# PYTHON PROJECT
# DJANGO STUDY ROOM

DONE BY:

T. GOKULNATH

## INDEX

## ABOUT PYTHON: -

Python programming language was developed by Guido Van Rossum in February 1991. Python is based on or influenced with two programming languages:  ABC language, a teaching language created as a replacement of BASIC, and  Modula-3 Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

## ABOUT DJANGO: -

Django is a powerful and versatile web framework that enables rapid development of secure and maintainable websites. It includes an ORM (Object-Relational Mapping) system for database interactions, a robust URL routing system, and a templating engine for rendering dynamic content.

### Installation:

To get started with Django, you need to install it. You can use the following pip command:

```
pip install Django
```

## Creating a Django Project:

After installation, you can create a new Django project using the following command:

```
django-admin startproject projectname
```

This command will create a directory with the project structure

## Defining Models:

Django models define the structure of your database tables. You can create models by defining classes in the ‘models.py’ file of your app. For example:

```
from django.db import models

class MyModel(models.Model):
    field1 = models.CharField(max_length=100)
    field2 = models.IntegerField()
```

## Creating Migrations:

After defining models, you need to create migrations to apply changes to the database. Run the following commands:

```
python manage.py makemigrations
python manage.py migrate
```

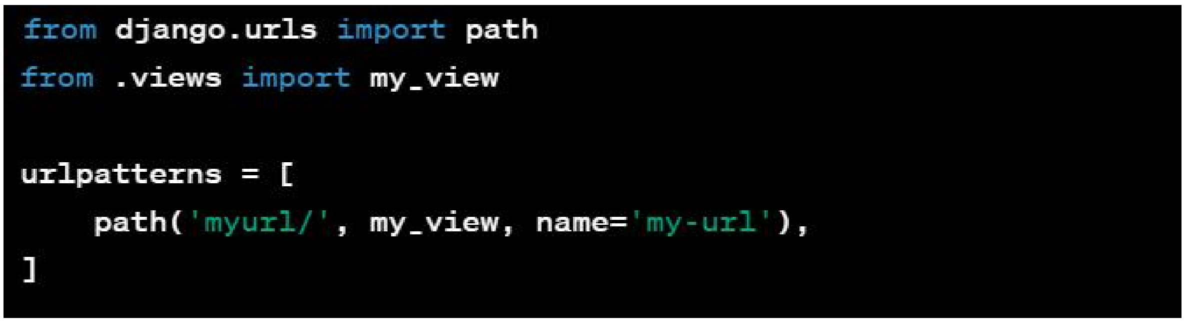
## Creating Views:

Views in Django handle the logic of your application. They receive requests, process them, and return responses. Views are defined in the ‘views.py’ file. Example:

```
from django.shortcuts import render
from django.http import HttpResponse

def my_view(request):
    return HttpResponse("Hello, Django!")
```

## URL Routing:

URL patterns are defined in the urls.py file. This tells Django which view to call for a given URL. Example:



```
from django.urls import path
from .views import my_view

urlpatterns = [
    path('myurl/', my_view, name='my-url'),
]
```

## Templates:

Django templates allow you to generate dynamic HTML. Create a templates directory in your app and use the render function in your views to render templates.

### Static Files:

Store your CSS, JavaScript, and other static files in the static directory. Django automatically collects these files and serves them efficiently.

## Admin Interface:

Django comes with a built-in admin interface. You can customize it to manage your models easily. Register your models in the admin.py file.

## Middleware:

Middleware components process requests and responses globally before reaching the view. You can use middleware for tasks like authentication, security, etc.

## Authentication and Authorization:

Django provides a robust authentication system. You can easily integrate user authentication and control access to views using decorators.

## PROJECT DESCRIPTION: -

The Study Room project is a web application developed using Django, HTML, and CSS.

The project includes the following main components:

- User authentication for account creation and login.
- Subjects model to categorize study rooms.
- Books model to represent individual study materials.
- Views to handle user interactions.
- HTML templates for rendering dynamic content.
- CSS styling for a visually appealing and user-friendly interface

### Features:

- User registration and authentication.
- Display of subjects with brief descriptions.
- User-specific reading lists.
- Ability to add books to the reading list.
- Responsive design for various devices.
- Clean and organized layout.

### Project Structure:

- Study room/: Django project folder.
- Base/: Django app folder.
- models.py: Defines the Subject and Book models.
- views.py: Contains views for handling user interactions.
- urls.py: Defines URL patterns for the app.
- templates/: Folder for HTML templates.
- static/: Folder for CSS and other static files.

### User Authentication:

The project utilizes Django's built-in authentication system for user registration and login.

• Users can create accounts or log in to personalize their study experience.

• Django's User model is extended to include additional fields.

### Views:

Home View:

• Displays a list of available subjects.

• Shows the user's reading list with the option to add books

### Templates (HTML):

Base Template:

• Includes the common structure (header, footer, navigation).

• Uses Django template tags for dynamic content.

Home Template:

• Displays the list of subjects.

• Shows the user's reading list with options to add books.

### CSS Styling:

• Responsive design for various devices.

• Calm and conducive colour scheme.

• Readable fonts for a comfortable reading experience.

• Clean and organized layout for easy navigation.

