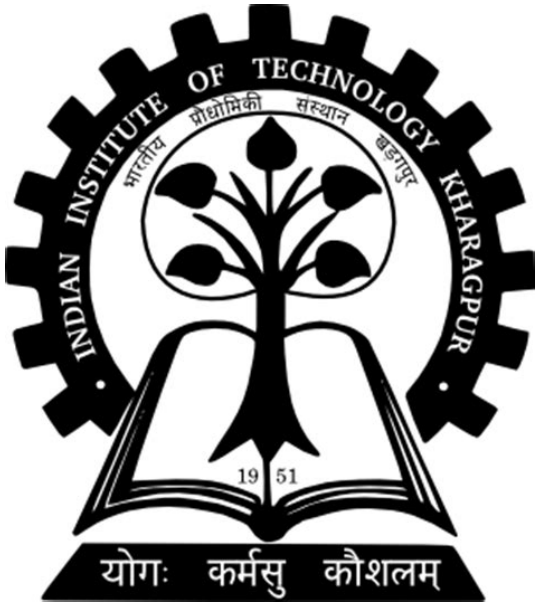
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



LABORATORY ASSIGNMENT 4

(Database Management Systems Laboratory)

Team **React-ers**

Abhay Kumar Keshari 20CS10001

Likhith Reddy Moreddigari 20CS10037

Shivansh Shukla 20CS10057

Venkata Sai Suvvari 20CS10067

Shashank Goud Boorgu 20CS30013

Contents

Chapter 1

Introduction

1.1 Problem Statement

This project is an Interactive Web Application for a Hospital Database Management System. The system registers patients, schedules appointment with doctors, maintains patient information about diagnostics tests and treatments administered, maintains information about doctors'/healthcare professionals, stores admit/discharge information about the patients and manages the database, sends weekly reports to doctors, inform doctor about emergency cases and provide many more rich interactive functionalities.

1.2 Target Users

There are 4 types of Users who use this Application to carry out their respective jobs. The Users are of 4 types:

i. Front Desk Operators

ii. Doctors

iii. Data Entry Operators

iv. Database Administrators

Chapter 2

Specifications

The following are the functionalities that the specified users can do:

2.1 Front Desk Operators

The functionalities that Front Desk Operator can perform are:

a. Register a new patient to the database

b. Give appointment to the patient based on available slots and priorities and the appointment will appear on the corresponding doctor's dashboard

c. Allocate room to the patient and changing the availability status

d. Send **mail** to the doctor when admitting an emergency case

e. Sending the **mail** to each doctor about their respective weekly statistics

f. Discharge patients and changing the availability status

2.2 Data Entry Operator

The functionalities that Data Entry Operator can perform are:

a. Search a patient by name

b. Add a Test to a patient

c. Add a Treatment to a patient

d. Update Result of a Test

e. View prescription of a patient

2.3 Doctor

The functionalities that Doctor can perform are:

a. Get all the appointments

b. Get past tests and results of a particular patient

c. Get the past treatment history of a patient

d. Get all the prescribed medicines of a patient, by all doctors

e. Add a prescription to a patient

2.4 Administrator

a. Can Add new users viz., Front Desk Operators, Data Entry Operators, Doctors

b. Can edit details of users viz., Front Desk Operators, Data Entry Operators, Doctors

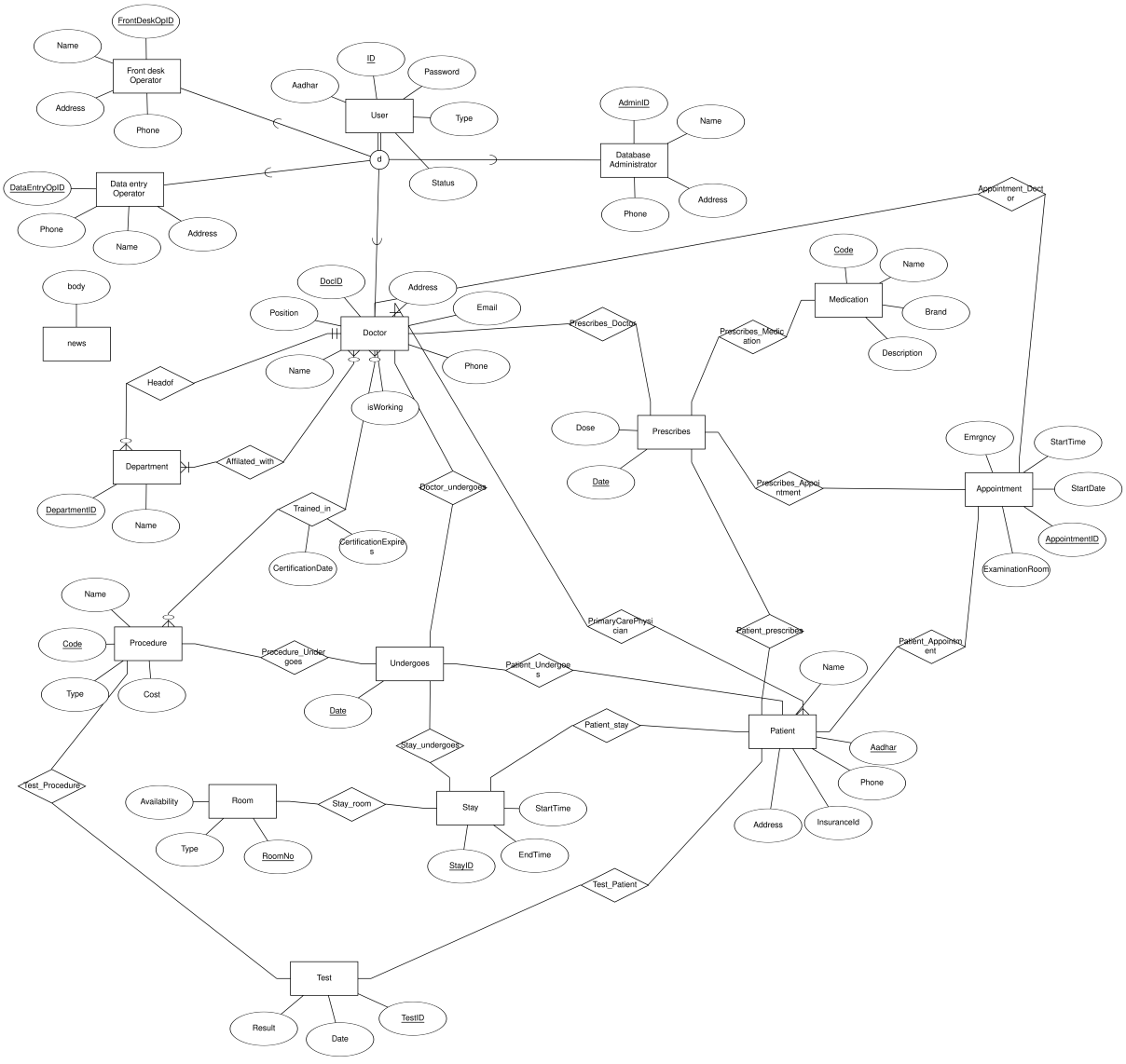
c. Can delete users viz., Front Desk Operators, Data Entry Operators, Doctors

