

**Web Systems and Technologies (EIE4432)**  
**Lab 3: Drink Menu & Reservation System**

***Expected Outcomes***

- Develop an enhanced web form with jQuery to improve UI/UX, including dynamic input fields and real-time form validation
- Utilize jQuery AJAX to fetch data from a local JSON file, dynamically display the results, and implement sorting or filtering options
- Create an interactive SVG seat map using jQuery, allowing users to book tables and seats with changes reflected in the SVG graphics
- Integrate web storage techniques (e.g., localStorage) to persist user information or selected seat data across sessions
- Demonstrate proficiency in handling AJAX techniques for fetching data and making reservation requests, providing a seamless user experience

***Submissions***

- Please refer to the teaching schedule for the deadline
- This is an individual work
- Students should host their work on a designated hosting service and provide a public domain for viewing the website live
- Students need to submit a report with screen captures showing they have achieved the outcomes stated above
- Every screen capture should be full screen and show the **Student ID, System Date & Time**, otherwise **1 grade would be deducted** from the laboratory
- Late submission would deduct 20-50% of the mark of this laboratory
- Please declare the use of GenAI tools and how they have been used in the appendix page similar to the following:
  - I declare that Generative AI tools have been used to prepare the submitted work. The Generative AI tools used and the manner in which they were used are as follows:  
\_\_\_\_\_.
  - You must reference them in according with the accepted academic conventions (e.g. IEEE or APA styles).

***Assessment Criteria***

The report will be assessed according to

- The clarity of presentation,
- Technical correctness, and
- Whether the screen captures can demonstrate that the outcomes have been achieved.

***Lab Activities***

- Task 1: Enhancing UI/UX Experience in Ordering Form (60 mins)
- Task 2: Fetch Data for Drink Menu (60 mins)
- Task 3: Advanced SVG Seat Map with LocalStorage Integration (60 mins)
- Task 4: Publish your Website (Optional)

## Task 1: Enhancing UI/UX Experience in Ordering Form (60 mins)

### Expected Outcome

#### Drink Ordering Page (order.html)

- Apply form validation and visual feedback to the "Name" field
- Apply form validation and visual feedback to the "Drink" field
- Implement a dynamic preview of the selected drink
- Implement a confirmation message with a fade-out effect after placing the order

### Suggested File Structure

```

└─ <Student ID>_EIE4432_LAB3 (Project Root Folder)
    ├── assets
    │   ├── img1.jpg
    │   ├── img2.jpg
    │   └── etc...
    ├── scripts
    │   ├── jquery-3.7.1.min.js
    │   ├── order.js
    │   ├── reserve-table.js
    │   └── menu.js
    ├── index.html
    ├── menu.html
    ├── order.html
    └── reserve.html
  
```

1. Create a folder on designated location, such as desktop, name it as "<Student ID>\_EIE4432\_LAB3"
2. Reuse the "index.html", "order.html", "reserve.html" & "order.js" developed in Lab 2, save them into the Lab 3 folder created according to suggested file structure
3. **You are strongly recommended to create a new **private** repository on GitHub, and keep the project in GitHub, the deployment to cloud service would need to connect with a GitHub repository at the end of the lab exercise**
4. In every pages, here are some modification:
  - Modify the navigation bar by adding 1 more item "Menu", it must link to "menu.html"
  - Edit the **footer** from **EIE4432 Lab 2 Project** to **EIE4432 Lab 3 Project**
5. To test your website throughout the lab exercise, **you are highly recommended to start a local server for testing, DO NOT double click the html file for testing.** For more detailed steps, please refer to Lab 1.
6. Go to jQuery official website (<https://jquery.com/>) to download the jQuery file
  - For the detailed download method, please refer to Lecture 2 handout
7. In "order.html", add a script tag at the end of <head> section, with the src attribute of the jQuery file
  - e.g. <script src="scripts/jquery-3.7.1.min.js"></script>
8. In Lab 2, the external JavaScript file "order.js" is included in the <head> section. Make sure the jQuery file is included before "order.js"

9. The validation of “Name” field is checking on the emptiness of the value
- If the value is empty, the border of text box turns red when user clicks outside
  - Otherwise, the border turns green to indicate the input value is valid

Name:

Name:

10. Create 2 classes' style in <head> section

- Class 1: error
  - With red border color (required), with red box shadow (optional)
- Class 2: error-free
  - With green border color (required), with green box shadow (optional)

### Lab #1.1

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

- Source code of order.html (<style> part must be included)

