HOSPITAL MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

PRIYADARSHINI M 220701206

RADHIKA RAKESH 220701208

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

RAJALAKSHMI
ENGINEERING COLLEGE

2023-2024

BONAFIDE CERTIFICATE

Certified that this project report "**HOSPITAL MANAGEMENT SYSTEM**"

is the bonafide work of

"PRIYADARSHINI M (220701206) & RADHIKA RAKESH (220701208)"

who carried out the project work under my supervision.

Submitted for the Practical Examination held on ______________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who has contributed to the successful completion of this mini project.

First and foremost, I am deeply thankful to my Professor **Mrs. K. Maheshmeena ,** my project advisor, for their invaluable guidance, insightful feedback, and continuous support throughout the duration of this mini project. Their expertise and encouragement have been instrumental in shaping my research and bringing this mini project to fruition.

I would also like to express my appreciation to the faculty and staff of the **Computer Science and Engineering** Department at **Rajalakshmi Engineering College** for providing the necessary resources and a conducive learning environment. We express our sincere thanks to **Dr. P. Kumar, M.E., Ph.D.,** Professor and Head of the Department Computer Science and Engineering for his guidance and encouragement throughout the project work.

My heartfelt thanks go to my peers and friends for their collaboration, constructive criticism, and moral support.

Thank you all for your contributions, both direct and indirect, to the success of this project.

ABSTRACT

Database Management Systems (DBMS) play a critical role in ensuring efficient data handling, improved patient care, and streamlined administration within hospital management. This project aims to develop a user-friendly, secure, and efficient hospital management system using Python for front end development and MySQL as the relational database management system. There are a lot of benefits involved such as Improved Efficiency(Streamline administrative tasks such as appointments, billing, record management to save time and resources),Enhanced Data Management (Organize and centralize hospital data for easier access, analysis, and decision-making), Increased Accuracy(Reduce errors by automating data entry and validation),Improved Patient Care(Facilitate better communication between healthcare professionals and patients), Scalability(The system can grow and adapt as the hospital's needs evolve) etc. The project will focus on core functionalities like patient registration, List of doctors, Services available, Appointment scheduling and management, Modifying existing data. This paper presents the development and implementation of a Hospital Management System (HMS) utilizing Python, a versatile and powerful programming language. It employs a relational DBMS called MySQL to store and manage patient data, ensuring data integrity, consistency, and accessibility. The use of Python facilitates seamless integration with the database, enabling dynamic querying, data manipulation, and real-time updates.

TABLE OF CONTENTS

S.NO	CONTENTS	PAGE NO
1.	**INTRODUCTION**	
1.1 Introduction		5
1.2 Objectives		5
1.3 Modules		7
2.	**SURVEY OF TECHNOLOGIES**	
2.1 Software Description		9
2.2 Languages		11
2.2.1 SQL		11
2.2.2 Python		12
3.	**REQUIREMENTS AND ANALYSIS**	
3.1 Requirement Specification		14
3.2 Hardware and Software Requirements		14
3.3 Architecture Diagram		15
3.4 ER Diagram		16
3.5 Normalization		17
4.	**PROGRAM CODE**	18
5.	**RESULTS AND DISCUSSION**	
6.	**CONCLUSION**	
7.	**REFERENCES**	

1.INTRODUCTION

1.1 INTRODUCTION:

Effective patient data management and administrative process management are critical in today's quickly changing healthcare environment. Every day, medical facilities produce enormous volumes of data, including clinical histories, appointment books, billing information, and patient records. To properly manage this data, you need reliable solutions that guarantee accessibility, security, and accuracy. These demands are met by the use of Database Management Systems (DBMS) in healthcare administration, which offer an organised method for storing, retrieving, and manipulating data. Several healthcare functions, including as patient registration, medical record administration, appointment scheduling, and billing procedures, are intended to be streamlined by the proposed HMS. The system makes sure that patient data is kept in an orderly and structured way by using a relational database management system (DBMS), which makes it easier to access and maintain. Complex queries and data linkages are supported by the relational model as well, which is essential for producing insightful reports and insights. The creation of this hospital management system is evidence of Python's ability to provide safe, scalable, and effective solutions for the healthcare sector. The HMS improves patient care and outcomes while also increasing operational efficiency by utilising Python's capabilities. In order to further expand the system's capabilities, future enhancements will concentrate on integrating cutting-edge technology, such as artificial intelligence and machine learning, to provide predictive analytics and decision support.

1.2 OBJECTIVES:

The creation of a Database Management System (DBMS) project for a hospital system aims to achieve several key objectives:

1. **Efficient Data Management:**
    - Develop a system that can efficiently store, retrieve, and manage large volumes of healthcare data, including patient records, appointment schedules, billing information, and clinical histories.
2. **Data Integrity and Consistency:**
    - Ensure that the data stored within the system is accurate, consistent, and reliable. This includes implementing mechanisms to prevent data duplication and maintain data integrity across various operations.

3. **Enhanced Accessibility and Usability:**
    - Create a user-friendly interface that allows healthcare providers and administrative staff to easily access and manage patient data. The interface should support intuitive navigation and efficient data entry and retrieval processes.
4. **Security and Privacy:**
    - Implement robust security measures to protect sensitive patient information. This includes data encryption, user authentication, and access control mechanisms to ensure compliance with regulatory standards such as HIPAA.
5. **Scalability:**
    - Design the system to be scalable, capable of handling increasing amounts of data and users without compromising performance. This includes optimizing database queries and system architecture for high efficiency.
6. **Interoperability:**
    - Ensure the system can integrate and communicate with other healthcare systems and standards, facilitating seamless data exchange and coordination of care. This includes support for common data formats and communication protocols.
7. **Comprehensive Reporting:**
    - Develop advanced reporting capabilities to generate meaningful insights from the data. This includes the ability to create various reports, such as patient summaries, appointment logs, and financial statements, to support decision-making.
8. **Real-time Data Processing:**
    - Implement features that allow for real-time data processing and updates, ensuring that healthcare providers have access to the most current information when making decisions about patient care.
9. **Automation of Administrative Tasks:**
    - Automate routine administrative tasks, such as appointment scheduling, billing, and notifications, to reduce the workload on healthcare staff and increase operational efficiency.
10. **Support for Predictive Analytics:**
    - Lay the groundwork for future integration of machine learning and artificial intelligence to provide predictive analytics and decision support, enhancing the system's ability to support proactive and personalized patient care.

By achieving these objectives, the DBMS project aims to improve the overall efficiency, accuracy, and quality of healthcare management, ultimately leading to better patient outcomes and streamlined administrative processes.

1.3 ENTITIES

PATIENT Attributes:

- PatientID
- Name Age
- Gender
- Phone Address
- Bloodgroup

APPOINTMENT Attributes:

- AppointmentID
- PatientID
- DoctorID
- Date Time

ROOM Attributes:

- RoomNumber
- DepartmentID
- OccupancyStatus

DOCTOR Attributes:

- DoctorID Name
- Gender
- Specialization
- DepartmentID
- RoomNumber
-

DEPARTMENT Attributes:

- DepartmentID
- Name
- HeadDoctorID
-

SERVICE Attributes:

- ServiceID
- Name
- Description
- DepartmentID

2. SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION:

Operating System	Windows 11 pro
Python Version	3.10.2
Database	MySQL
Pycharm IDE	For Project Creation
mysql-connector-python	For database connectivity with the program

Key components:

1. **Database Layer:**
    - **DBMS:** The system uses a relational DBMS such as MySQL to store and manage hospital data. The relational model ensures data integrity, supports complex queries, and maintains relationships between different data entities (e.g., patients, doctors, appointments).
    - **Data Schema:** The database schema includes tables for patients, doctors, appointments, medical records, billing, and staff, with defined relationships and constraints to ensure data consistency.
2. **Backend Layer:**
    - **Python Framework:** The backend is developed using Python with a web framework such as Flask or Django, which provides a robust environment for building web applications.
3. **Frontend Layer:**
    - **Web Interface:** The frontend is built using Python.
    - **User Roles:** The system supports multiple user roles (e.g., doctors, nurses, administrative staff, and patients) with role-based access controls to ensure that users have access only to relevant data and functionalities.
4. **Security Measures:**
    - **Authentication and Authorization:** The system employs secure authentication mechanisms
    - **Data Encryption:** Sensitive data, both at rest and in transit, is encrypted using industry-standard encryption algorithms to prevent unauthorized access.
    - **Audit Logs:** The system maintains detailed logs of all access and modifications to data, providing an audit trail for security and compliance purposes.

5. **Functional Modules:**
    - **Patient Management:** Handles patient registration, demographic information, medical history, and appointment scheduling.
    - **Doctor Management:** Manages doctor profiles, specialties, schedules, and appointment availability.
    - **Appointment Scheduling:** Facilitates the booking, rescheduling, and cancellation of appointments, with notifications for patients and doctors.
    - **Medical Records:** Stores and manages detailed medical records, including diagnoses, treatments, prescriptions, and lab results.
    - **Billing and Payments:** Automates the billing process, generating invoices, processing payments, and managing insurance claims.
    - **Reporting and Analytics:** Provides tools for generating reports on various metrics and analyzing data for informed decision-making.
6. **Interoperability:**
    - **APIs for External Access:** Exposes APIs for third-party applications to access and interact with the system, enabling broader interoperability.

Development tools:

- **Programming Language:** Python
- **Database:** MySQL
- **Frontend Technologies:** Python

2.2. LANGUAGES:

Front-End: The system will utilize a user-friendly interface developed with a popular programming language like:

- Python

2.2.1 Python

User Interface Design:

- **Emphasis on User Experience:** Design a visually appealing and user-friendly interface with clear labels, intuitive navigation, and well-organized menus.

- **Data Input Forms:** Create forms with appropriate input fields (text boxes, drop-down menus, date pickers) for different information categories (name, address, phone number, etc.).
- **Data Interaction Features:** Implement functionalities like:
  - **User Registration:** Allow users to create accounts with secure password hashing.
  - **Login:** Provide a secure login mechanism for user authentication.
  - **Data Entry:** Enable users to add new information to their PIMS database.
  - **Data Editing:** Allow users to modify existing information.
  - **Data Deletion:** Provide a way to securely delete information with confirmation to prevent accidental loss.
- **Search and Filtering:** Implement functionalities to search for specific information within the user's data based on keywords or criteria (e.g., contact name, date range).

Database Connectivity

- **Libraries:** Utilize libraries like MySQL dB for MySQL or sqlite3 for SQLite to connect to your chosen DBMS.
- **Data Transfer:**
  - **Sending Data:** When users interact with the interface (adding information, searching), the front-end will use the chosen library to send data manipulation requests (INSERT, UPDATE, DELETE) or queries (SELECT) to the back-end DBMS.
  - **Receiving Data:** The back-end will process the requests and send back the requested information or confirmation of successful operations.exclamation The front-end will then display the retrieved data or confirmation messages to the user.

