

## **ABSTRACT**

In the fast-paced environment of a bustling city, the efficiency of restaurant operations is paramount to ensuring customer satisfaction and seamless service. This project presents a comprehensive Restaurant Management System (RMS) developed in C programming, aimed at optimizing the various facets of restaurant management. The system is designed to handle key operations including menu management, order processing, inventory control, and financial transactions.

The RMS allows for dynamic management of menu items, enabling admin to insert, delete, and update item details such as names and prices. Customers are provided with a user-friendly interface to browse the menu, view dish details, and place orders effortlessly. The system assigns unique identifiers to each order and calculates the total cost, with an optional discount for students.

Orders are meticulously recorded and stored using file handling techniques, facilitating future reference and ensuring data integrity.

For restaurant staff, the system offers a consolidated view of all placed orders, detailing items, quantities, and total prices, thereby aiding in efficient order fulfillment and maintaining a clear record of transactions. This streamlined approach not only enhances the dining experience for customers but also simplifies the operational workload for restaurant staff, promoting an organized and well-managed dining establishment.

Overall, the Restaurant Management System shows how technology can help make restaurants run more smoothly. It helps balance great customer service with efficient operations, making things easier for both customers and staff.

## **Table of Contents**

### **Chapter 1**

<b>Introduction</b>	<b>4</b>
1.1 About the system	4
1.2 Purpose	4
1.3 Why this system is necessary?	5

### **Chapter 2**

<b>Features</b>	<b>06</b>
2.1 About the features of your system	06

### **Chapter 3**

<b>Implementation</b>	<b>07</b>
3.1 C concepts you used for your project	07

### **Chapter 4**

<b>System Testing</b>	<b>09</b>
4.1 Introduction	09
4.2 Input and desired output	09

### **Chapter 5**

<b>Conclusion</b>	<b>14</b>
5.1 Good features	14
5.2 Limitation of the system	14
5.3 Future Enhancement	15

## **Chapter-1 (Introduction)**

### **1.1 About the system**

This project aims to develop a Restaurant Management System (RMS) using C programming. The current system at the quaint restaurant is likely inefficient, lacking features to streamline operations. This can lead to issues like slow order processing, inventory mismanagement, and inaccurate record keeping. So we create a RMS that can handle menu items, orders, and inventory efficiently. It allows administrators to manage menu items by inserting, deleting, or updating their details. Customers can browse the menu, place orders, and avail discounts if applicable. Orders are assigned unique identifiers and stored for future reference. Payment processing and inventory management functionalities are also included. Additionally, restaurant staff can view and manage placed orders for efficient order fulfillment and transaction recording.

### **1.2 Purpose**

The purpose of the restaurant management system developed in C programming is to streamline and optimize the operations of a quaint restaurant in a bustling city. It provides a user-friendly interface for customers to view the menu with item names and prices. Enable customers to place orders easily, specifying their desired menu items and quantities. Allow for potential discounts, such as for students, to be applied to orders. For staff, it stores order details, including items, quantities, and total prices, for future reference and record-keeping. Offer restaurant staff an efficient way to view and manage all placed orders, aiding in order fulfillment and maintaining clear records. Allow admin to insert, delete, and update menu items, including their names and prices. Maintain an organized structure for menu items, ensuring accurate and up-to-date information is available to customers. Overall, the system is designed to create an efficient and organized environment where customers can enjoy a delightful dining experience while restaurant staff can manage orders, inventory, and transactions with ease and accuracy.

### 1.3 Why this system is necessary?

An RMS is crucial for any restaurant's smooth operation. Here's how it will benefit:

- Improved Customer Experience: Customers can easily browse menus, place orders efficiently, and potentially avail student discounts.
- Enhanced Order Management: Staff can quickly view orders, prioritize fulfillment, and maintain an accurate log of transactions.
- Reduce Time: RMS saves a lot of time.
- Streamlined Operations: Automation of tasks reduces processing time and human error, allowing staff to focus on delivering exceptional service.

## **Chapter-2 (Features)**

### **2.1 About the feature of system**

User Management:

Registration : Admin and staff can register with their details.

Login : Users log in as admin or staff with verified credentials.

Admin Features:

Insert, Display, Update, and Delete Menu Items: Admins manage the menu, including adding, viewing, updating, and deleting items.

Staff Features:

View Orders : Staff can see all placed orders.

Pay Bill : Staff can mark orders as paid by table number.

Customer Features :

Place Order : Customers order items from the menu, with a student discount option

Order Summary : Customers receive a summary of their order with details and total cost.

File Handling: Uses text files to store menu items, orders, and user credentials.  
The system ensures efficient restaurant management by reducing manual tasks  
And maintaining accurate records.

## **Chapter-3 (Implementation)**

### **3.1. C concepts you used for your project:**

This code uses a variety of fundamental C programming concepts to implement a simple restaurant management system. Below is a brief description of each key concept used:

#### **1. Structs**

Structs are used to define custom data types that group related variables together. In this code, structs are used to define:

- ``struct FOOD``: Represents a food item with a unique ID, name, and price.
- ``struct Update``: Represents an update to a food item with a new name and price.
- ``struct Order``: Represents a customer's order with table number, item ID, quantity, price, student status, and payment status.
- ``struct login``: Represents login details with first name, last name, username, and password.

