**DATABASE MANAGEMENT**

**SYSTEMS LAB**

**Assignment - 4**

**Hospital Management System**

**Team members**

Nikhil Saraswat (20CS10039)

Vivek Jaiswal (20CS10077)

Shah Dhruv Rajendrabhai (20CS10088)

Amit Kumar (20CS30003)

Jay Kumar Thakur (20CS30024)

**INTRODUCTION**

Hospitals require effective software management systems to make sure they deliver high-quality healthcare services to patients in today's fast-paced, technologically advanced environment. The usage of web-based software programmes is essential for hospital administration because it promotes effective bookkeeping and facilitates seamless communication and collaboration between healthcare professionals, patients, hospital personnel, and other stakeholders. The functions of a web-based hospital management system should include patient registration, appointment, treatment, and test scheduling, patient admission and discharge, management of electronic medical records, display of patients' health records to the treating doctor(s), and storing general information about all the doctors, staff members, etc. To support the hospital's expansion, the software has to be scalable, secure, and user-friendly.

In general, a web-based hospital management system is an important tool that may aid hospitals in providing better medical treatment, increasing productivity, and improving patient outcomes. We have created a web-based software application for a hospital administration system as a result of this project. We made every attempt to include all necessary features and create a comprehensive, dynamic, and user-friendly application.

DATABASE SCHEMAS

1. User Schema:

CREATE TABLE User (

    userID INT NOT NULL,

    email VARCHAR(50) NOT NULL,

    role ENUM('Admin', 'DataEntry', 'FrontDesk', 'Doctor'),

    pass VARCHAR(50) NOT NULL,

    PRIMARY KEY (userID)

);

* All the four users have common user table which are distinguished by **role** attribute.
* Here **userID** attribute is the primary key. We can also make email as primary key but as we are storing json-web-token in the front-end storage so anyone can access it so we are storing only **userID** in json-web-token to prevent access of personal details of user.
* The Admin **role** represents a system/database administrator for the hospital. The Admin can register (add new entities) new (or delete existing) doctors, front-desk, and data-entry operators. This ensures that no doctor or front-desk/data entry operator can temper with details of other such entities.
* The Doctor **role** represents the doctors affiliated with the hospital. Additional Information of the doctor is stored in **Doctor Schema**.
* The Front Desk **role** represent the front-desk operator for the hospital. They are the first point of contact for patients and visitors at the hospital’s reception/front desk. A front-desk operator is responsible for patient registration, scheduling patient appointment, handling patient’s admission/discharge, and having access to information about all patients.
* The Data Entry **role** represents data entry operator who is in charge of updating the results of such tests, treatments and appointments when they are finished, and entering patients' medical records in accordance with specifications.

1. Doctor Schema:

CREATE TABLE Doctor(

    doctorID INT NOT NULL,

    Name VARCHAR(50) NOT NULL,

    Department VARCHAR(50) NOT NULL,

    Date\_of\_joining DateTIme NOT NULL,

    Position VARCHAR(50) NOT NULL,

    Gender ENUM('Male', 'Female', 'Prefer Not To Say') NOT NULL,

    PRIMARY KEY (doctorID),

    FOREIGN KEY (doctorID) references User(userID)

);

* Apart from User Information, Doctors have additional information like Name, Department, Position etc. This information are stored in Doctor Schema.
* Here **doctorID** attribute is the primary key. **doctorID** is also foreign key from **User Schema.**

1. Patient Schema:

CREATE TABLE Patient(

    patientID INT NOT NULL AUTO\_INCREMENT,

    Name VARCHAR(50) NOT NULL,

    DOB DateTime NOT NULL,

    Gender ENUM('Male', 'Female', 'Prefer Not To Say') NOT NULL,

    Address VARCHAR(250) NOT NULL,

    Phone VARCHAR(10)NOT NULL,

    email VARCHAR(50) NOT NULL,

    Registration\_Date DateTime NOT NULL,

    PRIMARY KEY (PatientID)

);

* The patient entity stands in for a person who is a hospital patient seeking care. The front desk staff member can help a patient arrange a doctor's visit, schedule tests and treatments, and have the data entry staff member update his medical records. A patient is not, however, a user of the programme. It represents a passive entity for storing a patient's information.
* Here **patientID** attribute is the primary key which gets auto incremented.

1. Room Schema:

CREATE TABLE Room(

    roomNumber INT NOT NULL,

    roomType ENUM('General', 'Ventilator', 'ICU') NOT NULL,

    maxCapacity INT NOT NULL,

    currentOccupancy INT NOT NULL,

    PRIMARY KEY (roomNumber)

);

* The room entity represents a potential hospital room for an admitted patient, such as an ICU, general ward, ventilator ward, single rooms, etc.
* Each of these rooms has a distinct **roomNumber** (also the primary key).

1. Stay Schema:

CREATE TABLE Stay(

    stayID INT NOT NULL AUTO\_INCREMENT,

    patientID INT NOT NULL,

    roomNumber INT NOT NULL,

    AdmitDate DateTIme NOT NULL,

    DischargeDate DateTIme,

    PRIMARY KEY (stayID),

    FOREIGN KEY (patientID) references Patient(patientID),

    FOREIGN KEY (roomNumber) references Room(roomNumber)

);

* The stay entity stores information about admit date and discharge date of the patient and in which room he/she was stayed.
* Each stay has a unique attribute **stayID** which is auto incremented and is the primary key. **patientID** is the foreign key reference to **patientID** of **Patient** **Schema** and **roomNumber** is a foreign key reference to **roomNumber** of **Room** **Schema**.

1. Appointment Schema:

CREATE TABLE Appointment(

    appointmentID INT NOT NULL AUTO\_INCREMENT,

    patientID INT NOT NULL,

    doctorID INT NOT NULL,

    Date DateTIme NOT NULL,

    priority INT NOT NULL,

    Reason VARCHAR(250) NOT NULL,

    Status ENUM('Pending', 'Completed', 'Cancelled') NOT NULL,

    PRIMARY KEY (appointmentID),

    FOREIGN KEY (patientID) references Patient(patientID),

    FOREIGN KEY (doctorID) references Doctor(doctorID)

);

* The appointment entity stands for a patient's scheduled appointment with a doctor. **patientID** and **doctorID** are used to identify the patient and doctor, respectively. The **Date** property keeps the appointment's date and time window (all appointment time slots are 15 mins).
* There are two different appointment scheduling options: regular and emergency. Any doctor will be assigned to a patient who requests an emergency appointment.
* The primary key is the **appointmentID**, while **patientID** is the foreign key reference to **patientID** of **Patient** **Schema** and **doctorID** is a foreign key reference to **doctorID** of **Doctor** **Schema**.

