**SECTION ONE (first\_test\_django\_project and first\_app)**

**Django Project**

* It act as a parent directory, it mean Overall an Web app or Web application
* To create the django project using django cli command : django-admin startproject <project\_name>
* This cmd create one folder(parent dir) with the givename, within that it will create one more folder(secondary directory) with the same name and manage.py file also created
* Few more files created under the second folder like \_\_init\_\_.py, setting.py,urls.py,asgi.py and wagi.py
* It is also called as **Project scope Files**
* To start the server : python manage.py runserver <PORT\_NUMBER> is used, it should run the parent directory

**Django App**

* It is sub-dir or sub-component
* It is main purpose is the make the huge project into multiple App/Module. For example, If we have Social Media application, we may have Message features, video call features. So we can put two app as message and video calling into single Django Project (Social Media Application)
* Goto the parent directory of the Django Project and create any number of apps into it.
* To create App : python manage.py startapp <APP\_NAME>
* This cmd will create a folder by given name and within that few files created like app.py admin.py , model.py , view.py and test.py
* It is also called as **App Scope Files**
* If needed or if our application big, we can create urls.py in the App Scope File
* We can have N no of App inside the Project
* Each app may have many views related files so we can configure it in the url.py which in the App Scoped Folders
* Then all Url.py file in the each app scoped folder, can be mapped into the url.py in the secondary folder which is created at the time of Project creation
* We can see the **first\_app** project which is done based the above two section.

**SECTION TWO (view\_route\_urls)**

**Views and URL**

* View means what information needs to show to the end-user
* URL means from where that information is shown to the end-user
* View is the webpage, using URL we can route to the desired view from the browser
* URL configuration can be in the both Project(Project scoped file) level and App(App scoped file) level using the urls.py file
* Using path() and include() Django functions we can configure the URLS in the urls.py files
* Urlpattern is the array type of variable used to wrap the path() and include() in the urls.py
* Connecting View to Url using path() function
* In the path() first argument is the **url** and second argument is the **view or function** and third optional arguments is **kwargs** and fourth optional arguments is **name**

**Dynamic Views**

* This is solution for the creating a each and every view whenever it is required
* Python Dictionary object will be helpful to make the View to ender dynamically
* Passing the topic as an argument into the view we can render the dynamic view with the help of dictionary.
* Path convertor can be done in Django, it means, we can receive the any data type from the request

**Redirects and 404**

* In the dynamic view we have one issue, if any value given which is not in the dictionary, it gives error
* We should handle it by giving proper response.
* We can use the Django 404 page to that error
* Redirects used to redirect the one page to another page using Http

**URL names and reverse()**

* In the above redirects is not dynamic way so reserve() will be sued
* For each url we can give a name
* That name can be used in the reverse() method across Django project

**Connecting to Templates**

* We wont want to type HTML code or HTML response directly into the view.py file
* To address this, template comes into the picture
* So we can move all out HTML files into separate folder and make the our view.py to communicate within them
* Recommended way to store the templates in the App level, but for now we can store it in the Project top level
* Edit the setting.py(TEMPLATES.DIRS key value) to inform Django Project to communicate the Template folder and View.py

**SECTION THREE(templates)**

**Template Directory**

* We should keep the template directory with the App Scoped Files
* It can be re-use in the multiple Django Projects wherever we need it
* To make aware the Template Directory in the App Scope to the Django Project , we should register the app in the INSTALLED\_APP keys in the setting.py
* STEPS :
  + Open cmd prompt and run “python manage.py migrate”
  + We need tell Django project , we have this app and needs to install, for that we should open the apps.py file in the corresponding each app in the project
  + In that file we can find a class has been generated and copy the name of the class
  + Goto setting.py file, add the path of this class in the INSTALLED\_APP variable
  + Run “python manage.py makemigrations <APP\_NAME>” we should run it for each and every app in our Django project.
  + Open cmd prompt and run “python manage.py migrate”
  + In setting.py make sure APP\_DIRS key needs to be true under the TEMPLATES Key
  + Create a folder under each app in the name of “templates” and also create a sub folder under this in the name of <CORRESPONDING\_APP\_NAME>
  + Place the html file in the created folder
  + Start the server and access the application

**Template Rendering / Django Template Language / Context Insertion:**

* We can pass the context value from the view to template.
* Context value can be any type, like variables, functions and another python file or re-direction logic and etc.
* In the render() we have one more parameter called “context”, it will pass the value to the html template.
* To access the passed value in the HTML, we should use syntax of double curly braces open and close {{ }}
* To access the python collection object the syntax should be little different , check in the official documentation

**Filter and Tags**

* Filters are built-in modifiers in Django templating that allow you to quickly apply a change to a variable on the template side, rather than in python script.
* We can place the filter in the template directly or same filter logic can be used in the view.py file also
* There are lot of easy and complex built-in filters available , please checkout in the official documentation.
* Tags are able to provide further logic at the template in rendering process.
* Tags includes for loop, if, else if and URL linking.
* Tags are denoted by {% %}

**URL Names in Templates**

* We should give one name to the app in the url.py file
* Give the name for each path
* Add <a>tag whereever we need and pass URL name in the href attributes like {% url ‘<app\_name>:<path\_name>’ %}

**Template Inheritance**

* Create a folder named “templates” at the top root level of the project
  + Or create folder named “templates” under the app level folder
* Create html file in this folder, which act as a base/parent html template
* Add {% block content %} {% endblock %}
* Create child html file inside the app level template folder
  + In this if you want to extend the base.html from the root level folder use {% extends '<HTML\_FILE\_NAME>' %}
  + In this if you want to extend the base.html from the app level folder use {% extends '<APP\_NAME>/<HTML\_FILE\_NAME>' %}

**Template Specifics – Static Files**

* Static file called as JS and CSS files
* Using Tags, Django can serve the static files instead of having full file path.
* To use the static file, in setting.py file double check the “Django.contrib.staticfiles” is available under the INSTALLED\_APPS
* And check the STATIC\_URL is available in the setting.py file, if you like to change the folder path, we can update under this name.
* As Template folder we should set the static folder
* Using {% load static%} and {% static <APP\_NAME>/<FILE\_NAME> %}

**Template Specific – Custom 404 file**

* Django comes up with the many build-in templates like 404 error page, admin page, user credentials entry page and etc
* But Django can able to overwrite these page based on our needed.
* We can do it the Project Level Template Directory and App Level Template Directory
* By default , it gives some unwanted information in the 404 error page.
* To Enable or show the Django 404 page or To show Django 404.html file under the Project level template folder, we should do the below changes in the settings.py
  + Change the Value of “DEBUG” from True to False
  + Add the IP address of localhost under the “ALOWED\_HOSTS”
* If no File is not under the name of 404.html, it will display the Django 404 html page.
* To use the custom error page other than 404.html filename, check the Django Documentation