Project Diary

Nathan O'Donnell

People often make consistent superficial judgements of a strangers’ personality and life from their appearance. While these judgements can often be very inaccurate, the fact that many people can feel a similar way is interesting and can be used to reveal cultural bias and measure the limits of human intuition. The goal of this project is to create a realistic ‘character sheet’ as a json data structure that defines a person and their appearance in an image. By getting crowd workers to label fictional and real people we can analyse how people make judgements about others and how accurate and consistent such judgements can be.

GitHub Repository: <https://github.com/nathan0donnell/character-sheet>

Table of Contents

[Week #0 - Prologue 5](#_Toc70000229)

[Week #1 5](#_Toc70000230)

[Saturday 16th January 5](#_Toc70000231)

[Tuesday 19th January 6](#_Toc70000232)

[Wednesday 20th January 7](#_Toc70000233)

[Meeting Notes: 7](#_Toc70000234)

[Thursday 21st January 9](#_Toc70000235)

[Friday 22nd January 9](#_Toc70000236)

[Week #3 10](#_Toc70000237)

[Monday 25th January 10](#_Toc70000238)

[Tuesday 26th January 11](#_Toc70000239)

[Wednesday 27th January 12](#_Toc70000240)

[Meeting Notes: 12](#_Toc70000241)

[Thursday 28th January 13](#_Toc70000242)

[Week #4 13](#_Toc70000243)

[Monday 1st February 13](#_Toc70000244)

[Tuesday 2nd February 15](#_Toc70000245)

[Meeting Notes: 15](#_Toc70000246)

[Wednesday 3rd February 15](#_Toc70000247)

[Thursday 4th February 16](#_Toc70000248)

[Friday 5th February 17](#_Toc70000249)

[Reasoning for Each Page 17](#_Toc70000250)

[Sunday 7th February 18](#_Toc70000251)

[Week #5 19](#_Toc70000252)

[Monday 8th February 19](#_Toc70000253)

[Tuesday 9th February 19](#_Toc70000254)

[Wednesday 10th February 20](#_Toc70000255)

[Meeting Notes: 20](#_Toc70000256)

[Friday 12th February 20](#_Toc70000257)

[Week #6 20](#_Toc70000258)

[Monday 15th February 20](#_Toc70000259)

[Tuesday 16th February 21](#_Toc70000260)

[Wednesday 17th February 22](#_Toc70000261)

[Meeting Notes: 22](#_Toc70000262)

[Saturday 20th February 22](#_Toc70000263)

[Sunday 21st February 22](#_Toc70000264)

[Week #7 23](#_Toc70000265)

[Monday 22nd February 23](#_Toc70000266)

[Tuesday 23rd February 23](#_Toc70000267)

[Wednesday 24th February 24](#_Toc70000268)

[Meeting Notes: 24](#_Toc70000269)

[Thursday 25th February 24](#_Toc70000270)

[Week #8 25](#_Toc70000271)

[Monday 1st March 25](#_Toc70000272)

[Tuesday 2nd March 26](#_Toc70000273)

[Wednesday 3rd March 27](#_Toc70000274)

[Meeting Notes: 27](#_Toc70000275)

[Thursday 4th March 27](#_Toc70000276)

[Week #9 32](#_Toc70000277)

[Wednesday 10th March 32](#_Toc70000278)

[Meeting Notes: 32](#_Toc70000279)

[Thursday 11th March 32](#_Toc70000280)

[Week #10 34](#_Toc70000281)

[Thursday 18th March 34](#_Toc70000282)

[Sunday 21st March 34](#_Toc70000283)

[Week #11 35](#_Toc70000284)

[Monday 22nd March 35](#_Toc70000285)

[Tuesday 23rd March 35](#_Toc70000286)

[Wednesday 24th March 36](#_Toc70000287)

[Meeting Notes: 36](#_Toc70000288)

[Sunday 28th March 36](#_Toc70000289)

[Week #12 36](#_Toc70000290)

[Wednesday 31st March 36](#_Toc70000291)

[Easter Break 37](#_Toc70000292)

[Sunday 11th April 37](#_Toc70000293)

[Monday 12th April 38](#_Toc70000294)

[Sunday 18th April 38](#_Toc70000295)

[Tuesday 20th April 39](#_Toc70000296)

[Wednesday 21st April 40](#_Toc70000297)

[Thursday 22nd April 40](#_Toc70000298)

[Final Thoughts 41](#_Toc70000299)

# Week #0 - Prologue

My project has been confirmed as my first choice of “real-world character sheet”, I am not sure where to start so I will look at the basics of Wikidata and how to query it.

I looked for a video on YouTube to explain what the Wikidata API can do. The [video](https://www.youtube.com/watch?v=j5KwP4ifHWs), by Indian Pythonista, helped show how simple it was to query Wikidata.

I wanted to do something over the Christmas break, and I thought starting to learn Python would be most beneficial to me as most areas of computer science make use of it. I’m glad I get to practice it over the course of this project.

# Week #1

## Saturday 16th January

To begin I should list the characteristics I want to be able to take from an image of a person. Basic characteristics such as:

* Race – by continent or similar such as Russia or otherwise.
* Sex
* Age
* Height (?) – could be an issue if there are no objects around the person to reference but as a back-up, we can use an average height from the country they are from.
* Weight (?) – calculated from height, back-up would be like the height back-up.

More advanced characteristics would be more difficult. This data could be taken from assumptions based on the basic characteristics, stereotyping the specific group that they fit into, hopefully removing as much discriminatory bias from the system as possible. I think discriminatory bias is a less accurate way of analysing a person. Examples of advanced assumptions may include:

* Religion
* Sexual Orientation
* Political Views
* Education
* Personality Characteristics:
  + Aggressive
  + Relaxed
  + Strong
  + Weak
  + Funny
  + Insecure

I have an idea to create a google form, asking participants the assumptions they make on people just by looking at them and if they know, an estimate of the percentage of time they’re correct about the assumption. I can send the form to friends and family or anyone else to get as many responses as possible from different people.

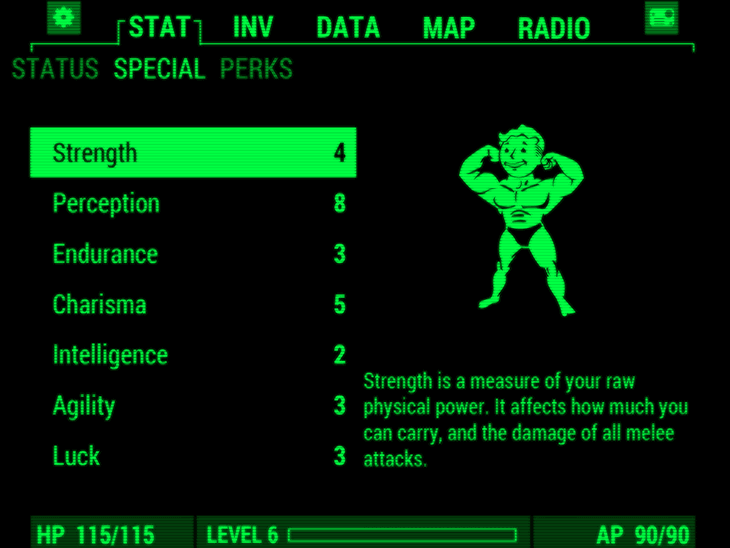
## Tuesday 19th January

So, it’s a few days after making the form and giving the link to around 40 people….

I’ve gotten 6 responses. But all the results are similar, most assumptions having the same level of success. Most of the responses to the age and sex questions were higher at around 70%-90% success, while race assumptions are less so, at around 50% success. This gives me a target range of success in my assumptions. For the advanced assumptions section, the results were much more split.

The wealth question was interesting however, as all responses indicated the participants were successful most of the time at 80% or higher. I think the reason for this is a purely material-based assumption, meaning they assume based on more expensive clothing, better quality make-up or brand-name possessions. If I use web scraping to find a person’s clothing on the internet and compare the brand names to the most “hype-beast” brands, such as Supreme or Off-White, this could be a way to determine the wealth of a person.

Obviously, expensive clothing does not mean you are extremely wealthy but I’m unsure if the data in my character sheet is supposed to be what humans assume or if it should be accurate. For example, women with piercings are perceived to be less religious and more creative[[1]](#footnote-1) but this is how they are perceived and doesn’t necessarily mean that the perceptions are correct. If it is based on human assumption, I can use psychological studies on perceptions of people in my algorithms. This would be very useful in the personality section of the character sheet.

I have also had a few ideas for the appearance of the personality section such as The Sims (specific traits) or Fallout (character points):

I will do a formal plan of what I want soon but right now I am just exploring some ideas.

## Wednesday 20th January

## Meeting Notes:

* Don’t need to design project around AI system.
* Aim is to capture what humans think by looking at someone.
* Predict how people behave with others and how they behave themselves – such as a possible romantic partner.
* I focus on the properties of people not imaging – Dean starts imaging and gathers diverse pictures of people.
* Python code using json files take JSON file and produce web form that when submitted changes the JSON file.
* life behaviour would be formalised like a CV - CV of romantic history.

