

CHAPTER 1

INTRODUCTION

In this chapter, Database and its types, advantages, components etc. are discussed.

1.1 DBMS

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize, create, retrieve, update and manage data in a database. Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used including business, electronic, commerce, engineering, medicine, genetics, law, education, and library science.

The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of data. Meaningful data combined to form information. Hence, information is interpreted data-data provided with semantics. MS ACCESS is one of the most common examples of database management software. Database Management Systems are now as indispensable tool for managing information, and a course on the principles and practice of database systems is now an integral part of computer science curricular. This book covers the fundamentals of modern database management systems, in particular relational database systems.

1.1.1 Causes to use DBMS

- To develop software applications in less time.
- Data independence and efficient use of data.
- For uniform data administration.
- For data integrity and security.
- For concurrent access to data, and data recovery from crashes.
- To use user-friendly declarative query language.

1.1.2 Uses of DBMS

- **Airlines:** reservations, schedules, etc.
- **Telecom:** calls made customer details, network usage, etc.
- **Universities:** registration, results, grades, etc.
- **Sales:** products, purchases, customers, etc.
- **Banking:** all transactions etc.

1.1.3 Advantage of DBMS

A DBMS manage data and has many advantages. These are:

Data independence

Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.

Efficient data access

DBMS utilizes a mixture of sophisticated concepts and techniques for storing and retrieving data competently, and this feature becomes important in cases where the data is stored on external storage devices.

Data integrity and security

If data is accessed through the DBMS, the DBMS can enforce integrity constraints on the data.

Data administration

When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.

1.1.4 Components of DBMS

Users: Users may be of any kind such as DB administrator, System developer or database users.

Database application: Database application may be Departmental, Personal, organization's and / or Internal.

DBMS: Software that allows users to create and manipulate database access.

Database: Collection of logical data as a single unit.

PHP (Hypertext Pre-processor) is an open-source HTML-embedded server-side scripting language which is used to develop dynamic and interactive web applications and also used as a general-purpose programming language.

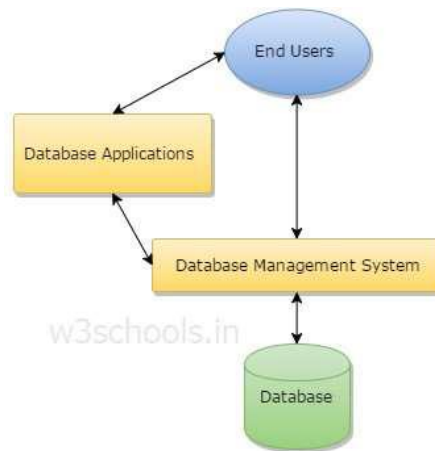


Fig 1.1: components of database management system

1.2 TYPES OF DBMS:

There are four main types of Database Management Systems (DBMS) and these are based upon their management of database structures. In other words, the types of DBMS are entirely dependent upon how the database is structured by that particular DBMS.

1.2.1 Hierarchical DBMS

A DBMS is said to be hierarchical if the relationships among data in the database are established in such a way that one data item is present as the subordinate of another one or a sub unit. The hierarchical data model was developed by IBM in 1968 and introduced in information management systems. This model is like a structure of a tree with the records forming the nodes.

1.2.2 Network DBMS

A DBMS is said to be a Network DBMS if the relationships among data in the database are of type many-to-many. The many-to-many communication paradigm is one of three major Internet computing paradigms, characterized by multiple users contributing and receiving information, with the information elements often interlinked across different websites.

1.2.3 Relational DBMS

A DBMS is said to be a Relational DBMS or RDBMS if the database relationships are treated in the form of a table. There are three keys on relational DBMS: relation, domain and attributes.

1.2.4 Object-oriented DBMS

Able to handle many new data types, including graphics, photographs, audio, and video, object oriented databases represent a significant advance over their other database cousins. Hierarchical and network databases are all designed to handle structured data; that is, data that fits nicely into fields, rows, and columns.

