Technical Report

By Wendelle So

Goal of the Project: Build a Dictionary Website with an API from Merriam-Webster using Node.JS and MongoDb

How the project was made:

Main technologies

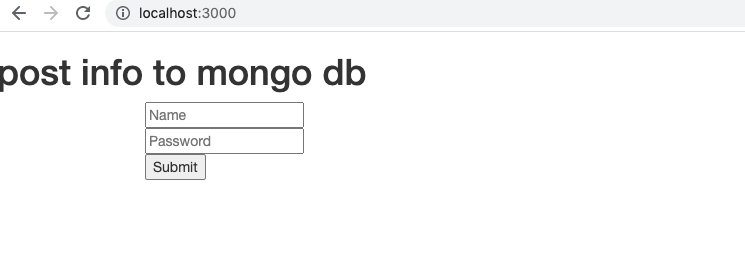
HTML, CSS, and Bootstrap for the HTML pages index.html and proj1.html

Node.JS, Express JS, and MongoDB for the Javascript pages for index.js and server.js

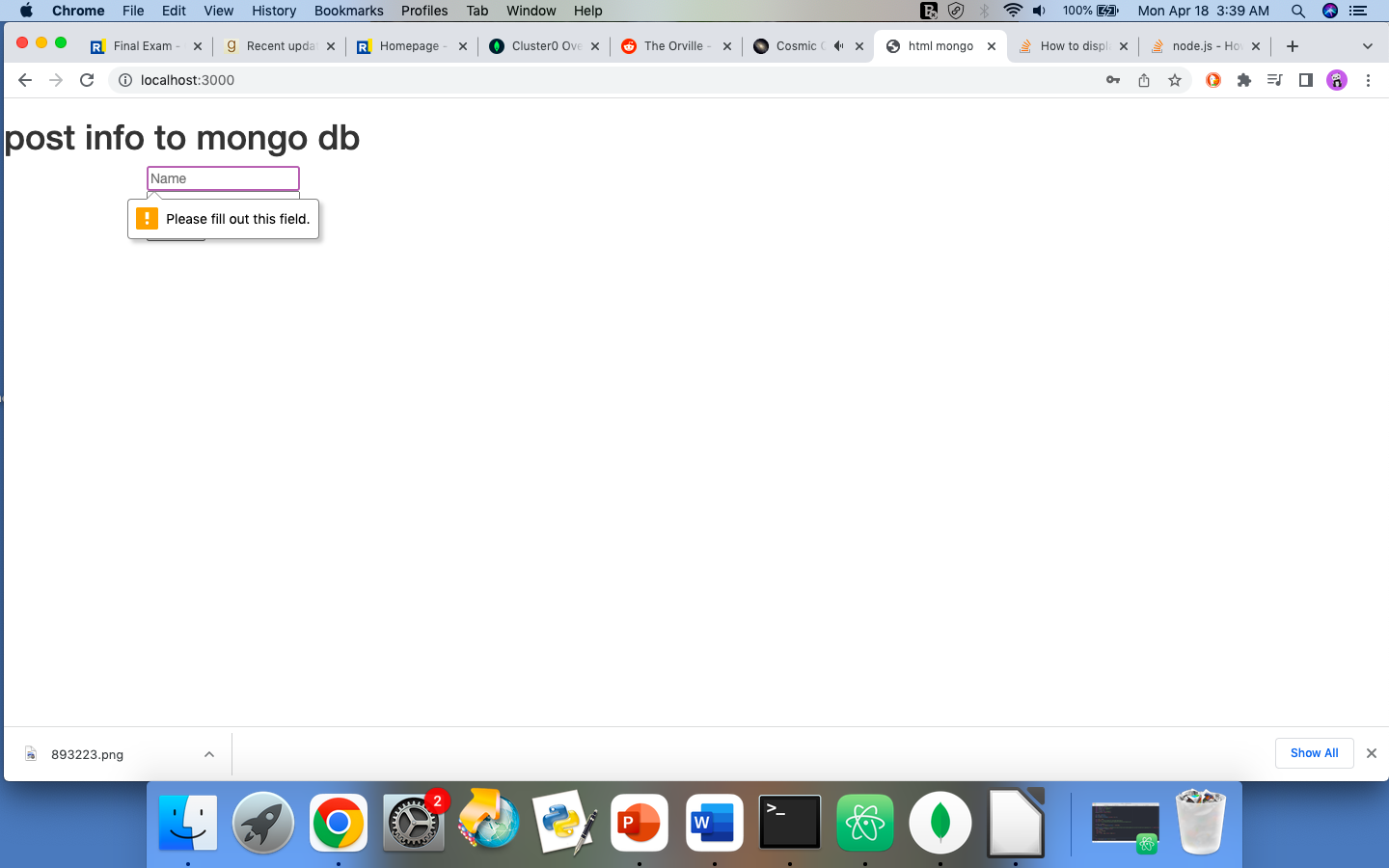
Npm installations Axios, Body-Parser, Mongoose and Validator for support of Node.JS and MongoDB

