Freelancing System based on Blockchain

IT5613 SOCIALLY RELEVANT PROJECT LABORATORY

A PROJECT REPORT

Submitted by

BALASUBRAMANIAM N (2020506020)

BALA NATESH R M (2020506018)

JAWAHAR A S (2020506035)

SARUKESH S (2020506084)

DEPARTMENT OF INFORMATION TECHNOLOGYMADRAS INSTITUTE OF TECHNOLOGY CAMPUSANNA UNIVERSITY, CHENNAI-600 044.

JANUARY 2023 - JUNE 2023

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report **"FREELANCING SYSTEM BASED ON BLOCKCHAIN"** is the bonafide work of **BALASUBRAMANIAM N (2020506020), BALA NATESH R M (2020506018), JAWAHAR A S (2020506035), and SARUKESH S (2020506084)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M R SUMALATHA	**Dr. M R SUMALATHA**
HEAD OF DEPARTMENT	**SUPERVISOR**
Professor	Professor
Department of Information Technology	Department of Information Technology
MIT Campus, Anna University	MIT Campus, Anna University
Chennai – 600044	Chennai **-** 600044

ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement made us accomplish this project.

We express our thankfulness to our project supervisor **Dr. M. R. Sumalatha**, Professor, MIT Campus, for providing project guidance and encouragement to do the project.

We express our gratitude and sincere thanks to our respected Dean of MIT Campus, **Dr. J Prakash**, for providing computing facilities.

BALASUBRAMANIAM N	**2020506020**
BALA NATESH R M	**2020506018**
JAWAHAR A S	**2020506035**
SARUKESH S	**2020506084**

ABSTRACT

This project presents a novel freelancing system implemented using blockchain technology. The system allows users to post jobs and enables multiple individuals to bid on those jobs. Upon completion of the bidding phase, the job poster can accept a bid, assigning the job to the winning bidder. A chat application is integrated into the website, leveraging the power of blockchain for secure communication. Each message exchanged within the chat is treated as a transaction within the blockchain network. Additionally, the system facilitates monetary transactions, allowing users to send money to the individuals they are engaging with in the chat application. The blockchain-based implementation ensures transparency, security, and reliability throughout the freelancing process, providing an efficient and trustworthy platform for job allocation and communication. Furthermore, the freelancing system incorporates a user-friendly interface that simplifies the job posting and bidding process, enhancing the overall user experience. Users can easily navigate through various pages, such as the "About" section, which provides comprehensive information about the system's features, benefits, and guidelines.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
API	Application Programming Interface
JSON	JavaScript Object Notation
XML	Extensible Markup Language
IPFS	InterPlanetary File System
MERN	MongoDB, Express.js, React.js, Node.js
ODM	Object Data Modeling
CRUD	Create Read Update Delete
UI	User Interface
SPDX	Software Package Data Exchange
SEO	Search Engine Optimization
PWA	Progressive Web App

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

This project introduces a revolutionary freelancing system that harnesses the power of blockchain technology to address the pressing issue of unemployment amidst a rapidly rising global population and an increasingly competitive world. The system provides a platform where individuals can post job opportunities, inviting bids from a diverse pool of talented freelancers. Through a transparent and decentralized process, the person who posted the job has the autonomy to evaluate and accept bids, effectively assigning the job to the most suitable candidate.

To facilitate seamless communication and collaboration, the system incorporates a sophisticated chat application within the website. Notably, the chat application utilizes blockchain technology, treating each message as a transaction on the blockchain network. This ground-breaking approach ensures the integrity, security, and immutability of all conversations, fostering trust and transparency between users.

Furthermore, the system offers an integrated payment feature, enabling users to securely transfer funds to the individuals they are engaging with via the chat application. By seamlessly combining communication and financial transactions, the platform streamlines the freelancing process, providing a comprehensive solution to both job allocation and payment settlement.

Ultimately, this blockchain-based freelancing system aims to revolutionize the labour market by providing an efficient, transparent, and secure platform for job seekers and employers. By harnessing the potential of blockchain technology, it offers a viable solution to the challenges posed by unemployment in a rapidly evolving and competitive world, empowering individuals to connect, collaborate, and thrive in the digital economy.

1.2 RESEARCH CHALLENGES

Implementing a freelancing system based on blockchain technology presents several research challenges. Scalability is a key concern, necessitating exploration of efficient consensus mechanisms and network architectures. Balancing transparency with privacy requires innovative approaches to ensure confidentiality while maintaining transparency. User experience must be optimized to enhance adoption, reducing transactional friction. Establishing fair governance models and efficient dispute resolution mechanisms is crucial for credibility. Security and smart contract auditing are vital to identify vulnerabilities and protect against malicious activities. Integrating blockchain with traditional systems also poses challenges in terms of interoperability and compatibility.

1.3 OBJECTIVE

The objective of this project is to develop a blockchain-based freelancing system that enables job posting, bidding, chat communication, and secure financial transactions, aiming to address unemployment challenges in a rapidly growing population and competitive world.

1.4 SCOPE OF THE PROJECT

The scope of this project encompasses the development and implementation of a comprehensive blockchain-based freelancing system. This includes creating a user-friendly platform for job posting, bidding, and chat communication, leveraging blockchain technology for secure and transparent messaging. Additionally, the scope extends to integrating a secure payment system to facilitate financial transactions between users. The project aims to provide a holistic solution to address unemployment issues while ensuring efficiency, transparency, and trust in the freelancing process.

1.5 CONTRIBUTION

The project contributes to address the challenges of unemployment by creating a freelancing system based on blockchain technology. Users can post jobs and receive bids, while a chat application powered by blockchain ensures secure communication. Financial transactions can also be conducted within the platform. The goal is to leverage blockchain's transparency and security to foster trust between freelancers and employers. The system's effectiveness will be evaluated through real-world scenarios, assessing its potential to reduce unemployment in a competitive world with a rapidly growing population.

1.6 BLOCKCHAIN

Blockchain is a transformative technology that has gained significant attention and adoption across various industries. At its core, blockchain is a decentralized and distributed digital ledger that records transactions in a transparent, immutable, and secure manner. Unlike traditional centralized systems, blockchain eliminates the need for intermediaries by relying on a network of computers, or nodes, to validate and store transactions. This distributed nature ensures consensus and prevents tampering or fraud. The key characteristics of blockchain include transparency, as all participants can view the entire transaction history, immutability, as transactions are recorded in blocks that are linked and secured through cryptographic algorithms, and security, as data is stored across multiple nodes, making it highly resistant to hacking or unauthorized modifications.

1.7 GANACHE

Ganache is a popular development tool and personal blockchain emulator used in Ethereum development. It provides a local testing environment for developers to simulate Ethereum networks, allowing them to deploy, interact with, and debug smart contracts seamlessly. With Ganache, developers can create multiple accounts, simulate transactions, and test their applications without the need for a live blockchain network. It offers a user-friendly interface, detailed transaction logs, and robust debugging capabilities, making it a valuable tool for building and testing decentralized applications (DApps) efficiently. Ganache significantly simplifies the development process by providing a reliable and accessible environment for Ethereum developers to iterate and refine their smart contracts and DApps.

