IS216 *Mock* Lab Test 1 [36 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 "KIM Pyong Yang" and email ID is kim.pyongyang.2023, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files

`<!--`
`Name:  KIM Pyong Yang`
`Email: kim.pyongyang.2023`
`-->`
	`/*`
	`Name:  KIM Pyong Yang`
	`Email: kim.pyongyang.2023`
	`*/`

- 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://drive.google.com/file/d/1uFh9kuF9AV2cj4UUNWX1qK4yt_6hjlVz/view

Submission Instructions

- **Due Date**
  - **3 October 2023 (Thursday) 90 minutes into mock lab test start 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, **kim.pyongyang.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-G3/G4 eLearn** page: https://elearn.smu.edu.sg/d2l/home/393850
  - Go to **Assignments → Week 7 – Mock LT1 (In Class) →** ***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.

Table of Contents

Q1. CSS: Conference Speakers [10 Marks]

Given resources in folder `Q1`

✏️ `conference.html`

✏️ `conference.css`

🚫 `photos/* (there are 4 PNG image files)`

Scenario

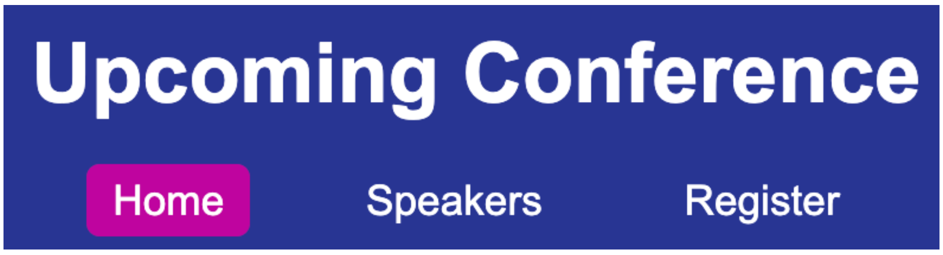
You are tasked with creating a landing page for an upcoming conference. The page consists of a header, a main content section, and a footer.

Instructions

- Open conference.html, conference.css, and style the page using **only vanilla CSS**. You **cannot use any frameworks such as Bootstrap**.
- The HTML page (when rendered by the web browser) does **NOT** have to be responsive. Use a screen width of **800px** for this question *(see on Pages 6-8 for sample screen grabs)*.

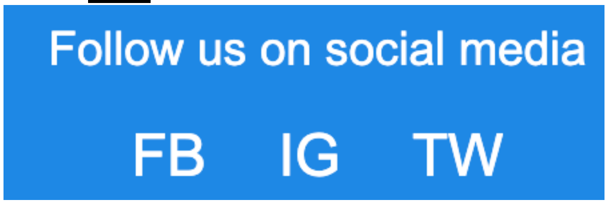
Tasks

1. **Header (3 marks)**
   - The header must have a background color of **#283593** and **white** text.
   - The navigation links should be displayed **inline** with **spacing** between them.
     1. The links should have a **hover** effect, changing the background color to **#BF039F** and **rounding the corners**.
     2. (See below) When hover over "Home", its background color changes and the box encasing the text has rounded corners. The same applies to "Speakers" and "Register".

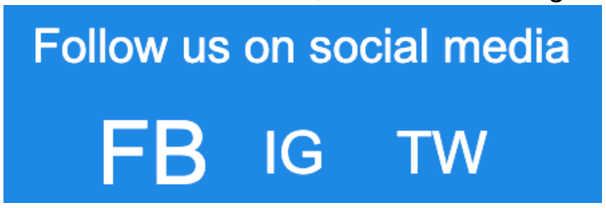


2. **Main Content (2 marks)**
   - The speakers' section should use a **three-column layout**, where each speaker's profile has a border, an image, and a description.
   - **All images of the speakers must have a fixed height of 100px** while maintaining their aspect ratio (width should auto-adjust accordingly).
   - Each speaker card should have a hover effect that slightly **enlarges the card** by a factor of **1.05**.

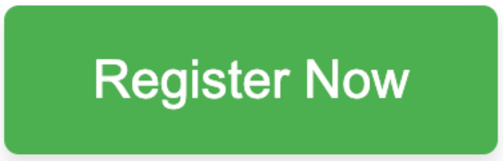
3. **Footer (2 marks)**
   - The footer should **always stay at the bottom of the page**, with a background color of **#1E88E5**.
   - Add social media icons (simple text, such as "FB," "IG," "TW") that **scale up** when hovered over.
     1. When <u>**NOT**</u> hovered over:



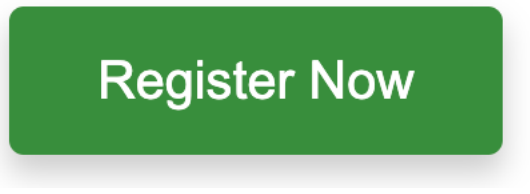
     2. When **hover over "FB"**, the text **FB** enlarges by a factor of **1.5**.



4. **Button (3 marks)**
   - Include a "Register Now" button in the main content area that has a green background **(#4CAF50**), rounded corners, and a hover effect that changes the color to a darker green **(#388E3C**) and adds a shadow.
   - The button does <u>**NOT**</u> have any border.
   - The font size of the text inside of the button should be **18 pixels**.
   - When NOT hovered over:

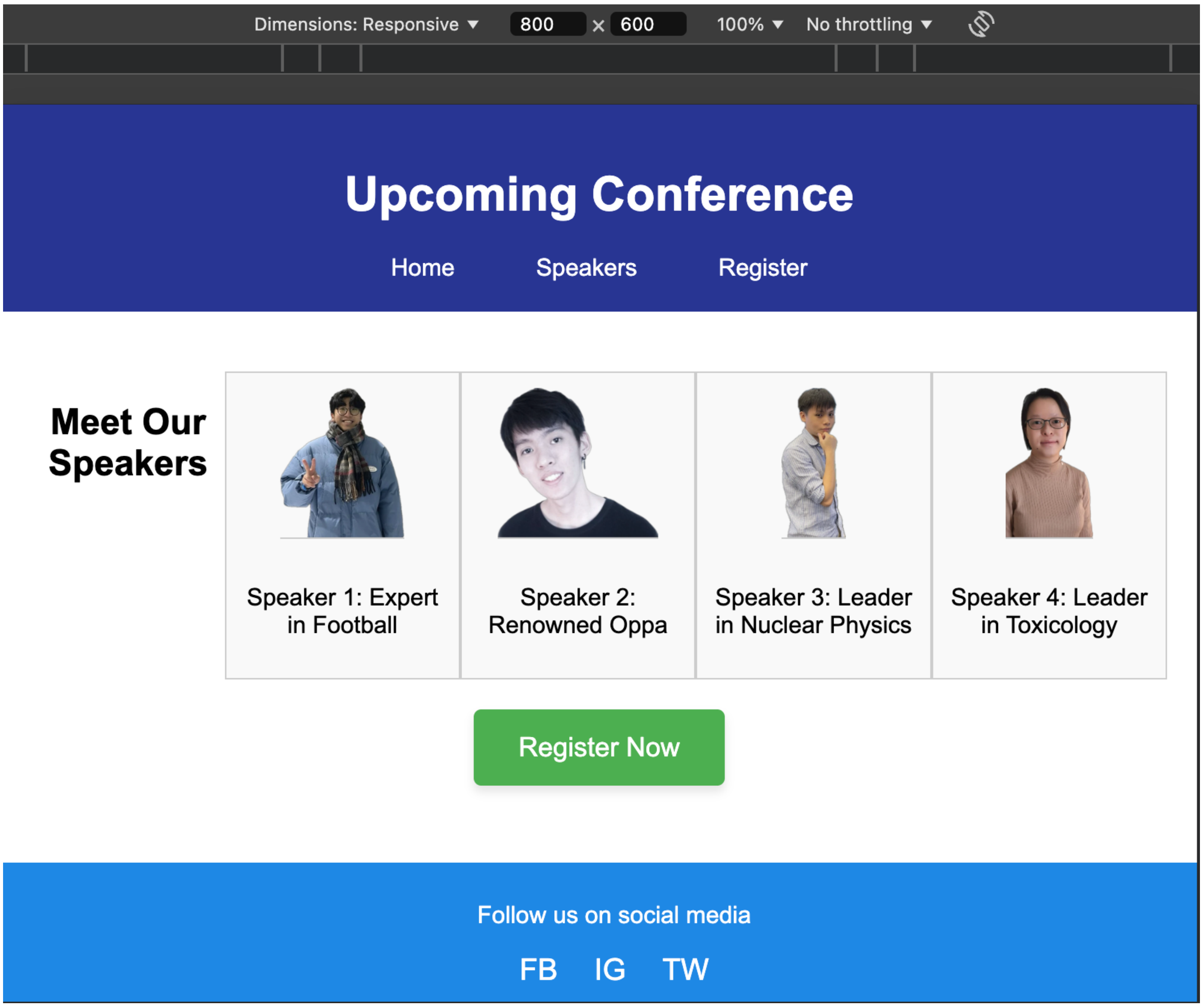


   - When **hover over** the button, the button's color becomes **darker green**. There is a **shadow** around the box.



