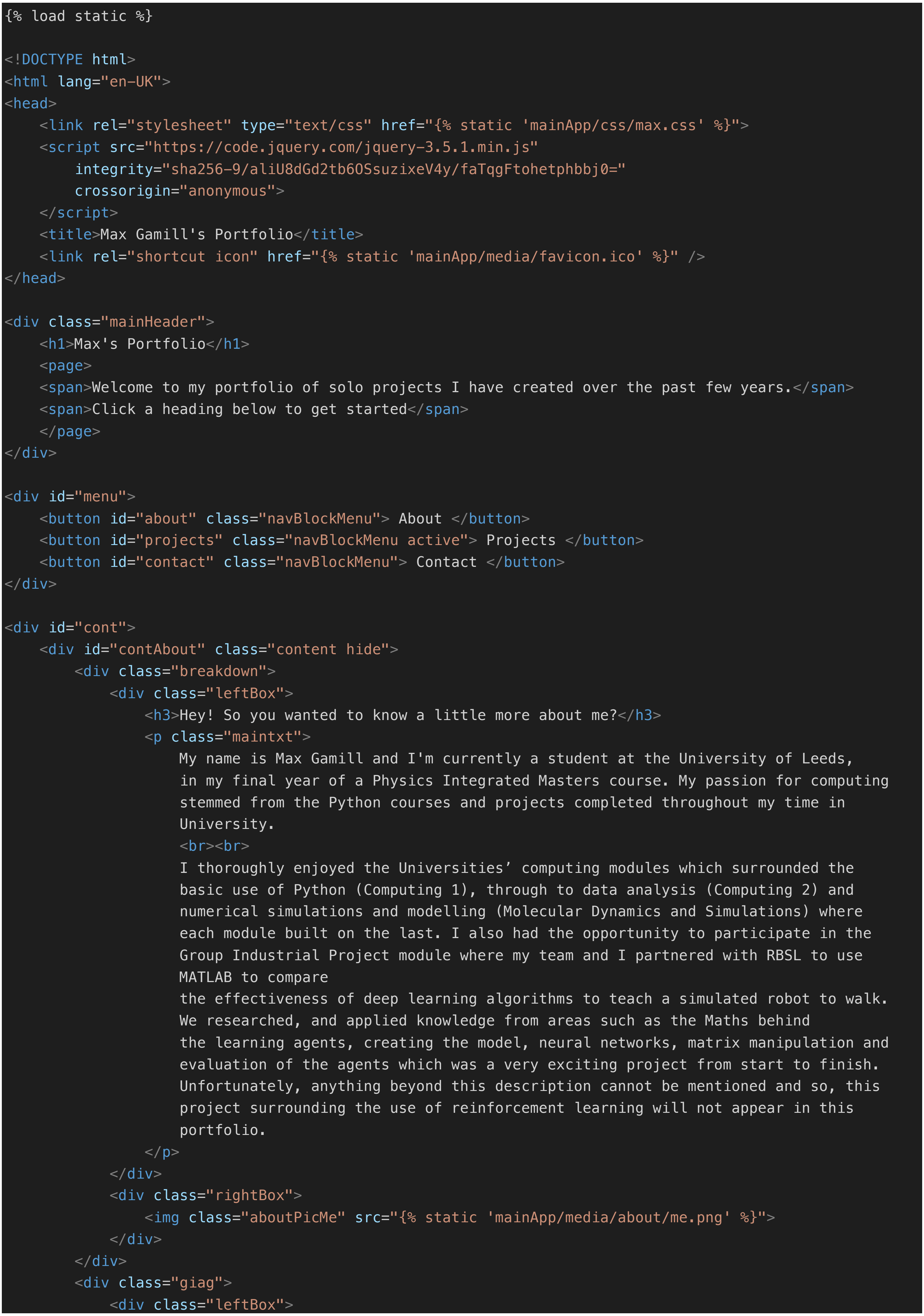
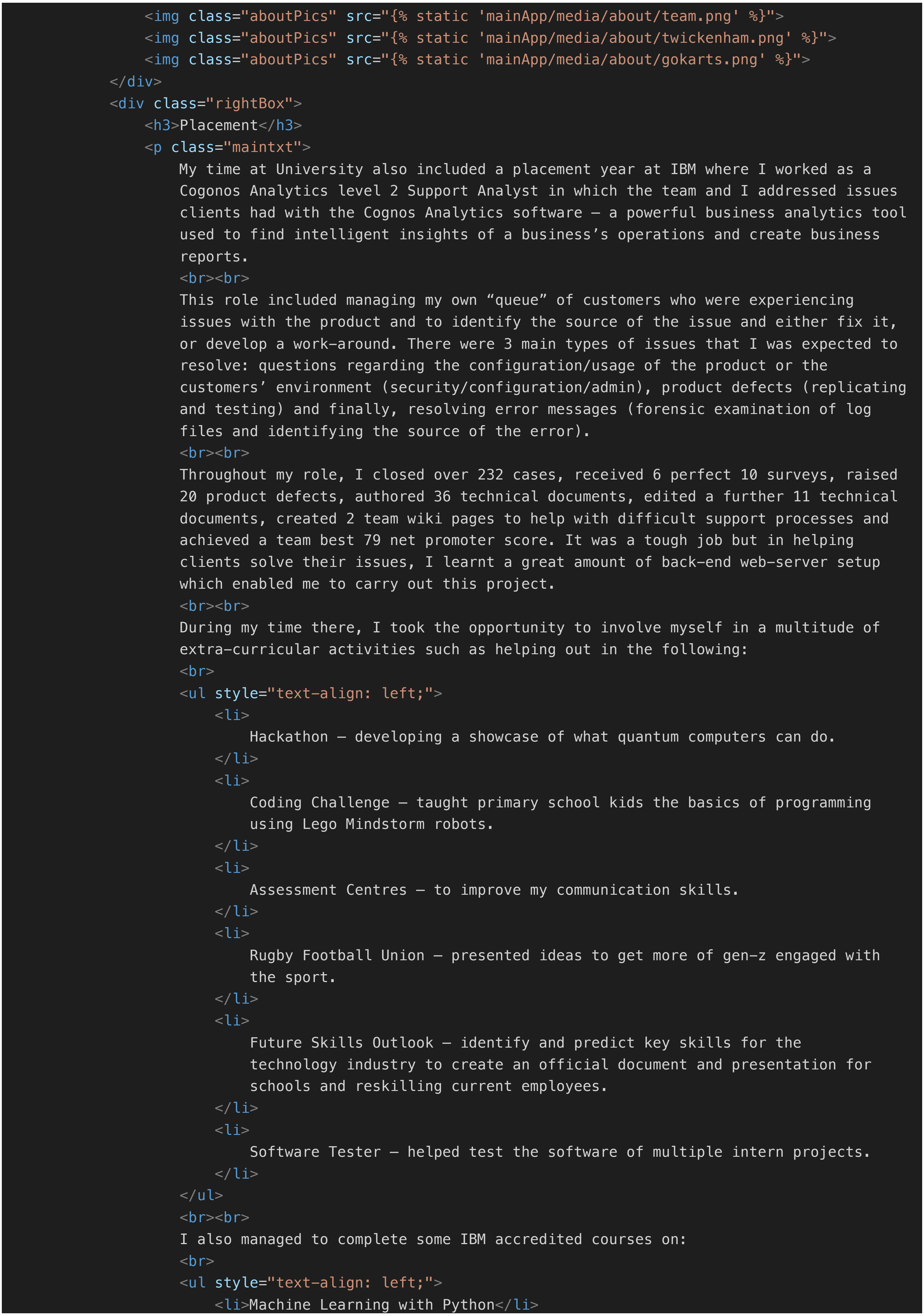
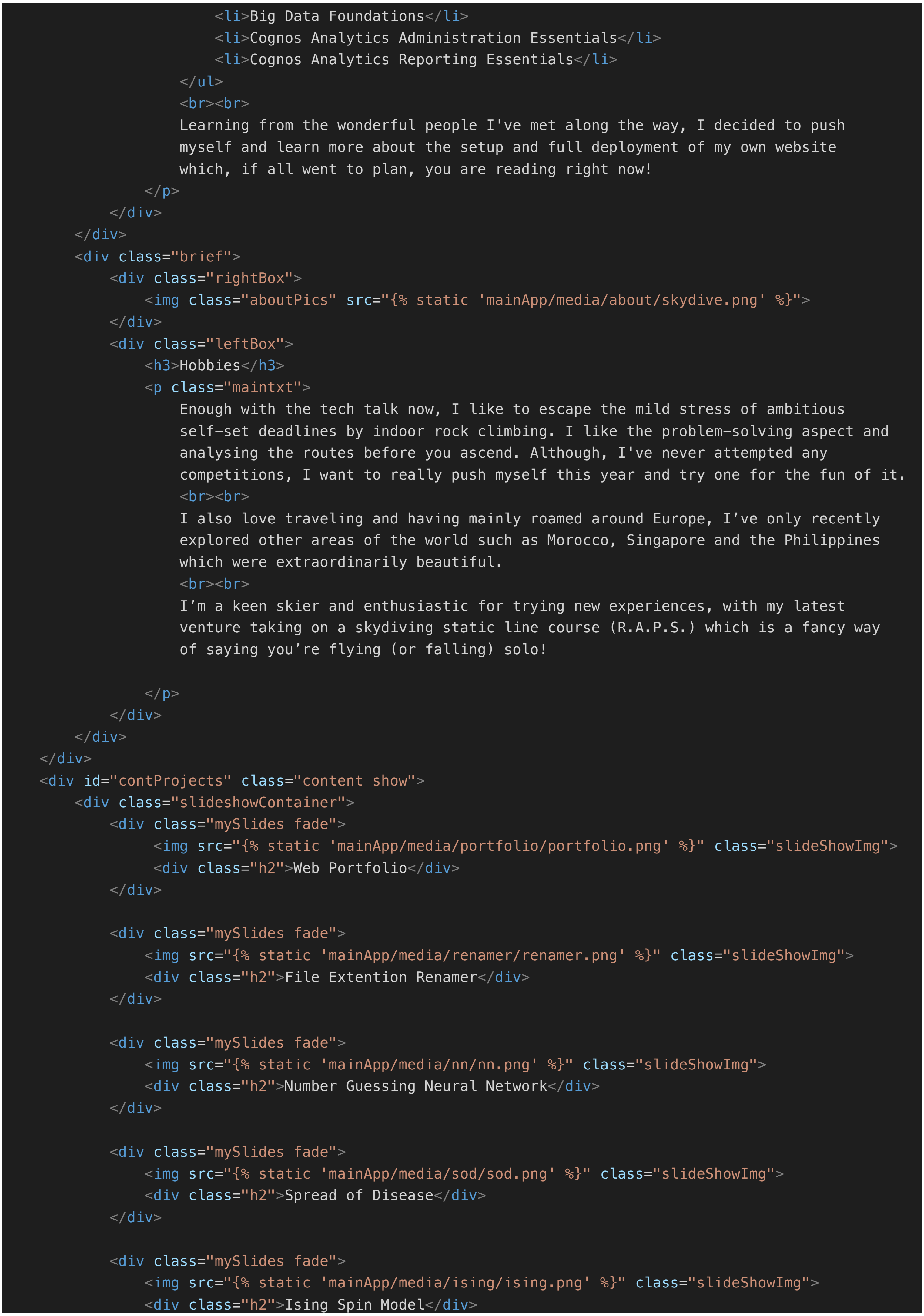


```
{% load static %}

<!DOCTYPE html>
<html lang="en-UK">
<head>
    <link rel="stylesheet" type="text/css" href="{% static 'mainApp/css/max.css' %}">
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"
        integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
        crossorigin="anonymous">
    </script>
    <title>Max Gamill's Portfolio</title>
    <link rel="shortcut icon" href="{% static 'mainApp/media/favicon.ico' %}" />
</head>

<div class="mainHeader">
    <h1>Max's Portfolio</h1>
    <page>
    <span>Welcome to my portfolio of solo projects I have created over the past few years.</span>
    <span>Click a heading below to get started</span>
    </page>
</div>

<div id="menu">
    <button id="about" class="navBlockMenu"> About </button>
    <button id="projects" class="navBlockMenu active"> Projects </button>
    <button id="contact" class="navBlockMenu"> Contact </button>
</div>

<div id="cont">
    <div id="contAbout" class="content hide">
        <div class="breakdown">
            <div class="leftBox">
                <h3>Hey! So you wanted to know a little more about me?</h3>
                <p class="maintxt">
                    My name is Max Gamill and I'm currently a student at the University of Leeds,
                    in my final year of a Physics Integrated Masters course. My passion for computing
                    stemmed from the Python courses and projects completed throughout my time in
                    University.
