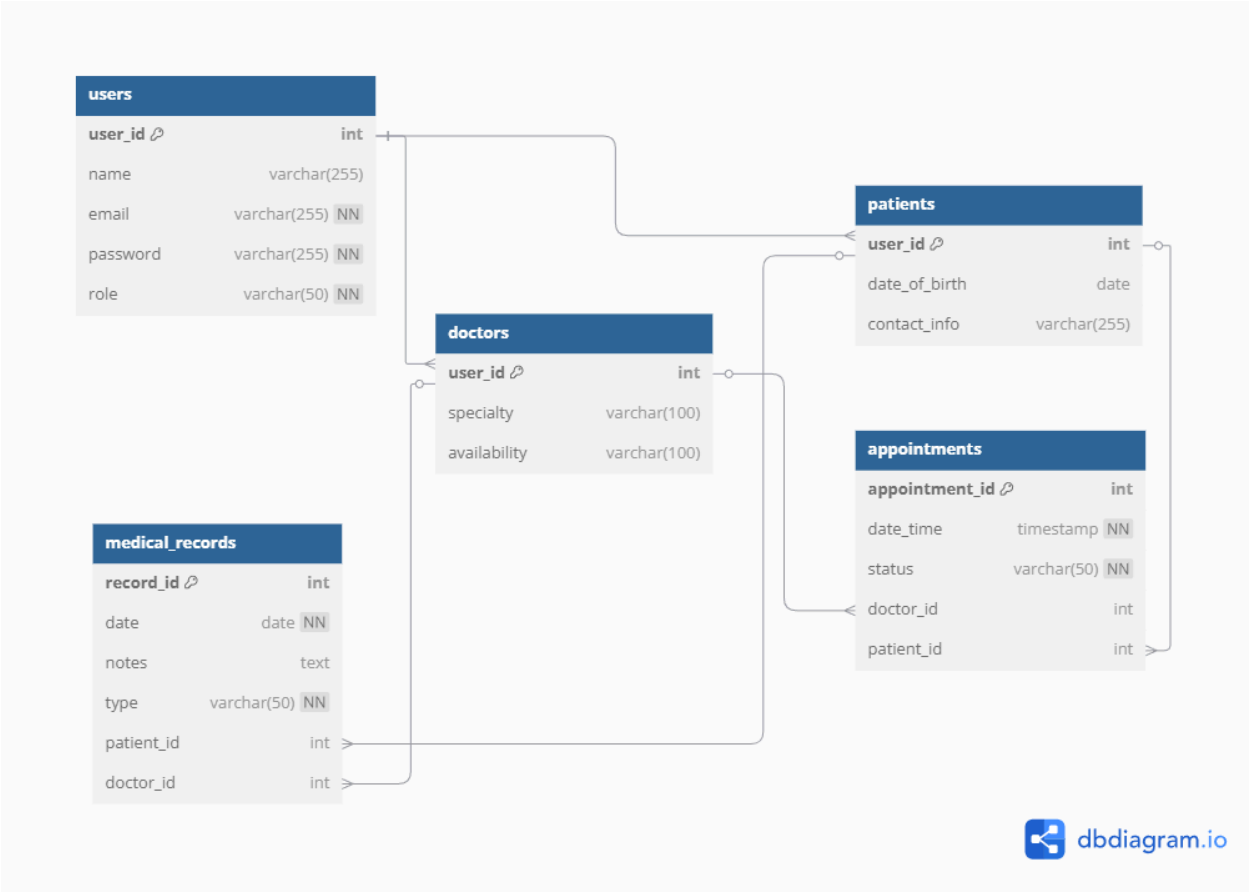
Project Architecture Documentation

Overview

The **Smart Healthcare Web App** is a modern web application designed to enhance healthcare accessibility by allowing users to book medical appointments, access healthcare services, and track their medical history securely. The project follows a **modular architecture**, ensuring scalability, maintainability, and security.