d. Can add news, updates and highlights to the login page

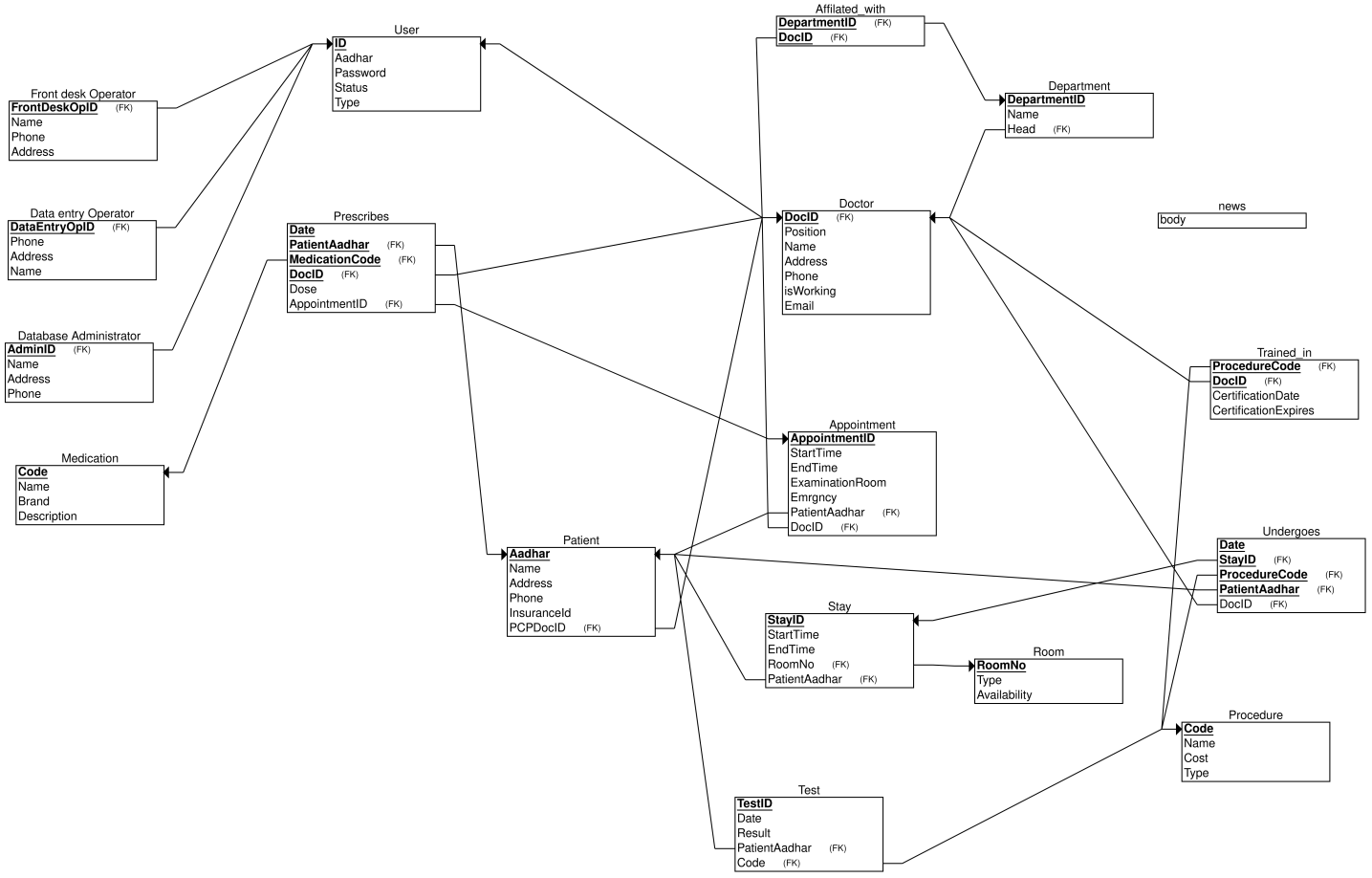
Chapter 3

Database Schema

3.1 ER Diagram



3.2 Relational Table



3.3 Tables

```sql
CREATE TABLE User
(
    ID INTEGER NOT NULL AUTO_INCREMENT,
    Aadhar VARCHAR(12) NOT NULL UNIQUE,
    Password VARCHAR(100) NOT NULL,
    Type INT NOT NULL,
    Status BOOLEAN NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Front_desk_operator
(
    FrontDeskOpID INTEGER NOT NULL,
    Name VARCHAR(50) NOT NULL,
```

```
    Phone VARCHAR(20) NOT NULL,
    Address VARCHAR(200) NOT NULL,
    PRIMARY KEY (FrontDeskOpID),
    FOREIGN KEY (FrontDeskOpID) REFERENCES User (ID)
);

CREATE TABLE Data_entry_operator
(
    DataEntryOpID INTEGER NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Phone VARCHAR(20) NOT NULL,
    Address VARCHAR(200) NOT NULL,
    PRIMARY KEY (DataEntryOpID),
    FOREIGN KEY (DataEntryOpID) REFERENCES User (ID)
);

CREATE TABLE Database_administrator
(
    AdminID INTEGER NOT NULL,
    Name VARCHAR(50) NOT NULL,
    Phone VARCHAR(20) NOT NULL,
    Address VARCHAR(200) NOT NULL,
    PRIMARY KEY (AdminID),
    FOREIGN KEY (AdminID) REFERENCES User (ID)
);

CREATE TABLE Medication
(
    Code INTEGER NOT NULL,
    Name VARCHAR(100) NOT NULL,
    Brand VARCHAR(100) NOT NULL,
    Description VARCHAR(200) NOT NULL,
    PRIMARY KEY (Code)
);

CREATE TABLE Doctor
(
    DocID INTEGER ,
    Position VARCHAR(50) ,
    Name VARCHAR(50) ,
    Phone VARCHAR(20) ,
    Address VARCHAR(200) ,
    isWorking BOOLEAN,
    Email VARCHAR(100),
    PRIMARY KEY (DocID),
```

```
    FOREIGN KEY (DocID) REFERENCES User(ID)
);

CREATE TABLE Department
(
    DepartmentID INTEGER NOT NULL,
    Name VARCHAR(50),
    Head INTEGER,
    PRIMARY KEY (DepartmentID),
    FOREIGN KEY (Head) REFERENCES Doctor(DocID)
);

CREATE TABLE Affiliated_with
(
    DepartmentID INTEGER NOT NULL,
    DocID INTEGER NOT NULL,
    PRIMARY KEY (DepartmentID, DocID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),
    FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
);

CREATE TABLE Patient
(
    Aadhar VARCHAR(12) ,
    Name VARCHAR(50),
    Phone VARCHAR(20),
    Address VARCHAR(200),
    InsuranceID INTEGER,
    PCPDocID INTEGER,
    Age INTEGER,
    Gender VARCHAR(10),
    PRIMARY KEY (Aadhar),
    FOREIGN KEY (PCPDocID) REFERENCES Doctor(DocID)
);

CREATE TABLE Appointment
(
    AppointmentID INTEGER NOT NULL AUTO_INCREMENT,
    StartTime TIME,
    StartDate DATE,
    ExaminationRoom VARCHAR(50),
    PatientAadhar VARCHAR(12),
    DocID INTEGER,
    Emrgncy BOOLEAN,
    PRIMARY KEY(AppointmentID),
```

```
    FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar),
    FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
);

CREATE TABLE Prescribes
(
    Date DATETIME,
    PatientAadhar VARCHAR(12),
    MedicationCode INTEGER,
    DocID INTEGER,
    Dose VARCHAR(100),
    AppointmentID INTEGER,
    PRIMARY KEY (Date, PatientAadhar, MedicationCode, DocID),
    FOREIGN KEY (MedicationCode) REFERENCES Medication(Code),
    FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar),
    FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
);

CREATE TABLE Room
(
    RoomNo INTEGER,
    Type VARCHAR(100),
    Availability BOOLEAN,
    PRIMARY KEY (RoomNo)
);

CREATE TABLE Stay
(
    StayID INTEGER NOT NULL AUTO_INCREMENT,
    StartTime DATETIME,
    EndTime DATETIME,
    RoomNo INTEGER,
    PatientAadhar VARCHAR(12),
    PRIMARY KEY (StayID),
    FOREIGN KEY (RoomNo) REFERENCES Room(RoomNo),
    FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar)
);

CREATE TABLE `Procedure`
(
    Code INTEGER,
    Name VARCHAR(100),
    Type INT NOT NULL,
    Cost INTEGER,
    PRIMARY KEY (Code)
```

```
);

CREATE TABLE Test
(
    TestID INTEGER NOT NULL AUTO_INCREMENT,
    Date DATETIME,
    Result VARCHAR(50),
    PatientAadhar VARCHAR(12),
    Code INTEGER,
    PRIMARY KEY(TestID),
    FOREIGN KEY(PatientAadhar) REFERENCES Patient(Aadhar),
    FOREIGN KEY(Code) REFERENCES `Procedure`(Code)
);

CREATE TABLE Undergoes
(
    Date DATETIME,
    StayID INTEGER,
    ProcedureCode INTEGER,
    PatientAadhar VARCHAR(12),
    DocID INTEGER,
    PRIMARY KEY (Date, StayID, ProcedureCode, PatientAadhar),
    FOREIGN KEY (StayID) REFERENCES Stay(StayID),
    FOREIGN KEY (ProcedureCode) REFERENCES `Procedure`(Code),
    FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar),
    FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
);

CREATE TABLE Trained_in
(
    ProcedureCode INTEGER,
    DocID INTEGER,
    CertificationDate DATETIME,
    CertificationExpires DATETIME,
    PRIMARY KEY (ProcedureCode, DocID),
    FOREIGN KEY (ProcedureCode) REFERENCES `Procedure`(Code),
    FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
);

CREATE TABLE TimeSlot
(
    StartDate DATETIME,
    DocID INTEGER,
    EndDate DATETIME,
    AppointmentID INTEGER,
```

```
195     PRIMARY KEY (StartDate, DocID),
196     FOREIGN KEY (DocID) REFERENCES Doctor(DocID),
197     FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
198 );
199
200 Create TABLE news
201 (
202     body TEXT
203 );
```

