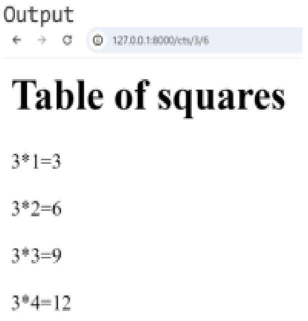
USN | | | | | | | | | | |

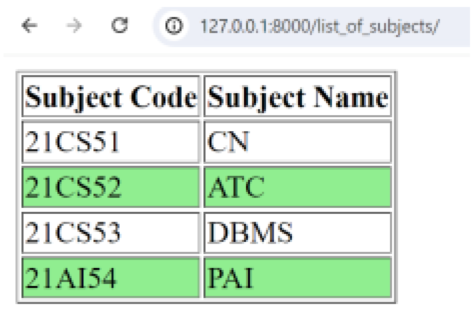
CELEBRATING 30 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test I – June-2024

Sub:	FULL STACK DEVELOPMENT				Sub Code:	21CS62	Branch:	CSE	
Date:	03/06/24	Duration:	90 mins	Max Marks:	50	Sem/Sec:	VI B&C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	a) Define a web framework and explain the purpose of using one in web development.	[5]	CO1	L1
	b) Discuss common types of errors encountered in Django web development and their causes.	[5]	CO1	L2
2	a) What are wildcard patterns in URL routing, and how are they used in Django? Give an example.	[5]	CO1	L2
	b) What is virtual environment in Django. Why it is used. Write steps/commands for the following (in Ubuntu) ● Install /activate virtual environment. ● Install Django, create project, create application and to run server.	[5]	CO1	L2
3	a) Explain the role of Django's template system in web development. Write a Django view function that renders a template with context data, and provide the corresponding template code that displays the context data using Django's template language.	[5]	CO3	L3
	b) Create a DJango application that will display the table of given numbers mentioned in the picture below. Output  127.0.0.1:8000/cts/3/6 **Table of squares** 3*1=3 3*2=6 3*3=9 3*4=12	[5]	CO1	L3

4	a) Discuss three commonly used template tags and three commonly used template filters in Django. Provide examples of how each tag and filter is used in a Django template.	[5]	CO3	L2,L3
	b) Describe the concept of template inheritance in Django. Create a base template with a header and footer, and then create a child template that extends this base template to include a content block.	[5]	CO3	L2,L3
5	a) How do you define a model in Django? Write a Django model class Book with fields title, author, published_date, and price. Explain the significance of each field type used in the model.	[5]	CO2	L2,L3
	b) Develop a Django app that displays a list of subject codes and subject names of any semester in tabular format.  127.0.0.1:8000/list_of_subjects/ Subject Code / Subject Name: 21CS51 CN; 21CS52 ATC; 21CS53 DBMS; 21AI54 PAI	[5]	CO3	L2,L3
6	Demonstrate how to perform the following CRUD (Create, Read, Update, Delete) operations using python shell: ● Create a model name Employee. ● Insert 3 employees names, emp ids, salaries. ● Retrieve all Employee records from the database. ● Update the Salary of a specific employee. ● Delete a specific employee record from the database. Provide code snippets for each operation.	[10]	CO2	L3

Subject Code	Subject Name
21CS51	CN
21CS52	ATC
21CS53	DBMS
21AI54	PAI

CI **CCI** **HOD**

USN | | | | | | | | | | |

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test I – June-2024

Sub:	FULL STACK DEVELOPMENT			Sub Code:	21CS62	Branch:	CSE	
Date:	03/06/24	Duration:	90 mins	Max Marks:	50	Sem/Sec:	VI B&C	OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	a) Define a web framework and explain the purpose of using one in web development. **Solution:** A web framework is a software framework designed to support the development of web applications, providing developers with pre-built components, libraries, and tools to streamline the process. The purpose of using a web framework is to simplify and standardize web development tasks, such as handling HTTP requests, managing databases, and generating HTML responses. It promotes code reuse, maintainability, and scalability by following established patterns and best practices.	[5]	CO1	L1
	b) Discuss common types of errors encountered in Django web development and their causes. **Solution:** ● Common errors in Django web development include: ● HTTP 404 (Page Not Found) errors: Caused by missing or misconfigured URL patterns. ● HTTP 500 (Internal Server Error) errors: Caused by server-side issues such as misconfigured settings, database connection problems, or unhandled exceptions in views. ● Database-related errors: Caused by incorrect database queries, schema mismatches, or database connection issues. ● Form validation errors: Caused by invalid user input or misconfigured form validation logic. ● These errors can occur due to misconfiguration, programming mistakes, data inconsistencies, or environmental issues.	[5]	CO1	L2
2	a) What are wildcard patterns in URL routing, and how are they used in Django? Give an example. **Solution:**	[5]	CO1	L2

