

# PROJECT REPORT

# INVENTORY MANAGEMENT SYSTEM

Submitted by:

Parth Shende

25BCE10024

Course Code: CSE1021

SLOT: A14+D11+D12

FACULTY: SANDEEP MONGA SIR

# INTRODUCTION

Managing a shop is hard work. Keeping track of every single item, updating stock after every sale, and calculating bills manually takes a lot of time. Many small shopkeepers still use pen and paper, which leads to mistakes and lost records.

I built this **Inventory Management System** to solve that. It is a simple computer program that moves all that manual work into a clean, digital dashboard. It helps shop owners run their business smoothly without needing expensive software or an internet connection.

## PROBLEM STATEMENT

The main problem is that manual inventory tracking is slow and risky.

1. **Human Error:** It is easy to make math mistakes when calculating bills in a rush.
2. **Stock Issues:** Without a live count, shop owners often don't realize an item is out of stock until a customer asks for it.
3. **Data Loss:** Paper records can get lost, damaged, or just become hard to read over time.

This project aims to fix these issues by providing a digital, automated way to handle stock and sales.

## FUNCTIONAL REQUIREMENTS

To handle the daily needs of a shop, the system performs these main functions:

- **Secure Login:** Users must enter a username and password to access the data.

- **Product Management:** The user can add new items (with Price and Stock count) and delete old ones.
- **Billing:** The user can select an item and quantity. The system calculates the total price automatically.
- **Stock Updates:** As soon as an item is sold, the system automatically subtracts it from the main inventory.
- **Sales History:** The system saves a permanent log of every sale with the date and time.

## NON-FUNCTIONAL REQUIREMENTS

- **Usability:** The interface is simple, with clear buttons and error messages.
- **Performance:** The application loads instantly and processes sales in milliseconds.
- **Reliability:** Data is stored in an ACID-compliant database (SQLite) to prevent corruption.
- **Portability:** The compiled .exe file runs on any Windows PC without installation.

## SYSTEM ARCHITECTURE

The project follows Monolithic **Desktop Architecture**:

- **Presentation Layer:** Built using Python's **Tkinter** library (Windows, Buttons, Tables).
- **Logic Layer:** Python functions handle the calculations (e.g.,  $\text{total} = \text{price} * \text{quantity}$ ) and validation.

- **Data Layer:** SQLite is used for storage. It is a serverless database engine that stores data in a single file (inventory\_system.db).