1.8 METAMASK

Metamask is a widely used browser extension that serves as a cryptocurrency wallet and a gateway to decentralized applications (DApps) on the Ethereum blockchain. It allows users to securely store and manage their Ethereum-based digital assets, such as Ether and ERC-20 tokens, while providing a seamless interface for interacting with DApps. Metamask acts as a bridge between the user's web browser and the Ethereum network, enabling easy integration of blockchain functionalities into regular web applications. It provides a user-friendly interface for transactions, contract interactions, and identity management, making it a valuable tool for both developers and end-users navigating the decentralized ecosystem.

1.9 ETHEREUM

Ethereum is a decentralized, open-source blockchain platform that enables the development and execution of smart contracts and decentralized applications (DApps). It introduced the concept of programmable blockchain, allowing developers to create and deploy their own applications on top of the Ethereum network. Ethereum utilizes its native cryptocurrency, Ether (ETH), as a means of value exchange and to incentivize network participants. It pioneered the concept of Initial Coin Offerings (ICOs) and has become a prominent platform for token creation and crowdfunding. With its Turing-complete programming language, Solidity, Ethereum has fostered innovation in various sectors, including finance, supply chain management, gaming, and decentralized finance (DeFi).

1.10 FREELANCE

Freelancing refers to the practice of offering services or skills on a project basis, typically as an independent contractor or self-employed individual. Freelancers work for multiple clients on a flexible schedule, providing services ranging from graphic design, writing, programming, marketing, and more. They enjoy the freedom to choose their projects, set their rates, and work remotely, offering convenience and autonomy. Freelancing has gained popularity due to advancements in technology, connectivity, and the rise of online platforms that connect freelancers with clients globally. It provides opportunities for individuals to showcase their expertise, earn income, and enjoy a flexible lifestyle while catering to the evolving demands of the job market.

1.11 WEB3

Web3 refers to the next generation of the internet, characterized by the integration of blockchain technology and decentralized protocols. It aims to shift the power dynamics of the internet from centralized authorities to individual users. Web3 enables users to have more control over their data, identities, and digital assets. It leverages blockchain's transparency, security, and immutability to create decentralized applications (DApps) that operate without intermediaries. Web3 also encompasses the concept of self-sovereign identity, where users have ownership and control over their personal data. This paradigm shift holds the potential to foster greater privacy, security, and user empowerment in the digital realm.

1.12 HTTP PROTOCOL

The Hypertext Transfer Protocol (HTTP) is an application layer protocol that governs the communication and transfer of data between web servers and web browsers. It serves as the foundation for the World Wide Web, enabling the retrieval of web pages, images, videos, and other resources. HTTP functions on a client-server model, where the client, typically a web browser, sends requests to the server, and the server responds with the requested data. HTTP is based on a stateless model, meaning each request-response cycle is independent of previous ones. It employs various methods such as GET, POST, PUT, DELETE, etc., to facilitate different types of interactions and data manipulation between clients and servers.

1.13 REST API

REST (Representational State Transfer) API, is an architectural style for designing web services that allows communication between client and server over HTTP. It is based on a set of principles that emphasize simplicity, scalability, and interoperability. REST APIs are stateless, meaning each request from the client to the server contains all the necessary information for the server to process and respond. They utilize HTTP methods like GET, POST, PUT, DELETE to perform different operations on resources. REST APIs often return data in a common data format such as JSON or XML, making it easier for different clients to consume and interact with the API.

1.14 MERN STACK

The MERN stack is a popular web development stack that combines four key technologies: MongoDB, Express.js, React.js, and Node.js. MongoDB is a NoSQL database that allows for flexible and scalable data storage. Express.js is a web application framework for Node.js that simplifies server-side development. React.js is a JavaScript library for building user interfaces, providing a fast and interactive front-end experience. Node.js is a runtime environment that allows server-side JavaScript execution. Together, these technologies form a full-stack development environment that enables developers to create modern, efficient, and scalable web applications. The

MERN stack is particularly well-suited for creating single-page applications (SPAs) and real-time applications.

1.15 TRUFFLE

Truffle is a popular development framework for building decentralized applications (DApps) on the Ethereum blockchain. It provides a suite of tools and utilities that simplify the entire development lifecycle, from smart contract creation and testing to deployment and management. With Truffle, developers can write smart contracts in Solidity and easily compile, migrate, and interact with them using a standardized development workflow. Truffle also offers a testing framework that enables comprehensive and automated testing of smart contracts. Additionally, it integrates with popular blockchain networks and provides a development console for interactive contract debugging. Truffle's robust features and seamless integration with the Ethereum ecosystem make it a valuable tool for developers looking to build secure and scalable DApps.

1.16 SOLIDITY

Solidity is a high-level programming language designed specifically for writing smart contracts on blockchain platforms, with Ethereum being the most prominent one. It offers a syntax similar to JavaScript and is optimized for executing on the Ethereum Virtual Machine (EVM). Solidity enables developers to define the behavior and rules of decentralized applications (DApps) by creating self-executing contracts with predefined conditions and actions. It supports features like inheritance, libraries, and complex data structures, empowering developers to build sophisticated and secure smart contracts. Solidity also provides access to Ethereum's native functionality, including interacting with other contracts, managing digital assets, and handling events. To ensure code reliability and security, Solidity supports comprehensive testing and debugging capabilities. Its widespread adoption and rich ecosystem of tools and frameworks make Solidity an essential language for developing decentralized applications on the Ethereum blockchain.

1.17 ORGANIZATION OF THE THESIS

The thesis is structured as follows: Chapter 2 provides a comprehensive literature survey on video streaming, blockchain-based approaches for freelance systems, and chat applications. In Chapter 3, the architecture and system design of the proposed solution are modeled, outlining the system's architectural components and design principles. Chapter 4 delves into the implementation details, discussing the specifications and the environment in which the system was developed. The achieved results are presented in Chapter 5. Finally, Chapter 6 concludes the thesis, summarizing the findings and suggesting potential directions for future research in the field.

CHAPTER 2
LITERATURE SURVEY

2.1 DECENTRALIZATION APPROACH

The implementation of peer-to-peer transactions in the commercial sector has shown promise, especially in the growing freelance economy. With over 10 million freelancers, India has emerged as the largest market for freelance work. However, centralized freelancing websites present challenges as they can be manipulated, and both freelancers and employers rely on third parties for payment contracts. To address these issues, a proposed solution is a decentralized system based on the public Blockchain Ethereum (Prathmesh Deshmukh et al., 2018). This system facilitates peer-to-peer transactions using cryptocurrency and incorporates peer-to-peer reviews on the distributed ledger of Ethereum. By leveraging the transparency and security of blockchain technology, the proposed system aims to mitigate the problems associated with third-party dependencies in the freelance ecosystem.

2.2 SECURE MESSAGING

Organizations face challenges in managing the boundaries between personal and business communications, leading to security risks from unsecured applications used in the workplace. Lack of awareness about employee device usage poses additional concerns. Securing critical business communication involving trade secrets and strategic decisions is a challenge, as publicly available platforms lack regulation, tracking, and scalability. This raises data protection risks across industries. Instant Messaging applications raise privacy and security concerns. The research aims to develop a secure chat application for enterprise-level communication, leveraging blockchain's decentralized model (U. P. Ellewala et al., 2020). This solution overcomes drawbacks of traditional messaging apps, ensuring confidentiality, integrity, availability of official data, and advanced auditing features.

