**DESIGNING A MINI SOFTWARE FOR CUSTOMER MANAGEMENT SYSTEM**

MINOR PROJECT REPORT

By

**RITI MATANGI (RA2211056010035)**

**DHIVYA (RA2211056010021)**

**ROYCE MARK (RA2211056010040)**

Under the guidance of   
**Dr. ARCHANA K S***In partial fulfillment for the Course*

of

**21CSC201T – Data Structures and Algorithms**

in COMPUTER SCIENCE ENGINEERING

(SPECIALISATION IN DATA SCIENCE)



**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 " **Designing a mini software for customer management system**" is the bonafide work of **Dhivya (RA2211056010021), Royce mark(RA2211056010040)** and **Riti Matangi(RA2211056010035)** who carried out the work under my supervision.

**SIGNATURE**

**Dr.ARCHANA K S**

**Asst. Professor**

**CTECH IN DATA SCIENCE**

**SRM Institute of Science and Technology**

**Kattankulathur**

**PROBLEM STATEMENT**

The objective of implementing a Customer Management System is to address these challenges and provide the following solutions:

**Data Integration:**

Create a centralized repository for customer data, ensuring consistency and accuracy.

**Effective Communication:**

Implement features for personalized communication, including email marketing, SMS alerts, and chat support.

**Data Analytics**:

Incorporate analytics tools to gain insights from customer data, enabling data-driven decision-making.

**Automation:**

Automate routine tasks such as data entry, appointment scheduling, and customer follow-ups to improve operational efficiency.

**Security and Compliance:**

Ensure that the CMS complies with data privacy regulations and maintains robust security measures to protect customer data.

**PROBLEM EXPLANATION**

**Customer Information Management:**

* Users can add, update, and retrieve customer information.
* Common customer data includes names, contact details, addresses, and more.

**Customer Interaction Tracking:**

* The system records customer interactions such as inquiries, complaints, and feedback.
* Communication history, including emails, phone calls, and chats, is often logged.

**Order and Purchase History:**

* The CMS tracks customer orders and purchase history.
* Details like order date, product or service details, quantity, and order status are recorded.

**Support and Maintenance:**

* Ongoing support and maintenance are provided, including bug fixes, updates, and improvements.



**Client device**

* This is the device that the customer uses to send and receive messages, such as a smartphone, tablet, or computer.

**Business chat service**

* This is the service that provides the chat functionality, such as routing messages, storing chat history, and providing analytics.

**Business agent**

* This is the person who responds to customer messages.

The diagram also shows the flow of messages between the three components. When a customer sends a message, it is routed to the business chat service. The business chat service then stores the message and delivers it to the business agent. The business agent then responds to the message, and the response is routed back to the client device.

**DATA STRUCTURES**

* **ARRAY**

An array in C is a fundamental data structure that allows you to store a collection of elements of the same data type under a single name. Arrays provide a convenient way to work with multiple values of the same type, such as integers, characters, or floating-point numbers. Arrays are essential in C for tasks like data storage, manipulation, and efficient memory management.

* **QUEUE**

A queue in C is a linear data structure that follows the "First-In-First-Out" (FIFO) principle. It's similar to a real-world queue or line, where the first person to join the queue is the first one to be served. In C, queues are typically implemented using arrays or linked lists.

**Two Fundamental Operations:**

**enqueue:** To add an element to the back of the queue.

**dequeue:** To remove an element from the front of the queue.

**FLOW CHART**

Start

|

v

+-----------------------+

| Choose an Operation |

|-----------------------|

| 1. Add Customer |

| 2. Process Customer |

| 3. Exit |

+-----------------------+

|

| (User Input)

v

+-----------------------+

| Add Customer |

|-----------------------|

| Enter Customer ID |

| Enter Customer Name |

| Enqueue Customer |

| Display Success |

+-----------------------+

| |

v |

+-----------------------+

| Process Customer |

|-----------------------|

| Dequeue Customer |

| Display Customer ID |

| Display Customer Name|

+-----------------------+

| |

v |

+-----------------------+

| Exit |

|-----------------------|

| End Program |

+-----------------------+

|

v

End

**MAIN CODE**

#include <stdio.h>

#include <stdlib.h>

// Define the structure for customer information

typedef struct customer {

  int customer\_id;

  char name[50];

  char contact\_number[10];

  char purchase\_history[100];

  struct customer \*next;

} customer;

// Define the front and rear pointers for the queue

customer \*front = NULL;

customer \*rear = NULL;

// Function to enqueue a customer

void enqueue(customer \*new\_customer) {

  if (front == NULL) {

    front = new\_customer;

    rear = new\_customer;

  } else {

    rear->next = new\_customer;

    rear = new\_customer;

  }

}

// Function to dequeue a customer

customer \*dequeue() {

  if (front == NULL) {

    return NULL;

  } else {

    customer \*temp = front;

    front = front->next;

    return temp;