11. Write a `$(document).ready()` function for wrapping code inside, it is to ensure that the code execute once the DOM has finished loading
12. Select the name field text box by any method with jQuery
- If it is associated with id, `$("#<id name>")`
  - If it is associated with class, `$("<class name>")`
  - If it is only associated with a tag, `$("<tag name>")`
13. Attach an event listener to the “input” event using `.on()` method
- This event listener triggers whenever user inputs something into the input field
  - Reference: [https://www.w3schools.com/jquery/event\\_on.asp](https://www.w3schools.com/jquery/event_on.asp)
14. Use `$(this).val()` to retrieve the value of the selected input field
- Here, `$(this)` should refer to the “name” input field
15. Trim the value and check whether it is empty or not
16. When it is empty, to enable the visual effect of red border color, add a class to selected element using `.addClass()` method
17. Remove the class of the opposite effect using `.removeClass()` method
18. By using chain function, reduce the code length, add & remove the class at the same time
- Chain Function Example:  
[https://www.w3schools.com/jquery/jquery\\_chaining.asp](https://www.w3schools.com/jquery/jquery_chaining.asp)

19. Do the same effect for drink selection by checking whether user selects the default value

- If default value is selected, change border to red with **(optional)** shadow effect
- Otherwise, change border to green with **(optional)** shadow effect
- Hint: the event listener should trigger by “change” event for selection menu

Drink:

Please Select
▼

Drink:

Iced Bubble Milktea
▼

### Lab #1.2

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

- Source code of “order.js”
- Screen capture of selecting default drink value
- Screen capture of selecting any valid drinks

20. When user selects any drink, show a preview image for them to preview what is the drink looks like, for enhancing the UX experience

21. Download appropriate images and save into “assets” folder

22. Add a <div> with class names: **row mb-3 d-none justify-content-center** beneath the selection menu in “order.html”

23. Add <img> tag in the newly created <div>

- For “src” & “alt” attribute, leave them empty for dynamic changes
- For better UI, set the width of image to 25% by using class “w-25”
- Give a unique id to the <img>

Drink:

Iced Bubble Milktea
▼



Drink:

Iced Latte



Drink:

Iced Pineapple Juice



24. Back to “order.js”, when drink select menu is changed,
  - “d-none” in <div> controls the visibility of the element
  - Remove the class “d-none” for showing the images inside this <div>
  - Show the correct image by `.attr()` method to update both “src” & “alt”
  - Reference: [https://www.w3schools.com/jquery/html\\_attr.asp](https://www.w3schools.com/jquery/html_attr.asp)
25. Also, if default value is selected, there is no image to display, the <div> should not be displayed, otherwise show the <div> and allows image for preview
  - Add the class “d-none” back to the <div> for hiding the images
26. Back to `placeOrder(event)` function developed, remove the alert for showing successful order message
27. In “order.html”, add 1 more row beneath the price information with class: `row mb-3 d-none message`
28. In `placeOrder(event)` function, when an order is successfully placed, display a confirmation message with fade-out effect

**Price: \$26**

Order placed successfully! Thank you for your order.

**Price: \$0**

Place Order

Reset

29. Select the message <div>, add the message content into the row using `.html()` method, so the message content is wrapped by another <div> element
30. Next, select the <div> element created inside the <div> with class message using “>” combinator

31. Add the classes from Bootstrap alert to the message content <div>
  - Reference: <https://getbootstrap.com/docs/5.3/components/alerts/>
32. Give fade-in effect to the component for 0.5 seconds
33. Then make it stay for 3 seconds using `.delay()` method
  - Reference: [https://www.w3schools.com/jquery/eff\\_delay.asp](https://www.w3schools.com/jquery/eff_delay.asp)
34. Give fade-out effect to the component for 0.5 seconds
35. When the fade-out effect is finished:
  - The alert element itself will be removed using `.remove()` method
  - Reset the form to its initial state
  - Update the price display to \$0 after form resetting

**Lab #1.3**

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

- Source code of “order.js”
- Screen capture of placing order successfully with displaying message under price information

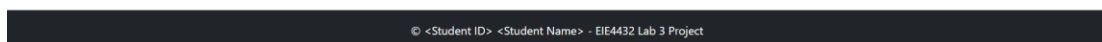
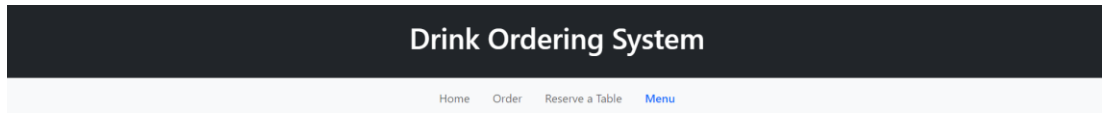
## Task 2: Fetch Data for Drink Menu (60 mins)

### Expected Outcome

- Utilize jQuery's AJAX GET method to asynchronously fetch data from a local JSON file
- Handle potential errors during the AJAX request and display appropriate error messages
- Utilize jQuery to manipulate the DOM and update the UI based on the fetched data

### Instructions

1. Download “drink-menu.json” (with 2 drink information only) from Blackboard, save it into “assets” folder
2. Create a new file “menu.html”, re-use the “index.html” and clear content in container
3. Make sure “Menu” is actively selected on navigation bar