conference.html

(Upon fresh page loading where the user hasn't done any action such as clicking or hovering)



Upcoming Conference

Home Speakers Register

Meet Our Speakers

Speaker 1: Expert in Football

Speaker 2: Renowned Oppa

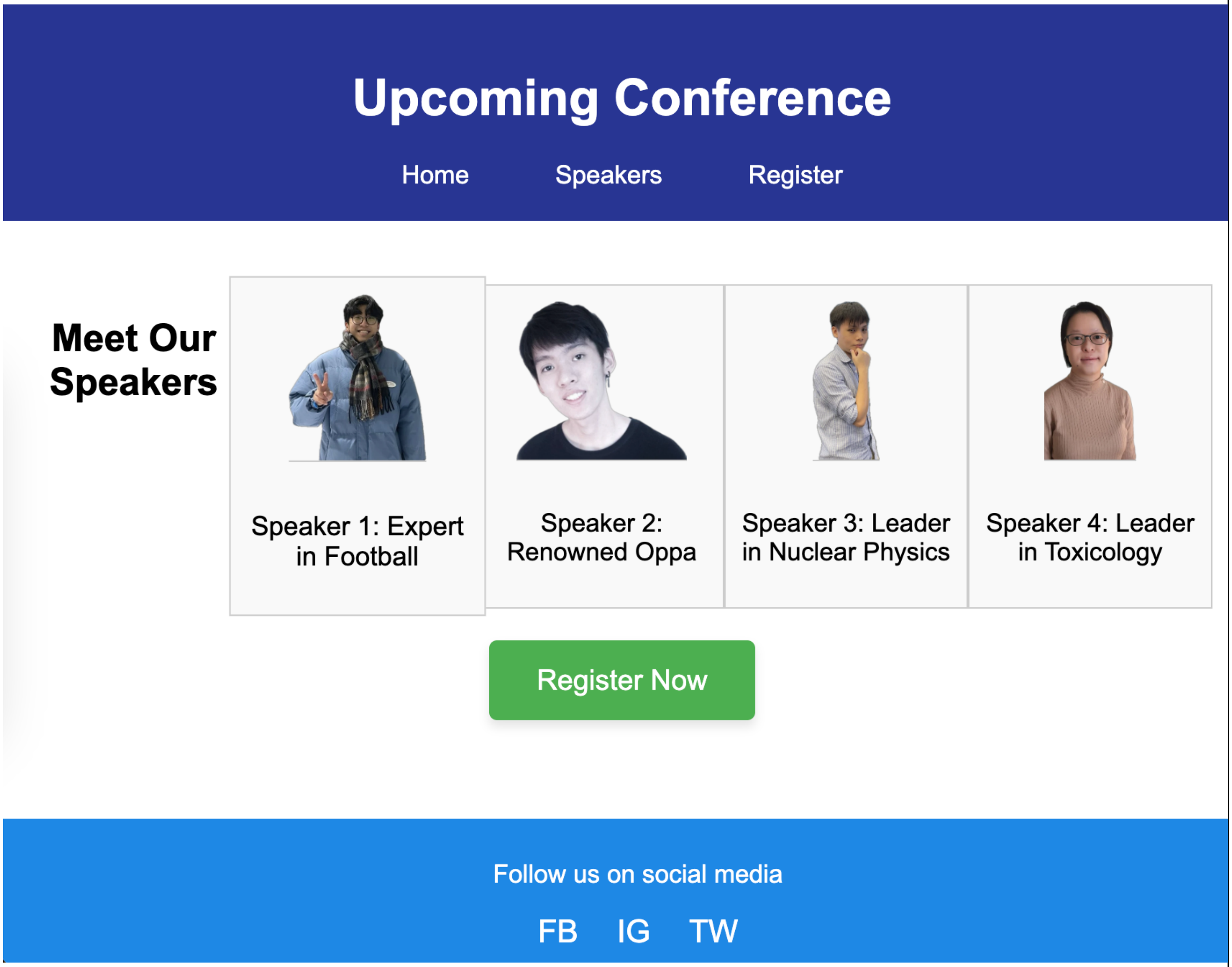
Speaker 3: Leader in Nuclear Physics

Speaker 4: Leader in Toxicology

Follow us on social media

conference.html

*(Hover over **Speaker 1**)*


Upcoming Conference

Home Speakers Register

Meet Our Speakers

Speaker 1: Expert in Football

Speaker 2: Renowned Oppa

Speaker 3: Leader in Nuclear Physics

Speaker 4: Leader in Toxicology

Register Now

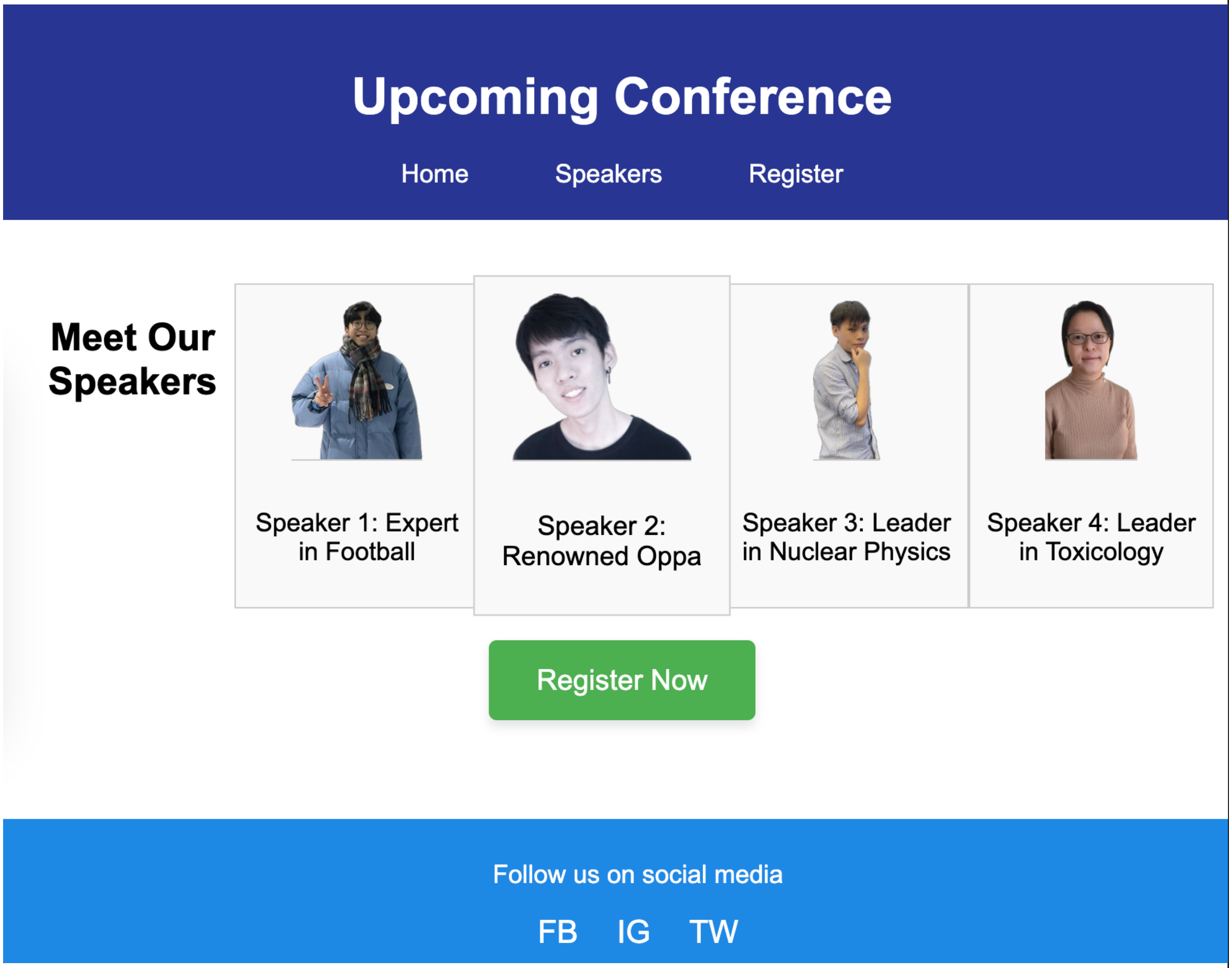
Follow us on social media

FB IG TW

When the user **hovers** (mouse cursor) over **Speaker 1**, the card that encases the speaker's information **enlarges**.

conference.html

*(Hover over **Speaker 2**)*



Upcoming Conference

Home Speakers Register

Meet Our Speakers

Speaker 1: Expert in Football

Speaker 2: Renowned Oppa

Speaker 3: Leader in Nuclear Physics

Speaker 4: Leader in Toxicology

When the user **hovers** (mouse cursor) over **Speaker 2**, the card that encases the speaker's information **enlarges**.