  }

}

// Function to display the customer information for a given customer ID

void display\_customer\_info(customer \*cust) {

  printf("Customer Name: %s\n", cust->name);

  printf("Customer Contact Number: %s\n", cust->contact\_number);

  printf("Customer Purchase History: %s\n", cust->purchase\_history);

}

// Function to display the customer information for a given customer ID

void display\_customer\_info\_by\_id(int customer\_id) {

  customer \*temp = front;

  while (temp != NULL) {

    if (temp->customer\_id == customer\_id) {

      display\_customer\_info(temp);

      return;

    }

    temp = temp->next;

  }

  printf("Customer with ID %d does not exist.\n", customer\_id);

}

// Function to update the customer information for a given customer ID

void update\_customer\_info\_by\_id(int customer\_id) {

  customer \*temp = front;

  while (temp != NULL) {

    if (temp->customer\_id == customer\_id) {

      printf("Enter the new customer name: ");

      scanf("%s", temp->name);

      printf("Enter the new customer contact number: ");

      scanf("%s", temp->contact\_number);

      printf("Enter the new customer purchase history: ");

      scanf("%s", temp->purchase\_history);

      return;

    }

    temp = temp->next;

  }

  printf("Customer with ID %d does not exist.\n", customer\_id);

}

// Function to display all customers in the queue in order

void display\_all\_customers() {

  customer \*temp = front;

  while (temp != NULL) {

    printf("Customer ID: %d\n", temp->customer\_id);

    display\_customer\_info(temp);

    temp = temp->next;

  }

}

// Main function

int main() {

    int choice, customer\_id;

    customer \*new\_customer;

    customer \*dequeued\_customer;

    while (1) {

        printf("1. Enqueue a new customer\n");

        printf("2. Dequeue a customer\n");

        printf("3. Display customer information by ID\n");

        printf("4. Update customer information by ID\n");

        printf("5. Display all customers in order\n");

        printf("6. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                // Enqueue a new customer

                new\_customer = (customer \*)calloc(1, sizeof(customer));

                printf("Enter the customer ID: ");

                scanf("%d", &new\_customer->customer\_id);

                printf("Enter the customer name: ");

                scanf("%s", new\_customer->name);

                printf("Enter the customer contact number: ");

                scanf("%s", new\_customer->contact\_number);

                printf("Enter the customer purchase history: ");

                scanf("%s", new\_customer->purchase\_history);

                enqueue(new\_customer);

                printf("Customer successfully enqueued.\n");

                break;

            case 2:

                // Dequeue a customer

                dequeued\_customer = dequeue();

                if (dequeued\_customer != NULL) {

                    printf("Customer ID: %d\n", dequeued\_customer->customer\_id);

                    display\_customer\_info(dequeued\_customer);

                    free(dequeued\_customer);

                } else {

                    printf("The queue is empty.\n");

                }

                break;

            case 3:

                // Display customer information by ID

                printf("Enter the customer ID: ");

                scanf("%d", &customer\_id);

                display\_customer\_info\_by\_id(customer\_id);

                break;

            case 4:

                // Update customer information by ID

                printf("Enter the customer ID: ");

                scanf("%d", &customer\_id);

                update\_customer\_info\_by\_id(customer\_id);

                break;

            case 5:

                // Display all customers in order

                display\_all\_customers();

                break;

            case 6:

                // Exit

                exit(0);

                break;

            default:

                printf("Invalid choice.\n");

                break;

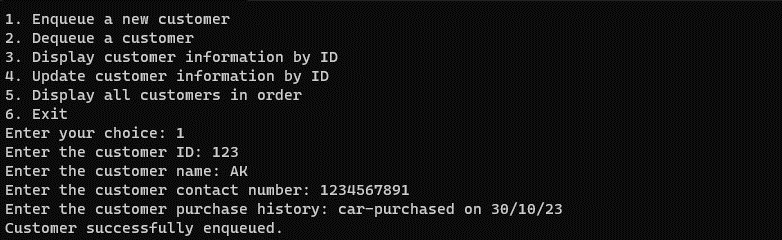
        }

    }

    return 0;

}

**OUTPUT**

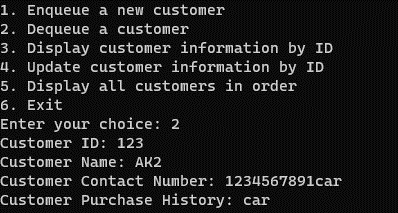


A screenshot of a computer

Description automatically generated

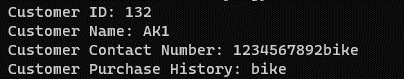
A screenshot of a computer

Description automatically generated



A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**CONCLUSION**

In conclusion, the customer management system program using C with a queue data structure provides a simple yet effective way to manage a queue of customers. The program allows users to perform two primary operations: adding customers to the queue and processing them in a first-come, first-served manner.