AIM

The aim of this program is to simulate a **Customer Service Desk System** using a **Deque (Double-Ended Queue)** to manage customer complaints efficiently. The system allows customers to submit complaints (either product-related or service-related) and allows employees to process these complaints from either end of the queue, mimicking a real-world customer service operation.

FUNCTIONALITIES

1. User Management:

- **Login**:
    a. Employees and customers can log in by entering their unique ID and name.
    b. If the ID starts with "EMP-", the user is considered an employee; otherwise, the user is treated as a customer.
    c. The system checks if the entered ID and name match an existing account in the linked list of employees or customers.

- **Sign Up**:
    a. New users (both employees and customers) can sign up by entering their name.
    b. Employees get an ID starting with "EMP-" followed by a unique number (e.g., `EMP-1000`).
    c. Customers get a numeric ID starting from 1000 (e.g., `1000`).
    d. The newly created user is added to the respective linked list (either employee list or customer list).

2. Complaint Management:

- **Create Complaint**:
    - A complaint is created by a customer and linked to the customer's ID and name.

    - Complaints are stored in the `CustomerComplaint` structure, which contains:
        - Customer ID
        - Customer name
        - The complaint text
        - Pointers to the next and previous complaints in a doubly linked list (for deque functionality).
- **Add Complaints**:
    - Customers can add complaints in two categories:
        - **Product-related complaints**: Added to the **front** of the deque.
        - **Service-related complaints**: Added to the **rear** of the deque.
    - Complaints are added using the `ProductComplaints` and `ServiceComplaint` functions respectively.
- **Display Complaints**:
    - Customers can view all complaints associated with their ID. The system searches the deque for complaints with a matching customer ID and prints them.
    - If no complaints are found for the given ID, a message is shown indicating that no complaints exist for that customer.

3. Complaint Processing (by Employees):

- **Process Complaints**:
    - Employees can process complaints either:
        - **From the front of the deque** (LIFO - Last In, First Out) using the `processComplaintFront` function.
        - **From the rear of the deque** (LIFO - Last In, First Out) using the `processComplaintRear` function.
    - When a complaint is processed, it is removed from the deque, and the memory is freed for that complaint.

4. Complaint Queue Management (Deque):

- The deque is implemented using a **doubly linked list**:

    - Each complaint node (`CustomerComplaint`) has pointers to both the **next** and **previous** nodes, which allows complaints to be added or removed from both the front and rear.
- **Add Complaints at Front**: Complaints can be added at the front using the `ProductComplaints` function.
- **Add Complaints at Rear**: Complaints can be added at the rear using the `ServiceComplaint` function.
- **Remove Complaints from Front**: Complaints can be processed and removed from the front using the `processComplaintFront` function.
- **Remove Complaints from Rear**: Complaints can be processed and removed from the rear using the `processComplaintRear` function.

5. Search for Users (Employee or Customer):

- The system allows searching for employees or customers in the respective linked lists using their ID and name.
- If a match is found, the user is allowed to perform the actions specific to their role (either employee or customer).
- If no match is found, the user is notified that the ID and name do not match any existing records.

6. Main Menu Options:

- **Login**:
    - Allows users (employees or customers) to log in by entering their ID and name.
    - Based on the type of ID (employee or customer), the system displays the respective menu options for employees or customers.
- **Sign Up**:
    - Allows users to sign up by providing their name and choosing whether they are an employee or a customer.
    - New employees are given an ID starting with "EMP-" and new customers are given numeric IDs starting from `1000`.
- **Exit**:
    - Exits the program.

7. Employee Functionalities:

- Employees can:
    - **Process Complaints**: Employees can process complaints either from the front or rear of the deque.
    - **Search Customer Complaints by ID**: Employees can search for complaints associated with a specific customer ID and display them.

8. Customer Functionalities:

- Customers can:
    - **Add Complaints**: Customers can add product-related or service-related complaints.
    - **View Their Complaints**: Customers can view all complaints associated with their ID.
    - **Logout**: Customers can log out from the system.

9. Complaint Structure:

- The complaint is represented by the `CustomerComplaint` structure which includes:
    - `id`: Customer's unique ID.
    - `name`: Customer's name.
    - `complaint`: The complaint text.
    - `next`: Pointer to the next complaint in the list.
    - `prev`: Pointer to the previous complaint in the list (used for deque functionality).

10. Memory Management:

- Dynamically allocated memory for users (`User`) and complaints (`CustomerComplaint`) is freed when complaints are processed.

ALGORITHM

Step 1: Define Structures

1. **User Structure**:
   1.1. `id`: User's unique ID
   1.2. `name`: User's name
   1.3. `next`: Pointer to next user in the linked list
2. **CustomerComplaint Structure**:
   2.1. `id`: Customer's ID
   2.2. `name`: Customer's name
   2.3. `complaint`: Complaint details
   2.4. `next`: Pointer to the next complaint in the deque
   2.5. `prev`: Pointer to the previous complaint in the deque

Step 2: Main Menu

1. **Display Main Menu**:

   Options:

   1.1. Login
   1.2. Sign Up
   1.3. Exit
2. **User Input**: User selects an option.

Step 3: User Login (if option 1 selected)

1. **Prompt for ID and Name**.
2. **Identify Role**:
   2.1. If ID starts with "EMP-", user is an **Employee**.
   2.2. If ID is numeric, user is a **Customer**.
3. **Search for User**:
   3.1. Search the linked list (either `emplisthead` for employees or `custlisthead` for customers) for a matching `id` and `name`.
   3.2. If a match is found:
      3.2.1. Allow user to proceed to respective menu (Employee or Customer).
   3.3. If no match is found:
      3.3.1. Display "Invalid ID or Name" message.

