1. **Aim of the Microproject**

The aim of this micro project is to develop a simplified Bookshop Management System (BMS) using C++. This system will streamline key processes, including story books, General Books , Magazine and sales tracking.

A central feature is the customer registration system, ensuring each customer is unique and allowing them to check their registration status. The system will facilitate book sales, ensuring each book is sold only if in stock. Accurate sales tracking is essential, and the system will provide ways to tally sales for each book, displaying results in real-time. The user interface will be simple and intuitive, making management easy for both staff and customers.

1. **Proposed Methodology**

The proposed methodology for enhancing the online bookstore application focuses on improving code organization, user experience, and data management. First, we recommend implementing a clear separation of concerns by creating distinct classes for user interface handling, data validation, and core business logic. This not only enhances maintainability but also allows for easier testing and future feature expansions, such as adding search and sorting functionalities for books. Additionally, robust error handling should be introduced to manage invalid inputs and operations gracefully, ensuring a smooth user experience.

Secondly, incorporating data persistence mechanisms would allow users to save and load their shopping cart and book inventory, enhancing the application's practicality. Implementing input validation and clearer user prompts will further streamline interactions. Lastly, providing comprehensive documentation and user guides will assist users in navigating the system effectively. By following this methodology, the online bookstore can evolve into a more user-friendly, reliable, and scalable application, catering to a broader audience and facilitating better user engagement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Detail of activity** | **Plan Start Date** | **Plan Finish**  **Date** |
|  | Collect information on existing Bookshop management systems and best practices. | 14/10/2024 | 15/10/2024 |
| 1. 2 | Implement the code for the Bookshop management system, including classes and methods. | 15/10/2024 | 18/10/2024 |
| 1. 3 | Create a report about the system. | 19/10/2024 | 20/10/2024 |
|  | Test Functionality | 21/10/2024 | 22/10/2024 |
| 1. 4 | Review and finalize the report. | 25/10/2024 | 25/10/2024 |

1. **Action Plan**
2. **Brief Description of Micro project**

The Micro Project on Bookshop Management involves creating a software application designed to streamline the online shopping experience for customers. This system manages an inventory of books, allowing users to view available titles across various categories, including general books, storybooks, and magazines. Each book is represented by a dedicated class containing relevant attributes such as title, author, price, and Book id. The user-friendly interface facilitates easy navigation, enabling customers to browse books, add them to a shopping cart, and manage their selections seamlessly.

Additionally, the application includes features for customer registration, allowing users to input their details, which enhances the shopping experience. Customers can view their shopping cart, modify their orders, and generate bills that detail their purchases. The system ensures effective inventory management while providing immediate feedback on user actions, contributing to a transparent and efficient online bookstore operation.

**4.1 Used Concept**

The Bookshop Management System utilizes object-oriented programming concepts in C++ to create a structured and efficient online bookstore application. Key components include the `Book`, CartItem , and `ShoppingCart` classes, which encapsulate relevant data and functionality. The `Book` class represents individual books with attributes like title, author, price, ISBN, and category, while the `CartItem` class pairs books with their quantities in the shopping cart. The `ShoppingCart` class manages customer details and facilitates operations such as adding or removing books and generating bills.

**1. Classes and Objects:**

- The code defines multiple classes (`Book`, `CartItem`, `ShoppingCart`, `OnlineBookstore`) that encapsulate data and behavior. Each class represents a distinct entity in the system.

**2. Encapsulation:**

- Data members (attributes) and member functions (methods) are encapsulated within classes. Access to these members is controlled, typically through public and private access modifiers.

**3. Constructors:**

- Constructors are used to initialize objects. The `Book` and `CartItem` classes have constructors that allow for initialization with parameters.

**4. Array Usage:**

- Arrays (`cart` in `ShoppingCart` and `books` in `OnlineBookstore`) are used to store multiple objects of the same type, demonstrating how to manage collections of data.

**5. Dynamic Behavior:**

- The code uses a simple approach to dynamically manage the shopping cart, allowing users to add, view, and delete books.

**6. Input/Output (I/O):**

- The program uses `cin` and `cout` for user input and output, allowing for interaction with the user.

**7. Control Structures:**

- Various control structures (if statements, loops, and switch statements) are utilized to manage the flow of the program, handle user choices, and implement logic.

**8. Functions:**

- Member functions are defined within classes to encapsulate behaviors (e.g., `addBook`, `viewCart`, `generateBill`), showcasing the modularity of code.