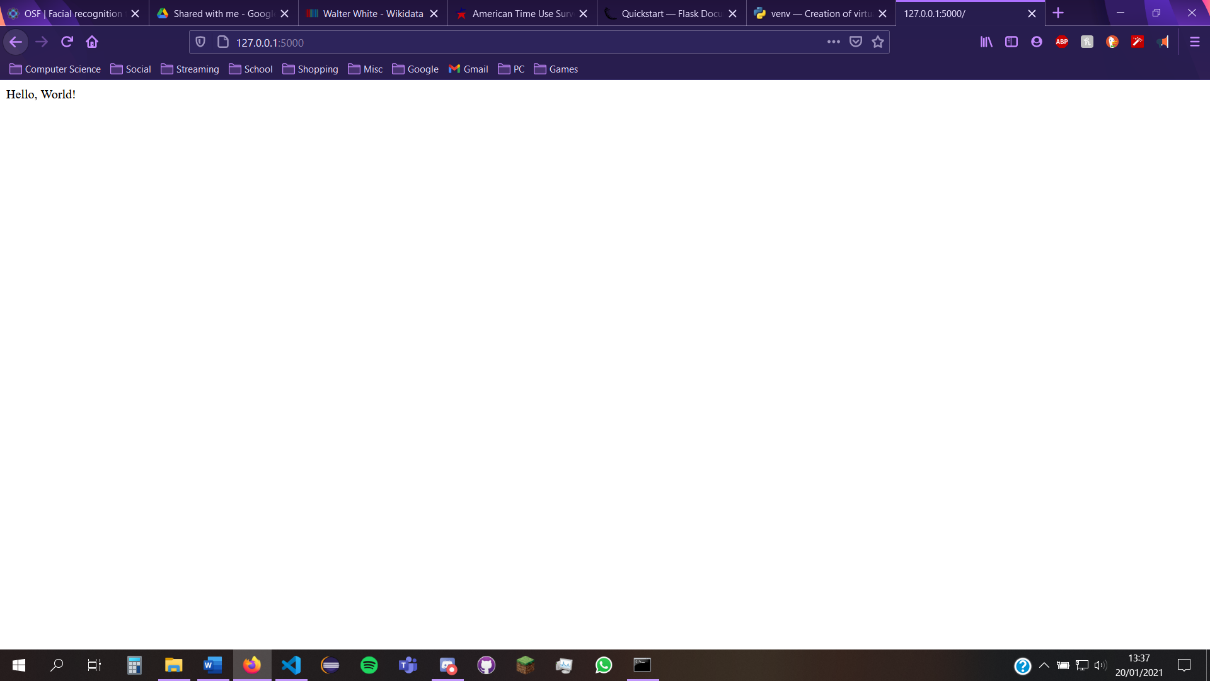
Technical Task:

So, the technical task for me for this week is to create a web server using Flask, and create the Python code that creates a form, inserts the JSON data into said form and be able to edit the data, changing the contents of the JSON file. Very easy!

To begin on this, I’m going to install Flask and try to learn the basics of it. Installed Flask onto the venv and I am currently testing that it is installed with the test code provided by the Flask guide.

I ran into some problems changing the interpreter for VS Code to the virtual environment, but the problem was mostly that there were multiple venv folders of slight name variations which was the reason some had the flask package and others did not.

The code worked!



Doing this, I had a few import problems, but this was only because I was in the wrong directory in the command prompt, but this was an easy fix. Next, I am going to try and find a way to display and edit a form for the website, I will tackle how the data and JSON conversion works after that.

I’ve figured out the HTML file import thing, all you need to do is create a folder called template in the same folder as the application, then use the render\_template function, imported from Flask, then display by calling the function with the name of the HTML file as a parameter.

I am now going to move on to parsing (not sure if I’m using the phrase correctly) the JSON file, looking for a character’s name and the short description of them, I’ll look online for short instruction video on this.

I figured out how to navigate the JSON file, the way the character files are structured means you have to navigate through arrays of arrays of arrays, etc. This means just to get a name you have to say:

print(character['entities']['Q838512']['labels']['en']['value'])

This could be difficult to navigate if this structure varies across the website, but I’ll trust in Wikidata until I encounter this problem.

So far, I’m really enjoying getting a challenge and feel more confident in my ability to do the project well. I’m going to try display some information for a few characters on the website, but I will probably leave writing to the JSON file until tomorrow unless it’s really simple.

I’ve been able to retrieve the name, description, and aliases of a character. This can be displayed in a clear format. I’ve been able to convert the file contents of a JSON file into properties of a Python file – with added exception handling for when there are no available properties such as description. I have tested this on 3 different JSON files. I found a way around the character ID problem by creating a function which finds the unique ID of a set of data.

Going to finish up for today by uploading my work to the GitHub repo along with some descriptions. Tomorrow, my aims are:

* Create a Character class
* Gather more JSON files of characters
* Test more characters
* Display character details on web server

## Thursday 21st January

A character class shouldn’t be too hard so I will do this first, I have been putting all my focus on this module so I should probably not spend too much time on it today.

I made changes to the json\_to\_property code because returning “N/A” was much shorter and makes it easier to find errors. This means I need to change some code in json\_testing and in the Character Class, which has a basic structure now. I also left comments for future me to implement as actual code. I will download more JSON files of characters from the Google Drive folder so i have a larger sample size. I aim for about 10 characters total as I don’t want to spend too much time on that, as I view it as non-important.

I got more characters, cleaned the files up format-wise and tested them. All seem to be working but I want to find an easier way to access the file path of each file in a folder, I imagine there’s a for loop involved but i think it overcomplicates the whole thing. When I need to access a lot more data, however, I will need to find a solution to this.

I’m going to leave the HTML translation of this until tomorrow, but I have a general idea of how to go about it. I’ll update the GitHub repo now then I’ll be done for the day.

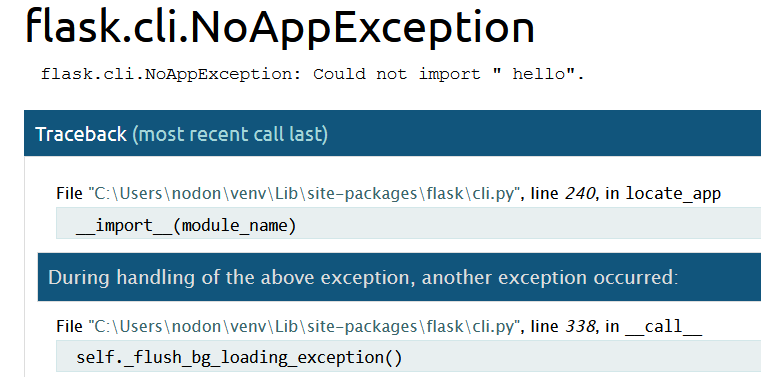
## Friday 22nd January

I probably won’t be able to work on the project much over the weekend, due to personal reasons, but I am going to try get as much done today as I can:

To Do:

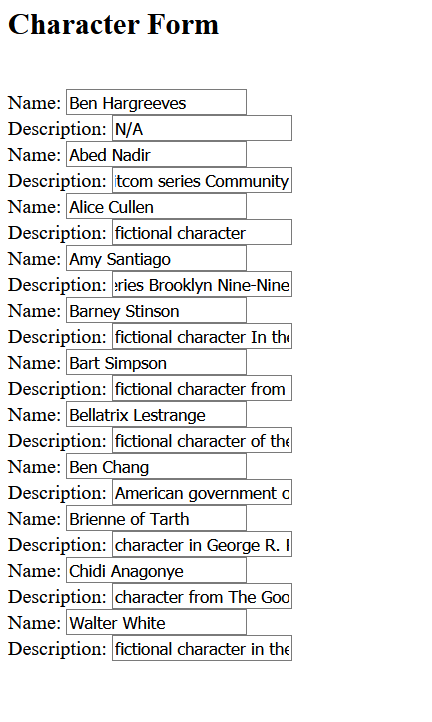
* HTML translation
* Put code onto server and be able to read it
* Start the writing to JSON file

I’ll start with the HTML translation. The idea is to have the same basic frame of code for all characters with varying values for name, description, and aliases.



I’ve been able to convert the JSON files into HTML code however, I am now having problems running the flask server with hello.py.

I resolved the issue by setting FLASK\_APP to the absolute file path rather than the relative file path.

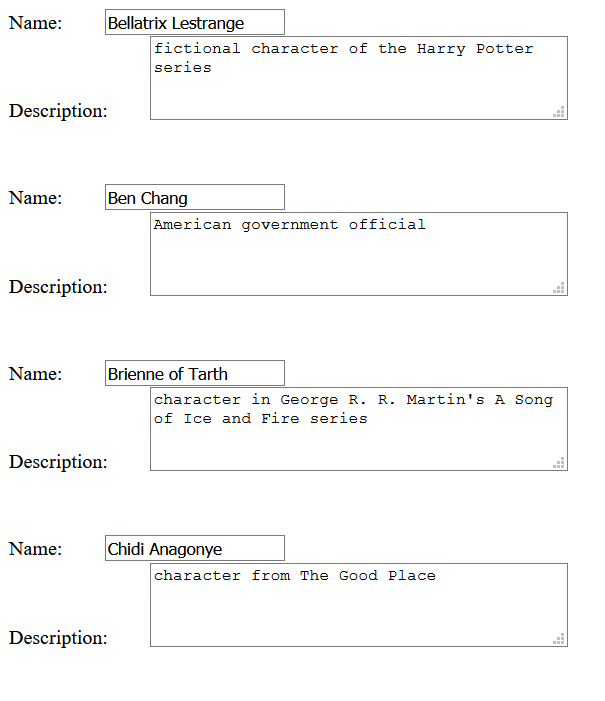
The HTML code runs on the site, however there are some small issues to fix up such as text boxes being too small:

This is just a formatting problem so I will just increase the size of the text boxes to fit the text. However, I will do this on Monday as I have run out of time today.

# Week #3

## Monday 25th January

Continuing on from Friday’s work, I will fix the format of the code. Just by making the length of the text box larger as the appearance is not important.



