Python Programming Projects

A. Shiva Kiran

inligntech

List of Projects

Files :server.py

Chapter 6:Compress

Files :compress.py

Files :compressmodule.py

Files :decompress.py

Files :gui.py

Files :gui_with_filedialog.py

Chapter 7:CreditCardValidator

Files :demo.py

Chapter 8:CRUD_Commandline

Files :crud.py

Files :text.py

Chapter 9:CRUD_Databases_Student_management_system

Files :demo_mysql.py

Files :demo_mysqlserver.py

Files :demo_tkinter_mssqlserver.py

Files :demo_tkinter_mysql.py

Chapter 10:CRUD_pyqt6

Files :demo.py

Chapter 11:Custom_Color_Picker

Files :demo.py

Chapter 12:CvGenerator_app

Files :demo.py

Chapter 13:Developer Survey Data Analysis Project With Python & Pandas

Chapter 14:Django_Booksite

Chapter 15:Django_todo_app

Chapter 16:Flask-Site

Files :app.py

Files :test.py

Chapter 17:FlaskProject_ExpenseManager

Files :app.py

Chapter 18:Generate_PDF_with_Python

Files :demo.py

Files :demo2.py

Files :demo_create_pdf.py

Files :demo_csv.py

Files :demo_links.py

Files :test.py

Chapter 19:icons

Chapter 20:InvoiceGenerator_PDF

Files :demo.py

Chapter 21:opencv

Files :demo.py

Files :demo_blurring_Averaging.py

Files :demo_drawcircle.py

Files :demo_gaussian_filtering.py

Files :demo_geometric_transformation_rotation.py

Files :demo_geometric_transformation_scaling.py

Files :demo_geomtric_transformation_Translation.py

Files :demo_image_thresholding.py

Files :demo_manipulatepixelarea.py

Files :demo_manipulatePixels.py

Files :demo_object_tracking.py

Files :demo_objeect_tracking2.py

Files :demo_remove_noise_blurring.py

Files :demo_shapes.py

Files :demo_thresholding_2.py

Files :demo_video_capture.py

Chapter 22:Password_Validator

Files :demo.py

Files :demo_tkinter.py

Chapter 23:Pyqt6

Files :demo_add.py

Files :demo_beverage_Calc.py

Files :demo_button_click.py

Files :demo_checkbox.py

Files :demo_helloworld.py

Files :demo_input.py

Files :demo_Layoutmgmt.py

Files :demo_nestedlayput.py

Files :demo_QGridlayout.py

Files :demo_QmessageBox.py

Files :demo_QmessageBox2.py

Files :demo_window.py

Chapter 24:Pyqt6_Notepad

Files :demo.py

Chapter 25:Pyqtform

Files :demo_Contact.py

Files :demo_formlayout.py

Files :demo_menu.py

Files :demo_qstackedlayout.py

Files :demo_toolbar.py

Files :demo_toolbar_icon.py

Chapter 26:Pyqt_Calculator

Files :demo.py

Chapter 27:Pyqt_Database

Files :demo.py

Files :demo2.py

Chapter 28:Pyqt_Paint

Files :demo.py

Files :demo2.py

Files :sample.py

Chapter 29:Pyqt_Stylesheets_QtDesigner

Files :beveragemaker.py

Files :demo4_containers.py

Files :demo_label.py

Files :demo_styling.py

Files :demo_styling2.py

Files :demo_tabs.py

Chapter 30:Pytest

Files :test_demo.py

Files :test_demo2.py

Files :__init__.py

Chapter 31:Pytest_app

Files :demo.py

Files :demo2.py

Files :__init__.py

Chapter 32:QrcodeGenerator

Files :demo.py

Chapter 33:Qtapp

Files :demo.py

Files :demo2.py

Files :form.py

Files :todolist.py

Chapter 34:REGEX

Files :regularexpressions.py

Chapter 35:Snake_Pygame

Files :demo.py

Chapter 36:Unittesting

Files :demo.py

Files :demo2.py

Files :test_demo.py

Files :test_demo2.py

Chapter 37:youtubevideodownloader

Files :demo.py

Files :demo2.py

Chapter 1:AudiotoText

This is the project description.

