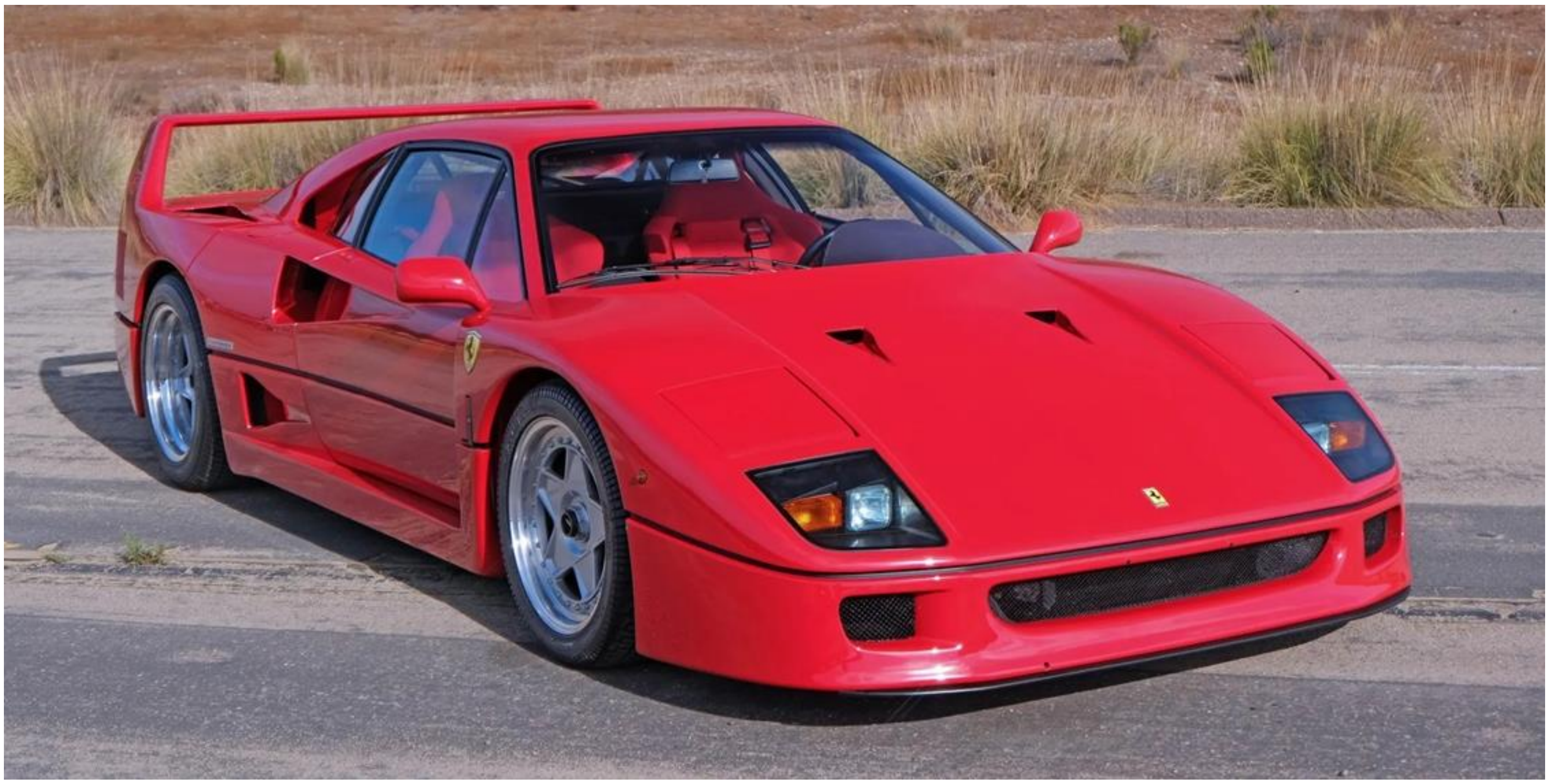
Virtual Dream Car Garage



Timmy Egley First Year Engineering

PURDUE UNIVERSITY®

Table of Contents

References

DeMuro, Doug. “Cars & Bids: Auctions of Cool Modern Cars, Trucks, and SUVs.” Cars & Bids: Auctions of Cool Modern Cars, Trucks, and SUVs, carsandbids.com/.

