CA377 Programming Fundamentals (Project)

Name	Mihail Gaidau
Student Number	15490698
Programme	EC3
Module Code	CA377
Assignment Title	Programming Fundamentals (Project)
Submission date	12/12/2018
Module coordinator	Suzanne Little & Marija Bezbradica

Name: Mihail Gaidau   Date: 12/12/2018

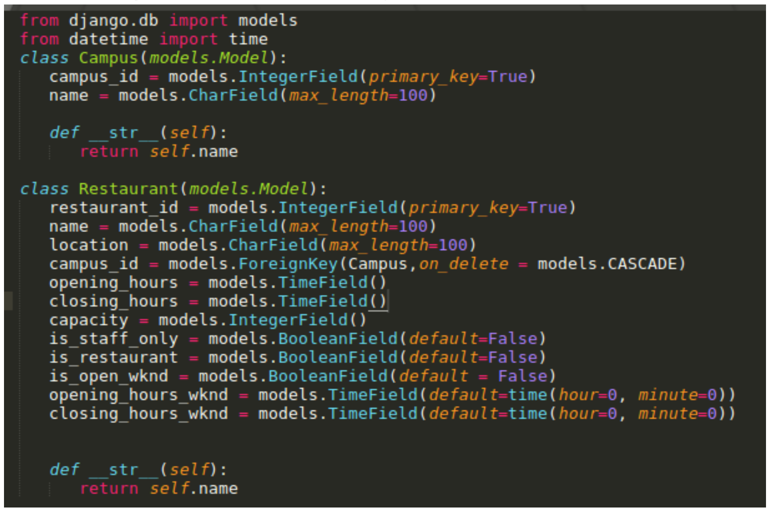
Link : [*http://gaidaum2.pythonanywhere.com/eatatdcu/*](http://gaidaum2.pythonanywhere.com/eatatdcu/)

Introduction

This report will outline what this assignment was about. We are required to develop a website using Python Web Framework Django. The developed web app's objective is to help students, staff and guests of DCU to find suitable food places over all the 4 campuses. The campuses that are included are the main Glasnevin campus, St. Patricks campus, DCU alpha and All hallows Campus. The web app shows the different names of the restaurants and cafes corresponding to the campus you have searched for. It also displays the opening and closing times and whether is open on weekends. A virtual Json web service updates a special for today for all restaurants. For all restaurants there is a link "Special" beside the opening times. Once it is clicked it shows the special today meal for that restaurant.

System Architecture

Django uses multiple layers such as the model, view, URL and template layers. The Django views are responsible for processing any requests and responses. Views can be as simple as returning a string of text. They can also return more complex items such as a template or querying databases and processing credit cards. After the request has been processed a web response is then sent back to the user.
For this project we have a database of campuses and the correlating restaurants and cafes for each campus. This is how we get the information for the campus query. A campus and restaurant classes are made in the models where the database is read in and processed. Each line processes the designated line of the database.



```python
from django.db import models
from datetime import time
class Campus(models.Model):
    campus_id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Restaurant(models.Model):
    restaurant_id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=100)
    location = models.CharField(max_length=100)
    campus_id = models.ForeignKey(Campus,on_delete = models.CASCADE)
    opening_hours = models.TimeField()
    closing_hours = models.TimeField()
    capacity = models.IntegerField()
    is_staff_only = models.BooleanField(default=False)
    is_restaurant = models.BooleanField(default=False)
    is_open_wknd = models.BooleanField(default = False)
    opening_hours_wknd = models.TimeField(default=time(hour=0, minute=0))
    closing_hours_wknd = models.TimeField(default=time(hour=0, minute=0))


    def __str__(self):
        return self.name
```

In order to get the information for special meals an external JSON file read in and the information is parsed.

The templates have to registered in the URL.py page.

```
app_name = 'eatatdcu'
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^restaurants$', views.restaurants, name='restaurants'),
    url(r'^restaurants/specials/(?P<restaurant>[a-z0-9 ]+)$', views.specials, name='specials'),
    url(r'^contact$', views.contact, name='contact'),
    url(r'^explore$', views.explore, name='explore'),


    ]
```

