PROJECT REPORT

ON

PRISONER INFORMATIONS PORTAL

Submitted

By

VAKKANTI LAKSHMI REVANTH KUMAR – U202405204810354

BATCH – PGDEA 83

DEPARTMENT OF PG DIPLOMA IN VLSI DESIGN AND VERIFICATION

CRANES VARSITY

BANGALORE, KARNATAKA, INDIA - 560001

AUGUST - 2024

TABLE OF CONTENTS

ABSTRACT

The Prisoner Information Portal is a C-based console application is developed for the management of prisoner records in a jail environment. The program provides a secure and efficient way to handle prisoner data, allowing authorized users to perform operations such as adding, searching, editing, deleting, and viewing records. It uses both text and binary files for data storage, ensuring that the data is both human-readable and efficiently processable by the system. The application is built around a user-friendly, menu-driven interface that guides users through various functionalities with ease. A key feature of the application is its login system, which requires users to authenticate themselves before accessing any data. This security measure ensures that only authorized personnel can interact with the prisoner records, thereby protecting sensitive information. The login credentials are stored in a separate file and are validated before granting access. The application allows multiple login attempts, and if these fail, the program exits to prevent unauthorized access.

INTRODUCTION

The Prisoner Information Portal is a simple C-based console application designed for managing prisoner records in a jail. The application allows authorized users to add, search, edit, delete, and view prisoner details stored in both text and binary files. The main purpose of the program is to efficiently handle prisoner data through a user-friendly menu-driven interface. Each record includes comprehensive information about the prisoner, such as their ID, name, age, gender, crime committed, conviction details, sentence duration, and cell number.

The code is structured around a `prisoner` structure, which defines the data fields for each prisoner. The program supports basic CRUD (Create, Read, Update, Delete) operations, which are essential for any data management system. These operations are facilitated through a series of functions that interact with the text file (jail_details.txt) and binary file (jail_details.dat) to store and retrieve the prisoner records. The use of both text and binary files ensures flexibility in data storage and retrieval, catering to different needs such as human readability and efficient data processing.

A notable feature of the program is the login system, which restricts access to the prisoner records. Users must enter valid credentials stored in a separate file (`login.txt`) before they can proceed to use the application. This login mechanism enhances the security of the system by preventing unauthorized access. Additionally, the application provides multiple attempts for logging in, after which the program exits to ensure the integrity of the data.

The program's interface is designed to be intuitive, guiding users through various options via a simple menu. Functions like `addRecord`, `searchRecord`, `editRecord`, `deleteRecord`, and `viewRecords` are implemented to provide a complete and user-friendly experience. Each function performs specific tasks, such as capturing user input, searching for records by ID, or updating and deleting records from the files. Overall, the code serves as a basic yet effective demonstration of file handling and data management techniques in C.

The retrieval and management of records are efficiently handled through both text and binary file operations. The program includes functions to find records by prisoner ID, making it easy to search for specific individuals. Additionally, the editing and deletion of records are done in a secure manner, ensuring that only the correct records are modified or removed. This system offers a robust solution for managing prisoner records, combining secure access control with comprehensive data management functionalities.

OBJECTIVE

This project focuses on designing and developing a secure, user-friendly platform for managing prisoner records. The platform aims to streamline the monitoring process, ensuring efficient data access and reporting capabilities. By incorporating robust security measures, the system safeguards the integrity and confidentiality of sensitive information, making it an effective tool for prison administration. The solution is built to be intuitive, allowing authorized users to easily interact with the system while maintaining strict control over data access and modifications.

