# Module 9: Python DB and Framework

## 1. HTML in Python

**Q-1]** Introduction to embedding HTML within Python using web frameworks like Django or Flask.

- Python web frameworks like **Django** and **Flask** allow embedding HTML using templates.
- HTML files are placed inside a **templates folder**, and Python views render them with dynamic data.
- For Example in Django, HTML is not written directly inside Python code.
- Django uses a **template system** that lets you embed **dynamic data** into HTML pages using **template tags and variables**.
- The view (Python function) sends data to the template, which is then rendered into HTML and displayed to the user.

🔹 **Example: Embedding HTML in Django using Templates**

**Step 1: Create a Django view:-**

```
# myapp/views.py
from django.shortcuts import render
def home(request):
    # Python data
    context = {
        'title': 'Welcome to Django',
        'username': 'Alice'
    }
    # Render the HTML template with the data
    return render(request, 'home.html', context)
```

**Step 2: Create a URL pattern (urls.py)**

```
# myapp/urls.py
from django.urls import path
from . import views
urlpatterns = [
    path('', views.home, name='home'),
```

]

**Step 3: Create the HTML Template (templates/home.html)**

```
<!DOCTYPE html>
<html>
<head>
   <title>{{ title }}</title>
</head>
<body>
   <h1>{{ title }}</h1>
   <p>Hello, {{ username }}! Welcome to your first Django web page.</p>
</body>
</html>
```

- **Output:-**

Welcome to Django

Hello, Alice! Welcome to your first Django web page.

**Q-2]** Generating dynamic HTML content using Django templates.

- Django templates use **template tags** and **variables** to generate dynamic HTML.

➢ Example:

```
<h1>Welcome, {{ user.username }}!</h1>
{% if user.is_authenticated %}
   <p>You are logged in.</p>
{% endif %}
```

- The view passes a context dictionary:

```
return render(request, 'index.html', {'user': user})
```

## 2. CSS in Python

**Q-3]** Integrating CSS with Django templates.

- CSS files are placed in the **static** directory.
- Load static files in the template:

➢ **Example:-**

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

**Q-4]** How to serve static files (like CSS, JavaScript) in Django.

➢ Add 'django.contrib.staticfiles' in **INSTALLED_APPS**.
➢ Define STATIC_URL and STATICFILES_DIRS in **settings.py**:
➢ STATIC_URL = '/static/'
➢ STATICFILES_DIRS = [BASE_DIR / "static"]
➢ Use {% load static %} in templates.

## 3. JavaScript with Python

**Q-5]** Using JavaScript for client-side interactivity in Django templates.

- JavaScript enhances interactivity (form validation, AJAX calls).
- Include scripts in templates:

```
<script>
  function greet() {
    alert("Hello from JavaScript!");
  }
</script>
<button onclick="greet()">Click Me</button>
```

**Q-6]** Linking external or internal JavaScript files in Django.

➢ Place JS files in the static directory:
➢ {% load static %}
➢ <script src="{% static 'js/script.js' %}"></script>

## 4. Django Introduction

**Q-7]** Overview of Django: Web development framework.

 Django is a **high-level Python web framework** for rapid development.

 It follows the **MVT architecture** and includes tools for ORM, admin, authentication, and templating.

**Q-8]** Advantages of Django (e.g., scalability, security).

 Scalable and secure

 Rapid development

 Built-in ORM and admin

 Automatic URL routing and form handling

 Reusable apps

**Q-9]** Django vs. Flask comparison: Which to choose and why.

| **Feature** | **Django** | **Flask** |
|---|---|---|
| Type | Full-stack framework | Micro-framework |
| Structure | MVT (Model-View-Template) | Flexible, minimal |
| Use case | Large projects | Small/simple apps |
| Built-in | Admin | Yes No |
| ORM | Yes | Optional |

## 5. Virtual Environment

**Q-10]** Understanding the importance of a virtual environment in Python projects.

- Keeps dependencies **isolated** per project.
- Prevents version conflicts between packages.
- Makes projects reproducible across systems.

**Q-11]** Using venv or virtualenv to create isolated environments.

- python -m venv env

source env/bin/activate  # Linux/Mac

env\Scripts\activate  # Windows

## 6. Project and App Creation

**Q-12]** Steps to create a Django project and individual apps within the project.

- django-admin startproject myproject
- cd myproject
- python manage.py startapp myapp
- Add the app to INSTALLED_APPS in settings.py.

**Q-13]** Understanding the role of manage.py, urls.py, and views.py.

- **manage.py**: Command-line utility for running and managing the project.
- **urls.py**: Maps URLs to specific views (routing).
- **views.py**: Contains Python functions/classes that handle HTTP requests.

