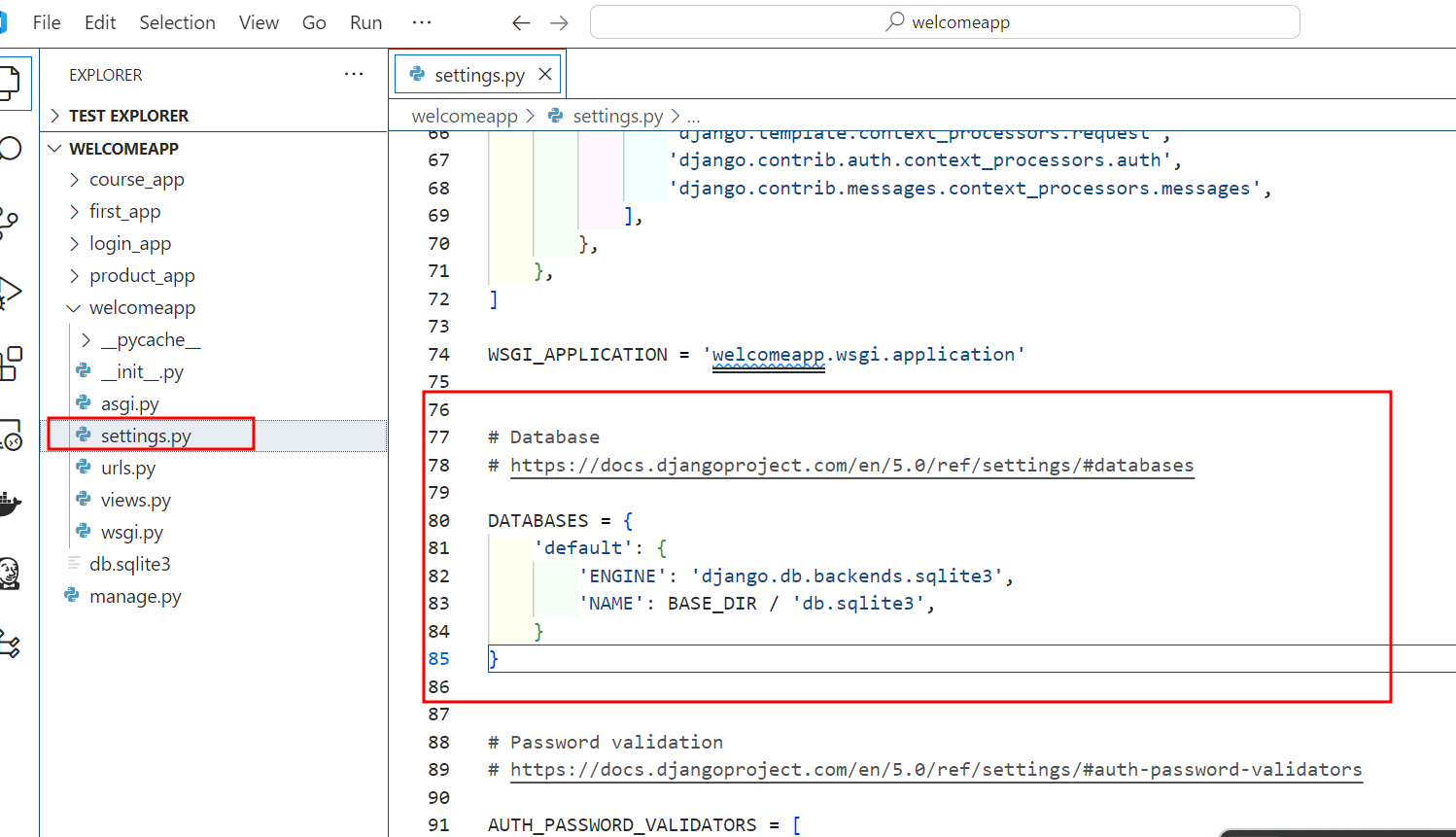
Day 3 – 16 Aug 2024

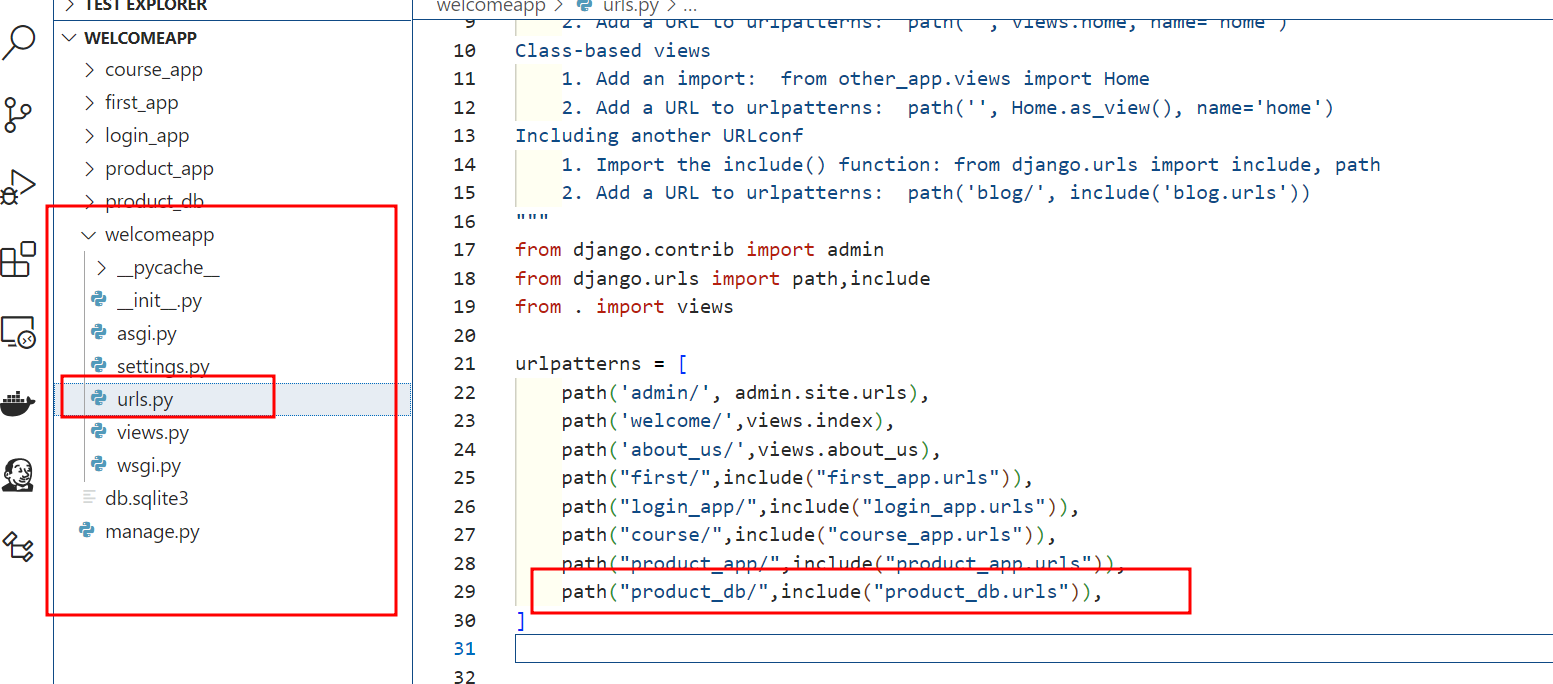
Django with Database.

By default Django provide SQLite database configuration.

