A

Major Project Report

On

Decentralized Payment Network: Blockchain Money Transactions

*Submitted to **CMREC, HYDERABAD***

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted
By

K. Akshitha	**(218R1A6736)**
P. Giridhar	**(218R1A6750)**
B. Chiranjeevi	**(218R1A6716)**
P. Arun	**(218R1A6751)**

Under the Esteemed guidance of

Mr. E. Laxman

Assistant Professor, Department of CSE (Data Science)

Department of Computer Science & Engineering (Data Science)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad) Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NBA,Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering(Data Science)

CERTIFICATE

This is to certify that the project entitled **"Decentralized Payment Network: Blockchain Money Transactions"** is a bonafide work carried out by

K. Akshitha	**(218R1A6736)**
P. Giridhar	**(218R1A6750)**
B. Chiranjeevi	**(218R1A6716)**
P. Arun	**(218R1A6751)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**	**External Examiner**
Mr. E. Laxman	**Mrs. G. Shruthi**	**Dr. M. Laxmaiah**	
Assistant Professor CSE (Data Science), CMREC	Assistant Professor CSE (Data Science), CMREC	Professor & H.O.D CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled "**Decentralized Payment Network : Blockchain Money Transactions”** is a record of bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source.We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

K. Akshitha	**(218R1A6736)**
P. Giridhar	**(218R1A6750)**
B. Chiranjeevi	**(218R1A6716)**
P. Arun	**(218R1A6751)**

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, HOD,

Department of CSE (Data Science), CMR Engineering College for their constant support**.**

We are extremely thankful to **Mr. E. Laxman,** Assistant Professor, Internal Guide, Department of CSE(DS), for his/ her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G. Shruthi**, Assistant Professor, CSE(DS) Department**,** Major Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, we are very much thankful to our parents who guided me for every step.

K. Akshitha (218R1A6736)
P. Giridhar (218R1A6750)
B. Chiranjeevi (218R1A6716)
P. Arun (218R1A6751)

ABSTRACT

In recent years, blockchain technology has revolutionized digital transactions by offering a decentralized, transparent, and secure method of transferring assets without intermediaries. The **Decentralized Payment Network: Blockchain Money Transactions (CRYPT)** project aims to harness the power of blockchain to facilitate seamless peer-to-peer financial transactions. Traditional payment systems often involve high transaction fees, delays, and security risks due to centralized control, making blockchain-based solutions a promising alternative.

CRYPT is a Web 3.0 application designed to provide users with a decentralized, trustless payment system using Ethereum smart contracts. The system enables users to register with an email and password and subsequently connect their MetaMask wallet for transactions. Utilizing Solidity for smart contract development and Hardhat as a testing and deployment framework, the project is built on the Sepolia Ethereum test network through Alchemy's blockchain infrastructure. The integration with MetaMask ensures secure and user-controlled fund transfers, enhancing privacy and eliminating the need for intermediaries.

One of the core features of CRYPT is its ability to process real-time transactions with transparency and immutability. The Ethereum blockchain records each transaction in a public ledger, preventing fraudulent activities and unauthorized modifications. When a transaction is initiated, the user confirms it through MetaMask, and a loading state is displayed until confirmation is received.

Security and efficiency are the primary focuses of the CRYPT network. By leveraging blockchain's decentralized nature, the project mitigates risks associated with centralized payment gateways, such as data breaches and unauthorized access. Additionally, the use of smart contracts automates the execution of transactions, reducing operational costs and human intervention. The decentralized payment network developed in CRYPT showcases the potential of blockchain technology in transforming digital payments. It provides a scalable, efficient, and secure solution that empowers users with full control over their financial transactions. Future enhancements may include multi-chain support, advanced transaction analytics, and integration with real-world payment systems to bridge the gap between blockchain-based finance and traditional banking systems.

CONTENTS

LIST OF FIGURES

1. Introduction

1.1 Overview

The evolution of digital payments has significantly altered the way financial transactions are conducted, shifting from traditional banking methods to online and mobile payment platforms. However, centralized financial systems continue to pose several challenges, including high transaction costs, long processing times, and security vulnerabilities. The emergence of blockchain technology has introduced a decentralized alternative that offers enhanced security, transparency, and efficiency in digital transactions.

This decentralized payment network is designed to address these challenges by leveraging blockchain technology for peer-to-peer payments. Unlike conventional banking systems, this system operates without intermediaries, ensuring that transactions are direct, cost-effective, and secure. The platform is built on the Ethereum blockchain, utilizing smart contracts to automate and validate transactions, thereby eliminating the need for third-party verification.

A key component of this system is its integration with MetaMask, a widely used cryptocurrency wallet that enables users to securely connect to decentralized applications. This integration allows users to initiate and confirm transactions with ease while maintaining full control over their funds. Every transaction is recorded on the Ethereum blockchain, ensuring immutability and transparency, which enhances trust among users.

Furthermore, this system introduces a **Latest Transactions** feature that provides users with real-time transaction history, promoting financial accountability and tracking. By using the Sepolia test network through Alchemy’s blockchain infrastructure, this system ensures a seamless and efficient transaction process during the development phase before deploying to a mainnet environment.

The project aims to revolutionize digital payments by offering an accessible, decentralized, and efficient solution for financial transactions. As blockchain adoption grows, this system has the potential to bridge the gap between traditional finance and decentralized payment systems, paving the way for a future where users have greater control over their financial activities.

To ensure the accuracy and reliability of the expert system, it should undergo rigorous testing and validation with real patient cases, and continuous updates should be made to the knowledge base as new research and treatment methods emerge.

1.2 Research Motivation

The growing reliance on digital transactions has highlighted critical limitations in traditional financial systems, including inefficiencies, high transaction fees, and security vulnerabilities. Centralized payment gateways, which act as intermediaries in financial transactions, often introduce delays, impose restrictions, and expose users to risks such as fraud, data breaches, and unauthorized access. These concerns have fueled the demand for alternative solutions that offer greater security, transparency, and financial autonomy.

Blockchain technology presents a transformative opportunity by providing a decentralized, secure, and transparent method for conducting financial transactions. By leveraging smart contracts and cryptographic security mechanisms, blockchain-based payment systems eliminate the need for intermediaries, thereby reducing transaction costs and processing times. The potential to facilitate direct, peer-to-peer transactions in a trustless environment has made blockchain an attractive solution for digital payments.

This research is motivated by the need to explore and develop a decentralized payment network that can address the shortcomings of traditional financial infrastructures. By integrating Ethereum smart contracts with a user-friendly interface, this system ensures secure and verifiable transactions while giving users full control over their funds. Unlike conventional banking systems, which require extensive verification processes and centralized authorization, blockchain-based payment networks operate on a distributed ledger, enhancing both accessibility and efficiency.

Additionally, the rise of Web 3.0 and decentralized finance (DeFi) has demonstrated the potential of blockchain in reshaping global financial interactions. With an increasing number of users adopting digital assets and decentralized applications, there is a need to create scalable and robust payment solutions that align with these emerging trends.

The objective of this research is to design and implement a decentralized payment system that enhances financial security, ensures seamless user experience, and promotes broader adoption of blockchain-based transactions. By analyzing existing limitations in digital payments and exploring innovative blockchain solutions, this study contributes to the advancement of decentralized financial technologies and their real-world applications.

1.3 Problem Statement

The problem statement for the development of a decentralized payment network using blockchain technology can be articulated as follows:

Traditional financial systems rely on centralized authorities such as banks and third-party payment processors to facilitate transactions. While these systems provide essential financial services, they are often associated with high transaction fees, long processing times, security vulnerabilities, and financial restrictions. Users must place their trust in intermediaries to manage and verify transactions, which increases the risk of fraud, data breaches, and unauthorized access. Additionally, centralized financial institutions impose regulatory constraints and geographic limitations, making cross-border transactions expensive and inefficient.

Blockchain technology offers a decentralized alternative that enhances security, transparency, and efficiency in digital transactions. However, existing blockchain-based payment systems face challenges such as network congestion, high transaction costs (gas fees), and a steep learning curve for non-technical users. Many users find it difficult to interact with blockchain applications due to the complexity of wallet management, private key security, and smart contract execution. These barriers limit the widespread adoption of decentralized financial systems.

To address these issues, this study focuses on the development of a decentralized payment network that eliminates intermediaries, reduces transaction costs, and ensures seamless peer-to-peer transactions. By leveraging Ethereum smart contracts, this system provides a trustless, transparent, and immutable payment solution where users retain full control over their funds. The integration of MetaMask facilitates secure wallet connectivity, allowing users to perform transactions with ease while maintaining financial autonomy.

The key challenges that this research aims to address include ensuring transaction efficiency, enhancing the user experience, and improving the scalability of blockchain payments. By developing a robust, user-friendly, and decentralized payment network, this study seeks to contribute to the broader adoption of blockchain-based financial solutions, promoting a more secure, efficient, and accessible global payment ecosystem.

1.4 Applications

The development of a decentralized payment network using blockchain technology holds vast potential across various applications within the financial sector and beyond.

- **Peer-to-Peer Transactions:** One of the primary applications of the decentralized payment network is facilitating secure and direct peer-to-peer transactions. Users can send and receive funds without the need for intermediaries such as banks or payment processors. This reduces transaction fees and processing times while enhancing financial autonomy and security.
- **Cross-Border Payments:** Traditional cross-border transactions often involve high fees, currency conversion costs, and delays due to multiple intermediaries. A decentralized payment network allows for instant, low-cost international transactions, making global financial interactions more efficient and accessible. This is particularly beneficial for freelancers, businesses, and expatriates who frequently send or receive money across borders.
- **Decentralized Finance (DeFi) Integration:** The decentralized payment network can seamlessly integrate with DeFi applications, enabling users to participate in decentralized lending, borrowing, staking, and yield farming. By leveraging smart contracts, users can access financial services without relying on traditional banks, making financial tools more inclusive and efficient.
- **E-Commerce and Online Payments:** Businesses can adopt decentralized payment systems to accept cryptocurrency transactions, providing an alternative to traditional online payment gateways. This reduces dependency on centralized platforms like PayPal or credit card networks, lowering transaction fees and minimizing fraud risks. Additionally, businesses can benefit from the transparency and security offered by blockchain technology.
- **Remittances and Financial Inclusion:** Many individuals in underbanked or unbanked regions struggle with access to financial services. A decentralized payment network provides an inclusive financial solution that allows people to send and receive funds securely using just a digital wallet. This promotes financial empowerment and economic participation for populations without access to traditional banking.
- **Smart Contract-Based Payments:** Businesses and individuals can utilize smart contracts for automated and conditional payments. For example, payments can be executed only when predefined conditions are met, ensuring trust and reducing disputes in transactions such as real estate deals, contract-based employment, and supply chain management.

- **Tokenization and Micropayments:** Blockchain-based payments allow for tokenized transactions and micropayments, which are useful in digital content creation, gaming, and subscription-based services. Users can make small payments for digital goods, services, and in-app purchases without incurring high processing fees.
- **Secure and Transparent Transactions:** Every transaction within the decentralized payment network is recorded on the blockchain, ensuring immutability and transparency. This makes the system resistant to fraud, chargeback abuse, and unauthorized modifications, fostering trust among users.
- **Charity and Crowdfunding:** The decentralized payment network can be leveraged for transparent and traceable charitable donations and crowdfunding campaigns. Donors can track how funds are utilized, ensuring accountability and trust in nonprofit organizations and fundraising initiatives.
- **Future Financial Innovations:** As blockchain technology continues to evolve, decentralized payment networks can serve as a foundation for future financial innovations, including programmable money, automated taxation, and integration with traditional banking services to create a hybrid financial ecosystem.

2. Literature Survey

Blockchain technology has revolutionized the financial sector by providing a decentralized, transparent, and secure method for money transactions. Decentralized payment networks leverage blockchain to enable peer-to-peer transactions without intermediaries, reducing costs and enhancing security. Several studies have explored blockchain-based financial systems, focusing on transaction efficiency, scalability, and security. The concept of decentralized finance (DeFi) has grown significantly, providing alternative payment solutions without reliance on traditional banking infrastructure. The rise of cryptocurrencies like Bitcoin and Ethereum has further driven research into decentralized payment systems, highlighting their potential to disrupt conventional financial models.

One of the key aspects of decentralized payment networks is security. Traditional financial transactions rely on centralized institutions that act as intermediaries, making them vulnerable to cyberattacks and fraud. Blockchain, with its cryptographic principles and decentralized ledger, eliminates single points of failure, enhancing transaction security. Various consensus mechanisms, such as Proof of Work (PoW) and Proof of Stake (PoS), ensure the integrity and immutability of transactions. Nakamoto (2008) introduced Bitcoin as the first cryptocurrency utilizing PoW, laying the foundation for secure digital transactions. Later research focused on optimizing consensus mechanisms to improve transaction speed and energy efficiency, with Ethereum introducing PoS to reduce computational requirements. Scalability remains a critical challenge in blockchain-based payment systems. Bitcoin and Ethereum, the most widely used blockchains, face transaction processing limitations due to block size constraints and network congestion. Studies have explored solutions such as the Lightning Network, a second-layer protocol that facilitates faster micropayments off-chain, reducing the burden on the main blockchain. Similarly, Ethereum's transition to Ethereum 2.0 incorporates sharding, a technique that divides the blockchain into smaller segments to improve processing speed. Research on alternative blockchain frameworks, such as Directed Acyclic Graphs (DAGs) used in IOTA, aims to enhance scalability without compromising decentralization. Smart contracts play a crucial role in decentralized payment networks by enabling automated and trustless transactions. Introduced by Szabo (1994), smart contracts execute predefined conditions without intermediaries, ensuring transparency and efficiency. Ethereum popularized smart contract functionality, allowing developers to build decentralized applications (DApps) for financial transactions. Research has explored the security risks associated with smart contracts, such as reentrancy attacks and vulnerabilities in Solidity programming. Auditing tools and formal verification techniques have been developed to mitigate these risks, ensuring that smart contracts function as intended. Interoperability between different blockchain networks is another area of research in decentralized payment systems. With numerous blockchains operating independently, seamless transaction processing across multiple networks remains a challenge. Cross-chain communication protocols, such as Polkadot and Cosmos, aim to bridge different blockchains, facilitating smooth asset transfers. Studies have examined the effectiveness of atomic swaps, which enable direct cryptocurrency exchanges between users on different blockchains without intermediaries. The implementation of interoperability solutions enhances the adoption of decentralized payment networks, making them more practical for global financial transactions. The regulatory landscape for decentralized payment networks varies across jurisdictions. Governments and financial institutions have expressed concerns regarding money laundering, tax evasion, and consumer protection in blockchain transactions. Research has analyzed different regulatory approaches, from outright bans to progressive frameworks that integrate blockchain within existing financial systems. Countries like Switzerland and Singapore have embraced blockchain innovation with clear guidelines, while others impose restrictions due to concerns over financial stability. Compliance solutions, such as Know Your Customer (KYC) and Anti-Money Laundering (AML) protocols, have been integrated into blockchain networks to address regulatory challenges while maintaining decentralization.

User adoption of decentralized payment systems depends on ease of use and accessibility. Traditional banking systems provide familiar interfaces and consumer protections, whereas blockchain transactions require users to manage private keys and understand cryptographic principles. Studies have explored user-friendly wallet designs, decentralized identity management, and blockchain-based financial inclusion initiatives. Mobile-based blockchain wallets and biometric authentication mechanisms have been proposed to simplify user interactions while maintaining security. Research on integrating decentralized payment networks with existing financial systems has also gained attention, allowing users to switch between fiat and cryptocurrencies seamlessly. Decentralized payment networks also contribute to financial inclusion by providing banking services to unbanked populations. According to the World Bank, billions of people worldwide lack access to traditional banking, limiting their financial opportunities. Blockchain-based payment solutions offer an alternative by enabling direct peer-to-peer transactions without the need for intermediaries. Studies have highlighted the impact of cryptocurrencies in developing economies, where individuals use digital assets for remittances, microtransactions, and business transactions. Stablecoins, which are pegged to fiat currencies, provide a stable alternative to volatile cryptocurrencies, making them more suitable for everyday transactions. The environmental impact of blockchain transactions, particularly in PoW-based networks, has been a topic of debate. Bitcoin mining consumes significant energy due to its computational requirements, leading researchers to explore greener alternatives. PoS, Delegated Proof of Stake (DPoS), and other consensus mechanisms have been proposed to reduce energy consumption while maintaining security. Research on hybrid blockchain models combines the benefits of both PoW and PoS, optimizing energy efficiency without compromising decentralization. The development of sustainable blockchain solutions ensures the long-term viability of decentralized payment networks.

