# CPSC 2600 Individual Final Project

The purpose of this project is for you to have a chance practice the techniques we have learned in this class. Your job is to create a fully-functional web application with database, server-side and client-side components. The project can be about whatever you want - choose a topic that interests you!

Here are the technical requirements:

- Must use a MongoDB database with Mongoose. (Or if you'd like to try a different DBMS, please talk to me first.)
    - Must use at least two schemas with some kind of relationship between two types of documents (such as embedded subdocuments or references.)
- Use Node.js and Express on the backend.
    - Create an API that will be used by your client-side application:
        - Don't worry about authorization or authentication;
        - Data submitted through the API must be validated and sanitized;
        - Your API must have GET and POST routes;
        - Use REST conventions when designing the API. Don't worry about more advanced REST features like HATEOAS, caching, asynchronous responses, etc. but do implement the following:
            - responses in JSON format
            - URL versioning
            - resource-based endpoint names
            - sensible response codes
            - filter parameters for large collections
    - Your server-side application must be organized with routing and controller logic separated. Use middleware where appropriate - don't write all logic in one file!
- You must have a client-side, single-page application created with React:
    - Your React application must have some kind of stateful logic, and
    - at least one form, using the controlled component pattern.
- You may ***not*** use a project (or any code from a project) that you created in another course. All work must be original and unique to this course. All the work must be created by you alone; this is ***not*** a group project.
- Use environment variables for credentials. Use Webpack to bundle your front-end code. Be sure to style your front end using CSS.
- When you've almost completed your project, meet with me (during our regular lab time) to review it together. Note that our last lab session of the semester will probably be extremely busy, so try to do it before then.
- Your project must be deployed to the web using some online hosting service. I will provide instructions later for using **Heroku** to host your application, but you may use another service if you prefer.

Some other notes:

- The challenge level and originality of your project will be factors in your grade. As you only have four weeks to implement this (while also working on other assignments and projects) I'm not expecting anything too complicated, but it should at least demonstrate all topics we've learned in the class and be significantly different than the assignments. On the other hand, don't bite off more than you can chew! I'd be happy to review a project proposal if that would be helpful.
- The user interface should be relatively attractive and easy to use. Each action taken by a user should have an obviously-perceivable effect. Error messages should be rendered in the client and visible to the user, including validation errors. Raw JSON responses should NOT be shown.
- Your application should be well designed and engineered, from a software design perspective. Ensure that your code is appropriately modularized so that each part does only the thing it's responsible for, and apply abstraction appropriately. (For example, the client-side code should not require any special knowledge of how the database works.) Format your code (you can use an auto-formatter) and provide comments where necessary. Delete code that has been commented out.
- Test your application thoroughly! Get your friends to use it. Test it after you've uploaded to the web. Make sure it's impossible to break the application, since I will intentionally be trying to break it! Use Postman to test your API endpoints.

## Hand In

Please follow this procedure:

- Place a clickable hyperlink to your project in the comments of your Brightspace submission.
- Rename your project folder to **project-firstname-lastname** and delete the node_modules folder. Make sure all necessary modules are saved in package.json.
- Include in package.json scripts that are necessary to build and start the application.
- Include a **readme** file (in md, html, or txt format) with basic documentation, including the following:
    - A clickable hyperlink to your project;
    - Instructions for building and running the project;
    - A brief, one-or-two sentence description of your application;
    - A list of features you're proud of that you would like me to consider;
    - Instructions on how to use your application;
    - API documentation, including the following:
        - Endpoints and methods with a brief description of each;
        - Response format;
        - Expected POST body format;
        - Examples on how to use each endpoint;
- Zip your folder and submit it to Brightspace.
- Feel free to include a link to your Github or other repository if applicable.

## Grading:

- **Technical requirements:**
  - [3 marks] API using REST conventions
  - [2 marks] Database with relationship between entities and schemas
  - [1 marks] Data validation implemented
  - [2 marks] Back-end application is well designed
  - [4 marks] single-page client-side React application with state and at least one form
  - [2 marks] Application is easy to use
  - [2 marks] Review meeting with me has been completed

  Total: 16

- **Challenge and originality** - your grade in this section will be **multiplied** by the above grade to form your overall project grade. Here are the possible grades you could receive:
  - **[1.063]** A moderately complex project with non-trivial implementations of all technical requirements and a highly original concept.
  - **[1.0]** A simple project with an original concept, or a moderately-complex project with some features borrowed from the assignments. Some features implemented trivially, meaning they've been implemented simply to meet the project criteria but are not essential to the overall application.
  - **[0.875]** A simple project with many features borrowed from assignments or implemented in a trivial way.

As the above criteria are somewhat subjective, please feel free to ask me to provide feedback on your project before submitting.

Grading example: if you scored **11/16** for **technical requirements** and **0.875** for **challenge and originality**, your overall grade would be calculated as follows: **(11/16) * 0.875 = 0.60 =>** **60%**

Note that it's possible to get **more than 100%** for an exceptionally well done project. Up to **-4** may be deducted from the overall grade for improper hand in, not accessible on the web, missing documentation, disorganized files, messy or poorly-formatted code, bugs in the project, etc.