**Ethereum Smart Contract Network by Blockchain**



A Project presented to the National University in partial fulfillment of the requirement for The degree of Bachelor of Science (Hon’s) in Computer Science & Engineering

**Supervised By Nusrhat Jahan Sarker**

Lecturer, Department of CSE, Daffodil Institute of IT

**Submitted By**



**Sourav Saha**

Reg. no.: 17502005075

Session: 2017-18

Department of Computer Science & Engineering Daffodil Institute of IT, Dhaka

Under National University Bangladesh Date of Submission: 04.09.23

## APPROVAL

The project “**Ethereum Smart Contract Network by Blockchain”** is submitted to the Department of Computer Science & Engineering, DIIT under the National University of Bangladesh in partial fulfillment of the requirement for the degree of Bachelor of Science (Hon’s) in Computer Science and Engineering and approved as to its style and content.

**Examiner Examiner**

|  |  |  |
| --- | --- | --- |
| **Nusrhat Jahan Sarker** |  | **Md. Imran Hossain** |
| Project Supervisor |  | Head |
| Dept. of CSE |  | Dept. of CSE |
| Daffodil Institute of IT |  | Daffodil Institute of IT |

## DECLARATION

I declare that the project work titled “**Ethereum Smart Contract Network By Blockchain**” being submitted in partial fulfillment for the degree of B.Sc. (Hon’s) in Computer Science & Engineering is the original work carried out by me. It has not formed part of any other project work submitted for any degree or diploma, either in this or any other University.

**Submitted By**

**Sourav Saha**

Reg. no.: 17502005075

Session: 2017-18

**ACKNOWLEDGEMENT**

Despite my efforts, the success of this project depends largely on the encouragement and guidance of my mentors. I would like to take this opportunity to express my gratitude to the people who are playing a vital role in the successful completion of this project.

My sincere thanks to **Prof. Dr. Mohammed Shakhawat Hossain,** principal of DIIT for giving me an opportunity to undertake this project.

My sincere thanks to **Md. Imran Hossain,** department head of the Computer Science & Engineering Department, DIIT for giving me an opportunity to undertake this project.

My cordial thanks to my Project Supervisor **Nusrhat Jahan Sarker,** Lecturer, Department of Computer Science & Engineering, DIIT for his valuable guidance and support to meet the successful completion of my project.

I express my gratitude to **Poly Bhoumik,** Senior Lecturer, Department of Computer Science & Engineering, DIIT for appreciating my goal.

My sincere thanks to **Saidur Rahman,** Senior Lecturer and Batch Coordinator, Department of Computer Science & Engineering, DIIT for appreciating my goal.

I express my gratitude to **Safrun Nesa Saira,** Lecturer, Department of Computer Science & Engineering, DIIT for appreciating my goal.

My sincere thanks to **Mizanur Rahman,** Lecturer, Department of Computer Science & Engineering, DIIT for appreciating my goal.

I express my gratitude to **Moumita Akter,** Lecturer, Department of Computer Science & Engineering, DIIT for appreciating my goal.

I express my gratitude to **Md Mushfiqur Rahman,** Lecturer, Department of Computer Science & Engineering, DIIT for appreciating my goal.

I extend my sincere thanks to my family and friends for their constant support throughout this project.

Finally, I would be grateful to the **National University, Bangladesh,** and the coordinator of the **Bachelor of Science in Computer Science and Engineering** degree program for giving me this opportunity to apply the knowledge that I have gained through the study.

## ABSTRACT

This project presents the development of an Ethereum Smart Contract Network by Blockchain. The goal is to provide users with a secure and decentralized platform to manage their digital assets. By leveraging Ethereum's smart contracts, the wallet ensures transparency and control over transactions, enhancing user experience while adhering to industry security standards. The project contributes to the adoption of decentralized financial solutions, empowering users in the blockchain ecosystem.

**TABLE OF CONTENTS**

[APPROVAL ii](#_TOC_250046)

[DECLARATION iii](#_TOC_250045)

ACKNOWLEDGMENT iv

[ABSTRACT v](#_TOC_250044)

Contents

[Chapter 1: Introduction 1-6](#_TOC_250043)

* 1. [Introduction 2](#_TOC_250042)
  2. [Objective of the project 2-3](#_TOC_250041)
  3. Business Perspective 3-4
  4. [Limitations of the existing system 4](#_TOC_250040)
  5. [Comparison Between Traditional & Blockchain Technology 5](#_TOC_250039)
  6. [Centralized vs Decentralized Architecture (Diagram) 6](#_TOC_250038)
  7. Features 6
  8. [Summary 6](#_TOC_250037)

[Chapter 2: Background Study 7-11](#_TOC_250036)

* 1. [Introduction to background processing 8](#_TOC_250035)
  2. [Challenges in background processing 8-9](#_TOC_250034)
  3. [Feasibility 9-10](#_TOC_250033)
  4. [Applicability of our system 10-11](#_TOC_250032)
  5. [Summary 11](#_TOC_250031)

Chapter 3: Design of our Proposed System 12-16

* 1. [Introduction 13](#_TOC_250030)
  2. [Blockchain Technology 13](#_TOC_250029)
  3. [Proposed Blockchain System Architecture 13-14](#_TOC_250028)
  4. [Modeling 14-16](#_TOC_250027)
  5. [Summary 16](#_TOC_250026)

[Chapter 4: Methodology 17-20](#_TOC_250025)

* 1. [Introduction 18](#_TOC_250024)
  2. [Our Used Methodology 18](#_TOC_250023)
     1. [Agile Software Development 18](#_TOC_250022)
     2. [Advantages of Agile Method 19](#_TOC_250021)
  3. When We Choose Agile Model 19-20
  4. [Summary 20](#_TOC_250020)

Chapter 5: System Specification 21-25

[5.1 Introduction to requirements 22](#_TOC_250019)

* 1. Development Requirements 22
  2. JavaScript 22
  3. VS Code 22-23
  4. Web Browser 23
  5. MetaMask 23

5.6 Web 3.0 24

* 1. CSS 24
  2. React JS 24-25
  3. Ganache 25
  4. Summary 25

Chapter 6: Design of the System 26-33

* 1. [Introduction to system overview 27](#_TOC_250018)
  2. [Workflow Diagram 27](#_TOC_250017)
  3. [Flow Chart 28](#_TOC_250016)
  4. [E-R Diagram 29](#_TOC_250015)
  5. Data Flow Diagram (DFD) 30-31
  6. [Sequence Diagram 31](#_TOC_250014)
  7. [Class Diagram 32](#_TOC_250013)
  8. [Use CASE Diagram 33](#_TOC_250012)
  9. [Summary 33](#_TOC_250011)

Chapter 7: Implementation 34-39

* 1. [Introduction 35](#_TOC_250010)
  2. [Login Page 35](#_TOC_250009)
  3. [Home Page 36](#_TOC_250008)
  4. Language Selection 36-38
  5. [Transaction Page 38](#_TOC_250007)
  6. [Record Page 39](#_TOC_250006)
  7. [Summary 39](#_TOC_250005)

Chapter 8: Future Enhancement & Conclusion 40-41

* 1. [Introduction 41](#_TOC_250004)
  2. Limitation of This Project 41
  3. [Future Enhancement 41](#_TOC_250003)
  4. [Conclusion 41](#_TOC_250002)
  5. [Summary 41](#_TOC_250001)

[REFERENCES 42](#_TOC_250000)

Appendix 43-46

**List of Figure**

|  |  |  |
| --- | --- | --- |
| **SL** | **Figure Name** | **Page No** |
| 1.6 | Centralized vs Decentralized Architecture (Diagram) | 6 |
| 3.3 | Proposed Blockchain System Architecture | 14 |
| 3.4 | Modeling | 15 |
| 6.2 | Work Flow Diagram | 28 |
| 6.3 | Flowchart | 29 |
| 6.4 | ER Diagram | 30 |
| 6.5.1 | DFD Level-0 | 31 |
| 6.5.2 | DFD Level-1 | 31 |
| 6.5.3 | DFD Level-2 | 32 |
| 6.6 | Sequence Diagram | 32 |
| 6.8 | Class Diagram | 33 |
| 6.9 | Use Case Diagram: | 34 |

**List of Table**

|  |  |  |
| --- | --- | --- |
| **SL** | **Table Name** | **Page No** |
| 1.5 | Comparison Between Traditional & Blockchain  Technology | 5 |

# Chapter 1 Introduction

## Introduction

Our project is a platform for researchers and academics to easily access and share their research papers with a global audience. Our project aims to facilitate the discovery of new research papers and scholarly works by making them easily searchable and accessible to a wider community. Our goal is to increase the visibility and impact of research papers by making them easily discoverable by other researchers, students, and professionals.