Step 4: Employee Menu (if Employee logs in)

1. **Display Employee Options**:
    - 1.1.4. Process Complaint from Front
    - 1.1.5. Process Complaint from Rear
    - 1.1.6. Search Complaints by Customer ID
    - 1.1.7. Logout
2. **Employee Input**: Select action based on menu options.
3. **Process Complaint**:
    - 3.1. If **option 1 (Front)** selected:
        - 3.1.4. Process complaint from the front of the deque (`front` pointer).
    - 3.2. If **option 2 (Rear)** selected:
        - 3.2.4. Process complaint from the rear of the deque (`rear` pointer).
    - 3.3. If **option 3 (Search Complaints)** selected:
        - 3.3.4. Search for complaints associated with a specific customer ID.
    - 3.4. If **option 4 (Logout)** selected:
        - 3.4.4. Log out and return to main menu.

Step 5: Customer Menu (if Customer logs in)

1. **Display Customer Options**:
    - 1.1.4. Add Product-related Complaints
    - 1.1.5. Add Service-related Complaints
    - 1.1.6. Display Your Complaints
    - 1.1.7. Logout
2. **Customer Input**: Select action based on menu options.
3. **Add Complaint**:
    - 3.1. If **option 1 (Product-related)** selected:
        - 3.1.4. Create a new complaint and add it to the front of the deque.
    - 3.2. If **option 2 (Service-related)** selected:
        - 3.2.4. Create a new complaint and add it to the rear of the deque.
4. **Display Complaints**:
    - 4.1. If **option 3 (View Complaints)** selected:
        - 4.1.4. Display all complaints associated with the customer's ID.
5. **Logout**:
    - 5.1. If **option 4 (Logout)** selected:
        - 5.1.4. Log out and return to the main menu.

Step 6: User Sign Up (if option 2 selected)

1. **Prompt for Role**:
    - 1.1. Ask user if they are an **Employee** or **Customer**.

2. **Get User Details**:
    2.1. Prompt for user’s name.
3. **Assign ID**:
    3.1. If user is an **Employee**:
        3.1.1. Generate an ID starting with `EMP-` followed by a unique number.
    3.2. If user is a **Customer**:
        3.2.1. Generate an ID starting from `1000` and increment for each new customer.
4. **Add User**:
    4.1. Add the new user to the respective linked list (`emplisthead` or `custlisthead`).

Step 7: Complaint Management

Step 7.1: Create a New Complaint

1. **Create Complaint Node**:
    1.1. Accept customer `id`, `name`, and `complaint` text.
    1.2. Create a new `CustomerComplaint` structure and store the complaint details.
2. **Add Complaint**:
    2.1. If the complaint is **Product-related**, add it to the **front** of the deque using the `ProductComplaints` function.
    2.2. If the complaint is **Service-related**, add it to the **rear** of the deque using the `ServiceComplaint` function.

Step 7.2: Process Complaints

3. **Process Complaint from Front**:
    3.1. Check if the deque is empty. If not, process the complaint at the **front** and remove it from the deque.
4. **Process Complaint from Rear**:
    4.1. Check if the deque is empty. If not, process the complaint at the **rear** and remove it from the deque.

Step 8: Display Complaints for a Customer

1. **Search for Complaints by Customer ID**:
    1.1. Traverse the deque from `front` to `rear`.

    1.2. For each complaint, check if the customer ID matches.
    1.3. Display all complaints for that customer.

Step 9: Exit

1. **Exit Program**:
    1.1. Display exit message and terminate the program.

EXPECTED OUTPUT

Sign Up and Login as a Customer to Add Complaint

```
--- Main Menu ---

1. Login
2. Sign Up
3. Exit
Enter your choice: 2
Are you an Employee? (Y/N): N
Enter your name: John Doe
Account created successfully with ID: 1000

--- Main Menu ---

1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: 1000
Enter your name: John Doe

Welcome, John Doe. (Customer)
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 1
Enter your complaint: Product not working as expected.
Complaint added to the front.
```

Employee Processing Complaint

```
--- Main Menu ---

1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: EMP-1000
Enter your name: Jane Smith
Welcome, Jane Smith. (Employee)
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 1
Processing complaint for ID 1000: Product not working as expected.
```

Displaying Complaints for a Customer

```
--- Main Menu ---

1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: 1000
Enter your name: John Doe

Welcome, John Doe. (Customer)
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 3
Complaints for Customer ID 1000:
Complaint: Product not working as expected.
```

Exiting the Program

```
--- Main Menu ---

1. Login
2. Sign Up
3. Exit
Enter your choice: 3
Exiting the program.
```

PROGRAM

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

int lastEmployeeID = 1000;
int lastCustomerID = 1000;

typedef struct User {
   char id[9];
   char name[50];
   struct User *next;
} User;

typedef struct CustomerComplaint {
   char id[9];
   char name[50];
   char complaint[200];
   struct CustomerComplaint *next;
   struct CustomerComplaint *prev;
} CustomerComplaint;

User *emplisthead = NULL;
User *custlisthead = NULL;

CustomerComplaint *front = NULL;
CustomerComplaint *rear = NULL;

User* createUser(char *id, char *name) {
   User *newUser = (User *)malloc(sizeof(User));
   strcpy(newUser->id, id);
   strcpy(newUser->name, name);
   newUser->next = NULL;
   return newUser;
}


User* addUser(User *list, char *id, char *name) {
   User *newUser = createUser(id, name);
   newUser->next = list;
   list = newUser;
```

```
    printf("Account created successfully with ID: %s\n", id);
    return list;}