                    <br><br>
                    I thoroughly enjoyed the Universities’ computing modules which surrounded the
                    basic use of Python (Computing 1), through to data analysis (Computing 2) and
                    numerical simulations and modelling (Molecular Dynamics and Simulations) where
                    each module built on the last. I also had the opportunity to participate in the
                    Group Industrial Project module where my team and I partnered with RBSL to use
                    MATLAB to compare
                    the effectiveness of deep learning algorithms to teach a simulated robot to walk.
                    We researched, and applied knowledge from areas such as the Maths behind
                    the learning agents, creating the model, neural networks, matrix manipulation and
                    evaluation of the agents which was a very exciting project from start to finish.
                    Unfortunately, anything beyond this description cannot be mentioned and so, this
                    project surrounding the use of reinforcement learning will not appear in this
                    portfolio.
                </p>
            </div>
            <div class="rightBox">
                <img class="aboutPicMe" src="{% static 'mainApp/media/about/me.png' %}">
            </div>
        </div>
        <div class="giag">
            <div class="leftBox">
```

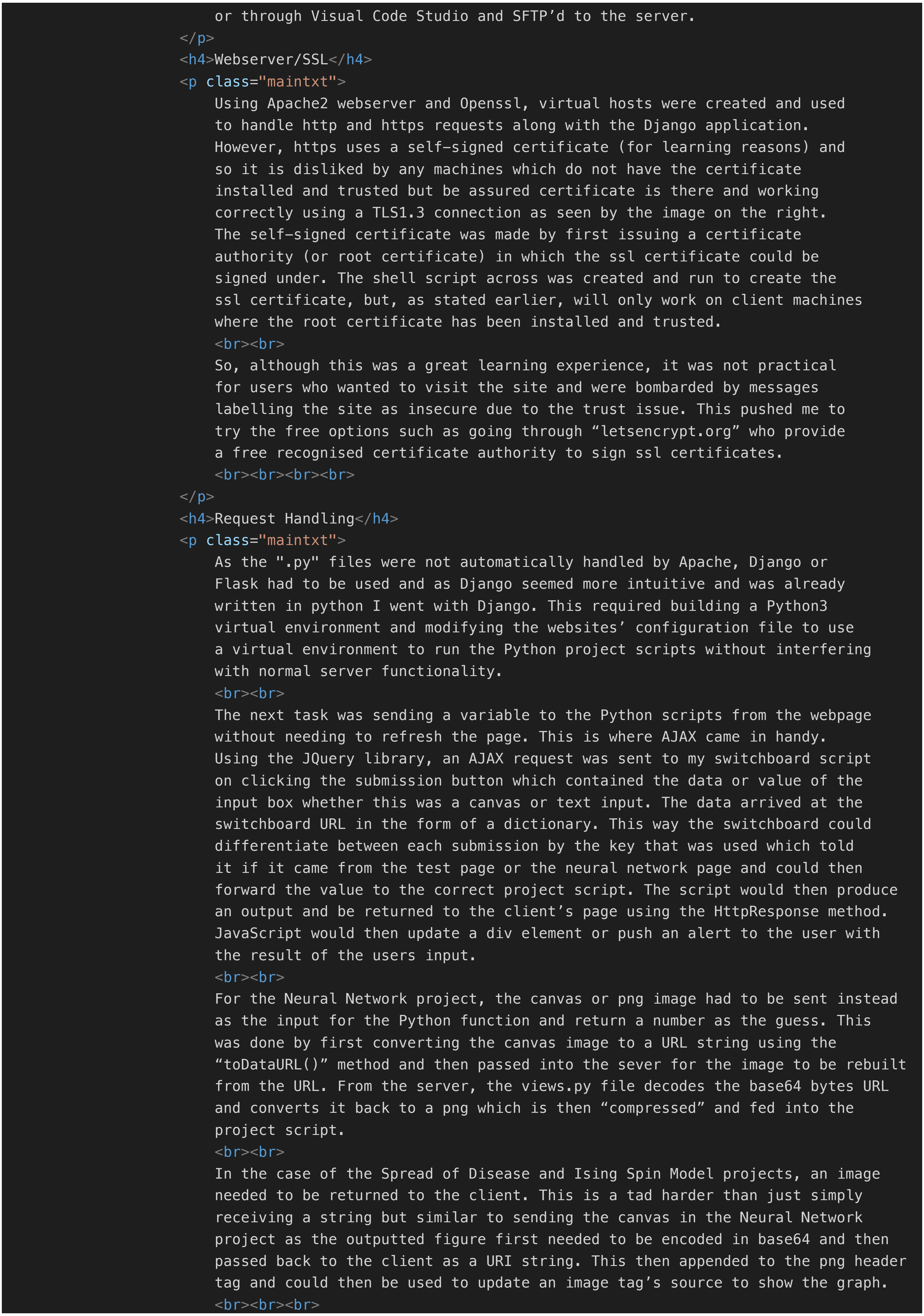


```
            <img class="aboutPics" src="{% static 'mainApp/media/about/team.png' %}">
            <img class="aboutPics" src="{% static 'mainApp/media/about/twickenham.png' %}">
            <img class="aboutPics" src="{% static 'mainApp/media/about/gokarts.png' %}">
        </div>
        <div class="rightBox">
            <h3>Placement</h3>
            <p class="maintxt">
                My time at University also included a placement year at IBM where I worked as a
                Cogonos Analytics level 2 Support Analyst in which the team and I addressed issues
                clients had with the Cognos Analytics software – a powerful business analytics tool
                used to find intelligent insights of a business’s operations and create business
                reports.
                <br><br>
                This role included managing my own “queue” of customers who were experiencing
                issues with the product and to identify the source of the issue and either fix it,
                or develop a work-around. There were 3 main types of issues that I was expected to
                resolve: questions regarding the configuration/usage of the product or the
                customers’ environment (security/configuration/admin), product defects (replicating
                and testing) and finally, resolving error messages (forensic examination of log
                files and identifying the source of the error).
                <br><br>
                Throughout my role, I closed over 232 cases, received 6 perfect 10 surveys, raised
                20 product defects, authored 36 technical documents, edited a further 11 technical
                documents, created 2 team wiki pages to help with difficult support processes and
                achieved a team best 79 net promoter score. It was a tough job but in helping
                clients solve their issues, I learnt a great amount of back-end web-server setup
                which enabled me to carry out this project.
                <br><br>
                During my time there, I took the opportunity to involve myself in a multitude of
                extra-curricular activities such as helping out in the following:
                <br>
                <ul style="text-align: left;">
                    <li>
                        Hackathon – developing a showcase of what quantum computers can do.
                    </li>
                    <li>
                        Coding Challenge – taught primary school kids the basics of programming
                        using Lego Mindstorm robots.
                    </li>
                    <li>
                        Assessment Centres – to improve my communication skills.
                    </li>
                    <li>
                        Rugby Football Union – presented ideas to get more of gen-z engaged with
                        the sport.
                    </li>
                    <li>
                        Future Skills Outlook – identify and predict key skills for the
                        technology industry to create an official document and presentation for
                        schools and reskilling current employees.
                    </li>
                    <li>
                        Software Tester – helped test the software of multiple intern projects.
                    </li>
                </ul>
                <br><br>
                I also managed to complete some IBM accredited courses on:
                <br>
                <ul style="text-align: left;">
                    <li>Machine Learning with Python</li>
```

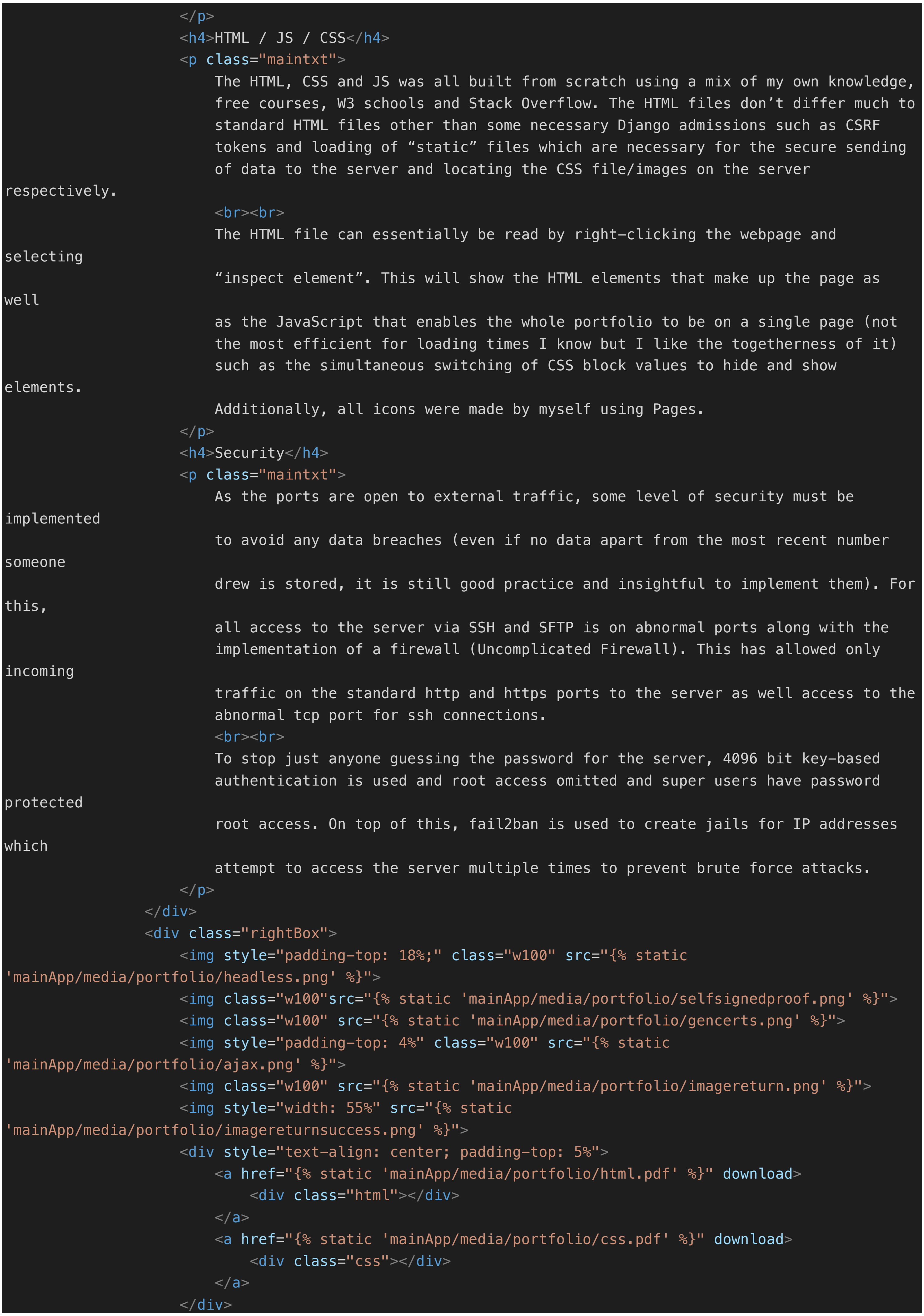


```
                        <li>Big Data Foundations</li>
                        <li>Cognos Analytics Administration Essentials</li>
                        <li>Cognos Analytics Reporting Essentials</li>
                    </ul>
                    <br><br>
                    Learning from the wonderful people I've met along the way, I decided to push
                    myself and learn more about the setup and full deployment of my own website
                    which, if all went to plan, you are reading right now!
