Inventory Control System

Developed by: Youcef Yousfi
Module: Computer Science
Project Task 3: Inventory Management System
Due Date: 8 May 2025

1. Problem Definition and Analysis

In retail and small businesses, tracking inventory is crucial to avoid stockouts or overstocking. Many small businesses still manage stock manually, which can lead to errors, loss of efficiency, and customer dissatisfaction. The objective of this project was to create a simple, console-based Inventory Management System using Python and SQLite that helps users manage products effectively by adding, updating, and querying stock items. I chose this project because it aligns with the core skills I developed during the course working with databases, using Python for file and data manipulation, and applying logical structures such as loops and conditional statements. Python's simplicity and SQLite's lightweight structure made them ideal for a beginner-friendly, yet functional solution. The system is designed for small shop owners or warehouse managers who need a basic inventory tracking tool. The stages in this project included database creation, function development for CRUD operations, input validation, and testing. I referred to SQLite documentation and Python standard library resources for best practices in handling data securely and efficiently.

2. Documented Design

The system is structured in a modular format where each core function (adding, updating, searching) is defined separately and interacts with a local SQLite database. The system operates via a command-line interface (CLI), which guides the user through various options. System Design: START -> Main Menu -> [1] Add Product -> Save to DB -> [2] Search Product -> Query DB -> [3] Update Stock -> Modify DB -> [4] Exit -> END Modules include: 1. Database Connection Module 2. Add Item Function 3. Search Function 4. Stock Update Module 5. Input Validation Module Data stored includes ID (TEXT), Item Name (TEXT), Type (TEXT), Cost (REAL), and Amount (INTEGER). The system uses structured functions and validation to ensure accuracy and reliability.

3. Testing and Evaluation

Testing was performed using a range of valid and invalid inputs: - Add Valid Item: All fields correct -> Success - Duplicate ID: Prevented -> Error handled - Invalid Cost (text): Rejected - Negative Quantity: Rejected - Search by Category: Correct item shown Evaluation: - Database creation: Completed - Add/Search/Update: Working as intended - Input validation: Successfully implemented Future improvements include adding a graphical interface using Tkinter, export features to CSV, and login system enhancements.

Test Case	Input	Expected Output	Result
Add valid item	All correct fields	Success message	Passed
Duplicate ID	Existing ID	Error or overwrite warning	Handled
Non-numeric cost	'five' instead of number	Input rejected	Passed
Negative stock quantity	-10	Input rejected	Passed
Search by category	'Stationery'	Matching item(s) displayed	Passed

Improvements and Feedback

To further improve the system:
- Add a GUI using Tkinter
- Implement export to CSV for reports
- Integrate a login system for admin users

Throughout development, I sought feedback from peers and my tutor, which helped refine the code structure and interface clarity. Overall, the system met its intended objectives and can be expanded for real-world use with minimal upgrades.