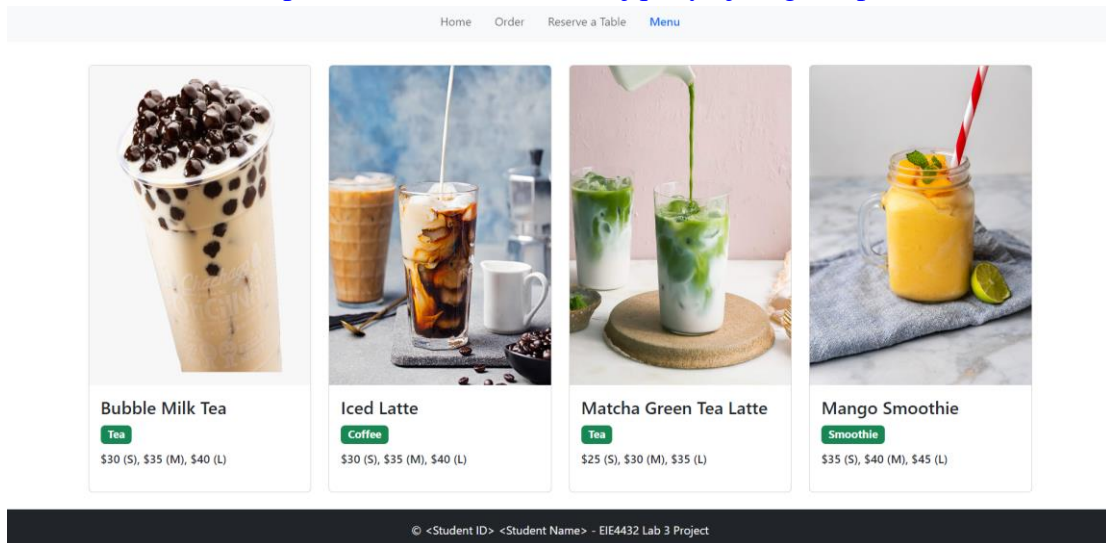
4. Create a new file “menu.js” and save it into “scripts” folder
5. Include jQuery & “menu.js” in the <head> section
6. Add a row inside container <div>, it is for displaying drinks in card components
  - Give an “id” to this new row, e.g. “drink-menu”
  - For large device, show at most 4 cards in a row
  - For medium device, show at most 2 cards in a row
  - For small device, show only 1 card in a row
  - Hint: Each column contains 1 card
  - Reference: <https://getbootstrap.com/docs/5.3/layout/grid/#row-columns>





7. Data fetching would be done by AJAX GET request using jQuery `$.get()` method, so that the menu can be real-time updated

- Reference: [https://www.w3schools.com/jquery/ajax\\_get.asp](https://www.w3schools.com/jquery/ajax_get.asp)



8. Create a `$(document).ready()` function to ensure code will run when the document has finished loading
9. Use the `$.get()` function to make a GET request to the “drink-menu.json”, make sure providing a **relative path** to the JSON file as the 1<sup>st</sup> argument
10. Inside the callback function, handle both success and fail cases, the `data` parameter will contain the fetched drink menu JSON data

```
$(document).ready(function () {
    $.get("<relative path of JSON file>", function (data) {
        // Success Case Handling Here
    }).fail(function (error) {
        // Fail Case Handling Here
    });
});
```

11. Display the fetched data in console, and display an error message in console if fails

### Lab #2.1

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

- Source code of “menu.js”
- Screen capture of console of **successful case**
- Screen capture of console of **fail case**

12. When a successful request is made, iterate over each drink in the data array and display with card components
  - Hint: Use either traditional `for-loop` or `forEach()` method for iteration
  - Card Component Sample: <https://getbootstrap.com/docs/5.3/components/card/#example>

13. For easy-to-read implementation, construct the HTML for each drink card using template literals (backticks: ```) & call variable using `${<value/variable>}` syntax. Here is an example of using this technique

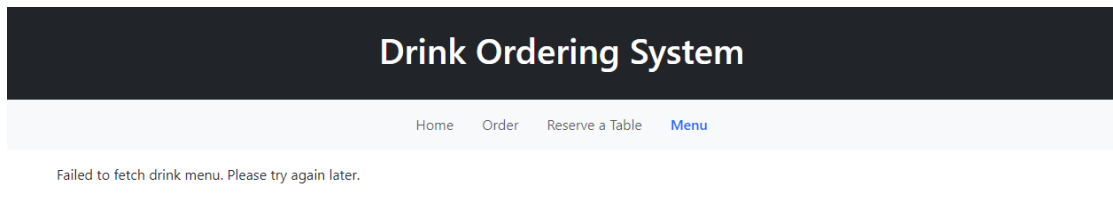
```
{
  name: "Purrscloud",
  species: "Cat",
  favFoods: ["wet food", "dry food", "<strong>any</strong> food"],
  birthYear: 2017,
  photo: "https://learnwebcode.github.io/json-example/images/cat-2.jpg"
},
```

```
<div class="animal">
  
  <h2 class="pet-name">${pet.name}</h2>
  <span class="species">${pet.species}</span></h2>
  <p><strong>Birth Year:</strong> ${pet.birthYear}</p>
</div>
```

