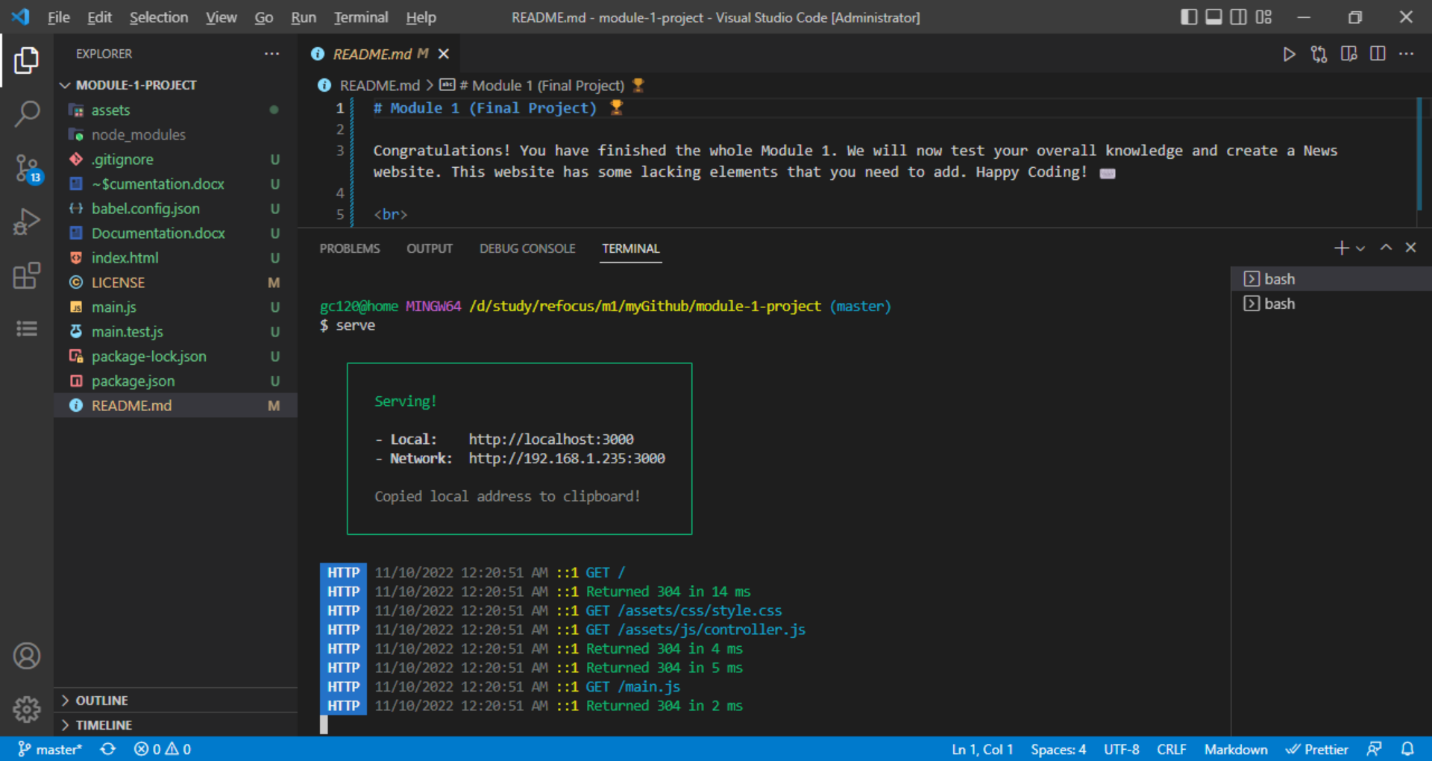
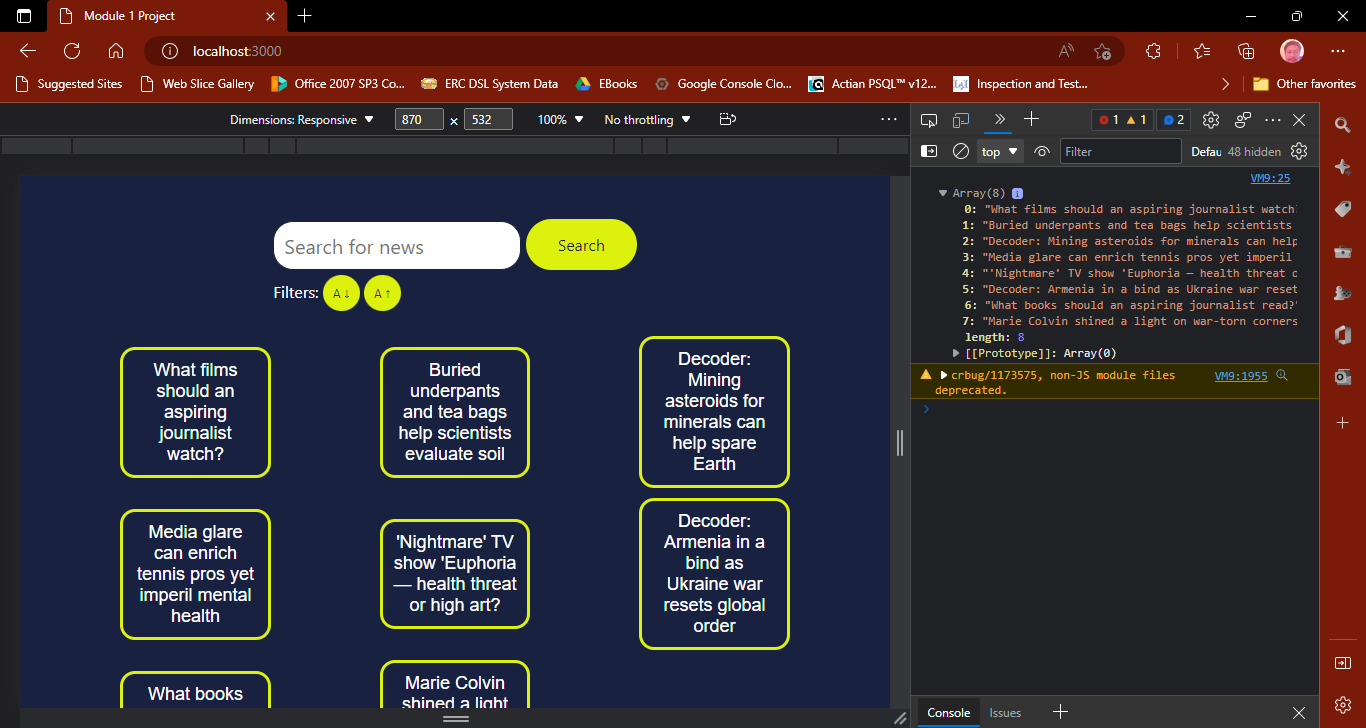
**Documentation for Module 1 Project**



