AI 기반 실시간 모니터링으로
산불 조기 감지 및 대응

지능형 산불 연기 발생 상황 분석 시스템

구교웅

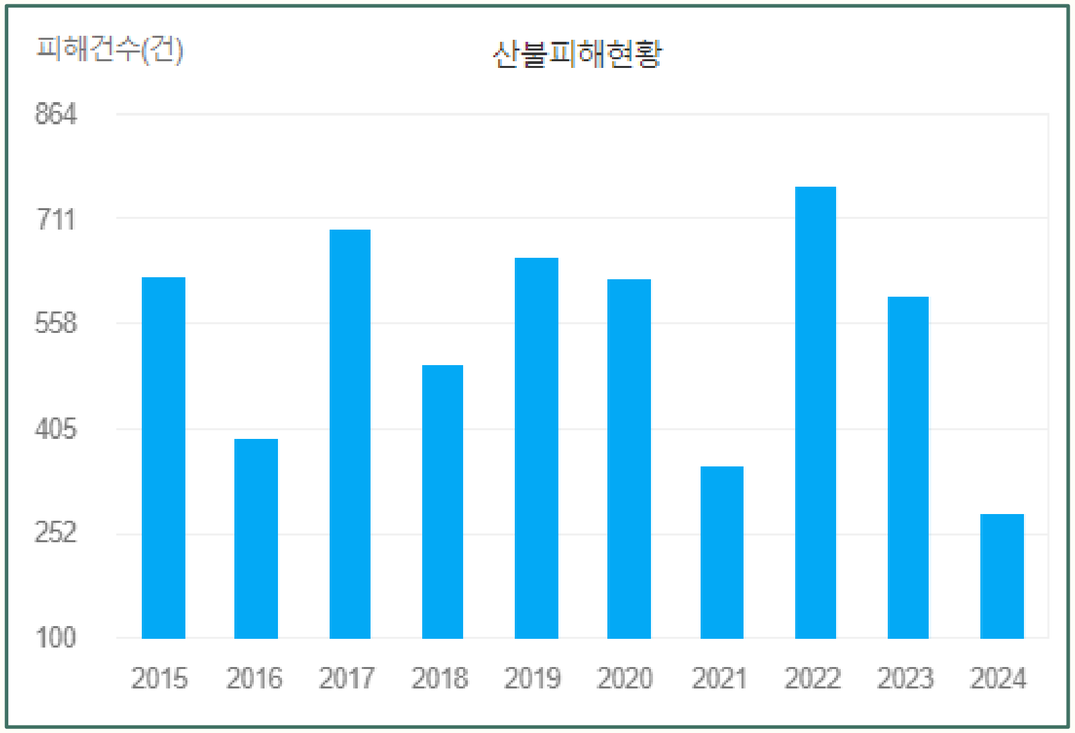
이수민

이창조

이현성

목차

수행 배경 및 목표



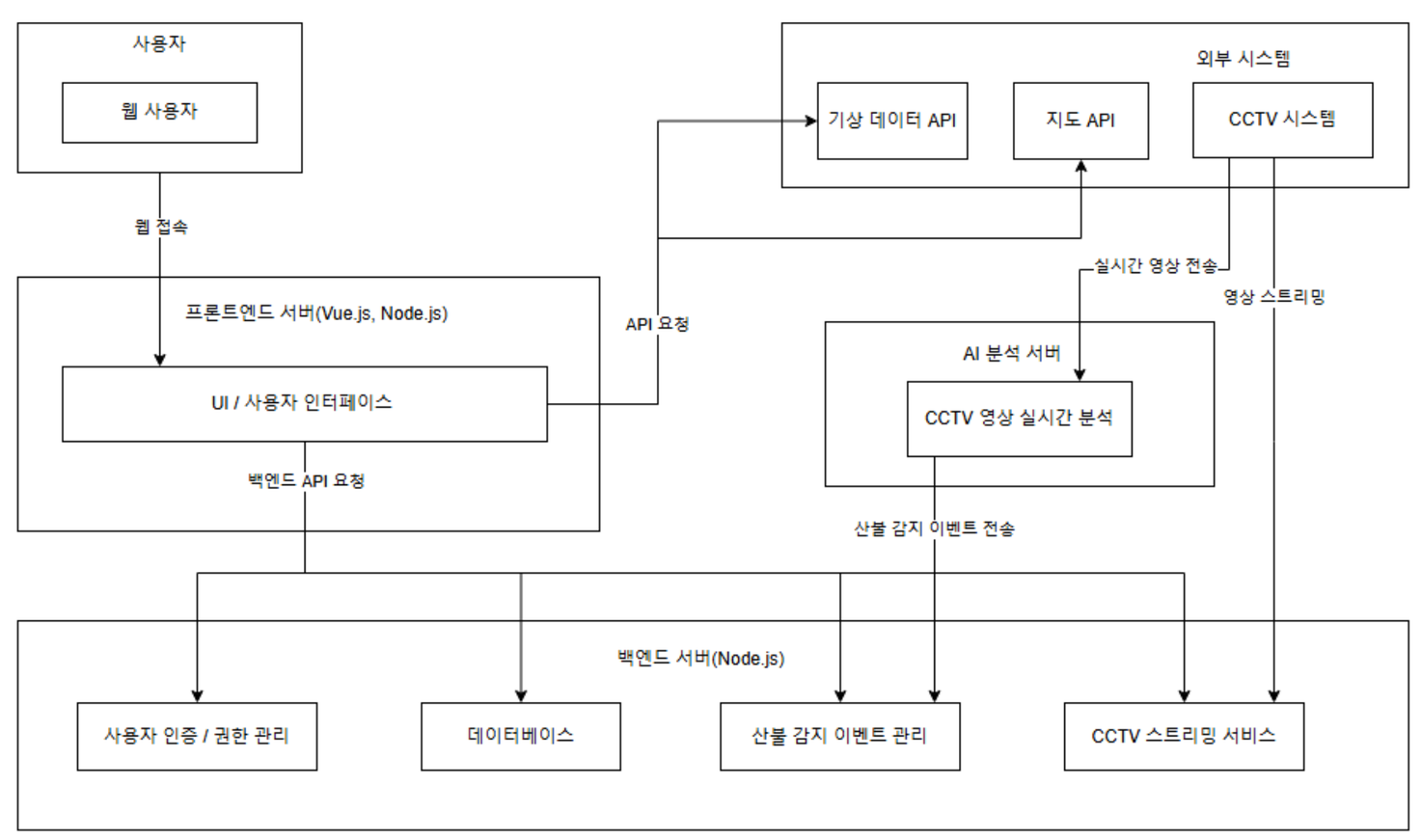
출처 : 산림청 「산불통계연보」

산불 조기 감지의 중요성

- 산불은 예측이 어렵고 초기 대응이 중요
- AI 기반 실시간 모니터링 시스템 필요
- 골든타임 내 진압으로 피해 최소화

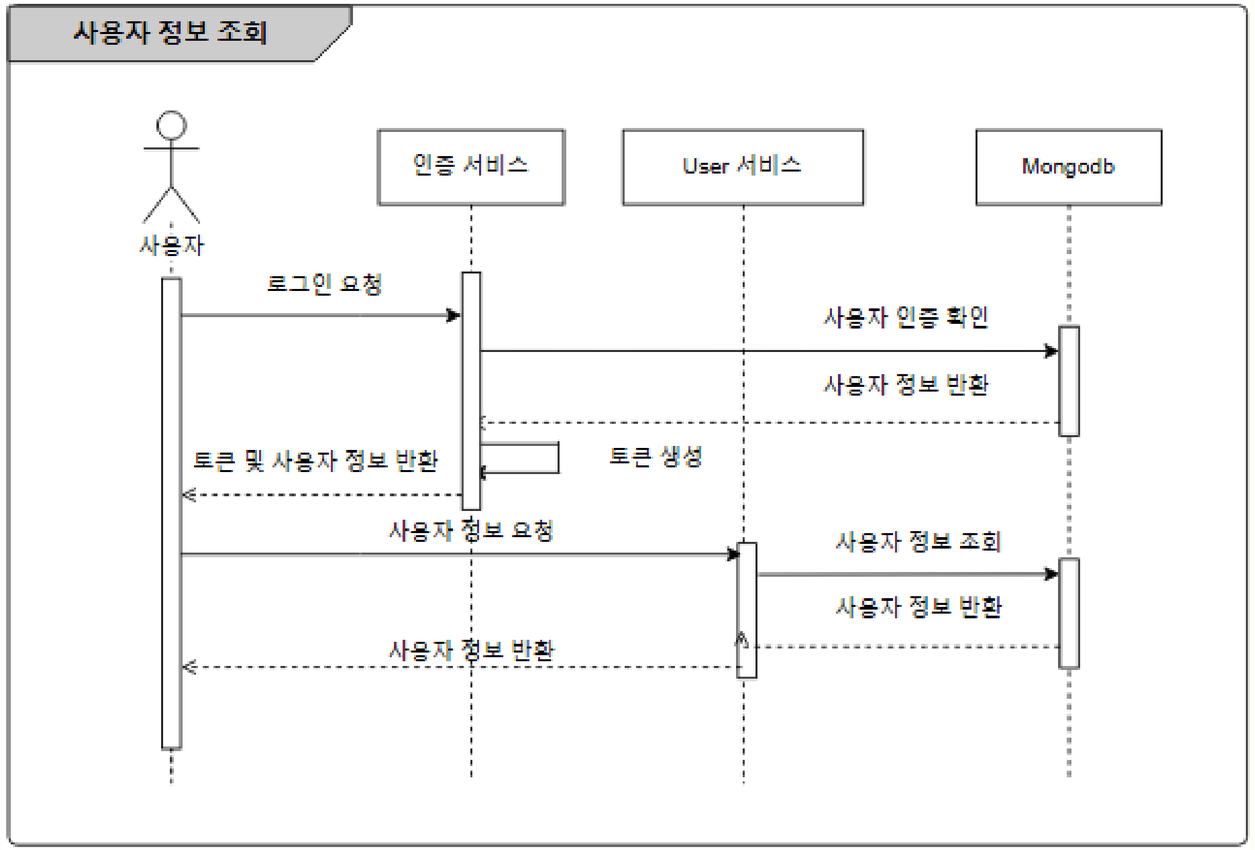
시스템 모델 및 설계

1. 시스템 구조



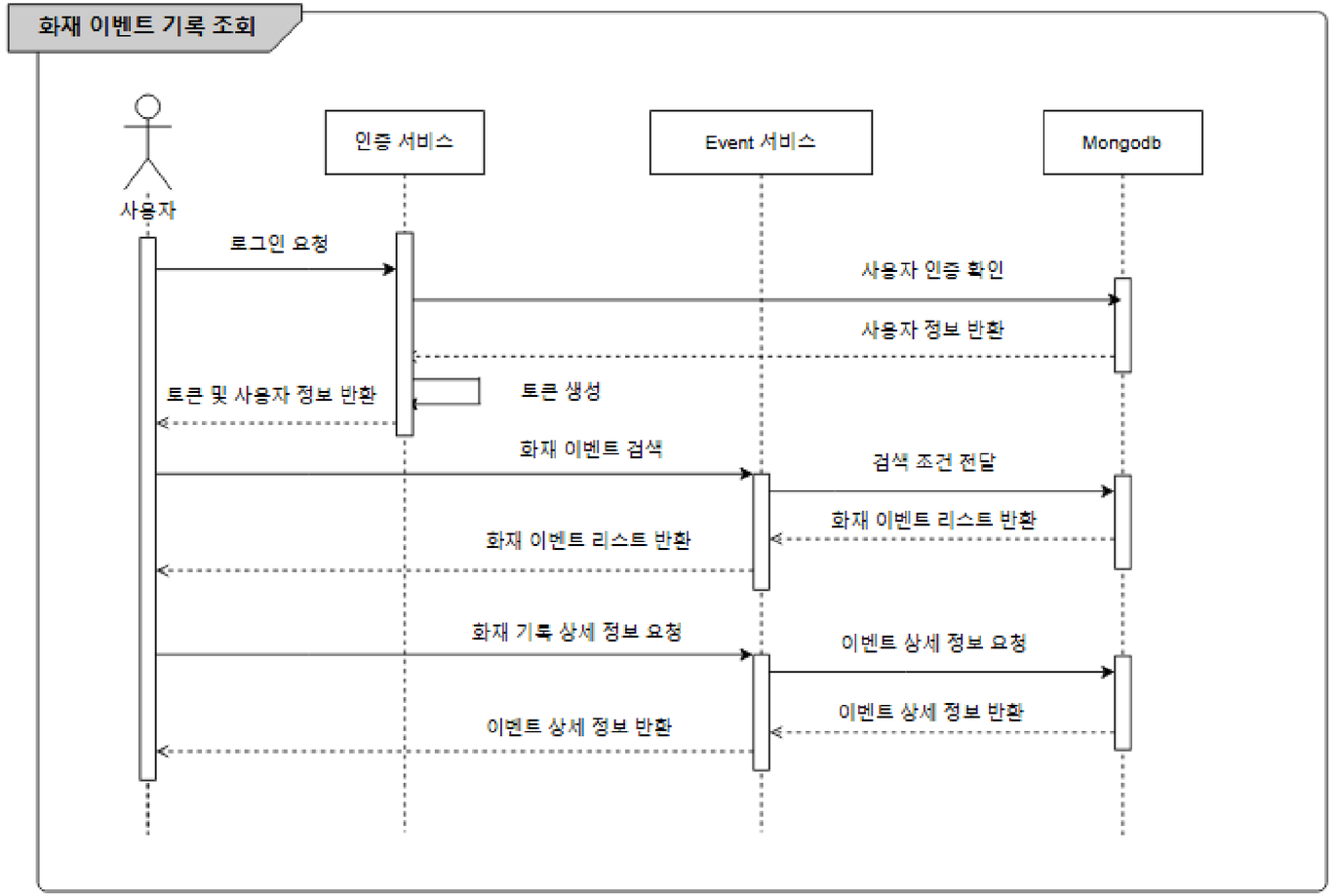
시스템 모델 및 설계

2. Sequence Diagram – 사용자 정보 조회



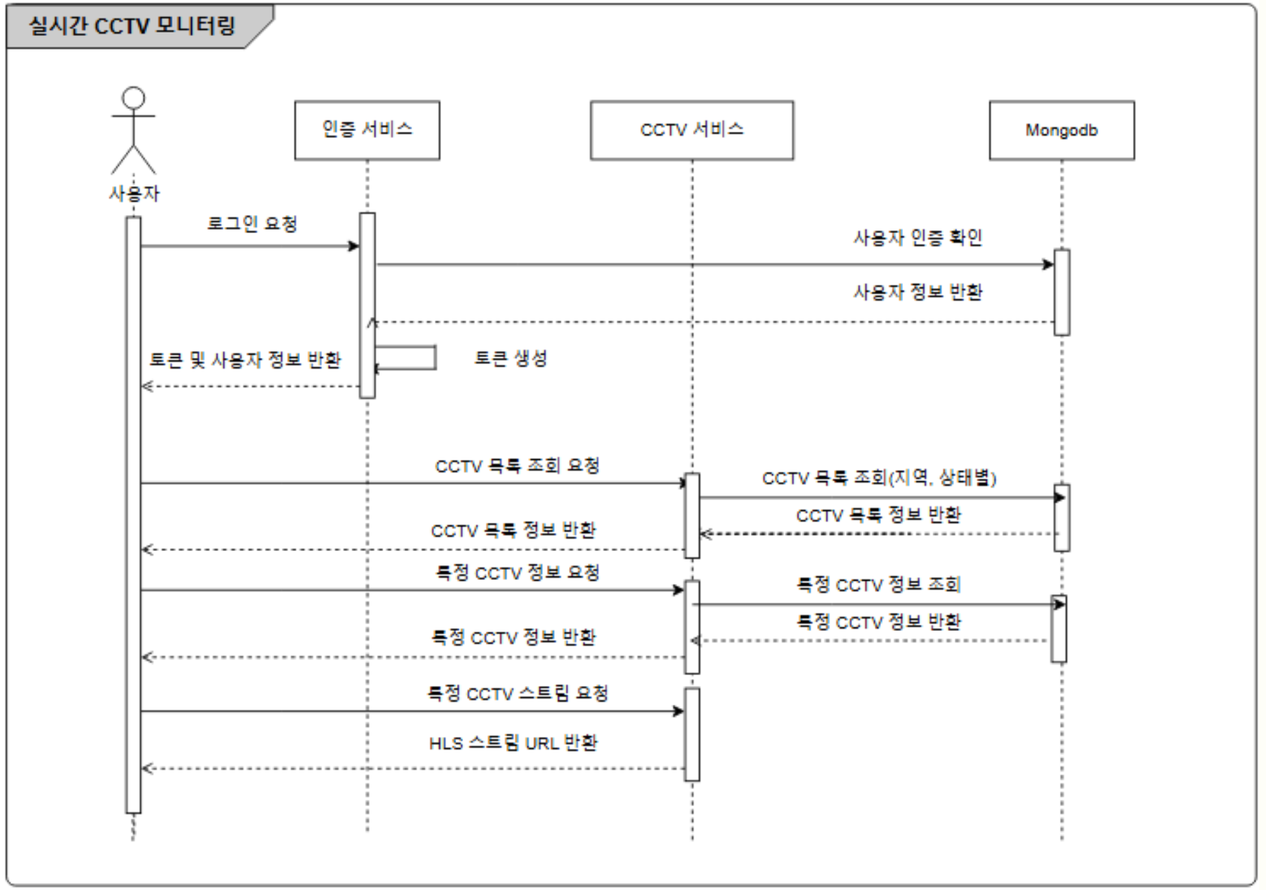
시스템 모델 및 설계