Our platform provides a space for researchers to collaborate and share their research findings with others in their field, promoting open access to research papers and making them freely available to anyone with an internet connection. We believe that an open platform for researchers to share, discuss, and critique research papers and scholarly works can facilitate the exchange of ideas and the sharing of knowledge among researchers, students, and professionals in different disciplines and fields.

Our project book provides an easy-to-use and user-friendly interface for researchers and academics to access and share their research papers. We aim to improve the quality of research papers by providing a platform for peer review and feedback, and to create a space for researchers to publish their papers and get cited by others.

## Objective of the project

The objective of developing an Ethereum Smart Contract Network by Blockchain is to create a secure and user-friendly decentralized financial solution for managing digital assets. The project aims to achieve the following specific objectives:

* + - **Security:** Implement robust security measures to protect users' private keys anddata, ensuring a high level of trust and confidence in the wallet's integrity
    - **Decentralization:** Leverage the power of Ethereum's blockchain and smartcontracts to establish a decentralized infrastructure, reducing reliance on centralauthorities and enabling peer-to-peer transactions.
    - **User Experience:** Design an intuitive and responsive user interface using React,offering a seamless and enjoyable experience for users while managing their digital assets.
    - **Transparency:** Utilize smart contracts to facilitate transparent and auditable transactions, enabling users to track and verify their transactions on the public blockchain.
    - **Asset Management:** Enable users to have full control over their digital assets, allowing them to store, send, receive, and manage various cryptocurrencies from a single wallet.
    - **Industry Compliance:** Adhere to industry best practices and follow the latest Ethereum standards to ensure compatibility with other blockchain applications and wallets
    - **Adoption of Decentralized Finance:** Contribute to the broader adoption of decentralized financial solutions by providing a secure and accessible platform for managing digital assets.

Overall, the project aims to foster a decentralized financial ecosystem by empowering users with greater control and security over their digital assets, thereby promoting the advantages of blockchain technology and Ethereum's smart contracts in the realm of financial services [1].

## Business Prospect

The development and implementation of an Ethereum Smart Contract Network by Blockchain present compelling opportunities from a business perspective. The wallet's integration of Ethereum's smart contracts and React's user-friendly interface opens up numerous possibilities in the evolving landscape of decentralized finance. From a business standpoint, the Ethereum Smart Contract Wallet offers several key advantages:

* + - **Market Differentiation:** By providing a secure and user-friendly platform for managing digital assets, the business can differentiate itself in the competitive DeFi market. The combination of Ethereum's blockchain technology and React's intuitive UI/UX sets the wallet apart from conventional wallets, attracting users seeking enhanced security and a seamless experience.
    - **Revenue Generation:** The wallet's implementation can be monetized through various avenues. These include transaction fees for user-initiated smart contract operations, subscription-based premium features, or partnerships with DeFi protocols to share in the revenue generated from users' interactions with those protocols.
    - **Increased User Adoption**: The user-centric approach of the wallet, coupled with its focus on security and decentralized control, appeals to a broader audience. This can lead to increased user adoption and retention rates, attracting both novice users seeking simplicity and experienced users valuing security and autonomy.
    - **Partnerships and Integrations**: The Ethereum Smart Contract Wallet can form partnerships with DeFi projects and platforms, allowing users to access a widerarray of decentralized financial services directly through the wallet. This mutually beneficial collaboration can increase the wallet's utility and attract more users to both the wallet and partner platforms.
    - **Data Insights and Analytics:** As users interact with the wallet, valuable data insights can be collected, such as user preferences, asset types, and transaction patterns. This data can be anonymized and used to refine marketing strategies, improve user experiences, and drive product enhancements.
    - **Community Building**: Developing an active and engaged community around thewallet can be pivotal to its success. Regular communication, support, and feedback from the community can lead to iterative improvements and foster brand loyalty.
    - **Regulatory Compliance:** Emphasizing regulatory compliance will instill trust among users and investors. Adhering to relevant financial and data privacy regulations will help the business navigate the evolving regulatory landscape.

In summary, from a business perspective, the Ethereum Smart Contract Wallet represents a promising venture in the DeFi space. The integration of Ethereum's smart contracts and React's user interface offers a competitive advantage, revenue generation opportunities, and the potential for increased user adoption. By focusing on user needs, security, and building strong partnerships, the wallet can establish itself as a leading player in the decentralized financial ecosystem.

## Limitations of the existing system:

The existing system of traditional financial institutions and centralized cryptocurrency wallets has several limitations that can impede the efficiency and user experience for individuals:

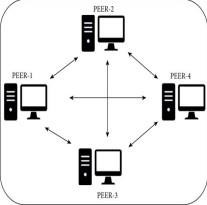
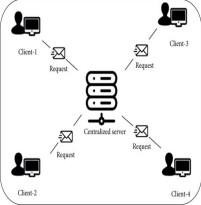
* **Lack of Control:** In centralized financial systems, users have to rely on third- party intermediaries, such as banks and exchanges, to manage their assets.
* **Security Vulnerabilities:** Centralized systems are more susceptible to security breaches and hacks.
* **Limited Transparency**: Centralized financial institutions often lack transparency intheir operations.
* **High Fees:** Traditional financial systems and cryptocurrency exchanges typically charge high transaction fees for services like fund transfers.
* **Slow Transaction Times:** Centralized systems can suffer from slow transaction processing times.
* **Single Point of Failure:** Centralized systems have a single point of failure, meaningthat if the central entity experiences technical issues or goes offline, users' access totheir funds and services is disrupted.
* **Limited Access:** Some centralized financial services may not be accessible to users in certain regions or countries due to regulatory restrictions or compliance issues.
* **Lack of Financial Inclusivity:** Centralized systems can exclude individuals who donot have access to traditional banking services, hindering financial inclusivity.
* **Data Privacy Concerns:** Centralized systems often require users to disclose personalinformation.
* **Dependence on Fiat Currencies:** Many centralized systems primarily rely on fiat currencies, making it challenging for users to interact directly with digital assets and cryptocurrencies without converting back and forth.

Addressing these limitations through the development of a secure and user-friendly Ethereum Smart Contract Wallet using React can provide a viable solution to the challenges faced by users in the existing financial ecosystem. By leveraging the strengths of blockchain technology and smart contracts, this proposed wallet aims to offer users enhanced security, transparency, control, and accessibility, thereby promoting a more inclusive and decentralized financial landscape [7][8].

