Name: Sushant Tulasi

Class: D20B

Roll: 60

MULTI-SIGNATURE WALLET USING BLOCKCHAIN

ABSTRACT

This project aims to build a secure and decentralized **Multi-Signature Wallet (MultiSig Wallet)** on the Ethereum blockchain. A MultiSig Wallet increases the security of digital fund management by requiring multiple approvals before executing any transaction.
It eliminates the risk of a single point of failure or misuse of funds by ensuring that no single owner can move funds without the consent of others. The smart contract is written in **Solidity**, deployed using **Hardhat**, and connected to a user-friendly **React.js** frontend integrated with **MetaMask**.The entire decentralized application (DApp) is deployed on **AWS** for scalability and real-world accessibility.

INTRODUCTION

In traditional financial systems, a single person often has complete control over transactions. This poses security risks, especially in organizations or partnerships.
A **Multi-Signature Wallet** provides a decentralized way to manage shared funds where multiple signatures (approvals) are required for a transaction to be executed.
This project demonstrates how blockchain technology can be used to achieve collective financial decision-making using smart contracts.

Objectives

- To design and implement a secure MultiSig Wallet on Ethereum.
- To ensure multiple owners approve transactions before execution.
- To create a simple, responsive web interface using React.js.

- To deploy the system on the cloud using AWS infrastructure.

SYSTEM ARCHITECTURE

The MultiSig Wallet system consists of three major components:

1. **Smart Contract Layer (Ethereum + Solidity)**
    - Manages wallet ownership, transactions, and approvals.
    - Ensures immutable and transparent transaction rules.
2. **Application Layer (React + Ethers.js)**
    - Provides the user interface for interacting with the smart contract.
    - Allows users to connect via MetaMask, submit, and approve transactions.
3. **Deployment Layer (AWS Hosting)**
    - Hosts the frontend using AWS EC2 or S3 + CloudFront.
    - Ensures public accessibility and scalability of the DApp.

SYSTEM DESIGN

Architecture Diagram

User → MetaMask → Smart Contract (on Sepolia Testnet) → Blockchain Storage → React Frontend (on AWS)

Modules

1. **Owner Management Module** – Registers and verifies wallet owners.
2. **Transaction Module** – Handles submission, approval, and execution of transactions.
3. **Approval Logic Module** – Ensures transactions execute only after required approvals.

4. **Frontend Interface** – Connects to MetaMask and displays wallet details and transaction history.

IMPLEMENTATION DETAILS

1. Smart Contract

- Written in **Solidity (v0.8.x)**.
- The contract accepts a list of owner addresses and a required approval count during deployment.
- Functions include:
    - `submit(address to, uint value, bytes data)` – to propose a transaction.
    - `approve(uint txId)` – to approve a pending transaction.
    - `revoke(uint txId)` – to withdraw approval.
    - `execute(uint txId)` – to finalize and execute once approvals meet the threshold.

2. Backend Deployment (Hardhat)

- Used **Hardhat** framework for compiling, deploying, and testing.
- The contract was deployed on the **Sepolia Test Network** using Infura or Alchemy RPC endpoints.
- Environment variables like private key and RPC URL were stored securely in a `.env` file.

Example deployment command:

npx hardhat run scripts/deploy.js --network sepolia

3. Frontend Implementation

- Built using **React.js**.
- Integrated **Ethers.js** to interact with the deployed smart contract.
- Used MetaMask for wallet connection and signing.
- The UI displays:
  - Connected wallet address
  - ETH balance
  - Submitted transactions
  - Buttons for “Submit”, “Approve”, and “Execute” actions

4. AWS Deployment

The React application is deployed on **AWS** to make it accessible publicly.
Deployment Steps:

