MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Database Systems Lab (CSE 2262)

HOSPITAL MANAGEMENT SYSTEM

SUBMITTED
BY

Drishaan Prasad – 210905282 – 47 – A
Devesh Shukla – 210905238 – 43 – A

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.

April 2023

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
00/00/2023

CERTIFICATE

This is to certify that the project titled **Hospital Management System** is a record of the bonafide work done by **Drishaan Prasad (210905282) and Devesh Shukla (210905238)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. **Dr. Anup Bhat, Assistant Professor, CSE Dept.**

2. **Prof. Musica Supriya, Assistant Professor (Senior Scale), CSE Dept.**

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

This kind of web-based system or software application is designed to manage the functioning of a hospital or any other medical setup effectively and smoothly. A systematic and standardized record of patients, doctors, and rooms is created with the help of this application in such a way that an administrator can have control over it. A unique ID is provided to all patients and doctors related to the database based on the ongoing treatments. All details like hospital admission, patients' discharge summary, duties of nurses and ward boys, medical stores, etc., will be maintained by separate modules.

CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES

Problem Statement:
Developing a hospital management system in order to effectively manage most aspects of hospitals such as booking appointments, managing patient records and keeping medical history. Organizations such as hospitals have to deal with a lot of patients regularly. Hence it is very important for a hospital to have a DBMS with a frontend that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

Functional Requirements:
1. Separate interfaces for patients and doctors. Patients and doctors should have separate logins.
2. Allow patients to book appointments and view/update/cancel them later if necessary.
3. Allow doctors to cancel appointments.
4. The system should avoid clash of appointments and allow appointments only when a doctor is not already busy or does not have a break.
5. Doctors should be able access patient history and profile and add to patient history.
6. Doctors should be able to give prescriptions and diagnosis.
7. Patients should be able to see complete diagnosis, prescriptions, and medical history.
8. Cancelled appointments should create free slots for other patients.

Objectives:
-Design a system for better patient care.
-Reduce hospital operating costs.
-Boost the effectiveness and caliber of medical care.
-Better co-ordination among the different departments.
-Provide top management a single point of control.

CHAPTER 3: METHODOLOGY

MySQL:
MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter My, and "SQL", the acronym for Structured Query Language. A relational database organizes data into one or more data tables in which data may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify, and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, and facilitates testing database integrity and creation of backups.

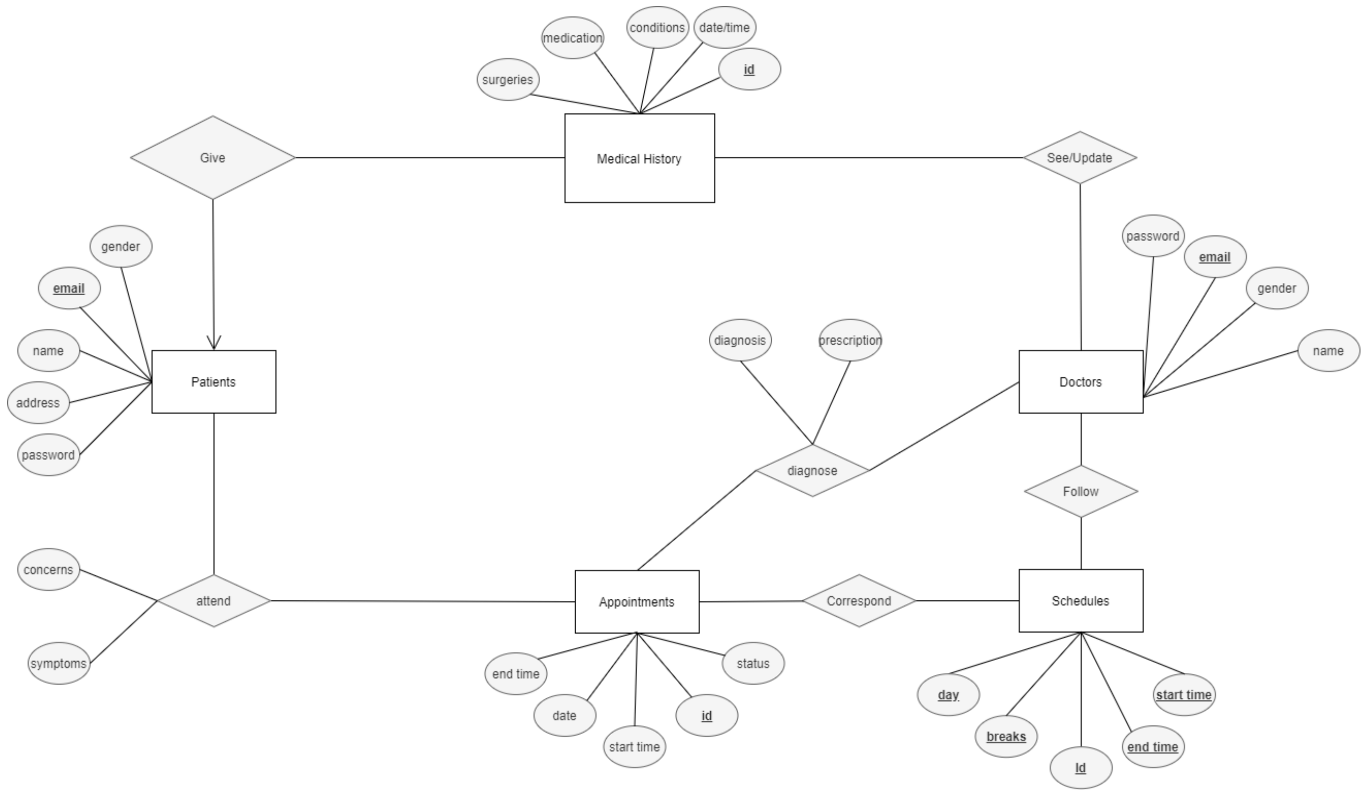
Python:
Python is a popular and powerful programming language known for its simplicity, readability, and versatility. With a focus on code readability and an extensive library ecosystem, Python is widely used for web development, data analysis, scientific computing, artificial intelligence, and automation. Its intuitive syntax, dynamic typing, and active community make it an ideal choice for both beginners and experienced developers.

HTML:
HTML (Hypertext Markup Language) is the standard markup language used for creating web pages and applications. It provides the structure and content of a webpage, defining the elements and their relationships. HTML uses tags to markup different parts of the document, such as headings, paragraphs, images, links, and more. These tags define the structure and semantic meaning of the content, allowing browsers to interpret and display the webpage correctly. HTML is the backbone of the World Wide Web, enabling the creation of interactive and accessible websites that can be viewed on various devices and platforms.

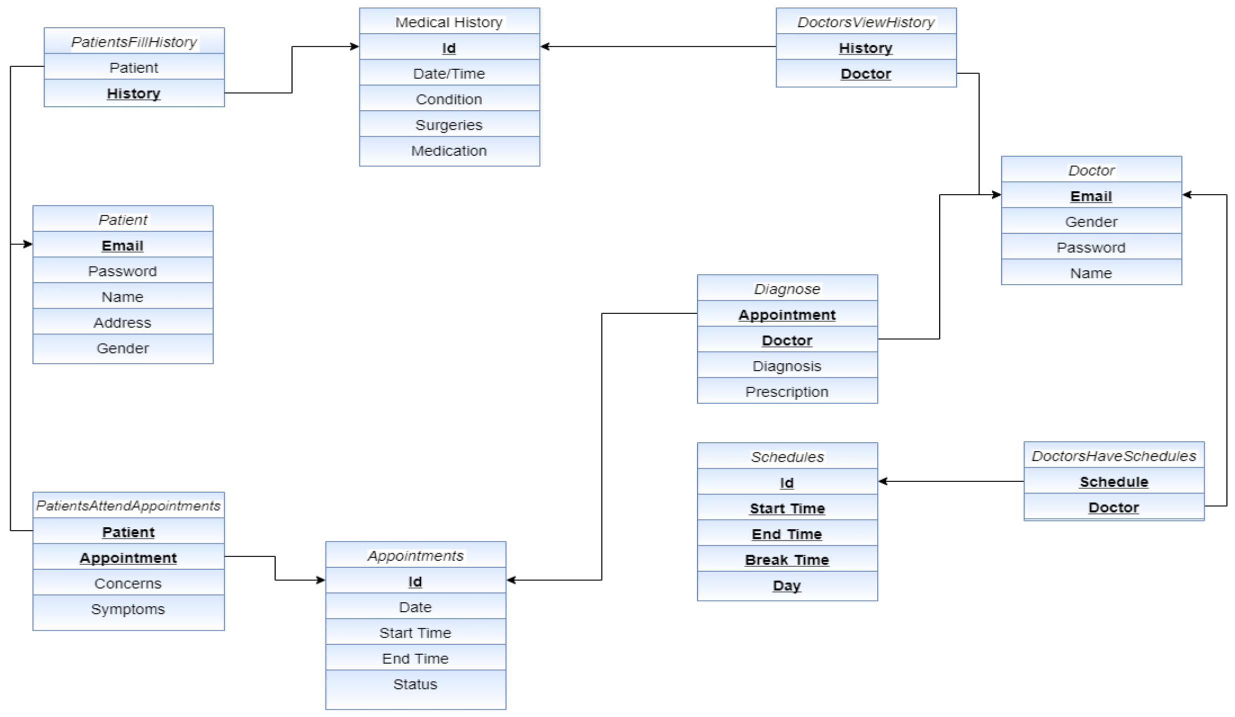
CSS:
CSS (Cascading Style Sheets) is a styling language used to describe the appearance and formatting of HTML documents. It allows web developers to control the layout, colors, fonts, and other visual aspects of a website. CSS works by selecting HTML elements and applying styles to them using selectors, properties, and values. It enables the separation of content and presentation, making it easier to maintain and update the design of a website. CSS is essential for creating visually appealing and consistent web pages across different devices and screen sizes.