FLOWCHART

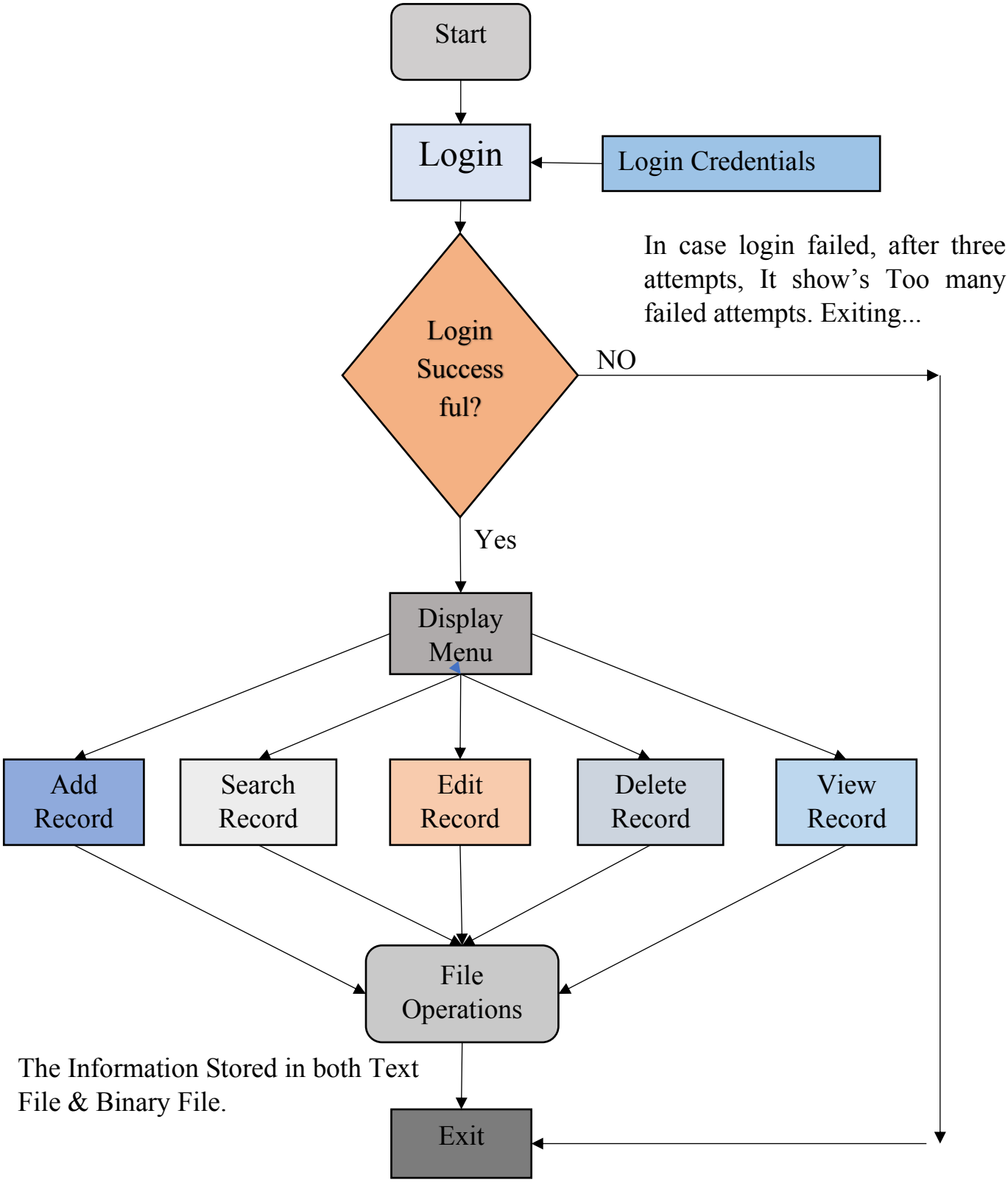

Fig1: Flow Chart of Prisoner Information Portal

USE CASE DIAGRAM

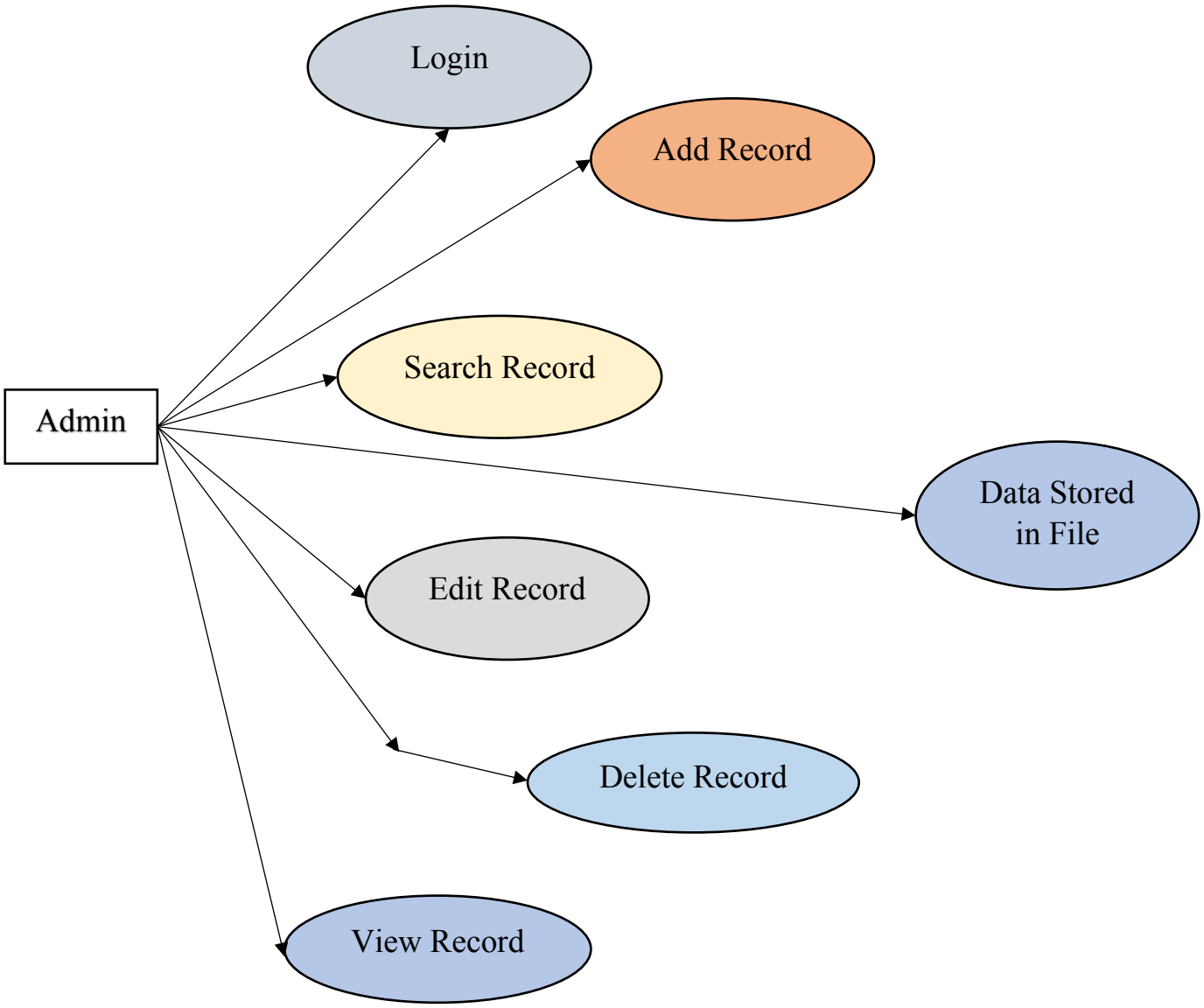

Fig2: Use Case Diagram of Prisoner Information Portal

IMPLEMENTATION & FUNCTIONALITIES

Implementation:

I implement this project using file handling functions like fopen (), fclose (), fgets (), fputs () etc...

