Assignment 1

Cloud Computing

Building an EV Database

By-Shubhkumar Bharatkumar Patel

MSc-Big Data and analysis

Student ID – 3077432

1. About EV Application

This application is based on platform as a service (PaaS) in which users may save information about different Electric Vehicles. Users can add, remove and compare different vehicles saved in the database. Some of these features only operate when the user checked in with their login ID only. Similarly, any user can search the database, filtering by attributes to find EVS that meet their requirements. EVS are shown as hyperlinks that direct viewers to the information page, where they can modify, delete, and write reviews.

This application aims to store Electric vehicles according to their name, manufacturer, year, battery size, WLTP, cost and power. And compare them according to the need of a user

2. Reference material used for learning

The 'Google App Engine with Python' pdf provided as part of the module was used to master the basics of Google App Engine. Google App Engine documentation was used to better grasp the concepts of entity, key, datastore, and queries. At the end of this page, you'll find a list of all the references.[1]

3. Script Used

INDEX.YAML

We normally create an index on the entity's property in this file. However, no index is created in this application. As a result, in this scenario, all the indexes will be generated automatically.

index. YAML has a single list element called indexes.

Kind: The entity's kind for the query.

Properties: A list of properties that should be included as index columns.

The property's name is in the datastore.

The sorting direction is either ascending or descending, with ASC being ascending and desc being descending.

MAIN.PY

This is the main Python file, which contains all of the classes and is necessary for the program's effective implementation.

1. Os- We've imported os

2. Google.oauth2.id token- Enables the verification of 'OpenID Connect ID Tokens,' especially those generated by Google infrastructure.

3. Flask- The flask object is the main WSGI application implementation object. The name of the application's module or package is given to it. After it's finished, it'll be used as a central repository for view functions, URL rules, template configuration, and much more.

4. Requests- The requests module allows you to use Python to send HTTP requests.

5. Datetime- Although a date is not a data type in Python, we may work with dates as date objects by using the DateTime class.

Def. Keyword.

The def keyword is used to define a class's methods in the case of classes. The likely practical application is that it provides the property of code reusability. Instead of writing the same piece of code over and over, we can use the def keyword to declare a function and write the code inside the function.

retrieveUserInf

All logged-in users will be granted unrestricted access to a different application path.

createUserInf

It will be used to generate the user's login and logout URLs.

add_Ev

Using the default option, this technique creates default ids for EVS. The EV object and the user status are rendered to add.html, which displays the form for inputting EV data.

Show_ev

The list of EVs is retrieved from the database and saved in an object, which is then rendered to list.html with each EV name as a hyperlink. This technique also renders values to the information page(EV_Info.html), which is the page that appears when the hyperlink is clicked, and URLs for each EV are generated dynamically. Each EV's id is sent to the information page through HashMap, which

displays information about that particular EV based on its id. The average rating and reviews of each EV are also given in reverse chronological order on the information page. This review query (data object for review and rating built based on each EV's id) is created and filtered by matching id.

add_evs

This function adds New EVs,s By using attributes As follows:

a.The StringProperty 'name' is declared. Because the name of an Electric Vehicle is primarily alphabetical with a few letters or digits thrown in, it is saved as a string.

b. The StringProperty'manufacturer' is declared. Vehicle manufacturers are commonly represented by alphabetic characters with a few digits thrown in for good measure. Both are accepted by StringProperty.

c. The property 'year' has been declared as an integer property. For the year field 1830-2022, page limits are included in the add.html file. After reviewing facts about the first Electric Vehicle, this decision was made. This was a decision made only for proper validation.

If incorrect year integer values are allowed, the data's integrity will be compromised.

d. FloatProperties are declared for 'battery size,' 'Cost,' 'Power,' and 'WLTP Range.'

e. The FloatProperty 'cost' is declared.

f. The property 'power' has been declared as a FloatProperty.

g. 'WLTP_Range' is declared as FloatProperty

Electric Vehicles' battery size, cost, power, and WLTP Range properties all contain decimal values, hence they're declared as FloatProperty. Validation, on the other hand, is limited to Cost is allowed to two decimal places, and revenue is allowed to three decimal places. the other three characteristics The'step' option in HTML is used to do this.

search

Users can use this technique to query the EV datastore. If the action is 'Search' (from the button on the search.html page), data from the EV database is filtered out. The EVS that meet the criteria is listed as hyperlinks with their names as links. This is accomplished by utilizing conditional statements to determine whether or not all of the fields in the search form are filled. This function retrieves data from user-input fields and conducts a query operation on these characteristics. Through this function, we are searching attributes like name, manufacturer, year_low, year_high etc.

edit_ev

This function is responsible for HTTP Get requests for the Edit class.

Delete

This function is used to request the deletion of data from Datastore from the user.

compare_evs

This solution displays the list of EVs as checkboxes on the compare page (compare.html).

get_reviews

This function renders EV_Info.html with the current user object and the EV Review object. Every EVReview's id will be set to default.

get_average_score

This solution Displays the Average score of compare results on the compare page (compare.html)

Root

This Function Redirects to localhost (127.0.0.1)with Port no.8088.

POST Action

The structure is similar to the root function, although there are a few minor modifications. For starters, this will monitor the relative URL for a post-action.

Flask

Flask is a web framework .it is provides us with tools, libraries, and technologies that allow us to build a web application.

Firebase

Firebase is a platform where we can build mobile and web applications. we have used for SIGN IN Method and give Authentication to a user for Google (only LocalHost) and Email SIGN IN and we have added a firebase configuration script in the app-setup.js file.

Cloud-Datastore

Cloud storage allows us to save data and files in an off-site location that you can access either over the public internet or a dedicated private network connection. Because actual files may be replaced with cloud storage records, this is a particularly cost-effective solution for enterprises. We have used Datastore to add data to the database and make entities for Ev cars and Review info.

google-auth

The Google authentication library is implemented in Python by Google-auth. This module allows you to authenticate to Google APIs using a variety of methods. It's also compatible with several HTTP libraries.

- Default Google Apps credentials are accepted.

- Support for the JWT signature and verification protocol.

- The generation of Google ID Tokens is possible.

- ID Tokens can be validated and decoded.

- Google Service Account credentials are accepted.

- It is possible to mimic Google Credentials.

- Credentials for Google Compute Engine are accepted.

- You can use your standard Google App Engine credentials.

Request library

Request Library is a simple HTTP library for Python and built for human beings. Queries make it very simple to send HTTP/1.1 requests. There's no need to add query strings to your URLs or form-encode your POST data manually. Thanks to urllib3, keep-alive and HTTP connection pooling are completely automated. Requests are ready For today's webpage records.

Index.html

It's the primary index.html document. It works as the home page. The login and logout of users are done here. Then you can use to search option for Ev Car Search we can see cars data in the below index I have added two buttons to locate to add EV Page and compare page.

Add.html

This file contains every single entity, from name to power.

• There is an Add button on it.

• This will cause the data to be uploaded.

Compare.html

The data is shown here at the start in Scroll Option

• Checkboxes are used to search the search button.

• When we select, only the values for class data are displayed.

Edit_Page.html

When we click on the table's name attribute, which is a hyperlink, we are taken to this page.

• There are three buttons on this. Edit, remove and review your work.

EV_info.html

Here We Redirect to Select Ev Car Info and We can See all detail from Name to Power. The reviews are entered into the text box at the bottom of the page also included is a 0-to-10 scale for client ratings. The review is submitted when you click the submit button.

References:-

1. Denby, B., 2017. Google App Engine By Example.

Learning to Web Design A Beginner's Guide to HTML, CSS, Graphics, and Beyond

Python.org