CustomerComplaint* createComplaintNode(char *id, char *name, char *complaint) {
    CustomerComplaint *newComplaint = (CustomerComplaint
*)malloc(sizeof(CustomerComplaint));
    strcpy(newComplaint->id, id);
    strcpy(newComplaint->name, name);
    strcpy(newComplaint->complaint, complaint);
    newComplaint->next = NULL;
    newComplaint->prev = NULL;
    return newComplaint;
}

void ProductComplaints(CustomerComplaint *newComplaint) {
    if (front == NULL) {
        front = rear = newComplaint;
    } else {
        newComplaint->next = front;
        front->prev = newComplaint;
        front = newComplaint;
    }
    printf("Complaint added to the front.\n\n");
}

void ServiceComplaint(CustomerComplaint *newComplaint) {
    if (rear == NULL) {
        front = rear = newComplaint;
    } else {
        newComplaint->prev = rear;
        rear->next = newComplaint;
        rear = newComplaint;
    }
    printf("Complaint added to the rear.\n\n");
}
void displayComplaints(char *id) {
    CustomerComplaint *current = front;
    int found = 0;
    printf("\nComplaints for Customer ID %s:\n", id);
    while (current != NULL) {
        if (strcmp(current->id, id) == 0) {
            printf("Complaint: %s\n", current->complaint);
            found = 1;
        }
        current = current->next;
    }
    printf("\n");
    if (!found) printf("No complaints found for this customer ID.\n\n");
}

void processComplaintFront() {
    if (front == NULL) {
        printf("No complaints to process.\n\n");
        return;
    }
    CustomerComplaint *processed = front;
    printf("Processing complaint for ID %s: %s\n\n", processed->id,
processed->complaint);
    front = front->next;
    if (front) front->prev = NULL;
    else rear = NULL;
    free(processed);
}

void processComplaintRear() {
    if (rear == NULL) {
        printf("No complaints to process.\n\n");
        return;
    }
    CustomerComplaint *processed = rear;
    printf("Processing complaint for ID %s: %s\n\n", processed->id,
processed->complaint);
    rear = rear->prev;
    if (rear) rear->next = NULL;
    else front = NULL;
    free(processed);
}

User* searchUser(User *head, char *id, char *name) {
    User *current = head;
    while (current != NULL) {
        if (strcmp(current->id, id) == 0 && strcmp(current->name, name) == 0) {
            return current;
        }
        current = current->next;
    }
    return NULL;
}
```

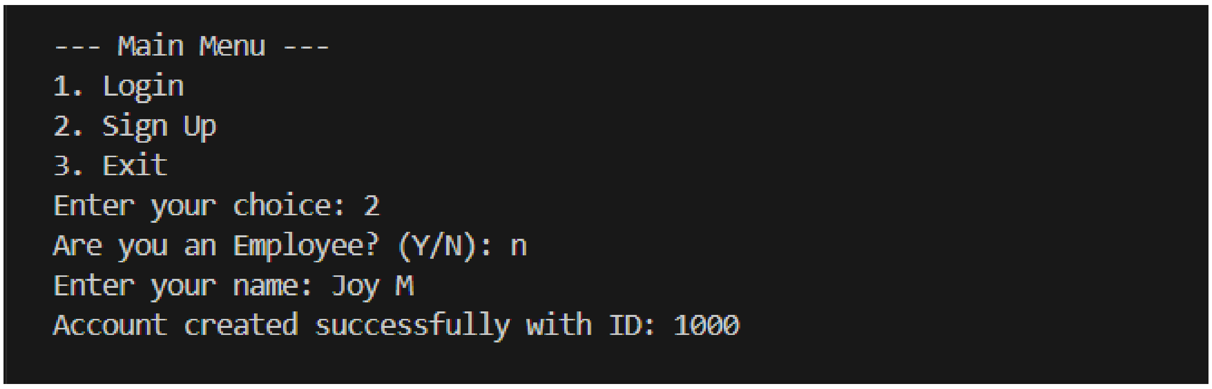
```
int main() {
    int choice;
    char id[9], name[50], complaint[200];
    char ch;
    User *user;

    while (1) {
        printf("\n--- Main Menu ---\n");
        printf("1. Login\n");
        printf("2. Sign Up\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {              printf("Enter your ID: ");
            scanf("%s", id);
            printf("Enter your name: ");
            scanf(" %[^\n]", name);
            if (strncmp(id, "EMP-", 4) == 0) {                 user = searchUser(emplisthead,
id, name);
                if (user) {
                    printf("Welcome, %s. (Employee)\n", user->name);
                    while (1) {
                        printf("1. Process Complaint from Front\n");
                        printf("2. Process Complaint from Rear\n");
                        printf("3. Search Customer Complaints by ID\n");
                        printf("4. Logout\n");
                        printf("Enter your choice: ");
                        scanf("%d", &choice);

                        if (choice == 1) processComplaintFront();
                        else if (choice == 2) processComplaintRear();
                        else if (choice == 3) {
                            printf("Enter Customer ID to search complaints: ");
                            scanf("%s", id);
                            displayComplaints(id);
                        } else if (choice == 4) {
                            printf("Logging out.\n");
                            break;
                        } else {
                            printf("Invalid choice. Try again.\n");
                        }
                    }
                } else {
                    printf("Employee ID and Name do not match.\n");
                }
            } else {                  user = searchUser(custlisthead, id, name);
                if (user) {
                    printf("Welcome, %s. (Customer)\n", user->name);
                    while(1){
                               printf("1. Add Product related Complaints\n");
                               printf("2. Add Service related Complaints\n");
                               printf("3. Display Your Complaints\n");
                               printf("4. Logout\n");
                               printf("Enter your choice: ");
                               scanf("%d", &choice);

                               if (choice == 1 || choice == 2) {
                                   printf("Enter your complaint: ");
                                   scanf(" %[^\n]", complaint);
                                   CustomerComplaint *newComplaint =
           createComplaintNode(id, user->name, complaint);
                                   if (choice == 1) ProductComplaints(newComplaint);
                                   else ServiceComplaint(newComplaint);
                               } else if (choice == 3) {
                                   displayComplaints(id);
                               } else if (choice == 4) {
                                   printf("Logging out.\n");
                                   break;
                               } else {
                                   printf("Invalid choice. Try again.\n");
                               }
                     }
                } else {
                    printf("Customer ID and Name do not match.\n");
                }
            }

        } else if (choice == 2) {             printf("Are you an Employee? (Y/N): ");
            scanf(" %c", &ch);
            getchar();

            printf("Enter your name: ");
            scanf(" %[^\n]", name);

            if (ch == 'Y' || ch == 'y') {
                sprintf(id, "EMP-%d", lastEmployeeID++);
```

```
            emplisthead = addUser(emplisthead, id, name);           } else {
            sprintf(id, "%d", lastCustomerID++);
            custlisthead = addUser(custlisthead, id, name);          }

