

Florida International University
Knight Foundation School of Computing and Information Sciences

Computer Science Focus

Final Deliverable

Project Title: AI Driven Warehouse Stock Management

Team Members: Angela Banov, Matt Cobos, Momin Salar Khan, Steven Palacios, Vianne Novo

Product Owner(s): Ravi Venugopal

Mentor(s): Dinesh Cyril Selvaraj and Sriram Anbu

Instructor: Masoud Sadjadi

Abstract

This Final Deliverable document provides a comprehensive overview of the development process for the AI-Driven Warehouse Stock Management platform, which was designed to help organizations monitor inventory, prevent shortages, and redistribute stock across multiple warehouses. Traditional ERP and WMS systems do not solve the issue of dynamically balancing stock levels or identifying where excess inventory can be repurposed. Our platform addresses this challenge by providing a user-friendly interface that visualizes warehouse conditions and supports informed decision-making.

The development process entailed implementing structured sprint planning, daily scrum meetings, and product owner feedback sessions using user stories to guide feature implementation. The system incorporates modern technologies including FastAPI for backend services, PostgreSQL for operational data storage, and Streamlit for the frontend dashboard, alongside a separate ledger service used to simulate live stock events. Following agile methodologies, the project included iterative testing, data validation, and continuous refinement to ensure alignment with user requirements and realistic warehouse operations. This document will also include our sprint review reports, which highlight the collaborative nature of the project and the foundation established for future agent-driven optimization.

Table of Contents

<i>Introduction.....</i>	<i>5</i>
Current System.....	5
Purpose of New System	5
Implemented User Stories	6
Pending User Stories.....	7
<i>Project Plan</i>	<i>7</i>
Hardware and Software Resources	7
Sprints Plan	8
Sprint 1	8
Sprint 2	10
Sprint 3	13
Sprint 4	15
Sprint 5	18
Sprint 6	21
Sprint 7	23
<i>System Design</i>	<i>26</i>
Architectural Patterns.....	26
System and Subsystem Decomposition.....	26
Deployment Diagram.....	27
<i>Glossary</i>	<i>28</i>
<i>Appendix.....</i>	<i>30</i>
Appendix A - UML Diagram.....	30
Appendix B - Sprint Review Reports	32
Review Meeting Minutes (Sprint 1)	32
Review Meeting Minutes (Sprint 2)	32
Review Meeting Minutes (Sprint 3)	33
Review Meeting Minutes (Sprint 4)	34
Review Meeting Minutes (Sprint 5)	35
Review Meeting Minutes (Sprint 6)	36
Review Meeting Minutes (Sprint 7)	37

Appendix C - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents	38
<i>Installation Guide</i>	<i>38</i>
<i>Pre-Installation Requirements</i>	<i>38</i>
<i>Installation Process</i>	<i>39</i>

INTRODUCTION

Warehouse networks rely on continuous movement of inventory to satisfy demand across multiple distribution centers. When one location holds excess product while another experiences shortages, operational performance declines. Traditional enterprise systems such as ERP and WMS track static inventory records, but they do not provide proactive transfers, prioritization, or automated balancing. In real business environments, teams must manually analyze tables, dashboards, and spreadsheets to determine which unit is available next and where it should ship from, which is slow, reactive, and error prone.

The AI Driven Warehouse Stock Management platform was created to establish the foundation for an intelligent inventory balancing system. The solution integrates a persistent operational database, a scalable backend, and an interactive dashboard that shows inventory conditions across multiple warehouses. Unlike static systems, it is designed to evolve into an agent-driven platform that continuously monitors stock levels, forecasts demand and recommends transfers with high accuracy.

Current System

Companies like Giggso frequently operate multiple warehouse locations without a centralized decision-support layer. Their ERP and WMS systems can show raw stock counts but cannot determine the next available unit, the fastest warehouse to ship from, or which distribution center is overstocked or at risk of stockout. Users must interpret this information manually, which results in delayed redistribution decisions, underutilization of surplus inventory, and reduced fulfillment reliability. There is no automated method to suggest transfers or balance stock across facilities in real time.

Purpose of New System

The objective of the new system is to provide a platform that enables automated situational awareness across warehouse networks. Instead of requiring operators to interpret raw records, the system visualizes current stock levels, compares conditions between locations, and presents an early-stage transfer recommendation. The platform acts as the groundwork for an intelligent

assistant that will monitor events, detect imbalance, and ultimately propose redistribution actions with high confidence. This architecture is intended to replace manual decision making with proactive inventory balancing supported by real data.

User Stories

User stories were developed in collaboration with our product owners to capture the expectations of warehouse operations personnel and supply chain analysts. Stories were prioritized based on feasibility, impact, and alignment with agent-driven goals. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

- User Story #1: Generate Synthetic Data – As a user, I want data from the warehouse, SKU, and order data so that I can test the system without needed real data
- User Story #2: Basic Inventory Dashboard – As a user, I want a simple dashboard showing stock levels across warehouses so that I can monitor inventory at a glance.
- User Story #3: Connect Dashboard to Backend (API) - As a user, I want the dashboard to pull data through the API so that it mimics real-time system behavior.
- User Story #5: Forecasting Setup - As a user, I want a forecasting function so that I can see which items might run low soon.
- User Story #6: Low Stock Alert – As a user, I want a dashboard to highlight low stock SKUs so that I can quickly see which products need replenishment.
- User Story # 8: Forecasting Expansion (Multi-SKU with Warehouse Segmentation) - As a user, I want to forecast demand for multiple SKUs at the warehouse level so I can anticipate stockouts across different locations.
- User Story #10: Transactions - As a user, I want to log adjustments so that inventory levels are always accurate.
- User Story #11: AI Integration for Predictive Stock Insight – As a user, I want the system to generate AI-driven recommendations for restocking and demand forecasting so that I can make smarter, data informed inventory decisions.
- User Story #12: Deployment Pipeline Setup – As a developer, I want to deploy our FastAPI and Streamlit applications to a cloud platform so that the system can be tested and accessed externally for demonstration and showcase purposes.