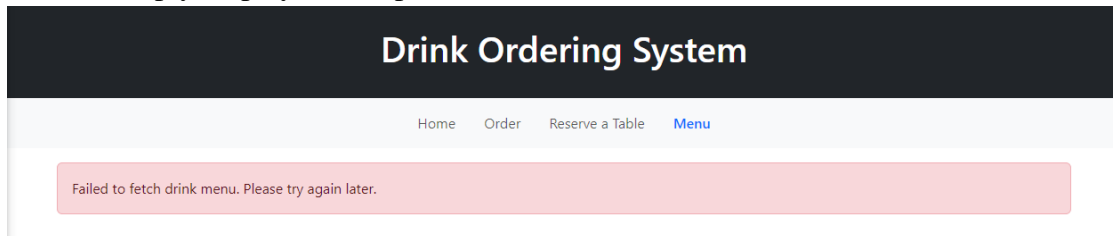
- Reference 1: [https://www.w3schools.com/js/js\\_string\\_templates.asp](https://www.w3schools.com/js/js_string_templates.asp)
- Reference 2: <https://www.digitalocean.com/community/tutorials/understanding-template-literals-in-javascript#expression-interpolation>

14. Append every constructed card to the “drink-menu” container using `.append()` method
15. Create more drink items (**at least 10 items**), and change to different viewport size to see the outcome

16. To handle fail request, convert the message displayed in console to the “drink-menu” container



- Simply display it with plain text



- (Optional) Use Bootstrap alert component to display the error message

### Lab #2.2

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

- Source code of “menu.js”
- Screen capture of menu page in large screen device (1 row 4 drink cards)
- Screen capture of menu page in medium screen device (1 row 2 drink cards)
- Screen capture of menu page in small screen device (1 row 1 drink cards)
- Screen capture of error message if request is failed

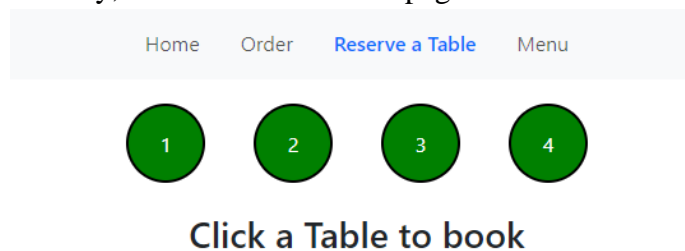
## Task 3: Advanced SVG Seat Map with LocalStorage Integration (60 mins)

### Expected Outcome

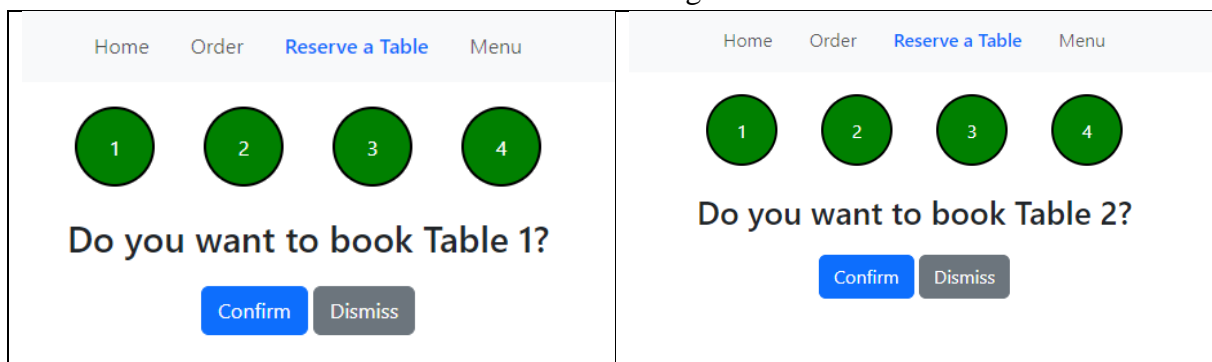
- Store the booking information in the browser's local storage using JavaScript
- Update the appearance of the SVG graphics dynamically based on the booking status
- Provide feedback to users about the availability and status of seats or tables, such as highlighting booked seats or displaying a legend

### Instructions

1. Go to “reserve.html”, it should be finished with 4 circles from Lab 2, and click event is allowed, so users can select the table they want to book
2. Create a new file “reserve-table.js” and save it into “scripts” folder
3. Include both jQuery & “reserver-table.js” for this file
4. Remove the `selectTable(tableNumber)` function implemented last time and delete the `<script>` tag
5. Remove all onclick event listener for all `<circle>` elements
6. Add “id” to each `<circle>` for uniquely identify them later in the script
7. Create a style for class “booked” in `<style>`, for indicating booked seat later
  - It should be similar to the “table” class style created in Lab 2
  - Red in fill color
  - Black stroke color
  - Stroke-width is a custom number
  - Cursor will be in “not-allowed” value to indicate the table is not clickable
8. Modify the UI as following:
  - Initially, when user enters this page



- When user clicks and table for booking



9. Next, go to “reserve-table.js” to implement the functionality for booking table
10. Again, create a `$(document).ready()` function to ensure code will run when the document has finished loading

11. We will use localStorage to store the booking status with a key “bookedTable”, an array will be stored to indicate which table(s) is/are booked.

