DHA Suffa University

Department of Computer Science

Final Year Project



**Automatic Inventory Management System**

**S29F24**

**Software Design Specifications**

Submitted by

**Team Lead: Rimsha younus(cs211237)**

**Team Member: Summaya Abid (cs211240)**

**Team Member: Khunsha Noman (cs211248)**

**Team Member: Muhammad Mohsin (cs211202)**

Supervisor(s)

Co-Supervisor Name (Miss Raazia Sosan)

Supervisor Name (Sir Conrad)

**Fall 2024**

**Document Sign off Sheet**

### Document Information

|  |  |
| --- | --- |
| **Project Title** | Automatic Inventory Management System |
| **Project Code** | S29F24 |
| **Document Name** | Software Requirements Specifications |
| **Document Version** | <1.0> |
| **Document Identifier** | FYP01- S29F24 |
| **Document Status** | Initial draft |
| **Author(s)** | Rimsha, Summaya, Khunsha, Mohsin |
| **Approver(s)** | <Conrad D silva> |
| **Issue Date** | <Date of issuance of this document> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Date** |
| Rimsha Younus | Team Lead |  |  |
| Summaya Abid | Team Member 2 |  |  |
| Khunsha Noman | Team Member 3 |  |  |
| M.Mohsin | Team Member 4 |  |  |
| Conrad D Silva | Supervisor |  |  |
| Miss Raazia | Co-Supervisor |  |  |
| Sir Saad | Project Coordinator |  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 13/1/2025 | 1.0 | initial document | Rimsha, Summaya, KhMohsin, Khunsha |
|  |  |  | Khunsha, Mohsin |
|  |  |  |  |
|  |  |  |  |

**Definition of Terms, Acronyms, and Abbreviations**

*[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.]*

|  |  |
| --- | --- |
| **Term** | **Description** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Table of Contents

[1. Introduction 8](#_Toc4677804)

[1.1 Purpose of Document 8](#_Toc4677805)

[1.2 Intended Audience 8](#_Toc4677806)

[1.3 Abbreviations 8](#_Toc4677807)

[2. Overall System Description 9](#_Toc4677808)

[2.1 Project Background 9](#_Toc4677809)

[2.2 Problem Statement 9](#_Toc4677810)

[2.3 Project Scope 9](#_Toc4677811)

[2.4 Not In Scope 9](#_Toc4677812)

[2.5 Project Objectives 9](#_Toc4677813)

[2.6 Stakeholders & Affected Groups 9](#_Toc4677814)

[2.7 Operating Environment 9](#_Toc4677815)

[2.8 System Constraints 9](#_Toc4677816)

[2.9 Assumptions & Dependencies 9](#_Toc4677817)

[3. External Interface Requirements 10](#_Toc4677818)

[3.1 Hardware Interfaces 10](#_Toc4677819)

[3.2 Software Interfaces 10](#_Toc4677820)

[3.3 Communications Interfaces 10](#_Toc4677821)

[4. System Functions / Functional Requirements 11](#_Toc4677822)

[4.1 System Functions 11](#_Toc4677823)

[4.2 Use Cases 12](#_Toc4677824)

[4.2.1 List of Actors 12](#_Toc4677825)

[4.2.2 List of Use Cases 12](#_Toc4677826)

[4.2.3 Use Case Diagram 3](#_Toc4677827)

[4.2.4 Description of Use Cases 13](#_Toc4677828)

[5. Non - Functional Requirements 16](#_Toc4677829)

[5.1 Performance Requirements 16](#_Toc4677830)

[5.2 Safety Requirements 16](#_Toc4677831)

[5.3 Security Requirements 16](#_Toc4677832)

[5.4 Reliability Requirements 16](#_Toc4677833)

[5.5 Usability Requirements 16](#_Toc4677834)

[5.6 Supportability Requirements 16](#_Toc4677835)

[5.7 User Documentation 16](#_Toc4677836)

[6. References 17](#_Toc4677837)

# Introduction

## Purpose of Document

This document gives a simple and clear summary of the Automatic Inventory Management System project, including its goals, design, and plan.

The purpose is to:

* **Inform**: Explain the system's purpose, features, and technical details to stakeholders, such as developers, managers, and jury members.
* **Guide**: Act as a helpful reference for the development team during the project, keeping everything on track and aligned with the goals.
* **Clarify**: Make it easy for jury members and others to understand the system's design, structure, and the decisions made while creating it.

.

## Intended Audience

It is primarily intended for:

* **Jury Members**: To understand the objectives, structure, and design of the project.
* **Project Managers & Developers**: To follow the development process and guidelines.
* **End Users (SMB Owners/Managers)**: To understand the system’s features and how it will benefit their business.

## Project Overview

The Automatic Inventory Management System is a software tool that helps small and medium-sized businesses manage their inventory more easily. It tracks stock levels, orders, and product movements in real-time. It updates inventory when items are sold or purchased and helps businesses avoid running out of stock or having too much inventory. Users can set up alerts for low stock, manage inventory across different locations, and generate reports to see how their products are doing.

AIMS also uses AI to give helpful suggestions, predict trends, and automate tasks, making it easier for businesses to make smarter decisions and improve how they run their operations.

The system will be built with a clear, easy-to-follow design, focusing on managing inventory, creating reports, and making it simple for users to interact with the platform.

## Scope

The Automatic Inventory Management System (AIMS) will focus on providing essential features to help businesses manage their inventory efficiently. The features in scope include:

**INSCOPE:**

* **Inventory Management**: Track stock levels, manage product orders, and update inventory in real-time.
* **Sales and Purchase Management**: Keep records of sales and purchase transactions, ensuring inventory is updated automatically.
* **Reports**: Generate detailed reports on inventory performance, sales, and purchase history.
* **AI-Powered Suggestions**: AI will provide recommendations based on past data to help businesses optimize their inventory, predict trends, and improve decision-making.
* **Chatbot integration**: Chatbot integration to answer user queries and questions
* **Alerts**: Set up notifications for low stock levels to prevent stockouts or overstocking etc
* **User Management**: Allow different user roles (admin, manager, staff) to access specific features based on their responsibilities.

**OUTSCOPE**:

The following features are **out of scope** for the Automatic Inventory Management System (AIMS):

* **E-commerce Functionality**: The system will not support online retail features like product catalogs, shopping carts, or payment gateways for online stores.
* **Advanced Accounting Tools**: AIMS will not include advanced financial management features such as tax calculations or profit and loss statements.
* **Multi-location Management**: While the system will track inventory, it will not support complex multi-location or multi-warehouse management features.
* **Mobile Application**: AIMS will not offer a native mobile app for iOS or Android devices. It will be a web-based application only.
* **Enterprise-Level Features**: The system is designed for small and medium-sized businesses (SMBs), so it will not support complex operations like those required by large corporations, such as multi-department coordination or international supply chain management

# Design Considerations

This section explains important things to think about before designing the Automatic Inventory Management System.

