

# HOSPITAL MANAGEMENT SYSTEM QUINTET



## TEAM DETAILS:

Paila Syam Manoj Kumar - 20CS10041

Pandiri Adithya - 20CS10042

Ganjikunta Vijay Kumar - 20CS30018

Madugula Hemanth Kumar - 20ME10053

## Languages and Tools Used

- MySQL Database
- HTML and CSS for Front End
- Python Django Framework
  - for integrating the Front End with the database

## Schema of the Underlying Database

Doctor:

EmployeeID (PK)	Name	Department	Email	Phone	Password	Working

Patient:

PatientID(PK)	Name	Gender	Age	Phone	Diagnosis

Appointment:

AppointmentID(PK)	Doctor (FK)	Patient(FK)	Date	StartTime

Prescribes\_Medication:

Doctor (PK, FK)	Patient (PK, FK)	Appointment (PK, FK)	Medication

Prescribes\_Treatment:

Doctor (PK,FK)	Patient (PK,FK)	Appointment(FK)	Date (PK)	Treatment

Prescribes\_Test:

Doctor (PK,FK)	Patient (PK,FK)	Appointment(FK)	Date (PK)	Test

Room:

Number(PK)	Type	Available

Stay:

Patient (PK,FK)	Room (PK,FK)

Slots:

Doctor (PK,FK)	Date(PK)	StartTime(PK)	EndTime	Available

SlotsForTests:

Test(PK)	Date(PK)	StartTime(PK)	Available

SlotsForTreatments:

Treatment(PK)	Date(PK)	StartTime(PK)	Available

Scheduled\_Tests:

Patient(PK)	Test(PK)	Date(PK)	StartTime(PK)

Scheduled\_Treatments:

Patient(PK)	Treatment(PK)	Date(PK)	StartTime(PK)

FrontDeskOp:

Username (PK)	Password	Email	Name

DataEntryOp:

Username (PK)	Password	Email	Name

DbAdmin:

Username (PK)	Password

---

## Home

- Redirects to four different login pages
  - It verifies login details (username, password) of a user in corresponding tables and redirects to their corresponding pages. Note that EmployeeID of doctor is the username for the doctor to login.
  - If details are wrong it will display a popup saying wrong credentials
- 

## Functionalities of a DATABASE ADMINISTRATOR:

- Will be able to add or delete new users using the tables
  - FrontDeskOp
  - Doctor
  - DataEntryOp.
- These six functionalities will be available on screen

### Add Front-Desk Operator:

This page is used to add a new Front Desk Operator with details such as Username, Password, Email and Name. This Username is the primary key and has to be unique. An SQL query runs and adds this Front Desk Operator to the FrontDeskOp table in the backend database.

### Delete Front-Desk Operator:

This page is used to delete an existing Front Desk Operator. The page displays a list of all the usernames of the Front Desk Operators. The administrator enters the username of a Front Desk Operator that he wishes to remove. An SQL query runs that takes this username and removes the corresponding entry from the FrontDeskOp table in the backend database.

### **Add Doctor:**

This page is used to add a new Doctor with details such as Name, Department, Email, Phone and Password. A new EmployeeID is given to this Doctor which acts as his primary key. This is done automatically to avoid repetition as EmployeeID is an integer. An SQL query runs and adds this Doctor with this newly generated EmployeeID in the Doctor table in the backend database.

### **Delete Doctor:**

This page is used to delete an existing Doctor. The page displays a list of all the Doctors in the EmployeeID-Name-Password format. The administrator enters the EmployeeID of a Doctor that he wishes to remove. An SQL query runs that takes this EmployeeID and removes the corresponding entry from the Doctor table in the backend database.

### **Add Data-Entry Operator:**

This page is used to add a new Data-Entry Operator with details such as Username, Password, Email and Name. This Username is the primary key and has to be unique. An SQL query runs and adds this Data-Entry Operator to the DataEntryOp table in the backend database.

