INVENTORY MANAGEMENT SYSTEM

MINOR PROJECT REPORT

By

SUDIPTA SUNDAR PAL (RA22110260103339), KUSHAGRA PUROHIT(RA2211026010391) DIBYAJYOTI BHATTACHARJEE(RA2211026010340)

Under the guidance of

Dr. Maheshwari

In partial fulfilment for the Course

of

21CSC201J DATA STRUCTURES AND ALGORITHMS

in Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC201J DATA STRUCTURES AND ALGORITHMS** entitled in "**INVENTORY MANAGEMENT SYSTEM**" is the bonafide work of SUDIPTA SUNDAR PAL(RA2211026010339), KUSHAGRA PUROHIT(RA2211026010391), DIBYOJYOTI BHATTACHARJEE(RA2211026010340) who carried out the work under my supervision.

SIGNATURE

Dr. Maheshwari

DSA– Course Faculty

Faculty Advisor

Department of Computational Intelligence

SRM Institute of Science and Technology

Kattankulathur

SIGNATURE

Dr Annie Uthra R

Head of the Department

Professor

Department of Computational Intelligence

SRM Institute of Science and Technology

Kattankulathur

ABSTRACT

The inventory management system presented is a C program that utilizes a linked list data structure to efficiently manage product information. Each product is represented by a node in the linked list, containing details such as ID, name, price, and quantity. The system offers functionalities including adding, removing, editing, listing products, and saving data to a file. Dynamic memory allocation facilitates a scalable and flexible approach to inventory management. The linked list structure allows for the dynamic manipulation of products, providing an adaptable solution for businesses to handle varying inventory sizes. The program demonstrates a menu-driven interface, enabling users to interact with and manage the inventory seamlessly. This inventory management system enhances operational efficiency by employing a linked list, allowing for dynamic product updates without the constraints of a fixed-size array. The program ensures data integrity by saving and loading inventory details from a file. With features such as product addition, removal, and modification, it offers a user-friendly interface for effective control and organization of inventory data.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. C.Malathy, Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. Maheshwari, DSA– Course Faculty, Department of Computational Intelligence** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr Annie Uthra R , Head of the Department, Department of Computational Intelligence** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

INTRODUCTION

We are developing an Inventory Management System in C that allows users to manage and maintain a list of products. The system should provide the following functionality: adding a new product to the inventory, removing a product, editing product details, listing all products in the inventory, and quitting the application. The inventory data should be saved and loaded from a file for persistent storage.
Our goal is to create a C program that enables users to manage an inventory of products. The program should provide the following features:

1. Add Product: Users can add a new product to the inventory. They need to enter the product's ID, name, price, and quantity. The product's ID must be unique.

2. Remove Product: Users can remove a product from the inventory by specifying its ID. If the product is found, it will be removed from the list.

3. Edit Product: Users can modify the details of an existing product. They can change the name, price, and quantity of a product by specifying its ID.

4. List Products: Users can view a list of all products in the inventory. The list should include the product's ID, name, price, and quantity.

5. Quit: Users can exit the application. Before quitting, the program should save the current inventory data to a file for future use.

MOTIVATION

Motivation for Developing the Inventory Management System:

1. Operational Efficiency Improvement:
   - The existing manual inventory management system at Company has proven to be time-consuming and error-prone. Automating the process through a digital solution aims to enhance operational efficiency by reducing manual workload and minimizing the chances of human error.

2. Real-Time Visibility:
   - The lack of real-time visibility into stock levels and product data poses challenges in decision-making. The motivation for this project is to implement a system that provides instant access to accurate and up-to-date information, allowing for better-informed decisions regarding stock levels, procurement, and order fulfillment.

3. Error Reduction and Data Accuracy:
   - Manual data entry is prone to errors, leading to inaccuracies in inventory records. The Inventory Management System is motivated by the need to eliminate errors, improve data accuracy, and maintain a reliable database of product information, ensuring a more trustworthy foundation for business decisions.

4. Cost Savings and Resource Optimization:
   - Inefficient inventory management practices often result in unnecessary costs associated with overstocking, stockouts, and suboptimal resource allocation. The digital solution aims to optimize inventory levels, minimize holding costs, and ultimately contribute to cost savings for Company.

5. Streamlined Order Processing:
   - The motivation behind the project is to streamline the order processing workflow. Automation of tasks such as order creation, fulfillment, and tracking contributes to faster order processing, improved customer satisfaction, and a more efficient supply chain.

6. Adaptability to Business Growth:
   - As company continues to grow, the manual inventory management system may become increasingly inadequate. The motivation is to implement a scalable and

adaptable solution that can accommodate the evolving needs of the business, supporting its growth trajectory.

7. Compliance and Reporting:
- Regulatory compliance and accurate reporting are essential aspects of business operations. The Inventory Management System is motivated by the necessity to establish a system that ensures adherence to industry regulations, facilitates audit processes, and generates precise reports for financial and operational analysis.

8. Enhanced Customer Satisfaction:
- Timely and accurate order fulfillment is crucial for customer satisfaction. By addressing the inefficiencies in the current system, the project aims to enhance the overall customer experience by ensuring product availability, minimizing delays, and meeting customer demands more effectively.

9. Technological Advancements:
- The rapid advancement of technology offers new and innovative ways to manage inventory. The motivation for the project is to leverage modern technologies and data structures, such as linked lists, to create a more robust and dynamic Inventory Management System that aligns with current industry standards.

10. Competitive Edge:
- A well-implemented Inventory Management System can provide Company with a competitive edge in the market. The motivation is to establish a sophisticated and efficient system that not only meets current needs but also positions the company for success in a competitive business environment

OBJECTIVES

Objectives of the Inventory Management System:

1. Automated Inventory Tracking:
   - Develop an automated system that tracks inventory levels in real-time, providing an accurate and up-to-date overview of product availability.

2. Error Reduction and Data Accuracy:
   - Minimize errors associated with manual data entry by implementing a digital solution, ensuring the accuracy of inventory records.

3. Efficient Order Processing:
   - Streamline the order processing workflow to enhance efficiency, reduce fulfillment times, and improve overall customer satisfaction.

4. Optimized Stock Levels:
   - Implement algorithms and strategies to optimize stock levels, preventing both overstock and stockout situations and minimizing holding costs.

5. Enhanced Decision-Making:
   - Provide decision-makers with comprehensive data analytics and reporting tools to facilitate better-informed decisions related to procurement, production, and inventory management.

6. Streamlined Product Additions and Removals:
   - Simplify the processes of adding new products to the inventory and removing obsolete or discontinued products, ensuring flexibility and adaptability.