2.3 SECURE PAYMENT

Current blockchain relay schemes require the immediate validation of each relayed block header by the destination blockchain. This leads to high operating cost when deploying these relays between Ethereum-based blockchains where validating block headers on-chain is computationally expensive. To overcome these limitations, a novel relay scheme was introduced so that it employs a validation-on-demand pattern combined with economic incentives to reduce the cost of operating a relay between Ethereum-based blockchains by up to 92% (P. Frauenthaler et al., 2020). With this relay scheme, decentralized interoperability between blockchains like Ethereum and Ethereum Classic becomes feasible.

.

2.4 FREELANCING MARKETPLACE

Freelancing marketplace is a site or platform that connects two parties in processing service transactions at an hourly rate or per project. A conventional freelancing marketplace is a place for freelancers to find work and transact digitally. This study aims to propose a prototype of a freelancing marketplace system that is distributed and decentralized, secure, and transparent using smart contract- based blockchain technology. The method used in this study is a prototype which is a fast method of developing a software system. The developed prototype is a system that is based on the Ethereum public blockchain network, utilizes a smart contract mechanism in its transaction activities, and use IPFS in the storage and sharing of documents on it (Irawan Afriantoet et al., 2021). According to the findings of the research, transaction data input in the freelancing marketplace system prototype environment can be executed by smart contracts and saved on the blockchain network, indicating that the transaction data will be stored more securely, tamper proof, and transparent.

2.5 AUCTION BASED SYSTEM

Auction-based mechanisms are extremely relevant in modern day electronic procurement systems since they enable a promising way of automating negotiations with suppliers and achieve the ideal goals of procurement efficiency and cost minimization. This paper surveys recent research and current art in the area of auction-based mechanisms for e-procurement. The survey delineates different representative scenarios in e-procurement where auctions can be deployed and describes the conceptual and mathematical aspects of different categories of procurement auctions. Three broad categories were discussed: 1) single-item auctions: auctions for procuring a single unit or multiple units of a single homogeneous type of item; 2) multi-item auctions: auctions for procuring a single unit or multiple units of multiple items; and 3) multiattribute auctions where the procurement decisions are based not only on costs but also on attributes, such as lead times, maintenance contracts, quality, etc. In our review, we present the mathematical formulations under each of the above categories, bring out the game theoretic and computational issues involved in solving the problems, and summarize the current art.

2.6 BLOCKCHAIN INTEGRATION

The focus is on designing and implementing a blockchain infrastructure for efficient tracking and tracing in a supply chain. While the paper primarily discusses the "Blockchain Data Management" (BCDM) design pattern and its integration into a supply chain infrastructure, it does not explicitly delve into the integration of blockchain with React, a JavaScript library for building user interfaces.

However, when considering the integration of blockchain with React in the context of the research paper's objectives, React can be utilized for developing the user interface components of the supply chain application. React's component-based architecture and reusability make it well-suited for creating interactive and responsive user interfaces that can interact with the blockchain backend.

The integration can be achieved by utilizing web3 libraries or frameworks like Web3.js, which provide APIs for interacting with the blockchain network. React components can leverage these APIs to fetch and display blockchain data, such as tracked and traced information from the supply chain. Additionally, React's state management capabilities can be employed to handle user interactions, trigger blockchain transactions, and update the user interface accordingly.

Overall, integrating blockchain with React allows for the development of a user-friendly interface that seamlessly interacts with the underlying blockchain infrastructure, facilitating efficient tracking and tracing in the supply chain.

2.7 SUMMARY OF THE LITERATURE SURVEY

Thus, to solve this problem of having a decentralized approach along with a chat application remains a challenge. Blockchain integration on an auction based system coupled with auction seems a problem that is yet to be solved. Also, importantly how can a system be decentralized. The Blockchain network is used to make the system decentralized. The findings emphasize the importance of implementing a decentralized architecture for enhanced security and trust in the freelancing platform. Integrating blockchain technology enables secure messaging and payment transactions, ensuring transparency and immutability.

CHAPTER 3
SYSTEM ARCHITECTURE AND DESIGN

3.1 SYSTEM ARCHITECTURE

The system architecture of the blockchain application is designed to provide a robust and scalable platform for conducting auctions and facilitating secure communication. The architecture follows a full stack MERN (MongoDB, Express.js, React.js, Node.js) approach. On the backend, the Express.js framework handles the API routes and business logic related to the auction functionality. It communicates with the MongoDB database to store and retrieve data associated with auctions, user profiles, and transaction history. The backend also includes the necessary middleware to interact with the blockchain network, such as the Web3.js library for Ethereum integration.

The React.js library is utilized to build the frontend, which offers an intuitive and interactive user interface for participants to engage in the auction process. The frontend communicates with the backend API to fetch and display auction listings, user profiles, and real-time updates. In addition to the auction functionality, the architecture includes a separate React application for the chat feature. This chat application allows users to communicate securely within the platform. The chat application is integrated with Metamask, which provides a secure connection to the Ethereum network. This integration enables secure messaging and facilitates transactions within the chat application, ensuring trust and transparency.

To ensure the integrity and fairness of the chat process, smart contracts are compiled, migrated, and deployed onto the blockchain network. These contracts contain the rules and conditions for messages and transactions. The contracts are executed autonomously on the blockchain, eliminating the need for a centralized authority and ensuring a trustless environment.

Overall, the system architecture combines the power of the MERN stack, Metamask integration, and blockchain technology to create a comprehensive platform for conducting auctions securely and facilitating transparent communication between participants. The architecture ensures the reliability, scalability, and integrity of the auction platform, providing a seamless user experience and fostering trust in the freelancing ecosystem.

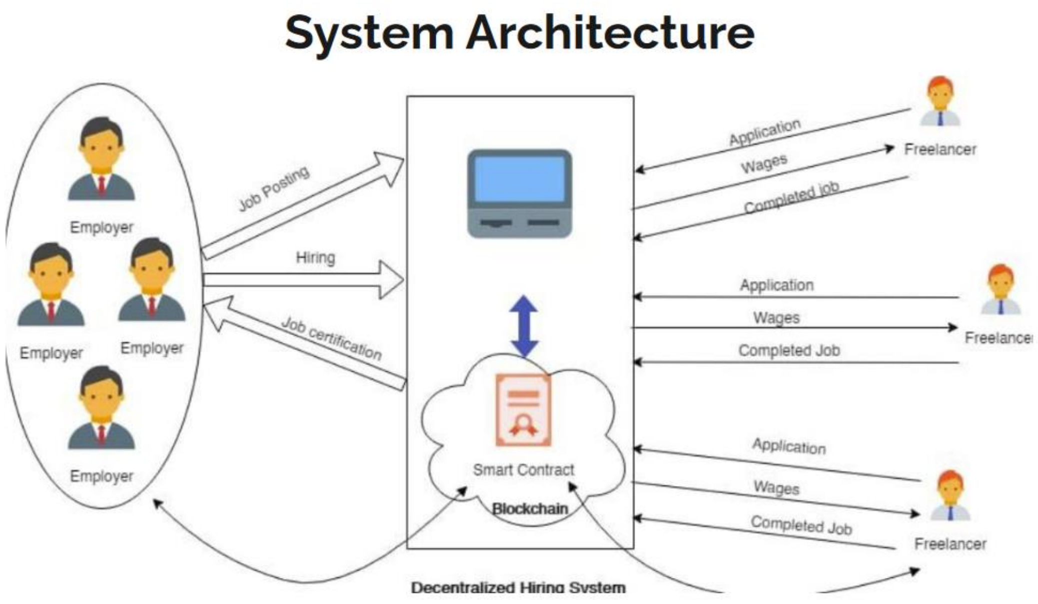


Figure 3.1 – System Architecture

3.2 AUCTION

The auction part of the blockchain application is a key feature that enables users to participate in and conduct auctions in a secure and transparent manner. The auction functionality is implemented. Participants can list items for auction, specify starting bids, and set the duration of the auction. Other users can then place bids on the listed items, with each bid being recorded on the website. The smart contract handles bid validation, ensuring that only valid bids are accepted. The auction process is transparent, as all bids and related information are publicly visible on the network.

