

| **Title: Prepare design document and Plan of project** |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Objective:** Chapter No.3 of Mini Project Report will include detailed design document and plan of implementation of the project

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

|  | **At the end of successful completion of the course the student will be able to** |
| --- | --- |
| CO2 | Identify various hardware and software requirements for problem solution |
| CO5 | Prepare a technical report based on the Mini project. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

**1.**

**2.**

**3.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**The students are expected to prepare chapter no 3 in the format given below**

**Chapter 3**

**Design Document and Project plan**



**User Interface Design**

### UI/UX Design Elements

The user interface is designed with a clean, modular layout that ensures ease of use for all stakeholders. The primary design elements include:

* Minimalist Dashboard UI with role-specific navigation
* Color-coded status indicators for drug verification (e.g., green = authentic, red = counterfeit)
* Accessible Typography and Contrast following WCAG 2.1 standards
* Responsive Design for both desktop and tablet views
* Smart notifications for transaction confirmations and errors
* Form validation with real-time feedback during drug entry or transfer

### Wireframes

* **Login/Connect Wallet Page**: Simple interface using RainbowKit for wallet connection.
* **Role Application Page**: Dropdown for role selection, form fields for organization info.
* **Manufacturer Dashboard**: Drug batch form with fields like Name, Batch ID, Expiry Date, etc.
* **Distributor View**: Table of available drugs with actions to accept or ship.
* **Retailer Interface**: Scan/verify interface with lookup results from blockchain.
* **Consumer Page**: Enter or scan drug ID to verify source and authenticity.

### c. Navigation Flow

Landing Page -> Wallet Connect -> Role Selection -> Role-Specific Dashboard -> Action Pages (Register, Transfer, Verify Drug)

1. **Introduction:**

**Purpose of the Document:**This document outlines the user interface (UI) and user experience (UX) design for the **Blockchain-Based Pharmaceutical Supply Chain Management System**. It serves as a blueprint for the development team, UI/UX designers, testers, and stakeholders. The document ensures that the system offers an intuitive, accessible, and efficient user experience across all modules.

**Expected Audience:**

* Project Managers
* Frontend and Backend Developers
* UI/UX Designers
* QA Engineers
* Stakeholders and Domain Experts

**Scope of the Project:**

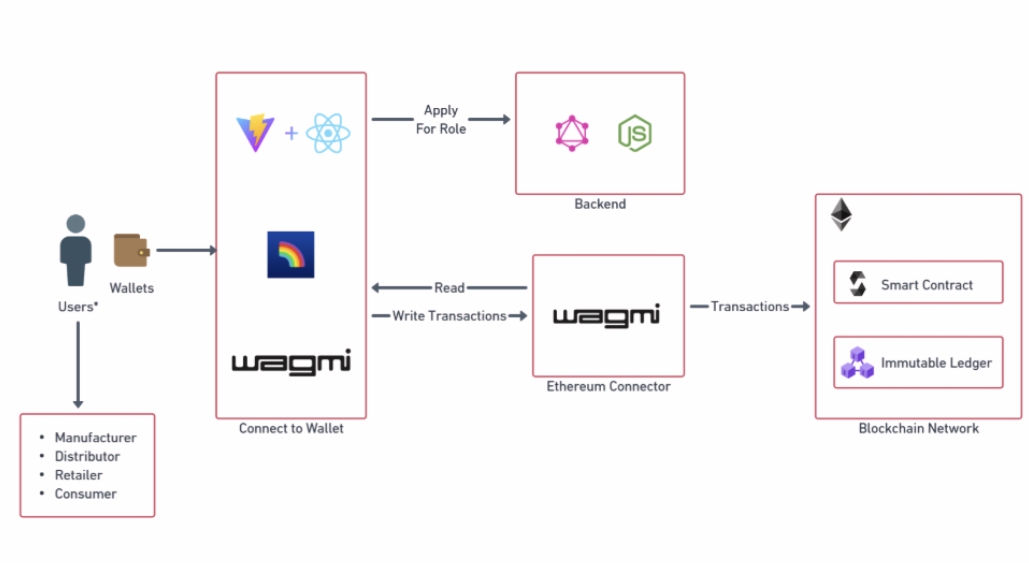
1. **Drug Tracking:** The system will provide end-to-end tracking of pharmaceutical products from manufacturers to consumers.
2. **Blockchain Ledger:** Every transaction will be recorded on a secure and tamper-proof blockchain ledger, ensuring transparency and accountability.
3. **Product Authentication:** Unique product identifiers (QR codes, RFID tags) will be used for verifying drug authenticity at every stage of the supply chain.
4. **Stakeholder Access:** Manufacturers, distributors, pharmacies, and consumers will have role-based access to relevant supply chain data.
5. **Web-Based Application:** The system will be accessible via a user-friendly web application, ensuring seamless interaction for all stakeholders.
6. **Scalability:** The project will be designed to scale for multiple regions and regulatory requirements, making it adaptable to different markets.

