

## **2CSDE86 Application Development Frameworks**

### **Lab-1 Task**

**Submitted by: Labdhi Sheth 18BCE101**

#### **Aim:**

- 1. Comparative study of MVC and MVT architectures**
- 2. Installation of python, vscode, and configuring the basic Django project.**

## Part 1

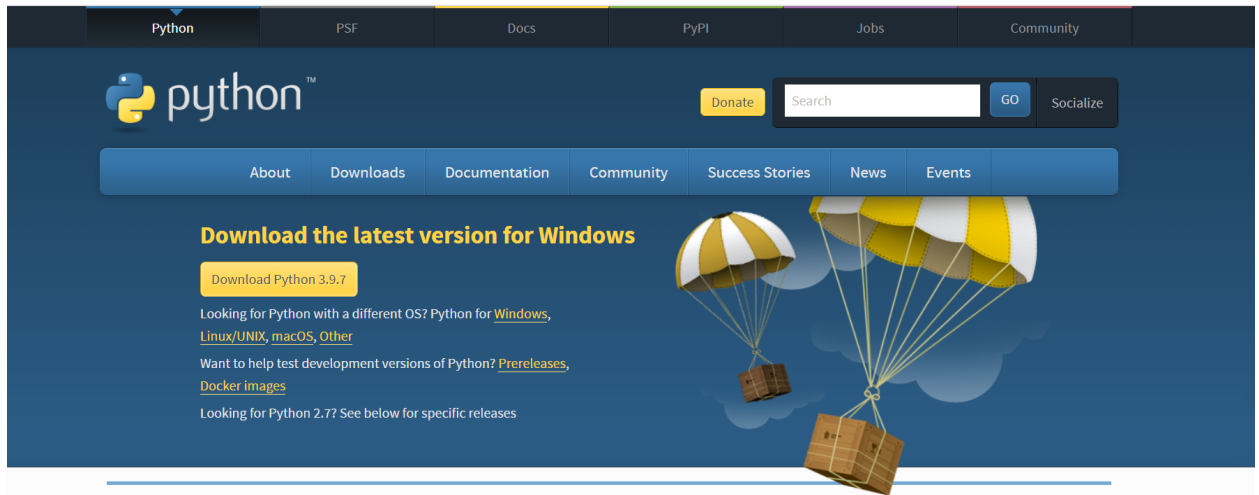
Features	MVC	MVT
<b>Abbreviation</b>	Model View Controller	Model View Template
<b>Role of each</b>	<p>Model: encapsulates the access to the database layers and adds, deletes, changes, and checks the data in the database</p> <p>View: encapsulates the results, generates HTML content displayed on the page</p> <p>Controller: receives requests, processes business logic, interacts with Model and View, and returns results.</p>	<p>Model: interacts with the database for data processing.</p> <p>View: receives requests, performs business processing, and returns a response.</p> <p>Template: responsible for encapsulating the HTML to be returned</p>
<b>Framework type</b>	Common design framework	Django framework
<b>Coupling</b>	Highly coupled	Loosely coupled
<b>Ease of modifying</b>	Difficult	Easy
<b>Applications type</b>	Suitable for the development of large applications but not for small applications.	Suitable for both small and large applications.
<b>Approach</b>	HTTP request is sent to the controller which then either tells the Model to make changes and update the View or returns View based on a Model. This view is controlled by Model and Controller.	HTTP request is sent to the View which then sends the query to the Model and collects the result set. The view then fills the result in template and sends the HTTP response to the user
<b>Code</b>	Here one has to write all the control specific code	The framework takes care of that.
<b>Mapping of URLs</b>	It does not involve the mapping of URLs.	URL pattern mapping takes place.
<b>Features</b>	<ul style="list-style-type: none"> <li>• Easy and frictionless testability, highly testable, extensible, and pluggable framework.</li> <li>• Supports for Test Driven Development (TDD)</li> </ul>	<ul style="list-style-type: none"> <li>• When a user makes an HTTP request, the corresponding view performs a query on the Model and collects the result set from the Model. The View then fills the result in a template and sends it to the user.</li> </ul>

<b>Flow</b>	Flow is easy to understand.	Flow is sometimes hard to understand.
<b>Visualization of flow</b>	<pre> graph TD     Model["Model Defines data structure e.g. updates application to reflect added item"]     View["View Defines display (UI) e.g. user clicks 'add to cart'"]     Controller["Controller Contains control logic e.g. receives update from view then notifies model to 'add item'"]     Model -- "Updates e.g. let item to show added item" --&gt; View     View -- "Sends input from user" --&gt; Controller     Controller -- "Manipulates" --&gt; Model     View -- "Sometimes updates directly" --&gt; Model </pre>	<pre> graph TD     Model["Model Object Relational Mapping (ORM)"]     View["View Business Logic"]     Template["Template Display Logic"]     Model -- "Datasets" --&gt; View     View -- "User Input" --&gt; Template     View -- "Data for Display" --&gt; Template     View -- "Create Update Delete" --&gt; Model </pre>
<b>Pros</b>	<ul style="list-style-type: none"> <li>• Easy code maintenance which is easy to extend and grow</li> <li>• MVC Model component can be tested separately from the user</li> <li>• Easier support for new types of clients</li> <li>• Development of the various components can be performed parallelly.</li> <li>• It helps you to avoid complexity by dividing an application into three units. Model, view, and controller.</li> </ul>	<ul style="list-style-type: none"> <li>• Less coupled.</li> <li>• Suitable for small to large-scale applications.</li> <li>• Easy to Modify.</li> <li>• In Django, more emphasis is placed on explicit programming rather than implicit programming, making it one of the ideal frameworks for applications that require rapid changes.</li> </ul>
<b>Cons</b>	<ul style="list-style-type: none"> <li>• Difficult to read, change, unit test, and reuse this model</li> <li>• The framework navigation can some time complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.</li> <li>• No formal validation support</li> <li>• Increased complexity and Inefficiency of data</li> <li>• The difficulty of using MVC with the modern user interface</li> </ul>	<ul style="list-style-type: none"> <li>• Sometimes, understanding the flow can be confusing.</li> <li>• Modification of Models / Views should be done carefully without affecting Templates</li> <li>• Unlike most web development frameworks, Django can't handle multiple requests simultaneously.</li> </ul>
<b>Examples</b>	ASP.NET MVC, Spring MVC, etc...	Django
<b>Web frameworks</b>	Ruby on rails, Catalyst, Laravel	Django