Flask:
Flask is a lightweight and flexible web framework for Python. It provides a simple and elegant way to build web applications by offering a minimalistic set of tools and libraries. Flask follows the microframework approach, focusing on simplicity and extensibility. It allows developers to quickly create web applications with routing, request handling, and template rendering capabilities. Flask is known for its modular design, which encourages the use of third-party libraries to add additional functionality as needed. It is widely used for developing small to medium-sized web applications, RESTful APIs, and prototypes, making it a popular choice among Python developers.

ER Diagram:



Normalized Relational Schemas:



Relational Schemas:

1. Patient

Email	Password	Name	Address

2. Doctor

Email	Gender	Password	Id	Name

3. Medical History

Surgery	Medication	Condition	Date/Time	**Email**

4. Appointment

Id	Date	Start	End	Status

5. Schedules

Id	Start	End	Holidays	Breaks

6. Diagnose

Appt Id	Doctor	Diagnosis	Prescription

Functional Dependencies and Normalization:

1. Patient : R = (Email, Password, Name, Address, Gender)
   FDs:
   a. Email -> Password
   b. Email -> Name
   c. Email -> Address
   d. Email -> Gender
   Table is in 1NF since all attributes are atomic.
   Table is in 2NF since there is no partial dependency.
   Table is in 3NF due to absence of any transitive dependency.

2. Medical History : R = (id, Date, Conditions, Surgeries, Medication)
   FDs:
   a. id -> Password
   b. id -> Date
   c. id -> Conditions
   d. id -> Surgeries
   e. id -> Medication
   Table is in 1NF since all attributes are atomic.
   Table is in 2NF since there is no partial dependency.
   Table is in 3NF due to absence of any transitive dependency.

3. Doctor : R = (email, gender, password, name)
   FDs:
   a. email -> gender
   b. email -> password
   c. email -> name
   Table is in 1NF since all attributes are atomic.
   Table is in 2NF since there is no partial dependency.
   Table is in 3NF due to absence of any transitive dependency.

4. Appointment: R = (id, date, start time, end time, status)
   FDs:
   a. id -> date
   b. id -> start time
   c. id -> end time
   d. id -> status
   Table is in 1NF since all attributes are atomic.
   Table is in 2NF since there is no partial dependency.
   Table is in 3NF due to absence of any transitive dependency.

5. PatientsAttendAppointments: R = (patient, appointment, concerns, symptoms)

FDs:

a. (patient, appointment) -> concerns

b. (patient, appointment) -> symptoms

Table is in 1NF since all attributes are atomic.

Table is in 2NF since there is no partial dependency.

Table is in 3NF due to absence of any transitive dependency.

6. Schedule: R = (id, start time, end time, break time, day)

Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.

7. PatientsFillHistory: R = (Patient, History)

FDs:

a. History -> Patient Table is in 1NF since all attributes are atomic.

Table is in 2NF since there is no partial dependency.

Table is in 3NF due to absence of any transitive dependency.

8. Diagnose: R = (appointment, doctor,diagnosis, prescription)

FDs:

a. (appointment, doctor) -> diagnosis

b. (appointment, doctor) -> prescription

Table is in 1NF since all attributes are atomic.

Table is in 2NF since there is no partial dependency.

Table is in 3NF due to absence of any transitive dependency.

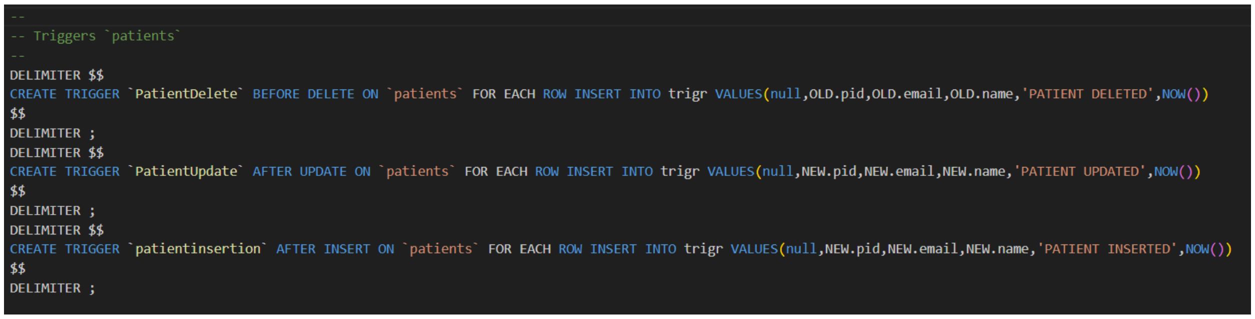
9. DoctorsHaveSchedules: R = (Schedule, Doctor)

Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.

10. DoctorViewsHistory: R = (history, doctor)

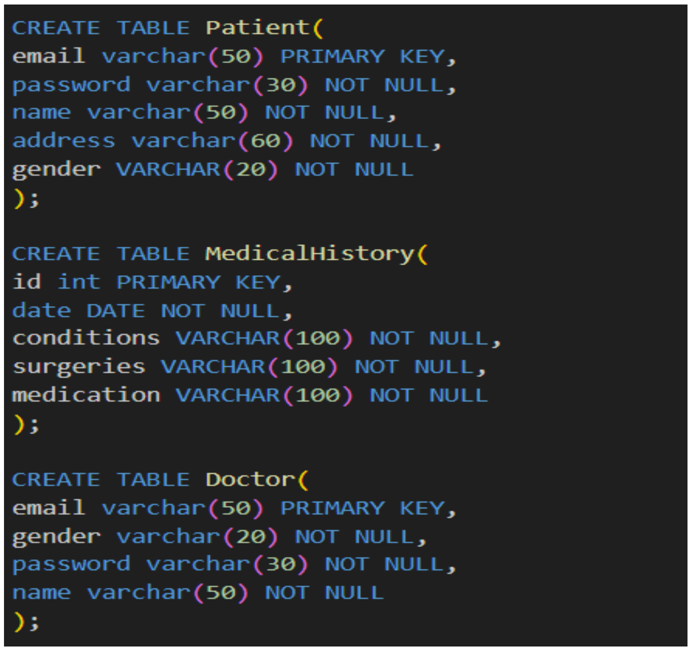
Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.

Triggers:

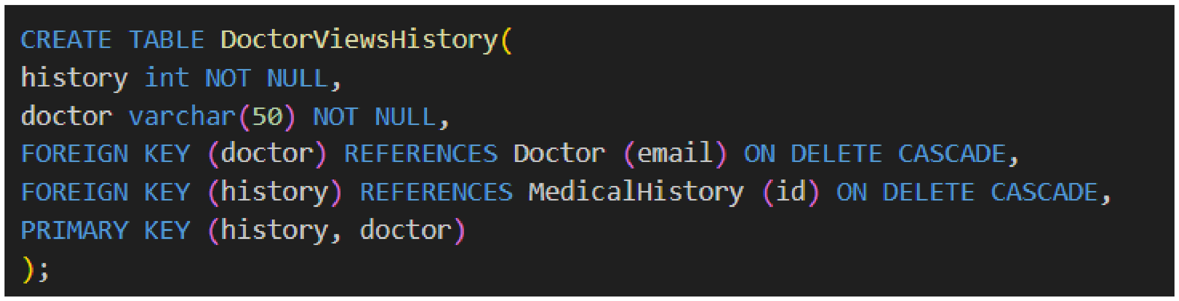


```sql
--
-- Triggers `patients`
--
DELIMITER $$
CREATE TRIGGER `PatientDelete` BEFORE DELETE ON `patients` FOR EACH ROW INSERT INTO trigr VALUES(null,OLD.pid,OLD.email,OLD.name,'PATIENT DELETED',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER `PatientUpdate` AFTER UPDATE ON `patients` FOR EACH ROW INSERT INTO trigr VALUES(null,NEW.pid,NEW.email,NEW.name,'PATIENT UPDATED',NOW())
$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER `patientinsertion` AFTER INSERT ON `patients` FOR EACH ROW INSERT INTO trigr VALUES(null,NEW.pid,NEW.email,NEW.name,'PATIENT INSERTED',NOW())
$$
DELIMITER ;
```

