CM3035 Advanced Web Development

Lesson L12

Created by Ben Gay

Introduction to Web Sockets

real-time applications e.g. Online Chat

Dual-channel full-duplex bidirectional communication simply means that the client and the server can talk in real-time without having to continuously make requests, and contrary to HTTP where request is always initiated by the client, and response is processed by the server, with web sockets the communication can go either way i.e. from server to client or client to server.

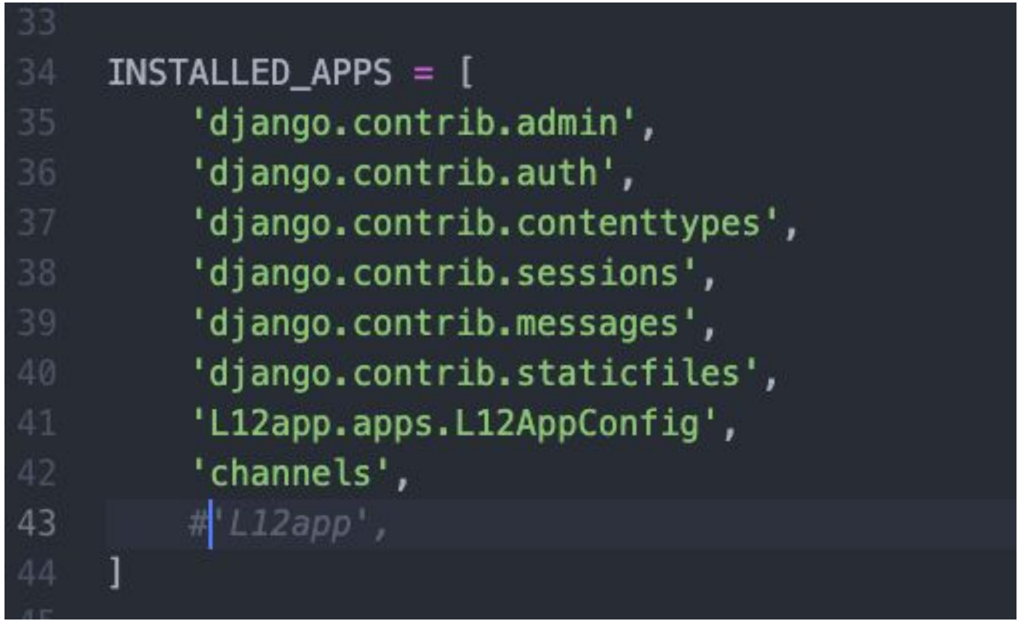
ASGI (Asynchronous Server Gateway Interface)

In order to handle a WebSocket connection, channels has **routing.py** and **consumers.py**.

Let’s begin with starting Virtual environment , **L12proj** and **L12app**

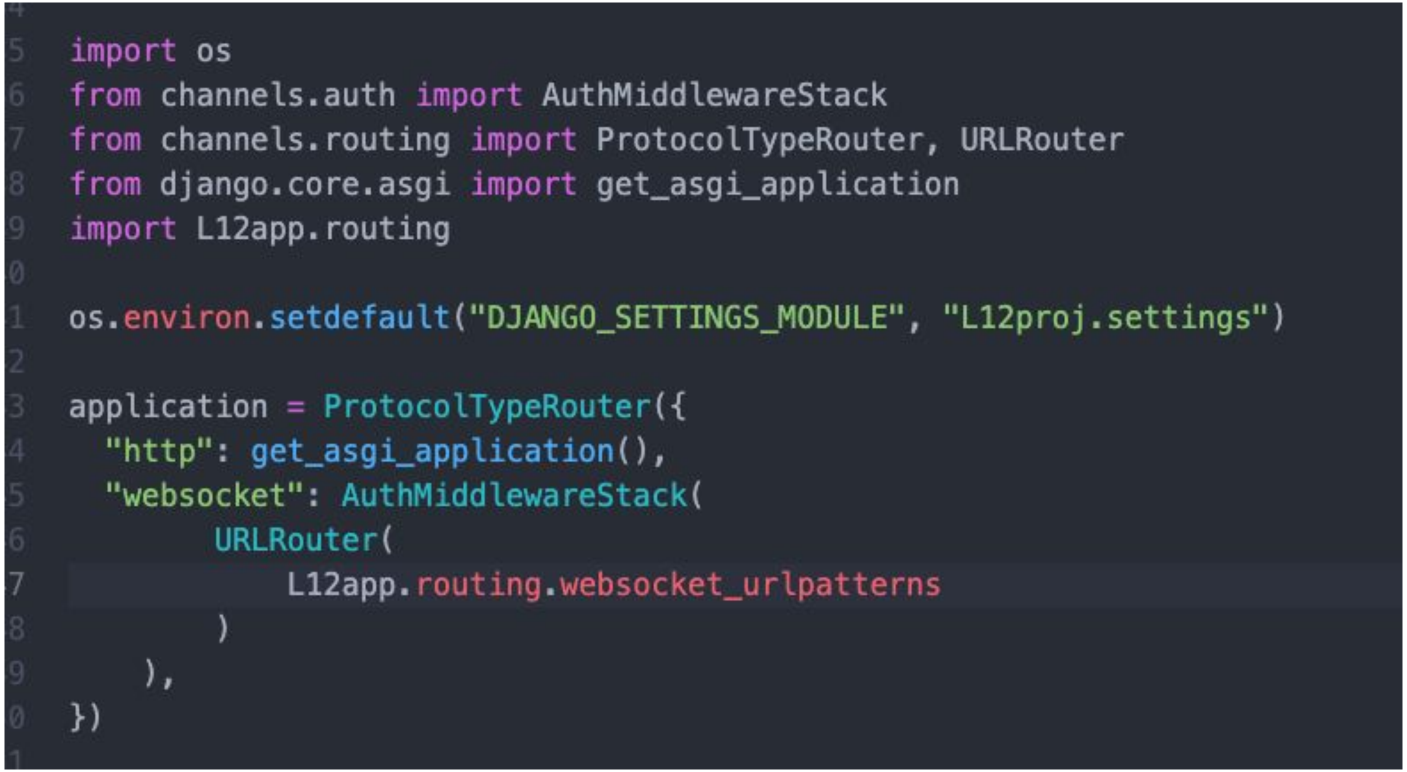
pip3 install channels

L12proj/settings.py



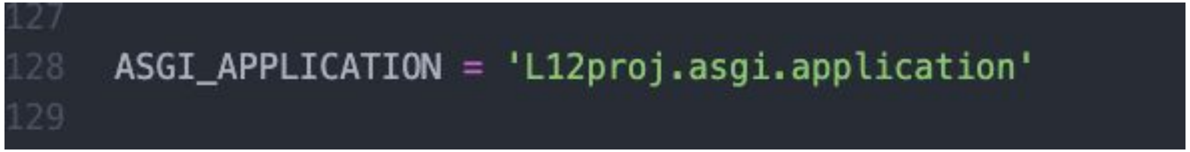
```python

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'L12app.apps.L12AppConfig',
    'channels',
    #'L12app',
]
```

