

MAKERERE UNIVERSITY

**SOFTWARE DESIGN DOCUMENT**

**Project Title:** Smart wheat Supply System (SWSS)

**Team Name:** The WheatChain Innovators

**Group Number:** G-24  
**Supervisor:** Lule Emmanuel  
**GitHub Link:** <https://github.com/Kayeerasoftware/Smart-wheat-Supply-System-SWSS-G-24>

**Group Members:**

| **Name** | **Registration Number** | **Student Number** |
| --- | --- | --- |
| Ojok Erick | 24/U/10592/EVE | 2400710592 |
| Kayeera Nathan | 24/U/25913/EVE | 2400725913 |
| Lubega Henry | 24/U/06446/PS | 2400706446 |
| Kintu Johnmark | 22/U/3245/EVE | 2200703245 |
| Moola Joseph | 24/U/06873/EVE | 2400706873 |

**TABLE OF CONTENTS**

[1. INTRODUCTION 1](#_Toc200087764)

[1.1. Purpose 1](#_Toc200087765)

[1.2. Scope 1](#_Toc200087766)

[1.3. Overview 1](#_Toc200087767)

[1.4. Reference Material 1](#_Toc200087768)

[1.5. Definitions and Acronyms 1](#_Toc200087769)

[2. SYSTEM OVERVIEW 2](#_Toc200087770)

[3. SYSTEM ARCHITECTURE 3](#_Toc200087771)

[3.1. Architectural Design 3](#_Toc200087772)

[3.2. Decomposition Description 4](#_Toc200087773)

[3.3. Design Rationale 8](#_Toc200087774)

[4. DATA DESIGN 9](#_Toc200087775)

[4.1. Data Description 9](#_Toc200087776)

[4.2. Data Dictionary /Data base 12](#_Toc200087777)

[5. COMPONENT DESIGN 19](#_Toc200087778)

[5.1. Authentication Component: 19](#_Toc200087779)

[5.2. Order Processing Component: 20](#_Toc200087780)

[5.3. Inventory Management Component: 20](#_Toc200087781)

[5.4. Vendor Validation Component: 21](#_Toc200087782)

[5.5. Machine Learning Analytics Component: 21](#_Toc200087783)

[5.6. Customer Segmentation Component: 22](#_Toc200087784)

[6. HUMAN INTERFACE DESIGN 23](#_Toc200087785)

[6.1. 6.1 Overview of User Interface 23](#_Toc200087786)

[6.2. Screen Images 24](#_Toc200087787)

[6.3. Screen Objects and Actions 25](#_Toc200087788)

[6.4. Login Screen Objects: 25](#_Toc200087789)

[6.5. Dashboard Screen Objects: 25](#_Toc200087790)

[6.6. Order Management Screen Objects: 25](#_Toc200087791)

[6.7. Actions and Interactions: 26](#_Toc200087792)

[7. MACHINE LEARNING MODEL 27](#_Toc200087793)

[8. APPENDICES 29](#_Toc200087794)

**LIST OF FIGURES**

[figure 3. 1 System Architecture Overview 3](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087889)

[figure 3. 2 Component Interaction Diagram 4](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087890)

[figure 3. 3 Use Case Diagram 5](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087891)

[figure 3. 4 Activity diagram 6](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087892)

[figure 3. 5 Sequence Diagram 7](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087893)

[Figure 6. 1 Login Interface Mock up 24](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087897)

[Figure 6. 2 Dashboard Interface Mock up 24](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087898)

[Figure 6. 3 Order Management Interface Mock up 25](file:///E:\From%20Desktop%20December\10_2024\For%20submission\Final%20Capstone_Project_Report_Group3_BSE1_EVE.docx#_Toc200087899)

**LIST OF TABLES**

[*Table 1: Database Tables Overview* 11](#_Toc200088142)

[*Table 2: ML Dataset Description* 11](#_Toc200088143)

[*Table 3: Component Functions Summary* 19](#_Toc200088144)

# INTRODUCTION

## Purpose

This software design document describes the architecture and system design of the Wheat Supply Chain Management System. The document is intended for software developers, system architects, project managers, and stakeholders involved in the development and implementation of the wheat supply chain management platform.

## Scope

The Wheat Supply Chain Management System provides a comprehensive platform for monitoring and managing the entire wheat supply chain process from raw material suppliers to retail stores. The system encompasses demand prediction through machine learning, customer segmentation, inventory management, order processing, workforce distribution, vendor validation, and stakeholder communication. The system aims to optimize production efficiency, enhance customer satisfaction, and streamline supply chain operations through automated analytics and intelligent recommendations.

## Overview

This document is organized into eight main sections covering system introduction, overview, architecture, data design, component design, human interface design, machine learning implementation, and appendices. Each section provides detailed technical specifications and design rationale for the wheat supply chain management system implementation.

## Reference Material

* IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions
* Laravel Framework Documentation
* MySQL Database Documentation
* Machine Learning best practices for supply chain management

## Definitions and Acronyms

* **SDD**: Software Design Document
* **ML**: Machine Learning
* **API**: Application Programming Interface
* **CRUD**: Create, Read, Update, Delete
* **PDF**: Portable Document Format
* **SCM**: Supply Chain Management
* **UI**: User Interface
* **DB**: Database

# SYSTEM OVERVIEW

The Wheat Supply Chain Management System is designed to provide end-to-end visibility and control over the wheat supply chain process. The system manages the complete lifecycle from wheat farming and harvesting through processing, distribution, and final retail delivery.

The order management module initiates when stakeholders need to place orders. Users log into the system, access the appropriate order form based on their role, fill in required details, and submit orders. Upon submission, a confirmation message displays to the user, and the receiving entity gets notified through the system's notification mechanism.

The inventory management module continuously monitors stock levels across all supply chain nodes. When inventory reaches predefined thresholds, the system automatically generates replenishment alerts and can initiate purchase orders based on demand forecasting algorithms.

The vendor validation module processes PDF applications from potential vendors through a Java server. The server analyzes financial stability metrics, reputation scores, and regulatory compliance data. Successful applicants automatically receive facility visit scheduling notifications.

The analytics module processes historical sales data, inventory movements, and customer behavior patterns to generate predictive insights. Machine learning algorithms analyze this data to forecast demand and segment customers for personalized recommendations.