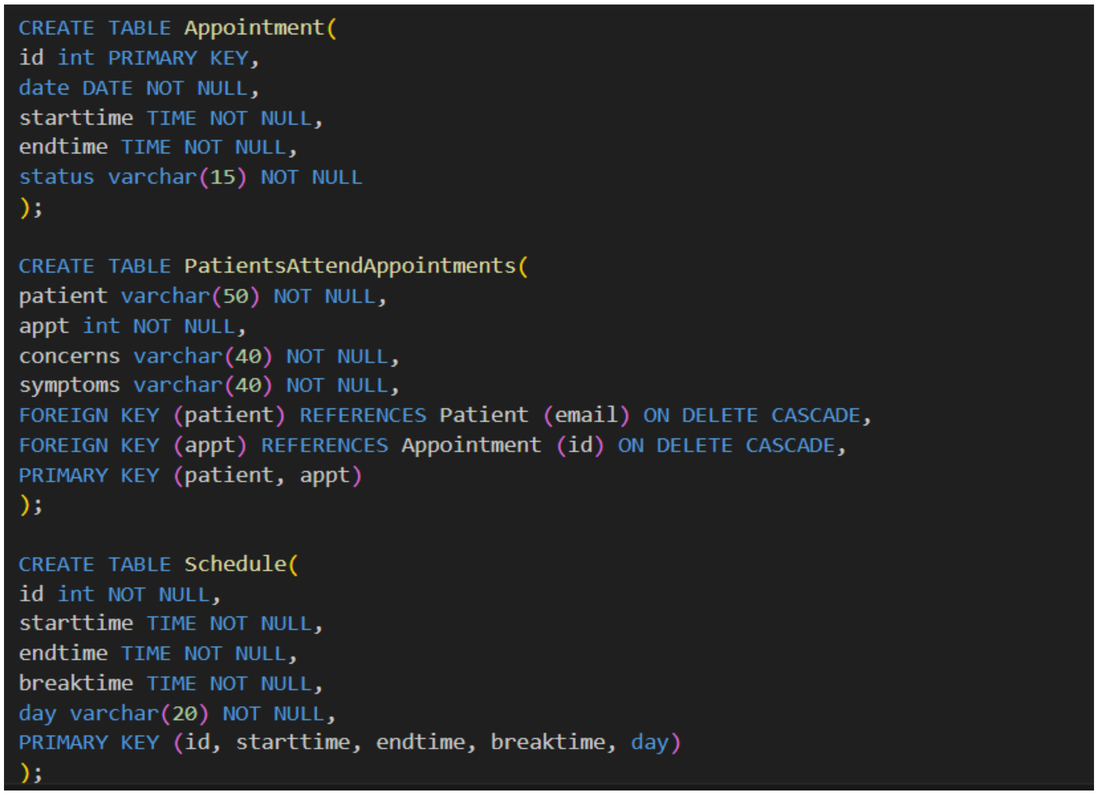
DDL Commands:



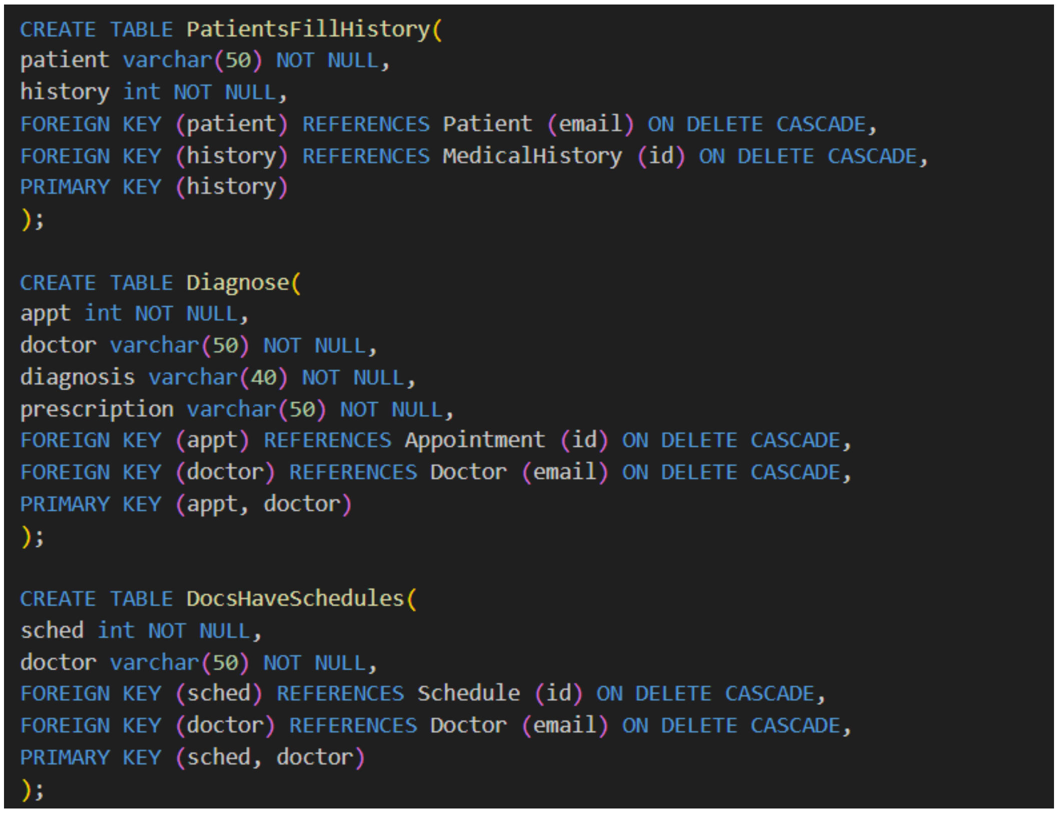
```sql
CREATE TABLE Patient(
email varchar(50) PRIMARY KEY,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL,
address varchar(60) NOT NULL,
gender VARCHAR(20) NOT NULL
);

CREATE TABLE MedicalHistory(
id int PRIMARY KEY,
date DATE NOT NULL,
conditions VARCHAR(100) NOT NULL,
surgeries VARCHAR(100) NOT NULL,
medication VARCHAR(100) NOT NULL
);

CREATE TABLE Doctor(
email varchar(50) PRIMARY KEY,
gender varchar(20) NOT NULL,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL
);
```



```sql
CREATE TABLE DoctorViewsHistory(
history int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE CASCADE,
PRIMARY KEY (history, doctor)
);
```



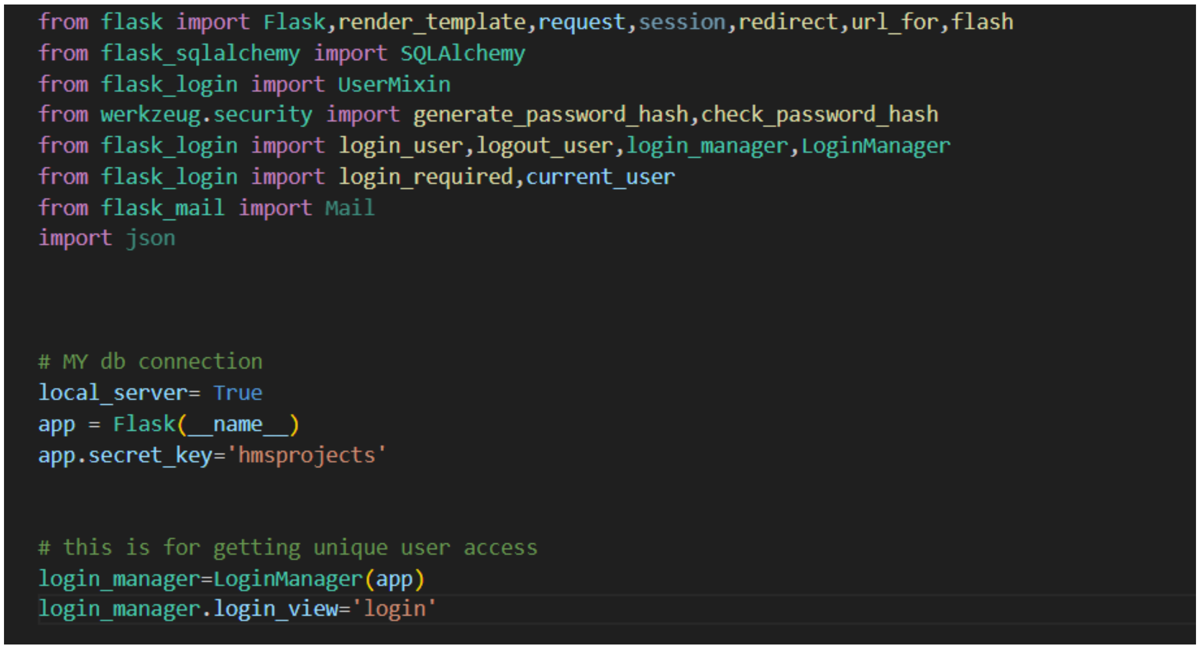
```sql
CREATE TABLE Appointment(
id int PRIMARY KEY,
date DATE NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
status varchar(15) NOT NULL
);

CREATE TABLE PatientsAttendAppointments(
patient varchar(50) NOT NULL,
appt int NOT NULL,
concerns varchar(40) NOT NULL,
symptoms varchar(40) NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
PRIMARY KEY (patient, appt)
);

CREATE TABLE Schedule(
id int NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
breaktime TIME NOT NULL,
day varchar(20) NOT NULL,
PRIMARY KEY (id, starttime, endtime, breaktime, day)
);
```



```sql
CREATE TABLE PatientsFillHistory(
patient varchar(50) NOT NULL,
history int NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE CASCADE,
PRIMARY KEY (history)
);

CREATE TABLE Diagnose(
appt int NOT NULL,
doctor varchar(50) NOT NULL,
diagnosis varchar(40) NOT NULL,
prescription varchar(50) NOT NULL,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (appt, doctor)
);

CREATE TABLE DocsHaveSchedules(
sched int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (sched) REFERENCES Schedule (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (sched, doctor)
);
```