L12proj/asgi.py



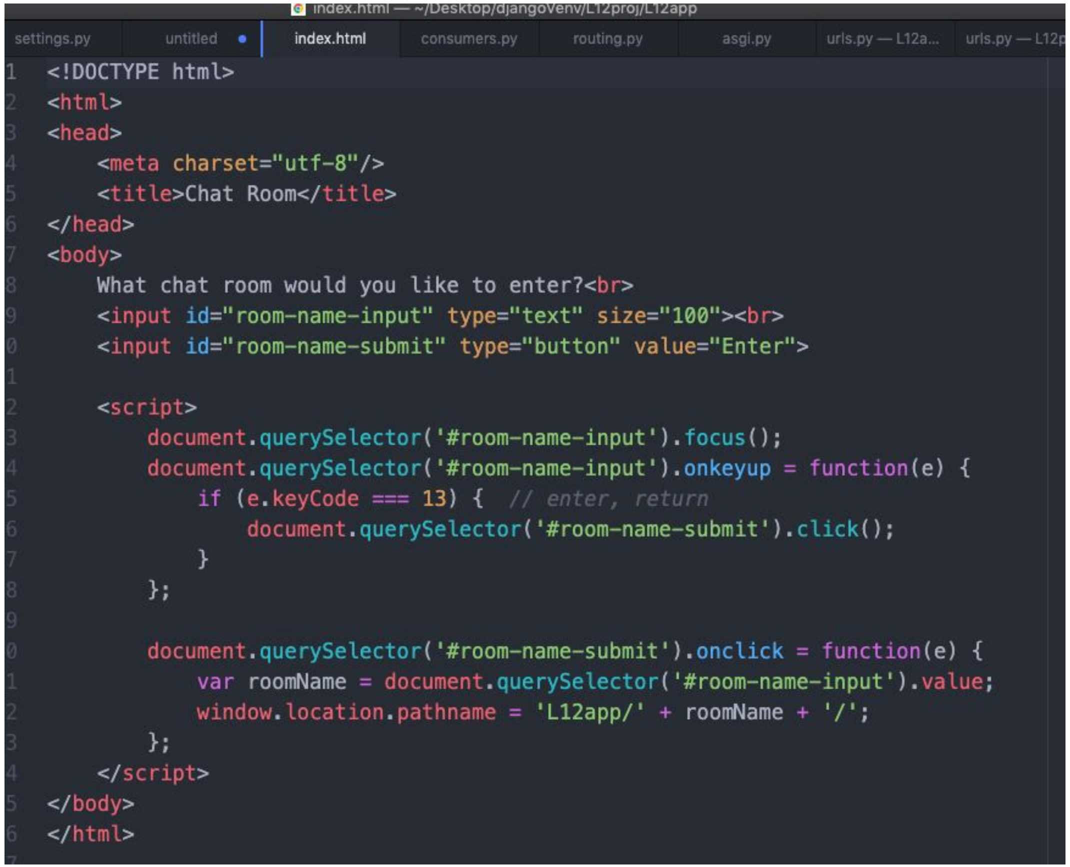
```python
import os
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
from django.core.asgi import get_asgi_application
import L12app.routing

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "L12proj.settings")

application = ProtocolTypeRouter({
  "http": get_asgi_application(),
  "websocket": AuthMiddlewareStack(
        URLRouter(
            L12app.routing.websocket_urlpatterns
        )
    ),
})
```

L12proj/settings.py



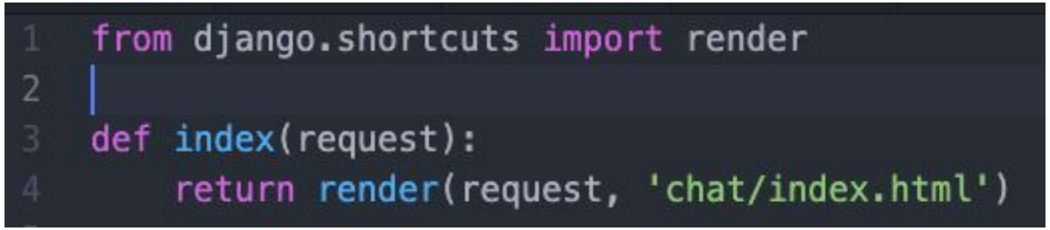
```python
ASGI_APPLICATION = 'L12proj.asgi.application'
```

templates/chat/index.html



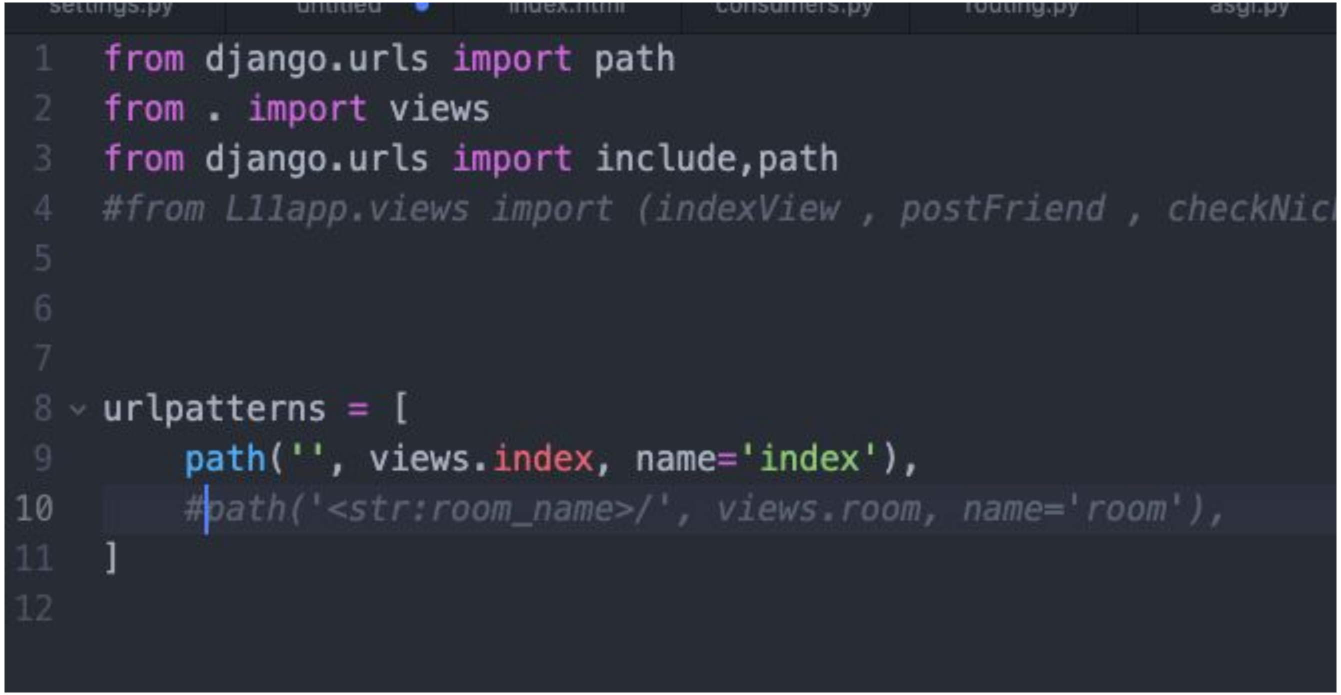
```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Chat Room</title>
</head>
<body>
    What chat room would you like to enter?<br>
    <input id="room-name-input" type="text" size="100"><br>
    <input id="room-name-submit" type="button" value="Enter">

    <script>
        document.querySelector('#room-name-input').focus();
        document.querySelector('#room-name-input').onkeyup = function(e) {
            if (e.keyCode === 13) {  // enter, return
                document.querySelector('#room-name-submit').click();
            }
        };

        document.querySelector('#room-name-submit').onclick = function(e) {
            var roomName = document.querySelector('#room-name-input').value;
            window.location.pathname = 'L12app/' + roomName + '/';
        };
    </script>
</body>
</html>
```

