

IS216 Mini Lab Test

[25 marks]

General Instructions:

- This is a time-bound (1 hour 30 minutes), open-book, open-Internet, and **individual** test.
- You must test your web pages using **Google Chrome Web Browser** Version 103.0.x or later). Your graders will be using only **Google Chrome Web Browser** (Version 103.0.x or later) to test your web pages.
- **No questions will be entertained** by the IS216 teaching team (faculty/instructor/Teaching Assistants) during the test period. If necessary, make your own assumptions and proceed to complete test questions.
- You must **use only standard HTML5, CSS, Bootstrap (Version 5.3), JavaScript and Axios** in your solutions unless the question specifies otherwise. Do not use any other third-party libraries (e.g. Angular, React, or others).
- Use of innerHTML/v-html is prohibited. Any occurrence of innerHTML/v-html found in your code will result in a penalty of 1 mark per occurrence.
- Use meaningful names for HTML class/id and JavaScript variables and functions. You must indent your code (HTML/CSS/JavaScript) properly. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question. For example, if your registered name is "TAN So Tong" and email ID is tan.sotong.2023, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files
<pre><!-- Name: TAN So Tong Email: tan.sotong.2023 --></pre>	<pre>/* Name: TAN So Tong Email: tan.sotong.2023 */</pre>

- You may wish to comment out the parts in your code which cause errors. Commented code will not be marked.

Academic Integrity

- All student submissions will be thoroughly checked by an SMU-approved source code plagiarism checker software program AND an additional external software program. The source code checking will be conducted across all submissions from all sections of IS216.
- Suspected plagiarism cases will be reported immediately to the IS216 faculty in charge and SCIS Dean's Office for further investigation.

- Students in the suspected cases will be informed accordingly by their section faculty, and the incident will be escalated to the SMU University Council of Student Conduct.
- More information about the SMU Student Code of Conduct can be found at this link: <https://smu.sharepoint.com/sites/ucsc/Documents/Code%20of%20Academic%20Integrity.pdf>

Submission Instructions

- Due Date**
 - 4 October 2024 (Friday) 17:15 PM Singapore Time**
 - Late submission policy is as follows:

Submit within 5 minutes of set deadline	10% penalty of your entire test's score
Beyond 5 minutes	0 mark

- Zip** up all files in Q1/Q2/Q3/Q4 folders into <YOUR_SMU_ID>.zip
 - For example, **tan.sotong.2023.zip**
 - Verify by unzipping this zip file – check the content inside**
 - Incorrect submission file name** WILL attract a penalty of up to **20%** of your score for the entire test.
- Only **zip** format is accepted.
 - .7z, rar** or other **compression formats** are **NOT** accepted.
 - Until the correct **zip** format is submitted again by the student, it will be assumed that the student has NOT made the submission and late submission policy will apply.
- Submit the **zip** file to the following location:
 - IS216-MERGED eLearn** page: <https://elearn.smu.edu.sg/d2l/home/398535>
 - Go to **Assignments** → **Mini Lab Test** → **Submit your ZIP file**
 - It is your (student's) individual responsibility to ensure that the zip file submission was successful.**
 - Your section faculty and Teaching Assistants will NOT verify the submission for you.**

Legend



Do NOT edit this given resource file.





Your answer/code goes here into this given resource file.


Q1. Basic CSS

[9 Marks]

Given resources in folder Q1

 q1.html

 images/book_cover_image.png

 q1.css

Instructions

1. You are to style the HTML elements using **vanilla CSS** (no Bootstrap).
2. Open *q1.html* in a browser and examine how it looks.
3. You are to edit only *q1.css* file.

Note: You are to define CSS styles in the labelled locations e.g. /* part A */ within the css file to match the screenshots to the best of your interpretation **where the requirement is not stated explicitly**. This includes margin, padding, font style, font size, font weight, etc. **DO NOT** remove the existing styles or add new selectors in the css file.

Part A: Style the header as shown below. (1.5 marks)



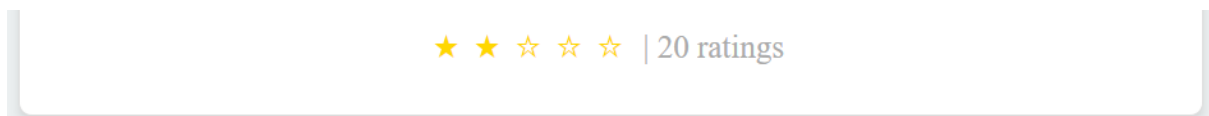
- Set the size of the text as 40 pixels.
- Centralize the text.
- Set the color of the text as white or #fff in hexadecimal.
- Set the background color as #3498db in hexadecimal.

