REST API's Course Assignment^2

Please read the following instructions carefully. If any part is unclear or you have questions, contact the designated teachers through Microsoft Teams direct message.

Discussing the course assignment in class channels is strictly prohibited.

Additionally, any discussions about the assignment outside of official teacher-student communications, including private messages or in-person conversations with classmates, are not allowed.

The course assignment will be graded on a "pass / not pass" basis.

IMPORTANT

The **Git Classroom link** for this assignment will be posted in your **class channel on Microsoft Teams** on the day the assignment opens. Expect the link by 09H00 when the assignments open.

Your GIT repository for this assignment will be private, accessible only by you and the course instructors and graders.

Only commits made in the Git Classroom will be considered for grading. Any repositories you create outside the provided Git Classroom link will not be graded.

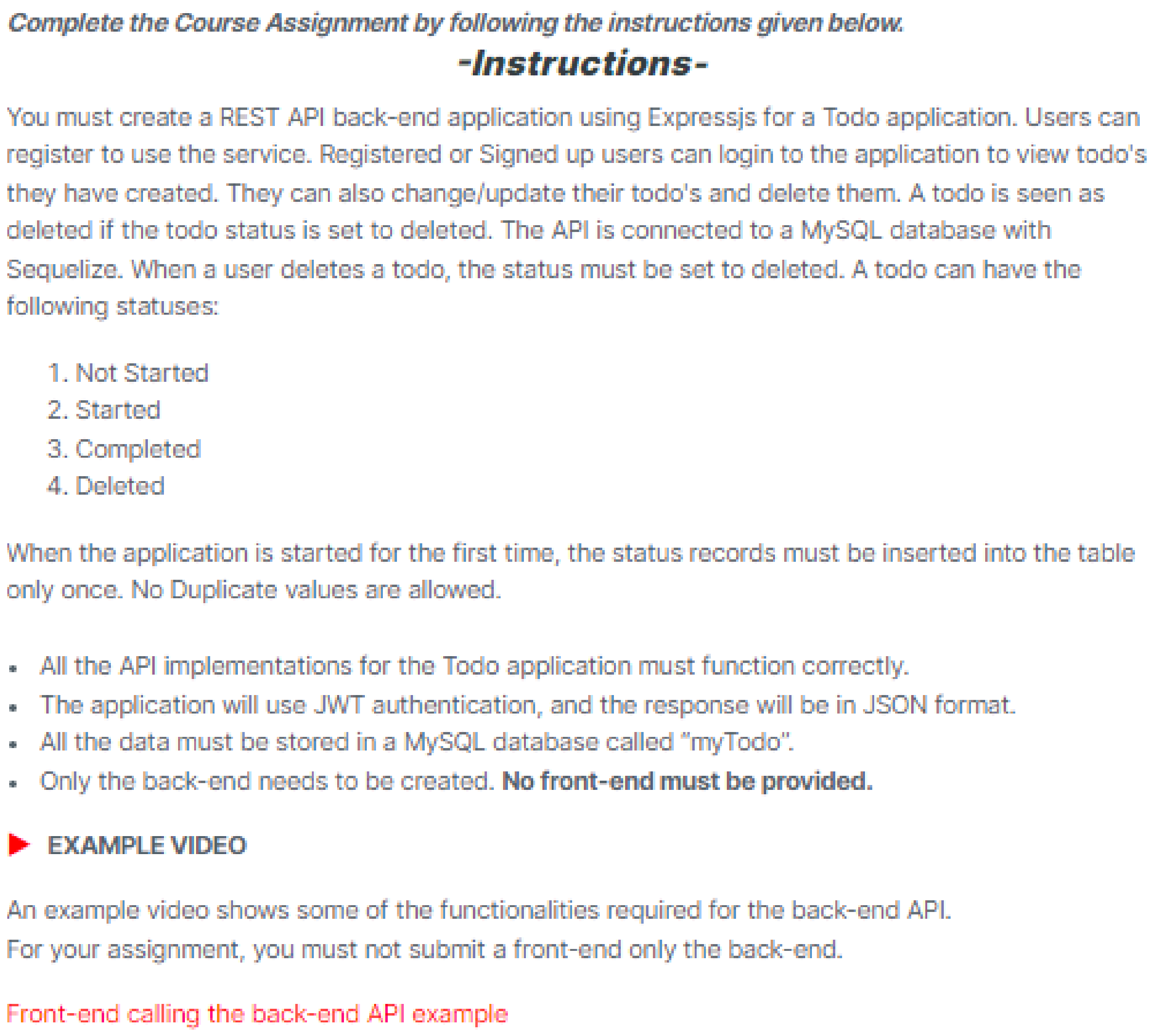
Ensure all your work is committed **before the course assignment deadline.**

After the deadline, you cannot commit any more code to your repository or Git classroom. Any attempts to commit after the deadline will be automatically rejected.