Recent advancements in blockchain technology continue to shape the future of decentralized payment networks. Layer 2 solutions, improved consensus algorithms, and regulatory developments contribute to the evolution of blockchain-based financial systems. Research on quantum-resistant cryptography addresses the potential threat of quantum computing to blockchain security, ensuring the longevity of decentralized transactions. The integration of artificial intelligence (AI) and blockchain has also been explored, with AI-driven fraud detection and predictive analytics enhancing transaction security and efficiency.

In conclusion, decentralized payment networks leveraging blockchain technology offer secure, transparent, and efficient financial transactions. Research in this domain has addressed key challenges such as security, scalability, interoperability, regulation, and user adoption. While significant progress has been made, ongoing advancements in blockchain technology and regulatory frameworks will determine the widespread adoption of decentralized payment solutions. The CRYPT project aims to contribute to this evolving landscape by developing a blockchain-based payment network that prioritizes security, efficiency, and accessibility, aligning with the latest research trends in decentralized finance.

Table 1. Strengths and Weaknesses of Blockchain-Based Decentralized Financial Models

Study	Domain	Task	Method	Strengths	Weaknesses
Blockchain Solutions for Mortgage Loan Origination	Mortgage Loan Processing	Improve transparency and reduce costs in loan origination	Blockchain-based mortgage system (Wipro Ventures)	Enhances security and efficiency in loan approval; reduces processing time	Limited scalability; lack of standardization in smart contracts; regulatory uncertainties
Decentralized Loan Management Application Using Smart Contracts on Blockchain	Loan Management	Prevent fraudulent activities in loan sanctioning	Ethereum-based smart contract system	Reduces fraud risk; increases automation and transparency	Limited integration with various loan products; high computational cost; scalability concerns
Blockchain Smart Contracts: Applications, Challenges, and Future Trends	Smart Contracts	Comprehensive analysis of blockchain-enabled smart contracts	Survey of technical and usage aspects	Identifies key challenges and future trends; highlights practical use cases	Limited real-world implementation examples; does not address sector-specific adoption barriers
Legality of Blockchain and Smart Contracts in India	Legal Studies	Analyze blockchain's compatibility with Indian regulations	Legal review of smart contract enforceability	Identifies gaps in Indian law regarding smart contracts; discusses legal implications	Lacks clarity on cross-border blockchain transactions; absence of well-defined regulatory policies

Secure Balance Planning of Off-blockchain Payment Channel Networks	Off-Blockchain Payment Networks	Optimize balance distribution in payment networks	Payment network planning (PnP)	Reduces reliance on third parties; enhances security; ensures efficient fund distribution	Scalability issues in larger networks; limited real-world testing; lacks flexibility for dynamic demands
Design and Implementation of a Cloud-Based Decentralized Cryptocurrency Transaction Platform	Cryptocurrency Transactions	Develop a decentralized crypto exchange	Multi-signature authentication & cryptocurrency API integration	Enhanced security; multi-signature authentication prevents unauthorized access	Limited focus on user experience; scalability issues; higher transaction latency
Regulating Decentralized Financial Technology: A Qualitative Study	Financial Regulation	Assess regulatory challenges in DeFi	Qualitative analysis of regulations	Highlights gaps in regulatory frameworks; provides insights into emerging financial models	Lacks concrete regulatory solutions; does not propose implementation strategies

3. EXISTING SYSTEM

3.1 Traditional Payment System vs. Decentralized Payment System

A traditional payment system requires a trusted third-party authority, such as a bank or financial institution, to authenticate and validate transactions. This approach has inherent risks, such as single points of failure, security vulnerabilities, and restricted access based on geography or banking regulations. In contrast, a decentralized payment network eliminates intermediaries by leveraging blockchain technology. Transactions are verified by a distributed network of nodes, ensuring transparency, security, and immutability.

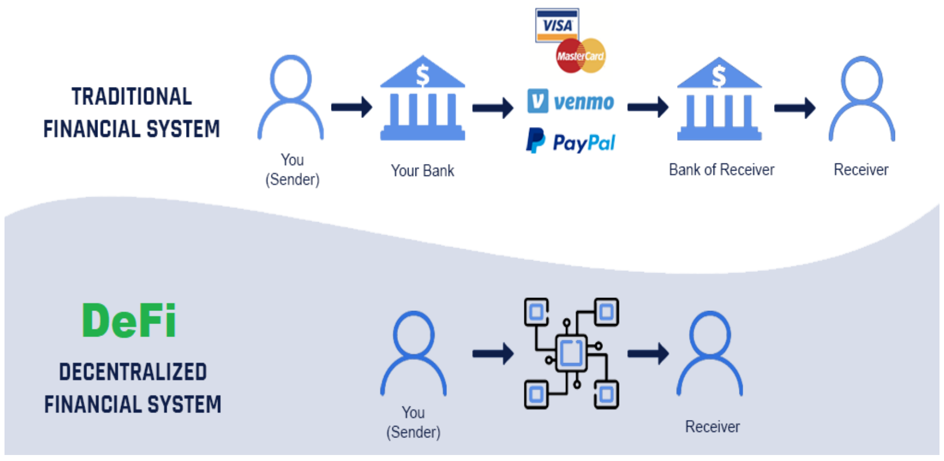


Figure 3.1: Traditional vs. Decentralized Payment System

3.2 Blockchain Transaction Flow

In decentralized payment systems, blockchain technology ensures secure and transparent transactions through a distributed ledger. Each transaction is recorded in a block and validated through consensus mechanisms such as Proof of Work (PoW) or Proof of Stake (PoS). The transaction flow consists of the following steps:

1. The sender initiates a transaction by signing it with their private key.
2. The transaction is broadcast to the blockchain network.
3. Miners or validators verify the transaction.
4. The verified transaction is added to a block and linked to the previous block.
5. The updated blockchain ledger is distributed across all network participants.

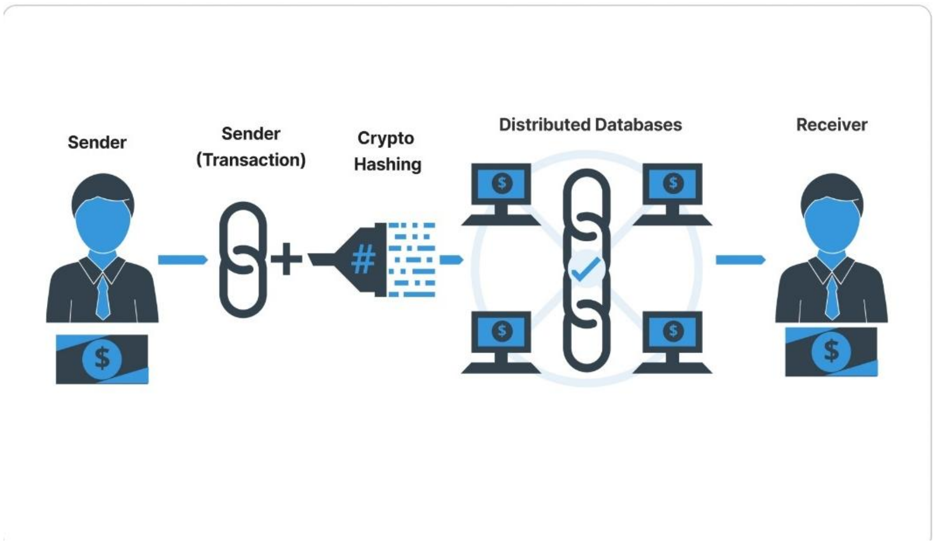


Figure 3.2 Blockchain Transaction Process

Example of a Blockchain Transaction Flow:

Let's consider a real-world example where **Alice** wants to send **0.5 ETH** (Ethereum) to **Bob** using a decentralized payment system.

Transaction Initiation

Alice opens her **MetaMask wallet** and enters Bob's **Ethereum wallet address**.

She specifies **0.5 ETH** as the amount and **signs the transaction** using her private key.

Broadcasting the Transaction

Once Alice confirms the transaction, it is **broadcast** to the Ethereum blockchain network.

Transaction Verification

Miners (if using PoW) or validators (if using PoS) receive Alice’s transaction.

They check if Alice has enough balance and if the transaction follows blockchain rules.

Adding to the Blockchain

After verification, the transaction is included in a **new block**.

This block is cryptographically linked to the previous block, ensuring **immutability**.

Transaction Completion & Ledger Update

Once confirmed, the updated blockchain ledger reflects that **Alice’s balance decreases by 0.5 ETH (plus gas fees)**, and **Bob’s balance increases by 0.5 ETH**.

Bob can now see the received amount in his wallet.

3.3 Smart Contract Execution in Ethereum

Smart contracts are self-executing programs stored on the blockchain that facilitate trustless, automated transactions by enforcing predefined rules. These contracts eliminate the need for intermediaries, reducing costs, improving security, and ensuring transparency.

In the CRYPT project, smart contracts play a crucial role in processing payments securely. Transactions occur only when specific conditions are met, ensuring a decentralized, immutable, and tamper-proof financial system.

Working of Ethereum-Based Smart Contracts

1. **User Interaction:** A user initiates a transaction by interacting with a deployed smart contract using a **web application (DApp)** or a **wallet like MetaMask**.
2. **Condition Verification:** The smart contract evaluates whether all predefined conditions (such as sufficient balance, correct recipient address, or contract execution time) are met.
3. **Transaction Execution:** If the conditions are met, the smart contract executes the function, transferring tokens or triggering other actions.
4. **Blockchain Storage:** The transaction and contract execution details are recorded permanently on the Ethereum blockchain, ensuring transparency and security.
5. **Finalization & Immutability:** Once executed, the transaction cannot be reversed or altered, preventing fraud and ensuring compliance with the original contract terms.

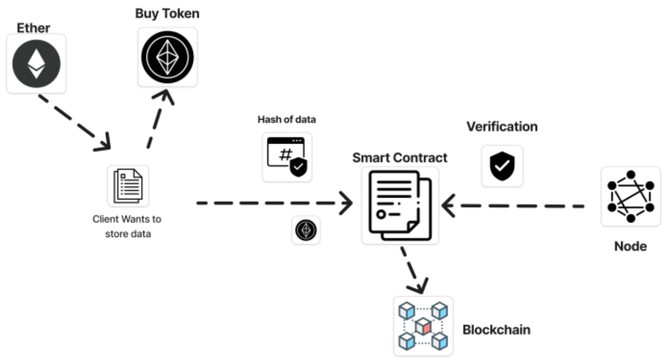


Figure 3.3: Smart Contract Flow

3.4 MetaMask Wallet Integration

For user interaction, decentralized applications (DApps) often integrate with cryptocurrency wallets like MetaMask. MetaMask allows users to manage their private keys securely and interact with blockchain networks seamlessly. In the CRYPT project, users connect their MetaMask wallet to initiate and approve transactions. The interaction process involves:

1. The user connects their MetaMask wallet to the DApp.
2. The DApp requests transaction approval from the user.
3. MetaMask signs and broadcasts the transaction to the blockchain.
4. The transaction status is updated in real-time.

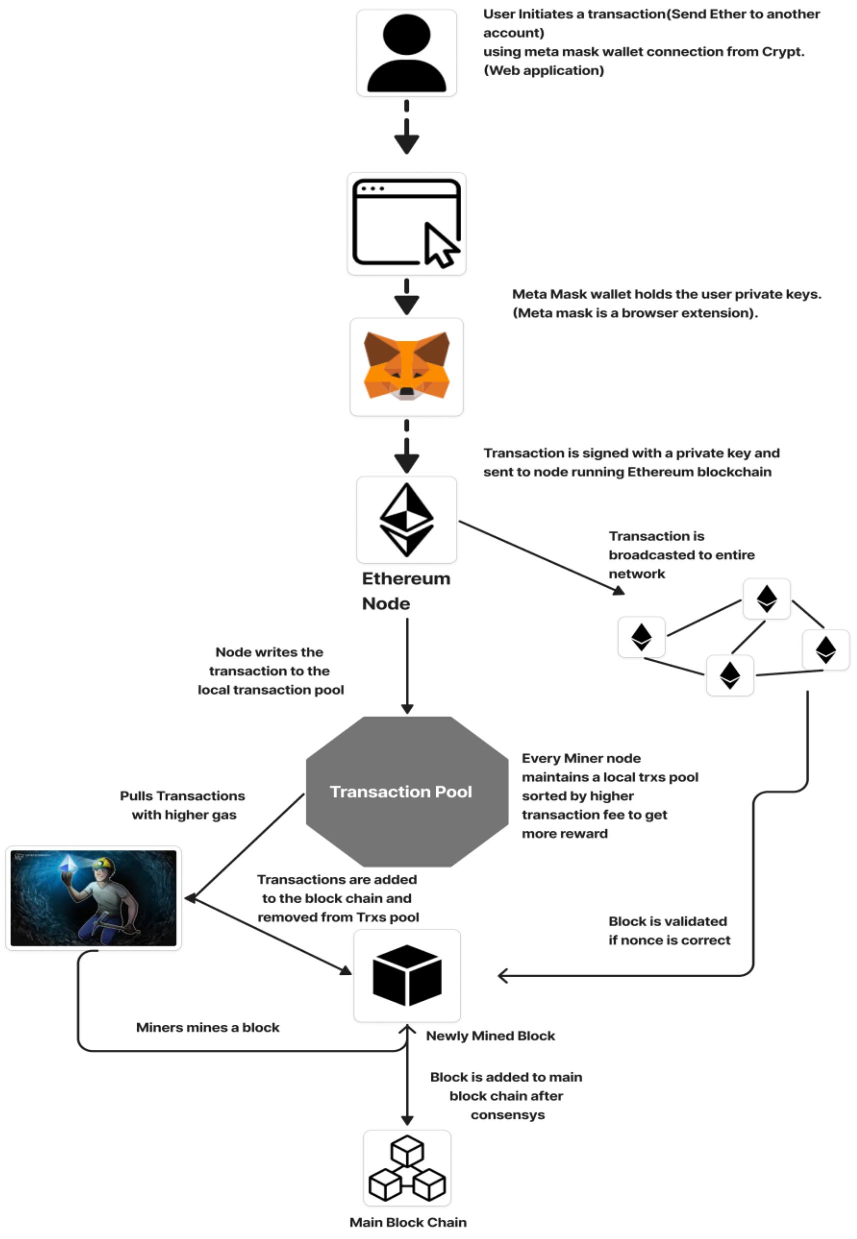

User Initiates a transaction(Send Ether to another account)
using meta mask wallet connection from Crypt. (Web application)
Meta Mask wallet holds the user private keys. (Meta mask is a browser extension).
Transaction is signed with a private key and sent to node running Ethereum blockchain
Ethereum Node
Transaction is broadcasted to entire network
Node writes the transaction to the local transaction pool
Transaction Pool
Every Miner node maintains a local trxs pool sorted by higher transaction fee to get more reward
Pulls Transactions with higher gas
Transactions are added to the block chain and removed from Trxs pool
Block is validated if nonce is correct
Miners mines a block
Newly Mined Block
Block is added to main block chain after consensys
Main Block Chain

Figure 3.4 Meta Mask Wallet and Transaction Flow

3.5 Drawbacks of the Existing System

While decentralized payment networks offer significant advantages, they also have limitations:

- **Transaction Speed & Scalability**: Blockchain networks can become congested, leading to slow transaction processing times.
- **High Gas Fees**: Ethereum-based transactions incur high gas fees, making microtransactions costly.
- **Regulatory Challenges**: Governments and financial institutions are still developing regulations for blockchain-based transactions.
- **Security Risks**: While blockchain is secure, smart contract vulnerabilities can be exploited, leading to potential financial losses.
- **User Experience**: Managing private keys and interacting with blockchain wallets can be complex for non-technical users.