Functionalities:

 Login **Function** (login ()):

- Prompts the user for a username and password.
- Compares the entered credentials with the stored ones in the login.txt file.
- If the credentials match, the user is granted access. Otherwise, after three failed attempts, the program exits.

 Display **Menu Function** (displayMenu ()):

- Displays a menu with six options:
    1. Add a record
    2. Search for a record
    3. Edit a record
    4. Delete a record
    5. View all records
    6. Exit
- Based on the user's selection, it calls the appropriate function.

 Add **Record Function** (addRecord ()):

- Prompts the user to enter detailed information about a prisoner.

- After gathering the data, it calls writeRecordToFile () to save the record in a text file and writeRecordToBinaryFile () to save the record in a binary file.

 Write **Record to File** (writeRecordToFile ()):

- Writes the prisoner's data in a CSV format to the jail_details.txt file.

 Write **Record to Binary File** (writeRecordToBinaryFile ()):

- Writes the prisoner's data in binary format to the jail_details.dat file.

 Search **Record Function** (searchRecord ()):

- Prompts the user to enter the ID of the prisoner they want to search for.
- It then calls findRecordById () to search the text file for the record.
- If found, the prisoner's details are displayed.

 Find **Record by ID** (findRecordById ()):

- Searches for a prisoner's record by ID in the text file (jail_details.txt).
- Returns the prisoner's data if found, otherwise returns NULL.

 Find **Record by ID in Binary** (findRecordByIdBinary ()):

- Similar to findRecordById (), but searches the binary file (jail_details.dat).

 Edit **Record Function** (editRecord ()):

- Prompts the user for the ID of the prisoner they wish to edit.
- Searches for the record using findRecordByIdBinary ().
- If the record is found, the user can update the prisoner's details.
- The program then updates the binary file with the new data, ensuring the old record is replaced.

 Delete **Record Function** (deleteRecord ()):

- Prompts the user for the ID of the prisoner they wish to delete.
- Searches the binary file for the record.
- If found, it skips writing that record to a temporary file, effectively removing it from the database when the temporary file replaces the original binary file.

 View **All Records Function** (viewRecords ()):

- Reads and displays all prisoner records stored in the binary file.

RESULTS

```
revanth@revanth-inspiron:~/Desktop/c/prison$ gcc jail.c -o jail
revanth@revanth-inspiron:~/Desktop/c/prison$ ./jail

       No login file found. Please contact the administrator.

revanth@revanth-inspiron:~/Desktop/c/prison$
```

Fig3: Login File Not Found

```
jail.c ×        login.txt ×
1 cranes@4922 my_jail
```

Fig4: Login Details in Text File

```
                ENTER YOUR LOGIN CREDENTIALS

Enter username: cranes@4922
Enter password: my_jail
```

Fig5: Enter Login Details

```
                WELCOME TO PRISONER RECORD ADMINISTRATION
*****************

1. Add Record
2. Search Record
3. Edit Record
4. Delete Record
5. View All Records
6. Exit

Enter your selection here:
```

Fig6: Main Menu

```
                FEEL FREE TO ADD THE PRISONER INFORMATION

Enter prisoner ID: 420
Enter prisoner name: Mohan
Enter prisoner age: 45
Enter gender: male
Enter crime: murder
Enter conviction: hanging till death
Enter sentence duration (in months): 1
Enter cell number: 10
Enter weight (in kg): 56
Enter height (in cm): 5.46
Enter hair color: black
Enter eye color: blue
Enter court: supreme court
Enter emergency contact name: Revanth Kumar
Enter emergency contact relation: Brother
Enter emergency contact phone number: 961286****

              "Record added successfully."
```

Fig7: Add Prisoner Details

```
                VIEW ALL PRISONER RECORDS


ID: 420
Name: Mohan
Age: 45
Gender: male
Crime: murder
Conviction: hanging till death
Sentence Duration: 1 months
Cell Number: 10
Weight: 56.00 kg
Height: 5.46 cm
Hair Color: black
Eye Color: blue
Court: supreme court
Emergency Contact Name: Revanth Kumar
Emergency Contact Relation: Brother
Emergency Contact Phone: 961286****

ID: 150
Name: ram reddy
Age: 56
Gender: male
Crime: cybercrime
Conviction: fine
Sentence Duration: 84 months
Cell Number: 96
Weight: 75.00 kg
Height: 5.12 cm
Hair Color: black
Eye Color: black
Court: high court
Emergency Contact Name: tarun
Emergency Contact Relation: father
Emergency Contact Phone: 8332649**
******************
```

