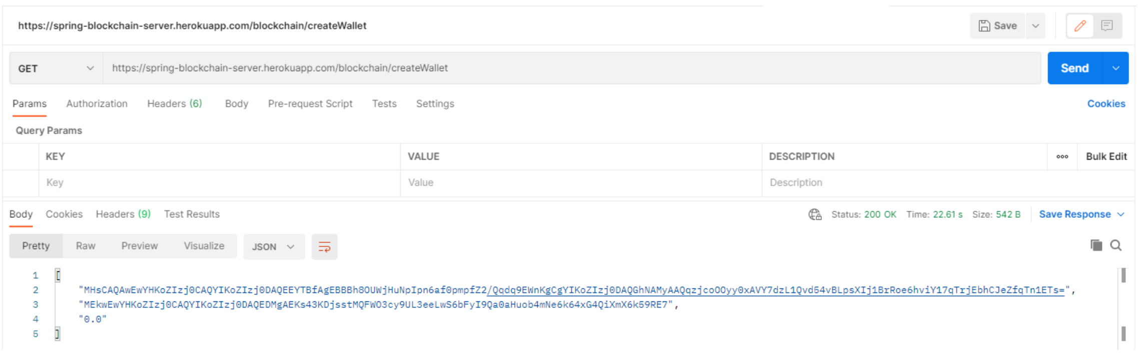
Figure 5.2: Responsive Wallet.

5.2.1 Front-End Deployment

The objective of the Front-End was to be accessible by anyone from any browser, so the front-end was built and deployed to Heroku, this was achieved by pushing the front-end code to a Heroku repository and allowing the Heroku server to change the URL links in the code, by accessing the ”server.js” file in the ”blockchainapp” folder. The deployed front-end was compared to the local front-end and followed all the same testing processes. The deployed front-end can be accessed at https://spring-angular-blockchain.herokuapp.com/.

5.3 Back-End Service Evaluation

Back-End is evaluated on the requests that are sent through the Spring-Boot service, the requests should work locally and on the cloud. The objective of the service was to create a bridge between the blockchain and front-end this was achieved by creating a spring controller that processes incoming requests from the user and return a value from the blockchain. Before anything is returned the requests is sent from the controller to the service where it is processed and then the service reaches out to the blockchain and a result is returned, the result is sent back to the controller where the value is returned to the users. The MySQL database connected to the Spring-Boot server was evaluated by scanning and checking that the correct data is being saved into the database and no leaks of the data are occurring. This requests handling was evaluated using Postman, all requests were sent through postman and tested making sure the correct value was returned, this was tested for the local server and for the deployed server on Heroku located at https://spring-blockchain-server.herokuapp.com/. Figure -5.3 is an example of a test that was performed for the deployed server, in this test the ”createwallet” GET request is being tests to check if the correct values are returned in body when a new wallet is being generated. Once all the tests were completed of the Back-End service the blockchain was fully compatible with the front-end allowing for all requests to work correctly.


https://spring-blockchain-server.herokuapp.com/blockchain/createWallet

GET https://spring-blockchain-server.herokuapp.com/blockchain/createWallet

Status: 200 OK Time: 22.61 s Size: 542 B

```
[
    "MHsCAQAwEwYHKoZIzj0CAQYIKoZIzj0DAQEEYTBfAgEBBBh8OUWjHuNpIpn6af0pmpfZ2/Qqdq9EWnKgCgYIKoZIzj0DAQGhNAMyAAQqzjcoOOyy0xAVY7dzL1Qvd54vBLpsXIj1BrRoe6hviY17qTrjEbhCJeZfqTn1ETs=",
    "MEkwEwYHKoZIzj0CAQYIKoZIzj0DAQEDMgAEKs43KDjsstMQFWO3cy9UL3eeLwS6bFyI9Qa0aHuob4mNe6k64xG4QiXmX6k59RE7",
    "0.0"
]
```

Figure 5.3: Generate Wallet GET Request Test.

5.3.1 Back-End Service Limitation

As a big part of a blockchain is to be a distributed ledger, where every participant on the network has a copy of the blockchain, spring-boot limits the back-end to be distributed as it has no feature that works directly to distribute the back-end among multiple nodes. From research undertaken Spring-Boot needs an outside software add-on to help it be distributed, limiting the possibility of a distributed blockchain. However, to have a backup of the blockchain stored is a necessity in case of any problem in the system. This is why MySQL and PostgreSQL have been used to have a stored version of the blockchain at all times.

Chapter 6

Conclusion

The final chapter brings the thesis to a close, summarising the conclusions of the project. The rational and initial goals of the project are discussed. The final chapter also briefly discusses the outcomes of the project, the insight that were gained by happenchance during the project, the opportunities that were identified for future investigations of the project topic and an overall conclusion on the project.

The work presented in the thesis details the design and development of a functional coin-oriented blockchain system. The system is a robust, immutable blockchain that is tamper proof from any attacks eliminating the need and trust factor of a third party. The project objectives aimed to deliver a functional blockchain that is also accessible by a front-end through a browser for any user that wants to be a participant of the network. The bridge between the front-end and blockchain is built by a Sprint-Boot framework, handling requests through a local or deployed environment, the Spring framework also backup the blockchain blocks in a MySQL database and PostgreSQL database. The project follows all necessary blockchain protocols, to create a functional blockchain, the main protocols being consensus algorithm (Proof of Work), cryptography, immutability, and decentralization. Additional functionality is added to the design on top of the main functionality protocols of the blockchain to make it a coin-oriented blockchain to achieve the project objective.

6.1 Project Findings

During project development, expected and new findings were discovered throughout the research and evaluation stage. This section discusses the outcomes of the project, insights that were gained during while working on project and opportunities that were identified for future investigation.