The basic idea of this project is to convert text into audio file

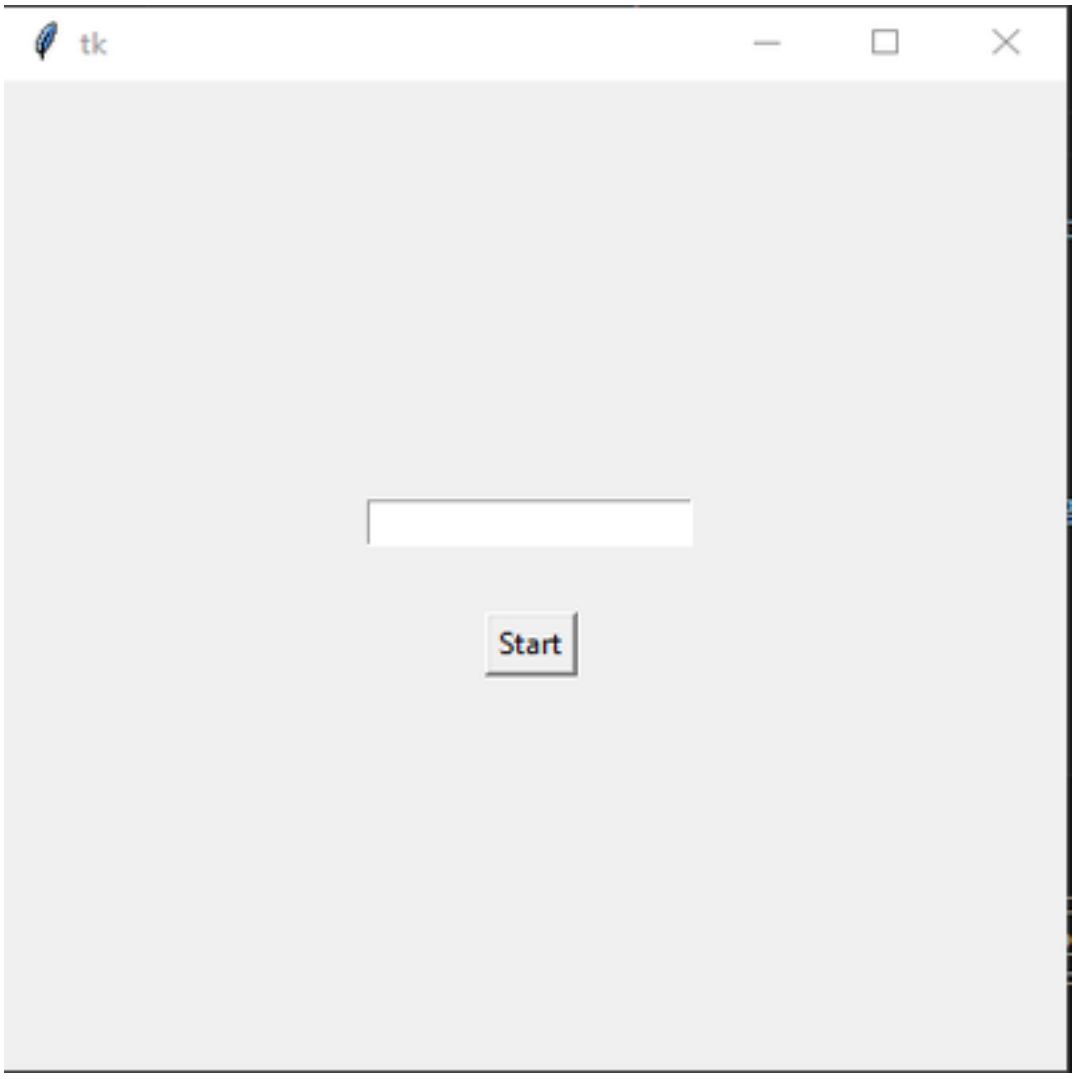

Image :D:\inligntech\AudiotoText\demo_tkinter.PNG

The libraries used in this project are:

1. gTTS

2. os

3. tkinter

The gTTS library is used to convert text into audio file.

The os library is used to play the audio file.

The tkinter library is used to create the GUI for the project.

1)The code used in this project is:

```
from gtts import gTTS
import os


text="LOL this is really funny"
output = gTTS(text=text,lang="en",slow=False)
output.save('output.mp3')

os.system("start output.mp3")
```

This code is used to convert the text file into audio file.
The text is read and stored in the variable text.

2)The code used in this project is:

```
from gtts import gTTS
import os

text =open("demo.txt","r",encoding="utf-8").read()
language="hi"
output =gTTS(text,lang=language,slow=False)
output.save('fileoutput.mp3')
os.system('start fileoutput.mp3')
```

This code is used to convert the text file into audio file.
The text file is opened in read mode and the content is read and stored in the variable text.

3)The code used in this project is:

```
from gtts import gTTS
import os
```

```
from tkinter import *



root = Tk()
canvas = Canvas(root,width=400,height=400)
canvas.pack()

def textToSpeech():
    text = entry.get()
    language="en"
    output = gTTS(text=text,lang=language,slow=False)
    output.save('output.mp3')
    os.system("start output.mp3")



entry = Entry(root)
canvas.create_window(200,180,window=entry)
button = Button(text="Start",command=textToSpeech )
canvas.create_window(200,230,window=button)

root.mainloop()
```

This code is used to convert the text file into audio file.

the text is read from the entry box and a text to speech function is used to convert the text into audio file.

