

Cover Page

**STUDENTS, PLEASE COPY THIS PAGE AND USE AS THE COVER PAGE FOR
YOUR FINAL SUBMISSION**

Module No:	COMP6013	Module title:	Computing Project
Degree Programme :		Computer Science for Cyber Security	
Project title :		hChat - Blockchain Messaging Application	
Supervisor :		Hashem Dehghanniri	
Due date and time:		1pm 17th April 2023	
Estimated total time to be spent on assignment:			90 hours per student
Student No:		Student Name:	
19067074		Harrison John Stoddart	

Statement of Compliance (please tick to sign)

☒ I declare that the work submitted is my own and that the work I submit is fully in accordance with the University regulations regarding assessments (www.brookes.ac.uk/uniregulations/current)

Regulations governing the deposit and use of Oxford Brookes University Projects and Dissertations

Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Library.

Statement of Permission (please tick to sign)

<input checked="checked" type="checkbox"/>	I agree that this dissertation may be available for reading and photocopying in accordance with the Regulations governing use of the Library.
--	---

LEARNING OUTCOMES

On successful completion of this module, students will be able to achieve the module following learning outcomes (LOs):	
1	Create, design, manage, plan, carry out, and evaluate a project involving the solution of a practical problem set in an appropriate social and economic context, taking into account other relevant factors such as risk
2	Apply practical and analytical skills acquired in the programme to the investigation of a substantial topic
3	Apply the scientific method and report findings using accepted formalisms
4	Identify and utilise trustworthy information sources, such as the ACM Digital Library to develop a coherent understanding of issues in the domain
5	Demonstrate the ability to carry out a substantial piece of work independently and critically evaluate the student's achievements and their own personal development
6	Use appropriate technologies such as online libraries and databases to find, critically evaluate and utilise both non-specialist and technical information pertinent to the project
7	Demonstrate an awareness of and work in a manner guided by the legal, professional, ethical, security and social issues relevant to the IT and telecommunications industry

Engineering Council AHEP4 LOs assessed (from S1 2022-23):	
B3	Select and apply appropriate computational and analytical techniques to model broadly-defined problems, recognising the limitations of the techniques employed
B4	Select and evaluate technical literature and other sources of information to address broadly-defined problems
B5	Design solutions for broadly-defined problems that meet a combination of societal, user, business and customer needs as appropriate. This will involve consideration of applicable health & safety, diversity, inclusion, cultural, societal, environmental and commercial matters, codes of practice and industry standards
B6	Apply an integrated or systems approach to the solution of broadly-defined problems
B7	Evaluate the environmental and societal impact of solutions to broadly-defined problems
B8	Identify and analyse ethical concerns and make reasoned ethical choices informed by professional codes of conduct
B9	Use a risk management process to identify, evaluate and mitigate risks (the effects of uncertainty) associated with a particular project or activity
B10	Adopt a holistic and proportionate approach to the mitigation of security risks
B13	Select and apply appropriate materials, equipment, engineering technologies and processes
B15	Apply knowledge of engineering management principles, commercial context, project management and relevant legal matters
B17	Communicate effectively with technical and non-technical audiences

FORMATIVE FEEDBACK OPPORTUNITIES

Your supervisor will give you the following formative feedback:
<ul style="list-style-type: none"> • Weekly, during project supervision meetings • Written feedback on Proposal (See Appendix A) • Written feedback on Progress Report (See Appendix B)

Oxford Brookes University
School of Engineering, Computing and Maths

hChat - Blockchain Messaging Application

By
Harrison Stoddart

March 2023

Acknowledgment

I would like to express my sincere gratitude to my dissertation advisor Hashem Dehghanniri whose help, inspiration and coherent guidance encouraged me throughout the overall fulfilment of the project.

I am very grateful, thank you.

Contents

Cover Page	1
Acknowledgment	4
List of Figures	8
List of Tables	9
Abstract.....	10
1 Introduction	11
1.1 Background	11
1.2 Motivation and Aim.....	13
1.3 Objectives	13
1.4 Project Overview	14
1.4.1 Scope	14
1.4.2 Audience.....	15
2 Background Review	16
2.1 Summary of Existing Approaches	16
2.1.1 WhatsApp – Web2.....	16
2.1.2 Blockchain	19
2.1.3 Status – Web3 Message Application	24
2.2 Brief Summary of Related Literature	26
2.2.1 Privacy Issues Relevant to Centralised Applications.....	26
2.2.2 Security Issues Relevant to Centralised Applications.....	27
2.2.3 Reliability Issues Relevant to Centralised Applications	28
2.2.4 Decentralised Blockchain Messaging Applications.....	28
2.2.5 Literature Conclusion.....	30
3 Methodology	31
3.1 Software Development Methodology	31
3.1.1 Agile Method.....	31
3.2 Requirements	32
3.2.1 Functional Requirements.....	32
3.2.2 Non-Functional Requirements	32
3.3 Specification	33
3.3.1 Specification for Functional Requirements	33
3.3.2 Specification for Non-Functional Requirements	33
3.4 Software Design	34
3.4.1 Overall System	34
3.4.2 Initial Draw.io UI Designs:.....	36
3.4.3 – Use Case Diagram	39
3.4.4 – Sequence Diagrams (Appendix A.2.1 – A2.5).....	39
3.4.5 – Flowchart	40

3.4.6 – hChat Network Architecture	40
3.4.7 Software Architecture using UML diagrams	42
3.5 Implementation	43
3.5.1 – User stories	43
3.5.2 Sprint 1 – Developing and testing a client server	44
3.5.3 Sprint 2 – Developing and testing inter-server communications and blockchain.....	45
3.5.3 Sprint 3 – Developing and testing logging in and registering functionalities via a SQL database	48
3.5.4 Sprint 4 – Developing and testing a main menu webpage	49
3.5.5 Sprint 5 - Developing and testing chatrooms.....	50
3.6 Version management	51
4 Results	52
4.1 Results	52
4.2 Issues encountered	53
4.3 Constraints and limitations	54
5 Professional Issues.....	56
5.1 Gantt Table	56
5.2 Gantt Chart.....	56
5.3 Risk Analysis	57
5.4 Professional Issues	57
5.4.1 Security Issues – GDPR Act.....	57
5.4.2 Legal – BCS Code of Conduct.....	58
5.4.3 Social – BCS Code of Conduct.....	58
5.4.4 Ethical	59
5.4.5 Environmental.....	59
5.4.6 Intellectual Property	59
6 Conclusion	60
6.1 What was achieved	60
6.2 Future work	61
7 References	62
8 Appendix.....	67
Appendix A – Software Modelling and Diagrams:	67
Appendix A.1 – Use Case Diagram	67
Appendix A.1.1 – Use Case Documentation:	68
Appendix A.2 – Sequence Diagrams.....	70
Appendix B – Professional Issues.....	75
Appendix B.1 – Gantt Chart.....	75
Appendix B.2 Risk Management	76
Appendix B.3 - Gantt Table	79

Appendix C – Third-Party Libraries Used:.....	80
Appendix C.1 – Node.js.....	80
Appendix C.2 – React.js	81
Appendix C.3 – Crypto-js.....	81
Appendix C.4 – Elliptic.....	81
Appendix C.5 – Socket.io-client.....	82
Appendix C.6 SQL Server	82
Appendix D – Software used to plan and develop dissertation	83
Appendix D.1 – Draw.io	83
Appendix D.2 – Visual Studio IDE	83
Appendix D.3 - GitHub.....	83
Appendix E - Testing	84
E.1 User Functional Requirements Testing	84
E.2 Sprint Plans and Testing	92
Appendix F	104
9 Glossary.....	105

List of Figures

Figure 1 - Comparison of Centralised and Decentralised Network Architecture.....	12
Figure 2 - An example of blockchain which consists of a continuous chain of blocks	19
Figure 3 – Overall Block Structure	20
Figure 4 - Demonstration of Digital Signatures used in Blockchain	21
Figure 5 - Initial Design of Login Page.....	36
Figure 6 - Initial Design of Registration Webpage	37
Figure 7 - Initial Design of Main Menu	38
Figure 8 - Initial Design of Chatroom	38
Figure 9 - Flowchart of Application	40
Figure 10 – Network Architecture of hChat with the potential of decentralisation.....	41
Figure 11 - UML Diagram of Overall System	42
Figure 12 - UML of Blockchain.....	42
Figure 13 - User SQL Database	43
Figure 15 - Use Case 1 – Sign up Sequence Diagram.....	70
Figure 16 - Use Case 2 - Login Sequence Diagram	71
Figure 17 - Use Case 3 – View Chatroom Sequence Diagram	72
Figure 18 - Use Case 4 – Send Messages Sequence Diagram	73
Figure 19 - Use Case 5 – Receive Messages Sequence Diagram.....	74
Figure 20 - Project Gantt Table.....	75
Figure 21 - Project Gantt Chart.....	79

List of Tables

Table 1 - Web3 Alternatives of Existing Web2 Application	13
--	----

Abstract

hChat - Blockchain Messaging Application

By

Harrison Stoddart

hChat is a messaging application that uses blockchain technology, which has its roots in cryptocurrencies and avoids dependencies on third parties. hChat uses the blockchain hChain which is stored on a single service node, which could be used in a distributed service node network, consequently making the need for a centralised server redundant. The agile method was chosen to carry out the project due to contributing reasons of better flexibility, quicker deliverable times, and iterative characteristics outweighed the benefits of other models. User stories for the application were created and spread out over 5 individual sprints, with an overall sprint backlog created as well as individual sprint plans. Testing concluded that a web interface had been successfully developed, allowing users to login and register an account. Logged-in users are displayed all application users and can choose any chatroom to join. Chatrooms show the conversation history between the parties and provide the application's main functionality, to send and receive messages in real-time utilising a blockchain. Overall, hChat has demonstrated that blockchain technology can be used outside the cryptocurrency realm, showing that it can provide integrity of user messages through an immutable blockchain, authentication through digital signatures and avoid manual intervention through smart contracts.

1 Introduction

1.1 Background

The primary messaging services available today include WhatsApp, Facebook Messenger, and Instagram, which have 2 billion, 931 million, and 2 billion monthly active worldwide users, respectively [1]. The one thing in common they all share is that they are all Web2 applications which utilize centralised networks. Centralised networks work by having a central authority, such as a central server or a group of servers owned and operated by the company of the application, controlling and managing all the resources and communications on the network.

Centralised networks create a dependency on the central authority to manage all communications, making a single point of failure; if the server fails, the entire system fails. The central authority is effectively in absolute control of the network, creating concerns that users have no control over who controls their data on the network and what actions the central authority performs, leading to concerns regarding transparency, security, privacy, and censorship.

"Data is the new oil" [2] outlines how commercial companies strive to have users' behavioural data to create insights into the customer. This vision of individual preferences is utilised in purposeful advertising to mine the maximum financial benefit of the data and can be sold to third parties. The fitting saying, "if you are not paying for the product, then you are the product", raises the question can you trust the owners of the central authorities not to redeem the vast potential financial gain from your data?

The last decade has seen major centralised messaging systems infamously fined over how transparent they are when handling and storing information and how conformed their attentions are to GDPR. For example, Meta, the owner of Facebook, Instagram, and WhatsApp, was fined €225 million in 2021 [3] for breaching privacy ordered by the Irish Data Protection Commission and a fine of €265 million in 2022 [4] for a large-scale data breach of the personal details of over 553 million users. The unfavourable history creates deep concern about how these companies cannot be trusted not to misuse and safely secure user data.

Due to the inadequate protection and privacy displayed in today's Web2 messaging applications, the project focuses on how new technology can be utilised to build a Web3 application that incorporates decentralisation and leverages blockchain technology to develop an application to provide privacy and trust in a trustless environment. Please refer to Figure 1 for a comparison of network architectures.

Blockchain technology provides a digital ledger to store user messages securely. Data stored on the blockchain is contained in blocks. These blocks are secured and linked to each other by a cryptography hash function, making it near impossible to alter data to create a trustless, immutable blockchain to store sensitive information. The ledger is stored, distributed, and maintained by an array of nodes on the network. The network control is distributed between nodes, which handle data trafficking, maintain network security, and perform consensus to validate data before its added to the ledger. Due to decentralisation, the system does not have a central point of failure to provide minimal downtime; the network layout provides higher protection against Denial of Service (DoS) attacks and the network is not governed by a central authority to make censorship impossible.

Smart contracts are created and executed on the blockchain to aid in automating processes to release the redundancy of manual intervention on the network. These factors contribute to the trustless network's efficiency and privacy and increase cost efficiency.

The messaging application hChat has been created to fix the present void for user privacy and protection when using popular concurrent centralised systems in messaging applications. hChat implements a network using a single node called a service node, which could be easily implemented into a multiple-node network, thereby decentralising the network, excluding the dependency on one central authority to route messages—decentralisation benefits from providing a trustless environment reducing points of weakness and offering direct connections to contacts without third-party interference.

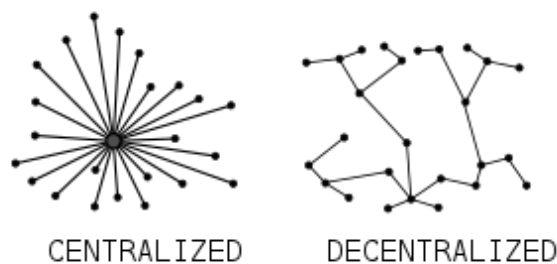


Figure 1 - Comparison of Centralised and Decentralised Network Architecture

Many Web3 applications are starting to replace well-known and established Web2 applications. Several organizations are moving away from centralised infrastructure and towards decentralised infrastructure to entice users with trustless applications that give users greater control over their data and interactions. See Table 1 for Web3 alternatives of Web2 Applications.

Table 1 - Web3 Alternatives of Existing Web2 Application

Web2	Web3
Google Chrome	Brave
Spotify	Audius
WhatsApp	Secretum
YouTube	Odysee
Google Search Engine	PSearch
Facebook	Sola

1.2 Motivation and Aim

The project aims to show how blockchain technology can be incorporated into a messaging application to securely hold conversations, automate processes using a similar form of smart contracts without third-party involvement and display its potential of being decentralised to create a true Web3 application in future work.

1.3 Objectives

- Ob1 – Research and explore current developments of messaging applications utilising decentralisation and blockchain technology.
- Ob2 – Establish a range of suitable layouts of blockchain networks concerning service nodes, light nodes, and web interface applications.
- Ob3 – Identify suitable blockchain configurations concerning how transactions and blocks will be constructed and mined.
- Ob4 – Research and explore existing web interface messaging application layouts.
- Ob5 – Evaluate a final overall model concerning the requirements needed.

1.4 Project Overview

1.4.1 Scope

The project will encompass four main tasks.

First Task

Creation of a blockchain adapted to be used in a messaging application to enable users to store their messages securely. The blockchain must be able to perform self-validation and use cryptographic links between blocks to achieve immutability.

Second Task

Creation of a service node which stores and provides a copy of the blockchain, listens for transactions and mines blocks.

Third Task

Design a server running as a light node on the network to enable connections from the client server and then onto the service node. The light node will be able to process users' requests sent from the client server.

Fourth Task

Develop a web interface application using a client server, which allows user to log in or register to allow them to join a chatroom to carry out their messaging.

1.4.2 Audience

Crypto Enthusiasts

Crypto enthusiasts have keen interests in blockchain technology and constantly seek ways to improve their privacy and security. Secure messaging platforms give people a safe channel to discuss cryptocurrency-related topics without concerns about possible monitoring.

Journalists, Whistle-blowers, and Activist Groups

The above requires a secure platform to exchange confidential information without the burden of being tracked/watched. hChat provides this as a secure blockchain messaging application.

Business Professionals

Professionals need private and secure communication methods to discuss sensitive topics like financial reports, marketing plans, and business negotiations. Applications for secure blockchain messaging give them a dependable and safe platform to transmit private information without concerns about being compromised.

2 Background Review

2.1 Summary of Existing Approaches

2.1.1 WhatsApp – Web2

WhatsApp is a free-to-use, worldwide, trustful, centralised instant messaging application owned by Meta. Users can send text and voice messages, make voice and video calls, and share documents, images, and user locations.

Creating an account requires users to input their phone number, which is used to identify users on the network uniquely. However, this creates several concerns and disadvantages for users:

- **Privacy:** Phone numbers are often linked to a person's real identity, making it easier for third parties to monitor their online activity.
- **Security:** Phone numbers are not 100% secure, as they can be easily stolen or spoofed, making it easier for malicious actors to access personal information or online accounts.
- **Limited Availability:** People living in developing countries cannot easily get access to a phone number, excluding them from being able to access the service.
- **Inflexibility:** Phone numbers are either fixed to a specific device or sim card, making it difficult for users to switch to a different device.

Additionally, users are asked for a display name, their region and date of birth, which is stored on their user database. With all this information from a user, WhatsApp can start to create meaningful models for each user, each saturated with vast potential possibilities for financially motivated targeted advertisement.

When sending messages, users construct and send them from the application running on their phone to WhatsApp's central cluster of servers. WhatsApp uses an open-sourced protocol called Singal Protocol [5] to provide end-to-end encryption of messages by implementing Double Ratchet Algorithm, prekeys, Triple Diffie Hellman, Curve25519, AES and HMAC_SHA256 [6], allowing only the sender and receiver, not even WhatsApp, to view the sent messages. However, due to WhatsApp not being open source, users may fear about data misuse and choose applications with an open-source nature to provide more confidence.

WhatsApp uses a modified version of the protocol XMPP [7] for messaging. XMPP is run on the user-end device and the server to open an SSL socket to the WhatsApp servers. The sent encrypted message is queued on the server until the recipient reconnects to the server to retrieve their message(s). Once the recipient has successfully retrieved their awaiting message(s), successful retrieval of a message is sent back to the WhatsApp server. The server then informs the sender that the message has been received, and subsequently, the queued message is deleted from the server memory. Messages can only be stored on the server for 30 days; if the receiver has not accepted the message within the timeframe, the message is automatically deleted.

WhatsApp's use of a centralized network offers several advantages over a decentralised network:

- **Easier to Maintain:** Having one central server is easier to keep updated than having to update the array of nodes of a decentralised network individually, this also contributes to less management time spent and less IT management required.
- **Faster Communication:** Messages can be delivered more quickly than decentralised networks, as messages do not need to be relayed between multiple nodes on the network. Furthermore, all user records are stored in one location, allowing recipient enquiries to be performed quickly.
- **Cheaper:** Centralised systems do not require as many resources as decentralised networks, which is highly beneficial for services with high user bases.
- **Easier to control and track data:** As all data must go through the central server, it enables data to be controlled, collected, and tracked across the network easily by the authority.
- **Consistently:** Interactions between clients and servers are standardised.

However, due to WhatsApp's network architecture, its advantages are also disadvantages. With data concentrated in one location, the data becomes susceptible to large-scale data breaches; if unauthorised access is gained to the database, all data is vulnerable. Ease of control and management of the network can lead to user censorship from the central authority, breaching users' freedom of speech. Furthermore, with the standardisation of interactions managed by a central server, conversations between users are not peer-to-peer; an intermediate is involved, which users are forced to trust, creating a trustful environment. The intermediate cannot access user-encrypted messages; however, because of the abundance of metadata detailing the time, recipient, and sender of the message in the data flow over these centralised servers, it is vulnerable to data mining, which is quickly figuring out how to harvest valuable personal information from this unsecured resource [8].

To conclude, a centralised network is a better fit due to WhatsApp's large user base and instant messaging users' expectations. However, recent fines regarding how conformed their attentions are to privacy raise the question.

Can WhatsApp be trusted to run a centralised network?

2.1.2 Blockchain

Blockchain technology provides an immutable digital ledger which records the expanding number of data assets, such as a conventional public ledger. Figure 2 displays an example of a blockchain. The blockchain holds a chain containing a continuous sequence of blocks, with each block pointing to the previous block, the parent block, by storing the previous block's hash as a reference. The first block on a blockchain is called a genesis block, which has no parent block.

