Lab 4: Choose Your Own Web Dev Adventure [HTML, CSS, JavaScript, Node.js, APIs]

Learning Outcomes:

- Learn to work with asynchronous requests.
- Apply advanced JS techniques and incorporate the use of Node.js and APIs
- Learn to interact with APIs using HTTP methods (e.g., GET, POST, PUT, DELETE), and data formats like JSON.
- Develop your problem-solving skills by debugging code and troubleshooting API issues.
- Understand the importance of unit testing both client-side and server-side code, as well as integration testing.

Instructions:

- For this lab, you will be tasked with creating an engaging and creative web application using your knowledge of HTML5, CSS and styling frameworks, and advanced JavaScript. Additionally, as part of this lab, you will also be incorporating the use of Node.js and external APIs for developing a small web application.
- In this lab, you will need to choose your web development adventure based on the following application options:

    1. **Meme Generator App**

        a) Build a web-based meme generator that allows users to upload images, add custom text , and apply some effects.

        b) You may work with any API to help you in this task. Some examples of APIs include **Unsplash Image API**, **Pixabay**, **Memegen.link API** and **Supermeme.ai API**.

    2. **Recipe Recommender App**

        a) Create a personalized recipe recommender app that suggests recipes based on user-defined dietary restrictions (e.g., vegetarian, gluten free) and available ingredients.

        b) You may work with any API to help you in this task. Some examples of APIs include **Spoontacular API** to fetch recipe data and generate recommendations, **Edamam Recipe Search API**, and **TheMealDB**.

    3. **Music Discovery App**

        a) Develop a music discovery application that recommends new artists and songs based on user preferences (e.g., favourite artists or genres).

b) You may work with any API to help you in this task. Some examples of APIs include **Spotify Web API** (Free Tier), **MusicBrains API** to fetch Artist information and discover/recommend similar artists, or **Discogs API**.

- Regardless of the adventure option you choose, you will be expected to meet the following requirements:

1. Front-End Requirements

a) Strive to develop a user-friendly interface using HTML5, CSS, and JavaScript, providing clear instructions (through the UI) and helpful feedback to your user. You may also integrate the use of Styling Frameworks such as Bootstrap, Tailwind CSS, Semantic UI, Materialize, Foundation.

2. Back-End Requirements

a) Develop a Node.js server for handling API requests, data processing, and server-side logic (if needed). You may use Express.js (if needed) to structure your server.

b) Integrate and use your chosen API (based on your choice of adventure) on both the client and server-side.

c) Use JavaScript to interact with the Node.js server via API calls (e.g., using Fetch API)

d) Properly deploy your application, ensuring proper configuration for both front-end and back-end components of your application.

3. Testing and Error Handling Requirements

a) Implement appropriate Error Handling throughout your application.

b) Properly test your application by writing unit tests for both client and server-side code, testing the interaction between the client and server, and testing your application for cross-browser compatibility.

Note: The purpose of a **Unit Test** is to allow you to verify the individual components of an application by isolating the component being tested (i.e., the Unit) from other parts of your code (e.g., dependencies, APIs, databases), in order to help you to pinpoint if and where an error occurs. In other words, the main goal of Unit Testing is to verify that an application will work as expected. For this requirement, you are NOT expected to write extensive documentation for your Unit Tests, but rather provide a brief description (in your README file) of what you did to properly test your application (e.g., which components needed to be tested, if needed then how were they isolated, did any errors need to be addressed or did the component work as expected).

4. Accessibility Requirements

a) Implement the WCAG guidelines to ensure basic accessibility considerations are met.

b) In your README file, properly cite your work by including a brief description of the application

you have created, along with the list of APIs and other technologies used, and a brief mention of any issues or limitations you encountered along the way and how you addressed them (if you were able to).

