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	-
1.1	2025-04-15	Neeraj & Aman	Creating Introduction page	-
1.2	2025-04-17	Neeraj & Aman	Adding Methodology	-
2.0	2025-04-25	Neeraj & Aman	Adding Overall Description	-

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

2. Overall Description

This section provides context and background for AgriTracker, outlining its key aspects, stakeholders, and the environment in which it will operate.

2.1 Product Perspective

AgriTracker is a standalone web-based platform built on Django REST Framework (backend) and a responsive front-end (HTML/CSS/Bootstrap) to support large and mid-scale farmers across India. It integrates three primary modules—Farm Management, Marketplace, and Community Blog—each communicating over secure RESTful APIs with a PostgreSQL database. External dependencies include WhatsApp Business API for messaging, SMTP for email notifications, and PDF/Excel reporting libraries. AgriTracker will be deployable on cloud infrastructure (e.g., AWS/GCP) or on-premises servers.

2.2 Product Functions

- **Farm Management Module:**
    - Record and track crop cycles, irrigation schedules, fertilizer and pesticide usage.
    - Generate performance reports (PDF/Excel) and interactive visualizations of yield vs. inputs.
    - Manage resource inventories and historical data analytics.
- **Marketplace Module:**
    - Create and manage crop listings with attributes (type, quantity, quality grade, harvest date, price).
    - Advanced search and filter for buyers by location, crop type, price range, and availability.
    - In-app messaging integration via WhatsApp and phone, plus buyer-seller negotiation logs.
- **Community Blog Module:**
    - Publish user-generated articles on best practices, profit/loss case studies, and crop management tips.
    - Comment and rating features for community feedback and knowledge sharing.

2.3 User Classes and Characteristics

- **Administrators:** Manage user accounts, moderate content, oversee system health and analytics; technical proficiency: high.
- **Farmers (Sellers):** Primary users responsible for data entry, listing produce, and reading blog content; technical proficiency: low to moderate.
- **Buyers:** Search, filter, and purchase listings; moderate web literacy; authenticated via email/phone.
- **Support/Analysts:** Generate aggregate reports, respond to user queries, and maintain knowledge base; technical proficiency: moderate.

2.4 Operating Environment

- **Server:** Linux-based VM or container with Docker, minimum 4 vCPU, 8 GB RAM, 100 GB storage.
- **Client:** Modern browsers (Chrome, Firefox, Edge, Safari) on desktops, tablets, and smartphones; responsive for 320×568 to 1920×1080 resolutions.
- **Network:** Internet connectivity with minimum 512 Kbps bandwidth; secure HTTPS (TLS 1.2+).

2.5 Design and Implementation Constraints

- Must use Django 4.x and Django REST Framework for backend.
- PostgreSQL 12+ as the primary datastore; data encryption at rest.
- Front-end implemented with Bootstrap 5; comply with WCAG 2.1 AA accessibility standards.
- Integration limited to WhatsApp Business API, SMTP, and PDF/Excel libraries with non-commercial licenses.
- All code developed in Python 3.10+; adhere to PEP 8 and use pytest for TDD.

2.6 ARCHITECTURE DESIGN

2.6.1 Data Flow Diagram

A DFD is a Pictorial representation used to visualize the flow of data for a system. It illustrates how data moves through various processes, data stores, and external entities. DFDs help in understanding the functional aspects of a system by mapping out data inputs, outputs, processing steps, and storage points. They are use in systems analysis and design to provide a clear and structured depiction of how information is handle. By breaking down complex processes into simpler components, DFDs aid in identifying inefficiencies, redundancies, and potential improvements in the data handling process.

2.6.2 Notations Used

Notations in the Data Flow Diagram (DFD) include processes, data stores, data flows, and external entities, representing system functions, storage, data movement, and user interactions respectively.

There are four basic symbols used to construct data flow diagram:-

A rectangle represents the source and destination.

It represents the flow of data.

Circle or oval representing a system function.