Kepple, Sarah. “Pygame.” *Teaching Coding through Game Creation*, 2017, pp. 121–42.

Leno, Jay. “Jay Leno’s Garage.” Ultimate Car Care from the Ultimate Garage, www.lenosgarage.com/.

Nonnenberg, Randy. “The Best Vintage and Classic Cars for Sale Online: Bring A Trailer.” The Best Vintage and Classic Cars for Sale Online | Bring a Trailer, bringatrailer.com/.

Sharma, Promod. “Need for Speed Unbound.” Electronic Arts - Official Website, Criterion Games, www.ea.com/games/need-for-speed/need-for-speed-unbound.

van den Burg, G. J. J., et al. “Wrangling Messy CSV Files by Detecting Row and Type Patterns.” Data Mining and Knowledge Discovery, vol. 33, no. 6, 2019, pp. 1799–820

Introduction

Project Overview: This project takes the user on a tour of a virtual dream car garage. The user selects an automotive brand from a list of 20 brands. Then, a list of vehicles from the selected manufacturer is displayed. The user gets to choose one of these vehicles. Once valid inputs have been made, an image of the selected vehicle is displayed. In addition, the exhaust note of the vehicle is played. The user can pick from a list of over 100 vehicles, each of which has a unique picture and exhaust note. From classic American muscle cars to 90s JDM heroes to modern supercars, the virtual dream car garage has it all!

Project Motivation: I have been fascinated by cars ever since I was a kid. From attending car shows to working on my 1990 Corvette ZR-1 project car, I love spending time in and around cars. My passion for cars is also one of the reasons I want to go into industrial engineering: I would really enjoy optimizing and improving manufacturing operations for a major automotive company.

Given that numerous classic cars shot up in value after the pandemic, many owners are storing their vehicles in private collections out of fear that their assets could lose value if they drive them. Unfortunately, this means these vehicles will collect dust, never to be driven again. This is such a shame, as classic cars are works of art that the public should be able to enjoy. This virtual experience cannot replace what it feels like to see a classic car in person, much less drive one. However, it provides an alternative way to enjoy and learn about vehicles that are rarely shown to the public.

Description of Inputs and Outputs

Program Description: My program is structured around data stored in a csv file. My csv file contains four columns: one for automotive brands, one for vehicle, one for image file names, and one for audio file names. Since I imported the entire csv file into a NumPy array, I can reference information from each of these columns as needed throughout the program. To display copious quantities of information, such as the list of all available automotive brands, my program employs for loops to iterate through the relevant column of the csv file until the final piece of data is located. Array indexing is used to locate specific information, such as the audio file of the user's selected vehicle. Utilizing these methods allows my program to work with any amount of data in the csv file. This means that the list of available brands and vehicles can be updated over time without ever needing to update the Python program.

In addition to NumPy, my program utilizes four other imported libraries: Matplotlib, Pygame, time, and os. I used matplotlib's image processing capabilities to display the image of the user's selected vehicle. The title function allowed me to display the year, make, and model of the user's selected vehicle along with the respective image. The Pygame module contains a mixer, which I used to play the exhaust note of the user's selected vehicle. The time library contains a variety of functions that allow Python programs to manage scheduling and organizational tasks. I utilized the sleep function to pause the program at strategic points, giving the user time to read messages printed to the screen. The os library was used to suppress the Pygame welcome message.

Inputs: Other than the imported data from the csv file, the program requires two user inputs to operate properly. After displaying a list of automotive brands, the user is asked to input the name of a vehicle brand. This triggers a function that prints a list of all the vehicles from the brand. Then, the user is asked to input their vehicle choice. This determines the picture of the car and the exhaust note that is played. Both inputs are strings, and the program will not stop running until the user inputs a valid automotive brand and vehicle choice.

Input Validation: My program utilizes input validation techniques to ensure that mistakes on part of the user do not interfere with the operation of the program. For each of the two user defined functions that collect user input, there is a while loop with nested conditional statements. If the user enters an invalid input, be it a brand or vehicle not contained within the collection, an error message is printed to the screen. The program continues to ask for user input until correct inputs are made. This ensures that the user has an opportunity to correct their mistakes and that the program executes with successful outputs.

