Parthib Kumar Deb *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 9, September- 2022, pg. 1-12 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Setting up the Django Framework for Web Development

Parthib Kumar Deb[1]; Anirban Bhar[2]; Soumya Bhattacharyya[3]
[1]B. Tech student, Department of Information Technology, Narula Institute of Technology, Kolkata, India
[2,3]Assistant Professor, Department of Information Technology, Narula Institute of Technology, Kolkata, India
[1] parthibkumardeb@gmail.com
[2] anirban.bhar@nit.ac.in
[3] soumya.bhattacharyya@nit.ac.in

DOI: 10.5281/zenodo.7088915

Abstract
Web development has now become the most valuable and necessary expertise in the software development business. Web development has existed for three main reasons: brand, product, and information. In the twenty-first century, everyone wants and uses a specific website to expand their ambitions, from business and e-commerce to IT industries, from hospitals and medical services to e-learning. As a result, the need for full-fledged web developers is fast expanding. According to a Stack Overflow Developer study, full stack web development will be the most popular occupation in 2022 (49.47 percent), and DJANGO will be the 10th most popular web framework (55.28 percent of workers worldwide). The Django framework is built on the well-known programming language PYTHON, which works on Backend of a website. Django is used to create a variety of strong, practical web applications that we use frequently in our daily lives, including chat, video conference, navigation, and food ordering apps. Django also allows us to develop different application programming interphases (APIs).

***Keywords**:* Full Stack Web Development, Django, Application Programming Interphases (APIs)

1. Introduction
Building Dynamic Websites face the challenge of reinventing certain standard features over and over. In general, web development is a messy process. Incompatibilities between browsers, malicious bots, bandwidth and server restrictions, and a design that appears to defy extensive testing are among issues that developers must deal with. Taking care of many of those problematic areas—the 20% of the job that can consume 80% of the developer's time—is another goal of this effort. The questions and difficulties facing the developers have remained the main focus. Python and Django are excellent choices for addressing the need for full-stack web development. Django is a high-level framework that makes it possible to create Web applications with only a small amount of code. Its simplicity, dependability, and flexibility allow for the quick design of solutions. Python, an object-oriented application development language that combines the strength of systems languages like C/C++ and Java with the simplicity and speed of scripting languages like Ruby and Visual Basic, was used to create Django. This enables its users to produce software that addresses a wide range of issues.
Some of the necessary Python abilities for a successful Django developer are demonstrated in this work. The emphasis is on those Python ideas that are "must-haves" for Django developers.
Basically, there are 2 components of any website: FRONT-END, and BACK-END.

FRONTEND: A website's frontend is made up of two elements: the graphic design (look) and the user interface (the feel). This section of a website handles user queries and routes them to the backend. The creators use HTML (HYPERTEXT MARKUP LANGUAGE), CSS (CASCADING STYLE SHEETS), and JAVASCRIPT, which are the three essential building blocks of every website, for this part exclusively. If we use the human body as an example, HTML is the basic building block of a website, just as bones are in the actual body. CSS is used to make the website more appealing to users, much like our skin, and JAVASCRIPT is the events that occur in response to the user's requests. Any front-end developer's actual job is to do this.

BACKEND: A website's backend is made up of three components: a server, an application, and a database. The technology that powers those components, which enables the frontend, is built and maintained by a backend developer. To put it another way, the server delivers data on demand, the application channels it, and the database organises it. That is, the backend sends a cluster of data to the browser in response to the user's request, which the browser divides apart and displays.

FRAMEWORK: A framework, or more specifically a software framework, is a platform that allows users to specialise common code with broad capabilities. Frameworks are a type of library that is language dependent and has a well-defined API (Application Program Interface) that may be reused anywhere. The goal of adopting any framework is to make the development environment easier to navigate and allow developers to focus only on their projects.
Examples:

FRONT-END: REACT.JS, ANGULAR.JS, VUE.JS, TAILWIND CSS etc.
BACK-END: NODE.JS, DJANGO, SPRING BOOT etc.