Part B: Style the article section with the class of “book-review” as shown below. (3 marks)



- Set the background color to white or #fff in hexadecimal.
- Insert a solid lined border that is 1 pixel thick with color #ddd in hexadecimal.
- Set the radius of the border to 8 pixels.
- Centralize the contents (including the book image) within the article section.
- Set the spacing between the top border and contents contained within the article section and the spacing between the bottom border and contents contained within the article section to be both 30 pixels.
- Change the width of the book image to 200 pixels.

Part C: Style the rating section as shown below. (3 marks)



- Set the color of the stars as gold.
- Set size of the star as 20 pixels.
- The stars are displayed in a single row (with no bullet points).
- The stars and the text “20 ratings” are displayed in a single row (with the given padding applied).

Part D: Style the footer as shown below. (1.5 marks)

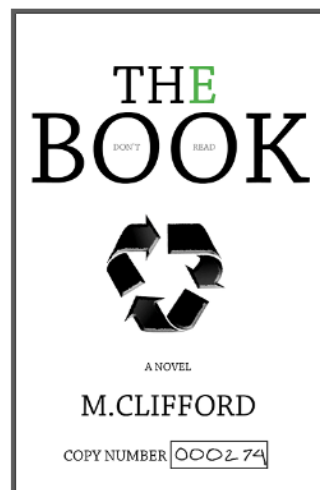
© 2024 Literary Corner. All rights reserved.

- Set the color of the text as white or #fff in hexadecimal.
- Set the background color as #2c3e50 in hexadecimal.
- Centralize the footer text by inserting an adequate CSS style within the “footer p” selector (This is the only accepted way of fulfilling this requirement).
- Do not override the existing width property.

The final webpage should look like the output shown in the next page:

Literary Corner

Featured Review



by M. Clifford

The book that does not deserved to be read


★ ★ ☆ ☆ ☆ | 20 ratings

© 2024 Literary Corner. All rights reserved.

Q2. Bootstrap: Debug Layout

[5 Marks]