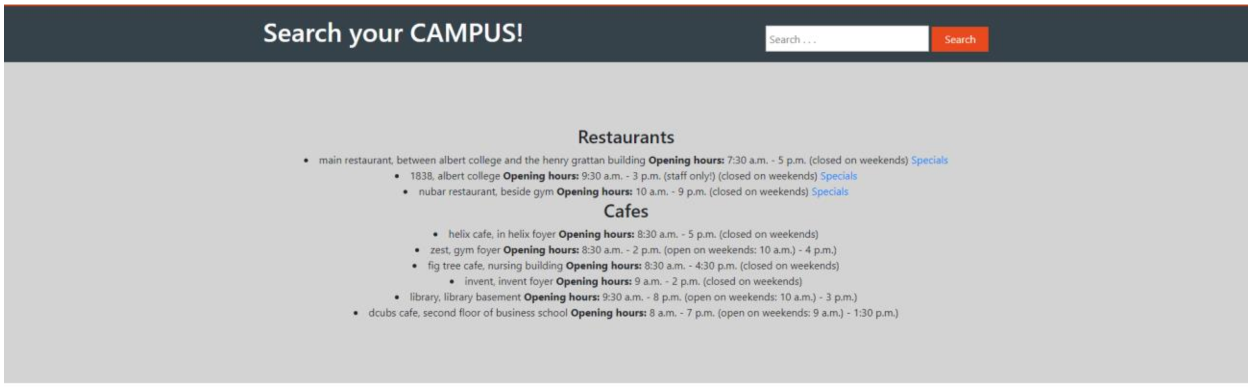
As the templates are registered in the URL page Django then knows what to do since in the views page the function is called to request the "Contact" html. Which then is rendered in the Web Browser.

```
def specials(request,restaurant):
    template = loader.get_template('eatatdcu/specials.html')
    webservice_url = 'http://jfoster.pythonanywhere.com/specials/'+restaurant
    real_time_info = requests.get(webservice_url).json()
    if 'error_msg' in real_time_info:
        return HttpResponse(template.render({'error':real_time_info['error_msg']},request))
    else:
        return HttpResponse(template.render(real_time_info,request))
def contact(request):
    return render(request, 'eatatdcu/contact.html',{})

def explore(request):
    return render(request, 'eatatdcu/explore.html',{})
```

UI Description

I tried to make the UI look a little minimalistic without much things popping onto the screen. As my web development was quite rusty I turned to online tutorials to make it look worthy (All the tutorials I used are in the references section below).

When you first load the website, you are greeted with a Homepage and the showcase image of the DCU canteen with an engaging caption. The Homepage is considered to be the heart of the website. If you scroll down there is a little "about" section which describes the functionality of the website. In the middle of the page there is a search bar where you can search for your campus to find the restaurants/café's in your campus.
For example, this what the result for "Glasnevin" Search looks like:



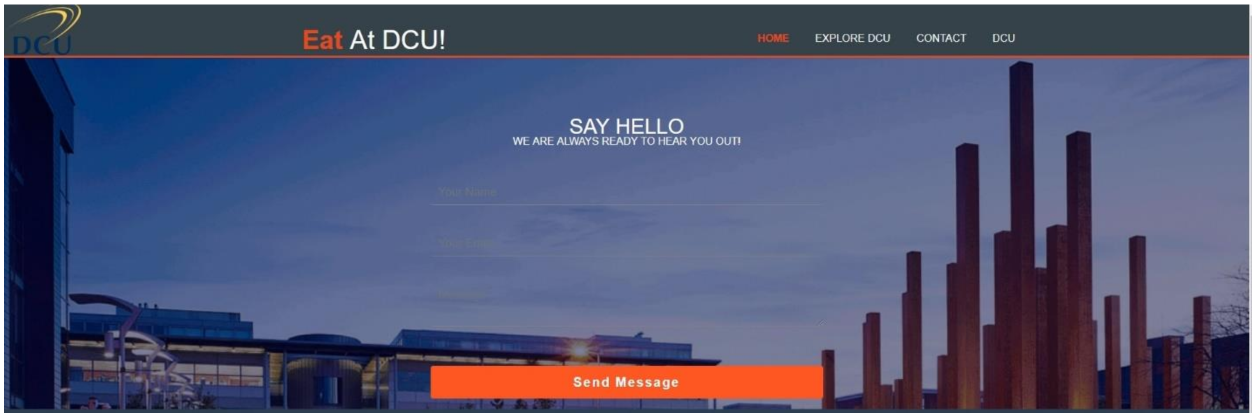
Restaurants

- main restaurant, between albert college and the henry grattan building **Opening hours:** 7:30 a.m. - 5 p.m. (closed on weekends) Specials
- 1838, albert college **Opening hours:** 9:30 a.m. - 3 p.m. (staff only!) (closed on weekends) Specials
- nubar restaurant, beside gym **Opening hours:** 10 a.m. - 9 p.m. (closed on weekends) Specials

Cafes

- helix cafe, in helix foyer **Opening hours:** 8:30 a.m. - 5 p.m. (closed on weekends)
- zest, gym foyer **Opening hours:** 8:30 a.m. - 2 p.m. (open on weekends: 10 a.m.) - 4 p.m.)
- fig tree cafe, nursing building **Opening hours:** 8:30 a.m. - 4:30 p.m. (closed on weekends)
- invent, invent foyer **Opening hours:** 9 a.m. - 2 p.m. (closed on weekends)
- library, library basement **Opening hours:** 9:30 a.m. - 8 p.m. (open on weekends: 10 a.m.) - 3 p.m.)
- dcubs cafe, second floor of business school **Opening hours:** 8 a.m. - 7 p.m. (open on weekends: 9 a.m.) - 1:30 p.m.)

