SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
An Autonomous Institution
MANGALURU

Project Report on
Cryptography & Network Security (21CS61)

Submitted in partial fulfillment of the requirements for VI Semester of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

Submitted By

Mahaan N Bhat	**4SF21CD015**
Zeeshaan S Shaikh	**4SF21Cd032**

Under the Guidance of

Mrs. Shwetha Shetty

Associate Professor

Department of Information Science & Engineering at

SAHYADRI

College of Engineering & Management
An Autonomous Institution
Mangaluru – 575 007

2023-2024

Blockchain Authentication App Project

Introduction

The Blockchain Authentication App project aims to create a secure, decentralized, and transparent authentication system that ensures data integrity, empowers users, and automates access controls using smart contracts. This report outlines the objectives, implementation process, security aspects, use cases, and future aspects of the project.

Objective

The objective of the Blockchain Authentication App project is to create a secure, decentralized, and transparent authentication system that ensures data integrity, empowers users, and automates access controls using smart contracts.

Implementation Process

1. Set Up Development Environment

- Install Node.js and npm: Ensure the latest versions are installed.
- Install Truffle: Use npm install -g truffle for the development framework.
- Install Ganache: Download Ganache for local blockchain development.

2. Create a Truffle Project

- Initialize Project: Run truffle init to set up the project.
- Configure Truffle: Update truffle-config.js to connect to Ganache.

3. Develop Smart Contracts

- Write Contracts: Create smart contracts for user registration and authentication in Solidity.

- Compile and Migrate: Compile with truffle compile and deploy to Ganache with truffle migrate.

4. Set Up React Frontend

- Create React App: Use npx create-react-app to generate a new React project.
- Install Web3.js: Use npm install web3 to interact with the blockchain.

5. Integrate Smart Contracts with React

- Import Contracts: Add compiled contract files to the React project.
- Initialize Web3: Connect Web3 to the local blockchain (Ganache).
- Develop Registration and Login Forms: Create forms to handle user registration and authentication.

6. Develop Frontend Components

- Registration Form: Capture user details and register them on the blockchain.
- Login Form: Verify user credentials with the blockchain.
- Dashboard: Display user-specific information upon successful login.

7. Test the Application

- Unit Testing: Test smart contracts using Truffle's testing framework.
- Integration Testing: Ensure seamless interaction between the frontend and smart contracts.
- User Testing: Validate the user experience and authentication flow.

8. Deploy to a Live Network

- Test Network Deployment: Deploy to a test network (e.g., Ropsten) for further testing.
- Mainnet Deployment: Deploy to the Ethereum mainnet once testing is complete.
- Frontend Deployment: Host the React app on a web server or service like Vercel or Netlify.

9. Monitor and Maintain

- Network Monitoring: Use tools like Etherscan to track transactions.
- Application Maintenance: Regularly update dependencies and address security vulnerabilities.

Security Aspects

- Data Integrity: Immutability and cryptographic hashing protect user data.
- Access Control: Private keys and smart contracts ensure secure access.
- Decentralization: Distributed ledger reduces risk of data loss and tampering.
- Encryption: Ensures secure data transmission and storage.
- Authentication Mechanisms: Multi-factor authentication and smart contract audits.
- Transparency and Traceability: Public ledger and traceable transactions.
- Resilience to Attacks: DDoS protection and Sybil resistance.
- User Control and Privacy: Self-sovereign identity and selective data sharing.
- Regular Security Updates: Smart contract and dependency updates.

Use Cases

- Secure Access Management: Authorized user access to systems and data.
- Decentralized Identity Verification: No central authority needed.
- Immutable Audit Trails: Tamper-proof records of authentication.
- Privacy-Preserving Login: Secure authentication without exposing personal info.
- Multi-Party Collaboration: Secure interactions in a decentralized

manner.
- Smart Contracts: Automate permissions and access controls.