2. Related Work

Setting up a solid framework hierarchy is the first step in framework design. Smaller elements like a grid, buttons, and icons should be designed first, then larger elements like colour and font. We also need to keep testing the design framework [1].
The delivery of services such as storage, servers, analytics, and more is known as cloud computing. over the internet, which offers advantages such as pricing, flexibility, performance, and safety. There are various types of cloud installations, such as public, private, and hybrid clouds, allowing the user to select the one that best suits their needs. Examples are Gmail's drop container and cloud tweets [2].
It explains the characteristics and actions of an internet page. Hyper Text Markup Language is what HTML is. The W3C has advanced it. Without the use of any 0.33-celebration programmes, it is possible to include images, animation, audio, and other types of content [4].
Any internet site's homogeneity, layout, and navigation can be improved with the usage of templates. Templates are essentially pre-designed web pages; they are a collection of HTML code that streamlines the process of creating web applications. Website templates are a quick and easy way to reduce burden. Mapping Relating to Objects ORM is a kind of digital database for devices that may be used directly from the programme. Reduced coding complexity is the primary purpose of ORM [3].
The Open Net Software Security Initiative makes it easier for enterprises, organisations, and other entities to create, update, and maintain applications. OWASP can be used to prevent specialised attacks like XSS and SQL injections that trade in order to obtain information [5].
Selecting an appropriate platform for the frameworks should be one of the main challenges. Typically, frameworks are made to run on the majority of the current operating systems, including Linux, Windows, and others [6].
Debugging is the process of identifying and eliminating errors or flaws from code. Additionally, the builders might find this particular movement challenging. Debugging can help or transfer the developer's false alert. To

find bugs in the code, it employs the debugger. The frameworks Meteor, Flask, Larvae, Phoenix, spring, and express are other excellent examples [7].

3. The Django Framework

The Django framework was created in Lawrence, Kansas, press agency in 2003. Django is a Python-based framework that is released under the BSD licence, ensuring that online applications can be freely used and modified. It's a backend framework, which means it's in charge of any DYNAMIC WEBSITE'S SERVER-SIDE RENDERING.

3.1 Characteristics

● Django is a framework whose primary objective is to quickly create dynamic web pages. Django takes care of the developers' headaches when building a website, which is why it comes with a lot of pre-installed packages and files.
● Django was created for users who want understandable and straightforward code that can be reused, and this leads to Django's second feature: the DRY (Don't Repeat Yourself) philosophy, which allows us to copy the required parts of a Django-codebase and use them elsewhere.
● Django is completely flexible; thus, developers can simply adjust to it by writing modules or overriding framework methods.
● Because Django is based on Python, suitable OOPS concepts are used to create database schemas.
● ORM stands for OBJECT RELATIONAL MODEL, and this feature in Django elevates it to a new level by allowing developers to interface with databases and perform CRUD (CREATE, READ, UPDATE, DELETE) operations on specific MODELS to make database changes.
● Django comes with its own administration panel. It reads metadata from models to give a rapid, model-centric interface where trusted users can simply manage material thanks to this significant key feature. We can also customise this panel according to our needs.
● Django places a high priority on website security, making it a more secure backend for developers. That's why Django has a special token called the CSRF (CROSS-SITE REQUEST FORGERY) token. This token ensures that forms (such as contact forms) are only submitted by trustworthy users using the GET or POST methods.

3.2 Architecture

MVT (MODEL-VIEW-TEMPLATE) architecture is used by Django. This architecture, unlike MVC, uses raw HTML FILES in the TEMPLATES section to provide flawless coordination between MODELS, VIEWS, and TEMPLATES.

MODELS: A model is the ultimate source of data information provided by the user. It contains the data's most important fields and behaviours. In most cases, each model corresponds to a single database table. Some user-defined objects are inserted into a model. The MODELS.PY file contains all of the models (Later on, this topic will be discussed).

