# SOFTWARE DESIGN DOCUMENT

**Project Title:** Smart Bread Supply System (SBSS)

**Team Name:** The BreadChain Innovators

**Group Number:** G-24

**Supervisor:** Lule Emmanuel

**GitHub Link**: https://github.com/Kayeerasoftware/The-Bread-Chain-Innovators-G-24.git

## 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 |

**Prepared for**: Professional Min Software Engineering Practical Project recess term for academic year 2025

# TABLE OF CONTENT

# LIST OF TABLES

# TABLE OF FIGURES

# Glossary of Terms

| Term | Meaning |
|------|---------|
| SBSS | Smart Bread Supply System |
| ML | Machine Learning |
| ARIMA | AutoRegressive Integrated Moving Average |
| K-Means | Clustering algorithm for segmentation |
| Vendor | A person or company supplying raw materials |

*Table 1: Showing Glossary of Terms*

# 1. Introduction

## 1.1.   Purpose

This Software Design Document (SDD) outlines the architecture, design, and imple- mentation details of the Smart Bread Supply System (SBSS), a web-based platform to streamline the bread supply chain from raw material acquisition to retail distribution. It serves as a blueprint for developers, stakeholders, and the supervisor to ensure alignment during development, testing, and deployment.

## 1.2.   Scope

SBSS is a web-based supply chain management system built with Laravel (PHP), MySQL, and a Java server for vendor validation. It tracks bread production and distribution from raw material suppliers to factories, through distributors, and finally to retailers. Key features include:

- ML-powered demand forecasting (ARIMA)
- Customer segmentation (K-means clustering)
- Real-time stakeholder communication
- Inventory and order lifecycle management
- Workforce allocation and task scheduling
- Vendor application validation via Java and PDF analysis
- Role-based dashboards and access control

The system aims to minimize stock wastage, optimize operations, and enhance visibility across the supply chain.

## 1.3.   Overview

This SDD adheres to IEEE standards for software design documentation. It covers the system architecture, data models, component designs, user interfaces, and ML algorithms used in forecasting and segmentation,

ensuring a clear path to developing a functional system.

## 1.4. User stories

- **Supplier:** I want to receive raw material requests and track delivery schedules to ensure timely supply.

- **Factory Worker:** I want to monitor baking tasks, inventory levels, and production schedules.

- **Distributor:** I want a clear delivery schedule and the ability to update delivery statuses in real-time.

- **Retailer:** I want to place orders and receive bread stock before peak sales hours.

- **Admin:** I want to validate vendor applications, manage users, and generate operational reports.

## 1.5. Project Timeline

| Phase | Timeline |
|---|---|
| System Design | May 2025 (Ongoing) |
| Development | June 2025 |
| Testing | July 2025 |
| Deployment | Mid-July 2025 |
| Presentation | Late July 2025 |

*Table 2: Showing Project Timeline*

# 2. System Overview

SBSS digitizes and manages the full journey of bread production—from raw material suppliers (e.g., flour, yeast) to factories, through distributors, and down to retail stores. It provides role-based dashboards, ML-driven analytics, and seamless coordination tools.

The system supports five key roles:

- **Raw Material Suppliers** – Manage stock and fulfill factory requests
- **Factory Workers** – Oversee production and inventory
- **Distributors** – Handle transportation and delivery tracking
- **Retailers** – Place orders and monitor stock levels
- **Admins** – Manage users, validate vendors, and access analytics

Each role has unique access to dashboards and tools aligned with their responsibilities in the bread supply chain.

# 3. System Architecture

## 3.1. Architectural Design

SBSS employs a layered architecture for modularity and scalability:

- **Presentation Layer**: Laravel Blade templates with Tailwind CSS for responsive, role-based dashboards

- **Business Logic Layer**: Laravel controllers and services handle inventory, orders, ML analytics, and reports.
- **Data Access Layer:** Eloquent ORM for MySQL database interactions
- **Machine Learning Module:** Python-based ARIMA and K-Means for forecasting and segmenta- tion.
- **Vendor Validation Server:** Java-based REST API for secure PDF processing
- **Chat Module:** Laravel Echo + WebSockets with Pusher for real-time communication