Chapter 1:Building_RestAPI_Django_python_framework

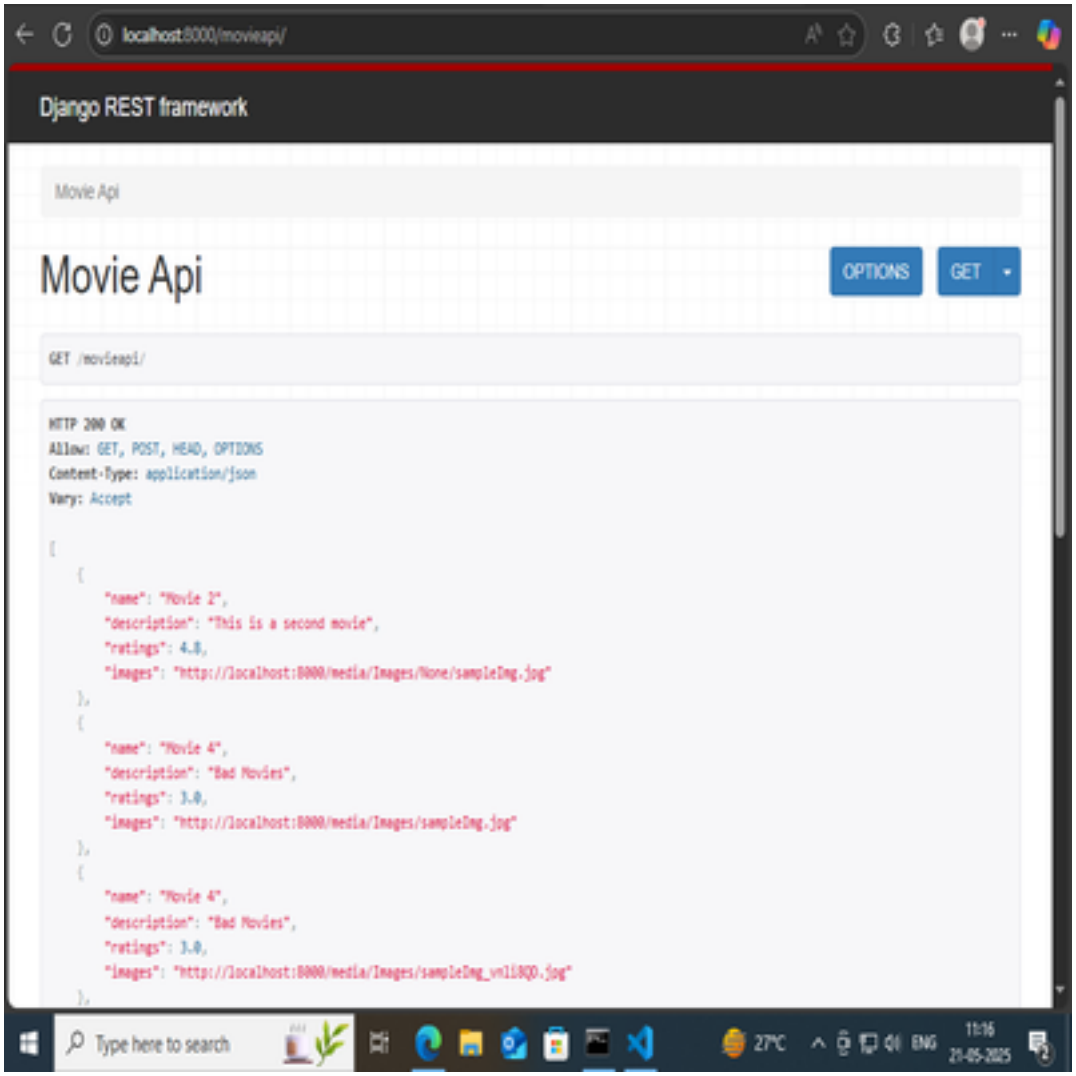

Image :D:\inligntech\Building_RestAPI_Django_python_framework\movieapi.PN

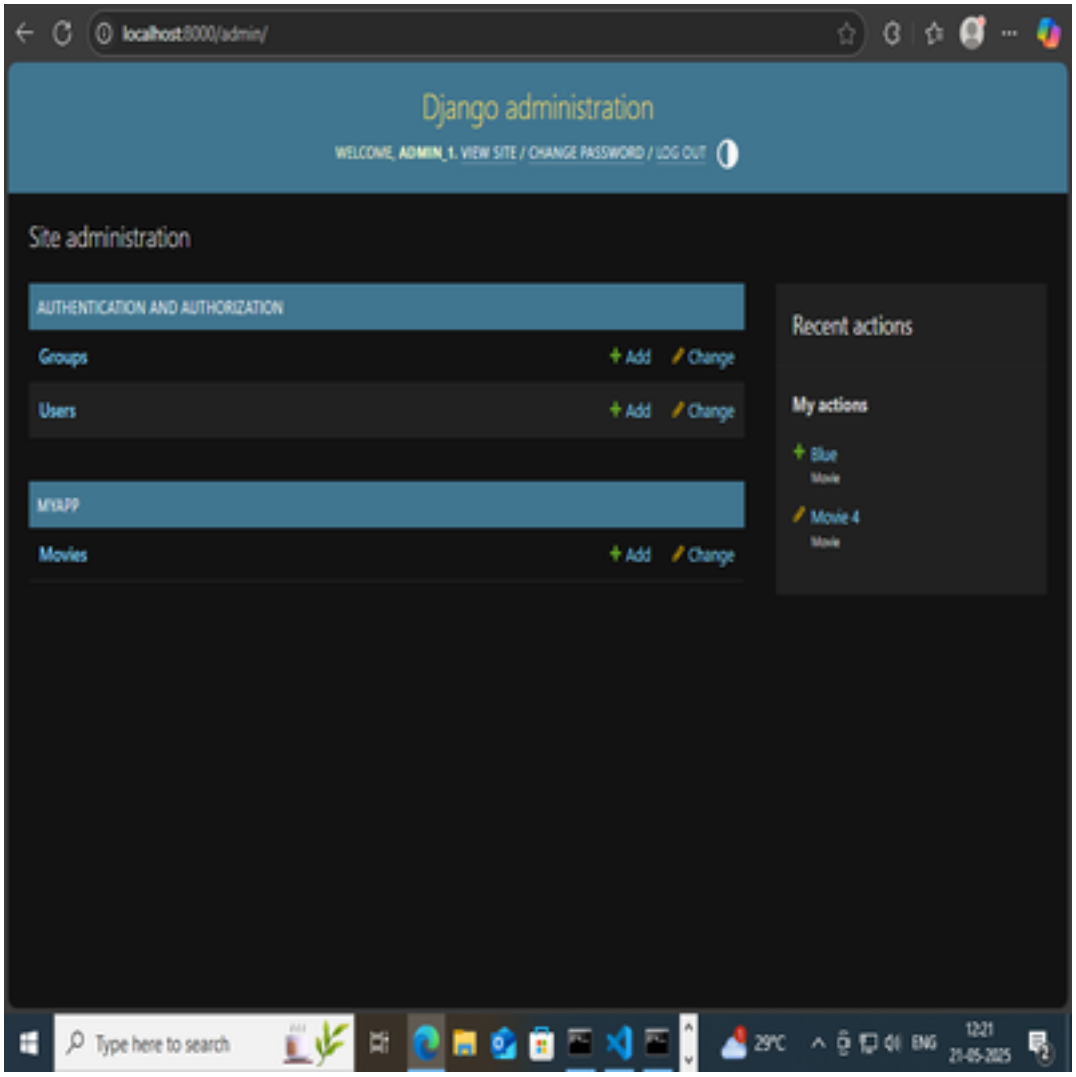

Image :D:\inligntech\Building_RestAPI_Django_python_framework\api_admin.P

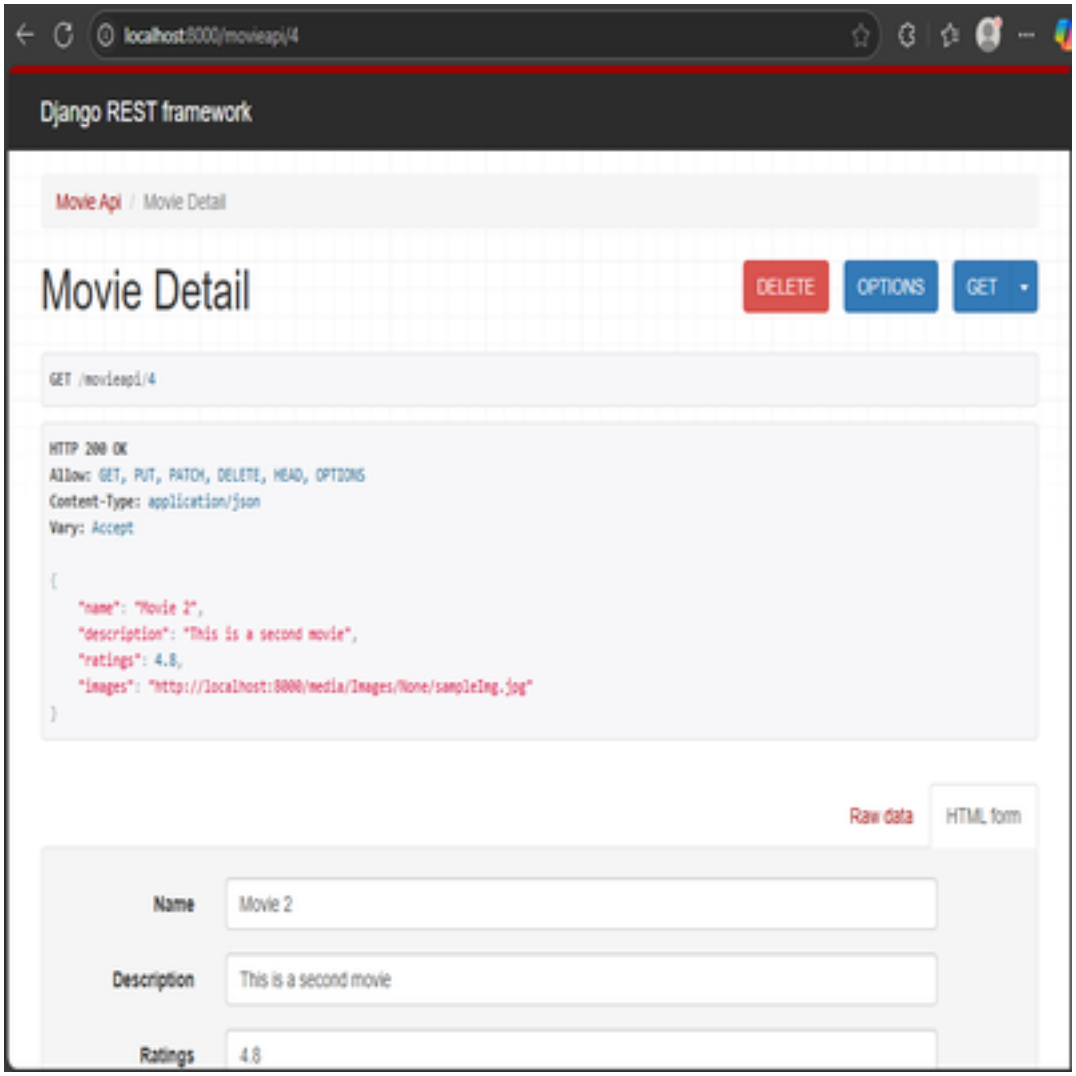

Image :D:\inligntech\Building_RestAPI_Django_python_framework\movie_detail

This is the project description.

Building_RestAPI_Django_python_framework .

This is the backend for the React Frontend.

The basic idea of this project is to create a REST API using Django Rest Framework.

The Django Rest Framework is a powerful toolkit for building Web APIs.

Please install the required libraries using the command:

pip install -r requirements.txt

1)create a Django project

django-admin startproject mysite

2)create a myapp

django-admin startapp myapp

3)Register the movie app in the settings.py file

4)create a movie model

```
from django.db import models
# Create your models here.
class Movie(models.Model):
    def __str__(self):
        return self.name
                    images = models.ImageField(upload_to = 'Images',default="Images/None/sampleImg.jpg")
    name = models.CharField(max_length=100)
    description= models.CharField(max_length=200)
    ratings = models.FloatField()
```

5)Register the movie model in the admin.py file

```
from django.contrib import admin
from .models import Movie
# Register your models here.
admin.site.register(Movie)
```

6)create a movieapi

7)Register the movieapi app in the settings.py file

8)create a movieapi serializer

```
from myapp.models import Movie
from rest_framework import serializers

class MovieSerializer(serializers.ModelSerializer):
    images = serializers.ImageField(max_length = None, use_url=True)
    class Meta:
        model = Movie
        fields =['name','description','ratings','images']
```