Fig8: View All Prisoner Details

Fig9: Search Prisoner Details

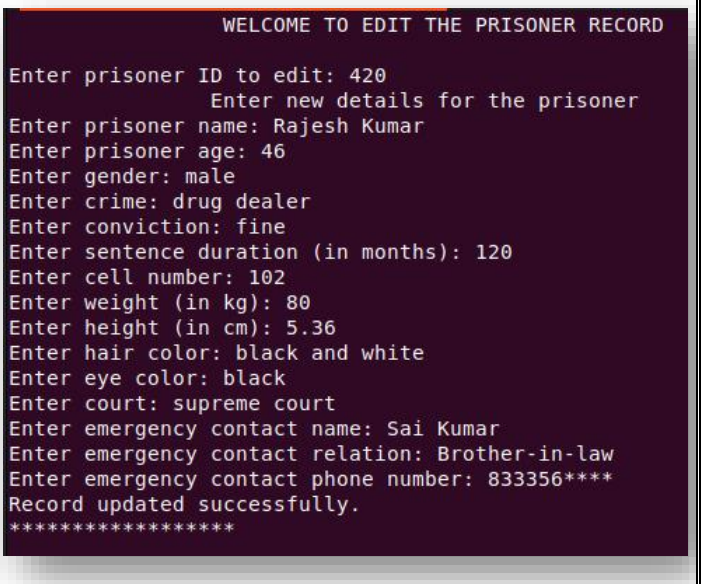

Fig10: Edit Prisoner Details

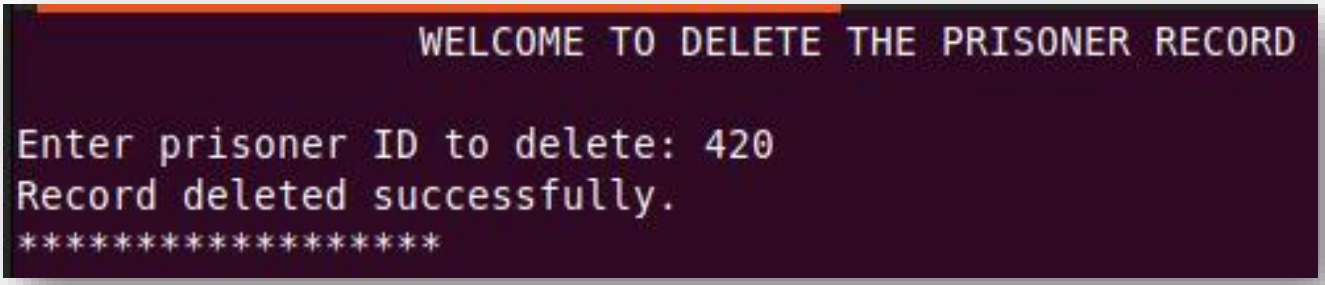

Fig11: Delete Prisoner Details

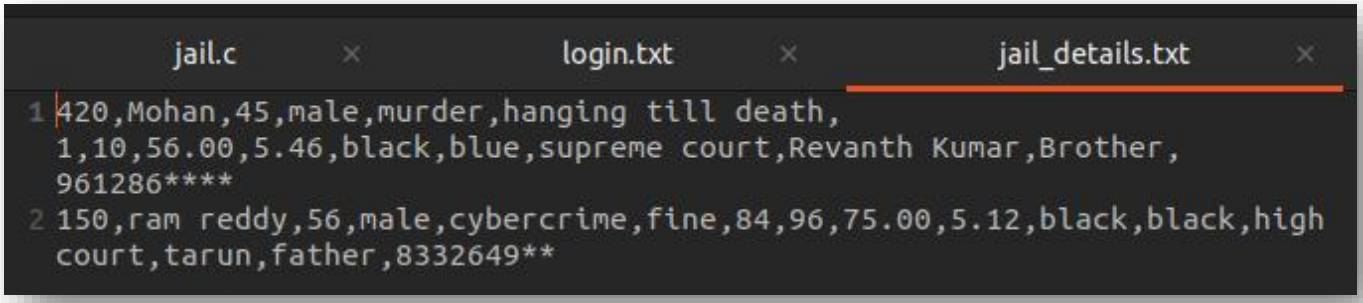

Fig12: Prisoner Details Stored in Text File

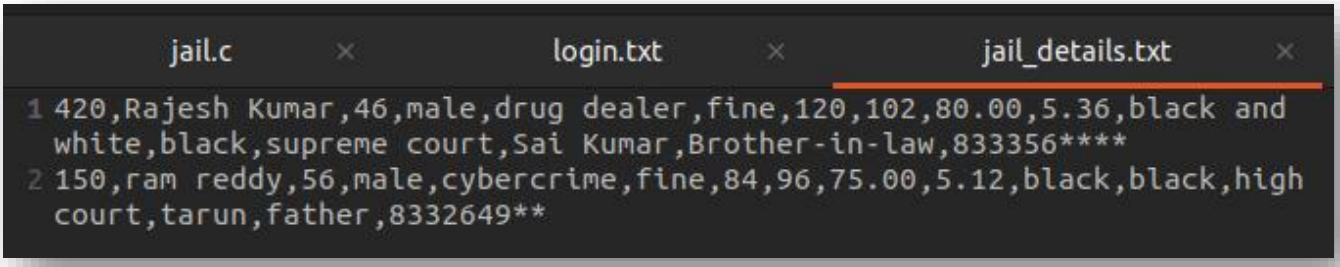

Fig13: After Edited Prisoner Details Stored in Text File

Fig14: Prisoner Details Stored in Binary File

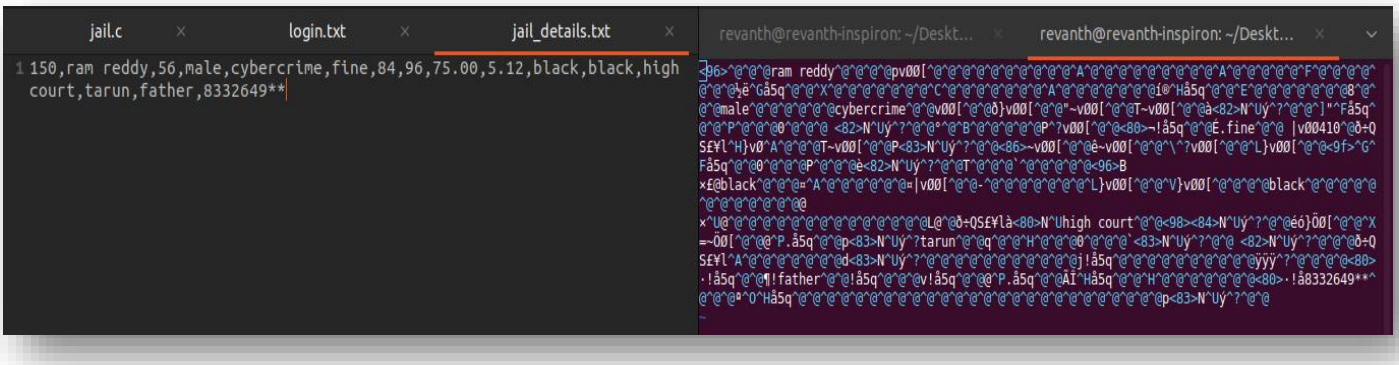

Fig15: After Deleted Prisoner Details Stored in both Text File & Binary File

ADVANTAGES

- Secure Login System
- Efficient Record Management
- Data Storage
- User-Friendly Interface

APPLICATIONS

- Inmate Management
- Visitor Management
- Security and Surveillance
- Healthcare Management
- Legal and Case Management

CONCLUSION

The Prisoner Information Portal is a streamlined tool designed to manage prisoner records using a file-based system. It operates through a command-line interface, offering essential functionalities such as adding, searching, editing, deleting, and viewing records. This simplicity makes it easy to implement and use, catering to basic record-keeping needs without the complexity of more advanced database systems. The file-based approach ensures that data is stored in a straightforward manner, making it accessible and modifiable with minimal technical overhead.

However, while the system is effective for small-scale or individual use, it has limitations when it comes to handling larger datasets or ensuring data security. As the number of records grows, the system may become slower and more prone to errors, and the lack of built-in security measures could leave sensitive information vulnerable. Enhancing the portal with advanced data management techniques, such as migrating to a database system, and incorporating robust security features, would make it more suitable for larger, more complex environments where data integrity and protection are critical.

So, the outcome of all the-the hard work did for the Prisoner Information Portal is here. It is software that helps the user to work with the prisons easily. This software reduces the amount of manual data entry and gives greater efficiency. The User Interface of it is very friendly and can be easily used by anyone.

It also decreases the amount of time taken to write FIR details and other modules. In the end, we can say that this software is performing all the tasks accurately and is doing the work for which it is made.

REFERENCES

