INVENTORY MANAGEMENT SYSTEM

A PROJECT REPORT

submitted by

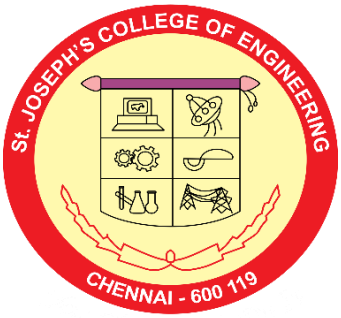
SANJAY SRIKUMAR S

(312323205197)

BACHELOR OF TECHNOLOGY

IN

DEPARTMENT OF INFORMATION TECHNOLOGY



ST. JOSEPH'S COLLEGE OF ENGINEERING

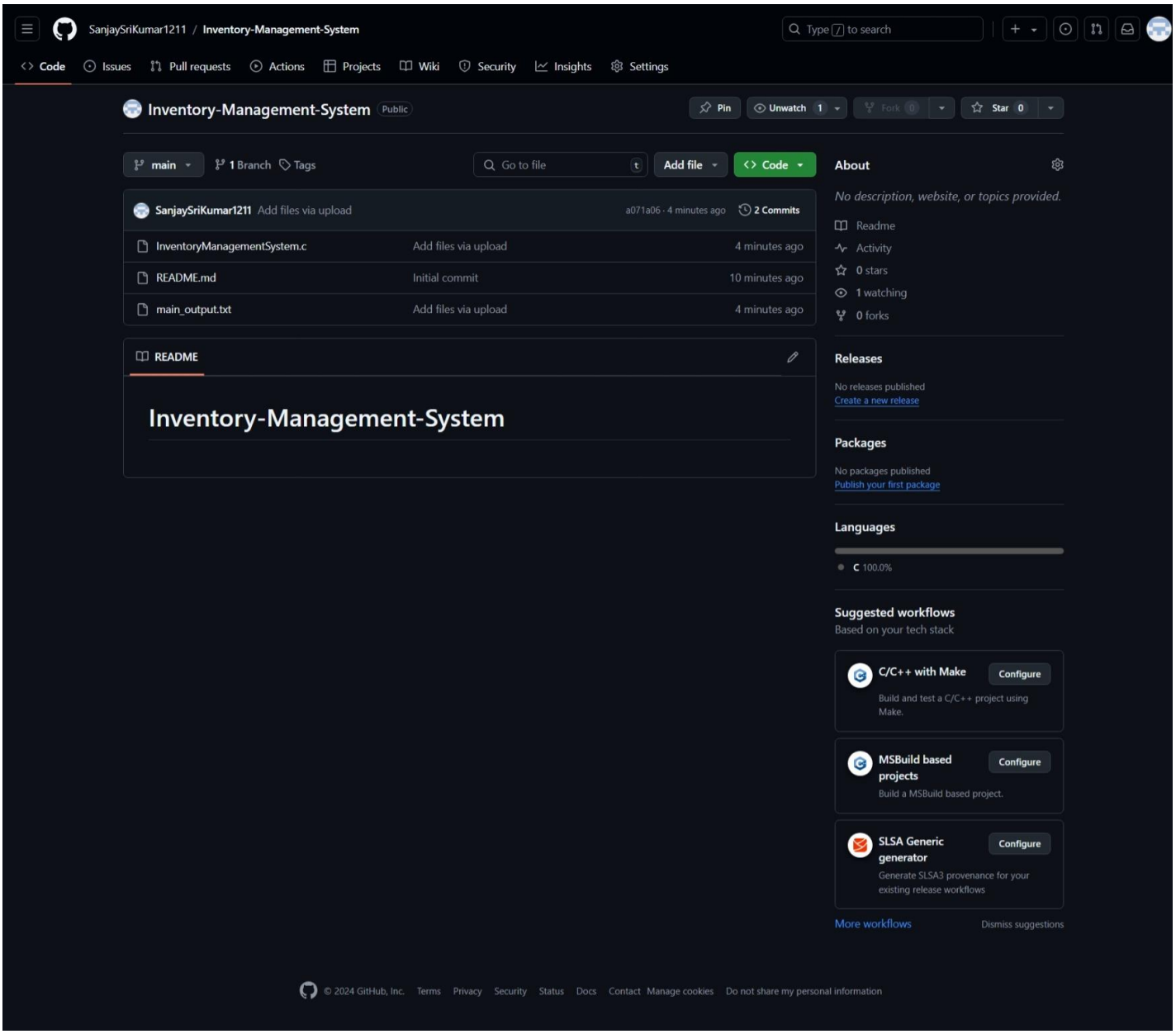
(An Autonomous Institution)

