



## **“DISCUSSION FORUM”**

### **A MINI PROJECT REPORT**

*Submitted by*

**ANISHA S HIEMATH [1NH18IS009]**

*Under the guidance of,*

**Mrs. K M Bilvika**

**Assistant Professor, ISE, NHCE**

*In partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION SCIENCE AND ENGINEERING**

**FOR**

**COURSE NAME: MINI PROJECT**

**COURSE CODE: 20ISE59**



## **CERTIFICATE**

Certified that the project work entitled “**DISCUSSION FORUM**” carried out by **Ms. ANISHA S HIREMATH**, USN **1NH18IS009**, a bonafide student of V semester in partial fulfillment for the award of Bachelor of Engineering in Information Science and Engineering of New Horizon College of Engineering, an Autonomous institute under the Visvesvaraya Technological University, Belagavi during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

**Signature of the Guide**

**Mrs. K M Bilvika**

**Signature of the HOD**

**Dr. R.J Anandhi**

**Signature of the Principal**

**Dr. Manjunatha**

**Examiners:**

**Name**

**Signature**

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

\_\_\_\_\_

## ACKNOWLEDGEMENT

Any project is a task of great enormity and it cannot be accomplished by an individual without support and guidance. I am grateful to a number of individuals whose professional guidance and encouragement has made this project completion a reality.

I have a great pleasure in expressing my deep sense of gratitude to the beloved Chairman **Dr. Mohan Manghnani** for having provided me with a great infrastructure and well-furnished labs.

I take this opportunity to express my profound gratitude to the Principal **Dr. Manjunatha** for his constant support and management.

I am grateful to **Dr. R J Anandhi**, Professor and Head of Department of ISE, New Horizon College of Engineering, Bengaluru for his strong enforcement on perfection and quality during the course of my project work.

I would like to express my thanks to the guide **Mrs. K M Bilvika**, Assistant Professor, Department of ISE, New Horizon College of Engineering, Bengaluru who has always guided me in detailed technical aspects throughout my project.

I would like to mention special thanks to all the Teaching and Non-Teaching staff members of Information Science and Engineering Department, New Horizon College of Engineering, Bengaluru for their invaluable support and guidance.

**ANISHA S HIEMATH**

**1NH18IS009**

## ABSTRACT

Discussion Forum is nothing but a place (in our case a website) that enables us to post our views over a certain topic publicly and also allows us to see other's opinions on the same. This is a web application developed using python and a very popular python framework called "Django" which provides many inbuilt functionalities and thus make our work much simpler and less time-consuming.

Project Prerequisites:

To implement this project, we need to have a knowledge of the following:

- 1) Basic concepts of Python
- 2) HTML
- 3) CSS
- 4) Bootstrap
- 5) Django framework

## TABLE OF CONTENT

<b>CHAPTER 1: INTRODUCTION-----</b>	<b>1</b>
1.1 PROBLEM DEFINITION	
1.2 PROPOSED SYSTEM	
1.3 OBJECTIVES	
 <b>CHAPTER 2: SYSTEM REQUIREMENTS-----</b>	 <b>3</b>
2.1 HARDWARE REQUIREMENTS	
2.2 SOFTWARE REQUIREMENTS	
2.3 FUNCTIONAL REQUIREMENTS	
 <b>CHAPTER 3: ARCHITECTURE-----</b>	 <b>4</b>
 <b>CHAPTER 4: PYTHON FEATURES-----</b>	 <b>5</b>
4.1 FEATURES	
4.2 CHARACTERISTICS OF PYTHON	
4.3 APPLICATIONS OF PYTHON	
 <b>CHAPTER 5: HTML, CSS-----</b>	 <b>8</b>
5.1 FEATURES OF HTML	
5.2 APPLICATIONS OF HTML	
5.3 HTML TAGS	
5.4 FEATURES OF CSS	
 <b>CHAPTER 6: DJANGO FRAMEWORK-----</b>	 <b>11</b>
6.1 DJANGO	
6.2 ARCHITECTURE	
6.3 FILES IN DJANGO PROJECT ROOT DIRECTORY	
6.4 IMPLEMENTATION	
6.5 DJANGO FLOWCHART-	

<b>CHAPTER 7: BOOTSTRAP-----</b>	<b>26</b>
<b>7.1 FILE STRUCTURE</b>	
<b>7.2HTML TEMPLATE</b>	
 <b>CHAPTER 8: IMPLEMENTATION-----</b>	<b>29</b>
 <b>CHAPTER 9: OUTPUT SNAPSHOTS-----</b>	<b>41</b>
 <b>CHAPTER 10: CONCLUSION-----</b>	<b>45</b>
 <b>BIBLIOGRAPHY-----</b>	<b>45</b>

## **CHAPTER 1**

### **INTROUCTION**

- This website “ONLINE DISCUSSION FORUM” is made for providing a platform for having discussions.
- This forum provides the platform under one roof to interact with different members which maybe the experts in a particular field or a normal employee for seeking or to give advices.
- An online space on which anybody can have discussions regarding any research, academic documents or any latest technology free of cost.
- A website which helps to resolve doubts, queries related to any field by having discussions with other registered users.

#### **1.1 PROBLEM DEFINITION**

- With highly growing of the telecommunication infrastructures, such as Internet and the development of the high performance have brought about new era of rapid advances in information technology.
- Internet has become an ocean of information related to every aspect which has existence in this world. The main purpose of this website is to develop a one roof platform for the effective interaction, effective exposure, and a right direction toward communication.
- My aim is to provide our users an opportunity to enhance their knowledge by sharing their views on this platform by having discussions with other users.

#### **1.2 PROPOSED SYSTEM**

- HTML, CSS and JAVASCRIPT are used here as front end which are the most portable languages used in GUI applications.

### 1.3 OBJECTIVES

- To be able to build an interface that can connect public to local service providers.
- To be able to design an interactive GUI using html, CSS and JavaScript to make it user-friendly.
- To be able to provide our users an opportunity to enhance their knowledge by sharing their views on this platform by having discussions with other users.



## CHAPTER 2

### REQUIREMENTS

#### 2.1. HARDWARE REQUIREMENTS

- **Processor** : above 500 MHz
- **RAM** : 4GB
- **Hard disk** : 500 GB

#### 2.2. SOFTWARE REQUIREMENTS

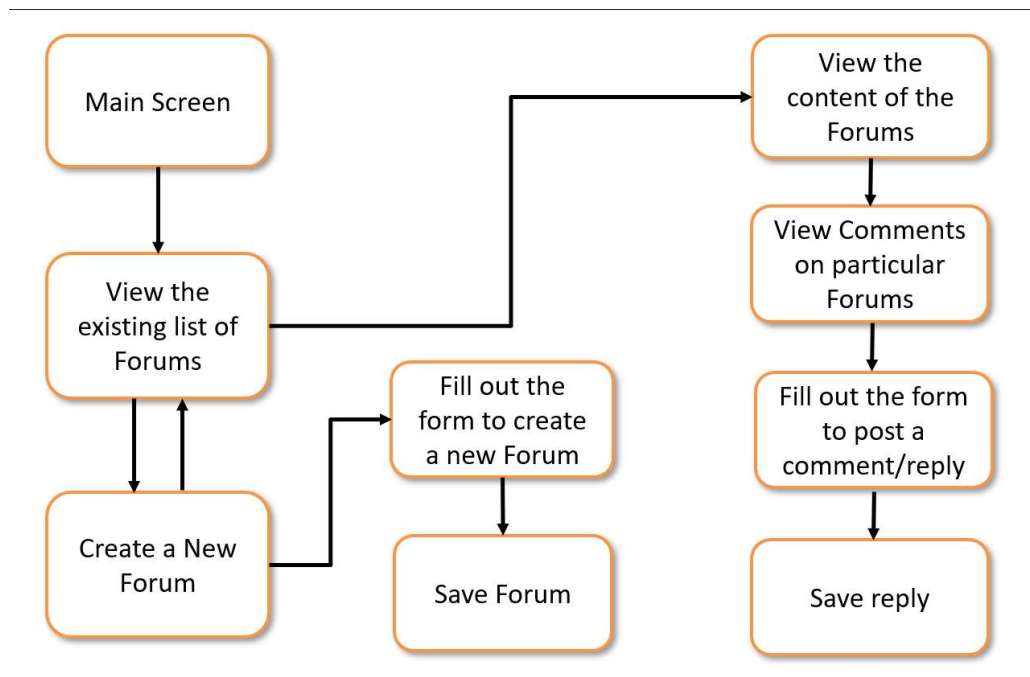
- **Operating system** : Windows10
- **Front end** : HTML, CSS, Django
- **Backend** : MYSQL, Python
- **Text-Editor** : Visual Studio Community

