**Instructions**

**Please read the instructions carefully before answering the questions:**

* You can use the materials on your computer, notebooks, and textbooks.
* You are **NOT allowed** to use electronic devices and other methods to share data with others.
* Do not arbitrarily disconnect the **FPTU\_EXAMONLINE** network while doing the test. If any network problems occur, immediately notify the exam supervisor for assistance.

**In addition to the above conditions, students must fulfill the following requirements:**

1. Use **Visual Studio Code** (VS Code), **MongoDB Compass, Postman** tools to do the exam
2. Use the available browser software on your computer

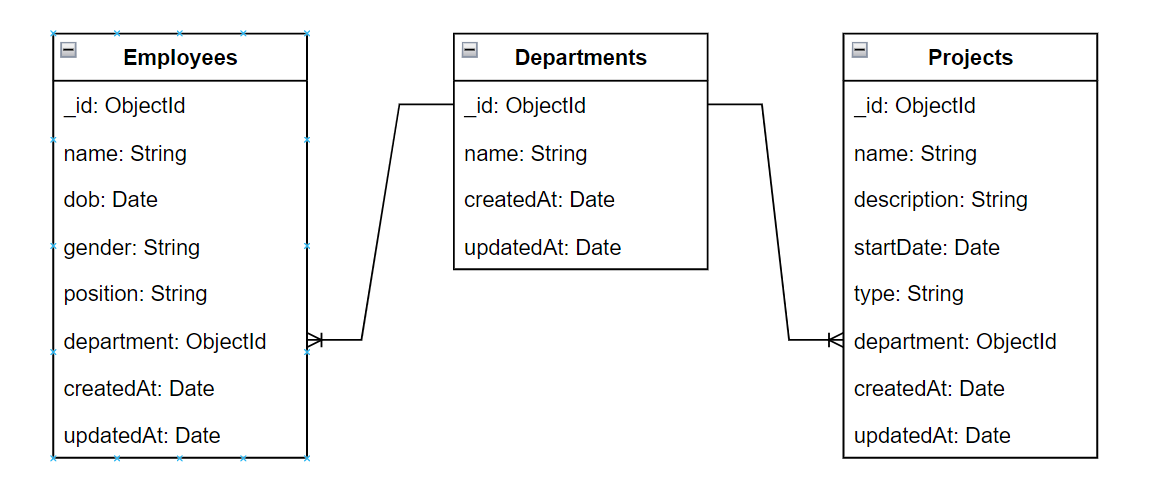
**This part is very important, please read carefully and follow the instructions**

* You are provided with the files containing data in the **given** directory.
* You must use the solution provided in advance via email and install the additional library included in the accompanying request.
* You are **not allowed** to download any additional libraries while taking the test
* Violating one of the above, your exam is considered invalid

**Instructions for submission**

* Before submitting the solution, delete the [**node\_modules**] folders in your test to reduce the space, to meet the allowed capacity of the **PEA\_Client** software
* Use **PEA\_Client** software to browse the **resource** directory to complete the submission process.

**Use the following database diagram for doing this exam:**



**Requirements:**

* Using **MongoDB Compass**, create a database named “**SDN301\_Spring2024\_B1**”. Create collections corresponding to the data in the **given** directory. Then, import data into the collections created in the database.



**Question 1. (5 points)**

In this question, you are asked to write an application (**back-end** given folder), that provide some API to manage: **Projects**, **Employees** and **Departments**.

Note: 0 will be given to the work that

* Not using database connection string in the **.env** file
* Not configuring the root path of API Web App at: **http://localhost: 9999**

You are asked to build three APIs, as following:





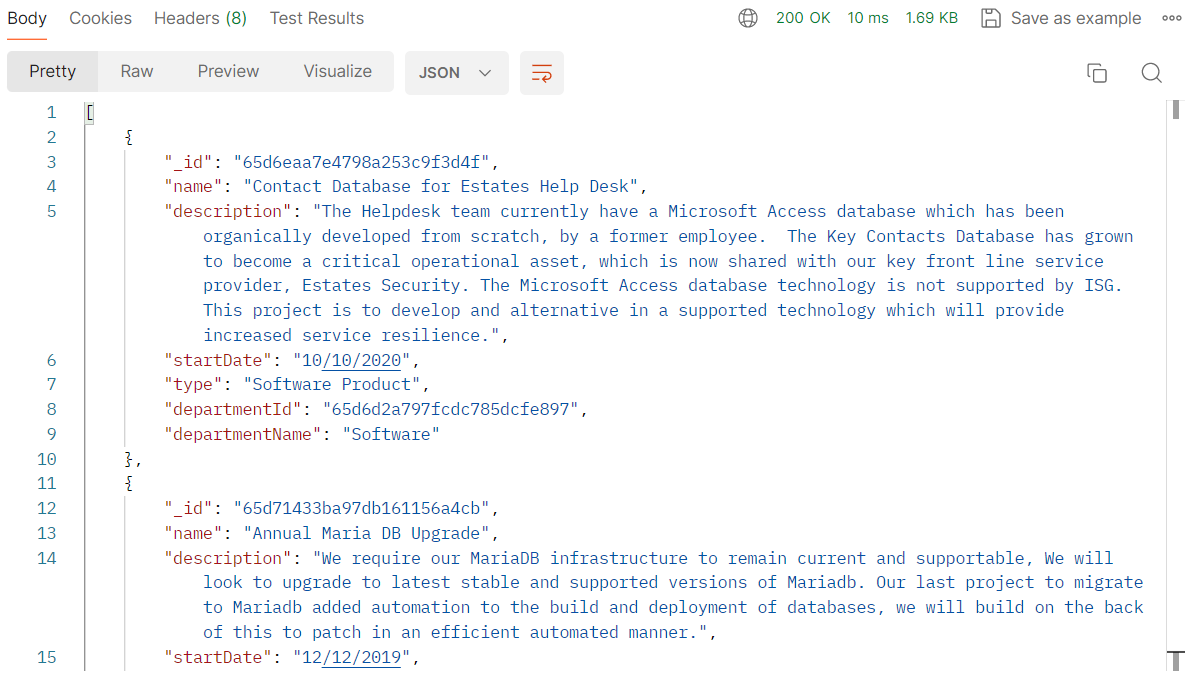


* 1. **(2 points)**

The API at url: [**http://localhost:9999/projects**](http://localhost:9999/projects) return information of all **projects** in database including corresponding **department** data, using GET method.

Each **project** requires the information shown in *Figure 1*. **Note** that:

* Reduced by **1 point** if the output is not formatted.



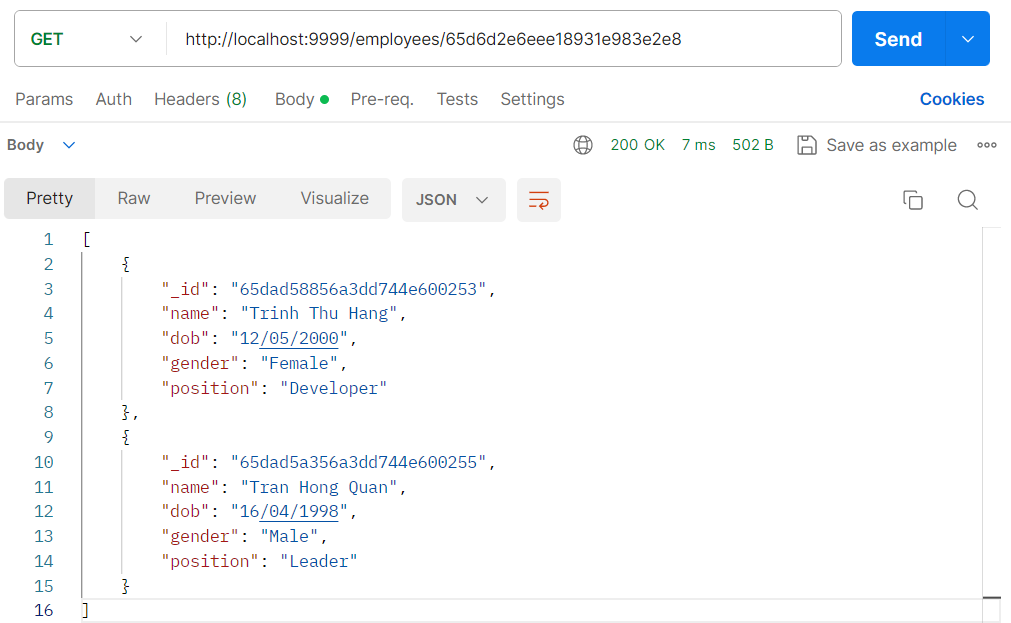
*Figure 1 – The result of API at* [***http://localhost:9999/projects***](http://localhost:9999/projects) *(using method GET)*

* 1. **(1.5 points)**

The API at url: [**http://localhost:9999/employees/:dept**](http://localhost:9999/employees/:dept) (**dept** - is an ObjectId of the Department) return information of all employees by **\_id** of the department in database, using GET method.