Similarly, for **Speaker 3** and **Speaker 4**, hovering over each speaker must result in each speaker's card to **enlarge**.

Q2. Bootstrap: Products [9 Marks]

Given resources in folder `Q2`

✏️ `product.html`

✏️ `product.css`

🚫 `photos/* (there are 3 JPG image files)`

Scenario

You are tasked with creating a **product showcase** for an online store. The showcase will display three products with brief descriptions, and users can click on a **"View Details"** button for each product to view more information via a **Bootstrap modal**. The page must be **responsive**, using **Bootstrap's grid system** to ensure a seamless layout on various screen sizes.

Instructions

1. You will create a responsive product display using **Bootstrap's grid system**.
2. Each product should have a name, price, and image, along with a **"View Details"** button that opens a **Bootstrap modal** with additional information about the product.
3. The page should dynamically adjust from a **three-column layout on large screens** (lg breakpoint) to a **single-column layout on small screens** (sm breakpoint).
4. You are **NOT** allowed to use JavaScript for triggering the modal; **rely solely on Bootstrap's modal attributes**.

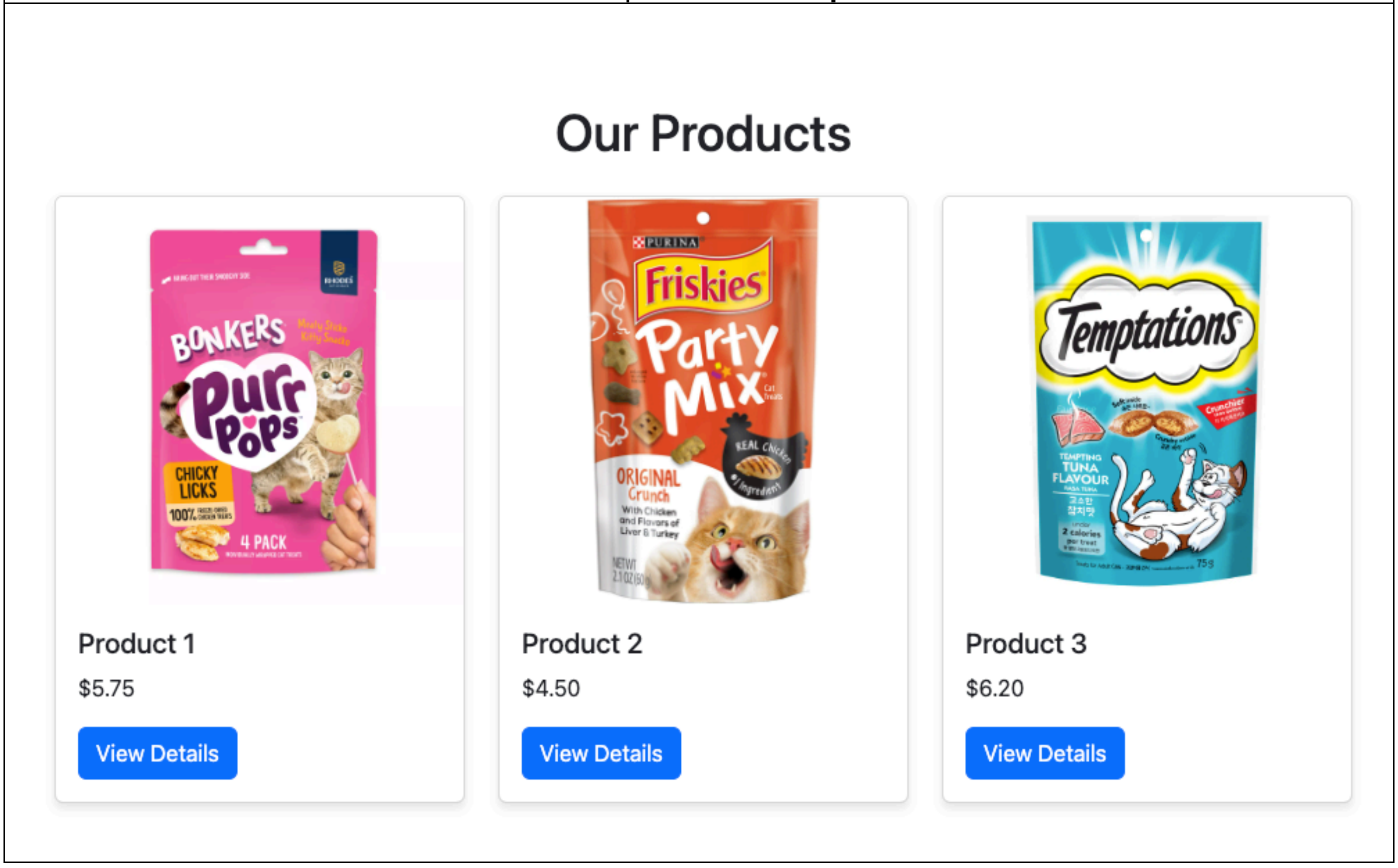
Tasks

1. **Create the Product Grid (3 marks)**
   - At the lg (≥ 992px) breakpoint, display the products in a **three-column layout**.
   - At the md (≥ 768px but < 992px) breakpoint, display the products in a **two-column layout**.
   - At the sm (< 768px) breakpoint, switch to a **single-column layout**.
   - Each product should have an image, name, price, and a **"View Details"** button.
2. **Use Bootstrap Modals (3 marks)**
   - When users click the **"View Details"** button, display more information about the product in a **Bootstrap modal**.
   - Each product must have its own modal with details such as the product name, description, and additional images.
3. **Styling (3 marks)**
   - Use Bootstrap's components and utilities for spacing, buttons, and layout.
   - Ensure that the design looks modern and clean without additional heavy custom CSS. You may add (should you need) additional vanilla CSS code in **product.css**.

product.html

(Upon fresh page loading where the user hasn't done any action such as clicking)

Web browser viewpoint width: **992px** and **above**



Our Products

Product 1

$5.75

View Details

Product 2

$4.50

View Details

Product 3

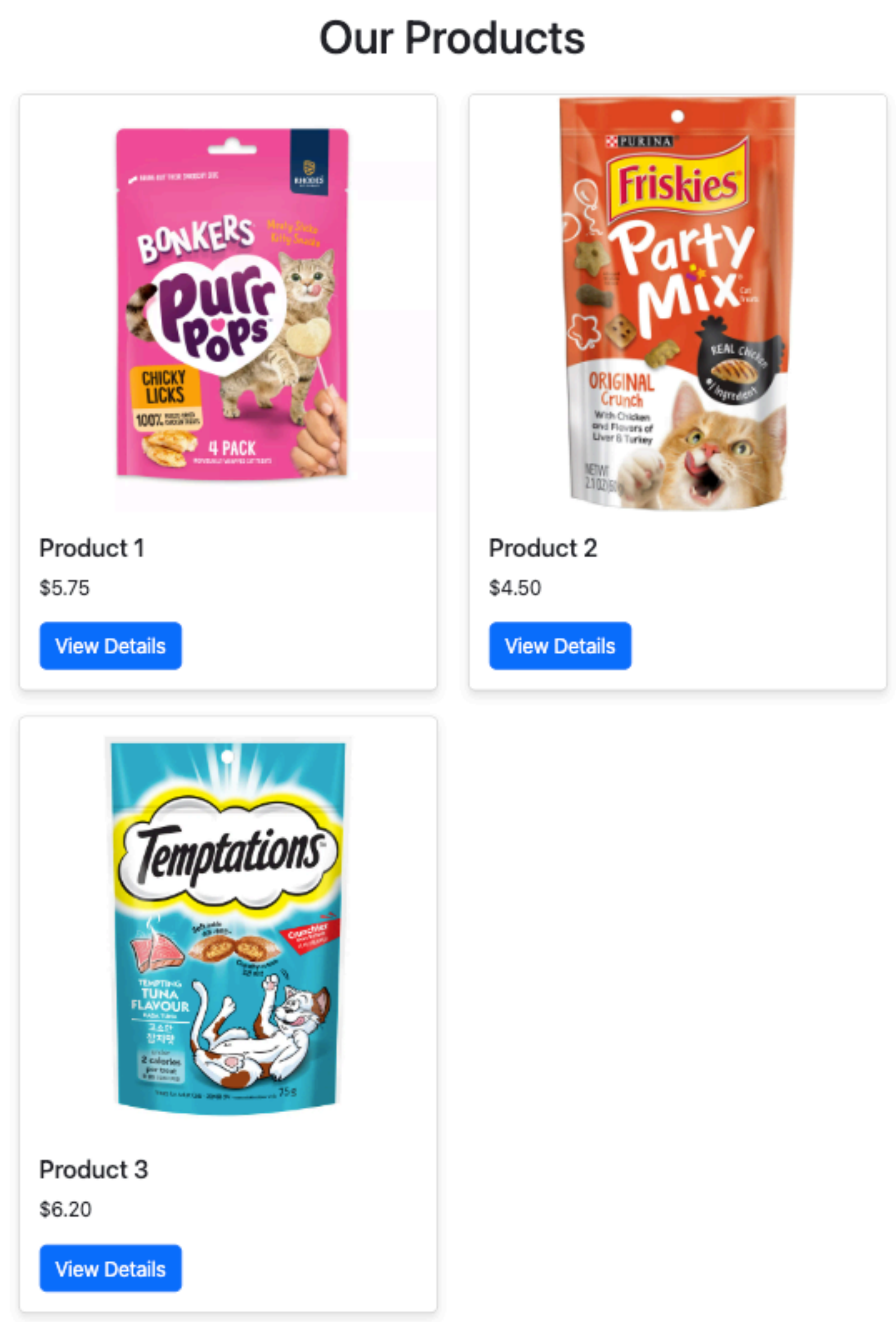
$6.20

View Details

product.html

(Upon fresh page loading where the user hasn't done any action such as clicking)