## SOFTWARE SPECIFICATION: -

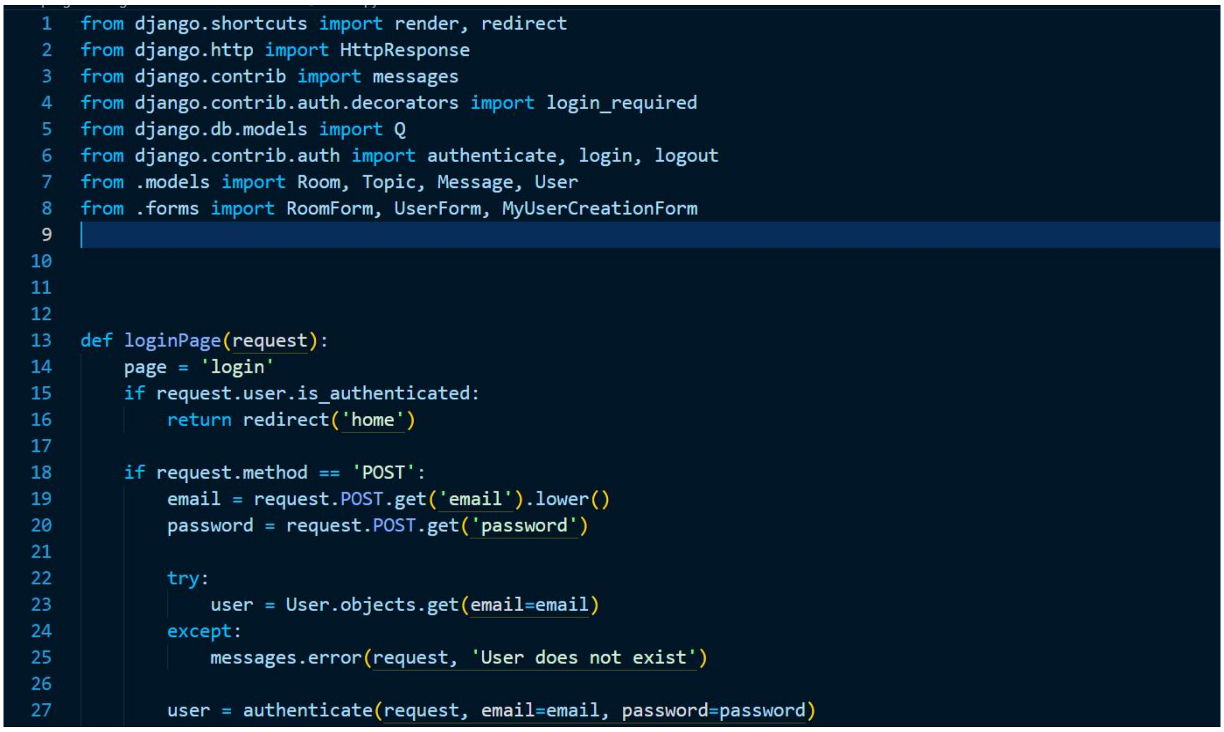
Operating system: Windows 11,

Platform: Visual studio code, Python idle 3.12 64 bits

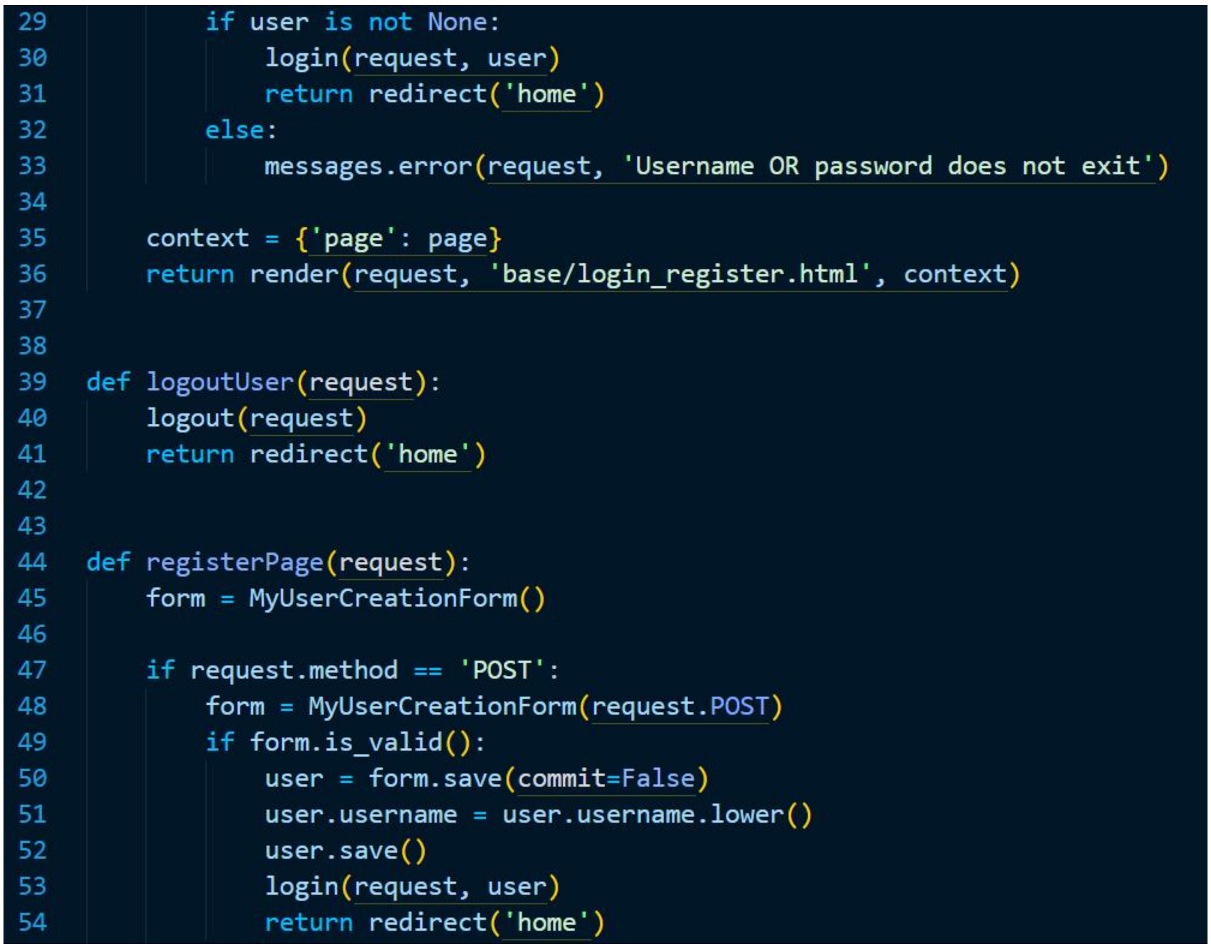
Language: Python Django, HTML, CSS

# SOURCE CODE: -

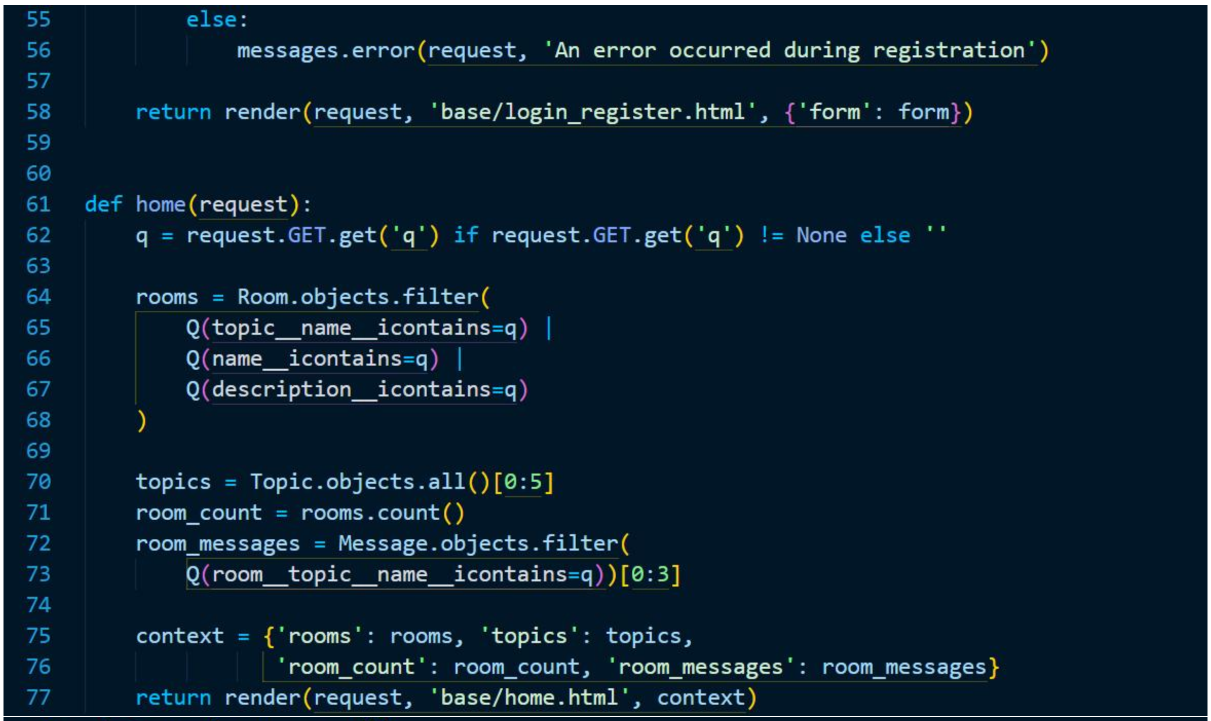
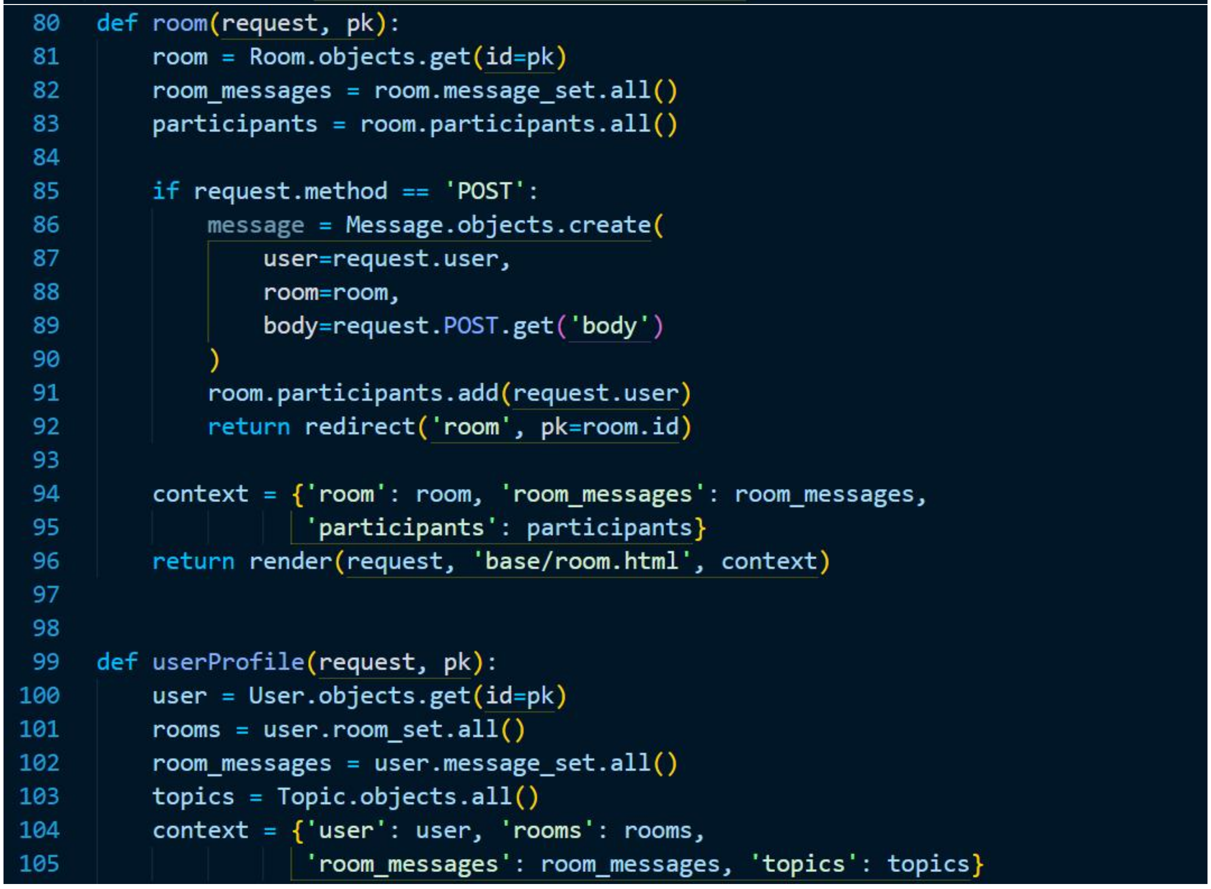
## views.py:

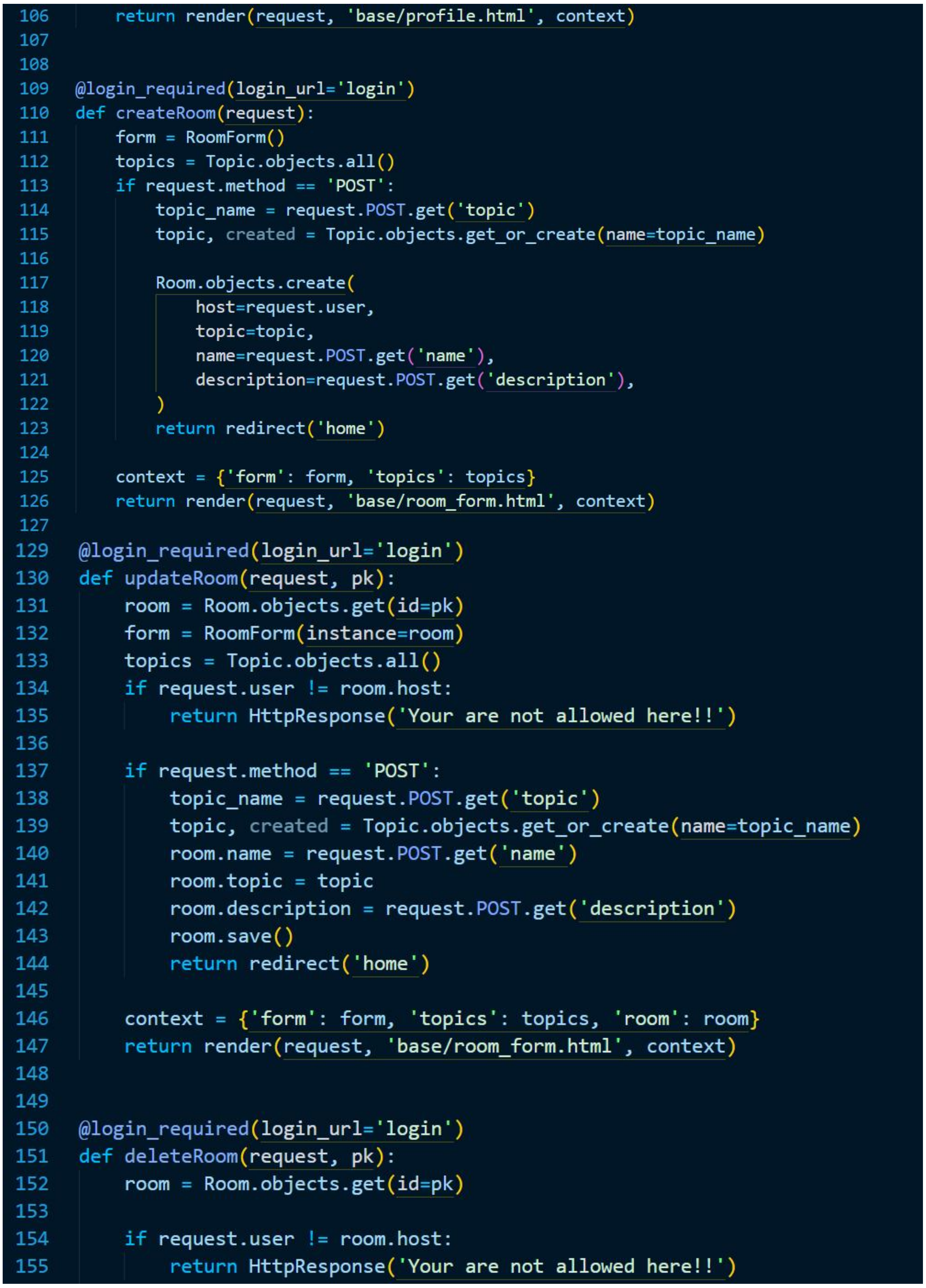


```python
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from django.contrib.auth import authenticate, login, logout
from .models import Room, Topic, Message, User
from .forms import RoomForm, UserForm, MyUserCreationForm




def loginPage(request):
    page = 'login'
    if request.user.is_authenticated:
        return redirect('home')

    if request.method == 'POST':
        email = request.POST.get('email').lower()
        password = request.POST.get('password')

        try:
            user = User.objects.get(email=email)
        except:
            messages.error(request, 'User does not exist')

        user = authenticate(request, email=email, password=password)
```

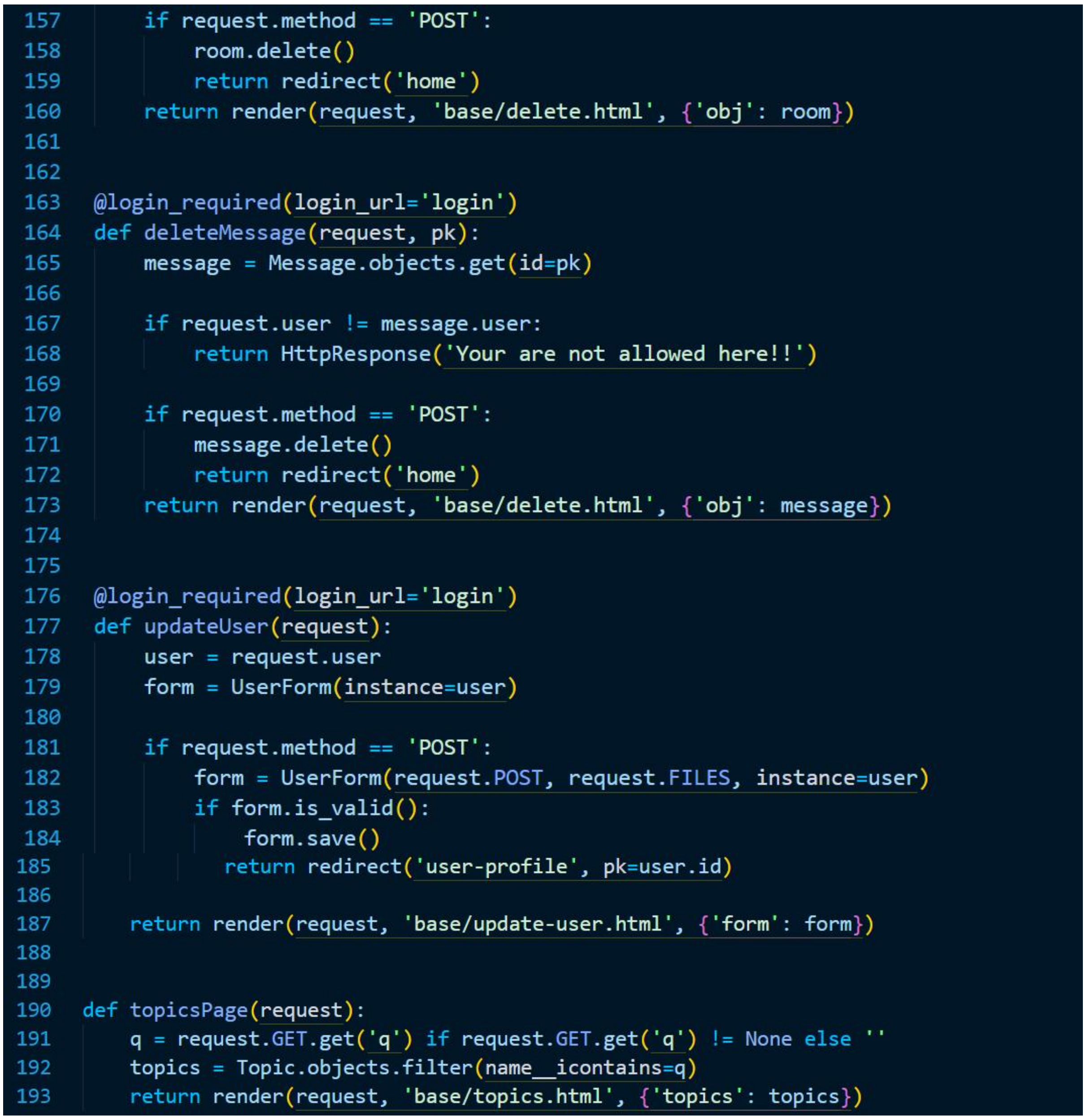


```python
        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Username OR password does not exit')

    context = {'page': page}
    return render(request, 'base/login_register.html', context)


def logoutUser(request):
    logout(request)
    return redirect('home')


def registerPage(request):
    form = MyUserCreationForm()

    if request.method == 'POST':
        form = MyUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.username = user.username.lower()
            user.save()
            login(request, user)
            return redirect('home')
```


```python
        else:
            messages.error(request, 'An error occurred during registration')

    return render(request, 'base/login_register.html', {'form': form})


def home(request):
    q = request.GET.get('q') if request.GET.get('q') != None else ''

    rooms = Room.objects.filter(
        Q(topic__name__icontains=q) |
        Q(name__icontains=q) |
        Q(description__icontains=q)
    )

    topics = Topic.objects.all()[0:5]
    room_count = rooms.count()
    room_messages = Message.objects.filter(
        Q(room__topic__name__icontains=q))[0:3]

    context = {'rooms': rooms, 'topics': topics,
               'room_count': room_count, 'room_messages': room_messages}
    return render(request, 'base/home.html', context)
```

```python
def room(request, pk):
    room = Room.objects.get(id=pk)
    room_messages = room.message_set.all()
    participants = room.participants.all()

    if request.method == 'POST':
        message = Message.objects.create(
            user=request.user,
            room=room,
            body=request.POST.get('body')
        )
        room.participants.add(request.user)
        return redirect('room', pk=room.id)

    context = {'room': room, 'room_messages': room_messages,
               'participants': participants}
    return render(request, 'base/room.html', context)


def userProfile(request, pk):
    user = User.objects.get(id=pk)
    rooms = user.room_set.all()
    room_messages = user.message_set.all()
    topics = Topic.objects.all()
    context = {'user': user, 'rooms': rooms,
               'room_messages': room_messages, 'topics': topics}
```