### **Delete Front-Desk Operator:**

This page is used to delete an existing Data-Entry Operator. The page displays a list of all the usernames of the Data-Entry Operators. The administrator enters the username of a Data-Entry Operator that he wishes to remove. An SQL query runs that takes

this username and removes the corresponding entry from the DataEntryOp table in the backend database.

---

## **Functionalities of a DOCTOR:**

- There will be a table displaying all the patients being treated by the doctor
- A query which takes a PatientID and displays the complete data of the patient. This functionality takes the following tables :
  - Doctor
  - Patient
  - Prescribes\_Treatment
  - Prescribes\_Medication
  - Prescribes\_Test
- The Doctor can also add Medication, Treatment or Test. This functionality takes the following tables :
  - Doctor
  - Appointment
  - Patient
  - Prescribes\_Treatment
  - Prescribes\_Medication
  - Prescribes\_Test

### **Query for a patient:**

On this page, there will be a table displaying the patients who are being treated by the doctor who logged in. This table will only contain the basic info, using the query functionality, the doctor can get the complete info of a patient.

### **Add Medication, Treatment or Test:**

The doctor can update the medication, treatment and (or) test of a particular patient by entering the patient id and appointment id.

---

## **Functionalities of a Front Desk Operator:**

There are six major functionalities for the front desk operator. The detailed explanation of these functionalities is as follows:

### **Patient Registration:**

This page is used to register the patient when he visits the hospital. In this page, the details of the patient i.e, Name, Gender, Age, Phone are taken as the input and are added to the Patient Table of the backend database and a unique Patient ID gets assigned to the registered patient. This patient ID is used for further communication.

### **Patient Discharge:**

This page is used to discharge the patient. In this page, we take the Patient ID and the Room number assigned to the patient as the input and we delete the entry from the stay table of the backend database and we update the Available column to '1' in the Room table of backend database.

### **Assign Room:**

This page is used to assign rooms if the patient wants to be admitted to the hospital. In this page, the list of available rooms is always displayed at the top. We select one of the rooms from the above list and we assign the Room Number and Room Type to the input Patient ID. In the backend database, this information is added to the stay table and the Available column of the assigned room is set to '0' in the Room table of the backend database.

### **Doctor Appointment:**

This page is used to schedule an appointment between doctor and patient. First, we enter the Doctor ID with



whom we want to schedule an appointment as the input and query for the available slots. This query inturn refers to the Slots table of the backend database and outputs those slots whose Available column of the Slots table is set to '1' for the input Doctor ID. Next, we select one of the slots from above depending upon the priority and we assign it to the patient with the given Patient ID. This information is added to the Appointment table of the backend database and we set the Available columns of the Slots table to '0' for the slot which is assigned just now.

### **Test Scheduling:**

This page is used to schedule the tests prescribed by the doctor. First, we enter the Test Name as the input and query for the available slots. This query inturn refers to the SlotsforTests table of the backend database and output those slots whose Available column of the SlotsforTests table is set to '1' for the input Test Name. Next, we select the first available slot from the above list and we assign it to the patient with the given Patient ID. This information is added to the Scheduled\_Tests table of the backend database and we set the Available column of the SlotsforTests table to '0' for the slot which is assigned just now.

### **Treatment Scheduling:**

This page is used to schedule the treatments prescribed by the doctor. First, we enter the Treatment Name as the input and query for the available slots. This query inturn refers to the SlotsforTreatments table of the backend database and output those slots whose Available column of the SlotsforTreatments table is set to '1' for the input Treatment Name. Next, we select the first available slot from the above list and we assign it to the patient with the given Patient ID. This information is added to the Scheduled\_Treatments table of the backend database and we set the Available columns of the SlotsforTreatments table to '0' for the slot which is assigned just now.

---

## **Functionalities Of Data Entry Operator:**

This role has 3 operations. Which are add Administered Treatments, View Administered Treatments, Add Test Results.

