SIMPLE E-- COMMERCE SYSTEM DESIGN USING C LANGUAGE

NAME – RN Umashankar
COLLEGE- REVA University
SEMESTER- 7TH
B TECH Electronics and Communication

INTRODUCTION

- A simple e-commerce system in C serves as a foundational platform that allows users to browse, select, and purchase products online.
- This system typically includes key functionalities such as user registration and login, product management, a shopping cart, and order processing.
- By implementing essential components like user and product management modules, the system enables users to create accounts, view available products, add items to their cart, and complete purchases.
- Utilizing basic file handling techniques for data storage, this system provides a practical introduction to the concepts of e-commerce, user interactions, and data management in a C programming environment.
- This project not only enhances programming skills but also offers insights into the inner workings of online retail platforms.

OBJECTIVES

•**User Management**:

•**User Registration**: Allow users to create an account by providing essential information such as username, password, and email. Ensure data validation to prevent duplicate registrations.

•**User Authentication**: Implement a secure login process that verifies user credentials and grants access to their account.

•**Profile Management**: Enable users to view and update their account details, including password changes.

•**Product Management**:
•**Product Catalog**: Develop a module to display a list of products with essential details like name, description, price, and stock availability.

•**Product Addition**: Provide functionality for administrators to add new products to the catalog, ensuring proper data validation for product information.
•**Product Search and Filtering**: Implement search functionality that allows users to find products by name or category.

- **Shopping Cart Functionality**:

- **Add to Cart**: Allow users to easily add products to their shopping cart and specify quantities.

- **View Cart**: Create a user-friendly interface for users to view items in their cart, including total price calculations.

- **Update and Remove Items**: Enable users to adjust quantities or remove items from the cart as needed.

- **Order Processing**:

- **Checkout Process**: Design a streamlined checkout process where users can review their cart, provide shipping information, and confirm their order.

- **Order Confirmation**: Implement a confirmation mechanism that displays order details and generates a summary after successful payment processing.

- **Data Storage and Management**:

  Persistent Storage: Use file handling to save and retrieve user accounts, product information, and order histories, ensuring data is retained between sessions.

- **Data Integrity**: Implement measures to ensure the integrity and consistency of data during transactions and updates.

- **User Interface Design**:

- **Text-Based Menu**: Develop a clear and intuitive text-based user interface that guides users through different functionalities of the e-commerce system.

- **Error Handling**: Implement robust error handling to manage invalid inputs and provide user-friendly feedback.

- **Code Modularity and Documentation**:
- **Modular Design**: Structure the code into distinct modules (e.g., user, product, cart, order) for better organization and maintainability.

- **Documentation**: Provide clear comments and documentation within the code to explain functionality, making it easier for future developers to understand and modify.

- **Future Expansion**:

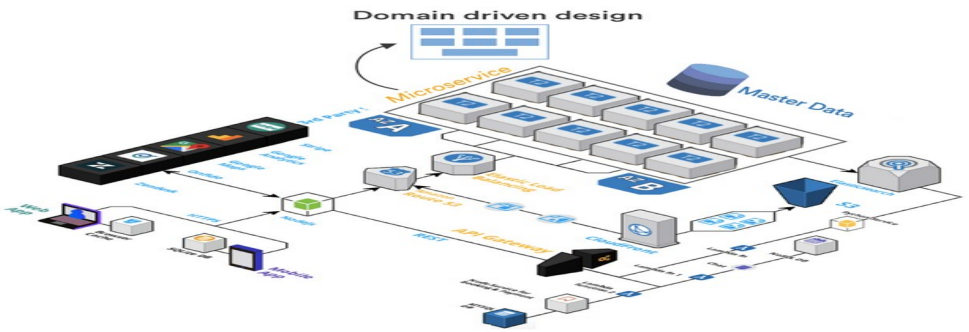
- **Scalability Considerations**: Design the system with future enhancements in mind, such as integrating payment gateways, inventory management, and advanced user analytics.

- **Learning and Improvement**: Use the project as a learning tool to explore more advanced programming concepts and technologies that can be integrated into a full-fledged e-commerce platform.