#### 2.3. FUNCTIONAL REQUIREMENTS

- **Mouse**
- **Keyboard**
- **Monitor**

## CHAPTER 3

### ARCHITECTURE



## CHAPTER 4

### PYTHON FEATURES

#### 4.1 FEATURES

- **Broad Standard Library:**

Python has huge collection of defined libraries which makes very easy to code in python. Its library is portable and compatible with all platforms like Macintosh, UNIX, and Windows. You don't have to write your own code for each and every thing as it provides rich sets of modules and functions. It has various libraries for web browsing, regular expressions, etc.

- **Interpreted Language:**

Python is one of the Interpreted Language as its code is executed line by line at a time. It is not required to compile our code like in other languages like java, c++, etc. which makes it easier to debug our code. The source code of python is converted into an immediate form called byte code.

- **Supports for GUI Programming:**

Python provides various modules like PyQt, Tkinter, wxPython through which user can create Graphical User Interface for mobile applications. The most popular for creating graphical apps using python is PyQt5. Tkinter also provides all of the required options to create a beautiful user interface even Gaffer is made importing Tkinter module. This user interface can be connected to the backend using any one of the DBMS also supported by python, makes it more beautiful.

- **Object Oriented Programming Language:**

Python is an object-oriented programming language which include the concept of class and object. It supports all OOPs concepts like inheritance, data abstraction, polymorphism, encapsulation etc.

- **Scalable and Extendable:**

Python provides a better structure and support for large programs than shell scripting. You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

## 4.2 Characteristics of Python

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## 4.3 Applications of Python

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

## CHAPTER 5

### HTML

#### 5.1 FEATURES OF HTML

- HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages.
- HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- HTML pages once loaded can be made available offline on the machine without any need of internet.
- HTML5 has native support for rich experience and is now useful in gaming M arena as well.
- A typical HTML document will have the following structure –

```
<!DOCTYPE html>
<html>

    <head>
        <title> Title here </title>
    </head>

    <body>
        Web page content goes here.
    </body>

</html>
```

#### 5.2 Applications of HTML:

As mentioned before, HTML is one of the most widely used language over the web. I'm going to list few of them here:

- **Web pages development** - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.

- **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- **Offline support** HTML pages once loaded can be made available offline on the machine without any need of internet.
- **Game development**- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

### 5.3 HTML Tags:

Sl. No	TAG	DESCRIPTION
1.	<code>&lt;!DOCTYPE...&gt;</code>	This tag defines the document type and HTML version.
2.	<code>&lt;html&gt;</code>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <code>&lt;head&gt;...&lt;/head&gt;</code> and document body which is represented by <code>&lt;body&gt;...&lt;/body&gt;</code> tags.
3.	<code>&lt;head&gt;</code>	This tag represents the document's header which can keep other HTML tags like <code>&lt;title&gt;</code> , <code>&lt;link&gt;</code> etc.
4.	<code>&lt;title&gt;</code>	The <code>&lt;title&gt;</code> tag is used inside the <code>&lt;head&gt;</code> tag to mention the document title.
5.	<code>&lt;body&gt;</code>	This tag represents the document's body which keeps other HTML tags like <code>&lt;h1&gt;</code> , <code>&lt;div&gt;</code> , <code>&lt;p&gt;</code> etc.
6.	<code>&lt;h1&gt;</code>	This tag represents the heading.
7.	<code>&lt;p&gt;</code>	This tag represents a paragraph.

## CSS

### 5.4 FEATURES OF CSS

- **Cascading Style Sheets**, fondly referred to as **CSS**, is a simple design language intended to simplify the process of making web pages presentable.
- **CSS saves time**, you can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So, it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

## CHAPTER 6

### DJANGO FRAMEWORK

#### 6.1 DJANGO

This project has been developed with **Django**, a **python-based** free and open-source web framework which follows the model-template-view (MTV) architectural pattern.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update, delete interface that is generated dynamically through introspection and configured via admin models.

It consists of an object relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("**Model**"), a system for processing HTTP requests with a web templating system ("**View**"), and a regular expression-based URL dispatcher ("**Controller**").

Also included in the core framework are:

- A lightweight and standalone web server for development and testing
- A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming
- A caching framework that can use any of several cache methods
- Support for middleware classes that can intervene at various stages of request processing and carry out custom functions
- An internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
- An internationalization system, including translations of Django's own components into a variety of languages
- A serialization system that can produce and read XML and/or JSON representations of Django model instances
- A system for extending the capabilities of the template engine



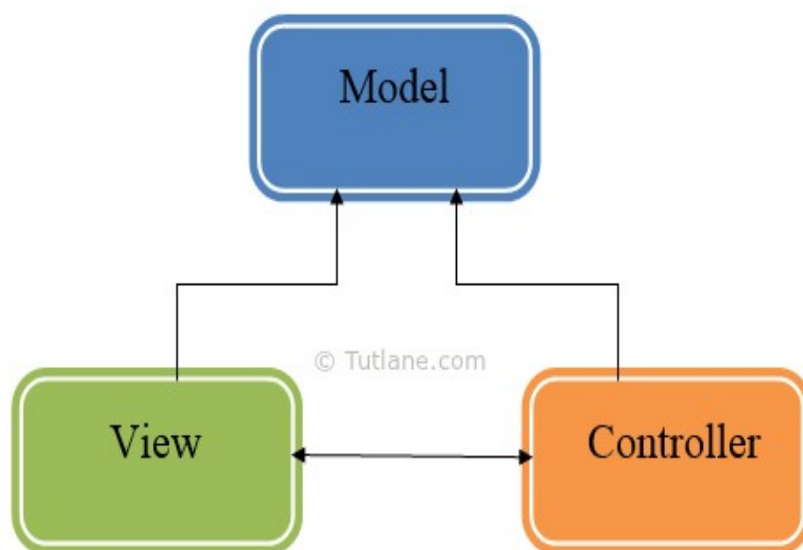
- An interface to Python's built-in unit-test framework
- Django REST framework is a powerful and flexible toolkit for building Web APIs.

The main Django distribution also bundles a number of applications in its "contrib" package, including:

- An extensible authentication system
- The dynamic administrative interface
- Tools for generating RSS and Atom syndication feeds
- A "Sites" framework that allows one Django installation to run multiple websites, each with their own content and applications
- Tools for generating Google sitemaps
- Built-in mitigation for cross-site request forgery, cross-site scripting, SQL injection, password cracking and other typical web attacks, most of them turned on by default
- A framework for creating GIS applications.