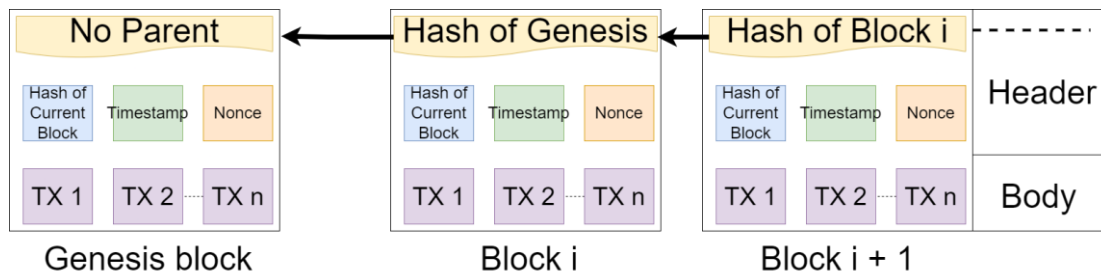


Figure 2 - An example of blockchain which consists of a continuous chain of blocks

Block

There is a wide range of blockchain networks available today, each using different varieties of blockchains that better suit their needs for the tasks they need to accomplish. Each block has a header and a body. What consists of a block's header and body depends on how the blockchain it is a part of has been developed. Please refer to Figure 3 to view a Bitcoin block's overall structure.

Contents of Bitcoin Block Header

Block version number – Indicates which set of block validation rules to follow.

Block's index – Indicates how many blocks have come before the block.

Previous/Parent block hash – a 256-bit hash value pointing to the previous block.

Current block's hash – a 256-bit hash value of the contents of the block.

Merkle tree root hash – hash value of all the stored transactions within the block.

Timestamp – The current timestamp (seconds since 1970-01-01T00:00 UTC).

nBits (Mining Difficulty) – A packed representation of the hexadecimal target threshold that the block's header hash must be less than or equal to. This number is changed to accommodate the number of miners on the network and their combined hashing power.

Nonce – An integer that starts at 0 and increments for every hash calculation.

Bitcoin Block Body

The body of a block consists of transactions and a transaction counter recording the number of transactions being held. This is utilized due to blocks having a maximum number of transactions that they can hold, which depends on the block size and the size of each block, which can be derived from the block version.

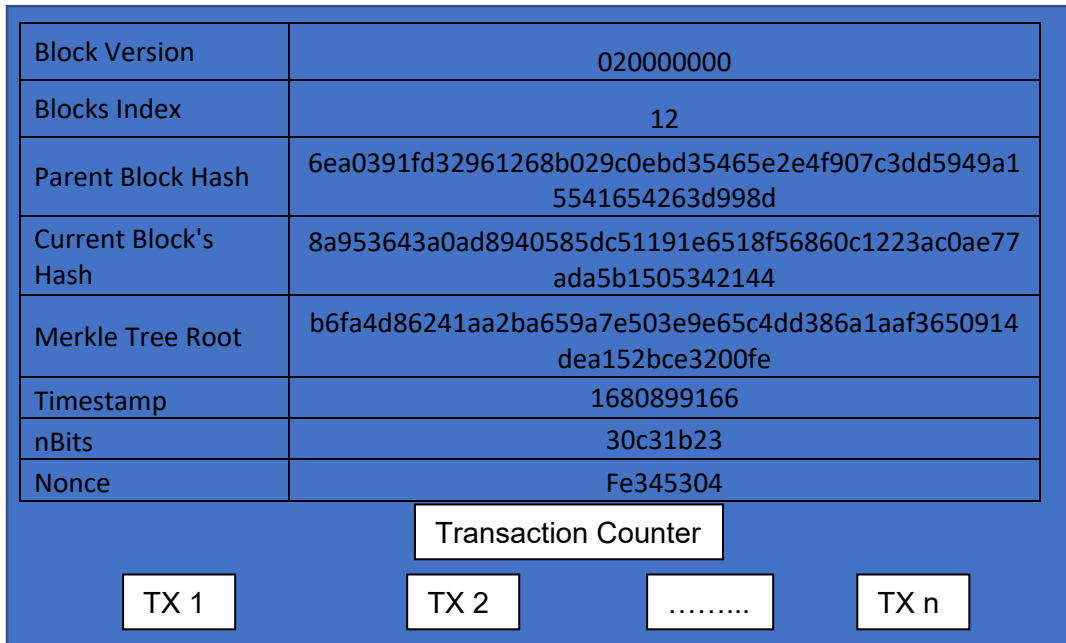


Figure 3 – Overall Block Structure

Genesis Block

The genesis block is the first block in a blockchain network, serving as the starting point and the foundation of the blockchain by establishing the rules and parameters through establishing a block version. Such rules may include the initial distributions of tokens, the mining and validation consensus algorithms, the maximum number of tokens that will exist and other network-specific parameters.

Digital Signature

Ownership of transactions is validated using asymmetric cryptography, where each user owns a pair of private and public keys. For example, Alice wants to sign her transaction digitally, so when Bob receives the transaction, he has a way of 100% knowing that Alice has sent the transaction. A digital signature involves a signing phase and a verification phase. Use Figure 4 as an example and for visual understanding.

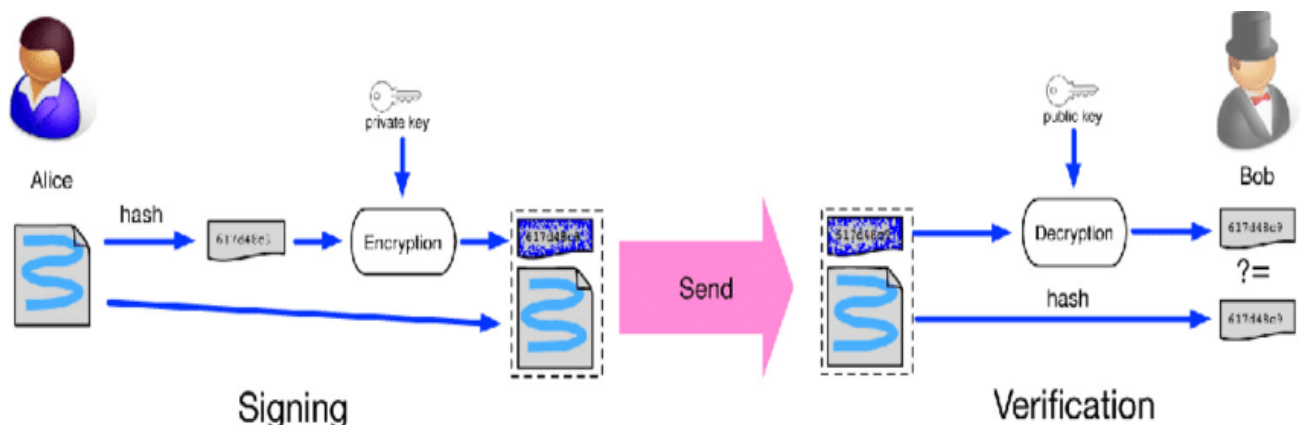


Figure 4 - Demonstration of Digital Signatures used in Blockchain

Alice first hashes the transaction and encrypts the output with her private key to create a digital signature. Alice then sends the transaction along with the digital signature to Bob. Bob verifies the transaction by first hashing the transaction. Bob then uses Alice's public key to decrypt the signature and compares it with the transaction's hash. If they match, then he knows Alice has sent the transaction as Alice is the only person that has access to her private key.

Examples of Consensus Algorithms in Blockchain Networks

Consensus algorithms secure the blockchain network using authorised nodes to verify the validity of transactions and blocks.

Proof of Work (PoW) – Miner Nodes

Implemented in the Bitcoin network, PoW is a consensus strategy that requires miner nodes to perform the mining of blocks. Mining works by calculating a hash value of a block header equal to or smaller than the target threshold by incrementing the value of the nonce to vary the hash value. Once a valid value has been obtained, the majority of the nodes (51%) on the network must mutually validate the value. Once validated, transactions within the block must be verified again by the majority, denoted by the block added to the blockchain. Due to calculations being time-consuming and costly due to the substantial required electricity consumption, when a miner successfully mines a block, they are given a small portion of a bitcoin which acts as an incentive. [9]

Proof of Stake (PoS) – Stake Nodes

PoS is an energy-saving alternative to PoW, which works by having validators on the network that are randomly selected to confirm transactions and validate block information. [10]

Smart Contracts

Blockchain adopts smart contracts to prevent the intervention of a third party. This allows transactions between two parties to be handled by an automated procedure or by a piece of code known as a smart contract which are executed on the blockchain.

Characteristics of Blockchain

- **Decentralised:** Blockchains can be incorporated into a decentralised network to avoid the need for a sole intermediate acting as a central point, which provides fault tolerance; if a node breaks down, data is accessible through other nodes. Decentralisation also avoids performance bottlenecks in central servers and implements a flat topology to provide a trustless environment where all nodes are equal and work together to govern the network, increasing censorship resistance.
- **Immutable:** As each block stores a reference of the previous block when modifying a transaction in a block, it subsequently changes the hash value and, due to each block being linked, causes all the following blocks to change, causing the blockchain to be invalid. This ensures that the information stored on the blockchain is tamper-proof and provides high trust and transparency.
- **Anonymity:** Users are assigned an unambiguous, uniquely generated address to interact and engage with a blockchain network.
- **Auditability:** Consensus algorithms can be implemented on a blockchain network. Each transaction is verified using consensus algorithms and timestamped, enabling them to be easily validated and verified by users. This provides traceability and transparency towards data stored on the blockchain.

Comparison of a Public, Private/Permissioned and Hybrid Blockchain

Public Blockchain: Allows anyone to join the blockchain network and become an authorised node. The user can access the blockchain and conduct mining to verify transactions.

Private/Permissioned Blockchain: Users must be invited and verified using the required information asked for to join the blockchain network. Due to the owner/operator having the right to delete and edit blocks, the network is controlled by a central authority.

Hybrid Blockchain: Combines elements of a private and public blockchain, where transactions recorded on the blockchain are not made public but can be verified and accessed by being granted access via a smart contract.

Other applications of Blockchain

Blockchain was initially developed to support cryptocurrencies. However, it has received much attention for its potential to completely alter other industries due to its immutability, verification and redundancy of an intermediate. These traits make it excellently fitted for storing and verifying sensitive information by implementing secure, efficient, and tamper-proof logs.

Healthcare: Personal data that is recorded and held can be securely encrypted and stored on the blockchain without privacy and integrity concerns. A distributed blockchain network can also be incorporated to tackle a common problem in the healthcare industry of data being isolated between medical devices.

Real Estate: Immediate verification of finances using consensus algorithms and with the aid of immutability on a blockchain can help expedite home sales, provide prevention from fraud, and allow transparency throughout the process.

Voting: Blockchain can be utilized to provide a distributed, immutable, transparent digital ledger to hold votes securely. Due to the distributed ledger throughout the network, officials can access the tamper-proof ledger easily to speed up the tally process. When a vote has been created, consensus algorithms verify votes by checking that the ID of the person that voted is not on the blockchain already.

2.1.3 Status – Web3 Message Application

Status is a private and secure Web3 messenger, which is free to use and available to be downloaded for both iOS and Android devices [11]. The open-source application entices users with levels of security and privacy that the Web2 messaging application cannot match.

The application is built on the public Ethereum blockchain, run by a network of decentralised nodes to enable the distribution of computation to avoid having a central authority in charge. Communication on the network is decentralised which avoids centralised choke points in which all users must try and connect to the cluster of central servers. The communication principles also create a trustless environment where users do not have to trust a central server governed by the central authority to route messages, avoiding the potential of their metadata being mined for targeted advertisement.

Signing up, the application requires no identifying information, such as a phone number, which can connect to a whole host of personal data. Instead, when users create an account, they are assigned an unambiguous Ethereum address, serving as their unique identity on the network, which is stored on the blockchain [12].

The Status network is made up of two types of nodes:

Relay Nodes (Waku Node) - Relay nodes are used in relaying messages between nodes to reach their destination using the Waku messaging protocol [13].

History Nodes (Mail server) - History nodes store historical messages in a SQL database and deliver them when queried, so when peers on the network receive a message but are not online, making them incapable of receiving the message, peers can request the message history of the conversation from the history node. Messages are stored for a maximum of 30 days before they are deleted off the node.

The user-centric design allows users to store their data locally or on a decentralised network of Status nodes, giving each user greater control over their data.

Messages sent through the Status application use the Whisper protocol incorporating end-to-end encryption using the Double Ratchet Algorithm [14]. The algorithm uses symmetric and asymmetric encryption and key exchange protocols between parties to derive new keys for every message sent using the key derivation function. This ensures that a compromised key cannot expose previous or future messages and only permits the sender and receiver to view the message's contents – not even Status can read the message [15].

Overall, status provides a trustless, transparent, decentralised network utilising secure messaging encryption techniques to give users a secure instant messaging application that offers high levels of privacy and security, with no interactions from third parties and no central authority, all whilst providing users with complete control over their data.

2.2 Brief Summary of Related Literature

2.2.1 Privacy Issues Relevant to Centralised Applications.

Issues Handling Data

The source “Sylo Protocol: Secure Group Messaging” [16] outlines the current privacy issues of the most popular worldwide social media apps due to centralised networks and how it has created a demand for decentralised social applications. The paper illustrates that the users are provided with a lack of transparency towards their data due to applications not being open source and using intermediaries controlled by authorities to route messages. Due to insufficient openness, tech companies potentially expose data to government agencies without acquiring consent or third-party entities for targeted advertisement campaigns [17]. The paper scrutinises the titan technology companies over how safely they store user data, how legal their intentions are and that users have the right to control their data—stating that a new generation of decentralised applications is required. The paper presents a decentralised system where user profile data is stored locally on user devices, avoiding having to trust external entities to store and process personal user data. Furthermore, smart contracts have been added to the development of the application, allowing agreements relevant to personal data to be automated without intermediate involvement.

Network traffic analysis

The peer-reviewed paper “A Decentralised Application for Secure Messaging in a Trustless Environment” [18] highlights how communications must be secure and anonymous. The paper explains that messages are secured via end-to-end encryption using asymmetric keys, such as WhatsApp [19]. The encrypted message is then sent to an intermediate, which routes the message to the recipient. However, when using protocols such as Kademlia [20], these intermediaries can gather unprotected metadata about the encrypted message, such as the sender and recipient, to perform analysis of the network’s traffic. Data mining techniques can be utilised on the network traffic for knowledge discovery on each user [21], to be shared with third parties or advertisers for financial gain, leading to serious privacy concerns in an otherwise secure system. To avoid the dependency on such intermediaries, the paper deploys a decentralised blockchain network messaging application to allow users to engage in secure and anonymous communication without needing a trustful intermediate.

Censorship issues with a trustful environment

The paper “Key Agreement for Decentralized Secure Group Messaging with Strong Security Guarantees” [22] discusses the trustful environment created when depending on a single central server for routing. When using centralised systems, the network is governed by the dictator, the central authority in charge of the server. When users sign up and agree to the terms and services of the application, users trust and hope that the authority will behave in their best interests to provide the services agreed upon. Due to the dictatorship, the central authority can block whoever they like on the network to enforce censorship and efficiently suppress specific initiatives and points of view from user accounts. The authority can also abuse the user’s right to free speech and have complete control over what material users can access [23]. The paper provides a decentralised messaging application to create a trustless system. The system mitigates the need for a dictatorship due to a decentralised network, enabling users to have complete control over publishing and accessing materials on the network.

2.2.2 Security Issues Relevant to Centralised Applications

Storing data in a central server

The report "The Devil is in the Metadata – New Privacy Challenges in Decentralised Online Social Networks" [24] defines centralised services such as Facebook as having design-inherent security flaws from having a single data aggregation point. With data in one central location, all valuable and sensitive information is accessible from one location, producing a security risk. If the perpetrator gains access to the database on the central server, then all the user records on the network become exposed to the data breach. The paper introduces a trustless, decentralised online social network that makes it the users' obligation to securely store their information locally, enabling the distribution of sensitive information over the network.

2.2.3 Reliability Issues Relevant to Centralised Applications

Having one single point of failure

In their paper, "Grasping the Concept of Decentralized Systems for Instant Messaging" [25], Gebhardt and Leinweber analyse the optimal digital infrastructure approach for Internet-based services like instant messaging. To aid in selecting a centralised or decentralised system, the report discusses the reliability of each method. Centralised systems consist of a single piece of technology on which the messaging service is solely dependent; this is known as a single point of failure. If this single point stops working, the entire service will also stop. Due to the obvious single point of failure, the system is involuntarily making itself vulnerable to Denial of Service (DoS) attacks, thus threatening its reliability. An explanation from the report deduces that a decentralised system can work as a patch to fix this vulnerability to distribute workloads around the network.

2.2.4 Decentralised Blockchain Messaging Applications

Decentralised Blockchain Architecture

Sparber's paper "Blockchain-based end-to-end encryption for Matrix instant messaging" [26] describes the motivation of preventing one single point of control to improve fault tolerance and reliability for instant messaging applications. Such applications use centralised systems which rely on a single server to carry out all communications on the network, which are prone to downtime due to being overwhelmed by large amounts of traffic. Sparber presents a decentralised network with a collection of independent nodes, each holding a digital ledger which records conversations on the network to ensure that network control is distributed around the network. If one node stops working, messages are rerouted to another node to add the message to the blockchain. This fixes the fault-tolerant and single-point-of-failure vulnerabilities evident within centralised networks to avoid potential downtime of the system.

The report “Secure Messaging Platform Based on Blockchain” [27] claims that businesses are not aware of the insufficient levels of encryption used within centralised applications used for instant messaging with other colleagues, highlighting that the interactive and instantaneous nature of the available applications has overshadowed their choice for business communications. A relevant example is Microsoft Teams which does not have end-to-end encryption to secure user communications. Motivated by the apparent prioritisation of the real-time nature of centralised applications, the report has developed an instant messaging application using blockchain technology to provide users with real-time communication, security, and privacy. Confidentiality and privacy are present by implementing a private permissioned blockchain to avoid entities gaining access to the blockchain to conclude the behavioural patterns of users on the network. In addition, security is maintained by encrypting user messages on the digital ledger using asymmetric and symmetric cryptography, enabling only the sender and receiver to view the message sent and stored on the blockchain. The blockchain also incorporates digital signatures so that transactions can be authenticated to ensure they have been sent by the intended sender and have not been tampered with in transit.

2.2.5 Literature Conclusion

For the past few decades, the supporting architecture of the most popular worldwide messaging systems has been centralised with a central authority effectively in charge of the whole network. Centralisation benefits from central authority controlling the entire system to ensure the network runs smoothly, ease of maintenance and data synchronisation because all messages are synced with a central server.

These advantages and the fact that Web3 is still evolving and relatively new make and explain why Web2 is an attractive choice when creating a messaging system. However, it is also important to know that these advantages gained from the network architecture also offer gaps relevant towards user security and privacy, which the project aims to amend.

More in-depth Aims and Objectives collected from the research

A hybrid blockchain will be created, allowing anyone to join the network after identity verification. A web interface will be designed to allow users to register and sign into an account to access the network, enabling them to communicate via chatrooms on the network. Users will only be able to view the history of their own conversations and no other conversations.

Blockchain technology will provide a digital ledger that stores the history of all messages recorded on the network. Messages will be held in transactions in blocks on the blockchain. Blocks store a hash value of the block and the previous block's hash to create cryptographic links to provide immutability on the blockchain. Authentication of transactions is provided through digital signatures, and access of messages are granted through similar forms of smart contracts that are executed on the blockchain to provide automation.

The project also aims to show how hChat can be incorporated into a decentralised network which provides a trustless environment with no central authority in control, avoiding a dependency on a governed central server to route messages. This makes censorship impossible, avoids potential user message metadata being mined and prevents a single point of failure on the network, which is susceptible to denial-of-service attacks.

3 Methodology

3.1 Software Development Methodology

3.1.1 Agile Method

When choosing a methodology, the decision was to use an agile method throughout the project's development. However, when researching the pros and cons of methods, certain factors needed to be more relevant due to the team size being 1 throughout. Contributing reasons of better flexibility, quicker deliverable times, and iterative characteristics outweighed the benefits of a waterfall model incapable of dealing with minor changes throughout the development.

