**PROJECT REPORT ON**

**Demand Based Inventory Management System**

**INTEL AI MANUFACTURING CERTIFICATE COURSE**

**Team Members**

|  |  |
| --- | --- |
| 1. | Hitesh Prajapati |
| 2. | Aastha Bhatt |
| 3. | Jay Daiya |
| 4. | Yash Panchal |
| 5. | Jay Patel |

**DEPARTMENT OF**

**COMPUTER ENGINEERING**

**INSTITUTE**

**SARDAR PATEL COLLEGE OF ENGINEERING**

**ABSTRACT**

The demand-based inventory management system is developed as part of the Intel AI for Manufacturing Certificate Program with the aim of solving a real-world problem in the dairy industry. In dairy manufacturing, inventory mismanagement often results in overproduction, understocking, and significant product wastage due to expiry. This project proposes a data-driven solution that uses machine learning to forecast weekly product demand and integrates that intelligence into a web-based inventory tracking system.

An XGBoost regression model was trained to predict the 5-week demand for ten key dairy products using historical sales data and time-based features. The backend of the application is built with FastAPI in Python, the frontend with HTML, CSS, and JavaScript, and the database is managed using MySQL hosted on Railway. The system features batch-level inventory tracking, real-time stock updates, expiry alerts, and automatic reorder suggestions through an intuitive web interface.

The project has achieved a forecast accuracy of around 88%, enabling smarter procurement decisions and improving overall operational efficiency. By proactively balancing supply with predicted demand, the system helps minimize waste, reduce manual tracking efforts, and support continuous availability of essential dairy products.

**TABLE OF CONTENT**

[1. Introduction 5](#_Toc203349189)

[1.1 Project Title 5](#_Toc203349190)

[1.2 Project Description 5](#_Toc203349191)

[1.3 Benefits 5](#_Toc203349192)

[1.4 Risks 5](#_Toc203349193)

[2. Objectives 6](#_Toc203349194)

[2.1 Primary Objective 6](#_Toc203349195)

[2.2 Secondary Objectives 6](#_Toc203349196)

[2.3 Measurable Goals 6](#_Toc203349197)

[3. Methodology 7](#_Toc203349198)

[3.1 Approach 7](#_Toc203349199)

[3.2 Phases 7](#_Toc203349200)

[3.3 Deliverables 7](#_Toc203349201)

[3.4 Testing & Quality Assurance 7](#_Toc203349202)

[3.5 Risk Management 8](#_Toc203349203)

[4. Architecture 9](#_Toc203349204)

[4.1 System Design 9](#_Toc203349205)

[4.2 Block Diagrams 9](#_Toc203349206)

[5. Technologies Used 10](#_Toc203349207)

[5.1 Programming Languages 10](#_Toc203349208)

[5.2 Development Frameworks 10](#_Toc203349209)

[5.3 Database & Tools 10](#_Toc203349210)

[5.4 Development Tools 10](#_Toc203349211)

[5.5 Testing Tools 10](#_Toc203349212)

[5.6 Cloud Services 11](#_Toc203349213)

[5.7 Security 11](#_Toc203349214)

[5.8 APIs and Web Services 11](#_Toc203349215)

[6. Results 12](#_Toc203349216)

[6.1 Key Metrics 12](#_Toc203349217)

[6.2 Screenshots of Implementation 12](#_Toc203349218)

[6.3 Return on Investment (ROI) 13](#_Toc203349219)

[7. Conclusion 14](#_Toc203349220)

[7.1 Recap 14](#_Toc203349221)

[7.2 Key Takeaways 14](#_Toc203349222)

[7.3 Future Plans 14](#_Toc203349223)

[7.4 Successes and Challenges 14](#_Toc203349224)

[8. Project Specifics 15](#_Toc203349225)

1. Introduction

1.1 Project Title

**Demand-Based Inventory Management System**

An AI-powered system that predicts weekly product demand and helps manage perishable inventory in an efficient, data-driven way.

1.2 Project Description

Inventory management in the dairy industry presents critical challenges due to fluctuating demand and short product shelf life. Overstocking leads to waste and financial loss, while understocking causes supply gaps and missed opportunities.

This project introduces a smart inventory management system that uses machine learning to forecast product demand and manage stock levels automatically. The system enables batch-wise inventory tracking, expiry alerts, and intelligent reorder suggestions. It is built using FastAPI for the backend, HTML/CSS/JS for the frontend, and MySQL hosted on Railway for the database.