See *Figure 2* for more detail. **Note** that:

* Reduced by **0.5 point** if the output is not formatted.

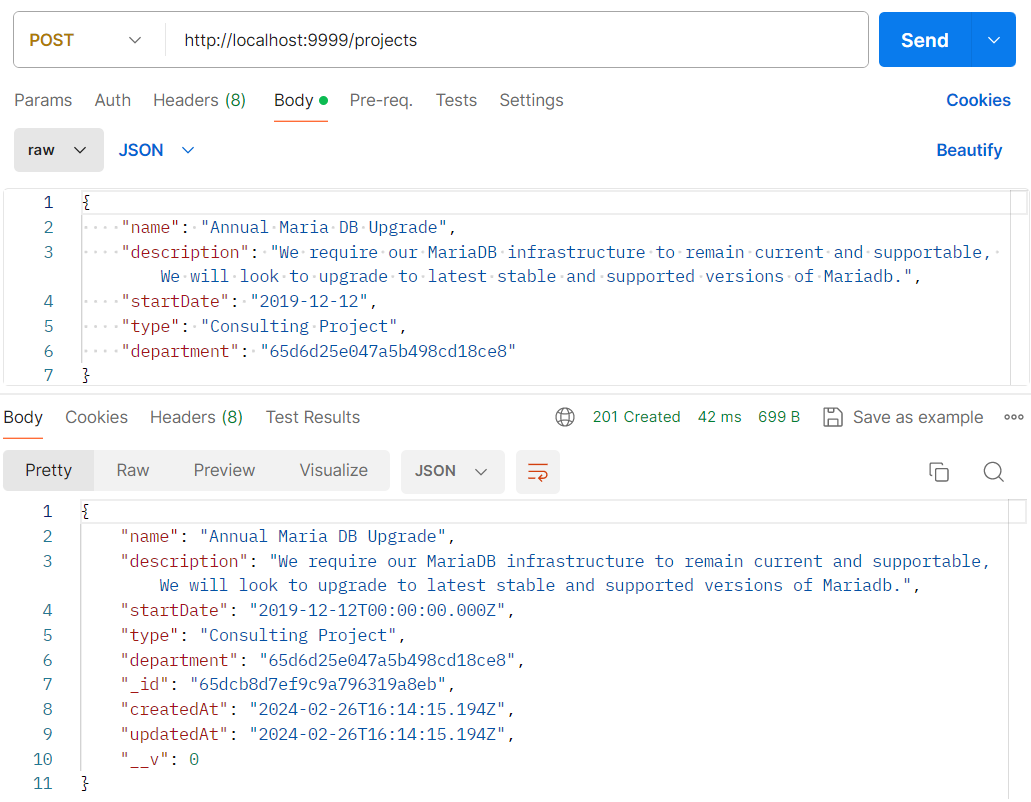


*Figure 2 - The result of API at* ***/employees/65d6d2e6eee18931e983e2e8*** *using GET method*

* 1. **(1.5 points)**

The API at url: [**http://localhost:9999/projects**](http://localhost:9999/projects) , using POST method to create a new Project.

See *Figure 3* for more detail.



