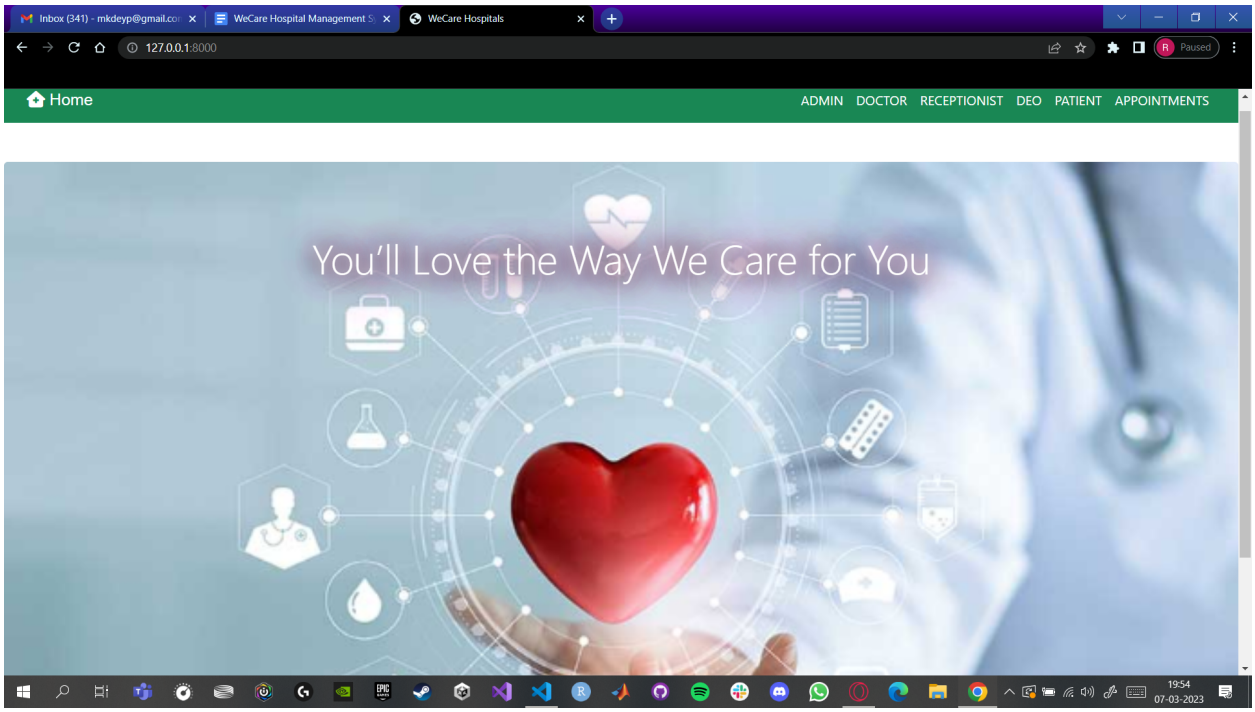
Database Management Systems Laboratory

Group Members-
Kriti Bhardwaj(20CS30028)
Nirbhay Kumar(20CS10040)
Vinod Meena(20CS10074)
Sonu Kumar Yadav(20CS10061)
Pranil Dey(20CS30038)

WeCare Hospitals Management System



Database schemas and characteristics of our System:-

1) Schema for database administrator:

```
auth_user (

  id integer NOT NULL PRIMARY KEY AUTOINCREMENT,

  password varchar(128) NOT NULL,

  last_login datetime NULL,

  is_superuser bool NOT NULL,

  username varchar(150) NOT NULL UNIQUE,

  first_name varchar(30) NOT NULL,
```

```
  last_name varchar(30) NOT NULL,
  email varchar(254) NOT NULL,
  is_staff bool NOT NULL,
  is_active bool NOT NULL,
  date_joined datetime NOT NULL,
  CONSTRAINT
auth_user_username_2e2e9cfde2e5e35a3e3a78e59e7b6141_uniq UNIQUE
(username)
);
```

The auth_user entity represents a database administrator for the hospital.The entity represents the user accounts and their authentication in the system.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each user account, which is automatically incremented with each new account created.
- **password**: A hashed representation of the user's password, which is stored securely in the database.
- **last_login**: The date and time of the user's last successful login, or NULL if the user has never logged in.
- **is_superuser**: A boolean value indicating whether the user has superuser privileges (i.e., administrative access to the system).
- **username**: The unique username chosen by the user to identify their account.
- **first_name**: The user's first name.
- **last_name**: The user's last name.

- **email**: The user's email address.
- **is_staff**: A boolean value indicating whether the user is a staff member (i.e., has access to administrative tools).
- **is_active**: A boolean value indicating whether the user's account is active (i.e., they can log in).
- **date_joined**: The date and time the user's account was created.

Additionally, we have added a unique constraint on the username column to ensure that each username is unique within the table.

2) Schema for Doctor:

```
doctor (

    id INT PRIMARY KEY AUTO_INCREMENT,

    user_id INT NOT NULL UNIQUE,

    profile_pic VARCHAR(255),

    address VARCHAR(40) NOT NULL,

    mobile VARCHAR(20),

    department VARCHAR(50) NOT NULL,

    status BOOLEAN DEFAULT FALSE,

    FOREIGN KEY (user_id) REFERENCES user(id)

);

ALTER TABLE doctor ADD COLUMN name VARCHAR(255) GENERATED ALWAYS AS
(CONCAT((SELECT first_name FROM user WHERE id=user_id), ' ', (SELECT
last_name FROM user WHERE id=user_id))) STORED;
```

The doctor entity represents doctors or medical professionals affiliated in a healthcare system.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each doctor, which is automatically incremented with each new doctor added.
- **user_id**: The unique identifier of the user associated with this doctor, likely a foreign key referencing the id column of a user table.
- **profile_pic**: The filename or URL of an image file representing the doctor's profile picture.
- **address**: The doctor's address, which is required.
- **mobile**: The doctor's mobile number.
- **department**: The department or area of expertise the doctor is associated with, which is required.
- **status**: A boolean value indicating whether the doctor is currently available or not, with a default value of FALSE.
- **FOREIGN KEY (user_id) REFERENCES user(id)**: A foreign key constraint that references the id column of the user table, which likely stores information about users of the healthcare system.

Additionally, the last statement adds a new column to the **doctor** table called **name**, which will always contain the concatenated value of the first and last name of the user associated with the doctor, stored as a computed column in the table.

3) Schema for Data entry operator:

```
deo (

    id INT PRIMARY KEY AUTO_INCREMENT,

    user_id INT NOT NULL UNIQUE,

    address VARCHAR(40),

    mobile VARCHAR(20),

    status BOOLEAN DEFAULT FALSE,

    FOREIGN KEY (user_id) REFERENCES user(id)

);

ALTER TABLE deo ADD COLUMN name VARCHAR(255) GENERATED ALWAYS AS
(CONCAT((SELECT first_name FROM user WHERE id=user_id), ' ', (SELECT
last_name FROM user WHERE id=user_id))) STORED;
```

The deo entity represents Data Entry Operators (DEOs) in our healthcare system who are members of the hospital management staff and responsible for scheduling tests/treatments for the patients, updating the test/treatment results after completion, and adding patients' health records as per requirements.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each DEO, which is automatically incremented with each new DEO added.
- **user_id**: The unique identifier of the user associated with this DEO, likely a foreign key referencing the id column of a user table.
- **address**: The DEO's address.

- **mobile**: The DEO's mobile number.
- **status**: A boolean value indicating whether the DEO is currently available or not, with a default value of FALSE.
- **FOREIGN KEY (user_id) REFERENCES user(id)**: A foreign key constraint that references the **id** column of the **user** table, which stores information about users of the system.

