Riga Technical University

Faculty of Electronics and Telecommunication

TELECOMMUNICATION SOFTWARE

Python cloud full stack development

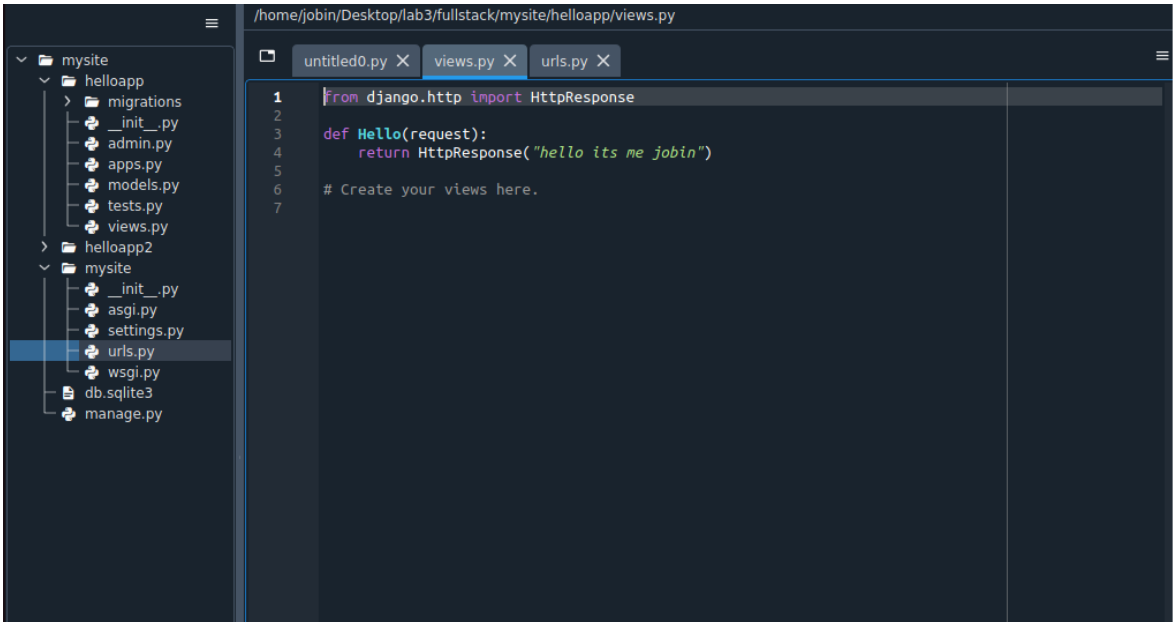
Jobin Kachappally Mathai

211AEG004

Riga, 2023

Example1: Django's Hello World program (Tips: • This application inspection of a specific function, please return a string-related web page that can combine with CSS and JS to design your style.)

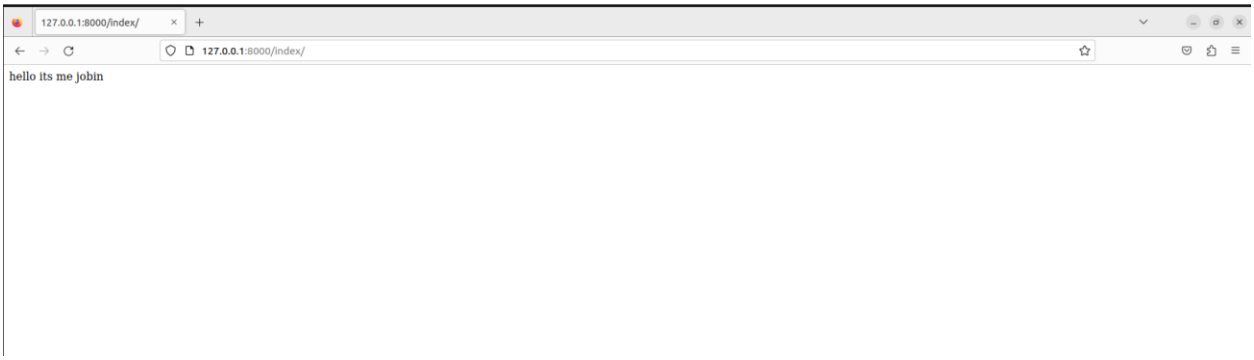
In the view.py we give the output in an function Hello