Agile software development is a methodology that achieves customer satisfaction and flexibility and enhances collaboration. The method works by breaking down the overall development process into smaller, more achievable goals using an iterative process, enabling incremental development. The model strives to respond effectively and quickly to the requirements and conditions demanded throughout the process, with frequent releases and continuous feedback being managed by the model's adaptability. In this approach, the method ensures precision and quality, and the User Stories are continually understood throughout the development process, with numerous chances for end-user opinions to be considered before, during and after each sprint.

The largest drawback of an agile model is that it can be a struggle to manage. Failing to do so can cause confusion on progress and budgets, leading to missed deadlines and cost overruns. Compared to other models, such as Waterfall, which has a linear and more organised approach, agile is more flexible and continually changing, demanding better and more attentive management.

3.2 Requirements

3.2.1 Functional Requirements

Functional Requirement 1: Users should be able to register an account.

Functional Requirement 2: Users should be able to authenticate their identity through a login system.

Functional Requirement 3: Users should be able to communicate with any other registered user on the network.

Functional Requirement 4: Users should be able to send messages to the blockchain to be stored.

Functional Requirement 5: Users should see their conversation history when entering a chatroom.

Functional Requirement 6: Users should be able to move between chatrooms.

3.2.2 Non-Functional Requirements

Non-Functional Requirement 1: The system should be able to always provide high availability of the service towards users, with minimal downtime.

Non-Functional Requirement 2: The system should be able to ensure the integrity of user messages stored in the blockchain.

Non-Functional Requirement 3: The system should be able to authenticate user messages.

Non-Functional Requirement 4: The system should be able to provide an intuitive interface enabling users to access all features easily.

Non-Functional Requirement 5: The system should be able to facilitate compatibility to enable users to use the application on mobile devices, tablets, and computers.

Non-Functional Requirement 6: The system should be able to handle and process all services without a third-party intervention.

3.3 Specification

3.3.1 Specification for Functional Requirements

Functional Requirements 1 & 2: A Client-server will be hosted by WebSocket, which talks to the SQL server on the network. The MySQL server manages and creates user accounts, ensuring unique names are used when registering an account and credentials are valid when logging in.

Functional Requirement 3: Once a user has logged in, the client-server will request the list of users stored in the SQL database and display this information for their selection.

Functional Requirement 4: The user's text is passed to the client-server. The client-server forms this into a text message and passes it to the light node; from there, it is formed into a transaction and passed to the service node to be mined and added to the blockchain.

Functional Requirement 5: When the user enters the chatroom, a request is passed up to the service node for the chatroom history.

Functional Requirement 6: When in a chatroom, users can use the cancel_chat button to return to the main menu and from there, can enter another chatroom.

3.3.2 Specification for Non-Functional Requirements

Non-Functional Requirement 1: Mining has been implemented on the service node to control the flow of blocks added to the blockchain, and therefore, it cannot be swamped to achieve a consistent service availability.

Non-Functional Requirement 2: Cryptographic links between blocks on the blockchain prevent potential data modifications.

Non-Functional Requirement 3: Digital signatures will be implemented using private and public keys to create and verify signatures.

Non-Functional Requirement 4: Design of webpages were based on well-established existing chat programs researched in my literature review.

Non-Functional Requirement 5: The webpages are designed to locate in the centre of the screen and therefore are suitable for mobile devices, tablets, or computers.

Non-Functional Requirement 6: All services are handled by the servers, and no external resources are required.

3.4 Software Design

The software design phase lays the foundations for the entire project, making it a vital stage for software development. Blueprints of how the architecture, structure, and interfaces define how the whole system will be developed. To carry out the project's foundations, requirements and objectives must be clarified to ensure the system implements the correct functionalities. These implementations can be carried out with minimal risk by creating software designs to identify any issues early to prevent any possibilities of project failure or delays in the future.

3.4.1 Overall System

The overall system can be split into four different parts, listed below, which will interact with one another via TCP connections to full fill the blockchain messaging application.

Web Interface using a client-server

The web interface will include a login, registration, main menu (user list) and chatroom webpages.

Login webpage - Enables users to login

Registration webpage – Enables users to register

Main menu webpage – Lists all registered users and allow the user to select who they wish to chat to.

Chatroom – Enables users to view conversations and send and receive messages.

Light Node

This will be designed as the gateway between the client-server and the service node. It will be able to process requests and messages from the client-server and action them. This will be achieved in the case of login and registration by accessing the SQL database to confirm the authenticity and in the case of text messages to process them into a transaction and forward it to the service node. This solicits a response from the service node and a similar form of a smart contract will be applied to distribute messages to the relevant users.

Service Node

The service node will hold the blockchain. The service node will listen out for any transactions from the light node and add it to a transaction pool. This pool will be integrated by a mining service and processed, resulting in a block, holding the transaction, being added to the blockchain, and then sent back to the light node.

SQL Database

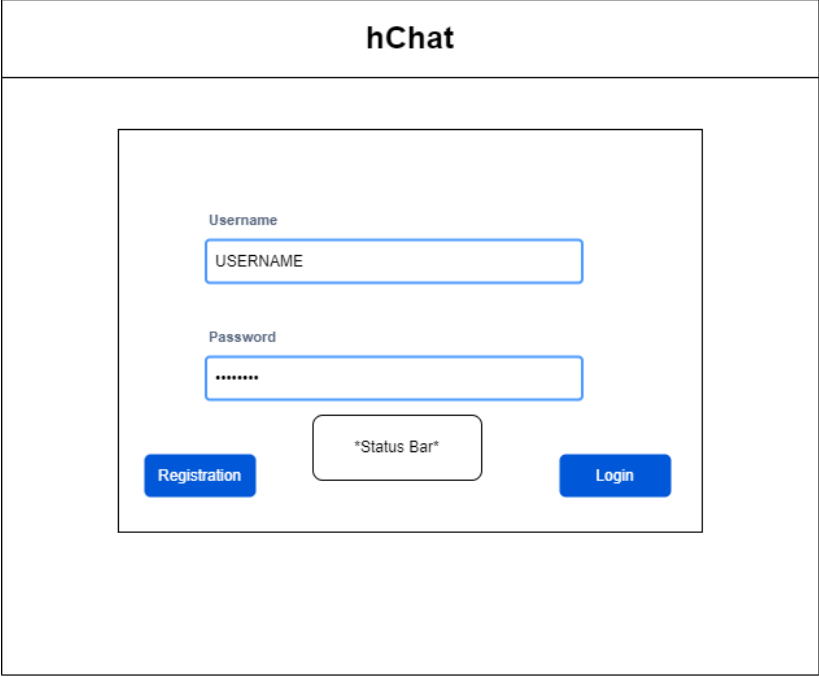
A SQL server will run on the network, which will store a user database. The light node will connect to the SQL database and request user information as required.

3.4.2 Initial Draw.io UI Designs:

The research into creating intuitive and straightforward designs was incorporated into the development plan. Designing before developing enabled me to focus on user needs and identify possible issues early; therefore, during the development phase, nothing needed to be changed, allowing the overall development speed of the UI to be efficient.

Login webpage

Figure 5 shows that users can log in by entering their credentials and pressing login. Or create a new account on the application by pressing the registration button, which directs them to the registration web page. When an unsuccessful login has been attempted, the status bar informs them of the invalid login attempt.

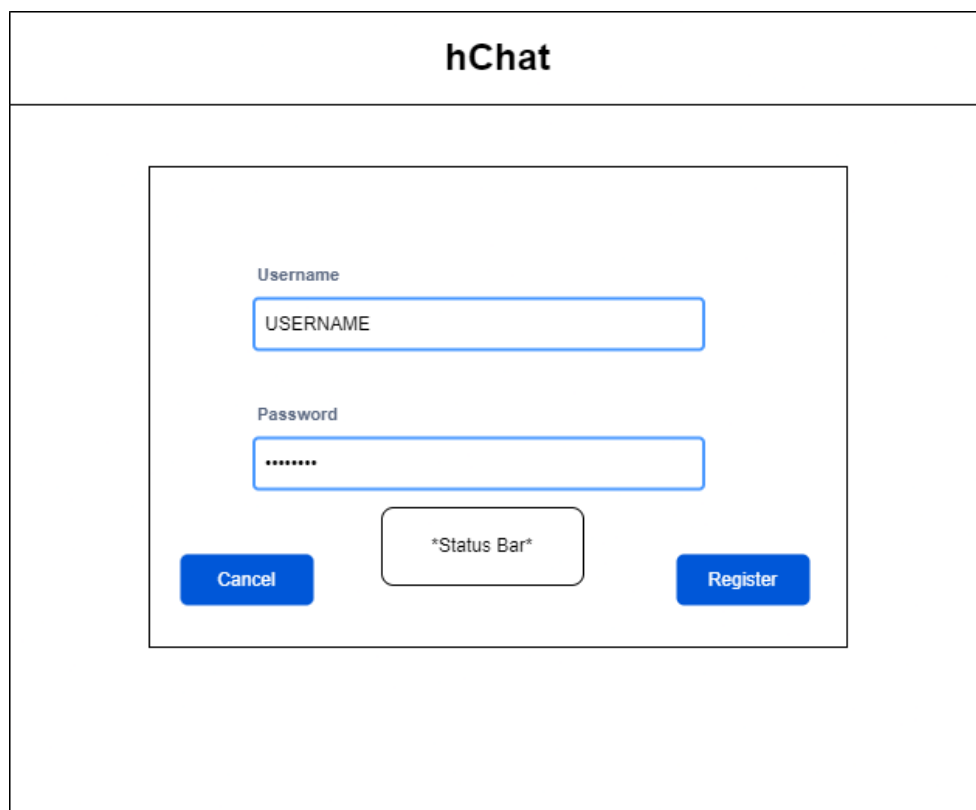


The image shows a wireframe for the 'hChat' login page. At the top, there is a header bar with the text 'hChat'. Below this, the main content area contains a login form. The form has two input fields: 'Username' with the placeholder text 'USERNAME' and 'Password' with placeholder dots. Below the password field is a 'Status Bar' placeholder. At the bottom of the form, there are two buttons: 'Registration' on the left and 'Login' on the right.

Figure 5 - Initial Design of Login Page

[Register webpage](#)

Figure 6 shows that users can enter their credentials and press register to make an account for the application. On unsuccessful registrations, the status bar informs the user of why the registration failed and when a successful registration occurs, users are directed to the login page. A cancel button allows users to return to the login page.

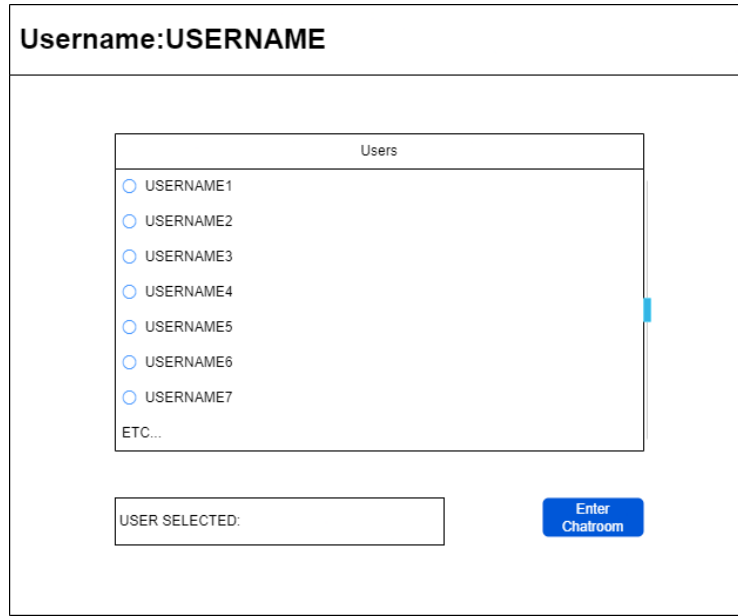


The image shows a web form titled "hChat" for user registration. It features two input fields: "Username" with the placeholder text "USERNAME" and "Password" with masked characters "*****". Below the password field is a rectangular box labeled "*Status Bar*". At the bottom of the form are two blue buttons: "Cancel" on the left and "Register" on the right.

Figure 6 - Initial Design of Registration Webpage

Main Menu Webpage

Figure 7 demonstrates how all the users on the network are displayed, with users allowed to select a name with a radio button. The chosen username is shown in the USER SELECTED box, with the user prompted with the Enter Chatroom button. Users are provided with a scroll bar to view all users and can enter a chatroom by selecting a usernames radio button and pressing the enter chatroom button.

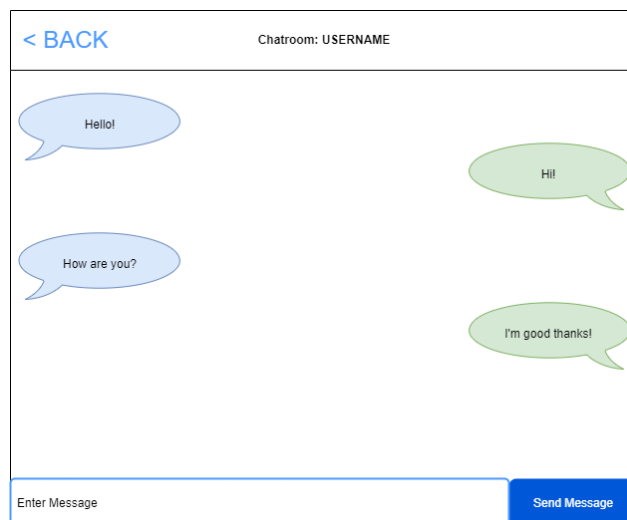


The diagram shows a web interface for the Main Menu. At the top, there is a header area labeled "Username: USERNAME". Below this, a large container holds a scrollable list of users. The list is titled "Users" and contains radio buttons next to the following usernames: USERNAME1, USERNAME2, USERNAME3, USERNAME4, USERNAME5, USERNAME6, USERNAME7, and ETC... A vertical scrollbar is visible on the right side of the list. Below the list, there is a text input field labeled "USER SELECTED:" and a blue button labeled "Enter Chatroom".

Figure 7 - Initial Design of Main Menu

Chatroom Webpage

Figure 8 presents how users can scroll through the conversation and are prompted with a text box to initiate conversations. The selected user's name is displayed at the top, along with the user's sent messages on the right in green and the received messages on the left in blue. Users are given a back button to go back to the list of users so that they enter other chatrooms.



The diagram shows a web interface for a Chatroom. At the top, there is a header area with a blue "< BACK" button on the left and the text "Chatroom: USERNAME" on the right. Below the header, the chat area displays a conversation. Received messages are shown in blue speech bubbles on the left, with the text "Hello!" and "How are you?". Sent messages are shown in green speech bubbles on the right, with the text "Hi!" and "I'm good thanks!". At the bottom, there is a text input field labeled "Enter Message" and a blue button labeled "Send Message".

Figure 8 - Initial Design of Chatroom

[3.4.3 – Use Case Diagram](#)

Use case diagrams provided a clear and complete understanding between the interactions of the system and its users for all the different situations in which the system would be utilised. This documentation ensured that the system functionalities would all be defined to meet the users' needs and be ready for implementation.

Refer to **Appendix A.1** to view the use case diagram illustrating the listed use cases below.

Refer to **Appendix A.1.1** that describes the listed use cases.

1. Sign up
2. Login
3. View Chatroom
4. Send messages
5. Receive messages

[3.4.4 – Sequence Diagrams \(Appendix A.2.1 – A2.5\)](#)

Sequence diagrams provided a visual representation establishing the behaviours needed within the system as well as identifying and addressing potential problems before implementation.

Refer to **Appendix A.2** to the view sequence diagrams for all of the listed use cases below.

1. Sign up
2. Login
3. View Chatroom
4. Send messages
5. Receive messages

3.4.5 – Flowchart

Figure 9 helped me understand each possibility a user could perform on the system and account for them accordingly, with the addition of informative textual aids to help the flow of the experience for each user.

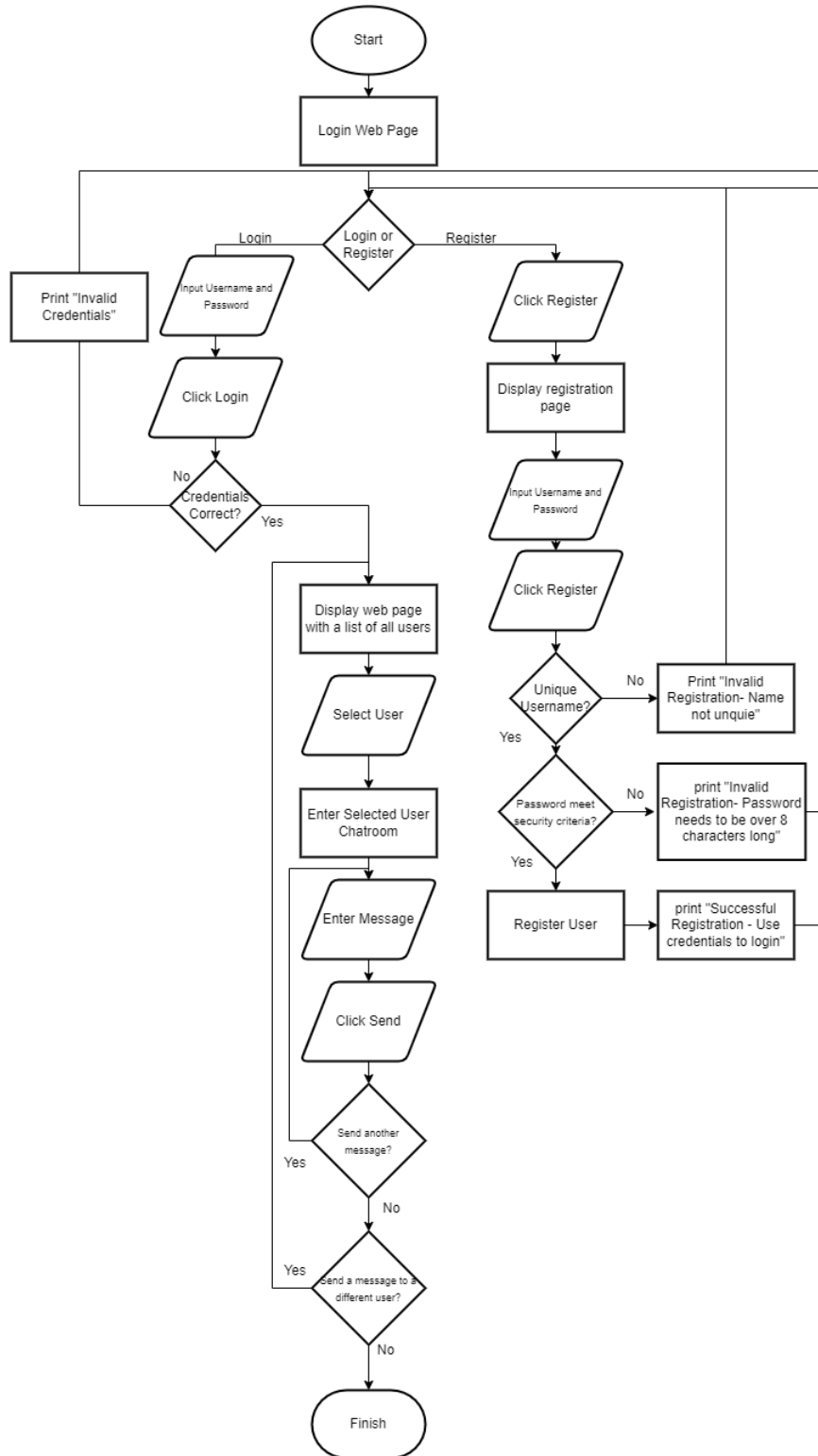


Figure 9 - Flowchart of Application

3.4.6 – hChat Network Architecture

Potential hChat decentralised network

The hChat network is potentially part of a decentralised network, where all service nodes are interlinked, and all users can access any client server off any node. The SQL database is a local resource and potentially could be linked to other SQL databases.

hChat Network

hChat is designed so that different users and devices can connect to a client server for its web pages such as login, registration, main menu, and chatrooms. The network will be centralised but has the potential to be part of a decentralised network.