Despite these challenges, decentralized payment systems continue to evolve, with Layer 2 solutions, improved consensus mechanisms, and regulatory advancements working towards addressing these issues.

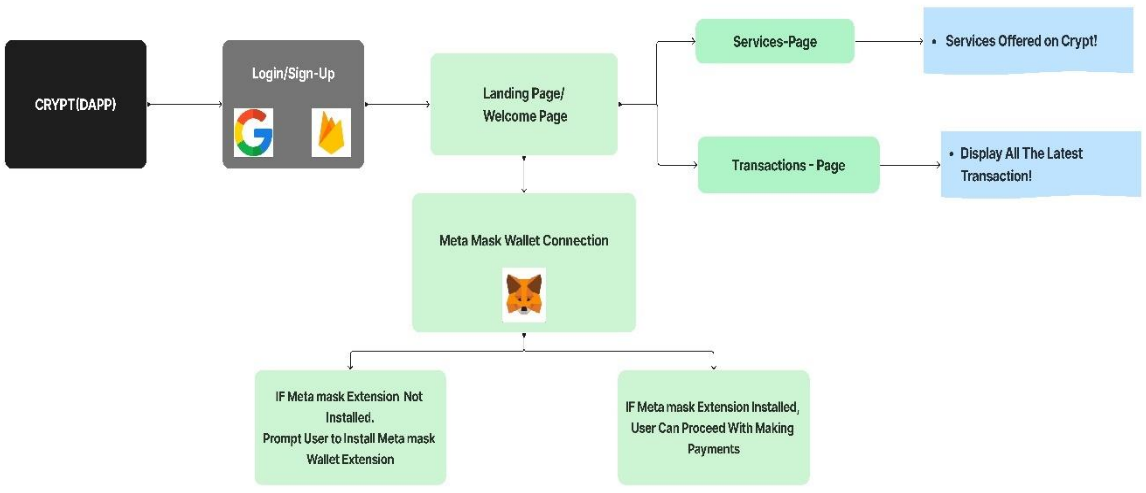
4. PROPOSED SYSTEM

4.1 Overview

The **CRYPT** project introduces a **decentralized payment network** utilizing blockchain technology to provide secure and trustless financial transactions. Traditional payment systems rely on banks and third-party authorities, leading to inefficiencies such as **high transaction fees, slow processing times, and security risks**. The proposed system eliminates intermediaries by using **Ethereum-based smart contracts**, ensuring transactions are **automated, immutable, and transparent**.

This methodology leverages **MetaMask wallet integration** for secure user authentication and **blockchain consensus mechanisms** such as Proof of Stake (PoS) to validate transactions. Users initiate payments directly through smart contracts, reducing fraud risks and enhancing security. Each transaction is **permanently recorded on the distributed ledger**, preventing unauthorized modifications.

By decentralizing payment processing, **CRYPT** provides a borderless, cost-efficient, and scalable solution for digital transactions. The following diagram illustrates the workflow of the proposed methodology.


CRYPT(DAPP)
Login/Sign-Up
Landing Page/ Welcome Page
Services-Page
• Services Offered on Crypt!
Transactions - Page
• Display All The Latest Transaction!
Meta Mask Wallet Connection
IF Meta mask Extension Not Installed. Prompt User to Install Meta mask Wallet Extension
IF Meta mask Extension Installed, User Can Proceed With Making Payments

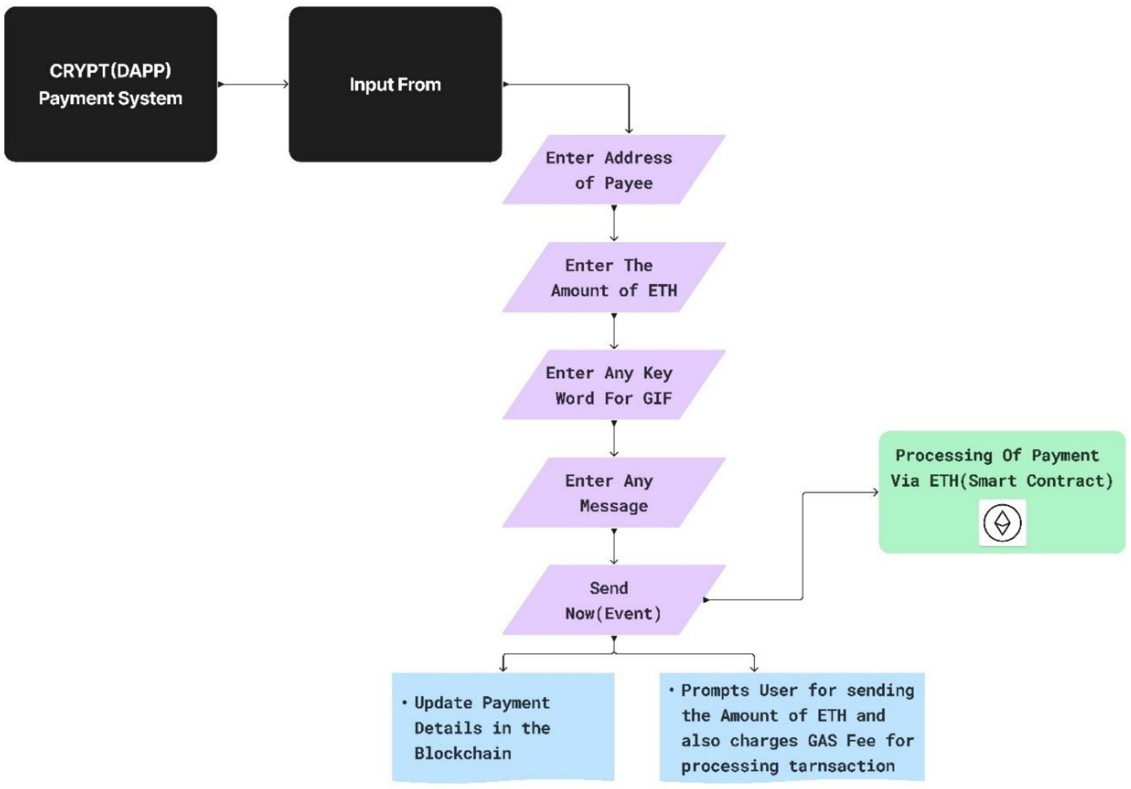

CRYPT(DAPP) Payment System
Input From
Enter Address of Payee
Enter The Amount of ETH
Enter Any Key Word For GIF
Enter Any Message
Send Now(Event)
Processing Of Payment Via ETH(Smart Contract)
• Update Payment Details in the Blockchain
• Prompts User for sending the Amount of ETH and also charges GAS Fee for processing tarnsaction

Fig. 4.1: Block diagram of proposed system.

Figure 4.1 shows the proposed system model. The detailed operation illustrated as follows:

The proposed system model for CRYPT, a decentralized payment network, is designed to facilitate secure and transparent blockchain transactions using Ethereum smart contracts. The system follows a structured workflow that ensures user authentication, wallet connectivity, transaction processing, and blockchain ledger updates.

User Authentication: The first step in the system involves user authentication, which is essential for access control and security. Users must log in or sign-up using Firebase Authentication or Google authentication. This ensures that only authorized users can initiate transactions. By implementing a login system, the platform can also provide a personalized experience, such as displaying transaction history and allowing users to manage their digital assets efficiently.

Once authenticated, users are redirected to the landing page, which serves as the main navigation hub. Here, they can explore the available services, view transaction records, or proceed to make a new transaction.

Connecting to MetaMask Wallet: To initiate transactions, users must connect their MetaMask wallet. This step is crucial as it links their decentralized identity to the blockchain network. If the MetaMask extension is not installed, the system prompts users to install it before proceeding. For users who already have MetaMask, the system verifies the connection and allows them to continue with transactions. Establishing this wallet connection ensures that all transactions are securely processed on the Ethereum blockchain.

Services and Transactions Page: Once logged in and connected to a wallet, users can access the Services and Transactions pages. The Services page provides an overview of the features available on the platform, while the Transactions page displays the latest transactions conducted by the user. This section is designed to offer a seamless and transparent experience, allowing users to track their previous transactions and manage their digital funds efficiently.

Transaction Input and Validation: To make a transaction, users need to input details such as the recipient's Ethereum address, the amount of ETH to be transferred, a keyword for transaction categorization, and an optional message. These inputs are collected through a structured form to ensure accuracy. Before proceeding with payment, the system validates the entered information to prevent errors, such as incorrect wallet addresses or insufficient balance.

Processing Payment via Ethereum Smart Contract: Once the transaction details are validated, the payment process is initiated using an Ethereum smart contract. The smart contract securely processes the transfer, ensuring that the transaction is immutable and transparent. Users are prompted to confirm the transaction in their MetaMask wallet, which includes details such as the transaction amount and the associated gas fee required for processing. This step ensures that users have full control over their transactions and can review the necessary details before proceeding.

Updating Blockchain Records: After the payment is successfully processed, the system updates the transaction details on the blockchain. This ensures that all records are securely stored and cannot be altered, providing a reliable and tamper-proof transaction history. The updated details are then reflected on the user's Transactions page, allowing them to track their payments and verify successful transactions.

Transaction History and Confirmation: The final step in the system involves displaying transaction confirmations and maintaining a historical record. Users can view their latest transactions, including recipient details, transaction amounts, timestamps, and confirmation status. This feature enhances transparency and provides users with a clear overview of their financial activities on the platform. Additionally, any errors or failed transactions are flagged, ensuring users remain informed about the status of their payments.

4.2 Implementation Methodology

The implementation of the proposed **Decentralized Payment Network (CRYPT)** is centered around blockchain technology to enable secure, transparent, and decentralized financial transactions. The system eliminates the need for intermediaries, ensuring direct peer-to-peer (P2P) money transfers on the Ethereum blockchain. This approach enhances security, reduces transaction costs, and increases accessibility for users. The core components of the implementation include **Ethereum blockchain integration, smart contract development, MetaMask wallet authentication, and a React-based frontend for user interaction**.

Working Principle:

The working principle of the CRYPT system revolves around smart contracts and blockchain transaction validation. The system operates in the following structured steps:

Step1: User Authentication

Users sign up or log in using Firebase Authentication or Google Authentication. This ensures that only authorized users can access the platform and initiate transactions.

Step 2: Wallet Connection

Users must connect their MetaMask wallet, which acts as their decentralized identity and allows interaction with the Ethereum blockchain. If the MetaMask extension is not installed, the system prompts users to install it.

Step 3: Transaction Input

The user provides transaction details such as:

- The recipient's Ethereum address
- The amount of ETH to be transferred
- An optional message for reference

Step 4: Transaction Validation

The system performs validation checks to ensure:

- The Ethereum address is valid
- The user has **sufficient balance** in their wallet
- The input fields are correctly filled

Step 5: User Confirmation

Once the validation is successful, the system prompts the user to confirm the transaction via **MetaMask wallet approval**, ensuring they authorize the payment.

Step 6: Smart Contract Execution

The **Ethereum smart contract** securely processes the transaction by:

- Deducting the specified amount from the sender's wallet
- Transferring the amount to the recipient's wallet
- Storing transaction details on the blockchain

Step 7: Blockchain Record Update

The transaction is permanently recorded on the Ethereum blockchain, making it **immutable** and **transparent**. The transaction hash and timestamp are stored for reference.

Step 8: Transaction History Display

Users can view their **latest transactions** on the platform, including:

- Transaction amount
- Recipient details
- Timestamp
- Confirmation status

  This ensures transparency and allows users to track their financial activities efficiently.

4.3 Advantages of the Proposed System

Decentralization – CRYPT operates on the Ethereum blockchain, eliminating the need for banks or intermediaries and increasing transaction efficiency.
Security **and Transparency** – Smart contracts ensure that transactions are **tamper-proof and publicly verifiable**, reducing the risk of fraud.
Cost **Efficiency** – Traditional banking systems involve high fees. CRYPT transactions occur directly on the Ethereum network, requiring only minimal **gas fees**.
User **Control and Ownership** – Users have **full control** over their funds, managing their digital assets directly from their MetaMask wallets without third-party intervention.
Fast **and Borderless Transactions** – Unlike conventional banking, which can take days for international transfers, blockchain transactions are **processed within minutes**, enabling instant cross-border payments.
Immutable **Transaction History** – All transactions are **recorded permanently on the blockchain**, ensuring transparency and preventing data tampering.
Scalability **and Future Expansion** – CRYPT is designed to support future upgrades, including:
Multi-currency support (other cryptocurrencies)
Token-based payments
Integration with DeFi applications

5. SYSTEM DESIGN

5.1 ARCHITECTURE DESIGN:

The CRYPT Web 3.0 application enables secure Ethereum transactions using MetaMask Wallet and Google Authentication. Built with React, Node.js, and Solidity, it allows users to log in, connect their wallet, and send ETH seamlessly. The backend interacts with Ethereum Sepolia Testnet via Alchemy, ensuring decentralized transaction execution. Users can track their transaction history, with confirmations and loading states displayed for a smooth experience.

