Inventory Management System

Project Report

EEC 521/CIS 534 – Software Engineering Project

Submitted By

Jaydeep Ravaliya (2873999)

Kevin Panchal (2871507)

Jaykumar Mistry (2875386)

Date:- 12/05/2024

1. Introduction

1.1. Purpose and Scope

The purpose of this report is to document the development, design, and implementation of the **Inventory Management System (IMS)**. This system addresses the critical need for businesses to manage inventory efficiently and accurately, ensuring that stock levels are optimized and operational processes are streamlined.

The **scope** of the IMS includes:

- Management of inventory items, including adding, editing, and deleting products.
- Tracking stock levels and generating notifications for low stock.
- Generating reports based on inventory data, such as stock movement and supplier records.
- Ensuring system usability through a user-friendly interface with responsive design.

The system is designed for use by warehouse staff, inventory managers, sales managers, and system administrators. It focuses on small-to-medium-scale operations and includes functionalities for tracking and reporting but excludes advanced integrations like third-party tools and extensive load testing.

1.2 Product Overview

The **Inventory Management System (IMS)** is a web-based application developed using Python and Django. It offers a comprehensive solution for managing inventory with the following **capabilities**:

- **Inventory Management**: Add, edit, delete, and search inventory items.
- **Stock Tracking**: Monitor stock levels and generate alerts when thresholds are crossed.
- **Supplier Management**: Maintain supplier details for efficient procurement.
- **Reporting**: Generate detailed reports for inventory movement, category-specific data, and date ranges.
- **User Management**: Administer user roles and permissions for secure system access.

Scenarios for using the IMS:

1. **Warehouse Management**: A warehouse staff member updates stock levels after receiving a shipment.
2. **Sales Reporting**: A sales manager generates a report to analyze inventory data for a specific product category.
3. **Procurement**: An inventory manager receives low-stock alerts and contacts suppliers for restocking.
4. **System Administration**: An administrator adds new users or updates their access permissions.

The IMS is user-centric, prioritizing accessibility and efficiency in daily operations.

1.3 Structure of the Document

This document is organized as follows:

- **Section 1:** Introduction to the project, including its purpose, product overview, document structure, and relevant terminology.
- **Section 2:** Project management plan covering organization, lifecycle model, risk analysis, resource requirements, and deliverables.
- **Section 3:** Requirement specifications, including stakeholders, use cases, and non-functional requirements.
- **Section 4:** Architectural design and rationale.
- **Section 5:** Detailed design, including UI, components, and database design.
- **Section 6:** Test management, with test cases, results, and defect reports.
- **Section 7:** Conclusions, outcomes, lessons learned, and potential future enhancements.
- **Section 8:** References and supporting materials.

1.4 Terms, Acronyms, and Abbreviations

Term/Acronym	Description
IMS	Inventory Management System
UI	User Interface
Django	A high-level Python web framework used for back-end development
SQLite3	A lightweight relational database used for storing application data
UAT	User Acceptance Testing
CRUD	Create, Read, Update, Delete operations
SDLC	Software Development Lifecycle
HTML	HyperText Markup Language, used for structuring the front-end
CSS	Cascading Style Sheets, used for styling the front-end
SMTP	Simple Mail Transfer Protocol, used for sending email notifications
Agile	A project management methodology focused on iterative development and collaboration

2. Project Management Plan

2.1 Project Organization

The project team is structured to ensure efficient collaboration and task distribution:

- **Team Members and Roles**:
  - **Jaydeep Ravaliya (2873999)**: Project Manager and Lead Developer Responsibilities: Coordinating tasks, ensuring deadlines are met, overseeing system design and coding.
  - **Jaykumar Mistry (2875386)**: Front-End Developer Responsibilities: Designing and implementing the user interface, ensuring responsiveness and accessibility.
  - **Kevin Panchal (2871507)**: Back-End Developer and Tester Responsibilities: Developing server-side logic, managing the database, and overseeing testing processes.
- **Key Stakeholders**:
  - **Supervising Faculty**: Provides project guidance and feedback.
  - **Potential End Users**: Warehouse staff, sales managers, and system administrators who validate usability during User Acceptance Testing (UAT).

2.2 Lifecycle Model Used

The **Agile Software Development Lifecycle (SDLC)** model was chosen due to its flexibility and focus on iterative development. This model allowed the team to:

- Deliver working increments of the system in shorter cycles (sprints).
- Gather and incorporate feedback from stakeholders at each iteration.
- Prioritize features based on user needs and adjust the scope if necessary.

Each sprint followed a four-week cycle:

1. **Sprint Planning**: Define goals and deliverables for the sprint.
2. **Development**: Implement the planned features or modules.
3. **Testing**: Conduct unit, integration, and system testing for developed components.
4. **Review and Retrospective**: Assess the sprint outcomes and plan improvements for the next cycle.

2.3 Risk Analysis

Risk management was integral to the project to mitigate potential issues. Key risks and mitigation strategies are summarized below:

Risk	Impact	Likelihood	Mitigation Strategy
Delays in feature implementation	High	Medium	Set realistic deadlines and track progress with weekly meetings.
Database limitations (SQLite3)	Medium	High	Plan for migration to a scalable database in future iterations.
Incomplete or incorrect requirements	High	Medium	Conduct frequent stakeholder reviews to validate requirements.
Security vulnerabilities	High	Low	Implement secure coding practices and conduct vulnerability assessments.
Incompatibility with mobile devices	Medium	Medium	Use responsive design principles and test on various screen sizes.

2.4 Hardware and Software Resource Requirements

Hardware Requirements:

- **Development Machines**:
  - Processor: Intel i5 or equivalent
  - RAM: 8 GB minimum
  - Storage: 256 GB SSD or higher
- **Test Machines**:
  - Variety of devices, including Windows, macOS, and mobile devices (Android/iOS).

Software Requirements:

- **Development Tools**:
  - Django framework (Python) for back-end development.
  - SQLite3 for database management.
  - HTML5, CSS3, JavaScript for front-end development.
- **Testing Tools**:
  - Browser tools for UI responsiveness testing.
- **Version Control**:
  - Git and GitHub for collaborative code management.
- **Project Management**:
  - Jira for tracking tasks and managing sprints.

2.5 Deliverables and Schedule

Week	Dates	Task
Week 1	24 Sep 2024 - 29 Sep 2024	Requirement Gathering & Initial Planning
		Define project scope, features, and constraints
Week 2	30 Sep 2024 - 6 Oct 2024	System Design
		Design database schema, create wireframes, system architecture

Week 3	7 Oct 2024 - 13 Oct 2024	Frontend Development (HTML/CSS/Bootstrap)
		Develop templates for inventory pages, dashboard, and navigation
Week 4	14 Oct 2024 - 20 Oct 2024	Backend Development
		Implement models, views, and URL routing in Django
Week 5	21 Oct 2024 - 27 Oct 2024	User Authentication Setup
		Implement user registration, login, logout, and roles
Week 6	21 Oct 2024 - 27 Oct 2024	CRUD Operations for Inventory
		Implement add, update, delete functionalities
Week 7	4 Nov 2024 - 10 Nov 2024	Quantity Alerts & Messaging
		Implement stock alerts and Django messages
Week 8	11 Nov 2024 - 17 Nov 2024	Testing & Validation
		Unit testing and integration testing
Week 9	18 Nov 2024 - 24 Nov 2024	Final Report Preparation
		Write final project report, system documentation
Week 10	25 Nov 2024 - 1 Dec 2024	Presentation & Submission
		Prepare slides, demo, and submit final project

