# Database Management Systems Lab

## Hospital Management System

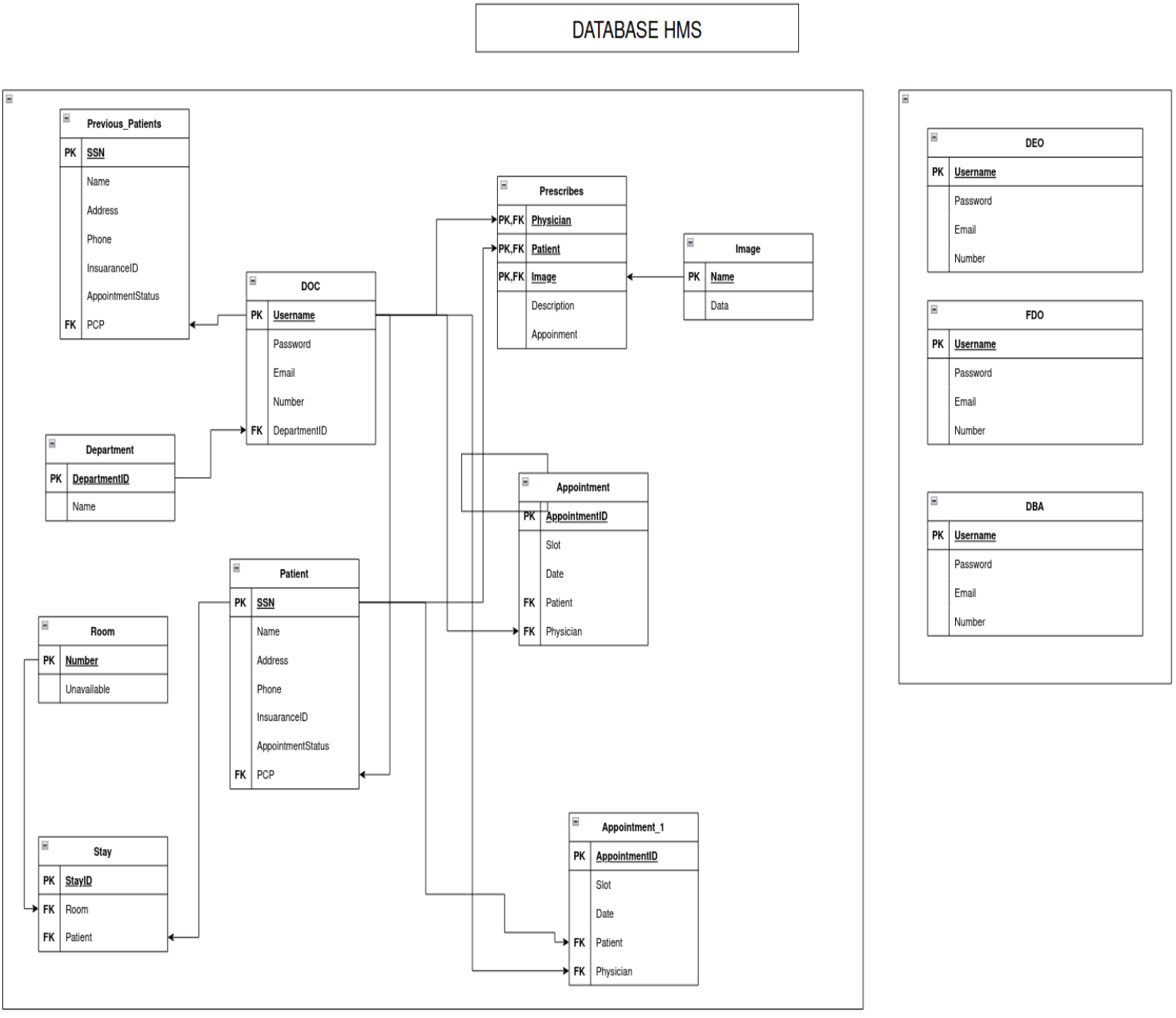
**Gangaram Arvind Sudewad {20CS30017}**
**Kanchi Mohan Krishna {20CS10030}**
**Lav Jharwal {20CS30031}**

## 1. Schema of the underlying database

From the above we can see that there are three tables kept separate; they are just to store the username and password of the users.

For DOC as a user we are connecting it with other tables in the database Doctor has some department which is taken from the table Department.

FDO can register new patients and then they are admitted by FDO with an algorithm which will select the least busiest physician from the specified department. Also the date and slots are asked as input and in the back end we are checking using a query such that no two slots clash occurs. So FDO appoints a doctor to the patient and admits to an available room.