In this urls.py we give the path

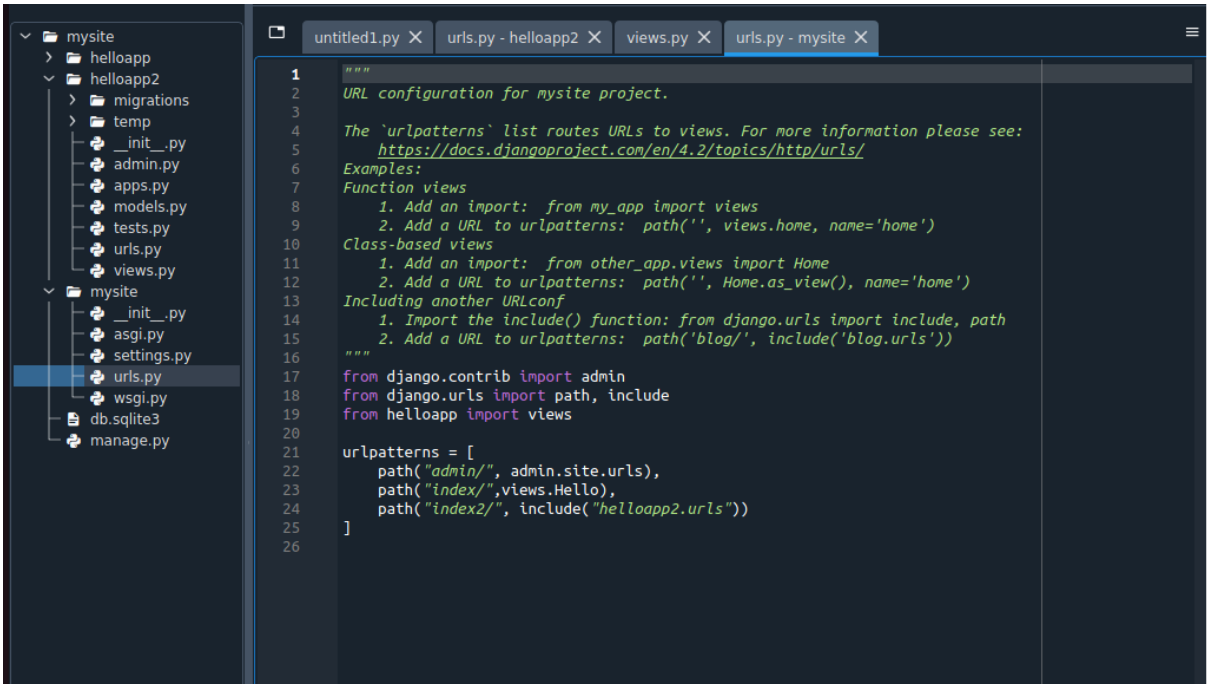
```python
"""
URL configuration for mysite project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from helloapp import views

urlpatterns = [
    path("admin/", admin.site.urls),
    path("index/",views.Hello)
]
```

