Advanced Web Technologies CourseWork 1

Sonas MacRae
40277542@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09103)

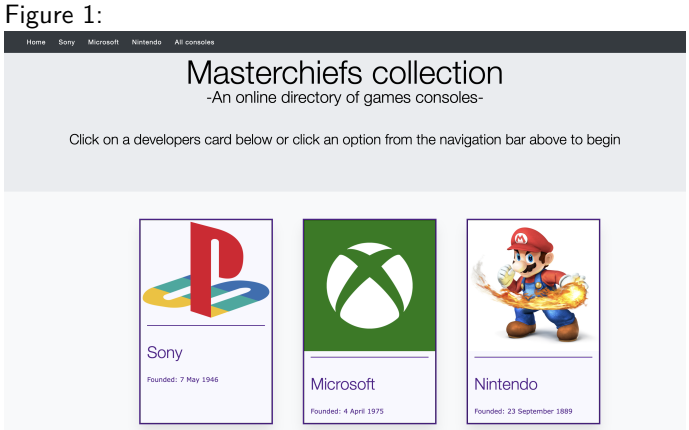
1 Title

Masterchiefs Collection

2 Introduction

This web-app is a directory of popular games consoles dating back to the 1990's. This project was written using HTML, CSS, Python, Jinja2 and JSON. The web app contains information about each console, including a review, with each console having its own profile page. The user is able to search through the console directory by developer or by the name of the console its self. On top of this, each developer have their own profile page that includes some trivial information about the company as well as displaying all of the games consoles of theirs that feature on the site. The web-app includes a page that displays all of the games consoles displayed as cards, the user is able to click on any of these cards which will redirect them to their respective profile page. The user is able to view all of the games consoles, or use the drop down menu present on the page which has some filter options.
The screen shot below shows the design of the homepage, it shows the navigation bar at the top, the section which contains the title and some information about how to use the website. The screen shot also shows how the cards are laid out and how they show an image as well as some basic info about the subject, in this case the cards are links to the developers pages.

Figure 1:



3 Design

This web-app is structured the way it is so it can utilize the dynamic generation of HTML pages using data from JSON files. The web-app contains information about games consoles manufactures and some of the games consoles they manufactured. For example from the home page a user can navigate to a manufacturer, where they will be presented with a list of games consoles developed by the company as links to the games consoles profile. This is easy to digest and use and for that reason it was implemented as the URL hierarchy design of the web-app. The idea was to have a card for each games console and manufacturer which held some of the information such as the name and date of release/date of foundation and a picture of the logo, these cards acted as a link to the subjects profile page. Another reason why the website was designed the way it is, is because of how easy it would be to expand it, it can be easily scaled. If another layer to the URL hierarchy were to be added, all the work that would be needed to be done would be creating another JSON file, an HTML template page and one function in the Python file as well as linking up the new pages to the site in appropriate places.

The manufacturers profile pages contained cards for all of the games consoles they produced that feature on the site as well as some trivial information about the developer its self. Furthermore, each console profile page contains a link back to its respective developers page, for example the PlayStation page contains a link back to the Sony profile page.

Each page on the site has the same navigation bar, this bar includes links to the home page, a link to each of the three developers pages as well as a link to a page which displays all of the consoles. Every page on the website is designed to allow the user to easily navigate deeper or shallower throughout the URL's without trouble. This was implemented by having a HTML file which contained the navigation bar that was extended to every page.
Below shows the basic URL structure and then how the user would be able to navigate throughout the website (without considering the navigation bar):

Figure 2:

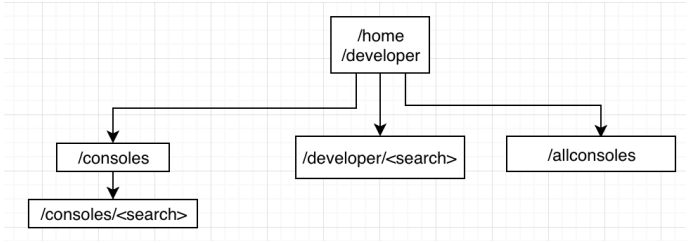
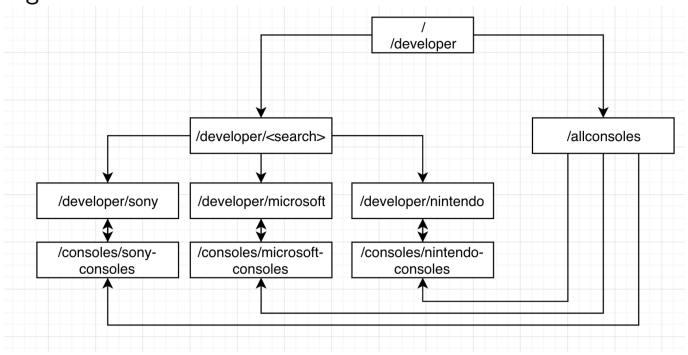
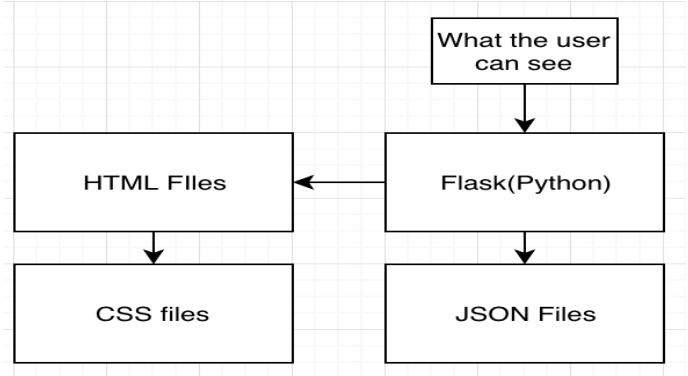