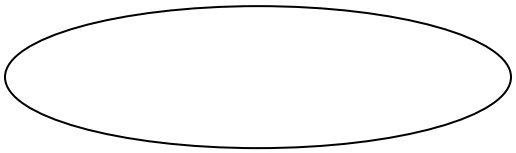


Fig 2.6.1 Notations Use

2.6.3 Data Flow Diagram of Seller

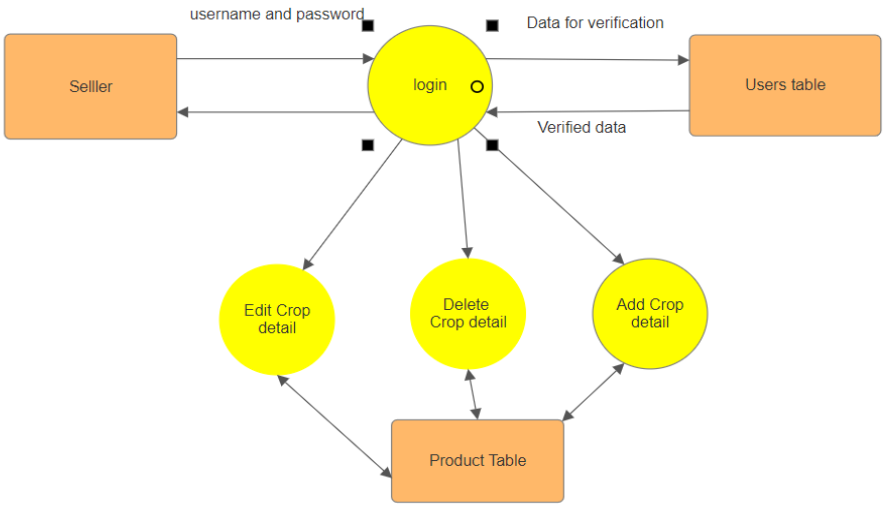


Fig 2.6.2 DFD for Seller

2.6.4 Data Flow Diagram for Buyer

