2CSDE86 Application Development Frameworks

Lab-1 Task

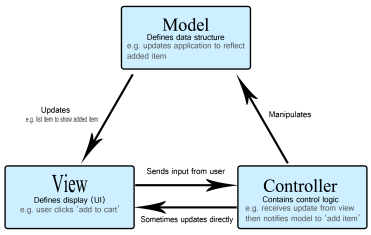
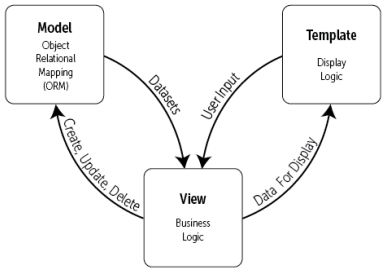
Submitted by: Labdhi Sheth 18BCE101

<u>Aim:</u>

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

Part 1

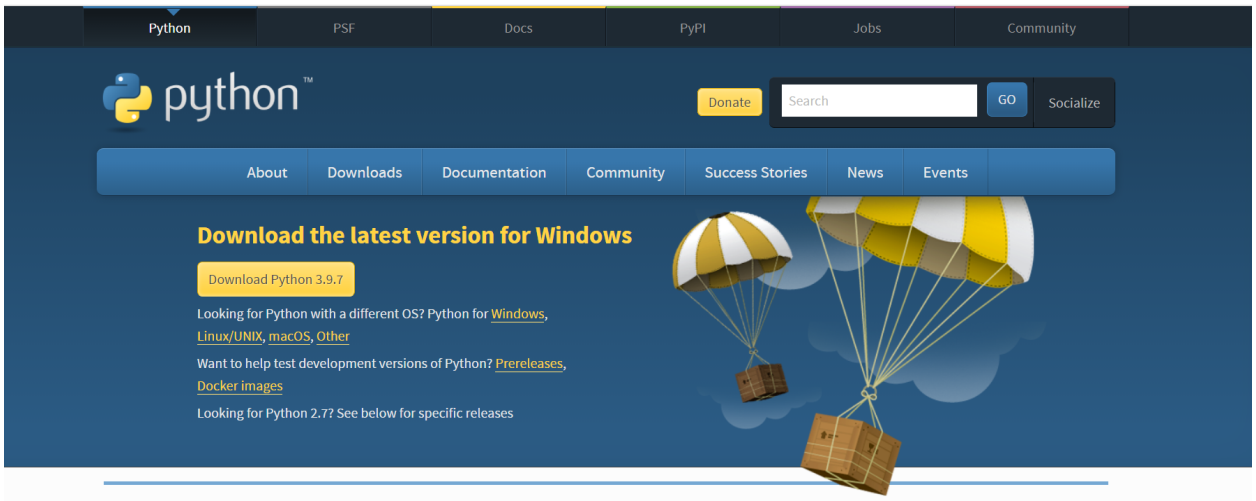
Features	MVC	MVT
Abbreviation	Model View Controller	Model View Template
Role of each	Model: encapsulates the access to the database layers and adds, deletes, changes, and checks the data in the database View: encapsulates the results, generates HTML content displayed on the page Controller: receives requests, processes business logic, interacts with Model and View, and returns results.	Model: interacts with the database for data processing. View: receives requests, performs business processing, and returns a response. Template: responsible for encapsulating the HTML to be returned
Framework type	Common design framework	Django framework
Coupling	Highly coupled	Loosely coupled
Ease of modifying	Difficult	Easy
Applications type	Suitable for the development of large applications but not for small applications.	Suitable for both small and large applications.
Approach	HTTP request is sent the 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 send 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
Code	Here one has to write all the control specific code	The framework takes care of that.
Mapping of URLs	It does not involve the mapping of URLs.	URL pattern mapping takes place.
Features	• Easy and frictionless testability, highly testable, extensible, and pluggable framework. • Supports for Test Driven Development (TDD)	• 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.

Flow	Flow is easy to understand.	Flow is sometimes hard to understand.
Visualization of flow		
Pros	● Easy code maintenance which is easy to extend and grow ● MVC Model component can be tested separately from the user ● Easier support for new types of clients ● Development of the various components can be performed parallelly. ● It helps you to avoid complexity by dividing an application into three units. Model, view, and controller.	● Less coupled. ● Suitable for small to large-scale applications. ● Easy to Modify. ● 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.
Cons	● Difficult to read, change, unit test, and reuse this model ● 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. ● No formal validation support ● Increased complexity and Inefficiency of data ● The difficulty of using MVC with the modern user interface	● Sometimes, understanding the flow can be confusing. ● Modification of Models / Views should be done carefully without affecting Templates ● Unlike most web development frameworks, Django can't handle multiple requests simultaneously.
Examples	ASP.NET MVC, Spring MVC, etc...	Django
Web frameworks	Ruby on rails, Catalyst, Laravel	Django