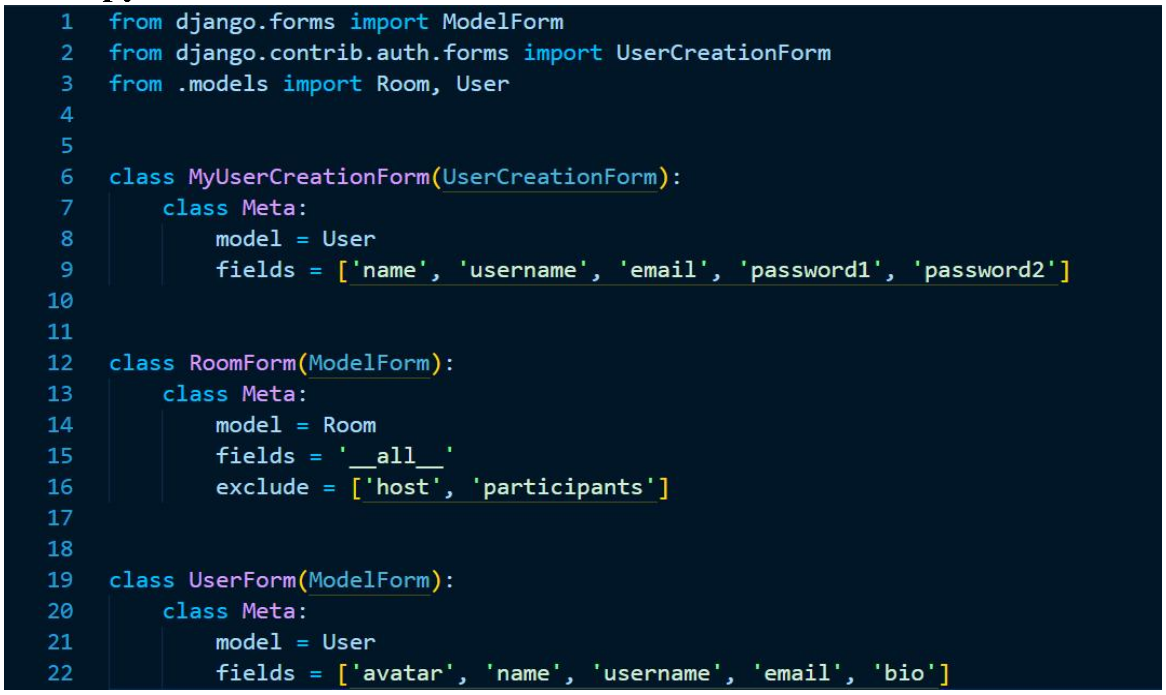


```python
    return render(request, 'base/profile.html', context)


@login_required(login_url='login')
def createRoom(request):
    form = RoomForm()
    topics = Topic.objects.all()
    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)

        Room.objects.create(
            host=request.user,
            topic=topic,
            name=request.POST.get('name'),
            description=request.POST.get('description'),
        )
        return redirect('home')

    context = {'form': form, 'topics': topics}
    return render(request, 'base/room_form.html', context)

@login_required(login_url='login')
def updateRoom(request, pk):
    room = Room.objects.get(id=pk)
    form = RoomForm(instance=room)
    topics = Topic.objects.all()
    if request.user != room.host:
        return HttpResponse('Your are not allowed here!!')

    if request.method == 'POST':
        topic_name = request.POST.get('topic')
        topic, created = Topic.objects.get_or_create(name=topic_name)
        room.name = request.POST.get('name')
        room.topic = topic
        room.description = request.POST.get('description')
        room.save()
        return redirect('home')

    context = {'form': form, 'topics': topics, 'room': room}
    return render(request, 'base/room_form.html', context)


@login_required(login_url='login')
def deleteRoom(request, pk):
    room = Room.objects.get(id=pk)

    if request.user != room.host:
        return HttpResponse('Your are not allowed here!!')
```



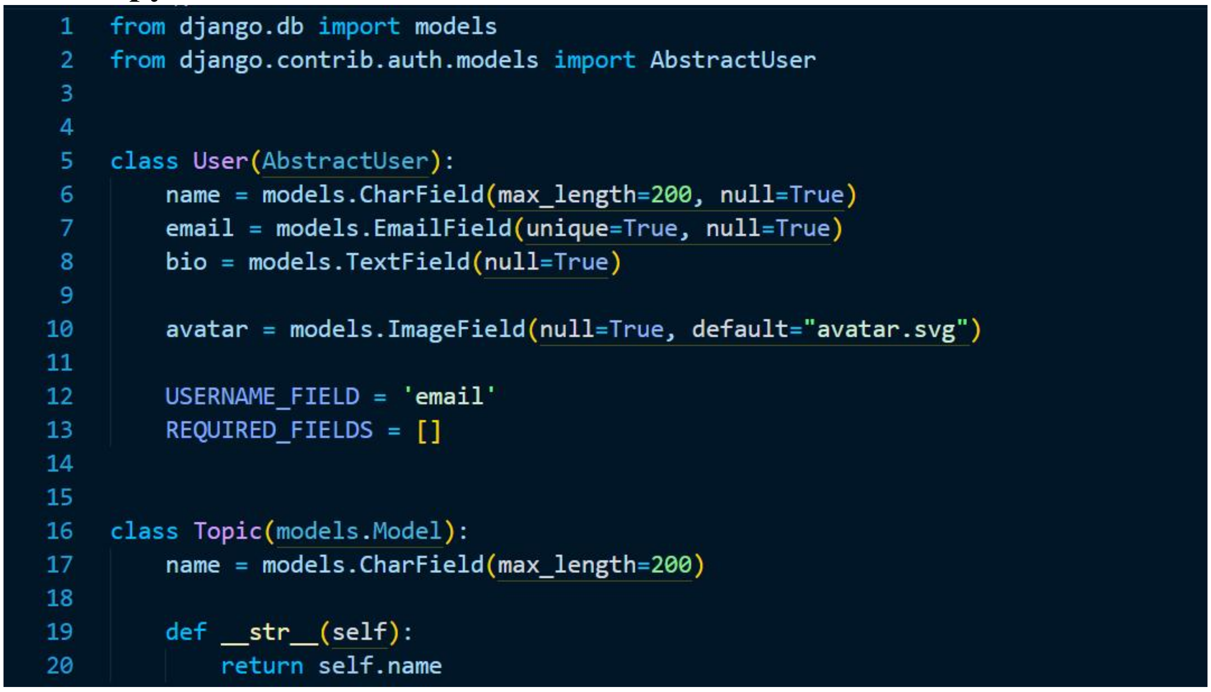
```python
    if request.method == 'POST':
        room.delete()
        return redirect('home')
    return render(request, 'base/delete.html', {'obj': room})


@login_required(login_url='login')
def deleteMessage(request, pk):
    message = Message.objects.get(id=pk)

    if request.user != message.user:
        return HttpResponse('Your are not allowed here!!')

    if request.method == 'POST':
        message.delete()
        return redirect('home')
    return render(request, 'base/delete.html', {'obj': message})


@login_required(login_url='login')
def updateUser(request):
    user = request.user
    form = UserForm(instance=user)

    if request.method == 'POST':
        form = UserForm(request.POST, request.FILES, instance=user)
        if form.is_valid():
            form.save()
            return redirect('user-profile', pk=user.id)

    return render(request, 'base/update-user.html', {'form': form})


def topicsPage(request):
    q = request.GET.get('q') if request.GET.get('q') != None else ''
    topics = Topic.objects.filter(name__icontains=q)
    return render(request, 'base/topics.html', {'topics': topics})
```

**Form.py:**

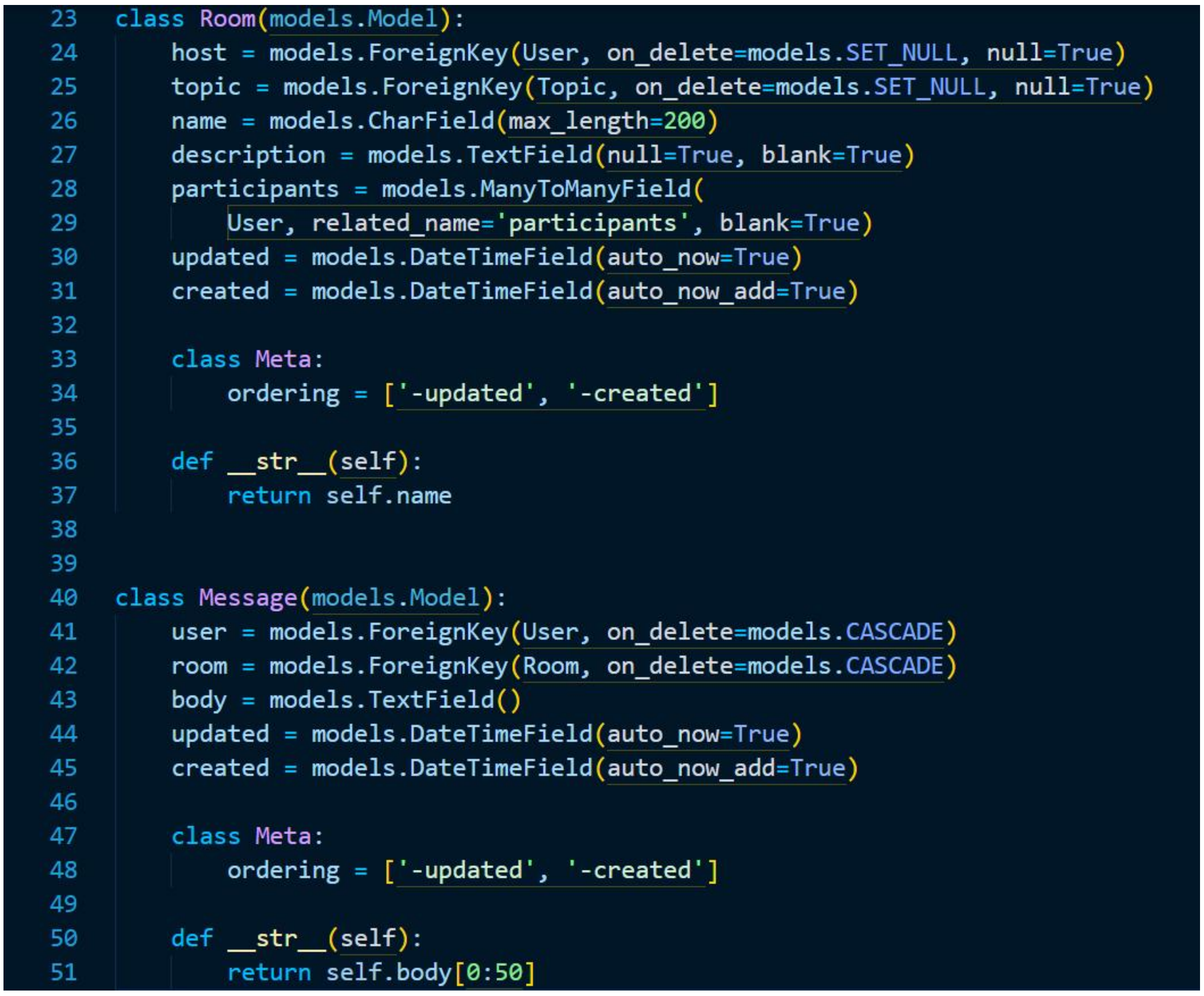


```python
from django.forms import ModelForm
from django.contrib.auth.forms import UserCreationForm
from .models import Room, User


class MyUserCreationForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['name', 'username', 'email', 'password1', 'password2']


class RoomForm(ModelForm):
    class Meta:
        model = Room
        fields = '__all__'
        exclude = ['host', 'participants']


class UserForm(ModelForm):
    class Meta:
        model = User
        fields = ['avatar', 'name', 'username', 'email', 'bio']
```

**Models.py:**

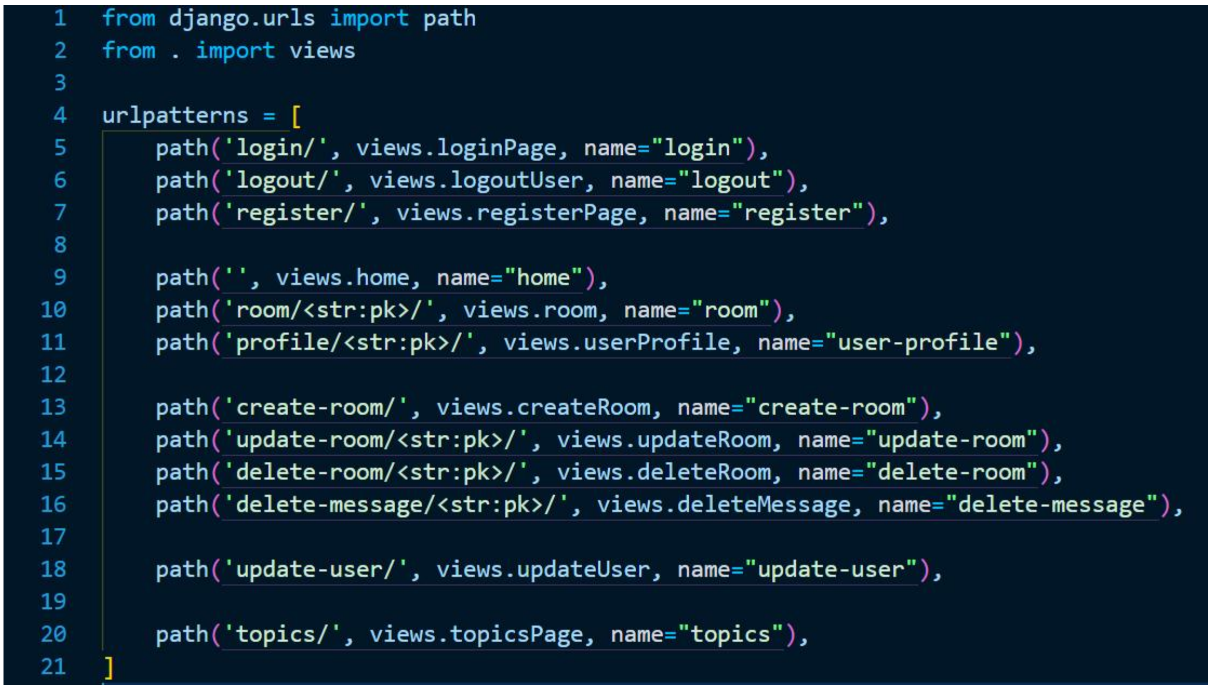


```python
from django.db import models
from django.contrib.auth.models import AbstractUser


class User(AbstractUser):
    name = models.CharField(max_length=200, null=True)
    email = models.EmailField(unique=True, null=True)
    bio = models.TextField(null=True)

    avatar = models.ImageField(null=True, default="avatar.svg")

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = []


class Topic(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name
```