Benefits of using Python with DBMS:

- **Ease of Development**
- **Large Community**
- **Database Connectivity Libraries:** Established libraries simplify communication with various DBMS options.
- **Focus on User Interface:** Python allows you to focus on creating a user-friendly interface without getting bogged down in complex coding.

Back-End: Data will be stored and managed in a secure database. The choice between a relational database management system (RDBMS) like MySQL.

2.2.2 MySQL

As a popular and easy-to-use RDBMS, MySQL is a strong choice for most personal information management systems. It offers good performance, scalability, and a large user community for support.

Implementation with MySQL:

- **MySQL Server:** You'll need to install and configure a MySQL server to act as the back-end database storage.
- **Database Creation:** Use tools like MySQL Workbench or command-line tools to create the database based on your designed schema.
- **Table Creation:** Define tables within the database, specifying data types for each attribute (e.g., text for names, numbers for phone numbers).
- **Data Manipulation:** Develop functionalities to insert, update, and delete data within the tables using Structured Query Language (SQL). This can be done through code within your back-end application or using tools like MySQL Workbench.

Benefits of RDBMS:

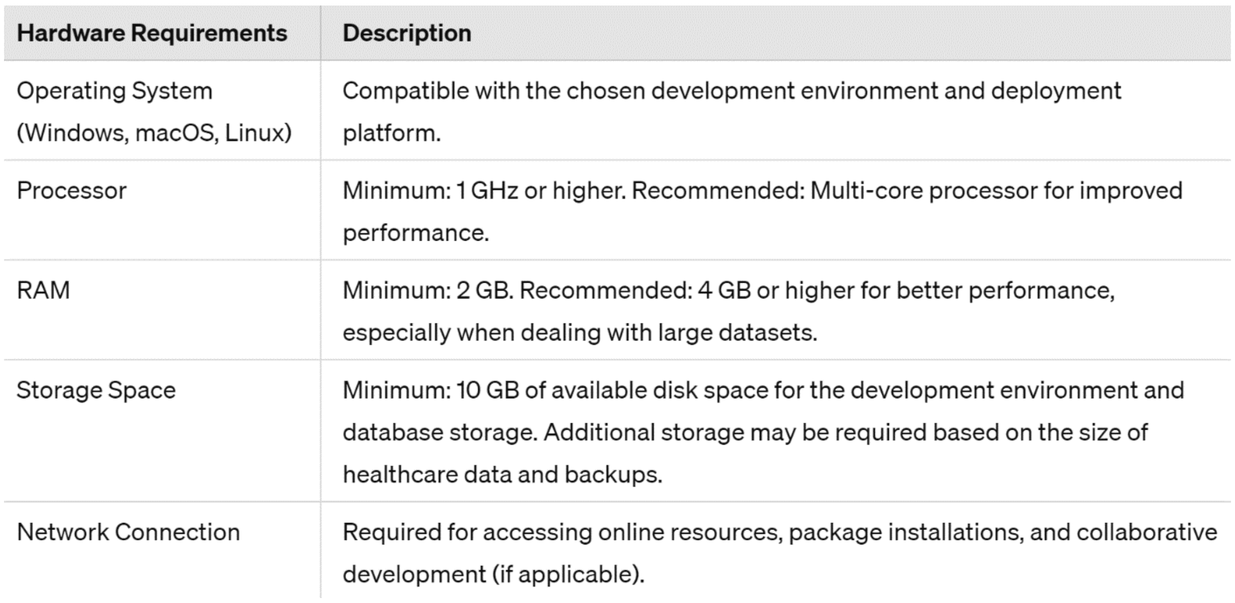
- **Organization:** Structured data storage allows for efficient organization and retrieval of personal information.
- **Data Integrity:** RDBMS enforces data integrity rules to minimize inconsistencies and errors.
- **Scalability:** MySQL can handle growing amounts of data as your information needs evolve.
- **Security:** Offers various security features like user authentication and access control.

3.REQUIREMENT ANALYSIS

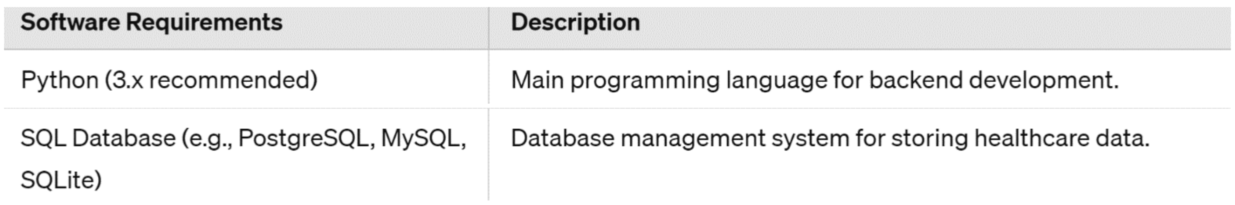
3.1 REQUIREMENT SPECIFICATION:

The Hospital Management System (HMS) project, developed using Python and SQL, aims to provide a comprehensive solution for efficiently managing healthcare-related data within a healthcare facility. Leveraging technologies like Flask or Django for backend development, SQL databases for data storage, and Docker for deployment, the HMS project promises to streamline healthcare operations, enhance patient care, and optimize resource utilization within healthcare facilities.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

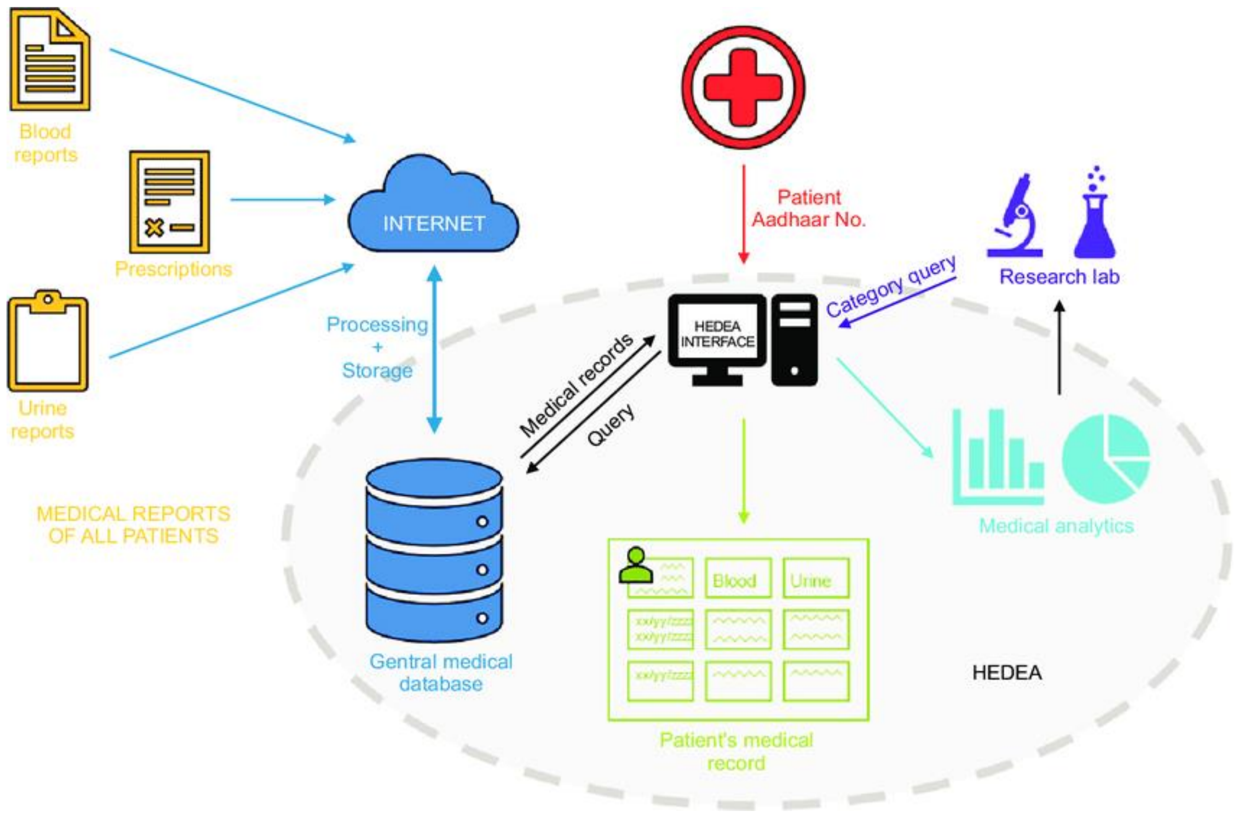


Hardware Requirements	Description
Operating System (Windows, macOS, Linux)	Compatible with the chosen development environment and deployment platform.
Processor	Minimum: 1 GHz or higher. Recommended: Multi-core processor for improved performance.
RAM	Minimum: 2 GB. Recommended: 4 GB or higher for better performance, especially when dealing with large datasets.
Storage Space	Minimum: 10 GB of available disk space for the development environment and database storage. Additional storage may be required based on the size of healthcare data and backups.
Network Connection	Required for accessing online resources, package installations, and collaborative development (if applicable).

