




기린 A+ 북 마켓

: 대학생들을 위한
중고 서적 플랫폼


SW개발환경 및 응용 11조 김재혁 김서영 정혜정

Part 0-1. Github 주소


https://github.com/jaehyuk328/Giraffe_BookMarket.git

 Giraffe_BookMarket Public Watch 1

main 1 Branch 3 Tags Add file <> Code

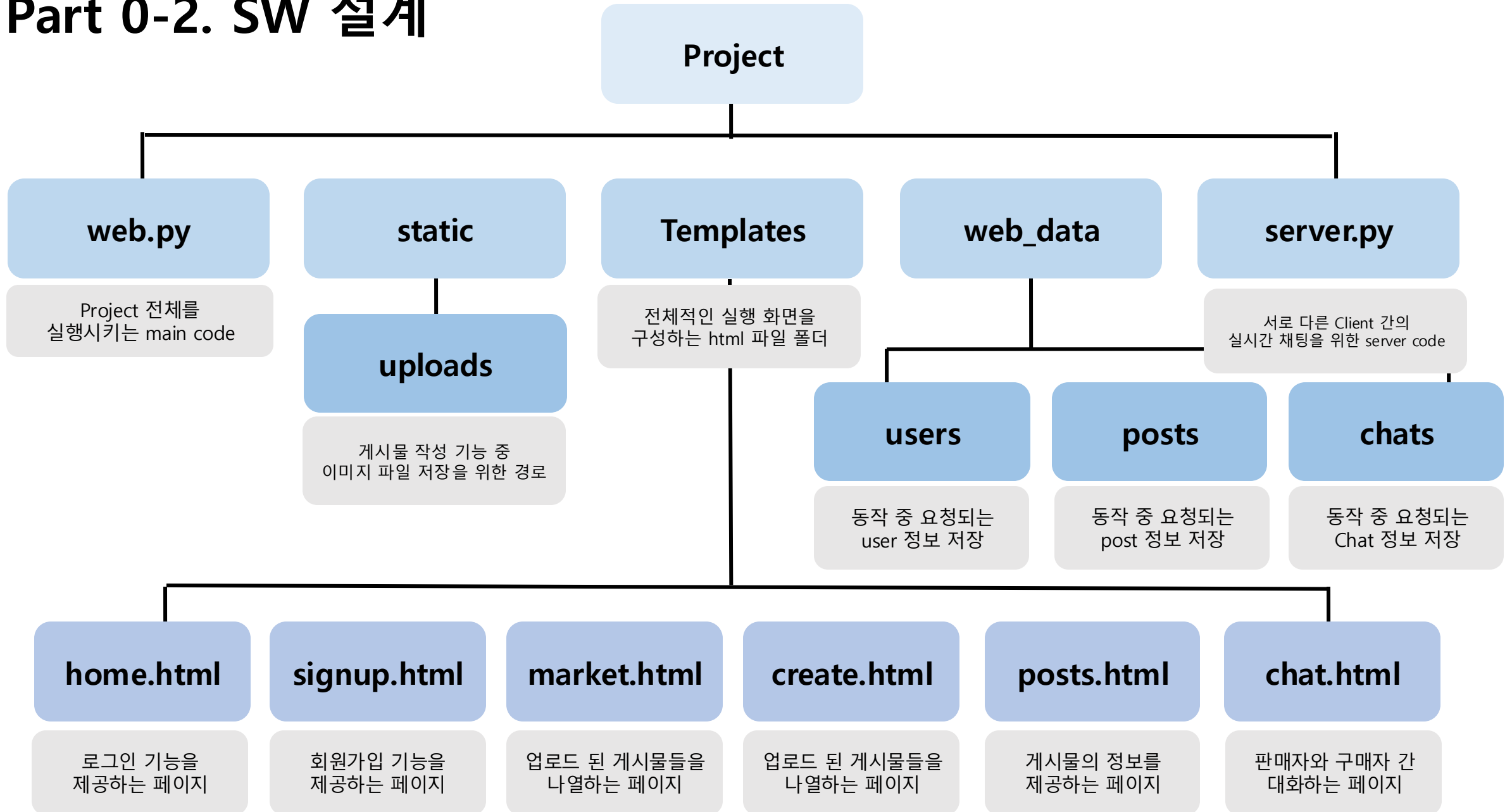
 seoyoung0627

 이미지 업로드 기능 구현 및 UI 수정 ba45126 · 11 hours ago 3 Commits

 Project

 이미지 업로드 기능 구현 및 UI 수정 11 hours ago

Part 0-2. SW 설계



Part 1-1. 로그인 시스템 (필수기능 1)



로그인 시스템

회원가입과 로그인 기능 제공
(ID/PW/전공/email)



