Oral Questing----------

1. Which file is responsible for defining the database settings for a Django project?

Ans: settings.py

2. Explain the purpose of the django admin site?

Ans: The Django admin site is a built-in web-based interface that provides a powerful and customizable administrative interface for managing a Django web application.

3. What is the purpose of a URL pattern in django?

Ans: In Django, a URL pattern is a mechanism used to define how URLs should be processed and handled by the application.

Example:

```
	from django.urls import path
	urlpatterns = [
		path("", function_name, name="path_name"),
	]
```

4. What is the defference between path() and re_path() functions in django URL patterns?

Ans:

# Path = The path() function is used for simple, straightforward URL patterns that do not involve complex regular expressions.

# The re_path() function is used when you need to use regular expressions to define more complex and flexible URL patterns.

Example:

```
	from django.urls import path
	urlpatterns = [
		path("", function_name, name="path_name"),
		re_path(r'^example/(?P<slug>[\w-]+)/$', dynamic_view),
	]
```

6. What does the 'yesno' filter do in Django templates?

Ans: In Django templates, the yesno filter is used to display one of two specified strings based on whether a given value is true or false.

Exam:

```
{{ value | yesno:"yes,no,maybe" }}
```

5. What does the 'date' flter do in Django templates?
Ans: In Django templates, the date filter is used to format date and time values according to a specified format. It allows you to present date and time information in a human-readable way by applying a specific format to a datetime object.
Example:
```
        {{ my_date_variable|date:"m F, Y" }}
```

7. What is a Django form and what is its purpose?
Ans: In Django, a form is a Python class that defines the structure and behavior of an HTML form. The purpose of a Django form is to simplify the process of collecting and validating user input on the server side.
Example:
```
        from django import forms
        class ContactForm(forms.Form):
            fields_name = forms.CharField(label='Field Label Name',
widget=forms.TextInput())
        def contact_view(request):
                form = ContactForm()
                return render(request, 'contact.html', {'form': form})
```

8. How is a Django form submitted in a template?
Ans: In Django, forms are typically submitted in templates using HTML <form> elements.
Example:
```
        from django import forms
        class ContactForm(forms.Form):
                fields_name = forms.CharField(label='Field Label Name',
max_length=50, widget=forms.TextInput())
        def contact_view(request):
                form = ContactForm()
                return render(request, 'contact.html', {'form': form})
        <!-- templates/contact.html -->
        <form method="post" action="{% url 'contact_view' %}">
                {% csrf_token %}
                {{ form.as_p }}
                <button type="submit">Submit</button>
        </form>
```

9. What is a static file URL in web development?

Ans: In web development, a static file URL refers to the web address (URL) used to access static files such as images, stylesheets (CSS), JavaScript files, fonts, and other assets that do not change dynamically based on user input or server-side processing.

Example:

```
	<link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}">
```

10. What is the purpose of serializers in DRF?

Ans: In Django Rest Framework (DRF), serializers are used to converting complex data types, such as Django models or querysets, into native Python datatypes that can be easily rendered into JSON, XML, or other content types Additionally, serializers handle the reverse process, deserializing data from incoming requests into complex Python data types.

Example:

```
	from rest_framework import serializers
	class BookSerializer(serializers.Serializer):
		class Meta:
			model = Book
			fields = ['field1', 'field2']
	from rest_framework.views import APIView
	class BookAPIView(APIView):
		def get(self, request, *args, **kwargs):
			books = Book.objects.all()  # Fetch all books from the database
			serializer = BookSerializer(books, many=True)  # Serialize the
	queryset
			return Response(serializer.data)
		def post(self, request, *args, **kwargs):
			serializer = BookSerializer(data=request.data)
			if serializer.is_valid():
				serializer.save()
				return Response({"message": "Book created successfully"})
			else:
				return Response(serializer.errors, status=400)
```

Write The Short Answer of following Question---

1. Which file is responsible for defining the database settings for a django project?

Ans: In a Django project, the database settings are typically defined in the settings.py file.

Example:

```
	DATABASES = {
		'default': {
			'ENGINE': 'django.db.backends.sqlite3',
			'NAME': BASE_DIR / 'db.sqlite3',
		}
	}
```

2. Where should you store your static files like CSS and JavaScripts in a Django project?

Ans: In Django Project, static files such as CSS and JavaScripts files are store 'static' folder in project main directory or within each app. Use {% static %} template tag to reference them in templates.

Example:

```
	myproject/
	|-- myproject/
	|-- myapp/
	|-- static/
	|   |-- image/
	|   |-- styles.css
	|   |-- script.js
	|-- templates/
	|-- manage.py
```

3. How is a URL pattern defined in django's 'urls.py' file?

Ans:Define URL patterns in Django's urls.py using urlpatterns list with path() or re_path() functions, associating each pattern with a view function.