VIEWS: This file, also known as VIEWS.PY, is responsible for rendering the template files. Essentially, we must develop multiple user-defined METHODS (functions) that render the templates, and these methods are also utilised to generate the entire URLs of the website in the URLS.PY file.

TEMPLATES: Finally, this is the key component that goes into creating the website's FRONT-END. In the .html files, we actually write HTML, CSS, and JS (Later on, this topic will be discussed).

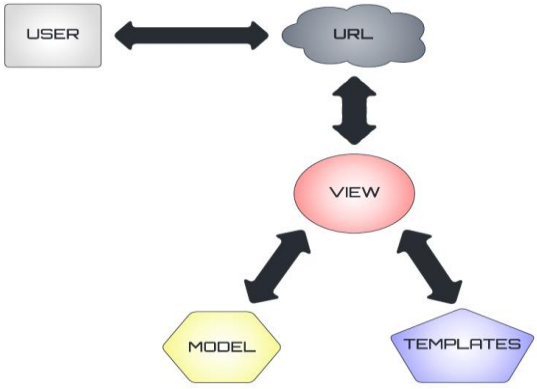

Figure 1: The Working strategy of MVT architecture

So, basically in MVT architecture the VIEWS.PY extracts all data from MODELS.PY and renders those via TEMPLATES. Thus, the architecture works. Because of these amazing features, popular applications like INSTAGRAM, SPOTIFY, YOUTUBE, DISQUS, BIT-BUCKET, EVENTBRITE, MOZILLA, and others use Django as a backend.
Despite being strong and well-known for its development capabilities, the Django web framework is still not the main one for many programmers. Django has some drawbacks, and seasoned developers find it challenging to adapt. The following is a list of several drawbacks to Django development.

● Due to its extensive coding requirements, which soak up server processing time and bandwidth while being developed, the Django Framework is not ideal for smaller projects or products with few features. It is typically utilised for projects that require scaling or that will be launched on a huge scale. Small-scale developers do not wish to use this framework for development because of this.

● The Django framework offers a specific method for task definition and execution. The file structure is reasonable and simple to understand. However, this also makes it necessary for you to be unable to use your own file structure. The reason for this is that the framework adheres to a method of operation that is referred to as its own way. You might not be able to use Django to deploy anything if you don't go by their guidelines.

● Compared to other frameworks, Django is unique, although this distinction can occasionally work against it. Contrary to the majority of web development frameworks, Django cannot manage many requests at once. They are requests for specific operations, and it takes time to complete each request. Django falls short in this regard by failing to offer quick development.

That's why, one should use Django for some specific conditions, which are:
1. For developing a website having an API as BACKEND.
2. For RAPID and SCALABLE development of a project.
3. An ideal ORM for working directly with databases as opposed to doing database queries.

4. Making Projects in Django

With some few simple steps, a project in Django can be created.

4.1 Installation of Python

● For WINDOWS, we have to download Python from ***python.org*** . Now if any system contains windows of version 7 or lesser than it, then latest versions of python like 3.10 will not support that system. That's why it is required to choose the perfect version of python.
● For LINUX, packet manager APT can be used.
● Like the users of windows, MAC-OS users have to visit the site to get the current version of python.

Figure 2: Download page of PYTHON

4.2 Installation of IDE

The IDE (Integrated Development Environment) is a software development environment that may be used to create any type of programme. These are intended to assist developers in their work environment and help them be more productive. TEXT EDITING, COMPILING /INTERPRETING, and DEBUGGING are some of the most basic functions of an IDE. However, there are other popular IDEs on the market, such as PYCHARM, SPYDER, PYDEV, and so on, and we will use VS CODE as our IDE.
So, from Visual Studio Code website, we will download our IDE as per our system's OPERATING SYSTEMS.

