HospitalPlus Database Project Assignment Outline

1. Project Title

Hospital Management System (HospitalPlus)

2. Project Overview

HospitalPlus is a relational database system designed to manage key operations of a hospital, including appointments, patient records, laboratory tests, doctor assignments, and donor registrations.

3. Conceptual Design

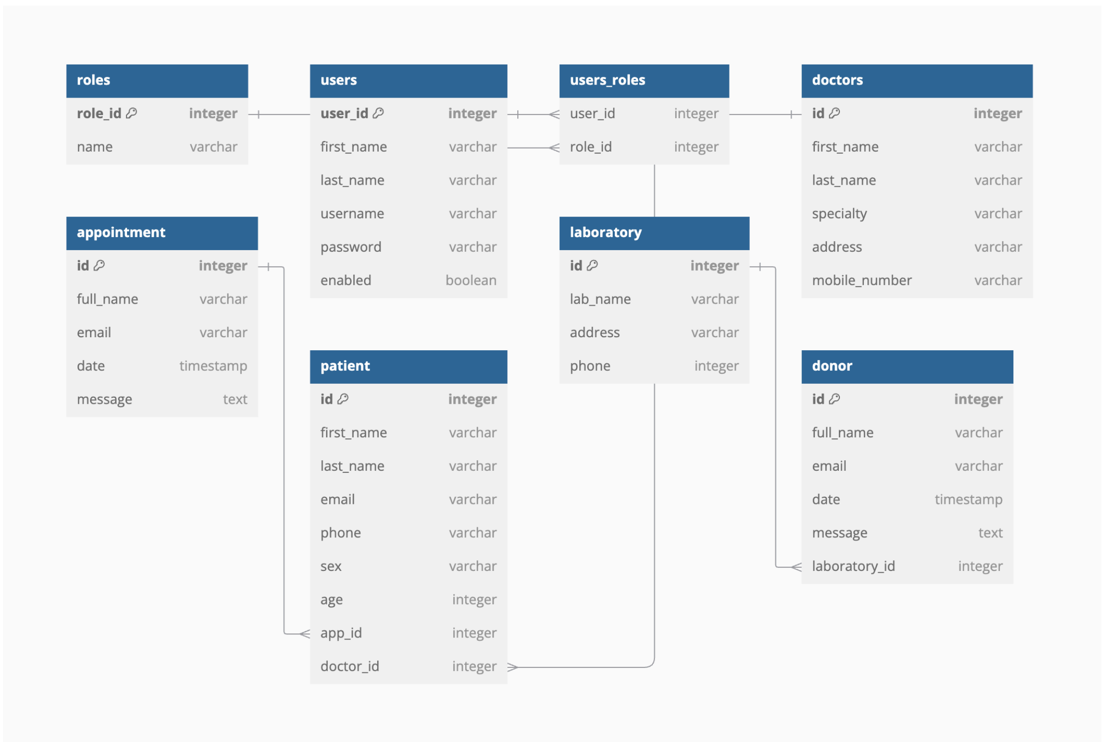
Entity-Relationship (E-R) Model:

Entities:

- **Doctor**: Handles doctor details like specialty and contact.
- **Patient**: Tracks patient demographics and appointment information.
- **Appointment**: Manages scheduling and details of hospital visits.
- **Laboratory**: Maintains laboratory details and donors.
- **Donor**: Stores information about registered donors.
- **User**: Tracks admin and other hospital users for authentication and authorization.

Relationships:

- **Doctor-Patient**: One-to-Many (A doctor can treat multiple patients).
- **Patient-Appointment**: One-to-One (Each patient has a single appointment).
- **Donor-Laboratory**: Many-to-One (A laboratory manages multiple donors).
- **User-Roles**: Many-to-Many (A user can have multiple roles).



4. Logical Design

Relational Model:

Tables:

1. **Users**
2. **Roles**
3. **Users_Roles**
4. **Doctors**
5. **Patients**
6. **Appointments**
7. **Laboratories**
8. **Donors**

Primary and Foreign Keys:

- `user_id` (Primary Key in `Users`)
- `role_id` (Primary Key in `Roles`)
- Foreign keys for relationships in `Users_Roles`, `Patients`, and `Donors`.

5. Data Generation

There is «db.sql» file in the code with sql queries.

6. Implementation

- Used **PostgreSQL** for database implementation.
- SQL scripts to define:
  - Table schemas.
  - Data insertion scripts.
  - Queries for CRUD operations.

7. Query Questions

Examples:

- Retrieve a list of all doctors and their specialties.
- Find patients with appointments scheduled this week.
- List all donors and their associated laboratories.
- Calculate the total number of appointments handled by a specific doctor.
- Identify labs with the highest donor registrations.
- Find patients treated by a specific doctor.
- Retrieve the latest appointments made.
- Calculate hospital revenue (if billing is included).

8. Application Development (Java)

- Use **Spring Boot** for backend REST API development.
- Implement the following CRUD APIs:
  - **Doctors**: Add, edit, delete, and view doctors.
  - **Patients**: Add, edit, delete, and view patients.
  - **Appointments**: Manage appointments.
  - **Laboratories**: Manage laboratory data.
  - **Donors**: Manage donors.