How the requirements were fulfilled

I. “Users should register a new account to be able to use the service. ”

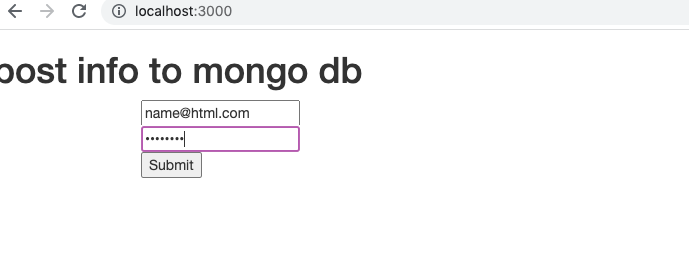


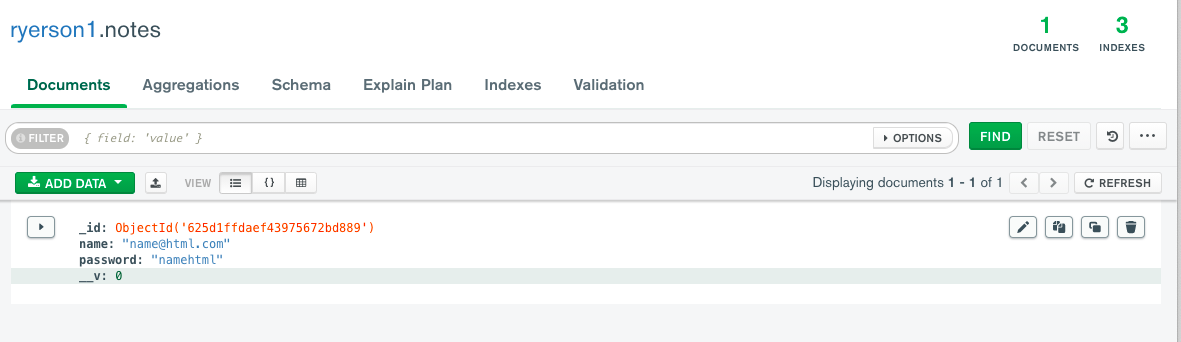
Opening localhost:3000 results in a registration page for Name and password

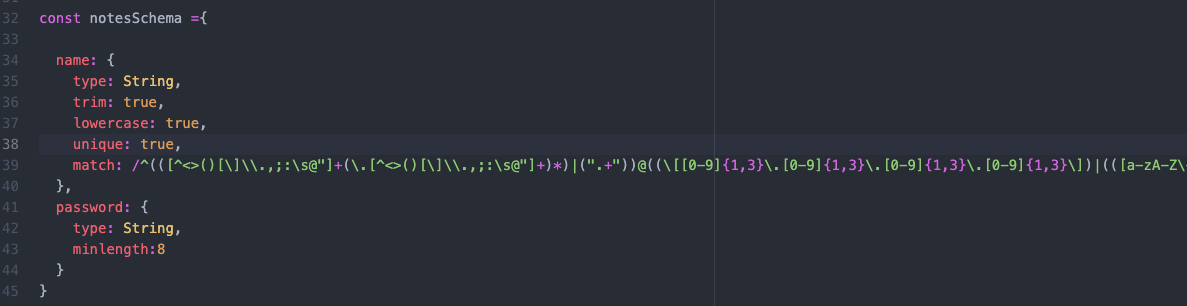


Failing to fulfill this requirement by not entering Name or password will result in staying the same page, with a prompt to fill in the line

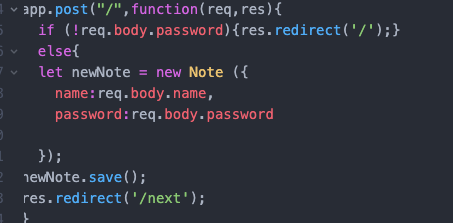
II. “The users and password should be registered to the database (user should be the email address, the password should be at least 8 characters)”



Using mongoose.connect to my assigned MongoDb cluster, and a schema that constrains the format of the entries for email address (to be of form \_@\_.\_) and password (string of minimum 8 characters), a new object based on the entered body.username and body.password is entered into the MongoDb database created for these entries.