Initially, there is no such key in localStorage, it will be created when there is first booking, here is an example of the localStorage record

■ **Array must be stored, instead of string**

Key	Value
bookedTable	[1, 2]

▼ ["1", "2"]  
0: "1"  
1: "2"

12. Create 2 variables for remembering booking status & selected table

■ **Variable 1: booking information from localStorage using**

**localStorage.getItem()**

- As it would store as JSON string later, **JSON.parse()** method is needed to convert it into array object for implementation
- Otherwise, if no such key found in localStorage, store an empty array to this variable

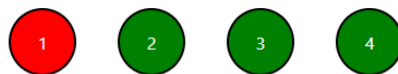
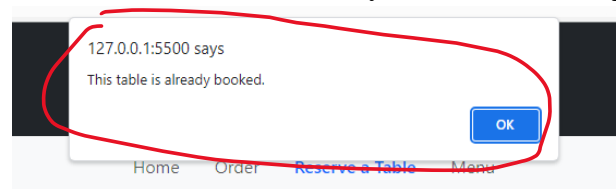
■ **Variable 2: selected table ID, initially there is no table selected**

13. When all elements are loaded, update the appearance of the tables according to booking status found in localStorage

- Iterate each table by all means (Hint: **for-loop, forEach(), .each()**)
- For each table, get the value of its id using **.attr()** method
- Check whether the ID is existing in the localStorage or not
- If it is booked already, add the “booked” class to the table using **.addClass()** method

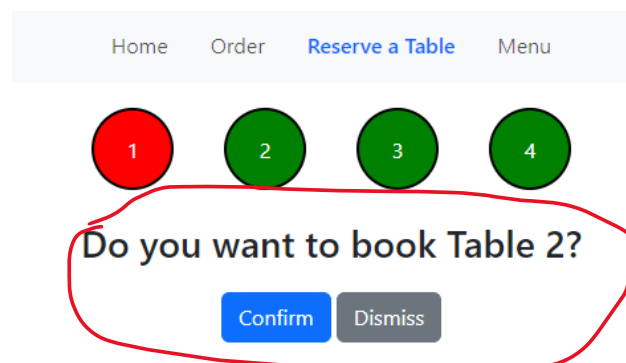
14. When each table is clicked, the system should ask user is going to book it or not

- If the table is booked already, show an alert to prompt user it is not available



Click a Table to book

- Otherwise, ask for confirmation by showing which table is selected, and show the action buttons



15. When “Confirm” button is clicked:

- Add the table ID to the array of booked table
- Save the booked table data back to localStorage by using `localStorage.setItem()` method
- However, we would like to store the data as JSON string instead of normal string, `JSON.stringify()` method can be used to preserve the array data type
- Then, update the appearance of the table by adding class “booked”
- Do a table status update again from localStorage, so user can see the table color changes once “Confirm” button is clicked
- Hide the booking buttons by any means

16. Vice versa, when “Dismiss” button is clicked:

- Show the original message for asking user to click a table to book
- Hide the booking buttons by any means

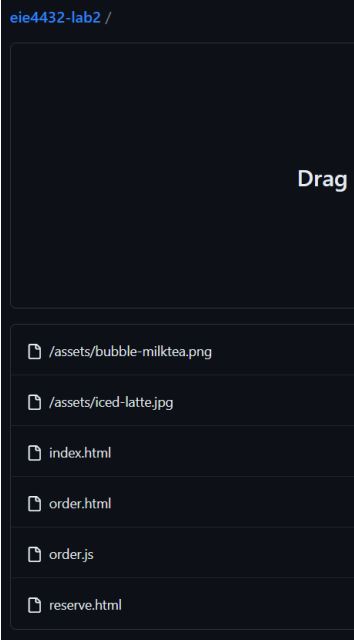
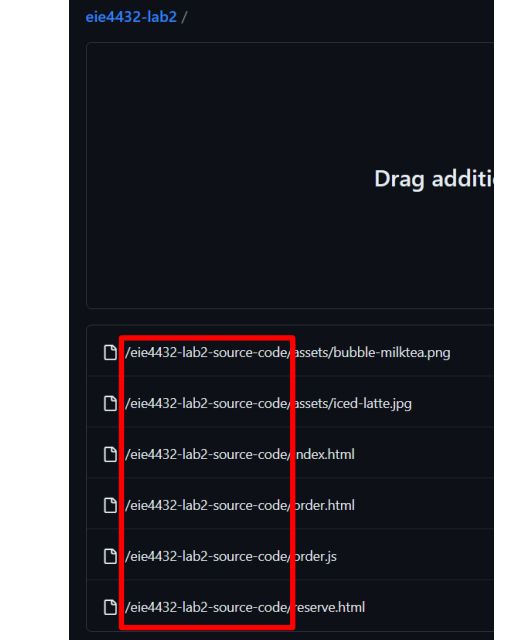
### **Lab #3.1**