User 정보 저장

로그인 시스템에서
받은 정보를 저장



네트워크

데이터 전송과 통신을 위한
네트워크 기능 구현



서버 관리

안정적인 운영을 위한
모니터링과 롤백 기능

```
@app.route('/signup', methods=['POST'])
def signup():
    username = request.form['username']
    password = request.form['password']
    confirm_password = request.form['confirm_password']
    major = request.form['major']

    if len(password) < 8:
        flash("Password must be at least 8 characters long.")
        return redirect(url_for('signup_page'))
    if not any(char.isdigit() for char in password):
        flash("Password must contain at least one number.")
        return redirect(url_for('signup_page'))
    if not any(char.isalpha() for char in password):
        flash("Password must contain at least one letter.")
        return redirect(url_for('signup_page'))
    if password != confirm_password:
        flash("Passwords do not match!")
        return redirect(url_for('signup_page'))

    users = load_user_data()
    if username in users:
        flash("Username already exists!")
        return redirect(url_for('signup_page'))

    users[username] = {"user_id": username, "password": password, "major": major}
    save_user_data(users)
    flash("Registration successful! You can now log in.")
    return redirect(url_for('home'))
```

Web.py

- 회원가입을 위한 정보 username, password, major를 받음
- 취약하지 않는 비밀번호를 설정하기 위하여 password를 위한 4가지 조건 설정
 - Password 길이 8자리 이상
 - 적어도 한 자리는 숫자 포함
 - 적어도 한 자리는 문자 포함
 - 비밀번호 재확인
- 중복 username이 생성되는 것을 방지하기 위하여, 기존에 존재하는 username인지 검사

Part 1-2. User 정보 저장 (필수기능 2)



로그인 시스템
회원가입과 로그인 기능 제공
(ID/PW/전공/email)



User 정보 저장
로그인 시스템에서
받은 정보를 저장



네트워크
데이터 전송과 통신을 위한
네트워크 기능 구현



서버 관리
안정적인 운영을 위한
모니터링과 롤백 기능

```
@app.route('/signup', methods=['POST'])
def signup():
    username = request.form['username']
    password = request.form['password']
    confirm_password = request.form['confirm_password']
    major = request.form['major']

    if len(password) < 8:
        flash("Password must be at least 8 characters long.")
        return redirect(url_for('signup_page'))
    if not any(char.isdigit() for char in password):
        flash("Password must contain at least one number.")
        return redirect(url_for('signup_page'))
    if not any(char.isalpha() for char in password):
        flash("Password must contain at least one letter.")
        return redirect(url_for('signup_page'))
    if password != confirm_password:
        flash("Passwords do not match!")
        return redirect(url_for('signup_page'))
```

```
def save_user_data(data):
    with open(os.path.join(user_data_dir, 'users.json'), 'w') as f:
        json.dump(data, f, indent=4)
```

```
users[username] = {"user_id": username, "password": password, "major": major}
save_user_data(users)
flash("Registration successful! You can now log in.")
return redirect(url_for('home'))
```

```
~/Project/
├── Templates/
│   ├── home.html
│   ├── signup.html
│   ├── chat.html
│   └── stor
├── venv/
├── web_data/
│   ├── users/
│   │   └── users.json
│   ├── posts/
│   │   └── posts.json
└── web.py
```

User 정보가 저장되는 경로

User 정보가 저장되는 모습

```
web_data > u
1  {
2    "seoyoung0627": {
3      "user_id": "seoyoung0627",
4      "password": "qwer1234",
5      "major": "AI \uc735\ud569\uc804\uacf5"
6    },
7    "kim": {
8      "user_id": "kim",
9      "password": "qwer1234",
10     "major": "AI \uc735\ud569\uc804\uacf5"
11   }
12 }
```

Part 1-2. User 정보 저장 (필수기능 2)

3-1) 필수 기능



로그인 시스템

회원가입과 로그인 기능 제공
(ID/PW/전공/email)