## Comparison Between Traditional & Blockchain Technology

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Traditional** | **Blockchain (Potential Solution)** |
| **Type of System** | It is a Centralized System | It is a Decentralized System |
| **Fraud Cases** | Huge cases of claimers posing as the seller of land or property.  Thus increasing the chances of fraud cases. | The digital land registration platform could serve as proof of land ownership, record existence, exchange, and transaction by keeping  immutable records of the transactions.  Hence reducing fraud cases. |
| **Middlemen/ Brokers** | Middlemen/ Brokers are involved. | No need for middlemen or brokers |
| **Time** | It is a time expensive process | This system accelerates the process. |
| **Human Error** | The Land Registry system is paper based. Hence, the  chances of bugs increase in the land registry system. | As the user confirms the transfer of land title, smart contracts activate to update the land ownership for a new buyer and respective  transactions get stored on the blockchain.  Thus, the history of the ownership records is traced making the system paperless. Thus, decreasing the  chances of errors in the system. |
| **Need of Witness** | Need at least 2 witnesses | No need for witnesses |

## Centralized vs Decentralized Architecture (Diagram)



**Fig 1.6(a): Centralized Architecture Fig 1.6 (b): Decentralized Architecture**

## Features of our system

The features include:

* + - Send/Receive Ethereum
    - Transition Record
    - Wallet Transition Verification
    - Ethereum Amount Monitoring
    - Private Key Management
    - Multilingual Support

## Summary

Here in this chapter, we have discussed the objectives, benefits, limitations, and features of this project. It shows an overview of the advantages of our system and disadvantages of the existing system.

# Chapter 2 Background Study

## Introduction to background processing

In a rapidly changing digital world, a group of passionate visionaries embarked on a transformative journey to develop an Ethereum Smart Contract Network by Blockchain using the powerful React framework. Their mission was fueled by a shared belief in the potential of blockchain technology to revolutionize finance, promoting decentralization, security, and financial inclusivity. As they delved into the project, they envisioned a platform that would empower individuals with direct control over their digital assets, free from the limitations of traditional financial systems.

Throughout the development process, the team encountered various challenges. They had to navigate the complexities of blockchain development, gain a deep understanding of Ethereum's smart contracts, and integrate the React framework seamlessly to create an intuitive user experience. Yet, their determination and unwavering commitment to their vision drove them forward, surmounting hurdles one by one.

Security and transparency were paramount to the team. They implemented rigorous testing and audits, adopting the latest security measures to ensure the safety of users' funds and data. Recognizing the weight of building a financial platform, they remained dedicated to instilling trust and confidence in their users.

As the project evolved, community involvement became central to their ethos. The team actively engaged with the blockchain community, seeking feedback and insights to continually improve their product. Embracing the open-source nature of the project, they welcomed contributions from developers and enthusiasts worldwide, fostering a collaborative and innovative environment.

The Ethereum Smart Contract Wallet using React they envisioned was not just a project; it represented a vision of the future—a future where individuals have the freedom to manage their financial assets without intermediaries, where financial inclusion becomes a reality for all, and where transparency and decentralization shape a more equitable financial landscape.

As the project nears completion, the team eagerly anticipates the moment they can share their creation with the world. Their journey has been one of passion, perseverance, and a commitment to making a meaningful impact in the world of finance. With the Ethereum Smart Contract Wallet using React, they seek to contribute to the ongoing blockchain revolution, bridging the gap between traditional finance and the decentralized future they envision[2].

## Challenges in background processing

Background processing, also known as asynchronous processing, refers to the execution of tasks or operations outside the main thread of an application. While background

processing offers various benefits , it also presents several challenges that developers need to address:

* + - **Synchronization:** Coordinating data and state between the main thread and background processes can be complex. Developers must implement effective synchronization mechanisms to ensure consistency and avoid race conditions or data corruption.
    - **Resource Management**: Background processing can consume significant system resources, such as CPU and memory. Proper resource management is crucial toprevent performance bottlenecks and crashes.
    - **Error Handling:** Detecting and handling errors that occur during background processing can be challenging. Without proper error handling mechanisms, failed tasks might go unnoticed, leading to unpredictable application behavior.
    - **Prioritization:** In scenarios where multiple background tasks are queued for execution, prioritization becomes essential. Deciding the order of task execution based on importance or urgency requires careful consideration.
    - **Latency**: Background processing can introduce latency between the request andexecution of tasks. Ensuring timely completion of critical tasks is vital to maintain a responsive user experience.
    - **Scalability:** As the workload and complexity of background tasks increase, scaling the background processing system to handle higher volumes of requests can become a significant challenge.
    - **Dependencies:** Background tasks might rely on external services or dependencies, leading to potential bottlenecks if those services are unavailable or experience latency.
    - **Monitoring and Debugging:** Monitoring and debugging background processes can be more challenging than debugging tasks in the main application thread. Real-time visibility into background tasks' status and performance is essential for effective troubleshooting.
    - **Task Abandonment:** Ensuring that background tasks are completed, even in case of application termination or system failures, requires implementing mechanisms to handle task abandonment gracefully.
    - **Data Integrity:** Background processing might involve operations that modify data. Ensuring data integrity and consistency across all background tasks is critical to avoid data corruption or loss.

Addressing these challenges requires careful design, architecture, and the use of appropriate tools and libraries for background processing. Developers must be mindful of potential pitfalls and be proactive in implementing robust solutions to create efficient and reliable background processing systems.

## Feasibility

A feasibility study is an assessment of the practicality of a proposed plan or project. A feasibility study analyzes the viability of a project to determine whether the project or venture is likely to succeed.

* **Technical Feasibility:** We can strongly say that it is technically feasible since there will not be much difficulty in getting the required resources for the development and maintaining the system as well.
* **Economic Feasibility:** The development of this application is highly economically feasible. The organization needed not to spend much money on the development of the system already available. we can attain the maximum usability of the corresponding resources. Even after the development, the organization will not be in a condition to invest more in the organization. Therefore, the system is economically feasible.
* **Behavioral Feasibility:** The proposed system is also behaviorally feasible as it is very user-friendly. Extensive training of the users is not required. The users can easily learn to use the system and can adapt themselves according to the system.
* **Operational Feasibility:** It is mainly related to human organizations and political aspects. The questions to be considered are:

1. What changes will be brought with the system?
2. What organization structures are disturbed?
3. What new skills will be required?

The system is operationally feasible as it is very easy for the End users to operate it. It only needs basic information about the Windows platform.

* **Schedule feasibility:** Time evaluation is the most important consideration in the development of a project. The time schedule required for the development of this project is very important since more development time affects machine time, and costs and causes delays in the development of other systems.

## Applicability of our system

The Ethereum Smart Contract Network by Blockchain presents a highly applicable solution in the realm of decentralized finance (DeFi) and cryptocurrency management. Its unique features and benefits make it suitable for various scenarios and user groups:

* + - **Crypto Enthusiasts:** The wallet appeals to cryptocurrency enthusiasts who value security, autonomy, and direct control over their digital assets. By leveraging Ethereum's smart contracts, users can securely manage a diverse range of cryptocurrencies from a single platform.
    - **DeFi Participants:** The wallet is well-suited for users actively engaging in decentralized finance protocols. It allows direct interaction with DeFi platforms and dApps, enabling users to access lending, borrowing, yield farming, and other DeFi services without the need for third-party intermediaries.
    - **Financial Sovereignty Seekers**: Individuals who prioritize financial sovereignty and privacy will find the wallet's decentralized nature appealing.

Users maintainexclusive ownership of their private keys.