Entity Relationship Diagram (ERD)

Below is the **Entity Relationship Diagram (ERD)** representing the database structure:



Database Entities and Relationships

1. **Users (`users` table)**
    - **user_id (int, PK)**: Unique identifier for each user.
    - **name (varchar(255))**: Full name of the user.
    - **email (varchar(255), NN, Unique)**: Unique email address for login.
    - **password (varchar(255), NN)**: Encrypted password for authentication.
    - **role (varchar(50))**: Specifies whether the user is a doctor or a patient.

2. **Doctors (`doctors` table)**
    - **user_id (int, PK, FK to users.user_id)**: Unique identifier for the doctor, linked to `users`.
    - **specialty (varchar(100))**: The doctor's area of expertise.
    - **availability (varchar(100))**: The available working hours of the doctor.

3. **Patients (`patients` table)**
    - **user_id (int, PK, FK to users.user_id)**: Unique identifier for the patient, linked to `users`.
    - **date_of_birth (date)**: Patient's date of birth.
    - **contact_info (varchar(255))**: Contact details for the patient.

4. **Appointments (`appointments` table)**
    - **appointment_id (int, PK)**: Unique identifier for each appointment.
    - **date_time (timestamp, NN)**: Date and time of the appointment.
    - **status (varchar(50), NN)**: Status of the appointment (e.g., scheduled, completed, canceled).
    - **doctor_id (int, FK to doctors.user_id)**: Links the appointment to a specific doctor.
    - **patient_id (int, FK to patients.user_id)**: Links the appointment to a specific patient.

5. **Medical Records (`medical_records` table)**

    - **record_id (int, PK)**: Unique identifier for the medical record.
    - **date (date, NN)**: Date when the record was created.
    - **notes (text)**: Medical notes added by the doctor.
    - **type (varchar(50), NN)**: Type of record (e.g., diagnosis, prescription, test results).
    - **patient_id (int, FK to patients.user_id)**: Links the record to a patient.
    - **doctor_id (int, FK to doctors.user_id)**: Links the record to the doctor who created it.

Relationships Between Entities

- A **user** can be either a **doctor** or a **patient**, identified by the `role` column in the `users` table.
- Each **doctor** is linked to the `users` table through `user_id`.
- Each **patient** is linked to the `users` table through `user_id`.
- A **doctor** can have multiple **appointments**, and each appointment is associated with one **patient**.
- A **patient** can have multiple **appointments**, and each appointment is associated with one **doctor**.
- A **doctor** can create multiple **medical records** for different patients.
- A **patient** can have multiple **medical records**, each linked to a specific **doctor**.

Project Structure

The project follows a **structured MVC (Model-View-Controller)** architecture for clarity and maintainability. Below is the directory structure:

```
📦 Smart-Healthcare-Web-App
├── 📂 backend              # Backend-related files
│   ├── 📂 controllers      # API controllers
│   ├── 📂 models           # Database models (Prisma/Sequelize)
│   ├── 📂 routes           # API endpoints
│   ├── 📂 prisma           # Prisma setup (if applicable)
│   ├── 📂 node_modules     # Dependencies (ignored by Git)
│   ├── .env                # Environment variables (ignored by Git)
│   ├── index.js            # Backend entry point
│   ├── package.json        # Dependencies & scripts
│   └── package-lock.json   # Dependency lock file
├── .gitignore              # Global ignore file (recommended)
├── README.md               # Main project documentation
```

Breakdown of Key Components

- **Controllers (`controllers/`)**: Handles API logic, processing requests, and returning responses.
- **Models (`models/`)**: Defines database schema and relationships.
- **Routes (`routes/`)**: Defines RESTful API endpoints and connects them to controllers.
- **Prisma (`prisma/`)**: ORM configuration (only if Prisma is used).
- **Environment Variables (`.env`)**: Stores sensitive configuration data.
- **Entry Point (`index.js`)**: Initializes the server and connects routes.

Technology Stack

- **Backend:** Node.js, Express.js
- **Database:** PostgreSQL
- **ORM:** Sequelize or Prisma
- **Authentication:** JWT for user authentication
- **API Documentation & Testing:** Swagger / Postman
- **Version Control:** Git, GitHub/GitLab