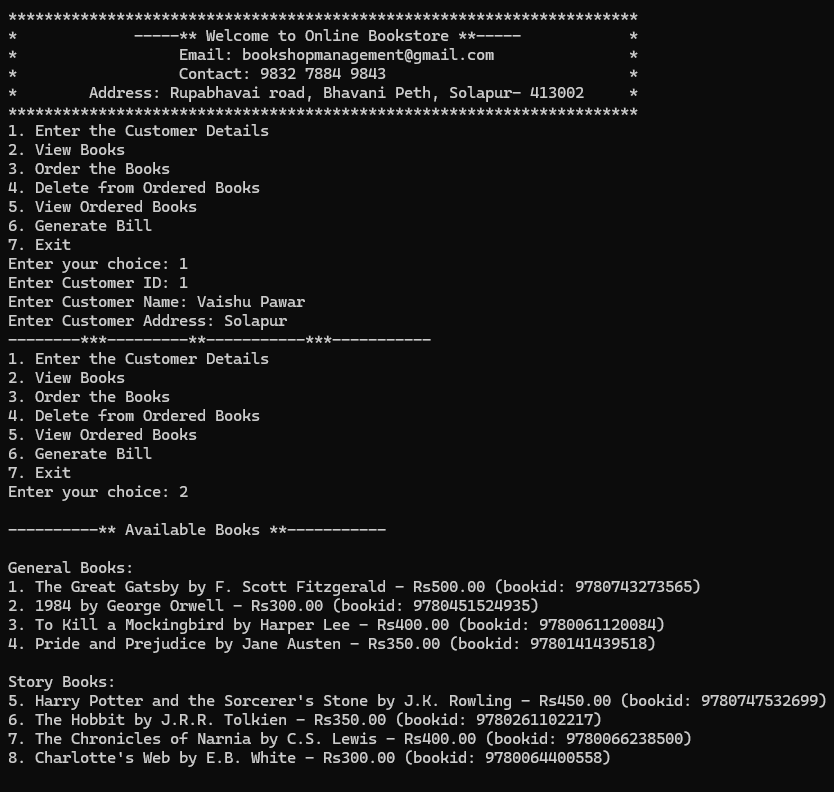
**9. String Manipulation:**

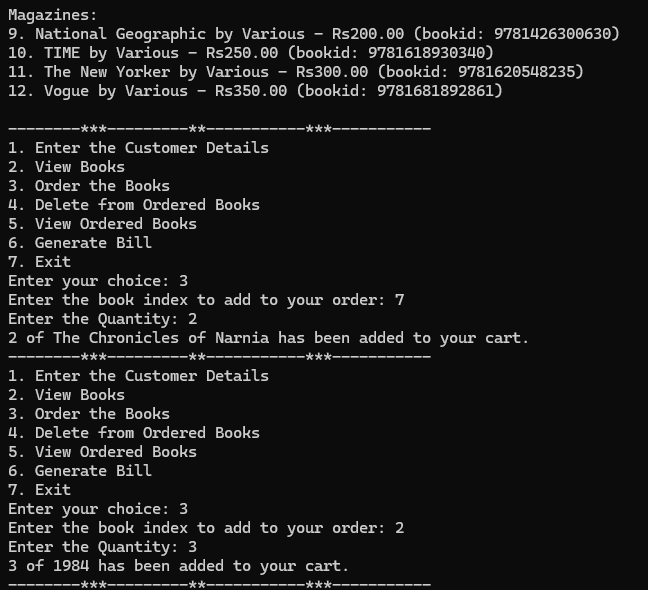
- The program makes use of the `string` class for handling text, such as book titles, author names, and customer details.

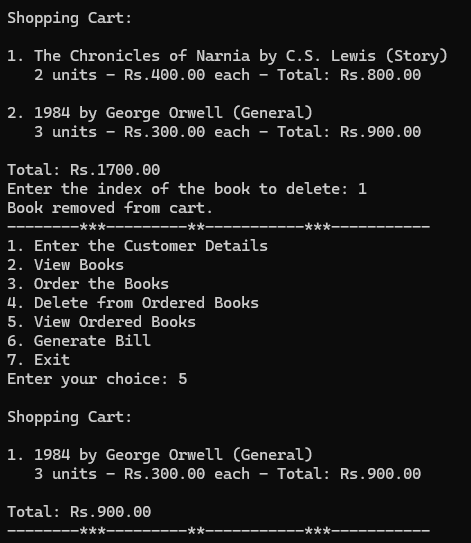
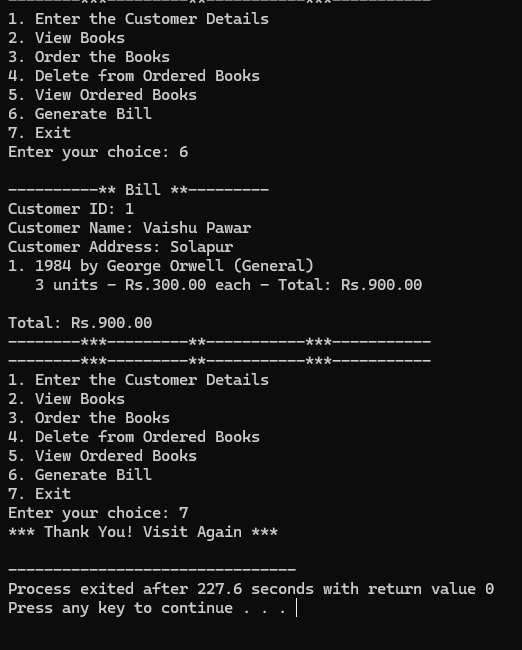
These concepts together create a functional and interactive online bookstore application, demonstrating object-oriented programming principles effectively.