1. Treatment Schema:

CREATE TABLE Treatment(

    treatmentID INT NOT NULL AUTO\_INCREMENT,

    patientID INT NOT NULL,

    doctorID INT NOT NULL,

    treatmentDate DateTIme NOT NULL,

    treatmentStatus ENUM('Pending', 'Completed', 'On Going') NOT NULL,

    Description VARCHAR(250) NOT NULL,

    PRIMARY KEY (treatmentID),

    FOREIGN KEY (patientID) references Patient(patientID),

    FOREIGN KEY (doctorID) references Doctor(doctorID)

);

* The treatment entity stands for a patient's scheduled treatment with a doctor. **patientID** and **doctorID** are used to identify the patient and doctor, respectively. The **treatmentDate** property keeps the treatment's date and time window (all treatment time slots are 1 hour).
* The primary key is the **appointmentID**, while **patientID** is the foreign key reference to **patientID** of **Patient** **Schema** and **doctorID** is a foreign key reference to **doctorID** of **Doctor** **Schema**.

1. Test Schema:

CREATE TABLE Test(

    testID INT NOT NULL AUTO\_INCREMENT,

    patientID INT NOT NULL,

    testCenter VARCHAR(50) NOT NULL,

    testName VARCHAR(50) NOT NULL,

    testDate DateTIme NOT NULL,

    testStatus ENUM('Pending', 'Completed', 'Cancelled') NOT NULL,

    testResult VARCHAR(250),

    PRIMARY KEY (testID),

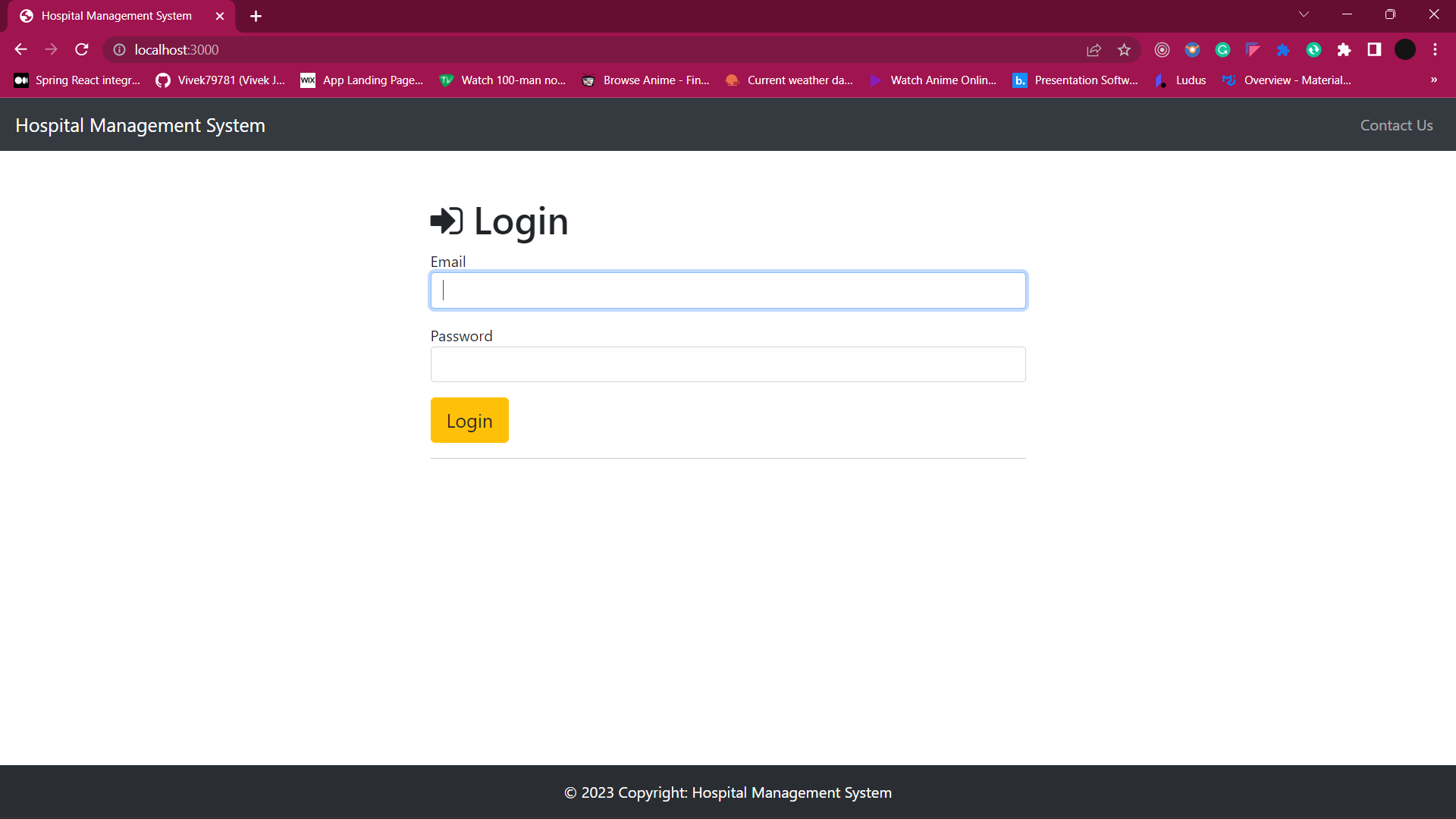
    FOREIGN KEY (patientID) references Patient(patientID)

);

* The tests entity stands for a diagnostic test used in medicine (such as an RT-PCR test, blood test, etc.).
* Additional Attributes are **testCenter** which stores where patient is tested, **testName** which stores all the test name, **testStatus** which stores status of test result, **testResult** which stores result of test, **testDate** which stores date of result and **patientID** which refers which patient has taken the test.
* Its primary key is the **testID**, which is automatically increased while **patientID** is the foreign key reference to **patientID** of **Patient** **Schema.**