The chat functionality enables real-time communication between supply chain participants. Suppliers, manufacturers, distributors, and retailers can communicate directly through secure messaging channels integrated into their respective dashboards.

Workforce distribution management optimizes human resource allocation across different supply centers based on demand forecasts, seasonal variations, and operational capacity requirements.

The reporting module generates automated, scheduled reports tailored to specific stakeholder needs, ensuring relevant information reaches appropriate decision-makers at optimal intervals.

# SYSTEM ARCHITECTURE

## Architectural Design

The system follows a three-tier architecture comprising presentation, application, and data layers. The presentation layer handles user interfaces through Laravel framework, the application layer processes business logic and integrates machine learning capabilities, and the data layer manages information storage through MySQL database.

The web server component receives HTTP requests from client browsers and routes them to appropriate Laravel controllers. Controllers interact with model classes that encapsulate business logic and data access patterns. The MySQL database management system stores all persistent data including user information, inventory records, orders, and analytics data.

A separate Java server handles vendor validation processes by accessing uploaded PDF documents from the filesystem, processing application data, and updating the main database with validation results. This server integrates with the main Laravel application through RESTful API endpoints.

The machine learning engine operates as a background service, periodically analyzing historical data to generate demand forecasts and customer segmentation insights. These results are stored in dedicated analytics tables for real-time access by the web application.

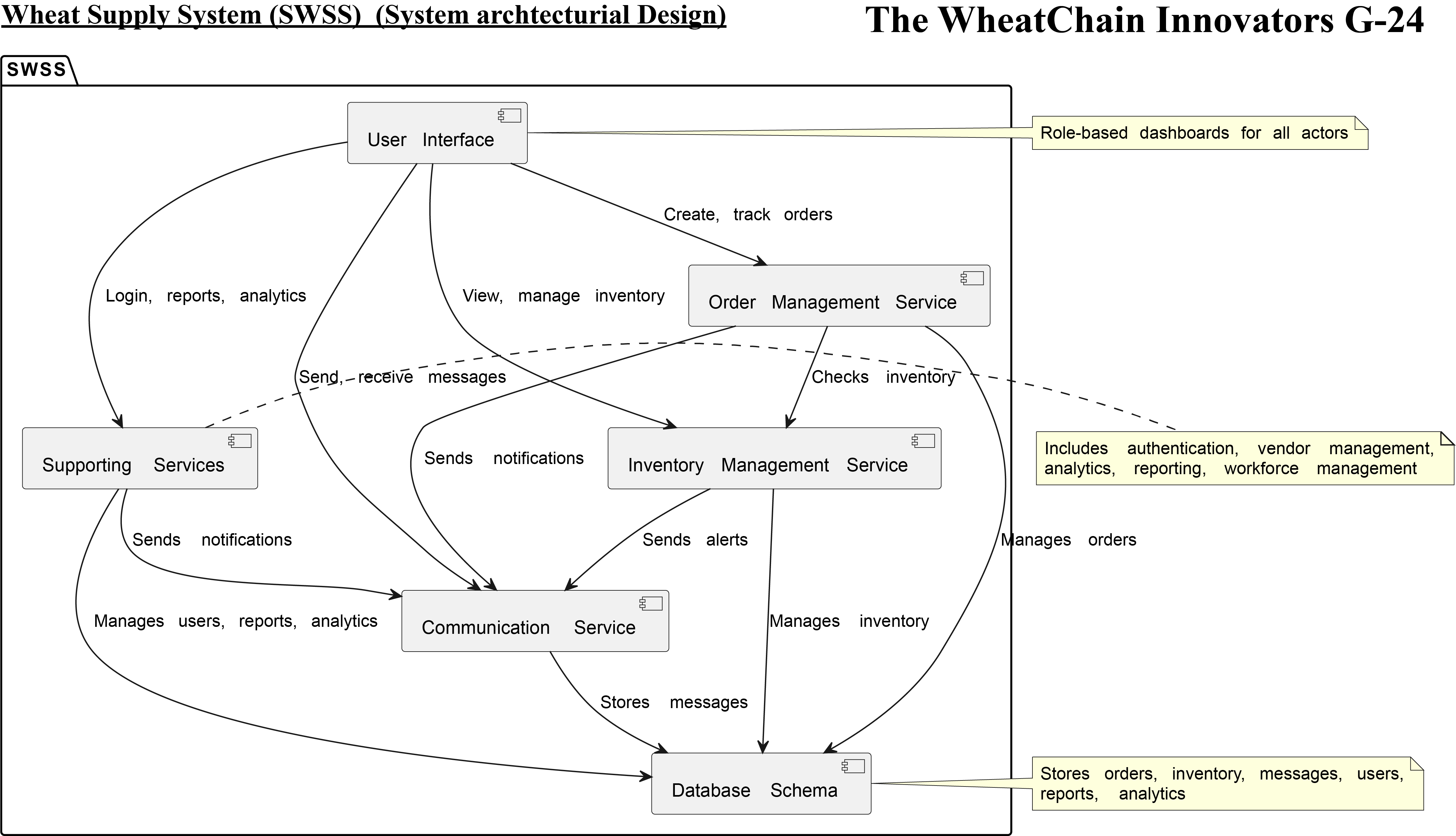


figure 3. 1 System Architecture Overview

## Decomposition Description

The system decomposes into several key subsystems that collaborate to deliver complete functionality:

**User Management Subsystem**: Handles authentication, authorization, and user profile management for different stakeholder categories including farmers, suppliers, manufacturers, distributors, and retailers.

**Order Processing Subsystem**: Manages order lifecycle from creation through fulfillment, including order validation, inventory checking, and status tracking across the supply chain.

**Inventory Management Subsystem**: Monitors stock levels, tracks inventory movements, generates replenishment alerts, and maintains inventory optimization algorithms.

**Vendor Management Subsystem**: Processes vendor applications, validates credentials, schedules facility visits, and maintains vendor performance metrics.

**Analytics Subsystem**: Implements machine learning algorithms for demand forecasting and customer segmentation, generates reports, and provides decision support insights.

**Communication Subsystem**: Facilitates chat functionality, notification delivery, and automated report distribution to stakeholders.

