AI 기반 코딩/스터디 플랫폼

이미지 기반 문제 인식부터 코드 생성, 피드백까지 지원하는 AI 코딩 스터디 플랫폼

팀 7
20230851 신아라
20231384 박성진
20210877 최우성
20231388 이경현
20221380 이민경

목차

1.1 선정 배경

AI 도구(Copilot, ChatGPT)는 발전했지만
→ **학습자들은 여전히 문제만 풀고 끝남**

실제 개발 현장에서는
→ **코드 리뷰, 이슈 관리, 협업, AI 활용**이 더 중요

대학 과제나 코딩 스터디에서는
→ **코드 품질 피드백이나 개인화 학습 경로 제공이 부족**

AI와 협업하는
실전형 플랫폼

1.2 프로젝트 목표

AI와 함께 문제 해결, 코드 공유, 리뷰까지 가능한 통합 코딩 학습 플랫폼 구축

2. 주요 기능

1. 이미지 업로드 및 흐림도 분석

- 사용자가 업로드한 이미지의 흐림 정도(Blur Score)를 OpenCV로 분석

2. 문제 데이터 관리 및 API 응답

- 문제별 폴더에 설명/코드 보관
- /problem/<id> API를 통해 문제 내용 JSON 응답 가능

3. LLM 기반 자연어 질문 응답

- 사용자의 질문에 대해 로컬 LLaMA 모델이 텍스트 또는 코드 형태로 응답 생성

2. 주요 기능

4. 다이어그램 구조 인식 및 코드 자동 생성

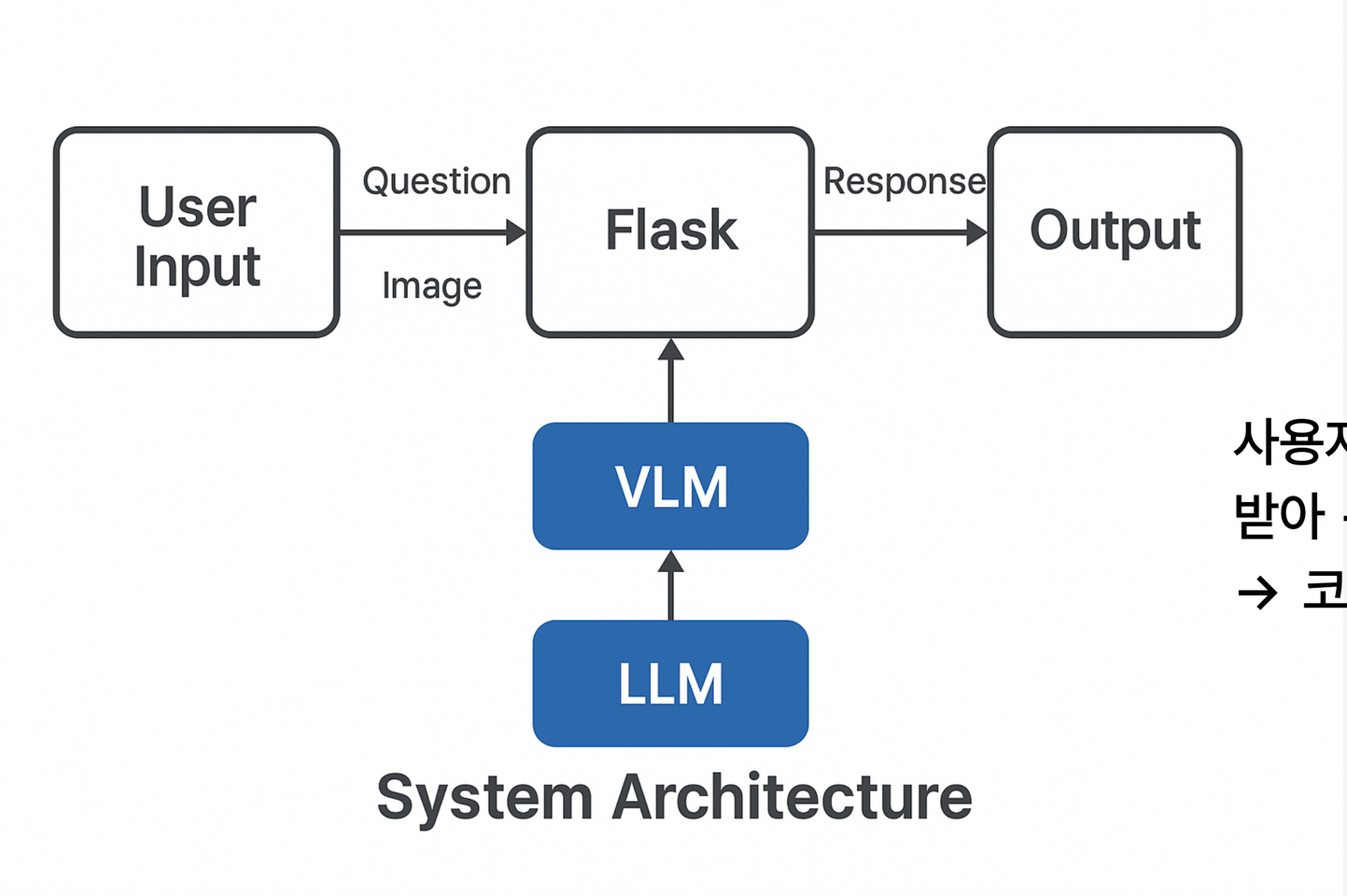
- 트리 / 그래프 이미지 업로드 시 구조 자동 분석
- 인식된 노드 및 엣지 정보를 기반으로 Python 코드(BST, Graph) 자동 생성