User 정보 저장

로그인 시스템에서
받은 정보를 저장



네트워크

데이터 전송과 통신을 위한
네트워크 기능 구현



서버 관리

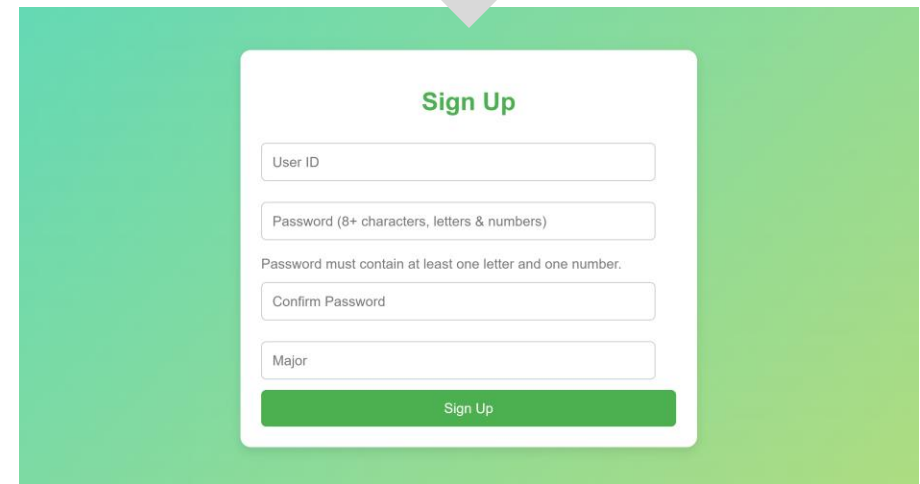
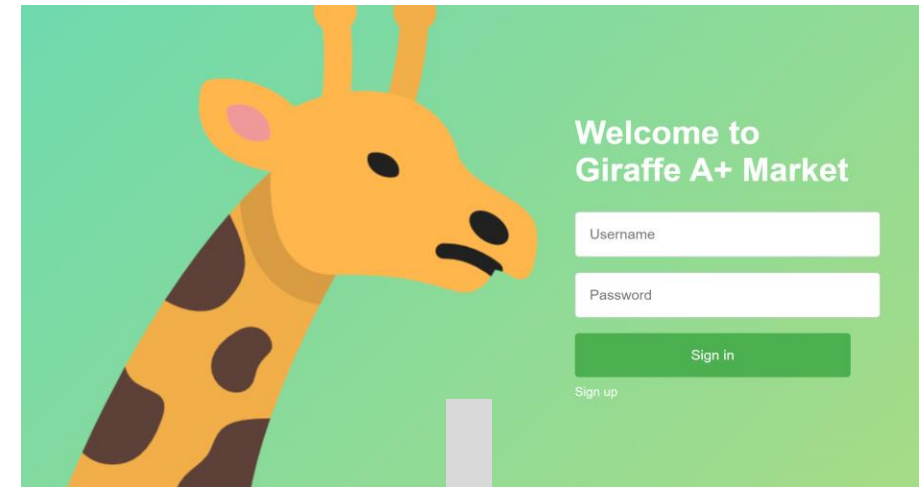
안정적인 운영을 위한
모니터링과 롤백 기능

home.html

signup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: linear-gradient(135deg, #64D9B5, #AEDC81);
      height: 100vh;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
    }
    .login-section {
      text-align: left;
      background: white;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 24px;
      color: #4CAF50;
    }
    input {
      width: 100%;
      padding: 10px;
      margin: 10px 0;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    button {
      width: 100%;
      padding: 10px;
      background: #4CAF50;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    button:hover {
      background: #45a049;
    }
    a {
      display: block;
      text-align: center;
      margin-top: 10px;
      color: #4CAF50;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <div class="login-section">
    <h1>Login</h1>
    <form action="/login" method="POST">
      <input type="text" name="username" placeholder="User ID" required>
      <input type="password" name="password" placeholder="Password" required>
      <button type="submit">Login</button>
    </form>
    <a href="/signup">Sign up</a>
  </div>
</body>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sign Up Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: linear-gradient(135deg, #64D9B5, #AEDC81);
      height: 100vh;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
    }
    .signup-section {
      width: 100%;
      max-width: 400px; /* 컨테이너 너비 제한 */
      text-align: left;
      background: white;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 24px;
      color: #4CAF50;
      text-align: center; /* 제목을 중앙 정렬 */
    }
    input {
      width: calc(100% - 20px); /* 패딩에 맞춰 필드 너비 조정 */
      padding: 10px;
      margin: 10px 0;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-sizing: border-box; /* 패딩과 너비 포함 계산 */
    }
    button {
      width: 100%;
      padding: 10px;
      background: #4CAF50;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    button:hover {
      background: #45a049;
    }
    small {
      display: block;
      margin-top: 5px;
      color: #777;
    }
  </style>
</head>
```



Part 2-1. 서버 관리 (필수기능 4)

서버 관리

서비스 제공

사용자 요구를 충족하기 위해 안정적이고 효율적인 서버 서비스를 제공합니다.



보안 관리

서버를 사이버 위협으로부터 보호하고 데이터 무결성을 유지합니다.



설치 및 구성

서버의 초기 설정 및 최적화를 포함하여 원활한 운영을 보장합니다.

모니터링 및 유지보수

서버 성능을 지속적으로 모니터링하고 문제를 해결하여 가동 중지 시간을 최소화합니다.



로그인 시스템

회원가입과 로그인 기능 제공
(ID/PW/전공/email)



User 정보 저장

로그인 시스템에서
받은 정보를 저장



네트워크

데이터 전송과 통신을 위한
네트워크 기능 구현



서버 관리

안정적인 운영을 위한
모니터링과 롤백 기능

```
~/Project/
├── Templates/
│   ├── home.html
│   ├── signup.html
│   ├── chat.html
│   └── store.html
├── venv/
├── web_data/
│   ├── users/
│   │   └── users.json
│   ├── posts/
│   │   └── posts.json
└── web.py
```

bash

코드 복사

```
sudo apt install mdadm # 패키지 설치
```

```
sudo mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sda /dev/sdb /dev/sdc
```

Part 2-2. 게시물 작성 및 업로드 (선택기능1)



```

# 입력 데이터 받기
user_id = request.form.get('user_id')
title = request.form.get('title')
content1 = request.form.get('content1') # 책 제목
content2 = request.form.get('content2') # 가격
content3 = request.form.get('content3') # 책 상태
content4 = request.form.get('content4') # 장소
  
```

```

# 이미지 처리
image = request.files.get('image') # 파일 가져오기
image_filename = "default.png" # 기본 이미지 파일명 설정
if image and allowed_file(image.filename):
    # 이미지 파일명 안전하게 처리
    image_filename = secure_filename(image.filename)
    image.save(os.path.join(UPLOAD_FOLDER, image_filename))
  
```

