Lab 6: Setting up and Deploying a React Portfolio [JS, React, Bootstrap, Netlify / AWS Amplify]

Learning Outcomes:

- Apply your knowledge of React fundamentals to create a multi-page web app, using functional components, props, and routing.
- Implement UI design principles and integrate the use of Bootstrap to create a visually cohesive user interface.
- Create interactive elements within a React application, demonstrating your design and development skills.
- Test and troubleshoot common React development errors and issues, demonstrating your ability to debug and maintain a React project.
- Evaluate and deploy a fully functional React application to a cloud platform, and assess the success of the deployment.
- Learn to create accessible designs that properly applies the Web Consortium Accessibility Guidelines (WCAG) to ensure users of all abilities are properly supported in their web experience.

Instructions:

- For this lab, you will be tasked with creating an engaging and creative React-based Portfolio site using, in addition to your React skills, your knowledge of HTML5, CSS and styling frameworks (e.g., Bootstrap), and advanced JavaScript. This lab is the first of a set of three connected deliverables (i.e., Lab 6, Lab 7, and Lab 8), which means you will be adding some features and functionality from one lab to the next.

  Note: While responsiveness is not a requirement of this lab, it is important for you to design with responsiveness in mind as it will be required in a following Lab.

- In this lab, you will be creating a React-based personal portfolio site that could be use to market your skills and showcase your work when job hunting. Lab 6 will focus on the basics, as such you will be expected to meet the following requirements:

  1. **Front-End Requirements**

     a) Set-up a React project using Vite or Create React App (CRA).

     b) Install React Router and create at least THREE (3) pages:

        1. **Home:** This page will be your home or landing page and should provide a brief introduction.
        2. **About:** This page should include information related to your education, experience, career goals, and technical skills (e.g., programming languages, tools/software). In particular, this section should highlight your passion for technology and your related field, as well as any specialized skill(s) or areas of expertise.

3. **Projects:** At this time, you may use this page as a placeholder with some static content. However, this page should included detailed descriptions of projects (e.g., problem solved, technologies used, your specific role) that highlight your practical skills, problem-solving abilities, and enthusiasm for learning. Projects use may be academic or personal projects, as well as projects from any internship or part-time work experience.

c) Create **Header** and **Footer components** to be re-used across pages.

1. **Header Component:** Your Header component should implement a navigation bar.

d) Apply AND customize the use of Bootstrap 5 (or an alternative CSS framework of your choosing) to create a visually appealing and engaging design to your personal portfolio website.

1. **UI Design:** You should strive to implement a clean and modern design that with a visually appealing and intuitive layout.

Note: Though responsiveness is not a requirement of this lab, it will be required in a later Lab. As such, you are encouraged to create a design that considers a future application of responsive development approaches.

2. Back-End Requirements

a) At this moment, Lab 6 will be focusing on the Front-End of your personal portfolio site. Therefore, **NO Back-End setup is required** at this point. However, you are encouraged to structure your React project in Lab 6 so that API integration (e.g., fetching dynamic content) can be added in Lab 7 with minimal to no re-structuring required.

3. Testing and Error Handling Requirements

a) Create a 404 Page to re-direct all unknown routes to this page by default, **DO NOT** hard-code this requirement.

b) Your 404 Page should display a relevant message and link or button or navigation bar that helps direct the user back to the Home Page.

c) Properly test your application by writing unit tests for both client- and server-side code, testing the interaction between the client and server, as well as 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) Your Personal Portfolio Website must implement the use of **WCAG Guidelines** to ensure accessibility considerations are met.

- 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 6 you will be expected to submit a README file, a Git Repo, and a remotely accessible lab on **Netlify OR AWS Amplify**. ***See Brightspace Lab 6 module.***

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

Note: You must ensure that the URL you receive for your deployment (through Netlify of AWS Amplify) 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 OR AWS Amplify 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.