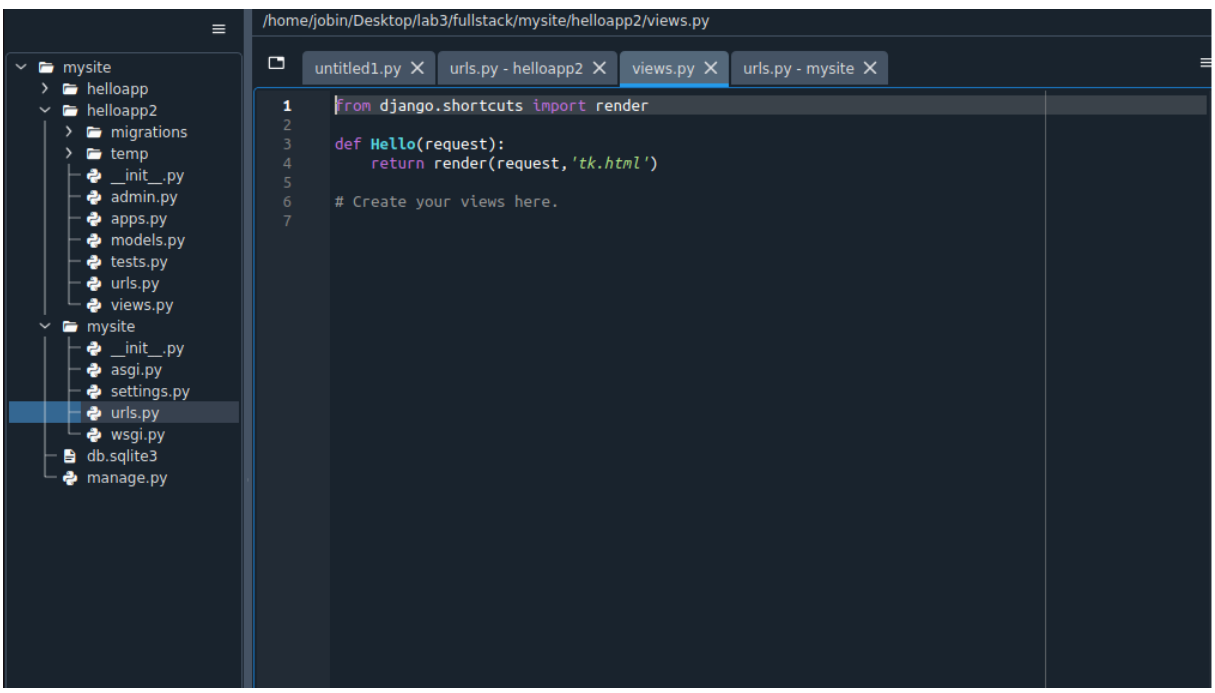
The output on the server

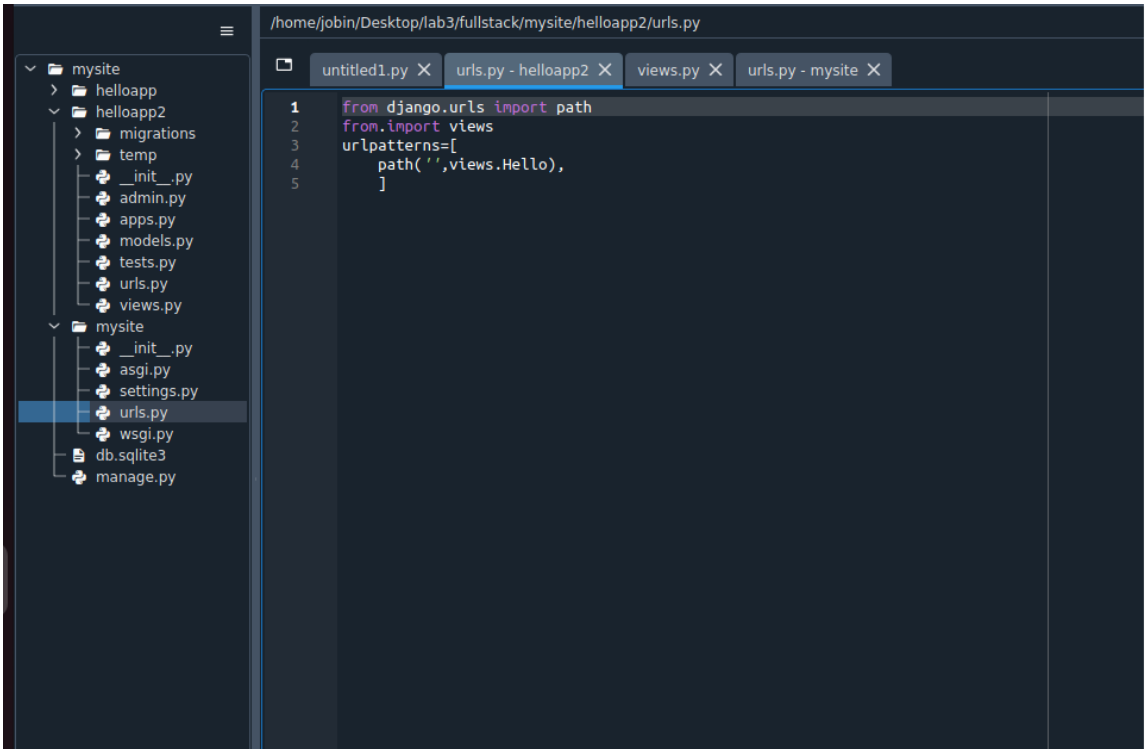


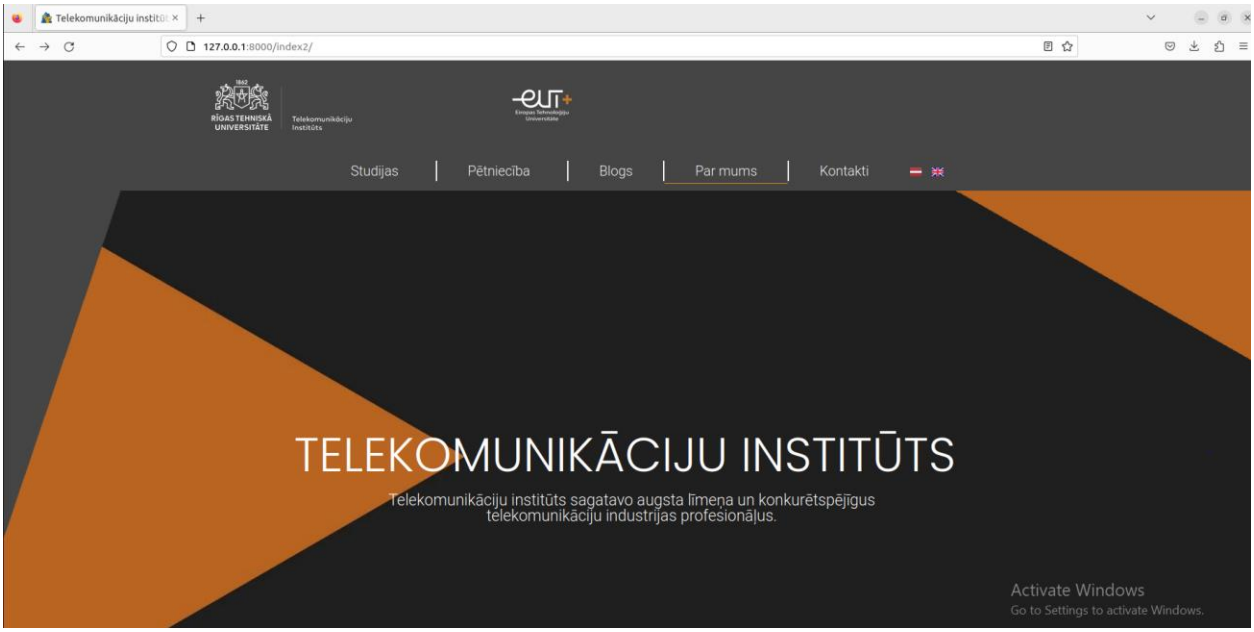
Example 2: Improve example 1 and return an HTML page, not a string; please use MVT (Model View Template) to design the pattern. You are welcome to use multiple decorations on your HTML page.

From the web site given in the class I take the html file and it saved in the temp file and I added the location of the temp file in settings


mysite
helloapp
helloapp2
migrations
temp
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
mysite
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
manage.py
untitled1.py
urls.py - helloapp2
views.py
urls.py - mysite
"""
URL configuration for mysite project.
The `urlpatterns` list routes URLs to views. For more information please see:
https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from helloapp import views
urlpatterns = [
path("admin/", admin.site.urls),
path("index/",views.Hello),
path("index2/", include("helloapp2.urls"))
]


/home/jobin/Desktop/lab3/fullstack/mysite/helloapp2/views.py
mysite
helloapp
helloapp2
migrations
temp
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
mysite
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
manage.py
untitled1.py
urls.py - helloapp2
views.py
urls.py - mysite
from django.shortcuts import render
def Hello(request):
return render(request,'tk.html')
Create your views here.


/home/jobin/Desktop/lab3/fullstack/mysite/helloapp2/urls.py
mysite
helloapp
helloapp2
migrations
temp
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
mysite
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
manage.py
untitled1.py
urls.py - helloapp2
views.py
urls.py - mysite
from django.urls import path
from.import views
urlpatterns=[
path('',views.Hello),
]


127.0.0.1:8000/index2/
Studijas
Pētniecība
Blogs
Par mums
Kontakti
TELEKOMUNIKĀCIJU INSTITŪTS
Telekomunikāciju institūts sagatavo augsta līmeņa un konkurētspējīgus telekomunikāciju industrijas profesionāļus.
Activate Windows
Go to Settings to activate Windows.