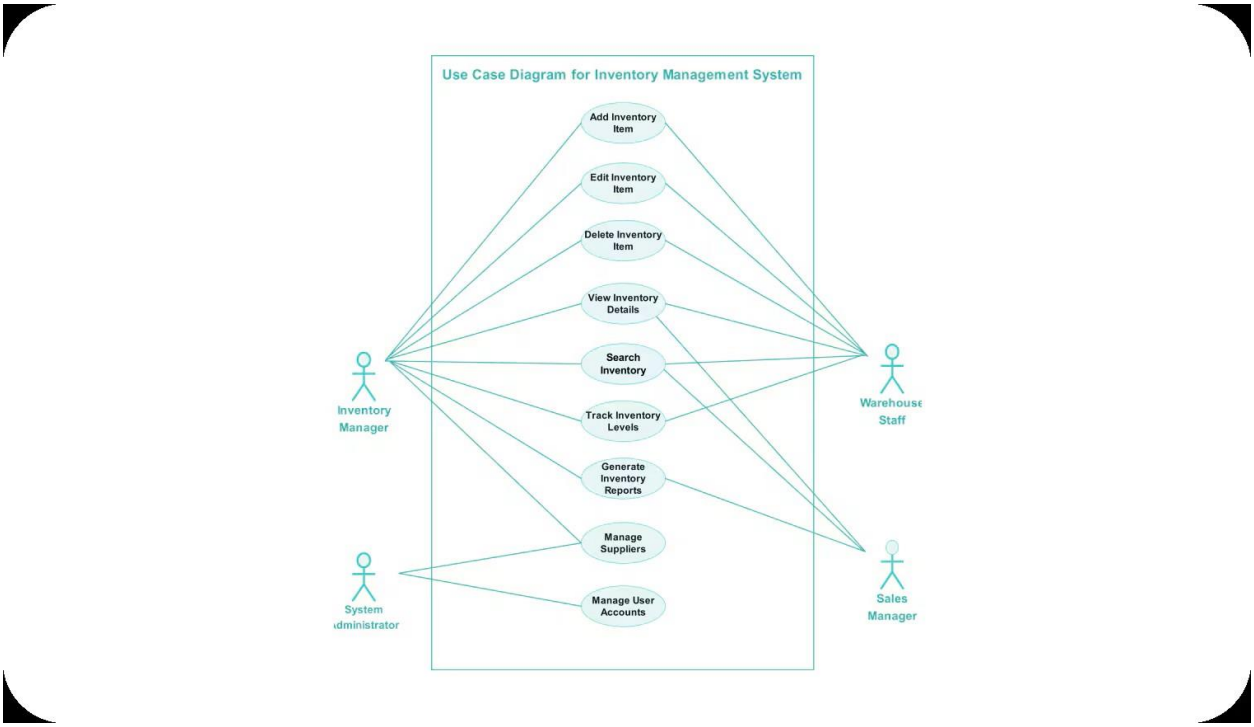
3. Requirement Specifications

3.1 Stakeholders for the System

- Business Owners (Primary users)
- Employees (Secondary users)

3.2 Use Cases

3.2.1 Graphic Use Case Model



3.2.2 Textual Description for Each Use Case

1. Add Inventory Item

- **Actors:** Inventory Manager
- **Description:**
    - Allows the Inventory Manager to add new products to the inventory database. This includes providing details such as product name, category, supplier information, and initial stock levels.

2. Edit Inventory Item

- **Actors:** Inventory Manager
- **Description:**
    - Enables modifications to the details of existing inventory items. For example, correcting product names, updating supplier information, or adjusting categories.

3. Delete Inventory Item

- **Actors:** Inventory Manager
- **Description:**
    - Facilitates the removal of outdated or discontinued products from the inventory. The system ensures data integrity by prompting confirmations before deletion.

4. View Inventory Details

- **Actors:** Warehouse Staff, Sales Manager
- **Description:**
    - Provides a detailed view of the inventory, including product availability, stock levels, and supplier details. This helps Warehouse Staff and Sales Managers monitor and plan operations.

5. Search Inventory

- **Actors:** Warehouse Staff, Sales Manager
- **Description:**
    - Allows users to quickly locate specific products using filters such as product name, category, or supplier.

6. Track Inventory Levels

- **Actors:** Inventory Manager, Warehouse Staff
- **Description:**
    - Monitors real-time stock levels of all products. The system generates alerts for low stock levels to help prevent shortages.

7. Generate Inventory Reports

- **Actors:** Inventory Manager, Sales Manager
- **Description:**
    - Provides reporting features that generate summaries based on categories, date ranges, or stock levels. Sales Managers can use this for decision-making, while Inventory Managers ensure proper stock management.

8. Manage Suppliers

- **Actors:** Inventory Manager
- **Description:**
    - Maintains a record of suppliers and their details, including contact information and associated products. This feature ensures seamless supplier relationships.

9. Manage User Accounts

- **Actors:** System Administrator
- **Description:**
    - The System Administrator manages user roles, permissions, and account settings. This ensures secure access control for different actors in the system.

3.3. Rationale for Your Use Case Model

- The **use case model** for the Inventory Management System (IMS) was developed to provide a clear and concise representation of the system's functionality and how it interacts with various user roles. The rationale behind this model is as follows:
- **Comprehensive Coverage of Functional Requirements**
    - The use case model covers all essential functionalities, such as inventory management, stock tracking, reporting, and user account management. This ensures the system supports the day-to-day operational needs of the organization.

- **Role-Specific Functionality**
    - By defining distinct roles (e.g., Inventory Manager, Warehouse Staff, Sales Manager, System Administrator), the model reflects the real-world responsibilities of each actor. This helps ensure that tasks are assigned appropriately and permissions are managed securely.
- **Prioritization of Core Operations**
    - The use cases emphasize critical operations like adding, editing, deleting inventory items, and generating reports. These are foundational processes that ensure smooth inventory management.

- **Scalability and Extensibility**
    - The modular nature of the use case model allows for easy extension. For instance, additional use cases can be added for advanced features such as predictive analytics or integration with third-party tools.
- **Improved Communication**
    - The use case diagram serves as a communication tool between stakeholders, developers, and testers, ensuring a shared understanding of system functionality.
- **Facilitation of Testing and Validation**
    - By clearly defining the use cases, the model enables the development of targeted test cases, ensuring all functional aspects of the system are verified during testing.

3.4. Non-Functional Requirements

Requirement ID	Requirement Statement	Explanation
NFR 001	The system must respond to user actions within 3 seconds under normal load.	Performance requirement to ensure smooth user experience.
NFR 002	The system must support 100 simultaneous users.	Scalability requirement to handle multiple concurrent users.

NFR 003	The system must encrypt sensitive data, including user credentials.	Security requirement to protect data from unauthorized access.
NFR 004	Operating System Compatibility	The application should be compatible with macOS operating systems, with support for different versions and configurations
NFR 005	The system must allow regular data backups.	Backup and recovery requirement to ensure data integrity and recoverability.
NFR 006	The system should have minimal downtime during planned maintenance.	Minimize disruption to users by reducing maintenance window duration.
NFR 007	The system should support thousands of product entries.	Scalability requirement for large-scale inventory data.
NFR 008	The system should be compatible with all major web browsers.	Compatibility with Chrome, Firefox, Safari, etc., across different operating systems.

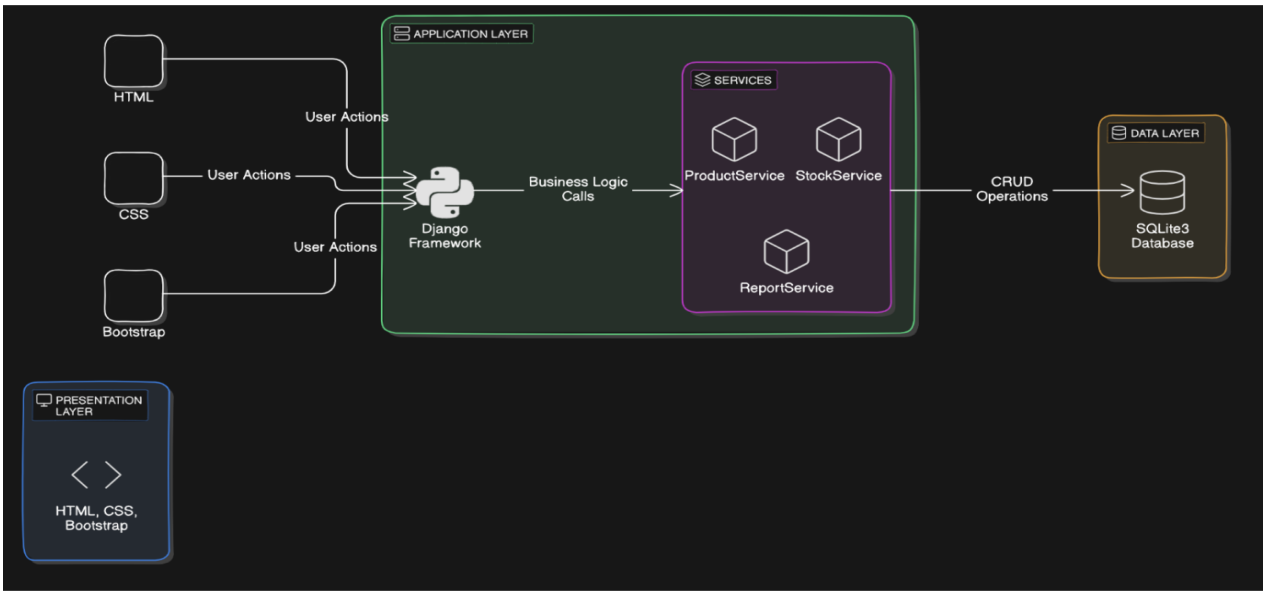
4. Architecture

4.1 Architectural Style(s) Used

The project uses a three-tier architecture: Presentation, Application, and Data Layers.

