MINI PROJECT
(2020-2021)

RT Messenger

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by-
Prashant Asthana
(181500487)
Tanishq Tripathi
(181500751)
Shriyanshi Baranwal
(181500691)

Supervised By: -
Mr. Amir Khan
Technical Trainer
Department of Computer Engineering & Applications

Contents

Abstract **3**

1. Introduction **4**

1.1 General Introduction to the topic **4**

1.2 Technical Details **4**

1.3 Hardware and Software Requirements **5**

2. Objectives **6**

3. Implementation Details **7**

4. Some Screenshots **8**

Abstract

Lately we see a constant switch towards real-time applications. With the wide support for WebSockets in recent browsers, more and more frameworks are giving us the ability to use them. Django, which was primarily a request/response-based framework, is doing the switch with Django Channels. Django channels is a lot more than just support for WebSockets, it's a complete architectural change for Django and - in my honest opinion - a great move towards the new era of frameworks.

We are creating a platform where a particular person can send a mail to different ids at a same time without having to check the database of their employees. This saves a lot of time. This project saves a lot of storage as there would be no use of creating any other database for sending mail to a large number of people. It creates a database of the details entered. This data can be updated any time, very easily and be securely saved.

Introduction

1.1 General Introduction to the topic

Nowadays, when all sorts of messenger have become extremely popular, when every second large company has launched or developed its own instant messenger, when an increase of smiles and change in the text size is considered as innovation, in the era of Gmail, yahoo, Orkut, Whatsapp etc. We will use Django-channels.

Python is fast becoming a popular coding language in the world, and there are many popular frameworks that build on Python. One of them is Django and it has many functionalities and supporting libraries. For this article, we like to explore one interesting method that builds on Django to handle not only HTTP but also long running connections such as WebSockets, MQTT, etc.

1.2 Technical Details

Following are the technologies used in this project:

- Python-Django
- WebSockets
- Web Development
- Python

It uses the Django auth system to provide user accounts; users are only be able to use the chat once logged in, and this provides their username details for the messenger.

This package allows our application to interact with a user not only using HTTP 1.1 (request-response), but also using HTTP/2 and WebSocket.

WebSocket is designed for exchanging messeges between the client and the web server in real time. You should consider it as an open channel between the client and the server, with the ability to subscribe to the events sent to it.

1.3 Hardware and Software Requirements

Software Specification:

- Technology Implemented: JavaScript, Django, python, WebSocket
- User Interface Design : Web based Application
- Web Browser: Chrome

Hardware Requirement:

- Processor: Intel CORE i3
- Operating System: Windows 10
- RAM:4 GB
- Hardware System: Computer System
- Hard Disk: 64 GB

Objectives

The main objective of creating this real-time messenger is:

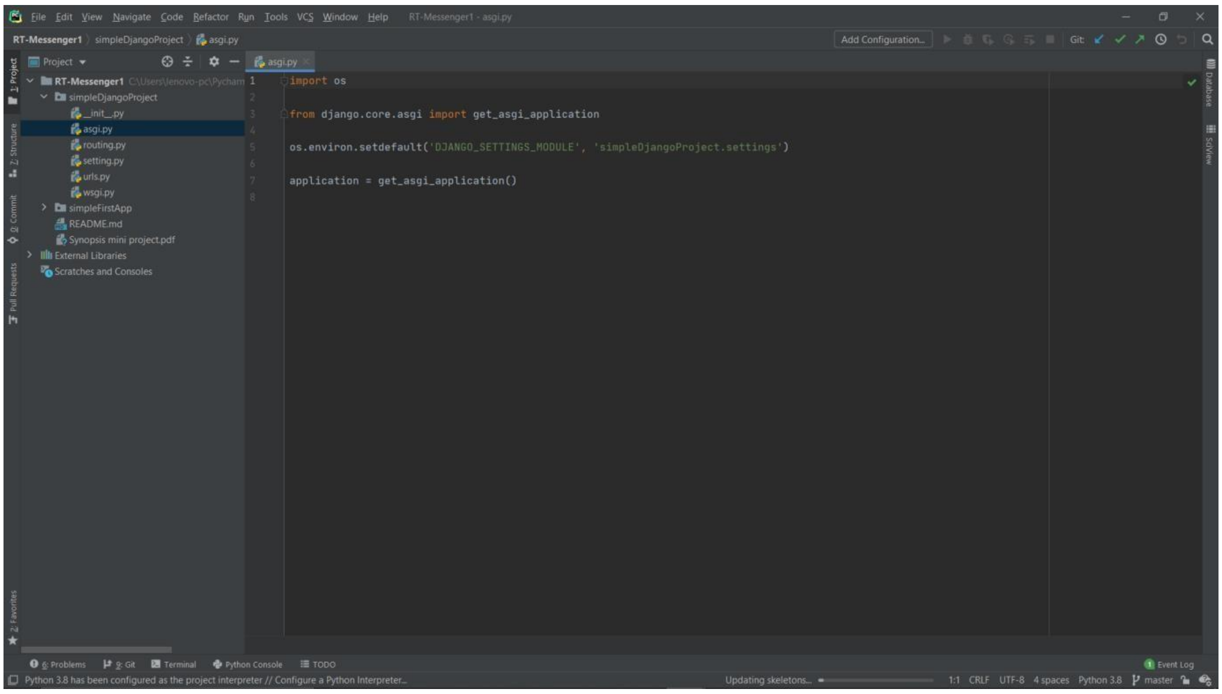
Firstly, we are creating a platform where a particular person can send a mail to different ids at a same time without having to check the database of their employees. This saves a lot of time.

Secondly, this project saves a lot of storage as there would be no use of creating any other database for sending mail to a large number of people.

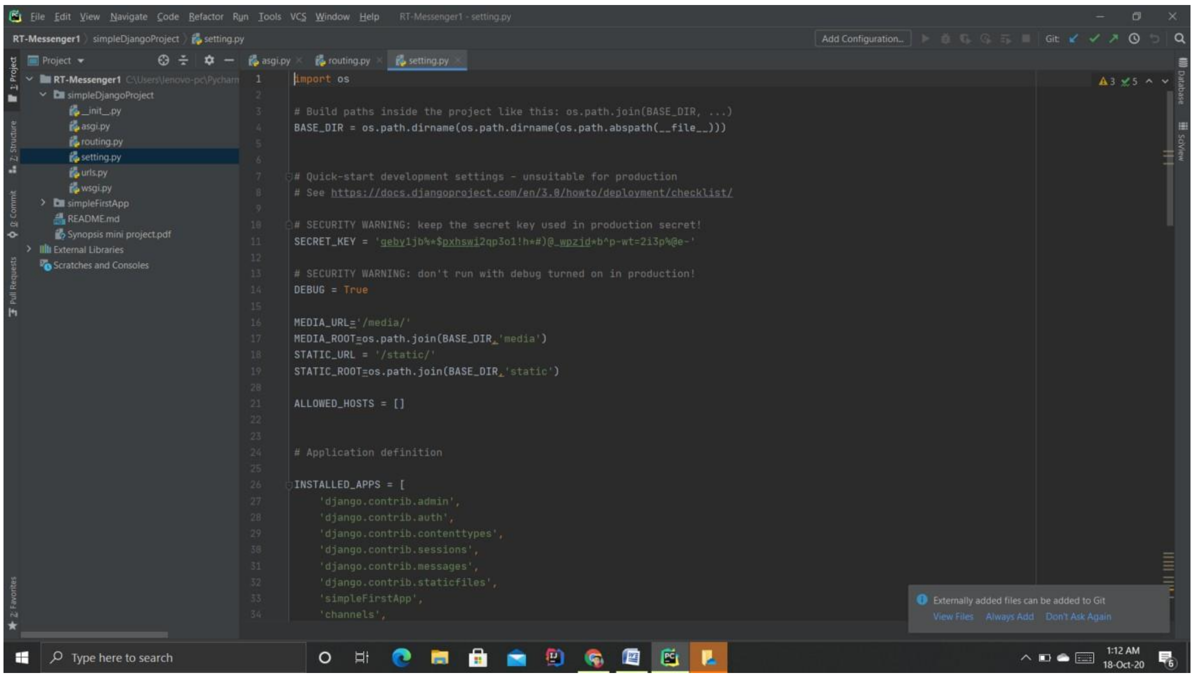
Thirdly, it creates a database of the details entered. This data can be updated any time, very easily and be securely saved.

