Vehicle Fuel Consumption Ratings

D532 Final Project (Part 3)

Team

Anne Wesley (aewesley@indiana.edu),

Shubham Saurabh (ssaurab@iu.edu),

Varun Joshi (vajoshi@iu.edu)

1. Web App Architecture

1.1 Data Storage

We are using SQLite database to store all the data needed for the application. The screenshot of the code snippet below shows the creation of the SQLite database and the storage of the initial data.

```
import sqlite3
import pandas as pd

connection = sqlite3.connect('emission_rating.db')



with open('schema.sql') as f:
    connection.executescript(f.read())

cur = connection.cursor()
read_raw = pd.read_csv(r'C:\Users\vajoshi\PycharmProjects\d532project\MY2022 Fuel Consumption Ratings.csv')
read_raw.to_sql('raw', connection, if_exists='append', index = False)

cur.execute("INSERT INTO fuel (fuel_id) SELECT DISTINCT `Fuel Type` FROM raw")
cur.execute("INSERT INTO car (car_id,model_year, make, model, vehicle_class, engine_size_l,cylinders, transm
cur.execute("INSERT INTO fuel_consumption (source, car_id, fuel_consumption_city_lp100km, fuel_consumption_h
cur.execute("INSERT INTO emission (car_id, co2_emissions_gpkm, co2_rating, smog_rating) SELECT car_id,`CO2 E
connection.commit()
connection.close()
```

1.2 Application Backend

Our application backend is built using Flask in Python. In general, our application is controlled using Python allowing us to manipulate the data in SQLite database and route the Flask application to return HTML output. Screenshot of the code snippet to build application backend is shown below:

```
from flask import Flask, render_template
import sqlite3


def get_db_connection():
    conn = sqlite3.connect('emission_rating.db')
    conn.row_factory = sqlite3.Row
    return conn

app = Flask(__name__)

@app.route('/')
def index():
    conn = get_db_connection()
    car = conn.execute('SELECT * FROM car').fetchall()
    conn.close()
    return render_template('index.html', car=car)

def get_car(car_id):
    conn = get_db_connection()
    car = conn.execute('SELECT c.model_year,c.make,c.model, fc.fuel_consumption_combo_lp100km ,fc.fuel_cons
```