St. Joseph's Group of Institutions

OMR, Chennai- 60019

INVENTORY MANAGEMENT SYSTEM

GITHUB:



ABSTRACT:

The Inventory Management System (IMS) is a software application designed to help businesses manage their product inventory effectively. Developed in C programming language, this system allows users to perform various tasks such as adding new products, displaying the existing products, updating stock quantities, and removing products. It aims to automate and simplify inventory management, ensuring that businesses can maintain accurate records of product quantities and prices. With this system, users can keep track of products, avoid stock discrepancies, and improve decision-making related to product procurement and sales.

INTRODUCTION:

In modern businesses, managing inventory is one of the most critical aspects to ensure smooth operations. An efficient Inventory Management System (IMS) helps businesses maintain optimal stock levels, reduce waste, and prevent stockouts. This system, developed in C, addresses the need for effective inventory management by allowing users to manage their stock, including adding new products, displaying the inventory, updating product quantities, and removing obsolete products. The system provides an easy-to-use interface, ensuring that users can perform these tasks quickly and with minimal errors. Designed for small and medium-sized businesses, the IMS provides a solution to streamline inventory management tasks and support better decision-making.

OBJECTIVES:

The main objectives of the Inventory Management System (IMS) are:

1. **Efficient Stock Management**:
   Allow users to add, view, update, and remove products in the inventory efficiently, ensuring accurate records are maintained.
2. **Real-Time Inventory Tracking:**
   Provide real-time updates on product stock levels, ensuring businesses have an up-to-date view of their inventory at all times.
3. **Improved Decision-Making:**
   Enable businesses to make informed decisions based on accurate data about stock quantities and product details.
4. **Simplify Inventory Operations:**
   Automate repetitive tasks such as updating product stock and removing products, saving time and reducing the risk of human error.
5. **Enhance Record Keeping:**
   Maintain detailed records of each product, including its ID, name, quantity, and price, making it easier for businesses to track their inventory history.
6. **User-Friendly Interface:**
   Provide an easy-to-use menu-driven interface that makes it simple for users to navigate and perform necessary inventory tasks.

SCOPE OF THE PROJECT:

1. Inventory Management:
   The system allows businesses to manage their products easily by adding new products, updating stock levels, and removing outdated products.
2. Tracking Product Details:
   The system stores essential details of each product, such as product ID, name, quantity, and price. It keeps this information updated in real time.
3. Basic Operations:
   Users can perform simple operations like adding products, updating their quantities, removing unwanted products, and displaying the current product list.

4. Simple User Interface:

The system uses a menu-driven interface that is easy to understand, allowing users to perform tasks without confusion.

5. Future Enhancements:

The system can be expanded to include features like product search, report generation, and multi-user support to make it more useful for larger businesses in the future.

TECHNOLOGIES USED:

C Programming Language:

The entire system is developed using the C programming language, which is known for its efficiency, speed, and wide usage in system-level applications. It allows for easy memory management and faster processing of inventory-related tasks.

1.Data Structures:

The system uses structures in C to store the details of each product, such as product ID, name, quantity, and price. This provides a structured and organized way to manage product information.

1. **File Handling (Optional Enhancement):**

   Although not implemented in the basic version, file handling can be used to save inventory data to a file. This would allow for persistent storage, enabling the system to load the data whenever it is run, instead of losing all information after exiting the program.