5. 다이어그램 코드 생성 처리 과정 (OCR + VLM)

- 이미지 업로드 → EasyOCR로 텍스트 추출
- LLava (VLM)에 이미지·텍스트 입력
- Python 코드 자동 생성·출력

(이미지 → EasyOCR → LLava 순서로 처리)

3. 전체 구조도



사용자 입력(질문/이미지)을 Flask 서버가 받아 → 입력 타입에 따라 LLM 또는 VLM 처리 → 코드/설명 결과를 생성 → 웹 페이지로 출력

4. 주요 코드 설명

```python
@app.route("/generate", methods=["POST"])
def generate():
    question = request.form.get("question")
    image_file = request.files.get("image")

    if image_file and image_file.filename:
        image_path = os.path.join(UPLOAD_FOLDER, image_file.filename)
        image_file.save(image_path)
        code_path = run_diagram_pipeline(image_path)
        with open(code_path, 'r') as f:
            answer = f.read()
    else:
        answer = get_llm_response(question)

    return answer
```

main.py

- generate() 함수 (Flask 라우터)
사용자의 입력이 텍스트인지 이미지인지 판단하고, 적절한 AI 처리로 분기함

4. 주요 코드 설명

```python
def run_diagram_pipeline(image_path: str):
    explicit = detect_label(image_path)
    numbers = extract_numbers_easyocr(image_path)
    needs_pre = blur_score(image_path) <= BLUR_THRESH

    try:
        vlm_out = recognize(image_path)
        nodes, edges = parse_structure(vlm_out)
        if needs_pre or not nodes or not edges or vlm_out.get("confidence", 0) <= CONF_THRESH:
            raise ValueError
    except Exception:
        img = preprocess(image_path)
        temp_path = "temp.png"
        save_temp(img, temp_path)
        vlm_out = recognize(temp_path)
        nodes, edges = parse_structure(vlm_out)

    if explicit:
        vlm_out['type'] = explicit
    if vlm_out.get("type") == "bst" and numbers:
        nodes = [{"value": n} for n in numbers]
        edges = []

    print(f"Type: {vlm_out.get('type')}, Nodes:{len(nodes)}, Edges:{len(edges)}, Conf:{vlm_out.get('confidence',0):.2f}")

    if vlm_out.get("type") == "bst":
        code = gen_bst_code(nodes, edges)
        out_file = "bst.py"
    else:
        code = gen_graph_code(nodes, edges)
        out_file = "graph.py"

    with open(out_file, "w") as f:
        f.write(code)
    print(f"Generated code saved to {out_file}")
    return out_file
```

