**Django Module 2: my first application**

**Setup Environment.**

* To setup environment I need to download following tools
  + Download or upgrade python to latest version and confirm it using python3 –version from terminal
  + Download pipenv from terminal using this command: pip3 install pipenv
  + Download code editor.

**First Django Project**

* To create my first Django application I first go to where I want to create it, let’s say Desktop: cd Desktop and create project folder using: mkdir projectName.
* Then go inside the project folder and create a Django project using pipenv : pipenv install django
  + this will create a virtual environment for my project and provide me the location for that venv.
  + If I open the location provided I will see a Django project virtual environment.
  + also two files will be created inside my project contain Django.
* Then I activate the local environment inside my project using command: pipenv shell.
* Creating a project:
  + In order to create a project I use following command:
    - django-admin startproject projectName. (I use dot to create project inside the current directory)
    - or : python3 manage.py startproject
* to run server: python3 manage.py runserver
* the reason I use manage.py while also Django-admin have same commands is because Django-admin does not know about application settings.

**Creating Your First App (Django preinstalled apps and project structure)**

* setting.py file contains the application setting such ass installed app, cors, secret key, db configurations and so on
  + Django Apps:
    - Admin: an admin interface allows me to interact with my application such as crud operations and manage my application.
    - Auth: for authenticating application users.
    - Messaging: for send message only once for users.
    - Session: to manage user data on the server such as credentials and so on.
    - ContentTypes.
    - staticFiles.
  + To create a new app inside the project I run following command:
    - Command to create a new app: python3 manage.py startapp projectName.
    - Then add it to the apps inside setting.
    - The new app structure:
      * Migration folder: for generating database tables.
      * Admin: to define how the admin will interact with the model.
      * Apps: for configuring the app.
      * Models: where I define my model class and persist database.
      * Tests.py: to write unit tests for the application.
      * views.py: to define my application request handlers.

**Writing Views & Mapping URLs to Views**

* in order to create handler methods I need to configure it.
* Creating route steps:

1. Inside views.py:
   1. from Django.http import httpResponse to handle the user request with a response
   2. Define a method that take request object as parameter.
   3. Do some logic.
   4. Return response with HttpResponse().
2. Create url.py inside the app module.
   1. From Django.url import path so I can use the path constructor (it takes two parameters one is the url, and the second is the method created inside view)
   2. import the views file so I can use methods defined inside it.
   3. Create urlpatterns =[] list that take list of path including patterns related to the module that defined in view and serve them on url specified.
3. Inside the main app urls.py:
   1. import the include() method from Django.urls.
   2. Inside the urlpatterns in that file use the path constructor to link url to my app views
   3. Path(‘url/’, include(‘appName.urls’)) : this line of code means if anything come to this specific url/ go to my app urls.
      1. When app find url/ before it redirect it to myapp urls it removes the url/ and send the rest to my application

**Using Templates**

* Templates re html pages allow me to inject dynamic content inside html page.
* Server create the variables, then inject my backend code before sending the response to the user.
* To render template I have to follow following steps:
  + Create templates folder and inside it create html page.
  + Inside the views vile I create a path object that take two parameters.
    - The request.
    - The render() method contains a string or the file name
    - Returning example: return render(request, ‘index.html’)
  + Then add the path() object that contain the url we are going to serve the template on and the views.methodName.
* The method render take another dict parameter I can use it to pass dynamic content to the template.
* Inside the template I use douboe curly braces to inject variables {{ var\_name }}
* Also I can write some logic like so:

{% if condition %}

# do something

{% else %}

# do something else

{% endif %}

* + I must end the if statement

**Building a Data Model**

**Introduction to Date Modeling**

**Building an E-Commerce Data Model**

**Organizing Models in Apps**