Chapter 4

Workflows and Triggers

4.1 Doctor

4.1.1 Add Prescriptions

Used a Procedure insertPrescribes, which inserts prescription if that medicine code exists and that appointment exists as below:

```
DELIMITER $$
create procedure insertPrescribes(IN AppointmentIDParam int, IN MedicationCode int, IN Dose varchar(50))
begin
    declare DoctorID int;
    declare PatientAadharVal varchar(12);
    declare MedicationPresent boolean;
    select DocID, PatientAadhar into DoctorID, PatientAadharVal from Appointment where AppointmentID = AppointmentIDParam;
    if DoctorID is null or PatientAadharVal is null
    then
        signal sqlstate  '45000'
        set message_text = 'AppointmentID does not exist';
    end if;
    select count(*) >= 1 into MedicationPresent from Medication where MedicationCode = MedicationCode;
    if MedicationPresent = false
    then
        signal sqlstate  '45000'
        set message_text = 'Medication does not exist';
    end if;
    insert into Prescribes(MedicationCode, Dose, DocID, AppointmentID, PatientAadhar, Date) values (MedicationCode, Dose, DoctorID, AppointmentIDParam, PatientAadharVal, CURRENT_TIMESTAMP);
end;
```

```
$$
DELIMITER ;
```

Used a Trigger to Validate Prescribes, which checks if doctor is prescribing after the starttime of appointment, which in turn calls a Procedure getAppointmentTimeGivenID

```
DELIMITER $$
create procedure getAppointmentTimeGivenID(IN id int, OUT ok boolean)
begin
    select count(*) >= 1 into ok from Appointment where AppointmentID = id and
    (StartDate < curdate() or (StartDate = curdate() and StartTime < curtime()));
end;
$$
DELIMITER ;
```

```
DELIMITER $$
create trigger validateprescribes
before insert on Prescribes
for each row
begin
    if length(new.Dose) <= 0
    then signal sqlstate  '45000'
        set message_text = 'Quantity cannot be negative';
    end if;
    set @ok = false;
    call getAppointmentTimeGivenID(new.AppointmentID, @ok);
    if @ok = false
    then
        signal sqlstate  '45000'
        set message_text = 'Appointment is in the future';
    end if;
end;
$$
DELIMITER ;
```

4.1.2 Get All Patients

```
SELECT Patient.Name as patientname, Patient.Aadhar as patientaadhar,
    Patient.Gender as gender, Patient.Age as age, Appointment.AppointmentID as
    appointmentid, Appointment.StartTime as starttime, Appointment.StartDate as
    startdate, Patient.Phone as phone, Patient.Address as address,
    Appointment.Emrgncy as Emrgncy
FROM Patient
JOIN Appointment ON Patient.Aadhar = Appointment.PatientAadhar
WHERE Appointment.DocID = ${docId}
ORDER BY Appointment.StartDate DESC,Appointment.StartTime DESC;
```

4.1.3 Get a Patient's Details

```
-- Get Tests and Results
SELECT TestID as testid, Name as procedurename, Date as date, Result as result
FROM Test, \`Procedure\`
WHERE Test.PatientAadhar = '${patientId}' AND Procedure.Code = Test.Code;

-- Get Procedures
SELECT Procedure.Name as procedurename, Undergoes.Date as undergoesdate,
  ↪  Doctor.Name as doctorname
FROM Undergoes, \`Procedure\`, Doctor
WHERE Undergoes.PatientAadhar = '${patientId}' AND Procedure.Code =
  ↪  Undergoes.ProcedureCode AND Doctor.DocID = Undergoes.DocID;

-- Get Prescriptions
SELECT Medication.Name as medicationname, Prescribes.Dose as prescribesdose,
  ↪  Doctor.Name as doctorname, Prescribes.Date as prescribesdate
FROM Prescribes, Doctor, Medication
WHERE Prescribes.PatientAadhar = '${patientId}' AND Medication.Code =
  ↪  Prescribes.MedicationCode AND Doctor.DocID = Prescribes.DocID;
```

4.2 Front Desk Operator

4.2.1 Validate Patient

Used a Trigger to Validate Patient when registering i.e., it checks if the length of the entered fields is correct or not.

```
DELIMITER $$
create trigger validatepatient
before insert on Patient
for each row
begin
    if length(new.Aadhar) < 12 or length(new.Aadhar) > 12
    then signal sqlstate  '45000'
        set message_text = 'Aadhar Should be 12 characters';
    end if;
    if length(new.Phone) < 12 or length(new.Phone) > 12
    then signal sqlstate  '45000'
        set message_text = 'Phone Number Should be 10 characters';
    end if;
end;
$$
DELIMITER ;
```

4.2.2 Validate Appointment

Used a Trigger to Validate Appointment which checks if the entered appointment time correct.

```
DELIMITER $$
create trigger validateappointment
before insert on Appointment
for each row
begin
    if new.StartDate < curdate() or (new.StartDate = curdate() and new.StartTime
    ↪ < curtime())
    then signal sqlstate  '45000'
        set message_text = 'Appointment Date cannot be in the past';
    end if;
end;
$$
DELIMITER ;
```

4.2.3 Validate Stay

Used a Trigger to Validate Stay which checks if the time of stay is correct.

```
DELIMITER $$
create trigger validatestay
before insert on Stay
for each row
begin
    if new.StartTime < concat(curdate(), ' ', curtime())
    then signal sqlstate  '45000'
        set message_text = 'Start Time cannot be in the past';
    end if;
end;
$$
DELIMITER ;
```

4.2.4 Functionalities

```
-- Register Patient
INSERT INTO Patient (Aadhar, Name, Address, Phone, InsuranceId, PCPDocID, Age,
    ↪  Gender)
VALUES ('${Aadhar}', '${Name}', '${Address}', '${Phone}', ${InsuranceID},
    ↪  ${PCPDocID}, ${Age}, '${Gender}')

-- Add Appointment
INSERT INTO Appointment (StartTime, StartDate, ExaminationRoom, PatientAadhar,
    ↪  DocID, Emrgncy)
VALUES ('${StartTime}', '${StartDate}', '${ExaminationRoom}', '${PatientAadhar}',
    ↪  ${DocID}, ${emrgncy})

-- Add Stay
-- Add to stay table
INSERT INTO Stay (StartTime, RoomNo, PatientAadhar)
VALUES ('${StartTime}', ${RoomNo}, '${PatientAadhar}')
-- Update Room Status
UPDATE Room
SET Availability = 0
WHERE RoomNo = ${RoomNo}

-- Discharge Patient
-- Update Stay Endtime
UPDATE Stay
INNER JOIN (SELECT StayID FROM Stay WHERE Stay.PatientAadhar = '${id}' ORDER BY
    ↪  StartTime DESC LIMIT 1) AS S2
ON Stay.StayID = S2.StayID
SET EndTime = '${dateTime}';
-- Update Room Status
SELECT RoomNo FROM Stay
```

```
INNER JOIN (SELECT StayID FROM Stay WHERE Stay.PatientAadhar = '${id}' ORDER BY
    ↪  StartTime DESC LIMIT 1) AS S2
ON Stay.StayID = S2.StayID
UPDATE Room
SET Availability = 1
WHERE RoomNo = ${result[0].RoomNo}
```

4.3 Data Entry Operator

4.3.1 Functionalities

```
-- Get all patients details
Select * from Patient

Search a patient by name
Select * from Patient where Name regexp '${req.params.Search}'


-- Update a test result
SELECT * FROM Test WHERE TestID = ${testId};
UPDATE Test SET Result = '${filename}' WHERE TestID = ${testId};

-- Add treatment to a patient
SELECT Code FROM \`Procedure\` WHERE Name = '${Name}';
SELECT StayID FROM Stay WHERE PatientAadhar = ${req.params.patientId} ORDER BY
 ↪  StartTime DESC LIMIT 1;
INSERT INTO Undergoes(Date, StayId, ProcedureCode, PatientAadhar, DocID) VALUES
 ↪  ('${new Date().toISOString().slice(0, 19).replace('T', ' ')}', ${StayID},
 ↪  ${Code}, '${req.params.patientId}', ${DocID});

-- Add test to a patient
SELECT Code FROM \`Procedure\` WHERE Name = '${Name}';
INSERT INTO Test( Date, Result, PatientAadhar, Code) VALUES ( '${date}','Not Yet
 ↪  Available','${req.params.patientId}', ${Code});

-- Get all treatment names
SELECT Name from \`Procedure\` WHERE Type=1

-- Get all test names
SELECT Name from \`Procedure\` WHERE Type=0

-- Get all test names which donot have a result for a patient
SELECT TestID,Name from Test,\`Procedure\` where Test.Code=\`Procedure\`.Code AND
 ↪  Result='NOT YET AVAILABLE' AND PatientAadhar='${req.params.patientID}'

-- Get details of a particular patient
SELECT * FROM Patient WHERE Aadhar = ${req.params.patientId};

-- Get prescription details of a patient
SELECT * FROM Prescribes WHERE PatientAadhar = ${req.params.patientID}