After running “npm install” command in the CLI, launching the project webpage using “**serve”** command.



Sample web page output for the project after invoking “serve” on CLI

main.js

//---To Note---

//1. All pre-built functions must return a new array

//2. You can make new functions as long as you want but do not change or update the pre-built ones

//3. You can udpdate the values inside the \*newsList variable if you want but do not change the variable name

//4. Happy coding :)

// this is to mark that unit testing is in progress

// this can be any value except normal english words

const testMarker = "[\*\*/\*\*@#>><<]";

//newsList is the variable that holds an array of strings

export var newsList = [

  "What films should an aspiring journalist watch?",

  "Buried underpants and tea bags help scientists evaluate soil",

  "Decoder: Mining asteroids for minerals can help spare Earth",

  "Media glare can enrich tennis pros yet imperil mental health",

  "'Nightmare' TV show 'Euphoria — health threat or high art?",

  "Decoder: Armenia in a bind as Ukraine war resets global order",

  "What books should an aspiring journalist read?",

  "Marie Colvin shined a light on war-torn corners of the world",

];

export function searchString(str){

  // the newsList Global Vairiable is filetered

  // in such a way that, cases are the non word characters where ignored.

  // the filtered newsList will always contain whatever the contents in the

  // variable "str".

  // the variable "x" is the place holder of each item when the filter starts to iterate internally

  // the arrow function is a special case of a call back function that will be called by tbe .filter method

  // when it iterates to each individual items in the newsList.

  return newsList.filter((x) => x.toLowerCase().replace(/[\W\_]/g,"").includes( str.toLowerCase().replace(/[\W\_]/g,"") ));

}

//added a parameter variable testStr with a default value of  const testMarker

// for the search function so that the search func is readily testable using JEST

// but it was done carefully/elaborately in such a manner that

// it would not affect the pre existing underlying code that is calling it

export function search(testStr = testMarker) {

  //this part will be executed when unit testing with jest

  if (testStr !== testMarker) return searchString(testStr);

  //inputValue is the variable that contains the search string

  const inputValue = document.getElementById("search-input").value;

  //Write your code here for the search function

  // this part will be executed when called internally

  return searchString(inputValue);

}

export function sort(type) {

  let ret = [];

  if (type == "ascending") {

    //Write your code here for sorting (ascending)

    // ascending sort that disregards cases and non word characteres was performed

    // using toLowerCase and replace with reg exp respectively.

    // a call back function with params (prev,pres) is assigned as a

    // parameter for the sort method that will be called each time it iterates through each

    // item in the newsList.

    // the pres variable holds the value of the item of the present iteration

    // the prev variable holds the value of the item of the previous iteration.

    // such that it compares the two values alphabetically using the "greater than" operator

    ret = newsList.sort((prev,pres) => prev.toLowerCase().replace(/[\W\_]/g,"") > pres.toLowerCase().replace(/[\W\_]/g,"") ? 1 : -1);

  } else {

    //Write your code here for sorting (descending)

    // the one liner code explanation is just like its opposite code above

    // but it does oppositely by using "is less than" operator

    ret = newsList.sort((prev,pres) => prev.toLowerCase().replace(/[\W\_]/g,"") < pres.toLowerCase().replace(/[\W\_]/g,"") ? 1 : -1);

  }

  return ret;

}

main.test.js

import { sort, search, newsList } from "./main.js";

//Example testing for search function

/\*

test("Search function testing", () => {

  expect(search(newsList)).toBe("I am a cool web developer");

});

\*/

test("Search function testing, using 'Daddy Cool PapaG' as input", () => {

  // Start of Arrange Stage

  const a1 = "PapaG";

  const theExpectedResult = [];

  // Start of Act Stage

  const result = search(a1);

  // Start of Assert Stage

  expect(result).toEqual(theExpectedResult); // Assert

});

test("Search function testing, using 'light' as input", () => {

  // Start of Arrange Stage

  const a1 = "light";

  const theExpectedResult = [newsList[7]];

  // Start of Act Stage

  const result = search(a1);

  // Start of Assert Stage

  expect(result).toEqual(theExpectedResult); // Assert

});

test("Search function testing, using 'JourNalist' as input", () => {

  // Start of Arrange Stage

  const a1 = "JourNalist";

  const theExpectedResult = [newsList[0],newsList[6]];

  // Start of Act Stage

  const result = search(a1);

  // Start of Assert Stage

  expect(result).toEqual(theExpectedResult); // Assert

});

test("Search function testing, using empty string as input", () => {

  // Start of Arrange Stage

  const a1 = "";

  const theExpectedResult = newsList;

  // Start of Act Stage

  const result = search(a1);

  // Start of Assert Stage

  expect(result).toEqual(theExpectedResult); // Assert

});

test("Search function testing, using 'decoder' as input", () => {

  // Start of Arrange Stage

  const a1 = "decoder";

  const theExpectedResult = 2;

