1. **What are the key features of the Django Template language, and how do they contribute to its advantages in webdevelopment?**

The Django Template Language (DTL) is a powerful tool in Django web development, offering several key features that contribute to its advantages:

1. **Template Inheritance**: DTL allows for the creation of a base template that can be extended by other templates. This promotes code reusability and maintainability by reducing redundancy in HTML code.
2. **Template Tags and Filters**: DTL provides a rich set of built-in template tags and filters for performing various operations within templates. These include looping constructs, conditionals, date formatting, string manipulation, and more. They enhance the expressiveness and flexibility of templates.
3. **Context Variables**: Django templates can access variables passed to them via the view context. This allows for dynamic content generation based on data retrieved from the database or other sources.
4. **Security Features**: DTL comes with built-in security features to prevent common vulnerabilities such as Cross-Site Scripting (XSS) attacks. It automatically escapes HTML content by default, reducing the risk of injecting malicious scripts.
5. **Extensibility**: Django templates are highly extensible, allowing developers to create custom template tags and filters to suit specific project requirements. This flexibility enables developers to extend the functionality of templates as needed.
6. **Integration with Django ORM**: Django templates seamlessly integrate with Django's Object-Relational Mapping (ORM) layer, allowing easy access to database objects and query results directly from templates.

Overall, these features make the Django Template Language a robust and efficient tool for building dynamic web applications with Django, promoting clean code, security, and rapid development.

Top of Form

**2) Create a Django view using both function-based and class-based approaches. Utilize different Django**

**Template Language (DTL) tags within these views.(attach screenshots of SITE)**

**Function based**

from django.shortcuts import render

from datetime import datetime

from django.views import View

def home(request):

    context = {

        'message': 'Hello from function-based view!',

        'current\_time': datetime.now(),

        'items': ['Apple', 'Banana', 'Orange'],

    }

    return render(request, 'home.html', context)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Function-based View</title>

</head>

<body>

    <h1>{{ message }}</h1>

    <p>Current Time: {{ current\_time }}</p>

    <ul>

        {% for item in items %}

            <li>{{ item }}</li>

        {% endfor %}

    </ul>

</body>

</html>

from django.urls import path

from . import views

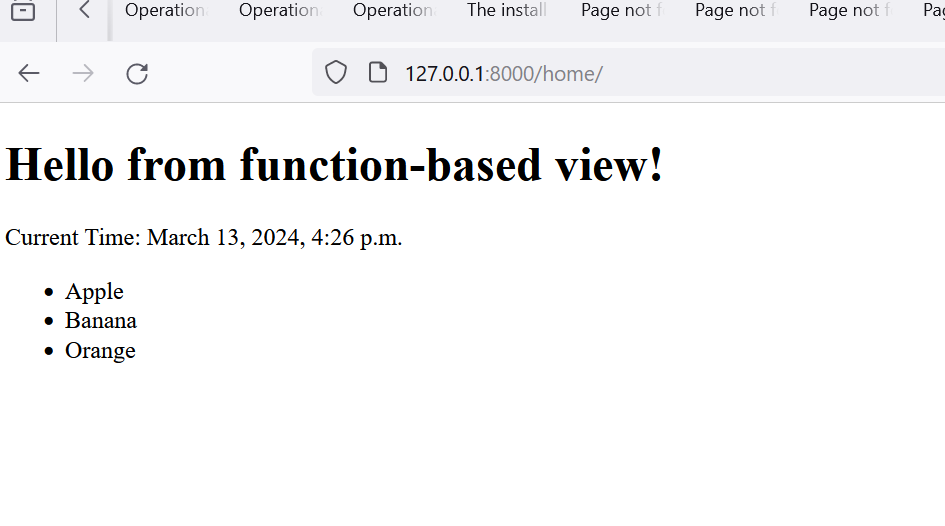
urlpatterns = [

    path('home/',views.home),

    path('Class/',views.Classbased.as\_view())

]

**Output**

****

**Class Based**

from django.shortcuts import render

from datetime import datetime

from django.views import View

def home(request):

    context = {

        'message': 'Hello from function-based view!',

        'current\_time': datetime.now(),

        'items': ['Apple', 'Banana', 'Orange'],

    }

    return render(request, 'home.html', context)

class Classbased(View):

    def get(self, request, \*args, \*\*kwargs):

        context1 = {

            'message': 'Hello from class-based view!',

            'current\_time': datetime.now(),

            'items': ['Mango', 'Pineapple', 'Grapes'],

        }

        return render(request, 'class.html', context1)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Class-based View</title>

</head>

<body>

    <h1>{{ message }}</h1>

    <p>Current Time: {{ current\_time }}</p>

    <ul>

        {% for item in items %}

            <li>{{ item }}</li>

        {% endfor %}

    </ul>

</body>

</html>

from django.urls import path

from . import views

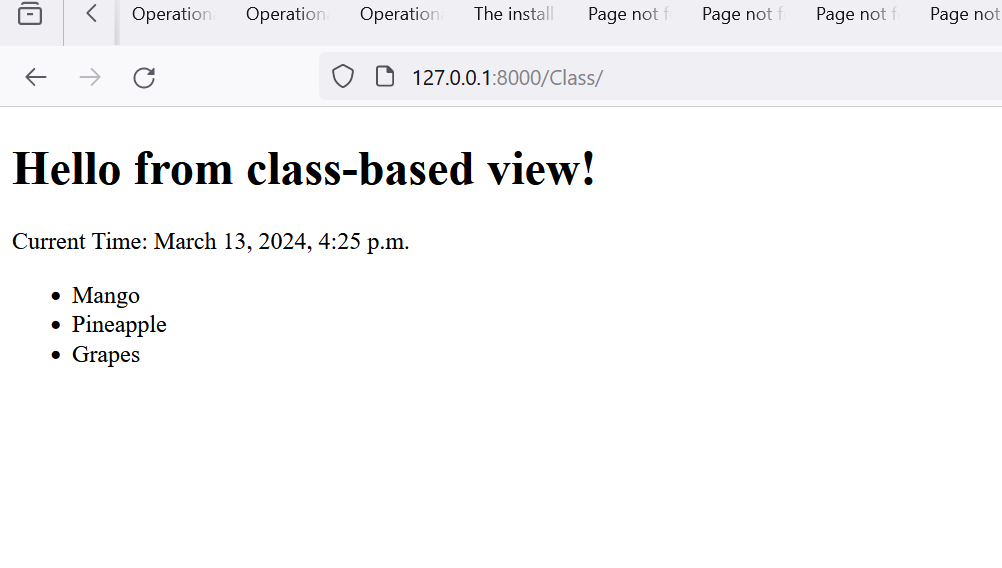
urlpatterns = [

    path('home/',views.home),

    path('Class/',views.Classbased.as\_view())

]

**Output**

****