2.Basic Algorithms:

Basic algorithms are used for managing inventory operations like searching for products, updating stock quantities, and removing products, making the system functional and efficient.

Description of the Project:

The Inventory Management System is a software application designed to manage and track products in an inventory. The system allows users to add new

products, update product stock levels, remove products, and view the list of products currently available in the inventory. It is built using the C programming language, offering a simple and efficient solution for small businesses or personal use.

The main objective of the system is to automate the process of managing products, reducing manual work and errors. It ensures that businesses can keep track of product details, such as product ID, name, quantity, and price, while providing an easy way to update and maintain accurate records.

Key features of the system include:

- Adding Products: Users can input new product details (ID, name, quantity, and price).
- Updating Stock: Users can update the quantity of products in the inventory whenever needed.
- Removing Products: Users can delete products that are no longer in stock or needed.
- Displaying Products: Users can view all the products available in the inventory along with their details.

The system is designed with a simple menu-driven interface, making it easy to navigate and use. It provides basic inventory management capabilities but can be extended with additional features like searching products, generating reports, or integrating with a database for larger-scale use.

Overall, this Inventory Management System streamlines the management of products and provides a user-friendly tool to help businesses keep their inventory up to date and organized.

INVENTORY MANAGEMENT SYSTEM CODE:

```c
#include <stdio.h>
#include <string.h>

#define MAX_PRODUCTS 50

// Structure to store product details
typedef struct {
    int productID;
    char productName[50];
    int quantity;
    float price;
} Product;

// Function prototypes
void addProduct(Product inventory[], int *numProducts);
void displayProducts(Product inventory[], int numProducts);
void updateStock(Product inventory[], int numProducts);
void removeProduct(Product inventory[], int *numProducts);

int main() {
    Product inventory[MAX_PRODUCTS];
    int numProducts = 0; // Keeps track of the number of products in the inventory
    int choice;

    do {
        // Display menu options
        printf("\nInventory Management System\n");
        printf("1. Add Product\n");
        printf("2. Display Products\n");
        printf("3. Update Stock\n");
        printf("4. Remove Product\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");

        scanf("%d", &choice);
        getchar();  // Consume newline character left by scanf

        switch (choice) {
            case 1:
                addProduct(inventory, &numProducts);
                break;
            case 2:
                displayProducts(inventory, numProducts);
                break;
            case 3:
                updateStock(inventory, numProducts);
                break;
            case 4:
                removeProduct(inventory, &numProducts);
                break;
            case 5:
                printf("Exiting the system...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 5);

    return 0;
}

// Function to add a new product to the inventory
void addProduct(Product inventory[], int *numProducts) {
    if (*numProducts < MAX_PRODUCTS) {
        Product newProduct;
        printf("Enter product ID: ");
        scanf("%d", &newProduct.productID);
```

```
        getchar();  // Consume newline character left by scanf

        printf("Enter product name: ");
        fgets(newProduct.productName, sizeof(newProduct.productName), stdin);
        newProduct.productName[strcspn(newProduct.productName, "\n")] = '\0'; // Remove
newline

        printf("Enter product quantity: ");
        scanf("%d", &newProduct.quantity);

        printf("Enter product price: ");
        scanf("%f", &newProduct.price);

        inventory[*numProducts] = newProduct;
        (*numProducts)++;
        printf("Product added successfully!\n");
    } else {
        printf("Inventory is full! Cannot add more products.\n");
    }
}

// Function to display all products in the inventory
void displayProducts(Product inventory[], int numProducts) {
    if (numProducts == 0) {
        printf("No products in the inventory.\n");
    } else {
        printf("\nProduct Inventory:\n");
        printf("ID\tName\t\tQuantity\tPrice\n");
        for (int i = 0; i < numProducts; i++) {
            printf("%d\t%-20s\t%d\t\t%.2f\n", inventory[i].productID,
inventory[i].productName, inventory[i].quantity, inventory[i].price);
        }
    }
```

```
}

// Function to update the stock of an existing product
void updateStock(Product inventory[], int numProducts) {
    int productID, newQuantity;
    printf("Enter product ID to update: ");
    scanf("%d", &productID);
    getchar();  // Consume newline character left by scanf

    int found = 0;
    for (int i = 0; i < numProducts; i++) {
        if (inventory[i].productID == productID) {
            found = 1;
            printf("Enter new quantity for %s: ", inventory[i].productName);
            scanf("%d", &newQuantity);
            inventory[i].quantity = newQuantity;
            printf("Stock updated successfully!\n");
            break;
        }
    }

    if (!found) {
        printf("Product ID not found.\n");
    }
}

// Function to remove a product from the inventory
void removeProduct(Product inventory[], int *numProducts) {
    int productID;
    printf("Enter product ID to remove: ");
    scanf("%d", &productID);
    getchar();  // Consume newline character left by scanf
```

```
    }

    if (!found) {
        printf("Product ID not found.\n");
    }
}

// Function to remove a product from the inventory
void removeProduct(Product inventory[], int *numProducts) {
    int productID;
    printf("Enter product ID to remove: ");
    scanf("%d", &productID);
    getchar();  // Consume newline character left by scanf

    int found = 0;
    for (int i = 0; i < *numProducts; i++) {
        if (inventory[i].productID == productID) {
            found = 1;
            // Shift products to fill the removed product's position
            for (int j = i; j < *numProducts - 1; j++) {
                inventory[j] = inventory[j + 1];
            }
            (*numProducts)--;
            printf("Product removed successfully!\n");
            break;
        }
    }

    if (!found) {
        printf("Product ID not found.\n");
    }
}
```