### **Pages Implemented:**

#### **Find Patient:**

This requires us to enter the patient ID and the appointment ID. If there exists an appointment in the Appointment Table with the entered ID then we will be redirected to the respected page for the three options which are redirected to View Administered Treatments, Add Test Results, Add Administered Treatments page for each option.

#### **View Administered Treatments:**

#### **Add Test Results:**

This displays the tests which were conducted in the past these contain even the ones which results were not entered. So we look up in the table and update the test results. Here we assumed that there are no same treatments within the same appointment for a patient.

#### **Add Administered Treatments:**

This page displays the Treatments that are completed till the time of access. As the tests entered in the Administered Treatments table are entered only after completion, we will display all the entries in the Administered Treatments table which match the patient ID and appointment ID given in the previous page. It can also take in an argument of Treatment Type and adds it to the database of the already entered patient ID and appointment ID's position and the table will be updated with the new Treatment entered.

### **Triggers implemented:**

#### **Patient\_age\_trigger:**

This trigger checks whether the patient age is less than zero or not. If the patient age entered by the front desk operator in the patient registration page is less than zero, then this trigger throws the error.

#### **Patient\_phone\_trigger:**

This trigger checks whether the length of the patient's mobile number is equal to 10 or not. If the patient mobile number entered by the front desk operator in the patient registration page has less than 10 digits, then this trigger throws the error.

#### **Appointment\_date\_trigger:**

This trigger ensures the appointment date entered by the front desk operator is not in the past. If the appointment date entered by the front desk operator is in the past, then this trigger throws the error.

#### **Appointment\_time\_trigger:**

This trigger ensures the appointment time entered by the front desk operator is chosen from the given list of slots. If the appointment time entered by the front desk operator is not in the given list of slots, then this trigger throws the error.

**test\_date\_trigger:**

This trigger ensures the test date entered by the front desk operator is not in past. If the test date entered by the front desk operator is in the past, then this trigger throws the error.

**test\_time\_trigger:**

This trigger ensures the test time entered by the front desk operator is chosen from the given list of slots. If the test time entered by the front desk operator is not in the given list of slots, then this trigger throws the error.

**treatment\_date\_trigger:**

This trigger ensures the treatment date entered by the front desk operator is not in the past. If the treatment date entered by the front desk operator is in the past, then this trigger throws the error.

