**IV&V Web Form Documentation**

**Technology used**

* Django
  + Django is a python framework used to develop web pages
  + I followed this tutorial to get a better idea of how to use Django <https://tutorial.djangogirls.org/en/django_start_project/>
  + Once a Django project is created, Django automatically creates some files for you
    - Manage.py is the important one and you need this file to do other things
    - Manage.py is used to create an app, which is where the code is stored
      * Run the command python manage.py startapp <app\_name>
      * This is already done, and all the code is in the ‘webform’ directory
    - It takes three steps get the project running from this code
      * 1. Run python manage.py createsuperuser
        + Then fill out all the fields
      * 2. Run python manage.py migrate and python manage.py makemigrations webform in order to create the database and models
      * 3. Run python manage.py runserver to run the application
* Sql lite database
  + A super light weight database that will definitely need to be replaced should this prototype actually be made into production

**Important Files**

* Within the ‘webform’ directory, there are a couple of important files: admin.py, models.py, forms.py, urls.py, and views.py
* **Models.py**
  + In this file, each table in the database is defined. The table structure is as follows:
    - State table w/ ID, submitter table w/ID, Life cycle w/ID, recommendation cycle w/ID, risk table w/ID, and IVV table w/ID
    - Because the state and submitter tables have a one to one relationship with the IV&V table, the IV&V table has fields for state ID and submitter ID
    - Because Life Cycle, Risk, and Recommendation all have a many to one relationship with the IV&V table, they all fields for the IV&V ID
    - Then, there is an Entire Form table, that simply links the User with all of the other table’s ID.
      * This is so that eventually the functionality of displaying and editing all forms for a specific user can be used
* **Admin.py**
  + This is a very straightforward file, each model must be registered so that it shows up on the admin page.
  + The admin page is used as a GUI for the database and you must be logged in as a superuser (which is created with the createsuperuser command) in order to see it
* **Forms.py**
  + This file contains all the definitions for forms
  + I used Django’s modelForms class, which bases the forms off of the models defined in models.py
  + You simply state which model a form is based off of and then indicate either the fields to be included or the ones to be excluded
* **Views.py**
  + This is where the behavior for each page is called
  + All the functions in this file take in a request parameter and will either POST or GET data depending on the behavior
* **Urls.py**
  + Defines the URL and links it to a behavior stated in views.py
  + Ex. 127.1.1.8000/display links to the display method in views.py and shows the form after it has been submitted

**The Rest of the File Structure**

* **HTML Code**
  + All of the html code is stored within the webform/templates/webform directory
  + These pages get linked to a URL through the urls.py file
  + In these html files there are a few keywords
  + {% load staticfiles %} will load all of the Javascript/CSS/images
  + {% extends webforms/base.html %} will put the base template for the web pages in the html, this goes along with Django’s theme of Don’t Repeat Yourself and allows you to reuse html without having to retype it
  + {% block content %} and {% endblock %}, in the webform/base.html file, the header and nav bar are defined, but the rest of the page is blank and to be filled in depending on what the page should look like. These two keywords just indicate where the rest of the page starts
  + {{variable}} uses a variable passed in by views.py,
    - Views.py passes in context like the following. It passes in the python variable ivv\_results as ‘ivv\_results’, so that in the html if I call {{ivv\_results}} I will be able to access the fields of the python variable
* context = {'ivv\_results':ivv\_results,
* 'lc\_results': lc\_results,
* 'state\_results': state\_results,
* 'submitter\_results':submitter\_results,
* 'rec\_results':rec\_results,
* 'risk\_results':risk\_results
* }
* return render(request, 'webform/display.html', context)
  + - The render function takes in a request parameter, the proper html template, and the context
* **Static Files**
  + Inside the static directory, there are four sub-directories.
  + These are css, fonts, js, and imgs.
  + If you want to place new css, fonts, javascript, or image files all you need to do is insert them into these directories.

**TO DO**

* Ensure that the web site can handle multiple requests
* Create the edit functionality for previous forms
* Find somewhere to host the site
* Create a save button and allow users to come back to the form
* Edit the Excel to make it in the same format SPoTT needs