DJANGO

1. What is Django?

Django is a Full-stack web development framework that facilitates the creation and maintenance of high-quality Dynamic pages while also encouraging rapid development and a clean, pragmatic style. Django makes it easier to automate repeated operations, resulting in a more efficient development process with fewer lines of code.

2.Name some Companies that use Django.

Some companies that use the Django framework are Instagram, Mozilla, Disqus, Bitbucket, Nextdoor, and Clubhouse.

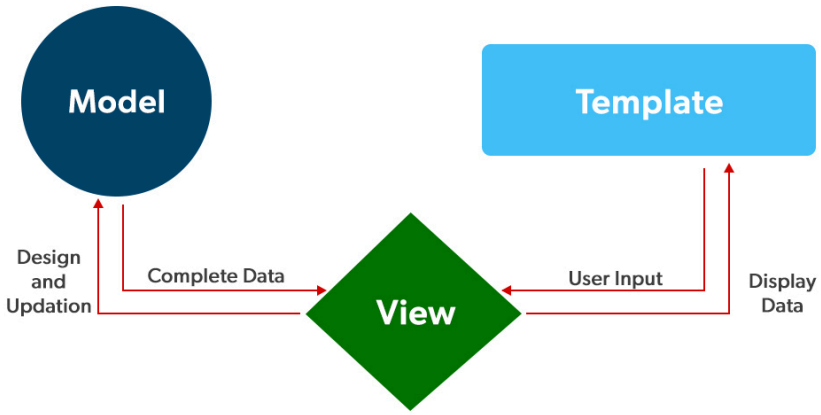
3.What is the difference between a Project and an App?

The main difference between a project and an app is that a project is defined as the entire application whereas, an app is part of the project that is self-sufficient to perform any task.

4. Explain Django's architecture.

The Model-View-Template also known as MVT architecture is used by Django. It is a software design pattern for developing a web application. The Django MVT Structure is made up of three parts:

- The model will serve as an interface for your data. It is in charge of data management. A database represents the logical data structure that supports the entire application such as MySql and Postgres.
- The View is the user interface, that renders a website page in your browser. HTML/CSS/Javascript and Jinja files are used to represent it.
- A template is made up of both static sections of the desired HTML output and specific syntax that describes how dynamic content will be included.



6. What are the Features of using Django?

- Flexible server arrangement,
- Model relation database
- Provide object-relational mapper
- Web templating system

- **Middleware class support**
- **Regex-based URL Dispatcher**
- **Unit testing framework**
- **Admin Interface**
- **Django is SEO optimized.**
- **In-build mitigation**
- **Easy inheritance**

7.Explain the Django project directory structure.

When you first start a Django project, it comes with some basic files like manage.py and view.py.

1. **init.py – It's an empty Python file. It is called when the package or one of its modules is imported. This file tells the Python interpreter that this directory is a package and that the presence of the __init.py_ file makes it a Python project.**
2. **manage.py – This file is used to interact with your project from the command line utility. with the help of this command, we can manage several commands such as:**
    1. **manage.py runserver**
    2. **manage.py makemigration**
    3. **manage.py migrate' etc**
3. **setting.py – It is the most important file in Django projects. It holds all the configuration values that your web app needs to**

work, i.e. pre-installed, apps, middleware, default database, API keys, and a bunch of other stuff.

4. views.py – The View shows the user the model's data. The view knows how to get to the data in the model, but it has no idea what that data represents or what the user may do with it.
5. urls.py – It is a universal resource locator which contains all the endpoints, we store all links of the project and functions to call it.
6. models.py – The Model represents the models of web applications in the form of classes, it contains no logic that describes how to present the data to a user.
7. wsgi.py – WSGI stands for Web Server Gateway Interface, This file is used for deploying the project in WSGI. It helps communication between your Django application and the web server. more...
8. admin.py – It is used to create a superuser Registering model, login, and use the web application.
9. app.py – It is a file that helps the user to include the application configuration for their app.

8. How do you create a Django project?

We can create a Django project with the help of the following command

```
django-admin startproject projectname
```

9. How do you create a Django app?

We can create a Django app with the help of the following command

python manage.py startapp appname

10. How do we start our development server?

We can start our development server with the help of the following command

python manage.py runserver

11. Importance of virtual environment setup for Django.

A virtual environment allows you to establish separate dependencies of the different projects by creating an isolated environment that isn't related to each other and can be quickly enabled and deactivated when you're done.

