ELECTRIC VEHICLE

Name: Gokul Sreeletha Kannan
Student ID:2985939
Course: MSCC
Assignment: Building an Electric Vehicle manager
Module: Cloud Platforms and Applications

Introduction

This is a documentation for the web application designed using webapp2 framework on the Google app engine. Interfaces are rendered using the Jinja2 template engine and ndb is the database used to store data. The application is a database for Electric Vehicles. The user can login and add a vehicle to the database. Anybody can go to the application and search for vehicles. They can see all the specifications of the vehicles. Also can compare two or more vehicles to find the best one for their needs. Also logged in users can rate a particular vehicle and can also submit a review to the vehicle. Guest users will be able to see the details of a vehicle including reviews and average ratings, but only logged in users can edit the database.

main.py

main.py is the main handler of the application. It consist of MainPage class which renders the main.html template. It is the home page of the application. The MainPage checks for user login and renders the template accordingly. It also provides links to other pages.

eVehicle.py

This is the data model for the application. It consists of EVehicle class that is inherited from ndb.model. It consists all the attributes of an electric vehicle.

Attributes:

- *name: This is a string property that holds the name of the vehicle.*
- *manufacturer: This is a string property that holds the name of the manufacturer of the vehicle.*
- *year: This is an integer property that holds the year which the vehicle has came into market.*
- *batterySize: This is a floating point number that holds the battery size of the vehicle.*
- *wltpRange: This is a floating point number that holds the wltp range of the vehicle.*

- *cost: This is a floating point number that holds the cost of the vehicle.*
- *power: This is a floating point number that holds the power of the vehicle.*
- *reviews: This is a string array that holds all the reviews of the vehicle.*
- *rating: This is an integer array that holds all the ratings of the vehicle.*

The class also consists of two methods.

- *isUnique(): This method checks if there is another vehicle in the database that has the same name, manufacturer and year and returns true or false accordingly.*
- *avgRating(): This method returns average rating for a particular vehicle*

addEV.py.

This page consists of AddEV() class that allows logged in users to add a new electric vehicle. There is a get() method that renders the page using the addEV.html template. It takes in the values for the attributes of the vehicle and creates a new object of the EVehicle class. The isUnique() method is called on this object to check if there is a duplicate object already in the database and the object is written into the database if there isn't.

search.py

This page is rendered using the search.html. This page allows anyone to search for electric vehicles in the database. The user can filter through each of the attributes. The page consists of Search() class. The get() method renders the default template. The template contains the forms for searching. In the post() method, the values in the form are saved into some variables. Then all the vehicles in the database is queried in the query variable (as the default case). Then the query is filtered using the filter property of ndb. The final result is passed into the template as template_values and is re-rendered using the same template. Then each of the desired output is shown in the interface as links using <a> tags. The <a> tag links to the ev.py which shows the features of the selected link. The id of the specific vehicle should also be passed to the ev.py to make this possible.

ev.py

This page consists of EV() class which is rendered using ev.html. This class is used to show the users the details about the specific vehicle they selected on the search page. The id passed from the search is saved here to a variable and the object that has that id is taken in the get() method. Each of the attribute of the object is given to the template as template values. The id is also passed as the logged in users should be able to edit the details in the post() method. In the post() method, the same object is taken again from the database and the attributes are

overwritten. The if condition for rating and review is used to avoid writing empty reviews and ratings into the database as they are arrays.

compare.py

This page consists of Compare() class which is rendered using compare.html. In the get() method, all the vehicles in the from the database is queried and is passed into the template. The template renders each vehicle as a checkbox. The value of each checkbox is the id of the displaying vehicle. The user can select two or more vehicles from the checkbox and hit the compare button to compare. In the post() method, all the ids that the user selected is taken and is stored into an array called carIDs. An empty array called car is also created. By looping through carIDs, each vehicle that represents each of the ids in the carIDs is taken and is added to the car array. Then, it is checked if the user has selected less than 2 vehicles by checking the length of the car array and an error is rendered if it is less than two. Otherwise, for the color coding, the max and min value of each of the attribute calculated using functions from the functions.py page. These values along with other necessary ones are passed into the template. In the template, the color coding is done using some if-elif-else loops and table tag.

functions.py

This page does not have a template. This page is created to define the methods that calculate the max and min values of the attributes.

- getMaxAvgRating(): returns the max value of average rating
- getMinAvgRating(): returns the min value of average rating
- getMaxBattery(): returns the max value of battery size
- getMinBattery(): returns the min value of battery size
- getMaxWltpRange(): returns the max value of the wltp range
- getMinWltpRange(): returns the min value of the wltp range
- getMaxCost(): returns the the max value of cost
- getMinCost(): returns the the minvalue of cost
- getMaxPower(): returns the max value of power
- getMinPower(): returns the min value of power