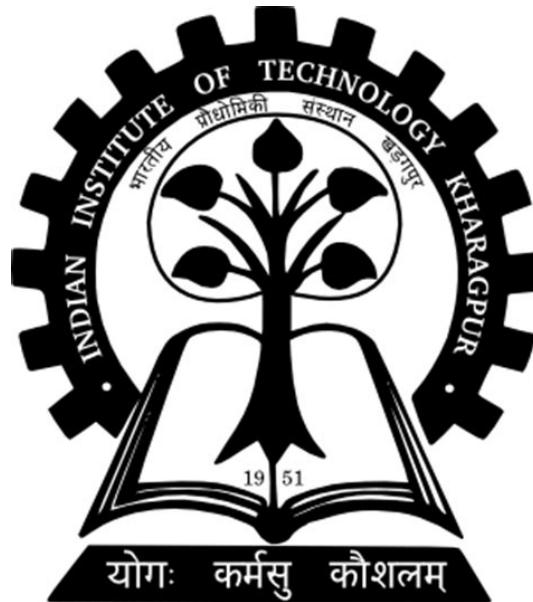


INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

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

1	Introduction	3
1.1	Problem Statement	3
1.2	Target Users	3
2	Specifications	4
2.1	Front Desk Operators	4
2.2	Data Entry Operator	4
2.3	Doctor	5
2.4	Administrator	5
3	Database Schema	6
3.1	ER Diagram	6
3.2	Relational Table	7
3.3	Tables	7
4	Workflows and Triggers	13
4.1	Doctor	13
4.1.1	Add Prescriptions	13
4.1.2	Get All Patients	14
4.1.3	Get a Patient's Details	15
4.2	Front Desk Operator	16
4.2.1	Validate Patient	16
4.2.2	Validate Appointment	16
4.2.3	Validate Stay	16
4.2.4	Functionalities	17
4.3	Data Entry Operator	19
4.3.1	Functionalities	19
4.4	Administrator	20
4.4.1	Functionalities	20
5	Technologies Used	22
5.1	Front End	23
5.2	Server	23
5.3	Database	23

6 Interface	24
6.1 Login	24
6.2 Front Desk Operator	25
6.2.1 Dashboard	25
6.2.2 Register Patient	25
6.2.3 Appointment	26
6.2.4 Room Allocation	26
6.2.5 Discharge Patient	26
6.2.6 Emergency Mail	27
6.2.7 Weekly Report	27
6.3 Data Entry Operator	28
6.3.1 Dashboard	28
6.3.2 Search a Patient	28
6.3.3 Select an Option	28
6.3.4 Add Test	29
6.3.5 Add Treatment	29
6.3.6 Update Result	29
6.4 Doctor	29
6.4.1 Dashboard	29
6.4.2 View Patient Details	30
6.4.3 View Test Results	30
6.4.4 View Treatments	30
6.4.5 View/Add Prescriptions	31
6.5 Administrator	31
6.5.1 Dashboard	31
6.5.2 Show Administrators	32
6.5.3 Show Data Entry Operators	32
6.5.4 Show Doctors	32
6.5.5 Show Front Desk Operators	33
6.5.6 Add Administrator	33
6.5.7 Add Doctor	34
6.5.8 Add Data Entry Operator	35
6.5.9 Add Front Desk Operator	35