## 7. MVT Pattern Architecture

**Q-14]** Django's MVT (Model-View-Template) architecture and how it handles request-response cycles.

- **Model**: Handles database interaction.
- **View**: Contains business logic.
- **Template**: Defines HTML presentation.
- **Request-response cycle:**
- Request → URL → View → Model → Template → Response

## 8. Django Admin Panel

**Q-15]** Introduction to Django's built-in admin panel.

- Automatically created for managing database models.
- Accessible at /admin.
- Requires superuser creation:
- python manage.py createsuperuser

**Q-16]** Customizing the Django admin interface to manage database records.

- Register models and customize display:

```
from django.contrib import admin
from .models import Product
class ProductAdmin(admin.ModelAdmin):
    list_display = ('name', 'price')
admin.site.register(Product, ProductAdmin)
```

## 9. URL Patterns and Template Integration

**Q-17]** Setting up URL patterns in urls.py for routing requests to views.

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

**Q-18]** Integrating templates with views to render dynamic HTML content.

```
def home(request):
    return render(request, 'home.html', {'name': 'Alice'})
```

- **Template (home.html):**

```
<h1>Hello, {{ name }}</h1>
```

## 10. Form Validation using JavaScript

**Q-19]** Using JavaScript for front-end form validation.

```
<form onsubmit="return validateForm()">
  <input id="email" type="email" required>
  <button type="submit">Submit</button>
</form>
<script>
function validateForm() {
    let email = document.getElementById("email").value;
    if (!email.includes("@")) {
        alert("Invalid email!");
        return false;   }   }
</script>
```

## 11. Django Database Connectivity (MySQL or SQLite)

**Q-20]** Connecting Django to a database (SQLite or MySQL).

- **settings.py:**

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

- **For MySQL:**

```
'ENGINE': 'django.db.backends.mysql',
'NAME': 'dbname',
'USER': 'root',
'PASSWORD': 'password',
'HOST': 'localhost',
'PORT': '3306',
```

**Q-21]** Using the Django ORM for database queries.

```
from myapp.models import Student
Student.objects.all()
Student.objects.filter(name="John")
Student.objects.create(name="Jane", age=20)
```

## 12. ORM and QuerySets

**Q-22]** Understanding Django's ORM and how QuerySets are used to interact with the database.

- ORM (Object Relational Mapper) lets developers interact with the database using **Python objects**.
- **QuerySets** are collections of model instances:
- students = Student.objects.filter(age__gte=18)

## 13. Django Forms and Authentication

**Q-23]** Using Django's built-in form handling.

```
from django import forms
```

```
class ContactForm(forms.Form):
    name = forms.CharField()
    email = forms.EmailField()
```

**Q-24]** Implementing Django's authentication system (sign up, login, logout, password management).

- Use Django's built-in authentication:
- from django.contrib.auth import authenticate, login, logout
- Includes views for **signup**, **login**, **logout**, **password reset**, etc.

## 14. CRUD Operations using AJAX

**Q-25]** Using AJAX for making asynchronous requests to the server without reloading the page.

- Example:-

```
$.ajax({
  url: '/add/',
  type: 'POST',
  data: {name: 'John'},
  success: function(response) {
     alert('Added successfully!');  }
});
```

- The server returns JSON responses handled without page reload.

## 15. Customizing the Django Admin Panel

**Q-26]** Techniques for customizing the Django admin panel.

- Use ModelAdmin options:
    - list_display, search_fields, list_filter
    - Add custom forms and inline models.

## 16. Payment Integration Using Paytm

**Q-27]** Introduction to integrating payment gateways (like Paytm) in Django projects.

- Use **Paytm Developer API** for secure payments.
- Steps:

1. Register for merchant account.
2. Generate credentials (Merchant ID, Key).
3. Create payment view in Django.
4. Handle callback and transaction status.

## 17. GitHub Project Deployment

**Q-28]** Steps to push a Django project to GitHub.

- git init
- git add .
- git commit -m "Initial commit"
- git branch -M main
- git remote add origin https://github.com/username/repo.git
- git push -u origin main

## 18. Live Project Deployment (PythonAnywhere)

**Q-29]** Introduction to deploying Django projects to live servers like PythonAnywhere.

➢ Steps for deploying Django project:-

1. Create an account on PythonAnywhere.
2. Upload project or clone from GitHub.
3. Configure WSGI file and virtual environment.
4. Set up static files and run migrations.
5. Reload web app.

## 19. Social Authentication

**Q-30]** Setting up social login options (Google, Facebook, GitHub) in Django using OAuth2.

- Use **django-allauth**:
- pip install django-allauth
- Add to INSTALLED_APPS and configure providers like Google or GitHub in **settings.py**.

## 20. Google Maps API

**Q-31]** Integrating Google Maps API into Django projects.

- Include Google Maps API script:

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY"></script>
```

- Initialize map in template:

```
<div id="map" style="height:400px"></div>
<script>
   var map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: 28.6139, lng: 77.2090},
      zoom: 10
   });
</script>
```

---