- For this lab, **you are not expected to implement a user profile management system** (e.g., login, registration, profile page) or maintain any session information or store any user information. Instead, y**ou are expected to focus on the main purpose and core functionality of your application** based on the adventure you have chosen. In other words, your meme generator should generate a meme (based on the uploaded image), your recipe recommender should recommend a recipe (based on some parameters), and your music discovery app should recommend an artist or song (based on user preferences).

  Note: Though you have complete creative freedom for this lab, you are strongly encourage to keep in mind the timeline for this lab and properly manage your time accordingly.

- You have complete creative freedom for this assignment. You are encouraged to work towards an aesthetically pleasing website that applies the design and development principles you have learned thus far in your academic and/or web development career to create a user-friendly design for your lab that presents a visually appealing interface for your IMS. You may use Creative Commons images and/or logos with proper author attribution (provided through code comments, and/or **README.txt** file).

  Note: Do keep in mind that as part of this assignment, you are expected to work individually, you may discuss ideas with your classmates, but do refrain from sharing any code.

- Your lab should make use of `error handling techniques` to handle unexpected situations such as invalid input, or missing data, and test your work to verify the correctness of your functions and DOM interaction. (e.g., unit tests, JS console)

- As in previous labs and as specified in the **Submission Section**, for Lab 4 you will be expected to submit a README file, a Git Repo, and a remotely accessible lab on **Netlify**. ***See Brightspace Lab 4 module.***

- For the purpose of this lab, you will be deploying your work through **Netlify** (not Timberlea). Submission instructions for Netlify can be found in the **Submission Section** of this handout.

  Note: You must ensure that the URL you receive for your deployment (through Netlify) is included in your README file. Failure to submit your work through Netlify will result in a grade of **ZERO (0).** Failure to ensure your work is remotely accessible through a web browser, using the URL you specify in your README file will result in a grade of **ZERO (0).** If you can see your work through the URL you provided, then the TAs and Instructor will also be able to view and mark it.

- **Include in your README.txt file,** the URL from which your lab can be accessed. All pages you develop for this assignment will need to be accessible through that link.

Note: If you decide to use and modify any existing code, e.g., code found on online or printed sources or code used during in-class/tutorial examples, you are expected to provide author attribution in your code comments, and a more detail explanation of your sources in your README file (i.e., providing an explanation of why the piece of code is necessary for your work, where, how and why the code or section of code was modified). Keep in mind that simply stating "code was modified" does not provide sufficient information required in your programming assignments.

Submission:

- For this lab, you will need to **submit your work through Netlify and GitHub, Brightspace, and GitLab as follows:**

Submitting your Work through Netlify

Step 1: Creating a repository on GitHub

- Create a **copy** of your Gitlab repository on your personal GitHub account.
- Netlify will not allow you to deploy the app directly from Gitlab (partly due to how we have structured our GitLab project repo for this course). While you will use GitHub to deploy your app through Netlify, **code assessments will be done directly through GitLab**.

Step 2: Create an account on Netlify

- As shown on Figures 1 and 2, create a Netlify account and get it verified from your registered email address. Answer sign-up questions if required.