It is not necessary to use a virtual environment without a virtual environment we can work with Django projects. However, using virtualenv is thought to be the ideal practice. Because it eliminates dependencies and conflicts.

12.Give a brief about the Django admin interface.

Django provides us with a default admin interface that is very helpful for creating, reading, updating, and deleting model objects that allow the user to do administrative tasks. It reads a set of data that explains and provides information about data from the model

in order to create a quick interface where the user can alter the application's contents. This is an in-built module.

13. What are Django URLs?

The routing of a website is determined by its URLs. In the program, we build a python module or a file called urls.py. The navigation of your website is determined by this file. When a user visits a given URL route in the browser, the URLs in the urls.py file are compared. The user then receives the response for the requested URL after retrieving a corresponding view method.

14. What are the views of Django?

Django views are an important feature of the MVT Structure. This is a function or class that takes a web request and delivers a Web response, according to the Django script. This response could be an HTML template, a content of a Web page, an XML document, a PDF, or images. the view is a part of the user interface that renders the HTML/CSS/Javascript in your Template files into what you see in your browser when we render a web page.

15. What are the models in Django?

Model is a built-in feature of Django that contain a definitive source of information such as behavior and essential fields about the data we are storing. It allows us to build tables, fields, and constraints and organize tables into models. Generally, each model maps to a single database table. To use Django Models, you'll need a project and an app to work with. Also, Django makes use of SQL to access the database. SQL is a complex language with many

queries for generating, removing, updating, and other database-related tasks.

16. What do the following commands do?

- **python manage.py makemigrations**
- **python manage.py migrate**

The makemigration command scans the model in your application and generates a new set of migrations based on the model file modifications. This command generates the SQL statement, and when we run it, we obtain a new migration file. After running this command, no tables will be created in the database.
Running migrate command helps us to make these modifications to our database. The migrate command runs the SQL instructions (produced by makemigrations) and applies the database changes. After running this command, tables will be created.

17. Define static files and explain their uses.

Static files are used to save files such as CSS, JavaScript, pictures, and other types of static files. We keep them in distinct folders, for as the js folder, which has all of the JavaScript files, and the images folder, which contains all of the images. These files are kept in the static subfolder of the project app. Django provides django.contrib.staticfiles which helps us to manage static files.
There are different uses for static files:

- **It clarifies the use of file methods and attributes.**
- **It is platform-independent in many ways, whereas generic files are not.**
- **Subclasses can be used to extend it.**

18.What are templates in the Django language?

A Django template is a text document that is used to give a front and a layout for our website. It is the third and most significant aspect of Django's MVT Structure. In Django, a template is an HTML file that contains HTML, CSS, and Javascript. The Django framework efficiently manages and generates dynamically generated HTML web pages for end-user viewing. Django is mostly a backend framework, thus we use templates to give a layout for our website. There are two ways to incorporate the template into our website.

- We can utilize a single template directory that will be distributed throughout the project.
- We can make a separate template directory for each app in our project.

19.What does the settings.py file do?

settings.py is a core file in Django projects. It holds all the configuration values that your web app needs to work; database settings, logging configuration, where to find static files, API keys if you work with external APIs, and a bunch of other stuff.

21. Difference between MVC and MVT design patterns?

- Model and View are both driven by the controller in MVC, whereas Views in MVT are used to receive HTTP requests and return HTTP responses.
- We must write all of the control-specific code in MVC whereas, We must write all of the control-specific code in the View and Template components in MVT.
- MVC is Highly coupled whereas, MVT is loosely coupled.
- In MVC, it is difficult to modify whereas, Modification is easy in MVT.
- MVC is suitable for large applications, but MVT is suitable for both small and large applications.
- MVC does not involve any URL mapping, whereas in MVT URL pattern mapping takes place.
- Flow is clear and easy to understand, whereas MVT is sometimes harder to understand.

22.What is Django ORM?

