Clinic Appointment System Documentation

Overview

The Clinic Appointment System is a web-based application designed to facilitate the booking and management of appointments at a medical clinic. It provides functionalities for users to sign up, log in, book appointments with doctors, view their booked appointments, and for clinic staff to manage appointments.

Features

- User Signup and Login

- Book Appointments

- View Appointments

- Email Notifications for Signup

- Authentication with JWT

- User and Appointment management with MongoDB

Architecture

The system follows a client-server architecture with the following components:

Frontend: Implemented using HTML, CSS, and JavaScript for user interaction.

Backend: Developed with Node.js and Express.js, handling business logic, data storage, and API interactions.

database: MongoDB used for storing user data and appointment information.

Authentication: JWT (JSON Web Token) used for user authentication and session management.

Email Service: Node mailer integrated for sending email notifications.

Setup Instructions

Prerequisites

1. Node.js installed on your machine.

2. MongoDB installed and running locally or accessible remotely.

3. Email account (e.g., Gmail) for sending notification emails.

Installation Steps

1. Clone the repository

git clone https://github.com/your-username/clinic-appointment-system.git

cd clinic-appointment-system

2. Install dependencies

npm install

3. Configure MongoDB

-Open `config/db.js` and set the MongoDB connection URI.

4. Configure Email Service

Open `config/mailer.js` and update the email service settings with your credentials.

5. Start the Server

npm start

This will start the server at `http://localhost:3000`.

6. Access the Application

Open your web browser and navigate to `http://localhost:3000` to access the application.

API Documentation

Authentication

User Signup

Endpoint: POST `/api/signup`

Request Body:

```json

{

"firstName": "John",

"lastName": "Doe",

"gender": "Male",

"dob": "1990-01-01",

"address": "123 Main St, City",

"contactNumber": "123-456-7890",

"email": "john.doe@example.com"

}

```

Response: `{ "success": true }` or `{ "success": false }`

User Login

Endpoint: POST `/api/login`

- \*\*Request Body\*\*:

```json

{

"email": "john.doe@example.com",

"password": "password"

}

```

Response:

```json

{

"success": true,

"token": "JWT token"

}

```

or `{ "success": false }`

Appointments

Book Appointment

Endpoint: POST `/api/book-appointment`

Request Body:

```json

{

"appointmentDate": "2024-07-01",

"time": "10:00 AM",

"doctor": "Dr. Smith",

"purpose": "Regular checkup"

}

```

Response: `{ "success": true }` or `{ "success": false }`

Get User Appointments

Endpoint: GET `/api/my-appointments`

Response: Array of appointment objects.

Additional Information

- Ensure MongoDB is running (`mongod` command) before starting the server.

- Use appropriate CORS settings in production for security.

- Passwords are hashed using bcrypt before storage.

- Adjust JWT secret and expiration time (`jwt.js`) based on security needs.

- Implement HTTPS for secure data transmission in production environments.

This Word document provides a structured and detailed guide to understanding, setting up, and using the Clinic Appointment System. It covers architecture, setup instructions, API documentation, and essential information for running and testing the system effectively. Adjust paths, URLs, and credentials according to your specific implementation details before deployment.