

**TEAM** : CP045

**SECTION** : 4CSE3

**TEAM MEMBER DETAILS** :--

**NAME USN**

**Yashaswini H D 4MT21CS189**

**Suveeksha 4MT21CS168**

**Shubhashree 4MT21CS155**

**Suma 4MT21CS165**

**Shravya Kumari 4MT22CS412**

**TABLE OF CONTENTS**

**Problem Statement**

**1.Abstract**

**2.Introduction**

**2.1 Background**

**2.2 Objectives**

**3.Technologies Used**

**4.System Architecture**

**4.1 Front-End**

**4.2 Back-End**

**4.3 Database**

**5.Project Modules**

**5.1 Module 1: User Authentication**

**5.2 Module 2: Menu Handling**

**5.3 Module 3: Element Manipulation**

**6.Design and Implementation**

**6.1 Front-End Design**

**6.2 Back-End Design**

**6.3 Database Design**

**7.Features and Functionality**

**7.1 Feature 1: User Login**

**7.2 Feature 2: Menu Navigation**

**7.3 Feature 3: Element Management**

**8.Testing**

**8.1 Unit Testing**

**8.2 Integration Testing**

**8.3 User Acceptance Testing**

**9.Challenges Faced**

**10.Future Enhancements**

**11.Conclusion**

**12.References**

**13.Appendices**

**13.1 Screenshots**

**13.2 Code Snippets**

**Problem Statement**:

Sorting Algorithms (Bubble Sort, Insertion Sort, etc.)

Take input from the user and ask the user to enter in which technique they want to sort then the application should give the output according to the requirements.

Features:

Admin can do the CRUD operation on the sorting Algorithms

User can give the input and get the output for that particular Algorithms

**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(user), as well as element manipulation and sorting 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 sorting 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(user).

**5.3 Module 3**: Element Manipulation

- Provides functions for adding, displaying, deleting, updating and sorting 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(user) have separate menus.

**7.3 Feature 3: Element Management**

- Administrators can add elements, and customers(user) can sort the elements in required algorithm.

**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 sorting algorithms.

**10. Future Enhancements**

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

**11. Conclusion**

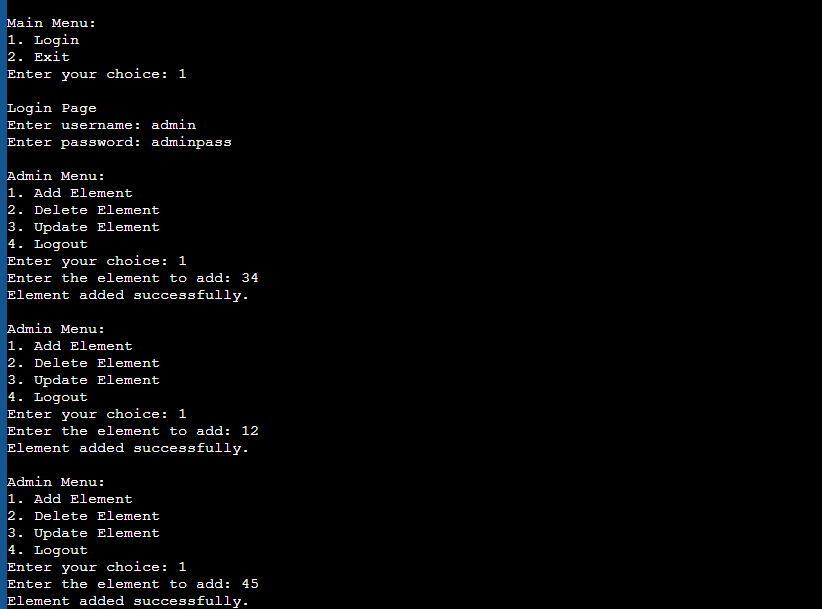
- The project successfully implements a basic user authentication and menu system in C, serving as a foundation for more complex applications.

