ASSIGNMENT

Centre of Excellence

1 Simple Web Based to-do tracker

Create a simple web-based (accessible from web browser) to-do tracker in web frontend framework/library of your choice. The application should run in modern web browsers (Google Chrome, Microsoft Edge, Mozilla Firefox, …). You should use general web technologies such as HTML, CSS, JavaScript/Typescript.

If you choose to do so, you can store the items in the supplied API, storing the items is not required – to-do items can be stored in memory of the web browser.

1.1 Description

The to-do tracker should allow for:

- Entering new to-do items
- Updating existing items
- Deleting existing to-do items
- Display a list of all to-do items

1.2 Application expectations

The web application front-end should contain components or pages for performing the required actions:

- Add new to-do item
- View all to-do items
- Delete a to-do item
- Modify an existing to-do item

The user should not have to click around too much to perform a specific task.

1.3 Data model

The to-do tracker has the following data model items

1.3.1 To-do item

To-do item is described with the following attributes:

Attribute name	Attribute type	Is Required	Description
Id	string	yes	Unique id of the to-do record
CreatedTime	Date Time	yes	Date and time when the record was created
TaskDescription	string	yes	Text description of the to-do task
IsCompleted	boolean	yes	Indication if the to-do task was completed or not

2 API connection

The To-do REST API enables you to create, update, delete and retrieve to-do items from a storage table.

To-do REST Api Open API definition is available at the following address https://betodoapimng.azure-api.net/apis/BEternaCoeTodoAPI/open-api-definition?subscription-key=14ab1ca20dc041e082caabc3264a2f2c.

When connecting to the API use the following subscription key: **b74cd6a4078c484a9603a5d70b4c9a18**

The subscription key can be used:

- as query string parameter `subscription-key`
- or as a http request header `Ocp-Apim-Subscription-Key`

API base URL address: **https://betodoapimng.azure-api.net/todo**.

2.1.1 API endpoints

All API endpoints expect the HTTP payload request in **application/json** content type. The responses from the API endpoints are also formatted in **application/json** format.

2.1.1.1 POST /todo

Create a new to-do item. The body of the HTTP request contains the to-do item schema as defined in **2.1.2.1 Todo**. The endpoint will return the newly created record.

2.1.1.1.1 Sample request

```
POST https://betodoapimng.azure-api.net/todo/todo HTTP/1.1
Content-Type: application/json
Ocp-Apim-Subscription-Key: b74cd6a4078c484a9603a5d70b4c9a18
Ocp-Apim-Trace: true


{
  "id": "string",
  "createdTime": "string",
  "taskDescription": "string",
  "isCompleted": false
}
```

2.1.1.2 GET /todo

The endpoint returns a list (an array) of to-do items. The schema is defined in **2.1.2.1 Todo**.

2.1.1.2.1 Sample request

```
GET https://betodoapimng.azure-api.net/todo/todo HTTP/1.1
Ocp-Apim-Subscription-Key: b74cd6a4078c484a9603a5d70b4c9a18
Ocp-Apim-Trace: true
```

2.1.1.3 GET /todo/{id}

The endpoint returns the to-do record specified by the **id** parameter.

2.1.1.3.1 Sample request

```
GET https://betodoapimng.azure-api.net/todo/todo/{id} HTTP/1.1
Ocp-Apim-Subscription-Key: b74cd6a4078c484a9603a5d70b4c9a18
Ocp-Apim-Trace: true
```

2.1.1.4 DELETE /todo/{id}

The endpoint will delete the to-do record specified by the **id** parameter.

2.1.1.4.1 Sample request

```
DELETE https://betodoapimng.azure-api.net/todo/todo/{id} HTTP/1.1
Ocp-Apim-Subscription-Key: b74cd6a4078c484a9603a5d70b4c9a18
Ocp-Apim-Trace: true
```

2.1.1.5 PUT /todo/{id}

The endpoint will update the to-do record specified by the **id** parameter. The HTTP request payload contains the **2.1.2.2 todoUpdateModel** object.

2.1.1.5.1 Sample request

```
PUT https://betodoapimng.azure-api.net/todo/todo/{id} HTTP/1.1
Content-Type: application/json
Ocp-Apim-Subscription-Key: b74cd6a4078c484a9603a5d70b4c9a18
Ocp-Apim-Trace: true


{
  "taskDescription": "string",
  "isCompleted": false
}
```

2.1.2 API objects (schemas)

2.1.2.1 Todo

Describes the To-do record.

```
{
     id                     string
     createdTime            string
     taskDescription        string
     isCompleted            boolean

}
example: {"id": "string", "createdTime": "string", "taskDescription": "string",
"isCompleted": false}
```

2.1.2.2 todoUpdateModel

Describes the to-do record update parameters.

```
{
     taskDescription        string
     isCompleted            boolean

}
example: {"taskDescription": "string", "isCompleted": false}
```

3 Resources

List of helpful resources to speed up the assignment:

- https://www.youtube.com/watch?v=E1E08i2UJGI
- https://www.youtube.com/watch?v=Wts7z-twq7c
- https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_todo_list_beginning

- https://www.freecodecamp.org/news/create-a-solid-to-do-app-with-react/
- https://towardsdatascience.com/build-a-simple-todo-app-using-react-a492adc9c8a4
- https://www.w3schools.com/howto/howto_js_todolist.asp
- https://github.com/sumitkharche/fluent-ui-todo-app
- https://developer.microsoft.com/en-us/fluentui#/get-started/web
- https://github.com/Azure-Samples/TodoFunctions
- https://vuejsexamples.com/a-simple-todo-list-vue-application/
- https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_todo_list_beginning
- https://www.teclogiq.com/blog/angular-todo-application/