3.2.1 Auction Model

The auction model is defined using Mongoose, which is an Object Data Modeling (ODM) library for MongoDB and Node.js. The auction model defines the structure and properties of an auction listing.

The auctionSchema is created using the mongoose.Schema constructor. It includes various fields that represent different aspects of the auction: mail, jobTitle, jobExpRequired, jobDescription, jobTitle, baseAmount, available, bidders, blockchainAddress. The { timestamps: true } option ensures that the createdAt and updatedAt timestamps are automatically added to the auction document.

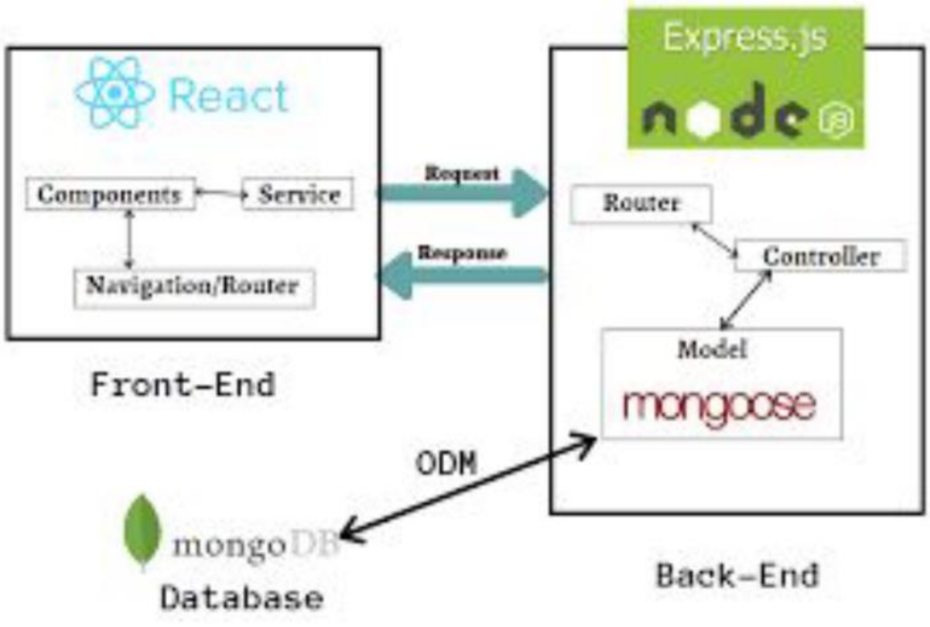
The Auction model is created using mongoose.model, which compiles the schema into a model and allows performing CRUD operations on the "Auction" collection in MongoDB.Finally, the Auction model is exported to be used in other parts of the application. In summary, the auction model defines the structure of an auction listing, including details such as the job title, description, starting bid, bidders' information, and blockchain address associated with the auction.

3.2.2 Auction Routes

First, the required dependencies and controller functions are imported. The controller functions handle the logic for creating auctions, retrieving auctions, bidding on auctions, closing auctions, and managing user-specific auctions and bids. The router is created using express.Router(), which allows defining the routes for the auction-related functionality.

3.3 TECHNICAL ARCHITECTURE

The project follows the MERN stack architecture, which consists of MongoDB, Express, React, and Node.js. MongoDB serves as the database to store auction-related information. Express is used as the web application framework for building the backend server and handling HTTP requests. Node.js provides the runtime environment for executing JavaScript code on the server-side. React is utilized for developing the frontend user interface, enabling the creation of dynamic and interactive components. Together, these technologies allow for a seamless integration of the different layers of the application, facilitating efficient data management, server-side logic, and responsive user interfaces.



3.4 BLOCKCHAIN ARCHITECTURE

The architecture involves the use of Web3, React, and a smart contract deployed on the Ethereum blockchain. The architecture follows a client-server model, where the client is the React application running in the user's browser, and the server is the Ethereum blockchain. When the Chat component is mounted, it goes through the initialization process by calling various functions. These functions handle tasks such as loading the Web3 library, connecting to an Ethereum node (using either MetaMask or a local node), retrieving the user's Ethereum accounts, fetching the user's account balance, and loading the chat smart contract.

The smart contract is loaded using the contract's ABI (Application Binary Interface) and the network ID to identify the contract's address on the specific Ethereum network. The architecture also includes several event listeners, which are set up to listen for specific events emitted by the smart contract. These events include message sent events, Ether sent events, Ether ask events, and messages fetched events. When these events occur on the blockchain, corresponding functions are called to update the UI and provide real-time updates to the user.

The React component maintains various state variables, such as the chat messages, user accounts, account balances, and transaction-related information. These state variables are updated by event handlers and UI update functions.he UI elements are rendered based on the state variables, including the chat messages, sender and recipient addresses, transaction counts, and balances. Users can send messages by entering text in the input field, and the application triggers the smart contract function to send the message to the intended recipient. Users can also send Ether by entering a specific command in the input field, and the application interacts with the smart contract to execute the transfer.

Overall, this blockchain architecture leverages Web3, React, and Ethereum smart contracts to create a decentralized chat application that allows users to communicate and send cryptocurrency securely and transparently on the Ethereum blockchain.

CHAPTER 4
ALGORITHM DEVELOPMENT AND IMPLEMENTATION

4.1 BLOCKCHAIN APPROACH

The blockchain approach used in the code above is a decentralized and transparent architecture based on the Ethereum blockchain. It utilizes smart contracts to handle the logic of the chat application and relies on Web3 and React for interacting with the blockchain and building the user interface. This approach allows for secure messaging and the transfer of cryptocurrency (Ether) directly between users without the need for a centralized server. The blockchain ensures data integrity, immutability, and transparency, while the smart contract enforces the rules and functionalities of the chat application.

4.2 MIGRATION CONTRACT

1. Declare the contract name as "Migrations."
2. Declare two state variables:
    2.1. "owner": an address variable that will store the owner's address.
    2.2. "last_completed_migration": an unsigned integer variable that will store the last completed migration.
3. Define a constructor function that is executed once during contract deployment:
    3.1. Set the value of the "owner" variable to the address of the message sender (the account that deployed the contract).
4. Define a modifier named "restricted":
    4.1. Check if the message sender is the owner of the contract.
    4.2. If the condition is met, proceed with the execution of the function containing this modifier.