**Workforce Management Subsystem**: Optimizes human resource allocation, tracks productivity metrics, and manages scheduling across supply centers.

figure 3. 2 Component Interaction Diagram

User Interface

Authentication

Authorization

Order processing

Inventory Management

Analytics Engine

ML Algorithms

Reporting

Communication

Notifications

figure 3. 3 Use Case Diagram

Wheat Supply Chain Management System

**Farmer**

Manage Production

Quality Control

Tracking

Place Order

View Forecasts

Chat with Suppliers

**Supplier**

Process Orders

Manage Inventory

Validate vendors

Generate reports

Manufacturer

Manage

Quality

Batch

Retailer

POS Integration

Customer Analytics

Inventory Management

View Recommendations

System Admin

User

System

Monitor

Generate

ML Analytics

Demand Forecasting

Customer Segmentation

Price Prediction

Inventory optimization

Distributor

Route Optimization Ooptimization

Manage

Delivery scheduling

Configuration

Inventory allocation

Performance

Track Shipments

System Reports

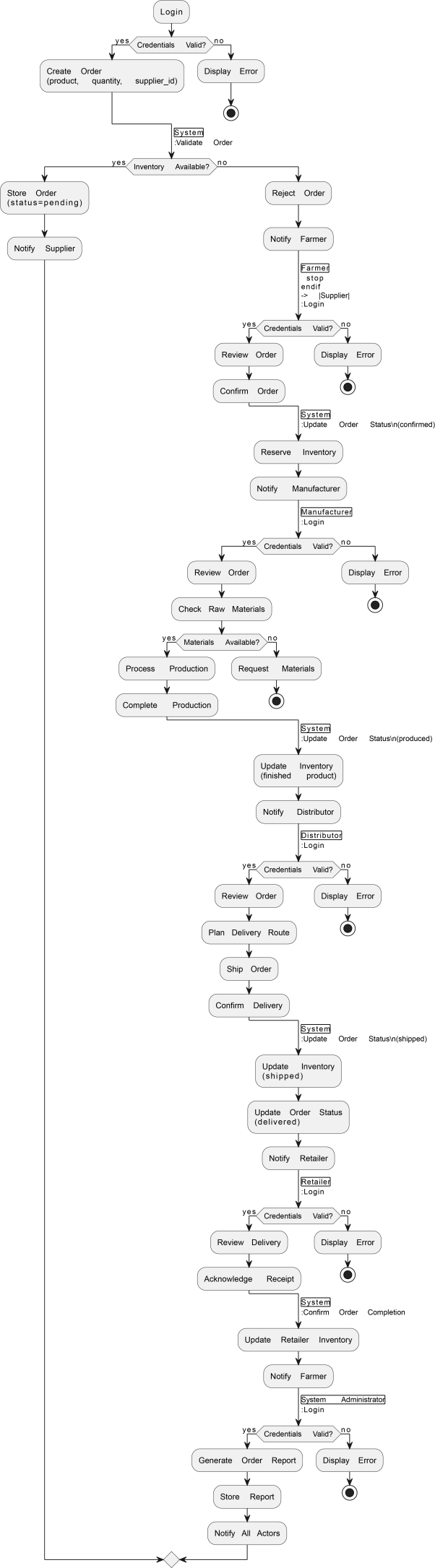


figure 3. 4 Activity diagram

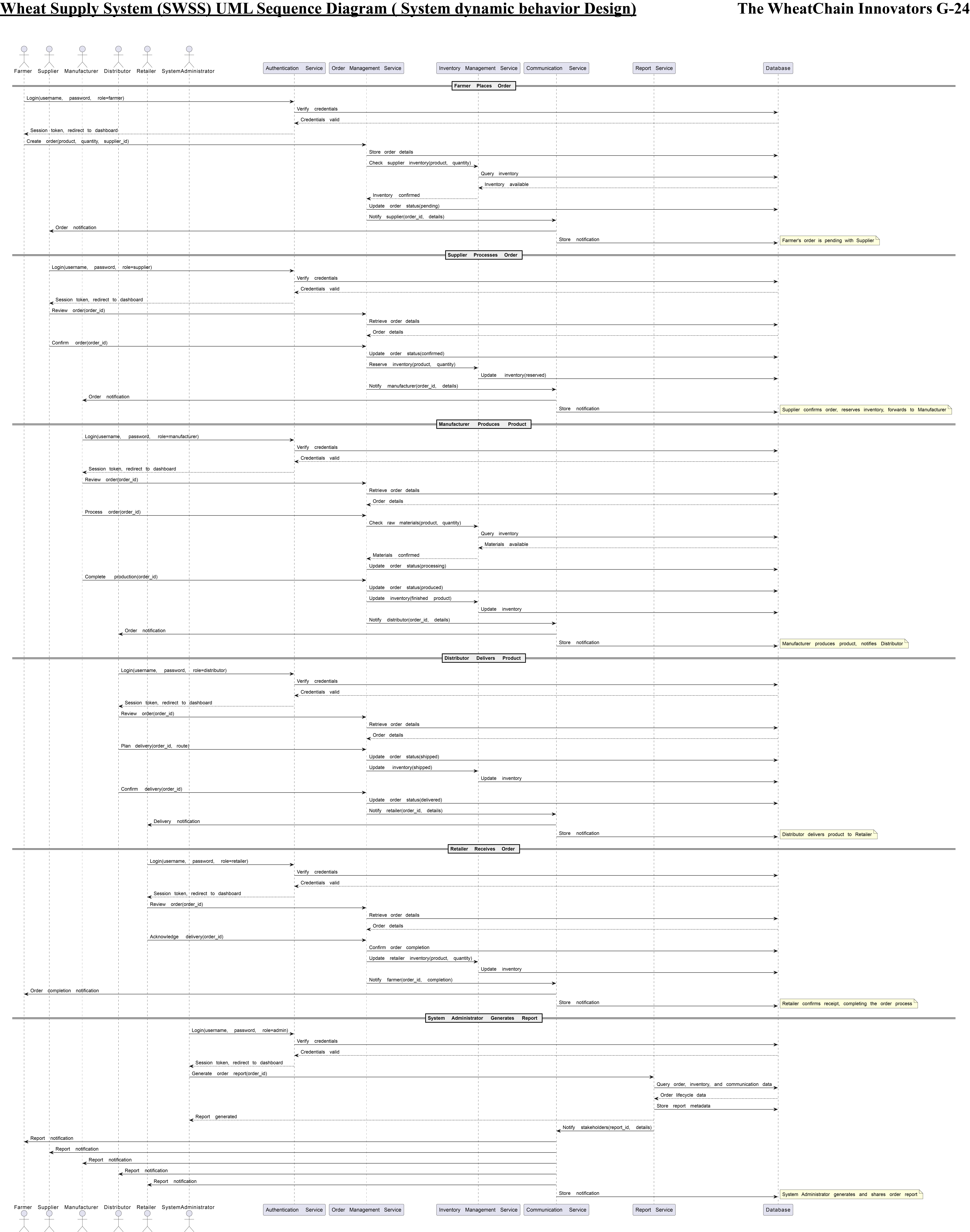


figure 3. 5 Sequence Diagram

## Design Rationale

The three-tier architecture was selected to ensure scalability, maintainability, and separation of concerns. Laravel framework provides robust MVC patterns, built-in security features, and extensive ecosystem support for rapid development. MySQL offers reliable relational data management with ACID compliance essential for supply chain data integrity.

The separate Java server for vendor validation ensures processing isolation and leverages Java's strong PDF processing capabilities. This design prevents validation processes from impacting main application performance while maintaining data consistency through API integration.

Machine learning implementation as a background service allows for computationally intensive analytics without affecting user experience. The modular subsystem design enables independent development, testing, and deployment of different functional areas.