	Wildcard Patterns in Django **In Django, wildcard patterns can be created using both regular expressions and path converters. These allow you to capture parts of the URL and pass them as arguments to your view functions.** **Path Converters (Django 2.0+)** **Path converters are a simpler way to define URL patterns compared to regular expressions. They allow you to specify the type of variable expected in the URL and automatically convert it. Here are some common path converters:** ● `<str:variable>`**: Matches any non-empty string, excluding the path separator (**`/`**).** ● `<int:variable>`**: Matches an integer.** ● `<slug:variable>`**: Matches a slug (letters, numbers, underscores, and hyphens).** **Example:** **from django.urls import re_path** **from . import views** **urlpatterns = [** **re_path(r'^article/(?P<id>\d+)/(?P<title>[\w-]+)/$', views.article_detail, name='article_detail'),** **]**			
	b) What is virtual environment in Django. Why it is used. Write steps/commands for the following (in Ubuntu) ● Install /activate virtual environment. ● Install Django, create project, create application and to run server. Solution: A virtual environment in Django is an isolated workspace that allows developers to manage dependencies and packages specific to a Django project without affecting the global Python environment. This isolation ensures that each project can maintain its own versions of libraries and dependencies, preventing conflicts and compatibility issues with other projects. Virtual	[5]	CO1	L2

	environments help create a consistent development setup across different machines and team members, making it easier to manage project-specific dependencies and avoiding potential issues arising from system-wide package installations. **need:** The need for a virtual environment in Django (and Python development in general) arises primarily from the requirement to manage project-specific dependencies and avoid conflicts. Each Django project might depend on different versions of libraries or packages, and installing these globally can lead to version conflicts and unpredictable behavior across projects. Virtual environments create an isolated environment for each project, allowing developers to install and manage dependencies independently. This isolation ensures consistency, makes it easier to replicate development environments across different machines, facilitates better collaboration among team members, and simplifies dependency management and deployment. Overall, virtual environments contribute to more reliable and maintainable development practices. ● **Install /activate virtual environment: python-m venv env1** ● **command of Install Django, create project: pip install django django-admin startproject ProjectName** ● **command of create application and to run server: py manage.py startapp appName**			

3	a) Explain the role of Django's template system in web development. Write a Django view function that renders a template with context data, and provide the corresponding template code that displays the context data using Django's template language.
	Solution:
	Django's template system plays a crucial role in web development by separating the presentation layer from the business logic. It allows developers to define HTML templates that can dynamically render content based on the context provided by views. This system supports template inheritance, enabling the reuse of common structures and components, which promotes DRY (Don't Repeat Yourself) principles. Additionally, Django templates offer a rich set of built-in tags and filters for manipulating data and controlling the presentation logic directly within the templates. This separation of concerns enhances code maintainability, readability, and makes it easier for developers and designers to collaborate on the frontend and backend aspects of web applications.
	
	Example:
	
	# views.py
	from django.shortcuts import render
	
	def example_view(request):
	    context = {
		[5]	CO3	L3

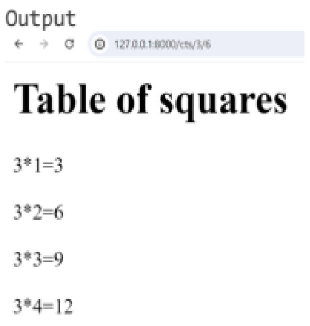
```
        'title': 'Welcome to My Website',
        'description': 'This is an example description.',
        'items': ['Item 1', 'Item 2', 'Item 3'],
    }
    return render(request, 'example_template.html', context)
```

<!-- example_template.html -->

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ title }}</title>
</head>
<body>
    <h1>{{ title }}</h1>
    <p>{{ description }}</p>

    <ul>
        {% for item in items %}
            <li>{{ item }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

	b) Create a DJango application that will display the table of given numbers mentioned in the picture below.	[5]	CO1	L3



