## 

**DSA [20ES117] COURSE PROJECT REPORT**

**On**

**“POLICE FIR RECORD MANAGEMENT SYSTEM”**

Developed By:

H.T.NO STUDENT NAME

2103A51109 SUMMYA MEHVEEN

Under the Guidance of

Dr. A. SIVA KRISHNA REDDY

Associate Professor

Submitted to

Department Computer Science and Artificial Intelligence

SR University

Ananthasagar(V), Hasanparthy(M), Hanamkonda(Dist.) – 506371

[www.sru.edu.in](http://www.sru.edu.in)

**December 2022**

**Department of Computer Science and Artificial Intelligence**

**CERTIFICATE**

This is to certify that the DSA course project report entitled **“POLICE FIR RECORD MANAGEMENT SYSTEM”** is a record of Bonafede work carried out by the student **“Summya Mehveen”** bearing roll number **“2103A51109”** of Computer Science and Artificial Intelligence department during the academic year 2022-23.

**Supervisor**

(Dr. A. Siva Krishna Reddy)

**PROBLEM STATEMENT:**

Traditionally, in all police stations the FIR records [the details of the convicts in an organized way] are written and store in a FIR record notebook. And a police officer is specially appointed for the maintenance of these records. But the problem arises as it requires a lot of place and paper to store the data, a lot of manual work is required and the security cannot be assured as a trespasser can access the data or they may be lost along with the time.

**PROPOSED SOLUTION:**

To overcome the mentioned problem, I have come up with an idea of developing a “Police FIR record management system”.

**OBJECTIVE:**

This project is based on a concept to maintain all the Fir records of a police station. From this system, the user can easily add each and every convict’s detail such as convict’s id, name, age, gender, conviction etc. Apart from this, the user can view/check a list of the records, search all the records and also this application is password protected. Only authenticated users can use the application.

**FEATURES:**

* To provide a rich feature for digital service.
* To build a feature easy for managing records
* Adding records
* Easy maintenance
* Password Protected
* Can access data even after long time
* User-friendly system

**MODULES:**

1. Login:

The user who wants to enter a record can access the modules of application. It contains user name and passwords as the attributes.

1. Main Menu:

This module will provide the brain Strom of the complete application. The user can add a record, search a record, view all the records or can logout of the modules.

1. Add record:

The user can add the records into the system using this module. One can add the following details:

* Name
* ID
* Age
* Crime
* Gender
* Court name etc.

1. Display:

This module displays the all the details of the record inserted into the application.

1. Search:

By this module, the user can search the details of the required convict. It can be done through the registered FIR numbers.

1. Generate FIR:

By this module, the application generates the FIR number after every FIR insertion.

1. Check user:

By using this module, the application verifies whether the record was accessed by an authorised person or not.

**KNOWLEDGE REQUIRED TO DEVELOP THIS APPLICATION:**

* Linked lists (single linked list)
* Structures (structures and nested structures)
* Control statements (if, if else, else)
* Switch-case
* Looping statements (while, for)
* Functions (any type of user defined functions)
* Pointers (pointers to structures)

**Introduction to linked lists:**

**Linked lists:**

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers. They include a series of connected nodes. Here, each node stores the data and the address of the next node.



**Why linked lists?**

Arrays can be used to store linear data of similar types, but arrays have the following limitations:

**The size of the arrays is fixed**: So, we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.

**Insertion of a new element / Deletion of an existing element in an array of elements is expensive**: The room has to be created for the new elements and to create room existing elements have to be shifted but in Linked list if we have the head node then we can traverse to any node through it and insert new node at the required position.

**Example:**   
In a system, if we maintain a sorted list of IDs in an array id [] = [1000, 1010, 1050, 2000,2040].   
If we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id [], everything after 1010 has to be moved due to this so much work is being done which affects the efficiency of the code.

**Advantages of Linked Lists over arrays:**

* Dynamic Array.
* Ease of Insertion/Deletion.

**Drawbacks of Linked Lists:**

* Random access is not allowed. We have to access elements sequentially starting from the first node (head node). So, we cannot do a [binary search with linked lists](https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/) efficiently with its default implementation.
* Extra memory space for a pointer is required with each element of the list.
* Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

**Types of Linked Lists:**

* **Simple Linked List** – In this type of linked list, one can move or traverse the linked list in only one direction
* **Doubly Linked List** – In this type of linked list, one can move or traverse the linked list in both directions (Forward and Backward)
* **Circular Linked List** – In this type of linked list, the last node of the linked list contains the link of the first/head node of the linked list in its next pointer and the first/head node contains the link of the last node of the linked list in its prev pointer

**Basic operations on Linked Lists:**

* [Deletion](https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/)
* [Insertion](https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/)
* [Search](https://www.geeksforgeeks.org/search-an-element-in-a-linked-list-iterative-and-recursive/)
* Display

**Representation of Linked Lists:**

* A linked list is represented by a pointer to the first node of the linked list. The first node is called the head of the linked list. If the linked list is empty, then the value of the head points to NULL.
* Each node in a list consists of at least two parts:
* A Data Item (we can store integer, strings, or any type of data).
* Pointer (Or Reference) to the next node (connects one node to another) or An address of another node

**SOURCE CODE:**

// DSA Project

//Police FIR record management System

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

struct record

{

char name [20], gender [7], complaint [50], court [50], details [50], crime 11], date [40];

int age, appointment, id;

struct record \*next;