## DESIGN DECISIONS & RATIONALE

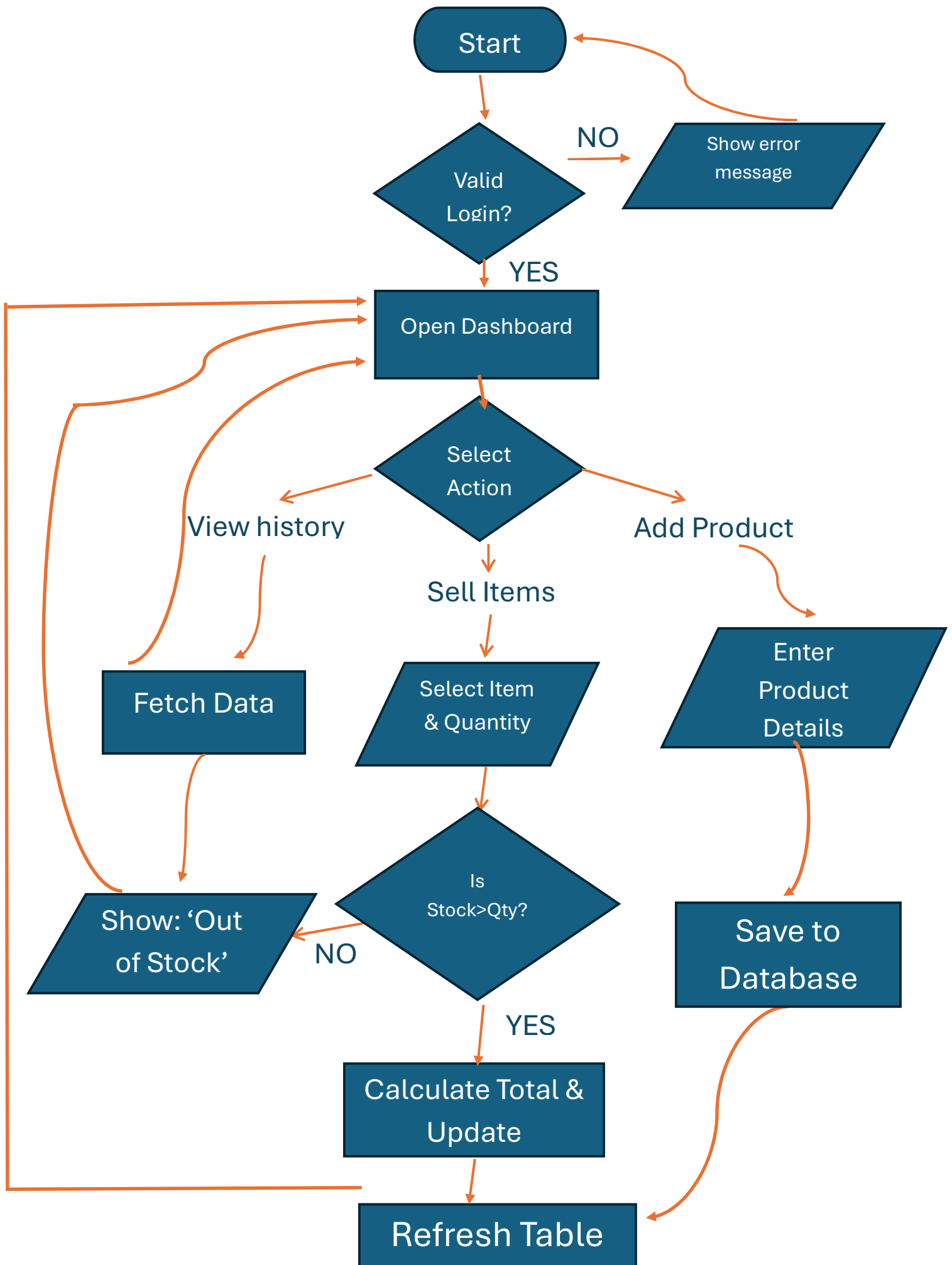
- **Python & Tkinter:** I chose this combination because Tkinter is part of the standard Python library. It is lightweight and creates a native Windows look that is easy for shopkeepers to navigate.
- **SQLite (Serverless):** I chose SQLite over MySQL because it does not require a background server installation. The entire database is a single file, making the project fully portable (it can run directly from a USB drive).
- **PyInstaller:** I used this to compile the script into an .exe file. This ensures the end-user can run the app by double-clicking, without needing to install Python or libraries.
- **Offline Architecture:** I built this as a desktop app rather than a website so that it works 100% offline, which is crucial for shops with unstable internet connections.

## IMPLEMENTATION DETAILS

I organized the code into Classes to keep it clean:

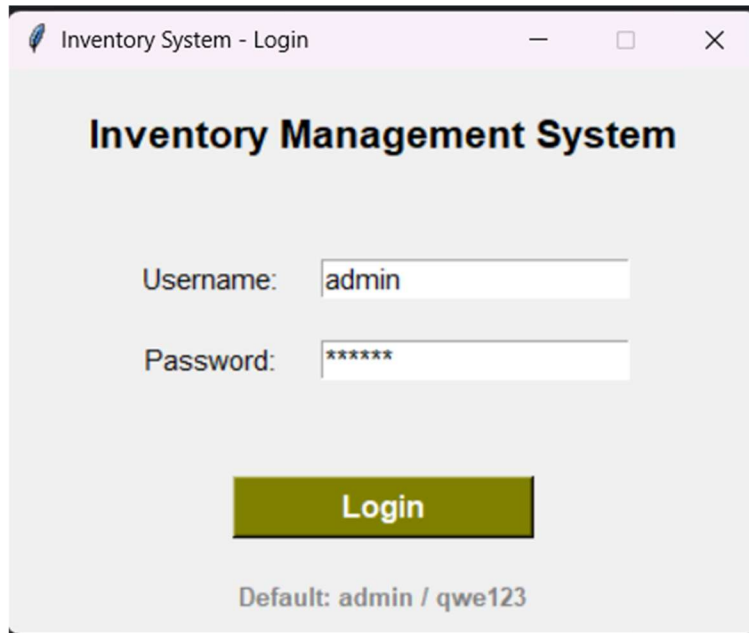
- **Database Class:** This handle connects to the file and creates tables if they are missing.
- **Login Window Class:** This checks if the username/password matches what is in the database.
- **Dashboard Window Class:** This is the main screen. It uses a "Tree view" (a table widget) to show the products.