Output

127.0.0.1:8000/cts/3/6

Table of squares

3*1=3

3*2=6

3*3=9

3*4=12

Solution:
Views.py

```
from datetime import date
from django.http import HttpResponse
from django.shortcuts import render
from django.template import Context, Template
def create_table_of_squares(request,s,n):
result=""
for i in range(1,n+1):
result+="<p>"+str(s)+"*"+str(i)+"="+str((s*i))+"</p>"
return HttpResponse(result)
```

	URLS.py from django.contrib import admin from django.urls import path, re_path from ap2.views import create_table_of_squares urlpatterns = [path('admin/', admin.site.urls), path('cts/<int:s>/<int:n>', create_table_of_squares),]			

4	a) Discuss three commonly used template tags and three commonly used template filters in Django. Provide examples of how each tag and filter is used in a Django template.
	
	Solution:
	
	Commonly Used Template Tags in Django:
	1. {% for %}
	The `{% for %}` tag is used to loop over a sequence, such as a list or a queryset. It's similar to a for loop in Python and allows you to iterate over items in a context variable.
	`<ul>`
	    `{% for item in items %}`
	        `<li>{{ item }}</li>`
	    `{% endfor %}`
	`</ul>`
	
	2. **{% if %}**
	
	The `{% if %}` tag is used to perform conditional statements within templates. It allows you to display content based on whether a condition is true or false.
	
	`{% if user.is_authenticated %}`
	
	    `<p>Welcome, {{ user.username }}!</p>`
	
	`{% else %}`
	
	    `<p>Please log in.</p>`
	
	`{% endif %}`
		[5]	CO3	L2,L3

3. **{% block %} and {% extends %}**

The `{% block %}` tag is used in conjunction with `{% extends %}` for template inheritance. `{% block %}` defines a block of content that child templates can override, while `{% extends %}` is used to inherit the layout of a base template.

```
<!-- base.html -->

<!DOCTYPE html>

<html>

<head>

  <title>{% block title %}My Website{% endblock %}</title>

</head>

<body>

  {% block content %}{% endblock %}

</body>

</html>


<!-- child.html -->

{% extends "base.html" %}


{% block title %}

  Child Page Title

{% endblock %}


{% block content %}

  <p>This is the child page content.</p>

{% endblock %}
```

Commonly Used Template Filters in Django

1. **length**

The `length` filter returns the number of items in a list or characters in a string. It's useful for displaying counts or validating lengths.

```
<p>This list has {{ items|length }} items.</p>
```

2. **default**

   The `default` filter provides a fallback value if the variable is not defined or is empty. It's useful for ensuring that templates display meaningful content even when some data might be missing.

   ```
   <p>{{ user.profile.bio|default:"This user has no bio." }}</p>
   ```

3. upper:

   Keep the data in upper letters.

   ```
   {{str| upper}}
   ```

	b) Describe the concept of template inheritance in Django. Create a base template with a header and footer, and then create a child template that extends this base template to include a content block.	[5]	CO3	L2,L3

Solution:
Template inheritance in Django allows you to create a base template that contains common elements such as headers, footers, and navigation bars. You can then create child templates that inherit from the base template and override specific blocks to insert unique content. This approach promotes code reuse and a consistent layout across your web application.

Step 1: Create the Base Template

First, create a base template that includes a header and footer. This template will define blocks that child templates can override.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}My Site{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/styles.css' %}">
</head>
<body>
    <header>
        <h1>Welcome to My Site</h1>
        <nav>
            <ul>
                <li><a href="/">Home</a></li>
                <li><a href="/about/">About</a></li>
                <li><a href="/contact/">Contact</a></li>
```

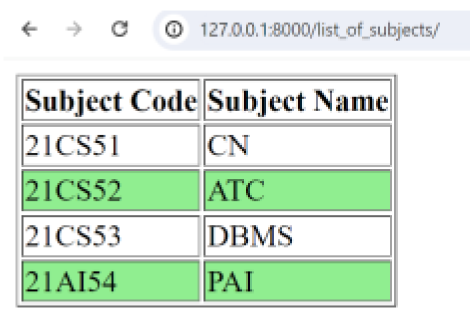
	</ul> </nav> </header> <main> {% block content %} <!-- Default content can go here if needed --> {% endblock %} </main> <footer> <p>&copy; 2024 My Site. All rights reserved.</p> </footer> </body> </html> **Step 2: Create a Child Template** Next, create a child template that extends the base template. This template will override the content block to include specific content. {% extends "base.html" %} {% block title %}Home Page{% endblock %} {% block content %} <h2>Home Page Content</h2> <p>Welcome to the home page of my site. Here is some introductory content.</p> {% endblock %} **Usage in Views** To render these templates in your Django views, you would use the `render` function. For example: from django.shortcuts import render def home_view(request): return render(request, 'child.html')			
5	a) How do you define a model in Django? Write a Django model class Book with fields title, author, published_date, and price. Explain the significance of each field type used in the model. **Solution:** To define a model in Django, you create a class in `models.py` that inherits from `models.Model`. Each attribute of the class represents a database field. from django.db import models class Book(models.Model): title = models.CharField(max_length=200) # Title of the book	[5]	CO2	L2,L3