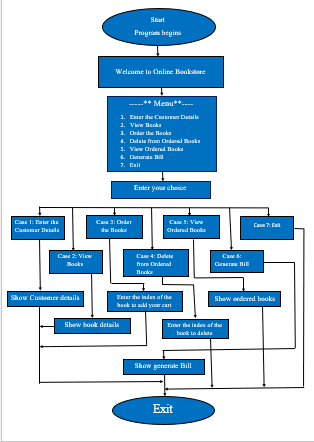
**4.2 Programs or code of relevant concepts and output**

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <string>  using namespace std;  const int MAX\_BOOKS = 100; // Maximum number of books  const int MAX\_CART\_ITEMS = 100; // Maximum items in the cart  class Book {  public:  string title;  string author;  double price;  string ISBN;  string category;  Book() : title(""), author(""), price(0.0), ISBN(""), category("") {}  Book(string t, string a, double p, string isbn, string cat) : title(t), author(a), price(p), ISBN(isbn), category(cat) {}  };  class CartItem {  public:  Book book;  int quantity;  CartItem() : quantity(0) {}  CartItem(Book b, int q) : book(b), quantity(q) {}  };  class ShoppingCart {  public:  CartItem cart[MAX\_CART\_ITEMS];  int cartSize;  string customerID;  string customerName;  string customerAddress;  public:  ShoppingCart() : cartSize(0) {}  void setCustomerDetails(const string& id, const string& name, const string& address) {  customerID = id;  customerName = name;  customerAddress = address;  }  bool hasCustomerDetails() const {  return !customerID.empty() && !customerName.empty() && !customerAddress.empty();  }  void addBook(const Book& book, int quantity) {  for (int i = 0; i < cartSize; ++i) {  if (cart[i].book.title == book.title) {  cart[i].quantity += quantity;  return;  }  }  cart[cartSize++] = CartItem(book, quantity);  }  void deleteBook(int index) {  if (index < 0 || index >= cartSize) {  cout << "Invalid index." << endl;  return;  }  for (int i = index; i < cartSize - 1; ++i) {  cart[i] = cart[i + 1];  }  cartSize--;  cout << "Book removed from cart." << endl;  }  void viewCart() {  if (cartSize == 0) {  cout << "\nYour shopping cart is empty.\n" << endl;  return;  }  cout << "\nShopping Cart:\n" << endl;  double total = 0;  for (int i = 0; i < cartSize; ++i) {  double itemTotal = cart[i].book.price \* cart[i].quantity;  cout << i + 1 << ". " << cart[i].book.title << " by " << cart[i].book.author << " ("  << cart[i].book.category << ")" << endl  << " " << cart[i].quantity << " units - Rs."  << fixed << setprecision(2) << cart[i].book.price << " each - Total: Rs."  << itemTotal << endl;  total += itemTotal;  cout << endl;  }  cout << "Total: Rs." << fixed << setprecision(2) << total << endl;  }  void generateBill() {  if (!hasCustomerDetails()) {  cout << "Customer details are required to generate a bill." << endl;  return;  }  if (cartSize == 0) {  cout << "Your shopping cart is empty. Cannot generate bill." << endl;  return;  }  cout << "\n----------\*\* Bill \*\*---------" << endl;  cout << "Customer ID: " << customerID << endl;  cout << "Customer Name: " << customerName << endl;  cout << "Customer Address: " << customerAddress << endl;  double total = 0;  for (int i = 0; i < cartSize; ++i) {  double itemTotal = cart[i].book.price \* cart[i].quantity;  cout << i + 1 << ". " << cart[i].book.title << " by " << cart[i].book.author << " ("  << cart[i].book.category << ")" << endl  << " " << cart[i].quantity << " units - Rs."  << fixed << setprecision(2) << cart[i].book.price << " each - Total: Rs."  << itemTotal << endl;  total += itemTotal;  cout << endl;  }  cout << "Total: Rs." << fixed << setprecision(2) << total << endl;  cout << "--------\*\*\*---------\*\*-----------\*\*\*-----------" << endl;  }  };  class OnlineBookstore {  private:  Book books[MAX\_BOOKS];  int bookCount;  ShoppingCart cart;  public:  OnlineBookstore() : bookCount(0) {  // Sample books with categories  books[bookCount++] = Book("The Great Gatsby", "F. Scott Fitzgerald", 500.00, "9780743273565", "General");  books[bookCount++] = Book("1984", "George Orwell", 300.00, "9780451524935", "General");  books[bookCount++] = Book("To Kill a Mockingbird", "Harper Lee", 400.00, "9780061120084", "General");  books[bookCount++] = Book("Pride and Prejudice", "Jane Austen", 350.00, "9780141439518", "General");    // Adding story books  books[bookCount++] = Book("Harry Potter and the Sorcerer's Stone", "J.K. Rowling", 450.00, "9780747532699", "Story");  books[bookCount++] = Book("The Hobbit", "J.R.R. Tolkien", 350.00, "9780261102217", "Story");  