Future Aspects

- Interoperability: Seamless authentication across platforms.
- Advanced Cryptography: Enhanced privacy and security.
- Scalability: Efficient handling of more authentication requests.
- Decentralized Identity Standards: Widespread identity management.
- IoT Integration: Secure IoT device authentication.
- User Experience: More user-friendly interfaces.
- AI and ML: Detecting and responding to threats.
- Regulatory Compliance: Adherence to data protection laws.
- Financial Services: Enhanced transaction security.
- DAOs: Secure membership and voting management.

Some Snaps of the project…

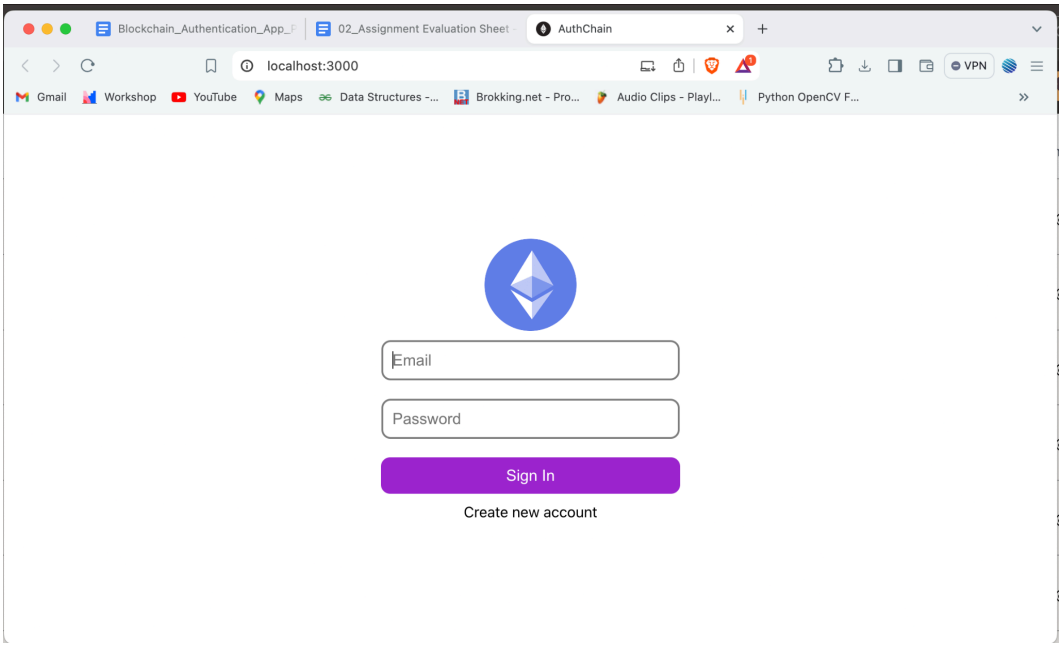
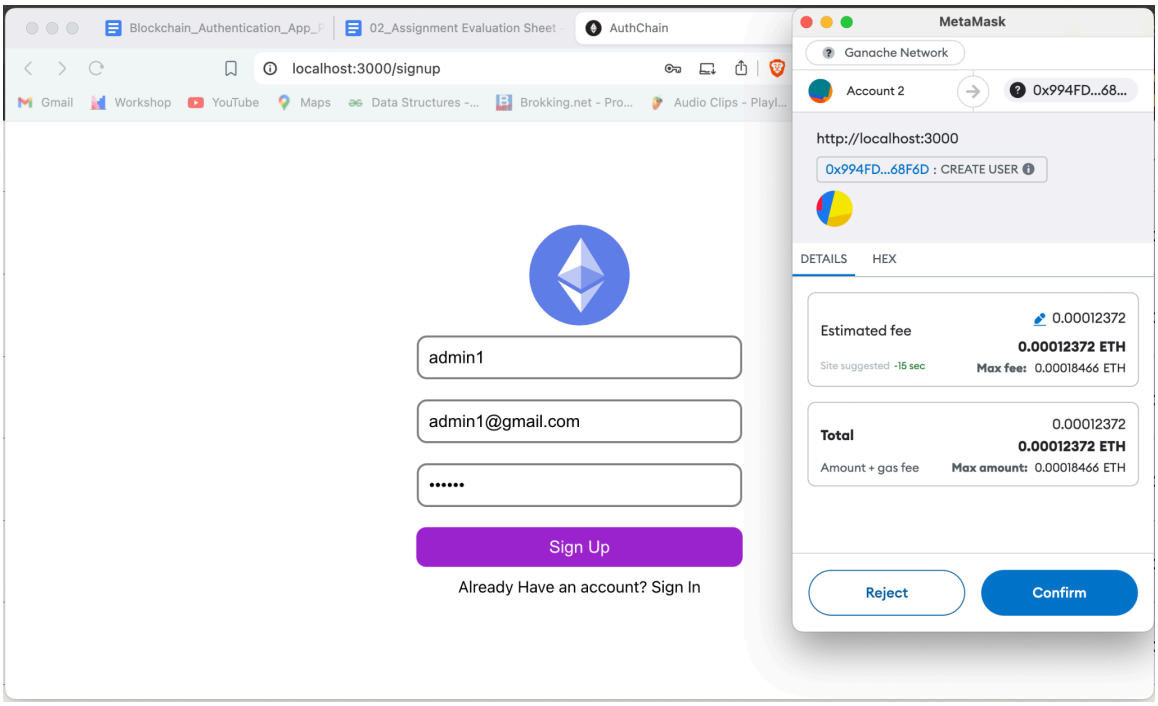


