**DEPARTMENTAL STORE MANAGEMENT**

**A PROJECT REPORT**

***Submitted by***

**DANIE LAMUEL P – 312322205036**

**BACHELOR OF TECHNOLOGY**

***in***

**INFORMATION TECHNOLOGY**



# St. JOSEPH’S COLLEGE OF ENGINEERING

**(An Autonomous Institution)**

**St. Joseph’s Group of Institutions OMR, Chennai 600 119**

**ANNA UNIVERSITY: CHENNAI**

**April - 2024**

1

**ANNA UNIVERSITY: CHENNAI 600 025**

**CERTIFICATE OF EVALUATION**

**COLLEGE NAME :** St. Joseph’s College of Engineering, Chennai-600119.

**BRANCH : B.TECH., Information Technology**

**SEMESTER :** IV

|  |  |  |  |
| --- | --- | --- | --- |
| **SI NO.** | **NAME OF THE STUDENT** | **TITLE OF THE PROJECT** | **NAME OF THE**  **SUPERVISOR WITH DESIGNATION** |
| 1. | DANIE LAMUEL P (312322205036) | DEPARTMENTAL STORE MANAGEMENT | Mrs.Heltin Genitha M.E |

**ACKNOWLEDGEMENT**

At the outset we would like to express our sincere gratitude to the beloved **Chairman, Dr.Babu Manoharan, M.A.,M.B.A.,Ph.D.,** for his constant guidance and support.

We would like to express our heartfelt thanks to our respected **Managing Director Mrs. S. Jessie Priya, M.Com.,** for her kind encouragement and blessings.

We wish to express our sincere thanks to our **Executive Director Mr. B. Shashi Sekar, M.Sc.,** for providing ample facilities in the institution.

We express our deepest gratitude and thanks to our beloved **Principal Dr.Vaddi Seshagiri Rao, M.E., M.B.A., Ph.D., F.I.E.,** for his inspirational ideas during the course of the project.

We wish to express our sincere thanks and gratitude to **Dr.V. Muthu**

**Lakshmi, M.E.,Ph.D.,** Head of the Department , Department of Information

Technology & Artificial Intelligence and Machine Learning , St.Joseph’s

College Of Engineering for her guidance and assistance in solving the

various intricacies involved in the project.

It is with deep sense of gratitude that we acknowledge our indebtness

to our supervisor **Mrs.Heltin Genitha** for her expert guidance and

connoisseur suggestion.

Finally we thank our department staff members who helped us in the successful completion of this project.

### Abstract :

* The "Departmental Store Management System" is a simple C program that allows users to manage products in a departmental store.
* The program provides four main functionalities: adding a new product, displaying all products, making a sale, and exiting the system.
* It utilizes an array of structures to store product information such as product ID, name, price, and quantity.
* The program is menu-driven, where the user can choose the desired operation by entering a corresponding number.
* If the product list reaches its maximum capacity, the program informs the user that no more products can be added.
* Invalid inputs are handled with appropriate error messages. Overall, this system helps departmental store owners efficiently manage their product inventory and sales.
* 1.**Introduction :**
* Welcome to the world of efficient and organized retail management - the Departmental Store Management! In this fast-paced and competitive landscape, managing a departmental store requires a unique blend of skills, innovation, and customer-centric strategies. As the heart of retail operations, a well-run departmental store is a hub of diverse products and services, catering to the diverse needs and preferences of its valued customers.
* In this dynamic industry, effective departmental store management encompasses a multitude of vital aspects. From maintaining a comprehensive inventory to optimizing visual merchandising, from crafting exceptional customer experiences to ensuring seamless operations, each facet plays a crucial role in the store's success.
* As departmental store managers, we are the orchestrators of this retail symphony. We are responsible for inspiring and leading our dedicated team, fostering a collaborative work environment that fosters growth and empowers employees to deliver outstanding service. By staying attuned to market trends, understanding consumer behavior, and harnessing the power of technology, we aim to stay ahead of the curve and continuously elevate our store's performance.
* In this ever-changing landscape, our commitment to innovation is unwavering. From implementing cutting-edge POS systems to exploring sustainable practices, we embrace modernity while preserving the essence of traditional retail values.
* At the core of our departmental store management lies a profound dedication to our customers. We are driven to provide a delightful and memorable shopping experience, exceeding expectations at every touchpoint. Our goal is to build lasting relationships with our customers, transforming them into loyal advocates who trust us with their retail needs.
* As we embark on this journey, we recognize that challenges will arise, but with a passionate team, a customer-centric mindset, and a forward-thinking approach, we are poised to overcome any obstacle that comes our way. Together, we will create a departmental store that is not just a place of commerce, but a vibrant hub where shoppers can discover, connect, and find joy in their retail endeavors.
* Join us on this exciting adventure in Departmental Store Management, where innovation, teamwork, and customer satisfaction are our guiding stars, and together, we will set new standards of excellence in the world of retail.
* 2.**Design :**

### The departmental store management system is an integrated and efficient solution designed to streamline and enhance the operations of a modern retail establishment. The system encompasses a comprehensive range of features, including inventory management, point-of-sale (POS) processing, customer relationship management (CRM), and employee tracking. Through a user-friendly interface, store managers can monitor real-time stock levels, track sales, and analyze customer preferences to optimize product offerings.

