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

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. 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.

1.2. 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.

1.3. 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.

1.4. 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

1.5. 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

2. 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.

3. SYSTEM ARCHITECTURE

3.1. 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.

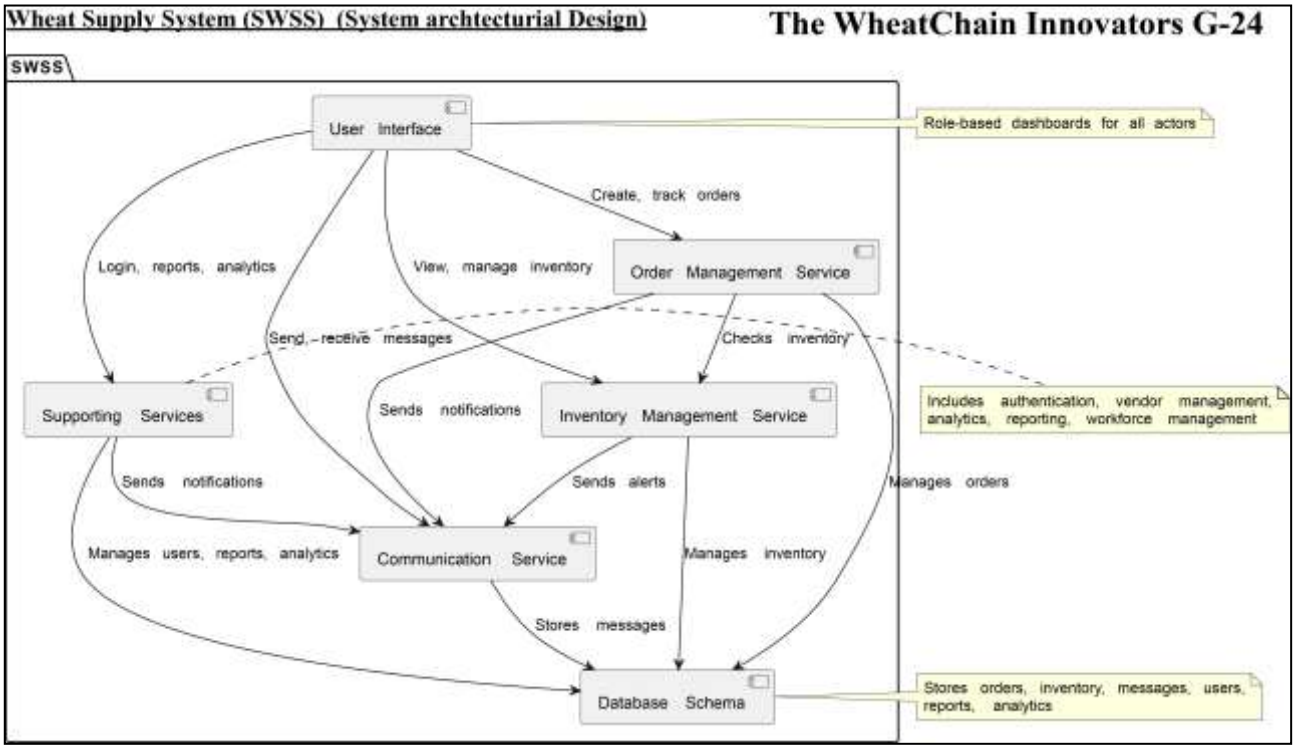

figure 3. 1 System Architecture Overview

3.2. 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.