**Python:** Django is built using the Python programming language and relies heavily on its features and capabilities.

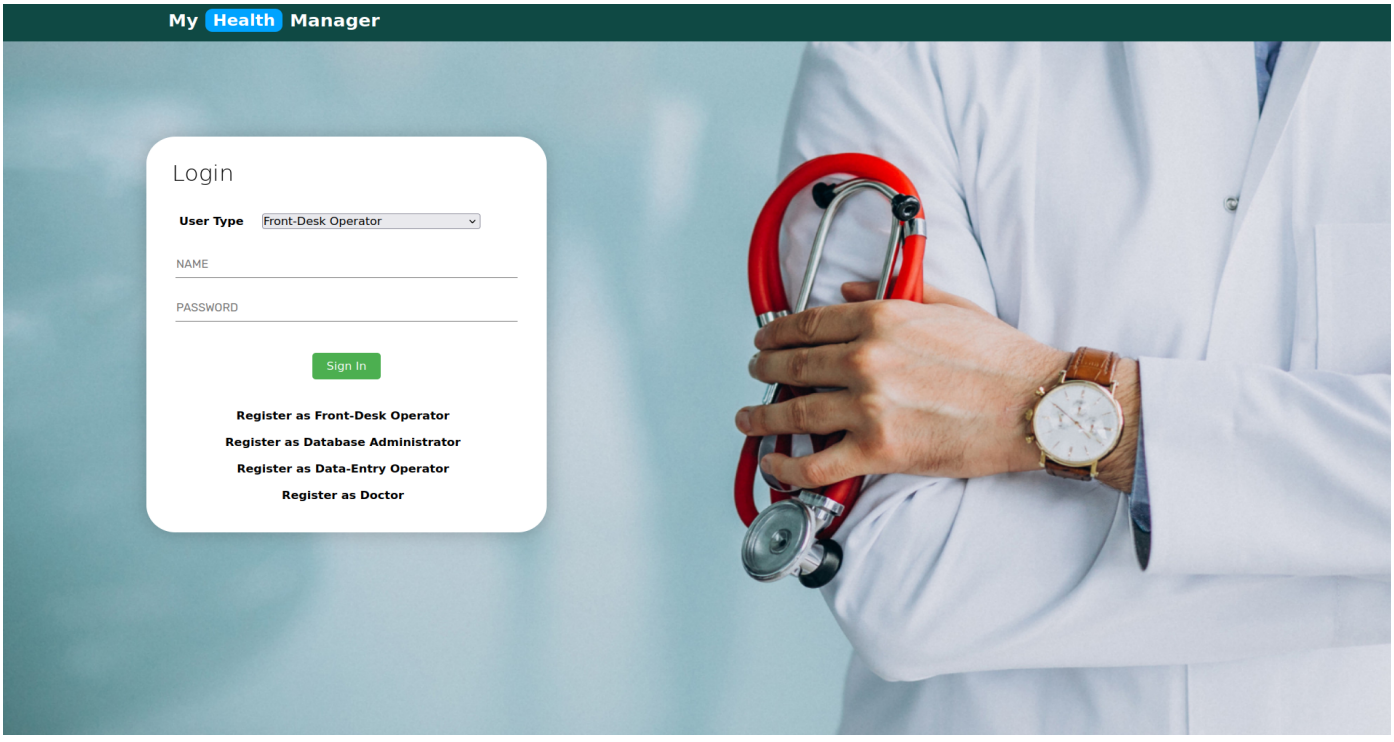
**HTML/CSS/JavaScript:** Django provides support for rendering dynamic web pages using templates written in HTML/CSS/JavaScript.

**SQL:** Django supports multiple databases, and it uses SQL to interact with them.

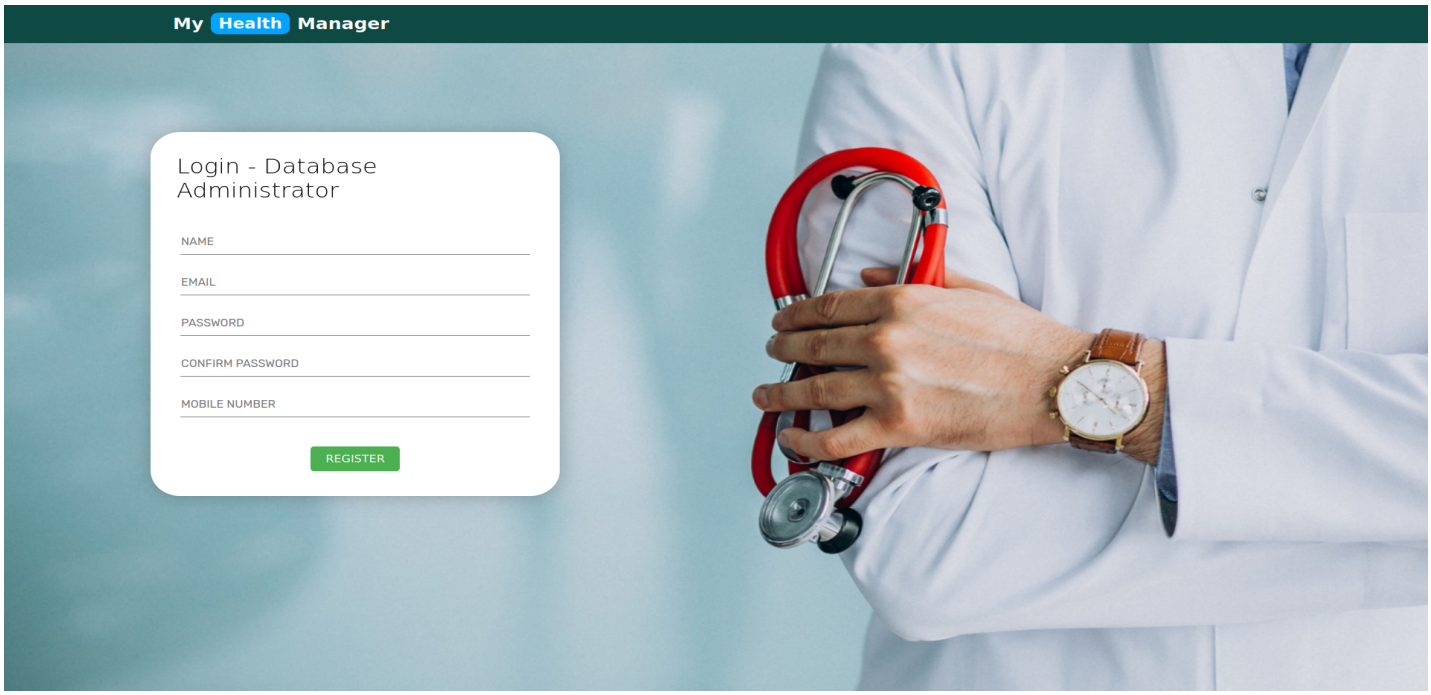
We have used the mysql connector present in python to connect to our local mysql database in place of django's inbuilt databases.

### 3. Triggers and workflows implemented

The first page is the login page where all users are allowed to login if they already have an account. If not then they can register too.

