BIG DATA MANAGEMENT AND ANALYTICS

ARAVINDAN SRINIVASAN 14/03/2019
2981707
aravindan.srinivasan@student.griffith.ie

CLOUD COMPUTING ASSIGNMENT 1

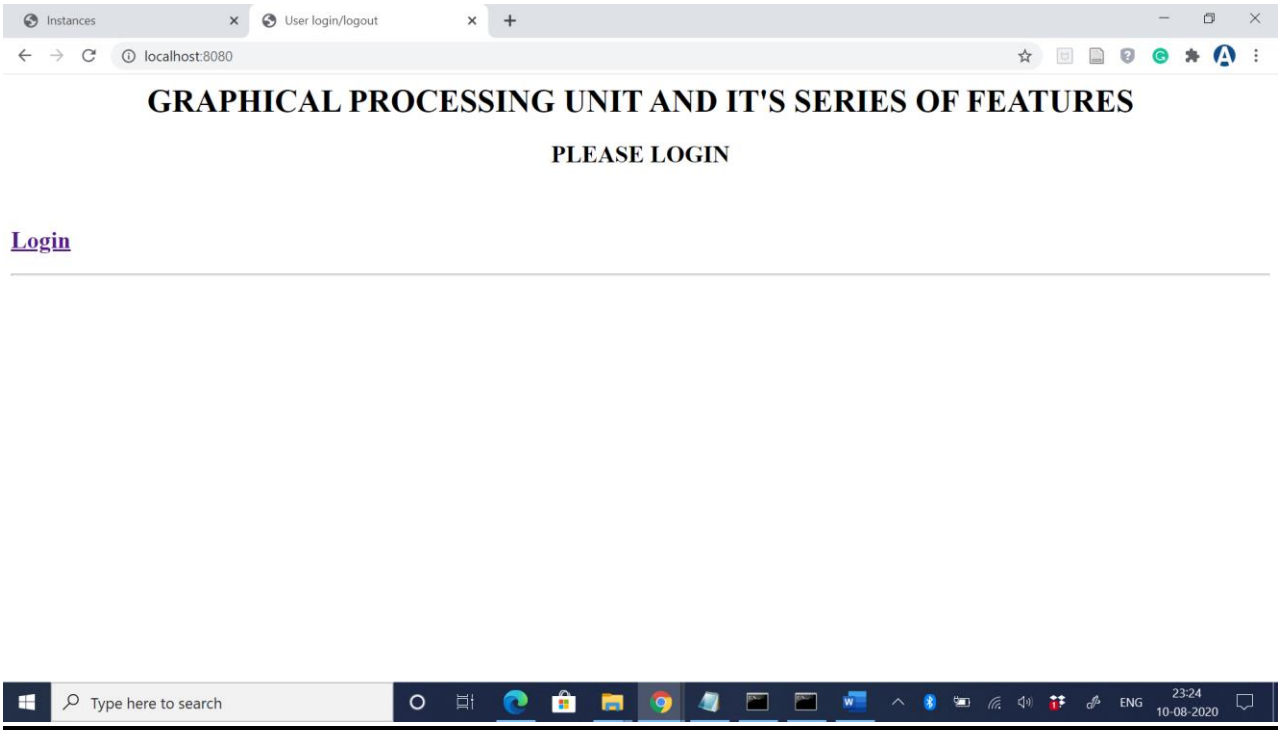
OVERVIEW:

In this pdf, I'm going to depict and explain the methods, I have used in this project. Overview of the project is Creating and processing GPU. Totally, there are 7 brackets we are covering in this pdf.

