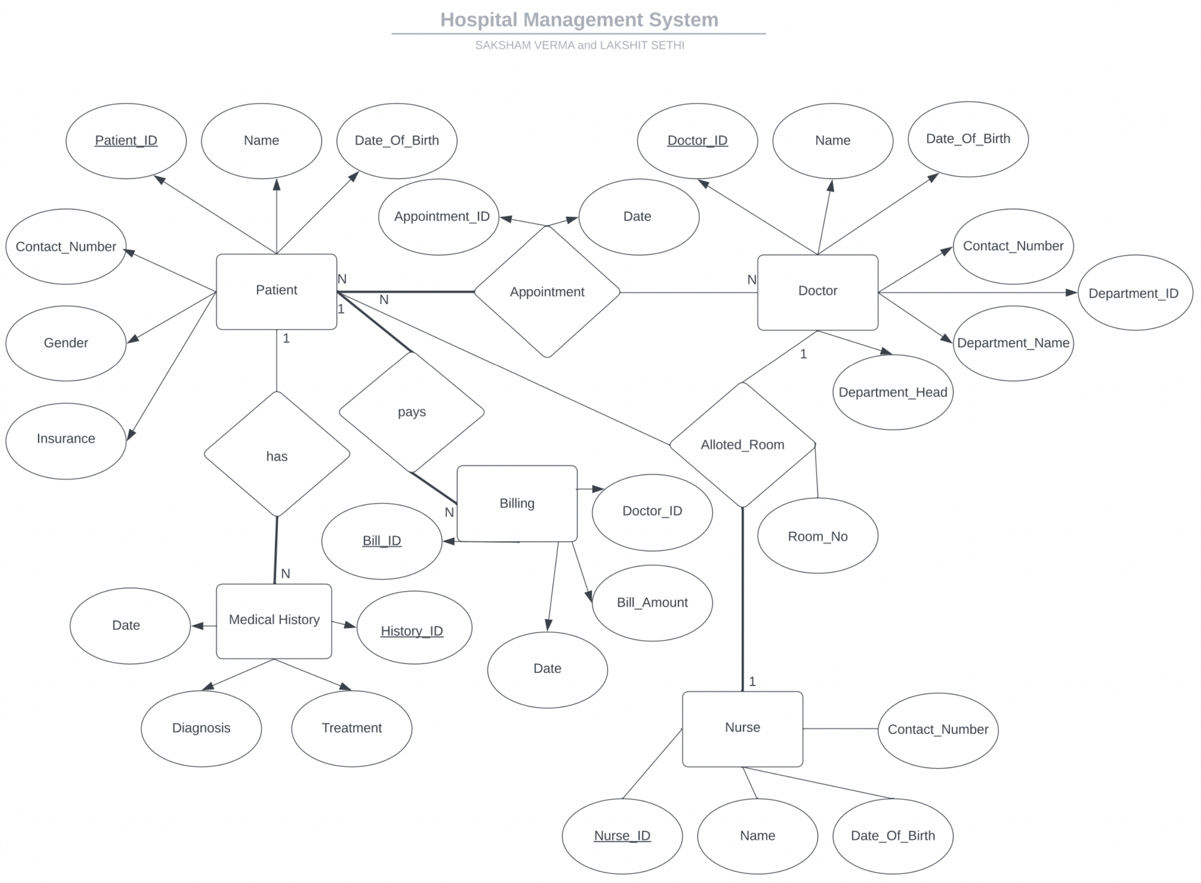
HOSPITAL MANAGEMENT SYSTEM

Database Systems Project,Spring Semester 2023

SAKSHAM VERMA(2021A7PS2414P)
LAKSHIT SETHI(2021A7PS2434P)

Video Link(https://drive.google.com/drive/folders/1kQkvpqe0EV44yhBrwvoUAT7k-ZH6PaE9?usp=sharing)

ER DIAGRAM



Entities:

1. Patient-This entity represents the patient who is being treated in the hospital.The attributes of this entity are:

- Patient_ID (Primary Key):A unique identifier assigned to each patient.
- Name: The name of the patient.
- Date_Of_Birth: The date of birth of the patient.
- Contact_Number: The contact number of the patient.
- Gender: The gender of the patient.
- Insurance: Whether the patient has insurance or not.

2. Doctor - This entity represents a doctor who is responsible for treating patients. The attributes of this entity are:

- Doctor_ID (Primary Key): A unique identifier assigned to each doctor.
- Name: The name of the doctor.
- Date_Of_Birth: The date of birth of the doctor.
- Contact_Number: The contact number of the doctor.
- Department_ID:The unique identifier of the department where the doctor works.
- Department_Name: The name of the department where the doctor works.
- Department_Head: The head of the department where the doctor works.

3. Nurse - This entity represents a nurse/staff who is responsible for taking care of patients. The attributes of this entity are:

- Nurse_ID (Primary Key): A unique identifier assigned to each nurse.
- Name: The name of the nurse.
- Date_Of_Birth: The date of birth of the nurse.
- Contact_Number: The contact number of the nurse.

4. Medical History - This entity represents the medical history of a patient. The attributes of this entity are:

- History_ID (Primary Key): A unique identifier assigned to each medical history record.
- Date: The date on which the medical history record was created.
- Diagnosis: The diagnosis given to the patient.
- Treatment: The treatment given to the patient.

5. Billing - This entity represents the billing information for a patient. The attributes of this entity are:

- Bill_ID (Primary Key): A unique identifier assigned to each billing record.
- Date: The date on which the bill was generated.
- Bill_Amount: The amount of the bill.
- Doctor_ID:Represents the doctor id of the doctor who treated the patient.

Relationships:

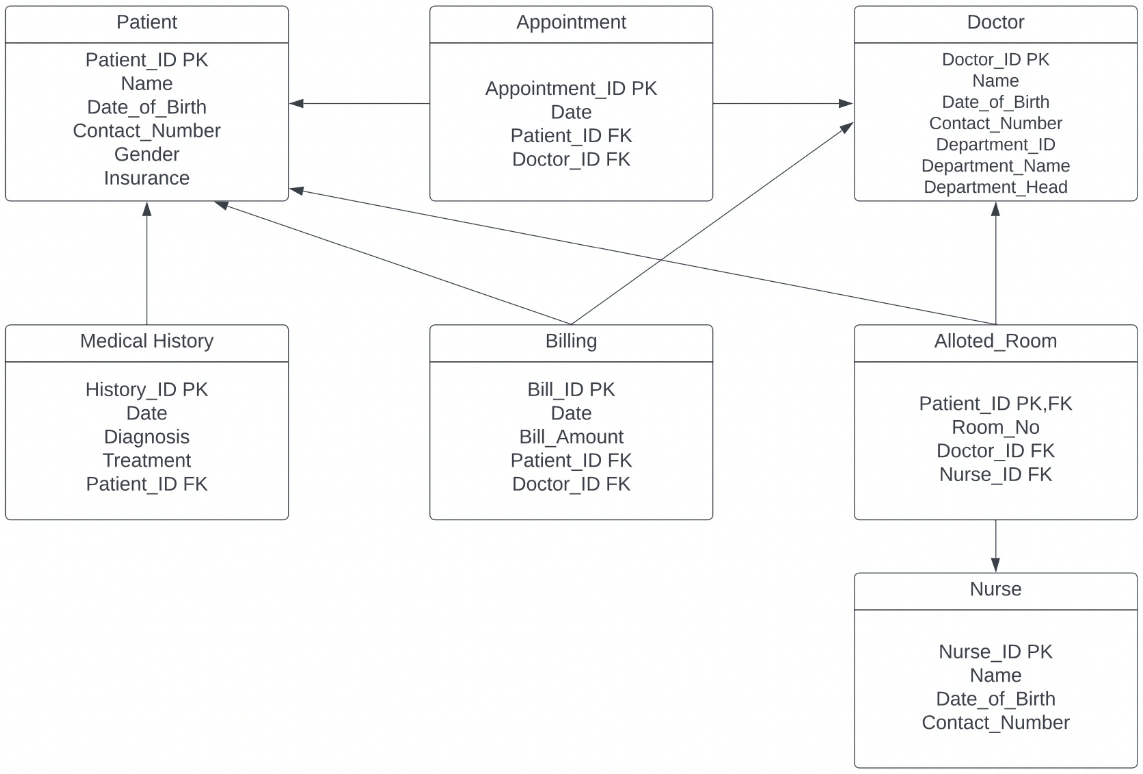
1. Has - This relationship connects the Patient entity with the Medical History entity. It represents the fact that each patient has one or more medical histories. The cardinality of this relationship is 1:N, meaning that each patient can have multiple medical histories, but each medical history belongs to only one patient.

