**API Documentation for Appointment Management System**

**AuthController**

This controller handles user authentication operations such as registration and login. Users must register and log in to obtain a token. The token must be included in the request headers (as Authorization: Bearer <token>) to access secured endpoints.

**Base URL:** https://localhost:7018

**{Base URL**} /api/auth

**1. Register User**

Registers a new user in the system.

* **Endpoint**: POST /register
* **Request Body**:
* {
* "username": "string",
* "passwordHash": "string" [user needs to input a password]
* }
* **Response**:
  + **201 Created**:

{

"message": "User registered successfully.",

"data": "username"

}

* + **400 Bad Request**:

{

"message": "Username already exists, try another!"

}

* + **500 Internal Server Error**:

{ "message": "User creation failed! Please try again!"}

**2. Login User**

Authenticates a user and generates a JWT token.

* **Endpoint**: POST /login
* **Request Body**:

{

"username": "string",

"passwordHash": "string" [user needs to input a password]

}

* **Response**:
  + **200 OK**:

{

"token": "JWT\_token"

}

* + **401 Unauthorized**:

{

"message": "Invalid username or password."

}

**AppointmentController**

This controller handles CRUD operations for appointments. Users must include a valid JWT token in the Authorization header to access these endpoints.

**Base URL**: /api/appointments

**1. Create Appointment**

Creates a new appointment.

* **Endpoint**: POST /
* **Request Headers**:

{"Authorization": "Bearer <JWT\_token>"}

* **Request Body**:

{

"patientName": "string",

"patientContactInfo": "string",

"appointmentDateTime": "datetime",

"doctorId": "int"

}

* **Response**:
  + **201 Created**:

{

"message": "Created",

"data": {

"patientName": "string",

"patientContactInfo": "string",

"appointmentDateTime": "datetime",

"doctorId": "int"

}

}

* + **400 Bad Request**:

{

"message": "Invalid data!"

}

* + **500 Internal Server Error**:

{

"message": "The appointment was not successfully created, please try again!"

}

**2. Get All Appointments**

Retrieves all appointments.

* **Endpoint**: GET /
* **Request Headers**:

{

"Authorization": "Bearer <JWT\_token>"

}

* **Response**:
  + **200 OK**:

{

"message": "Data found!",

"data": [ { /\* appointments \*/ } ]

}

* + **404 Not Found**:

{

"message": "Data not found!"

}

* + **500 Internal Server Error**:

{

"message": "Error fetching appointments, please try again!"

}

**3. Get Appointment by ID**

Retrieves a specific appointment by its ID.

* **Endpoint**: GET /{id}
* **Request Headers**:

{

"Authorization": "Bearer <JWT\_token>"

}

* **Response**:
  + **200 OK**:

{

"message": "Data found!",

"data": { /\* appointment \*/ }

}

* + **404 Not Found**:

{

"message": "Data not found!"

}

* + **500 Internal Server Error**:

{

"message": "Error fetching appointment, please try again!"

}

**4. Update Appointment**

Updates an existing appointment.

* **Endpoint**: PUT /
* **Request Headers**:

{

"Authorization": "Bearer <JWT\_token>"

}

* **Request Body**:

{

"appointmentId": "int",

"patientName": "string",

"patientContactInfo": "string",

"appointmentDateTime": "datetime",

"doctorId": "int"

}

* **Response**:
* **200 OK**:

{

"message": "Appointment updated successfully.",

"data": { /\* updated appointment \*/ }

}

* **400 Bad Request**:

{

"message": "Invalid data!"

}

* **404 Not Found**:

{

"message": "Data not found!"

}

* **500 Internal Server Error**:

{

"message": "Error occurred while updating the appointment, please try again!"

}

**5. Delete Appointment**

Deletes an appointment by its ID.

* **Endpoint**: DELETE /{id}
* **Request Headers**:

{

"Authorization": "Bearer <JWT\_token>"

}

* **Response**:
  + **200 OK**:

{

"message": "Appointment deleted successfully."

}

* + **404 Not Found**:

{

"message": "Data not found!"

}

* + **500 Internal Server Error**:

{

"message": "Error occurred while deleting the appointment, please try again!"

}

This documentation outlines the endpoints, required headers, payloads, and responses for seamless integration and token-based authentication.