MSc. Big Data Analytics & Management

Cloud Computing - [MSCBD-CC]

Assignment 01 - Documentation

MERT ALTINTAS - 2982775

1. Project Overview

In this assignment, the given task was to build a GPU database using the Google Cloud Platform and its components. The project's back-end was realized by Python programming language and Google Cloud's NoSQL database (ndb) was used in order to store the GPU' features such as its name, manufacture, issued date, etc. On the other hand, the front-end tasks were performed by using HTML, CSS, Bootstrap, and JavaScript. The version control was made via GitHub.

In this paper, firstly, the classes and their methods will be explained. After that section, the detailed information about the data structure and the models are going to be presented. Finally, the methodology behind the UI will be described.

2. Classes & Methods

The functionality of the application is basically realized with 5 Python classes and their methods. The classes used in the project is as follows:

- **MainPage Class**

  This class is mainly responsible from rendering the landing page, user login process and GPU Add function. It has two methods which are get and post.

  Basically, get method sends information to the Google Cloud Platform. MainPage class firstly sets up the 'Context-Type' as HTML and controls if there is any user has already signed in. According to the return value, it directs the user to the landing page by creating a login url from where user can login, or to the main page where user can add a new GPU and log out from the system.

  The post method controls the Add button. When user wants to add a new GPU, the GPU form should be filled and Add button must be pressed to submit the form. Once the button is pressed, the post method starts to run and firstly, it takes the name of the GPU from the form which is used as the key value in the datastore. If the GPU already exists it renders the main page with an error message and without adding the GPU. In the other case, it takes the other values from the form and creates GPU instance with these features by setting the key value as its name. Finally, the method adds this instance to the datastore and redirects the user to the main page.

- **GPUList Class**

GPUList class is responsible from listing the existing GPUs in the datastore and filtering them according to user's input. Like the other classes, it has two methods: get and post.

The get method mainly does the rendering operation of the GPU List page. After setting the 'Context-Type' and controlling the user authentication, the method fetches all the GPU's from the datastore and passes them to the page template. By rendering the template, the GPU List page becomes available for the user.

On the GPU List page, there is also a filtering option which gives the user the permission to search GPUs according to their needs by selecting boolean features. Hence, the post method executes the GPU filtering in the datastore. When the user presses the 'Search' button, the post method gets all the values as a list from the related form and it queries the database by applying the filters. When the querying ends, it fetches the filtered GPUs and sends them to the template. Filtered GPUs are shown in the GPU List page.

- **GPUDetail Class**

GPUDetail class's task is to show selected GPU's features to the user. It has only get method.

In this class, the get method sends the relevant information to the HTML page same as the other get methods. To fetch the selected GPU from the datastore, the key ie. the GPU name should be gathered. Thus, the selected GPU name is passed from the GPU List page to the get method via the URL. So, after the usual authentication control, the method gets this information from the URL and fetches the relevant GPU from the datastore by using its key. Finally, it passes the GPU to the template and renders the page.

- **GPUEdit Class**

GPUEdit class is responsible from editing selected GPU's information. It has get and post methods.

The get method firstly controls the user authentication as usual. The main task of this method is to get selected GPU's name from the URL and fetch the correspondent GPU from the datastore. After gathering the GPU from the datastore, it passes the values to the template and renders the relevant page.

Secondly, the post method controls the buttons on the page. If the user clicks the 'Apply' button, the method gets the values from the HTML form, updates the GPU and puts the updated GPU back to the datastore. At the end of the process, it redirects the user to the GPU Detail page of the relevant GPU. On the other hand, if the 'Cancel' button is pressed, the method redirects the user to the GPU List page without doing anything.

- **GPUCompare Class**

GPUCompare class is responsible from comparing two GPU's and rendering GPU compare page. It consists of only the get class.

The get class does the user authentication and gets the keys ie. names of the selected GPUs from the form. It also controls if the user selected two GPUs. Finally, the method fetches the GPUs from the datastore and passes them to the template.

3. Models

Only one model is used during the realization of the application which is GPU model. Basically, this model represents the structure of the every GPU in the system. It has 9 properties. 'Name' and 'Manufacturer' are set as StringProperty. 'dateIssued' feature is DateProperty which lets us represent the dates in 'YYYY-MM-DD' format. The other 6 features such as geometryShader, tesselationShader, shaderInt16, etc. are set as BooleanProperty because we want to check if they are supported by the GPU or not.

4. UI Design

The UI is designed to present a user-friendly environment to the user in order to facilitate the usage of the application. During the development of the interface, CSS, Bootstrap, and JavaScript are used. Thanks to Bootstrap, pages become responsive which means that they change their form automatically according to the window size.

The layout of each page is really simple. On each page, there is a navigation bar at the top that consists of the application name, menu, and Log In/Out button. The middle part is the body where you can find specific content related with that page. In addition, at the bottom, there is a footer in which you can find the credentials and the navigation links.

The main color scheme for the application is chosen as orange and white. According to the design perspective, shades of orange gives to the user the sense of friendliness and it is a great color to attract attention to the important points. On the other hand, it is not sharp as red so it doesn't strain users' eyes. Plus, white as the body background and font color is the best match for the shades of orange. It emphasizes the content and softens the view.

The tables, forms, and buttons are designed proportionally so that it presents a proper view to the user. No unnecessary information is used. On the other hand, complexity is avoided while designing the pages. In addition, all the platforms such as Internet Explorer, Google Chrome, Safari, etc. are considered while placing each element on the pages. That's because JavaScript is used for creating a calendar in order to pick the issue date of the GPU properly. (Safari doesn't support the input type 'date')