**treatment\_time\_trigger:**

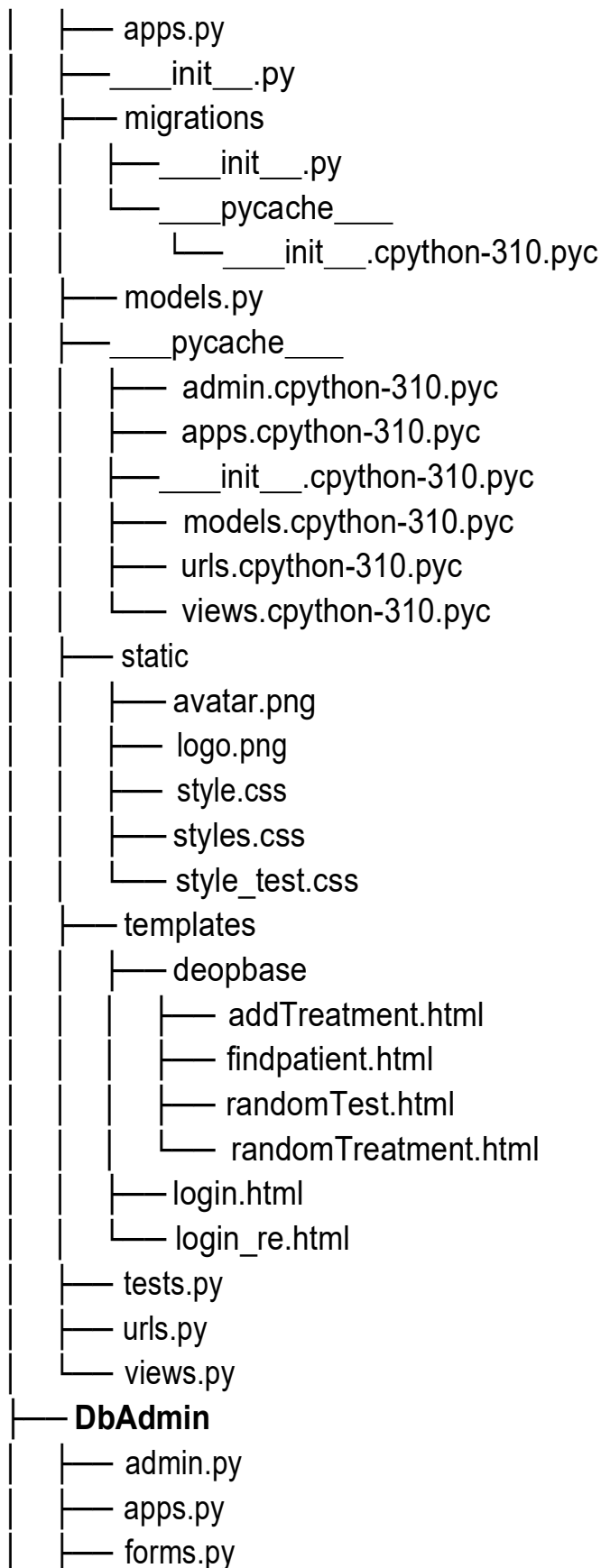
This trigger ensures the treatment time entered by the front desk operator is chosen from the given list of slots. If the treatment time entered by the front desk operator is not in the given list of slots, then this trigger throws the error.

## **Appendix : Code listing of Queries and Interfaces**

### **Appendix A : File Structure**

#### **hospital\_management\_system**

```
|— DataEntryOp
|  |— admin.py
```



```

├── __init__.py
├── migrations
│   ├── __init__.py
│   └── __pycache__
│       └── __init__.cpython-310.pyc
├── models.py
├── __pycache__
│   ├── admin.cpython-310.pyc
│   ├── apps.cpython-310.pyc
│   ├── forms.cpython-310.pyc
│   ├── __init__.cpython-310.pyc
│   ├── models.cpython-310.pyc
│   ├── urls.cpython-310.pyc
│   └── views.cpython-310.pyc
├── static
│   ├── avatar.png
│   └── logo.png
├── templates
│   ├── action.html
│   ├── add_deop.html
│   ├── add_doc.html
│   ├── add_fdop.html
│   ├── delete_deop.html
│   ├── delete_doc.html
│   ├── delete_fdop.html
│   ├── login.html
│   └── login_re.html
├── tests.py
├── urls.py
├── views.py
└── Doctor
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    └── migrations