### References

* **SRS Document** – Internal Software Requirements Specification for the Blockchain-Based Pharmaceutical Supply Chain Management System, detailing functional and non-functional requirements, user roles, and system constraints.

1. **System Overview**

**2.1 System Architecture:**



### 1. Users

* **Participants**: Manufacturer, Distributor, Retailer, Consumer.
* **Access**: Each user interacts with the system using **wallets** (e.g., MetaMask), which serve as identity and transaction tools.

### 

### 2. Wallet Connection (Frontend)

* Built using **React + Vite** for UI and **RainbowKit** for wallet integration.
* Uses **Wagmi**, a library for Ethereum interactions, to:  
  + Connect the user’s wallet.
  + Manage authentication via wallet signatures.
  + Send and retrieve blockchain data.

### 3. Role Application (Backend Interaction)

* When a user logs in, they can **apply for a role** (e.g., manufacturer, distributor).
* This request is sent to the **backend**, which is built with:  
  + **Node.js**: Handles server logic.
  + **GraphQL**: Manages role requests and other queries/mutations.
* The backend verifies and assigns roles, storing any off-chain metadata as needed.

### 4. Ethereum Connector (Wagmi)

* Acts as a bridge between the **frontend** and **blockchain**.
* Handles:  
  + **Read operations**: Fetches data like drug status, origin, etc.
  + **Write operations**: Sends transactions such as creating a drug batch, transferring ownership, or marking deliveries.

### 5. Blockchain Network

* Built on **Ethereum**, interacting via smart contracts.
* Components:  
  + **Smart Contracts**: Implement business logic (e.g., transfer drug ownership, verify authenticity).
  + **Immutable Ledger**: Ensures all transactions are tamper-proof and permanently recorded.

### 2.2 Design Goals

The proposed decentralized pharmaceutical supply chain management system is designed with the following core principles in mind:

### 1. Scalability

* The system must be capable of handling a large number of users and transactions across multiple supply chain participants globally.
* Use of modular architecture and stateless backend services ensures ease of scaling.

### 2. Security

* All transactions and user authentications are handled through blockchain-based wallets to prevent identity theft.
* Data immutability and role-based access via smart contracts mitigate fraud and unauthorized access.

### 3. Performance

* Fast frontend rendering using **React + Vite**.
* Smart contract optimizations for low gas usage.
* Efficient use of off-chain storage (MongoDB/IPFS) to reduce on-chain congestion.

### 4. Maintainability

* Clean separation of concerns between frontend, backend, and blockchain layers.
* Reusable smart contracts and API components.
* Use of TypeScript and GraphQL for better error checking and modular codebase.

### 5. Transparency and Traceability

* Each drug’s journey from manufacturing to consumption is traceable via blockchain.
* Every transaction is verifiable on-chain, increasing accountability.

.

**3. Detailed Design**

**3.1 Module Description**

#### 1. User Module

**Responsibilities:**

* Allow users to connect their wallet.
* Enable role-based login and interaction (manufacturer, distributor, retailer, consumer).

**Interaction:**

* Uses RainbowKit for wallet connection.
* Sends GraphQL mutation to backend for role application.
* Reads role status and drug history from the blockchain via Wagmi.

#### 2. Role Management Module

**Responsibilities:**

* Assign roles to connected users.
* Validate and approve role requests from backend.

**Interaction:**

* GraphQL API sends role request to backend.
* Backend interacts with smart contract to assign roles on-chain.

#### 3. Drug Management Module

**Responsibilities:**

* Manufacturers can register a drug batch
* Distributors and retailers update transfer and delivery details.
* Consumers can scan and verify drug authenticity.

**Interaction:**

* Write operations trigger smart contract functions (e.g., createDrug, transferDrug).
* Wagmi handles contract calls, while updates are reflected in the UI.

#### 4. Blockchain Integration Module

**Responsibilities:**

* Manage smart contract transactions.
* Ensure correct sequencing of state changes on Ethereum.

**Interaction:**

* Smart contracts are written in **Solidity**.
* Wagmi is used for contract interaction from frontend.
* Uses Sepolia Testnet or mainnet Ethereum for deployment.

#### 5. Backend Services Module

**Responsibilities:**

* Handle role assignment, verification, and metadata storage.
* Manage GraphQL APIs.

**Interaction:**

* Built using **Node.js + Apollo Server**.
* Stores off-chain data like metadata or auxiliary information in **MongoDB**.
* Verifies role requests before writing to blockchain.

