Long Questions – Django Overview

1. **Explain the Django MVT (Model-View-Template) architecture with an example.**
   Django follows the **MVT (Model-View-Template)** architecture:
   - **Model:** Handles database operations. (e.g., a Product model storing product details)
   - **View:** Contains the business logic and processes requests. (e.g., product_detail(request, id))
   - **Template:** Defines how data is presented (HTML files).

Example:

- models.py → class Product(models.Model): name = models.CharField(max_length=100)
- views.py → def product_detail(request, id): product = Product.objects.get(id=id)
- template.html → {{ product.name }}

---

2. **What is a virtual environment (virtualenv), and why is it used in Python projects?**
   A **virtual environment (venv/virtualenv)** is an isolated Python environment used to manage dependencies for a project without affecting global packages.

Why is it used?

- Prevents conflicts between dependencies.
- Allows different projects to use different versions of libraries.
- Provides a clean working environment.

---

3. **Explain how CRUD operations are implemented in Django with an example.**
   CRUD (Create, Read, Update, Delete) operations can be implemented using Django models and views:

Example: Managing Products

- **Create:** Product.objects.create(name="Laptop", price=50000)
- **Read:** Product.objects.all() or Product.objects.get(id=1)
- **Update:** product = Product.objects.get(id=1); product.price = 55000; product.save()
- **Delete:** product.delete()

CRUD operations can be implemented using Django's **views and forms** in a web application.

---

4. **Explain API and RESTFUL API in brief.**

- **API (Application Programming Interface):** A way for different software components to communicate.
- **RESTful API (Representational State Transfer API):** A web-based API that follows REST principles:
  - Uses HTTP methods (GET, POST, PUT, DELETE).
  - Follows a stateless architecture.
  - Uses JSON or XML for data exchange.

---

5. **Explain the working of Django Middleware and its role in request-response processing.**
   Middleware in Django is a layer that processes requests and responses globally before reaching views.

Working:

1. A request is received.
2. Middleware processes it (e.g., authentication, security).
3. The request is passed to the view.
4. The view returns a response.
5. Middleware processes the response before sending it to the client.

Example Middleware:

- AuthenticationMiddleware (checks user authentication).
- SessionMiddleware (handles user sessions).

---

6. **What is the significance of SQL in Django applications, and how does Django interact with databases?**
   - **SQL (Structured Query Language)** is used to manage databases in Django applications.
   - Django interacts with databases using **Django ORM (Object-Relational Mapping)**, which converts Python code into SQL queries.

Example:

- Instead of SELECT * FROM products, Django uses:
- Product.objects.all()
- Django supports multiple databases (PostgreSQL, MySQL, SQLite).

---

7. **Describe the key components of Django's ORM (Object-Relational Mapping).**
   Django ORM allows developers to interact with databases using Python instead of SQL.

Key components:

- **Models:** Define database structure (class Product(models.Model)).
- **QuerySet API:** Fetch or modify records (Product.objects.all()).
- **Migrations:** Manage database schema (python manage.py migrate).
- **Relationships:** Define relationships (ForeignKey, ManyToManyField).

Short Questions – Django Overview

1. **What is Django?**
   - Django is a high-level Python web framework that follows the **MVT (Model-View-Template)** architecture.
2. **What is the role of the requests library in Python?**
   - The requests library is used to send HTTP requests (GET, POST, PUT, DELETE) in Python applications.
3. **What is Virtualenv, and why is it used in Django?**
   - virtualenv is a tool to create isolated Python environments, preventing conflicts between dependencies.
4. **Explain the key components of Django architecture.**
   - **Model:** Handles database operations.
   - **View:** Processes requests and returns responses.
   - **Template:** Defines the UI layout.
   - **URL Dispatcher:** Maps URLs to views.
   - **Middleware:** Handles request-response processing.
5. **How do you create a new Django project?**
6. django-admin startproject projectname
7. **What command is used to create a Django app?**
8. python manage.py startapp appname
9. **How do you create a superuser in Django?**
10. python manage.py createsuperuser

11. **What is the purpose of the Django manage.py file?**
    - manage.py is a command-line tool for running Django commands like migrations, running the server, and creating users.
12. **Define RESTful architecture in web development.**
    - RESTful architecture follows principles like **statelessness, resource-based URLs, and HTTP methods (GET, POST, PUT, DELETE)**.
13. **What is SQL, and why is it important in web applications?**

- SQL (Structured Query Language) is used for database management, allowing CRUD operations on stored data.

11. **What are the basic CRUD operations in SQL?**

- **Create:** INSERT INTO table VALUES(...)
- **Read:** SELECT * FROM table
- **Update:** UPDATE table SET column=value WHERE condition
- **Delete:** DELETE FROM table WHERE condition