                </p>
            </div>
        </div>
        <div class="brief">
            <div class="rightBox">
                <img class="aboutPics" src="{% static 'mainApp/media/about/skydive.png' %}">
            </div>
            <div class="leftBox">
                <h3>Hobbies</h3>
                <p class="maintxt">
                    Enough with the tech talk now, I like to escape the mild stress of ambitious
                    self-set deadlines by indoor rock climbing. I like the problem-solving aspect and
                    analysing the routes before you ascend. Although, I've never attempted any
                    competitions, I want to really push myself this year and try one for the fun of it.
                    <br><br>
                    I also love traveling and having mainly roamed around Europe, I’ve only recently
                    explored other areas of the world such as Morocco, Singapore and the Philippines
                    which were extraordinarily beautiful.
                    <br><br>
                    I’m a keen skier and enthusiastic for trying new experiences, with my latest
                    venture taking on a skydiving static line course (R.A.P.S.) which is a fancy way
                    of saying you’re flying (or falling) solo!

                </p>
            </div>
        </div>
    </div>
    <div id="contProjects" class="content show">
        <div class="slideshowContainer">
            <div class="mySlides fade">
                 <img src="{% static 'mainApp/media/portfolio/portfolio.png' %}" class="slideShowImg">
                 <div class="h2">Web Portfolio</div>
            </div>

            <div class="mySlides fade">
                <img src="{% static 'mainApp/media/renamer/renamer.png' %}" class="slideShowImg">
                <div class="h2">File Extention Renamer</div>
            </div>

            <div class="mySlides fade">
                <img src="{% static 'mainApp/media/nn/nn.png' %}" class="slideShowImg">
                <div class="h2">Number Guessing Neural Network</div>
            </div>

            <div class="mySlides fade">
                <img src="{% static 'mainApp/media/sod/sod.png' %}" class="slideShowImg">
                <div class="h2">Spread of Disease</div>
            </div>

            <div class="mySlides fade">
                <img src="{% static 'mainApp/media/ising/ising.png' %}" class="slideShowImg">
                <div class="h2">Ising Spin Model</div>
```

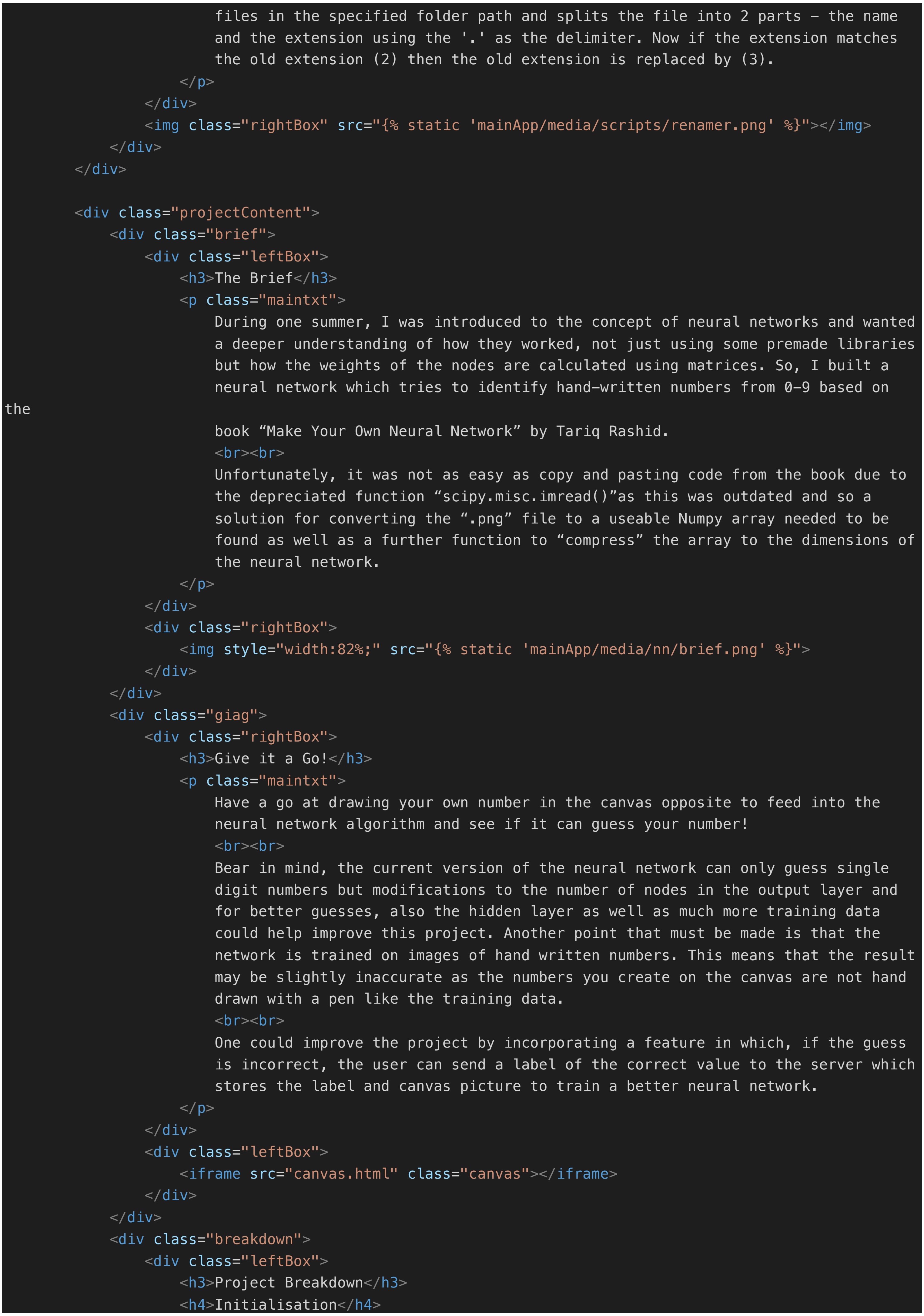
```html
            </div>

            <a class="arrowButtonPrev" onclick="plusProject(-1)">&#10094</a>
            <a class="arrowButtonNext" onclick="plusProject(1)">&#10095</a>
            <div style="text-align:center">
                <span class="slideshowDot" onclick="currentProject(1)"></span>
                <span class="slideshowDot" onclick="currentProject(2)"></span>
                <span class="slideshowDot" onclick="currentProject(3)"></span>
                <span class="slideshowDot" onclick="currentProject(4)"></span>
                <span class="slideshowDot" onclick="currentProject(5)"></span>
            </div>
        </div>

        <div class="projectContent">
            <div class="brief">
                <div class="leftBox">
                    <h3>The Brief</h3>
                    <p class="maintxt">
                        The idea of building a web portfolio was a way to take on the challenge
                        of learning new programming languages as well as the backend of web
                        development to create an interactive webpage showing my previous coding
                        projects. This project enabled me to gather a hands-on experience with
                        RaspberryPi OS (a Linux-esque operating system), HTML, CSS, JS, Django
                        and a dash of AJAX.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width: 52%" src="{% static 'mainApp/media/portfolio/brief.png' %}">
                </div>
            </div>
            <div class="giag">
                <div class="rightBox">
                    <h3>Give it a Go!</h3>
                    <p class="maintxt">
                        This section is not used in this project as you're already using it!
                        <br><br>
                        However, as an overview for this section in other projects, it can be
                        used by yourself to input your own values for variables used in the python
                        code of other projects and receive an output of that coding project back
                        to the webpage to give a bit of interactivity and flair.
                        <br><br>
                        Test it out by typing a message and clicking submit to see if you get a
                        response from the server to make sure everything is OK.
                    </p>
                </div>
                <div class="leftBox" style="height:15.81em;">
                    <input type="text" id="testMessage" class=inputTest></input>
                    <button type="button" id="testBtn">Submit</button> {% csrf_token %}
                </div>
            </div>
            <div class="breakdown">
                <div class="leftBox">
                    <h3>Project Breakdown</h3>
                    <h4>Introduction</h4>
                    <p class="maintxt">
                        Now for the technical bit! This project runs off a Raspberry Pi which is
                        being used as the subject of port-forwarding to operate as the webserver.
                        The Raspberry Pi is used headless with a Linux-based OS (RaspberryPi OS)
                        and hence all modifications to programs on the Pi was done via command
                        line SSH and SFTP connections and modification of files was done using vi
```



```
                        or through Visual Code Studio and SFTP’d to the server.
                    </p>
                    <h4>Webserver/SSL</h4>
                    <p class="maintxt">
                        Using Apache2 webserver and Openssl, virtual hosts were created and used
                        to handle http and https requests along with the Django application.
                        However, https uses a self-signed certificate (for learning reasons) and
                        so it is disliked by any machines which do not have the certificate
                        installed and trusted but be assured certificate is there and working
                        correctly using a TLS1.3 connection as seen by the image on the right.
                        The self-signed certificate was made by first issuing a certificate
                        authority (or root certificate) in which the ssl certificate could be
                        signed under. The shell script across was created and run to create the
                        ssl certificate, but, as stated earlier, will only work on client machines
                        where the root certificate has been installed and trusted.
                        <br><br>
                        So, although this was a great learning experience, it was not practical
                        for users who wanted to visit the site and were bombarded by messages
                        labelling the site as insecure due to the trust issue. This pushed me to
                        try the free options such as going through “letsencrypt.org” who provide
                        a free recognised certificate authority to sign ssl certificates.
                        <br><br><br><br>
                    </p>
                    <h4>Request Handling</h4>
                    <p class="maintxt">
                        As the ".py" files were not automatically handled by Apache, Django or
                        Flask had to be used and as Django seemed more intuitive and was already
                        written in python I went with Django. This required building a Python3
                        virtual environment and modifying the websites’ configuration file to use
                        a virtual environment to run the Python project scripts without interfering
                        with normal server functionality.
                        <br><br>
                        The next task was sending a variable to the Python scripts from the webpage
                        without needing to refresh the page. This is where AJAX came in handy.
                        Using the JQuery library, an AJAX request was sent to my switchboard script
                        on clicking the submission button which contained the data or value of the
                        input box whether this was a canvas or text input. The data arrived at the
                        switchboard URL in the form of a dictionary. This way the switchboard could
                        differentiate between each submission by the key that was used which told
                        it if it came from the test page or the neural network page and could then
                        forward the value to the correct project script. The script would then produce
                        an output and be returned to the client’s page using the HttpResponse method.
                        JavaScript would then update a div element or push an alert to the user with
                        the result of the users input.
                        <br><br>
                        For the Neural Network project, the canvas or png image had to be sent instead
                        as the input for the Python function and return a number as the guess. This
                        was done by first converting the canvas image to a URL string using the
                        “toDataURL()” method and then passed into the sever for the image to be rebuilt
                        from the URL. From the server, the views.py file decodes the base64 bytes URL
                        and converts it back to a png which is then “compressed” and fed into the
                        project script.
                        <br><br>
                        In the case of the Spread of Disease and Ising Spin Model projects, an image
                        needed to be returned to the client. This is a tad harder than just simply
                        receiving a string but similar to sending the canvas in the Neural Network
                        project as the outputted figure first needed to be encoded in base64 and then
                        passed back to the client as a URI string. This then appended to the png header
                        tag and could then be used to update an image tag’s source to show the graph.
