DECLARATION

It is hereby declared that the project work entitled "Newebify Blockchain" submitted to Kalinga Institute of Industrial Technology (KIIT), Bhubaneswar, is a record of bonafide work carried out by Waibhav Jha under the guidance of **Prof. Bindu Agarwalla**, Department of Computer Science and Engineering, KIIT, Bhubaneswar. This project work is submitted in the partial fulfilment of the requirements of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Engineering of KIIT University, Bhubaneswar during the year 2023.

WAIBHAV JHA
21052209

ABSTRACT

Newebify Blockchain is a platform that will serve as a public space on the internet where visitors can send a friendly greeting along with a message. The unique feature of this platform is that all data will be securely stored on the blockchain using an Ethereum smart contract. In essence, a smart contract is a set of self-executing code deployed on the blockchain, enabling both reading and writing operations. [**?**]

The main objective of the project is to provide an elegant and intuitive user interface for users to connect their wallets and interact with the smart contract. This project combines web development with blockchain technology for an engaging and secure user experience making it an enriching experience for users to help facilitate the learning of Blockchain.

This project is a fully functional decentralized application that is built using tools like Solidity, Hardhat, React, CSS, JavaScript.

Blockchain is a transformative technology revolutionizing the way we manage and secure data. At its core, it's a decentralized ledger system that records transactions across a network of computers. Each transaction is grouped into a block, and these blocks are linked together using cryptographic hashes, forming an immutable chain. This distributed nature ensures transparency and security, as altering one block would require consensus across the entire network, making it highly resistant to fraud.[**?**]

Solidity is a high-level programming language specifically designed for writing smart contracts on the Ethereum blockchain. It provides the foundation for creating self-executing contracts that automatically perform actions when certain conditions are met. Solidity is known for its syntax similarity to JavaScript, making it accessible for developers familiar with web development. It also incorporates security features to prevent vulnerabilities in smart contracts, ensuring robust and reliable blockchain applications.

Hardhat is a powerful development environment for building, testing, and deploying smart contracts on the Ethereum blockchain. It streamlines the process of writing and deploying smart contracts by providing a comprehensive suite of tools and plugins. It also integrates seamlessly with popular Ethereum development frameworks and testing libraries, making it a go-to choice for Ethereum development projects. With features like built-in support for TypeScript and Ethereum mainnet forking, Hardhat is a versatile tool that accelerates the smart contract development process.[**?**]

React is an open-source JavaScript library used for building user interfaces, particularly for single-page applications where user experience is crucial. React allows developers to create reusable UI components that update in real-time as data changes. Its virtual DOM (Document Object Model) system optimizes rendering performance, enhancing the responsiveness of web applications.[2]

ACKNOWLEDGEMENTS

I am profoundly grateful to **Prof. Bindu Agarwalla** for her expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

I would also like to express my deepest appreciation towards **Dr. Tanmaya Mohanty**, Principal, KIIT, Bhubaneswar and **Prof. Biswajit Sahoo**, Head of Department, Computer Science and Engineering for their ongoing support and encouragement.

Lastly, yet most importantly i would like to express my gratitude toward my family and friends, without whose support this project would have been impossible to complete.

Waibhav Jha

Contents

1 Introduction

1.1 Problem Statement

Develop a decentralized application which revolutionize online interactions. It's a platform where visitors can send messages. All data is safeguarded on the blockchain through an Ethereum smart contract. Imagine it as a digital vault with self-operating security protocols. Even if you're not tech-savvy, this project offers a golden opportunity to explore the world of blockchain. It's a user-friendly, secure, and innovative user interface that welcomes anyone intrigued by the potential of this cutting-edge technology.

1.2 Objectives of the Project Work

The main objective of the Newebify Blockchain project is to establish an open digital space on the internet, and to provide an elegant and intuitive user interface, making it an enriching experience for users to send messages. The project's standout feature lies in its robust security protocol, ensuring that all information is stored on the blockchain through an Ethereum smart contract and also to help facilitate the learning of Blockchain, all on the cloud.[3]

This application operates entirely on the server and doesn't necessitate any extra local software installation. Additionally, it seamlessly integrates with Metamask for enhanced functionality and can also be used with a wide range of modern web browsers, including Google Chrome, Apple Safari, Mozilla Firefox, Microsoft Internet Explorer and the Brave browser.