```
    author = models.CharField(max_length=100)        # Author's name
    published_date = models.DateField()              # Date of publication
    price = models.DecimalField(max_digits=10, decimal_places=2)  # Price of the
book
```

Field Types Explained

- **`CharField`**: Used for short text fields. `max_length` specifies the maximum number of characters.
    - `title`: `max_length=200`
    - `author`: `max_length=100`
- **`DateField`**: Used for date values (e.g., `published_date`).
- **`DecimalField`**: Used for precise decimal numbers.
    - `max_digits=10`: Total number of digits.
    - `decimal_places=2`: Number of digits after the decimal point.
    - `price`: Suitable for storing prices.

	Question	Marks	CO	Level
	b) Develop a Django app that displays a list of subject codes and subject names of any semester in tabular format.	[5]	CO3	L2,L3



127.0.0.1:8000/list_of_subjects/

Subject Code	Subject Name
21CS51	CN
21CS52	ATC
21CS53	DBMS
21AI54	PAI

Solution:

Views.py

```
from datetime import date
from django.http import HttpResponse
from django.shortcuts import render
from django.template import Context, Template
def list_of_subjects(request):
s1={"scode":"21CS51","sname":"cn"}
s2={"scode":"21CS52","sname":"ATc"}
s3={"scode":"21CS53","sname":"DbMS"}
s4={"scode":"21AI54","sname":"PAI"}
l=list()
l=[s1,s2,s3,s4]
```

```
return render(request,'list_of_subjects.html',{"l":l})
```

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path
urlpatterns = [
path('list_of_subjects/', list_of_subjects),
]
```

Template file: list_of_subjects.html

```
<html>
<body>
<table border>
<tr>
<th>Subject Code</th>
<th>Subject Name</th>
</tr>
{% for subject in l %}
{% if forloop.counter|divisibleby:"2" %}
<tr>
<td style="background-color: lightgreen;">{{ subject.scode }}</td>
<td style="background-color: lightgreen;">{{ subject.sname|upper

}}</td>
</tr>
{% else %}
<tr>
<td>{{ subject.scode }}</td>
<td>{{ subject.sname|upper }}</td>
</tr>
{% endif %}
{% endfor %}
</table>
</body>
</html>
```

6	Demonstrate how to perform the following CRUD (Create, Read, Update, Delete) operations using python shell:	[10]	CO2	L3

- Create a model name Employee.
- Insert 3 employees names, emp ids, salaries.
- Retrieve all Employee records from the database.
- Update the Salary of a specific employee.
- Delete a specific employee record from the database.

Provide code snippets for each operation.

Solution:

1. **Create class Employee in models.py**

```
 class Employee(models.Model):
        name=models.CharField(max_length=100)
        id=models.IntegerField()
        salary=models.IntegerField()
```

2. obj1 = Employee(name = ‘A’, id = 101, salary = 20000)

```
obj2 = Employee(name = ‘B’, id = 102, salary = 250000)
obj3 = Employee(name = ‘C’, id = 103, salary = 150000)
list = [obj1, obj2,obj3]
for i in list:
   i.save()
```

3. **Employee.objects.all().values()**

4. Update: Update the salary of a specific employee

```
# Retrieve the employee
employee_to_update = Employee.objects.get(id=102)

# Update the salary
employee_to_update.salary = 65000.00
employee_to_update.save()

# Verify the update
updated_employee = Employee.objects.get(id=102)
print(updated_employee.name, updated_employee.salary)
```

5. Delete: Delete a specific employee record

```
# Retrieve the employee
employee_to_delete = Employee.objects.get(id=103)

# Delete the employee
employee_to_delete.delete()
```

	# Verify deletion
	remaining_employees = Employee.objects.all()
	for employee in remaining_employees:
	    print(employee.name, employee.id, employee.salary)			