1.3 Existing System

The existing system for the criminal database project comprises a relational database management system (RDBMS) storing criminal records in structured tables. Users interact with the system through a command-line interface or graphical user interface to perform CRUD operations on criminal data. The system facilitates efficient management of criminal information, aiding law enforcement agencies in tracking and investigating criminal activities. It provides functionalities for adding, retrieving, updating, and deleting criminal records, enhancing the organization and accessibility of critical data.

1.4 Propose System

The proposed system for the criminal database project aims to efficiently manage and organize criminal records for law enforcement agencies. It will include features such as storing detailed information about criminals, tracking their criminal activities, and facilitating seamless search and retrieval of data.

In this chapter, hardware requirements have been discussed in section 2.1 and software requirements has been discussed in 2.2

System Specification can be divided into two-

2.1 Hardware specifications

2.2 Software specifications

CHAPTER 2

REQUIREMENTS SPECIFICATION

In this chapter, hardware requirements have been discussed in section 2.1 and software requirements has been discussed in 2.1

Table 2.1: Specifications

Hardware specifications	Software specifications
32/64 bit operating system	Operating System: windows
2 GB RAM or above	Languages: PHP,HTML,JS,CSS(Bootstrap),python
40 GB hard disk or above	Database: sqlite3
VGA COLOR Monitor	Server: XAMPP enabled with Apache and sqlite3
Keyboard	Browser: Chrome etc.
Mouse	Text editor: Sublime Text.

CHAPTER 3

MODULE DESCRIPTION

The modules needed for a Criminal Database Management System project

3.1 Criminal Management Module

This module handles the management of criminals' data, including adding new criminals, updating existing records, and deleting records if necessary.

3.1.1 Add Criminal

Allows authorized users to input data for a new criminal and store it in the database. Inputs typically include the criminal's name, gender, age, height, weight, and any other relevant information.

3.1.3 Update Criminal Record

Enables authorized users to modify the details of an existing criminal's record in the database. This function may include options to update attributes such as age, height, weight, or other identifying information.

3.1.4 Delete Criminal Record: Provides the functionality to delete a criminal's record from the database. This action may be performed in cases of data entry errors, duplicate records, or when the criminal's information is no longer relevant.

3.1.5 Search Criminals

Allows users to search for specific criminals based on various criteria such as name, gender, age range, height, weight, or any other available information. This function helps in retrieving relevant data efficiently.

3.1.6 View Criminal Details

Provides the ability to view detailed information about a specific criminal stored in the database, including all attributes associated with that individual.

3.1.1 Generate Criminal Reports

Enables users to generate reports summarizing information about criminals stored in the database. These reports may include statistics, trends, or other insights derived from the criminal data.

3.2 Security Measures

3.2.1 Access Control

Implements access control mechanisms to ensure that only authorized users can perform operations within the module, protecting sensitive criminal data from unauthorized access.

3.2.2 Data Encryption

Utilizes encryption techniques to secure the transmission and storage of sensitive criminal information, safeguarding it from unauthorized interception or disclosure.

3.3 Integration

3.3.1 Integration with Other Modules

Integrates seamlessly with other modules within the database project, such as the Crime Management Module and Case Management Module, to provide a cohesive solution for managing criminal data throughout the system.

3.4 Scalability and Performance

3.4.1 Scalable Architecture

Designed with scalability in mind to accommodate a growing volume of criminal data over time without sacrificing system performance or responsiveness.

3.4.2 Performance Optimization

Implements performance optimization techniques to ensure efficient data retrieval, manipulation, and storage operations within the module, enhancing overall system performance.

CHAPTER 4

BACK END DESIGN

4.1 SQL SERVER (DATABASE)

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database. A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as row or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name. During an SQL Server Database design project, the analysis of your business change over time, you define any additional fields or change the definition of existing fields.

4.2 SQL Server Tables

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

4.2.1 Primary Key

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

4.2.2 Foreign Key

When a field in one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values

match those of the primary key of another table needs identifies all the fields or attributes of interest.