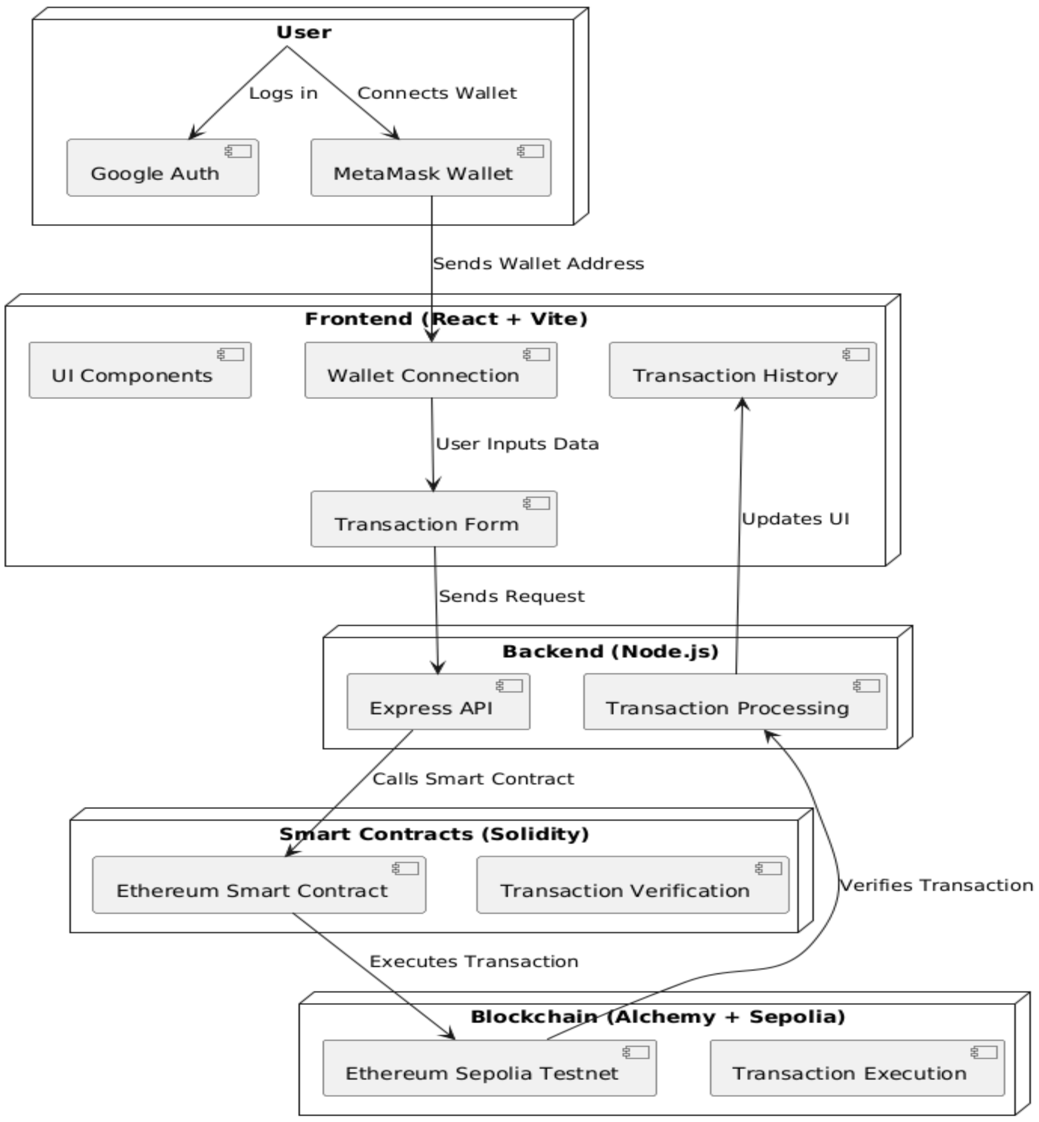


Fig 5.1.1 System Architecture for CRYPT

5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

5.1 Class diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

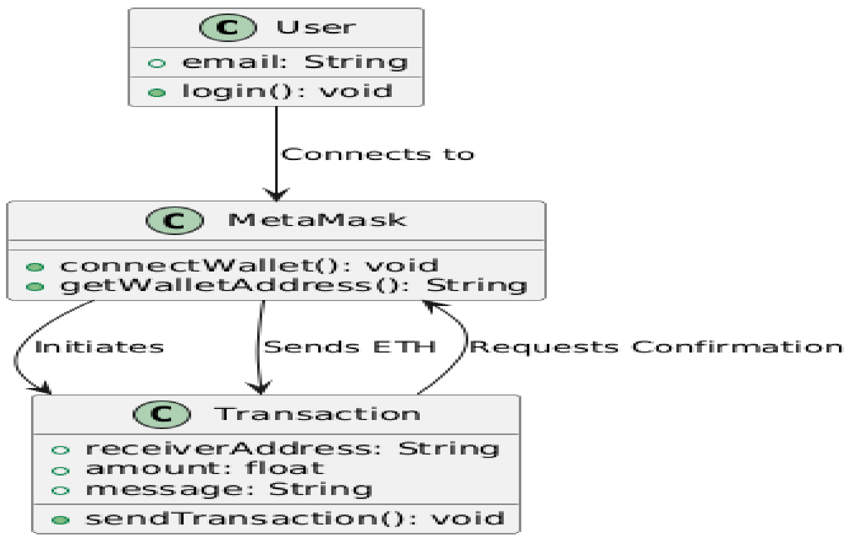


Fig 5.2.1 Class Diagram for crypt

5.2 Use case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

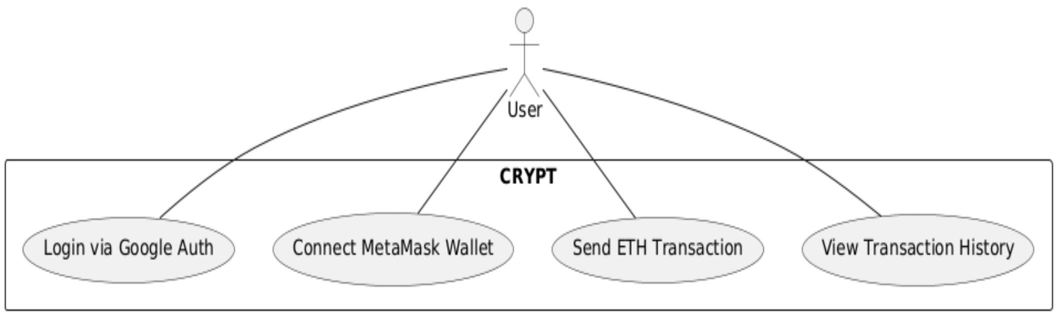


Fig 5.2.2 Use Case Diagram for crypt

5.3 Sequence Diagram

A **sequence diagram** in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

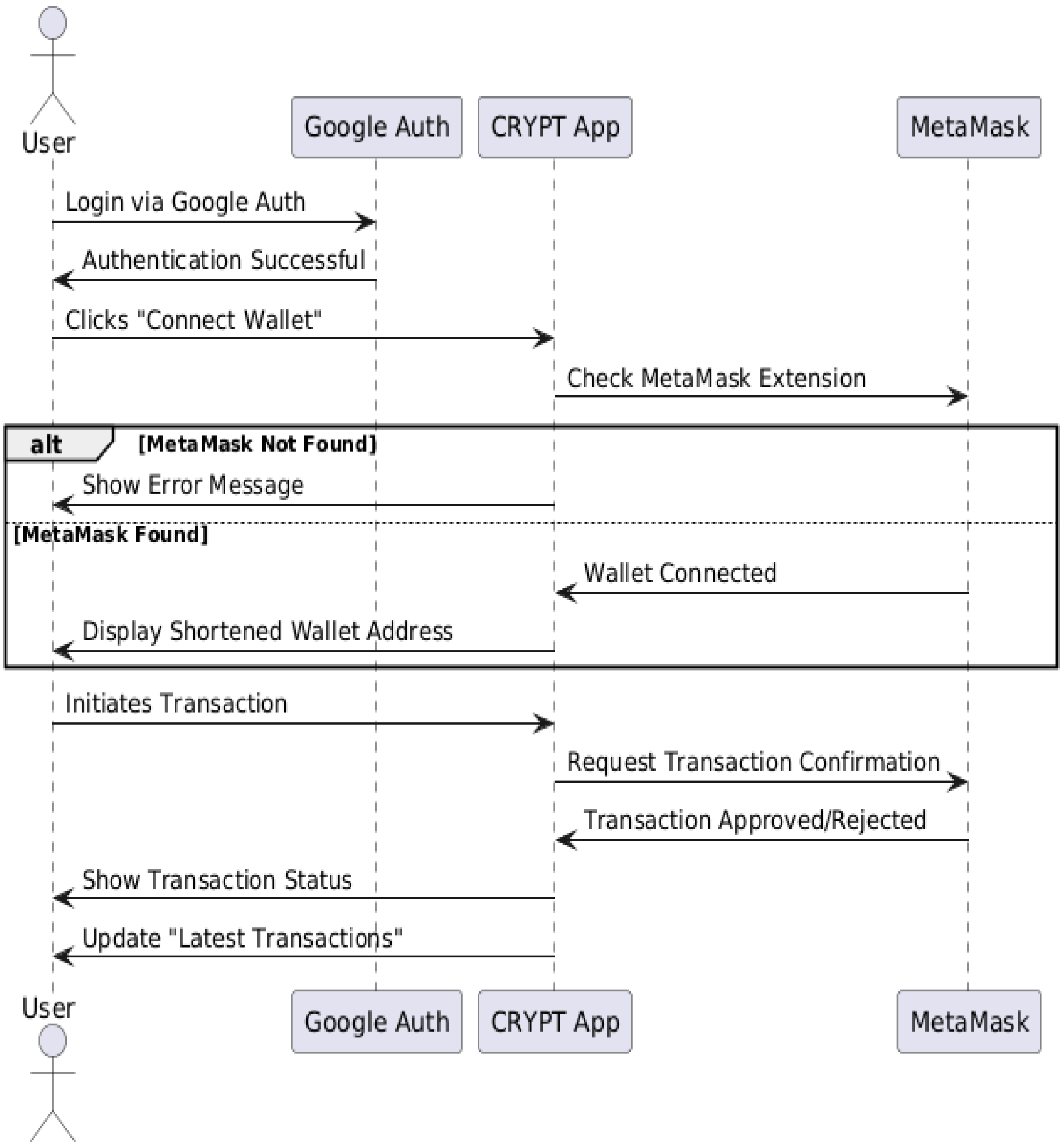


Fig 5.2.3 Sequence Diagram for crypt

5.4 Activity diagram: Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency.

In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

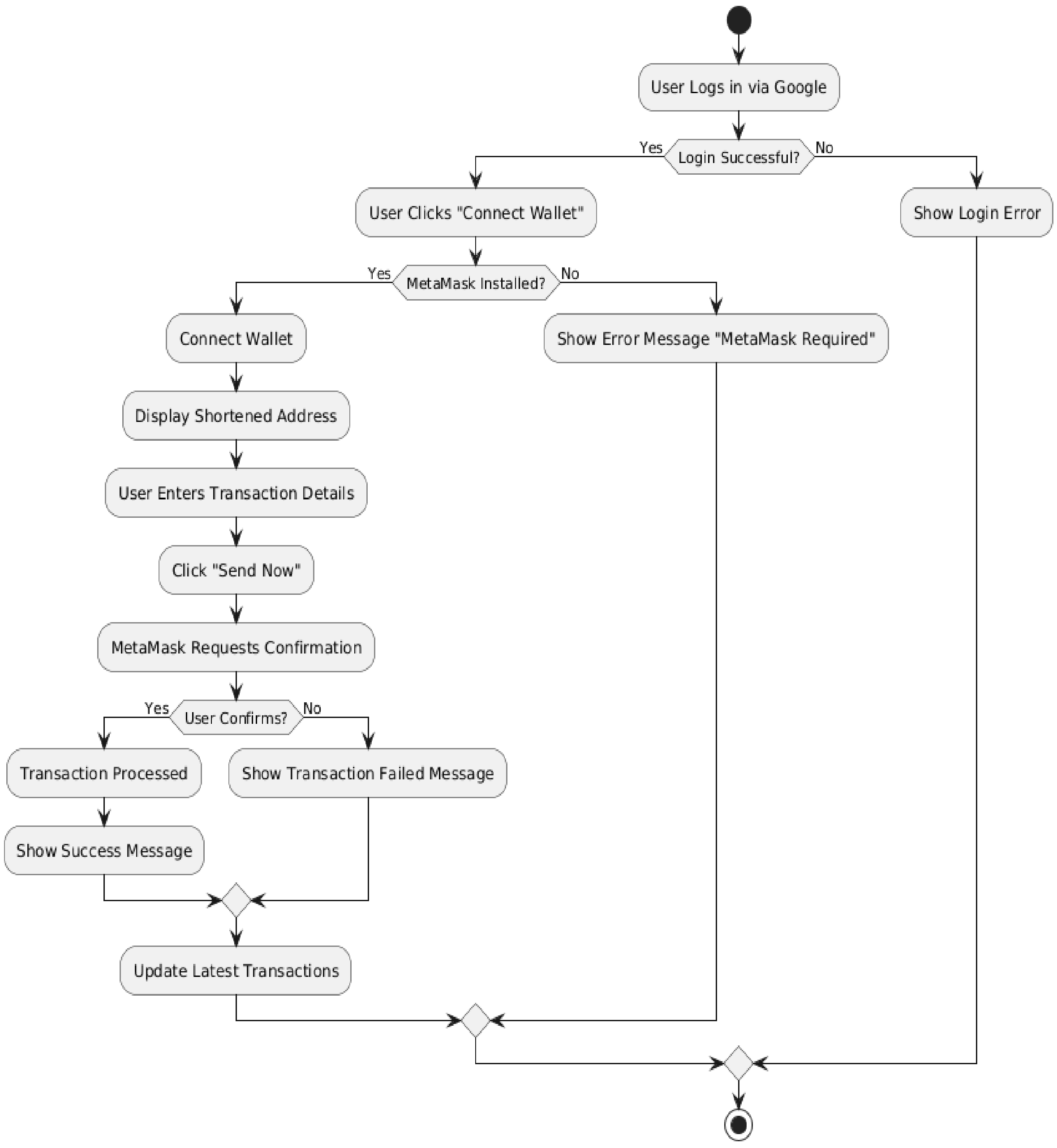

User Logs in via Google
Login Successful?
Yes
No
User Clicks "Connect Wallet"
Show Login Error
MetaMask Installed?
Yes
No
Connect Wallet
Show Error Message "MetaMask Required"
Display Shortened Address
User Enters Transaction Details
Click "Send Now"
MetaMask Requests Confirmation
User Confirms?
Yes
No
Transaction Processed
Show Transaction Failed Message
Show Success Message
Update Latest Transactions

Fig 5.2.4 Activity Diagram for crypt

6. SOFTWARE ENVIRONMENT

6.1 What is Node.js?

Node.js is an open-source, cross-platform, server-side runtime environment that allows JavaScript to run outside a web browser. It is built on Chrome's V8 JavaScript engine and is widely used for building scalable, high-performance network applications.

- Node.js is a lightweight, fast, and efficient runtime that enables asynchronous, event-driven programming.
- It supports both Object-Oriented and Functional Programming paradigms.
- Node.js applications are highly scalable, making it ideal for microservices and real-time applications.
- It is widely used by top tech companies like Netflix, LinkedIn, Walmart, Uber, PayPal, and eBay.
- The biggest strength of Node.js is its vast ecosystem of open-source libraries available via **npm (Node Package Manager)**.

Common Use Cases of Node.js

Node.js provides extensive support for various types of applications, including:

- **Web Servers & APIs** (Express.js, Nest.js)
- **Real-Time Applications** (Chat apps, Live streaming)
- **Microservices Architecture**
- **Blockchain & Cryptocurrency** (Ethereum-based apps)
- **Automation & Scripting**
- **Server-Side Rendering** (Next.js)
- **Database Connectivity** (MongoDB, MySQL, PostgreSQL)

Advantages Of Node.js

Let's see how Python dominates over other languages.

1. Asynchronous and Non-Blocking

Node.js uses an event-driven, non-blocking I/O model, which makes it lightweight and highly efficient. Unlike traditional synchronous operations, it can handle multiple requests simultaneously.

2. Fast Performance

Built on Google's V8 engine, Node.js converts JavaScript into highly optimized machine code, making it one of the fastest runtime environments available.

3. Single Programming Language

With Node.js, developers can write both frontend and backend code using JavaScript, reducing the need to learn multiple languages and making full-stack development easier.

4. Scalability

Node.js applications can handle a large number of concurrent users with minimal resources. Its event-driven architecture makes it an excellent choice for microservices and cloud-based applications.

5. Large Community & Rich Ecosystem

The **npm** (Node Package Manager) has thousands of libraries and modules that accelerate development and reduce coding effort.

6. Real-Time Applications

Node.js is ideal for real-time applications like chat applications, online gaming, and collaboration tools because of its ability to maintain persistent connections using WebSockets.

7. Easy to Learn

Since JavaScript is already a popular language for frontend development, developers can quickly adapt to Node.js for backend programming.

8. Microservices & Cloud-Friendly

Node.js is widely used for developing microservices-based architectures, making it a perfect fit for cloud-native applications.

9. Cross-Platform Development

Node.js allows developers to create cross-platform applications using frameworks like

Electron.js, making it possible to build desktop apps with web technologies.

10. Open-Source and Free

Node.js is completely open-source and free to use, making it cost-effective for individuals and companies of all sizes.

Advantages of Node.js Over Other Languages

1. High Speed & Low Latency

Node.js executes code faster than Python and PHP because of its asynchronous nature. It is ideal for real-time applications where speed is critical.

2. Cost-Effective & Resource-Efficient

Since Node.js allows full-stack JavaScript development, companies can reduce costs by hiring fewer developers compared to using separate backend and frontend technologies.

3. Universal Compatibility

Node.js runs on all major operating systems, including Windows, macOS, and Linux, making deployment easier across different environments.

4. Strong Industry Adoption

Many tech giants and startups use Node.js for its scalability and performance benefits, making it a widely trusted and in-demand technology.

Disadvantages of Node.js

1: Single-Threaded Limitations

Node.js uses a **single-threaded event loop**, making it less suitable for **CPU-intensive tasks** like video processing and large calculations.

2: Callback Hell

Due to its asynchronous nature, Node.js applications often face **callback hell**, where multiple nested callbacks can make the code difficult to manage and debug.

3: **Unstable API**

Node.js frequently releases updates, sometimes changing APIs in a way that requires modifications in the existing codebase.

4: Weak with Heavy Computation

Unlike multi-threaded technologies like Java, Node.js struggles with **CPU-bound tasks**, which can lead to performance issues in data-heavy applications.

5: Security Concerns

The vast number of third-party **NPM modules** increases the risk of **security vulnerabilities**, requiring developers to carefully monitor dependencies.