Parthib Kumar Deb *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 9, September- 2022, pg. 1-12 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

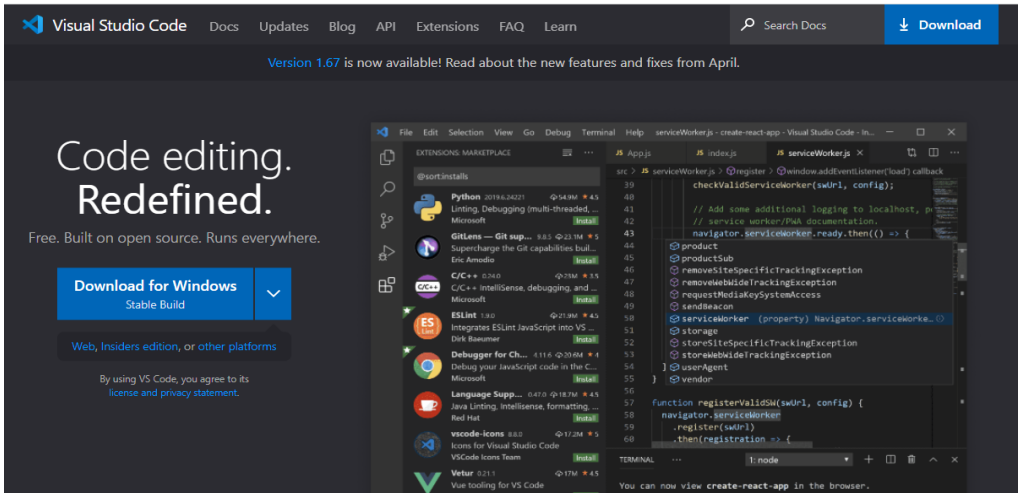

Figure 3: Download page of VISUAL STUDIO CODE

4.3 Installation of Django

As a result, we'll utilise PIP to install DJANGO via the VS CODE console or power shell. PIP is a Python package manager that installs Python packages using the Python Package Index.

However, it is to be established first a project folder on own device. Then, using this command, Django must be installed in that exact folder [8].

COMMAND: pip install Django

Projects must be built after installing Django. If a user wants to build projects in the VIRTUAL ENVIRONMENT, the following steps should be followed:

- The python-scripts path must be copied (here, F:\PY 3.10.4\Scripts) with a right-click on This Pc.
- Then need to proceed to Advanced System Settings by clicking on Properties.
- Finally, a click on Environmental variables is necessary, then further click on Path, and finally to be paste it.

Let's look at what manage.py and Django-admin are, because from here on out, every command will be generated by one of these two crucial files.

- **DJANGO-ADMIN and MANAGE.PY**

Django-admin is a command-line utility for administration tasks in Django. It creates a site space for the user to perform various tasks (like Create, Read, Update, Delete). This saves a lot of time during development and makes testing user modules very simple.

Manage.py is similar to Django-admin in that it assists users in managing their Django projects by pointing to the project's settings.py file.

Now, to create our first Django project, if the command Django-admin just run, numerous commands will be seen. Among them startproject has been taken as the main command. Then it is required to choose a name for the suitable project, let's say: MYPROJ.

So, the final command will be: Django-admin startproject MYPROJ

Parthib Kumar Deb *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 9, September- 2022, pg. 1-12 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

```
PS F:\DJANGO PROJECTS> Django-admin startproject MYPROJ
PS F:\DJANGO PROJECTS>
```

Figure 4: Creation of MYPROJ

5. Django Project Structure

A project in Django is made up of the following files: init.py, asgi.py, settings.py, urls.py, wsgi.py, and manage.py. All of these files, runs under the main APP, which has the name of the project.

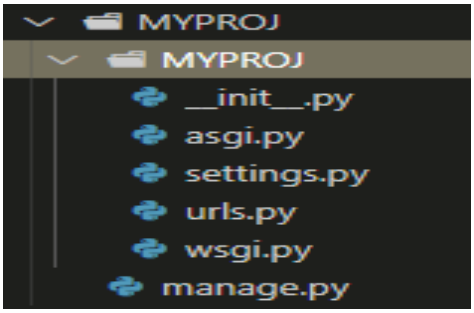

Figure 5: Files of MYPROJ

5.1 *init.py*

It's a python file in the package directory that is called when the package or one of its modules is imported. This file is used to identify python package folders on your hard drive. The file is normally left blank, but it can be used to export parts of the package under a more convenient name.

