IS216 Sample Large Lab Test (Set B) [40 marks]

General Instructions:

- This is a time-bound (2 hours), 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 5.2), JavaScript, Axios, and Vue.js (Version 3)** in your solutions unless the question specifies otherwise. Do not use any other third-party libraries (e.g. Angular, React, or others).

- 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 Jong Un" and email ID is kim.jongun.2020, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files

`<!--`
`Name:  KIM Jong Un`
`Email: kim.jongun.2021`
`-->`
	`/*`
	`Name:  KIM Jong Un`
	`Email: kim.jongun.2021`
	`*/`

- 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 11 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**
  - **XX November 2022 (Friday) XX:XX 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, **kim.jongun.2021.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/333446
  - Go to **Assignments → Large 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.

IMPORTANT

Inside **resources** folder, you will find **Bootstrap** files. **Please do NOT edit these files.**

➔ `Bootstrap\`
- ◆ 🚫`bootstrap.bundle.min.js`
- ◆ 🚫`bootstrap.min.css`

Q1. [CSS, Bootstrap, JavaScript] Grocery   [10 marks]

Given resources:
- ➔ `Q1\`
  - ◆ ✏️ `grocery.html`
  - ◆ ✏️ `grocery.js`

You are to **edit `grocery.html`** and **`grocery.js` only**. Do not create additional files.

IMPORTANT

You are free to define HTML elements and CSS style required 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.

- **`grocery.html`** must only contain HTML and CSS code. You must write **ALL** JavaScript code inside **`grocery.js`**.
- JavaScript code written inside **`grocery.html`** will **_NOT_** be considered for grading.

Part A: Complete `grocery.html`

Complete the implementation of **`grocery.html`** such that when rendered in a web browser, **`grocery.html`** displays the following:

Please take note of the following **user interface layout** guidelines:

The **main content** spans **80%** of the web browser **width**

Shopping List

What to Buy   Enter item name   Add

Calculate Total Amount
Calculate

The **main content** spans **80%** of the web browser **width**

15 pixels of space

20 pixels of space between Web Browser's **Top Border** and **<h4>**

30 pixels of space between Web Browser's **<h4>** and the content below

Shopping List

What to Buy   Enter item name   Add

Calculate Total Amount
Calculate

15 pixels of space

15 pixels of space

15 pixels of space

15 pixels of space

15 pixels of space

Shopping List

What to Buy   Enter item name   Add

Calculate Total Amount
Calculate

15 pixels of space

15 pixels of space

Part B: Complete `grocery.js`

Implement **`addItem()`** and **`processItems()`** such that the following **user scenario** is fulfilled.

grocery.html (before clicking Add button)
Shopping List
What to Buy · bread · Add · · Calculate Total Amount · Calculate

1. The user keys in string/text **bread** in the input text field.
2. The user **clicks** the **Add** button.

Grocery.html (after clicking Add button)
Shopping List
What to Buy · Enter item name · Add · · Calculate Total Amount · Calculate · ☐ bread

Item **bread** has been **added** with a **checkbox**.
Next, the user will add **one more item**.

grocery.html (before clicking Add button)
Shopping List
What to Buy · coffee · Add · ☐ bread · · Calculate Total Amount · Calculate

3. The user keys in string/text **coffee** in the input text field.
4. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)
Shopping List
What to Buy · Enter item name · Add · ☐ bread · ☐ coffee · · Calculate Total Amount · Calculate

Item **coffee** has been **added** with a **checkbox**.

grocery.html (before clicking Calculate button)

5. The user selects both **bread checkbox** and **coffee checkbox**.
6. The user **clicks** the **Calculate** button.

grocery.html (after clicking Calculate button)

Below the **Calculate** button, each item's name and its price are listed.

- Each item's **price** information can be found in the **shopList** array in **grocery.js**.

Furthermore, the **total cost** is calculated and displayed below the item list.