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

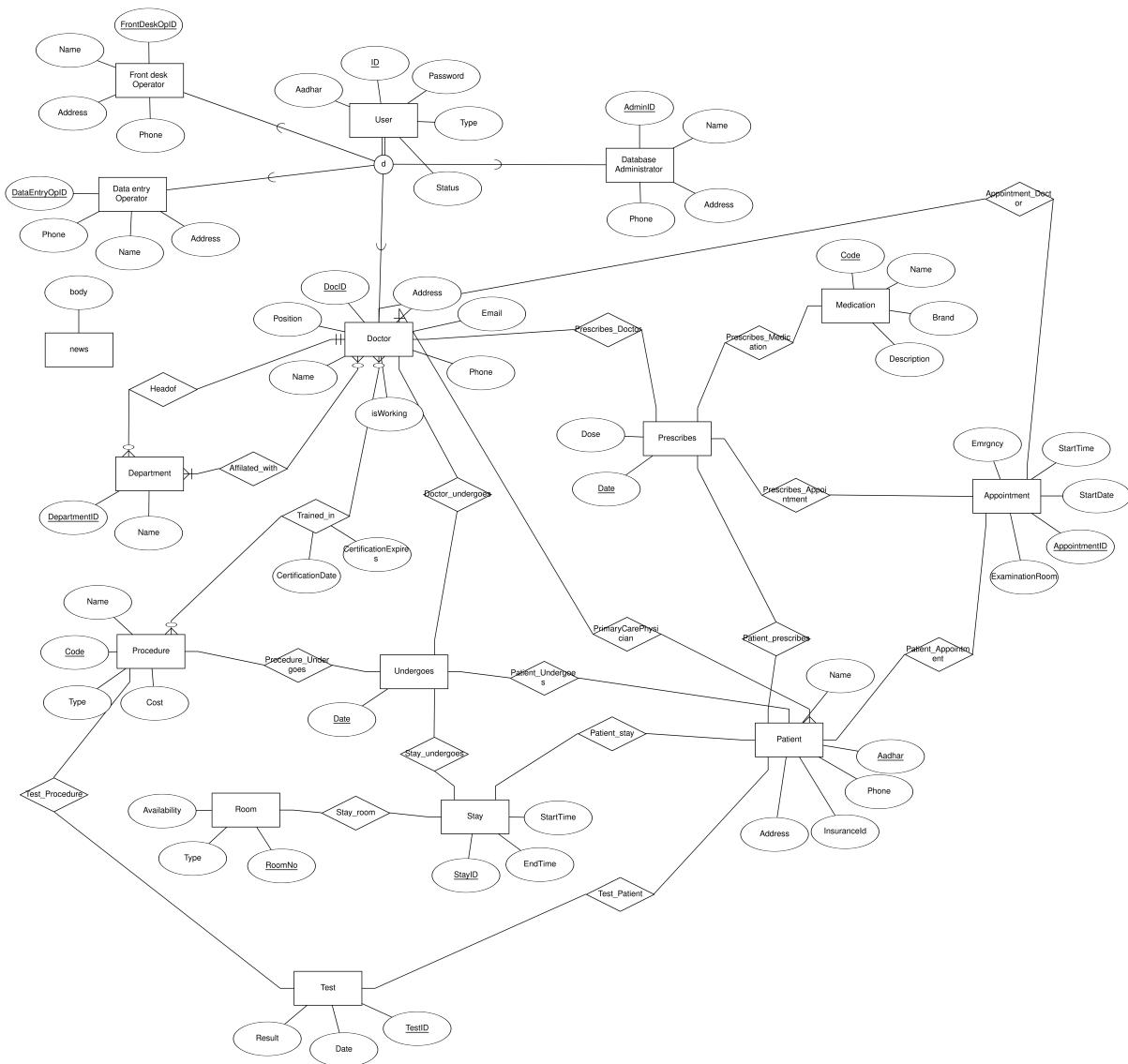
2.4 Administrator

- 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

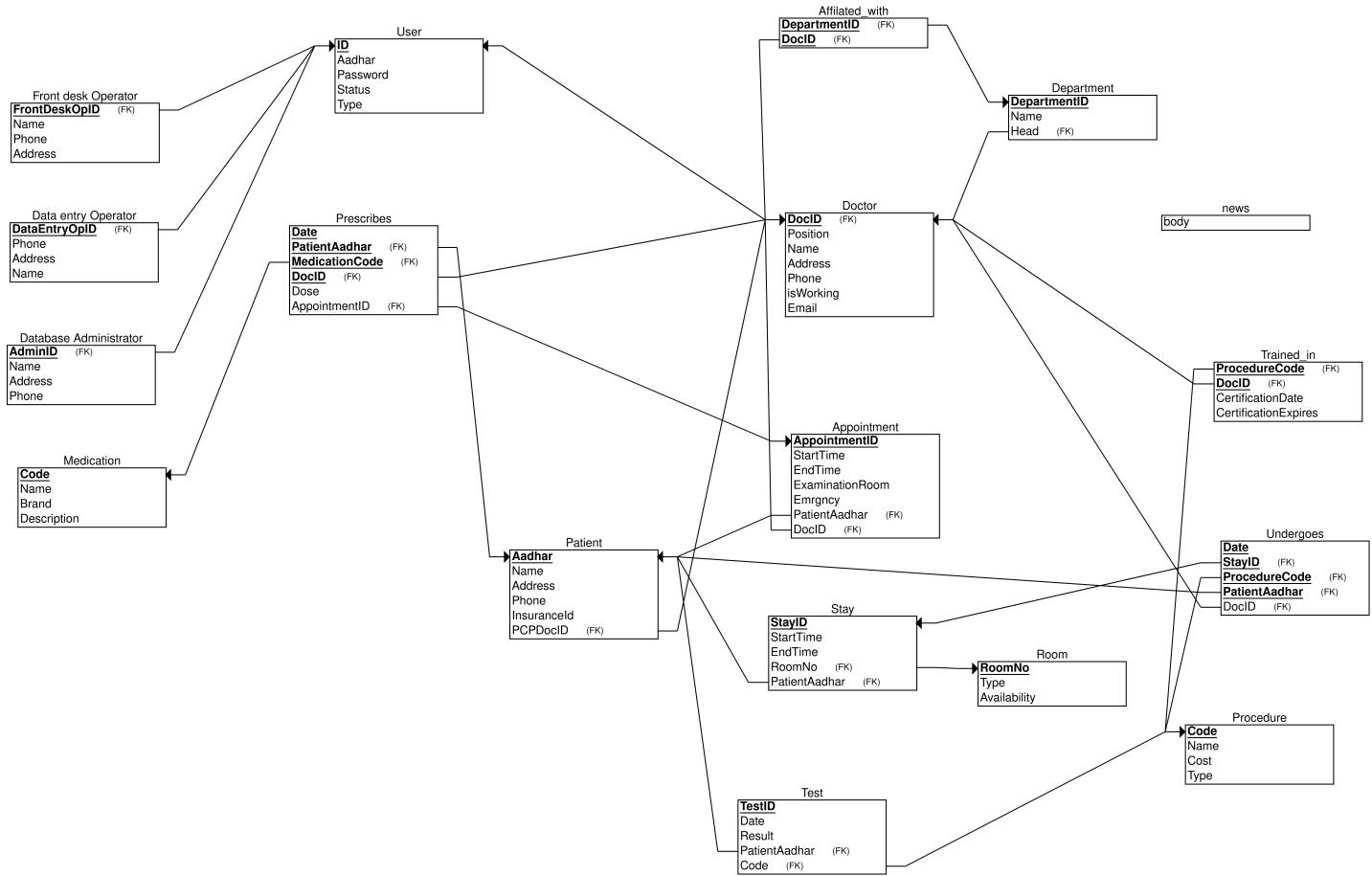
Chapter 3

Database Schema

3.1 ER Diagram



3.2 Relational Table



3.3 Tables

```

1  CREATE TABLE User
2  (
3      ID INTEGER NOT NULL AUTO_INCREMENT,
4      Aadhar VARCHAR(12) NOT NULL UNIQUE,
5      Password VARCHAR(100) NOT NULL,
6      Type INT NOT NULL,
7      Status BOOLEAN NOT NULL,
8      PRIMARY KEY (ID)
9  );
10
11 CREATE TABLE Front_desk_operator
12 (
13     FrontDeskOpID INTEGER NOT NULL,
14     Name VARCHAR(50) NOT NULL,