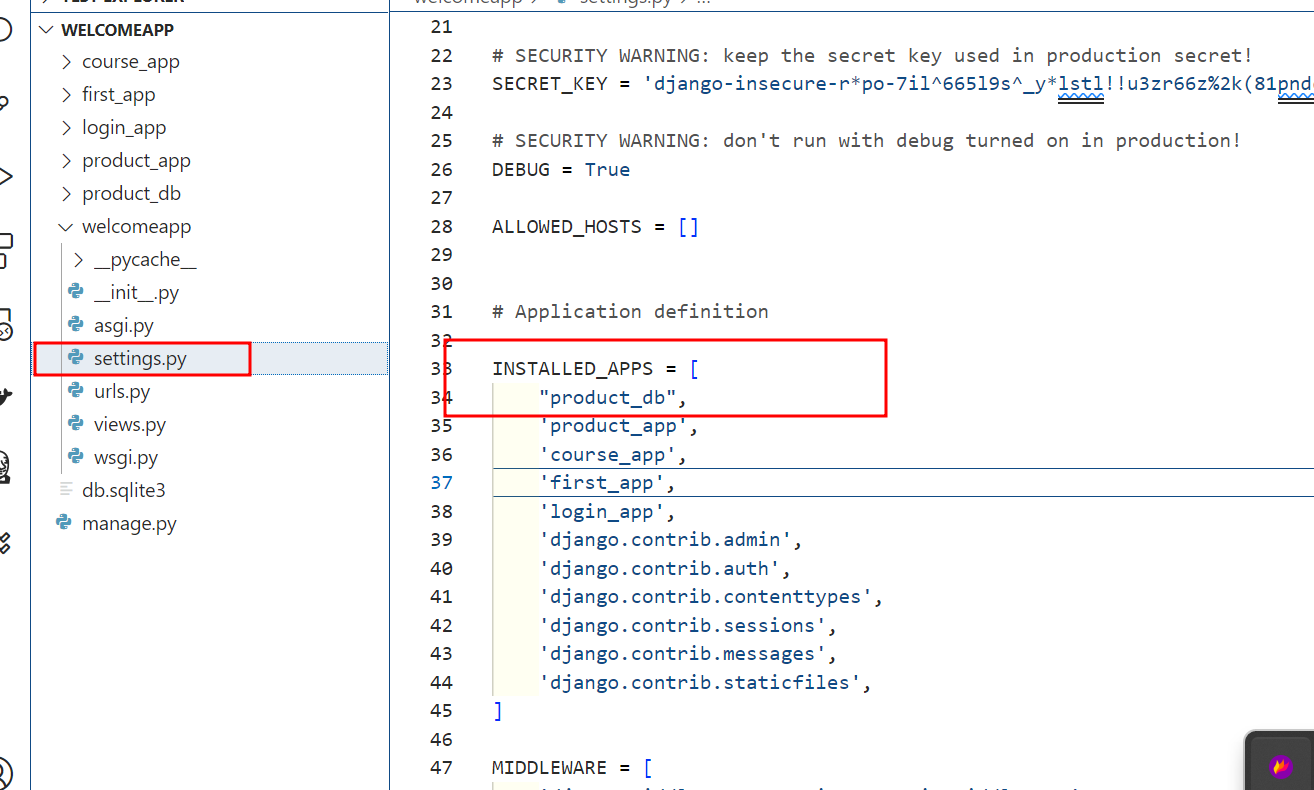


python manage.py startapp product\_db

please provide product\_db module or app details in urls.py file in main project.



Now in settings.py file provide product\_db app or modules details.



MVT : View, Template

Model : In Django, A model is a fundamental component of the Django framework that represents and manage the data in our application. Model is layer of MVT architecture. Models are python classes that define the structure of our data and provide an abstract layer for interacting with database by default database is SQLite.

Create the model

from django.db import models

# Create your models here.

class Product(models.Model):

    product\_id=models.AutoField   # auto increment

    product\_name=models.CharField(max\_length=100)

    product\_price=models.FloatField

Then we need to run two command ie makemigrations and migrate

python manage.py makemigrations

python manage.py migrate

now we will do some operation using admin dashboard

so we will create admin user.

python manage.py createsuperuser

connection postgres database

open the terminal

psql -U postgres -h localhost

password :

after connected postgres database

if you want to create new database please run below command

create database databasename

\c databasename

\dt this command is use to view all relation or table present in your database ie mydb

So if we want to connect postgres database using Django we need to install postgres engine ie dependency

Please run the command as

pip install psycopg2 this command is use to install third party module to connect postgres database.

pip list this command is use to check all modules or dependencies present in current project

now we need to provide postgres database details in settings.py file

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql\_psycopg2',

        'NAME': 'mydb',

        'USER':'postgres',

        "PASSWORD":'postgres',

        'HOST':'127.0.0.1',

        'PORT':'5432'

    }

}

Then run the command as makemigrations and migrate

python manage.py makemigrations

python manage.py migrate

if we use MVT design pattern. To display the record or insert the records we are using normal html or Django template. Django Template tightly coupled with Django Model and View.

