Matthew Tucker – Documentation

**Task: Build a Django REST Framework API that returns JSON and a C# app to access the data. The database is built on specifications provided by the botanical gardens.**

**Issues:**

1. Learn Django and Django REST Framework:
   1. At the beginning of this project I had no background using Django, or the REST framework. I also had no experience using APIs. Django was very difficult to get my head around to begin with as it uses a very structure hierarchy and passes a lot of data between files. The problem was figuring out what bits went where. For example, the models.py file is responsible for defining the classes we are using in our database, the serializers.py file was responsible for converting data to and from JSON, and the views.py file was used to send a receive database requests based on url queries.
2. Learn how to send web requests and parse returning JSON using C#
   1. Sending web requests in C# was a relatively quick learn. We started by sending requests to DigitalNZ (A website that exposes an API and provides extensive documentation on how to access said information). Once we had a good understanding of this and a json string to work with we started on the parsing process.
   2. To parse the JSON string we first needed a C# class that could hold the parsed information. This class had variables for each bit of the JSON string we were grabbing data from. Our JSONParser was then able to deserialize a JSON string into a list of this type of class. Once in this list we could access the data as we like.
3. Make a user interface that allows the user to search for a particular item in the gardens, or search for all items in a collection.
   1. We required a way to display the information in an ordered and user friendly way. We settled on using a dataGridView to display each item returned by the search. An issue we encountered here was that an item search vs a collection search would return different bits of information, meaning we had to change the number of columns and their headings depending upon the search. To further add to the problem, to display an image we needed an image cell which had to be added manually. To fix these problems required a lot of googling and trial and error but eventually we found a set up that worked.
   2. To keep the interface simple we decided to have only two search fields. One the user manually entered a search term, and any results that corresponded to that were displayed in the dataGridView. The other we dynamically populated a drop-down list with each collection in the database. When a collection was chosen a collection description was automatically displayed so the user could get an idea of what they were searching. When the user hit the search button, the dataGridView was populated with all the information about the items in that collection.s