# MANGALORE INSTITUTE OF TECHNOLOGY &ENGINEERING

*Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified*
*Institution(A Unit of Rajalaxmi Education Trust®, Mangalore -575001)*
**Affiliated to V.T.U., Belagavi, Approved by AICTE, New Delhi.**

**MITE**

**Invent Solutions**

**TEAM** : CP050

**SECTION** : CS3

**TEAM MEMBER DETAILS** :

| NAME | USN |
|---|---|
| Sushmitha | 4MT21CS167 |
| Shwetha | 4MT21CS156 |
| Prithvi | 4MT22CS411 |
| Shobith H | 4MT21CS147 |
| Sandeep | 4MT21CS132 |
| Praveen Aladakatti | 4MT22CS410 |
| Mohammed Hussain | 4MT22CS408 |
| Pratham | 4MT22CS409 |

# TABLE OF CONTENTS

**Problem Statement**:

Search Me :

Create application to searching the given element with the help of searching algorithms

(binary search, linear search etc..)

Take the inputs from the user like in which algorithm they want to find the element and  take

the searched element .

admin : can able to do the basic CRUD operations  customer : able to do give the

inputs and by using this application they need to get  exact answer for both admin

and customer we need login page

## 1. Abstract

The project aims to create a user authentication and menu-driven system in C. It includes user login, menu handling for administrators and customers, as well as element manipulation and searching functions. This report provides an overview of the project's design, implementation, features, testing, challenges, and potential future enhancements.

## 2. Introduction

### 2.1 Background

The need for a user authentication and menu system arises in various applications, including software with different user roles and access levels. This project addresses this need by implementing a basic authentication and menu system in C.

### 2.2 Objectives

The objectives of this project are to:

- Create a user authentication system.

- Implement menu navigation for administrators and customers.

- Develop functions for adding, displaying, and searching elements in a list.

### 3. Technologies Used

- Programming Language: C

- Standard Libraries: `<stdio.h>`, `<stdlib.h>`, `<string.h>`

### 4. System Architecture

#### 4.1 Front-End

The front-end of this project is text-based and relies on user input and output through the command line.

#### 4.2 Back-End

The back-end manages user authentication, menu navigation, and element manipulation.

#### 4.3 Database

The system does not use a database as it stores user and element data in memory.

### 5. Project Modules

**5.1 Module 1**: User Authentication

- Handles user login and authentication.

**5.2 Module 2**: Menu Handling

- Manages menu navigation for administrators and customers.

**5.3 Module 3**: Element Manipulation

- Provides functions for adding, displaying, and searching elements.

### 6. Design and Implementation

#### 6.1 Front-End Design

- The front-end design is minimalistic and relies on text-based input and output.

#### 6.2 Back-End Design

- The back-end handles user login, menu selection, and element manipulation.

### 6.3 Database Design

- The system does not use a database.

### 7. Features and Functionality

### 7.1 Feature 1: User Login

- Users can log in with a username and password.

### 7.2 Feature 2: Menu Navigation

- Administrators and customers have separate menus.

### 7.3 Feature 3: Element Management

- Administrators can add elements, and customers can search for elements.

### 8. Testing

### 8.1 Unit Testing

- Each function has been tested individually.

### 8.2 Integration Testing

- The integration of functions has been tested.

### 8.3 User Acceptance Testing

- The system has been tested with user input to ensure it functions as expected.

### 9. Challenges Faced

- Challenges included handling user input and implementing search algorithms.

### 10. Future Enhancements

- Future enhancements could include data persistence with file storage and a more user-friendly interface.
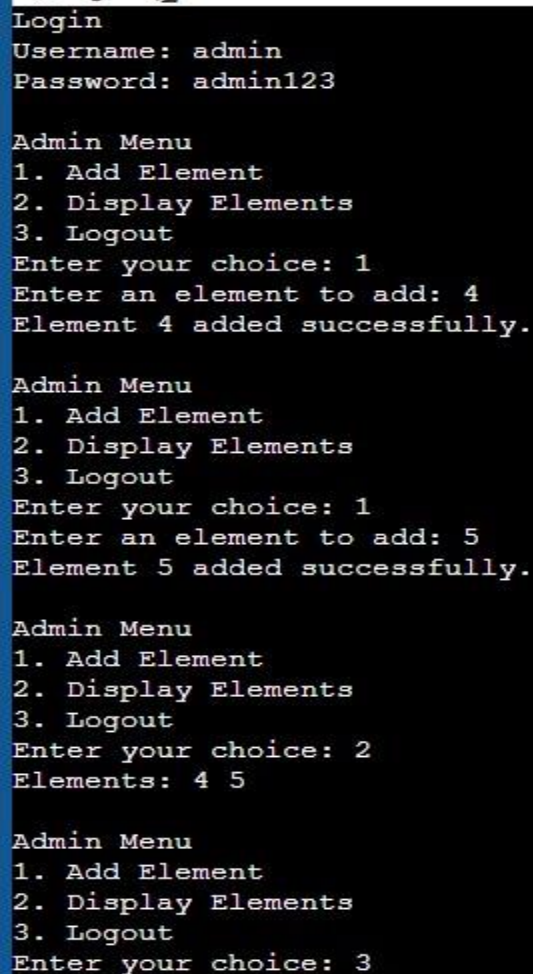
## 11. Conclusion

- The project successfully implements a basic user authentication and menu system in C, serving as a foundation for more complex applications.

## 12.References

- No external references were used for this project.