2. Sequence Diagram – 화재 이벤트 기록 조회



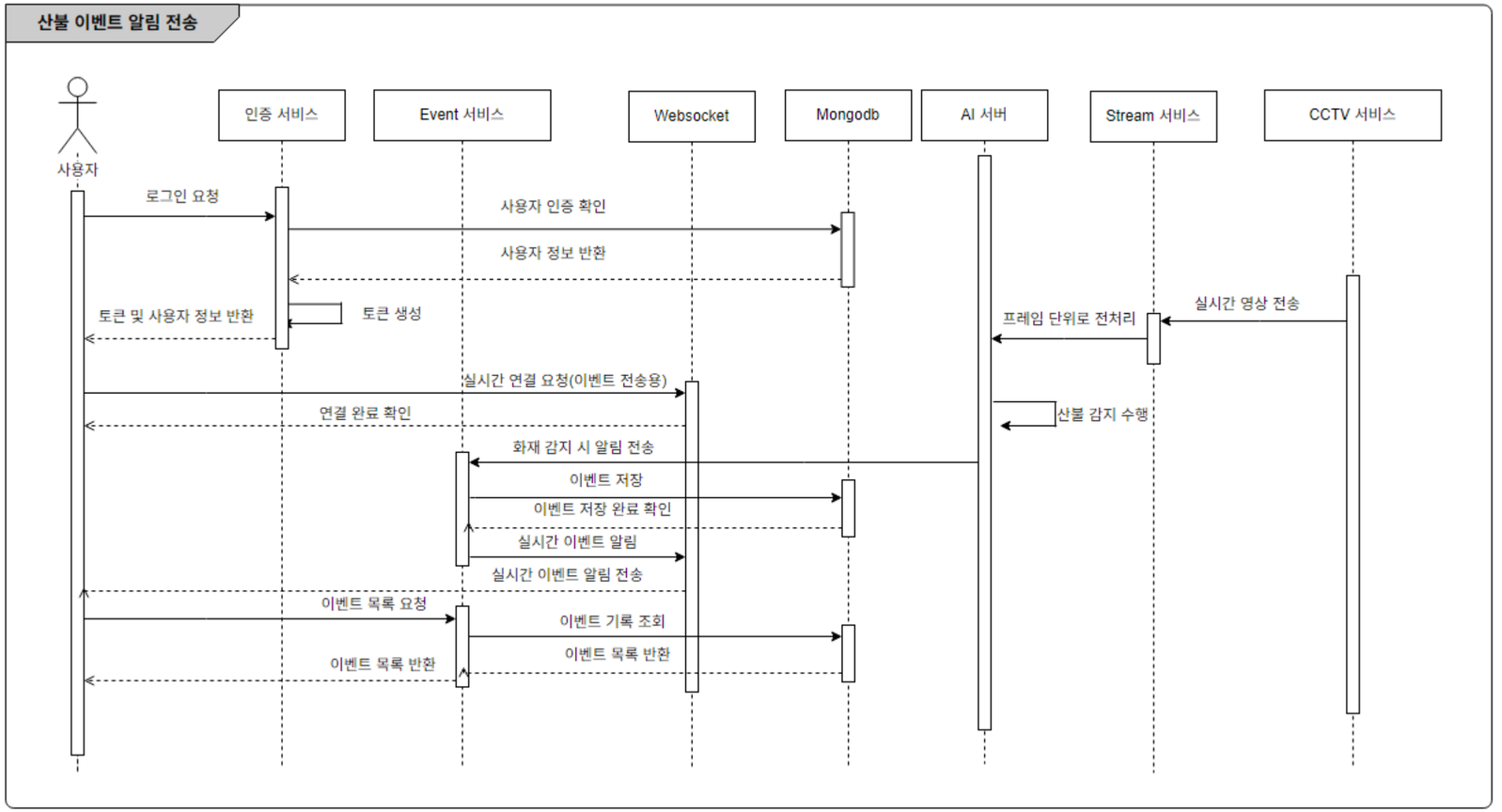
시스템 모델 및 설계

2. Sequence Diagram – 실시간 CCTV 모니터링



시스템 모델 및 설계

2. Sequence Diagram – 산불 이벤트 알림 전송



진행 상황

프런트 & 백엔드 - API 명세서

Index	기능	HTTP method	API path	Request	Token
[인증]	로그인	POST	api/auth/login	id password	X
[인증]	계정 생성(관리자)	POST	api/auth/admin/register	id password region role	O(admin)
[인증]	계정 삭제(관리자)	DELETE	api/auth/admin/users/	-	O(admin)
[인증]	로그아웃	POST	api/auth/logout	-	O
[인증]	토큰 갱신	POST	api/auth/refresh	refreshToken	X