Build React app using

```
npm run build
```

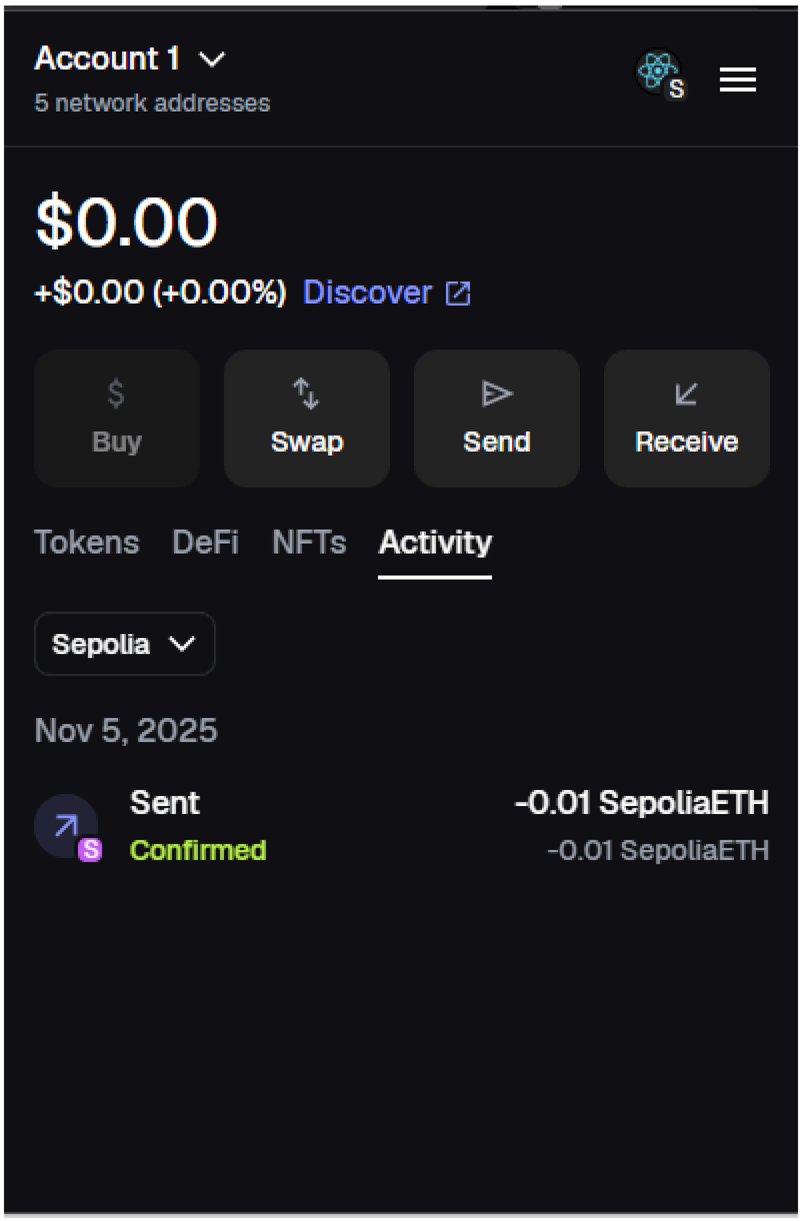
1. Deploy build folder to AWS S3 or host using EC2.
2. Use **AWS CloudFront** for HTTPS support and caching.
3. Link custom domain if needed.

RESULTS

- The MultiSig Wallet was successfully deployed on the **Sepolia testnet**.
- The contract correctly handled multiple owners and required approvals before transaction execution.