Angular Framework, React JS or Vue JS : Framework or Library

Web Service : Giving the service for web application when more than one application running using different technologies.

Django or Java or asp.net

Amazon -🡪 payment ----🡪 Paytml Django

Paypal Java

Gpay Php

Net banking .net

Credit card etc

Application 1 XML/JSON Application 2

Django Java

XML : eXtensible Markup language

JSON : JavaScript Object Notation

Types of Web Service

2 types

1. SOAP base web service: Simple Object Access Protocol. In SOAP web service we can consume as well as produce the data only in the form of xml.
2. Rest full web service : In Rest full web service we can consume as well as produce the data in any other format like xml, json, plain text, html etc.

p1=Product(id=100,pname=”TV”,price=56000); python object

p1 object in python format other technologies can’t understand.

Product p2 = new Product(101,”Computer”,34000);

p2 object in java format other technologies can’t understand.

Xml format

<Product>

<id>100</id>

<name>TV</name>

<price>56000</price>

</Product>

Json format : {“id”:101,”pname”:”Computer”,”price”:34000}

We create few rest api. These rest api generally use http protocol methods like

Get : get the resource like select clause: get all product information

Post : post to create the resource like insert query means store record in database.

Put : update the existing the resource like update query

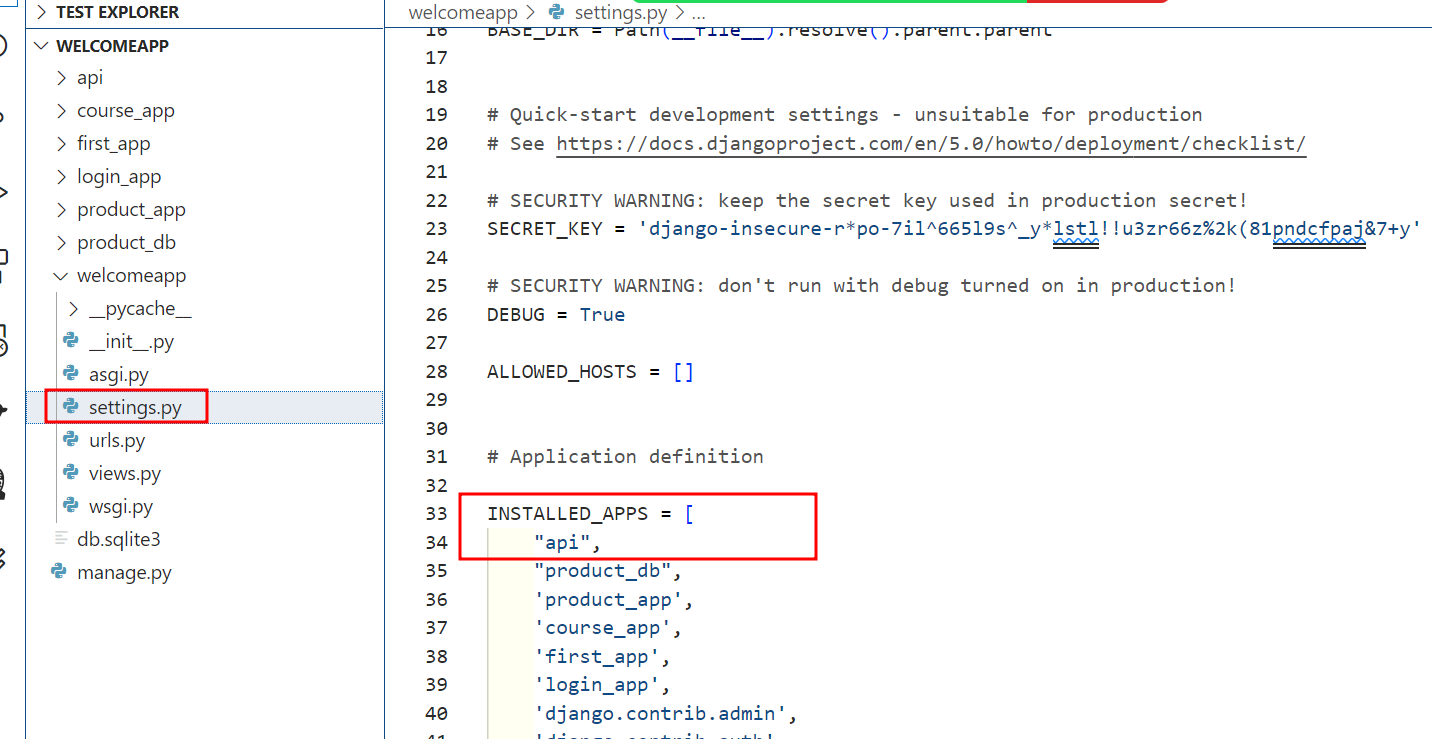
Delete : delete the resource like delete query