ORM which is also known as the object relation model enables us to interact with our database. It allows us to add, delete, change, and query objects (Object Relational Mapper). Django uses a database abstraction API called ORM to interface Viewed with its database models, to use Django object relation Models, we must first have a project and an app running. Models can be created in app/models.py after an app has been started. The Django ORM may be accessed by running the following command in our project directory.

```
python manage.py shell
```

This opens a Python console where we may add objects, retrieve objects, modify existing items, and delete objects.

23.What is Superuser?

superuser is the most powerful user with permission to create, read, delete, and update on the admin page which includes model records and another user. Users of Django have access to an Admin Panel. Before using this feature, you must have to migrate your project; otherwise, the superuser database will not be created. To create a superuser, first, reach the same directory, and run the following command

```
python manage.py createsuperuser
```

24.What is Jinja templating?

Jinja is also known as jinja2, which is the most recent version. It's a template engine that allows you to make HTML, XML, and other markup types. Jinja2 is valuable since it features a templating tag syntax and because the project has been extracted as a standalone open-source project that may be utilized as a dependency by other code libraries. Some of its features are:

- HTML Escaping – It provides automatic HTML Escaping as <, >, & characters have special values in templates and if using a regular text, these symbols can lead to XSS Attacks which Jinja deals with automatically.
- Sandbox Execution – This is a framework for automating the testing process in a sandbox (or protected) environment.
- Template Inheritance
- Produces HTML templates far more quickly than the default engine.
- When compared to the default engine, it is easier to debug.

25.What do you mean by the csrf_token?

Cross-Site Request Forgery (CSRF) is one of the most serious vulnerabilities, and it can be used to do everything from changing a user's information without their knowledge to obtaining full control of their account. To prevent malicious attacks, Django provides a per cent token per cent tag {% csrf_token %} that is implemented within the form. When generating the page on the server, it generates a token and ensures that any requests coming back in are cross-checked against this token. The token is not included in the inbound requests, thus they are not executed.

26.Explain the use of Middlewares in Django.

Middleware is a lightweight plugin in Django that is used to keep the application secure during request and response processing. The application's middleware is utilized to complete a task. Security, session, CSRF protection, and authentication are responsible for doing some specific functions. The application's security is maintained by the usage of the middleware component, AuthenticationMiddleware which is associated with user requests using sessions.

27.What are 'signals'?

Signals are used to take action in response to the modification or creation of a database entry. Its utilities help us to connect events with their action. we can create methods that will run a signal when it is called. For example, as soon as a new user instance is generated in the Database, one might want to create a profile instance. Generally, There are 3 types of signals.

1. pre_delete/post_delete: This signal is thrown before the remove() method is used to delete a model's instance.

2. pre_init/post_init: This signal is thrown before/after instantiating a model (__init__() method)
3. pre_save/post_save: This signal works before/after the method save().

28.What is Media Root?

Media root is used to upload user-generated content. We can serve user-uploaded media files locally, using the MEDIA_ROOT and MEDIA_URL settings. User-upload these files are referred to as Media or Media Files in Django. The first step is to include the code below in the settings.py file.

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'

MEDIA_ROOT: It is for the server path to store files in the computer.
MEDIA_URL: It is the referring URL for the browser to access the files over HTTP.

29.How you can include and inherit files in your application?

Using the extends tag in Templates, we can inherit our files in Django, The extends tag is used to inherit these templates. The syntax for inheriting these templates is given as:
{% extends 'template_name.html' %}
This syntax helps us to add all the elements of an HTML file into another without copy-pasting the entire code. Django templates not only allow us to pass variables from view to template, but they also provide some programming capabilities like loops, comments, and extensions.

30.How do you connect your Django Project to the database?

We need to configure our database in the settings.py file. By default, SQLite is mentioned there, and we need to change this setting accordingly like Postgres, MongoDB, and MySql.

31.Explain the caching strategies of Django. ?

Django has its own inbuilt caching system that allows us to store our dynamic pages. So that we don't have to request them again when we need them. The advantage of the Django Cache framework is that it allows us to cache data such as templates or the entire site. Django provides four different types of caching options, they are:

- per-site cache – It is the most straightforward to set up and caches your entire website.
- per-view cache – Individual views can be cached using the per-view cache.
- Template fragment caching allows you to cache only a portion of a template.
- low-level cache API – It can manually set, retrieve, and maintain particular objects in the cache using the low-level cache API.

32.Give the exception classes present in Django.