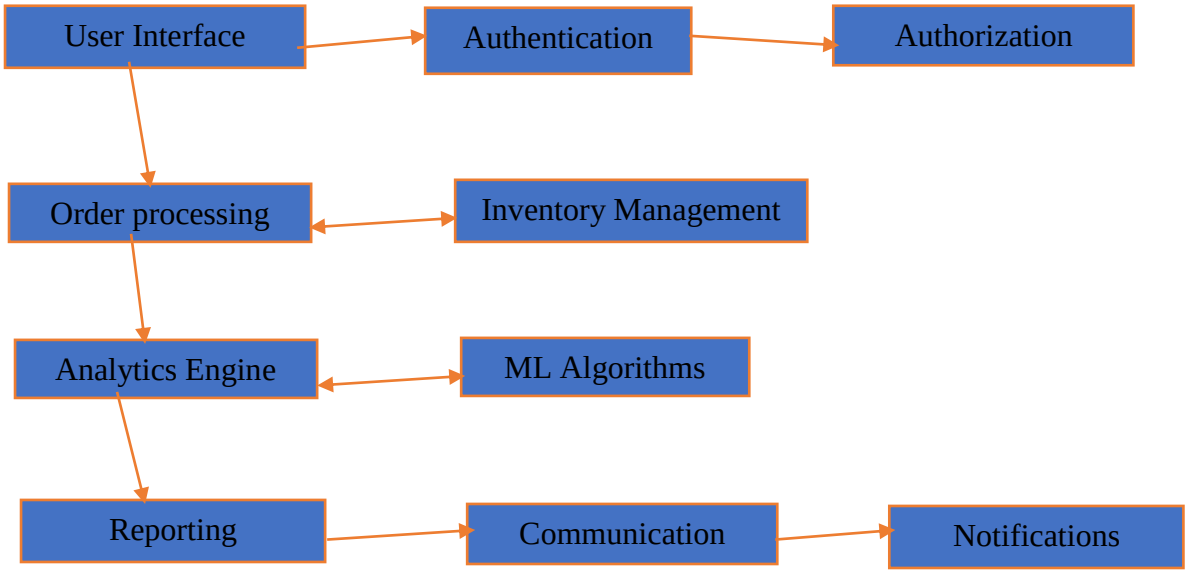

figure 3. 2 Component Interaction Diagram

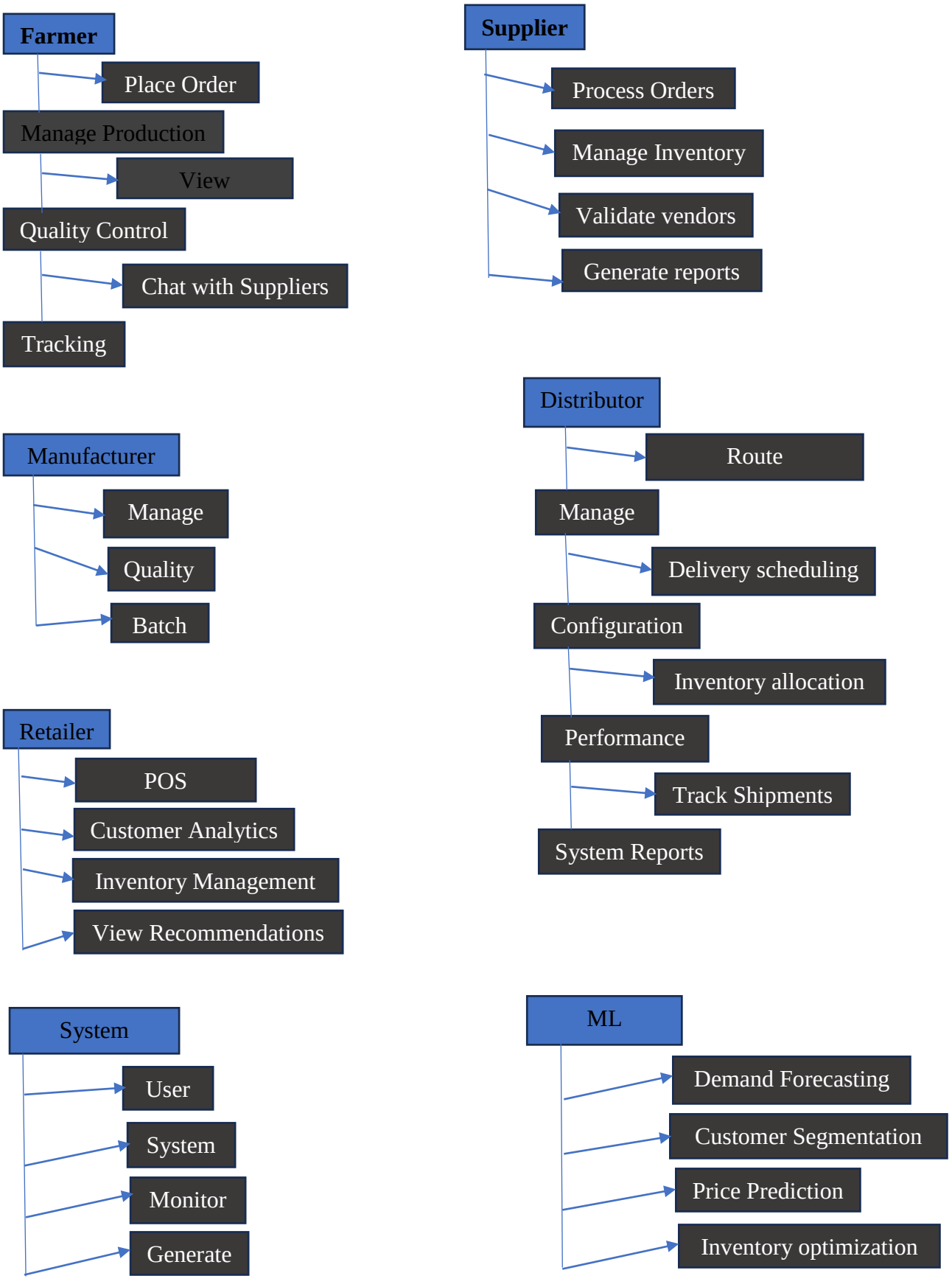