```

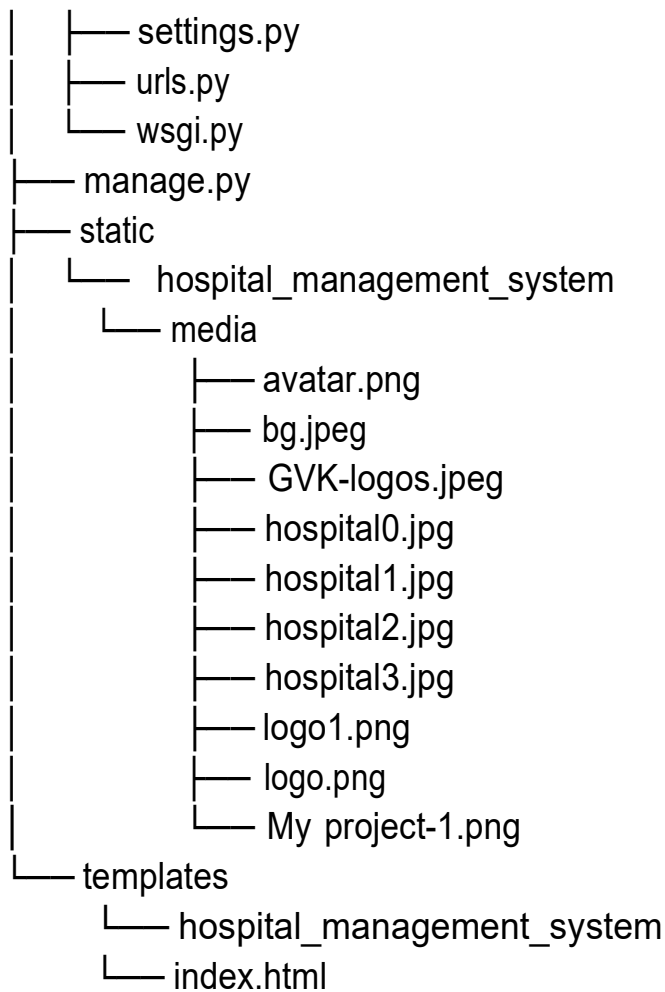


```

├── __pycache__
│   ├── admin.cpython-310.pyc
│   ├── apps.cpython-310.pyc
│   ├── __init__.cpython-310.pyc
│   ├── models.cpython-310.pyc
│   ├── urls.cpython-310.pyc
│   └── views.cpython-310.pyc
├── static
│   ├── avatar.png
│   ├── images
│   │   ├── patient_doctor.jpg
│   │   └── receptionist.png
│   ├── logo.png
│   └── style.css
├── templates
│   ├── assign_room.html
│   ├── doctor_appt.html
│   ├── frontdeskoperator.html
│   ├── login.html
│   ├── login_re.html
│   ├── patient_dis.html
│   ├── patient_reg.html
│   ├── test_scheduling.html
│   └── treatment_scheduling.html
├── tests.py
├── urls.py
├── views.py
└── hospital_management_system
    ├── asgi.py
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-310.pyc
    │   ├── settings.cpython-310.pyc
    │   ├── urls.cpython-310.pyc
    │   └── wsgi.cpython-310.pyc

```





## Appendix B : SQL Queries

```
CREATE TABLE Doctor(EmployeeID int NOT NULL,  
    Name varchar(30) NOT  
    Department varchar(30) NOT NULL,  
    Email varchar(30) NOT NULL,  
    Phone varchar(13) NOT NULL,  
    Password varchar(30) NOT NULL,  
    Working int NOT NULL,  
    PRIMARY KEY(EmployeeID));
```

```
CREATE TABLE FrontDeskOp(Username varchar(30) NOT NULL,  
    Password varchar(30) NOT NULL,  
    Email varchar(30) NOT NULL,
```

```
Name varchar(30) NOT NULL,  
PRIMARY KEY(Uname));
```

```
CREATE TABLE DataEntryOp(Uname varchar(30) NOT NULL,  
Password varchar(30) NOT NULL,  
Email varchar(30) NOT NULL,  
Name varchar(30) NOT NULL,  
PRIMARY KEY(Uname));
```

```
CREATE TABLE DbAdmin(Uname varchar(30) NOT NULL,  
Password varchar(30) NOT NULL,  
PRIMARY KEY(Uname));
```

```
CREATE TABLE Patient(PatientID int NOT NULL  
AUTO_INCREMENT,  
Name varchar(255) NOT NULL,  
Gender varchar(1) NOT NULL,  
Age int NOT NULL,  
Phone varchar(255) NOT NULL,  
Diagnosis varchar(255) NOT NULL,  
PRIMARY KEY(PatientID));
```

```
CREATE TABLE Appointment ( AppointmentID int NOT NULL  
AUTO_INCREMENT,  
Doctor int NOT NULL,  
Patient int NOT NULL,  
Date date NOT NULL,  
StartTime time NOT NULL,  
PRIMARY KEY (AppointmentID),  
FOREIGN KEY (Doctor) REFERENCES  
Doctor(EmployeeID),  
FOREIGN KEY (Patient) REFERENCES  
Patient(PatientID));
```

```
CREATE TABLE Prescribes_Medication(Doctor int NOT NULL,
```

```

        Patient int NOT NULL,
        Appointment int NOT NULL,
        Medication varchar(255) NOT NULL,
        PRIMARY KEY (Doctor, Patient,
Appointment),
        FOREIGN KEY (Doctor) REFERENCES
Doctor(EmployeeID),
        FOREIGN KEY (Patient) REFERENCES
Patient(PatientID),
        FOREIGN KEY (Appointment)
REFERENCES Appointment(AppointmentID));