The project has been implemented with a number of functions/modules. The discussion of the flow of the project has been discussed in this chapter.

## 6.2 ARCHITECTURE



This project is implemented using MVC architecture. MVC is an acronym for Model View Control.

MVC architecture is Product Development Architecture. It solves traditional approaches' drawback of code in one file i.e., that MVC has different files for different aspect for our web application/website.

The MVC has three components namely Model, View and Controller.

This difference between components helps the developer to focus on one aspect of the web-app and therefore, better code for one functionality with better testing, debugging and scalability.

- **MODEL**

The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application's data domain. There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model.

**For example:**

When you sign up on any website you are actually sending information to the controller which then transfers it to the models which in turn applies business logic on it and stores in the database.

- **VIEW**

This component contains the UI logic in the Django architecture.

View is actually the User Interface of the web-application and contains the parts like HTML, CSS and other frontend technologies. Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

**For example:**

When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interacting with the specific components.

- **CONTROLLER**

The controller as the name suggests is the main control component. What that means is, the controller handles the user interaction and selects a view according to the model.

The main task of the controller is to select a view component according to the user interaction and also applying the model component.

This architecture has lots of advantages and that's why Django is also based on this architecture. It takes the same model to an advanced level.

**For example:**

When we combine the two previous examples, then we can very clearly see that the component which is actually selecting different views and transferring the data to the model's component is the controller.

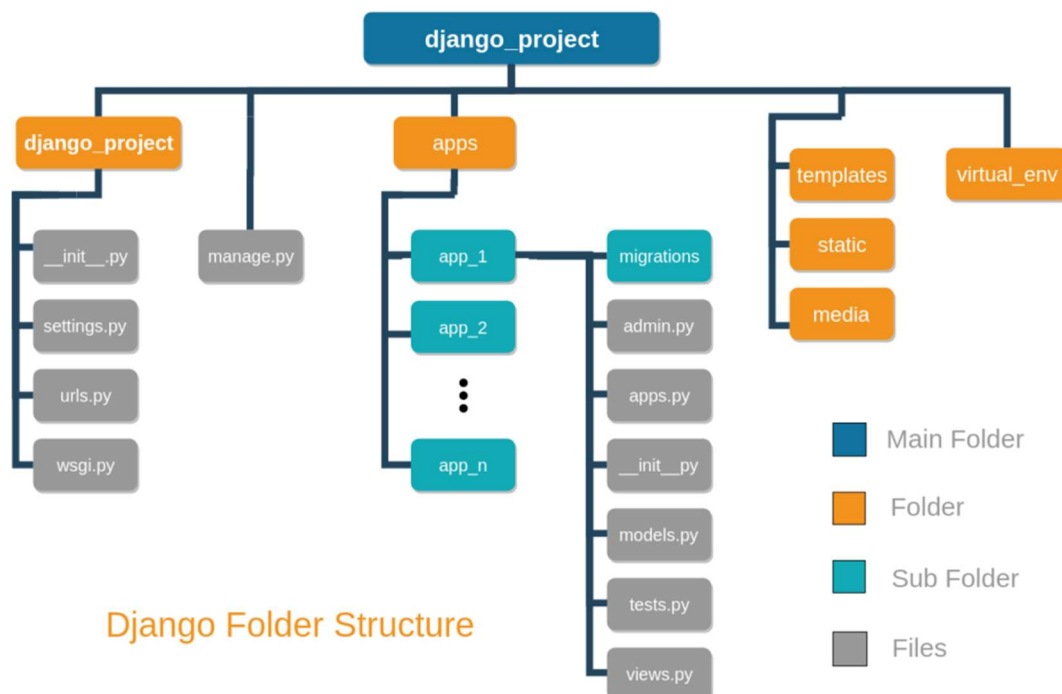
Django is mainly an MTV (Model-Template-View) framework. It uses the terminology Templates for Views and Views for Controller.

Template relates to the View in the MVC pattern as it refers to the presentation layer that manages the presentation logic in the framework and essentially controls the content to display and how to display it for the user.

Thus, our Python code will be in views and models and HTML code will be in templates.

## 6.2 DJANGO PROJECT LAYOUT AND DIFFERENT FILES STRUCTURE IN ROOT DIRECTORY

When you create a Django project, the Django framework itself creates a root directory of the project with the project name on it. That contains some files and folder, which provide the very basic functionality to your website and on that strong foundation you will be building your full scaled website.



### 6.3 FILES IN DJANGO PROJECT ROOT DIRECTORY:

By root directory, we mean about the directory which contains your manage.py file. Additional files like db.sqlite, which is a database file may be present when we will be migrating our project.

Django root directory is the default app which Django provides you. It contains the files which will be used in maintaining the whole project.

The name of Django root directory is the same as the project name you mentioned in `django-admin startproject [projectname]`. This root directory is the project's connection with Django.

The files in root directory have their specific purposes. All these files are just for completing special purposes and they are in very logical order.

We will be starting with the manage.py file.

#### 1. **manage.py**

This file is the command line utility of the project and we will be using this file only to deploy, debug and test with the project.

The file contains the code for starting the server, migrating and controlling the project through command-line.

This file provides all the functionality as with the `django-admin` and it also provides some project specific functionalities. During this tutorial, we will frequently use some of the commands that are `runserver`, `makemigrations`, `migrate` etc. We will be using these commands more frequently than others.

**runserver** is a command to start the test server provided by Django framework and that is also one of the advantages of Django over other frameworks.

**makemigrations** is the command for integrating your project with files or apps you have added in it. This command will actually check for any new additions in your project and then add that to the same.

**migrate**, the last command is to actually add those migrations you made in the last command with the whole project. You can get the idea as the former command is used for saving the changes in the file and later one to actually apply that change to the whole project then the single file.

## 2. my\_project

my\_project is the python package of the project. It has the configuration files of project settings.

- **\_\_init\_\_.py**

The \_\_init\_\_.py file is empty and it exists in the project for the sole purpose of telling the python interpreter that this directory is a package. Although we won't be doing anything in this file.

- **settings.py**

The settings.py is the main file where we will be adding all our applications and middleware applications. As the name suggests this is the main settings file of the Django project. This file contains the installed applications and middleware information which are installed on this Django project.

Every time you install a new app or custom application you will be adding that in this file.

This file should look like the image shown below. You can see that there are some pre-installed applications. These applications are by default there to provide all the basic functionality you will ever need for your website like Django admin app which we will be covering in the upcoming tutorials.

```
1  """
2  Django settings for dataflair project.
3
4  Generated by 'django-admin startproject' using Django 2.1.7.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/2.1/topics/settings/
8
9  For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/2.1/ref/settings/
11 """
12
13 import os
14
15 # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16 BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'k47fla0l)c_-ncy3e3kixifxdbjqg33jqen9*!&lro-^6sx@('
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39 ]
40
```

- **urls.py**

urls.py file contains the project level URL information. URL is universal resource locator and it provides you with the address of the resource (images, webpages, web-applications) and other resources for your website.