* + - **Cross-Border Transactions:** The wallet's utilization of blockchain technology streamlines cross-border transactions, enabling faster and more cost-efficient transfers without the complexities associated with traditional financial systems.
    - **Financial Inclusivity:** The wallet fosters financial inclusivity by providing access to financial services for individuals without access to traditional banking. It enables participation in global financial markets without geographic restrictions.
    - **Privacy and Security Advocates:** The smart contract wallet prioritizes security and transparency, making it an attractive option for users who prioritize privacy and want to avoid sharing personal information with centralized institutions.
    - **Innovators and Developers:** The wallet's open-source nature encourages collaboration and contributions from developers, promoting innovation in the blockchain ecosystem.
    - **Fostering DeFi Adoption:** As the decentralized finance landscape continues to grow, the wallet can contribute to the broader adoption of DeFi solutions by providing a secure and user-friendly gateway to decentralized financial services.
    - **Micropayments and Remittances**: The wallet's low transaction fees and efficiency make it applicable for micropayments and remittances, facilitating seamless and cost-effective transfer of funds.
    - **Decentralized Application Integration:** Developers of decentralized applications can integrate with the wallet, broadening their user base and offering users a convenient means to access their applications and services.

In summary, the Ethereum Smart Contract Wallet using React holds wide applicability across various user groups, industries, and financial use cases. Its emphasis on security, transparency, and user-friendliness makes it an appealing choice for crypto enthusiasts, DeFi participants, privacy advocates, and those seeking a more decentralized and inclusive financial ecosystem [3].

## Summary

Here in this project, we have discussed the background history of this project. It shows the challenges, feasibility, and applicability of our project.

# Chapter 3 Proposed System

## Introduction

In this chapter, we will discuss the design and details architecture of our project.

## Blockchain Technology

Blockchain is a digital and distributed ledger of transactions or decentralized database that holds continuously updated digital records in real-time through a network of computers. A Google Doc is a basic analogy for blockchain comprehension. But a blockchain provides a decentralized distribution chain that allows everyone to read the text simultaneously. Of course, it is more complicated than Google Docs, but the analogy is fair since three significant technological concepts are highlighted. A quick overview of blockchain:

* Digital properties are stored rather than copied or moved
* The asset is decentralized, providing direct access to the property in real-time
* A straightforward ledger of modifications maintains the credibility of the record, ensuring confidence in the asset

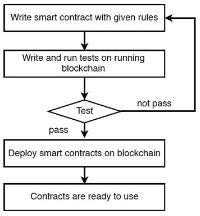
Blockchain is a fascinating and innovative technology that reduces risk and eliminates fraud and accountability for numerous applications. Based on the pair-to-pair (P2P) topology, blockchain is a technology that enables data to be stored globally on multiple servers – enabling everyone on the network to access the entries of anyone else in almost live time. This makes it really complicated for a user to track a network. Blockchain aims to provide businesses with transactional accountability. It has the potential to build reliable real-time communication networks to help partners all over the world.Towards a quick increase in application development and pilot testing in a variety of industries,Blockchain has been powered mainly by financial technology (fintech) investments and produced revenue of more than $10.6 billion dates by 2023, according to an ABI research report. The bulk of this revenue is anticipated from products and services of software [4].

## Proposed Blockchain System Architecture

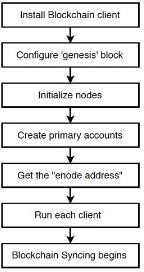
The proposed blockchain system architecture for a smart contract application will be designed based on the specific requirements of the use case. However, here is a general overview of the components that may be included in a typical blockchain system architecture:

* + - **User Interface:** This component provides the user interface for interacting with the smart contract application. It can be a mobile application, web application or desktop application.
    - **Client:** This component interacts with the blockchain network and sends transactions to execute the smart contract. It can be a lightweight client or a full node.
    - **Smart Contract:** This component contains the code that implements the businesslogic of the application and is executed on the blockchain network.
    - **Blockchain Network:** This component is the underlying infrastructure that provides the distributed ledger for the smart contract application. It can be a public blockchain such as Ethereum or a private blockchain such as Hyperledger Fabric.
    - **Consensus Protocol:** This component ensures that the blockchain network is secure and that all nodes agree on the state of the ledger.
    - **Smart Contract Storage:** This component stores the state of the smart contract on the blockchain network. It can be implemented as a key-value store or a relational database.
    - **External Data Sources:** This component provides external data to the smart contract, which can be used to trigger the execution of the contract or to update its state.
    - **Wallet:** This component provides the means for users to manage their digital assets and interact with the blockchain network.
    - **Security and Auditing:** This component ensures that the smart contract application is secure and that all transactions are auditable [5].

The proposed blockchain system architecture for a smart contract application will be designed with the aim of achieving the desired functionality, performance, and security of the application while maintaining the reliability and scalability of the underlying blockchain network.



**Fig.3.3(b): Smart contracts implementation workflow**



**Fig.3.3(a): Blockchain implementation workflow**

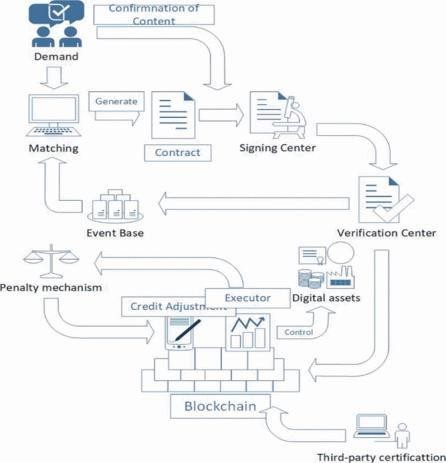
## Modeling

In this part, we divide smart contracts model into execution model, structure model and state model, and propose a classification method of smart contracts. The execution model is used to explain the operation mechanism of each contract in the smart contract system. To facilitate computer execution of contracts, a method of dividing each contract into several sub-contracts is proposed. We assume that all contracts are error- free, there is no ambiguity, and the contract is unique in the split result. Because

different asset types have diverse calling methods and detection methods, different types of contracts control different types of assets to protect the contract's control over assets. The state model of the contract represents the whole life cycle of a smart contract from deployment and execution to final termination. When the contract is in different states, the control content that needs to be accessed is different, and the control state of its external interface is different under different states. It is worth to know that the contract state here refers to the entire contract.

## Execution Model

The execution of smart contracts can be divided into four stages: signing, deploying, executing and ending. The execution model of smart contracts is shown in Figure 4.4.1. After the user makes a request, the contract text is automatically generated. After the two parties sign the confirmation, the contract needs to be verified by the verification tool to be deployed to the contract executor on the blockchain to start running. If it is a new contract, it is saved in the contract event library as a backup. During the operation of the contract, various digital assets are manipulated according to the contract content. If there is a breach of contract, the penalty is imposed according to the contract, and the credit situation is updated; if there is a dispute, the third party (usually the authority) is involved in the forensics.

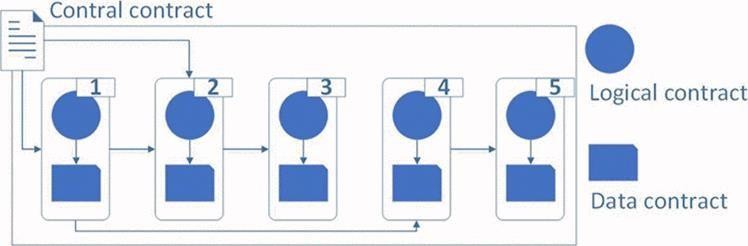


**Fig.3.4.1 Execution model of smart contract**

## Structural Model

Analogous to the relationship between the main contract and the subordinate contract existing in the traditional contract, we divide the model design of the smart contract into control contract and based contract, in which the based contract can be divided into logical contract and data contract in the end. The method of contract splitting will be instructed in the next section. The logical contract does not store the state, focus on the running function and running state; the data contract provides the data structure

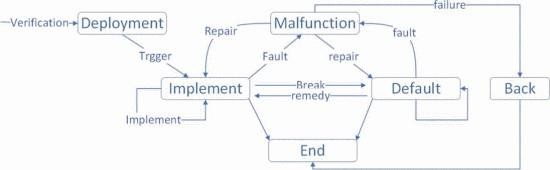
definition; stores the interface for reading and writing data; saves the assets, status, user and other data to the blockchain; and can be from the blockchain



## State Model

**Fig.3.4.2 Structural model of smart contract**

According to the possible situations during the contract operation, we divide the contract into five states: deployment, execution, failure, default and cancellation, as shown in Figure 6. Inside, deployment is pre-run state. The system needs to ensure that no deadlock is found in the contract[6].



