Project Proposal

Project Logo

Project Title

HealthMart Pharmacy

Project Idea

HealthMart Pharmacy: the project in C is a pharmacy store that minimizes human effort in the sales of drugs and other medication through efficient and quick stock updating and vending. The customer would be presented with a list of all the drugs currently available at the store along with the description of the same. The customer can choose from the list of drugs and create a shopping cart. Once done selecting the medicines, the customer can checkout once the cost is calculated. The stock of the currently available medicines would be immediately updated in the system to help the next customer. The program would utilize file handling, arrays, loops and various arithmetic operations to make the program as efficient as possible.

Health Mart Pharmacy Management System

Introduction

The HealthMart Pharmacy Management System is a project developed in C programming language aimed at revolutionizing the pharmacy retail experience. In today's fast-paced world, pharmacies face increasing demands to streamline operations and provide efficient services to customers. This project seeks to address these challenges by automating critical aspects of the pharmacy sales process, ultimately enhancing customer satisfaction and operational efficiency.

In addition to simplifying the sales process, the HealthMart Pharmacy Management System also prioritizes customer safety and well-being. By maintaining an accurate and up-to-date inventory of medicines, the system helps pharmacists ensure that essential drugs are always available when needed. Moreover, the system minimizes the risk of errors by automating stock management and medication dispensing. It reduces the likelihood of dispensing incorrect or expired medications, enhancing patient safety and trust in the pharmacy's services.

Furthermore, the project aims to empower pharmacists with valuable insights into customer preferences and medication trends. The system can generate comprehensive reports on sales volumes, popular medications, and inventory turnover rates through data analytics and reporting features. These insights enable pharmacists to make informed decisions regarding stock replenishment, pricing strategies, and marketing initiatives, ultimately driving business growth and competitiveness in the pharmaceutical retail market.

Features

1. **Efficient Stock Management**: The HealthMart Pharmacy Management System revolutionizes stock management by providing pharmacists with a seamless and efficient way to update medicine stocks. Leveraging robust file-handling mechanisms, the system maintains a comprehensive inventory database that accurately reflects the availability of medicines in real-

time. Pharmacists can easily add new medicines, update quantities, and remove expired items, ensuring that customers always have access to their needed medications. By automating stock management processes, the system minimizes human error and streamlines inventory control, ultimately enhancing the pharmacy's operational efficiency and reducing the likelihood of stockouts.

2. **Interactive Drug Selection**: With the HealthMart Pharmacy Management System, customers enjoy a user-friendly and intuitive interface for selecting medicines. The system presents customers with a visually appealing display of all available drugs, complete with detailed descriptions to aid in informed decision-making. Customers can effortlessly browse the list of medications, filter by category or condition, and view essential information such as dosage instructions and potential side effects. This interactive drug selection feature enhances the shopping experience, empowering customers to make educated choices about their healthcare needs.
3. **Shopping Cart Functionality**: The system offers robust shopping cart functionality, allowing customers to assemble their desired medicines quickly. Using efficient array management techniques, the program seamlessly organizes the items customers select, providing a clear overview of the contents of their virtual cart. Customers can add or remove items as needed, adjust quantities, and review their selections before checkout. This shopping cart functionality enhances convenience and flexibility, enabling customers to tailor their purchases to meet their specific medication requirements.
4. **Automatic Cost Calculation**: At checkout, the HealthMart Pharmacy Management System automatically calculates the total cost of the selected medicines, ensuring transparency and accuracy in transactions. By employing various arithmetic operations, the system accurately computes the subtotal, taxes, and any applicable discounts or promotions, providing customers with a clear breakdown of their expenses. This automatic cost calculation feature eliminates the

need for manual calculations, reducing errors and enhancing customer confidence in the accuracy of their transactions.

5. **Real-time Stock Updates**: After a successful transaction, the system immediately updates the stock of purchased medicines in real-time. This feature ensures that the inventory remains current and reflects the latest sales data. By automatically adjusting stock levels based on customer purchases, the system minimizes the risk of stockouts and optimizes inventory management processes. Pharmacists can rely on the system to maintain accurate stock levels, enabling them to efficiently manage inventory and meet customer demand without disruption.

System Architecture

The HealthMart Pharmacy Management System is architectured with modularity in mind, aiming to simplify maintenance and support scalability. Here are the critical components of the system:

1. **Main Program**: As the system's nucleus, the main program functions as the central controller. It orchestrates the flow of the program, presenting options to pharmacists and customers alike. Acting as the primary interface ensures smooth interaction between users and the system's functionalities, thus facilitating efficient pharmacy operations management.
2. **Structs for Medications**: The system employs structs to represent medication data in line with the C code provided. Each medication is encapsulated within a struct, containing attributes such as medication ID, name, description, price, and availability status. This structured approach ensures the organized storage and retrieval of medication information, enhancing the system's efficiency in managing pharmaceutical inventory.
3. **Arrays**: While not extensively utilized in the provided code, arrays support the shopping cart functionality within the HealthMart Pharmacy Management System. By utilizing arrays, the system can effectively organize and track the items customers select, offering them the flexibility to add, remove, and modify their chosen medications effortlessly.