An exception is a rare occurrence that causes a program to fail. Django has its own exception classes to cope with this circumstance, and it also supports all fundamental Python exceptions. some of the exception classes are listed below:

- MultipleObjectsReturned – If just one item is anticipated but many objects are returned, this error is thrown by the query.

- **ViewDoesNotExist – When a requested view does not exist, Django.URLs raise this exception.**
- **PermissionDenied – It's triggered when a user doesn't have the necessary permissions to perform the requested activity.**
- **SuspiciousOperation – The query throws this error if only one item is expected but several things are returned.**
- **ValidationError – It's triggered when data validation fails on a form or a model field.**
- **FieldDoesNotExist – It raises when the requested field does not exist.**
- **ObjectDoesNotExist – The base class for DoesNotExist exceptions.**
- **AppRegistryNotReady – It is raised when attempting to use models before the app loading process.**
- **EmptyResultSet – If a query does not return any result, this exception is raised.**

33.What is No SQL and Does Django support NoSQL?

NoSql is also known as a non-relational database that stores data in a form of non-tabular, instead it stores data in the form of a storage model that is optimized for specific requirements of the type of data being stored. The types of NoSQL databases include pure documented databases, graph databases, wide column databases, and a key-value store.
No, Django does not officially support no-SQL databases such as CouchDB, Redis, Neo4j, MongoDB, etc.

34.What are the different model inheritance styles in Django?

Django supports 3 types of inheritance. They are

- **Abstract base classes**

- Multi-table Inheritance
- Proxy models

35.What databases are supported by Django?

Databases that are supported by Django are SQLite(Inbuild), Oracle, PostgreSQL, and MySQL. Django also uses some third-party packages to handle databases including Microsoft SQL Server, IBM DB2, SAP SQL Anywhere, and Firebird. Django does not officially support no-SQL databases such as CouchDB, Redis, Neo4j, MongoDB, etc.

36.How would you query all the items in the database table?

XYZ.objects.all()

Where XYZ is some class created in the model.

37.How to query one item from the database table?

XYZ.objects.get(id=1)

Where XYZ is some class created in the model.

38.What is Django Rest Frameworkcont

The REST Framework is an HTTP-based standard for listing, generating, modifying, and deleting data on your server. The Django REST framework which is also known as DRF is a powerful and flexible toolkit built on top of the Django web framework that simplifies the creation of REST interfaces by reducing the amount

of code required. there are different advantages of using REST Framework like:

- Web browsable API that provides huge usability for developer
- Authentication policy which includes packages for 0auth1 and auth2.
- It supports both ORM and non-ORM data sources.
- It has extensive documentation and great community support.

39.Explain the Django Response lifecycle.

The Django Response lifecycle is responsible for data interchange between clients and servers with the help of the HttpRequest object, whenever a request is made by the client to the server it passes information through the system using request and response objects. these request/response objects are transmitted over the web which contains request metadata such as images, HTML, CSS, and javascript. These data are then loaded and presented to the user by Django, which passes the HttpRequest as the first argument to the view method. It is the responsibility of each view to return a HttpResponse object.

40.How do filter items in the Model?

To filters items present in our database we use a QuerySet. It is a database collection of data that is built up as a list of objects. QuerySet makes our work easy by allowing us to filter and organize the data, it is also easier to retrieve the information that we need with the help of QuerySet. we can filter our data with the help filter() method that allows us to return only the rows that match the search word.

41.What is the difference between CharField and TextField in Django?

- TextField is a database field that is used to store big amounts of text. Paragraphs, data, and other items can be saved. CharField should be used to hold tiny text such as First name and Last name. Let's make a new instance of the TextField we just made and see whether it works.
- CharField is a string field, for small- to large-sized strings. It is generally used for storing small strings like first names, last names, etc. To store larger text TextField is used.

42.Give a brief about the settings.py file.

It's the Django file's main settings file, as the name says. Everything inside the Django project is saved in this file as a dictionary or list, including databases, middlewares, backend engines, templating engines, installed applications, static file URLs, main URL configurations, authorized hosts, servers, and security keys. When Django files are started, the settings.py file is first executed, followed by the loading of the appropriate databases and engines to swiftly serve the request.

43.What are Django cookies?