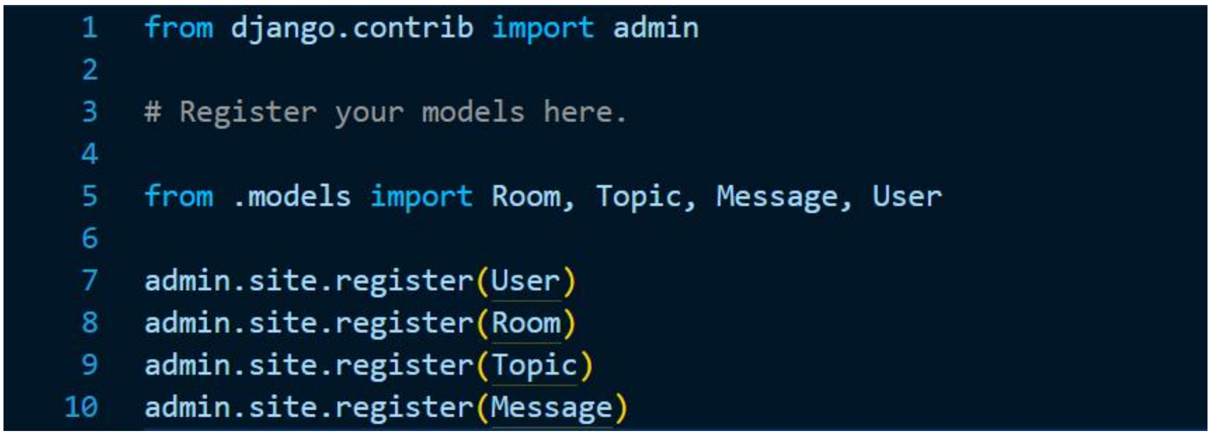
```python
class Room(models.Model):
    host = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    topic = models.ForeignKey(Topic, on_delete=models.SET_NULL, null=True)
    name = models.CharField(max_length=200)
    description = models.TextField(null=True, blank=True)
    participants = models.ManyToManyField(
        User, related_name='participants', blank=True)
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.name


class Message(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    room = models.ForeignKey(Room, on_delete=models.CASCADE)
    body = models.TextField()
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-updated', '-created']

    def __str__(self):
        return self.body[0:50]
```

**Urls.py:**



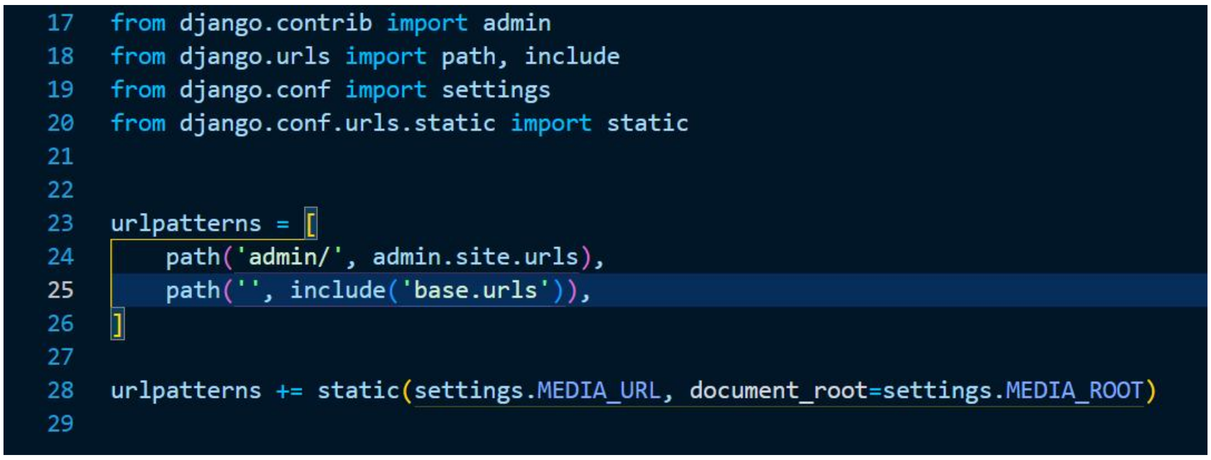
```python
from django.urls import path
from . import views

urlpatterns = [
    path('login/', views.loginPage, name="login"),
    path('logout/', views.logoutUser, name="logout"),
    path('register/', views.registerPage, name="register"),

    path('', views.home, name="home"),
    path('room/<str:pk>/', views.room, name="room"),
    path('profile/<str:pk>/', views.userProfile, name="user-profile"),

    path('create-room/', views.createRoom, name="create-room"),
    path('update-room/<str:pk>/', views.updateRoom, name="update-room"),
    path('delete-room/<str:pk>/', views.deleteRoom, name="delete-room"),
    path('delete-message/<str:pk>/', views.deleteMessage, name="delete-message"),

    path('update-user/', views.updateUser, name="update-user"),

    path('topics/', views.topicsPage, name="topics"),
]
```

**Admin.py:**



```
from django.contrib import admin

# Register your models here.

from .models import Room, Topic, Message, User

admin.site.register(User)
admin.site.register(Room)
admin.site.register(Topic)
admin.site.register(Message)
```

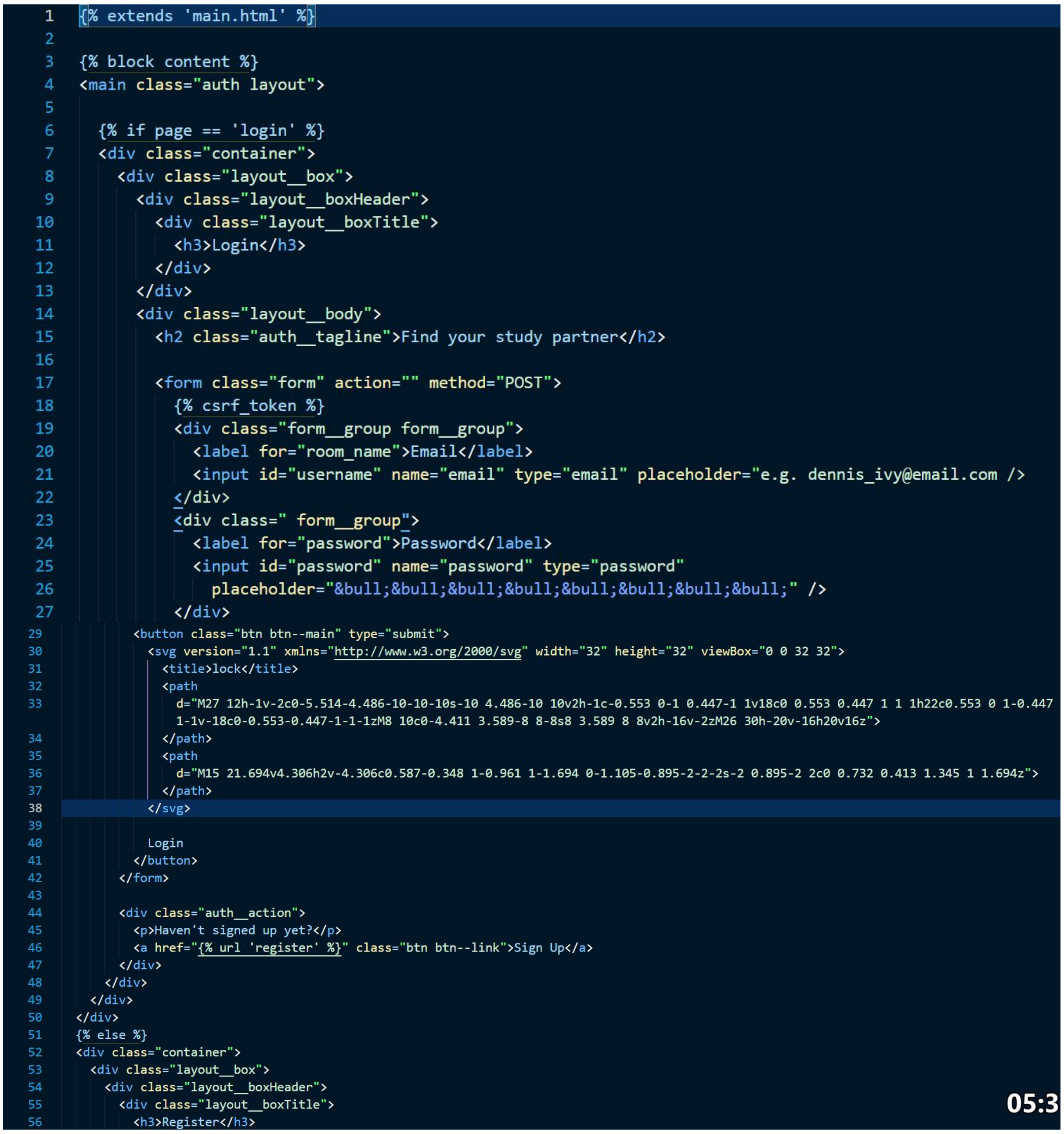
**URL to connect the app:**



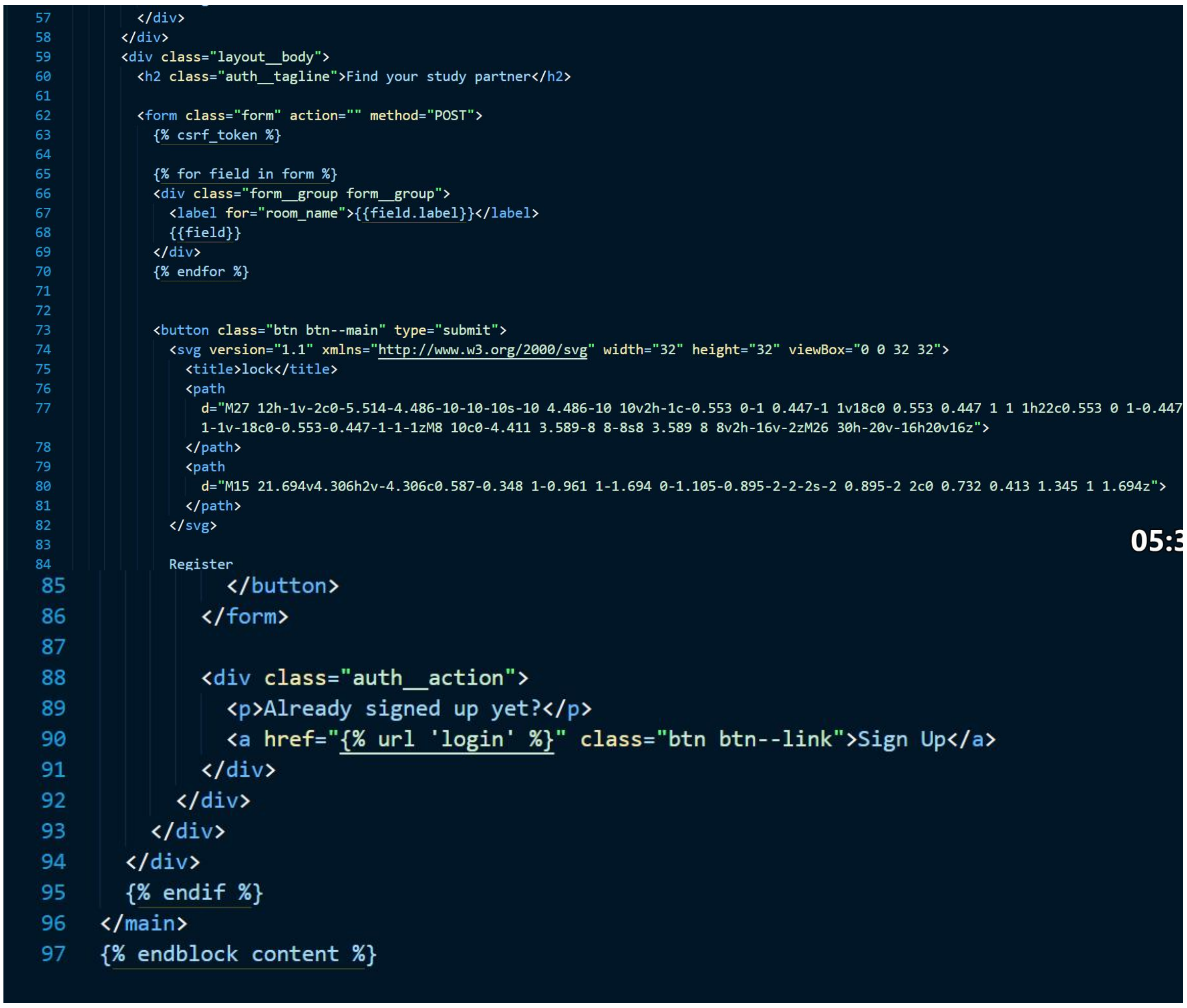
```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('base.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

## Html source code:

**Login.html:**

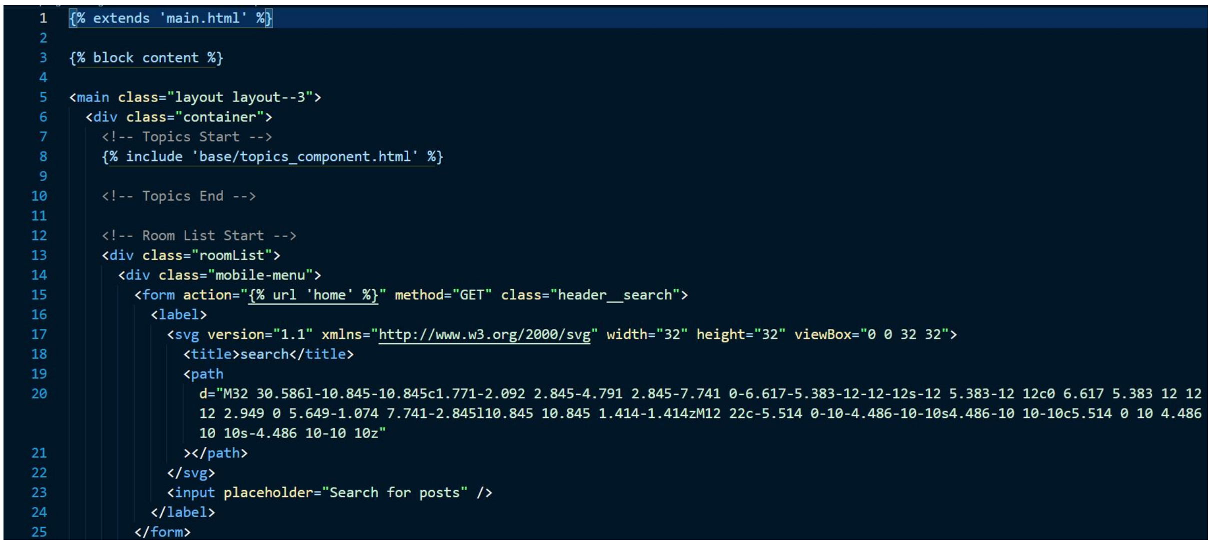