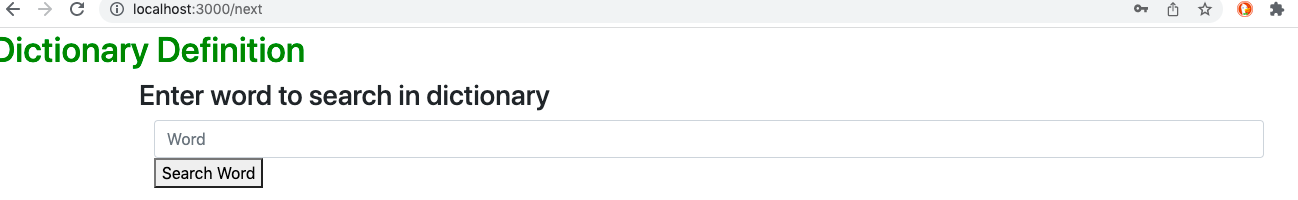


Validation happens through constraints in the Schema, for Email, “unique” is set to true so the registration should be new login name, match is set to a regex expression that corresponds to email format, while for password, the minlength is set to 8.



if they don’t provide login, they will be redirected back to login page (“/”) instead of redirected to next page with dictionary (“/next”)

III. “After login, they can take a look at the English to English definition and its illustration by inserting a word in a text box. This calls the API to receive data from a provider (<https://dictionaryapi.com/products/api-collegiate-dictionary>) and it will show the results (**definition**, **illustration** (show the image in the page if it is provided by API, if not show [this image](https://i.imgur.com/D1nM11A.png))) underneath the text box and store results in the database as well”



After login they are asked to enter a word to look up in the dictionary. This is powered by HTML file, proj1.html, which is sent by app.get and the content is taken in app.post.

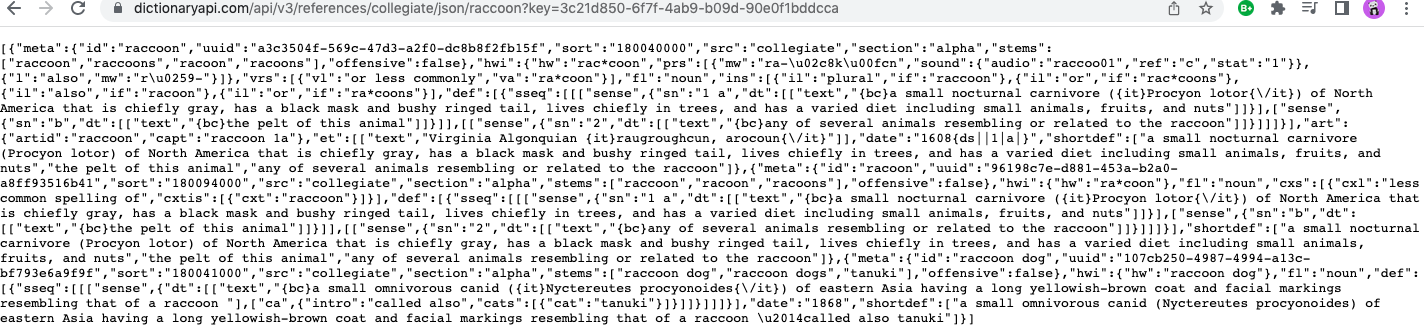
There are 2 possibilities at this stage.

A. If the item has not been called for by search in the previous 10 minutes, a direct API call is made to the external API.

The restriction to these criteria of being a new search of the current word in the past 10 minutes is fulfilled through the .find() ability of mongoDB as follows:

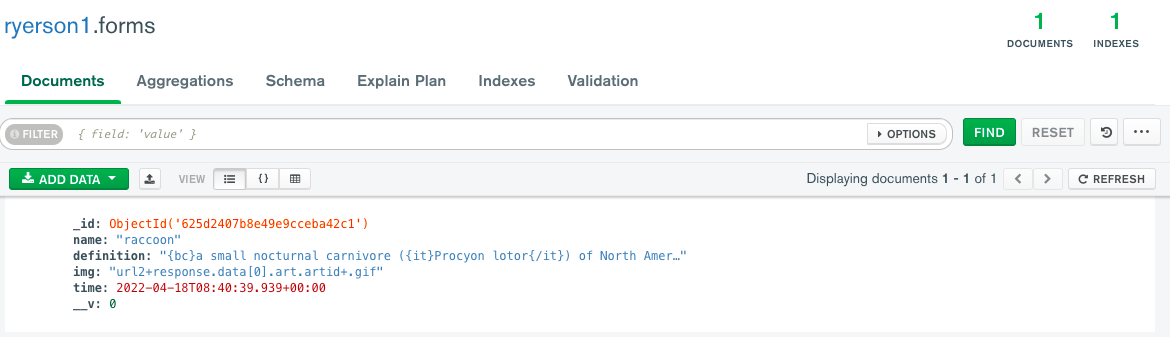


if mongoDB shows this has not been called in the past 10 minutes because the find() has returned search results of empty length, then direct API call is made. This is done by using an API key provided by the Merriam-Webster website. For example, if “raccoon” term is searched, the API will provide all the following JSON,