Web browser viewpoint width: **768px** to **991px** *(both inclusive)*



Our Products

Product 1

$5.75

View Details

Product 2

$4.50

View Details

Product 3

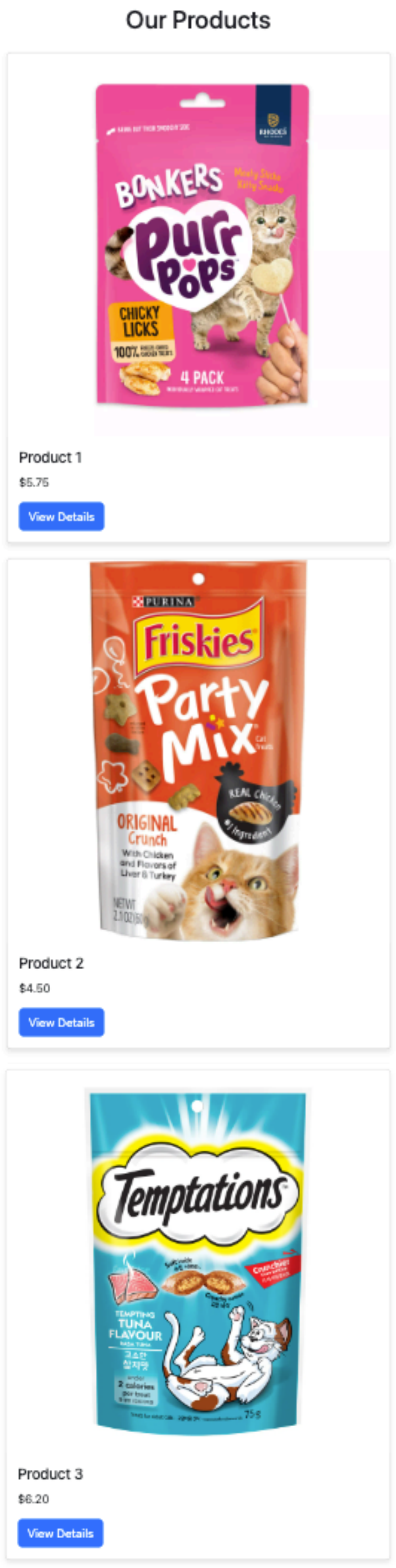
$6.20

View Details

product.html

(Upon fresh page loading where the user hasn't done any action such as clicking)