```

```

CREATE TABLE Prescribes_Treatment(Doctor int NOT NULL,
        Patient int NOT NULL,
        Appointment int NOT NULL,
        Date DATE NOT NULL,
        Treatment varchar(255) NOT NULL,
        PRIMARY KEY (Doctor, Patient,
Appointment,Date),
        FOREIGN KEY (Doctor) REFERENCES
Doctor(EmployeeID),
        FOREIGN KEY (Patient) REFERENCES
Patient(PatientID),
        FOREIGN KEY (Appointment)
REFERENCES Appointment(AppointmentID));

```

```

CREATE TABLE Prescribes_Test(Doctor int NOT NULL,
        Patient int NOT NULL,
        Appointment int NOT NULL,
        Date DATE NOT NULL,
        Test varchar(255) NOT NULL,
        Results varchar(1000),
        PRIMARY KEY (Doctor, Patient,
Appointment,Date),

```

```
                FOREIGN KEY (Doctor) REFERENCES
Doctor(EmployeeID),
                FOREIGN KEY (Patient) REFERENCES
Patient(PatientID),
                FOREIGN KEY (Appointment) REFERENCES
Appointment(AppointmentID));
```

```
CREATE TABLE Room(Number int NOT NULL,
                    Type varchar(50) NOT NULL,
                    Available boolean NOT NULL,
                    PRIMARY KEY(Number));
```

```
CREATE TABLE Stay(Patient int NOT NULL,
                   Room int NOT NULL,
                   PRIMARY KEY(Patient, Room),
                   FOREIGN KEY (Patient) REFERENCES
Patient(PatientID),
                   FOREIGN KEY (Room) REFERENCES
Room(Number));
```

```
CREATE TABLE Slots(Doctor int NOT NULL,
                    Date date NOT NULL,
                    StartTime time NOT NULL,
                    EndTime time NOT NULL,
                    Available boolean NOT NULL,
                    FOREIGN KEY (Doctor) REFERENCES
Doctor(EmployeeID), PRIMARY KEY(Doctor,
Date, StartTime));
```

```
CREATE TABLE SlotsforTests(Test varchar(100) NOT NULL,  
                             Date date NOT NULL,  
                             StartTime time NOT NULL,  
                             Available boolean NOT NULL,  
                             PRIMARY KEY(Test, Date, StartTime));
```

```
CREATE TABLE SlotsforTreatments(  
    Treatment varchar(100) NOT NULL,  
    Date date NOT NULL,  
    StartTime time NOT NULL,  
    Available boolean NOT NULL,  
    PRIMARY KEY(Treatment, Date, StartTime));
```

```
CREATE TABLE Scheduled_Tests(Patient int NOT NULL,  
                               Test varchar(100) NOT NULL,  
                               Date date NOT NULL,  
                               StartTime time NOT NULL, PRIMARY  
                               KEY(Patient, Test, Date, StartTime));
```

```
CREATE TABLE Scheduled_Treatments(Patient int NOT NULL,  
    Treatment varchar(100) NOT NULL,  
    Date date NOT NULL,  
    StartTime time NOT NULL,  
    PRIMARY KEY(Patient, Treatment, Date, StartTime));
```

```
CREATE TABLE AdministeredTreatments(Patient int NOT NULL,  
    Appointment int NOT NULL,  
    Treatment varchar(50) NOT NULL,  
    PRIMARY KEY(Patient, Appointment, Treatment),  
    FOREIGN KEY(Patient) REFERENCES  
    Patient(PatientID),
```