### The system also facilitates seamless transactions, ensuring a smooth and hassle-free shopping experience for customers. Additionally, it supports employee scheduling, performance monitoring, and payroll management, contributing to a productive and motivated workforce. With its robust reporting and analytics capabilities, the departmental store management system empowers decision-makers to make data-driven choices, thereby driving growth and profitability for the retail business.

### 2.1 Code Structure :

* The provided code is a simple C program for a Departmental Store Management System. The program allows users to add products, display all products, and update the quantity of a specific product. The data of products is stored using a struct Product. The program uses a switch-case loop to handle user choices until the user decides to exit the program.
* **Header Files:** The program starts by including necessary header files, such as ‘<stdio.h>’, ‘<string.h>’ and ‘<stdlib.h>’ which provides input/output functionality, manipulate a string or array of characters and declares various utility functions. These header files define the necessary functions and structures used in the program.
* **Structure Definition:** The ‘Product’ structure is defined to represent each product's information. It includes attributes like ‘name’, ‘price’, ‘quantity’ and ‘ID’. The structure definition encapsulates the data related to a product and allows for organised storage and access of product information.
* **Function Prototypes:** After the structure definition, function prototypes are declared. This section lists the functions that will be defined later in the program. Function prototypes provide information about the function's name, return type, and input parameters, enabling the compiler to check for correct usage and allow functions to be called before they are defined.
* **Function Definitions:** The program proceeds with the definition of various functions used in the program. These functions include the ‘addProduct’ function, which displays the name, quantity, ID and price of the product, and the ‘main’ function, which serves as the entry point of the program.
* **Main Function:** The ‘main’ function is where the program execution begins. It starts by declaring local variables, such as ‘numProducts' to store user inputs. It prompts the user to enter the number of players participating in the game and reads this input using the ‘scanf’ function.
* **Array Declaration:** Based on the number of products entered, an array of product structures called ‘Product’ is declared with a size equal to ‘numProducts’. This array provides a container to store the product’s information.
* **User Input Loop:** A ‘for’ loop is used to collect the details of each product. Within the loop, the program prompts the user to enter the product's name, quantity, ID and price. These inputs are read using the ‘scanf’ function and stored in the corresponding attributes of the ‘Product’ structure within the product array.
* **Function Call:** After the user input loop, the ‘addProducts’ function is called, passing the ‘products’ array and the number of products as arguments. This function iterates over the array, printing the name, ID, quantity and price of each product, effectively displaying the product details.
* **Return Statement:** Finally, the ‘main’ function concludes with a return statement, usually returning 0 to indicate successful program execution.
  + The code is well-structured and easy to understand. It utilizes a struct Product to represent each product with its ID, name, price, and quantity. The user can interact with the system by selecting options from the menu, and the functions handle the respective operations. The program continues to run until the user chooses to exit by entering 4.
* 2.2 **Data Structure :**
* The provided code implements a simple Departmental Store Management System in C. The system allows users to add new products to the store, display all existing products, and update the quantity of a specific product. The data structure used to store product information is an array of structs.
* Here's a breakdown of the code and the data structure it uses:
* Data Structure:
* The data structure used for representing each product is defined as a struct named `Product`. It consists of the following members:
  + - `int id`: The unique identifier for each product.
    - `char name[50]`: The name of the product (up to 49 characters long).
    - `float price`: The price of the product.
    - `int quantity`: The quantity of the product available in the store.
* The array `departmentalStore` is used to store instances of the `Product` struct. It has a maximum size of `MAX\_PRODUCTS` (defined as 100) to limit the number of products that can be stored in the store.
* 2. Functions:
  + - `addProduct()`: Allows the user to add a new product to the store by entering its ID, name, price, and quantity. It checks if there is enough space in the `departmentalStore` array before adding a new product.
    - `displayProducts()`: Displays all products currently stored in the `departmentalStore` array. If no products are present, it notifies the user accordingly.
    - `findProductIndex(int productId)`: Searches for a product in the `departmentalStore` array based on its ID and returns the index if found; otherwise, it returns -1.
    - `updateProductQuantity()`: Allows the user to update the quantity of a specific product by providing its ID. It uses the `findProductIndex` function to locate the product and then updates its quantity.
* 3. Main Function:
  + - The `main()` function serves as the entry point of the program. It presents a simple menu-driven interface to the user, allowing them to choose from various options (add a product, display products, update quantity, or exit the program). The user's choice is processed using a `switch` statement, and the corresponding function is called accordingly. The program continues to display the menu until the user chooses to exit.
* The code provides a basic example of a departmental store management system using C and a simple data structure to manage product information. However, keep in mind that this implementation has limitations and does not include more advanced features such as data persistence (storing data between program runs), error handling, or user input validation. In a real-world application, these aspects should be considered and implemented for a more robust system.
* 2.3 **Error Handling :**
* To add error handling to the existing Departmental Store Management System, you can perform input validation and handle potential errors that may arise during user interactions. Here are some improvements you can make:
* **Input Validation :**
  + - When accepting the product ID, price, and quantity from the user, you should validate the input to ensure they provide valid data. For example, if the user enters non-numeric characters for the ID or negative price/quantity, it should be considered invalid input.
    - Use a loop to repeatedly prompt for input until valid data is provided.
* **Clear Input Buffer:**
  + - After using `scanf()` to read data, it's essential to clear the input buffer to avoid potential issues with leftover characters in the buffer. You can do this by consuming any remaining characters in the buffer after each `scanf()` call.
* Check ID Uniqueness:
  + - Ensure that each product ID is unique and not already present in the `departmentalStore` array before adding a new product. If the user attempts to add a product with an existing ID, display an error message.
* Here's an updated version of the code with error handling improvements:
* // ... (previous code remains the same) ...
* void clearInputBuffer() {
* int c;
* while ((c = getchar()) != '\n' && c != EOF);
* }
* void addProduct() {
* if (numProducts >= MAX\_PRODUCTS) {
* printf("Cannot add more products. Store is full.\n");
* return;
* }
* struct Product newProduct;