4.2 Architectural Model

- **Presentation Layer:** HTML, CSS, Bootstrap.
- **Application Layer:** Django.
- **Data Layer:** SQLite3.



4.3 Technology, Software, and Hardware Used

- Python, Django ORM for backend.
- HTML/CSS **Bootstrap** for the frontend.
- SQLite3 for Data.

4.4 Rationale for Architectural Style

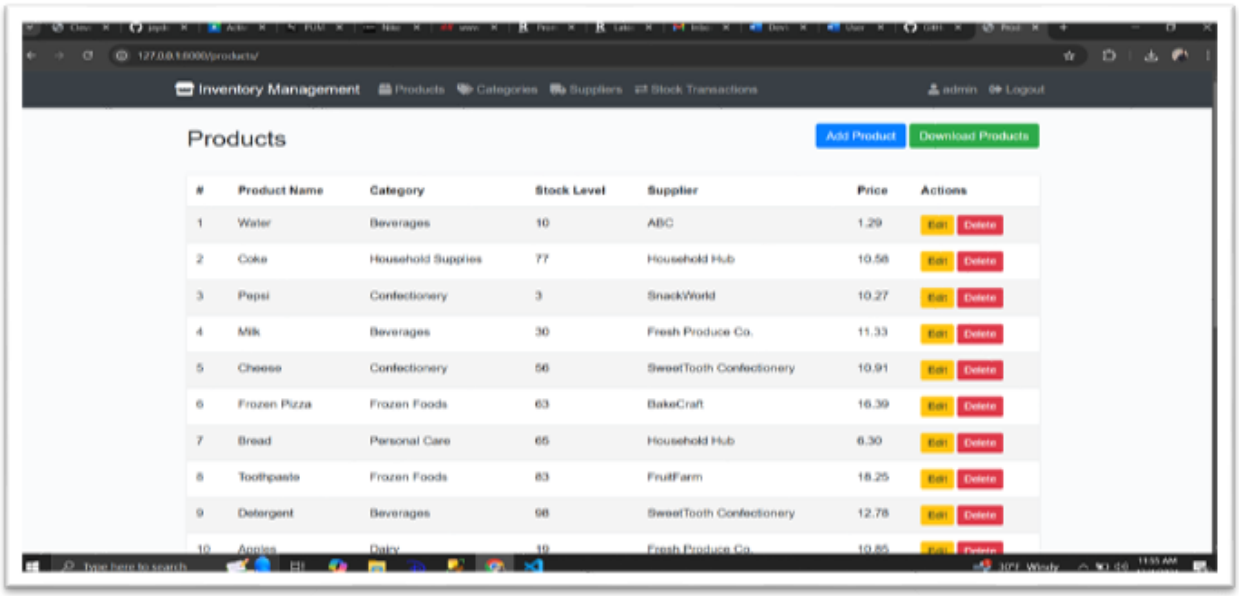
This architecture separates concerns and enhances maintainability.

