Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Simple Storage DApp Implementation Procedure

Phase 1: Project Setup

1. Initialize React application with required dependencies (ethers.js, toast notifications)
2. Configure environment variables for contract address and network
3. Set up project structure with components for wallet connection and contract interaction

Phase 2: Wallet Connection System

1. Detect MetaMask availability in user's browser
2. Request account access using eth_requestAccounts method
3. Establish Ethers.js provider connection to blockchain
4. Handle account/network changes with real-time event listeners
5. Implement clean disconnection procedure

Phase 3: Smart Contract Integration

1. Load contract ABI and address from configuration
2. Create contract instance using Ethers.js Contract class
3. Implement read function (get) to fetch current stored value
4. Implement write function (set) to update value via transactions

Phase 4: User Interface Flow

1. Display connection status and wallet information
2. Show current stored value with refresh capability
3. Provide input form for new value submission
4. Handle transaction lifecycle (pending, confirmation, error)
5. Update UI automatically after successful transactions

Phase 5: Transaction Management

1. Initiate transaction with user confirmation in MetaMask
2. Track transaction status using transaction hash
3. Provide visual feedback during pending state
4. Handle success/failure cases with appropriate notifications
5. Refresh contract data after transaction confirmation

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Coding Phase: Pseudo Code / Flow Chart / Algorithm

Phase 6: Error Handling & UX

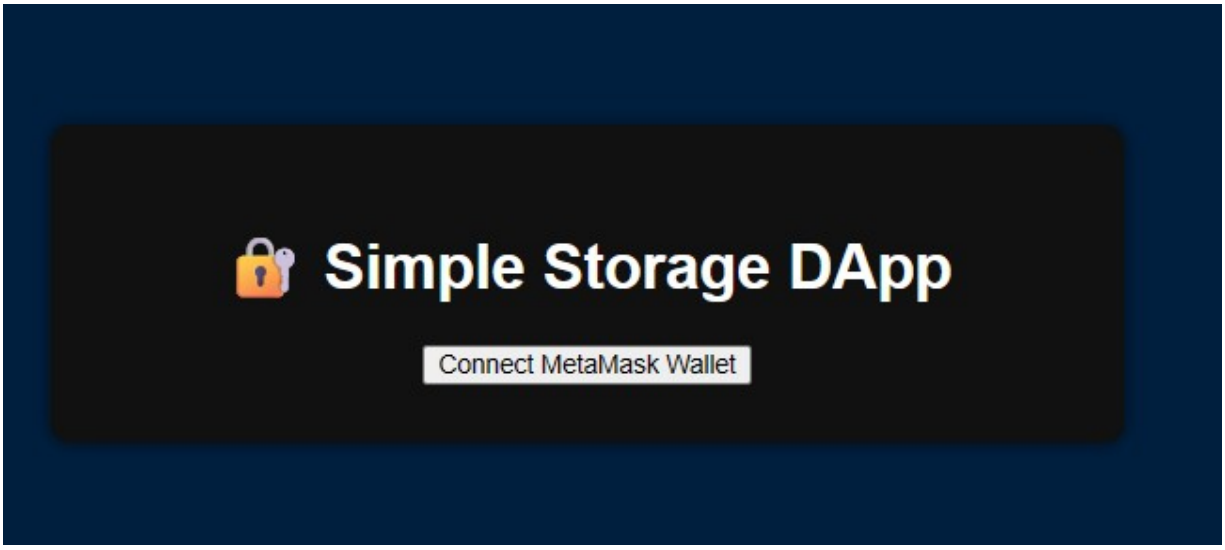
1. Validate user inputs before transaction submission
2. Handle common MetaMask errors (rejected transactions, wrong network)
3. Provide helpful error messages and recovery suggestions
4. Ensure responsive design for different screen sizes

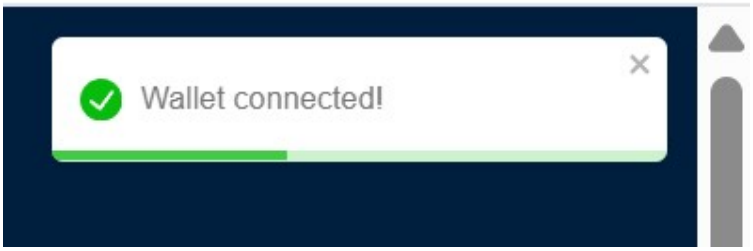
This procedure creates a complete feedback loop where user actions trigger blockchain interactions, which in turn update the UI state, providing a seamless Web3 experience.

* Softwares used

- **React**
- **Ether.js**
- **Metamask**
- **Solidity language**
- **Node.js**
- **Web3**

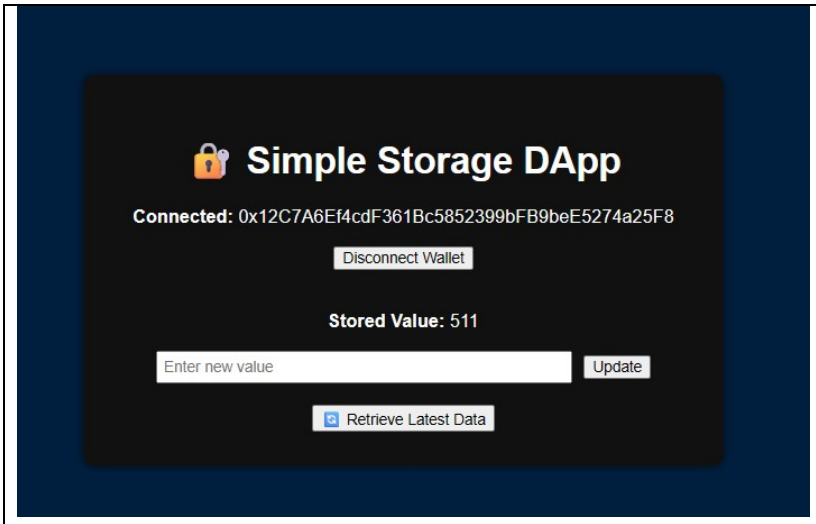
* Implementation Phase: Final Output (no error)





*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)



This procedure creates a complete feedback loop where user actions trigger blockchain interactions, which in turn update the UI state, providing a seamless Web3 experience.

* Observations

Technical Observations

- Web3 Integration: Demonstrates seamless blockchain-frontend connection
- Real-time Updates: UI automatically reflects blockchain state changes
- Transaction Lifecycle: Shows complete flow from initiation to confirmation
- Error Handling: Robust handling of common Web3 scenarios (rejected transactions, wrong network)

User Experience Observations

- Wallet Integration: Smooth MetaMask connection/disconnection flow
- Visual Feedback: Clear status indicators for all operations
- Responsive Design: Works across different devices and screen sizes
- Loading States: Proper handling of asynchronous blockchain operations

ASSESMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment. Two sheets per experiment (10-20) to be used.*