  // Start of Act Stage

  const result = search(a1).length;

  // Start of Assert Stage

  expect(result).toBe(theExpectedResult); // Assert

});

test("Sort function testing, using 'ascending' as input", () => {

  // Start of Arrange Stage

  const a1 = "ascending";

  const theExpectedResult1 = "Buried underpants and tea bags help scientists evaluate soil";

  const theExpectedResult2 = "What films should an aspiring journalist watch?";

  // Start of Act Stage

  const result = sort(a1);

  // Start of Assert Stage

  expect(result[0]).toEqual(theExpectedResult1); // Assert

  expect(result[result.length-1]).toEqual(theExpectedResult2); // Assert

});

test("Sort function testing, using 'descending' as input", () => {

  // Start of Arrange Stage

  const a1 = "descending";

  const theExpectedResult1 = "What films should an aspiring journalist watch?";

  const theExpectedResult2 = "Buried underpants and tea bags help scientists evaluate soil";

  // Start of Act Stage

  const result = sort(a1);

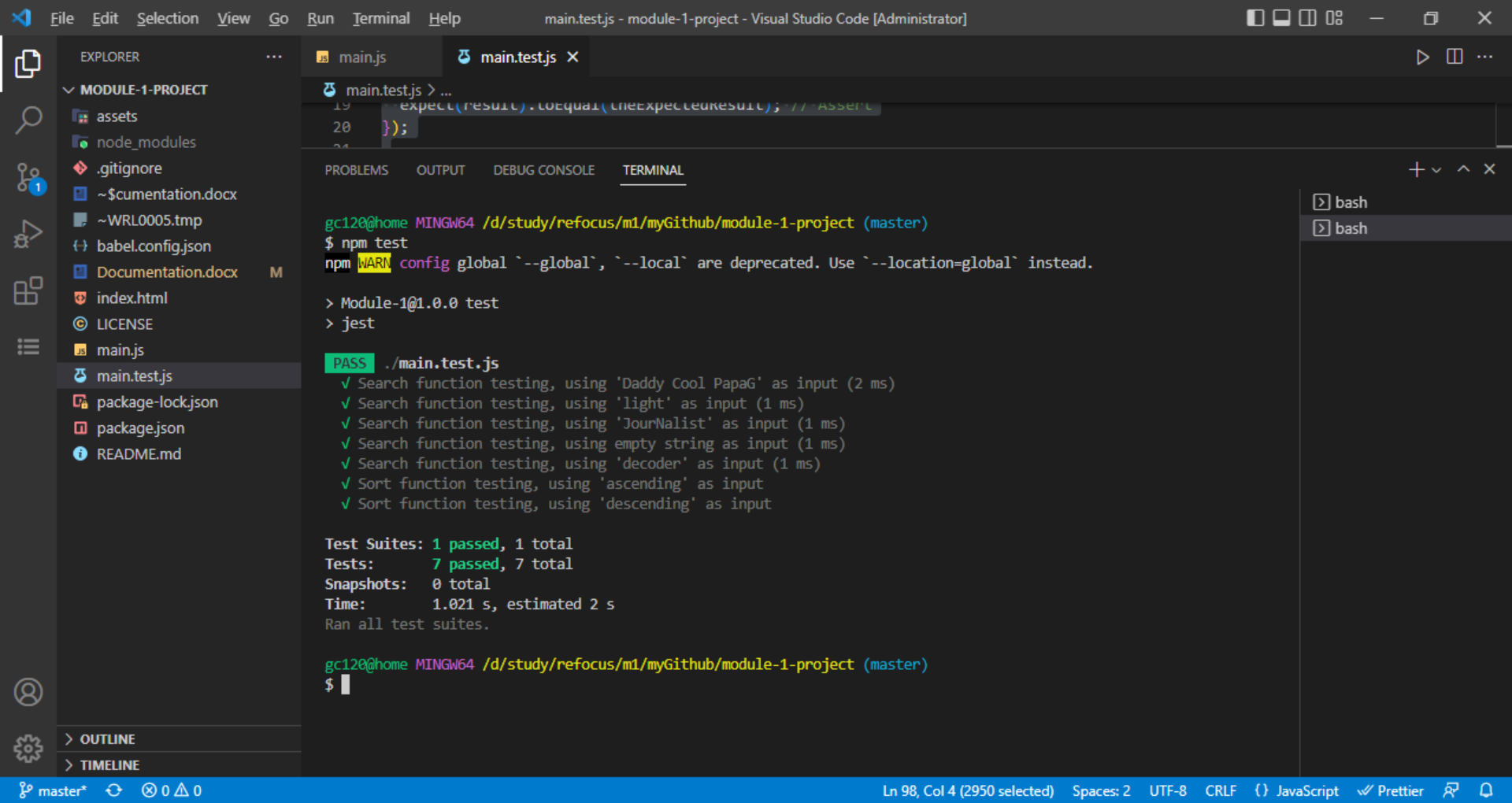
  // Start of Assert Stage

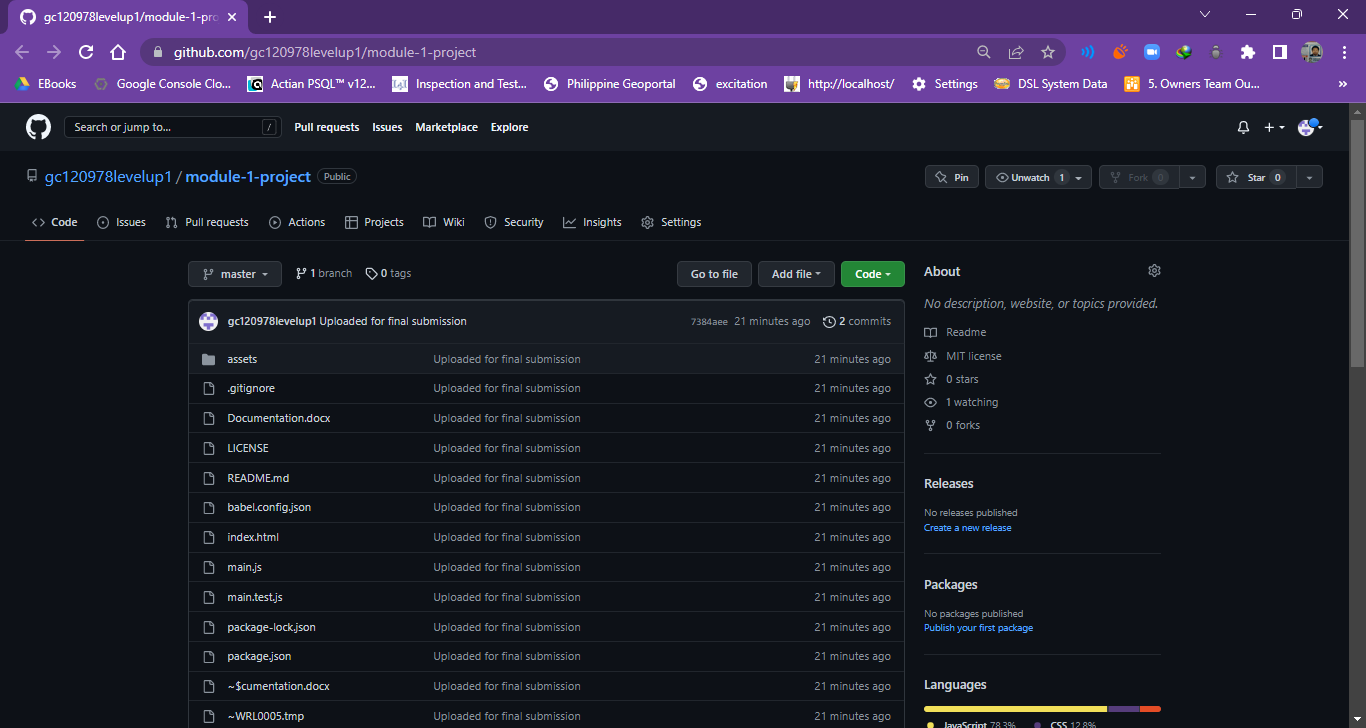
  expect(result[0]).toEqual(theExpectedResult1); // Assert

  expect(result[result.length-1]).toEqual(theExpectedResult2); // Assert

});

