Inventory Management System - Project Report

---

1. Project Title

Inventory Management System Using Python Flask

---

2. Abstract

The **Inventory Management System** is a web-based application designed to efficiently manage products, locations, and stock movements within a warehouse. This system allows administrators and users to track inventory in real-time, log incoming and outgoing product movements, and generate reports on stock availability. Built with **Python Flask**, **SQLAlchemy**, and **Flask-Login**, the system provides a secure, easy-to-use interface for managing warehouses and products.

---

3. Introduction

Inventory management is a critical aspect of warehouse operations. Manual tracking of products and stock levels is inefficient, error-prone, and time-consuming. This project automates inventory tracking, product movement, and stock reporting. It provides:

- A dashboard summarizing current stock and product movements.
- User authentication for secure access.
- Product and location management.
- Product movement tracking between different locations.
- Stock reports with incoming, outgoing, and available quantities.

---

4. Objectives

1. Provide an easy-to-use web interface for warehouse inventory management.
2. Ensure secure access with user authentication and role management.
3. Track the movement of products between different warehouse locations.
4. Maintain accurate stock levels and generate detailed reports.
5. Automate repetitive inventory tasks to reduce manual errors.

---

5. Tools & Technologies Used

Category	Tools/Technologies
Backend Framework	Python Flask
Frontend	HTML, CSS, Bootstrap, Jinja2

Category	Tools/Technologies
Database	SQLite / MySQL / PostgreSQL
ORM	SQLAlchemy
Authentication	Flask-Login
Password Security	Werkzeug Security (hashing)
Form Handling	Flask-WTF
IDE/Editor	VS Code / PyCharm / Sublime Text
Deployment	Localhost / Optional: Heroku/Vercel

---

6. Modules Description

6.1 User Authentication Module

- Users can **register** and **login**.
- Passwords are securely stored using **hashing**.
- Only authenticated users can access dashboard and other features.
- **Logout** option is available.

6.2 Product Management Module

- Users can **add**, **edit**, and **delete** products.
- Each product has:
  - **ID**
  - **Name**
  - **Description**
- Product data is stored in the database and linked with movements.

6.3 Location Management Module

- Users can **add**, **edit**, and **delete** warehouse locations.
- Each location has:
  - **ID**
  - **Name**
- Locations are used in product movements to track stock transfers.

6.4 Product Movement Module

- Tracks movement of products **between locations** or **incoming/outgoing stock**.
- Users can **add**, **edit**, and **delete** movements.
- Each movement has:
    - **Product**
    - **From Location** (can be empty)
    - **To Location** (can be empty)
    - **Quantity**
    - **Timestamp**
- Validations ensure numeric quantities and correct selection of locations.

6.5 Dashboard & Reporting

- The dashboard provides:
    - Total products
    - Total locations
    - Recent movements
- The **Report module** calculates:
    - Incoming stock
    - Outgoing stock
    - Current stock per location
- Users can view detailed reports to analyze inventory levels.

---

7. Database Design

Entities and Relationships:

1. **User**
    - id (PK)
    - name
    - email
    - password
2. **Product**
    - id (PK)
    - name

- description

3. **Location**
    - id (PK)
    - name
4. **ProductMovement**
    - id (PK)
    - product_id (FK → Product.id)
    - from_location (FK → Location.id, nullable)
    - to_location (FK → Location.id, nullable)
    - qty
    - timestamp

Relationships:

- One Product can have multiple movements.
- One Location can be a source or destination for multiple movements.

---

8. Flow of the Application

1. **User Registration/Login** → Access Control
2. **Dashboard** → Overview of products, locations, movements
3. **Products** → Add/Edit/Delete products
4. **Locations** → Add/Edit/Delete locations
5. **Movements** → Add/Edit/Delete movements of stock
6. **Reports** → Generate real-time stock reports

---

9. Advantages

- Real-time tracking of inventory.
- Reduces manual errors and paperwork.
- Secure authentication and role management.
- Easy-to-use interface with clean navigation.
- Generates accurate reports for warehouse analysis.

---

10. Limitations

- Basic user roles; no admin vs regular user distinctions in current version.

- Not optimized for large-scale warehouse networks.
- No barcode/RFID integration.
- Reports are limited to quantity; no graphical visualization yet.

---

11. Use Cases

1. **Small to Medium Warehouses**
    - Manage daily product stock and movements.
2. **E-commerce Startups**
    - Track inventory across multiple storage locations.
3. **Retail Shops**
    - Monitor products moving between stores or branches.

---

12. Screenshots (Example)

- **Login/Register Page**
- **Dashboard**
- **Products Page**
- **Locations Page**
- **Add/Edit Movement Page**
- **Stock Report Page**

(You can include actual screenshots from your running Flask app.)

---

13. Future Enhancements

- Add **admin and user roles** with permissions.
- Integrate **barcode/RFID scanning** for faster stock updates.
- Add **graphical reports and analytics dashboards**.
- Enable **exporting reports to CSV or PDF**.
- Add **email notifications** for low stock alerts.

---

14. Conclusion

The **Smart Warehouse Inventory Management System** successfully automates warehouse operations, providing a **secure, reliable, and user-friendly system** for tracking products and stock levels. With real-time data and comprehensive reports, it reduces errors, saves time, and improves operational efficiency. Future improvements could extend it to larger warehouses with advanced analytics and automated alerts.