I decided to change the element of the description to text area rather than text box. This helps and I don’t think it will be detrimental to the program:

It makes the description more readable so that is good.

I am not really sure where this fits into the character sheet yet. I think that the JSON file to property converter will come in handy for displaying stuff from the internet onto the character sheet.

I’m going to make a start on the updating any changes made to the texts.

My general idea of this is:

* Check if any changes have been made to the html code, if not then no changes will be made. If there are changes:
* Find what has been changed
* Find who has been changed
* Update the appropriate JSON file contents

I can check that it works by manually looking through the JSON file and checking for my changes.

While researching methods of getting form feedback, most of the methods involved PHP, which I know nothing about. I would like to talk about this with John on Wednesday but so far, I am happy with what I have learnt and what I have made.

## Tuesday 26th January

I may look into some PHP tutorials to get an idea of what it is, just so I can try my hand at it before asking John. I know what I need from PHP, so I am not going to try learning everything about it, just whatever is necessary.

Looking at w3schools, PHP is a server scripting language. There’s a lot of strange characters when writing which would make it difficult to learn so thankfully, I’m not doing that yet.

I’ve just had the realisation that I will need to get information from the flask server. I remember a section of the flask user guide dedicated to returning information, that will be very useful so I will look at that. If there’s a way to just return the input of each text box/text area that would be perfect. Well one thing at a time, PHP first.

It looks like once you click the submit button, you are directed to a “something.php” page where your data from the form is also sent. So, if I can list the name of each element, which has the ID of the character in it, along with its contents, I can maybe export this to Python, where I can read it and make appropriate changes to the according JSON file. Very easy yes.

I used a for loop to make a PHP file showing the contents of each text box, I will update the flask program to incorporate a PHP page showing the contents of the form.

On the first try at displaying content, I got a 405-error meaning Method not allowed. I did some research and found that there may be problems using PHP on flask. I am not sure if the two conflict. I would like to get feedback tomorrow but I’m not sure where to go from here, all I could do would be to compare a non-existent updated form with the current file.

I’m going to update my repo then send the project diary off, hoping tomorrow will be useful.

## Wednesday 27th January

## Meeting Notes:

* Paste in project description to first page of project diary
* “don’t be reasonable and rational” note from last week - This is wrong - We should treat the labelling of information as a sensor, recorded by a human – e.g., measure dimensions of the head and use rules when determining beauty - turning fuzzy world into scientific facts
* Two types of qualities:
  + Personality stuff: stuff people say about someone
  + Life Choices: Would you rather – How do people differ in a way that’s meaningful, what car would this person buy if they had this budget? – Who do they find attractive out of a group of people?
* New web server code makes web server maintenance easier
* Every label should be unique – be able to uniquely identify each element
* When making UI, code should automatically make the UI from the data in JSON
* We make new properties using text boxes which will then be made into JSON code
* DOB = data dictionary with two members:
  + type = string
  + value = date
* Use bootstrap when making web pages – calendar – w3schools
* Form makes character sheet editing easier later on in project
* Every character sheet will be a folder
* Don’t use templating as Flask recommends – Manually construct HTML
* Form will use GET request:
  + easier to debug
  + let’s other programs interact with it
* UI controls that will modify the form
* End goal for form is to be able to go through a JSON file and manipulate and output the contents.
* Go through JSON file recursively to display file’s contents
* Construct the label name through the ID of the character and property

The meeting was very helpful, I feel like I understand what I need to do now. The new web server code was very useful as it makes putting the web server up easier. There are some problems executing the Python code.

The source of the error was the IP and Port the server was meant to run on, I’m not sure why it will not allow the server to run here, but that does not matter, I removed the parameters from the app.run() function and let the default server, localhost:5000, hold the server. Now it runs perfectly.

I want to next be able to create a recursive function which will comb through a JSON file to find the name of an attribute and the value associated with it. The idea is to have a function which can read a json file character by character and get the name and value of attributes, this can help separate the json into sections by finding “{“ characters, and the “:” will show that the left-hand side is the name of the attribute and the right-hand side is the value. This also helps as it doesn’t specify attributes which are only useful to the character sheet. If I do this right, I can make an editable JSON form that will be easy and accessible.

## Thursday 28th January

To help with the reading function, I was thinking that developing a small key could help when implementing the code.

|  |  |
| --- | --- |
| Character | Meaning |
| { | A new section is created , maybe indicate this at first through a \t |
| } | The section is closed, Delete the \t |
| [ | A list is opened |
| ] | A list is closed |
| , | Next value in the same section |
| : | The next value belongs to the previous value |
| Opening “ or ‘ | The beginning of a value/name |
| Closing “ or ‘ | The end of a value/name |
| None | No value |
| Digit | Numeric value |

I don’t think that I can iterate character by character over a JSON file so I will cast it to a string.

The iteration character by character isn’t going too well and my laptop sounds like it is about to take off. I don’t think this is efficient enough to work well. I need to think over other solutions.

# Week #4

## Monday 1st February

I’ve had an extremely stressful weekend dealing with cancelling accommodation, banks, and family illnesses so the project hasn’t been on my mind much, which is annoying because I enjoy working on the project. I’m a little stuck on how to go through the JSON file.

The issue I had with my laptop sounding like it was taking off was due to the fact that I had an undiscovered infinite loop because I didn’t account for single quotation marks in the code. The function is able to go through the JSON file until it gets to the end where there is a index out of range exception but at least it’s nearly there!

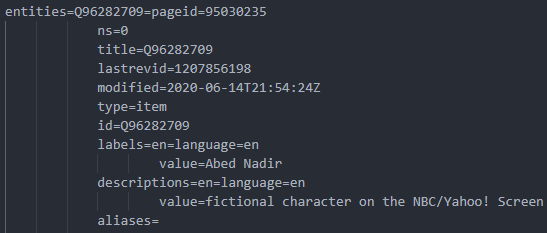
The How-To guide is due this week, I think I will do my How-To around JSONs, but I don’t know if I know enough to be explaining something. All I know is the structure, what they can be used for and, if I can finish it, converting JSONs to forms. I guess I’ll ask in the meeting tomorrow.

I left my computer for a while so I could come back and figure out the problem, and I finally did it! I actually output something without an error and it’s very relieving even if the output looks rubbish.

I’ve made it semi-readable so it’s easy to see different values, the problem is that the program loops over the same section, so I made the recursive function return an array with the text in the first element and the new index value in the second. Now the code works perfectly, and with the added indents in the code, the text is properly formatted to show the different sections. I think this was the difficult part and translating this new code into a HTML form will be easier than this disaster of a day… hopefully!

Before:

After:



It may not look like much, but this will be much easier to translate into HTML, I’m aiming for the final function to cut out the middleman and just translate straight to HTML from JSON.

## Tuesday 2nd February

## Meeting Notes:

* JSON Parser Library
* No 3 - https://tech.signavio.com/2017/json-type-information
* How-to guide – how to on r&d at beginning, converting html to json and back the other way, survey article, restrictions we can add, to describe a specific element we add its navigation such as entities-Q49839-aliases-0
* Don’t focus on code more on researching ways of going about the project
* <https://www.learnwithjason.dev/blog/get-form-values-as-json>
* <https://stackoverflow.com/questions/34236651/how-i-convert-json-output-to-html-form>
* “Everything I’m thinking about, someone has been done before, maybe not done well or communicated well but it’s there”
* Google it before you spend hours on it.

I went and downloaded the Komodo Edit software and it’s pretty good, the built-in browser is useful, and I think it will save lot of time. I chose my template from the template site and I’ve taken out all the HTMLTemplates.com stuff so it’s a blank slate. I know what the general topic for my guide is going to be but not sure about the specifics. I’ll do a general overview in a word document then go about writing the text for it.

## Wednesday 3rd February

The template that I am using is the same as the example shown to us but it is a good template so I can see why they chose it. I’m making the text for the how-to guide and I’m worried I don’t have enough for each section. The research sections will be difficult to flesh out I think, but as long as they’re useful, I’ll be happy.

I haven’t used HTML much before other than coding along with an instructor, so this is my first semi-solo project, obviously having a template to work off of makes it a lot easier. I’ve had some issues with image placement, but I think I’ve figured it out mostly. The layout is not the best, but it is readable and doesn’t burn my eyes too much. I have a meeting for group coursework tomorrow morning so I can’t work on this all night unfortunately, I’m hoping at this pace I’ll have plenty of time to submit, fingers crossed.

## Thursday 4th February

I was worried I didn’t have enough information in my guide but for someone getting started on JSON in Python, I think it’ll actually help them.

The website is going well I have the first three sections done and the website layout is looking well. I’m about halfway through the 4th sections text hoping to be nearly finished today, adding some finishing touches tomorrow.

I am finished with the writing now, I made some changes like changing an image of a table to a HTML table, making it more readable. I am disappointed I did not get further in my work, meaning I could not give a full tutorial. It’s for the best that I did not pretend to have figured out how to convert between JSON and HTML because if someone actually tried it and it didn’t work, I’d get a bad reputation before my career’s even really started. Once I get my converter working, I will either update my current site or write another guide.

I’m happy with the guide and I think it provides useful help, generally for people beginning to use JSON in Python, and for guidance on how to convert between JSON and HTML forms. I like the way it looks and think it is easy to read, I’m sure if I had more experience in HTML or CSS, I could make it look better. I think it makes the work I’ve been doing over the past few weeks useful and working on the site has helped me understand my own work better as well.