figure 3. 3 Use Case Diagram

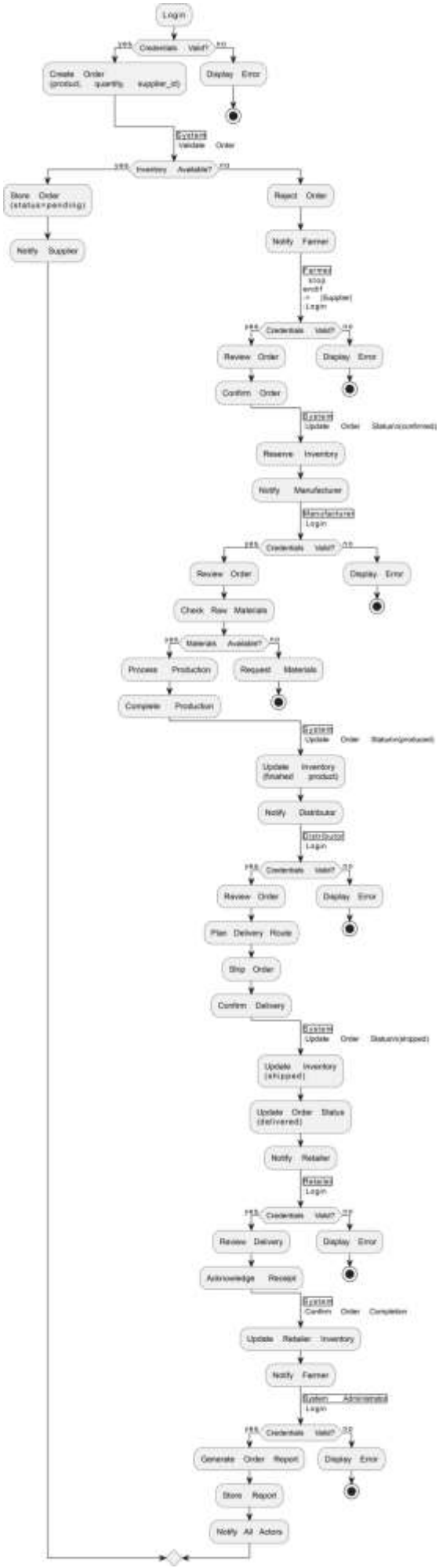

figure 3. 4 Activity diagram