METHODOLOGY

- **Requirement Analysis**
- **Identify User Needs**: Gather requirements by defining the target user base and understanding their needs (e.g., browsing products, making purchases).
- **Define Functional Requirements**: Outline key features such as user registration, product management, shopping cart functionality, and order processing.
- **Determine Non-Functional Requirements**: Establish performance criteria, usability standards, and security measures.

- **System Design**
- **Architecture Design**:
  - Design the overall system architecture, dividing it into modules (User Management, Product Management, Cart Management, Order Processing).
- **Data Structures**:
  - Define data structures for users, products, and shopping carts, ensuring they accommodate necessary attributes (e.g., username, product name, price).
- **File Management**:
  - Plan how data will be stored (e.g., using text files for user accounts and product listings) and how the system will read and write to these files.



Function Implementation:

- Write functions for core functionalities:
  - **User Module**: registerUser, loginUser, updateProfile
  - **Product Module**: addProduct, viewProducts, searchProducts
  - **Cart Module**: addToCart, viewCart, removeFromCart
  - **Order Module**: checkout, confirmOrder

Implementation

- **Development Environment Setup**:
  - Choose a suitable IDE or text editor for C programming (e.g., Code::Blocks, Visual Studio, or a simple text editor).
- **Modular Programming**:
  - Implement each module separately, focusing on one functionality at a time (e.g., start with user registration, then move to product management).

- **User Interface Development**
- **Menu Design**:
  - Create a text-based menu that provides users with options for registration, login, product browsing, and order management.
- **Input Handling**:
  - Implement input validation to ensure user entries are valid and provide error messages for incorrect inputs.

- **Testing**
- **Unit Testing**:
  - Test individual functions and modules to ensure they work as expected. Check for edge cases and handle errors gracefully.
- **Integration Testing**:
  - Test the interaction between different modules to ensure they work together seamlessly (e.g., verify that users can add products to their cart and proceed to checkout).

- **Documentation**
- **Code Comments**:
  - Write clear comments in the code to explain the purpose of functions and complex logic.
- **User Documentation**:
  - Create a user manual that outlines how to navigate the system, including registration, logging in, and making purchases.
- **Deployment**
- **Compile and Build**:
  - Compile the code and ensure there are no errors. Create an executable file for users.
- **Distribution**:
  - Share the application with users, along with documentation and installation instructions.

- **Learning and Reflection**
- **Post-Mortem Analysis**:
  - Reflect on the development process, noting challenges faced and lessons learned to improve future projects.
- **Explore Advanced Concepts**:
  - Investigate advanced programming techniques and tools (e.g., databases, web frameworks) that could enhance the e-commerce system.
- **Maintenance and Future Enhancements**
- **Feedback Loop**:
  - Collect feedback from users post-deployment to identify issues and areas for improvement.
- **Iterative Improvements**:
  - Plan for future enhancements, such as adding a graphical user interface (GUI), integrating payment processing, and expanding product features.

IMPORTANCE OF USING C LANGUAGE IN THIS PROJECT

•**Performance and Efficiency**:

•C is known for its high performance and low-level access to memory, allowing for efficient resource management.

•This is particularly important for applications that require quick data processing, like handling user requests and managing transactions in an e-commerce system.

•**Portability**:

•Programs written in C can be easily ported to different platforms with minimal changes. This portability allows the e-commerce system to run on various operating systems, making it versatile for deployment.

- **Control Over System Resources**:

- C provides direct access to memory and system resources through pointers and manual memory management.

- This level of control is beneficial for optimizing the system's performance and managing resources effectively, especially in scenarios with high user traffic.

- **Foundation for Understanding Other Languages**:

- Learning C enhances understanding of programming concepts that are applicable in higher-level languages (e.g., C++, Java).

- This foundational knowledge can be advantageous when extending the system or transitioning to more complex programming environments.

- **Extensive Libraries**:
- C has a rich set of libraries that can be utilized for various functionalities, including string manipulation, file handling, and data structure management.

- These libraries simplify the development process and reduce the time required to implement features.

