# Final Project Phase 1 Step-by-step Guide: setup the Django website and implement file uploading

ECE 157 TAs

Fall 2020

## 1 Django Setup

- Verify python installation with Python 3.6 or later.

```
python --version
```

- Follow tutorial on Django REST framework website:
  `https://www.django-rest-framework.org/tutorial/quickstart/`
- Create your project directory

```
mkdir YOUR_PROJECT_DIRECTORY
cd YOUR_PROJECT_DIRECTORY
```

- OPTIONAL: install virtualenv

```
pip install virtualenv
```

- OPTIONAL: Create a virtual environment to isolate our package dependencies locally

```
python -m venv env
source env/bin/activate
# On Windows use ‘env\Scripts\activate‘
```

  - You can also activate virtual environment by selecting python interpreter on Vscode.
  - Possible error:
    `https://stackoverflow.com/questions/4037939/powershell-says-execution-of-scripts-is-disabled-on-this-system`

- Install Django and Django REST framework into the virtual environment

```
pip install django
pip install djangorestframework
```

- Set up a new project with a single application

```
django-admin startproject YOUR_PROJECT_NAME .
cd YOUR_PROJECT_NAME
django-admin startapp YOUR_APPLICATON_NAME
```

- Sync your database for the first time:
  See what is migration here: `https://docs.djangoproject.com/en/3.1/topics/migrations/`

```
python manage.py migrate
```

- Run the following command and go to `http://127.0.0.1:8000/` on your browser. You should see a congratulation page. You have started a server on your local host.

```
python manage.py runserver
```

- Add `'rest_framework'` and your application `'your_application_name'` to `INSTALLED_APPS`. The settings module will be in `YOUR_PROJECT_NAME/settings.py`

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'your_application_name',
]
```

- Configure Django media setting for storing files. Go to `YOUR_PROJECT_NAME/settings.py` and add the following lines to the end of the python file.
  More info here: `https://docs.djangoproject.com/en/3.1/topics/files/`

```
MEDIA_ROOT  = os.path.join(BASE_DIR, 'media')
MEDIA_URL ='/media/'
```

  - If it complains about `'os'` not defined, add `import os` at the beginning of the file.

## 2 Implement File Uploading

- Create the file model. Open YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/models.py and copy the following code into it:
  See what is a Django model here: https://docs.djangoproject.com/en/3.1/topics/db/models/

```
from django.db import models

# Create your models here.

class FileModel(models.Model):

    file_name = models.CharField(max_length=50)
    file_content = models.FileField(upload_to="upload")
```

- Update database with the newly created FileModel:

```
python .\manage.py makemigrations
python .\manage.py migrate
```

- Create a new python file named YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/serializers.py that we'll use for our data representations. Copy the following code to YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/serializers.py
  See what is a serializer here: https://www.django-rest-framework.org/api-guide/serializers/

```
from .models import FileModel
from rest_framework import serializers

class FileSerializer(serializers.ModelSerializer):
    class Meta:
        model = FileModel
        fields = '__all__'
```

- Create a new folder YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/templates to store HTML templates. Download the HTML templates provided by TAs, and put the two HTML files in this folder:
  YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/templates/base.html and
  YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/templates/index.html
  See how to render customized HTML in Django:
  https://www.django-rest-framework.org/topics/html-and-forms/

- Implement the API endpoint that allows files to be uploaded. Open YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/views.py and copy the following code into it:
  See what is a Django view here: `https://www.django-rest-framework.org/api-guide/views/`

```
from django.shortcuts import render

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.parsers import FormParser, MultiPartParser, JSONParser
from rest_framework.renderers import TemplateHTMLRenderer
from rest_framework import status

from .models import FileModel
from .serializers import FileSerializer

class YourViewName(APIView):

    parser_classes = [JSONParser, FormParser, MultiPartParser]
    renderer_classes = [TemplateHTMLRenderer]
    template_name = 'index.html'

    # See Django REST Request class here:
    # https://www.django-rest-framework.org/api-guide/requests/

    def get(self, request):

        return Response(status=status.HTTP_200_OK)

    def post(self, request):

        # Upload form
        if 'upload' in request.data:
            file_serializer = FileSerializer(data=request.data)

            if file_serializer.is_valid():
                file_serializer.save()
                return Response({'status': 'Upload successful!'},
                                status=status.HTTP_201_CREATED)
            else:
                return Response(status=status.HTTP_400_BAD_REQUEST)
```

- Create a new python file named YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/urls.py. This module maps URL path expressions to Python functions (your views). Copy the following python code into this file:

See how Django URL dispatcher works here: https://docs.djangoproject.com/en/3.1/topics/http/urls/

```
from your_application_name import views
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static

app_name = 'your_application_name'
urlpatterns = [
    path('', views.YourViewName.as_view(), name='your_appication_name'),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

- Configure the project level url.
  Open YOUR_PROJECT_NAME/YOUR_PROJECT_NAME/urls.py and copy the following code:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('your_application_name.urls')),
]
```

# 3 Test Your API

- Start the server and go to local host at http://127.0.0.1:8000/. You should be able to see the file uploading page.

```
python manage.py runserver
```

- Try upload a test file. After click the Upload file buttion, you should see an upload successful response. Go to your project directory you should be able to see a new folder created at YOUR_PROJECT_NAME/media/upload with the uploaded file in it.

# 4 Take a look into Data Base

- Download and install the SQLite Tools following instruction here: https://www.sqlitetutorial.net/download-install-sqlite/

- Add path to the executable `sqlite3.exe` to your system path
- Go to your Django project directory, locate your database file: YOUR_PROJECT_NAME/db.sqlite3.
- Open command prompt, type `sqlite3` and hit enter. This should open the SQLite command shell.
- Enter the following command you should be able to see the table content in your FileModel. As seen, there is one table entry showing your recently uploaded csv file.

```
sqlite> .open db.sqlite3
sqlite> .tables
auth_group                      django_admin_log
auth_group_permissions          django_content_type
auth_permission                 django_migrations
auth_user                       django_session
auth_user_groups                your_application_name_filemodel
auth_user_user_permissions
sqlite> SELECT * FROM  your_application_name_filemodel;
1|test|upload/unknowns.csv
sqlite>
```

# 5 Review

Review all the steps we went through to implement the upload button. Start thinking about how to implement the other functionalities! There will be no step-by-step guide for them. But a hint is that you are most likely only need to modify the logic dealing with the HTTP request in YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/views.py along with its rendred HTML. Of course, you may also need to import your machine learning analytic scripts in phase 2.
For extra credits, you will need to modify the file model and implement two more models in YOUR_PROJECT_NAME/YOUR_APPLICATON_NAME/models.py - one for storing the ML model and scripts, and the other for storing analytic results. For implementing the admin feature, take a look at `https://docs.djangoproject.com/en/3.1/ref/contrib/admin/`