### printf("Enter product ID: ");

### while (scanf("%d", &newProduct.id) != 1) {

### clearInputBuffer();

### printf("Invalid input. Enter a valid product ID: ");

### }

### clearInputBuffer();

### // Check if the ID is already in use

### for (int i = 0; i < numProducts; i++) {

### if (departmentalStore[i].id == newProduct.id) {

### printf("Product with this ID already exists. Choose a unique ID.\n");

### return;

### }

### }

### printf("Enter product name: ");

### fgets(newProduct.name, sizeof(newProduct.name), stdin);

### newProduct.name[strcspn(newProduct.name, "\n")] = '\0'; // Remove the newline character

### printf("Enter product price: ");

### while (scanf("%f", &newProduct.price) != 1 || newProduct.price < 0) {

### clearInputBuffer();

### printf("Invalid input. Enter a valid product price: ");

### }

### clearInputBuffer();

### printf("Enter product quantity: ");

### while (scanf("%d", &newProduct.quantity) != 1 || newProduct.quantity < 0) {

### clearInputBuffer();

### printf("Invalid input. Enter a valid product quantity: ");

### }

### clearInputBuffer();

### departmentalStore[numProducts++] = newProduct;

### printf("Product added successfully!\n");

### }

### // ... (previous code remains the same) ...

### int main() {

### // ... (previous code remains the same) ...

### return 0;

### }

### ```

### In this updated version, the code now validates input for product ID, price, and quantity. It ensures that the user provides valid data and that the product ID is unique before adding a new product. Additionally, the `clearInputBuffer()` function is used to clear the input buffer after each `scanf()` call to avoid issues with leftover characters in the input buffer.

### Remember that this is just a basic example of error handling, and in a real-world application, you may want to implement more sophisticated error handling mechanisms and handle exceptions more gracefully.

### 3.Requirements :

* **AC compiler:**
* You need a C compiler installed on your system to compile and execute the C source code.
* **Operating System:**
* The program can be compiled and run on various operating systems, including Windows, macOS, and Linux.
* **Development Environment:**
* You can use any text editor or integrated development environment (IDE) to write the C code.
* **C Standard Library:**
* The program uses the standard C library, which is typically available by default with the C compiler.
* **4.Implementation :**
* #include <stdio.h>
* #include <stdlib.h>
* #include <string.h>
* #define MAX\_PRODUCTS 100
* struct Product {
* int id;
* char name[50];
* float price;
* int quantity;
* };
* struct Product departmentalStore[MAX\_PRODUCTS];
* int numProducts = 0;