Example:

```
	from django.urls import path
	from .views import function_name
	urlpatterns = [
		path('', function_name, name="path_name"),
	]
```

4. What is the purpose of the 'reverse()' function in Django with respect to URLs?
Ans: reverse() in Django generates a URL for a view or URL pattern name, promoting maintainable code by avoiding hardcoded URLs.
Example:
```
        # urls.py
        from django.urls import path
        from .views import my_view
        urlpatterns = [
                path('my-url/', my_view, name='my-view'),
        ]
        from django.urls import reverse
        url = reverse('my-view', args=[1, 'example'])
```

5. What does the 'date' filter do in django templates?
Ans: In Django templates, the date filter is used to format date and time values according to a specified format. It allows you to present date and time information in a human-readable way by applying a specific format to a datetime object.
Example:
```
        {{ my_date_variable|date:"m F, Y" }}
```

6. What does the purpose of the 'length' filter in Django templates?
Ans:In Django templates, the length filter is used to get the length of a variable. It works with various type of variable such as string, list, querysets etc.
Example:
```
        {% with my_list=[1, 2, 3, 4, 5] %}
                The length of the list is: {{ my_list|length }}
        {% endwith %}
```

8. How do you perform a bulk delete operation on multiple records in Django models?
Ans: In Django, you can perform a bulk delete operation on multiple records in a model using the delete() method provided by the QuerySet.
Example:
```
        from yourapp.models import YourModel
        YourModel.objects.filter(age__lt=30).delete()
```

7. How do you create a new database record (object) using Django ORM?

Ans: Create a new database record (object) using Django ORM by using the create or save build-in functions.

Example:

```
	from yourapp.models import YourModel

	new_object = YourModel(name='John Doe', age=25,
email='john@example.com')
	new_object.save()
	or
	YourModel.objects.create(name='John Doe', age=25,
email='john@example.com')
```

9. How do you define a form class in django?

Ans: To define a form class in Django, you create a subclass of forms.Form and specify the form fields using classes from the forms module.

Example:

```
	# forms.py
	from django import forms
	class YourForm(forms.Form):
		name = forms.CharField(max_length=100)
		age = forms.IntegerField()
		email = forms.EmailField()
```

12. How can you include Bootstrap in your HTML file?

Ans: To include Bootstrap in your HTML, by using the Bootstrap CDN (Content Delivery Network) or download the Bootstrap files locally in your project.

Example:

```
	<!-- Include Bootstrap CSS -->
	<link rel="stylesheet" href="filepath/bootstrap.min.css">
	<script src="filepath//bootstrap.min.js"></script>

	<!-- Include Bootstrap CSS from CDN -->
	<link rel="stylesheet" href="/bootstrap-cdn-link/bootstrap.min.css">
	<script src="/bootstrap-cdn-link/bootstrap.min.js"></script>
```

10. How do you access form data submitted through POST method in a Django view?

Ans: In a Django view, you can access form data submitted through the POST method using the request.POST dictionary. When a form is submitted via POST, the form data is sent as part of the request, and you can retrieve it in your view.

Example:

```
    from django import forms
    class YourForm(forms.Form):
        name = forms.CharField(max_length=100)
    def your_view(request):
        if request.method == 'POST':
            form = YourForm(request.POST)
            if form.is_valid():
                name = request.POST['name']
        else:
            form = YourForm()
        return render(request, 'your_template.html', {'form': form})
```

11. What is the role of the 'get_context()' method in custom form widgets in django?

Ans: The get_context() method in custom Django form widgets is used to provide extra context data for widget rendering, enhancing customization options in templates. It allows you to add information that can be accessed within the template using field.widget.custom_data.

13. What is the role of the 'is_authenticated' attribute in Django user authentication?

Ans: In Django, the is_authenticated attribute is a built-in attribute of the User object provided by the authentication system. This attribute is used to check whether a user is authenticated or not.
When a user is successfully authenticated, the is_authenticated attribute returns True.

Example:

```
    if request.user.is_authenticated:
        # Code for authenticated users
    else:
        # Code for anonymous users
```

14. What are Bootstrap models, and how are they implemented?
Ans: Bootstrap modals are dynamic dialog boxes in web development used to display content or forms; they are implemented using HTML, CSS, and Bootstrap's JavaScript functionality.

15. Name two commonly used web servers with Django and briefly explain their roles.
Ans: Two commonly used web servers with Django are:

1. **Gunicorn**
2. **uWSGI**