Part C: Complete `grocery.js`

Modify `addItem()` and `processItems()` such that the following **user scenario** is fulfilled.

grocery.html (before clicking Add button)
Shopping List
What to Buy · Enter item name · Add · Calculate Total Amount · Calculate

1. The user does NOT key in anything in the input text field (leaving it empty).
2. The user **clicks** the **Add** button.

grocery.html (after clicking on Add button)
Shopping List
What to Buy · Aiyo! Enter item name! · Add · Calculate Total Amount · Calculate

The input text field's **placeholder** value has been updated to **Aiyo! Enter item name!**
Next, the user will add **one item**.

grocery.html (before clicking Add button)
Shopping List
What to Buy · bread · Add · Calculate Total Amount · Calculate

3. The user keys in string/text **bread** in the input text field.
4. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)
Shopping List
What to Buy · Enter item name · Add · Calculate Total Amount · Calculate
☐ bread

Item **bread** has been **added** with a **checkbox** button.
Next, the user will add **one more item**.

grocery.html (before clicking Add button)
Shopping List What to Buy   coffee   Add   **Calculate Total Amount**   Calculate ☐ bread

5. The user keys in string/text **coffee** in the input text field.
6. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)
Shopping List What to Buy   Enter item name   Add   **Calculate Total Amount**   Calculate ☐ bread ☐ coffee

Item **coffee** has been **added** with a **checkbox** button.

grocery.html (before clicking Add button)
Shopping List What to Buy   kimchi   Add   **Calculate Total Amount**   Calculate ☐ bread ☐ coffee

7. The user keys in string/text **kimchi** in the input text field.
8. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)
Shopping List What to Buy   Sorry! Don't have it!   Add   **Calculate Total Amount**   Calculate ☐ bread ☐ coffee

Item **kimchi** is **_NOT_** available in this shop (refer to the variable **shopList** in **grocery.js** file).
The input text field's **placeholder** value has been updated to **Sorry! Don't have it!**

9. The user does NOT select any item **checkboxes**.
10. The user **clicks** the **Calculate** button.

grocery.html (after clicking Calculate button)
Shopping List What to Buy — Sorry! Don't have it! — Add ☐ bread ☐ coffee **Calculate Total Amount** Calculate You need to select items for calculation!

At the bottom of the webpage, an **alert** message is displayed with text **You need to select items for calculation!**

11. Again, the user does NOT select any item **checkboxes**.
12. The user **clicks** the **Calculate** button.

grocery.html (after clicking Calculate button for the 2nd time)
Shopping List What to Buy — Sorry! Don't have it! — Add ☐ bread ☐ coffee **Calculate Total Amount** Calculate You need to select items for calculation!

At the bottom of the webpage, an **alert** message is displayed with text **You need to select items for calculation!**

● If the user repeatedly **clicks** on the **Calculate** button without any items selected, **grocery.html** must display only **ONE (1) alert notification** at all times.

grocery.html (before clicking Calculate button)
Shopping List What to Buy — Sorry! Don't have it! — Add ☑ bread ☑ coffee **Calculate Total Amount** Calculate You need to select items for calculation!

13. Finally, the user selects both **bread checkbox** and **coffee checkbox**.
14. The user **clicks** the **Calculate** button.

grocery.html (after clicking Calculate button)

Shopping List

What to Buy	Sorry! Don't have it!	Add

☑ bread

☑ coffee

Calculate Total Amount

[Calculate]

bread - $1.60

coffee - $3.60

The total cost is : $5.20

Below the **Calculate** button, each item and its price are listed. Furthermore,
- The **total cost** is calculated and displayed, and;

The **alert notification** is removed from the web page.

Q2. [Bootstrap, JavaScript, Axios] Oscars   [10 marks]

Given resources:

➔ `Q2\`

- 🚫`krazyoscars/*` *(more folders & files)*
- 🚫`axios.js`
- ✏️ `oscars.html`
- ✏️ `oscars.js`

You are to **edit `oscars.html`** and **`oscars.js` only**. Do not create additional files.

IMPORTANT

Q2 ➔ **krazyoscars** contains **API** files.

- You do NOT need to see and understand the API source codes.

