Review Paper on Django Web Development

[1]Tokpam Sushilkumar Singh, [2]Harmandeep Kaur
[1]*Scholar; Department of Computer Application, Rayat Bahra University*
[2]*Lecturer; Department of Computer Application, Rayat Bahra University*

Abstract - This research paper examines Django, a Python-based web platform. It is an open source framework that adheres to the fundamental Model View Template structure with some modifications that are explained in the paper. It also helps us understand why we choose Django over other web frameworks that are available in the market, as well as how to install it on our system and use it to create a simple project. After studying the various modules that are available to us, our paper also studies how to understand the MVT structure. When we study more into the paper, we also learn how to develop applications inside the framework and add them to our primary project, how to build internal views of it, how it interacts with databases, and how it uses databases how centralizing access to the apps makes creating dynamic websites and tasks easier. This essay essentially covers everything needed for a person or student to get started with the Django framework, learn the fundamentals, develop some easy projects connected to the Django web framework, and make learning engaging and straightforward even for a lame person.

I. INTRODUCTION

Django is a web application framework which is open source and written in the Python language. It uses MVT design structure (MVT stands for Model View Template). Due to its rapid development feature. Django is very demanding in the current Market. It takes less time to build any kind of application. Why we say this Model View Template because this framework will work based upon the model as a database and view as a controlling functionality and template will work as a user side for communication interaction.

Using two key instructions, such as python manage.py make migrations, the Django model will manage the database. Django will take into account the modifications to the models.py file and be prepared to send data into the sqlite3 (or other database) file. Then we run migrate in Python manage.py. Then all changes will be saved by the Django system in his database system.

II. HISTORY

Django was planned, developed, and released to the public in 2003 by Lawrence. In 2005, the BSD licence was made public. Django Software Foundation is currently in charge of maintenance and new releases.
Django is widely recognised and used by several well-known websites, including:

1. Spotify
2. YouTube
3. Dropbox
4. NASA

III. INSTALLATION AND DEVELOPMENT OF A NEW PROJECT

- We must install Django on our local machine. If you haven't previously, install Python and Pip in your environment. After installing Python and Pip, run pip3 install Django in the terminal to install Django.
- After installing Django, we proceed to the following step, which is to create a new project.
- Use Django -admin start project Name to generate a set of startup files for our project.
- Type cd project Name to enter the new project folder.
- To start the server, type in the terminal python manage.py run server to start a local server on your PC.
- Using your web browser to view the default page, go to http://localhost:8000/.

IV. WHY IS DJANGO USED

- Open-source implies free.
- Development is faster.
- It is completely scalable

- Security is a top focus.
- Integrated administration portal

V. DJANGO'S MVT STRUCTURE

To understand Django's MVT structure, we must first understand what MVT structure is.
Model View Template is the full form of MVT.
MVT Structure is made up of three sections. Model 1. View 2. Template 3.
Model: This component of the MVC framework serves as a conduit for storing data from the user in the database. This is accountable for managing the logical component of the web application as well as how the data is stored in the data base.
Views: This is a graphical user interface. It is in charge of presenting data from databases and storing user-supplied information. Views Django are not the same as they are in the standard MVC framework.

VI. DJANGO APP DEVELOPMENT

Method 1: To create an app, navigate to the project directory using the terminal and type and run the following command: startapp python manage.py <APPNAME>

Method 2: To create an app, navigate to the project directory using the terminal and type and run the following command:
Django - admin app name start app
We will now obtain the directory structure of the app folder.
To make that app operate, we must include it in INSTALLED_APPS, which is found in setting.py in the proj1 directory.
After completing the preceding stages, we have finally constructed our app; however, in order to render it, we must supply the URL to our app so that the app is inside our main project and the URL we gave is redirected to that app. To accomplish so, we must do the following steps:
Move to project_directory -> project_directory - > urls.py and put this code within the file's header. include Django . urls import
Now, within the URL pattern code, we must give the name of the app for the URL of the app we generated; here is an example:

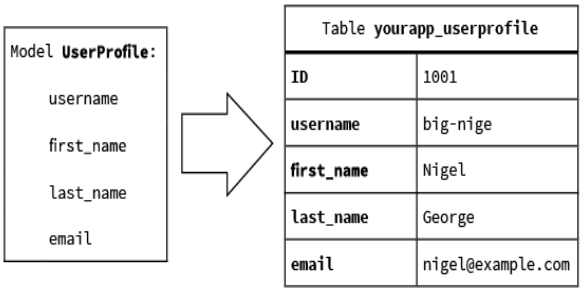
```
urlpatterns = [
    path('main/', main.site.urls)
   path('', include("proj1_name.urls")),
]
```

We can now utilise the basic MVT model to construct models, views, templates, and URLs within the app, and they will be instantly uploaded to the main project. This concludes this section, and we will now go to the next.
Models.py is the file that defines the model. The file may include several models.
The Django Model
Django Model provides a database abstraction API for creating, retrieving, updating, and deleting records from a map. Contains critical fields and behaviour for the data you save. Each model is typically mapped in a data table. Django Model a single SQL Database that is utilised by Django. Models simplify the process and organise tables into models. The mappings for each model are often stored in the same data table. Models in Django offer compatibility, simplicity, version control, and extensive metadata management.

```
from django.db import models
        class c1 (models.Model):
                    n1 =
models.CharField(max_length=30)
                    n2   =
models.CharField(max_length=30)
```

Django CRUD
Data creation, reading, updating, and deletion. Django enables us to exchange prior data.
Types : - add, remove, change, and query things using an ORM (Object Relational Mapper) database generator. Within our project guide, we can access Django ORM by running the following command. Most general information is organised using some kind of SQL, however each database employs SQL in its own unique method. SQL may also be complex and time-consuming to master. The ORM tool simplifies the database system by providing a simple map between an object (represented by the letter 'O' in

ORM) and a basic database. This implies that the producer does not need to understand the data structure and does not need to use complicated SQL to manipulate and retrieve data.
ORM enables simple data access without the need for complicated SQL.
The model in Django is anything that is specified in the database. Django utilises SQL to generate a comparable table in the database without requiring the user to write a single line of SQL. Django gives the table and you a nam
Request for Django. The model also connects related data to a database.



Making a Model
As an example, here is a genuine model that was produced. Our category contains four properties (three CharFields and one Integer), all of which will be table fields.

```
models.IntegerField()

class Meta:

db_table = "examplel"
```

VII DJANGO VIEW

Django views are made up of a collection of function classes. Within the vistas, everything is contained. Enter py in the app directory. A view is a user interface that appears in a browser when we are producing a website.
"A view function is a Python function that receives a web request and responds with a web response." Responses might take the form of HTML web pages, XML documents, pictures, or a 404 error. When a different URL is accessed, functions and logic are executed.

A simple view function:

```
from django.http import HttpResponse

import library_name

    def function_name(request):


n1=library_name.library_name.request_n
ame(:

        html="<html><body>Some Random
text according to need
      %s.</body></html>"%n1

       return HttpResponse(html)
```

Let's go over the steps:
Line 1 imports the class from the Django http module.
Line 2 imports the Python datetime library.
Line 3 defines a function named cur_datetime. This is exactly how a view function appears.
Lines 5–7 describe the structure of a web page, and the view returns a response as a HttpResponse.

VIII. DJANGO SUPPORTS TWO TYPES OF VIEWS

1. Views Based on Functions
2. Class-Based Perspectives

A function-based view is a Python function that accepts a web request and delivers a web response. The answer can take the shape of HTML text, an XML document, a 404 error, and so forth. The initial parameter of all view functions is a Http Request object.

```
def function_name(request):

    return
    HttpResponse('html/variable/text')
```

IX. TEMPLATES FOR DJANGO

Django's template programme makes it simple to construct sophisticated HTML. HTML, CSS, and Javascript are commonly used to develop Django templates. The Django template is well-managed and generates HTML pages that are visible to the end user. Django makes extensive use of background finishes, and templates are used to provide the framework of every website.