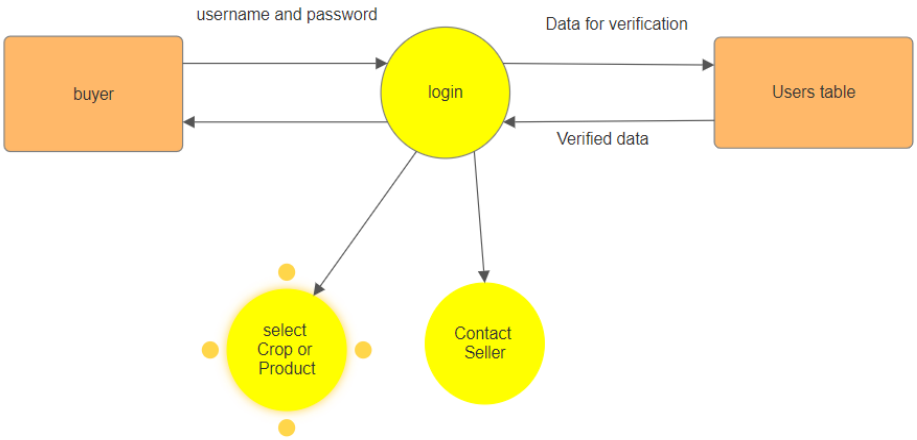


Fig: 2.6.3 Data Flow Diagram for Buyer

2.6.5 Sequence Diagram for Verification

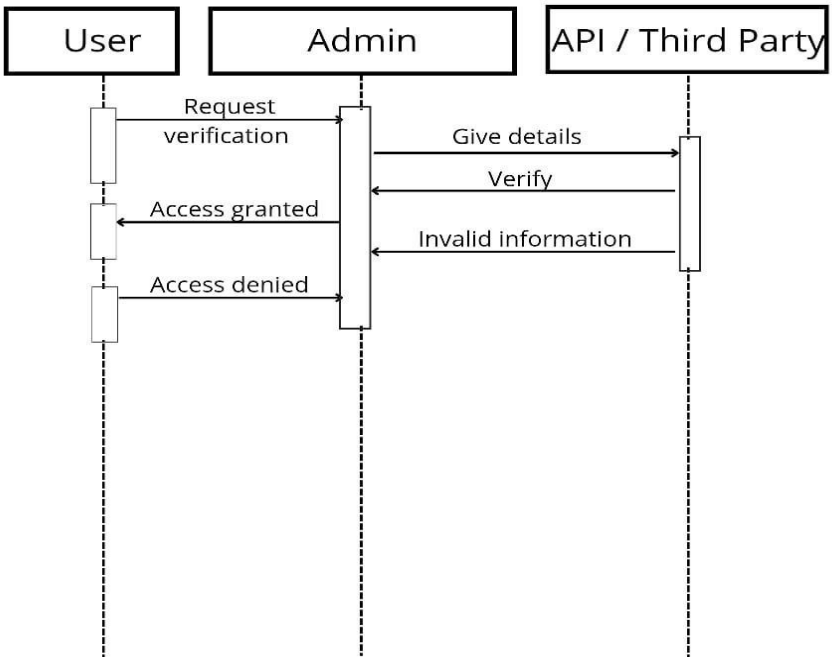
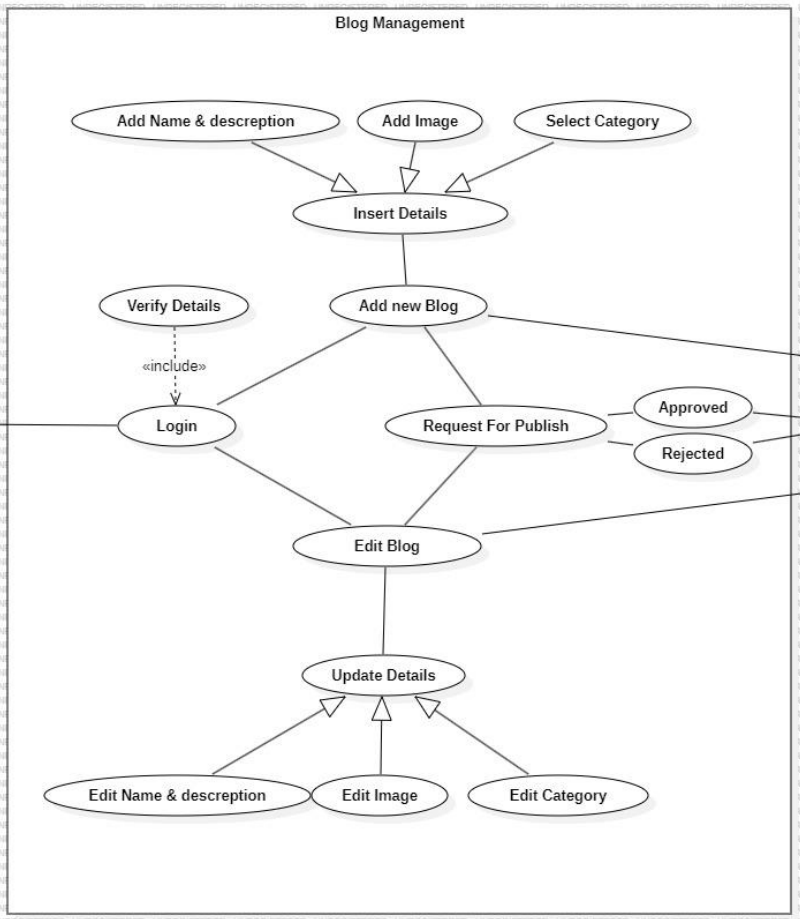


