React- Module5

Assignment

Q1. Why Django should be used for web-development?

Explain how you cancreate a project in Django?

Ans. Django is a powerful and high-level web framework written in Python, designed to make the process of building web applications easier, faster, and more secure. Here are several reasons why Django is widely used for web development:

Rapid Development:

Django’s design encourages rapid development by providing a clean and pragmatic approach. It comes with a lot of built-in functionality, which reduces the amount of code developers need to write. It has features like authentication, URL routing, and database handling already included.

Batteries Included:

Django follows the "batteries-included" philosophy, meaning it comes with a wide array of features like ORM (Object Relational Mapper), admin panel, authentication, and sessions, which eliminates the need for third-party packages in many cases.

Security:

Django has strong security features built in, helping developers avoid common security pitfalls such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking. Django’s user authentication system provides a secure way to manage user accounts and passwords.

Scalability:

Django is highly scalable, supporting both small-scale and large-scale projects. It can handle high traffic and is used by large organizations such as Instagram, Pinterest, and Mozilla.

Versatile:

Django can be used to build all kinds of websites—from content management systems (CMS) and social networks to scientific computing platforms and enterprise-level applications.

DRY Principle (Don't Repeat Yourself):

Django encourages reusability of code and adherence to the DRY principle. This leads to cleaner, more maintainable, and efficient code.

Community and Documentation:

Django has a large and active community that provides extensive documentation, tutorials, and third-party packages, making it easier to learn and implement.

Q2. How to check installed version of django?

Ans. To check the installed version of Django, you can use any of the following methods:

1. Using the Command Line

Open a terminal or command prompt and run:

python -m django –version

2. Using Python Shell

You can also check the version directly from a Python shell: python

Then type: import django

print(django.get\_version())

**3. Using pip List**

To view the version along with other installed packages, run: pip show Django

Q3. Mention what command line can be used to load data into

Django?

Ans. To load data into Django from a fixture (a file containing serialized data), you can use the following command line: python manage.py loaddata <fixture\_name>

Q4. Explain what does django-admin.py make messages

command is usedfor?

Ans. The django-admin.py makemessages command is used in Django's internationalization (i18n) framework. It helps create or update translation files (message files) for your project to support multiple languages.

django-admin.py makemessages -l <language\_code>

Q5. What is .Django URLs?make program to create django urls.

Ans. In Django, a URL (Uniform Resource Locator) is a way to map a URL to a specific view function that handles the request and returns a response. Django URLs are used to define the structure of URLs in a Django project. They are an essential part of Django's MTV (Model-Template-View) architecture.

Eg: # myproject/urls.py

from django.urls import path

from . import views

urlpatterns = [

path('', views.homepage, name='homepage'),

path('about/', views.about, name='about'),

]

Q6. What is a QuerySet?Write program to create a new

Post object indatabase.

Ans. In Django, a QuerySet is a collection of objects from a database that can be filtered, ordered, and manipulated using various methods. QuerySets are used to retrieve data from the database and are a fundamental part of Django's ORM (Object-Relational Mapping) system.

Eg: # models.py

from django.db import models

class Post(models.Model):

title = models.CharField(max\_length=255)

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

# views.py

from django.shortcuts import redirect

from .models import Post

def create\_post(request):

if request.method == 'POST':

title = request.POST['title']

content = request.POST['content']

post = Post.objects.create(title=title, content=content)

return redirect('post\_detail', pk=post.pk)

return render(request, 'create\_post.html')

Q7. Mention what command line can be used to load data into

Django?

Ans. To load data into Django, you can use the following command-line tools: undefined

Q8. Explain what does django-admin.py make messages

command is usedfor?

Ans. The django-admin.py makemessages command is used to extract translatable strings from your Django project's source code and create or update message files for translation. It searches for files to translate and invokes the (x)gettext utility to extract the translatable strings.

The makemessages command is usually used in conjunction with the compilemessages command, which compiles the translated message files into a format that can be used by Django.

Q9. Make Django application to demonstrate following things o There will be2 modules(Admin,Product manager)

o Admin can add product name (ex.Product id and product name) ex. (1, Samsung), (2, Apple)...etc. Datashould store in Product\_mst table with product id as primary key

o Admin canadd product subcategory details Like (Product price, product image,

o Product model, product Ram) data should store in Product\_sub\_cat table

o Admin can get product name as foreign key from product\_mst table in product\_sub\_category\_details page o Admin can view, update and delete all registered details of product o product manager can search product on search bar and get alldetails about product.

Ans. Step 1: Create a new Django project and app:

django-admin.py startproject product\_management

cd product\_management

python manage.py startapp product

Step 2: Define the models:

from django.db import models

class ProductMst(models.Model):

product\_id = models.AutoField(primary\_key=True)

product\_name = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return self.product\_name

class ProductSubCat(models.Model):

product = models.ForeignKey(ProductMst, on\_delete=models.CASCADE)

product\_price = models.DecimalField(max\_digits=10, decimal\_places=2)

product\_image = models.ImageField(upload\_to='product\_images')

product\_model = models.CharField(max\_length=255)

product\_ram = models.CharField(max\_length=255)

def \_\_str\_\_(self):

return f"{self.product.product\_name} - {self.product\_model}"

Step 3: Create the database tables:

python manage.py makemigrations

python manage.py migrate

Step 4: Create the admin interface:

from django.contrib import admin

from .models import ProductMst, ProductSubCat

admin.site.register(ProductMst)

admin.site.register(ProductSubCat)