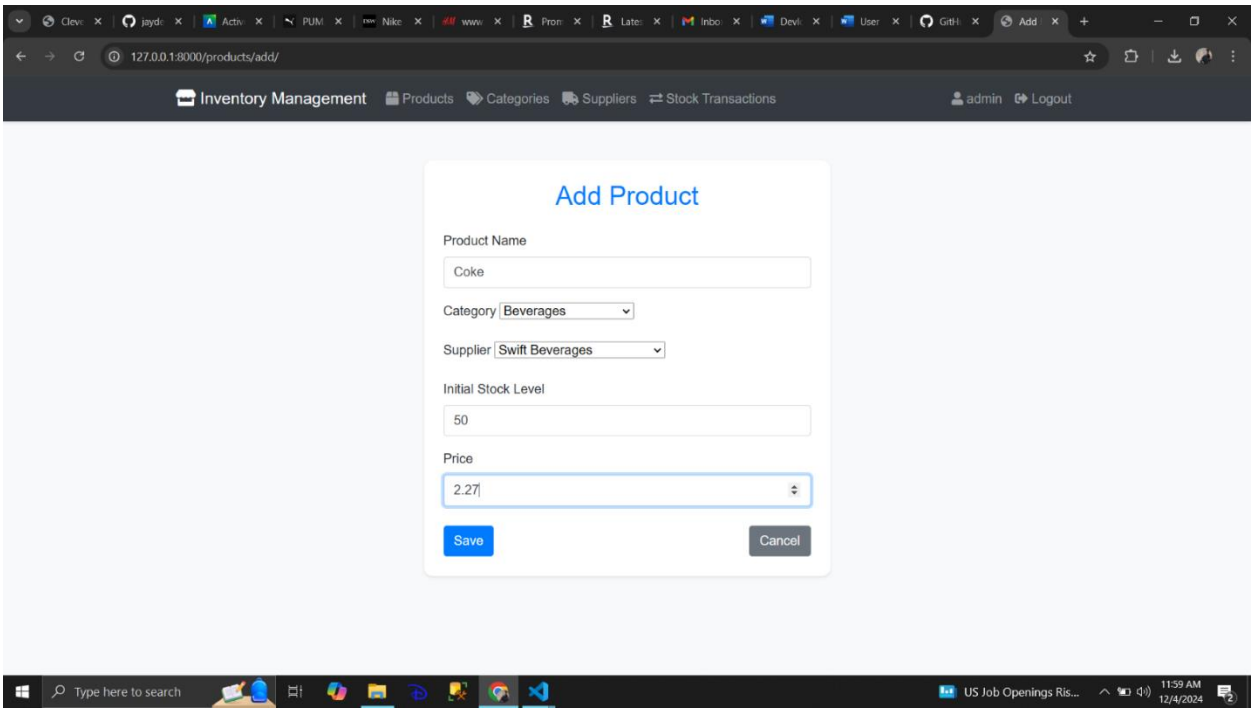
5. Design

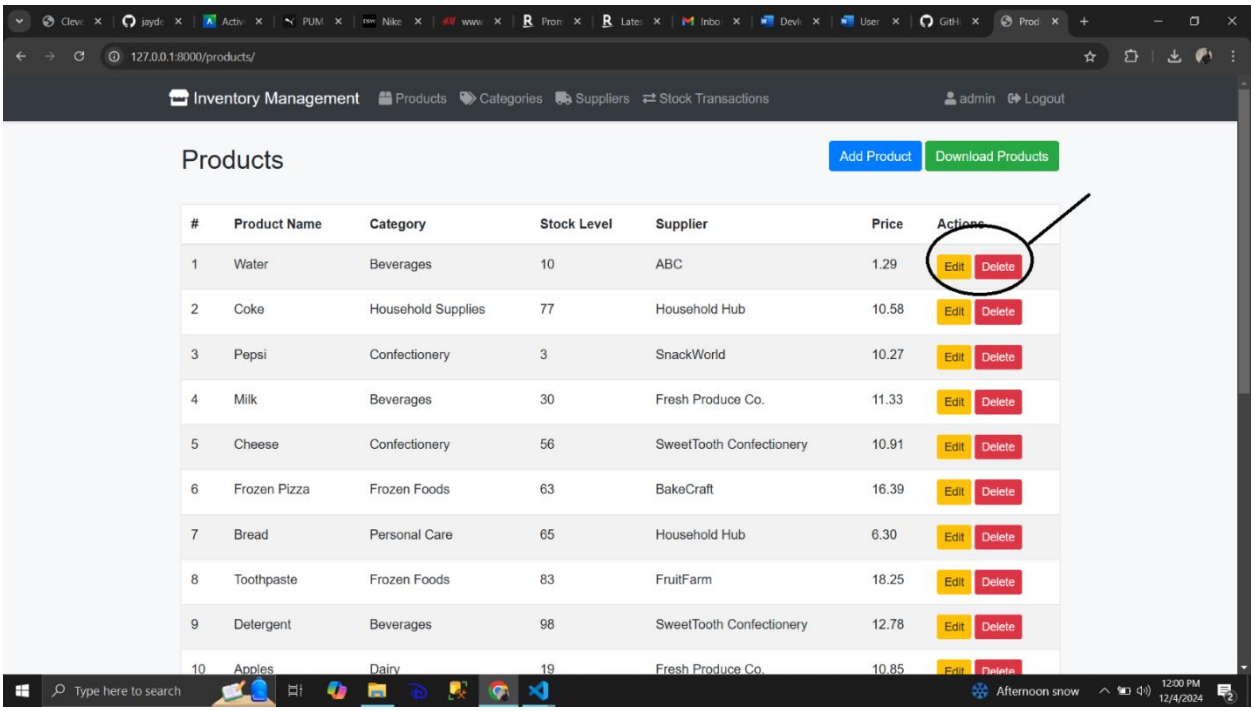
5.1 User Interface Design

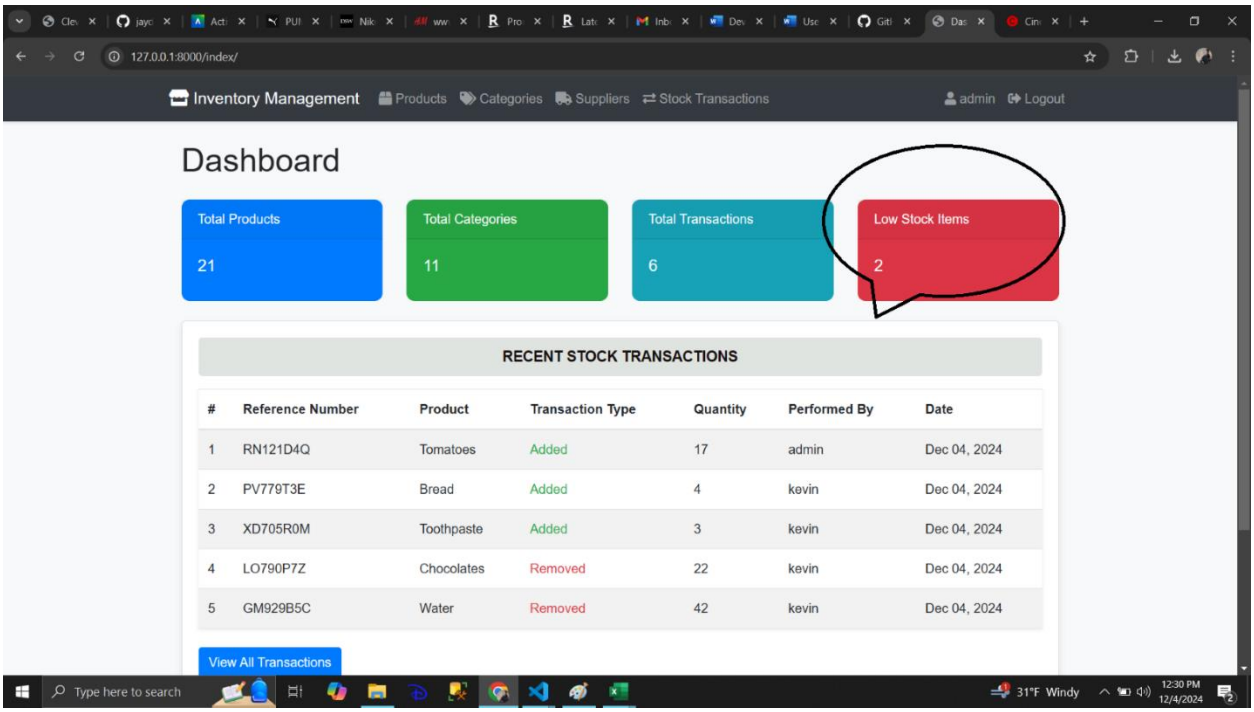
The user interface (UI) of the Inventory Management System (IMS) is designed to provide an intuitive, responsive, and visually appealing experience for all user roles.

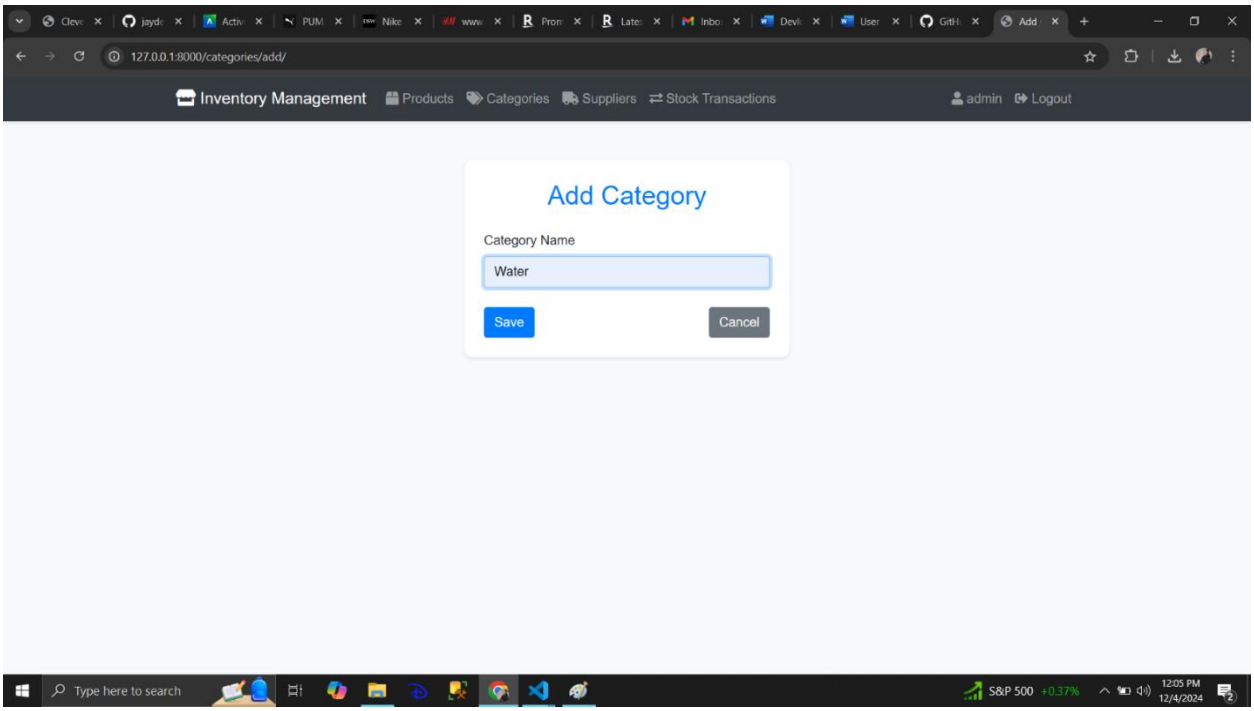
- **Key Features:**
  - **Dashboard:** Displays real-time statistics on inventory status, such as low-stock alerts and total inventory value.
  - **Inventory Management Pages:** Allow users to add, edit, view, or delete inventory items using forms.
  - **Role-Based Views:**
    - Inventory Managers: Full access to all inventory-related features.
    - Warehouse Staff: Limited access to update stock levels.
  - **Search and Filter:** Advanced filtering options for quickly finding inventory items by category, supplier, or date.
  - **Notifications:** Visual and email-based notifications for low stock or other alerts.
  - **Accessibility:** Keyboard navigation, ARIA roles, and high-contrast themes for better usability.