main.py

이미지 전처리 및 정보 추출

시각 모델을 통한 구조 인식

유형 보정 및 구조 재구성

코드 생성 및 파일 저장

- run_diagram_pipeline 함수

이미지 → 코드 생성의 핵심 처리 로직

4. 주요 코드 설명

```python
PROJECT_ROOT = os.path.dirname(os.path.abspath(__file__))
DATA_DIR = os.path.join(PROJECT_ROOT, "data")
SOLUTIONS_DIR = os.path.join(PROJECT_ROOT, "solutions")

with open(os.path.join(DATA_DIR, "problems.json"), encoding="utf-8") as f:
    problems_data = json.load(f)

@app.route("/problems", methods=["GET"])
def get_problems():
    return {
        "problems": [
            {
                "id": p["id"],
                "title": p.get("title", ""),
                "name": p.get("name", ""),
                "difficulty": p.get("difficulty", "")
            } for p in problems_data
        ]
    }

@app.route("/problem/<pid>", methods=["GET"])
def get_problem(pid):
    prob = next((p for p in problems_data if str(p["id"]) == pid), None)
    if not prob:
        return {"error": "Problem not found"}, 404
    return prob
```

- 파일 로딩 및 경로 설정

problems.json 파일을 불러오기 위한 디렉토리 설정 및 데이터 로딩

- get_problems() 함수

/problems 라우터 (GET)

모든 문제 목록을 JSON 배열 형태로 반환하는 API

- get_problem(pid) 함수

특정 문제 ID에 해당하는 데이터를 찾아 JSON으로 반환하는 API

4. 주요 코드 설명

```
def recognize(image_path: str) -> dict:
```

diagram_sketch/diagram_recognizer.py

- recognize(image_path) 함수
이미지에서 트리/그래프 구조를 인식하는 VLM 기반 핵심 모델 호출 함수

```
def gen_bst_code(nodes: List[Dict], edges: List[Dict]) -> str:
    lines = [
for n in nodes:
    lines.append(f"    bst.insert({n['value']})")
```

