Voting DApp Project Report

1. Project Proposal

1.1 Objective

The primary objective of this project is to create a **decentralized voting DApp** using blockchain technology, focusing on achieving a high level of transparency and immutability in the voting process. Traditional voting systems often face issues such as vote tampering and manipulation of results, but blockchain's trustless nature can help address these challenges. By leveraging decentralization, blockchain ensures the integrity of voting data and prevents any tampering or manipulation, thereby increasing the credibility of the voting system.

The project will utilize **smart contracts** to automate the voting process, minimizing human intervention. Smart contracts are immutable and self-executing programs that enforce predefined rules. Users will be able to cast their votes on the blockchain, with all voting actions being permanently recorded and visible to the public, ensuring transparency. This automation will reduce the risk of human error or fraud, promoting fairness and accuracy.

Key features include:

- **Decentralized Governance**: Community-based decision-making without centralized control.
- **Real-time Voting Results**: Users can monitor voting progress and results as they happen.
- **Encrypted Voter Identity**: Voter privacy will be protected using encryption techniques.

These features will be elaborated further in **Chapter 5** of the report.

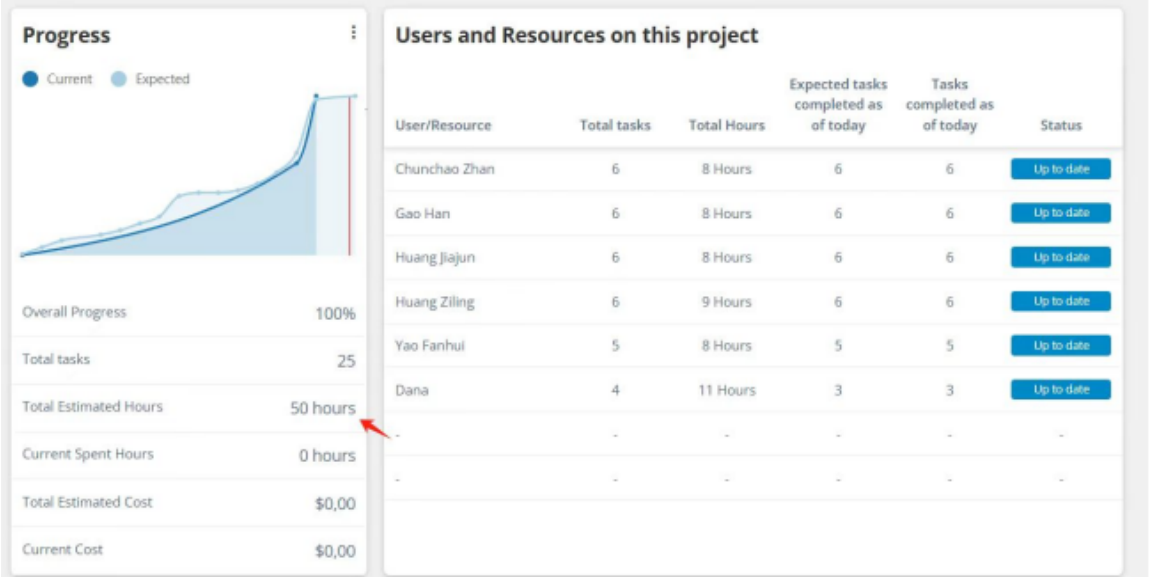
1.2 Scope

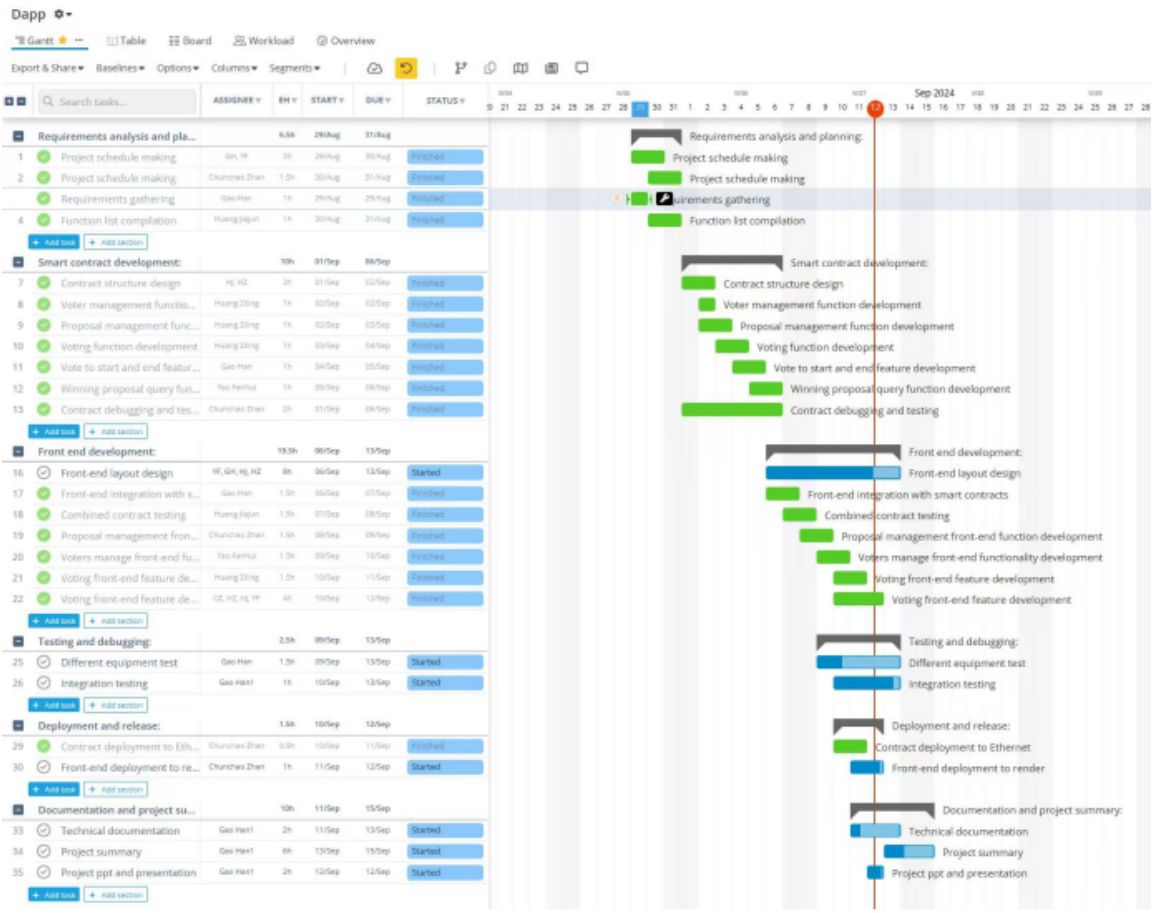
The project scope includes the following key areas:

1. **Smart Contract Development**:
    - **Smart contracts** will be used to manage voting processes such as voter registration, proposal submission, and vote counting. Thorough testing

and auditing will ensure the security and reliability of these contracts.

2. **Front-end User Interface**:
    - A **user-friendly interface** will be developed for voters, allowing them to register, view proposals, submit votes, and track results in real-time. The UI will support both mobile and desktop platforms and adapt to screen orientation.