Given resources in folder Q2

 q2.html

Instructions

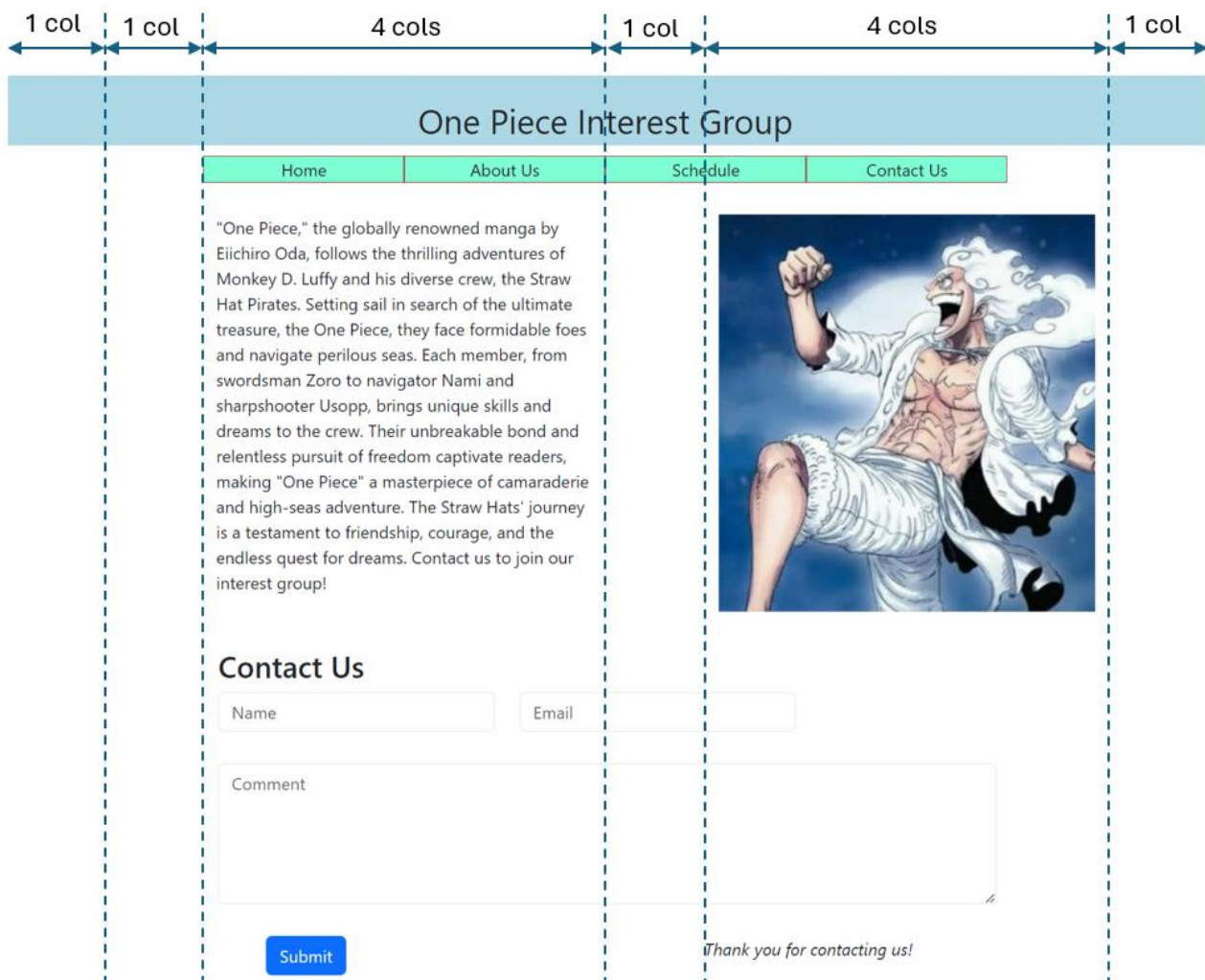
1. You are to solve the web page layout problem using Bootstrap's *grid and breakpoints*.
2. Open q2.html in a browser and examine how it looks when you change the browser width.
3. You should notice that the given code has already set grid layout and breakpoints for different browser widths, but the code is buggy.

Note:

- You are only required to debug and add code to achieve the web page layout at different browser widths specified below.
- You are not required to add/edit other styles, such as margin, padding, font style, font size, etc., not mentioned in the requirements.

Debug and add code in q2.html so that it behaves as follows:

Part A: When the browser width is ≥ 992 px (lg breakpoint), the form should appear as shown below. (1 mark)



- The div with `id="desc"` (the paragraph of text) and the div with `id="image"` (the image) should occupy **4 columns** of Bootstrap 12-column grid, each.

Part B: When the browser width is ≥ 768 px (md breakpoint) but < 992 px (lg breakpoint), the “Contact Us” portion of the form should appear as shown below. (2 marks)

The diagram illustrates a Bootstrap 12-column grid layout for a 'Contact Us' form in the medium (md) breakpoint. The grid is defined by vertical dashed lines and horizontal arrows indicating column counts. The layout is as follows:

- Header:** A horizontal arrow at the top indicates a total width of 12 columns, divided into four segments: 1 col, 5 cols, 5 cols, and 1 col.
- Form Title:** The text "Contact Us" is centered in the first 5-column segment.
- Form Fields:**
 - The "Name" input field occupies the first 5-column segment.
 - The "Email" input field occupies the second 5-column segment.
 - The "Comment" text area occupies the first 5-column segment.
 - A second, larger text area occupies the second 5-column segment.
- Footer:**
 - The text "Thank you for contacting us!" is centered across the 12-column width.
 - The "Submit" button is centered below the thank you message.

- Name and Email fields occupy **5 columns** of Bootstrap 12-column grid, each.
- The “Thank you for contacting us!” text **comes before** the Submit button. This requirement applies even when the browser width is < 768 px.

Part C: When the browser width is < 768 px (md breakpoint), the form should appear as (partially) shown below. (2 marks)

1 col 5 cols 5 cols 1 col

One Piece Interest Group

Home About Us

Schedule Contact Us

"One Piece," the globally renowned manga by Eiichiro Oda, follows the thrilling adventures of Monkey D. Luffy and his

⋮

Contact Us

Name

Email

Comment

Thank you for contacting us!