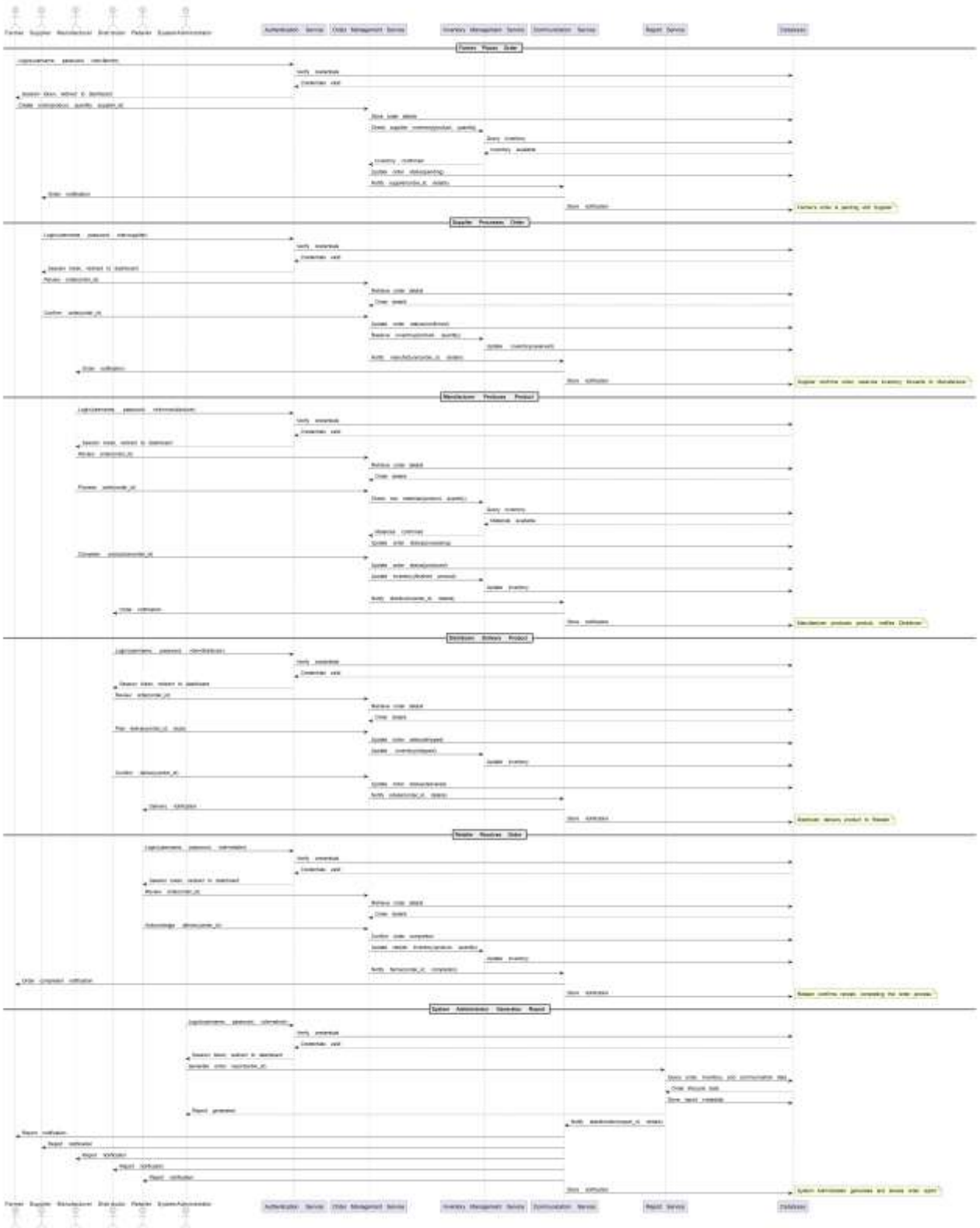

figure 3. 5 Sequence Diagram

3.3. 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.