Additionally, the last statement adds a new column to the **deo** table called **name**, which will always contain the concatenated value of the first and last name of the user associated with the DEO, stored as a computed column in the table.

4) Schema for Front Desk Operator:

```
receptionist (
    id INT NOT NULL AUTO_INCREMENT,
    user_id INT NOT NULL,
    address VARCHAR(40),
    mobile VARCHAR(20),
    PRIMARY KEY (id),
    FOREIGN KEY (user_id) REFERENCES user(id)
);
```

The receptionist entity represents the Front Desk Operators (FDOs) in our healthcare system who are the first point of contact for patients and visitors at the hospital's reception. A front-desk operator is responsible for patient registration, scheduling patient appointments, handling patients' admission/discharge, and having access to information about all patients.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each receptionist, which is automatically incremented with each new receptionist added.
- **user_id**: The unique identifier of the user associated with this receptionist, likely a foreign key referencing the id column of a user table.
- **address**: The receptionist's address.
- **mobile**: The receptionist's mobile number.
- **PRIMARY KEY (id)**: Specifies that the id column is the primary key of the table, ensuring each row has a unique identifier.
- **FOREIGN KEY (user_id) REFERENCES user(id)**: A foreign key constraint that references the **id** column of the **user** table, which stores information about users of the system.

5) Schema for Patient:

```
patient (

    id INT PRIMARY KEY AUTO_INCREMENT,

    user_id INT NOT NULL UNIQUE,

    profile_pic VARCHAR(255),

    address VARCHAR(40) NOT NULL,

    mobile VARCHAR(20) NOT NULL,

    symptoms VARCHAR(100) NOT NULL,

    assigned_doctor_id INT,

    admit_date DATE DEFAULT CURRENT_TIMESTAMP,
```

```
    status BOOLEAN DEFAULT FALSE,
    tests VARCHAR(100) DEFAULT 'Not Treated',
    prescription VARCHAR(500) DEFAULT 'Not Treated',
    FOREIGN KEY (user_id) REFERENCES user(id),
    FOREIGN KEY (assigned_doctor_id) REFERENCES doctor(id)
);

ALTER TABLE patient ADD COLUMN name VARCHAR(255) GENERATED ALWAYS AS
(CONCAT((SELECT first_name FROM user WHERE id=user_id), ' ', (SELECT
last_name FROM user WHERE id=user_id))) STORED;
```

The patient entity represents a patient seeking treatment in the hospital. A patient can book an appointment with a doctor with the help of the front-desk operator, get their tests/treatments scheduled, and their health records updated by the data-entry operator. However, a patient is not a user of the application. It represents a passive entity for storing a patient's information.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each patient, which is automatically incremented with each new patient added.
- **user_id**: The unique identifier of the user associated with this patient, likely a foreign key referencing the id column of a user table.
- **profile_pic**: The path to the patient's profile picture.
- **address**: The patient's address.
- **mobile**: The patient's mobile number.
- **symptoms**: A string describing the patient's symptoms.

- **assigned_doctor_id**: The ID of the doctor assigned to this patient, likely a foreign key referencing the id column of a doctor table.
- **admit_date**: The date when the patient was admitted to the hospital, with a default value of the current timestamp.
- **status**: A boolean value indicating whether the patient is still admitted or has been discharged.
- **tests**: A string describing any tests the patient has undergone or is scheduled to undergo, with a default value of "Not Treated".
- **prescription**: A string describing any medication prescribed to the patient, with a default value of "Not Treated".
- **PRIMARY KEY (id)**: Specifies that the id column is the primary key of the table, ensuring each row has a unique identifier.
- **FOREIGN KEY (user_id) REFERENCES user(id)**: A foreign key constraint that references the id column of the user table, which likely stores information about users of the system.
- **FOREIGN KEY (assigned_doctor_id) REFERENCES doctor(id)**: A foreign key constraint that references the **id** column of the **doctor** table, indicating the doctor assigned to this patient.

Additionally, the last statement adds a new computed column called **name** to the **patient** table, which is similar to the **name** column added to the **doctor** and **deo** tables. The **name** column is generated automatically based on the **first_name** and **last_name** values of the user associated with the patient.

6) Schema for storing details about appointment:

```
appointment (

    id INT PRIMARY KEY AUTO_INCREMENT,

    patient_id INT,
```

```
    doctor_id INT,
    patient_name VARCHAR(40),
    doctor_name VARCHAR(40),
    appointment_date DATE DEFAULT CURRENT_TIMESTAMP,
    description TEXT,
    status BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (patient_id) REFERENCES patient(id),
    FOREIGN KEY (doctor_id) REFERENCES doctor(id)
);
```

The appointment entity represents a patient's appointment to consult a physician.

Here is a brief overview of the table columns:

- **id**: A unique integer identifier for each appointment, which is automatically incremented with each new appointment added.
- **patient_id**: The ID of the patient associated with this appointment, likely a foreign key referencing the id column of a patient table.
- **doctor_id**: The ID of the doctor associated with this appointment, likely a foreign key referencing the id column of a doctor table.
- **patient_name**: A string representing the name of the patient associated with this appointment.
- **doctor_name**: A string representing the name of the doctor associated with this appointment.
- **appointment_date**: The date of the appointment, with a default value of the current timestamp.

- **description**: A text field describing the details of the appointment.
- **status**: A boolean value indicating whether the appointment has been completed or is still pending.
- **PRIMARY KEY (id)**: Specifies that the id column is the primary key of the table, ensuring each row has a unique identifier.
- **FOREIGN KEY (patient_id) REFERENCES patient(id)**: A foreign key constraint that references the id column of the patient table, indicating the patient associated with this appointment.
- **FOREIGN KEY (doctor_id) REFERENCES doctor(id)**: A foreign key constraint that references the **id** column of the **doctor** table, indicating the doctor associated with this appointment.

7) Schema to store the patient discharge details:

```
patient_discharge_details (

    id INT PRIMARY KEY AUTO_INCREMENT,

    patient_id INT,

    patient_name VARCHAR(40),

    assigned_doctor_name VARCHAR(40),

    address VARCHAR(40),

    mobile VARCHAR(20),

    symptoms VARCHAR(100),

    test_results VARCHAR(500),

    admit_date DATE NOT NULL,

    release_date DATE NOT NULL,
```

```
    days_spent INT NOT NULL,
    room_charge INT NOT NULL,
    medicine_cost INT NOT NULL,
    doctor_fee INT NOT NULL,
    other_charge INT NOT NULL,
    total INT NOT NULL,
    FOREIGN KEY (patient_id) REFERENCES patient(id)
);
```

The patient_discharge_details entity represents the details of a patient's discharge from the hospital after treatment.

Here is a brief overview of the table columns:

- **id**: An auto-incrementing integer value that serves as the primary key of the table.
- **patient_id**: The ID of the patient who was discharged, referencing the id column of the patient table.
- **patient_name**: The name of the patient who was discharged.
- **assigned_doctor_name**: The name of the doctor who was assigned to the patient.
- **address**: The address of the patient who was discharged.
- **mobile**: The mobile phone number of the patient who was discharged.
- **symptoms**: The symptoms that the patient had when they were admitted to the hospital.
- **test_results**: The results of the tests that were performed on the patient.
- **admit_date**: The date on which the patient was admitted to the hospital.

- **release_date**: The date on which the patient was discharged from the hospital.
- **days_spent**: The number of days that the patient spent in the hospital.
- **room_charge**: The cost of the hospital room for the duration of the patient's stay.
- **medicine_cost**: The cost of the medicines that were given to the patient during their stay.
- **doctor_fee**: The fee charged by the doctor who treated the patient.
- **other_charge**: Any other charges that were incurred during the patient's stay.
- **total**: The total amount charged for the patient's treatment.
- **FOREIGN KEY (patient_id) REFERENCES patient(id)**: A foreign key constraint that ensures that the **patient_id** column references the **id** column of the patient table.