        } else if (choice == 3) {
            printf("Exiting the program.\n");
            break;
        } else {
            printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

OUTPUTS

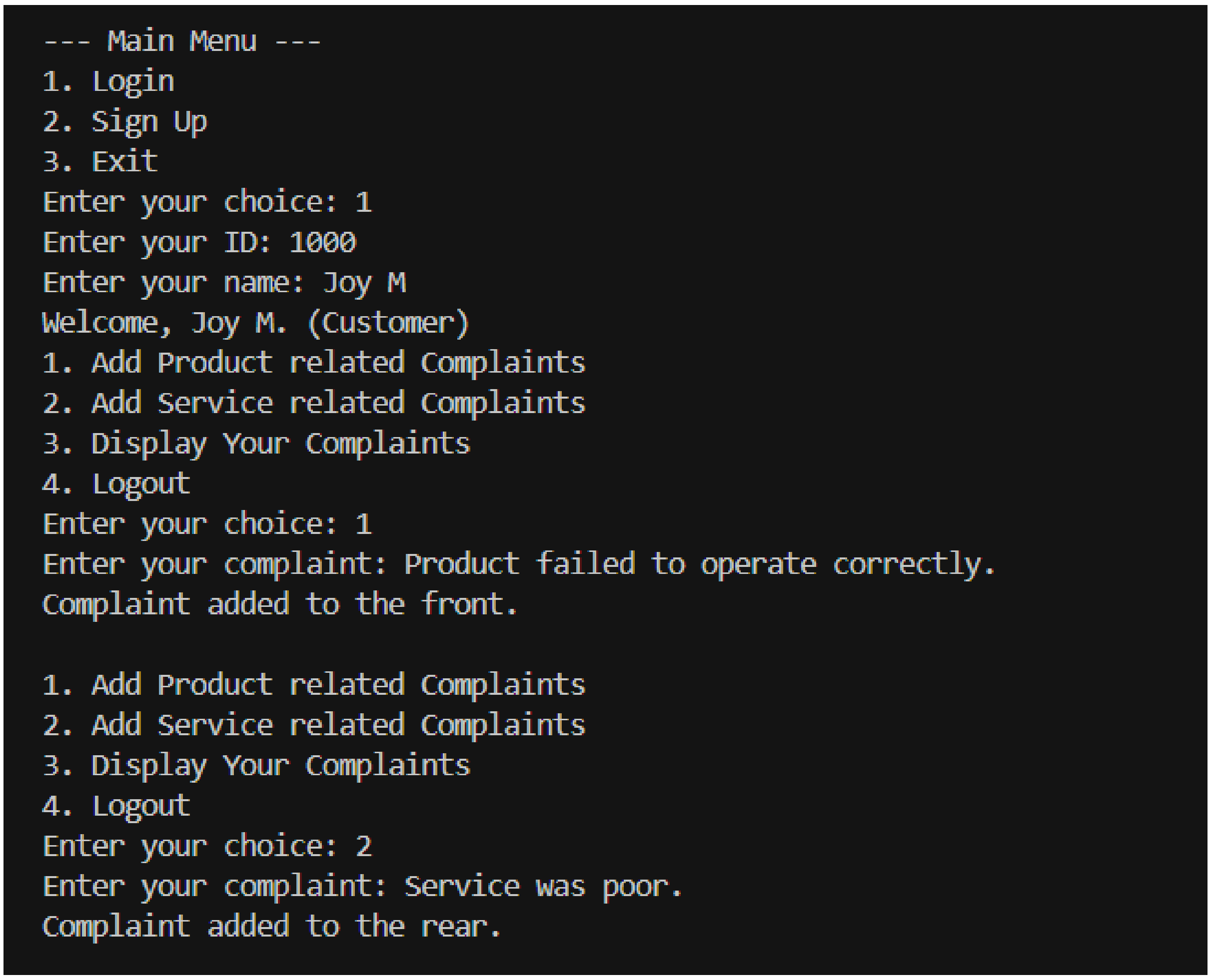
Customer

Customer signup(ID 1000)



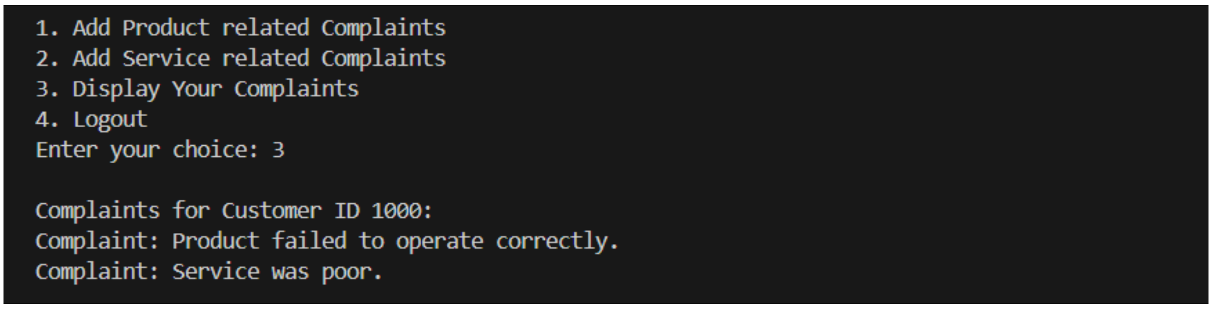
```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 2
Are you an Employee? (Y/N): n
Enter your name: Joy M
Account created successfully with ID: 1000
```