books[bookCount++] = Book("The Chronicles of Narnia", "C.S. Lewis", 400.00, "9780066238500", "Story");  books[bookCount++] = Book("Charlotte's Web", "E.B. White", 300.00, "9780064400558", "Story");  // Adding magazines  books[bookCount++] = Book("National Geographic", "Various", 200.00, "9781426300630", "Magazine");  books[bookCount++] = Book("TIME", "Various", 250.00, "9781618930340", "Magazine");  books[bookCount++] = Book("The New Yorker", "Various", 300.00, "9781620548235", "Magazine");  books[bookCount++] = Book("Vogue", "Various", 350.00, "9781681892861", "Magazine");  }  void displayBooks() {  cout << "\n----------\*\* Available Books \*\*-----------\n" << endl;  cout << "General Books:\n";  for (int i = 0; i < bookCount; ++i) {  if (books[i].category == "General") {  cout << i + 1 << ". " << books[i].title << " by " << books[i].author << " - Rs"  << fixed << setprecision(2) << books[i].price << " (bookid: " << books[i].ISBN << ")" << endl;  }  }  cout << endl;  cout << "Story Books:\n";  for (int i = 0; i < bookCount; ++i) {  if (books[i].category == "Story") {  cout << i + 1 << ". " << books[i].title << " by " << books[i].author << " - Rs"  << fixed << setprecision(2) << books[i].price << " (bookid: " << books[i].ISBN << ")" << endl;  }  }  cout << endl;  cout << "Magazines:\n";  for (int i = 0; i < bookCount; ++i) {  if (books[i].category == "Magazine") {  cout << i + 1 << ". " << books[i].title << " by " << books[i].author << " - Rs"  << fixed << setprecision(2) << books[i].price << " (bookid: " << books[i].ISBN << ")" << endl;  }  }  cout << endl;  }  void addToCart(int bookIndex) {  if (bookIndex < 1 || bookIndex > bookCount) {  cout << "Invalid book index." << endl;  return;  }  int quantity;  cout << "Enter the Quantity: ";  cin >> quantity;  if (quantity <= 0) {  cout << "Quantity must be positive." << endl;  return;  }  cart.addBook(books[bookIndex - 1], quantity);  cout << quantity << " of " << books[bookIndex - 1].title << " has been added to your cart." << endl;  }  void deleteFromCart() {  if (cart.cartSize == 0) {  cout << "Your cart is empty. No items to delete." << endl;  return;  }    cart.viewCart();  int itemIndex;  cout << "Enter the index of the book to delete: ";  cin >> itemIndex;  cart.deleteBook(itemIndex - 1);  }  void viewCart() {  cart.viewCart();  }  void generateBill() {  cart.generateBill();  }  void inputCustomerDetails() {  string id, name, address;    cout << "Enter Customer ID: ";  cin >> id;  cout << "Enter Customer Name: ";  cin.ignore();  getline(cin, name);  cout << "Enter Customer Address: ";  getline(cin, address);  cart.setCustomerDetails(id, name, address);  }  void start() {  int choice = 0;  cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "\* -----\*\* Welcome to Online Bookstore \*\*----- \*" << endl;  cout << "\* Email: bookshopmanagement@gmail.com \*" << endl;  cout << "\* Contact: 9832 7884 9843 \*" << endl;  cout << "\* Address: Rupabhavai road, Bhavani Peth, Solapur- 413002 \*" << endl;  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  while (true) {  cout << "1. Enter the Customer Details" << endl;  cout << "2. View Books" << endl;  cout << "3. Order the Books" << endl;  cout << "4. Delete from Ordered Books" << endl;  cout << "5. View Ordered Books" << endl;  // cout << "5. Enter the Customer Details" << endl;  cout << "6. Generate Bill" << endl;  cout << "7. Exit" << endl;  cout << "Enter your choice: ";  cin >> choice;  switch (choice) {  case 1:  inputCustomerDetails();  break;  case 2:  displayBooks();  break;  case 3: {  int bookIndex;  cout << "Enter the book index to add to your order: ";  cin >> bookIndex;  addToCart(bookIndex);  break;  }  case 4:  deleteFromCart();  break;  case 5:  viewCart();  break;  // case 5:  // inputCustomerDetails();  // break;  case 6:  generateBill();  break;  case 7:  cout << "\*\*\* Thank You! Visit Again \*\*\*" << endl;  return;  default:  cout << "Invalid choice, please try again." << endl;  break;  }  cout << "--------\*\*\*---------\*\*-----------\*\*\*-----------" << endl;  }  }  };  int main() {  OnlineBookstore system;  system.start();  return 0;  } |