Provide the following screenshots with your **Student ID and System Date & Time**. Your code should be indented correctly (each nested tag should be indented exactly once inside of its parent tag)

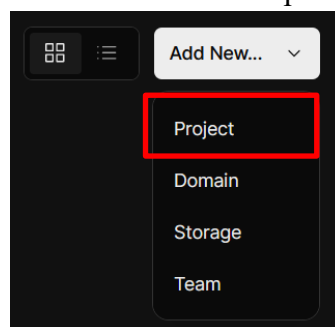
- Source code of “reserve-table.js”
- Screen capture of initial state that 4 tables have not been booked
- Screen capture of a table is booking
- Screen capture of a table is being booked successfully
- Screen capture of a table is booked, but user clicks again
- Screen capture of **at least 2 tables** are booked successfully
- Screen capture of the localStorage value (from Inspector > Application/Memory)

## Task 4: Publish your Website (Optional)

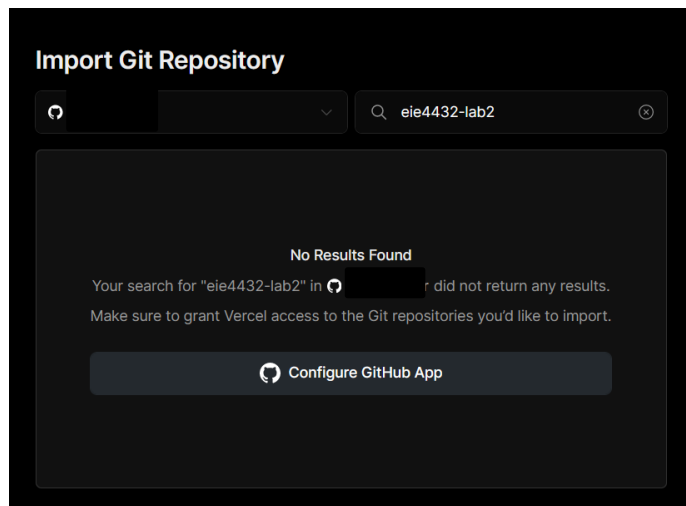
- Make sure the source code is uploaded to GitHub as a new **PRIVATE** repository
  - Do not upload a whole folder to repository, it would affect how the web server access the files
  - Make sure there is a "index.html" file in your repository

Correct Practice (Without further folder name)	Wrong Practice (With further folder name)
	

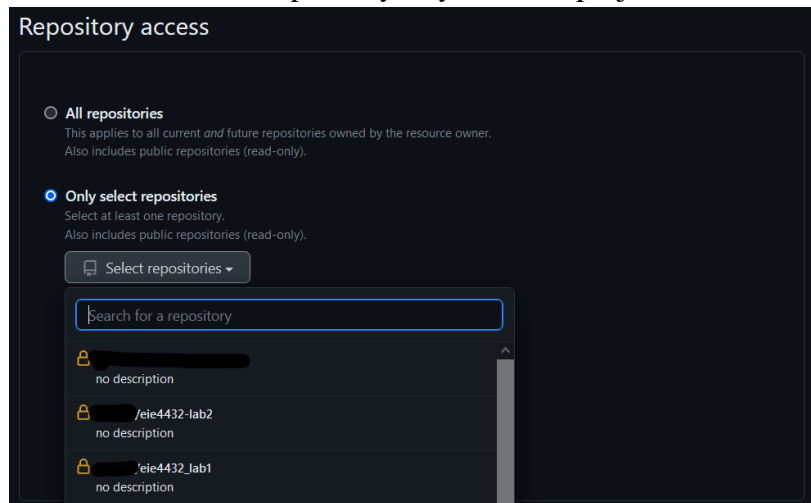
- Go to Vercel (<https://vercel.com/>) to sign up an account with **GitHub** account to allow the best integration with the platform. For detailed instructions on how to register a Vercel account, please refer to the last two pages of this lab manual.
  - This is a cloud platform for deploying and hosting websites and web applications
  - It specializes in static site generation and serverless functions
  - Offers seamless integration with popular frontend frameworks like Next.js and Nuxt.js
  - Provides features like automatic deployments, domain management, and scalability
  - Supports collaborative development and offers various deployment options, including Git-based workflows
- In the dashboard of Vercel, click “Add New...” dropdown menu, and select “Project”