- User Story #13: System Testing and Documentation – As a developer, I want to implement structured testing and update project documentation so that the system is stable, maintainable, and easy to understand for new contributors.
- User Story #14: UI Polish and Role Optimization – As a user, I want a clearer and more responsive dashboard so that experience feels professional and ready for showcase presentation.
- User Story #15: Full System Testing and Quality Assurance – As a developer, I want to conduct comprehensive testing on all modules so that the final system is stable, accurate, and ready for presentation.
- User Story #16: End-to-End Integration and Performance Enhancement – As a developer, I want the front-end, back-end, and AI components fully integrated and optimized so that the app performs efficiently and reliably for the Capstone showcase.
- User Story #17: Showcase Preparation and Final Documentation – As a developer, I want to prepare all documentation visuals, and demonstration flow so that our Capstone presentation is polished, engaging, and easy to follow.

Pending User Stories

- User Story #4: Stock Ledger Agent – As a user, I want to provide inventory updates so that other agents and the dashboard can see current stock levels
- User Story # 7: Inventory Transactions – As a user, I want to log stock adjustments, such as receiving, shipping, and transfers, so that inventory levels are always accurate.
- User Story #9: Role Based Dashboard Views – As a user, I want role-based dashboards so that only I can see features relevant to me.

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Hardware:

- Local developer laptops (Apple M-series and Windows x86)
- Dockerized containers running locally

Software Stack:

- Python 3.11
- FastAPI — REST backend
- PostgreSQL — operational database
- SQLAlchemy ORM
- Streamlit — visualization UI
- Requests module — HTTP communication
- Docker and Docker Compose
- VS Code + GitHub for version control
- Lucidchart — UML and system diagrams
- Pydantic — schema validation

Sprints Plan

Sprint 1
Sprint Planning Meeting Minutes

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 11:00 am

End time: 12:00 pm

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Generate Synthetic Data

- As an Inventory Planner, I want a synthetic warehouse, SKU, and order data so that I can test the system without needing real data.
- Acceptance Criteria:
 - Python scripts generate CSVs for warehouses, SKUs, customers, and orders.
 - Data looks realistic (IDs, names, locations, quantities).
 - CSVs can be easily loaded into the dashboard/backend.

User Story 2: Basic Inventory Dashboard

- As an E-commerce Manager, I want a simple dashboard showing stock levels across warehouses so that I can monitor inventory at a glance.
- Acceptance Criteria:
 - Streamlit app displays inventory levels from synthetic data.
 - Includes at least one chart (e.g., bar chart of stock by warehouse).
 - Shows basic KPIs (e.g., total SKUs, average Days on Hand).

User Story 3: Connect Dashboard to Backend API (Mock)

- As a Warehouse Manager, I want the dashboard to pull data through an API so that it mimics real-time system behavior.
- Acceptance Criteria:
 - Streamlit fetches data from a mock FastAPI endpoint.
 - API returns inventory data in JSON format.
 - Dashboard updates when API data changes.

User Story 4: Stock Ledger Agent (Stub)

- As a Stock Ledger Agent, I want to provide inventory updates so that other agents and the dashboard can see current stock levels
- Acceptance Criteria:
 - Basic FastAPI service runs as the Stock Ledger Agent.
 - Responds with current stock levels from synthetic data.
 - Registered with the MCP server for future communication.

User Story 5: Forecasting Setup (Initial Model)

- As an Inventory Planner, I want a forecasting function so that I can see which items might run low soon.
- Acceptance Criteria:
 - Use Prophet or a simple model to forecast demand for one SKU.
 - Show forecast results in a table or simple chart.
 - Runs on synthetic order data.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 1: Generate Synthetic Data

Vianne Novo Grifol - User Story 4: Stock Ledger Agent (Stub)

Steven S. Palacios - User Story 2: Basic Inventory Dashboard

Matthew Cobos - User Story 3: Connect Dashboard to Backend API (Mock)

Momin Salar Khan - User Story 5: Forecasting Setup (Initial Model)

Sprint 2 **Sprint Planning Meeting Minutes**

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 9:00 am

End time: 10:00 am

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Basic Inventory Dashboard

- As an E-commerce Manager, I want a simple dashboard showing stock levels across warehouses so that I can monitor inventory at a glance.
- Acceptance Criteria:
 - Streamlit app displays inventory levels from synthetic data.
 - Includes at least one chart (e.g., bar chart of stock by warehouse).
 - Shows basic KPIs (e.g., total SKUs, average Days on Hand).

User Story 2: Connect Dashboard to Backend API (Mock)

- As a Warehouse Manager, I want the dashboard to pull data through an API so that it mimics real-time system behavior.
- Acceptance Criteria:
 - Streamlit fetches data from a mock FastAPI endpoint.
 - API returns inventory data in JSON format.
 - Dashboard updates when API data changes.

User Story 3: Stock Ledger Agent (Stub)

- As a Stock Ledger Agent, I want to provide inventory updates so that other agents and the dashboard can see current stock levels

- Acceptance Criteria:
 - Basic FastAPI service runs as the Stock Ledger Agent.
 - Responds with current stock levels from synthetic data.
 - Registered with the MCP server for future communication.

User Story 4: Forecasting Setup (Initial Model)

- As an Inventory Planner, I want a forecasting function so that I can see which items might run low soon.
- Acceptance Criteria:
 - Use Prophet or a simple model to forecast demand for one SKU.
 - Show forecast results in a table or simple chart.
 - Runs on synthetic order data.