Figure 1: Deployed Project

Figure 2: Signup and Login



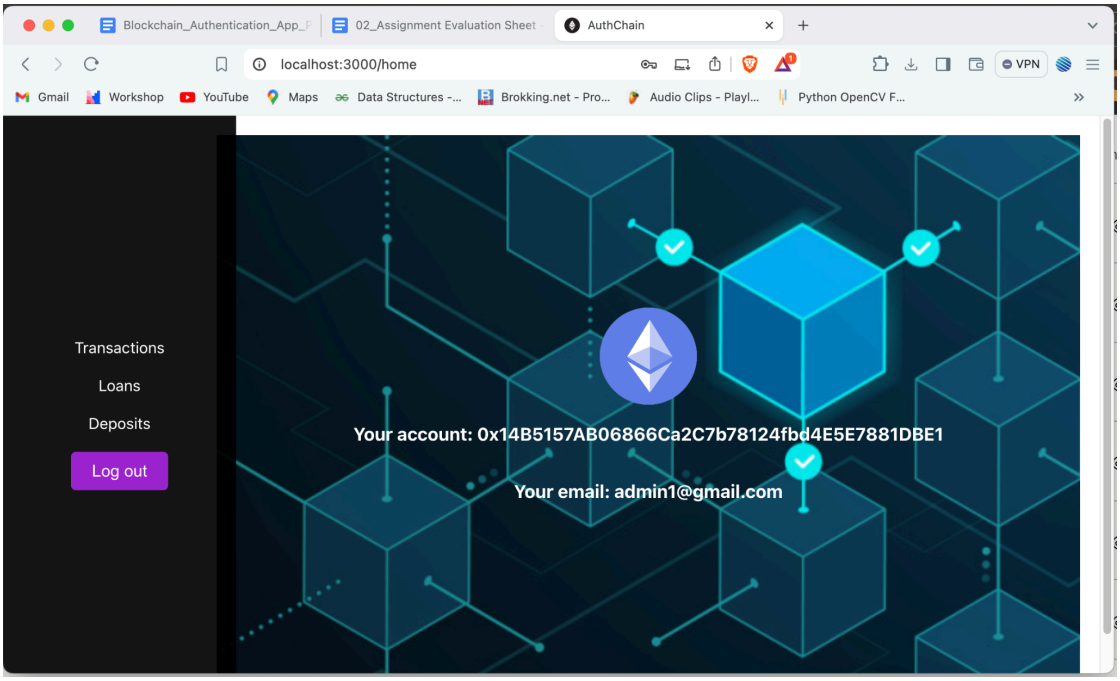


Figure 3: Backend Client (Ganache Local Server)