CODE EXPLANATION:

main():

- This is the main part of the program that shows a menu and asks the user what they want to do.
- It keeps asking the user for a choice until they choose to exit.
- Depending on the choice, it calls different functions to:
    - Add a product,
    - Show the products,
    - Update a product's stock,
    - Remove a product,
    - Exit.

addProduct():

1. Asks the user to enter product details: ID, name, quantity, and price.
2. Checks if the inputs for quantity and price are correct.
3. If there is space in the inventory, it adds the product to the list.
4. If the inventory is full, it tells the user that no more products can be added.

displayProducts():

1. Shows the list of all products in the inventory.
2. If there are no products, it tells the user the inventory is empty.

updateStock():

1. Asks for the **product ID** of the product to update.
2. Finds the product and asks for a **new quantity**.
3. Updates the quantity of that product.
4. If the product is not found, it tells the user that the product doesn't exist.

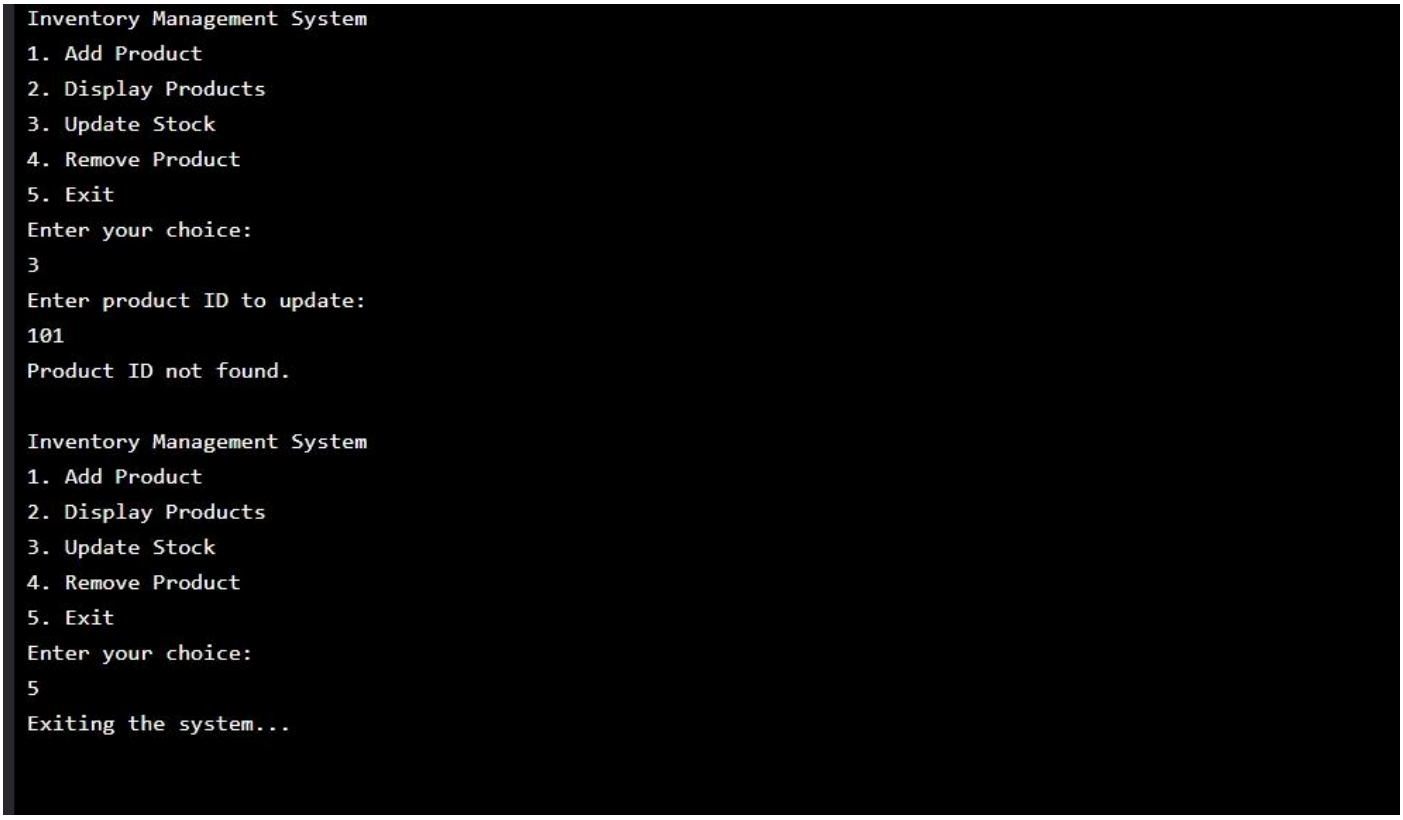
removeProduct():

1. Asks for the **product ID** of the product to remove.
2. Finds the product and removes it from the list.
3. If the product is not found, it tells the user that the product doesn't exist.

OUTPUT:

```
Inventory Management System
1. Add Product
2. Display Products
3. Update Stock
4. Remove Product
5. Exit
Enter your choice:
1
Enter product ID:
101
Enter product name:
HP laptop
Enter product quantity:
20