- You can view the **API documentation** at:

  > http://localhost/.../**Q2**/**krazyoscars**/**index.html**

- Go to **Q2** ➔ **krazyoscars** ➔ **config** ➔ **database.php**

```
6        private $host = "localhost";
7        private $db_name = "oscars";
8      ✔private $username = "root";
9      ✔private $password = ""; // MAMP "root", WAMP empty string
10     ✔private $port = 3306; // Check in PHPMyAdmin for port number
```
Please edit **Lines 8, 9, 10** as per your local computer's setup

- Go to **Q2** ➔ **krazyoscars** ➔ **db** ➔ **load.sql**

```
1    drop database if exists oscars;
2
3    create database oscars;
4
5    use oscars;
6
7    CREATE TABLE if not exists `winner` (
```
Use **PHPMyAdmin** (http://localhost/phpmyadmin/) or any **MySQL client** (e.g. MySQL WorkBench) to run all lines of SQL statements inside **load.sql**.

- Now, **KrazyOscars API** should be ready and you will be able to connect to and interact with it from your JavaScript code (**`oscars.js`**).

You are free to define HTML elements and CSS style required 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.

- **`oscars.html`** must only contain HTML and CSS code. You must write **ALL** JavaScript code inside **`oscars.js`**.
- JavaScript code written inside **`oscars.html`** will ***NOT*** be considered for grading.

Part A: Complete the top heading in `oscars.html`

Edit `oscars.html` such that when rendered in a web browser, `oscars.html` displays the following at the top of the page:

oscars.html
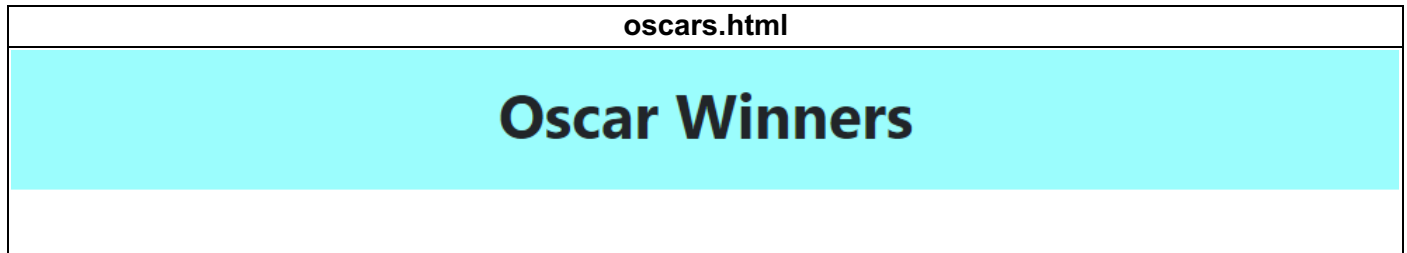
Please take note of the following **user interface** guidelines:

- The height of the **heading box** (in aqua color) is **100 pixels**.
- The **heading box** spans **100%** of the web browser's **width**
- There is no margin between the **heading box** and the top border of the web browser
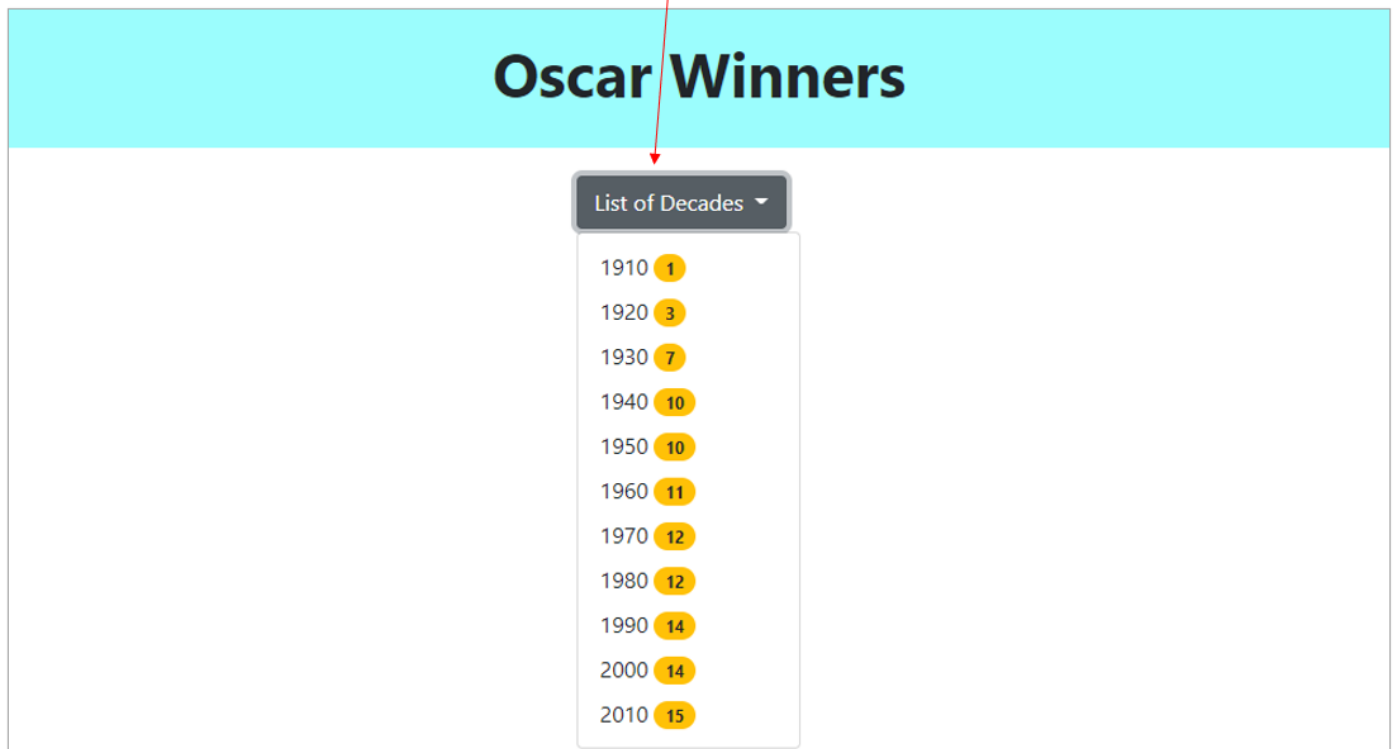
Part B: Complete the dropdown menu in `oscars.html`

Complete implementation of `oscars.html` such that it displays the following when the user clicks on the **"List of Decades"** *dropdown* menu:
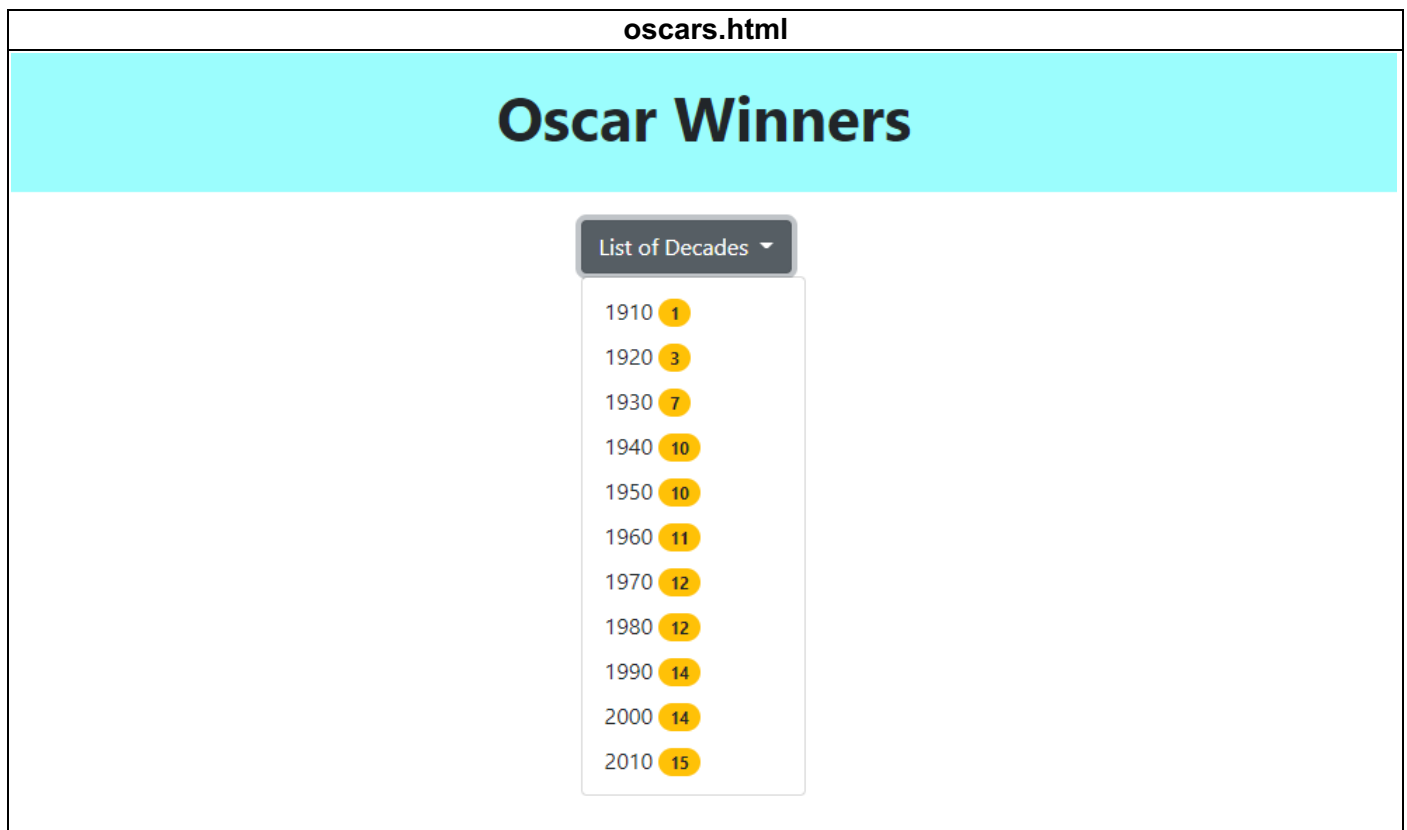
oscars.html