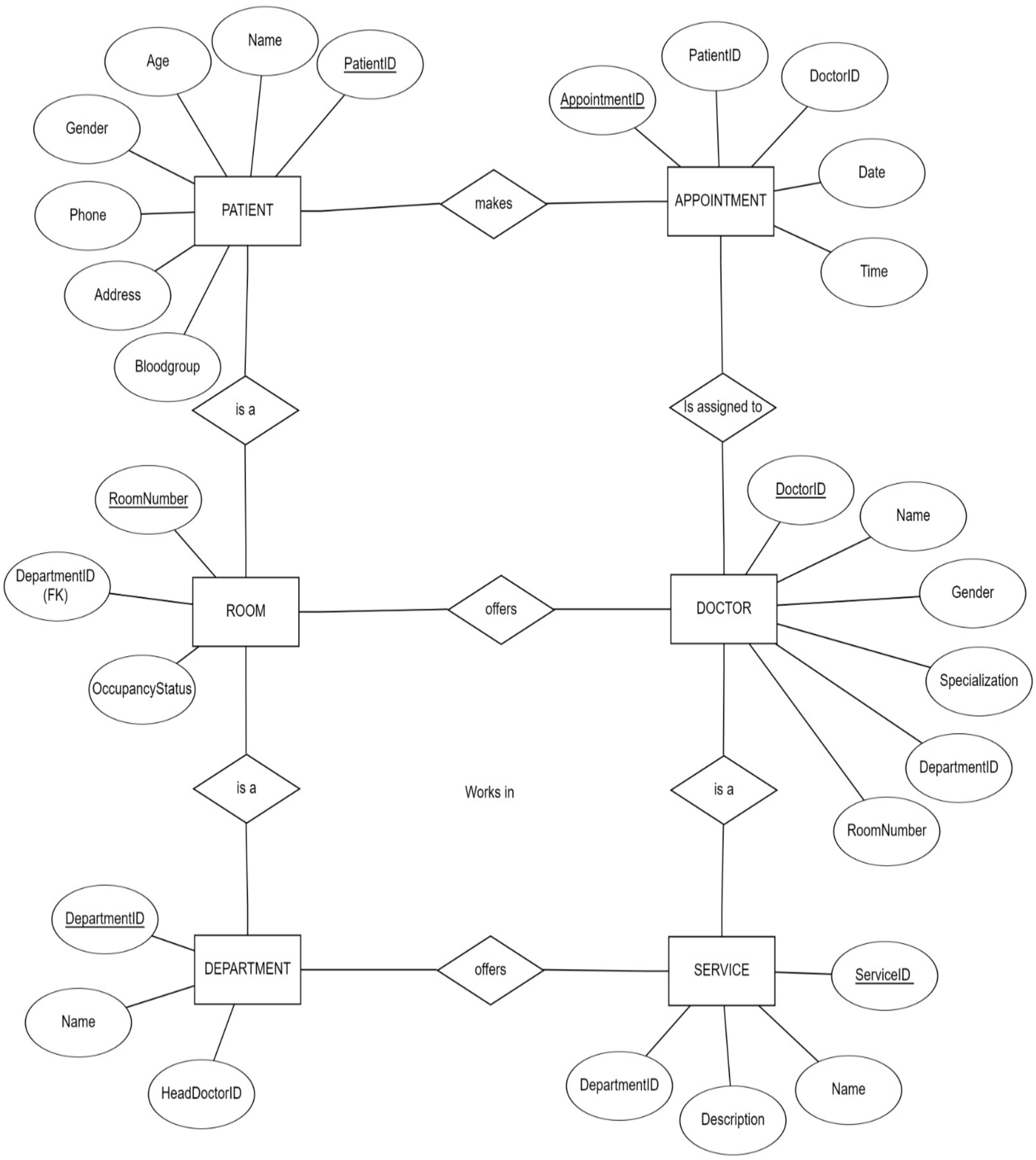


Software Requirements	Description
Python (3.x recommended)	Main programming language for backend development.
SQL Database (e.g., PostgreSQL, MySQL, SQLite)	Database management system for storing healthcare data.

3.3 ARCHITECTURE DIAGRAM:



3.4 ER DIAGRAM:



An Entity-Relationship (ER) diagram is a visual representation of the entities in a database and the relationships between them. For a Hospital Management System, the ER diagram helps to model the various components and their sinteractions within the system.

3.5 NORMALIZATION:

Normalizing the Hospital Management System

Let's take a scenario for the Hospital Management System and normalize it progressively through 1NF, 2NF, 3NF, and BCNF.

Scenario:

We have a table named "Patient_Info" with the following attributes:

- PatientID (Primary Key)
- Patient Name
- Date of Birth
- Address
- Phone Number
- Doctor Name
- Department

Normalization Steps:

1. **1NF (First Normal Form):** The table already adheres to 1NF as it has a unique identifier (PatientID) for each record, and there are no repeating groups of data within a single column.
2. **2NF (Second Normal Form):** The table violates 2NF. The attributes "Doctor Name" and "Department" are partially dependent on the primary key "PatientID." "Doctor Name" might also depend on "Department."

Solution:

- Create a new table named "Doctor_Info" with attributes:
  - DoctorID (Primary Key)
  - Doctor Name
  - Department
- Modify the "Patient_Info" table:
  - Keep PatientID (Primary Key)
  - Patient Name
  - Date of Birth
  - Address
  - Phone Number
  - DoctorID (Foreign Key referencing DoctorID in "Doctor_Info")

This removes the partial dependency on the primary key in the "Patient_Info" table.

3. **3NF (Third Normal Form):** The current structure satisfies 3NF. There are no transitive dependencies. No attribute in either table depends on another non-key attribute through another attribute.

BCNF (Boyce-Codd Normal Form):

In this scenario, achieving 3NF is sufficient. BCNF is a stricter form that eliminates specific anomalies not captured by 3NF. For most hospital management systems, 3NF provides a good balance between data integrity and avoiding unnecessary complexity.

By normalizing the table, we achieve:

- Reduced data redundancy: Eliminates storing the same doctor information for multiple patients.
- Improved data integrity: Updates to doctor information in "Doctor_Info" are reflected across all patients associated with that doctor.
- Easier data manipulation: Allows for independent updates to patient and doctor information.

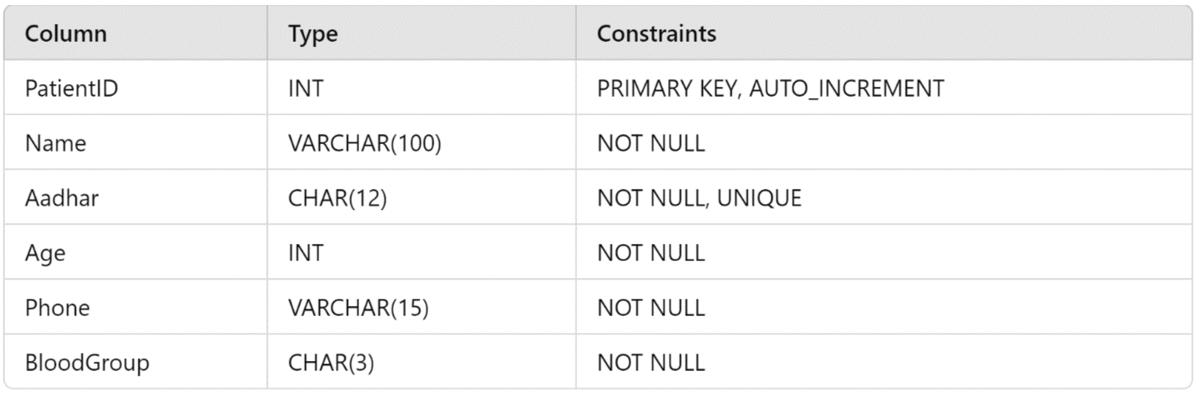
This normalization approach streamlines data management within the Hospital Management System.

4. 4NF (Fourth Normal Form):

- Addresses multi-valued dependencies, where a single record can have multiple sets of related values.

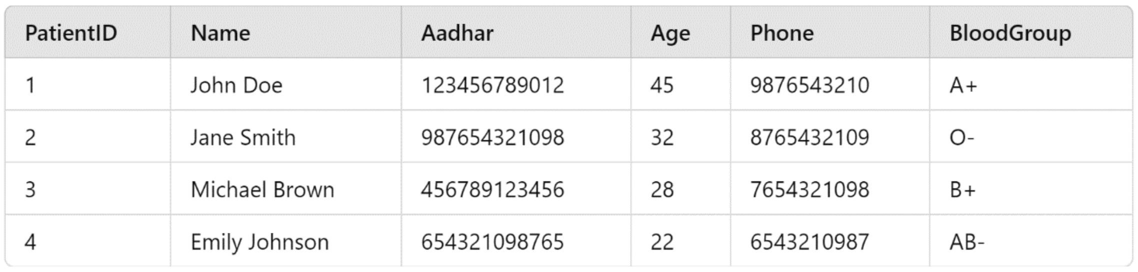
Potential Solution (For Referential Integrity):

We can introduce a new attribute "Role" in the "Admissions_Doctor" table (This clarifies the role of each doctor associated with the admission. 4NF and 5NF are more theoretical and less commonly applied in practical database design for hospital management systems. Achieving them might introduce complexity that outweighs the benefits.



Column	Type	Constraints
PatientID	INT	PRIMARY KEY, AUTO_INCREMENT
Name	VARCHAR(100)	NOT NULL
Aadhar	CHAR(12)	NOT NULL, UNIQUE
Age	INT	NOT NULL
Phone	VARCHAR(15)	NOT NULL
BloodGroup	CHAR(3)	NOT NULL

SAMPLE DATA VISUALIZATION:



PatientID	Name	Aadhar	Age	Phone	BloodGroup
1	John Doe	123456789012	45	9876543210	A+
2	Jane Smith	987654321098	32	8765432109	O-
3	Michael Brown	456789123456	28	7654321098	B+
4	Emily Johnson	654321098765	22	6543210987	AB-

This table structure allows you to store and manage patient information efficiently, ensuring that each patient has a unique identifier and that critical information like Aadhar numbers and phone numbers are unique and properly constrained.

