|  |
| --- |
| SET09103 |
| Advanced Web Technologies CW2 (JDM CARS) |
| Usman Ashraf - 40096834 |

|  |
| --- |
| 11/30/2016 |

Contents

[JDM Cars Web App Introduction 2](#_Toc465220278)

[Design 3](#_Toc465220279)

[Enhancements 4](#_Toc465220280)

[Critical Evaluation 5](#_Toc465220281)

[Personal Evaluation 6](#_Toc465220282)

[References and Resources 7](#_Toc465220283)

# JDM Cars Web App Introduction

My web app is very simple, however shares a lot of features that the majority of our most favourite, most visited websites have and what we take for granted. Clicking the header takes you back to the home page for example, and the title of each page will take you to the top of the current page that you’re on. **I have extended on this web app from the previous, and have added some key features that I will explain later in the report.**

The web app is simply a catalogue style website, in which I picked a handful of cars from the Japanese domestic market (JDM), and gave a brief summary of each vehicle. On the home page you will see a small paragraph of text explaining the significance of these vehicles and why they are cherished upon every petrol head.

# Design

The web app as said in the previous page consists of seven tabs as listed below:

* Homepage
* DC2
* DC5
* Supra
* S13
* S15
* Chaser
* Map

Each tab other than the homepage has information relative to the tab name, and has a brief summary of the vehicle, along with a couple of pictures to give the user an idea of what the vehicle looks like. This is a very basic design however is extremely clean, and simple to navigate around which was a main objective of mine.

However before I even began to create my html pages and CSS, I created my python flask file so that I could have my web app in a working and testing state. This allowed me to always have the web app running as then I could refresh the page any time I made any changes, to make sure the outcome is what I expected it to be. Once I was confident with my routes on flask and tested to see if my html pages and URL’s all connected, I began coding each page individually, the most time consuming of course being the index (homepage). I designed the page using my existing knowledge of html, ensuring that DIV ID’s are in place to make the styling of the web app a lot easier. Once I had a text and links within my html, I began styling the page with css, creating the tabs and general layout of the app. I also used a background image to give the app a little bit of life and colour. I also uploaded images and used my stylesheet to have them centered and bordered, all using the same DIV ID so that they all behave in the same way, are the same size and are lined up correctly for neatness.

Once the index page and stylesheet were completed to the standard that I wanted them, the rest was fairly simple, I could make copies of the index page and change the titles, and have them all connected to the same stylesheet. Of course I had to add to my index.py python script to add more routes so that my tabs and html pages could be called with relevant URL hierarchy.

My final step was to put in an error handler into my python script, so that the user gets a personalised error message rather than their internet browser giving them a generic failed to load response.

**I have made several upgrades to the previous web app from the last CW. The first being a proper error handling page.**

**Now when you go into a URL that does not exist, the web app has a full template designed the same as the rest of the website, with an error message and a picture of an old beaten up car for humour.**

**There is also a new tab on the web app now called “Map”, which is a full functioning google maps API, and has a pointer to Tokyo, japan, more specifically the Honda Headquarters, which I found was an extremely cool addition to my web app.**

**These were the changes made in the design regard to my web app, the other changes/additions are behind the scenes which will be explained further into this report.**

# Enhancements

As with everything in life, there is always room for improvement and enhancement. In the case of my web app, there were a range of enhancements I could have made.

Adding some javascript and jQuery to my web app would have been great, to allow drop down menus and doing things with my images to allow features such as toggling, for example being able to cycle through images by clicking them, that would have allowed more input from the user which is always a positive thing.

Another enhancement would have been to have a more ‘busy’ web app with a lot more information and content, and perhaps extra html pages so that there can be a proper hierarchy, for example localhost:5000/DC2/specs/pricelist etc.

**I have now added some javascript in the form of the google maps API, however rather than making the web app “look” more busy, I was more interested this time round in making it functionally better, and having additional features outside of the users best interest, and more to do with working on and extending the website.**

# Critical Evaluation

The python flask file index.py is general routing code which must be used for flask to call upon the html pages; the error handling addition is good as it gives a personalised message if an incorrect link is used. However one thing which is generally frowned upon is putting HTML code straight into flask, which is something that I have done. I should have created another html template and rendered it, and perhaps add a little styling too, however my main purpose was to have a very basic error handler, therefor I hard coded the error message right in, it saved me time and got the same result I was after if I was to create another template, therefor in my case I believe doing it the simpler, quicker way was more efficient.