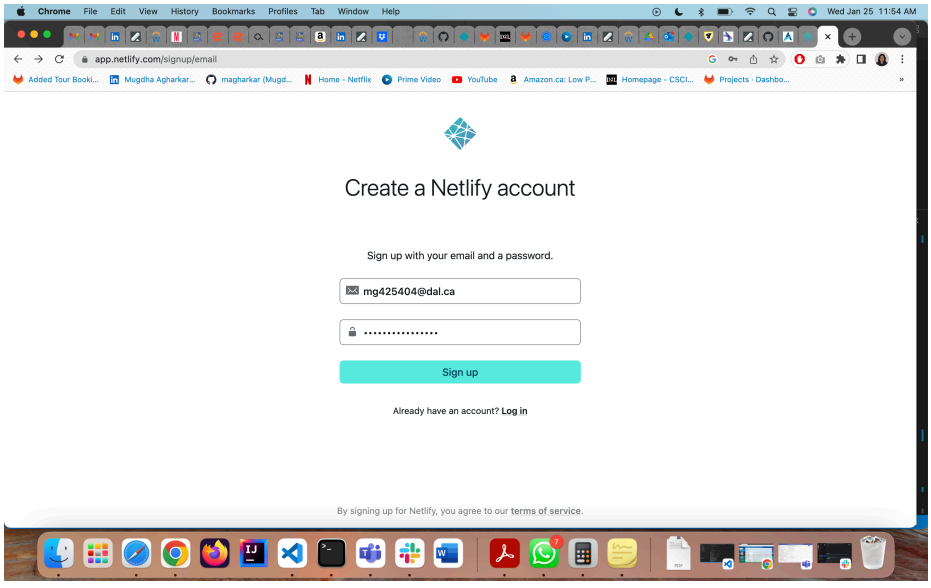

Figure 1. *Creating a Netlify Account.*

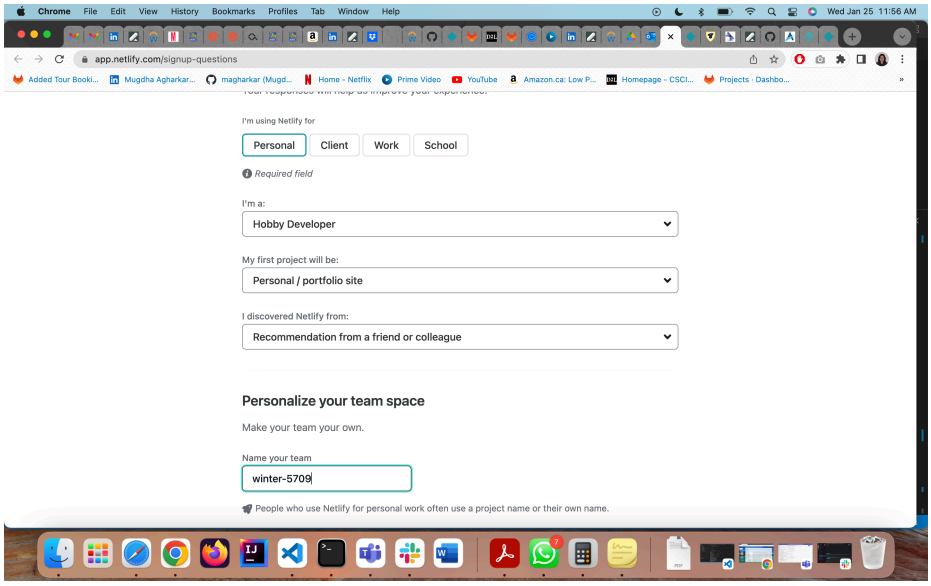

Figure 2. *Creating a Netlify Account II.*

Step 3: Deploying the project from GitHub

- After signing up, you will be redirected to a quick import page, where you can click on 'import an existing project'. Note that you can also do this later in the application if you do not have a repository ready. *See Figure 3.*