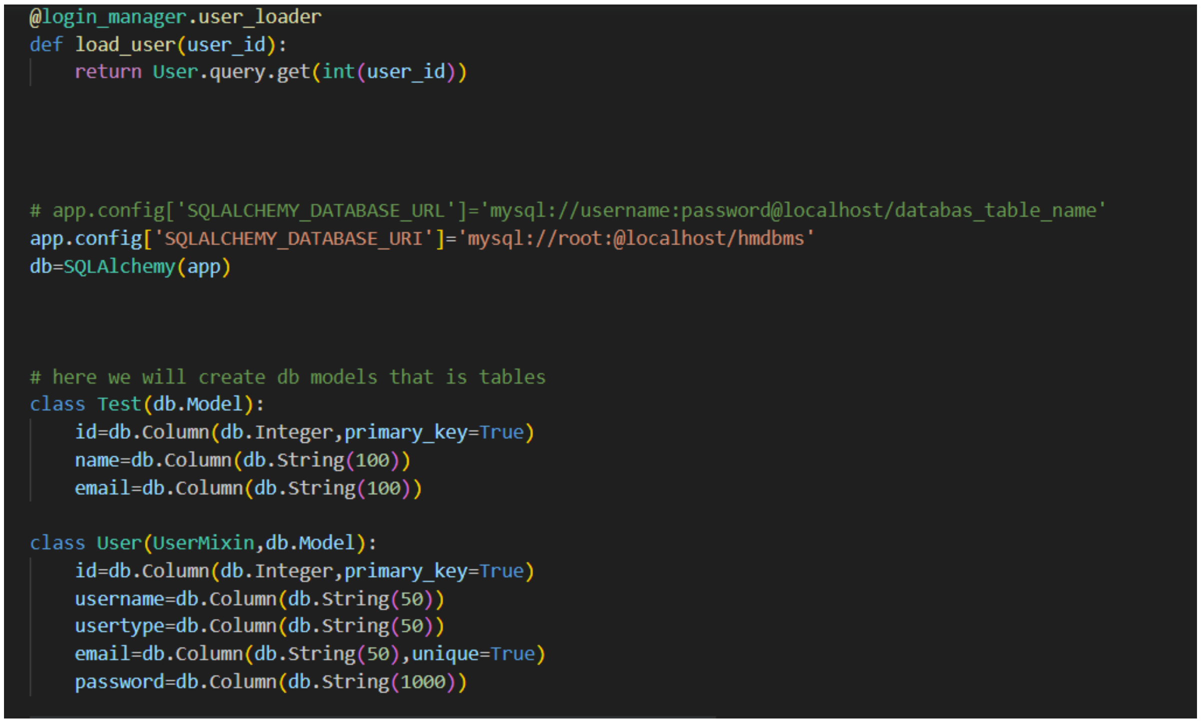
Setting Up Flask MySQL Database:

1. Database connection

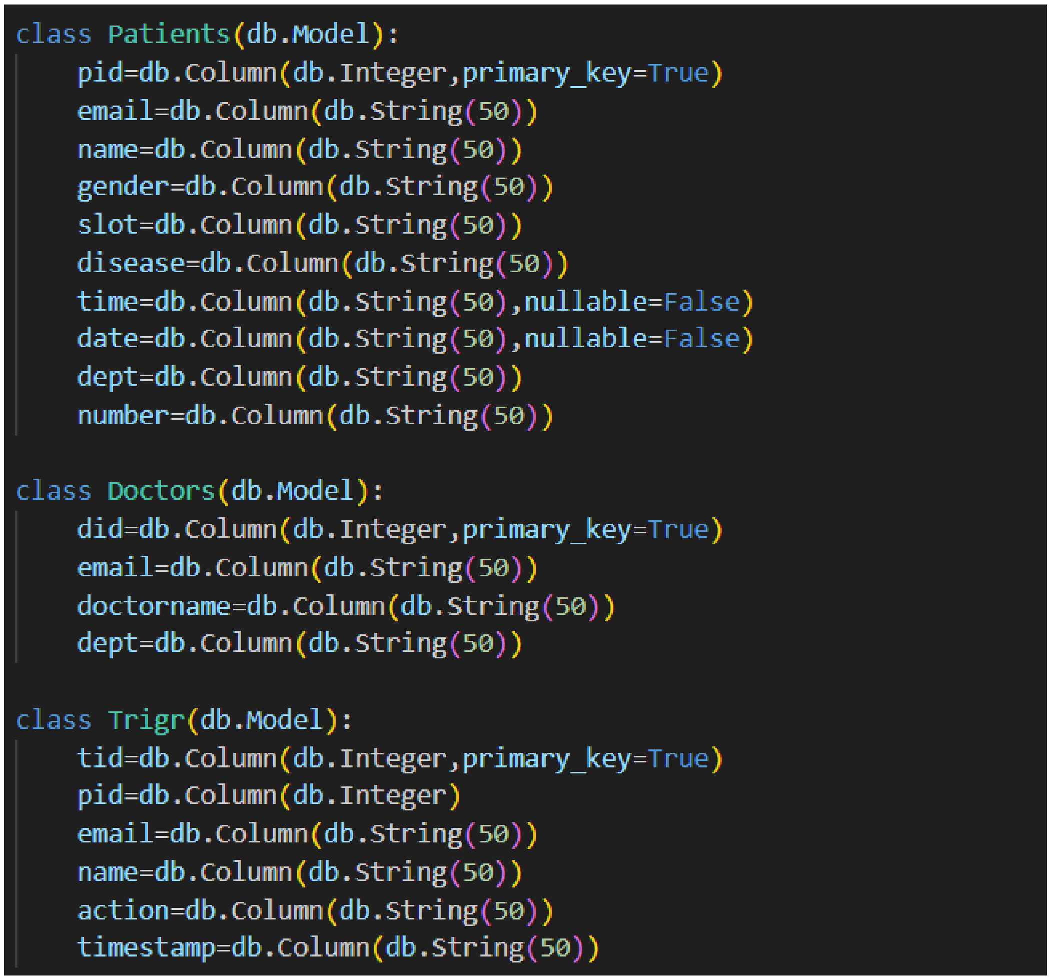


```python
from flask import Flask,render_template,request,session,redirect,url_for,flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import login_user,logout_user,login_manager,LoginManager
from flask_login import login_required,current_user
from flask_mail import Mail
import json



# MY db connection
local_server= True
app = Flask(__name__)
app.secret_key='hmsprojects'


# this is for getting unique user access
login_manager=LoginManager(app)
login_manager.login_view='login'
```

2. Creating DB Models (Tables)

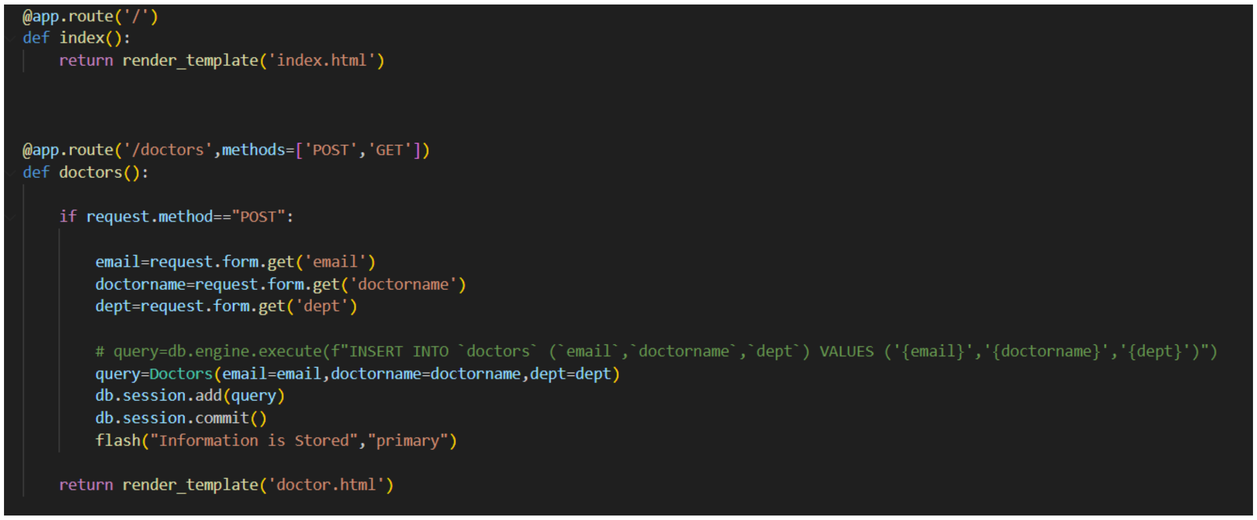


```python
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))




# app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/databas_table_name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/hmdbms'
db=SQLAlchemy(app)



# here we will create db models that is tables
class Test(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(100))
    email=db.Column(db.String(100))

class User(UserMixin,db.Model):
    id=db.Column(db.Integer,primary_key=True)
    username=db.Column(db.String(50))
    usertype=db.Column(db.String(50))
    email=db.Column(db.String(50),unique=True)
    password=db.Column(db.String(1000))
```



