Software Requirements Specification

for

AgriTracker

Version 1.0

Prepared by : Neeraj Ku. Kannoujiya
Aman Sahu

Date : 06/04/2025

Approved by : Dr. Gaurav Srivastava

Table of Contents

Revision History

Version	Date	Author	Description of Changes	Approved By
1.0	2025-04-06	Neeraj & Aman	Initial draft	-
				-
				-
				-

1. Introduction

1.1 Purpose

This document provides a comprehensive specification of requirements for **AgriTracker**, a web-based agricultural management and marketplace platform designed for Large/Mid-scale farmers in India. The purpose is to:

- Define **functional** and **non-functional requirements**
- Serve as a **reference** for developers, testers, and stakeholders
- Establish **project scope** and **objectives**
- Ensure alignment with **IEEE 29148-2018 standards** for SRS documentation

1.2 Document Conventions

- **Bold**: Key terms and definitions
- *Italics*: Emphasis on critical requirements
- Monospace: Technical references and code elements
- **Acronyms**:
    - SRS: Software Requirements Specification
    - UI: User Interface
    - API: Application Programming Interface

1.3 Intended Audience

Role	Responsibility	Usage
Developers	Implementation	System architecture, feature development
Testers	Quality Assurance	Validation against requirements
Project Managers	Oversight	Timeline and resource planning
Farmers (End Users)	Primary users	Understand system capabilities
Agricultural Organizations	Stakeholders	Evaluate impact and benefits

1.4 Background and Problem Statement

1.4.1 Background

Small-scale farmers in India face critical challenges:

- **Data Management**: Reliance on manual record-keeping limits ability to track crop yields, resource usage, and financial performance.
- **Market Access**: Dependence on middlemen reduces profit margins and creates information asymmetry.

AgriTracker addresses these issues by providing:

- **Data Management Tools**: Digital recording of farm operations (crop cycles, irrigation, inputs).
- **Direct Marketplace**: Platform for farmers to list produce and connect directly with buyers.
- **Blog Functionality:** Allows farmers to share their experiences, profits/losses, and best practices (such as effective pesticides/insecticides) with the farming community.

1.4.2 Problem Statement

The core problems are:

- **Data Inaccessibility**:
  - No centralized system for farm data analysis.
  - Manual methods hinder performance tracking.
- **Market Inefficiency**:
  - Middlemen dominate supply chains, reducing farmer profits.
  - Buyers lack transparency in pricing and produce availability.

1.5 Objectives

Objective	Description	Success Metric
Data-Driven Decisions	Enable farmers to record and analyze crop/resource data	80% adoption rate among pilot users
Market Linkage	Facilitate direct farmer-buyer transactions	50% reduction in middlemen dependency
Sustainability	Promote resource optimization	30% improvement in water/fertilizer efficiency

1.6 Scope

1.6.1 In Scope

- **Farm Management Module**:
    - Crop cycle tracking
    - Resource (water, fertilizer) logging
    - Report generation (PDF/Excel)
- **Marketplace Module**:
    - Product listings (type, quantity, price)
    - Search/filter functionality for buyers
    - Communication tools (WhatsApp/phone integration)

1.6.2 Out of Scope

- Online payment processing (Phase 2)
- Advanced predictive analytics (AI/ML)

1.7 Methodology

AgriTracker follows **Agile development** with these phases:

1. **Requirement Gathering**: Farmer interviews (n=100+ across 5 states).
2. **Prototyping**: Low-fidelity UI mockups validated with users.
3. **Development**:
    - Frontend: Django-based responsive design
    - Backend: Python (Django REST Framework)
    - Database: PostgreSQL with encryption
4. **Testing**:
    - Unit testing (PyTest)
    - User acceptance testing (UAT) with farmers
5. **Deployment**: Gradual rollout to 500+ farms in Phase 1.

1.8 References

1.8.1 Frameworks

❖ **Django Framework:**

- Django Software Foundation. (2024). *Django: The Web Framework for Perfectionists with Deadlines.* Retrieved from https://www.djangoproject.com/

❖ **Bootstrap:**

- Bootstrap. (2024). *Bootstrap: The World's Most Popular Framework for Building Responsive, Mobile-First Sites.* Retrieved from https://getbootstrap.com/

❖ **HTML, CSS, and JavaScript:**

- W3C. (2024). *HTML & CSS Standards.* Retrieved from https://www.w3.org/standards/webdesign/htmlcss

❖ **Database (SQLite):**

- SQLite Consortium. (2024). *SQLite: Small. Fast. Reliable. Choose Any Three.* Retrieved from https://www.sqlite.org/index.html

1.8.2 Error Solving and Debugging

❖ **General Django Errors:**

- Common Issues: Common Django Issues and Fixes
- Stack Overflow: Django Tag on Stack Overflow

❖ **Bootstrap, HTML, CSS, JavaScript Errors:**

- W3Schools: HTML, CSS, and JavaScript Validation
- Stack Overflow: Bootstrap Tag on Stack Overflow

❖ **SQLite Errors:**

- SQLite FAQ: SQLite Frequently Asked Questions
- Stack Overflow: SQLite Tag on Stack Overflow