Django is a web building framework based on the Python programming language. It uses a Model-View-Template ("MVT") architecture. Django is being popular because it is fast, secure, and a scalable high-level Python programming for web building. Learning Django is not that easy, but when you get used to it, it has many great functionalities.

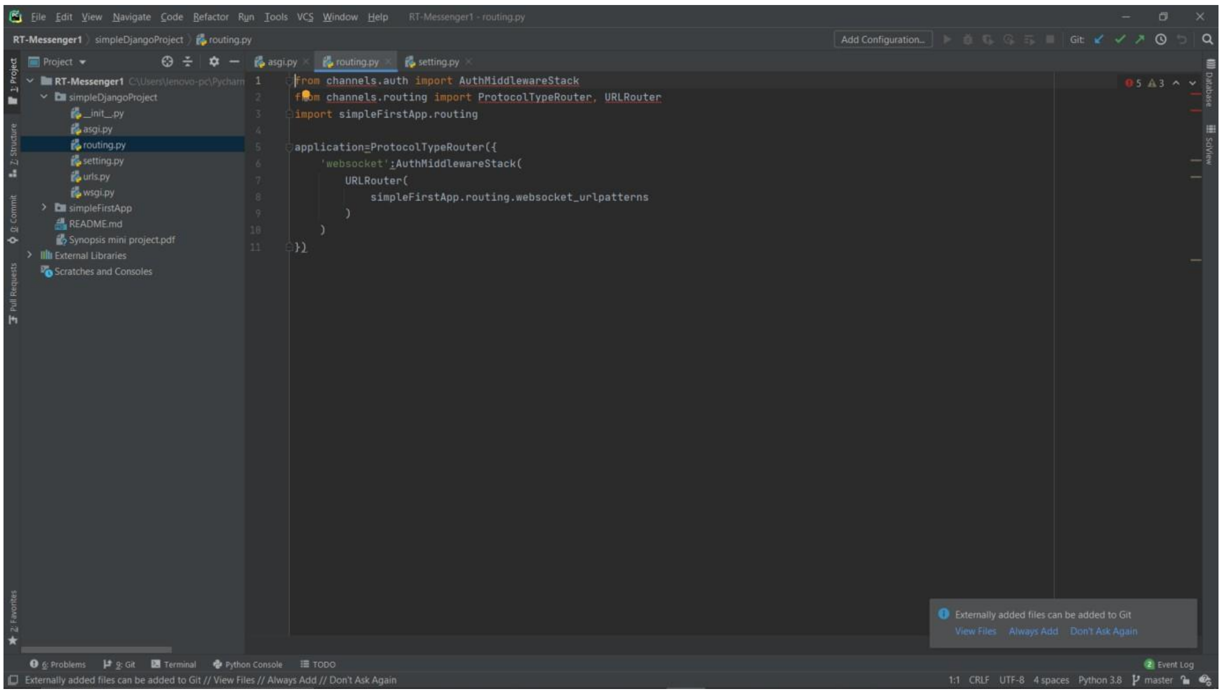
Some Screenshots



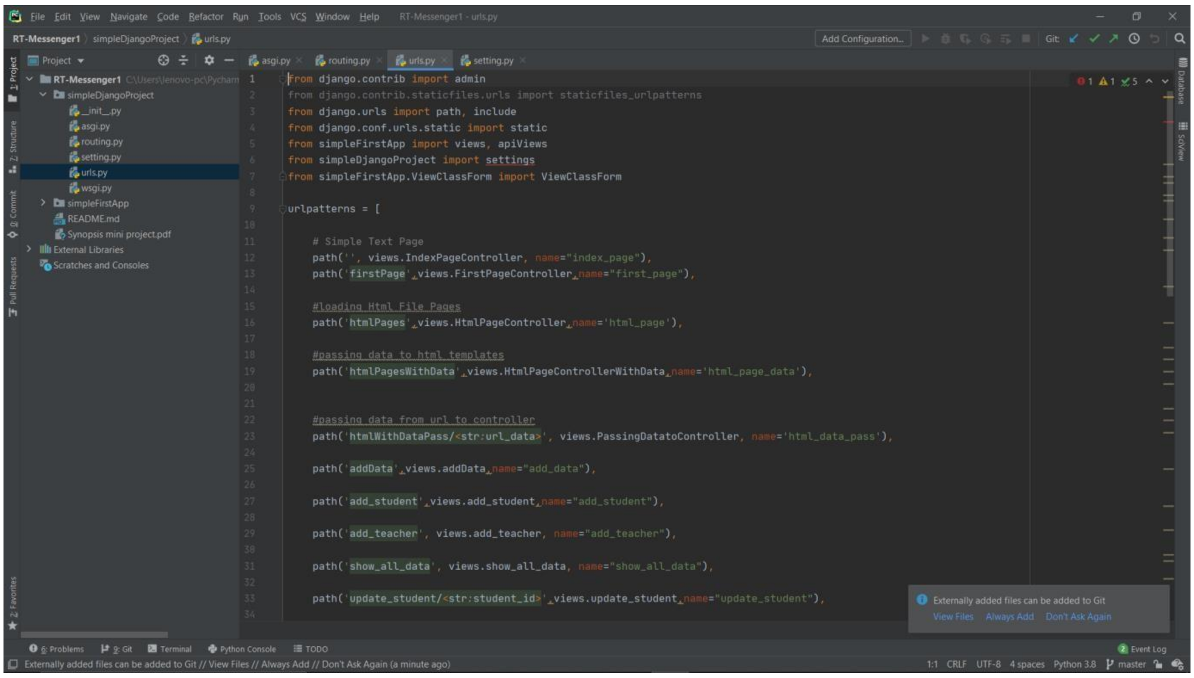
```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'simpleDjangoProject.settings')

application = get_asgi_application()
```



