Name: - Aditya Ganguly

Reg.No.:- 25BCE10661

Branch: - CSE Core

Faculty:-Dr.G.Ganesan

VIT Bhopal

Project Report

Title:- Medical Inventory Tracker

Introduction:-

This project, the **Medicine Inventory Tracker**, is a console-based application developed using the **Python 3** programming language. It is designed to simulate a fundamental stock management system for a small pharmacy, clinic, or medical supply unit. The core objective is to demonstrate the mastery of introductory Computer Science principles, including modular programming, fundamental data structures (Dictionaries), and persistent data management through file input/output (I/O). A key feature of the application is the robust implementation of **explicit input validation** using **if/else** control flow, ensuring data integrity without relying on advanced exception handling mechanisms. The application allows users to load, view, add, update, and save medicine stock information, providing essential low-stock alerts.

Problem Statement

Manual inventory tracking in healthcare environments (using spreadsheets or paper ledgers) is highly inefficient, prone to human error, and lacks the ability to provide real-time status updates. Errors in tracking stock can lead to critical shortages, impacting patient care and operational costs.

Proposed Solution

The **Medicine Inventory Tracker** provides a simple, menu-driven interface accessible via the console. This solution automates the processes of recording stock transactions and utilizes a text file for data persistence. This approach prioritizes **simplicity, reliability, and clear demonstration of core programming logic** over complex graphical interfaces or database management.

Data Structure Design

The inventory data is stored within Python as a nested structure, leveraging the efficiency of the **Dictionary** data type:

- **Outer Structure (Inventory):** A master dictionary where the **Medicine Name** (a string) acts as the unique key.
- **Inner Structure (Details):** The value associated with the key is a dictionary containing item details:
    - 'quantity': An integer representing the current stock count.
    - 'expiry': A string storing the expiry date (MM/YYYY).

Goal of this program:-

The main goal of the program is to provide an inventory management system to the Pharmacists and small clinics to keep track of their medicinal stocks. Using simple menu- driven approach and inventory accessing methods, this becomes an easy to understand program and thus is of practical real world use too. This system of management reduces the errors made by humans and keeps tack of any low stocks to fulfil them immediately.

Explanation of the Program:-

This program begins by importing the OS module. The os module in Python provides functions to interact with the operating system in a portable way. Here we use this module to interact with the directories and files of the operating system to store the value of medicines and access them as and when required by us. We then create several functions that contain the working of how the directory is accessed to get the values or to store and edit the values of medicines in it. The program is mainly built with the help of loops, if- else constructs and basic input output statements to get the values. The program also shows the medicines which are low in stocks and need to be replenished back to the store before stock out.

The application is entirely menu-driven and implements the following essential inventory features:

Data Persistence: Utilizes a standard text file (inventory.txt) to load and save all inventory data, ensuring that stock information is retained between program executions.

Inventory Management: Allows users to add new medicines (with initial stock and expiry date) and update existing stock (via addition or dispensing/removal).

Input Validation: Employs explicit if/else logic to ensure that all inputs, particularly stock quantities, are valid non-negative whole numbers, preventing logical errors.

Low Stock Alert: Automatically identifies and lists medicines whose current quantity is below a predefined low stock indicator (e.g., 10 units), aiding in timely reordering.

Reporting: Displays the entire inventory in a clear, formatted tabular view directly in the console.

Results

This application successfully manages the medicine inventory, providing a functional, persistent, and reliable text interface that accurately reflects real-world stock management scenarios.

Conclusion:-

The **Medicine Inventory Tracker** project is a successful demonstration of core programming proficiency in Python. It effectively utilizes dictionaries for data organization, functions for modular design, and robust **if/else** conditional logic for comprehensive input and data validation. The successful implementation of data persistence confirms the student's understanding of File I/O and fundamental application design.

What I learnt in this project:-

While making this project on inventory management, I learnt the use of os module and how it interacts with the python program to store the values so that it can be accessed anytime when needed. I gained the knowledge of how to store the files in the directories of the system and can be used as a information management system also. I also learnt a broader use of if-else and looping constructs. While working out the program, I also got to know about try-except type of constructs and got to know it is similar to if-else except that it requires an extra condition which is required to detect the errors quickly without much of a long logic statements as required in if-else constructs.

Future Scope

To evolve this project beyond the introductory level, the following enhancements could be implemented:

1. **Database Migration:** Replace text file storage with a relational database (e.g., SQLite) for better query performance and structured data management.
2. **Graphical User Interface (GUI):** Implement a desktop interface using libraries such as Tkinter or PyQt to improve user experience.
3. **Advanced Alerting:** Integrate functions to check and alert the user about medicines that are close to their expiry date (e.g., less than 90 days remaining).
4. **Reporting:** Add the ability to generate printable reports (e.g., PDF or CSV) of all low-stock items or recent transactions.