[회원]	사용자 프로필 조회 (담당 지역, 권한, 이메일 , 전화번호)	GET	api/users/profile	-	O
[회원]	사용자 프로필 수정 (이메일, 전화번호)	PUT	api/users/profile	phone email	O
[CCTV]	특정 CCTV 정보 조회	GET	api/cctv/{cctvID}/stream	-	O
[CCTV]	CCTV 목록 조회	GET	api/cctv?region={region}&status={status}	-	O
[이벤트]	이벤트 리스트 조회	GET	api/events	region startDate endDate lastEventId limit	O
[이벤트]	이벤트 상세 조회	GET	api/events/{eventId}	-	O
[지역]	지역 정보 조회	GET	api/regions	-	O
[실시간]	실시간 스트림 HLS	GET	api/stream/hls/{cctvID}/{playlist}.m3u8	-	O
[실시간]	실시간 데이터 연결	WS	/ws/events	region	O
[이벤트, 실시간]	이벤트 수신	POST	/ai/detection	detectionData	O (AI 서버)

API 명세서를 바탕으로
프런트 & 백 에드는 작업.

AI-백엔드-프론트엔드 데이터 흐름

실시간 영상 처리 흐름

1. **AI 서버 - 영상 수신 및 처리**:
   - -CCTV에서 RTSP 스트림을 수신
   - -실시간 영상 분석 및 연기/화재 감지
   - -처리된 스트림을 백엔드로 전달
2. **백엔드 - 스트림 중계**:
   - -AI 서버에서 처리된 영상 데이터 수신
   - -HLS로 변환하여 HTTP를 통한 스트리밍 (높은 호환성)
   - -스트림 메타데이터 및 상태 관리
3. **프론트엔드 - 스트림 표시**:
   - -필요에 따라 적절한 스트리밍 방식 선택:
   - -안정성이 중요하므로 HLS 스트림 사용
   - -실시간성이 중요하면 Websocket 사용 가능
   - -영상 플레이어를 통해 사용자에게 표시
   - -WebSocket을 통한 스트림 상태 업데이트 수신

