Homework 01 – HTML, CSS & JavaScript

Out of 65 Marks

DUE: 2022-03-27 before 23:59

IMPORTANT NOTES:

- This is an individual assignment.
- Homework assignments are based on assessment objectives. If an objective has been achieved a mark will be allocated.

INSTRUCTIONS:

- In this assignment, you will be given high-level requirements and can implement them in a context you see fit.
- **Source Code:** Zip your source code files together and name it **uXXXXXXXX_HW01.zip**, where the XXXXXXXX is your student number, e.g., u12345678_HW01.zip.
- **Video Demo:** DO NOT Zip your Video Demo. Name the video demo **uXXXXXXXX_HW01.mp4**, where the XXXXXXXX is your student number, e.g., u12345678_HW01.mp4.

NOTE ON DEMO SOFTWARE

- Please change the compression ratio and Frames Per Second (FPS) of the desktop recording software that you use to reduce the file size of your demo. These details have been noted at the top of the homework submission page.
- Please use desktop recording software as suggested on the homework submission page.
- DO NOT use your phone to record your demo as it is extremely difficult to follow what is going on in your demo and it creates unnecessarily large files making the video demo upload problematic. We suggested desktop recording software as it simply streamlines the process and creates smaller files that are easier to upload.

SUBMISSION DEADLINE: 2022-03-27 before 23:59

- There shall be no extensions to the deadline.
- If homework submissions are uploaded too late then upload errors WILL happen.
- Do not wait to the last minute to complete the assignment.
- Start working on the assignment as soon as it is posted.
- Verify the completeness of your upload.
- There are multiple upload opportunities enabled if your upload is incomplete.
- Incomplete uploads will be considered unsubmitted work.
- E-mail submissions WILL NOT be accepted.
- Late submissions WILL NOT be accepted.
- **NO EXCEPTIONS WILL BE MADE FOR ANYONE.**

IMPORTANT

- We download your program source code to make sure that you do not upload empty programs.
- In the past we have encountered people demoing other people's programs. That is considered academic dishonesty and there are severe consequences if you are found to have committed academic dishonestly. Please refer to the student code of conduct regarding the consequences of academic dishonestly.

1. TOOLS, SKILLS, KNOWLEDGE REQUIRED:

- In order to complete the Homework 1 assessment, you will have to apply your knowledge of JavaScript 1, HTML, and CSS.

2. USE CASE:

- For this assignment, you will be creating a simple time management system. You will need to program three use cases that will make up your time management system. The three use cases are Module, Task, and To-do. For these use cases you will be required to capture the following:
    - Module
        - Module Code
        - Module Lecturer
        - Unique Identifier
    - Task
        - Task Name
        - Task Date
        - Unique Identifier
    - To Do
        - Module Code
        - Task Name
        - Task Date
        - Description
        - Unique Identifier

3. STANDARD REQUIREMENTS:

The following is a list of mandatory requirements.
Your page should look like the following screenshot (*keep in mind that a lot of the CSS style choices are up to you and the requirements for such will be stipulated later*).