The template function requires three parameters:

1. Request is the initial request.
2. The path to produce templates - The TEMPLATE_DIRS option is associated with the project.py variables that are changing.
3. Parameters Dictionary - A dictionary that contains every element required by the template. We may use local persons () to transfer all of the views local variables, or we can construct our own variable.

X. DJANGO TEMPLATE LANGUAGE (DTL)

It offers a simple language for defining the program's front end, which the user sees.
A Django template is a Python thread that is used with the Django template's language.
A template engine is also aware of and can translate some structures.
Tags and variable variables. The template is given in context. Offers have variable values that are searched up in context and then labelled.
Django is a framework that allows us to divide Python and HTML, with Python going within views and HTML going inside templates. Django relies on dedicated performance and the language of the Django template to connect the two.
Tags: Tags allow us to work on jobs such as if status, loop, template asset, and many more.
Tag for: The 'for' tag, like 'if', functions in the same way as Python. Let's shift our perspective so we can relocate the list to our template.

```
def hello (request):
 n1 = library_name.library_name.calling().
Internal function ()
   samplearray =
   ['1', '2', '3', '4', '5', '6', '7']
   roll back (
       request, "m1.html",
      {"n1": n1,
     "calling_array":samplearray})
```

As we all know, HTML is a static markup language, and as a result, we cannot write Python code within our HTML. Because Python is a programming language, a browser cannot understand it.

XI. MAKIN USE OF DJANGO TEMPLATES

Templates not only display static data but include data from other sources linked to the dictionary.

```
<! DOCTYPE html>
<html lang = "en">
<head>
     <meta charset = "UTF-8">
     <meta name = "viewport">
    <title> sample template </title>
</head>
<body>
          <h1> Some title</h1>
          <p> Data is here {{data}} </p>

</body>
</html>
```

XII. DJANGO FORMS

HTML forms are a fundamental component of contemporary webpages. It is the most important source of gathering information from website visitors and users. Django has a Form class for creating HTML forms. We can accomplish all of the work from Django forms using sophisticated HTML, but Django makes it easier and more efficient for you, particularly the form validation component. You'll quickly forget about HTML forms if you love working with Django forms. Django completes three separate pieces of the forms work. It is preparing and reorganising data so that it may be rendered. It generates data forms in HTML. It also accepts and processes forms and client data that are supplied.

VIII. FORM CLASS IS USED TO CREATE DJANGO FORMS

Let's make a Django form using the form class. To do so, we'll need to create a new file within the application folder called forms.py. To construct a Django form, paste the following code into forms.py.

```
from django import forms

    class sample_form(forms.Form):

                id1=forms.CharField()

                #here length is not required

               id2=forms.CharField()
```

After that, when this form is rendering its look like this

```
<tr>
     <th>
   <label for="id1">label: </label>
    </th>

    <td>
            <input  type="text"  name="id1"
required id="id1">
    </td>
</tr>
```

```
<tr>
    <th>
     <label for="id2"> id2 :</label>
    </th>

    <td>
      <input  type="text"  name="id2"
        required id="id2" >
    </td>
</tr>
```

Display form the user

To display the form to the user, we must first construct an instance of the Form class in views.py, then pass the object to template files and utilise the Form object in the template file.

```
from django.shortcuts import render

from.forms import sample_form

def showformdata(request):

        fm=id1()
    return
render(request,"enroll/formdata.html",{'for
m':fm})
```

After that, the context form is sent into the formdata template, which looks like thisgs:

//templates/enroll/formdata.htm

```
<!DOCTYPE html>
<head>
     <meta charset="UTF-8">
     <meta name="viewport"
        content="width=device-width,
        initial-scale=1.0">
   <title>FormData</title>
</head>

<body>
<form action="" method="get">
```

Now, provide the URL in urls.py file.

```
from django.urls import path
from . import views
urlpatterns=[
      path('new/',views.showformdata),

 ]
```

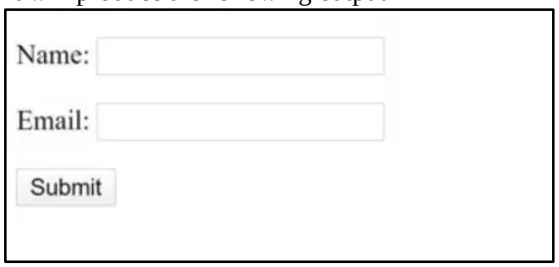
Now, Include the above url in the main urls.py file in the inner project folder.

```
from django.contrib import admin

Form django.urls import path, include

urlpatterns=[
       path('main/',admin.site.urls),

       path('new/',include(enroll.urls))
  ]
```

Now, It's time to save the all code and run the server and access the form to achieve this type

Command: python mange.py runserver

It will produce the following output-



XIV. CONFLICTS OF INTEREST

The authors state that they do not have any conflicts of interest.

1. Adamya Shyam and Nitin Mukesh are two references. An Educational Resource Based on

Django References Website for sharing: Shreic, Volume 64, Issue 1, 2020.
2. Josh Juneau is number two.James BakerSoto Victor NgLeoWeb Applications With Django, Frank Wierzbicki, The Definitive Guide To Jython, pages 281-325
3. Jian Chou, Lin Chen Hui Ding, Jingxuan Tu, and Baowen Xu, 2013 10th Web Information System and Application Conference, A Method of Optimising Django Based on Greedy Strategy
4. Arnold Rosenbloom, The 23rd Western Canadian Conference, A Simple MVC Framework for Web Development Course
5. Abhishek Bera, Shubham Darda, Ashish Gaikwad, Shivam Chanage, Prof. Suresh Rathod, Endorseme - Professional Networking Using Django, Volume 6, Issue 5, May 2016.
6. M. Aniche, G. Bavota, C. Treude, M.A. Gerosa, and A. van Deursen, "Code smells for Model-View-Controller architectures," Empirical Software Engineering, vol. 23, no. 4, pp. 2121-2157, 2018.
7. Jeff Forcier, Paul Bissex, and Wesley J Chun Django Web Development
8. Django for Beginners: Build Websites with Python and Django by William S. Vincent
9. Nigel George Create a Website with Django 3: A Comprehensive Introduction to Django
10. Django for APIs: Build Web APIs with Python and Django, William S. Vincent

Controller: In MVC, this component is in charge of the whole logic and workings of the web application. When a user makes an HTTP request, the controller receives it and returns the appropriate answer.

Creating App in Django

Method 1: To build an app, navigate to the project directory using the terminal and type and run the following command: *python manage.py startapp<APPNAME >*

Method 2: To create an app, navigate to the project directory using the terminal and type and run the following command:

django-admin startapp app_name

To make that app run we need to mention it under INSTALLED_APPS which is located inside setting.py inside the proj1 directory. After completing the above steps we have finally created our app but to render it we need to specify the URL to our app so that the app is inside our main project and the URL we provided will be redirected to that app. To do so we need to follow the given steps :

Move to project_directory -> project_directory - > urls.py and we need to add this code inside the header of the file.

XV.CONCLUSION

Django has evolved as a robust and popular web development framework with various advantages for developers. Its scalability, stability, and adherence to the Model-View-Controller (MVC) architectural pattern make it an excellent choice for developing sophisticated and feature-rich online applications. Django offers developers a simplified workflow with its broad set of built-in tools and frameworks, decreasing the time and effort necessary for application development.