# HTTP5222 JS/XML Assignment 2 (React)

|  |
| --- |
| **Choice 1:**  Using React, develop the front-end for your portfolio site. Using the JSON API you created in your PHP course (PHP backend), retrieve the data and use it in your React site. This is along the same principle as a headless CMS site. So, for example, the PHP API you create might return a list of projects like:  [  {  project\_name: “…”,  project\_thumbnail: “…”,  project\_url: “…”,  project\_description: “…”,  project\_languages: “…”  },  …  ]  A second API endpoint could be used for skills (e.g. to list your skills like skill name and icon image path).  NOTE: If you would rather use MongoDB, Express, and Node for your backend API instead this is also an option. |

|  |
| --- |
| **Choice 2:**  If you already have a portfolio site up and ready and want to do something else, you can create a simple API using Node, Express and MongoDB and use React for your front-end (MERN stack). If you choose this option, then you’ll need to submit both the Node/Express app files along with the React app files. For MongoDB, you should have at least two collections (e.g. if a store, you can store products and other data you’d need to list, such as store locations). |

Suggestions:

* To start, you can make a rough wireframe on paper to organize your thoughts as to the info you'll need and where you'll display it.
* Even if you don't have the JSON API (PHP) complete or if you don't get it deployed, you can save your JSON as a file and retrieve the data via an HTTP request (e.g. using a Fetch request).
  + When the API is deployed, you can then just change the request URL to the live one and not for your local file.

**Requirements:**

1. Website must be built using React.
2. Website data must be retrieved from a backend API (i.e. your PHP API endpoints or Express API endpoints spit out JSON). In other words, your React portfolio site will make API requests to your backend API to retrieve the site data. (If, for the PHP API, you **don't** manage to get the JSON API deployed, copy the JSON output into JSON files and make requests to those files.)
3. Include important information such as:
   1. If creating a portfolio site:
      1. Your name
      2. Your desired role (e.g. web developer, web designer, full stack developer, etc.)
      3. Your projects including important info about each project such as (for example):
         1. Project name
         2. Screenshot
         3. URL (if applicable)
         4. Short blurb about project
         5. Language listing
      4. Some way to contact you (contact info and/or a contact form).
   2. If creating a different site, consider the important information you’d need and (possible) pages you’d need.
4. Deploy your website. (You can use Vercel. Note that if a portfolio site, using a custom domain is nicer.)

You will be marked on the criteria below. Note that the original rubric is out of 15 (see below) but this has been **scaled up to being out of 20 in Blackboard**.

* **Code quality (5):** Does it work? Does it follow coding best practices? Are there helpful comments? Formatting?
* **Design (5):** Is the design professional? Does the design work for the type of website (e.g. portfolio)?
* **Usability (5):** Is it easy to use? Is it clear what the site is about? Are important pieces of info available?

This assignment is individual and is worth 20% of your final grade. For your code, you can put your React app code on GitHub and just submit your GitHub link so I can view the code. In a README, include the URL to your site online. If you don't manage to get the site online by the due date, include any information I may need to run the site locally. (If you choose to do a MERN stack application, submit files for both the backend API and the React app so that I can run and view the data.)

Due date: **August 16, 2024 @ 11:59pm** (**DO NOT submit this assignment late** as this may result in your assignment not being accepted.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **Excellent**  **(5 – 5 points)** | **Good**  **(4 – 5 points)** | **Satisfactory**  **(3 – 4 points)** | **Somewhat Unsatisfactory (2 – 3 points)** | **Unsatisfactory**  **(0 – 2 points)** |
| **Code Quality** | Code works, is efficient, well-commented and formatted. Code follows best practices. | Code works, is commented and formatted well. Code mostly follows best practices but for a few minor issues. | Code is a little buggy, is not always commented or formatted well but a good attempt. Code not always efficient. | Code is buggy and poorly commented. Some issues following best practices. | Code doesn't work and is not commented. Code has many major issues. |
| **Design** | Design is professional. | Design is mostly professional but with a few minor issues. | A good attempt but feels like student work. | Design is not really complete and is obvious it's student work. | Design not really fleshed out at all. Design is not complete. |
| **Usability** | Website is easy to use and follow and is accessible. It makes sense for the site subject matter (e.g. a portfolio site). | Website is easy to use and follow but with some minor issues. | Website is mostly fine but with many minor issues or at most one or two major issues. | Website has some major issues around usability/accessibility. | Website is confusing/hard to use. Website not accessible. |