2. Alloted_Room - This relationship connects the Patient, Doctor, and Nurse entities. It represents the fact that each patient, doctor, and nurse is assigned to a room. The cardinality of this relationship is N:1:1, meaning that every patient will have just one doctor and nurse while each doctor and nurse will have many patients.

3. Appointment - This relationship connects Patient and Doctor entities. It represents the fact that each patient can have one or more appointments with a doctor, and each doctor can have one or more appointments with a patient. The cardinality of this relationship is N:N (Note:Although it was clarified that the cardinality of appointment is 1:1 I found N:N cardinality more suitable for my ER diagram as each patient can be treated by more than one doctor and one doctor can treat more than one patient.)

4. Pays - This relationship connects the Patient entity with the Billing entity. It represents the fact that each patient can have one or more bills, and each bill has one patient.

Relational Model (Before Normalization)



1. Patient Table - This entity represents the patient who is being treated in the hospital.The attributes of this entity are:

- Patient_ID (Primary Key):A unique identifier assigned to each patient.
- Name: The name of the patient.
- Date_Of_Birth: The date of birth of the patient.
- Contact_Number: The contact number of the patient.
- Gender: The gender of the patient.
- Insurance: Whether the patient has insurance or not.

2. Doctor Table - This entity represents a doctor who is responsible for treating patients. The attributes of this entity are:

- Doctor_ID (Primary Key): A unique identifier assigned to each doctor.
- Name: The name of the doctor.
- Date_Of_Birth: The date of birth of the doctor.
- Contact_Number: The contact number of the doctor.
- Department_ID:The unique identifier of the department where the doctor works.
- Department_Name: The name of the department where the doctor works.
- Department_Head: The head of the department where the doctor works.

3. Nurse Table - This entity represents a nurse/staff who is responsible for taking care of patients. The attributes of this entity are:

- Nurse_ID (Primary Key): A unique identifier assigned to each nurse.
- Name: The name of the nurse.
- Date_Of_Birth: The date of birth of the nurse.
- Contact_Number: The contact number of the nurse.

4. Medical History Table - This entity represents the medical history of a patient. The attributes of this entity are:

- History_ID (Primary Key): A unique identifier assigned to each medical history record.
- Date: The date on which the medical history record was created.
- Diagnosis: The diagnosis given to the patient.
- Treatment: The treatment given to the patient.

- Patient_ID (Foreign Key): A unique identifier assigned to the patient.Since the ‘has’ relationship between medical history and patient was N:1 the relationship was removed when converting ER diagram to Relational Schema and Patient ID which is the Primary Key of Patient Table was added as a foreign key to the medical history table.

5. Billing Table - This entity represents the billing information for a patient. The attributes of this entity are:

- Bill_ID (Primary Key): A unique identifier assigned to each billing record.
- Date: The date on which the bill was generated.
- Bill_Amount: The amount of the bill.
- Doctor_ID:Represents the doctor id of the doctor who treated the patient.
- Patient_ID (Foreign Key): A unique identifier assigned to the patient.Since the ‘pays’ relationship between Billing and patient was N:1 the relationship was removed when converting ER diagram to Relational Schema and Patient ID which is the Primary Key of Patient Table was added as a foreign key to the Billing table.

6. Alloted_Room Table**(Weak Entity)** - The Alloted_Room table represents the relationship between Patient, Doctor, and Nurse.In the ER Diagram,Alloted_Room was a relationship with cardinality N:1:1.This relationship was thus converted into a table.Since cardinality was N:1:1,the primary key of this table will be the primary key of the table with the many cardinality i.e.,Patient Table.It has the following attributes:

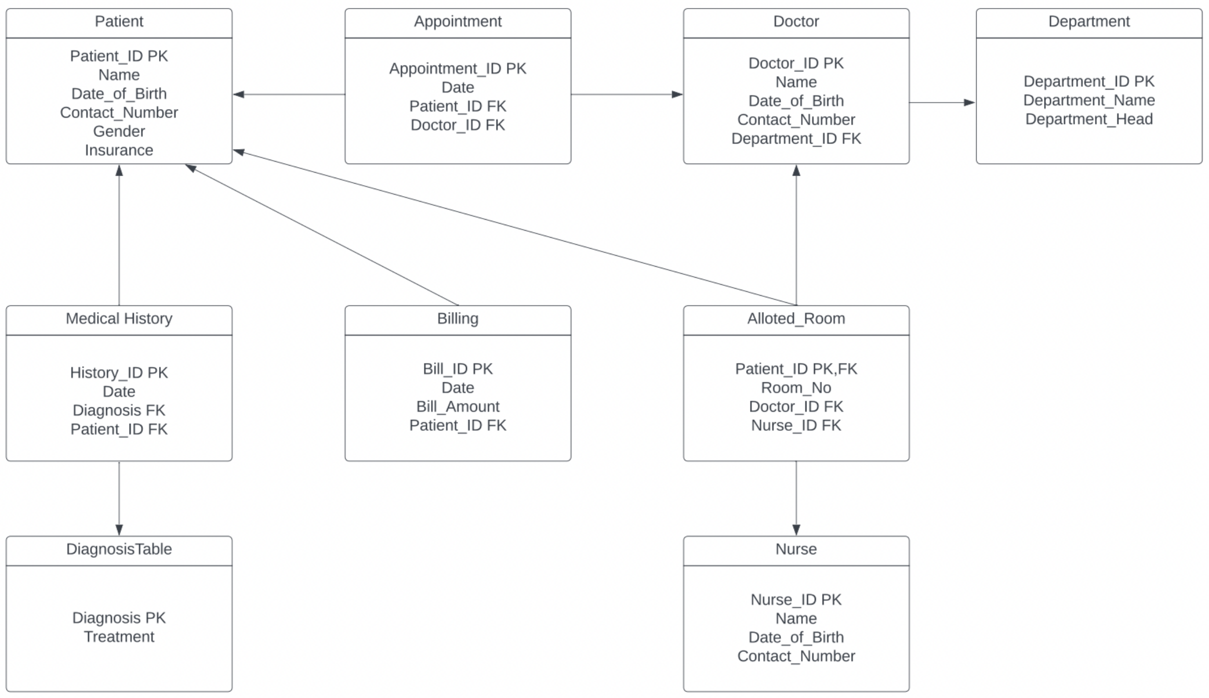
- Patient_ID (Primary Key)(Foreign Key): a reference to the Patient table, indicating the patient that is allotted the room.
- Room_No: the room number allotted to the patient.
- Doctor_ID (Foreign Key): a reference to the Doctor table, indicating the doctor that is allotted to the patient's room.
- Nurse_ID (Foreign Key): a reference to the Nurse table, indicating the nurse that is

allotted to the patient's room.