4. PROGRAM CODE:

PYTHON CODE:

```
import tkinter.messagebox

from tkinter import  *

import mysql.connector as sqlcon

import random as rd


con=sqlcon.connect(host="127.0.0.1",user="root",password="root")#connection to mysql

cur = con.cursor(buffered=True)

if (cur):

   # Carry out normal procedure

   print ("Connection successful")

else:

   print ("Connection unsuccessful")

cur.execute("create database if not exists Hospital")

cur.execute("use Hospital")

cur.execute("create table if not exists appointment"

          "("

          "idno varchar(12) primary key,"

          "name char(50),"

          "age char(3),"

          "gender char(1),"

          "phone varchar(10),"

          "bg varchar(3))")

cur.execute("create table if not exists appointment_details"

          "("

          "idno varchar(12) primary key,"

          "doctor varchar(50),"

          "date varchar(20),"
```

```
        "time varchar(20),"
        "appointment_no varchar(10))")
#  Message for registration
def entry():
   global e1,e2,e3,e4,e5,e6
   p1=e1.get()
   p2=e2.get()
   p3=e3.get()
   p4=e4.get()
   p5=e5.get()
   p6=e6.get()
   query='insert into appointment values("{}", "{}", "{}", "{}", "{}", "{}")'.format(p1,p2,p3,p4,p5,p6)
   con.commit()
   cur.execute(query)

   tkinter.messagebox.showinfo("DONE", "YOU HAVE BEEN REGISTERED")
#  For registration
def register():
   global e1,e2,e3,e4,e5,e6
   root1=Tk()
   label=Label(root1,text="REGISTER YOURSELF",font='arial 25 bold')
   label.pack()
   frame=Frame(root1,height=500,width=200)
   frame.pack()
   l1=Label(root1,text="AADHAR CARD NO.")
   l1.place(x=10,y=130)
   e1=tkinter.Entry(root1)
   e1.place(x=100,y=130)
   l2=Label(root1,text="NAME")
   l2.place(x=10,y=170)
```

```
    e2=tkinter.Entry(root1)
    e2.place(x=100,y=170)
    l3=Label(root1,text="AGE")
    l3.place(x=10,y=210)
    e3=tkinter.Entry(root1)
    e3.place(x=100,y=210)
    l4=Label(root1,text="GENDER M\F")
    l4.place(x=10,y=250)
    e4=tkinter.Entry(root1)
    e4.place(x=100,y=250)
    l5=Label(root1,text="PHONE")
    l5.place(x=10,y=290)
    e5=tkinter.Entry(root1)
    e5.place(x=100,y=290)
    l6=Label(root1,text="BLOOD GROUP")
    l6.place(x=10,y=330)
    e6=tkinter.Entry(root1)
    e6.place(x=100,y=330)
    b1=Button(root1,text="SUBMIT",command=entry)
    b1.place(x=150,y=370)



    root.resizable(False,False)
    root1.mainloop()

# Message for appointment
def apo_details():
    global x1,x2,h,p1,p2,p3,o,x4,x3
    p1=x2.get()
    p2=x3.get()
```

```
p3=x4.get()
if int(p1)==1:
   i=("Dr. sharma \nRoom no:- 10")
   j=("Dr. Verma \nRoom no:- 11")
   q=(i,j)
   h=rd.choice(q)
   u=(23,34,12,67,53,72)
   o=rd.choice(u)
   det = (
   "Your appointment is fixed with: {}\n"
   "Date: {}\n"
   "Time: {}\n"
   "Appointment no: {}"
    ).format(h, p2, p3, o)
   query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
   cur.execute(query)
   tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)

elif int(p1)==2:
   i=("Dr. Sidharth \nRoom no. 16")
   j=("Dr. Tendulkar \nRoom no. 17")
   q=(i,j)
   h=rd.choice(q)
   u=(23,34,12,67,53,72)
   o=rd.choice(u)
   det = (
   "Your appointment is fixed with: {}\n"
   "Date: {}\n"
   "Time: {}\n"
   "Appointment no: {}"
```

```
     ).format(h, p2, p3, o)
    query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
    cur.execute(query)
    tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)


elif int(p1)==3:
    i=("Dr. Kumar \nRoom no. 12")
    j=("Dr. Khan \nRoom no. 13")
    q=(i,j)
    h=rd.choice(q)
    u=(23,34,12,67,53,72)
    o=rd.choice(u)
    det = (
    "Your appointment is fixed with: {}\n"
    "Date: {}\n"
    "Time: {}\n"
    "Appointment no: {}"
     ).format(h, p2, p3, o)
    query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
    cur.execute(query)
    tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)
elif int(p1)==4:
    i=("Dr. Virat, \nRoom no. 18")
    j=("Dr. Leo \nRoom no. 19")
    q=(i,j)
    h=rd.choice(q)
    u=(23,34,12,67,53,72)
    o=rd.choice(u)
    det = (
    "Your appointment is fixed with: {}\n"
```

```
        "Date: {}\n"
        "Time: {}\n"
        "Appointment no: {}"
         ).format(h, p2, p3, o)
        query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
        cur.execute(query)
        tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)
    elif int(p1)==5:
        i=("Dr. Kohli \nRoom no. 14")
        j=("Dr. singh \nRoom no. 15")
        q=(i,j)
        h=rd.choice(q)
        u=(23,34,12,67,53,72)
        o=rd.choice(u)
        det = (
        "Your appointment is fixed with: {}\n"
        "Date: {}\n"
        "Time: {}\n"
        "Appointment no: {}"
         ).format(h, p2, p3, o)
        query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
        cur.execute(query)
        tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)
    elif int(p1)==6:
        i=("Dr. Irfan \nRoom no. 001")
        j=("Dr. John \nRoom no. 002")
        k=("Dr. Sanjay \nRoom no. 003")
        l=("Dr. Shahid \nRoom no. 004")
        q=(i,j,k,l)
        h=rd.choice(q)
        u=(23,34,12,67,53,72)
        o=rd.choice(u)
        det = (
        "Your appointment is fixed with: {}\n"
        "Date: {}\n"
        "Time: {}\n"
        "Appointment no: {}"
         ).format(h, p2, p3, o)
        query='insert into appointment_details values("{}", "{}", "{}", "{}", "{}")'.format(p1,h,p2,p3,o)
        cur.execute(query)
        tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)
    else:
        tkinter.messagebox.showwarning('WRONG INPUT','PLEASE ENTER VALID VALUE')

#  For appointment
def get_apoint():
    global x1,x2,x3,x4
    p1=x1.get()
    cur.execute('select * from appointment where idno=(%s)',(p1,))
    dat=cur.fetchall()
    a=[]
    for i in dat:
        a.append(i)
    if len(a)==0:
        tkinter.messagebox.showwarning("ERROR", "NO DATA FOUND!!")
    else:
        root3=Tk()
        label=Label(root3,text="APPOINTMENT",font='arial 25 bold')
        label.pack()
        frame=Frame(root3,height=500,width=300)
```

```
frame.pack()
if i[3]=='M' or i[3]=='m':
    x="Mr."
    name2=Label(root3,text=i[1])
    name2.place(x=140,y=80)
else:
    x="Mrs\Ms."
    name2=Label(root3,text=i[1])
    name2.place(x=170,y=80)
for i in dat:
  name=Label(root3,text='WELCOME')
  name.place(x=50,y=80)
  name1=Label(root3,text=x)
  name1.place(x=120,y=80)
  age=Label(root3,text='AGE:-')
  age.place(x=50,y=100)
  age1=Label(root3,text=i[2])
  age1.place(x=100,y=100)
  phone=Label(root3,text='PHONE:-')
  phone.place(x=50,y=120)
  phone1=Label(root3,text=i[4])
  phone1.place(x=100,y=120)
  bg=Label(root3,text='BLOOD GROUP:-')
  bg.place(x=50,y=140)
  bg1=Label(root3,text=i[5])
  bg1.place(x=150,y=140)
L=Label(root3,text='DEPARTMENTS')
L.place(x=50,y=220)
L1=Label(root3,text="1.Orthopaedic surgeon ")
L1.place(x=50,y=250)
```

```
        L2=Label(root3,text='2.Physician')
        L2.place(x=50,y=270)
        L3=Label(root3,text='3.Nephrologist')
        L3.place(x=50,y=290)
        L4=Label(root3,text='4.Neurologist')
        L4.place(x=50,y=310)
        L5=Label(root3,text='5.Gynaecologist')
        L5.place(x=50,y=330)
        L6=Label(root3,text='6.X-ray')
        L6.place(x=50,y=350)
        L7=Label(root3,text='Enter your choice')
        L7.place(x=100,y=370)
        x2=tkinter.Entry(root3)
        x2.place(x=200,y=370)

        L7=Label(root3,text=('enter date')).place(x=100,y=400)
        x3=tkinter.Entry(root3)
        x3.place(x=200,y=400)
        L8=Label(root3,text=('enter time in 24 hour format')).place(x=48,y=430)
        x4=tkinter.Entry(root3)
        x4.place(x=200,y=430)

        B1=Button(root3,text='Submit',command=apo_details)
        B1.place(x=120,y=480)
        root3.resizable(False,False)
        root3.mainloop()

#  For AADHAAR
def apoint():
    global x1
```

```
   root2=Tk()

   label=Label(root2,text="APPOINTMENT",font='arial 25 bold')

   label.pack()

   frame=Frame(root2,height=200,width=200)

   frame.pack()

   l1=Label(root2,text="AADHAAR NO.")

   l1.place(x=10,y=130)

   x1=tkinter.Entry(root2)

   x1.place(x=100,y=130)

   b1=Button(root2,text='Submit',command=get_apoint)

   b1.place(x=100,y=160)

   root2.resizable(False,False)

   root2.mainloop()


#  List of doctors

def lst_doc():

   root4=Tk()


   l=["Dr. sharma","Dr. Verma","Dr. Kumar","Dr. Khan","Dr. Kohli","Dr. singh","Dr. Sidharth","Dr.
tendulkar","Dr. Virat","Dr. Leo",'Dr. Irfan','Dr. John',

      'Dr. Sanjay','Dr. Shahid']

   m=["Orthopaedic surgeon","Orthopaedic
surgeon","Nephrologist","Nephrologist","Gynaecologist","Gynaecologist","Physician","Physician","Neu
rologist",

      "Neurologist",'X-ray','X-ray','X-ray','X-ray']

   n=[10,11,12,13,14,15,16,17,18,19,20,21,22,23]

   frame=Frame(root4,height=500,width=500)

   frame.pack()




   l1=Label(root4,text='NAME OF DOCTORS')
```