Figure 3:



It appears that the user is "stranded" when they navigate to either a developer page or a games console page when observing Figure 3, this was the motivation to include the navigation bar on every page, allowing the user to short cut to key pages on the site, this prevents the user from being stuck at a page if they aren't familiar with how URLs work. Each page follows the same distinct design, this is done by using HTML and CSS to split up the pages into a navigation bar at the top, followed by a header section then a content section below this. The reason behind laying out the pages like this is so that the same design works for any page, as all of the pages on the site require a navigation bar, a header section and space to display all of the relevant content. The idea was to cut down on the amount of code for this project, this is why templates were used. Based on the URL, the Python file would fetch the relevant data from the JSON file, filter it if necessary, and render an HTML file passing in a list of object(s), with each HTML template page calling a CSS file for the styling and inheriting any commonly used elements from another HTML file. Figure 4 below shows how data is moved around the files, the arrows represent where the file fetches data from.

Figure 4:



4 Enhancements

Features that could be added to this web-app would include adding a forum which will allow users to discuss their favourite games consoles and topics that circulate these games consoles. This would enhance the user experience as they would feel involved with the site instead of just reading the information provided. There would have to be admin-users to be able to monitor the comments, the admins would be able to delete any comment as well as suspending/deleting accounts off the system.

The profile pages for the developers and games consoles could include more information, they could go into more detail and include more images. The website could include a section which displays up-to-date news articles about current games consoles to keep users informed with the gaming world. On top of this on the page which displays all of the consoles could have more filter options, for example a filter which orders the consoles from highest sales to lowest sales, or one which orders them by age. Although it can be said that these features aren't so necessary considering how well the consoles are split up between the developers and how easy they are to find. A search bar could be implemented to allow users to search for consoles, developers or games, the information related to the search could have all been displayed on a results page with links to the profile pages that are relevant to the information returned.

This coursework doesn't feature any JavaScript, using JavaScript would make the site more interesting, allowing it to be more responsive to user input, an example of this could be changing the colour of the page depending on the developer/console page that the user is present on. The colours could match the systems/companies colours, this was a design feature of this coursework that wasn't implemented due to a shortage of time. The home page of the website could have had a section which could contain a slide show of consoles that feature on the site. Every video games console on the website has its own boot up sound, a sound clip could be added to each console profile page. JavaScript could also be used to add a setting which would translate all of the text on the website into a different language, allowing the website to be read by more people, attracting a larger audience.

In terms of design, it's clear that more effort could have been done with the HTML/CSS side of this project, a perfect example of this would be that the site doesn't feature a footer and that every page adopts the same colour scheme. Elements such as the drop down menu and the "back to developer" button look cheaply designed. On top of this there is little variety between the pages, all of the profile pages adopt the same layout and same colour scheme, the difference between them is only the content, this was a very efficient way to build the website using JSON files to store almost all of the data used on the site, yet it can be seen that this isn't the most appealing way of presenting the information as the website eventually gets boring with every page looking very similar. Simply adding more colour to the website would make it look more attractive, it could have gone further as to make more elements stand out from the

surroundings as only the "back to developer" button and the drop down filter stand out compared to the rest of the pages.

5 Critical evaluation

This web-app is a directory of games consoles and the companies that developed them. The user is able to navigate through pages that contain information about the games consoles and the companies. A feature that works really well is the filter system which allows users to view all of the games consoles or filter them by manufacturer. This feature is located on the page that displays all of the consoles on the website as a default. This is very easy to use as well as being fast and responsive. Each games console profile includes a link to an in-depth review, allowing users to further research the respective topic. This works well as it doesn't limit the educational element of the site to just the content displayed on the pages. The navigation bar appears on every page of the site, this is a good feature to have as it allows the user to quickly re-direct to any of the pages that feature on the navigation bar, if a page didn't have an obvious way to return to the rest of the site then the user might get confused and refuse to use the website anymore. The console and developer profile pages aren't as appealing as they were intended to be, although they do display all of the information in a consistent fashion which is clear and easy to read.