- **Strong Typing and Efficiency in Data Handling**:
- C is a statically typed language, which helps in catching errors at compile-time rather than runtime.

- This can lead to more robust code, essential in an e-commerce environment where data integrity is critical.

- **Real-Time Processing**:

- C is often used in real-time systems due to its efficiency and speed.

- For an e-commerce platform, real-time processing of transactions and user interactions can enhance the overall user experience.
- **Wide Usage in Systems Programming**:
- Many operating systems, databases, and network protocols are implemented in C.

- Understanding C can provide insights into how these systems work, which can be useful when developing features related to data storage, retrieval, and networking in the e-commerce system.

- **Community and Resources**:

- C has a vast community and abundant resources, including documentation, tutorials, and forums.

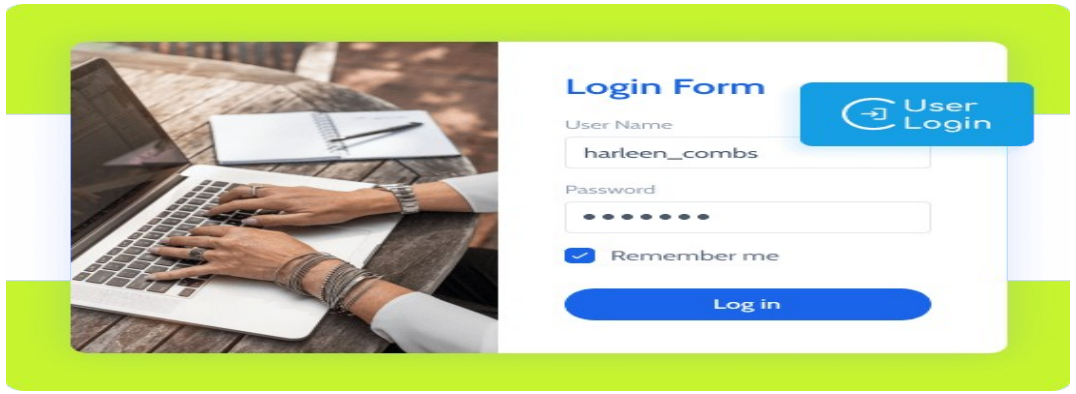
- This support can be invaluable for developers, especially those new to programming or specific challenges encountered during development.

- **Legacy Support**:

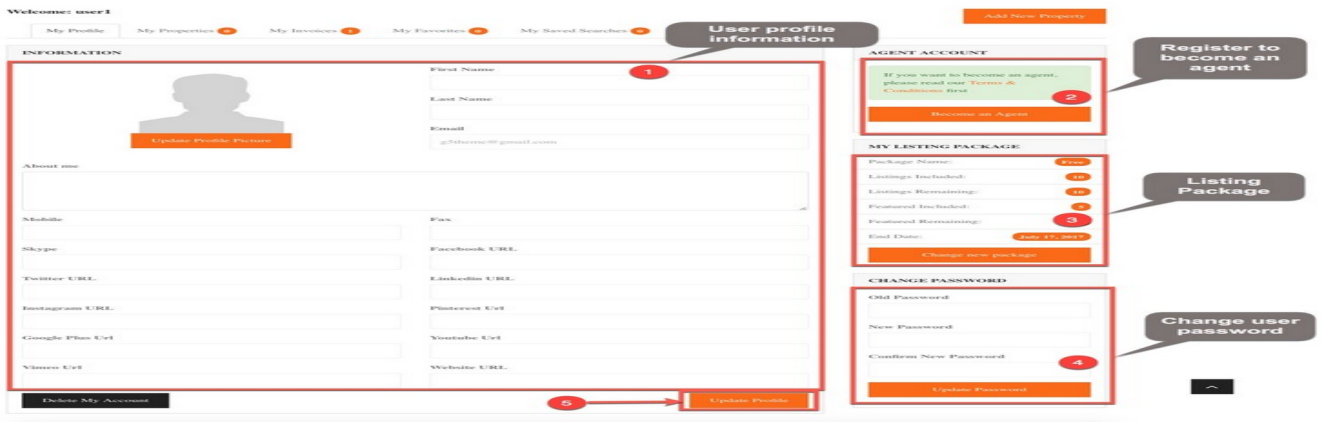
- Many existing systems and applications are built using C. Developing a new system in C allows for better integration and communication with legacy systems, which may still be in use in many organizations.