#### Usability

* **Easy to Use**: The system should be simple enough for staff with little technical knowledge.
* **Different Roles**: Admins and staff will have different levels of access to the system.

#### Performance

* **Fast**: The system should work quickly, especially for things like updating inventory or creating invoices.
* **Handles Growth**: It should be able to handle more users and data as the business grows.

#### Security

* **Protect Data**: Sensitive information (like customer details) must be kept safe using encryption.
* **Controlled Access**: Only certain people (based on their role) should be allowed to do certain tasks.

#### Integration

* **Works with Excel**: The system should be able to export reports to Excel for easy sharing.
* **Future Connections**: The system should be able to work with other tools in the future.

#### Data Management

* **Efficient Database**: The system needs a good database to store data like sales and inventory.
* **Backup and Recovery**: Data should be backed up automatically so it’s not lost if something goes wrong.

#### AI Integration

* **Smart Suggestions**: AI will help predict sales trends and suggest inventory changes to improve business.
* **Helpful Tips**: AI will also give suggestions to improve business processes.

#### Mobile Compatibility

* **Mobile-Friendly**: The system should be easy to use on phones or tablets as well as computers.

#### System Constraints

* **Hardware/Software**: The design should work within the limits of the chosen server and software.

#### Testing and Maintenance

* **Easy to Test**: The system should be easy to check for errors and fix them.
* **Regular Updates**: The system should be updated regularly, and any problems should be tracked.

## Assumptions and Dependencies

### 4.1 Assumptions and Dependencies

This section focuses on assumptions and dependencies specifically related to the design of the Automatic Inventory Management System (AIMS). These considerations are key to successfully implementing the design and should be kept in mind during the development phase.

#### Assumptions:

* **Internet Connectivity**: The system assumes that users will have a stable internet connection for accessing the web-based platform.
* **User Familiarity**: It is assumed that users (admins and staff) have basic knowledge of using web-based systems but might need minimal training to use advanced features.
* **System Scalability**: The design assumes that the system will be able to scale as the business grows, handling an increased number of users, transactions, and inventory data.
* **Data Integrity**: The design assumes that users will input accurate data into the system, and validation checks will be in place to minimize errors.

#### Dependencies:

* **Django Framework**: The project heavily depends on Django for backend development. Its proper configuration and version compatibility are essential.
* **SQL Database**: The design assumes the use of SQL-based databases (like MySQL or PostgreSQL) for managing data storage, and proper database setup is necessary for system performance.
* **AI Integration**: The AI-driven features depend on the availability of machine learning models and algorithms for making predictive suggestions based on data.
* **External Libraries/Tools**: For reporting and export features, the system may depend on libraries like pandas or integrations with Excel for smooth data export and presentation.
* **Hardware**: The system assumes the availability of a server with enough capacity (minimum 8GB RAM, 500GB storage) for proper performance and hosting.

These assumptions and dependencies will guide the design choices and determine the tools and technologies required for the project’s success.

## Risks and Volatile Areas

### Risks and Volatile Areas

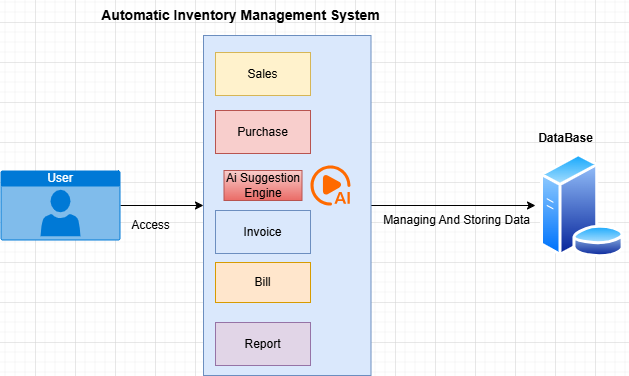
This section outlines potential risks and areas of uncertainty that may affect the design of the Automatic Inventory Management System (AIMS), and strategies to mitigate these risks.

#### Risks:

1. **Changing Business Requirements**:
   * **Impact**: As businesses evolve, new requirements may arise, such as additional reporting features, advanced inventory tracking options, or integrations with other business tools.
   * **Mitigation**: The system will be designed with flexibility in mind, allowing for easy updates and feature additions without disrupting core functionality.
2. **Technological Changes**:
   * **Impact**: Changes in the technologies used (e.g., updates to Django or database systems) may introduce incompatibilities or require redesigns.
   * **Mitigation**: Regular technology assessments and version control will help track updates and ensure the system is compatible with the latest technologies.
3. **User Adaptability**:
   * **Impact**: Users may struggle to adapt to new features, especially if the interface changes frequently or is complex.
   * **Mitigation**: The system will include clear documentation, onboarding guides, and training resources to help users smoothly transition.
4. **Data Security and Privacy**:
   * **Impact**: With sensitive business and customer data, there is a risk of breaches or misuse.
   * **Mitigation**: Strong encryption protocols and regular security audits will be incorporated into the design to ensure that data remains secure.
5. **Integration with External Tools**:
   * **Impact**: If future integrations with other tools (e.g., third-party payment systems or CRM software) are needed, they may be challenging to implement.
   * **Mitigation**: The system will be designed with open APIs and modular architecture to allow for smooth integration with external systems.
6. **Performance Issues with High User Load**:
   * **Impact**: As the number of concurrent users grows, the system may experience performance slowdowns, especially in data-heavy tasks like generating reports or processing transactions.
   * **Mitigation**: The system will be designed to handle a specified number of users, with load balancing and optimization for database queries to ensure performance remains stable.

# System Architecture

## System Level Architecture



The **System Architecture** defines how different components of the Inventory Management System (IMS) interact with each other to deliver the required functionality. Here's a breakdown:

1. **User Interface (UI)**:
   * **Responsibility**: Allows users (admins and staff) to interact with the system.
   * **Technology**: Built using HTML, CSS, JavaScript (Bootstrap for responsive design).
   * **Interaction**: User actions like adding sales, purchases, or generating reports are captured by the UI.
2. **AI Integration**:
   * **Responsibility**: Powers intelligent suggestions and predictions for better decision-making (e.g., forecasting sales, inventory management).
   * **Technology**: Python-based machine learning models.
   * **Interaction**: AI components work with the backend to provide insights that improve business operations.
3. **Reports and Analytics Module**:
   * **Responsibility**: Provides real-time sales, inventory, and purchase reports.
   * **Technology**: Integrated with Excel or CSV export functions.
4. **Database**:
   * **Responsibility**: Stores inventory, sales, purchase, and user data.
   * **Technology**: SQL-based database (MySQL or PostgreSQL).
   * **Interaction**: The backend queries the database to fetch, update, or store data.