Fig 2.6.4 Sequence Diagram for Verification

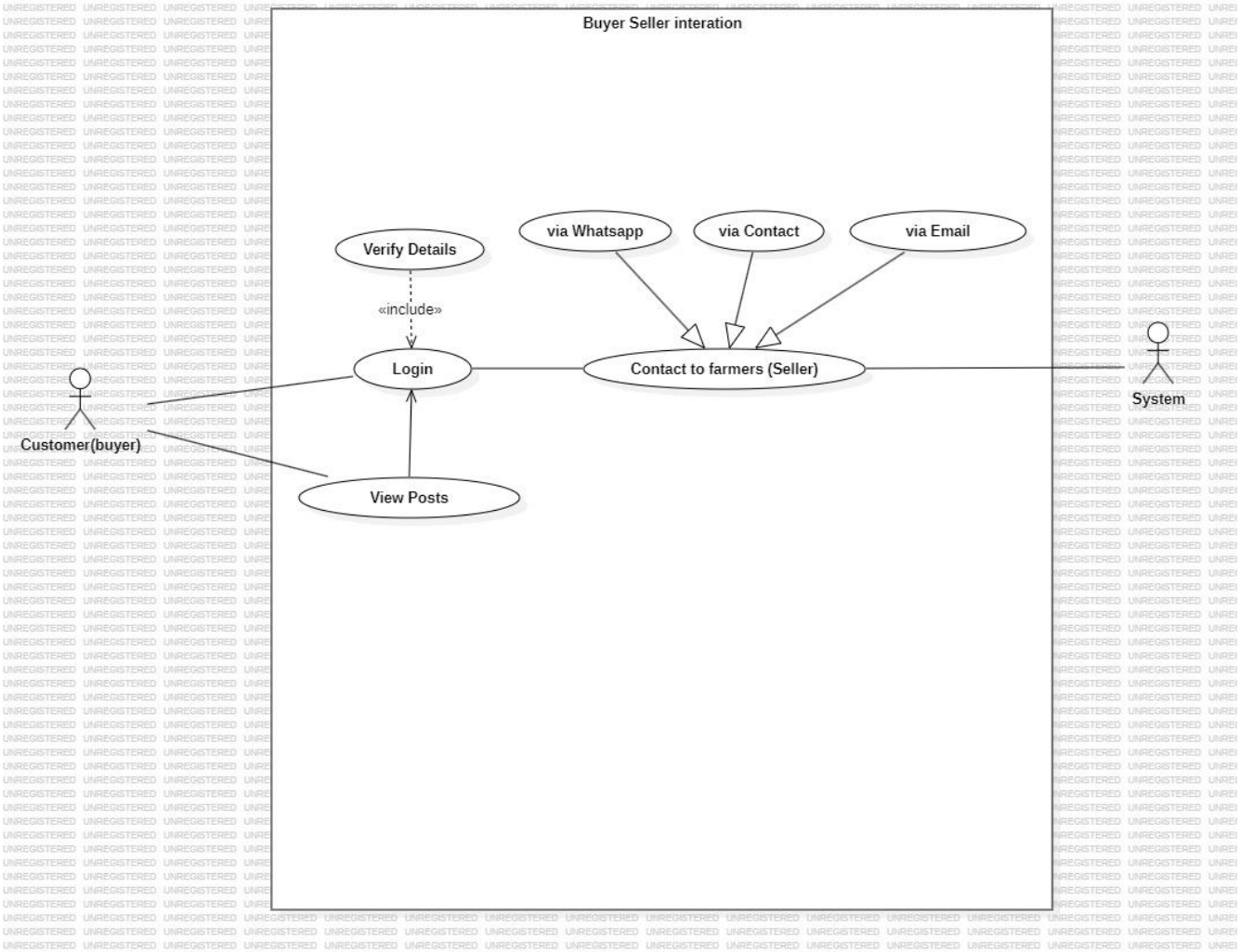
2.6.6 Blog Management Use case Diagram:

- Users can add, edit, and update blogs with details like name, description, image, and category.
- Blogs need verification before publishing, and they can be approved or rejected.
- Includes login and a "Request for Publish" step.



2.6.7 Buyer-Seller Interaction:

- Buyers can verify details and log in to interact with sellers.
- Communication happens via WhatsApp, email, or by visiting the system.
- Buyers can view posts shared by sellers (farmers).



2.6.8 Login and Registration Process:

Send OTP via Email" is the core use case, triggered when users need to verify their identity.