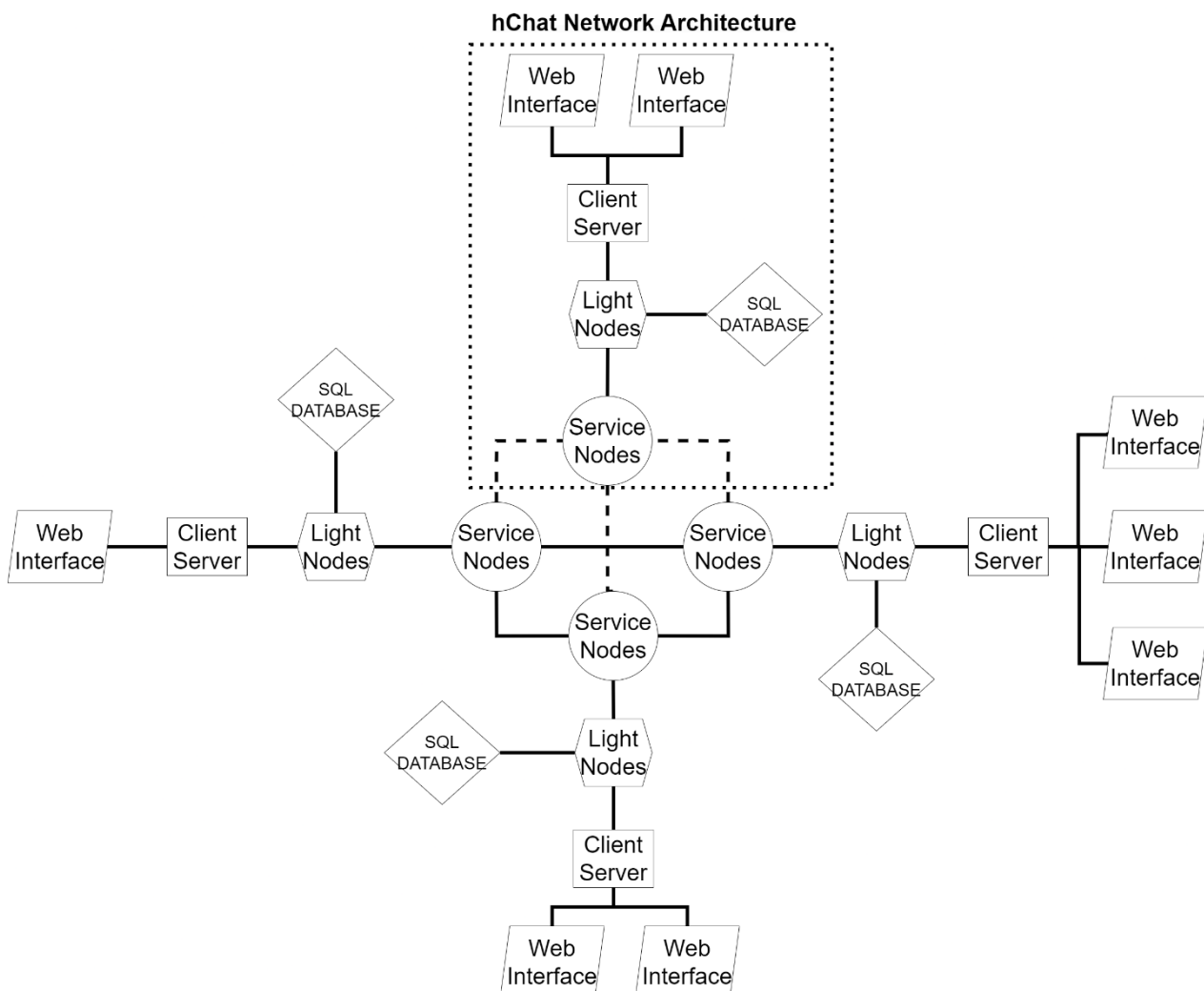


Figure 10 – Network Architecture of hChat with the potential of decentralisation

3.4.7 Software Architecture using UML diagrams

Overall System

Figure 11 breaks down the nodes on the network, detailing the specific variables and methods for each node's configuration to be able to run and communicate on the network.

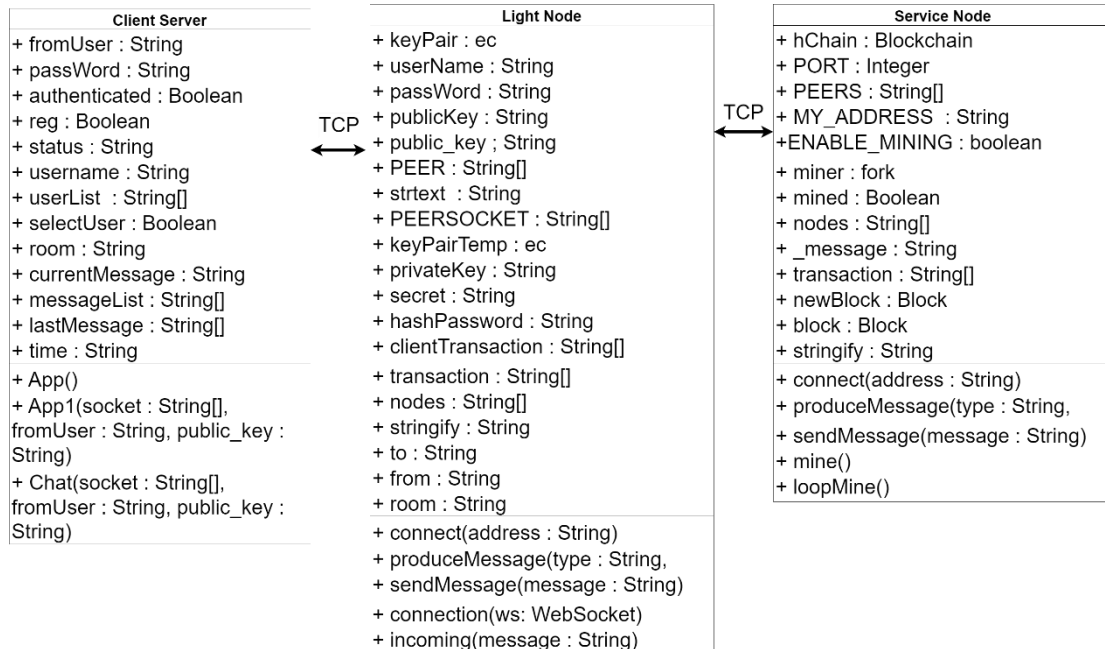


Figure 11 - UML Diagram of Overall System

Blockchain

Figure 12 displays the variables and methods needed to implement an immutable blockchain for a messaging application that can be validated. The blockchain stores a chain which consists of an array of blocks, each containing only one transaction for simplicity. The blockchain also holds a transaction pool for transactions pending to be mined, `blockTime` indicating the desired time interval of blocks added to the chain and the mining difficulty.

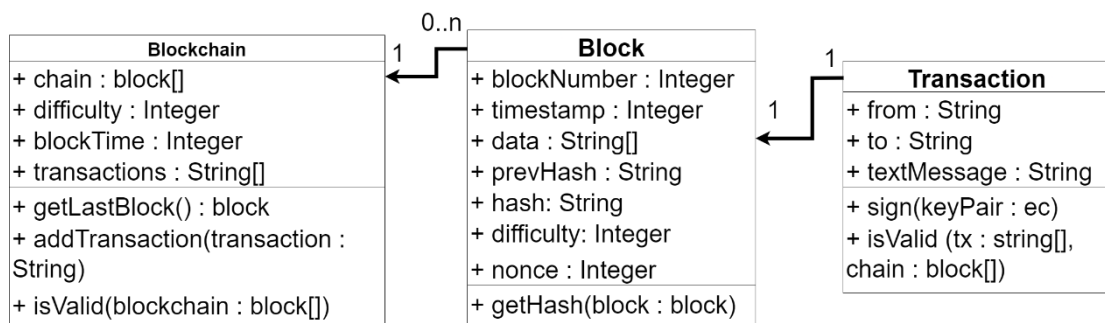


Figure 12 - UML of Blockchain

[SQL Database for User Credentials](#)

Figure 13 shows a record in the user credentials database. Passwords will be hashed before stored as a security measure.

<u>Username</u>	<u>HashedPassword</u>	<u>PublicKey</u>	<u>PrivateSeed</u>

Figure 13 - User SQL Database

[3.5 Implementation](#)

The implementation of the project was done use Node.js, React.js, third party libraries (**refer to Appendix C**) and other software (**refer to Appendix D**)

For the sprint backlog please refer to **Appendix E.2.1**

For each sprints plan and testing please refer to **Appendix E.2.2-6**

[3.5.1 – User stories](#)

1. As a user, I want the application to run on a web interface so that I can access the application via a phone or computer.
2. As a user, I want to communicate on a network that uses blockchain technology so that my messages are stored and are retrievable on an immutable blockchain.
 - a. As a user, I want smart contracts to automate the process of handling messages.
 - b. As a user, I want blocks to be mined before added on the blockchain.
 - c. As a user, I want a digital ledger to hold all my messages sent on the network.
3. As a user, I want to be able to have an account so that I can be uniquely identified on the network.
 - a. As a user, I want to be able to sign into an account.
 - b. As a user, I want to be able to register an account.
4. As a user, I want to be able see a list of all users of the application so that I can start 1:1 chatting with anyone of them.
5. As a user, I want to be able to chat with registered users on the application so that I can talk to my friends.
 - a. As a user, I want to be able to enter chatrooms with registered users.
 - b. As a user, I want to be able to send messages in the chatroom.
 - c. As a user, I want to be able to receive messages in the chatroom.
 - d. As a user, I want to be able to see the history of messages that have been sent between the two parties in the chatroom.

3.5.2 Sprint 1 – Developing and testing a client server

The main objective of this sprint was to have a client-server with bidirectional communication to and from the web interface and the light node.

This sprint aims to enable users to connect and communicate back and forth to the light node, which sends requests and receives responses to and from the SQL database for logging in and registration and the service node for chatting on the web interface.

Background research pointed me towards Socket.io-client for communication to the light node and React to run the client-server.

Client-Server

Client-server to Web interface

React is used because of its ability to display user-defined variables on webpages using JavaScript embedded in the HTML pages. This is used for displaying logging and registration status, user list and chat messages. This is achieved using React's useStates which allows a variable to be defined and used in an HTML form.

Client-server to Light node

Socket.io-client is used to send requests to the light node; these consist of authentication and registration requests. The response from these requests is handled with React's useEffect, which is used to acknowledge login or registration requests. Within the acknowledgement of the login, it has the user's public key.

Testing of Sprint 1

From the outset, it was realised that using React was beneficial but had the limitation of not being able to interact with a SQL database. To overcome this limitation, a decision was taken to insert a light node between the client-server and the service node. Testing consisted of accessing the client-server and checking to see the response of a hello message to ensure possible connectivity between the web browser and the client-server.

3.5.3 Sprint 2 – Developing and testing inter-server communications and blockchain

Within Sprint 2, the objective was to create a light node to handle and process requests from the client-server. Such requests as authentication, registration, user lists and chat messages. A service node is then required to receive each chat message from the light node, mine it and then store it on the blockchain, with the block being sent back to the light node.

In this sprint, it was decided that it was best to implement the bespoke blockchain created in the design stage that would be accustomed to the nature of the messaging application. Time was spent researching already developed blockchains to create a blockchain called hChain that would store user messages and be able to be retrieved.

Further research revealed that each server's interface code would be node.js. Node.js is a server-side JavaScript runtime environment well-suited for developing web applications due to its high performance and scalable capabilities. Furthermore, research revealed that Bcrypt, a third-party library, would be helpful to hash passwords, and EC was chosen for its powers of generating key pairs (private and public). Finally, WS WebSocket was selected for communication between the light node and the service node because it was intended to run a WS WebSocket server on the service node, as no HTTP was involved.

User Story 2a - Light Node

Light node to Client Server

Express is used to create a server to communicate with the client-server. The light node server processes logins and registrations using a SQL database. The client-server constructs a textMessage from the user input and forwards it to the light node. The light node receives the textMessage, creates a digitally signed transaction, and relays it onto the service node.

Light node to SQL database

Logging In: A username and password are received from the client-server. The light node requests a user record from the SQL database, and if found, it hashes the received password using Bcrypt and compares it with the password in the user record. If they match, then respond to the client-server with an authenticated true if not, authenticated false.

Registration: When the light node receives the new username and password, it first checks with the SQL database that the username is unique. From here, a temporary key pair is generated. The private key is then hashed to create a secret (seed) to generate a reconstructable key pair. We now have a public key and a secret (seed) which can be used in other areas to reconstruct a key pair. The username, hashed password, public key, and secret (seed) are stored on the database.

User list (main menu): When the light node receives “getUsers”, it queries the SQL database for all usernames and forwards this list to the client-server.

Light node to Service node

The light node communicates using WS WebSocket to the service node. Once constructed a transaction, it is then passed to the service node. The response from the service node indicates that a block containing the transaction has been mined and put onto the blockchain. This initiates a similar form of a smart contract where the message part of the transaction is forwarded to the two relevant users involved in the conversation. This is achieved by using the Socket.io room function, where a socket connection is put into a room with another socket connection; thereby, both receive any messages sent to that room. To achieve this end, each user pair is compared numerically and stored into a single string, and then the string is used as the room's name. This ensures that no matter what combination of username pairs is used, there will always result in the same room name. i.e., John to Fred results in a room called JohnFred, and conversely, Fred to John will also result in a room called JohnFred due to $\text{John}=4\text{A}6\text{F}686\text{E} > \text{Fred}=46726564$.

User Story 2b - Service Node

WS WebSocket server is used to create a server to handle and process transactions, block requests and mining. When receiving a transaction from a light node, the transaction is added to the pool of transactions. The service node constantly checks for a pending transaction in the pool. It uses multithreading to conduct the mining process and checks the transaction pool simultaneously to keep services available for users. For every 100th block, the mining processing time is calculated and compared against the set blockTime on the blockchain. If the mining takes too long, the difficulty is reduced by one or increased by one if the mining is being processed too quickly. This maintains a fixed interval between the addition of blocks to the blockchain.

Service node to Light node

After the transaction has been mined, it is placed on the blockchain and forwarded back to the light node.

Blockchain

A blockchain is created with a Genesis block as the starting point. This is a block with an index of 0, holding no transactions, a default difficulty of 1 and a hash value of the complete block, which will be incorporated in the next block.

User Story 2c - Mining

Mining works by taking the transaction, placing it within a block and processing a hash value according to the blockchain difficulty.

Testing of Sprint 2

Testing consisted of verifying the connection between the client-server and the light node as well as the response of the light node. From here, I tested the connection between the light and service nodes. Overall, this test ensured connectivity between servers.

Further testing ensured the blockchain containing the genesis block was created when the service node started.

3.5.3 Sprint 3 – Developing and testing logging in and registering functionalities via a SQL database

This sprint aimed to create a SQL database that securely holds user information. This allows users to login or register on the system via a web server.

This sprint aimed to provide users with unique access to the application.

Database

A SQL database called “nodelogin” was created to store all the user credentials for the application and is stored on a SQL Server that runs on the network.

The SQL record is comprised such as in the design stage with each user record consisting of:

Id number – This is a primary key

Username varchar(50) –The user’s username, the field is unique to them and can only be a maximum of 50 characters.

Password varchar(255) – The user’s password which is hashed using Bcrypt and can be only a maximum of 255 characters.

Public_key varchar(512) – The user’s public key which is used to uniquely assign them on the network.

Private_seed varchar(255) – The user’s private seed which is used to create a key pair.

Upon successful registration, the associated username and hashed password entered are stored, and a public key and a private seed are generated for the user and stored.

Login Webpage

This webpage was implemented as stated in the design stage. The user inputs a username and password, which is forwarded to the client-server. The response of this is then displayed to the user.

[Registration Webpage](#)

This webpage was implemented as stated in the design stage. Users input their preferred username and password, which is forwarded to the client-server. The response of this is displayed to the user.

[Testing of Sprint 3](#)

Registration testing consisted of trying combinations of usernames and passwords to ensure the username's uniqueness and the password's length; this also tested the SQL database. This ensures that each user record is uniquely identifiable and protects the user's account from unauthorised access.

[3.5.4 Sprint 4 – Developing and testing a main menu webpage](#)

Within sprint 4, the objective was to create a main menu web page displayed once a user has successfully logged in. The webpage was implemented as stated in the design.

After a user has selected who they want to have a conversation with, this solicits the corresponding chatroom webpage to be displayed.

[Testing of Sprint 4](#)

Testing consisted of checking the main menu webpage to ensure that all registered users on the SQL database were displayed and that any user could be selected to converse with. This test aims to allow users to have a 1:1 chat with any other registered user.

3.5.5 Sprint 5 - Developing and testing chatrooms

This sprint aimed to implement a 1:1 conversation between users using individual chatrooms. Chatroom webpages were implemented as stated in the design stage.

This aims to ensure that each user only sees their conversation with their selected party.

Testing of Sprint 5

The test focuses on first ensuring that when a user enters a chatroom, the conversation history between the two parties is displayed. The purpose of this was to ensure that users were capable of viewing conversation histories between parties.

Secondly, any messages sent out would appear on the right-hand side of the chatroom; this is after the message has been added to the blockchain and forwarded back to the user. Also, a check was made on the receiving party's chatroom to ensure that the received message was displayed on the left-hand side. This test ensures that messages can be transferred up and down the line to and from the blockchain to each user.

This is the final testing of end-to-end connectivity on the network for chatting capabilities.

3.6 Version management

When conducting the risk assessment, worries about how I could access the code if I was using my laptop, my desktop at home or a desktop at university came to light. Other worrying contributions include if I wanted to work on different devices, I would have to use a USB stick to hold the latest code copy. However, faulty USB sticks are notorious, and with the fact that they could be misplaced or stolen, I needed a better way of source control.

I chose to use a centralised code repository called GitHub (**Appendix D.3**) which provides a centralised location to store all code for the application enabling the ease of managing multiple versions. This allowed me to easily access the code from any device I wished to work on without holding copies of the code on USB sticks and worrying about losing it due to hardware malfunctions, as the code was backed up on the cloud.

Version control allowed me to easily keep track of the changes to the code over the whole development phase, with the ability to add informative comments to each version to list the new updates. This particularly helped me when cases the implementation of new functionalities stopped other functionalities from working, allowing me to go back to previous versions of the code and see why it was previously working.

4 Results

4.1 Results

Testing concluded a successful blockchain messaging application, but it was noted that due to the mining operation of the blocks, a delay was encountered in displaying messages in the chatroom. This was because the mining algorithm is designed to limit the number of blocks added over a fixed period by increasing or decreasing the mining difficulty.

Initial testing proved that the initial design of the webpages would not be suitable for mobile devices and, therefore, had to be redesigned to ensure user ability on all devices.

Furthermore, it was discovered that a user could connect to himself, which is redundant.

Apart from this, testing was successful and led to an application that incorporated all functional and non-functional requirements. (Please refer to **Appendix E.1**)

4.2 Issues encountered

As React was chosen to run the client-server, it meant the client-server could not talk to a SQL database. This issue was solved by implementing a light node that could communicate between the database to verify registrations and logins.

Further testing found that moving from the chatroom to the main menu had side effects that were unable to rectify, which led to a workaround in that when a user escapes from a chatroom, he is redirected to the login page instead of the main menu.

The biggest issue encountered was the ability to store a keypair because I used the function `keypair.signature`, and if you store keypair as a string or an object on a database when you restore it, you only have a string or an object and not a function. To get around this, a seed is stored; from this, the same keypair is generated each time it is needed.

For remote networking, fixed addressing had to be implemented so that remote web users could access the client-server. An IP address had to be assigned to the light node, and the client-server had to be pointed to this address. This was somewhat simplified furthermore by using DDNS entries and ensuring DDNS was implemented on the hosting PC.

Finally, when a user requests the conversation history of the chatroom, this is downloaded to him. The side effect is that when the other party joins the chatroom and does the same thing, another copy of the history is downloaded to the chatroom, causing the first party to have two copies of the history. This was resolved by the client-server testing to see if any messages have already been displayed in the chatroom, and if so, discard them. As each message is timestamped, any duplication will be picked up.

4.3 Constraints and limitations

The main limitation throughout the project was time. Other modules on the Cyber Security course required my full attention and needed to be prioritized at times over the development of the project to ensure that I could achieve a result from my degree that I'm proud of. My retail job over the Christmas holiday, the busiest time for retail, also had to be prioritized due to my self-funded university experience. Furthermore, with colleagues calling off sick, further sacrifices had to be made to demonstrate to my current and future employers that I'm a reliable and responsible team member.