# DATA DESIGN

## Data Description

The system's information domain transforms into normalized relational database structures optimized for supply chain operations. Core entities include Users, Products, Orders, Inventory, Vendors, and Analytics data.

User data encompasses authentication credentials, role-based permissions, and profile information for different stakeholder categories. Product information includes wheat varieties, specifications, pricing, and availability across supply chain nodes.

Order data captures complete transaction details including quantities, pricing, delivery requirements, and status tracking information. Inventory records maintain real-time stock levels, location information, and movement history.

Vendor information stores application data, validation results, performance metrics, and relationship history. Analytics data includes processed machine learning results, demand forecasts, customer segments, and historical performance indicators.

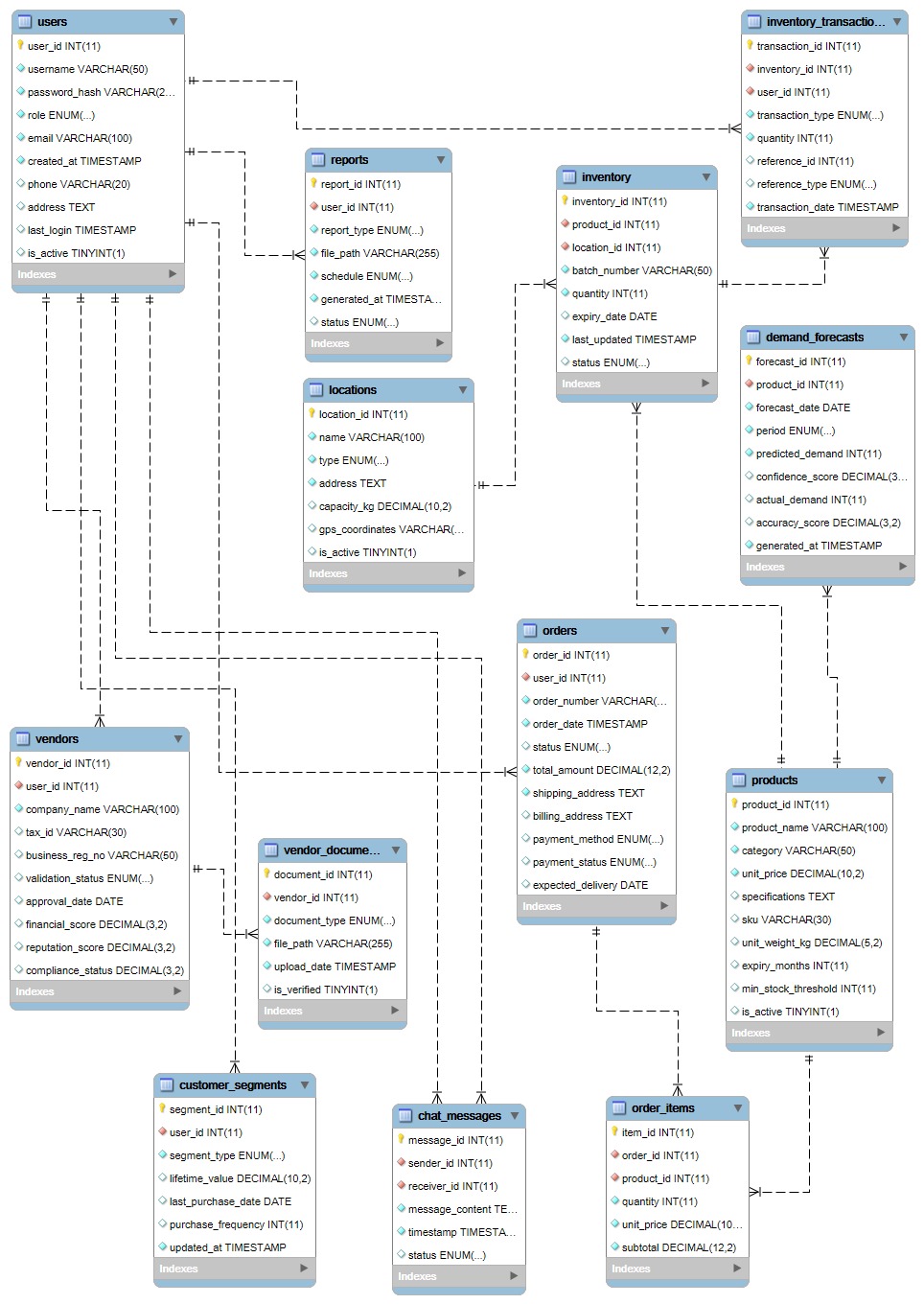


Figure 4. 1 Enhanced Entity Relationship Diagram (EERD)

| **Table Name** | **Purpose** | **Key Relationships** |
| --- | --- | --- |
| users | Store user authentication and profile data | Related to orders, inventory |
| products | Wheat product specifications and pricing | Referenced by orders, inventory |
| orders | Order transaction details and status | Links users, products, vendors |
| inventory | Stock levels and location tracking | References products, locations |
| vendors | Vendor information and validation status | Connected to orders, users |
| analytics | ML results and performance metrics | Aggregates from multiple tables |
| chat\_messages | Communication between stakeholders | Links users, references context |
| reports | Generated report metadata and scheduling | Associated with user roles |