Login and Adding complaint



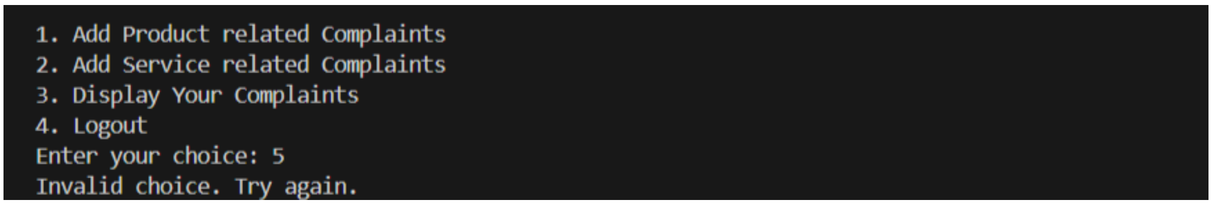
```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: 1000
Enter your name: Joy M
Welcome, Joy M. (Customer)
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 1
Enter your complaint: Product failed to operate correctly.
Complaint added to the front.

1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 2
Enter your complaint: Service was poor.
Complaint added to the rear.
```

Displaying complaints



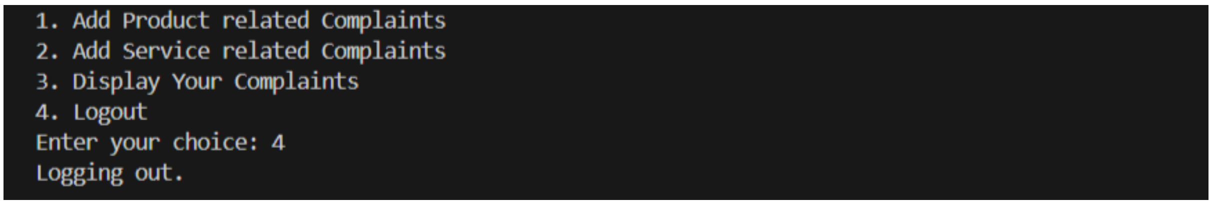
```
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 3

Complaints for Customer ID 1000:
Complaint: Product failed to operate correctly.
Complaint: Service was poor.
```

Invalid option checking



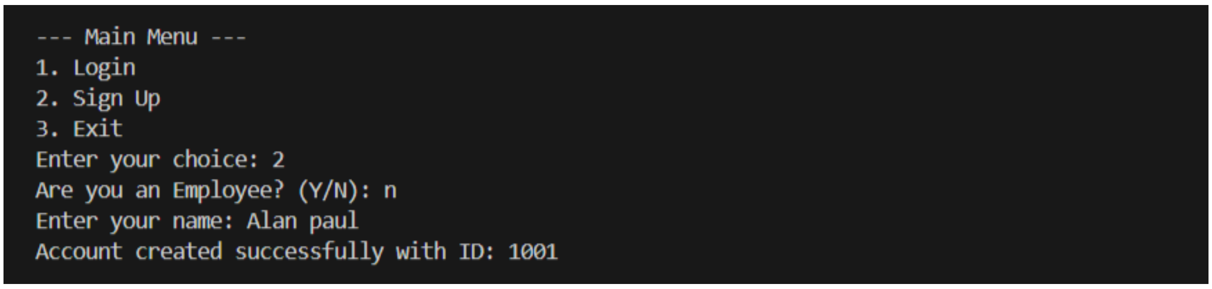
```
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 5
Invalid choice. Try again.
```

Logging out



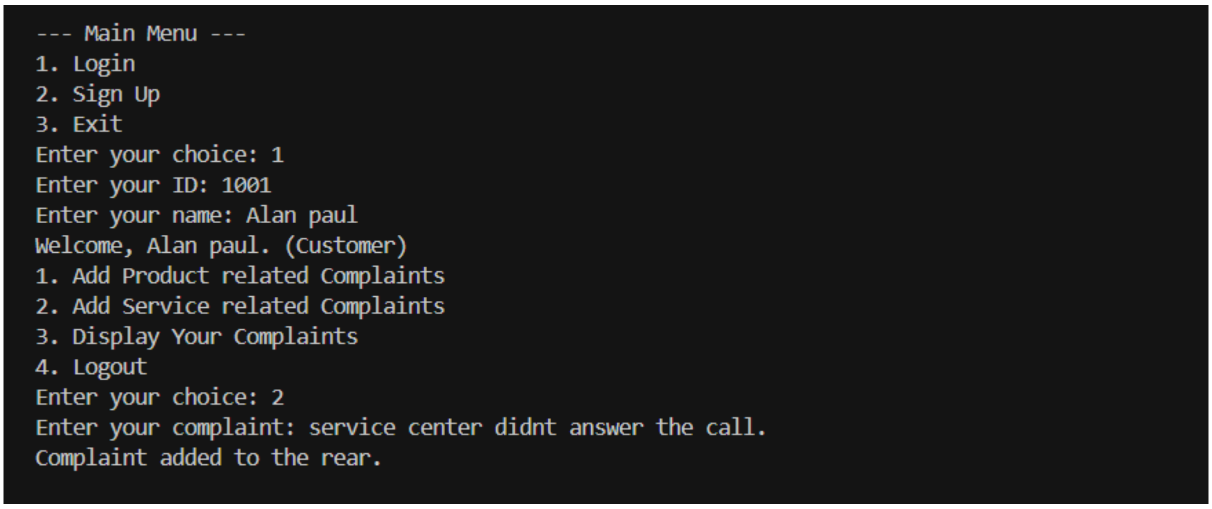
```
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 4
Logging out.
```

New customer sign up(ID 1001)



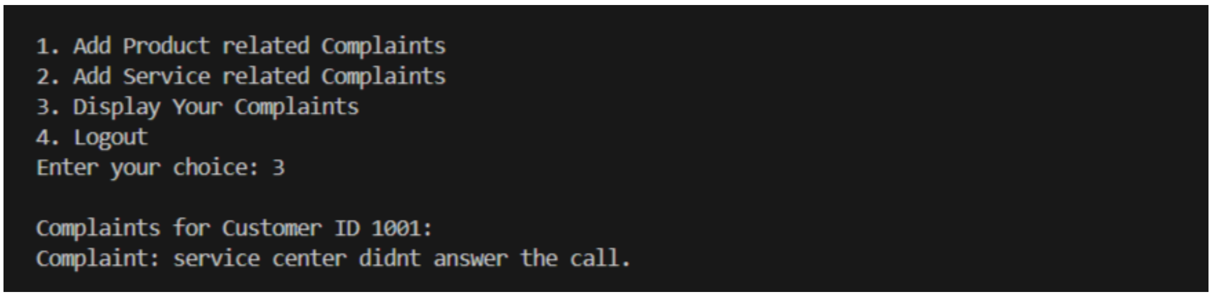
```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 2
Are you an Employee? (Y/N): n
Enter your name: Alan paul
Account created successfully with ID: 1001
```