### Workflow:

* The user interacts with the UI to initiate tasks (e.g., sales, inventory updates).
* Frontend sends requests to the backend, which processes them and communicates with the database.
* AI suggestions and insights are generated based on data and sent back to the frontend.
* Reports are generated as needed and can be exported.

This modular architecture ensures flexibility, scalability, and ease of maintenance while providing a seamless user experience.

## Software Architecture

This section explains how the system is organized and how different parts work together to provide the desired functionality.

#### Layers of the System:

1. **User Interface (UI) Layer**:
   * **Role**: This is the front-end where users interact with the system.
   * **Technologies**: HTML, CSS, JavaScript, Bootstrap.
   * **Responsibility**: Displays data and takes input from the user (e.g., sales, inventory updates).
2. **Middle Tier (Backend)**:
   * **Role**: This is the server-side part that processes business logic.
   * **Technologies**: Django (Python).
   * **Responsibility**: Handles user requests, processes data, and communicates with the database.
3. **Data Access Layer (Database)**:
   * **Role**: This is where all the business data is stored.
   * **Technologies**: MySQL or PostgreSQL.
   * **Responsibility**: Stores data (sales, inventory, invoices) and ensures it’s secure and accessible.

### ****How the Layers Work Together****:

1. **User Interaction**: Users interact with the **UI** to input or view data.
2. **Backend Processing**: The **UI** sends requests to the **Backend (Django)** for processing.
3. **Database Interaction**: The **Backend** communicates with the **Database** to store or retrieve data.
4. **Response to User**: The **Backend** sends the processed data back to the **UI** for display.

### ****Advantages****:

* **Clear Separation**: Each part (UI, Backend, Database) has its own responsibility.
* **Scalable**: The system can easily grow and add more features.
* **Secure**: Sensitive data is managed safely in the Database, and the Backend controls access.

This architecture makes the system organized, easy to update, and secure.

# Design Strategy

This section outlines key design decisions and strategies that guide the overall structure of the system. These strategies are designed to ensure flexibility, scalability, and ease of use, while considering the long-term needs of the system.

#### ****Future System Extension or Enhancement****:

* **Strategy**: The system is designed with modularity in mind, allowing for easy addition of new features (e.g., advanced reporting, multi-location support) in the future.
* **Reasoning**: Modularity ensures that new features can be integrated with minimal disruption to existing functionality, enabling smooth future enhancements.

#### ****User Interface Paradigms****:

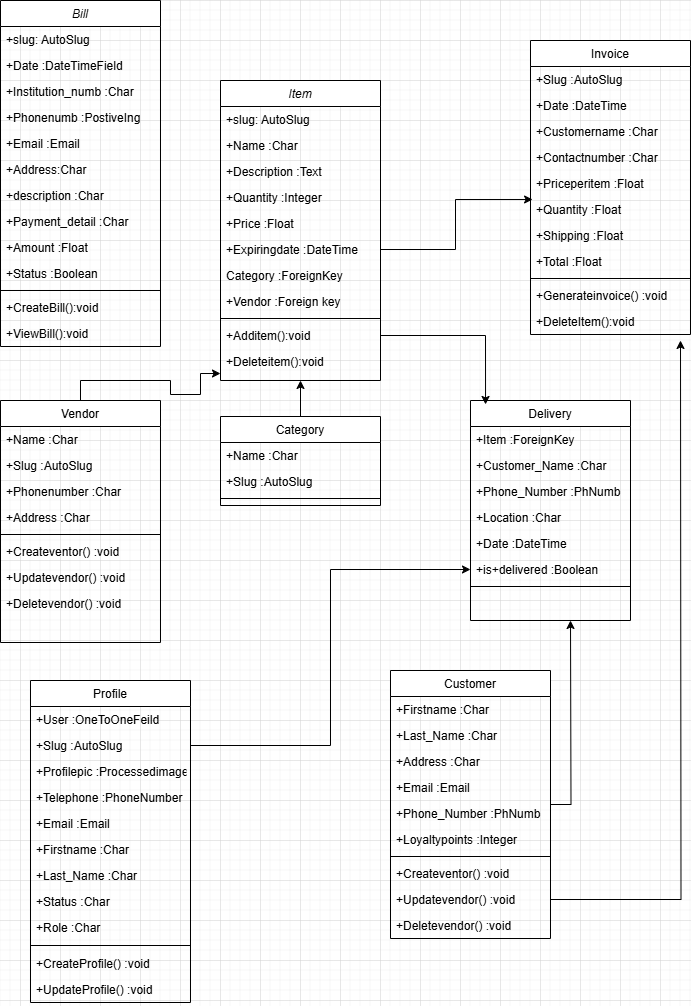
* **Strategy**: A simple, intuitive, and responsive UI is designed to ensure ease of use, even for users with limited technical expertise.
* **Reasoning**: User-friendliness is a priority, as SMBs often have employees who aren’t tech-savvy. The interface will employ drag-and-drop functionality and clear instructions to minimize training time.

#### ****Data Management (Storage, Distribution, Persistence)****:

* **Strategy**: Use a centralized relational database (SQL) to store all business-critical data (sales, purchases, inventory).
* **Reasoning**: SQL databases provide a strong, reliable way to store and manage data. They also support data integrity, fast querying, and scalability.