Enter product price:
2450000
Product added successfully!

Inventory Management System
1. Add Product
2. Display Products
3. Update Stock
4. Remove Product
5. Exit
Enter your choice:
2

Product Inventory:
ID      Name            Quantity        Price
101     HP laptop               20              2450000.00
```

```
Inventory Management System
1. Add Product
2. Display Products
3. Update Stock
4. Remove Product
5. Exit
Enter your choice:
3
Enter product ID to update:
101
Enter new quantity for HP laptop:
20
Stock updated successfully!

Inventory Management System
1. Add Product
2. Display Products
3. Update Stock
4. Remove Product
5. Exit
Enter your choice:
4
Enter product ID to remove:
101
Product removed successfully!

Inventory Management System
1. Add Product
2. Display Products
3. Update Stock
4. Remove Product
5. Exit
```



CONCLUSION:

The Inventory Management System provides an efficient way to manage products in an inventory, offering functionalities such as adding, displaying, updating, and removing products. It allows users to easily interact with the system through a simple menu-driven interface. The program ensures that the inventory is well-organized and allows for real-time updates to product quantities and prices.

Key features include:

- Adding new products with details like ID, name, quantity, and price.
- Displaying all products in the inventory.
- Updating the stock of existing products based on user input.
- Removing products when no longer needed.
- Input validation to ensure correct data is entered.