Languages/Technologies Used:-

1) Frontend

- **HTML**
- **CSS**
- **BootStrap**

2) Backend

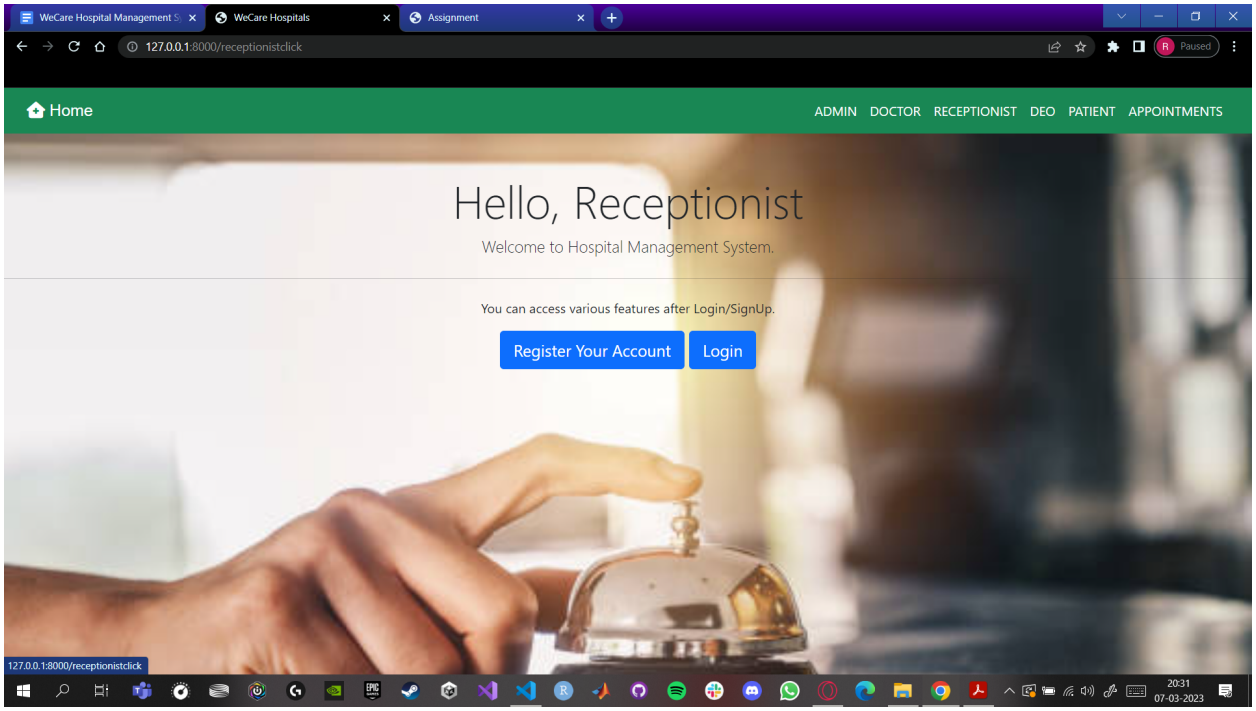
- **Python with Django framework**
- **Database- MySQL**

3) Tools

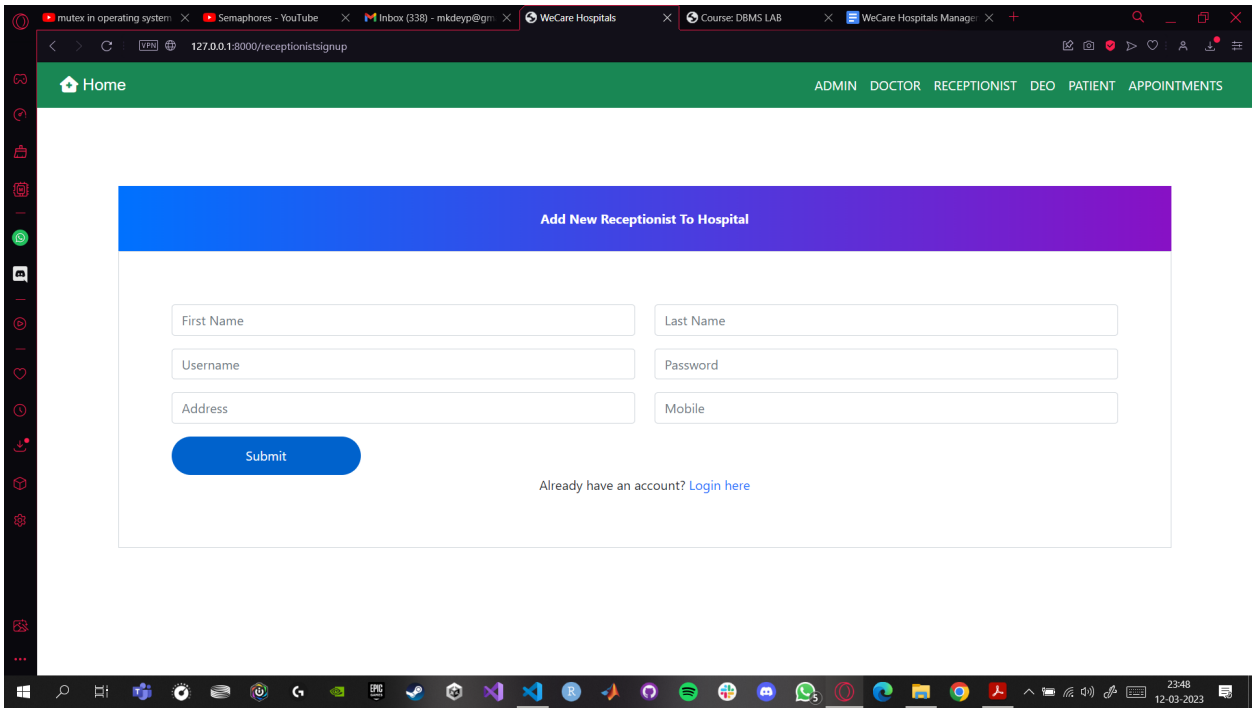
- **Visual Studio Code Editor**
- **Internet resources for learning**