Submit




1 col 10 cols 1 col

- The menu items of "Home" and "About Us" are side-by-side and occupy **5 columns** of Bootstrap 12-column grid, each.
- The menu items of "Schedule" and "Contact Us" are side-by-side and occupy **5 columns** of Bootstrap 12-column grid, each.
- The Name field occupies **10 columns** of Bootstrap 12-column grid.
- The Email field occupies **10 columns** of Bootstrap 12-column grid.
- The "Thank you for contacting us!" text still **comes before** the Submit button.

Q3. JavaScript DOM

[6 Marks]

Given resources in folder Q3

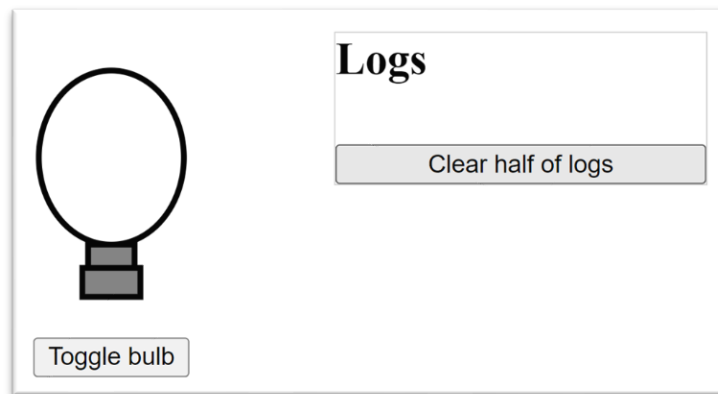
-  q3.html
-  q3.css
-  q3.js

Instructions

1. You are to edit ONLY q3.js.
2. In q3.html, the “Toggle bulb” button is already written such that it will call the `changeColor()` function in q3.js. Similarly, the “Clear half of logs” button is already written such that it will call the `halveLogs()` function in q3.js. You need not worry about calling these two functions.
3. This question will be marked by an automated testing script.
4. Debug and Add Code in q3.js so that q3.html behaves as follows.

Original output

When q3.html is open in the browser for the first time, it should show the following output:



Tasks to complete

Part A: Change color of bulb (1 mark)

Add code in the `else` block of the function `changeColor()` so that when the user clicks on the “Toggle bulb” button, the style of the `fill` property of the bulb toggles between the color white and yellow i.e. if the bulb’s fill is white, it should change to yellow when the button is clicked, and if the bulb’s fill is yellow, it should change to white when the button is clicked.

Note: If you are successful, when you click on the “Toggle bulb” button, the bulb changes color with a transition time of 1 second. This transition time of 1 second is dictated by q3.css, which you do NOT need to change.

Part B: Delay use of “Toggle bulb” button (1 mark)

Add code in the `delayButton()` function so that when the user clicks on the “Toggle bulb” button, the “Toggle bulb” button will be disabled for 1 second, before it is enabled for use again.

For addLog () function

The `addLog ()` function is supposed to add logs of user interaction to the `div` of `id="logs"`.

- If the user pressed the “Toggle bulb” button to turn the fill of the bulb **yellow**, the three logs to add should be:

"User interacts.", "ON.", "Bulb lights up."

- If the user pressed the “Toggle bulb” button to turn the fill of the bulb **white**, the three logs to add should be:

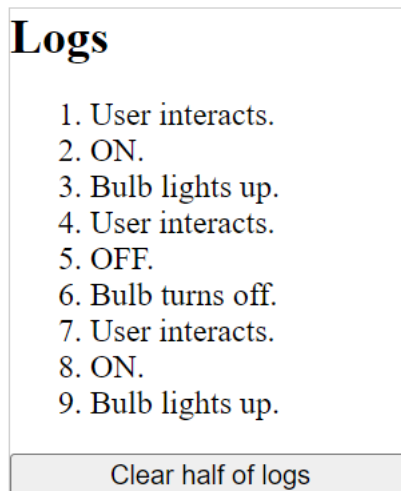
"User interacts.", "OFF.", "Bulb turns off."

Part C: Add code to addLog() function (1 mark)

Add code to `addLog ()` so that for each element of the `newLog` array passed in, it:

- creates an `` element,
- adds the string content of the element to the `` element,
- appends the `` to the page using `logBox`, which points to the `` with `id="logs"`, and
- updates the `logNumber` variable.