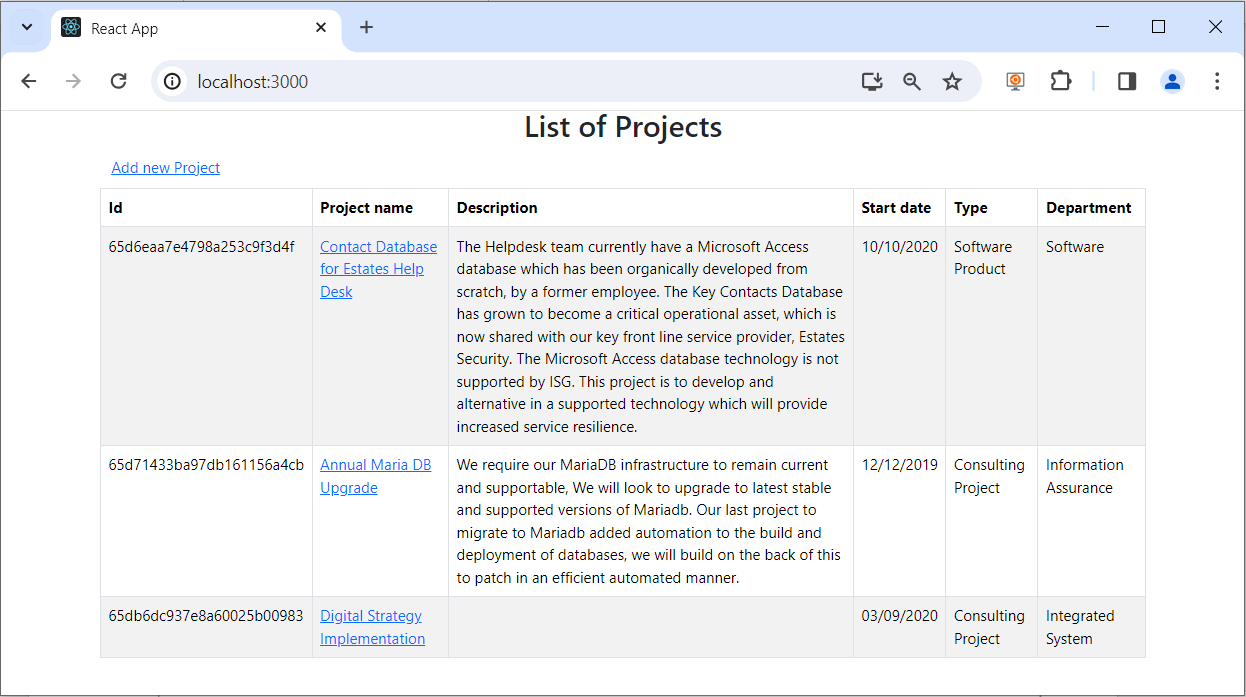
*Figure 3 - Using POST method to create a new Project at url:* [*http://localhost:9999/projects*](http://localhost:9999/projects)

**Note:** Reduced by **0.5 point** if not return the result after successful creation.

**Question 2. (5 points)**

In this question, you are asked to write a React web application (in **front-end** given folder), that using APIs of **Question 1** to manage **Projects** and **Employees**.

**Note**: using the root path [**http://localhost:3000**](http://localhost:3000) to call APIs.



*Figure 4 - The page when first loaded*

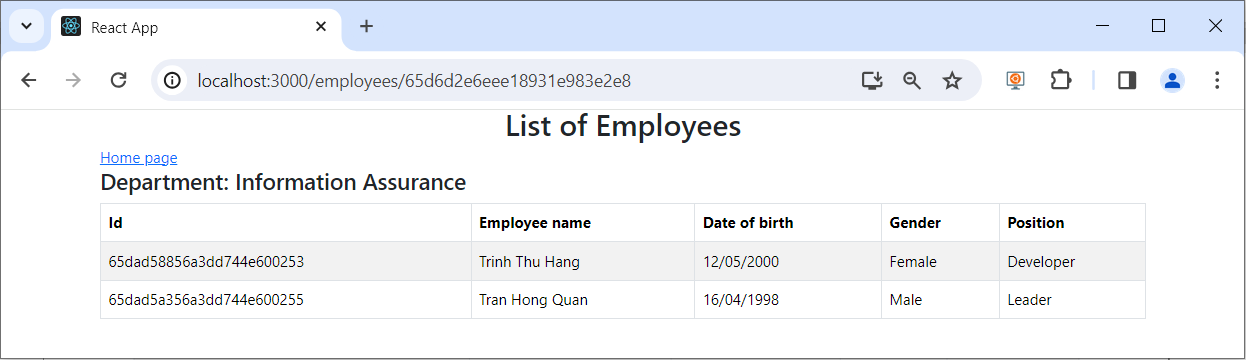
* 1. **(2 points)**

When user open [**http://localhost:3000/**](http://localhost:3000/) , load all **Projects** from database like *Figure 4*.