#### **2. File I/O**

File operations are extensively used to store and retrieve data:

- ``fopen``, ``fclose``: To open and close files.
- ``fprintf``, ``fscanf``, ``fgets``, ``fputc``, ``fgetc``: To write to and read from files.
- ``remove``, ``rename``: To delete and rename files.

#### **3. Control Structure**

Various control structures are used for flow control:

- ``while``: Used for looping until a condition is met (e.g., main menu loop, reading from files).
- ``if-else``: Used for decision-making.
- ``switch-case``: Used for menu selection.

#### **4. Functions**

Functions are used to modularize code and perform specific tasks:

- ``insertItem``, ``displayItem``, ``updateItem``, ``deleteItem``: Admin operations on food items.
- ``placeOrder``: For customers to place orders.
- ``viewOrders``, ``payBill``: Staff operations to view orders and mark bills as paid.
- ``registration``, ``login``: For user registration and login.

## 5. Arrays

Arrays are used to store multiple items of the same type:

- ``struct Order orders [MAX_ITEMS]``: Stores multiple orders up to a defined limit.

## 6. Standard Input/Output

Standard I/O functions are used for user interaction:

- ``printf``: To display output to the user.
- ``scanf``: To read input from the user.

## 7. String Handling

String operations are used to handle text input and output:

- ``strcpy``, ``strcmp``: To copy and compare strings.
- ``fgets``, ``strcspn``: To read strings and handle newline characters.

## 8. Data Validation

Basic validation is performed to ensure data integrity:

- Checking if a file was opened successfully.
- Ensuring valid item IDs and quantities when placing orders.

## 9. Conditional Compilation

Preprocessor directives are used for constants:

- ``#define MAX_ITEMS 10``, ``#define MAX_ORDERS 100``: To define constant values used in the code.

## 10. Basic Arithmetic and Discounts

Basic arithmetic operations are used to calculate prices and discounts:

- Calculating the subtotal, discount, and total cost for orders.

By combining these concepts, the code provides a functional and modular system for managing a restaurant's operations, including user registration, login, food item management, and order processing

## **Chapter-4 (System testing)**

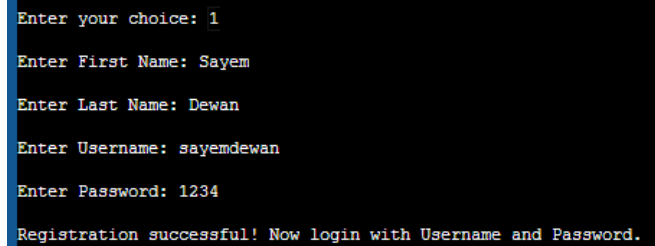
### **4.1. Introduction**

System testing involves evaluating the entire system and presenting a comprehensive report. This process allows clients to verify if their requirements have been met. By conducting thorough testing, hidden errors are identified and corrected, enhancing the overall performance of the product. Additionally, testing ensures that the software remains functional across different devices, preventing negative user experiences. Ultimately, testing improves compatibility and adaptability, leading to a more reliable product.

### **4.2. Input and desired output**

#### **(a) Registration**

The registration system is for admins and staff only. They will register with their username and password, which will be useful later during login. Registration successful message and user details should be saved in login.txt.

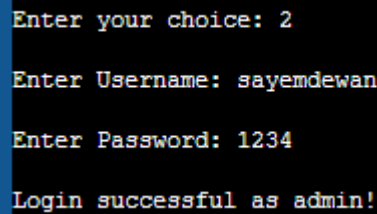


```
Enter your choice: 1
Enter First Name: Sayem
Enter Last Name: Dewan
Enter Username: sayemdewan
Enter Password: 1234
Registration successful! Now login with Username and Password.
```

**Figure-4.2(i): Registration**



### (b) Login system



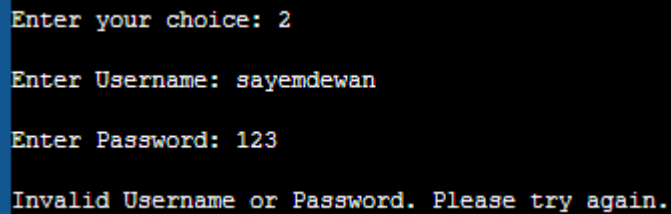
```
Enter your choice: 2
Enter Username: sayemdewan
Enter Password: 1234
Login successful as admin!
```

A terminal window with a black background and a blue vertical bar on the left. It displays a login process where the user enters '2' for choice, 'sayemdewan' for username, and '1234' for password, resulting in a successful login message.

*Figure-4.2(ii) Login system*

Here we aimed to keep the login system as simple as possible. For this reason, we took the user name and password only. Admin or staff can login to enter their username and password. Login successful as admin message and admin menu should be displayed.

