Documentation:

1. get method:

this method is response to request from browser. for example when user execute a URL <http://localhost/> URL router will check the respected class mapping.

app = webapp2.WSGIApplication([

('/', MainPage),

])

WSGI will map the URL to Main Page class where get method resides. then this method return response webpage which will display to user.

Similarly we have different class for different purpose and each will have there get method if required. likewise if user wasn’t to see vehicle detail page we ShowVehicle class

app = webapp2.WSGIApplication([

('/show', ShowVehicle),

])

when user execute URL <http://localhost/show> then Show Vehicle class will be called where get method is there which will return show-vehicle.html page in response and it will be displayed to user.

Parameters:

get method can have single parameter self or can have more than one parameters

e.g.

* def get(self)
* def get(self, ev\_id)
* def get(self, data)

above we can see get method with multiple parameters which we have used as needed.

ll. Post method:

Post method handle post request from browser when form get submitted or similar action happened using Post protocol. for example from add.vehicle.html page we have

<form onsubmit="return checkev()" method="POST" action="/add">

when above form submit it fires a post request at "/add" URL. this URL is mapped to AddVehicle class

app = webapp2.WSGIApplication([

('/add', AddVehicle),

])

hence in Add Vehicle class it search for Post method once found it will send data to it and that method will process the information as in this case it will store EV vehicle information to database.

similar to get method post method can have multiple parameters passing different objects. in our case we have

* def post(self)
* def post(self, ev\_id)

Post request is more secure than get request as it doesn't pass parameters in URL.

hence for sensitive information we use Post method only.

lll. get\_current\_user()

method retrieve logged in user information from user object.

lV. create\_login\_url(self.request.uri)

Method used for creating login URL for user who have logged out or haven't logged in.

lV. create\_logout\_url(self.request.uri)

Method used for creating logout URL for user who have logged in.

V. get\_template(page.html)

Method used for passing html webpage to Jinja environment

Vl. response.write()

Method used to write information to webpage or send webpage to browser using JINJA.

Vll. render()

Method use for passing variables to html page using JINJA.

Vlll. query()

Method used to query values from datastore(database).

lX. order()

Method used to sort queried values of database in perticular order

X. fetch()

Method return queried values of database in list

Xl. strip()

Removes whitespace present in string in bothe sides.

Xll. capitalize()

Makes first word of string capital and rest small

Xlll. filter()

Filter query response with particular value or field

XlV. int()

Convert float value to int

XV. get()

Retrieve particular value from database basically a search in database.

XVl. dumps()

convert python object to json formatted string.

XVll. count()

count number of retrieved results of particular query.

XVlll. put()

Stores object values to datastore.

XlX. delete()

Delete object from database

XX. loads()

convert json string to python object

XXl. get\_by\_id()

Retrieve data from database using id.

XXll. append()

Append data to list.

XXlll. min()

find minimum value is a list

XXlV. max()

returns maximum value in a list

Models used:

EvDatabase:

* name: used for storing vehicle name
* manufacturer: used for storing manufacturer name
* year: used for year in which vehicle is manufactured
* battery\_size: stores value of battery used in vehicle in Kwh
* range: store how much distance in Km an ev can travel on full charge
* cost: cost of ev is stored in this column
* power: how much power ev have in Kw
* created\_by: save user object so that we know who have created this entry and only that person can edit or delete it later
* average\_score: average score of Ev obtained from Reviews and Ratings
* carkey: carkey we have used to identify if Ev of same manufacturer, nam and production year is in the database or not
* date: date of entry created which help to sort the list of Ev's.

Review:

* created\_by: save user object so that we know who have created this review
* review: review given by user for particular ev
* rating: rating given to particular ev by user ranging from 1 to 10.
* carkey: carkey we have used to identify the Ev for which user is writing review
* date: date of review created which help to sort the list of reviews.

UI consideration:

We make ui as simple as possible so that user didn't get confused while operating.

simple login page consist of two options "login" or "continue as guest" is easy to understand.

further on dashboard page we use regular element and hints so that user can understand what button or input method functions.

for searching in different categories we used list for selecting type of search which is easy to understand

for ev comparison we have used checkbox so that user can select as much Ev he want

when user select 2 or more Ev then only compare button appears. so that accidental comparison when less than 2 Ev selected should not trigger.

When showing reviews when no review is there we display no review message instead of numbers making it more user friendly.