PROJECT REPORT

Project Title: Pharmacy Billing & Management System

Course: Introduction to Problem Solving & Programming (IPSP)

Student Name: DEVANSH KAUSHIK

Registration No: 25BSA10016

1. Introduction

The Pharmacy Billing System is a console-based application developed in Python. It is designed to automate the manual processes involved in a retail pharmacy, such as maintaining inventory records and generating bills for customers. By utilizing file handling techniques, the system ensures that stock data is persistent and secure, eliminating the need for manual ledger keeping.

2. Problem Statement

Traditional manual billing and stock keeping in small pharmacies are prone to human error, calculation mistakes, and difficulty in tracking available stock. There is a need for a lightweight, efficient digital solution that can handle basic transactions and inventory management without requiring complex database software.

3. Objectives

- To create a user-friendly interface for billing and stock management.
- To implement a persistent storage system using CSV files.
- To apply Python programming concepts like lists, loops, and modular functions to solve real-world problems.
- To ensure accurate calculation of bills including tax/discounts.

4. Methodology & Algorithm

The project follows a modular design approach where specific tasks are handled by dedicated functions.

General Algorithm:

1. **Start** the application.
2. **Load** existing inventory data from `med.csv` into a Python list.
3. **Display** the Main Menu (Billing, Add Stock, Delete Stock, View Stock).
4. **Accept** user input.

- If **Billing**: Search for medicine -> Check availability -> Calculate cost -> Update Stock -> Print Bill.
- If **Stock Management**: Accept new details -> Append/Update List -> Write to CSV.

5. **Loop** the menu until the user chooses to exit.
6. **Stop**.

5. Technical Implementation (Concepts Used)

This project demonstrates the practical application of several core Python concepts:

A. List & Nested Lists

- **Definition:** A list is a mutable, ordered collection of elements. A nested list is a list containing other lists.
- **Application:** I used a nested list structure (e.g., `[['Paracetamol', '100', '5.0'], ['Dolo', '50', '2.0']]`) to hold the inventory in the program's memory (RAM). This allowed for easy iteration and retrieval of medicine details using index positions (Index 0 for Name, 1 for Qty, 2 for Price).

B. File Handling (CSV)

- **Definition:** File handling allows the program to read from and write to permanent storage files on the disk.
- **Application:** The `csv` module was used to read the database (`med.csv`) at the start of the program and write updated data back to it. This ensures that stock changes (like selling 10 tablets) are saved permanently even after closing the code.

C. Functions (Modularity)

- **Definition:** Functions are reusable blocks of code designed to perform a specific task.
- **Application:** The code is divided into logical blocks such as `billing()`, `addStock()`, and `deleteStock()`. This makes the code readable, easier to debug, and strictly follows the "Divide and Conquer" problem-solving strategy.

D. Control Structures (Loops & Conditionals)

- **Definition:** Loops (`while`, `for`) repeat a block of code, while conditionals (`if-elif-else`) execute code based on specific conditions.
- **Application:**
  - `While True`: Used to keep the main menu running indefinitely until the user exits.
  - `For loop`: Used to iterate through the inventory list to find a specific medicine match.
  - `If-Else`: Used to validate user inputs (e.g., preventing a user from buying more stock than is available).

6. Testing & Validation

The system was tested with various test cases:

- **Case 1 (Valid Transaction):** User purchases an item within stock limits. -> *Result: Success, Bill generated, Stock reduced.*
- **Case 2 (Stock Overflow):** User tries to buy more quantity than available. -> *Result: System displays "Not enough stock" error.*
- **Case 3 (Data Persistence):** Program closed and reopened. -> *Result: Previous changes to inventory were retained.*

7. Conclusion

The Pharmacy Billing System successfully achieves its objective of automating daily pharmacy operations. It demonstrates the power of Python in handling data and logic efficiently. The project highlights the importance of data structures (Lists) and persistent storage (CSV) in building useful software applications.