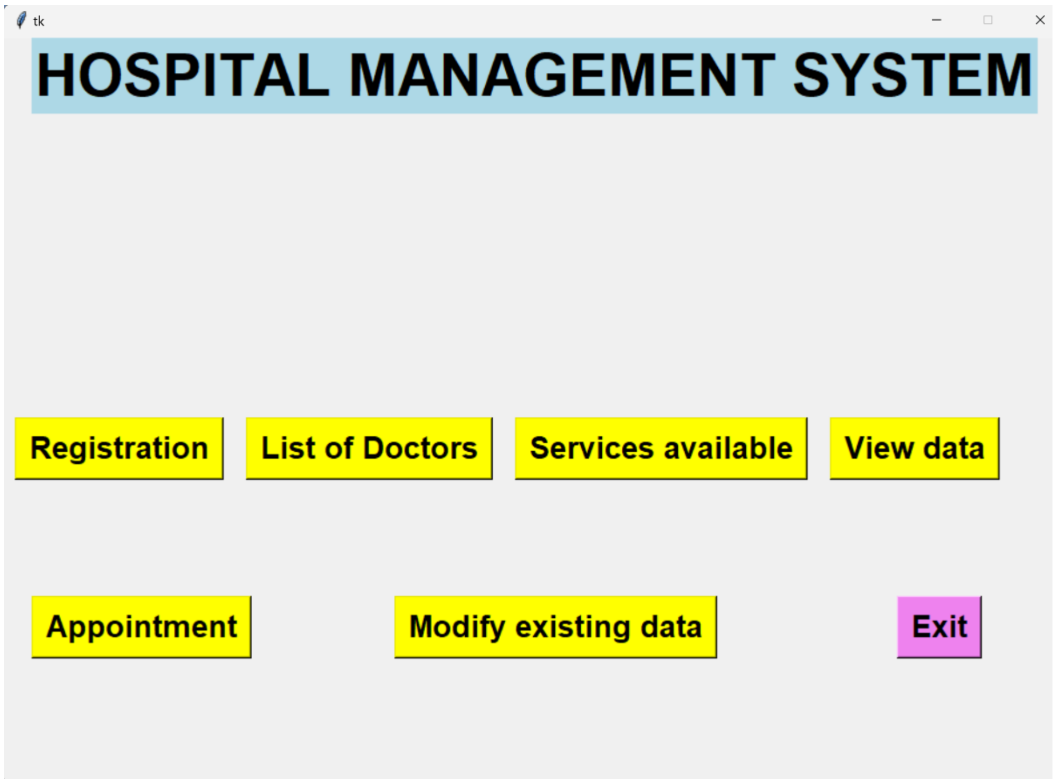
```
    l1.place(x=20,y=10)
    count=20
    for i in l:
      count=count+20
      l=Label(root4,text=i)
      l.place(x=20,y=count)
    l2=Label(root4,text='DEPARTMENT')
    l2.place(x=140,y=10)
    count1=20
    for i in m:
      count1=count1+20
      l3=Label(root4,text=i)
      l3.place(x=140,y=count1)
    l4=Label(root4,text='ROOM NO')
    l4.place(x=260,y=10)
    count2=20
    for i in n:
      count2=count2+20
      l5=Label(root4,text=i)
      l5.place(x=260,y=count2)
    root.resizable(False,False)
    root4.mainloop()
def ser_avail():


    root5=Tk()
    frame=Frame(root5,height=500,width=500)
    frame.pack()
    l1=Label(root5,text='SERVICES AVAILABLE')
    l1.place(x=20,y=10)
    f=["ULTRASOUND","X-RAY","CT Scan","MRI","BLOOD
COLLECTION","DIALYSIS","ECG","CHEMIST","LAB"]
```

```
    count1=20
    for i in f:
        count1=count1+20
        l3=Label(root5,text=i)
        l3.place(x=20,y=count1)
    l2=Label(root5,text='ROOM NO.')
    l2.place(x=140,y=10)
    g=[1,2,3,4,5,6,7,8,9]
    count2=20
    for i in g:
        count2=count2+20
        l4=Label(root5,text=i)
        l4.place(x=140,y=count2)
    l5=Label(root5,text='To avail any of these please contact on our no.:- 94887-43479')
    l5.place(x=20,y=240)
    root5.resizable(False,False)
    root5.mainloop()

def modify():
    global x3,x4,choice,new,x5,root6
    p1=x3.get()
    cur.execute('select * from appointment where idno=(%s)',(p1,))

    dat=cur.fetchall()
    a=[]
    for i in dat:
        a.append(i)
    if len(a)==0:
        tkinter.messagebox.showwarning("ERROR", "NO DATA FOUND!!")
    else:
```

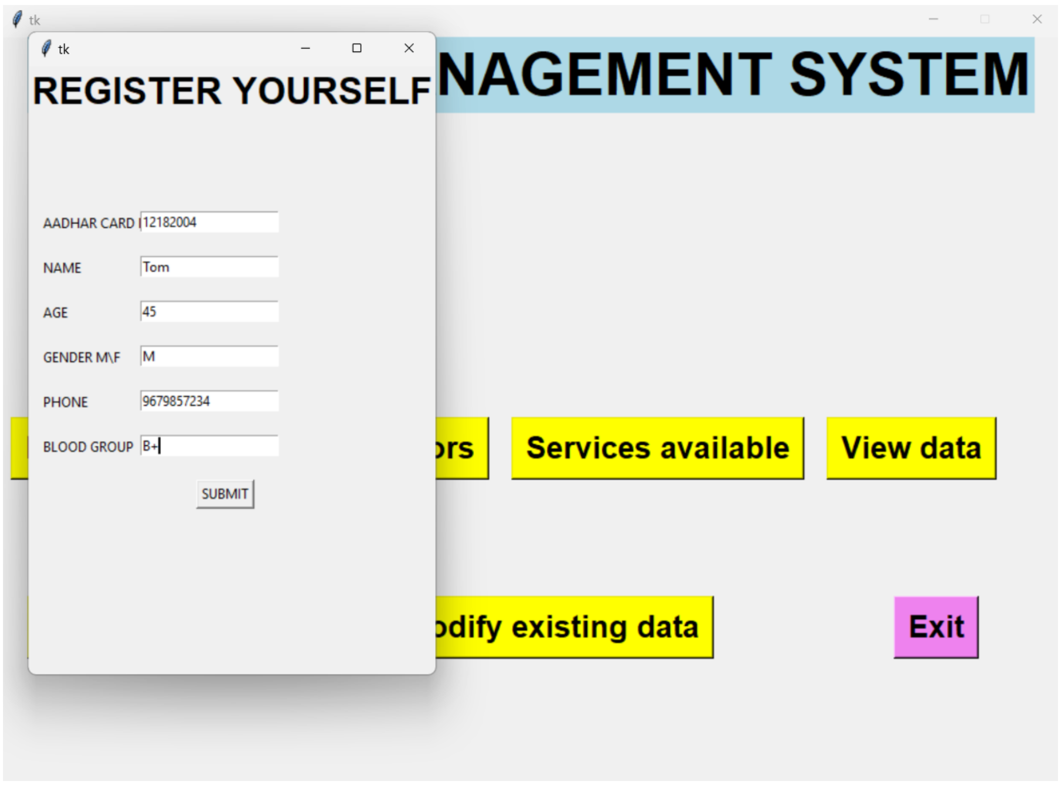
```
root6=Tk()
frame=Frame(root6,height=500,width=500)
frame.pack()
l1=Label(root6,text='DATA MODIFICATION',font="arial 15 bold")
l1.place(x=75,y=10)
l2=Label(root6,text='WHAT YOU WANT TO CHANGE')
l2.place(x=50,y=200)
l3=Label(root6,text='1.NAME')
l3.place(x=50,y=220)
l4=Label(root6,text='2.AGE')
l4.place(x=50,y=240)
l5=Label(root6,text='3.GENDER')
l5.place(x=50,y=260)
l6=Label(root6,text='4.PHONE')
l6.place(x=50,y=280)
l7=Label(root6,text='5.BLOOD GROUP')
l7.place(x=50,y=300)
x2=Label(root6,text='Enter')
x2.place(x=50,y=330)
x4=tkinter.Entry(root6)
choice=x4.get()
x4.place(x=100,y=330)
for i in dat:
    name=Label(root6,text='NAME:-')
    name.place(x=50,y=80)
    name1=Label(root6,text=i[1])
    name1.place(x=150,y=80)
    age=Label(root6,text='AGE:-')
    age.place(x=50,y=100)
    age1=Label(root6,text=i[2])
```

```
            age1.place(x=150,y=100)
            gen=Label(root6,text='GENDER:-')
            gen.place(x=50,y=120)
            gen1=Label(root6,text=i[3])
            gen1.place(x=150,y=120)
            pho=Label(root6,text='PHONE:-')
            pho.place(x=50,y=140)
            pho1=Label(root6,text=i[4])
            pho1.place(x=150,y=140)
            bg=Label(root6,text='BLOOD GROUP:-')
            bg.place(x=50,y=160)
            bg1=Label(root6,text=i[5])
            bg1.place(x=150,y=160)
        b=Button(root6,text='Submit',command=do_modify)
        b.place(x=50,y=400)
        L1=Label(root6,text='OLD DETAILS')
        L1.place(x=50,y=50)
        L2=Label(root6,text='ENTER NEW DETAIL')
        L2.place(x=50,y=360)
        x5=tkinter.Entry(root6)
        new=x5.get()
        x5.place(x=160,y=360)
        root6.resizable(False,False)
        root6.mainloop()

    def do_modify():
      global ad,x3,x4,x5
      ad=x3.get()
      choice=x4.get()
      new=x5.get()
      if choice=='1':
          cur.execute('update appointment set name="{}" where idno="{}"'.format(new,ad))
      elif choice=='2':
          cur.execute('update appointment set age="{}" where idno="{}"'.format(new,ad))
      elif choice=='3':
          cur.execute('update appointment set gender="{}" where idno="{}"'.format(new,ad))
      elif choice=='4':
          cur.execute('update appointment set phone="{}" where idno="{}"'.format(new,ad))
      elif choice=='5':
          cur.execute('update appointment set bg="{}" where idno="{}"'.format(new,ad))
      else:
          pass
      root6.destroy()
      tkinter.messagebox.showinfo("DONE", "YOUR DATA HAS BEEN MODIFIED")

    choice=None
    new=None
    ad=None

    def mod_sub():
      global x3,ad
      root7=Tk()
      label=Label(root7,text="MODIFICATION",font='arial 25 bold')
      label.pack()
      frame=Frame(root7,height=200,width=200)
      frame.pack()
      l1=Label(root7,text="AADHAAR NO.")
      l1.place(x=10,y=130)
      x3=tkinter.Entry(root7)
      x3.place(x=100,y=130)
      ad=x3.get()
      b1=Button(root7,text='Submit',command=modify)
      b1.place(x=100,y=160)
      root7.resizable(False,False)
      root7.mainloop()

    def search_data():
      global x3,ad
      root7=Tk()
      label=Label(root7,text="SEARCH DATA",font='arial 25 bold')
      label.pack()
      frame=Frame(root7,height=200,width=200)
      frame.pack()
      l1=Label(root7,text="AADHAAR NO.")
      l1.place(x=10,y=130)
      x3=tkinter.Entry(root7)
      x3.place(x=100,y=130)
      ad=x3.get()
      b1=Button(root7,text='Submit',command=view_data)
      b1.place(x=100,y=160)
      root7.resizable(False,False)
      root7.mainloop()

    def view_data():
      global p1
      p1=x3.get()
      cur.execute('select * from appointment where idno=(%s)',(p1,))

      dat=cur.fetchall()
      print(dat)
      a=[]
      for i in dat:
          a.append(i)
      if len(a)==0:
          tkinter.messagebox.showwarning("ERROR", "NO DATA FOUND!!")
      else:
          det=a
          tkinter.messagebox.showinfo("APPOINTMENT DETAILS",det)


    root=Tk()
    label=Label(root,text="HOSPITAL MANAGEMENT SYSTEM",font="arial 40 bold",bg='light blue')
    b1=Button(text="Registration",font="arial 20 bold",bg='yellow',command=register)
    b2=Button(text="Appointment",font="arial 20 bold",bg='yellow',command=apoint)
    b3=Button(text="List of Doctors",font="arial 20 bold",bg='yellow',command=lst_doc)
    b4=Button(text="Services available",font='arial 20 bold',bg='yellow',command=ser_avail)
    b7=Button(text="View data",font='arial 20 bold',bg='yellow',command=search_data)
    b5=Button(text="Modify existing data",font='arial 20 bold',bg='yellow',command=mod_sub)
    b6=Button(text="Exit",font='arial 20 bold',command=root.destroy,bg='violet')
    label.pack()
    b1.pack(side=LEFT,padx=10)
    b3.pack(side=LEFT,padx=10)
    b4.pack(side=LEFT,padx=10)
    b2.place(x=25,y=500)
    b7.pack(side=LEFT,padx=10)
    b5.place(x=350,y=500)
    b6.place(x=800,y=500)
    frame=Frame(root,height=600,width=50)
    frame.pack()
    root.resizable(False,False)
    root.mainloop()
```