User Story 5: Low-Stock Alerts

- As a Warehouse Manager, I want the dashboard to highlight low-stock SKUs so that I can quickly see which products need replenishment.
- Acceptance Criteria:
 - Define a threshold (e.g., `on_hand < 100` units).
 - Dashboard shows a red alert banner or table of low-stock items.
 - Alerts update automatically from the CSV (or API when connected).
 - Must include: SKU ID, name, warehouse, and current stock.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: Low-Stock Alerts

Vianne Novo Grifol - User Story 3: Stock Ledger Agent (Stub)

Steven S. Palacios - User Story 1: Basic Inventory Dashboard

Matthew Cobos - User Story 2: Connect Dashboard to Backend API (Mock)

Momin Salar Khan - User Story 4: Forecasting Setup (Initial Model)

Sprint 3
Sprint Planning Meeting Minutes

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 8:00 pm

End time: 9:00 pm

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Connect Dashboard to Backend API (Mock)

- As a Warehouse Manager, I want the dashboard to pull data through an API so that it mimics real-time system behavior.
- Acceptance Criteria:
 - Streamlit fetches data from a mock FastAPI endpoint.
 - API returns inventory data in JSON format.
 - Dashboard updates when API data changes.

User Story 2: Stock Ledger Agent (Stub)

- As a Stock Ledger Agent, I want to provide inventory updates so that other agents and the dashboard can see current stock levels
- Acceptance Criteria:
 - Basic FastAPI service runs as the Stock Ledger Agent.
 - Responds with current stock levels from synthetic data.
 - Registered with the MCP server for future communication.

User Story 3: Inventory Transactions (Adjustments & Movements)

- As a Warehouse Clerk, I want to log stock adjustments (receiving, shipping, transfers) so that inventory levels are always accurate.
- Acceptance Criteria:
 - Add a Streamlit form to input transactions (SKU, qty, type, warehouse).
 - Dashboard updates instantly after a transaction.
 - Transactions saved in a CSV or via mock API.
 - Transaction log table with timestamp, SKU, qty, and type.

User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)

- As an Inventory Planner, I want to forecast demand for multiple SKUs at the warehouse level so I can anticipate stockouts across different locations.
- Acceptance Criteria:
 - Extend forecasting to handle multiple SKUs × warehouses at once (not just global SKUs).
- Display results in interactive line charts where users can filter by SKU or warehouse.
- Highlight SKUs projected to fall below threshold per warehouse, not just globally.
- Forecasting runs on synthetic historical order data for at least 3 months per SKU.
- Include a downloadable CSV of forecasts with columns: SKU, warehouse, forecasted demand, projected stockout date.

User Story 5: Role-Based Dashboard Views

- As a System User, I want role-based dashboards (Manager, Planner, Clerk) so I only see features relevant to me.
- Acceptance Criteria:
 - Sidebar dropdown or login role selector.
 - Manager view: global KPIs + low-stock alerts.

- Planner view: forecasting + replenishment suggestions.
- Clerk view: transaction entry + log.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: Role-Based Dashboard Views

Vianne Novo Grifol - User Story 2: Stock Ledger Agent (Stub)

Steven S. Palacios - User Story 3: Inventory Transactions (Adjustments & Movements)

Matthew Cobos - User Story 1: Connect Dashboard to Backend API (Mock)

Momin Salar Khan - User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)

Sprint 4 **Sprint Planning Meeting Minutes**

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 9:00 am

End time: 10:00 am

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Inventory Transactions (Adjustments & Movements)

- As a Warehouse Clerk, I want to log stock adjustments (receiving, shipping, transfers) so that inventory levels are always accurate.
- Acceptance Criteria:
 - Add a Streamlit form to input transactions (SKU, qty, type, warehouse).
 - Dashboard updates instantly after a transaction.
 - Transactions saved in a CSV or via mock API.
 - Transaction log table with timestamp, SKU, qty, and type.

User Story 2: AI Integration for Predictive Stock Insights

- As a Warehouse Manager, I want the system to generate AI-driven recommendations for restocking and demand forecasting so that I can make smarter, data-informed inventory decisions.
- Acceptance Criteria:
 - Integrate an AI model (e.g., linear regression or Prophet) to predict demand based on past order data.
 - Display top 3 AI recommendations in a “Smart Suggestions” panel on the dashboard.
 - Ensure recommendations update dynamically as new transactions are logged.
 - Include an “AI Insights” label or icon for transparency.

User Story 3: Deployment Pipeline Setup

- As a Developer, I want to deploy our FastAPI and Streamlit applications to a cloud platform so that the system can be tested and accessed externally for demo and showcase purposes.
- Acceptance Criteria:
 - Choose a reliable cloud deployment option (e.g., GCP, Streamlit Cloud, or Render).

- Configure FastAPI backend and Streamlit frontend to connect seamlessly in production.
- Document all deployment steps clearly in the README.
- Application must run without local dependencies or manual file path fixes.

User Story 4: System Testing and Documentation

- As a Quality Assurance Engineer, I want to implement structured testing and update project documentation so that the system is stable, maintainable, and easy to understand for new contributors.
- Acceptance Criteria:
 - Implement at least 5 unit tests for core modules (API endpoints, data processing, AI predictions).
 - Create integration tests ensuring frontend–backend communication works properly.
 - Update the README with setup, usage, and testing instructions.
 - Document known issues and improvement suggestions for Sprint 6.

User Story 5: UI Polish and Role Optimization

- As a System User, I want a cleaner and more responsive dashboard interface so that the experience feels professional and ready for showcase presentations.
- Acceptance Criteria:
 - Improve dashboard layout consistency and spacing using Streamlit columns and containers.
 - Add responsive elements (charts resize, text scales properly).
 - Refine color palette and typography for better readability.
 - Ensure role-based views load efficiently without refresh delays.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: UI Polish and Role Optimization

Vianne Novo Grifol - User Story 4: System Testing and Documentation

Steven S. Palacios - User Story 1: Inventory Transactions (Adjustments & Movements)

Matthew Cobos - User Story 3: Deployment Pipeline Setup

Momin Salar Khan - User Story 2: AI Integration for Predictive Stock Insights

Sprint 5 **Sprint Planning Meeting Minutes**

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 9:00 am

End time: 10:00 am

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Inventory Transactions (Adjustments & Movements)

- As a Warehouse Clerk, I want to log stock adjustments (receiving, shipping, transfers) so that inventory levels are always accurate.
- Acceptance Criteria:

- Add a Streamlit form to input transactions (SKU, qty, type, warehouse).
- Dashboard updates instantly after a transaction.
- Transactions saved in a CSV or via mock API.
- Transaction log table with timestamp, SKU, qty, and type.