4. **Loops and Arithmetic Operations**: Loops and arithmetic operations form the backbone of various functionalities crucial for the system's smooth operation. Loops enable iterative processes such as updating medication stock levels, iterating through lists of medications, and processing customer transactions. On the other hand, arithmetic operations facilitate tasks such as calculating costs, applying discounts, and computing tax amounts. Loops and arithmetic operations enhance the system's efficiency and accuracy, ensuring seamless execution of critical functions throughout the pharmacy management process.

This architecture ensures that the HealthMart Pharmacy Management System is robust, reliable, flexible, and scalable to meet the evolving needs of pharmacies and their customers.

Usage Instructions

1. **Updating Stock**: Pharmacists can update the stock of medicines seamlessly through the system's intuitive interface. By accessing the designated menu option, pharmacists can enter essential information such as the medicine name, quantity, and price. This streamlined process enables efficient medication inventory management, ensuring stock levels are accurately maintained to meet customer demand.
2. **Customer Interaction**: The system provides an interactive and user-friendly platform for customers to explore the extensive list of available medicines. Customers can easily navigate the medication catalog, accessing detailed information about each product. With the ability to select their desired items and add them to their shopping cart with a simple click, customers experience a seamless browsing experience tailored to their needs and preferences.
3. **Checkout Process**: Upon finalizing their medication selections, customers can quickly proceed to the checkout process. The system calculates the total cost of the selected medicines, considering factors such as quantity and price. Customers are presented with a clear and transparent summary

of their purchases, facilitating informed decision-making. With a straightforward checkout interface, customers can swiftly complete their transactions, confident in the accuracy and reliability of the billing process.

4. **Real-time Updates**: One of the system's key features is its ability to provide real-time updates on medication stock levels. After a successful transaction, the system automatically adjusts the stock of purchased medicines, reflecting the latest inventory status. This dynamic updating mechanism ensures that pharmacists access accurate and up-to-date information, empowering them to make informed medication replenishment and inventory management decisions. By facilitating real-time updates, the system enhances operational efficiency and minimizes the risk of stockouts, ensuring seamless service delivery to customers.

Explanation of the Code

```
int main() {
    int choice, drug_id, quantity;
    int cashorcard;

    struct Drug demo_drugs[] = {
        {1, "Paracetamol", "Pain reliever", 2.5, 50},
        {2, "Ibuprofen", "Anti-inflammatory", 3.0, 40},
        {3, "Aspirin", "Blood thinner", 1.8, 60},
        {4, "Omeprazole", "Antacid", 4.5, 30},
        {5, "Loratadine", "Antihistamine", 3.2, 20},
        {6, "Mortar-LC", "To stop flared up Sinus", 6.9, 100},
        {7, "Meftal-Spas", "Stop Stomach-ache", 2.0, 70},
        {8, "Flomist", "Prevent Nose-Bleeding", 4.2, 55},
        {9, "Dettol", "Disinfectant", 2.0, 90},
        {10, "Benadryl", "Cure chest congestion", 3.9, 10}
    };
```

Variable Declarations:

- int choice, drug_id, quantity: These variables store user input for drug selection, drug ID, and quantity.
- int cashorcard: Stores the user's payment method choice.

Struct Definition:

struct Drug: Defines a structure named Drug with fields:

- int id: ID of the Drug.
- char name[100]: Name of the Drug.
- char description[100]: Description of the Drug.
- float price: Price of the Drug.
- int quantity: Quantity of the Drug available.

Array Initialization:

- struct Drug demo_drugs[]: Declares an array of Drug structs named demo_drugs and initializes it with data for 10 different drugs.