1. **Output**

****

****

1. **Flow of Program**
2. **Advantages**

Here are four advantages of the Bookshop Management System:

1. Streamlined Shopping Experience: The application simplifies the process of browsing, selecting, and purchasing books, making it easy for customers to find and order items quickly.

2. Effective Inventory Management: With the ability to categorize books, the system helps manage a diverse range of titles, making it easier for both customers and store operators to keep track of available stock.

3. Personalized Customer Interaction: The system allows customers to input their details for billing, fostering a personalized shopping experience and enhancing customer service.

4. Comprehensive Billing Features: The capability to generate detailed bills improves transparency in transactions, helping customers understand their purchases and total costs easily.

1. **Disadvantages.**

Here are four disadvantages of the Bookshop Management System:

1. Limited Scalability: The system has a predefined maximum number of books and cart items (100 each), which may not be sufficient for larger bookstores or those looking to expand their inventory.

2. Basic User Interface: The console-based interface may not be user-friendly or visually appealing, which could deter some customers who prefer more modern, graphical interfaces.

3. No Online Payment Integration: The absence of payment processing features means transactions must be handled separately, potentially complicating the checkout process and affecting customer convenience.

4. Lack of Data Analytics: The system does not provide analytics or reporting tools, limiting the ability to track sales trends, customer preferences, and inventory performance for better decision-making.

1. **Applications**
2. **Shopping Cart Functionality**

The system allows users to add multiple books to a shopping cart, adjust quantities, and easily delete items, ensuring a flexible shopping experience.

1. **Customer Details and Billing**

Users can enter their personal information to generate a detailed bill at checkout, making the transaction process smooth and organized.

1. **Search Functionality**

Implement a search feature that allows users to search for books by title, author, or Book id. This would make it easier for users to find specific books quickly.

1. **User Registration and Login**

Add a user authentication system where customers can create accounts, log in, and manage their orders. This could include saving cart items for later and viewing order history.

1. **Order History and Tracking**

Allow users to view their past orders, including order details and statuses. This feature can enhance the user experience by keeping track of their purchases.

1. **Book Ratings and Reviews**

Enable users to rate and review books they’ve purchased. This adds a community aspect to the bookstore and can help other customers make informed decisions.

1. **Conclusion**

In conclusion, the Bookshop Management System provides a foundational framework for managing book inventory and customer transactions effectively. While it offers advantages such as organized cataloging of books, a straightforward shopping cart feature, and basic customer interaction capabilities, it also presents notable drawbacks. Limitations in scalability, a basic user interface, lack of online payment options, and insufficient data analytics hinder its effectiveness in a competitive retail environment. For bookstores aiming for growth and enhanced customer experiences, further development and modernization of the system would be essential to meet evolving market demands and improve operational efficiency.

1. **Reference**

<https://www.geeksforgeeks.org/bookshop-management-system-using-file-handling/>

<https://www.scribd.com/doc/124378514/Book-Shop-Management-System-Documentation>

<https://sites.google.com/site/ignoubcafinalyearprojects/project-report/book-shop-management-system-project-report>