```
{% extends 'main.html' %}

{% block content %}
<main class="auth layout">

  {% if page == 'login' %}
  <div class="container">
    <div class="layout__box">
      <div class="layout__boxHeader">
        <div class="layout__boxTitle">
          <h3>Login</h3>
        </div>
      </div>
      <div class="layout__body">
        <h2 class="auth__tagline">Find your study partner</h2>

        <form class="form" action="" method="POST">
          {% csrf_token %}
          <div class="form__group form__group">
            <label for="room_name">Email</label>
            <input id="username" name="email" type="email" placeholder="e.g. dennis_ivy@email.com />
          </div>
          <div class=" form__group">
            <label for="password">Password</label>
            <input id="password" name="password" type="password"
              placeholder="&bull;&bull;&bull;&bull;&bull;&bull;&bull;&bull;" />
          </div>
          <button class="btn btn--main" type="submit">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>lock</title>
              <path
                d="M27 12h-1v-2c0-5.514-4.486-10-10-10s-10 4.486-10 10v2h-1c-0.553 0-1 0.447-1 1v18c0 0.553 0.447 1 1 1h22c0.553 0 1-0.447
                1-1v-18c0-0.553-0.447-1-1-1zM8 10c0-4.411 3.589-8 8-8s8 3.589 8 8v2h-16v-2zM26 30h-20v-16h20v16z">
              </path>
              <path
                d="M15 21.694v4.306h2v-4.306c0.587-0.348 1-0.961 1-1.694 0-1.105-0.895-2-2-2s-2 0.895-2 2c0 0.732 0.413 1.345 1 1.694z">
              </path>
            </svg>

            Login
          </button>
        </form>

        <div class="auth__action">
          <p>Haven't signed up yet?</p>
          <a href="{% url 'register' %}" class="btn btn--link">Sign Up</a>
        </div>
      </div>
    </div>
  </div>
  {% else %}
  <div class="container">
    <div class="layout__box">
      <div class="layout__boxHeader">
        <div class="layout__boxTitle">
          <h3>Register</h3>
```

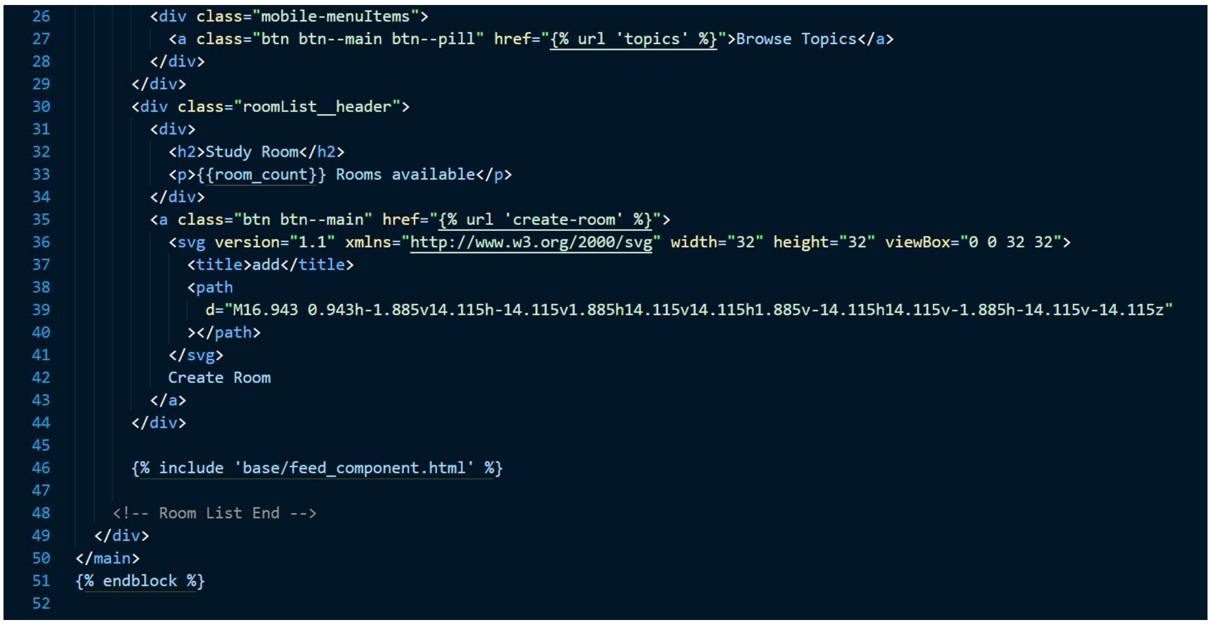


```
        </div>
      </div>
      <div class="layout__body">
        <h2 class="auth__tagline">Find your study partner</h2>

        <form class="form" action="" method="POST">
          {% csrf_token %}

          {% for field in form %}
          <div class="form__group form__group">
            <label for="room_name">{{field.label}}</label>
            {{field}}
          </div>
          {% endfor %}


          <button class="btn btn--main" type="submit">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>lock</title>
              <path
                d="M27 12h-1v-2c0-5.514-4.486-10-10-10s-10 4.486-10 10v2h-1c-0.553 0-1 0.447-1 1v18c0 0.553 0.447 1 1 1h22c0.553 0 1-0.447
                1-1v-18c0-0.553-0.447-1-1-1zM8 10c0-4.411 3.589-8 8-8s8 3.589 8 8v2h-16v-2zM26 30h-20v-16h20v16z">
              </path>
              <path
                d="M15 21.694v4.306h2v-4.306c0.587-0.348 1-0.961 1-1.694 0-1.105-0.895-2-2-2s-2 0.895-2 2c0 0.732 0.413 1.345 1 1.694z">
              </path>
            </svg>

            Register
          </button>
        </form>

        <div class="auth__action">
          <p>Already signed up yet?</p>
          <a href="{% url 'login' %}" class="btn btn--link">Sign Up</a>
        </div>
      </div>
    </div>
  </div>
  {% endif %}
</main>
{% endblock content %}
```

## Home.html:

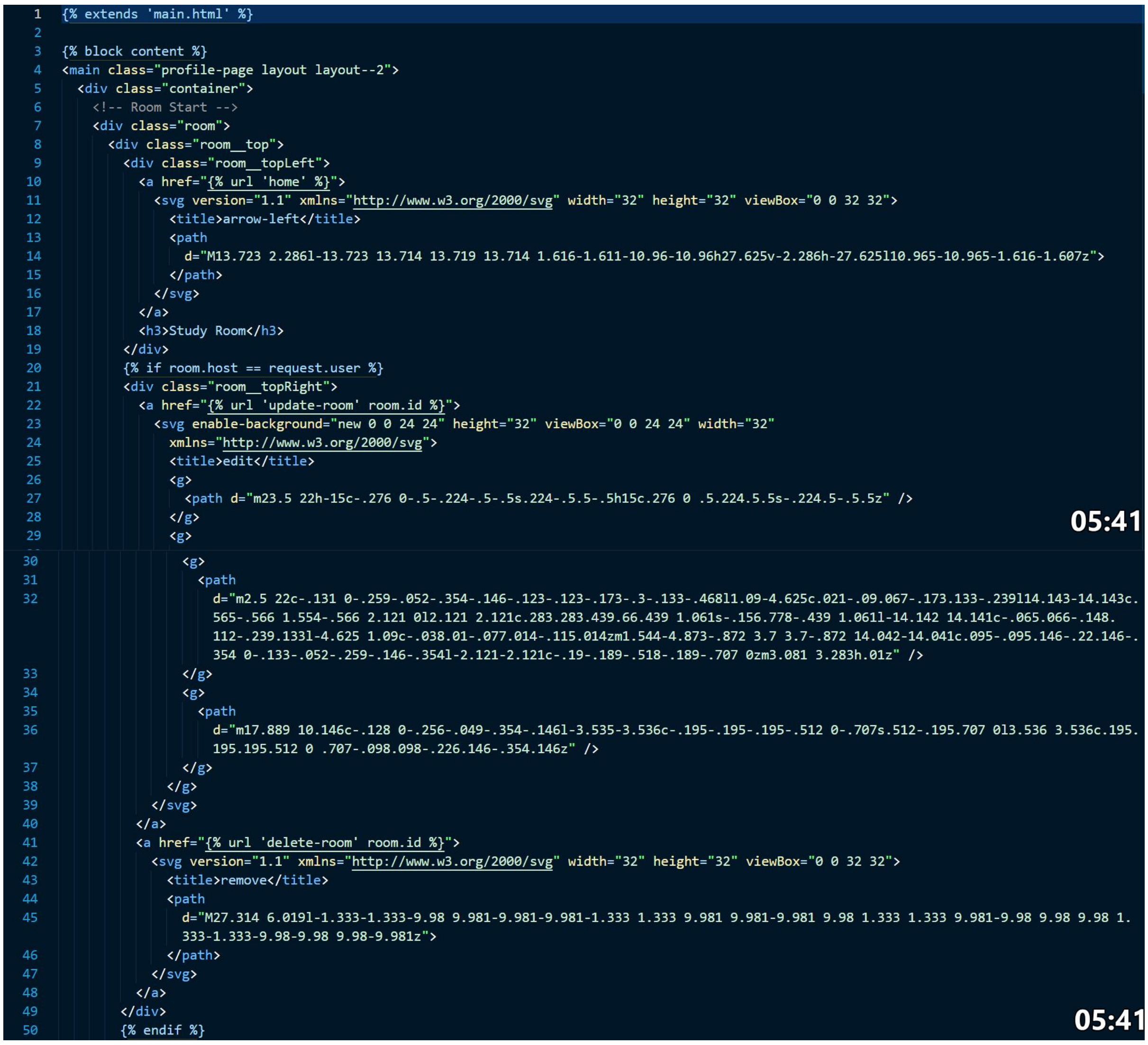


```
{% extends 'main.html' %}

{% block content %}

<main class="layout layout--3">
  <div class="container">
    <!-- Topics Start -->
    {% include 'base/topics_component.html' %}