Time being a heavy constraint meant certain functionalities of the system that would have further supported the application in filling the gap in the market could not be implemented as originally intended.

Encryption of stored messages on the blockchain

Messages are stored in plaintext on the blockchain. Storing messages in an encrypted format using asymmetric cryptography would enable only the sender and the receiver to view the message, benefitting user privacy and the system's compliance towards the GDPR.

HTTPS instead of HTTP and WSS instead of WS

The client-server uses Hypertext Transfer Protocol (HTTP) to interact with the web interface. However, implementing Hypertext Transfer Protocol Secure (HTTPS) would benefit users with a more secure and private connection to the client-server by ensuring data confidentiality using encryption, authentication of servers using digital certificates and data integrity using digital signatures. The light node communicates with the service node using WS Sockets and could also benefit from using WSS Socket like the above.

Distributed SQL database

All user accounts are stored in one single SQL database. Having a distributed service of databases all holding user account records benefits the user by providing increased security due to data replication across multiple servers, making it harder for a malicious actor to try to comprise the whole database. Further benefits towards the user include increased response times as distributed databases can store user credential data close to the end-users, reducing latency when accessing data.

Decentralised and distributed blockchain

Currently, only one service node contains the only copy of the blockchain.

Implementing a multiple service node network, each holding a synchronized copy of the blockchain, would achieve a Web3 messaging application. This benefits the user by providing greater security as no single point of failure makes it less vulnerable to malicious attacks and provides a trustless environment where network control is distributed between the service nodes meaning users do not have to trust a central authority. Furthermore, messages can be processed to the blockchain through multiple service nodes, enabling the network to handle higher volumes of data to benefit the user with a more efficient system.

5 Professional Issues

5.1 Gantt Table

Please refer to **Appendix B.1** to view the Gantt Table.

The Gantt timeline for the project is used throughout the program to provide a high-level view of the overall project. The dates are scheduled around the individual milestones to aid cooperation and guidance throughout the project. Thus, providing a broad-range perspective enables us to see what specific actions are taking place. The column labelled “Completed”, and the colour of the Gantt Chart serve as indicators as to what stage the activity is at. For example, **RED** represents the task being incomplete/not started. **YELLOW** means that the task has been started but still needs further attention. Finally, **GREEN** indicates that the task has been completed sufficiently.

5.2 Gantt Chart

Please refer to **Appendix B.3** to view the Gantt Table.

To visualize the entire project schedule, a Gantt chart was implemented and represented on a single line to help identify potential issues and manage resources more efficiently relevant to dependencies, tasks, and deadlines.

The Gantt Chart was configured on a 2-dimensional bar chart showing the sequence of tasks and the time spent on each task.

This brought a range of benefits, such as the tasks being more clearly and visually represented to identify inefficiencies and areas of improvement, being able to spot overlapping activities so that you can see which tasks are dependent on each other and helping track the overall process of the application to spot delays in areas and adjust the project schedule in other areas as necessary to stay on track.

5.3 Risk Analysis

Please refer to **Appendix B.2** to view the Risk Management.

5.4 Professional Issues

The developed blockchain messaging application processes sensitive information, which must conform to the many present codes of conduct to ensure user safety. The GDPR Act [28], Computer Misuse Act [29], Intellectual Property Act [30] and the BCS Code of Conduct [31] are the codes researched and quoted throughout the specific issues arising from developing a blockchain messaging application.

5.4.1 Security Issues – GDPR Act

The issue and the relevant article the application must conform to are listed with how the application has or can obey.

Data breaches - Article 33 Personal Data Breaches: User messages should be encrypted and stored securely on an immutable digital ledger using blockchain technology.

Man in the middle attacks - Article 32 Security of processing: Communication between the client server and nodes requires secure protocols.

Data loss and corruption - Article 33 Personal Data Breaches: Data replication on an implemented array of service node network mitigates the risk and ensures that the data is available even if a service node goes offline.

DDoS attacks - Article 18 Right to restrict processing: Implementation of a distributed network enabling the processing of messages around the network, allowing multiple nodes to share the load of the network.

Storage of User accounts - Article 5 Principles relating to processing of personal data: Users are only allowed to use a strong password to register an account, which is encrypted when stored in the database.

Distribution of malware/phishing – Computer Misuse Act Fraud Act 2006: Smart contracts can be further implemented to only allow verified authenticated software to be executed on the network.

5.4.2 Legal – BCS Code of Conduct

Obtain user consent - Article 4 Definitions: The application would need to gain consent from the user providing explicit and unambiguous consent for their data to be collected, processed, and stored on the blockchain network.

Provide Transparency and Protection of User Data - Article 12 Transparent information, communication, and modalities for the exercise of the rights of the data subject and Article 13 Information to be provided where personal data are collected from the data subject:

Application could be made open-source to provide transparency to users of how the application exactly works. Messages should be encrypted and stored on the blockchain to ensure that user data is stored securely.

Respond to User Requests - Article 15 Right of access by the data subject: The application gives users the right to request their personal data to allow them to edit or delete their stored personal user information on the SQL database; however, messages stored on the blockchain cannot be removed, making it complicated to comply with the GDPR Act fully.

5.4.3 Social – BCS Code of Conduct

The relevant sections of the code of conduct have been listed, with how the application aims to conform to the code.

Section 1c Public Interest - conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age, or disability, or of any other condition or requirement: The application will only ask users for a username and a password to identify accounts on the applications, making it impossible for the application to perform any discrimination towards users of the application.

Section 1d Public Interest - promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise: The application should be distributed equally between society to prevent the widening of the digital divide and the exclusion of certain groups from accessing the technology.

5.4.4 Ethical

The following ethical issues that users and developers should be aware of in the blockchain messaging application.

Privacy of messages on the blockchain: Depending on the nature of the blockchain in question, user may not like the transparent nature of the blockchain due to their messages being publicly accessible to everyone on the network, even if they are encrypted.

The anonymity of the network - 1997 Harassment Act.: Users are provided with anonymous communication on the network, which can enable the application to be used in unethical activities such as cyber bullying.

Adoptability: Although the sector is growing rapidly, blockchain-related businesses are still largely unregulated in most countries. Sadly, in some circles, the use of cryptocurrency is still perceived as a mechanism or tool to facilitate criminal activity, creating difficulties for user adoptability.

5.4.5 Environmental

Energy consumption:

Mining process requires a large amount of computational power to solve complex mathematical problems, contributing to carbon emissions. This has been addressed by low-power, fully optimized, proof-of-work algorithms.

5.4.6 Intellectual Property

Ownership of digital assets: Users' intellectual property stored on the blockchain is digitally signed to ensure authenticity of ownership can be proved.

6 Conclusion

6.1 What was achieved

The paper has explored the development of a blockchain messaging application, which has the potential to revolutionise how people can communicate and exchange information online.

Relevant background research into existing messaging applications displayed the ongoing issues presented by users, illustrating the potential gaps in the messaging application market. Further investigation discovered how blockchain technology can be utilised to tackle this market gap.

The paper describes the technical approach to designing, implementing, and testing a blockchain messaging application using a blockchain network and a user interface to provide a real-time messaging application called hChat. Testing the application aided and concluded a robust application that fulfils all user requirements by leveraging blockchain technology to supply consumers with chatting capabilities.

Future work of the paper explains how the application can incorporate a decentralised network making it a Web3 application, and the benefits this has towards users.

Overall, blockchain technology has demonstrated itself as being capable of being used outside the cryptocurrency realm, making it a pivotal contender in paving the way for future security in messaging applications.

6.2 Future work

- **Group chats** – Implementing group chatrooms which benefit the user by creating a convenient way for groups such as friends, family, and remote teams to connect and share all at once.
- **Status of messages** – Implement a tick underneath each sent message that would be displayed indicating to the user the recipient has viewed the message benefitting the sender with satisfaction that the message has been received and read.
- **Status of users** – When the list of users is displayed, usernames will be highlighted in green, indicating to the user which users are online. This enhances the real-time nature of the communication experience.
- **Message notifications** – When users are in a chatroom and receive a message in another chatroom, a message notification will appear on the screen informing them of the chatroom in which they have received a message. This ensures that users stay informed about new messages and activities in other chatrooms.
- **Messages received when offline** - When viewing the list of users, message notifications will be marked on the chatrooms where they have received messages while offline. This ensures that users can be updated on all messages received while offline.

7 References

- [1] S.Dixon, 2023. *Most popular social networks worldwide as of January 2023, ranked by number of monthly active users.* [Online]
Available at: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>
[Accessed 12 October 2022].
- [2] PyCoach, T., 2022. *Is Data The New Oil of the 21st Century or Just an Overrated Asset?.* [Online]
Available at: <https://towardsdatascience.com/is-data-the-new-oil-of-the-21st-century-or-just-an-overrated-asset-1dbb05b8ccdf#:~:text=%E2%80%9CData%20is%20the%20new%20oil.,for%20it%20to%20have%20value.%E2%80%9D>
[Accessed 17 October 2022].
- [3] Bateman, T., 2021. *WhatsApp rewrites its Europe privacy policy after a record €225 million GDPR fine.* [Online]
Available at: <https://www.euronews.com/next/2021/11/22/whatsapp-rewrites-its-europe-privacy-policy-after-a-record-225-million-gdpr-fine>
[Accessed 19 October 2022].
- [4] Coker, J., 2021. *Ireland's DPC Fines Meta €265m Following Large-Scale Data Leak.* [Online]
Available at: <https://www.infosecurity-magazine.com/news/ireland-dpc-fines-meta-data-leak/>
[Accessed 17 October 2022].
- [5] Marlinspike, M., 2016. *WhatsApp's Signal Protocol integration is now complete.* [Online]
Available at: <https://signal.org/blog/whatsapp-complete/> /
[Accessed 10 November 2022].

- [6] Panghal, A., 2018. WhatsApp's End to End Encryption, How does it work?. [Online]
Available at: <https://medium.com/@panghalamit/whatsapp-s-end-to-end-encryption-how-does-it-work-80020977caa0>
[Accessed 6 November 2022].
- [7] Cressler, C., 2021. Understanding WhatsApp's Architecture & System Design. [Online]
Available at: <https://www.cometchat.com/blog/whatsapps-architecture-and-system-design#:~:text=WhatsApp%20uses%20a%20highly%20modified,socket%20to%20retrieve%20the%20messages>
[Accessed 10 November 2022].
- [8] Yadav, V.K., Singh, S. and Sharma, G., 2022, December. A Review of Cryptography Techniques for Securing Data on Messaging and Cloud Applications. In 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 1880-1884). IEEE. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10074564>
- [9] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System, <https://bitcoin.org/bitcoin.pdf> namecoin (2014) Namecoin, <https://www.namecoin.org/>
- [10] Frankenfield, J. (2019). *Proof of Stake (PoS)*. [online] Investopedia. Available at: <https://www.investopedia.com/terms/p/proof-stake-pos.asp>.
- [11] Graves, D. / S. (2020). Status wants to be the world's most private instant messaging app. [online] Decrypt. Available at: <https://decrypt.co/25629/status-wants-to-be-the-worlds-most-private-instant-messaging-app> [Accessed 19 November 2022].
- [12] Status. (n.d.). Status - Status Node Runners. [online] Available at: <https://status.im/status-nodes/#:~:text=Status%20nodes%20support%20decentralized%2C%20private>
[Accessed 8 Apr. 2023].
- [13] Status. (n.d.). Status - Running Status Node. [online] Available at: https://status.im/technical/run_status_node.html [Accessed 8 Apr. 2023].

- [14] Manning, D.A. and Zerouali, M. (2018). *Status ENS Integration: Smart Contract Security Review*. [online] Sigma Prime. Available at: <https://blog.sigmaprime.io/status-ens-review.html> [Accessed 14 Novemeber. 2022].
- [15] Status Specification. (n.d.). 10/WAKU-USAGE. [online] Available at: <https://specs.status.im/spec/10> [Accessed 18 Novemeber. 2022].
- [16] Schlitter, F., San Pedro, J.C., Freeman, P. and Lowcay, C., 2020. Sylo Protocol: Secure Group Messaging. Available at: <https://dev.sylo.io/whitepaper/sylo-protocol.pdf>
- [17] Clarke, J.; Castro, R.R.; Sharma, A.; Lopez, J.; Suri, N. *Trust & security RTD in the internet of things: Opportunities for international cooperation*. In *Proceedings of the First International Conference on Security of Internet of Things*, Kollam, India, 17–19 August 2012; pp. 172–178. Available at: https://web.archive.org/web/20160913225729id/http://www.bic-trust.eu/files/2013/01/Clarke_ACM_SecurIT_Dec20121.pdf
- [18] Abdulaziz, M., Çulha, D. and Yazici, A., 2018, December. A decentralized application for secure messaging in a trustless environment. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)* (pp. 1-5). IEEE. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8625362>
- [19] WhatsApp, 2021. About end-to-end encryption. [Online] Available at: <https://faq.whatsapp.com/820124435853543> [Accessed 15 December 2022].
- [20] P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,” in *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65. Available at: https://moodle.epfl.ch/pluginfile.php/2327928/mod_resource/content/2/Kademlia.pdf
- [21] Joshi, M. and Hadi, T.H., 2015. A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*. Available at: <https://arxiv.org/ftp/arxiv/papers/1507/1507.05722.pdf>

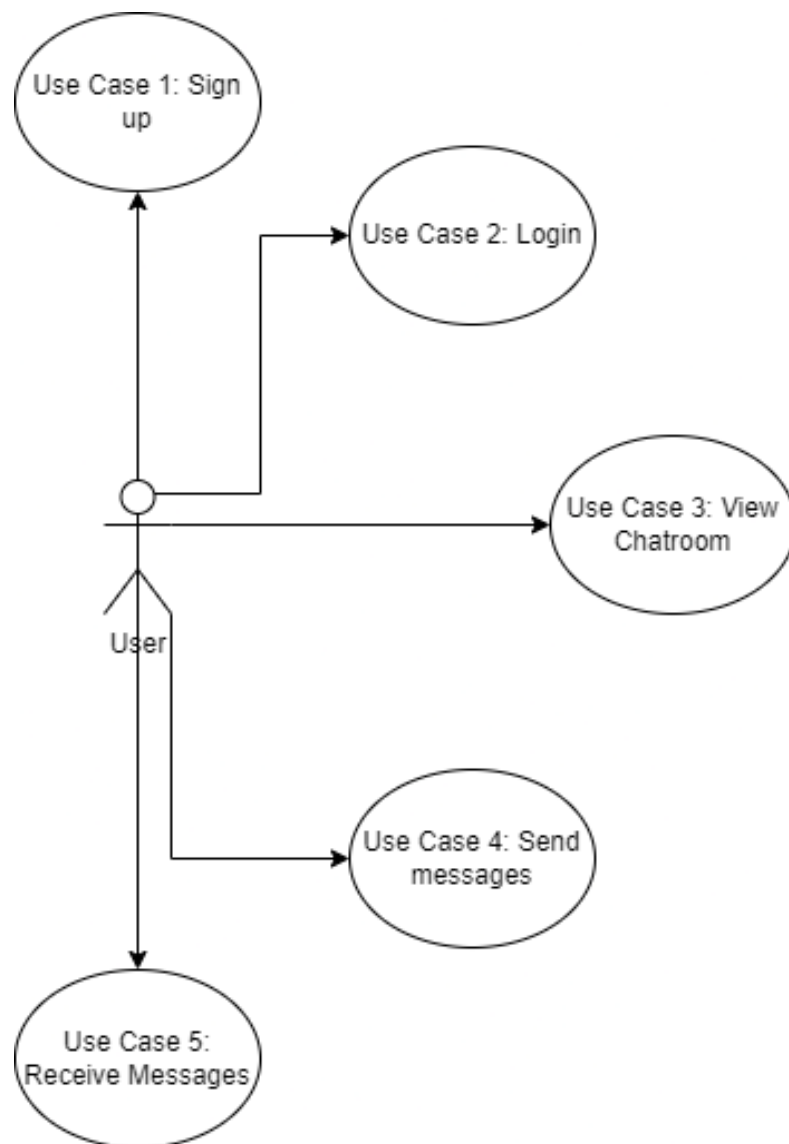
- [22] Weidner, M., Kleppmann, M., Hugenroth, D. and Beresford, A.R., 2021, November. Key agreement for decentralized secure group messaging with strong security guarantees. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2024-2045). Available at: <https://dl.acm.org/doi/pdf/10.1145/3460120.3484542>
- [23] Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková. 2021. Mesh Messaging in Large-Scale Protests: Breaking Bridgefy. In *Cryptographers' Track at the RSA Conference*. Springer, 375–398. Available at: https://link.springer.com/chapter/10.1007/978-3-030-75539-3_16
- [24] Greschbach, B., Kreitz, G. and Buchegger, S., 2012, March. The devil is in the metadata—new privacy challenges in decentralised online social networks. In *2012 IEEE international conference on pervasive computing and communications workshops* (pp. 333-339). IEEE. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6197506&casa_token=qFwTw9QdahcAAAAA:zZ0oZeRS7A5x4u7GS15fHOTXpF13l_MwVnT-MbvA5DDraIWKQZScNKrF8viQtPUzR6kH5EU&tag=1
- [25] Gebhardt, L., Leinweber, M., Jacob, F. and Hartenstein, H., 2022, October. Grasping the Concept of Decentralized Systems for Instant Messaging. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education* (pp. 1-6). Available for download at: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Gebhardt%2C+L.%2C+Leinweber%2C+M.%2C+Jacob%2C+F.+and+Hartenstein%2C+H.%2C+2022%2C+October.+Grasping+the+Concept+of+Decentralized+Systems+for+Instant+Messagin g.+In+Proceedings+of+the+17th+Workshop+in+Primary+and+Secondary+Computin g+Education+%28pp.+1-6%29.+&btnG=
- [26] Ferretti, S., Zichichi, M. and Sparber, J., Blockchain-based end-to-end encryption for Matrix instant messaging. Available at: <https://mirkozichichi.me/assets/tutor/theses/mscsparber.pdf>

- [27] Ellewala, U.P., Amarasena, W.D.H.U., Lakmali, H.S., Senanayaka, L.M.K. and Senarathne, A.N., 2020, December. Secure Messaging Platform Based on Blockchain. In 2020 2nd International Conference on Advancements in Computing (ICAC) (Vol. 1, pp. 317-322). IEEE. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9357306&casa_token=483GdrNq738AAAAA:PZgxIW3pPiJk21-IAR4jXSaLRfPWPCPwkPDJkTu0ps4EOS3j3II0qBAF3Q4kl2ZQhFUYYrs&tag=1
- [28] GDPR (2018). General Data Protection Regulation (GDPR). [online] General Data Protection Regulation (GDPR). Available at: <https://gdpr-info.eu/>.
- [29] legislation.gov.uk (1990). Computer Misuse Act 1990. [online] Legislation.gov.uk. Available at: <https://www.legislation.gov.uk/ukpga/1990/18/contents>.
- [30] Anon (2011) Copyright, Designs and Patents Act 1988. Legislation.gov.uk. Available at <https://www.legislation.gov.uk/ukpga/1988/48/contents>
- [31] BCS (2015). BCS, THE CHARTERED INSTITUTE FOR IT CODE OF CONDUCT FOR BCS MEMBERS. [online] Available at: <https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf>.

8 Appendix

Appendix A – Software Modelling and Diagrams:

Appendix A.1 – Use Case Diagram



Appendix A.1.1 – Use Case Documentation:

Use Case 1 – Sign up

Use Case Name:	Sign up
Description:	This use case allows users to register an account on the application using a web interface, so that they can login.
Primary Actor:	User
Preconditions:	1. User has to input a unique username and a password that's 8 characters or longer
Postconditions:	1. The system displays the login web page.

Use Case 2 - Login

Use Case Name:	Login
Description:	This use case allows users to login via a web interface to access their account, so that they use the services provided.
Primary Actor:	User
Preconditions:	1. User has to have a registered account
Postconditions:	1. The system displays the main menu webpage

Use Case 3 – View Chatroom

Use Case Name:	View Chat room
Description:	This use case allows users to select a radio button and press select, to be displayed the chosen user's chatroom between the two parties.
Primary Actor:	User
Preconditions:	1. User must be logged in
Postconditions:	1. The system displays the chatroom webpage and displays the conversation between the logged in user and the selected user.