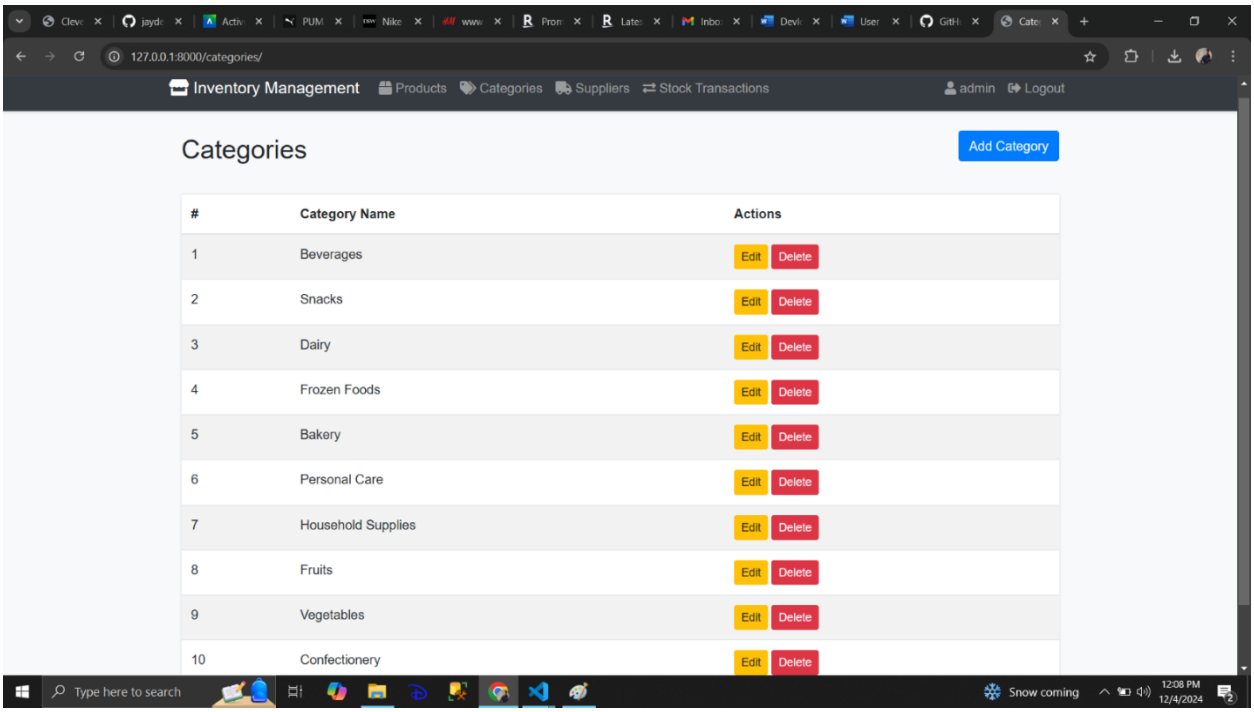

Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Products
Add Product
Download Products
Product Name Category Stock Level Supplier Price Actions
1 Water Beverages 10 ABC 1.29 Edit Delete
2 Coke Household Supplies 77 Household Hub 10.58 Edit Delete
3 Pepsi Confectionery 3 SnackWorld 10.27 Edit Delete
4 Milk Beverages 30 Fresh Produce Co. 11.33 Edit Delete
5 Cheese Confectionery 56 SweetTooth Confectionery 10.91 Edit Delete
6 Frozen Pizza Frozen Foods 63 BakeCraft 16.39 Edit Delete
7 Bread Personal Care 65 Household Hub 6.30 Edit Delete
8 Toothpaste Frozen Foods 83 FruitFarm 18.25 Edit Delete
9 Detergent Beverages 98 SweetTooth Confectionery 12.78 Edit Delete
10 Apples Dairy 19 Fresh Produce Co. 10.85 Edit Delete

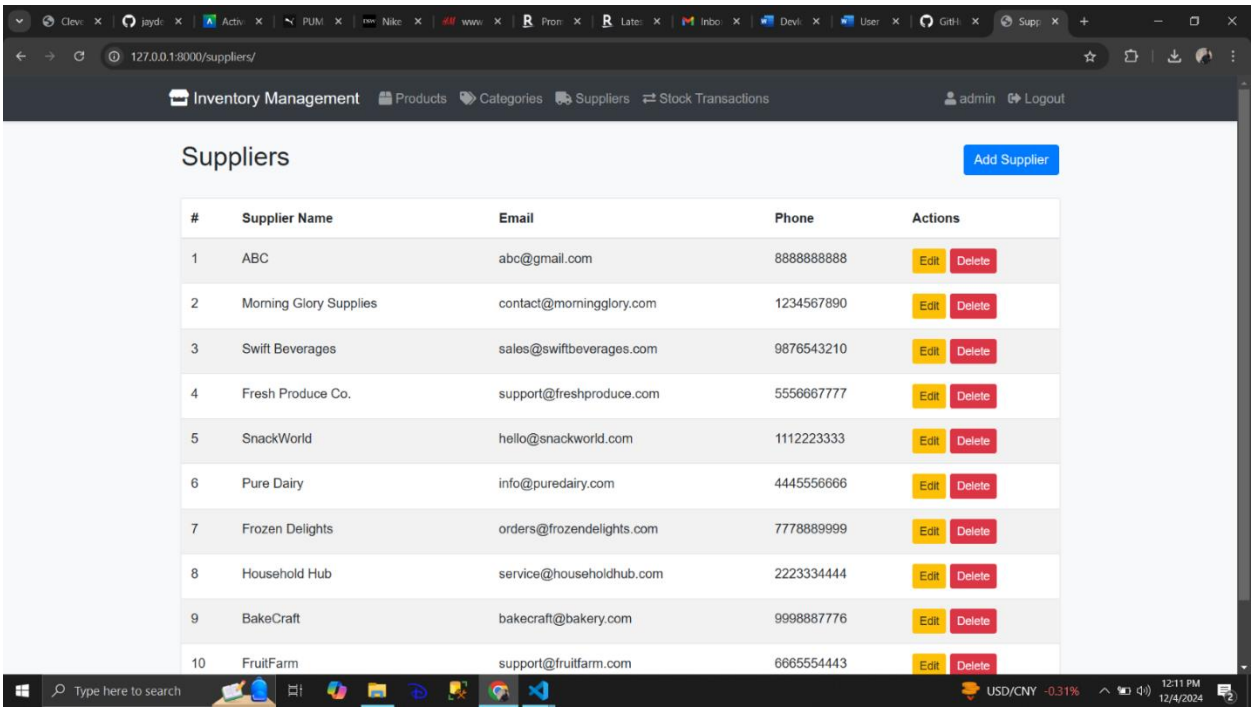

127.0.0.1:8000/products/add/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Add Product
Product Name
Coke
Category Beverages
Supplier Swift Beverages
Initial Stock Level
50
Price
2.27
Save
Cancel
Type here to search
US Job Openings Ris...
11:59 AM
12/4/2024

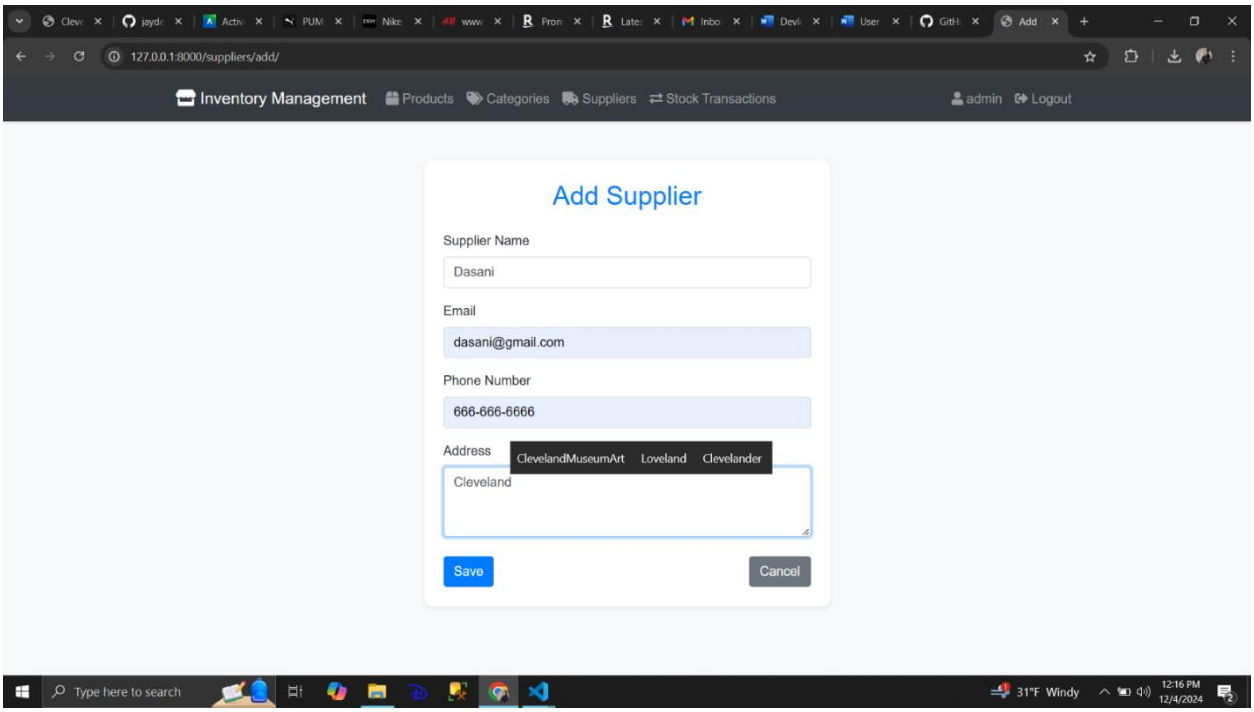

127.0.0.1:8000/products/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Products
Add Product
Download Products
Product Name Category Stock Level Supplier Price Actions
1 Water Beverages 10 ABC 1.29 Edit Delete
2 Coke Household Supplies 77 Household Hub 10.58 Edit Delete
3 Pepsi Confectionery 3 SnackWorld 10.27 Edit Delete
4 Milk Beverages 30 Fresh Produce Co. 11.33 Edit Delete
5 Cheese Confectionery 56 SweetTooth Confectionery 10.91 Edit Delete
6 Frozen Pizza Frozen Foods 63 BakeCraft 16.39 Edit Delete
7 Bread Personal Care 65 Household Hub 6.30 Edit Delete
8 Toothpaste Frozen Foods 83 FruitFarm 18.25 Edit Delete
9 Detergent Beverages 98 SweetTooth Confectionery 12.78 Edit Delete
10 Apples Dairy 19 Fresh Produce Co. 10.85 Edit Delete
Type here to search
Afternoon snow
12:00 PM
12/4/2024