    <!-- Topics End -->

    <!-- Room List Start -->
    <div class="roomList">
      <div class="mobile-menu">
        <form action="{% url 'home' %}" method="GET" class="header__search">
          <label>
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>search</title>
              <path
                d="M32 30.586l-10.845-10.845c1.771-2.092 2.845-4.791 2.845-7.741 0-6.617-5.383-12-12-12s-12 5.383-12 12c0 6.617 5.383 12 12
                12 2.949 0 5.649-1.074 7.741-2.845l10.845 10.845 1.414-1.414zM12 22c-5.514 0-10-4.486-10-10s4.486-10 10-10c5.514 0 10 4.486
                10 10s-4.486 10-10 10z"
              ></path>
            </svg>
            <input placeholder="Search for posts" />
          </label>
        </form>
```

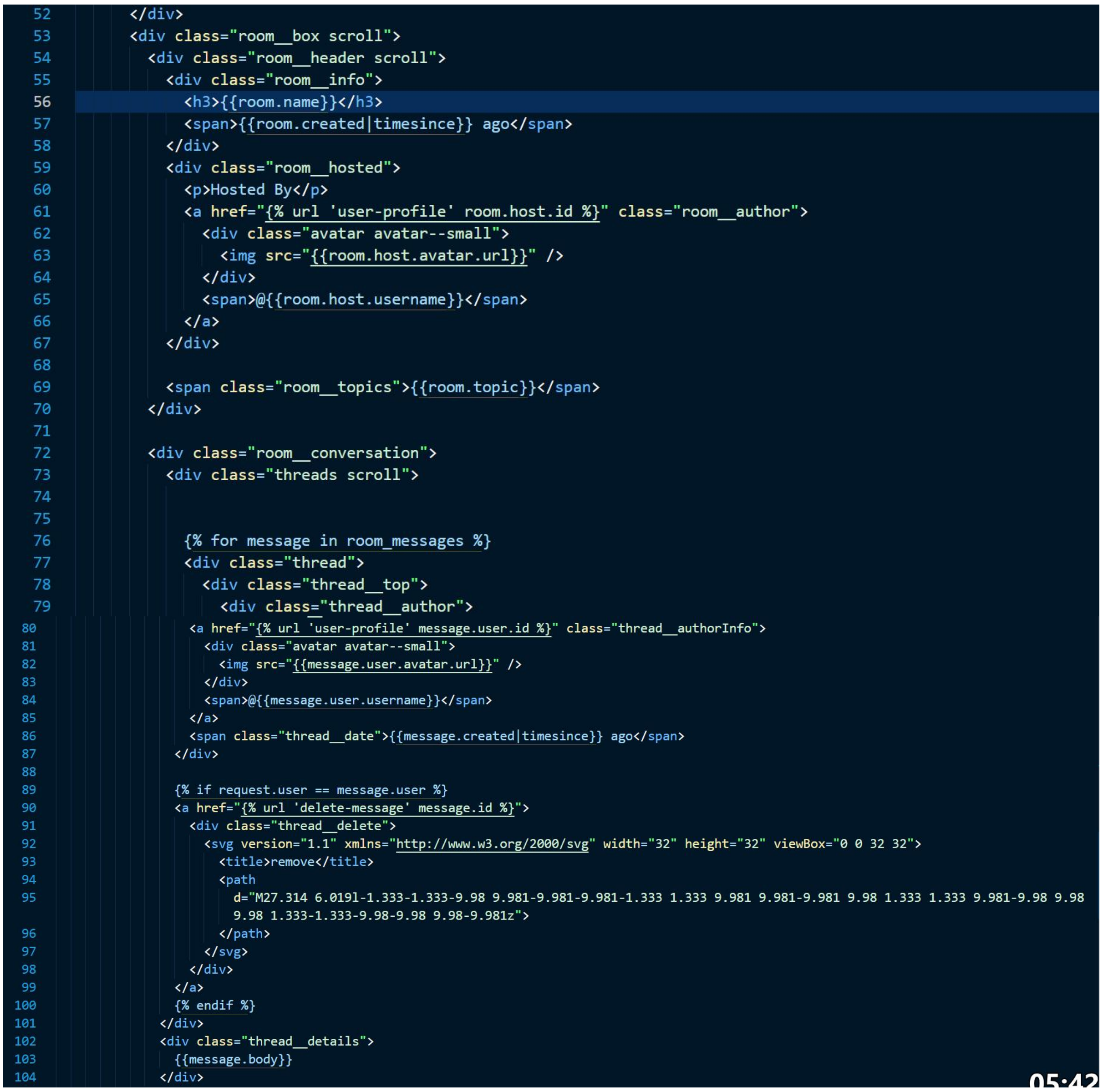


```html
        <div class="mobile-menuItems">
          <a class="btn btn--main btn--pill" href="{% url 'topics' %}">Browse Topics</a>
        </div>
      </div>
      <div class="roomList__header">
        <div>
          <h2>Study Room</h2>
          <p>{{room_count}} Rooms available</p>
        </div>
        <a class="btn btn--main" href="{% url 'create-room' %}">
          <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
            <title>add</title>
            <path
              d="M16.943 0.943h-1.885v14.115h-14.115v1.885h14.115v14.115h1.885v-14.115h14.115v-1.885h-14.115v-14.115z"
            ></path>
          </svg>
          Create Room
        </a>
      </div>

      {% include 'base/feed_component.html' %}

    <!-- Room List End -->
  </div>
</main>
{% endblock %}
```

## Room.html:

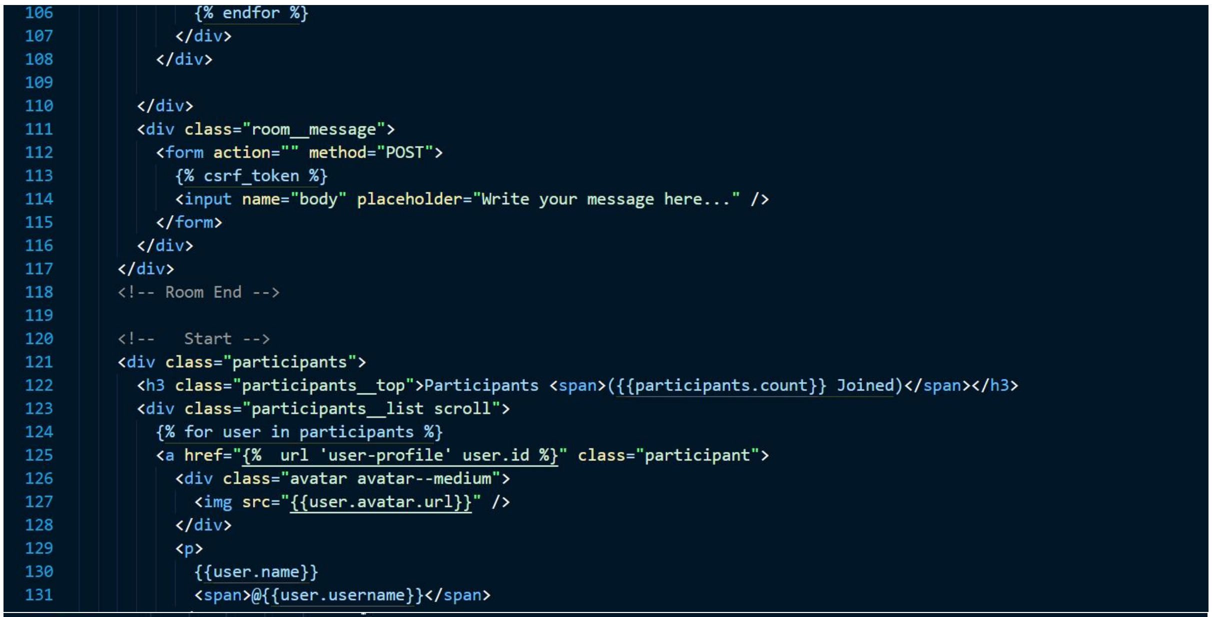


```html
{% extends 'main.html' %}

{% block content %}
<main class="profile-page layout layout--2">
  <div class="container">
    <!-- Room Start -->
    <div class="room">
      <div class="room__top">
        <div class="room__topLeft">
          <a href="{% url 'home' %}">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>arrow-left</title>
              <path
                d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z">
              </path>
            </svg>
          </a>
          <h3>Study Room</h3>
        </div>
        {% if room.host == request.user %}
        <div class="room__topRight">
          <a href="{% url 'update-room' room.id %}">
            <svg enable-background="new 0 0 24 24" height="32" viewBox="0 0 24 24" width="32"
              xmlns="http://www.w3.org/2000/svg">
              <title>edit</title>
              <g>
                <path d="m23.5 22h-15c-.276 0-.5-.224-.5-.5s.224-.5.5-.5h15c.276 0 .5.224.5.5s-.224.5-.5.5z" />
              </g>
              <g>
                <g>
                  <path
                    d="m2.5 22c-.131 0-.259-.052-.354-.146-.123-.123-.173-.3-.133-.468l1.09-4.625c.021-.09.067-.173.133-.239l14.143-14.143c.
                    565-.566 1.554-.566 2.121 0l2.121 2.121c.283.283.439.66.439 1.061s-.156.778-.439 1.061l-14.142 14.141c-.065.066-.148.
                    112-.239.133l-4.625 1.09c-.038.01-.077.014-.115.014zm1.544-4.873-.872 3.7 3.7-.872 14.042-14.041c.095-.095.146-.22.146-.
                    354 0-.133-.052-.259-.146-.354l-2.121-2.121c-.19-.189-.518-.189-.707 0zm3.081 3.283h.01z" />
                </g>
                <g>
                  <path
                    d="m17.889 10.146c-.128 0-.256-.049-.354-.146l-3.535-3.536c-.195-.195-.195-.512 0-.707s.512-.195.707 0l3.536 3.536c.195.
                    195.195.512 0 .707-.098.098-.226.146-.354.146z" />
                </g>
              </g>
            </svg>
          </a>
          <a href="{% url 'delete-room' room.id %}">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>remove</title>
              <path
                d="M27.314 6.019l-1.333-1.333-9.98 9.981-9.981-9.981-1.333 1.333 9.981 9.981-9.981 9.98 1.333 1.333 9.981-9.98 9.98 9.98 1.
                333-1.333-9.98-9.98 9.98-9.981z">
              </path>
            </svg>
          </a>
        </div>
        {% endif %}
```

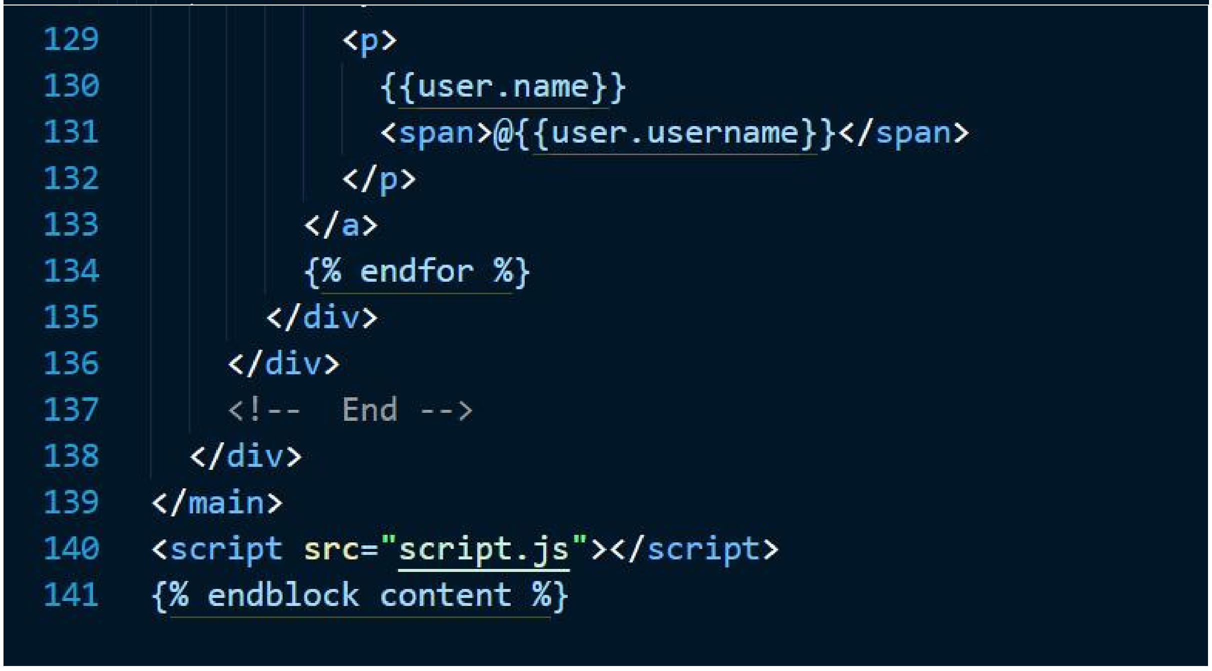


```
</div>
<div class="room__box scroll">
  <div class="room__header scroll">
    <div class="room__info">
      <h3>{{room.name}}</h3>
      <span>{{room.created|timesince}} ago</span>
    </div>
    <div class="room__hosted">
      <p>Hosted By</p>
      <a href="{% url 'user-profile' room.host.id %}" class="room__author">
        <div class="avatar avatar--small">
          <img src="{{room.host.avatar.url}}" />
        </div>
        <span>@{{room.host.username}}</span>
      </a>
    </div>

    <span class="room__topics">{{room.topic}}</span>
  </div>

  <div class="room__conversation">
    <div class="threads scroll">


      {% for message in room_messages %}
      <div class="thread">
        <div class="thread__top">
          <div class="thread__author">
            <a href="{% url 'user-profile' message.user.id %}" class="thread__authorInfo">
              <div class="avatar avatar--small">
                <img src="{{message.user.avatar.url}}" />
              </div>
              <span>@{{message.user.username}}</span>
            </a>
            <span class="thread__date">{{message.created|timesince}} ago</span>
          </div>

          {% if request.user == message.user %}
          <a href="{% url 'delete-message' message.id %}">
            <div class="thread__delete">
              <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
                <title>remove</title>
                <path
                  d="M27.314 6.019l-1.333-1.333-9.98 9.981-9.981-9.981-1.333 1.333 9.981 9.981-9.981 9.98 1.333 1.333 9.981-9.98 9.98
                  9.98 1.333-1.333-9.98-9.98 9.98-9.981z">
                </path>
              </svg>
            </div>
          </a>
          {% endif %}
        </div>
        <div class="thread__details">
          {{message.body}}
        </div>
```

05:42

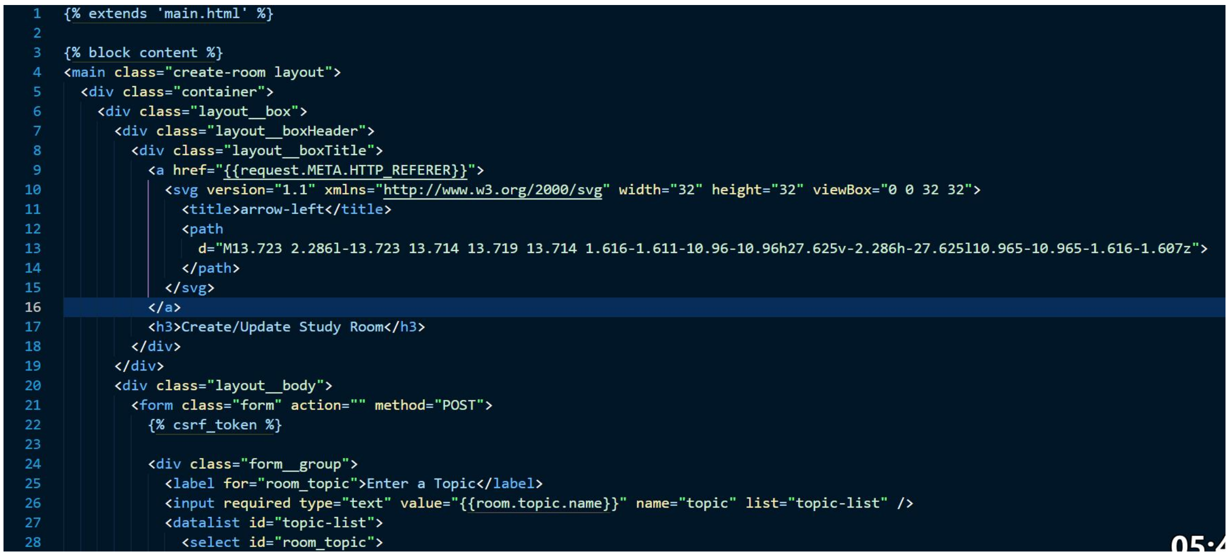