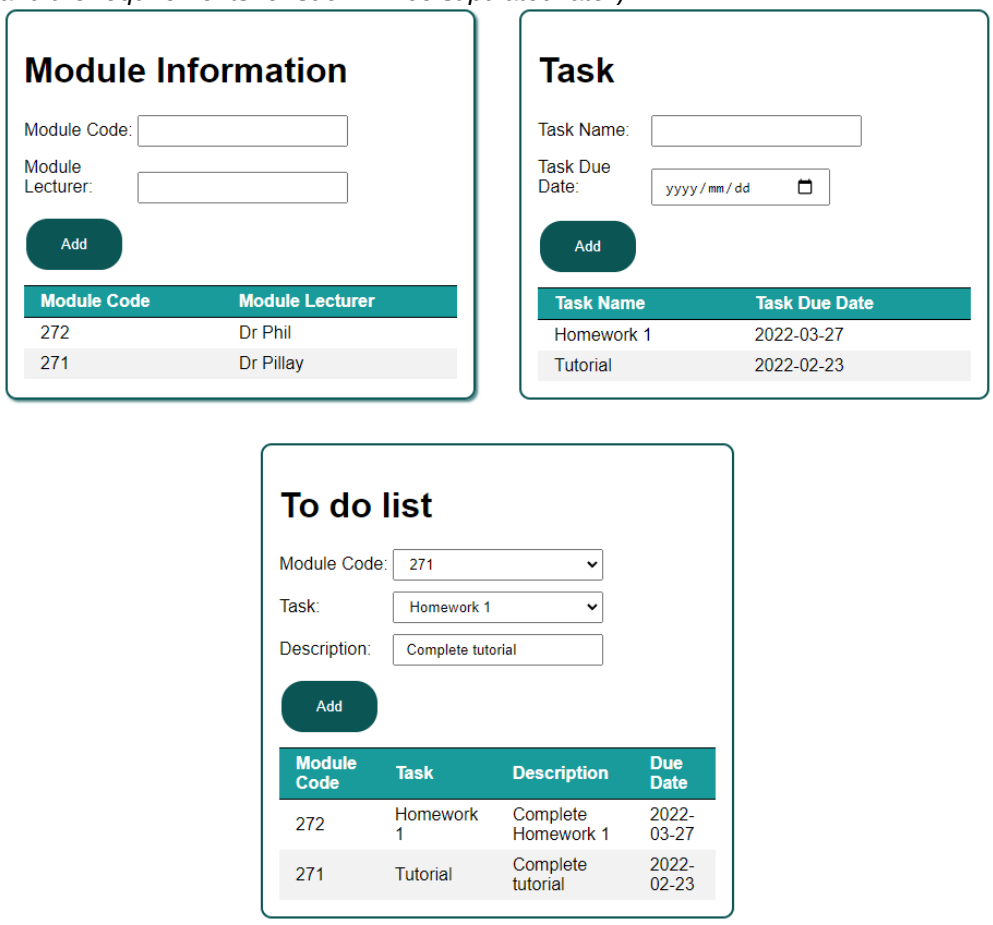

Figure 1 - General Page Layout

- Each use case consists of a form that will allow you to enter and capture the data provided and a table that allows you to display the captured data.
- Module
    - HTML Requirements
        - Label and text input for Module Code
        - Label and text input for Module Lecturer
        - Form submit button to add information to the table.
        - Table with the columns “Module Code” and “Module Lecturer”
    - JavaScript Requirements
        - When the user has entered a module code and module lecturer into the respective fields and clicks the add/submit button, those details must be stored in a JavaScript array of objects where the objects have the fields mentioned in the Use Case section of this document. The unique identifier should be a timestamp created when the user clicks the button.
        - When there is a change in the Module array (module added), the module select list in the To-do section must be updated (added).

- Tasks
    - HTML Requirements
        - Label and text input for Task Name
        - Label and date picker for Task Due Date
        - Form submit button to add information to the table.
        - Table with the columns “Task Name” and “Task Due Date”
    - JavaScript Requirements
        - When the user has entered a Task Name and Task Due Date into the respective fields and clicks the add/submit button, those details must be stored in a JavaScript array of objects where the objects have the fields mentioned in the Use Case section of this document. The unique identifier should be a timestamp created when the user clicks the button.
        - When there is a change in the Task array (task added), the Task select list in the To-do section must be updated (added).
- To-do
    - HTML Requirements
        - Label and drop down for Module Code
        - Label and drop down for Task
        - Label and text input for To-do description.
        - Form submit button to add information to the table.
        - Table with the columns “Module Code”, “Task Name”, “Description”, and “Task Due Date”
    - JavaScript Requirements
        - When the user has selected a Module Code and Task as well as inserted a description and clicks the add/submit button, those details must be stored in a JavaScript array of objects where the objects have the fields mentioned in the Use Case section of this document. The unique identifier should be a timestamp created when the user clicks the button.
        - When the user clicks on the add/submit button, the due date of the task must be retrieved from the task making use of the unique identifier.
        - When the user clicks on a **row** in the table, that row must be deleted and the corresponding entry in the array needs to be deleted too (Hint: make use of a “data” attribute for the row element to keep track of the unique identifier for the entry).
- CSS Requirements (**Check page 4 for examples**)
    - Change the font used on the page.
    - The cards should be positioned as shown in the example image i.e., the Module Information and Task card should be next to one another, and the To-Do List card should be underneath the Module Information and Task card. The To-Do List card should also be centred on the screen.
    - Change the colour of all the borders, backgrounds, etc. The background of the main page should be done by making use of linear gradient.
    - When you hover over any of the cards a box shadow should be shown.
    - You can style button however you want; however, it cannot be the default button style (change in background colour, text colour, button size, etc) and the cursor should be changed to a pointer when you hover over the button.
    - You can style the tables however you want; however, it cannot be the default table styles (change in colours, header background, cell padding, etc).
    - When you hover over a table row in the To Do List card the background and font colour should change to indicate the row will be deleted on click. (Not an animation but the cursor style should change to a pointer when you hover over a row). See the screenshot below for an example of the row when hovered over.