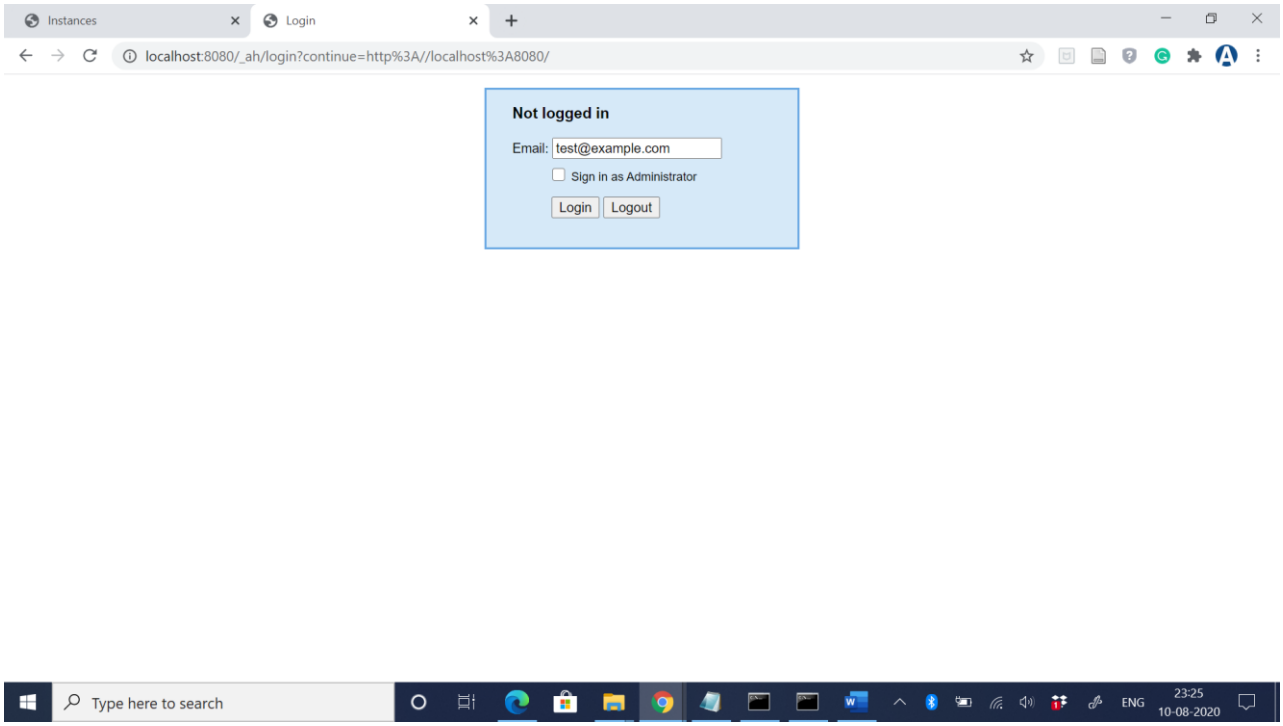
- **Bracket 1**: Providing login and logout service for users.
- **Bracket 2**: Allowing user to create a model with GPU information and set of features. Input GPU name is made into key name.
- **Bracket 3**: Building User Interface form user to get information and store those information in our datastore
- **Bracket 4**: No same GPU key name is selected and listing all the available GPU names in the main page.
- **Bracket 5**: Making a hyperlink to view and edit the GPU information for users.
- **Bracket 6**: Allowing user to find the desired GPU by selecting their features.
- **Bracket 7**: Comparison of Two GPU's in separate page.