7. Dynamic Product Editing:
   - Enable the dynamic editing of product details, such as name, price, and quantity, allowing for adjustments to match evolving business requirements.

8. Centralized Vendor Management:
   - Facilitate centralized management of vendor relationships, optimizing the procurement process, and ensuring timely and cost-effective replenishment of inventory.

9. Scalability:
   - Design the system to be scalable, accommodating the growth of [Company/Organization Name] and its evolving inventory needs.

10. Regulatory Compliance:
    - Ensure compliance with industry regulations and standards related to inventory management, providing a reliable and auditable system.

11. User-Friendly Interface:
    - Develop a user-friendly interface that allows employees to interact seamlessly with the Inventory Management System, minimizing the learning curve and enhancing user adoption.

12. Data Security and Integrity:
- Implement robust security measures to safeguard sensitive inventory data and ensure the integrity of the system.

13. Cost Savings:
- Optimize inventory management to contribute to cost savings by minimizing holding costs, preventing overstock-related expenses, and reducing the risk of stockouts.

14. Adaptability to Technological Advances:
- Design the system to leverage current technological advancements and ensure its compatibility with future innovations in inventory management.

15. Customer Satisfaction:
- Improve customer satisfaction by ensuring product availability, reducing order processing times, and enhancing overall reliability in fulfilling customer demands.

By achieving these objectives, the Inventory Management System aims to transform Company inventory processes, fostering efficiency, accuracy, and adaptability in the dynamic business landscape.

PROBLEM STATEMENT

In the current business environment, Company faces significant challenges in effectively managing its inventory. The prevailing inventory tracking system heavily relies on manual processes, resulting in inefficiencies, inaccuracies, and a lack of real-time visibility into stock levels. This outdated approach consumes valuable time and resources, exposing the organization to the risks associated with human error, leading to overstocking, stockouts, and compromised decision-making.

A central issue with the existing system is its inability to adapt to the evolving needs of the business. As Company experiences growth, the limitations of the manual inventory management approach become increasingly apparent. The lack of scalability and flexibility impedes the organization's ability to effectively handle a growing product catalog, diverse inventory requirements, and fluctuations in customer demand.

Moreover, the absence of a centralized and automated solution hinders the company's capacity to maintain a cohesive and accurate database of product information. This results in challenges related to order processing, vendor management, and overall supply chain efficiency. Manual data entry processes are prone to errors, leading to inconsistencies in inventory records and hindering the organization's ability to make informed, data-driven decisions.

The current system also lacks mechanisms for dynamic product management. Adding new products to the inventory or removing obsolete items requires cumbersome manual interventions, contributing to delays and potential inaccuracies in the product database. As a consequence, Company faces difficulties in keeping pace with market demands, responding effectively to product lifecycle changes, and ensuring a seamless customer experience.

To address these multifaceted challenges and propel Company towards operational excellence, there is a compelling need for the implementation of a robust and comprehensive Inventory Management System. This system should leverage modern technologies, including a dynamic data structure such as linked lists, to automate and optimize the entire inventory management process.

The primary objectives of the proposed Inventory Management System are to automate inventory tracking, reduce errors associated with manual data entry, streamline order processing, optimize stock levels, and enhance overall decision-making. The system must provide a user-friendly interface to ensure seamless interaction for employees at all levels, from warehouse personnel to management.

Additionally, the Inventory Management System should be scalable, adapting to the growth trajectory of Company, and should prioritize data security and integrity to protect sensitive inventory information. The implementation of reporting and analytics tools is essential to facilitate data-driven decision-making, while compliance with industry regulations ensures that the system operates within established standards.

By addressing these challenges and meeting the outlined objectives, the proposed Inventory Management System aims to transform Company's inventory processes into a dynamic, efficient, and adaptable framework. This transformation is not merely a technological upgrade but a strategic imperative to ensure Company remains competitive, responsive to market demands, and well-positioned for sustained growth in the ever-evolving business landscape.

CHALLENGES

Implementing an effective Inventory Management System comes with various challenges that organizations often encounter. These challenges can impact operational efficiency, decision-making processes, and overall business performance. Here are some common challenges faced in Inventory Management Systems:

1. Inaccurate Demand Forecasting:
   - Forecasting demand accurately is challenging, and inaccuracies can lead to overstocking or stockouts. Fluctuations in market demand, seasonality, and unforeseen events can contribute to forecasting challenges.

2. Manual Data Entry Errors:
   - Relying on manual data entry increases the risk of errors in inventory records. Inaccuracies in product quantities, prices, or other details can lead to misguided decision-making.

3. Lack of Real-Time Visibility:
   - Traditional inventory systems may lack real-time visibility into stock levels, making it difficult for businesses to respond promptly to changes in demand or supply chain disruptions.

4. Supply Chain Complexity:
   - Global and intricate supply chains introduce complexities in managing inventory. Delays, disruptions, or changes in supplier relationships can impact the availability of products.

5. Overstocking and Stockouts:
   - Balancing inventory levels to avoid both overstocking and stockouts is a perpetual challenge. Overstocking ties up capital and warehouse space, while stockouts can lead to lost sales and customer dissatisfaction.

6. Technology Integration:
   - Integrating inventory management systems with other business systems, such as accounting or sales platforms, can be challenging. Incompatibility issues may hinder seamless data flow across departments.

7. Dynamic Product Lifecycle:
   - Managing products through various stages of their lifecycle, from introduction to obsolescence, requires a flexible system. Adapting to changes in product lines and discontinuing or introducing new items can be complex.

8. Vendor Management:
   - Coordinating with multiple vendors and managing relationships effectively is crucial. Issues such as delayed deliveries, quality concerns, or changes in supplier terms can impact inventory management.

9. Security Concerns:
   - Protecting sensitive inventory data from unauthorized access or cyber threats is a constant challenge. Ensuring data security and integrity is essential for maintaining trust in the system.

10. User Adoption and Training:
    - Introducing a new inventory management system requires training employees to use the system effectively. Resistance to change and a lack of understanding can hinder successful implementation.

11. Scalability:
- As businesses grow, the inventory management system must scale accordingly. Ensuring that the system can handle increased data volume, transaction loads, and user demands is a challenge.

12. Regulatory Compliance:
- Compliance with industry regulations, tax codes, and reporting standards adds complexity to inventory management. Ensuring that the system aligns with legal requirements is essential.

13. Technological Obsolescence:
- Rapid technological advancements may lead to the obsolescence of existing inventory management systems. Regular updates and investments in technology are necessary to stay current.