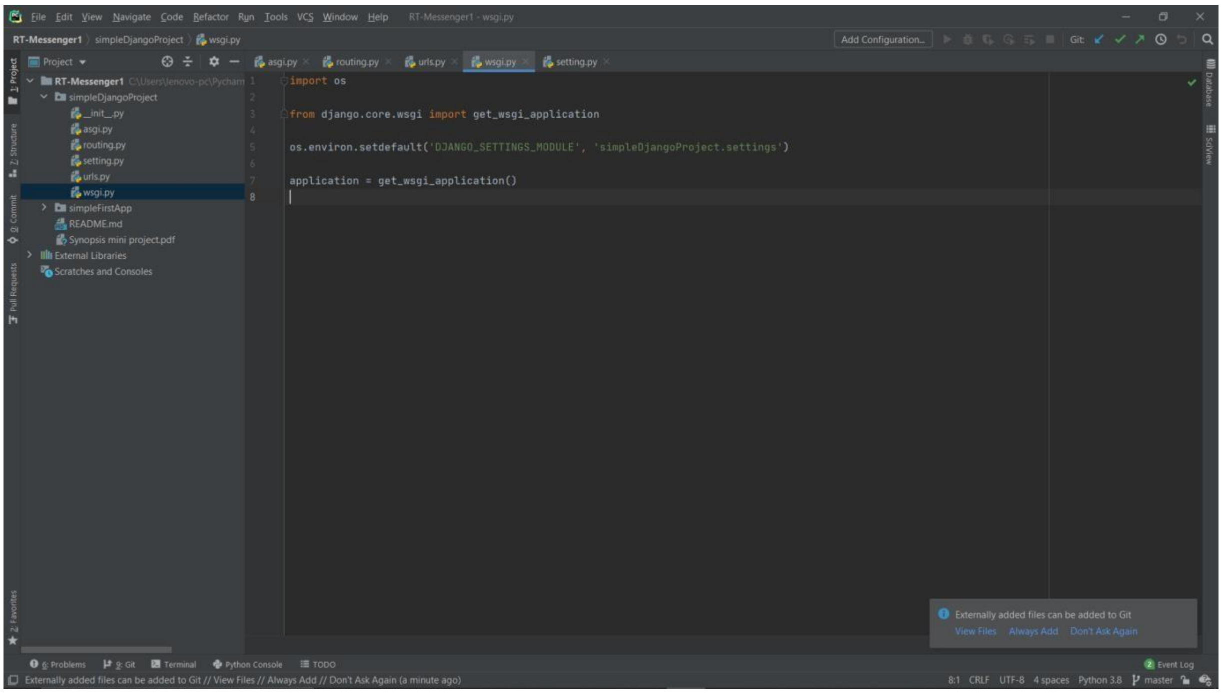
```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'geby1jb%*$pxhswi2qp3o1!h*#)@_wpzjd*b^p-wt=2i3p%@e-'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

MEDIA_URL='/media/'
MEDIA_ROOT=os.path.join(BASE_DIR,'media')
STATIC_URL = '/static/'
STATIC_ROOT=os.path.join(BASE_DIR,'static')

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'simpleFirstApp',
    'channels',
```



```
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
import simpleFirstApp.routing

application=ProtocolTypeRouter({
    'websocket':AuthMiddlewareStack(
        URLRouter(
            simpleFirstApp.routing.websocket_urlpatterns
        )
    )
})
```

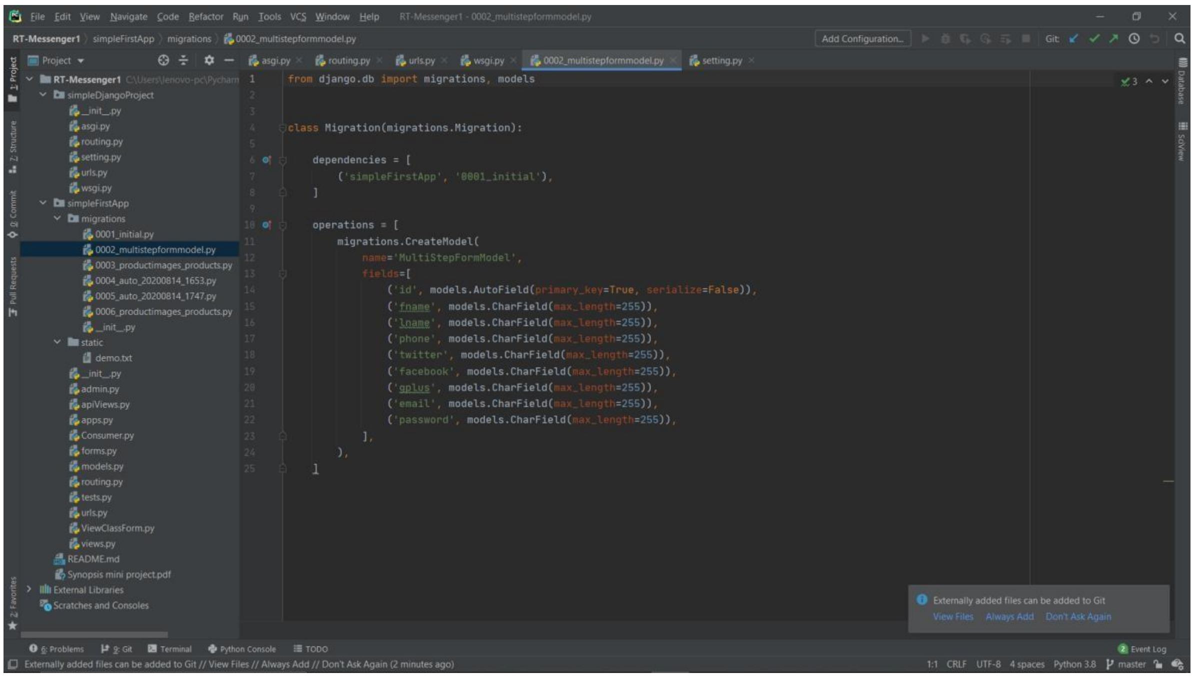


```
from django.contrib import admin
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.urls import path, include
from django.conf.urls.static import static
from simpleFirstApp import views, apiViews
from simpleDjangoProject import settings
from simpleFirstApp.ViewClassForm import ViewClassForm

urlpatterns = [

    # Simple Text Page
    path('', views.IndexPageController, name="index_page"),
    path('firstPage',views.FirstPageController,name="first_page"),

    #loading Html File Pages
    path('htmlPages',views.HtmlPageController,name='html_page'),

    #passing data to html templates
    path('htmlPagesWithData',views.HtmlPageControllerWithData,name='html_page_data'),


    #passing data from url to controller
    path('htmlWithDataPass/<str:url_data>', views.PassingDatatoController, name='html_data_pass'),

    path('addData',views.addData,name="add_data"),

    path('add_student',views.add_student,name="add_student"),

    path('add_teacher', views.add_teacher, name="add_teacher"),

    path('show_all_data', views.show_all_data, name="show_all_data"),

    path('update_student/<str:student_id>',views.update_student,name="update_student"),
```