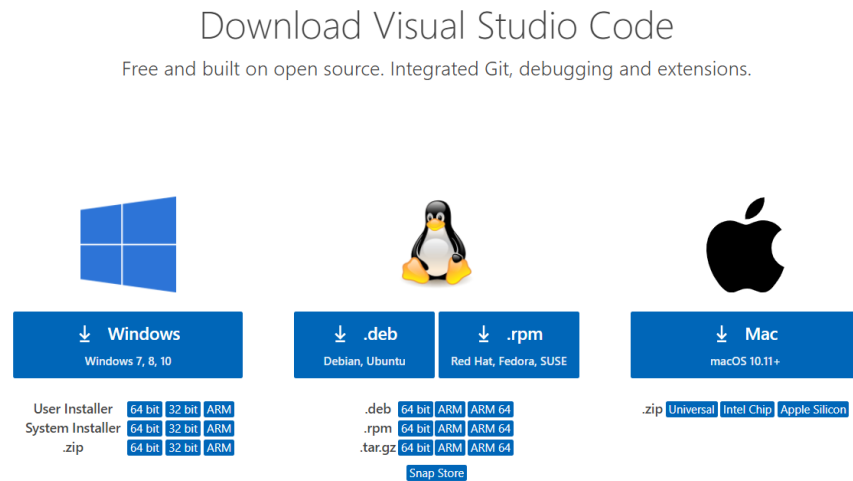
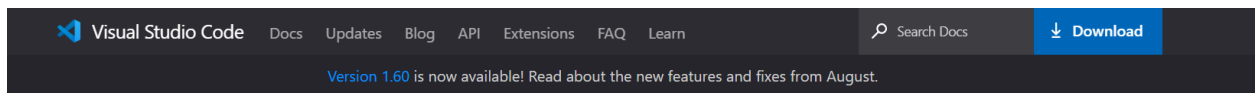
## Part 2

Installation of python, vscode, and configuring basic Django project.

### 1. Installation of python:



### 2. Vscode installation:



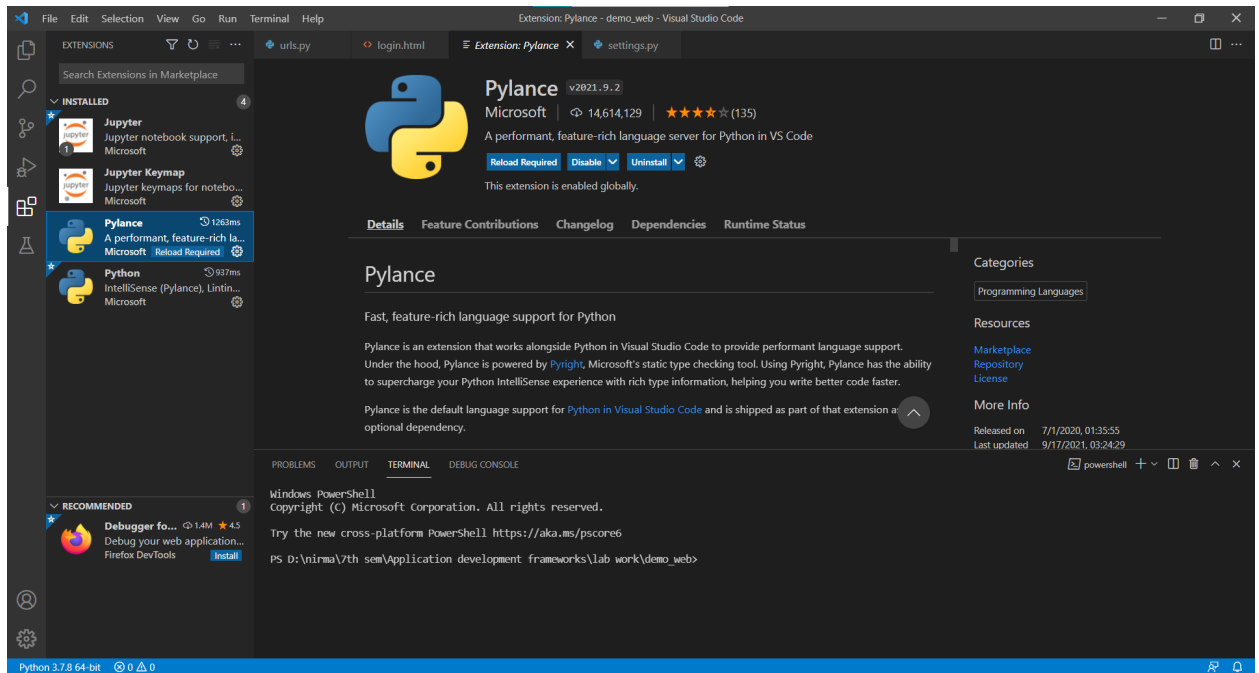
### 3. Run python pip install Django as:

```
python -m pip install Django
```