5.RESULTS AND DISCUSSION:

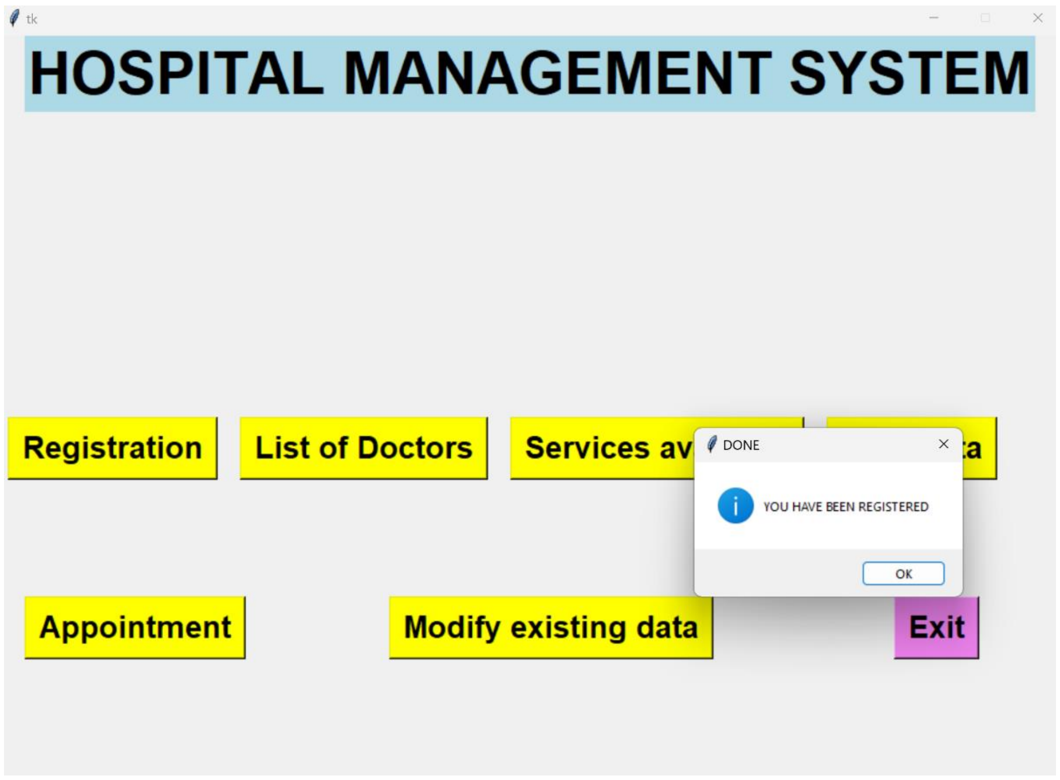


MAIN PAGE



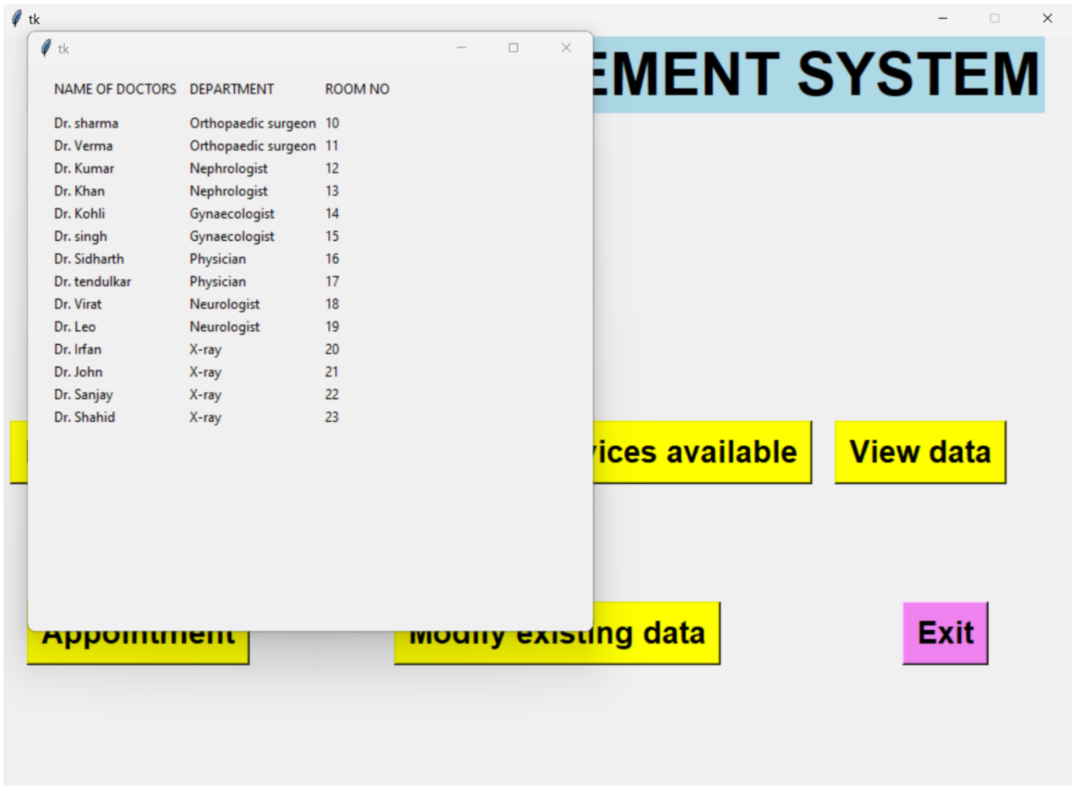
AADHAR CARD	12182004
NAME	Tom
AGE	45
GENDER M\F	M
PHONE	9679857234
BLOOD GROUP	B+

REGISTRATION PAGE



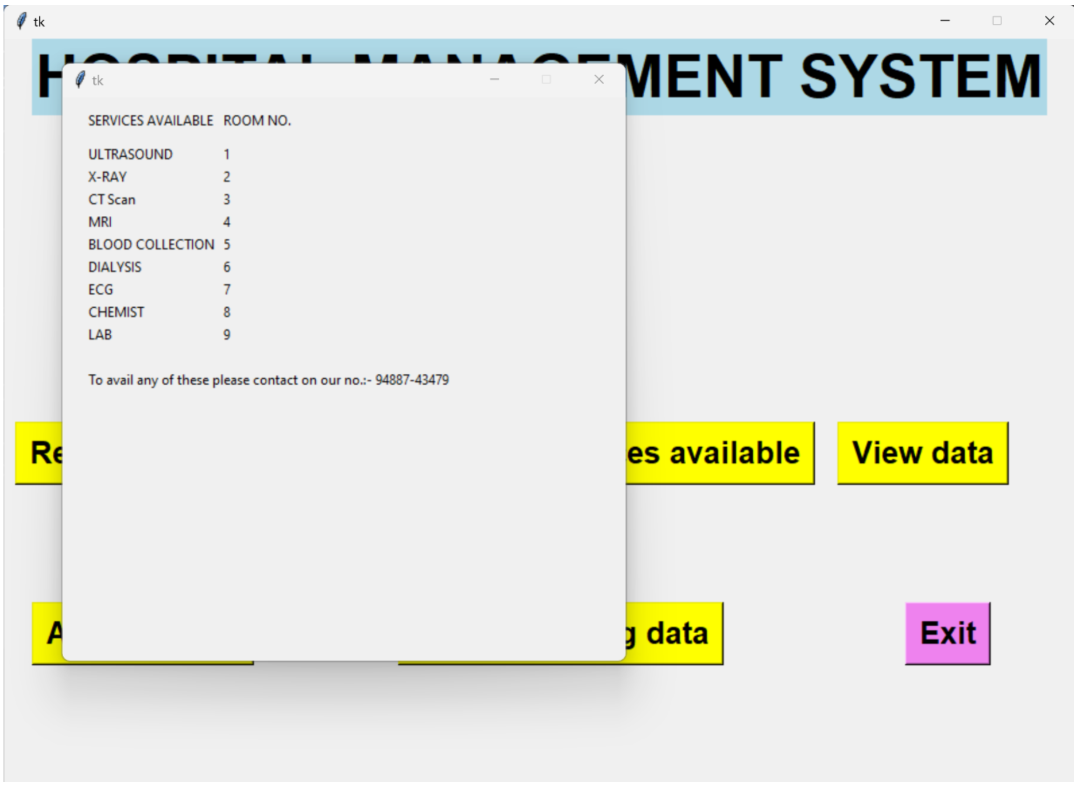
YOU HAVE BEEN REGISTERED

REGISTRATION SUCCESSFUL



NAME OF DOCTORS	DEPARTMENT	ROOM NO
Dr. sharma	Orthopaedic surgeon	10
Dr. Verma	Orthopaedic surgeon	11
Dr. Kumar	Nephrologist	12
Dr. Khan	Nephrologist	13
Dr. Kohli	Gynaecologist	14
Dr. singh	Gynaecologist	15
Dr. Sidharth	Physician	16
Dr. tendulkar	Physician	17
Dr. Virat	Neurologist	18
Dr. Leo	Neurologist	19
Dr. Irfan	X-ray	20
Dr. John	X-ray	21
Dr. Sanjay	X-ray	22
Dr. Shahid	X-ray	23