5. Define a function named "setCompleted" that takes a parameter "completed" of type uint (unsigned integer) and is publicly accessible:
   5.1. Apply the "restricted" modifier to this function to ensure that only the contract owner can execute it.
   5.2. Set the value of "last_completed_migration" to the provided "completed" value.
6. Define a function named "upgrade" that takes a parameter "new_address" of type address and is publicly accessible:
   6.1. Apply the "restricted" modifier to this function to ensure that only the contract owner can execute it.
   6.2. Create a new instance of the "Migrations" contract at the specified "new_address."
   6.3. Call the "setCompleted" function of the new instance, passing the value stored in "last_completed_migration" of the current contract.

4.3 CHAT CONTRACT

1. Import the necessary library for experimental features and specify the SPDX license identifier.
2. Define the contract name as "ChatApp."
3. Declare a mapping named "messages" to store messages between two addresses:
   3.1. The mapping has two levels: the first level maps "address" (sender) to another mapping.
   3.2. The second level maps "address" (recipient) to an array of "Message" structs.
4. Define a struct named "Message" to represent a single message:
   4.1. It contains two properties: "message" of type string and "from" of type address, representing the message content and sender address, respectively.
5. Define several event declarations to emit specific events during contract execution:

5.1. "messageSentEvent" to notify when a message is sent.
5.2. "etherSentEvent" to notify when Ether is sent from one address to another.
5.3. "etherAskEvent" to notify when an address requests Ether.
5.4. "messagesFetchedEvent" to notify when messages are fetched.

6. Define a function named "sendMsg" to send a message to another address:
   6.1. Takes two parameters: "to" of type address (recipient's address) and "message" of type string (message content).
   6.2. Adds the message to the "messages" mapping for both the sender and recipient addresses.
   6.3. Emits the "messageSentEvent" with the sender, recipient, and message details.
7. Define a function named "sendEther" to send Ether to another address:
   7.1. Takes one parameter: "to" of type address payable (recipient's address).
   7.2. Sends the specified amount of Ether (received with the transaction) to the recipient address.
   7.3. Emits the "etherSentEvent" with the sender, recipient, and success status of the Ether transfer.
   7.4. Throws an exception and reverts the transaction if the Ether transfer fails.
8. Define a function named "askEther" to request Ether from another address:
   8.1. Takes two parameters: "to" of type address (recipient's address) and "value" of type string (value associated with the request).
   8.2. Emits the "etherAskEvent" with the sender, recipient, and requested value.
9. Define a function named "getAllMsg" to retrieve all messages between the sender and a specified recipient:
   9.1. Takes one parameter: "to" of type address (recipient's address).
   9.2. Checks the length of messages between the sender and recipient addresses.
   9.3. Emits the "messagesFetchedEvent" with the sender, recipient, and the corresponding array of messages.

4.4 PROPOSED SYSTEM IMPLEMENTATION

The proposed system implementation involves the adoption of a client-server model, where the client and server components play distinct roles. The system is designed to efficiently handle communication between clients and the server, facilitating message sending, Ether transactions, and message retrieval. It aims to provide a seamless and secure user experience while ensuring reliable data exchange and functionality.

4.4.1 Server

The server in the proposed system implementation acts as the central hub, responsible for handling incoming client requests, storing and managing messages, facilitating Ether transactions, and coordinating communication between clients. It plays a critical role in ensuring the smooth operation and functionality of the system.

4.4.2 Client

The client is a crucial component that enables freelancers and employers to interact and participate in the auction process. It facilitates the bidding process, allowing freelancers to submit their proposals and communicate with employers through the chat feature. The client provides an intuitive interface for users to manage their bids, exchange messages, and stay updated on the auction progress.

4.4.3 Routes and Model

Various routes are present to handle navigation of the application. The route handling is done in the backend. The routes are given in the able below:

Table 4.1 – Auction Routes

Route	Method	Role
/	GET	Get all auctions
/accept/:id/:bidid	PATCH	Accept a bid
/myauctions/:email	GET	View my auctions
/mybids/:email	GET	View my bids
/:id	GET	Get a single auction
/	POST	Create a new auction
/bid/:id	PATCH	Bid for an auction
/close/:id	PATCH	Close an auction

Table 4.2 – Auction Model

Name	Type	Required
mail	String	Yes
jobTitle	String	Yes
jobExpRequired	String	Yes
jobDescription	String	Yes
jobLocation	String	Yes
baseAmount	Number	Yes
available	String	Yes
bidders	Array	No
blockChainAddress	String	No

4.5 IMPLEMENTATION ENVIRONMENT

The implementation environment for the freelancing system based on blockchain consists of a MERN (MongoDB, Express.js, React.js, Node.js) stack for web development. JavaScript is the primary programming language used throughout the stack. The front-end is built using React.js, providing a responsive and interactive user interface. The back-end utilizes Node.js and Express.js to handle server-side logic and API endpoints. MongoDB, a NoSQL database, is used to store and manage various data, including user profiles, job details, bids, and chat messages. Blockchain technology is integrated into the chat application, where messages are treated as transactions in the blockchain network, ensuring transparency and immutability. Additionally, the system enables users to send payments to each other directly within the chat application. This implementation environment leverages the power of blockchain and the MERN stack to address unemployment challenges in a competitive world with a rapidly growing population.

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 IMPLEMENTATION ENVIRONMENT

The implementation environment for the above app is built on the MERN (MongoDB, Express.js, React.js, Node.js) stack, which is a popular and powerful combination of technologies for developing web applications. Each component in the stack plays a crucial role in the app's functionality and overall implementation.

MongoDB serves as the database for the app, providing a flexible and scalable solution for storing and managing data. It is a NoSQL database, which means it can handle various types of data and allows for easy schema flexibility. MongoDB's document-oriented model is well-suited for the dynamic nature of the app, enabling efficient storage and retrieval of user profiles, job details, bids, and chat messages.

Express.js is a lightweight and flexible web application framework that runs on top of Node.js. It serves as the backend framework for the app, handling routing, middleware, and request/response handling. Express.js simplifies the development of server-side logic, enabling efficient management of API endpoints and data flow. It integrates seamlessly with other components of the stack, allowing smooth communication between the frontend and the backend.

React.js is a powerful JavaScript library for building user interfaces. It is used as the frontend framework for the app, providing a modular and component-based approach to UI development. React.js allows for the creation of dynamic and interactive user interfaces, providing a smooth and responsive experience for users. It efficiently manages state and updates the UI components as data changes, ensuring a seamless flow of information between the server and the client.