## DESIGN DIAGRAMS



# SCREENSHOTS/RESULTS

## 1)Login page



The screenshot shows a web browser window titled "Inventory System - Login". The page has a light gray background. At the top, the title "Inventory Management System" is displayed in a bold, black font. Below the title, there are two input fields: "Username:" with the value "admin" and "Password:" with the value "\*\*\*\*\*". A green "Login" button is positioned below the password field. At the bottom of the page, the text "Default: admin / qwe123" is visible.

Inventory System - Login

### Inventory Management System

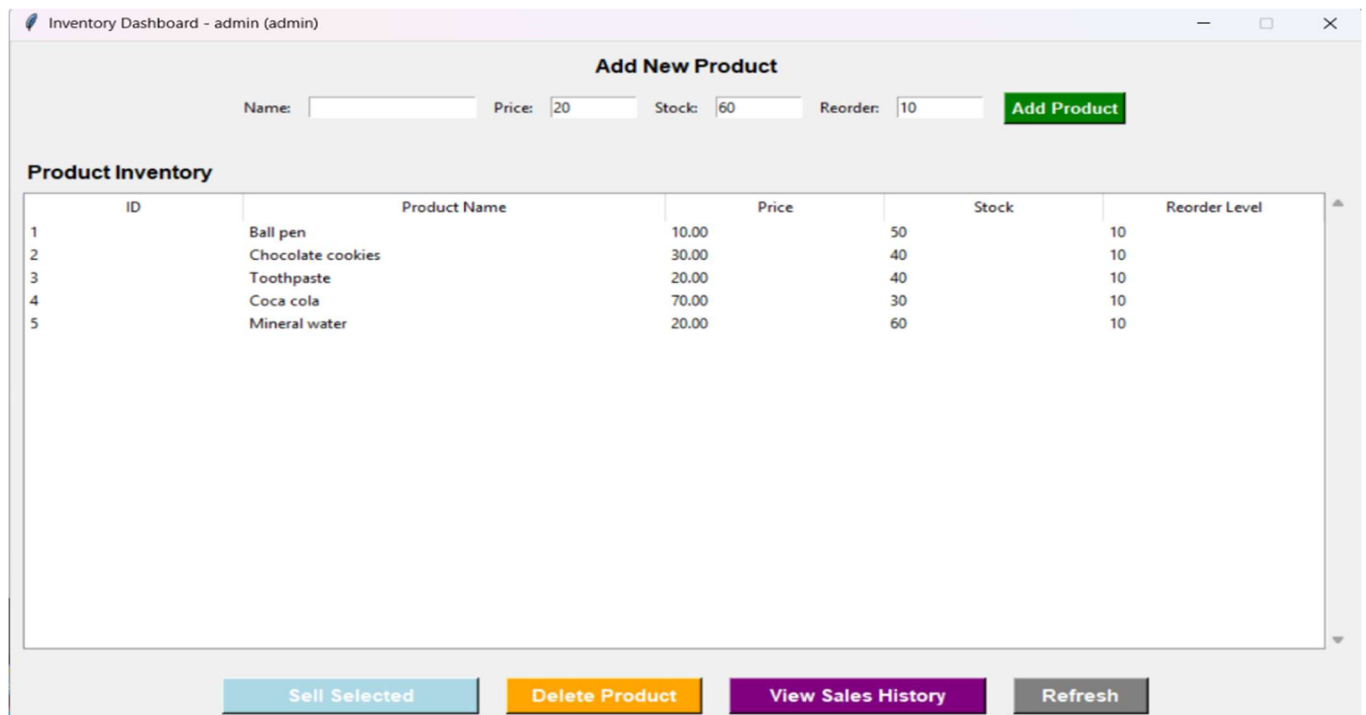
Username: admin

Password: \*\*\*\*\*

Login

Default: admin / qwe123

## 2)Inventory Dashboard



The screenshot shows a web browser window titled "Inventory Dashboard - admin (admin)". The dashboard has a light gray background. At the top, there is a section titled "Add New Product" with four input fields: "Name:", "Price:" (value 20), "Stock:" (value 60), and "Reorder:" (value 10). A green "Add Product" button is to the right of these fields. Below this section is a table titled "Product Inventory". The table has five columns: "ID", "Product Name", "Price", "Stock", and "Reorder Level". The table contains five rows of data. At the bottom of the dashboard, there are four buttons: "Sell Selected" (blue), "Delete Product" (orange), "View Sales History" (purple), and "Refresh" (gray).