이벤트 감지 및 알림 흐름

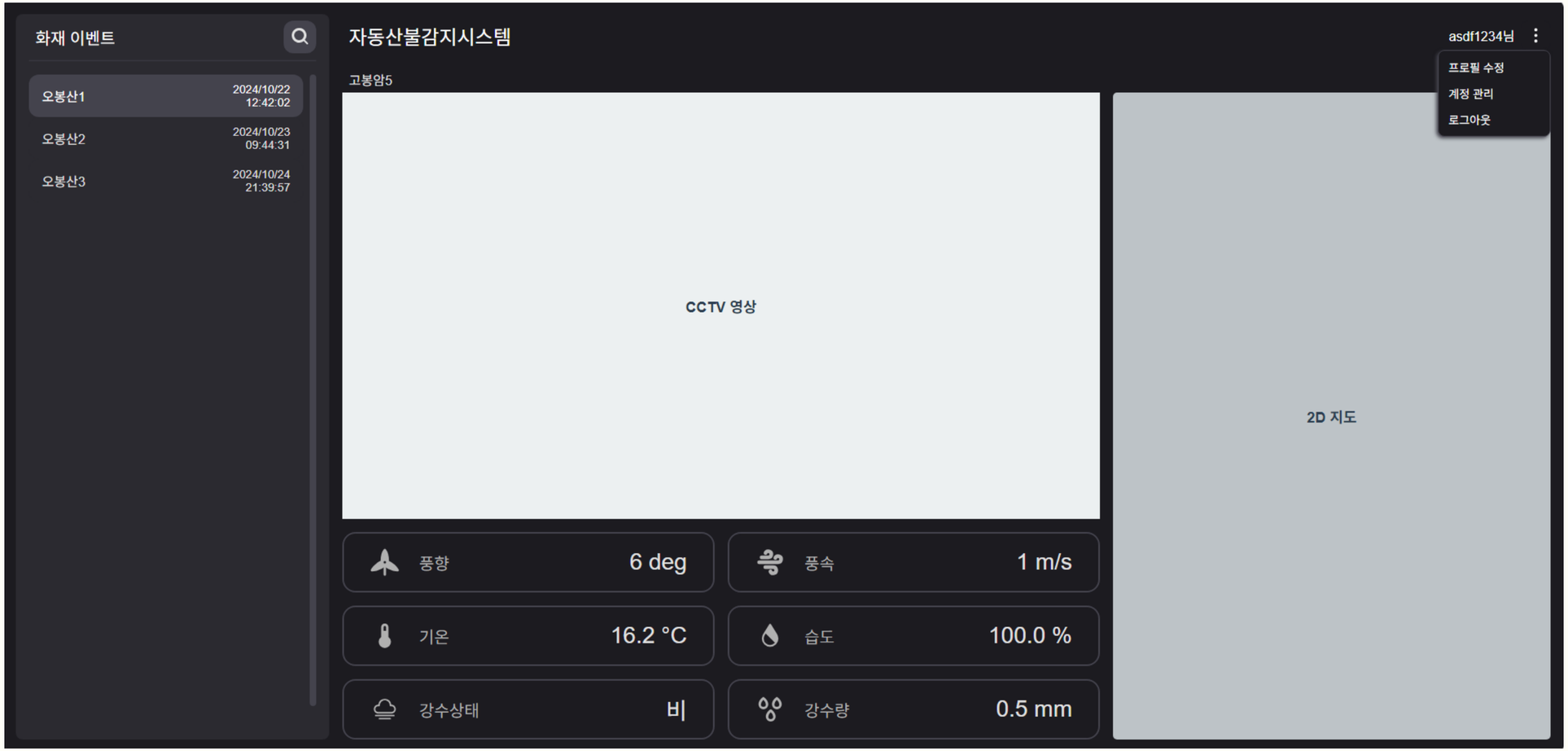
1. **AI 서버 - 이벤트 감지**:
   - -영상 스트림에서 산불 감지
   - -감지 정보(위치, 신뢰도, 이미지 등) 생성
   - -감지 데이터를 api/ai/detection API를 통해 백엔드로 전송
2. **백엔드 - 이벤트 처리**:
   - -AI 서버로부터 감지 데이터 수신 및 검증
   - -이벤트 생성 및 DB 저장
   - -이벤트 SMS 알림
   - -WebSocket을 통해 프론트엔드로 이벤트 알림 전송
3. **프론트엔드 - 이벤트 표시**:
   - -WebSocket을 통한 실시간 이벤트 알림 수신
   - -사용자 인터페이스에 이벤트 알림 표시
   - -이벤트 목록 업데이트 및 지도에 이벤트 위치 표시

진행 상황

프런트 엔드 - 웹 UI/UX 디자인(figma)