Outputs: Since my program continues to ask for user input, the program will always display an image and play an audio file for the user's selected vehicle. There are over 100 unique vehicles that the user can choose from. Each vehicle has a unique picture that will be displayed and an exhaust note sound that will be played if the user selects it.

Description of User Defined Functions

Overview: My program utilizes a “top-down” design. This allowed me to break up the larger project into smaller, more manageable components. I have seven user defined functions. Each user defined function is designed to fully complete one task, be it printing a list of available vehicles to the screen or collecting user input. My main function coordinates the execution of the entire program. This ensures that user-defined functions are called in the appropriate order and that all outputs displayed to the user are correct. I found remarkable success using this method.

Print Brands: This user-defined function prints all the available automotive brands to the user. There are two parameters of this function: data and rows. Data is a NumPy array of the csv file, and rows is the number of rows in the csv file. The main component of this function is a for loop that iterates through all the rows in the csv file. Every element in the first column of my csv file that does not contain blank spaces is printed to the screen.

Get Brand: This user-defined function collects input for the users selected brand. The only parameter is available brands, which is a list containing all the brands that the user can pick. The input function asks the user to pick a brand from the printed list of available brands. If the user’s choice is not contained within the available brands list, it is not a valid input. The while loop does not break until a valid choice is made.

Print Vehicles: This user-defined function prints all available vehicles to the screen. Its parameters are data, brand row, and rows. Data is a NumPy array of the csv file. A while loop prints the vehicles stored in the first column of the NumPy array, starting at the row of the selected automotive brand. When a blank space or the final row of the csv file is reached, the while loop terminates. The function also returns the row of the final printed vehicle, which will be used in the Get Vehicle user defined function for error handling.

Get Vehicle: This user-defined function collects the input for the user's selected vehicle. The parameters are all available vehicles, which is a list containing the available vehicles from the selected brand, and all vehicles, which is a list containing all vehicles in the second column of the csv file. To account for user mistakes, this program uses a while loop with nested if-else structures. The first condition checks whether the input is found within the available vehicles list. If the user input if it is valid, it is returned. The second condition checks whether the user's selected vehicle is contained in the all vehicles list. If true, the program implements recursion to restart the program, allowing the user to select their desired automotive brand. The third condition checks whether the user's selected vehicle is contained within the csv file. If it is not, an error message is printed, and the user is asked to make a valid input once again. A vehicle choice is returned only when the while loop is broken.

Row Finder: This user-defined function determines the row number of the csv file that contains the user’s input. There are five parameters: choice, data, start, end, and column. The choice is the user’s validated input, and data is a NumPy array of the csv file. The user defined function will search the specified column of the csv file for the user inputs. A for loop is used to iterate through the specified starting and ending rows. When the user’s inputs are found, the for-loop breaks. The function returns the row that the user input was found. The use of these five parameters makes this user defined function very versatile. It is used two times throughout the program.

Show Image: This user-defined function displays the image of the users selected vehicle. There are four parameters: image, year, selected brand, and selected vehicle. Image is the jpg file handle, which was collected prior to the function call in the main function. Year, selected brand, and selected vehicle are strings containing the corresponding information pertaining to the user’s selected vehicle. A variety of functions from the Matplotlib library are used to read, display, format, and appropriately title the image.

Play Audio: This user-defined function plays the exhaust note of the user’s selected vehicle. The only parameter to this function is audio, which is the wav file handle. Numerous functions regarding the Pygame mixer are used to load and play the audio file. A while loop is used to ensure that the file is played for the full duration.

User Manual

Installation Instructions: Follow this simple step-by-step guide to install this Python program and all supplementary files so that it can properly run on your own device.

1. Create a folder on your PC where all necessary components can be downloaded.
2. Download the following components to this folder:
   a) The "Individual_Project_tegley"
   b) The "Virtual_Dream_Car_Garage" csv file
   c) All 102 wav files (see the "Audio_Files" folder)
   d) All 102 jpg files (see the "Image_Files" folder)
3. Open Visual Studio code or another source-code editor that is compatible with Python.
4. Click on the 'File' button at the top right corner and open the folder containing everything downloaded in Step 2.
5. Run the program.
6. Follow the on-screen instructions and type in all necessary inputs.
7. Enjoy the image and exhaust note of your selected vehicle!