};

int count=1, d=1;

struct record \*start=NULL;

struct record \*addrecord (struct record \*start);

struct record \*display (struct record \*start);

struct record search (struct record \*start);

int genereatefir();

void mainmenu();

void check\_user (char username [10], char password [10]);

int main () {

char password [10], username[10], ch;

int i, option;

system("COLOR 8F");

system("cls");

printf("\t\t========================================================\n");

printf("\t\t==========POLICEFIRRECORDMANAGEMENT=================\n");

printf("\t\t==================TOWNPOLICESTATION===================\n");

printf("\t\t========================================================\n");

printf("\t\t========================================================\n");

printf("\t\t===================LOGINCREDENTIALS===================\n");

printf("\t\t========================================================\n");

printf ("\t\t\t->Enter User name: ");

scanf(“% s", username);

printf ("\t\t\t->Enter the password : ");

for(i=0;i<7;i++)

{

ch = getch ();

password[i] = ch;

ch = '\*' ;

printf ("%c", ch );

}

password[i] = '\0';

check\_user(username, password);

getch();

}

void check\_user(char username[10], char password[10]){

int n;

if(strcmp(username, "police") == 0 )

{

if(strcmp(password, "default") == 0)

{

printf("\n\t\t\t----------WELCOME OFFICER-----------\n");

mainmenu();

}

else

{

printf("\n\t\t\*\*\*\*\*Password or User ID entered is invalid\*\*\*\*\*\n");

printf("\n->Press 1 to Try AGAIN or press 2 to exit: ");

scanf("%d",&n);

if(n==1)

{

main();

}}

}}

int genereatefir(){

return count++;

}

struct record \*addrecord(struct record \*start){

int age, n ,val, y, id;

char name[20], gender[7], complaint[50], court[50], details[50], crime[11];

struct record \*new\_node, \*ptr;

new\_node=(struct record\*)malloc(sizeof(struct record));

printf("\n --------------ENTER CONVICT DETAILS-----------------\n");

printf("->Enter Convict ID");

scanf("%d",&id);

new\_node->id=id;

printf("\n->Enter Name: ");

scanf("%s",name);

strcpy(new\_node->name,name);

printf("->Enter Age: ");

scanf("%d",&age);

new\_node->age=age;

printf("\n ->Enter Gender: ");

scanf("%s",gender);

strcpy(new\_node->gender,gender);

printf("->Enter Crime:");

scanf("%s",crime);

strcpy(new\_node->crime,crime);

printf("->Enter Complaint of act: ");

scanf("%s",complaint);

strcpy(new\_node->complaint,complaint);

printf("->Enter Court:");

scanf("%s",court);

strcpy(new\_node->court,court);

printf("\n->Press 1. To insert the person in the appointment list else press 2: ");

scanf("%d",&y);

if(y==1)

{

val=genereatefir();

new\_node->appointment=val;

}

else

{

new\_node->appointment=0;

}

printf("\n \t\t\t =========== Convict Record added successfully ========\n");

if(start==NULL)

{

new\_node->next=NULL;

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=new\_node;

new\_node->next=NULL;

}

return start;

}

struct record \*display(struct record \*start)

{

if(start==NULL)

{

printf("\n \t-------The list of Convicts is empty...Do insert it first------------\n\n");

}

else

{

struct record \*ptr;

int i=1;

ptr=start;

while(ptr!=NULL)

{

printf("\n\n\*\*CONVICT %d\*\*",i);

if(ptr->appointment!=0)

{

printf("\n->FIR : %d",ptr->appointment);

}

printf("\n->ID:%d",ptr->id);

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

printf("\n->Age: %d",ptr->age);

printf("\n->Gender: %s",ptr->gender);

printf("\n->Crime: %s",ptr->crime);

printf("\n->Complaint of act: %s",ptr->complaint);

printf("\n->Court: %s",ptr->court);

ptr=ptr->next;

i++;

}

return start;

}

}

void mainmenu()

{

int choice;

do

{

printf("\n\t\t\t====================MAINMENU==================\n");

printf("\t\t\t\t ||\t1.Add new Record\t||\n");

printf("\t\t\t\t ||\t2.Display Records\t||\n");

printf("\t\t\t\t ||\t3.Search Records\t\t||\n");

printf("\t\t\t\t ||\t4.Log Out\t\t||\n");

printf("\t\t\t\t ||\t5.Exit\t\t\t||\n");

printf("\n\t\t\t==========================================\n");

printf("\n->Enter Your Option: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

start=addrecord(start);

break;

case 2:

start=display(start);

break;

case 3:

search(start);

break;

case 4:

main();

break;

case 5:

system("cls");

printf("\n\n\t\tTHANK YOU \n\n ");

exit(0);

}

}

While (choice!=6);

}

struct record search (struct record \*start)

{

int f, p;

if(start==NULL) {

printf ("\n \t--------The list of Convicts is empty...Do insert it first---------\n\n");

}

else

{

struct record \* ptr;

ptr =start;

printf ("\n ->Enter FIR no. to search:");

scan f("%d", &f);

while(ptr!=NULL)

{

if(ptr->appointment==f)

{

printf("\n->ID:%d",ptr->id);

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

printf("\n->Age: %d",ptr->age);

printf("\n->Gender: %s",ptr->gender);

printf("\n->Crime: %s",ptr->crime);

printf("\n->Complaint of act: %s",ptr->complaint);

printf("\n->Court: %s",ptr->court);

}

ptr=ptr->next;

}

}

}

**OUTPUT:**