## 3.2. Decomposition Description

| Layer | Description |
|---|---|
| Presentation | Role-based dashboards for suppliers, workers, distributors, retailers, and admins. |
| Business Logic | Inventory, orders, ML, reports, vendor evaluation |
| Data Access | Eloquent models for all entities |
| ML Module | ARIMA for forecasting, K-means for segmentation |
| Vendor Server | REST API for PDF scoring and validation |
| Chat Service | Real-time messaging using Pusher |

*Table 3: Showing Decomposition Description.*

## 3.3. Design Rationale

Laravel ensures rapid, structured development; MySQL provides relational storage; Java enhances secure vendor handling; Python delivers robust ML capabilities. Simpler than microservices but modular for scalability.

# 4. Data Design

## 4.1.   Data Description

| Entity | Description |
|--------|-------------|
| User | Stakeholder info (role, contact, auth) |
| Inventory | Stock records (location, quantity, stage) |
| Order | Transaction info between users |
| Vendor Application | PDF evaluation results |
| Report | Generated insights for each role |
| Chat Message | Sent/received user communications |

*Table 4: Showing Data Description.*

## 4.2.   Data Dictionary

**User**: { id (int, PK), name (varchar), email (varchar, unique), password (varchar), role (enum: supplier/factory/distributor/retailer/admin) }

**Inventory**: { id (int, PK), location (varchar), quantity (int), stage (enum: raw/processed/packag updatedat(timestamp)}

**Order**:{id(int, PK), fromuserid(int, FK), touserid(int, FK), quantity(i pending/shipped/delivered), createdat(timestamp)}

**VendorApplication**{ id (int, PK), applicantid(int, FK), financialscore(float), reputationscore(flo pending/approved/rejected), createdat(timestamp)}

**Report**:{id(int, PK), stakeholderid(int, FK), ty demand/inventory/sales), content(text), scheduledat(timestamp)}

**ChatMessage**: { id (int, PK), senderid(int, FK), receiverid(int, FK), message(text), sentat(timestam

## 4.3. Component Design

### 4.3.1. Inventory Management

Inventory tracks raw materials, in-process goods, and finished products. Suppliers update stock levels, factories monitor production inputs/outputs, and distributors/retailers view available stock.
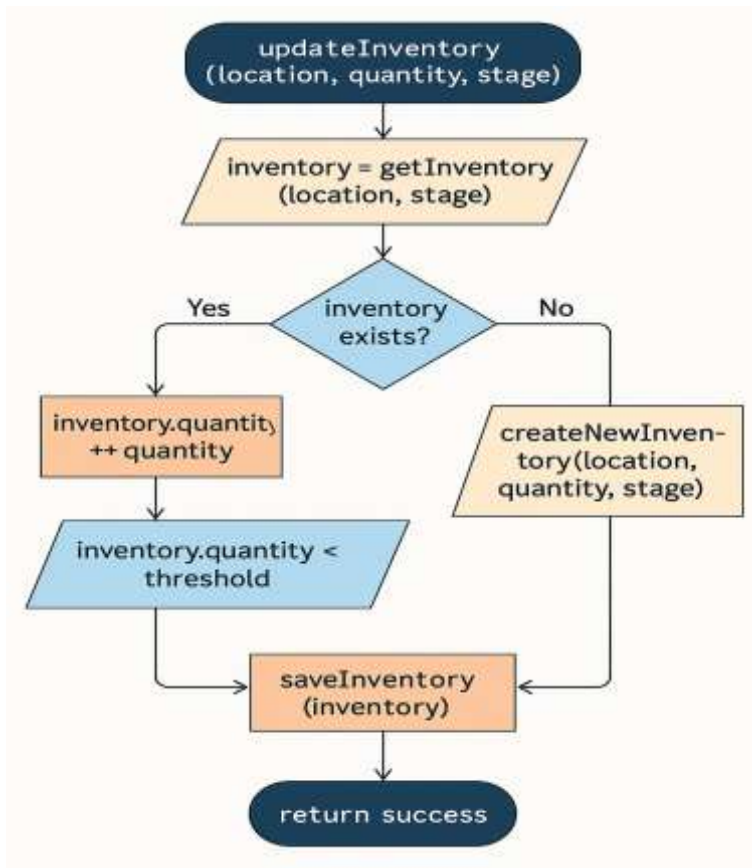