Functionalities:-

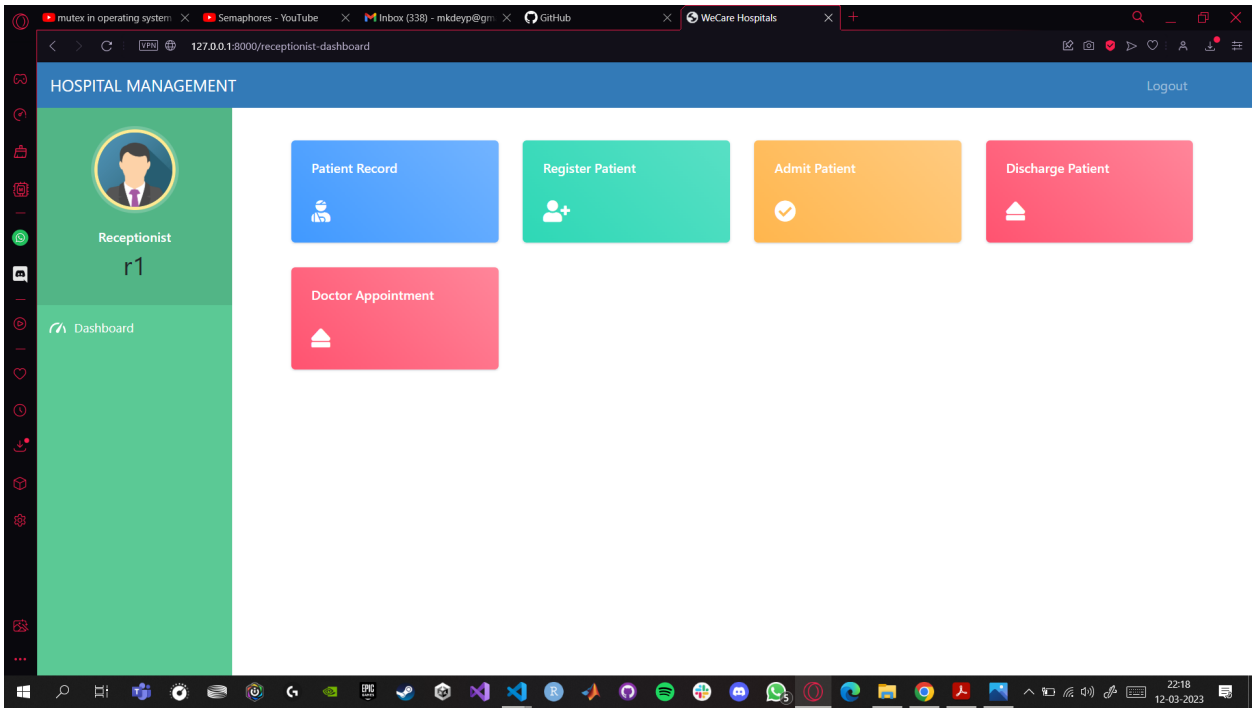
1) Front Desk Operators:



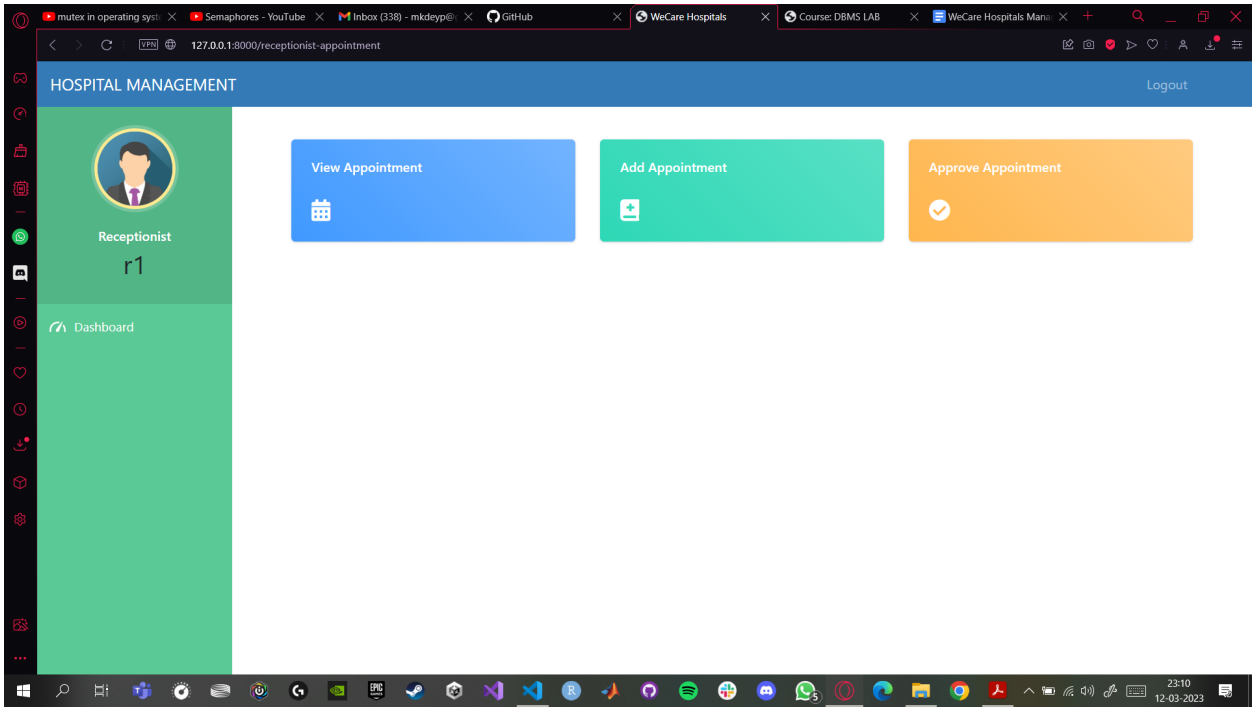
- Registers/admits/discharges patients
- Patient registration/discharge and doctor appointment/test scheduling – information about new patients are registered, appointments based on availability and priority are scheduled, doctors are notified about the appointments in a dashboard. For admitted patients a room is assigned based on available room capacity. For discharged patients information is preserved and room occupancy is updated.



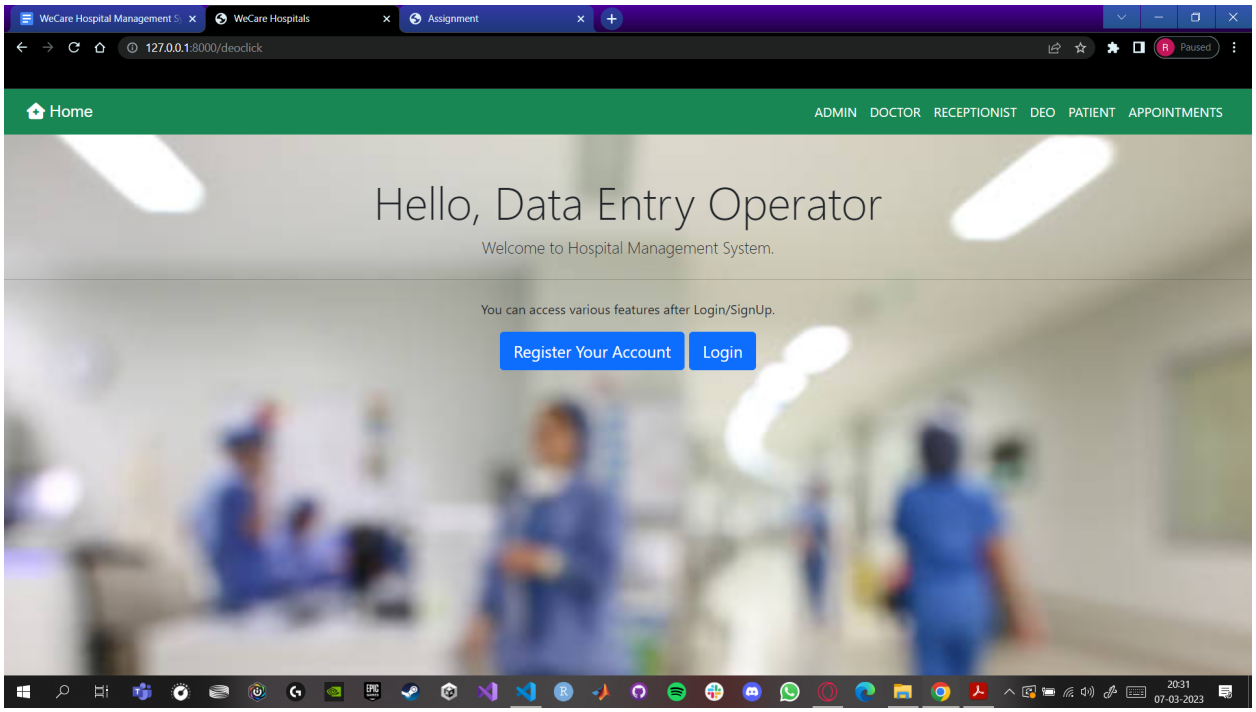
This is the registration page of the receptionist/front desk operator in our healthcare system.