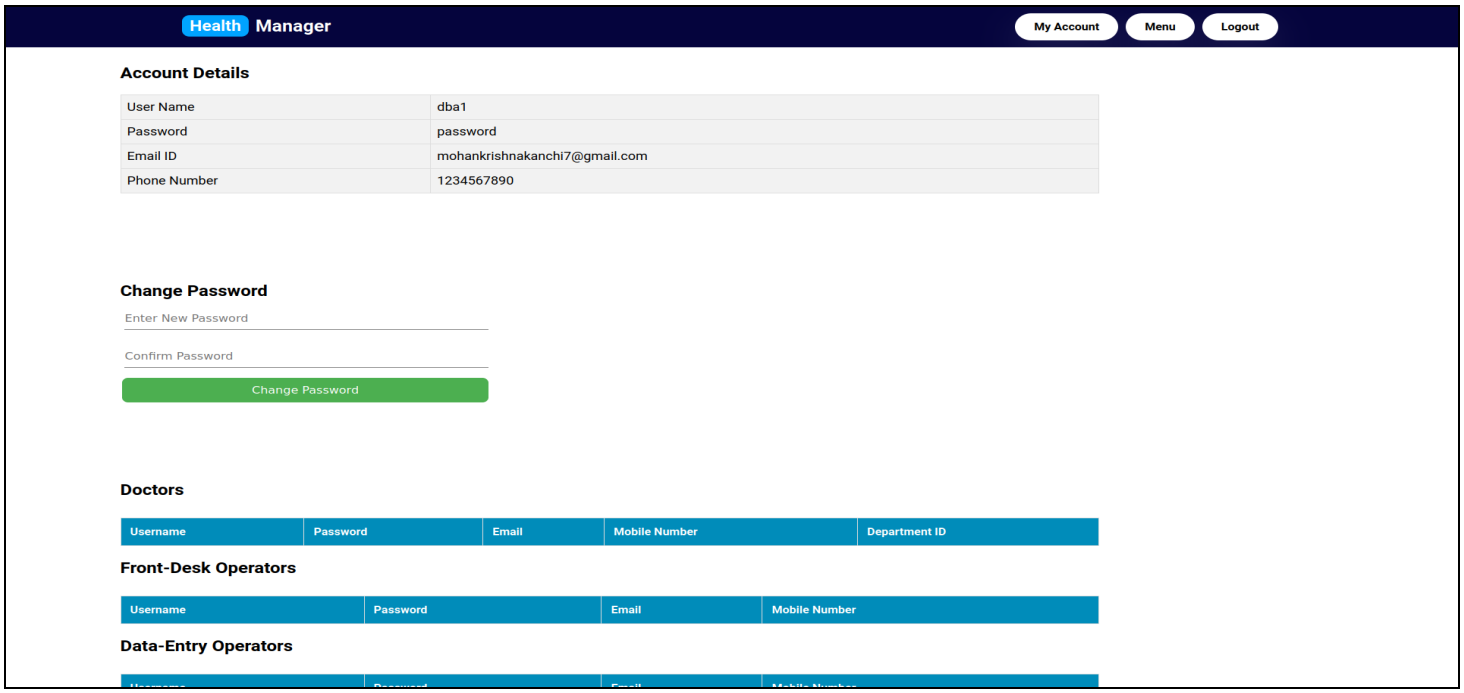



The register page asks for information which will be stored in the database.





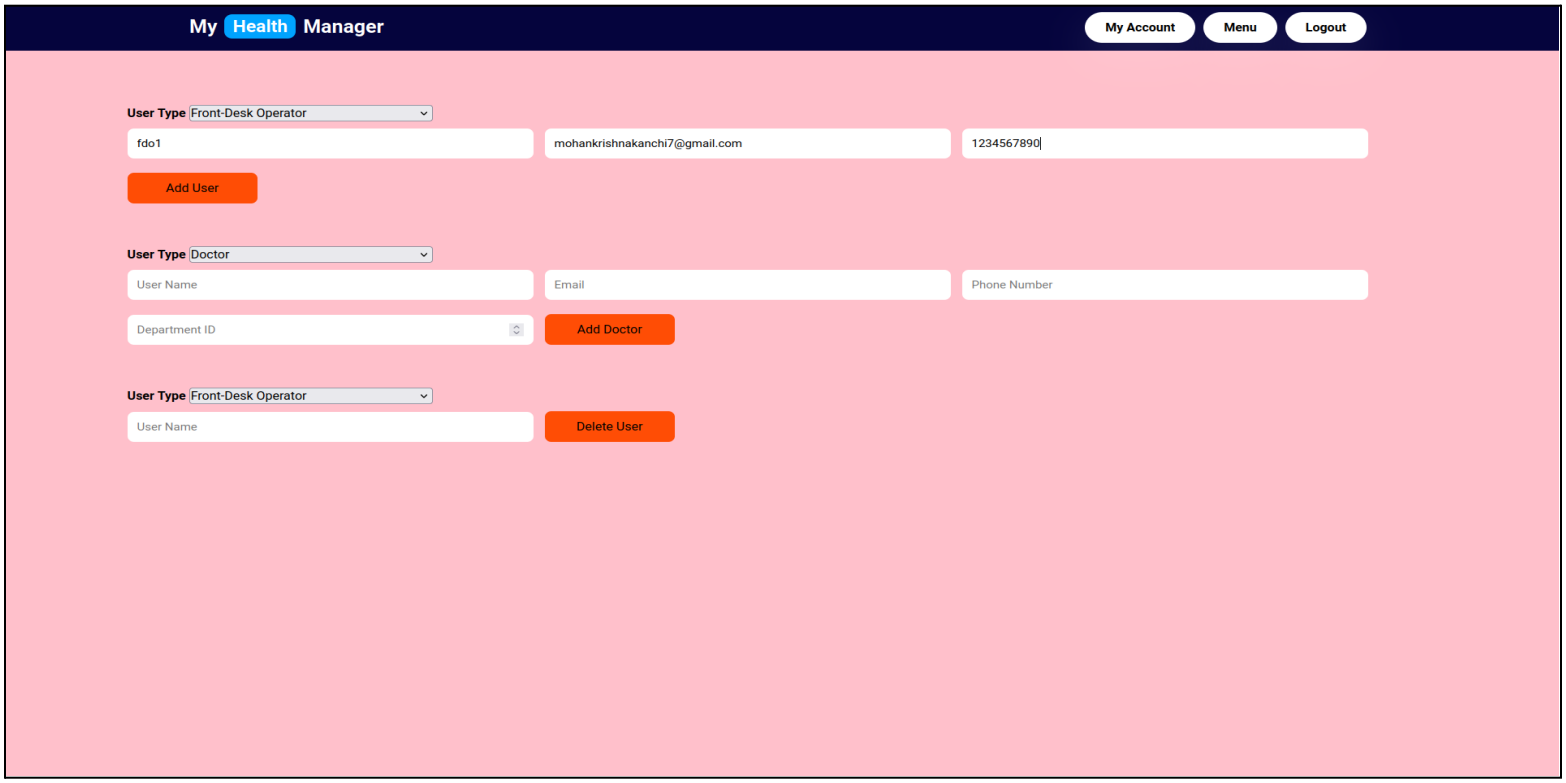
On logging in the user will get a home page from the above taskbar user can move to the account page which contains its account details.