Inventory Dashboard - admin (admin)

### Add New Product

Name: Price: 20 Stock: 60 Reorder: 10 Add Product

### Product Inventory

| ID | Product Name      | Price | Stock | Reorder Level |
|----|-------------------|-------|-------|---------------|
| 1  | Ball pen          | 10.00 | 50    | 10            |
| 2  | Chocolate cookies | 30.00 | 40    | 10            |
| 3  | Toothpaste        | 20.00 | 40    | 10            |
| 4  | Coca cola         | 70.00 | 30    | 10            |
| 5  | Mineral water     | 20.00 | 60    | 10            |

Sell Selected Delete Product View Sales History Refresh

### 3)Pop-up box appears after clicking on “Sell Selected”

Inventory Dashboard - admin (admin)

**Add New Product**

Name:  Price:  Stock:  Reorder:  Add Product

**Product Inventory**

| ID | Product Name      | Price | Stock | Reorder Level |
|----|-------------------|-------|-------|---------------|
| 1  | Ball pen          | 10.00 | 50    | 10            |
| 2  | Chocolate cookies | 30.00 | 40    | 10            |
| 3  | Toothpaste        | 20.00 | 40    | 10            |
| 4  | Coca cola         | 70.00 | 30    | 10            |
| 5  | Mineral water     | 20.00 | 60    | 10            |

Qty

Sell Ball pen

Quantity:

Confirm

Sell Selected Delete Product View Sales History Refresh

### 4)Pop-up box for Sales history

Inventory Dashboard - admin (admin)

**Add New Product**

Name:  Price:  Stock:  Reorder:  Add Product

**Product Inventory**

| ID | Product Name      | Price | Stock | Reorder Level |
|----|-------------------|-------|-------|---------------|
| 1  | Ball pen          | 10.00 | 50    | 10            |
| 2  | Chocolate cookies | 30.00 | 40    | 10            |
| 3  | Toothpaste        | 20.00 | 40    | 10            |
| 4  | Coca cola         | 70.00 | 30    | 10            |
| 5  | Mineral water     | 20.00 | 60    | 10            |

Sales History

| ID | NAME          | QTY | TOTAL | DATE                |
|----|---------------|-----|-------|---------------------|
| 3  | Mineral water | 12  | 240.0 | 2025-11-22 23:45:35 |
| 2  | Coca cola     | 10  | 700.0 | 2025-11-22 23:45:22 |
| 1  | Ball pen      | 5   | 50.0  | 2025-11-22 23:45:06 |

Sell Selected Delete Product View Sales History Refresh

## TESTING APPROACH

I tested the app manually to make sure it doesn't break:

1. **Wrong Password:** I tried logging in with a random password, and it showed "Invalid Login."
2. **Bad Input:** I tried typing "ABC" in the Price box. The system caught it and said "Numbers only."
3. **Selling Too Much:** I had 5 pens in stock and tried to sell 10. The system said "Invalid Quantity."
4. **Zero Stock:** I tried to sell an item with 0 stock, and it blocked the sale.

## CHALLENGES FACED

- **The Time zone Problem:** At first, the sales history was showing the wrong time (it was showing London time). I realized SQLite does that by default. I fixed it by writing code to get my computer's local time using `datetime.now()` before saving the sale.
- **Resizing:** When I made the window full screen, the table stayed small. I had to learn how to use `pack(fill=BOTH, expand=True)` to make it stretch.



## LEARNING AND KEY TAKEAWAYS

- I learned how to connect Python code to a real database using SQL commands like INSERT and SELECT.
- I learned how to make a standalone .exe file. It was cool to see my code running on a computer that didn't even have Python installed.
- I learned that handling user input errors (like typing text instead of numbers) is very important for a good app.

## FUTURE ENHANCEMENTS

If I work on this more, I would add:

- **Barcode Support:** So you can scan items instead of clicking them.
- **Print Receipt:** A button to print a bill for the customer.
- **Low Stock Warning:** A popup that tells you when items are running out.

# REFERENCES

## Python Official Documentation

- *Link:* <https://docs.python.org/3/library/sqlite3.html>
- *Use:* I used this to check the correct syntax for connecting to the database and executing SQL commands within Python.

## GeeksforGeeks

- *Link:* <https://www.geeksforgeeks.org/python-gui-tkinter/>
- *Use:* I referred to their tutorials to understand how to arrange buttons and create the table view (Treeview) in the dashboard.

## StackOverflow

- *Link:* <https://stackoverflow.com/>
- *Use:* I used this to find the solution for the timezone issue, specifically how to get the local system time instead of UTC.