### 4. Create a new django project as:

```
django-admin startproject django_demo
```

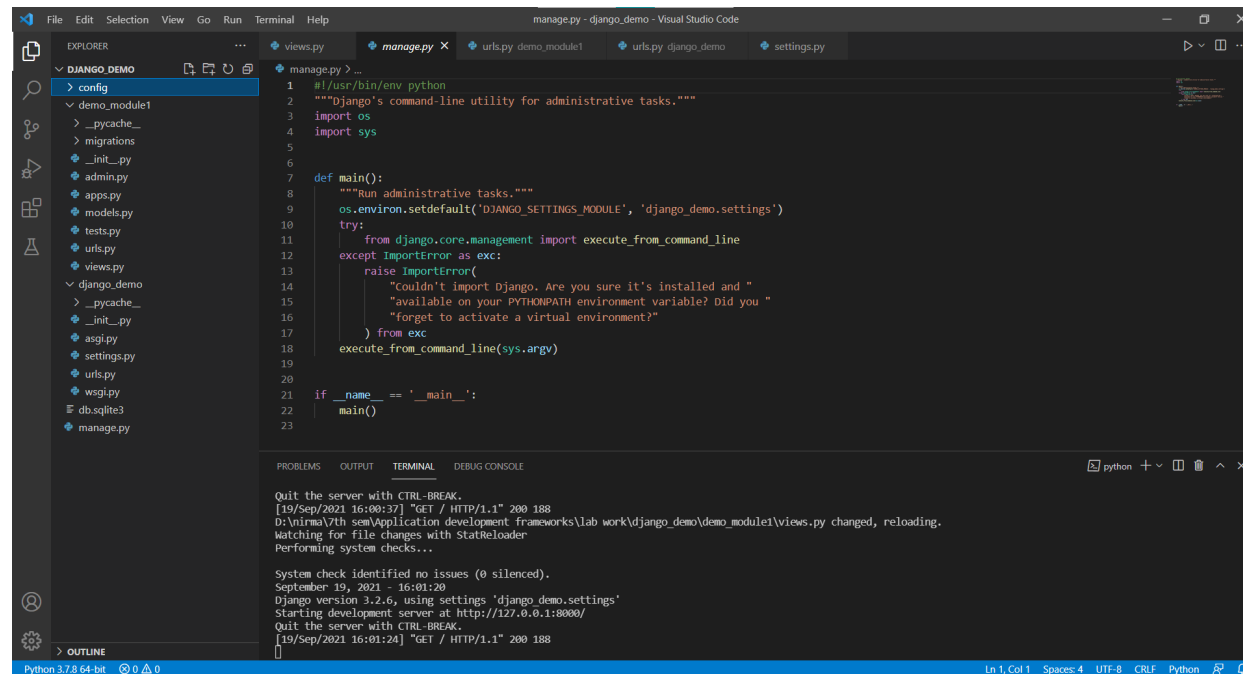
### 5. Install python and pylance extension in vscode.



## 6. Files in the django django\_demo project:

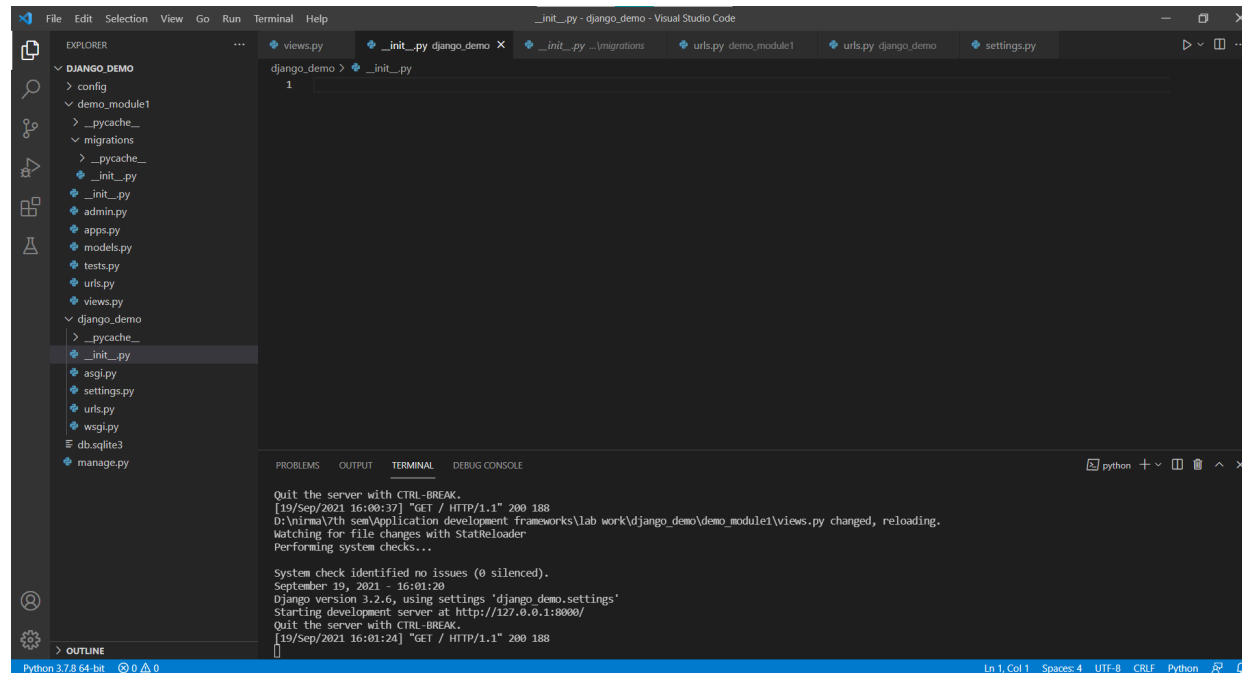
### a. manage.py:

This file is responsible for starting the sever, migrating and controlling. It provides some project specific functionalities.