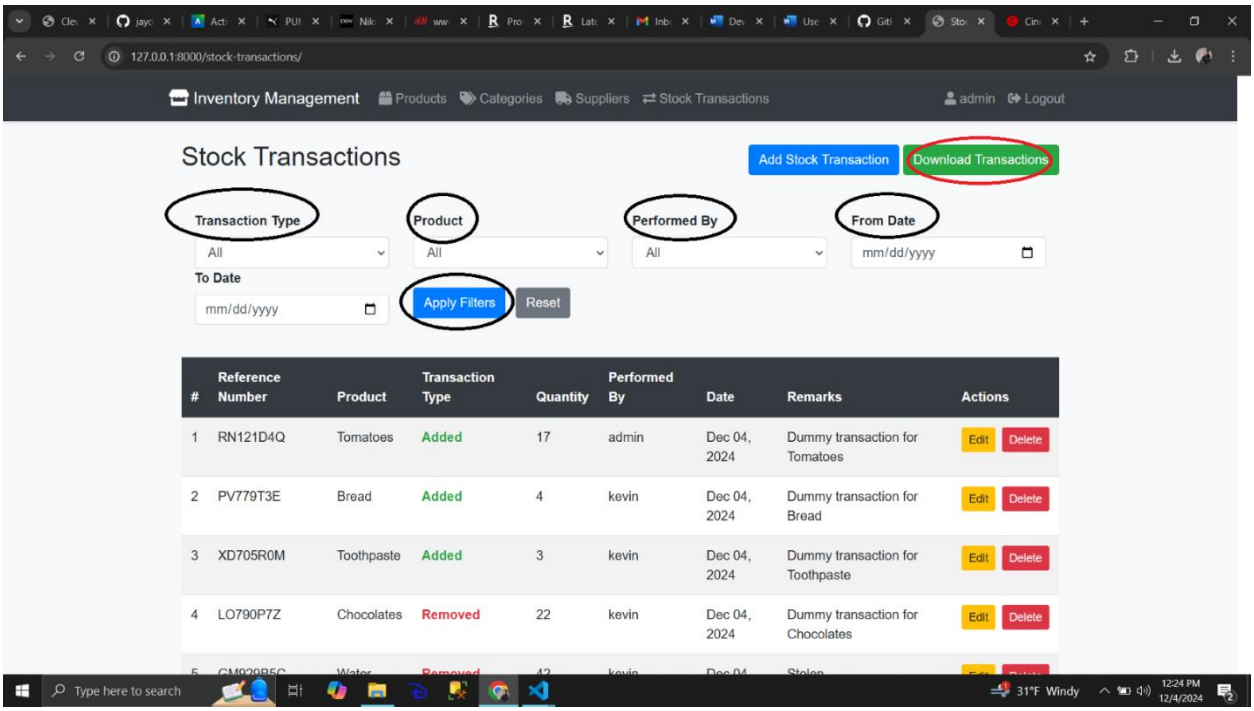

127.0.0.1:8000/index/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Dashboard
Total Products
21
Total Categories
11
Total Transactions
6
Low Stock Items
2
RECENT STOCK TRANSACTIONS
Reference Number Product Transaction Type Quantity Performed By Date
1 RN121D4Q Tomatoes Added 17 admin Dec 04, 2024
2 PV779T3E Bread Added 4 kevin Dec 04, 2024
3 XD705R0M Toothpaste Added 3 kevin Dec 04, 2024
4 LO790P7Z Chocolates Removed 22 kevin Dec 04, 2024
5 GM929B5C Water Removed 42 kevin Dec 04, 2024
View All Transactions
Type here to search
31°F Windy
12:30 PM
12/4/2024

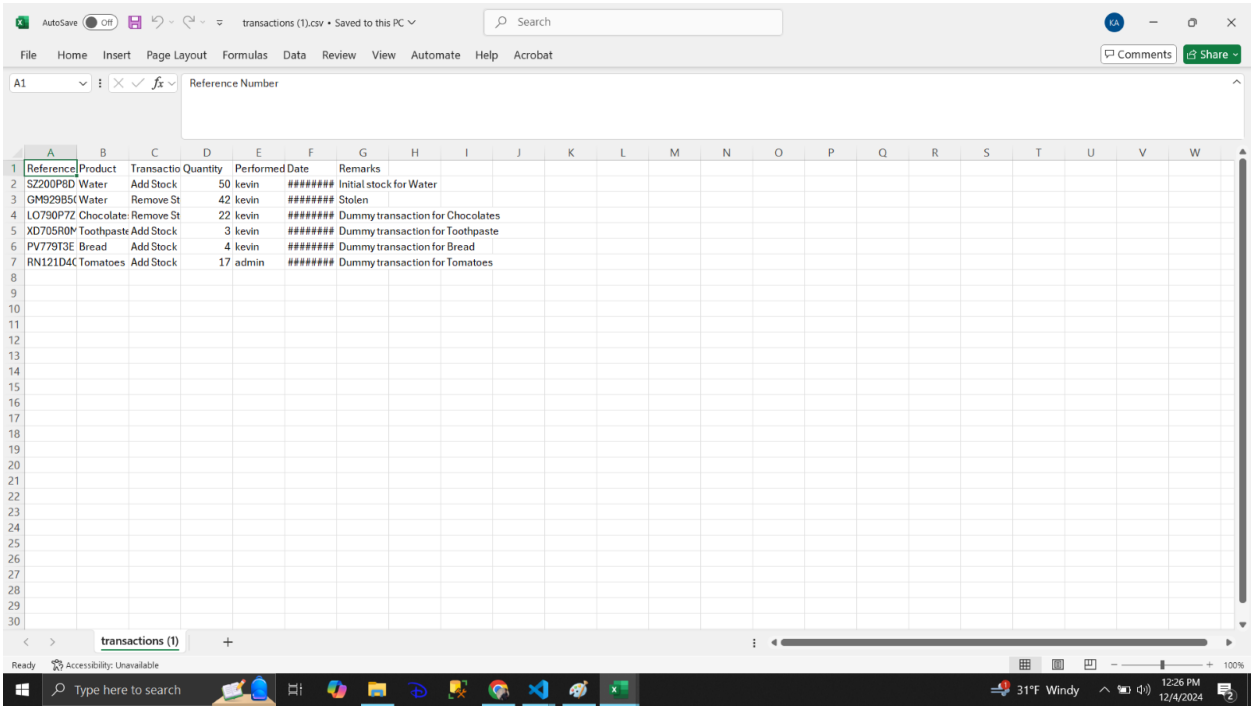

127.0.0.1:8000/categories/add/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Add Category
Category Name
Water
Save
Cancel
Type here to search
S&P 500 +0.37%
12:05 PM
12/4/2024


127.0.0.1:8000/categories/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Categories
Add Category
Category Name Actions
1 Beverages Edit Delete
2 Snacks Edit Delete
3 Dairy Edit Delete
4 Frozen Foods Edit Delete
5 Bakery Edit Delete
6 Personal Care Edit Delete
7 Household Supplies Edit Delete
8 Fruits Edit Delete
9 Vegetables Edit Delete
10 Confectionery Edit Delete
Type here to search
Snow coming
12:08 PM
12/4/2024


127.0.0.1:8000/suppliers/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Suppliers
Add Supplier
Supplier Name Email Phone Actions
1 ABC abc@gmail.com 8888888888 Edit Delete
2 Morning Glory Supplies contact@morningglory.com 1234567890 Edit Delete
3 Swift Beverages sales@swiftbeverages.com 9876543210 Edit Delete
4 Fresh Produce Co. support@freshproduce.com 5556667777 Edit Delete
5 SnackWorld hello@snackworld.com 1112223333 Edit Delete
6 Pure Dairy info@puredairy.com 4445556666 Edit Delete
7 Frozen Delights orders@frozendelights.com 7778889999 Edit Delete
8 Household Hub service@householdhub.com 2223334444 Edit Delete
9 BakeCraft bakecraft@bakery.com 9998887776 Edit Delete
10 FruitFarm support@fruitfarm.com 6665554443 Edit Delete
Type here to search
USD/CNY -0.31%
12:11 PM
12/4/2024