### (c) Test for securing Login



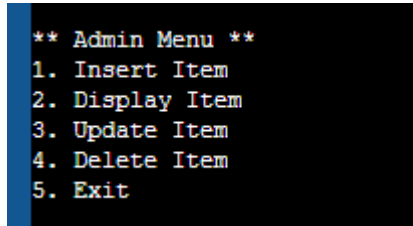
```
Enter your choice: 2
Enter Username: sayemdewan
Enter Password: 123
Invalid Username or Password. Please try again.
```

A terminal window with a black background and a blue vertical bar on the left. It displays a login attempt where the user enters '2' for choice, 'sayemdewan' for username, and '123' for password, resulting in an error message: 'Invalid Username or Password. Please try again.'

*Figure-4.2(IV) securing login*

If the user provides the wrong username or password, the following message will appear. This means that users who do not know the username and password will not be able to log in.

(d) Admin Menu



```
** Admin Menu **  
1. Insert Item  
2. Display Item  
3. Update Item  
4. Delete Item  
5. Exit
```

Figure-4.2(V) Admin menu

From Figure-4.2(V) we can see that admin can do four things. Insert, display, update and delete items. The admin will first insert the item where they will provide the food unique ID, name, and price. These will be stored in the menuitem file. Then, if they want to see the available food, they will click on the 'display item' option. If they want to delete something, they can provide the food ID, and it will be deleted. If they want to update any food, they should first select the ID, and then the option to change the name or price will appear. If admin want to change the name, admin will give the new name and if he want to change the price, he will give the new price

### (e) Customer

```
** Placing Order **

Menu:
ID      Name      Price
11      burger    230.00

Are you a student (1 for yes, 0 for no)? 1

Enter Today's Date, Table no, Item IDs and quantities (0 to finish):

Date: 05/06/2024

Table no: 2
Item ID: 11
Quantity: 2
Item ID: 0

Order Summary :
Date       : 05/06/2024
Table No   : 2
Total Items : 1
Subtotal   : 460.00
Discount   : 46.00
Total Cost : 414.00
```

Figure: 4.2(VI) Customer

There is a placing order for the customer. First, the menu will be displayed. There is an option of a discount for students, which is a very good option for them. Then, the customer will enter the date, table number, item ID, and quantity. After entering these, the order summary will appear in front of them where they can see their total cost.

## (f) Staff

```
** Staff Menu **
1. View Orders
2. Pay Bill
3. Exit

Enter your choice: 1

** All Orders **
Date : 05/06/2024
Order for Table No: 2
Item ID: 11, Quantity: 2, Price: 460.00, Payment Status: Not Paid
Subtotal: 460.00
Discount: 46.00
Total Cost: 414.00
```

Figure: 4.2(VII) Staff

```
Enter your choice: 2

** Pay Bill **
Enter the Table No: 2
Payment for Table No: 2 marked as successful.
```

Figure: 4.2(VIII) Staff

There are two options for the staff: view orders and paybill. In the view order option, the staff can see who has placed which order. There will be no mistakes because the table number will be included in the order. If the last customer pays, and the table number is provided in the pay bill, the customer's order will show as paid.

The system testing phase has been completed successfully. All major functionalities have been tested, and the application performed as expected. Below are the key outcomes:

- User Registration and Login: Both admin and staff login functionalities are working correctly.
- Admin Features: Admin can insert, display, update, and delete food items as intended.
- Staff Features: Staff can view orders and mark bills as paid.
- Customer Features: Customers can place orders, and the system correctly handles order details, including calculating the subtotal, applying student discounts, and recording orders.

## **Chapter-5 (Conclusions)**

### **5.1. Good Features:**

The restaurant management system designed in the provided code offers several benefits and features that enhance the overall experience for both customers and staff:

#### **User Registration and Login:**

The system allows users to register and login with a username and password. Separate login functionalities are available for admin and staff roles.

#### **Admin Operations:**

- Insert Item: Admins can add new items to the food menu.
- Display Item: Admins can view all items in the food menu.
- Update Item: Admins can update the name or price of an existing food item.
- Delete Item: Admins can delete an item from the food menu.

#### **Customer Operations:**

- Place Order: Customers can place orders by selecting items from the menu, specifying quantities, and indicating if they are students (to receive a discount).

### **5.2. Limitation of the system**

Despite its benefits, the restaurant management system has several limitations that need to be addressed:

- Admin & Staff Login
- Delete Item Order
- Single User Access
- Basic Error Handling

### 5.3. Future enhancement:

To address the limitations and further improve the system, the following enhancements are proposed:

**Role-based Access Control:** Implement a more robust role-based access control system where different users (admin, staff, customer) have clearly defined and separate permissions.

**Order Management Improvements:** Allow customers to modify or cancel their orders. Introduce more detailed order statuses (e.g., pending, preparing, served, paid)

**Inventory Management:** Implement inventory tracking for food items to ensure items are in stock before allowing orders to be place.

Implementing these enhancements will greatly improve the functionality, security, and user experience of the Restaurant Management System.