* Reduce by **0.5 points** if the design does not have the correct layout and styles, as shown in *Figure 4.*
* Reduced by **0.5 points** if the correct department is not displayed.
  1. **(1 point)**

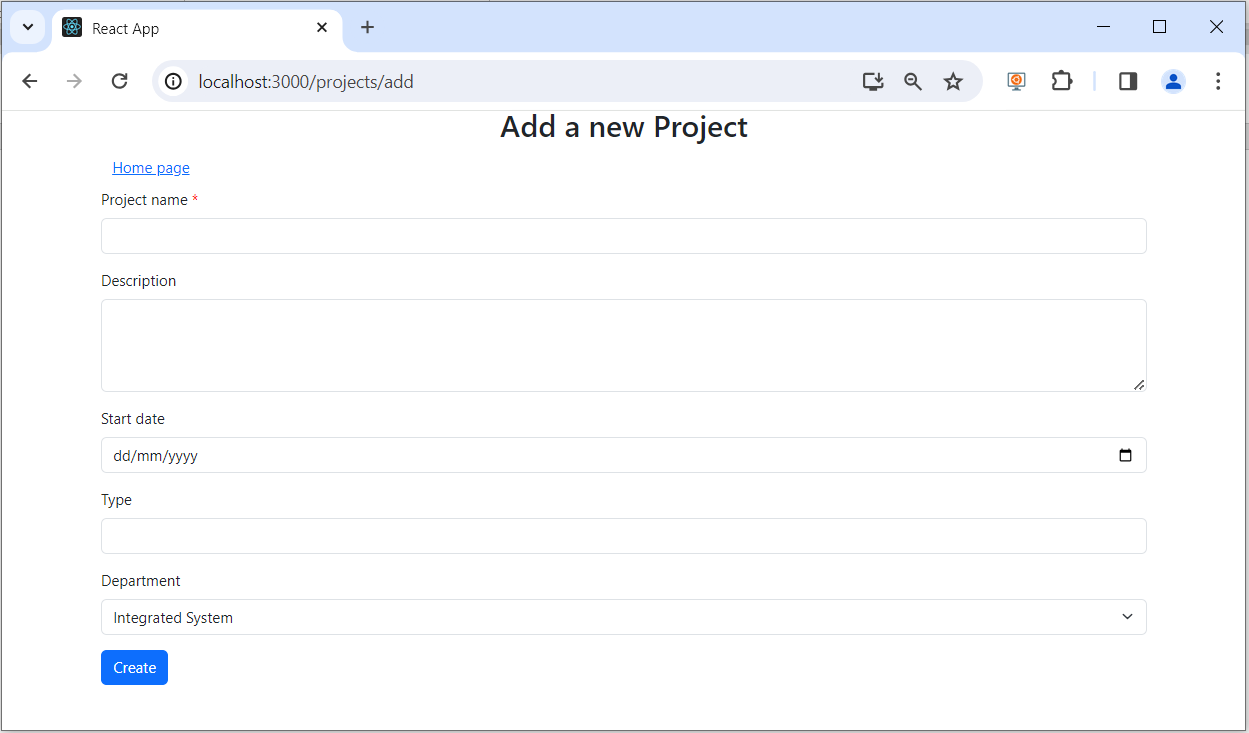
When a user clicks on a link by the **name of a project**, navigate to the URL: [**http://localhost:3000/employees/:dept**](http://localhost:3000/employees/:dept)(**dept** - is an ObjectId of the Department) . At the same time, display the list of corresponding employees, as shown in *Figure 5.*

* Reduced by **0.5 points** if Department name information is not displayed

*Figure 5 - List of employees by department*

* 1. **(2 points)**

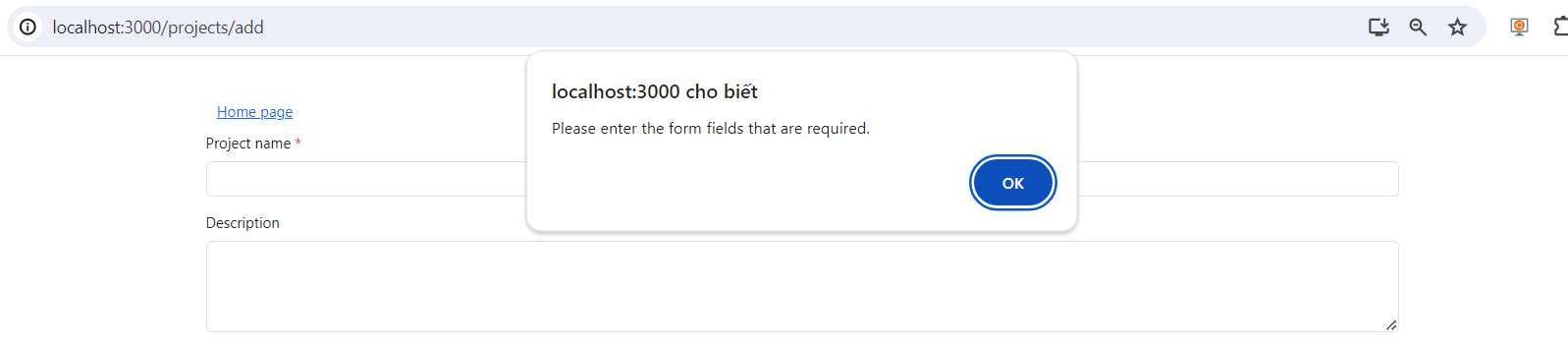
When the user clicks on the “**Add new Project**” link in *Figure 4*. Navigate to the URL: <http://localhost:3000/projects/add> , as shown in *Figure 6*