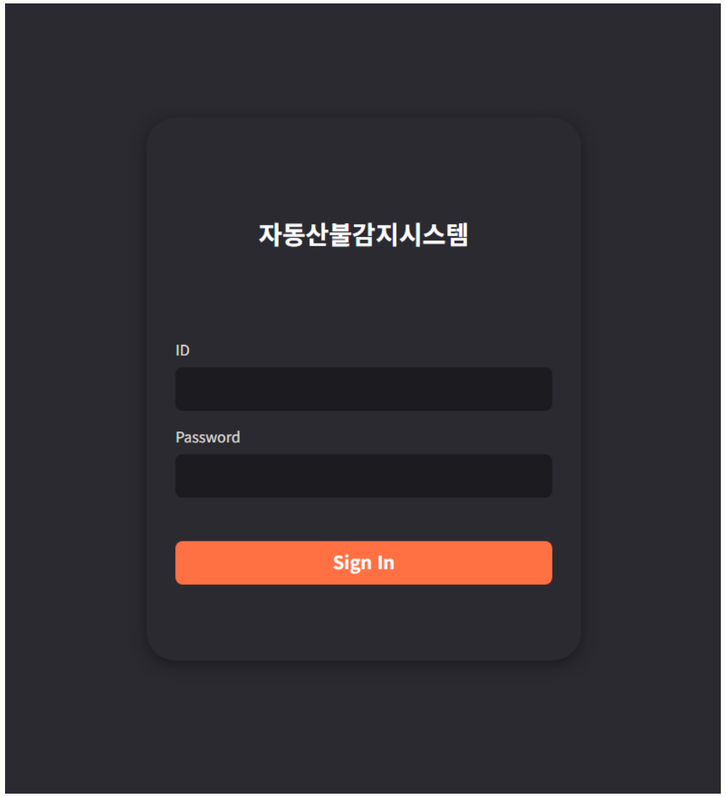
진행 상황

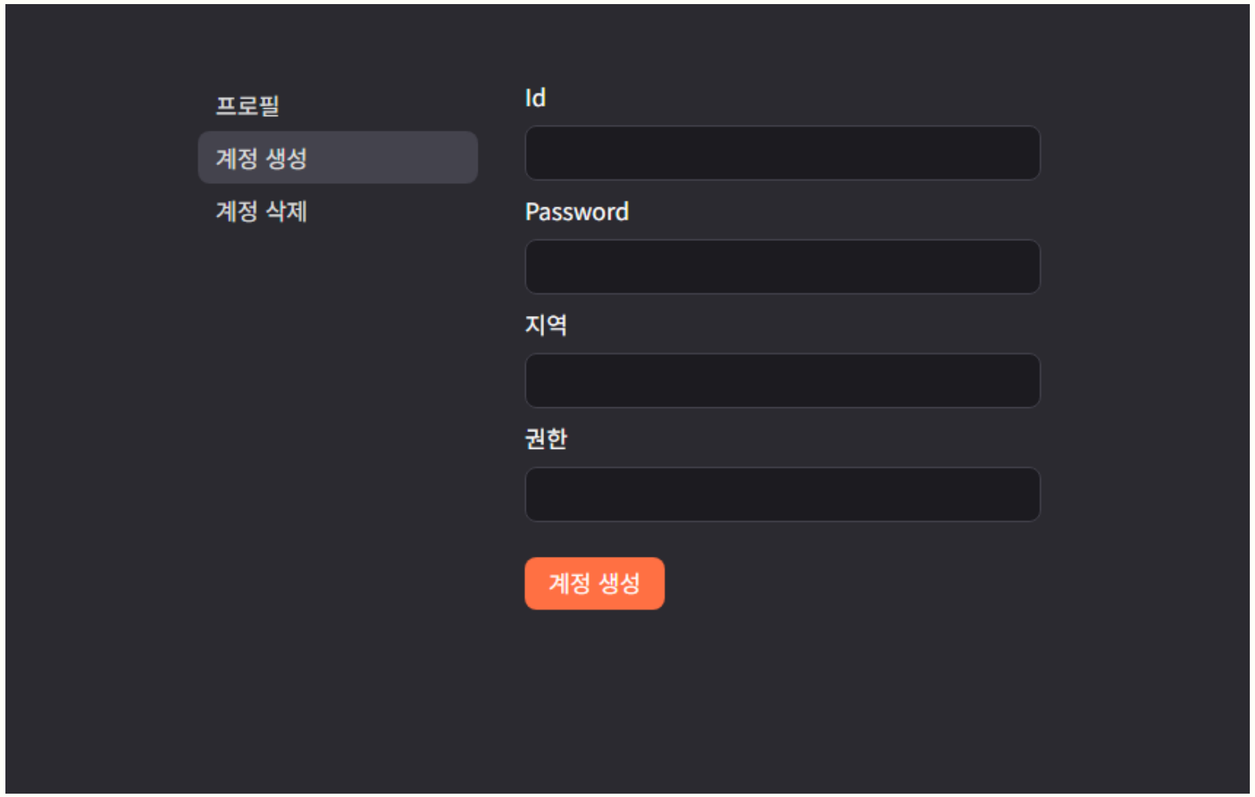
프런트 엔드 - 웹 UI및 레이아웃 구현(vue.js 3)