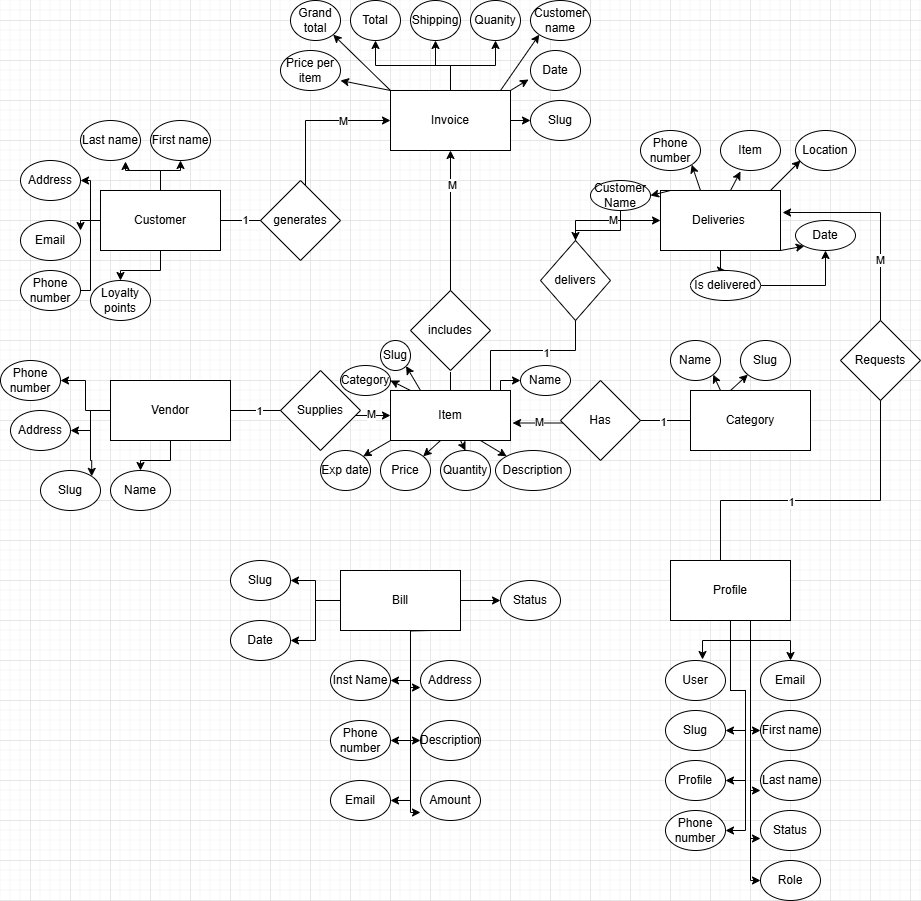
# Detailed System Design

## Design Class Diagram



## Database Design

### ER Diagram



### Data Dictionary

#### Data 1 Bill

|  |  |
| --- | --- |
| **Name** | **Bill - Represents a bill with details and payment status.** |
| Alias | Invoice, Receipt |
| Where-used/how-used | Used in the billing module to store and retrieve bill details for payments. |
| Content description | = slug + date + institution\_name + phone\_number + email + address + description + payment\_details + amount + status |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| id | Unique identifier for each bill | Integer | - | No | - | PK |
| slug | Unique slug for the bill | String | 255 | No | Auto-generated |  |
| date | Date of bill creation | DateTime | - | No | Current timestamp |  |
| institution\_name | Name of the institution | String | 30 | No | - |  |
| phone\_number | Contact number of the institution | Integer | - | Yes | Null |  |
| email | Email of the institution | Email | 255 | Yes | Null |  |
| address | Address of the institution | String | 255 | Yes | Null |  |
| description | Description of the bill | String | 255 | Yes | Null |  |
| payment\_details | Payment details | String | 255 | No | - |  |
| amount | Total amount owed | Float | - | No | - |  |
| status | Payment status (paid/unpaid) | Boolean | - | No | False |  |

#### Data 2 : Invoice

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Bill |
| Where-used/how-used | Used to generate and track customer purchases and payments. |
| Content description | = slug + date + customer\_name + contact\_number + item + price\_per\_item + quantity + shipping + total + grand\_total |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| id | Unique identifier for the invoice | Integer | - | No | - | PK |
| slug | Unique slug for the invoice | String | 255 | No | Auto-generated |  |
| date | Date of invoice creation | DateTime | - | No | Current timestamp |  |
| customer\_name | Name of the customer | String | 30 | No | - |  |
| contact\_number | Customer's contact number | String | 13 | No | - |  |
| item\_id | Foreign key to Item table | Integer | - | No | - | FK |
| price\_per\_item | Price per item | Float | - | No | - |  |
| quantity | Number of items purchased | Float | - | Yes | 0.0 |  |
| shipping | Shipping and handling charges | Float | - | No | - |  |
| total | Total amount (excluding shipping) | Float | - | No | Auto-calculated |  |
| grand\_total | Total amount (including shipping) | Float | - | No | Auto-calculated |  |

#### Data 3 Category

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Category |
| Where-used/how-used | Used to categorize products for inventory management, organization in product catalogs |
| Content description |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| name | Name of the category | CharField | 50 | No | None | PK |
| slug | Slug based on category name | AutoSlugField | 255 | No | None |  |

#### 

#### Data 4 Item

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Item |
| Where-used/how-used | Used to represent individual products in the inventory, track sales, and manage stock levels |
| Content description |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| slug | Unique slug based on item name | AutoSlugField | 255 | No | None | PK |
| name | Name of the item | CharField | 50 | No | None |  |
| description | Description of the item | TextField | 256 | No | None |  |
| category | Foreign key to the item category | ForeignKey | N/A | No | None | FK |
| quantity | Quantity in stock | IntegerField | N/A | No | 0 |  |
| price | Price of the item | FloatField | N/A | No | 0 |  |
| expiring\_date | Expiration date of the item | DateTimeField | N/A | Yes | None |  |
| vendor | Foreign key to the vendor | ForeignKey | N/A | Yes | None | FK |

#### Data 5 Delivery

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Delivery |
| Where-used/how-used | Used to track and manage the shipment and delivery of customer orders. |
| Content description |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| item | Foreign key to the delivered item | ForeignKey (Item) | N/A | Yes | None | FK |
| customer\_name | Name of the customer | CharField | 30 | Yes | None |  |
| phone\_number | Customer's phone number | PhoneNumberField | N/A | Yes | None |  |
| location | Delivery location | CharField | 20 | Yes | None |  |
| date | Delivery date | DateTimeField | N/A | No | None |  |
| is\_delivered | Delivery status | BooleanField | N/A | No | False |  |

#### Data 6 Profile

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Profile |
| Where-used/how-used | Used to store and manage user or customer information, including preferences, contact details, and account settings. |
| Content description |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| user | One-to-one relation to the User | ForeignKey (User) | N/A | No | None | PK |
| slug | Unique account ID | AutoSlugField | 255 | No | None |  |
| profile\_picture | User's profile picture | ProcessedImageField | N/A | Yes | 'profile\_pics/default.jpg' |  |
| telephone | User's telephone number | PhoneNumberField | N/A | Yes | None |  |
| email | User's email address | EmailField | 150 | Yes | None |  |
| first\_name | User's first name | CharField | 30 | Yes | None |  |
| last\_name | User's last name | CharField | 30 | Yes | None |  |
| status | User's status | CharField | 12 | No | 'INA' |  |
| role | User's role | CharField | 12 | Yes | None |  |

#### Data 7 Vendor

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Vendor |
| Where-used/how-used | Used to store vendor details for generating invoices and managing purchase orders |
| Content description |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| name | Name of the vendor | CharField | 50 | No | None | PK |
| slug | Slug based on vendor name | AutoSlugField | 255 | No | None |  |
| phone\_number | Vendor's phone number | BigIntegerField | N/A | Yes | None |  |
| address | Vendor's address | CharField | 50 | Yes | None |  |

#### Data 8 Customer

|  |  |
| --- | --- |
| **Name** | **Invoice - Represents invoices for purchases.** |
| Alias | Customer |
| Where-used/how-used | Used to store and manage customer information for generating invoices, processing orders, and tracking loyalty points. |
| Content description |  |

