**Project: Medicine Receipt System for Medical Stores**

**Description**  
This system allows users (pharmacy staff) to enter and manage medicine sales transactions. It includes the ability to create and print customer receipts, manage inventory, track sales, and maintain a database of medicines, including details like expiry dates, price, and stock quantity.

**Features and Functionalities**

1. **Medicine Inventory Management**
   * Add, update, or remove medicines.
   * Track details like medicine name, price, quantity, expiry date, and manufacturer.
   * Alerts for low stock and expired medicines.
2. **Receipt Generation and Sales Tracking**
   * Generate a receipt for each sale transaction, including itemized medicine list, quantities, total cost, and payment status.
   * Store receipts in a database for record-keeping and future reference.
3. **Search and Filtering Options**
   * Search by medicine name, expiry date, or price range.
   * Filter out expired medicines or those nearing expiry.
4. **Database Storage and Management**
   * Store data on medicines, sales transactions, and inventory details in a database for persistence.
   * Perform CRUD (Create, Read, Update, Delete) operations on the database.

**Technical Requirements**

* **Programming Language**: C++
* **Database**: SQLite or MySQL (to store medicines, sales transactions, and customer details)
* **Data Structures Used**:
  + **Linked List**: To handle medicine inventory with easy addition/removal of items.
  + **Hash Map**: For fast lookup of medicine details by name.
  + **Queue**: To handle receipts in order of transactions.
  + **Stack**: To manage an undo feature for recent transactions or inventory updates.
* **Algorithms**:
  + **Sorting**: For sorting medicines by expiry date, price, or alphabetical order.
  + **Searching**: For searching medicines by name or ID.
  + **CRUD Operations**: On the database for medicines and receipts.
  + **Filtering**: Expired medicines or near-expiry medicines.

**Database Schema (Example)**

1. **Medicines Table**
   * medicine\_id (Primary Key)
   * name
   * price
   * quantity
   * expiry\_date
   * manufacturer
2. **Receipts Table**
   * receipt\_id
   * date
   * total\_amount
   * customer\_name
   * payment\_status
3. **Sales Table**
   * sale\_id
   * receipt\_id
   * medicine\_id
   * quantity

**Example Workflow**

1. **Add Medicine**: Pharmacy staff can add new medicines with details like name, price, quantity, and expiry date. The system checks for existing stock and alerts if the medicine already exists.
2. **Sell Medicine and Generate Receipt**:
   * Staff selects medicines for sale, specifies quantities, and the system checks inventory availability.
   * A receipt is generated with an itemized list, total amount, and any applicable discounts.
   * Sale details are stored in the database.
3. **Check Inventory**: Staff can check stock levels, search for medicines by name, and get a list of near-expiry or out-of-stock medicines.
4. **View Receipts and Sales History**: Retrieve historical data on sales transactions, and print or view receipts as needed.

**Suggested File Structure**

* **Main.cpp**: Program entry, main menu for navigation.
* **Inventory.cpp/Inventory.h**: Handles inventory functions and data structure operations.
* **Receipt.cpp/Receipt.h**: Manages receipt creation, storage, and lookup.
* **Database.cpp/Database.h**: Handles database connectivity and CRUD operations.

**Bonus Features (Optional)**

* **Automatic Reordering Notification**: Alert staff when inventory for a medicine falls below a certain threshold.
* **Graphical User Interface (GUI)**: Use a simple GUI to display and manage data in a user-friendly manner.
* **Dynamic Pricing Calculation**: Apply discounts or tax calculations to total receipt costs.

**Benefits of this Project**

* **Practicality**: This project closely models real-world medical store operations.
* **Integration of DSA Concepts**: Uses a variety of data structures and algorithms to create an efficient system.
* **Database Practice**: Teaches database design and CRUD operations.
* **User Experience**: Develops skills in creating a user-focused interface and error handling.