Use Case 4 – Send Messages

Use Case Name:	Send Messages
Description:	This use case allows users in the chatroom to input and send a message to the light node, which sends it to the service node, so that it can be mined and added to the blockchain.
Primary Actor:	User
Preconditions:	1. User must be in a chatroom
Postconditions:	1. The message is mined and added to the blockchain and sent to the light node. The light node receives the message and sends it to the client server which updates the chatroom.

Use Case 5 – Receive Messages

Use Case Name:	Receive Messages
Description:	This use case allows users to receive messages in real-time.
Primary Actor:	User
Preconditions:	1. User must be in the chatroom of the user who's sending the message.
Postconditions:	1. Recipient's message is added to the blockchain, and the block is sent to the light node. The light node receives the block and retrieves the message, which is sent to the client server, which updates the chatroom with the message.

Appendix A.2 – Sequence Diagrams

Use Case 1 – Sign up

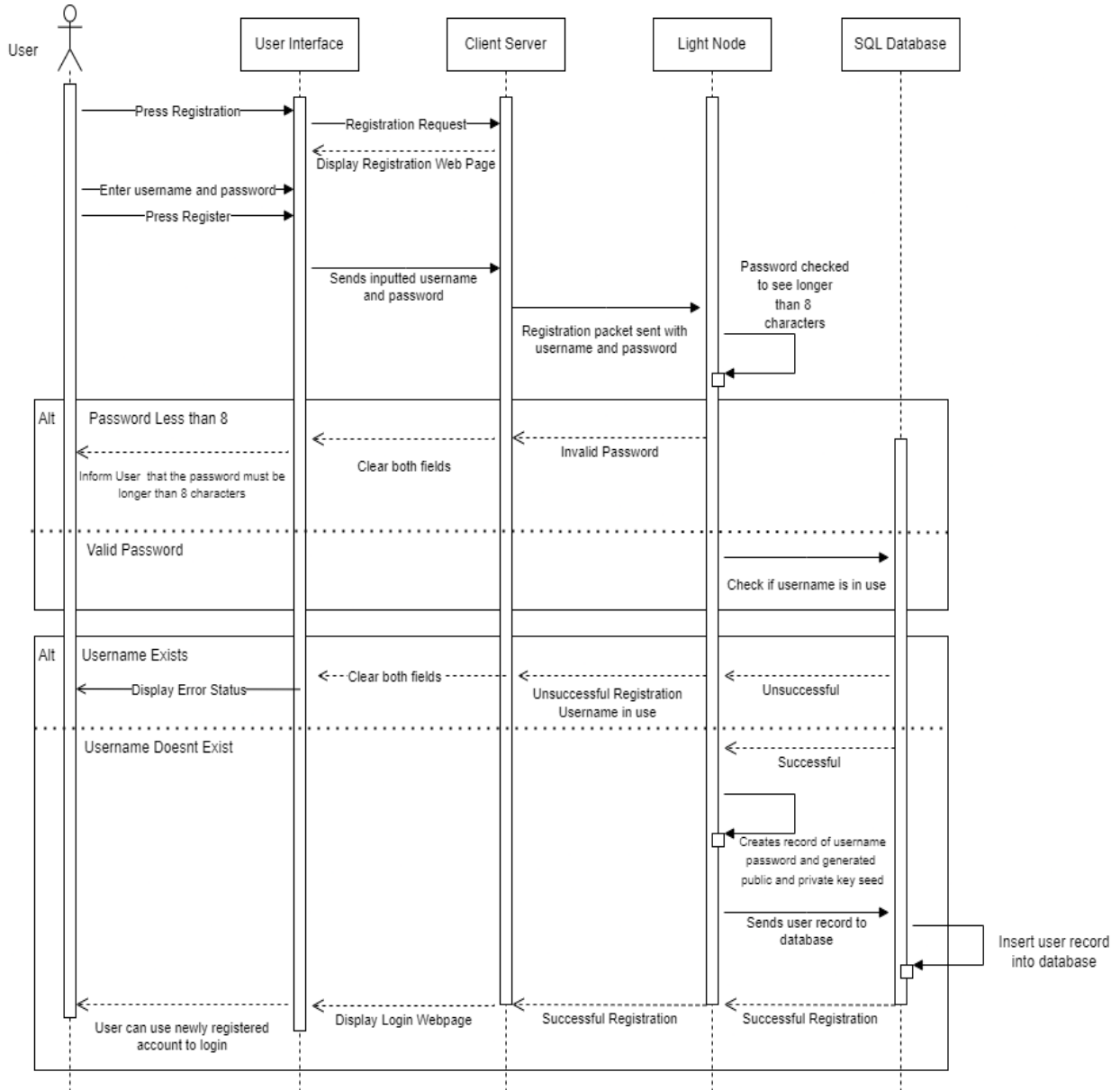


Figure 14 - Use Case 1 – Sign up Sequence Diagram

Use Case 2 - Login

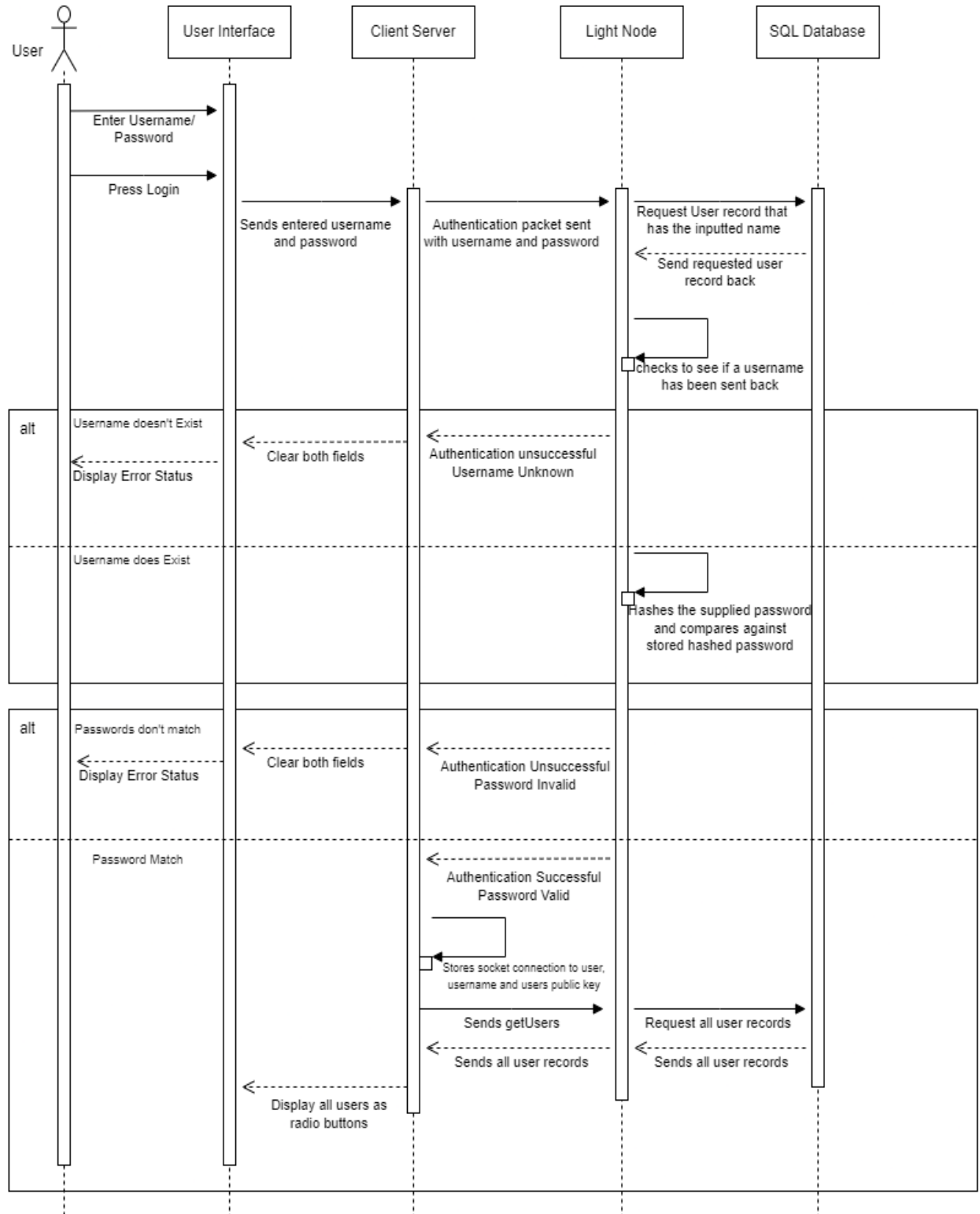


Figure 15 - Use Case 2 - Login Sequence Diagram

Use Case 3 – View Chatroom

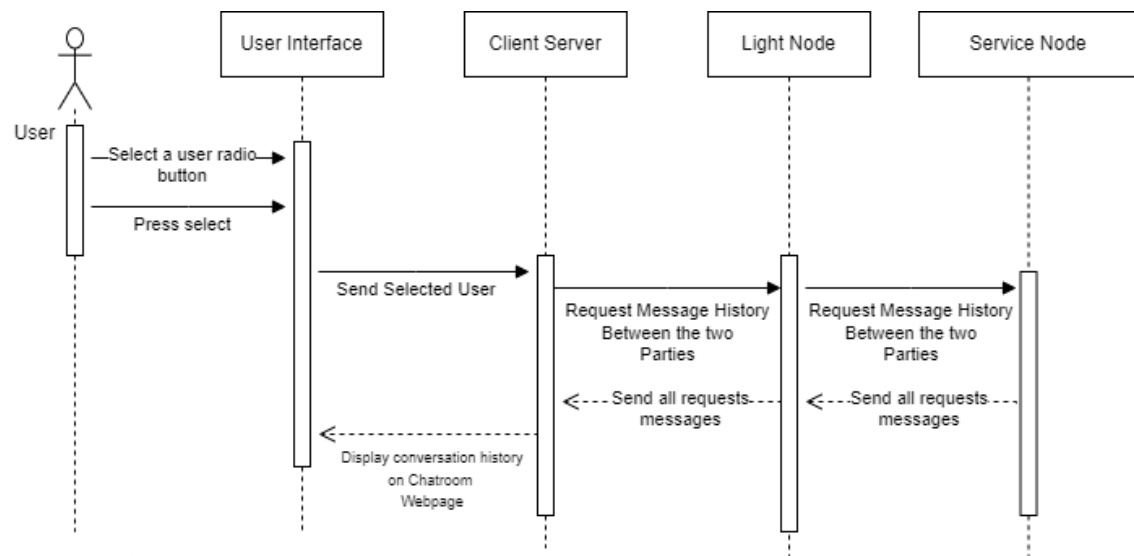


Figure 16 - Use Case 3 – View Chatroom Sequence Diagram

Use Case 4 – Receive Messages

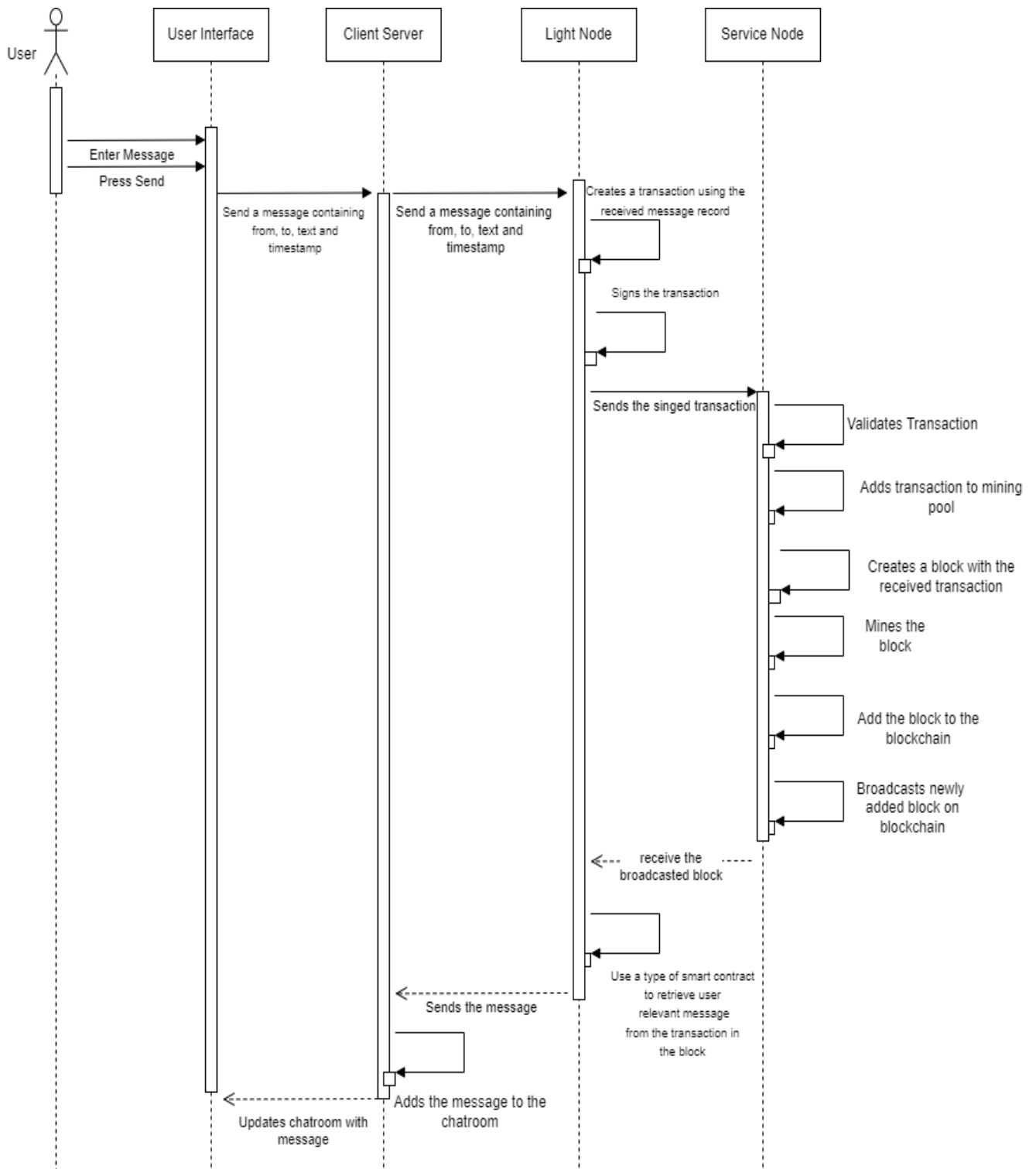
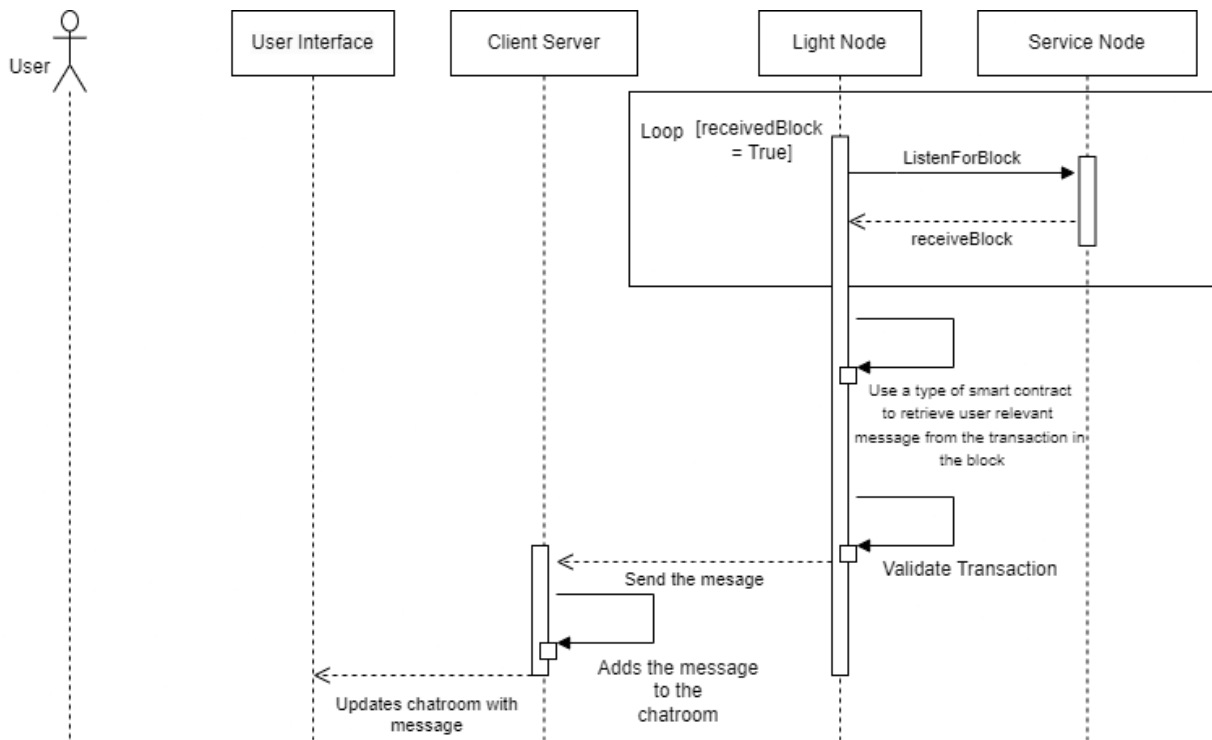


Figure 17 - Use Case 4 – Send Messages Sequence Diagram

Use Case 5 – Receive Messages



Appendix B – Professional Issues

Appendix B.1 – Gantt Chart

Task	Start Date	End Date	Duration (DAYS)	Completed
Literature Search	03/10/2022	10/10/2022	7	✓
Develop Gantt Chart	10/10/2022	13/10/2022	3	✓
Create Annotated Bibliography	13/10/2022	20/10/2022	7	✓
Develop Project Proposal	20/10/2022	29/10/2022	9	✓
Literature Review	29/10/2022	10/11/2022	12	✓
Requirements, Gathering and Modelling	10/11/2022	23/11/2022	13	✓
Review Project Proposal Feedback	15/11/2022	23/11/2022	8	✓
Design Program	23/11/2022	01/12/2022	8	✓
Program Implementation	01/12/2022	03/04/2023	123	✓
Prototype 1	01/12/2022	17/12/2022	16	✓
Prototype 2	17/12/2022	27/01/2023	41	✓
Prototype 3	27/01/2023	24/03/2023	56	✓
Final Prototype 4	24/03/2023	29/03/2023	5	✓
Testing and Evaluating	29/03/2023	05/04/2023	7	✓
Write Final Report	05/04/2023	15/04/2023	10	✓
Create Video	15/04/2023	16/04/2023	1	✓
Create Poster	16/04/2023	17/04/2023	1	✓

Figure 19 - Project Gantt Table

Appendix B.2 Risk Management

Risk Tables

Likelihood of Risk

Likelihood of Risk	
Chance	Score
Rare	1
Unlikely	2
Possible	3
Likely	4
Certain	5

Attention required to risk

Risk Score	Category
0-6	Acceptable
7-14	Attention Needed
15-25	Unacceptable

Severity

Severity	
Damage	Score
Minor	1
Moderate	2
Major	3
Catastrophic	4

Risk Analysis

Risk	Cause	Severity	Likelihood	Risk Rating	Mitigation
Missed Deadline	Illness	4	2	8	Healthy diet and regular exercise. Applying for exceptional circumstance if severity of illness is halting progress.
	Other Assignments taking priority	4	2	8	Creating a balanced work plan which uses a priority level system which considers assignments difficulty, length, and due date.
	Christmas Job—working too many hours	4	1	4	When arranging hours to work, make sure to say I can only work a maximum of 15 hours per week.
	Lacking Motivation	5	2	10	Commit to ways of self-care to improve motivation, such as walking in nature. As well as giving self-appraisal to keep positive throughout project.
	Poor time management	3	1	3	Check Gantt chart schedule daily to notify what tasks need to be done for the day. As well as cracking bigger problems into smaller, more achievable tasks.
Loss of Data	Poor Version Control	4	4	16	Commit to good practices of version control. Such as committing files with a single purpose, with detailed comments explaining additions to program. As well as making sure each commit is traceable to aid debugging.