                        <br><br><br>
```



```
                    </p>
                    <h4>HTML / JS / CSS</h4>
                    <p class="maintxt">
                        The HTML, CSS and JS was all built from scratch using a mix of my own knowledge,
                        free courses, W3 schools and Stack Overflow. The HTML files don’t differ much to
                        standard HTML files other than some necessary Django admissions such as CSRF
                        tokens and loading of “static” files which are necessary for the secure sending
                        of data to the server and locating the CSS file/images on the server
respectively.
                        <br><br>
                        The HTML file can essentially be read by right-clicking the webpage and
selecting
                        “inspect element”. This will show the HTML elements that make up the page as
well
                        as the JavaScript that enables the whole portfolio to be on a single page (not
                        the most efficient for loading times I know but I like the togetherness of it)
                        such as the simultaneous switching of CSS block values to hide and show
elements.
                        Additionally, all icons were made by myself using Pages.
                    </p>
                    <h4>Security</h4>
                    <p class="maintxt">
                        As the ports are open to external traffic, some level of security must be
implemented
                        to avoid any data breaches (even if no data apart from the most recent number
someone
                        drew is stored, it is still good practice and insightful to implement them). For
this,
                        all access to the server via SSH and SFTP is on abnormal ports along with the
                        implementation of a firewall (Uncomplicated Firewall). This has allowed only
incoming
                        traffic on the standard http and https ports to the server as well access to the
                        abnormal tcp port for ssh connections.
                        <br><br>
                        To stop just anyone guessing the password for the server, 4096 bit key-based
                        authentication is used and root access omitted and super users have password
protected
                        root access. On top of this, fail2ban is used to create jails for IP addresses
which
                        attempt to access the server multiple times to prevent brute force attacks.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="padding-top: 18%;" class="w100" src="{% static
'mainApp/media/portfolio/headless.png' %}">
                    <img class="w100"src="{% static 'mainApp/media/portfolio/selfsignedproof.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/portfolio/gencerts.png' %}">
                    <img style="padding-top: 4%" class="w100" src="{% static
'mainApp/media/portfolio/ajax.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/portfolio/imagereturn.png' %}">
                    <img style="width: 55%" src="{% static
'mainApp/media/portfolio/imagereturnsuccess.png' %}">
                    <div style="text-align: center; padding-top: 5%">
                        <a href="{% static 'mainApp/media/portfolio/html.pdf' %}" download>
                            <div class="html"></div>
                        </a>
                        <a href="{% static 'mainApp/media/portfolio/css.pdf' %}" download>
                            <div class="css"></div>
                        </a>
                    </div>
                    <img style="width: 60%; padding-top: 15%;" src="{% static
'mainApp/media/portfolio/security.png' %}">
                </div>
            </div>
        </div>

        <div class="projectContent">
            <div class="brief">
                <div class="leftBox">
                    <h3>The Brief</h3>
                    <p class="maintxt">
                        During one of my laboratory modules in University, I was tasked with
                        investigating the surfaces of different materials using an atomic force
                        microscope. The issue was that I had to take many images and the software
                        exported these in a '.dib' format whereas the analysing software I was using
                        gave poor results for this format and better results in the '.bmp' format due
                        to the types of signals each of these images used. Hence, I designed a file
                        extension renamer to save time during this operation.
                        <br><br>
                        Obviously, this is the brute-force method to get this to work as file
                        extensions cannot normally be changed in such a way. However, this solution
                        worked well for the problem I encountered.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width:76%" src="{% static 'mainApp/media/renamer/brief.png' %}">
                </div>
            </div>
            <div class="giag">
                <div class="rightBox">
                    <h3>Give it a Go!</h3>
                    <p class="maintxt">
                        Unfortunately, I don't think you'd want your own files to switch extensions
                        and so this part will not be intractable. However, on your left you will see
                        a short video of the script in use.
                        <br><br>
                        Building on this project, one may suggest modifying the code to allow the user
                        to input their own file path and file extensions to rename, but in my case,
                        this would have slowed things down as the atomic force microscopy images came
                        in large doses and rather infrequently so it was easier just to run the script
                        as and when.
                    </p>
                </div>
                <div class="leftBox">
                    <video class="w100" autoplay loop>
                        <source src="{% static 'mainApp/media/renamer/renamer.mp4' %}" type="video/mp4">
                    </video>
                </div>
            </div>
            <div class="breakdown">
                <div class="leftBox">
                    <h3>Project Breakdown</h3>
                    <p class="maintxt">
                        As seen looking at the code on the right, this one is pretty simple. Firstly,
                        it imports the “os” library which allows modification of the system files etc.
                        Then the variables of 1) A string of the path to the folder containing the files
                        to be renamed, 2) A string of the files’ extension that you want to rename, and
                        3) The string of the new file extension.
                        <br><br>
                        The for loop just loops over all the
```

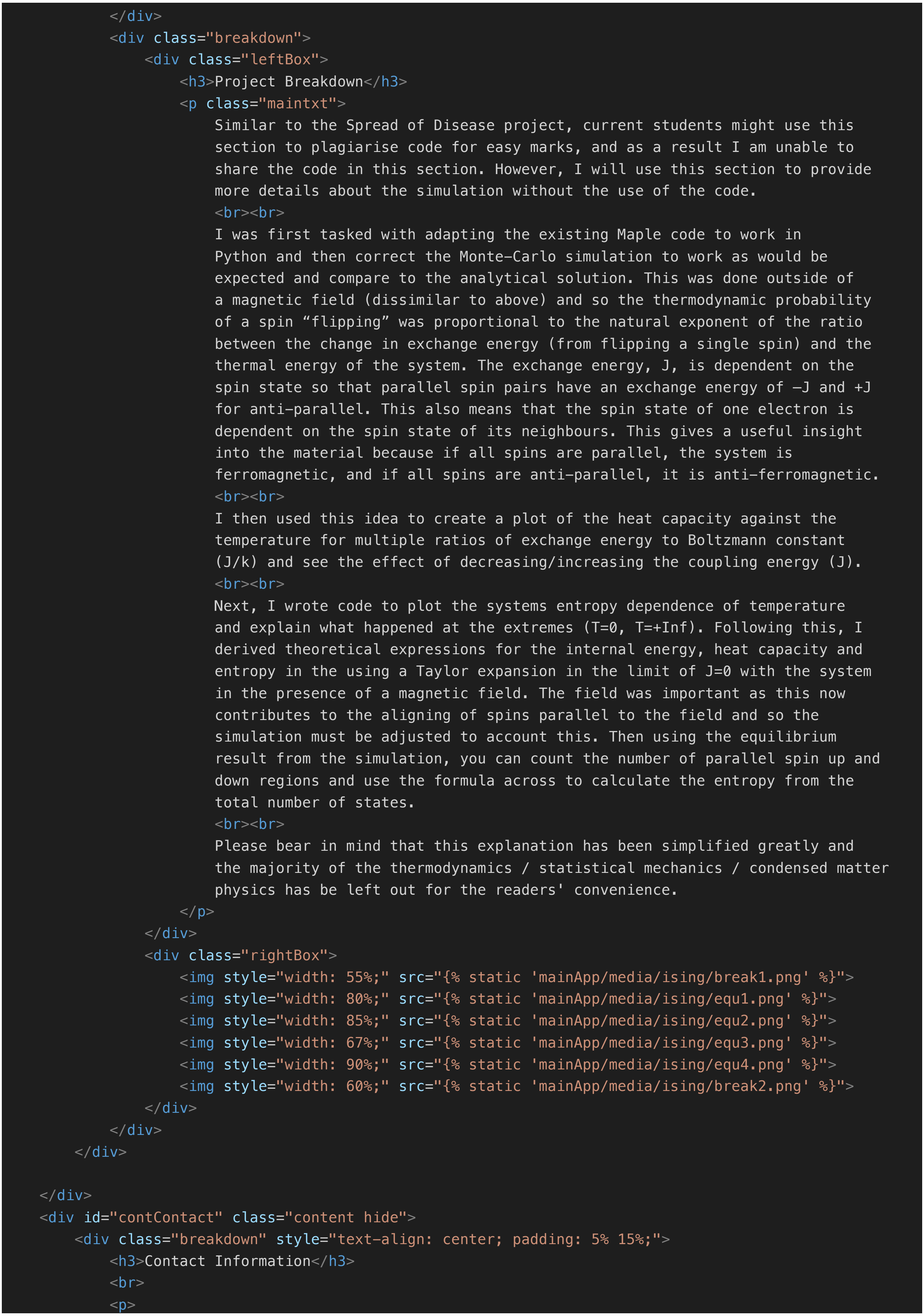


```
            files in the specified folder path and splits the file into 2 parts - the name
            and the extension using the '.' as the delimiter. Now if the extension matches
            the old extension (2) then the old extension is replaced by (3).
          </p>
        </div>
        <img class="rightBox" src="{% static 'mainApp/media/scripts/renamer.png' %}"></img>
      </div>
    </div>

    <div class="projectContent">
      <div class="brief">
        <div class="leftBox">
          <h3>The Brief</h3>
          <p class="maintxt">
            During one summer, I was introduced to the concept of neural networks and wanted
            a deeper understanding of how they worked, not just using some premade libraries
            but how the weights of the nodes are calculated using matrices. So, I built a
            neural network which tries to identify hand-written numbers from 0-9 based on
the
            book “Make Your Own Neural Network” by Tariq Rashid.
            <br><br>
            Unfortunately, it was not as easy as copy and pasting code from the book due to
            the depreciated function “scipy.misc.imread()”as this was outdated and so a
            solution for converting the “.png” file to a useable Numpy array needed to be
            found as well as a further function to “compress” the array to the dimensions of
            the neural network.
          </p>
        </div>
        <div class="rightBox">
          <img style="width:82%;" src="{% static 'mainApp/media/nn/brief.png' %}">
        </div>
      </div>
      <div class="giag">
        <div class="rightBox">
          <h3>Give it a Go!</h3>
          <p class="maintxt">
            Have a go at drawing your own number in the canvas opposite to feed into the
            neural network algorithm and see if it can guess your number!
            <br><br>
            Bear in mind, the current version of the neural network can only guess single
            digit numbers but modifications to the number of nodes in the output layer and
            for better guesses, also the hidden layer as well as much more training data
            could help improve this project. Another point that must be made is that the
            network is trained on images of hand written numbers. This means that the result
            may be slightly inaccurate as the numbers you create on the canvas are not hand
            drawn with a pen like the training data.
            <br><br>
            One could improve the project by incorporating a feature in which, if the guess
            is incorrect, the user can send a label of the correct value to the server which
            stores the label and canvas picture to train a better neural network.
          </p>
        </div>
        <div class="leftBox">
          <iframe src="canvas.html" class="canvas"></iframe>
        </div>
      </div>
      <div class="breakdown">
        <div class="leftBox">
          <h3>Project Breakdown</h3>
          <h4>Initialisation</h4>
```

```
                    <p class="maintxt">
                        Firstly, let’s check out the neural network class. When the class is first
called,
                        some variables are initially set. These are the number of input, hidden and
output
                        nodes which are used to create a matrix of weights between each of the layers of
                        nodes. These weights are initially set using random values from a normal
                        distribution about the mean of 0 and a standard deviation based on the number of
                        nodes in the next layer. The probability distribution of the normal function can
                        be seen below:
                    </p>
                    <img style="width:35%;" src="{% static 'mainApp/media/nn/equ-1.png' %}">
                    <p class="maintxt">
                        The returned matrix is a many-to-many representation so that each node in the
                        first layer has associated weights with all nodes in the next layer. Then the
                        activation function is made using the expit function to decide whether the
signal
                        passed through to the nodes is strong enough to be counted as this converts all
                        signals to the range 0 to 1. The expit function can be seen below:
                    </p>
                    <img style="width:25%;" src="{% static 'mainApp/media/nn/equ-2.png' %}">
                    <h4>Training</h4>
                    <p class="maintxt">
                        The training function takes the inputs of the training data (training_list)
                        along with their labelled answers (targets_list) and a save flag. For this
                        project, the training data is a “.txt” or “.csv” file containing the numbers’
                        label turned into an array where it is the only possible answer (e.g. “6” ->
                        “0,0,0,0,0,0,1,0,0,0”), and comma separated greyscale colour values for the
                        28x28 image.