L12app/views.py



```python
from django.shortcuts import render

def index(request):
    return render(request, 'chat/index.html')
```

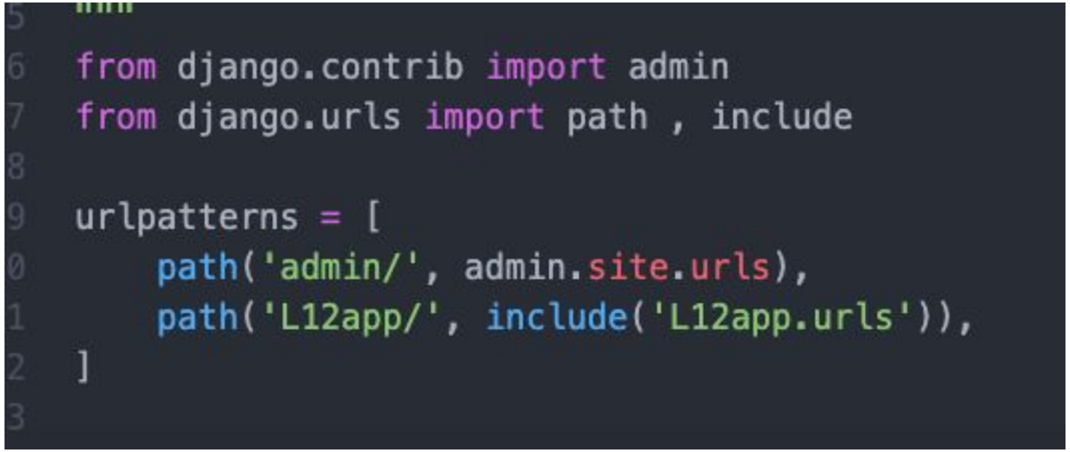
L12app/urls.py



settings.py untitled index.html consumers.py routing.py asgi.py

```python
from django.urls import path
from . import views
from django.urls import include,path
#from L11app.views import (indexView , postFriend , checkNic



urlpatterns = [
    path('', views.index, name='index'),
    #path('<str:room_name>/', views.room, name='room'),
]

```

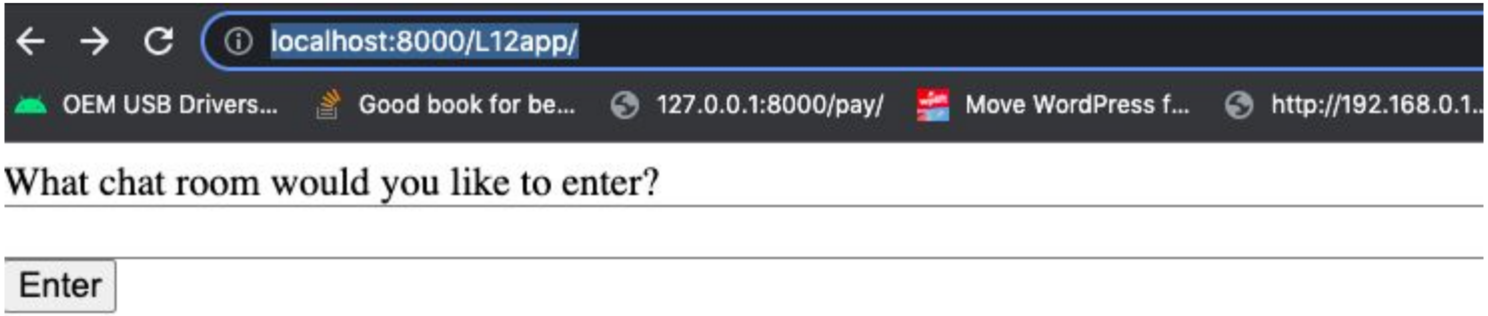
L12proj/urls.py



```
"""

from django.contrib import admin
from django.urls import path , include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('L12app/', include('L12app.urls')),
]

```

python manage.py runserver

http://localhost:8000/L12app/

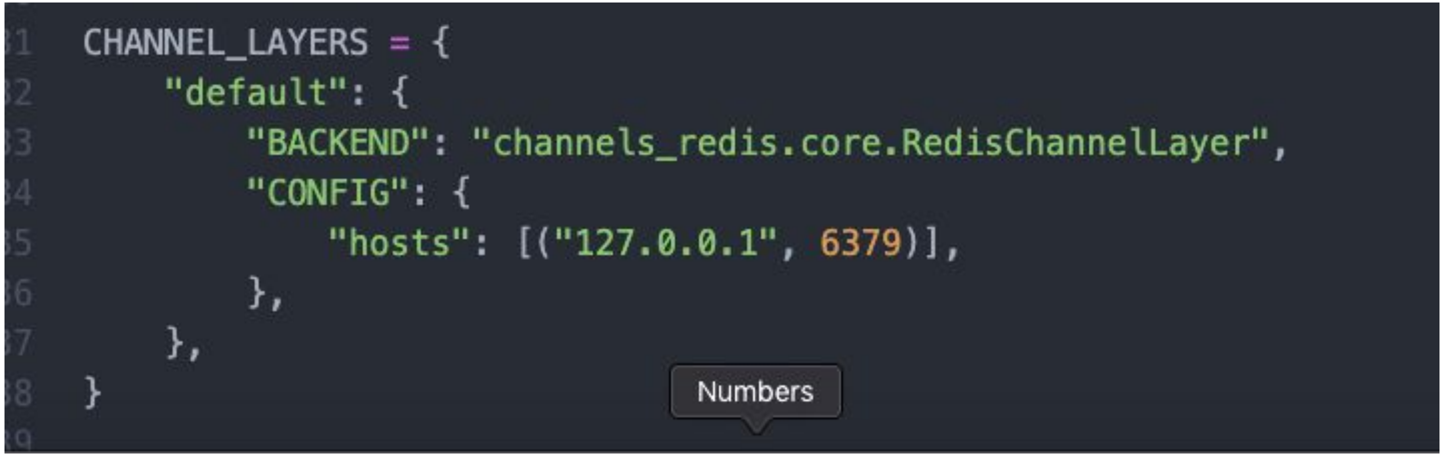


What chat room would you like to enter?