## Friday 5th February

Reasoning for Each Page:

Welcome:

* The first page gives the reader a disclaimer saying that the tutorial is not complete, but a later article will detail the conversion of form feedback into JSON syntax. I didn’t want to lead the reader into thinking the tutorial was complete.
* The prerequisites just tell the reader what they need to have installed, and what they have a basic knowledge of, before trying the tutorial. This is needed because the user should know what I assume before attempting to follow the guide.

Introduction to JSON:

* The second page details what JSON stands for, how it is used, and gives a simple example of a JSON file. This is a beginner’s guide to JSON, and I think it explains the file format in an accessible fashion.
* The example is also used on other pages to give the guide a sense of continuity.

JSON and Python:

* This page details to the reader the link between Python and JSON and they are compatible. It combines what I’ve learnt so far in using JSON in Python.
* I wanted to especially write instructions that would have saved me time when I was first starting out, such as how to access the elements once you have converted the file into a dictionary.
* It gives the reader basic tools to use for anything JSON-related in Python.

Converting to a Form:

* I give the reader a description of an example method which extracts information and converts it to HTML code, I didn’t want to give a full implementation of a method because people have different styles of coding and may want to implement their own, a full implementation and explanation would also take up too much space on the tutorial and may make the reader lose attention.
* The element naming convention part of this page is important, I give the user a good example of a naming convention which helps track each element and other advantages, but also make sure to say that there are other ways they can do it.
* I include an introduction to building forms, again I didn’t want to expand too much on the explanation of how to build a form because I assume the reader has basic knowledge of HTML.

Converting to JSON:

* The final page of the guide is the most limited, I explain on this page that I have yet to find a solution to converting form feedback into JSON syntax, I do, however, give some tips when converting into JSON format, such as adding type information and how to write a dictionary to a JSON file using the json.dump() method.
* To finish off the article, I give the reader some links to articles I am finding useful at this stage of development. I also include a StackOverflow link just in case they have specific issues.

## Sunday 7th February

Got my marks back for the how-to guide, very disappointed with it. I think I’m too focused on JSON, but I don’t know where to go from here. Going to contact John tomorrow to help develop a timeline or some milestones to work towards. What I take from the notes is that I need to reach out more and not be so independent which is going to be difficult, but I need to start somewhere.

Notes from feedback:

1. Communicate more with John so he can keep me on track:
2. Don’t lose momentum, keep active on the project.
3. Ask for help with issues.
4. Step back from my work, create a plan for the rest of the project.
5. Keep work tidy; polish and document work before moving on to the next part.
6. Ensure work can be used by others and make it easily understandable.

Action:

* Message John tomorrow asking where to go from here and what milestones I can work towards
* Create a project plan from that
* Create a daily itinerary of tasks to complete, make sure tasks act towards a greater goal
* When I am stuck on a problem for too long or feel lost, message John
* Create a document which stores the work I do, explaining what everything does, document ant possible improvements that can be made on the work
* Don’t over complicate the work, comment each function and what they should do

Notes for future blog post:

1. Explain work in enough detail that others will be happy to build on the work.
2. Explain goals of the project.
3. Create a smaller amount of work to a higher standard which will be attractive to strangers online.

Notes for Improving future How-To Guides:

1. Google the title for similar articles.
2. Provide a short description of your guide to sell it to the reader.
3. Create a target audience profile to cater towards them.
4. Get user feedback, ask more experienced people to correct/improve upon it.
5. Create a video for the guide.
6. If there are too many steps, think about setting up an install script including a sample project.
7. Explain any function calls or include a link which explains the function call.
8. Add a troubleshooting section including common problems with the system.
9. Identify how to promote the article:
   1. Figure out target audience
   2. Figure out what they like
   3. Find out what the most popular articles look like
   4. Add an enticing picture to accompany article
   5. Test a range of titles

# Week #5

## Monday 8th February

I got my recursive function up and, nearly fully, running, showing a form showing JSON file attributes. All that’s left to do in the code is to figure out a way to show lists of values, could use a in\_list boolean variable to indicate an attribute is part of a list. The need for this is for when I’m converting back into JSON and I need something to indicate that an attribute is part of a list. I need to figure out the unique naming of each element, I know what it needs to be, but I don’t know how to find the file path yet. I could also add a “type” textbox beneath each “value” textbox for the user to indicate the type of value they are inputting. But for now, I’m glad I have a form to display.

The future aims for the form program are:

* Figure out how to find address of value in dictionary, research online
* Figure out how to display lists
* Updating JSON files – Converting form feedback into JSON
* Adding new attributes – a plus sign which, when clicked, adds a label, “value” textbox/textarea and a “type” textbox.

## Tuesday 9th February

I sent my code off to John to get advice on it. The feedback was good, I am no longer converting the JSON dictionary to a string, which I didn’t feel too confident about in the first place. The idea now is to keep it in dictionary format which allows the program to find out what type of instance it is, e.g., dictionary, list, string, etc. Then from here, have separate functions for each (basic?) type. If we have a typed dictionary, we can get this type and call a more specialised function, such as money\_to\_form or date\_to\_form.

The feedback also included a skeleton structure of a path\_to\_id function with comments saying how it should be implemented. The ID can’t have special characters, so my idea is to use the Unicode functions ord() and chr() to replace these characters. But before that I need a list of all special characters that can’t be used. From here I could:

* Iterate through each character in the special characters list:
  + If the character is in the path string:
    - replace(character, “\_\_”+ord(character)

This would make converting back to special characters easier as the chr() function takes the number given by ord(character) and returns the character.

At the moment, I’m trying to get a converter working which converts special characters to Unicode values and back. There needs to be a way to find the value so I used a template: “\_-“+number+“\_” this lets us find the number in a string, meaning we can convert the Unicode value back to its special character. I’ve tested this on the string.printable string which includes a variety of special characters and it was converted back and forth perfectly.

I am not sure how to initially convert the key into a path, from what I’ve read online there is no short clean way, they mostly involve either for loops, iterating through each key or involve getting a list of all keys and a list of values and finding the match. Neither of these account for more than one occurrence so I don’t think this is the way to go. I could have a variable that records the name of the current instance. I am really unsure. I will ask for feedback.

## Wednesday 10th February

## Meeting Notes:

* Looked at code for writing form:
  + Insert code for changing special characters
  + Need to figure out a “type” solution
  + Make methods that handle each type of data and convert them to html elements
* Looked at code for getting form feedback and then a brief explanation of how to change it back to python.
* Begin to think about how to polish code and make the project attractive to other programmers to want to continue.

The project is beginning to look a lot cleaner and more focused; I can see what the goals are. I’m first going to attempt to get the very basics done then the work can continue to evolve from there incorporating:

* adding elements
* using more specific data types
* formatting of the form to make it attractive
* maybe adding JavaScript to use more advanced features

## Friday 12th February

It is difficult trying to balance the coursework for different modules and trying to keep on top of notes as well as everything outside of university. Today I’m going to get the str\_to\_element

function working so that I have some output to the form.

I built the str\_to\_element and int equivalent, which is very similar to the str function, casting the integer values to string equivalent. The problem now is that the program interprets the sub-dict “entities” as a string and I am not sure how to get out of this, I have messaged John and I am now waiting for a response. In the meantime, I will continue with trying to convert the ID to path when parsing the form. I have inserted John’s pseudocode as a guideline because I think it is a good baseline to work off of.

# Week #6

## Monday 15th February

I’ve been thinking over the weekend about how to best go about solving the problem. I am going to go through line by line adding variable to the watch and seeing where possible problems occur.

I’ve tried many solutions, but I seem to have found the problem. I misunderstood the code, I needed to use js[key] variable instead of key as the key variable is the name of the dictionary. I am not sure why there’s probably some difference between those but now it goes through every item in the sub-dicts. A new error has appeared now called a memory error which, if I have googled correctly, when the program runs out of ram. I think it is because the variable which holds the HTML code, “prev” has run out of storage to hold the full string. I am going to try and cut out the middleman and instead of using this variable, I will just write each element to the file as it gets to it.

So, the memory error has gone, but now the issue is that the output is only the last sub dictionary. I’m not sure what is wrong with it.

The problem was that the open mode of the file was in write, so the problem was that a lot of the elements got overwritten. There was an easy fix, which was changing it to append mode.

Remaining Problems:

* strings do not show – check str function
* repeating form elements

The problem was literally just indenting in the wrong place, so I won’t go into that.

I’ve got the form outputting things now so the next problem to face is cleaning up the labels, as some of them are way too long.

I reworked the code defining the label to find the last underscore in the element ID, this means only the relative part of the path is being used. Some problems I see at the minute:

* Empty fields don’t show in form
* No way to see lists – bullet points
* Different sections – padding for the divs

## Tuesday 16th February

I tried a few other json files to test out the program and find any faults. I found a bug when working with lists and updated the code to check if the element is a list then passing the key as a parameter instead of js[key]. It was a similar problem to the dict problem, so I just copied that code but other than that the program runs well. The Bart Simpson JSON creates about 6500 lines of HTML code so I should think about implementing a search option to find things easier in the future.

For the empty fields problem, I have an idea of how to get around it by checking if the value is null then creating an empty textbox which can be filled in. I will add padding for each dictionary and list but not for the individual elements such as strings and integers just so the form will be easier to make sense of. But I am not sure of the specifics on doing this.

Getting feedback from John now. The label parameter definitely cleans up the code, but I had to make some changes so that the program would work. I didn’t know you could pass a file as a parameter so that cleans up the code too, removing the opening of files for most of the functions.

The newest version fixes my earlier problems of not being able to see different sections of the divs and also shows lists which is very handy. It incorporates bootstrap and indenting the divs making the format way nicer.