History of Node.js

Node.js was developed by Ryan Dahl in 2009 as an open-source, cross-platform runtime environment that enables JavaScript to run outside of web browsers. Before Node.js, JavaScript was primarily a client-side scripting language used for adding interactivity to web pages. Dahl introduced Node.js to address the limitations of traditional web servers, such as blocking I/O operations, and to provide an event-driven, non-blocking architecture that allows for high performance and scalability.

Node.js is built on Google's V8 JavaScript engine, which compiles JavaScript code into machine code for faster execution. It introduced the event loop mechanism, allowing applications to handle multiple requests concurrently without waiting for previous ones to complete. This made Node.js ideal for building real-time applications like chat apps, gaming platforms, and streaming services.

In 2010, the introduction of NPM (Node Package Manager) revolutionized package management in JavaScript, making it easy for developers to install and share reusable modules. Companies such as LinkedIn, Netflix, PayPal, and Uber quickly adopted Node.js due to its ability to handle high-traffic applications efficiently.

Over the years, Node.js has undergone significant improvements. In 2015, a fork called io.js emerged, bringing rapid updates, but it later merged back into Node.js. Newer versions of Node.js introduced ES module support, async/await features, and improved security. Today, Node.js remains a preferred choice for building scalable web applications, APIs, and microservices, with an active community and continuous enhancements in performance, security, and developer experience. Its impact on the JavaScript ecosystem has been transformative, making full-stack development more seamless than ever.

Node.js Development Steps

Ryan Dahl introduced Node.js in 2009 as a runtime environment for executing JavaScript outside the browser. It was designed to offer an event-driven, non-blocking I/O model for scalable and efficient web applications. The first release of Node.js included the V8 JavaScript engine, a built-in HTTP module, and an event loop mechanism.

In 2010, the release of Node Package Manager (NPM) revolutionized JavaScript development, allowing developers to share and manage packages easily. This significantly boosted the adoption of Node.js in web development. By 2011, companies like LinkedIn and Netflix started using Node.js to improve their application performance and scalability.

The Node.js Foundation was established in 2015 to ensure the continuous development and maintenance of the platform. That same year, a forked version called io.js emerged, offering faster updates, but it later merged back into Node.js. Node.js 6.0, released in 2016, introduced long-term support (LTS) and enhanced performance improvements, making it more stable for enterprise applications.

The introduction of async/await in Node.js 8 (2017) simplified asynchronous programming, making it easier to write clean and efficient code. In 2019, Node.js 12 introduced ES module support, allowing developers to use import/export syntax natively.

Recent versions of Node.js have focused on security improvements, performance optimization, and better developer experience. Today, Node.js continues to evolve, with an active community and widespread usage in backend development, APIs, microservices, and real-time applications. It remains a dominant technology in modern web development, offering flexibility, speed, and scalability for building server-side applications.

6.2 Modules Used in Project

fs (File System)

The fs module in Node.js is a built-in module that provides an API for interacting with the file system. It allows developers to read, write, update, delete, and manipulate files and directories. The module supports both synchronous and asynchronous operations, making it efficient for handling large-scale file processing tasks. Common methods include fs.readFile() for reading files, fs.writeFile() for writing data to files, and fs.unlink() for deleting files. The fs module is widely used for logging, file uploads, and data persistence, making it an essential component in server-side development.

http/https:

The http and https modules are built-in modules in Node.js that enable communication between the server and clients over the web. The http module is used to create HTTP servers and handle requests and responses, while the https module is used for secure communication using SSL/TLS encryption. Developers use these modules to build RESTful APIs, proxy servers, and web applications. The http.createServer() method is commonly used to set up an HTTP server, while https.createServer() is used for secure data transmission.

Express:

Express.js is a widely used third-party web application framework for Node.js that simplifies the development of server-side applications. It provides a robust set of features for routing, middleware integration, request handling, and response management. Express allows developers to define API endpoints using app.get(), app.post(), and other methods, making it easy to build RESTful services. It also supports middleware functions for logging, authentication, and error handling. Express is lightweight, flexible, and widely adopted in the industry, making it an essential framework for building scalable and maintainable web applications.

jsonwebtoken(JWT):

The jsonwebtoken (JWT) module is used for implementing authentication and secure data exchange between a client and server. It allows developers to create, sign, and verify JSON Web Tokens, which are used for user authentication and authorization. JWTs are widely used in token-based authentication systems, replacing traditional session-based authentication. The jsonwebtoken.sign() method is used to generate tokens, while jsonwebtoken.verify() is used to validate them

Cors:

CORS (Cross-Origin Resource Sharing) is a security feature implemented in web browsers to restrict cross-origin HTTP requests. The cors module in Node.js allows developers to enable or restrict access to APIs from different origins. This is particularly useful when a frontend application (React, Angular, Vue) needs to communicate with a backend hosted on a different domain. The cors module allows configuring rules such as allowed origins, methods, headers, and credentials. By using app.use(cors()), developers can prevent errors like "Blocked by CORS policy" while ensuring security and proper API usage.

6.3 Install Node.js Step-by-Step in Windows and Mac

Node.js, a powerful JavaScript runtime built on Chrome’s V8 engine, allows developers to build scalable and efficient server-side applications. It was initially released in 2009 and has since become a fundamental technology for backend development. Node.js follows an event-driven, non-blocking I/O model, making it ideal for building real-time applications such as chat apps, APIs, and microservices.

Unlike some other programming environments, Node.js does not come pre-installed on Windows or macOS. Developers must install it manually to start using its features. Below is a step-by-step guide to installing Node.js on both Windows and macOS.

How to Install Node.js on Windows and Mac

Node.js has evolved significantly since its initial release in 2009, bringing numerous updates and performance improvements over the years. As a popular runtime environment, it allows developers to run JavaScript outside the browser, making it a key technology for backend development, APIs, and real-time applications. The latest stable version of Node.js is recommended for most users, while developers looking for cutting-edge features can opt for the current version.

If you’re new to Node.js, you may wonder how to install it on your computer. This guide will walk you through the installation process for both Windows and macOS.

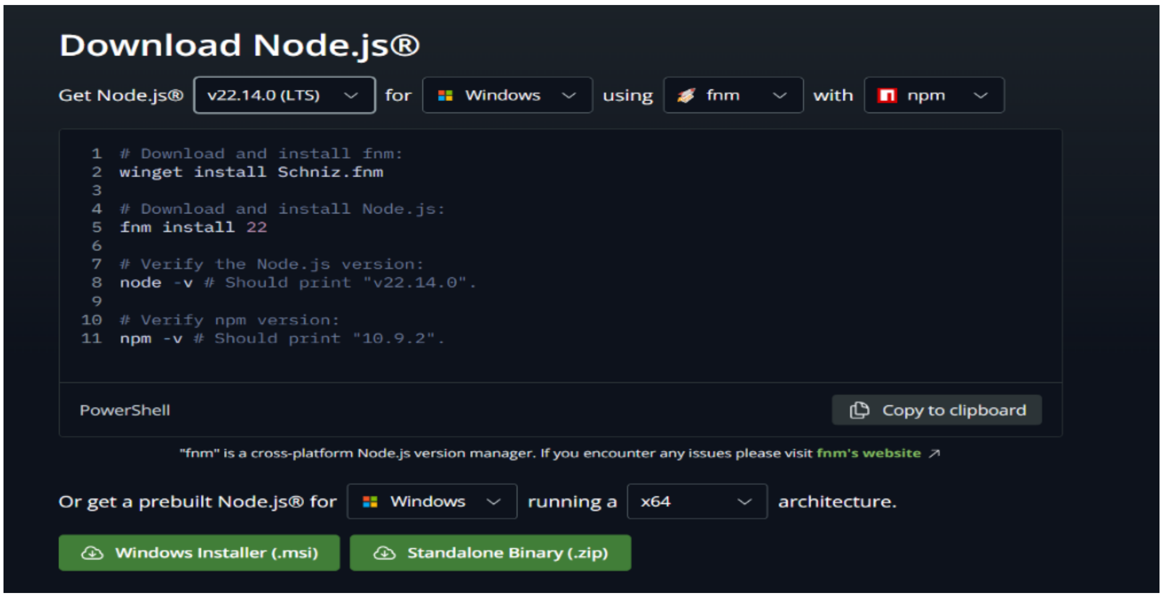
Note: Older versions of Node.js may not support certain features or work efficiently with the latest JavaScript frameworks. It is always recommended to install the latest **Long-Term Support (LTS)** version for stability.

Before installing Node.js, it is important to check your system requirements. The installation process varies depending on the operating system and processor type. You need to download the appropriate Node.js version based on your system architecture.

For example, my system is a Windows 64-bit operating system, so I will install the 64-bit version of Node.js. The steps below will guide you through installing Node.js on a Windows device. These instructions are applicable for Windows 10, 8, and 7.

Download the Correct version into the system

Step 1: Go to the official site to download and install node.js using Google Chrome or any other web browser. OR Click on the following link: https://nodejs.org



Download Node.js®

Get Node.js® v22.14.0 (LTS) for Windows using fnm with npm

```
# Download and install fnm:
winget install Schniz.fnm

# Download and install Node.js:
fnm install 22

# Verify the Node.js version:
node -v # Should print "v22.14.0".

# Verify npm version:
npm -v # Should print "10.9.2".
```

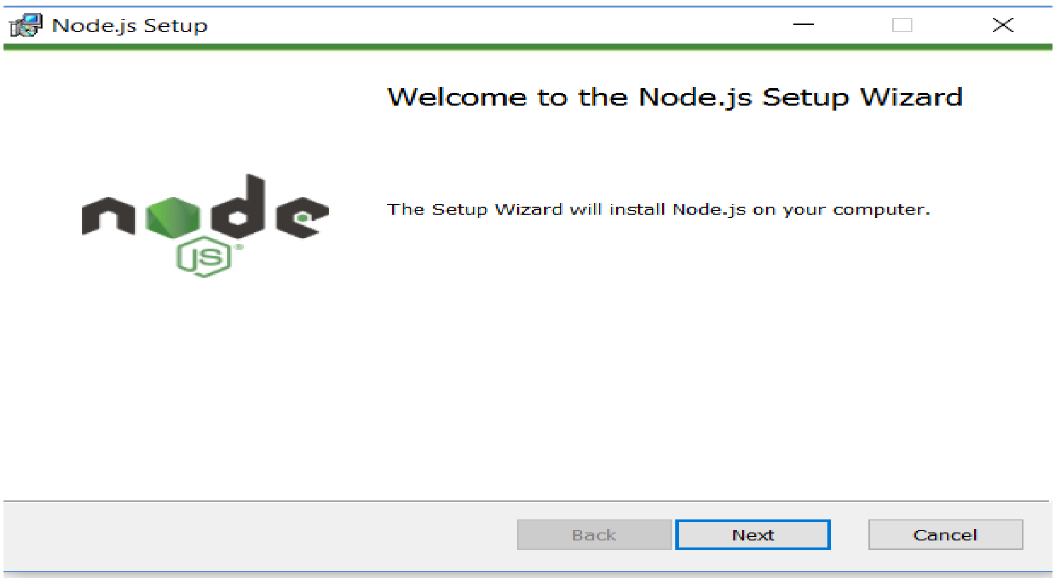
PowerShell

"fnm" is a cross-platform Node.js version manager. If you encounter any issues please visit fnm's website

Or get a prebuilt Node.js® for Windows running a x64 architecture.

Download the **Windows Installer** from **NodeJs official website**. Make sure you have downloaded the latest version of NodeJs. It includes the NPM package manager.
Here, we are choosing the 64-bit version of the Node.js installer.
The LTS (Long-term Support) version is highly recommended for you. After the download of the installer package, install it with a double-click on it.
Now .msi file will be downloaded to your browser. Choose the desired location for that.

Step 2: Install Node.js and NPM



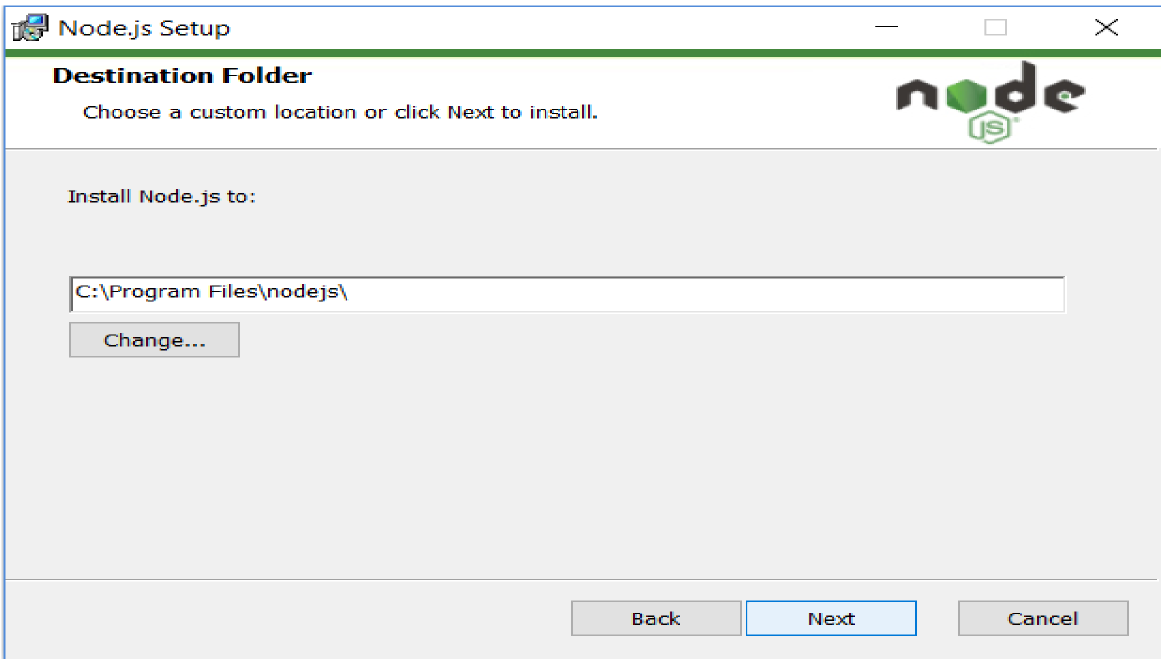
Welcome to the Node.js Setup Wizard

The Setup Wizard will install Node.js on your computer.

After choosing the path, double-click to install .msi binary files to initiate the installation process. Then give access to run the application.

You will get a welcome message on your screen and click the "Next" button. The installation process will start.

- Choose the desired path where you want to install Node.js.



Destination Folder

Choose a custom location or click Next to install.

Install Node.js to:

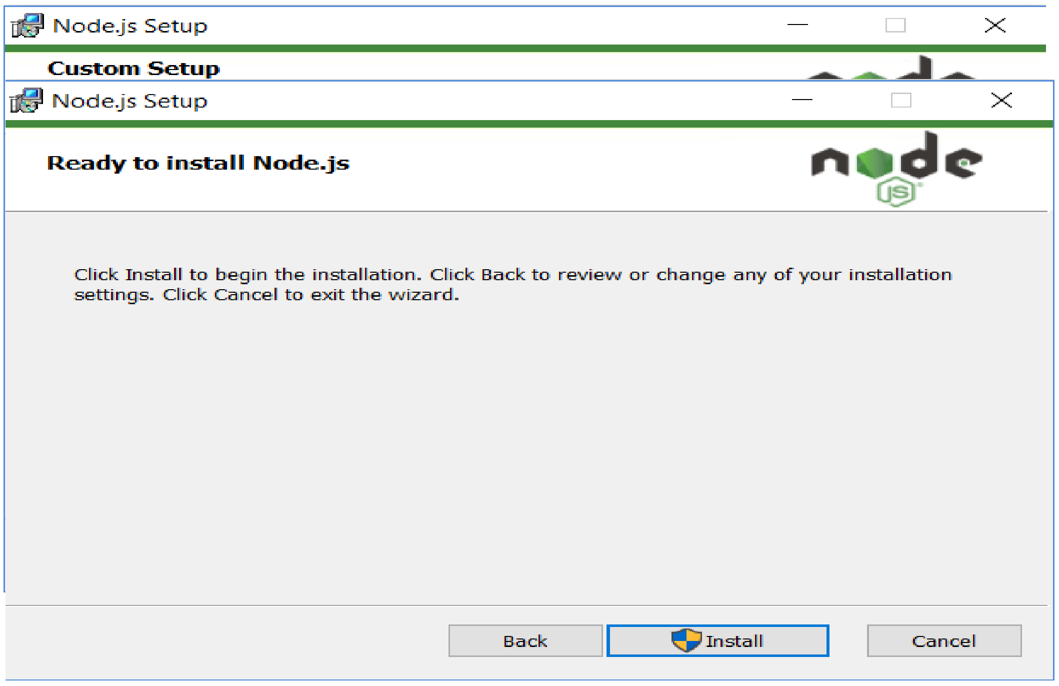
C:\Program Files\nodejs\

- By clicking on the Next button, you will get a custom page setup on the screen. Make sure you choose **npm package manager** , not the default of **Node.js runtime** . This way, we can install Node and NPM simultaneously.