### b. \_\_init\_\_.py:

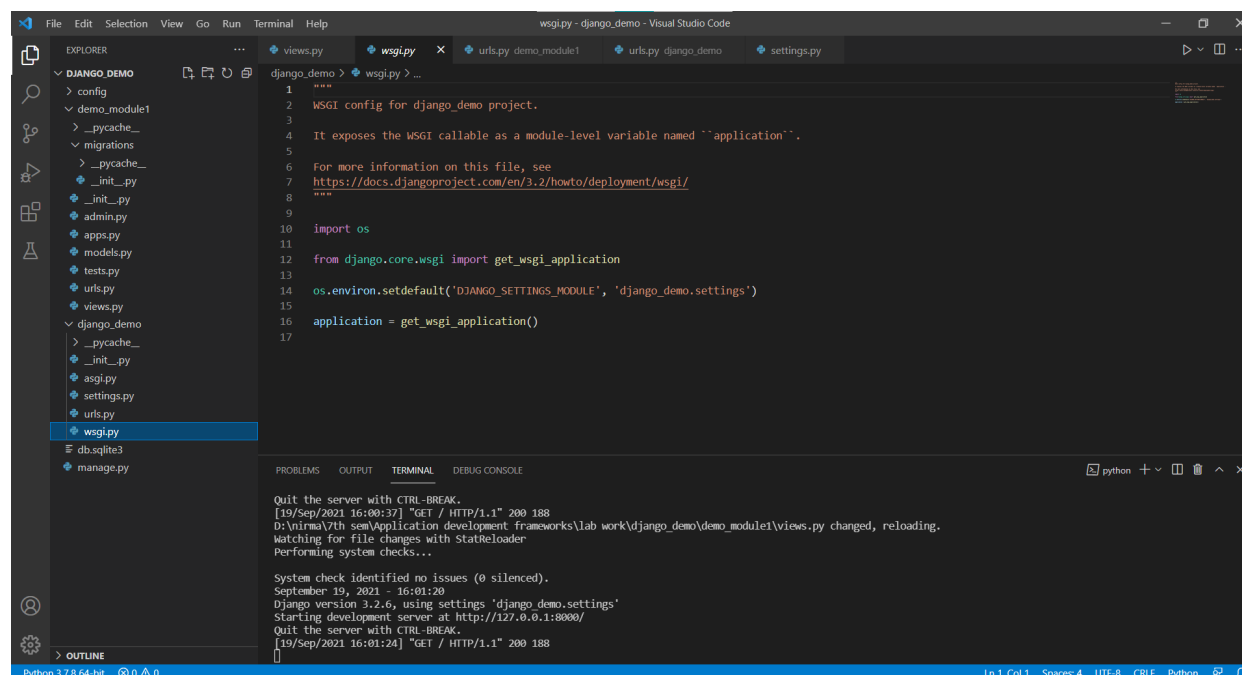
It is a constructor type of file which is always empty. It tells that this directory is a package.



The screenshot shows the Visual Studio Code interface with the Django demo project. The Explorer pane on the left shows the project structure, including the `django_demo` directory. The `__init__.py` file is highlighted. The main editor shows the content of `__init__.py`, which is empty. The terminal at the bottom shows the output of the Django development server, including the message "Quit the server with CTRL-BREAK." and the URL "http://127.0.0.1:8000/".

c. `wsgi.py`:

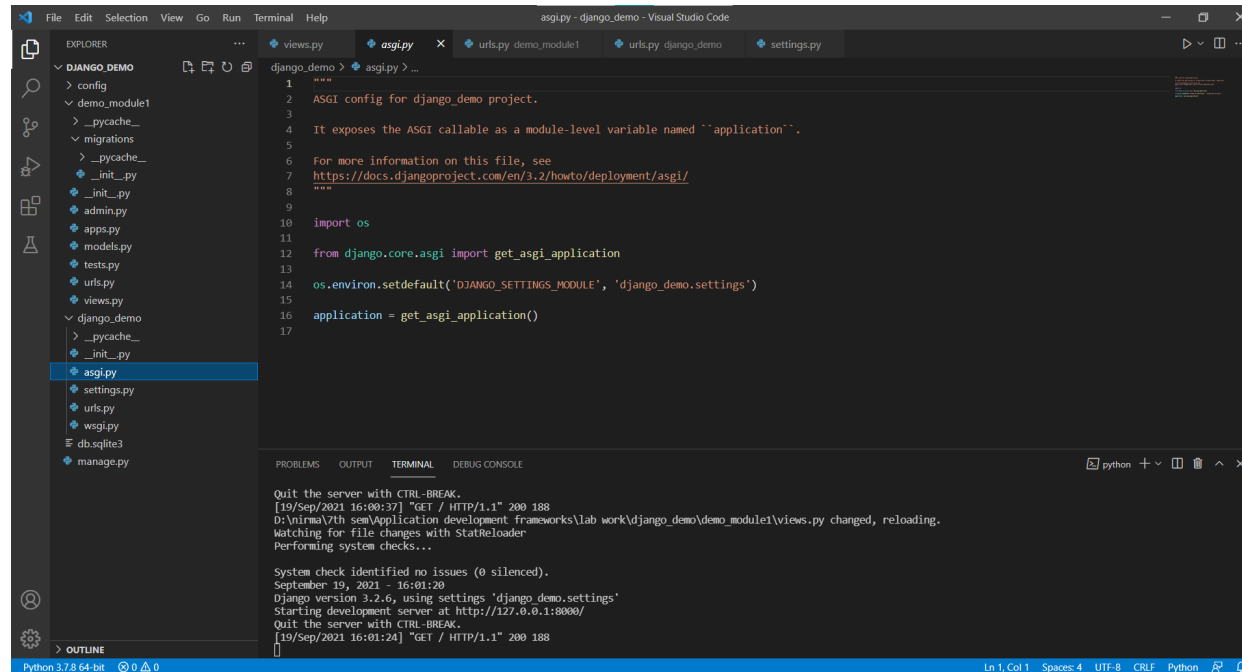
This files is responsible for deploying applications on different servers. It describes the way of iteration of the server with the application and thus is the mediator between server and application communication. It stands for web server interface.