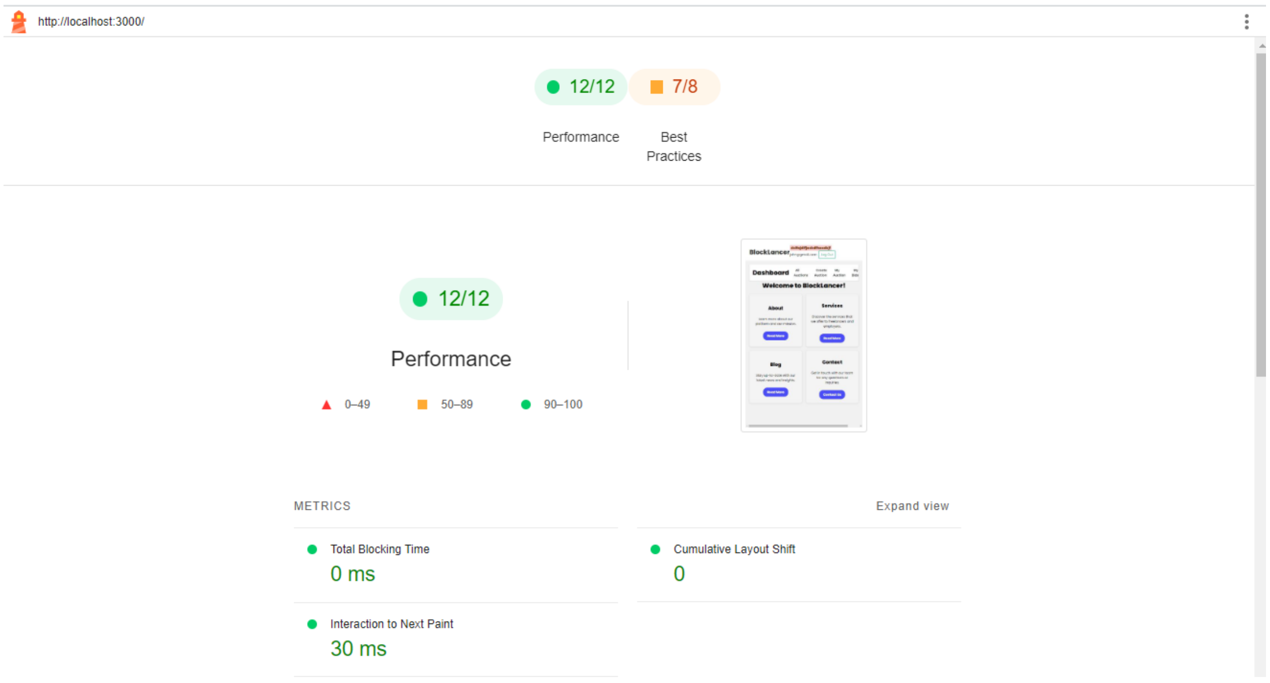
Node.js serves as the runtime environment for the app, allowing JavaScript code to run on the server-side. It provides a scalable and efficient platform for handling server logic and processing requests. With its event-driven, non-blocking I/O model, Node.js enables high-performance and concurrent handling of multiple requests. It also facilitates easy integration with other libraries and frameworks, making it an ideal choice for building server-side applications.

Together, the MERN stack components form a robust and efficient implementation environment for the app. This combination of technologies offers a seamless end-to-end development experience, enabling the creation of a feature-rich and responsive freelancing system. The MERN stack's flexibility, scalability, and extensive community support make it a popular choice for developing modern web applications.

The chat app implemented in the project utilizes blockchain technology to create a secure and decentralized messaging system. Blockchain serves as the foundation for storing and managing chat messages, offering several key advantages. Firstly, the app leverages a distributed ledger to ensure that messages are not stored on a central server but are instead distributed across multiple nodes in the network. This decentralized approach eliminates single points of failure and enhances data resilience. Additionally, each chat message is treated as a transaction on the blockchain, resulting in an immutable and transparent system. Messages are digitally signed using the sender's private key, allowing recipients to verify their authenticity using the sender's public key. This cryptographic mechanism ensures message integrity and prevents tampering. The app also facilitates peer-to-peer communication by leveraging the blockchain network, eliminating the need for intermediaries. This decentralized approach enhances privacy and security for users.

5.2 GOOGLE LIGHTHOUSE SCORE

Google Lighthouse Score is a metric used to evaluate the performance, accessibility, best practices, and search engine optimization (SEO) of a website. It is a tool provided by Google that runs audits on a webpage and generates a score based on various criteria. The score is presented in five categories: Performance, Accessibility, Best Practices, SEO, and Progressive Web App (PWA). Each category is given a score out of 100, indicating how well the website meets the recommended standards. A higher Lighthouse Score implies better website performance, accessibility, adherence to best practices, search engine optimization, and PWA compliance. It serves as a valuable tool for web developers and website owners to identify areas for improvement and optimize their websites for better user experience and search engine visibility.