The Newebify Blockchain offers the following features and functionalities:

1. It is an independent application with all data securely stored on the blockchain through an Ethereum smart contract, with Metamask integration enabling seamless interaction with the stored information.

2. This project utilize the Goerli testnet, a specialized Ethereum test network, to thoroughly evaluate and refine the platform's functionality, this ensures a robust and secure user experience upon full implementation.

3. This platform incorporates a function allowing users to type and transmit messages, this user-generated content remains securely and transparently stored.

4. Supports creation of user-generated content as a guarantee of resilient and reliable framework for secure data management on the blockchain driven by languages such as Solidity and JavaScript.

1.3 Applications

1. Simulate messages on the fly.

2. Users can leave messages securely stored on the blockchain, creating a digital legacy.

3. Facilitates easy testing and debugging

4. Supports inline editing and documentation.

5. Loved by users and friendly with developers, comes as a standalone web application, which is easy to integrate, customise and extend.
6. Supports attachments including files and images.

2 System Requirements

2.1 Hardware Requirements

Minimum System Requirements:

1. Processor: Intel Pentium 2.6 GHz
2. Memory: 512 MB DDR2 @800 MHz
3. Display: Monitor with screen resolution of 1024 x 768
4. Graphics: Intel HD 4000 series (with Open GL support)

Recommended System Requirements:

1. Processor: Intel Core i3 Quad Core 2.8 GHz
2. Memory: 1 GB DDR3 @1333 MHz
3. Display: Monitor with screen resolution of 1440 x 900
4. Graphics: Intel IRIS series (with Open GL ES 2.0 support)

2.2 Software Requirements

1. OS: Windows 7 or later / Mac OSX 10.6 Snow Leopard or later / Ubuntu 14.04 or later
2. Web browser with Adobe Flash support.[1]

2.3 Functional Requirements

2.3.1 External Interface Requirements

1. Front-End (GUI) : An elegant and intuitive is graphical user interface is provided to the user for sending messages. It is also designed to help beginners to hone their skills in the field of blockchain.
2. Communication Interfaces HTTP: Hyper Text Transfer Protocol is a method used to transfer or convey information on the World Wide Web. Its original purpose was to provide the way to publish and retrieve HTML pages. An HTTP client initiates a request by establishing Transmission Control Protocol (TCP) connection to a particular port on a remote host. An HTTP server listening on that port waits for the client to send a request message. Upon receiving request, the server sends back a sta- tus line, such as HTTP/1.1 200 OK and a message of its own, the body of which is perhaps the request file, an error message, or some other information. Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs) (or, more specifically, URLs). Us- ing the HTTP: or HTTPS URI schemes.

2.4 Non Functional Requirements

2.4.1 Performance Requirements

The web server that hosts the Newebify Blockchain must be able to handle and support multiple instances of application. The time delay between request and acknowledgement should be very small in case of online help required. Minimum time should be taken by the application to display the preview of the messages developed by the user. In case of power failure, the data should be stored in the state that was last saved by the user.

3 Tools and Technologies Used

Newebify Blockchain makes use of the following tools:

1. Solidity:
   Solidity plays a crucial role in defining the functionality and logic of smart contracts that run on the Ethereum blockchain.
   Solidity is essential for writing the smart contracts that power the core functionality of the DApp. These contracts define rules, automate actions, and manage the interactions within the application.

2. CSS:
   CSS plays a crucial role in enhancing the visual appeal and user experience of React applications. Additionally, CSS facilitates responsive design, ensuring seamless adaptation to various screen sizes and devices.[4]

3. JavaScript:
   JavaScript also facilitates seamless communication between the front-end interface and the blockchain, allowing for actions like sending transactions and retrieving data. JavaScript framework like ethers.js provide essential tools for interacting with smart contracts.[5]

4. React:
   React's state management and lifecycle methods enable seamless handling of complex user interactions and dynamic data updates, enhancing the user experience. React's virtual DOM and one-way data binding streamline the development process.[**?**]

4 System Design

Blockchain is a distributed ledger technology that provides a secure and transparent way to record transactions across a network of computers. It operates on a decentralized architecture, where data is stored in blocks that are linked together in a chronological order. With its immutable and tamper-resistant nature, blockchain finds applications in various industries, from finance to supply chain management, offering a reliable foundation for building secure and transparent systems.