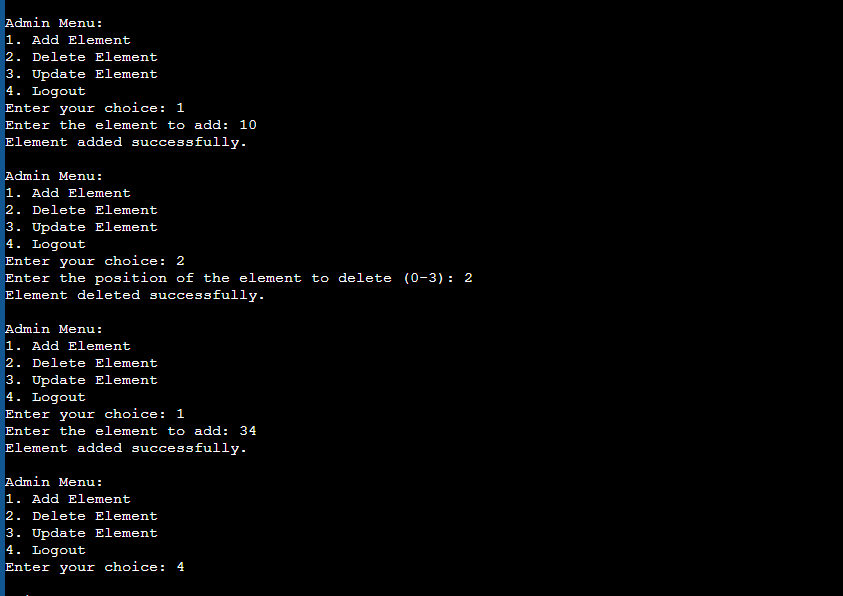
**12.References**

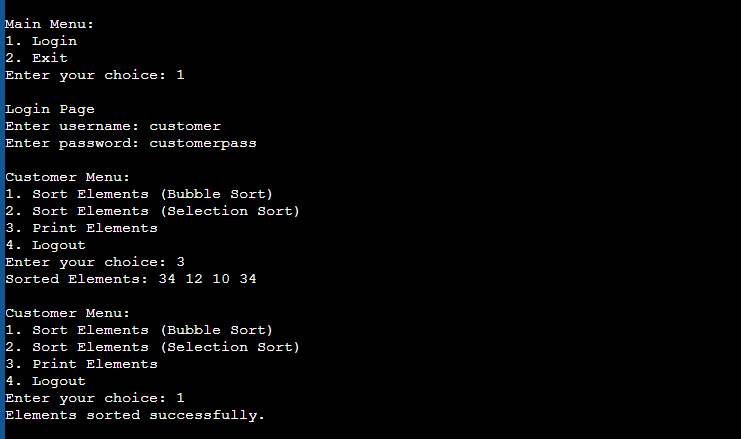
- No external references were used for this project.

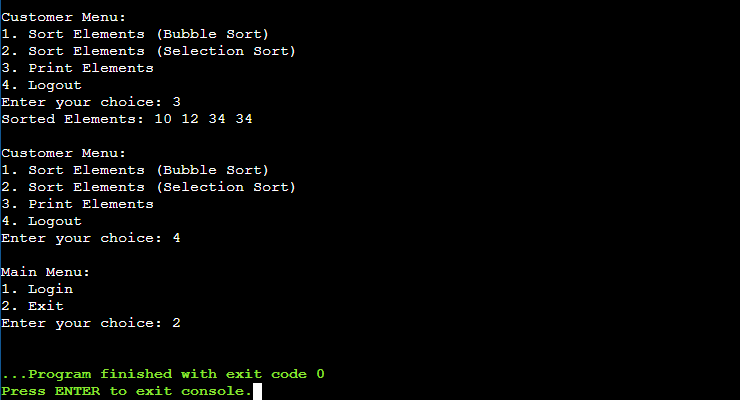
**13. Appendices**

**13.1 Screenshots**









**13.2 Code Snippets**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define constants for maximum users and elements

#define MAX\_USERS 100

#define MAX\_ELEMENTS 100

// Structure to represent user data

struct User {

char username[50];

char password[50];

int isAdmin;

};

// Structure to represent an element

struct Element {

int value;

};