The layout of all of the pages are consistent, they all have a navigation bar at the top, a section for the header, where the title of the page and some information is displayed, then the body where text and cards are displayed. This is executed very well and has no issues as the style works for all of the pages. All of the cards are the same size and do not disrupt the layout of any of the other elements, in fact none of the elements on any of the pages are out of place or cause a disruption, this shows that the website is designed very well.

When a 404 error comes up, the user is redirected to a custom error page, where the navigation bar is still present as well as a big button which redirects the user back to the home page. This is implemented well, it does its job and has no issues. The best feature of this website is the fact that the user is able to search for consoles or developers with the URL bar, for example if they were to search "/developer/Sony" then a page with all of the information relating to Sony would render. This works because the Python file searches the JSON file for a developer object with the name of "Sony" and combines this object with any console objects found in the JSON files whose developer matches the name "Sony". If a search returns empty then the user is redirected to the error page mentioned previously. There are both benefits and side effects to storing the data between many JSON files, the major benefit is that the JSON files do not need to be opened and read until required, whereas if all of the objects were nested then the JSON file would have to be opened and read much more often, and certainly with larger websites the search would take longer as it would have far more data to filter through. This is the reason why behind this projects data was split up between three files, another reason is that if a console would have two separate developers, then nesting the same game console in two separate developer objects is less efficient than having two "developer" values for a console object.

Throughout the site, the use of horizontal lines are used to split up content, for example between the images and the trivial information on the cards, and between the sections on the profile pages where the content from one JSON file ends and the next one starts. This is a great feature as it spits up the content making it easier to digest.

In terms of the code that was written to create this website, it can be said that it is very well designed, there are few HTML pages, very few CSS pages, the Python file doesn't feature repeated code and is commented. The code inside all of the files is indented properly and the Python file features comments. The interaction between all of the code is very efficient and has been carried out to a high standard. The data is split up appropriately between the JSON files and feature Primary and Foreign keys so that the search functions in the Python file can be carried out effectively.

When a template is rendered, objects are passed in using lists, Jinja2 is used to display the contents of the objects, figure 4 below shows how this was done. The pages make good use of the objects attributes, an example of this appears on the console pages, where there is a button to back-track to the corresponding developers page, this is achieved by calling the function from the Python file which is used to render the developer page and passing in the "developer" value of the console object as the parameter, this redirects the user to the correct developer page. This has been tested and works perfectly for all the console pages.

Figure 4:

6 Personal evaluation

From this coursework I have greatly improved my knowledge of the Python programming language, this was the second project I have attempted using Python so at the start it was difficult to visualize how I would solve the problems I was going to face. After starting this coursework I carried out some online Python tutorials to make it easier to complete the coursework. Having a basic understanding of the language and how the syntax worked allowed me to attempt more advanced and efficient methods of creating web pages. By the end of the coursework I felt confident using Python to build web apps.

While I had the option to use Bootstrap, and although I am capable of using Bootstrap to style a website, I opted against this idea and decided to style it myself, although the design is inspired through observing many websites, none of the code was stolen, the font for example, 'Helvetica Neue' is fairly common, I just softened the font-weight to give it a new look. One of the major hurdles I had to get over when completing this project was learning how to use Vim and how to deal with the Universities server to host the website. Often the content displayed on a browser wouldn't update, I would have to wait and this would waste a lot of time, to design the pages of my site I would create CSS and HTML files locally as I was able to view the results instantly. It was only towards the end where I had very nearly finished the site that I had learned that by pressing cmd+shift+r on Google Chrome would the content update properly, which then allowed me to implement the design with the content from the JSON files and test the search functions.

My submitted project was my second draft, with the first draft I started manually writing all of the HTML files, though, with the intent of having lots of pages with some duplicated content throughout the web-app I came to the conclusion that this method was simply too time-consuming and this time could be better spent implementing more features and designing the pages to look clean and more appealing. I became familiar with Jinja2 and JSON, I learned that I could retrieve a list of objects from a JSON file using Python, then return this list to a rendered HTML page which looped through the object-list displaying elements of the objects where the Jinja2 tags were placed.

The pages that contained the cards were sized automatically depending on the number of cards, they were placed inside a grid with three columns. This worked perfectly after I had to solve the issue of figuring out how to display any number of sections of content onto a certain page. The thought of online shopping was the inspiration to use the cards, the way that items are presented in card-like sections with a photo of the product as well as some information below, for example the price, size or name of the product lead me to research how this was done. The next problem was working out how to display these cards in an orderly fashion and be able to display as many of them on a page as I had liked, I would like to give credit to w3schools as it was their website where I learned how to implement a auto-sized grid system from which I could display the cards. This practice is used throughout the website.