```

```

15     Phone VARCHAR(20) NOT NULL,
16     Address VARCHAR(200) NOT NULL,
17     PRIMARY KEY (FrontDeskOpID),
18     FOREIGN KEY (FrontDeskOpID) REFERENCES User (ID)
19 );
20
21 CREATE TABLE Data_entry_operator
22 (
23     DataEntryOpID INTEGER NOT NULL,
24     Name VARCHAR(50) NOT NULL,
25     Phone VARCHAR(20) NOT NULL,
26     Address VARCHAR(200) NOT NULL,
27     PRIMARY KEY (DataEntryOpID),
28     FOREIGN KEY (DataEntryOpID) REFERENCES User (ID)
29 );
30
31 CREATE TABLE Database_administrator
32 (
33     AdminID INTEGER NOT NULL,
34     Name VARCHAR(50) NOT NULL,
35     Phone VARCHAR(20) NOT NULL,
36     Address VARCHAR(200) NOT NULL,
37     PRIMARY KEY (AdminID),
38     FOREIGN KEY (AdminID) REFERENCES User (ID)
39 );
40
41 CREATE TABLE Medication
42 (
43     Code INTEGER NOT NULL,
44     Name VARCHAR(100) NOT NULL,
45     Brand VARCHAR(100) NOT NULL,
46     Description VARCHAR(200) NOT NULL,
47     PRIMARY KEY (Code)
48 );
49
50 CREATE TABLE Doctor
51 (
52     DocID INTEGER ,
53     Position VARCHAR(50) ,
54     Name VARCHAR(50) ,
55     Phone VARCHAR(20) ,
56     Address VARCHAR(200) ,
57     isWorking BOOLEAN,
58     Email VARCHAR(100),
59     PRIMARY KEY (DocID),

```

```

60      FOREIGN KEY (DocID) REFERENCES User(ID)
61  );
62
63  CREATE TABLE Department
64  (
65      DepartmentID INTEGER NOT NULL,
66      Name VARCHAR(50),
67      Head INTEGER,
68      PRIMARY KEY (DepartmentID),
69      FOREIGN KEY (Head) REFERENCES Doctor(DocID)
70  );
71
72  CREATE TABLE Affiliated_with
73  (
74      DepartmentID INTEGER NOT NULL,
75      DocID INTEGER NOT NULL,
76      PRIMARY KEY (DepartmentID, DocID),
77      FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID),
78      FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
79  );
80
81  CREATE TABLE Patient
82  (
83      Aadhar VARCHAR(12) ,
84      Name VARCHAR(50),
85      Phone VARCHAR(20),
86      Address VARCHAR(200),
87      InsuranceID INTEGER,
88      PCPDocID INTEGER,
89      Age INTEGER,
90      Gender VARCHAR(10),
91      PRIMARY KEY (Aadhar),
92      FOREIGN KEY (PCPDocID) REFERENCES Doctor(DocID)
93  );
94
95  CREATE TABLE Appointment
96  (
97      AppointmentID INTEGER NOT NULL AUTO_INCREMENT,
98      StartTime TIME,
99      StartDate DATE,
100     ExaminationRoom VARCHAR(50),
101     PatientAadhar VARCHAR(12),
102     DocID INTEGER,
103     Emrgnicy BOOLEAN,
104     PRIMARY KEY(AppointmentID),

```

```

105    FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar) ,
106    FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
107 );
108
109 CREATE TABLE Prescribes
110 (
111     Date DATETIME,
112     PatientAadhar VARCHAR(12),
113     MedicationCode INTEGER,
114     DocID INTEGER,
115     Dose VARCHAR(100),
116     AppointmentID INTEGER,
117     PRIMARY KEY (Date, PatientAadhar, MedicationCode, DocID),
118     FOREIGN KEY (MedicationCode) REFERENCES Medication(Code),
119     FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar),
120     FOREIGN KEY (AppointmentID) REFERENCES Appointment(AppointmentID)
121 );
122
123 CREATE TABLE Room
124 (
125     RoomNo INTEGER,
126     Type VARCHAR(100),
127     Availability BOOLEAN,
128     PRIMARY KEY (RoomNo)
129 );
130
131 CREATE TABLE Stay
132 (
133     StayID INTEGER NOT NULL AUTO_INCREMENT,
134     StartTime DATETIME,
135     EndTime DATETIME,
136     RoomNo INTEGER,
137     PatientAadhar VARCHAR(12),
138     PRIMARY KEY (StayID),
139     FOREIGN KEY (RoomNo) REFERENCES Room(RoomNo),
140     FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar)
141 );
142
143 CREATE TABLE `Procedure`
144 (
145     Code INTEGER,
146     Name VARCHAR(100),
147     Type INT NOT NULL,
148     Cost INTEGER,
149     PRIMARY KEY (Code)

```

```

150 );
151
152 CREATE TABLE Test
153 (
154     TestID INTEGER NOT NULL AUTO_INCREMENT,
155     Date DATETIME,
156     Result VARCHAR(50),
157     PatientAadhar VARCHAR(12),
158     Code INTEGER,
159     PRIMARY KEY(TestID),
160     FOREIGN KEY(PatientAadhar) REFERENCES Patient(Aadhar),
161     FOREIGN KEY(Code) REFERENCES `Procedure`(Code)
162 );
163
164 CREATE TABLE Undergoes
165 (
166     Date DATETIME,
167     StayID INTEGER,
168     ProcedureCode INTEGER,
169     PatientAadhar VARCHAR(12),
170     DocID INTEGER,
171     PRIMARY KEY (Date, StayID, ProcedureCode, PatientAadhar),
172     FOREIGN KEY (StayID) REFERENCES Stay(StayID),
173     FOREIGN KEY (ProcedureCode) REFERENCES `Procedure`(Code),
174     FOREIGN KEY (PatientAadhar) REFERENCES Patient(Aadhar),
175     FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
176 );
177
178 CREATE TABLE Trained_in
179 (
180     ProcedureCode INTEGER,
181     DocID INTEGER,
182     CertificationDate DATETIME,
183     CertificationExpires DATETIME,
184     PRIMARY KEY (ProcedureCode, DocID),
185     FOREIGN KEY (ProcedureCode) REFERENCES `Procedure`(Code),
186     FOREIGN KEY (DocID) REFERENCES Doctor(DocID)
187 );
188
189 CREATE TABLE TimeSlot
190 (
191     StartDate DATETIME,
192     DocID INTEGER,
193     EndDate DATETIME,
194     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:

```
1  DELIMITER $$  
2  create procedure insertPrescribes(IN AppointmentIDParam int, IN MedicationCode  
→  int, IN Dose varchar(50))  
3  begin  
4      declare DoctorID int;  
5      declare PatientAadharVal varchar(12);  
6      declare MedicationPresent boolean;  
7      select DocID, PatientAadhar into DoctorID, PatientAadharVal from Appointment  
→  where AppointmentID = AppointmentIDParam;  
8      if DoctorID is null or PatientAadharVal is null  
9      then  
10          signal sqlstate '45000'  
11          set message_text = 'AppointmentID does not exist';  
12      end if;  
13      select count(*) >= 1 into MedicationPresent from Medication where  
→  MedicationCode = MedicationCode;  
14      if MedicationPresent = false  
15      then  
16          signal sqlstate '45000'  
17          set message_text = 'Medication does not exist';  
18      end if;  
19      insert into Prescribes(MedicationCode, Dose, DocID, AppointmentID,  
→  PatientAadhar, Date) values (MedicationCode, Dose, DoctorID,  
→  AppointmentIDParam, PatientAadharVal, CURRENT_TIMESTAMP);  
20  end;
```

```

21  $$  

22  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`

```

1  DELIMITER $$  

2  create procedure getAppointmentTimeGivenID(IN id int, OUT ok boolean)  

3  begin  

4      select count(*) >= 1 into ok from Appointment where AppointmentID = id and  

    ↳ (StartDate < curdate() or (StartDate = curdate() and StartTime < curtime()));  

5  end;  

6  $$  

7  DELIMITER ;

```



```

1  DELIMITER $$  

2  create trigger validateprescribes  

3  before insert on Prescribes  

4  for each row  

5  begin  

6      if length(new.Dose) <= 0  

7          then signal sqlstate '45000'  

8              set message_text = 'Quantity cannot be negative';  

9      end if;  

10     set @ok = false;  

11     call getAppointmentTimeGivenID(new.AppointmentID, @ok);  

12     if @ok = false  

13         then  

14             signal sqlstate '45000'  

15             set message_text = 'Appointment is in the future';  

16         end if;  

17     end;  

18  $$  

19  DELIMITER ;

```

4.1.2 Get All Patients

```

1  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.Emrgnyc as Emrgnyc  

2  FROM Patient  

3  JOIN Appointment ON Patient.Aadhar = Appointment.PatientAadhar  

4  WHERE Appointment.DocID = ${docId}  

5  ORDER BY Appointment.StartDate DESC, Appointment.StartTime DESC;

```

4.1.3 Get a Patient's Details

```
1  -- Get Tests and Results
2  SELECT TestID as testid, Name as procedurename, Date as date, Result as result
3  FROM Test, Procedure
4  WHERE Test.PatientAadhar = '${patientId}' AND Procedure.Code = Test.Code;
5
6  -- Get Procedures
7  SELECT Procedure.Name as procedurename, Undergoes.Date as undergoesdate,
8  ↳ Doctor.Name as doctorname
9  FROM Undergoes, Procedure, Doctor
10 WHERE Undergoes.PatientAadhar = '${patientId}' AND Procedure.Code =
11   ↳ Undergoes.ProcedureCode AND Doctor.DocID = Undergoes.DocID;
12
13 -- Get Prescriptions
14 SELECT Medication.Name as medicationname, Prescribes.Dose as prescribesdose,
15   ↳ Doctor.Name as doctorname, Prescribes.Date as prescribesdate
16 FROM Prescribes, Doctor, Medication
17 WHERE Prescribes.PatientAadhar = '${patientId}' AND Medication.Code =
18   ↳ 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.

```
1  DELIMITER $$  
2  create trigger validatepatient  
3  before insert on Patient  
4  for each row  
5  begin  
6      if length(new.Aadhar) < 12 or length(new.Aadhar) > 12  
7      then signal sqlstate '45000'  
8          set message_text = 'Aadhar Should be 12 characters';  
9      end if;  
10     if length(new.Phone) < 10 or length(new.Phone) > 10  
11     then signal sqlstate '45000'  
12         set message_text = 'Phone Number Should be 10 characters';  
13     end if;  
14  end;  
15  $$  
16  DELIMITER ;
```

4.2.2 Validate Appointment

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

```
1  DELIMITER $$  
2  create trigger validateappointment  
3  before insert on Appointment  
4  for each row  
5  begin  
6      if new.StartDate < curdate() or (new.StartDate = curdate() and new.StartTime  
7      < curtime())  
8      then signal sqlstate '45000'  
9          set message_text = 'Appointment Date cannot be in the past';  
10     end if;  
11  end;  
12  $$  
13  DELIMITER ;
```

4.2.3 Validate Stay

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

```

1  DELIMITER $$  

2  create trigger validatestay  

3  before insert on Stay  

4  for each row  

5  begin  

6      if new.StartTime < concat(curdate(), ' ', curtime())  

7          then signal sqlstate '45000'  

8              set message_text = 'Start Time cannot be in the past';  

9          end if;  

10 end;  

11 $$  

12 DELIMITER ;

```

4.2.4 Functionalities

```

1  -- Register Patient  

2  INSERT INTO Patient (Aadhar, Name, Address, Phone, InsuranceId, PCPDocID, Age,  

   ↵  Gender)  

3  VALUES ('${Aadhar}', '${Name}', '${Address}', '${Phone}', ${InsuranceID},  

   ↵  ${PCPDocID}, ${Age}, ${Gender})  

4  

5  -- Add Appointment  

6  INSERT INTO Appointment (StartTime, StartDate, ExaminationRoom, PatientAadhar,  

   ↵  DocID, Emrgncy)  

7  VALUES ('${StartTime}', '${StartDate}', '${ExaminationRoom}', '${PatientAadhar}',  

   ↵  ${DocID}, ${Emrgncy})  

8  

9  -- Add Stay  

10 -- Add to stay table  

11 INSERT INTO Stay (StartTime, RoomNo, PatientAadhar)  

12 VALUES ('${StartTime}', ${RoomNo}, '${PatientAadhar}')  

13 -- Update Room Status  

14 UPDATE Room  

15 SET Availability = 0  

16 WHERE RoomNo = ${RoomNo}  

17  

18 -- Discharge Patient  

19 -- Update Stay Endtime  

20 UPDATE Stay  

21 INNER JOIN (SELECT StayID FROM Stay WHERE Stay.PatientAadhar = '${id}' ORDER BY  

   ↵  StartTime DESC LIMIT 1) AS S2  

22 ON Stay.StayID = S2.StayID  

23 SET EndTime = '${dateTime}';  

24 -- Update Room Status  

25 SELECT RoomNo FROM Stay

```

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

4.3 Data Entry Operator

4.3.1 Functionalities

