Université Euro Méditerranéenne Fès
Euro Mediterranean University of Fez
Ecole d'Ingénierie Digitale et d'Intelligence Artificielle
(EIDIA)

End-of-Module Project

Field: : 1st Year AI Engineering Cycle

Semestre : 4

Module : SQL and NoSQL Databases

Theme :

Inventory Management System

Supervised by:

Pr. Z. ALAMI

Prepared by the Students

- Mr. Gebli Achraf

-Mr.Jabbar Aymen

1. Project Overview

The Inventory Management System is a web-based application designed to help businesses manage their inventory efficiently. It allows users to add, edit, delete, and view products, track transactions, and generate reports. The system is built using Flask (Python) for the backend, MongoDB for the database, and Bootstrap for the frontend.

2. Key Features

The system provides the following features:

2.1 Product Management

Add Product: Users can add new products with details such as name, quantity, price, and category.

Edit Product: Users can update product details.

Delete Product: Users can remove products from the inventory.

View Products: Users can view a list of all products in the inventory.

2.2 Search Functionality

Users can search for products by name in the View Products page.

2.3 Transaction Management

Users can view a list of transactions, including details like product ID, quantity, price, and timestamp.

2.4 Reporting

Users can generate a Product Report in PDF format, which includes details of all products in the inventory.

2.5 User Authentication

Admin login functionality is implemented to restrict access to authorized users only.

3. Technologies Used

Frontend

HTML5: For structuring the web pages.

CSS3: For styling the application (custom styles and Bootstrap).

Bootstrap: For responsive design and pre-built components.

Backend

Flask (Python): For handling server-side logic and routing.

MongoDB: For storing product and transaction data.

FPDF: For generating PDF reports.

4. Project Structure

Frontend Files

index.html: Dashboard page with navigation links.

add_product.html: Form to add new products.

edit_product.html: Form to edit existing products.

view_products.html: Displays a list of products with search functionality.

transactions.html: Displays a list of transactions.

search_product.html: Displays search results for products.

landing.html: Landing page with a "Get Started" button.

login.html: Admin login page.

Backend Files

app.py: Flask application with routes for handling requests (e.g., adding products, editing products, generating reports).

Database

MongoDB Collections:

products: Stores product details (name, quantity, price, category).

transactions: Stores transaction details (product ID, quantity, price, timestamp).

admins: Stores admin credentials (username and password).

5. How It Works

Admin Login:

Admins log in using their credentials (username: admin, password: admin123).

Unauthorized users are redirected to the login page.

Dashboard:

After logging in, admins are directed to the dashboard, where they can:

Add products.

View products.

View transactions.

Generate reports.

Product Management:

Admins can add, edit, or delete products.

Products are stored in the products collection in MongoDB.

Transaction Management:

Admins can view a list of transactions stored in the transactions collection.

Search Functionality:

Admins can search for products by name in the View Products page.

Reporting:

Admins can generate a PDF report of all products in the inventory.

6. Challenges Faced

Search Functionality:

Implementing a dynamic search feature that filters products in real-time using JavaScript.

PDF Generation:

Generating a PDF report using the FPDF library and ensuring the file is deleted after download.

User Authentication:

Implementing session-based authentication to restrict access to authorized users.

7. Future Improvements

Enhanced User Roles:

Add support for multiple user roles (e.g., admin, manager, staff) with different permissions.

Advanced Search:

Allow users to search by multiple criteria (e.g., category, price range).

Barcode Integration:

Add support for scanning barcodes to add or update products.

Email Notifications:

Send email alerts when product quantities fall below a certain threshold.

Mobile App:

Develop a mobile version of the application for on-the-go inventory management.

Data Visualization:

Add charts and graphs to visualize inventory trends and sales data.