1.3 Benefits

* **Reduced Inventory Waste:** Accurate demand forecasting helps prevent overstocking and reduces losses due to expired products.
* **Smarter Procurement Planning:** The system enables timely and data-driven purchase decisions based on predicted weekly demand**.**
* **Real-Time Inventory Tracking:** Batch-wise stock tracking ensures visibility of quantity, expiry, and production dates.
* **Automated Alerts:** Low-stock and expiry warnings help prevent shortages and reduce manual monitoring.
* **Expandable Design:** Easily adaptable to support more products or future enhancements.

1.4 Risks

* **Limited Historical Data:** A small or incomplete dataset can reduce the model’s forecasting accuracy.
* **Demand Volatility:** Sudden changes in product demand (seasonal spikes, holidays) may not be captured by the model.
* **Batch Expiry Conflicts:** Complex logic is required to track multiple batch expirations accurately.
* **Scalability:** As the number of products or batches increases, database performance and forecasting speed may become bottlenecks.
* **Overfitting Risk:** The model may perform well on training data but fail to generalize to unseen weeks.

1. Objectives

2.1 Primary Objective

The primary objective of this project is to design and implement a demand-driven inventory management system that leverages machine learning to forecast weekly product demand and optimize inventory levels. The goal is to minimize product wastage, enhance procurement efficiency, and improve inventory decision-making in a structured and automated manner.

2.2 Secondary Objectives

In support of the primary goal, the project also aims to:

* Develop a batch-wise inventory tracking mechanism, including production and expiry date monitoring.
* Implement automated alert systems for low stock levels and approaching expiration dates.
* Provide a user-friendly, responsive web interface for real-time inventory visibility and control.
* Ensure seamless integration of the forecasting model, backend APIs, and database infrastructure.
* Design the system architecture to support future scalability and ease of extension.

2.3 Measurable Goals

To evaluate the success and impact of the project, the following measurable targets were defined:

* Achieve at least **85% forecasting accuracy** for weekly product demand.
* Ensure **inventory update and alert latency under 2 seconds** for a responsive user experience.
* Reduce expired stock losses by **15% to 30%** through proactive planning.
* Maintain **API response time below 600 milliseconds** under average load.
* Deliver a **fully deployed, 24×7 accessible system** using cloud infrastructure.

1. Methodology

3.1 Approach

The project followed an **Agile-inspired development approach**, with clearly defined, iterative stages to manage each core component of the system. Development was broken down into weekly milestones for tasks like data preparation, model training, API development, UI implementation, and deployment. This modular process allowed flexibility in implementation, continuous integration of features, and easier debugging at each stage.

3.2 Phases

The overall development was carried out in the following key phases:

* **Phase 1 – Problem Understanding & Dataset Preparation:** Collected and cleaned product demand data using Microsoft Excel. Identified key features (e.g., product ID, week number) and structured the data for forecasting.
* **Phase 2 – Model Training & Evaluation:** Built a machine learning model using XGBoost regression to predict 5-week product demand. Evaluated the model using standard accuracy metrics and fine-tuned it for performance.
* **Phase 3 – Backend Development (FastAPI):** Designed RESTful APIs to handle inventory updates, forecasting results, sales logging, and product summaries. Connected the model and database with business logic.
* **Phase 4 – Frontend Development (HTML/CSS/JS):** Created a responsive web interface for inventory control, stock visualization, and forecasting charts. Integrated API calls for dynamic data updates.
* **Phase 5 – Integration, Testing & Deployment:** Linked all components and deployed the full-stack application on Railway. Verified UI/API consistency and cloud-based functionality.

3.3 Deliverables

At the completion of each phase, the following deliverables were produced:

* Cleaned dataset (Excel format)
* Trained machine learning model (.pkl file)
* FastAPI backend code and endpoints
* Inventory and forecasting web pages (HTML, CSS, JS)
* Integrated MySQL database schema
* Fully deployed live system (frontend + backend)

3.4 Testing & Quality Assurance

Quality assurance was ensured through manual and functional testing at each stage:

* **Backend Testing:** Verified all FastAPI endpoints using test data inputs, checked for correct JSON responses and status codes.
* **Frontend Testing:** Manually tested forms, modals, scrollable charts, and responsive layout on various screen sizes.
* **Data Testing:** Simulated updates and deletions in the inventory to confirm accurate batch-level handling.
* **Forecast Testing:** Cross-validated forecast output against historical patterns to confirm model behaviour.