diagram_sketch/code_generator.py

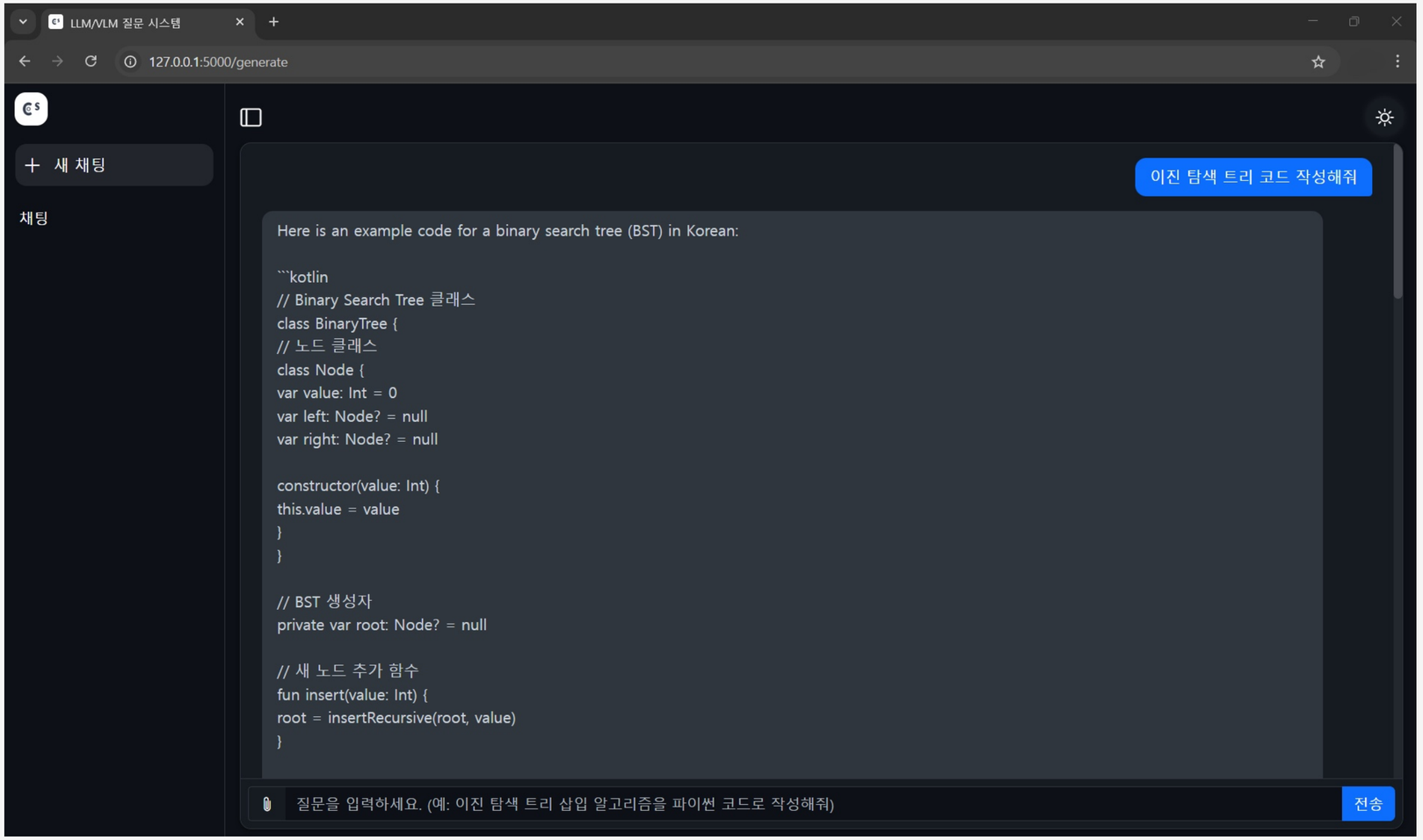
- gen_bst_code(nodes, edges) 함수
노드 리스트를 기반으로 BST 삽입 코드를 자동 생성하여 실행 가능한 파이썬 코드 형태로 반환

```
for edge in edges:
    if isinstance(edge, dict):
        u = edge['from']
        v = edge['to']
    else:
        u, v = edge
    lines.append(f"    g.add_edge({u}, {v})")
```

diagram_sketch/code_generator.py

- for edge in edges 반복문
그래프 간선 정보를 바탕으로 연결 코드(add_edge)를 자동 생성하는 구문.

5. 구현 화면



5.1 코드 생성 기능

입력 : "이진 탐색 트리 코드 작성해줘"