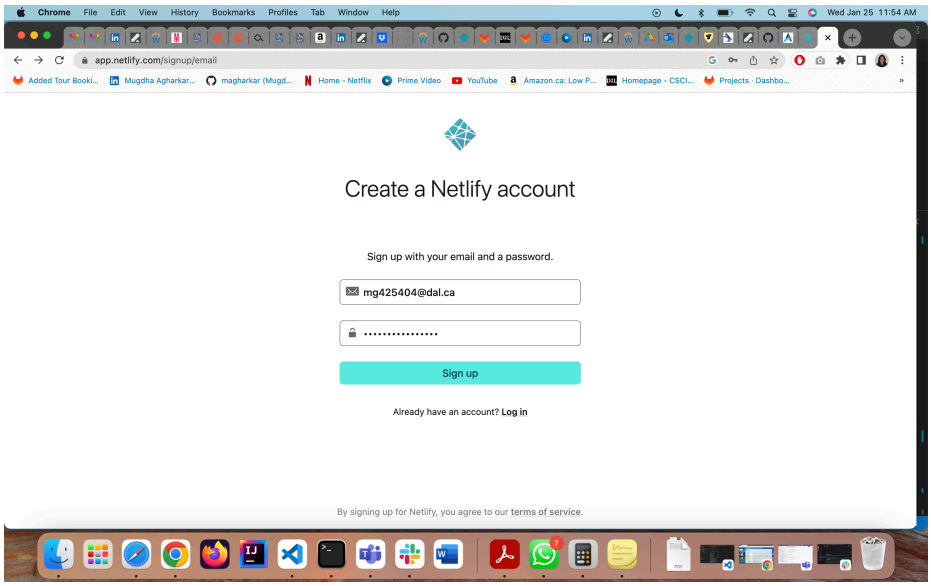

Figure 1. *Creating a Netlify Account.*

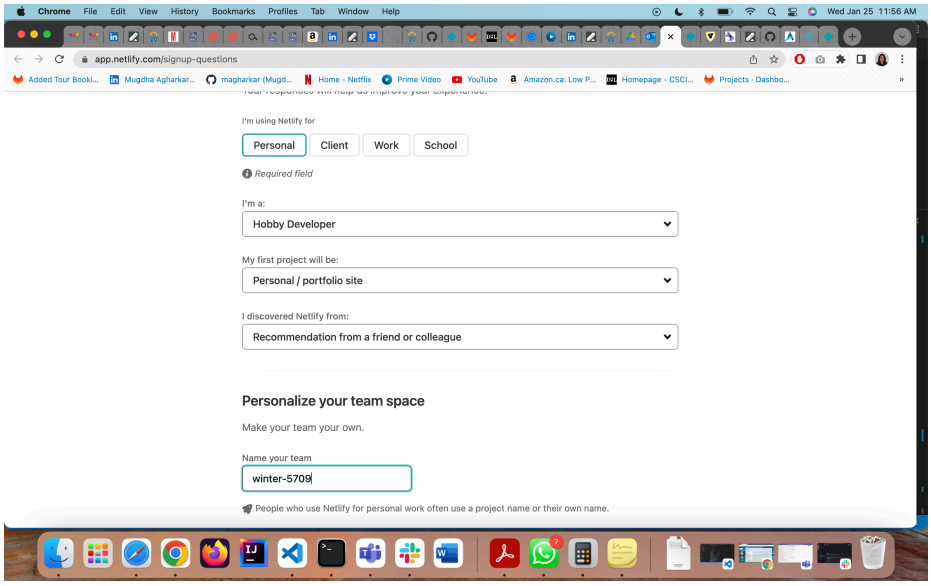

Figure 2. *Creating a Netlify Account II.*

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.*

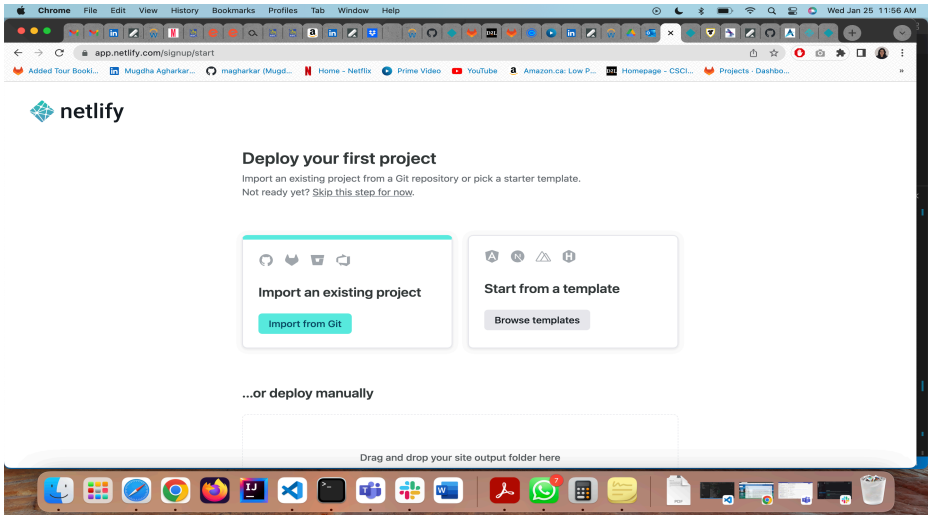

Figure 3. *Deploying your first project on Netlify.*

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

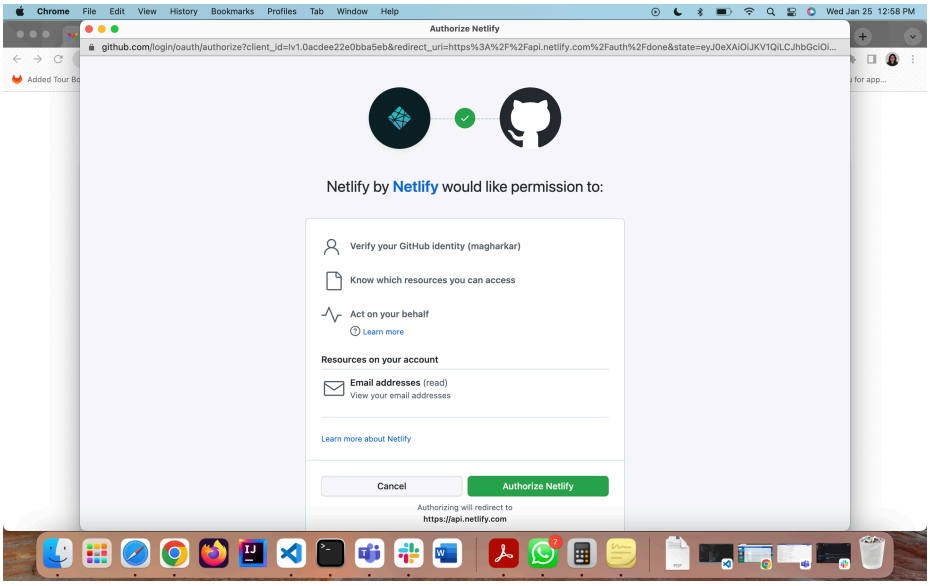

Figure 4. *Connecting your GitHub account to Netlify.*

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

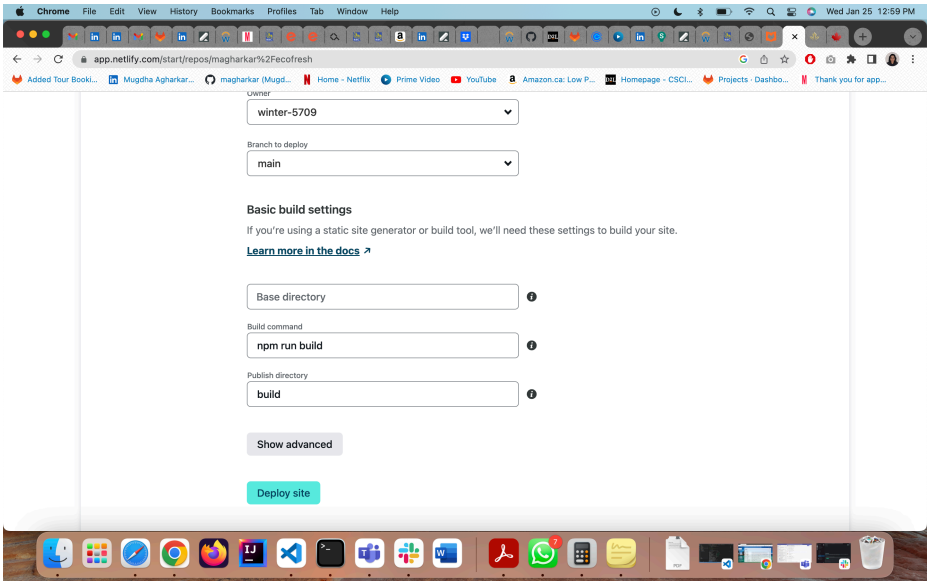

Figure 5. *Selecting your branch and build options on Netlify.*