4.2.3 Apache

The Apache HTTP server software notable for playing a key role in the initial growth of the World Wide Web. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. In 2009 it became the first web server software to surpass the 100 million website milestone.

4.2.4 XAMPP

It is a small and light Apache distribution containing the most common web development technologies in a single package. Its contents, small size and portability make it the ideal tool for students developing and testing applications in PHP and MySQL. XAMPP is available as a free download in two specific packages: full and lite.

4.3 Triggers

In a DBMS, a trigger is a SQL procedure that initiates an action (i.e., fires an action) when an event (INSERT, DELETE or UPDATE) occurs. Since triggers are event-driven specialized procedures, they are stored in and managed by the DBMS. A trigger can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-driven and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

4.4 Stored Procedures

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. Uses for stored procedures include data-validation (integrated into the database) or access-control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. To save time and memory, extensive or complex processing that requires execution of

several SQL statements can be saved into stored procedures, and all applications call the procedures. One can use nested stored procedures by executing one stored procedure from within another. Stored procedures may return result sets, i.e., the results of a SELECT statement. Such result sets can be processed using cursors, by other stored procedures, by associating a result set locator, or by applications. Stored procedures may also contain declared variables for processing data and cursors that allow it to loop through multiple rows in a table. Stored procedure flow-control statements typically include IF, WHILE, LOOP, REPEAT and CASE statements, and more. Stored procedures can receive variables, return results or modify variables and return them, depending on how and where the variable is declared.

4.5 SCHEMA DIAGRAM

+-----+	+-----+	+-----+	+-----+
Criminal	Crime	Location	Case
+-----+	+-----+	+-----+	+-----+
id (PK)	id (PK)	id (PK)	id (PK)
name	name	name	criminal_id
gender	description	address	crime_id
age	+-----+	latitude	location_id
height		longitude	date
weight		+-----+	status
+-----+			+-----+

Fig. 4.5.1 Schema diagram of Criminal Database Management System



Fig. 4.5.2 E-R diagram of Criminal Database Management System

4.6 DATABASE TABLES

The database consists of one or more tables. Each table is made of rows and column. Each row in a relational uniquely by primary key. This can be by one or more sets of columns value in most of scenarios it is a single column.

Criminals Table: This table stores information about individual criminals

Column	Data Type	Description
id	INT	Primary Key, Auto-increment
name	VARCHAR(255)	Name of the criminal
age	INT	Age of the criminal
crime	VARCHAR(255)	Description of the crime
sentence	VARCHAR(255)	Sentence or penalty

TABLE 4.6.1

Case Table: This table stores details about criminal cases.

TABLE 4.6.2

Case Table:This table stores details about criminal cases.

Column	Data Type	Description
id	INT	Primary Key, Auto-increment
name	VARCHAR(255)	Name of the location
address	VARCHAR(255)	Address of the location
latitude	FLOAT	Latitude coordinate of location
longitude	FLOAT	Longitude coordinate of location

TABLE 4.6.3

Case Table:This table stores details about criminal cases.

Column	Data Type	Description
id	INT	Primary Key, Auto-increment
criminal_id	INT	Foreign Key referencing Criminals table
crime_id	INT	Foreign Key referencing Crime table
location_id	INT	Foreign Key referencing Location table

Column	Data Type	Description
id	INT	Primary Key, Auto-increment
name	VARCHAR(255)	Name of the crime
description	VARCHAR(255)	Description of the crime
date	DATE	Date of the crime
status	VARCHAR(50)	Status of the case (e.g., open, closed)

TABLE 4.6.4

CHAPTER 5

FRONT END DESIGN

In this chapter we have explained about the front end design tools such as Sublime Text Editor along with front end language PHP.

5.1 Sublime Text Editor

Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface (API). It actively supports many programming languages and markup languages, and functions can be added by users with plug-in, typically community-built and maintained under free-software licenses.

