《移动互联网》项目报告——“Tape”

一．成员与分工：

学号	姓名	分工
19307130117	杨之恒	前端+后端部分
19307130124	江刻优	后端部分+数据库

二．Git仓库地址：

```
https://github.com/JsCrying/Tape.git
```

三．前后端逻辑结构：

1. 前后端分离

$$Browser \overset{Call\ Server\ API}{\underset{Data}{\rightleftarrows}} Server$$

- 浏览器不关心`Server`如何拿到数据，只要知道约定的`API`可以获取到想要的数据即可，而`Server`也不关心浏览器如何使用数据，只要根据约定的`API`返回数据即可。这里需要了解`ajax`，这是两者交互较为常用的方式。
- `Server API`可以有两种形式，一种是把约定的`API`名字放到`url`上传给`server`，`server`解析`url`执行`API`并返回结果；另一种是把`API`作为参数的一部分传递给`server`，`server`读取参数时顺便读取到了`API`的名字。两种方式都不错，可以任意选用。

2. 前端

$$HTML + CSS + JS + AJAX$$

3. 后端

$$Flask\ Request\ Handler \leftrightarrow Server\ API\ handlers \leftrightarrow MySQL$$

四．具体运行：

在终端运行`python ./app.py`即可

五．功能效果与具体实现：

1.主页设计：

l 功能截图：

| 具体实现：

1.导航栏采用Flex布局的navbar标签
2.主体内容栏采用Grid布局进行纵向分割

2.登录与注册：

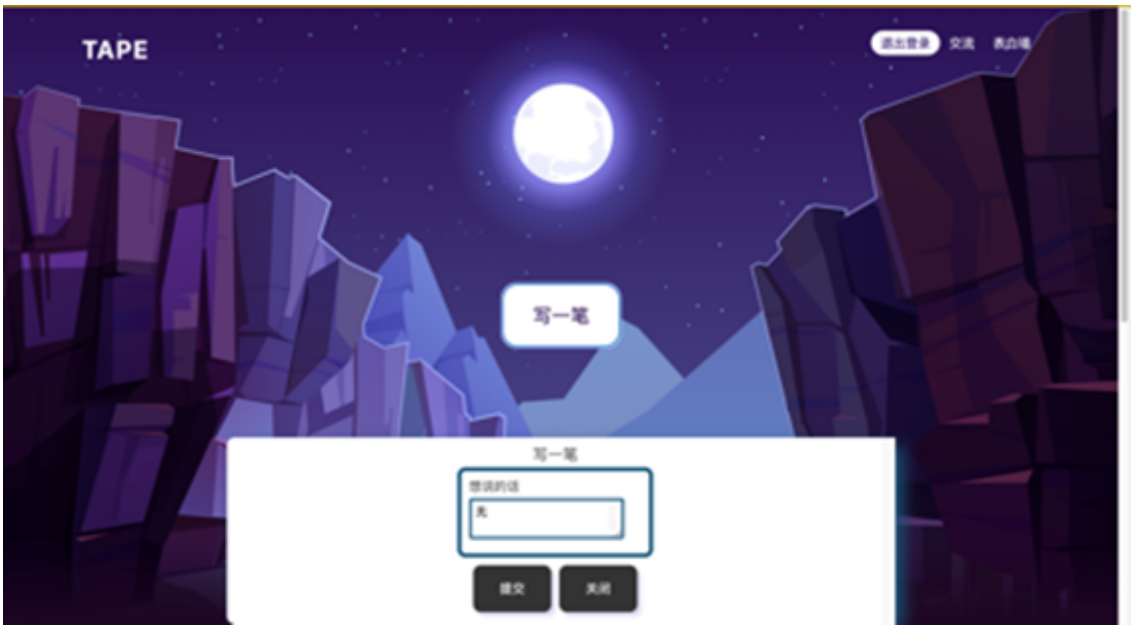
| 功能截图：

| 具体实现：

登录：
1.前端JS进行用户名与密码的非空输入检测
2.后端访问数据库进行用户登录合法性判断，若失败则重定向页面并给出错误提示；若成功则为该用户分配session，记录语言信息和登录状态，利用动态路由跳转至用户个人页面
注册：
1.前端JS进行输入合法性检测
2.后端同样进行合法性检测，如果没有错误则写入数据库，将页面重定向到登录页面；否则在前端给出错误提示