7. Appointment Table -The Appointment table represents the relationship between Patient and Doctor in the given ER diagram.In the ER Diagram,Appointment was a relationship with cardinality N:N.This relationship was thus converted into a table.Since the cardinality was N:N,the primary key of this table will consist of its attribute but not of the primary key of the two entities it is relating i.e.,Patient and Doctor.It has the following attributes:

- Appointment_ID (Primary Key): a unique identifier for each appointment.
- Diagnosis: the diagnosis of the appointment.
- Date: the date of the appointment.
- Patient_ID (Foreign Key): a reference to the Patient table, indicating the patient who has the appointment.
- Doctor_ID (Foreign Key): a reference to the Doctor table, indicating the doctor who has the appointment.

Relational Model (After Normalization)



To convert the relational model to 3NF, we followed the following steps:

1.Check which normal form the schema is currently in:

- Since there are no multivalued attributes,schema is in 1NF form
- Since every table has only one primary key,the schema is in 2NF form as well.
- There are some transitive dependencies in the schema in the Medical_History and Doctor table so the schema is not in 3NF form.

Therefore the schema is in 2NF form and we need to convert it into 3NF form using normalization.

Step 1: Identify the functional dependencies in each table.

Doctor (Doctor_ID PK, Name, Date_Of_Birth, Contact_Number,Department_ID, Department_Name, Department_Head)

Functional dependencies:

- Doctor_ID -> Name, Date_Of_Birth, Contact_Number, Department_Name, Department_Head
- Department_ID -> Department_Name,Department_Head

Medical_History (History_ID PK, Date, Diagnosis, Treatment, Patient_ID FK references Patient(Patient_ID))

Functional dependencies:

- History_ID -> Date, Diagnosis, Treatment, Patient_ID
- Diagnosis -> Treatment

The rest of the tables have all attributes dependent only on the primary key,i.e.,they are already in 3NF form so we don't need to do anything further on them.

Step 2: Check for transitive dependencies and remove them by creating new tables.

In the Doctor table, we can see that Department_Head and Department_Name are dependent on Department_ID. This means there is a transitive dependency, and the table is not in 3NF. To remove this dependency, we can create a new table for departments.

Doctor (Doctor_ID PK, Name, Date_Of_Birth, Contact_Number, Department_ID FK references Department(Department_ID))

Department (Department_ID PK,Department_Name, Department_Head)

In the Medical_History table, we can see that Treatment is dependent on Diagnosis. This means there is a transitive dependency, and the table is not in 3NF. To remove this dependency, we can create a new table for diagnoses.

Medical_History (History_ID PK, Date, Diagnosis FK references DiagnosisTable(Diagnosis), Patient_ID FK references Patient(Patient_ID))

DiagnosisTable (Diagnosis PK, Treatment)

Step 3: Check for any remaining functional dependencies that violate 3NF.

All the tables are now in 3NF.

The final normalized relational model is:

Patient (Patient_ID PK, Name, Date_Of_Birth, Contact_Number, Gender, Insurance)

Doctor (Doctor_ID PK, Name, Date_Of_Birth, Contact_Number, Department_ID FK references Department(Department_ID))

Department (Department_ID PK,Department_Name, Department_Head)

Nurse (Nurse_ID PK, Name, Date_Of_Birth, Contact_Number)

Medical_History (History_ID PK, Date, Diagnosis FK references DiagnosisTable(Diagnosis), Patient_ID FK references Patient(Patient_ID))

DiagnosisTable (Diagnosis PK, Treatment)

Alloted_Room (Patient_ID FK references Patient(Patient_ID), Doctor_ID FK references Doctor(Doctor_ID), Nurse_ID FK references Nurse(Nurse_ID), Room_No)

Appointment (Appointment_ID PK, Date, Patient_ID FK references Patient(Patient_ID), Doctor_ID FK references Doctor(Doctor_ID))

Bill (Bill_ID PK, Date, Bill_Amount,Doctor_ID FK references Doctor(Doctor_ID), Patient_ID FK references Patient(Patient_ID))

The above relational schema has been normalized and is now in 3NF form.