[1] https://github.com/ss8987/Prison-Management-System

[2] https://luminousinfoways.com/products/prison-management-information-system-pmis/

[3] https://www.lovelycoding.org/priso-management system/?srsltid=AfmBOoqDvLAGQp8Wk2brUOiJPTJRFo3v5Mfg4lO0lSLDn-wFphqnSl0D

[4] https://github.com/anuragprakash009/Prison-Management-System

APPENDIX

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define MAX_NAME_LENGTH 100

#define MAX_DETAIL_LENGTH 50

#define FILENAME "jail_details.txt"

#define BINARY_FILENAME "jail_details.dat"

#define LOGIN_FILE "login.txt"


struct prisoner

{

    int id;

    char name[MAX_NAME_LENGTH];

    int age;

    char gender[10];

    char crime[MAX_NAME_LENGTH];

    char conviction[MAX_NAME_LENGTH];

    int sentenceDuration;

    int cellNumber;

    float weight;

    float height;

    char hairColor[MAX_DETAIL_LENGTH];
```

```
    char eyeColor[MAX_DETAIL_LENGTH];
    char court[MAX_DETAIL_LENGTH];
    char emergencyContactName[MAX_NAME_LENGTH];
    char emergencyContactRelation[MAX_DETAIL_LENGTH];
    char emergencyContactPhone[MAX_DETAIL_LENGTH];
};
typedef struct prisoner p;


void login();
void displayMenu();
void addRecord();
void searchRecord();
void editRecord();
void deleteRecord();
void viewRecords();
void writeRecordToFile(p *prisoner);
void writeRecordToBinaryFile(p *prisoner);
p *findRecordById(int id);
p *findRecordByIdBinary(int id);
```

```
int main()
{
    login();
    system("clear");
    displayMenu();
    return 0;
}

void login()
{
    char username[50], password[50];
    char storedUsername[50], storedPassword[50];
    int attempts = 3;

    FILE *file = fopen(LOGIN_FILE, "r");
    if (file == NULL)
    {
        printf("No login file found. Please contact the administrator.\n");
        exit(0);
    }

    fscanf(file, "%s %s", storedUsername, storedPassword);
    fclose(file);
```

```
    while (attempts > 0)
    {
        system("clear");
        printf("\n\t\tENTER YOUR LOGIN CREDENTIALS\n\n");

        printf("Enter username: ");
        scanf("%s", username);
        printf("Enter password: ");
        scanf("%s", password);

        if (strcmp(username, storedUsername) == 0 && strcmp(password, storedPassword) ==
0)
        {
            printf("Login successful.\n");
            return;
        }
        else
        {
            attempts--;
            if (attempts > 0)
            {
                printf("Invalid credentials. You have %d attempt(s) remaining.\n\n", attempts);
            }
        }
    }
```