User Inputs: Throughout the program, you will be asked for two inputs:

1) Select an automotive brand from the list printed to the screen
2) Select an available vehicle from the list printed to the screen

All inputs must exactly match one of the displayed options. Everything should be typed into the Python terminal.

Accessibility:

If you make a mistake, no worries! This program uses a variety of input validation techniques to catch instances where invalid inputs are made. Error messages specific to the situation will be displayed in these instances. The program will not display any images or play any audio files until valid inputs are made. This makes the program very user friendly.

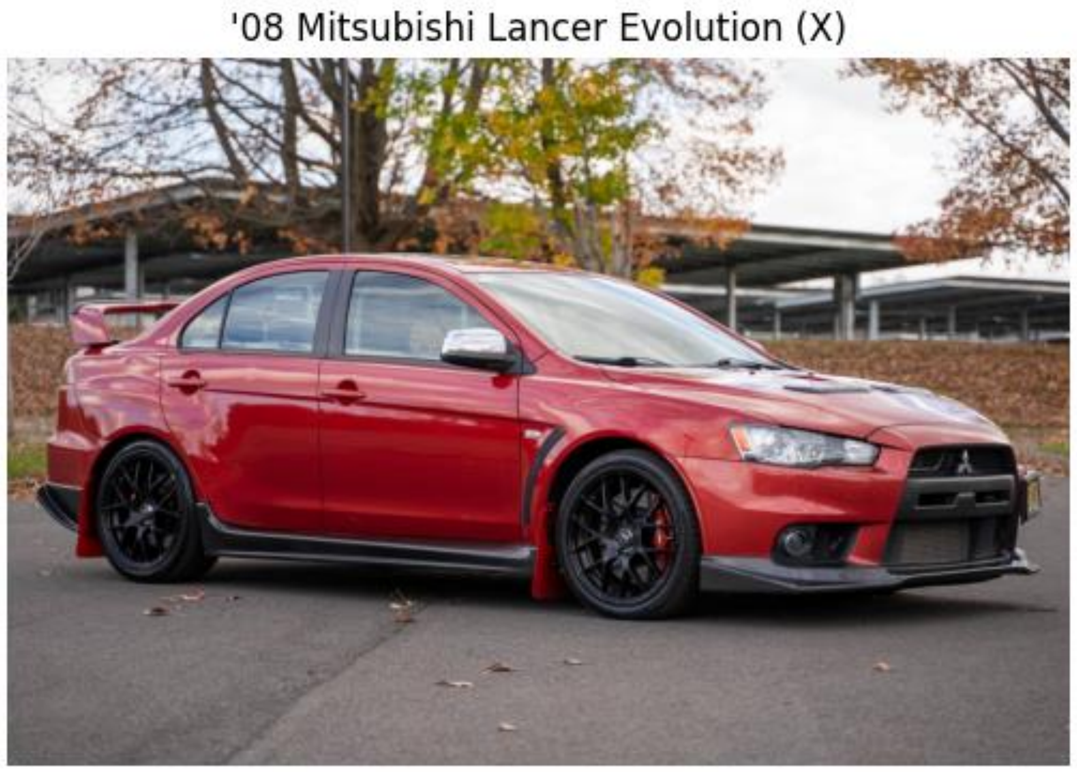
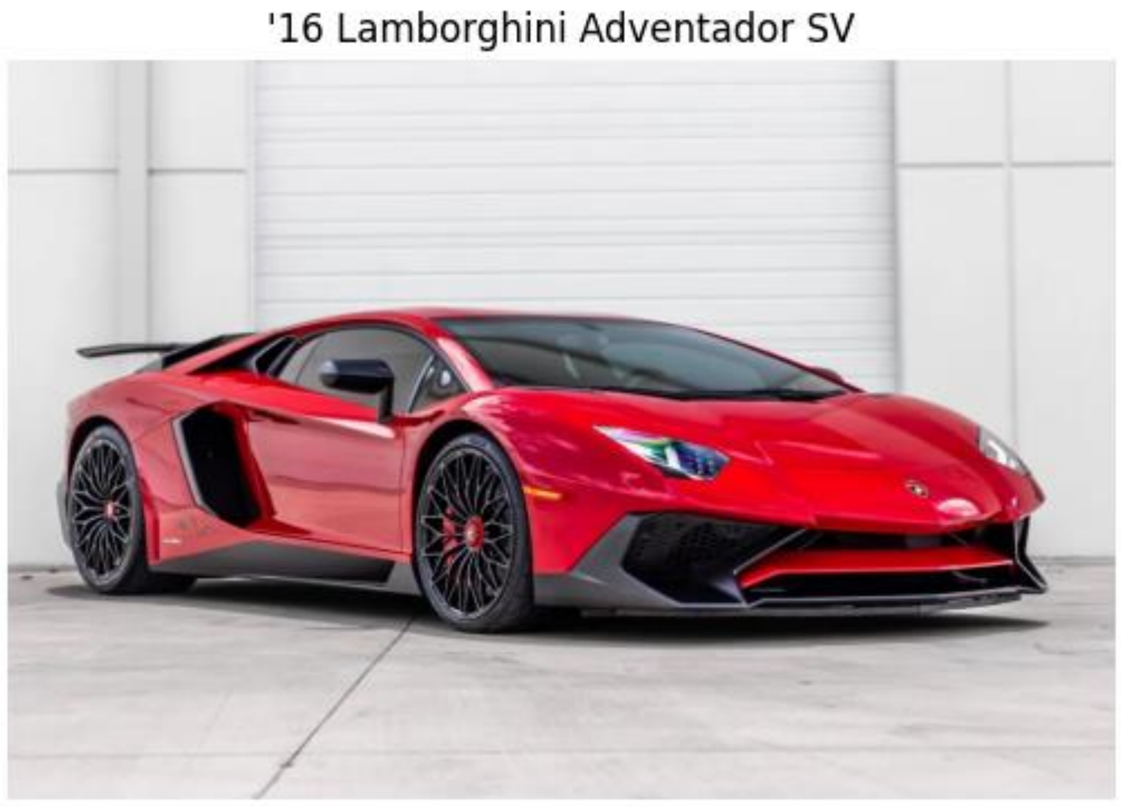
Project Capabilities:

This program has image and audio files for 102 vehicles from 20 automotive brands. While this is a considerable amount, this is nowhere near the total number of vehicles in the world. Due to time consuming process of finding high-quality vehicle images and audio clips, I limited the available vehicles to just 102. When finalizing the list of vehicles, I did my best to include a variety of brands and vehicle types.

My code is designed to work with any number of vehicles and brands stored in Virtual Dream Car Garage csv file. Therefore, if additional vehicles and brands are to be added in the future, one simply needs to update the csv file. My program is quite versatile in this regard.

Image and Audio Sources:

One of the most challenging parts of this project was determining where to find high-quality pictures and audio clips of my cars. I found my pictures using the car auction websites Bring-A-Trailer and Cars & Bids. Each of these sites have professional car photographers that take pictures of customers’ cars. I got my audio clips from the videogame Need For Speed Unbound, which is well known for its large quantity of realistic car sounds.

User Input	Output
Select a brand - > Mitsubishi Pick your car - > Lancer Evolution (X)	**Image Output:**  **Audio Output:** Evo_X.wav file is played
Select a brand - > Lamborghini Pick your car – > Adventador SV	**Image Output:**  **Audio Output:** Adventador.wav file is played

Select a brand - > Chevrolet Pick your car - > Corvette ZR1 (C7)	**Image Output:** **Audio Output:** C7_ZR1.wav file is played
Select a brand - > Dodge Pick your car - > Challenger SRT Demon 170	**Image Output:** **Audio Output:** Demon.wav file is played

Select a brand - > Audi “Unfortunately, the brand you have selected is not a part of our collection. Please select a different brand.” Select a brand - > Mercedes-Benz AMG Pick your car - > SLS	**Image Output:** **Audio Output:** SLS.wav file is played

Select a brand - > Nissan Pick your car - > RSX “It looks like the car you selected is from a different brand.” Do you want to select that brand (y or n)? -> y Select a brand - > Honda Pick your car - > RSX	**Image Output:** '06 Acura RSX **Audio Output:** RSX.wav file is played