WORKING OF THIS PROJECT

- **User Login**:
- **Functionality**: Users can log in to access their accounts.
- **Process**:
  - The user enters their credentials.
  - The system reads from the user data file and verifies the credentials.
  - Successful login grants access to the main menu, while failures prompt the user to retry.



- **Profile Management**:
- **Functionality**: Users can view and update their account information.
- **Process**:
  - After logging in, users select the profile management option.
  - The system displays current user information.
  - Users can update their password or other details, which are then rewritten to the data file.



- **Product Management Module**
- **Adding Products** (Admin Functionality):
  - **Functionality**: Admins can add new products to the catalog.
  - **Process**:
    - Admin selects the option to add a product.
    - Admin inputs product details (name, description, price, stock).
    - The product information is saved in a products text file.

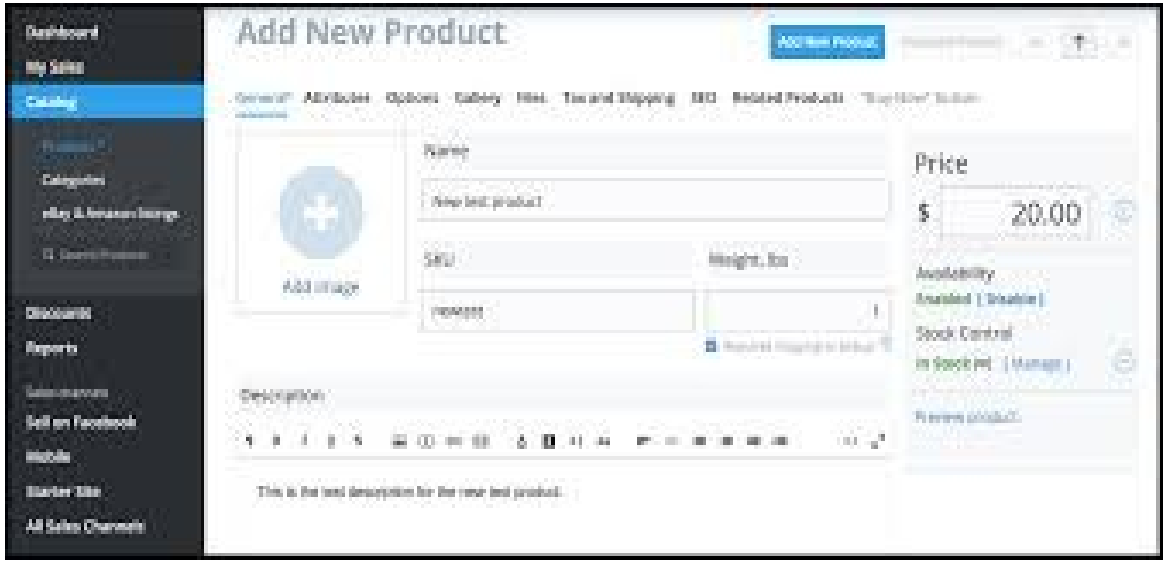
- **Viewing Products**:
  - **Functionality**: Users can browse the product catalog.
  - **Process**:
    - The user selects the view products option.

    - The system reads from the product data file and displays all available products in a user-friendly format.