## Summary

**Fig.3.4.3 State model of smart contract**

We demonstrated our proposed system with complete design in this chapter .These are the building blocks of this project.

# Chapter 4 Methodology

## Introduction

In this chapter, we will briefly discuss the methodology we have used in our project. There are a number of software development models in software engineering. Of them, we have used Agile Software Development Model. We will explain exactly why we used that particular model in our project.

## Our Used Methodology

In our project, we will use the “Agile Software Development” model. There are a number of properties in the Agile Model that helps us to build our project “Ethereum Smart Contact Network by Blockchain”. We will discuss why it is best for our project throughout this chapter.

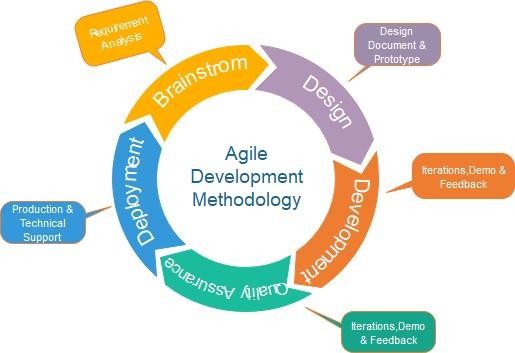
## Agile Software Development

Agile software development is a methodology for software development that prioritizes flexibility, collaboration, and iterative development. It emphasizes the importance of delivering working software quickly and adapting to changing requirements and priorities.

The Agile approach is based on a set of values and principles outlined in the Agile Manifesto, which emphasizes:

* + - * Individuals and interactions over processes and tools
      * Working software over comprehensive documentation
      * Customer collaboration over contract negotiation
      * Responding to change over following a plan

Agile software development is typically organized into short, time-boxed iterations or sprints, during which a team of developers work on specific tasks and deliver working software that meets the needs of the customer or end-user. The team collaborates closely with stakeholders, including the product owner, to ensure that the software meets their needs and is delivered on time and within budget.

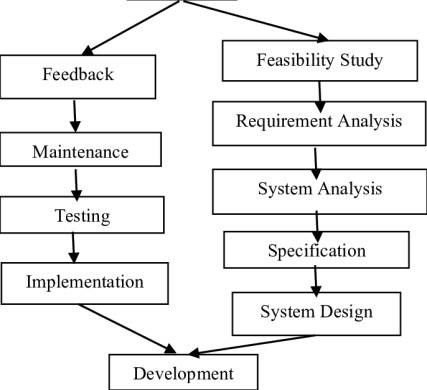


**Fig 3.2.1: Agile Software Development Method**

## Advantages of Agile Method

Some of the key advantage of Agile software development include:

* + - * Faster delivery of working software
      * Increased flexibility and adaptability to changing requirements
      * Improved collaboration between team members and stakeholders
      * Greater customer satisfaction and value delivery
      * Reduced risk of project failure due to early detection of issues and defects.



**Fig 3.2.2: Advantage of Agile Method**

## Why We Choose Agile Model

Agile software development can be an effective methodology for developing smart contracts. Like other software development projects, the development of smart contracts involves multiple stages, from design and development to testing and deployment. Agile software development can help ensure that these stages are executed efficiently and effectively, while also allowing for flexibility and adaptation to changing requirements.

In an Agile development environment, the development of a smart contract can be broken down into a series of iterations or sprints, each focused on delivering a specific set of features or functionality. The team can work closely with the product owner or stakeholders to ensure that the smart contract meets their needs and is delivered on time and within budget.

One of the key benefits of Agile development in the context of smart contracts is the ability to quickly iterate and make changes based on feedback or changes in requirements. Because smart contracts are stored on a blockchain network, they cannot be easily modified once they are deployed. Therefore, it is important to ensure that the smart contract meets the requirements of all stakeholders before it is deployed. Agile development can help ensure that this happens by allowing for continuous feedback and iteration throughout the development process.

Overall, Agile software development can be an effective methodology for developing smart contracts, helping to ensure that they are delivered efficiently and effectively, while also allowing for flexibility and adaptation to changing requirements.

## Summary

In this chapter, we have discussed the methodology we used for our project. In this project, we have used the agile software development model. Here we have explained that why we used the agile software development model and mentioned the benefits of this model over other models.

# Chapter 5 System Specification

## Introduction to requirements

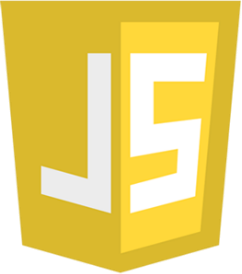
The process to gather the software requirements from clients and analyze and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain a sophisticated and descriptive ‘System Requirements Specification’ document.

## Development Requirements

* Intel Core i3 7th Generation or Upper
* RAM (At least 8GB)
* Code Editor (VS code or others)
* Web Browser (Google chrome / Firefox)
* MetaMask Extension
* Ganache
* React JS
* CSS

## JavaScript

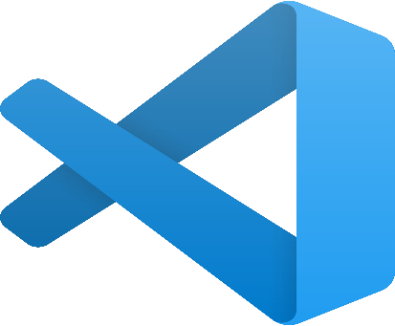
JavaScript is a versatile and widely-used programming language that empowers web developers to create dynamic and interactive websites. As a client-side language, it runs directly within web browsers, enabling enhanced user experiences and interactivity. Its lightweight nature and ease of integration with HTML and CSS make it the backbone of modern web development. JavaScript's event-driven and asynchronous nature allows developers to handle user actions efficiently and perform tasks in the background, ensuring a responsive user interface. Moreover, its extensive ecosystem of libraries and frameworks, such as React, Angular, and Vue.js, further simplifies web development, enabling the creation of complex and feature-rich applications. As JavaScript evolves, it continues to be a fundamental language driving innovation and pushing the boundaries of web development and the broader realm of software engineering.



## VS Code

Visual Studio Code (VS Code) is a powerful and popular source code editor developed by Microsoft. It has gained widespread adoption among developers due to its versatility, extensibility, and ease of use. VS Code supports multiple programming languages, offering robust code editing features like syntax highlighting, auto-completion, and intelligent code suggestions. Its integrated terminal enables seamless command-line interactions within the editor, streamlining the development workflow. Moreover, a

vast ecosystem of extensions allows developers to customize and enhance their coding experience, catering to specific needs and technologies. With its frequent updates and responsive community support, Visual Studio Code continues to be a top choice for developers across various domains, making it a go-to tool for efficient and productive software development.



## Web Browser

A web browser is a software program that allows a user to locate, access, and display web pages. In common usage, a web browser is usually shortened to "browser." Web browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML). Browsers translate web pages and websites delivered using Hypertext Transfer Protocol (HTTP) into human-readable content. They also have the ability to display other protocols and prefixes, such as secure HTTP (HTTPS), File Transfer Protocol (FTP), email handling, and files.

## MetaMask

MetaMask is a widely-used browser extension that serves as a crucial gateway to the world of decentralized applications (dApps) and blockchain networks. It acts as a digital wallet, allowing users to securely manage their cryptocurrencies and interact with Ethereum-based dApps directly from their web browsers. By seamlessly integrating with popular browsers like Chrome and Firefox, MetaMask enables users to access decentralized finance (DeFi) services, participate in token sales, and execute smart contracts without the need for additional software or installations. Its user-friendly interface and robust security features, like private key management, give users complete control over their digital assets and transactions. MetaMask has played a pivotal role in popularizing decentralized applications, empowering users to navigate the decentralized web with ease and convenience[5].