3.5 Risk Management

The following key risks were identified and addressed during the development of the project:

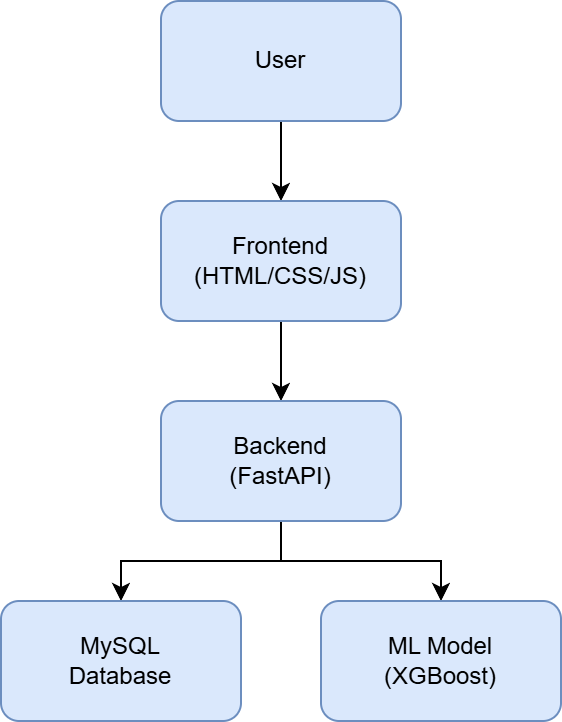
* **Limited Historical Data:** A small dataset could affect forecast accuracy. This was managed through careful data cleaning and model validation.
* **Model Overfitting:** Overfitting was avoided using regularization techniques and testing on unseen data.
* **Integration Challenges:** API responses were standardized to ensure smooth communication between backend and frontend.
* **Batch Handling Complexity:** Batch expiry and quantity tracking were managed with strict validation and logical checks.
* **Responsive UI Issues:** Manual testing and media queries were used to ensure layout compatibility across devices.
* **Cloud Hosting Constraints:** Potential cold starts or delays on free-tier hosting were reduced by optimizing API performance.

1. Architecture

4.1 System Design

* The system follows a **layered modular architecture**, combining frontend, backend, machine learning, and database components in a well-defined workflow.
* The **User Interface** (UI), developed with HTML, CSS, and JavaScript, serves as the interaction layer where users view stock summaries, forecast graphs, and perform inventory actions.
* The **Backend Layer**, built using FastAPI (Python), acts as the intermediary between the UI, the machine learning model, and the database. It exposes RESTful API endpoints to handle stock updates, forecast queries, and product summaries.
* The **Machine Learning Module** integrates a trained XGBoost regression model loaded by the backend. It receives input features (like product ID and week number) and returns 5-week demand predictions.
* The **Database Layer**, built on MySQL, stores all core data: product info, batch-wise inventory (with expiry/production dates), and logged sales. All inventory operations are validated and tracked here.
* The entire system is deployed on **Railway**, allowing seamless cloud integration, live accessibility, and continuous API-model-database communication.

4.2 Block Diagrams



1. Technologies Used

The development of this demand-based inventory management system involved a combination of programming languages, frameworks, tools, and services to ensure scalability, usability, and performance. Each component was carefully selected to suit the system's real-time requirements and cloud deployment goals.

5.1 Programming Languages

* **Python:** Used for backend development and machine learning model implementation. FastAPI was built entirely in Python.
* **JavaScript:** Enabled dynamic interactivity and API communication on the frontend.
* **HTML & CSS:** Structured and styled the responsive user interface.

5.2 Development Frameworks

* **FastAPI:** A modern Python web framework used to build backend APIs for inventory, forecasting, and stock updates.
* **XGBoost:** A powerful regression-based machine learning library used to forecast weekly product demand.
* **Chart.js (or Custom SVG/JS Logic):** Used for visualizing demand and stock levels through graphs.

5.3 Database & Tools

* **MySQL:** Used to store all core data including products, inventory batches, and sales records. The schema was designed to support batch-level tracking and expiry management.
* **MySQL Workbench:** Served as the primary tool for database design, testing, and remote connection with the Railway MySQL instance.