127.0.0.1:8000/suppliers/add/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Add Supplier
Supplier Name
Dasani
Email
dasani@gmail.com
Phone Number
666-666-6666
Address
ClevelandMuseumArt Loveland Clevelander
Cleveland
Save
Cancel
Type here to search
31°F Windy
12:16 PM
12/4/2024


127.0.0.1:8000/stock-transactions/
Inventory Management
Products
Categories
Suppliers
Stock Transactions
admin
Logout
Stock Transactions
Add Stock Transaction
Download Transactions
Transaction Type
All
Product
All
Performed By
All
From Date
mm/dd/yyyy
To Date
mm/dd/yyyy
Apply Filters
Reset
Reference Number Product Transaction Type Quantity Performed By Date Remarks Actions
1 RN121D4Q Tomatoes Added 17 admin Dec 04, 2024 Dummy transaction for Tomatoes Edit Delete
2 PV779T3E Bread Added 4 kevin Dec 04, 2024 Dummy transaction for Bread Edit Delete
3 XD705R0M Toothpaste Added 3 kevin Dec 04, 2024 Dummy transaction for Toothpaste Edit Delete
4 LO790P7Z Chocolates Removed 22 kevin Dec 04, 2024 Dummy transaction for Chocolates Edit Delete
Type here to search
31°F Windy
12:24 PM
12/4/2024


transactions (1).csv • Saved to this PC
Search
File Home Insert Page Layout Formulas Data Review View Automate Help Acrobat
Comments
Share
A1
Reference Number
Reference Product Transactio Quantity Performed Date Remarks
SZ200P8D Water Add Stock 50 kevin ######## Initial stock for Water
GM929B5(Water Remove St 42 kevin ######## Stolen
LO790P7Z Chocolate: Remove St 22 kevin ######## Dummy transaction for Chocolates
XD705R0M Toothpaste Add Stock 3 kevin ######## Dummy transaction for Toothpaste
PV779T3E Bread Add Stock 4 kevin ######## Dummy transaction for Bread
RN121D4C Tomatoes Add Stock 17 admin ######## Dummy transaction for Tomatoes
transactions (1)
Ready
Accessibility: Unavailable
100%
Type here to search
31°F Windy
12:26 PM
12/4/2024

5.2 Components Design

Static Models:

The system is organized using a **three-tier architecture** with the following layers:

1. **Presentation Layer:**
    a. **Web Interface**: Built with HTML5, CSS, and JavaScript for responsiveness.
    b. **Bootstrap Framework**: Provides consistency across devices.
2. **Application Layer:**
    a. **Controllers**:
        i. **InventoryController**: Manages CRUD operations for inventory items.
        ii. **UserController**: Handles authentication and user role management.
        iii. **ReportController**: Generates reports based on user-selected parameters.
    b. **Services**:
        i. **NotificationService**: Sends email notifications using SMTP.
        ii. **ValidationService**: Ensures data integrity.
3. **Data Layer:**
    a. **SQLite Database**: Stores all inventory, user, and supplier data.

Dynamic Models:

- **Inventory CRUD Workflow:**
    - User submits a request through the UI to add/update/delete an inventory item.
    - The request is processed by the **InventoryController**, which interacts with the **InventoryService**.
    - Data is validated and saved to the database.
    - A success/failure response is sent back to the UI.
- **Low Stock Alert Workflow:**
    - The system periodically checks inventory levels.
    - If stock is below the threshold, the **NotificationService** sends an email alert to the Inventory Manager.

5.3 Database Design

The **Inventory Management System (IMS)** database schema was meticulously designed to ensure optimal performance, scalability, and data consistency, while meeting the functional requirements of inventory tracking, stock management, and reporting.

Schema Overview

The database consists of four primary tables that interrelate to handle products, categories, suppliers, and stock transactions. The relationships between these entities are modeled to ensure referential integrity and efficient data retrieval.

Entities and Relationships

1. **Category**
    a. **Attributes**:
        i. id (Primary Key): Unique identifier for each category.
        ii. category_name (Unique, Not Null): Name of the category.
    b. **Relationships**:
        i. One-to-Many with the Product table. Each category can have multiple products.
2. **Supplier**
    a. **Attributes**:
        i. id (Primary Key): Unique identifier for each supplier.
        ii. name (Not Null): Supplier's name.
        iii. email (Unique, Not Null): Contact email.
        iv. phone_number (Not Null): Contact phone number.
    b. **Relationships**:
        i. One-to-Many with the Product table. Each supplier can supply multiple products.
3. **Product**
    a. **Attributes**:
        i. id (Primary Key): Unique identifier for each product.
        ii. name (Not Null): Product name.
        iii. category_id (Foreign Key): References the Category table.
        iv. supplier_id (Foreign Key): References the Supplier table.
        v. stock_level (Default = 0, Not Null): Current stock level.
        vi. price (Decimal, Max Digits: 10, Decimal Places: 2, Not Null): Price per unit.

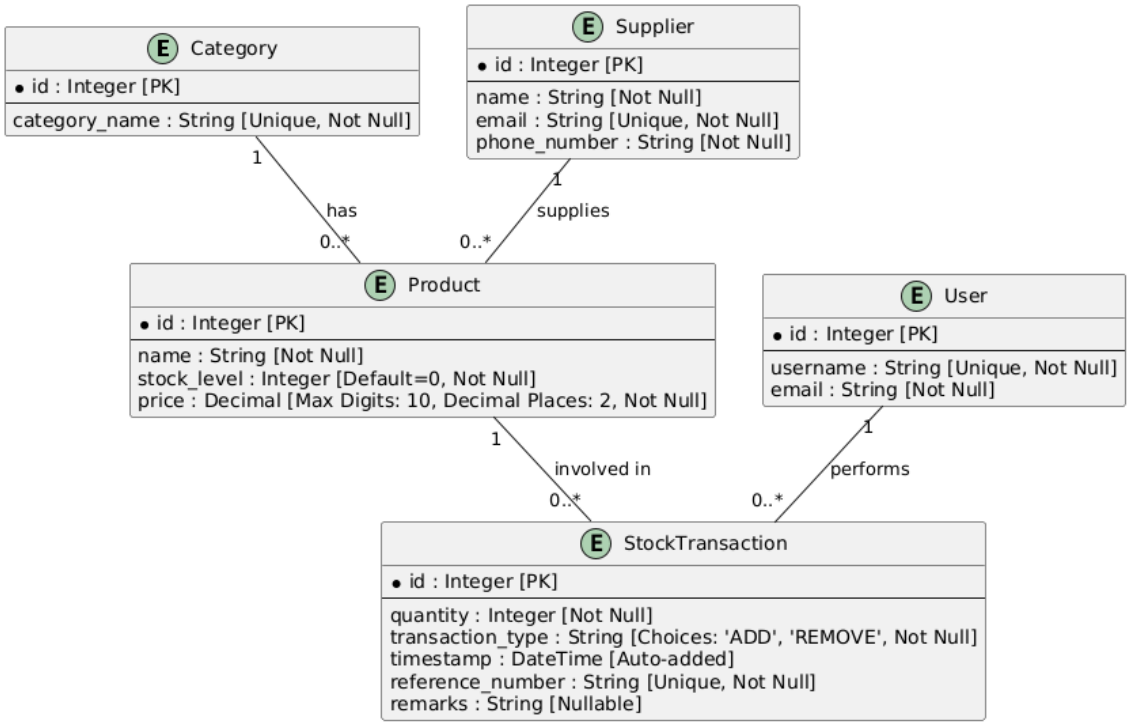
b. **Relationships**:
    i. Many-to-One with Category.
    ii. Many-to-One with Supplier.
    iii. One-to-Many with StockTransaction.

4. **StockTransaction**
    a. **Attributes**:
        i. id (Primary Key): Unique identifier for each transaction.
        ii. product_id (Foreign Key): References the Product table.
        iii. quantity (Not Null): Quantity of stock added or removed.
        iv. transaction_type (Enum, Not Null): Either "ADD" or "REMOVE".
        v. timestamp (Default = Current DateTime, Not Null): When the transaction occurred.
        vi. performed_by (Foreign Key): References the Django User table, recording the user responsible for the transaction.
        vii. reference_number (Unique, Not Null): A system-generated unique reference code.
        viii. remarks (Optional): Additional comments about the transaction.
    b. **Relationships**:
        i. Many-to-One with Product.

Entity Relationship Diagram (ERD)

The ERD illustrates the relationships between these tables:

- **Category → Product**: One-to-Many
- **Supplier → Product**: One-to-Many
- **Product → StockTransaction**: One-to-Many



Key Features and Design Rationale

1. **Data Integrity**:
    a. Foreign Key constraints ensure data consistency between Category, Supplier, Product, and StockTransaction.
    b. Unique constraints (e.g., reference_number, email) prevent duplicate records.
2. **Scalability**:
    a. The schema is designed to accommodate additional features, such as new entities for predictive analytics or supplier order history.
3. **Usability**:
    a. Logical table structures and intuitive relationships ensure that the system is easy to maintain and extend.