5.2 *wsgi.py and asgi.py*

Web Server Gateway Interface (WSGI) is an acronym for Web Server Gateway Interface. WSGI, the Python standard for web servers and applications, is Django's primary deployment platform. This file is primarily needed to deploy the project in Django. It's used to communicate with the web server and with our Django application.

```
<DJANGO WSGI CODE>
      import os
      from django.core.wsgi import get_wsgi_application
      os.environ.setdefault('DJANGO_SETTINGS_MODULE','MYPROJ.settings')
      application = get_wsgi_application()
```

ASGI stands for Asynchronous Server Gateway Interface. ASGI acts as a bridge between async Python web servers and applications, and it supports all WSGI features. Actually, in Django, ASGI is a channel that handles, in addition to HTTP, Websockets, Chat Protocols, IoT Protocols, and other protocols. In short it helps to make Real-Time Web Applications.

```
<DJANGO ASGI CODE>
      import os
      from django.core.asgi import get_asgi_application
      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'MYPROJ.settings')
      application = get_asgi_application()
```

5.3 *settings.py*

All Django projects must have a file called Settings.py. It holds all of the configuration parameters essential for a web app to function effectively, such as database settings, logging configuration, static file location, and API keys (if the user wants to work with external APIs).

A settings.py is just a python module with module level variables.

● BASE_DIR: BASE_DIR points to the project's top hierarchy, which in our instance is MYPROJ. All paths defined in the project are relatives of BASE _DIR. We will need to use the OS module given by Python to use BASE_DIR.

```
<DJANGO BASE_DIR CODE>
      BASE_DIR = Path(_file_).resolve().parent.parent
```

● DEBUG: Error is highly visible in development. There is no joy in writing a programme that is error-free. However, dealing with errors can be stressful at times. Django includes an in-built debugger, which makes life easier for developers.

● ALLOWED_HOSTS: ALLOWED_HOSTS is loss of having addresses of all domains which can run our Django project.

```
<DJANGO DEBUG CODE>
      DEBUG = True
      ALLOWED_HOSTS = []
```

● INSTALLED_APPS: Django includes a registry of installed applications that maintains configurations and allows introspections. It also keeps track of available models. So, the installed apps in Django are-

1. Contrib.admin – The ADMIN site.
2. Contrib.auth – An AUTHENTICATION system.
3. Contrib.contenttypes – A framework for content types.
4. Contrib.sessions – A SESSION framework.
5. Contrib.messages – A MESSAGING framework.
6. Contrib.staticfiles – A framework to collect static files from everywhere of a project.

```
<DJANGO INSTALLED_APPS CODE>
      INSTALLED_APPS=[
      'django.contrib.admin',
      'django.contrib.auth',
      'django.contrib.contenttypes',
      'django.contrib.sessions',
      'django.contrib.messages',
      'django.contrib.staticfiles',
      'APP']
```


● MIDDLEWARE: Middleware in Django is a lightweight plugin that runs during request and response execution. Middleware is used in applications to execute a specific function. Security, session, CSRF protection, authentication, and more functions are possible.

```
<DJANGO MIDDLEWARE CODE>
      MIDDLEWARE = [ 'django.middleware.security.SecurityMiddleware',
      'django.contrib.sessions.middleware.SessionMiddleware',
      'django.middleware.common.CommonMiddleware',
      'django.middleware.csrf.CsrfViewMiddleware',
      'django.contrib.auth.middleware.AuthenticationMiddleware',
      'django.contrib.messages.middleware.MessageMiddleware',
      'django.middleware.clickjacking.XFrameOptionsMiddleware', ]
```

● DATABASES: Django generally supports sqlite3 as its default database but it also supports PostgreSQL, MariaDB, MongoDB, MySQL, Oracle.

```
<DJANGO DATABASE CODE>
      DATABASES = {'default': {'ENGINE': 'django.db.backends.sqlite3',
      'NAME': BASE_DIR / 'db.sqlite3', } }
```

● STATIC_URLS Provides a suitable url to store static file (e.g: images, videos etc).