Web browser viewpoint width: **767px** and **below**



Our Products

Product 1

$5.75

View Details

Product 2

$4.50

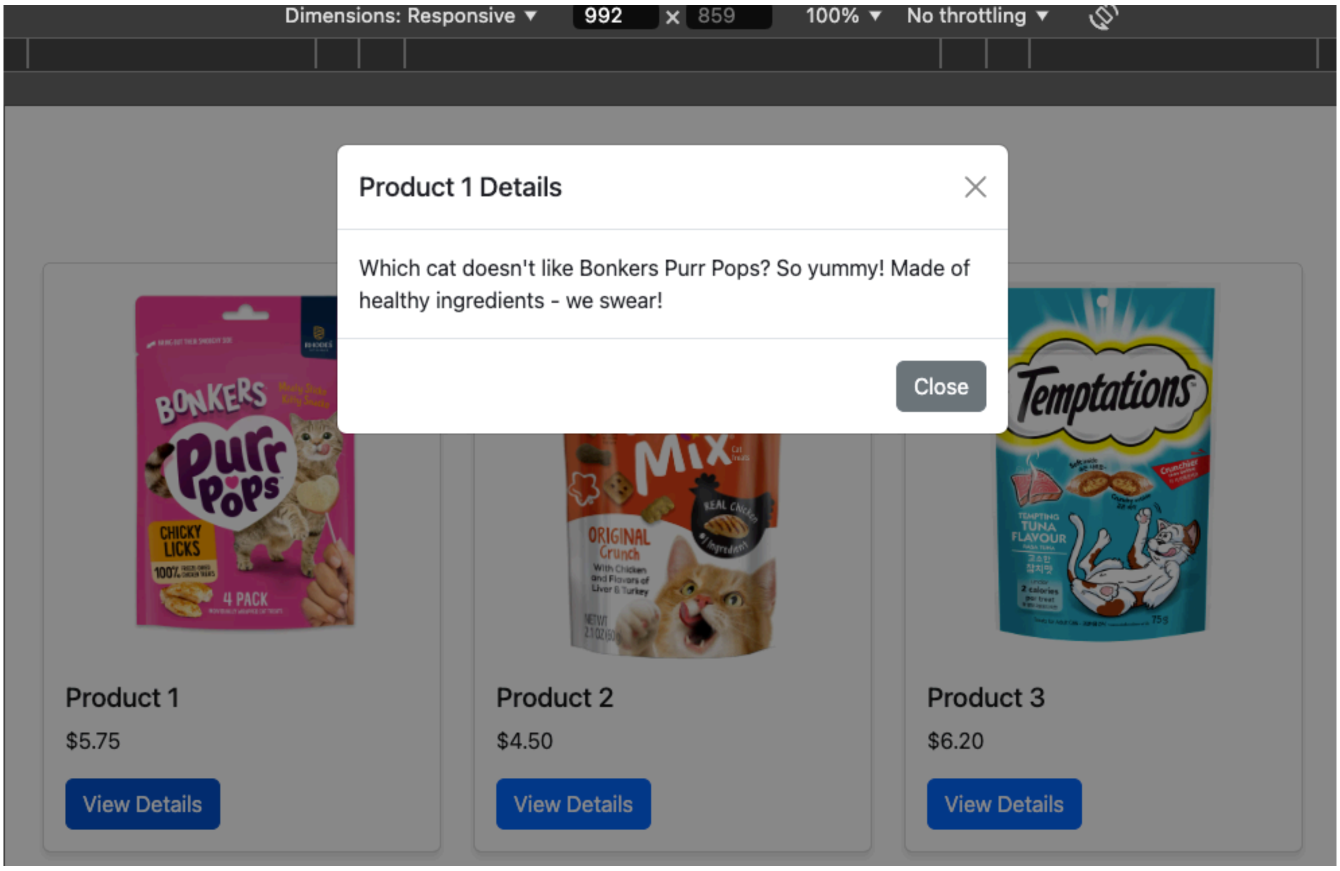
View Details

Product 3

$6.20

View Details

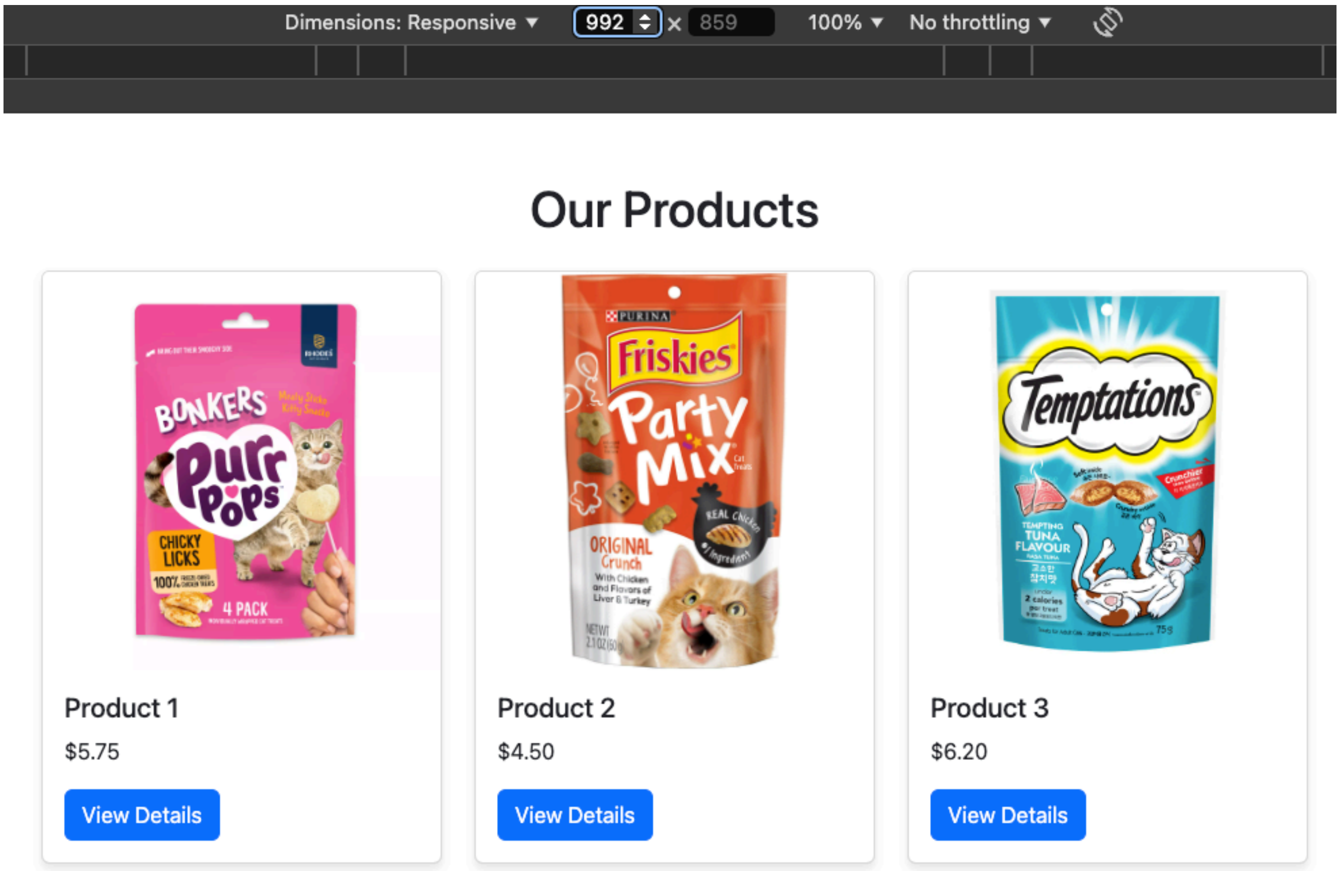
When the user clicks on **Product 1**'s **View Details** button, a modal will be displayed **(as shown below)**.



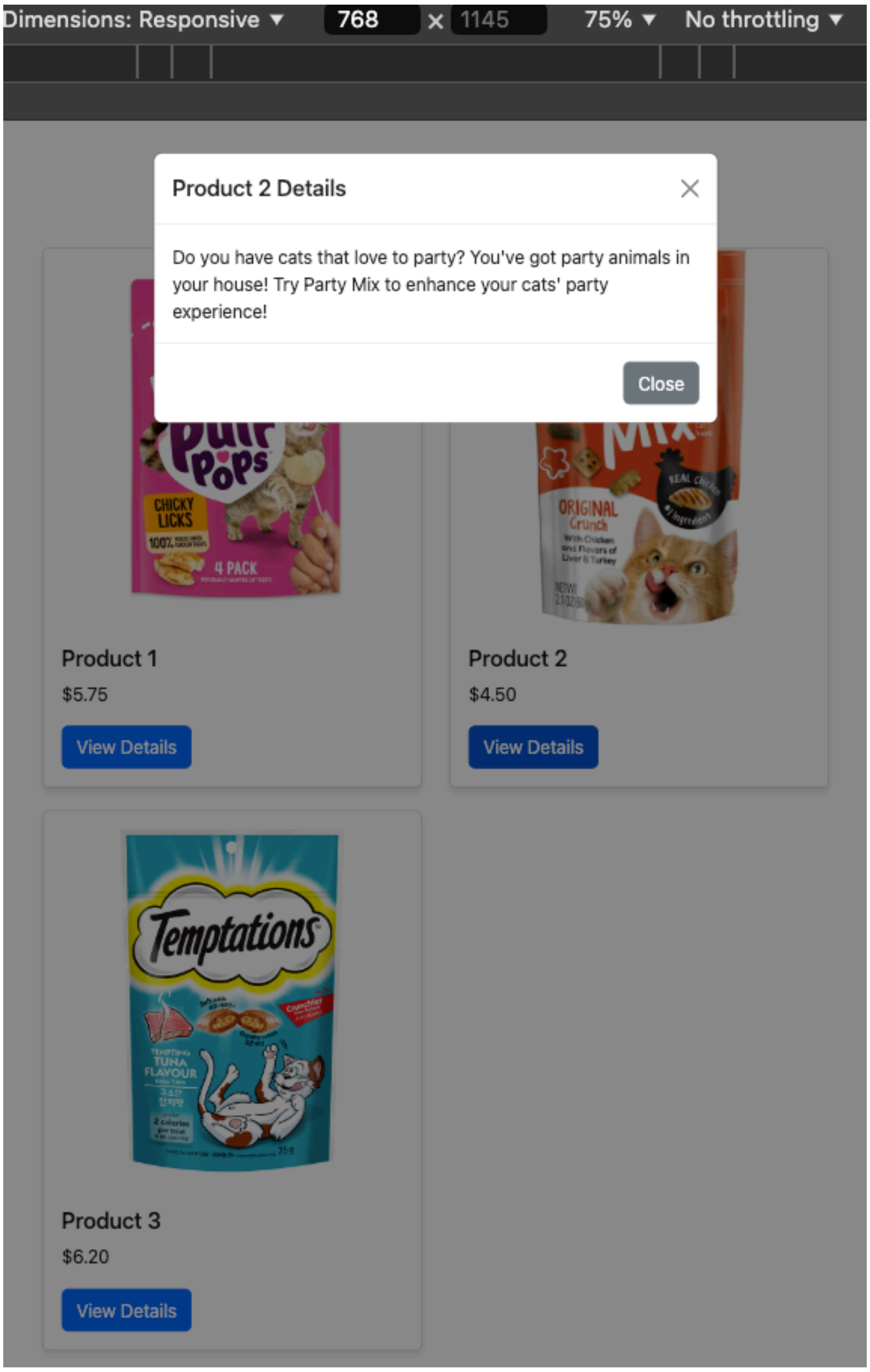
Product 1 Details

Which cat doesn't like Bonkers Purr Pops? So yummy! Made of healthy ingredients - we swear!

When the user clicks on **Close** button (in the modal), the modal must close, and the user will return to the main screen **(as shown below)**.



When the user clicks on **Product 2**'s **View Details** button, a modal will be displayed **(as shown below)**.



Product 2 Details

Do you have cats that love to party? You've got party animals in your house! Try Party Mix to enhance your cats' party experience!

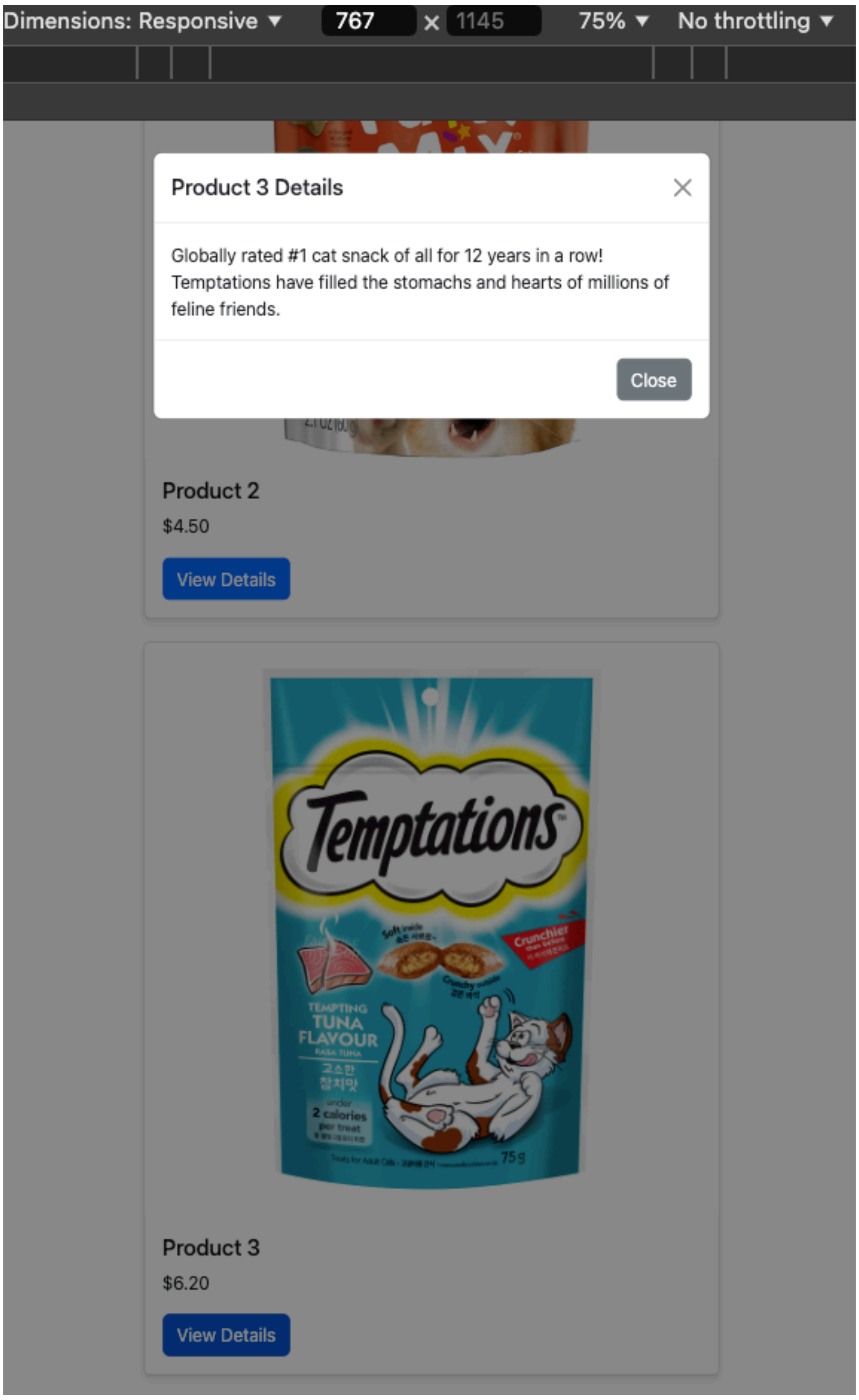
Product 1
$5.75

Product 2
$4.50

Product 3
$6.20

When the user clicks on **Close** button (in the modal), the modal must close and the user will return to the main screen.