                        <br><br>
                        The training data array is then used to calculate the signals passing into and
                        out of each neural network layer utilising the activation function mentioned
                        earlier. Then the error from the neural network needs to be calculated for each
                        layer and back-propagated to update the values of the node weights. This is done
                        using via a scaling factor or learning rate as to not update the weights too
                        greatly where they can produce much larger errors and miss the optimum weight
                        values. The formula for updating the weights follows that for a large error, the
                        final outputs will be small (but not zero) and so will be updated correctly over
                        multiple iterations. For a small error, the updating of the weights will be
small
                        and should get progressively smaller with every iteration tending to a maximum.
                        <br><br>
                        The save flag dictates whether files containing the weights of each layer in the
                        network after it has finished training will be saved. This sets a foundation for
                        the use of the neural network to be queried without running the very lengthy
                        training process.
                    </p>
                    <h4 style="padding-top: 8%">Querying</h4>
                    <p class="maintxt">
                        The query function takes the comma separated greyscale image values
(inputs_list)
                        and a trained network flag. If the trained network flag is false, the train
                        function must have been used previously to train the network. The signals are
then
                        passed through each layer via a dot product with the weights and the results
                        undergo the activation function before returning a 1x10 array (representing each
                        possible single digit number) with values of the networks probability for each
                        number as the final outputs.
                        <br><br><br><br><br><br><br><br><br><br><br><br>
                        If the trained network flag is true, the previously saved files of the trained
                        network weights are opened and read. They undergo some processing to put the
values
                        into the correct format of the “Wih” and “Who” arrays, then are used for
calculating
                        the signals passing through each layer the same way as before, again, returning
the
                        final output probability array.
                    </p>
                    <h4 style="padding-top: 27%">Compression</h4>
                    <p class="maintxt">
                        As the training data was 28x28 pixels, the neural network was setup to handle
this
                        input but the image received from the webpage was much larger, a compression
function
                        had to be made for the queried image file to work with the neural network. This
was
                        much more pleasing then to have the user draw their number in a 28x28 pixel
canvas.
                        <br><br>
                        It works by first calculating the size ratio of the square input image to the
square
                        compressed resolution the user wants. The first for loop counts each compressed
                        resolution pixel along the x axis and the second for loop does the same for the
y
                        axis. Inside the loop, the input image is cut into segments with area of the
ratio
                        squared and greyscale values normalised. Then the greyscale values of these
segments
                        are summed to a single value which is appended to a list. This process is
repeated
                        over the whole image until the list of single values is put into an array and
reshaped
                        into the compressed resolution. Returning the compressed image as an array of
greyscale
                        values.
                    </p>
                    <h4>Switchboard Function</h4>
                    <p class="maintxt">
                        Now I have to piece it all together inside the views.py and switchboard.py files
                        inside the Django web app. I start with the AJAX request sending the canvas
base64
                        data to the switchboard URL. The AJAX dictionary is checked to make sure the
POST
                        request is from the canvas and then the base64 decoded data is written to a file
cutting
                        off the png header.
                        <br><br><br><br><br><br><br><br><br><br><br><br><br>
                        The file is then used as the input to the trainedNetwork function inside the
switchboard
                        where the learning rate and layer node variables are set. The input image,
although black
                        and white, still has RGB colour values so only the greyscale colour value of the
array is
                        taken and then used to create the compressed 28x28 image. The image is then put
into a 1-D
                        array and the greyscale values modified so that instead of pure white being 255,
it is 0,
                        and pure black once 0 is now 1, to replicate the training data. Then the neural
network is
```

```
                        initialised and queried using trained weights (as the trainedNetwork flag is
true) and the
                        largest value of the output probability array is chosen as the guessed label and
returned
                        to the webpage as a HttpResponse.
                    </p>
                </div>
                <div class="rightBox">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-init.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-train.png' %}"
style="padding-top: 5%">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-query-if.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-query-else.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-compress.png' %}">
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-urls.png' %}"
style="padding-top: 10%" >
                    <img class="w100" src="{% static 'mainApp/media/scripts/nn-switchboard.png' %}">

                </div>
            </div>
        </div>

        <div class="projectContent">
            <div class="brief">
                <div class="leftBox">
                    <h3>The Brief</h3>
                    <p class="maintxt">
                        During my "Molecular Dynamics: Theory and Simulation" module, I had to take
                        on multiple projects to boost our understanding of implementing models for
                        mathematical simulations. In this project, I was tasked with producing graphs
                        which modelled an epidemic (topical I know!) based on the S.I.R. model. This
model
                        assumes one or more infected persons are introduced to a community where all
                        are equally susceptible to the disease, it is spread by contact and the epidemic
                        lasts much shorter than the life expectancy of the persons (so the population
                        remains the same).
                        <br><br>
                        I then used a Monte-Carlo simulation to solve for the number of
                        infected students and the number of susceptible students, day-by-day, from a
                        larger school population. Within the school there could be older students which
                        already had the infection and are immune to the disease, some who are
                        susceptible and others that are infected. It was then required that I calculate
                        the size of the epidemic (total number of students infected).
                        <br><br>
                        In this project, I scored 10/10.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width: 60%; padding: 22% 0%;" src="{% static
'mainApp/media/sod/brief.png' %}">
                </div>
            </div>
            <div class="giag">
                <div class="leftBox">
                    <div class="sodbox">
                        <input type="number" id="pop" class="inputSod" value="1000">Population</input>
                        <input type="number" id="fract" class="inputSod" value="0.01">Immune
Fraction</input>
                        <input type="number" id="recover" class="inputSod" value="0.2">Recovery
Chance</input>
```

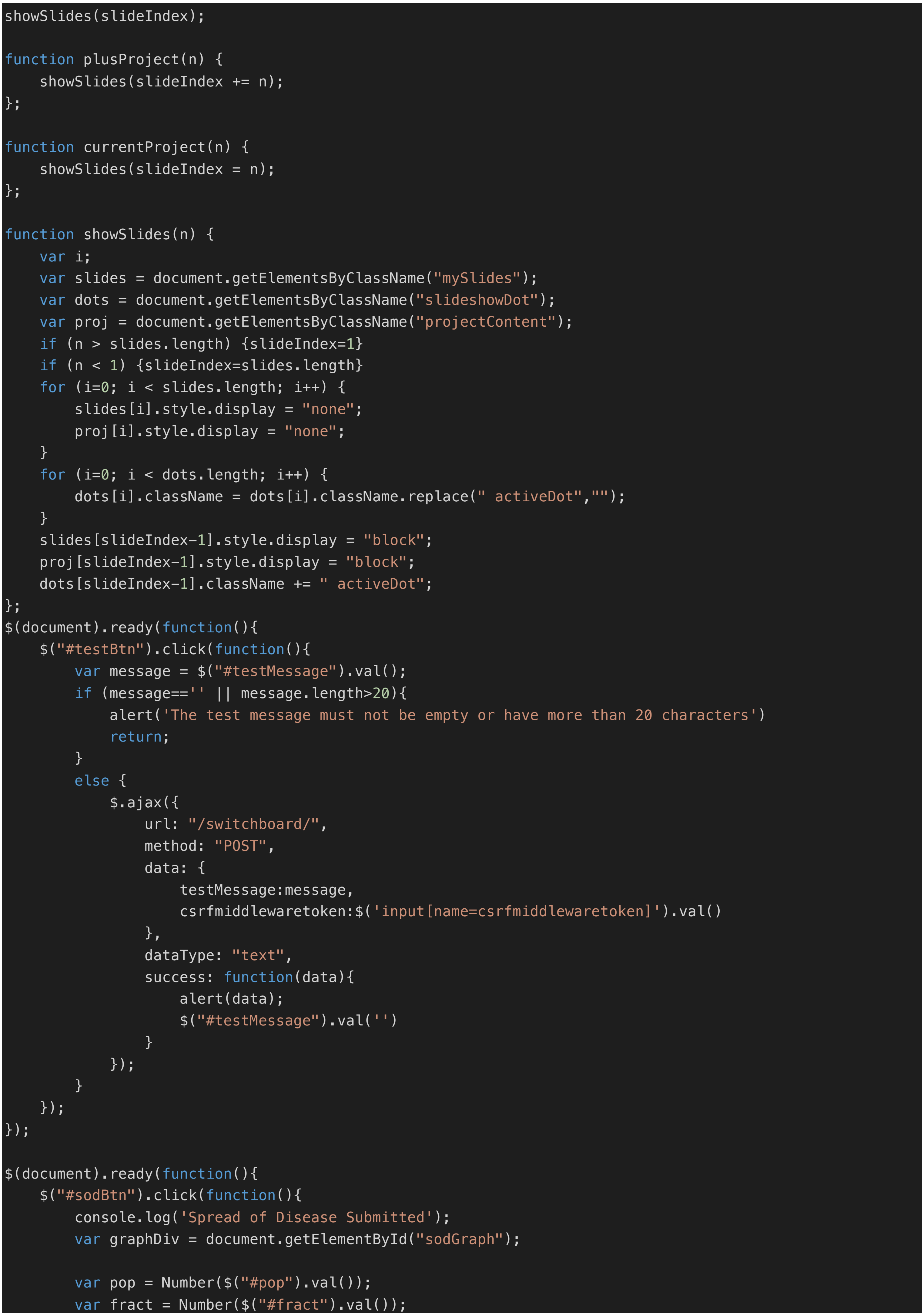
```
                    </div>
                    <div class="sodbox">
                        <input type="number" id="infections" class="inputSod"
value="1">Infections</input>
                        <input type="number" id="numd" class="inputSod" value="40">Epidemic
Length</input>
                        <input type="number" id="reps" class="inputSod" value="20">Repeats</input>
                    </div>
                    <div class="sodbox">
                        <input type="number" id="reinfect" class="inputSod" value="0">Re-infect</input>
                    </div class="sodbox">
                    <button class="submit" type="button" id="sodBtn">Submit</button> {% csrf_token %}
                    <div class="loader"></div>
                    <img id="sodGraph" class="graph" src='' >
                </div>
                <div class="rightBox">
                    <h3>Give it a Go!</h3>
                    <p class="maintxt">
                        Choose your own variables to pass to the Spread of Disease project code. The
result
                        will be a graph which shows the length and size of the epidemic. The size is
shown
                        by the total number of those infected over the duration of the epidemic, the
number
                        of those susceptible and the daily infection count.
                        <br><br>
                        The variables are:
                    </p>
                    <ul style="text-align: left;">
                        <li>Population - The total population of the sample.</li><br>
                        <li>Immune Fraction - The fraction of the total population that is
immune.</li><br>
                        <li>Recovery Chance - The probability each day that the infected person will
recover.</li><br>
                        <li>Infections - The number of people the infection can spread to each
day.</li><br>
                        <li>Epidemic Length - The data range of the epidemic to view.</li><br>
                        <li>Repeats - Number of repeat Monte-Carlo simulations to run to provide
averages.</li><br>
                        <li>Re-Infect - Model if the recovered become immune (0) or susceptible again
(1).</li>
                    </ul>
                </div>
            </div>
            <div class="breakdown">
                <div class="leftBox">
                    <h3>Project Breakdown</h3>
                    <p class="maintxt">
                        Unfortunately, current students might use this section to plagiarise code for
                        easy marks, so, I am unable to share the code in this section. However, I will
                        use this section to provide more details about the simulation without the use
                        of the code.