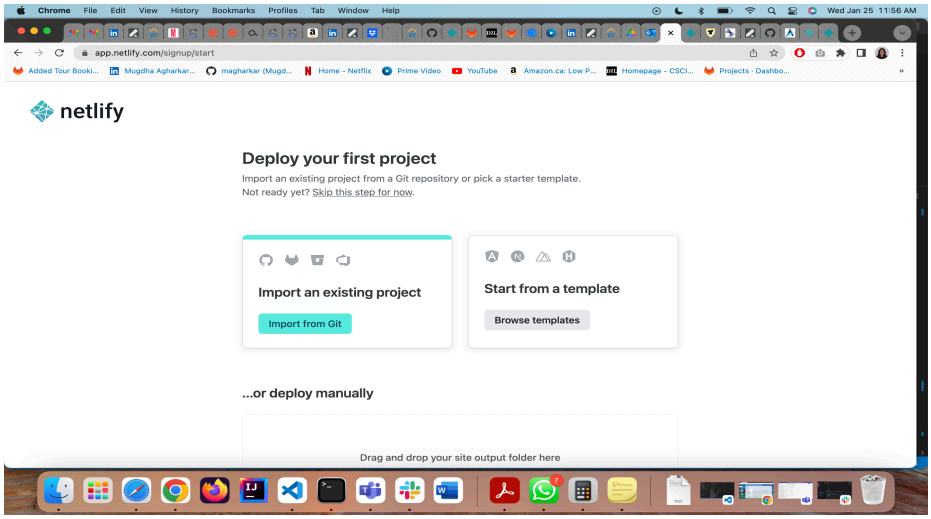

Figure 3. *Deploying your first project on Netlify.*

- Authorize Netlify to connect to your personal GitHub account. *See Figure 4.*

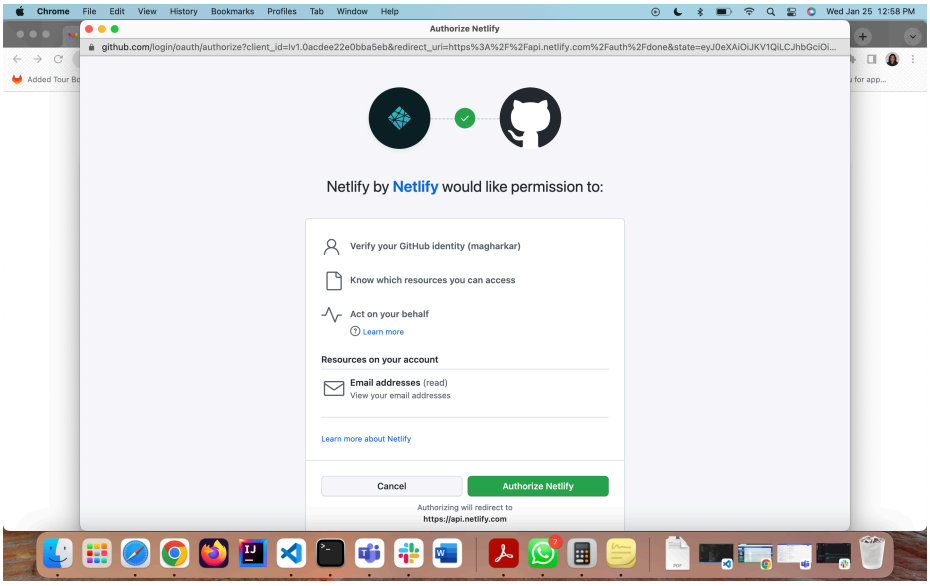

Figure 4. *Connecting your GitHub account to Netlify.*

- Select the branch and build options. *See Figure 5.*

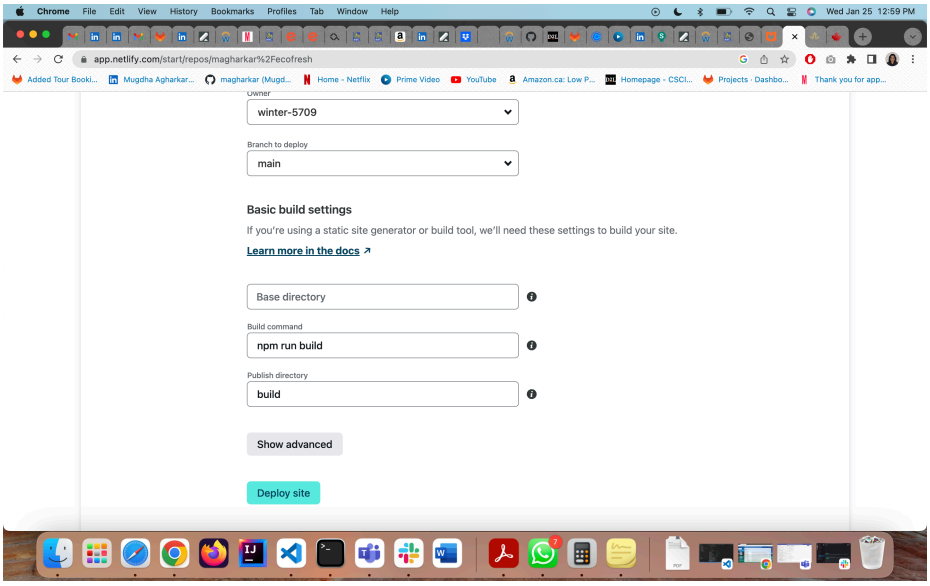

Figure 5. *Selecting your branch and build options on Netlify.*

- Done! Your site will be deployed in a few minutes. *See Figure 6.*

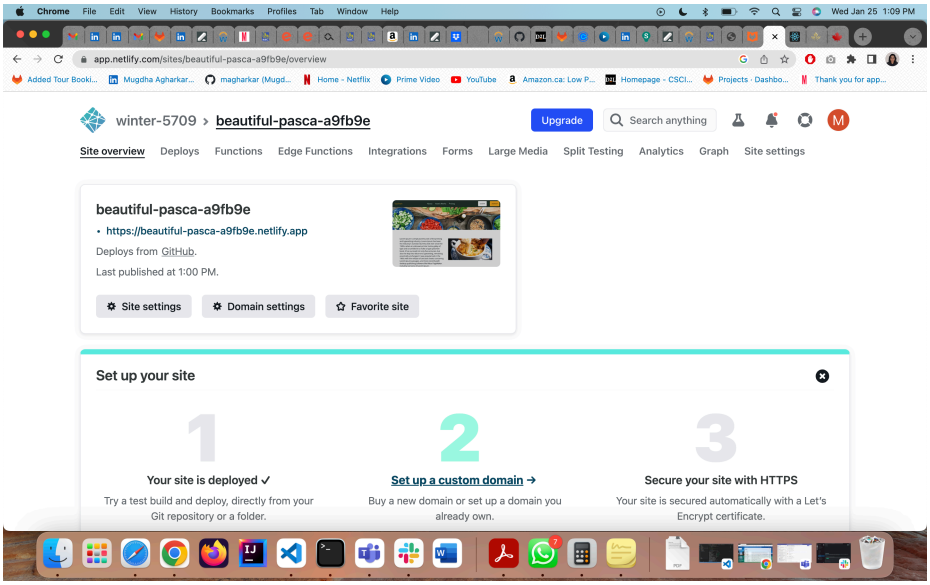

Figure 6. *Completing deployment of your application on Netlify.*

- Visit **https://<app_name>.netlify.app/** on any browser and ensure you can view your work.

Note: Visit your app's URL on any browser and ensure you can view your work. Failure to submit your work through Netlify will result in a grade of **ZERO (0)**. Failure to ensure your work is remotely accessible through a web browser, using the specified URL will result in a grade of **ZERO (0).**

- Test your lab to ensure cross-browser compatibility.
- **Include the URL to your Web App in your README file**

Submitting your Work through GitLab

- Create a **git** repository on the **FCS Gitlab site**, and clone it to your local system using the following command:

```
git clone *your repo https link*
```

- Copy the HTML or JS file to the local copy of your repo and push it to the git repo using the following commands:

```
git add .
git commit -m "your commit message"
git push
```

- Setup your GITlab repo as a private project and add the course **Teaching Assistants (TAs) and Instructor as 'Maintainers'** to your project, using their **CS IDs.** *See Lab 1 Brightspace module.*

Note: The CSIDs for this course will be provided during our lab session. Failure to add the course CS ID as 'Maintainer' for your work on GitLab will result in a maximum possible grade of 50%.

Submitting your Work through Brightspace

- Download the **README template** available on Brightspace. *See Resources section on left-hand side menu on Brightspace.* There are TWO versions of this template, you may use whichever you feel more comfortable with.

- Edit the README template to include any citations for your code and/or images used for this Lab.

Note: If the work you are submitting as part of your Lab is work done by you without the use of any external sources, then please specify so within your README file.