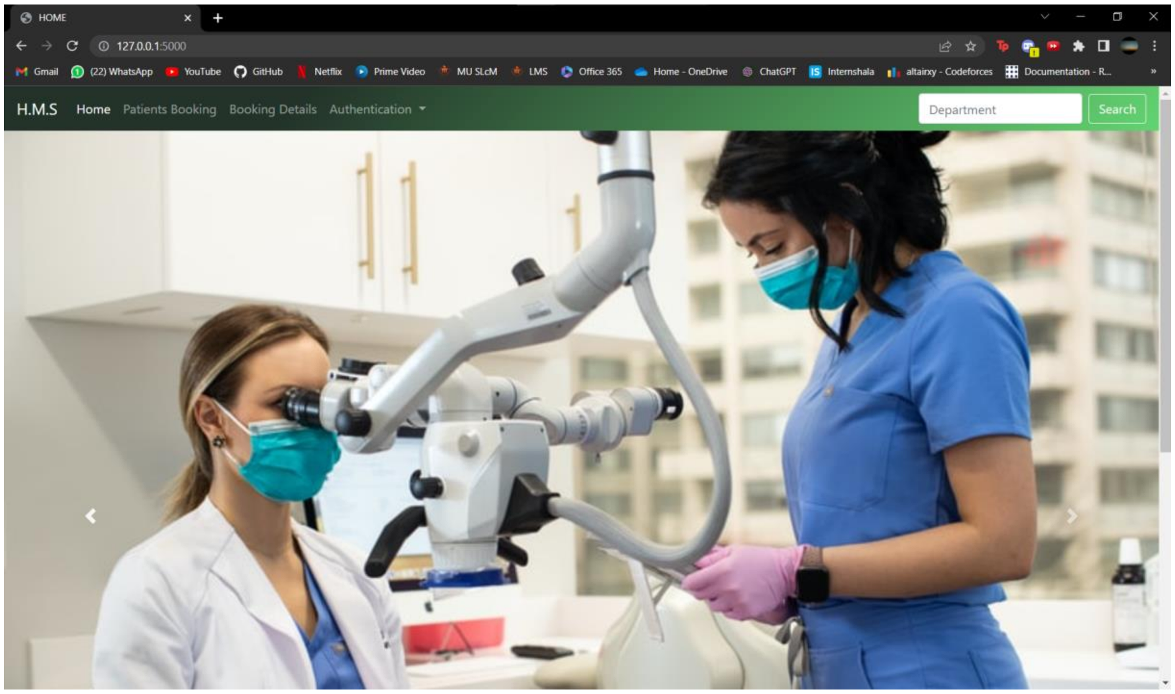
```python
class Patients(db.Model):
    pid=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
    gender=db.Column(db.String(50))
    slot=db.Column(db.String(50))
    disease=db.Column(db.String(50))
    time=db.Column(db.String(50),nullable=False)
    date=db.Column(db.String(50),nullable=False)
    dept=db.Column(db.String(50))
    number=db.Column(db.String(50))

class Doctors(db.Model):
    did=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
    doctorname=db.Column(db.String(50))
    dept=db.Column(db.String(50))

class Trigr(db.Model):
    tid=db.Column(db.Integer,primary_key=True)
    pid=db.Column(db.Integer)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
    action=db.Column(db.String(50))
    timestamp=db.Column(db.String(50))
```

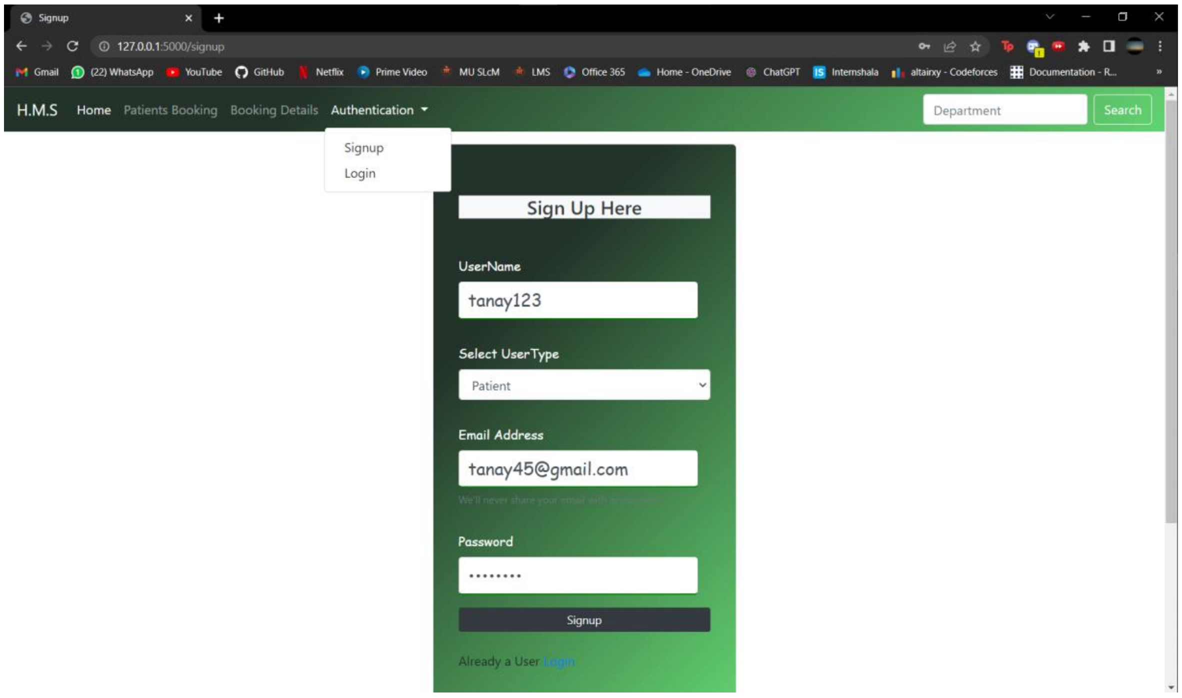
3. Passing endpoints and running the function



```python
@app.route('/')
def index():
    return render_template('index.html')



@app.route('/doctors',methods=['POST','GET'])
def doctors():

    if request.method=="POST":

        email=request.form.get('email')
        doctorname=request.form.get('doctorname')
        dept=request.form.get('dept')

        # query=db.engine.execute(f"INSERT INTO `doctors` (`email`,`doctorname`,`dept`) VALUES ('{email}','{doctorname}','{dept}')")
        query=Doctors(email=email,doctorname=doctorname,dept=dept)
        db.session.add(query)
        db.session.commit()
        flash("Information is Stored","primary")

    return render_template('doctor.html')
```