Example: After a page refresh, when the “Toggle bulb” button is clicked 3 times, the following logs should be shown:

**Debugging Task****Part D: Show errorMsg when there are too many logs (0.5 marks)**

In the `changeColor ()` function, debug the erroneous code to add an error message to the `div` with `id="errorMsg"` when the number of logs in the `div id="logs"` will potentially exceed the maximum number allowed under `maxLogs=10` if the user interacts with the “Toggle bulb” button. Do not change the given style associated with `id="errorMsg"`.

Example: After a page refresh, when the “Toggle bulb” button is clicked a 4th time, the following output should be shown:

Logs

1. User interacts.
2. ON.
3. Bulb lights up.
4. User interacts.
5. OFF.
6. Bulb turns off.
7. User interacts.
8. ON.
9. Bulb lights up.

Clear half of logs

Clear some logs before proceeding

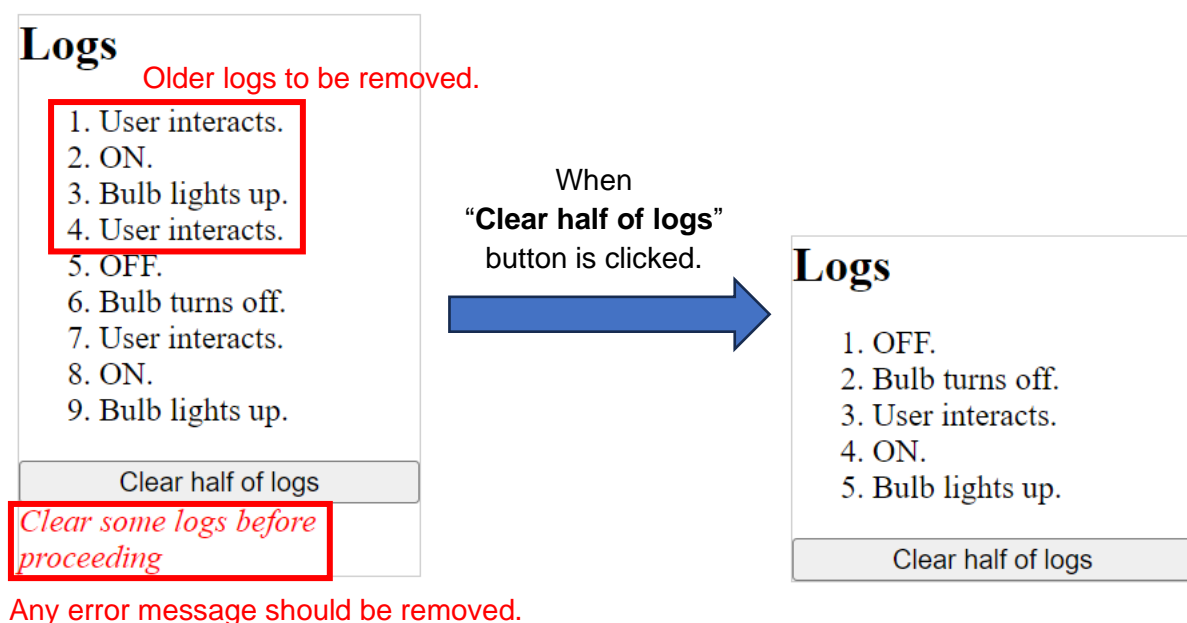
For halveLogs () function

The `halveLogs ()` function is called when the user clicks on the “Clear half of logs” button.

Part E: Add code to halveLogs() function (2 marks)

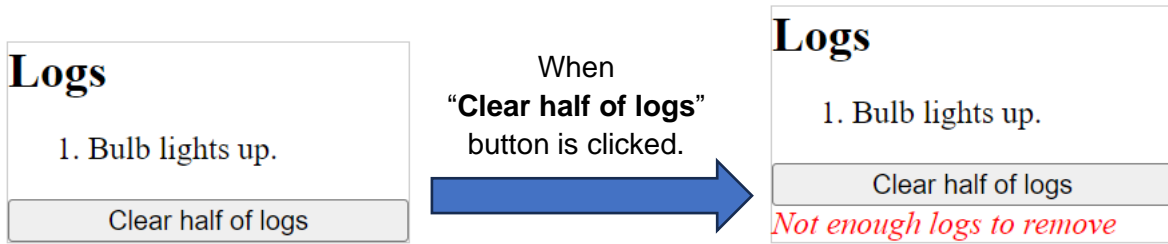
Requirement 1: If there is an odd number of existing logs, remove a whole number of oldest logs just less than half of the number of logs. If there was an error message, remove it. (1 mark)

Example: Original number of logs is 9. After the “Clear half of logs” button is pressed, 4 of the older logs (4 being the whole number just less than 4.5, half of 9) should be removed. Error message removed.