```
1  -- Get all patients details
2  Select * from Patient
3
4  Search a patient by name
5  Select * from Patient where Name regexp '${req.params.Search}'
6
7
8  -- Update a test result
9  SELECT * FROM Test WHERE TestID = ${testId};
10 UPDATE Test SET Result = '${filename}' WHERE TestID = ${testId};
11
12 -- Add treatment to a patient
13 SELECT Code FROM `Procedure` WHERE Name = '${Name}';
14 SELECT StayID FROM Stay WHERE PatientAadhar = ${req.params.patientId} ORDER BY
15   ↵ StartTime DESC LIMIT 1;
16 INSERT INTO Undergoes(Date, StayId, ProcedureCode, PatientAadhar, DocID) VALUES
17   ↵ ('${new Date().toISOString().slice(0, 19).replace('T', ' ')}', ${StayID},
18   ↵ ${Code}, '${req.params.patientId}', ${DocID});
19
20 -- Add test to a patient
21 SELECT Code FROM `Procedure` WHERE Name = '${Name}';
22 INSERT INTO Test( Date, Result, PatientAadhar, Code) VALUES ( '${date}', 'Not Yet
23   ↵ Available', '${req.params.patientId}', ${Code});
24
25 -- Get all treatment names
26 SELECT Name from `Procedure` WHERE Type=1
27
28 -- Get all test names
29 SELECT Name from `Procedure` WHERE Type=0
30
31 -- Get all test names which do not have a result for a patient
32 SELECT TestID,Name from Test,`Procedure` where Test.Code=`Procedure`.Code AND
33   ↵ Result='NOT YET AVAILABLE' AND PatientAadhar='${req.params.patientID}'
34
35 -- Get details of a particular patient
36 SELECT * FROM Patient WHERE Aadhar = ${req.params.patientId};
37
38 -- Get prescription details of a patient
39 SELECT * FROM Prescribes WHERE PatientAadhar = ${req.params.patientID}
```

4.4 Administrator

4.4.1 Functionalities

```
1  -- Add News
2  INSERT INTO news
3  VALUES('${news}')
4
5  -- Add Admin
6  INSERT INTO Database_administrator/AdminID, Name, Phone, Address)
7  VALUES(${req.body.AdminID}, '${req.body.Name}', '${req.body.Phone}',
8       '${req.body.Address}')
9
10 -- Update Admin
11 UPDATE Database_administrator
12 SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
13      '${req.body.Address}'
14 WHERE AdminID = ${req.body.AdminID}
15
16 -- Add Doctor
17 INSERT INTO Doctor(DocID, Position, Name, Phone, Address, isWorking, Email)
18 VALUES(${req.body.DocID}, '${req.body.Position}', '${req.body.Name}',
19        '${req.body.Phone}', '${req.body.Address}', ${req.body.isWorking},
20        '${req.body.Email}')
21
22 -- Update Doctor
23 UPDATE Doctor
24 SET Name = '${req.body.Name}', Phone = '${req.body.Phone}', Address =
25      '${req.body.Address}', Position = '${req.body.Position}', isWorking =
26      ${req.body.isWorking}, Email = '${req.body.Email}'
27 WHERE DocID = ${req.body.DocID}
28
29 -- Delete Doctor
30 UPDATE User
31 SET Status=0
32 WHERE ID = ${req.body.DocID}
33
34 -- Add Data Entry Operator
35 Select *
36 from Data_entry_operator
37 WHERE 1=(SELECT Status FROM User WHERE ID=DataEntryOpID)
38
39 -- Update Data Entry Operator
40 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

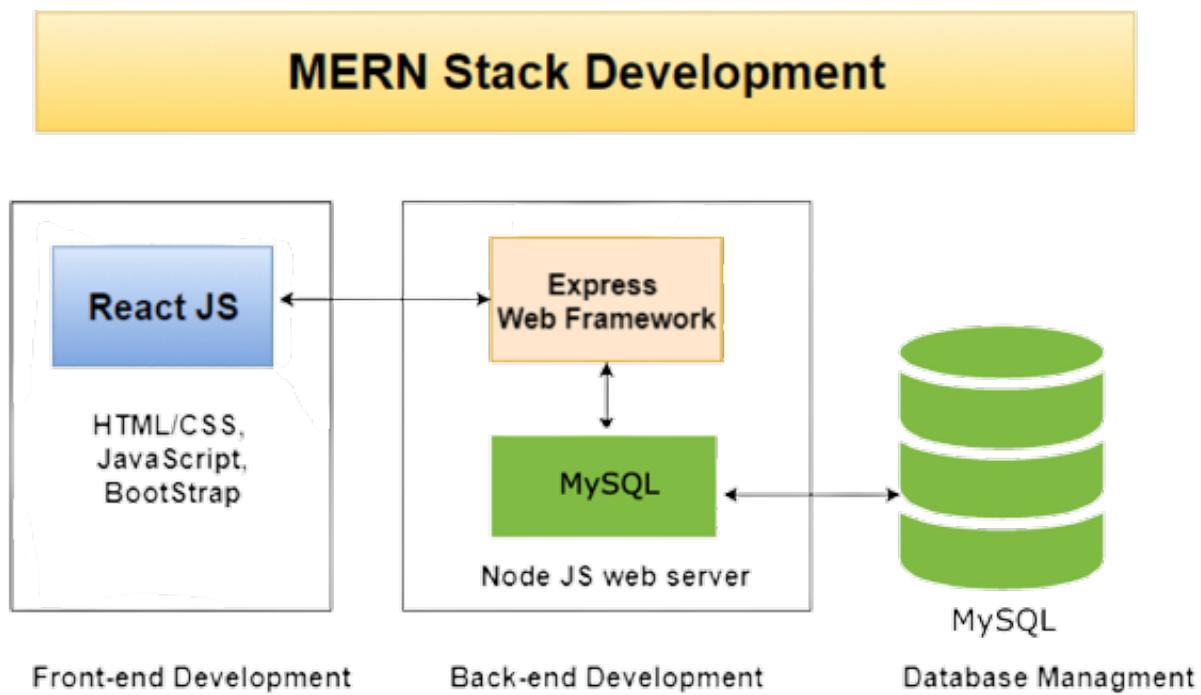
We have used MERN Stack for this project. Where