CHAPTER 4: RESULTS & SNAPSHOTS





UserName: tanay123

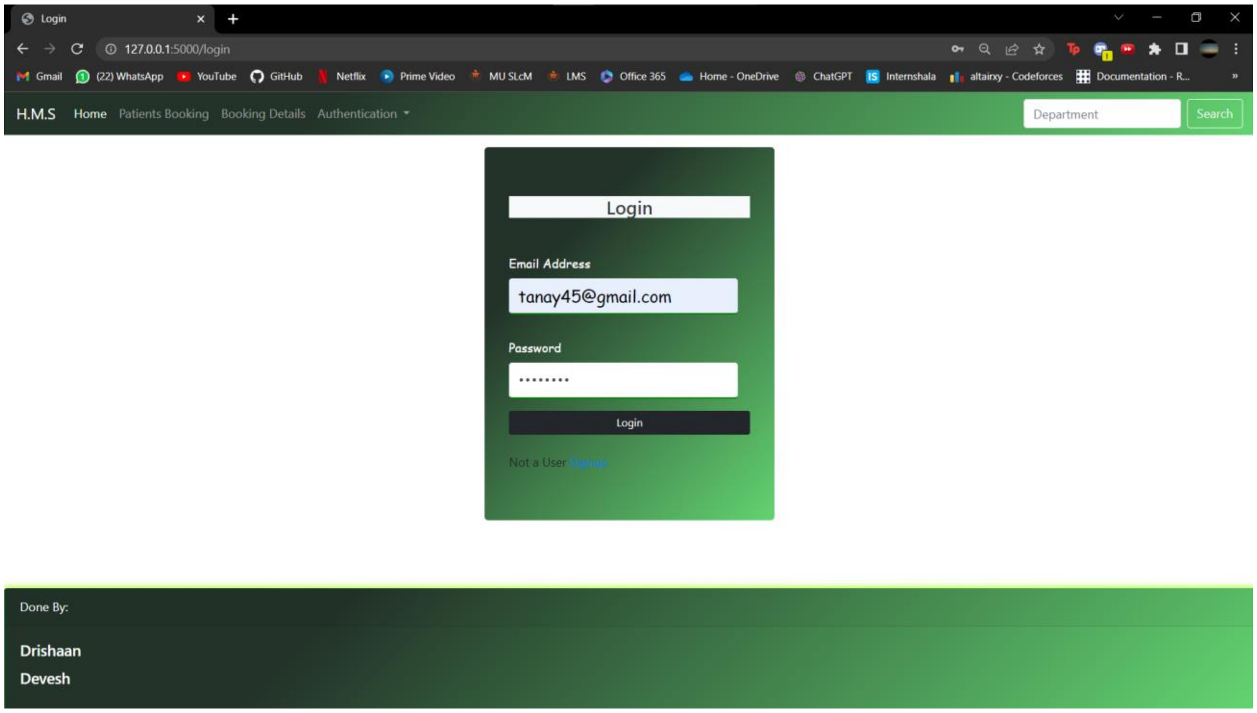
Select UserType: Patient

Email Address: tanay45@gmail.com

Password: ••••••••

Signup

Already a User Login



H.M.S Home Patients Booking Booking Details Authentication

Login

Email Address
tanay45@gmail.com

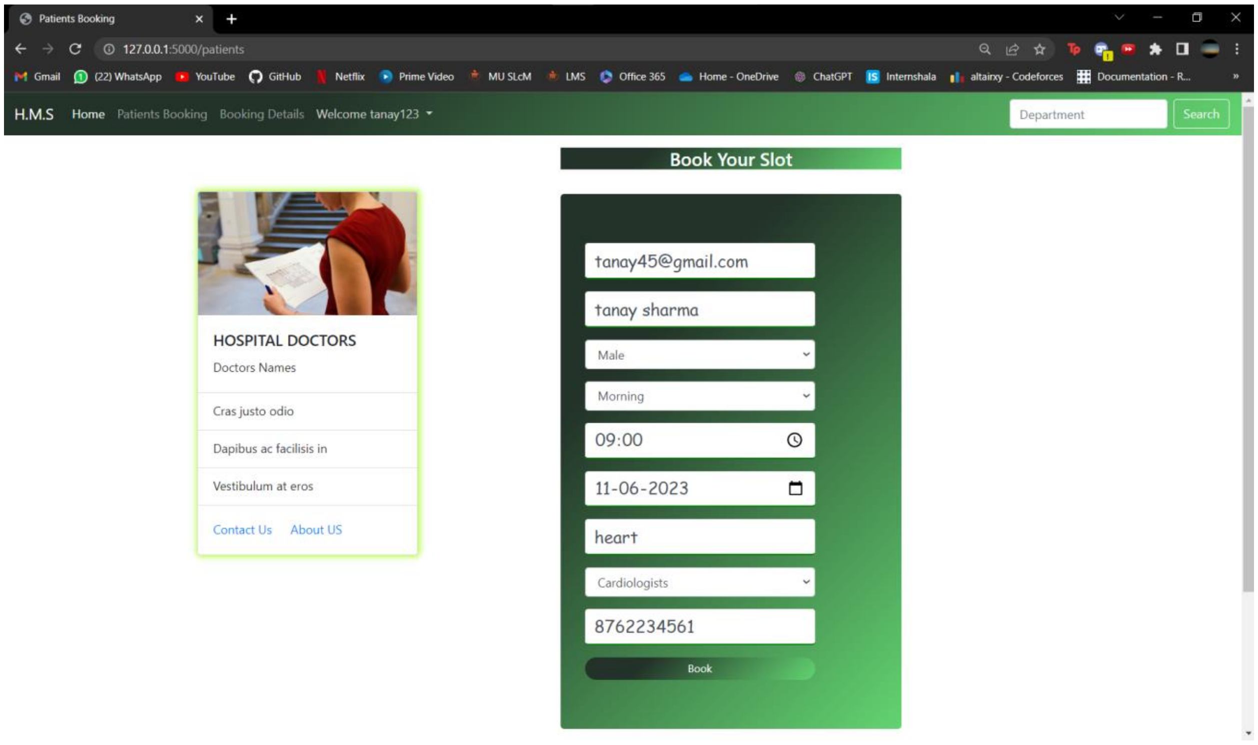
Password
••••••••

Login

Not a User Signup

Done By:

Drishaan
Devesh

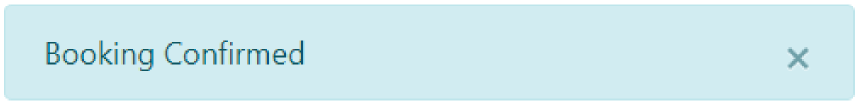


H.M.S Home Patients Booking Booking Details Welcome tanay123

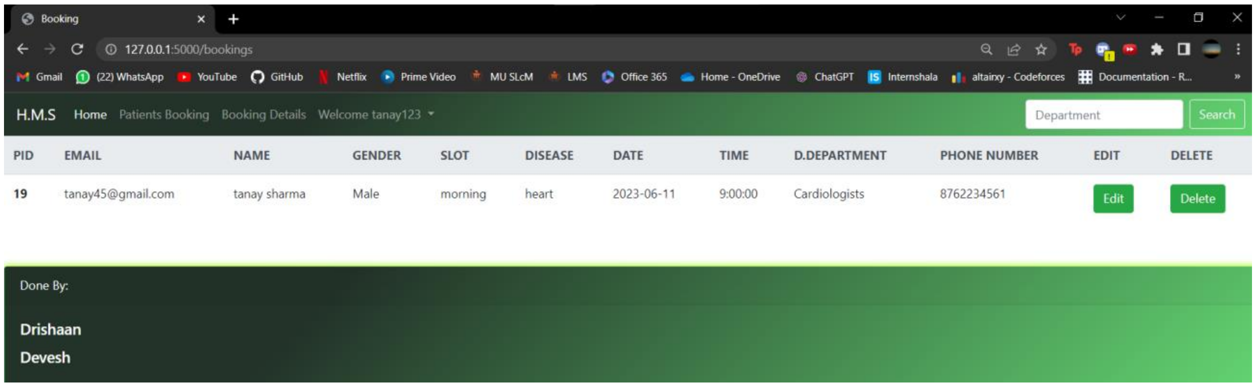
Book Your Slot

HOSPITAL DOCTORS
Doctors Names
Cras justo odio
Dapibus ac facilisis in
Vestibulum at eros
Contact Us About US

tanay45@gmail.com
tanay sharma
Male
Morning
09:00
11-06-2023
heart
Cardiologists
8762234561
Book



Booking Confirmed ×

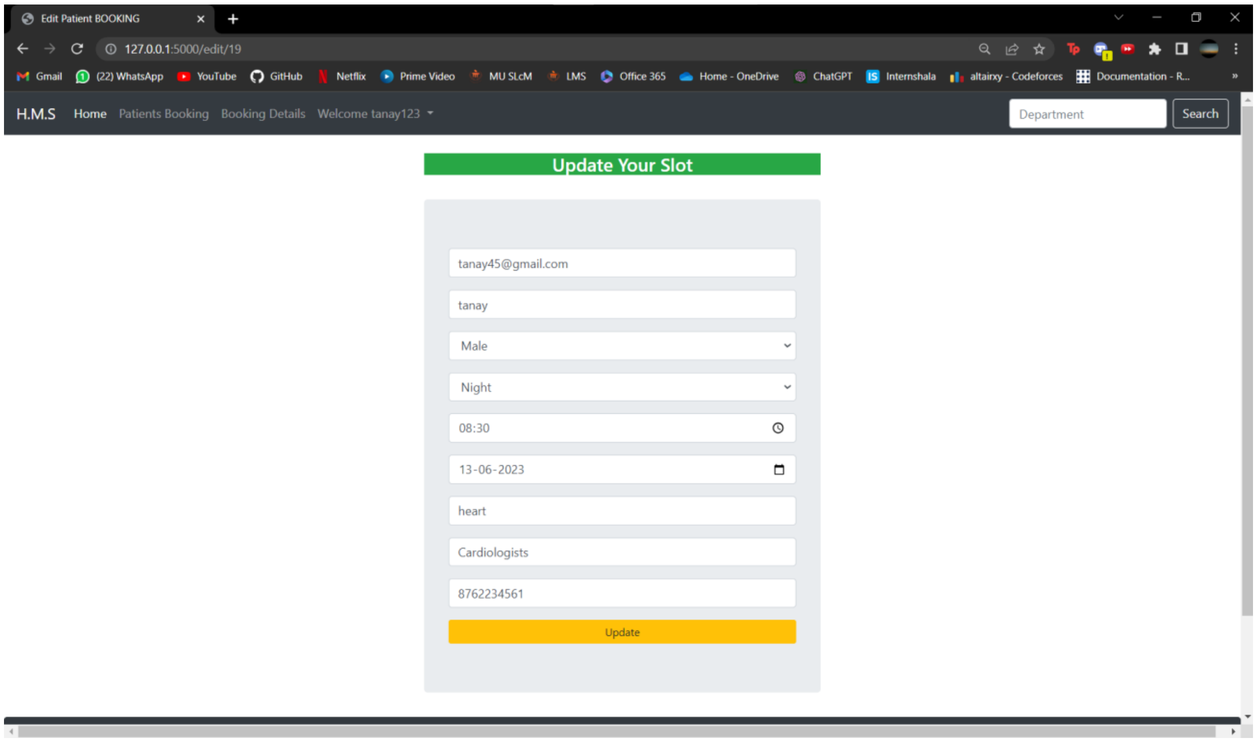


H.M.S Home Patients Booking Booking Details Welcome tanay123

PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER	EDIT	DELETE
19	tanay45@gmail.com	tanay sharma	Male	morning	heart	2023-06-11	9:00:00	Cardiologists	8762234561	Edit	Delete

Done By:

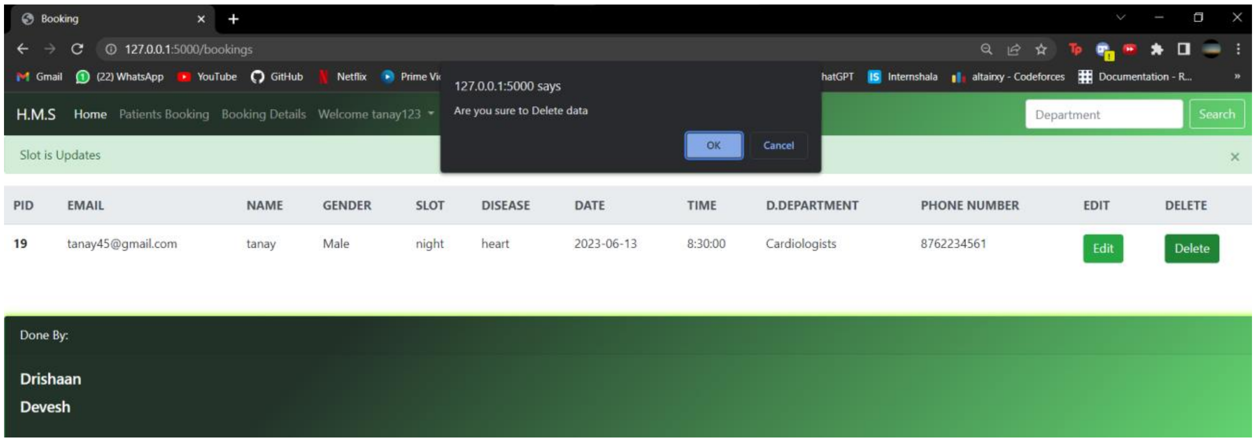
Drishaan
Devesh



H.M.S Home Patients Booking Booking Details Welcome tanay123

Update Your Slot

tanay45@gmail.com
tanay
Male
Night
08:30
13-06-2023
heart
Cardiologists
8762234561
Update