출력 : Python 코드 자동 생성

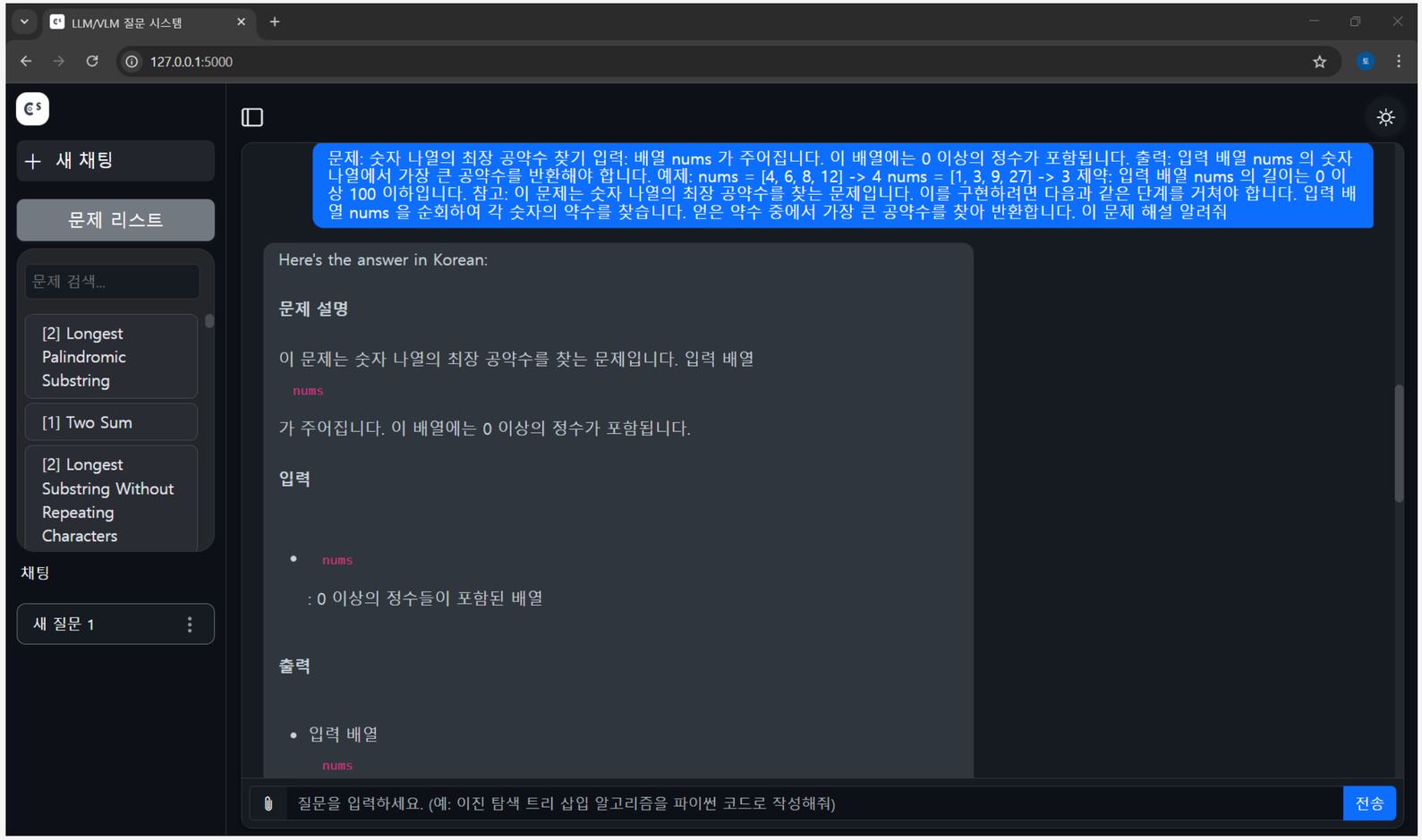
5. 구현 화면

5.2 문제 생성 기능

입력 : "leetcode 문제 만들어줘"