                        <br><br>
                        The theory as stated in the brief is based on the S.I.R. model where the
                        population is either susceptible (S), infected (I) or removed (R). The scheme
                        follows the equations across:
                        <br><br>
                        Where the first step describes the transmission of the disease and the second
                        step describes the recovery from the infection with k2 and k1 as the respective
                        rate constants for each step.
                        <br><br>
                        In terms of the code, initially variables are initialised and a matrix
                        containing the three categories of student infection are made where 0 represents
                        immune, 1 is susceptible and 2 is infected. Then, loop through the population
                        matrix so when an infected person is found, choose at random someone to try and
                        infect if not immune. Then, for the infected person initially found, undergo a
                        Monte-Carlo simulation to see whether the student recovers and make them immune
                        and continue to loop through the population matrix and do this for each day of
                        the epidemic.
                        <br><br>
                        The Monte-Carlo simulations done in this task use the law of random numbers and
                        proportionality to see if, for example, someone recovers from the infection. In
                        this case, you can see if a number that is randomly generated from 0 to 1 is
                        within the limit to be cured (less than the recovery probability). Over many
                        iterations, and if the numbers are indeed truly randomly generated, you can
                        prove that the proportion of those iterations that lie under the boundary is
                        that of the recovery probability.
                        <br><br>
                        This is a very basic model and so is not as accurate and one may like,
                        although, it is a good estimation. One could build upon the model to include
                        enhancements such as, for a deadly disease, a chance of death with another
                        Monte-Carlo simulation and include a death count in the graph. Another
                        inclusion could be a Monte-Carlo simulation for the generation of a medicine
                        and one to include the probability of someone actually getting the medicine
                        which could provide immunity or even just improve the recovery chance and
                        hence altering the graph to see a sharp decline in infections after the
                        creation of the medicine.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width: 50%; padding: 4% 0%;" src="{% static 'mainApp/media/sod/equ1.png'
%}">
                    <img style="width: 50%; padding: 4% 0%;" src="{% static 'mainApp/media/sod/equ2.png'
%}">
                    <img style="width: 50%; padding: 6% 0%;" src="{% static
'mainApp/media/sod/break.png' %}">
                    <p style="padding: 0% 15%;">
                        Monte-Carlo simulation to estimate pi based on the number ratio of points
                        inside a circle to the total points. Points within the the circle can be
                        identified using the equation of a circle as an inequality.
                    </p>
                </div>
            </div>
        </div>

        <div class="projectContent">
            <div class="brief">
                <div class="leftBox">
                    <h3>The Brief</h3>
                    <p class="maintxt">
                        Another completed project in the Molecular Dynamics module featured 1-D
                        Spins and the Ising model. This model shows how a 1-D spin chain of spin
                        “up” and spin “down” electrons behaves under the exchange interaction,
                        and is used to predict macroscopic magnetic properties of solids. A single
                        spin in the chain may flip from up to down due to an interaction with its
                        neighbour which will then in turn influence its neighbours and hence
                        propagate along the chain until an equilibrium is reached. The spin state
                        of an electron depends on the ratio of the exchange energy to the thermal
                        energy.
                        <br><br>
                        I was then tasked to modify pseudo code and perform Monte-Carlo
                        simulations to calculate the heat capacity and compare it to the analytical
                        solution. I also had to calculate the entropy of the equilibrium state by
                        counting the number of like spin-chains and show this was a good estimate
                        when compared to the analytical approach.
                        <br><br>
                        In this project, I scored 9.5/10.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width: 75%; padding: 10% 0%;" src="{% static
'mainApp/media/ising/brief.png' %}">
                </div>
            </div>
            <div class="giag">
                <div class="leftBox">
                    <div class="sodbox">
                        <input type="number" id="sites" class="inputSod" value="500">Sites</input>
                        <input type="number" id="temp" class="inputSod" value="10">Temperature</input>
                        <input type="number" id="mag" class="inputSod" value="0.1">B-Field</input>
                    </div>
                    <div class="sodbox">
                        <input type="number" id="mom" class="inputSod" value="3.63">Magnetic
Moment</input>
                        <input type="number" id="calc" class="inputSod" value="200">Calculations</input>
                        <input type="number" id="points" class="inputSod" value="10">Points</input>
                    </div>
                    <button class="submit" type="button" id="isingBtn">Submit</button> {% csrf_token %}
                    <div class="loader"></div>
                    <img id="isingGraph" class="graph" src='' >
                </div>
                <div class="rightBox">
                    <h3>Give it a Go!</h3>
                    <p class="maintxt">
                        As there are a lot of different stages to this project, this GIAG will only
                        focus on one of them and produce a plot of the internal energy, heat
                        capacity and entropy against the sample temperature of a ferromagnetic solid
                        inside a magnetic field. It will also compare the results of the analytical
                        solution which is found via a Taylor expansion with the Monte-Carlo
                        simulation to show that the simulation gives an accurate representation and
                        would be much faster if up-scaled.
                        <br><br>
                        Please note that this simulation may take a long time depending to the
                        variables chosen and this may cause a timeout with the server. This
                        timeout setting has been included to drop long connections and free up
                        worker threads.
                        <br><br>
                        The variables used to create the graph are shown below:
                    </p>
                    <ul style="text-align: left;">
                        <li>Sites - Number of sites / spins to solve for.</li>
                        <li>Temperature - Temperature range from 0.1 Kelvin to this value (must be a low
temperature).</li>
                        <li>B-Field - Magnetic Field (in Tesla).</li>
                        <li>Magnetic Moment - Magnetic moment (in μB, default is Iron).</li>
                        <li>Calculations - Number of Monte-Carlo calculations per site.</li>
                        <li>Points - Number of temperature points to be plotted in the specified
range.</li>
                    </ul>
                </div>
```



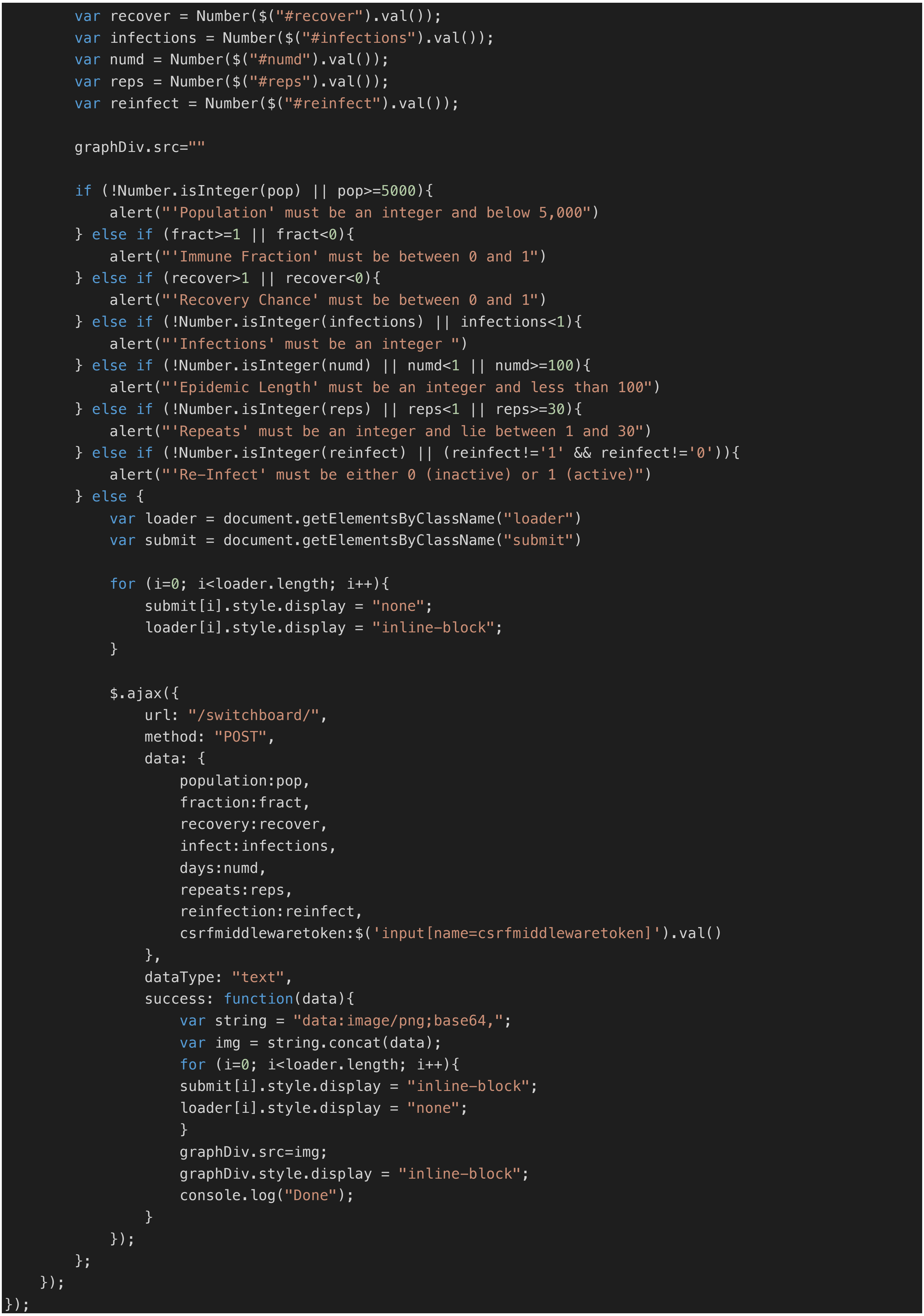
```
            </div>
            <div class="breakdown">
                <div class="leftBox">
                    <h3>Project Breakdown</h3>
                    <p class="maintxt">
                        Similar to the Spread of Disease project, current students might use this
                        section to plagiarise code for easy marks, and as a result I am unable to
                        share the code in this section. However, I will use this section to provide
                        more details about the simulation without the use of the code.
                        <br><br>
                        I was first tasked with adapting the existing Maple code to work in
                        Python and then correct the Monte-Carlo simulation to work as would be
                        expected and compare to the analytical solution. This was done outside of
                        a magnetic field (dissimilar to above) and so the thermodynamic probability
                        of a spin “flipping” was proportional to the natural exponent of the ratio
                        between the change in exchange energy (from flipping a single spin) and the
                        thermal energy of the system. The exchange energy, J, is dependent on the
                        spin state so that parallel spin pairs have an exchange energy of -J and +J
                        for anti-parallel. This also means that the spin state of one electron is
                        dependent on the spin state of its neighbours. This gives a useful insight
                        into the material because if all spins are parallel, the system is
                        ferromagnetic, and if all spins are anti-parallel, it is anti-ferromagnetic.
                        <br><br>
                        I then used this idea to create a plot of the heat capacity against the
                        temperature for multiple ratios of exchange energy to Boltzmann constant
                        (J/k) and see the effect of decreasing/increasing the coupling energy (J).
                        <br><br>
                        Next, I wrote code to plot the systems entropy dependence of temperature
                        and explain what happened at the extremes (T=0, T=+Inf). Following this, I
                        derived theoretical expressions for the internal energy, heat capacity and
                        entropy in the using a Taylor expansion in the limit of J=0 with the system
                        in the presence of a magnetic field. The field was important as this now
                        contributes to the aligning of spins parallel to the field and so the
                        simulation must be adjusted to account this. Then using the equilibrium
                        result from the simulation, you can count the number of parallel spin up and
                        down regions and use the formula across to calculate the entropy from the
                        total number of states.
                        <br><br>
                        Please bear in mind that this explanation has been simplified greatly and
                        the majority of the thermodynamics / statistical mechanics / condensed matter
                        physics has be left out for the readers' convenience.