9) create the urls.py for the movieapi app

```
from django.contrib import admin
from django.urls import path, include
from .views import MovieAPIView,MovieDetail
urlpatterns = [
```

```
    path('',MovieAPIView.as_view()),
    path('<int:pk>',MovieDetail.as_view()),
]
```

10)create the views.py for the movieapi app

```
from django.shortcuts import render
from rest_framework import generics
from myapp.models import Movie
from .serializers import MovieSerializer
# Create your views here.
class MovieAPIView(generics.ListCreateAPIView):
    queryset = Movie.objects.all()
    serializer_class = MovieSerializer

class MovieDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = Movie.objects.all()
    serializer_class = MovieSerializer
```

11)your installed apps should look like this:

```
INSTALLED_APPS = [
    'corsheaders',
```

'movieapi',

'rest_framework',

'myapp',

"django.contrib.admin",

"django.contrib.auth",

"django.contrib.contenttypes",

"django.contrib.sessions",

"django.contrib.messages",

"django.contrib.staticfiles",

]

12)your middleware should look like this:

MIDDLEWARE = [

'corsheaders.middleware.CorsMiddleware',

"django.middleware.security.SecurityMiddleware",

"django.contrib.sessions.middleware.SessionMiddleware",

"django.middleware.common.CommonMiddleware",

"django.middleware.csrf.CsrfViewMiddleware",

"django.contrib.auth.middleware.AuthenticationMiddleware",

"django.contrib.messages.middleware.MessageMiddleware",

"django.middleware.clickjacking.XFrameOptionsMiddleware",

]

13) your CORS_ORIGIN_WHITELIST = (

'http://localhost:3000',

'http://localhost:8000',

)

14)setup th media root in the settings.py file

MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

15)setup the static root in the settings.py file

STATIC_URL = '/static/'