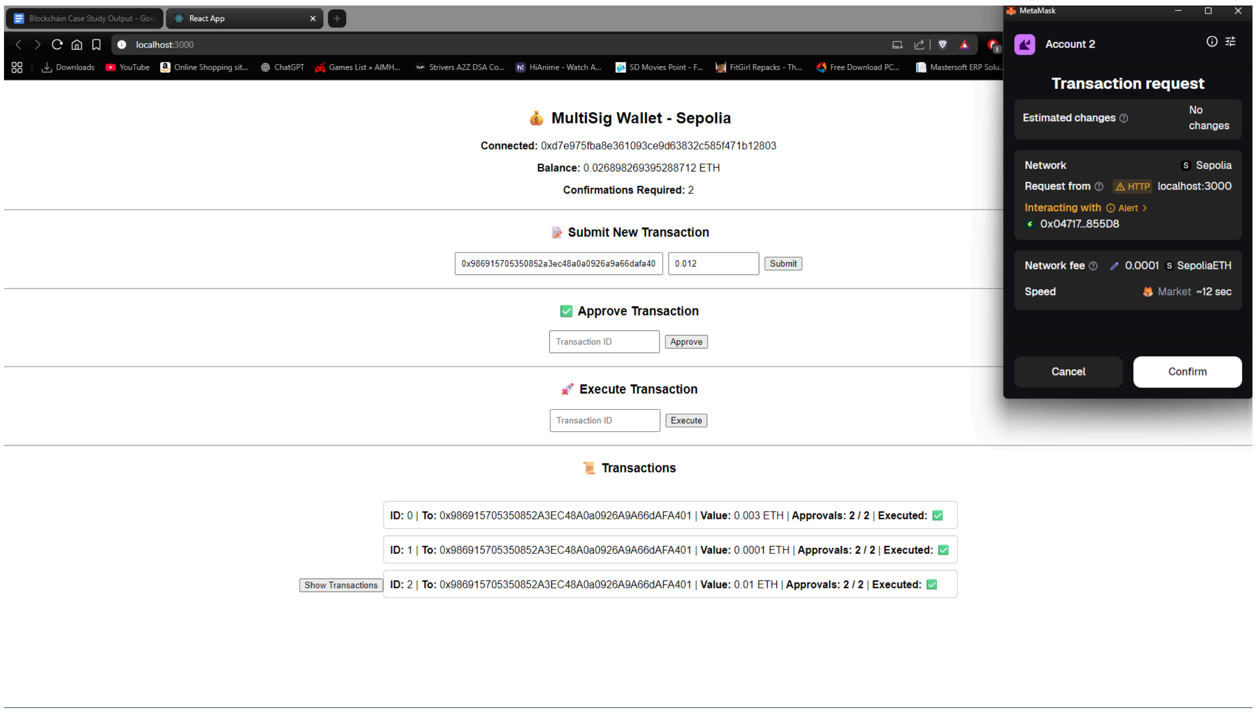
- The React interface connected smoothly with MetaMask.
- Transactions could be submitted, approved, revoked, and executed transparently.
- The DApp hosted on AWS performed reliably and was accessible globally.

SCREENSHOTS



Nov 5, 2025

Sent -0.01 SepoliaETH
Confirmed -0.01 SepoliaETH



💰 MultiSig Wallet - Sepolia

Connected: 0xd7e975fba8e361093ce9d63832c585f471b12803

Balance: 0.026898269395288712 ETH

Confirmations Required: 2

Submit New Transaction

0x986915705350852a3ec48a0a0926a9a66dafa40 | 0.012 | Submit

✅ Approve Transaction

Transaction ID | Approve

Execute Transaction

Transaction ID | Execute

Transactions

Show Transactions

ID: 0 | **To:** 0x986915705350852A3EC48A0a0926A9A66dAFA401 | **Value:** 0.003 ETH | **Approvals: 2 / 2** | **Executed:** ✅

ID: 1 | **To:** 0x986915705350852A3EC48A0a0926A9A66dAFA401 | **Value:** 0.0001 ETH | **Approvals: 2 / 2** | **Executed:** ✅

ID: 2 | **To:** 0x986915705350852A3EC48A0a0926A9A66dAFA401 | **Value:** 0.01 ETH | **Approvals: 2 / 2** | **Executed:** ✅