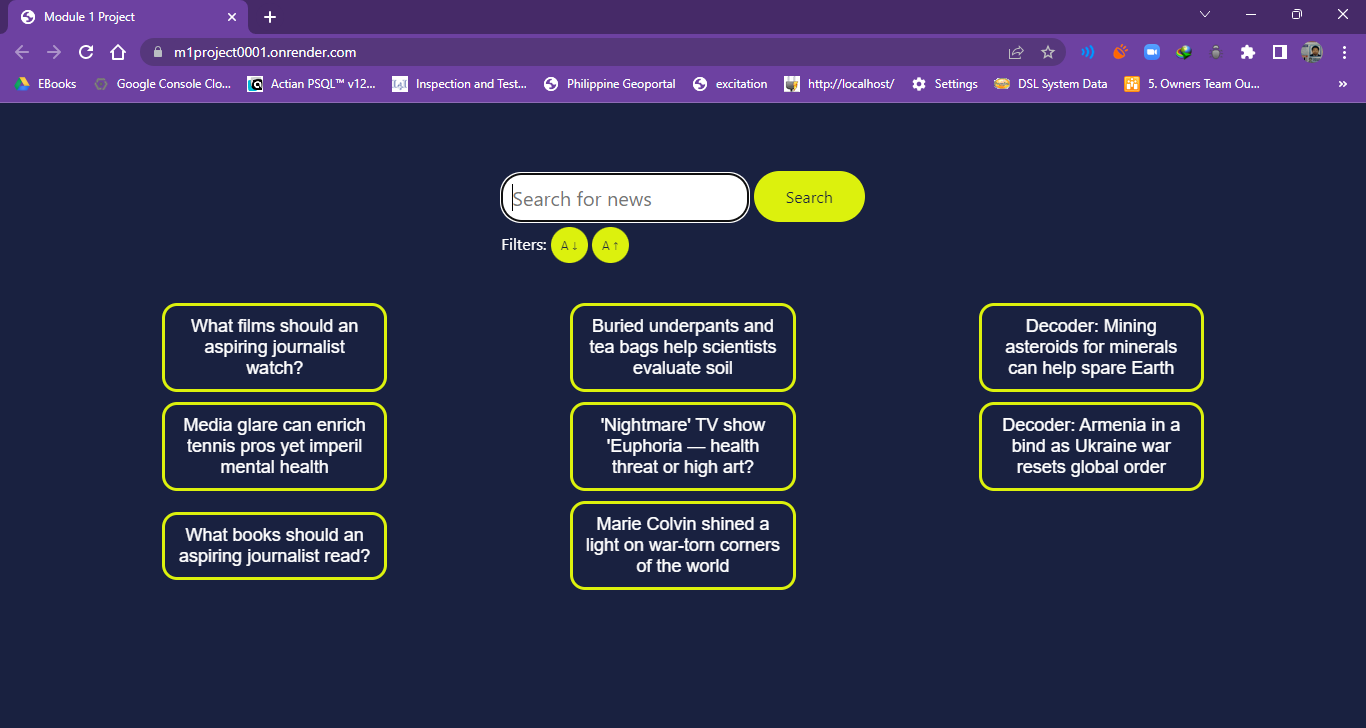
Test Results:





Link to GitHub:

<https://github.com/gc120978levelup1/module-1-project.git>



Link to Render Site where the code is hosted:

<https://m1project0001.onrender.com/>