5.4 Traceability from Requirements to Design

The design decisions were based on the following rationale:

- **Three-Tier Architecture:** Provides clear separation of concerns, which improves maintainability and scalability.
- **Role-Based Access Control:** Enhances security by limiting access to specific features based on roles.
- **SQLite Database:** Chosen for its simplicity and suitability for small-scale projects.
- **Responsiveness and Accessibility:** Ensures that the system is usable on all devices and by users with disabilities.

5.5 Traceability from Requirements to Detailed Design Models

Requirement	Design Element
Manage Inventory Items	InventoryController, Inventory Table
Generate Inventory Reports	ReportController, Transactions Table
Role-Based Access Control	UserController, Role field in Users Table
Low Stock Alerts	NotificationService, ReorderLevel in Inventory Table
Search and Filter Inventory	Search Bar in UI, SQL Queries
Track Stock Movement	Transactions Table, Dynamic Workflow for Stock Updates
Accessibility and Responsiveness	UI Design with Bootstrap and ARIA Roles

6. Test Management

6.1 Test Cases

- Product Management**:**

Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC001	Add a new product	Navigate to Add Product → Enter all valid details → Save	Product is added successfully	Pass
TC002	Add product with missing required fields	Navigate to Add Product → Leave mandatory fields blank → Save	Error message displayed for required fields	Pass

TC003	Update product details	Select an existing product → Edit details → Save	Product details updated successfully	Pass
TC004	Delete a product	Select an existing product → Click Delete	Product deleted successfully	Pass
TC005	Search for a product	Enter product name in search → Execute	Matching product(s) displayed	Pass
TC006	Add duplicate product	Add a product with the same name and category → Save	System prevents duplicate product creation	Pass

Category Management

Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC007	Add a new category	Navigate to Add Category → Enter valid category name → Save	Category added successfully	Pass
TC008	Add duplicate category	Add a category with the same name → Save	System prevents duplicate category creation	Pass
TC009	Delete a category	Delete a category with no linked products	Category deleted successfully	Pass
TC010	Delete a category with linked products	Attempt to delete a category linked to products	System prevents deletion and shows dependency error	Pass

3. Supplier Management

Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC011	Add a new supplier	Navigate to Add Supplier → Enter all valid details → Save	Supplier added successfully	Pass
TC012	Update supplier details	Select an existing supplier → Edit details → Save	Supplier details updated successfully	Pass
TC013	Delete a supplier	Delete a supplier with no linked products	Supplier deleted successfully	Pass
TC014	Delete a supplier with linked products	Attempt to delete a supplier linked to products	System prevents deletion and shows dependency error	Pass

4. Stock Management

Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC015	Update stock quantity	Navigate to Update Stock → Select product → Enter new quantity → Save	Stock updated successfully	Pass
TC016	Check low stock alert	Reduce stock level of a product below threshold	Low stock alert generated via email	Pass

5. Reporting

Test Case ID	**Test Scenario**	**Steps to Execute**	**Expected Result**	**Status**
TC017	Generate inventory report	Navigate to Reports → Select Inventory → Generate	Inventory report displayed/downloaded	Pass
TC018	Generate supplier report	Navigate to Reports → Select Supplier → Generate	Supplier report displayed/downloaded	Pass

6. User Interface Testing

Test Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC019	Verify UI responsiveness	Access the system on various screen sizes (desktop, tablet, mobile)	UI adjusts seamlessly	Pass
TC020	Verify consistent color scheme	Navigate across pages	Color scheme remains consistent	Pass

7. System Testing

Test Case ID	Test Scenario	Steps to Execute	Expected Result	Status
TC021	Test application performance under load	Simulate 100 concurrent users adding products	System performs without significant delay	Pass
TC022	Test database integrity	Add, update, and delete records simultaneously	No data corruption occurs	Pass

6.2 Traceability to Use Cases

Each test case directly relates to a defined use case.

Use Case ID	Use Case Description	Related Test Case IDs
UC001	Add a new product	TC001, TC002, TC006
UC002	Update product details	TC003
UC003	Delete a product	TC004
UC004	Search for a product	TC005
UC005	Add a new category	TC007
UC006	Delete a category	TC009, TC010
UC007	Add a new supplier	TC011
UC008	Update supplier details	TC012
UC009	Delete a supplier	TC013, TC014
UC010	Update stock quantity	TC015
UC012	Generate inventory report	TC017

UC013	Generate supplier report	TC018
UC014	Ensure responsive UI	TC019
UC015	Ensure consistent UI design	TC020
UC016	Handle multiple concurrent users	TC021
UC017	Ensure database integrity	TC022

6.3 Techniques used for test case generation

- **We have used** Use Case Testing: Deriving test cases from functional use cases to ensure all scenarios are covered. Equivalence Partitioning: Dividing input data into valid and invalid partitions

The application passes all functional and usability tests.

6.4 Defect Reports

Defect ID	Module	Defect Description	Severity	Priority	Status	Remarks
D001	Add Product	System allows duplicate product names in the same category	Medium	High	Resolved	Added a unique product constraint
D002	Add Product	Validation error message not displayed for missing category field	Low	Mediu m	Resolved	Updated validation logic
D003	Update Stock Levels	Negative stock levels allowed for product update	High	High	Resolved	Added validation to prevent issue
D004	Generate Reports	Report displays incorrect stock levels for products updated recently	High	High	Resolved	Corrected query logic for reports
D005	Low Stock Alerts	Alerts not triggered for threshold levels when system idle	Medium	Mediu m	Resolved	Adjusted real-time alert logic

D006	Supplier Management	Supplier email field accepts invalid formats	Low	Medium	Resolved	Updated email validation regex
D007	User Interface	UI freezes when adding more than 100 products in bulk	High	High	Resolved	Optimized bulk product insertion
D008	System Performance	Response time exceeds 3 seconds under 50 simultaneous users	Medium	High	Resolved	Improved database query indexing
D009	Backup and Recovery	Data not backed up after supplier deletion	High	Medium	Resolved	Added backup trigger on delete

7. Conclusions

7.1 Outcomes of the Project

The Inventory Management System (IMS) successfully meets the primary objectives outlined at the project's inception. Key outcomes include:

1. **Feature Implementation:**
    a. The system provides comprehensive inventory management functionality, including adding, editing, and deleting inventory items, generating reports, and sending low-stock alerts.
    b. Role-based access control is effectively implemented to ensure data security and restrict access based on user roles.
2. **User Experience:**
    a. A responsive and accessible user interface improves usability across multiple devices and for users with disabilities.
    b. The system facilitates efficient workflows for inventory managers, warehouse staff, and sales managers.

3. **Operational Goals:**
    a. Real-time notifications for low-stock levels help minimize inventory shortages.
    b. The database structure enables efficient data management and ensures data consistency and reliability.

All initial goals have been achieved, making the project a functional and valuable tool for managing inventory in small to medium-sized businesses.

7.2 Lessons Learned

1. **Technical Lessons:**
    a. The importance of **scalable architectures** became evident when designing for future expansion.
    b. Leveraging **Django's modular design** proved invaluable in organizing and maintaining the codebase.
    c. Using SQLite, while simple, highlighted limitations when considering scaling the application for larger databases.
2. **Team Collaboration:**
    a. Clear communication and task delegation among team members (Jaykumar Mistry, Jaydeep Ravaliya, and Kevin Panchal) were essential for meeting deadlines.
    b. Regular progress reviews ensured alignment with project goals.
3. **Challenges Faced:**
    a. Designing an intuitive user interface required iterative feedback to meet user expectations.
    b. Ensuring the system's responsiveness across all devices demanded additional testing and fine-tuning.
    c. Balancing academic responsibilities alongside the project timeline was challenging but manageable with proper planning.

7.3 Future Development

1. **Advanced Features:**
    a. Integration of **predictive analytics** to forecast inventory demand based on historical data and trends.
    b. Addition of a **barcode scanner** module for faster inventory updates.

2. **Enhanced User Experience:**
    a. Implementation of **multi-language support** to cater to a diverse user base.
    b. Development of a **mobile application** for on-the-go inventory management.
3. **Performance Optimization:**
    a. Improving query performance for faster search and filtering of inventory items.
    b. Implementing caching mechanisms for frequently accessed data.

By incorporating these future improvements, the Inventory Management System can further evolve to meet the needs of larger organizations and provide a more comprehensive solution for inventory management.