Advantages over conventional systems:

1. Unlike traditional centralized systems, blockchain operates on a decentralized network of nodes.

2. Transactions on a blockchain are recorded in a tamper-proof and transparent manner. Once added to the chain, it becomes extremely difficult to alter or delete data, enhancing security and trust.

3. The data stored on a blockchain is immutable, meaning that once recorded, it cannot be changed.

4. The transparent and auditable nature of blockchain builds trust among participants.

4.1 Data

File and Data formats
The input data will be in the form of textual information. User can save the file on the cloud for future reference.

4.2 Code

The most radical core functions of the Newebify Blockchain are as follows:

1. The getAllWaves() Function:

   let allWaves = await waveContract.getAllWaves();

   This line of JavaScript code is used to retrieve data from a smart contract deployed on the Ethereum blockchain. It retrieves data, likely a list of interactions or waves, and assigns it to the variable allWaves for further use in the application.

5 System Implementation

The Newebify Blockchain is implemented with the assistance of Ethereum's decentralized ledger technology and supported by the deployment of an Ethereum smart contract. This contract serves as the backbone of the platform, enabling secure data storage and interaction with the blockchain.

The decentralized nature of the blockchain means that no single entity has control over the data, enhancing security and eliminating the risk of central points of failure.

5.1 Screenshots of output

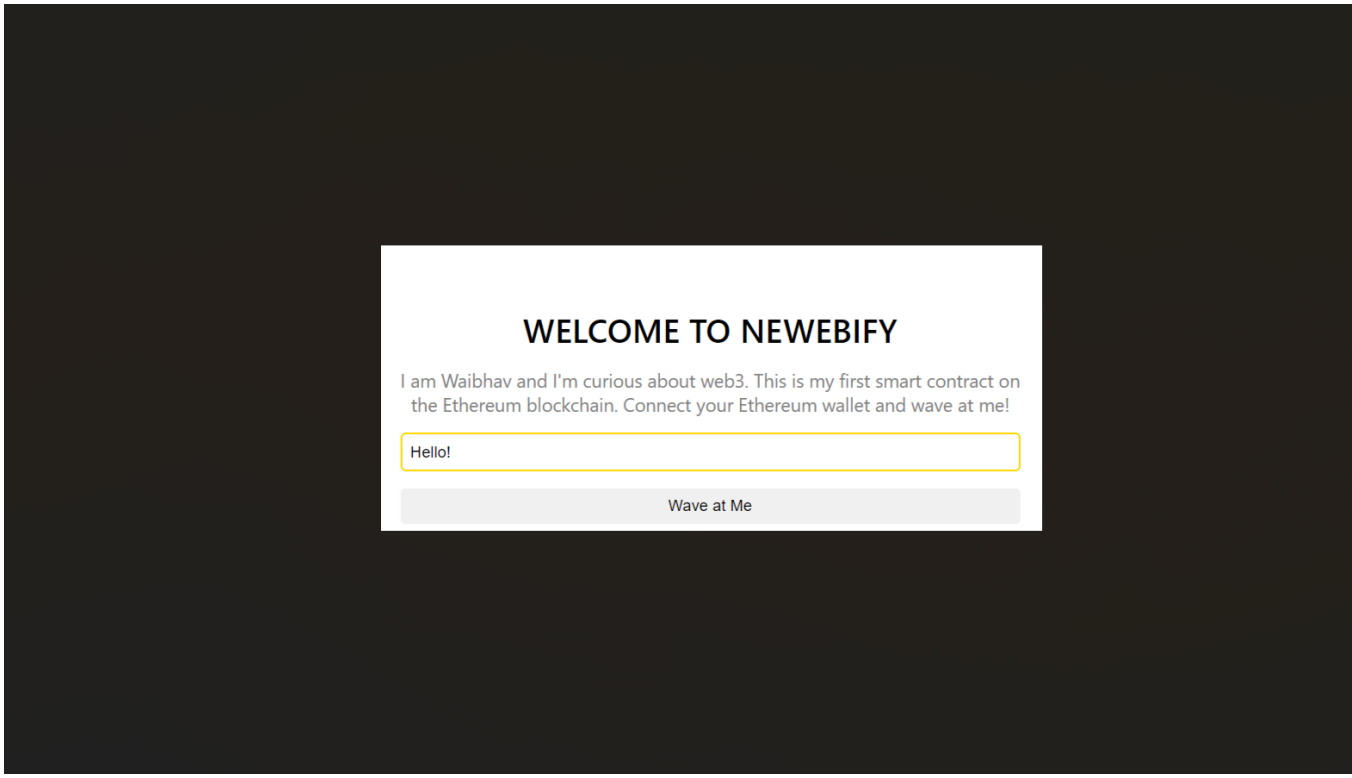

Figure 1: Hompage

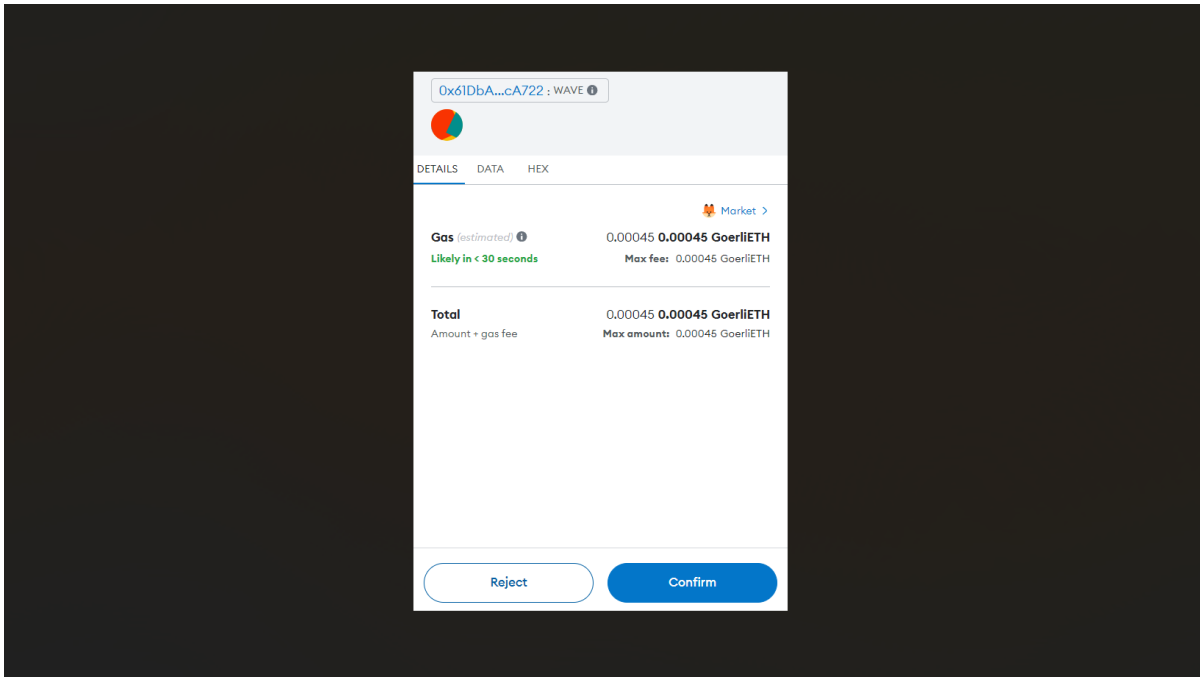

Figure 2: Metamask