User Story 2: AI Integration for Predictive Stock Insights

- As a Warehouse Manager, I want the system to generate AI-driven recommendations for restocking and demand forecasting so that I can make smarter, data-informed inventory decisions.
- Acceptance Criteria:
 - Integrate an AI model (e.g., linear regression or Prophet) to predict demand based on past order data.
 - Display top 3 AI recommendations in a “Smart Suggestions” panel on the dashboard.
 - Ensure recommendations update dynamically as new transactions are logged.
 - Include an “AI Insights” label or icon for transparency.

User Story 3: Deployment Pipeline Setup

- As a Developer, I want to deploy our FastAPI and Streamlit applications to a cloud platform so that the system can be tested and accessed externally for demo and showcase purposes.
- Acceptance Criteria:
 - Choose a reliable cloud deployment option (e.g., GCP, Streamlit Cloud, or Render).
 - Configure FastAPI backend and Streamlit frontend to connect seamlessly in production.
 - Document all deployment steps clearly in the README.
 - Application must run without local dependencies or manual file path fixes.

User Story 4: System Testing and Documentation

- As a Quality Assurance Engineer, I want to implement structured testing and update project documentation so that the system is stable, maintainable, and easy to understand for new contributors.
- Acceptance Criteria:
 - Implement at least 5 unit tests for core modules (API endpoints, data processing, AI predictions).
 - Create integration tests ensuring frontend–backend communication works properly.
 - Update the README with setup, usage, and testing instructions.
 - Document known issues and improvement suggestions for Sprint 6.

User Story 5: UI Polish and Role Optimization

- As a System User, I want a cleaner and more responsive dashboard interface so that the experience feels professional and ready for showcase presentations.
- Acceptance Criteria:
 - Improve dashboard layout consistency and spacing using Streamlit columns and containers.
 - Add responsive elements (charts resize, text scales properly).
 - Refine color palette and typography for better readability.
 - Ensure role-based views load efficiently without refresh delays.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: UI Polish and Role Optimization

Vianne Novo Grifol - User Story 4: System Testing and Documentation

Steven S. Palacios - User Story 1: Inventory Transactions (Adjustments & Movements)

Matthew Cobos - User Story 3: Deployment Pipeline Setup

Momin Salar Khan - User Story 2: AI Integration for Predictive Stock Insights

Sprint 6
Sprint Planning Meeting Minutes

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 9:00 pm

End time: 10:00 pm

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Deployment Pipeline Completion and Optimization

- As a Developer, I want to successfully deploy both the FastAPI backend and Streamlit frontend to a reliable cloud platform so that the system can be publicly accessed, tested, and demonstrated seamlessly.
- Acceptance Criteria:
 - Deploy backend and frontend using GCP or Streamlit Cloud with proper connection endpoints.
 - Verify all environment variables, dependencies, and APIs load correctly in production.
 - Fix previous deployment errors (missing paths, API URL mismatches).
 - Add the deployment section with screenshots or URLs in the README.

- Application must be fully functional without local file dependencies.

User Story 2: Full System Testing and Quality Assurance

- As a Quality Assurance Engineer, I want to conduct comprehensive testing on all modules so that the final system is stable, accurate, and ready for presentation.
- Acceptance Criteria:
 - Implement at least 8 unit tests across API, AI, and data handling functions.
 - Perform integration and regression tests between backend and frontend.
 - Use pytest or unittest framework for reproducibility.
 - Document all test results and known issues in a “Testing Summary” section.
 - Achieve at least 90% test coverage before closing this story.

User Story 3: UI Polish and Role-Based Optimization

- As a System User, I want an improved and more intuitive dashboard interface so that the experience looks professional and responsive during demos.
- Acceptance Criteria:
 - Optimize dashboard layout using Streamlit’s columns, containers, and expanders.
 - Refine color scheme, font sizes, and chart alignment for readability.
 - Add role-based filtering or toggle views (e.g., Admin vs. Clerk).
 - Ensure all visuals adjust dynamically on different devices (desktop/tablet).
 - Include branding elements such as project logo or banner for presentation.

User Story 4: End-to-End Integration and Performance Enhancement

- As a Product Owner, I want the frontend, backend, and AI components fully integrated and optimized so that the app performs efficiently and reliably for the Capstone Showcase.
- Acceptance Criteria:
 - Connect Streamlit UI to live API endpoints and AI modules with no broken calls.
 - Monitor latency and optimize loading speed (under 2 seconds).
 - Create and test real transaction flows (inventory → AI prediction → dashboard update).
 - Debug and fix any API response or data sync issues.
 - Verify system consistency across devices and browsers.

User Story 5: Showcase Preparation and Final Documentation

- As a Team Presenter, I want to prepare all documentation, visuals, and demo flow so that our Capstone presentation is polished, engaging, and easy to follow.
- Acceptance Criteria:
 - Finalize README with setup, usage, and testing instructions.
 - Create a final slide deck and demo walkthrough script.
 - Include screenshots or short demo video links in documentation.
 - Assign presenter roles and rehearse demo sequence.
 - Conduct one internal mock presentation and gather feedback.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: Showcase Preparation and Final Documentation

Vianne Novo Grifol - User Story 2: Full System Testing and Quality Assurance

Steven S. Palacios - User Story 4: End-to-End Integration and Performance Enhancement

Matthew Cobos - User Story 1: Deployment Pipeline Completion and Optimization

Momin Salar Khan - User Story 3: UI Polish and Role-Based Optimization

Sprint 7 **Sprint Planning Meeting Minutes**

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 9:00 am

End time: 10:00 am

After discussion, the velocity of the team was estimated to be 25 points (5 points per user story)

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1: Deployment Pipeline Completion and Optimization

- As a Developer, I want to successfully deploy both the FastAPI backend and Streamlit frontend to a reliable cloud platform so that the system can be publicly accessed, tested, and demonstrated seamlessly.
- Acceptance Criteria:
 - Deploy backend and frontend using GCP or Streamlit Cloud with proper connection endpoints.
 - Verify all environment variables, dependencies, and APIs load correctly in production.
 - Fix previous deployment errors (missing paths, API URL mismatches).
 - Add the deployment section with screenshots or URLs in the README.
 - Application must be fully functional without local file dependencies.