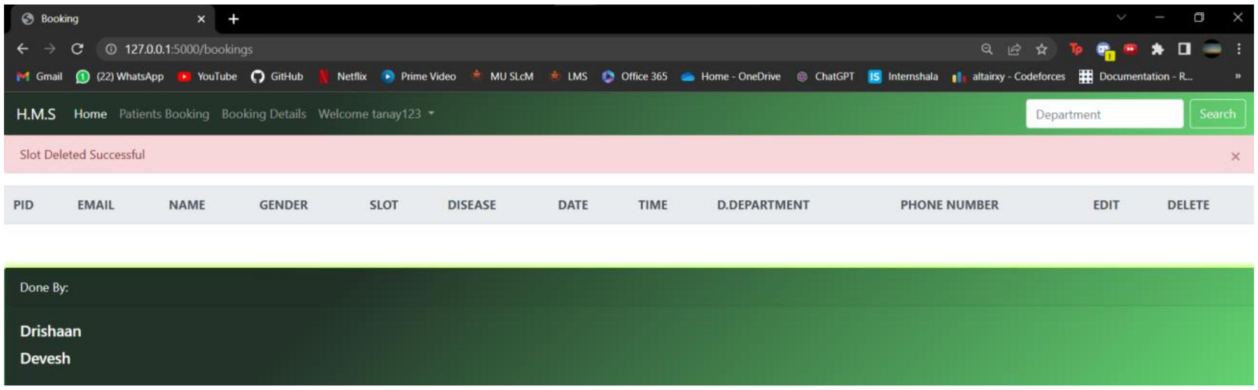
127.0.0.1:5000 says
Are you sure to Delete data
OK Cancel

Slot is Updates

PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER	EDIT	DELETE
19	tanay45@gmail.com	tanay	Male	night	heart	2023-06-13	8:30:00	Cardiologists	8762234561	Edit	Delete

Done By:

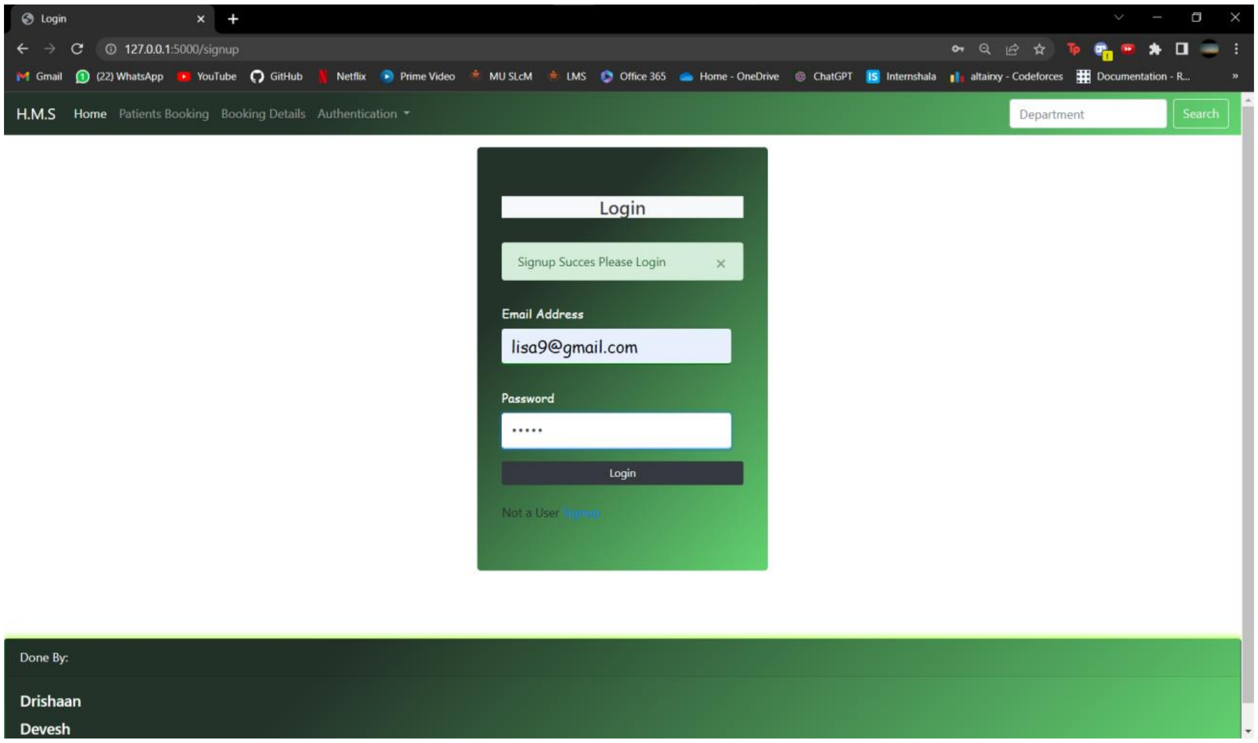
Drishaan
Devesh


Slot Deleted Successful

PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER	EDIT	DELETE

Done By:

Drishaan
Devesh


Login

Signup Succes Please Login

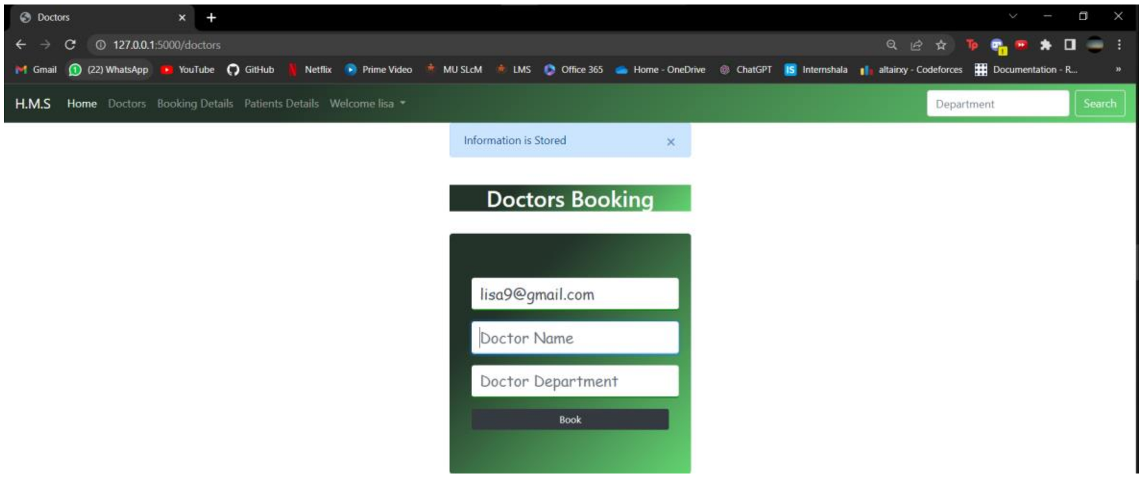
Email Address: lisa9@gmail.com

Password: •••••

Not a User Signup

Done By:

Drishaan
Devesh


Information is Stored

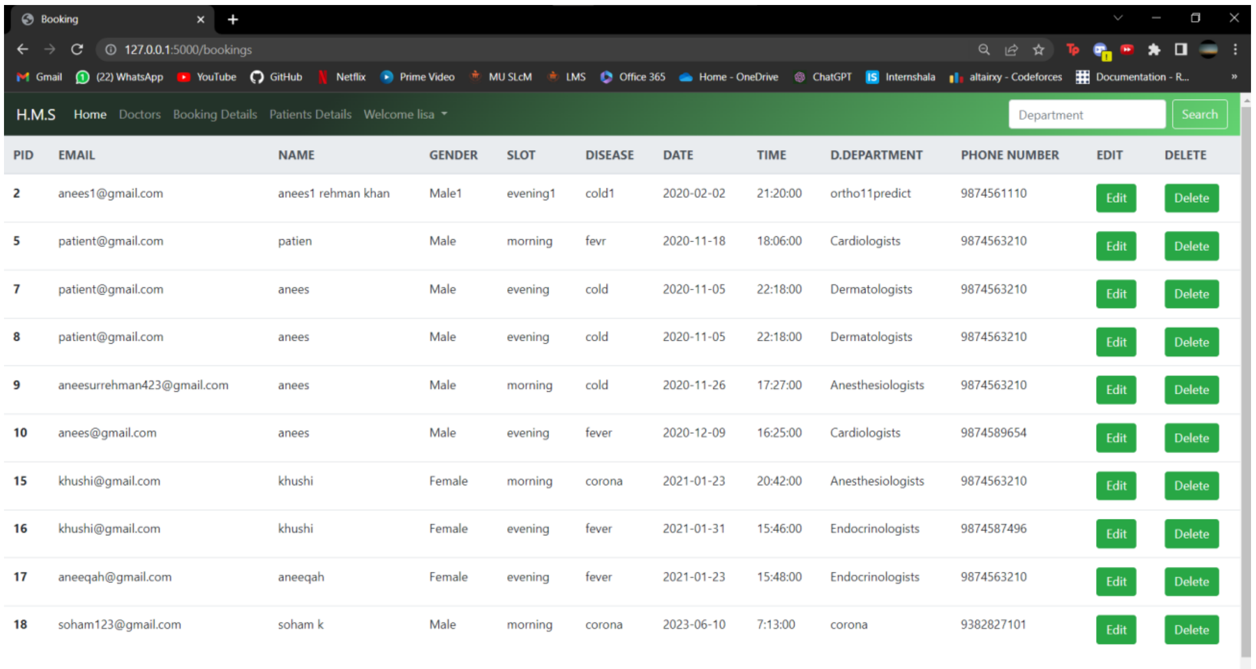
Doctors Booking

lisa9@gmail.com

Doctor Name

Doctor Department

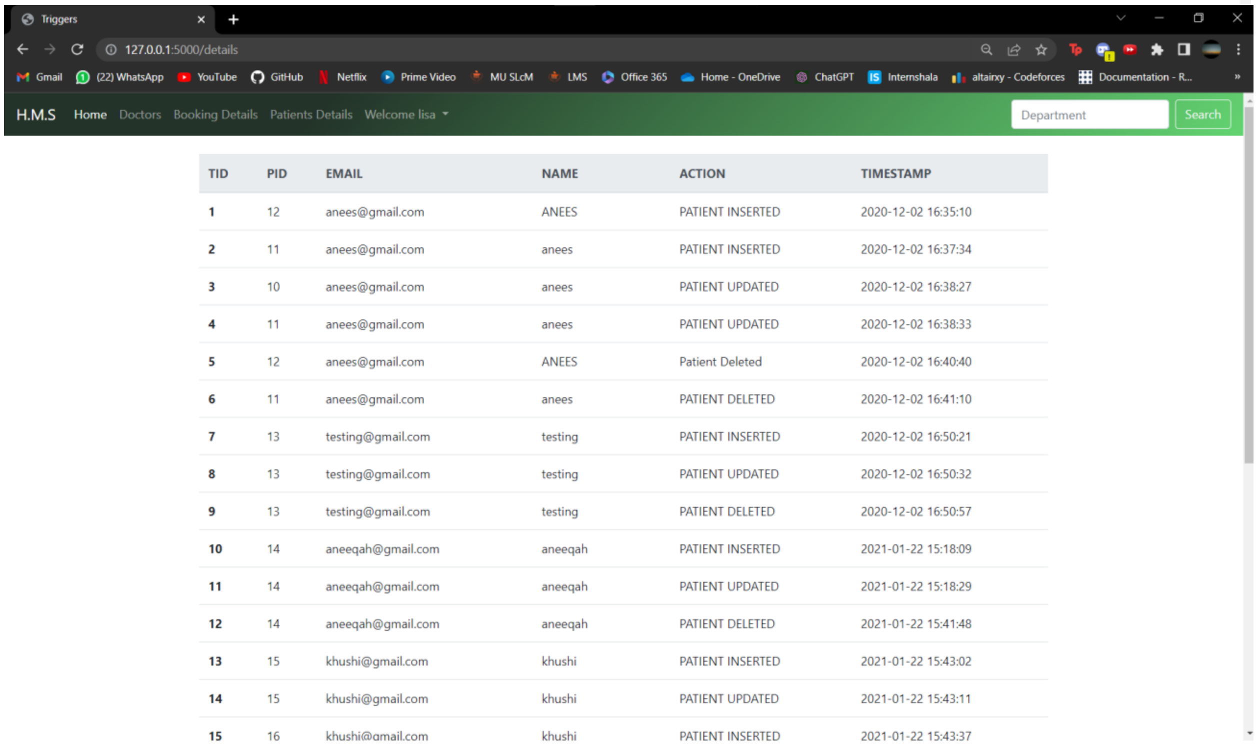
Book



127.0.0.1:5000/bookings

H.M.S Home Doctors Booking Details Patients Details Welcome lisa

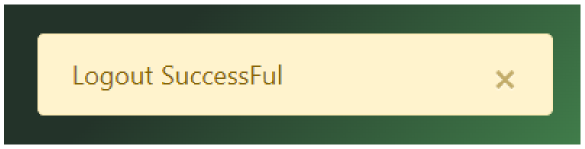
PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER	EDIT	DELETE
2	anees1@gmail.com	anees1 rehman khan	Male1	evening1	cold1	2020-02-02	21:20:00	ortho11predict	9874561110	Edit	Delete
5	patient@gmail.com	patien	Male	morning	fevr	2020-11-18	18:06:00	Cardiologists	9874563210	Edit	Delete
7	patient@gmail.com	anees	Male	evening	cold	2020-11-05	22:18:00	Dermatologists	9874563210	Edit	Delete
8	patient@gmail.com	anees	Male	evening	cold	2020-11-05	22:18:00	Dermatologists	9874563210	Edit	Delete
9	aneesurrehman423@gmail.com	anees	Male	morning	cold	2020-11-26	17:27:00	Anesthesiologists	9874563210	Edit	Delete
10	anees@gmail.com	anees	Male	evening	fever	2020-12-09	16:25:00	Cardiologists	9874589654	Edit	Delete
15	khushi@gmail.com	khushi	Female	morning	corona	2021-01-23	20:42:00	Anesthesiologists	9874563210	Edit	Delete
16	khushi@gmail.com	khushi	Female	evening	fever	2021-01-31	15:46:00	Endocrinologists	9874587496	Edit	Delete
17	aneeqah@gmail.com	aneeqah	Female	evening	fever	2021-01-23	15:48:00	Endocrinologists	9874563210	Edit	Delete
18	soham123@gmail.com	soham k	Male	morning	corona	2023-06-10	7:13:00	corona	9382827101	Edit	Delete



127.0.0.1:5000/details

H.M.S Home Doctors Booking Details Patients Details Welcome lisa

TID	PID	EMAIL	NAME	ACTION	TIMESTAMP
1	12	anees@gmail.com	ANEES	PATIENT INSERTED	2020-12-02 16:35:10
2	11	anees@gmail.com	anees	PATIENT INSERTED	2020-12-02 16:37:34
3	10	anees@gmail.com	anees	PATIENT UPDATED	2020-12-02 16:38:27
4	11	anees@gmail.com	anees	PATIENT UPDATED	2020-12-02 16:38:33
5	12	anees@gmail.com	ANEES	Patient Deleted	2020-12-02 16:40:40
6	11	anees@gmail.com	anees	PATIENT DELETED	2020-12-02 16:41:10
7	13	testing@gmail.com	testing	PATIENT INSERTED	2020-12-02 16:50:21
8	13	testing@gmail.com	testing	PATIENT UPDATED	2020-12-02 16:50:32
9	13	testing@gmail.com	testing	PATIENT DELETED	2020-12-02 16:50:57
10	14	aneeqah@gmail.com	aneeqah	PATIENT INSERTED	2021-01-22 15:18:09
11	14	aneeqah@gmail.com	aneeqah	PATIENT UPDATED	2021-01-22 15:18:29
12	14	aneeqah@gmail.com	aneeqah	PATIENT DELETED	2021-01-22 15:41:48
13	15	khushi@gmail.com	khushi	PATIENT INSERTED	2021-01-22 15:43:02
14	15	khushi@gmail.com	khushi	PATIENT UPDATED	2021-01-22 15:43:11
15	16	khushi@gmail.com	khushi	PATIENT INSERTED	2021-01-22 15:43:37



Logout SuccessFul

CHAPTER 5: CONCLUSION

In conclusion, the Hospital Database Management System mini project developed using MySQL and Python provides an efficient and reliable solution for managing various aspects of a hospital's operations. By leveraging the power of a relational database management system like MySQL, combined with the flexibility and ease of use of Python programming language, this system enables effective storage, retrieval, and manipulation of patient records, doctor information, medical history, appointments, and other vital data.

The project ensures data integrity and security by implementing appropriate validation checks, access controls, and encryption techniques. It enhances the efficiency of hospital staff by automating tasks such as patient registration, appointment scheduling, and generating reports. Additionally, it facilitates seamless communication and collaboration among different departments within the hospital, improving overall coordination and patient care.

With its user-friendly interface and intuitive functionalities, the Hospital Database Management System mini project serves as a valuable tool for hospitals and healthcare institutions to streamline their administrative tasks, enhance data management practices, and ultimately provide better healthcare services to patients.

CHAPTER 6: LIMITATIONS & FUTURE WORK

Although the Hospital Management System project has been implemented successfully, it has some limitations that can be improved in the future. Some of the limitations are:

1. Scalability: As the volume of data increases over time, the performance of the database system may start to degrade. Scaling the system to handle larger datasets and increasing user demands can pose challenges.

2. Data Integration: Hospital database management systems often need to integrate with external systems and devices such as medical equipment, billing systems, or electronic health records. Ensuring smooth data integration can be complex, requiring standardized data formats and interfaces.

3. Data Privacy and Security: Hospital databases contain sensitive patient information, making data privacy and security a critical concern. Ensuring compliance with privacy regulations, implementing robust access controls, and preventing data breaches are ongoing challenges that require constant monitoring and updating.

4. User Training and Adoption: Introducing a new database management system requires training hospital staff on its usage and ensuring smooth adoption. Resistance to change or lack of familiarity with the system may initially impact efficiency until users become comfortable with the new system.

The following are some of the areas of future work that can be considered to improve the Hospital Management System project:

1. Advanced Analytics: Implementing advanced analytics capabilities, such as data mining and predictive modeling, can help identify patterns and trends in patient data, leading to improved diagnosis, treatment, and decision-making processes.

2. Telemedicine Integration: Integrating telemedicine capabilities into the database management system would allow for remote consultations, virtual patient monitoring, and seamless sharing of information between healthcare professionals and patients.

3. Internet of Things (IoT) Integration: Connecting medical devices and sensors to the database system can enable real-time monitoring of patient vital signs, automated data collection, and enhanced patient care.

4. Mobile Access: Developing mobile applications or web interfaces that allow healthcare professionals to access the database system on the go can improve efficiency and provide timely access to critical patient information.

5. Machine Learning and AI: Utilizing machine learning and artificial intelligence algorithms can assist in clinical decision support, disease prediction, and optimizing resource allocation based on historical data and trends.

Overall, the future work for a hospital database management system involves leveraging emerging technologies, enhancing data analytics capabilities, and continuously adapting to evolving privacy and security requirements to further improve patient care and operational efficiency.

CHAPTER 7: REFERENCES

1. Database System Concepts Abraham Silberschatz, Henry F. Korth, S. Sudarshan

2. https://dev.mysql.com/doc/refman/8.0/en/

3. https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/odb_quickstart/odb_quick_start.html#t4

4. https://erdplus.com

5. https://flask.palletsprojects.com/en/2.3.x/