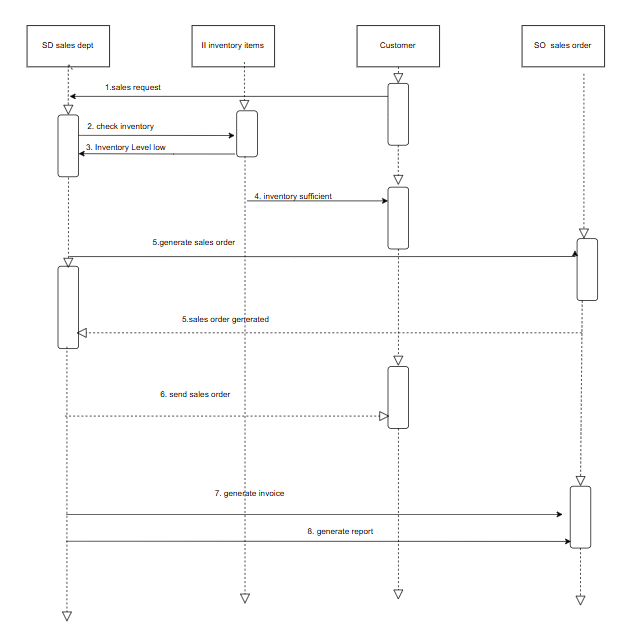
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| first\_name | Customer's first name | CharField | 256 | No | None | PK |
| last\_name | Customer's last name | CharField | 256 | Yes | None |  |
| address | Customer's address | TextField | 256 | Yes | None |  |
| email | Customer's email | EmailField | 256 | Yes | None |  |
| phone | Customer's phone number | CharField | 30 | Yes | None |  |
| loyalty\_points | Customer's loyalty points | IntegerField | N/A | No | 0 |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## Application Design

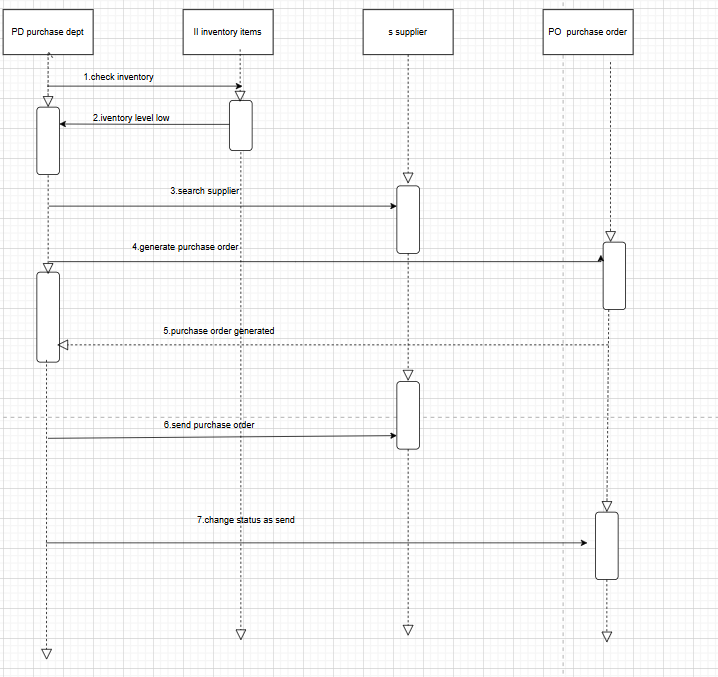
### Sequence Diagram

#### <Sequence Diagram 1> Sales



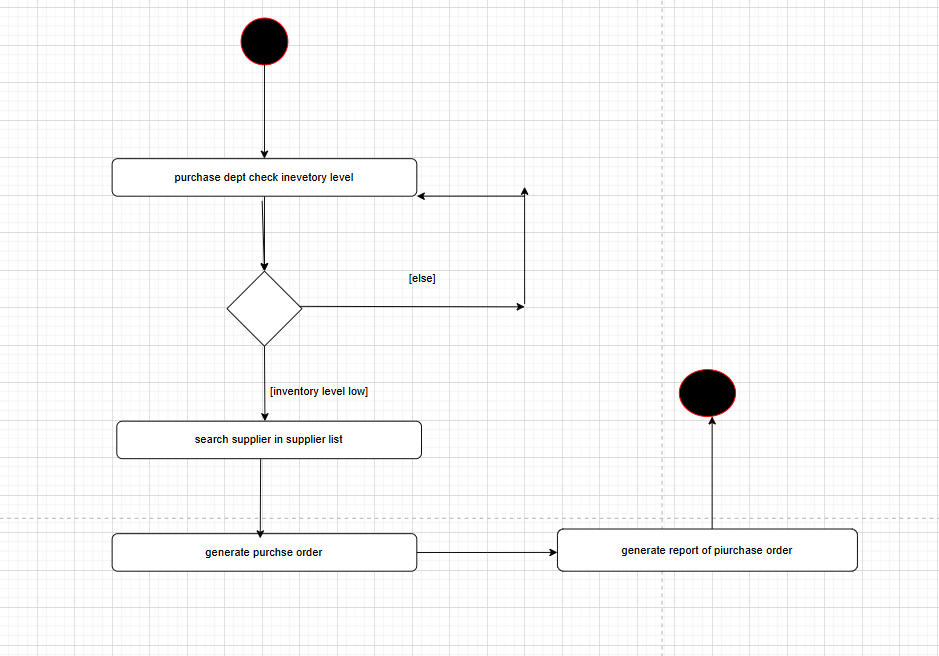
Description: This diagram shows the process of handling a sales order. First, the sales department makes a sales request. The inventory is checked to ensure availability. If the inventory is sufficient, a sales order is generated and sent to the customer. Then, an invoice is created, and finally, a report is generated to summarize the transaction. If the inventory is low, the process halts.

