ASSIGNMENT 4: AUTHENTICATION

Create a new project and upload the source code to a **new** GitHub repository. Build a backend API using Node.js on port 4000. You can use either Express or NestJS to meet the project requirements. Deploy your backend API to cloud provider like Render.com

R1: Create API endpoint return your information with your full name and your student code.

Description:

* This API retrieves user information in the form of a JSON object.

Request:

* Method: GET
* URL: <http://localhost:4000/info>

Response Body:

{

"data": {

"fullName": "Nguyen Van A",

"studentCode": "QNUO1234"

}

}

Response Details:

* data: The main object containing user information.
* name: A string representing the user’s name (e.g., "Nguyen Van A").
* code: A string representing the user’s unique code (e.g., "HELO1234").

R3: API specification for user registration, login, JWT-based authentication, and user profile management using MongoDB. This includes the request/response formats, JWT token handling, and security checks.

API Specification:

1. User Registration

Method: POST

Endpoint: /auth/register

Description: Registers a new user. The password must be hashed before storing in MongoDB.

Request Body:

{

"fullName": "John Doe",

"username": "johndoe",

"password": "yourpassword"

}

Response:

201 Created

Response Body:

{

"success": true,

"message": "User registered successfully",

"data": {

"id": "60a6c72bf25b4e1f88d81a44",

"fullName": "John Doe",

"username": "johndoe"

}

}

400 Bad Request (if username is already taken)

2. User Login

Method: POST

Endpoint: /auth/login

Description: Logs in the user, checks credentials, and returns a JWT token.

Request Body:

{

"username": "johndoe",

"password": "yourpassword"

}

Response:

200 OK

Response Body:

{

"success": true,

"message": "Login successful",

"token": "JWT-TOKEN"

}

401 Unauthorized (if credentials are incorrect)

3. Get User Information

Method: GET

Endpoint: /users/me

Description: Retrieves the logged-in user's information. Requires a valid JWT token.

Headers:

Authorization: Bearer JWT-TOKEN

Response:

200 OK

Response Body:

{

"success": true,

"data": {

"id": "60a6c72bf25b4e1f88d81a44",

"fullName": "John Doe",

"username": "johndoe"

}

}

401 Unauthorized (if JWT token is missing or invalid)

4. Edit User Information (Full Name)

Method: PUT

Endpoint: /users/me

Description: Allows the logged-in user to update their full name. Requires a valid JWT token.

Headers:

Authorization: Bearer JWT-TOKEN

Request Body:

{

"fullName": "John Doe Updated"

}

Response:

200 OK

Response Body:

{

"success": true,

"message": "User information updated successfully",

"data": {

"id": "60a6c72bf25b4e1f88d81a44",

"fullName": "John Doe Updated",

"username": "johndoe"

}

}

401 Unauthorized (if JWT token is missing or invalid)

JWT Authentication:

After a successful login, the server returns a JWT token. This token should be sent in the Authorization header as Bearer JWT-TOKEN for all protected routes (like /users/me and /users/me [PUT]).

The token should be validated on each protected route. If the token is missing or invalid, the server should respond with 401 Unauthorized.

Error Responses

400 Bad Request: For missing or invalid input during registration.

{

"success": false,

"message": "Username already exists"

}

401 Unauthorized: For incorrect credentials or missing/invalid JWT tokens.

{

"success": false,

"message": "Invalid username or password"

}

{

"success": false,

"message": "Token is missing or invalid"

}

500 Internal Server Error: For unexpected errors on the server.

{

"success": false,

"message": "Internal server error"

}

MongoDB User Schema

The user data will be stored in a MongoDB collection. Please design database as requirement above.

This API specification ensures that only logged-in users can access and update their own profile information using JWT-based authentication. All user data is stored securely in a MongoDB collection.

Submit file studentCode\_assignment\_4.doc contains link to github repository and url of deployed backend API.