### void addProduct() {

### if (numProducts >= MAX\_PRODUCTS) {

### printf("Cannot add more products. Store is full.\n");

### return;

### }

### struct Product newProduct;

### printf("Enter product ID: ");

### scanf("%d", &newProduct.id);

### printf("Enter product name: ");

### scanf(" %[^\n]", newProduct.name);

### printf("Enter product price: ");

### scanf("%f", &newProduct.price);

### printf("Enter product quantity: ");

### scanf("%d", &newProduct.quantity);

### departmentalStore[numProducts++] = newProduct;

### printf("Product added successfully!\n");

### }

### void displayProducts() {

### if (numProducts == 0) {

### printf("No products found in the store.\n");

### return;

### }

### printf("----------------------------------------------------\n");

### printf("ID\tName\t\tPrice\tQuantity\n");

### printf("----------------------------------------------------\n");

### for (int i = 0; i < numProducts; i++) {

### printf("%d\t%-20s%.2f\t%d\n", departmentalStore[i].id, departmentalStore[i].name,

### departmentalStore[i].price, departmentalStore[i].quantity);

### }

### printf("----------------------------------------------------\n");

### }

### int findProductIndex(int productId) {

### for (int i = 0; i < numProducts; i++) {

### if (departmentalStore[i].id == productId) {

### return i;

### }

### }

### return -1; // Product not found

### }

### void updateProductQuantity() {

### int productId, quantity;

### printf("Enter the product ID to update quantity: ");

### scanf("%d", &productId);

### int productIndex = findProductIndex(productId);

### if (productIndex == -1) {

### printf("Product not found!\n");

### } else {

### printf("Enter the new quantity: ");

### scanf("%d", &quantity);

### departmentalStore[productIndex].quantity = quantity;

### printf("Product quantity updated successfully!\n");

### }

### }

### int main() {

### int choice;

### do {

### printf("\nDepartmental Store Management System\n");

### printf("1. Add a new product\n");

### printf("2. Display all products\n");

### printf("3. Update product quantity\n");

### printf("4. Exit\n");

### printf("Enter your choice: ");

### scanf("%d", &choice);

### switch (choice) {

### case 1:

### addProduct();

### break;

### case 2:

### displayProducts();

### break;

### case 3:

### updateProductQuantity();

### break;

### case 4:

### printf("Exiting program. Goodbye!\n");

### break;

### default:

### printf("Invalid choice. Please try again.\n");

### break;

### }

### } while (choice != 4);

### return 0;

### }

### 5.Output :

* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 1
* Enter product ID: 101
* Enter product name: Laptop
* Enter product price: 1200.00
* Enter product quantity: 5
* Product added successfully!
* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 1
* Enter product ID: 102
* Enter product name: Smartphone
* Enter product price: 500.50
* Enter product quantity: 10
* Product added successfully!
* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 2
* ----------------------------------------------------
* ID Name P rice Quantity
* ----------------------------------------------------
* 101 Laptop 1200.00 5
* 102 Smartphone 500.50 10
* ----------------------------------------------------
* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 3
* Enter the product ID to update quantity: 102
* Enter the new quantity: 8
* Product quantity updated successfully!
* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 2
* ----------------------------------------------------
* ID Name Price Quantity
* ----------------------------------------------------
* 101 Laptop 1200.00 5
* 102 Smartphone 500.50 8
* ----------------------------------------------------
* Departmental Store Management System
* Add a new product
* Display all products
* Update product quantity
* Exit
* Enter your choice: 4
* Exiting program. Goodbye!

### 6.Conclusion :

* The C program presented above is a simple Departmental Store Management system. It allows users to manage products and customers, perform transactions, and view product and customer information. The program showcases basic functionality that could be expanded and integrated into a more comprehensive management system.
* The program consists of several components:
* Data structures: The structures `Product` and `Customer` are defined to hold information about products and customers, respectively.
* Function prototypes: All functions used in the program are declared before the `main()` function.
* Initialization functions: The `initializeProducts()` and `initializeCustomers()` functions populate the product and customer arrays with some sample data for demonstration purposes. In a real application, this data would typically come from a database or file.
* Display functions: The `displayMenu()`, `displayProducts()`, and `displayCustomers()` functions are responsible for displaying the main menu, product information, and customer information, respectively.
* Add functions: The `addProduct()` and `addCustomer()` functions allow the user to add new products and customers, respectively. These functions prompt the user to enter details and update the corresponding arrays.
* Transaction function: The `makeTransaction()` function facilitates making a transaction between a customer and a product. It prompts the user to select a product and a customer for the transaction and performs any necessary updates.
* Main function: The `main()` function is the entry point of the program. It initializes the product and customer arrays, displays the main menu, and handles user input through a switch statement.
* Overall, this C program serves as a foundation for a Departmental Store Management system, which can be further developed and customized to meet specific requirements in real-world scenarios.
* 7.**Reference :**
* LearnProgramo:<https://learnprogramo.com/50-interesting-programming-c-projects-download-with-source-code/>
* Book : Programming in C [Second Edition][Oxford] ,
* Reema Thareja [Assistant Professor,
* Department Of Computer Science ,
* SPM College For Women ,
* University Of Delhi .