Requirement 2: If there is only 1 log left, do **NOT** remove the last remaining log. Instead, show an error message in the div with `id="errorMsg"` with the following text: "Not enough logs to remove". Do not change the given style associated with `id="errorMsg"`. (1 mark)

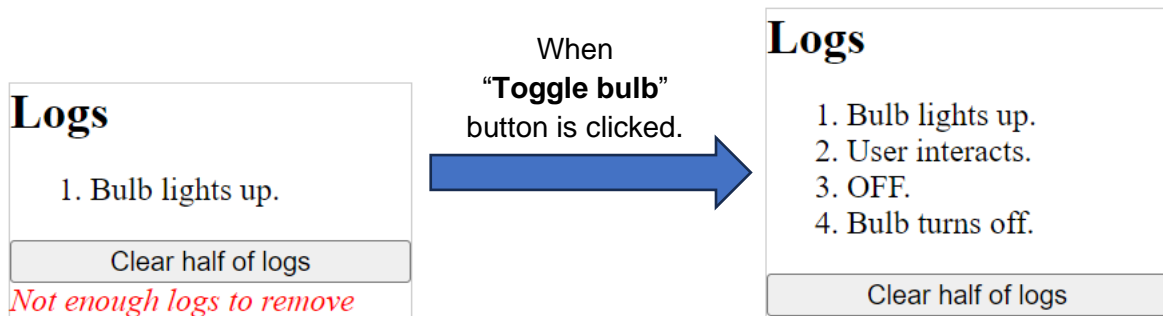
Example:



Part F: Remove any error message if "Toggle bulb" works i.e. logs can be added. (0.5 marks)

In any part of `q3.js`, add code to remove any error message if "Toggle bulb" works i.e. logs can be added again.






Example: The user tried to clear the logs, but as there was only 1 log left, an error message is shown. The user then clicks on the "Toggle bulb" button, which adds the necessary logs, and clears the error message.



Q4: JavaScript AXIOS & JSON

[5 Marks]

Given resources in folder Q4

-  q4.js
-  q4.html
-  q4.js
-  api.php
-  bike_stations.json

Instructions

1. You will need to copy the given files into your designated WAMP/MAMP folder.
2. You are to only insert codes into q4.js to fulfil the following requirements.

Part A: (2 marks)

1. Create the function “getStations”. Inside the function, an Axios GET request to api.php (no parameter required) is used for retrieving the list of bike stations and their availability information.
2. Insert codes into the function “displayStations” to show the list of bike stations and the availability of bicycles and docks in each station.
3. The static HTML for formatting the DOM is given as comments in q4.html. The list of bike stations with the bicycles and docks’ availability should be dynamically displayed using JavaScript though.

```
<!--  
    <div class="station-item">  
        <h3>Central Park Station</h3>  
        <p>Available Bikes: 10 </p>  
        <p>Available Docks: 3 </p>  
    </div>  
-->
```

4. The desired output is shown below.

City Bike Share

Central Park Station

Available Bikes: 11

Available Docks: 4

Downtown Plaza Station

Available Bikes: 5

Available Docks: 10

Riverside Park Station

Available Bikes: 8

Available Docks: 7

University Campus Station

Available Bikes: 3

Available Docks: 12

Shopping District Station

Available Bikes: 2

Available Docks: 2

Rent a Bike

Return a Bike

Select a station



Rent Bike

Select a station



Return Bike

Part B: (1 mark)

1. Complete the function “populateStationDropdowns” to populate the list of station locations in the “Select a station” dropdown lists of both “Rent a Bike” and “Return a Bike”.
2. The list of station locations should be retrieved from the Axios GET request to api.php (in Part A). The static HTML for formatting the DOM of the “Select a station” dropdown list is given as comments in q4.html. The list of bike stations in “Select a station” dropdown list should be dynamically formulated using JavaScript though. Do note that the option value should be set to the designated station id.

```
<!--
<option value="1">Central Park Station</option>
-->
```