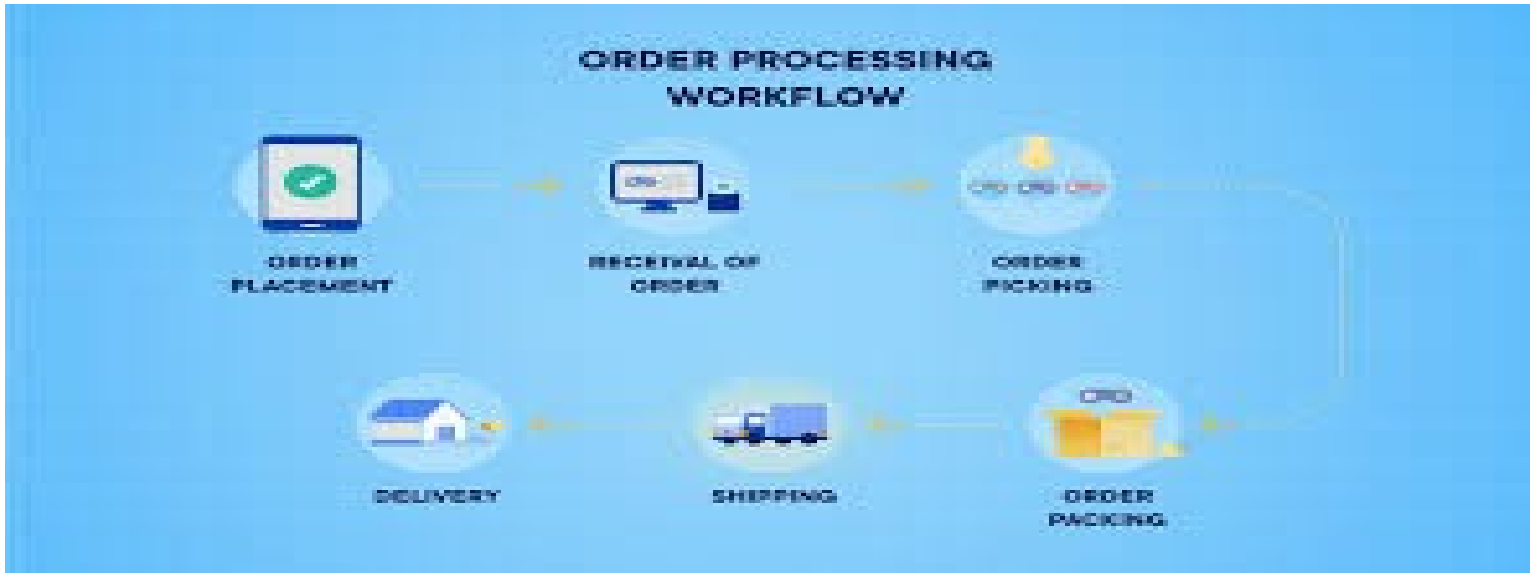
- **Searching for Products**:
- **Functionality**: Users can search for specific products.
- **Process**:
  - Users enter search keywords.
  - The system filters the product list and displays matching results based on name or category.

- **Shopping Cart Module**
- **Adding Items to the Cart**:
  - **Functionality**: Users can add selected products to their shopping cart.
  - **Process**:
    - Users select a product and specify the quantity.
    - The product and quantity are added to an in-memory cart structure (e.g., an array or linked list).
- **viewing the Cart**:
- **Functionality**: Users can review items in their cart before checkout.
- **Process**:
  - The user selects the view cart option.
  - The system displays all items, including product details and total price.

- **Updating and Removing Items**:
- **Functionality**: Users can modify cart items.
- **Process**:
  - Users specify changes (e.g., increase quantity, remove items).
  - The cart is updated accordingly, and changes are reflected in the total price.



- **Order Processing Module**
- **Checkout Process**:
  - **Functionality**: Users finalize their purchase.
  - **Process**:
    - Users select the checkout option from the cart.
    - The system prompts for shipping information and confirms the order details.
    - The order is saved to an orders file, including user and product details.



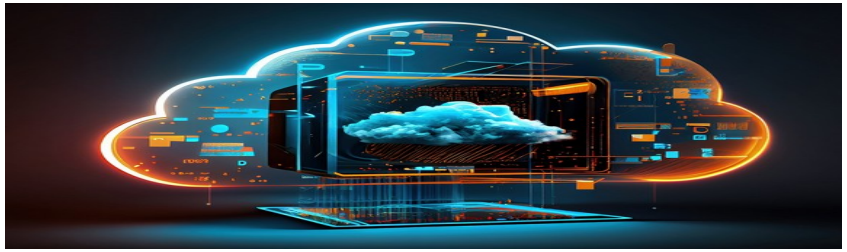
Menu-Based User Interface

The system employs a simple console-based menu system:

- The main menu allows users to choose between registering, logging in, or exiting the application.