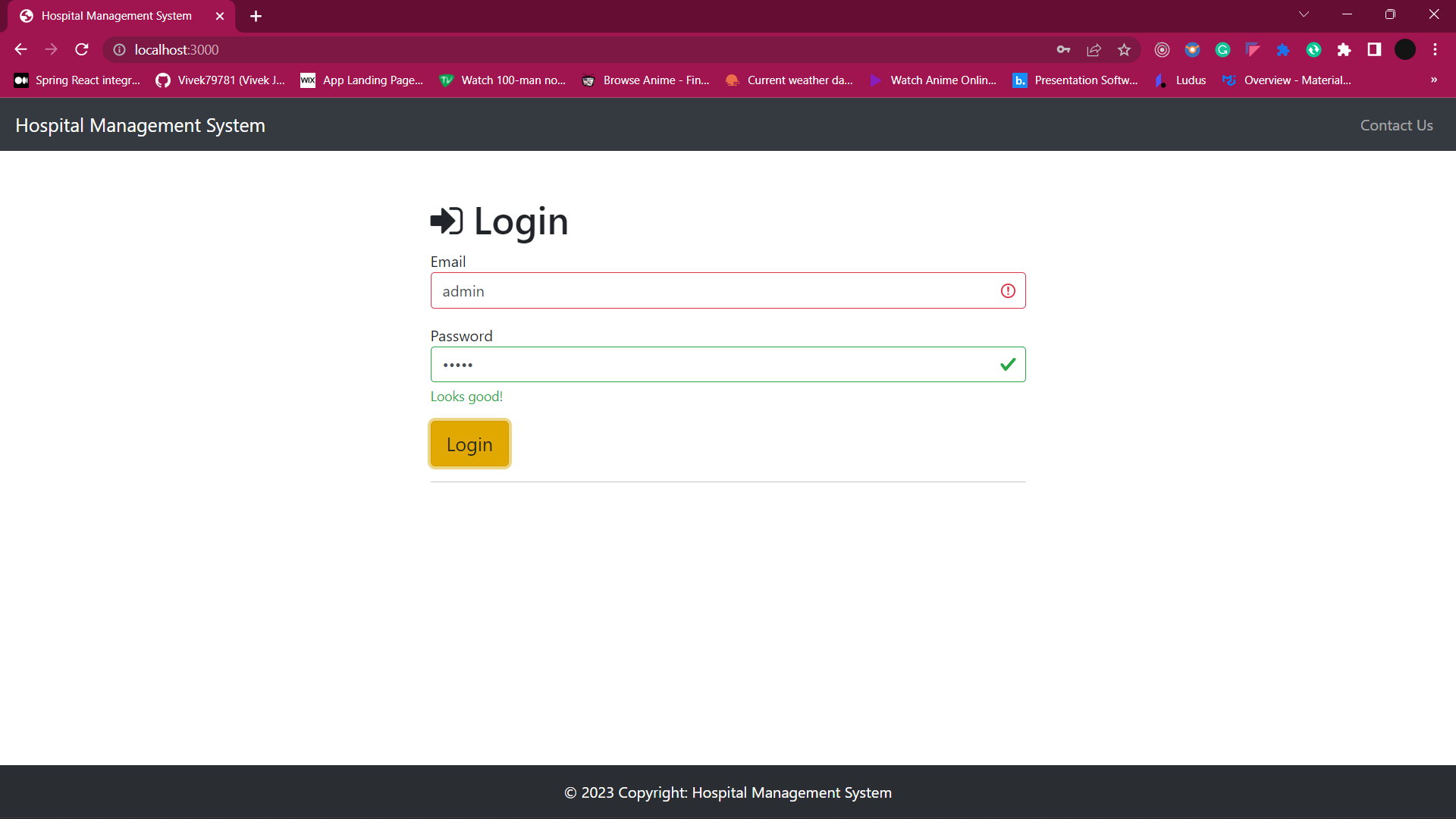
**WEBPAGE USER INTERFACE:**

**HOME PAGE:**

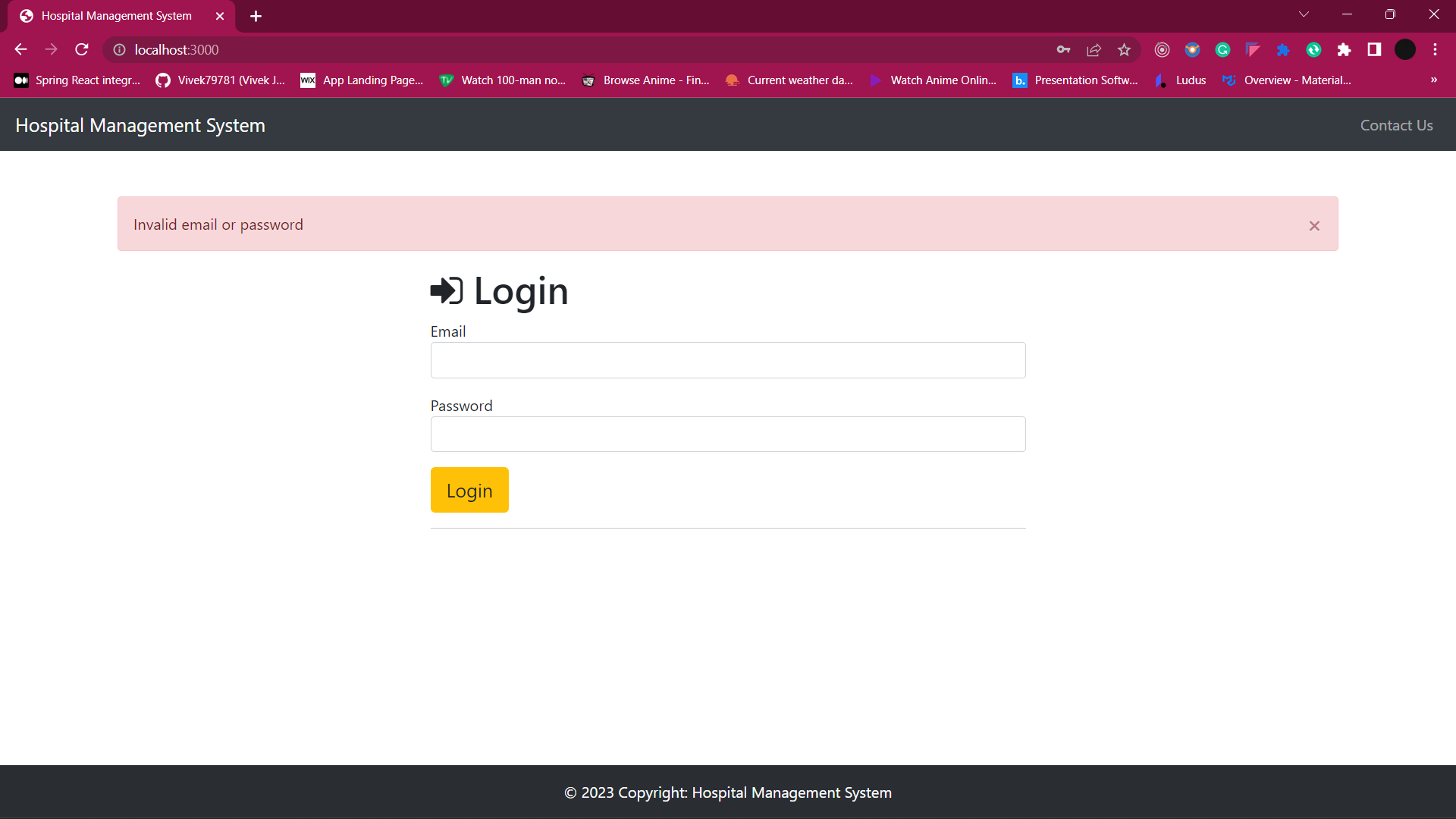


**LOGIN:**

Case 1: Invalid Email Format:

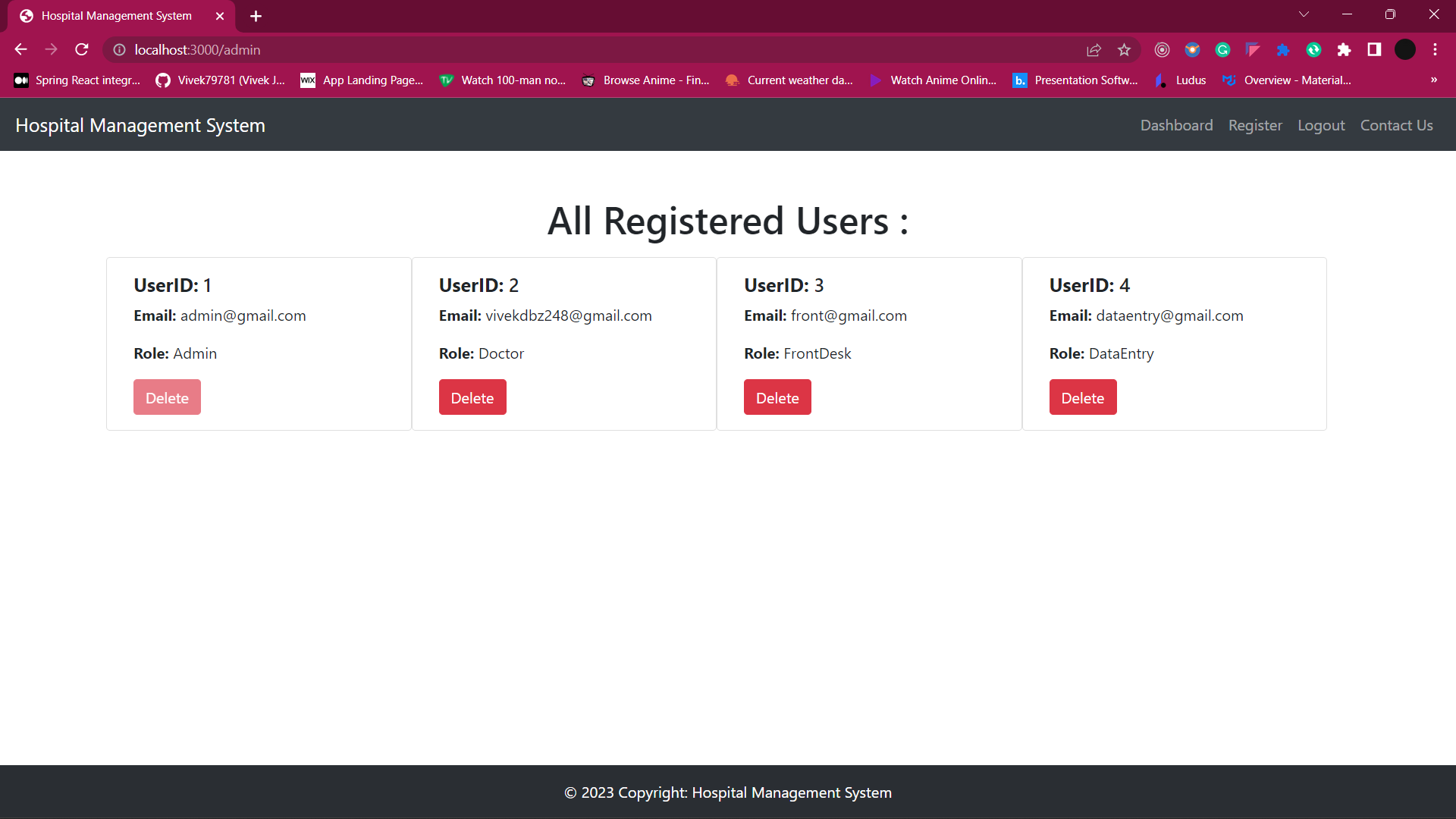


CASE 2: Invalid Email or Password:



CASE 3: ADMIN LOGIN:

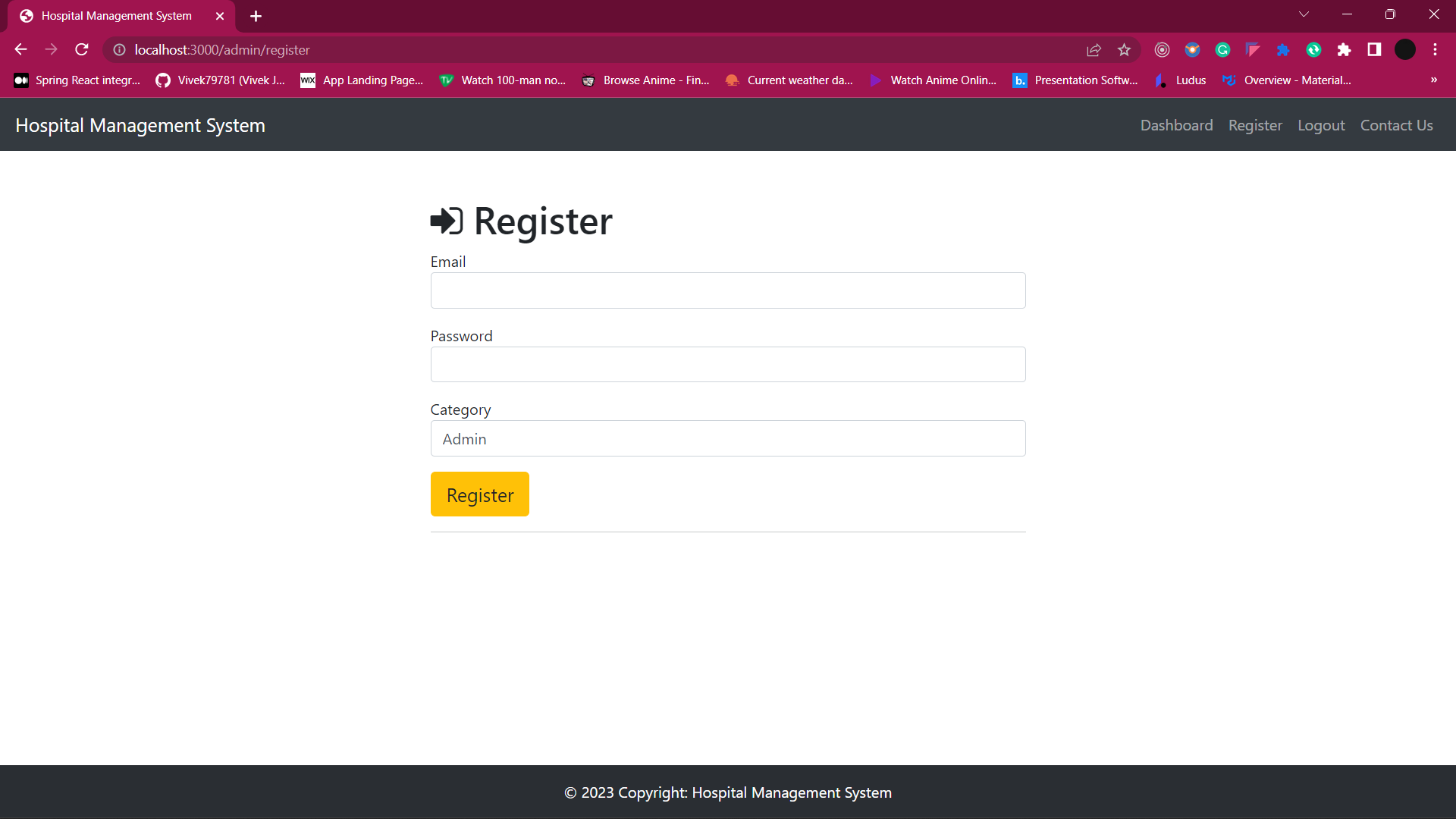
Redirected to ‘/admin’ and showing admin dashboard.



**ADMIN USER:**

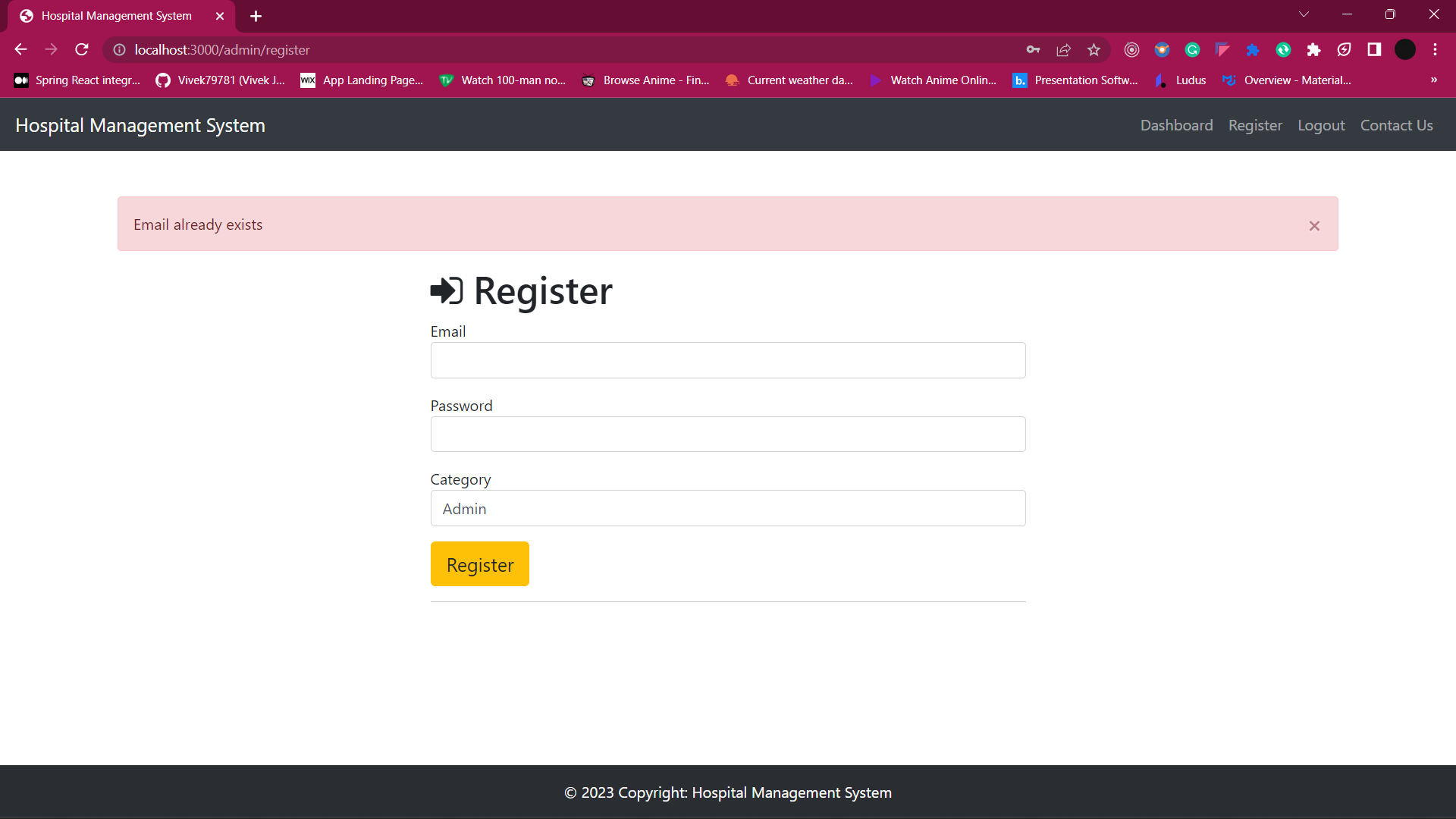
REGISTER AN USER:

Click on the register button on navbar, it will redirect to ‘/admin/register’:

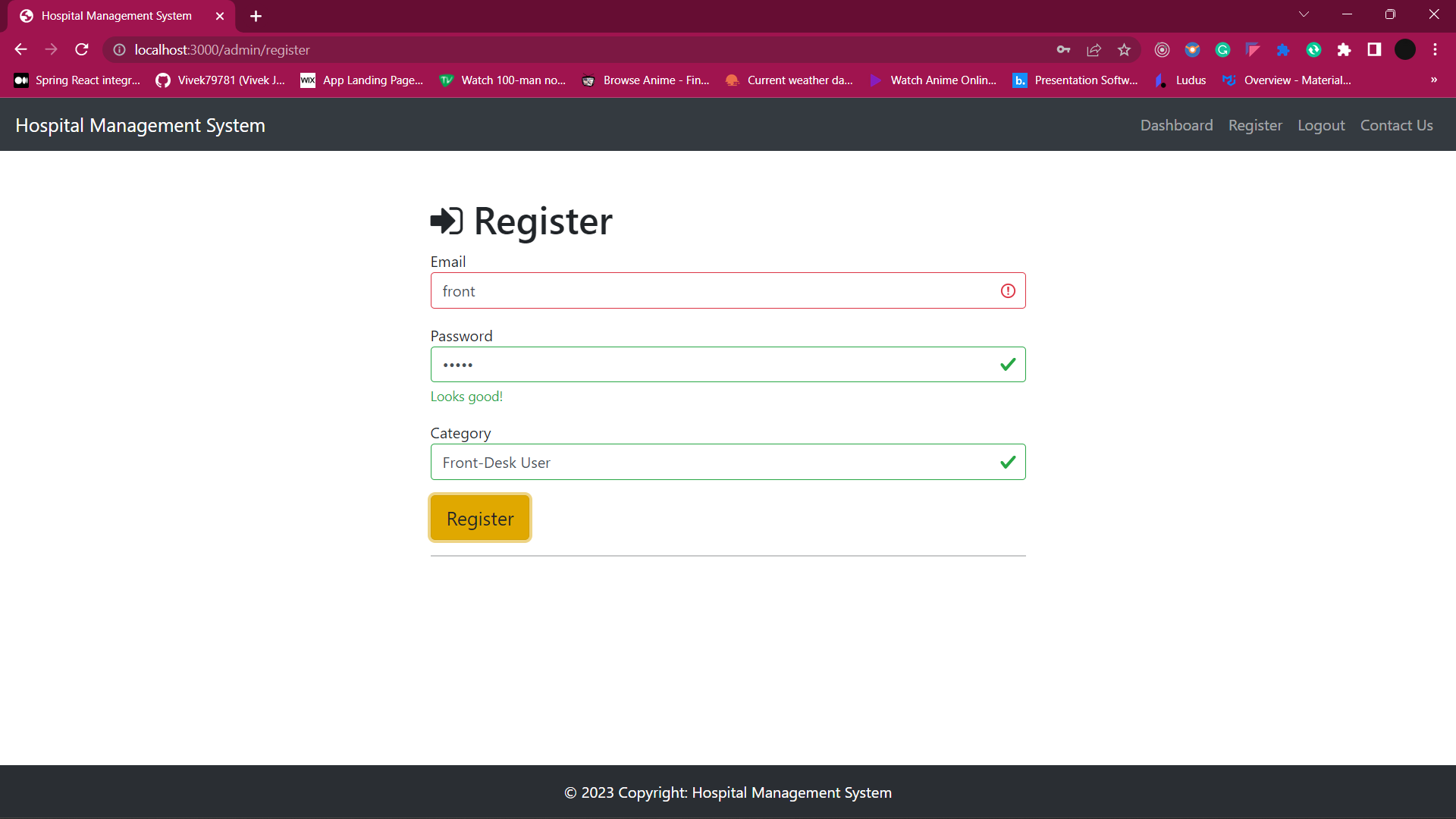


FILL UP CREDENTIALS:

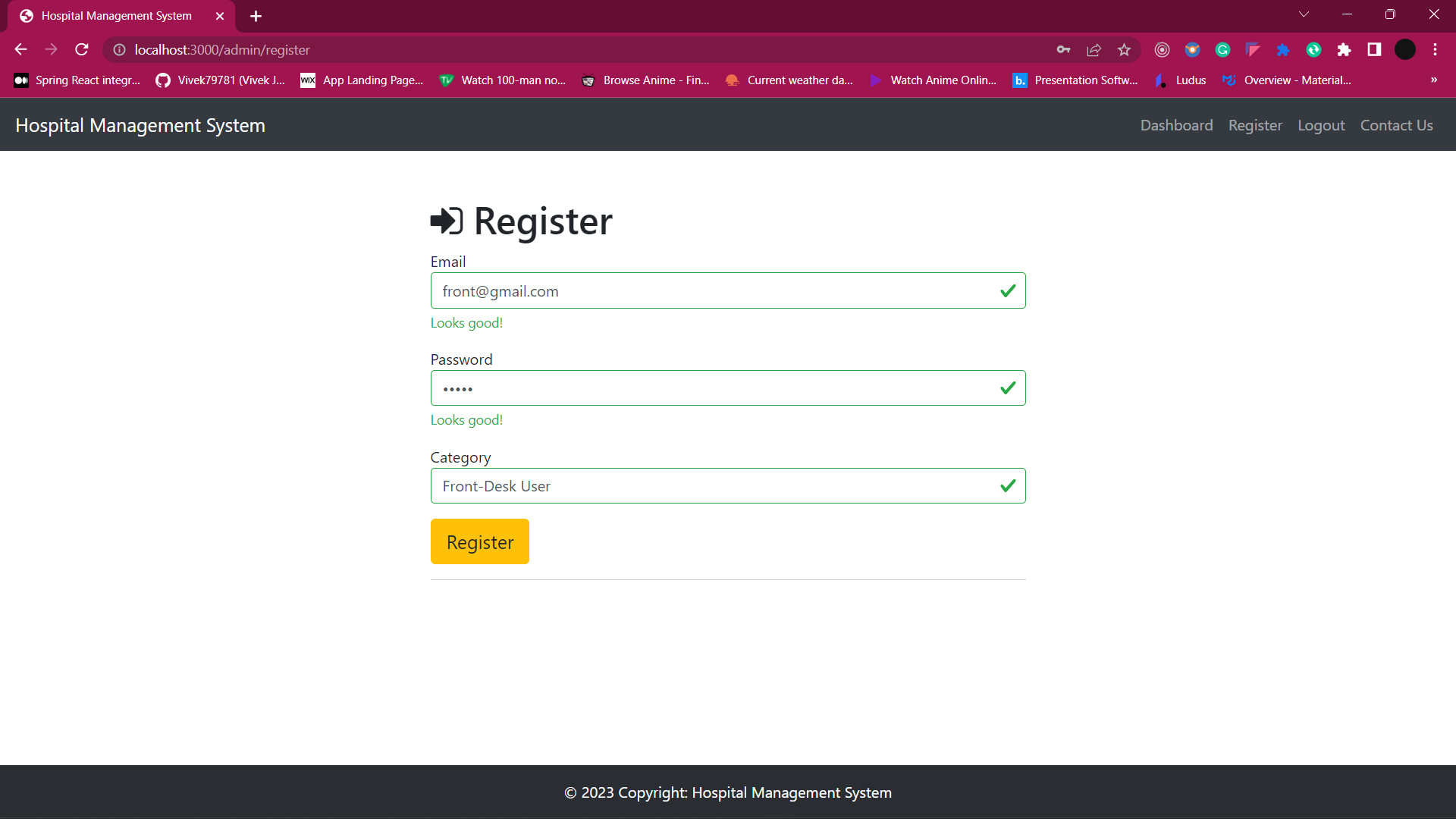
CASE 1: Email Already Registered:



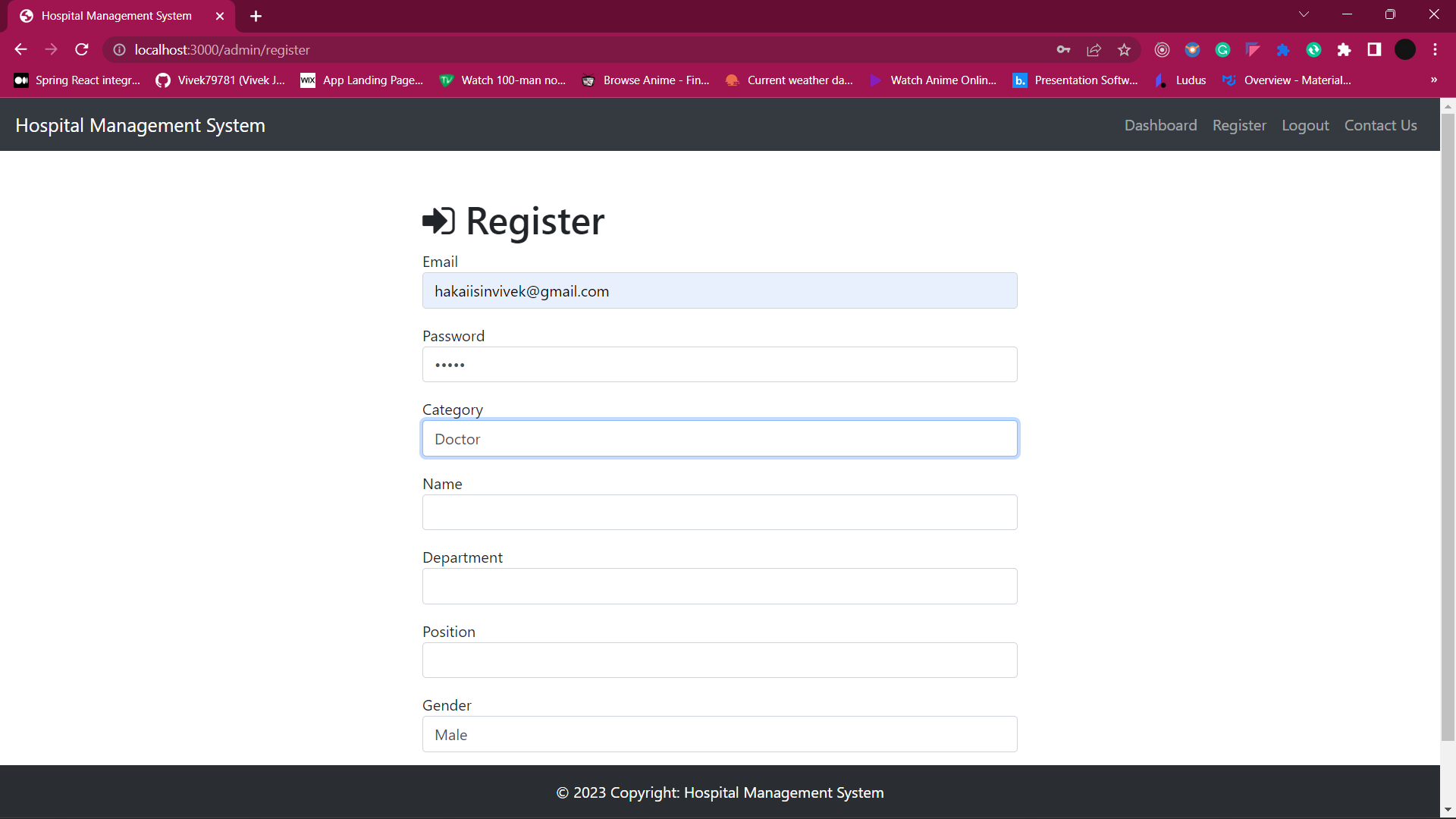
CASE 2: Invalid Credentials Pattern:



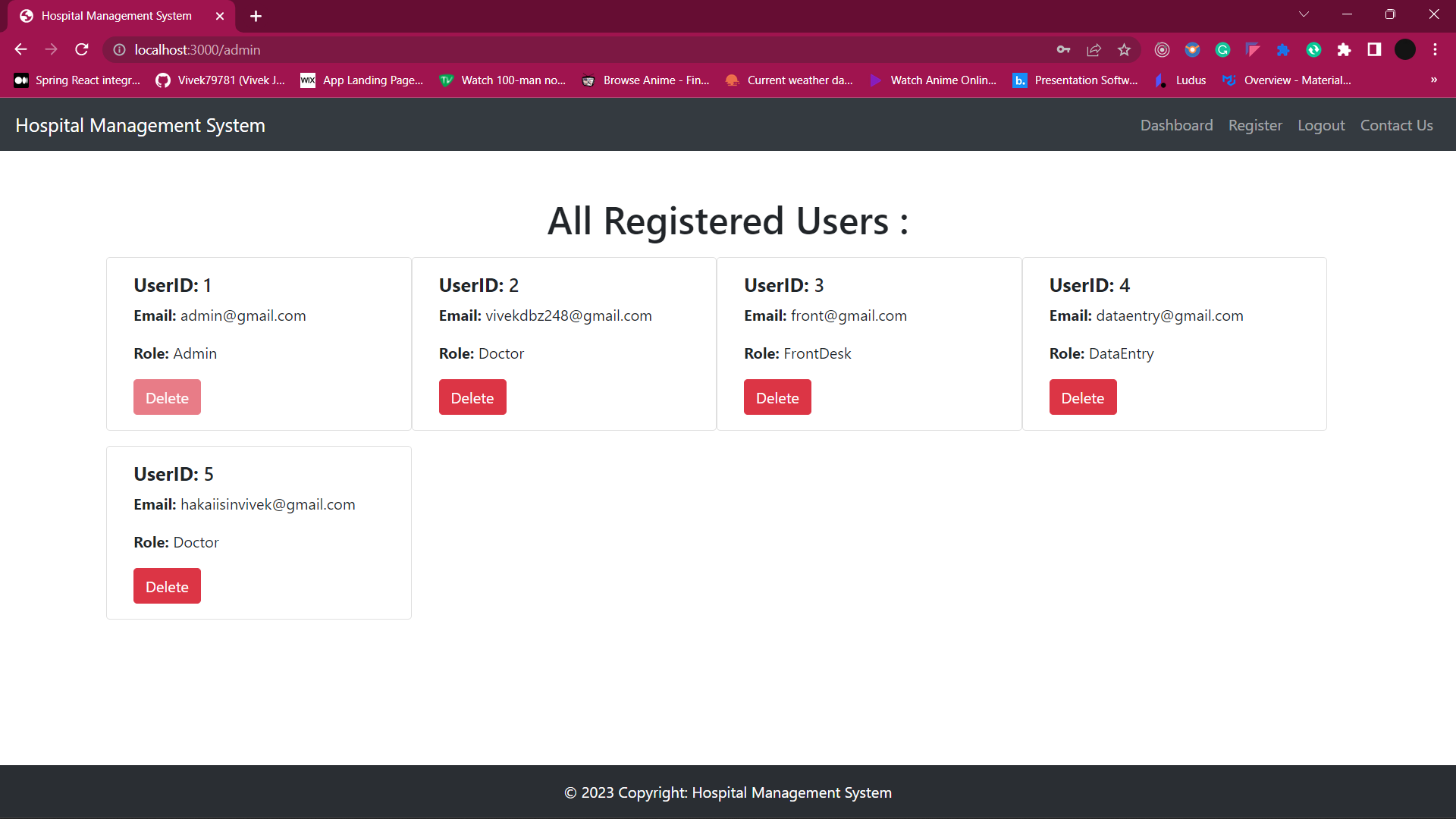
CASE 3: Valid Credentials Pattern:



CASE 4: Doctor Form Expansion:



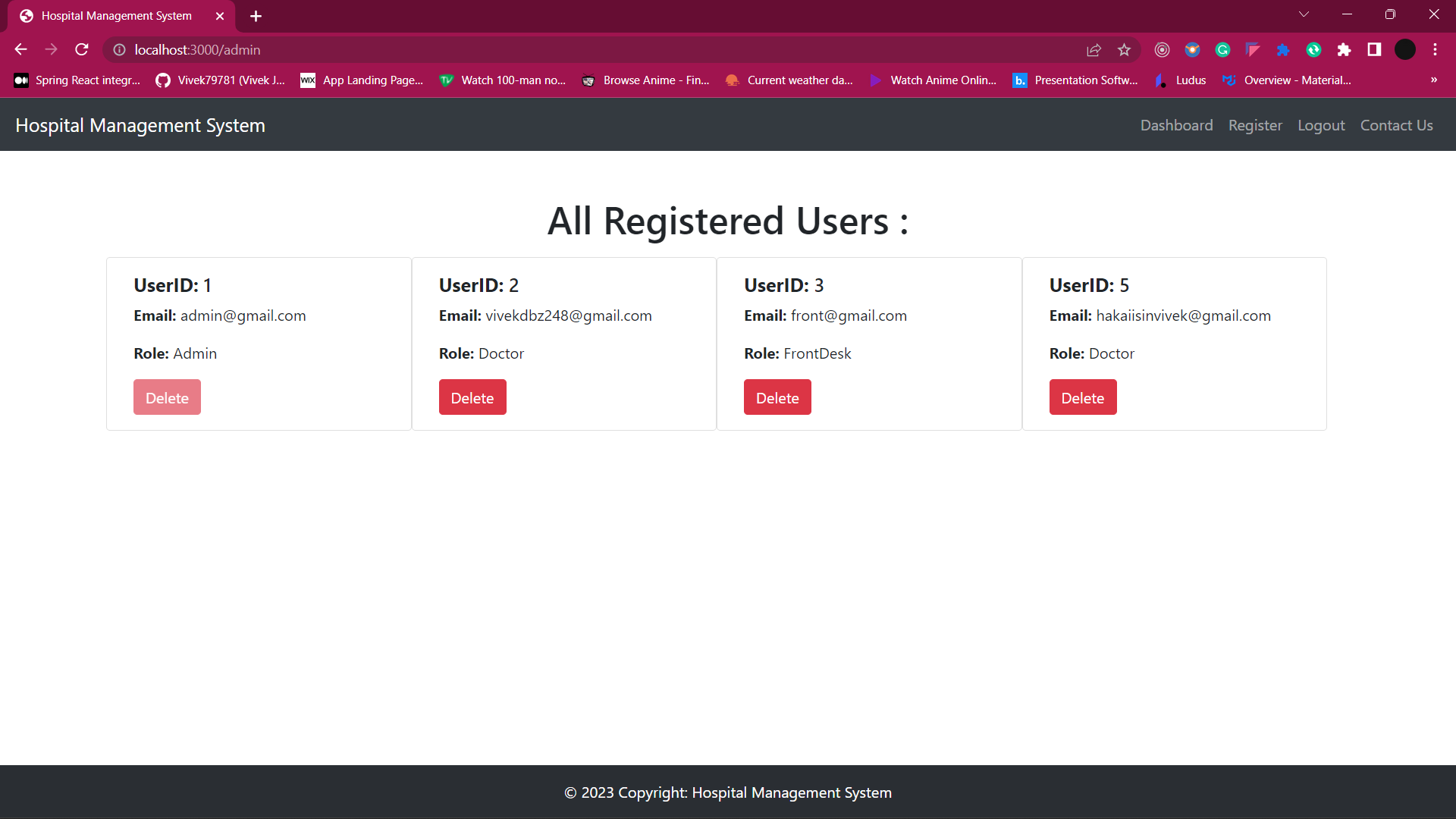
CASE 5: Valid Registration:



DELETE AN USER:

Click on the delete button of the user admin wants to delete.

Here we clicked on delete of userID: 4



USER CAN’T ACCESS ANY UNAUTHORISED PAGE

Example, doctor cannot access any admin route or frontdesk route.

Below is the example of doctor tries to access admin route.