*Table 1: Database Tables Overview*

Machine learning datasets integrate sales history, market trends, weather patterns, and economic indicators to support predictive analytics. Primary datasets include historical wheat sales data from agricultural databases, customer transaction records, and market price fluctuations.

| **Dataset** | **Source** | **Purpose** | **Features** |
| --- | --- | --- | --- |
| Wheat Sales History | USDA Agricultural Database | Demand forecasting | Date, quantity, price, region, variety |
| Customer Transactions | Internal system data | Customer segmentation | Purchase frequency, amount, preferences |
| Market Prices | Commodity exchange data | Price prediction | Daily prices, volatility, seasonal trends |
| Weather Data | Meteorological services | Crop yield forecasting | Temperature, rainfall, seasonal patterns |

*Table 2: ML Dataset Description*

## Data Dictionary /Data base

**Users Table**

**CREATE TABLE `users` (**

**`user\_id` INT NOT NULL AUTO\_INCREMENT,**

**`username` VARCHAR(50) NOT NULL,**

**`password\_hash` VARCHAR(255) NOT NULL,**

**`role` ENUM('farmer', 'supplier', 'manufacturer', 'distributor', 'retailer') NOT NULL,**

**`email` VARCHAR(100) NOT NULL,**

**`created\_at` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**`phone` VARCHAR(20),**

**`address` TEXT,**

**`last\_login` TIMESTAMP NULL,**

**`is\_active` BOOLEAN DEFAULT TRUE,**

**PRIMARY KEY (`user\_id`),**

**UNIQUE KEY `idx\_username` (`username`),**

**UNIQUE KEY `idx\_email` (`email`)**

**) ENGINE=InnoDB;**

**Products Table**

**CREATE TABLE `products` (**

**`product\_id` INT NOT NULL AUTO\_INCREMENT,**

**`product\_name` VARCHAR(100) NOT NULL,**

**`category` VARCHAR(50) NOT NULL,**

**`unit\_price` DECIMAL(10,2) NOT NULL,**

**`specifications` TEXT,**

**`sku` VARCHAR(30) UNIQUE,**

**`unit\_weight\_kg` DECIMAL(5,2),**

**`expiry\_months` INT,**

**`min\_stock\_threshold` INT DEFAULT 100,**

**`is\_active` BOOLEAN DEFAULT TRUE,**

**PRIMARY KEY (`product\_id`),**

**KEY `idx\_category` (`category`)**

**) ENGINE=InnoDB;**

**Locations Table**

**CREATE TABLE `locations` (**

**`location\_id` INT NOT NULL AUTO\_INCREMENT,**

**`name` VARCHAR(100) NOT NULL,**

**`type` ENUM('farm', 'warehouse', 'processing\_plant', 'retail') NOT NULL,**

**`address` TEXT NOT NULL,**

**`capacity\_kg` DECIMAL(10,2),**

**`gps\_coordinates` VARCHAR(50),**

**`is\_active` BOOLEAN DEFAULT TRUE,**

**PRIMARY KEY (`location\_id`)**

**) ENGINE=InnoDB;**

**Inventory Table**

**CREATE TABLE `inventory` (**

**`inventory\_id` INT NOT NULL AUTO\_INCREMENT,**

**`product\_id` INT NOT NULL,**

**`location\_id` INT NOT NULL,**

**`batch\_number` VARCHAR(50) NOT NULL,**

**`quantity` INT NOT NULL,**

**`expiry\_date` DATE,**

**`last\_updated` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**`status` ENUM('in\_stock', 'reserved', 'shipped', 'expired') DEFAULT 'in\_stock',**

**PRIMARY KEY (`inventory\_id`),**

**KEY `idx\_product\_location` (`product\_id`, `location\_id`),**

**CONSTRAINT `fk\_inventory\_product` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`) ON DELETE RESTRICT,**

**CONSTRAINT `fk\_inventory\_location` FOREIGN KEY (`location\_id`) REFERENCES `locations` (`location\_id`) ON DELETE RESTRICT**

**) ENGINE=InnoDB;**

**Inventory Transactions Table**

**CREATE TABLE `inventory\_transactions` (**

**`transaction\_id` INT NOT NULL AUTO\_INCREMENT,**

**`inventory\_id` INT NOT NULL,**

**`user\_id` INT NOT NULL,**

**`transaction\_type` ENUM('purchase', 'sale', 'transfer', 'adjustment', 'loss') NOT NULL,**

**`quantity` INT NOT NULL,**

**`reference\_id` INT,**

**`reference\_type` ENUM('order', 'shipment', 'manual'),**

**`transaction\_date` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**PRIMARY KEY (`transaction\_id`),**

**KEY `idx\_inventory` (`inventory\_id`),**

**KEY `idx\_user` (`user\_id`),**

**KEY `idx\_transaction\_date` (`transaction\_date`),**

**CONSTRAINT `fk\_transaction\_inventory` FOREIGN KEY (`inventory\_id`) REFERENCES `inventory` (`inventory\_id`) ON DELETE RESTRICT,**

**CONSTRAINT `fk\_transaction\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE RESTRICT**

**) ENGINE=InnoDB;**

**Orders Table**

**CREATE TABLE `orders` (**

**`order\_id` INT NOT NULL AUTO\_INCREMENT,**

**`user\_id` INT NOT NULL,**

**`order\_number` VARCHAR(20) NOT NULL,**

**`order\_date` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**`status` ENUM('pending', 'confirmed', 'shipped', 'delivered') DEFAULT 'pending',**