                    </p>
                </div>
                <div class="rightBox">
                    <img style="width: 55%;" src="{% static 'mainApp/media/ising/break1.png' %}">
                    <img style="width: 80%;" src="{% static 'mainApp/media/ising/equ1.png' %}">
                    <img style="width: 85%;" src="{% static 'mainApp/media/ising/equ2.png' %}">
                    <img style="width: 67%;" src="{% static 'mainApp/media/ising/equ3.png' %}">
                    <img style="width: 90%;" src="{% static 'mainApp/media/ising/equ4.png' %}">
                    <img style="width: 60%;" src="{% static 'mainApp/media/ising/break2.png' %}">
                </div>
            </div>
        </div>

    </div>
    <div id="contContact" class="content hide">
        <div class="breakdown" style="text-align: center; padding: 5% 15%;">
            <h3>Contact Information</h3>
            <br>
            <p>
```

```
                Please feel free to contact me if you have any questions about any of the
                projects or even about myself at: py16mg@leeds.ac.uk
                <br><br>
                Alternatively, check out my Linked-In profile or 2-page CV below.
            </p>
        </div>
        <div class="giag" style="padding: 2%;">
            <div class="contImages">
                <a href="mailto:py16mg@leeds.ac.uk">
                    <div class="email"></div>
                </a>
            </div>
            <div class="contImages">
                <a href="https://www.linkedin.com/in/max-gamill/" target="_blank">
                    <img src="{% static 'mainApp/media/contact/linkedin.png' %}" class="linkedin">
                </a>
            </div>
            <div class="contImages">
                <a href="{% static 'mainApp/media/contact/cv.pdf' %}" download>
                    <div class="cv"></div>
                </a>
            </div>
        </div>
    </div>
</div>
```

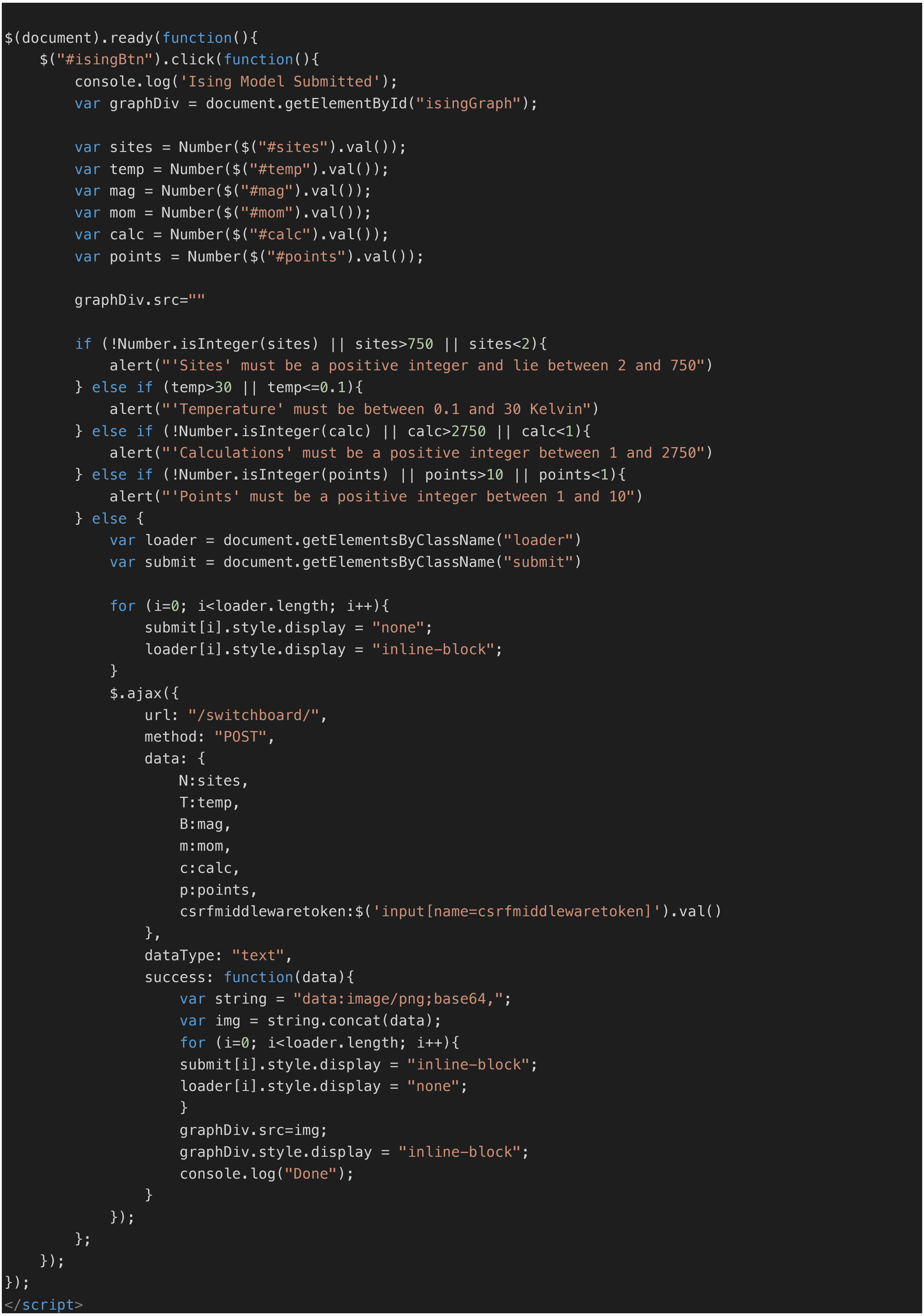
```
<script>
//Changes the active button and active content block to show contents of nav buttons

var cont = document.getElementById("cont").getElementsByClassName("content");
var btns = document.getElementById("menu").getElementsByClassName("navBlockMenu");
for (var i = 0; i < btns.length; i++) {
    btns[i].addEventListener("click", function() {
        var current = document.getElementsByClassName("active");
        var showCont = document.getElementsByClassName("show");
        if (current.length > 0) {
            current[0].className = current[0].className.replace(" active", "");
            showCont[0].className = showCont[0].className.replace(" show"," hide");
        }
        this.className += " active";

        var clicked;
        switch(this.id) {
            case "about":
                var clicked = document.getElementById("contAbout");
                break;
            case "projects":
                var clicked = document.getElementById("contProjects");
                break;
            case "contact":
                var clicked = document.getElementById("contContact");
                break;
        }
        clicked.className = clicked.className.replace(" hide", " show");
    })
};

//change project on click of arrow or dot
var slideIndex = 1;
```



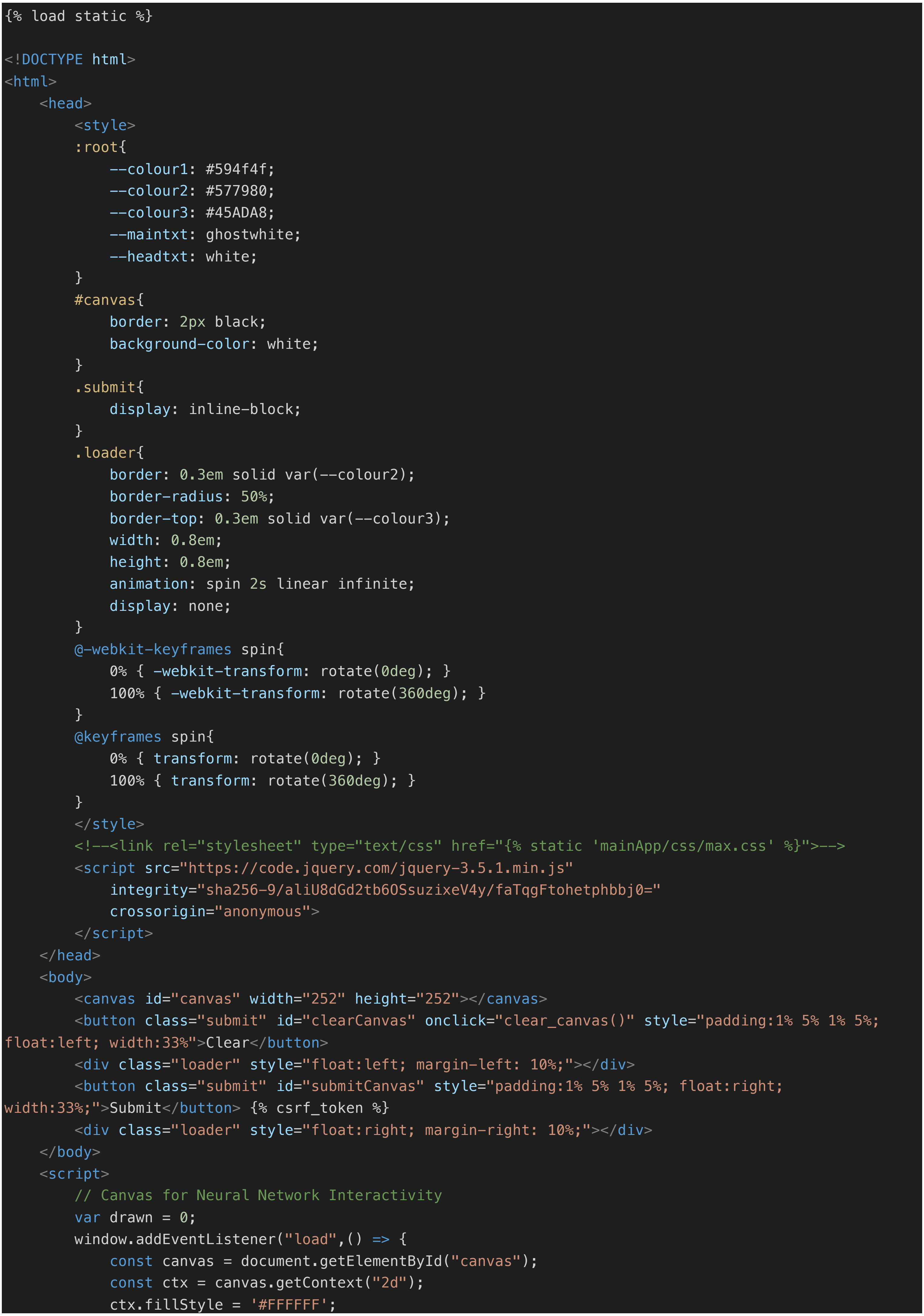
```
showSlides(slideIndex);

function plusProject(n) {
    showSlides(slideIndex += n);
};

function currentProject(n) {
    showSlides(slideIndex = n);
};

function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("slideshowDot");
    var proj = document.getElementsByClassName("projectContent");
    if (n > slides.length) {slideIndex=1}
    if (n < 1) {slideIndex=slides.length}
    for (i=0; i < slides.length; i++) {
        slides[i].style.display = "none";
        proj[i].style.display = "none";
    }
    for (i=0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" activeDot","");
    }
    slides[slideIndex-1].style.display = "block";
    proj[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " activeDot";
};
$(document).ready(function(){
    $("#testBtn").click(function(){
        var message = $("#testMessage").val();
        if (message=='' || message.length>20){
            alert('The test message must not be empty or have more than 20 characters')
            return;
        }
        else {
            $.ajax({
                url: "/switchboard/",
                method: "POST",
                data: {
                    testMessage:message,
                    csrfmiddlewaretoken:$('input[name=csrfmiddlewaretoken]').val()
                },
                dataType: "text",
                success: function(data){
                    alert(data);
                    $("#testMessage").val('')
                }
            });
        }
    });
});

$(document).ready(function(){
    $("#sodBtn").click(function(){
        console.log('Spread of Disease Submitted');
        var graphDiv = document.getElementById("sodGraph");

        var pop = Number($("#pop").val());
        var fract = Number($("#fract").val());
```



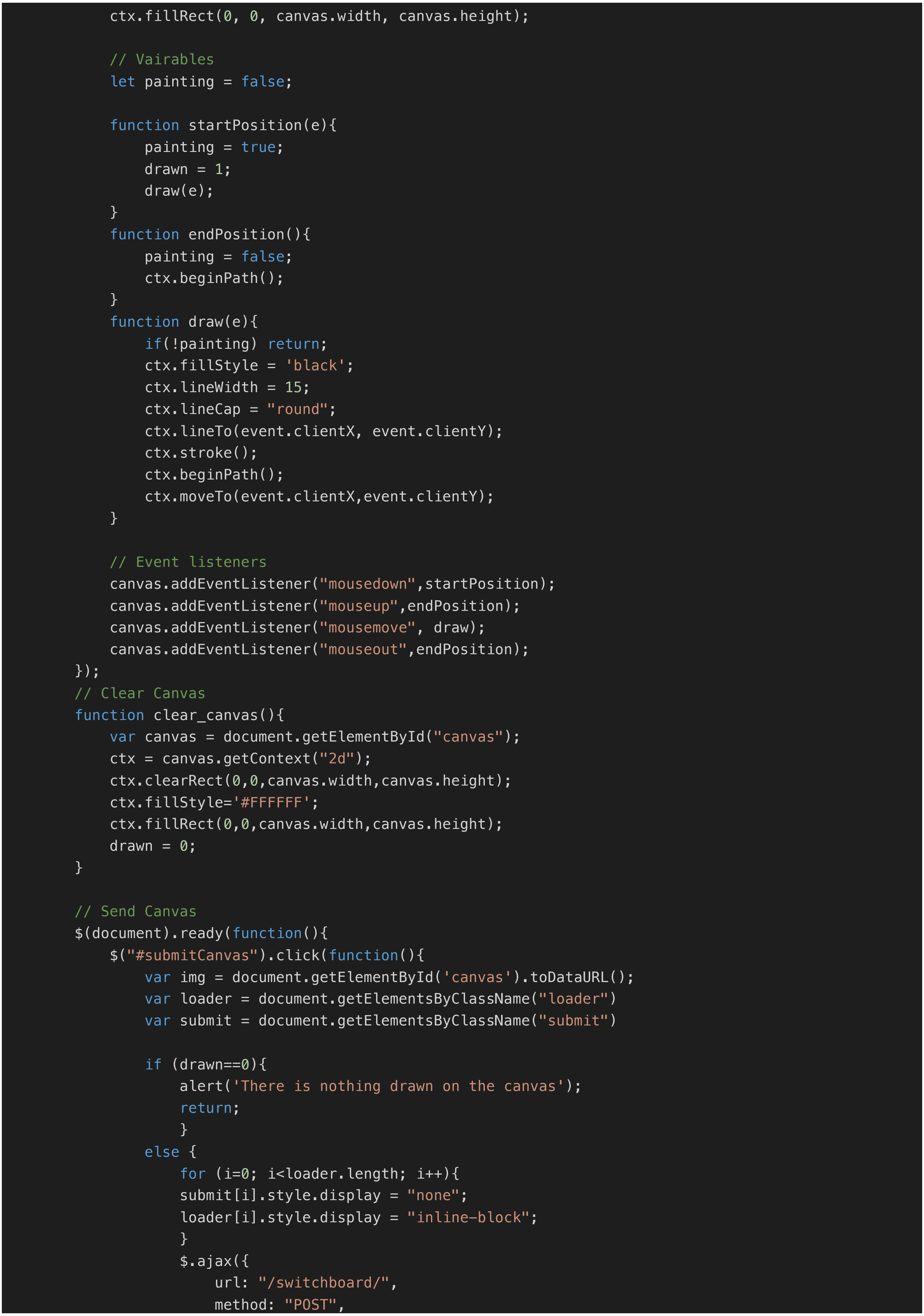
```
        var recover = Number($("#recover").val());
        var infections = Number($("#infections").val());
        var numd = Number($("#numd").val());
        var reps = Number($("#reps").val());
        var reinfect = Number($("#reinfect").val());

        graphDiv.src=""

        if (!Number.isInteger(pop) || pop>=5000){
            alert("'Population' must be an integer and below 5,000")
        } else if (fract>=1 || fract<0){
            alert("'Immune Fraction' must be between 0 and 1")
        } else if (recover>1 || recover<0){
            alert("'Recovery Chance' must be between 0 and 1")
        } else if (!Number.isInteger(infections) || infections<1){
            alert("'Infections' must be an integer ")
        } else if (!Number.isInteger(numd) || numd<1 || numd>=100){
            alert("'Epidemic Length' must be an integer and less than 100")
        } else if (!Number.isInteger(reps) || reps<1 || reps>=30){
            alert("'Repeats' must be an integer and lie between 1 and 30")
        } else if (!Number.isInteger(reinfect) || (reinfect!='1' && reinfect!='0')){
            alert("'Re-Infect' must be either 0 (inactive) or 1 (active)")
        } else {
            var loader = document.getElementsByClassName("loader")
            var submit = document.getElementsByClassName("submit")

            for (i=0; i<loader.length; i++){
                submit[i].style.display = "none";
                loader[i].style.display = "inline-block";
            }

            $.ajax({
                url: "/switchboard/",
                method: "POST",
                data: {
                    population:pop,
                    fraction:fract,
                    recovery:recover,
                    infect:infections,
                    days:numd,
                    repeats:reps,
                    reinfection:reinfect,
                    csrfmiddlewaretoken:$('input[name=csrfmiddlewaretoken]').val()
                },
                dataType: "text",
                success: function(data){
                    var string = "data:image/png;base64,";