The screenshot shows the Visual Studio Code interface with the Django demo project. The Explorer pane on the left shows the project structure, including the `django_demo` directory. The `wsgi.py` file is highlighted. The main editor shows the content of `wsgi.py`, which is a Django WSGI application. The terminal at the bottom shows the output of the Django development server, including the message "Quit the server with CTRL-BREAK." and the URL "http://127.0.0.1:8000/".

d. asgi.py:

This file is similar to wsgi but has better functionalities. It stands for asynchronous server gateway interface.



The screenshot shows the Visual Studio Code editor with the 'asgi.py' file open. The file contains the following code:

```
1 """
2 ASGI config for django_demo project.
3
4 It exposes the ASGI callable as a module-level variable named ``application``.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/3.2/howto/deployment/asgi/
8 """
9
10 import os
11
12 from django.core.asgi import get_asgi_application
13
14 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'django_demo.settings')
15
16 application = get_asgi_application()
17
```

The terminal output shows the following messages:

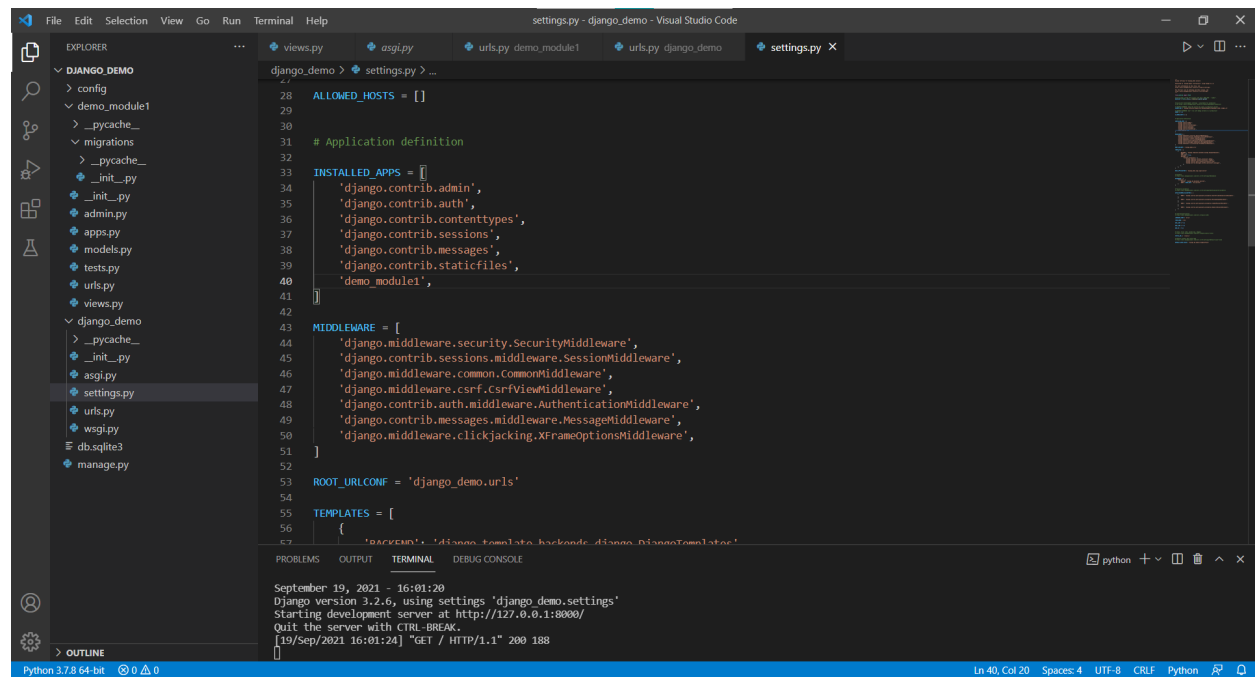
```
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:00:37] "GET / HTTP/1.1" 200 188
D:\nirma\7th sem\Application development frameworks\lab work\django_demo\demo_module1\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188
```

e. settings.py

It is a important file and th main settings file. It contains list of inbuilt and custom middleware and installed apps. Middlewares are used which server application communication so as to transfer the flow to views. In this project the module demo\_module1 was added to the INSTALLED\_APPS in settings.py. It is created as