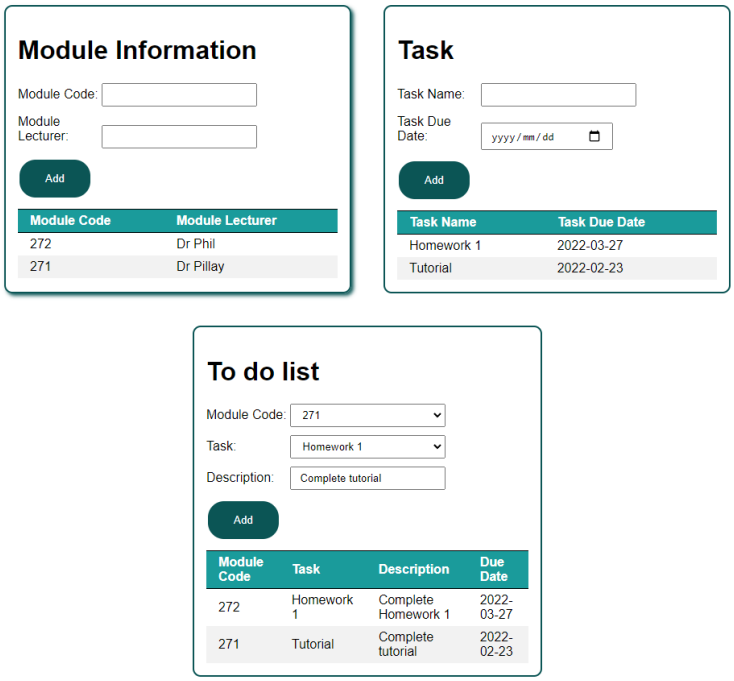

Figure 2 - Display of card shadow on hover (Module Information card has shadow)

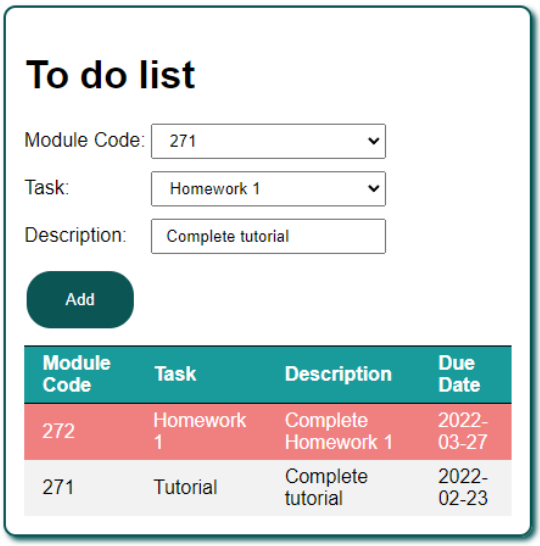

Figure 3 - Display styling when hovering on a table row

Example of what a JavaScript array of objects will look like when console logged:

```
▼(2) [{…}, {…}]
  ▶0: {id: 1645441336592, code: '272', lecturer: 'Dr Phil'}
  ▼1:
     code: "271"
     id: 1645441341377
     lecturer: "Dr Pillay"
    ▶[[Prototype]]: Object
   length: 2
  ▶[[Prototype]]: Array(0)
```

4. RESEARCH / READING REQUIREMENTS

The following is required reading and/or research.