SQL Tables and Queries

Creating Tables and Populating them

```
CREATE DATABASE IF NOT EXISTS HOSPITAL;

USE HOSPITAL;


CREATE TABLE IF NOT EXISTS Department (

    Department_ID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,

    Department_Name VARCHAR(255),

    Department_Head TEXT,

    PRIMARY KEY (Department_ID)

);


CREATE TABLE IF NOT EXISTS DOCTOR (

    Doctor_ID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,

    Name TEXT,

    DOB DATE,

    Contact_Number BIGINT(10) CHECK (LENGTH(Contact_Number) = 10),

    Department_ID INTEGER NOT NULL ,

    FOREIGN KEY (Department_ID) REFERENCES Department (Department_ID),

    PRIMARY KEY (Doctor_ID)

);

CREATE TABLE IF NOT EXISTS Nurse (

    Nurse_ID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,

    Name TEXT,

    DOB DATE,

    Contact_Number BIGINT(10) CHECK (LENGTH(Contact_Number) = 10),

    PRIMARY KEY (Nurse_ID)

);
```

```
CREATE TABLE IF NOT EXISTS Patient (
    Patient_ID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
    Name TEXT,
    Contact_Number BIGINT(10) CHECK (LENGTH(Contact_Number) = 10),
    DOB DATE,
    Gender VARCHAR(10),
    Insurance VARCHAR(10),
    PRIMARY KEY (Patient_ID)
);

CREATE TABLE IF NOT EXISTS Alloted_Room (
    Room_No INTEGER NOT NULL,
    Patient_ID INTEGER NOT NULL,
    Nurse_ID INTEGER NOT NULL,
    Doctor_ID INTEGER NOT NULL,
    PRIMARY KEY (Patient_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient (Patient_ID),
    FOREIGN KEY (Nurse_ID) REFERENCES Nurse (Nurse_ID),
    FOREIGN KEY (Doctor_ID) REFERENCES DOCTOR (Doctor_ID)
);

CREATE TABLE IF NOT EXISTS DiagnosisTable (
    Diagnosis VARCHAR(100),
    Treatment VARCHAR(100),
    PRIMARY KEY (Diagnosis)
);
```

```
CREATE TABLE IF NOT EXISTS Medical_History (
    History_ID INTEGER NOT NULL UNIQUE AUTO_INCREMENT,
    Date DATE,
    Diagnosis VARCHAR(100),
    Patient_ID INT,
    PRIMARY KEY (History_ID),
    FOREIGN KEY (Diagnosis) REFERENCES DiagnosisTable(Diagnosis),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);

CREATE TABLE IF NOT EXISTS Appointment (
    Appointment_ID INTEGER NOT NULL AUTO_INCREMENT,
    Patient_ID INTEGER NOT NULL,
    Doctor_ID INTEGER NOT NULL,
    A_date date NOT NULL,
    PRIMARY KEY (Appointment_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    FOREIGN KEY (Doctor_ID) REFERENCES DOCTOR(Doctor_ID)
);

CREATE TABLE IF NOT EXISTS Billing (
    Patient_ID INTEGER,
    Bill_ID INTEGER NOT NULL AUTO_INCREMENT,
    Bill_Amount INTEGER,
    Doctor_ID INTEGER,
    PRIMARY KEY (Bill_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    FOREIGN KEY (Doctor_ID) REFERENCES DOCTOR(Doctor_ID)
);
```

```
INSERT INTO Department (Department_Name, Department_Head, Contact_Number)
VALUES
    ('Cardiology', 'Dr. Suresh Sharma', 9876543210),
    ('Oncology', 'Dr. Meera Singh', 8765432109),
    ('Neurology', 'Dr. Sanjay Patel', 7654321098),
    ('Orthopedics', 'Dr. Ravi Kumar', 6543210987),
    ('Gynecology', 'Dr. Rekha Gupta', 5432109876),
    ('Pediatrics', 'Dr. Shalini Verma', 4321098765),
    ('General Medicine', 'Dr. Rakesh Singh', 3210987654),
    ('Dermatology', 'Dr. Pooja Sharma', 2109876543);


INSERT INTO DOCTOR (Name, DOB, Contact_Number, Department_ID) VALUES
  ('Dr. Ramesh Gupta', '1980-06-05', 9876543210, 1),
  ('Dr. Preeti Singh', '1992-08-12', 8765432101, 2),
  ('Dr. Rajesh Sharma', '1978-03-25', 7654321098, 1),
  ('Dr. Priya Patel', '1985-11-15', 6543210987, 3),
  ('Dr. Sanjay Verma', '1990-01-20', 5432109876, 2),
  ('Dr. Anjali Mehta', '1982-07-08', 4321098765, 1),
  ('Dr. Sameer Khan', '1995-04-03', 3210987654, 3),
  ('Dr. Nisha Choudhary', '1989-09-29', 2109876543, 2),
  ('Dr. Deepak Patel', '1983-12-18', 1098765432, 1),
  ('Dr. Shalini Gupta', '1993-02-28', 9876543210, 3),
  ('Dr. Rohit Singh', '1987-10-07', 8765432101, 2),
  ('Dr. Kiran Sharma', '1981-05-21', 7654321098, 1),
  ('Dr. Rohini Patel', '1994-11-01', 6543210987, 3),
  ('Dr. Vikas Verma', '1986-01-12', 5432109876, 2),
  ('Dr. Anuja Mehta', '1991-08-31', 4321098765, 1);
```

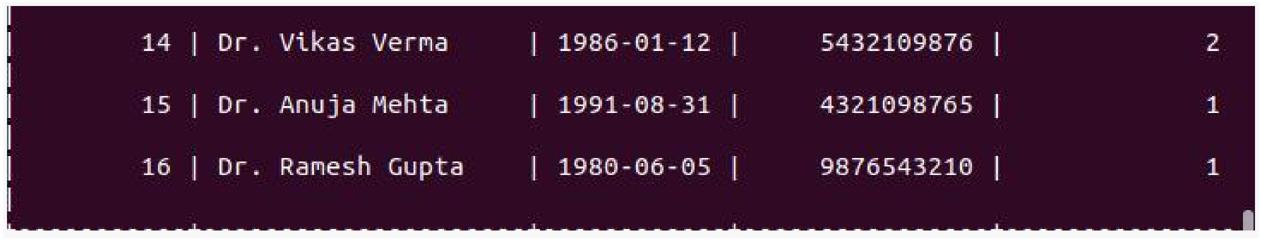
```
INSERT INTO Nurse (Name, DOB, Contact_Number)
VALUES
    ('Asha Singh', '1988-06-10', 9876543210),
    ('Rohit Sharma', '1993-12-15', 8765432109),
    ('Meena Patel', '1975-03-20', 7654321098),
    ('Komal Gupta', '1982-08-07', 6543210987),
    ('Manoj Kumar', '1990-11-25', 5432109876),
    ('Neha Singh', '1988-04-12', 4321098765),
    ('Sudha Sharma', '1986-01-30', 3210987654),
    ('Ravi Patel', '1979-09-05', 2109876543),
    ('Rekha Kumari', '1995-06-17', 1098765432),
    ('Anil Verma', '1983-12-22', 9876543210),
    ('Suman Singh', '1991-05-08', 8765432109),
    ('Amita Sharma', '1976-02-14', 7654321098),
    ('Rajeshwari Patel', '1989-10-01', 6543210987),
    ('Krishna Verma', '1981-07-18', 5432109876),
    ('Sanjay Kumar', '1977-04-05', 4321098765);
```

SQL Queries:

1. Insertion of doctor with Specialization:

```
INSERT INTO DOCTOR (Name,DOB,Contact_Number,Department_ID) VALUES ('Dr. Ramesh Gupta',
'1980-06-05', 9876543210, 1);
```

This query inserts a new doctor's record into the "DOCTOR" table with the specified name, date of birth, contact number, and department ID.

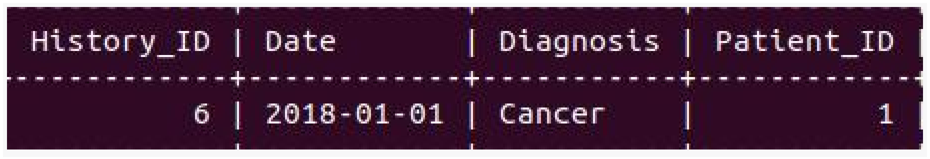


```
|        14 | Dr. Vikas Verma     | 1986-01-12 |     5432109876 |             2
|        15 | Dr. Anuja Mehta     | 1991-08-31 |     4321098765 |             1
|        16 | Dr. Ramesh Gupta    | 1980-06-05 |     9876543210 |             1
```

2. Insertion of Diagnosis and Patient Records:

```
INSERT INTO DiagnosisTable(Diagnosis, Treatment) VALUES ('Cancer', 'Chemotherapy');

INSERT INTO Medical_History (Date, Diagnosis, Patient_ID) VALUES ('2018-01-01',
'Cancer', 1);
```

This query inserts a new diagnosis record into the "Diagnosis" table with the specified diagnosis and treatment.This query inserts a new medical history record into the "Medical_History" table with the specified date, diagnosis, and patient ID.