The main purpose of this file is to connect the web-apps with the project. Anything you will be typing in the URL bar will be processed by this urls.py file. Then, it will correspond your request to the designated app you connected to it.

```
from django.contrib import admin
from django.urls import path

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

Here this file by default adds one URL to the admin app. The path () takes two arguments.

1st is the URL to be searched in the URL bar on the local server and 2nd is the file you want to run when that URL request is matched, the admin is the pre-made application and the file is URL's file of that app. This file is the map of your Django project.

- **wsgi.py**

Django is based on python which uses WSGI server for web development. This file is mainly concerned with that and we will not be using this file much.

wsgi is still important though if you want to deploy the applications on Apache servers or any other server because Django is still backend and you will need its support with different servers. But you need not to worry because for every server there is a Django middleware out there which solves all the connectivity and integration issues and you just have to import that middleware for your server.

These files are the very basics and will be present in any **Django** application you create individually. Their main task is to provide you all the backend support and resolve connectivity issues. It happens while you handle the database creation. It also manages frontend and the uniqueness of your website or web-application.

## 6.4 IMPLEMENTATION

### 1. To create django app

The main reason we are using web applications in the first place is to implement Django's code reusability feature.

This enables us to not only migrate the pre-built apps in our project but also customize web applications made by us.

All the commands are given in our root directory or in the directory where we have the manage.py file.

Also, before creating our Django application just execute this command on your system inside the project/root directory.

***python manage.py makemigrations***

After completion of this command run this command:

***python manage.py migrate***

Execute this command to start django project:

***django-admin startapp application-name***

In our root directory we have a new directory as a news(application-name) the application we just created. This directory contains all the files that our new application may need and we will be modifying them to achieve our goals.

```
migrations
66 admin.py
88 apps.py
60 models.py
63 tests.py
66 views.py
0 __init__.py
```

These are the files which come pre-installed when your application is created and the formats of each file for each application remains the same, that's the Django Way.

## 2. Installing Django app

Once we have created our application, even though it's in the same folder as the main Django project, the application needs to be installed.

To **install django application** in the project we will first open settings.py and modify it. Remember we will be modifying the main settings.py file of our Django project.

You will just have to type your app-name in the INSTALLED\_APPS list as in the image, after adding our application in this list of Installed Apps.

```
32
33 INSTALLED_APPS = [
34     'news.apps.NewsConfig',
35     'users.apps.UsersConfig',
36     'crispy_forms',
37     'django.contrib.admin',
38     'django.contrib.auth',
39     'django.contrib.contenttypes',
40     'django.contrib.sessions',
41     'django.contrib.messages',
42     'django.contrib.staticfiles',
43 ]
44
```

## 3. Adding apps in ulrs.py file

To add the app in the ulrs.py, you will again have to write some code. We will first need to create a new python file in the news directory (app directory) and write the code shown below in that file.

```
urls.py
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('home/', views.home, name='news-home'),
6     path('', views.cybertech, name='news-cybertech'),
7     path('programming/', views.programming, name='news-programming'),
8     path('datascience/', views.datasciencenews, name='news-datascience'),
9     path('database/', views.db, name='news-database'),
10    path('security/', views.sec, name='news-security'),
11    path('unix/', views.unixx, name='news-unix'),
12    path('blockchain/', views.Blockchain, name='news-blockchain'),
13    path('virtualreality/', views.VR, name='news-virtualreality'),
14 ]
```

Here in the code, we are just telling our Django project that you have to initiate this function in the views.py file. Here we have imported the django. urls package and path function from



there. We have passed 2 arguments here, first, the URL which was searched and was passed to the URL bar from the browser, the other argument is to execute the file or function which is index function in our case.

Now, we will be modifying the urls.py file in our main Django project.

There is only one item in the list and you will just have to add the new data.

```
16 from django.contrib import admin
17 from django.contrib.auth import views as auth_views
18 from django.urls import path,include
19 from django.conf import settings
20 from django.conf.urls.static import static
21 from users import views as user_views
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
```

After adding the apps urls.

```
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.urls import path,include
from django.conf import settings
from django.conf.urls.static import static
from users import views as user_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('register/', user_views.register,name='register'),
    path('profile/', user_views.profile,name='profile'),
    path('login/', auth_views.LoginView.as_view(template_name='users/login.html'),name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.html'),name='logout'),
    path('password-reset/', auth_views.PasswordResetView.as_view(template_name='users/password_reset.html'),name='password_reset'),
    path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(template_name='users/password_reset_done.html'),name='password_reset_done'),
    path('password-reset-confirm/<uidb64>/<token>', auth_views.PasswordResetConfirmView.as_view(template_name='users/password_reset_confirm.html'),name='password_reset_confirm'),
    path('', include('news.urls')),
]
```

By doing this you are telling the system to direct your server to check the URL for demo keyword.

After that, direct the URL to the urls.py file inside demo application directory.

#### 4. Creating a views.py file

Finally, we will be creating the views file of our Django project. This file contains the logic which is to be displayed to the user and will make the view on the browser.

Here we write the code to scraping all the news. We import packages like **beautiful soup**, **requests**, **urllib3**, **re** required for web scraping.

We also import models.py file in views.py file because all the news that are scraped are stored in database/ models created in models.py file.

## 5. Creating models.py file

In this file we write the code to create database tables as classes. An example of news model is shown below.