*Figure 4. 1: Showing Inventory Management Flow/Logic*

### 4.3.2. Order Processing

Orders are initiated by retailers or distributors, routed to factories, and fulfilled by sup- pliers. Status updates (pending, shipped, delivered) are tracked in real-time.
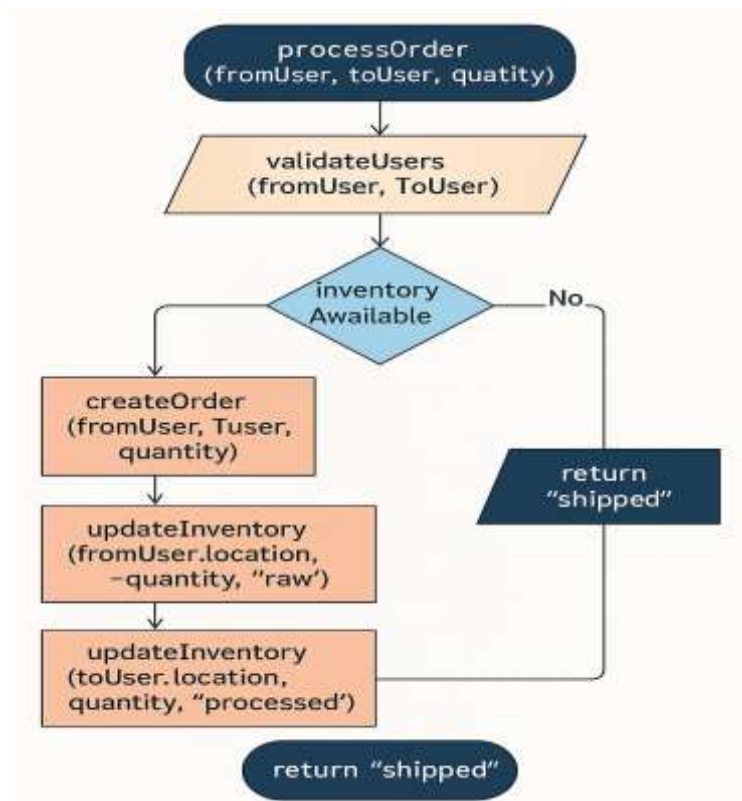


*Figure 4. 2: Showing Order Processing Flow/Logic*

### 4.3.3. Workforce Distribution

Tasks are assigned to factory workers based on production schedules and inventory needs. Admins monitor task completion and reallocate resources as needed.
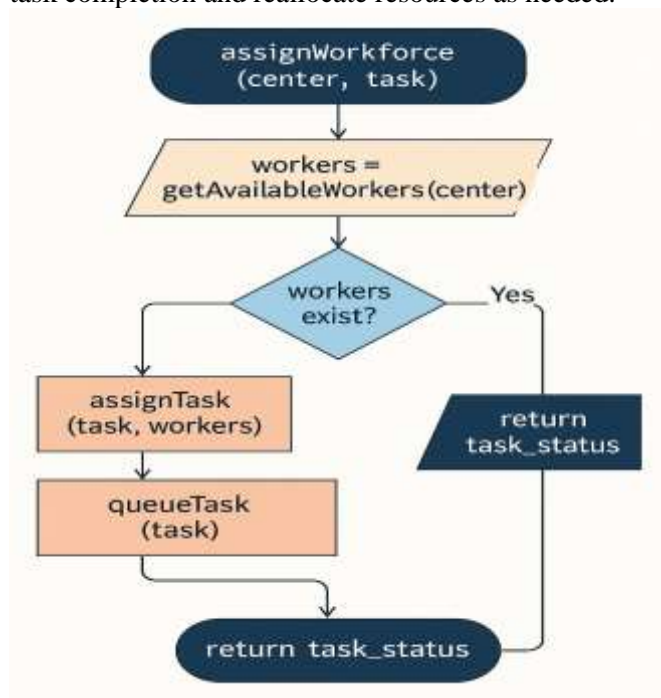