Please take note of the following **user interface layout** guidelines:

1. The **dropdown menu** lists different **decades** (1910, 1920… and so on) during which Oscar awards were given.
   a. For example, the **decade** of "1920" encompasses years **1920**, **1921**… all the way to **1929**.
2. The yellow circle on the right-hand side of each **decade** denotes the total number of **Oscar winners** in that **decade**.

Make appropriate changes in one or more **source file(s)** so that **oscars.html** displays the dropdown menu as shown above.

IMPORTANT
- o Do **NOT** hardcode the **decades** (e.g. 1910, 1920, etc.) in the dropdown menu.
  - ○ You must retrieve it *dynamically* by calling KrazyOscars API.

- o Do **NOT** hardcode the total number of Oscar winners in the dropdown menu.
  - ○ Your code must dynamically retrieve the values by calling KrazyOscars API.

- o Your code **MUST** use **relative URLs** for **API endpoints** and **images**.
  - ○ For example, an example of a **relative URL** (relative to the current file, e.g. **oscars.html**) is:
    - ■ <img src="krazyoscars/images/cher.jpg"> ***---> Use this!!!***
    - ■ <img src="http://localhost/krazyoscars/images/cher.jpg"> ***---> DO NOT USE THIS!!!***
  - ○ Failure to do so will attract a penalty of up to **20%** of your score for the entire **Question 4**.

Part C: Complete the main body in `oscars.html`

Complete implementation of `oscars.html` such that when rendered in a web browser, `oscars.html` displays the following:

oscars.html
(Web Browser Width 576 pixels)

Oscar Winners

List of Decades ▾

Emil Jannings

The Last Command (1928)
At the first Oscar ceremony, Jannings won two performances. In 'The Last Command,' Jannings plays a former Russian general who ends up as a Hollywood extra. And in 'The Way of All Flesh,' he takes on the role of a bank clerk who is separated from his family for over two decades.

Warner Baxter

In Old Arizona (1929)
Baxter won his Oscar for playing The Cisco Kid in the first Hollywood western made with sound.

Lionel Barrymore

A Free Soul (1931)
The patriarch of the famed acting dynasty won his performance as a defense attorney who must defend his daughter's ex-boyfriend who is accused of murdering a mobster.

Charles Laughton

The Private Life of Henry VII (1933)
Laughton won for his performance as the zaftig British monarch who goes through multiple wives.

… there's more content below…

oscars.html
(Web Browser Width 575 pixels)

Oscar Winners

List of Decades ▼

Emil Jannings

The Last Command (1928)

At the first Oscar ceremony, Jannings won two performances. In 'The Last Command,' Jannings plays a former Russian general who ends up as a Hollywood extra. And in 'The Way of All Flesh,' he takes on the role of a bank clerk who is separated from his family for over two decades.