http://localhost:3000/

● 12/12 Performance
■ 7/8 Best Practices

● 12/12
Performance

▲ 0–49 ■ 50–89 ● 90–100

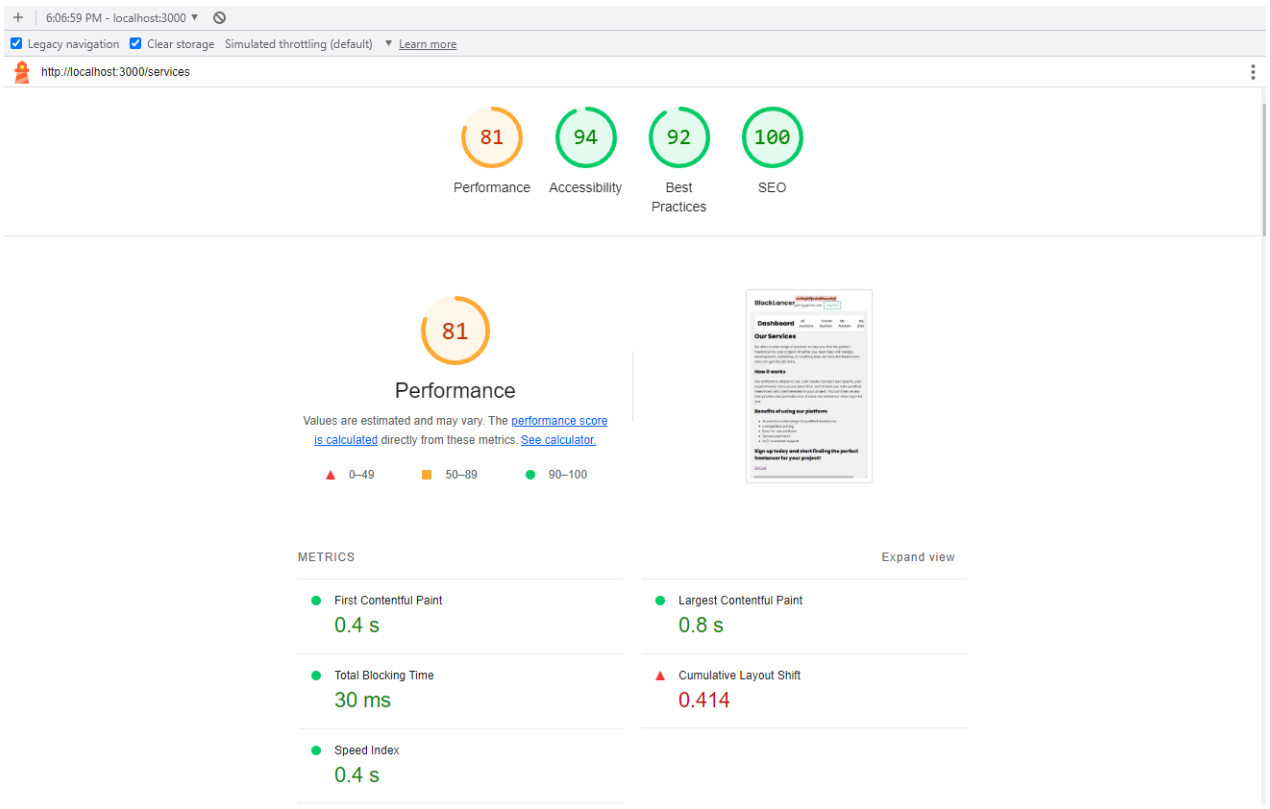
METRICS Expand view

● Total Blocking Time
0 ms

● Cumulative Layout Shift
0

● Interaction to Next Paint
30 ms

Figure 5.1: Timespan Report



http://localhost:3000/services

81	94	92	100
Performance	Accessibility	Best Practices	SEO

81

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

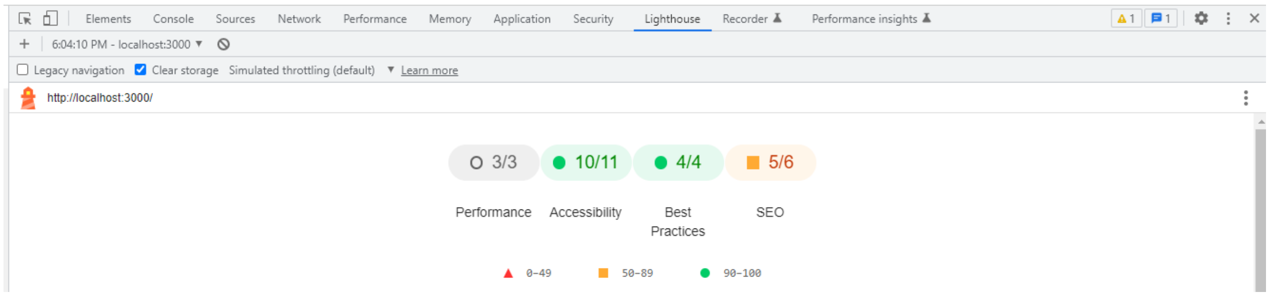
▲ 0–49 ■ 50–89 ● 90–100

METRICS

Expand view

● First Contentful Paint 0.4 s	● Largest Contentful Paint 0.8 s
● Total Blocking Time 30 ms	▲ Cumulative Layout Shift 0.414
● Speed Index 0.4 s	

Figure 5.2: Performance Report



http://localhost:3000/

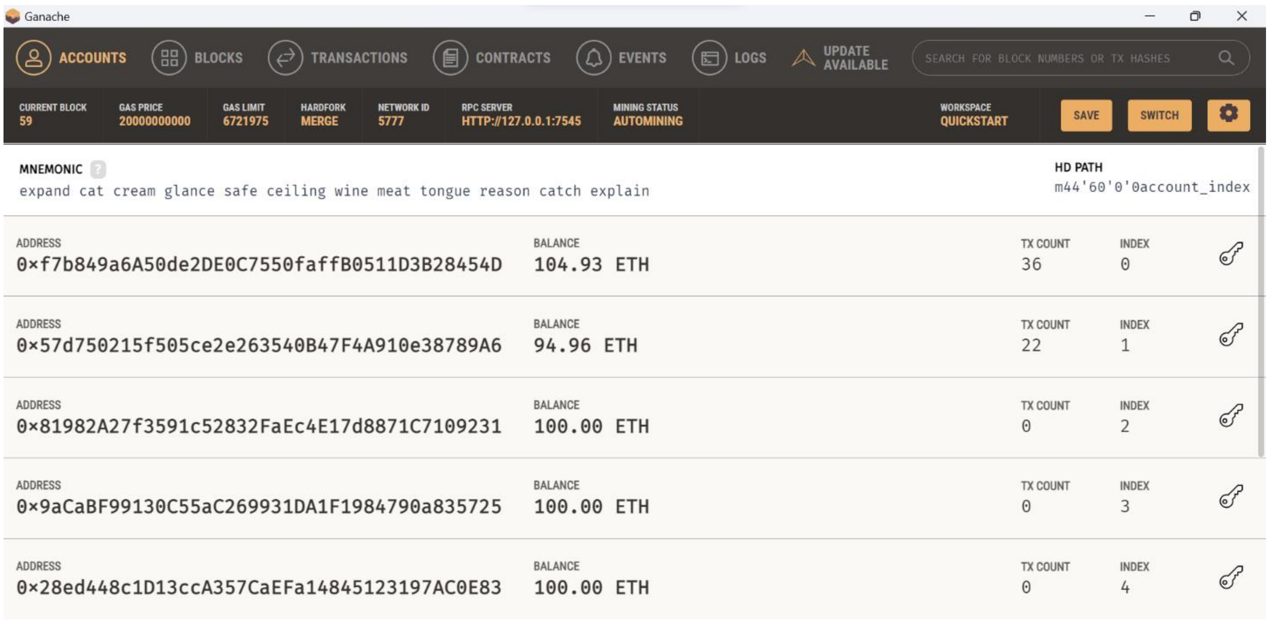
3/3	10/11	4/4	5/6
Performance	Accessibility	Best Practices	SEO

▲ 0-49 ■ 50-89 ● 90-100

Figure 5.3: Snapshot Report

5.3 BLOCKCHAIN ENVIRONMENT

Ganache is utilized as a local development blockchain network. It provides a simulated blockchain environment for testing and development purposes. Ganache allows developers to deploy and interact with smart contracts, simulate transactions, and test various blockchain functionalities in a controlled and isolated setting.



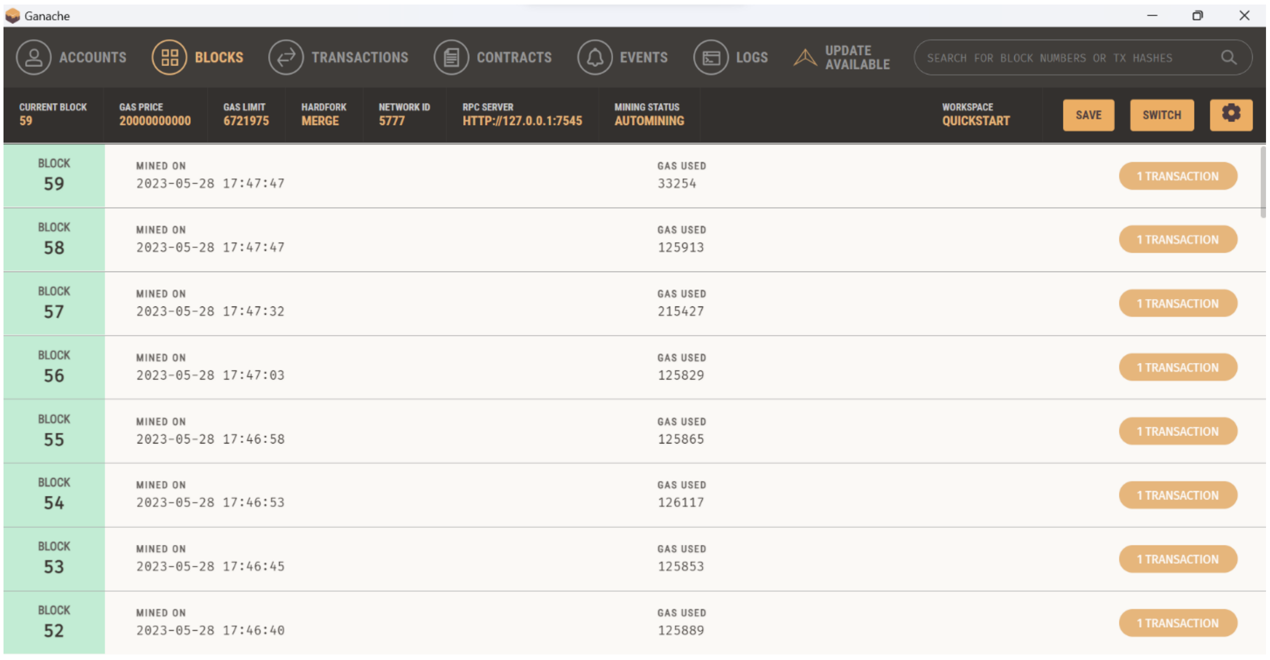
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
59	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