// Global variables

struct User users[MAX\_USERS];

struct Element elements[MAX\_ELEMENTS];

int numUsers = 0;

int numElements = 0;

struct User \*currentUser = NULL; // To track the logged-in user

// Function prototypes

void login();

void adminMenu();

void customerMenu();

void addElement();

void deleteElement();

void updateElement();

void sortElements(int algorithm);

void printElements();

int main() {

// Sample admin and customer users (for demonstration)

strcpy(users[numUsers].username, "admin");

strcpy(users[numUsers].password, "adminpass");

users[numUsers].isAdmin = 1;

numUsers++;

strcpy(users[numUsers].username, "customer");

strcpy(users[numUsers].password, "customerpass");

users[numUsers].isAdmin = 0;

numUsers++;

int choice;

// Main application loop

while (1) {

printf("\nMain Menu:\n");

printf("1. Login\n");

printf("2. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

login();

break;

case 2:

exit(0);

default:

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

}

}

return 0;

}

void login() {

char username[50];

char password[50];

printf("\nLogin Page\n");

printf("Enter username: ");

scanf("%s", username);

printf("Enter password: ");

scanf("%s", password);

// Check if the user exists and the password is correct

for (int i = 0; i < numUsers; i++) {

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

currentUser = &users[i];

if (currentUser->isAdmin) {

adminMenu();

} else {

customerMenu();

}

return;

}

}

printf("Invalid username or password. Please try again.\n");

}

void adminMenu() {

int choice;

while (1) {

printf("\n Admin Menu:\n");

printf("1. Add Element\n");

printf("2. Delete Element\n");

printf("3. Update Element\n");

printf("4. Logout\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addElement();

break;

case 2:

deleteElement();

break;

case 3:

updateElement();

break;

case 4:

currentUser = NULL;

return;

default:

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

}

}

}

void customerMenu() {

int choice;

while (1) {

printf("\nCustomer Menu:\n");

printf("1. Sort Elements (Bubble Sort)\n");

printf("2. Sort Elements (Selection Sort)\n");

printf("3. Print Elements\n");

printf("4. Logout\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

sortElements(1); // 1 for Bubble Sort

break;

case 2:

sortElements(2); // 2 for Selection Sort

break;

case 3:

printElements();

break;

case 4:

currentUser = NULL;

return;

default:

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

}

}

}

void addElement() {

if (numElements < MAX\_ELEMENTS) {

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

scanf("%d", &elements[numElements].value);

numElements++;

printf("Element added successfully.\n");

} else {

printf("Maximum element limit reached. Cannot add more elements.\n");

}

}

void deleteElement() {

if (numElements > 0) {

int pos;

printf("Enter the position of the element to delete (0-%d): ", numElements - 1);

scanf("%d", &pos);

if (pos >= 0 && pos < numElements) {

for (int i = pos; i < numElements - 1; i++) {

elements[i] = elements[i + 1];

}

numElements--;

printf("Element deleted successfully.\n");

} else {

printf("Invalid position.\n");

}

} else {

printf("No elements to delete.\n");

}

}

void updateElement() {

if (numElements > 0) {

int pos, newValue;

printf("Enter the position of the element to update (0-%d): ", numElements - 1);

scanf("%d", &pos);

if (pos >= 0 && pos < numElements) {

printf("Enter the new value: ");

scanf("%d", &newValue);

elements[pos].value = newValue;

printf("Element updated successfully.\n");

} else {

printf("Invalid position.\n");

}

} else {

printf("No elements to update.\n");

}

}

void sortElements(int algorithm) {

int temp;

for (int i = 0; i < numElements - 1; i++) {

for (int j = 0; j < numElements - i - 1; j++) {

if (algorithm == 1) { // Bubble Sort

if (elements[j].value > elements[j + 1].value) {

// Swap elements[j] and elements[j + 1]

temp = elements[j].value;

elements[j].value = elements[j + 1].value;

elements[j + 1].value = temp;

}

} else if (algorithm == 2) { // Selection Sort

if (elements[j].value > elements[j + 1].value) {

// Swap elements[j] and elements[j + 1]

temp = elements[j].value;

elements[j].value = elements[j + 1].value;

elements[j + 1].value = temp;

}

}

}

}

printf("Elements sorted successfully.\n");

}

void printElements() {

if (numElements > 0) {

printf("Sorted Elements: ");

for (int i = 0; i < numElements; i++) {

printf("%d ", elements[i].value);

}

printf("\n");

} else {

printf("No elements to print.\n");

}

}