Channel Layers

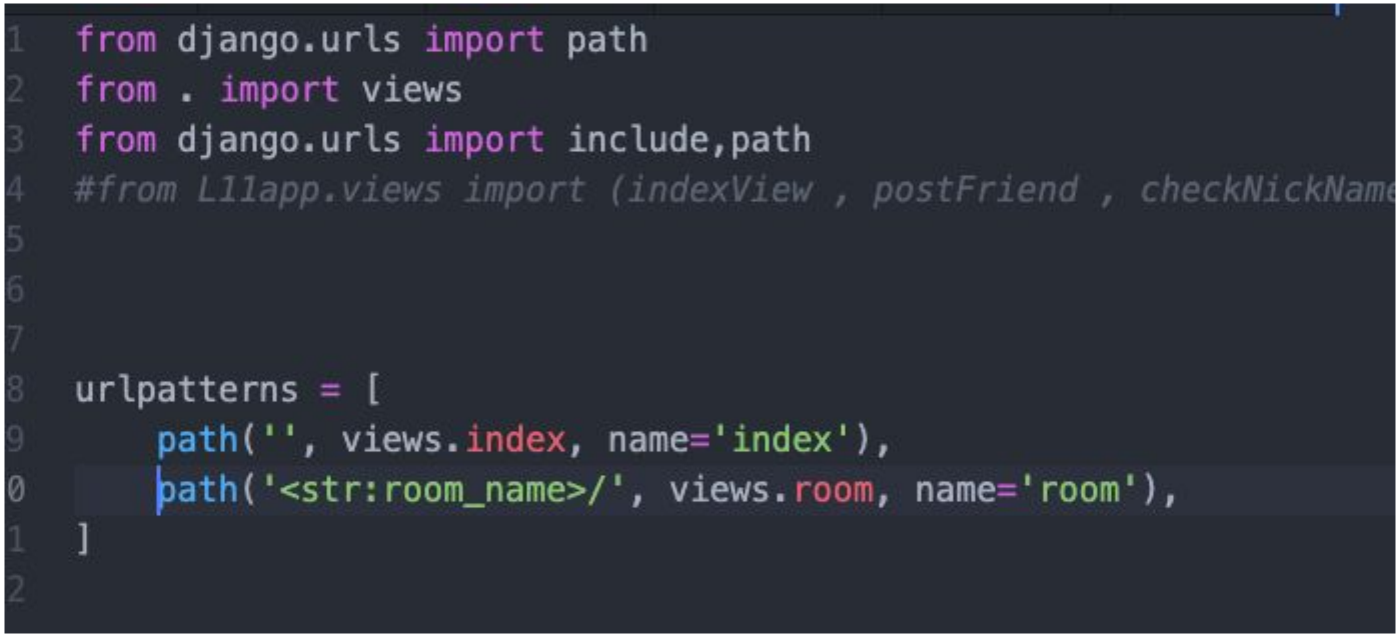
```
pip3 install channels_redis
```

L12proj/settings.py



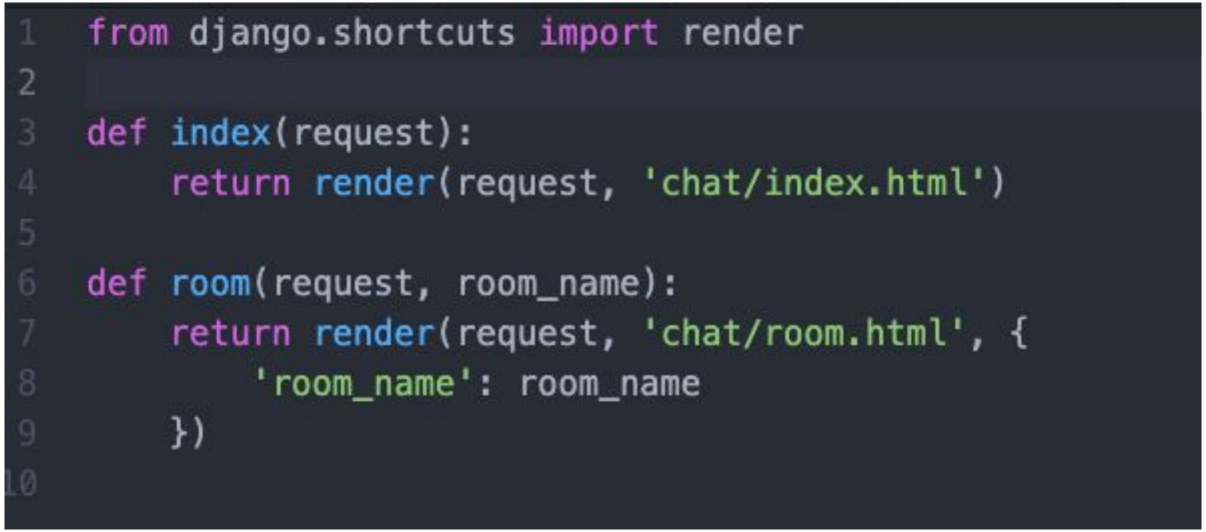
```
CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("127.0.0.1", 6379)],
        },
    },
}
```

L12app/urls.py



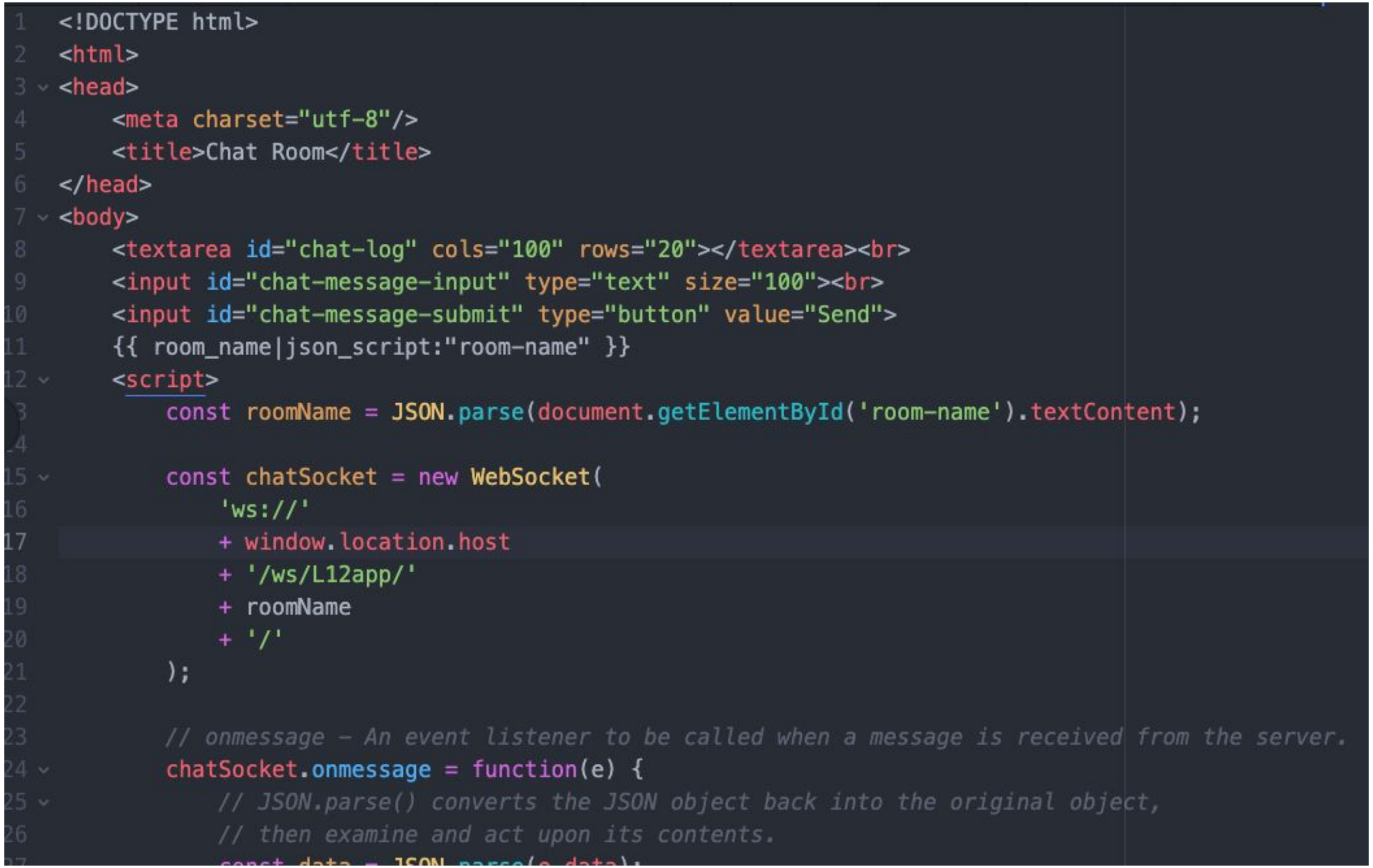
```python
from django.urls import path
from . import views
from django.urls import include,path
#from L11app.views import (indexView , postFriend , checkNickName



urlpatterns = [
    path('', views.index, name='index'),
    path('<str:room_name>/', views.room, name='room'),
]
```

L12app/views.py



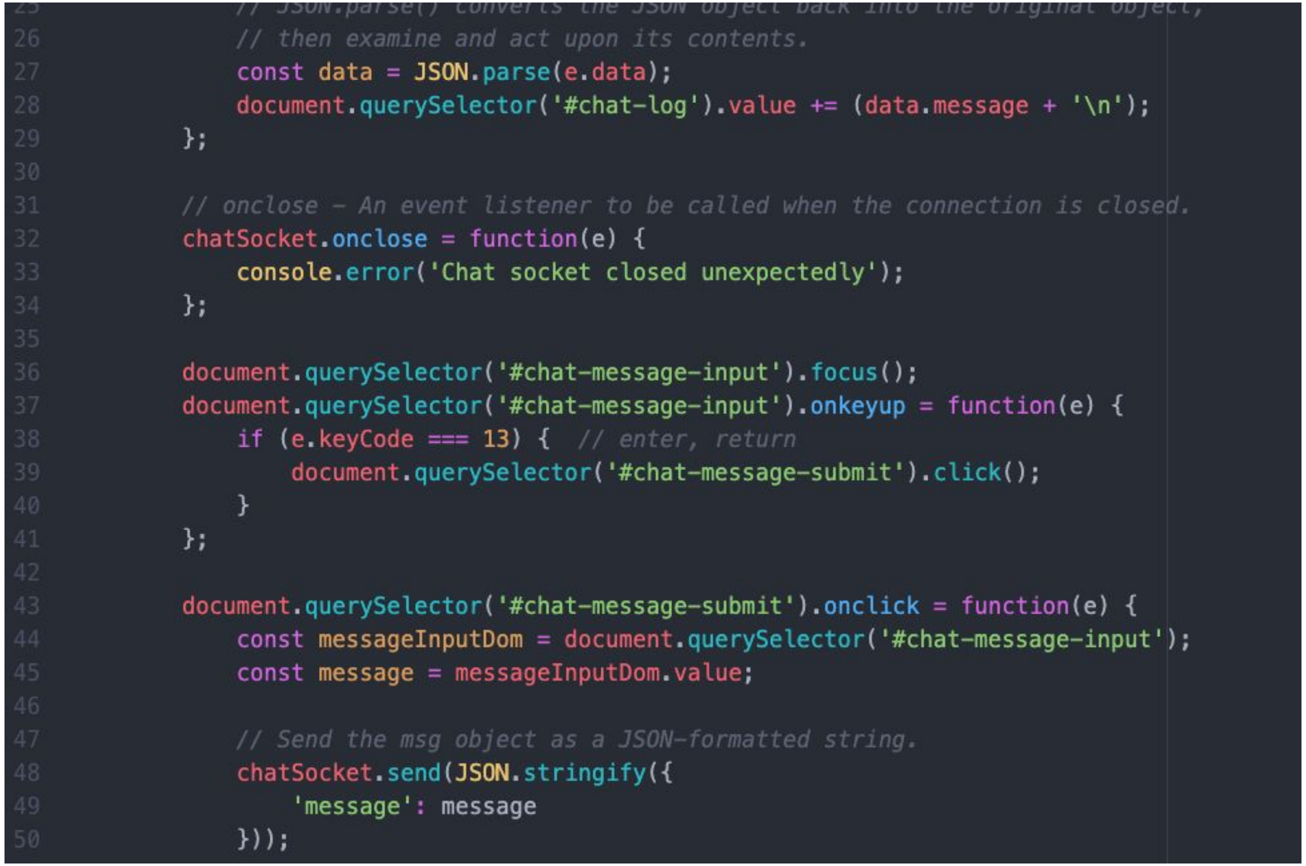
```python
from django.shortcuts import render

def index(request):
    return render(request, 'chat/index.html')

def room(request, room_name):
    return render(request, 'chat/room.html', {
        'room_name': room_name
    })
```

