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., total = price * 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

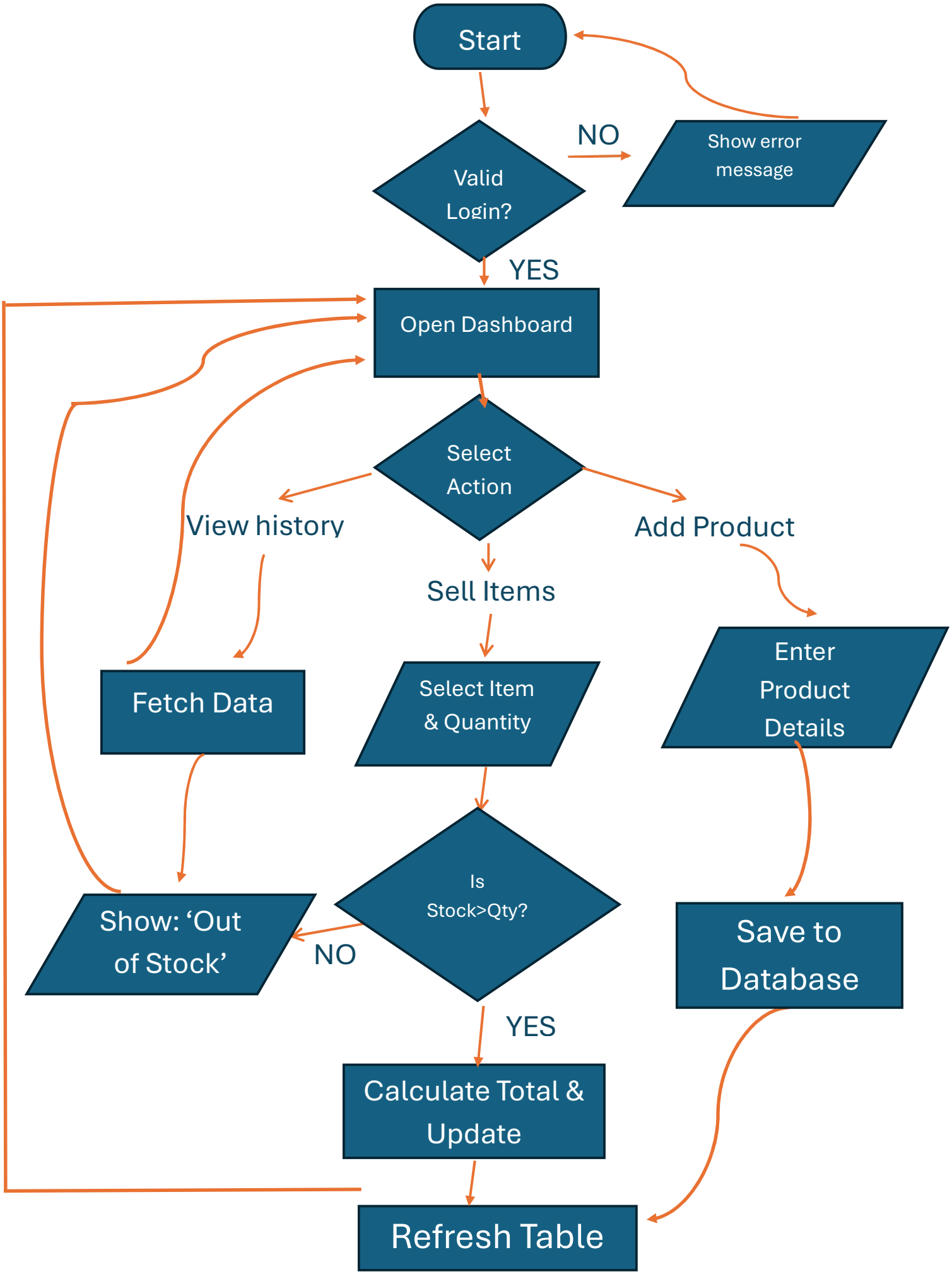
- **Python & Tkinter:** I chose this combination because Tkinter is part of the standard Python library. It is lightweight and creates a native Windows look that is easy for shopkeepers to navigate.
- **SQLite (**Serverless**):** I chose SQLite over MySQL because it does not require a background server 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:

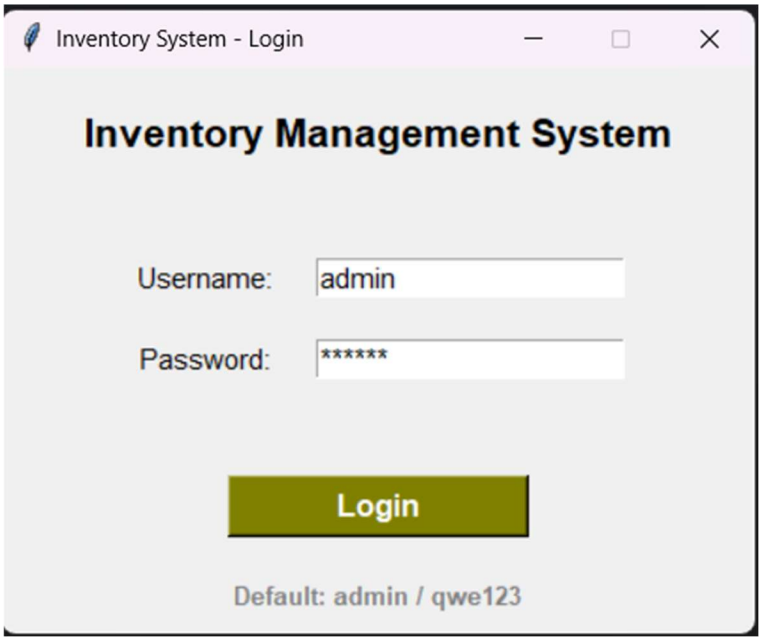
- **Database Class:** This handle connects to the file and creates tables if they are missing.
- **Login Window Class:** This checks if the username/password matches what is in the database.
- **Dashboard Window Class:** This is the main screen. It uses a "Tree view" (a table widget) to show the products.