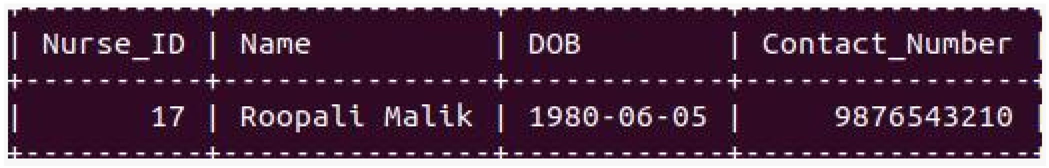


```
 History_ID | Date       | Diagnosis | Patient_ID |
------------+------------+-----------+------------+
          6 | 2018-01-01 | Cancer    |          1 |
```

3. Insertion of Hospital Staff:

```
INSERT INTO Nurse (Name,DOB,Contact_Number) VALUES ('Roopali Malik', '1980-06-05',
9876543210);
```

This query inserts a new nurse record into the "Nurse" table with the specified name, date of birth, and contact number.

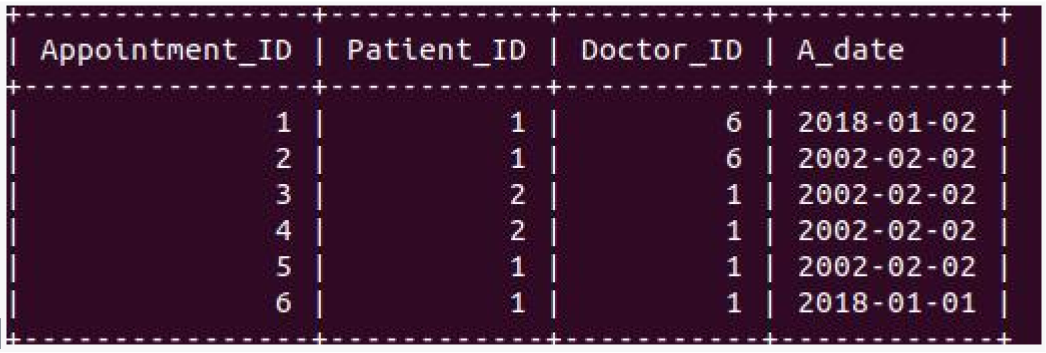


```
+----------+---------------+------------+----------------+
| Nurse_ID | Name          | DOB        | Contact_Number |
+----------+---------------+------------+----------------+
|       17 | Roopali Malik | 1980-06-05 |     9876543210 |
+----------+---------------+------------+----------------+
```

4. Booking an appointment with specialized doctor:

```
INSERT INTO Appointment (Patient_ID,Doctor_ID,A_date) VALUES (1, 1, '2018-01-01');
```

This query inserts a new appointment record into the "Appointment" table with the specified patient ID, doctor ID, and appointment date.



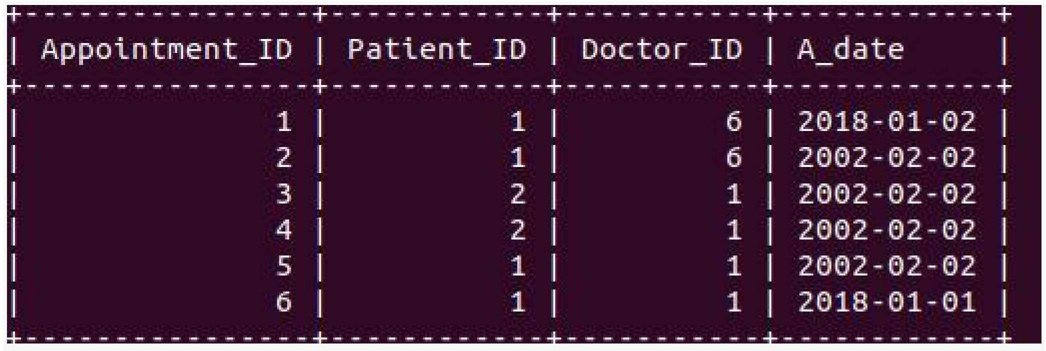
```
+----------------+------------+-----------+------------+
| Appointment_ID | Patient_ID | Doctor_ID | A_date     |
+----------------+------------+-----------+------------+
|              1 |          1 |         6 | 2018-01-02 |
|              2 |          1 |         6 | 2002-02-02 |
|              3 |          2 |         1 | 2002-02-02 |
|              4 |          2 |         1 | 2002-02-02 |
|              5 |          1 |         1 | 2002-02-02 |
|              6 |          1 |         1 | 2018-01-01 |
+----------------+------------+-----------+------------+
```

]

5. Rescheduling Appointment Date:

```
UPDATE Appointment SET A_date='2018-01-02' WHERE Appointment_ID=1;
```

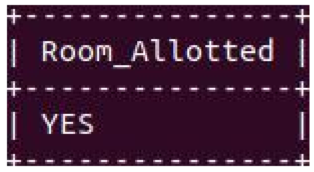
This query updates the appointment date for the specified appointment ID in the "Appointment" table.


```
+----------------+------------+-----------+------------+
| Appointment_ID | Patient_ID | Doctor_ID | A_date     |
+----------------+------------+-----------+------------+
|              1 |          1 |         6 | 2018-01-02 |
|              2 |          1 |         6 | 2002-02-02 |
|              3 |          2 |         1 | 2002-02-02 |
|              4 |          2 |         1 | 2002-02-02 |
|              5 |          1 |         1 | 2002-02-02 |
|              6 |          1 |         1 | 2018-01-01 |
+----------------+------------+-----------+------------+
```

6. Updating Room Status(Availability):

```
SELECT IF(EXISTS(SELECT * FROM Alloted_Room WHERE Room_No = 2), 'YES', 'NO') AS
Room_Allotted;
```

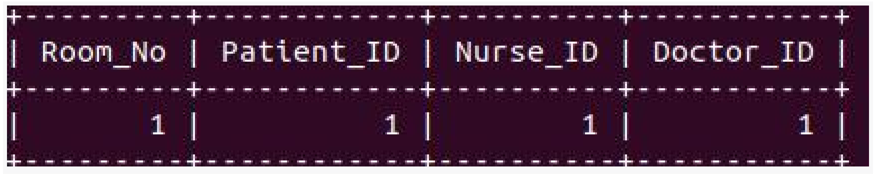
Checks whether the room has been allotted or not and returns 'YES' or 'NO' accordingly.


```
+---------------+
| Room_Allotted |
+---------------+
| YES           |
+---------------+
```

7. Allot Staff to a particular Room:

```
INSERT INTO Alloted_Room (Room_No, Patient_ID, Nurse_ID, Doctor_ID) VALUES (1, 1, 1,
1);
```

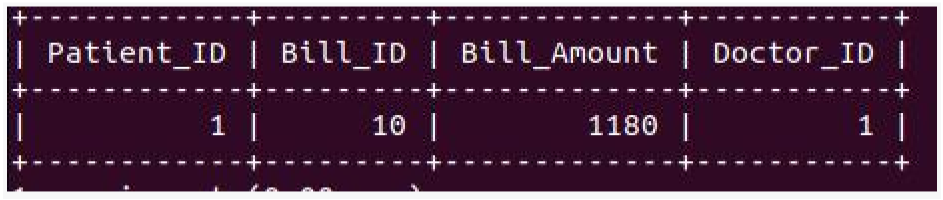
This query inserts a new record into the "Alloted_Room" table with the specified room number, patient ID, nurse ID, and doctor ID.


```
+---------+------------+----------+-----------+
| Room_No | Patient_ID | Nurse_ID | Doctor_ID |
+---------+------------+----------+-----------+
|       1 |          1 |        1 |         1 |
+---------+------------+----------+-----------+
```

8. Generating Bills:

```
INSERT INTO Billing (Patient_ID, Doctor_ID, Bill_Amount) VALUES (1, 1, 1000);
```

This query inserts a new billing record into the "Billing" table with the specified patient ID, doctor ID, and bill amount.