Warner Baxter

In Old Arizona (1929)

Baxter won his Oscar for playing The Cisco Kid in the first Hollywood western made with sound.

… there's more content below…

1. When the web page is loaded for the first time, it must display **ALL** available/retrievable **Oscar winners** and display each **winner** in a **Bootstrap card**. There are **109 Oscar winners** in the API's database.

2. Note that each **Oscar winner** is to be rendered inside a **Bootstrap card**.

3. Each **card** shows the **details** of an **Oscar winner**. Inside each card, style the content (e.g. Oscar winner's name, movie title, year, and description) appropriately. Here is an example:
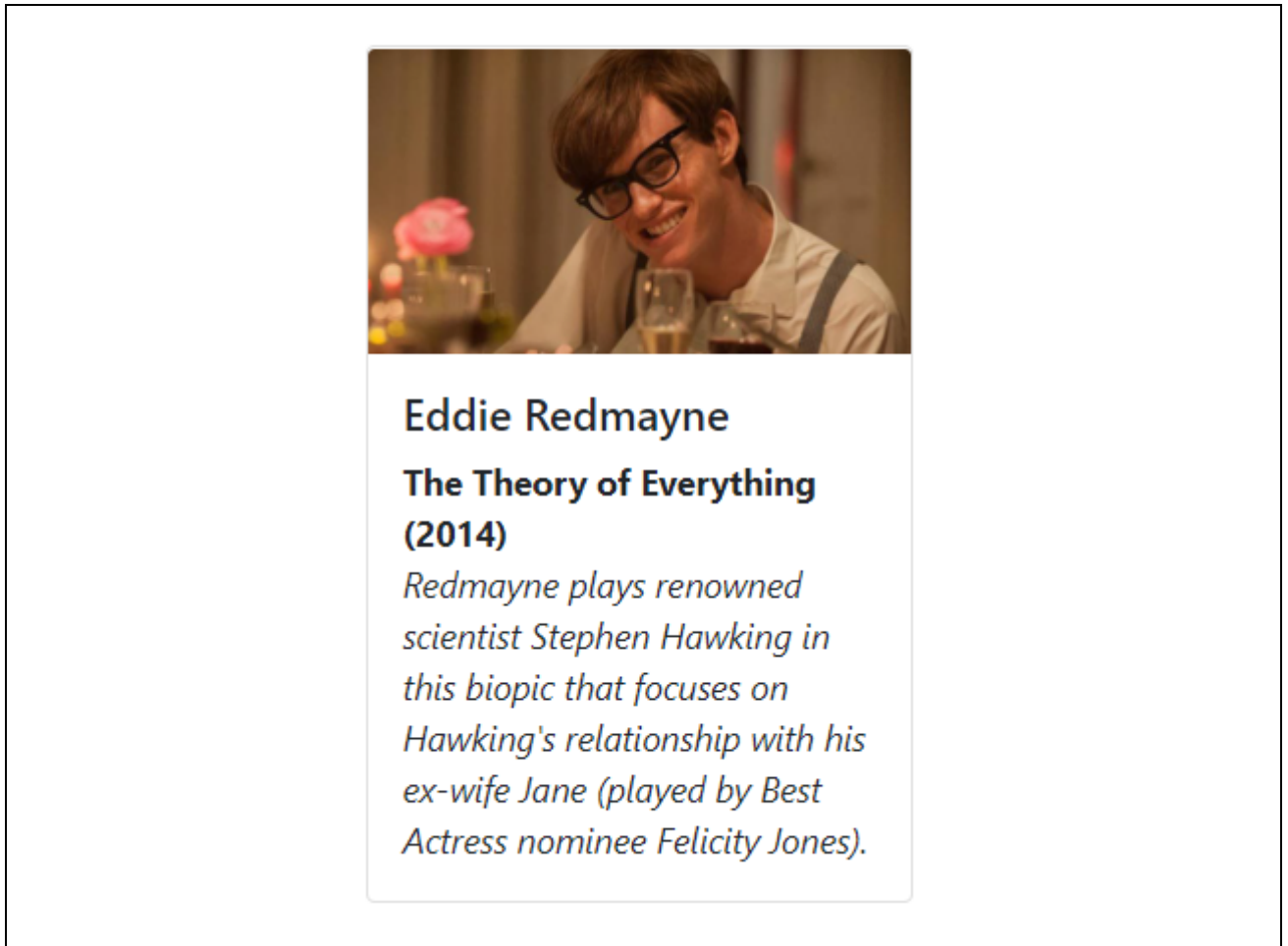
4. Make appropriate changes in one or more **source file(s)** so that `oscars.html` displays as described above.