DESIGN DIAGRAMS

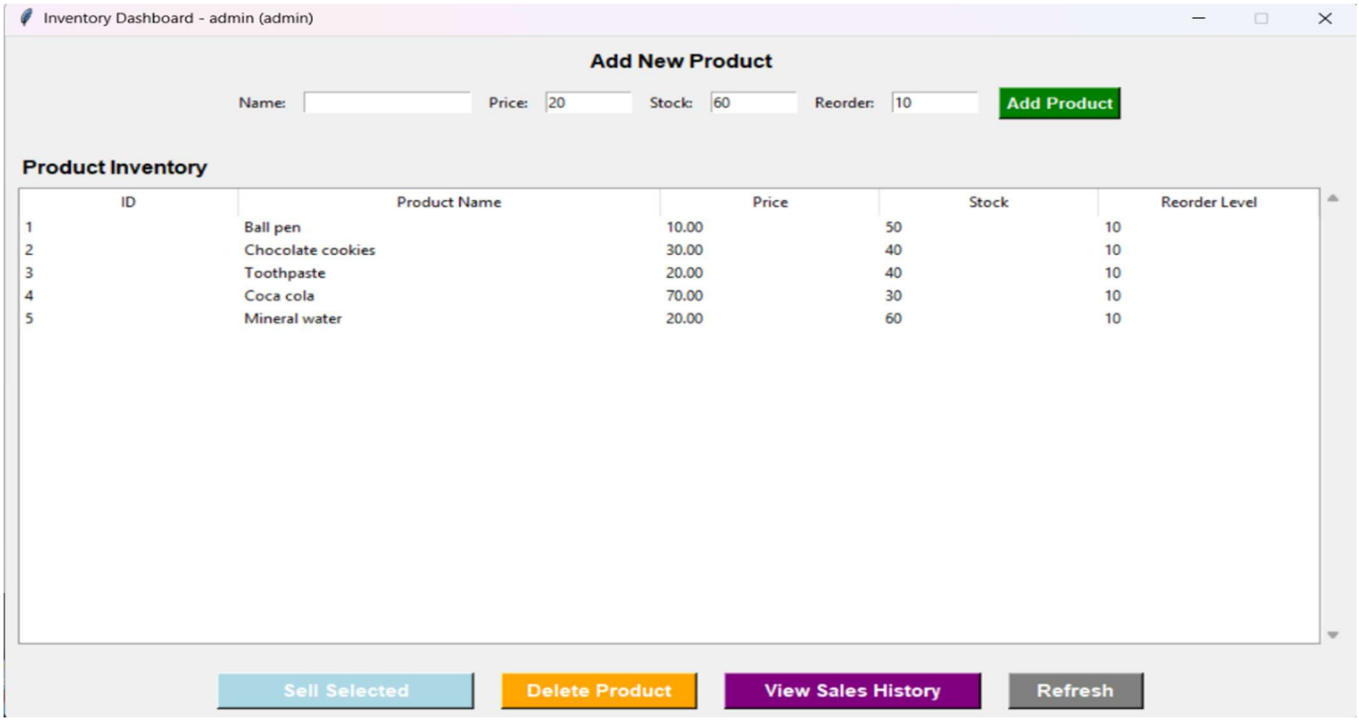


SCREENSHOTS/RESULTS

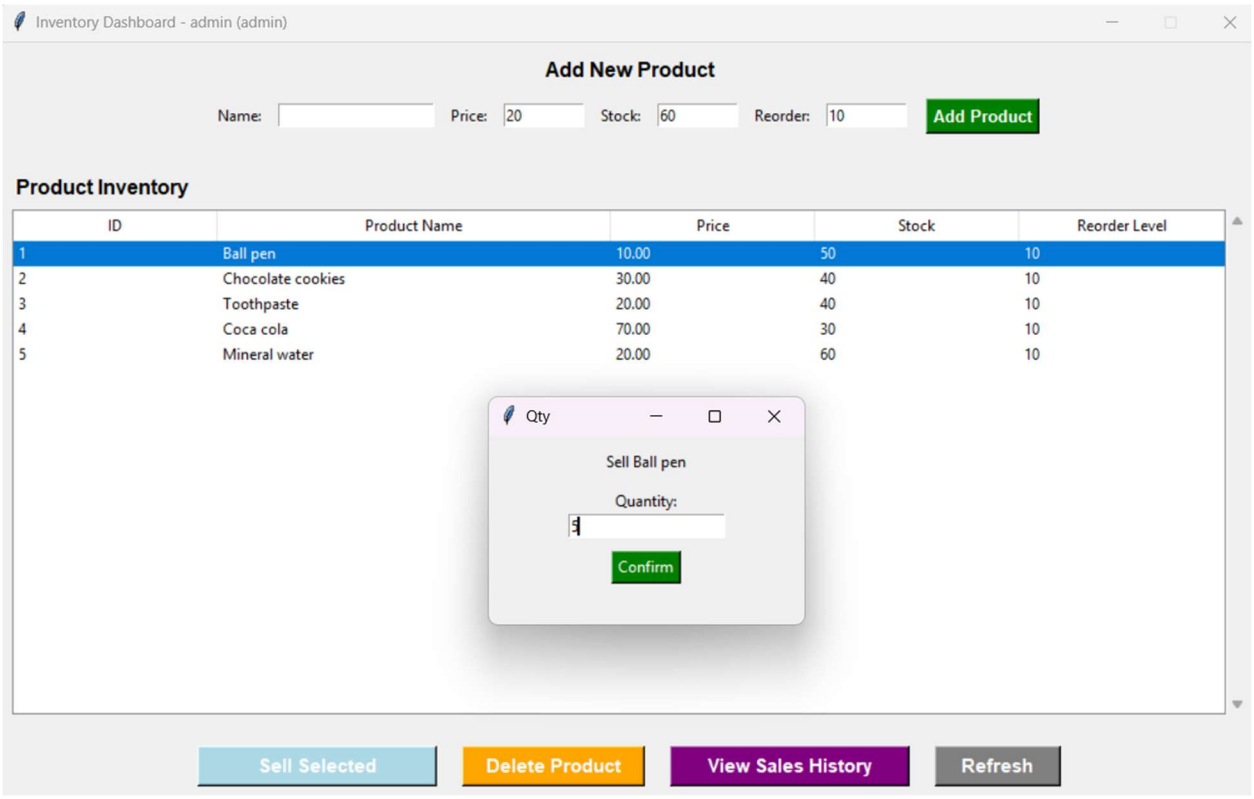
1)Login page



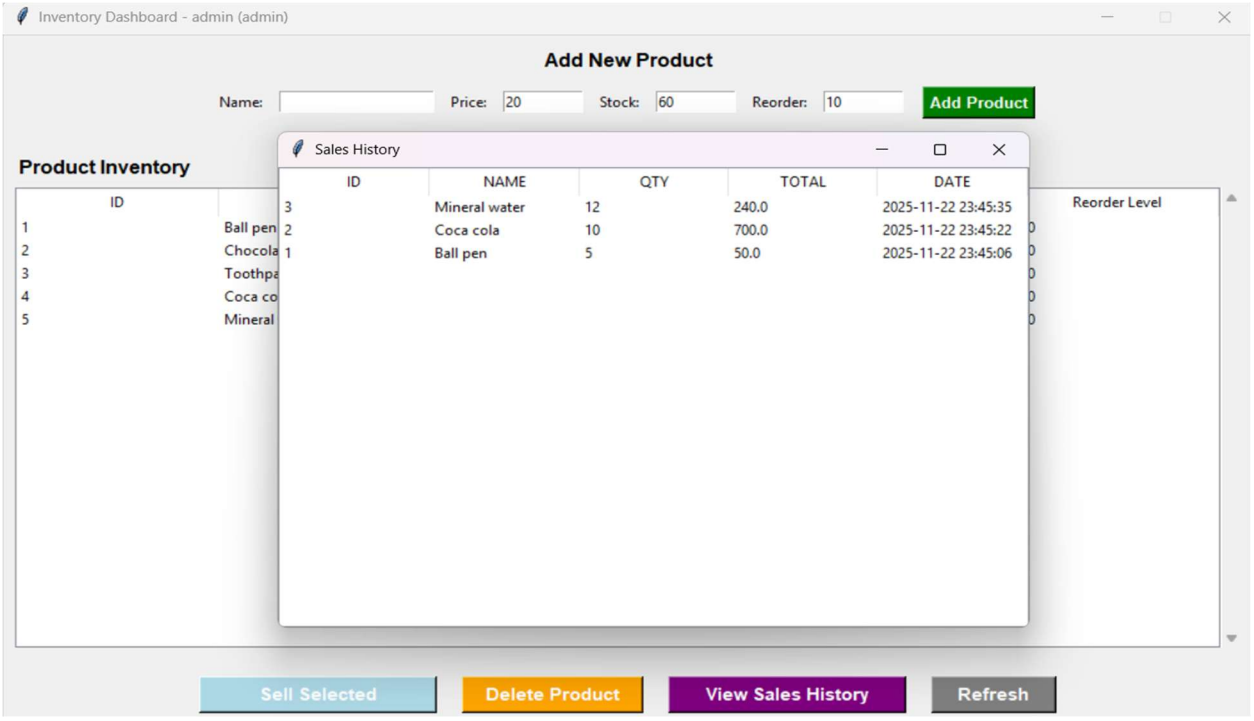
2)Inventory Dashboard



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



4)Pop-up box for Sales history



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.