# Kisan Yantra: Detailed Project Requirements Document

## 1. Introduction and Project Scope

The primary goal of the Kisan Yantra platform is to digitize and optimize the agricultural equipment rental ecosystem. This platform will connect **Equipment Owners** (FPOs, CHCs) with **Renters** (Farmers) via a streamlined, intelligent system, reducing equipment idle time and maximizing farm mechanization efficiency.

### 1.1 Scope of Work

The project encompasses the development of a single, scalable web application instance leveraging Python/Django (with PostGIS extensions) and integrating the Gemini API for natural language processing and intelligent booking.

### 1.2 Architectural Foundation: Multi-Tenancy Model (CRITICAL)

To manage multiple distinct entities (FPOs/CHCs) on a single platform instance while ensuring data security, the architecture will employ the **Shared Database, Row-Level Isolation** model.

* **Requirement:** Every business-critical data model (e.g., Equipment, Booking, FinancialReport) **MUST** include a mandatory Foreign Key reference to the OwnerOrganization (Tenant) model.
* **Security:** All Owner-facing interfaces (Web App) must strictly filter data using the current user’s Organization ID, preventing data leakage between tenants.
* **Search Functionality:** The Renter-facing AI search will be the *only* functional area permitted to perform global, cross-tenant queries on the Equipment model to find the nearest and best match.

## 2. User Roles and Access

| **Role** | **Access Interface** | **Primary Goal** | **Architectural Note** |
| --- | --- | --- | --- |
| **Renter (Farmer)** | WhatsApp/AI Chat Interface (Primary) | Easily find, compare, and book the nearest, best-suited equipment. | Anonymous or light user profiles only; primary interaction is through the AI service (Phase 3). |
| **Owner (FPO/CHC Staff)** | Web/Mobile Management Application | Manage equipment fleet, track availability, confirm bookings, generate financial reports. | Strictly bound by Row-Level Tenancy (sees only their organization's data). |
| **System Admin** | Django Administration Interface | System configuration, tenant onboarding/management, auditing, and global platform oversight. | Exempt from multi-tenancy filtering for auditing purposes. |

## 3. Functional Requirements (FR)

Requirements are grouped by the core pitch phases.

### Phase 0: Minimum Viable Product (MVP) and Core Platform

| **ID** | **Requirement Description** | **Role(s)** | **Dependencies** |
| --- | --- | --- | --- |
| **FR 0.1** | **Tenant Onboarding & User Linking:** Allow System Admin to create a new OwnerOrganization (FPO/CHC) and link Owner user accounts to it. | Admin, Owner | OwnerOrganization, OwnerProfile Models. |
| **FR 0.2** | **Equipment Catalog Management:** Owners must be able to add, edit, and delete equipment records, including: Asset ID, Name, Description, Rate/Hour, and operational **Geo-tagged Location** (using PostGIS). | Owner | Equipment Model. |
| **FR 0.3** | **Basic Renter Booking Request:** Implement an API endpoint that allows a Renter (via any interface) to submit a preliminary booking request (Equipment ID, Renter Name/Contact, Start/End Time). | Renter, System | Booking Model. |
| **FR 0.4** | **Owner Booking Confirmation Panel:** Provide an Owner dashboard to view all PENDING bookings for **their** organization and change the status to CONFIRMED or CANCELLED. | Owner | Multi-Tenancy Filter. |
| **FR 0.5** | **Renter Notifications:** Send basic SMS/WhatsApp notifications to the Renter when the booking status changes (CONFIRMED, CANCELLED). | System | External SMS/WhatsApp API (TBD). |

### Phase 1: Automated Dispatch, Inventory, and Geospatial Intelligence

| **ID** | **Requirement Description** | **Role(s)** | **Dependencies** |
| --- | --- | --- | --- |
| **FR 1.1** | **Real-Time Availability Engine:** Implement a function to calculate the precise availability of a machine based on existing confirmed bookings, maintenance schedules, and current operational status. | System | Booking, Maintenance Models. |
| **FR 1.2** | **Geo-Spatial Search API:** Implement a PostGIS-enabled API to efficiently query all available equipment and return a ranked list based on the Renter’s provided location (distance). | System | PostGIS extension. |
| **FR 1.3** | **Equipment Maintenance Logging:** Owners must be able to log maintenance events (scheduled and unscheduled). The system must automatically update Equipment.is\_available status during the maintenance period. | Owner | Maintenance Model. |
| **FR 1.4** | **Utilization Tracking:** Calculate and display the utilization rate (hours booked vs. available hours) for each equipment item and the entire organization's fleet. | Owner, System | Data Aggregation. |