9

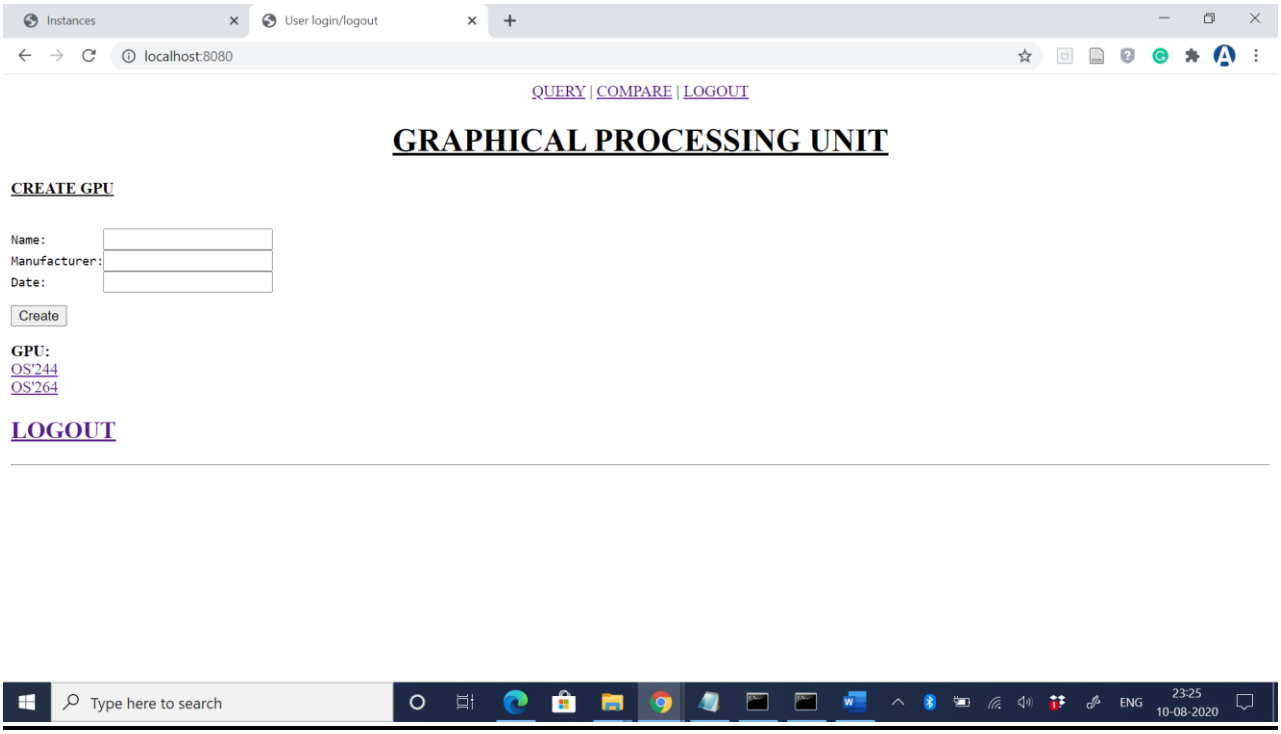
BRACKET 1



This is our first page in our project. Here we are displaying the title of the project and a log in Uri. Initially in the header file, we are importing the **users** from **google app engine** in order to fetch the users from login. This particular module will link to the **google login/logout service**. We are using main.py and main.html for implementing the login/logout service. We are using html templates to render the values on the flow. The package which is useful for such operations is **JINJA**. In main.html, **"{% %}"** the tags which we use in main.html is not actually html commands. They are from jinja templating system. This particular tag are called as statement delimiters and **"{}"** and these are called as expression delimiters which helps to fetch values and print it. Now we are importing "os" which helps jinja to work correctly. In main.py, We are creating a variable named as **"JINJA_ENVIRONMENT".** We are setting up the environment for jinja to work. **"loader=jinja2.FileSystemLoader"**, This command tells jinja to find all the templates that will be used by the request handler. In this command **"(os.path.dirname(_file_))"**, we are asking JINJA to use the same directory as the current file we are editing. We are creating a class "MainPage", We are using the command **users.get_current_users** to pull the current user from the request. If we fetch a user, we are creating a logout url for the user who logged in our project with the command "**users.create_logout_url(self.request.uri)".** In the variable **"template_values"**, we are passing all the values to our html file to be printed. And finally, we are creating a object **"app"** which is useful to start the operations of whole process.