A cookie is a piece of information stored in the client's browser. To set and fetch cookies, Django provides built-in methods to use the set_cookie() method for setting a cookie and the get() method for getting the cookie. we can also use the request.COOKIES['key'] array to get cookie values.

44.How to check the version of Django installed on your system?

Open the command prompt and enter the following command
py -m django --version

45.Why is Django called a loosely coupled framework?

Django is known as a loosely connected framework because of its MTV architecture. Django's design is an MVC variant, and MTV is advantageous since it totally separates server code from the client's hardware. The client machine has Models and Views, and templates are only returned to the client. All of the architectural elements make distinct from one another.

46.Explain Django Security.

The security of users' data is an important aspect of any website design. Django provides adequate security against a number of common threats. Django's security features are as follows:

- Cross-site scripting (XSS) protection
- SQL injection protection
- Cross-site request forgery (CSRF) protection
- Enforcing SSL/HTTPS
- Session security
- Clickjacking protection
- Host header validation

47.Explain user authentication in Django.

Django comes with an authentication system configured by default to handle objects like users, groups, permissions, and so on. The authentication system's core is made up of user objects. It not only authenticates users but also authorizes them. Aside from using the default, we can employ a variety of web apps instead of using the

default system to enable more user authentication. The default system objects are as follows:

- Users
- Permissions
- Groups
- Password Hashing System
- Forms Validation
- A pluggable backend system

48. What is the “Django.shortcuts.render” function?

We need the render function when a View function produces a web page as a HttpResponse instead of a basic string. Render is a shortcut for passing a template and a data dictionary. This function combines templates with a data dictionary using a templating engine. Finally, render() provides a HttpResponse containing the rendered text as well as the data from the models. Syntax: render(request, template_name, context=None, content_type=None, status=None, using=None)

49.What is a context in Django?

In Django, a context is a dictionary in which the keys represent variable names and the values reflect the values of those variables. This dictionary or context is supplied to the template, which finally outputs the dynamic content using the variables. i.e. {var1: 11, var2: 12}, when you pass this context to the template render method, {{ var1 }} would be replaced with 11 and {{ var2 }} with 12 in your template.

50. What is serialization in Django?

Serializers in the Django REST Framework are responsible for transforming objects into data types that javascript and front-end frameworks can understand. After validating the incoming data, serializers also enable deserialization, which allows parsed data to be transformed back into complex types.

51.Why is Django Called a Loosely Coupled Framework?

Django is called a loosely coupled framework because its components (models, views, templates) are independent and interchangeable. This modularity allows developers to modify or replace each component without affecting the others significantly. For instance, you can change the database layer without needing to alter the view logic or presentation layers, adhering to the principle of separation of concerns.

52.Is Django a REST API

Django itself is not a REST API; it is a web framework for building web applications using Python. However, you can build REST APIs within Django using Django REST Framework (DRF), a powerful and flexible toolkit that works seamlessly with Django to build web APIs.

52.How can you retrieve data from the database using Django's ORM?

You can retrieve data using the Django ORM by using methods like .objects.all(), .filter(), .get(), and more on a model's QuerySet. These

methods generate SQL queries and return Python objects, making database interactions more intuitive.

54.What are the disadvantages of Django?

Following is the list of disadvantages of Django:

- Django' modules are bulky.
- It is completely based on Django ORM.
- Components are deployed together.
- You must know the full system to work with it.

55.What are Django cookies?

A Django cookie is a little piece of data stored in the client's browser. Django has built-in methods for setting and retrieving cookies. We use the set_cookie() method to create a cookie and the get() method to retrieve it.
To obtain cookie values, utilize the request.COOKIES['key'] array.

56.List the inheritance styles in Django?

Ans.:
There are three possible inheritance styles in Django, and they are:

- Abstract Base Classes: This is used when we want to make the parent class hold the information which we don't want to repeat again in the child class. Abstract Base Classes does not create any database table. It just makes changes in the database table created by the derived class.
- Multi-table Inheritance: This is used when we want to subclass an existing model and there must be a database table designed for each model on its own. It means Multi-table

Inheritance creates a database table for each derived class and base class

- Proxy models: This is used to modify the Python level behavior of the model, without modifying the model's fields. It does not make any changes to the database.

57.Explain how you would use Django signals.