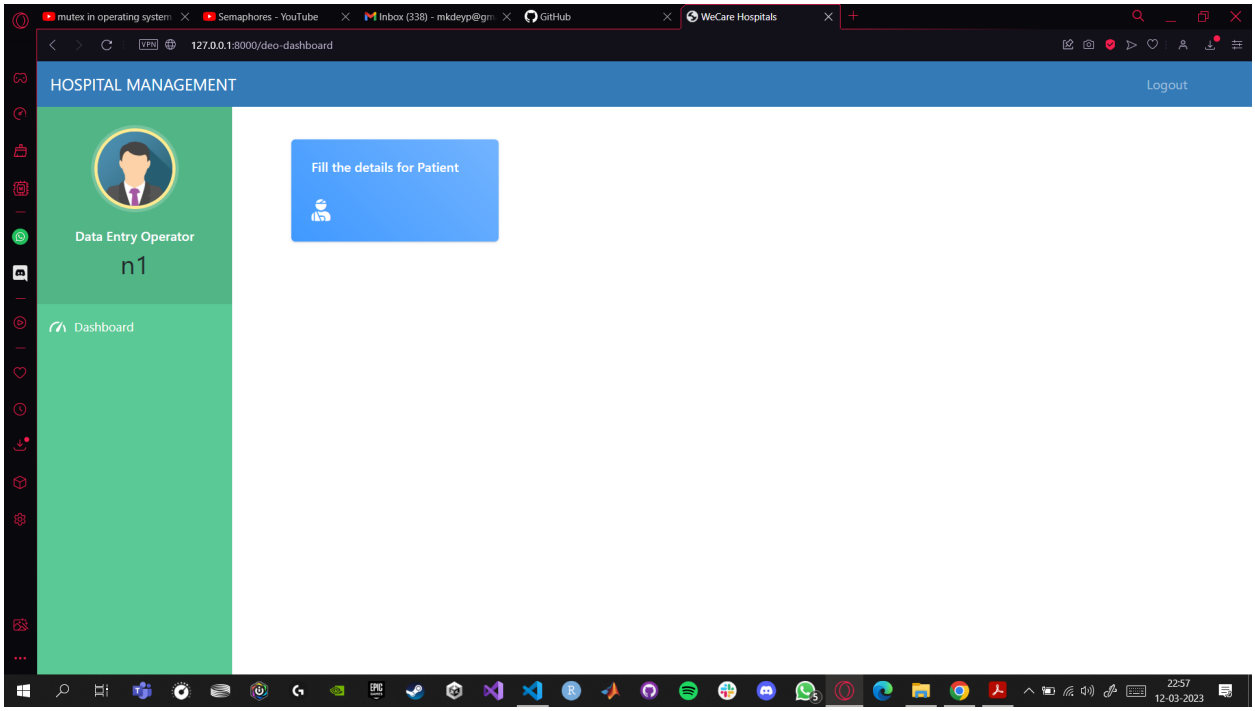
This is the dashboard of the frontdesk operator/receptionist of our system. It provides functions like maintaining patient records, registering a patient, admitting them and discharging a patient. Also, they can add, view and approve doctor appointments according to their schedule.



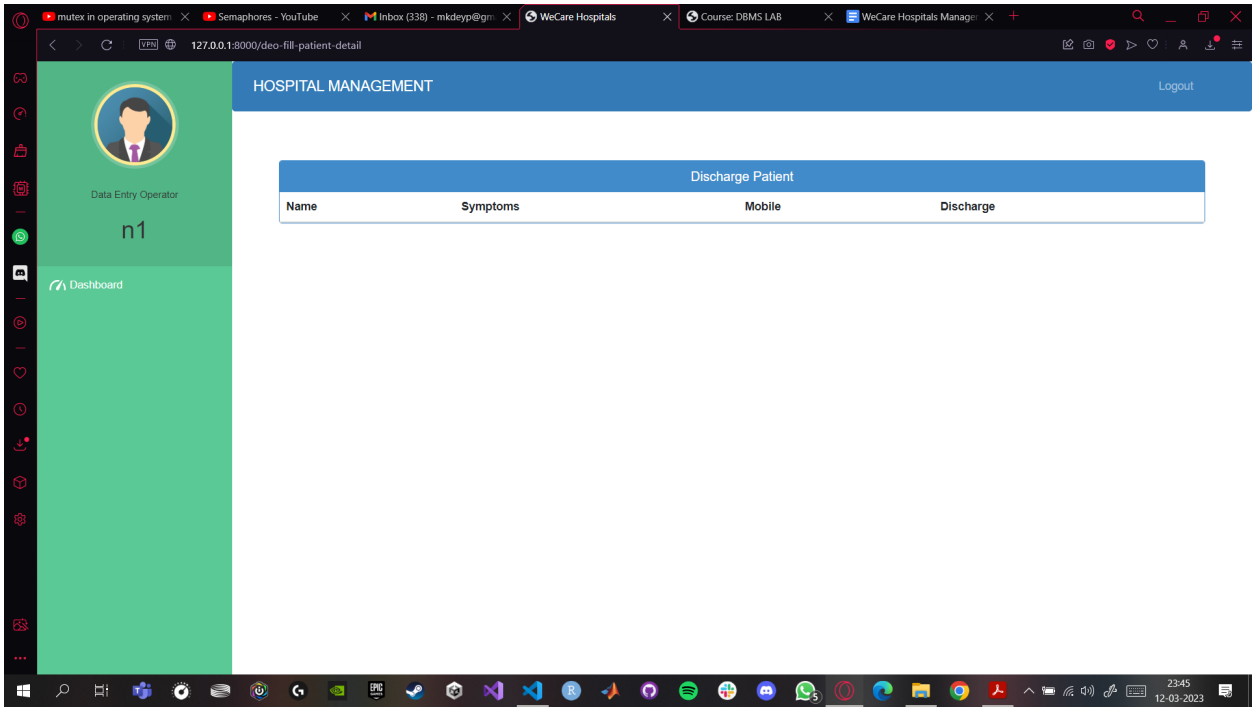
2) Data Entry Operators:



- Enters patient data about tests and treatments

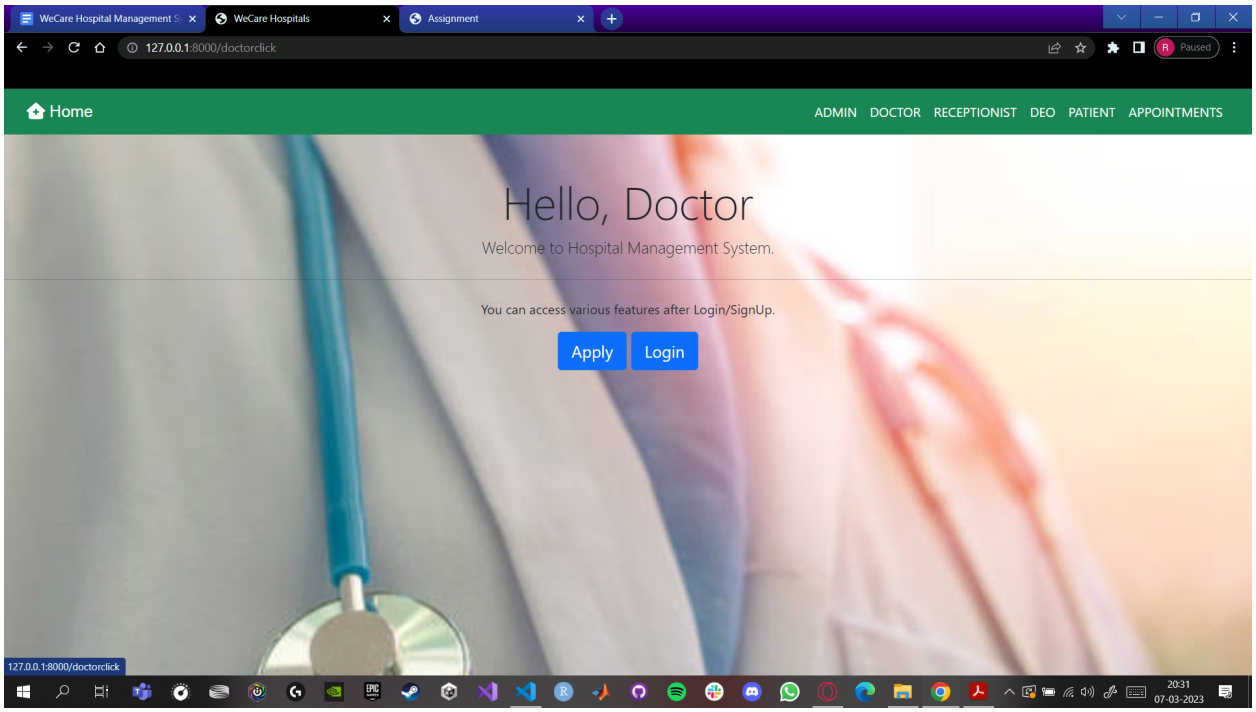


This is the dashboard of a data entry operator/nurse. The function of a nurse is to maintain the details of the patients and update the treatments and records of the patients.

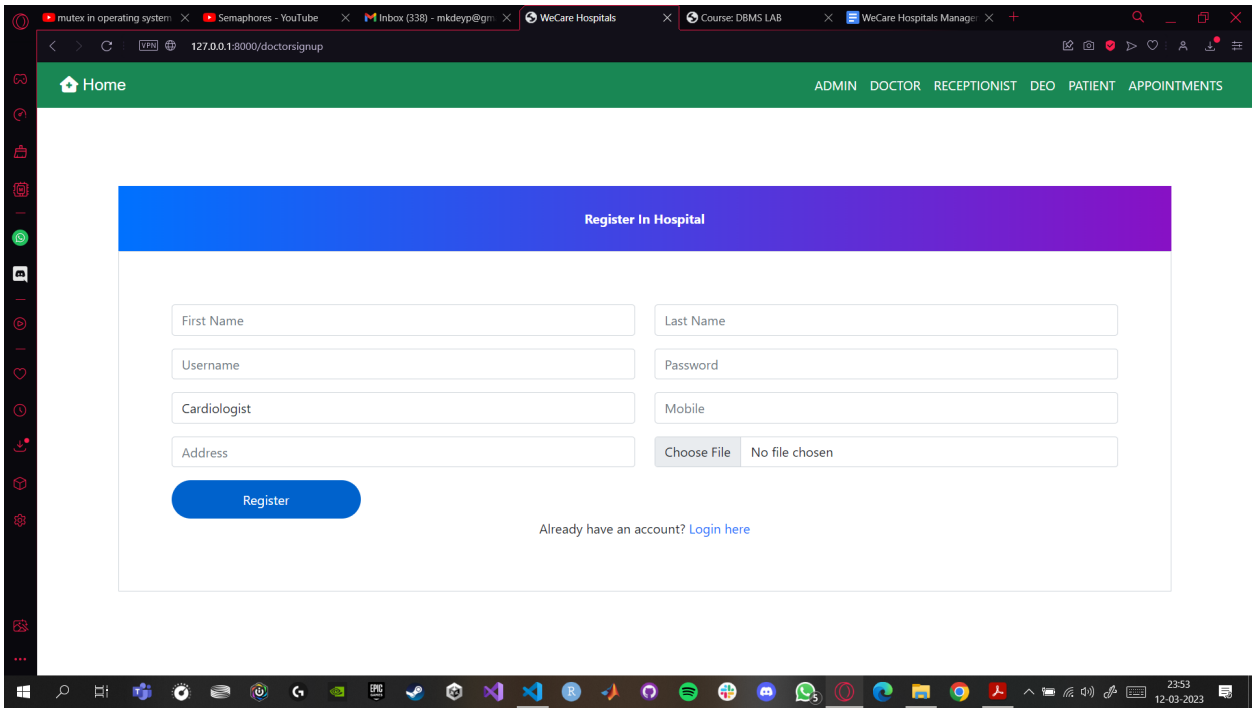


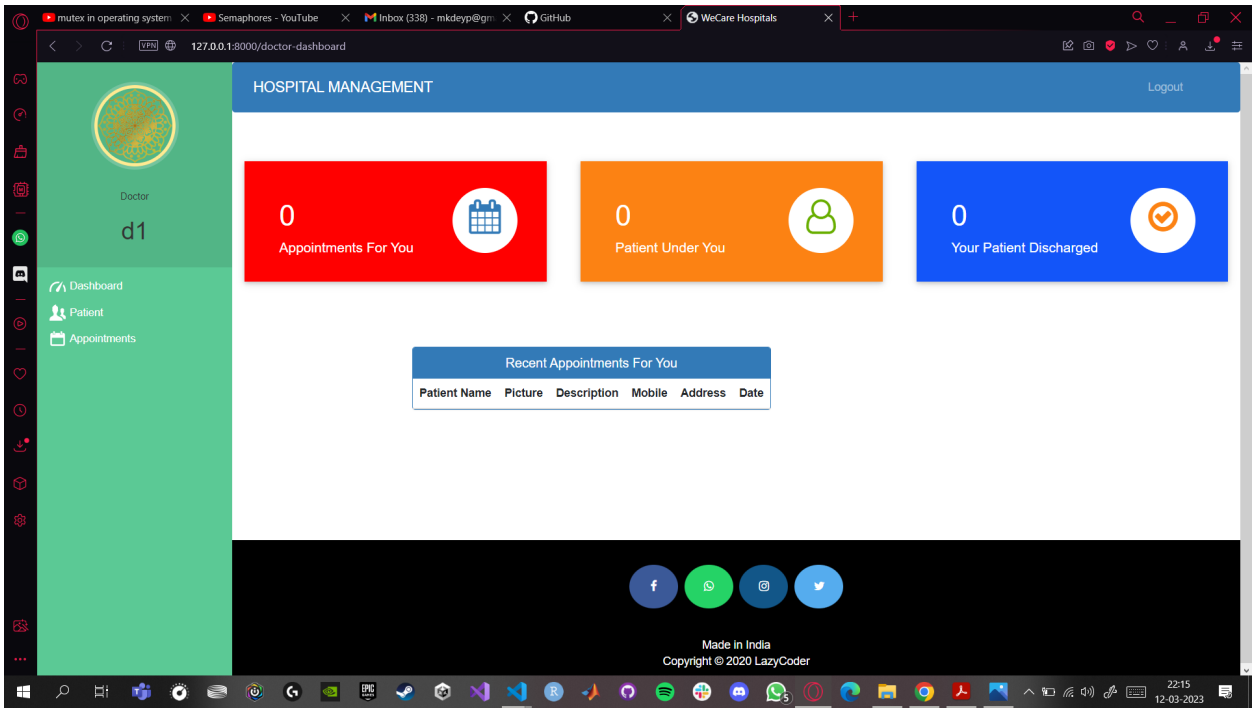
This is where the details of the patient are kept/updated by our nurse.

