计算机通信网络（A 类）项目技术报告

——多人在线聊天程序

姓名：商进秩

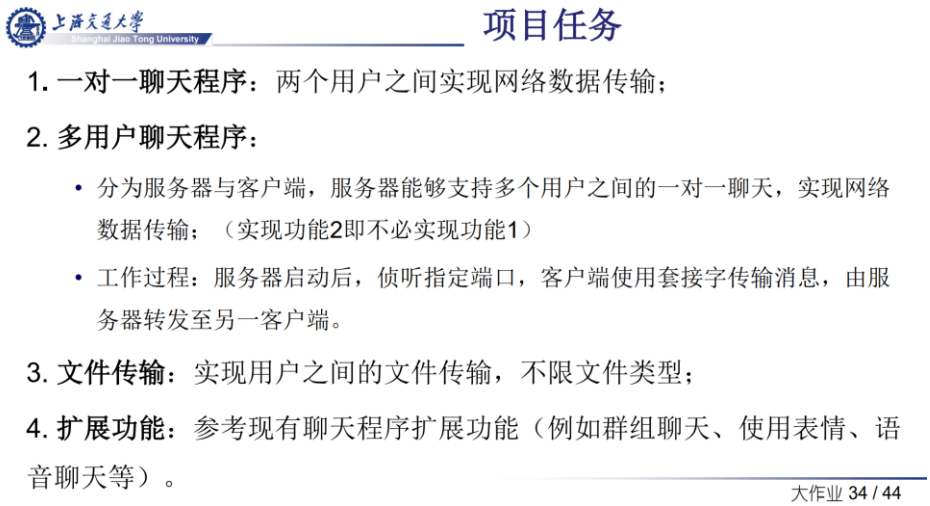
专业：信息安全

学号：5190309101 74

选题：项目六

1. 项目简介

本项目的选题为项目六：聊天程序，具体项目任务如下所示：



经过开发，本项目最终实现了项目任务 2. 的大体功能，实现了多用户之间的在线聊天室。

2. 项目分析

本项目采用 Django 框架，后端以 Python 为语言搭建 Web 界面作为聊天室的载体。通过借助 Django 的 channels 插件，在 Django 框架中实现基于 Websocket 协议的数据传输，实现项目要求的具体任务。