```
1 from django.db import models
2 from django.utils import timezone
3 from django.contrib.auth.models import User
4
5
6 class news(models.Model):
7     author=models.CharField(max_length=100)
8     title=models.CharField(max_length=1000)
9     content=models.TextField()
10    link=models.URLField(max_length=200)
11    date_posted=models.DateTimeField(default=timezone.now)
12
13    def __str__(self):
14        return self.title
15
```

Here we import **model** function from **django. db**, **timezone** function from **django. utils** and **User** function from **django. contrib. auth. models**.

In this example we can see that a table called **news** is created and attributes like **author**, **title**, **content**, **link**, **date\_posted** are created with their data types.

*A Model in Django is a class imported from the **django. db** library that acts as the bridge between your database and server.* This class is a representation of the data structure used by your website. It will directly relate this data-structure with the database. **Django ORM**, a tool that provides you with the ability to write **python data structures** instead of a database language.

When you create a model, Django executes SQL to design a corresponding table in the database (as shown below) without the need to write even a single line of SQL. Django prefixes the name of the table with your Django application name. Also, the model links related information in the database.

The model links related information in the database.

Model news:

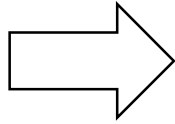
author

title

content

Link

date\_posted



ID	1002
author	Corey m s
title	Python tutorial
content	How to create a django project...
link	<a href="http://coreyms/python/django">http://coreyms/python/django</a>
Date_posted	20/03/2019

Now we have to execute the following commands:

***python manage.py makemigrations***

***python manage.py migrate***

## 6. Creating templates

*Django templates are a combination of static HTML layout and Django Syntax which is essentially **python code**. Both of these components help in generating HTML pages dynamically and make your website more user-engaging.*

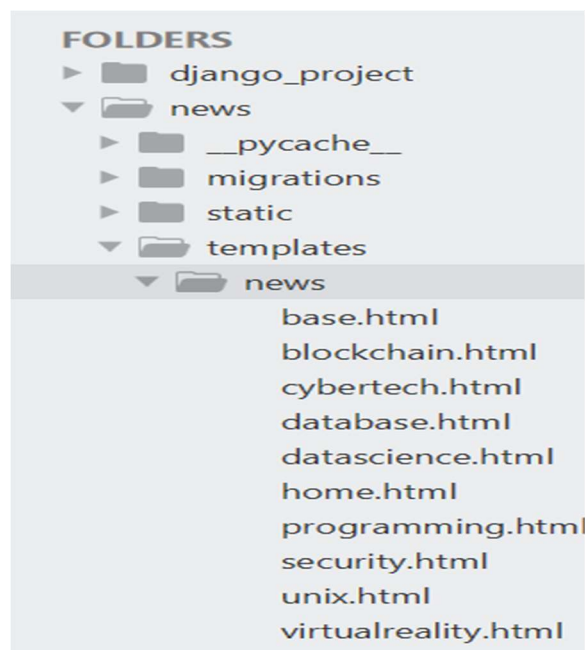
The **main purpose of Django templates is to separate the data representation with the data itself**. That means, the browser part is only to render the HTML sent to it by the server, and all the relevant data is given to the template by Django itself. This makes the process much easier and pages render easily as there is less clutter in both the front-end and back-end.

Django templating is done via templating engines, there are multiple templating engines (Jinja templating) although the one which Django ships in is Django template as you can see in your projects' settings.py file.

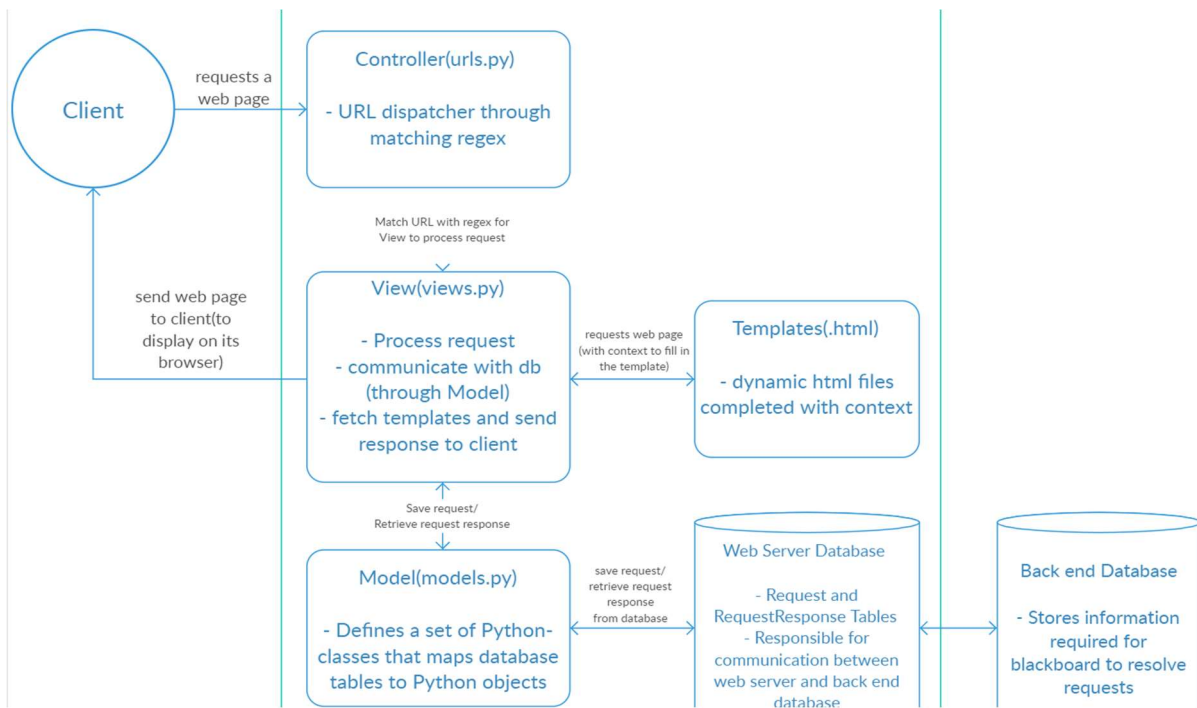
```
57 TEMPLATES = [  
58     {  
59         'BACKEND': 'django.template.backends.django.DjangoTemplates',  
60         'DIRS': [],  
61         'APP_DIRS': True,  
62         'OPTIONS': {  
63             'context_processors': [  
64                 'django.template.context_processors.debug',  
65                 'django.template.context_processors.request',  
66                 'django.contrib.auth.context_processors.auth',  
67                 'django.contrib.messages.context_processors.messages',  
68             ],  
69         },  
70     },  
71 ]  
72
```

### Create the Django template rendered in views.py file

Make a new directory named news same as the application name, and inside that create one HTML file named cybertech.html which we also passed in the render function.



## 6.5 DJANGO FLOWCHART:



## CHAPTER 7

### BOOTSTRAP

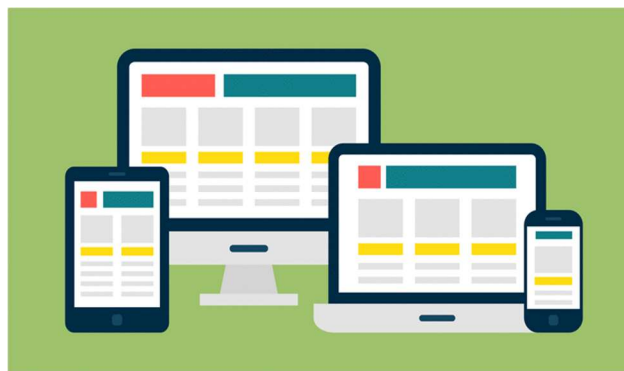
Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and JavaScript.

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter. It was released as an open-source product in August 2011 on GitHub.

- **Mobile first approach** – Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
- **Browser Support** – It is supported by all popular browsers.



- **Easy to get started** – With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also, the Bootstrap official site has a good documentation.
- **Responsive design** – Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles. More about the responsive design is in the chapter [Bootstrap Responsive Design](#).



- Provides a clean and uniform solution for building an interface for developers.
- It contains beautiful and functional built-in components which are easy to customize.
- It also provides web-based customization.
- And best of all it is an open source.

#### Bootstrap Package Includes:

- **Scaffolding** – Bootstrap provides a basic structure with Grid System, link styles, and background. This is covered in detail in the section **Bootstrap Basic Structure**
- **CSS** – Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section **Bootstrap with CSS**.
- **Components** – Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section **Layout Components**.
- **JavaScript Plugins** – Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section **Bootstrap Plugins**.
- **Customize** – You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

**File Structure:**

Once the compiled version Bootstrap is downloaded, extract the ZIP file, and you will see the following file/directory structure –

```
bootstrap/  
├── css/  
│   ├── bootstrap.css  
│   ├── bootstrap.min.css  
│   ├── bootstrap-theme.css  
│   └── bootstrap-theme.min.css  
├── js/  
│   ├── bootstrap.js  
│   └── bootstrap.min.js  
└── fonts/  
    ├── glyphicons-halflings-regular.eot  
    ├── glyphicons-halflings-regular.svg  
    ├── glyphicons-halflings-regular.ttf  
    └── glyphicons-halflings-regular.woff
```

**HTML TEMPLATE:**

A basic HTML template using Bootstrap would look like this –

```
<!DOCTYPE html>
<html>

<head>
  <title>Bootstrap 101 Template</title>
  <meta name = "viewport" content = "width = device-width, initial-scale = 1.0">

  <!-- Bootstrap -->
  <link href = "css/bootstrap.min.css" rel = "stylesheet">

  <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

  <!--[if lt IE 9]>
  <script src = "https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
  <script src = "https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
  <![endif]-->

</head>

<body>
  <h1>Hello, world!</h1>

  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
  <script src = "https://code.jquery.com/jquery.js"></script>

  <!-- Include all compiled plugins (below), or include individual files as needed -->
  <script src = "js/bootstrap.min.js"></script>