The next step now is parsing the incoming data from the form and creating a JSON structure from this data. Some points to work towards:

* Research “request data” in Flask
* Converting IDs to paths
* Building a dictionary from the form data with the exact names used before (big goal)

## Wednesday 17th February

## Meeting Notes:

* Folder of default classes we can create – person, clothing
* Folder for instances of each class
* Still need type system
* Next deliverable is how-to guide and blog post
* Blog post should be catering to someone who is willing to read your work

## Saturday 20th February

I made some changes to the path\_to\_id function so that the conversion back to the path could work. I have yet to test it, but I would like to have the parser working so that I can have a baseline to talk about in my blog post. I need to start planning my blog post and the changes I will be making to the guide, which should be easy considering the feedback I’ve gotten. I am going to continue implementing the parsing form method for formalise.

The request.form object is empty when I click submit. I am unsure why; the HTML form is being sent but nothing is being sent with it. I may be misunderstanding the documentation, but this should send form data back to the program.

The issue is now fixed, the elements of the form needed names as well as IDs so now that the elements have names, the request form object is now being passed. I tested the method with the Abed Nadir JSON file as it’s the smallest and it creates the JSON file near perfectly. The file is not formatted so appears in one line but that’s a cosmetic issue. There is also a problem with the names. There is a ‘\”’ in every name for the name/value pairs. It does not appear on the names of dictionaries or lists so the problem probably lies in the str\_to\_element or int equivalent functions. I’ll go through and debug the parse form function tomorrow.

## Sunday 21st February

There is no mention of \” anywhere in formalise so I’m thinking maybe it comes from the form data. To make sure it was not an issue with the parser, I cast the request.form data to a str and wrote it to a text file, the ‘\”’ occurs here too. I assume the problem is that there is a random quotation mark somewhere, but I cannot find it yet. I am going to use a HTML validator online to find the problem quickly.

I don’t know why I didn’t think of that sooner, that was such a mistake. There was a missing quote meaning the whole input code was messed up. I’ll definitely be using this more often. The validator also sees a problem with this area of code: “</div></li></ul>” but I am not seeing the problem.

While fixing one bug another few appear:

1. Numeric data is not retrieved
2. Lists aren’t recognised so they are dictionaries with a sub dictionary for each element
3. Empty elements are not recorded
4. There was no name attribute in the int\_to\_element function, easy fix. However, when parsing this data all numbers are strings. This can be fixed later with type implementation.
5. I think the list problem could also be fixed with type implementation, e.g., dicts would be typed as “dict” and lists as “list” then test for this in the parser. Maybe that could work but it’s just an idea.
6. Empty fields could be set as an empty text box available for the user to type information. Test the value of js to make sure it isn’t null and if it is, create an empty textbox.

Tomorrow, after the database test, I’ll make the plan for the pages.

# Week #7

## Monday 22nd February

What the blog post needs to be is:

* Appealing to developers who are around my level
* Clear and concise so the reader understands but doesn’t lose focus
* Interesting to read so the reader may want to continue working on the project

My template for the pages so far is:

* Introduction to the project
  + Aims
  + How it can be used
* Building the project
  + The steps required
  + JSON Editor
  + Maybe Dean’s Part or a link to his work
  + Future Plans
  + Possible Features
  + GitHub Link to Repository
* References
  + Explaining Methods in Code
  + Flask Link
  + JSON Editor How-To Guide Link
  + Maybe Link to Dean’s guide

I will need to ask Dean if it’s alright to include a link to his guide and a small description of the work that he’s done with stylegan. But I will be sure to credit him in all the parts that reference his work. I think showing that a lot of work is already done would help in making others want to continue work on this project.

## Tuesday 23rd February

I’ve got the template which I will be using for my blog post so all now that’s needed is to write the post. I’ve written some of the introduction, mostly just explaining the project and what our aim is. I explained how the details of the sheet are calculated with an example making sure I was right in what I was saying with a confirmation from John. I have talked to Dean too and he seems happy enough with the idea of trading some information about what we’ve developed, I think this will be very useful because to be honest, I am not really sure what he’s doing on his side of the project. I’m just trying to make the introduction hype up the project as much as possible to reel in any potential readers. I’m not sure if including that I’m a student would entice the reader more because that could be a put-off to the reader.

I’m currently making a template of a Person JSON file with possible attributes, I haven’t put in type information yet because this is just the baseline and won’t actually be used in the project. I’ll get John’s opinion then maybe make some people examples with famous people for testing purposes.

The new Person JSON file has less sensitive information so we can start off easy and non-controversial, which makes sense. I think now, I gather datasets about people and take data from that to insert into the Person template.

## Wednesday 24th February

## Meeting Notes:

* Type system still needs implemented probably after the blog post is finished
* Lay out possible formalisations in blog post, all of the possible things we can predict about people
* How we can formalise attributes in different sections, big five, different fallout tabs

I’m mostly focusing on the blog post this week so all of these will be written about in the future plans section of the blog post. I will update the how to guide to include the parsing form and issues with that. I need to clean up my GitHub repository so that it looks clean and presentable.

## Thursday 25th February

I’ve been emptying out the template to make room for my post. From looking online and at examples John gave for good blog posts and tutorials, like pyimagesearch, I need to keep it clean, I think rather than have different pages for each section, I will just have one continuous page with maybe a table of contents for parts in the post. I think that would be useful if any student/developer decides to continue the project using my work as a base, as it lets them find areas of the post quicker. I will keep my site clean and easy-to-read. I’m including my social links for shameless self-promotion and I don’t think it’s too off-putting to the display. I’m just going to call the blog “Nathan’s Blog” because I don’t actually have a blog at the moment. I think now all that’s left is to finish writing the blog.

# Week #8

## Monday 1st March

I think I should give a disclaimer at the beginning, saying why I am doing the project. Just a little thing about the module and the description I was given. This is what I was given so it’s only fair to give any potential viewers the same project description. I’ve separated the blog into different sections:

* Introduction
  + Details on the project
  + Why I’m doing the project
* WikiData
  + Use of formalisations of people as test data
  + Gave useful information about JSON structuring
* Flask
  + What it is
  + What I’m using it for
* JSON Editor
  + Creating the form
  + Getting Form Feedback
  + Parsing feedback
  + Future Type information
* StyleGAN
  + TBA
* Future Plans
  + Possible Features
  + Uses
  + End Goal Reiteration
* Useful Links
  + Python 3 Installation
  + VS Code Installation
  + GitHub Repo
  + Flask User Guide
  + WikiData API
  + Personality Traits Formalisations: Open Psychometrics and WordNet Adjectives
  + Papers with Code: Hair and Fashion Formalisations

I want to give any reader who wants to continue the project all of the resources I have been given so far, so I think the useful links section covers it mostly. I’m not sure I have enough for the post but at this point there isn’t much I can do. I will just have to detail what ideas I have for the future and have a plan for going about it.

I have half of the blog finished, my code is up to date on GitHub but I still have to make my guide changes but that should be grand because I made a list of things to do after getting my mark.

## Tuesday 2nd March

Today I’ll take a break from the blog writing to make my changes to the How-To Guide. The changes I have made are detailed below:

|  |  |
| --- | --- |
| Change | Reason |
| Added a description detailing the guide | Most articles and how to guides have a part in the intro detailing what the guide is for |
| Added links in the prerequisites part of the intro | If the user does not have the prerequisites downloaded, a link would make it much easier for them. |
| Edited the Feedback section to finish the guide | The guide needed finished to make it have value. |
| Added a section in the form creation section for Flask | Previously there was no explanation of Flask and I feel like this adds value to the guide. |
| Explained code in more depth | There was not adequate explanation previously. I think the last two pages have much more value now. |
| Added more detail to form creation | The guide assumes the reader knows how to create a form and leaves this out, so I corrected this mistake. |
| Added more images of code | The visuals accompanying the written text helps the reader to understand what I’m explaining better. |

I think the new how-to guide is much better than the previous version and I am happy that I redid it. The next thing I have to do is write the JSON Editor part of the blog and I think that will be much easier now that I have the guide finished. After that I have the section on Dean’s work and the future plans I have for the project.

## Wednesday 3rd March

## Meeting Notes:

* Blog name does not matter
* Content limit = how much a student is willing to read
* Going to get feedback on the content from Dean and I’ll give him feedback on his

Today, I’m going to get the JSON Editor blog part done and make a start on the future plans section.

While writing the JSON editor part, I noticed myself getting bored by some parts. I took the unimportant rambles out and kept the mildly uninteresting necessary parts. I wanted to break up the text, so I separated it using headers, smaller paragraphs, and images of code. I will probably only include a short description of Dean’s part in StyleGAN as I don’t want to talk too much about something I know very little about. I will be checking what I write about machine learning with John just to make sure I don’t give out false information.

Tomorrow I need to finish the future plans section, get Dean to send an explanation of his work for me to explain in my own words for the blog, and then do the same for him. From there all that’s left to do is get feedback on the blog, give feedback on other people’s, and do some final checks.

## Thursday 4th March

So far today I wrote a small paragraph about Stylegan2-ada, just giving the reader a heads up that someone else is also working on the project, giving them a link to Dean’s blog. I just gave a short description of what stylegan2-ada was and it’s use in the project, making sure to check with Dean that all that I say is true and correct. I also included an image of a StyleGAN function of sorts just to show what it does.

I want to write a few more paragraphs for the future plans section then I’ll be able to send it off to a few friends and others for feedback on:

* Readability
* Level of Interest
* Appearance
* Quality of Communication

I’ll ask for written feedback on how it could be improved. I will include this in the Diary and improve it based on the feedback.