## Web 3.0

Web3, short for "Web 3.0," refers to the next generation of the internet, which aims to decentralize data, applications, and user interactions. At its core, Web3 is centered around blockchain technology, particularly public blockchains like Ethereum. It envisions a future where users have more control over their data and digital assets, enabling peer-to-peer interactions without relying on centralized intermediaries. Web3 introduces concepts like decentralized applications (dApps), smart contracts, and non- fungible tokens (NFTs) to revolutionize how we interact with digital content and services. With Web3, users can participate in a global, trustless economy, execute programmable transactions, and engage in decentralized finance (DeFi) protocols. As the technology matures, Web3 has the potential to reshape the internet, emphasizing .



## CSS

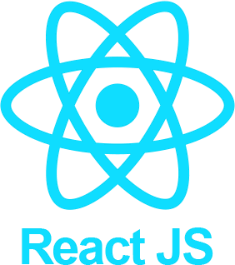
CSS (Cascading Style Sheets) is a fundamental web technology used to control the presentation and layout of HTML documents. It enhances the aesthetics and visual appeal of websites, enabling developers to apply styles, colors, fonts, and positioning to HTML elements. By separating content from presentation, CSS promotes a clean and organized structure, making it easier to maintain and update web pages. Its ability to create responsive designs allows websites to adapt seamlessly to different screen sizes and devices, improving user experience. CSS is continually evolving with new features and capabilities, ensuring that web developers can stay at the forefront of design trends and deliver visually stunning and user-friendly websites. With CSS, developers can transform plain HTML into engaging and captivating web experiences that leave a lasting impression on visitors.



## React JS

React.js, commonly known as React, is a powerful and widely-used JavaScript library for building modern and interactive user interfaces (UIs). Developed and maintained by Facebook, React follows a component-based architecture that allows developers to create reusable UI components, making code more maintainable and efficient. It efficiently updates and renders components based on changes in data, optimizing

performance for dynamic applications. React's virtual DOM concept minimizes direct interaction with the actual DOM, resulting in faster rendering and improved user experience. Additionally, React's popularity is fueled by its strong ecosystem, which includes a wealth of libraries, tools, and community-driven support. As a result, React has become a go-to choice for web developers seeking to build responsive, scalable, and feature-rich web applications.



## Ganache

Ganache is a popular and powerful personal blockchain emulator used primarily for Ethereum development and testing purposes. It provides a local blockchain environment that allows developers to deploy and interact with smart contracts, create and manage accounts, and simulate various blockchain scenarios without the need for an actual network. Ganache offers a user-friendly interface, real-time blockchain visualization, and detailed transaction logs, making it easy to debug and analyze smart contract interactions. Its integration with popular development frameworks like Truffle and Remix enhances the efficiency of the development process. Whether used for testing, prototyping, or learning Ethereum development, Ganache has become an indispensable tool for blockchain developers, offering a safe and controlled environment for blockchain experimentation and application development.

## Summary

Here in this chapter, we have discussed the hardware and software that are needed to build our project. It includes programming language IDE, framework, editor, etc. we have also discussed the dependencies and databases that are needed to build our project.

**Chapter 6 Design of Our System**

## Introduction to system overview

Agile Model is a process of software development where requirements are broken down into multiple standalone modules of the software development cycle. Incremental development is done in steps from analysis, design, implementation, testing/verification, and maintenance. In this chapter we will show all the diagrams that describe our project properly. The diagrams are –Flowchart , DFD, ER diagram, workflow diagram, use case diagram, sequence diagram

## Workflow Diagram

A workflow diagram is a visual representation of a series of steps or processes necessary to complete a task or achieve a specific goal. It uses symbols and shapes to illustrate the flow of information or materials within a system. Workflow diagrams are valuable tools for understanding, analyzing, and improving complex workflows. They help identify bottlenecks, inefficiencies, and potential areas of improvement, leading to enhanced productivity and streamlined operations. By providing a clear and concise overview of a process, workflow diagrams facilitate effective communication among team members and stakeholders.

Initiate Account

Generate Account in MetaMask

Request Transaction

Receiver Private Key/Amount

Select Language

Send Ethereum

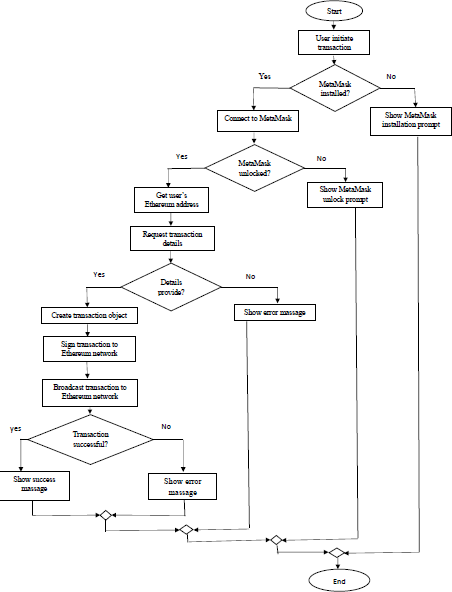
Terminate

**Fig 6.2: Workflow Diagram of**

**Ethereum Smart Contract Network by Blockchain**

## Flow Chart

A website flowchart (also known as a sitemap) maps out the structure and complexity of any current or future website. A well-structured sitemap or flowchart makes your website easily searchable. Each piece of content should ideally give users accurate search results, based on keywords connected to your web content.



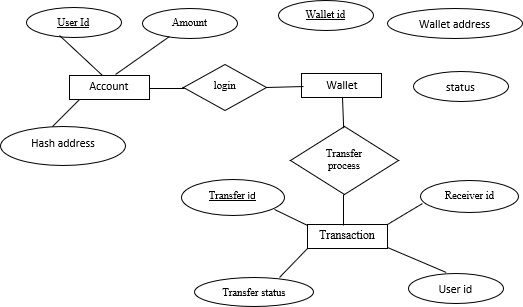
**Fig 6.3: Flow chart of**

**Ethereum Smart Contract Network by Blockchain**

## E-R Diagram

An Entity-Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects, or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education, and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals, and connecting lines to depict the interconnectedness of entities, relationships, and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs. The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

* **One to one** – When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So, the relationship will be one-to-one.
* **Many to one** – When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course, but one course can be taken by many students. So, the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.
* **Many to many** – When entities in all entity sets can take part more than once in the relationship, cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So, the relationship will be many to many.



**Fig 6.4: E-R Diagram of**

**Ethereum Smart Contract Network by Blockchain**

## Data Flow Diagram

### DFD Level - 0:

It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

send

receive

receive

Ethereum

system

send

User 1

User 2

**Fig 6.5.1: DFD level-0 of**

**Ethereum Smart Contract Network by Blockchain**

### DFD Level-1:

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and break down the high-level process of 0-level DFD into subprocesses.