VIDEO LINK: **Git classroom informative video**

Commits or submissions past the deadline will not be considered for grading.

Late submissions will not be accepted, and there will be no exceptions to this rule.



Complete the Course Assignment by following the instructions given below.

-Instructions-

You must create a REST API back-end application using Expressjs for a Todo application. Users can register to use the service. Registered or Signed up users can login to the application to view todo's they have created. They can also change/update their todo's and delete them. A todo is seen as deleted if the todo status is set to deleted. The API is connected to a MySQL database with Sequelize. When a user deletes a todo, the status must be set to deleted. A todo can have the following statuses:

1. Not Started
2. Started
3. Completed
4. Deleted

When the application is started for the first time, the status records must be inserted into the table only once. No Duplicate values are allowed.

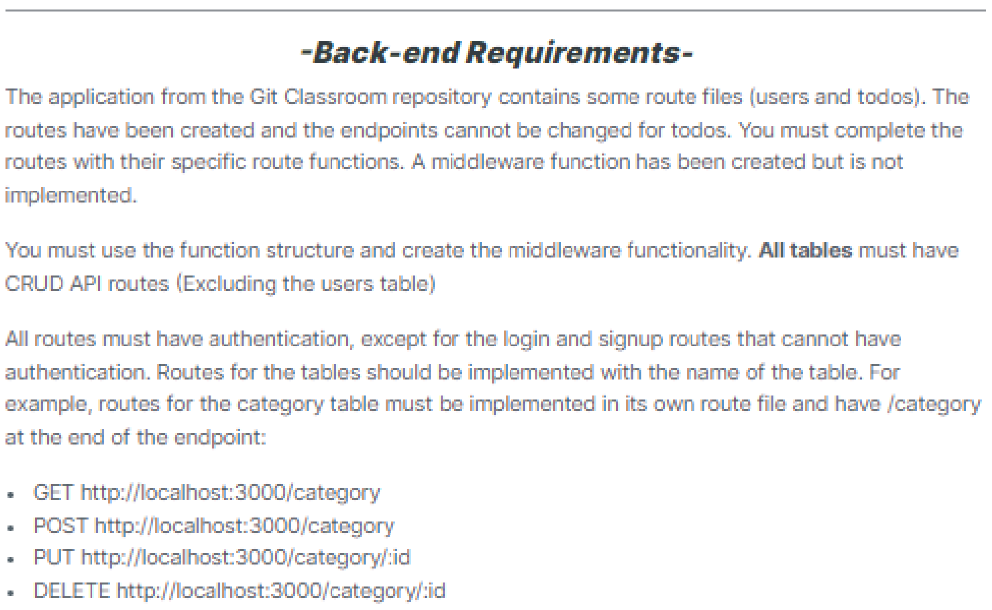
- All the API implementations for the Todo application must function correctly.
- The application will use JWT authentication, and the response will be in JSON format.
- All the data must be stored in a MySQL database called "myTodo".
- Only the back-end needs to be created. **No front-end must be provided.**

▶ **EXAMPLE VIDEO**

An example video shows some of the functionalities required for the back-end API.
For your assignment, you must not submit a front-end only the back-end.

Front-end calling the back-end API example

Front-end calling the back-end API example



-Back-end Requirements-

The application from the Git Classroom repository contains some route files (users and todos). The routes have been created and the endpoints cannot be changed for todos. You must complete the routes with their specific route functions. A middleware function has been created but is not implemented.

You must use the function structure and create the middleware functionality. **All tables** must have CRUD API routes (Excluding the users table)

All routes must have authentication, except for the login and signup routes that cannot have authentication. Routes for the tables should be implemented with the name of the table. For example, routes for the category table must be implemented in its own route file and have /category at the end of the endpoint:

- GET http://localhost:3000/category
- POST http://localhost:3000/category
- PUT http://localhost:3000/category/:id
- DELETE http://localhost:3000/category/:id

NOTE: A Category cannot be deleted if its id has been assigned to a todo.

Each user creates their own category list.
Error handling and validation must be implemented per endpoint. The API responses for each endpoint must be in JSON format, followed by the following structure.