```

placeholder_file = os.path.join(UPLOAD_FOLDER, ".placeholder")
if not os.path.exists(placeholder_file):
    with open(placeholder_file, "w") as f:
        f.write("")
  
```

Web.py

- 게시물작성창에서 입력값을 받는 기능을 수행하기 위해 user_id, title, content1,2,3,4, image 7개의 값을 설정함
- 이미지업로드를 하지않았을 경우 default값으로 대체
- UPLOAD_FOLDER에 .placeholder라는 빈 파일을 생성해서 폴더가 비어있는 경우에도 폴더가 삭제되지 않도록 하는 기능
 - Git은 디렉토리가 비어있으면 디렉토리를 추적하지않음

Part 2-2. 게시물 작성 및 업로드 (선택기능1)

Target user(판매자) : Senario 1



1. 로그인



2. 게시물 작성



3. 판매자와 1:1 대화

판매자



게시글작성 클릭 시 이동

게시글작성 정보를 받아
저장 버튼을 누른 후 get_post로 이동

```
# 새 게시물 추가
new_post = {
    "user_id": user_id,
    "title": title,
    "content1": content1,
    "content2": content2,
    "content3": content3,
    "content4": content4,
    "image": image_filename, # 이미지 파일명 추가
    "id": len(posts) + 1 # 게시물 ID
}
posts.insert(0, new_post) # 최신 글을 맨 앞에 추가
```

최신 글 맨 위에 추가

```
@app.route('/get_posts', methods=['GET'])
def get_posts():
    if os.path.exists(POST_DATA_FILE):
        with open(POST_DATA_FILE, 'r') as f:
            posts = json.load(f)
    else:
        posts = []

    return render_template('market.html', posts=posts)

posts.insert(0, new_post) # 최신 글을 맨 앞에 추가
```

Part 2-2. 게시물 작성 및 업로드 (선택기능1)

Target user(구매자) : Senario 2



구매자



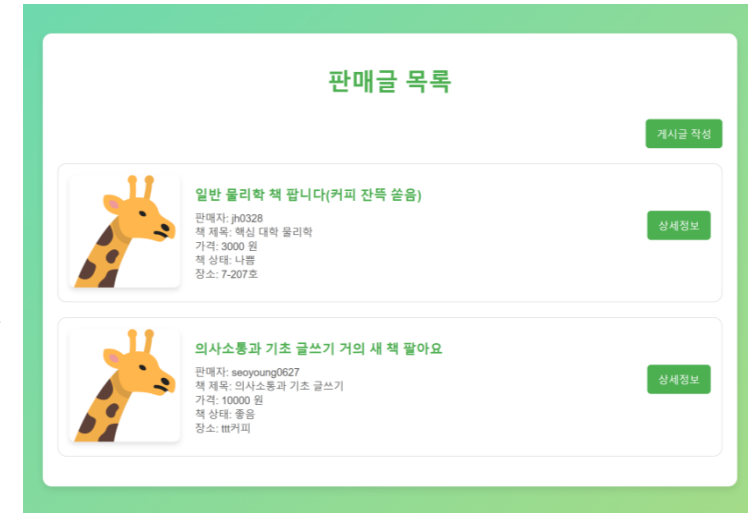
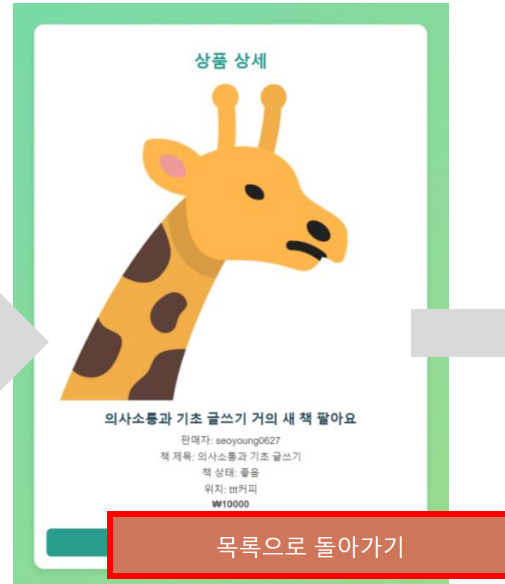
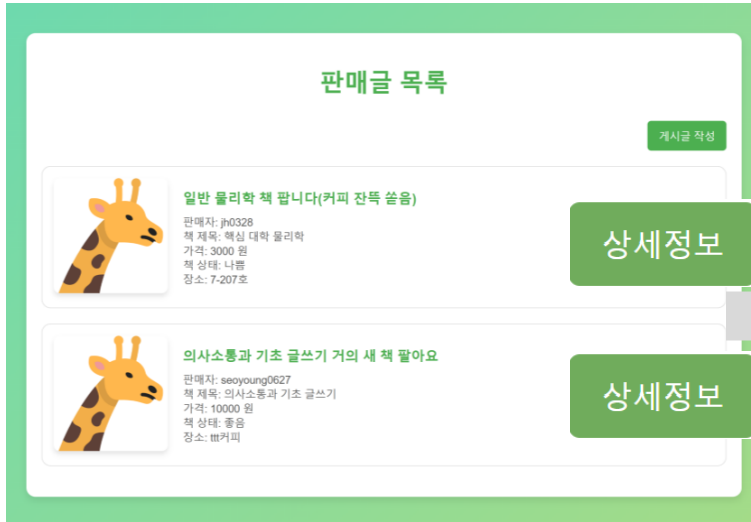
1. 로그인



2. 서적 검색



3. 구매자와 1:1 대화



상세정보 클릭 시 이동
작성자 id와 일치하는 정보를 posts에
불러오면서 상품 상세 페이지로 이동

목록으로 돌아가기 클릭 시 이동