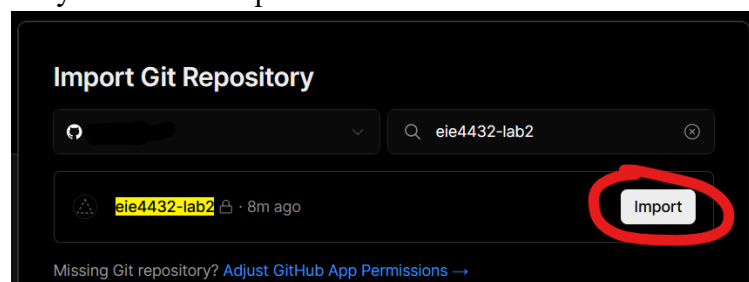
- Search the repository name in the search field, as it is a **PRIVATE** repository, it can't be found directly, click “Configure GitHub App” to grant access for Vercel



5. A pop-up window asks to enter password, after that, it will redirect to a setting page
6. Scroll to find “Repository access” section, select “All repositories” is the most convenient method
7. If you decided to select “Only select repositories”, click “Select repositories” dropdown menu, and select the repository of your lab 2 project



8. After setting, click “Save” and return back to Vercel
9. Find the repository and click “Import”



10. Deploy the website as the following settings, click “Deploy” when everything is ready
  - Note: Uppercase letters & underscore are not allowed in project name



### Configure Project

Project Name  
student\_id-eie4432-lab2

Framework Preset  
Other

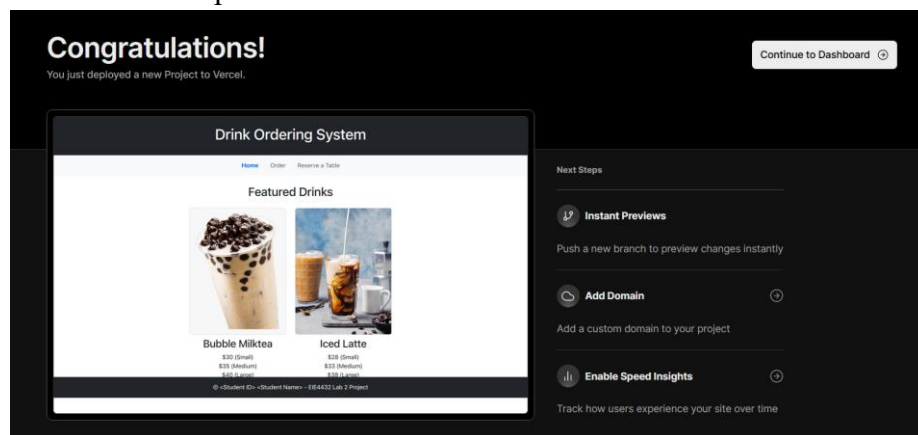
Root Directory  
./

> Build and Output Settings

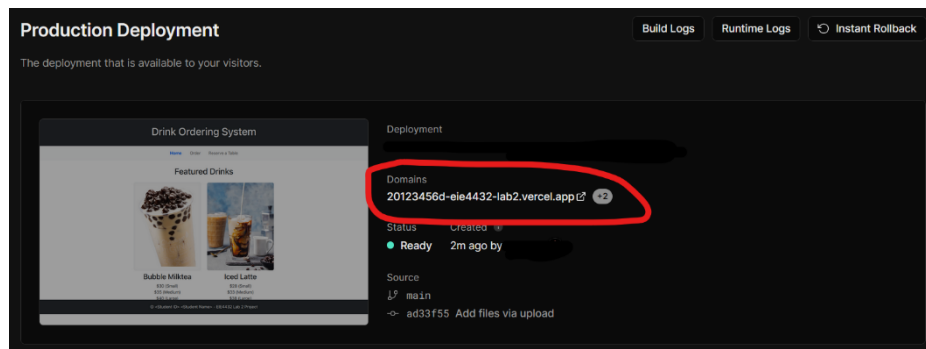
> Environment Variables

Deploy

11. You will see following screen when it is successfully deployed, click “Continue to Dashboard” to find the public URL



12. Public URL can be found under “Domains”



#### Lab #4.1 (Optional)

Provide the public URL of the lab project, eg. <https://xxx.vercel.app>.