Login and Adding complaint



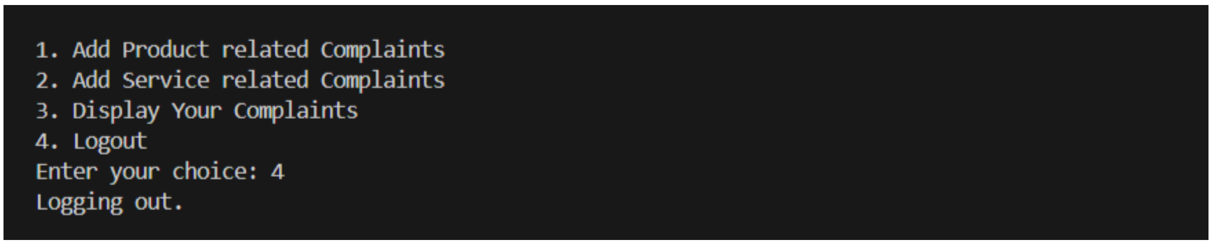
```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: 1001
Enter your name: Alan paul
Welcome, Alan paul. (Customer)
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 2
Enter your complaint: service center didnt answer the call.
Complaint added to the rear.
```

Displaying complaint



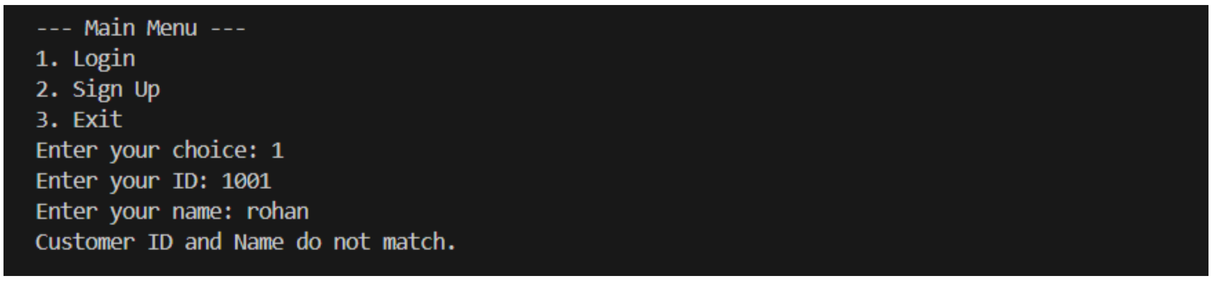
```
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 3

Complaints for Customer ID 1001:
Complaint: service center didnt answer the call.
```

Logging out



```
1. Add Product related Complaints
2. Add Service related Complaints
3. Display Your Complaints
4. Logout
Enter your choice: 4
Logging out.
```

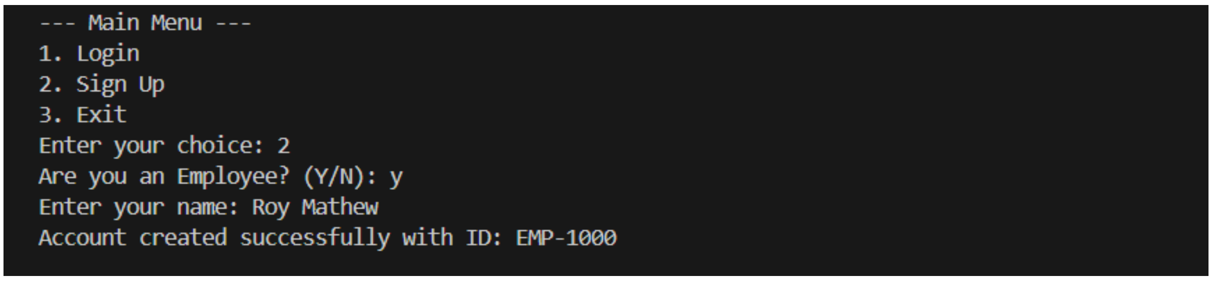
Invalid user name and ID



```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: 1001
Enter your name: rohan
Customer ID and Name do not match.
```