*Figure 4. 3: Showing Workflow Distribution Flow/Logic*

### 4.3.4. ML Analytics

The ML module uses ARIMA for demand forecasting (7–30 days) and K-Means for customer segmentation based on order patterns. Outputs guide production and delivery planning.
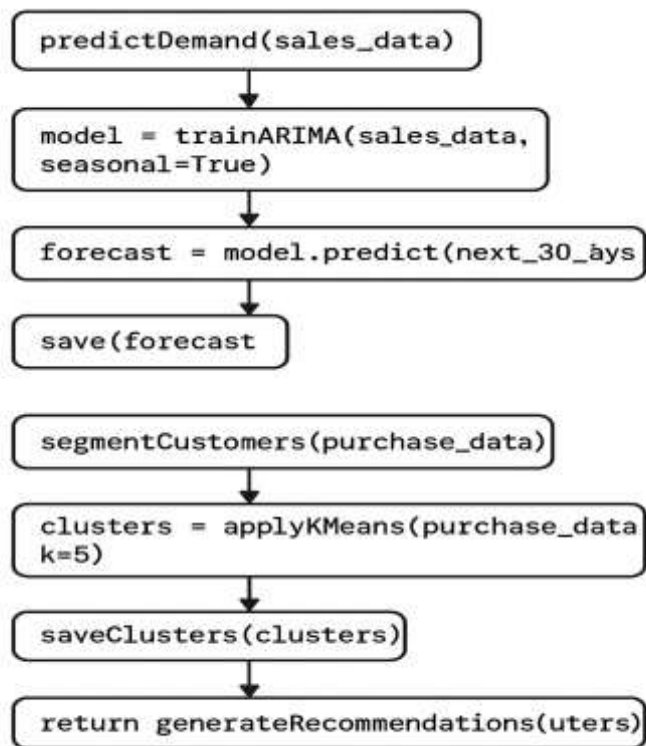
```
predictDemand(sales_data)

model = trainARIMA(sales_data,
seasonal=True)

forecast = model.predict(next_30_ays

save(forecast

segmentCustomers(purchase_data)

clusters = applyKMeans(purchase_data
k=5)

saveClusters(clusters)

return generateRecommendations(uters)
```

*Figure 4. 4: Showing ML Analytics Flow/Logic*

### 4.3.5. Vendor Validation (Java Server)

The Java server processes vendor PDF applications, scoring financial stability, reputation, and compliance. Results are integrated into the Laravel system via a REST API
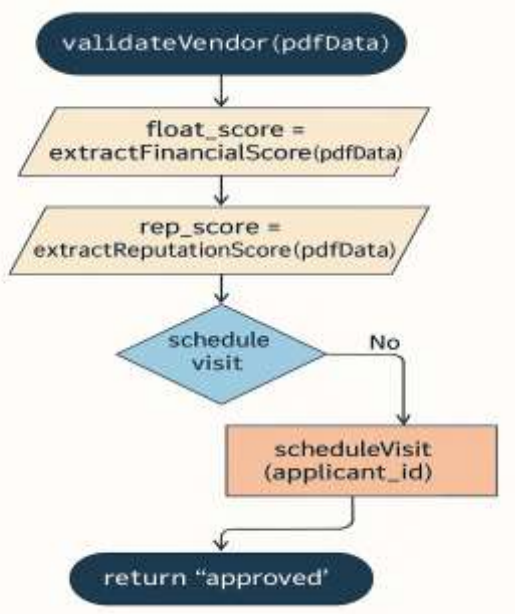
```
validateVendor(pdfData)

float_score =
extractFinancialScore(pdfData)

rep_score =
extractReputationScore(pdfData)

schedule visit ──No──┐
                     │
scheduleVisit
(applicant_id)

return "approved'
```