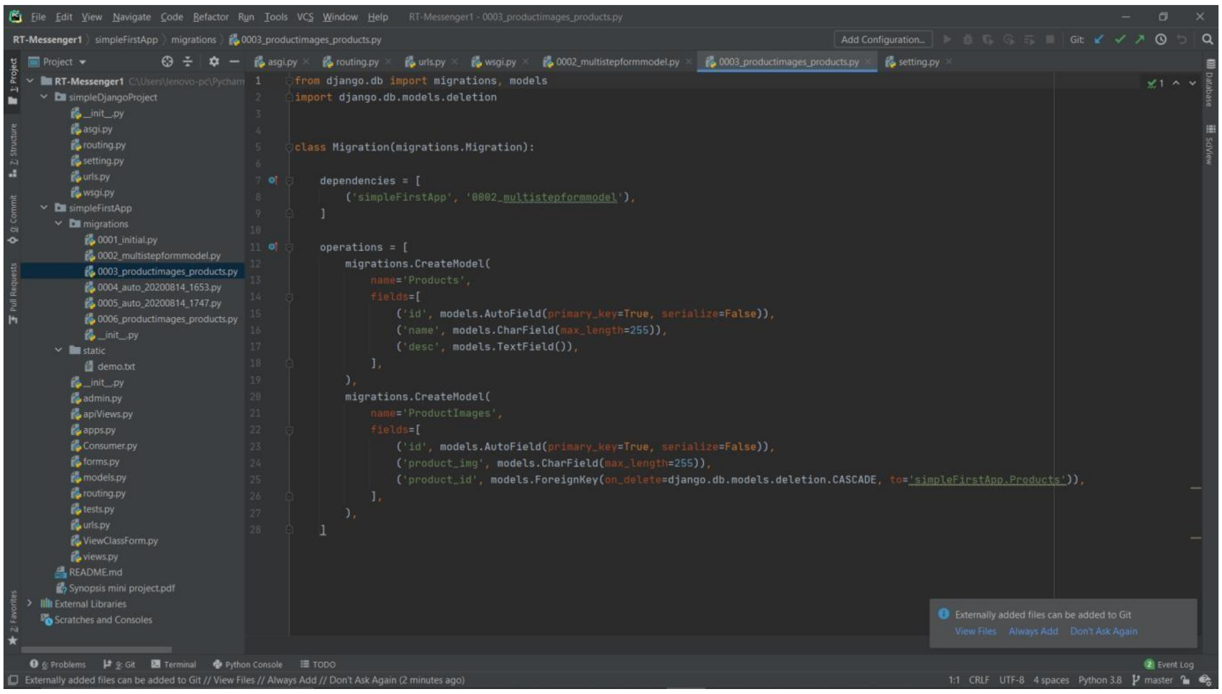
```
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'simpleDjangoProject.settings')

application = get_wsgi_application()
```

Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (a minute ago)

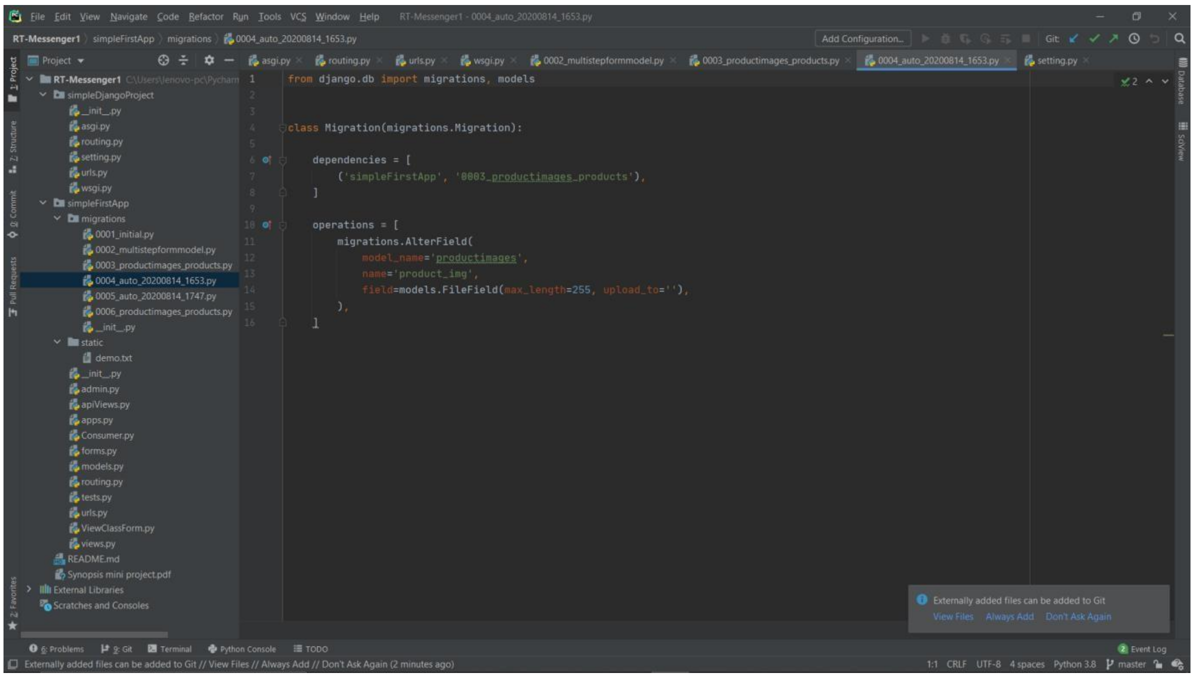


```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('simpleFirstApp', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='MultiStepFormModel',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('fname', models.CharField(max_length=255)),
                ('lname', models.CharField(max_length=255)),
                ('phone', models.CharField(max_length=255)),
                ('twitter', models.CharField(max_length=255)),
                ('facebook', models.CharField(max_length=255)),
                ('gplus', models.CharField(max_length=255)),
                ('email', models.CharField(max_length=255)),
                ('password', models.CharField(max_length=255)),
            ],
        ),
    ]
```