python manage.py startapp demo\_module1



```
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
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     'demo_module1',
40 ]
41
42 MIDDLEWARE = [
43     'django.middleware.security.SecurityMiddleware',
44     'django.contrib.sessions.middleware.SessionMiddleware',
45     'django.middleware.common.CommonMiddleware',
46     'django.middleware.csrf.CsrfViewMiddleware',
47     'django.contrib.auth.middleware.AuthenticationMiddleware',
48     'django.contrib.messages.middleware.MessageMiddleware',
49     'django.middleware.clickjacking.XFrameOptionsMiddleware',
50 ]
51
52 ROOT_URLCONF = 'django_demo.urls'
53
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': [],
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
64                 'django.contrib.messages.context_processors.messages',
65             ],
66         },
67     },
68 ]
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

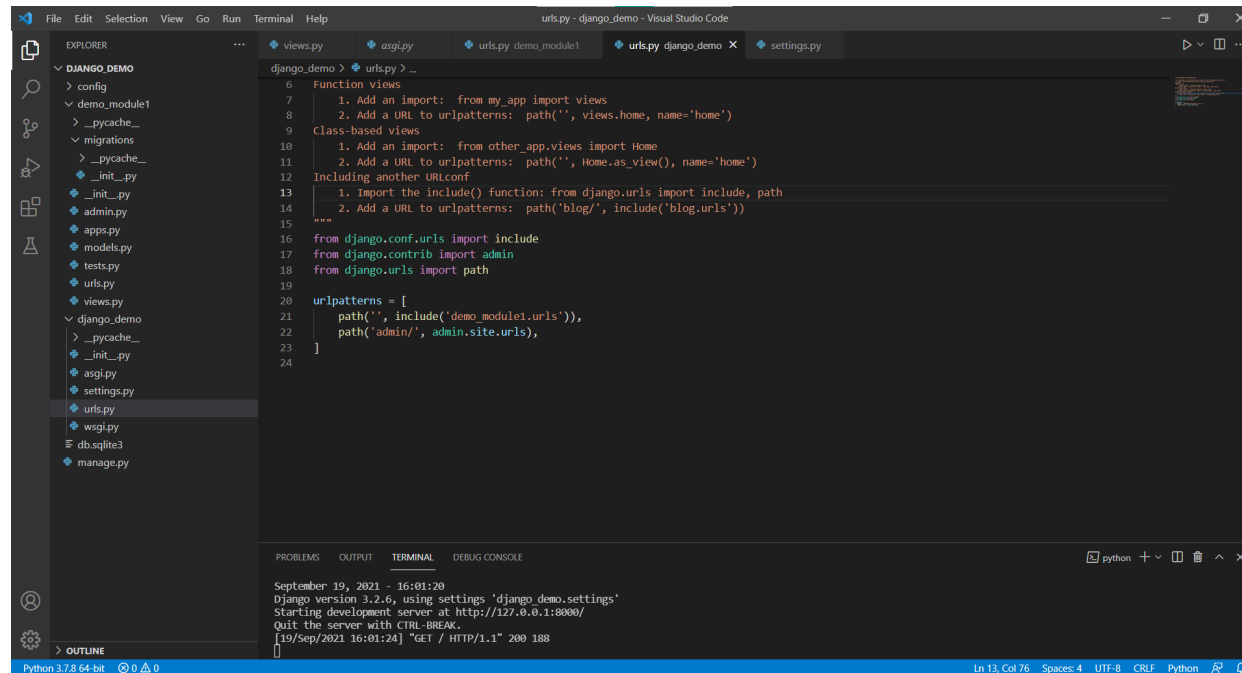
September 19, 2021 - 16:01:20  
Django version 3.2.6, using settings 'django\_demo.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.  
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188

f. urls.py:

This file denotes the project level urls. It provides the address of resources that are present in the web pages. In simpler words this file tells Django that if a user comes with xyz url then direct them to the particular website or any other kind of response.



urls.py for django\_demo: it denotes that the default page with show the output of demo\_module1.



```
File Edit Selection View Go Run Terminal Help
urls.py - django_demo - Visual Studio Code

EXPLORER
DJANGO_DEMO
  config
  demo_module1
  migrations
  __pycache__
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  urls.py
  views.py
  django_demo
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py

django_demo > urls.py > ...
6 Function views
7     1. Add an import: from my_app import views
8     2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 from django.conf.urls import include
17 from django.contrib import admin
18 from django.urls import path
19
20 urlpatterns = [
21     path('', include('demo_module1.urls')),
22     path('admin/', admin.site.urls),
23 ]
24