The main thing I wanted to achieve was to have a working web app, and ensure everything such as my stylesheet was working correctly and linked together with my html files, therefor I used absolute references to link them together. However in hindsight, I should have used the url\_for Jinja2 function as by doing this, it would enable me to move the directories around without messing up my stylesheet link. This is more a future proof method than anything else; however it is definitely the correct way to do it.

Finally, my html pages are direct copies of each other with the content being changed, such as the paragraphs, images and headings. Instead of making direct copies, I should have used Jinja2 template inheritance. By doing this, some simple lines of code would mean I do not need to make copies, but instead my second, third, fourth, fifth and sixth could just inherit code from the first page. This would also make it more future proof as if I was to make a change to something on my web app, it would go across all pages automatically rather than having to edit one page at a time. However this is something Learned after implementing the bulk of my work, therefor I did not have time to go in and make my web app more efficient. Either way, from a user standpoint, the app would have looked the same, so not all is lost in that aspect.

**Outside of the design changes in the CW2 web app, you will find two key elements introduced to my upgraded version. One is TESTING, I have carried out ten different tests, which individually check that each html page has been called correctly. This is important as I was previously having issues where the wrong page was being called up, which was due to syntax error, but with such a simple web app it’s not always easy to notice that the same page has been recalled twice, therefor if this error was to happen again, a quick way of seeing what has happened would be to run the test script. It works by going to the appropriate page, and looking for certain data within the page, I used key words (headers) on this occasion.**

**I have also implemented error LOGGING. This only works when you are not in debug mode, as when you’re in debug mode, you don’t necessarily need logging as your errors are displayed right in front of you, and in the real world, web apps are not in debug mode. What the logger does is, if there is a break in the routing and you receive any kind of “internal server error” message when trying to access the web app, a text file is created called errorlog.txt, and within that is the error messages of what went wrong. At the moment of course, my web app has no errors, so I tested that this works by simply creating an error (for example, I made flask calculate 1 / 0), to ensure it works.**

# Personal Evaluation

Due to personal circumstances, I was unable to put a lot of time into this coursework, with that in mind I feel as though I have achieved a lot. I learned even more about html and css, on top of that python and jinja2 were two completely new things to learn, however I feel as though I have picked them up quickly and now in a strong place for the next coursework.

I did not have many issues with flask, as Simons workbook was extremely helpful for this and is an excellent piece of work in regard to teaching beginners the fundamentals of these web app methods.

I learned various Jinja2 techniques, such as the url\_for function for linking my web pages together, and template inheritance although I did not use it for this piece of coursework; I now understand it and will be using it going forward.

My two main issues were setting up my GIT, I was getting various errors however I looked at a couple of tutorials via links on Simons workbook and it gave me a clearer picture of how to set it up and the benefits of consistently pushing and pulling data. The other being issues with my levenix permissions, however email communication with Simon helped me overcome this issue also.

I was capable of achieving a lot more in this piece of coursework; however I feel like I have achieved a solid pass and ticked the majority of expectations and requirements for this module so far.

**Unfortunately this semester has been a challenge for me due to issues I had at the start, causing a big backlog of work. However I feel as though I have learned a lot within this module, and am now compliant with using any kind of linux server, and my knowledge of creating web apps and using python has increased greatly.**

**I am proud of what I have been able to achieve in the time frame I’ve had, Simon was an excellent tutor and the tutorials and work he provided was very clear and enjoyable.**

# References and Resources

*All of the following references are clickable hyperlinks.*

[Advanced Web Technologies by Dr Simon Wells](https://www.dropbox.com/s/k41vw5a49y64nt7/workbook.pdf?dl=1)

[Github's 15 minute tutorial](https://try.github.io/levels/1/challenges/1)

[The open vim tutorial](http://www.openvim.com/)

[Learn Python Tutorials](http://www.learnpython.org/)

[Python reading and interactive tutorials](https://www.fullstackpython.com/flask.html)

[HTML reading](http://html.com/)

[Linux forums](http://www.linuxquestions.org/questions/linux-forums-50/)

[CSS reading](http://www.tutorialspoint.com/css/)

[Google Maps API](https://developers.google.com/maps/)