DJANGO TASK – 1

1. What is a framework?

   Framework:

   The framework is a collection of modules or packages. Using Python frameworks helps in reducing development time as it provides a prebuilt implementation of redundant tasks and one doesn't need to worry about the common details like protocols, sockets, etc and they just need to focus on application logic.

   Python Frameworks for Web App Development

   1. Django

      Django is free, open-source Python framework that lets developers write complex code and apps quickly and elegantly. Django may be used to create web apps of excellent quality. Among Python frameworks for web development, Django is the most widely used and can be used to create web applications and APIs rapidly. This framework's numerous libraries, low code count, and component reuse make it popular.

   2. CherryPy

      This is a simple, quick, and reliable Python framework for creating web apps. It is an open-source software that can be used with any Python-supporting working framework. When utilising CherryPy, you can access data and create templates using any kind of technology. It is capable of creating sessions, static content, file uploads, cookies, and more. It functions similarly to a web framework.

   3. Pyramid

      Many tech giants, like Dropbox and Yelp, use it as an open-source Pyramid web development framework. The framework for web development is popular because it is adaptable and lightweight, and it can be used to create complicated projects as well as vital web applications. Python is used to run Pyramid.

   4. Grok

      Grok's open-source web framework is built on the Zope toolkit (ZPK) and helps to accelerate app development. It gives developers an agile development experience.

   5. TurboGears

      This Python web application framework was created to address the deficiencies of commonly used web and mobile app development frameworks. TurboGears 2 is developed on top of previous web frameworks like TurboGears 1, Rails, and Django.

6. Web2Py
   This framework is free and open source, built in Python. It is ideal for the rapid construction of secure database-driven web applications. It is a complete framework.

7. Flask
   This Python web framework is widely used for creating complicated web applications. Although this framework makes suggestions, it does not impose any dependencies or project layouts.

8. Bottle
   Bottle is a WSGI micro-web framework for building Python online applications that is lightweight, fast, and easy. It is solely dependent on the standard Python library.

9. Tornado
   Tornado is a Python web framework and a framework library with non-blocking framework I/O. The framework is designed in such a way that, when properly configured, it can manage over 10,000 concurrent connections. As a result, it is an important tool for developing apps with a large number of simultaneous clients.

10. BlueBream
    BlueBream is an open-source and best Python framework for web development, server, and library for website developers. It was previously known as Zope 3 & was created by the Zope team. It excels at medium- to large-scale operations and aids in segmentation into multiple reusable segments.

2. What is Django, what are the advantages of using Django?
   Django:
   Django is a free, open-source Python framework that enables rapid development of complicated code and applications by programmers. Python web developers can use it to create high-quality web apps. Django is widely used to construct APIs and web applications and is one of the top Python frameworks.

   Advantages
   - Helps you define URL patterns for your application.
   - Integrated authentication system.
   - Easy and effective URL scheme.
   - The database language for object-oriented programming that provides the finest data storage and recovery.

- Customizable modification, addition, and deletion is made possible by the automatic admin interface feature.
- Several cache mechanisms are supported by a framework for caching.

Limitations

- Not ideal for smaller projects, as it is a high-level framework.
- Can lead to slow websites depending upon the volume of requests to be handled.
- Lacks coding conventions like Rails
- Needs in-depth knowledge to be effectively used.

3. What is MVC and MVT pattern, what is the difference between them.
   MVC:
   Model View Controller, known as **MVC** separates the code as three components. MVC separates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC architecture has become popular for designing web applications as well as mobile apps.

   - Model – This layer deals with data-related logic. For example, it can retrieve, change, and save data to the database.

   - View – We can call it the presentation layer. It is responsible for collecting data from the model or user and presenting it. In a web application, everything that is displayed in the browser falls under View.

   - Controller – It controls the data flow and interaction between view and model. For example, a controller, based on a request or action, will collect data from a database with the help of Model and send it to the user through Views.

   Advantages

   - Makes it easy to develop large applications.
   - Easy for multiple developers to collaborate and work together.

   Disadvantages

   - View is controlled by Model and Controller.
   - Not suitable for small applications.

   MVT:

Model View Template, widely known as **MVT** is another design pattern similar to MVC. Like MVC, the MVT design pattern also separates the code into three parts.

- Model – Same as the model in MVC. It contains the code responsible for dealing with data and databases.

- View – In MVT design pattern, the View is decides what data should be displayed.

- Template – Templates are used to specify a structure for an output. Data can be populated in a template using placeholders. It defines how the data is presented. An example is a Generic list view that we can use to display a set of records from the database.

Advantages

- Less coupled.
- Suitable for small to large-scale applications.
- Easy to Modify.

Disadvantages

- Sometimes, understanding the flow can be confusing.
- Modification of Models / Views should be done carefully without affecting Templates.

Difference between MVC & MVT

S.NO.	Model View Controller (MVC)	Model View Template (MVT)
1.	MVC has controller that drives both Model and View.	MVT has Views for receiving HTTP request and returning HTTP response.
2.	View tells how the user data will be presented.	Templates are used in MVT for that purpose.
3.	In MVC, we have to write all the control specific code.	Controller part is managed by the framework itself.
4.	Highly coupled	Loosely coupled
5.	Modifications are difficult	Modifications are easy
6.	Suitable for development of large applications but not for small applications.	Suitable both small and large applications.
7.	Flow is clearly defined thus easy to understand.	Flow is sometimes harder to understand as compared to MVC.

8.	It doesn't involve mapping of URLs.	URL pattern mapping takes place.

4. What is a virtual environment? Why is it recommended to use the virtual environment in projects?

   Virtual Environment:

   A virtual environment is a Python environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in other virtual environments, and (by default) any libraries installed in a "system" Python, i.e., one which is installed as part of your operating system.

   Here's why using virtual environments is recommended in projects:

   - Isolate dependencies: Different projects may have different requirements for Python libraries and their versions. Using a virtual environment ensures that each project has its own set of dependencies, preventing conflicts between projects.

   - Maintainability: Virtual environments make it easier to manage and maintain projects, especially when collaborating with others. Everyone working on the project can use the same virtual environment with the exact dependencies, reducing errors and ensuring consistency.

   - Avoids conflicts: By isolating project dependencies, virtual environments prevent conflicts that might arise if system-wide libraries were used. This is especially important when working on multiple projects that require different versions of the same library.

   - Reproducibility: Virtual environments make it easier to reproduce a project's environment on different machines. By sharing the virtual environment requirements, other developers can easily set up the same environment to run or contribute to the project.