Name: Kashyap Sorathiya

Batch: 06_Sept_Python

Assignment Module 5 (DB and Python Framework)

- Why Django should be used for web-development? Explain how you can create a project in Django?

Ans.

- Django is a full-stack web framework that is open-sourced and follows an MVT (Model View Template) type of architecture. It is comprised of a set of components and modules that aids in faster development and is used by a few of the world's top companies like Instagram, Mozilla, Spotify, Quora, YouTube, Reddit, Pinterest, Dropbox, Google, etc. It simplifies development to a huge extent, provides inbuilt and up-to-date security features, suitable for any type of project, implements DRY (don't repeat yourself) and KISS (keep it simple and short), contains support for REST APIs and massive community support.
- **Steps for creating a new project:-**

1. Open **cmd**
2. activate Virtual Env
3. Go back to the Base Directory **cd..**
4. Run this code **django-admin startproject NewProject**
5. Open the NewProject folder **cd NewProject**
6. Run to open VC code **code .**

- How to check installed version of django?

Ans.

- To check the Django version run this code in cmd. Code- **django-admin version**

- Explain what does django-admin.py make messages command is used for?

Ans.

- This command line executes over the entire source tree of the current directory and abstracts all the strings marked for translation. It makes a message file in the locale directory.

- What are Django URLs? Make program to create Django URLs.

Ans.

- A URL is a web address. For example, google.com is also a URL. In Django, we use URLconfig, which is a set of patterns that Django will try to match the requested URL to find the correct view.

– URL file

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

– Now creating a view for **login.html**

```
def login(request):
    return render(request, 'login.html')
```

– Creating a URL for **login.html**

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', views.login),
]
```

- What is a Query Set? Write program to create a new Post object in database:

Ans. A Query Set is a collection of database query results in Django. It allows you to retrieve, filter, and manipulate data from the database. It is used to perform read and write operations on the database using the Django ORM (Object-Relational Mapping) system.

- Mention what command line can be used to load data into Django?

Ans. For example, if you have a file named **mydata.json** in the directory of your app, you can load the data using the following command:

>> python manage.py loaddata mydata.json

- Explain what does django-admin.py make messages command is used for?

Ans. The **django-admin.py make messages** command is used to create or update message files for translation. Message files contain the translatable strings in your Django application. This command searches your application for translatable strings and extracts them into a message file, which can then be translated into different languages.

- Make Django application to demonstrate following things:
  There will be 2 modules (Admin,Product manager)
  Admin can add product name (ex.Product id and product name) ex. (1, Samsung), (2, Apple)...etc. Data should store in?

Ans. In the Admin module, we can create a form to add a new product to the database. This form will include fields for the product ID and product name. We will store the data in the Product model.

```
from django.db import models

class Product(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=255)
```

In admin.py, we can register the Product model with the Django admin site, so that an admin user can add new products from the admin interface:

```
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```

In the Product Manager module, we can create a view that displays all the products in the database, and a search form that allows the product manager to search for products by name or ID.

In the product_list view in views.py:

```
def product_list(request):
    data = request.GET.get('d')
    if data:
        products = Product.objects.filter(name__icontains=data) |
Product.objects.filter(id=data)
    else:
        products = Product.objects.all()
    return render(request, 'product_list.html', {'products': products, 'data':
data})
```

In product_list.html, we can display the list of products and the search form:

```
<h1>Products</h1>
  <form method="get">
    <input type="text" name="q" value="{{query}}" placeholder="Search
Products...">
    <button type="submit">Search</button>
  </form>
  <ul>
    {% for product in products %}
      <li>{{ product.id }} - {{ product.name }}</li>
```

```
    {% endfor %}
  </ul>
```

- Product_mst table with product id as primary key

Admin can add product subcategory details Like (Product price, product image, Product model, product Ram) data should store in Product_sub_cat table

Admin can get product name as foreign key from product_mst table in product_sub_category_details page Admin can view, update and delete all registered details of product manager can search product on search bar and get all details about product.

Ans. In models.py, we can define the Product and ProductSubcategory models

```
from django.db import models

class Product(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=255)

class ProductSubcategory(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    image = models.ImageField(upload_to='products')
    model = models.CharField(max_length=255)
    ram = models.CharField(max_length=255)
```

In admin.py, we can register the Product and ProductSubcategory models with the Django admin site, so that an admin user can add new products and their subcategories from the admin interface:

```
from django.contrib import admin
from .models import Product, ProductSubcategory

class ProductSubcategory(admin.ModelAdmin):
    model = ProductSubcategory

class ProductAdmin(admin.ModelAdmin):
    data = [ProductSubcategoryInline]

admin.site.register(Product, ProductAdmin)
admin.site.register(ProductSubcategory)
```

In the Product Manager module, we can create a view that displays all the products and their subcategories in the database, and a search form that allows the product manager to search for products by name or ID.

The product_list view in views.py:

```
from django.shortcuts import render
from .models import Product

def product_list(request):
    data = request.GET.get('d')
    if data:
        products = Product.objects.filter(name__icontains=data) |
Product.objects.filter(id=data)
    else:
        products = Product.objects.all()
    return render(request, 'product_list.html', {'products': products, 'data':
data})
```