3. **Proposal Management**:
    - The system will automatically publish new proposals and allow users to participate in real-time. Voting results will be updated instantly.
4. **Project Timeline**:
    - The project is divided among team members and allocated 50 hours for completion. Team members are assigned specific tasks, with task distribution outlined. The team uses a **Kanban-style board** for managing tasks, divided into key areas such as requirements analysis, smart contract development, front-end development, testing, and deployment.





2. Research and Literature Review

Decentralized voting platforms are designed to bring transformative improvements to traditional voting systems by enhancing transparency, credibility, and efficiency through blockchain technology. The literature review focuses on existing decentralized voting platforms and the challenges faced in implementing such systems.

2.1 Case Studies

- **Decentralized Autonomous Organizations (DAOs)**: Zhang, X., and Chen, Q. (2021) analyzed DAO governance structures, highlighting the use of smart contracts and token economies. However, issues such as **token centralization** and **low voter participation** were noted.
- **Compound Platform**: Williams, K., and Lee, A. (2023) studied the **Compound** governance system, which is based on **COMP tokens**. While the token-based governance model encourages community involvement, challenges like **centralization** and **lack of voting incentives** persist.

2.2 Challenges and Solutions

Key challenges for decentralized voting platforms include:

- **Identity Verification**: Liu, M., and Zhao, Y. (2020) proposed multi-factor authentication and biometric technologies as solutions for ensuring voter authenticity.
- **Data Privacy**: Patel, S., and Kumar, R. (2021) explored using **encryption** and **zero-knowledge proofs** to balance transparency with privacy.
- **Voting Result Transparency**: Johnson, L., and Davis, H. (2022) suggested that well-designed voting mechanisms and auditing systems are necessary to maintain fairness and accuracy.