```

4.4 Administrator

4.4.1 Functionalities

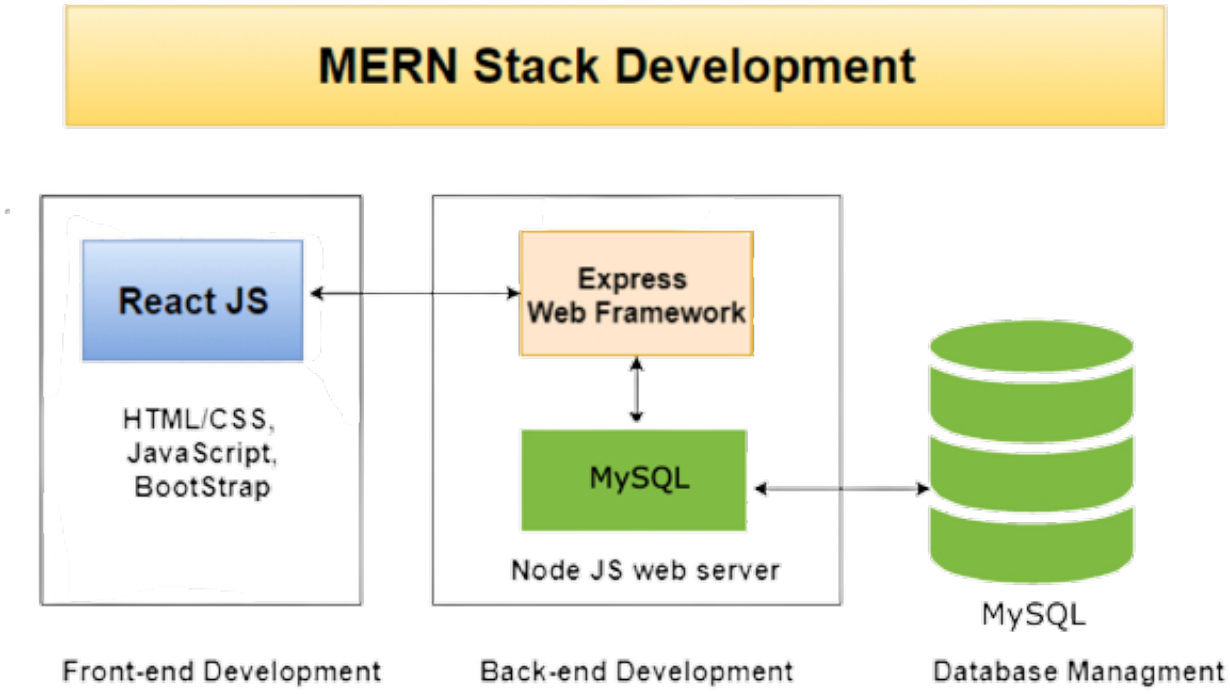
```
-- Add News
INSERT INTO news
VALUES('${news}')

-- Add Admin
INSERT INTO Database_administrator(AdminID, Name, Phone, Address)
VALUES(${req.body.AdminID}, '${req.body.Name}', '${req.body.Phone}',
 ↪  '${req.body.Address}')

-- Update Admin
UPDATE Database_administrator
SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
 ↪  '${req.body.Address}'
WHERE AdminID = ${req.body.AdminID}

-- Add Doctor
INSERT INTO Doctor(DocID, Position, Name, Phone, Address, isWorking, Email)
VALUES(${req.body.DocID}, '${req.body.Position}', '${req.body.Name}',
 ↪  '${req.body.Phone}', '${req.body.Address}', ${req.body.isWorking},
 ↪  '${req.body.Email}')

-- Update Doctor
UPDATE Doctor
SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
 ↪  '${req.body.Address}', Position = '${req.body.Position}', isWorking =
 ↪  ${req.body.isWorking}, Email = '${req.body.Email}'
WHERE DocID = ${req.body.DocID}

-- Delete Doctor
UPDATE User
SET Status=0
WHERE ID = ${req.body.DocID}

-- Add Data Entry Operator
Select *
from Data_entry_operator
WHERE 1=(SELECT Status FROM User WHERE ID=DataEntryOpID)

-- Update Data Entry Operator
UPDATE Data_entry_operator
```

```
35 SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
   ↪  '${req.body.Address}'
36 WHERE DataEntryOpID = ${req.body.DataEntryOpID}
37
38 -- Delete Data Entry Operator
39 UPDATE User
40 SET Status=0
41 WHERE ID = ${req.body.DataEntryOpID}
42
43 -- Add Front Desk Operator
44 INSERT INTO Front_desk_operator(FrontDeskOpID, Name, Phone, Address)
45 VALUES(${req.body.FrontDeskOpID}, '${req.body.Name}', '${req.body.Phone}',
   ↪  '${req.body.Address}')
46
47 -- Update Front Desk Operator
48 UPDATE Front_desk_operator
49 SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
   ↪  '${req.body.Address}'
50 WHERE FrontDeskOpID = ${req.body.FrontDeskOpID}
51
52 -- Delete Front Desk Operator
53 UPDATE User
54 SET Status=0
55 WHERE ID = ${req.body.FrontDeskOpID}
```

Chapter 5

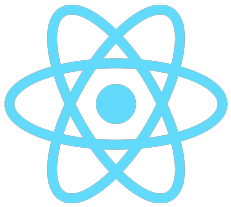
Technologies Used

We have used MERN Stack for this project. Where
M stands for MySQL
E stands for Express
R stands for React
N stands for Node



5.1 Front End

React.js is at the top of the development stack. It is declarative JavaScript framework for creating dynamic client-side applications.



5.2 Server

Express.js is the next step of MERN stack, running on a Node.js server. Express.js is a fast, minimalist web framework for Node.js.

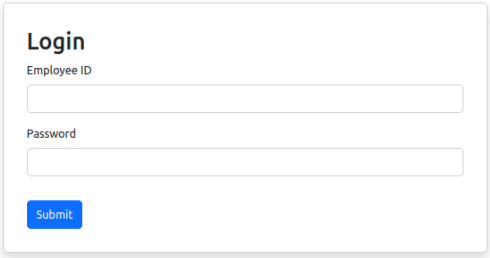
5.3 Database

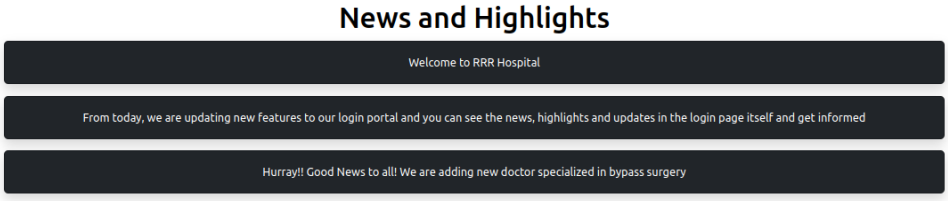
MySQL Database Software is a client/server system that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