User Story 2: Full System Testing and Quality Assurance

- As a Quality Assurance Engineer, I want to conduct comprehensive testing on all modules so that the final system is stable, accurate, and ready for presentation.
- Acceptance Criteria:
 - Implement at least 8 unit tests across API, AI, and data handling functions.
 - Perform integration and regression tests between backend and frontend.
 - Use pytest or unittest framework for reproducibility.
 - Document all test results and known issues in a “Testing Summary” section.
 - Achieve at least 90% test coverage before closing this story.

User Story 3: UI Polish and Role-Based Optimization

- As a System User, I want an improved and more intuitive dashboard interface so that the experience looks professional and responsive during demos.

- Acceptance Criteria:
 - Optimize dashboard layout using Streamlit's columns, containers, and expanders.
 - Refine color scheme, font sizes, and chart alignment for readability.
 - Add role-based filtering or toggle views (e.g., Admin vs. Clerk).
 - Ensure all visuals adjust dynamically on different devices (desktop/tablet).
 - Include branding elements such as project logo or banner for presentation.

User Story 4: End-to-End Integration and Performance Enhancement

- As a Product Owner, I want the frontend, backend, and AI components fully integrated and optimized so that the app performs efficiently and reliably for the Capstone Showcase.
- Acceptance Criteria:
 - Connect Streamlit UI to live API endpoints and AI modules with no broken calls.
 - Monitor latency and optimize loading speed (under 2 seconds).
 - Create and test real transaction flows (inventory → AI prediction → dashboard update).
 - Debug and fix any API response or data sync issues.
 - Verify system consistency across devices and browsers.

User Story 5: Showcase Preparation and Final Documentation

- As a Team Presenter, I want to prepare all documentation, visuals, and demo flow so that our Capstone presentation is polished, engaging, and easy to follow.
- Acceptance Criteria:
 - Finalize README with setup, usage, and testing instructions.
 - Create a final slide deck and demo walkthrough script.
 - Include screenshots or short demo video links in documentation.
 - Assign presenter roles and rehearse demo sequence.
 - Conduct one internal mock presentation and gather feedback.

The team members indicated their willingness to work on the following user stories.

Angela Banov - User Story 5: Showcase Preparation and Final Documentation

Vianne Novo Grifol - User Story 2: Full System Testing and Quality Assurance

Steven S. Palacios - User Story 4: End-to-End Integration and Performance Enhancement

Matthew Cobos - User Story 1: Deployment Pipeline Completion and Optimization

Momin Salar Khan - User Story 3: UI Polish and Role-Based Optimization

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram, and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

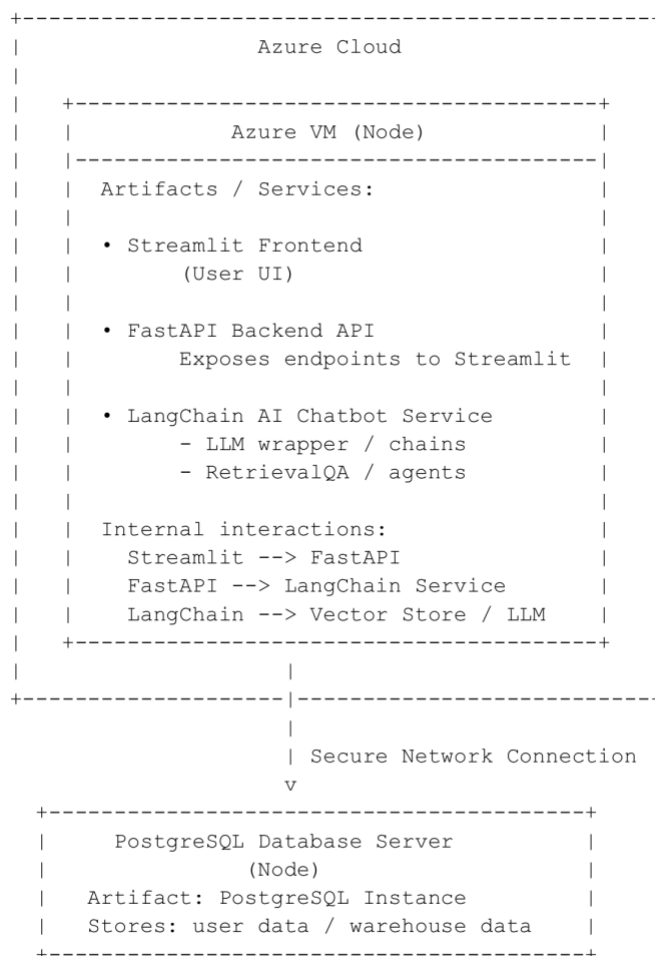
The architectural patterns used for the development of the application were based around Model-View-Controller (MVC) to organize the different layers of data-to-interface incorporated within our application. This involves the database and its organization connecting through the API into the front-end services. This also coincides with the integrated AI chatbot and forecasting services pulling from the same database to enable accurate projections and summaries of the available stock. The model is shown through database, controller through the API, and viewer through the dashboard and front-end designs.