5.4 *urls.py*

In urls.py, the most important thing is url patterns tuple where the user can define the mapping between URLS and views.

```
<DJANGO URLS.PY(MYPROJ) CODE>
      from django.contrib import admin
      from django.urls import path,include
      urlpatterns = [ path('admin/', admin.site.urls),
      path('include('APP.urls'))]
```

Django is well-known for its distinct and fully controlled app structure. An app may be designed as a totally standalone module for each capability.

Some benefits of using Django apps are—

- Django applications are reusable, which means they may be utilised in a variety of projects.
- It becomes easy to debug and organise code.
- It contains several built-in capabilities, such as ADMIN PANEL, that decrease the time and effort required to create one from scratch.

So, actually we are creating our application, which is APP. our final command will be : python manage.py startapp APP.

6. Django Application Structure

A typical Django application includes the files migrations(init.py) init.py, admin.py, apps.py, models.py, tests.py, and views.py.

Here, we must specifically write another urls.py inside the APP to communicate with PROJECT URLS.

6.1 *admin.py*

The admin.py file is basically used to show user created MODELS inside the Django-Admin Panel.

To do Create, Read, Update, Delete operations with those models, inside the admin panel, we must declare a superuser at first via python manage.py createsuperuser command.

6.2 *apps.py*

This file's purpose is to assist the user with incorporating the app's customizations.

```
<DJANGO APPS.PY CODE>
        from django.apps import AppConfig
        class AppConfig(AppConfig): default_auto_field =
        'django.db.models.BigAutoField'
        name = 'APP'
```

6.3 *tests.py*

Django has a test framework with a tiny hierarchy of classes based on the Python unit-test library standard. In Spite of its name, this test framework may be used for both unit and integration testing (Testing all parts of a codebase together).

- Creation of models.py and Connection with Migration File:

Django is ideal for building database-driven websites because it has simple yet powerful ways to run database queries with Python. The brief introduction of models in the MVT section was previously recognised. Every model is a subclass of a Python class. A database field is represented by each model attribute. Django provides you with a database-access API that is automatically created.

A sample model is given by:

```
<DJANGO MODELS.PY CODE>
from django.db import models
        class Employee(models.Model):
        First_Name = models.CharField(max_length=30)
        Last_name = models.CharField(max_length=30)
```

There are two fields in this model: First Name and Last Name. Each field is defined as a class attribute, and each attribute corresponds to a database column. The Employee model would be used to generate a database table. When a user wishes to add a new model to the database, they must use the python manage.py makemigrations first, then execute python manage.py migrate to place the models in the STAGING section and store them permanently to the database. One MIGRATION FILE is formed after entering a new schema into the database (check within migrations), and this is how, when we make any modifications inside any model or insert new ones, more migration files are generated.

- Model Field References:

These are actually the types of fields Django offers to the user which help the person to create a model. Some field types er—
A. CHARFIELD- A string field, for small-sized strings.
B. DATEFIELD- A field to show dates.
C. INTEGERFIELD- A field to store integer values from -2147483648 to 2147483647.

- Django Templating System:

The template system in Django makes it simple to create dynamic HTML pages. This subject was already covered in the MVT architecture section. So, what's the point of using a Django template? We can't write python code in.HTML files since the code is only understood by the python interpreter, not the browser. HTML is a static MARKUP LANGUAGE, but Python is a dynamic programming language, as we all know.