```
    printf("Too many failed attempts. Exiting...\n");
    exit(0);
}

void displayMenu()
{
    int choice;
    printf("\n\t\t WELCOME TO PRISONER RECORD ADMINISTRATION\n\n");
    while (1)
    {
        printf("**************************************************\n");
        printf("\n1. Add Record\n");
        printf("2. Search Record\n");
        printf("3. Edit Record\n");
        printf("4. Delete Record\n");
        printf("5. View All Records\n");
        printf("6. Exit\n\n");

        printf("Enter your selection here: ");
        scanf("%d", &choice);
```

```
switch (choice)
{
    case 1:
        addRecord();
        break;
    case 2:
        searchRecord();
        break;
    case 3:
        editRecord();
        break;
    case 4:
        deleteRecord();
        break;
    case 5:
        viewRecords();
        break;
    case 6:
        system("clear");
        printf("\n\n\n\t\t Thank you for coming, and do visit again.\n\n");
        exit(0);
    default:
        printf("\nNot a valid option. Kindly give it another attempt.\n\n");
```

```
        }
    }
}

// Function to add a record
void addRecord()
{
    p prisoner;
    system("clear");
    printf("\t\t FEEL FREE TO ADD THE PRISONER INFORMATION\n\n");

    printf("Enter prisoner ID: ");
    scanf("%d", &prisoner.id);
    getchar();

    printf("Enter prisoner name: ");
    fgets(prisoner.name, MAX_NAME_LENGTH, stdin);
    prisoner.name[strcspn(prisoner.name, "\n")] = '\0';

    printf("Enter prisoner age: ");
    scanf("%d", &prisoner.age);
    getchar();

    printf("Enter gender: ");
```

```
fgets(prisoner.gender, sizeof(prisoner.gender), stdin);
prisoner.gender[strcspn(prisoner.gender, "\n")] = '\0';

printf("Enter crime: ");
fgets(prisoner.crime, MAX_NAME_LENGTH, stdin);
prisoner.crime[strcspn(prisoner.crime, "\n")] = '\0';

printf("Enter conviction: ");
fgets(prisoner.conviction, MAX_NAME_LENGTH, stdin);
prisoner.conviction[strcspn(prisoner.conviction, "\n")] = '\0';

printf("Enter sentence duration (in months): ");
scanf("%d", &prisoner.sentenceDuration);
getchar();

printf("Enter cell number: ");
scanf("%d", &prisoner.cellNumber);
getchar();

printf("Enter weight (in kg): ");
scanf("%f", &prisoner.weight);
getchar();

printf("Enter height (in cm): ");
```

```
    scanf("%f", &prisoner.height);
    getchar();

    printf("Enter hair color: ");
    fgets(prisoner.hairColor, MAX_DETAIL_LENGTH, stdin);
    prisoner.hairColor[strcspn(prisoner.hairColor, "\n")] = '\0';

    printf("Enter eye color: ");
    fgets(prisoner.eyeColor, MAX_DETAIL_LENGTH, stdin);
    prisoner.eyeColor[strcspn(prisoner.eyeColor, "\n")] = '\0';

    printf("Enter court: ");
    fgets(prisoner.court, MAX_DETAIL_LENGTH, stdin);
    prisoner.court[strcspn(prisoner.court, "\n")] = '\0';

    printf("Enter emergency contact name: ");
    fgets(prisoner.emergencyContactName, MAX_NAME_LENGTH, stdin);
    prisoner.emergencyContactName[strcspn(prisoner.emergencyContactName, "\n")] = '\0';

    printf("Enter emergency contact relation: ");
    fgets(prisoner.emergencyContactRelation, MAX_DETAIL_LENGTH, stdin);
    prisoner.emergencyContactRelation[strcspn(prisoner.emergencyContactRelation, "\n")] =
'\0';

    printf("Enter emergency contact phone number: ");
```

```
    fgets(prisoner.emergencyContactPhone, MAX_DETAIL_LENGTH, stdin);
    prisoner.emergencyContactPhone[strcspn(prisoner.emergencyContactPhone, "\n")] = '\0';

    writeRecordToFile(&prisoner);
    writeRecordToBinaryFile(&prisoner);

    printf("\n\t\t\"Record added successfully.\"\n\n");
}

// Function to write a record to the text file
void writeRecordToFile(p *prisoner)
{
    FILE *file = fopen(FILENAME, "a");

    if (file == NULL)
    {
        printf("Unable to open file for writing.\n");
        return;
    }

    fprintf(file, "%d,%s,%d,%s,%s,%s,%d,%d,%.2f,%.2f,%s,%s,%s,%s,%s,%s\n",
            prisoner->id,
            prisoner->name,
            prisoner->age,
```

```
            prisoner->gender,
            prisoner->crime,
            prisoner->conviction,
            prisoner->sentenceDuration,
            prisoner->cellNumber,
            prisoner->weight,
            prisoner->height,
            prisoner->hairColor,
            prisoner->eyeColor,
            prisoner->court,
            prisoner->emergencyContactName,
            prisoner->emergencyContactRelation,
            prisoner->emergencyContactPhone);
    fclose(file);
}

// Function to write a record to the binary file
void writeRecordToBinaryFile(p *prisoner)
{
    FILE *file = fopen(BINARY_FILENAME, "ab");

    if (file == NULL)
    {
        printf("Unable to open binary file for writing.\n");
```