System and Subsystem Decomposition

The system as a whole is composed of several different subsystems which actively work in unison to provide engaged information. The different subsystems being with the user interface, or the dashboard, which handles client requests, as well as uses API endpoints, while also handling business logic to display information. Another layer of the system involves the API,

which connects the database and the dashboard. This acts as the controller of the system. An important feature of the system includes the forecasting system, which processes real-time data. Finally, the last subsystem is the AI chatbot integrated into the application for natural language processing in queries based on the information in the database, as well as being able to describe and translate information.

Deployment Diagram



GLOSSARY

Warehouse:

A physical facility where goods, products, or spare parts are stored, managed, and distributed. Warehouses track quantities, locations, and movement of inventory across supply chains.

SKU (Stock Keeping Unit):

A unique identifier assigned to a specific product or variation (model, size, configuration). SKUs allow systems to track items consistently across warehouses, orders, and databases.

Stockout:

A condition in which inventory for a product falls to zero or below demand, preventing an organization from fulfilling customer orders. Stockouts lead to delays, lost revenue, and dissatisfied customers.

Surplus:

Excess inventory beyond normal operating needs. Surplus creates storage overhead and ties up capital that could be used more efficiently.

ERP (Enterprise Resource Planning):

A large, centralized platform that integrates core business functions such as finance, procurement, sales, and operations. ERP systems typically provide historical and transactional data, but they do not automate inventory balancing.

WMS (Warehouse Management System):

Software specialized for warehouse operations such as receiving, put-away, picking, packing, and shipping. WMS platforms track physical inventory movement but rarely provide network-level redistribution recommendations.

Transfer Recommendation:

A data-driven suggestion to move a quantity of product from one warehouse to another. Recommendations consider surplus, stockouts, demand, lead times, and distance to improve system-wide availability.

Dashboard:

A visual interface that aggregates operational data into charts, tables, and summaries. Dashboards allow users to monitor metrics, search for products, and interact with management tools without manual database queries.

FastAPI:

A high-performance Python web framework for building REST and HTTP services. It supports type validation through Pydantic models and is optimized for speed and scalability.

ORM (Object Relational Mapping):

A programming technique that maps data stored in relational databases to object-based models in code. ORMs such as SQLAlchemy reduce manual SQL writing and preserve consistency across the application.

Ledger Event:

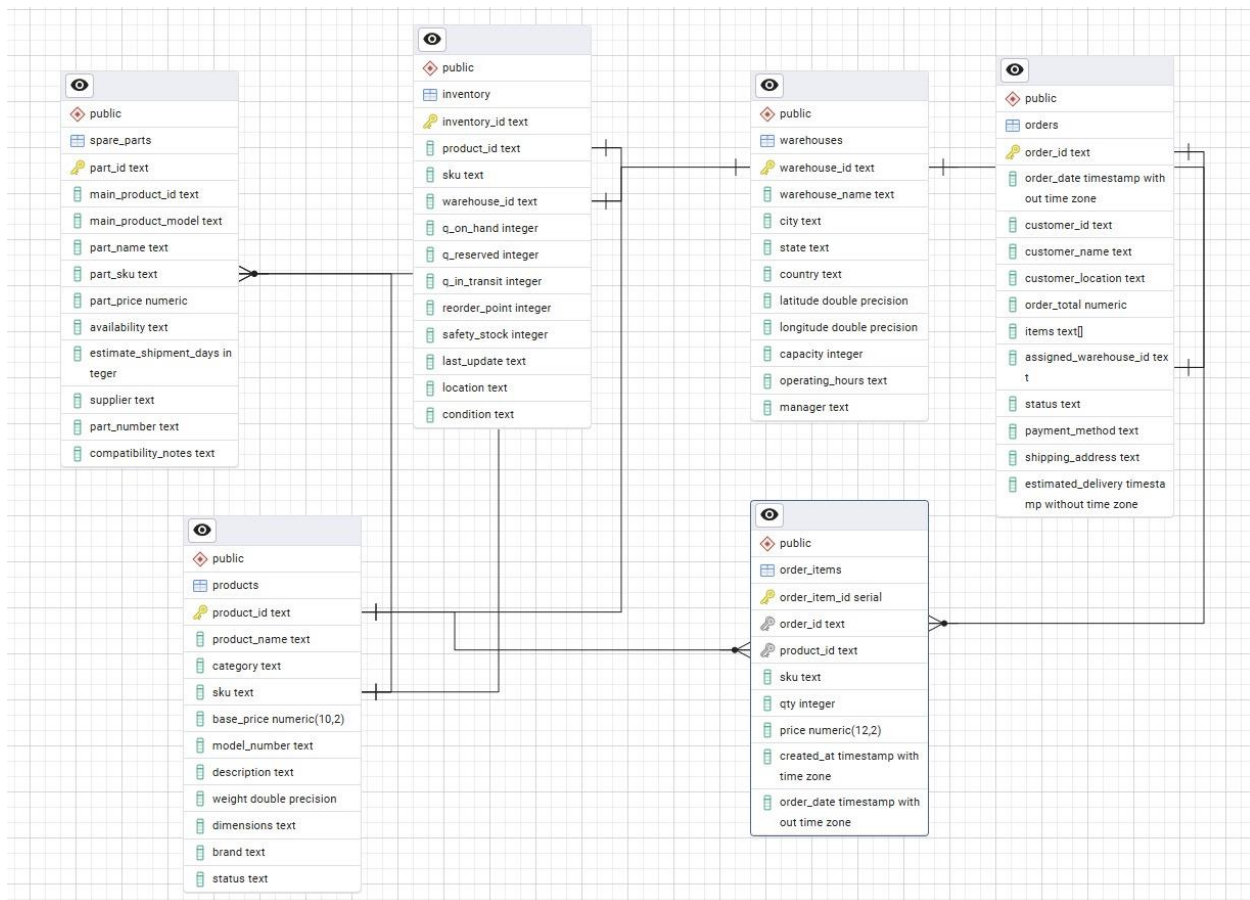
A state-changing action applied to inventory records such as receiving stock, shipping items, or adjusting quantities. Ledger events represent real warehouse activities and update system counts accordingly.