LIST OF DOCTORS

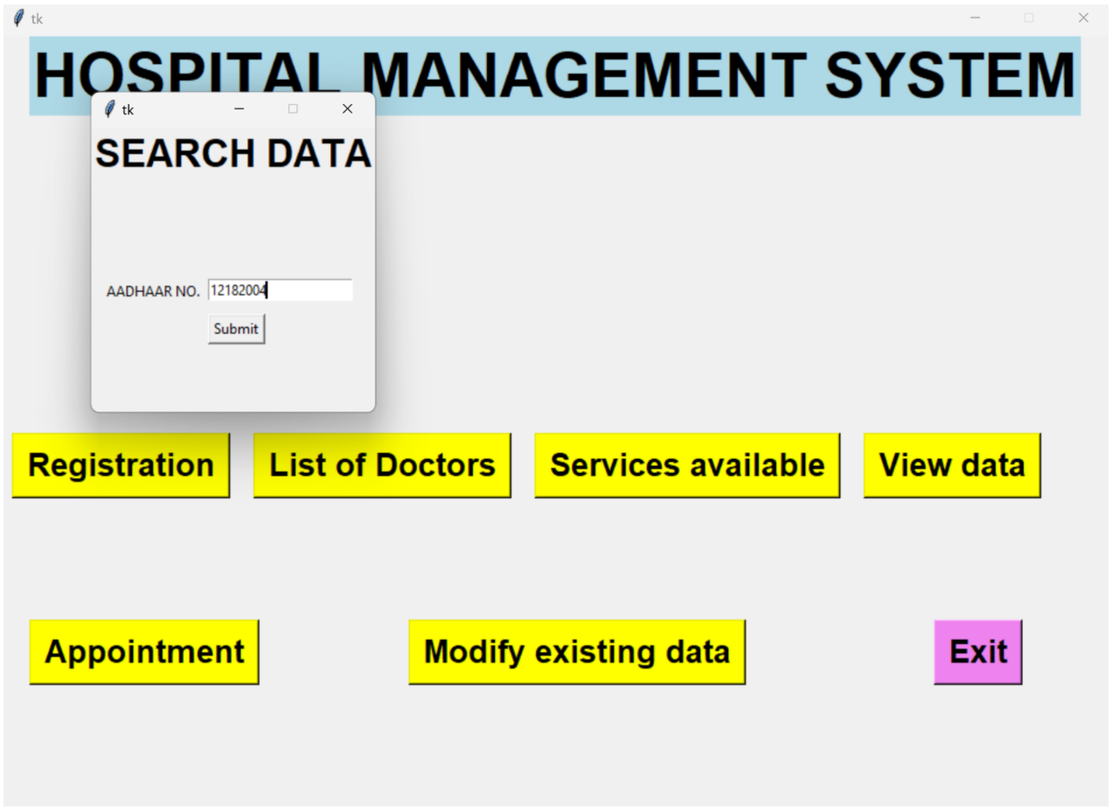


SERVICES AVAILABLE ROOM NO.

SERVICES AVAILABLE	ROOM NO.
ULTRASOUND	1
X-RAY	2
CT Scan	3
MRI	4
BLOOD COLLECTION	5
DIALYSIS	6
ECG	7
CHEMIST	8
LAB	9

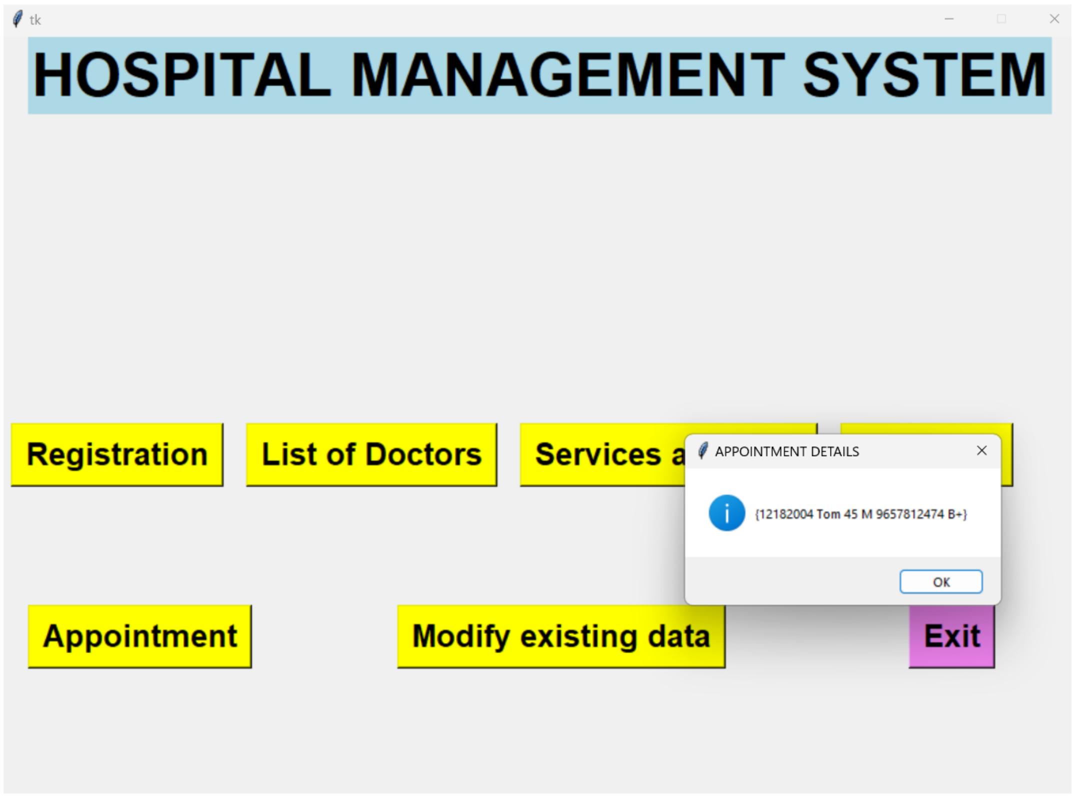
To avail any of these please contact on our no.:- 94887-43479

LIST OF SERVICES



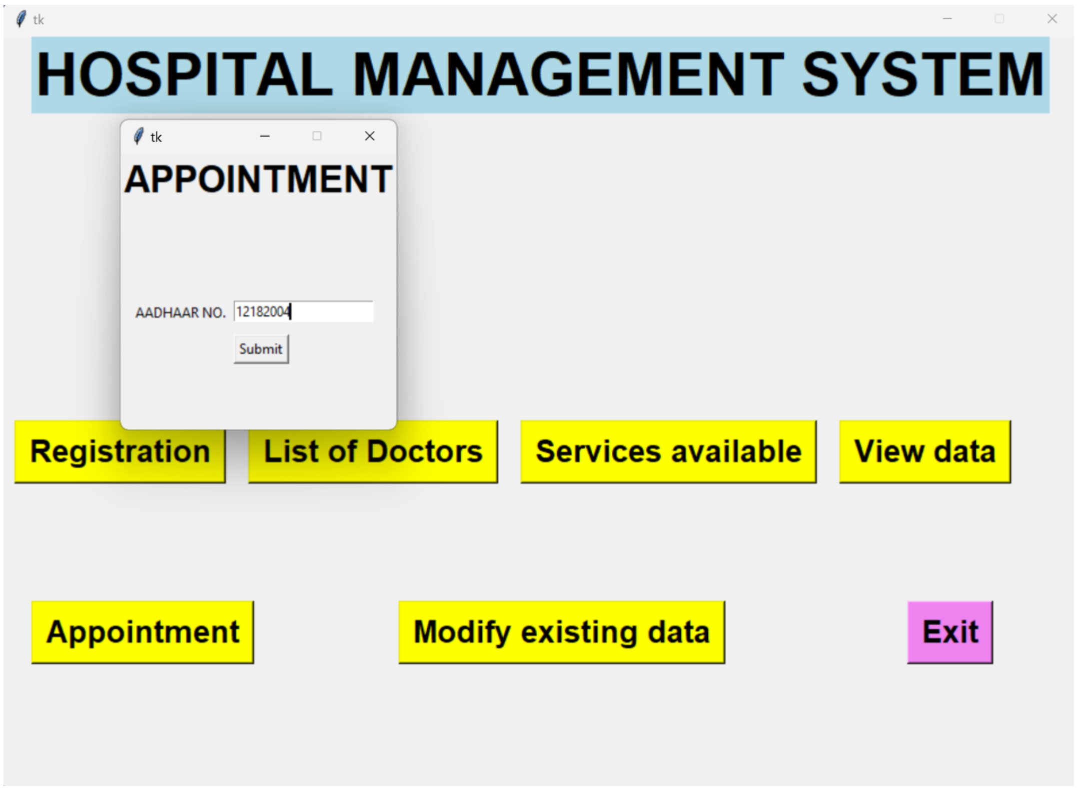
AADHAAR NO. 12182004

SEARCH DATA



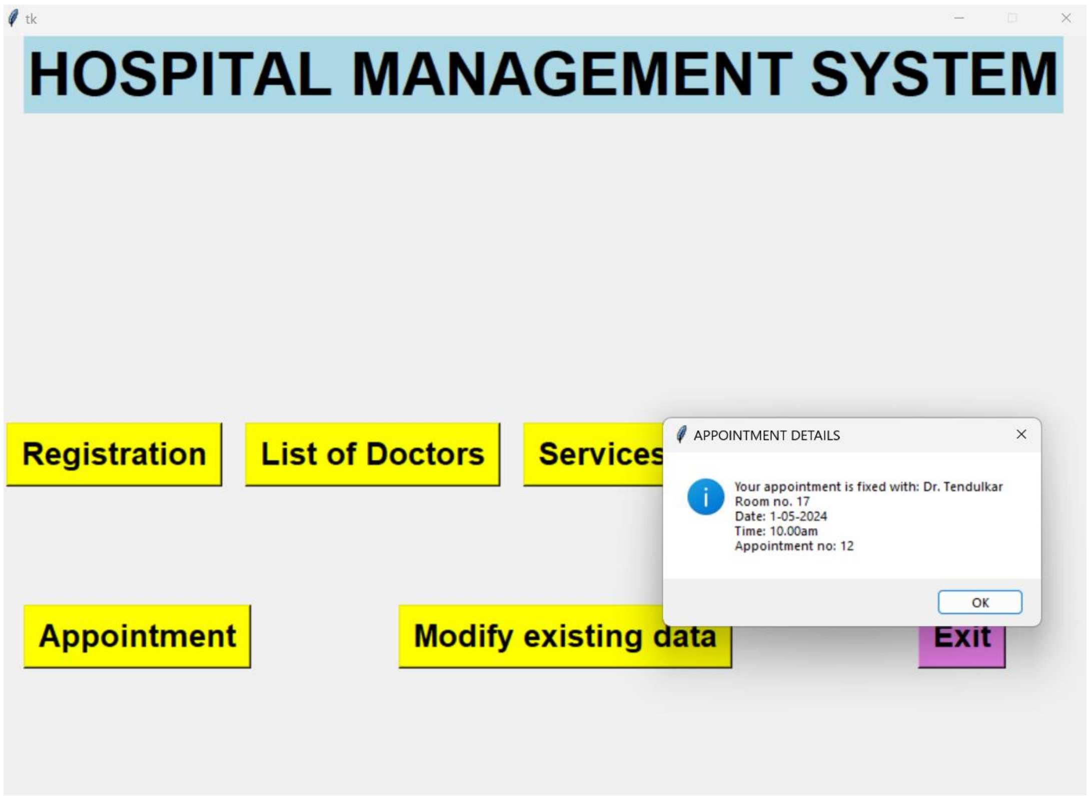
(12182004 Tom 45 M 9657812474 B+)