3.发布Tape：

| 功能截图：



| 具体实现：

1.添加功能：写入后，额外根据当前日期、用户名，由ajax向后端发送数据，后端检测合法后写入数据库

4.修改Tape：

| 功能截图：

| 具体实现：

1.修改功能：额外向后端传递当前信息在数据库的id、用户名，并在后端利用session进行合法性校验，后端检测合法后覆写数据库
2.删除功能：ajax向后端传送id、用户名，并在后端利用session进行合法性校验，后端检测合法后在数据库删除该数据
3.用户已写内容加载：前端向后端发送ajax请求，根据当前用户名在后端与session校验后，从数据库获取内容。利用事件委托的方法绑定事件，避免因动态加载的方式影响导致事件响应丢失

5.每日表白墙：

l 功能截图：

l 具体实现：

1.根据日期向后端发送ajax请求，获取当天数据库内容，并创建分页

6.实时交流：

l 功能截图：

l 具体实现：

1.前端创建WebSocket连接

```
    socket = io.connect('ws://' + document.domain + ':' + location.port +
'/window');
```

2.后端创建SocketIO服务器，分别响应连接、加入聊天室、离开聊天室、信息接收转发功能

```
    from flask import session
    from flask_socketio import emit, join_room, leave_room
    from .define_socket import chat_socketio
```

```
    print('import files')

    @chat_socketio.on('connect', namespace='/window')#响应连接
    def connect():
        print('Connect Success')

    @chat_socketio.on('joined', namespace='/window')#加入聊天室
    def joined(information):
        room_name = information.get('client_to_server')
        user_name = session.get('user_name')
        print(user_name + ' joined room!')
        join_room(room_name)
        emit(
            'status',
            {
                'user_name': user_name,
                'server_to_client': user_name + ' enters the room',
            },
            room = room_name
        )

    @chat_socketio.on('left', namespace='/window')#离开聊天室
    def left(information):
        room_name = information.get('client_to_server')
        user_name = session.get('user_name')
        if user_name:
            del session[user_name]
        leave_room(room_name)
        emit(
            'status',
            {
                'user_name': user_name,
                'server_to_client': user_name + ' has left the room',
            },
            room = room_name
        )

    @chat_socketio.on('msg', namespace='/window')#信息接收转发
    def msg(information):
        room_name = information.get('client_to_server')
        msg = information.get('msg').encode('raw_unicode_escape').decode()
        img = information.get('img') # 接收到的是 bytes
        timestamp =
information.get('timestamp').encode('raw_unicode_escape').decode()
        user_name = session.get('user_name')
        emit(
            'message',
            {
                'user_name': user_name,
                'msg': msg,
                'img': img,
                'timestamp': timestamp
            },
            room = room_name
```

```
        )
```

六. 创新与疑难点:

1.CSS动画设计:

| 功能截图:

| 具体实现:

```
1.背景CSS设置backdrop-filter设置毛玻璃屏样式
2.逐个字正则切割，并用span标签包裹，hover时添加CSS动画
3.CSS动画：移动+旋转+filter: blur()
```

2.中英文转换:

| 功能截图:

| 具体实现:

```
1.为要翻译的标签内容添加标记类
2.为具体词汇设置自己的字典，点击翻译按钮JS自动对页面内容进行替换
```

3.响应式布局：

l 功能截图：

l 具体实现：

```
1.根据屏幕显示宽度设置media(max-width: )
2.Flex布局根据宽度改变显示后，将flex-dorection改为垂直
```