Now on clicking Menu, User can perform assigned jobs as mentioned in the Assignment.

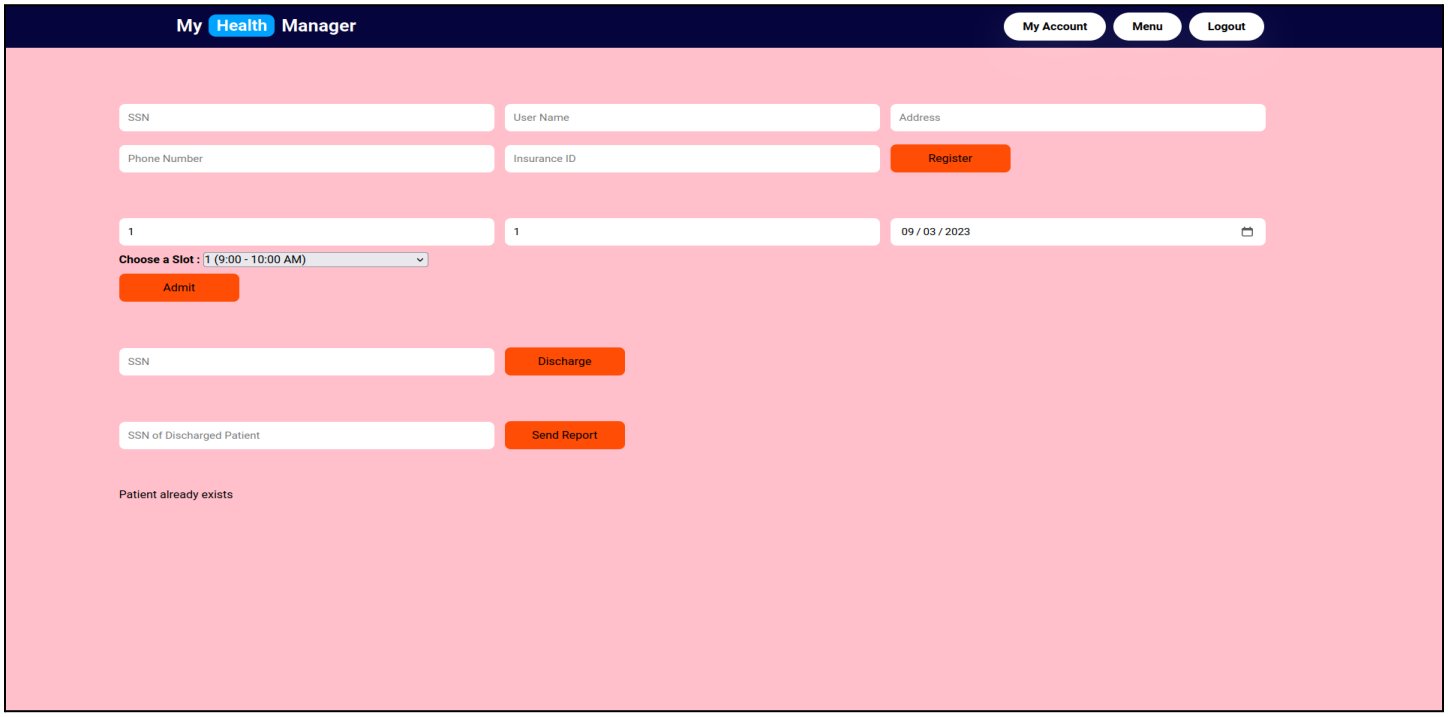
A **Database Administrator** can :

1) Add users of different types for whom the passwords are set to default, and the password can be further changed by the respective users. Which will later reflect in the database administrator's account.
2) Delete any existing users. Database administrators can keep an eye on each user and are able to remove users in cases of security reasons.



A **Front Desk Operator can**:

1) Register a patient by filling in the details of the patient.
2) Schedule an Appointment for a patient with the doctor of a suitable department and allocate a room based on its availability. If the patient selects a slot which clashes with any slot then it shows the slot clash as a message.
3) Discharge Patient, once patient has completed his appointment and prescribed medicines have been entered into the database by the Data-Entry Operator.

