# HealthCare Dashboard

## Overview

The HealthCare Dashboard is a full-fledged web application designed to improve the management of healthcare services, particularly focusing on the interaction between doctors and patients. The platform provides an intuitive interface for users to manage profiles, view and interact with healthcare records, and streamline communication within the healthcare system. The application is built using modern web technologies, ensuring both performance and scalability.

## Features

### 1. ****User Management****

* **Doctor Sign-Up & Login**: Doctors can create accounts, log in securely, and manage their profiles. The system provides them with a dashboard to view and manage their patients' data effectively.
* **Patient Sign-Up & Login**: Patients can register, log in, and access a personalized dashboard where they can view their health records, manage appointments, and interact with their doctors.

### 2. ****Dashboard Views****

* **Doctor Dashboard**: Doctors have access to a comprehensive dashboard where they can manage patient records, view scheduled appointments, and update their profiles.
* **Patient Dashboard**: Patients can view their medical history, upcoming appointments, and communicate with their doctors through their dashboard.

### 3. ****QR Code Integration****

* The platform generates unique QR codes for patients, which doctors can scan to quickly access the patient's health records. This feature enhances the security and efficiency of accessing patient data.

### 4. ****Responsive Design****

* The application is designed to be fully responsive, ensuring it functions smoothly on desktops, tablets, and mobile devices. This guarantees accessibility across various platforms.

### 5. ****Secure Authentication****

* The system uses robust authentication mechanisms to ensure that all user data is secure. This includes encrypted passwords, secure sessions, and protected routes.

## Technology Stack

### ****Frontend****

* **HTML/CSS**: The backbone for structuring and styling the web pages.
* **JavaScript**: Adds interactivity and dynamic content to the web pages.
* **Bootstrap**: A popular framework used for creating responsive and mobile-first designs.

### ****Backend****

* **Node.js**: A powerful server-side platform used for building the backend of the application.
* **Express.js**: A minimalist web framework for Node.js that simplifies routing and middleware management.

### ****Database****

* **MongoDB**: A NoSQL database used for storing user information, health records, and other essential data in a scalable and flexible manner.

## Installation

### Prerequisites

Ensure that Node.js and MongoDB are installed on your system.

### Steps to Set Up the Project

1. **Clone the Repository**:

git clone https://github.com/JatinBB/HealthCare-Dashboard.git

cd HealthCare-Dashboard

1. **Install Dependencies**:

npm install

1. **Configure the Database**:
   * Ensure MongoDB is running on your machine.
   * Set up your connection string in the .env file.
2. **Run Migrations**:
   * If the project includes any database migrations, run them to prepare your database schema.
3. **Start the Application**:

npm start

1. **Access the Application**:
   * Open your web browser and navigate to http://localhost:3000.

## Project Structure and File Explanations

Here's a breakdown of the main directories and files in the HealthCare Dashboard project:

HealthCare-Dashboard/

│

├── public/

│ ├── css/ # Stylesheets and design elements

│ ├── js/ # JavaScript files for front-end functionality

│ ├── images/ # Static images used in the application

│ └── index.html # Entry point for the web application

│

├── src/

│ ├── controllers/ # Handles request logic and interactions between models and views

│ │ ├── authController.js # Manages authentication logic for users

│ │ ├── doctorController.js # Handles doctor-related operations like profile and patient management

│ │ ├── patientController.js # Manages patient operations such as viewing health records

│ │ └── qrCodeController.js # Generates and manages QR code operations

│ │

│ ├── models/

│ │ ├── Doctor.js # MongoDB model/schema for doctor data

│ │ ├── Patient.js # MongoDB model/schema for patient data

│ │ └── User.js # MongoDB model/schema for generic user data, handling both patients and doctors

│ │

│ ├── routes/

│ │ ├── authRoutes.js # Routes related to authentication (login, sign-up, logout)

│ │ ├── doctorRoutes.js # Routes for doctor-specific operations

│ │ ├── patientRoutes.js # Routes for patient-specific operations

│ │ └── qrCodeRoutes.js # Routes to handle QR code generation and scanning

│ │

│ ├── views/

│ │ ├── doctors/ # EJS templates for doctor-related pages (profile, dashboard, etc.)

│ │ ├── patients/ # EJS templates for patient-related pages (dashboard, records, etc.)

│ │ └── auth/ # EJS templates for authentication pages (login, register)

│ │

│ └── app.js # Main application file, entry point for the Node.js server

│

├── .env # Environment variables for sensitive data like database connections

├── package.json # Node.js project dependencies and scripts

├── README.md # Project documentation

└── ... # Other configuration files

### Explanation of Key Files:

* **app.js**: This is the main application file that initializes the server, sets up middleware, and defines the base routes.
* **authController.js**: Manages user authentication, including login, registration, and logout functionalities.
* **doctorController.js**: Contains the logic for handling all doctor-related operations, such as managing patient data and viewing appointments.
* **patientController.js**: Focuses on patient-related tasks like viewing and updating personal health records.
* **qrCodeController.js**: Generates QR codes for patients and handles the logic for scanning and retrieving data using these codes.
* **Doctor.js, Patient.js, User.js**: These are the database models (schemas) defining the structure of the data stored in MongoDB for doctors, patients, and users, respectively.
* **authRoutes.js, doctorRoutes.js, patientRoutes.js, qrCodeRoutes.js**: These files define the API endpoints (routes) that the frontend will call to perform actions related to authentication, doctor/patient management, and QR code operations.
* **views/**: Contains EJS templates that render the HTML views for the different pages of the application, including user dashboards and login pages.

## Conclusion

The HealthCare Dashboard is a robust platform designed to simplify the management of healthcare data for both doctors and patients. It combines modern web technologies to provide a secure, scalable, and user-friendly interface that addresses the needs of healthcare providers and their patients. With its modular architecture, the application can easily be extended and customized to meet additional requirements or integrate with other systems.

By following the project structure and understanding the purpose of each file and directory, developers can efficiently navigate the codebase, contribute to the project, and implement new features with ease.