#### 6. Data Storage Module

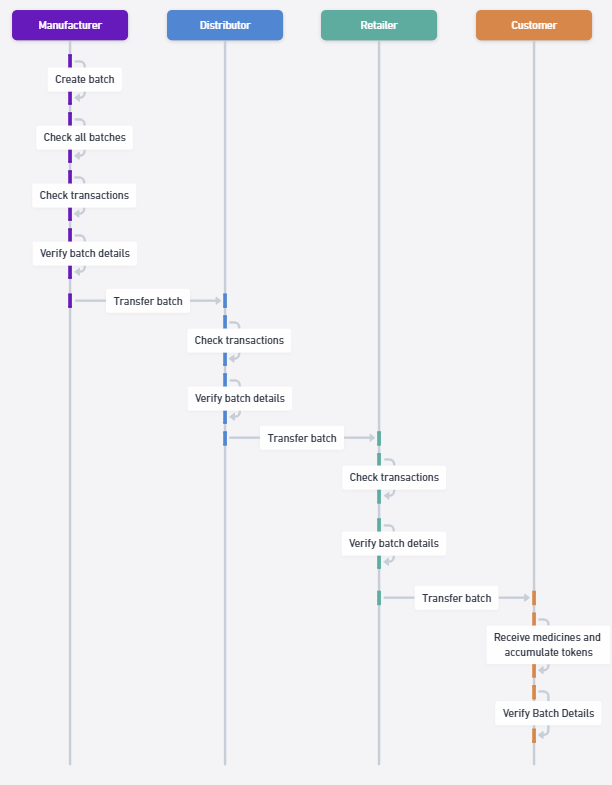
**Responsibilities:**

* Off-chain storage of non-sensitive metadata and UI-related data.
* Maintain IPFS hashes for decentralized file storage.

**Interaction:**

* MongoDB for fast querying.
* IPFS for secure, distributed file storage.

**3.2 Data Flow & Components**



### 3.3 Database Design

| **Field** | **Type** | **Required** | **Unique** | **Default** | **Description** |
| --- | --- | --- | --- | --- | --- |
| \_id | ObjectId | Yes | Yes | Auto-generated | Primary key generated by MongoDB |
| name | String | Yes | No | - | Full name of the user |
| walletAddress | String | Yes | Yes | - | Ethereum wallet address, must be unique |
| email | String | Yes | No | - | User's email address |
| companyName | String | Yes | No | - | Name of the user's organization |
| role | String | Yes | No | - | User role (e.g., Manufacturer, Admin) |
| ipfsHash | String | Yes | No | - | IPFS CID or hash for file reference |
| createdAt | Date | No | No | Date.now | Timestamp when the document is created |
| assigned | Boolean | No | No | false | Whether the user has been assigned a task |

### 3.4 External Interfaces

* Ethereum Blockchain (via Solidity)
* Hardhat (smart contract development)
* Remix IDE (smart contract testing/deployment)
* IPFS (for decentralized file storage)
* ethers.js (blockchain interaction in frontend)
* Wagmi (React hooks for Ethereum)
* RainbowKit (wallet connection UI)
* Node.js & Express.js (backend APIs)

**4. Project and Implementation Plan**

**4.1 Deliverables:**

* Source Code for Frontend, Backend, and Smart Contracts
* User Documentation (How to Use the DApp)
* Smart Contract Documentation (ABI references and interactions)
* Postman Collection for API Testing (GraphQL endpoints)

**4.2 Team Roles and Responsibilities and delivery schedule**

| **Name of the Task** | **Developer** | **Tester** | **Approver** | **Date of Delivery** |
| --- | --- | --- | --- | --- |
| Wallet Integration | Romil | Sanika | Minav | 15/03/2025 |
| Role Management | Sanika | Romil | Minav | 10/02/2025 |
| Drug Registration Module | Romil | Minav | Sanika | 10/02/2025 |
| Transfer Logic Implementation | Sanika | Romil | Minav | 21/02/2025 |
| UI Polishing & Testing | Romil | Minav | Sanika | 23/03/2025 |
| Final Deployment | Minav | Sanika | Romil | 12/04/2025 |

**4.3 Risk Management Plan:**

* **Wallet Disconnection**: Implement reconnect logic using Wagmi hooks.
* **Smart Contract Bugs**: Use unit tests and deploy only after passing Remix + Hardhat validations.
* **Data Inconsistency**: All role-sensitive operations are on-chain; fallback to read only queries in case of backend failure.
* **Security**: Enforce wallet authentication, smart contract permission checks, and IPFS hash validation.

**5. Testing & Deployment Plan**

### 5.1 Testing Strategy

* **Unit Testing**: Conducted using Hardhat and Chai to test smart contract logic independently.
* **Integration Testing**: Performed to validate interactions between the frontend (React + Wagmi) and backend (GraphQL + Node.js), and smart contracts.
* **System Testing**: Covers end-to-end workflows for each user role (manufacturer, distributor, retailer, consumer) from login to drug verification.
* **User Acceptance Testing (UAT)**: Simulated by team members representing different stakeholder roles to ensure usability and completeness.

### 5.2 Deployment Plan

* **Frontend Deployment**: Hosted on Vercel for continuous deployment and version control.
* **Backend Deployment**: Managed via Render for Node.js GraphQL APIs.
* **Smart Contract Deployment**: Contracts deployed on Ethereum Sepolia Testnet using Hardhat and verified via Remix IDE.
* **File Storage**: IPFS used for storing non-sensitive metadata securely.
* **Rollback Strategy**: Use Git-based CI/CD rollback if an issue is encountered post-deployment. Smart contracts can be redeployed using versioned migration scripts.