Chapter 6

Interface

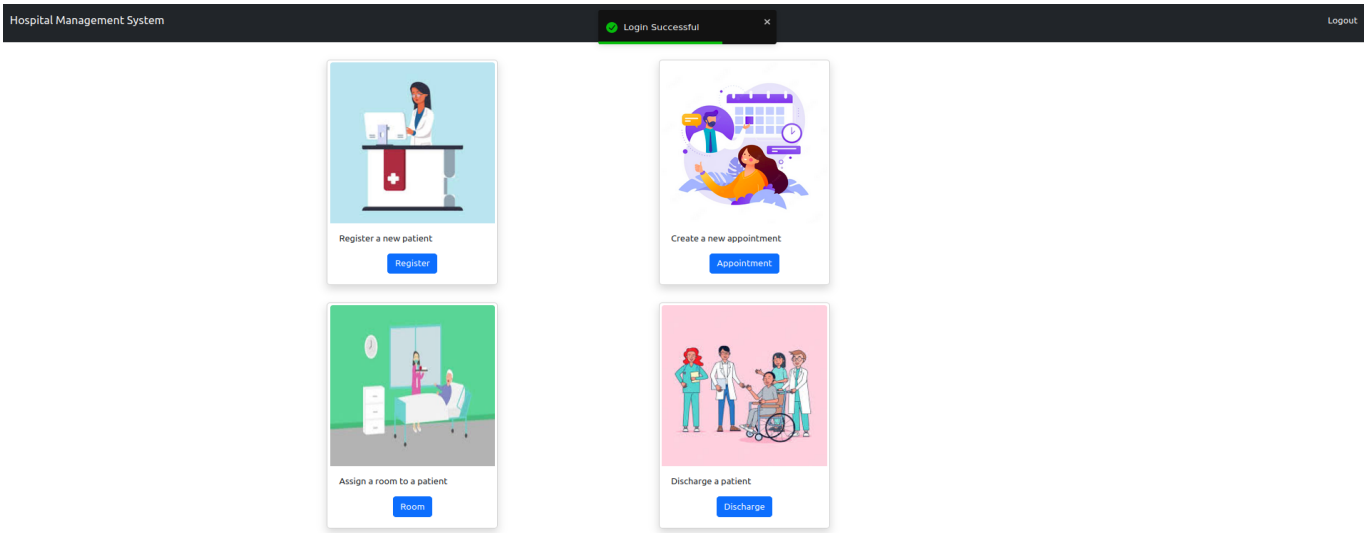
6.1 Login



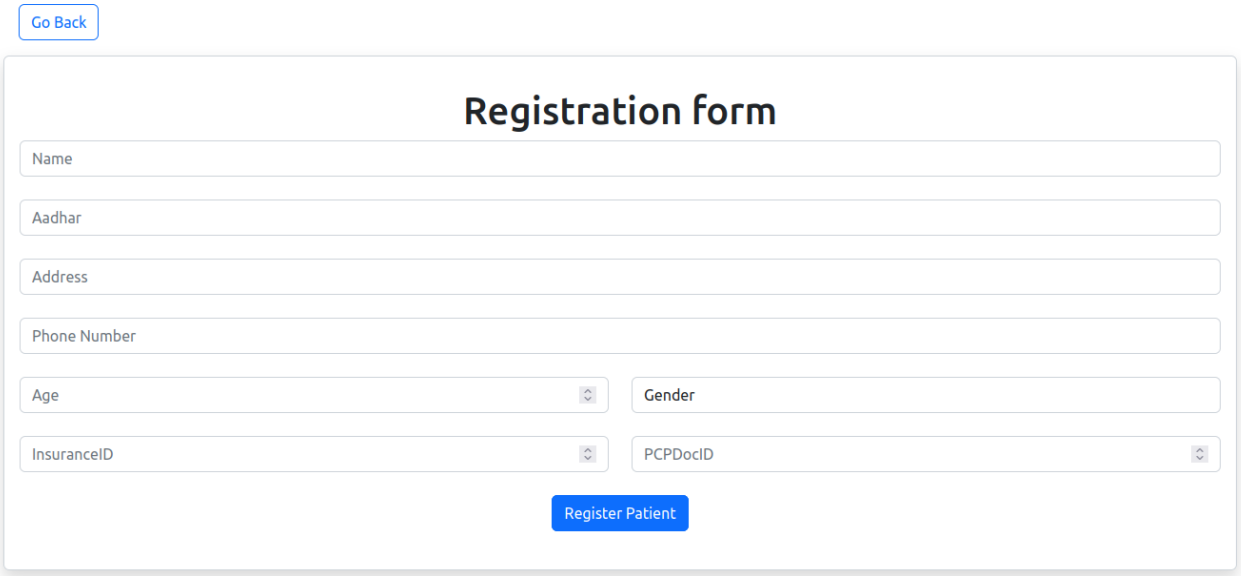


6.2 Front Desk Operator

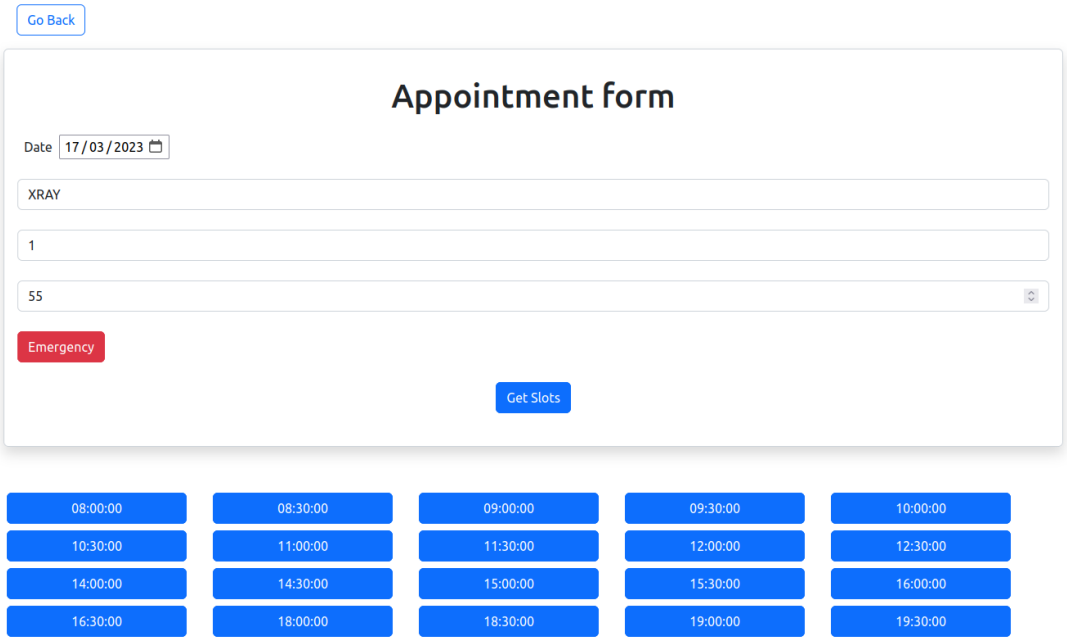
6.2.1 Dashboard



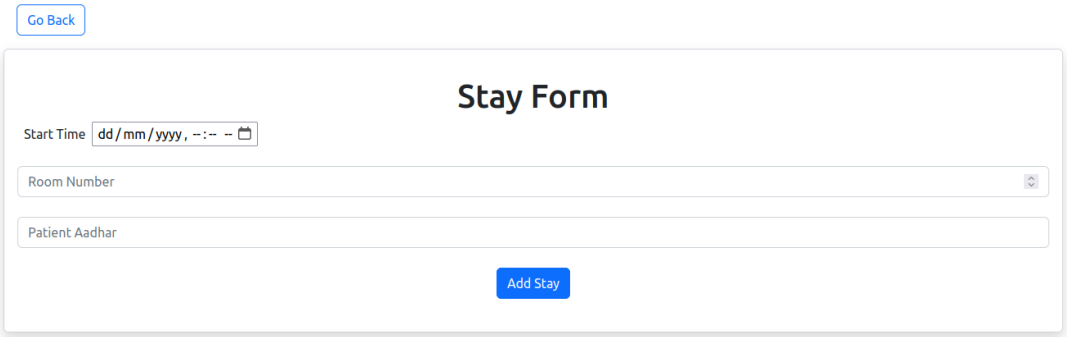
6.2.2 Register Patient



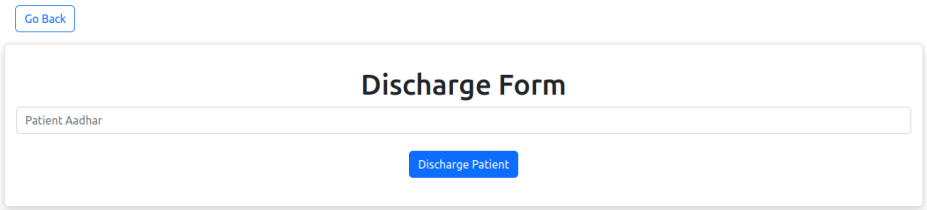
6.2.3 Appointment



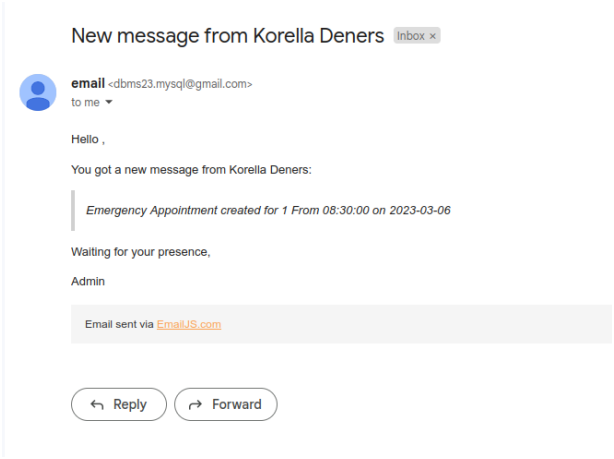
6.2.4 Room Allocation



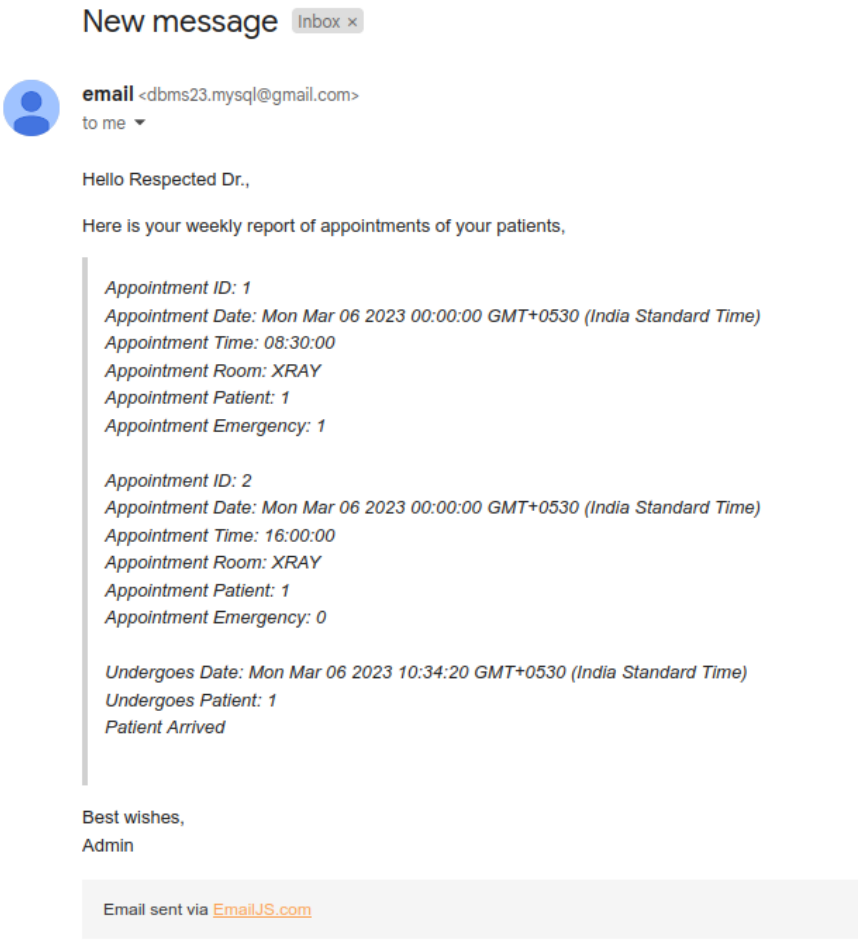
6.2.5 Discharge Patient