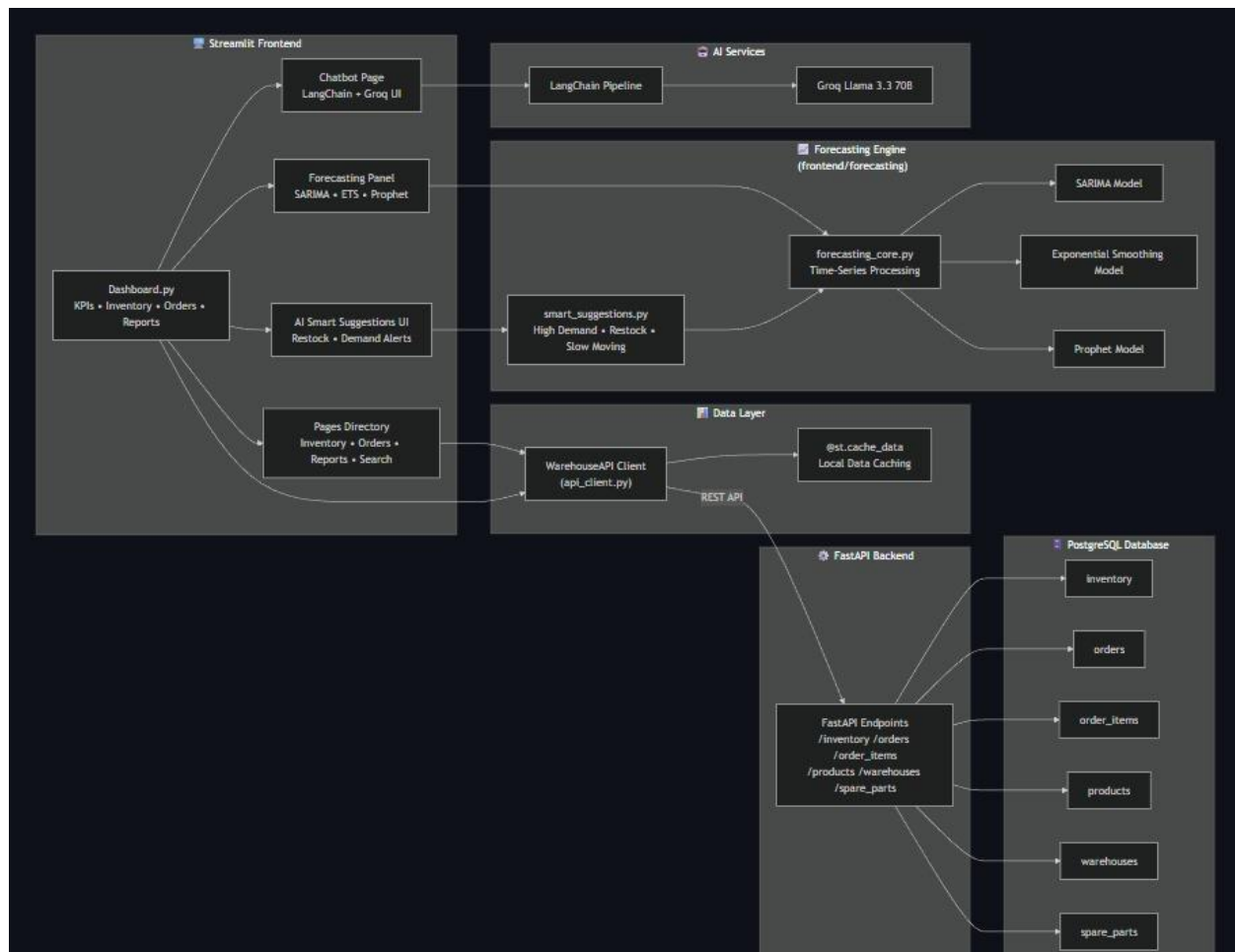
Agent:

A software component capable of performing tasks autonomously. In warehouse systems, an agent can monitor data, detect anomalies, forecast demand, and suggest redistribution actions without manual intervention.

APPENDIX

Appendix A - UML Diagram





Appendix B - Sprint Review Reports

Review Meeting Minutes (Sprint 1)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 11:00 am

End time: 12:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Generate Synthetic Data

User Story 2: Basic Inventory Dashboard

User Story 3: Connect Dashboard to Backend API (Mock)

User Story 4: Stock Ledger Agent (Stub)

User Story 5: Forecasting Setup (Initial Model)

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

All sprints above will be continued onto sprint 2.

Review Meeting Minutes (Sprint 2)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Basic Inventory Dashboard

User Story 2: Connect Dashboard to Backend API (Mock)

User Story 3: Stock Ledger Agent (Stub)

User Story 4: Forecasting Setup (Initial Model)

User Story 5: Low-Stock Alerts

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

User Story 2: Connect Dashboard to Backend API (Mock) will be moved back to the product backlog and continued onto Sprint 3

User Story 3: Stock Ledger Agent (Stub) will be moved back to the product backlog and continued onto Sprint 3

Review Meeting Minutes (Sprint 3)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Connect Dashboard to Backend API (Mock)

User Story 2: Stock Ledger Agent (Stub)

User Story 3: Inventory Transactions (Adjustments & Movements)

User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)

User Story 5: Role-Based Dashboard Views

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

User Story 1: Connect Dashboard to Backend API (Mock)

User Story 2: Stock Ledger Agent (Stub)

User Story 3: Inventory Transactions (Adjustments & Movements)

User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)

User Story 5: Role-Based Dashboard Views

Review Meeting Minutes (Sprint 4)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov

Vianne Novo Grifol

Steven S. Palacios

Matthew Cobos

Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Connect Dashboard to Backend API (Mock)
User Story 2: Stock Ledger Agent (Stub)
User Story 3: Inventory Transactions (Adjustments & Movements)
User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)
User Story 5: Role-Based Dashboard Views