"Django signals allow for sending and receiving notifications when certain actions occur. They're useful for executing code in response to model changes without altering the model's code. I've used signals to send email notifications when a new user signs up. However, I use them sparingly to avoid hidden side-effects and maintain code clarity."

58.What GIL (global interpreter lock)? How does it allow multi-threading in python?

Answer: GIL is a lock which makes sure python interpreter is held by only one thread at a time. That means only one thread can be in a execution status at any point of time. And this is why python is a single threaded programming language. This was introduced to solve the reference count problem which is used by python for garbage collection.
Then how is multi-threading module working in python, you may ask!
Well, this global interpreter lock applies only for CPU bound activities. So if there is any code which affects or uses CPU it automatically starts acting as a single threaded. But all the general programs can still work in a multi-threaded fashion.

58.How to Create Django Signals?

The Django Signals is a strategy to allow decoupled applications to get notified when certain events occur. Let's say you want to invalidate a cached page everytime a given model instance is updated, but there are several places in your code base that this model can be updated. You can do that using signals, hooking some pieces of code to be executed everytime this specific model's save method is trigged.

Another common use case is when you have extended the Custom Django User by using the Profile strategy through a one-to-one relationship. What we usually do is use a "signal dispatcher" to listen for the User's post_save event to also update the Profile instance as well.

59.How to filter items or values from database in django ?

Selecting and filtering out object can be done in below ways :
A) : Selecting All Records :
model.objects.all()it will select all record with all fields.

B) : Selecting Particular Records :

myuser.objects.get(first_name='john')

It will select records where first name is 'john'

C) : Selecting Particular Records and field :

myuser.objects.get(first_name='john').last_name

It will select all records with firstname as 'john' and give their 'last_name' value only not other fields of the record.

60.How Django response lifecycle work?

Whenever a web request is made Django creates an HTTP response to that metadata. Afterward, Django loads the view and passes the HTTP request to an argument toward the view function. Then each view returns an HTTP response.
The following steps show how the response cycle works.
Initially, the system.py file is created with MIDDLEWARE classes.
Then these middlewares are executed as mentioned in the MIDDLEWAREST.
The request is now moved to the URL router which accesses the URL path from the request and tries to match it with the developers' given URL path in urls.py.
When it's mapped, it will call a view function, from where the equivalent HTTP response is generated.
This response passes through MIDDLEWARE and is sent back to the client side.

61.What is Django middleware?

Django middleware is a way to add extra functionality to the request/response processing pipeline in a Django application. Middleware sits between the web server and the view, and can modify requests and responses or perform other actions. Examples of middleware in Django include authentication middleware, caching middleware, and middleware for handling exceptions.

62.What is the difference between a middleware and a view?

A: A view is a Django function or class that handles a particular HTTP request and returns an HTTP response. Middleware, on the other hand, sits between the web server and the view and can modify requests and responses or perform other actions.

Middleware is applied to every request/response that passes through it, whereas views are only called when a particular URL is requested.

63.What is mixin?

Mixin is a sort of multiple inheritances wherein you can consolidate practices and properties of more than one parent class. Mixins give a magnificent method to reuse code from different classes. For instance, conventional class-based perspectives comprise of a mixin considered TemplateResponseMixin whose reason for existing is to characterize render_to_response() strategy. At the point when this is joined with a class present in the View, the outcome will be a TemplateView class.
One downside of utilizing these mixins is that it gets hard to examine what a youngster class is doing and which strategies to abrogate if there should arise an occurrence of its code being excessively dispersed between different classes.

64.What is the difference between Flask and Django?

Flask	Django
Flask is a WSGI framework	Django is a Full-stack web framework
It allows multiple types of databases.	It doesn't support multiple types of databases.
Use SQL Alchemy	Build-in ORM
Diversified Working Style	Monolithic Working Style
Arbitrary structure	Conventional Project Structure

It supports API	**It does not have any support for API**
It does not support Dynamic HTML pages	**Django accepts Dynamic Pages.**
It has support for Visual debug	**No support for Visual Debug**
It doesn’t offer a built-in bootstrapping tool.	**Django-admin enables us to start building web applications without any external input,**
URL dispatcher is a RESTful request.	**URL dispatcher is Robust Documentation.**