LOVELY PROFESSIONAL UNIVERSITY

Farmers E-commerce Website
Software Requirements Specification

INT219
FRONT-END WEB DEVELOPER

Prepared By: Ram Kumar (12306472)
Tanya Yadav (12317972)
Sahil Pathan (12319927)

Prepared for
Continuous Assessment 2
Spring 2025

---

1. Introduction

1.1 Purpose:

The purpose of this document is to define the software requirements for the frontend of the **FarmerSupply** web application. It will serve as a guide for the development team to design and implement an intuitive and responsive user interface for all users, including customers, vendors, and administrators.

1.2 Scope:

The frontend system will be responsible for:

- User registration and login
- Displaying farming products
- Cart and checkout UI
- Admin dashboard interface
- Contact form
- Responsive design for mobile/tablet devices

1.3 Definitions, Acronyms, and Abbreviations:

- **UI** – User Interface
- **UX** – User Experience
- **HTML** – Hyper Text Markup Language
- **CSS** – Cascading Style Sheets
- **JS** – JavaScript
- **SPA** – Single Page Application

1.4 References:

- W3C standards for HTML/CSS
- Tailwind CSS– UI component library used for responsive design.

1.5 Overview:

This document outlines the requirements needed for frontend development, ensuring seamless communication with backend APIs, user-friendly layouts, and efficient workflows for both customers and administrators. Proper attention has been paid to usability, performance, and security, aligning with industry best practices for web applications.

2 General Description

2.1 Product Perspective:

The frontend acts as the visual and interaction layer for the FarmerSupply system, consuming REST APIs provided by the backend. It will be built using HTML, CSS, JavaScript.

2.2 Product Functions:

The frontend provides the following core functionalities:

2.2.1 Product Management:

- Vendors can add/edit/delete their products
- Admins can view and manage all products
- Product search and filtering capabilities

2.2.2 Cart and Checkout:

- Add, update, remove, and view cart items
- Session-based cart persistence
- Order placement and confirmation

2.2.3 Order Management:

- Store order details in the database
- Order tracking and status updates
- Admin and vendor access to order history

2.2.4 Contact Form Handling:

- Capture user queries or feedback
- Sanitize and validate user-submitted data
- Send emails or store messages in the database

2.2.5 Admin Dashboard:

- Manage users and vendors
- View reports and analytics
- System-level control and maintenance

2.2 User Characteristics:

There are three main types of users interacting with the backend system:

2.3.1 Farmers (Customers):

- Limited technical expertise
- Use the platform to browse, add products to cart, and place orders

2.3.2 Vendors:

- Intermediate technical expertise
- Use the platform to manage their inventory, view orders, and fulfill them

2.3.3 Administrators:

- Technically proficient
- Manage all users, monitor transactions, oversee platform health

2.3 General Constraints:

- Must be fully functional on all major modern browsers: Chrome, Firefox, Edge, Safari.
- Design should be responsive for optimal viewing on desktops, tablets, and mobile devices.
- Frontend must consume RESTful APIs provided by the backend for all dynamic data operations.
- The application should conform to accessibility standards as far as possible.

2.5 Assumptions and Dependencies:

- The frontend will be developed using HTML/CSS/JS and will consume backend APIs via HTTP.
- Internet access is available to connect to email services and remote deployments.
- Developers have access to a hosting environment that supports PHP and MySQL.
- Any future integration with payment gateways or mobile apps will reuse the existing API structure.

3 Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Authentication Pages:

- Login and Register with validation and feedback.
- Show/hide menus and content based on user login status and role.

3.1.1.2 Product Pages:

- Homepage with product categories and featured items.
- Product listing with sorting (price, popularity) and filtering (category, price range).
- Product details with image, description, price, and "Add to Cart" button.

3.1.1.3 Cart and Checkout:

- Cart with item summary, total price, and quantity adjustment.
- Checkout page with address and contact form.

3.1.1.4 Admin Dashboard:

- Interface to create, read, update, and delete (CRUD) products.
- View user and order lists.
- Status indicators for orders and stock levels.

3.1.1.5 Contact Us Form:

- A form with fields: Name, Email, Message.
- Form submission sends a POST request to the backend API.

3.1.2 Hardware Interfaces

- Keyboard, mouse, and display for input/output

- No direct hardware dependencies. The system is designed to run on standard server hardware (or localhost).