4. DATA DESIGN

4.1. 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.

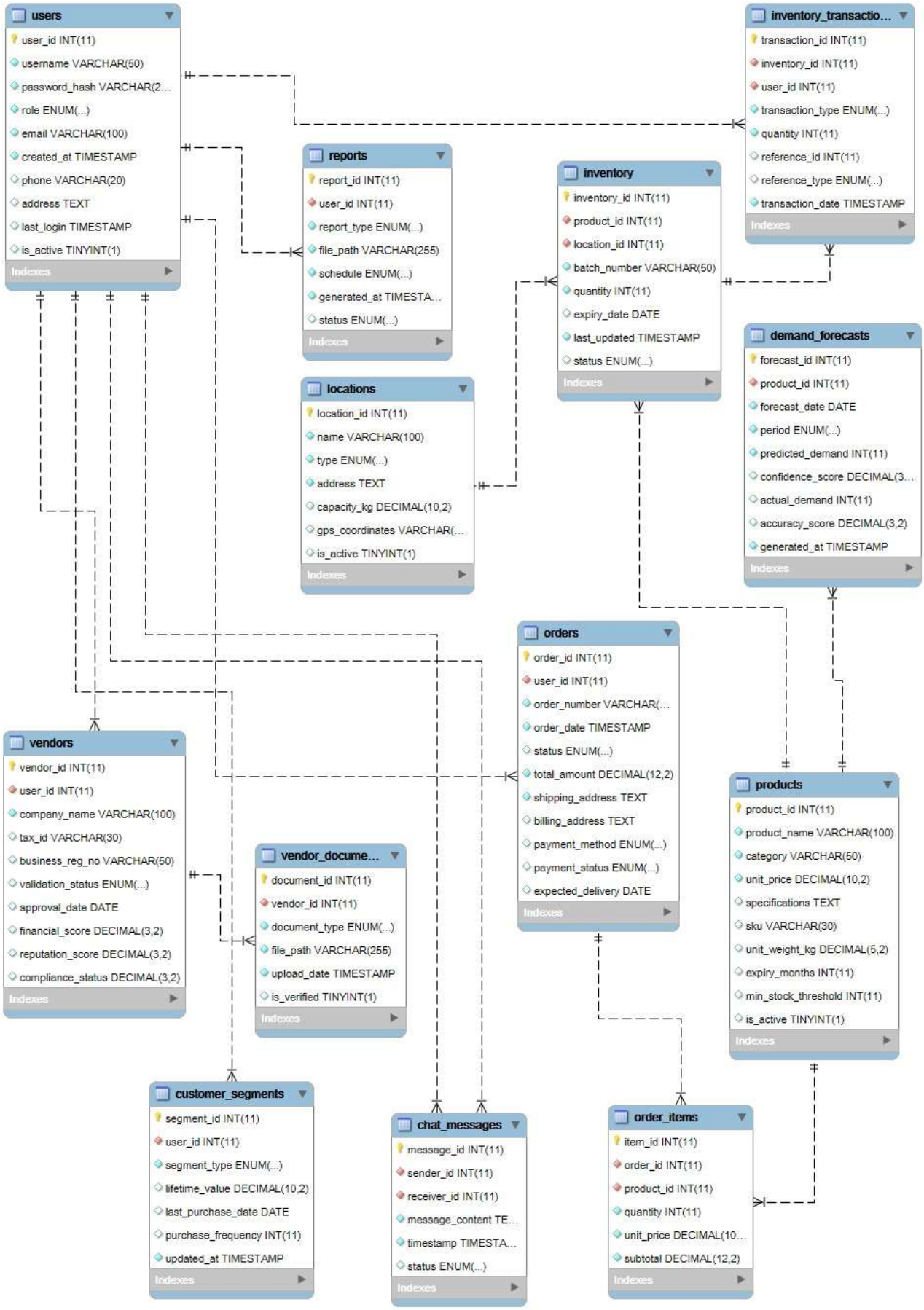

Figure 4. 1 Enhanced Entity Relationship Diagram (EERD)

•

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

4.2. 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);
```

5. 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

5.1. 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
```

5.2. 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
```

5.3. 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
```

5.4. 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
```

5.5. 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
```