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

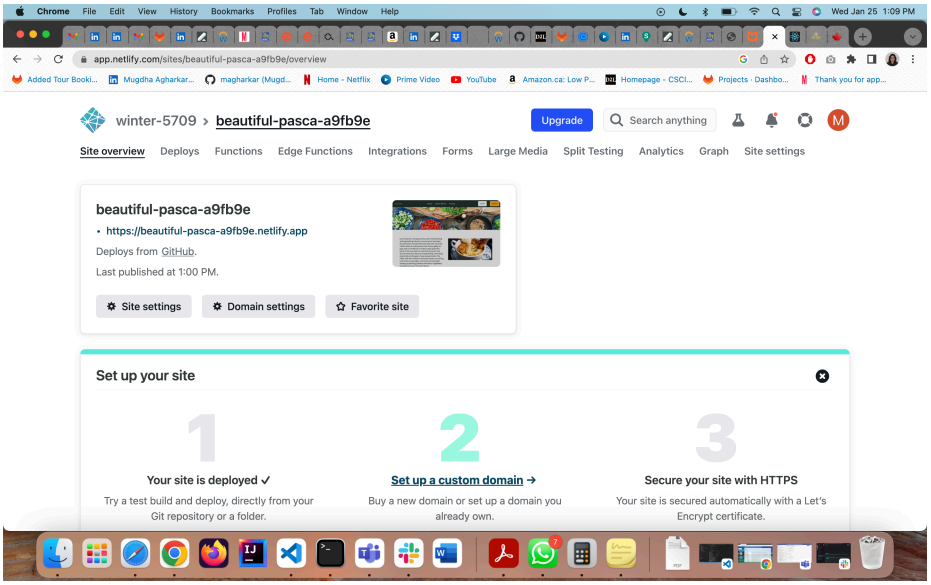

Figure 6. *Completing deployment of your application on Netlify.*

- 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 AWS Amplify

Step 1: Creating a repository on GitHub

- Create a **copy** of your Gitlab repository on your personal GitHub account.

Step 2: Create an account on AWS Amplify

- AWS Amplify is one of the services in the **AWS Cloud suite**. So you will need to sign-up for AWS first.

    1. Go to **AWS** create an account. During the sign-up process, AWS will ask for your bulling details. DO NOT worry, AWS offers a free tier which will meet the requirements of this lab.

    2. Choose a support plan, select "Basic Free" for now, and complete the sign-up process.

    3. Wait for AWS to activate your account (this may take a few minutes).

    4. Go to the AWS Management Console

Step 3: Create an app on Amplify

- In the AWS console, search “Amplify” in the search bar and select this option.
- Click "Get Started" under "Host your web app”.
- Choose "GitHub” and Authorize AWS to access your GitHub repository (this is a similar step to what you did to setup Netlify).
- Select your **repository** (e.g., my-react-app).
- Choose the "main" branch.