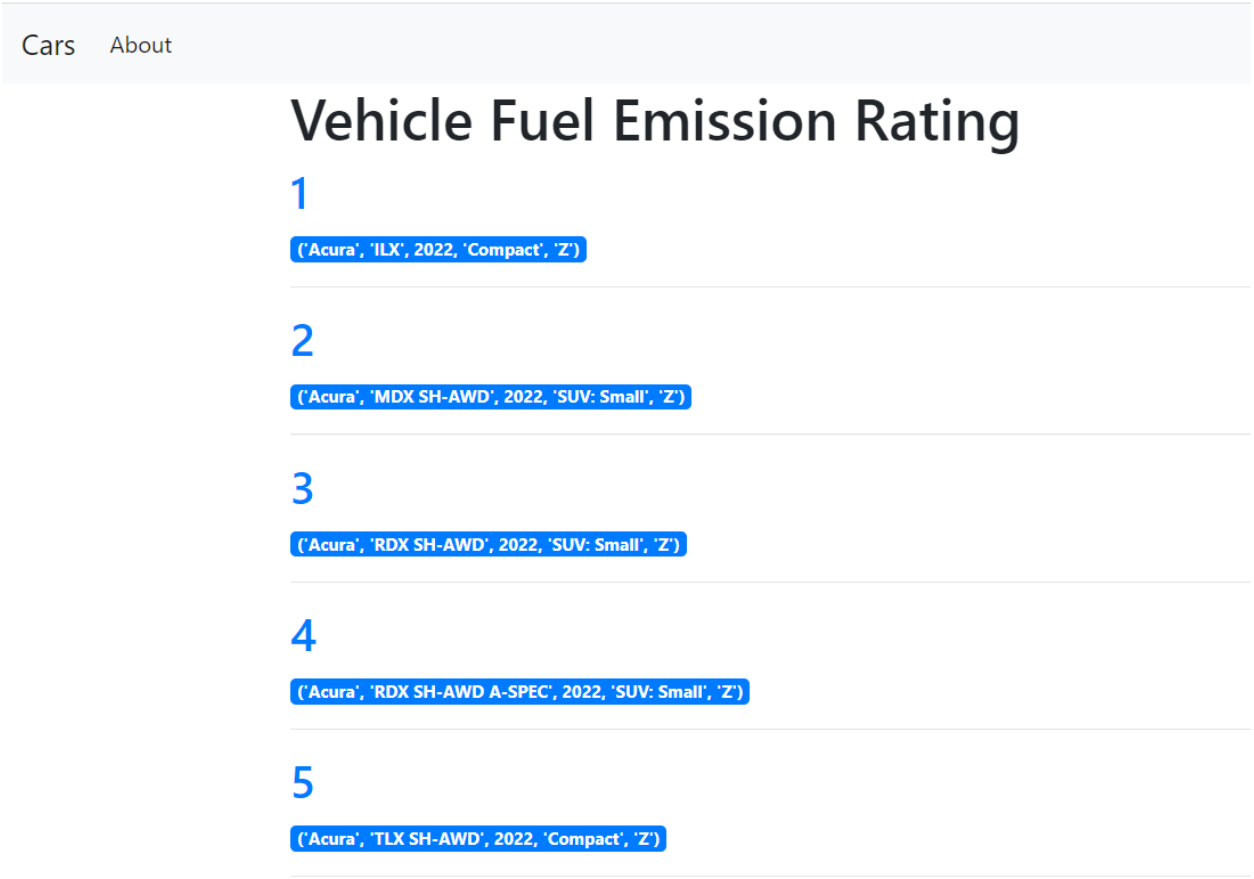
1.3 Database Access and Security

Our application is a general utility providing information to the retail car consumers about vehicle emission ratings as reported by the car manufacturers. Therefore, we restrict the database tables, which store the data provided by the car manufacturers, from any edits by the consumers. The end users are only allowed to update their own consumer reported fuel consumption data. The database tables are secured by applying necessary constraints to restrict update/delete on these tables.

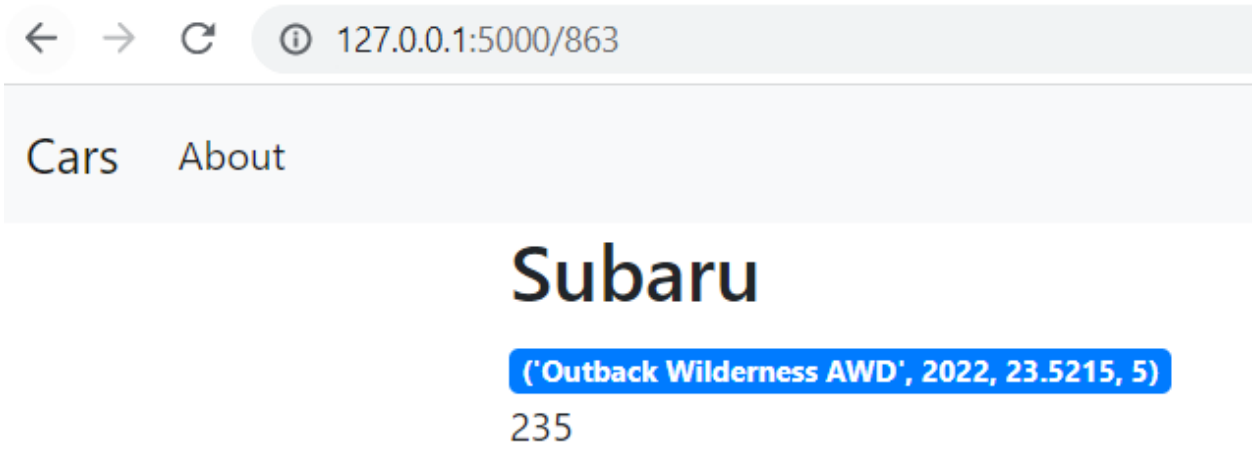
1.4 Application Front End

Our application uses HTML for building the front end. The overall app layout is built using Figma. The Flask app will render the output to HTML URLs. The screenshot of the raw HTML output (without design layout) is shown below:

All cars with car id:



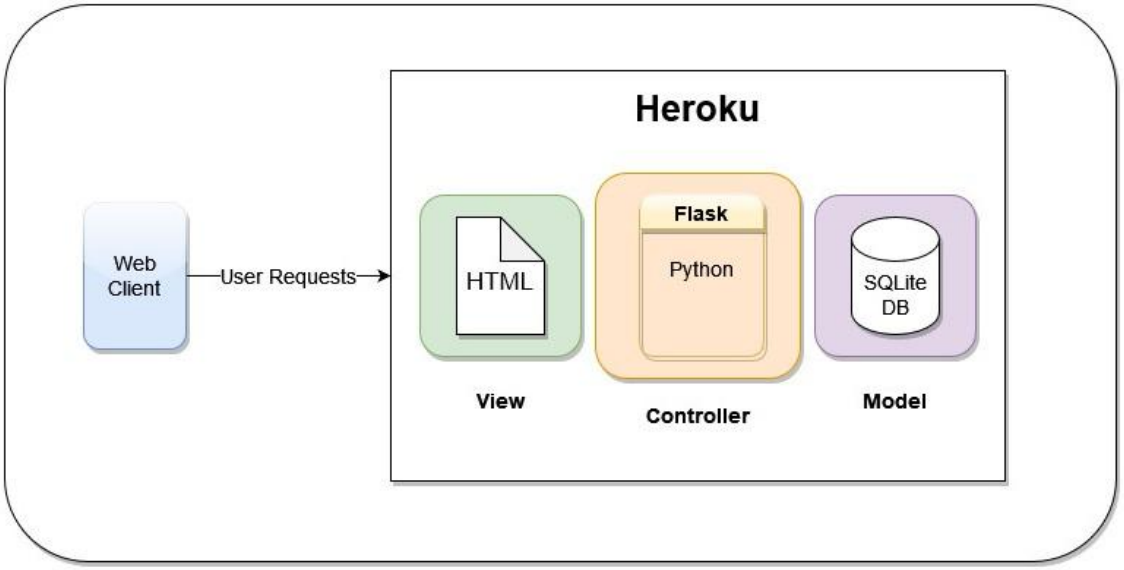
Car with a random car id (863) and CO2 emission rating:



1.5 Application Deployment

The application has been developed and deployed locally to reduce development and testing effort. The application will be deployed to Heroku which has support for Python applications.

