1. **Project Proposal Document**

**Introduction Government Procurement DApp**

This smart contract outlines a basic procurement process where the government can initiate a contract, the supplier can mark it as completed, and the government can release payment once the contract is verified as completed.

**Project Title: Decentralized Government Procurement System**

**Clear Objective:**

To create a decentralized application (DApp) that streamlines the government procurement process, ensuring transparency, accountability, and automation in contract execution and payment release.

**Problem Description:**

Traditional procurement systems used by governments are often prone to inefficiencies, corruption, delays, and lack of transparency. Suppliers have little visibility into payment timelines, and government agencies face challenges in tracking contract fulfillment milestones.

**Proposed Solution:**

The solution is a blockchain-based procurement DApp that automates:

* Government creation of contract offers with upfront funding.
* Supplier marking of contract completion.
* Government verification and approval before automatic payment release.

All actions are stored on-chain for full transparency and traceability.

**Key Features:**

* **Role-based Access**: Only government can create and pay; only suppliers can mark completion.
* **Smart Contract Logic**: Controls access, state, and funds transfer.
* **Contract Lifecycle**: Status updates — Created → Completed → Paid.
* **Immutable Records**: All actions recorded on-chain for auditing.
* **Escrowed Payments**: Government pre-funds contracts and funds are only released when completed.

**Technical Stack:**

| **Layer** | **Technology** |
| --- | --- |
| Blockchain | Ethereum (via Hardhat localhost) |
| Smart Contracts | Solidity |
| Frontend | React.js with plain HTML/CSS |
| Wallet Integration | MetaMask |
| Web3 Library | Ethers.js (v6) |
| Dev Tools | Hardhat, Node.js, VS Code |

**Security Measures:**

* onlyGovernment and onlySupplier modifiers to control function access.
* Checks for correct contract status transitions.
* require statements to prevent invalid transactions.
* Use of transfer() for payments to prevent reentrancy.
* Full test environment on local Hardhat network.
* No untrusted third-party dependencies.

**Implementation Plan:**

| **Week** | **Deliverables** |
| --- | --- |
| Week 1 (done) | Finalize smart contract (Solidity), deploy on Hardhat |
| Week 2 | Build React frontend, connect via Ethers.js |
| Week 3 | Implement contract creation, completion, and payment workflows |
| Week 4 | Perform testing, styling, and bug fixes |
| Week 5 (by May 5) | Record demo video, upload, and submit screencast link |

**Submission Reminders:**

* Submit this proposal by **April 30, 2025**
* Submit project video demo by **May 9, 2025**