When the user clicks on **Product 3**'s **View Details** button, a modal will be displayed **(as shown below)**.



Product 3 Details

Globally rated #1 cat snack of all for 12 years in a row! Temptations have filled the stomachs and hearts of millions of feline friends.

Product 2

$4.50

Product 3

$6.20

When the user clicks on **Close** button (in the modal), the modal must close, and the user will return to the main screen.

Q3. JavaScript DOM: Wishlist [9 Marks]

Given resources in folder `Q3`

🚫 `wishlist.html`

✏️ `wishlist.js`

✏️ `wishlist.css`

Scenario

You are tasked with creating a **product wishlist** application where users can add items to a wishlist, remove them, and mark items as purchased. The wishlist must be updated dynamically using **JavaScript DOM manipulation**. You will create and manipulate elements such as buttons and list items and use event listeners to handle user interactions like adding, removing, and marking products as purchased.

Instructions

1. The **HTML** and **CSS** structure for the page is already provided. You may modify the CSS file **though it already has all the CSS styling required for this question**.
2. Implement the **JavaScript functionality** to dynamically manage the wishlist.
3. The page should allow users to:
   a. Add a product to the wishlist using the provided input field.
   b. Remove items from the wishlist.
   c. Mark items as purchased, which changes the appearance of the text (adds a strikethrough) and changes the button background to black.

Tasks

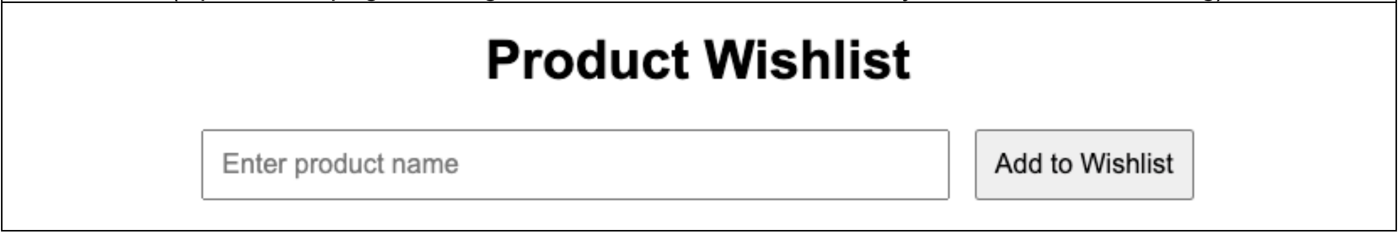
1. Before embarking on the below tasks, please open and inspect **wishlist.html**.
   - In particular, see **2 sample list items** inside of **<ul id='wishlist'>**. Uncomment each and see how each **list item** (that your JavaScript code will generate) should appear.
2. **Adding Products to the Wishlist (3 marks)**
   - Allow users to add products to the wishlist by typing a product name and clicking the "Add to Wishlist" button.
   - Capture user input from the “product input” input field and append a new item to the wishlist (un-ordered list). Each product should indicate the **Product Name** (user input) and appear with two buttons:
     - **"Remove"**: Removes the product from the wishlist.
     - **"Mark as Purchased"**: Strikes through the product name to indicate that it has been purchased.
3. **Removing Products from the Wishlist (3 marks)**
   - Implement the functionality to remove a product from the wishlist.
   - Each product in the wishlist should have a **"Remove"** button.
   - When the **"Remove"** button is clicked, remove the corresponding item from the list.

4. **Marking Products as Purchased (3 marks)**

   - Allow users to mark a product as purchased, which applies a **strikethrough** to the product name and changes the button background to **black**.
   - Each product should have a **"Mark as Purchased"** button.
   - When the button is clicked:
     - Add a strikethrough to the product name.
     - Change the button text to **"Unmark as Purchased"** and the background color to **black**.
   - Clicking the **"Unmark as Purchased"** button should:
     - Remove the strikethrough.
     - Change the button text back to **"Mark as Purchased"** and the background color to **green**.

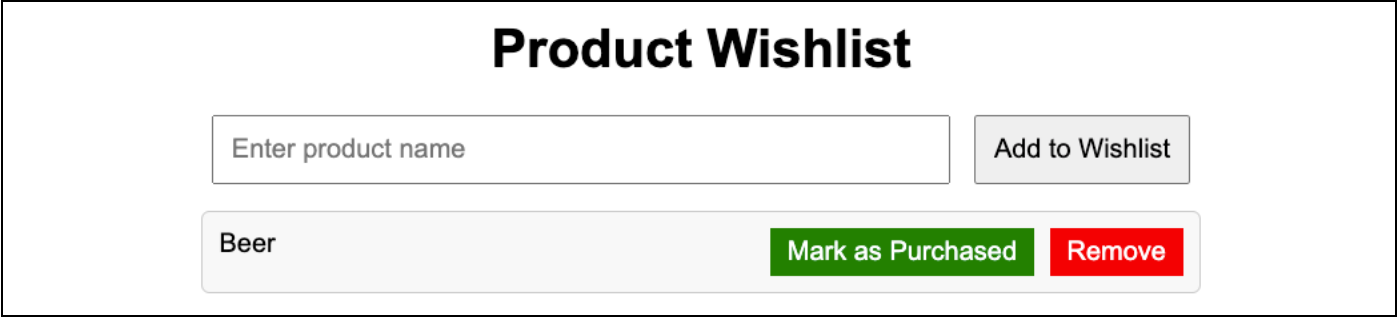
wishlist.html

(Upon fresh page loading where the user hasn't done any action such as clicking)

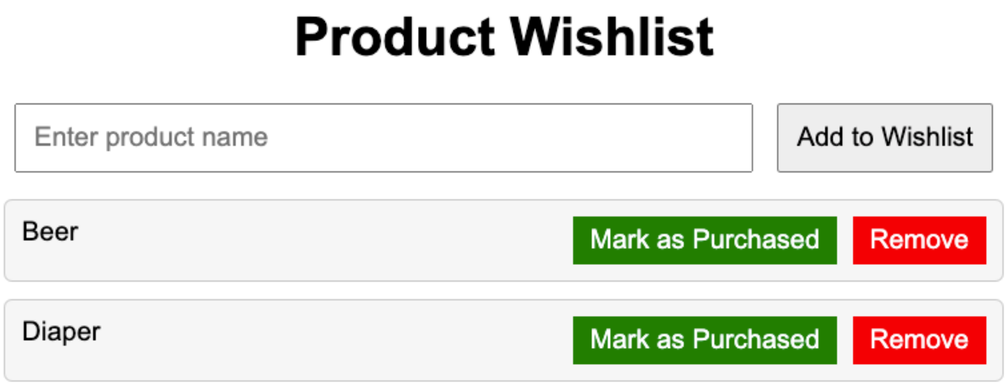


wishlist.html

*(Subsequently, upon keying in **Beer** in the input field and pressing **Add to Wishlist** button)*



As a result, a new box containing **Beer** and associated buttons has been **added**.

wishlist.html *(Subsequently, upon keying in **Diaper** in the input field and pressing **Add to Wishlist** button)*