You should have at least 150MB of free space to install Node.js and npm features.

The following features will be installed by default:

- Node.js runtime
- Npm package manager
- Online documentation shortcuts
- Add to Path

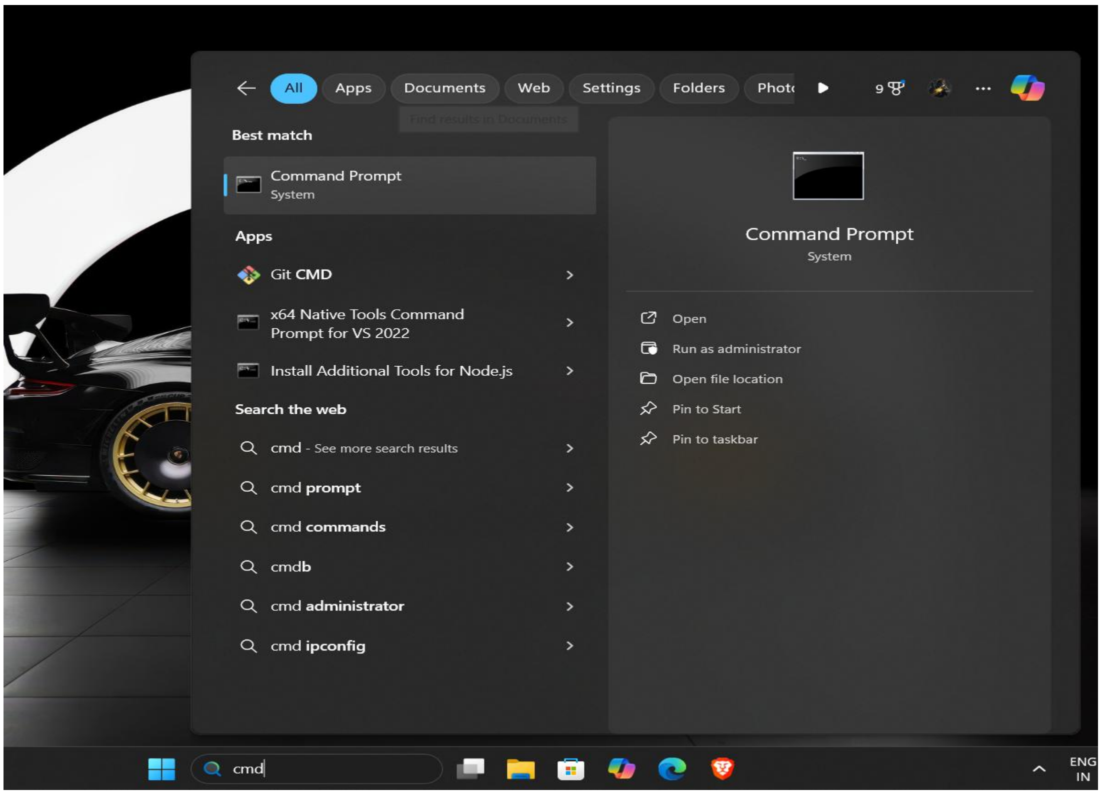


Ready to install Node.js

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Step 3: Check Node.js and NPM Version

- If you have a doubt whether you have installed everything correctly or not, let’s verify it with “Command Prompt”.

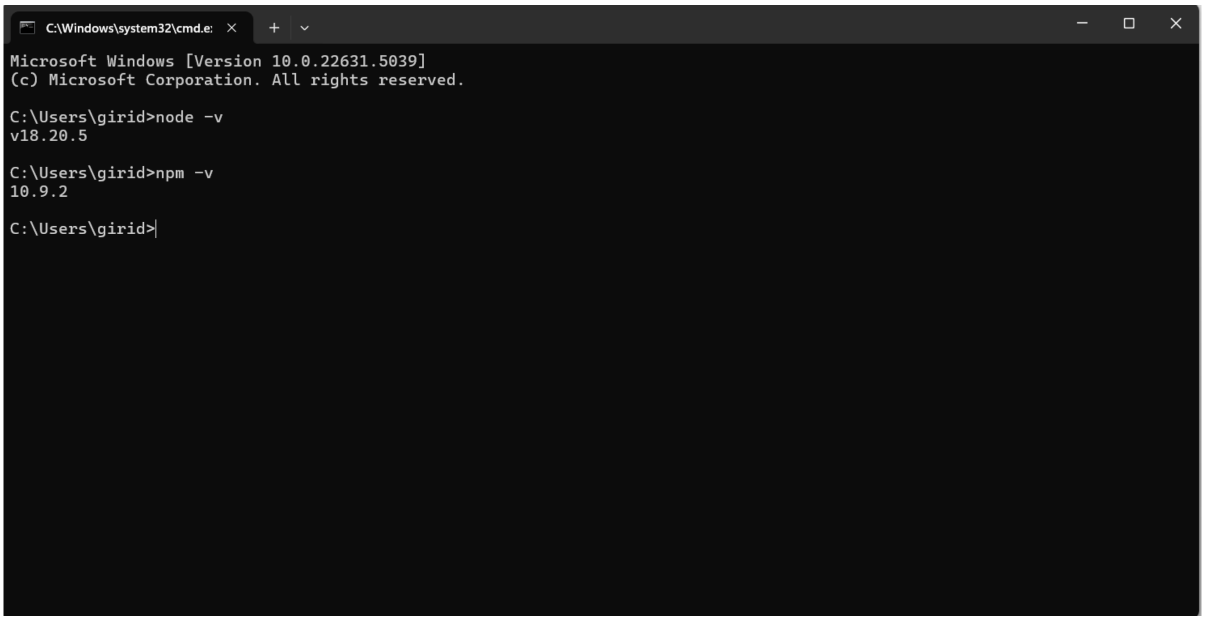


Command Prompt window will appear on the screen.

To confirm Node installation, type *node -v* command.

To confirm NPM installation, type *npm -v* command.

And you don't need to worry if you see different numbers than mine as Node and NPM are updated frequently.



```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Users\girid>node -v
v18.20.5

C:\Users\girid>npm -v
10.9.2

C:\Users\girid>
```

In my case, the version of node.js is v18.20.5 and npm is v10.9.2.

7. REQUIREMENTS SPECIFICATIONS

7.1 Software Requirements

The software requirements include functional specifications such as the **operating system, dependencies, frameworks, and development tools** necessary for building and running the project. These requirements define the overall development environment and constraints needed for efficient execution.

For this project, which involves **Node.js development**, the following software tools and frameworks are required:

- **Node.js (Latest LTS version)** – The core runtime environment for JavaScript-based backend development.
- **npm (Node Package Manager)** – Comes bundled with Node.js and is essential for managing dependencies.
- **React + Vite** – Frontend development framework for building the user interface.
- **Hardhat** – Ethereum development environment for compiling, deploying, and testing smart contracts.
- **Solidity Compiler** – Required for writing and compiling smart contracts.
- **Alchemy or Infura API** – To interact with the Ethereum blockchain network (Sepolia testnet).
- **MetaMask Extension** – A wallet to manage blockchain transactions and interact with smart contracts.

7.2 Hardware Requirements

The hardware requirements depend on the nature of Node.js applications, blockchain integration, and frontend complexity. Applications handling large datasets, smart contract interactions, and multiple API requests will need better performance configurations.

Minimum hardware requirements for this project:

- Operating System: Windows 10/11, macOS, or Linux (Ubuntu recommended for blockchain development).
- Processor: Minimum Intel Core i5 or AMD equivalent for smooth execution.
- RAM: At least 8 GB RAM (16 GB recommended for large-scale deployments).
- Storage: Minimum 500 GB SSD (for faster processing and smart contract interactions).
- Graphics: No special requirements unless using AI/ML-based blockchain analytics.

These specifications ensure smooth execution of the CRYPT Web 3.0 application, including MetaMask wallet connection, smart contract transactions, and frontend interactions. If real-time blockchain data processing is involved, higher configurations might be necessary.

8. FUNCTIONAL REQUIREMENTS

8.1 Output Design

The output design is a crucial aspect of any system as it determines how information is presented to the user. The **CRYPT** Web 3.0 application requires different types of outputs to communicate transaction details and system status. These outputs serve different purposes, such as user notifications, transaction confirmations, and data retrieval. External Outputs, whose destination is outside the organization

Types of Outputs in CRYPT:

- **External Outputs** – Blockchain transaction details that are recorded on the Ethereum blockchain and visible on blockchain explorers (e.g., Etherscan).
- **Internal Outputs** – System logs and transaction history stored within the user’s session.
- **User Interface Outputs** – Transaction status, error messages, confirmations, and balances displayed within the CRYPT application.
- **Operational Outputs** – Backend server logs and smart contract execution details for debugging and monitoring.
- **Interface Outputs** – MetaMask wallet connection prompts, pop-ups for transaction approvals, and loading indicators.

Output Definition

Each output in the system should be defined based on the following factors:

- Type of Output: Real-time transaction details, user notifications, and smart contract responses.
- Content of Output: Transaction hash, wallet address, ETH balance, confirmation status, and error messages.
- Format of Output: JSON data from blockchain responses, formatted UI elements for the user interface.
- Location of Output: Displayed in the web application and stored on the blockchain.
- Frequency of Output: Each time a transaction is executed or the wallet is connected.
- Volume of Output: Limited to individual transactions per user session.

The CRYPT application ensures that all outputs are user-friendly and provide real-time transaction feedback, allowing users to track their payments securely.

Input Design

The input design defines how users interact with the CRYPT application and how data is collected, processed, and validated before being sent to the blockchain. The goal is to provide a seamless and secure input mechanism while minimizing errors.

Objectives of Input Design:

- Ensure a cost-effective method for input collection.
- Achieve the highest level of accuracy in transaction data.
- Validate and verify data before sending it to the blockchain.
- Ensure user-friendly and error-free interaction.

Input Stages in CRYPT:

- Data Entry: Users enter their Ethereum wallet address and transaction amount.
- Data Verification: The system checks if MetaMask is installed and if the address format is valid.
- Transaction Authorization: Users approve transactions through the MetaMask pop-up.
- Data Transmission: The transaction is sent to the Ethereum blockchain via the Alchemy API.
- Validation: The system verifies if the transaction was successfully recorded.

Types of Inputs in CRYPT:

- External Inputs: ETH wallet address, transaction amount, and gas fees entered by the user.
- Internal Inputs: Smart contract responses and blockchain confirmations.
- Operational Inputs: Backend logs for transaction tracking and debugging.
- Interactive Inputs: Users interact with MetaMask for transaction approval.

Error Avoidance & Detection:

- Real-time validation: Ensures users enter valid Ethereum addresses and sufficient funds.

- MetaMask integration: Prevents incorrect wallet connections by enforcing wallet verification.
- Transaction failure handling: Displays error messages if a transaction is rejected or fails.

Data Validation in CRYPT:

- Ethereum wallet addresses must follow the correct format (0x followed by 40 characters).
- Transaction values must be greater than 0 ETH.
- Users must approve the transaction via MetaMask before submission.

These validation techniques ensure that only valid transactions are processed, reducing errors and fraudulent activities.

8.2 User Interface Design

A well-designed user interface (UI) ensures a smooth user experience. The CRYPT Web 3.0 application follows a modern, intuitive, and responsive design to allow seamless interaction with blockchain transactions.

Types of User Interfaces in CRYPT:

User-Initiated Interfaces:

- Command-Driven Interface: The user manually enters transaction details (wallet address and amount).
- Forms-Oriented Interface: The system provides an interactive form where users input their transaction details.

Computer-Initiated Interfaces:

- Menu System: Users navigate through different sections, such as "Connect Wallet," "Send Transaction," and "View History."
- Question-Answer Dialogs: The system asks users to confirm actions (e.g., "Are you sure you want to send this transaction?").

Features of the User Interface:

- MetaMask Wallet Connection – Users connect their wallets seamlessly.
- Transaction Status Updates – Displays "Pending," "Confirmed," or "Failed" messages.

- Error Handling Messages – Provides feedback when transactions fail or wallet connectivity issues occur.
- Loading Indicators – Shows progress when transactions are being processed.
- Dark Mode Support – Enhances UI experience based on user preference.

Error Message Design:

Error messages are designed to be informative and actionable. Examples include:

- "Invalid wallet address." (when an incorrect format is detected)
- "Insufficient balance." (when the user does not have enough ETH)
- "Transaction rejected." (when the user cancels the transaction in MetaMask)

8.3 Performance Requirements

The **performance** of the CRYPT application is measured by **transaction speed, blockchain response time, and UI responsiveness**.

Performance Metrics:

- **Transaction Processing Time:** Should be completed within **5-10 seconds** on the Ethereum testnet.
- **UI Load Time:** The application should load in under **2 seconds**.
- **MetaMask Connection Speed:** The wallet should connect within **3 seconds**.
- **Blockchain Query Response Time:** Fetching transaction history should take no longer than **5 seconds**.

System Efficiency Goals:

- **Interfacing with the Blockchain:** The application should smoothly communicate with the Ethereum blockchain via Alchemy APIs.
- **Accuracy:** Transactions must be executed with **100% accuracy** to prevent loss of funds.
- **User-Friendliness:** The application should be better than traditional payment methods in terms of **speed, security, and convenience**.

Reliability Considerations:

- **Error Handling Mechanisms:** The system should recover from failed transactions by prompting the user with retry options.
- **Scalability:** The system should handle **multiple users simultaneously** without

slowing down.

- **Security Measures:**
  - Transactions must be signed using **MetaMask** to prevent unauthorized access.
  - **SSL Encryption** should be used for secure data transmission.
  - **Smart contract audits** should be performed to prevent exploits.