5.4 Development Tools

* **Visual Studio Code (VS Code):** The main IDE used for writing and testing both backend and frontend code.
* **Git & GitHub:** Version control system and repository used to manage source code collaboratively and track changes.

5.5 Testing Tools

* **Browser Developer Tools (Chrome/Edge):** Used for UI debugging, responsive design testing, and API request validation.
* **Postman:** Used for manually testing RESTful API endpoints and verifying JSON responses during backend development.

5.6 Cloud Services

* **Railway:** Used to host both the FastAPI backend and the MySQL database. The platform provides deployment, live hosting, and remote access with minimal configuration.

5.7 Security

* **Input Validation:** Ensured clean and secure data handling across frontend forms and backend routes.
* **Structured API Access:** Routes were designed to prevent invalid or unauthorized operations through strict parameter handling.
* **Database Integrity Checks:** Inventory updates, deletions, and forecasting requests were validated before affecting the database.

5.8 APIs and Web Services

* **Custom RESTful APIs:** Built using FastAPI to handle forecasting, product summary, stock updates, and sales logging.
* **Internal ML Model Service:** The XGBoost model is served internally by the backend without exposing it publicly, ensuring data privacy and controlled access.

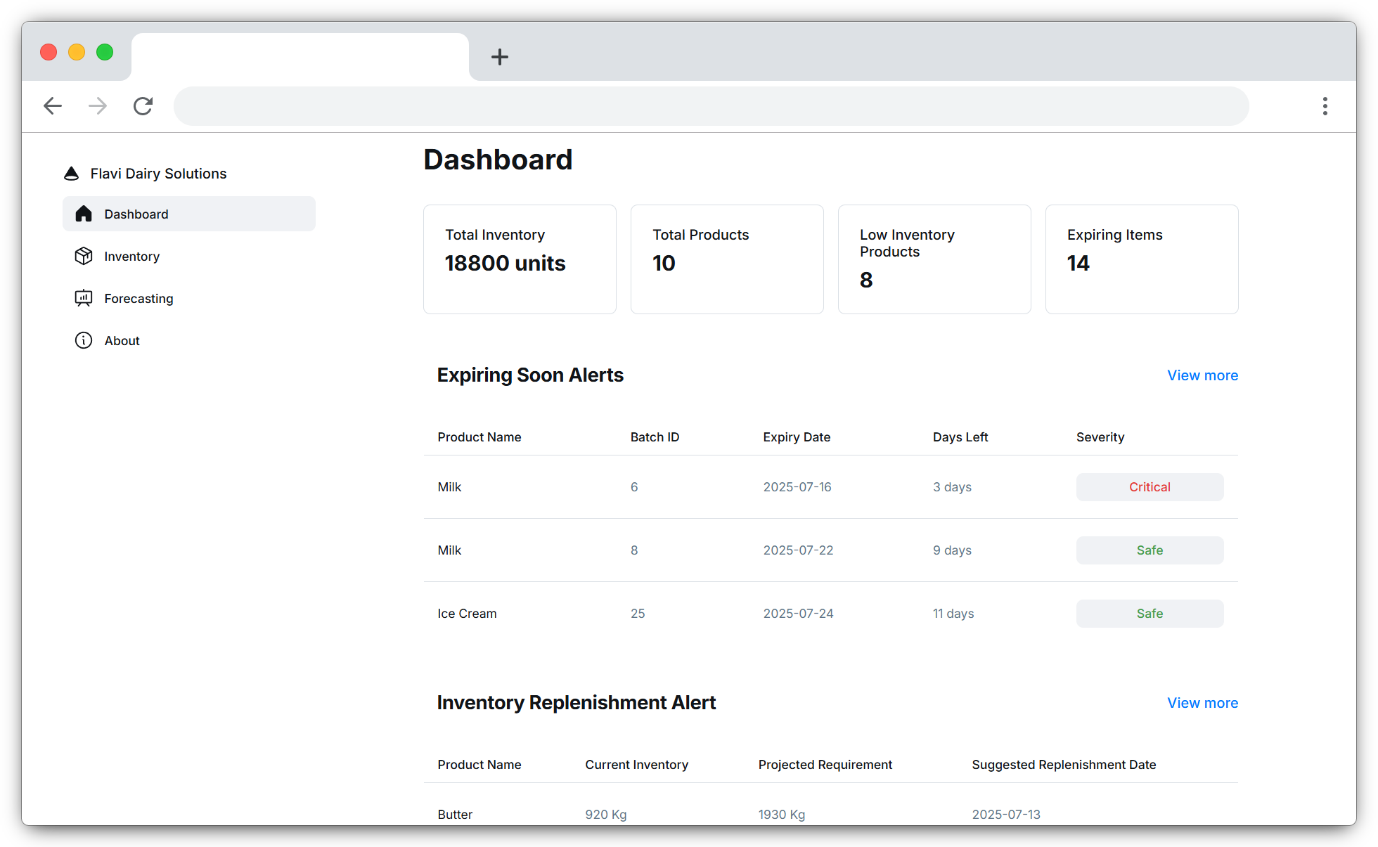
1. Results

This section outlines the performance of the system based on defined success metrics and highlights how the project delivers tangible operational benefits. The implemented solution effectively met its objectives in terms of forecasting accuracy, system performance, and usability.