14. Cost Management:
- Balancing the costs associated with holding inventory, including storage, insurance, and depreciation, while ensuring optimal stock levels, requires careful financial management.

15. Cultural Shift:
- Encouraging a cultural shift toward data-driven decision-making and efficient inventory practices may face resistance within the organization. Convincing stakeholders of the benefits of the new system is crucial.

Addressing these challenges requires a strategic approach, a commitment to ongoing improvement, and the adoption of modern technologies to create a robust and adaptive Inventory Management System.

LITERATURE SURVEY

The provided code implements a basic inventory management system using a linked list data structure for dynamic product storage. The literature survey with respect to this code can be categorized into several relevant areas:

1. Dynamic Data Structures in Inventory Management:
   - Existing literature explores the use of dynamic data structures like linked lists in inventory management systems. Studies may discuss the advantages of dynamic structures for flexible and efficient memory allocation, especially in scenarios where the number of products can vary dynamically.

2. File Handling in Inventory Systems:
   - Research may delve into best practices for file handling in inventory systems. This involves storing and retrieving data from external files, as seen in the code. Literature might discuss different file formats, serialization methods, and the impact of file handling on system performance.

3. User Interface and Experience in Inventory Management:
   - The code emphasizes a simple console-based user interface. Related literature could explore user interface design principles for inventory management systems, considering usability, user experience, and the impact on user adoption.

4. Inventory Management Best Practices:
   - The literature may discuss best practices in inventory management, including product identification, avoiding duplicates, and handling product updates. This is relevant to the code's functions like adding, removing, and editing products.

5. Data Persistence and Recovery:
   - The code saves and loads inventory data from an external file. Literature may address techniques for ensuring data persistence, preventing data loss, and recovering data in case of system failures.

6. Security Considerations in Inventory Systems:
   - The code lacks explicit security measures. Corresponding literature may explore security challenges in inventory systems, including data encryption, access control, and protection against unauthorized modifications.

7. Scalability and Performance Optimization:
   - Research may discuss strategies for optimizing the performance of inventory systems, especially in scenarios where the product count grows significantly. This could include discussions on algorithmic efficiency and memory management.

8. Real-Time Inventory Management:
   - The code doesn't provide real-time inventory updates. Relevant literature might explore technologies and approaches to achieving real-time inventory tracking, integrating with sensors, or leveraging IoT for continuous monitoring.

9. Integration with Business Processes:
   - Literature may delve into the integration of inventory management systems with broader business processes, such as order processing, supply chain management, and accounting systems.

10. Adoption of Inventory Management Systems in Small Businesses:
    - Research might explore the adoption and challenges of implementing inventory management systems, particularly in small businesses. This could cover cost considerations, ease of implementation, and scalability.

11. Comparison with Alternative Inventory Management Approaches:
    - The literature may compare the linked list approach used in the code with alternative data structures or database-driven inventory management systems. This could include discussions on trade-offs, performance implications, and suitability for different scenarios.

A comprehensive literature survey in these areas would provide valuable insights and guidance for further refining and expanding the functionality of the inventory management system implemented in the code.

REQUIREMENTS

1.1 Hardware and Software Requirement

Hardware: A computer with a 2 GHz processor and 4 GB of RAM.

Software: Windows 10 operating system and C programming language.

Data: A text file in which we can see the list of products and whatever changes we make in the code will reflected back in text file.

ARCHITECTURE AND DESIGN

Architecture and Design Overview: Inventory Management System

1. System Architecture:
   - The system adopts a modular architecture with distinct components for data storage, business logic, and user interface.
   - Implements a linked list structure for dynamic product storage, allowing flexibility in managing varying product counts.

2. Data Structure:
   - Utilizes a linked list of `Product` structures for efficient and dynamic storage.
   - Each `Product` node contains information such as ID, name, price, quantity, and a pointer to the next node.

3. Data Persistence:
   - Inventory data is stored in an external file ("inventory.txt") for persistence between system sessions.
   - The `loadInventory` function reads data from the file into the linked list during system startup.
   - The `saveInventory` function writes the linked list data back to the file to ensure data persistence.

4. User Interface:
   - The user interface is text-based and presented through the console.
   - Users interact with the system by choosing options from a menu (add, remove, edit, list products, quit).

5. Functionality:
   - Add Product:

- Dynamically allocates memory for a new `Product` node.
- Validates the uniqueness of the product ID.
- Accepts user input for ID, name, price, and quantity.
- Adds the new product to the linked list and updates the inventory file.

- Remove Product:
  - Accepts user input for the product ID to be removed.
  - Locates the product in the linked list, removes it, and updates the inventory file.

- Edit Product:
  - Accepts user input for the product ID to be edited.
  - Locates the product, allows users to update its name, price, and quantity.
  - Updates the linked list and the inventory file.

- List Products:
  - Displays a list of all products in the inventory, including ID, name, price, and quantity.

- Quit:
  - Saves the current inventory state to the file before exiting the system.

6. Memory Management:
   - Allocates and deallocates memory dynamically for `Product` nodes.
   - Ensures proper freeing of memory when products are removed or when the system exits.

7. File Handling:
   - Uses file I/O functions to read from and write to the external inventory file.
   - Ensures robust error handling for file operations.

8. User Input Handling:
   - Utilizes standard input functions (`scanf`, `fgets`) to receive user inputs, ensuring buffer overflow protection and handling newline characters.

9. Error Handling:

- Incorporates error messages and prompts for invalid inputs or file-related errors.
- Gracefully handles scenarios where the inventory file is not present during the system startup.

10. Scalability:

- The linked list structure provides flexibility for handling a variable number of products.
- However, for scalability to a larger system, considerations for performance optimization and alternative data structures may be explored.