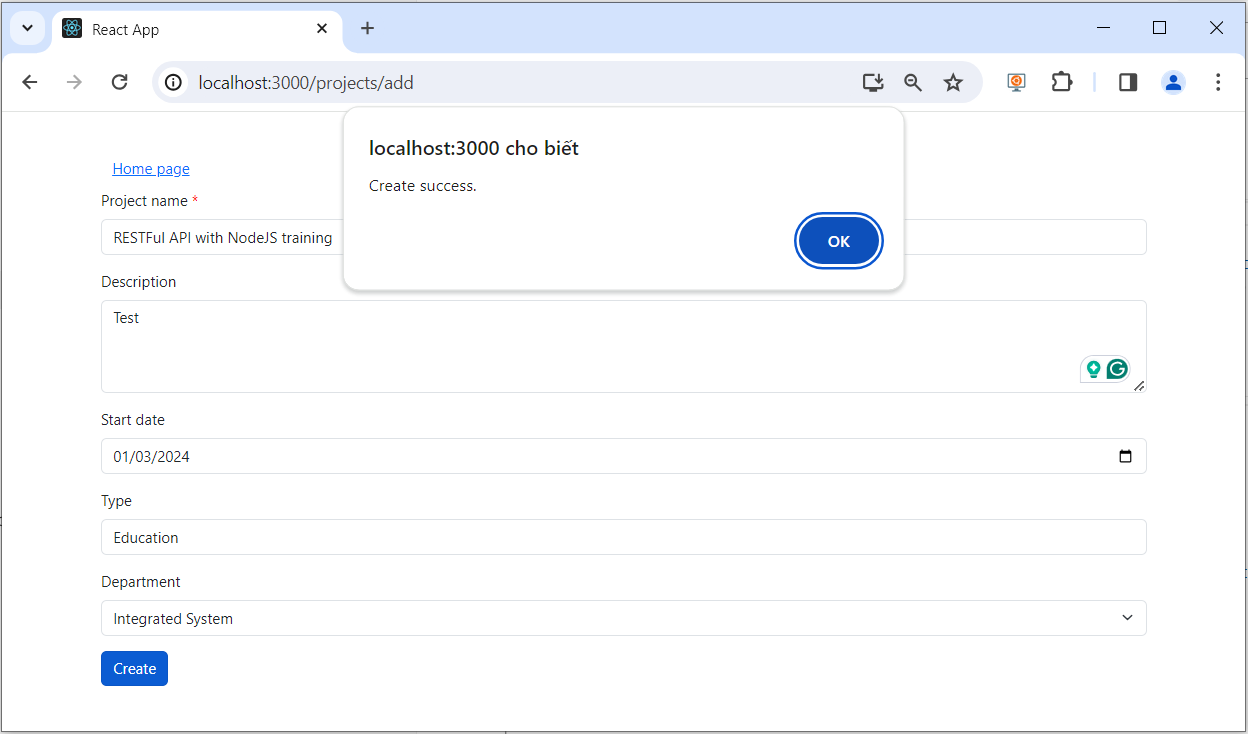
*Figure 6 - Add a new Project form*

* Design the form as required - **0.5 points**.
* Load all departments from the database into a drop-down list (an additional API can be written for this function) - **0.5 points**.
* Create a new project successfully and redirect to home page at url: <http://localhost:3000> (**1 point**)

**Note**: Reduced by 0.5 points if the Project name field is left blank



*Figure 7 - Check input for Project name*



*Figure 8 - Create new a project success*

**--- END ---**