**`total\_amount` DECIMAL(12,2) NOT NULL,**

**`shipping\_address` TEXT NOT NULL,**

**`billing\_address` TEXT,**

**`payment\_method` ENUM('credit\_card', 'bank\_transfer', 'cash\_on\_delivery'),**

**`payment\_status` ENUM('pending', 'paid', 'failed', 'refunded') DEFAULT 'pending',**

**`expected\_delivery` DATE,**

**PRIMARY KEY (`order\_id`),**

**UNIQUE KEY `idx\_order\_number` (`order\_number`),**

**KEY `idx\_user` (`user\_id`),**

**KEY `idx\_status` (`status`),**

**CONSTRAINT `fk\_order\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE RESTRICT**

**) ENGINE=InnoDB;**

**Order Items Table**

**CREATE TABLE `order\_items` (**

**`item\_id` INT NOT NULL AUTO\_INCREMENT,**

**`order\_id` INT NOT NULL,**

**`product\_id` INT NOT NULL,**

**`quantity` INT NOT NULL,**

**`unit\_price` DECIMAL(10,2) NOT NULL,**

**`subtotal` DECIMAL(12,2) NOT NULL,**

**PRIMARY KEY (`item\_id`),**

**KEY `idx\_order` (`order\_id`),**

**KEY `idx\_product` (`product\_id`),**

**CONSTRAINT `fk\_item\_order` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`order\_id`) ON DELETE CASCADE,**

**CONSTRAINT `fk\_item\_product` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`) ON DELETE RESTRICT**

**) ENGINE=InnoDB;**

**Vendors Table**

**CREATE TABLE `vendors` (**

**`vendor\_id` INT NOT NULL AUTO\_INCREMENT,**

**`user\_id` INT NOT NULL,**

**`company\_name` VARCHAR(100) NOT NULL,**

**`tax\_id` VARCHAR(30) UNIQUE,**

**`business\_reg\_no` VARCHAR(50),**

**`validation\_status` ENUM('pending', 'approved', 'rejected', 'pending\_visit') DEFAULT 'pending',**

**`approval\_date` DATE,**

**`financial\_score` DECIMAL(3,2),**

**`reputation\_score` DECIMAL(3,2),**

**`compliance\_status` DECIMAL(3,2),**

**PRIMARY KEY (`vendor\_id`),**

**UNIQUE KEY `idx\_user` (`user\_id`),**

**CONSTRAINT `fk\_vendor\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE**

**) ENGINE=InnoDB;**

**Vendor Documents Table**

**CREATE TABLE `vendor\_documents` (**

**`document\_id` INT NOT NULL AUTO\_INCREMENT,**

**`vendor\_id` INT NOT NULL,**

**`document\_type` ENUM('tax\_cert', 'license', 'insurance', 'financial\_statement', 'other') NOT NULL,**

**`file\_path` VARCHAR(255) NOT NULL,**

**`upload\_date` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**`is\_verified` BOOLEAN DEFAULT FALSE,**

**PRIMARY KEY (`document\_id`),**

**KEY `idx\_vendor` (`vendor\_id`),**

**CONSTRAINT `fk\_document\_vendor` FOREIGN KEY (`vendor\_id`) REFERENCES `vendors` (`vendor\_id`) ON DELETE CASCADE**

**) ENGINE=InnoDB;**

**Demand Forecasts Table**

**CREATE TABLE `demand\_forecasts` (**

**`forecast\_id` INT NOT NULL AUTO\_INCREMENT,**

**`product\_id` INT NOT NULL,**

**`forecast\_date` DATE NOT NULL,**

**`period` ENUM('weekly', 'monthly', 'quarterly') NOT NULL,**

**`predicted\_demand` INT NOT NULL,**

**`confidence\_score` DECIMAL(3,2),**

**`actual\_demand` INT,**

**`accuracy\_score` DECIMAL(3,2),**

**`generated\_at` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**PRIMARY KEY (`forecast\_id`),**

**KEY `idx\_product` (`product\_id`),**

**CONSTRAINT `fk\_forecast\_product` FOREIGN KEY (`product\_id`) REFERENCES `products` (`product\_id`) ON DELETE CASCADE**

**) ENGINE=InnoDB;**

**Customer Segments Table**

**CREATE TABLE `customer\_segments` (**

**`segment\_id` INT NOT NULL AUTO\_INCREMENT,**

**`user\_id` INT NOT NULL,**

**`segment\_type` ENUM('premium\_buyer', 'bulk\_purchaser', 'seasonal\_customer', 'price\_sensitive\_buyer', 'occasional\_purchaser') NOT NULL,**

**`lifetime\_value` DECIMAL(10,2),**

**`last\_purchase\_date` DATE,**

**`purchase\_frequency` INT,**

**`updated\_at` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**PRIMARY KEY (`segment\_id`),**

**UNIQUE KEY `idx\_user\_segment` (`user\_id`),**

**CONSTRAINT `fk\_segment\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE**

**) ENGINE=InnoDB;**

**Chat Messages Table**

**CREATE TABLE `chat\_messages` (**

**`message\_id` INT NOT NULL AUTO\_INCREMENT,**

**`sender\_id` INT NOT NULL,**

**`receiver\_id` INT NOT NULL,**

**`message\_content` TEXT NOT NULL,**

**`timestamp` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**`status` ENUM('sent', 'delivered', 'read') DEFAULT 'sent',**

**PRIMARY KEY (`message\_id`),**

**KEY `idx\_sender` (`sender\_id`),**

**KEY `idx\_receiver` (`receiver\_id`),**

**CONSTRAINT `fk\_message\_sender` FOREIGN KEY (`sender\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE,**

**CONSTRAINT `fk\_message\_receiver` FOREIGN KEY (`receiver\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE**

**) ENGINE=InnoDB;**

**Reports Table**

**CREATE TABLE `reports` (**

**`report\_id` INT NOT NULL AUTO\_INCREMENT,**

**`user\_id` INT NOT NULL,**

**`report\_type` ENUM('sales', 'inventory', 'vendor\_performance', 'demand\_forecast', 'customer\_segmentation') NOT NULL,**

**`file\_path` VARCHAR(255) NOT NULL,**

**`schedule` ENUM('daily', 'weekly', 'monthly', 'one\_time') NOT NULL,**

**`generated\_at` TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**`status` ENUM('pending', 'generated', 'failed') DEFAULT 'pending',**