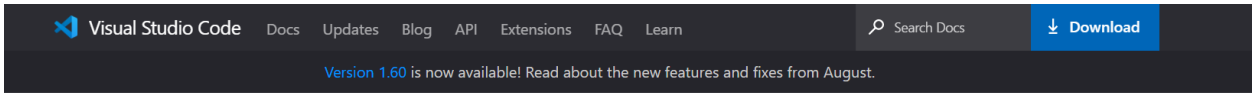
Part 2

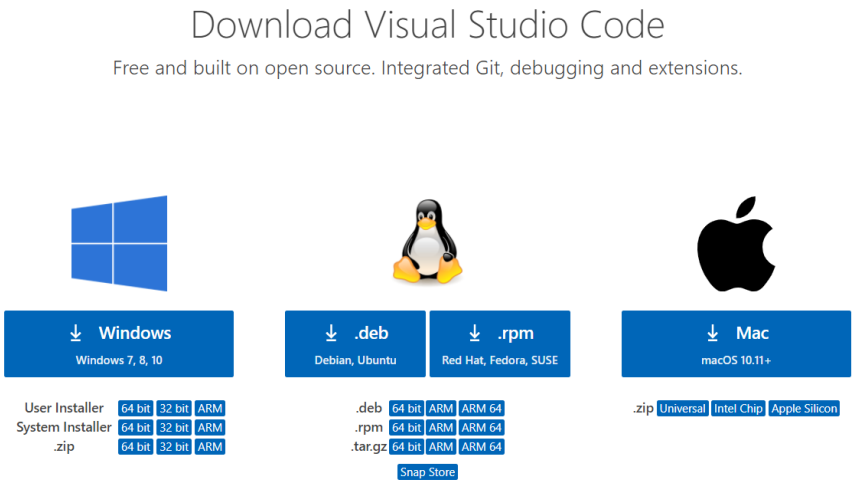
Installation of python, vscode, and configuring basic Django project.