VIEW DETAILS



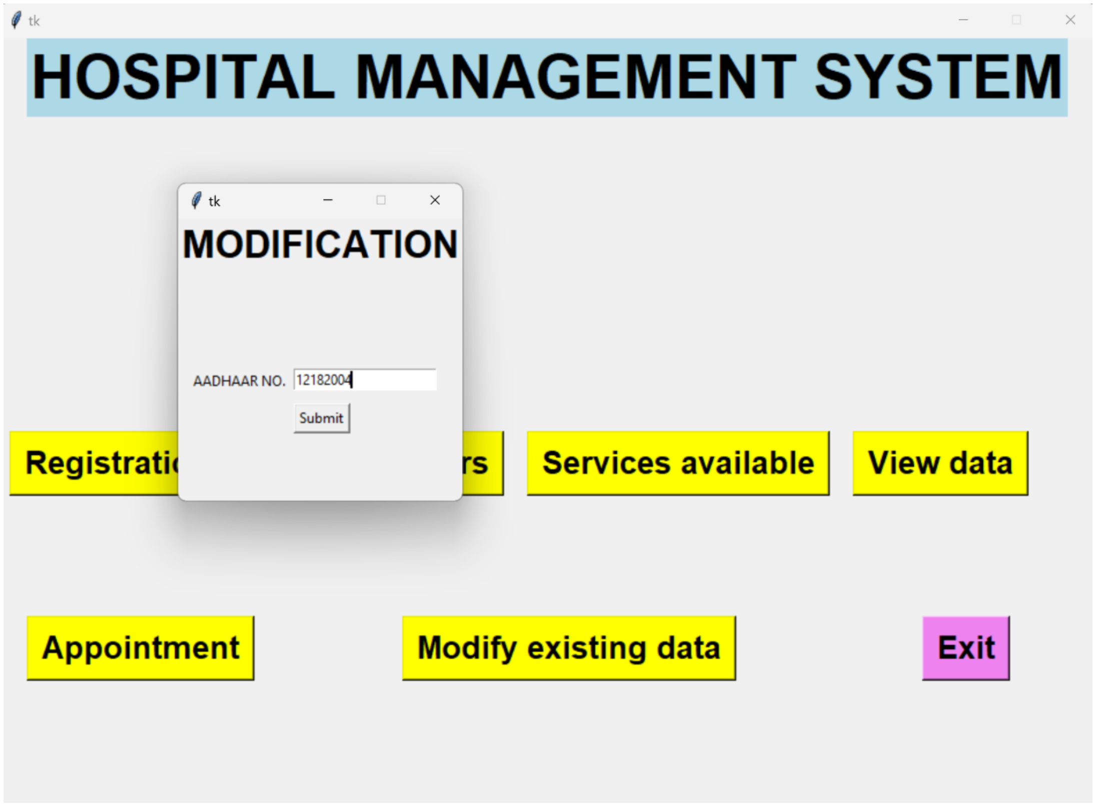
AADHAAR NO. 12182004

SEARCH APPOINTMENT USING AADHAR



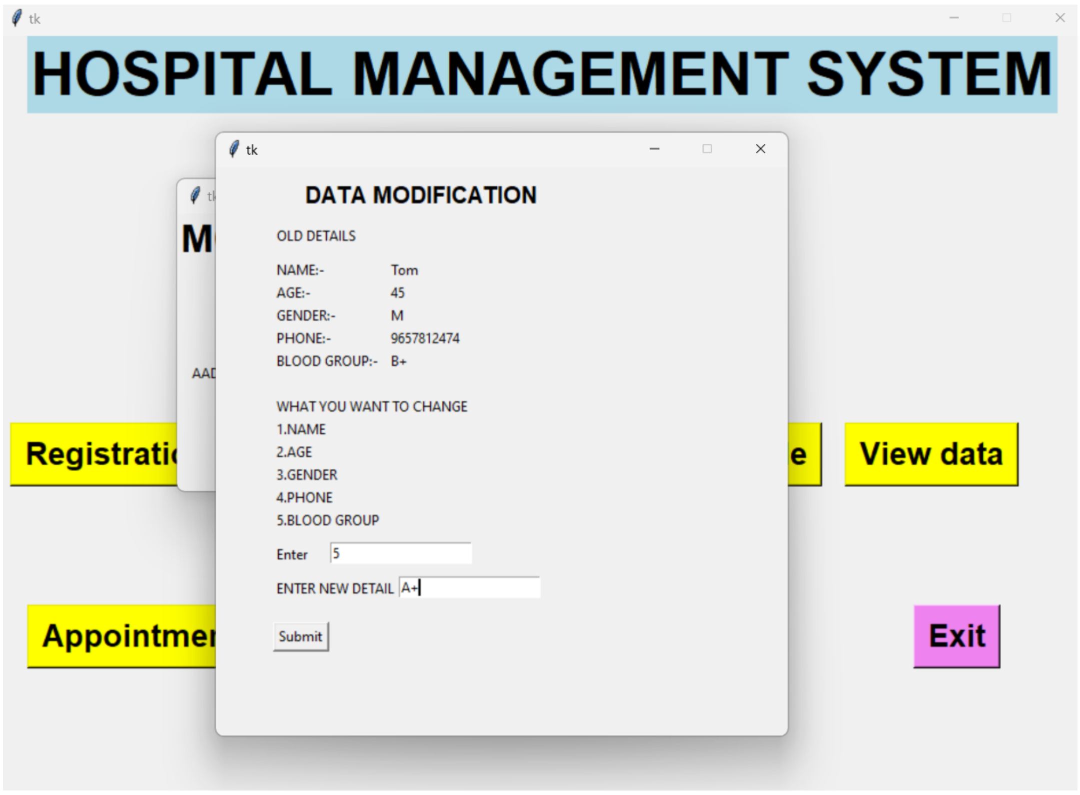
Your appointment is fixed with: Dr. Tendulkar
Room no. 17
Date: 1-05-2024
Time: 10.00am
Appointment no: 12

APPOINTMENT DETAILS



AADHAAR NO. 12182004

MODIFICATION



DATA MODIFICATION

OLD DETAILS

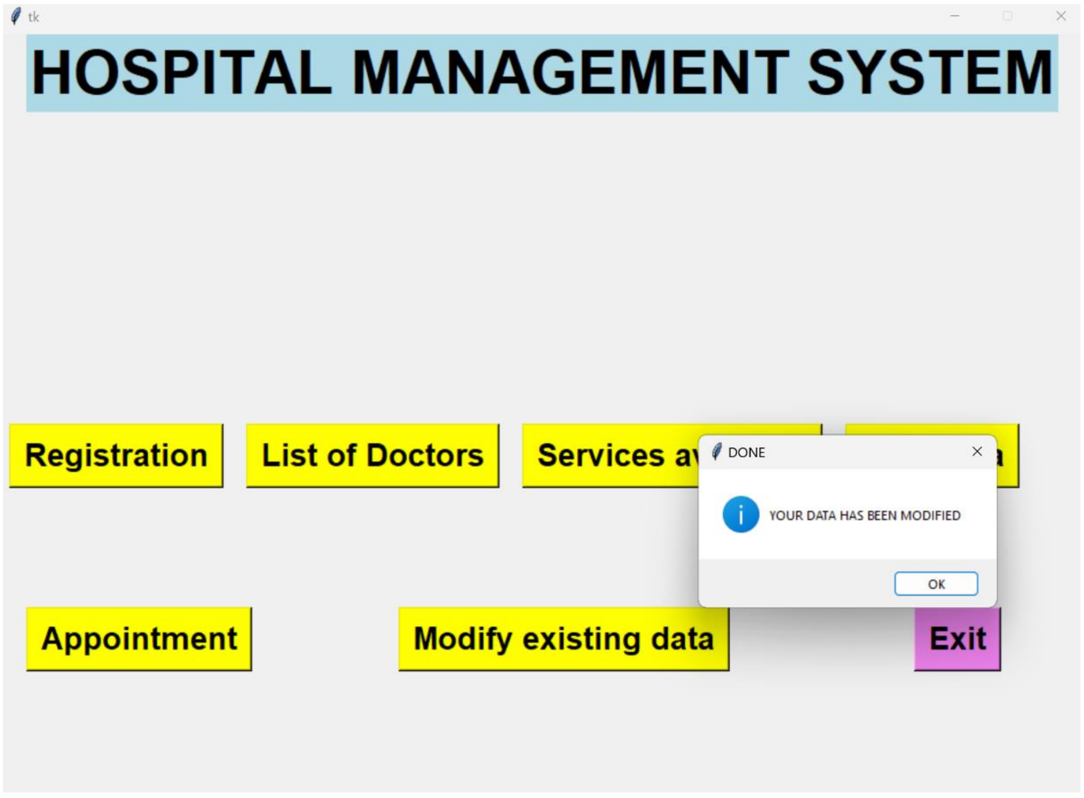
NAME:- Tom
AGE:- 45
GENDER:- M
PHONE:- 9657812474
BLOOD GROUP:- B+

WHAT YOU WANT TO CHANGE
1.NAME
2.AGE
3.GENDER
4.PHONE
5.BLOOD GROUP

Enter 5

ENTER NEW DETAIL A+

DATA MODIFICATION



YOUR DATA HAS BEEN MODIFIED

DATA MODIFIED AND EXIT PAGE

6.CONCLUSION

The project titled as Hospital Management System was deeply studied and analyzed to design the code and implement. It was done under the guidance of the experienced project guide. All the current requirements and possibilities have been taken care during the project time. It's usually a good idea to go with a Healthcare Management system that's built on a current system architecture to keep up with changing needs. The Hospital Management System (HMS) project, developed using Python, marks a significant advancement in enhancing healthcare delivery efficiency and quality. By integrating critical functionalities such as patient management, doctor and staff scheduling, medical records maintenance, billing, pharmacy management, and lab management into one cohesive system, HMS addresses the core needs of healthcare facilities. Python's versatility, along with frameworks like Flask or Django and ORMs like SQLAlchemy or Django ORM, ensures a robust, scalable, and secure system. This project not only automates and streamlines administrative tasks, improving operational efficiency and reducing errors, but also enhances patient care by providing comprehensive, easily accessible information. Robust security measures ensure data confidentiality and compliance with healthcare regulations like HIPAA. The system's scalability accommodates future growth, making it a sustainable solution for healthcare management. Overall, the successful implementation of HMS will transform healthcare facility operations, leading to better patient outcomes and increased efficiency, while also laying the groundwork for future technological innovations in the healthcare sector.

7. REFERENCE LINKS:

- https://onecompiler.com/mysql/3yaw7rkv5
- https://www.geeksforgeeks.org/hospitalmanagementsystem-using-python-and-mysql/
- https://searchcreators.org/blog01/hospital-management-system-in-python-and-mysql-dbm/