I’ve finished my blog and I’m currently waiting on some feedback from John regarding what I’m saying about machine learning as it is an area that I am not sure in. I’ll start sending out my work after he gets back to me.

I made some edits to incorporate the feedback I got from John, but while reading over my last few paragraphs, I noticed the writing just drops off and shows the links. I didn’t like this, so I added a final sentence saying that there are many places to go with this project to hype up the reader.

To make getting feedback easier I set up a GitHub Pages repository meaning I can just send someone the link and get feedback. Link: <https://nathan0donnell.github.io/>

I sent it to some of my family and friends to check the readability and grammar. First person came back and said it reads very well but pointed out a few mistakes in the writing which I have now fixed. Here are some responses below:

“Brilliant Nathan! Only question is you say at the start “as I mentioned this is ...” where do you say that?” – I corrected this mistake in my writing

“Very punchy & to the point.” – This is what I was hoping for, something common with all feedback is that they say it reads very well.

As my target audience is people like me, first-year students at a decent level of skill, I thought it would be very beneficial to ask someone in the CSC module to give feedback on my work. Who better than my project partner, Dean, here is his feedback:

“

* Intro was very nice, not too complicated
* 'flask is a micro web framework' explain in simpler terms if possible?
* Despite there being a lot of text, the blog is surprisingly easy to read, images breaking up text into smaller chunks is nice
* you state that 'the “request.form” is an immutable multi-dict' could you maybe put brackets after explaining this simpler?

After only reading through the blog post I would say I have a decent grasp on what has been done, and in a scenario where I would be spending a week looking this blog post and all of the other files provided, I believe in that time frame this blog post would be more than adequate in catching me up to speed. The only critique is that there was some language I was confused at so maybe just add brackets after technical terms explaining them in simple terms could be cool, also subtle but bolding things definitely helped break up the blocks of text and the future plans section seems very interesting, I like that you used The Sims as an example as it’s something pretty much any student could relate to.”

After taking his feedback into consideration, I feel he is right about the technical terms, I will expand on them in my writing and try to keep it understandable. I’m very happy with the response though, I am glad it reads well and that it’s not a convoluted wordy mess, a.k.a. the bane of students’ existence. I think the best part is that he says he would “have a decent grasp on what has been done” and would be able to continue on with the project using my work, this is exactly what I was aiming for when writing the post.

I will now make the changes to my post, adding in explanations anywhere I use more technical language.

After having made these changes and having a final few reads over the post and making corrections, I am very happy with how it came out and I think my standard of work has improved since last time.

Here is a detailed list of decisions I made for the blog post and my reasoning behind them:

|  |  |
| --- | --- |
| Decision | Reasoning |
| Starting with a section dedicated to WikiData | This is how I began my project and I wanted to have a chronological order for my blog post so that the reader can see my line of thinking more clearly. |
| Dedicating a section to Flask | Flask is what makes the JSON Editor run, it is very important, and I think no matter what way the reader decides to go, Flask will be of use to them at some point. It’s just very good software. |
| Dedicating a section to Dean’s work | Dean’s work is as much a part of the project as my own and it is important that any student wanting to work on the project knows exactly what is available to them in regard to resources. |
| Adding an image of The Sims | I think it brightens the post a lot and brings more colour, this is the same reason I added my other images. |
| Having the preview image of the blog post be a snippet of Dungeons and Dragons character sheet | My target audience is CS students around my level who would be interested in data analysis and who would enjoy the work that comes along with this project. Who better to appeal to than fans of Dungeons and Dragons? I feel it isn’t too divisive making non-fans feel excluded. It also fit in with the black and white of the page content around it. |
| One page rather than multiple | I think that in a guide it makes sense to separate the pages as it has a “step-by-step” structure. For the blog post, I wanted a clean, well-constructed page with content that is easy to read. Separating the pages would disrupt the structure and cause the reader to lose interest. |
| Title: A Real-World Character Sheet Based on First Impressions | Short and sweet, grabs your attention as it is understandable but leaves you wondering how it’s done. |
| Telling reader that I am a university student in the CSC Module | The reader should know the academic background and level of work they should expect going into the post. |
| Contact links | If a reader is having difficulties with the software or has questions in regard to the project, they can easily find me and message me. |
| The level of detail in regard to the JSON Editor | I wanted to give an adequate explanation of my code to the reader, but I did not want to bore them. I also gave a link to my how-to guide for building the JSON editor, if the reader wanted extra information. |
| Explanation of GET and POST feedback | In the getting feedback section, I wanted to explain how I got feedback but I thought if I explained HTML form attributes in too much detail, the reader would lose interest. I did not want to leave them lost however, so I gave a link to more information on GET and POST methods. |
| Explaining technical terms like ImmutableMultiDict and micro web framework | From the feedback I received it was clear that people were lost/confused at a few parts and this is not what I wanted so I added explanations of any highly technical terms. |
| Lengthy explanation of “parsing feedback” code | I knew that this would be where I would have to explain code in depth because it is less self-explanatory and requires knowledge of dictionaries in python and how they are navigated. I wanted to make sure the reader understands, which is why I repeat the point. |
| Type Implementation sub-section | Even though the type code is not implemented yet, I still think that it is important that the reader knows it will be implemented as it is good practice. |
| Future Plans section | The future plans section encompasses the “what can be done” part in contrast to the rest of the blog’s “what has been done” part. This gives the reader specific goals to work towards with surface-level instructions to follow. I know from personal experience that when you are given something to work towards you feel much more confident about the work you do, and you want to work towards the goals. |
| “the possibilities with where we can take this project are **endless”** | I thought it was important to end the post with an optimistic message which also gives the reader creative freedom in the direction they choose to bring the project. I know this freedom is something that definitely appeals to me and others I know. |
| “What you’ll need to continue this project” | I think this part of the blog is very useful as the reader can use the page as bookmark in the sense, nearly everything they will need is right there on the page. This will save them a lot of time on googling and researching. |

A link to the GitHub Repository for this project can be found here: <https://github.com/nathan0donnell/character-sheet>

# Week #9

## Wednesday 10th March

## Meeting Notes:

* Hand in date is day after the easter holidays
* Work out scope to work towards now
* Can work over easter holidays
* important code: json to elements data types, formalising a human, do more programming.
* better to copy code as text than imaging
* should not narrate, should explain
* laying out a plan for future, formalising a life
* "has someone else systematised something" - find more datasets - fashion, glasses - tidy them up.
* detailed versions of game-like features, the sims job -> o\*net, job descriptions, qualifications
* NHS have a classified every medical condition
* Wikipedia lists
* google dataset search
* reference type - a reference to another json file

This meeting helped a lot with defining the scope of the rest of the semester, it looks like I’ll be focusing a lot more on coding, like formalising People and other objects which I’m excited for. I also misunderstood the purpose of the blog, explaining my experience more than how to go about actually doing the work or replicating it. I will definitely keep this in mind when handing in the updated version of the blog, this should be easy to fix.

## Thursday 11th March

I wanted to fix the empty text box issue, I realised that the null json type was not considered when making the json\_to\_elements as None is a type, of sorts, in itself. I fixed this by making a none\_to\_elements function that just creates a label along with an empty textbox ready for input.

I will expand the json\_to\_elements more now so that this does not happen again. Types that need to be considered:

* True
* False
* float
* tuple

After consulting John, a type system detecting more specific types need to be included:

*“Try to implement the type system into the code that generates the form from the json. If detecting that a child is a dictionary, test if it has the "type" member within it then run custom code depending on the type. Try to implement a table "type" that creates an instance of a tabulator UI component (http://tabulator.info/), assume that the contents of the dictionary with the "type" conforms to a format that suits a table, (could have it match the API that tabulator uses) then create hidden form elements for the type and all the contents so the json will be save correctly. Then add call backs to tabulator so that when the values are changed, or rows are added or removed then the hidden form elements are created or removed (probably simplest to remove them all and recreate them from the table contents) so that when the form is submitted the json can recreate the tabulator table.”*

He also said to primarily focus on exploring the formalisation of a person.

While trying to work with the Person json file using the Editor, I found that some of the values are split by the parser, there will need to be naming conventions imposed when using the JSON editor. There is also the issue of Number values being converted to String when parsing the form back. I think that currently all the issues with the code could be solved with the implementation of a type system – possibly turning each value into a dictionary containing the value and type.

I extended the Person JSON file by including more personality traits, splitting them into positive, negative, and social, this can be extended further later. A hobby section was also introduced, with different sections which appeal to many types of people, as with personality this can be extended. Another section based on something John mentioned in the meeting, a section on fashion, where the user can put together an outfit, consisting of a shirt, coat, trousers, socks, shoes, and accessories, from lists of each clothing type.

# Week #10

## Thursday 18th March

As there are multiple assignments and work at the minute, it is difficult to find time for everything. I have found some bootstrap tabs code which I am currently trying to get working. At the moment, I think a good method of tabbing the JSON dict is by sub-dicts. However sub-dicts with only a few values should not be tabbed. A sidebar could be more useful considering the amount of sub-dicts our character sheet could have. I have created a recursive function which turns the dictionary into a sidebar, I think the page content should change as each button is clicked. I have found examples on google for sidebars using CSS and HTML, I will use these as examples.

## Sunday 21st March

I want to have useful work done for handing this in, I am going to focus a little more on the sidebar, research datasets for humans, clothes, cars, and other possibly useful characteristics.

I think that I have focused too much time on documentation such as the how-to guide and the website. I think this has negatively impacted the project as I have not developed enough code. I would have liked to continue this project further and gathered survey information and seen the final outcome, I will definitely keep updated on this project though.