## FOREIGN KEY(Appointment) REFERENCES Appointment(AppointmentID));



Delete Existing User

List of Front-Desk Operators

- arjun
- karen
- meghan

Username:

Delete

Back

Database Administrator

Logout

Front-Desk Operator

Add new

Delete existing

Doctor

Add new

Delete existing


Data-Entry Operator

Add new

Delete existing

© 2023 All Rights Reserved

Add Front-Desk Operator



Username:

arjun

Password:

\*\*\*

Email:

arj12@gmail.com

Name:

arjun

Create

Delete Existing User

List of Front-Desk Operators

- arjun
- karen
- mchan

Username:

Delete

Back

Welcome to the Front Desk Operator Website

We are dedicated to providing the best possible service to our patients

Dashboard

Logout

Patient Registration

Patient Discharge

Doctor Appointment

Test Scheduling

Assign Room

Treatment Scheduling



© 2023 Front Desk Operator Website. All Rights Reserved.

Patient Registration

Name:

vinay

Gender:

Male

Age:

21

Phone:

9123456789

Diagnosis:

fever

submit

Treatment Scheduling

Available Slots:

Treatment Name	Slot Start Time
Infusion	8:00:00
Infusion	11:00:00
Infusion	15:00:00
Infusion	17:00:00

Treatment Name:

Infusion

Get Available Slots

Patient Id:



Treatment Name:

[Get Available Slots](#)

Patient Id:

Treatment Name:

Treatment Date:

Treatment Time:

[Schedule Treatment](#)

[Back](#)

Bloodtest	8:00:00
Bloodtest	9:00:00
Bloodtest	10:00:00
Bloodtest	11:00:00
Bloodtest	14:00:00
Bloodtest	15:00:00
Bloodtest	16:00:00
Bloodtest	17:00:00

Test Name:

[Get Available Slots](#)

Test Name:

[Get Available slots](#)

Patient Id:

Test Name:

Test Date:

Test Time:

[Schedule Test](#)

[Back](#)

130	Personal Room
131	Personal Room
132	Personal Room

Patient ID:

Room Type:

Room Number:

Submit

Back

## Patient Discharge

Patient ID:

Submit

Back

2	9:00:00
2	10:00:00
2	11:00:00
2	14:00:00
2	15:00:00
2	16:00:00
2	17:00:00

Doctor Id:

Get Available Slots

Doctor Id:

Patient Id:

Appointment Date:

Appointment Time:

Schedule Appointment

Back

Doctor Dashboard

Dr.vinod

Logout

Patient Records

Patient ID	Name	Gender	Age	Diagnosis
1	Rahul	M	25	bone fracture
19	vinay	M	21	fever

Query Patient Information

Patient ID:

Query

Patient ID	Name	Gender	Age	Mobile Number	Diagnosis	Appointment ID	Date	Medicine List	Treatment	Tests	Results
19	vinay	M	21	9123456789	fever	36	March 15, 2023		None	None	None

Query

Record Medicines

Patient ID:

Appointment ID:

Medicine:

Treatment:

Test:

Record

© 2023 Doctor Dashboard. All rights reserved.

## Data Entry Operator

Patient ID:

Appointment ID:

Action:

## Treatment Table

### Treatments Done Previously

Patient ID	Treatment Type
------------	----------------

## Add Administered Treatment

Treatment Name

## Test Results

### Tests Done Previously

Test Type	Appointment ID	Date	Result
Bloodtest	36	March 15, 2023	None

## Add Tests

Test Type:

Test Result: