Title: Techno Cru | Author: Ian Kirkpatrick

Purpose:

This website is designed to make life easier for Cru team members who are working on technology for the Thursday night meetings. The focus of this project is organization for Cru team members.

After speaking with staff in Cru about different equipment needed for sound, media, band and other such things, I have decided that a centralized area for organizing wishlists for different departments of Cru would be very beneficial.

While thinking about ways I could do this, I also realized that there are other needs that could be solved by a centralized web server. There is already a web app that I created that deals with inventory but I thought of other things as well. This website will help to keep the tech team in cru organized and provide easy ways to do some different tasks.

The goal of the wishlist app is to provide a centralized space to request equipment for the Cru Team. For example, if two microphones are needed, a team member can add that request to this web app.

Architecture:

This website implements the Django web framework, written in Python. Specifically, this website implements Django 1.11, using Python 2.7.

Django is a Model Template View (MTV). This is similar to the Model View Controller architecture except for a few things. The difference is that the controller is encapsulated in the entire framework and is not necessarily implemented separately on its own. The view encapsulates the template and the code behind the template. The models are basically the same as the MVC in that they are classes that represent the data in a database.

File hierarchy:

The file hierarchy will follow that of the Django hierarchy. Django is an app oriented framework. Each system is divided up into its own app. There are two apps, main and wishlist. The main app will deal with the authentication and users. The wishlist app will deal with the wishlist logic.

Each app has four files. These are models.py, views.py, urls.py and forms.py. The wishlist app also has a package that has urls.py and a views.py file for the api.

The models.py contains the classes that are the models. These provide access to the data in the database.

The views.py file contains the views. These are the classes or functions that run the code behind the web pages.

The urls.py file contains the url conf that maps urls to their respective view in the views.py file.

Data storage:

Django uses a database and an ORM to interface with that database. Wishlist data is stored in that database.

When the items requested are saved, the view calls the ORM and saves the item data. The call looks like ItemRequest.objects.create(name=”…”, description=”…”, <other\_info>).

Front-end:

Javascript/JQuery, HTML and CSS are used for the front-end. This is the barebones technology used with Django. JQuery provides an easy way to make an interactive page, especially with the ajax library.

To call a url using the JQuery ajax library, use the following:

$.ajax({

url: “…”,

data: {…},

success: function(response) {…}

});

Back-end:

Django is the framework used for the backend. This is written in Python. Django is a MTV framework.

To create a view using Django, the following may be used:

class MyWebPage(TemplateView):

def get(request, \*args, \*\*kwargs):

return render(request, “template\_path”, context\_for\_template)

Database server:

This project uses SQLite3 for it’s database. Django uses SQLite3 as its startup database used to bootstrap projects.

Host:

This project is hosted on a Unix environment (Cause Windows is terrible). I specifically use Pythonanywhere to host this project. This is a Unix environment based on the web. Django works best in a Unix environment because of its heavy use of the Unix command line base.

Security:

For security, a user must be logged in to access the website. Django provides a user and group authentication framework. This also encrypts passwords set for each user.

Hosting Software:

Pythonanywhere is used to host this with its own code base and software.

Development model:

The closest concept that I can think of that I used for the development process is the Agile development process. I did this process in an iterative manner. This is what the agile development process includes.