When OTP is Used:

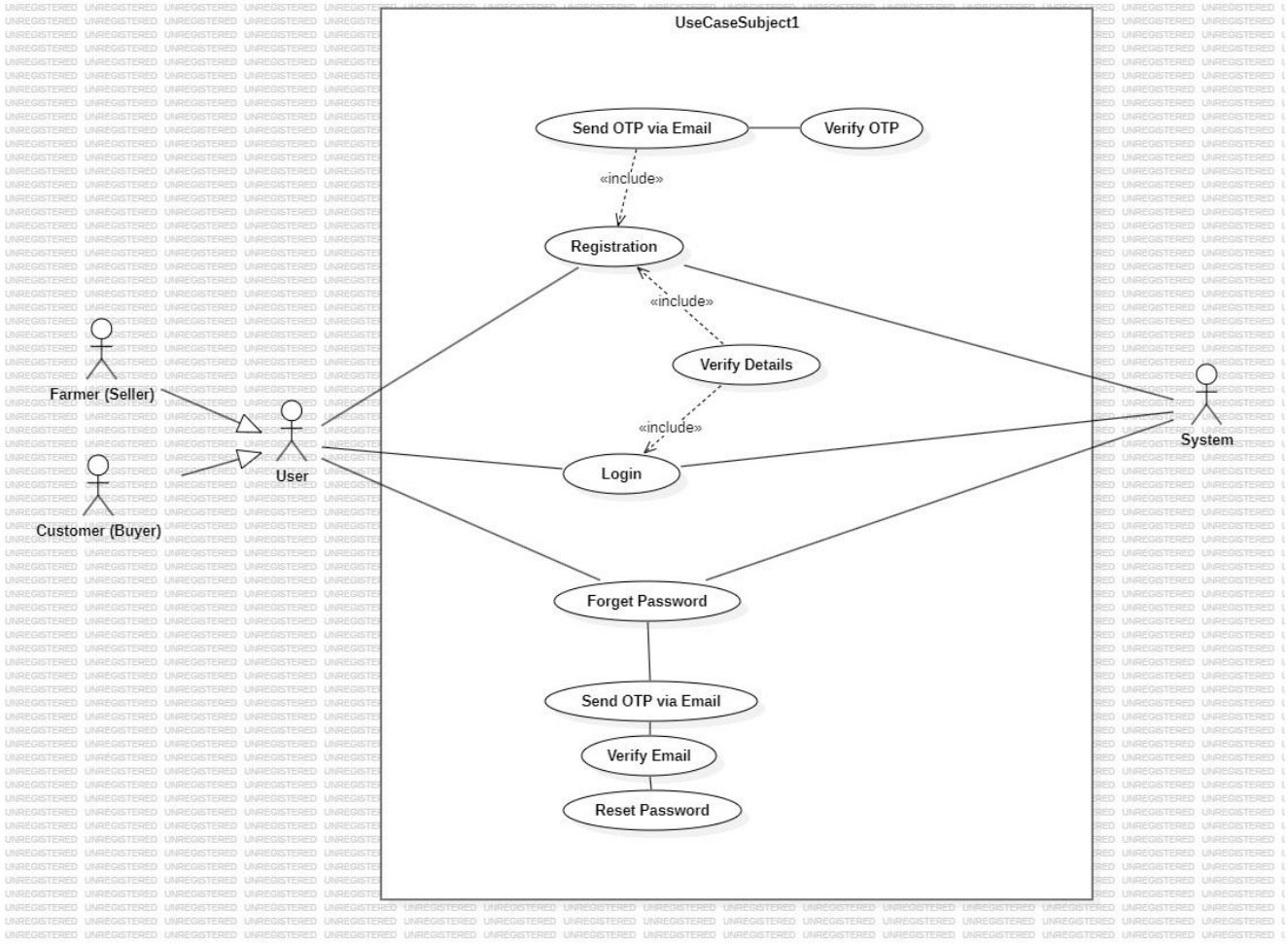
- **Registration**: New users verify their email during sign-up.
- **Login**: Secure access by confirming OTP (if enabled).
- **Forget Password**: Users request an OTP to reset their password.

Steps Involved:

Send OTP → Verify OTP → (Proceed to reset password, register, or log in).