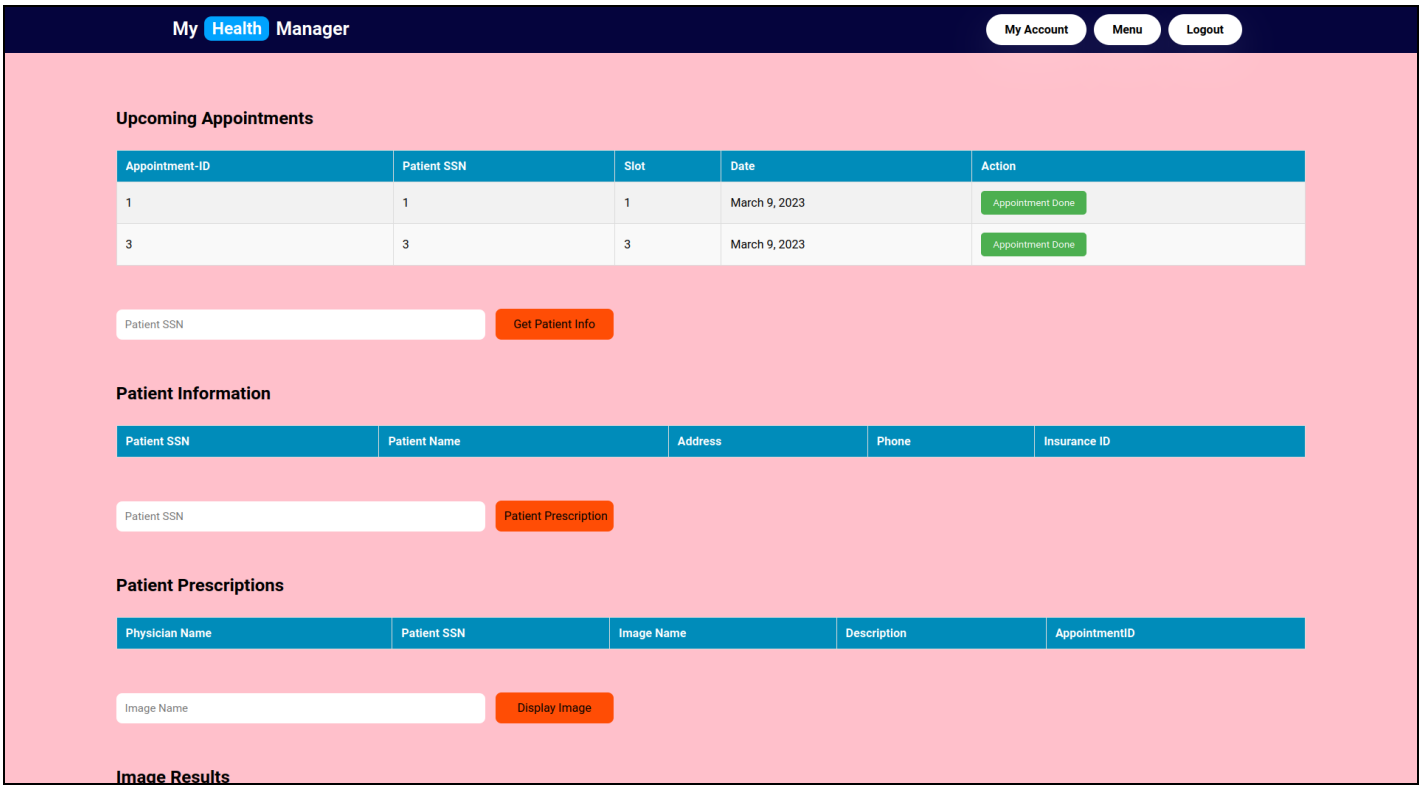



A **Doctor** can:

1) Enquire about any patient currently being treated.
2) Get Prescriptions , reports and X-ray results of any patient treated.

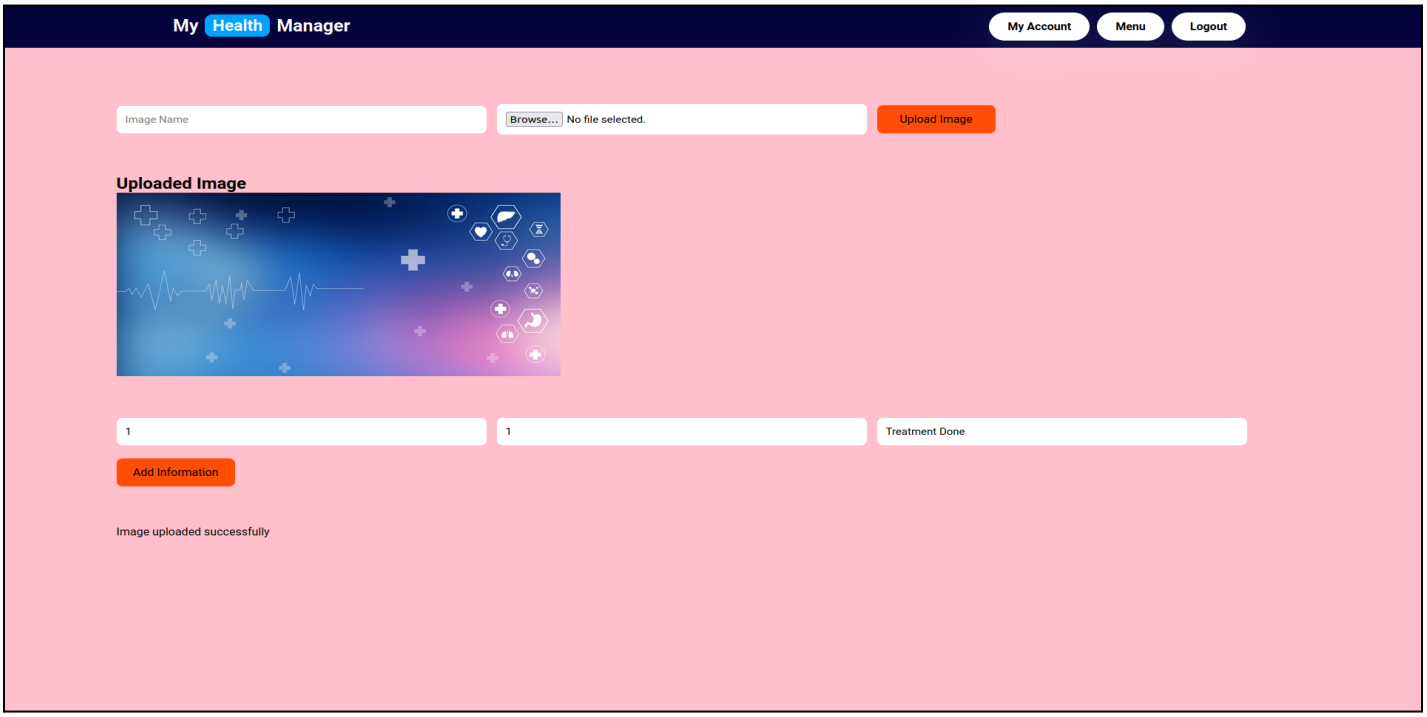
Also , When a Front Desk Operator admits a patient , the patient will be asked for the department and the slot for the appointment. Patient gets the doctor with the least number of appointments in the specific department along with a room to stay. The first room available is allotted to the patient. Either the room is not available or the slot is being clashed then we get an error message.