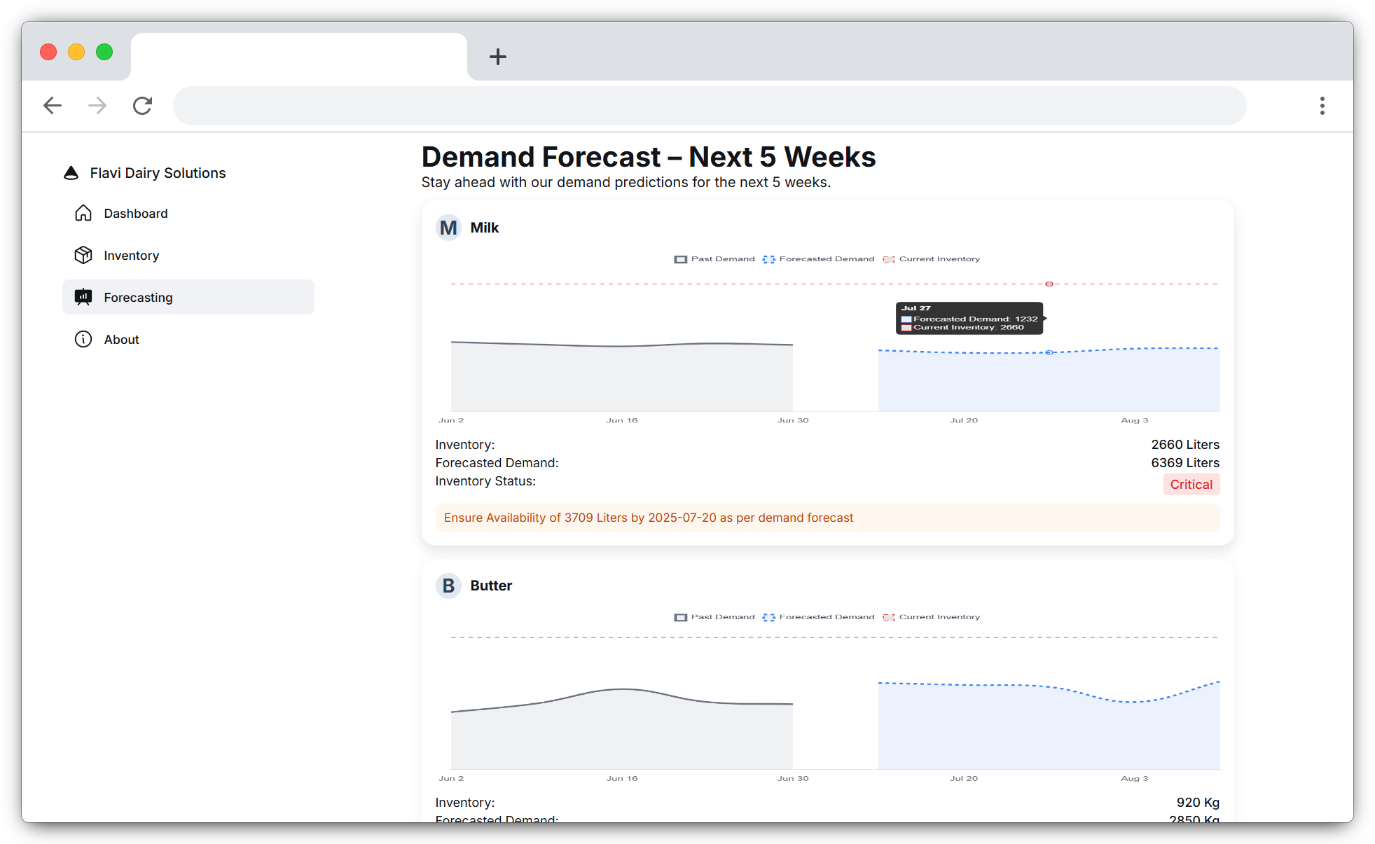
6.1 Key Metrics

* **Forecast Accuracy:** The XGBoost model achieved an average forecasting accuracy of approximately **88%**, based on evaluation against historical demand data.
* **System Responsiveness:** API calls for stock updates and forecast retrieval consistently returned responses within **400–600 milliseconds** under average load.
* **Inventory Update Time:** Stock additions and sales adjustments reflected in the system within **2 seconds**, enabling near real-time tracking.
* **UI Load Time:** The frontend interface loaded fully in under **2.5 seconds**, ensuring a smooth user experience across devices.
* **Data Consistency:** Batch-level expiry and quantity tracking were successfully validated and updated across inventory and sales tables without error.

6.2 Screenshots of Implementation



*Figure 1: Dashboard*



*Figure 2: Forecasting*

6.3 Return on Investment (ROI)

While not calculated in monetary terms, the system offers significant operational value:

* **Reduced Stock Wastage:** Accurate forecasting and expiry tracking help prevent overstocking, potentially reducing expired inventory by **15–30%**.
* **Improved Planning Efficiency:** Automated demand insights eliminate the need for manual estimation, reducing planning time and human error.
* **Scalability Without Extra Cost:** The cloud-hosted design supports future growth without additional infrastructure or licensing costs.
* **24×7 System Availability:** Deployment on Railway enables continuous access and monitoring without the need for internal hosting resources.

1. Conclusion

7.1 Recap

This project successfully delivered a demand-based inventory management system that combines machine learning with real-time batch tracking and inventory control. The primary goal was to forecast weekly product demand for perishable dairy items and automate procurement-related decisions. The system integrates a FastAPI backend, an XGBoost forecasting model, a MySQL database, and a responsive frontend—deployed as a full-stack solution on Railway.

7.2 Key Takeaways

* Machine learning–based demand forecasting can significantly reduce inventory waste in supply chain–sensitive sectors like dairy.