I have cleaned up the format of the web app and the sidebar correctly displays the dictionaries of the JSON dict. However, there is no function as of yet, but the idea is to change the page-content wrapper to display the correct information. There are some quality improvements I want to make, such as making the sidebar a tree format with the sub-dictionaries being collected under the correct parent.

I updated the format of the web form as I thought it wasn’t great. I put it through a HTML validator and corrected any mistakes I found in the code such as unpaired tags.

The type system has been slightly implemented on the form creation side however, the parsing form side is lacking. I may focus on this more soon then move onto gathering more datasets.

The check box for true/false values is probably not the best UI component as it makes parsing the form more difficult, I think a combo box containing true or false values would be better.

I think that after the “pathParts” part of parsing the form, the created dictionary should be iterated through and the value should change to the appropriate format – i.e., int, str, bool.

I have implemented the code which changes the types to the correct format. I have also made some changes to the blog post, changing the code from images to textboxes. This makes the blog more accessible allowing code to be copied if the user wants. I am going to ask John for feedback on the changes I’ve made and to see if there is an easier way to change the page-content than using separate pages.

I added code which formats the HTML code, it does not interfere with the functionality as far as I can tell however it does help when testing the code.

# Week #11

## Monday 22nd March

I have gathered more data for formalising a person, specifically in the career and personality area. I changed the personality to group the traits by the “Big 5” groups, with each group containing traits associated with the parent. I have also found the O\*NET database and taken the occupations they recorded – this includes an O\*NET SOC code, The title of the occupation and a short description of each occupation. I converted this from CSV to JSON format. This can be used as a list of possible jobs a person can have. I think a combo box type element can use these values as the source of possible values.

## Tuesday 23rd March

I received feedback from John and the problem with the boolean values was just a missing “\_value” in the value element name and ID.

I improved the dict\_to\_element function by checking if the dict is a type/value pair. This fixed the parse issue where there would be too many value sub-dicts and it shows the correct label to each value.

I will correct the sidebar to not show value/type dicts. I will also research how to change the contents of a div at the press of a button. I hope there’s a way to do it without complex JavaScript, however I will figure it out. The sidebar now only shows appropriate dicts as headers.

I need to see if there’s a way to insert html code into a div as most of my searches have resulted in solutions for basic string insertion.

I have changed my mind and I will now be using tabs instead because that is much easier.

I have now added the tab control feature, it still need work but there is some functionality. Currently, the sub-divs are not displayed under their parents in the sidebar. Also, the current method of showing tabs means that the sub-divs are made visible however, the parent is not visible meaning it does nothing. I think for sub-divs, it might be better just to show the parent-div then auto-scroll to the sub-div area.

I used JavaScript to make a while loop that makes all parents of the sub-divs visible then auto-scrolls to the sub-div specified. The only thing now is formatting the sidebar so that sub-divs are displayed under their parent in a drop-down format.

I want to add different UI elements for the different types like date, tables, and images. This shouldn’t be difficult.

I have added code for date and images, I’m not sure how the date values work, and the image code requires an ‘src’ and ‘alt’ sub-dict in the value sub-dict to work correctly but it’s working well. I need to parse the image properties as well, but feedback does not record img elements. I think using hidden inputs to record the important properties would work well. These would need to be updated anytime the image is updated.

I have now added the hidden inputs however, there is no way to edit the image yet.

## Wednesday 24th March

## Meeting Notes:

* Last meeting today – talked about how to improve the project:
  + Code should have been done quicker
  + On the programming side, I should have done more and been faster at coding, this was due to lack of programming experience in this area.
  + Type should not be added only detected
  + Have a greater understanding of code
  + A great deal of effort will need to be put in to get good marks
  + On the formalisation side of things, a lot can be done however it is a lot of effort of googling, finding datasets.
* Testing is unit tests to ensure code is working as needed

I split the JavaScript and CSS files into different files from the index.html to make the app more professional.

I removed beautiful soup to decrease the dependencies of the class.

## Sunday 28th March

I fixed the parser to parse boolean and int without a type field. There are certain cases where if a user enters True as a string value, it will be converted to boolean, but I do not think this error is an imminent threat right now. I am going to spend the majority of next week creating and looking for person instances to test the character sheet. The sidebar needs fixed again as it reads small dictionaries as tabs when they do not need a tab of their own right now. I will need to make some constraints to only make tabs of certain dicts.

# Week #12

## Wednesday 31st March

What I want to get done by the end of this module is:

* Have an attractive display of a character sheet
* Have an import button to load a json
* Have a good collection of person data
* Add tabulator UI
* Create unit tests

I tested my code and found no html errors. To make the UX of the web app better I removed the tab feature of only showing the tab it is on. Now clicking the buttons on the sidebar scrolls the section into view. I am going to add a hidden field holding the file name then an import button which will allow the user to import any json file.

I found a blog post which helps with this: <https://blog.miguelgrinberg.com/post/handling-file-uploads-with-flask>

# Easter Break

## Sunday 11th April

The long break in time was due to easter break and family issues, not much progress has been made since my last entry due to this.

The deadline is in 12 days. In this time, I have to complete the following tasks:

* Social Media Post
* Testing Plan
* Improved Blog Post
* Improvements to Code:
  + Tabulator UI
  + Formatting the Design
  + More interactivity
* Data Collection, which consists of:
  + Fashionpedia JSON data
  + Instances of Person Data
  + O\*Net Data for Career

Social Media Post:

At the moment, I am considering posting to Twitter as the tech community on Twitter is very active, particularly Data Science Twitter. I always see small tech accounts promoting their blogs, of which the variety of subjects is vast. Most of the posts I see gather likes in the 50-1K likes so the reception varies. Tweets are usually best received when the content is short yet descriptive so for my post, I’d phrase it so that I give the idea of the project, maybe livening up with some colourful, energetic language. I will add an image too as this will make the post more attractive.

Testing Plan:

I will create a spreadsheet for testing, making the next student’s work easier by seeing that my work is correct and without errors. I don’t think there should be too many unit tests, with most just pertaining to the UI elements, conservation of the file format, and properly updating the file content.

Improved Blog Post:

I want to change the narrative of the post so the headings can stay, but the content should go from “I did this” to “this is how to do this”. I will also add in the work I have done since the last writing of the post.

Code Improvements:

I want to get the Tabulator code working just so I can have a proper range of components to make up the character sheet. I want the web app to look like a professional character sheet where information is compact, and the reader can find what they are looking for easily. I’m sure I will not be able to make an interactive character sheet similar to the DNDBeyond character sheets, but I think they provide a good example of what I mean. I wanted to add import and download buttons but at this point, the way I have done this code, I don’t know how to go about that. It would mean having to get a file from the user through flask then update the index.html file and having to refresh the website, I don’t think this would be user friendly, so I don’t think it would be good. If I had more time, I would improve UX but at the moment I just want to add the functionality of looking and interacting like an editable character sheet.

Data Collection:

I want to get the Fashionpedia data and convert into a usable collection. I am also going to make around 10 instances of the Person JSON file to use as test data, I will probably use fictional characters as reference. I have already changed the O\*Net career collection from CSV to JSON, but I will clean it up.

Today, I am going to clean the data as I think that will be of use to later students.

I have downloaded the Fashionpedia JSON and taken out the relatively useless data, the data is now a collection of clothing types, I will separate them into their relative classes of t-shirt, dress, skirt, etc.

They have now been separated into their separate classes; they are some attributes which were relative to the Fashionpedia file but don’t make sense in this context so I will take these out.

## Monday 12th April

Today I want to get a collection of sample data of people to use for character sheet testing purposes. I am going to use fictional characters as there is already data available in the form of open psychometrics and fan-made wikis. I am not sure how to do the example outfit section so I may leave it blank for now.

I have made 13 JSON files of fictional characters referencing fan-made wikis and available data. I tried to find characters who have been fleshed out enough so that the files can be extended when more fields are included. I have also added images of each character as that is useful.

I am now going to update the GitHub Repo with the updated code for “formalise.py” and the sample data – O\*NET, Fashionpedia, and the character sheet files.