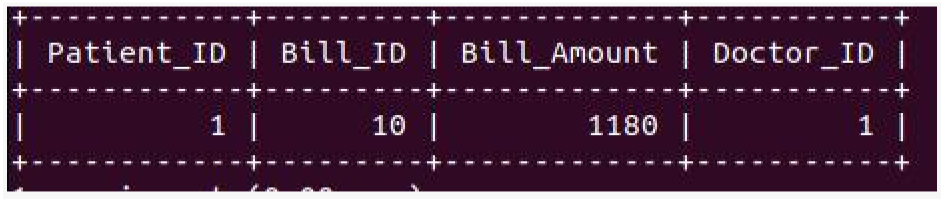


```
+------------+---------+-------------+-----------+
| Patient_ID | Bill_ID | Bill_Amount | Doctor_ID |
+------------+---------+-------------+-----------+
|          1 |      10 |        1180 |         1 |
+------------+---------+-------------+-----------+
```

9. Trigger to update the amount in billing table:

```
DELIMITER $$

CREATE TRIGGER add_tax_on_insert BEFORE INSERT ON Billing

FOR EACH ROW

BEGIN

   SET NEW.Bill_Amount = NEW.Bill_Amount * 1.18;

END $$

DELIMITER ;
```

This trigger is executed before inserting a new record into the "Billing" table. It updates the bill amount by adding an 18% tax to the specified bill amount.

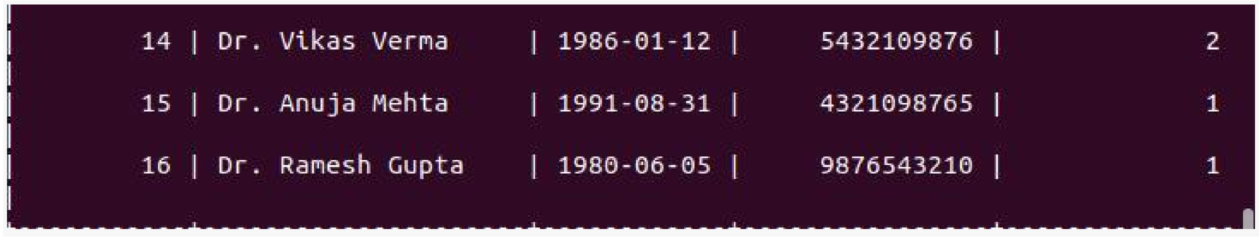


```
+------------+---------+-------------+-----------+
| Patient_ID | Bill_ID | Bill_Amount | Doctor_ID |
+------------+---------+-------------+-----------+
|          1 |      10 |        1180 |         1 |
+------------+---------+-------------+-----------+
```

10. View all doctors in the hospital:

```
CREATE VIEW All_Doctors AS

SELECT * FROM DOCTOR;
```

This query creates a view called "All_Doctors" that selects all the columns from the "DOCTOR" table.

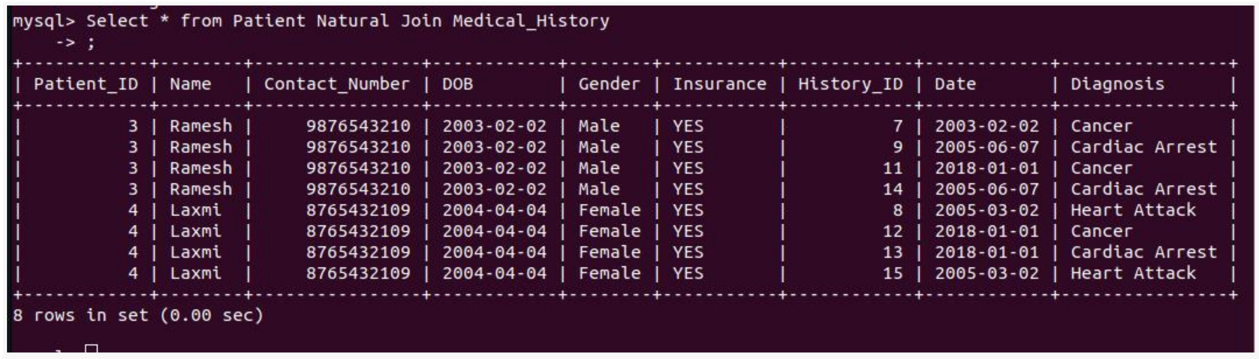


```
|         14 | Dr. Vikas Verma     | 1986-01-12 |     5432109876 |            2
|
|         15 | Dr. Anuja Mehta     | 1991-08-31 |     4321098765 |            1
|
|         16 | Dr. Ramesh Gupta    | 1980-06-05 |     9876543210 |            1
|
+------------+---------------------+------------+----------------+---------------
```

11. View all patients join their medical history:

```
CREATE VIEW All_Patients AS

SELECT * FROM Patient NATURAL JOIN Medical_History;
```

This query creates a view called "All_Patients" that selects all the columns from the "Patient" table and joins it with the "Medical_History" table on the "Patient_ID" field.

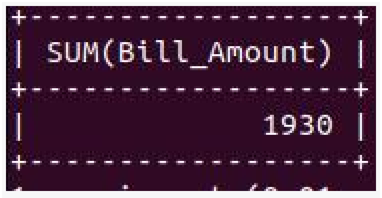


```
mysql> Select * from Patient Natural Join Medical_History
    -> ;
+------------+--------+----------------+------------+--------+-----------+------------+------------+-----------------+
| Patient_ID | Name   | Contact_Number | DOB        | Gender | Insurance | History_ID | Date       | Diagnosis       |
+------------+--------+----------------+------------+--------+-----------+------------+------------+-----------------+
|          3 | Ramesh |     9876543210 | 2003-02-02 | Male   | YES       |          7 | 2003-02-02 | Cancer          |
|          3 | Ramesh |     9876543210 | 2003-02-02 | Male   | YES       |          9 | 2005-06-07 | Cardiac Arrest  |
|          3 | Ramesh |     9876543210 | 2003-02-02 | Male   | YES       |         11 | 2018-01-01 | Cancer          |
|          3 | Ramesh |     9876543210 | 2003-02-02 | Male   | YES       |         14 | 2005-06-07 | Cardiac Arrest  |
|          4 | Laxmi  |     8765432109 | 2004-04-04 | Female | YES       |          8 | 2005-03-02 | Heart Attack    |
|          4 | Laxmi  |     8765432109 | 2004-04-04 | Female | YES       |         12 | 2018-01-01 | Cancer          |
|          4 | Laxmi  |     8765432109 | 2004-04-04 | Female | YES       |         13 | 2018-01-01 | Cardiac Arrest  |
|          4 | Laxmi  |     8765432109 | 2004-04-04 | Female | YES       |         15 | 2005-03-02 | Heart Attack    |
+------------+--------+----------------+------------+--------+-----------+------------+------------+-----------------+
8 rows in set (0.00 sec)
```

12. Total amount earned by a specific doctor:

```
SELECT SUM(Bill_Amount) FROM Billing WHERE Patient_ID=(SELECT Patient_ID FROM
Appointment WHERE Doctor_ID IN(SELECT Doctor_ID FROM DOCTOR WHERE Name='Dr. Ramesh
Gupta'));
```

This query calculates the total amount earned by a specific doctor by selecting the sum of all bill amounts from the "Billing" table where the patient ID matches the patient ID from the appointment with the specified doctor.

