**✅ Django Framework Interview Questions (Complete & No Gaps)**

**🔹 1. Django Basics**

**🔹 1. What is Django? Why is it popular?**

Django is a Python web framework known for its rapid development, security, and built-in features.

**🔹 2. What is the architecture of Django (MVT)?**

MVT stands for Model-View-Template, which separates data, logic, and presentation layers.

**🔹 3. How is MVT different from MVC?**

In MVT, Django handles the controller part as the framework itself, and the template replaces the view in MVC.

**🔹 4. What is the role of manage.py?**

manage.py is a command-line tool to manage the Django project (runserver, migrations, etc.).

**🔹 5. What is inside settings.py?**

settings.py contains project configurations like database, apps, middleware, static files, etc.

**🔹 6. What is the difference between a Django project and a Django app?**

A project is the full web application, while an app is a modular component within it.

**🔹 7. How do you start a new Django project and a new app?**

Use django-admin startproject projectname and python manage.py startapp appname.

**🔹 Final Interview-Friendly Version (Refined, but Easy Style):**

* **Rapid Development**: Django supports rapid development — it helps us build web applications quickly and efficiently by providing built-in tools.
* **Security**: Django includes built-in protection against common web attacks like SQL injection, XSS, CSRF, and clickjacking.
* **Built-in Features**: Django provides many features out-of-the-box like admin panel, ORM, authentication, URL routing, and form handling.
* **Model**: A Django model is a Python class that defines the structure of a database table and handles the interaction between the application and the database.
* **View**: Views handle the business logic of the application. They process user requests and return appropriate responses.
* **Template**: Templates control the presentation layer — they define what the user sees on the frontend (UI).
* **makemigrations**: This command creates migration files. It’s like a preparation step for applying database changes.
* **migrate**: This command applies the migrations and updates the database schema.
* **Database (DB)**: The entire storage system where application data is stored.
* **Schema**: Schema defines the structure of the database — it includes tables, fields, data types, relationships, and constraints.
* **Middleware**: Middleware is a layer that processes requests and responses between the client and the view.  
  (User → Request → Middleware → View → Response → Middleware → User)
* **Project Configuration**: These are the settings that define how the Django project behaves. This includes settings.py, URLs, WSGI, ASGI, etc.
* **Modular Component**: Django apps are modular components. Each app (like blog, users, payments) can be developed independently and reused in other projects.

**🔹 2. URL Routing and Views**

**🔹 1. What is Django's URL dispatcher and how does it work?**

Django’s URL dispatcher maps incoming HTTP requests to the appropriate view using patterns defined in urls.py.

**🔹 2. How do you map URLs using urls.py?**

You use path() or re\_path() in urls.py to link URL patterns to view functions or classes.

**🔹 3. What is the difference between path() and re\_path()?**

path() uses simplified syntax, while re\_path() allows complex URL matching using regular expressions.

**🔹 4. Function-based view vs Class-based view?**

Function-based views are simple Python functions; class-based views offer reusability and built-in behaviors using OOP.

**🔹 5. What is the use of HttpResponse, render(), and redirect()?**

* HttpResponse() returns raw HTML/text;
* render() combines template + context and returns HTML;
* redirect() sends users to another URL.

**🔹 6. How does template rendering work in Django?**

Django loads a template file, fills it with dynamic data from context, and returns it as an HTML response.

Notes:

* URL Pattern – it’s a route defined in a web framework that maps a specific url path to a corresponding view or hadler function

**🔹 3. Django Templates**

**🔹 1. What are Django templates and how do you use variables, filters, and tags?**

Django templates are HTML files with placeholders for variables, filters (to modify data), and tags (for logic).

**🔹 2. What is template inheritance?**

Template inheritance allows a child template to extend a base template to reuse common layout.

**🔹 3. What is the context dictionary?**

The context dictionary passes dynamic data from views to templates for rendering.

**🔹 4. How are static files (CSS/JS/images) handled?**

Static files are managed via the STATICFILES\_DIRS setting and served using {% static %} tag.

**🔹 5. What are {% block %}, {% extends %}, {% include %} used for?**

* {% block %} defines replaceable sections,
* {% extends %} inherits a parent template,
* {% include %} embeds another template.

**🔹 4. Models and ORM**

**🔹 1. What is Django ORM? How does it work?**

Django ORM lets you interact with the database using Python classes instead of raw SQL.

**🔹 2. Common field types:**

* CharField: For short text,
* IntegerField: For numbers,
* DateTimeField: For date & time,
* ForeignKey: For one-to-many relationships.

**🔹 3. What is the difference between makemigrations and migrate?**

makemigrations creates migration files; migrate applies them to the database.

**🔹 4. What is a QuerySet?**

A QuerySet is a collection of database rows retrieved using Django ORM.

**🔹 5. Common ORM queries:**

* .all() – fetches all,
* .filter() – filters records,
* .get() – gets a single match,
* .exclude() – excludes records,
* .values() – returns dicts,
* .annotate() – adds computed fields,
* .aggregate() – calculates totals/summaries.

**🔹 6. Model relationships:**

* OneToOneField: One-to-one link,
* ForeignKey: Many-to-one,
* ManyToManyField: Many-to-many link.

**🔹 7. What is the use of Meta class?**

The Meta class customizes model behavior like table name, ordering, and permissions.

**🔹 8. How to define a custom model manager?**

By creating a class inheriting models.Manager and defining custom query methods.