-----------------------------------------------------------------------------------------------

**SETUP DJANGO ENVIRONMENT**

conda create -n my\_django\_environment python=3.9

conda activate my\_django\_environment

conda deactivate

pip3 install django~=3.1

pip list –local

pip show Django or conda list Django

conda info –envs // List of virtual environments.

cd C:\Users\jaski\Downloads\UofA\Courses\MM802\_MULTIMEDIA COMMUNICATIONS\Assignments\Project\HRS

cd SourceCode

django-admin startproject HRS

cd HRS

python manage.py runserver 🡺 python manage.py runserver 8080 for specific server

http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.<https://www.kaggle.com/c/expedia-hotel-recommendations/overview>

---------------------------------------------------------------------------------------------

**File Structure:**

**HRS/**

* manage.py
* locallibrary/
* \_\_init\_\_.py
* settings.py
* urls.py
* wsgi.py
* asgi.py

The *HRS* project sub-folder is the entry point for the website:

* **\_\_init\_\_.py** is an empty file that instructs Python to treat this directory as a Python package.
* **settings.py** contains all the website settings, including registering any applications we create, the location of our static files, database configuration details, etc.
* **urls.py** defines the site URL-to-view mappings. While this could contain *all* the URL mapping code, it is more common to delegate some of the mappings to particular applications, as you'll see later.
* **wsgi.py** is used to help your Django application communicate with the webserver. You can treat this as boilerplate.
* **asgi.py** is a standard for Python asynchronous web apps and servers to communicate with each other. ASGI is the asynchronous successor to WSGI and provides a standard for both asynchronous and synchronous Python apps (whereas WSGI provided a standard for synchronous apps only). It is backward-compatible with WSGI and supports multiple servers and application frameworks.

The **manage.py** script is used to create applications, work with databases, and start the development web server.

------------------------------------------------------------------------------------------------------------------------------------

**python manage.py startapp catalog**

**Updated project directory should now look like this**:

**HRS/**

* manage.py
* HRS/
* catalog/
  + admin.py
  + apps.py
  + models.py
  + tests.py
  + views.py
  + \_\_init\_\_.py
  + migrations/

In addition we now have:

* A *migrations* folder, used to store "migrations" — files that allow you to automatically update your database as you modify your models.
* **\_\_init\_\_.py** — an empty file created here so that Django/Python will recognize the folder as a [Python Package](https://docs.python.org/3/tutorial/modules.html#packages) and allow you to use its objects within other parts of the project.

[**Registering the catalog application**](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/skeleton_website#registering_the_catalog_application)

Open the project settings file, **django\_projects/HRS/HRS/settings.py**, and find the definition for the INSTALLED\_APPS list.  Then add a new line at the end of the list, as shown below:

# Add our new application

'catalog.apps.CatalogConfig', #This object was created for us in /catalog/apps.py

## [Specifying the database](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/skeleton_website#specifying_the_database)

## SQLite database  is default database and is configured in ****settings.py**** but we can change that too.

TIME\_ZONE = 'Europe/London'

## [Hooking up the URL mapper](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/skeleton_website#hooking_up_the_url_mapper)

## Open **HRS/HRS/urls.py. Add below code to end of file**

## # Use include() to add paths from the catalog application

## from django.urls import include

## #Add URL maps to redirect the base URL to our application

## from django.views.generic import RedirectView

## # Use static() to add url mapping to serve static files during development (only)

## from django.conf import settings

## from django.conf.urls.static import static

## urlpatterns = [

## path('admin/', admin.site.urls),

## path('catalog/', include('catalog.urls')),

## path('', RedirectView.as\_view(url='catalog/')),

## ] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

## create a file inside your catalog folder called **urls.py**

from django.urls import path

from . import views

urlpatterns = [

]

## Running database migrations

## ****Open**  /HRS/catalog/migrations/** to automatically migrate the underlying data structure in the database to match the model.

## \HRS>

python manage.py makemigrations

python manage.py migrate

## run these when database is updated

## [Creating a superuser](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Admin_site#creating_a_superuser)

Call the following command, in the same directory as **manage.py**, to create the superuser. You will be prompted to enter a username, email address, and strong password.

python manage.py createsuperuser

Username (leave blank to use 'jaskirat'): admin

Email address: jjaskiratsingh211@gmail.com

Password:admin

Password (again):admin

The password is too similar to the username.

This password is too short. It must contain at least 8 characters.

This password is too common.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