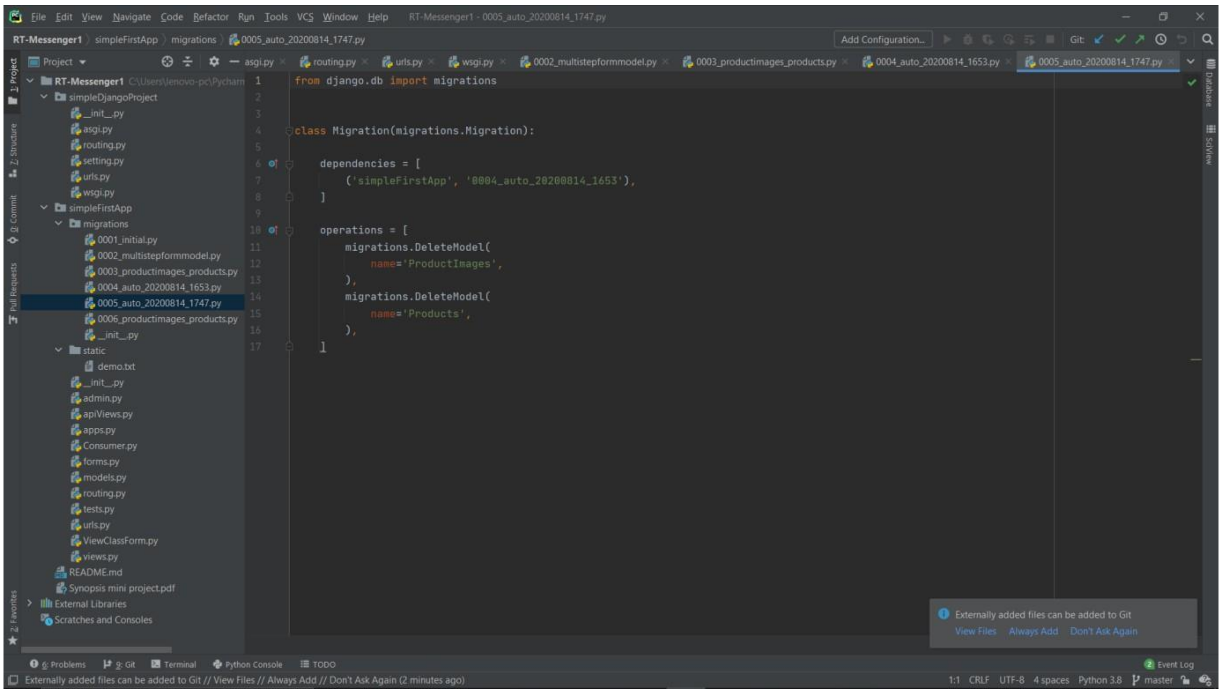
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (2 minutes ago)



```python
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        ('simpleFirstApp', '0002_multistepformmodel'),
    ]

    operations = [
        migrations.CreateModel(
            name='Products',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('name', models.CharField(max_length=255)),
                ('desc', models.TextField()),
            ],
        ),
        migrations.CreateModel(
            name='ProductImages',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('product_img', models.CharField(max_length=255)),
                ('product_id', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='simpleFirstApp.Products')),
            ],
        ),
    ]
```

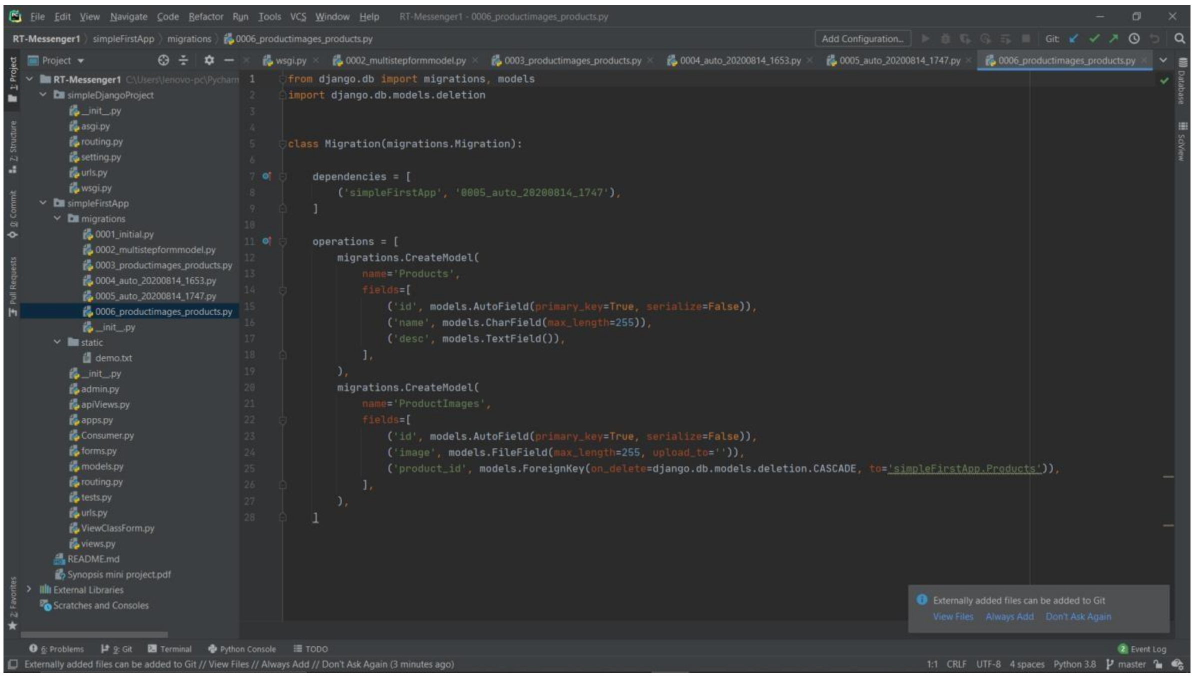


```python
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('simpleFirstApp', '0003_productimages_products'),
    ]

    operations = [
        migrations.AlterField(
            model_name='productimages',
            name='product_img',
            field=models.FileField(max_length=255, upload_to=''),
        ),
    ]
```



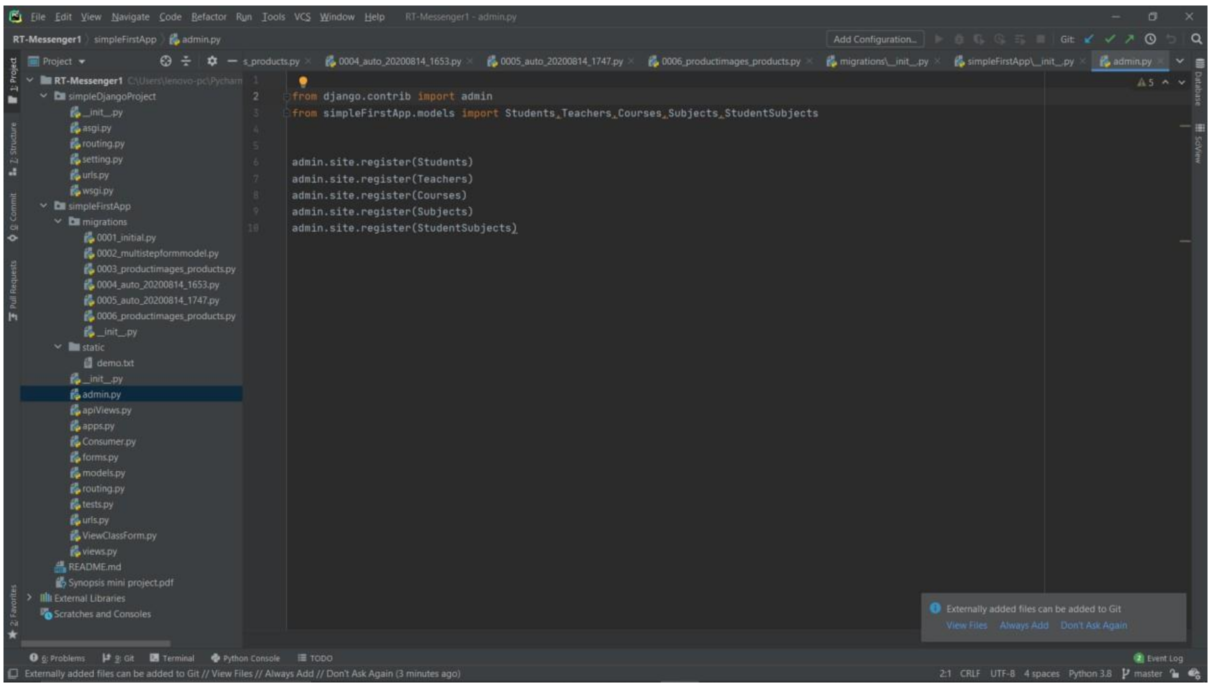
RT-Messenger1 - 0005_auto_20200814_1747.py
Externally added files can be added to Git
View Files
Always Add
Don't Ask Again

```python
from django.db import migrations


class Migration(migrations.Migration):

    dependencies = [
        ('simpleFirstApp', '0004_auto_20200814_1653'),
    ]

    operations = [
        migrations.DeleteModel(
            name='ProductImages',
        ),
        migrations.DeleteModel(
            name='Products',
        ),
    ]
```