#### <Sequence Diagram 2> Pruchase

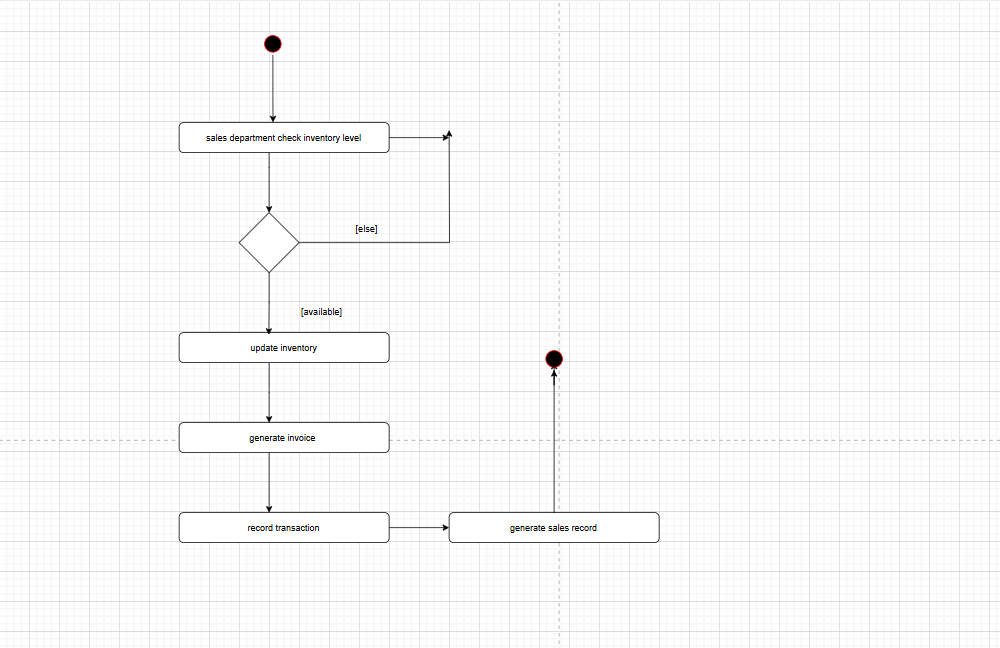


Description: The Purchase Department (PD) starts by checking the inventory levels to see if there’s enough stock. If the inventory is low, the system will alert them with a "low inventory" status. Next, the system searches through the supplier database to find a supplier who can fulfill the order. Once a supplier is selected, a purchase order is created by the system. The purchase order is then sent to the supplier to confirm the order. After the supplier receives and processes the order, an invoice is generated as part of the transaction to complete the process.This process involves checking stock levels, selecting a supplier, creating an order, and receiving an invoice to finalize the transaction.

### State Diagram



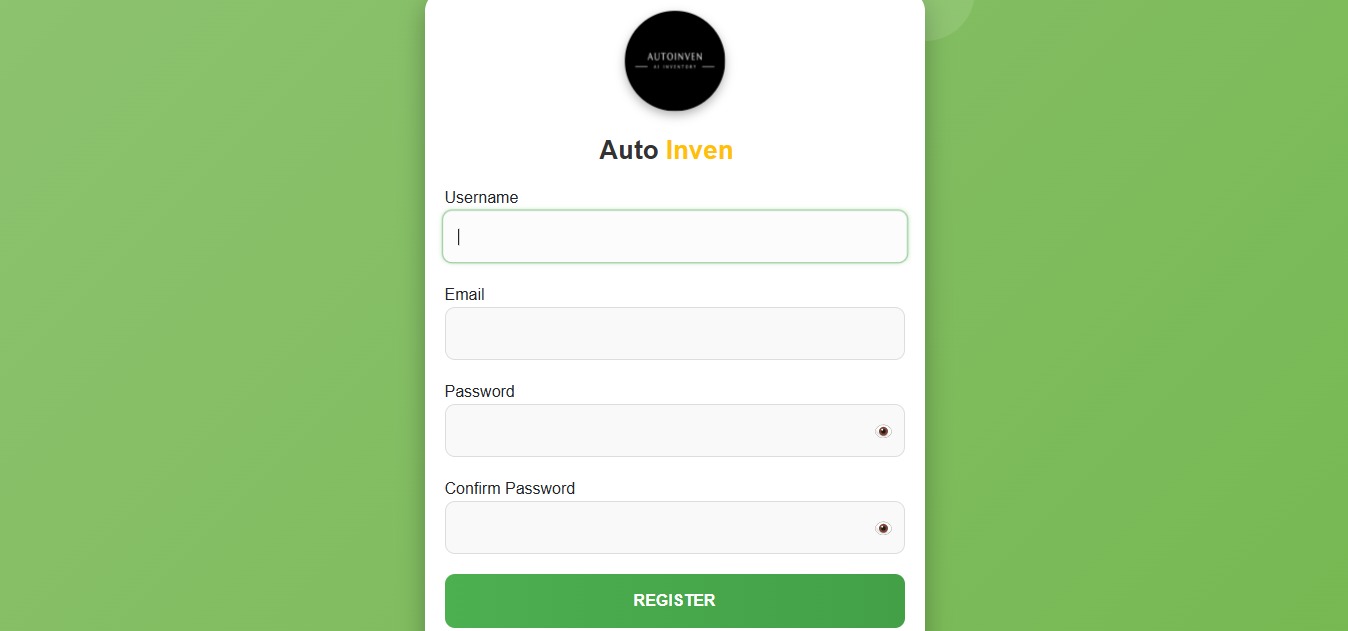
Description : The state diagram outlines the workflow of the **Purchase Department** for managing inventory. When inventory levels fall below a set threshold, the system searches the supplier list to identify the appropriate supplier. Once a supplier is selected, a purchase order is generated with product details, quantity, and supplier information. Finally, the system generates a purchase report summarizing the order, including costs and supplier details, completing the process. This ensures efficient handling of low inventory situations.



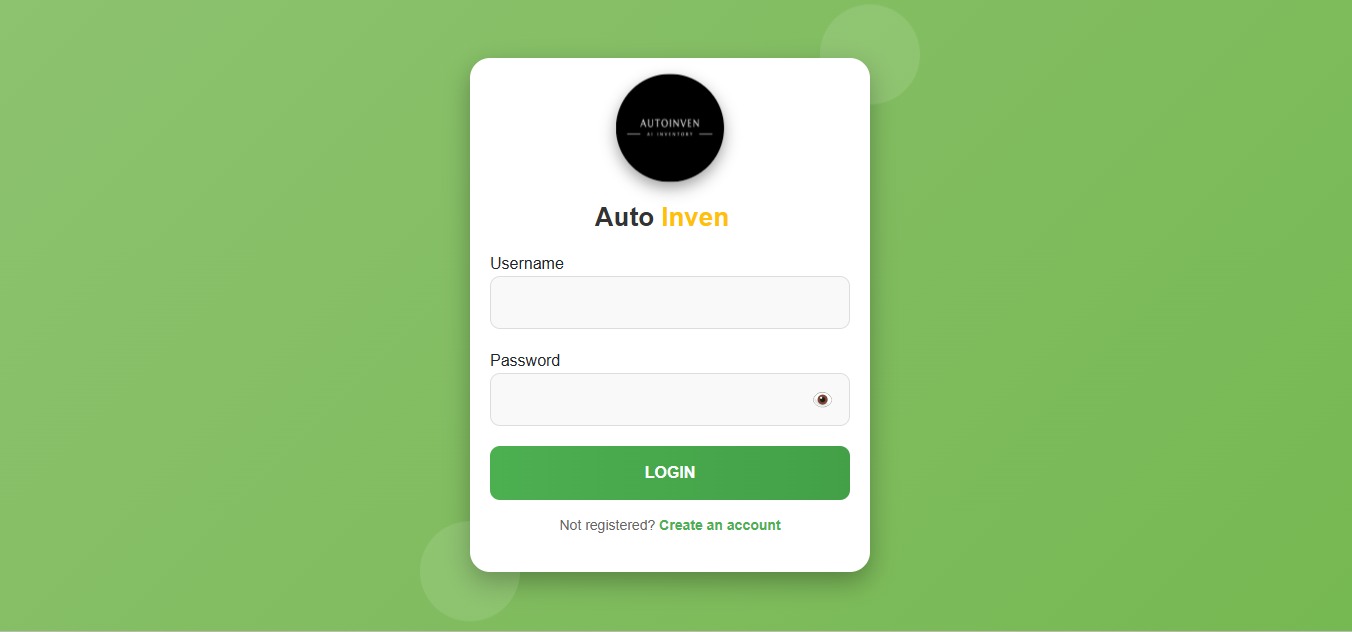
Description: The state diagram illustrates the workflow of the **Sales Department** for processing customer orders. When a sales request is received, the system checks inventory levels to ensure availability. If the requested items are in stock, the system updates the inventory, generates an invoice for the sale, and records the transaction. Finally, a sales report is generated, summarizing the transaction details, including product, quantity, and total amount, ensuring smooth and efficient sales operations.