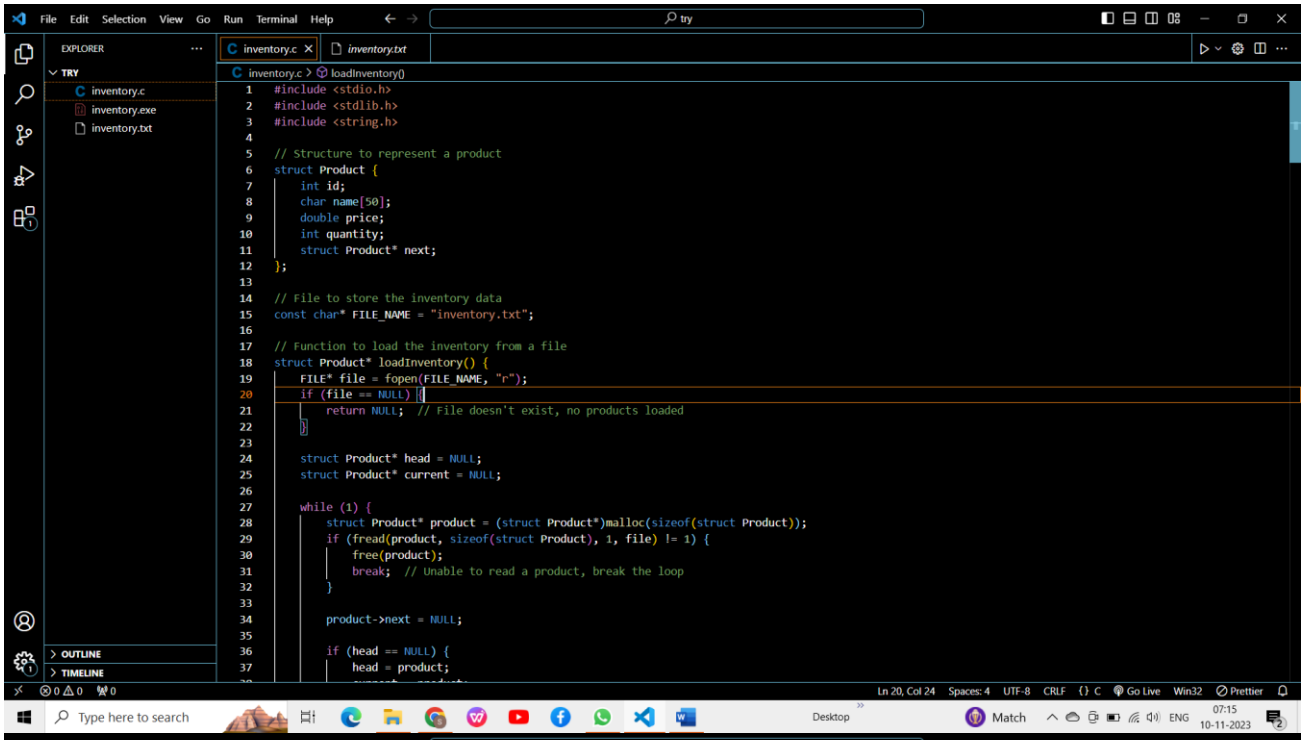
11. Extensibility:

- The modular design allows for easy extension with additional functionalities or improvements.
- Future enhancements could include more sophisticated input validation, security measures, or integration with external systems.

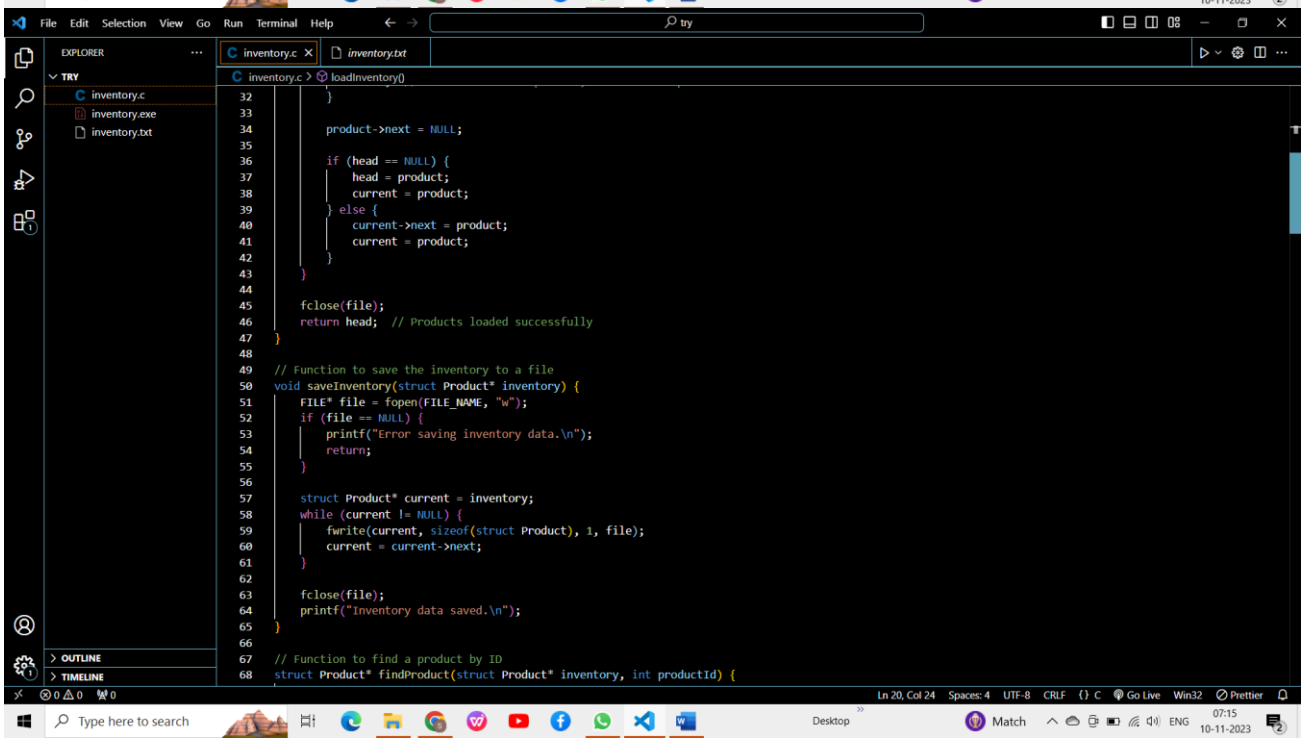
12. Usability:

- The text-based interface prioritizes simplicity for ease of use.
- Consideration for a more user-friendly graphical interface or integration with other platforms could enhance usability.