```
<a href="/posts" class="back-btn">목록으로 돌아가기</a>
```

다시 market 페이지 확인 가능

```
@app.route('/post/<int:id>', methods=['GET'])
def post_detail(id):
    if os.path.exists(POST_DATA_FILE):
        with open(POST_DATA_FILE, 'r') as f:
            posts = json.load(f)
    else:
        posts = []

    post = next((p for p in posts if p['id'] == id), None)
    if not post:
        flash("Post not found!")
```

Part 2-2. 게시물 작성 및 업로드 (선택기능1)

Target user(구매자) : Senario 3



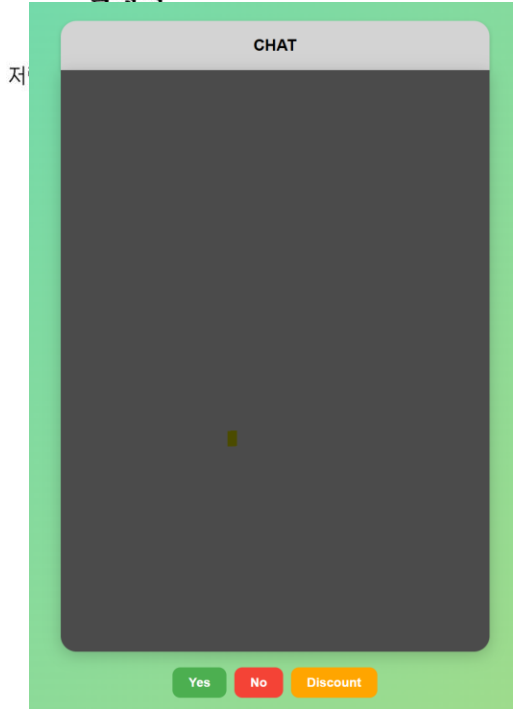
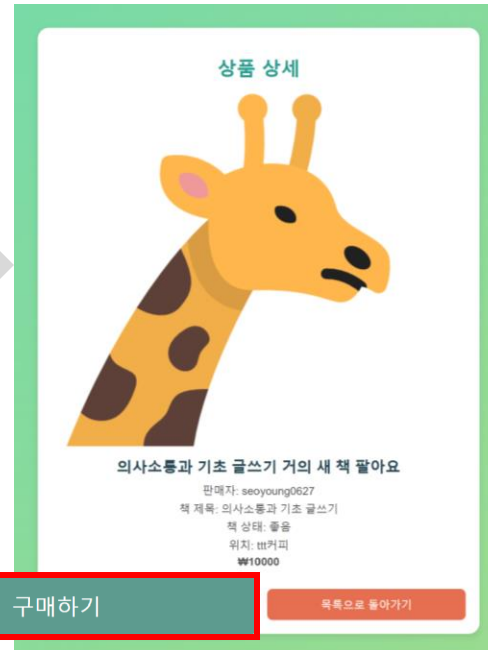
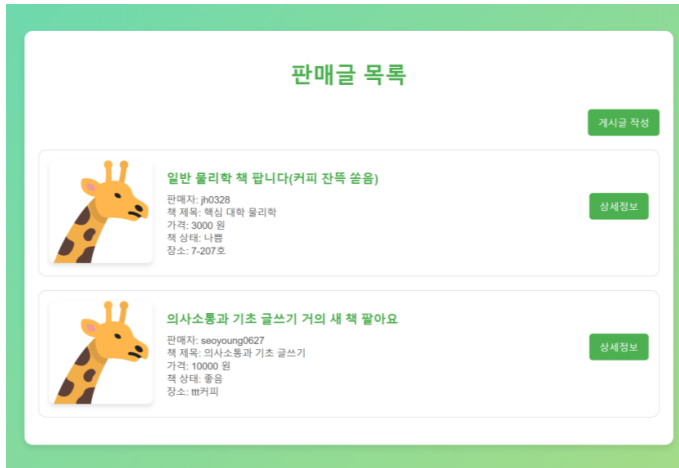
1. 로그인



2. 서적 검색



3. 구매자와 1:1 대화



상세정보 클릭 시 이동
작성자 id와 일치하는 정보를 posts에
불러오면서 상품 상세 페이지로 이동