2.3 Conclusion

While decentralized voting platforms have the potential to improve governance transparency and credibility, challenges such as **identity verification**, **data privacy**, and **transparency** in voting results remain. This project aims to address these issues by incorporating blockchain features such as encryption and decentralized governance mechanisms.

3. Technology and Tools

For the development of the Funds Transfer DApp, various technologies and tools were employed to achieve the functionality, security, and user experience required for decentralized applications. Below are the main technologies used in this project:

- **Ethereum Blockchain**: The core blockchain technology used for peer-to-peer funds transfers, ensuring transparency, security, and decentralization. Solidity is used for writing smart contracts that handle the transfer of funds between users.
- **Solidity**: The programming language used to write the smart contracts. Solidity is tailored for Ethereum and other blockchain platforms, and it ensures secure handling of transactions and data.
- **MetaMask**: A browser-based cryptocurrency wallet that allows users to interact with the Ethereum blockchain directly from their browsers. MetaMask is used for user authentication and signing transactions in the DApp.

- **Web3.js**: A JavaScript library that allows the DApp to interact with the Ethereum blockchain. Web3.js is used for sending and receiving funds, interacting with smart contracts, and fetching blockchain data in real-time.
- **Flask**: A lightweight Python web framework used for backend services like user registration, authentication, and data storage. Flask is responsible for handling communication between the front end and the smart contracts.
- **MariaDB**: A relational database used to store user registration data, transaction history, and other important information in the backend.
- **HTML/CSS/JavaScript**: These web technologies were used for developing the front end of the DApp. They ensure that the user interface is responsive, intuitive, and provides an efficient way for users to interact with the application.

---

4. Design and Development

The design and development of the DApp followed a structured approach, focusing on ensuring security, scalability, and ease of use. Below are the key phases of the design and development process:

- **User Interface Design**: The front-end interface was designed to be simple, clean, and user-friendly. The design ensures that users can quickly register, log in, and transfer funds without confusion. HTML, CSS, and JavaScript were used to build the UI, and the integration with MetaMask provides a smooth blockchain experience.
- **Database Design**: The backend database, MariaDB, was designed to store user data, including hashed passwords, transaction history, and user registration information. The data is structured to allow fast querying while ensuring security through encryption and access control.
- **Smart Contract Design**: The smart contracts were designed with security in mind. The contracts handle functions such as user registration, fund transfers, and fee calculation. Extensive testing was done to ensure the contracts are secure and resilient against attacks, such as reentrancy and overflow issues.
- **Security**: Passwords are hashed using the bcrypt algorithm to ensure that even if the database is compromised, user data remains secure. Additionally, user inputs are sanitized to prevent injection attacks.

- **Error Handling**: Error-handling mechanisms were implemented in both the front end and the smart contracts to handle issues like insufficient gas fees, incorrect transaction amounts, and unregistered users trying to send funds.

---

5. Implementation

Smart Contracts:

The core of the DApp lies in its smart contracts, written in Solidity. The main functionalities of the smart contracts include:

- **User Registration**: The smart contract maintains a mapping of registered users by their Ethereum addresses. Users can register themselves by invoking the `register()` function, which marks them as registered.
- **Funds Transfer**: The `transferFunds()` function allows users to send ETH to other registered users. It ensures that the sender and recipient are registered, calculates the transaction fee, and transfers the funds. The fee is automatically deducted and sent to the contract owner (admin).
- **Transfer Fee**: The contract includes a predefined percentage for the transaction fee, which can be updated by the contract owner. This allows flexibility in adjusting fees based on network conditions.
- **Security**: Various checks are included in the smart contract to prevent unauthorized access and incorrect transaction execution. The contract ensures that only registered users can transfer funds, and all transactions are transparent and traceable.

Frontend:

The frontend of the DApp was built using HTML, CSS, and JavaScript with the following functionalities:

- **User Registration and Authentication**: The frontend interacts with the backend (Flask) for user registration and authentication. It uses a form where users enter their usernames, passwords, and other credentials. Once submitted, the backend hashes the password before storing it in the database.
- **MetaMask Integration**: The DApp integrates with MetaMask to allow users to connect their Ethereum wallet. Users can initiate transactions directly

from the interface, and the DApp fetches the account balance and transaction details using Web3.js.

