FINAL PE – SDN302 - FALL 2024 #1

Create a backend API using Node.js on port 4000. You can use either Express or NestJS, following these requirements.

Q1: 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": {
    "name": "Nguyen Van A",
    "code": "HELO1234"
  }
}
```

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").

Q2: Create API endpoint calculate your age based on your year of birth?

Description:
- Calculates a user's age based on the provided birth year.

Request:
- Method: POST
- URL: http://localhost:4000/age

Request Body:
```
{
  "birthYear": 2000
}
```

Response Body:

```
{
  "data": {
    "age": 24
  }
}
```

Response Details:

- data: The main object containing user information.
- age: A number representing the age

Q3: Build API to manage To-Do list. The data will be stored in a MongoDB collection, where each To-Do item is represented as a document with fields:

- _id: Unique identifier for the To-Do item (auto-generated by MongoDB).
- title: The title of the task.
- description: A short description of the task.
- completed: A boolean indicating whether the task is completed.

Here's a REST API specification for managing a To-Do list using MongoDB for data storage:

Base URL:

- http://localhost:4000

Endpoints Overview:

- GET /todos: Fetch all To-Do items.
- GET /todos/:id: Fetch a specific To-Do item by ID.
- POST /todos: Create a new To-Do item.
- PUT /todos/:id: Update a specific To-Do item by ID.
- DELETE /todos/:id: Delete a specific To-Do item by ID.

1. GET /todos

Description:

- Fetch all To-Do items from the database.

Request:

- Method: GET
- URL: /todos

Response:

```
{
  "data": [
    {
      "_id": "610d1f43ef6f9e8a2f3b2876",
      "title": "Buy groceries",
      "description": "Milk, Bread, Eggs",
      "completed": false
    },
    {
      "_id": "610d1f43ef6f9e8a2f3b2877",
      "title": "Finish homework",
      "description": "Math exercises",
      "completed": true
    }
  ]
}
```

2. GET /todos/:id

Description:

- Fetch a specific To-Do item by its ID.

Request:

- Method: GET
- URL: /todos/:id

Response:

```
{
  "data": {
    "_id": "610d1f43ef6f9e8a2f3b2876",
    "title": "Buy groceries",
    "description": "Milk, Bread, Eggs",
    "completed": false
  }
}
```

3. POST /todos

Description:

- Create a new To-Do item.

Request:

- Method: POST
- URL: /todos

Request Body:

```
{
  "title": "Buy groceries",
  "description": "Milk, Bread, Eggs",
  "completed": false
}
```

Response:

```
{
  "data": {
    "_id": "610d1f43ef6f9e8a2f3b2876",
    "title": "Buy groceries",
    "description": "Milk, Bread, Eggs",
    "completed": false
  }
}
```

4. PUT /todos/

Description:

- Update a specific To-Do item by its ID.

Request:

- Method: PUT
- URL: /todos/:id

Request Body:

```
{
  "title": "Buy groceries",
  "description": "Milk, Bread, Eggs",
  "completed": true
}
```

Response:

```
{
  "data": {
    "_id": "610d1f43ef6f9e8a2f3b2876",
    "title": "Buy groceries",
    "description": "Milk, Bread, Eggs",
    "completed": true
  }
}
```

5. DELETE /todos/

Description:

- Delete a specific To-Do item by its ID.

Request:

- Method: DELETE
- URL: /todos/:id

Response:

```
{
  "data": "Item deleted"
}
```

Submission Guidelines:

- Submit your entire project without the node_modules folder.
- Compress the project folder into a .zip file.
- Include a README file with instructions on how to connect to the database.