9. REST API Testing

- Using **Postman** to test the backend APIs.

10. Documentation

1. Relational Model:

```
——————
-- Roles Table
CREATE TABLE roles (
   role_id SERIAL PRIMARY KEY,
   name VARCHAR(50) NOT NULL
);

-- Users Table
CREATE TABLE users (
   user_id SERIAL PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100),
   username VARCHAR(100) UNIQUE NOT NULL,
   password VARCHAR(255) NOT NULL,
   enabled BOOLEAN DEFAULT TRUE
);

-- Users and Roles Relationship Table
CREATE TABLE users_roles (
   user_id INT REFERENCES users(user_id) ON DELETE CASCADE,
   role_id INT REFERENCES roles(role_id) ON DELETE CASCADE,
   PRIMARY KEY (user_id, role_id)
);

-- Doctors Table
CREATE TABLE doctors (
   id SERIAL PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100),
   specialty VARCHAR(100),
   address VARCHAR(255),
   mobile_number VARCHAR(15)
);

-- Appointments Table
CREATE TABLE appointment (
   id SERIAL PRIMARY KEY,
   full_name VARCHAR(100),
   email VARCHAR(100),
   date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   message TEXT
);
```

```
-- Patients Table
CREATE TABLE patient (
   id SERIAL PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100),
   email VARCHAR(100),
   phone VARCHAR(15),
   sex VARCHAR(10),
   age INT,
   app_id INT REFERENCES appointment(id) ON DELETE SET NULL,
   doctor_id INT REFERENCES doctors(id) ON DELETE CASCADE
);

-- Laboratories Table
CREATE TABLE laboratory (
   id SERIAL PRIMARY KEY,
   lab_name VARCHAR(100),
   address VARCHAR(255),
   phone INT
);

-- Donors Table
CREATE TABLE donor (
   id SERIAL PRIMARY KEY,
   full_name VARCHAR(100),
   email VARCHAR(100),
   date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   message TEXT,
   laboratory_id INT REFERENCES laboratory(id) ON DELETE SET NULL
);
```
——————

2. SQL Scripts for Schema Creation and Sample Data

——————

-- Roles

INSERT INTO roles (role_id, name) VALUES (1, 'ADMIN');

INSERT INTO roles (role_id, name) VALUES (2, 'DOCTOR');

-- Users

INSERT INTO users (first_name, last_name, username, password, enabled)

VALUES ('Admin', 'Adminovich', 'admin', 'hashed_password_here', TRUE);

```
-- Users and Roles

INSERT INTO users_roles (user_id, role_id) VALUES (1, 1);


-- Doctors

INSERT INTO doctors (first_name, last_name, specialty, address, mobile_number)

VALUES ('John', 'Doe', 'Cardiologist', '123 Main St', '1234567890');


-- Appointments

INSERT INTO appointment (full_name, email, message)

VALUES ('Jane Smith', 'jane.smith@example.com', 'Checkup');


-- Patients

INSERT INTO patient (first_name, last_name, email, phone, sex, age, app_id, doctor_id)

VALUES ('Alice', 'Johnson', 'alice.j@example.com', '9876543210', 'Female', 30, 1, 1);


-- Laboratories

INSERT INTO laboratory (lab_name, address, phone)

VALUES ('Central Lab', '456 Elm St', 12345);


-- Donors

INSERT INTO donor (full_name, email, message, laboratory_id)

VALUES ('Bob Brown', 'bob.brown@example.com', 'Blood Donation', 1);
```

———————

3. API Documentation

API Endpoints

Users

- **POST** `/api/users`
    - Create a new user.
    - **Request Body:**
      `{`
    - `    "firstName": "Isi",`
    - `    "lastName": "Koichubaev",`
    - `    "username": "isi1",`
    - `    "password": "password"`
    - `}`
    - **Response:**
      `{`
    - `    "id": 1,`
    - `    "username": "isi1",`
    - `    "enabled": true`
    - `}`

Doctors

- **GET** `/api/doctors`
    - Retrieve all doctors.
    - **Response:**
      `[`
    - `    {`
    - `        "id": 1,`
    - `        "firstName": "Isi",`
    - `        "lastName": "Isi",`
    - `        "specialty": "Cardiologist"`
    - `    }`
    - `]`
- **POST** `/api/doctors`
    - Add a new doctor.
    - **Request Body:**
      `{`
    - `    "firstName": "John",`
    - `    "lastName": "Doe",`
    - `    "specialty": "Cardiologist"`
    - `}`

Patients

- **GET** `/api/patients`
    - Retrieve all patients.
- **POST** `/api/patients`
    - Add a new patient.
    - **Request Body:**

```
{
    "firstName": "Alice",
    "lastName": "Johnson",
    "doctorId": 1
}
```

Laboratories

- **GET** `/api/laboratories`
    - Retrieve all labs.
- **POST** `/api/laboratories`
    - Add a new lab.

Donors

- **GET** `/api/donors`
    - Retrieve all donors.
- **POST** `/api/donors`
    - Add a new donor.
    - **Request Body:**

```
{
    "fullName": "Bob Brown",
    "email": "bob.brown@example.com",
    "laboratoryId": 1
}
```

-