## Appendix: Vercel Account Registration Guide

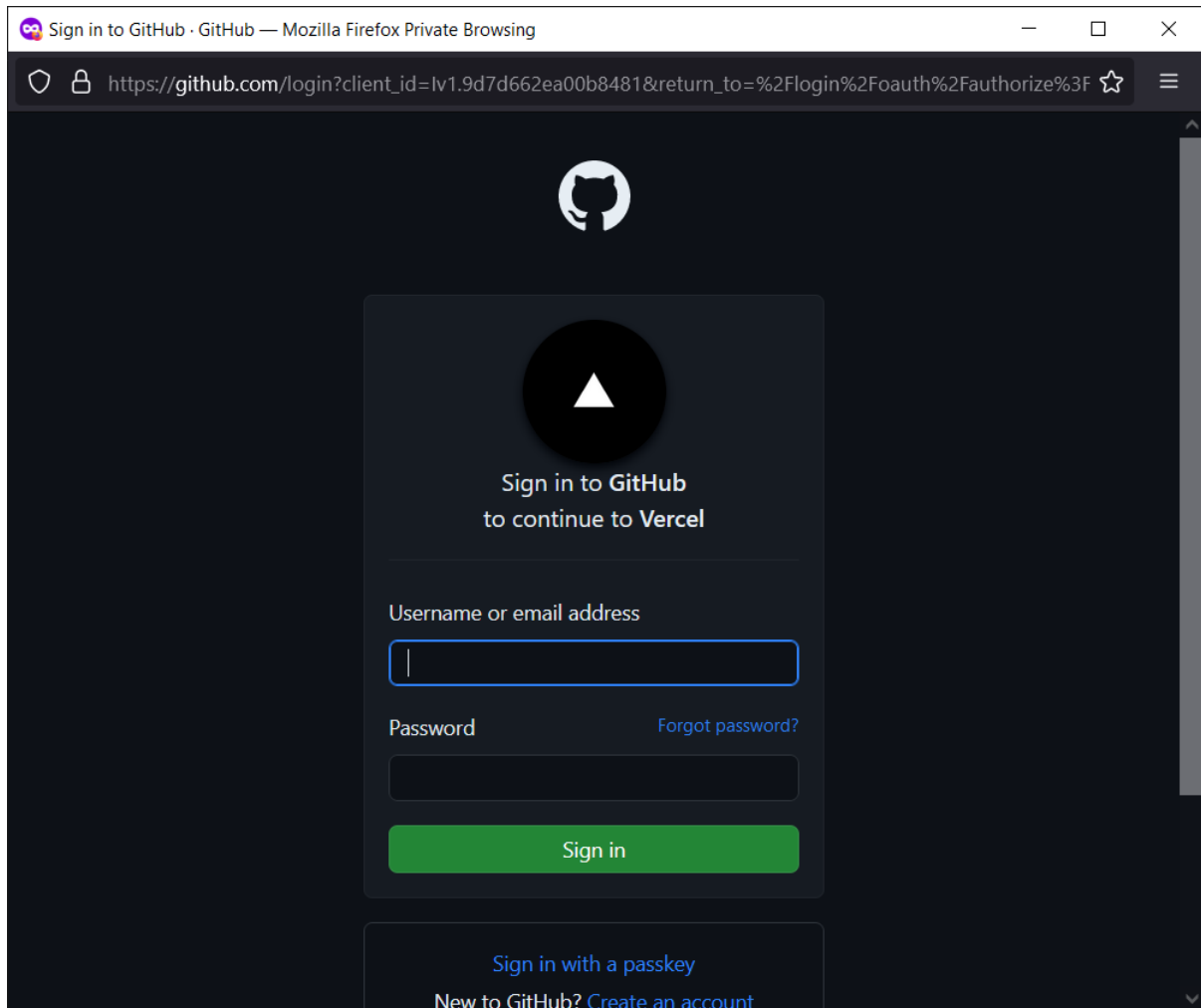
1. Go to <https://vercel.com>, click “Sign Up”, you will be redirected to this page. Select “Hobby” and provide a name, then click “Continue”.

The screenshot shows the Vercel sign-up page. At the top, there's a navigation bar with the Vercel logo and links for 'Contact' and 'Log In'. The main heading is 'Create Your Vercel Account'. Below this, there's a 'Plan Type' section with two options: 'Hobby' (selected with a blue checkmark) and 'Pro'. The 'Hobby' option is described as 'I'm working on personal projects'. The 'Pro' option is described as 'I'm working on commercial projects'. Below the plan type, there's a 'Your Name' field with a text input box. At the bottom of the form is a 'Continue' button. Below the button, there's a line of text: 'By joining, you agree to our [Terms of Service](#) and [Privacy Policy](#)'.

2. Click “Continue with GitHub”.

The screenshot shows the Vercel page for connecting a Git provider. The heading is 'Let's connect your Git provider'. Below this, there are three buttons: 'Continue with GitHub' (grey), 'Continue with GitLab' (purple), and 'Continue with Bitbucket' (blue). Below these buttons is a link 'Continue with Email →'. At the bottom, there's a line of text: 'By joining, you agree to our [Terms of Service](#) and [Privacy Policy](#)'. At the very bottom, there's a link: 'Have a complex company use case? [Get enterprise-grade assistance](#)'.

3. Provide your GitHub login credentials, click “Sign In”.



4. Vercel account is successfully created without providing any personal information!

**END OF LAB 3**