NAME-PUSHKAR KUMAR
REG_NO: 23BCE1634

SOFTWARE LAB ASSIGNMENT
PROJECT NAME: AGROCHAIN

SYSTEM ARCHITECTURE AND SRS

SRS:

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed overview of the Agri Supply Chain platform. It outlines the functional and non-functional requirements for a decentralized system that ensures transparency, traceability, and fair trade in agriculture.

1.2 Project Scope

The system enables farmers to list crops via decentralized auctions, allows logistics providers to track shipments using GPS recorded on-chain, and empowers consumers to verify product provenance through QR code scanning.

---

2. Overall Description

2.1 Product Perspective

This is a **4-Tier Decentralized Application (dApp)**. It uses a hybrid storage model where non-critical data is stored in **MongoDB**, while immutable transaction history and ownership records are stored on the **Ethereum Blockchain** and **IPFS**.

2.2 User Classes and Characteristics

- **Farmer**: Can list crops, initiate auctions, and manage harvest data.
- **Logistics/Distributor**: Responsible for updating GPS location and scanning QR codes during handover.
- **Consumer/Buyer**: Can place bids in auctions and scan QR codes to view the product's journey.
- **Administrator**: Manages user verification and monitors smart contract health.

---

3. Functional Requirements

3.1 Blockchain & Smart Contracts

- **Provenance Tracking**: The system must record every stage of the product lifecycle (Farming, Testing, Packaging, Logistics) on the ProvenanceContract.
- **Decentralized Auction**: The AuctionContract must handle bid placements, verify that bids are higher than current ones, and escrow funds until the auction concludes.
- **Shipment Tracking**: The TrackingContract must store GPS coordinates with $10^6$ precision to ensure accuracy despite Solidity's lack of floating-point support.

3.2 Application Backend (Node.js)

- **Web3 Integration**: The backend must use **Ethers.js** to sign transactions and relay them to the blockchain network.
- **QR Engine**: The system must generate a unique QR code for every new crop batch that contains a hashed reference to its blockchain ID.
- **Media Storage**: Product images and quality certificates must be uploaded to **IPFS**, with only the resulting CID (Hash) stored in the smart contract.

3.3 User Interface

- **Wallet Connectivity**: Users must be able to connect via **MetaMask** for identity verification and transaction signing.
- **Real-time Updates**: The dashboard must reflect live bidding and tracking status using WebSockets (WSS).

---

4. Non-Functional Requirements

4.1 Security

- **Immutability**: All critical data (provenance and financial bids) must be stored on-chain to prevent tampering.
- **Authentication**: Private keys must never be stored on the server; all transactions must be signed client-side via MetaMask.

4.2 Performance & Scalability

- **Hybrid Storage**: High-volume, non-essential data should be kept in MongoDB to reduce blockchain gas costs and latency.
- **Precision**: GPS tracking must update at intervals that balance data accuracy with gas consumption.

4.3 Availability

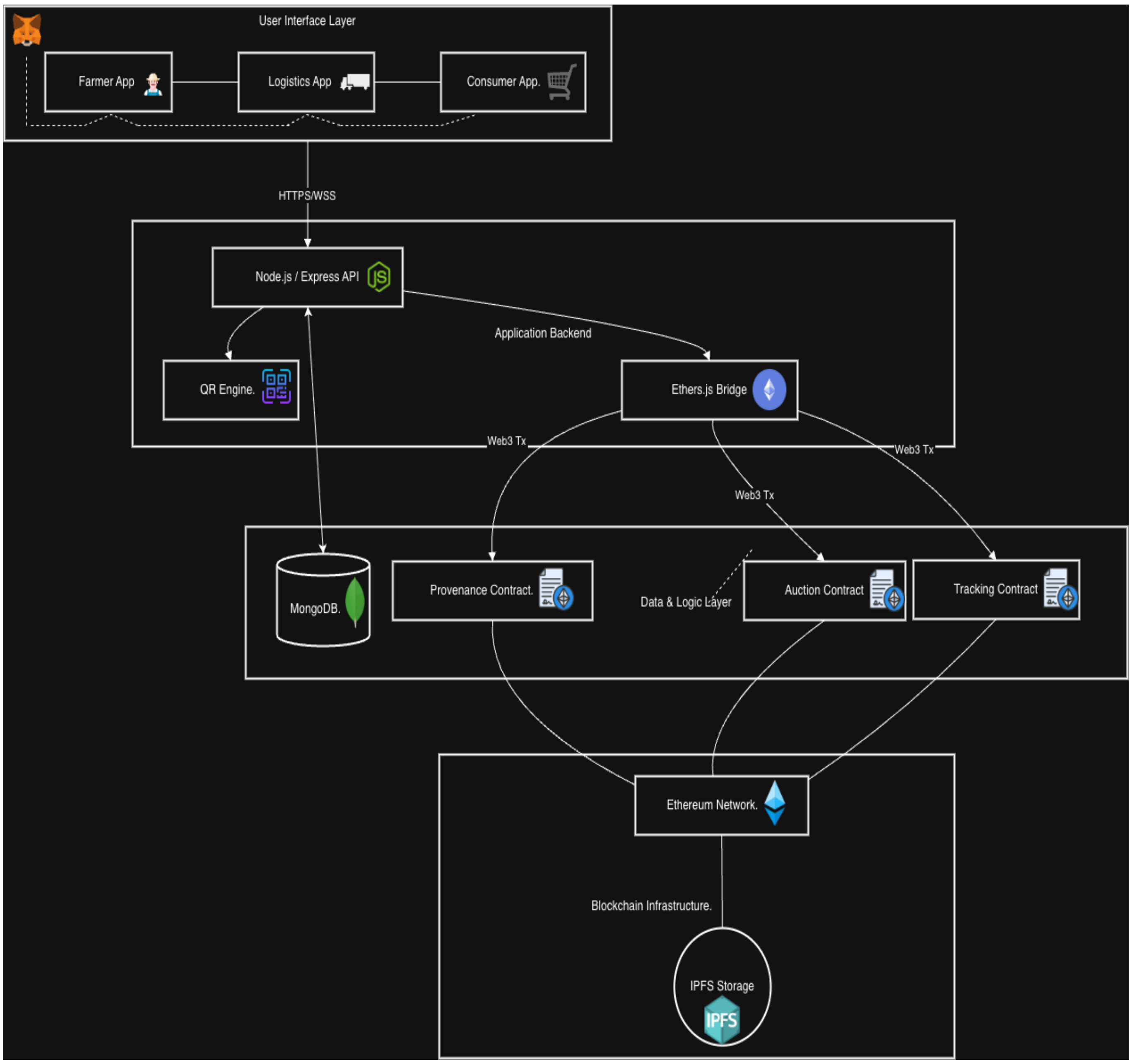
- The decentralized nature of the blockchain ensures that the core ledger remains available even if the central API server experiences downtime.

---

5. System Evolution

- **Future Scope**: Integration of IoT sensors for automated temperature and humidity logging during transit.
- **Scalability**: Moving from a local Hardhat node to an Ethereum Layer 2 (like Polygon) for lower transaction fees.

SYSTEM ARCHITECTURE :


User Interface Layer
Farmer App
Logistics App
Consumer App.
HTTPS/WSS
Node.js / Express API
Application Backend
QR Engine.
Ethers.js Bridge
Web3 Tx
Web3 Tx
Web3 Tx
MongoDB.
Provenance Contract.
Data & Logic Layer
Auction Contract
Tracking Contract
Ethereum Network.
Blockchain Infrastructure.
IPFS Storage
IPFS