**MSC – CC – Assignment 01**

# UI

* General UI Elements:

-A theme is maintained throughout the app by using consistent colour scheme.

-The page background is a light orange shade so the darker text stands out and is more readable.

- The navbar on each page is a black bar to separate it from the body.

- Accent colour is dark orange. It is used when hovering over navbar buttons, and is the same colour used for buttons.

* addgpu.html

-This page contains the UI to add a GPU into the database.

-The labels and corresponding input elements are displayed in a center-aligned HTML table so that they appear in line vertically. The left column (that contains the input labels) is aligned to the right and the right column (that contains the input elements) is aligned to the left so that the two are close together and look associated.

-If a table was not used, the position of the input elements would depend on the length of the labels.

-The table itself, however, does not have any borders so the UI looks neatly aligned and minimal without any lines running around the form.

- A jinja variable carries an error message. It is set to null at first. If a user tries to add a GPU with an existing name, the jinja value is populated with an error message.

-The error message vibrates to attract the attention of the user.

* displayeach.html

-This page displays the name of the selected GPU and the values of its fields.

A table with two columns is used.

-The column on the left contains the names of the GPU features.

-They are right aligned so that there is no gap between the feature and its corresponding value.

-The column on the right contains the values of the corresponding features.

-The name of the selected GPU is displayed in bold above the table to make sure it is visible and highlighted.

* displaygpu.html

-This page acts as the homepage of the whole application. It displays the GPU names, along with an edit button.

-The GPU name itself is a hyperlink and be clicked to display the features supported by that GPU.

-Again, a centre-aligned table is used to neatly display this information.

* editgpu.html

-The UI for this page is similar to the UI for the add page.

-It uses the same center-aligned HTML table with the left column right aligned and the right column left-aligned.

-One difference is that the name of the GPU selected for editing is displayed in bold at the top instead of taking it as an input from the user.

* query.html

-This page contains the UI required to query GPU according to required features.

-It displays 2 columns, one for GPU features, and the other containing checkboxes.

-The Search button at the bottom is within the table so that it is the same size as the column above it.

# Methods and Modules

* **addgpu.py**

1. get(self)

A get request to this class is used to populate the logout, featurelist, and error message for the addgpu.html page.

1. post(self)

-A post request to this method is used to respond to calls from two buttons: -Add and Cancel. If the add button was pressed, the datastore is checked for an existing entity. If it doesn’t exist, a new entity is created with the id set to the gpu name and is added to the datastore.

-If the cancel button is pressed, the user is redirected back to the displaygpu page.

* **displayeach.py**

1. get(self)

-This file is called when a user clicks on a GPU name out of the list of all GPUs to view that GPU’s details.

-This method extracts the GPU from the datastore using the GPU name that the user clicked on, and returns it using a jinja template.

- It also returns the logout link and the list of GPU features.

* **displaygpu.py**

1. get(self)

* The get request is used to populate the webpage with names of all GPU’s in the datastore.
* A query is run to retrieve all datastore objects of Kind ‘Features’ (i.e. all the gpu’s) and returned via jinja.
* It also sends the logout link and default sort order (special feature) via jinja

1. post(self)

* The post method is solely to serve the special feature i.e. toggling the sort order of GPU names between ascending and descending.
* The post request contains the current sort order.
* The function then toggles the sort order and reloads the HTML page with the new sort order.
* **editgpu.py**

1. post(self)

- This method edits GPU information.

- Two buttons point to this page: EDIT and SUBMIT

- The edit button is used to get the user to the edit page. It sends the id of the GPU to the edited along with the request.

-If the edit button was clicked, the post() method queries the datastore for the data of the selected GPU and sends back the GPU object as the response via jinja.

- It also sends back the logout link and list of GPU properties.

- If the submit button was clicked, the post() method accepts the changed values of the GPU properties. It gets the GPU object from the datastore, updates its values, and saves it back in the datastore before redirecting the user to the homepage.

* **gpu\_features.py**

-This file represents the ndb Model and is named “Features”

- It has 6 ndb String properties, one for each GPU Feature.

* **login.py**

**-**This is the landing page of the application.

**-**If a user exists, it redirects to the displaygpu page.

- If a user is not logged in, it displays the login page for the user to sign in.

-It also contains the URL definitions for the application at the end.

* **query.py**

1. **get(self):**

This method merely populates the query html page with the logout link, list of GPU features, and flags the ‘notempty’ variable with “wait” indicating the HTML page not to display the heading “RESULTS” because the user has not run a query yet.

1. **post(self):**

* This method filters all GPUs in the datastore and returns only the ones that satisfy all the fields the user has chosen.
* It starts by generating a list containing all the GPU entities in the datastore.
* It then checks if the checkbox for a GPU feature is checked. If it is, it filters out all the entities that do not contain that feature and keeps the ones that do.
* It then returns the resultant list to the HTML page to be displayed to the user.

## Special Feature : toggle GPU sort order

The special feature is to “**toggle the sort order of the list of GPU names between ascending and descending order”.**

1. **UI :**

-The UI for this feature consists of a small arrow in the table header of the table that displays all GPU names.

- The arrow points upwards to indicate the ascending order and downwards to indicate the descending order.

- The background colour of the arrow images changes according to sort order for better readability.

-The icons were created in Ms-Paint.

1. **Implementation:**

-The get(self) method of the displaygpu python file sets the default sort order as ascending, so that the appropriate icon is displayed.

-It also queries the datastore accordingly and returns the GPU names sorted in ascending order to the displayed to the html page.

- When the user clicks on the arrow icon to change the sort order, the post(self) method of the displaygpu file is called.

- This method gets the current sort order and toggles it. It queries the datastore and orders values in the reverse sort order.

-These new values are passed as a response to the HTML.

-The appropriate icon and reverse sorted list is displayed.