1. Installation of python:

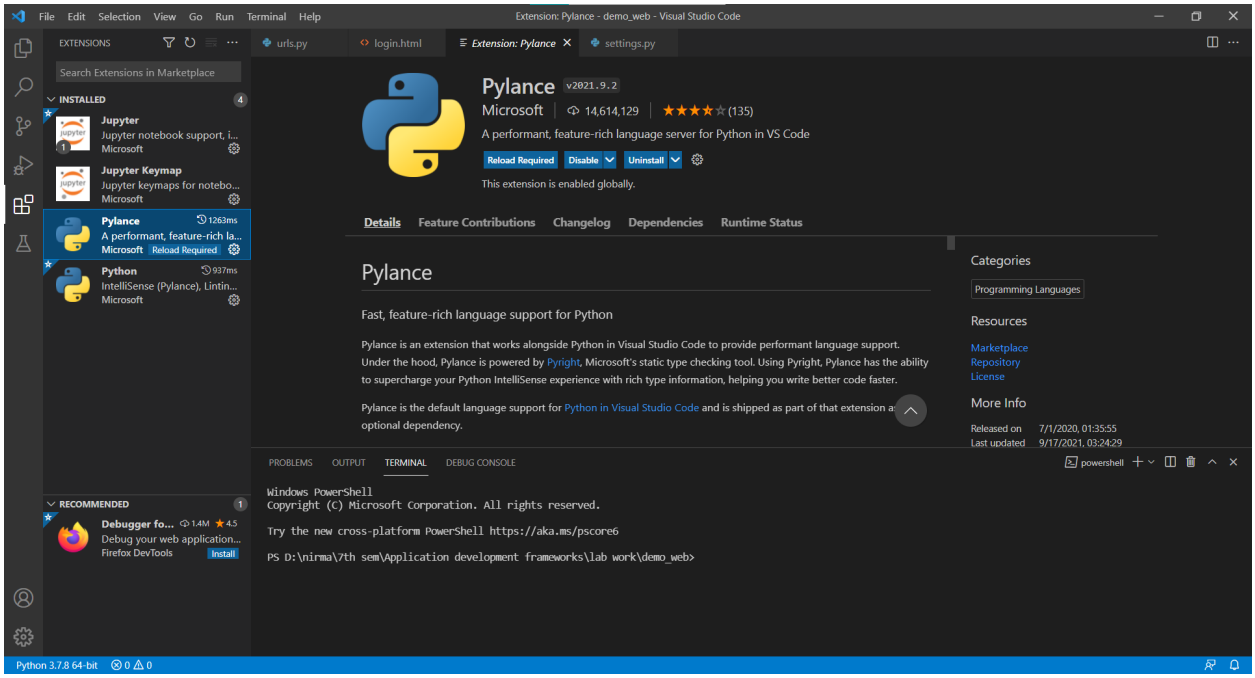


2. Vscode installation:

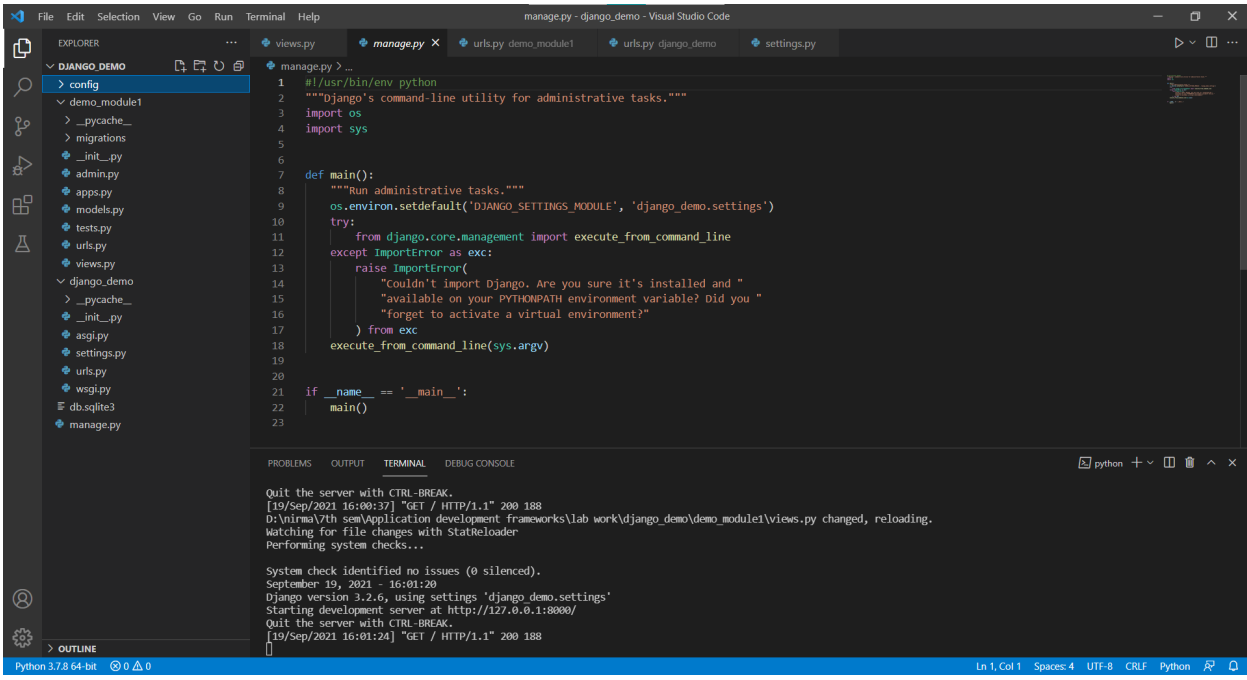




3. Run python pip install Django as:
   `python -m pip install Django`
4. Create a new django project as:
   `django-admin startproject django_demo`
5. Install python and pylance extension in vscode.

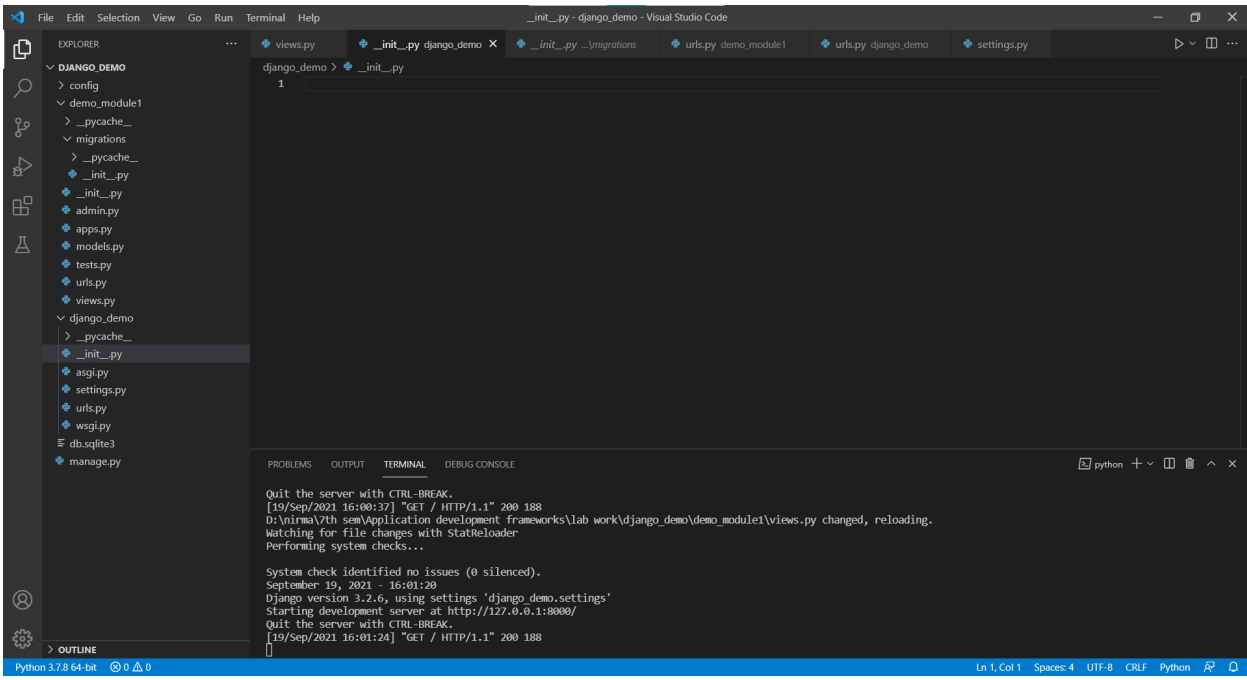


6. Files in the django django_demo project:
   a. manage.py:
      This file is responsible for starting the sever, migrating and controlling. It provides some project specific functionalities.



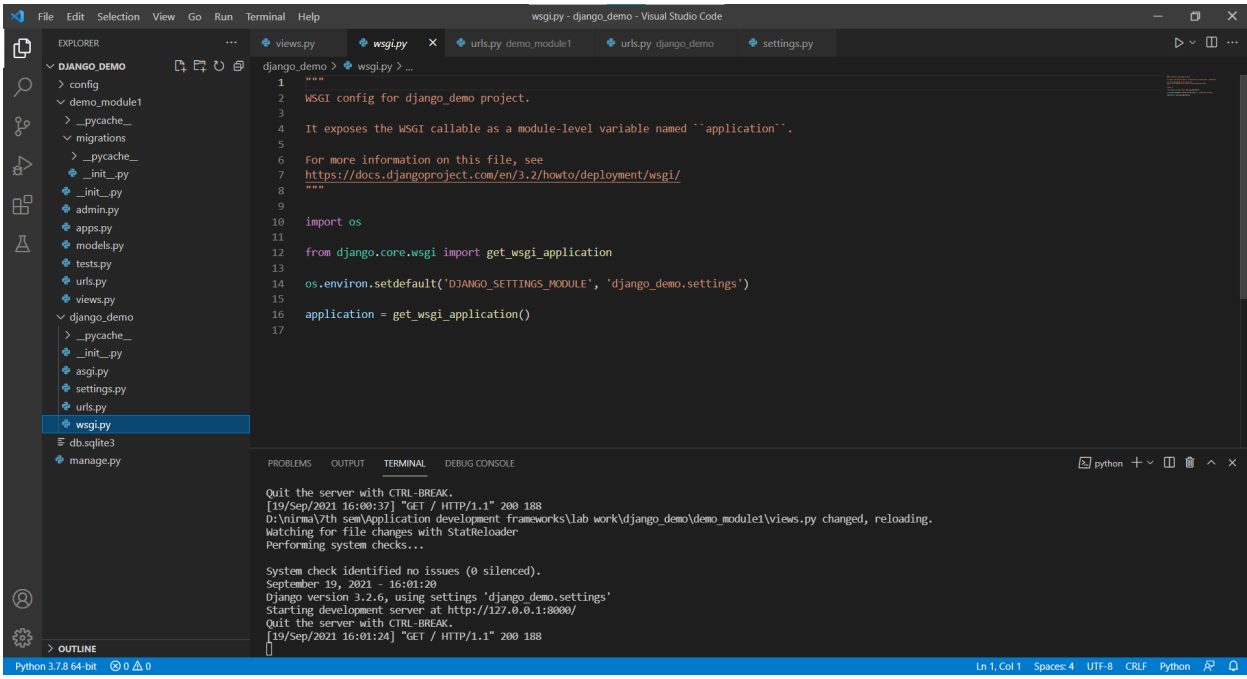
   b. __init__.py:

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



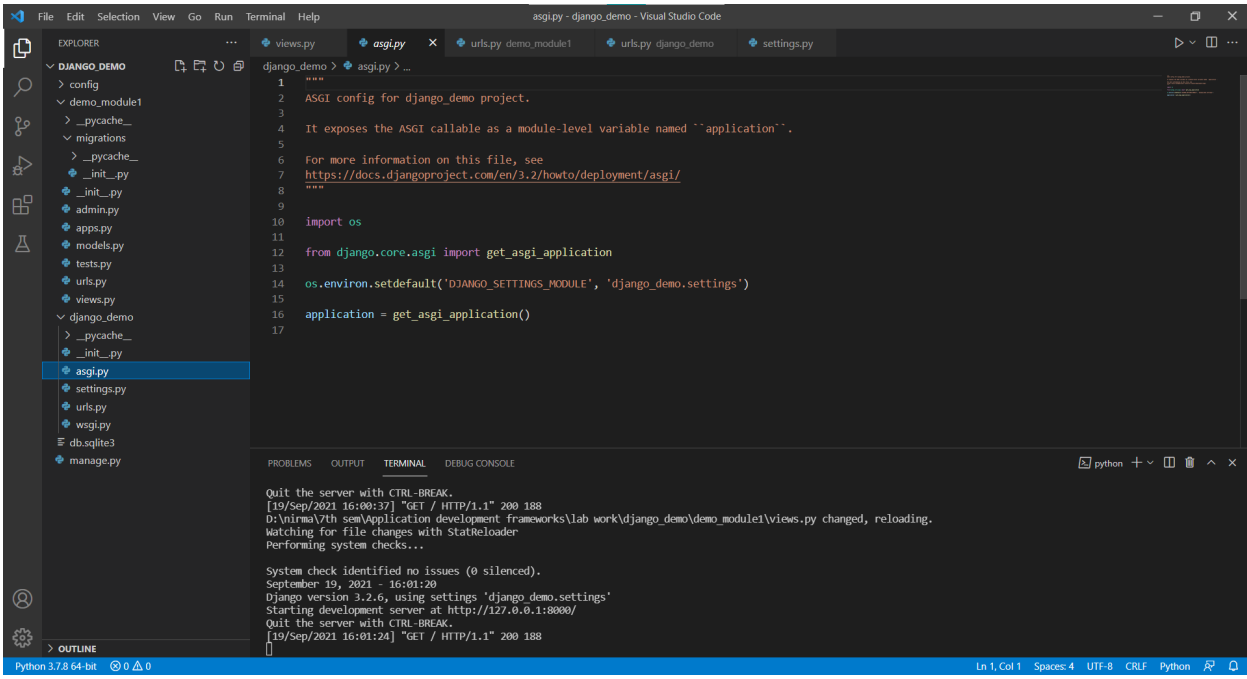
c. wsgi.py:

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.



d. asgi.py:

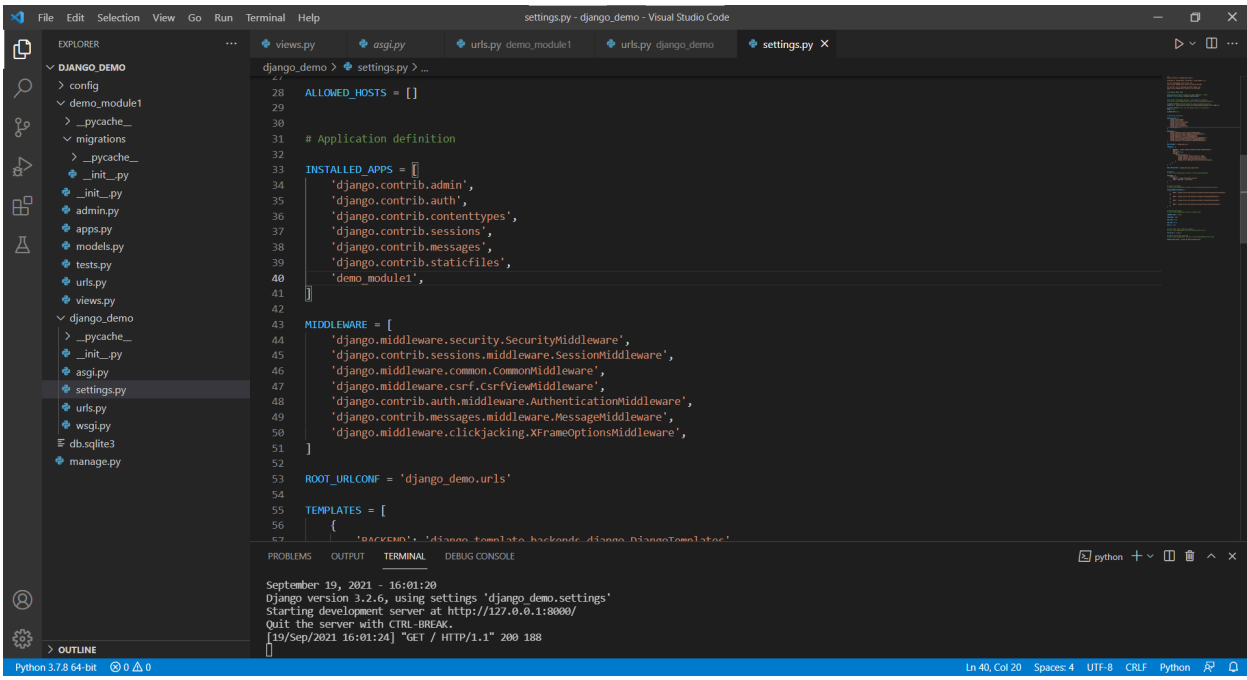
This file is similar to wsgi but has better functionalities. It stands for asynchronous server gateway interface.



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

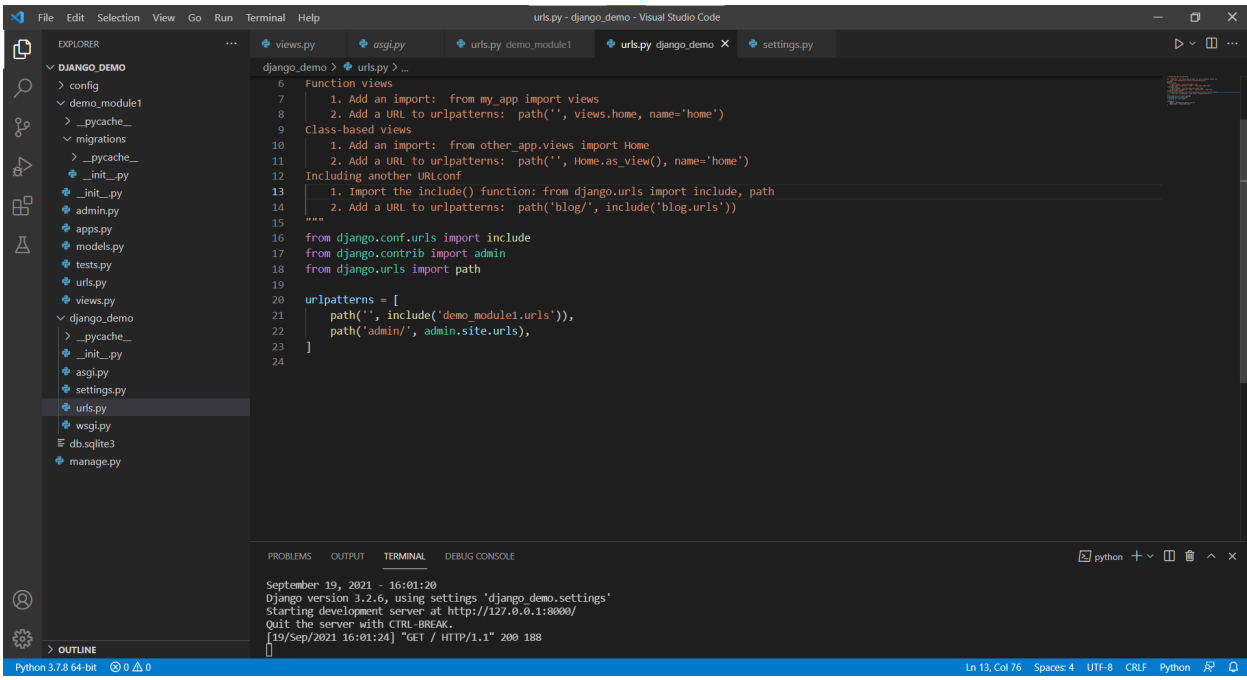
```
python manage.py startapp demo_module1
```



f. urls.py:

This file denotes the project level urls. It provides the addres 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.