```
@app.route('/buy/<int:id>', methods=['GET'])
def buy_post(id):
    # id가 유효한지 확인
    if os.path.exists(POST_DATA_FILE):
        with open(POST_DATA_FILE, 'r') as f:
            posts = json.load(f)
    else:
        posts = []

    post = next((p for p in posts if p['id'] == id), None)
    if not post:
        flash("게시글을 찾을 수 없습니다!")
        return redirect(url_for('get_posts'))

    # 해당 id를 사용하여 채팅 페이지로 리다이렉트
```

구매하기 클릭 시 채팅으로 이동

판매자와 채팅 가능

Part 3-1. 네트워크 (필수기능 3)

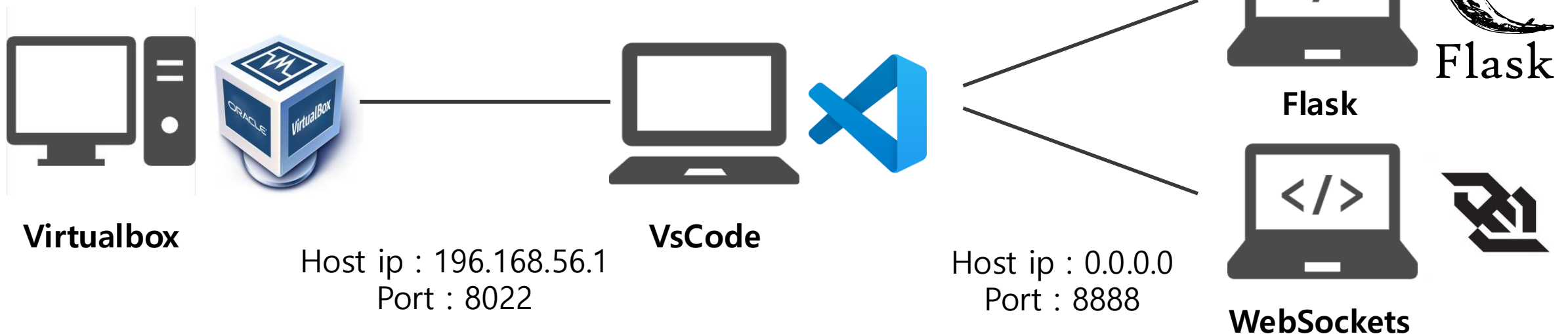
네트워크 어댑터 설정

- NAT(Network Address Translation) 방식으로 연결 설정.
- 네트워크 어댑터 1 활성화로 가상 환경과 호스트 간 통신 구현.

가상 환경과 호스트 통신

- VirtualBox 내에서 Flask와 WebSocket이 실행됨.
- 호스트(VS Code)와 게스트 간 데이터 송수신 가능.
 - Flask (Host IP: 0.0.0.0, Port: 5000)
 - WebSocket (Host IP: 0.0.0.0, Port: 8888)

3-1) 필수 기능



Part 3-1. 네트워크 (필수기능 3)



로그인 시스템

회원가입과 로그인 기능 제공
(ID/PW/전공/email)

User 정보 저장

로그인 시스템에서
받은 정보를 저장

네트워크

데이터 전송과 통신을 위한
네트워크 기능 구현

서버 관리

안정적인 운영을 위한
모니터링과 블랙 기능

```
def is_port_in_use(port):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        return s.connect_ex(('localhost', port)) == 0

def terminate_process_on_port(port):
    """특정 포트를 사용하는 프로세스를 종료합니다."""
    try:
        result = subprocess.check_output(f"lsof -ti:{port}", shell=True).decode().strip()
        if result:
            subprocess.call(f"kill -9 {result}", shell=True)
            print(f"{port} 포트를 사용하는 프로세스를 종료했습니다.")
    except subprocess.CalledProcessError:
        print(f"{port} 포트를 사용하는 프로세스가 없습니다.")

def run_server():
    """server.py WebSocket 서버를 실행합니다."""
    if is_port_in_use(8888):
        print("8888 포트가 이미 사용 중입니다. 프로세스를 종료합니다...")
        terminate_process_on_port(8888)

    try:
        process = subprocess.Popen(['python3', 'server.py'], cwd=os.getcwd())
        print(f"server.py가 PID {process.pid}로 시작되었습니다.")
    except Exception as e:
        print(f"server.py를 시작하지 못했습니다: {e}")
```

8888 포트가 이미 사용 중입니다. 프로세스를 종료합니다...

8888 포트를 사용하는 프로세스를 종료했습니다.

server.py가 PID 91910로 시작되었습니다.

WebSocket 서버가 8888 포트에서 실행 중입니다.

Web.py

- Port :8888(WebSockets) 이 사용되고 있는지 확인 후
만약 사용되고 있다면, `sudo kill -9 {PID}` 명령어를 통해 port 사용하는 프로세스를 종료 시키는 기능
- Port :8888(WebSockets)의 Port는 작동되는지 직접 확인하기에 어려워 port 실행 여부 출력과 어떤 PID에서 시작되었는지 보여주는 기능

Part 3-2. 1:1 채팅 기능

```
# 클라이언트로부터 초기 데이터 수신
initial_data = await websocket.recv()
client_data = json.loads(initial_data)
post_id = client_data.get("post_id")
role = client_data.get("role")
user_id = client_data.get("user_id") # 클라이언트에서 전달된 user_id

print(f"Received connection: Post ID={post_id}, Role={role}, User ID={user_id}")