- JavaScript arrays of objects
- CSS animations and linear gradient.
- HTML data attributes
- JavaScript Date object
- Passing HTML object in HTML function call

Homework 01 – HTML, CSS, & JavaScript

Out of 65 Marks

DUE: 2022-03-27 before 23:59

VIDEO INSTRUCTIONS:

- Make sure that everything is running when you start recording the video. The video should not be longer than **15 minutes** showing the items in the Checklist.
- **Simply scrolling through the program will be considered not presenting a demo which would equal to you immediately losing 50% of your mark. You need to explain your reasoning.** It is part of what is known as "reflective learning" and helps you solve problems when you encounter them during coding. It is extremely useful. Do not neglect your own learning process by skipping out on this important step
- When showing something in the Checklist, show us your code and explain why you did it in that way. If you prefer to demo everything first then explain code, you can do it that way, but we need an explanation of why you coded it that way and not just functionality. The explanation does not need to be too in-depth.
- If something did not work in your code, in the video explain to us what you wanted to do and what you wanted to achieve with your approach. **We will not mark your code that you show us unless you explain this**.

IMPORTANT NOTES:

- Both the Video Demo and Homework Source Code should be submitted in the correct upload area. **If the Video demo or the Homework Source Code is missing, you will get a zero.**
- If files are upload to the wrong upload area, we will not go and look for the upload. Uploads should be submitted correctly as indicated online. Incorrect uploads will lead to a zero being allocated.
- If you are caught for plagiarism, we will give you 0 and you will be reported for plagiarism immediately. There are no more warnings this year. We will audit historical assignments throughout the semester. You are here to learn, the more you know, the better for you.

Checklist Use the checklist as a script that would allow you to sequence your demonstration.	**MAX**
1. Requirement 1 – Functionality • Start by showing the functionality of the program. • Run the program and then go though one complete action. **Demonstration time allocation = 02 minutes**	**10**
2. Requirement 2 – Module Use Case • Inputs with correct types (**3x2:** 1 mark for input, 1 mark for correct type) • Input button has onclick event attached (**1** mark) • Object is correctly added to the Module array (**3**: 1 mark for each object field correctly added) • Table is updated when the array changes (**4:** 3 marks for correct row data; table must always reflect the array data, 1 mark for add) **Demonstration time allocation = 03 minutes**	**14**
3. Requirement 3 – Task Use Case • Inputs with correct types (**3x2:** 1 mark for input, 1 mark for correct type) • Input button has onclick event attached (**1** mark) • Object is correctly added to the Module array (**3**: 1 mark for each object field correctly added) • Table is updated when the array changes (**4:** 3 marks for correct row data; table must always reflect the array data, 1 mark for add) **Demonstration time allocation = 03 minutes**	**14**

Checklist Use the checklist as a script that would allow you to sequence your demonstration.	**MAX**
4. Requirement 4 – To-do Use Case • Select lists and input and (**1x2+2x2**: 1 mark for each select list, 1 mark input and 1 mark for correct input type) • Inputs with correct types (**1x2:** 1 mark for input, 1 mark for correct type) • Select Lists are correctly updated when there is a change in the respective lists (**3x2**: 1 mark for adding option to select list, 1 mark option text and 1 mark for option value; the module/task **must** be identified using their unique identifier) • Table is updated when the array changes (**4:** 3 marks for correct row data; table must always reflect the array data, 1 mark for add) • Object is correctly removed from array when table row is clicked (**3:** 1 mark for row onclick event, 2 marks for finding a correct object to remove) **Demonstration time allocation = 04 minutes**	**19**
5. Requirement 4 – CSS Styling • Alignment of cards (**2**: 1 mark for aligning the two top cards next to each other. 1 mark for aligning the third card in the centre below the 2 cards.) • Colour (**2**:1 mark for the colour of borders, 1 mark for the colour of the background using linear gradient) • Button Styling (**3**: 1 mark for each table CSS property changes **(Max 2)**, 1 mark for cursor change) • Table Styling (**2**: 1 mark for each table CSS property changes) • Box shadow (**2**) • Card Hover Animation (**2**: 2 marks for box-shadow shown) • Table Row Hover Animation (**5**: 2 marks for background colour change, 2 marks for font colour change, 1 mark for cursor change) **Demonstration time allocation = 03 minutes**	**18**
TOTAL MARK ALLOCATION **Total Demonstration time allocation = 15 minutes**	**65**