## Sunday 18th April

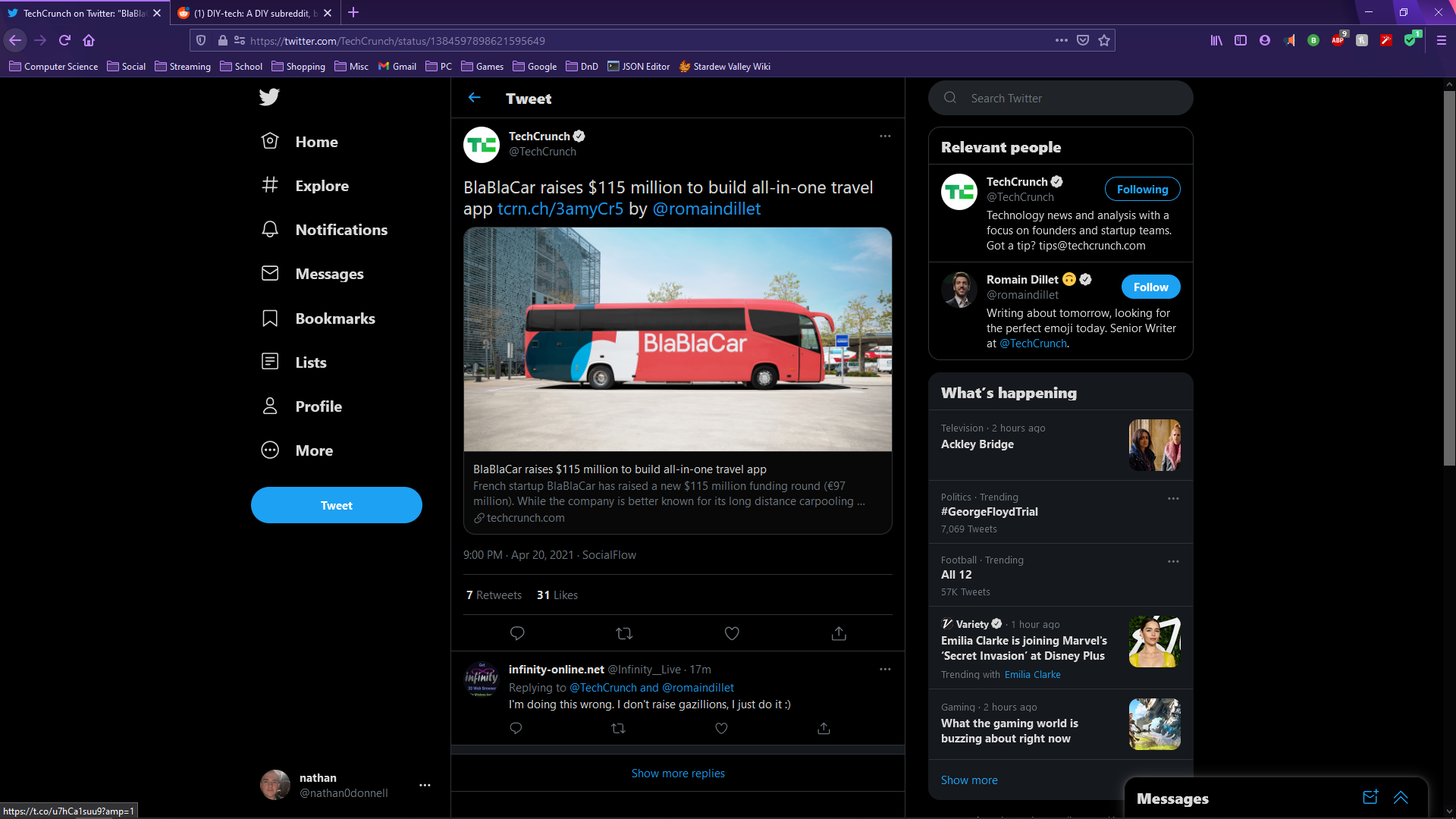
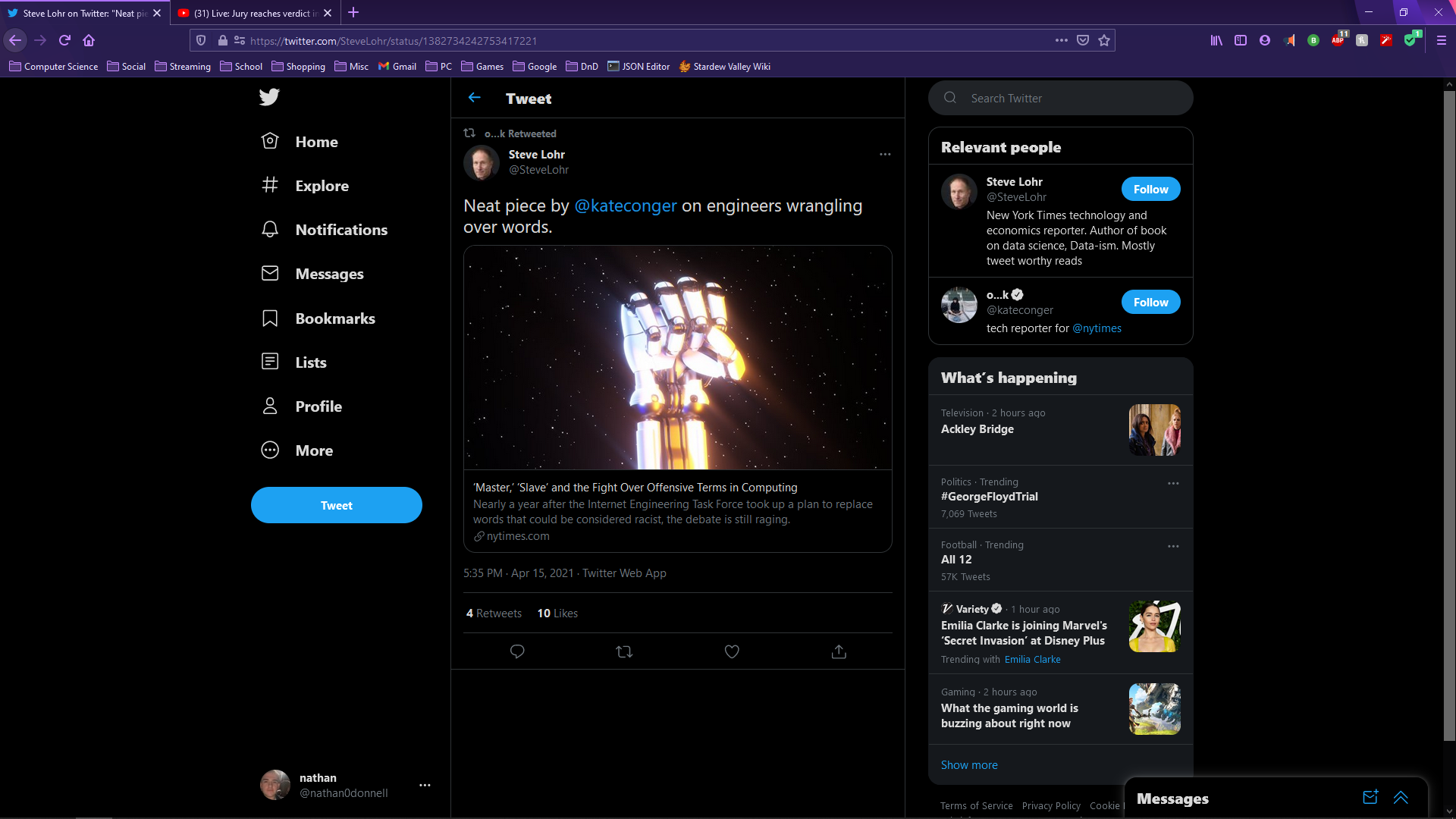
I am going to create a testing plan for the character sheet web app today. I think just a basic spreadsheet will work well.

I have written the tests; I think it tests the program well. I will add any more tests if I think of them. I will now execute the tests and write the results.

Testing is complete with short comments added for any failed tests and other comments added where appropriate.

I want to get started on the tabulator code, but I think that will involve some JavaScript which I’m not experienced in.

## Tuesday 20th April

For the rest of the week, I am going to spend time rewriting the blog post as I think this would be where I can best help the next student continuing this work, rather than creating messy tabulator code that probably won’t work right. After I have written the blog post, I can put the link up on Twitter where I can self-advertise my work. I am unsure where I would advertise on reddit as I don’t know the platform well enough. I think Twitter is a suitable place to share my blog post as it is casual enough for my level of work, yet still popular enough that the right people may see it. I’ve included a few tech tweets I have seen on my timeline and it seems short and sweet with an eye-catching image works well.

I have rewritten the Wikdata part of the blog to offer better insight into why it is used in the project and how to find the sample data I used.

## Wednesday 21st April

Today I will be rewriting more of the blog to better help the next student coming along. I’m starting with Flask, describing the features it offers and how to install Flask on their virtual environment using the command prompt and how Flask allows for getting form feedback.

In the next section, JSON Editor, I changed the subheading to “Character Sheet Editor” as that suits better. I also lengthened the Form Creation section, updating the information to what the program currently does. I added more information on the “type” implementation.

The parsing of form feedback didn’t need changed at all. I made another section describing the Sample Data directory and the collections of data I collated, making sure to say that this is where a lot of work can be done.

I used GitHub Pages to host my How-To Guide so that I can link the guide in my blog post. I need to get Dean’s Guide link also.

I’ve messaged him now and I just have to wait until he has a link ready for his colab. Once that is done, I can update the website hosting my blogpost and send out the tweet.

## Thursday 22nd April

While I wait for the Colab link, I may as well compose the tweet so that it is ready to send out.

Tweet:

Through @QUBEEECS "Computer Science Challenges" module, I was able to work on a project aiming to build a detailed character sheet when given a photo of a random person https://tinyurl.com/2us4huew

* Short enough but informative
* Tags QUBEECS and uses “Computer Science” phrase meaning algorithm will show it to correct audience

After some feedback from others, I made some minor changes to:

Through @QUBEEECS "Computer Science Challenges" module, I was able to work on a project building a detailed character sheet based off a photo of a person. Read more on my blog: https://nathan0donnell.github.io/

* Rewording to be more professional
* Saying what the link is
* Changing the link from a tinyurl one because this looks better

I have also added a photo of a section of a character sheet to make the tweet look more appealing.

Tweet has been sent out. I have taken a screenshot and uploaded it to the GitHub repository. All I need now is to finish the Project Diary up and upload that too.

# Final Thoughts

This module has given me an insight into what a project consists of, whether that be the amount of work that comes along with it or the stress of having all the work to do. I am glad to have taken this module as it let me gather experience in an area, I had no prior knowledge of and let me work with new technologies, like Flask, and computer languages, such as Python and HTML. This project humbled me and showed me how much I still have to learn, thank you to John for the experience and for putting together this module.

1. [Source](https://www.researchgate.net/publication/23450300_Perceptions_of_a_Photograph_of_a_Woman_with_Visible_Piercings) [↑](#footnote-ref-1)