LEARNING LOG DOCUMENTATION

1. First, we will create a virtual environment where we can install packages and isolate them from rest of the Python packages. It’s necessary to isolate them if we want to deploy the project/application.
2. For creating the virtual environment in the directory, we use the following command in the terminal:

“python -m venv ll\_env”

1. After creating, to activate the virtual environment, we use the following command:

“source ll\_env/bin/activate”

1. After activating, you can use the packages installed in this virtual environment. And also, these packages run only when the virtual environment is active.
2. To deactivate the virtual environment, we use “deactivate” command in the terminal. Or maybe, you can also terminate if you close the terminal window.
3. Now, it’s time to install Django in the virtual environment using the following command:

“pip install django”

1. To create a new project in the virtual environment, we use the following commands:

“django-admin startproject learning\_log .”

PS: DON’T FORGET THE DOT!

1. Then you can check the files it has in it by “ls” command.
2. “manage.py” file takes in commands which will be useful for working with the databases and running servers.
3. The directory contains 4 files which are very important:
   1. “settings.py” controls how Django interacts with the system and manages your project
   2. “urls.py” tells Python which pages to build in response to browser requests.
   3. “wsgi.py” helps Django serve the files it creates. WSGI-Web Server Gateway Interface
   4. “manage.py”
4. Now, we have to create the database for Django to work with. We use the following commands:

“python manage.py migrate”

1. For checking if the project has been set up correctly, we run the following command:

“python manage.py runserver”

<http://127.0.0.1:8000/> can be used to access the local server

1. For starting an app, we use the following command here:

“python manage.py startapp learning\_logs”

The command “startapp <app name>” tells python to create the infrastructure needed to build an app

1. After the app is started, we can check in the folder for the files it created. The most important files here are:
   1. “models.py” to define the data we want to manage in our app
   2. “admin.py”
   3. “views.py”
2. So the model here Topic. To tell Django about modifying the database so it can store information, we enter the following command in the terminal:

“python manage.py makemigrations learning\_logs”

“python manage.py makemigrations <app\_name>”

1. This will create a table for model Topic in the database. Now, we will apply this migration and have Django modify the database using the following command:

“python manage.py migrate”

1. The following steps need to be followed:
   1. Modify “models.py”
   2. Call “makemigrations” on learning\_logs <app\_name>
   3. Tell Django to “migrate” the project
2. Next, we have to set up “Admin Site”. Only site’s administration use Admin site, and not the general users.
3. Setting up a server:
   1. Django allows you to create a “Superuser”, a user who has all privileges available on the site
   2. To create a superuser, enter the following commands:
      1. “python manage.py createsuperuser”
      2. Enter the credentials of your own
      3. The user is created after you enter your email and password
4. \*\* Django doesn’t store the password you enter. It stores a string derived from the password, called Hash. Each time you enter your password, Django matches that entry with the existing Hash and validates your entry.
5. Registering a model with the Admin Site:
   1. Open “admin.py” file
   2. To register Topic with the admin site, we enter the following:
      1. “admin.site.register(Topic)”
      2. This code first imports the model that we want to register, i.e., Topic.
6. To use the superuser account, go to:

<http://localhost:8000/admin/>

1. My credentials:
   1. Username: cs1901
   2. Password: 123456789
2. To activate Django Shell, enter the following in terminal:
   1. “python manage.py shell”
3. In the shell,
   1. Topic.objects.all() 🡪 To get all the instances of the model Topic
   2. The list that is returned is called a query set

DEPLOYING “LEARNING LOG” APPLICATION ON HEROKU!!

Committing the project:

(In the directory of the web application)

* git init
* git add .
* git commit -am “Ready for deployment to heroku.”
* git status