**Functional & Non-Functional Requirements Document**

**Blockchain-Based Fresh Produce Supply Chain System (MVP)**

1. **Introduction**

This document outlines the functional and non-functional requirements for a **blockchain**-based supply chain system designed to improve transparency, fair pricing, and traceability in the fresh produce industry. The system leverages **Ethereum smart contracts** and a **user-friendly interface** to ensure immutable record-keeping and real-time tracking.

Key Problems Addressed

✅ Lack of transparency in produce sourcing

✅ Unfair pricing models for small-scale farmers

✅ Inability to trace produce origin

✅ High spoilage rates (future phase)

✅ Inconsistent grading (future phase)

2. **Functional Requirements (What the System Will Do)**

2.1. **User Roles & Permissions**

| **Role** |**Permissions**

Farmer - Register produce, view pricing history, receive payments

Distributor - Update shipment status, verify produce condition

Retailer - Verify produce quality, update inventory

Consumer - Scan QR code to view produce history

2.2. **Core Features**

**A. Produce Registration & Tracking**

| **Description**  **| How It Works**

Farmers register produce on blockchain -Smart contract records: - Batch ID , Farm location , Harvest , date ,Farmer’s wallet address

Generate a unique QR code per batch - QR code links to blockchain data for traceability |

**B. Fair Pricing & Smart Contracts**

| **Description**  |**How It Works**

Smart contract enforces fair pricing - Predefined pricing rules (e.g., market rates + premium for small ` ` farmers)

` - Automatically releases payment upon delivery confirmation

Farmers receive payments in crypto - Escrow-based payment system

(ETH/USDT) - Transparent transaction history |

**C. Traceability & Consumer Access**

|**Description**  | **How It Works**

Consumers scan QR code to view produce history - Displays: - Farm origin ,Harvest date , Transportation logs , Retailer details

**D. Admin & Compliance**

| **Description** | **How It Works**

Admin dashboard for dispute resolution - Allows manual overrides in case of errors

` - Monitors contract compliance

**3. Non-Functional Requirements (How Well the System Performs)**

3.1. **Performance & Scalability**

|**Description** | **How It’s Achieved**

System handles **1,000+ transactions/day** - Optimized smart contracts (gas-efficient) / Use of Ethereum Layer 2 (e.g., Polygon) if needed

QR code scan loads data in <2 sec - Caching frequently accessed blockchain data / Lightweight frontend design

3.2. **Security & Data Integrity**

| **Description** | **How It’s Achieved**

Immutable record-keeping - Ethereum blockchain ensures tamper-proof logs

Secure wallet integration - MetaMask for encrypted transactions

3.3. **Usability & Accessibility**

|  |  |
| --- | --- |
| **requirement** | **achieved by:** |

* Mobile-friendly interface - Responsive design (React.js) |
* Multilingual support (future phase) - Integration with translation APIs

3.4. **Compliance & Regulations**

* GDPR-compliant data handling achieved by Off-chain storage for sensitive data (IPFS) |

4. **System Workflow (How It Achieves Its Functions)**

**Step 1: Farmer Registers Produce**

- Enters details (origin, harvest date) → Smart contract generates a \*\*unique batch ID & QR code\*\*.

**Step 2: Smart Contract Manages Pricing**

- Automatically calculates fair price based on predefined rules.

- Locks payment in escrow until delivery confirmation.

**Step 3: Consumer Scans QR Code**

- Retrieves blockchain data → Shows full journey (farm → distributor → retailer).

5. **Expected Outcomes**

✔ Transparency: Consumers verify produce origin.

✔ Fair Pricing: Farmers receive automated, fair payments.

✔ Efficiency: Smart contracts reduce manual paperwork.

✔ Trust: Immutable blockchain records prevent fraud.

6. **Future Enhancements (Post-MVP)**

- IoT Sensors: Real-time temperature monitoring.

- AI Grading: Automated quality assessment.

- DeFi Integration: Farmers access loans against future produce.