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188

Python 3.7.8 64-bit In 13, Col 76 Spaces: 4 UTF-8 CRLF Python
```

## 7. Files in the demo\_module1:

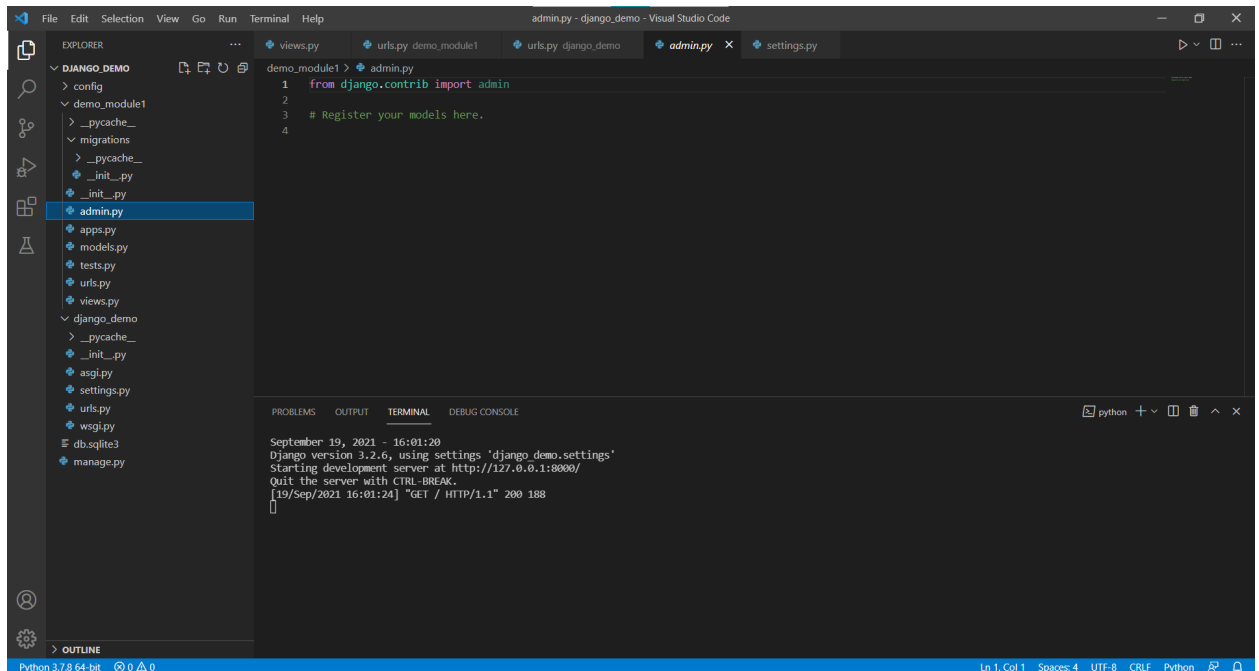
In 6(e) point I have mentioned how has the application been made under the project. The files under the same are:

### a. \_\_init\_\_.py

The same empty file with the same purpose

### b. admin.py

Used for registering the django models into django administration. It performs three major tasks as registering models, creating a superuser and login in and using the web application. It is empty in this practical.



The screenshot shows the Visual Studio Code editor with the Django project structure in the Explorer. The `admin.py` file is selected in the `demo_module1` folder. The code in `admin.py` is:

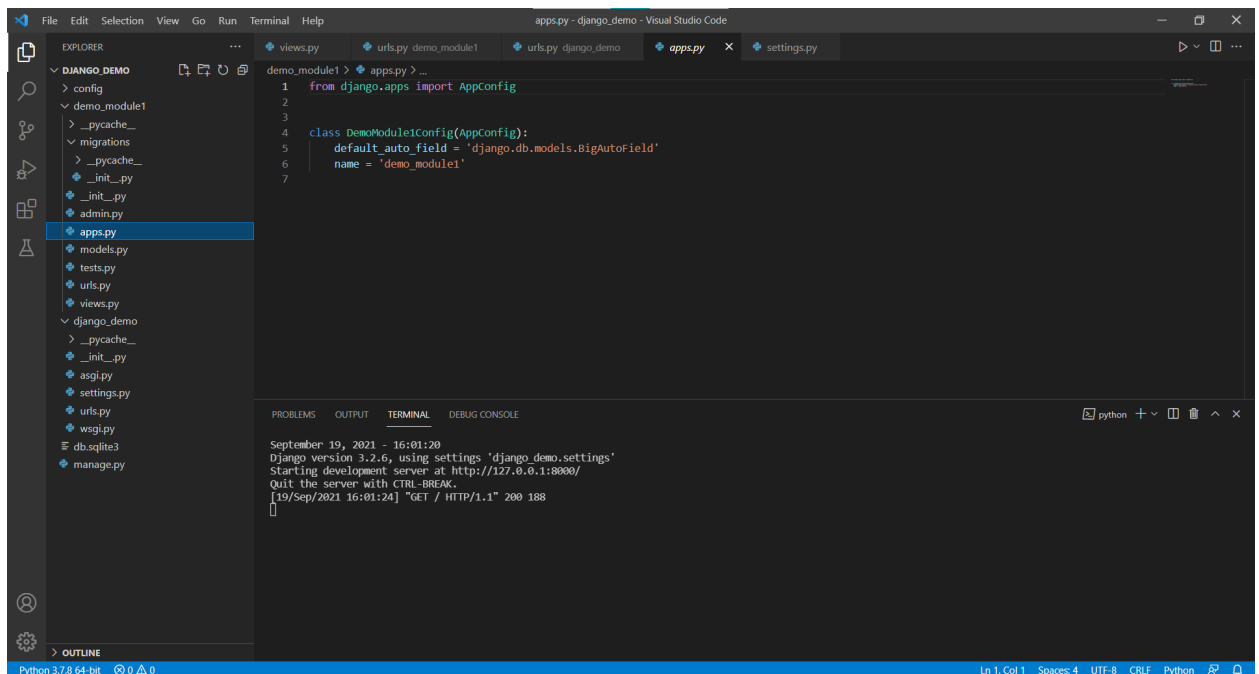
```
1 from django.contrib import admin
2
3 # Register your models here.
4
```

The terminal at the bottom shows the Django development server running:

```
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188
```

### c. `apps.py`

Used to help the user include the application configuration for their app.



The screenshot shows the Visual Studio Code editor with the Django project structure in the Explorer. The `apps.py` file is selected in the `demo_module1` folder. The code in `apps.py` is:

```
1 from django.apps import AppConfig
2
3
4 class DemoModule1Config(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'demo_module1'
7
```

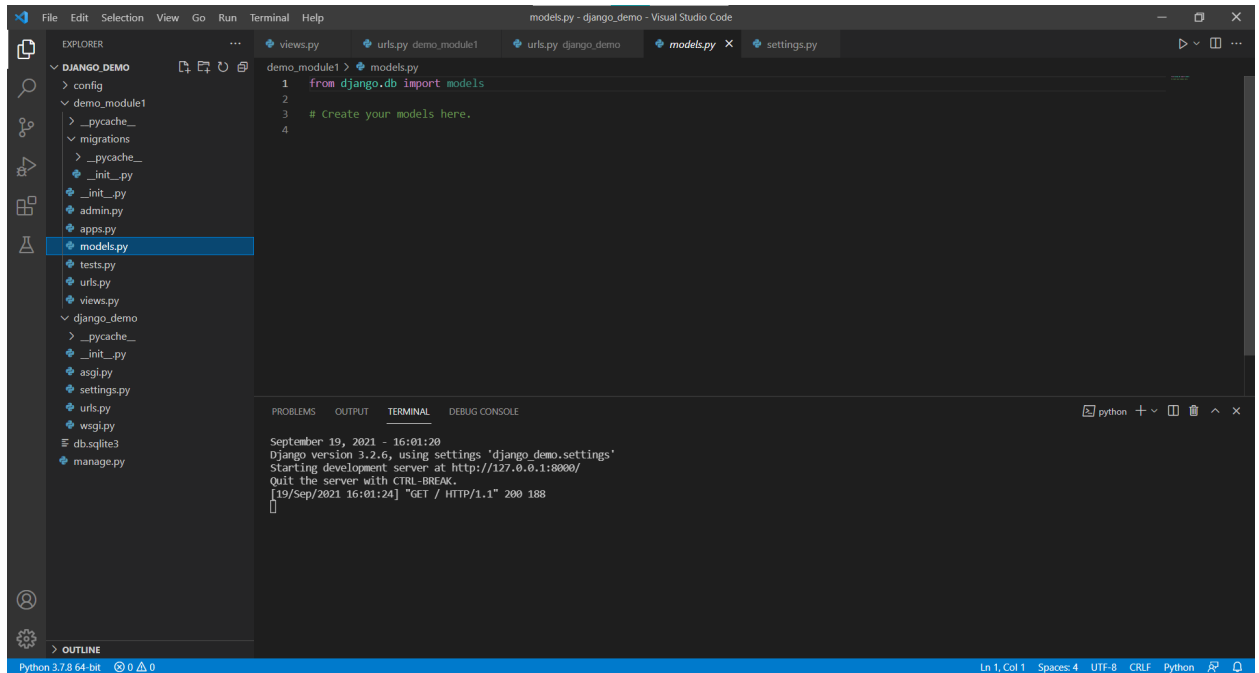
The terminal at the bottom shows the Django development server running:

```
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188
```

### d. `models.py`

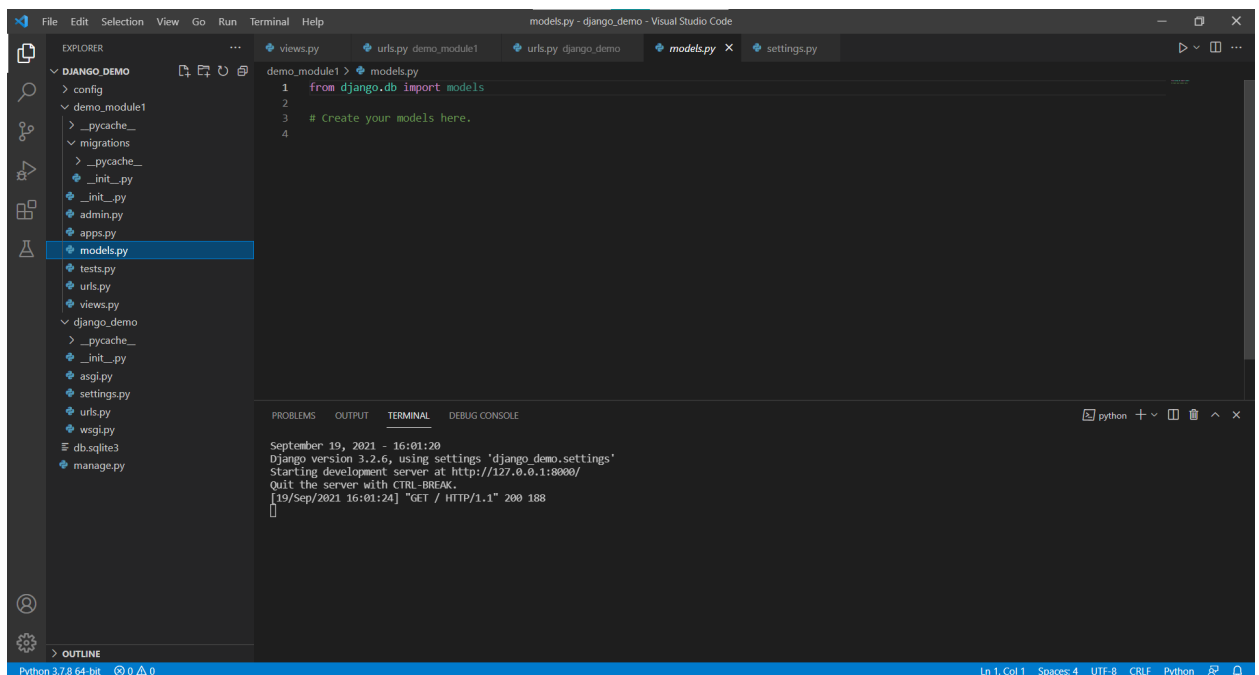
This represents the models of web applications in the form of classes. It is a part of app structure. It defines the structure of the database and tesla about

actual design, relationships between data sets.



e. `test.py`

This file is used when some test cases are to be implemented.



f. `urls.py`

It has the same purpose as stated before, here we are calling methods from `views.py` file.

```
demo_module1 > urls.py > ...
1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = []
6     path('', views.index, name='index'),
7 
```

```
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188
```

## g. views.py

This file tells about django app structure and provides a interface through which a user interacts with a django web application. It contains all the views in form of classes and functions. Class views are preferred because of their oop functionality. This files gives .html as the response.

```
demo_module1 > views.py > index
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5 def index(request):
6     html_string = <html>
7     <body>
8         <center>
9             <h1>18BCCE101</h1>
10            <h2>Practical 1</h2>
11            <h3>Generating a basic homepage</h3>
12        </center>
13    </body>
14    </html>...
15    return HttpResponse(html_string)
```

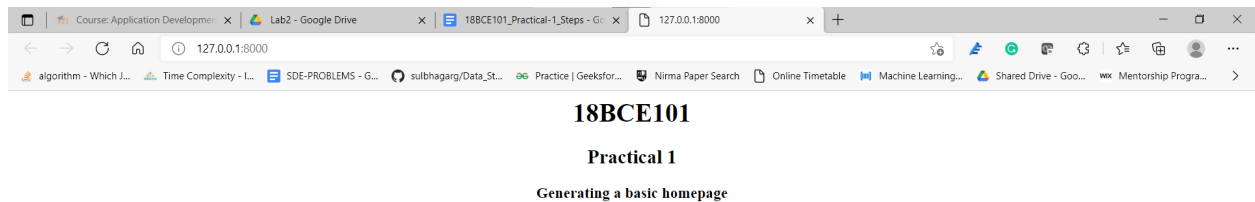
```
September 19, 2021 - 16:01:20
Django version 3.2.6, using settings 'django_demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[19/Sep/2021 16:01:24] "GET / HTTP/1.1" 200 188
```

## 8. Running the output:

Open a new terminal window to write the following commands:

```
python manage.py makemigrations (to set the changes in schema)
python manage.py migrate          (to create a sqlite database)
python manage.py runserver        (to start server)
```

Click on the link and the output shown will be :



## Conclusion:

This practical introduced to django framework and how MVC and MVT differs from each other. We even got an idea about the project and app structure and the importance of each files. A simple project was made which gave an html response back to the server.