Each drug entry has an ID, name, description, price, and quantity available.

```
96              case 4:
97                  display_cart();
98                  break;
99
100             case 5:
101                 checkout();
102                 printf("Please follow the checkout procedure below: \n");
103                 printf("Press 1 to pay by Cash or press 2 to pay by Card?: ");
104                 scanf("%d", &cashorcard);
105                 if(cashorcard==1)
106                 {
107                     printf("An employee will be attending you shortly to collect the cash.\n");
108                     printf("Have a good day.");
109                     exit(0);
110                 }
111                 else
112                 {
113                     printf("Please tap or insert your card in the machine on your right hand side.\n");
114                     printf("Have a good day.");
115                     exit(0);
116                 }
117                 break;
118
119             case 6:
120                 exit(0);
121
122             default:
123                 printf("Invalid choice!\n");
124         }
125     }
```

```c
    while (1)
    {
        printf("\n-- Drug Store Management System --\n");
        printf("1. Display Available Drugs\n");
        printf("2. Add Drugs to Cart\n");
        printf("3. Remove Drugs from Cart\n");
        printf("4. View Cart\n");
        printf("5. Checkout\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) //Switch used for ease of choice.
        {
            case 1:
                display_drugs();
                break;

            case 2:
                printf("Enter Drug ID: ");
                scanf("%d", &drug_id);
                printf("Enter Quantity: ");
                scanf("%d", &quantity);
                add_to_cart(drug_id - 1, quantity);  // Adjusting index.
                break;

            case 3:
                printf("Enter Drug ID (from cart): ");
                scanf("%d", &drug_id);
                printf("Enter Quantity to Remove: ");
                scanf("%d", &quantity);
                remove_from_cart(drug_id - 1, quantity);  // Adjusting index like in case 2.
                break;
```

The code presents a menu for a Drug Store Management System, allowing users to display available drugs, add or remove drugs from a cart, view the cart, or proceed to checkout. It loops indefinitely until the user opts to exit. Each menu option corresponds to a function call, or an exit command based on user input.

```c
void display_drugs()
{
    printf("\n** Available Drugs **\n");
    printf("ID | Name                | Description                                  | Price   | Quantity\n");
    printf("-------------------------------------------------------------------------------------------------------------------------------\n");

    for (int i=0; i<num_drugs; i++)
    {
        printf("%-2d | %-20s | %-50s | %.2f\t| %d\n", drugs[i].id, drugs[i].name, drugs[i].description,
        drugs[i].price, drugs[i].quantity - cart_quantity[i]);
    }
}
```

The function **display_drugs** prints a formatted list of available drugs, including their ID, name, description, price, and quantity. It iterates through each drug in the inventory, subtracting the quantity in the cart, and displays the information in a table format.

```c
// Function to add drugs to the cart
void add_to_cart(int drug_id, int quantity)
{

    if (drug_id<0 || drug_id>=num_drugs)
    {
        printf("Invalid drug ID.\n");
        return;
    }
    if (quantity<=0)
    {
        printf("Invalid quantity.\n");
        return;
    }
    if (drugs[drug_id].quantity-cart_quantity[drug_id] < quantity)
    {
        printf("Insufficient quantity in stock.\n");
        return;
    }

    // Add drug to cart
    cart[cart_size][0]=drug_id;
    cart[cart_size][1]=quantity;
    cart_size++;
    cart_quantity[drug_id]+=quantity;

    printf("Drug successfuly added to cart.\n");
}

```

The function `add_to_cart(int drug_id, int quantity)` checks if the provided drug ID and quantity are valid. If the drug ID is out of range or the quantity is non-positive, it displays an error message. If there is insufficient stock, it notifies the user. Otherwise, it adds the drug to the cart and updates the cart's size and quantity. Finally, it confirms the successful addition to the cart.

```c
void remove_from_cart(int drug_id, int quantity)
{

    int index=-1;
    for (int i=0; i<cart_size; i++)
    {
        if (cart[i][0] == drug_id)
        {
            index=i;
            break;
        }
    }

    // If the drug is not found in the cart
    if (index == -1)
    {
        printf("Drug not found in cart.\n");
        return;
    }

    // Check if the quantity to remove is valid
    if (quantity<=0 || quantity>cart[index][1])
    {
        printf("Invalid quantity.\n");
        return;
    }

    // Update the quantity in the cart
    cart[index][1] -= quantity;
    cart_quantity[drug_id] -= quantity;

    // If the quantity becomes zero, remove the drug entry from the cart
    if (cart[index][1] == 0)
    {
        for (int i=index; i<cart_size - 1; i++)
        {
            cart[i][0] = cart[i+ 1][0];
            cart[i][1] = cart[i+ 1][1];
        }
        cart_size--;
    }

    printf("Drug removed from cart.\n");
```

This code defines a function **remove_from_cart()** that removes a specified quantity of a drug from the shopping cart. Here's what it does:

1. It searches for the drug with the specified ID in the cart.
2. If the drug is not found in the cart, it prints a message indicating that the drug was not found and returns.
3. If the specified quantity to remove is invalid (i.e., less than or equal to 0, or greater than the quantity of the drug in the cart), it prints a message indicating that the quantity is invalid and returns.
4. It updates the quantity of the drug in the cart by subtracting the specified quantity to remove.
5. If the updated quantity becomes zero, it removes the drug entry from the cart by shifting the subsequent entries to fill the gap.
6. It decrements the **cart_size** variable to reflect the removal of the drug entry.
7. It prints a message indicating that the drug was successfully removed from the cart.

```
}

// Function to display the cart
void display_cart()
{
    printf("\n** Cart **\n");
    printf("ID | Name                | Description                                | Price   | Quantity\n");
    printf("-------------------------------------------------------------------------------------------------------------------\n");

    float total = 0;
    for (int i=0; i<cart_size; i++)
    {
        int id = cart[i][0];
        int quantity = cart[i][1];
        printf("%-2d | %-20s | %-50s | %.2f\t| %d\n", drugs[id].id, drugs[id].name, drugs[id].description, drugs[id].price, quantity);
        total += drugs[id].price * quantity;
    }

    total *= (1 + TAX_RATE);     // Adding  tax
    printf("Total (+ 8%% Texas State Tax): %.2f\n", total);
}
```

This code defines a function **display_cart()** that displays the contents of the shopping cart along with the total cost including tax. Here's a breakdown of what it does:

1. It prints a header indicating that it's displaying the cart.
2. It prints a table header with columns for item ID, name, description, price, and quantity.
3. Inside a loop iterating over each item in the cart:

- It retrieves the item ID and quantity from the **cart** array.
- It prints the item's ID, name, description, price, and quantity in a formatted manner.
- It calculates the subtotal for each item (price * quantity) and accumulates it into the **total** variable.

4. After the loop, it calculates the total cost by adding tax (defined by **TAX_RATE**) to the subtotal.
5. It prints the total cost, including tax, for all items in the cart.

Overall, this function provides a detailed display of the items in the cart, including their attributes, and calculates the total cost including tax.

```c
// Performing checkout
void checkout()
{
    printf("\n-- Checkout --\n");
    display_cart();
    printf("Total amount to pay (+ 8%% Texas State Tax): %.2f\n", calculate_total(cart, cart_size)*(1 + TAX_RATE));

    for (int i = 0; i < cart_size; i++)
    {
        int id = cart[i][0];
        int quantity = cart[i][1];
        drugs[id].quantity -= quantity;
    }

    cart_size = 0; //This is to empty the cart afterwards.
    memset(cart_quantity, 0, sizeof(cart_quantity));

    printf("Thank you for shopping with us!\n");
```

This code defines a function **display_cart()** that displays the contents of the shopping cart along with the total cost including tax. Here's a breakdown of what it does:

1. It prints a header indicating that it's displaying the cart.
2. It prints a table header with columns for item ID, name, description, price, and quantity.
3. Inside a loop iterating over each item in the cart:
    - It retrieves the item ID and quantity from the **cart** array.

- It prints the item's ID, name, description, price, and quantity in a formatted manner.
- It calculates the subtotal for each item (price * quantity) and accumulates it into the **total** variable.

4. After the loop, it calculates the total cost by adding tax (defined by **TAX_RATE**) to the subtotal.
5. It prints the total cost, including tax, for all items in the cart.

Overall, this function provides a detailed display of the items in the cart, including their attributes, and calculates the total cost including tax.

```
// Function to calculate the total cost
float calculate_total(int cart[][2], int cart_size)
{
    float total = 0;
    for (int i=0; i<cart_size; i++)
    {
        int id = cart[i][0];
        int quantity = cart[i][1];
        total += drugs[id].price * quantity;
    }
    return total;
} //End of code.
```

This code defines a function **calculate_total** that computes the total cost of items in a shopping cart. It takes a two-dimensional array **cart** representing the items in the cart and its size **cart_size**. Inside the function, it iterates over each item in the cart, calculates the cost of each item by multiplying its price (retrieved from a **drugs** array using its ID) by its quantity, and accumulates the total cost. Finally, it returns the total cost as a float value.

Conclusion

In conclusion, the HealthMart Pharmacy Management System represents a pivotal advancement in pharmaceutical retail, poised to revolutionize industry standards. Its amalgamation of cutting-edge technology and meticulous attention to customer-centricity underscores its commitment to operational excellence and patient well-being. By seamlessly integrating features such as efficient stock management, interactive drug selection, and real-time updates, the system enhances operational efficiency and elevates the overall quality of service delivery. Furthermore, its modular architecture ensures adaptability to dynamic market landscapes, ensuring sustained relevance and competitive edge for pharmacies. With a steadfast focus on precision, innovation, and customer satisfaction, the HealthMart Pharmacy Management System epitomizes the pinnacle of excellence in pharmaceutical retail management, heralding a new era of efficiency and efficacy in healthcare provision.