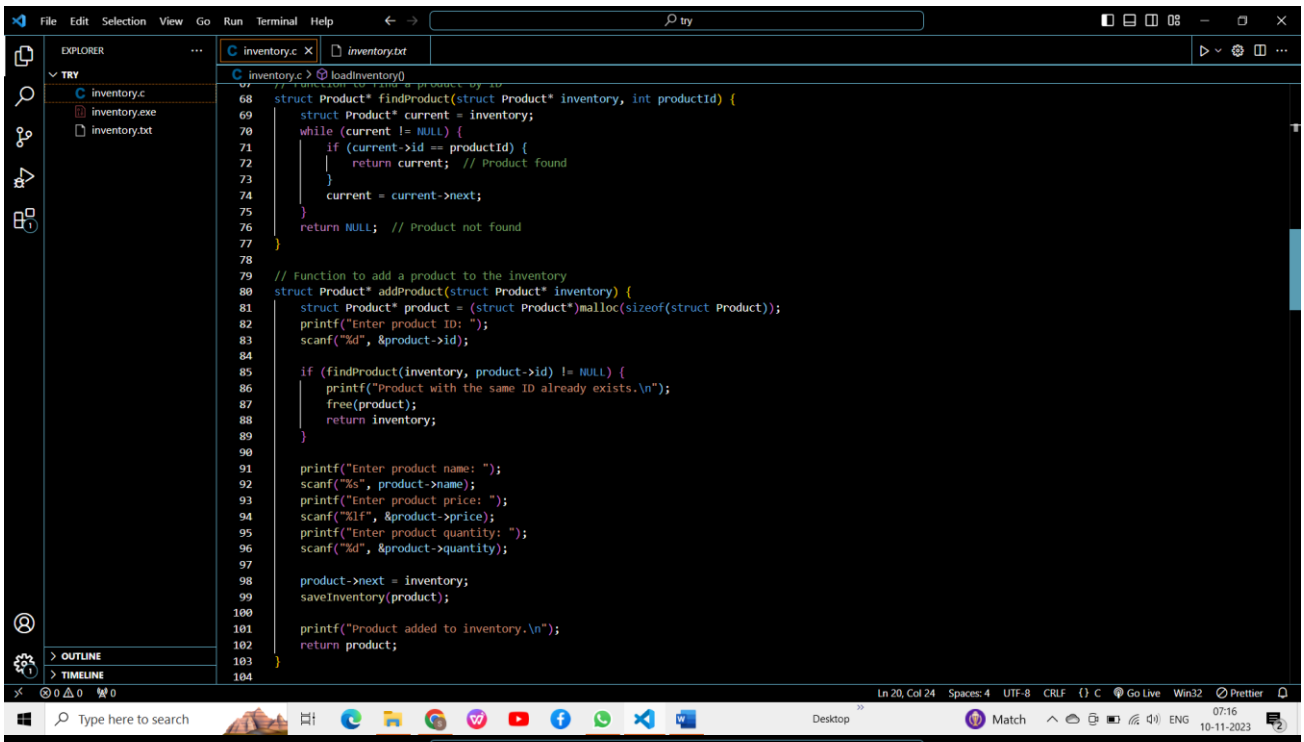
IMPLEMENTATION



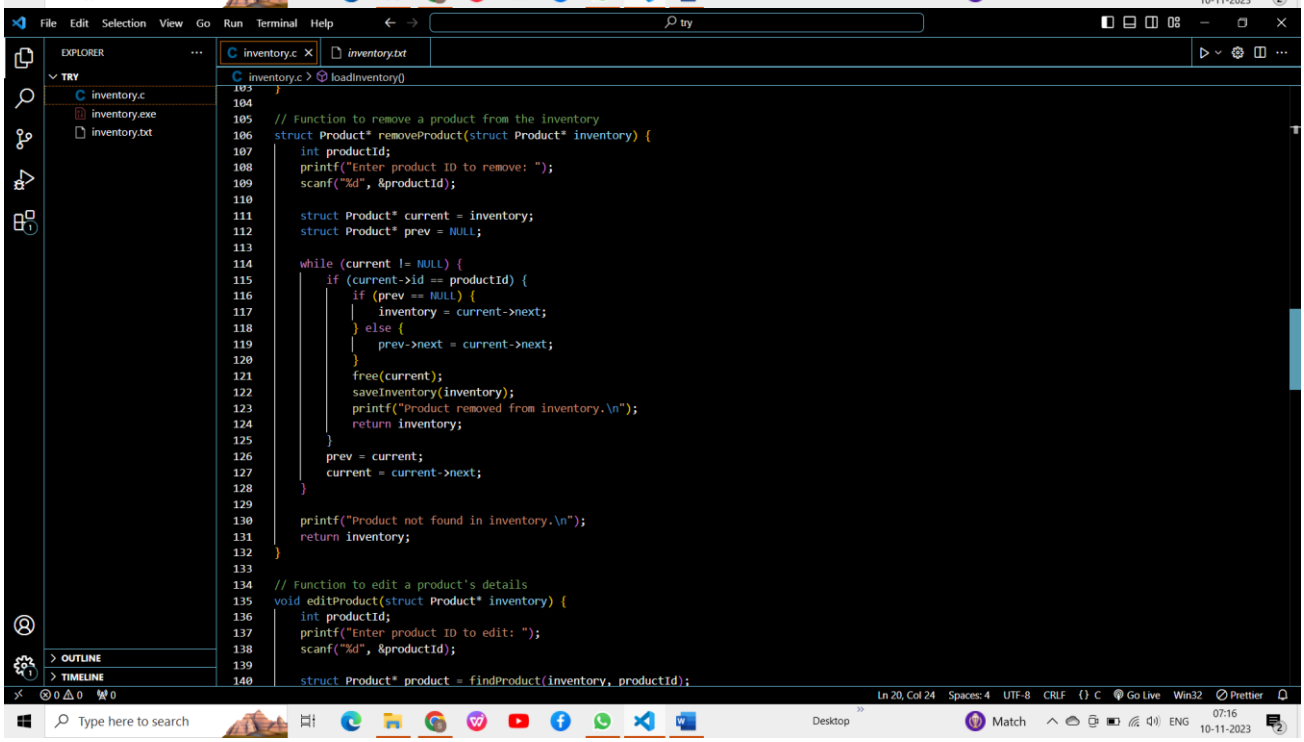
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Structure to represent a product
struct Product {
    int id;
    char name[50];
    double price;
    int quantity;
    struct Product* next;
};

// File to store the inventory data
const char* FILE_NAME = "inventory.txt";