M stands for MySQL

E stands for Express

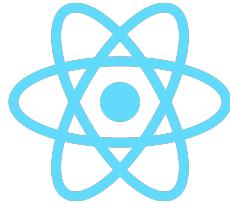
R stands for React

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

Login

Employee ID

Password

Submit

News and Highlights

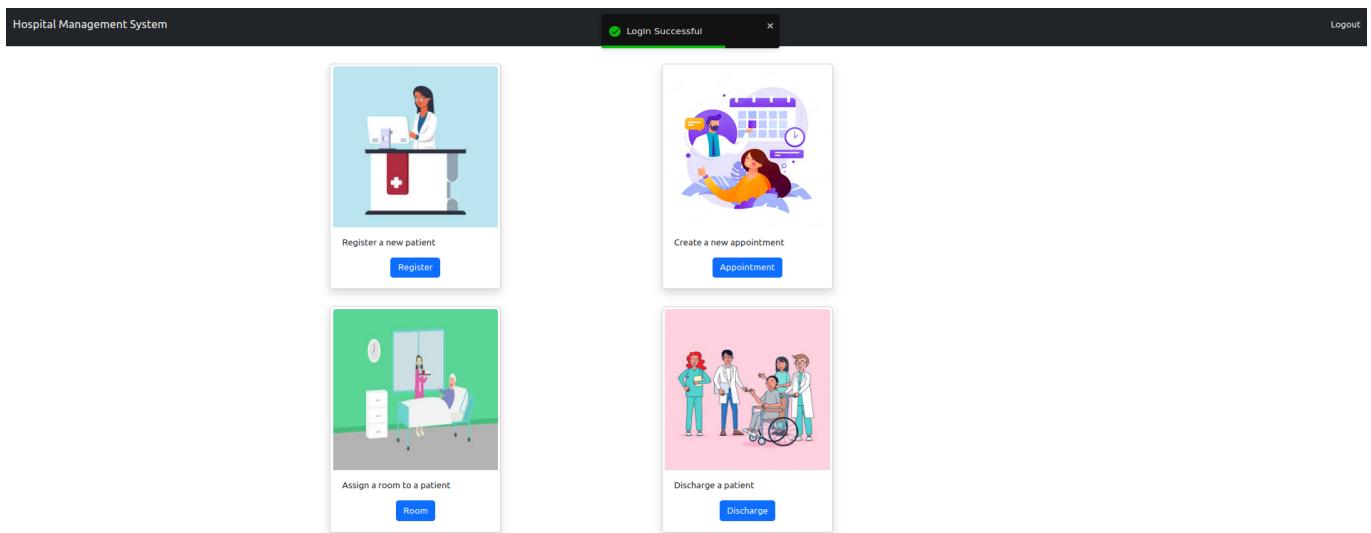
Welcome to RRR Hospital

From today, we are updating new features to our login portal and you can see the news, highlights and updates in the login page itself and get informed

Hurray!! Good News to all! We are adding new doctor specialized in bypass surgery

6.2 Front Desk Operator

6.2.1 Dashboard



6.2.2 Register Patient

The registration form is titled 'Registration form' and includes the following fields:

- Name
- Aadhar
- Address
- Phone Number
- Age
- Gender
- InsuranceID
- PCPDocID

At the bottom is a **Register Patient** button.

6.2.3 Appointment

[Go Back](#)

Appointment form

Date

XRAY

1

55

Emergency

[Get Slots](#)



08:00:00	08:30:00	09:00:00	09:30:00	10:00:00
10:30:00	11:00:00	11:30:00	12:00:00	12:30:00
14:00:00	14:30:00	15:00:00	15:30:00	16:00:00
16:30:00	18:00:00	18:30:00	19:00:00	19:30:00

6.2.4 Room Allocation

[Go Back](#)

Stay Form

Start Time

Room Number

Patient Aadhar

[Add Stay](#)

6.2.5 Discharge Patient

[Go Back](#)

Discharge Form

Patient Aadhar

[Discharge Patient](#)

6.2.6 Emergency Mail

New message from Korella Deners [Inbox](#)

 **email** <dbms23.mysql@gmail.com>
to me ▾

Hello ,
You got a new message from Korella Deners:

Emergency Appointment created for 1 From 08:30:00 on 2023-03-06

Waiting for your presence,
Admin

Email sent via [EmailJS.com](#)

[Reply](#) [Forward](#)

6.2.7 Weekly Report

New message [Inbox](#)

 **email** <dbms23.mysql@gmail.com>
to me ▾

Hello Respected Dr.,
Here is your weekly report of appointments of your patients,

*Appointment ID: 1
Appointment Date: Mon Mar 06 2023 00:00:00 GMT+0530 (India Standard Time)
Appointment Time: 08:30:00
Appointment Room: XRAY
Appointment Patient: 1
Appointment Emergency: 1*

*Appointment ID: 2
Appointment Date: Mon Mar 06 2023 00:00:00 GMT+0530 (India Standard Time)
Appointment Time: 16:00:00
Appointment Room: XRAY
Appointment Patient: 1
Appointment Emergency: 0*

*Undergoes Date: Mon Mar 06 2023 10:34:20 GMT+0530 (India Standard Time)
Undergoes Patient: 1
Patient Arrived*

Best wishes,
Admin

Email sent via [EmailJS.com](#)

6.3 Data Entry Operator

6.3.1 Dashboard



A screenshot of a web-based dashboard for a hospital management system. The interface has a dark header with a search bar containing the placeholder 'Search' and a magnifying glass icon. Below the header is a table with four columns: '#', 'Name', 'Aadhar', and 'Phone'. The table contains three rows of data:

#	Name	Aadhar	Phone
1	ram	1	099
2	shyam	2	099
3	dam	3	199

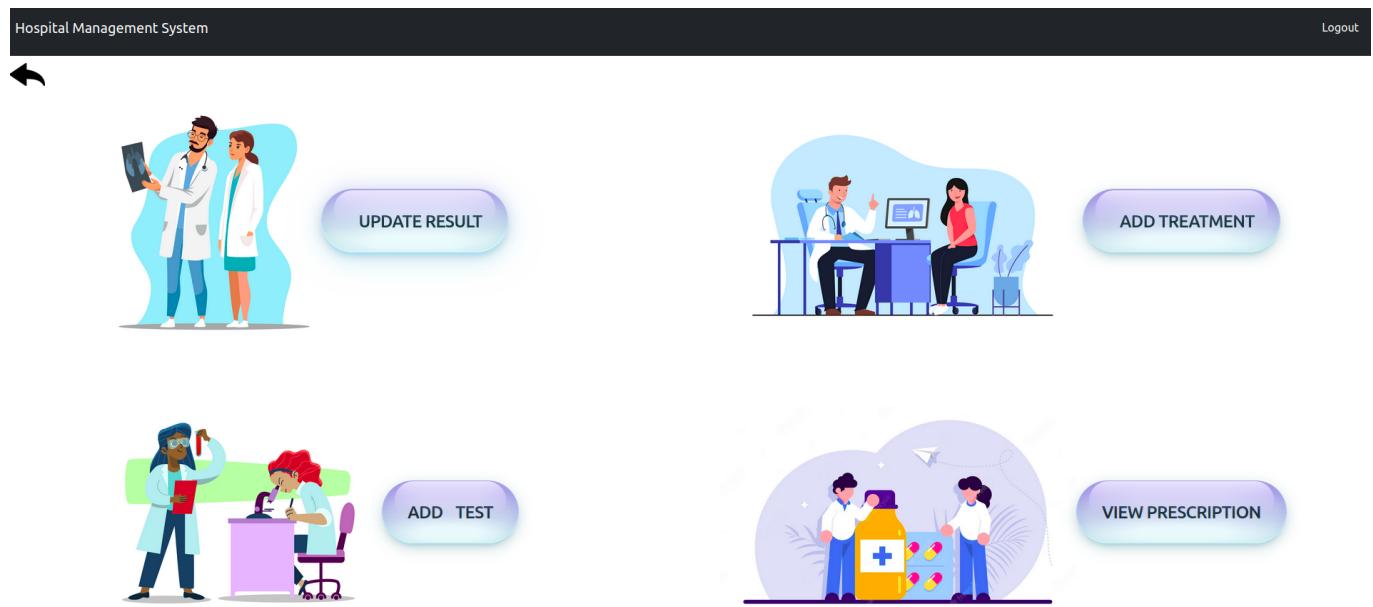
6.3.2 Search a Patient



A screenshot of a search results page for a patient named 'ram'. The page has a dark header with a search bar containing the letter 'r' and a magnifying glass icon. Below the header is a table with four columns: '#', 'Name', 'Aadhar', and 'Phone'. The table contains one row of data:

#	Name	Aadhar	Phone
1	ram	1	099

6.3.3 Select an Option



6.3.4 Add Test



Add test

Test
Add Test

6.3.5 Add Treatment



Add Treatment

Treatment
Doctor
Add Treatment

6.3.6 Update Result



Update Result

Test
Browse... No file selected.
Update Result

6.4 Doctor

6.4.1 Dashboard

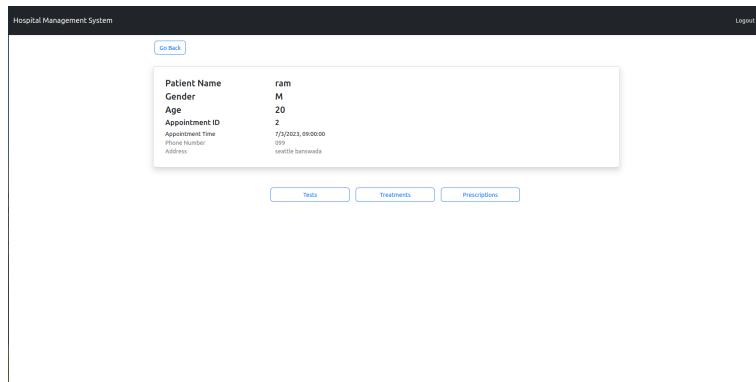
Hospital Management System

Logout

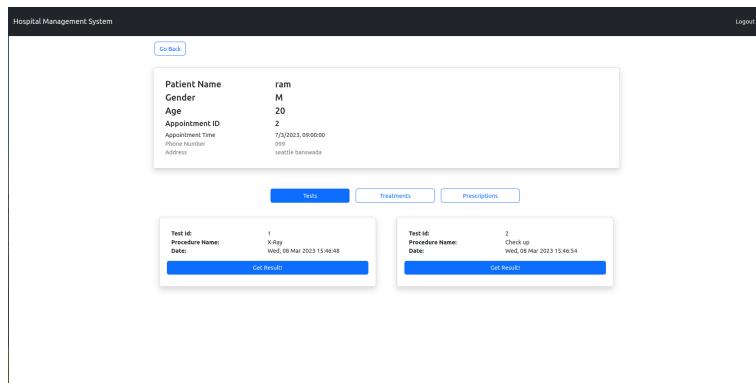
Appointment	Patient Name	Appointment ID	Appointment Time	Phone Number
	Patient 3	3	10/9/2023, 11:00:00	1111111111
	ram	2	7/3/2023, 09:00:00	099
	ram	1	6/6/2023, 18:00:00	099