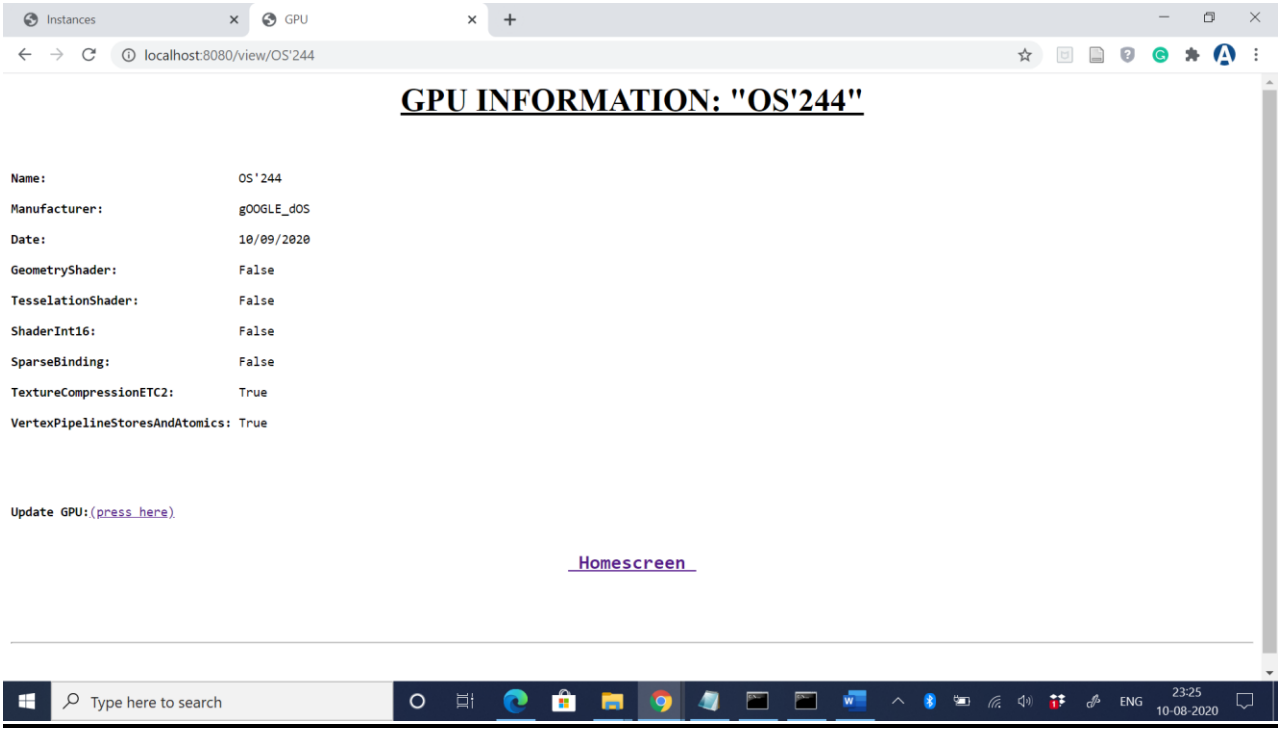


BRACKET 2:



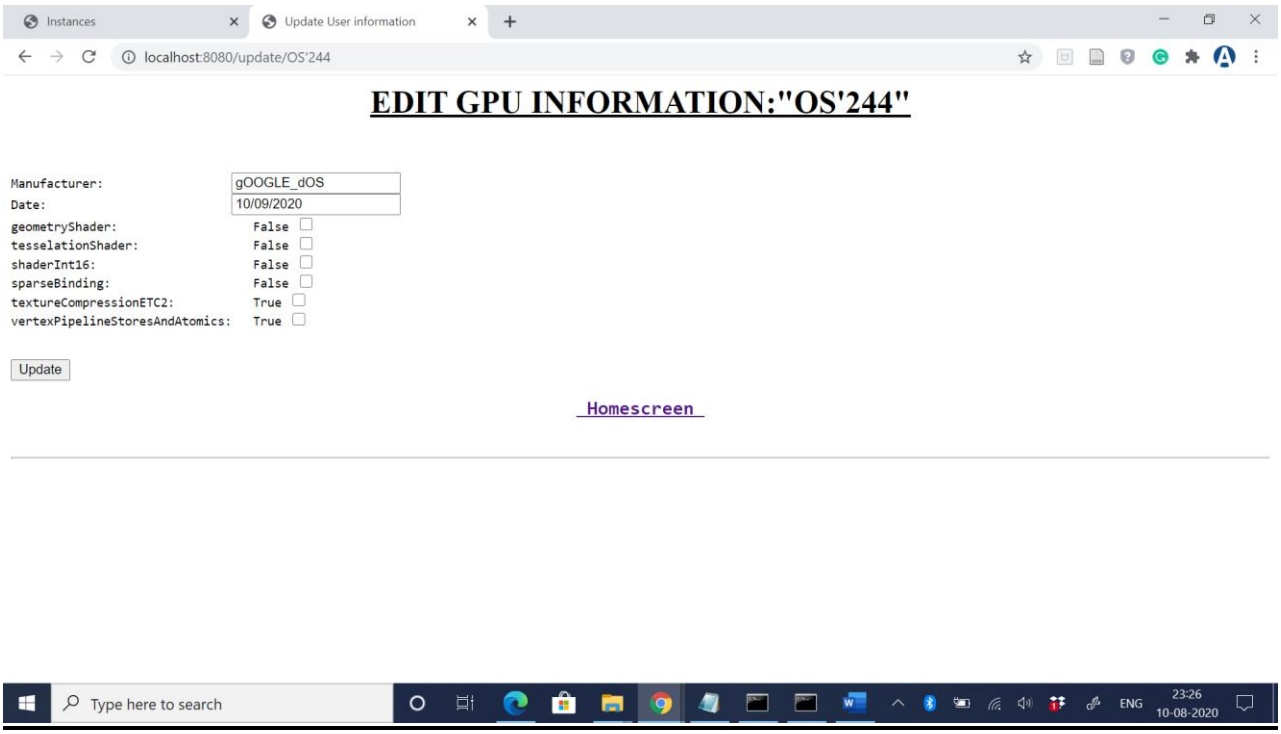
We are creating a python file named as **"myuser.py"**, In this file we are going to create some models to store all the values of GPU which will be given by user. Initially we will be importing a module **"ndb"** from **"google.appengine.ext"**. which allows us to fetch values and store values in that desired variable. We are creating nine variables like name, manufacturers name, date and remaining features of the GPU in our model. So, we are using **Stringproperty** and **booleanproperty** which helps us to store the string values and boolean values. **"ndb.model"**, this model helps us to enable all the data store operations in the python datastore. Now we are fetching the values from the user using the **self.request.user("users_name")** which helps us to store the information of gpu name. Once we fetched the values from the user, we are going to make that gpu name as the key name in the datastore. We know that that our gpu name will be empty, so we cannot able to fetch the key. With the command of **"variable_name=database_name(id=variable_gpu_name)"** we can able to give the store the name in the datastore and when we the command **"variable_name.put()".** This will create a gpu database with the gpu name.

BRACKET 3:



In this bracket, we are creating a form to get the information of all GPU and its features. In the main.py, under the class name MainPage, we are performing operations in the post function. Once we get the gpu name by name=**"self.request.get("users_name").** we will then create a key for that by issuing **"(id=name)".** Now we will get the key of the desired GPU and we will get the remaining values too by issuing **"myuser.name=name"**, **"myuser.manufacturer=self.request.get("users_manufactures").** Once we get all the information, we then issue the command of **myuser.put().** To store these informations in the database. The keyword **"users_name"** is used in the **"main.html"**, In that file we will getting by **value="users_name"**. So, the html file post method gets activated and fetched in main.py post function. In main.html, we will be using a button "Create". when user sumbits that button, we use if condition like **"if self.request.get('button')=='Create':" .**

BRACKET 4:

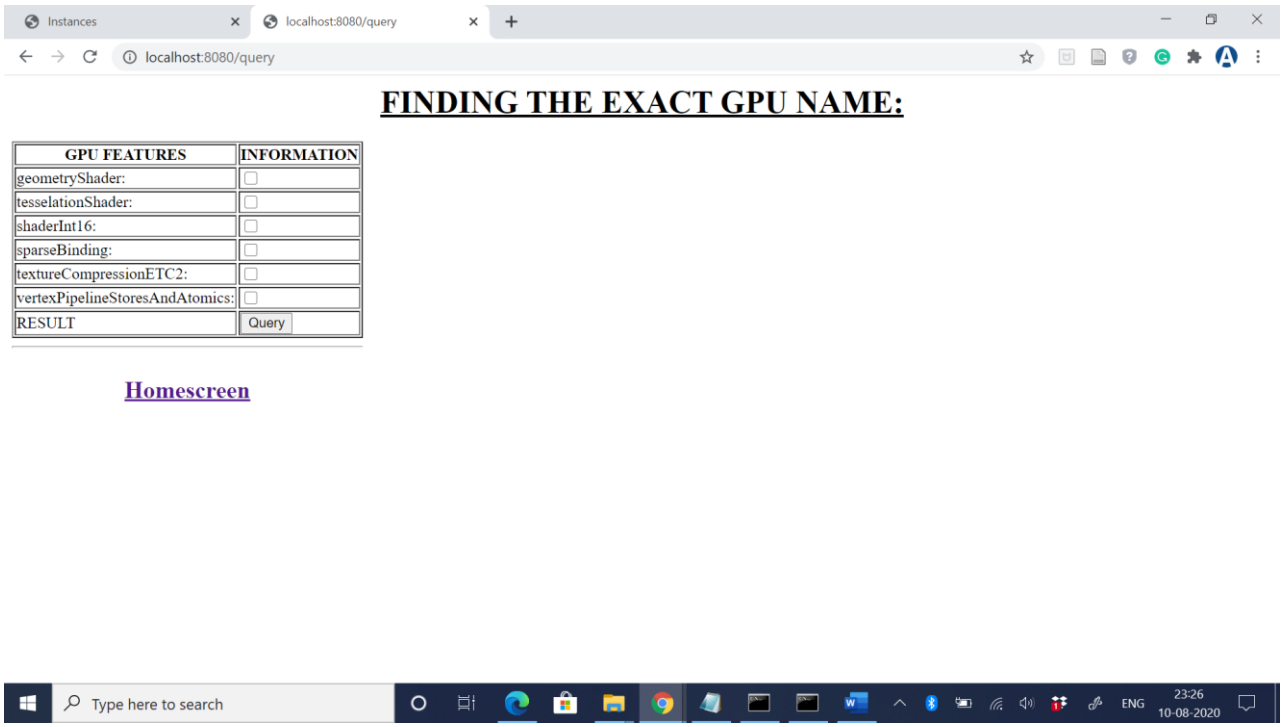


In this bracket, we are going to **prevent the overwriting of key GPU names** and we are going to **display** all the available **GPU names in the mainpage**. We are preventing the overwriting by, we know that if existing GPU name is selected, then it will fetch the key value and goes to the else part of **"post function" "if myuser==None: else:".** This will automatically enters the else part and hyperlink is created to show that, particular username is fetched. Now in "main.py", we issue a command **"result=MyGPU.query().fetch()"**, So this command fetch all the values from our database and store it in result variable. Now we will pass this result variable in "**template_values".** In our main.html file, we will be using a for loop like, **"{% for item in result %}".** Now we will fetch our values by issuing a command **"{ item.name }".** So this will display all the available GPU names from our database.

BRACKET 5:

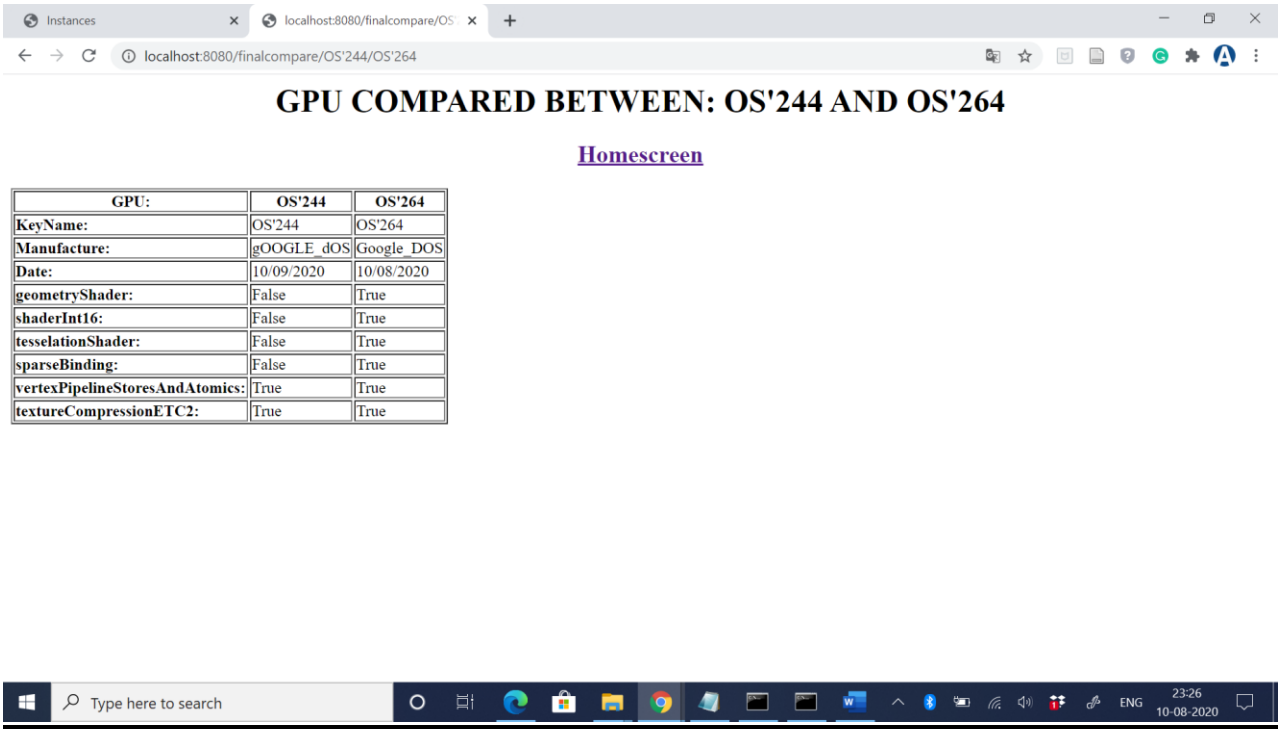
Hyperlink is created by the command **"<a href="link path" {{ item.name }}".** So, inside the for loop, instead of replacing we will be replacing the command {{ item.name }} to this command. So hyperlink is created for each and every databases. So when the gpu name is selected, we will be directing to **"/view/{{ item.name }}".** So now in our view.py, we will be creating a class name for this particular html file. In our get method we will be receiving this gpu name as **"def get(self,id):".** Now in that id we will be having our gpu name, so now we can able to fetch the key of that particular gpu name and passing the myuser in the html file to display all the values. For editing it repeats the same steps, except instead of showing information, we will creating again textbox and checkbox to get the new informations. So now new information is created and saved in our datastore by hitting the command Submit which will update by **"myuser.put()".**

BRACKET 6:



In this query part, we will be fetching all the boolean functions to compare with our databases. So when we get all the values, we will be issuing a query of **"q=MyGPU.query(ndb.AND(MyGPU.geometryShader==geometryShader,ndb.AND(MyGPU.vertexPipelineStoresAndAtomics==vertexPipelineStoresAndAtomics,ndb.AND(MyGPU.tesselationShader==tesselationShader,ndb.AND(MyGPU.shaderInt16==shaderInt16,ndb.AND(MyGPU.sparseBinding==sparseBinding,ndb.AND(MyGPU.textureCompressionETC2==textureCompressionETC2)))))).fetch()"**. So this single line query matches all the boolean functions our databases and fetches it. Now we will using a for loop to display the GPU name which matches these features. So, we will be using **ndb.AND** function to compare each and every boolean operations which is given in the database.

BRACKET 7:



GPU:	OS'244	OS'264
KeyName:	OS'244	OS'264
Manufacture:	gOOGLE_dOS	Google_DOS
Date:	10/09/2020	10/08/2020
geometryShader:	False	True
shaderInt16:	False	True
tesselationShader:	False	True
sparseBinding:	False	True
vertexPipelineStoresAndAtomics:	True	True
textureCompressionETC2:	True	True

In this bracket, we will be getting two input names of gpu with help of **select box** in our compare.py file. We issued the html command in the python file itself. When the user selects select the values from the select box, it is now fetched in python file with the command of **gpuname1=form.getvalue("dropdown").** Now repeating the same step to get the input of second gpuname. When these values are fetched, we use two keynames in different variables namely **myuser and myuser1.** Now we pass both the variables in html file and print all the values. Now both the gpu information