// Function to load the inventory from a file
struct Product* loadInventory() {
    FILE* file = fopen(FILE_NAME, "r");
    if (file == NULL) {
        return NULL;  // File doesn't exist, no products loaded
    }

    struct Product* head = NULL;
    struct Product* current = NULL;

    while (1) {
        struct Product* product = (struct Product*)malloc(sizeof(struct Product));
        if (fread(product, sizeof(struct Product), 1, file) != 1) {
            free(product);
            break;  // Unable to read a product, break the loop
        }

        product->next = NULL;

        if (head == NULL) {
            head = product;
```



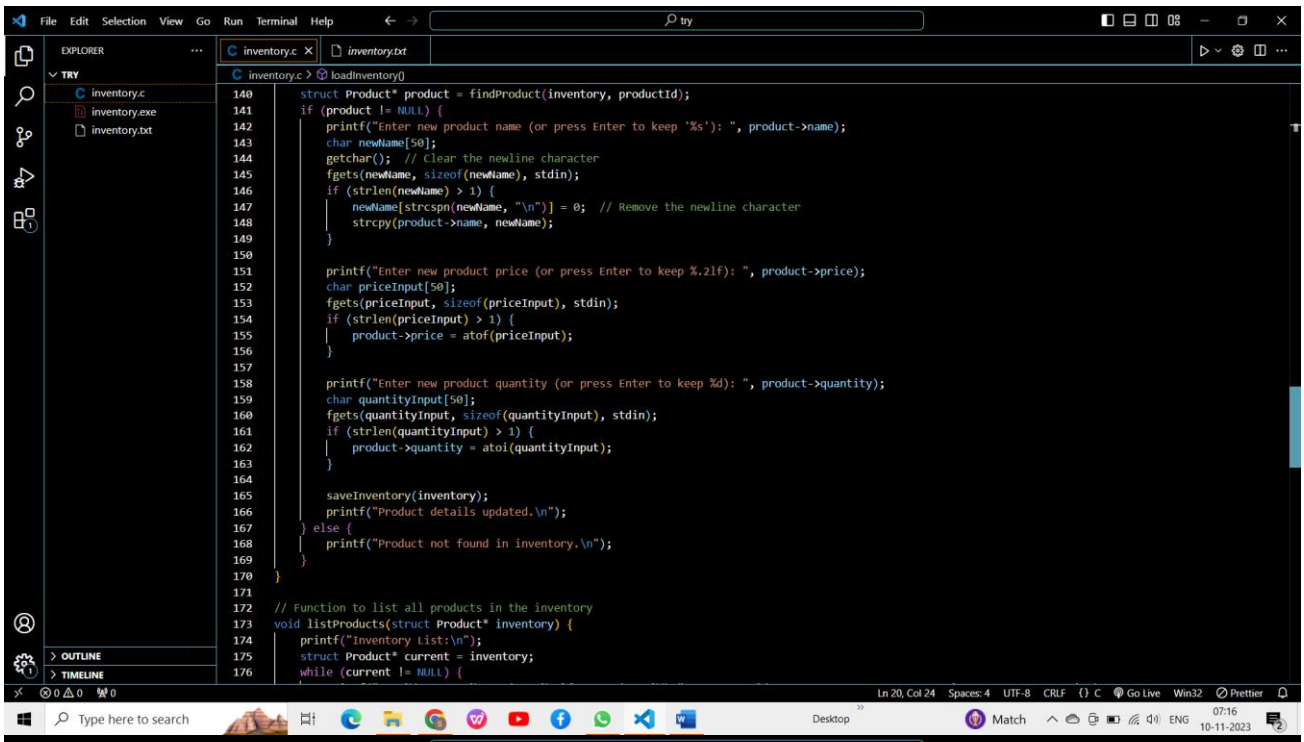
```c
        }

        product->next = NULL;

        if (head == NULL) {
            head = product;
            current = product;
        } else {
            current->next = product;
            current = product;
        }
    }

    fclose(file);
    return head;  // Products loaded successfully
}

// Function to save the inventory to a file
void saveInventory(struct Product* inventory) {
    FILE* file = fopen(FILE_NAME, "w");
    if (file == NULL) {
        printf("Error saving inventory data.\n");
        return;
    }

    struct Product* current = inventory;
    while (current != NULL) {
        fwrite(current, sizeof(struct Product), 1, file);
        current = current->next;
    }

    fclose(file);
    printf("Inventory data saved.\n");
}

// Function to find a product by ID
struct Product* findProduct(struct Product* inventory, int productId) {
```



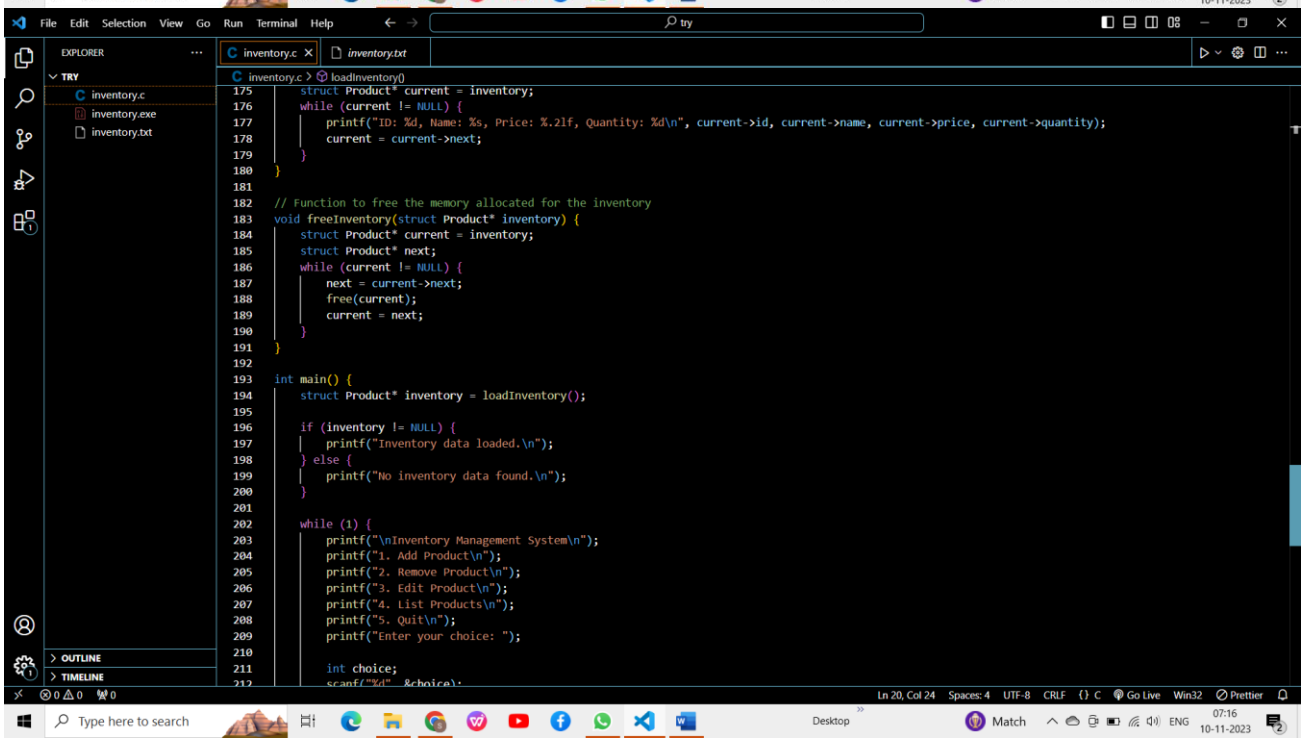
```c
// Function to find a product by ID
struct Product* findProduct(struct Product* inventory, int productId) {
    struct Product* current = inventory;
    while (current != NULL) {
        if (current->id == productId) {
            return current;  // Product found
        }
        current = current->next;
    }
    return NULL;  // Product not found
}