View

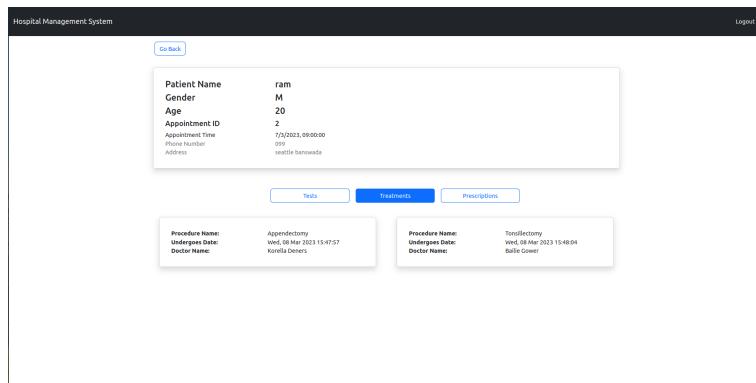
6.4.2 View Patient Details



6.4.3 View Test Results



6.4.4 View Treatments



6.4.5 View/Add Prescriptions

The screenshot shows the 'Prescription' section of the Hospital Management System. At the top, there is a summary box with patient details: Patient Name (ram), Gender (M), Age (20), Appointment ID (2), Appointment Time (7/3/2023, 09:00:00), Phone Number (999), and Address (seattle bennwade). Below this are three tabs: 'Tests', 'Treatments', and 'Prescriptions' (which is selected). A search bar shows 'Infants ibuprofen CVS Pharmacy' and a quantity input field showing '20 Tabs'. A 'Prescribe' button is below the search bar. At the bottom, there are two boxes for medication details. The left box shows: Medication Name (Dolo), Prescribed by (Korella Demers), Prescribed Date (Sat, 05 Mar 2023 18:30:00), and Prescribed Dose (3 Table). The right box shows: Medication Name (Dolo), Prescribed by (Korella Demers), Prescribed Date (Mon, 06 Mar 2023 07:32:41), and Prescribed Dose (4 Table).

6.5 Administrator

6.5.1 Dashboard

The screenshot shows the 'Database Administrator Homepage' of the Hospital Management System. The left sidebar has a dark background with white icons and text: Home, Database Administrators, Data Entry Operators, Doctors, Front Desk Operators, and Logout. The main content area has a light background with a title 'Hospital Management System Database Administrator Homepage'. Below the title is a welcome message: 'Welcome to the Hospital Management System Database Administrator Homepage! As the database administrator for the Hospital Management System, you play a critical role in ensuring that patient information is stored, managed, and protected appropriately. This homepage is designed to provide you with easy access to the tools and resources you need to perform your job effectively.' At the bottom, there is a form with a text input field 'What's the News?' and a blue 'Add News' button.

6.5.2 Show Administrators

Database Administrators

Employee ID	Name	Phone	Address	Actions
31	Hastie Le Fleming	850-448-0336	59 Memorial Point	<button>Edit</button>
32	Delbert Carnall	493-625-5739	3855 3rd Point	<button>Edit</button>
33	Nicolais De Stoop	533-124-0568	1 Mallard Avenue	<button>Edit</button>
36	Pegeen Sprague	845-196-0380	6319 Golf View Center	<button>Edit</button>
41	Hillard Eudall	105-644-7691	4 Express Way	<button>Edit</button>
42	Cullie Grimston	244-802-7403	89149 Mandrake Trail	<button>Edit</button>
43	Leeanne Watchorn	808-135-6919	1 Mifflin Street	<button>Edit</button>
45	Russ Bowbrick	729-508-3521	2 Ruskin Crossing	<button>Edit</button>

6.5.3 Show Data Entry Operators

Data Entry Operators

Employee ID	Name	Phone	Address	Actions
16	Gregorio Bohlmann	147-695-3997	813 Northport Court	<button>Edit</button> <button>Delete</button>
17	Wilbur Mosedale	389-337-3092	60 Declaration Circle	<button>Edit</button> <button>Delete</button>
18	Paulina Kleewein	387-683-3004	7447 Oxford Junction	<button>Edit</button> <button>Delete</button>
20	Abraham Pretor	619-915-9170	3 Brickson Park Parkway	<button>Edit</button> <button>Delete</button>
23	Adamo Everson	273-416-0071	56 Orin Lane	<button>Edit</button> <button>Delete</button>
24	Carlina Iacapucci	202-603-6475	60409 Crownhardt Road	<button>Edit</button> <button>Delete</button>
25	Lyssa Rockall	738-787-5760	62 Hayve Circle	<button>Edit</button> <button>Delete</button>
29	Corrinne Elven	208-231-3310	2008 Ilene Terrace	<button>Edit</button> <button>Delete</button>

6.5.4 Show Doctors

Doctors

Employee ID	Position	Name	Phone	Address	Present	Email	Actions
46	Junior Doctor	Ermentrude Simeons	896-257-4702	15010 Valley Edge Junction	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
48	Junior Doctor	Patricia Lambkin	984-456-8245	04546 Oak Valley Alley	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
50	Junior Doctor	Korella Deners	743-501-8844	4 Parkside Terrace	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
51	Senior Doctor	Bailie Gower	301-834-5370	4654 Gateway Center	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
53	Senior Doctor	Aron Bossingham	917-224-3396	10 Fairview Avenue	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
54	Head	Cirillo Pinock	587-336-9936	24931 Judy Way	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>
55	Senior Doctor	Eldridge Morfey	312-232-2460	17455 Manufacturers Parkway	1	likhith26090@gmail.com	<button>Edit</button> <button>Delete</button>

6.5.5 Show Front Desk Operators

Employee ID	Name	Phone	Address	Actions
1	Sebastien Lefever	239-199-7154	7 Calypso Point	<button>Edit</button> <button>Delete</button>
6	Laney Wanklin	533-963-8960	36 Artisan Street	<button>Edit</button> <button>Delete</button>
11	Lorens Bottomore	331-345-5086	4 Pawling Alley	<button>Edit</button> <button>Delete</button>
12	Katherina Humpage	901-476-9921	9 Twin Pines Point	<button>Edit</button> <button>Delete</button>
13	Arabelle Buddle	847-339-9243	510 Loeprich Parkway	<button>Edit</button> <button>Delete</button>

6.5.6 Add Administrator

Add a Database Administrator

Enter a name...

Enter aadhar number (12-digit)...

Enter a phone number (XXX-XXX-XXXX)...

Enter an address...

Enter a password...

Confirm password...

Add

Cancel

6.5.7 Add Doctor

Add a Doctor

Enter a name...

Enter aadhar number (12-digit)...

Enter Email...

Enter Position...

Enter a phone number (XXX-XXX-XXXX)...

Enter an address...

Enter a password...

Confirm password...

Add

Cancel

6.5.8 Add Data Entry Operator

Add a Data Entry Operator

Add

Cancel

6.5.9 Add Front Desk Operator

Add a Front Desk Operator

Add

Cancel