*Figure 4. 5: Showing Vendor Validation Flow/Logic*

### 4.3.6. Chat Service

Real-time communication is enabled via Laravel Echo and Pusher, allowing stakeholders to coordinate orders, deliveries, and issues.
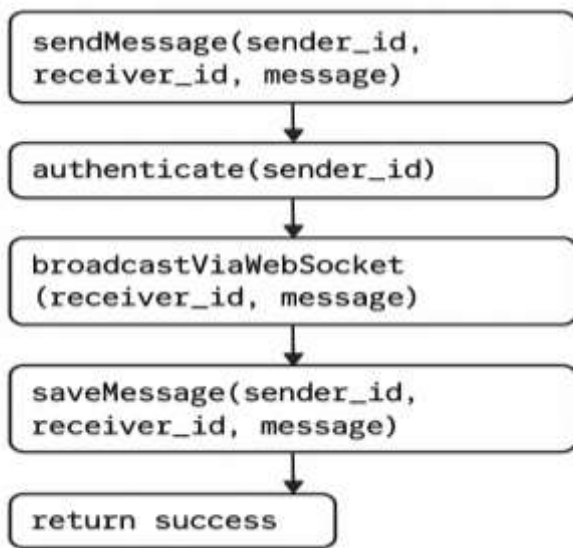


*Figure 4. 6: Showing Chat Service Flow/Logic*

### 4.3.7. Report Scheduling

Reports (demand, inventory, sales) are generated periodically or on-demand for stake- holders. Admins schedule and distribute reports via the dashboard
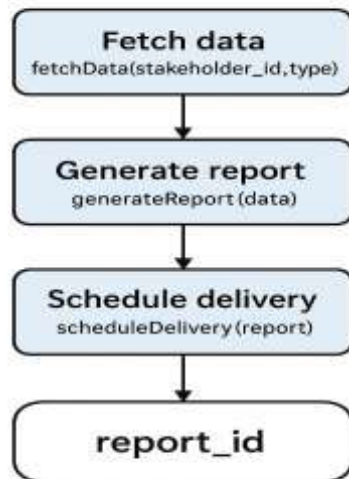


*Figure 4. 7: Showing Report Scheduling Flow/Logic*

# 5. Human Interface Design

## 5.1. Overview

The UI uses Laravel Blade and Tailwind CSS for responsive, role-specific dashboards. Each role (supplier, factory, distributor, retailer, admin) has a customized interface tai- lored to their tasks.

## 5.2.   Screen Objects and Actions

| Screen | Actions |
|---|---|
| Login Page | Enter email/password, authenticate user |
| Supplier Dashboard | View stock, submit harvests, accept orders, chart with factory. |
| Factory Dashboard | Track production, manage orders, chart with suppliers/distributors. |
| Distributor Dashboard | View delivery schedules, update statuses, re- view reports. |
| Retailer Dashboard | Place orders, view inventory, contact distributors. |
| Admin Dashboard | Approve vendors, manage users, and generate/view reports. |
| Vendor Panel | Upload PDF application, track approval status |

*Table 5: Showing Screen Objects and Action*

# 6. Requirements Matrix

| Req. ID | Requirement | Component |
|---|---|---|
| SRS-01 | Predict future demand | ML Analytics |
| SRS-02 | Segment customers | ML Analytics |
| SRS-03 | Real-time chat | Chat Service |
| SRS-04 | Analytics for decisions | ML & Report System |
| SRS-05 | Inventory management | Inventory Module |
| SRS-06 | Order processing | Order Module |
| SRS-07 | Workforce distribution | Workforce Module |
| SRS-08 | Scheduled stakeholder reports | Reporting Module |
| SRS-09 | Vendor validation | Java Server API |

*Table 6: Showing Requirements Matrix*

# 7. Appendices

## 7.1. Appendix A: Data Flow Diagram
Refer to GitHub repository.

## 7.2. Appendix B: Dataset Justification
The system uses the Bakery Supply Chain Data on Kaggle, which includes simulated data for inventory, procurement, distribution, and sales. This dataset is suitable for training ARIMA and K-Means models, ensuring accurate demand forecasting and customer seg- mentation.

## 7.3. Appendix C: Machine Learning Models

- **Demand Prediction:** Time series forecasting using ARIMA and Prophet trained on historical bread sales data (daily or weekly), providing 7- to 30-day demand forecasts to help bakeries optimize production and reduce waste.

- **Customer Segmentation:** K-means clustering applied to customer order frequency, bread type preferences, and purchase size. This enables grouping customers for targeted delivery routes, promotions, and bakery recommendations.