Example 3: Cloud message board. Basic function definition: 1, Submit message function: Users can set their own name as A, specify any name B and leave a message to B, record it as msg, and the message will be saved in the cloud. 2, Get message function: input name A, and the cloud will return the 20 latest message records.

Example 4: Please try to test Django's different response types, including HttpResponse class and subclasses (10 in total), JsonResponse class, StreamingHttpResponse, and FileResponse class. Please show those response files, JSON, or video on your screenshot.

```
# models.py

from django.db import models


class Message(models.Model):

    sender = models.CharField(max_length=100)

    receiver = models.CharField(max_length=100)

    message = models.TextField()

    timestamp = models.DateTimeField(auto_now_add=True)


# forms.py

from django import forms

from .models import Message


class MessageForm(forms.ModelForm):

    class Meta:

        model = Message

        fields = ['sender', 'receiver', 'message']


# views.py

from django.shortcuts import render, redirect

from .forms import MessageForm
```

```
def submit_message(request):
    if request.method == 'POST':
        form = MessageForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('message_board')
    else:
        form = MessageForm()
    return render(request, 'submit_message.html', {'form': form})


def get_messages(request):
    messages = Message.objects.all().order_by('-timestamp')[:20]
    return render(request, 'message_board.html', {'messages': messages})
```

This code defines a Message model in the models.py file, which has fields for the sender's name, the receiver's name, the message, and a timestamp.

In the forms.py file, a MessageForm is created, which is a Django ModelForm for the Message model. This form allows users to submit their messages by providing their name, the receiver's name, and the message.

In the views.py file, there are two view functions defined: submit_message and get_messages. The submit_message function handles the process of submitting a message to the message board. It checks if the request method is 'POST', and if it is, it creates a new MessageForm object with the data from the request, validates the form and saves it to the database if it's valid. If the request method is not 'POST', it creates an empty form. The view function then renders the submit_message.html template and passes the form to it.

The get_messages function handles the process of retrieving the latest 20 messages from the message board. It queries the database for all the messages, orders them by timestamp in descending order and takes the first 20 of them. It then renders the message_board.html template and passes the messages to it.

In this way, you have created a basic cloud message board that allows users to submit messages and view the latest 20 messages. You can add more functionality, such as pagination, authentication, and authorization, as well as design and styling to the message board to make it more visually appealing and user-friendly.

It is important to note that you need to set up your database in the settings.py file and make the necessary migrations to create the table in the database. Also, you need to define the URLs in the urls.py file to link the views functions to the URLs, you can use the path() or url() function to do that.

Example 4: Please try to test Django's different response types, including HttpResponse class and subclasses (10 in total), JsonResponse class, StreamingHttpResponse, and FileResponse class. Please show those response files, JSON, or video on your screenshot.

```
from django.http import HttpResponse, HttpResponseBadRequest, HttpResponseForbidden, HttpResponseNotFound, HttpResponseNotAllowed, HttpResponseNotModified, HttpResponseRedirect, HttpResponsePermanentRedirect, HttpResponseGone, HttpResponseServerError

from django.http import JsonResponse

from django.core.files import File

from django.core.files.storage import FileSystemStorage

from django.core.servers.basehttp import FileWrapper


def http_response(request):

    return HttpResponse("This is a simple HttpResponse.")


def http_response_bad_request(request):

    return HttpResponseBadRequest("This is a Bad Request HttpResponse.")


def http_response_forbidden(request):

    return HttpResponseForbidden("This is a Forbidden HttpResponse.")


def http_response_not_found(request):

    return HttpResponseNotFound("This is a Not Found HttpResponse.")


def http_response_not_allowed(request):

    return HttpResponseNotAllowed("This is a Not Allowed HttpResponse.")


def http_response_not_modified(request):

    return HttpResponseNotModified("This is a Not Modified HttpResponse.")
```

```
def http_response_redirect(request):
    return HttpResponseRedirect("/")

def http_response_permanent_redirect(request):
    return HttpResponsePermanentRedirect("/")

def http_response_gone(request):
    return HttpResponseGone("
```