3.1.3 Software Interfaces

- **Database**: MySQL 5.7+ for data storage.
- **Frontend communicates with backend** APIs using: JSON over HTTP/HTTPs
- **Server**: Apache (via XAMPP/WAMP).
- **PHP**: PHP 7.4+ for server-side scripting.
- **Mail**: Uses PHP mail() for sending contact emails.

3.1.4 Communications Interfaces

- Uses HTTP/HTTPS for all client-server communication.
- API endpoints consume and return JSON.
- May optionally support CORS for frontend requests from other domains.

3.2 Functional Requirements

3.2.1 User Registration and Authentication

- Users shall be able to register with a name, email, and password.
- Users shall be able to log in and maintain a session.
- Passwords shall be stored in hashed format.
- Different roles shall be assigned: customer, vendor, and admin.

3.2.2 Product Browsing and Search

- Users shall be able to view a list of available products.
- Users shall be able to filter products by category or name.
- Users shall be able to click on a product to view detailed information.

3.2.3 Shopping Cart Management

- Users shall be able to add products to their shopping cart.
- Users shall be able to update quantities or remove items.
- The system shall calculate total price dynamically.
- Cart shall persist during the session.

3.2.4 Checkout and Order Processing

- Users shall be able to proceed to checkout from the cart.
- Users shall provide delivery details and confirm the order.
- The system shall record the order and clear the cart upon success.
- A confirmation message shall be displayed.

3.2.5 Order Management

- Users shall be able to view their past orders.
- Admins and vendors shall be able to view and update order statuses.
- The system shall track orders by ID, product list, user, and status.

3.2.6 Product Management

- Admins or vendors shall be able to add, update, and delete products.
- Each product shall include: name, description, price, quantity, and image.

3.2.7 Contact Form Submission

- Users shall be able to submit a contact form with name, email, and message.
- The system shall validate input and send the message to a predefined admin email.
- The user shall be notified whether the message was successfully sent.

3.2.8 Profile Management

- Users shall be able to view and update their account information.

3.3 Non-Functional Requirements

3.3.1 Performance

- Pages and API responses should load within 1 second under normal load.

3.3.2 Reliability

- The system shall operate reliably during business hours with minimal downtime.

3.3.3 Availability

- The website shall be available 24/7 except during planned maintenance.

3.3.4 Security

- Input validation and sanitization shall be enforced.
- Passwords shall be encrypted using a secure hashing algorithm.
- Access control shall be implemented based on user roles.

3.3.5 Maintainability

- The codebase shall be modular to allow updates without affecting unrelated features.

3.3.6 Portability

- The system shall be deployable on any standard LAMP stack (Linux, Apache, MySQL, PHP).

3.4 Design Constraints

- Must use LAMP stack
- Adherence to HTML5/CSS3 standards
- No use of third-party authentication libraries
- The system must use open-source technologies.
- The system must support PHP version 7.4 or higher.
- MySQL should be used as the relational database.