1.6 WebApp Architecture



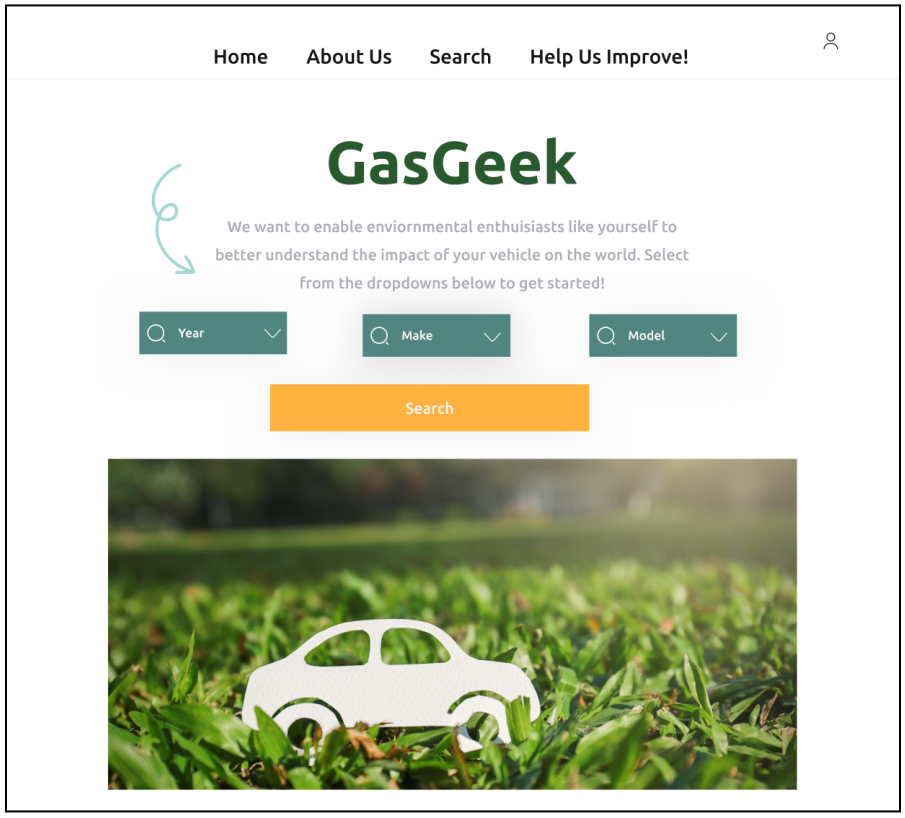
2. Web App Layout

We utilized Figma to create the layout for the User Interface of our web application. With Figma we have created the layout for 3 pages (described and shown below), the home page, a search page (read), and an improvements page (create/update/delete).

2.1 Home Page

The initial layout shown when a user opens the web application is the Home page shown below. The Home page has a navigation bar at the top (menu panel) to allow the user to easily open other pages of the site. Below the site logo, there is a brief description of the goal of the application. Below the description is 3 drop down menus that allow a user to input the Year, Make and Model of the vehicle they want to learn about, and a search button to initiate the read of our database. This functionality allows quick and easy read capabilities to our database, and when the user initiates the search, they are taken to the Search page. In terms of overall color scheme, we chose to go with a clean and minimalist style with hints of green and yellow due to the fact that this application is geared towards environmental enthusiasts and we felt that these colors reflected that theme.

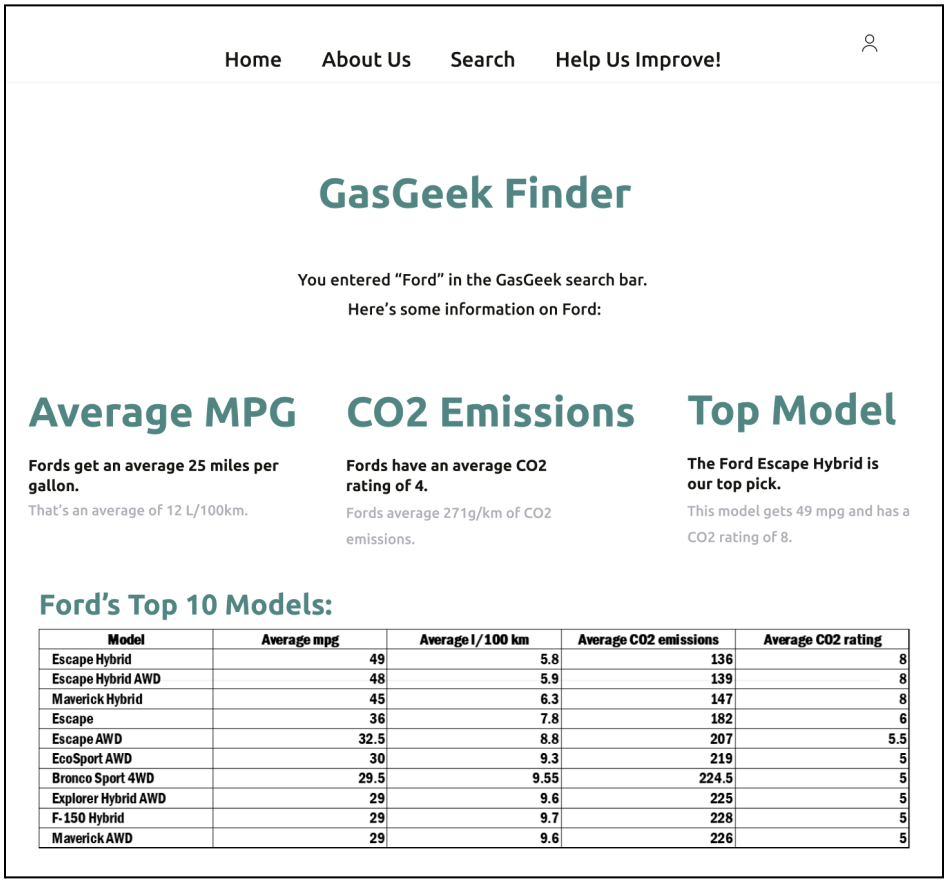
Main page



2.2 Search Page

The next page that we designed is the search page. This page displays information from our databases based on the choices made on the Home page. For the search page, we have chosen to show an example of what is displayed if a user enters only the Make of a vehicle, as we hope that users will use this application to research a variety of vehicles, not just their own. The page first displays some interesting emissions information pertaining to the Make entered by the user on the Home page, and displays what we feel is the top model based on MPG and CO2 Rating. We then display a table containing the top 10 models of the Make based on MPG and CO2 rating.