- After logging in, a secondary menu presents options for managing products and the shopping cart.

- Input is taken through `scanf`, and actions are performed based on user choices.



Data Storage

The application uses text files for data persistence:

- **User Data**: Stored in `users.txt` in the format `username password`, allowing for straightforward reading and writing.

- **Product Data**: Stored in `products.txt` in a structured format (ID, name, price, stock), enabling easy updates and retrievals.

ADVANTAGES OF THIS PROJECT

- **Performance**
- **Efficient Operations**: Since it’s a console application written in C, it can handle operations quickly without the overhead of a graphical user interface or web server.
- **Low Resource Usage**: This project can run on systems with limited resources, making it accessible for a wider range of environments.
- **Foundation for Advanced Projects**
- **Gateway to More Complex Systems**: This project serves as a stepping stone to more complex e-commerce platforms, allowing you to gradually learn about databases, web frameworks, and user interfaces.
- **Integration Opportunities**: You can explore integrating with databases (using SQLite or files), creating a graphical interface, or transitioning to web technologies.

- **Understanding E-Commerce Concepts**
- **Business Logic**: Implementing a cart and checkout process gives you insight into the basic workflows of e-commerce systems.
- **User Interaction**: Managing user input and outputs helps you understand how customers interact with online stores.
- **Customizability**
- **Tailored Features**: You can customize the system to fit specific needs or interests, whether adding new products, payment options, or other functionalities.
- **Learning Different Aspects**: Experimenting with the code allows you to learn various programming aspects, from basic syntax to more advanced data handling.

CODING PROGRAME OF C SAMPLE

```
• #include <stdio.h>
• #include <string.h>
• typedef struct {
•  int id;
• char name[50];
•  float price;
• } Product;
• Product products[100];
• int productCount = 0;
• void addProduct()
• {
•  Product p;
• printf("Enter Product ID: ");
•  scanf("%d", &p.id);
•   printf("Enter Product Name: ");
•  scanf("%s", p.name);
• printf("Enter Price: ");
• scanf("%f", &p.price);
• products[productCount++] = p;
•  printf("Product added!\n");}
```

- void displayProducts() {
- printf("ID\tName\tPrice\n");
- for (int i = 0; i < productCount; i++)
- {
- printf("%d\t%s\t%.2f\n",
- products[i].id,
- products[i].name,
- products[i].price);
- }
- }void placeOrder()
- {
- int id;
- printf("Enter Product ID to Order: ");
- scanf("%d", &id);
- for (int i = 0; i < productCount; i++)

- {
- if (products[i].id == id)
- {
- printf("Order placed for %s at price %.2f\n", products[i].name, products[i].price);
- return;
- }
- printf("Product not found.\n");
- }
- int main()
- {
- int choice;
- while (1)
- {
-

- printf("1. Add Product\n2. Display Products\n3. Place Order\n4. Exit\nChoose an option: ");
- scanf("%d", &choice);

```
 if (choice == 1)
{
addProduct();
 }
else if (choice == 2)
{
displayProducts();
 }
 else if (choice == 3)
 {
placeOrder();
  }
 else
 {
break;
 }    }
return 0;}
```

CONCLUSION

- The simple e-commerce system developed in C provides a comprehensive educational experience, illustrating both fundamental programming principles and practical applications relevant to the e-commerce domain. Below are detailed reflections on the various aspects of the project:
- **1. Understanding Core Concepts**
- **Programming Fundamentals**: The project emphasizes key programming concepts, such as:
  - **Data Structures**: The use of structs to represent products and shopping carts showcases how to organize and manage related data efficiently.
  - **Control Flow**: The implementation of conditional statements and loops demonstrates how to direct program execution based on user input, enhancing interactive functionality.
  - **Functions**: By breaking down the code into reusable functions, learners can grasp the importance of modularity and code reusability, which are essential for maintaining larger codebases.