- **Funds Transfer Interface**: Users can transfer funds by entering the recipient's address and the amount they wish to send. The interface provides feedback on the transaction status and updates users with a success or failure message.

Backend:

The backend was implemented using Flask and performs the following functions:

- **User Management**: The backend manages user registration and login. When a user registers, their password is hashed using bcrypt before being stored in the MariaDB database. During login, the entered password is compared with the hashed password to verify authenticity.
- **Database Interaction**: The backend interacts with MariaDB to store and retrieve user data, such as registration details and transaction history. It ensures that only authorized users can access or modify the data.
- **API for Smart Contract Interaction**: The backend communicates with the Ethereum blockchain through Web3.js, fetching user balances, sending funds, and updating the smart contract's state.

Testing:

Testing was an integral part of the implementation process to ensure the DApp's functionality, security, and performance:

- **Smart Contract Testing**: Smart contracts were tested in a local Ethereum environment (Ganache) to simulate transactions and validate the contract's functionality. Tests were performed to check edge cases, such as insufficient funds and unauthorized access.
- **Frontend Testing**: The frontend was tested to ensure that users can interact seamlessly with the blockchain through MetaMask. Testing also focused on user experience, form validation, and error handling.
- **Backend Testing**: The backend was tested to ensure that the registration and login processes work as expected. It also involved testing the database for security vulnerabilities, such as SQL injection.

This implementation ensures a seamless and secure user experience for registering, logging in, and transferring funds in a decentralized manner.

6. Challenges

Technical Challenges

- **Smart Contract Design and Security**: Ensuring the correctness and security of the smart contract logic was one of the core challenges. Any vulnerability in the contract could lead to vote tampering or malicious voting. We thoroughly tested the contract to ensure that voters are registered, duplicate voting is prevented, and results are correctly calculated after voting ends.
- **Decentralization and Data Storage**: Blockchain provides immutability, but storing large-scale voter or proposal data on-chain leads to high transaction costs. We opted to store non-core data (like proposal details) off-chain while keeping core voting data on-chain for transparency.
- **Integration with MetaMask or Other Wallets**: Ensuring stable dApp performance across different browsers and devices when interacting with MetaMask wallets posed compatibility challenges. We needed to ensure smooth interactions between the dApp and the blockchain across various environments.

User Experience (UX) Challenges

- **Usability**: Blockchain applications can be complex, especially for users unfamiliar with the technology. We designed a simple, intuitive process for connecting wallets, registering, voting, and displaying results to lower the technical barriers for users.
- **Real-Time Updates and Feedback**: On-chain transaction confirmation takes time. To avoid users thinking their vote failed, we implemented loading and status feedback mechanisms to inform them of voting progress.

Security and Privacy Challenges

- **Data Privacy and Anonymity**: Blockchain’s transparency raised privacy concerns. We focused on ensuring voter anonymity, using techniques like zero-knowledge proofs to enhance privacy while maintaining transparency in voting results.

- **Preventing Malicious Attacks**: Smart contracts, once deployed, cannot easily be modified. We performed thorough audits to prevent common attacks like Denial-of-Service (DoS) and reentrancy attacks and ensured that protective measures were in place.

Compliance and Legal Challenges

- **Legal Compliance and Ethical Review**: We followed NTU's guidelines and relevant laws to ensure the voting process was fair, transparent, and compliant with privacy regulations. Close collaboration with the university's ethics committee ensured adherence to regulations.

Blockchain Performance and Cost

- **On-Chain Operation Costs**: Each on-chain operation incurs gas fees, and as voting scales, so do costs. We optimized smart contract logic to reduce unnecessary on-chain operations and explored Layer 2 solutions to cut costs.
- **Scalability Issues**: Handling a large number of voters or proposals could bottleneck performance. We explored ways to enhance scalability without compromising data accuracy or security.

Development Tools and Debugging

- **Development Environment Debugging**: Debugging dApps, especially identifying transaction failures, is challenging due to the need to check on-chain logs. This increased complexity in development.

---

7. Outcome

Implementation of the Voting System

- **Decentralized Voting Mechanism**: We deployed a fully decentralized system where voting processes and data are stored and verified on the blockchain. Users participate by connecting their MetaMask wallets, ensuring the voting process is tamper-proof and transparent.
- **Security of Smart Contracts**: Our smart contracts went through rigorous reviews and testing. The system securely manages proposals and voter registration, ensuring that each voter can cast only one vote. Security

measures were implemented to prevent vulnerabilities like reentrancy attacks.