```
        return;
    }

    fwrite(prisoner, sizeof(p), 1, file);
    fclose(file);
}

// Function to search for a record in the text file
void searchRecord()
{
    int id;
    p *prisoner;

    system("clear");
    printf("\t\t WELCOME TO SEARCH THE PRISONER RECORD \n\n");

    printf("Enter prisoner ID to search: ");
    scanf("%d", &id);

    prisoner = findRecordById(id);
    if (prisoner != NULL)
    {
        printf("\nID: %d\nName: %s\nAge: %d\nGender: %s\nCrime: %s\nConviction:
%s\nSentence Duration: %d months\nCell Number: %d\nWeight: %.2f kg\nHeight: %.2f
```

```
cm\nHair Color: %s\nEye Color: %s\nCourt: %s\nEmergency Contact Name:
%s\nEmergency Contact Relation: %s\nEmergency Contact Phone: %s\n",
                prisoner->id,
                prisoner->name,
                prisoner->age,
                prisoner->gender,
                prisoner->crime,
                prisoner->conviction,
                prisoner->sentenceDuration,
                prisoner->cellNumber,
                prisoner->weight,
                prisoner->height,
                prisoner->hairColor,
                prisoner->eyeColor,
                prisoner->court,
                prisoner->emergencyContactName,
                prisoner->emergencyContactRelation,
                prisoner->emergencyContactPhone);
        free(prisoner);
    }
    else
    {
        printf("Record not found.\n");
    }
}
```

```
// Function to find a record by ID in the text file
p *findRecordById(int id)
{
    FILE *file = fopen(FILENAME, "r");
    if (file == NULL)
    {
        printf("Unable to open file for reading.\n");
        return NULL;
    }

    p *prisoner = malloc(sizeof(p));

    while (fscanf(file,
"%d,%[^,],%d,%[^,],%[^,],%[^,],%d,%d,%f,%f,%[^,],%[^,],%[^,],%[^,],%[^,],%[^\n]\n",
                &prisoner->id,
                prisoner->name,
                &prisoner->age,
                prisoner->gender,
                prisoner->crime,
                prisoner->conviction,
                &prisoner->sentenceDuration,
                &prisoner->cellNumber,
                &prisoner->weight,
                &prisoner->height,
```

```
            prisoner->hairColor,
            prisoner->eyeColor,
            prisoner->court,
            prisoner->emergencyContactName,
            prisoner->emergencyContactRelation,
            prisoner->emergencyContactPhone) != EOF)
    {
        if (prisoner->id == id)
        {
            fclose(file);
            return prisoner;
        }
    }

    fclose(file);
    free(prisoner);
    return NULL;
}

// Function to search for a record in the binary file
p *findRecordByIdBinary(int id)
{
    FILE *file = fopen(BINARY_FILENAME, "rb");
    if (file == NULL)
```

```
    {
        printf("Unable to open binary file for reading.\n");
        return NULL;
    }

    p *prisoner = malloc(sizeof(p));

    while (fread(prisoner, sizeof(p), 1, file))
    {
        if (prisoner->id == id)
        {
            fclose(file);
            return prisoner;
        }
    }

    fclose(file);
    free(prisoner);
    return NULL;
}

// Function to edit a record in the binary file
void editRecord()
{
```

```
int id;
p *prisoner;
FILE *file, *tempFile, *textFile, *tempTextFile;

system("clear");
printf("\t\t WELCOME TO EDIT THE PRISONER RECORD \n\n");

printf("Enter prisoner ID to edit: ");
scanf("%d", &id);

prisoner = findRecordByIdBinary(id);
if (prisoner == NULL)
{
   printf("Record not found.\n");
   return;
}

// Update the binary file
file = fopen(BINARY_FILENAME, "rb");
tempFile = fopen("temp.dat", "wb");

if (file == NULL || tempFile == NULL)
{
   printf("Unable to open binary file.\n");
```

```
    return;
}

p temp;
while (fread(&temp, sizeof(p), 1, file))
{
    if (temp.id == id)
    {
        printf("Enter new details for the prisoner:\n");

        printf("Enter prisoner name: ");
        getchar();
        fgets(prisoner->name, MAX_NAME_LENGTH, stdin);
        prisoner->name[strcspn(prisoner->name, "\n")] = '\0';

        printf("Enter prisoner age: ");
        scanf("%d", &prisoner->age);
        getchar();

        printf("Enter gender: ");
        fgets(prisoner->gender, sizeof(prisoner->gender), stdin);
        prisoner->gender[strcspn(prisoner->gender, "\n")] = '\0';

        printf("Enter crime: ");
```

```
fgets(prisoner->crime, MAX_NAME_LENGTH, stdin);
prisoner->crime[strcspn(prisoner->crime, "\n")] = '\0';

printf("Enter conviction: ");
fgets(prisoner->conviction, MAX_NAME_LENGTH, stdin);
prisoner->conviction[strcspn(prisoner->conviction, "\n")] = '\0';

printf("Enter sentence duration (in months): ");
scanf("%d", &prisoner->sentenceDuration);
getchar();

printf("Enter cell number: ");
scanf("%d", &prisoner->cellNumber);
getchar();

printf("Enter weight (in kg): ");
scanf("%f", &prisoner->weight);
getchar();

printf("Enter height (in cm): ");
scanf("%f", &prisoner->height);
getchar();