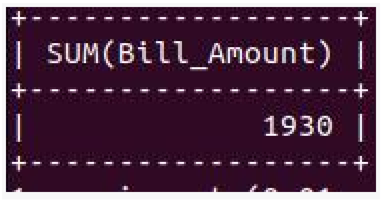


```
+------------------+
| SUM(Bill_Amount) |
+------------------+
|             1930 |
+------------------+
```

13. Total amount earned by a specific specialization:

```
SELECT SUM(Bill_Amount) from Billing where Doctor_ID in(select Doctor_ID FROM DOCTOR
where Department_ID= 1);
```

This query calculates the total amount earned by all doctors with a specific specialization by selecting the sum of all amounts from the "Billing" table where the doctor ID matches with the "DOCTOR" table with the specified department ID.

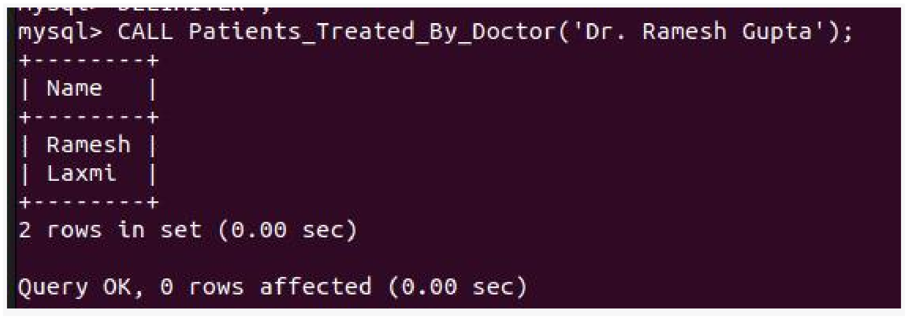


```
+------------------+
| SUM(Bill_Amount) |
+------------------+
|             1930 |
+------------------+
```

14. Function to find all patients a specific doctor has treated:

```
DELIMITER $$

CREATE PROCEDURE Patients_Treated_By_Doctor(IN Doctor_Name VARCHAR(50))

BEGIN

   SELECT Name FROM Patient WHERE Patient_ID IN (SELECT Patient_ID FROM Appointment
WHERE Doctor_ID IN(SELECT Doctor_ID FROM DOCTOR WHERE Name=Doctor_Name));

END $$

DELIMITER ;

CALL Patients_Treated_By_Doctor('Dr. Ramesh Gupta');
```

This procedure takes a doctor name as input and returns a list of all the patients that have been treated

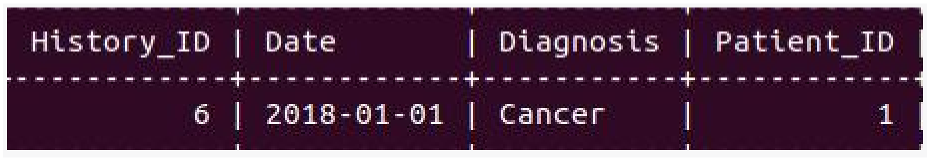


```
mysql> CALL Patients_Treated_By_Doctor('Dr. Ramesh Gupta');
+--------+
| Name   |
+--------+
| Ramesh |
| Laxmi  |
+--------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

15. Function to fetch all previous appointment diagnoses:

```
DELIMITER $$

CREATE PROCEDURE Previous_Appointment_Diagnosis(IN Patien_ID INT)

BEGIN

   SELECT Diagnosis FROM Medical_History WHERE Patient_ID=Patien_ID;

END $$

DELIMITER ;

CALL Previous_Appointment_Diagnosis(1);
```

This stored procedure can be used to easily retrieve the diagnosis of all previous appointments made by a particular patient.