</body>
</html>
```



## CHAPTER 8

### IMPLEMENTATION

#### 1) DataFlair\_discsnForum:

##### (a) asgi.py

```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'DataFlair_discsnForum.settings')

application = get_asgi_application()
```

**ASGI**, or the Asynchronous Server Gateway Interface, is the specification which Channels and Daphne are built upon, designed to untie Channels apps from a specific application server and provide a common way to write application and middleware code.

##### (b) Setting.py

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

TEMP_DIR = os.path.join(BASE_DIR, 'templates')

SECRET_KEY = '5&iar0*n*4cg-l%-73xei7r-qaofw$0h!1g7+2i@mt!^*9+9h$'

DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',
```

```
'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'Discussion_forum',

]

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',

]

ROOT_URLCONF = 'DataFlair_discsnForum.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [TEMP_DIR],

'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',

],

},

],
```

```
    },  
]  
  
WSGI_APPLICATION = 'DataFlair_discsnForum.wsgi.application'  
  
# Database  
  
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases  
  
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
    }  
}  
  
# Password validation  
  
AUTH_PASSWORD_VALIDATORS = [  
    {  
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',  
    },  
]  
  
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [

    os.path.join(TEMP_DIR, 'DataFlair_discsnForum/templates')

]
```

**(c) urls.py**

```
from django.contrib import admin

from django.urls import path

from Discussion_forum.views import *

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', home, name='home'),

    path('addInForum/', addInForum, name='addInForum'),

    path('addInDiscussion/', addInDiscussion, name='addInDiscussion'),

]
```

**(d) wsgi.py**

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'DataFlair_discsnForum.settings')
```

```
application = get_wsgi_application()
```

**WSGI** is a specification, laid out in PEP 333, for a standardized interface between Web servers and **Python** Web frameworks/applications. The goal is to provide a relatively simple yet comprehensive interface capable of supporting all (or most) interactions between a Web server and a Web framework

## 2) Discussion\_forum:

### (a) admin.py

```
from django.contrib import admin
```

```
from .models import *
```

```
# Register your models here.
```

```
admin.site.register(forum)
```

```
admin.site.register(Discussion)
```

### (b) apps.py

```
from django.apps import AppConfig
```

```
class DiscussionForumConfig(AppConfig):
```

```
    name = 'Discussion_forum'
```

The purpose of apps.py file is **Application** configuration objects store metadata for an **application**. Some attributes can be configured in AppConfig subclasses.

**(c) models.py**

```
from django.db import models
```

```
#parent model
```

```
class forum(models.Model):
```

```
    name=models.CharField(max_length=200,default="anonymous" )
```

```
    email=models.CharField(max_length=200,null=True)
```

```
    topic= models.CharField(max_length=300)
```

```
    description = models.CharField(max_length=1000,blank=True)
```

```
    link = models.CharField(max_length=100 ,null =True)
```

```
    date_created=models.DateTimeField(auto_now_add=True,null=True)
```

```
    def __str__(self):
```

```
        return str(self.topic)
```

```
#child model
```

```
class Discussion(models.Model):
```

```
    forum = models.ForeignKey(forum,blank=True,on_delete=models.CASCADE)
```

```
    discuss = models.CharField(max_length=1000)
```

```
    def __str__(self):
```

```
        return str(self.forum)
```

A **model** is a class that represents table or collection in our database, and where every attribute of the class is a field of the table or collection. **Models** are defined in the app/**models.py**

**(d) views.py**

```
from django.shortcuts import render, redirect
```

```
from .models import *
```

```
from .forms import *
```

```
# Create your views here.
```

```
def home(request):
```

```
    forums=forum.objects.all()
```

```
    count=forums.count()
```

```
    discussions=[]
```

```
    for i in forums:
```

```
        discussions.append(i.discussion_set.all())
```

```
    context={'forums':forums,
```

```
            'count':count,
```

```
            'discussions':discussions}
```

```
    return render(request,'home.html',context)
```

```
def addInForum(request):
```

```
    form = CreateInForum()
```

```
    if request.method == 'POST':
```

```
        form = CreateInForum(request.POST)
```

```
        if form.is_valid():
```

```
        form.save()

        return redirect('/')

    context = {'form': form}

    return render(request, 'addInForum.html', context)


def addInDiscussion(request):

    form = CreateInDiscussion()

    if request.method == 'POST':

        form = CreateInDiscussion(request.POST)

        if form.is_valid():

            form.save()

            return redirect('/')

    context = {'form': form}

    return render(request, 'addInDiscussion.html', context)
```

**(e) forms.py**

```
from django.forms import ModelForm

from .models import *
```

```
class CreateInForum(ModelForm):

    class Meta:

        model= forum

        fields = "__all__"


class CreateInDiscussion(ModelForm):

    class Meta:

        model= Discussion
```



---

```
fields = "__all__"
```

### 3) templates:

#### (a) home.html

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>DataFlair Discussion Forum</title>

    <link                                                    rel="stylesheet"
        href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
        9alt2nRpC12Uk9gS9baDI411NQApFmC26EwAOH8WgZl5MYxXfFc+NcPb1dKGj7Sk"
        crossorigin="anonymous">

    <style>

        .box{

            border: 4px solid black;

            margin: 0 auto;

        }

    </style>

</head>

<body id="bg" style="background-image: url('{% static 'Discussion_forum/images/a1.jpg'
    %}');">

<h1 style="font-family:Viner Hand ITC; text-align:center; color:DarkSlateGrey; background-
    color:white;">

    <b>Discussion Forum</b>

</h1>

<h2 class="jumbotron" style="font-family:Javanese Text; background-image: url('{% static 'Discussion\_forum/images/plant.jpg' %}');" >

Currently active forums: {{count}}

<form method="POST" action="{% url 'addInForum' %}">

{% csrf\_token %}

<button class="btn btn-success" title="Add another Forum" style="width:fit-content; padding: 4px; margin:10px;font-family:Poor Richard;color:black;">Click here to add another Forum</button>

</form>

</h2>

<div class="card-columns" style="padding: 10px; margin: 20px;"></div>

{%for forum in forums %}

<div class="card box container" style="font-family:HP Simplified; background-image:url('{% static 'Discussion\_forum/images/c.jpg' %}');">

<br>

<h5 class="card-title">

<a href='{{forum.link}}' style="color:Navy; font-family:Trebuchet MS; font-weight:bold;"><h3> Forum name: {{forum.topic}}</h3></a>

<div class="card-body container">

<p>{{forum.description}}</p>

</h5>

<hr>

<p> By : {{forum.name}}</p>

Email-id : {{forum.email}}

```
<hr>

<h4 style="font-family:Trebuchet MS; color:Navy"><u>Comments</u></h4>

{%for discuss in discussions%}

{%for objs in discuss%}

{% if objs.forum == forum %}

    {{objs.discuss}}

    <br>

{% endif %}

{%endfor%}

{%endfor%}

<form method="POST" action="{% url 'addInDiscussion' %}">

    {% csrf_token %}

    <button class="btn btn-success" style="width:fit-content; padding: 4px;
margin:10px;color:black;">Add                                a comment</button>

</form>

</div>

</div>

</div>

<br>

{%endfor%}


<script    src="https://code.jquery.com/jquery-3.5.1.slim.min.js"    integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

<script    src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvblyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
```

```
<script      src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
  integrity="sha384-
  OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
  crossorigin="anonymous"></script>

</body>

</html>
```

**(b) addInForum.html**

```
<head>

  <style>

    form{

      border:4px solid black;

      margin: 0 auto;

      padding: 40px;

      width: fit-content;