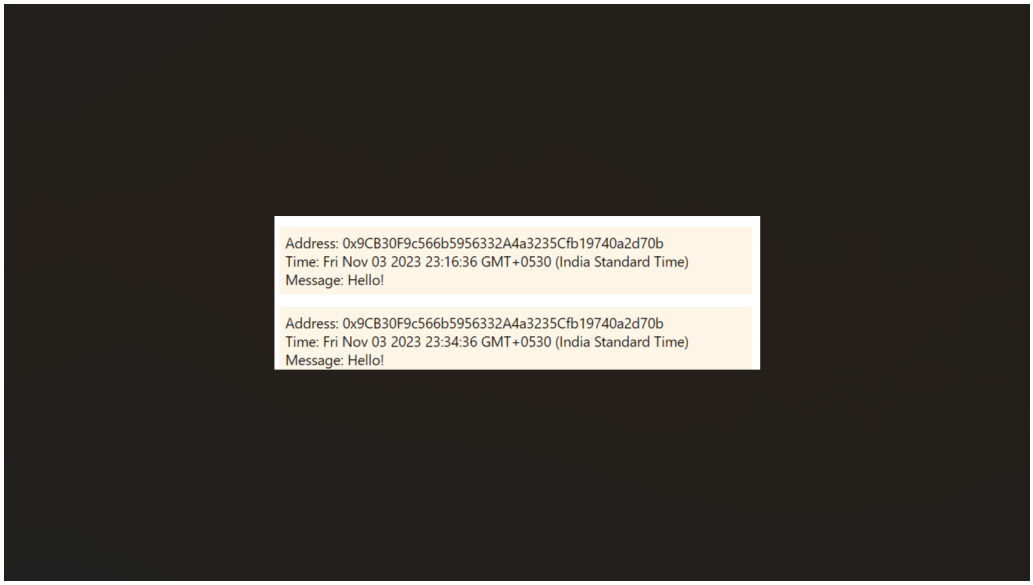

Figure 3: Message

6 System Testing and Result Analysis

In order to overcome the drawbacks the need to be able to offer clients the opportunity to send messages. This project?s goal is provide a basic interface to send messages.

The testing is divided into two sub stages:

1. Unit and integration level adherence to coding standards and successful communication between units.
2. The items to be tested includes, images, tables, horizontal lines, radio button, checkbox, text field, label, hyperlinks.

6.1 Features to be tested

Testing is performed manually and the browser will be brought down while the testing is carried out. The entrance criteria?s for each phase of testing must be met before the next phase can commence.

1. Accessibility
2. Coding Standards
3. Compatibility
4. Functional
5. Validation of Forms
6. Connectivity
7. Scalability
8. Security
9. Usability

6.2 Test Deliverables

The following documents will be generated as a result of these testing activities:

1. Master test plan (MTP - this document)
2. Test input and output data (Test cases).

6.3 Coding Standards

Each of the units of code that make up the module being tested (typically a single fully functional Web page) must be coded to all of the following coding standards, any deviations from the standard must be documented and approved.
The code must pass the following syntax and design requirements:

1. Each unit of code has been inherited or copied from the most appropriate object class or Template.

2. All data entry fields are checked for invalid data and an appropriate error message is displayed if the data is found to be invalid.

3. Using the browsers Tab key allows the client to tab through the input fields on the Form in a top to bottom, left to right order.