By meeting these performance requirements, the **CRYPT Web 3.0** application will provide a **secure, efficient, and scalable** platform for blockchain transaction

9. SOURCE CODE

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
      <link rel="icon" type="image/png" sizes="32x32"
href="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAACAAAAAgCAYA
AABzenr0AAAFh0lEQVR4nO2WXYhd1RXH19p7n33OuedrOvXKwAhqQjEoaAk0m
BLboKmtKCkixbRQSvGlrT5q6UtFSh/ahxb61pdqsYoa0YcyA6UmWlRQsSZiNOmDF
DM25mOSuXPPPZ9777WXL1cZ1Ds6SJCWrqdz2Puc9dv/tc5Zf4D/1mBmZGYJAPhFJ
Jeb3V/UxMyMAACnT59OrLW3blgTzCwuVmKx8ZTOubu6rnutqqq3nHOvWGtv+yTI
zxKbbpy+SCKiAwCw1t4EAL9s2zYcj8f/LIriKq31dmYOtNbH27b9UfRZqmr0+fVQBAiM
ib5ZgpGTMrRGREdGVZ7nDOPUlED50/f/6otfbEcDjcy8x10zRHmLlp2zbUWv/VWvv
npmkuR0SHiPxp/TETABHdqVOnLrHW/iEMw+f7vr/Ke99mWbZzba72tbdsjWusrwzBc
XF9f/5v3vkLEvKqqoRDiKSL69erqaoaItBnAx0owlT221v5ECHFv27bub
u7OqqkNRFF0uhPhy3/cAYBOkuR259wRIkIAoPF4/GKapl9JkmQbAPxRSvkXmFEO
8ZHkChGZiH5kjLm7bdtR13XPAkAfBEEEBAHIwGOxi5vPj8fiw1vqmMAx3dV13ShAhx
qXPurHNuZX5+fr/3Pj937twSEf287/ubZ5XjE0sgpUzKsjxkrX1PCDGHiMDQGHiMDMjIijuq6X
ASAfDof3SCnfVUqtA4Aqy/LvQRAshmF4fdM0r0kpy7m5udsQkaWUM/tgVg+QUioR
QmgA8MxMiOiYOc3z/PtJkuwTQojRaPQ7AHg5juOdaZru6bru33VdL2dZ9jWl1I6maV
7w3sezkgMAqFlgRNQAgAMAREQgIgUAx5n5qPf+Xuculfcj7Ms2+O9v65pmv1a67NFUb
zqvV+dTCaHlVJJlmW3IOIcANgtKUBElCTJTkTMvPclAIBSaui9f11K+TYzcxiGDwsh
/qWU2ptl2WNN02tTW2pe6rjud5/lPgyDYOZlMHvXenycyCimSrMUsCGYZgBwMK0xkx
ElRAiBoBESimbpjkQx/GDbdu+E0XR4SiKdgKANMasSCmPO+dO5nl+HyLWRGS2C
qC896tlWT6Z5/m3lVJXJXRFHkmHnU9/3BMAz3x3H8WNM0Po7jg8wMiDgCgE5KWS
AiGWNOAsArSqlAax3OApjVhBYAtqdp+j+j1r7fGyLB8PgmBBSrlXCKGVUje2bftDIrq
GiJ723ncAsAsARsRryaiNQBAKWXsnKustW8j8jYrBVAOG9r8uyXJJSLg4Gg31VVS2VZXl
ESvmMtfb3b3g8HgESGEEkLcboxphRAHELFARD0dSAQASggRAoCfBTDzKxBCmDz
Pb7HWnrDWHkuS5S5AZENHVdL0spr+37/lVmdsaY/1hr9H/9H3/Vqe599lZo+IXgihihKhj5g4
AtqYAEdXMTKPR6AlmTtM0vbWu62u62N1Xb8VRdF3lFItACw659659rJZPJEEAQ78jz/lvc+
mHoC75w7MxgMrgvDcJ+19uwsgFmzQBDRAWb+mWCPCC2EKJi5
```

```
m0wmzwVBsD2Komv7vl+21o6zLPvB2trawfn5+TuIqNdaj5j5/iAInpk26sdmwaZ+YGVl
JV5YWLjPe38zM4OU8soLFy4sBUFQpGm62xhzrOu6l5Mk+ab3/ktKqSEzT7TWIRH9J
gzDPwEATH/ln02BDUrID0ZpXdeXaa3vd859HQAK7/2F9fX1paIotgkhrqmq6g1ElHme
71ZKPXzmzJnfLi4uNh+oudlI3pIj6vv+q1LKB4wxl0kpL6mq6o2+7+uiKHZEUXTUGP
OrOI7f+egBPndMLfiHX4y1dr+19vm6rt80xiwbY/Zs2Ku24gm3CvKhOZ1CXb9hbUtm9
POCyA3XeNHs+KdAfDGJ/x//c/E+UtZMFgQphQQAAAAASUVORK5CYII=" />
      <link rel="icon" type="image/png" sizes="64x64"
href="/assets/favicon_64x64.6eff8989.png" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Crypt</title>
  <script type="module" crossorigin src="/assets/index.868fe0d9.js"></script>
  <link rel="modulepreload" href="/assets/react.b73aa22c.js">
  <link rel="modulepreload" href="/assets/icons.fb5dddf4.js">
  <link rel="stylesheet" href="/assets/index.fca65e40.css">
 </head>
 <body>
  <div id="root"></div>

 </body>
</html>
```

Index.css

```
/* General Styles */
body {
  font-family: sans-serif;
  line-height: 1.5;
  background-color: #f3f4f6;
  color: #1f2937;
  padding: 1rem;
}

/* Container */
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 1rem;
}

/* Flexbox Utilities */
.flex {
  display: flex;
}

.flex-col {
  flex-direction: column;
}

.items-center {
  align-items: center;
}

.justify-center {
  justify-content: center;
}

/* Grid */
.grid {
  display: grid;
}

.grid-cols-2 {
  grid-template-columns: repeat(2, 1fr);
}

.grid-cols-3 {
  grid-template-columns: repeat(3, 1fr);
}

/* Spacing */
.p-4 {
  padding: 1rem;
}

.m-4 {
  margin: 1rem;
}

/* Typography */
.text-xl {
  font-size: 1.25rem;
}

.text-2xl {
  font-size: 1.5rem;
}

.text-gray-700 {
  color: #4b5563;
}

.text-white {
  color: #ffffff;
```

```
    }

    /* Background Colors */
    .bg-blue-500 {
      background-color: #3b82f6;
    }

    .bg-gray-200 {
      background-color: #e5e7eb;
    }

    /* Buttons */
    .button {
      display: inline-block;
      padding: 0.75rem 1.5rem;
      border-radius: 0.375rem;
      text-align: center;
      cursor: pointer;
    }

    .button-primary {
      background-color: #3b82f6;
      color: white;
    }

    .button-secondary {
      background-color: #6b7280;
      color: white;
    }

    /* Rounded Corners */
    .rounded {
      border-radius: 0.375rem;
    }

    .rounded-lg {
      border-radius: 0.5rem;
    }

    /* Shadows */
    .shadow {
      box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
    }

    .shadow-lg {
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

10. RESULTS AND DISCUSSION

10.1 Implementation description

This implementation of machine learning models for predicting CO2 emissions, accompanied This section describes the implementation of the CRYPT Web 3.0 blockchain payment network. The system utilizes blockchain technology for decentralized transactions, ensuring security, transparency, and efficiency. Below is a detailed explanation of each component:

- Importing Libraries:
  - **ethers.js** – For interacting with Ethereum blockchain and smart contracts.
  - **web3.js** – To enable communication with Ethereum nodes.
  - **React & Tailwind CSS** – For frontend development and UI/UX enhancement.
  - **Solidity** – To develop the smart contract for secure transactions.
  - **Hardhat** – For testing, debugging, and deploying smart contracts.
  - **Alchemy API** – To connect to the Ethereum Sepolia network. Node.js and Vite: The project is developed using React with Vite for a faster frontend build process.

Setting Up the Environment:

Node.js and Vite: The project is developed using React with Vite for a faster frontend build process.

Solidity Compiler: Smart contracts are compiled using the Solidity programming language.

MetaMask Integration: Users connect their wallets using the MetaMask extension.

Smart Contract Development:

- **Contract Deployment:** The smart contract is deployed on the Sepolia test network.
- **Contract Functions:** The contract includes functions to send transactions, retrieve balances, and maintain transaction logs.
- **Security Measures:** Implements reentrancy guards and access control mechanisms to prevent unauthorized access.

Frontend Implementation:

- **Wallet Connection:** Users can connect their MetaMask wallet to initiate transactions.
- **Transaction Submission:** A user inputs the recipient's wallet address and the amount to be sent, which is then processed on the blockchain.
- **Loading Indicators:** A UI element displays transaction status updates in real time.
- **Transaction History:** Users can view their latest transactions, including timestamps and status confirmations.

Testing and Debugging:

- **Unit Testing:** Smart contract functions are tested using Hardhat.
- **Frontend Testing:** UI responsiveness and wallet integration are tested in multiple browsers.
- **Blockchain Transaction Monitoring:** Transactions are monitored on Etherscan to ensure accuracy.

10.2 Data Format in Crypt

Unlike traditional machine learning projects, CRYPT operates on blockchain transactions instead of structured datasets. However, the system maintains transaction records and logs, which are structured as follows:

Blockchain Transaction Data:

Each transaction record contains:

- **Transaction Hash:** A unique identifier for the blockchain transaction.
- **Sender Address:** The Ethereum address of the transaction initiator.
- **Receiver Address:** The recipient's Ethereum wallet address.
- **Transaction Amount:** The amount of cryptocurrency transferred.
- **Gas Fees:** The fee paid for processing the transaction.
- **Timestamp:** The exact time when the transaction was executed.
- **Status:** Indicates whether the transaction was **successful, pending, or failed**.

Data Flow in CRYPT:

1. **User Input:** Sender enters recipient address and transaction amount.
2. **Validation:** The system checks if MetaMask is connected and verifies input values.
3. **Transaction Execution:** The transaction is signed and sent to the Ethereum blockchain.
4. **Confirmation:** The system updates the status and displays confirmation details.

Summary of Transactions:

- The system logs every successful transaction on the Ethereum blockchain.
- Failed transactions are recorded with error messages for debugging.
- Users can retrieve their past transactions using the transaction history feature.

Results and description

- **Figure 1** demonstrates the process of **logging in via Google authentication and connecting to the MetaMask wallet**. This step ensures that users are securely authenticated before initiating transactions within the **CRYPT decentralized payment network**.

- **Figure 2** illustrates the **step-by-step transaction flow**, showing how a user **enters the recipient's wallet address, specifies the amount, and submits the transaction**. The figure also depicts how the system **validates the transaction, interacts with the blockchain, and provides a confirmation message** upon completion.

- **Figure 3** presents **summary statistics of transactions**, including details such as the **total number of successful and failed transactions, average transaction amount, gas fees, and processing time**. These insights help evaluate the **efficiency and reliability of the CRYPT payment system**.

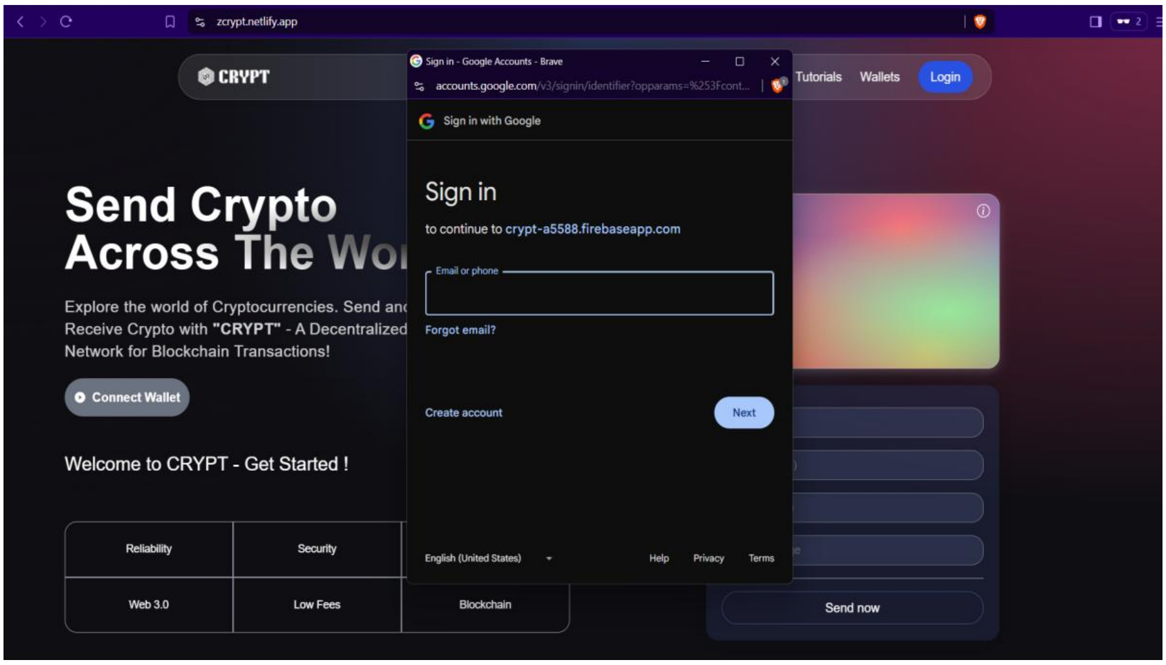

zcrypt.netlify.app
CRYPT
Tutorials
Wallets
Login
Sign in - Google Accounts - Brave
accounts.google.com/v3/signin/identifier?opparams=%253Fcont...
Sign in with Google
Sign in
to continue to crypt-a5588.firebaseapp.com
Email or phone
Forgot email?
Create account
Next
English (United States)
Help
Privacy
Terms
Send Crypto
Across The Wo
Explore the world of Cryptocurrencies. Send and
Receive Crypto with "CRYPT" - A Decentralized
Network for Blockchain Transactions!
Connect Wallet
Welcome to CRYPT - Get Started !
Reliability
Security
Web 3.0
Low Fees
Blockchain
Send now

Fig 10.1: Demonstrates user authentication via Google login.

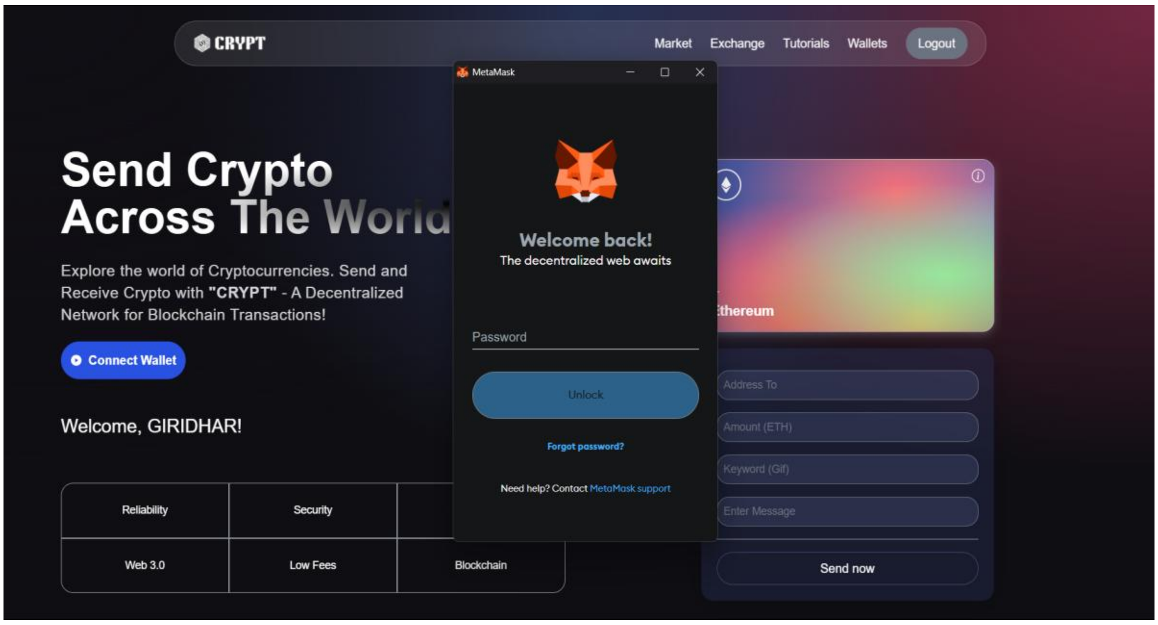
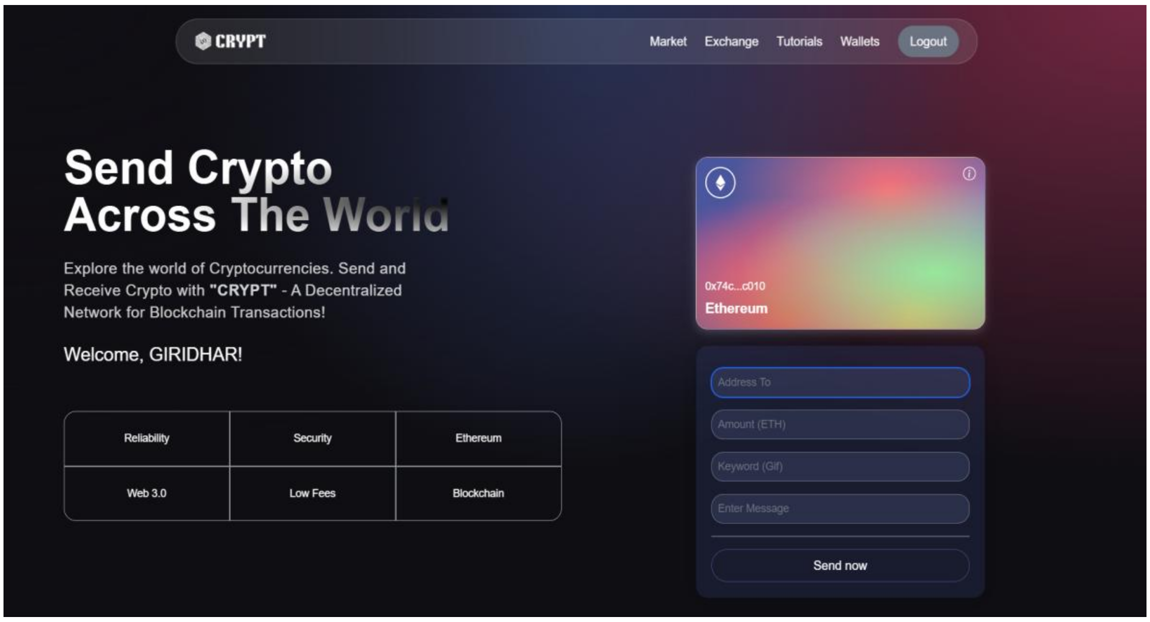


Fig 10.1.1: Demonstrates user authentication MetaMask wallet for blockchain transactions.