If you are experiencing issues, there is a contact page where users are able to leave feedback or reach out to us with any queries. The contact page can be found in the navbar and it will bring you to a contact page that looks something like this:



The rest of the UI comprises of a link to the main DCU website which is labelled on the navbar as “DCU” and the “Explore DCU” link. The Explore DCU page takes you to a little slideshow of best spots of DCU Glasnevin campus. Overall, I believe that my design suits the DCU colour scheme and it also utilises the DCU logo in a minimal manner.

Using Source Code Repository

For this we used the source code repository GitLab. We used it to get files from the lecturer. We were required to fork the lecturer’s repository and then clone our own repository to our local machine. We then added an upstream repository which was the lecturer’s repository. During the task we are required to commit and push all the files back to our repository on GitLab this lets the lecturers see the commit history. As we complete the task we do a final commit and push. At the start of every new task we had to fetch and merge “upstream” which is the lecturer’s repository. This basically means if there are new files in the repository we forked from we update our own repository. Having the new files in our repository if there are any conflicts we fix them and merge. We used “git diff upstream/master” to identify any conflicts.

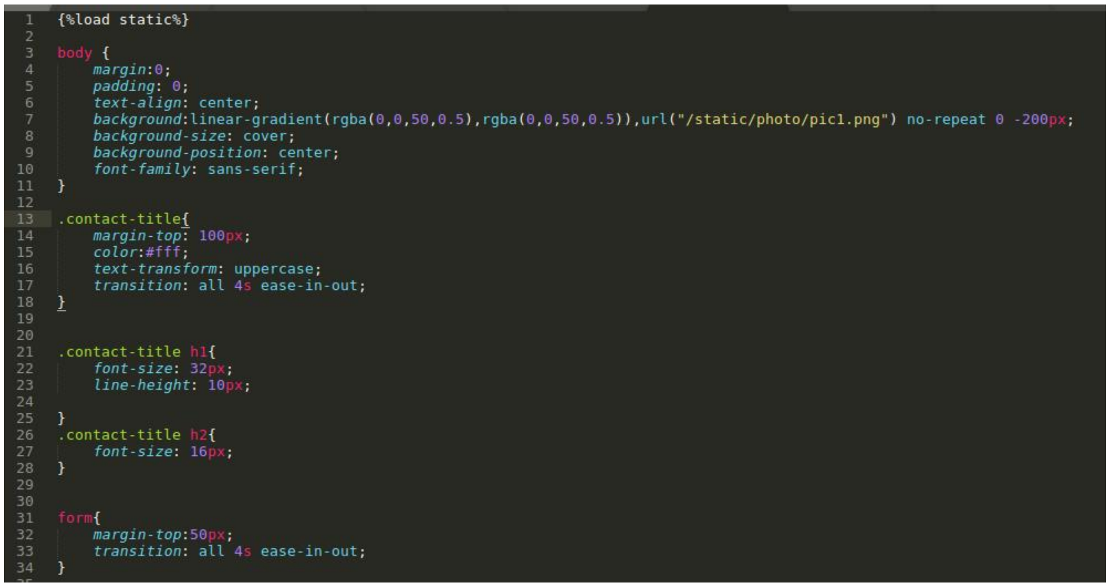
Although this is very handy and makes it easier to collaborate and allows the lecturers to mark our project faster. I did have some odd problems with GitLab. The demonstrators and lecturers were not able to resolve the issue. So, I had to resort to restarting my project 3 times during A2. Thankfully it was at the start of the project. One of the errors was “unable to push before pull”. Although this doesn’t seem bad at first but unfortunately, I have been given a chain errors when trying to resolve the issues.

However, once the issues were resolved working with GitLab was an enjoyable experience. It allows for the developers to look at the code and work further on it. For myself I could pull it from my home pc and work on it from home and continue to work on it where I left it in the labs.
One of the ways of how I could have used GitLab more effectively was to utilize branching. During the project I was pushing straight to master. In order to not hinder my progress, I should have pushed to branch and then if everything seems okay id merge with master. Thankfully there wasn’t any issues with just pushing to master, but from no one I’ll be utilizing branching to mitigate any future problems.