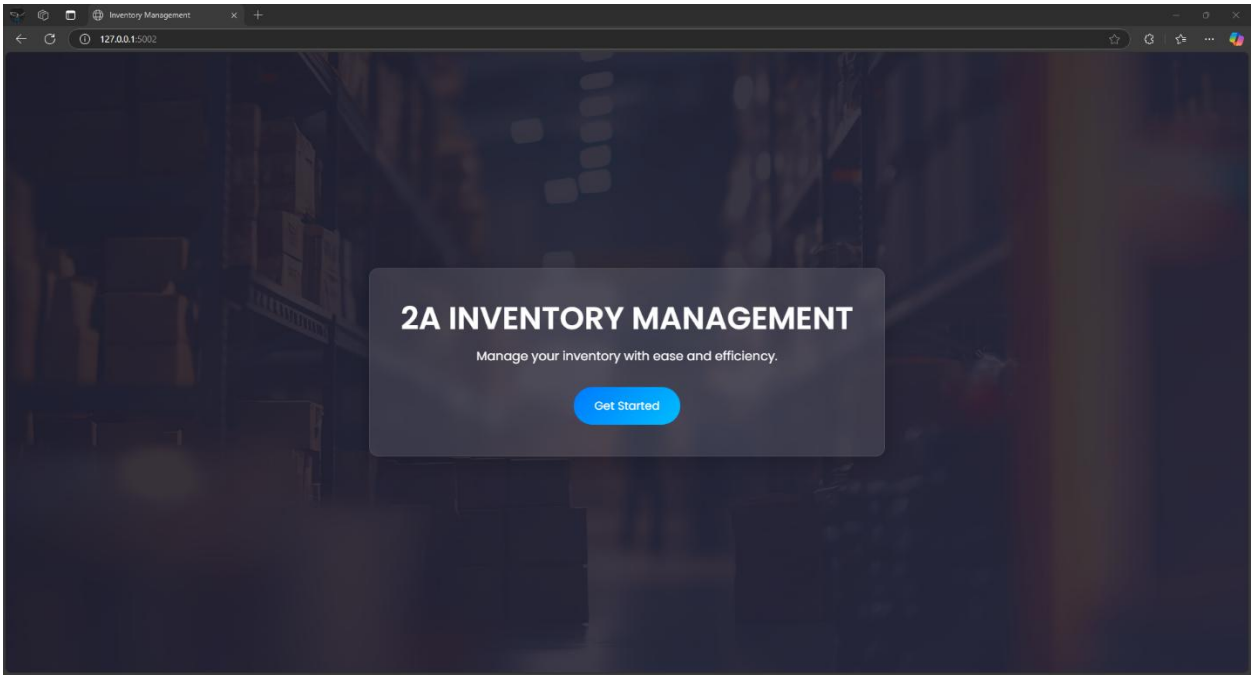
8. Conclusion

The Inventory Management System is a robust and user-friendly application that simplifies inventory management for businesses. It leverages modern web technologies to provide a seamless experience for users. With its modular design and scalable architecture, the system can be easily extended to include additional features in the future.