A problem that I had to deal with was rendering the console profile pages and the developer profile pages, for the other pages of the web-app I only had to pass in one object-list to display the data. The console profile page displays data from two separate JSON files, the consoles.JSON and the games.JSON files, retrieving the correct object from both was a challenge I couldn't solve easily. It was from my experience of working with databases in the past that I came up with the bright idea of assigning ID's to objects in the JSON files, where the ID of a "game" in the games.JSON file matches the name of a "console" in the consoles.JSON file. The same principle was applied between the developers.JSON and consoles.JSON file too where the developer of the "game" object matched the name of the "developer" object. The Python code I wrote to solve this problem is showed below:

Figure 5:

```
# Page to display one console in more detail
@app.route("/console/<search>")
def console(search):
        game = []
        console = []
        for c in consoleInput:
                if c["name"].lower() == search.lower() or c["ref"].lower() == search.lower():
                        console.append(c)
        for g in gameInput:
                if g["id"].lower() == search.lower():
                        game.append(g)
        if len(game) == 0 or len(console) == 0:
                abort(404)
        return render_template("consoleprofile.html", console=console, game=game)
```

From Figure 5, the variables 'consoleInput' and 'gameInput' are defined at the top of the Python file where the data from the JSON files are read into lists. This function also shows the efficiency of the Python file, as there is only one function per HTML page, this is very cleverly done and shows off my understanding of Python and Flask and I feel I did very well in doing this. The input from the JSON files are shown in Figure 6 below:

Figure 6:

```
# Loads the data from the JSON files
consoleData = json.load(open("data/consoles.json"))
developerData = json.load(open("data/developers.json"))
gameData = json.load(open("data/games.json"))

# Accesses a certain array from the JSON files that were loaded
developerInput = developerData["developers"]
consoleInput = consoleData["consoles"]
gameInput = gameData["games"]
```

And the way that the information is presented is shown in Figure 7 and Figure 8, they are two halves of the same page:

Figure 7:

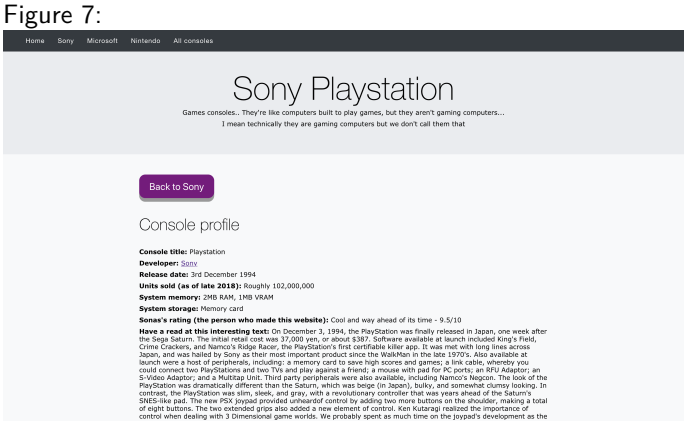
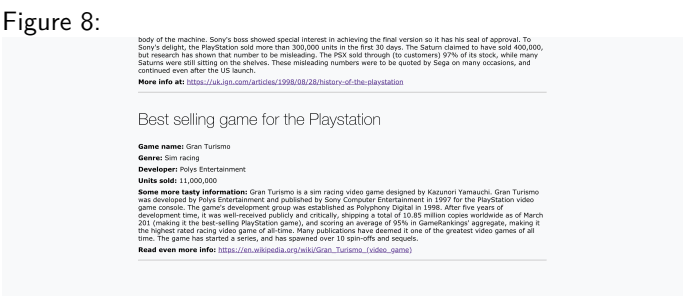


Figure 8:



Although it is evident that more work could been done on

the web-app, time was short because I decided to start the project over with just over a week before the deadline, I would have liked to add another level to the URL-hierarchy, this would be just to show my understanding of Flask and JSON. Overall I feel that this project is certainly above average, I have impressed myself and I am very happy with the outcome. The website has a professional look and feel and the way that it was constructed is even more impressive.

7 References

Below are links to all of the images that were downloaded and used throughout the website, there is one image per developer and one image per console, the links will be labelled by the name of the console/developer they correspond to:

Sony-Link to Sony image

Microsoft-Link to Microsoft image

Nintendo-Link to Nintendo image

Playstation-Link to Playstation 1 image

Playstation 2-Link to Playstation 2 image

Playstation 3-Link to Playstation 3 image

Playstation 4-Link to Playstation 4 image

Xbox Original-Link to Xbox Original image

Xbox 360-Link to Xbox 360 image

Xbox One-Link to Xbox One image

GameCube-Link to GameCube image

Wii-Link to Wii image

Wii U-Link to Wii U image

Nintendo Switch-Link to Nintendo Switch image