IMPORTANT: Your code **MUST** use **relative URLs** for **API endpoints** and **images**.
   ○ Failure to do so will attract a penalty of up to **20%** of your score for the entire **Question 4**.

Part D: Complete the main body in `oscars.html`

When the user clicks on a **decade** in the **dropdown menu, oscars.html** must display only those **Oscar winners** who received the award in **that decade**. For example, below, the user clicks on **1920** (there are **3 Oscar winners** in that decade). The **red circle** is an *annotation used for illustration* - it is NOT part of the web page.

oscars.html must display as shown below. Only those **Oscar winners** who received the award in the **1920s** (decade) are displayed.

Make appropriate changes in one or more **source file(s)** so that `oscars.html` displays as described above.

IMPORTANT: Your code **MUST** use **relative URLs** for **API endpoints** and **images**.
   o   Failure to do so will attract a penalty of up to **20%** of your score for the entire **Question 4**.

Q3. [Vue.js] Sentiment   [10 marks]

Given resources:
- ➔ Q3\
  - ◆ 🚫 vue.global.js
  - ◆ ✏️ sentiment.html
  - ◆ ✏️ sentiment.js

You are to **edit sentiment.html** and **sentiment.js only**. Do not edit the other files. Do not create additional files.

IMPORTANT

You are free to define HTML elements and CSS style required 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.

- **sentiment.html** must only contain HTML and CSS code. You must write **ALL** JavaScript code inside **sentiment.js** (including Vue.js code).
- JavaScript code written inside **sentiment.html** will _**NOT**_ be considered for grading.

First, please *carefully inspect* sentiment.js.

- It creates a new **Vue** app and binds to an HTML element with id="app".

- The **Vue** app already has **data properties** declared.
  - Please NOTE that you (as a developer) should NOT manually edit data() section inside sentiment.js except *test cases* (which you can simply comment/uncomment).
  - Your HTML and JavaScript code *CAN* access these data properties (e.g. read, edit).

Part A: Complete computed property greeting

Complete **computed property greeting()** in **sentiment.js** such that when rendered in a web browser, **sentiment.html** displays the following:

sentiment.html
Hello Sunday

Header <h1> text
- a. Please do NOT hardcode the header text.
- b. The day text (e.g. Monday, Tuesday,…) must be *derived* from the **current date/time**.
  - i. HINT: JavaScript *DATE()*
  - ii. You can also use **const numeric_to_day** at the top of **sentiment.js**.
- c. HINT: Which Vue directive can you use to retrieve the information from an existing Vue app **data property**?

Part B: Complete `sentiment.html`

Edit `sentiment.html` such that when rendered in a web browser, the web page displays the following:

1. **Journal Entry**
    a. Please do NOT hardcode the *journal entry text* shown above.
    b. HINT: Which Vue directive can you use to retrieve the information from an existing Vue app **data property**?
    c. Later on, when the user keys in new journal entry text, your web application should be able to take this **new data** and update the corresponding Vue app **data property**.
2. **Table** *of sentiment words*
    a. Please do NOT hardcode the table data inside <tbody>…</tbody>.
    b. HINT: Which Vue directives can you use to retrieve the information from an existing Vue app **data property** "sentiment_word_counts"?
3. **Assessment**
    a. Please do NOT hardcode the <img> tag's *src*.
    b. HINT: Which Vue directive can you use to retrieve the information from an existing Vue app **data property**?

Part C: Complete `process_entry()`

Complete **`sentiment.html`** and **`sentiment.js`** such that when the user clicks on **How's My Sentiment Today?** button, your web application will update the **Table** *(on the right-hand side)*.