Success

```
{
    "status": "success",
    "data": {
        "statusCode": 200,
        "result": ".........."
    }
}
```

Failed

```
{
    "status": "fail",
    "data": {
        "statusCode": 400,
        "result": "..........."
    }
}
```

Error

```
{
 "status": "error",
 "result": ".....",
}
```

NOTE

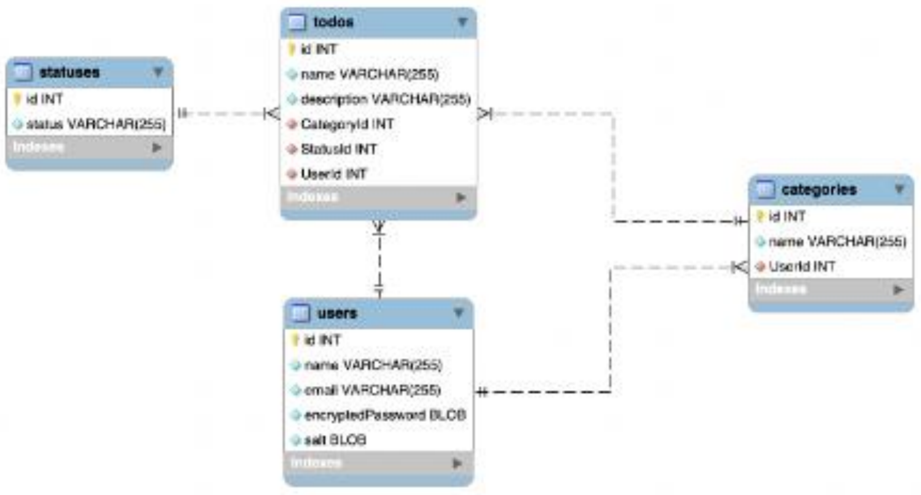
- **Express** needs to be used to create this API.
- You must use a .env file for the database connection settings, such as login information database and host.
- All router files must be created with the correct endpoints and path.
- **Only JSON responses** need to be supplied for ALL the endpoint results. (**JSend must be used**)
- **Error handling** for all endpoints must be implemented, and fail() should be used.
- Database implementations must be done with **Sequelize.**
- No user information can be sent to the API endpoints. Any identifiable user information must be in the JWT generated for the logged-in user.

- Initial Code-

Code:
The initial code base can be downloaded/cloned from the Git Classroom link in your class channel on Microsoft Teams.

-Database-

Below is an Entity Relationship Diagram of the "myTodo" database:



A single table is used for the Todo list items, while another table stores the Todo categories, and another the statuses for the Todo's

Another table stores registered users – these users can log in after they have signed up.

When the application is started for the first time, the database structure should be created from the repository's initial code.

-Authentication-

Authentication must be implemented for this application as follows:

- Passwords must be encrypted when a user is created and saved in the database.
  - Any hashing algorithm can be used - For example, sha256.
- JWT token needs to be implemented
- The email address and passwords for registering or logging in must be in the request's body.

All endpoints should be available only for authenticated users – excluding endpoints associated with Sign up and Logging-in.

-Testing-

Use **JEST** and **Supertest** packages.

Create the "Todos.test.js" file in which you will test the following scenarios:

1. Logging in with a valid account.
2. Using the token from 1. to get all the users Todos.
3. Using the token from 1. and add a new Todo item.
4. Deleting the created Todo item from number 3.
5. Trying to get Todos without sending JWT token in the header.
6. Trying to get Todos by sending an invalid JWT token.

-Documentation-

Documentation for your API must be implemented with swagger. The complete documentation for your API must be accessible from the /doc endpoint.
example http://localhost:3000/doc

-Finally-

Finally, **the link to your repository and your Github username** needs to be submitted on Moodle in a **.txt file** named the following:
"**FName_LName_API_CA_ClassXXFT.txt**" - if you are a Full-Time student,
"**FName_LName_API_CA_ClassXXPT.txt**" - if you are a Part-Time student

(Replace 'FName' with your First Name and 'LName' with your Last Name).
(Replace 'Class' with your class, e.g. 'Aug', 'Oct', etc)
(Replace 'XX' with your class year e.g. 22, 23)

EXAMPLE: FJ_BOTHA_API_CA_JAN23FT.txt

NOTE: This .txt file is the ONLY submission for this Course Assignment. (In this .txt file, you have to add your application repository link and GitHub username)