3. The desired output is shown below.

Rent a Bike

Select a station ▼

Select a station

Central Park Station

Downtown Plaza Station

Riverside Park Station

University Campus Station

Shopping District Station

Rent Bike

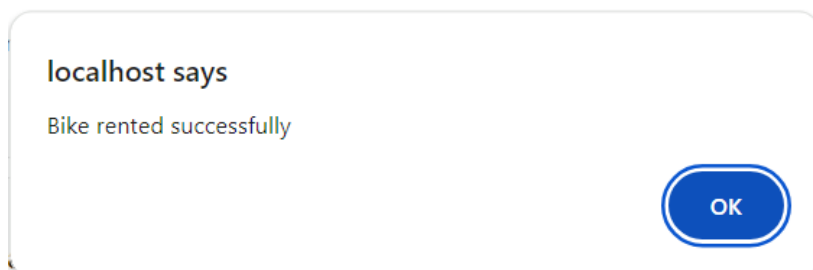
Return a Bike

Select a station ▼

Return Bike

Part C: (1.5 marks)

1. Complete the function “postAction” to perform either a bike rental or bike return by doing an Axios POST call (with parameters “action” and “station_id”) to api.php.
2. The function parameters “action” and “station_id” have to be passed as query parameters in the Axios POST call.
3. For successful bike rental or return, an alert box shown below should appear with the message as retrieved from the response object of the Axios POST call. Note: The message should **NOT** be **HARDCODED**.



4. Upon successful bike rental at a selected station e.g. Riverside Park Station, the “Available bikes” should decrease by 1 and the “Available docks” should increase by 1. Upon successful bike return, the “Available bikes” should increase by 1 and the “Available docks” should decrease by 1. A sample output for successful bike rental is shown below.

Note: The update of “Available bikes” and “Available docks” numbers should occur without reloading the page.

Before bike rental	After bike rental
<div> Riverside Park Station </div> <div> Available Bikes: 8 </div> <div> Available Docks: 7 </div>	<div> Riverside Park Station </div> <div> Available Bikes: 7 </div> <div> Available Docks: 8 </div>

Part D: (0.5 marks)

- When there are no available bikes at a selected station and the user tries to rent a bike at that station, an alert box with the message “Error renting bike” should appear. Hint: You will have to handle the HTTP 400 erroneous response code returned by the Axios POST call to api.php.

No available bikes at station	Error
<div> Shopping District Station </div> <div> Available Bikes: 0 </div> <div> Available Docks: 4 </div>	<div> localhost says Error renting bike </div> <div>OK</div>

- When there are no available docks at a selected station and the user tries to return a bike at that station, an alert box with the message “Error returning bike” should appear. Hint: You will have to handle the HTTP 400 erroneous response code returned by the Axios POST call to api.php.

No available docks at station	Error
<div> Shopping District Station </div> <div> Available Bikes: 4 </div> <div> Available Docks: 0 </div>	<div> localhost says Error returning bike </div> <div>OK</div>

Samples of api.php GET and POST calls using Postman:

The screenshot shows a Postman interface for a GET request. The URL is `localhost/is216/MLT_24T1/solutions/q4/api.php`. The request is configured with the following settings:

- Method: GET
- Body: JSON (selected)
- Headers: 6
- Pre-request Script: Empty
- Tests: Empty
- Settings: Default

The response is displayed in the "Body" tab, showing a JSON array of 5 objects. The response is formatted as "Pretty" JSON.

```
1 [
2   {
3     "id": 1,
4     "name": "Central Park Station",
5     "available_bikes": 11,
6     "available_docks": 4
7   },
8   {
9     "id": 2,
10    "name": "Downtown Plaza Station",
11    "available_bikes": 5,
12    "available_docks": 10
13  },
14  {
15    "id": 3,
16    "name": "Riverside Park Station",
17    "available_bikes": 8,
18    "available_docks": 7
19  },
20  {
21    "id": 4,
22    "name": "University Campus Station",
23    "available_bikes": 3,
24    "available_docks": 12
25  },
26  {
27    "id": 5,
28    "name": "Shopping District Station",
29    "available_bikes": 4,
30    "available_docks": 0
31  }
32 ]
```

POST

localhost/is216/MLT_24T1/solutions/q4/api.php

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

```
{ "action": "rent", "station_id": 1 }
```

Body

Cookies

Headers (10)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

```
{  
  "message": "Bike rented successfully"  
}
```

End of Paper