Search page

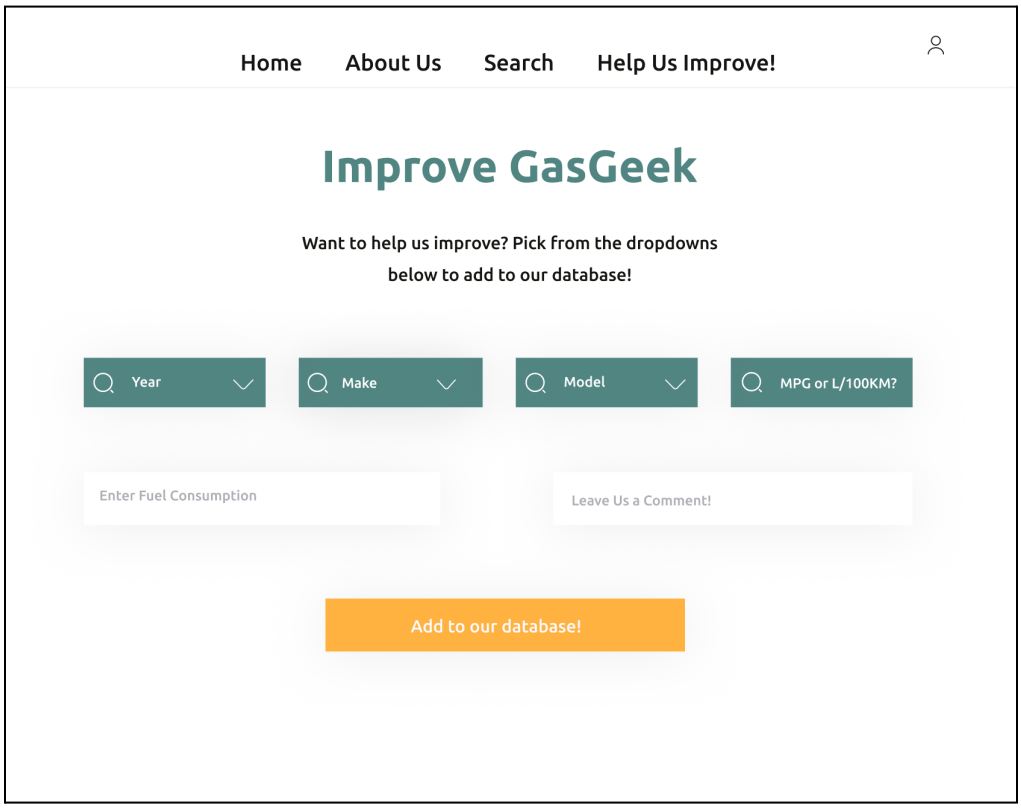


Model	Average mpg	Average l/100 km	Average CO2 emissions	Average CO2 rating
Escape Hybrid	49	5.8	136	8
Escape Hybrid AWD	48	5.9	139	8
Maverick Hybrid	45	6.3	147	8
Escape	36	7.8	182	6
Escape AWD	32.5	8.8	207	5.5
EcoSport AWD	30	9.3	219	5
Bronco Sport 4WD	29.5	9.55	224.5	5
Explorer Hybrid AWD	29	9.6	225	5
F-150 Hybrid	29	9.7	228	5
Maverick AWD	29	9.6	226	5

2.3 Improvements Page

The improvements screen will be accessible from the navigation bar and will allow users to add to or update our database. On this page there will be four drop downs available for the user to enter the Year, Make and Model of the vehicle they wish to add or update, as well as the desired units (either MPG or L/100KM). Below the four drop downs, there will be two boxes that allow users to enter their observed fuel consumption (in the units specified in the dropdown) and any comments on the vehicle that they want to add to the database. The yellow button allows the user to submit this information to the database.

Search page



3. Individual Assessment

Assessment Area	Comments
	1/10 -
	1/10 -