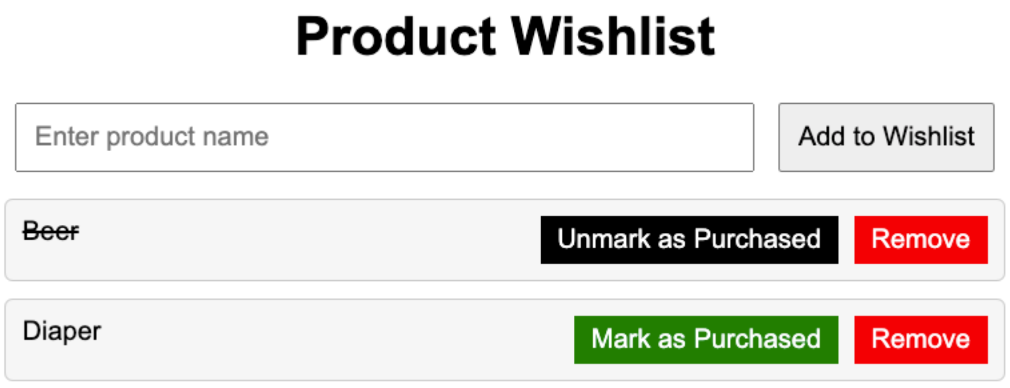
Product Wishlist

Enter product name | Add to Wishlist

Beer | Mark as Purchased | Remove

Diaper | Mark as Purchased | Remove

As a result, a new box containing **Diaper** and associated buttons has been **added**.

wishlist.html *(Subsequently, upon pressing **Mark as Purchased** button next to **Beer**)*


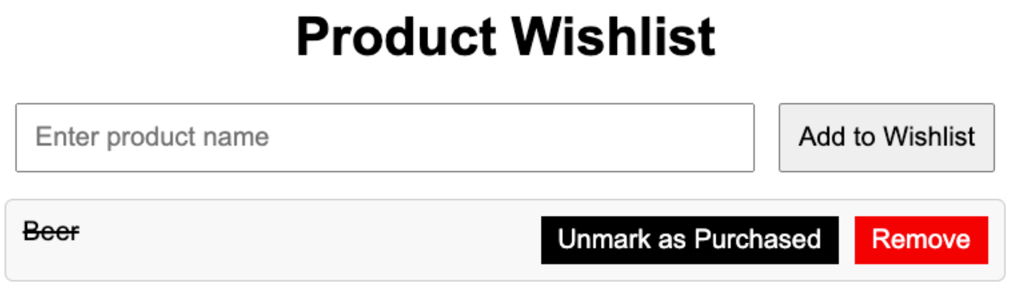
Product Wishlist

Enter product name | Add to Wishlist

~~Beer~~ | Unmark as Purchased | Remove

Diaper | Mark as Purchased | Remove

As a result, the text **Beer** text has a **strikethrough** applied to it.
Also, the "Mark as Purchased" button has become "Unmark as Purchased" with **black** background.

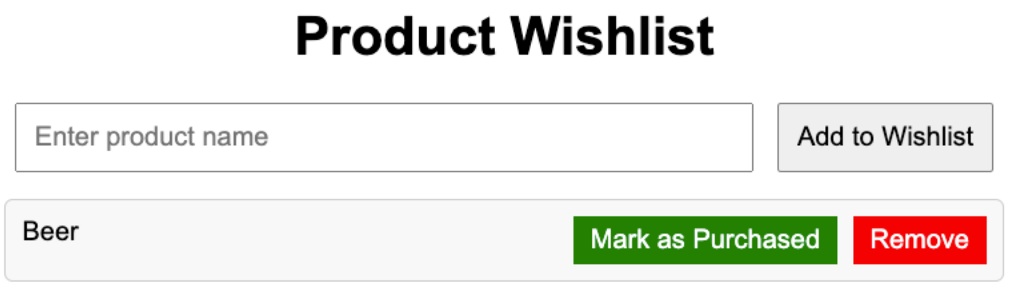
wishlist.html *(Subsequently, upon pressing **Remove** button next to **Diaper**)*


Product Wishlist

Enter product name | Add to Wishlist

~~Beer~~ | Unmark as Purchased | Remove

As a result, the entire box containing **Diaper** has been **removed**.

wishlist.html *(Subsequently, upon pressing **Unmark as Purchased** button next to **Beer**)*


As a result, the text **Beer** text has **strikethrough removed**.
Also, the “Unmark as Purchased” button has become “Mark as Purchased” with **green** background.

Q4: JavaScript Axios & API Interaction: Travel Packages [8 Marks]

Given resources in folder `Q4`

✏️ `travel.js`
🚫 `travel.css`
🚫 `travel.html`
🚫 `travel_api.php`

Scenario

You are tasked with building a **travel package selection tool** where users can choose from a list of travel packages. The HTML and CSS structure is already provided for you. The page fetches travel package details from a **local PHP API** using **Axios** when the user selects a valid package and clicks the **"Get Package"** button. The button is disabled until a valid package is selected.

Instructions

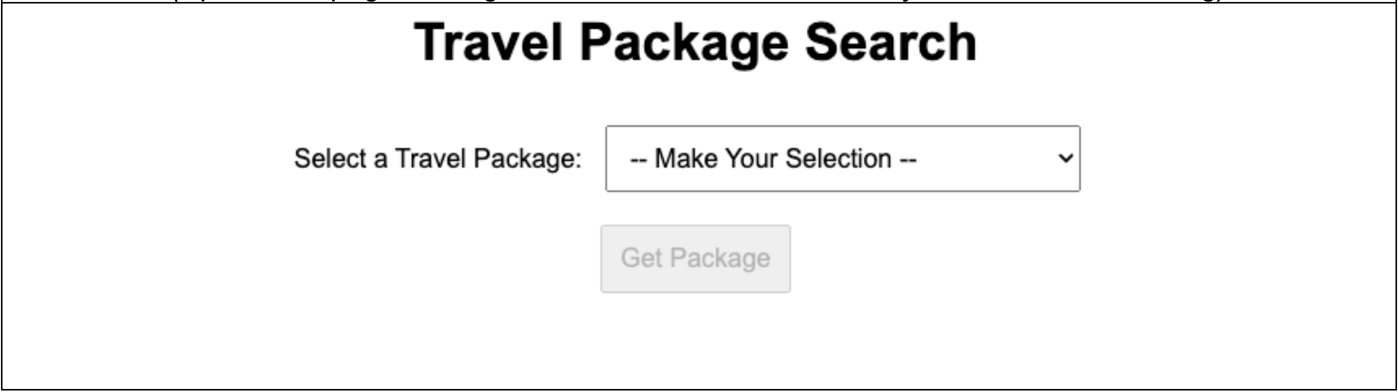
1. You are provided with the **HTML** and **CSS** structure, and **you must not modify them**.
2. Implement two JavaScript event listeners:
    a. One for the **dropdown selection change** to enable the **"Get Package"** button when a valid package is selected.
    b. One for the **"Get Package"** button to trigger an **Axios** request and display the travel package details.

Tasks

1. **Dropdown Event Listener (5 marks)**
    - Add an event listener to the **dropdown menu** to enable the **"Get Package"** button when a valid package is selected and hide the package details when a new dropdown menu selection is made.
2. **Button Event Listener (3 marks)**
    - Add an event listener to the **"Get Package"** button that sends an Axios request to the PHP API, fetches the data, and dynamically updates the package details on the page.
    - Use **Axios** to fetch data from the local PHP API, which will return travel package details based on the selected option.
    - Display the package details (name, duration, and description) after the data is fetched.

travel.html

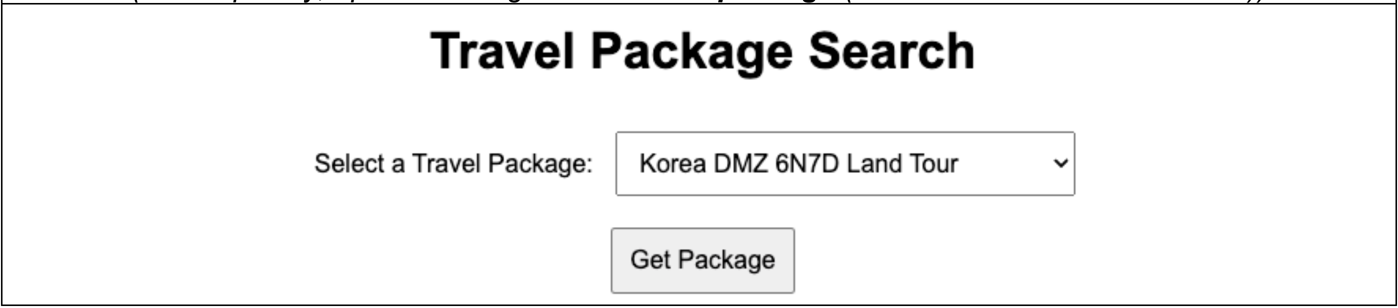
(Upon fresh page loading where the user hasn't done any action such as clicking)



By default, the **Get Package** button is **disabled** (until a **valid travel package is selected**).
Also, the section below the button that is designed to display travel package details is **hidden**.