## GUI Design

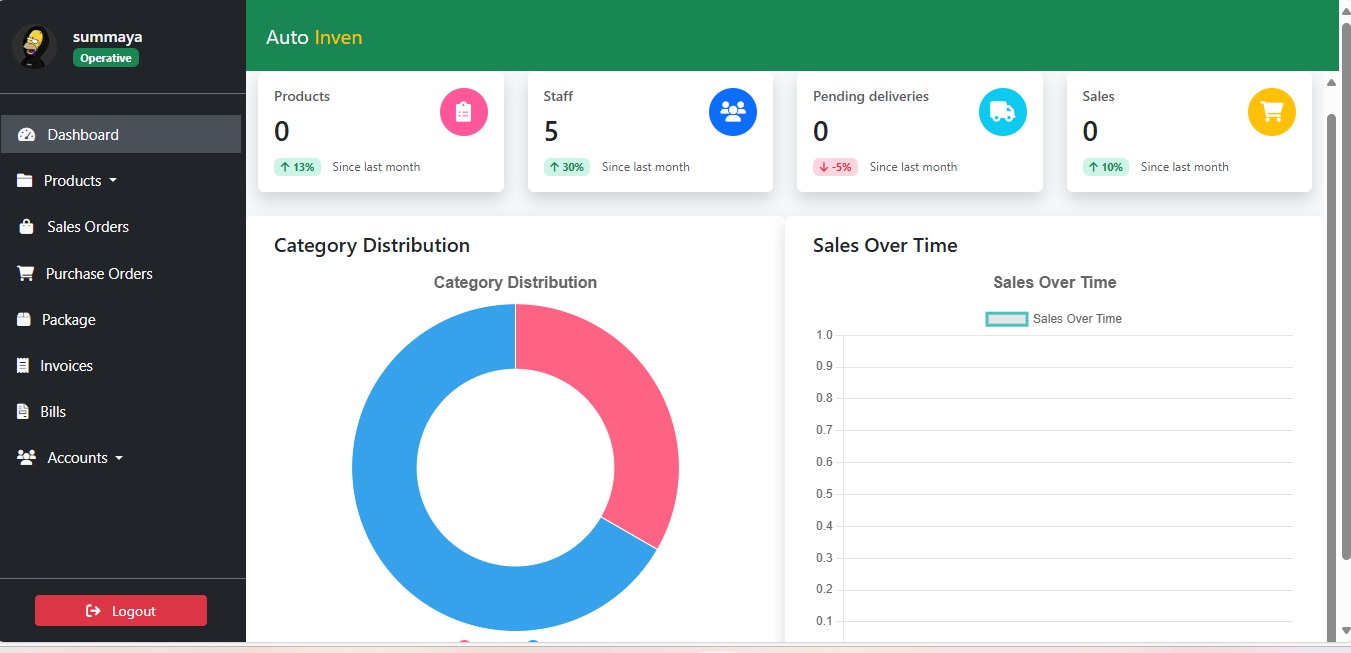
**REGISTER**



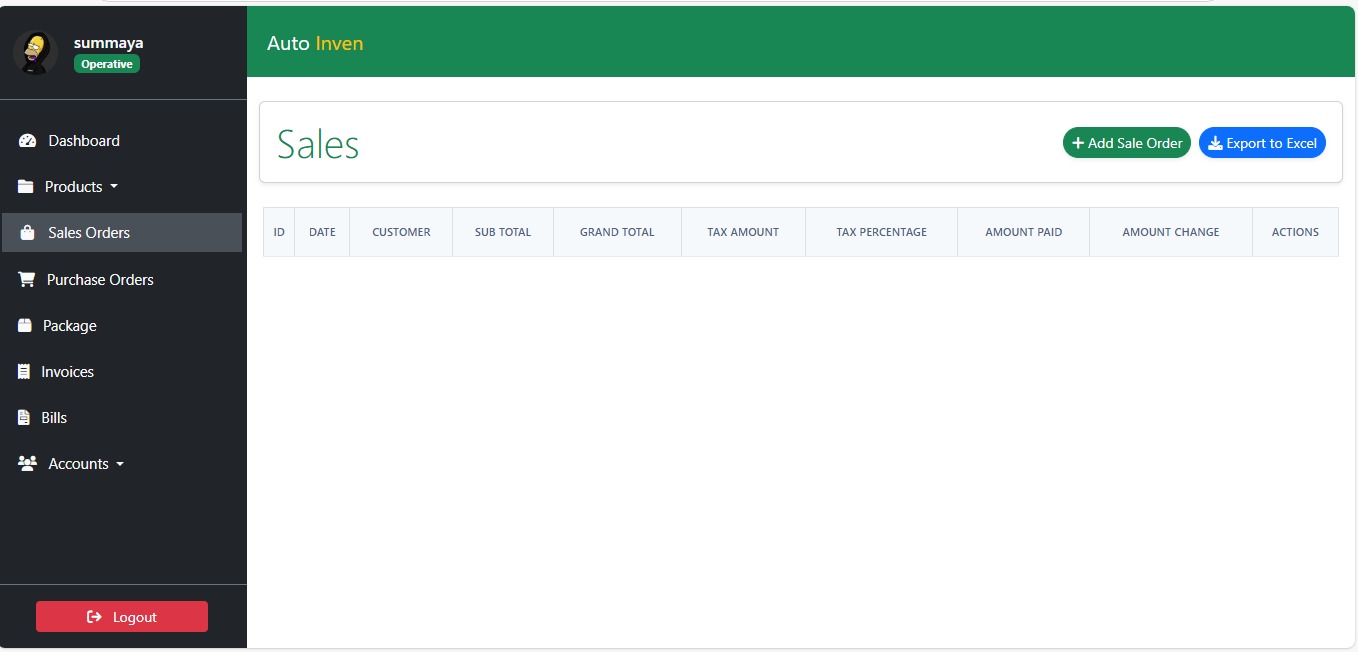
**LOGIN SCREEN**



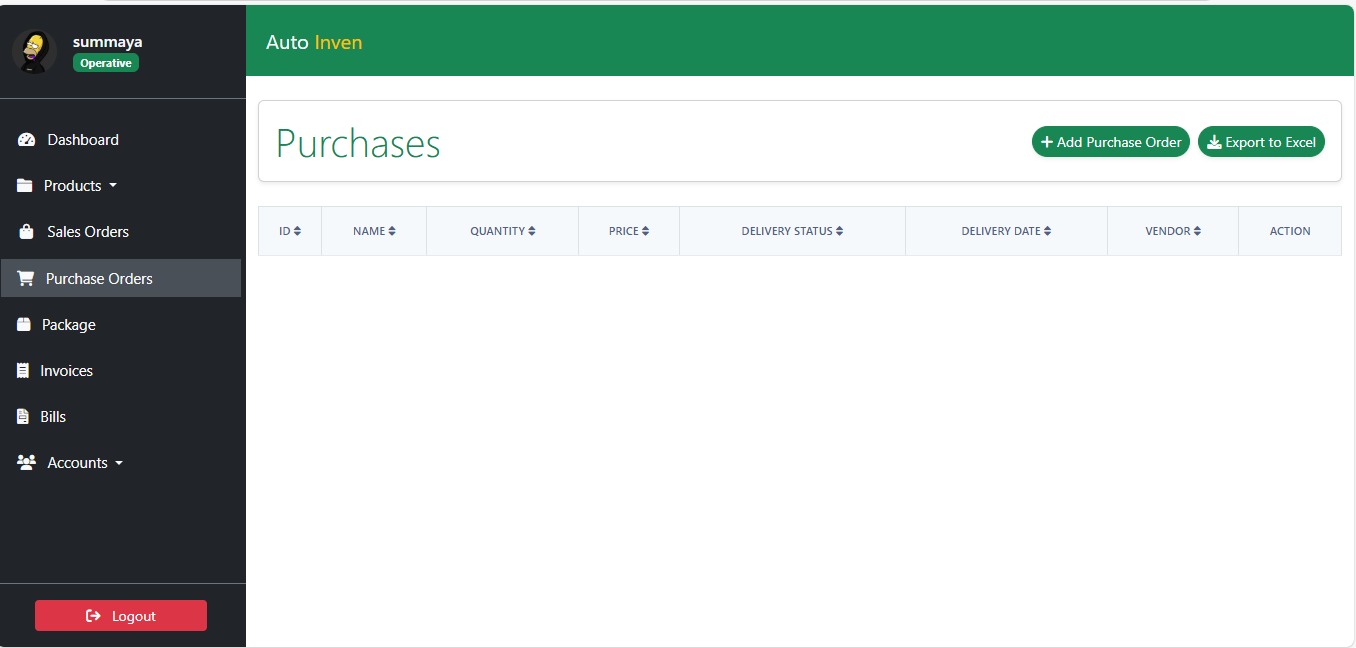
**DASHBOARD**



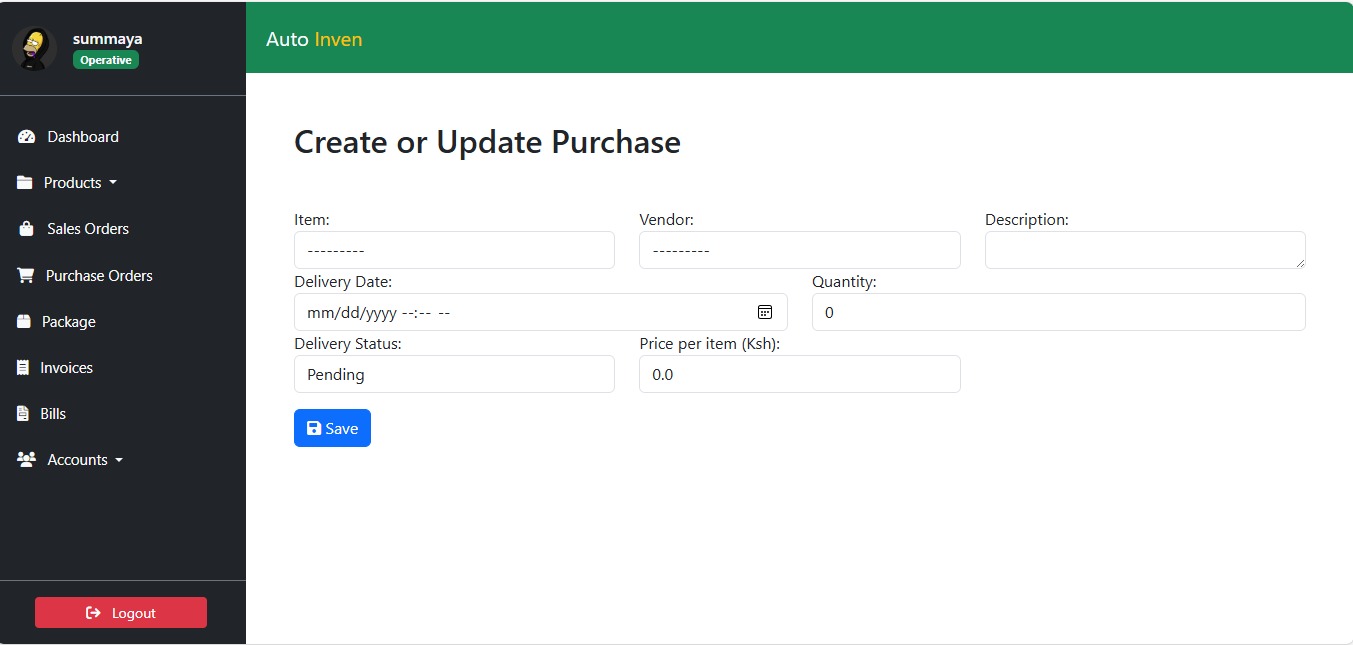
**SALES ORDER**



**PURCHASE ORDER**



**UPDATE PURCHASE**



# References

[This section should provide a complete list of all documents referenced at specific point in time. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained (This section is like the bibliography in a published book)].

# 