RT-Messenger1 - 0006_productimages_products.py
Externally added files can be added to Git
View Files
Always Add
Don't Ask Again

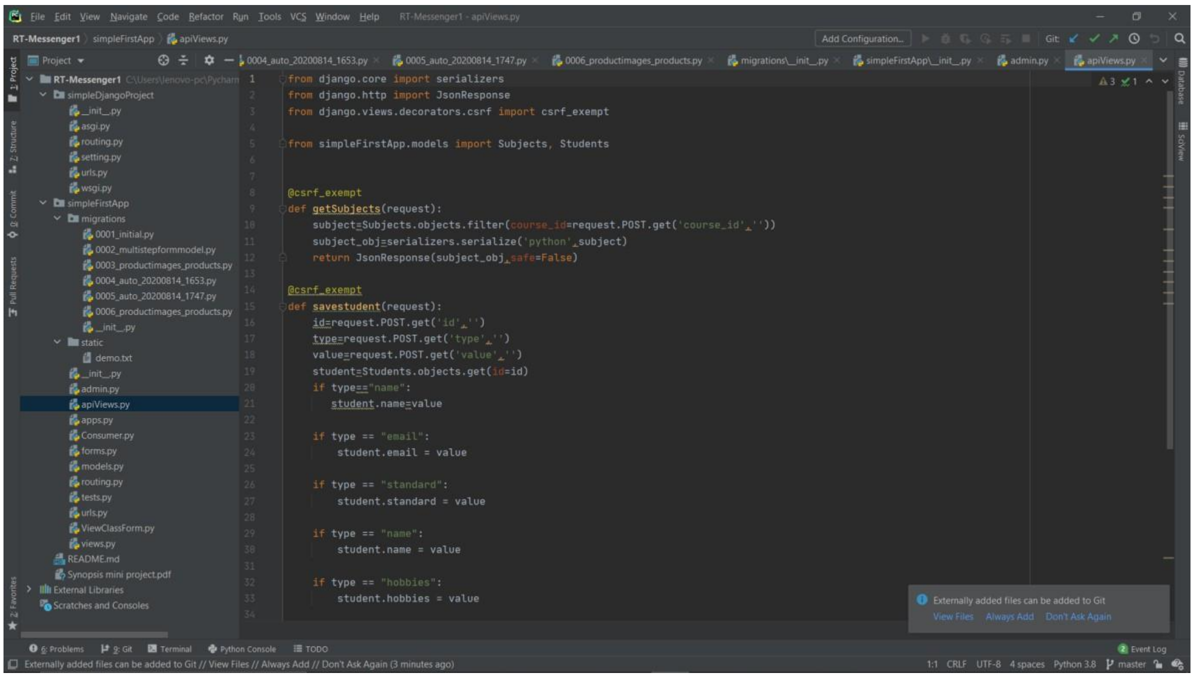
```python
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        ('simpleFirstApp', '0005_auto_20200814_1747'),
    ]

    operations = [
        migrations.CreateModel(
            name='Products',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('name', models.CharField(max_length=255)),
                ('desc', models.TextField()),
            ],
        ),
        migrations.CreateModel(
            name='ProductImages',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('image', models.FileField(max_length=255, upload_to='')),
                ('product_id', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='simpleFirstApp.Products')),
            ],
        ),
    ]
```



```python
from django.contrib import admin
from simpleFirstApp.models import Students,Teachers,Courses,Subjects,StudentSubjects


admin.site.register(Students)
admin.site.register(Teachers)
admin.site.register(Courses)
admin.site.register(Subjects)
admin.site.register(StudentSubjects)
```

Externally added files can be added to Git
View Files Always Add Don't Ask Again

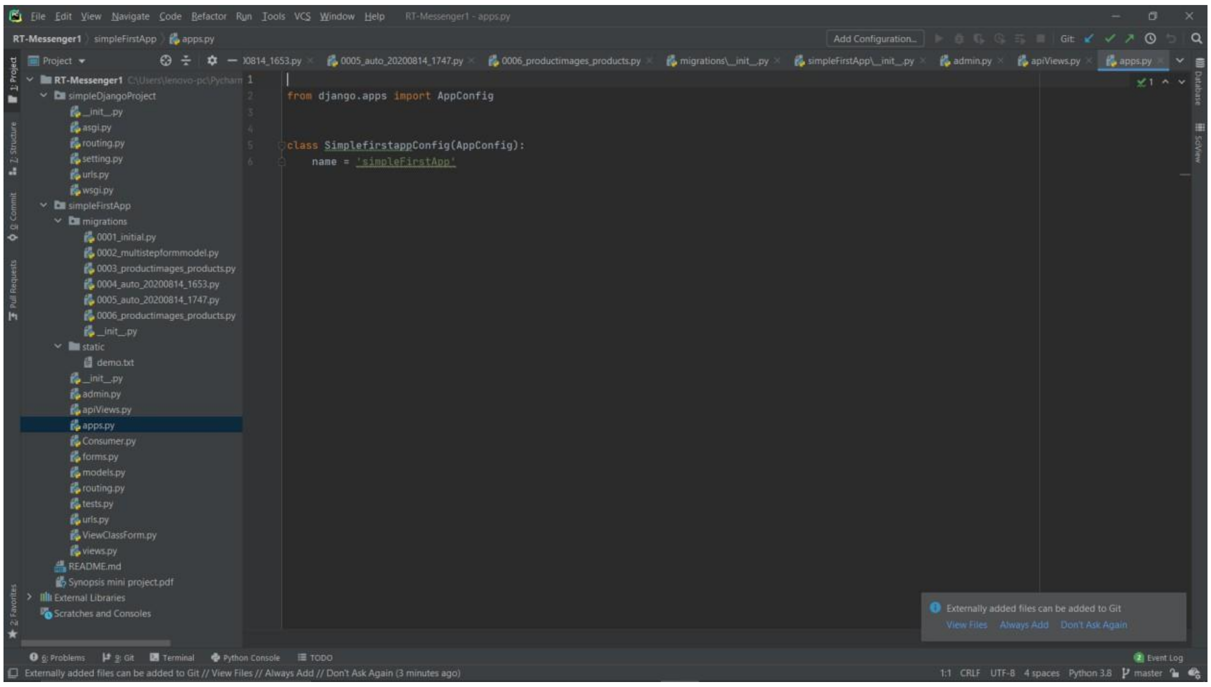
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (3 minutes ago)