진행 상황

프런트 엔드 - 페이지 라우팅 및 상호작용 이벤트 일부 구현

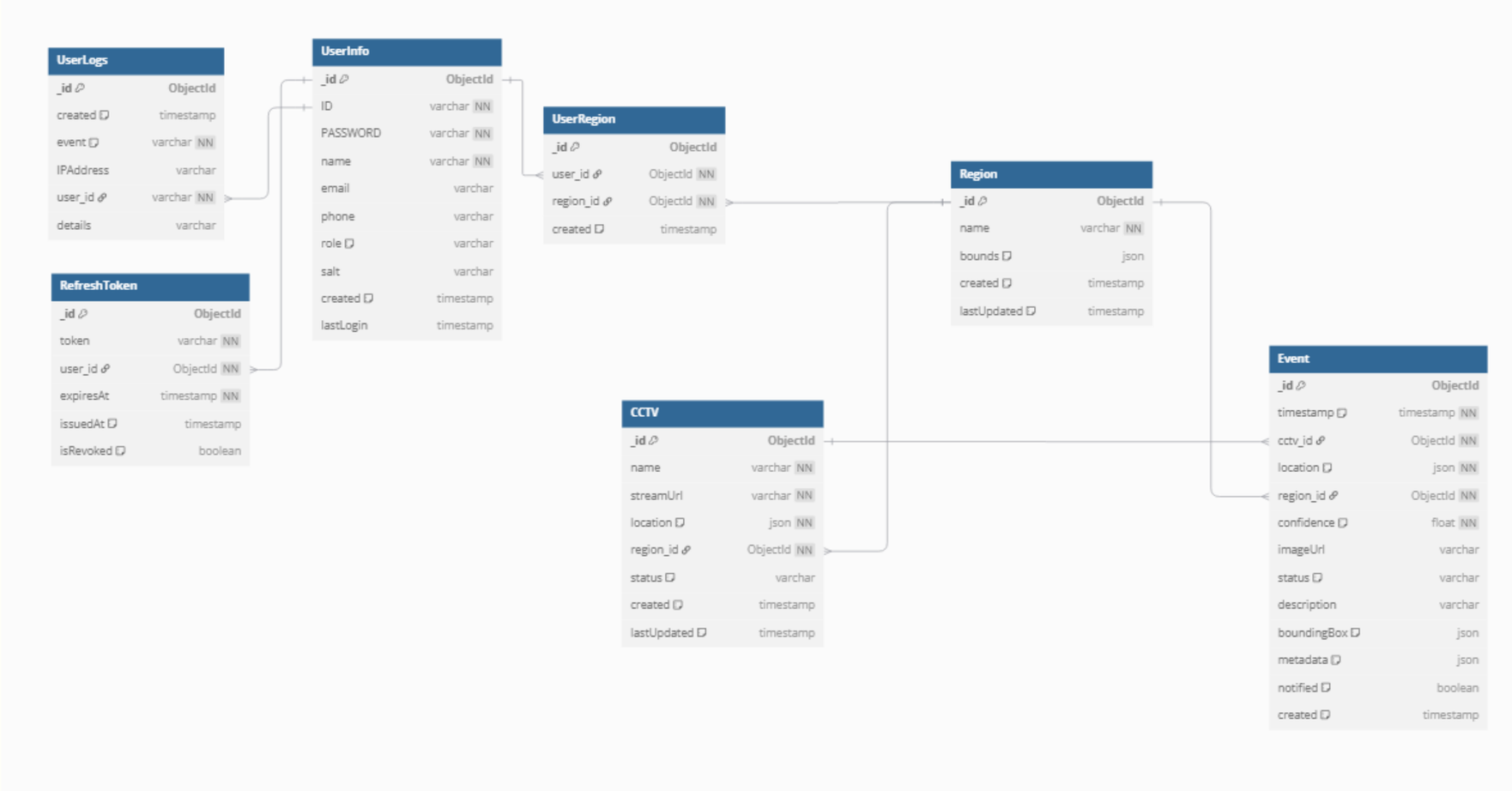




진행 상황

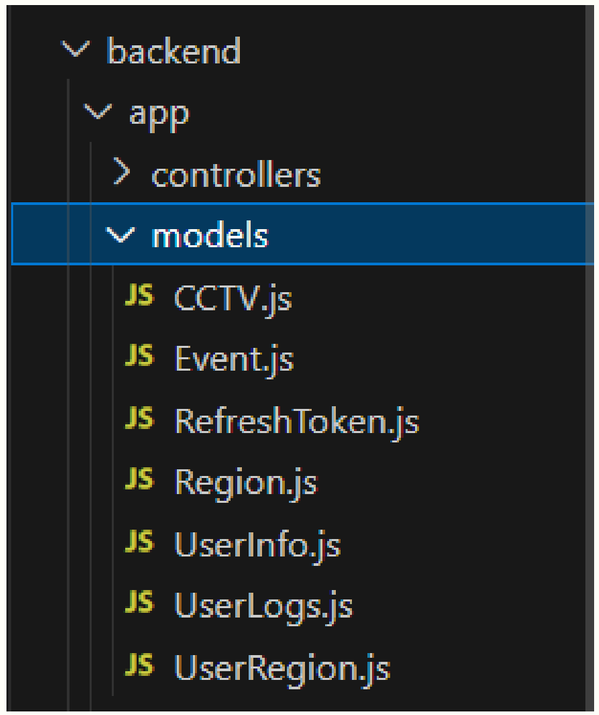
백엔드 - DB 스키마 & 구축

< DB 스키마 >



진행 상황

백엔드 - DB 스키마 & 구축



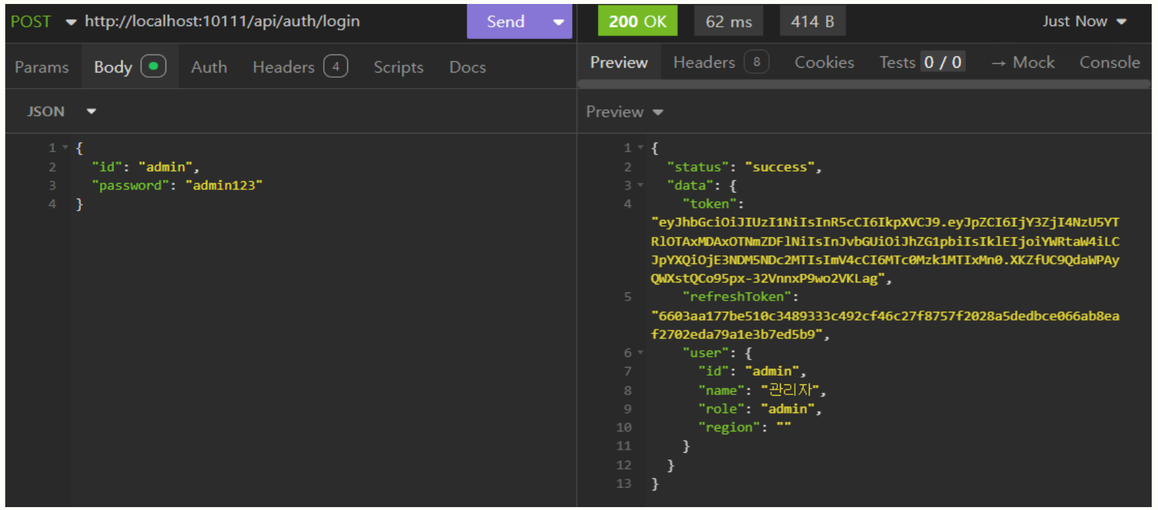
< DB 정보 >

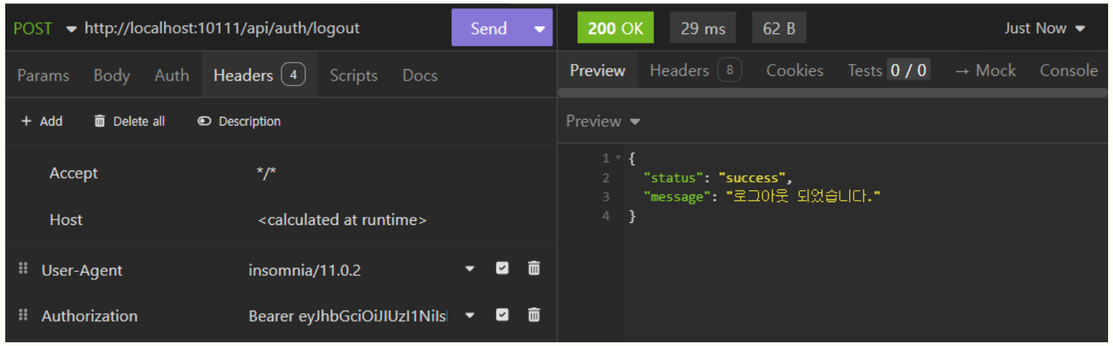
UserInfo	사용자 정보 (아이디, 비밀번호, 이름, 이메일, 전화번호, 역할 등)
UserLogs	사용자 활동 로그 (로그인, 로그아웃 등의 이벤트)
RefreshToken	인증 관련 리프레시 토큰 정보
Region	지역 정보 (이름, 지리적 경계)
UserRegion	사용자-지역 다대다 관계 (한 사용자가 여러 지역 담당 가능)
CCTV	CCTV 정보 (이름, 스트림 URL, 위치, 상태)
Event	이벤트 정보 (시간, 위치, 감지 신뢰도, 이미지 URL, 상태)