```
          {% endfor %}
        </div>
      </div>

    </div>
    <div class="room__message">
      <form action="" method="POST">
        {% csrf_token %}
        <input name="body" placeholder="Write your message here..." />
      </form>
    </div>
  </div>
  <!-- Room End -->

  <!--   Start -->
  <div class="participants">
    <h3 class="participants__top">Participants <span>({{participants.count}} Joined)</span></h3>
    <div class="participants__list scroll">
      {% for user in participants %}
      <a href="{% url 'user-profile' user.id %}" class="participant">
        <div class="avatar avatar--medium">
          <img src="{{user.avatar.url}}" />
        </div>
        <p>
          {{user.name}}
          <span>@{{user.username}}</span>
```

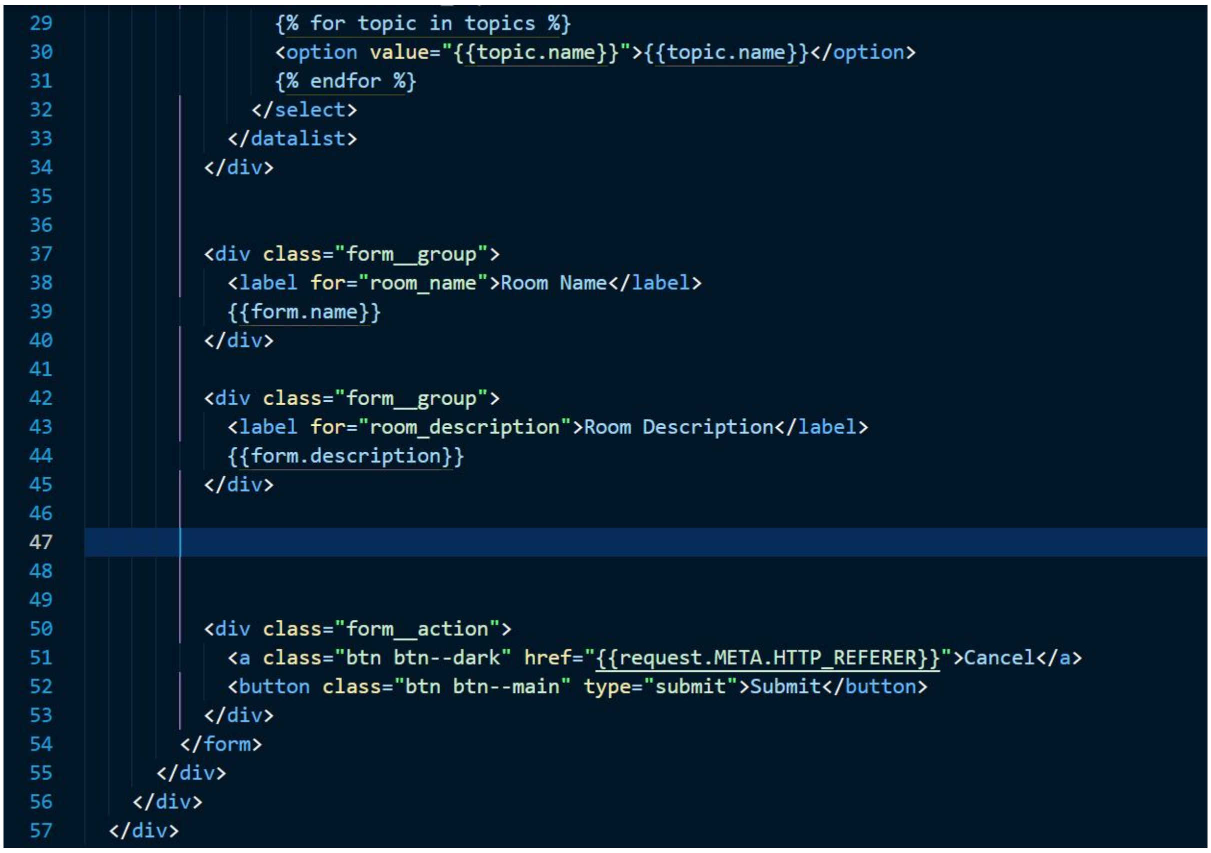


```
        <p>
          {{user.name}}
          <span>@{{user.username}}</span>
        </p>
      </a>
      {% endfor %}
    </div>
  </div>
  <!--   End -->
</div>
</main>
<script src="script.js"></script>
{% endblock content %}
```

## Room_form.html:

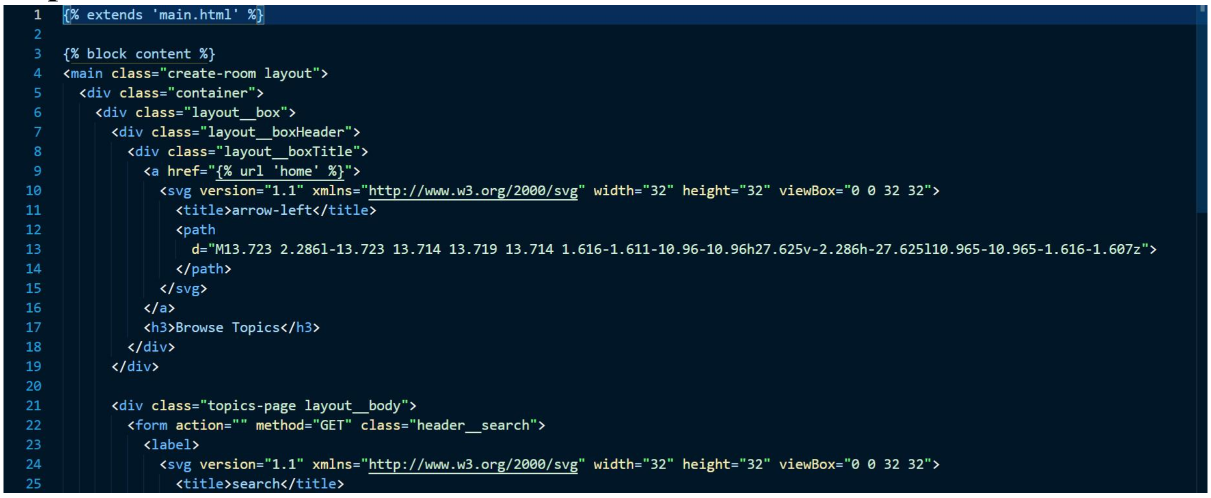


```
{% extends 'main.html' %}

{% block content %}
<main class="create-room layout">
  <div class="container">
    <div class="layout__box">
      <div class="layout__boxHeader">
        <div class="layout__boxTitle">
          <a href="{{request.META.HTTP_REFERER}}">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>arrow-left</title>
              <path
                d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z">
              </path>
            </svg>
          </a>
          <h3>Create/Update Study Room</h3>
        </div>
      </div>
      <div class="layout__body">
        <form class="form" action="" method="POST">
          {% csrf_token %}

          <div class="form__group">
            <label for="room_topic">Enter a Topic</label>
            <input required type="text" value="{{room.topic.name}}" name="topic" list="topic-list" />
            <datalist id="topic-list">
              <select id="room_topic">
```

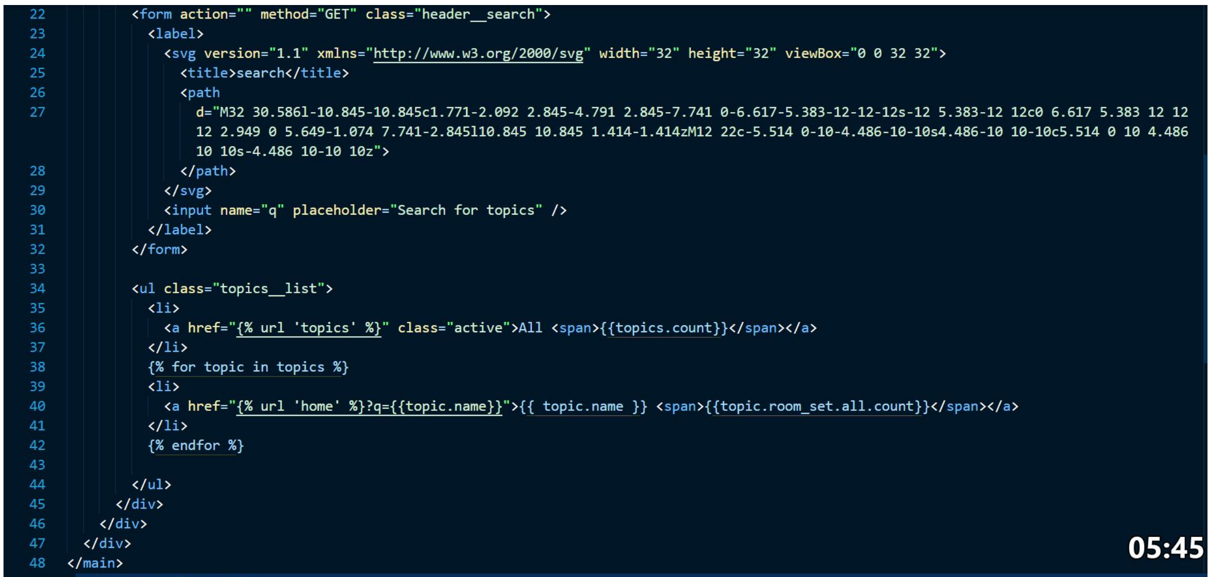


```
              {% for topic in topics %}
              <option value="{{topic.name}}">{{topic.name}}</option>
              {% endfor %}
            </select>
          </datalist>
        </div>


        <div class="form__group">
          <label for="room_name">Room Name</label>
          {{form.name}}
        </div>

        <div class="form__group">
          <label for="room_description">Room Description</label>
          {{form.description}}
        </div>




        <div class="form__action">
          <a class="btn btn--dark" href="{{request.META.HTTP_REFERER}}">Cancel</a>
          <button class="btn btn--main" type="submit">Submit</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

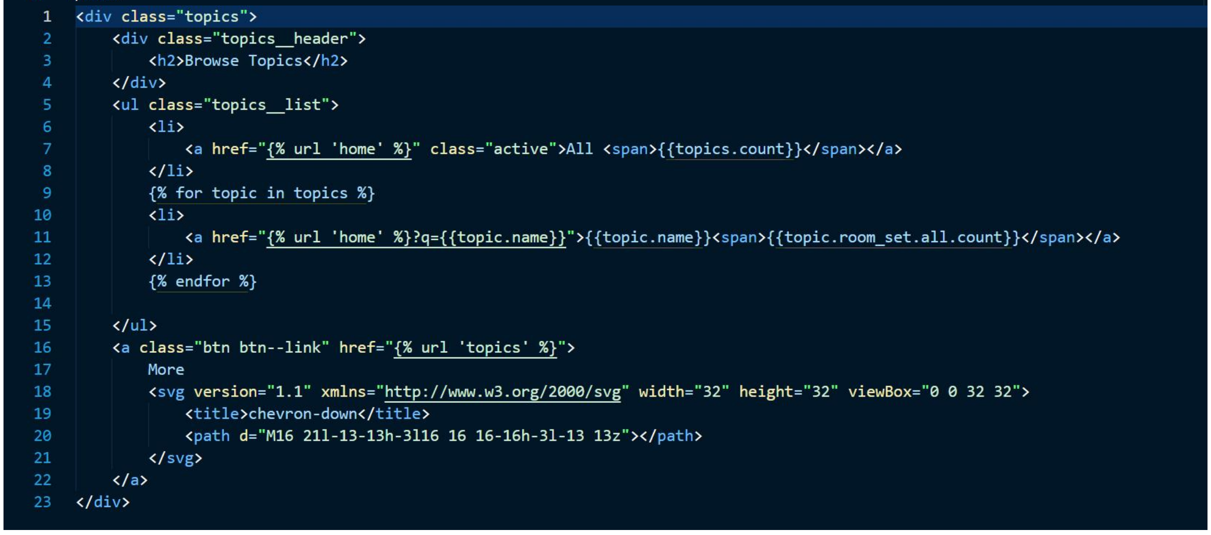
## Topics.html:



```
{% extends 'main.html' %}

{% block content %}
<main class="create-room layout">
  <div class="container">
    <div class="layout__box">
      <div class="layout__boxHeader">
        <div class="layout__boxTitle">
          <a href="{% url 'home' %}">
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>arrow-left</title>
              <path
                d="M13.723 2.286l-13.723 13.714 13.719 13.714 1.616-1.611-10.96-10.96h27.625v-2.286h-27.625l10.965-10.965-1.616-1.607z">
              </path>
            </svg>
          </a>
          <h3>Browse Topics</h3>
        </div>
      </div>

      <div class="topics-page layout__body">
        <form action="" method="GET" class="header__search">
          <label>
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>search</title>
```