6.2.6 Emergency Mail

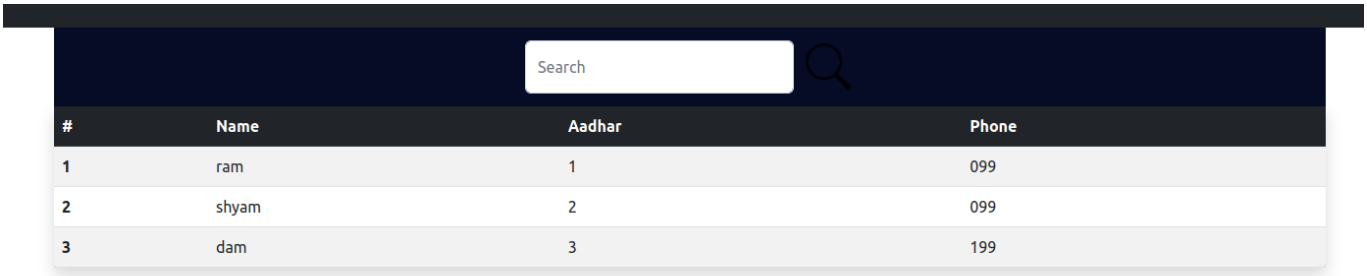


6.2.7 Weekly Report



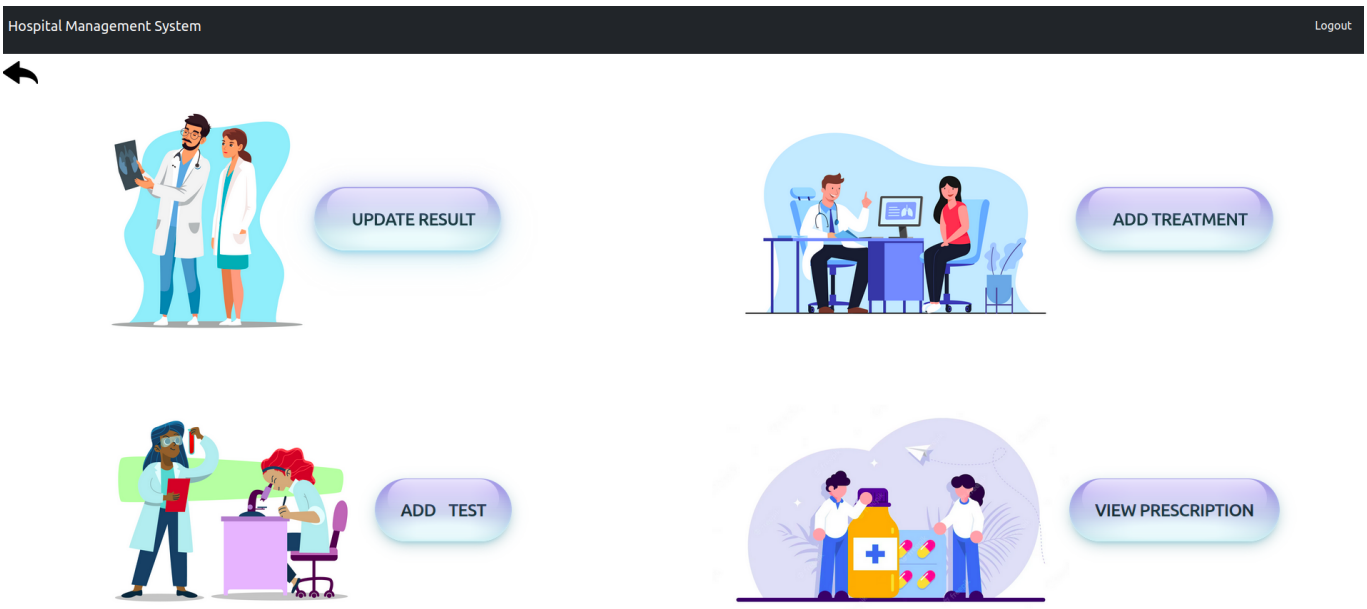
6.3 Data Entry Operator

6.3.1 Dashboard

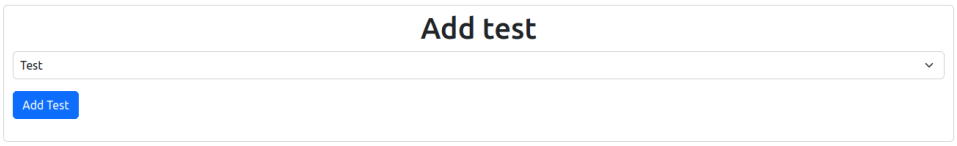


6.3.2 Search a Patient

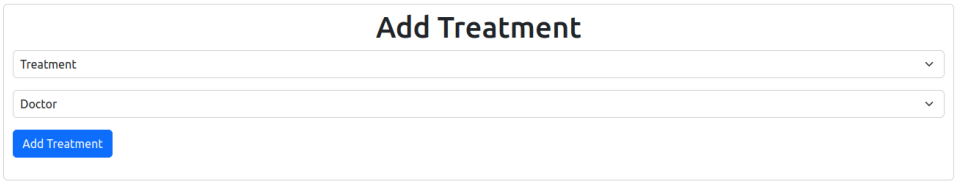
6.3.3 Select an Option



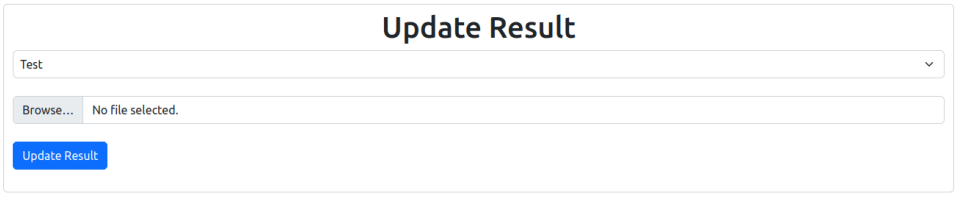
6.3.4 Add Test



6.3.5 Add Treatment

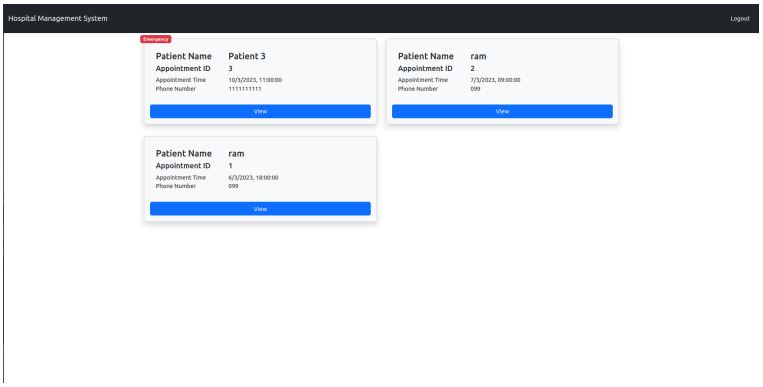


6.3.6 Update Result