**PRIMARY KEY (`report\_id`),**

**KEY `idx\_user` (`user\_id`),**

**KEY `idx\_report\_type` (`report\_type`),**

**CONSTRAINT `fk\_report\_user` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE RESTRICT**

**) ENGINE=InnoDB;**

**Stored Procedure for Order Number Generation**

**DELIMITER //**

**CREATE PROCEDURE `generate\_order\_number`(OUT new\_order\_number VARCHAR(20))**

**BEGIN**

**DECLARE prefix VARCHAR(3) DEFAULT 'ORD';**

**DECLARE sequence\_num INT;**

**SELECT COALESCE(MAX(SUBSTRING(order\_number, 5)), 0) + 1 INTO sequence\_num**

**FROM orders**

**WHERE order\_number LIKE CONCAT(prefix, '%');**

**SET new\_order\_number = CONCAT(prefix, LPAD(sequence\_num, 7, '0'));**

**END //**

**DELIMITER ;**

**Initial Data**

**INSERT INTO `users` (`username`, `password\_hash`, `role`, `email`, `phone`, `is\_active`)**

**VALUES**

**('admin\_user', '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'farmer', 'admin@swss.com', '+1234567890', TRUE);**

# COMPONENT DESIGN

| **Component** | **Primary Functions** | **Dependencies** |
| --- | --- | --- |
| Authentication | Login, logout, session management | User database, encryption |
| Order Management | Create, update, track orders | Inventory, user validation |
| Inventory Control | Stock monitoring, replenishment | Product database, analytics |
| Vendor Validation | Application processing, approval | File system, Java server |
| ML Analytics | Demand forecasting, segmentation | Historical data, algorithms |

*Table 3: Component Functions Summary*

## Authentication Component:

FUNCTION authenticateUser(username, password)

BEGIN

hash\_password = encryptPassword(password)

user\_record = queryDatabase("SELECT \* FROM users WHERE username = ? AND password\_hash = ?", username, hash\_password)

IF user\_record EXISTS THEN

createSession(user\_record.user\_id, user\_record.role)

RETURN success\_response

ELSE

RETURN authentication\_failed

END IF

END

## Order Processing Component:

FUNCTION processOrder(user\_id, product\_id, quantity)

BEGIN

validateUser(user\_id)

product = getProduct(product\_id)

IF checkInventoryAvailability(product\_id, quantity) THEN

order\_id = createOrder(user\_id, product\_id, quantity, product.unit\_price \* quantity)

updateInventory(product\_id, -quantity)

sendNotification(user\_id, "Order confirmed: " + order\_id)

RETURN order\_confirmation

ELSE

RETURN insufficient\_inventory\_error

END IF

END

## Inventory Management Component:

FUNCTION monitorInventoryLevels()

BEGIN

products = getAllProducts()

FOR each product IN products DO

current\_stock = getCurrentStock(product.product\_id)

minimum\_threshold = getMinimumThreshold(product.product\_id)

IF current\_stock < minimum\_threshold THEN

predicted\_demand = callMLPrediction(product.product\_id)

suggested\_quantity = calculateReplenishmentQuantity(predicted\_demand, current\_stock)

generateReplenishmentAlert(product.product\_id, suggested\_quantity)

END IF

END FOR

END

## Vendor Validation Component:

FUNCTION validateVendorApplication(application\_pdf)

BEGIN

extracted\_data = extractDataFromPDF(application\_pdf)

financial\_score = analyzeFinancialStability(extracted\_data.financial\_records)

reputation\_score = checkReputationMetrics(extracted\_data.references)

compliance\_status = verifyRegulatoryCompliance(extracted\_data.certifications)

overall\_score = (financial\_score \* 0.4) + (reputation\_score \* 0.3) + (compliance\_status \* 0.3)

IF overall\_score >= VALIDATION\_THRESHOLD THEN

scheduleFacilityVisit(extracted\_data.vendor\_id)

updateVendorStatus(extracted\_data.vendor\_id, "pending\_visit")

RETURN validation\_passed

ELSE

updateVendorStatus(extracted\_data.vendor\_id, "rejected")

RETURN validation\_failed

END IF

END

## Machine Learning Analytics Component:

FUNCTION generateDemandForecast(product\_id, forecast\_period)

BEGIN

historical\_sales = getHistoricalSalesData(product\_id, PAST\_24\_MONTHS)

seasonal\_factors = calculateSeasonalityFactors(historical\_sales)

trend\_analysis = performTrendAnalysis(historical\_sales)

external\_factors = getExternalMarketFactors()

ml\_model = loadTrainedModel("demand\_forecasting\_model")

input\_features = combineFeatures(seasonal\_factors, trend\_analysis, external\_factors)

demand\_prediction = ml\_model.predict(input\_features, forecast\_period)

storeAnalyticsResults(product\_id, demand\_prediction, CURRENT\_TIMESTAMP)

RETURN demand\_prediction

END

## Customer Segmentation Component:

FUNCTION segmentCustomers()

BEGIN

customer\_data = getAllCustomerTransactionData()

feature\_matrix = extractCustomerFeatures(customer\_data)

clustering\_model = loadTrainedModel("customer\_segmentation\_model")

customer\_segments = clustering\_model.fitPredict(feature\_matrix)

FOR each customer IN customer\_data DO

segment\_id = customer\_segments[customer.index]

updateCustomerSegment(customer.user\_id, segment\_id)

generatePersonalizationRecommendations(customer.user\_id, segment\_id)

END FOR

RETURN segmentation\_complete

END

# HUMAN INTERFACE DESIGN

## 6.1 Overview of User Interface

The user interface provides role-based access to system functionality through responsive web design optimized for desktop and mobile devices. Users authenticate through a centralized login system that redirects to appropriate dashboards based on their assigned roles.

Farmers access interfaces for crop planning, harvest recording, and order placement to suppliers. They receive demand forecasts and pricing recommendations to optimize their production decisions. The system displays weather integration and seasonal guidance to support agricultural planning.

Suppliers interact with order management interfaces to process requests from farmers and fulfill orders to manufacturers. They access inventory tracking, vendor validation status, and communication tools for coordinating with multiple stakeholders.

