1 Project 5

Due: May 9 by 11:59p. **No late submissions**

This document first provides the aims of this project. It then lists the requirements as explicitly as possible. Subsequently it describes the files which have been provided. Finally, it provides brief hints.

1.1 Aims

The aims of this project are as follows:

- To develop a single page web application.
- To use web services.
- To get some exposure to using reactjs.

1.2 Requirements

Set up your gitlab project so that it contains a `prj5/steg-app` subdirectory within the top level `submit` directory. This directory is referred to as the **application directory** in the rest of this project. It should be possible to run your project by cloning your repository and using a browser to access your application directory through a web server.

Your application directory should contain all the resources necessary to run your application. It must contain the following two HTML pages:

`setup.html` This HTML page should contain a form which makes it possible for the user to specify the base URL for the *Project 3* web services. The form should be submitted to `steg-app.html` with the provided web-services base URL contained in the query parameter `ws-url`.

`steg-app.html` This HTML page should contain the single-page web application described below.

The web application should consist of two tabs:

Hide Tab This tab should contain a form which makes it possible for the user to:

1. Select exactly one image from thumbnails for all the images in the `inputs` group retrieved from the web services running on `ws-url`.
2. Specify a message to be hidden by typing in the message or by selecting a file on the user's computer containing the message. The user interface should be designed so that the user can use exactly one of these methods for specifying the message.

3. Submit the form.

When the form is submitted, it should be validated. If the validation succeeds, the application should use the steganography web services running at `ws-url` to hide the provided message in the selected image.

- If the form validation or the hide request fail, a suitable error message should be displayed on the hide tab.
- If both the form validation and hide request succeed, then the application should switch over to the unhide tag with the newly created image preselected.

Unhide Tab This tab should display thumbnails of all images from the `steg` group retrieved from the web services running on `ws-url` and a message area for retrieved messages. It should provide some mechanism for the user to select exactly one of the displayed images. The message area should always display the message recovered from the currently selected message.

If errors are encountered, then a suitable error message should be displayed on the tab.

The application is subject to the following additional constraints:

- All thumbnails should have the same aspect ratio as their original images but should have a maximum width of 100 pixels. Each thumbnail should also be displayed with some kind of distinct name.
- The application should not make any requests to its hosting web server after the initial load of any contained artifacts like stylesheets and scripts.
- You should not have any code executing on the server hosting the application.
- All image retrieval and steganography services must be provided using the remote web services running at `ws-url`. You may assume that the web services are started with some images preloaded into the `inputs` group.
- You application must be built using reactjs.
- Switching between tabs should retain the previous contents / selections.
- In order to avoid a frustrating user experience, your application should not loose any user input. Specifically, no user input should be lost when an error occurs.
- Error messages should displayed as soon as reasonable from a user experience point-of-view. They should be cleared when they are no longer applicable.
- Your project submission must contain a `README` which minimally contains your name, B-number and email address at which you would like to receive

project related email. It may also contain additional comments which may be relevant to the grading of your project.

1.3 Provided Files

You are being provided with the following files:

setup.html Displays a page which allows the user to specify the base URL at which the steganography web services are running. You should not need to modify this file.

steg-app.html Displays a skeleton for your application. You should not need to modify this file.

steg-app.jsx A start at the code for your application. You will be modifying this file.

style.css A start at styling your application. You may modify this file as you desire.

tabs.css A stylesheet which implements the application tabs. You should not need to modify this file.

If the application is run using the above provided files, the application will display with the with the `Hide` and `Unhide` tabs fully functional but the tabs themselves will merely contain placeholders.

1.4 Hints

- The server running the steganography web services must be CORS-enabled. Note that the *project 3 solution* has been trivially modified to be CORS-enabled.
- You will need to serve up your application using a web server. It is trivial to run a web server using python. When in the application directory simply run:

  ```
  $ python -m SimpleHTTPServer
  ```

  This will start a web server listening on port 8000. If you point your browser to that port, you should see a listing of your application directory.
- You will be running your application in a browser. Some points worth noting:
  - If you are hosting your application on your VM, then you will need to run the browser on your VM since you cannot access any ports on your VM from outside your VM. Hence you **must** have GUI access

to your VM. Alternatives for accessing your VM using a GUI include vnc or x2go as outlined in the *Course VM* document.

  - If you host your application on your local workstation, then you can use your workstation browser. You can set up the web services server on your workstation and point your application to it using the setup page.

- You can use your browser's developer tools as a debugger; usually accessed using F12.
- It is possible to limit image widths to 100 pixels using CSS like the following:

```
.container {
  width: 100px;
}

.container img {
  object-fit: contain;
  max-width: 100%;
  max-height: 100%;
  width: auto;
  height: auto;
}
```

  You can and should adjust the CSS selectors to match the way you structure your HTML.

- Note that jsx is a super-set of JavaScript. Hence in the `<x>...</x>` syntax, one cannot use JavaScript keywords. In particular, use `className` for representing a HTML `class` attribute and `htmlFor` for representing a HTML `for` attribute.
- The provided `steg-app.jsx` file already:
  - Contains code for extracting the `ws-url` from the query string.
  - Provides a function which returns the contents of a file stored on the user's disk. This can be used when the user uses a file upload control to specify the message to be hidden.
  - Contains code for setting up the top-level `App` component as well as the setting up of the tabs. The top-level `App` component contains a `select()` function used for selecting a particular tab. Note that the top-level `App` component is passed down as a property to the `Hide` and `Unhide` components and can be used to invoke `App` functions from within those components.
  - Provides a start at the web services wrapper.

– Provides a start for `Hide` and `Unhide` components.

You will need to complete the web services wrapper and the `Hide` and `Unhide` components as indicated by the `TODO` markers.

1.5 Late Submissions

No late submissions are permitted.