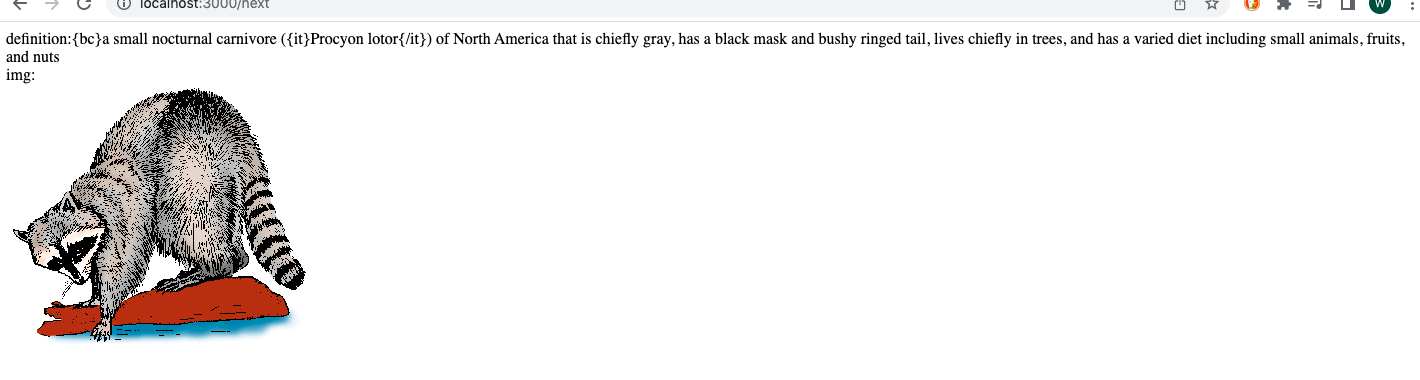


but by limiting the json called, through the help of a JSON tree view, just get the first definition and the url of ARTID if it exists. This fields, along with a timestamp of when the search was called, is entered as an entry into a MongoDB database :

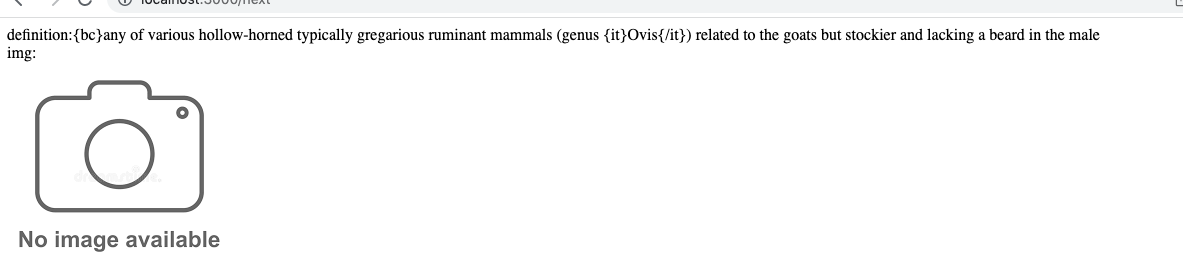




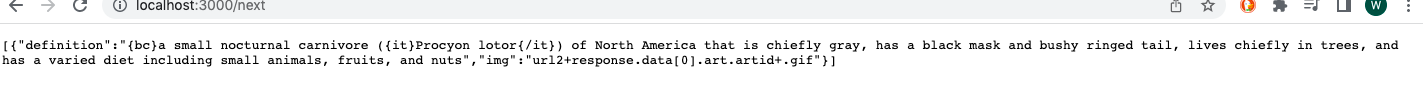
The display shows the definition and the ARTID. The ARTID is transformed into a picture from the url through img src:



However, if the ARTID does not exist, then the image is furnished from the picture dictated in the exam. For example, the ‘sheep’ dictionary entry doesn’t have a picture or ARTID.



B. The second possibility is that the current word has been searched less than 10 minutes ago, then instead of making an API call, we just take from the database, but make sure that the only fields shown in the webpage is the definition and the image.



The above is <10 minutes search for word “raccoon” which returns the database entry.