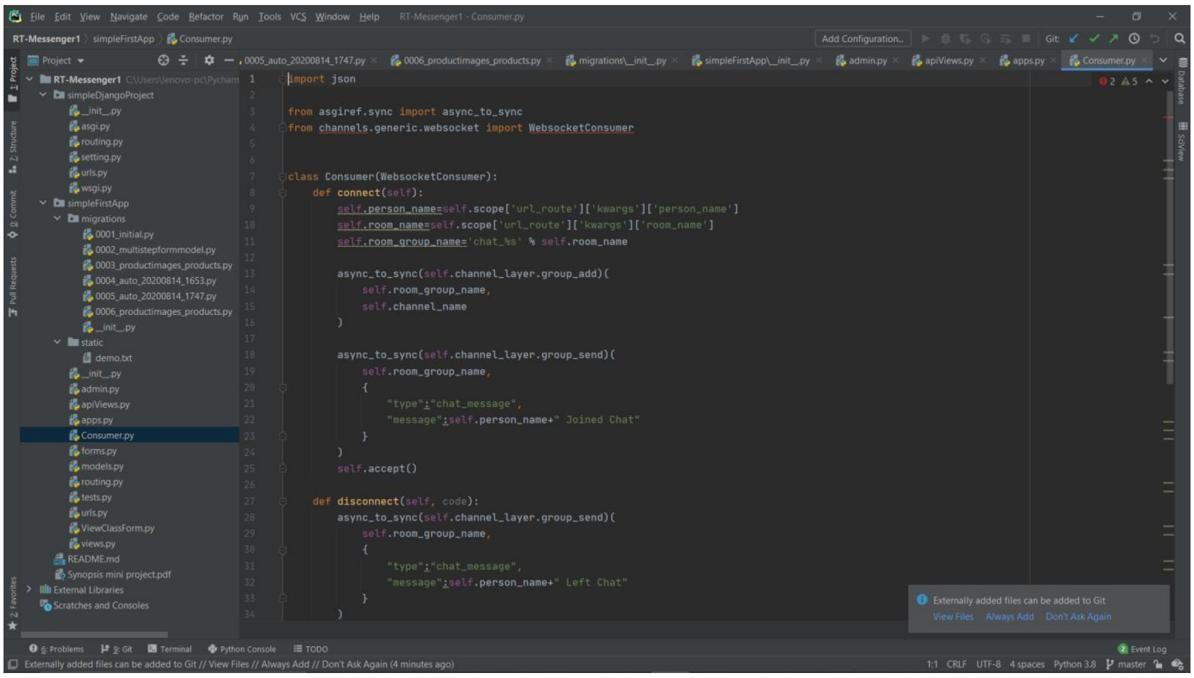
```python
from django.core import serializers
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

from simpleFirstApp.models import Subjects, Students


@csrf_exempt
def getSubjects(request):
    subject=Subjects.objects.filter(course_id=request.POST.get('course_id',''))
    subject_obj=serializers.serialize('python',subject)
    return JsonResponse(subject_obj,safe=False)

@csrf_exempt
def savestudent(request):
    id=request.POST.get('id','')
    type=request.POST.get('type','')
    value=request.POST.get('value','')
    student=Students.objects.get(id=id)
    if type=="name":
       student.name=value

    if type == "email":
        student.email = value

    if type == "standard":
        student.standard = value

    if type == "name":
        student.name = value

    if type == "hobbies":
        student.hobbies = value
```

Externally added files can be added to Git
View Files Always Add Don't Ask Again

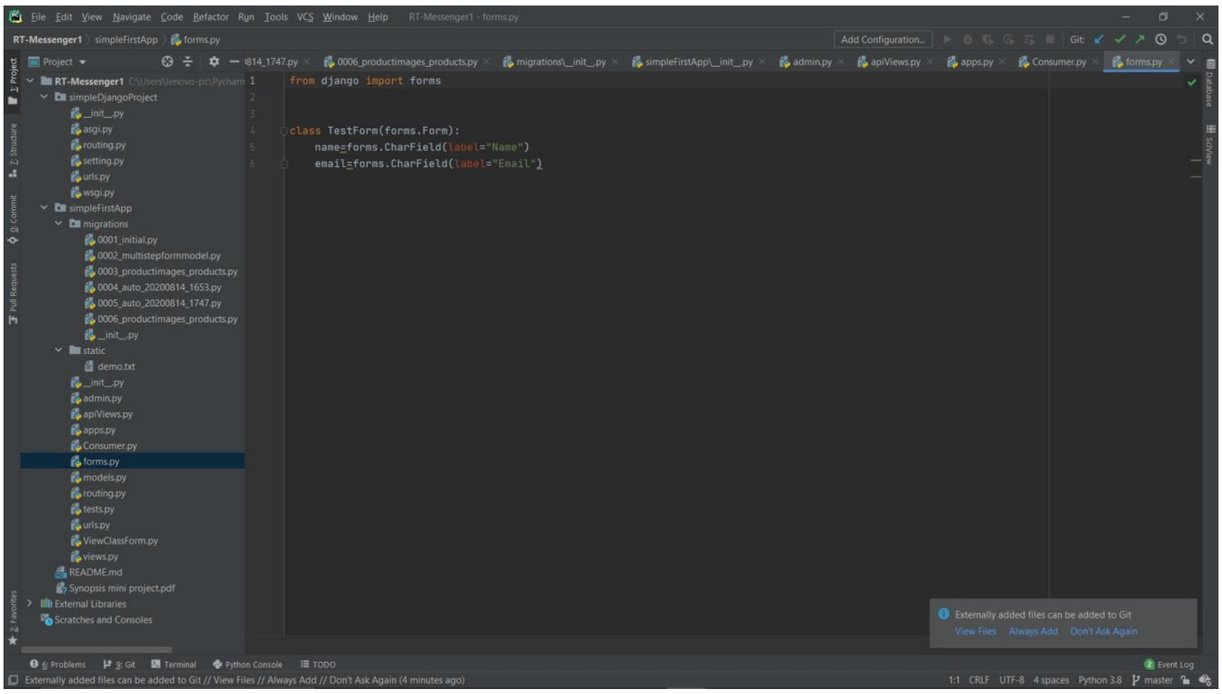
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (3 minutes ago)



```
from django.apps import AppConfig


class SimplefirstappConfig(AppConfig):
    name = 'simpleFirstApp'
```