printf("Enter hair color: ");
```

```
        fgets(prisoner->hairColor, MAX_DETAIL_LENGTH, stdin);
        prisoner->hairColor[strcspn(prisoner->hairColor, "\n")] = '\0';

        printf("Enter eye color: ");
        fgets(prisoner->eyeColor, MAX_DETAIL_LENGTH, stdin);
        prisoner->eyeColor[strcspn(prisoner->eyeColor, "\n")] = '\0';

        printf("Enter court: ");
        fgets(prisoner->court, MAX_DETAIL_LENGTH, stdin);
        prisoner->court[strcspn(prisoner->court, "\n")] = '\0';

        printf("Enter emergency contact name: ");
        fgets(prisoner->emergencyContactName, MAX_NAME_LENGTH, stdin);
        prisoner->emergencyContactName[strcspn(prisoner->emergencyContactName, "\n")]
= '\0';

        printf("Enter emergency contact relation: ");
        fgets(prisoner->emergencyContactRelation, MAX_DETAIL_LENGTH, stdin);
        prisoner->emergencyContactRelation[strcspn(prisoner->emergencyContactRelation,
"\n")] = '\0';

        printf("Enter emergency contact phone number: ");
        fgets(prisoner->emergencyContactPhone, MAX_DETAIL_LENGTH, stdin);
        prisoner->emergencyContactPhone[strcspn(prisoner->emergencyContactPhone, "\n")]
= '\0';
```

```
            fwrite(prisoner, sizeof(p), 1, tempFile);
        }
        else
        {
            fwrite(&temp, sizeof(p), 1, tempFile);
        }
    }

    fclose(file);
    fclose(tempFile);

    remove(BINARY_FILENAME);
    rename("temp.dat", BINARY_FILENAME);

    // Update the text file
    textFile = fopen(FILENAME, "r");
    tempTextFile = fopen("temp.txt", "w");

    if (textFile == NULL || tempTextFile == NULL)
    {
        printf("Unable to open text file.\n");
        return;
    }
```

```
    char buffer[MAX_NAME_LENGTH * 16];
    while (fgets(buffer, sizeof(buffer), textFile))
    {
        int tempId;
        sscanf(buffer, "%d,", &tempId);
        if (tempId == id)
        {
            fprintf(tempTextFile,
"%d,%s,%d,%s,%s,%s,%d,%d,%.2f,%.2f,%s,%s,%s,%s,%s,%s\n",
                    prisoner->id,
                    prisoner->name,
                    prisoner->age,
                    prisoner->gender,
                    prisoner->crime,
                    prisoner->conviction,
                    prisoner->sentenceDuration,
                    prisoner->cellNumber,
                    prisoner->weight,
                    prisoner->height,
                    prisoner->hairColor,
                    prisoner->eyeColor,
                    prisoner->court,
                    prisoner->emergencyContactName,
                    prisoner->emergencyContactRelation,
```

```
                prisoner->emergencyContactPhone);
        }
        else
        {
            fputs(buffer, tempTextFile);
        }
    }

    fclose(textFile);
    fclose(tempTextFile);

    remove(FILENAME);
    rename("temp.txt", FILENAME);

    free(prisoner);

    printf("Record updated successfully.\n");
}

// Function to delete a record in the binary file
void deleteRecord()
{
    int id;
    FILE *file, *tempFile;
```

```
system("clear");
printf("\t\t WELCOME TO DELETE THE PRISONER RECORD \n\n");

printf("Enter prisoner ID to delete: ");
scanf("%d", &id);

file = fopen(BINARY_FILENAME, "rb");
tempFile = fopen("temp.dat", "wb");

if (file == NULL || tempFile == NULL)
{
   printf("Unable to open file.\n");
   return;
}

p temp;
int found = 0;
while (fread(&temp, sizeof(p), 1, file))
{
   if (temp.id == id)
   {
      found = 1;
      continue;
```

```
        }
        fwrite(&temp, sizeof(p), 1, tempFile);
    }

    fclose(file);
    fclose(tempFile);

    remove(BINARY_FILENAME);
    rename("temp.dat", BINARY_FILENAME);

    if (found)
    {
        printf("Record deleted successfully.\n");
    }
    else
    {
        printf("Record not found.\n");
    }
}
```

```
// Function to view all records in the binary file
void viewRecords()
{
    FILE *file = fopen(BINARY_FILENAME, "rb");

    if (file == NULL)
    {
        printf("Unable to open file.\n");
        return;
    }

    p prisoner;
    system("clear");
    printf("\t\t VIEW ALL PRISONER RECORDS \n\n");

    while (fread(&prisoner, sizeof(p), 1, file))
    {
        printf("\nID: %d\nName: %s\nAge: %d\nGender: %s\nCrime: %s\nConviction: %s\nSentence Duration: %d months\nCell Number: %d\nWeight: %.2f kg\nHeight: %.2f cm\nHair Color: %s\nEye Color: %s\nCourt: %s\nEmergency Contact Name: %s\nEmergency Contact Relation: %s\nEmergency Contact Phone: %s\n",
                prisoner.id,
                prisoner.name,
                prisoner.age,
                prisoner.gender,
```

```
            prisoner.crime,
            prisoner.conviction,
            prisoner.sentenceDuration,
            prisoner.cellNumber,
            prisoner.weight,
            prisoner.height,
            prisoner.hairColor,
            prisoner.eyeColor,
            prisoner.court,
            prisoner.emergencyContactName,
            prisoner.emergencyContactRelation,
            prisoner.emergencyContactPhone);
    }

    fclose(file);
}
```