Transection

wallet

Module check

User 1

Login

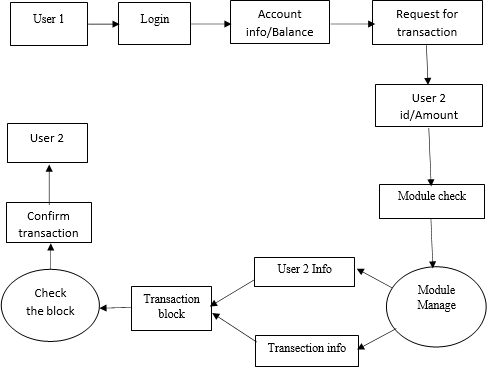
User 2

**Fig 6.5.2: DFD level-1 of**

**Ethereum Smart Contract Network by Blockchain**

### DFD Level-2:

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning

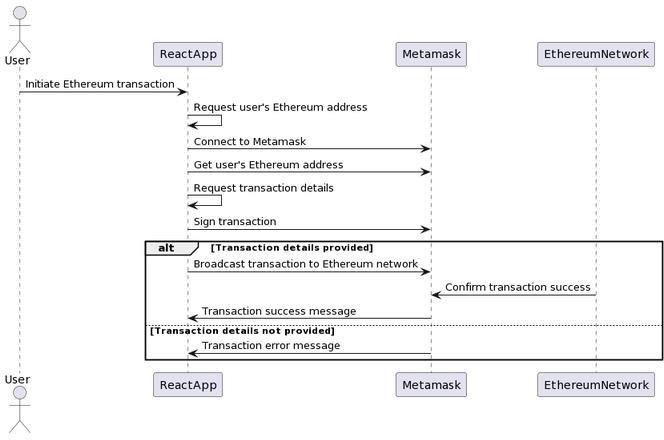


**Fig 6.5.3: DFD level-2 of**

**Ethereum Smart Contract Network by Blockchain**

## Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.



**Fig 6.7: Sequence diagram of**

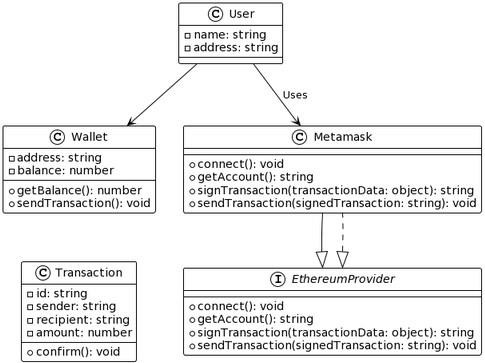
**Ethereum Smart Contract Network by Blockchain**

## Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed. In the diagram, classes are represented with boxes that contain three compartments:

* The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned, and the first letter is lowercase.
* The bottom compartment contains the operations the class can execute.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modeling, the classes of the conceptual design are often split into subclasses.

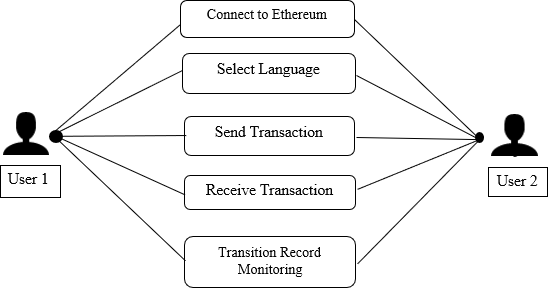


**Fig 6.7: Class diagram of**

**Ethereum Smart Contract Network by Blockchain**

## Use CASE Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent: Scenarios in which your system or application interacts with people, organizations, or external systems.



**Fig 6.7: Use Case diagram of**

**Ethereum Smart Contract Network by Blockchain**

## Summary

Here in this chapter, we have discussed the design models that are being followed to build our project. It includes a data flow diagram, flowchart, E-R diagram, use case diagram, class diagram, etc.

# Chapter 7 Implementation

## Introduction

Implementation is the process of building the web according to its design. A website implementation created by React JS, CSS and JavaScript.

## Login Page

Overview. We'll design a login page using React JS and CSS. In login page ,collects account information from the user and has a submit button to send the details for server. Then it will connect with user’s MetaMask account. A site's login page is a place where website users enter their credentials to gain access to site. These visitors may have an account on site, or some other reason for needing to log in to website.



* + 1. **Login : Connect**



## Home Page

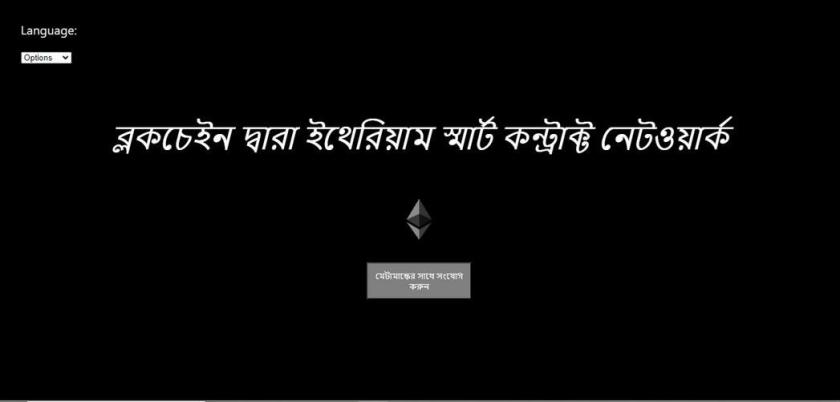
A home page is the default or front page of a site. It is the first page that visitors see when they load a URL. Web users can control the home page as a way of directing the user experience. In here user can transact Ethereum through a button.



## Language Selection

In here, user can select multiple language according their preference. Here is some sample of multiple language design.