travel.html

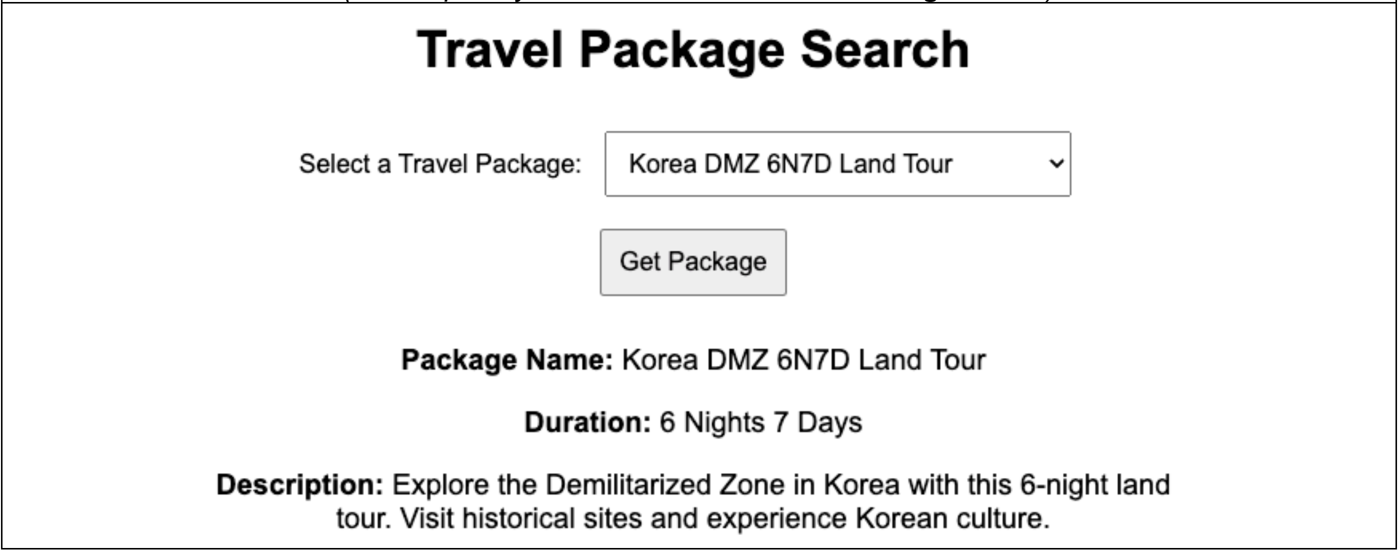
*(Subsequently, upon selecting a **valid travel package** ("Korea DMZ 6N7D Land Tour"))*



Since a **valid travel package** has been selected by the user, the **Get Package** button is now **enabled**.

travel.html

*(Subsequently, the user clicks on **Get Package** button)*



Package Name: Korea DMZ 6N7D Land Tour

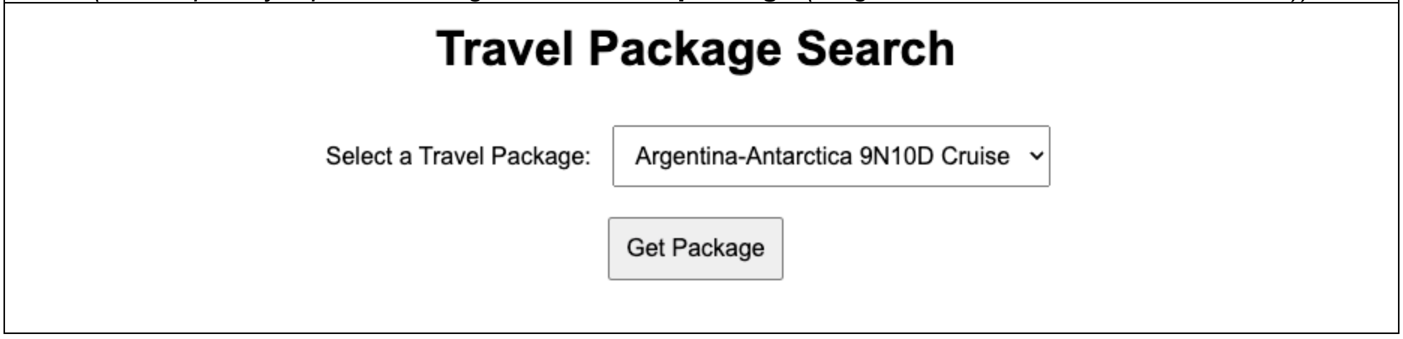
Duration: 6 Nights 7 Days

Description: Explore the Demilitarized Zone in Korea with this 6-night land tour. Visit historical sites and experience Korean culture.

As a result, the **package details** are **displayed/shown** to the user.

travel.html

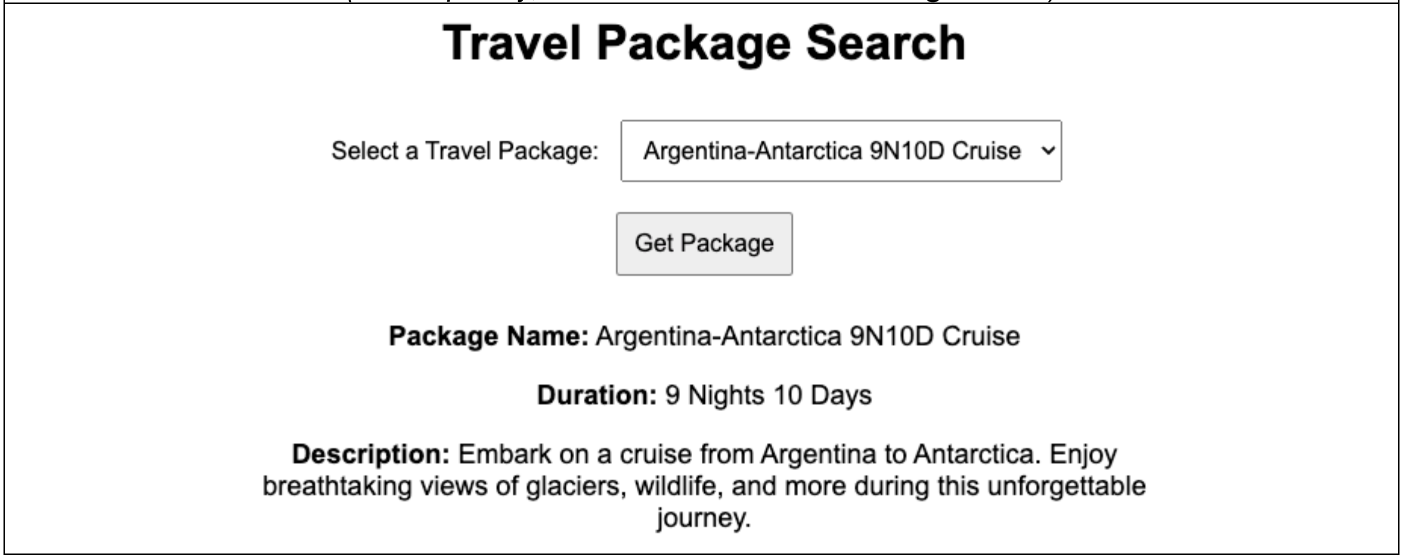
*(Subsequently, upon selecting a **valid travel package** ("Argentina-Antarctica 9N10D Cruise"))*



Since a **valid travel package** has been selected by the user, the **Get Package** button is **enabled**.
Also, the previously selected travel package's details disappeared and is not visible to the user anymore.

travel.html

*(Subsequently, the user clicks on **Get Package** button)*



Package Name: Argentina-Antarctica 9N10D Cruise

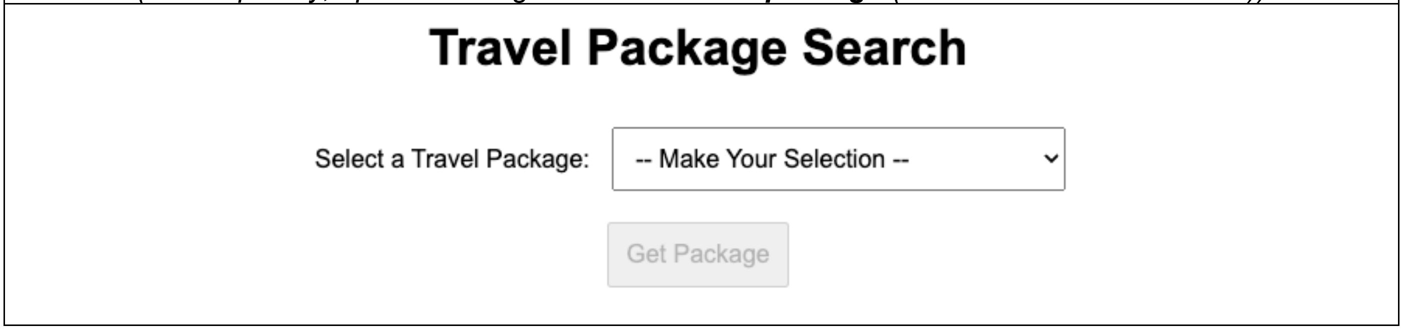
Duration: 9 Nights 10 Days

Description: Embark on a cruise from Argentina to Antarctica. Enjoy breathtaking views of glaciers, wildlife, and more during this unforgettable journey.

As a result, the **package details** are **displayed/shown** to the user.

travel.html

*(Subsequently, upon selecting an **invalid travel package** ("—Make Your Selection --"))*



Since the user selected an **invalid option**, the **Get Package** button is **disabled**.
Also, the section below the button that is designed to display travel package details is **hidden**.

End of Paper