We will create api

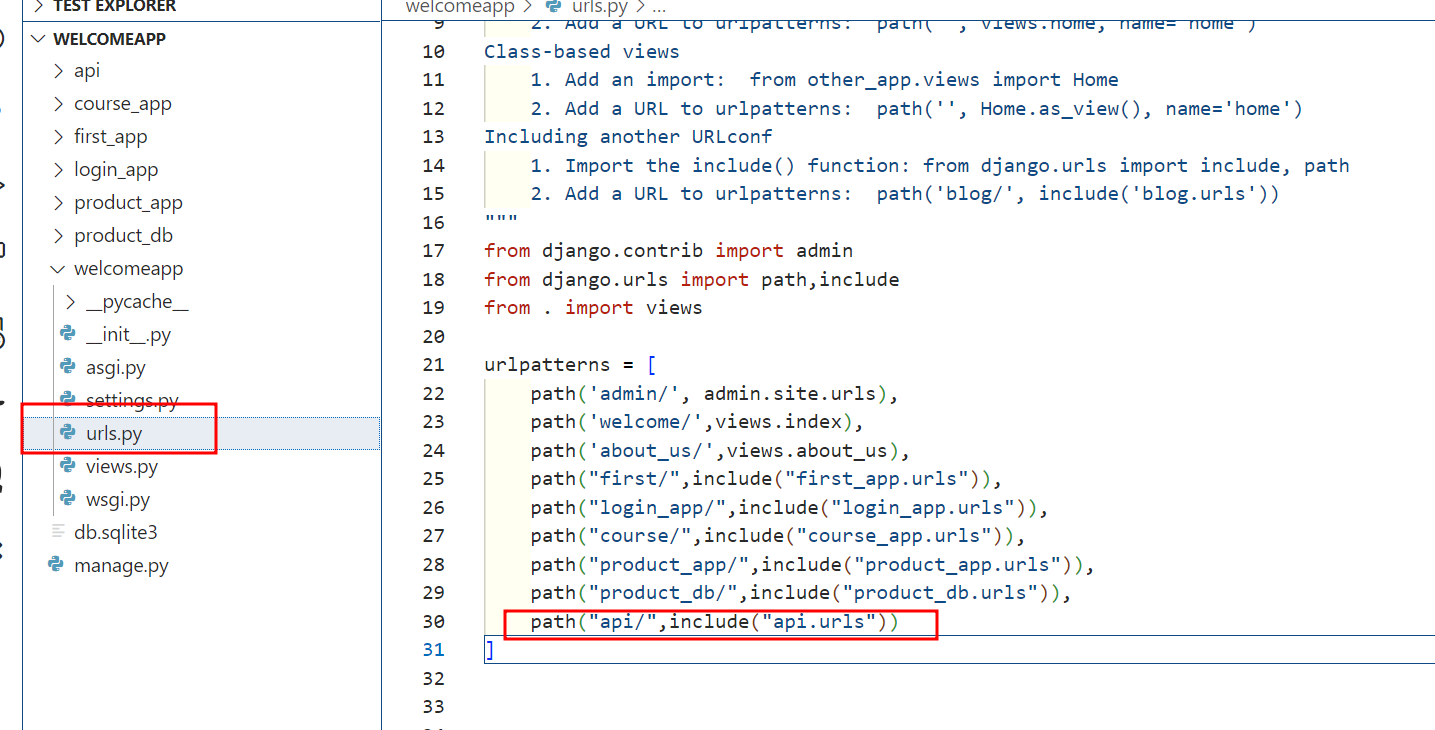
In same project create new app or module using below command as

python manage.py startapp api

in main project inside settings.py file provide new app ie api details in installed app section.