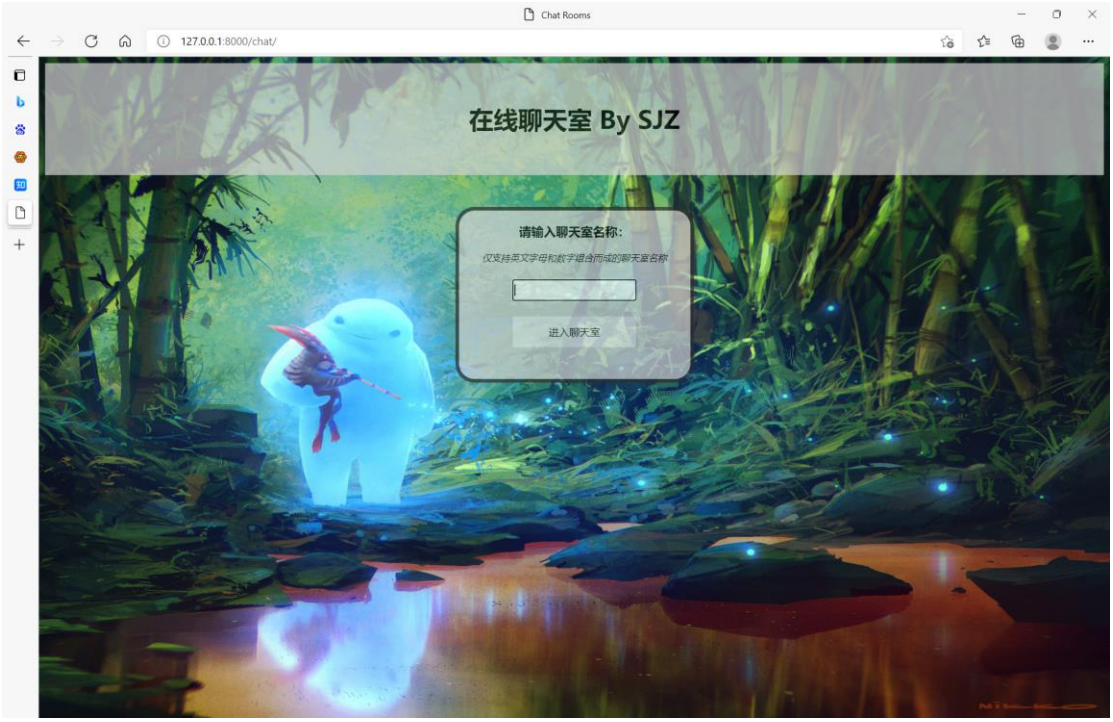
3. 实例展示

1）进入本地虚拟环境，启动 Django 服务器：

```
(env) PS D:\MyCode\Chatroom-IS301\env\chatroom> py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 29, 2021 - 20:11:19
Django version 3.2.10, using settings 'chatroom.settings'
Starting ASGI/Channels version 3.0.4 development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

2）访问 http://127.0.0.1:8000/chat，进入项目主页面：



3）在输入框内输入聊天室名称“test”，进入名为 test 的聊天室：

4）新开两个网页，分别进入”test”和“anothertest”两个聊天室。在聊天室中发送内容“Hello world!”，检查不同聊天室的收到情况：


Chat Room
127.0.0.1:8000/chat/test/
在线聊天室 By SJZ
[公告]欢迎来到test聊天室，请遵守国家法律法规，文明发言!
2021-12-29 20:20:10:Hello world!
Send


Chat Room
127.0.0.1:8000/chat/test/
在线聊天室 By SJZ
[公告]欢迎来到test聊天室，请遵守国家法律法规，文明发言!
2021-12-29 20:20:10:Hello world!
Send

可以看到，处在同一聊天室的两个客户端实现了在线聊天，处在不同聊天室的客户端之间无法进行在线聊天。

4. 项目目录：

```
D:.
    db.sqlite3
    manage.py
    requirements.txt

├─chat
    │  admin.py
    │  apps.py
    │  consumers.py
    │  models.py
    │  routing.py
    │  tests.py
    │  urls.py
    │  views.py
    │  __init__.py
    │
    ├─migrations
    │  │  __init__.py
    │  │
    │  └─__pycache__
    │          __init__.cpython-37.pyc
    │
    └─__pycache__
            admin.cpython-37.pyc
            apps.cpython-37.pyc
            consumers.cpython-37.pyc
            models.cpython-37.pyc
            routing.cpython-37.pyc
            urls.cpython-37.pyc
            views.cpython-37.pyc
            __init__.cpython-37.pyc
```

```
    └─__pycache__
            admin.cpython-37.pyc
            apps.cpython-37.pyc
            consumers.cpython-37.pyc
            models.cpython-37.pyc
            routing.cpython-37.pyc
            urls.cpython-37.pyc
            views.cpython-37.pyc
            __init__.cpython-37.pyc

├─chatroom
    │  asgi.py
    │  settings.py
    │  urls.py
    │  wsgi.py
    │  __init__.py
    │
    └─__pycache__
            asgi.cpython-37.pyc
            settings.cpython-37.pyc
            urls.cpython-37.pyc
            __init__.cpython-37.pyc

├─static
    │  background_index.png
    │  background_room.png
    │  index.css
    │  room.css
    │
└─templates
        index.html
        room.html
```

5. 核心代码：

consumers.py：负责处理通过 websocket 路由转发过来的请求和数据，类似于“views.py”的功能。

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


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

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

        self.accept()

    def disconnect(self, close_code):
        async_to_sync(self.channel_layer.group_discard)(
            self.room_group_name,
            self.channel_name
        )

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

        async_to_sync(self.channel_layer.group_send)(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': message
            }
        )

    def chat_message(self, event):
        message = event['message']
```

```
        datetime_str =
datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        self.send(text_data=json.dumps({
            'message': f'{datetime_str}:{message}'
        }))
```

room.html：前端关于即时通信的实现。

```
<!DOCTYPE html>
{%load static%}
<html>
<head>
    <meta charset="utf-8"/>
    <title>Chat Room</title>
    <link rel="stylesheet" type="text/css" href="{% static
'room.css' %}" charset="utf-8"/>
</head>
<body>
    <div class="header">
        <h1>在线聊天室 By SJZ</h1>
    </div>
    <br></br>
    <textarea id="chat-log" cols="100" rows="20"
readonly="readonly"></textarea><br>
    <input id="chat-message-input" type="text" size="100">
    <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 wss_protocol = (window.location.protocol == 'https:') ?
'wss://': 'ws://';
        const chatSocket = new WebSocket(
            wss_protocol + window.location.host + '/ws/chat/'  +
roomName + '/'
            );

        chatSocket.onopen = function(e) {
            document.querySelector('#chat-log').value += ('[公告]欢迎来
到' + roomName + '聊天室。请遵守国家法律法规，文明发言!\n')
```

```
        }

        chatSocket.onmessage = function(e) {
            const data = JSON.parse(e.data);
            document.querySelector('#chat-log').value += (data.message
+ '\n');
        };

        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) {
                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;

            chatSocket.send(JSON.stringify({
                'message': message
            }));
            messageInputDom.value = '';
        };
    </script>
</body>
</html>
```