출력 : 문제 설명 + 입출력 예시 자동 생성

5. 구현 화면

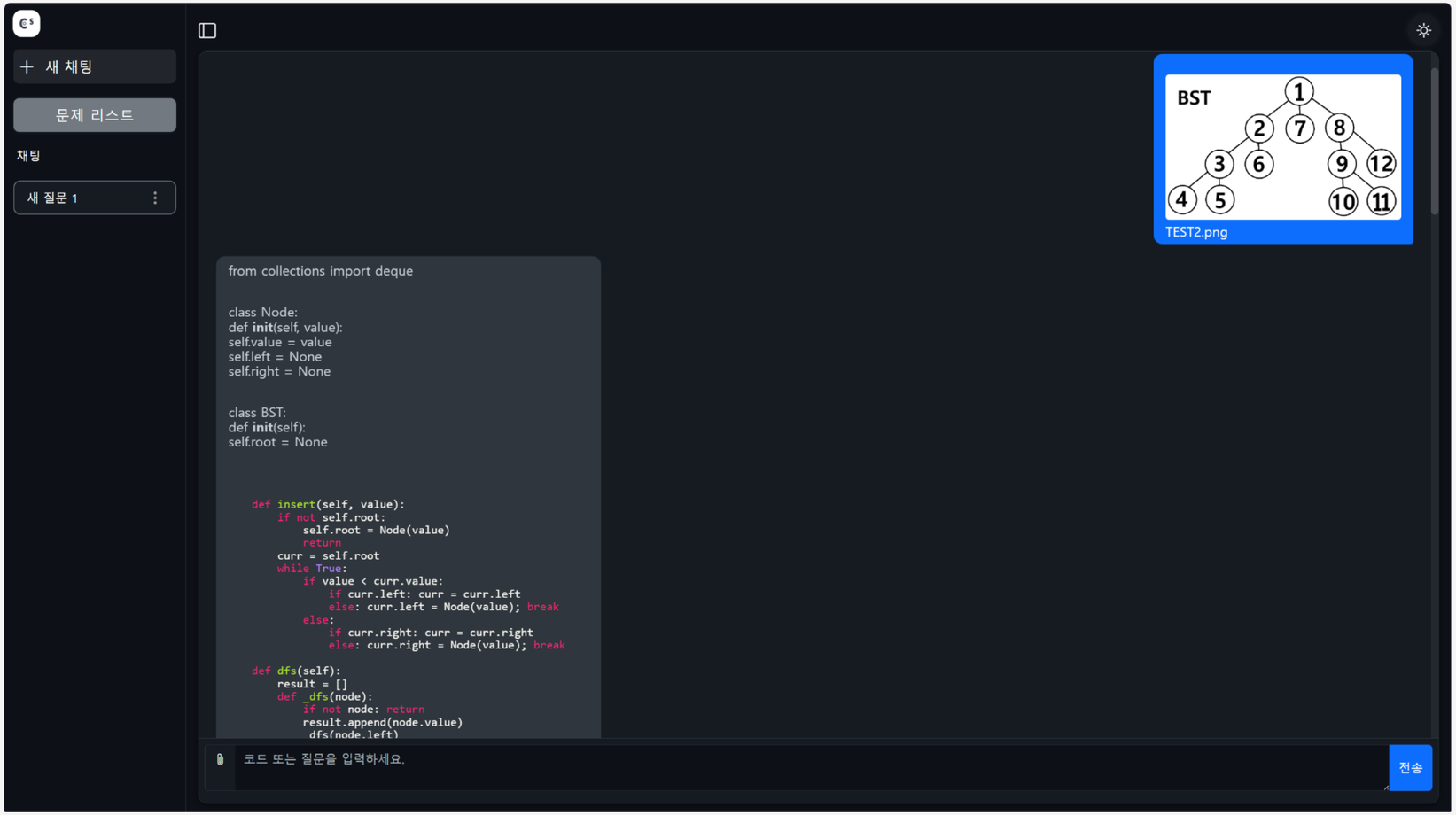


5.3 문제 해설 기능

입력 : 문제 지문

출력 : 상세 해설 및 풀이 과정 출력

5. 구현 화면

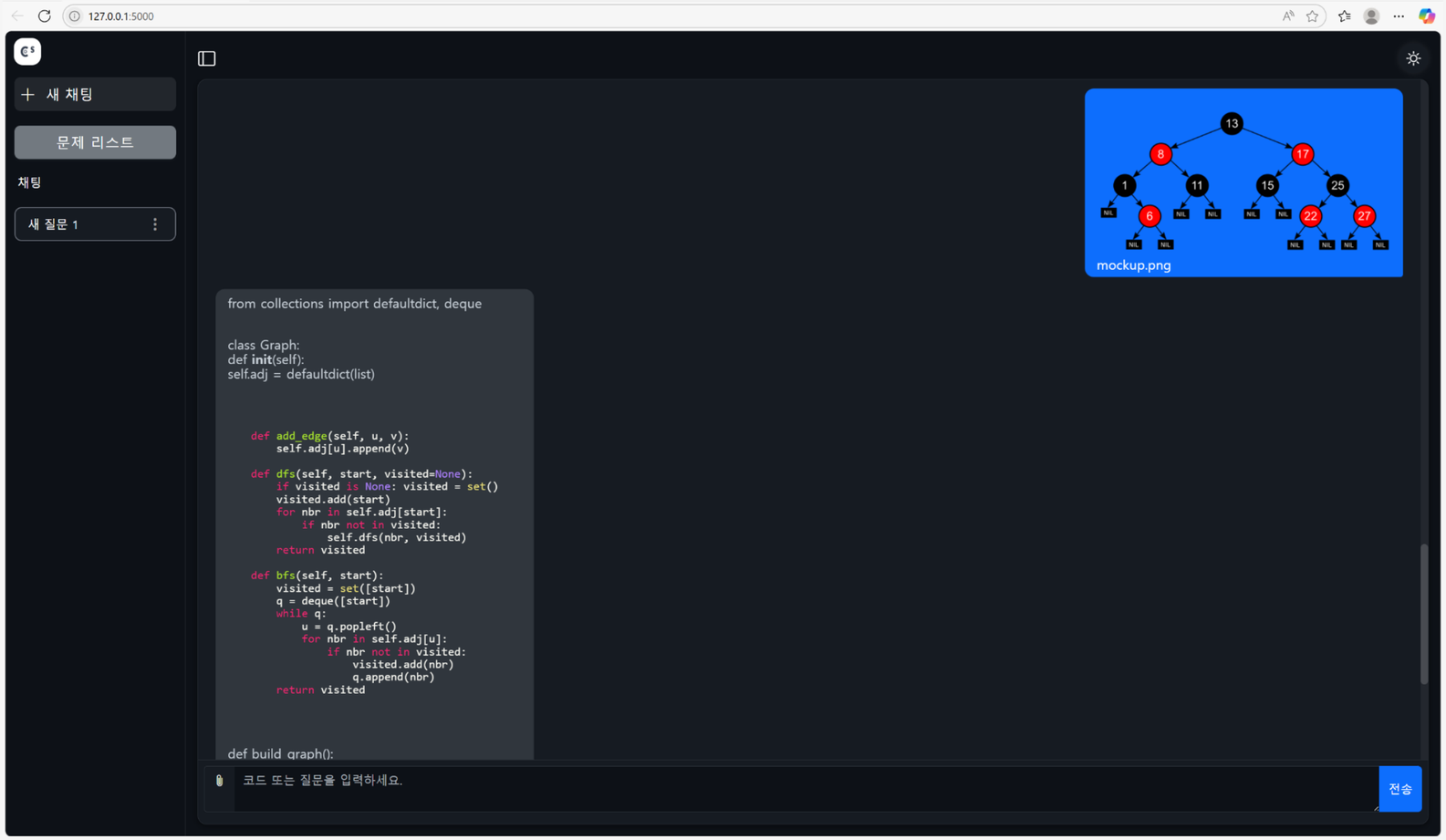


5.4 이미지 기반 코드 생성

입력 : BST 구조 이미지 업로드

출력 : 해당 구조를 구현하는 코드 생성

5. 구현 화면



5.5 이미지 기반 그래프 분석

입력 : 그래프 구조 이미지 업로드

출력 : 그래프 탐색(DFS, BFS) 코드 생성

6. 사용 기술

백엔드 : Python, Flask

모델 : Ollama (LLaVA, LLaMA), EasyOCR

프론트엔드 : HTML/CSS, JavaScript

기타 : VSCode, 명령어 기반 인터페이스

7. 팀원 및 업무분담

20231384 박성진
- 개발

20231388 이경현
- 개발

20221380 이민경
- 개발

20230851 신아라
- PPT 제작

20210877 최우성
- 발표

감사합니다