                    var img = string.concat(data);
                    for (i=0; i<loader.length; i++){
                    submit[i].style.display = "inline-block";
                    loader[i].style.display = "none";
                    }
                    graphDiv.src=img;
                    graphDiv.style.display = "inline-block";
                    console.log("Done");
                }
            });
        };
    });
});
```



```
$(document).ready(function(){
    $("#isingBtn").click(function(){
        console.log('Ising Model Submitted');
        var graphDiv = document.getElementById("isingGraph");

        var sites = Number($("#sites").val());
        var temp = Number($("#temp").val());
        var mag = Number($("#mag").val());
        var mom = Number($("#mom").val());
        var calc = Number($("#calc").val());
        var points = Number($("#points").val());

        graphDiv.src=""

        if (!Number.isInteger(sites) || sites>750 || sites<2){
            alert("'Sites' must be a positive integer and lie between 2 and 750")
        } else if (temp>30 || temp<=0.1){
            alert("'Temperature' must be between 0.1 and 30 Kelvin")
        } else if (!Number.isInteger(calc) || calc>2750 || calc<1){
            alert("'Calculations' must be a positive integer between 1 and 2750")
        } else if (!Number.isInteger(points) || points>10 || points<1){
            alert("'Points' must be a positive integer between 1 and 10")
        } else {
            var loader = document.getElementsByClassName("loader")
            var submit = document.getElementsByClassName("submit")

            for (i=0; i<loader.length; i++){
                submit[i].style.display = "none";
                loader[i].style.display = "inline-block";
            }
            $.ajax({
                url: "/switchboard/",
                method: "POST",
                data: {
                    N:sites,
                    T:temp,
                    B:mag,
                    m:mom,
                    c:calc,
                    p:points,
                    csrfmiddlewaretoken:$('input[name=csrfmiddlewaretoken]').val()
                },
                dataType: "text",
                success: function(data){
                    var string = "data:image/png;base64,";
                    var img = string.concat(data);
                    for (i=0; i<loader.length; i++){
                    submit[i].style.display = "inline-block";
                    loader[i].style.display = "none";
                    }
                    graphDiv.src=img;
                    graphDiv.style.display = "inline-block";
                    console.log("Done");
                }
            });
        };
    });
});
</script>
```



```
{% load static %}

<!DOCTYPE html>
<html>
    <head>
        <style>
        :root{
            --colour1: #594f4f;
            --colour2: #577980;
            --colour3: #45ADA8;
            --maintxt: ghostwhite;
            --headtxt: white;
        }
        #canvas{
            border: 2px black;
            background-color: white;
        }
        .submit{
            display: inline-block;
        }
        .loader{
            border: 0.3em solid var(--colour2);
            border-radius: 50%;
            border-top: 0.3em solid var(--colour3);
            width: 0.8em;
            height: 0.8em;
            animation: spin 2s linear infinite;
            display: none;
        }
        @-webkit-keyframes spin{
            0% { -webkit-transform: rotate(0deg); }
            100% { -webkit-transform: rotate(360deg); }
        }
        @keyframes spin{
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }
        </style>
        <!--<link rel="stylesheet" type="text/css" href="{% static 'mainApp/css/max.css' %}">-->
        <script src="https://code.jquery.com/jquery-3.5.1.min.js"
            integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
            crossorigin="anonymous">
        </script>
    </head>
    <body>
        <canvas id="canvas" width="252" height="252"></canvas>
        <button class="submit" id="clearCanvas" onclick="clear_canvas()" style="padding:1% 5% 1% 5%; float:left; width:33%">Clear</button>
        <div class="loader" style="float:left; margin-left: 10%;"></div>
        <button class="submit" id="submitCanvas" style="padding:1% 5% 1% 5%; float:right; width:33%;">Submit</button> {% csrf_token %}
        <div class="loader" style="float:right; margin-right: 10%;"></div>
    </body>
    <script>
        // Canvas for Neural Network Interactivity
        var drawn = 0;
        window.addEventListener("load",() => {
            const canvas = document.getElementById("canvas");
            const ctx = canvas.getContext("2d");
            ctx.fillStyle = '#FFFFFF';
```



```
    ctx.fillRect(0, 0, canvas.width, canvas.height);

    // Vairables
    let painting = false;

    function startPosition(e){
        painting = true;
        drawn = 1;
        draw(e);
    }
    function endPosition(){
        painting = false;
        ctx.beginPath();
    }
    function draw(e){
        if(!painting) return;
        ctx.fillStyle = 'black';
        ctx.lineWidth = 15;
        ctx.lineCap = "round";
        ctx.lineTo(event.clientX, event.clientY);
        ctx.stroke();
        ctx.beginPath();
        ctx.moveTo(event.clientX,event.clientY);
    }

    // Event listeners
    canvas.addEventListener("mousedown",startPosition);
    canvas.addEventListener("mouseup",endPosition);
    canvas.addEventListener("mousemove", draw);
    canvas.addEventListener("mouseout",endPosition);
});
// Clear Canvas
function clear_canvas(){
    var canvas = document.getElementById("canvas");
    ctx = canvas.getContext("2d");
    ctx.clearRect(0,0,canvas.width,canvas.height);
    ctx.fillStyle='#FFFFFF';
    ctx.fillRect(0,0,canvas.width,canvas.height);
    drawn = 0;
}

// Send Canvas
$(document).ready(function(){
    $("#submitCanvas").click(function(){
        var img = document.getElementById('canvas').toDataURL();
        var loader = document.getElementsByClassName("loader")
        var submit = document.getElementsByClassName("submit")

        if (drawn==0){
            alert('There is nothing drawn on the canvas');
            return;
            }
        else {
            for (i=0; i<loader.length; i++){
            submit[i].style.display = "none";
            loader[i].style.display = "inline-block";
            }
            $.ajax({
                url: "/switchboard/",
                method: "POST",
                        data: {
                            NNMessage:img,
                            csrfmiddlewaretoken:$('input[name=csrfmiddlewaretoken]').val()
                        },
                        success: function(data){
                            for (i=0; i<loader.length; i++){
                            submit[i].style.display = "inline-block";
                            loader[i].style.display = "none";
                            }
                            alert(data);
                            console.log(data);
                        }
                    });
                }
            });
        });
    </script>
</html>
```