## 13. Appendices

## 13.1 Screenshots

```
Login
Username: admin
Password: admin123

Admin Menu
1. Add Element
2. Display Elements
3. Logout
Enter your choice: 1
Enter an element to add: 4
Element 4 added successfully.

Admin Menu
1. Add Element
2. Display Elements
3. Logout
Enter your choice: 1
Enter an element to add: 5
Element 5 added successfully.

Admin Menu
1. Add Element
2. Display Elements
3. Logout
Enter your choice: 2
Elements: 4 5

Admin Menu
1. Add Element
2. Display Elements
3. Logout
Enter your choice: 3
```

```
Login
Username: customer
Password: customer123

Customer Menu
1. Linear Search
2. Binary Search
3. Logout
Enter your choice: 1
Enter the element to search for: 5
Element 5 found at index 1

Customer Menu
1. Linear Search
2. Binary Search
3. Logout
Enter your choice: 2
Enter the element to search for: 4
Element 4 found at index 0

Customer Menu
1. Linear Search
2. Binary Search
3. Logout
Enter your choice: 3
Login
```

**13.2 Code Snippets**

```c
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_USERS 10

#define MAX_ELEMENTS 100


// User structure

typedef struct {

char username[20];
```

```c
    char password[20];

    int isAdmin;

} User;


// Element structure typedef

struct {    int

data[MAX_ELEMENTS];

    int size;

} ElementList;


// Function prototypes int login(User users[], int

*loggedInUser); void adminMenu(ElementList

*list); void customerMenu(ElementList *list);

void addElement(ElementList *list, int element);

void displayElements(ElementList list); int

linearSearch(ElementList list, int target); int

binarySearch(ElementList list, int target);


int main() {

    User users[MAX_USERS];

ElementList elementList;    int

loggedInUser = -1;


    // Initialize users (username, password, isAdmin)        strcpy(users[0].username, "admin");

strcpy(users[0].password, "admin123");    users[0].isAdmin = 1;
```

```c
    strcpy(users[1].username, "customer");

strcpy(users[1].password, "customer123");    users[1].isAdmin

= 0;


    // Initialize element list

elementList.size = 0;


    while (1) {       int choice = login(users,

&loggedInUser);


        if (loggedInUser == -1) {          printf("Login

failed. Please try again.\n");         continue;

    }


        if (users[loggedInUser].isAdmin) {

adminMenu(&elementList);

    } else {

        customerMenu(&elementList);

    }

  }


    return 0;

}
```

```c
int login(User users[], int *loggedInUser) {

char username[20];    char password[20];


    printf("Login\n");

printf("Username: ");

scanf("%s", username);

printf("Password: ");    scanf("%s",

password);


    for (int i = 0; i < MAX_USERS; i++) {        if (strcmp(users[i].username, username) == 0 &&

strcmp(users[i].password, password) == 0) {

        *loggedInUser = i;

return i;

    }

  }


    return -1;

}


void adminMenu(ElementList *list) {

  int choice;

int element;


  while (1) {        printf("\nAdmin

Menu\n");        printf("1. Add
```

```c
Element\n");        printf("2. Display

Elements\n");        printf("3.

Logout\n");        printf("Enter your

choice: ");        scanf("%d",

&choice);


    switch (choice) {
case 1:

        printf("Enter an element to add: ");

scanf("%d", &element);            addElement(list,

element);

        break;        case 2:

displayElements(*list);

break;        case 3:

return;        default:

        printf("Invalid choice. Try again.\n");

    }

  }

}


void customerMenu(ElementList *list) {

  int choice;

int target;    int

result;
```

```c
    while (1) {
printf("\nCustomer Menu\n");

printf("1. Linear Search\n");

printf("2. Binary Search\n");

printf("3. Logout\n");

printf("Enter your choice: ");

scanf("%d", &choice);


        switch (choice) {
case 1:

            printf("Enter the element to search for: ");

scanf("%d", &target);            result = linearSearch(*list, target);

if (result != -1) {            printf("Element %d found at index

%d\n", target, result);

            } else {

                printf("Element %d not found\n", target);

            }

break;
case 2:

            printf("Enter the element to search for: ");

scanf("%d", &target);            result = binarySearch(*list, target);

if (result != -1) {            printf("Element %d found at index

%d\n", target, result);

            } else {

                printf("Element %d not found\n", target);
```

```c
            }
        break;
        case 3:
        return;
        default:
                printf("Invalid choice. Try again.\n");
            }
        }
}


void addElement(ElementList *list, int element) {    if (list-
>size < MAX_ELEMENTS) {       list->data[list->size++] =
element;       printf("Element %d added successfully.\n",
element);
    } else {
        printf("Element list is full. Cannot add more elements.\n");
    }
}


void displayElements(ElementList list) {
printf("Elements: ");    for (int i = 0; i <
list.size; i++) {       printf("%d ",
list.data[i]);
    }
    printf("\n");
```

```
    }

    int linearSearch(ElementList list, int target) {

    for (int i = 0; i < list.size; i++) {        if

    (list.data[i] == target) {           return i;

        }

      }

      return -1;

    }


    int binarySearch(ElementList list, int target) {

       int left = 0;     int right

    = list.size - 1;       while

    (left <= right) {         int

    mid = left + (right - left)

    / 2;


        if (list.data[mid] == target) {

    return mid;

        }


        if (list.data[mid] < target) {

    left = mid + 1;        } else {

    right = mid - 1;

        }
```

```
    }


    return -1;

}
```