Doctor gets these assigned patients in an appointment queue as shown in the diagram . The doctor gets to choose the patient to be treated. When he is done with his appointment he updates the same in the action column.

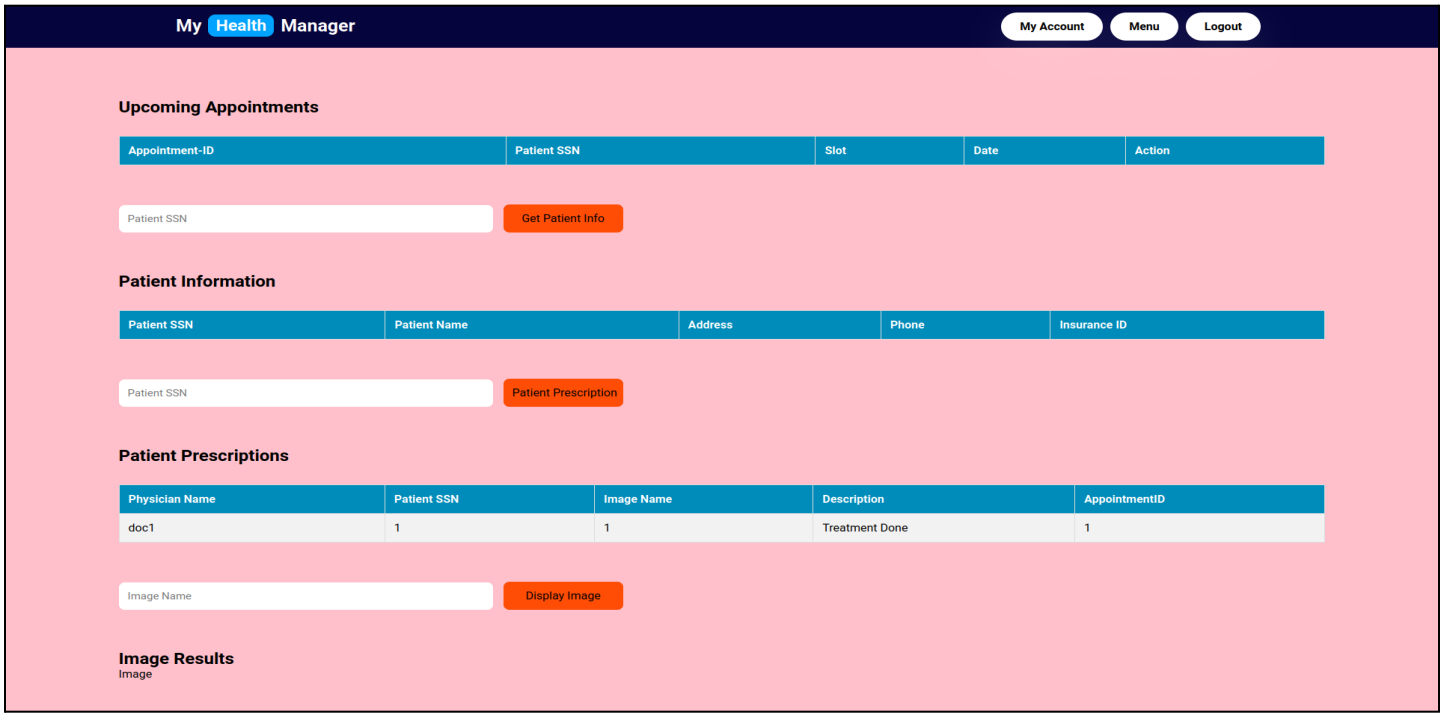



A **Data Entry Operator** can:

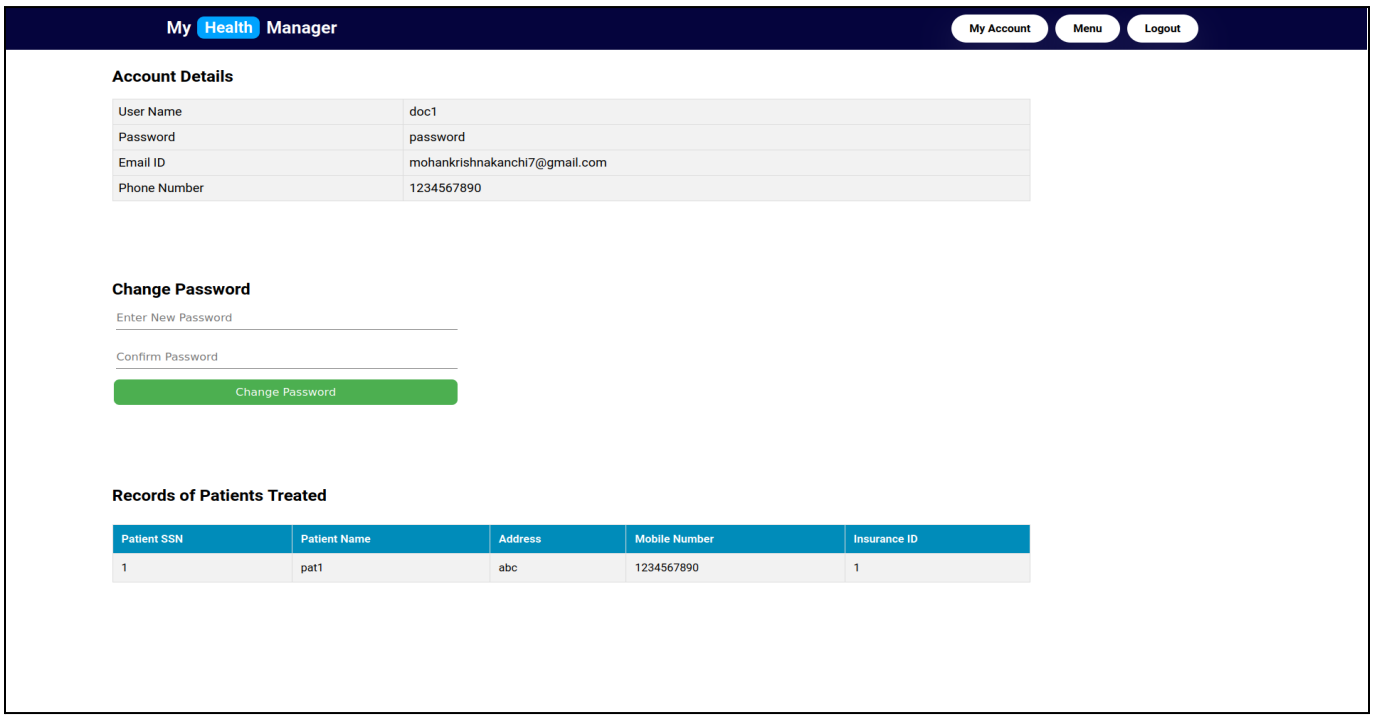
1. Upload the test results and images of the patient to the database.
2. Upload the prescriptions from the doctor to the patient's record along with the test results and images.