Manufacturers utilize production planning interfaces that integrate demand forecasts with inventory levels to optimize processing schedules. They access quality control tracking, batch management, and distribution coordination tools.

Distributors manage logistics interfaces including route optimization, delivery scheduling, and inventory allocation across multiple retail locations. They receive analytics on distribution efficiency and customer demand patterns.

Retailers access point-of-sale integration, inventory management, and customer analytics to optimize product placement and pricing strategies. They receive personalized recommendations for inventory stocking based on local customer segmentation analysis.

## Screen Images

Figure 6. 1 Login Interface Mock up

**Username:**

**Password:**

**Role:**

Dropdown Menu ▼

**LOGIN BUTTON**

**Forgot Password?** |

**Register**

**WHEAT SCM LOGIN**

Figure 6. 2 Dashboard Interface Mock up

**MENU**

Orders Invent Reports Chat Profile

Pending Orders

15

Inventory Alerts

3

Forecasts & ML Insights

**RECENT ACTIVITY FEED**

* New order from Supplier ABC
* Inventory low: Premium Wheat
* ML Alert: Demand spike predicted

**DASHBOARD OVERVIEW**

**SCM WHEAT** |

**Dashboard**

**Logout**

**Messages**

**Reports**

Figure 6. 3 Order Management Interface Mock up

**ORDER MANAGEMENT**

**NEW ORDER**

**EXPORT**

**FILTER** ▼

**Search:**

\_\_\_

**Order ID**

**Customer**

**Product**

**Qty**

**Status**

ORD-001 ORD-002 ORD-003

Mill Co. Baker Ltd Export AG

Hard Wheat Soft Wheat Premium

500 200 1000

Processing Shipped Confirmed

**PREV**

**NEXT**

## Screen Objects and Actions

## Login Screen Objects:

* Username input field: Text entry for user identification
* Password input field: Secure text entry with masking
* Role dropdown: Selection menu for user role specification
* Login button: Submits authentication credentials
* Forgot password link: Initiates password recovery process
* Register link: Redirects to new user registration

## Dashboard Screen Objects:

* Navigation menu: Provides access to all system modules
* Summary cards: Display key performance indicators and alerts
* Activity feed: Shows recent system events and notifications
* Quick action buttons: Enable rapid access to common functions
* User profile dropdown: Provides access to account settings and logout

## Order Management Screen Objects:

* Order table: Displays order information in sortable columns
* New order button: Opens order creation form
* Filter dropdown: Enables order filtering by various criteria
* Search field: Allows text-based order searching
* Export button: Generates order reports in various formats
* Pagination controls: Navigate through multiple pages of orders
* Status indicators: Visual representation of order progress

## Actions and Interactions:

* Click login button triggers authentication validation and role-based redirection
* Dashboard cards are clickable and expand to show detailed information
* Order table rows are selectable for bulk operations and detailed view access
* Filter and search functions provide real-time results updating
* Chat interface enables real-time messaging with typing indicators and message status
* Report generation triggers background processing with progress indicators
* Inventory alerts provide one-click access to replenishment workflows

# MACHINE LEARNING MODEL

The system implements two primary machine learning models to support supply chain optimization: demand forecasting and customer segmentation.

**Demand Forecasting Model**: The demand prediction model utilizes a Long Short-Term Memory (LSTM) neural network to analyze temporal patterns in wheat sales data. The model incorporates multiple input features including historical sales volumes, seasonal variations, market prices, weather conditions, and economic indicators.

Data preprocessing involves normalization of numerical features and encoding of categorical variables. The model training uses three years of historical data with 80% allocated for training and 20% for validation. Feature engineering includes rolling averages, lag variables, and seasonal decomposition components.

The LSTM architecture consists of two hidden layers with 50 neurons each, followed by a dense output layer. The model uses Adam optimizer with learning rate scheduling and early stopping to prevent overfitting. Cross-validation ensures model robustness across different time periods.

Model evaluation metrics include Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). The trained model achieves approximately 85% accuracy in demand prediction for one-month forecasts and 75% accuracy for three-month forecasts.

**Customer Segmentation Model**: The customer segmentation implementation uses K-means clustering algorithm to group customers based on purchasing behavior patterns. Input features include purchase frequency, average order value, product preferences, seasonal buying patterns, and customer lifetime value.

Feature scaling applies standardization to ensure equal weight for all clustering variables. The optimal number of clusters is determined using elbow method and silhouette analysis, resulting in five distinct customer segments: Premium Buyers, Bulk Purchasers, Seasonal Customers, Price-Sensitive Buyers, and Occasional Purchasers.

Each segment receives tailored recommendations for product offerings, pricing strategies, and communication preferences. The clustering model updates monthly to capture evolving customer behavior patterns and market dynamics.

Model validation includes cluster stability analysis and business interpretation alignment. The segmentation model demonstrates 78% classification accuracy when validated against known customer behavior patterns.

**Implementation Architecture**: Both models operate as background services integrated with the main Laravel application through API endpoints. Model training occurs on scheduled intervals using Apache Airflow for workflow orchestration. Trained models are serialized and stored in dedicated model repository for versioning and rollback capabilities.

Real-time prediction requests are served through Flask microservices that load pre-trained models and return predictions via RESTful APIs. Model performance monitoring includes drift detection and automatic retraining triggers when prediction accuracy degrades below defined thresholds.

# APPENDICES

**Appendix A: Database Schema Diagrams** Complete entity-relationship diagrams showing all table relationships, foreign key constraints, and indexing strategies for optimal query performance.

**Appendix B: API Documentation** Detailed REST API specifications including endpoint descriptions, request/response formats, authentication requirements, and error handling procedures.

**Appendix C: Machine Learning Model Specifications** Technical specifications for neural network architectures, hyperparameter configurations, training procedures, and performance benchmarks.

**Appendix D: Security Requirements** Comprehensive security measures including data encryption, access control policies, audit logging, and vulnerability assessment procedures.

**Appendix E: Deployment Guide** Step-by-step deployment instructions for development, staging, and production environments including server configuration, database setup, and monitoring implementation.

**Appendix F: Testing Strategy** Complete testing framework including unit tests, integration tests, performance tests, and user acceptance testing procedures for quality assurance.