* Batch-level inventory tracking with expiry management improves product flow and minimizes loss.
* Modular development with clear API integration ensures scalability and maintainability.
* UI responsiveness, even with dynamic data, is critical for a smooth user experience.
* Proper dataset preparation and validation are key to building reliable ML models.

7.3 Future Plans

* **User Authentication:** Add role-based login (e.g., admin, operator) for secure access.
* **Live Sales Tracking:** Automate sales entry and update forecasts based on consumption.
* **Mobile Optimization:** Improve layout and performance for mobile-first users.
* **Notification System:** Add push/email alerts for reorder and expiry reminders.
* **Extended Forecasting:** Support for long-term (monthly/seasonal) demand predictions.

7.4 Successes and Challenges

**Successes:**

* Achieved ~88% forecast accuracy using XGBoost.
* Fully deployed cloud-based system with live API/database integration.
* Real-time batch-level stock management with expiry validation.

**Challenges:**

* Solved repeated week number issue by mapping actual forecast dates.
* Addressed layout issues in scrollable graphs for better UX.
* Handled API-frontend sync errors through structured endpoint testing.

1. Project Specifics

**Project URL:**  
[*https://demand-based-inventory-management-system-production.up.railway.app/*](https://demand-based-inventory-management-system-production.up.railway.app/)   
*A live version of the full-stack inventory management system is deployed on Railway and accessible via the link above.*

**GitHub URL:**  
[*https://github.com/Hitesh4011/Demand-Based-Inventory-Management-System*](https://github.com/Hitesh4011/Demand-Based-Inventory-Management-System)  
*The complete source code, including frontend, backend, and model integration, is publicly available in this repository.*

**Dataset URL:**  
[*https://www.kaggle.com/datasets/suraj520/dairy-goods-sales-dataset*](https://www.kaggle.com/datasets/suraj520/dairy-goods-sales-dataset)   
*The cleaned Excel dataset used for training the forecasting model is available for reference and download.*