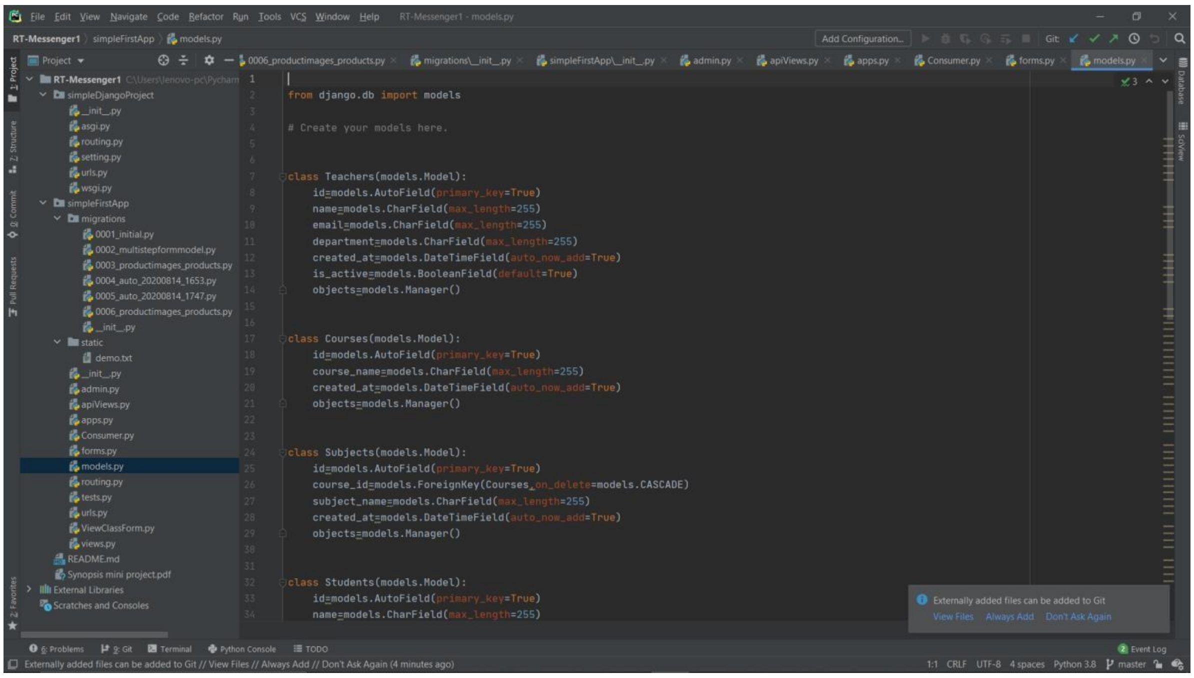


```
import json

from asgiref.sync import async_to_sync
from channels.generic.websocket import WebsocketConsumer


class Consumer(WebsocketConsumer):
    def connect(self):
        self.person_name=self.scope['url_route']['kwargs']['person_name']
        self.room_name=self.scope['url_route']['kwargs']['room_name']
        self.room_group_name='chat_%s' % self.room_name

        async_to_sync(self.channel_layer.group_add)(
            self.room_group_name,
            self.channel_name
        )

        async_to_sync(self.channel_layer.group_send)(
            self.room_group_name,
            {
                "type":"chat_message",
                "message":self.person_name+" Joined Chat"
            }
        )
        self.accept()

    def disconnect(self, code):
        async_to_sync(self.channel_layer.group_send)(
            self.room_group_name,
            {
                "type":"chat_message",
                "message":self.person_name+" Left Chat"
            }
        )
```



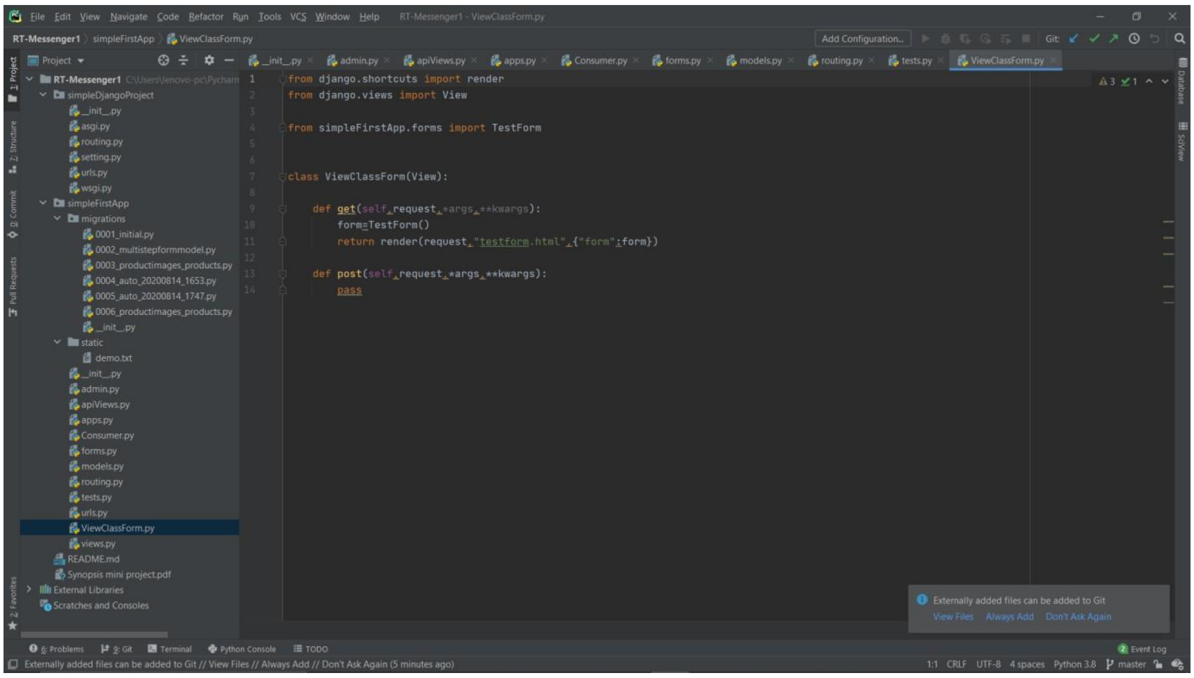
```
from django import forms


class TestForm(forms.Form):
    name=forms.CharField(label="Name")
    email=forms.CharField(label="Email")
```

Externally added files can be added to Git
View Files Always Add Don't Ask Again



```

from django.db import models

# Create your models here.


class Teachers(models.Model):
    id=models.AutoField(primary_key=True)
    name=models.CharField(max_length=255)
    email=models.CharField(max_length=255)
    department=models.CharField(max_length=255)
    created_at=models.DateTimeField(auto_now_add=True)
    is_active=models.BooleanField(default=True)
    objects=models.Manager()


class Courses(models.Model):
    id=models.AutoField(primary_key=True)
    course_name=models.CharField(max_length=255)
    created_at=models.DateTimeField(auto_now_add=True)
    objects=models.Manager()


class Subjects(models.Model):
    id=models.AutoField(primary_key=True)
    course_id=models.ForeignKey(Courses,on_delete=models.CASCADE)
    subject_name=models.CharField(max_length=255)
    created_at=models.DateTimeField(auto_now_add=True)
    objects=models.Manager()


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

Externally added files can be added to Git
View Files Always Add Don't Ask Again



```python
from django.shortcuts import render
from django.views import View

from simpleFirstApp.forms import TestForm


class ViewClassForm(View):

    def get(self,request,*args,**kwargs):
        form=TestForm()
        return render(request,"testform.html",{"form":form})

    def post(self,request,*args,**kwargs):
        pass
```

Externally added files can be added to Git

Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (5 minutes ago)