CPSC 2261 Final Project

Your task for the final project is to create a web application and deploy it on the web.

The topic and theme of your project can be whatever you want - choose something that interests you! (However, please do not use a project idea that you've already used previously in another course.)

There are two options for your final project:

- You can develop a web application using React where you consume a public API and display information from the API endpoints **(Option 1)**. This option is recommended for most people in the class. OR
- You can implement a full-stack web app using React on the front-end and ExpressJS on the backend with a MongoDB database (**Option 2)**.

If you are working in a group of 2, you must select Option 2.

Choose a topic that interests you and that you will enjoy working with over the remainder of the term. Ideally, this site could be a good addition to your portfolio. Think of this as a project that you can use to showcase your web development skills to potential employers.

Option 1: Front-end only React app | Consume a public API

Minimum set of features that need to be implemented:

- **Componentize** your React app.
    - At least five components;
    - At least one component with state;
    - Some props passed between components;
- **Event handlers implemented for interactivity**
    - **Adding, removing** *and* **editing functionality** *eg.* Think of the todolist app where you implemented functionality to add, remove and edit todo list items.
    - **Filtering** functionality *eg.* this could a search bar that allows you to filter a list of recipes or could be buttons which do the filtering (like the todolist app we worked on)
- Make use of **localStorage** to persist some state.
- **Consume a public API**:
    - You can find a list of free public APIs here. You can find APIs about various topics - ranging from animals, weather to cryptocurrency and finance.
    - You are free to use **axios** or **fetch** to make HTTP requests.
    - You should consume at least 3 separate endpoints.
- **Client-side routing**: You should create separate routes for different pages in your application.
- **Deploy** your web app to *Netlify* or some other hosting service.

Option 2: Full-stack web-app using the MERN Stack

Minimum set of features that need to be implemented:

- **Componentize** your React app.
    - At least three components;
    - At least one component with state;
    - Some props passed between components;

- **Event handlers implemented for interactivity**
    - **Implement at least 2 out of the 4 features:**
        - **Adding (1), removing (2)** *and* **editing (3) functionality** *eg.* Think of the todolist app where you implemented functionality to add, remove and edit todo list items.
        - **Filtering (4)** functionality *eg.* this could a search bar that allows you to filter a list of recipes or could be buttons which do the filtering (like the todolist app we worked on)

- **Client-side routing**: You should create separate routes for different pages in your application.

- Implement a **REST API** using Express with the following:
    - At least three GET routes that send data in JSON format;
    - At least one of these routes should send a collection and at least one route should send a specific member of a collection:
    - At least one POST route. (note that in real life, we would secure our API and authenticate users before allowing them to post data to our API, but don't worry about that for this project.)
    - Appropriate responses and status codes should be sent for the above routes;

- Connect your front-end React app to your Express backend using HTTP requests.
    - You are free to use **axios** or **fetch** to make HTTP requests.

- Use **MongoDB atlas** to persist application data..
    - The API should connect to a database.
    - Implement at least one collection;
    - Environment variables should be used for database credentials. (Remember to not commit these to any public repository, although you'll have to submit them to me when you hand in your project.)
    - Posted data should be validated and sanitized appropriately;

- **Deploy** your full stack web app to *Heroku* or some other hosting service.

Documentation

One of the routes/pages on your site should be called **Documentation**, and should include instructions for how to use your site, in addition to a list of features that you would like to contribute to your grade. If there's something you've implemented that you're proud of, mention it here, and include a link to that part of the site. A good practice would be to go through the checklist at the end of this document and make sure you address every point.

    - Let me know if there are any special instructions to build and run your project;
    - If it's not obvious, explain how your application meets the criteria for the assignment.
    - If it's not obvious, tell me how to use your application. (You may think it's obvious, but get a few friends to test it; they may find problems with the interface that you didn't notice.)

- **Try to be as original as possible**; I'll be considering creativity and challenge level when grading; try to use the tools we've learned throughout the course to create something different than what we've done in the assignments.

- **Think about usability when building your application**. A few things to consider:
    - When the user submits a form, there should be some indication on the page that the submission was either successful or unsuccessful.
    - Users should never be exposed to raw JSON data; make sure all responses are rendered as part of the application user interface.