7. Files in the demo_module1:

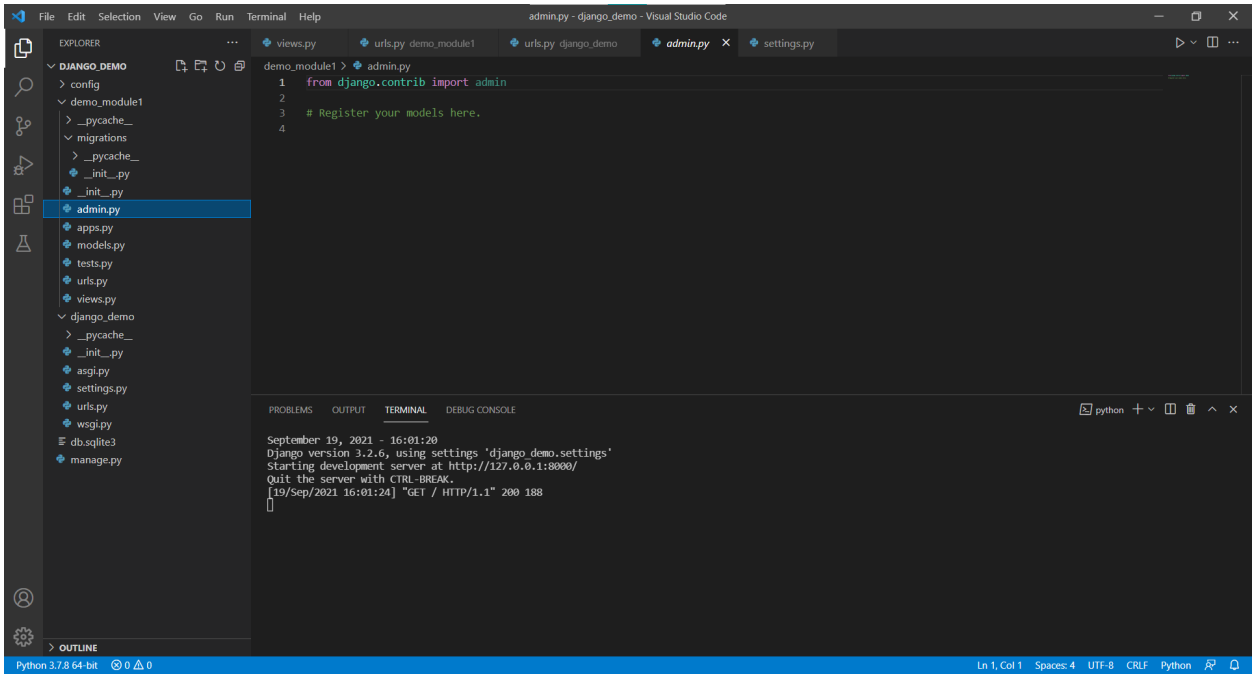
In 6(e) point I have mentioned how has the application been made under the project. The files under the same are:

a. __init__.py

   The same empty file with the same purpose

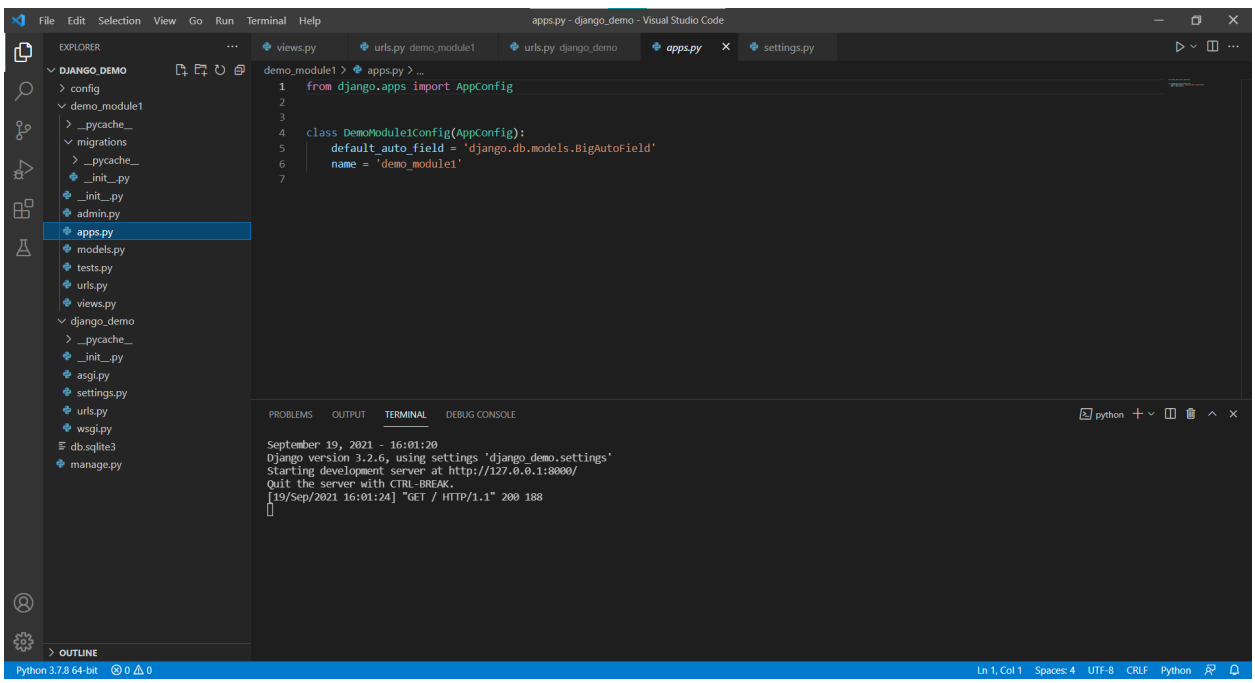
b. admin.py

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



c. apps.py

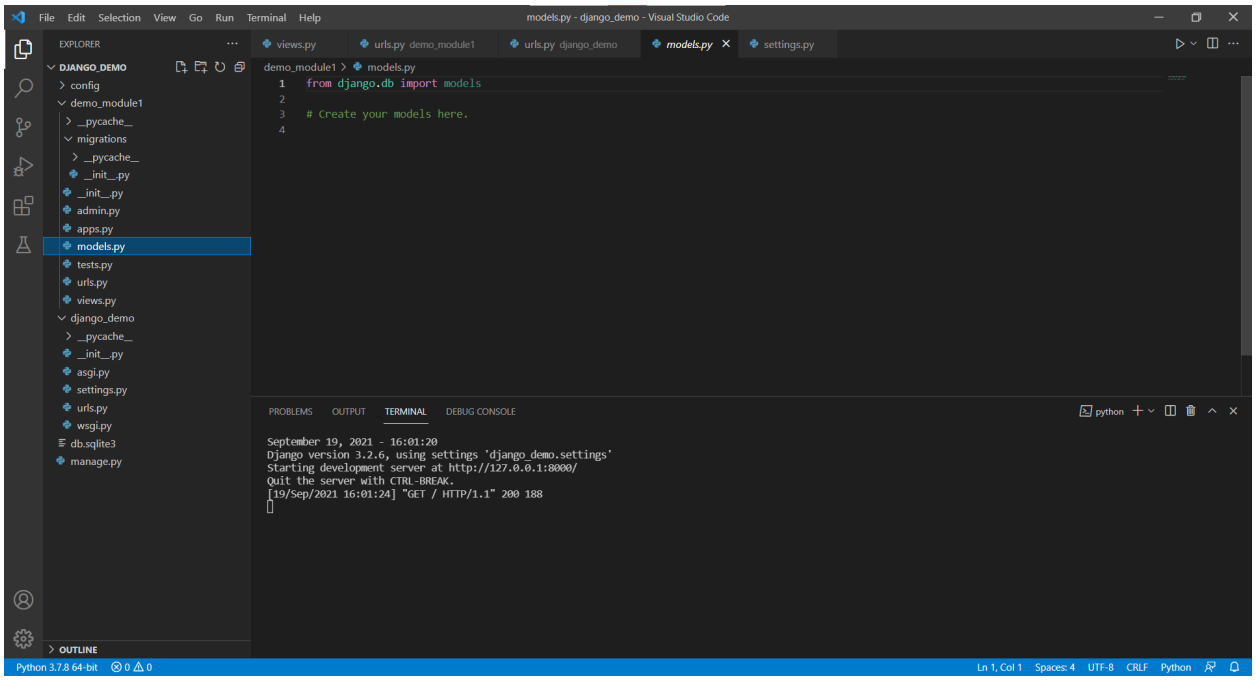
Used to help the user include the application configuration for their app.



d. models.py

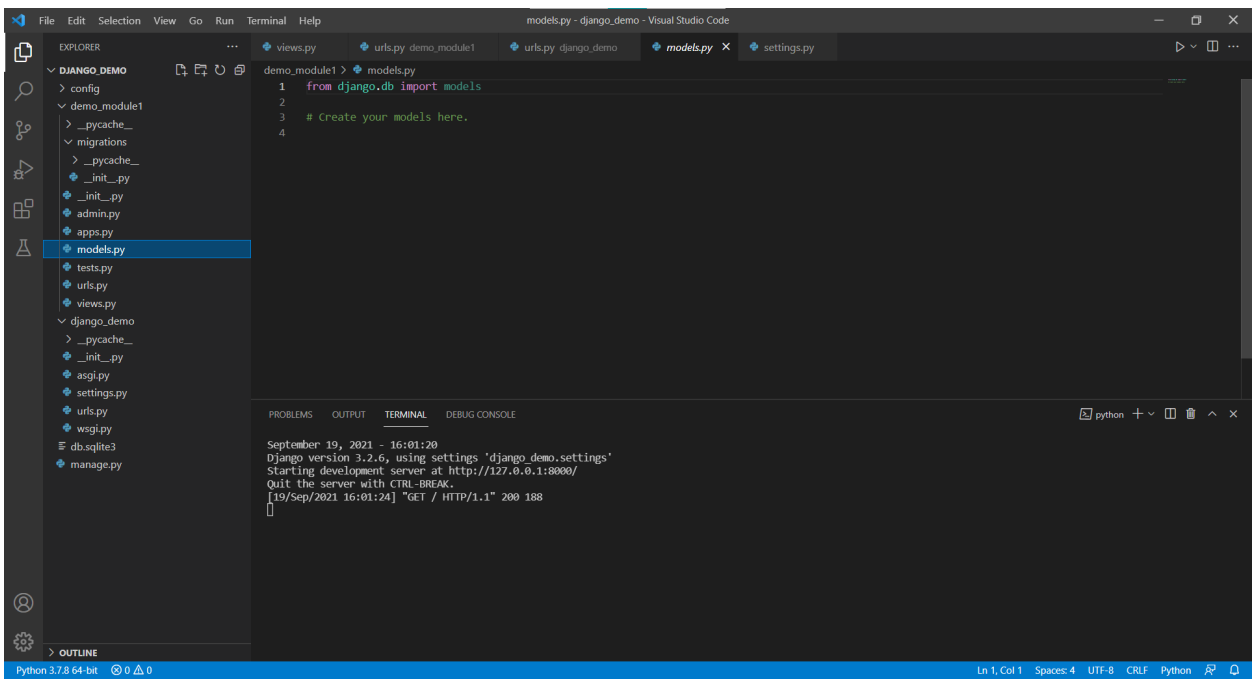
This represents the models of web applications in the form of classes. It is a part of app structure. It defines the structure of the database and tesla about

actual design, relationships between data sets.