4. All validations are performed (and error messages displayed) in a top- down, left-to-right Fashion.

5. Using equivalence partitioning techniques, all data entry fields will be checked to ensure that they are able to accept valid values and that their error checking routines can handle invalid data appropriately.

7 Conclusion and Future Work

An attempt is made to this innovative implementation ensuring a transparent, immutable, and decentralized approach to managing interactions and messages. It leverages the inherent strengths of blockchain technology, offering a secure and trustworthy environment for users to engage in meaningful exchanges.

In future work, this project can be extended to enhancing user experience and expanding functionalities. Potential avenues for exploration may include optimizing smart contract logic, exploring additional blockchain integrations.

The purpose and objective of this project is achieved. By providing extremely rich graphical user interface, sending messages is easy and in an aesthetic form. Thus, even a novice user can dream and accomplish their wish of sending the messages.

Additionally, ongoing efforts to enhance scalability and explore potential partnerships within the blockchain ecosystem will contribute to the continued growth and success of the Newebify Blockchain platform.

References

[1] Adobe Flash Player, http://www.adobe.com/products/flashplayer.html

[2] React Tutorials, https://react.dev/

[3] Google Cloud JavaScript Libraries, https://cloud.google.com/compute/docs/tutorials/javascript-guide

[4] Tutorial Points CSS, https://www.tutorialspoint.com/css/

[5] JavaScript, https://www.javascript.com/

[6] Solidity, https://soliditylang.org/