	Human Error	4	1	4	Make sure that each file worked on is backed up on the system and familiarise myself with how the backups can be made.
	Hardware Damage	5	1	5	Implement source control, with work being saved regularly. In the addition of keeping machinery in appropriate conditions, such as cleaning excess dust to mitigate overheating.
	Power Outrage	5	1	5	Adoption of automatic saves and backups where available.
Technical Issues	Software bugs and errors	3	4	12	Follow and use techniques such as Continuous Integration Continuous Testing (CICT) and Test-Driven Development (TDD).
	Inadequate test plan	3	3	9	Creation of a test plan that has focal points based around customer needs with multiple forms of testing, to ensure functionally and prevent delays.
	Inadequate software designs	2	3	6	Develop several software designs and continuously changing the portions that are ambiguous.

Appendix B.3 - Gantt Table

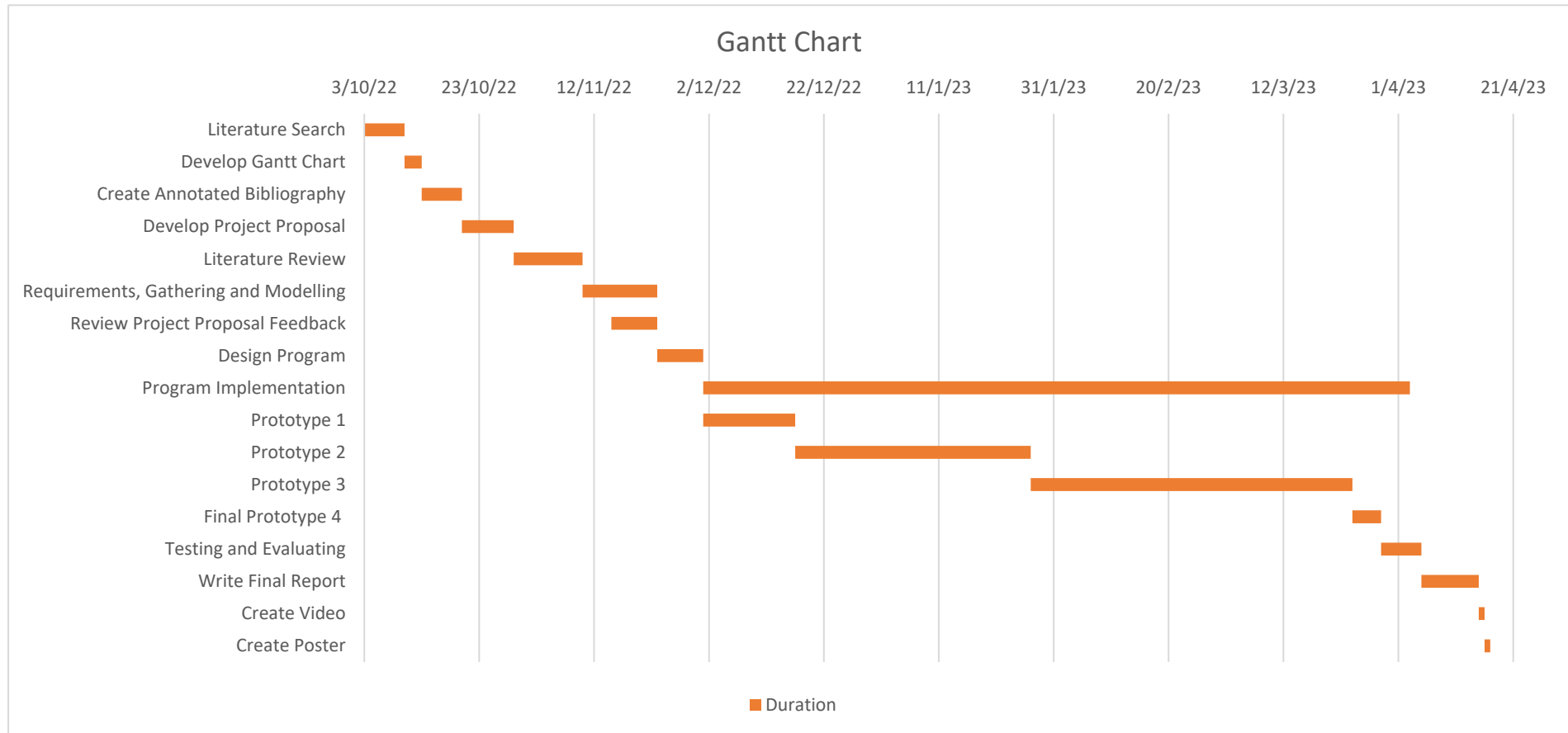


Figure 20 - Project Gantt Chart

Appendix C – Third-Party Libraries Used:

Appendix C.1 – Node.js

Node.js is an open-source, server-side JavaScript runtime environment that provides cross-platform support, including Windows, Linux and macOS. Node.js enables developers to write server-side code to develop real-time applications built with an event-driven nature to effectively manage asynchronous I/O activities. To achieve real-time performances for applications Node.js uses event loops, callbacks, and non-blocking I/O to enable nodes to meet the demands of modern users of applications. This library was used to configure the client server, light node and service node on the network.

Node.js Modules and Frameworks

Ws - Module

The ws library is a Node.js module that provides WebSocket server and client implementation. The library can be used to create a WebSocket that listens for incoming connections from clients, to enable bidirectional communication between the two parties. The library was used as a protocol for communication between the service node and the light node, service node and its peers and the creation of the service node server.

Bcrypt - Module

Bcrypt is a cryptographic hash function that is designed to hash and compare passwords using Bcrypt algorithms.

Bcrypt was used when a new user has registered. Their entered password is hashed and stored on the SQL database in an encrypted form. Bcrypt compare is used when a user logs in and their entered password is then hashed and then compared with stored hash password on the database.

Express.js - Framework

Express.js is an open-source web framework for node.js that offers extensive features and functionalities for creating web applications. The easy-to-use API that's provided means developers do not have to learn complex syntax and programming concepts, making it great for beginners. In the application Express was used to create the light node, which is a web server, that can communicate with the client server and the service node.

http - Module

This module enables Node.js to transfer data using the Hyper Text Transfer Protocol and was used by the express server when communicating.

[Appendix C.2 – React.js](#)

React.js is an open-source JavaScript framework and library that enables developers to build interactive user interfaces for web applications more efficiently as opposed to vanilla JavaScript. React.js functions by disassembling user's interfaces into interfaces into manageable, reusable parts and employing a virtual DOM to refresh the user interface when something changes. This library was used to create the user interface on the web interface application.

[React.js Module](#)

[ScrollToBottom - Module](#)

ScrollToBottom is a react container that auto scrolls to the top or bottom if the viewpoint is at the bottom and new content is added.

ScrollToBottom was used when displaying chat messages to ensure that the current message is displayed at the bottom of the chatroom window, with the messages scrolling upwards on arrival on subsequent messages.

[Appendix C.3 – Crypto-js](#)

Crypto-js is a JavaScript open-source library used in browser-side and service-side applications. The library holds and provides a collection of cryptographic algorithms and utilities for encryption, decryption, and hashing. The function SHA256 was used from the library to hash blocks and sign transactions and was not changed during the development of the application. Developers using the library can benefit from its flexibility and easy integration of web applications.

[Appendix C.4 – Elliptic](#)

Elliptic is a JavaScript library that contributes a collection of cryptographic primitives based on elliptic curve cryptography (ECC). ECC is a public-key cryptography that creates a secure mechanism for exchanging messages between two parties. The library was used to create a pair of private and public keys for each user, which was stored in the SQL user database. Public keys were used to identify who transactions were from and to, as well as to sign the transaction.

[Appendix C.5 – Socket.io-client](#)

Socket.io-client is a client-side JavaScript library that allows browsers to connect to a server using socket.io. It provides a simple API for managing and establishing bidirectional connections to a Socket.io server and features such as automatic reconnection, multiplexing and event acknowledgement. Socket.io-client was used to connect the user to the client-server to pass user entries and display messages to the user.

[Appendix C.6 SQL Server](#)

A SQL Server is based on the Structured Query Language (SQL) programming language, which stores and manages data stored on a database. The server can provide high availability using techniques such as clustering and includes security with built-in features of encryption and auditing.

A SQL server was used in the application to store user records consisting of username, hashed password, public key, and private seed. The server also received and processed requests from the light node to check if usernames existed and verify credentials from users logging in.

Appendix D – Software used to plan and develop dissertation

Appendix D.1 – Draw.io

Draw.io is a free web-based application that offers extensive processing options for creating professional-looking diagrams and charts.

This software was used to create the use case diagram, flowchart, sequence diagram, designs of web pages, UML of the blockchain software architecture, service node, light node and the client-server and network architecture.

Appendix D.2 – Visual Studio IDE

Visual Studio is an IDE that provides a broad range of tools and features for developing, testing, and debugging. The IDE offers users a code editor with syntax highlighting and code completion and provides powerful debugging by supporting users with breakpoints, call stacks, and watch windows. The IDE supports JavaScript applications and was used to create hChat.

Appendix D.3 - GitHub

GitHub is a web-based platform that enables software developers to store, share and manage their projects. The Git version control system, on which GitHub is based, provides functionalities such as allowing programmers to track changes to their code, make branches for various versions of their project, and combine modifications from many contributors.

Appendix E - Testing

E.1 User Functional Requirements Testing

Login Webpage

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Users can access registration webpage	Press "Registration" button	Registration webpage is displayed	Registration webpage displayed	✓
2	User can login into their account	Username: "h" and password "Harrison" (valid username and password) press login	Main menu webpage displayed	Main menu displayed	✓
3a	Users cannot login with invalid credentials	Username: "Harrison" and password "123" (valid username and invalid password) press login	Status bar contains "Invalid Login", inputs are cleared, and user allowed to try again	"Invalid" displayed in status bar; inputs cleared; user allowed to try again	✓
3b	Users cannot login with invalid credentials	Username: "123" and password "123" (invalid username and invalid password) press login	Status bar contains "Invalid Login", inputs are cleared, and user allowed to try again	"Invalid" displayed in status bar; inputs cleared; user allowed to try again	✓

Login webpage - Test Number 1

Registration

Register

Cancel

Login webpage - Test Number 2

Live Chat User h

- h;
- j;
- jack;
- a;
- s;
- g;
- r;
- n;
- asd;
- dfg;
- ghjk;
- dfgdfg;
- sdfs;
- aaaa;

Selected User:

Login Webpage - Test Number 3

Login

Login

Registration

invalid

Login Webpage - Test Number 4

Login


Login

Registration

invalid

Registration Webpage

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Users can register an account	Username: "Jack" password: "Jack1234" and press register (unique username and sufficient password length)	User informed of valid registration and is redirected to the login webpage	User successfully creates an account and is directed to the login webpage	✓
2	Usernames must be unique when registering	Username: "Harrison" password: "hello123" and press register (username exists and sufficient password length)	User is informed of the unsuccessful registration as the name is not unique, inputs cleared, and user allowed to try again	Status bar contains "username in use", clears inputs and allows the user to try again	✓
3	Passwords must be of a sufficient length (at least 8 characters long)	Username: "Henry" password: "123" and press register (unique username and insufficient length password)	User is informed of the unsuccessful registration as the password is not long enough, inputs cleared, and user allowed to try again	Status bar contains "invalid password", clears inputs, and allows the user to try again	✓
4	Usernames must be unique, and passwords must be at least 8 characters long	Username "Harrison" and password "123" and press register (username exists and password not 8 characters long)	User is informed that the username is not unique, and the password is not long enough, inputs cleared, and user allowed to try again	Status bar contains "invalid username & password" inputs cleared and allows the user to try again	✓

5	Users can return to the login webpage	Press cancel registration	User redirected to login webpage	User redirected to login webpage	
---	---------------------------------------	---------------------------	----------------------------------	----------------------------------	---

[Registration Webpage - Test Number 1](#)

Register: jack

Login

[Registration Webpage – Test Number 2](#)

Live Chat User h

- h;
- j;
- jack;

Selected User:

Select

[Registration Webpage - Test number 3](#)

Registration

Invalid password

[Registration Webpage - Testing number 4](#)

Registration

Invalid Username&password

Registration Webpage - Test Number 5

Login

Main Menu Webpage

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Users can scroll up and down the user list to see all users	Mouse is held down on the scroll bar and moved up and down	User is able to scroll up and down the list	User is able to see all registered users by scrolling up and down the list	✓
2	Users can select a user and enter the chosen chatroom	Select a radio button a press "Enter chatroom"	User is directed to the chatroom webpage between the selected user	User is directed to the chatroom webpage between the selected user	✓

Main Menu Webpage – Test Number 1

Live Chat User h

● jack;

● a;

● s;

● g;

● r;

● n;

● asd;

● dfg;

● ghjk;

● dfgdfg;

● sdfs;

● aaaa;

● aasdasd;

● asdasd;

Selected User:

Select

Main Menu Webpage – Test Number 2

Live Chat User h to j

hello!

23:21:7 h

Hi, how are you?

23:21:15 j

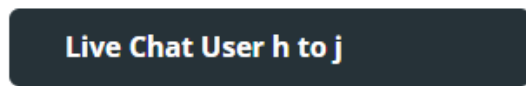
Hey...

Cancel_chat

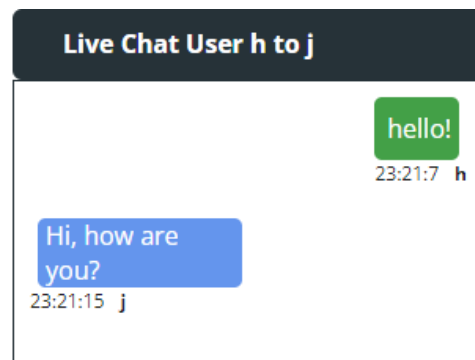
Chatroom Webpage

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Users are informed of the other party in the chatroom	N/A	Recipient chatroom member is displayed	Recipient chatroom member is displayed	✓
2	Users are displayed the conversational history of the chatroom	N/A	History of chatroom is displayed	History of chatroom is displayed	✓
3	Users can send messages in the chatroom	"hello" into input box	Message is displayed in green on the right-hand side of the chatroom	Message is displayed in green on the right-hand side of the chatroom	✓
4	Users can receive messages in the chatroom	Sender inputs "hello" into input box	Receiver sees message displayed in blue on the left-hand side of the chatroom	Message is displayed to the receiver on the left-hand side in blue in the chatroom	✓
5	Users can exit chatrooms	Click exit chatroom	User is directed to login webpage	User is directed to login webpage	✓

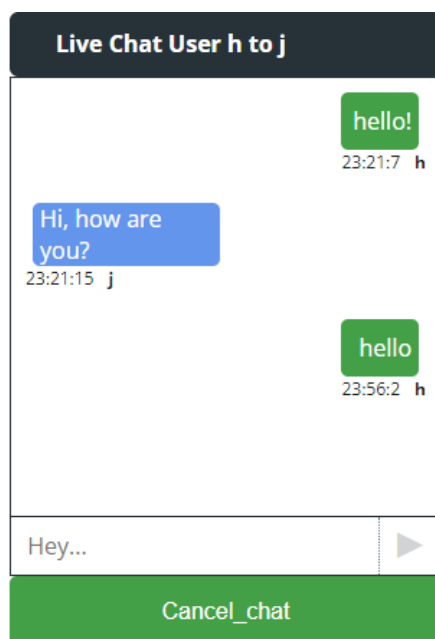
Chatroom Webpage – Test number 1



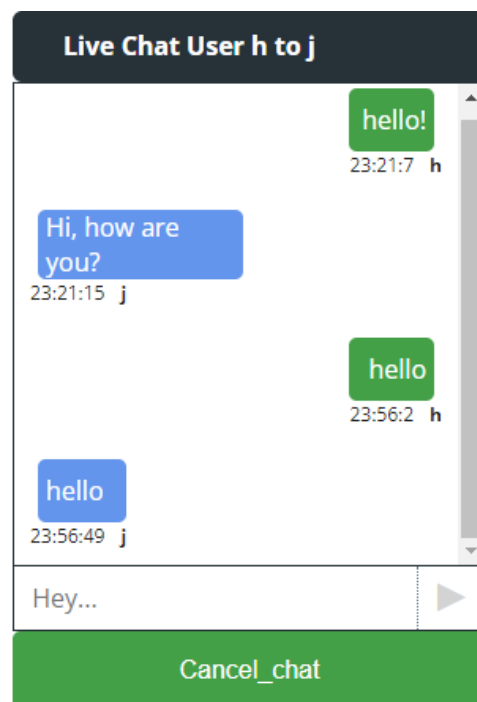
Chatroom Webpage – Test number 2



Chatroom Webpage – Test number 3



Chatroom Webpage – Test number 4



Chatroom Webpage – Test number 5

Login

Login

Registration

E.2 Sprint Plans and Testing

E.2.1 Sprint Backlog

ID	User Story	Tasks	Status	Effort (/5)
1	As a user, I want the application to run on a web browser so that I can access the application via a phone or computer	Create a client server	Completed	4
2	As a user, I want to communicate on a network that uses blockchain technology so that my messages are stored and are retrievable on an immutable blockchain.	Create a light node	Completed	5
		Create a similar form of smart contracts	Completed	2
		Create a service node	Completed	5
		Create a blockchain	Completed	5
		Create a mining function	Completed	5
3	As a user, I want to be able to have an account so that I can be uniquely identified on the network.	Create a database connected to the light node	Completed	3
		Create a login webpage	Completed	2
		Create a register webpage	Completed	2
4	As a user, I want to be able see a list of all users of the application so that when I select a user, I can start 1:1 chatting with anyone of them.	Create a main menu webpage	Completed	2
		Allow users to select a user and enter the chatroom	Completed	4
5	As a user, I want to be able to chat with registered users on the application so that I can talk to my friends.	Create a chatroom webpage	Completed	3
		Enable users to input messages and send them	Completed	5
		Establish a connection to the light node, to receive sent messages	Completed	3
		Establish a connection to the light node, to receive chatroom history	Completed	4

E2.2 Sprint 1 - Developing and testing a client server

Sprint Plan

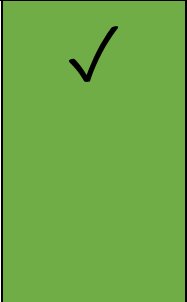
User Story:

1. As a user, I want the application to run on a web interface so that I can access the application via a phone or computer.

Plan of Action

Identifier	User Story	Expected Effort	Acceptance Criteria
1	Create a web interface	4	<ul style="list-style-type: none">• Create client server

Testing of Sprint 1

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Ensure connectivity between the web browser and the client server was possible	Connect to the client server	"hello" would be sent by the client server and received by the web browser	"hello" is received by the web browser	

Testing Sprint Number 1

Hello, is received in the console of the web browser sent from the client server.

```
> hello web browser!
```

E2.3 Sprint 2 - Developing and testing inter-server communications

Sprint plan

User Story:

2. As a user, I want to communicate on a decentralised network that uses blockchain technology so that there is not a central point of failure on the network and so that I don't have to trust a central authority.
 - e. As a user, I want type of smart contract to automate the process of handling messages.
 - f. As a user, I want a digital ledger to hold all my messages sent on the network.
 - g. As a user, I want blocks to be mined before added to the blockchain.

Plan of Action

Identifier	User Story	Expected Effort	Acceptance Criteria
2a	Smart contracts	4	<ul style="list-style-type: none">• Create a light node that enables communications from the web browser to the service node.• Create a similar form of smart contracts that run on the light node.
2b	Digital ledger	5	<ul style="list-style-type: none">• Create a service node to hold the blockchain• Create a blockchain that securely stores user messages.
2c	Mining	5	<ul style="list-style-type: none">• Create mining functionalities on the service node.

