1 Project 4

Due: Apr 22 by 11:59p

This document first provides the aims of this project. It then lists the requirements as explicitly as possible. Finally, it provides brief hints.

1.1 Aims

The aims of this project are as follows:

- To develop a web project given a blank slate without any existing code.
- To use web services.
- To generate HTML pages using server-side templating.

1.2 Requirements

Set up your gitlab project so that it contains a `prj4/steg` subdirectory within the top level `submit` directory. It should be possible to install your project by cloning your gitlab project, setting the current directory to its `submit/prj4/steg` subdirectory and then running `npm install`.

After the install, the `submit/prj4/steg` should contain a `index.js` executable file which must start a web server. It must be run using two command-line arguments:

PORT This first command-line argument should specify the port number on which the web server listens for incoming HTTP requests.

WS-URL This second command-line argument should specify the base url (of the form `http://`*domain*`:`*port*) of a web server providing the web services specified for your previous project.

Your web server should allow a browser to browse the pages:

Home Page at URL `/index.html` This should contain links to the Hide and Unhide pages specified below.

Hide Page This should display a form which allows the user to:

- Select exactly one image from all the images in the `inputs` image group from *WS-URL* web service. The images should be displayed as thumbnails having their original aspect ratio but having a width of 100 pixels. Each image should also be displayed with some kind of distinct name.
- Specify a message to be hidden by

- Allowing a user to type in the contents of the message.
- Select a file containing the message.

The message must be specified by the user using exactly one of these options.

When the form is submitted, the web server should validate the submission. If errors are detected, the page should be redisplayed with suitable error messages. If there are no errors, then the specified message should be hidden using the selected image with the resulting image being created in the `steg` group. This should be done using the web service running on *WS-URL*. If the hide fails, then the page should be redisplayed with a suitable error message.

If the hide succeeds, then the browser should display the hide success page specified below.

Hide Success Page This page should display the name of the image created in the `steg` group containing the hidden message. It should also contain links to the hide and unhide pages.

Unhide Page This should display a form which allows the user to:

- Select exactly one image from all the images in the `steg` image group from *WS-URL* web service. The images should be displayed as thumbnails having their original aspect ratio but having a width of 100 pixels. Each image should also be displayed with a distinct ID.

When the form is submitted, the web server should validate the submission. If errors are detected, the page should be redisplayed with suitable error messages. If there are no errors, then the message hidden in the selected image should be recovered using the web service running on *WS-URL*. If the unhide fails, then the page should be redisplayed with a suitable error message.

If the unhide succeeds, then the browser should display a unhide success page containing the text of the recovered message.

Unhide Success Page This page should display the contents of the recovered message after a successful unhide request. It should also contain links to the hide and unhide pages.

For all of the above pages you may assume that *WS-URL* can be accessed directly from the user's browser.

The following choices are entirely up to you:

- The URLs for all pages except the home page.
- The details of the exact text, layout and styling of the pages.

- The choice of form widgets and form submission method (but they must be technically appropriate).
- The exact error messages produced.

Your submission must meet the following additional constraints:

- Client-side JavaScript is not permitted.
- All image retrieval and steganography services must be provided using the remote web services running at *WS-URL*. You may assume that the *WS-URL* web services are started with some images preloaded into the `inputs` group.
- You should use axios as an HTTP client within your web server to access the remote web services.
- You should use mustache as the templating technology in your web server.
- You should not assume that a request to your server is generated only by your form. In particular, it may be generated by a malicious client; hence you **must** validate **all** parameters on the server before taking any action.
- In order to avoid a frustrating user experience, your application should not loose any user input. Hence when redisplaying a page with error message(s) after an error, the redisplayed page must retain any previously provided user input (the sole exception to this would be for fields with sensitive information like passwords).
- 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 Hints

It is possible to create image thumbnails with width 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.

1.4 Late Submissions

If you are submitting the project late, please adhere to the new late submission policy announced a few weeks ago. Specifically, please email me a message before the due date, followed by another one once you have completed the project.