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. What are models in Django?

A model in Django refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

What are views in Django?

A view function, or “view” for short, is simply a Python function that takes a web request and returns a web response. This response can be HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image, etc.

There are two types of views:

- Function-Based Views: In this, we import our view as a function.
- Class-based Views: It’s an object-oriented approach.

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.

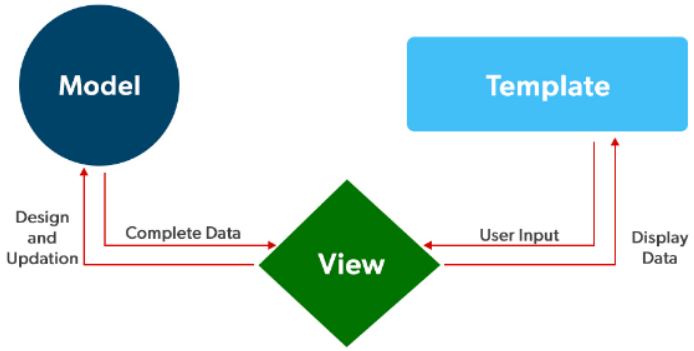
4.Explain Django’s architecture?

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.



5.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 teasing framework
- Admin Interface
- Django is SEO optimized.
- In-build mitigation
- Easy inheritance

6.How do you create a Django project?

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

- django-admin startproject projectname

7.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

8.How do we start our development server?

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

- python manage.py runserver

9.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.

10.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.

11.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.

12. 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.

13.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.

14.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

15.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.

16. What is context in the Django?

Context is a dictionary mapping template variable name given to Python objects in Django. This is the general name, but you can give any other name of your choice if you want

17.How to view all items in the Model?

ModelName.objects.all()

18. How to filter items in the Model?

ModelName.objects.filter(field_name="term")

19. How to get a particular item in the Model?

ModelName.objects.get(id=”term”)
Note: If there are no results that match the query, get() will raise a DoesNotExist exception. If more than one item matches the given get() query. In this case, it’ll raise MultipleObjectsReturned, which is also an attribute of the model class itself

Extra

1. Request-Response Cycle in Django

Django's request-response cycle follows this sequence:

- **Request**: When a user makes a request (like loading a webpage), it first hits the web server.
- **URL Routing**: Django then maps the request to a view based on the URL patterns.
- **View**: The appropriate view processes the request, interacting with the database if needed, and preparing a response.
- **Template Rendering**: If the view renders a template, it fetches the HTML, adds dynamic content, and returns it to the client.
- **Response**: Django returns an HTTP response (HTML, JSON, etc.) back to the client.

---

2. Difference between `get_object_or_404()`, `get()`, and `filter()`

- `get()`: Retrieves a single object that matches the query. If no object or multiple objects are found, it raises an exception.
- `filter()`: Returns a QuerySet of objects that match the query. If no match is found, it returns an empty QuerySet.
- `get_object_or_404()`: Similar to `get()`, but returns an HTTP 404 error if no object is found instead of raising an exception.

---

3. Does Django Support Multiple-Column Primary Keys?

No, Django does not directly support composite primary keys (multiple-column primary keys). Instead, you can create unique constraints across multiple fields using `unique_together` or `UniqueConstraint`.

---

4. How to See Raw SQL Queries Running in Django?

You can see raw SQL queries in Django using:

- `query` attribute: `queryset.query` will show the raw SQL of a QuerySet.
- Django Debug Toolbar: Install this tool to view SQL queries in the browser.
- Logging: Configure Django's logging settings to log database queries.

---

5. What is a QuerySet in Django?

A QuerySet is a collection of database queries to retrieve, filter, and manipulate data in Django. It represents a lazy, iterable sequence of database records. Examples include methods like `filter()`, `all()`, `exclude()`, etc.

---

6. What is a Mixin in Django?

A mixin is a type of inheritance that allows you to add common behavior or functionality to multiple classes. In Django, mixins are commonly used in views (like `LoginRequiredMixin`) to add functionality such as authentication, permissions, etc.

---

7. What is Django Field Class?

In Django models, a Field class defines the data type of a field and its behavior. Each model field (like `CharField`, `IntegerField`, etc.) is a subclass of the `Field` class, which defines properties like validation, database mapping, and form rendering.

---

8. How to Combine Multiple QuerySets in a View?

You can combine multiple QuerySets using:

- **`union()`**: Combines QuerySets with the same model and fields.
- **Chaining QuerySets**: Using `|` operator for an OR-like combination.

Example:

python
Copy code
```
combined_queryset = queryset1.union(queryset2)
```

---

9. Explain Q Objects in Django ORM

`Q` objects allow for complex queries using logical conditions (AND, OR, NOT) in Django ORM. They can be combined using `&` (AND) and `|` (OR), allowing more flexibility in filtering QuerySets.

Example:

python
Copy code
```
from django.db.models import Q
results = Book.objects.filter(Q(title__icontains='django') |
Q(author__name__icontains='john'))
```

---

10. Difference between `select_related()` and `prefetch_related()`

- **`select_related()`**: Performs a SQL join and selects related objects in a single query, which is efficient for **one-to-one** or **many-to-one** relationships.
- **`prefetch_related()`**: Executes separate queries for related objects and combines them in Python, making it useful for **many-to-many** or **reverse foreign key** relationships.

---

11. Query to Retrieve All Books Published in 2023

python
Copy code

```
books_2023 = Book.objects.filter(publication_date__year=2023)
```

---

12. Query to Retrieve All Authors Who Have Published At Least One Book in 2023

python
Copy code

```
authors_with_books_2023 = Author.objects.filter(books__publication_date__year=2023).distinct()
```

---

13. Optimize Query with `select_related` or `prefetch_related`

python
Copy code

```
authors_with_books_2023 = Author.objects.prefetch_related('books').filter(books__publication_date__year=2023).distinct()
```

---

14. Equivalent SQL Query for Task 13

sql
Copy code

```
SELECT DISTINCT author.id, author.name
FROM author
INNER JOIN book ON author.id = book.author_id
WHERE book.publication_date BETWEEN '2023-01-01' AND '2023-12-31';
```

---

15. Query to Retrieve Total Books and Average Price of Books for Each Author

python
Copy code

```
from django.db.models import Count, Avg

author_stats = Author.objects.annotate(
    total_books=Count('books'),
    average_price=Avg('books__price')
).values('name', 'total_books', 'average_price')
```

---

16. What is Context in Django?

In Django, the context refers to the data passed from the view to the template, allowing dynamic content rendering. It is often passed as a dictionary.

---

17. Difference Between `OneToOneField` and `ForeignKey` in Django

- **`OneToOneField`**: A one-to-one relationship. Each object in the related model is associated with exactly one object in the current model.
- **`ForeignKey`**: A many-to-one relationship. Multiple objects in the current model can be related to one object in the related model.

---

18. What is a QuerySet in Django?

(Already answered in question 5, see above).