As a result, we use the Django template engine to isolate the design from the Python code, allowing us to create dynamic web pages. We'll also make a static folder where we can store media assets like zombotron photographs and movies, among other things.

One need to be conscious that don't store any relevant server related information in the templates and static file folder. Because anyone can find these two folders while surfing into that specific website. That's why, It is suggested to store any public information, which you want to show them like public CSS files, public JS folders etc.

- Locating Django Template and static files:

To begin, we must manually enter the location in the DIRS of the MYPROJ settings.py file. We'll need to make a STATICFILES DIRS for static files.

```
<DJANGO TEMPLATES(SETTINGS.PY) CODE>
      TEMPLATES = [ { 'BACKEND':
      'django.template.backends.django.DjangoTemplates', 'DIRS': [BASE_DIR
      / "templates"], 'APP_DIRS': True, 'OPTIONS': { 'context_processors':
      [ 'django.template.context_processors.debug',
      'django.template.context_processors.request',
      'django.contrib.auth.context_processors.auth',
      'django.contrib.messages.context_processors.messages', ], }, }, ]
```

Template engines can be included or excluded from a Django project. Django comes with built-in backends for its template system, which it refers to as Django Template Language (DTL). Users can also create their own unique backends. Variables and Tags are the fundamental structural parts of a Django template. Let's know about these–

VARIABLES: A variable is any text enclosed by a pair of curly braces. That is, the value of the variable must be inserted within the variable name. The context, which is a dictionary-like object that maps keys to values, produces a value for any variable.
E.g: I have learned {{ NAME_1 }} and {{ NAME_2 }} for web development.
Now, the template will take values of NAME_1 and NAME_2 from
{'NAME_1' : 'Python' , ' NAME_2' : 'Django'} and will replace those values inplace of keys.
That means the complete result will be: I have learned Python and Django for web development.

FILTERS: Variables and filters are nearly identical, with the exception that FILTERS are used to modify the TITLE NAME of any page, whereas VARIABLES are used inside the content of that page. It also functions as a dictionary.
E.g : {{ MY_PROJ | TITLE }}, here the title name will get replaced after getting its value from {'TITLE': 'HOME'}

TAG: A tag or a BLOCK-TAG is any text enclosed by a pair of curly braces and a percent sign. Some types of block-tags are:

- Control-structure tags: Tags like {% if %} and {% else %} are called control-structure tags.
- Iteration tags: Tags like {% for item in list %} calls control-structure tags.
- Security tags: {% csrf_token %} is called security token, previously we have discussed this in the CHARACTERISTICS OF DJANGO section.

7. Conclusion

Despite the fact that Django has a few disadvantages, it is still helpful in the web development industry. Additionally, it adheres to the fundamental idea that building a website begins with understanding HTML, but in many contemporary frameworks, HTML is going to be replaced by JS and its packages. Because of this, many large IT companies and startups continue to use this framework as a solid BACKEND.

References

[1] Chen-Becker. D, M. Danciu, T. Weir, The Definitive Guide to Lift: A Scale-Based Web Framework, new edition, first Press, Berkeley, CA, New York, 2009.
[2] Josh Juneau Jim Baker Victor Ng Leo Soto Frank Wierzbicki. The Definitive Guide Web Applications with Django.
[3] Jeff Forcier, Paul Bissex, Wesley J Chun Python Web Development with Django.
[4] https://www.youtube.com/results?search_query=top+frameworks+for+websites&sp=CAM% 253D.
[5] Nigel George Build a Website with Django 3: A complete introduction to Django.
[6] Maria del, rafael-gracia, Giner alor analyzing best practices on Web development frameworks: The lift approach.
[7] William S. Vincent Django for Beginners: Build websites with Python and Django Book.
[8] Gore, H., Singh, R. K., Singh, A., Singh, A. P., Shabaz, M., Singh, B. K., & Jagota, V. (2021). Django: Web Development Simple & Fast. Annals of the Romanian Society for Cell Biology, 25(6), 4576-4585.

