Project Report – Django Configuration – Front End

**Introduction**

This deliverable is a front-end implementation of my previously created HTML files in the Django web framework. It required installing and configuring Django, importing my HTML files into Django templates, creating an app in Django to contain the necessary views, and finally creating those views.

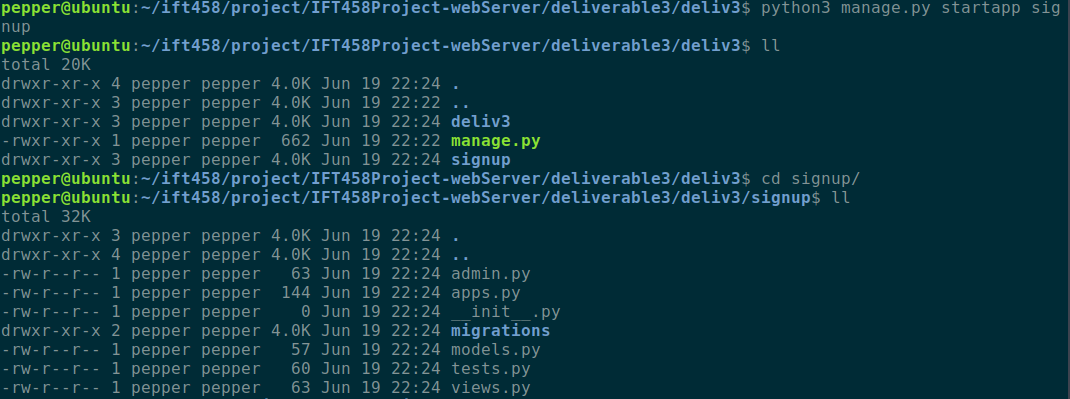
The implementation is bare bones, but it does fully implement the front-end designed in Deliverable 1, including all links and image references. I used git as version control and have the final deliverable files on my [GitHub repository](https://github.com/Pepper37/IFT458Project-webServer) for this class project. Below I will detail my work in creating and configuring the Django project.

**Work Done**

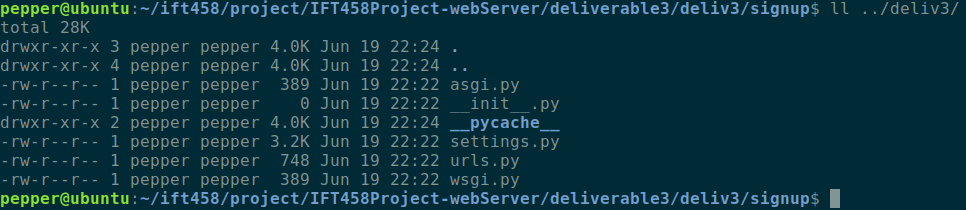
***Installing and Configuring Django***

I began by installing Python3-pip onto my Ubuntu machine, as pip is necessary to install Django. The version of pip installed was the one from the apt package manager. I then used pip to install Django following the official [installation guide](https://docs.djangoproject.com/en/3.2/intro/install/). I primarily used Derek Banas’ [video tutorial](https://youtu.be/Ky59C5geOtg) of setting up a basic Django app to guide me in creating the signup app that contains the views for this Django project.

The project is named “deliv3”. I had forgotten that this is in fact deliverable 4, as I did not have the git deliverable saved in this GitHub repository because it didn’t have to do with the web server project, apologies for the confusion. I also created an app within the Django project, called “signup”. This app will contain the views for the two webpages that I will be importing. After creating the project and the signup app, you can see my directory structure below:



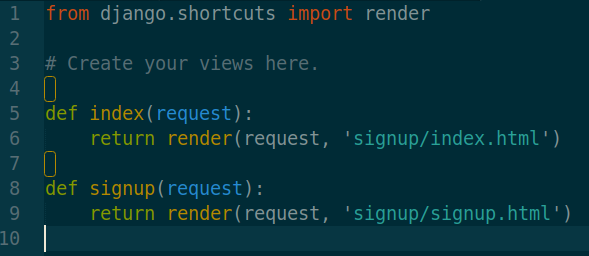
Project root directory and /signup/ app directory



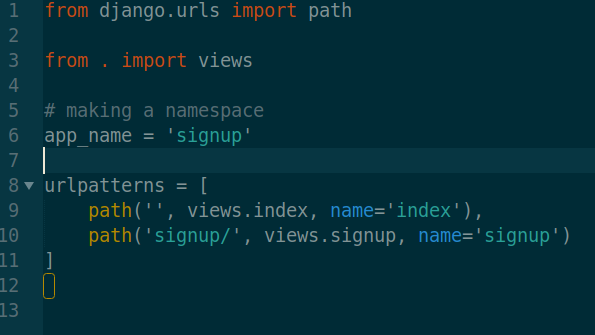
/deliv3/ directory containing settings.py and urls.py

***Configuring settings.py, views.py, and urls.py***

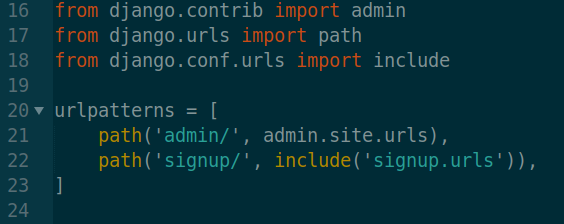
Next, I defined the functions needed within the views.py file under the signup app’s directory and configured the URL paths in the urls.py files within the project directory and the app directory. The contents of those files are below:



signup.views



signup.urls



deliv3.urls

The functions within signup.views load the HTML templates for the respective web pages, and the urls.py files handle the pathing. Of note is that in signup.urls I define a namespace for this app, and name that namespace “signup”. This is important as the namespace needs to be referenced when attempting to use these URLs within the templates.

Finally, I configured the settings.py file. I added the app named “signup.apps.SignupConfig” under INSTALLED\_APPS. I also indicated that the folders for templates and static are, respectively, “/deliv3/templates/” and “/deliv3/signup/static/”. I also created those directories and copied into them the necessary HTML, CSS, and image files. Of note concerning the settings.py file is that I set the timezone for the server to “America/Los\_Angeles”.

Another important thing to make note of is that I had accidentally uploaded the settings.py file to GitHub without obfuscating the SECRET\_KEY variable. As such, GitHub warned me that several scrapers on the site have likely recorded the secret key to this Django project, making it unsecure in production. If this were not a school project, I would take the steps to remedy this. However, considering that I do not intend to ever get this server up and running in production, I don’t consider this a problem and I decided to just let that secret be out there. It is however good to know to obfuscate this key if and when I work with Django in a real-world setting.

***Modifying HTML files into Django Templates***

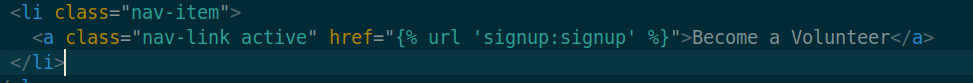
The final piece of work was to modify the index.html and signup.html files to run via Django. This was relatively straightforward. I first included `{% load static %}` at the beginning of each file. Then, any images are referenced using Django Template Language tags as follows:



I also use similar syntax to reference the static CSS files:



Any references (href) to webpage urls between the index.html page and the signup.html page I formatted using DTL tags as follows:



Note the requirement of including the namespace “signup” before the URL name.

**User Manual**

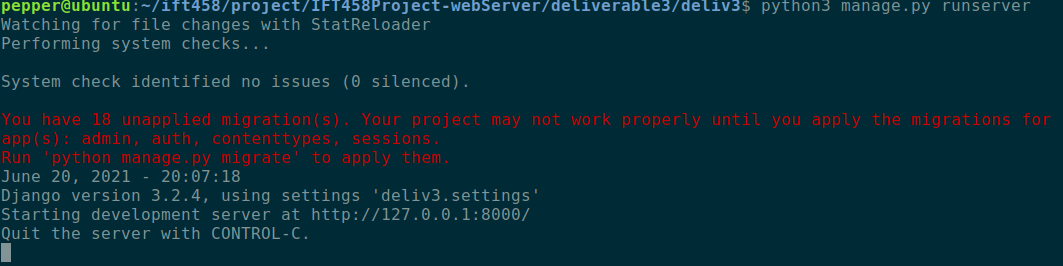
You need to have python3 and Django version 3 installed in order to run this project.

*Step 1:*

Unzip the project files to a location of your choosing

*Step 2:*

In the command line, navigate to the root Django project directory “/deliv3/”. This is the directory that contains “manage.py”. Run the command `python manage.py runserver` (or use python3, if on Linux or MacOS). The command line should give output similar to:



*Step 3:*

Open a web browser and navigate to `localhost:8000/signup`. This should load the index page of the IMPACT volunteer website. Clicking on “become a volunteer” in the header, or clicking on “Volunteer” in the footer, will bring you to the registration page that includes a form to fill out. Clicking on the name IMPACT in the header will take you back to the index.

The form is not operable currently as this site is not yet connected to a database. No other hyperlinks on the site will take you anywhere.

**Conclusion**

After completing this deliverable, I have successfully imported my front-end design into the Django framework. I have configured Django to open the HTML files when the correct links on the front-end are clicked and have configured the URLs to handle that pathing. I have also modified those HTML files to follow Django Template Language syntax, in order for them to be loaded properly by Django. Running the Django server successfully allows a user to open the views specified in the project on the development server by opening a page on the local host.

I learned a great deal about how a web server is implemented and what is necessary to connect to the front-end design in a much more substantial way than I had done before. I also learned what is meant by the “inversion of control” of using frameworks as opposed to normal programming.

One major challenge I faced was in learning how the urls.py and views.py files interact with one another and with the templates. I spent many hours struggling with Django just trying to render my HTML files, and after stepping away I realized I had misunderstood how I was meant to approach the framework. I had URLs pointing to the wrong places and had incorrect DTL syntax in my templates. I also had separate apps for index and signup, believing I needed each page of the site to have its own app. This added unnecessary complexity that frustrated me and kept me from making progress.

I deleted my work, started fresh, and the second time through, with some patience and attention to detail, I was able to understand how to use the framework much better than before. Through this process I grew to appreciate the power of a web framework like Django.

My implementation is very simple and straightforward. I can imagine many improvements that I can make to the site by extending the other pages and experimenting with the interactivity of the template language. As it stands, however, I am proud of my first Django app, and am excited to learn how to attach the database for real interactivity and data manipulation.