3) Doctors:

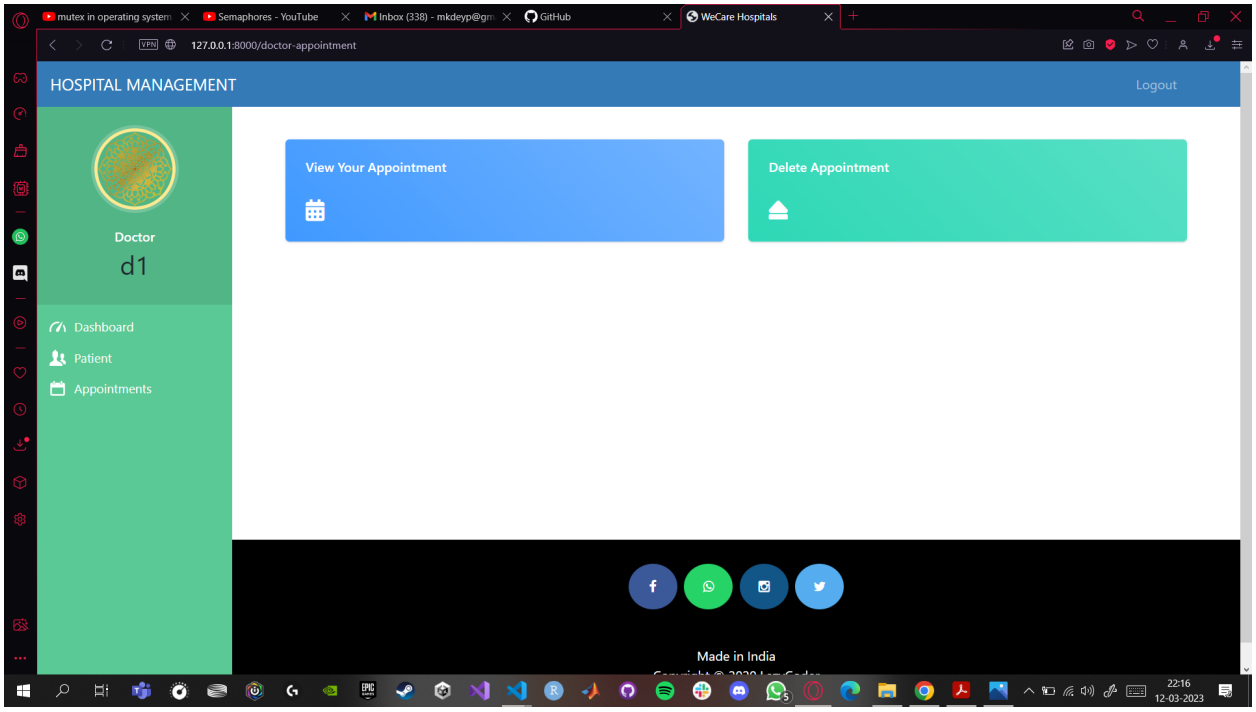


- Query patient information

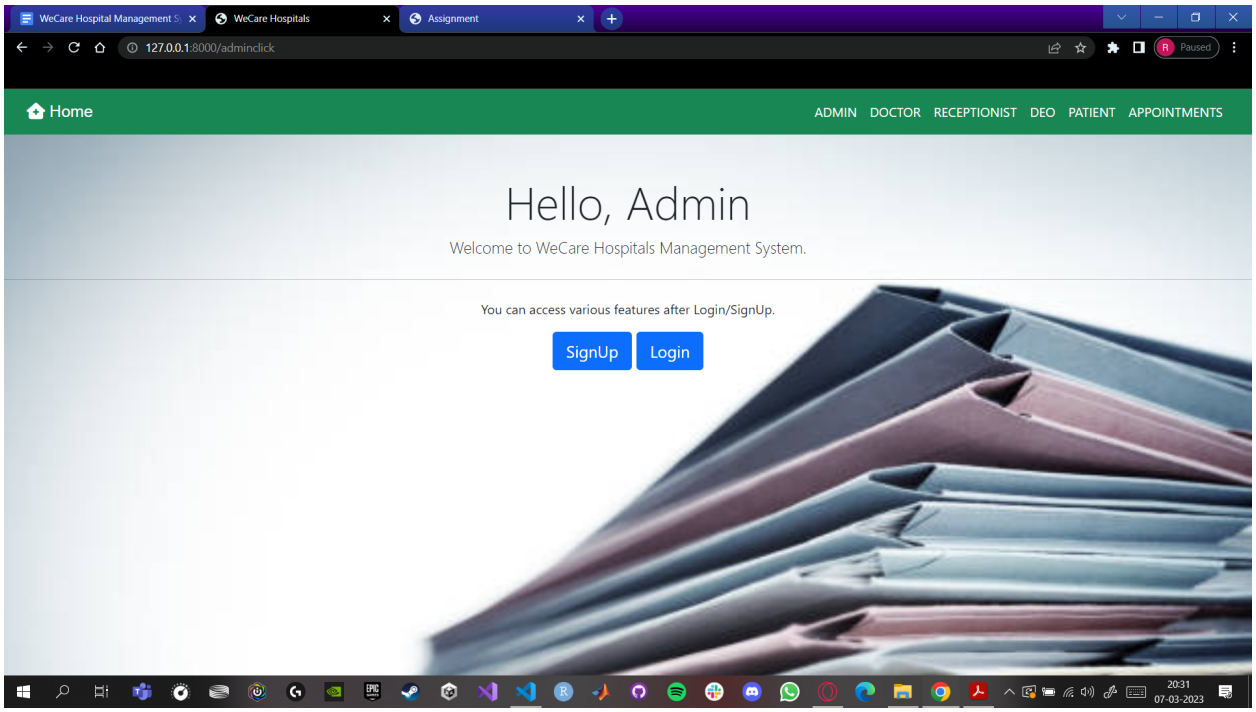




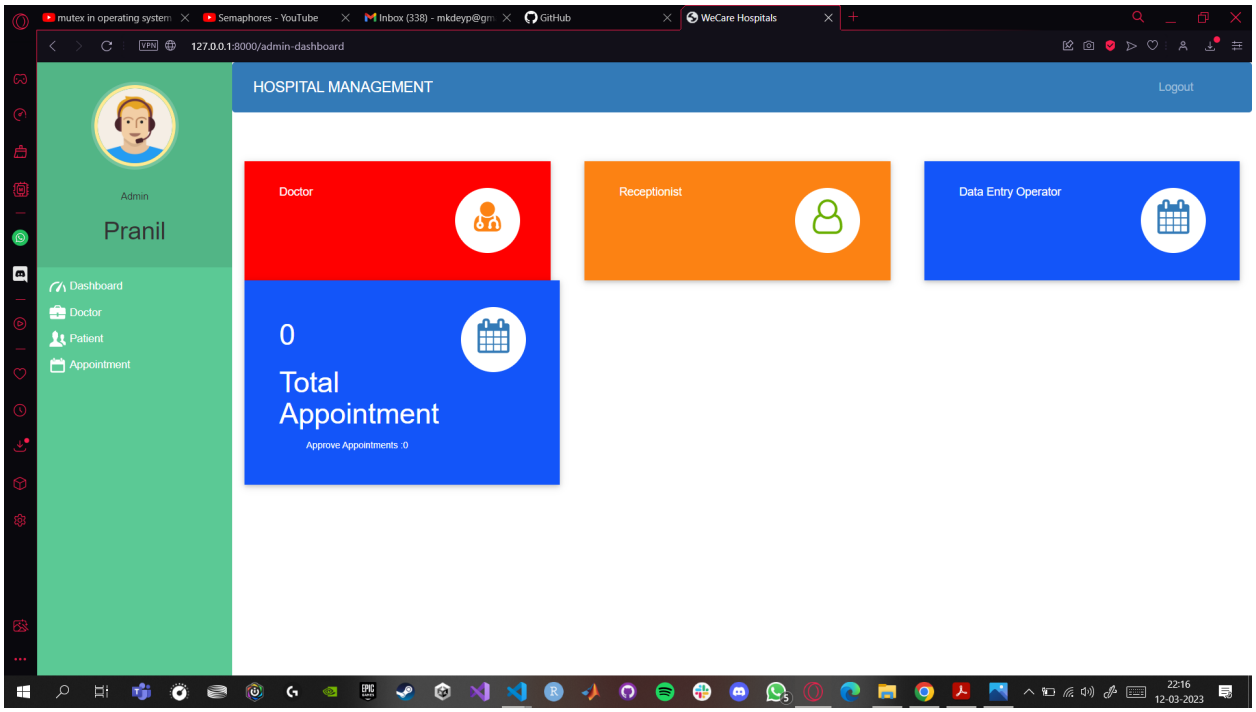
This is the doctor dashboard which is the page where a doctor of WeCare Hospitals can access the info of the patients that are under them. Doctors can also access and delete/cancel their upcoming appointments with patients.