Additional Functionality

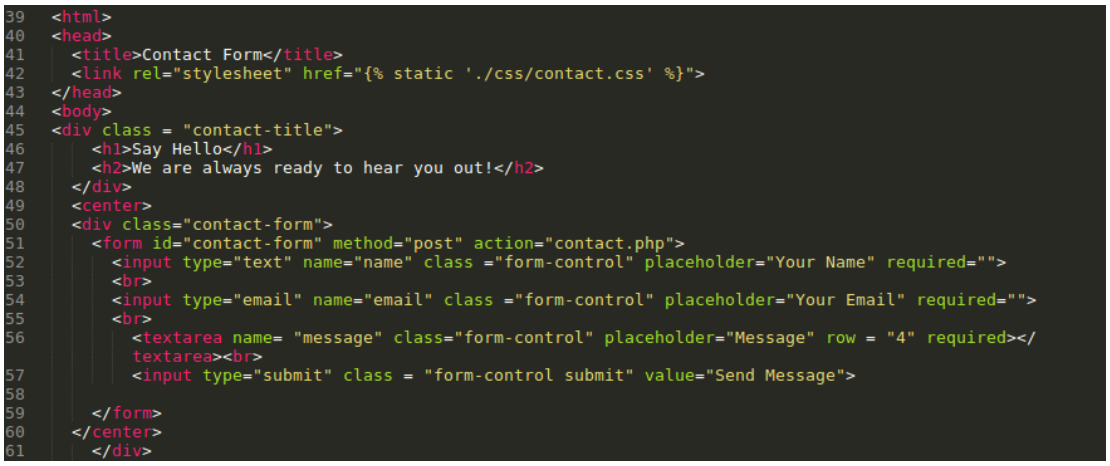
As for the extra functionality since I am aware of bootstrap and have used bootstrap and other tutorials to aid my development of the site I wanted to make it responsive to the screen size of what you're viewing the site with. Since nowadays the mobile usage is growing steadily and mobile friendly sites are almost compulsory. Bootstrap utilizes the 12-column grid system which allows the site to be made responsive to screen sizes easier. The site is responsive to resizing screen on the computer as well. Also noted that there is vast amount of resolution of screens in the industry. Therefore, the site has to be responsive to most if not all screen sizes.

I've also tried to implement a working contact page with PHP but unfortunately as far as I know it doesn't work and needs a server to be functional. I have looked at many tutorials for this to try and make it work. However, in my opinion it fits the website very well as many new users may have issues and would like to reach the creator. Also, in case of bug finding a contact page would serve as a real-life testing. Where if a user found a bug that previously wasn't found they would contact the developers and notify them.
Included below are the screenshots of the Contact HTML and CSS code.



```
{%load static%}

body {
    margin:0;
    padding: 0;
    text-align: center;
    background:linear-gradient(rgba(0,0,50,0.5),rgba(0,0,50,0.5)),url("/static/photo/pic1.png") no-repeat 0 -200px;
    background-size: cover;
    background-position: center;
    font-family: sans-serif;
}

.contact-title{
    margin-top: 100px;
    color:#fff;
    text-transform: uppercase;
    transition: all 4s ease-in-out;
}


.contact-title h1{
    font-size: 32px;
    line-height: 10px;

}
.contact-title h2{
    font-size: 16px;
}


form{
    margin-top:50px;
    transition: all 4s ease-in-out;
}
```

The CSS page is called externally in the HTML page on line 42.



```
<html>
<head>
  <title>Contact Form</title>
  <link rel="stylesheet" href="{% static './css/contact.css' %}">
</head>
<body>
<div class = "contact-title">
    <h1>Say Hello</h1>
    <h2>We are always ready to hear you out!</h2>
  </div>
  <center>
  <div class="contact-form">
    <form id="contact-form" method="post" action="contact.php">
      <input type="text" name="name" class ="form-control" placeholder="Your Name" required="">
      <br>
      <input type="email" name="email" class ="form-control" placeholder="Your Email" required="">
      <br>
        <textarea name= "message" class="form-control" placeholder="Message" row = "4" required></textarea><br>
        <input type="submit" class = "form-control submit" value="Send Message">

    </form>
  </center>
    </div>
```

Conclusion

During the CA377 project I learned to enjoy using Django and GitLab. Previously I had a lot of frustration with them. During the process of working on the project it gotten easier. I learned how to implement static files into Django HTML. Which is a little different than usual. One of the new experiences during this project was working with JSON which is a data interchange format. My favourite part during this project was putting my site online. For this we used Pythonanywhere.com. This lets us upload out Python/Django code online for other people to view. Overall the development was a pleasant experience and I enjoyed learning new things.

References:

Bootstrap 4: https://getbootstrap.com/
Theme: https://codepen.io/adamabundis/pen/BWyEEZ

Explore DCU slideshow: I cannot find the original source for this. These are some of the references that helped me: https://www.youtube.com/watch?v=KkzVFB3Ba_o
https://www.youtube.com/watch?v=7ZO2RTMNSAY
https://www.youtube.com/watch?v=nJWq74MHplc
Contact page: https://codepen.io/stephanrusu/pen/QwKLJX
https://www.youtube.com/watch?v=Iv93yjdvkWI