- Depending on the version of the template you chose, rename your README file as:
  L#_LastName_FirstName_README.txt

Note: Ensure your README file includes the URL to your Lab for remote access.

Marking Rubric:

The following grading criteria will be used for marking your lab:

Dimensions	Does Not Meet Expectations	Somewhat Meets Expectations	Meets Expectations	Exceeds Expectations
Functionality (25%)	Core functionalities (e.g., recommendation) have major errors or is missing, application does not function as intended. **(0 - 4 points)**	Core functionalities partially implemented with minor errors. Some features missing, noticeable functionality limitations. **(7 - 14 points)**	All core functionalities are implemented correctly with minor errors, overall application works as expected. **(16 - 22 points)**	All core functionalities implemented correctly with no errors. Goes beyond basic requirements. Implements innovative features. **(23 - 25 points)**
API Integration (20%)	No API integration or incorrect use of API. Lab is unable to retrieve or process data from the API. **(0 - 4 points)**	Basic API integration with minor errors in data retrieval or processing. Lab makes limited use of API features. **(5 - 10 points)**	Correct API integration, API effectively retrieves/process data. Lab demonstrates good understanding of the APIs capabilities. **(11 - 15 points)**	API integration done seamlessly and creatively. Uses advanced API features, demonstrates deep understanding of the API. **(18 - 20 points)**
UX / UI Design (20%)	UI is poorly designed, user feedback is confusing, does not implement WCAG, or difficult to use. **(0 - 3 points)**	UI is basic and functional but lacks polish. Web app is somewhat accessible but has some visual or feedback inconsistencies. **(4 - 7 points)**	UI is well-designed, visually appealing, easy to use, and accessible (applies WCAG). Provides a good user experience. **(8 - 12 points)**	UI exceptionally designed, applies innovative design elements, highly accessible with meaningful user feedback. **(13 - 15 points)**
Code Quality (20%)	Code is poorly written, difficult to read, understand and maintain. Lacks proper indentation and meaningful variable names, extensive use of unnecessary comments. **(0 - 4 points)**	Code is somewhat readable, minor issues with formatting, comments, or variable naming. **(7 - 14 points)**	Code is well-written, easy to read, and well-documented with only necessary comments and meaningful variable names. **(11 - 15 points)**	Code is exemplary, highly readable, well-structured, and maintainable. Clearly demonstrates best practices in coding style and conventions. **(18 - 20 points)**
Testing (15%)	No testing performed or inadequate testing done. Web app contains numerous bugs and errors. **(0 - 3 points)**	Basic testing performed, bug some bugs still exist. Limited test coverage implemented. **(4 - 7 points)**	Application has been thoroughly tested, with minimal or no critical bugs. Good test coverage with appropriate unit tests. **(8 - 12 points)**	Comprehensive testing with extensive unit tests and thorough cross-browser compatibility testing. **(13 - 15 points)**
Cross-Browser Compatibility & W3C Compliance	Lab is not cross-browser compatible, noticeable/ distracting differences visible AND lacks W3C compliance. **(-50 points)**	Student's lab is cross-browser compatible BUT lacks W3C compliance. **(-25 points)**		Student's lab is cross-browser compatible, no noticeable/distracting differences visible AND is W3C compliant. **(0 points)**
Git Repository	Code is not pushed to repo OR Student did not provide Maintainer access to their GitLab repo. **(-50 points)**			Code is properly pushed to git repo, AND provided Maintainer access to their Lab's GitLab repo. **(0 points)**

Netlify	Code is not uploaded on Netlify OR the URL doesn't load the HTML file. **(-100 points)**			Code is properly uploaded on Netlify and URL to file is provided as per requirements. **(0 points)**
README.txt	Student's did not submit a README.txt file and/or did not edit the template as expected. **(-100 points)**		Student's submitted a README.txt file that is incomplete or is missing the lab's URL. **(-50 points)**	Student's submitted a README.txt file properly edited to include all sources used. **(0 points)**