MetaMask

Account 2

Transaction request

Estimated changes: No changes

Network: Sepolia

Request from: ⚠ HTTP localhost:3000

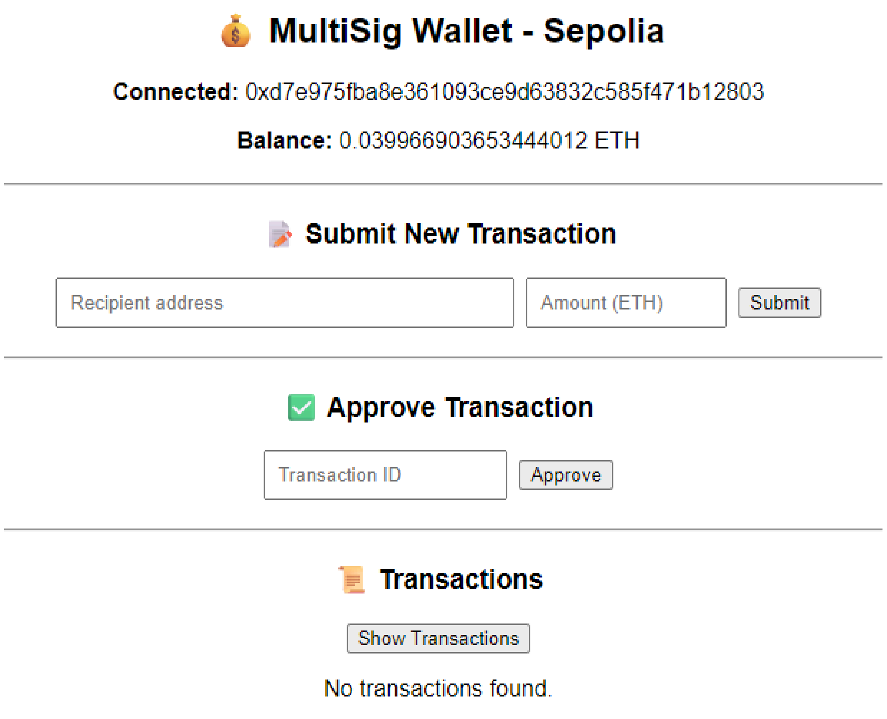
Interacting with ⓘ Alert >

0x04717...855D8

Network fee: 0.0001 SepoliaETH

Speed: Market ~12 sec

Cancel | Confirm



💰 MultiSig Wallet - Sepolia

Connected: 0xd7e975fba8e361093ce9d63832c585f471b12803

Balance: 0.039966903653444012 ETH

Submit New Transaction

Recipient address | Amount (ETH) | Submit

✅ Approve Transaction

Transaction ID | Approve

Transactions

Show Transactions

No transactions found.

ADVANTAGES

- Enhanced security and transparency
- No central authority control
- Easy integration with MetaMask
- Fully decentralized and verifiable logic

LIMITATIONS

- Dependent on gas fees on the Ethereum network
- Requires all owners to be active for approvals
- Limited UI simplicity for large-scale use

FUTURE ENHANCEMENTS

- Add WalletConnect for mobile wallet support.
- Implement transaction history filtering and pagination.
- Deploy on **Layer 2 chains** like Polygon to reduce fees.
- Integrate email/notification alerts for pending approvals.
- Implement multi-network support with backend analytics.

CONCLUSION

The MultiSig Wallet project demonstrates how blockchain technology can be used to create secure, shared, and decentralized financial systems.
By combining **Solidity**, **Hardhat**, **React**, and **AWS**, the project achieves complete decentralization from contract logic to cloud deployment.
This system provides a real-world example of how organizations or groups can manage shared funds safely without depending on third parties.

REFERENCES

1. Ethereum Documentation – https://ethereum.org
2. Hardhat Framework – https://hardhat.org
3. Ethers.js Library – https://docs.ethers.org
4. AWS Documentation – https://aws.amazon.com/documentation