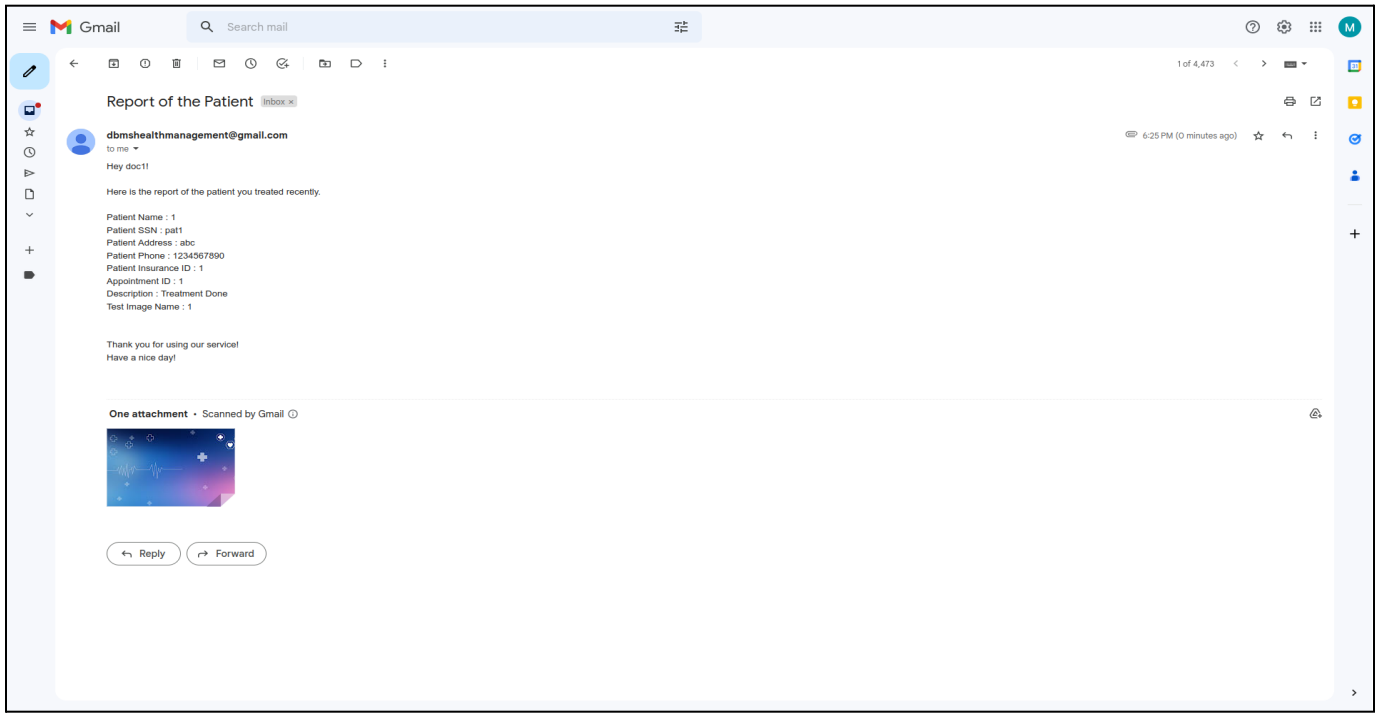
Now from the doctor's side he can see the patient’s prescriptions and enter the image name from the patient's prescription to get the x-ray images.