Testing of Sprint 2

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Testing the connectivity between the client server and the light node and the response of the light node	Send a message from the client server to the light node	Console log triggered on the light node to display "received" and sends "hello" back to client server	"received" displayed in the console on the light node, and then "hello" displayed in the console on the client server	✓
2	Testing the connection between the light node and the service node	Send a message from the light node to the service node	Console log triggered on the service node displayed "received"	"received" displayed in the console on the service node	✓
3	Test creation of blockchain and genesis block	Start service node	Blockchain created with a genesis block	Blockchain created with a genesis block	✓

Test Number 1

Message is received by the light node and displayed in the console. `received`

Light node sends "hello" back to client server

Client server receives hello from light node and displays hello in console. `hello`

Test Number 2

Service node receives message and displays "received" in console to show that it has been received. `received`

Test Number 3

```
Blockchain {
  chain: [
    Block {
      blockNumber: '',
      timestamp: undefined,
      data: undefined,
      prevHash: '',
      hash: '3bdc1d49f2bdd7096c20eb6c6314adf8ec3b992948db5959e6ca02b86cc92636',
      difficulty: 1,
      nonce: 0
    }
  ],
  difficulty: 1,
  blockTime: 40000,
  transactions: []
}
```

E.2.4 Sprint 3 - Developing and testing logging in and registering functionalities via a SQL database

Sprint Plan

Sprint Number 3

User Story:

3. As a user, I want to be able to have an account so that I can be uniquely identified on the network.
 - a. As a user, I want to be able to sign into an account.
 - b. As a user, I want to be able to register an account.

Plan of Action

Identifier	User Story	Expected Effort	Acceptance Criteria
3a	Log into my account	4	<ul style="list-style-type: none">• Create a login page
3b	Register an account	4	<ul style="list-style-type: none">• Create a database to store all user credentials.• Create a register webpage

Testing of Sprint 3

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	Users can access registration webpage	Press "Registration" button	Registration webpage Is displayed	Registration webpage displayed	✓
2	User inputs for registration are passed to the light node	Username and password on registration webpage	Username and password forwarded to light node	Username and password received by the light node	✓
3	User inputs for login are passed to the light node	Username and password on login webpage	Username and password forwarded to light node	Username and password received by the light node	✓
4	Light node can validate a successful registration using SQL database and store user record	Valid username and password on registration webpage	User record written to user database and valid response returned	User record written to user database and valid response returned	✓

5	Light node can detect an unsuccessful registration using SQL database and store user record	Invalid username or password on registration webpage	User record not written to database and invalid response returned	User record not written to database and invalid response returned	✓
6	Light node can validate a successful login using SQL database	Valid username and password on login webpage	Valid response returned	Valid response returned	✓
7	Light node can detect an unsuccessful login using SQL database	Invalid username or password on login webpage	Invalid response returned	Invalid response returned	✓

Test number 1

Test Number 2

Registration

newUser credentials have been passed to light node and verified. Registration is successful, as indicated by the client server printing the new registered user's username.

```
Register: newUser
```

Test Number 3

H credentials have been passed to the light node and verified. Log in is successful, as indicated by the Client server printing the logged in user's username.

```
User with ID: h
```

Test Number 4

User jack successfully registers an account and now has a user record stored in the SQL database.

```
3 jack $2b$10$EIZDsc2O7JWXWpZSCQTX8uDIpYE9rX... 045a041118136de2234b80611d4bc6ad2b6064... dd51f39
```

Test Number 5

The client server determines a null value, meaning that the registration was invalid.

```
null
```

Test Number 6

User logs in as “h”, which is stored in the database.

```
1 h $2b$10$.0/3VoPYMKhn23eMqQB2fObI/dFgm6a... 04a48233a1b2f3592dbf7866e9167d03f7718e5... da831d:
```

Light node successfully validates the login which is indicated by the h’s username being displayed on the client server.

```
User with ID: h
```

Test Number 7

The client server determines a null value, meaning that the log in was invalid.

```
null
```

E.2.5 Sprint 4 – Developing and testing a main menu webpage

Sprint Plan

User story:

4. As a user, I want to be able see a list of all users of the application so that when I select a user, I can start 1:1 chatting with anyone of them.

Plan of Action

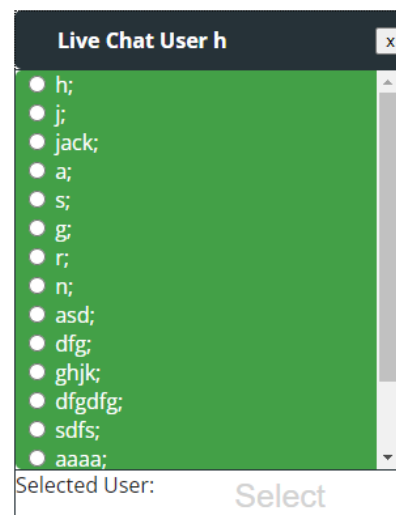
Identifier	User Story	Expected Effort	Acceptance Criteria
4	Display all registered users of the application	3	<ul style="list-style-type: none">• Create a main menu page.• Add a select button to enter the chatroom of the selected user.

Sprint Testing

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	To ensure all users on the SQL database are displayed and can be selected	Valid login	All users in the database are listed and can be selected using a radio button	All users in the database are listed and can be selected using a radio button	✓

Test Number 1

H successfully logs in and is displayed all users of the application on the main menu webpage.



E.2.6 Sprint 5 - Developing and testing chatrooms

Sprint Plan

User Story:

5. As a user, I want to be able to chat with registered users on the application so that I can talk to my friends.
 - a. As a user, I want to be able to enter chatrooms with registered users.
 - b. As a user, I want to be able to send messages in the chatroom.
 - c. As a user, I want to be able to receive messages in the chatroom.
 - d. As a user, I want to be able to see the history of messages that have been sent between the two parties in the chatroom.

Identifier	User Story	Expected Effort	Acceptance Criteria
5a	Create Chatrooms	3	<ul style="list-style-type: none">• Create a chatroom webpage
5b	Send Messages	2	<ul style="list-style-type: none">• Create an input box for messages• Establish a connection to the light node to send messages
5c	Receive Messages	2	<ul style="list-style-type: none">• Establish a connection to the light node to receive messages
5d	View Message History	3	<ul style="list-style-type: none">• Establish a connection to the light node to receive chatroom history

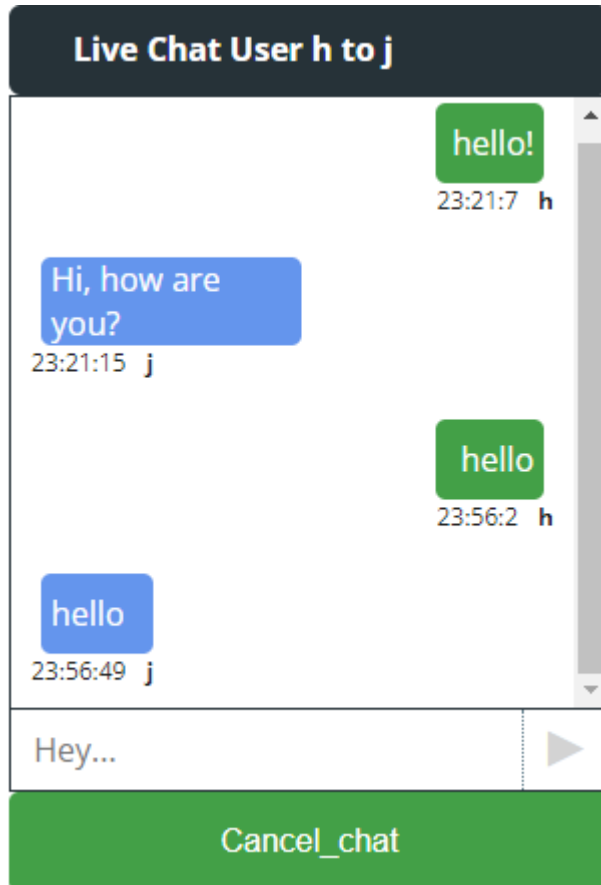
Sprint Testing

Test Number	Purpose	Input	Expected Result	Actual Result	Success?
1	The service node can supply a history of specific conversation	Select user on main menu webpage and enter chatroom	Service node supplies specific history of conversation	Service node supplies specific history of conversation	✓
2	Ensure that the service node can process chatroom messages and respond	Send message in chatroom	Chat message received and displayed in chatroom	Chat message received and displayed in chatroom	✓
3	Ensure that the service node can process chatroom messages and respond	Conversational partner sends message	Chat message received and displayed in chatroom	Chat message received and displayed in chatroom	✓

Test number 1

User enters chatroom, chatroom history is requested by the client server. It is retrieved and sent from the service node to the client server, which is displayed in the chatroom.

```
Conversation { from: 'h', to: 'j', message: 'hello!', time: '23:21:7' }  
Conversation { from: 'j', to: 'h', message: 'Hi, how are you?', time: '23:21:15' }  
Conversation { from: 'h', to: 'j', message: 'hello', time: '23:56:2' }  
Conversation { from: 'j', to: 'h', message: 'hello', time: '23:56:49' }
```



Test Number 2

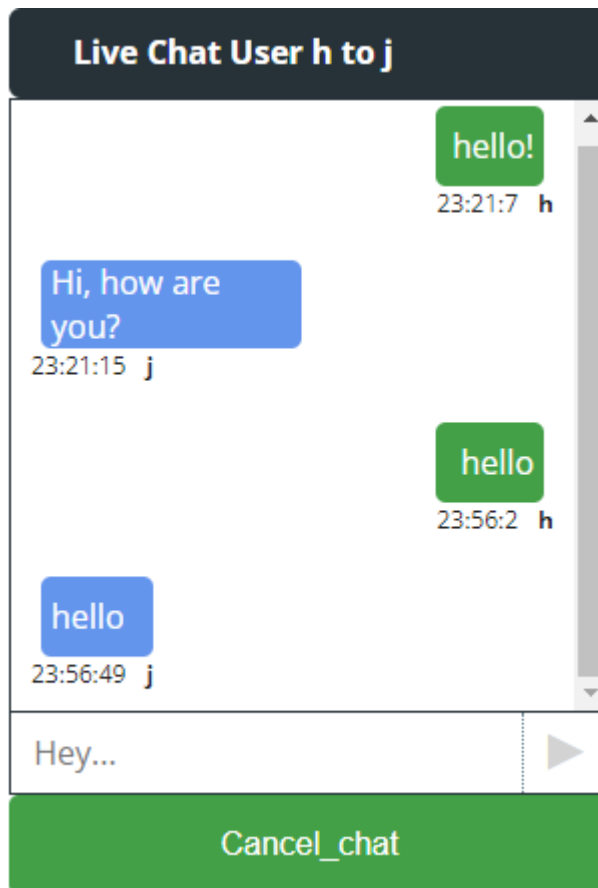
J sends H "hello". Client server sends textMessage to light node. Light node receives messages, creates transaction out of received textMessage, digitally signs it and sends it to the service node.

```
Sending Message {  
  type: 'TYPE_CREATE_TRANSACTION',  
  data: Transaction {  
    from: '04b91106f52d209d34fbd8c8908a9d6280bb7b233e1f54559ec43202bc8941b  
8e820c1aa2d97b563a969c7bdace5b140407b852aa4d8703d949572953c990b90440',  
    to: '04a48233a1b2f3592dbf7866e9167d03f7718e5e4e8a3c31f445da376f895704  
e48bdf20ec485892e9d4a8a2151d488bcbcf46e596211303297d0f112751cb29a3',  
    textMessage: { from: 'j', to: 'h', message: 'hello', time: '23:56:49' },  
    signature: '304502210096d672b0d74a13125b6632a702454dd92e66f445c0ea  
972baec8288f931678c1022041a8b903bfac28e9e8ce6172486d509683cd11936263d101e  
02295a445cb6bf1'  
  }  
}
```

Transaction is received by service node, which verifies the signature, adds it the transaction pool, mines the block and adds it to the blockchain.

```
MINER.JS CURRENTLY MINING!!
{
  blockNumber: 5,
  timestamp: 1681772210362,
  data: [
    {
      from: '04b91106f52d209d34fbdc8908a9d6280bb7b233e1f54559ec43202bc8941b8e820c1aa2d97b563a969c7bdace5b140407b852aa4d8703d949572953c990b90440',
      to: '04a48233a1b2f3592dbf7866e9167d03f7718e5e4e8a3c31f445da376f895704e48bdF20ec485892e9d4a8a2151d488bcbcf46e596211303297d0f112751cb29a3',
      textMessage: [Object],
      signature: '304502210096d672b0d74a13125b6632a702454dd92e66f445c0ea972baec8288f931678c1022041a8b903bfac28e9e8ce6172486d509683cd11936263d101e02295a445cb6bf1',
    }
  ],
  prevHash: '00003bd1067cd01c37f0136b01dc60f294d6c53b1b38fc88b636358cba9372e2',
  hash: '0000a5eef91d4f7db980717c16b87f160873045ac2e5cbd77bce2c9311a3fb66',
  difficulty: 1,
  nonce: 13148
}
```

Newly mined block is sent back to the light node and is displayed in chatroom.



Test Number 3

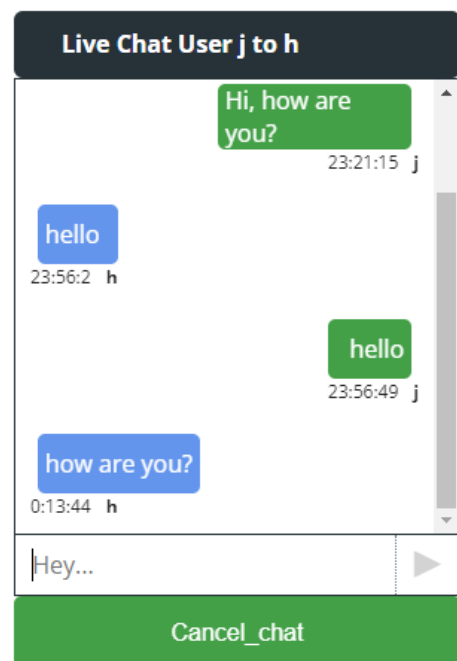
H sends J “how are you?”. Client server sends textMessage to light node. Light node receives messages, creates transaction out of received textMessage, digitally signs it and sends it to the service node.

```
Sending Message {
  type: 'TYPE_CREATE_TRANSACTION',
  data: Transaction {
    from: '04a48233a1b2f3592dbf7866e9167d03f7718e5e4e8a3c31f445da376f895704e48bdf20ec485892e9d4a8a2151d488bcbcf46e596211303297d0f112751cb29a3',
    to: '04b91106f52d209d34fbdc8908a9d6280bb7b233e1f54559ec43202bc8941b8e820c1aa2d97b563a969c7bdace5b140407b852aa4d8703d949572953c990b90440',
    textMessage: { from: 'h', to: 'j', message: 'how are you?', time: '0:13:44' },
    signature: '30440220734a2cc6ae48c8509bd67f0bb1cd1fe37f76134a085bbbe3d5d6fe2364cd5c470220617c63d6d716d1c72a425f5abcf1691967a05f71dbec71e28ab7b3dda5096338'
  }
}
```

Transaction is received by service node, which verifies the signature, adds it the transaction pool, mines the block and adds it to the blockchain.

```
MINER.JS CURRENTLY MINING!!
{
  blockNumber: 6,
  timestamp: 1681773224389,
  data: [
    {
      from: '04a48233a1b2f3592dbf7866e9167d03f7718e5e4e8a3c31f445da376f895704e48bdf20ec485892e9d4a8a2151d488bcbcf46e596211303297d0f112751cb29a3',
      to: '04b91106f52d209d34fbdc8908a9d6280bb7b233e1f54559ec43202bc8941b8e820c1aa2d97b563a969c7bdace5b140407b852aa4d8703d949572953c990b90440',
      textMessage: [Object],
      signature: '30440220734a2cc6ae48c8509bd67f0bb1cd1fe37f76134a085bbbe3d5d6fe2364cd5c470220617c63d6d716d1c72a425f5abcf1691967a05f71dbec71e28ab7b3dda5096338'
    }
  ],
  prevHash: '0000a5eef91d4f7db980717c16b87f160873045ac2e5cbd77bce2c9311a3fb66',
  hash: '000019a1884aab8a3f632b2972570cb4516fc40b94b58bd09a55c4854bd5d2d0',
  difficulty: 1,
  nonce: 18907
}
```

Newly mined block is sent back to the light node and is displayed in chatroom.



Appendix F

Google Drive Folder:

https://drive.google.com/drive/folders/1deancYVXW1_2gbxHB6_p55kwgXAtRu-k

9 Glossary

Blockchain:

A distributed digital ledger that records all transactions sent across the network, which is stored across a network of service nodes in a secure and tamper-proof manner.

Cryptography:

The process of utilizing mathematical methods to protect communication from unauthorized entities or alteration.

Decentralisation:

A system where decisions are democratic as opposed to a dictatorship by a single controlling party.

Distributed Ledger:

A database dispersed across several service nodes, each containing a copy of the same database and can mine and add blocks to the database.

Hash Function:

An algorithm that transforms inputted data into a fixed-length string of characters.

Immutable:

Data that, once posted to the blockchain, cannot be changed, or altered.

Mining:

The process carried out by the service node in which it solves complex mathematical problems to create a hash value for the block that starts with the same amount of 0s as the mining difficulty of the blockchain.

Service Node:

A computer that maintains a copy of the blockchain database and is a component of a blockchain network.

Light Node:

A computer that relays authentication and registration packets from the web browser to the SQL Server and relays messages sent from the web browser to the service node. It also listens for newly added blocks of the blockchain, using a form of smart contract to retrieve and disperse messages to client servers to relevant users.

Peer-to-Peer (P2P):

A network architecture where participants communicate directly with each other, without the need for intermediaries, such as a central server.

A similar form of smart Contract:

A self-executing contract stored on the light node that automates the process and handling of transactions with the terms and agreements written in code.

Block:

Stores a blocks index, timestamp of when created, transaction, previous blocks hash, hash of current block, mining difficulty and nonce.

Transaction:

A message exchange between two register parties of the application, which is recorded in a block.

Authentication Packet:

Holds a username and a hashed password and is sent to the SQL database to verify the credentials of a user.

Registration Packet:

Holds a username and a hashed password and is sent to the SQL database to verify that the credentials are valid.

Public Key:

A fixed length string that is used to identify users on the blockchain and authenticate transactions on the blockchain.

Private Key:

A fixed length string that is used to sign transactions to be stored on the blockchain.

Nonce:

An integer that is used in mining to alter the result of a hashed block.

Mining Difficulty:

The number of 0s a hash value for a block needs to begin with.

General Data Protection Regulation (GDPR):

An EU regulation that governs privacy and data protection in the EU and European Economic Area.

British Computing Society (BCS):

Establishes the UK's professional standards for competence, behaviour, and ethical practise in computers.

Computer Misuse Act:

A law in the United Kingdom that criminalizes unauthorized access to computer systems and the misuse of computer systems.

Intellectual Property: A law put in place to stop people from stealing and copying people's online property.

Trustful Environment:

Apparent in centralised networks, in which you must trust the central authority controlling the network.

Trustless Environment:

Enables all participants to agree on a single truth using a consensus mechanism without the requirement for a single overarching authority or for them to know or trust one another.

Web2:

Applications that place a strong emphasis on end-user interoperability, user-generated content, and ease of use.

Web3:

A set of decentralised programmes that are open-source and networked by driving blockchain computing architecture.

Digital Signature:

An electronic, encrypted authentication signature for digital data.

Whisper Protocol:

A set of protocols that permits user communication over the same network as the blockchain. (Used by Status)

dApp:

Decentralised Application

Double Ratchet Algorithm:

Based on a shared secret key and utilised by two parties to send encrypted messages.

Symmetric Cryptography:

A single shared secret is used to communicate encrypted data between a party's secret key, which two parties use to exchange encrypted messages.

Asymmetric Cryptography (Public Key Cryptography):

A procedure that encrypts and decrypts messages while safeguarding them from unauthorised access or usage using a pair of related keys—one public key and one private key.

Proof of Work:

A consensus mechanism that provides cryptographic proof in which a party has proved to other parties that a certain amount of computational power has been used.

Proof of Stake:

A consensus mechanisms used by a decentralised network to come to an agreement together on a transaction's validity.

SHA256 Hash Algorithm:

Produces an irreversible and almost unique 256-bit long hash of an input string.