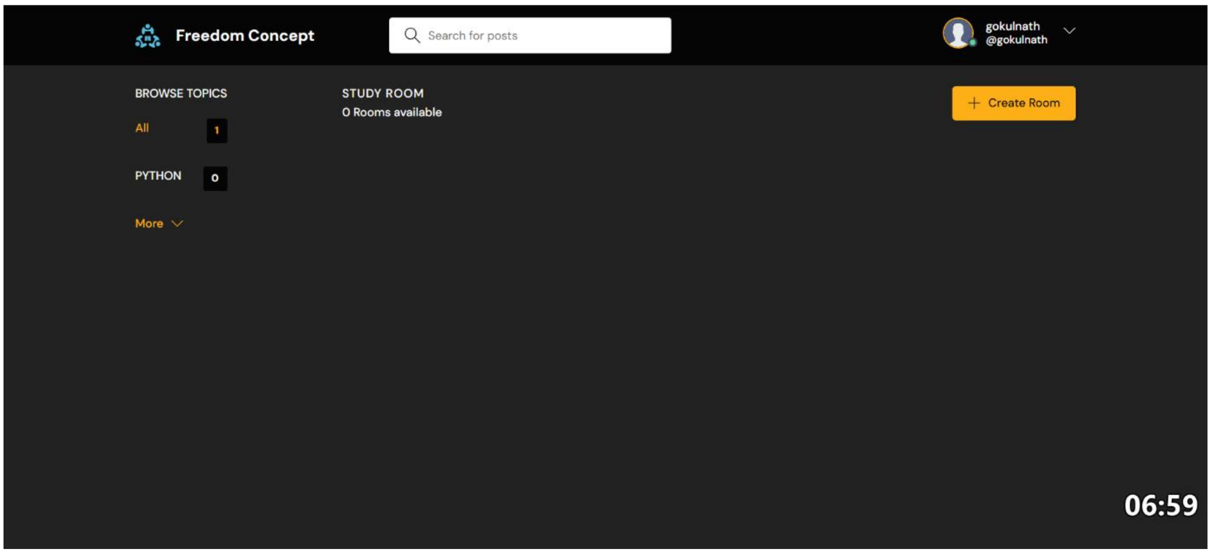


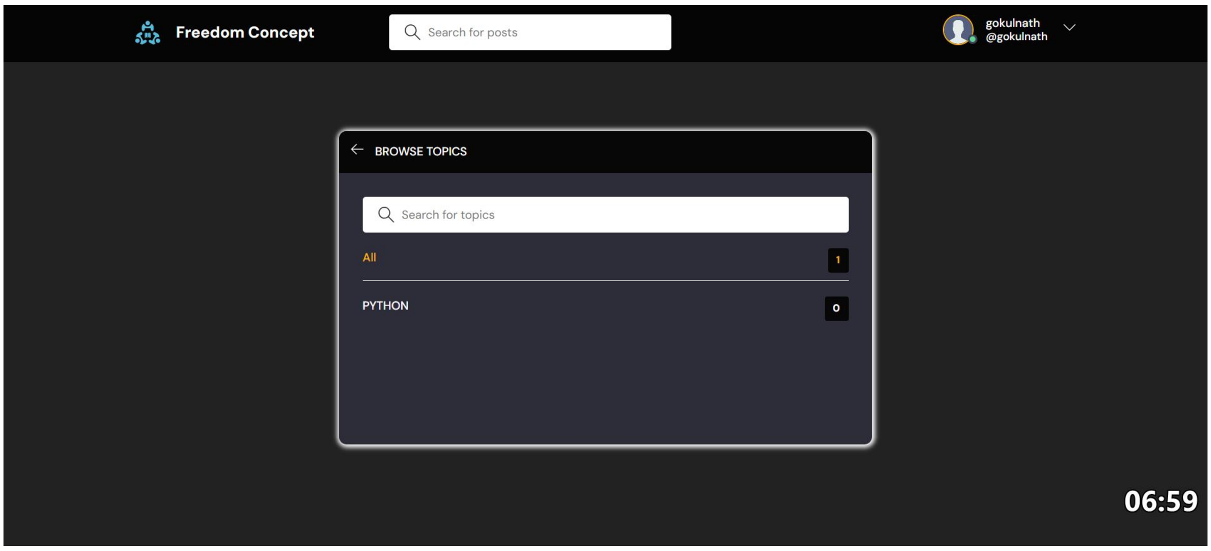
```
        <form action="" method="GET" class="header__search">
          <label>
            <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
              <title>search</title>
              <path
                d="M32 30.586l-10.845-10.845c1.771-2.092 2.845-4.791 2.845-7.741 0-6.617-5.383-12-12-12s-12 5.383-12 12c0 6.617 5.383 12 12
                12 2.949 0 5.649-1.074 7.741-2.845l10.845 10.845 1.414-1.414zM12 22c-5.514 0-10-4.486-10-10s4.486-10 10-10c5.514 0 10 4.486
                10 10s-4.486 10-10 10z">
              </path>
            </svg>
            <input name="q" placeholder="Search for topics" />
          </label>
        </form>

        <ul class="topics__list">
          <li>
            <a href="{% url 'topics' %}" class="active">All <span>{{topics.count}}</span></a>
          </li>
          {% for topic in topics %}
          <li>
            <a href="{% url 'home' %}?q={{topic.name}}">{{ topic.name }} <span>{{topic.room_set.all.count}}</span></a>
          </li>
          {% endfor %}

        </ul>
      </div>
    </div>
  </div>
</main>
```



```
<div class="topics">
    <div class="topics__header">
        <h2>Browse Topics</h2>
    </div>
    <ul class="topics__list">
        <li>
            <a href="{% url 'home' %}" class="active">All <span>{{topics.count}}</span></a>
        </li>
        {% for topic in topics %}
        <li>
            <a href="{% url 'home' %}?q={{topic.name}}">{{topic.name}}<span>{{topic.room_set.all.count}}</span></a>
        </li>
        {% endfor %}

    </ul>
    <a class="btn btn--link" href="{% url 'topics' %}">
        More
        <svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="32" height="32" viewBox="0 0 32 32">
            <title>chevron-down</title>
            <path d="M16 21l-13-13h-3l16 16 16-16h-3l-13 13z"></path>
        </svg>
    </a>
</div>
```

**SAMPLE OUTPUT:**




Freedom Concept

Search for posts

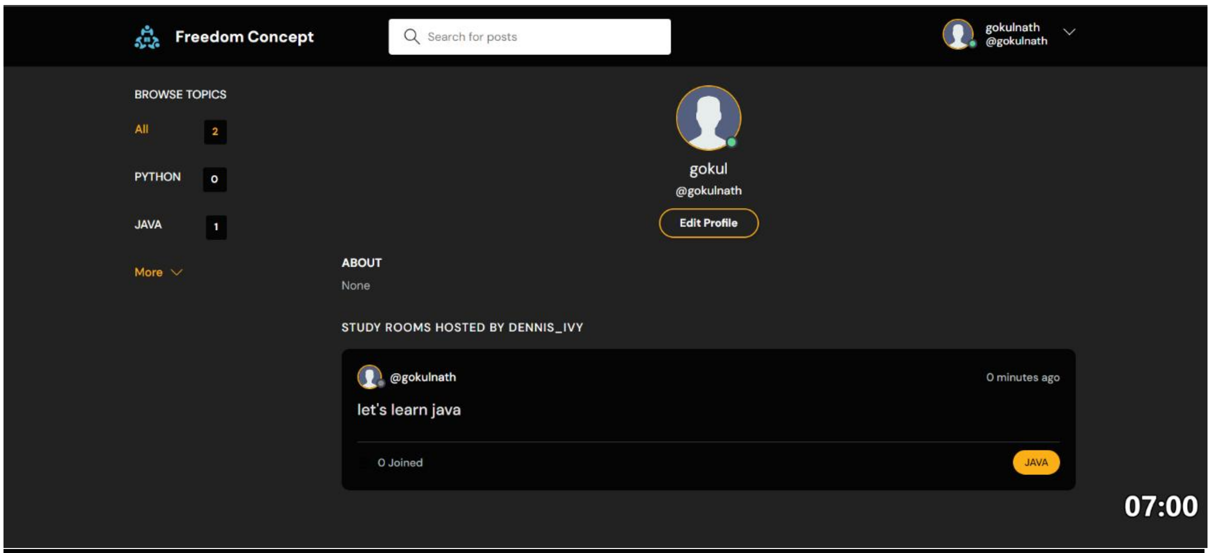
gokulnath
@gokulnath

BROWSE TOPICS

Search for topics

| | |
|---|---|
| All | 1 |
| PYTHON | 0 |

06:59


Freedom Concept

Search for posts

gokulnath
@gokulnath

BROWSE TOPICS

| | |
|---|---|
| All | 2 |
| PYTHON | 0 |
| JAVA | 1 |

More

gokul
@gokulnath

Edit Profile

ABOUT
None

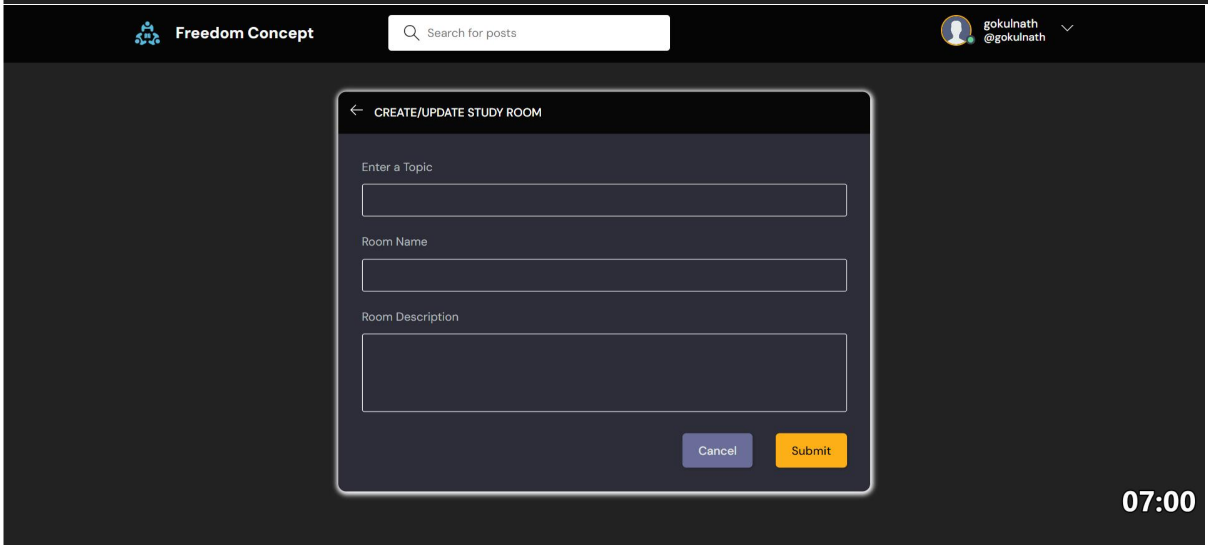
STUDY ROOMS HOSTED BY DENNIS_IVY

@gokulnath 0 minutes ago

let's learn java

0 Joined JAVA

07:00


Freedom Concept

Search for posts

gokulnath
@gokulnath

CREATE/UPDATE STUDY ROOM

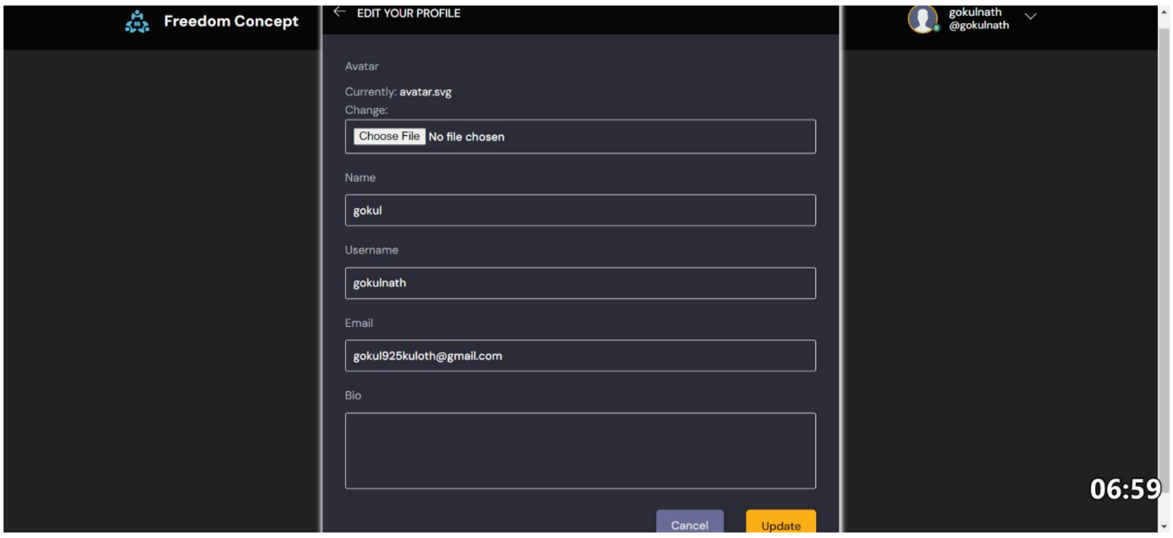
Enter a Topic

Room Name

Room Description

Cancel Submit

07:00



EDIT YOUR PROFILE

Avatar

Currently: avatar.svg

Change:

Choose File No file chosen

Name

gokul

Username

gokulnath

Email

gokul925kuloth@gmail.com

Bio

Cancel Update

## CONCLUSION:

The Study Room project demonstrates the effectiveness of Django in creating a versatile and feature-rich web application. Through thoughtful design and careful consideration of user needs, the project provides a valuable tool for individuals seeking an organized and efficient study environment. The development process adhered to best practices, and the documentation serves as a valuable resource for both users and future developers.

As the project continues to evolve, incorporating user feedback and introducing new features will contribute to its ongoing success. The Study Room project stands as a testament to the capabilities of web development with Django, offering a scalable foundation for educational platforms and study management applications.

## BIBLIOGRAPHY:

• Traversy media django-YouTube

- https://www.djangoproject.com
- Bro code-YouTube