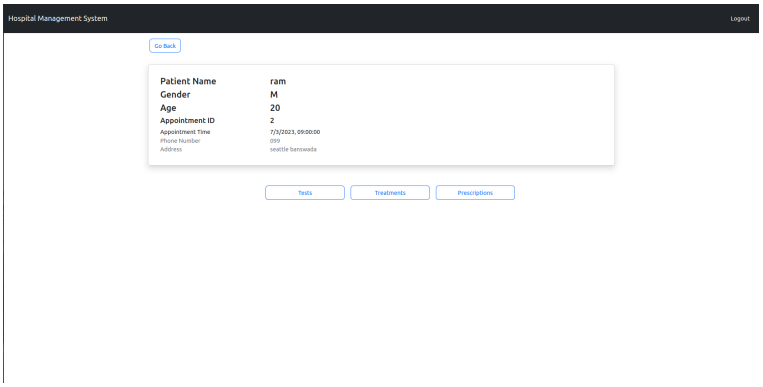


6.4 Doctor

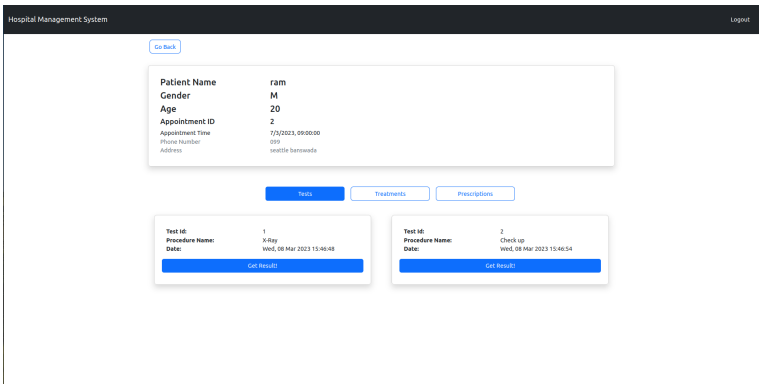
6.4.1 Dashboard



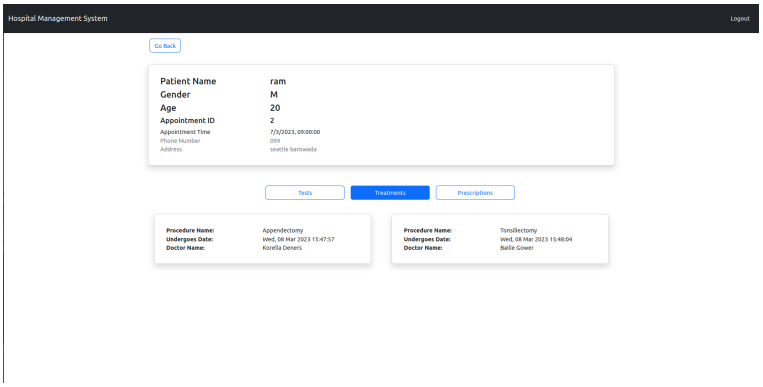
6.4.2 View Patient Details



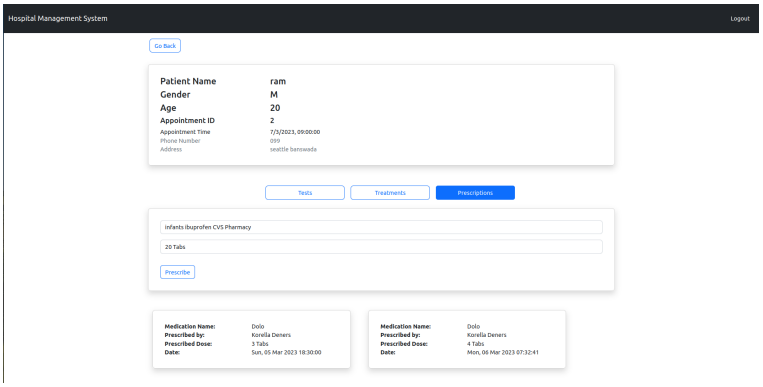
6.4.3 View Test Results



6.4.4 View Treatments

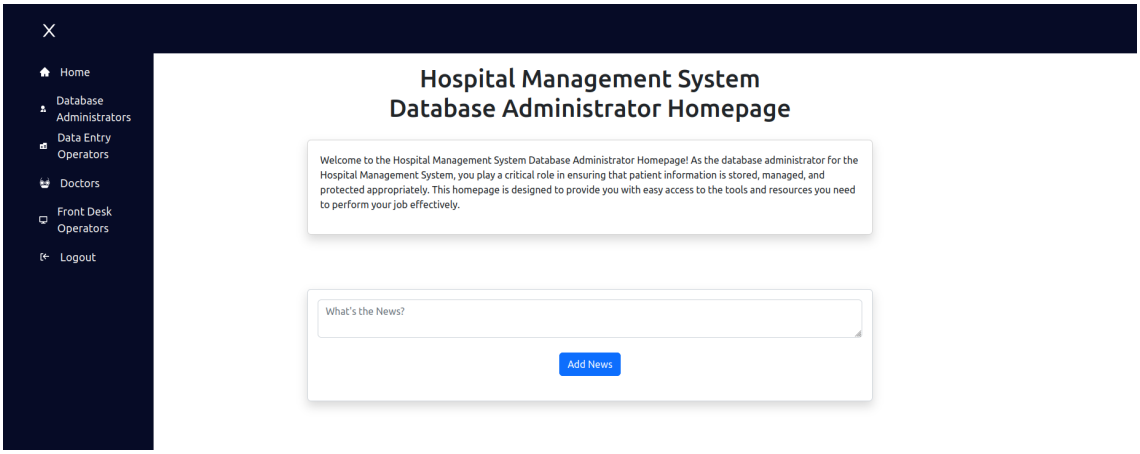


6.4.5 View/Add Prescriptions

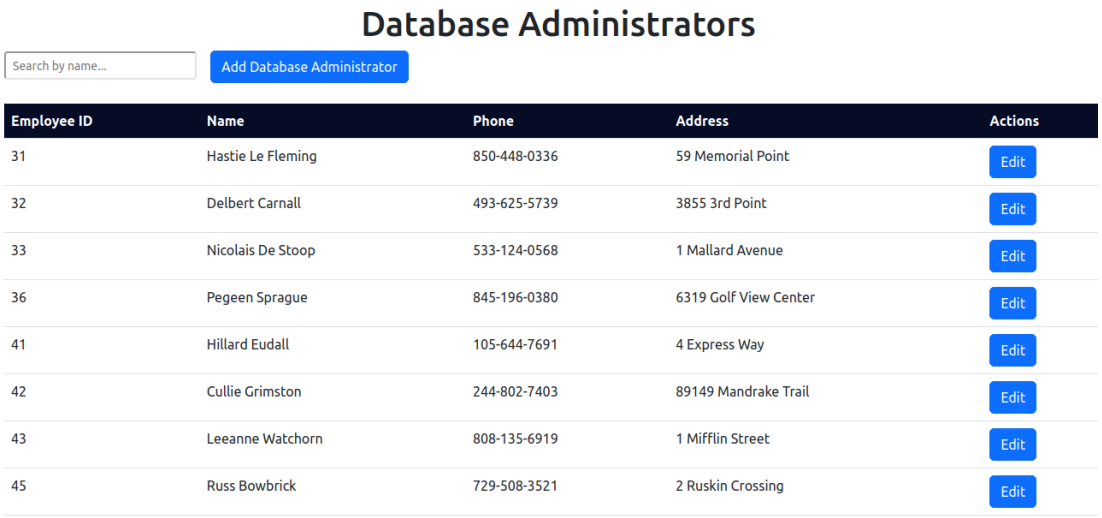


6.5 Administrator

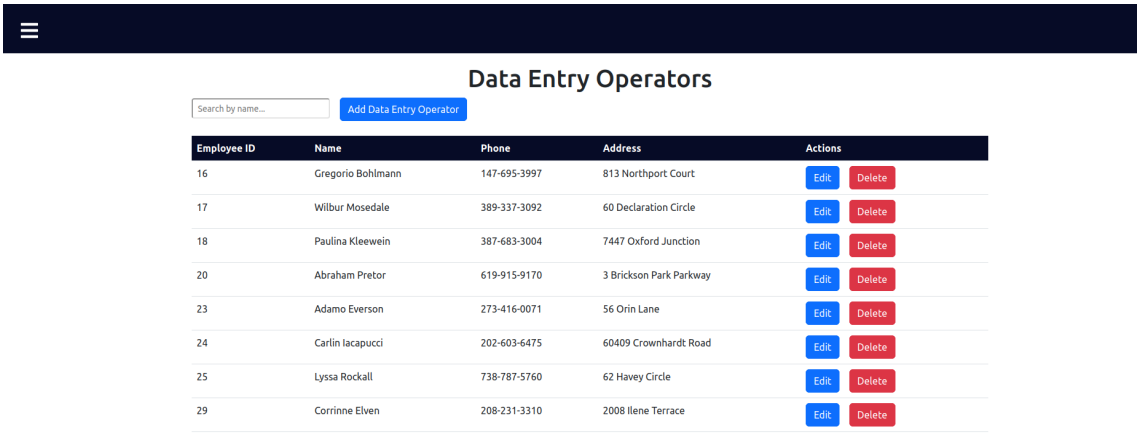