Content:

You are **not** required to create original content for this site, although you may choose to do so anyway. Feel free to use text from Wikipedia or some other source that provides content under a Creative Commons license. You may use any images that are either in the **public domain** or licensed for use under **Creative Commons**. Of course, you are free to use your own images and writing if you choose.

Some sources of free images and video:

- https://unsplash.com/
- https://morguefile.com/photos/morguefile/ (images and video)
- https://videos.pexels.com/ (video)
- https://www.pexels.com/
- https://pixabay.com/
- http://nos.twnsnd.co/
- http://jaymantri.com/

- http://pickupimage.com/
- https://picography.co/
- http://www.lifeofpix.com/

Please note that if you use any content created by someone else, including text, images and videos, **you must provide proper attribution; failure to do so will result in a mark of 0 on the project. You must cite every source by providing a link to the exact, specific page where you got the content, even if the source does not require attribution.**

You are free to use any npm modules that you like to extend your project and add more features. Mention them in your project proposal.

Notes on Sources and Attribution:

- **Google Images is *not* a valid source.**
- **Rather than simply providing a link to a stock image site, provide a link to the page for the *specific* image you used. For example, providing a link to "pexels.com" would *not* be sufficient.**
- **If an image does not have any copyright or license information, you must assume that it is *not* free to use.**

Failure to follow these practices will result in a mark of 0 on the project and will be reported to the college as academic misconduct. Ask me if you have any doubts; I'd be happy to help!

You should have a page called **Sources** that lists all sources of images, text, code, and other content used by you that was created by someone else.

In addition to the above:

Treat this project as if it were a professional contract for a client. Test it thoroughly and make sure there are no bugs. Some things to think about:

- Test the application in several browsers to make sure there are no unexpected problems with unsupported HTML, CSS or JavaScript;
- Test your API thoroughly; make sure the validation middleware is doing its job;
- Test your application again after deployment; sometimes things don't work quite the same way in a live server environment.
- Make sure your code is well-organized, formatted, and commented.
- Make sure your application is well-designed: avoid repeating code; make sure each module or component does only the one thing for which it's responsible; make sure the application can be easily expanded without having to rewrite a bunch of code; make sure it would be (relatively) easy for someone who is unfamiliar with your code to understand it.

Bonus:

I may award a small bonus for functionality that goes beyond what we've learned in the class, or represents a high difficulty level. **If you have any ideas for bonus functionality, feel free to ask me or include it in your proposal**.

Example of bonus functionality:

- Implementing animations using a library like Framer Motion
- Writing unit tests using Jest or react-testing-library
- Writing snapshot tests using enzyme

Check List (Option A)

- [3.0 points] App Componentization, Organization/Architecture
- [4.5 points] Adding, Removing, Editing, and Filtering features
- [2.0 points] Client-side routing
- [2.0 points] Used localStorage to persist some data
- [4.5 points] Consumed public API
- [1.0 point] Deployed front-end app to Netlify (or other service)
- [1.0 points] Well designed, tested and free from errors
- [2.0 points] Application is original and demonstrates creative use of the tools learned throughout the course;

- Bonus
  - [+1-2 marks] Approved bonus functionality

Total: 20 points

Check List (Option B)

- [2.0 points] App Componentization, Organization/Architecture
- [3.0 points] Adding, Removing, Editing, and Filtering features (>= 2)
- [1.5 points] Client-side routing
- [6.0 points] REST API using Express
- [2.5 points] Used MongoDB to persist some data
- [2.0 points] Deployed full stack app to Heroku (or other service)
- [1.0 points] Well designed, tested and free from errors
- [2.0 points] Application is original and demonstrates creative use of the tools learned throughout the course;

- Bonus
  - [+1-2 marks] Approved bonus functionality

Total: 20 points

Hand In:

- Remove any files that are not being used in the final version of your site, and make sure your code has been cleaned up/formatted/commented.
- Zip all necessary files (except node modules), rename the folder to firstname-lastname, and hand in the archive to D2L. Remember to include your deployed URL.
- If there are any special instructions, please include them in the comments.

*Please note that I may deduct up to -10% for poorly organized/commented/formatted code and for improper hand in.