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

User Story 2: Stock Ledger Agent (Stub)
User Story 3: Inventory Transactions (Adjustments & Movements)
User Story 4: Forecasting Expansion (Multi-SKU with Warehouse Segmentation)

Review Meeting Minutes (Sprint 5)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Inventory Transactions (Adjustments & Movements)
User Story 2: AI Integration for Predictive Stock Insights
User Story 3: Deployment Pipeline Setup
User Story 4: System Testing and Documentation
User Story 5: UI Polish and Role Optimization

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

User Story 3: Deployment Pipeline Setup

User Story 4: System Testing and Documentation

User Story 5: UI Polish and Role Optimization

Review Meeting Minutes (Sprint 6)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov

Vianne Novo Grifol

Steven S. Palacios

Matthew Cobos

Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Deployment Pipeline Completion and Optimization

User Story 2: Full System Testing and Quality Assurance

User Story 3: UI Polish and Role-Based Optimization

User Story 4: End-to-End Integration and Performance Enhancement

User Story 5: Showcase Preparation and Final Documentation

The following ones were moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

User Story 1: Deployment Pipeline Completion and Optimization

User Story 2: Full System Testing and Quality Assurance

User Story 3: UI Polish and Role-Based Optimization

User Story 4: End-to-End Integration and Performance Enhancement

User Story 5: Showcase Preparation and Final Documentation

Review Meeting Minutes (Sprint 7)

Project: AI-Driven Warehouse Stock Management

Attendees:

Angela Rumenov Banov
Vianne Novo Grifol
Steven S. Palacios
Matthew Cobos
Momin Salar Khan

Start time: 6:00 pm

End time: 7:00 pm

After a show and tell presentation, the implementation of the following user stories was accepted by the product owners: All.

User Story 1: Deployment Pipeline Completion and Optimization

User Story 2: Full System Testing and Quality Assurance

User Story 3: UI Polish and Role-Based Optimization

User Story 4: End-to-End Integration and Performance Enhancement

User Story 5: Showcase Preparation and Final Documentation

The following ones were moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

None

Appendix C - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

INSTALLATION GUIDE

This project is an AI-powered warehouse stock management platform that integrates real-time inventory, demand forecasting, smart replenishment insights, and an interactive dashboard for warehouse operations.

The system combines:

- FastAPI backend
- PostgreSQL database
- Streamlit dashboard
- LangChain + Groq AI chatbot
- SARIMA, ETS, Prophet forecasting models

The goal is to help operations teams reduce stockouts, prevent overstock, and improve decision-making through automated intelligence.

PRE-INSTALLATION REQUIREMENTS

Local Requirements:

- Python 3.10+
- Docker & Docker Compose
- Postgres 16 (if running locally)
- Git

Cloud Deployment Requirements:

- VM running Ubuntu 22.04

- Docker + Docker Compose installed
- Exposed ports:
 - 8000 (Backend API)
 - 8501 (Streamlit dashboard)
 - 5432 (optional — database)

INSTALLATION PROCESS

Step 1: Clone the Repository

Begin by cloning the repository into VS code or a similar environment. Create a new terminal session and run the following command:

```
git clone https://github.com/AngelaBanov23/Capstone-2---AI-Driven-Warehouse-Stock-Management.git
```

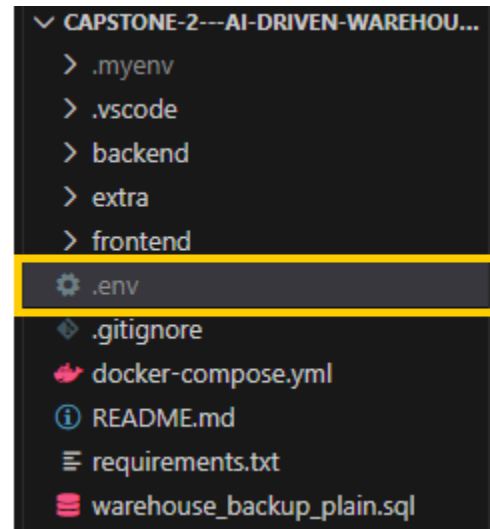
Then, navigate to the new directory with:

```
cd Capstone-2---AI-Driven-Warehouse-Stock-Management
```

Step 2: Create Environment Variables

For proper setup, an environment variable file (**.env**) must be created in the main project root/directory. Create a new file titled **".env"**. Inside this file, create the following variables:

```
.env
1  POSTGRES_USER=appuser
2  POSTGRES_PASSWORD="your_password_here"
3  POSTGRES_DB=WarehouseStockManagement
4  POSTGRES_HOST=db
5  POSTGRES_PORT=5432
6
7  API_BASE_URL="http://backend:8000"
8
9  GROQ_API_KEY="your_groq_api_key_here"
10
11
```



Step 3: Start the System with Docker

In a new terminal session, from the project root, run the following command to build the docker containers:

```
docker compose up --build
```

Service	Description	Port
backend	FastAPI service	8000
frontend	Streamlit dashboard	8501
db	PostgreSQL 16	5432

Step 4: Restoring the Database

For this project, the database and its contents must be injected into the database docker container. To do so, create a new terminal session in the main project root. Next, run the following commands:

```
docker cp warehouse_backup_plain.sql warehouse_db:/warehouse_backup_plain.sql
```

```
docker exec -it warehouse_db psql -U appuser -d WarehouseStockManagement -f  
/warehouse_backup_plain.sql
```

```
docker exec -it warehouse_db psql -U appuser -d WarehouseStockManagement -c "\dt"
```

5. Access the Application

Frontend Dashboard:

<http://localhost:8501>

API Documentation (Swagger UI):

<http://localhost:8000/docs>

Short Comings

Due to time constraints we couldn't fully implement all the features that we would have liked this project. Below is a list of the shortcomings:

- Static dataset (no live data ingestion)
- Missing authentication and user roles
- LLM chatbot limited to small data slices
- Suggestion engine not fully autonomous
- Backend missing update-and-action endpoints