5.2 Java Script

JavaScript, often abbreviated as JS, is a high-level interpreted scripting language. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

5.3 HTML

Hyper Text Mark-up Language is the standard mark-up language for documents designed in a web browser. It can be assisted by technologies such as cascading style sheets and scripting languages such as JavaScript.

5.4 PHP (Hypertext Pre-processor)

Hypertext Pre-processor (or simply PHP) is a server-side scripting language designed for Web development, and also used as a general-purpose programming language. It was originally created by Rasmus Lerdorf in 1994. The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the

generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge. The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014 work has gone on to create a formal PHP specification.

5.5 Pseudo code

5.5.1 Admin Login Page

An Admin Login Page serves as the entry point for authorized administrators, providing a secure interface to access administrative functions. It verifies user identity through credentials authentication, restricting access to sensitive areas. Additionally, it typically incorporates security features like encryption and CAPTCHA to mitigate unauthorized access attempts. The page may also include a "Forgot Password" option for password recovery. Overall, the Admin Login Page plays a crucial role in maintaining system security and control.

5.5.1 Home Page

The home page serves as the initial landing point for users visiting a website or application, offering a glimpse into its content and functionalities. It often features navigation elements such as menus, buttons, or links to guide users to different sections. The home page may also showcase key information, promotions, or highlights to engage visitors and encourage further exploration. Additionally, it provides a platform for branding elements, including logos, slogans, and visual design, to establish the site's identity. Overall, the home page serves as a pivotal gateway, setting the tone for the user experience and facilitating seamless navigation throughout the platform.

5.5.2 Database

The criminal users' database stores pertinent information about individuals involved in criminal activities, facilitating law enforcement efforts and legal proceedings. It typically includes data such as names, aliases, physical descriptions, criminal records, and associated case details. Additionally, the database may store biographical

Criminal Database Management System

information, arrest records, and fingerprints for identification purposes. Security measures like encryption and access controls are often employed to safeguard sensitive data from unauthorized access. The database plays a crucial role in tracking criminal activities, aiding investigations, and ensuring public safety.

CHAPTER 6

CODES AND OUTCOMES

Login Page

```
from tkinter import *
def click():
    text_entry=textentry.get()
    if text_entry == "password" :
        print("Login Approved")
        root.destroy()
        import os
        os.system('front.py')
    else :
        print("Access Denied")
        labell=Label(bottomFrame,text="Wrong Password. Try again",fg="black",bg="red")
        labell.grid(row=1,column=1)

root = Tk()
root.title("Login Window")
root.geometry("620x520+100+00")
root.configure(background = "dark blue")

topFrame = Frame(root,bg="light blue")
topFrame.pack()
bottomFrame = Frame(root)
bottomFrame.pack(side =BOTTOM)
photo = PhotoImage(file = "dea2.gif")
Label(topFrame ,image = photo).grid(row=1 ,column = 1 ,sticky = W)
label = Label(topFrame, text="CRIMINAL \n RECORD \n DATABASE ",fg="black",bg="light blue",font=(None, 25)).grid(row=1,column = 3,sticky= E)
```

Fig 6.1 - Print Code For Login Page

```
Label(topFrame, text="Enter Password",fg="black",bg="light blue",font=(None, 15)).grid(row=4,column=3)
textentry = Entry(topFrame , width =30 ,fg="black" ,bg="white",show = "**")
textentry.grid(row=5, column = 3,sticky=W)
Button(topFrame ,text="Submit",width=10 ,command = click).grid(row=6,column=3,sticky=W)

def close():
    exit()
Label(bottomFrame,text="Caution: Information entered incorrectly by an agency \n representaion in Database will be reflected on the laboratory \n report. NYPD will make corrections requested by the customer at\n any point befo")
Button(bottomFrame,text="Click to Exit",width=10,command=close).grid(row = 7,column =1,sticky = S)

root.mainloop()
```