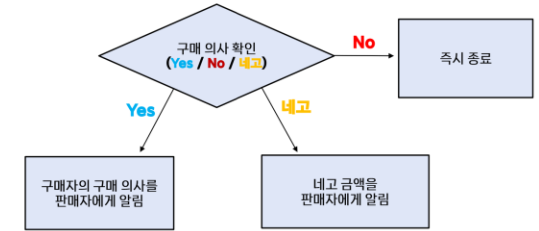
# 입력 데이터 검증
if not post_id or not role or not user_id:
    await websocket.send(json.dumps({"error": "필수 데이터가 누락되었습니다."}))
    return

# room_id 생성 (post_id와 user_id 결합)
room_id = f"{post_id}_{user_id}"

# 새로운 채팅 방 생성 또는 기존 방에 연결
if room_id not in chat_rooms:
    chat_rooms[room_id] = {"buyer": None, "seller": None}

# 역할 설정 및 WebSocket 연결
if role == "buyer":
    chat_rooms[room_id]["buyer"] = websocket
elif role == "seller":
    chat_rooms[room_id]["seller"] = websocket
else:
    await websocket.send(json.dumps({"error": "잘못된 역할"}))
    return
```

3-3) 선택 기능 2. 1:1 채팅



server.py

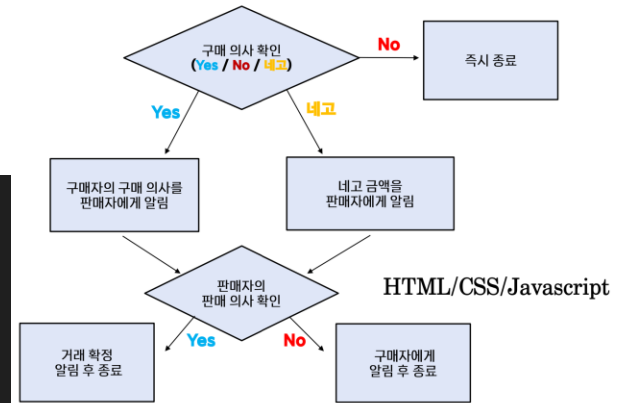
- 중복되지 않는 채팅방 생성을 위해 구매자의 "user_id"와 판매자의 "post_id"를 결합해서 "room_id"를 생성 함
 - "room_id" 생성 이유
 - 독립적인 채팅방을 생성하지 않으면 1:1이 아닌 다수의 채팅방이 만들어짐
- 등록된 정보들을 기반으로 구매하기를 눌렀을 때 구매자/판매자의 역할이 정해지도록 함
 - post_id, user_id는 사용자들이 등록한 정보가 저장되어있는 디렉토리에서 받아옴

Part 3-2. 1:1 채팅 기능

```
let isBuyerTurn = true; // 구매자 차례인지 판매자 차례인지 확인
let isChatClosed = false; // 채팅이 종료된 상태를 추적
const ws = new WebSocket('ws://10.0.2.15:8888'); // 적절한 IP로 변경
```

```
ws.onopen = () => {
  console.log("WebSocket 연결 성공!");
  const role = new URLSearchParams(window.location.search).get("role");
  if (role && role === "buyer") {
    ws.send('buyer'); // 구매자 역할로 설정
    console.log("구매자 역할로 설정됨");
  } else {
    ws.send('seller'); // 판매자 역할로 설정 (기본값)
    console.log("판매자 역할로 설정됨");
  }
};
```

3-3) 선택 기능 2. 1:1 채팅



chat.html

- "role" 파라미터를 통해 role = buyer가 포함
되 있을 경우 구매자 역할로 설정이 되고 그
외에는 판매자 역할로 설정

ex) <https://example.com/chat?role=buyer>
<https://example.com/chat?role=seller>

Part 3-2. 1:1 채팅 기능

chat.html

```
function updateButton() {
  const buttonContainer = document.getElementById('button-container');
  buttonContainer.innerHTML = '';

  if (isBuyerTurn) {
    const yesButton = document.createElement('button');
    yesButton.textContent = 'Yes';
    yesButton.classList.add('yes-button');
    yesButton.onclick = () => sendResponse('Yes');

    const noButton = document.createElement('button');
    noButton.textContent = 'No';
    noButton.classList.add('no-button');
    noButton.onclick = () => sendResponse('No');

    const negoButton = document.createElement('button');
    negoButton.textContent = 'Discount';
    negoButton.classList.add('nego-button');
    negoButton.onclick = () => sendResponse('Discount');

    buttonContainer.appendChild(yesButton);
    buttonContainer.appendChild(noButton);
    buttonContainer.appendChild(negoButton);
  } else {
    const yesButton = document.createElement('button');
    yesButton.textContent = 'Yes';
    yesButton.classList.add('yes-button');
    yesButton.onclick = () => sendResponse('Yes');