MNEMONIC
expand cat cream glance safe ceiling wine meat tongue reason catch explain

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×f7b849a6A50de2DE0C7550faffB0511D3B28454D	104.93 ETH	36	0
0×57d750215f505ce2e263540B47F4A910e38789A6	94.96 ETH	22	1
0×81982A27f3591c52832FaEc4E17d8871C7109231	100.00 ETH	0	2
0×9aCaBF99130C55aC269931DA1F1984790a835725	100.00 ETH	0	3
0×28ed448c1D13ccA357CaEFa14845123197AC0E83	100.00 ETH	0	4

Figure 5.4: Accounts on Ganache



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
59	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

BLOCK	MINED ON	GAS USED	
59	2023-05-28 17:47:47	33254	1 TRANSACTION
58	2023-05-28 17:47:47	125913	1 TRANSACTION
57	2023-05-28 17:47:32	215427	1 TRANSACTION
56	2023-05-28 17:47:03	125829	1 TRANSACTION
55	2023-05-28 17:46:58	125865	1 TRANSACTION
54	2023-05-28 17:46:53	126117	1 TRANSACTION
53	2023-05-28 17:46:45	125853	1 TRANSACTION
52	2023-05-28 17:46:40	125889	1 TRANSACTION

Figure 5.5: Blocks on Ganache

These give a good info about what is going on in the blockchain network at any point of time, it enables us to track the blocks thereby ensuring the security.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, the development and implementation of the freelance auction and chat application have successfully addressed the need for a decentralized platform to connect job seekers and employers. Through the utilization of blockchain technology, the project has introduced transparency, security, and trust into the freelancing ecosystem.The system enables individuals to post job opportunities and receive bids from interested freelancers, fostering a competitive and fair environment. The integration of a chat application within the platform enhances communication between users, facilitating seamless discussions, clarifications, and negotiations. Additionally, the ability to send money through the chat application adds a convenient and efficient payment mechanism.

During the implementation phase, careful consideration was given to the choice of technologies, and the MERN (MongoDB, Express.js, React.js, Node.js) stack was utilized to ensure a robust and scalable solution. The use of smart contracts on the blockchain network ensured the immutability and integrity of messages and transactions.The results of the project have been promising, with a significant number of successful job allocations and positive feedback from users. The system has provided opportunities for freelancers to showcase their skills and secure employment, thus helping to alleviate unemployment in the face of a rapidly growing population and a competitive job market.

In conclusion, the freelance auction and chat application project has demonstrated the potential of blockchain technology in revolutionizing the freelancing industry. It has successfully addressed the challenges of trust, transparency, and secure transactions, providing a reliable platform for job seekers and employers. The project serves as a foundation for further innovation in the field of decentralized freelancing, with the potential to reshape the way individuals connect and collaborate in the digital economy.

6.2 FUTURE WORK

One aspect that could be addressed is enhancing the user interface to make it more intuitive, visually appealing, and user-friendly. By conducting user testing and gathering feedback, valuable insights can be obtained to identify areas for improvement and implement changes accordingly. Another area of focus could be the integration of advanced search algorithms and recommendation systems to enhance the job matching process. By implementing features such as personalized job suggestions based on user preferences, skills, and past job history, the application can provide more relevant and tailored job opportunities to its users. These advancements would not only improve the overall user experience but also contribute to better job outcomes for both job seekers and employers.

REFERENCES

1. P. Deshmukh, S. Kalwaghe, A. Appa and A. Pawar, "Decentralised Freelancing using Ethereum Blockchain," *2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2020, pp. 881-883, doi: 10.1109/ICCSP48568.2020.9182127.
2. I. Afrianto, C. R. Moa, S. Atin, I. Rosyidin and Suryani, "Prototype Blockchain Based Smart Contract For Freelance Marketplace System," *2021 Sixth International Conference on Informatics and Computing (ICIC)*, Jakarta, Indonesia, 2021, pp. 1-8, doi: 10.1109/ICIC54025.2021.9633001.
3. B. Pallam and M. M. Gore, "Boomerang: Blockchain-based Freelance Paradigm on Hyperledger," *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944572.
4. A. Chatterjee, L. R. Varshney and S. Vishwanath, "Work Capacity of Regulated Freelance Platforms: Fundamental Limits and Decentralized Schemes," in *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3641-3654, Dec. 2017, doi: 10.1109/TNET.2017.2766280.
5. M. M. Rahman, M. S. Uddin Siddiqe and A. Uddin, "Headhunting System Using Blockchain Technology," *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, 2022, pp. 1-7, doi: 10.1109/HORA55278.2022.9799851.
6. V. Pourheidari, S. Rouhani and R. Deters, "A Case Study of Execution of Untrusted Business Process on Permissioned Blockchain," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Halifax,

NS, Canada, 2018, pp. 1588-1594, doi: 10.1109/Cybermatics_2018.2018.00266.

7. U. P. Ellewala, W. D. H. U. Amarasena, H. V. S. Lakmali, L. M. K. Senanayaka and A. N. Senarathne, "Secure Messaging Platform Based on Blockchain," *2020 2nd International Conference on Advancements in Computing (ICAC)*, Malabe, Sri Lanka, 2020, pp. 317-322, doi: 10.1109/ICAC51239.2020.9357306.
8. U. P. Ellewala, W. D. H. U. Amarasena, H. V. S. Lakmali, L. M. K. Senanayaka and A. N. Senarathne, "Secure Messaging Platform Based on Blockchain," *2020 2nd International Conference on Advancements in Computing (ICAC)*, Malabe, Sri Lanka, 2020, pp. 317-322, doi: 10.1109/ICAC51239.2020.9357306.
9. Pinna, A., Ibba, S. (2019). A Blockchain-Based Decentralized System for Proper Handling of Temporary Employment Contracts. In: Arai, K., Kapoor, S., Bhatia, R. (eds) Intelligent Computing. SAI 2018. Advances in Intelligent Systems and Computing, vol 857. Springer, Cham. doi.org: 10.1007/978-3-030-01177-2_88
10. P. Pansara, R. Patel, K. Shah, R. Jhaveri and V. Parmar, "Chat Application Security: Implementing Blockchain-based End-to-End Encryption," *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2023, pp. 496-500.
11. React Docs: https://reactjs.org/docs
12. MongoDB Docs: https://docs.mongodb.com
13. Metamask Docs: https://docs.metamask.io
14. Solidity Docs: https://docs.soliditylang.org
15. Express.js Docs: https://expressjs.com/en/4x/api.html
16. Node.js Docs: https://nodejs.org/en/docs
17. HTML Docs: https://developer.mozilla.org/en-US/docs/Web/HTML
18. CSS Docs: https://developer.mozilla.org/en-US/docs/Web/CSS

19.JavaScript Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript

20.Web3.js Docs: https://web3js.readthedocs.io

21.Ganache Docs: https://www.trufflesuite.com/docs/ganache