Client Initiation: Building the Portal for New Insurance Clients

Perform the following tasks before API development:

- Download the MySQL database by clicking on the provided link. **Insurance.sql**
- Access your database information by clicking **here**
- Configure your database connection in the **"*src/main/resources/application.properties*"** file, specifying the database name, username, and password.
- To set up the downloaded database in your database environment:

    1. In the **"DB"** tab, select **"localhost"** and enter your username and password.
    2. Go to the "Import" section, select the downloaded schema file, and click "GO."

In this task, you are required to implement a **POST**endpoint for creating new clients in the Insurance Management System. The endpoint will be accessible via the URL **/clients**. It expects a JSON object in the request body, containing client information such as name, contact information, address, and client type (individual or organization).

Files to Use:

- Clients.java (src/main/java/com/InsuranceManagementSystem/entity/Clients.java)
- ClientsRepository.java (src/main/java/com/InsuranceManagementSystem/repository/ClientsRepository.java)
- ClientsController.java (src/main/java/com/InsuranceManagementSystem/controller/ClientsController.java)
- ClientsService.java (src/main/java/com/InsuranceManagementSystem/service/ClientsService.java)

Steps to complete the task:

- In your ClientsController.java, use the method createClient() with API endpoint which receives the POST request along with the input JSON data of clients according to the structure in entity Clients.java.
- From the ClientsController.java the input is sent to ClientsService.java to the method createClient().
- The system ensures if the input data is incomplete (any of the fields is empty or null), the endpoint returns an error message indicating that the client information is incomplete.
- It verifies that the contactInformation in the input should be a valid mail id, and nothing else (say phone number), the endpoint returns an error message stating that the contact information is not valid.
- It also verifies that the clientType should be either an individual or an organization, the endpoint returns an error message stating that the client type is not valid.
- Check using the ClientsRepository object, if the contact information provided already exists for another client, if so then the endpoint returns an error message stating that the contact information is already in use by another client.
- Upon successful validation, the client creation is successful and stored in the "*clients*" table using the ClientsRepository object to persist it in the database.
- Return a successful or failed response map to createClient() in the ClientsController.java file, representing the created client.
- Return a JSON response from ClientsController.java with detailed information about the newly created client entry.

Input:
A JSON object representing a client with the specified properties:

```
{
  "name": "John Doe",
  "contactInformation": "john@example.com",
  "address": "123 Main St",
  "clientType": "Individual"
}
```

Output:
Successful JSON response representing the created client:

```
{
  "clientId":1
  "name": "John Doe",
  "contactInformation": "john@example.com",
  "address": "123 Main St",
  "clientType": "Individual"
}
```

By implementing this task, you will learn how to create a POST endpoint for adding new clients to your Insurance Management System. This includes handling input validation, ensuring the uniqueness of contact information, and returning appropriate HTTP responses for different scenarios. Additionally, you will understand how to save the client data into the database and how to structure error responses for client-side consumption.