    }

  </style>

  <link  rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
    integrity="sha384-
    9alt2nRpC12Uk9gS9baDI411NQApFmC26EwAOH8WgZl5MYxXfFc+NcPb1dKGj7Sk"
    crossorigin="anonymous">

</head>

<form action="{% url 'addInForum' %}" method="POST">

  {% csrf_token %}

  {{form.as_p}}

  <input type="submit" class="btn btn-success" value="submit">

</form>
```

**(c) addInDiscussion.html**

```
<head>

<style>

    form{

        border:4px solid black;

        margin: 0 auto;

        padding: 40px;

        width: fit-content;

    }

</style>

<link                                rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
    integrity="sha384-
    9alt2nRpC12Uk9gS9baDI411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
    crossorigin="anonymous">

</head>

<form action="{% url 'addInDiscussion' %}" method="POST">

    {% csrf_token %}

    {{form.as_p}}

    <input type="submit" class="btn btn-success" value="submit">

</form>
```

**4) Images should be saved as per the following address location:**

C:\Users\anish\Projects\Discussion\_Forum\Discussion\_forum\static\Discussion\_forum\images



**DF\_logo.jpg**



**c.jpg**

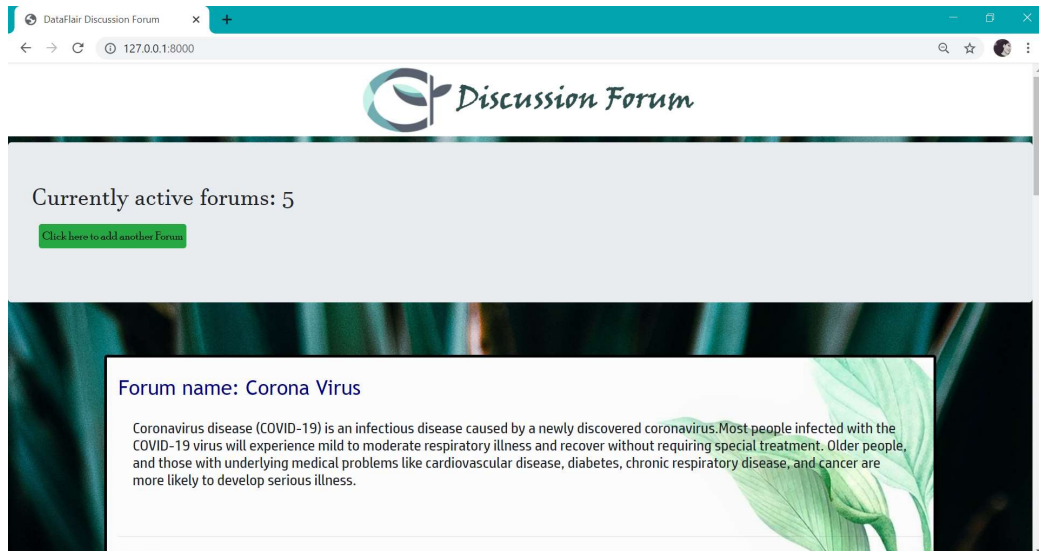


**a1.jpg**

## CHAPTER 9

### OUTPUT SNAPSHOTS

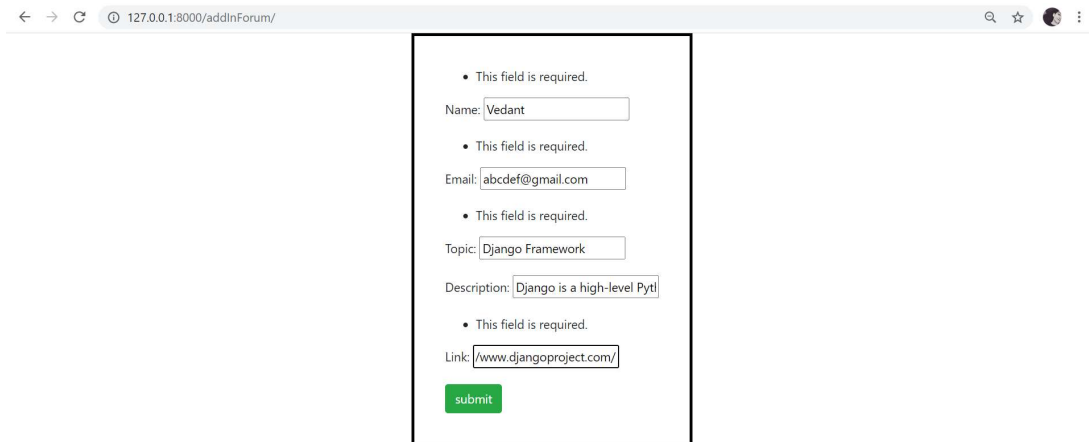
- 1) Main Page: It shows the number of active forums



- 2) When you click on “Click here to add another Forum” button

A screenshot of a web browser showing the '127.0.0.1:8000/addInForum/' page. The browser's address bar displays '127.0.0.1:8000/addInForum/'. The page contains a form with five input fields, each preceded by a red asterisk and the text 'This field is required.' The fields are labeled 'Name:', 'Email:', 'Topic:', 'Description:', and 'Link:'. Below the fields is a green button labeled 'submit'.

## 3) Fill in the details



← → ↻ 127.0.0.1:8000/addInForum/ 🔍 ☆ 👤 ⋮

- This field is required.

Name:

- This field is required.

Email:

- This field is required.

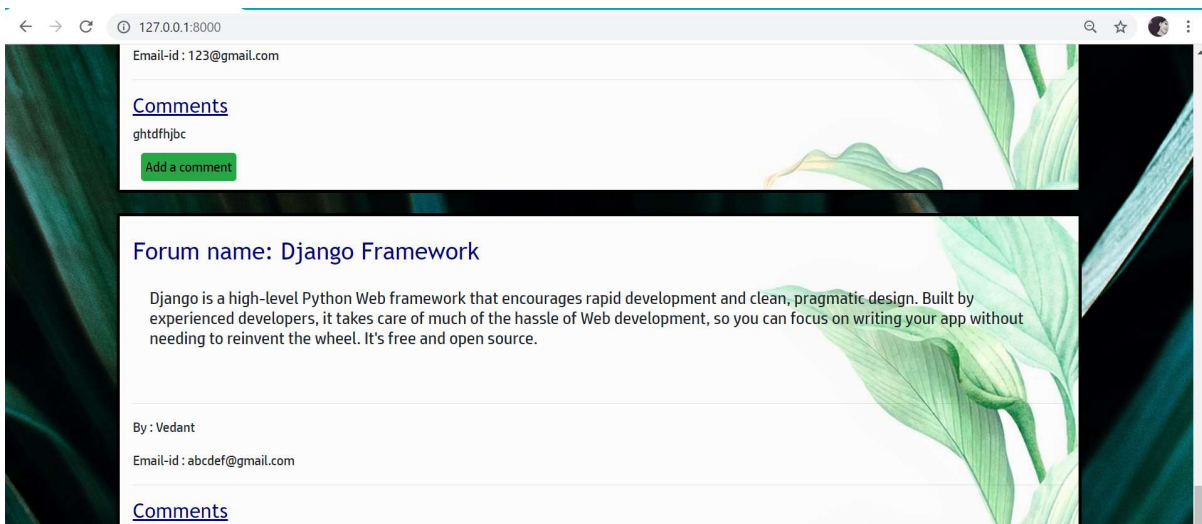
Topic:

Description:

- This field is required.

Link:

## 4) A new Forum is created



← → ↻ 127.0.0.1:8000 🔍 ☆ 👤 ⋮

Email-id : 123@gmail.com

[Comments](#)

ghtdfhjbc

---

**Forum name: Django Framework**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

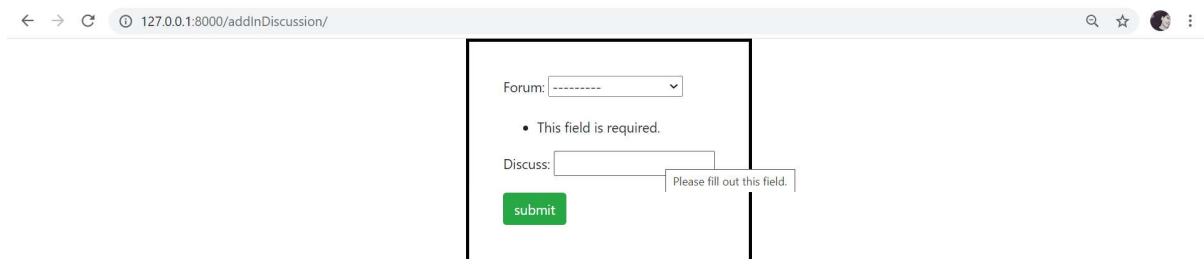
By : Vedant

Email-id : abcdef@gmail.com

[Comments](#)

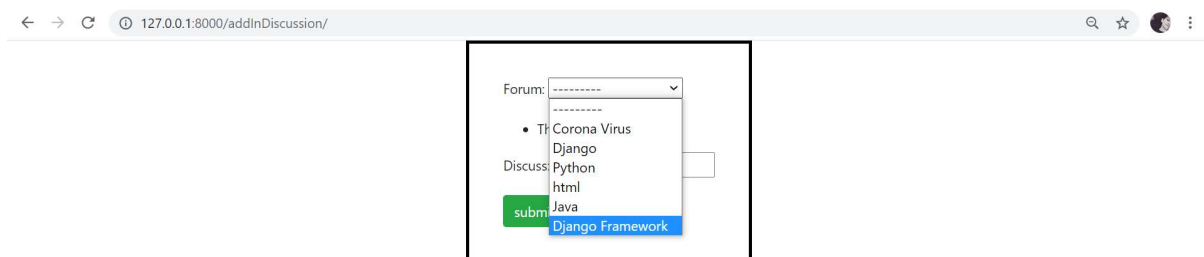


## 5) When you click on “Add a comment” button



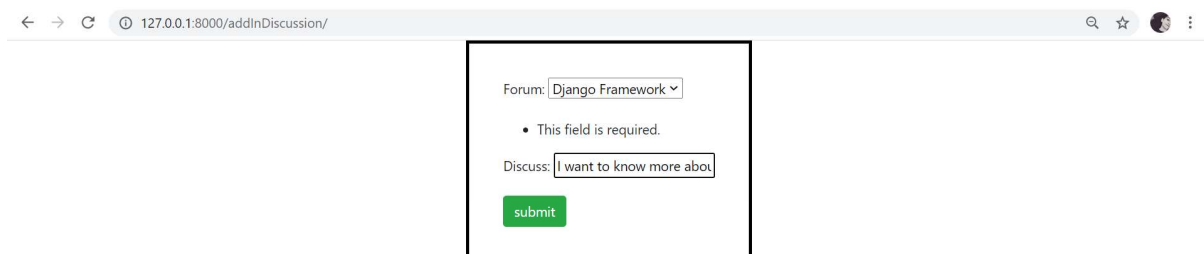
A screenshot of a web browser showing the URL `127.0.0.1:8000/addInDiscussion/`. The browser's address bar includes navigation icons and search, star, and user profile icons. The main content area displays a form with a 'Forum:' dropdown menu, a text input field for 'Discuss:', and a green 'submit' button. A red error message, '• This field is required.', is shown below the 'Forum:' dropdown. A tooltip with the text 'Please fill out this field.' is visible next to the 'Discuss:' input field.

## 6) Choose the Forum you want to comment on



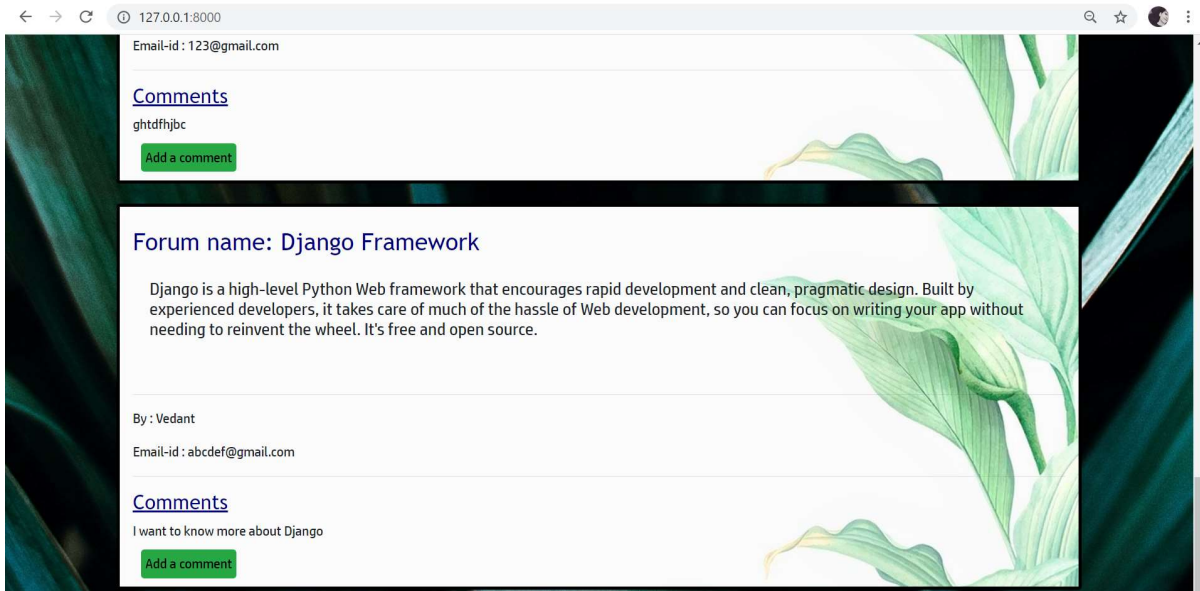
A screenshot of the same web browser showing the URL `127.0.0.1:8000/addInDiscussion/`. The 'Forum:' dropdown menu is open, displaying a list of forum topics: 'Corona Virus', 'Django', 'Python', 'html', 'Java', and 'Django Framework'. The 'Django Framework' option is highlighted in blue. The 'Discuss:' input field and the green 'submit' button are also visible.

## 7) Fill in the details



A screenshot of the same web browser showing the URL `127.0.0.1:8000/addInDiscussion/`. The 'Forum:' dropdown menu is now set to 'Django Framework'. The 'Discuss:' input field contains the text 'I want to know more about'. The green 'submit' button is visible at the bottom of the form.

8) Your comment will be added to the Forum you've chosen



## CHAPTER 10

### CONCLUSION

- The purpose of conducting this study and doing this project is to understand the significance web applications using html, CSS and Django Framework.
- Easily accessible, error-free and reliable system
- Efficient and less time consuming.

### REFERENCES

- <https://www.youtube.com/>
- <https://www.w3schools.com/>
- <https://www.geeksforgeeks.org/>