For example, see the above screenshot where the user wrote the following in the **textarea**:

> I am so, angry and sad.... and upset and depressed #)(*#@)(!! i feel terrible.
> but mala made me happy today. good food always makes me feel fine.

Next, user clicked on **How's My Sentiment Today?** button.

1. **`sentiment.html`** shows the total number of **positive words** and the total number of **negative words** found in the journal entry in the **Table** *(on the right-hand side)*.
   a. You can use the following declared at the top of **`sentiment.js`**:
      i. **`const sentiment_dictionary`**
      ii. **`const regex`**
2. Based on the positive/negative word counts, under **Assessment**, an image is displayed to reflect the overall sentiment.
   a. You can use the following declared at the top of **`sentiment.js`**:
      i. **`const image_sources`**

Another Test Case

Suppose the user keyed in the following journal entry:

i feel so FINE and happy. today is a terrific fabulous day !!!!

Next, user clicked on **How's My Sentiment Today?** button. `sentiment.html` shows the analysis results in the **Table**. This time, **Assessment** shows a green smiley image.

Q4. [Vue Components] format_obj [10 marks]

Given resources:

➔ `Q4\`
  ◆ 🚫`format_obj.html`
  ◆ ✏️ `format_obj.js`
  ◆ 🚫`vue.global.js`

You are to **edit `format_obj.js` only**. Do not edit the other files.

Look through the codes in the given file `format_obj.html`.

Edit the given file `format_obj.js` to implement the Vue component `format_obj` according to the following rules:

1. If `data` is a simple type (e.g. String, numbers, Boolean, etc.), display `data` as an HTML text.

2. If `data` is an **object**, display its properties as an HTML **unordered list**. Display each property in the following format where `<name>` and `<value>` are property name and property value respectively.

   `<name> : <value>`

3. If `data` is an **array**, display its elements as an HTML **unordered list**. Display each array element in the format where `<index>` and `<value>` are the index and value of the array element respectively.

   `[<index>] <value>`

4. Note that property values and array elements (`<value>` in points 2 and 3) can be another object, array or simple type. Apply the rules 1 to 3 recursively to display `<value>`.

Refer to the expected output of `format_obj.html` in browser for examples of the above rules:

Expected output	Corresponding `data`

Simple object (displayed as unordered list)
 	
• name : Simple object
• age : 18
	`{`
	`    name: 'Simple object',`
	`    age: 18`
	`}`

Simple array (displayed as ordered list) • [0] 1.5 • [1] horse • [2] false	`[1.5, "horse", false]`
Literal value (displayed as plain text) true	`true`

Complex object 1	```{ name: 'Parent', age: 32, children: [{ name: 'Child-1', age: 12}, { name: 'Child-2', age: 4},]}```

• name : Parent
• age : 32
• children :
    • [0]
        • name : Child-1
        • age : 12
    • [1]
        • name : Child-2
        • age : 4	

Complex object 1

- name : Parent
- age : 32
- children :
  - [0]
    - name : Child-1
    - age : 12
  - [1]
    - name : Child-2
    - age : 4

```
{
    name: 'Parent',
    age: 32,
    children: [
      { name: 'Child-1', age: 12},
      { name: 'Child-2', age: 4},
    ]
}
```

Complex object 2

- name : Very complex object
- arr :
  - [0] true
  - [1] a
  - [2] -5.3
  - [3]
    - fruit : apple
    - price : 2.45
- flag : true
- another_obj :
  - name : Child object
  - level : 2

```
{
    name: 'Very complex object',
    arr: [
        true,
        "a",
        -5.3,
        {
            fruit: 'apple',
            price: 2.45
        }
    ],
    flag: true,
    another_obj: {
        name: 'Child object',
        level: 2
    }
}
```

Complex array

- [0] Complex array
- [1] 31
- [2]
  - fruit : apple
  - color : red
- [3]
  - [0] nested array
  - [1]
    - name : Yet another object

```
[
    "Complex array",
    31,
    {
        fruit: 'apple',
        color: "red"
    },
    [
        'nested array',
        {name: 'Yet another object'}
    ]
]
```

Empty object

```
{}
```

Empty array

```
[]
```

Do **NOT** hardcode. The values in `format_obj.html` will be changed during grading.