* English Language
* Bangla Language



* Hindi Language



* Chinese Language



* Japanese Language



* Spanish Language



* German Language



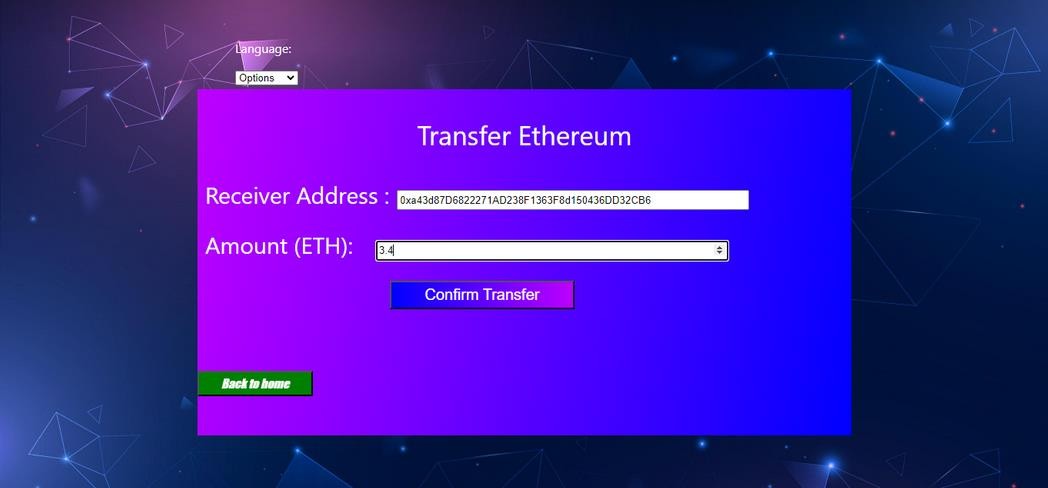
* Arabic Language



## Transaction page

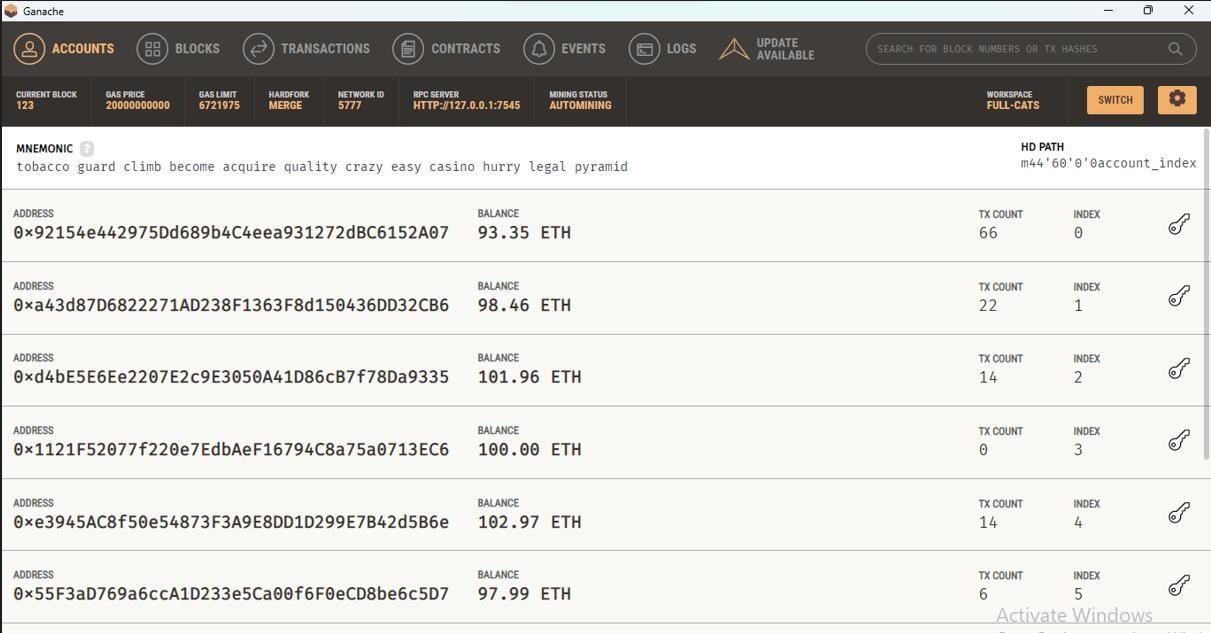
The transaction page is a web page that allows one user to transact Ethereum to another user easily and securely. After giving receiver id and amount there is a confirm button for transfer Ethereum.





## Record Page

In here user can see their transaction record, hash block. Gas limit. Balance etc.



## Summary

Here in this chapter , we have discussed the implementation and source code that build in our project.It include login page, home page, different language , transaction page and transaction record page.

**Chapter 8**

**Future Enhancement & Conclusion**

## Introduction

In this final chapter, we will bring a conclusion and future enhancement of our project report.

## Limitation of Our Project

Here are some of the limitations of Ethereum Smart Contact Network by Blockchain project.

* + - It is not open source.
    - It is not supported by all browsers.

## Future Enhancement

* Multi-Currency Support
* Offline Transaction Signing

## Conclusion

The Ethereum Smart Contact Networks by Blockchain signifies a breakthrough in decentralized finance, uniting traditional financial systems with blockchain's potential. This achievement, despite challenges, empowers users to securely manage digital assets, paving the way for a more transparent and inclusive financial landscape. As it emerges, the wallet epitomizes a pivotal stride towards a user-centric, blockchain- powered future.

## Summary

Here in this chapter, we have discussed the future enhancement and conclusion of our project.

## REFERENCES

 Hasan, H.R., Salah, K., 2018. Proof of delivery of digital assets using blockchain andsmart contracts, IEEE Access, vol. 6, pp. 65-439–65-448.

 White paper ethereum/wiki wiki github, https://github.com/ethereum/wiki/wiki/White-Paper.

 Ethereum, Ethereum is a global, open- source platform for decentralizedapplications,https://ethereum.org/.

 . Hasan, H.R., Salah, K., 2018. Blockchain-based solution for proof of delivery of physical assets. In: International Conference on Blockchain, Springer, pp. 139–152.

 Metamask, A crypto wallet & gateway to blockchain apps, https://metamask.io/.

 Salah, K., Rehman, M.H.U., Nizamuddin, N., Al-Fuqaha, A., 2019. Blockchain for ai:Review and open research challenges. IEEE Access, vol. 7, pp. 10 -127– 10-149.

 Winters T (2016) Decentralized exchange on the Ethereum blockchain. <https://www.ethnews.com/>decentralized-exchange-on-the-ethereum-blockchain

 Bartoletti M, Pompianu L (2017) An empirical analysis of smart contracts: Platforms, applications, and design patterns. In: International Conference on Financial Cryptography and Data Security, Springer, pp 494–509

 Dangl A Top trends 2020: Hyperautomation and smart contracts. Available online at [https://www.fabasoft.com/en/news/blog/top-trends-2020-](https://www.fabasoft.com/en/news/blog/top-trends-2020-hyperautomation-and-smart-contracts) [hyperautomation-and- smart-contracts](https://www.fabasoft.com/en/news/blog/top-trends-2020-hyperautomation-and-smart-contracts) (2019).

 DHarz D, Knottenbelt W (2018) Towards safer smart contracts: A survey of languagesand verification methods. arXiv preprint arXiv:[1809.09805](http://arxiv.org/abs/1809.09805)

 . Dickerson T, Gazzillo P, Herlihy M, Koskinen E (2019) Adding concurrency to smartcontracts. Distrib Comput 33:1–17

## Appendix

**Language Selection Code**

import "./App.css";

import Web3 from "web3";

import { useTranslation } from "react-i18next";

const languages = [

{ value: "", text: "Options" },

{ value: "en", text: "English" },

{ value: "hi", text: "Hindi" },

{ value: "bn", text: "Bengali" },

{ value: "ar", text: "Arabic" },

{ value: "zh", text: "Chines" },

{ value: "ja", text: "Japanese" },

{ value: "de", text: "German" },

{ value: "fr", text: "French" },

{ value: "el", text: "Greek" },

{ value: "it", text: "Italic" },

{ value: "es", text: "Spanish" },

{ value: "ta", text: "Tamil" },

{ value: "sw", text: "swahili" },

];

const MetamaskLogin = () => {

const [account, setAccount] = useState(""); const [balance, setBalance] = useState("");

const fullText = "Ethereum smart contract network by blockchain";

const nevigate = useNavigate();

const { t } = useTranslation();

const [lang, setLang] = useState("");

// This function put query that helps to

// change the language

const handleChange = (e) => { setLang(e.target.value);

let loc = "http://localhost:3000/"; window.location.replace(loc + "?lng=" + e.target.value);

};

## MetaMask Checking Code

//this funcion would check weather metamask is available or not, const checkMetamask = async () => {

if (typeof window.ethereum !== "undefined") { try {

//will request for network account

await window.ethereum.request({ method: "eth\_requestAccounts" }); const web3 = new Web3(window.ethereum);

//get all accounts in accounts array

const accounts = await web3.eth.getAccounts(); setAccount(accounts[0]); //set the first account of array as login

const balanceInWei = await web3.eth.getBalance(accounts[0]); //set balence of that account

setBalance(web3.utils.fromWei(balanceInWei, "ether"));

} catch (error) { console.error(error);

}

} else {

console.log("Metamask not installed");

}

};

## Transfer Page Code

const TransferEth = () => {

const [toAddress, setToAddress] = useState(""); const [amount, setAmount] = useState(""); const nevigate = useNavigate();

const transferEth = async () => { try {

// Check if Web3 is available

if (typeof window.ethereum !== "undefined") { const web3 = new Web3(window.ethereum); await window.ethereum.enable();

const accounts = await web3.eth.getAccounts();

const from = accounts[0]; //set the from Adress as loged in account

// Create the transaction object if(toAddress=='' || amount==''){

alert('values are not properly inserted !') return;

}

const transaction = { from,

to: toAddress,

value: web3.utils.toWei(amount, "ether"),

};

// Send the transaction

await web3.eth.sendTransaction(transaction);

alert("Ethereum transferred successfully!"); nevigate("/");

} else {

alert("Please install Metamask or use a Web3-enabled browser.");

}

} catch (error) {

alert("An error occurred while transferring Ethereum. Please try again."); console.error(error);

}

};