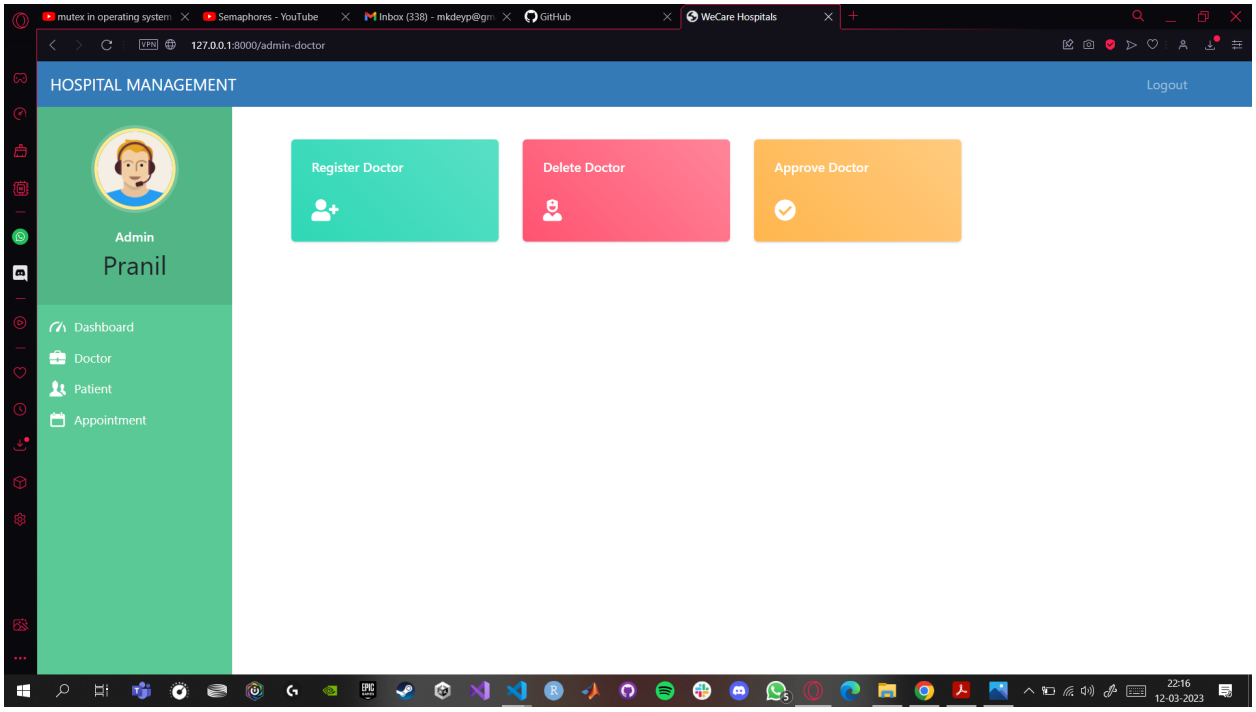
4) Database Administrators:



- Add/delete users

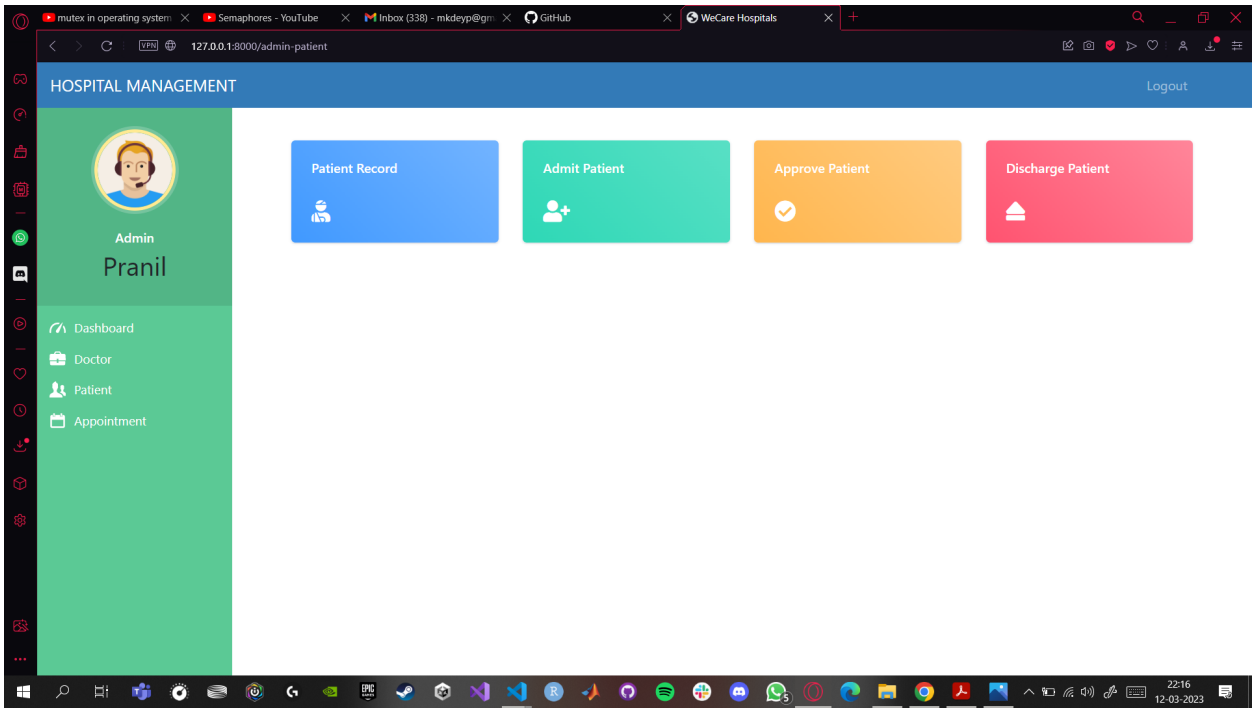


This is the admin dashboard which implements the functionality of a database administrator who can certify which doctor and nurse/data entry operator can be registered. They have control over user addition/deletion.



This is the doctor page in admin where the database admin can register,delete or approve a doctor in the healthcare system.

The admin can also control the information of any patient and can see all details,admit,approve and discharge them as shown below.


mutex in operating system
Semaphores - YouTube
Inbox (338) - mkdeyp@gm
GitHub
WeCare Hospitals
127.0.0.1:8000/admin-patient
HOSPITAL MANAGEMENT
Logout
Admin
Pranil
Dashboard
Doctor
Patient
Appointment
Patient Record
Admit Patient
Approve Patient
Discharge Patient
22:16
12-03-2023