Step 4: Configure Build Settings

- Keep everything in the default settings.
- As illustrated in Figure 7, ensure the build settings shows the following:

amplify.yml

```yaml
version: 1
frontend:
    phases:
        preBuild:
            commands:
                - 'npm ci --cache .npm --prefer-offline'
        build:
            commands:
                - 'npm run build'
    artifacts:
        baseDirectory: dist
        files:
            - '**/*'
    cache:
        paths:
            - '.npm/**/*'
```

Figure 7. *AWS Amplify Build Settings Configuration.*

- Save and deploy. Amplify will **automatically build and deploy** your app. This can take **2-5 minutes**. Once done, your app is **LIVE**.
- You can check all the details of the app just deployed in the console. Simply click the app in the Amplify app list (if you just deployed the app, then you are in the page already). If you see green, it means successfully deployed. See Figure 8.

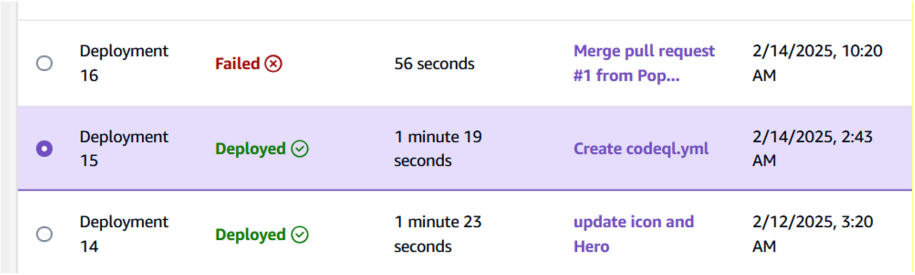

Figure 8. *AWS Amplify Deployment Confirmation.*

Step 5: Get Your Live URL

- Once the deployment is complete, **Amplify will show your live URL**. It will look something like: **https://your-app-name.amplifyapp.com**
- **Click the link** to see your deployed React app! This is the link you need to have in the 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
React Core Functionality (30%)	Significant errors in component structure, props, and routing. Application is not functional. **(0 - 10 points)**	Basic component structure present, but some errors are present. Routing is partially functional. **(15 - 20 points)**	Correct component structure, effective use of props, and functional routing. **(23 - 27 points)**	Highly organized, efficient component structure. Seamless and robust routing implemented. **(28 - 30 points)**
UX / UI Design (30%)	Missing header/footer, ineffective or incorrect use of Bootstrap, poorly implemented layout and design. Does not implement WCAG Guidelines. **(0 - 10 points)**	Header/footer present, but with flaws. Basic Bootstrap use with inconsistencies. Layout and design somewhat functional with issues. Some implementation of WCAG. **(15 - 20 points)**	Well-structured header/ footer components. Effective and consistent use of Bootstrap. Visually appealing layout. Implements WCAG Guidelines. **(23 - 27 points)**	Exceptional UI design with advanced Bootstrap features. Highly user-friendly UI and layout. Creative and polished design. Exceeds WCAG expectations. **(28 - 30 points)**
Page Content & Functionality (20%)	Does not meet 3-page requirement. Pages lack content AND/OR functionality. Content is copied of irrelevant. **(0 - 5 points)**	Meets the 3-page minimum requirement. Content is basic or incomplete. Some functionality present but with errors. **(10 - 14 points)**	Meets the 3-page minimum requirement. Pages content is unique and relevant. Pages are fully functional. **(15 - 17 points)**	Meets or exceeds 3-page requirement. Page content is engaging, original, well developed and presented. Demonstrates advanced functionality. **(18 - 20 points)**
Code Quality (20%)	Code is poorly written, difficult to read, understand and maintain. Lacks proper indentation and meaningful variable names, unnecessary comments. **(0 - 5 points)**	Code is somewhat readable, minor issues with formatting, comments, or variable naming. **(10 - 14 points)**	Code is well-written, easy to read, and well-documented with only necessary comments and meaningful variable names. **(15 - 17 points)**	Code is exemplary, highly readable, well-structured, and maintainable. Clearly demonstrates best practices in coding style and conventions. **(18 - 20 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)**
Deployment	Code is not uploaded on Netlify/AWS Amplify OR the URL doesn't load the HTML file. **(-100 points)**			Code is properly uploaded on Netlify/AWS Amplify 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)**