**Django Implementation for UI Framework**

**Team GetFramed**

**4/29/20**

1. **Overview**

When designing a front-end framework, there are many options nowadays. Vue, Angular, React, Django, Flask, and more. The initial goal for this project was to integrate both React and Django frameworks to create something both functional and user friendly. However, due to time constraints and the change in environment Django ended up being the only framework used in order to get a simple UI up and running. In respect to the scope and needs of the project, this UI does not fulfill the necessary requirements, but I believe that it serves as a good building block for future react integration.

1. **Getting Started**

There are only three dependencies that are needed for this Django server to be booted up. Just like the PySAL visualization portion, I would recommend using anaconda to download since all the necessary libraries will be included as well. Though Pip Install should work fine.

conda install -c anaconda Django

conda install -c conda-forge bootstrap

conda install -c conda-forge django-crispy-forms

Next, I will show how to setup an environment in anaconda. This step is not necessary, but I believe that is a good precaution to take so that there will not be any conflicts in your dependencies or libraries. To do so, enter this line into your command terminal.

conda create --name myenv

Then type “conda activate” to turn on the environment.

“myenv” is where you will enter the name that you want your environment to be. Starting a Django project will be next.

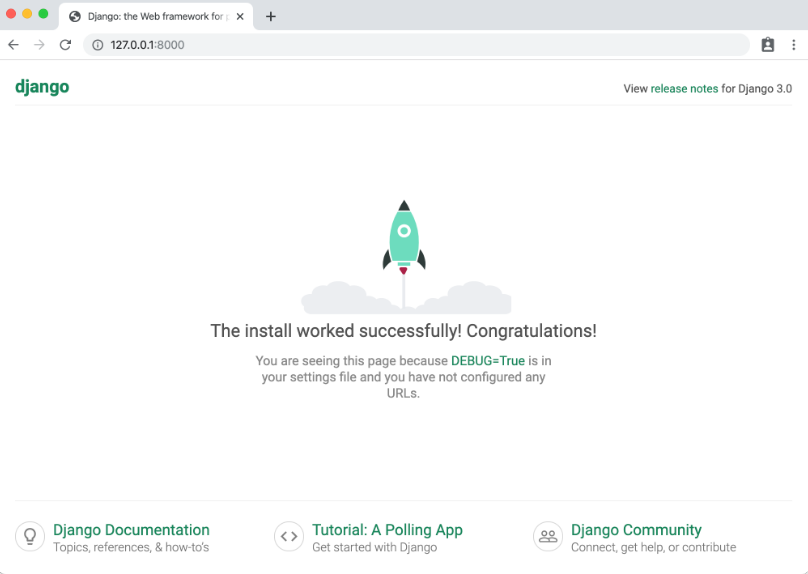
(djangoProject) $ django-admin startproject djangoProject

In the parentheses will be the name of your environment. It automatically appears once you have activated your environment. The “djangoProject” at the end of the line will be where you title your project. In creating this server, I ended up naming multiple aspects “djangoProject” and that resulted in some later aspects becoming more confusing than they should have been. So, I recommend that you have unique names for each part of your project.

Now that you have your project, you can run the server.

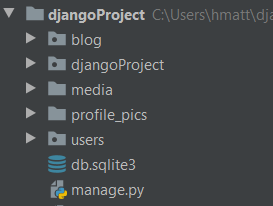
(djangoProject) python manage.py runserver

If everything has ran correctly up to this point than this should be the screen you see when you navigate to local 8000 in your web browser.



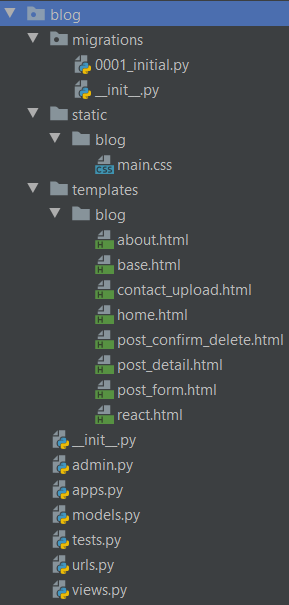
1. **Creating the UI**

When you open the project in wherever you plan on having your code, your directory should look something like this.



Since it is your first time opening the project, the only package should be “djangoProject” or the name of what you decided to name your project. To make things easier, go ahead and add the two other packages to the project, “blog” and “users”. You an do so by right clicking on the root directory, moving to “new” and choosing “package. If you want you can add the directory folders during this process too. Those directories are “media” and “profile\_pics”.

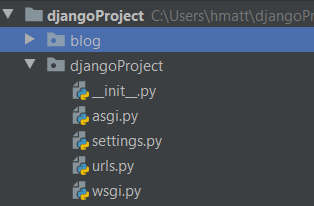
* 1. **Blog**



Your blog package should be empty, except for the .py files that are at the bottom. Everything should be added to the blog package in the same fashion that you added to the root directory. Once those are created, all the files to fill them up can be found in our git repository. This will make it much faster to get the UI running on your machine compared to coding each piece yourself. You will also need to replace the default “models”, “tests”, and “urls” page in the blog package with those that are found in the git repository.

* 1. **djangoProject**

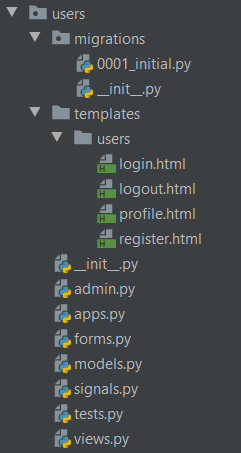
Now take a look inside your main project package.



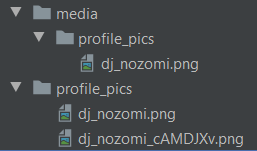
No new files or directories will need to be added here, rather just replace the “settings.py” and “urls.py” files with those that are in our repository. Minor changes were made to these files, but I think it is still easier to simply replace with the ones that already have those changes implemented.

* 1. **Users**

The final package that will need to be filled is the users package. It will look like this in the end.



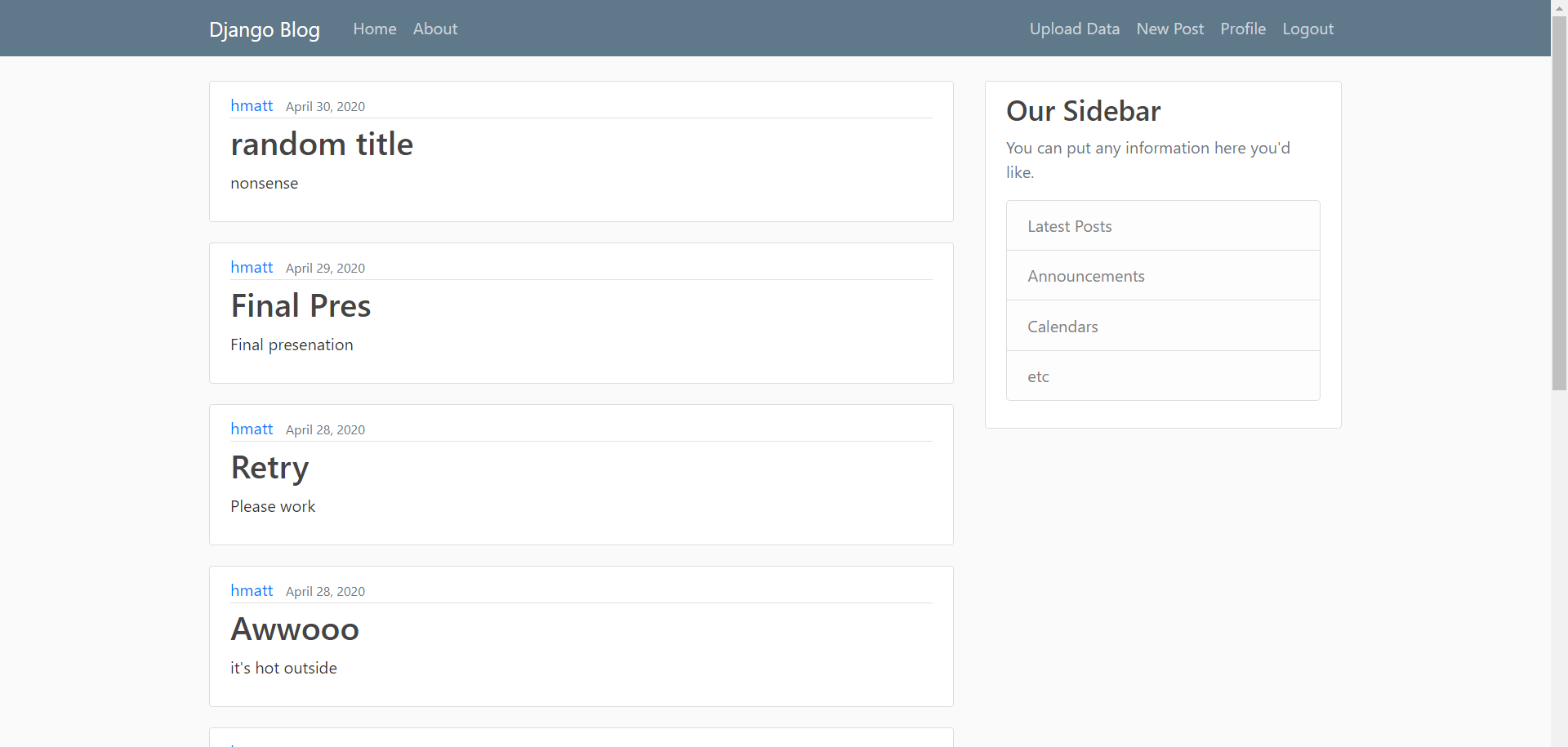
As the prior packages were, all the files to fill up the package can be pulled from the repository. The default .py files that need to be changed for the user package are “signals”, “tests”, and “views”. Those last directories that you have, media and profile\_pics, should still be empty at this point.



You can go ahead and place any picture of your choosing into these directories. They will serve as the default profile picture.

1. **Final Product**

Now you can run through and check for any errors throughout the code, probably the result of a missing dependency here or there. If everything looks good though then you can now run your server again and hopefully be greeted to a main page that looks like this.



Each link at the top will lead to its respective page and you should have full ability to register new users and login with existing ones, as well as to make new posts.

As was stated at the beginning, this is simply a blogging site. However, it is in a consistent state where integration with react should be possible in order to build upon this into something that is useful for the geovisualization project. Along with that, integration with the back-end programming should also be feasible so that users will be able to upload data and interact with it to some regard.