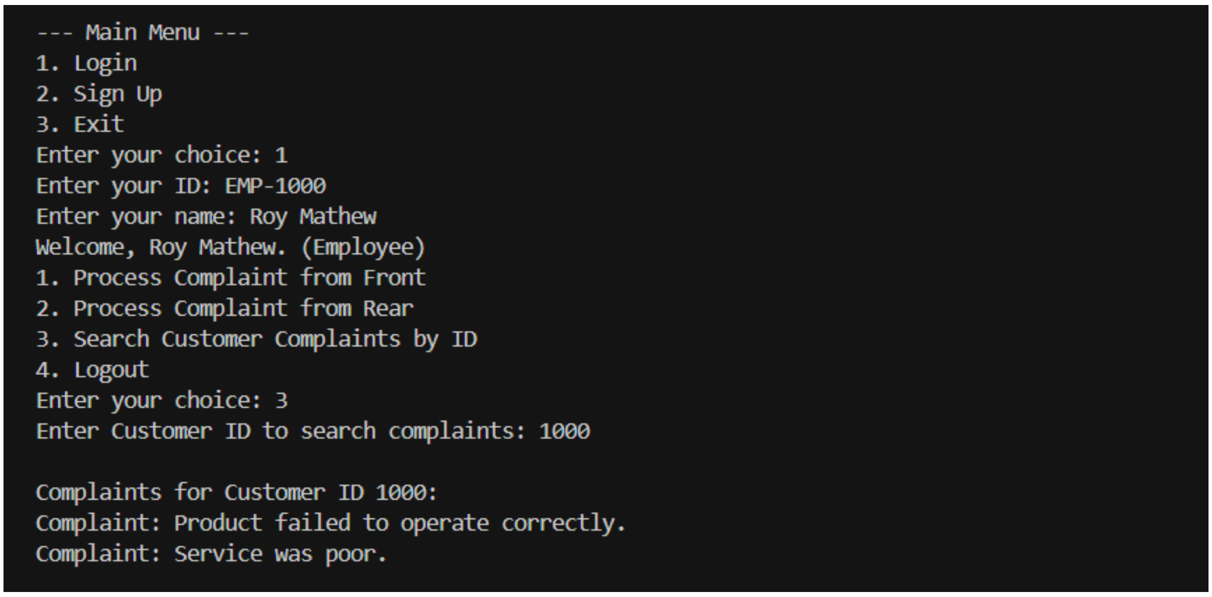
Employee

Employee sign up

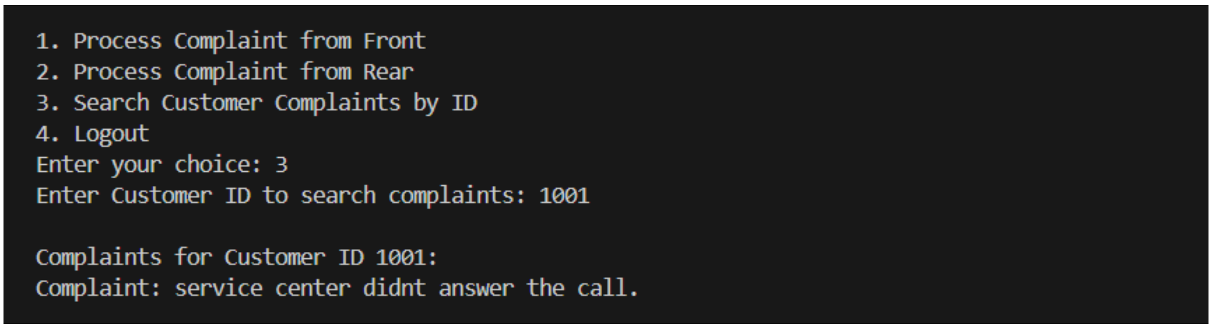


```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 2
Are you an Employee? (Y/N): y
Enter your name: Roy Mathew
Account created successfully with ID: EMP-1000
```

Login and searching complaint for a specific id

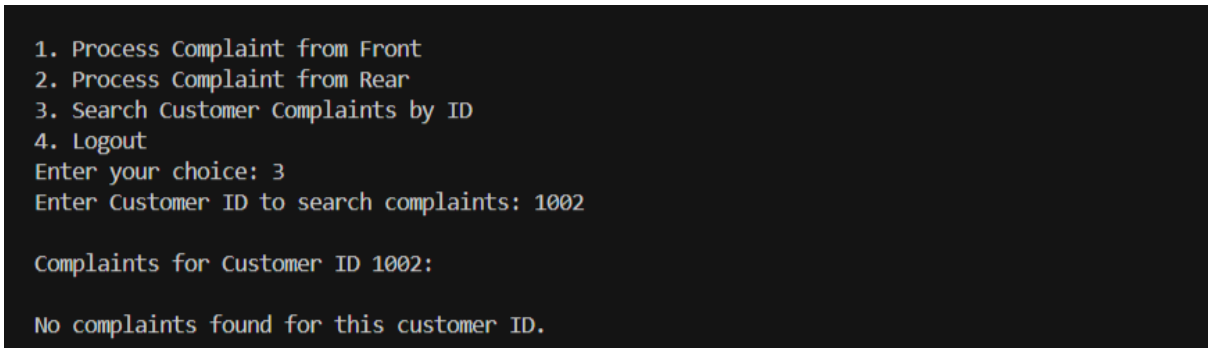


```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 1
Enter your ID: EMP-1000
Enter your name: Roy Mathew
Welcome, Roy Mathew. (Employee)
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 3
Enter Customer ID to search complaints: 1000

Complaints for Customer ID 1000:
Complaint: Product failed to operate correctly.
Complaint: Service was poor.
```



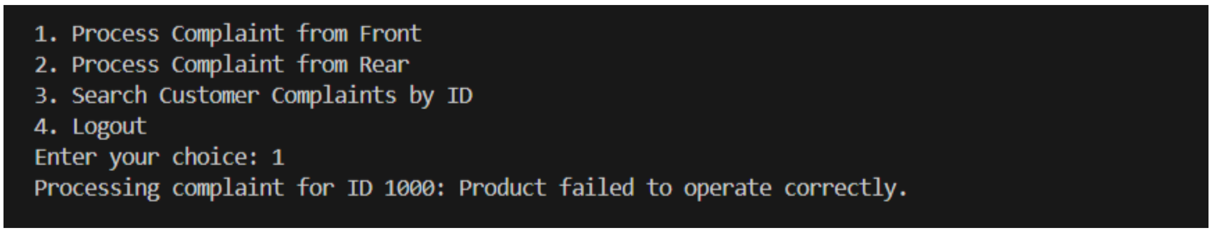
```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 3
Enter Customer ID to search complaints: 1001

Complaints for Customer ID 1001:
Complaint: service center didnt answer the call.
```

Searching Complaint for a specific ID that don’t exist

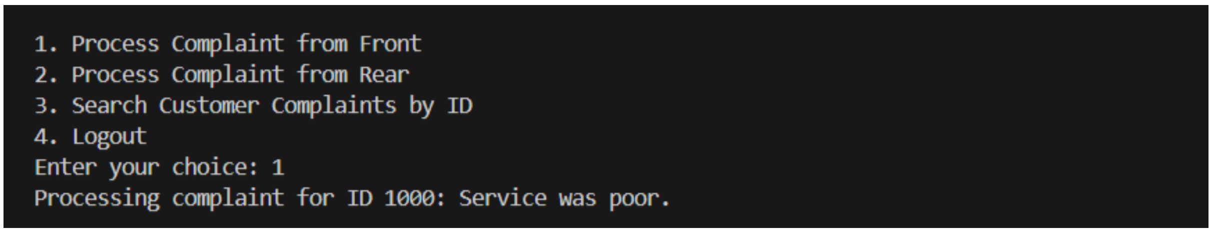


```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 3
Enter Customer ID to search complaints: 1002

Complaints for Customer ID 1002:

No complaints found for this customer ID.
```