5.6. 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
```

6. HUMAN INTERFACE DESIGN

6.1. 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.

6.2. Screen Images

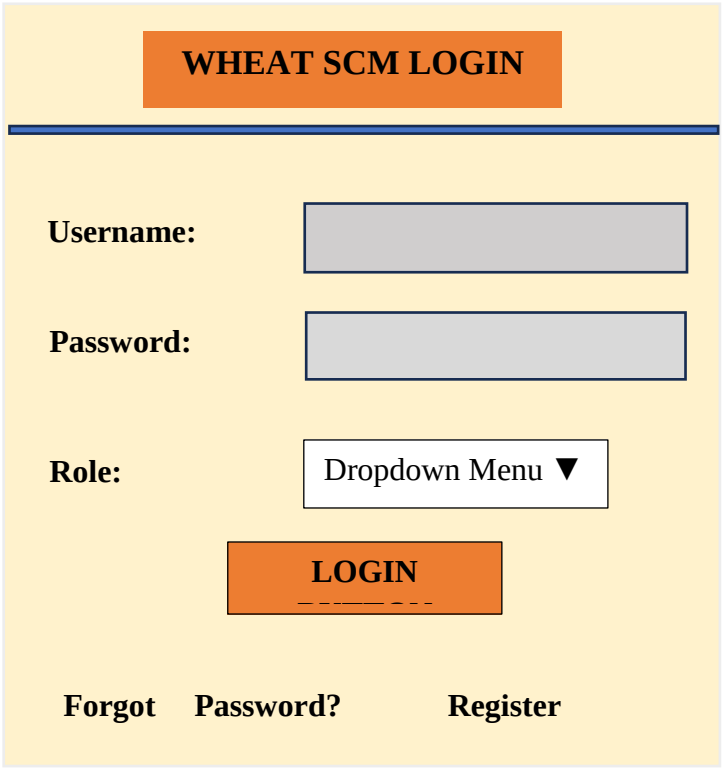

Figure 6. 1 Login Interface Mock up

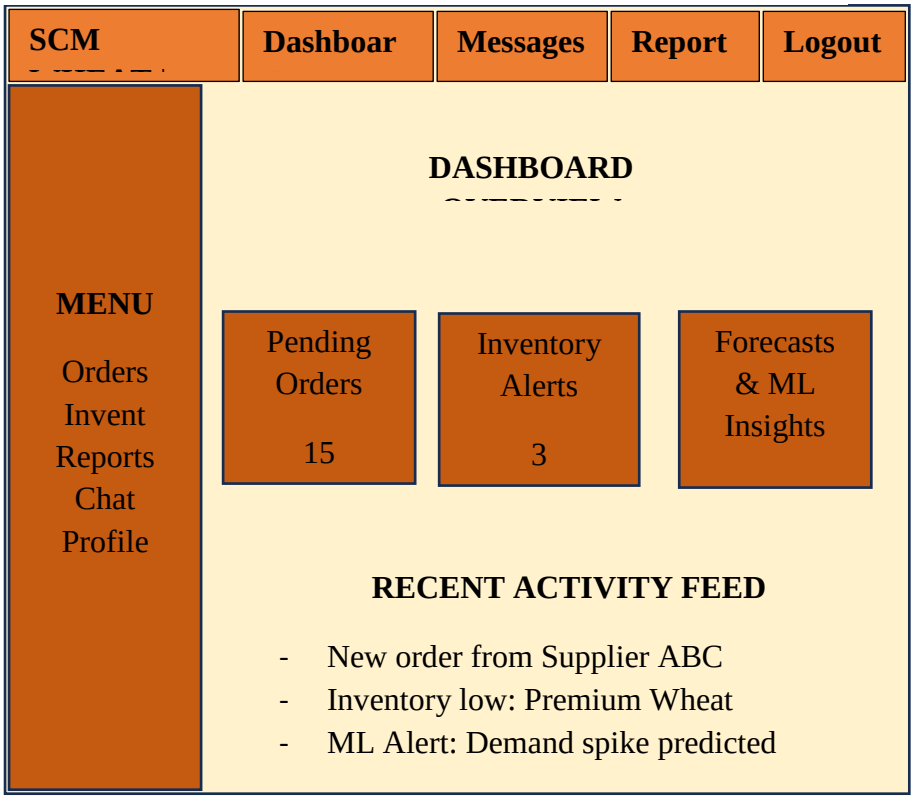

Figure 6. 2 Dashboard Interface Mock up

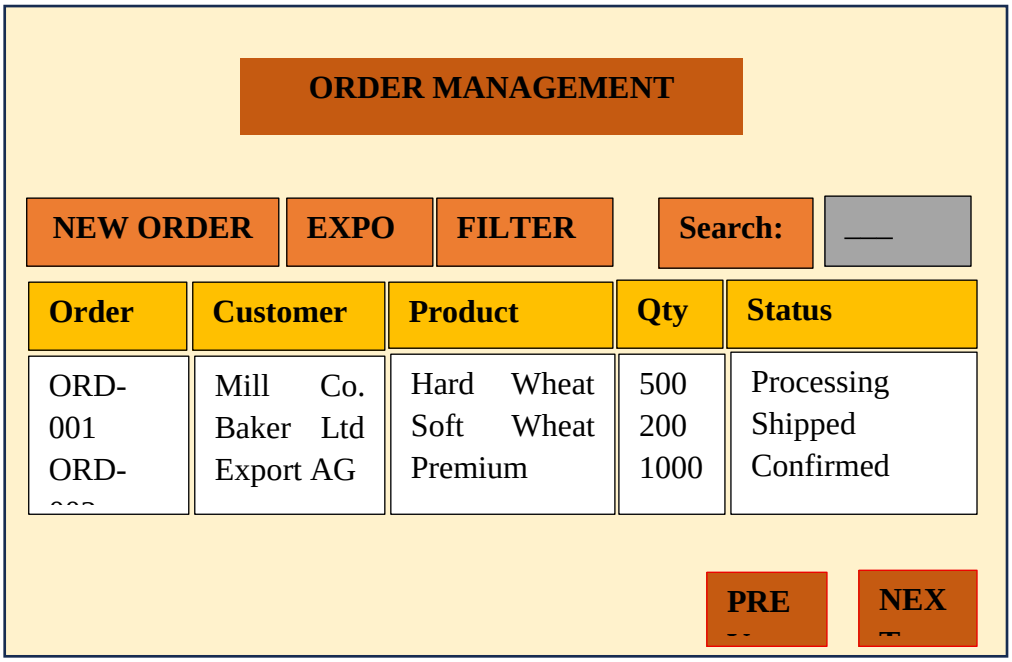

Figure 6. 3 Order Management Interface Mock up

6.3. Screen Objects and Actions

6.4. 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

6.5. 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

6.6. 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

6.7. 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

7. 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.

8. 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.