// Function to add a product to the inventory
struct Product* addProduct(struct Product* inventory) {
    struct Product* product = (struct Product*)malloc(sizeof(struct Product));
    printf("Enter product ID: ");
    scanf("%d", &product->id);

    if (findProduct(inventory, product->id) != NULL) {
        printf("Product with the same ID already exists.\n");
        free(product);
        return inventory;
    }

    printf("Enter product name: ");
    scanf("%s", product->name);
    printf("Enter product price: ");
    scanf("%lf", &product->price);
    printf("Enter product quantity: ");
    scanf("%d", &product->quantity);

    product->next = inventory;
    saveInventory(product);

    printf("Product added to inventory.\n");
    return product;
}
```



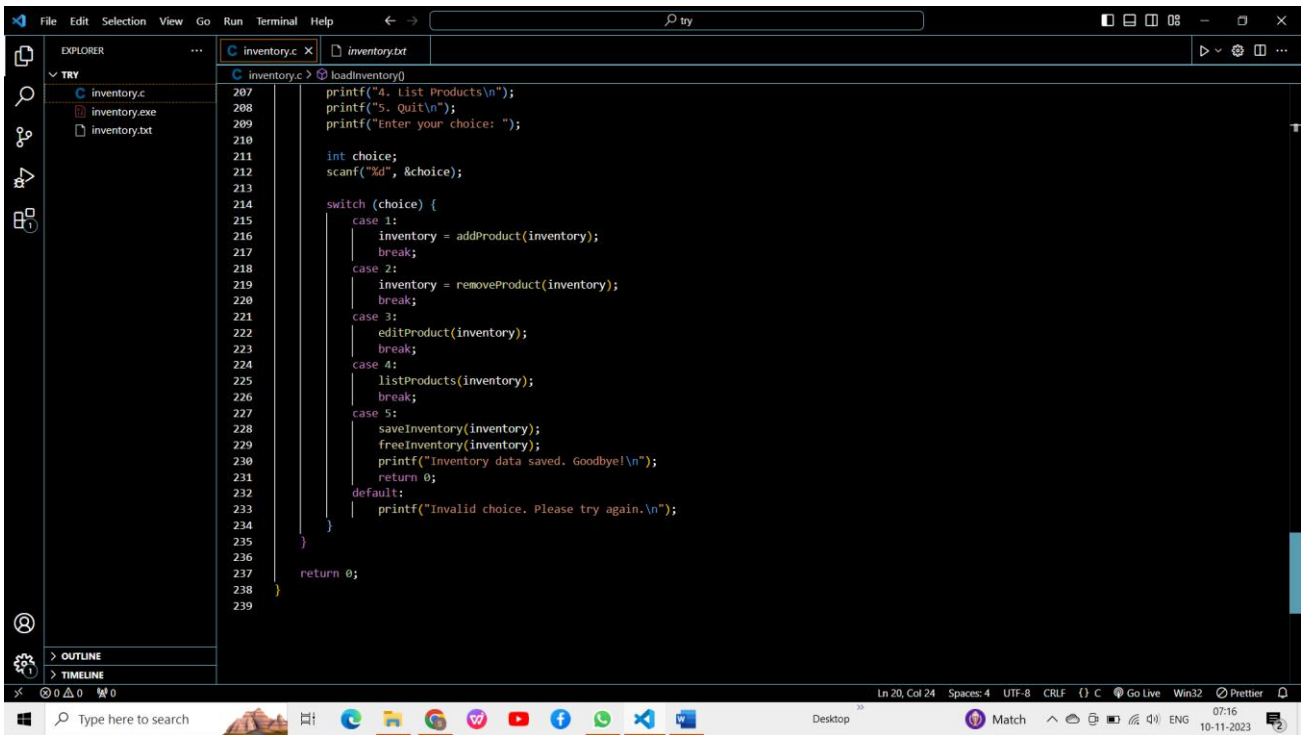
```c
}

// Function to remove a product from the inventory
struct Product* removeProduct(struct Product* inventory) {
    int productId;
    printf("Enter product ID to remove: ");
    scanf("%d", &productId);

    struct Product* current = inventory;
    struct Product* prev = NULL;

    while (current != NULL) {
        if (current->id == productId) {
            if (prev == NULL) {
                inventory = current->next;
            } else {
                prev->next = current->next;
            }
            free(current);
            saveInventory(inventory);
            printf("Product removed from inventory.\n");
            return inventory;
        }
        prev = current;
        current = current->next;
    }

    printf("Product not found in inventory.\n");
    return inventory;
}

// Function to edit a product's details
void editProduct(struct Product* inventory) {
    int productId;
    printf("Enter product ID to edit: ");
    scanf("%d", &productId);

    struct Product* product = findProduct(inventory, productId);
```



```
140     struct Product* product = findProduct(inventory, productId);
141     if (product != NULL) {
142         printf("Enter new product name (or press Enter to keep '%s'): ", product->name);
143         char newName[50];
144         getchar();  // Clear the newline character
145         fgets(newName, sizeof(newName), stdin);
146         if (strlen(newName) > 1) {
147             newName[strcspn(newName, "\n")] = 0;  // Remove the newline character
148             strcpy(product->name, newName);
149         }
150
151         printf("Enter new product price (or press Enter to keep %.2lf): ", product->price);
152         char priceInput[50];
153         fgets(priceInput, sizeof(priceInput), stdin);
154         if (strlen(priceInput) > 1) {
155             product->price = atof(priceInput);
156         }
157
158         printf("Enter new product quantity (or press Enter to keep %d): ", product->quantity);
159         char quantityInput[50];
160         fgets(quantityInput, sizeof(quantityInput), stdin);
161         if (strlen(quantityInput) > 1) {
162             product->quantity = atoi(quantityInput);
163         }
164
165         saveInventory(inventory);
166         printf("Product details updated.\n");
167     } else {
168         printf("Product not found in inventory.\n");
169     }
170 }
171
172 // Function to list all products in the inventory
173 void listProducts(struct Product* inventory) {
174     printf("Inventory List:\n");
175     struct Product* current = inventory;
176     while (current != NULL) {
```



