Tutorial 5: Back-End Frameworks I [Individual Deliverable]

Learning Outcomes:

- Work with the Back-End framework/library of your choice (i.e., the one you have decided to use for your project).
- Understand how API calls are written in the Back-End framework/library you have chosen (i.e., Express.js or Flask).
- Work individually to create a simple API.

Instructions:

- Use the code shown in the tutorial video for **GET API**, the **API** returns a list of user objects, and add new methods for a **POST**, **PUT** and a new **GET API** request.

GET : <your_application_link>/users

Sample Success Response :

```
{
      message : "Users retrieved",
      success : true,
      users : [{
            email : "abc@abc.ca",
            firstName : "ABC",
            id : "5abf6783"
            },
            {
            email : "xyz@xyz.ca",
            firstName: "XYZ",
            id : "5abf674563"
      }]
}
```

- The **PUT API** should be able to update **email and/or firstname** of an existing object in the list. The new email and firstname should be passed in the body of the request.

  PUT - <your_application_link>/update/:id

  body data:

```
{
      email : "xyz@xyz.ca",
      firstName: "XYZ",
}
```

  Sample Success Response :

```
{
      message : "User updated",
      success : true
}
```

- The **POST API** should add a **New User Object** to the list and **generate an ID** for the user. The object details should be passed in the body of the request.

  POST - <your_application_link>/add

  body data:

```
{
      email : "xyz@xyz.ca",
      firstName: "XYZ",
}
```

  Sample Success Response:

```
{
      message : "User added",
      success : true
}
```

- Finally, a new **GET API** should be written, which should return a single user object given its username. The username can be passed as a query or path parameter.

  GET- <your_application_link>/user/:id

  Sample Success Response :

```
{
      success : true,
      user : {
            email : "xyz@xyz.ca",
            firstName: "XYZ",
            id : "5abf674563"
      }
}
```

- No front-end is required for this tutorial.
- **You must consider failure responses for each request** (e.g. 404, 400, 500 responses and appropriate JSON messages)**.**

Submission Guidelines

Your tutorial must be submitted through Brightspace (i.e., README file), GIT and be remotely accessible.

To submit your work to Brightspace:

- Create a README.txt file, follow the guidelines specified in the README template provided through Brightspace
    - Rename your README file to match naming conventions specified in the Course Syllabus (**FName_LName_README.txt**).
    - **Include** the link to your repository and the deployed application.

  Note: Your README file should include your name, GIT repository link, deployment link, and any code references. Failure to submit a README file and/or not include a link to your deployed application will result in a grade of 0.

- Ensure you submit your work by the **due date specified on Brightspace.**

To submit your work to FCS GitLab:

- Push your code to a new git repository and deploy the application to Netlify or any other deployment platform of your choice.

  ***Note:** Make sure that the deployment link you include in your README file matches the deployment link for this tutorial. Your GIT repository must be individual and private and be accessible to the Instructor and Teaching Assistants.*

- Follow the folder structure requirement shown on **Figure 1.**

- **Ensure,** your repository includes a **README.txt file**, follow the guidelines specified in the README template provided through Brightspace.

```
CSCI 4177/5709 Tutorials
    - Tutorial1
    - Tutorial2
    ....

CSCI 4177/5709 Assignments
    - Assignment1
    - Assignment2
    ...

CSCI 4177/5708 Grp-xx
    - Individual name branch
```

Figure 1. *GitLab Folder Structure Example.*

Marking Rubric:

As this tutorial is a programming tutorial, the following grading criteria will be used for marking your tutorial:

- Tutorial implements a Backend-End Framework **[2 points]**
- Tutorial implements API for GET call for list of users **[1 point]**
- Tutorial implements API for POST call to create a user **[2 points]**
- Tutorial implements API for PUT call to update a user **[2 points]**
- Tutorial implements API for GET call for a specific user **[2 points]**
- Tutorial is remotely accessible **[1 point]**
    - **Ensure** a link to your application is included in your **README** file

Note: See Tutorial 2 handout for Instructor and Teaching Assistants' FCS GitLab account information. It may be possible for you to obtain partial credit for requirements that were implemented but are not fully functional. Please note that if Gitlab link or deployment link is missing, you will receive a negative mark of -100%. Additionally, if maintainer access for any TA or Professor is missing, you will receive a negative mark of -50%. Further, it may be possible for you to obtain partial credit for requirements that were implemented but are not fully functional.