Fig 6.2 - Print Code For Login Page



The image shows a web browser window titled "Login Window". The main content area has a dark blue background. On the left, there is a large circular seal of the New York City Police Department (NYPD). The seal features a central shield with a scale of justice and a sword, surrounded by the words "THE GREATEST DETECTIVES" and "IN THE WORLD". Below the shield is a blue banner with "NYPD" in yellow. To the right of the seal, the text "CRIMINAL RECORD DATABASE" is displayed in large, bold, black capital letters. Below this text, there is a light blue rectangular area containing the text "Enter Password" above a white input field. A "Submit" button is located to the right of the input field. At the bottom of the page, there is a white rectangular box with red text that reads: "Caution: Information entered incorrectly by an agency representation in Database will be reflected on the laboratory report. NYPD will make corrections requested by the customer at any point before the laboratory report is issued." Below this box is a "Click to Exit" button.

Fig 6.3: Admin Login Page

In the above fig 6.3 shows the login page, if the user wants to login, they must enter valid password and click on Submit button.

Home page

```
from tkinter import *
import back
window=Tk()
pic=PhotoImage(file="logo.gif")
label2=Label(window,image=pic).grid(row=0,column=4,rowspan=7,columnspan=5)
pic1=PhotoImage(file="NYFD.gif")
#label2=Label(window,image=pic1).grid(row=7,column=4,rowspan=5,columnspan=5)
def viewcommand():
    list1.delete(0,END)
    for crime_record in back.view():
        list1.insert(END,crime_record)
def searchcommand():
    list1.delete(0,END)
    for crime_record in back.search(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get()):
        list1.insert(END,crime_record)
def insertcommand():
    back.insert(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get())
    list1.delete(0,END)
    list1.insert(END,(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get()))
```

Fig 6.4 - Print Code For home Page (front end)

```
def insertcommand():
    back.insert(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get())
    list1.delete(0,END)
    list1.insert(END,(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get()))
def get_selected_row(event):
    try:
        global selected_tuple
        index=list1.curselection()[0]
        selected_tuple=list1.get(index)
        e1.delete(0,END)
        e1.insert(END,selected_tuple[0])
        e2.delete(0,END)
        e2.insert(END,selected_tuple[1])
        e3.delete(0,END)
        e3.insert(END,selected_tuple[2])
        e4.delete(0,END)
        e4.insert(END,selected_tuple[3])
        e5.delete(0,END)
        e5.insert(END,selected_tuple[4])
        e6.delete(0,END)
        e6.insert(END,selected_tuple[5])
        e7.delete(0,END)
        e7.insert(END,selected_tuple[6])
        e8.delete(0,END)
        e8.insert(END,selected_tuple[7])
    except IndexError:
        pass
```

Fig 6.5 - Print Code For home Page (front end)

```
def deletecommand():
    back.delete(selected_tuple[0])
    viewcommand()
def updatecommand():
    back.update(Criminal_id_text.get(),Name_text.get(),Gender_text.get(),Nationality_text.get(),Age_text.get(),Height_text.get(),Weight_text.get(),Crime_Committed_text.get())
    searchcommand()
window.wm_title("CRIMINAL RECORD DATABASE")
l1=Label(window,text="Criminal_ID")
l1.grid(row=0,column=0)
l2=Label(window,text="Name")
l2.grid(row=0,column=2)
l3=Label(window,text="Gender")
l3.grid(row=1,column=0)
l4=Label(window,text="Nationality")
l4.grid(row=1,column=2)
```

Fig 6.6 - Print Code For home Page (front end)

```
l7=Label(window,text="Weight (kg)")
l7.grid(row=3,column=0)
l8=Label(window,text="Crime_Committed")
l8.grid(row=3,column=2)
Criminal_id_text=StringVar()
e1=Entry(window,textvariable=Criminal_id_text)
e1.grid(row=0,column=1)
Name_text=StringVar()
e2=Entry(window,textvariable=Name_text)
e2.grid(row=0,column=3)
Gender_text=StringVar()
e3=Entry(window,textvariable=Gender_text)
e3.grid(row=1,column=1)
```