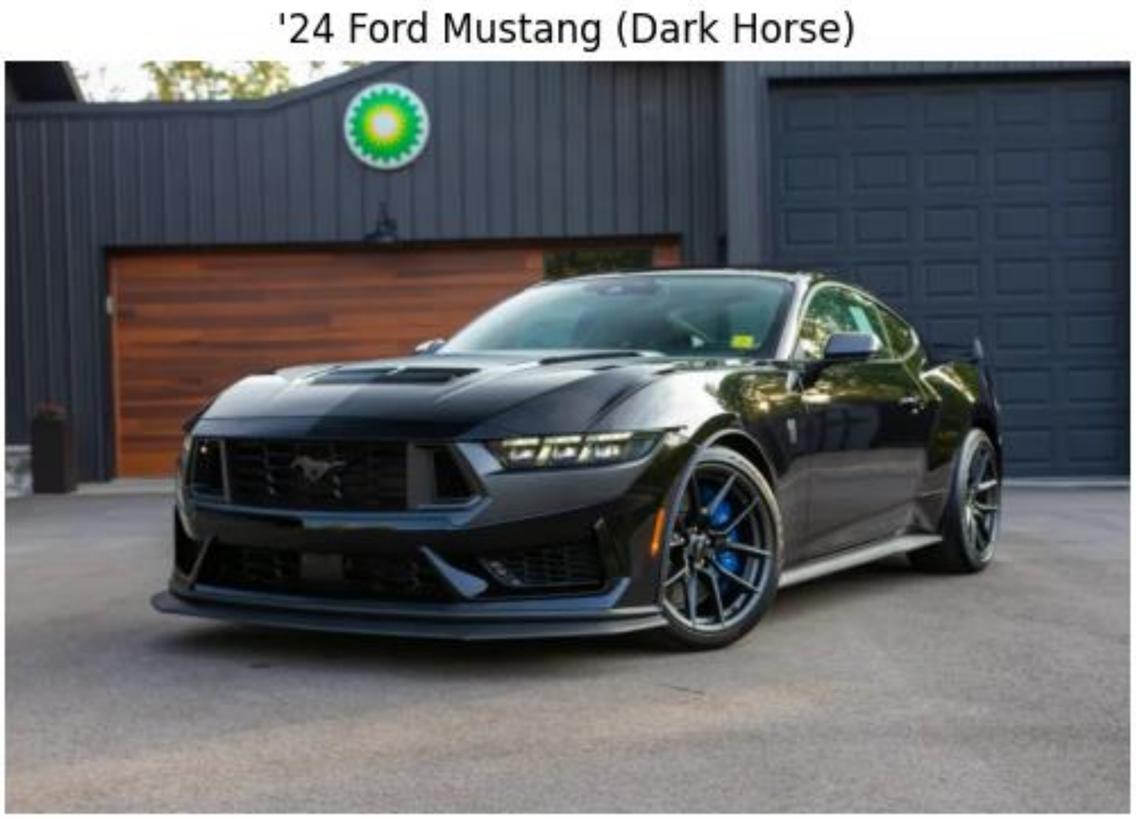
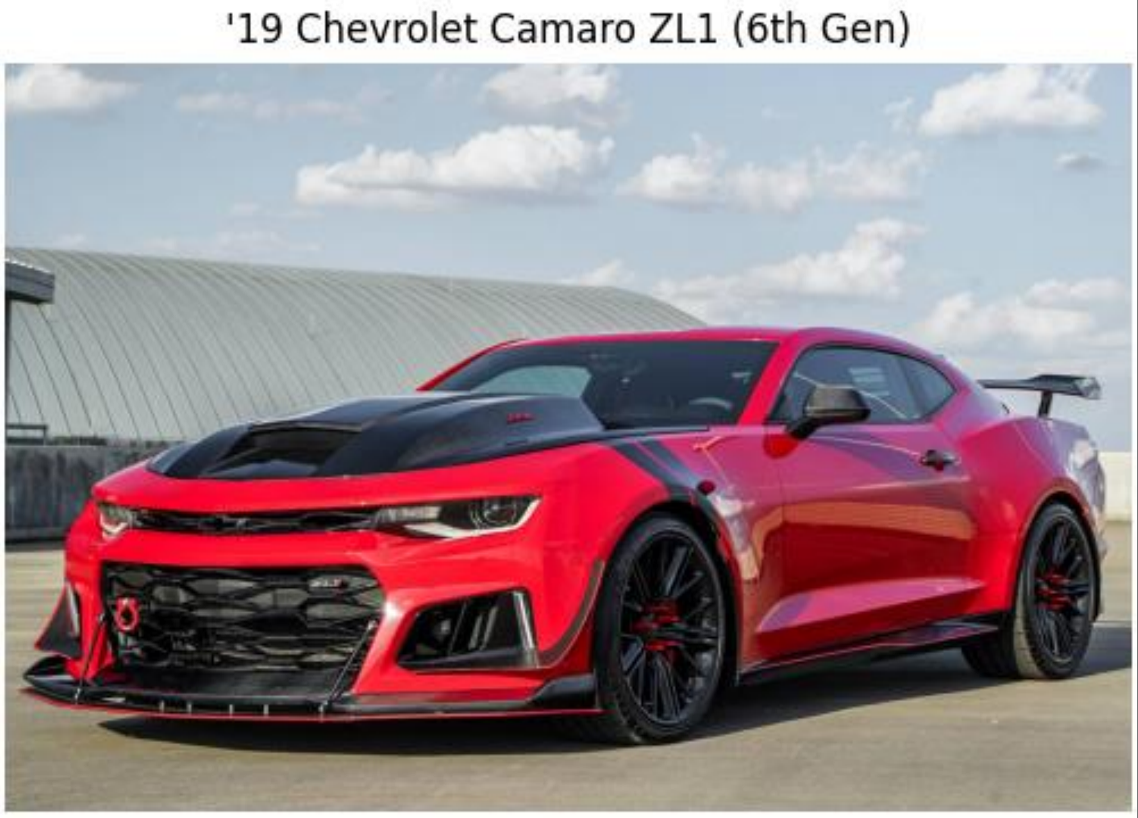
Select a brand - > Dodge Pick your car - > Charger King Daytona “Unfortunately, the vehicle you selected is not part of our collection. Please select one of the vehicles we have.” Pick your car - > Charger SRT Hellcat	**Image Output:** '19 Dodge Charger SRT Hellcat **Audio Output:** Hellcat.wav file is played