Who Uses It:

Farmers (Sellers) and **Customers** (Buyers)—both roles rely on OTP for secure authentication



.2.7 User Documentation

- **User Manual:** Step-by-step guide covering account setup, module usage, report generation, and marketplace workflows.
- **Online Help:** Contextual tooltips, FAQ section, and search-enabled knowledge base.
- **Tutorials:** Short video walkthroughs for mobile and web interfaces; hosted on YouTube or embedded player.
- **Release Notes:** Versioned change logs and known issues for each deployment.

2.8 Assumptions and Dependencies

- Farmers and buyers have basic literacy in English or Hindi and access to a smartphone or computer.
- Reliable internet access is available in target agricultural regions.
- WhatsApp Business API and SMTP services remain operational and within usage quotas.
- No real-time payment gateway integration in Phase 1.

3. External Interface Requirements

This section describes all inputs and outputs between AgriTracker and external entities.

3.1 User Interfaces

- **Dashboard Screens:** Web UI with navigation for Farm Management, Marketplace, and Blog.
- **Forms:** Responsive forms for data entry (crop details, resource logs) with client-side validation.
- **Reports:** Interactive charts (yield vs. time) and exportable tables; filter panels.

3.2 Hardware Interfaces

- **Storage Devices:** Interaction with server-mounted disks or cloud block storage for database and file uploads (images, reports).
- **Mobile Device:** Camera access via HTML5 for uploading crop images (optional).

3.3 Software Interfaces

- **Database:** PostgreSQL accessed via Django ORM.
- **Messaging API:** RESTful integration with WhatsApp Business API for notifications and chat.
- **Email/SMS:** SMTP or third-party service (e.g., Twilio) for alerts and password reset.

3.4 Communications Interfaces

- **Protocols:** HTTPS for all client-server traffic, TLS 1.2+ encryption.
- **API Endpoints:** RESTful JSON over HTTP; paginated GET requests for listings and logs.
- **Webhooks:** Receive asynchronous updates from WhatsApp API for message status.

4. System Features

Each feature described below includes a brief overview, inputs, processing, and outputs.

4.1 Farm Management Module

- **Overview:** Enables farmers to record crop cycles, resource usage, and view historical analytics.
- **Inputs:** Crop type, planting/harvest dates, resource quantities, weather conditions.
- **Processing:** Validation, storage in time-series tables, aggregation for reports.
- **Outputs:** PDF/Excel reports, interactive time-series graphs, inventory alerts.

4.2 Marketplace Module

- **Overview:** Allows sellers to list produce and buyers to search and negotiate.
- **Inputs:** Listing details (crop attributes, price), buyer search criteria, messages.
- **Processing:** Indexing listings, filtering by geolocation and price, forwarding messages via API.
- **Outputs:** Search results, chat threads, transaction logs.

4.3 Community Blog Module

- **Overview:** Facilitates knowledge sharing among users.
- **Inputs:** Blog posts (text, images), comments, ratings.
- **Processing:** Content moderation workflow, metadata tagging, ranking by popularity.
- **Outputs:** Rendered article pages, comment threads, trending topics list.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Page load time < 2 seconds under typical conditions (100 concurrent users).
- API response time < 300 ms for simple GET requests; < 500 ms for report generation.
- Support up to 1,000 concurrent active sessions in Phase 1.

5.2 Safety Requirements

- Automated daily backups of database and user-uploaded files; retention for 30 days.
- Graceful degradation: queued writes if database is temporarily unavailable, with user notification.

5.3 Security Requirements

- Role-based access control enforced at API and UI layers.
- Passwords hashed with bcrypt; TLS encryption for data in transit.
- OWASP Top 10 mitigations (SQL injection, XSS, CSRF protections).

5.4 Software Quality Attributes

- **Usability:** Intuitive UI with localization support (English/Hindi).
- **Reliability:** 99.5% uptime SLA; alerting on failures with average MTTR < 2 hours.
- **Maintainability:** Modular codebase with 80% unit test coverage.
- **Portability:** Docker containers supporting cloud or on-prem deployments.

5.5 Business Rules

- Sellers pay a 2% commission per transaction (Phase 1).
- Listings expire automatically after the specified harvest date + 30 days.
- Blog posts require Admin approval before publication.
- Farmers must confirm inventory accuracy weekly to maintain listing priority.