# Initial Setup

1. Install Python 3.7.3.
   1. <https://www.python.org/downloads/>
2. Install Django.
   1. Tutorial: <https://docs.djangoproject.com/en/2.2/intro/tutorial01/>
   2. Full documentation for the project is available at <https://www.django-rest-framework.org/>.
   3. First install pip
      1. pip is already installed if you are using Python 2 >=2.7.9 or Python 3 >=3.4 downloaded from [python.org](https://www.python.org/)
   4. The following command will install latest Django
      1. pip install Django
3. Since we are using Django Rest Framework, install with one of following ways.
   1. Pip install djangorestframework
   2. Git clone <https://github.com/encode/django-rest-framework>
4. Create a directory for a container.
   1. For this example: C:\ safechain
5. Move to C:\safechain.
   1. For verification of Python and Django version at command line
      1. python –-version
      2. python -m django –version
   2. django-admin startproject mysite
      1. mysite was safesite for this example
   3. Move to new directory safesite to create an app
      1. python manage.py startapp restapp for this example
6. Since we are going to use rest framework, update the setting.py.
   1. Add ‘rest\_framework’ to INSTALLED\_APPS
   2. Add ‘restapp’ to INSTALLED\_APPS
7. Move to where manage.py file is located and start server to see if any issue.
   1. python manage.py runserver
   2. control c to stop server

# Scenario for this project example

1. User database table.
2. Group database table.
3. User can be associated to multiple groups.
4. For new user, there will be 2 groups automatically created: “Officer”, “Admin”.
5. The following operations will be available:

● Add New User

● List Users

● Delete User

● Update User

● List Groups for given user

1. Output will be JSON.
2. [http://localhost:8000/api/<endpoint](http://localhost:8000/api/%3cendpoint)>

# Project Layout

1. URLs for this app.

urlpatterns = [

path('groups/', GroupView.as\_view()),

path('groups/<int:pk>', GroupView.as\_view()),

path('user/', UserView.as\_view()),

path('user/<int:pk>', UserView.as\_view()),

]

‘groups’ urls will do the group maintenance

‘user’ urls will do the user maintenance

GET – retrieves

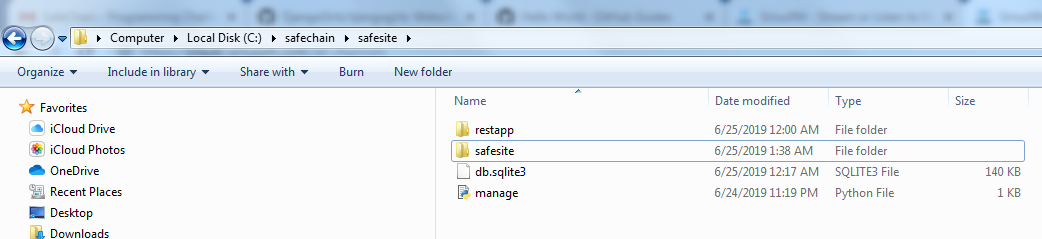
POST – creates

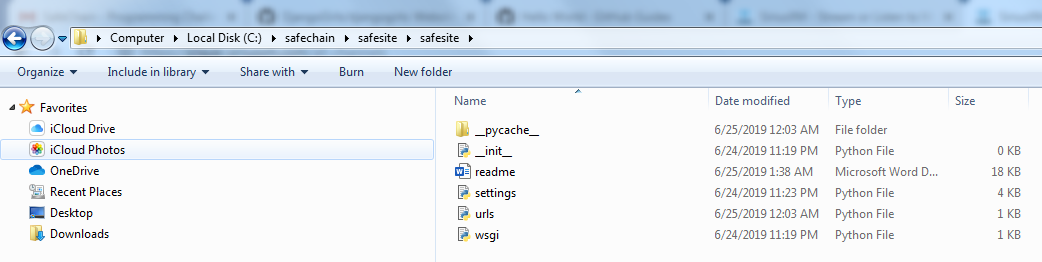
PUT – updates

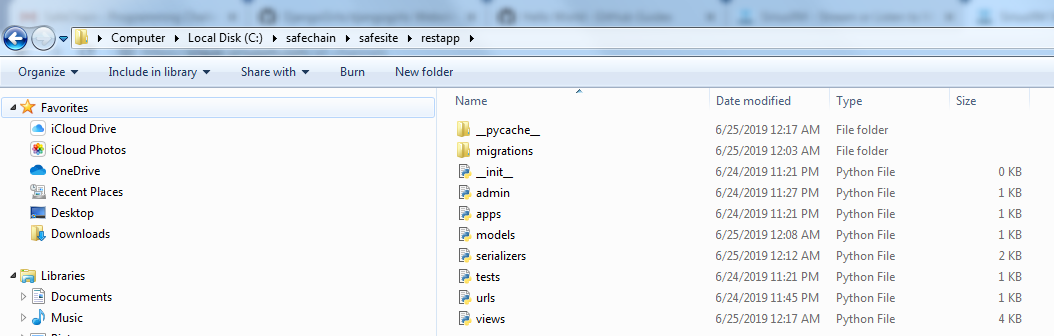
DELETE – deletes

1. Used SQL lite for this example. Most likely use a more robust database for a production app.
2. For this example, ‘safesite’ contains the web service configuration/setting files.
3. For this example, ‘restapp’ contains the app/api configuration/setting files.
4. After creating getting all the files for this project, we will need a run a few commands on command line where manage.py is located.
   1. python manage.py makemigrations
   2. python manage.py migrate
   3. python manage.py runserver
5. The server should be running now so we start using the application.
   1. [http://localhost:8000/api/<endpoint](http://localhost:8000/api/%3cendpoint)>
6. The following is a good resource which documents the different files.
   1. <https://docs.djangoproject.com/en/1.8/intro/tutorial01/>

# Screenshots of Directory Structure







# Some Testing Screeenshots