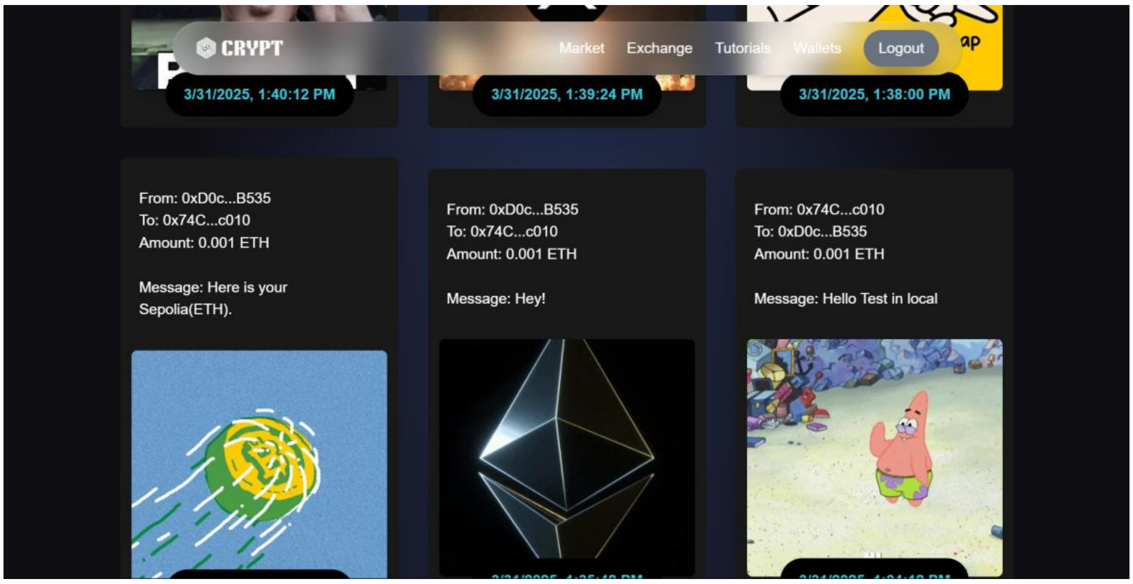
Send Crypto
Across The World

Explore the world of Cryptocurrencies. Send and Receive Crypto with "CRYPT" - A Decentralized Network for Blockchain Transactions!

Welcome, GIRIDHAR!

Reliability	Security	Ethereum
Web 3.0	Low Fees	Blockchain

Figure 10.2: Showcases the transaction initiation process, where users enter recipient details and transaction amounts before submitting on the blockchain.



3/31/2025, 1:40:12 PM

3/31/2025, 1:39:24 PM

3/31/2025, 1:38:00 PM

From: 0xD0c...B535
To: 0x74C...c010
Amount: 0.001 ETH

Message: Here is your Sepolia(ETH).

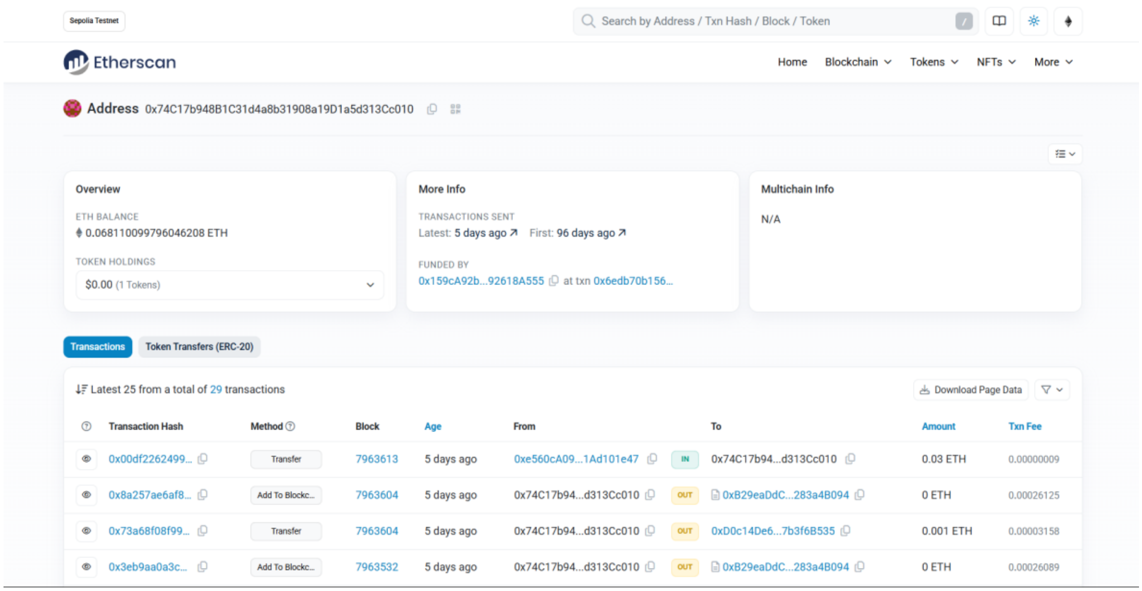
From: 0xD0c...B535
To: 0x74C...c010
Amount: 0.001 ETH

Message: Hey!

From: 0x74C...c010
To: 0xD0c...B535
Amount: 0.001 ETH

Message: Hello Test in local

Fig 10.3: Displays the Latest transaction Data.


Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs More

Address 0x74C17b948B1C31d4a8b31908a19D1a5d313Cc010

Overview	More Info	Multichain Info
ETH BALANCE 0.068110099796046208 ETH	TRANSACTIONS SENT Latest: 5 days ago ↗ First: 96 days ago ↗	N/A
TOKEN HOLDINGS $0.00 (1 Tokens)	FUNDED BY 0x159cA92b...92618A555 at txn 0x6edb70b156...	

Transactions | Token Transfers (ERC-20)

Latest 25 from a total of 29 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From		To	Amount	Txn Fee
0x00df2262499...	Transfer	7963613	5 days ago	0xe560cA09...1Ad101e47	IN	0x74C17b94...d313Cc010	0.03 ETH	0.00000009
0x8a257ae6af8...	Add To Blockc...	7963604	5 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00026125
0x73a68f08f99...	Transfer	7963604	5 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003158
0x3eb9aa0a3c...	Add To Blockc...	7963532	5 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00026089

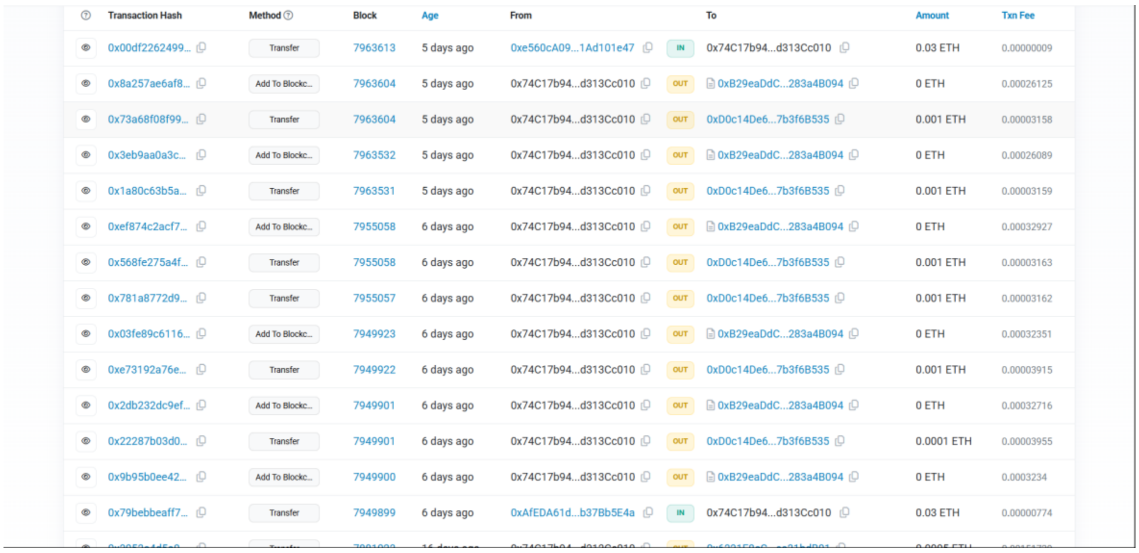

Transaction Hash	Method	Block	Age	From		To	Amount	Txn Fee
0x00df2262499...	Transfer	7963613	5 days ago	0xe560cA09...1Ad101e47	IN	0x74C17b94...d313Cc010	0.03 ETH	0.00000009
0x8a257ae6af8...	Add To Blockc...	7963604	5 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00026125
0x73a68f08f99...	Transfer	7963604	5 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003158
0x3eb9aa0a3c...	Add To Blockc...	7963532	5 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00026089
0x1a80c63b5a...	Transfer	7963531	5 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003159
0xef874c2acf7...	Add To Blockc...	7955058	6 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00032927
0x568fe275a4f...	Transfer	7955058	6 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003163
0x781a8772d9...	Transfer	7955057	6 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003162
0x03fe89c6116...	Add To Blockc...	7949923	6 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00032351
0xe73192a76e...	Transfer	7949922	6 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.001 ETH	0.00003915
0x2db232dc9ef...	Add To Blockc...	7949901	6 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.00032716
0x22287b03d0...	Transfer	7949901	6 days ago	0x74C17b94...d313Cc010	OUT	0xD0c14De6...7b3f6B535	0.0001 ETH	0.00003955
0x9b95b0ee42...	Add To Blockc...	7949900	6 days ago	0x74C17b94...d313Cc010	OUT	0xB29eaDdC...283a4B094	0 ETH	0.0003234
0x79bebbeaff7...	Transfer	7949899	6 days ago	0xAfEDA61d...b37Bb5E4a	IN	0x74C17b94...d313Cc010	0.03 ETH	0.00000774

Figure 10.3.1: Displays the transaction confirmation screen, showing the transaction hash, status, and other relevant details after a successful blockchain transaction.

11. CONCLUSION AND FUTURE SCOPE

CONCLUSION

In conclusion, the development of the **CRYPT** decentralized payment network has demonstrated the potential of blockchain technology in facilitating secure, transparent, and efficient financial transactions. By leveraging smart contracts on the Ethereum blockchain, this system eliminates the need for intermediaries, ensuring trustless and tamper-proof transactions. The seamless integration with MetaMask allows users to interact with the blockchain effortlessly, while features such as transaction history tracking and real-time confirmations enhance the overall user experience. This project highlights the transformative power of Web 3.0 in financial applications and sets the foundation for further advancements in decentralized payment systems.

Future Scope

Looking ahead, the scope for expanding **CRYPT** is extensive. Future enhancements could include multi-chain support, enabling transactions across different blockchain networks for improved interoperability. Implementing layer-2 scaling solutions, such as rollups or sidechains, could further reduce transaction costs and improve scalability. Additionally, integrating advanced security features, such as fraud detection mechanisms and biometric authentication, would enhance the system’s robustness. Another promising avenue is the incorporation of decentralized identity verification (DID) to strengthen user authentication without compromising privacy. Furthermore, collaboration with fintech platforms and decentralized finance (DeFi) ecosystems could position **CRYPT** as a key player in the next-generation digital economy. Ultimately, the ongoing evolution of blockchain technology will continue to provide opportunities for innovation, making decentralized financial systems more accessible and efficient for users worldwide.

References

[1] Arikumar K. S., Deepak Kumar A., Gowtham C., Sahaya Beni Prathiba. "Decentralized Loan Management Application Using Smart Contracts on Blockchain." St. Joseph's Institute of Technology, Chennai, India, and MIT Campus, Anna University, Chennai, India. Available at: https://ebooks.iospress.nl/pdf/doi/10.3233/APC210020

[2] Sridhar Sathyanarayan. "Blockchain Solutions for Mortgage Loan Origination." Managing Consultant, Product Banking Practice, Wipro Ltd. Available at: https://www.wipro.com/content/dam/nexus/en/industries/banking/latest-thinking/blockchain-solutions-for-mortgage-loan-origination.pdf

[3] I. Karamitsos, M. Papadaki, N. B. Al Barghuthi. "Blockchain Smart Contracts: Applications, Challenges, and Future Trends." International Journal of Computer Applications, 2019.

[4] Megha Ravindranath. "Legality of Blockchain and Smart Contracts in India." Journal of Intellectual Property Rights, vol. 23, pp. 213-221, 2018.

[5] Peng Li. "Secure Balance Planning of Off-blockchain Payment Channel Networks." The University of Aizu, Japan, 2020.

[6] Hamed Amini, Maxim Bichuch, Zachary Feinstein. "Decentralized Payment Clearing using Blockchain and Optimal Bidding." arXiv preprint arXiv:2109.00446, 2021. Available at: https://arxiv.org/abs/2109.00446

[7] Huapeng Li, Baocheng Wang. "PRETRUST: A Framework for Fast Payments in Blockchain Systems." arXiv preprint arXiv:2204.13827, 2022. Available at: https://arxiv.org/abs/2204.13827

[8] Shlok Dubey. "Blockchain-based Payment Systems: A Bibliometric & Network Analysis." arXiv preprint arXiv:2212.03790, 2022. Available at: https://arxiv.org/abs/2212.03790

[9] Yuan Liu, Shuai Sun, Zhengpeng Ai, Shuangfeng Zhang, Zelei Liu, Han Yu. "FedCoin: A

Peer-to-Peer Payment System for Federated Learning." arXiv preprint arXiv:2002.11711, 2020. Available at: https://arxiv.org/abs/2002.11711

[10] Zach Pandl, Isabel Rosenberg. "Opportunities and Risks in Decentralized Finance." 2021.

[11] Ziqiao Ao, Lin William Cong, Gergely Horvath, Luyao Zhang. "Is Decentralized Finance Actually Decentralized? A Social Network Analysis of the Aave Protocol on the Ethereum Blockchain." arXiv preprint arXiv:2202.00906, 2022. Available at: https://arxiv.org/abs/2202.00906

[12] Bank for International Settlements. "Project mBridge: Connecting Economies through CBDC." 2022. Available at: https://www.bis.org/publ/othp59.pdf

[13] Lewis Gudgeon, Sam Werner, Daniel Perez, William J. Knottenbelt. "DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency." Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, 2020.

[14] Alfred Lehar, Christine A. Parlour. "Systemic Fragility in Decentralized Markets." SSRN Electronic Journal, 2022.

[15] Lioba Heimbach, Eric Schertenleib, Roger Wattenhofer. "An Empirical Study of Liquidity and Volatility in Decentralized Financial Markets." Lecture Notes in Computer Science, Springer Nature Switzerland, 2023

\