### Phase 2: Payment and Financialization

| **ID** | **Requirement Description** | **Role(s)** | **Dependencies** |
| --- | --- | --- | --- |
| **FR 2.1** | **Digital Payment Gateway Integration:** Securely integrate a payment gateway (e.g., Stripe, local Indian gateways) to handle booking deposits and final payments. | System | PCI Compliance (NFR 4). |
| **FR 2.2** | **Owner Financial Reporting (Tenant-Specific):** Owners must be able to generate and export financial reports (e.g., Monthly Revenue, Transaction History, Outstanding Payments) strictly filtered by their OwnerOrganization ID. | Owner | Multi-Tenancy Filter. |
| **FR 2.3** | **Credit Scoring Data Logging:** The system must log all successful/failed bookings, payment timeliness, and Renter behavior in a structured format suitable for consumption by an external credit scoring or financing partner API (Phase 4). | System | **FR 2.2**, Data Warehouse (TBD). |

### Phase 3: The AI Engine (Kisan Yantra Core)

| **ID** | **Requirement Description** | **Role(s)** | **Dependencies** |
| --- | --- | --- | --- |
| **FR 3.1** | **AI Natural Language Processing (LLM Service):** Implement a service that uses the **Gemini API** to receive unstructured, multilingual Renter input (e.g., "Need a tractor for plowing near Village X on Tuesday") and translate it into a structured, executable JSON query (Equipment Type, Dates, Geo-coordinates). | Renter, System | Gemini API integration. |
| **FR 3.2** | **Automated Best Match & Quote:** The System must take the structured query (from FR 3.1), execute the Geo-Spatial Search (FR 1.2), check availability (FR 1.1), calculate the total estimated cost, and return the single best option as a conversational response. | System | **FR 1.1, FR 1.2**. |
| **FR 3.3** | **Multilingual AI Interface:** The AI chat system must support seamless communication in major regional languages (e.g., Hindi, Marathi, Telugu) for both input and response generation. | Renter | Gemini API's multilingual capability. |
| **FR 3.4** | **Conversational Booking Flow:** The AI must guide the Renter through the booking process, handle clarifications (e.g., "Do you need a heavy or light tractor?"), and confirm the final booking request. | Renter, System | **FR 0.3**. |

## 4. Non-Functional Requirements (NFR)

| **ID** | **Category** | **Requirement Description** | **Standard** |
| --- | --- | --- | --- |
| **NFR 1** | **Performance** | All critical API endpoints (e.g., Geo-Spatial Search, Availability Check) must respond to the Renter in under **1.0 second**. | P95 Latency < 1000ms. |
| **NFR 2** | **Security & Isolation** | Strict **Row-Level Isolation** must be enforced for all Owner data. Unauthorized cross-tenant access must be programmatically impossible. | Zero Data Leakage. |
| **NFR 3** | **Scalability** | The platform must be horizontally scalable to support 100+ active OwnerOrganizations and 10,000+ simultaneous Renter user interactions during peak season. | Cloud Native (AWS/GCP/Azure). |
| **NFR 4** | **Availability** | The core AI Booking Engine and Equipment Catalog API must maintain **99.9% uptime**. | Max 8.7 hours of downtime per year. |
| **NFR 5** | **Usability** | The Owner Web/Mobile Management Application must have an intuitive interface requiring minimal training time (less than 30 minutes). | High SUS Score (System Usability Scale). |
| **NFR 6** | **Disaster Recovery** | All production data, including PostGIS and financial records, must be backed up daily with a maximum Recovery Point Objective (RPO) of 24 hours. | RPO <= 24 Hours. |