Select a brand - > Subaru Pick your car - > WRX STI (2nd Gen)	**Image Output:** '07 Subaru WRX STI (2nd Gen) **Audio Output:** WRX_2.wav file is played
Select a brand - > Mitsubishi Pick your car - > Lancer Evolution (IX)	**Image Output:** '06 Mitsubishi Lancer Evolution (IX) **Audio Output:** Evo_IX.wav file is played

Select a brand - > Koenigsegg Pick your car - > Jesko	**Image Output:** '23 Koenigsegg Jesko **Audio Output:** Jesko.wav file is played
Select a brand - > Bugatti Pick your car - > Chiron	**Image Output:** '23 Bugatti Chiron **Audio Output:** Chiron.wav file is played

Select a brand - > McLaren Pick your car - > P1	**Image Output:** '15 McLaren P1 **Audio Output:** P1.wav file is played
Select a brand - > Porsche Pick your car - > 918 Spyder	**Image Output:** '15 Porsche 918 Spyder **Audio Output:** 918.wav file is played

Select a brand - > Ferrari Pick your car - > LaFerrari	**Image Output:** '17 Ferrari LaFerrari **Audio Output:** LaFerrari.wav file is played
Select a brand - > Lamborghini Pick your car – > Revuelto	**Image Output:** '24 Lamborghini Revuelto **Audio Output:** Revuelto.wav file is played

Select a brand - > Ford Pick your car - > Mustang (Dark Horse)	**Image Output:**  **Audio Output:** Dark_Horse.wav file is played
Select a brand - > Chevrolet Pick your car - > Camaro ZL1 (6th Gen)	**Image Output:**  **Audio Output:** Camaro_3.wav file is played

Select a brand - > Toyota Pick your car – > LFA	**Image Output:** '12 Lexus LFA **Audio Output:** LFA.wav file is played
Select a brand - > Porsche Pick your car – > Carrera GT	**Image Output:** '04 Porsche Carrera GT **Audio Output:** Carrera_GT.wav file is played

Select a brand - > Ford Pick your car - > Shelby Cobra	**Image Output:** '65 Ford Shelby Cobra **Audio Output:** Cobra.wav file is played
Select a brand - > Purdue Pick your car - > Boilermaker Special	**Image Output:** '40 Purdue Boilermaker Special **Audio Output:** Boilermaker_Special.wav file is played (Yes, this is the train whistle!)