Fig 6.7 - Print Code home Page (front end)

```
Age_text=StringVar()
e3=Entry(window,textvariable=Age_text)
e3.grid(row=2,column=1)

Height_text=StringVar()
e6=Entry(window,textvariable=Height_text)
e6.grid(row=2,column=3)

Weight_text=StringVar()
e7=Entry(window,textvariable=Weight_text)
e7.grid(row=3,column=1)

Crime_Committed_text=StringVar()
e8=Entry(window,textvariable=Crime_Committed_text)
e8.grid(row=3,column=3)

list1=Listbox(window,height=10,width=35)
list1.grid(row=4,column=0,rowspan=10,columnspan=2,pady=4)
list1.bind('<<ListboxSelect>>',get_selected_row)

scr=Scrollbar(window)
scr.grid(row=4,column=2,rowspan=6)
```

Fig 6.8 - Print Code For home Page (front end)

```
list1.configure(yscrollcommand=scr.set)
scr.configure(command=list1.yview)

b1=Button(window,text="View all",width=12,command=viewcommand)
b1.grid(row=4,column=3)

b2=Button(window,text="Search entry",width=12,command=searchcommand)
b2.grid(row=5,column=3)

b3=Button(window,text="Add entry",width=12,command=insertcommand)
b3.grid(row=6,column=3)

b4=Button(window,text="Update",width=12,command=updatecommand)
b4.grid(row=7,column=3)

b5=Button(window,text="Delete",width=12,command=deletecommand)
b5.grid(row=8,column=3)

b6=Button(window,text="Close",width=12,command=window.destroy)
b6.grid(row=9,column=3)

window.mainloop()
```

Fig 6.9 - Print Code For home Page (front end)

Back end

```
import sqlite3

def connect():
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS crime_record( Criminal_id PRIMARY KEY,text,Name text,Gender text,Nationality text,Age integer,Height float,Weight float,Crime_Committed text)")
    connection.commit()
    connection.close()

def insert(Criminal_id,Name,Gender,Nationality,Age,Height,Weight,Crime_Committed):
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("INSERT INTO crime_record VALUES (?, ?, ?, ?, ?, ?, ?, ?)",(Criminal_id , Name , Gender , Nationality , Age , Height , Weight , Crime_Committed))
    connection.commit()
    connection.close()

def view():
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("SELECT * FROM crime_record ")
    rows=cur.fetchall()
    connection.close()
    return rows
```

Fig 6.10 - Print Code For home Page (back end)