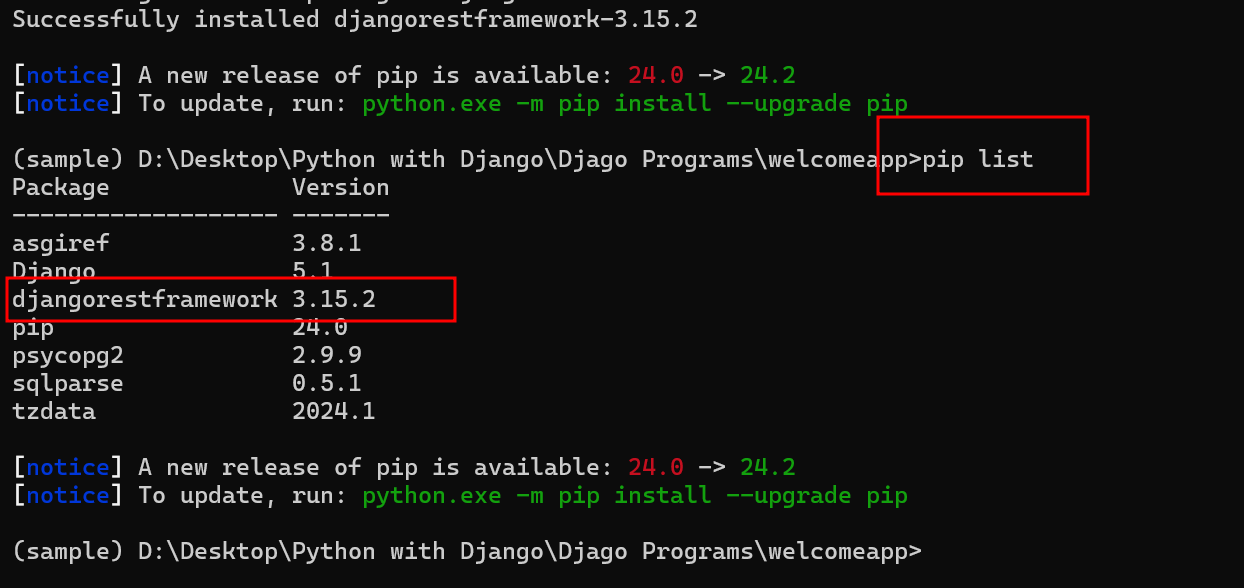
Now we api module url information in main project urls.py file provide it.



If we want to create rest api using Django framework we need to install third party module ie

pip install djangorestframework

pip list



Now we work on Model. Model class bind with database table.

from django.db import models

# Create your models here.

class Department(models.Model):

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

    department\_name=models.CharField(max\_length=200)

    class Meta:

        db\_table="department"

    def \_\_str\_\_(self):

        return self.department\_name

class Employee(models.Model):

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

    employee\_name=models.CharField(max\_length=200)

    employee\_salary=models.FloatField()

    department\_id=models.ForeignKey(Department,on\_delete=models.CASCADE)

    class Meta:

        db\_table="employee"

    def \_\_str\_\_(self):

        return self.employee\_name+", working in "+self.department\_id.department\_name

Now please create urls.py file without url rules

Then execute the command as makemigrations and migrate

python manage.py makemigrations

python manage.py migrate

Then in postgres sql database check employee and department table created or not.

In in admin.py file please register employee and department table. So using admin console we can do some operation on employee as well as department table.

**admin.py** file code

from django.contrib import admin

from .models import Employee,Department

# Register your models here.

admin.site.register(Employee)

admin.site.register(Department)

Now we will create Serializers

Serializers basically help use to convert complex data ie python object or model object into native python data types. That data easily we can render to view technologies ie angular or react or any rest client in the form of xml or json depending upon client requirements.

So now we need to create serializers.py file. Which contains Serializers classes for Employee and Department.