```
 History_ID | Date       | Diagnosis | Patient_ID |
------------+------------+-----------+------------
          6 | 2018-01-01 | Cancer    |          1 |
```

Frontend Documentation

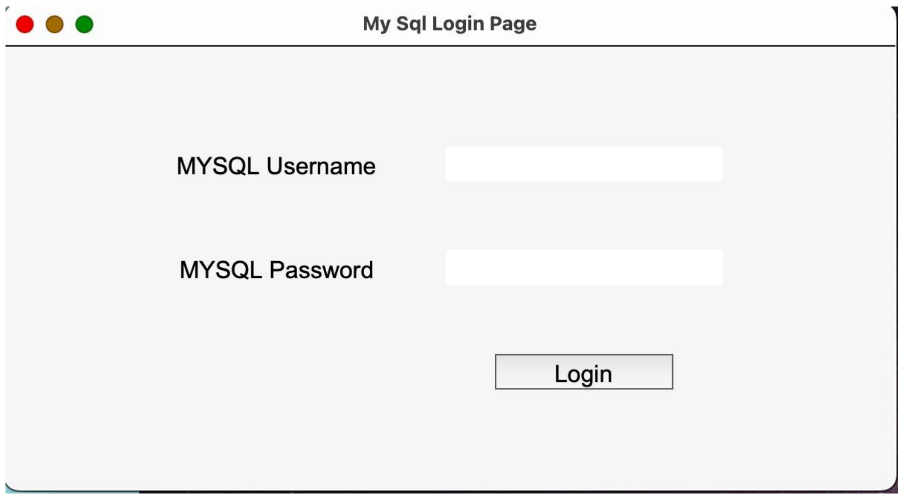
Note:SQL password will be asked for initially when the the interface is run.In case a new database is to be made,makedb.py file needs to be edited and username and password needs to be placed there.

Python is the programming language used for the frontend and sql integration.

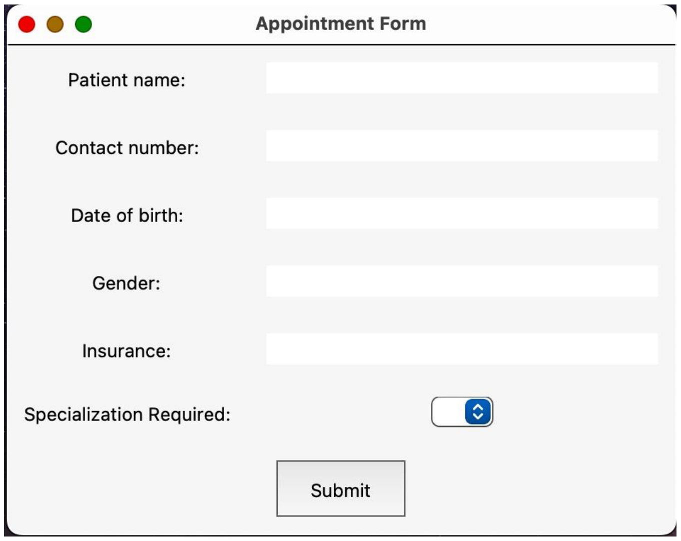
"Tkinter" module in python has been used for frontend.

The interface created for the Hospital Management System is as follows:

- A Login Page has been created in which different users(Hospital Doctors and Administrative Staff) can log into the management system.



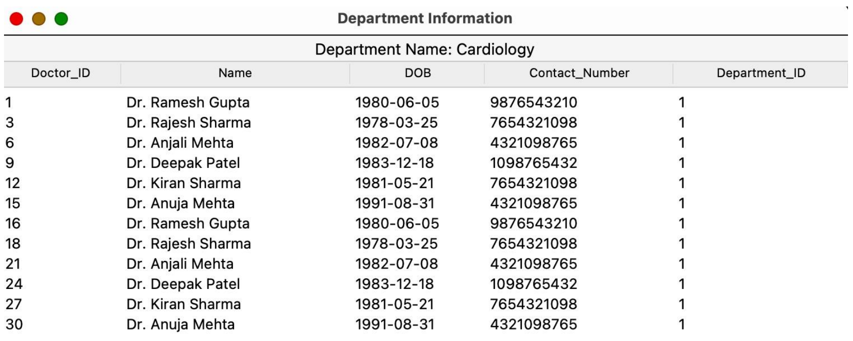
- Once you log in you have many features to choose from.One of the features is to Make an appointment.When you choose that it leads to a page where the patient enters their details and type of doctor(specialization) required and date.


Appointment Form

Patient name:	
Contact number:	
Date of birth:	
Gender:	
Insurance:	
Specialization Required:	

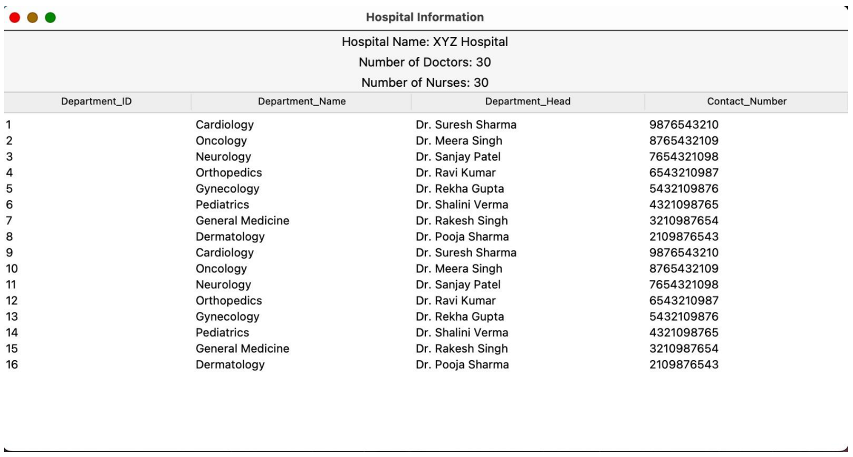
Submit

- The hospital details can be checked which shows all the different departments present in the hospital.On selecting each of these departments,We can further see the details of the doctors within each department.


Department Information

Department Name: Cardiology

Doctor_ID	Name	DOB	Contact_Number	Department_ID
1	Dr. Ramesh Gupta	1980-06-05	9876543210	1
3	Dr. Rajesh Sharma	1978-03-25	7654321098	1
6	Dr. Anjali Mehta	1982-07-08	4321098765	1
9	Dr. Deepak Patel	1983-12-18	1098765432	1
12	Dr. Kiran Sharma	1981-05-21	7654321098	1
15	Dr. Anuja Mehta	1991-08-31	4321098765	1
16	Dr. Ramesh Gupta	1980-06-05	9876543210	1
18	Dr. Rajesh Sharma	1978-03-25	7654321098	1
21	Dr. Anjali Mehta	1982-07-08	4321098765	1
24	Dr. Deepak Patel	1983-12-18	1098765432	1
27	Dr. Kiran Sharma	1981-05-21	7654321098	1
30	Dr. Anuja Mehta	1991-08-31	4321098765	1


Hospital Information

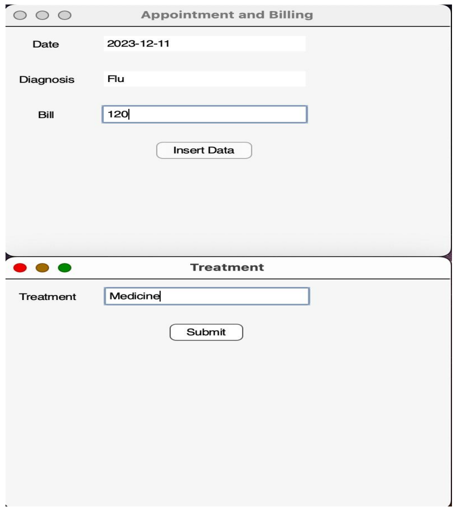
Hospital Name: XYZ Hospital

Number of Doctors: 30

Number of Nurses: 30

Department_ID	Department_Name	Department_Head	Contact_Number
1	Cardiology	Dr. Suresh Sharma	9876543210
2	Oncology	Dr. Meera Singh	8765432109
3	Neurology	Dr. Sanjay Patel	7654321098
4	Orthopedics	Dr. Ravi Kumar	6543210987
5	Gynecology	Dr. Rekha Gupta	5432109876
6	Pediatrics	Dr. Shalini Verma	4321098765
7	General Medicine	Dr. Rakesh Singh	3210987654
8	Dermatology	Dr. Pooja Sharma	2109876543
9	Cardiology	Dr. Suresh Sharma	9876543210
10	Oncology	Dr. Meera Singh	8765432109
11	Neurology	Dr. Sanjay Patel	7654321098
12	Orthopedics	Dr. Ravi Kumar	6543210987
13	Gynecology	Dr. Rekha Gupta	5432109876
14	Pediatrics	Dr. Shalini Verma	4321098765
15	General Medicine	Dr. Rakesh Singh	3210987654
16	Dermatology	Dr. Pooja Sharma	2109876543

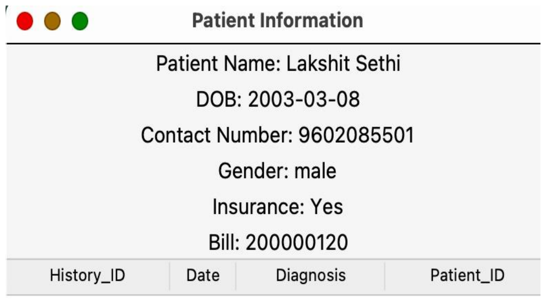
- Diagnosis and treatment can be recorded and Bill payable can be taken by the interface.



Date	2023-12-11
Diagnosis	Flu
Bill	120

Treatment	Medicine

- Patient Information and medical history details can also be seen using the interface.



Patient Name: Lakshit Sethi
DOB: 2003-03-08
Contact Number: 9602085501
Gender: male
Insurance: Yes
Bill: 200000120

History_ID	Date	Diagnosis	Patient_ID

These were some of the examples of the features that have been implemented in the frontend.

Example:Adding Doctor,Adding Patient,Making Appointment,Billing,Adding Medical History,Checking Medical History,Checking Hospital Details,Checking Doctors Details,Checking Doctor Earnings,Checking Department Earnings.

Ample comments have been put in the code in order to make it easier to understand and the interface is fully functional and interfaced with sql.

THANK YOU