Processing complaints(from front)

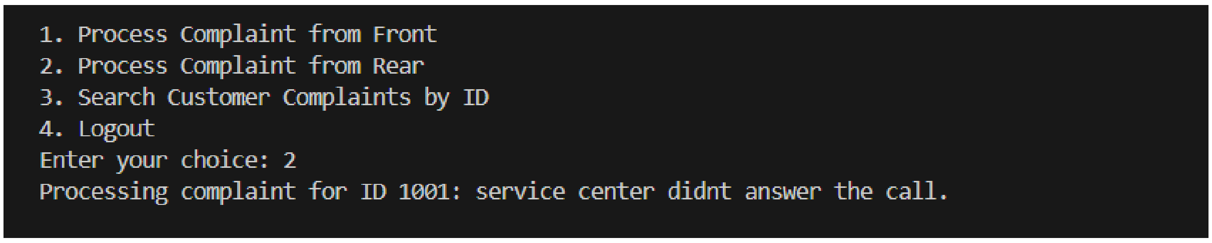


```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 1
Processing complaint for ID 1000: Product failed to operate correctly.
```



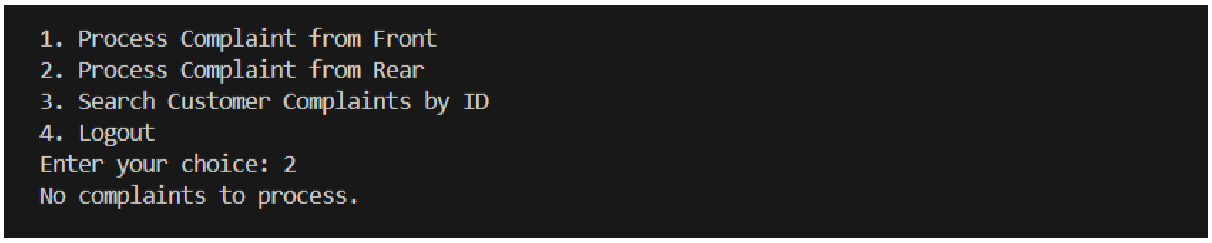
```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 1
Processing complaint for ID 1000: Service was poor.
```

Processing complaint(from rear)



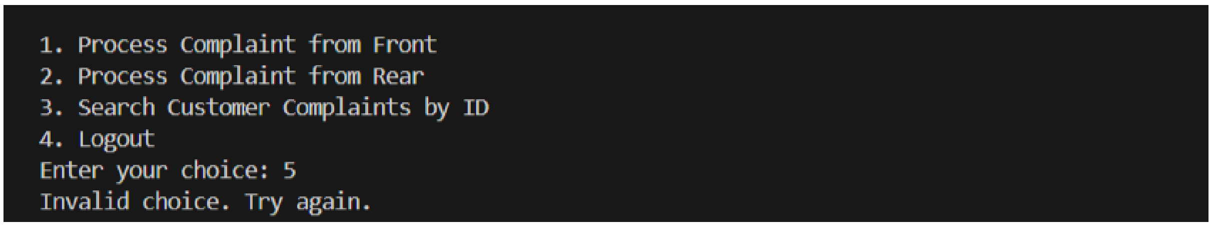
```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 2
Processing complaint for ID 1001: service center didnt answer the call.
```

Dequeue empty



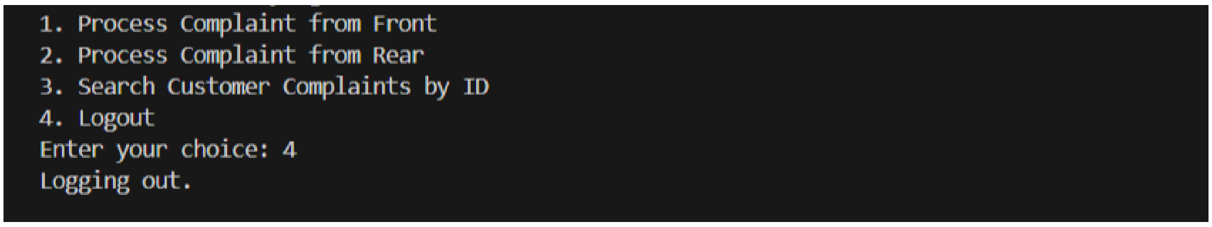
```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 2
No complaints to process.
```

Invalid choice



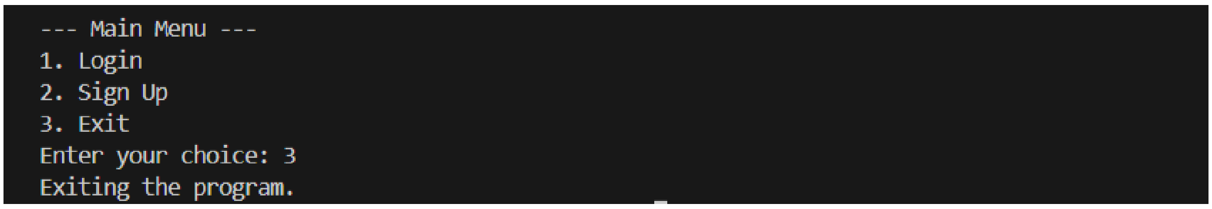
```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 5
Invalid choice. Try again.
```

Logging out



```
1. Process Complaint from Front
2. Process Complaint from Rear
3. Search Customer Complaints by ID
4. Logout
Enter your choice: 4
Logging out.
```

Exiting



```
--- Main Menu ---
1. Login
2. Sign Up
3. Exit
Enter your choice: 3
Exiting the program.
```