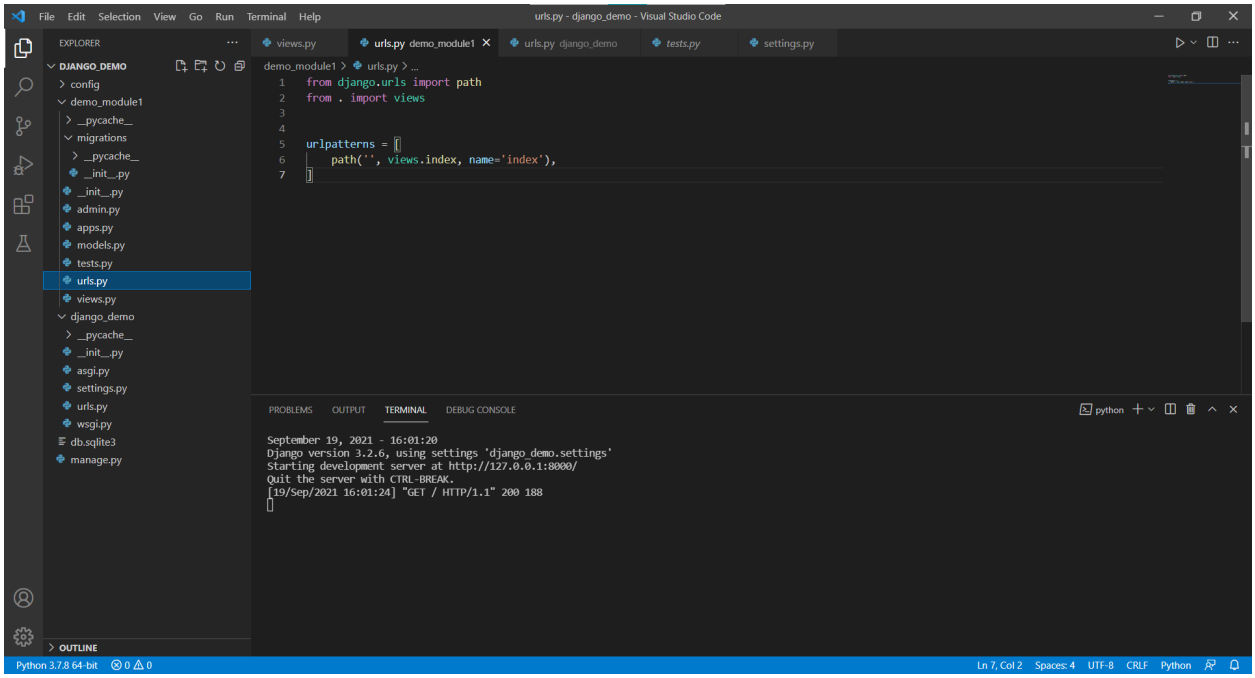
e. test.py

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



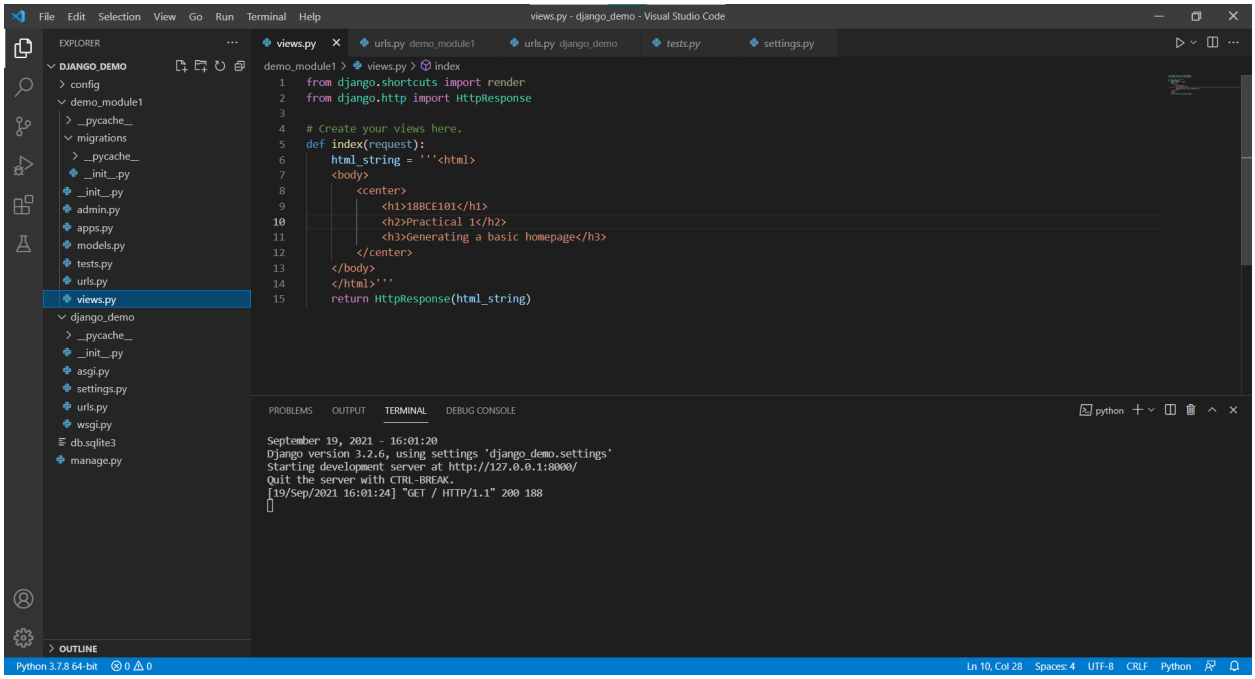
f. urls.py

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



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.

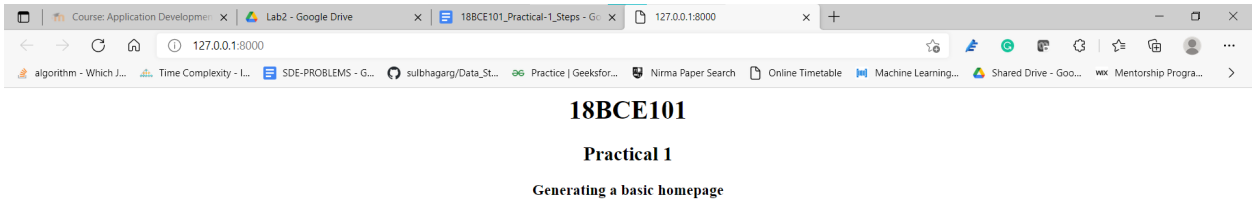


8. Running the output:

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.