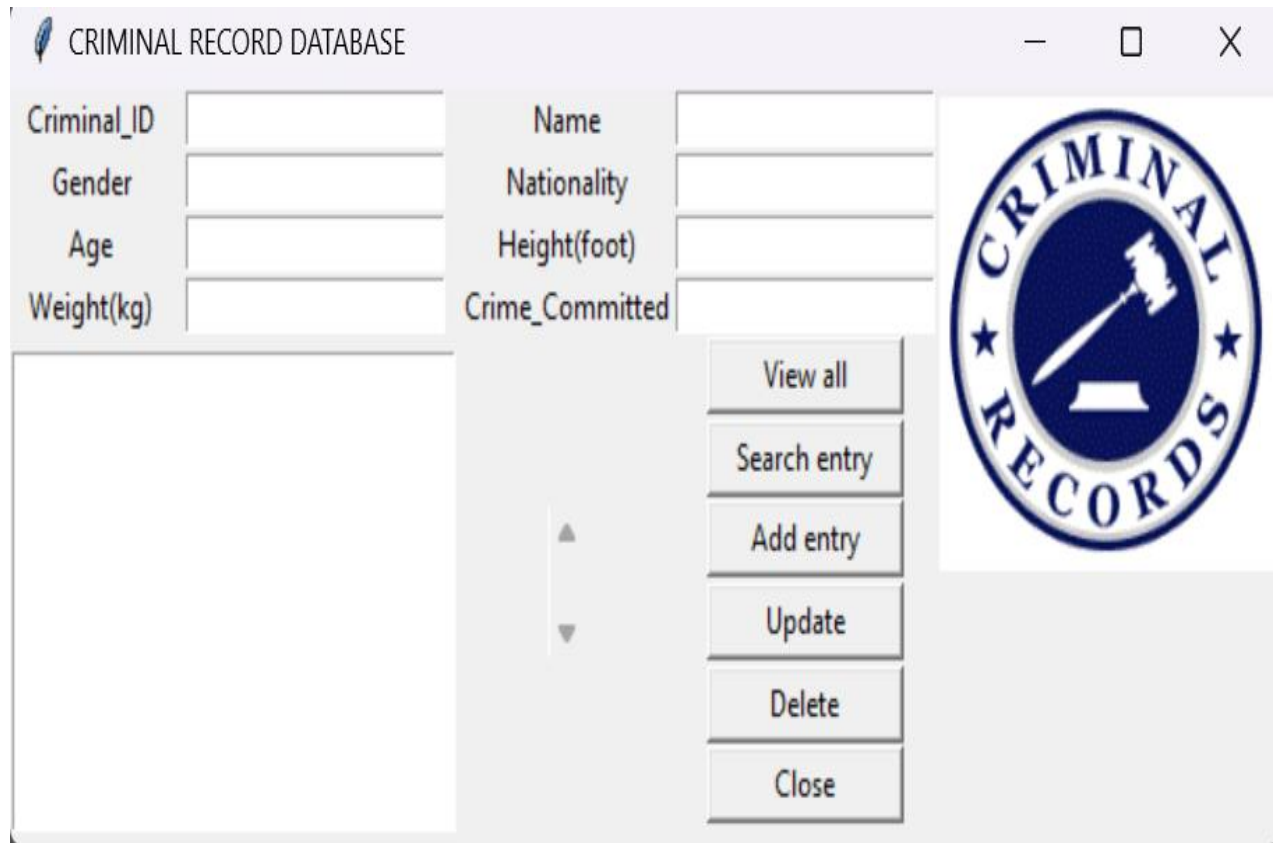
```
def search(Criminal_id="",Name="",Gender="",Nationality="",Age="",Height="",Weight="",Crime_Committed=""):
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("SELECT * FROM crime_record WHERE Criminal_id=? OR Name=? OR Gender=? OR Nationality=? OR Age=? OR Height=? OR Weight=? OR Crime_Committed=? ",(Criminal_id,Name,Gender,Nationality,Age,Height,Weight,Crime_Committed))
    rows=cur.fetchall()
    connection.close()
    return rows

def delete(Criminal_id):
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("DELETE FROM crime_record WHERE Criminal_id=?", (Criminal_id,))
    connection.commit()
    connection.close()

def update(Criminal_id,Name,Gender,Nationality,Age,Height,Weight,Crime_Committed):
    connection=sqlite3.connect("crime.db")
    cur=connection.cursor()
    cur.execute("UPDATE crime_record SET Criminal_id=?,Name=?,Gender=?,Nationality=?,Age=?,Height=?,Weight=?,Crime_Committed=? WHERE Criminal_id=?", (Criminal_id,Name,Gender,Nationality,Age,Height,Weight,Crime_Committed,Criminal_id))
    connection.commit()
    connection.close()

connect()
```

Fig 6.11 - Print Code For home Page (back end)



CRIMINAL RECORD DATABASE

Criminal_ID

Gender

Age

Weight(kg)

Name

Nationality

Height(foot)

Crime_Committed

CRIMINAL RECORDS

Fig 6.12: Home page

In the above fig 6.12 shows the home page information about the criminal database management system.