```
175     struct Product* current = inventory;
176     while (current != NULL) {
177         printf("ID: %d, Name: %s, Price: %.2lf, Quantity: %d\n", current->id, current->name, current->price, current->quantity);
178         current = current->next;
179     }
180 }
181
182 // Function to free the memory allocated for the inventory
183 void freeInventory(struct Product* inventory) {
184     struct Product* current = inventory;
185     struct Product* next;
186     while (current != NULL) {
187         next = current->next;
188         free(current);
189         current = next;
190     }
191 }
192
193 int main() {
194     struct Product* inventory = loadInventory();
195
196     if (inventory != NULL) {
197         printf("Inventory data loaded.\n");
198     } else {
199         printf("No inventory data found.\n");
200     }
201
202     while (1) {
203         printf("\nInventory Management System\n");
204         printf("1. Add Product\n");
205         printf("2. Remove Product\n");
206         printf("3. Edit Product\n");
207         printf("4. List Products\n");
208         printf("5. Quit\n");
209         printf("Enter your choice: ");
210
211         int choice;
212         scanf("%d", &choice);
```


File Edit Selection View Go Run Terminal Help
try
EXPLORER
TRY
inventory.c
inventory.exe
inventory.txt
inventory.c
inventory.txt
inventory.c > loadInventory()
printf("4. List Products\n");
printf("5. Quit\n");
printf("Enter your choice: ");
int choice;
scanf("%d", &choice);
switch (choice) {
case 1:
inventory = addProduct(inventory);
break;
case 2:
inventory = removeProduct(inventory);
break;
case 3:
editProduct(inventory);
break;
case 4:
listProducts(inventory);
break;
case 5:
saveInventory(inventory);
freeInventory(inventory);
printf("Inventory data saved. Goodbye!\n");
return 0;
default:
printf("Invalid choice. Please try again.\n");
}
}
return 0;
}
OUTLINE
TIMELINE
Ln 20, Col 24 Spaces: 4 UTF-8 CRLF { } C Go Live Win32 Prettier
Type here to search
Desktop
Match
ENG
07:16
10-11-2023

OUTPUT

```
PS C:\Users\SUDIPTA SUNDAR PAL\OneDrive\Desktop\try> cd "c:\Users\SUDIPTA SUNDAR PAL\OneDrive\Desktop\try\" ; if ($?) { gcc inventory.c -o inventory } ; if
($?) { .\inventory }
Inventory data loaded.

Inventory Management System
1. Add Product
2. Remove Product
3. Edit Product
4. List Products
5. Quit
Enter your choice: 1
Enter product ID: 100
Enter product name: pet supplies
Enter product price: Enter product quantity: Inventory data saved.
Product added to inventory.

Inventory Management System
1. Add Product
2. Remove Product
3. Edit Product
4. List Products
5. Quit
Enter your choice: Enter product ID: Enter product name: Enter product price: 1
Enter product quantity: 20
Inventory data saved.
Product added to inventory.

Inventory Management System
1. Add Product
2. Remove Product
3. Edit Product
4. List Products
5. Quit
Enter your choice: 1
Enter product ID: 30
Enter product name: toys
Enter product price: 100
Enter product quantity: 4
Inventory data saved.
Product added to inventory.

Enter your choice: 2
Enter product ID to remove: 90
Inventory data saved.
Product removed from inventory.

Inventory Management System
1. Add Product
2. Remove Product
3. Edit Product
4. List Products
5. Quit
Enter your choice: 4
Inventory List:
ID: 30, Name: toys, Price: 100.00, Quantity: 4
ID: 10617024, Name: supplies, Price: 1.00, Quantity: 20
ID: 100, Name: pet, Price: 0.00, Quantity: 0
ID: 5, Name: ;, Price: 0.00, Quantity: 0
ID: 12265920, Name: PAL\OneDrive\Desktop\try\", Price: 0.00, Quantity: 1432107587
```

CONCLUSION

The Inventory Management System presented offers a practical solution for managing product data through a console-based interface. This conclusion encapsulates the key aspects and takeaways from the system's implementation:

1. Functionality and Core Features:
   - The system effectively covers essential inventory operations, including adding, removing, editing products, and listing the inventory.
   - A linked list structure ensures dynamic and efficient storage of product data.

2. Data Persistence:
   - The integration of file-based data persistence allows the system to retain inventory information across sessions.
   - This ensures data integrity and provides a seamless experience for users.

3. User Interaction and Input Handling:
   - The text-based menu system provides a straightforward and accessible means for users to interact with the system.
   - Input handling mechanisms enhance user experience by addressing potential input errors.

4. Scalability and Modularity:
   - The use of a linked list allows for the management of varying product quantities, providing a degree of scalability.
   - The modular design facilitates ease of extension, allowing for the addition of new features in the future.

5. Documentation and Readability:
   - Code comments contribute to code clarity and documentation, aiding developers in understanding the implemented logic.
   - Future improvements could involve additional external documentation for end-users and developers.

6. Security Considerations:
   - The system currently lacks explicit security measures, making it susceptible to unauthorized access.
   - Future iterations could explore the implementation of user authentication and

access controls.

7. Usability and User Experience:
- The text-based interface, while functional, may benefit from enhancements to improve overall usability and user experience.
- Consideration for a graphical interface or additional user-friendly features could be explored.

8. Testing and Reliability:
- Rigorous testing, covering various scenarios and edge cases, is essential to ensure the reliability of the system.
- Thorough testing helps identify and address potential issues, contributing to the robustness of the application.

9. Future Directions:
- The current system forms a solid foundation that can be expanded with additional features, improved security measures, and refined user interfaces.
- Integration with external systems or platforms could broaden the system's utility in diverse environments.

In conclusion, the Inventory Management System demonstrates a functional implementation with room for future enhancements. While meeting its core objectives, the system invites further refinements to elevate its security, usability, and overall user experience in diverse inventory management scenarios.

REFERENCES

- "C Programming Absolute Beginner's Guide" by Perry and Miller
- "Data Structures and Algorithms in C++" by Drozdek
- Stack Overflow (for specific programming questions)
- GeeksforGeeks (for tutorials and code examples)
- Cprogramming.com (for C programming tutorials)
- GCC Online Documentation
- GitHub (for version control and code sharing)
- Git documentation and tutorials
- "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin
- "Code Complete" by Steve McConnell