PTX2 Internals

The technical specs of our project are not very complicated, but setup involved was very specific, and downloading components needed was quite a long-winded process. First of all, because our application is strictly web-based, there was not a specific OS that we needed to visit the application upon. Our project can be visited using any web browser. However, development was a different story. For the operating system, we strongly recommend developing on either on a Unix or Linux environment in order to fully support Django/Heroku development. At the beginning stages of our project's development, Alan started developing on Windows, but because many of the tools that we use are not supported for Windows, we recommended for a future developer to develop on Windows.

We used a web framework called Django, which is written in python, and thus needs python to be installed. You can download python here: https://www.python.org/. Django can be installed here: https://www.djangoproject.com/, and the version that we used was 1.6.4, the most recent, stable version of the framework at the time of development.

Our application was entirely contained within the topmost project directory, labelled "PTX2". Within the directory were three sub-directories: django-cas, which was used to require CAS login to use the site, ptx2, which was the Django site, containing meta-data about the project, and ptx2app, which contained our actual web application and the business logic involved with it. In ptx2, the two main files of interest were settings.py, wherein we configured the backend for Django, and urls.py, where we created the associations between urls and specific views. In ptx2app, we had views.py, which were all the backend functions that included queries, loading data into templates, and database accesses/changes for buying and selling, models.py, which were all the data structures needed for ptx2, and forms.py, which were forms that helped the user modify given models. There were also more sub-directories here. "scraper" contains the entirety of the scraper code, which consists of bbscrape.py, which turns Blackboard pages into "soup" for BeautifulSoup, clean.py, which cleans out duplicate courses, data.json and finalcopy.txt, which hold the results of scraping, pagewriter.py, which turns the registrar list into "soup", and scrape.py, which pulls everything together, actually carries out the scraping, and returns the data in the form of the aformentioned data.json and finalcopy.txt. There are two more directories in the folder: templates, which holds the HTML templates that comprised the front end of our page, and static, which holds the javascript, images, and CSS for the site.

We used Heroku to host our app on the web-server, which can be downloaded here: https://www.heroku.com/. Furthermore, on their website, they have a download called Heroku toolbelt, which can be found here: https://devcenter.heroku.com/articles/quickstart. Heroku toolbelt contains Heroku, a command-line tool for creating and managing your apps, foreman, an easy way to launch your app locally, and git, Heroku's version control system of choice, and we highly recommended using the toolbelt, as it simplifies the installation process greatly.

There are a few key settings in our Django app required to make it compatible with Heroku. First of all, Heroku uses PostgreSQL, different to the SQLite default of a django application. PostgreSQL can be installed here: http://www.postgresql.org/. The main compatibility actions that we had to add/change were first, make a Procfile, which is a file that declares what command to run to start the actual app on Heroku, and second, change some of settings.py to make Django compatible with PostgreSQL. There are some other minor changes, but nothing too major, and all of them are detailed here: https://devcenter.heroku.com/articles/getting-started-with-django.

For version control, we used Github. First, those of us who didn't have Github accounts before made accounts on https://github.com/. Then, Michael, the team leader, created a new repository for our project called PTX2 at https://github.com/limichaelc/PTX2, and added the other users as contributors. Each of us then initialized a git repository in our local folders where we would store our copies of this project, and made the Github PTX2 repository our remote. Setting it up this way allowed all of us to push and pull from Github in order to make changes and see changes in the project that other team members had made.

For the front end, we used the default Bootstrap CSS package, which can be downloaded at http://getbootstrap.com, with some additional CSS tweaks. These files are all located in static directory in ptx2app, with the customized CSS accessible in style.css. The additional javascript that we used can be found in scripts.js, in the same directory.

Summary
Operating system: web-app, so not very specific, but developed on a combination of OSX, Linux, and Windows
Languages: Python for Django; HTML5, CSS3, and Django's templating language for front end
Framework: Django, which can be easily downloaded at https://www.djangoproject.com
Host: Heroku
Database: PostgreSQL, which can be gotten from their website (http://www.postgresql.org), but can also be more easily installed using http://postgresapp.com
Version control: Github
Front-end: HTML5, CSS3, Bootstrap, which can be accessed at getbootstrap.com