templates/chat/room.html



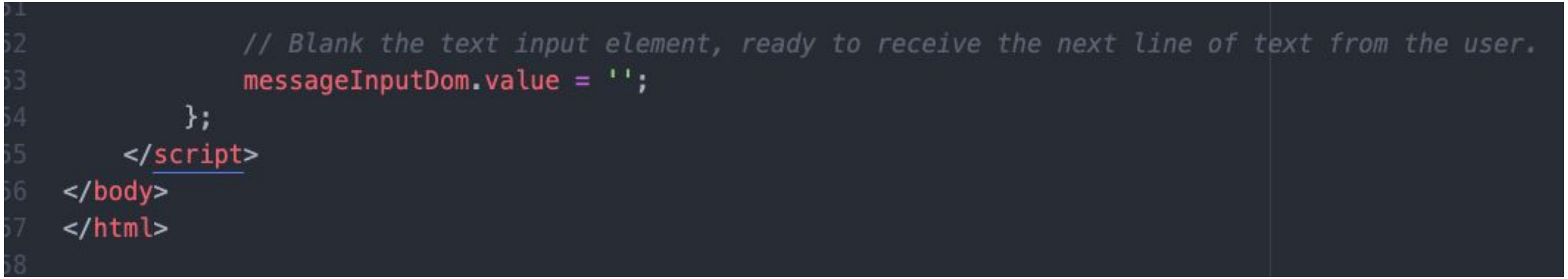
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Chat Room</title>
</head>
<body>
    <textarea id="chat-log" cols="100" rows="20"></textarea><br>
    <input id="chat-message-input" type="text" size="100"><br>
    <input id="chat-message-submit" type="button" value="Send">
    {{ room_name|json_script:"room-name" }}
    <script>
        const roomName = JSON.parse(document.getElementById('room-name').textContent);

        const chatSocket = new WebSocket(
            'ws://'
            + window.location.host
            + '/ws/L12app/'
            + roomName
            + '/'
        );

        // onmessage - An event listener to be called when a message is received from the server.
        chatSocket.onmessage = function(e) {
            // JSON.parse() converts the JSON object back into the original object,
            // then examine and act upon its contents.
```

templates/chat/room.html



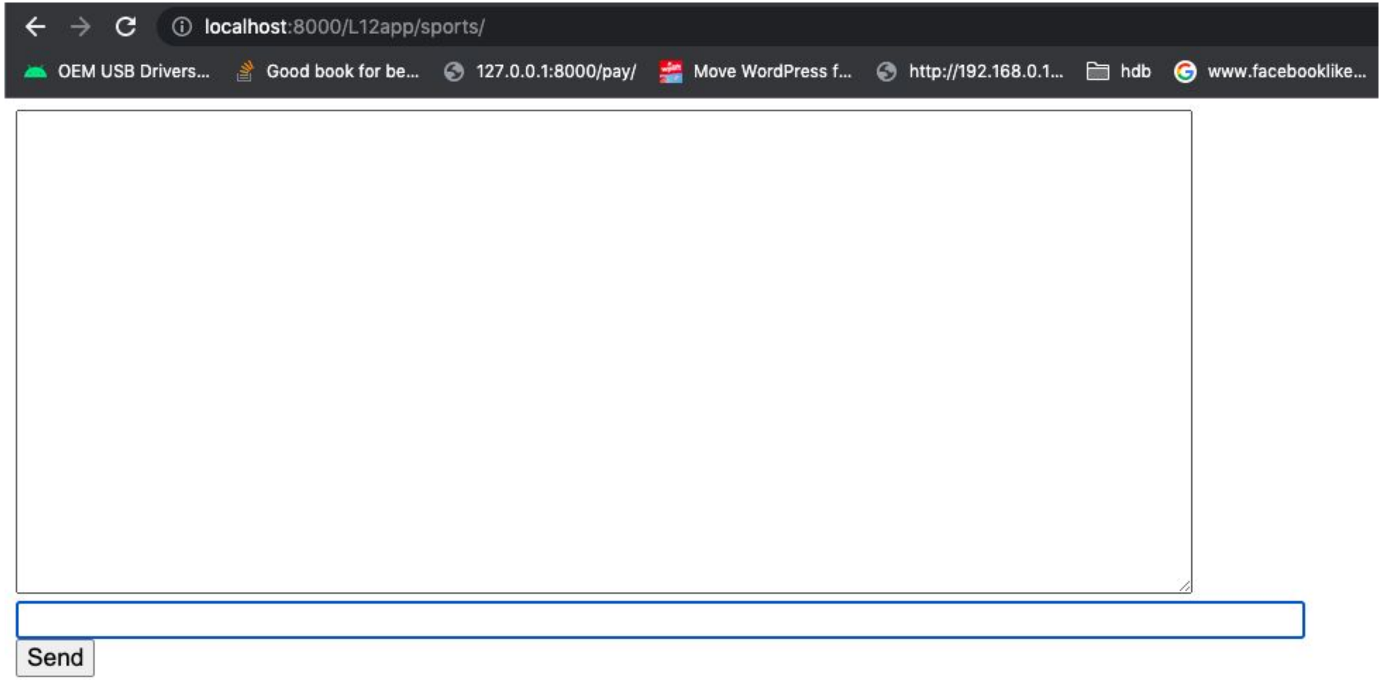
```
            // JSON.parse() converts the JSON object back into the original object,
            // then examine and act upon its contents.
            const data = JSON.parse(e.data);
            document.querySelector('#chat-log').value += (data.message + '\n');
        };

        // onclose - An event listener to be called when the connection is closed.
        chatSocket.onclose = function(e) {
            console.error('Chat socket closed unexpectedly');
        };

        document.querySelector('#chat-message-input').focus();
        document.querySelector('#chat-message-input').onkeyup = function(e) {
            if (e.keyCode === 13) {  // enter, return
                document.querySelector('#chat-message-submit').click();
            }
        };

        document.querySelector('#chat-message-submit').onclick = function(e) {
            const messageInputDom = document.querySelector('#chat-message-input');
            const message = messageInputDom.value;

            // Send the msg object as a JSON-formatted string.
            chatSocket.send(JSON.stringify({
                'message': message
            }));
```

templates/chat/room.html