Criminal_id	Name	Gender	Nationality	Age	Height	Weight	Crime_Co...
412	fathima	female	indian	20	5.8	50	robbery
1924	kaushik	male	indian	20	5.8	50	murder
1947	chetan	male	indian	20	5.8	50	murder
2415	Harshita	female	indian	20	5.8	50	robbery
5242	sai kiran	male	indian	21	5.8	50	attempt to ...
5248	jagannath	male	indian	21	5.8	50	murder
6582	farha	female	indian	20	5.8	50	robbery

Fig 6.13: Database

In the above fig 6.13 shows the database information about the criminal database management system.

CHAPTER 7

FUTURE ENHANCEMENT

Future enhancements for a criminal database project can focus on improving functionality, scalability, security, and usability. Here are some potential areas for enhancement:

7.1 Advanced Search Functionality

- Implement advanced search capabilities allowing users to perform complex queries based on multiple criteria, such as location, criminal profiles, crime types, and case details.

7.2 Data Analysis and Visualization

- Integrate data analysis and visualization tools to generate insights and trends from the criminal data. This could include graphical representations of crime trends over time, geographic crime heatmaps, and demographic analyses.

7.3 Predictive Analytics

- Incorporate machine learning algorithms to predict crime hotspots, identify patterns in criminal behavior, and assist law enforcement agencies in proactive crime prevention strategies.

7.4 Geospatial Integration

- Enhance the database with geospatial capabilities to support geographic queries, spatial analysis, and mapping of crime data. This can aid in identifying crime patterns, allocating resources, and planning law enforcement strategies effectively.

7.5 Mobile Access

- Develop a mobile application or optimize the existing system for mobile access, allowing law enforcement officers to access and update criminal data in the field securely. This can improve operational efficiency and real-time data management.

7.6 Audit Trail and Compliance

- Strengthen audit trail capabilities to track all data access, modifications, and user activities within the system. Ensure compliance with regulatory requirements and data protection laws by implementing robust data governance practices.

7.7 Integration with External Databases

- Integrate the criminal database with external data sources, such as national crime databases, forensic databases, and law enforcement agencies' databases, to enhance data completeness and interoperability.

7.8 Enhanced Security Measures

- Implement advanced security measures, including multi-factor authentication, role-based access control, and encryption of sensitive data at rest and in transit, to mitigate cybersecurity risks and safeguard against unauthorized access.

7.9 Natural Language Processing (NLP)

- Integrate NLP capabilities to analyze unstructured data sources, such as witness statements, police reports, and court transcripts, to extract valuable insights and enhance the investigative process.

7.10 User Feedback Mechanism

- Incorporate a feedback mechanism to gather input from users, including law enforcement officers, administrators, and analysts, to continuously improve system usability, functionality, and user satisfaction.

CHAPTER 8

CONCLUSION OF THE PROJECT

Here's a conclusion for a criminal database management system:

In conclusion, the Criminal Database Management System (CDBMS) presented here offers a comprehensive solution for managing criminal records efficiently. With its intuitive interface and robust functionality, it streamlines the process of storing, retrieving, updating, and deleting criminal information.

The system allows law enforcement agencies, legal authorities, and other relevant parties to access critical data regarding criminals swiftly and accurately. By providing features such as adding new criminals, searching for specific individuals, updating their records, and removing outdated information, it ensures the database remains up-to-date and reliable.

Moreover, the integration of MySQL as the backend database and Python as the programming language offers flexibility, scalability, and ease of maintenance. The use of SQL queries enables efficient data manipulation, while Python's simplicity facilitates rapid development and customization of the system.

Overall, the Criminal Database Management System presented here serves as a valuable tool in the fight against crime by empowering stakeholders with timely and actionable insights into criminal activities. As technology continues to advance, further enhancements and refinements to the system can be made to meet the evolving needs of law enforcement and criminal justice systems worldwide.

REFERENCES

- Silberschatz Korth and Sudharshan, Database System Concepts, 6th Edition, McGraw Hill, 2013.
- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishna, and Gehrke, 3rd Edition, 2014, McGrawHill, 2013.
- Google
- <https://www.w3schools.com>
- <https://github.com>