15)your urls.py file should look like this:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
    path("admin/", admin.site.urls),
    path('movieapi/',include('movieapi.urls'))
] + static(settings.MEDIA_URL, document_root =settings.MEDIA_ROOT)
```

We can run the server using the command:

python manage.py runserver

16)The server will run on the port 8000

17)The API will be available at the following URL:

http://localhost:8000/movieapi/

Now we can create a react frontend for this API.it will consume the API and display the data in a user friendly way.

18)The react frontend will be available at the following URL:

http://localhost:3000/

The libraries used in this project are:

asgiref

Django

django-cors-headers

djangorestframework

pillow

sqlparse

tzdata

The asgiref library is used to run the Django server.

The Django library is used to create the Django project.

The django-cors-headers library is used to allow cross-origin requests.

The djangorestframework library is used to create the REST API.

The pillow library is used to create the images.

The sqlparse library is used to parse the SQL queries.

The tzdata library is used to set the timezone.

Chapter 1:Building_RestAPI_React_Frontend

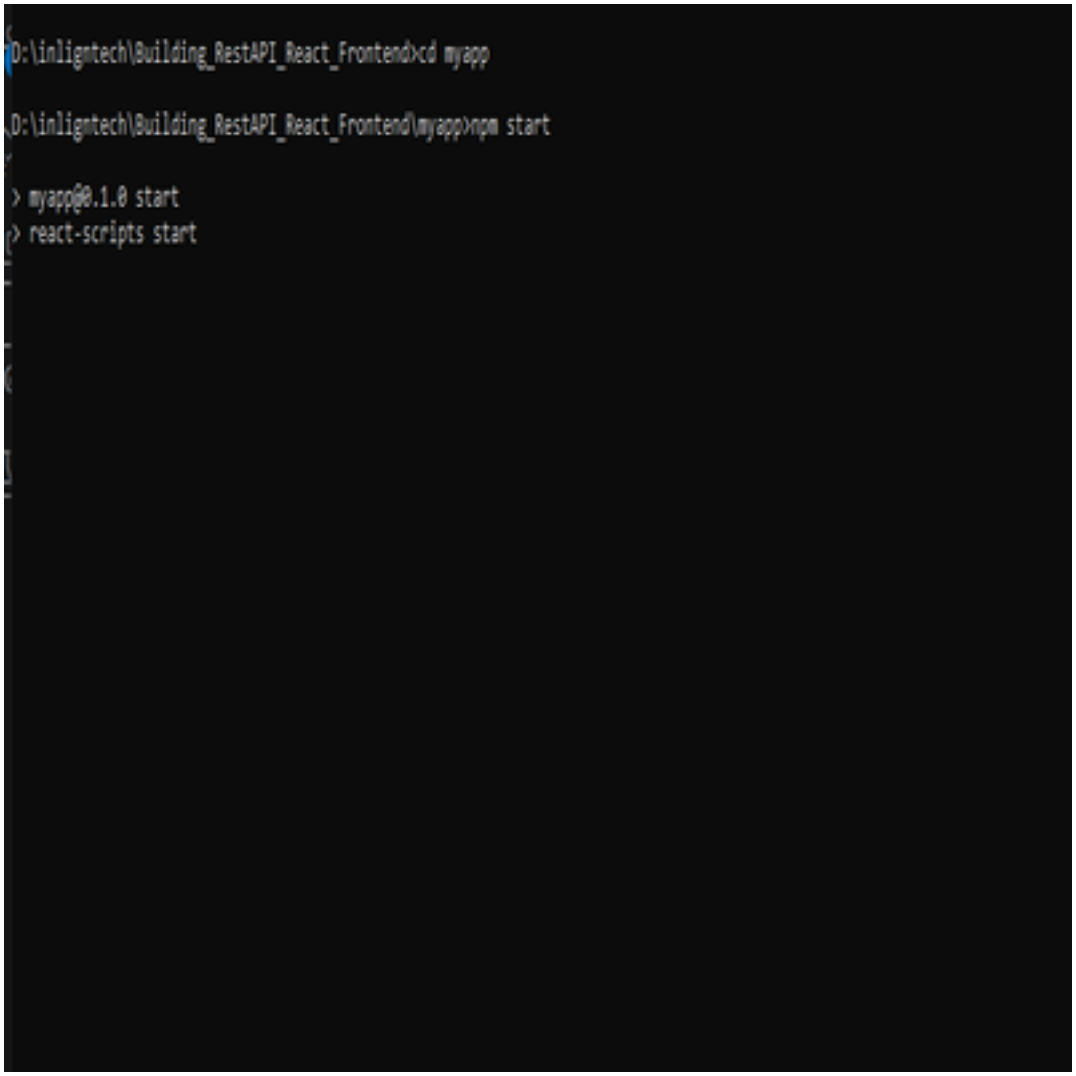

Image :D:\inligntech\Building_RestAPI_React_Frontend\start_node.PNG

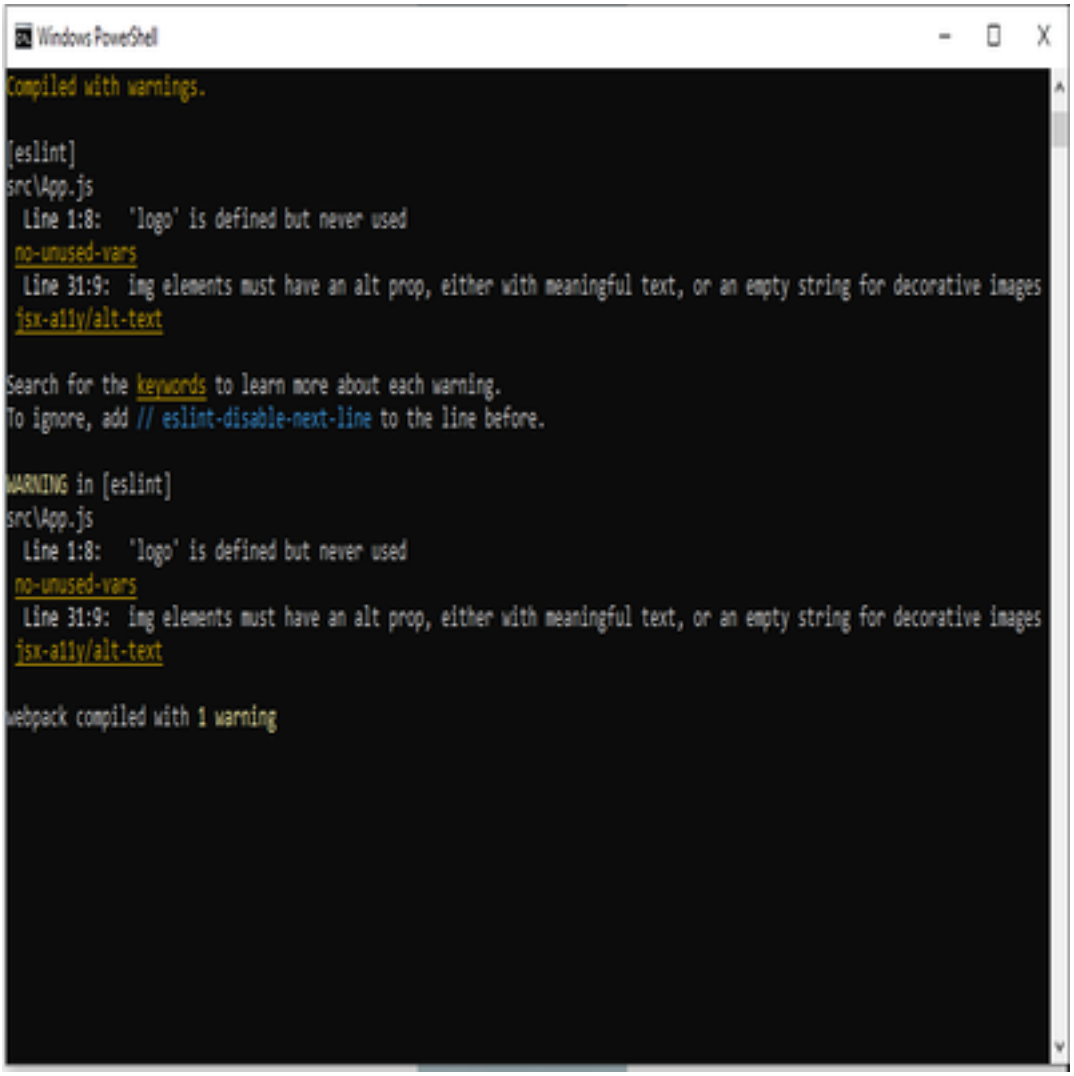

Image :D:\inligntech\Building_RestAPI_React_Frontend\start_node1.PNG

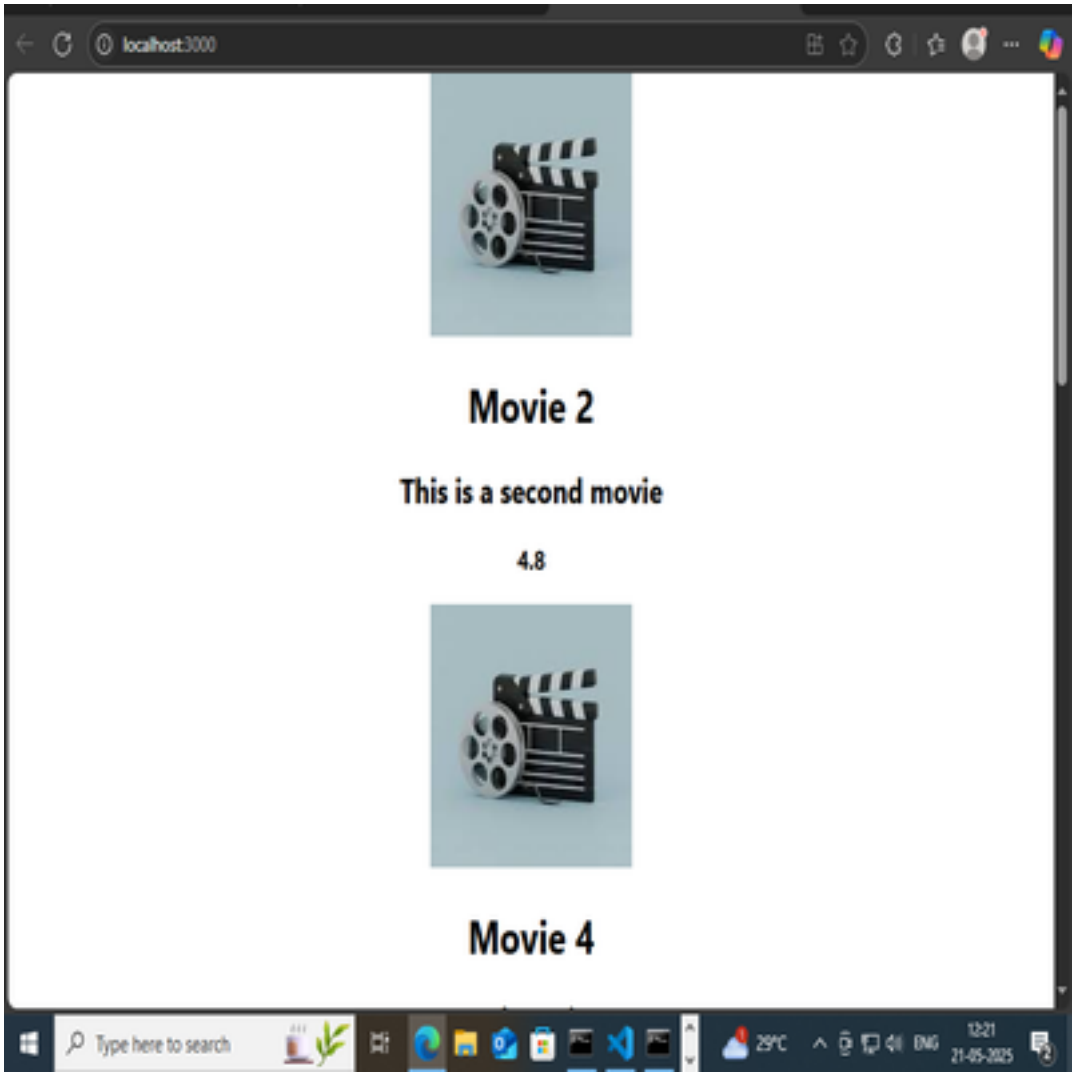

Image :D:\inligntech\Building_RestAPI_React_Frontend\react_frontend.PNG

this is the project description

we have created a react frontend for the Django Rest API.

The react frontend is created using the create-react-app command.

we did yarn install to install the required libraries.

we can start the app using the command:

yarn start

The app will run on the port 3000.

The API will be available at the following URL:

http://localhost:3000

In the src folder of the myapp react app we have created a file called App.js

The code in the App.js file is as follows:

```
import logo from './logo.svg';
import './App.css';

import React from 'react';
import axios  from 'axios';



class App extends React.Component {
  state= {
    movies: [],
  };
  componentDidMount(){

    this.getMovies()
```

```
}
getMovies()
{
  axios
  .get("http://localhost:8000/movieapi/")
  .then((res)=>{this.setState({movies: res.data})})
  .catch((error)=> {console.log(error);});
}

render(){
  return (
  <div className="App">
   {this.state.movies.map((movie)=>(

    <div key={movie.id}>
      <img src={movie.images}></img>
      <h1>{movie.name}</h1>
      <h2>{movie.description}</h2>
      <h3>{movie.ratings}</h3>

    </div>
```

```
    ))}
    </div>
  );
}
}
export default App;
```

we importes the axios library to make the API calls.
we created a class called App and in the constructor we created a state called movies.
we created a function called getMovies which will make the API call to the Django Rest API.

the styling for the react app is done using the App.css file.
The code in the App.css file is as follows:

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
```

```
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}
```

```
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

The libraries used in this project are:

1)react
2)axios

react is used to create the react app.
axios is used to make the API calls.

Chapter 1:Calculator_Tkinter

This is the project description.

The basic idea of this project is to create a calculator using tkinter.

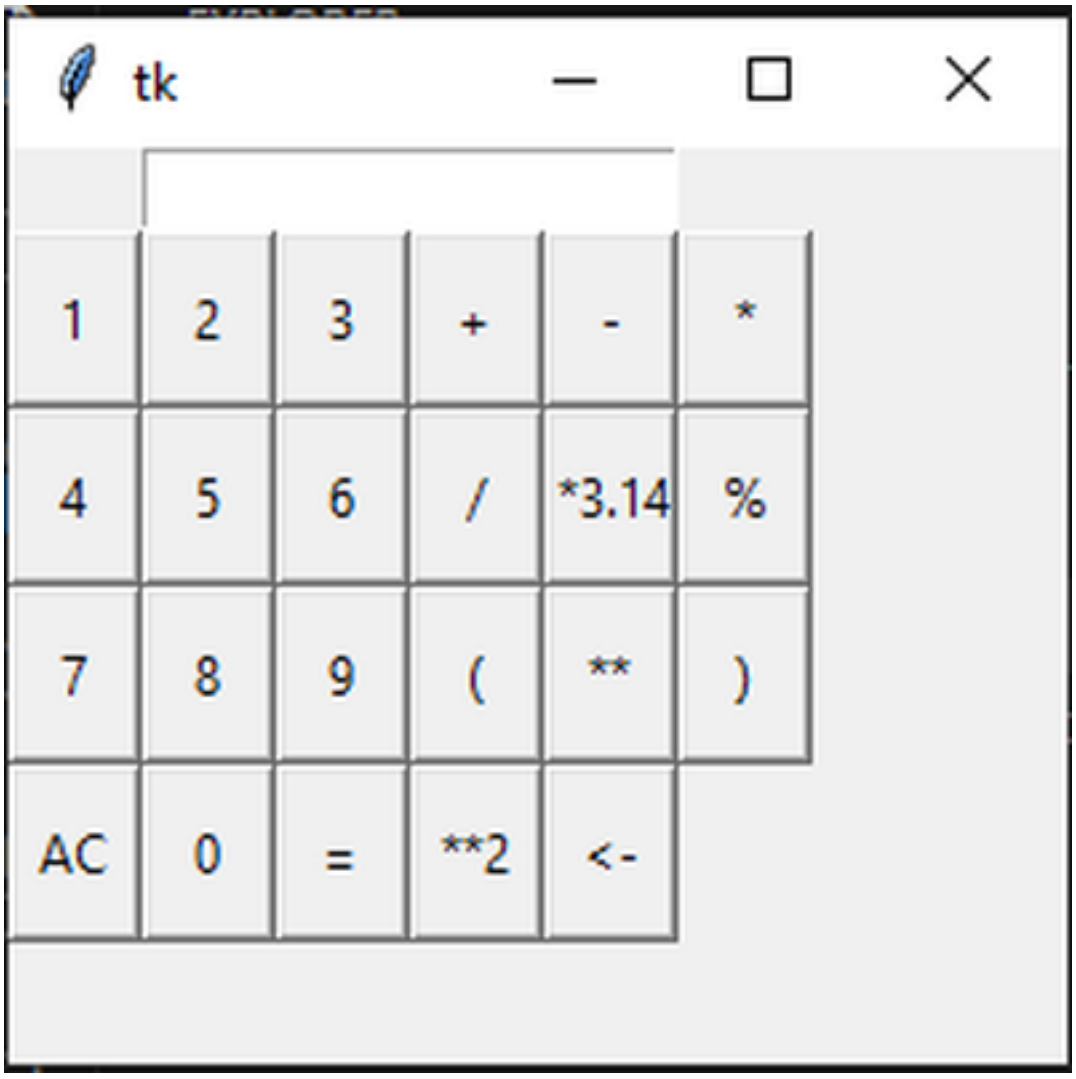

Image :D:\inligntech\Calculator_Tkinter\calculator.PNG

The code used in this project is:

```
from tkinter import *
import ast

root = Tk()
i=0
def get_number(num):
    global i
    display.insert(i, num)
    i+=1

def get_operation(operator):
    global i
    length = len(operator)
    display.insert(i, operator)
    i+=length

def clear_all():
    display.delete(0,END)

def calculate():
```

```
    entire_string = display.get()
    try:
        node = ast.parse(entire_string,mode='eval')
        result = eval(compile(node,"<string>",mode='eval'))
        clear_all()
        display.insert(0, result)
    except Exception:
        clear_all()
        display.insert(0, "Error")

def undo():
    entire_string = display.get()
    if len(entire_string)>0:
        entire_string = entire_string[:-1]
        clear_all()
        display.insert(0,entire_string)
    else:
        clear_all()
        display.insert(0,"Error")

display =Entry(root)
display.grid(row=1,columnspan=6)
```

```
numbers = [ 1,2,3,4,5,6,7,8,9]
counter =0
for x in range(3):
    for y in range(3):
        button_text = numbers[counter]
          button = Button(root,text=button_text,width=3,height=2, command = lambda text=button_text: get_number(text))
        button.grid(row=x+2,column=y)
        counter +=1

button =Button(root,text=0,width=3,height=2, command= lambda : get_number(0))
button.grid(row=5,column=1)


count=0
operations = ['+', '-', '*', '/','*3.14',"%","(","**", ")","**2"]
for x in range(4):
    for y in range(3):
        if count < len(operations):
                                                    button = Button(root,text=operations[count],width=3,height=2,command= lambda
```

```
text=operations[count]: get_operation(text))
        count +=1
        button.grid(row=x+2,column=y+3)

button
=Button(root,text="AC",width=3,height=2,command=clear_all).grid(row=5,c
olumn=0)
button
=Button(root,text="=",width=3,height=2,command=calculate).grid(row=5,co
lumn=2)
button
=Button(root,text="<-",width=3,height=2,command=undo).grid(row=5,colu
mn=4)



root.mainloop()
```

This code is used to create a calculator using tkinter.
The tkinter library is used to create the GUI for the project.

The ast library is used to parse the expression and evaluate it.

The eval function is used to evaluate the expression.

The get_number function is used to get the number from the button and insert it in the entry box.

The get_operation function is used to get the operation from the button and insert it in the entry box.

The clear_all function is used to clear the entry box.

The calculate function is used to evaluate the expression and display the result in the entry box.

The undo function is used to delete the last character from the entry box.