```
                // Blank the text input element, ready to receive the next line of text from the user.
                messageInputDom.value = '';
            };
        </script>
    </body>
    </html>
```

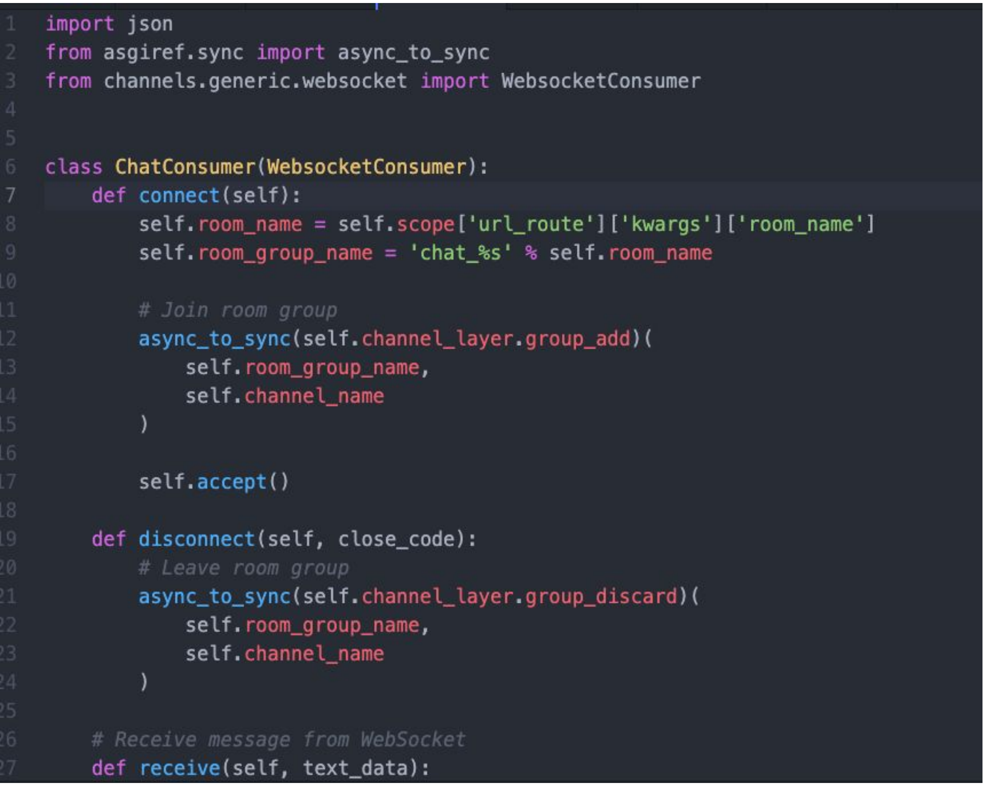
python3 manage.py runserver



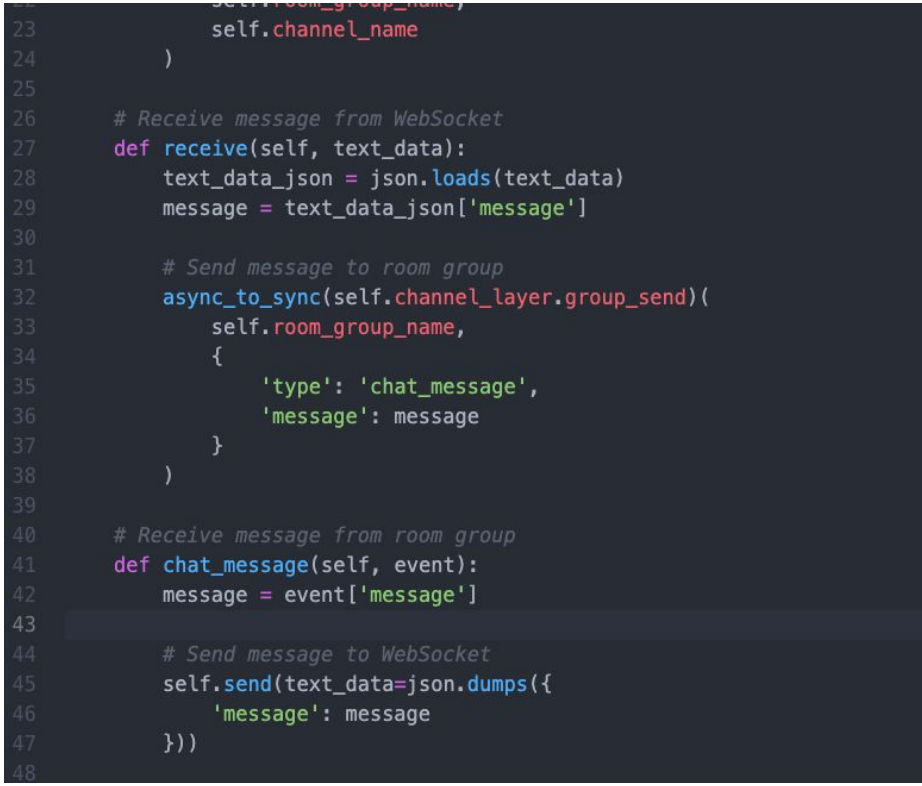
```
localhost:8000/L12app/sports/
```

SyncConsumer WebSocket

L12app/consumer.py

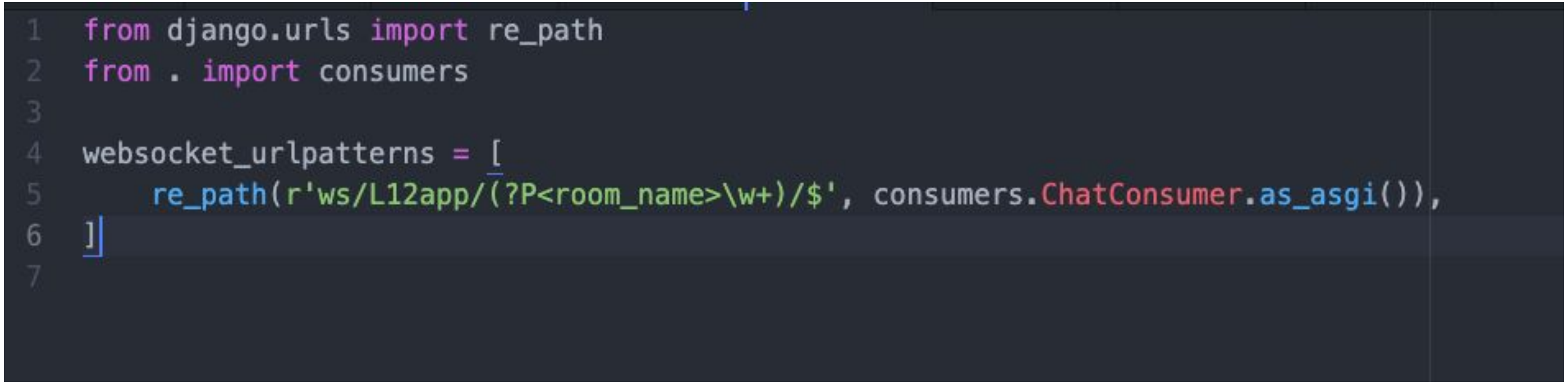


```python
import json
from asgiref.sync import async_to_sync
from channels.generic.websocket import WebsocketConsumer


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

        # Join room group
        async_to_sync(self.channel_layer.group_add)(
            self.room_group_name,
            self.channel_name
        )

        self.accept()

    def disconnect(self, close_code):
        # Leave room group
        async_to_sync(self.channel_layer.group_discard)(
            self.room_group_name,
            self.channel_name
        )