6.5.1 Dashboard



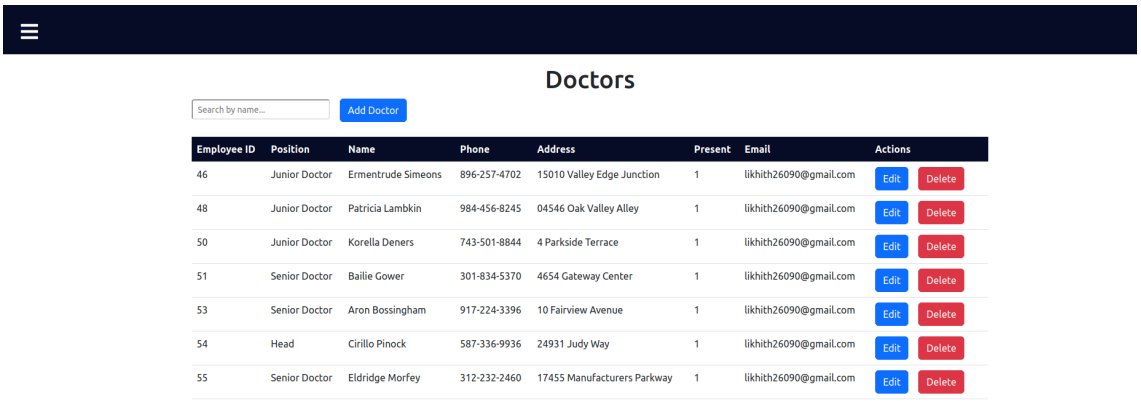
6.5.2 Show Administrators



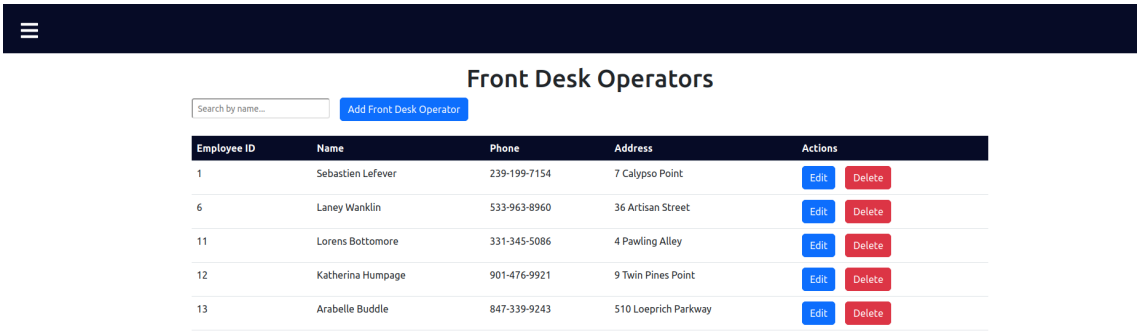
6.5.3 Show Data Entry Operators



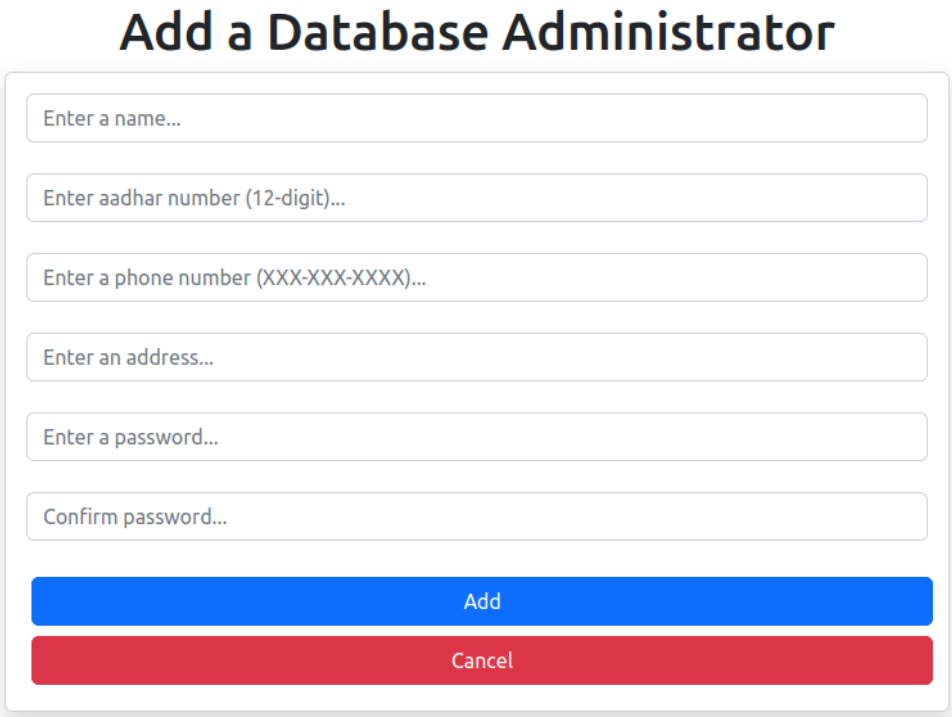
6.5.4 Show Doctors



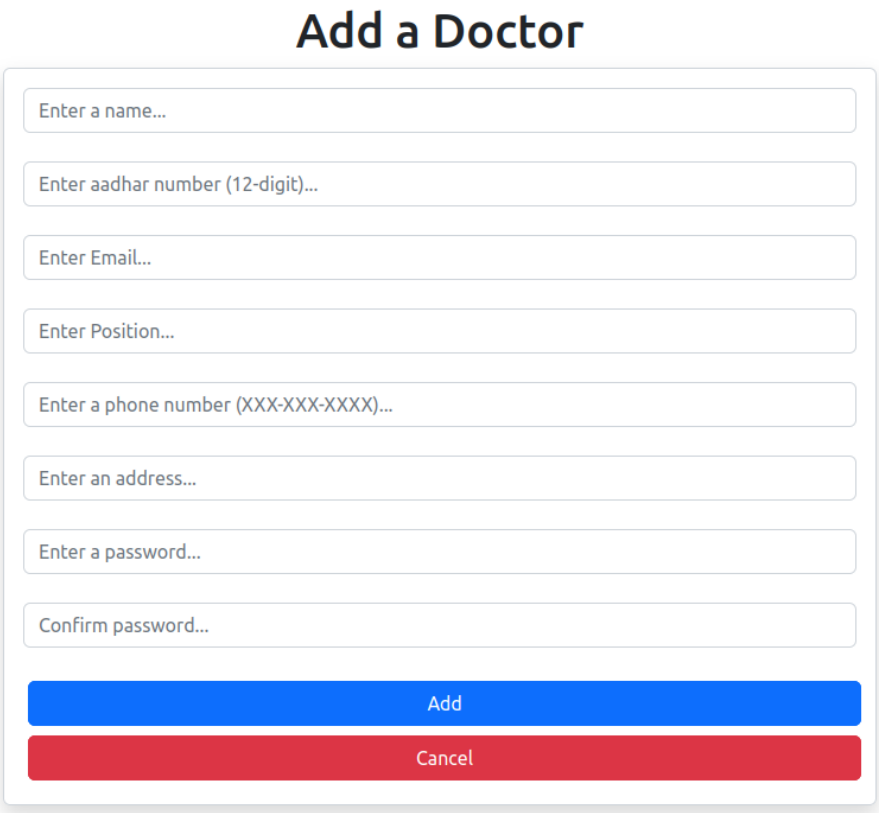
6.5.5 Show Front Desk Operators



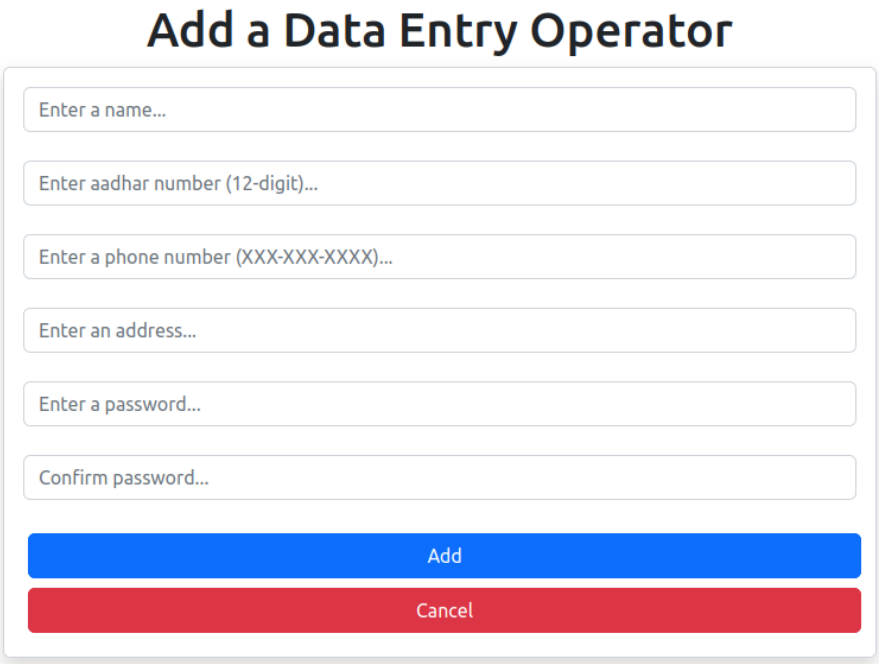
6.5.6 Add Administrator



6.5.7 Add Doctor



6.5.8 Add Data Entry Operator



6.5.9 Add Front Desk Operator