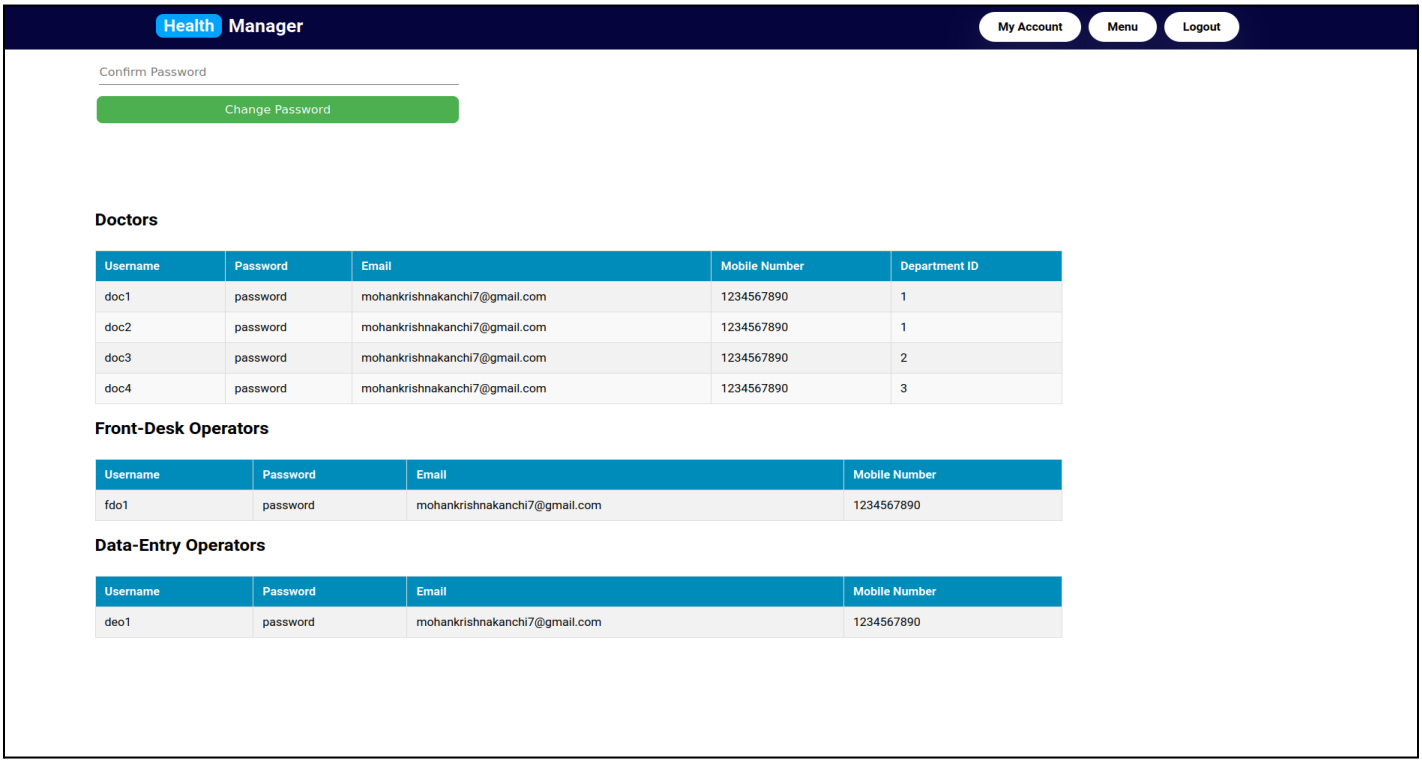

A patient can be discharged by the front desk operator when he completes his treatment. Once the patient is discharged his information will be reflected in previous patients of the doctor in the doctor's account section.



Also an auto generated email will be sent to the doctor consisting of the patient information and prescriptions along with the image attached to it.




Database Administrator's account section consists of data of all users.



## 4. Code listing as Appendix for queries and interfaces

## Some SQL queries to create the tables are:

## Query to create table FDO:

```
CREATE TABLE FDO (
        Username VARCHAR(50) NOT NULL PRIMARY KEY,
        Password VARCHAR(255) NOT NULL,
        Email VARCHAR(255) NOT NULL,
        Number varchar(50) NOT NULL
);

CREATE TABLE DEO (
        Username VARCHAR(50) NOT NULL PRIMARY KEY,
        Password VARCHAR(255) NOT NULL,
        Email VARCHAR(255) NOT NULL,
        Number varchar(50) NOT NULL
);

CREATE TABLE DBA (
        Username VARCHAR(50) PRIMARY KEY,
        Password VARCHAR(255) NOT NULL,
        Email VARCHAR(255) NOT NULL,
        Number varchar(50) NOT NULL
);

create table Department(
        DepartmentID int NOT NULL,
        name varchar(50) NOT NULL,
        primary key(DepartmentID)
);

create table DOC (
        Username varchar(50) NOT NULL,
        Password varchar(50) NOT NULL,
        Email varchar(50) NOT NULL,
        Number varchar(50) NOT NULL,
        DepartmentID int NOT NULL,
        primary key(Username),
        foreign key(DepartmentID) references Department(DepartmentID)
);
```

```
create table Patient(
	SSN int NOT NULL,
	Name varchar(50) NOT NULL,
	Address varchar(255) NOT NULL,
	Phone varchar(50),
	InsuranceID int NOT NULL,
	PCP varchar(50),
	AppointmentStatus BOOLEAN default false,
	primary key(SSN),
	foreign key(PCP) references DOC(Username)
);

create table Room(
	Number int NOT NULL,
	Unavailable BOOLEAN DEFAULT false,
	primary key(Number)
);

create table Stay(
	StayID int NOT NULL AUTO_INCREMENT,
	Room int NOT NULL,
	Patient int NOT NULL,
	primary key(StayID),
	foreign key(Room) references Room(Number),
	foreign key(Patient) references Patient(SSN)
);

create table Image(
	Name varchar(200) not null,
	Data LONGBLOB,
	primary key(Name)
);

create table Appointment(
	AppointmentID int AUTO_INCREMENT,
	Patient int NOT NULL,
	Physician varchar(50) NOT NULL,
	Slot INT NOT NULL,
	Date DATE NOT NULL,
	primary key(AppointmentID),
```

```
        foreign key(Patient) references Patient(SSN),
        foreign key(Physician) references DOC(Username));
create table Prescribes(
        Physician varchar(50) NOT NULL,
        Patient int NOT NULL,
        Image varchar(50) NOT NULL,
        Description varchar(1000),
        Appointment int,
        primary key(Physician,Patient,Image),
        foreign key(Physician) references DOC(Username),
        foreign key(Image) references Image(Name)
);

create table Previous_Patients(
        SSN int NOT NULL,
        Name varchar(50) NOT NULL,
        Address varchar(255) NOT NULL,
        Phone varchar(50),
        InsuranceID int NOT NULL,
        PCP varchar(50),
        AppointmentStatus BOOLEAN default false,
        primary key(SSN),
        foreign key(PCP) references DOC(Username)
);

create table Appointment_1(
        AppointmentID int AUTO_INCREMENT,
        Patient int NOT NULL,
        Physician varchar(50) NOT NULL,
        Slot INT NOT NULL,
        Date DATE NOT NULL,
        primary key(AppointmentID),
        foreign key(Patient) references Patient(SSN),
        foreign key(Physician) references DOC(Username)
);
```