6.1.1 Outcomes

- Consensus Algorithm: A Proof of Work consensus algorithm was developed to validate blocks onto the blockchain and allow mining new blocks.
- Cryptography: SHA-256 cryptographic algorithm is used to create encrypted hash values to secure the blockchain data making the data non-reversible. Elliptic Curve Digital Signature algorithm is used to generate pairs of keys (public and private) to create digital signature verifying ownership of cryptocurrency.
- Immutability: Immutability is achieved by the validation process that occurs on the blockchain, this process checks every little detail before approving a block being added. Also because of the design of the blocks that exists on the blockchain, each block is linked to the next block because of the data that exists inside the blocks, if someone tried to tamper with the blockchain they would need to have control of all blocks.
- Decentralization: The full decentralization functionality was not achieved, this is because the blockchain is not a distributed ledger, it only exists on one node and is being save to a centralised database. However, the fix to this problem is discussed in the future investigation section.
- Validation: The blockchain architecture is designed to be validated in and out, it has three main validation algorithms (blockchain all element validation, transaction validation, signature validation) to eliminate all security risks and eliminating the trust factor of a third party.
- Front-End: The front-end was implemented exactly as indented achieving the goal of a good user-experience. This front-end is responsive and reactive to all inputs and showcase all the intended blockchain functionality for the user to use. The front-end has a simple learning curve allowing the user to understand all functionality.
- Back-End Framework: The back-end creates the bridge for requests to be sent between the blockchain and front-end as intended, with all requests tested using Postman and working correctly. However, the service is not fully implemented as intended, it lacks request validation

in the case of an invalid request and the blockchain is not distributed to multiple nodes.

- Hosting: Hosting has been achieved for both the front-end and back-end service with both being deployed on the Heroku cloud and accessible through an internet browser.
  Service: https://spring-blockchain-server.herokuapp.com/
  Front-End: https://spring-angular-blockchain.herokuapp.com/

6.1.2 Insights

During the research, knew insights were discovered related to the project and unrelated insights specific to blockchains. An insight was made when researching the project, a conversion was had with a blockchain developer, an insight on consensus algorithm was made, that there exist different algorithms that have better security percentage and different use depending on how fast, secure, and cost specific the blockchain is being designed to be. An insight was also made was during the deployment stage, where there was a problem managing local JAR dependencies in the back-end (Spring-Boot Maven project), a local JAR dependency file has to have the same Java library as the existing project.

6.1.3 Future Investigation

For future investigation, to make the blockchain a fully implemented decentralized ledger an additional technology can be added. A cloud network architecture could be developed to keep a copy of the blockchain on multiple nodes. Apache Kafka can be implemented to distribute the blockchain into to multiple nodes these nodes would work together, each node would contain a copy of the blockchain. Kafka would help with zero data loss and zero down time as the blockchain would be a distributed ledger on multiple nodes, and if one of the nodes was tampered with a check would be done by all nodes in the network to see who the imposter is and would be replaced with a new clean node. This would mean the MySQL and PostgreSQL database can be removed as the blockchain will be stored on multiple nodes on the cloud.

6.2 Project Conclusion

Blockchain technology is still new, but its potential is enormous. This project showcases a section of what blockchain technology is, how blockchains function and how they can be used by anybody with an internet connection.

The project was a learning process of what it is like developing a functional blockchain and many new skills were acquired such good programming practices, writing solid algorithms, and learning new technologies throughout the research and development cycle. It is a big accomplishment to develop a functional robust blockchain that is accessible on the browser, and users can use it to trade their cryptocurrency on the network. However, there is still a lot to learn about blockchain technology and how it can used to transform the financial system by eliminating the trust factor of a third party.

Bibliography

[1] A. Body, “Blockchain : how to choose the right tech for your business.”

[2] J. FrankenField, “Cryptocurrency.”

[3] V. Buterin, “A next generation smart contract decentralized application platform.”

[4] IBM, “Smart contracts defined.”

[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system.”

[6] J. Library, “The legion of the bouncy castle.”

[7] Conor, “How to choose the right blockchain technology.”

[8] M. Somers, “The risks and unintended consequences of blockchain.”

[9] S. Madnick, “Blockchain is unbreakable? think again..”

[10] G. Wood, “Solidity.”

[11] M. Draper, “The most popular programming languages used in blockchain development.”

[12] D. Yankiever, “Programming languages used for blockchain development.”

[13] J. Gosling, “Java.”

[14] Chirag, “15 best programming languages for blockchain app development.”

[15] T. UI, “User interface.”

[16] T. Front-End, “Frontend.”

[17] Angular, “Introduction to angular concepts.”

[18] Typeescript, “Typescript configuration.”

[19] N. Baglivo, “A functional (programming) approach to error handling in typescript.”

[20] A. Material, “What is angularjs material?.”

[21] S. Bos, “java basics: What is spring boot?.”

[22] Spring, “Spring boot.”

[23] S. Initializr, “Spring initializr.”

[24] G. J. Knowledge, “How spring boot aplication works internally?.”

[25] I. gaba, “What is maven: Here’s what you need to know.”

[26] Amazon, “What is a relational database?.”

[27] MySQLtm, “What is mysql?.”

[28] G. Romero, “What is postman api test.”

[29] Azure, “What is cloud computing?.”

[30] Heroku, “What is heroku?.”

[31] H. D. Center, “How heroku works.”

[32] H. Management, “Managed postgresql from heroku.”

[33] PostegreSQL, “What is postgresql.”

[34] J. Juviler, “Whats is github?(and what is it used for?.”

[35] Glossary, “What is github? a beginner’s introduction to github.”