IDG2100-2024-oblig2

This document contains the description and starter code for `oblig2: IDG2100 Spring 2024`.

Goal

- Prove your understanding of Web `APIs` and how to use them.
- Show that you can design `APIs` by using best design practices and document your process (define business objectives, outline key user stories, select technology architecture, Write `API` specification)
- Demonstrate you can create and build a web server using `Node.js` and `Express.js`, which will be the backend technologies for your `API`. You must also demonstrate your understanding of `API` design best practices, especially regarding following the naming conventions of your `API` endpoints and `CRUD` operations.
- Show that you can develop an `API` that allows reading, writing, updating, and deleting data in a database (`MongoDB`)

The project is evaluated based on the previous goals. Therefore, you must show you understand all the previous concepts to get a passing grade. You may want to add a `readme` file to clarify any decision taking during the design/implementation process.

Context

In this "oblig" you will be presented with a brief description of the problem you have to solve using `APIs`.

This is an individual task, and you are required to build everything from the ground up. Although, you may utilize snippets of code from tutorials or official documentation, you must clearly acknowledge the sources in the comments of your code. Plagiarism or cheating will be deemed to have taken place if the submitted code shows substantial similarities to other students' assignments or projects found online. In such cases, the matter will be reported to the NTNU appeals committee for further examination. If you have any doubts regarding the

use of materials for your project, please reach out to the instructor for clarification.

If the assignment is graded as "not approved" you will have an additional opportunity based on the following conditions:

1. The first version of the project must have been delivered within the set deadline (never after);
2. The project must consist on a significant piece of work (i.e.: do not deliver an empty assignment);
3. The second version of the project will have to include an additional task (as described later - See Additional task).

Brief: the SUPER Assessor - Card API

Scenario

To innovate the landscape of educational assessments, the Department of Design in Trondheim has embarked on a project to digitalise the SUPER Assessor card game, a tool originally designed to aid educators in creating novel and effective assessment methods. The head of the department is very much interested in promoting initiatives like that as it can help teachers to be more innovative in their assessment methods. He also thinks making this tool freely available to other departments and universities could be interesting. If this proves helpful, it would create some visibility for our department. With the game's current form limited to physical interaction, the objective is to develop an online platform that will make the game accessible to educators around the globe but also enhance its utility by allowing users to explore, select, and combine various mission and assessment cards digitally. This initiative aims to leverage technology to foster a creative environment where educators can easily generate, refine, and share innovative assessment strategies, expanding the game's reach and impact. In the previous assignment you have created parts of the front-end, now it is time to create the back end.
The tool is still experimental; therefore, a basic API allowing reading, creating, updating, and deleting cards is sufficient. `CRUD` operations should be able to be performed for both Missions and Assessment Cards. Also, we would like to keep basic information about the API users. To summarise, the API must support `CRUD` operations to Cards and Users. In addition, you have to consider how the database will store information about the type of cards

(`Missions` and `Assessment`).

Task

- Define business objectives, outline key user stories, select technology architecture, and write API specifications.
- Build a local webserver with Node.js and Express.js. The data for your API should be stored in a database (MongoDB)
- Design and implement a MongoDB schema for Missions and Assessment Cards. Consider the attributes provided in the game description, such as card-id, card-type, card-name, card-description, and additional fields like card-details and card-category for assessment cards. Include a User collection that stores user information (e.g., name, email). This will be useful for future expansions of the project. The only mandatory information for players should be their name, surname, email, department, university, and position (e.g., Student, Teacher, TA).
- Populate your database with some dummy data (use card information you have from the previous assignment) and user info. Consider what you need to make the API work well without asking for irrelevant info (TIP: use the `json file` you created in the previous assignmend and consider adding a `script` in your `package.json` that you can execute to populate the database).
- Implementing CRUD Operations: Develop the API endpoints to allow CRUD operations for the Missions and Assessment Cards. Ensure that endpoints are included to create, read (individual and all), update, and delete for these resources. Implement endpoints to manage basic user information, focusing on creating and reading user data.
    - This new version will allow users to customise the icons the cards are using. By default, all the cards will use the icons predefined in the cards, however, users will be able to upload a custom icon (svg) for a specific card or set or cards.
- The `search` endpoint must be added to the functionalities. This endpoint will be used to provide random cards within card types and categories (e.g. mission cards, assessment card categories). Try to use query parameters to support the needed functionality. Check the following examples where different fileters are used:

```
//Example 1
/search?card-type=assessment&card-category=artefect&random=1

//Example 2
/search?card-type=assessment&card-category=artefact&random=10

//Example 3
/seach?card-type=assessment&card-category=artefact&random&exclude=[2,5,7]
```

- Prepare `POSTMAN` collections to test your `APIs`.

Note: you do not need to develop any of the front-end aspects of this project, only the back end.

Additional task

Implement this task if your assignment is graded as "Not approved" and you are entitled to deliver a second iteration (See Context)

For the additional/optional task, consider how you would create feedback about the cards. Feedback should be some short text. Users who experiment with the game may have suggestions for improving the cards. Connect this with the Oblig 1 so it pools cards randomly from the backend. Consider if you need a new database collection or want to incorporate feedback about the cards in the card or user collection.

Folder structure

- `documentation/` this folder contains the templates you must use and edit to document your `API`.
- `postman/` this folder must contain your postman queries showing how all your collections and endpoints work. There has to be a query for every endpoint and verb. The article Importing and exporting data
  explains how to export a `json` file with all your queries from Postman.
- `project/` this will be the root folder of your project. All your code and `API` must be here.

Delivery

This assignment must be delivered in two different places: GitHub classroom and Blackboard.

- To deliver the assignment in GitHub Classroom, you only need to make sure all your changes and commits are pushed to your Git repository.
    - A Pull request is created automatically when the repository is cloned. Feedback will be included there if needed. Do not remove or close that Pull Request.
    - Only the changes in the "main" branch will be considered for giving feedback or grading the assignment.
- It is imperative that you work exclusively with this Git repository to ensure that all modifications are trackable and your code is backed up on a regular basis. Hence, you should commit your progress directly to this repository each time you make advancements.
- Before delivering the assignment in Blackboard, make sure your project has all the files it needs. Delete any file, folder or info that is not needed (this is `.git/` folder, `node_modules`, etc.). Zip the project and upload the file to Blackboard.
- Don't forget to add all the `API` specs in `documentation` and your query collection in the `postman` folder.