    # Receive message from WebSocket
    def receive(self, text_data):
```



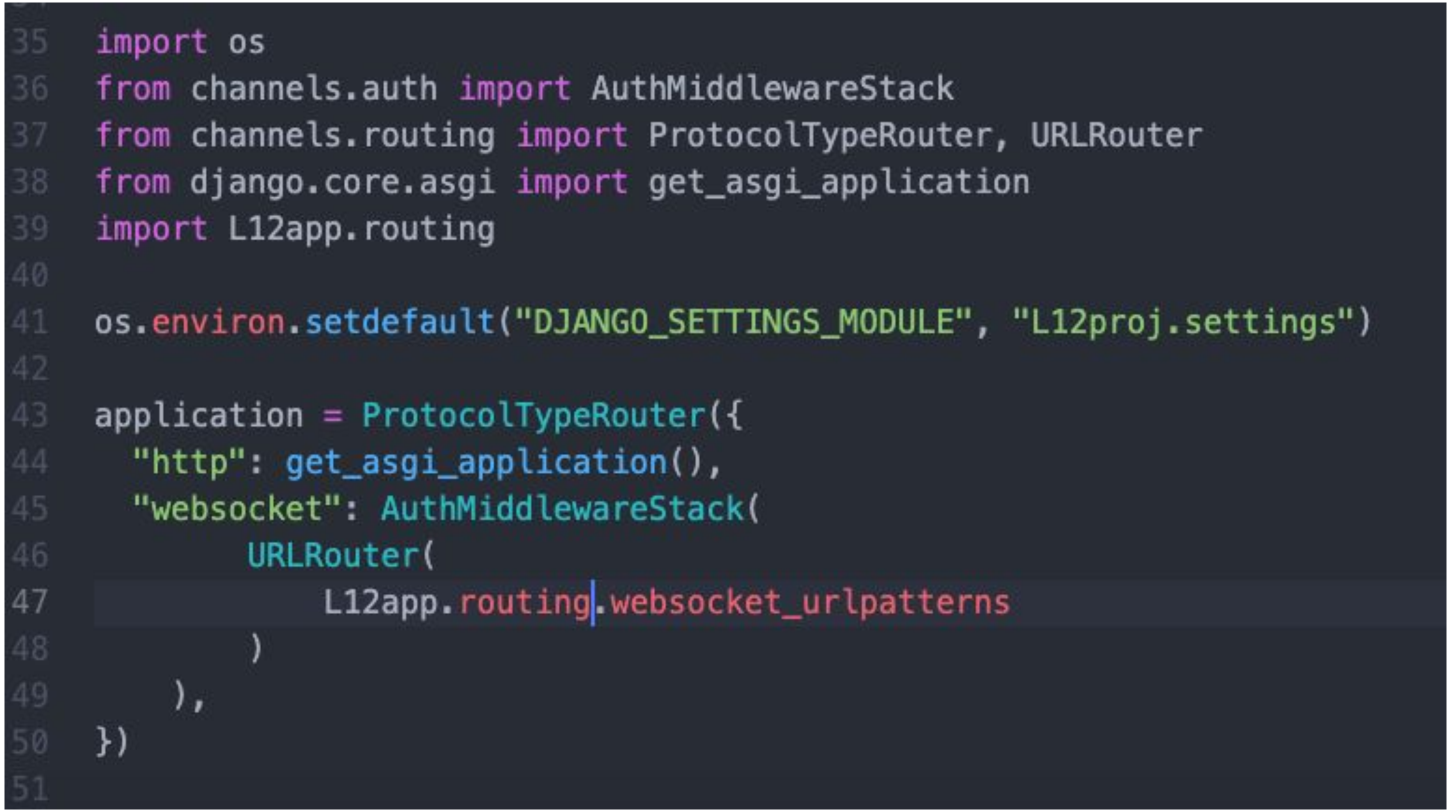
```python
            self.channel_name
        )

    # Receive message from WebSocket
    def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json['message']

        # Send message to room group
        async_to_sync(self.channel_layer.group_send)(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': message
            }
        )

    # Receive message from room group
    def chat_message(self, event):
        message = event['message']

        # Send message to WebSocket
        self.send(text_data=json.dumps({
            'message': message
        }))
```

L12app/routing.py



```python
from django.urls import re_path
from . import consumers

websocket_urlpatterns = [
    re_path(r'ws/L12app/(?P<room_name>\w+)/$', consumers.ChatConsumer.as_asgi()),
]
```

L12proj/asgi.py



```python
import os
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
from django.core.asgi import get_asgi_application
import L12app.routing

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "L12proj.settings")

application = ProtocolTypeRouter({
  "http": get_asgi_application(),
  "websocket": AuthMiddlewareStack(
        URLRouter(
            L12app.routing.websocket_urlpatterns
        )
    ),
})

```

Run Django server and Redis server

Install Redis server:

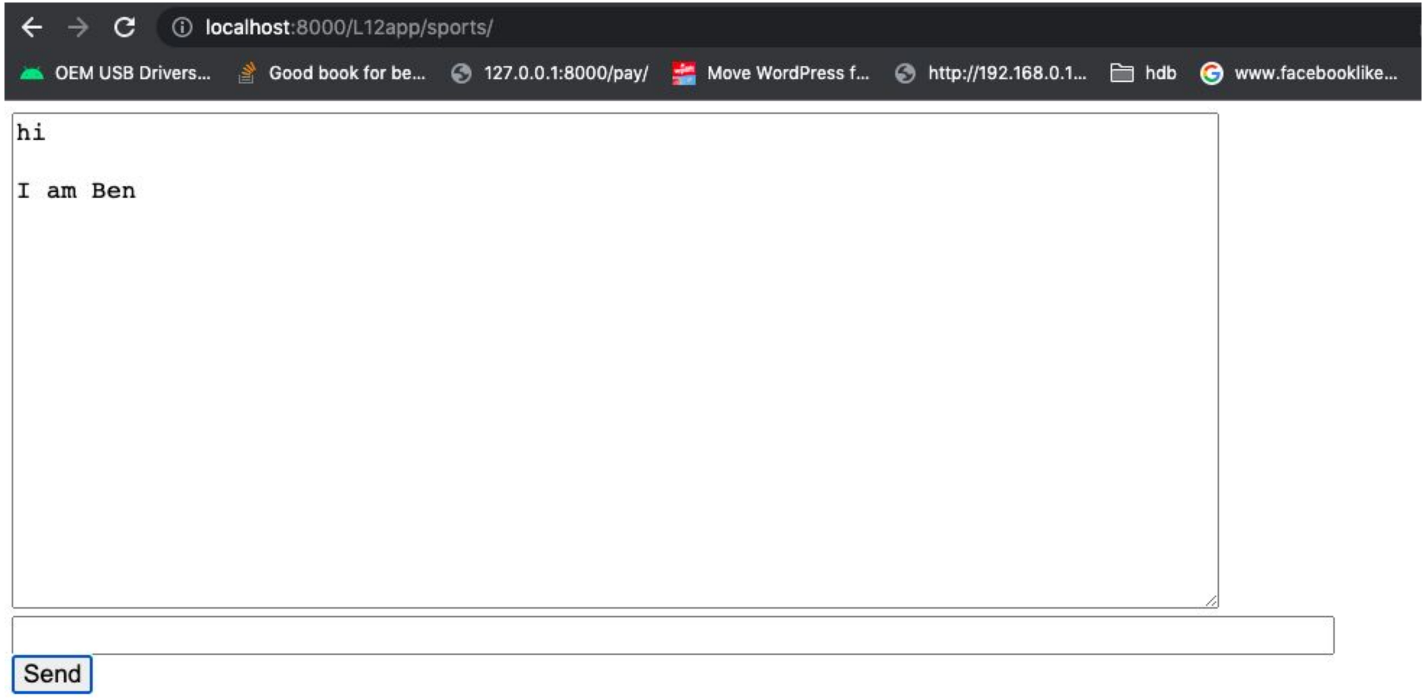
brew install redis

Run Redis server:

brew services start redis

Python3 manage.py runserver

Output



```
hi

I am Ben
```

End of Lesson 12