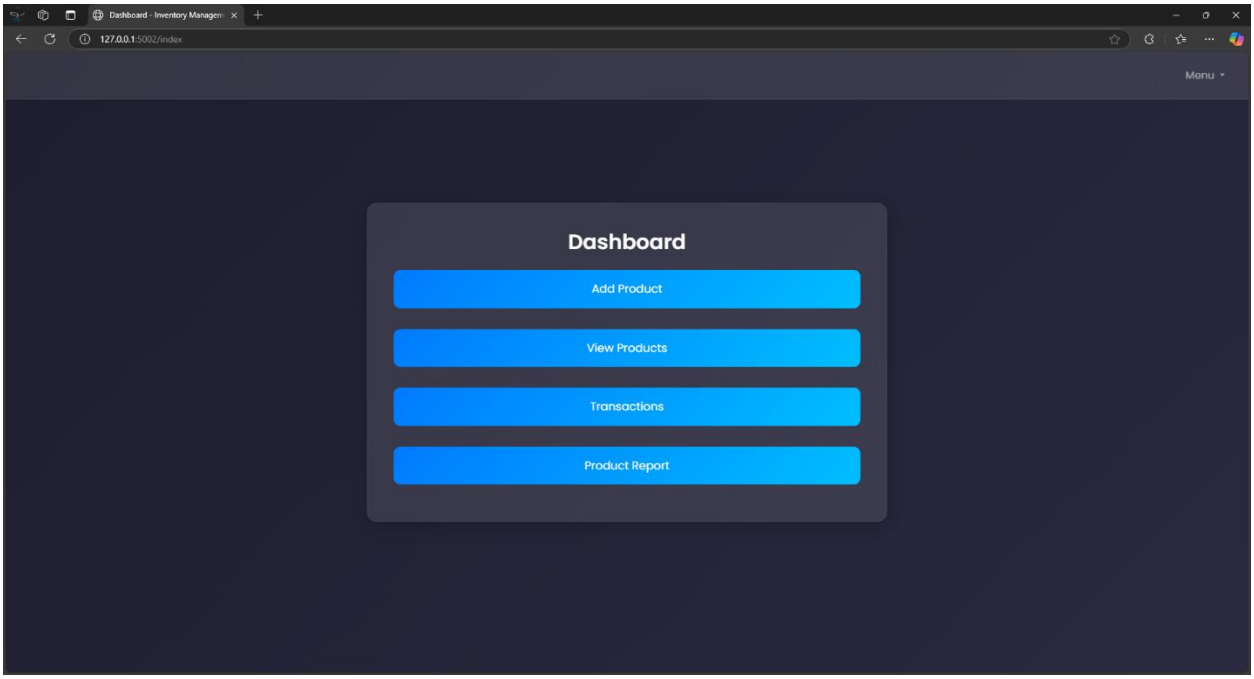
9. Screenshots

Here are some screenshots of the application:

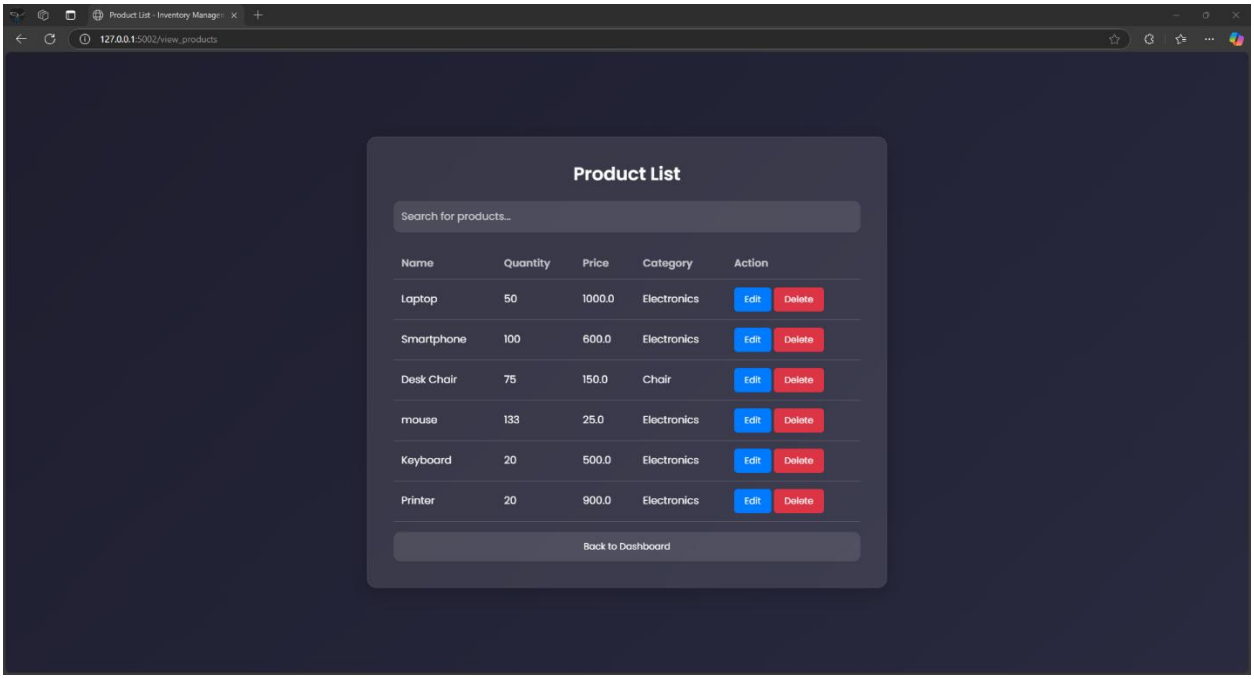
Landing Page:



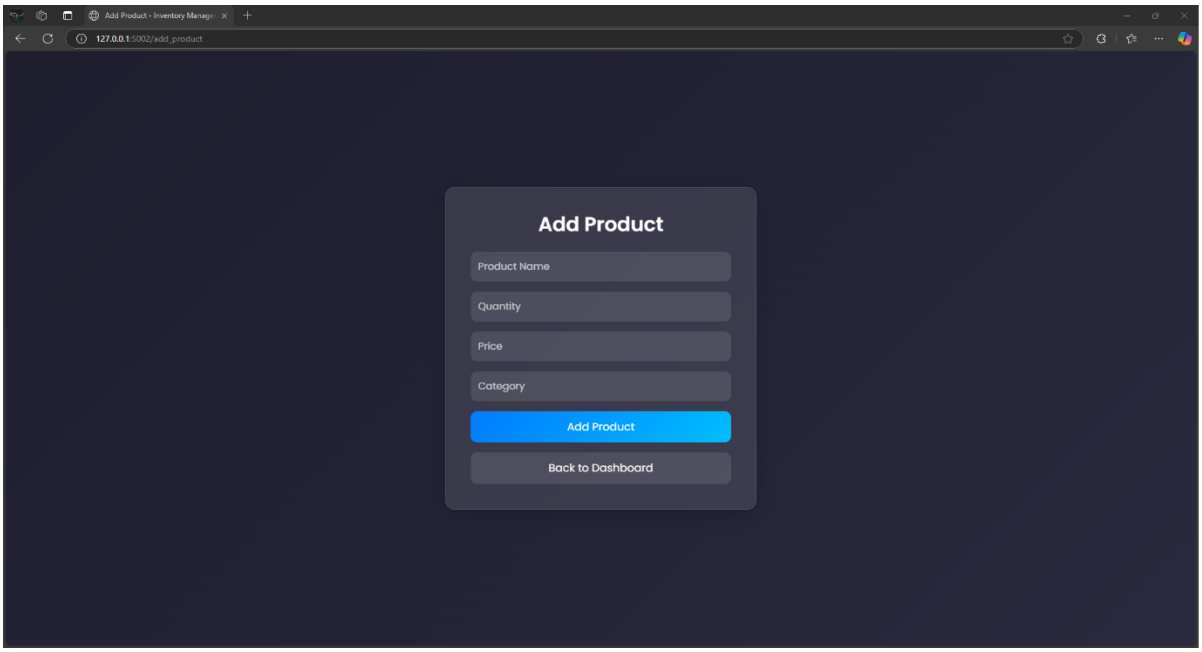
Dashboard:



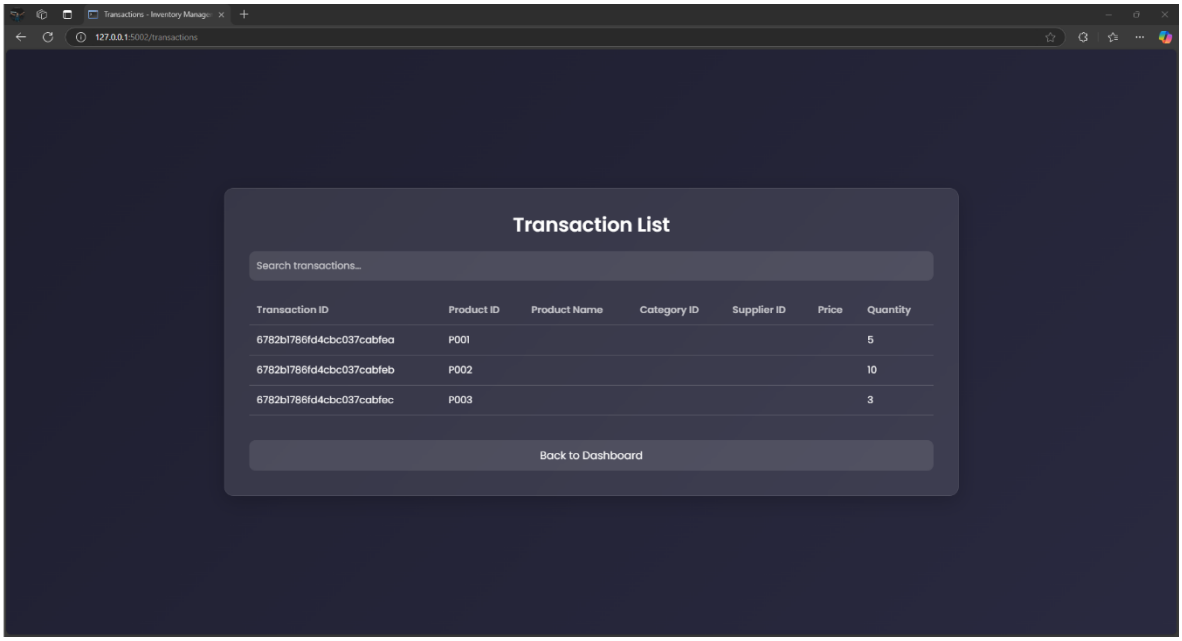
View Products:



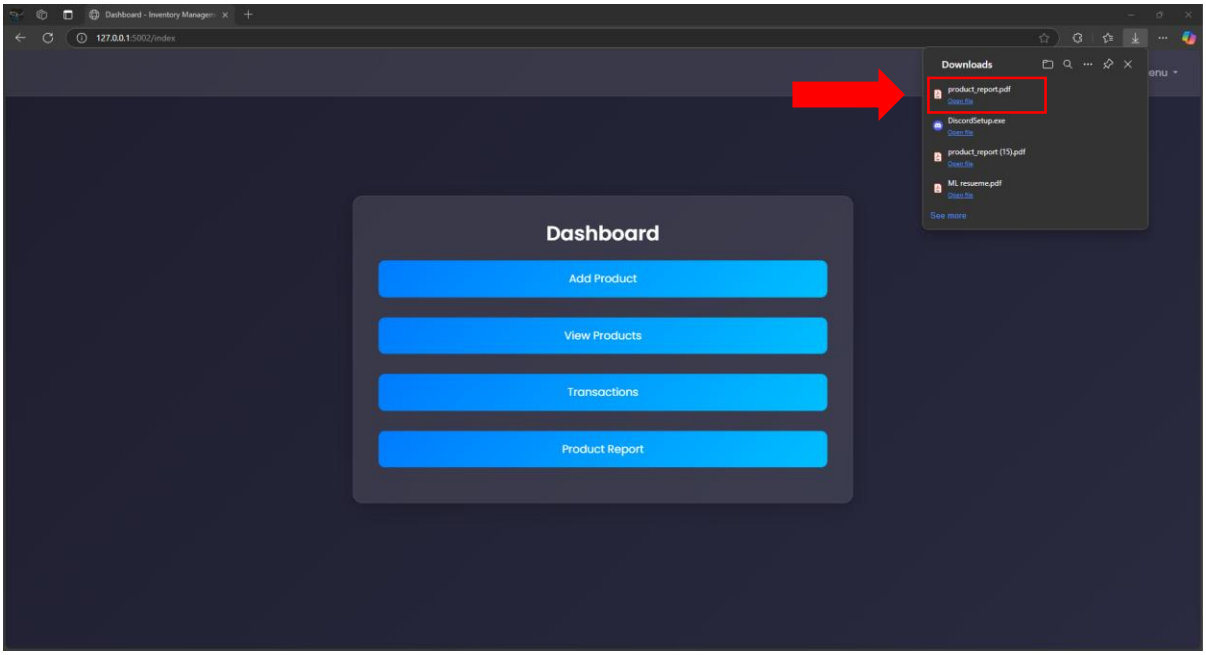
Add Product:

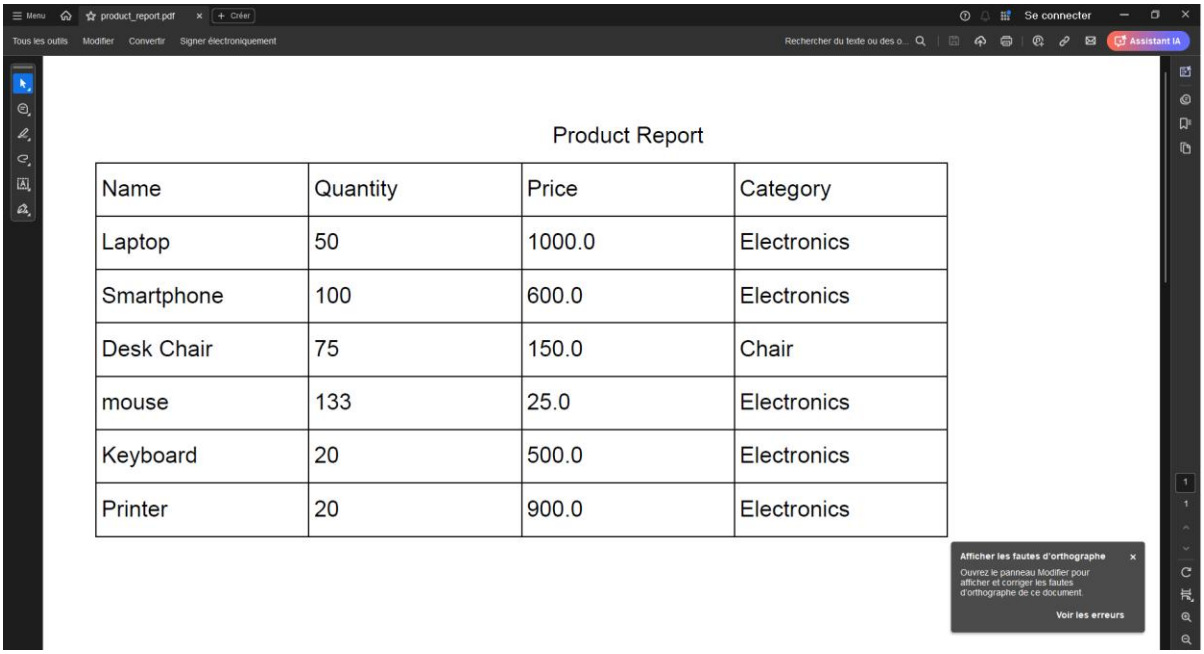


Transactions:



Product Report:





Product Report

Name	Quantity	Price	Category
Laptop	50	1000.0	Electronics
Smartphone	100	600.0	Electronics
Desk Chair	75	150.0	Chair
mouse	133	25.0	Electronics
Keyboard	20	500.0	Electronics
Printer	20	900.0	Electronics

10. Installation and Setup

To run the project locally, follow these steps:

Install Dependencies:

pip install Flask pymongo fpdf

Set Up MongoDB:

Install MongoDB and start the MongoDB server.

Create a database named inventory_db.

Run the Application:

python app.py

Access the Application:

Open your browser and navigate to http://localhost:5002.

11. Team Members

Achraf GEBLI : Project Developer 1

Aymen JABBAR : Project Developer 2

12. Acknowledgments

Flask Documentation: For guidance on building the backend.

Bootstrap Documentation: For designing the frontend.

MongoDB Documentation: For database setup and queries.