Overall, this system simplifies inventory management by making the process more automated and reducing the chances of human error, which can be especially useful for small businesses or organizations looking to streamline their product management.

Future Enhancements:

To make the Inventory Management System more robust and adaptable, several enhancements can be implemented in the future:

1. **Database Integration**:
    - Replace the static array with a database (e.g., MySQL, SQLite) to allow for scalable data storage and easier management of a larger inventory.
    - This would also enable data persistence across program restarts.
2. **User Authentication**:
    - Implement a login system with different user roles (e.g., Admin, Employee) to restrict access and provide security.
    - Admins could have permissions to add, update, and remove products, while regular employees might only view or update stock.
3. **Real-Time Inventory Updates**:
    - Introduce real-time tracking of inventory, which updates automatically as products are added, updated, or removed.
    - Integration with barcode scanners or RFID could automate this process even further.
4. **Search and Filter Options**:
    - Add advanced search and filtering options to allow users to find products by ID, name, category, or price range.
    - This feature would make it easier to locate specific products in larger inventories.
5. **Data Analytics and Reporting**:
    - Include analytical features that generate reports on stock levels, low-inventory alerts, and trends in product movement.
    - Sales forecasting and demand analysis can help businesses make better stocking decisions.
6. **Supplier and Order Management**:

- Introduce a module to manage suppliers and automate purchase orders when inventory levels reach a defined threshold.
- This would streamline the supply chain and reduce the risk of stockouts.

7. **Mobile App or Web Interface**:
   - Develop a mobile app or web interface for remote access to the inventory system, allowing users to check inventory and make updates from anywhere.
   - This would be particularly useful for larger teams or warehouse managers who need flexibility.
8. **Enhanced User Interface (UI)**:
   - Upgrade the console-based interface to a graphical user interface (GUI) using tools like Tkinter for Python or Qt for C++.
   - This would make the system more user-friendly, especially for non-technical users.
9. **Data Backup and Recovery**:
   - Implement a backup and recovery system to prevent data loss in case of hardware failure or accidental deletion.
   - Automated backups could be scheduled daily, weekly, or monthly, depending on the business needs.
10. **Inventory Forecasting Using Machine Learning**:
    - Utilize machine learning algorithms to analyze past data and forecast future inventory needs.
    - This can help businesses anticipate demand, optimize stock levels, and reduce storage costs.

References for Inventory Management System Project:

1. **Books on C Programming**:
   - *The C Programming Language* by B.W. Kernighan & D.M. Ritchie, 2nd Edition – Fundamental C programming principles.

- *C Primer Plus* by Stephen Prata, 5th Edition – Covers structures, file handling, and C programming essentials.

2. **Inventory Management Concepts**:
   - *Best Practice in Inventory Management* by Tony Wild – Practical insights on managing inventory effectively.
   - "How Many Parts to Make at Once" by F.W. Harris – Economic Order Quantity model for optimized inventory control.
3. **File Handling in C**:
   - **GeeksforGeeks**: “File Handling in C” – Detailed examples on reading, writing, and appending data.
   - **Programiz**: “C File Input/Output” – Explains file handling for data storage in C applications.
4. **Structures and Data Management**:
   - **Data Structures and Algorithm Analysis in C** by Mark Allen Weiss – Essential concepts on structuring data in C programs.
   - **GeeksforGeeks**: “Structures in C” – Practical guides for using struct in C for storing inventory records.
5. **User Interface and CLI Development**:
   - *The Art of Unix Programming* by Eric S. Raymond – General principles for designing user-friendly console interfaces.
   - **IBM Developer**: "Writing CLI applications in C" – Best practices for creating effective command-line applications.
6. **Online Tutorials and Coding Communities**:
   - *YouTube Channels*: *CodeWithHarry*, *ProgrammingKnowledge* – Step-by-step C programming tutorials for beginners.
   - **Stack Overflow** – Community support for troubleshooting C code issues, especially for file handling and struct operations.