Integration of Multiple Voting Systems

- **Managing Multiple Voting Activities**: We enabled multiple independent voting activities to run concurrently, allowing users to switch between elections. Each vote has its own proposals and voter base, maintaining election transparency.
- **Dynamic Display of Proposals and Results**: The system dynamically updates and displays the status of each voting activity. Results are automatically calculated and displayed once voting ends.

User Experience Optimization

- **User-Friendly Interface**: We built a simple interface, reducing the technical barriers to blockchain use. Users can easily register, view proposals, vote, and see the final results. Real-time feedback is provided to improve the user experience.
- **Real-Time Updates**: After voting, the system instantly shows the progress and results, reducing wait times and providing a smoother experience.

Compliance and Ethical Review

- **Ensuring Compliance**: We adhered to NTU’s guidelines, especially regarding data privacy and protection. Voter data was encrypted, and the voting process complied with NTU’s ethical standards.
- **Privacy Protection**: Data anonymization measures were implemented to ensure that voters’ identities and behavior remained private.

Cost Optimization and Performance Enhancement

- **Gas Fee Optimization**: Smart contract optimizations significantly reduced gas fees, making the system more cost-effective, especially when managing multiple concurrent voting activities.
- **System Scalability**: Adjustments to data storage and smart contract logic improved the system's scalability, ensuring efficient operation even with a large voter base.

Automated Smart Contract Processes

- **Automated Voting Settlement and Result Display**: Administrators can easily start or end voting, and the system automatically calculates and displays results. For multiple voting activities, independent result displays are provided.

---

8. Ethical Considerations and Compliance

Ethical Blockchain Use

- **Privacy Protection**: Blockchain's public nature can expose voter behavior, raising privacy concerns. We avoided linking personal information to the blockchain and explored zero-knowledge proofs for anonymity. In this project, mock data was used for testing.
- **Balancing Decentralization and Transparency**: While blockchain is transparent, we focused on ensuring that only voting results are public without revealing individual choices to prevent manipulation.
- **Ethical Use of Technology**: The system avoids collecting unnecessary data, and no entity can manipulate the voting outcome. We ensured the voting process remained fair and transparent.

Compliance

- **Data Protection and Privacy**: In line with NTU's Personal Data Protection Act (PDPA), voter data was securely stored and processed, adhering to all legal guidelines.
- **Informed Consent**: Participants were informed about how the system works and how their data is used, ensuring transparency.
- **Ethical Review**: We followed NTU's ethical review process to ensure the project complied with the institution's standards.

---

9. Reflection

After completing the blockchain voting DApp, we conducted a thorough self-assessment to identify challenges and areas for improvement.

Project Management

Time management was an issue, which led to delays in feature development. Although we achieved our goals, better time planning and team coordination

from the start would have helped. We plan to adopt agile development methodologies and more frequent progress reports to improve team efficiency.

Technical Challenges

Our smart contract development skills, especially regarding gas optimization and security mechanisms, needed improvement. We encountered challenges in testing and deploying contracts, and deeper knowledge of Solidity and security tools will help avoid these issues in future projects.

User Experience

While the interface was designed to be simple, user feedback indicated that improvements were needed, especially on mobile devices. Ensuring better cross-device compatibility and improving real-time feedback mechanisms will be key areas for future projects.

Compliance and Privacy

Although we followed NTU's guidelines, privacy and anonymization could be enhanced using more advanced techniques like zero-knowledge proofs. Strengthening our understanding of data privacy and compliance is essential.

Documentation and Testing

Our technical documentation and testing processes were not always updated promptly, which made debugging difficult later. In future projects, we will focus on synchronous documentation and more systematic testing, especially regarding smart contract security and frontend-backend interactions.

Despite many challenges, we made significant progress in several areas. In future projects, we aim to improve project management, technical skills, user experience, and compliance to deliver even better results.

Appendices:

- **Smart Contract Code**: See attached files
- **Backend Code**: See attached files
- **Frontend Code**: See attached files
- **Github link:** https://github.com/natalieybl/SC6113_Voting

- **Teamwork management:** https://app.instagantt.com/shared/s/XM8nEZD6rXlhQ7scBedG/latest
- **Render:** https://sc6113-voting.onrender.com