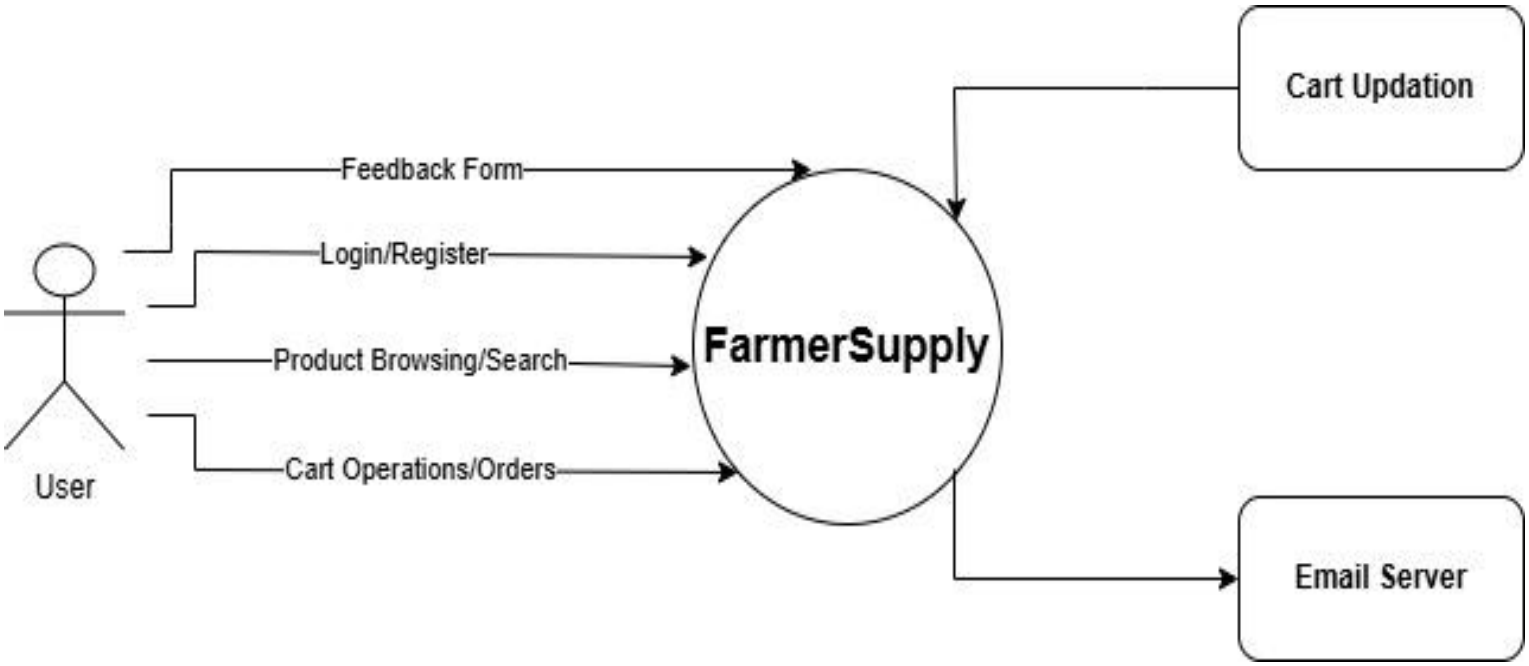
3.5 Other Requirements

- Scheduled database backups every 24 hours
- Audit log of user activity for admin review.
- The system must have a responsive design for mobile access.
- Logging shall be implemented for debugging and error handling.

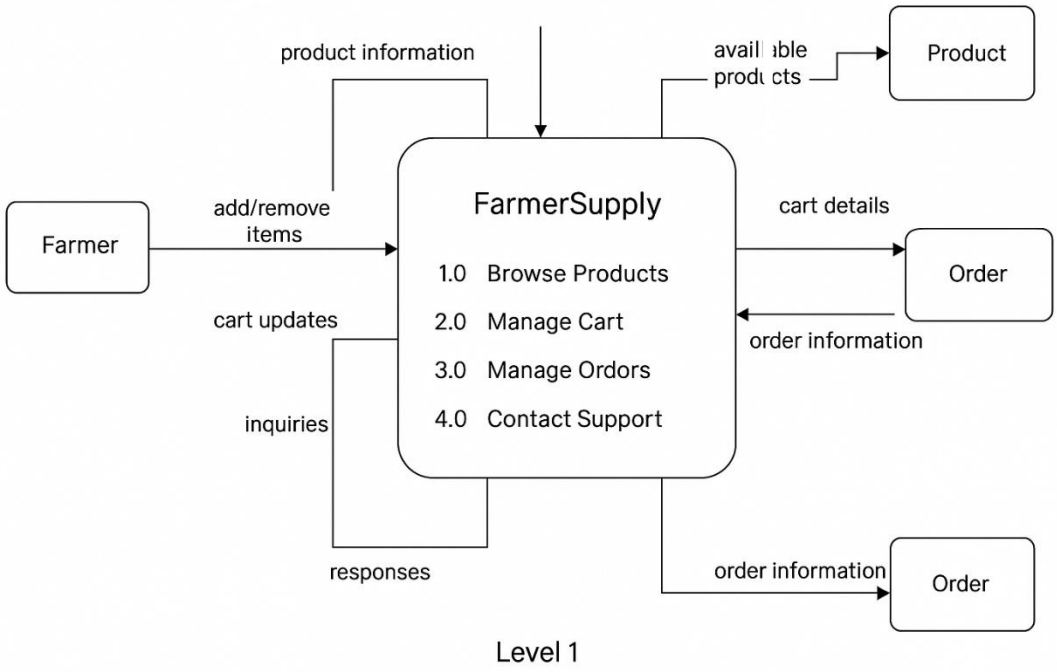
4 Analysis Models

4.1 Data Flow Diagrams (DFD)

Level 0 DFD:



Level 1 DFD: Expands CRUD operations:



Level 1

Github Link of the project: https://github.com/Ram9219/Portal-for-farmers-to-sell-the-produce-at-a-better-rate

Video Link of the project: https://www.youtube.com/watch?v=Jv1En9n5ueY