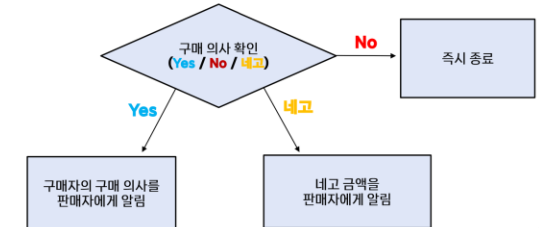
    const noButton = document.createElement('button');
    noButton.textContent = 'No';
    noButton.classList.add('no-button');
    noButton.onclick = () => sendResponse('No');

    buttonContainer.appendChild(yesButton);
    buttonContainer.appendChild(noButton);
  }
}
```

역할에 따른 채팅버튼 및 메시지

1. 구매자/판매자가 역할에 맞는 함수 내에서 버튼 선택
2. 그 후 버튼에 맞는 메시지 출력

3-3) 선택 기능 2. 1:1 채팅



이 페이지에는
네고하고 싶은 가격을 입력하세요:

확인 취소

이 페이지에는
채팅을 종료합니다.

확인

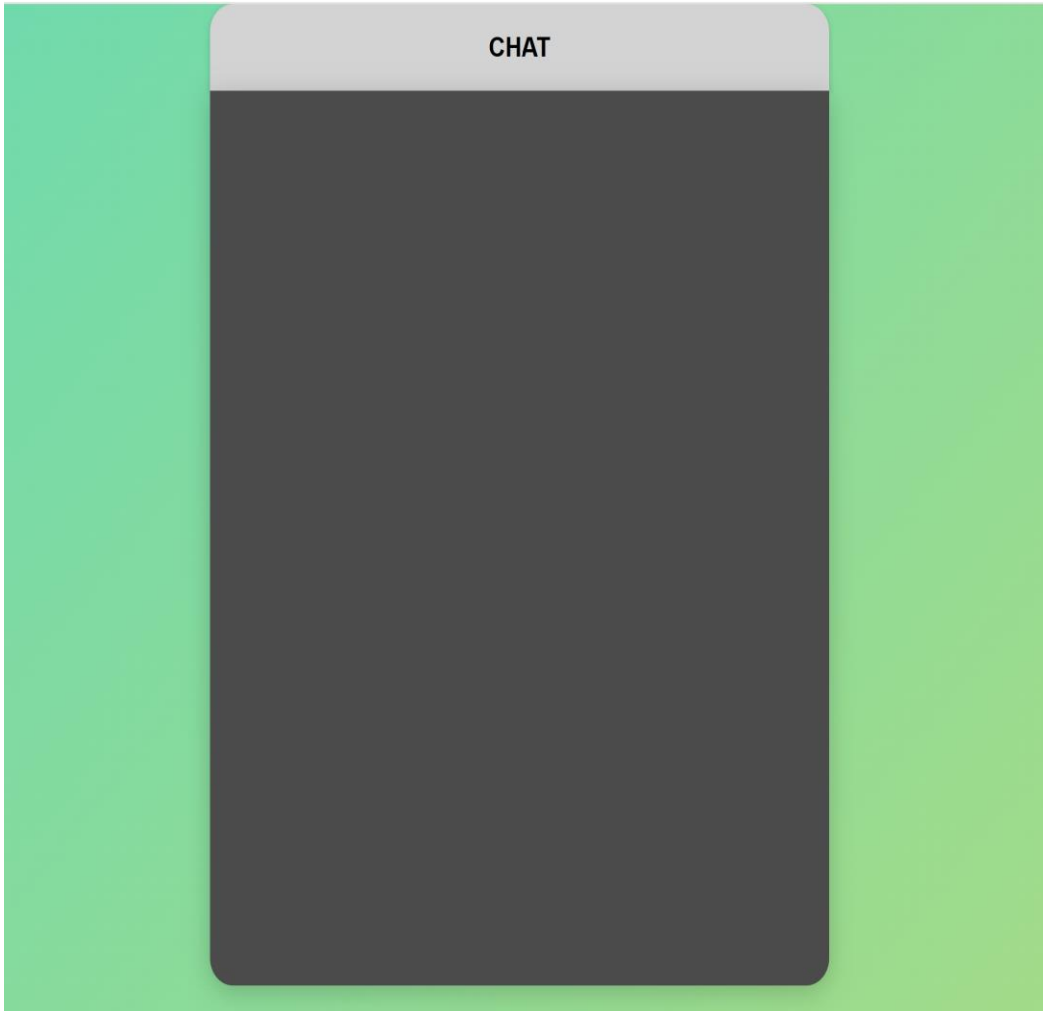
구매자: 9900원으로 네고 가능할까요?

판매자: 거래가 성립되었습니다.

판매자: 9900원에 판매하겠습니다!

Part 3-2. 1:1 채팅 기능

<구매자/판매자 설정 알고리즘 추가 후>



예상되는 문제점

1. 구매하기 버튼을 눌렀을 때 구매자를 구별하지 못함
2. updateButton() 함수가 정상작동 되지 않음
3. client 정보가 server 로 정상적으로 전달되지 않음
4. Websocket이 연결 됐을 때 함수가 정상작동 하지 않음



감사합니다

SW개발환경 및 응용 11조 김재혁 김서영 정혜정