진행 상황

백엔드 – 인증 시스템 구현 및 테스트

< Login & Logout >

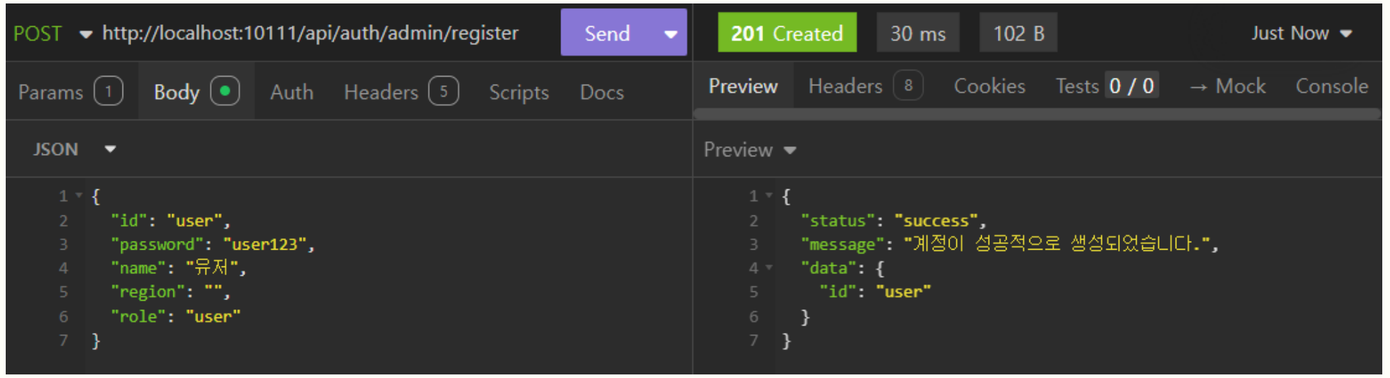


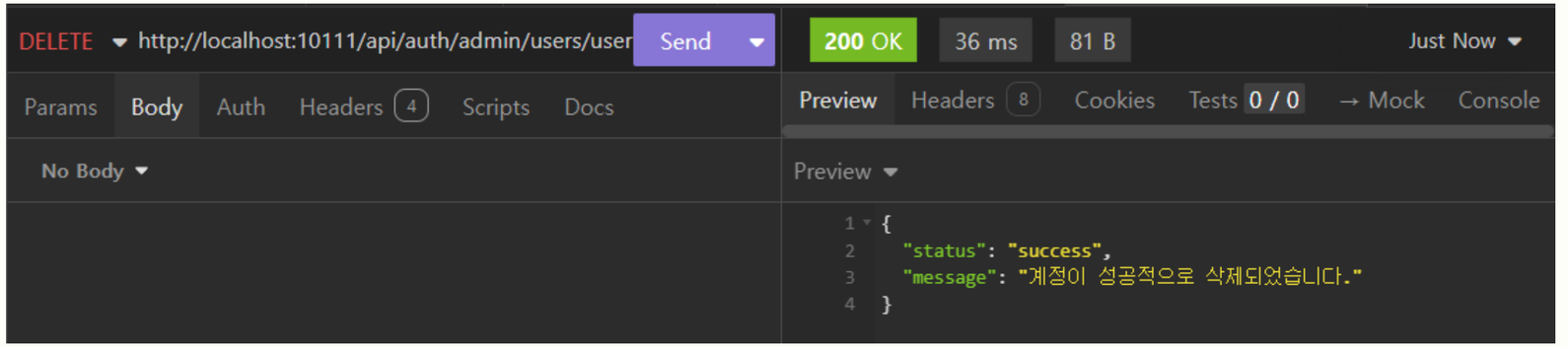


진행 상황

백 엔드 – 인증 시스템 구현 및 테스트

< 계정 생성 & 삭제 >





진행 상황

AI - Sample Data를 통한 학습.

< Json 파일 -> txt 변환. >

데이터 전처리

> 기존 JSON 파일에서 TXT 파일로 변환하는 과정

```
import os
import json

path = "/content/labels/" #json2txt.ipynb 파일과 동일한 경로에 있기 때문에 다음과 같이 선언

file_lists = os.listdir(path) #path 경로 내 모든 파일명 리스트 형태로 불러오기
print(file_lists)
```

```
['NegativeDB_구름_강원도강릉시난곡동_NP020001_001_없음_00003960.json', 'NegativeDB_구름_강원도강릉시난곡동_NP020001_001_없음_00002640.json', 'NegativeDB_구름_강원도강릉시난곡동_NP020001_001_없음_00002580.json', 'NegativeDB_구름_강원도강릉시난곡동_NP020001_0
```

```
[ ] #.json 확장자만 저장하기
    json_file_lists = []
    for file_name in sorted(file_lists):
        if file_name.split(".")[-1] == "json":
            json_file_lists.append(file_name)
```

진행 상황

AI - Sample Data를 통한 학습.

< Yolo11에 맞는 데이터 처리. >

```
[ ]  for j, json_file in enumerate(json_file_lists):
       #print(j,  json_file) #00000005.json 한개에 대해서 json2txt 변환하기 위해 해당 인덱스에 00000005.json 파일 불러오기
       with open(os.path.join(path, json_file),"r") as file:
           json_data = json.load(file) #json 파일 내 데이터 읽어오기

       WIDTH = float(json_data['images'][0]['width'])
       HEIGHT = float(json_data['images'][0]['height'])

       new_file = open(os.path.join(path, json_data['images'][0]['file_name'].split(".")[0]+'.txt'),"w") #"w": 쓰기 형식, "r": 읽기 형식, "a": 이어쓰기 형식
       #print(os.path.join(path, json_data['images'][0]['file_name'].split(".")[0]+'.txt'))
       #CLASS_NAMES = ['흑색연기', '백색/회색연기', '화염', '구름', '안개/연무', '굴뚝연기'] # 파일 내 커스텀 클래스 정보

       #json 데이터 내 'shapes' 정보 읽어오기
       for i, data in enumerate(json_data['annotations']):
           #print("   ", i) #순차적으로 for문이 진행되는지 확인하는 인덱스 번호 출력
           class_id = data['category_id']
           if class_id<0 or class_id>6:
             print(class_id)
           x_left_top = float(data['bbox'][0]) #물체의 바운딩 박스, left top x좌표
           y_left_top = float(data['bbox'][1]) #물체의 바운딩 박스, left top y좌표
           x_right_bottom = float(data['bbox'][2]) #물체의 바운딩 박스, right bottom x좌표
           y_right_bottom = float(data['bbox'][3]) #물체의 바운딩 박스, right bottom y좌표
           width = x_right_bottom - x_left_top #물체의 바운딩 박스 너비
           height = y_right_bottom - y_left_top #물체의 바운딩 박스 높이
           normalize_width = abs(width) / WIDTH #YOLO 학습을 위해 너비 정규화
           normalize_height = abs(height) / HEIGHT #YOLO 학습을 위해 높이 정규화
           ctr_x = x_left_top + width / 2 #물체 바운딩 박스 중심 x 좌표
           normalize_ctr_x = ctr_x / WIDTH #YOLO 학습을 위해 물체 바운딩 박스 중심 x 좌표 정규화
           ctr_y = y_left_top + height / 2 #물체 바운딩 박스 중심 y 좌표
           normalize_ctr_y = ctr_y / HEIGHT #YOLO 학습을 위해 물체 바운딩 박스 중심 y 좌표 정규화
           #텍스트 파일에 YOLO 학습용 데이터 포맷으로 정리
           #모든 값은 string 타입으로 저장해야함
           new_data = "{} {} {} {} {}₩n".format(str(class_id), str(normalize_ctr_x), str(normalize_ctr_y), str(normalize_width), str(normalize_height))

           new_file.write(new_data)

       new_file.close()
```

Yolo 11에 적합하게
Json파일을 읽은 후 변환.

진행 상황

AI - Sample Data를 통한 학습.

< Test 학습. >

```
[ ]  from ultralytics import YOLO

     # Load a COCO-pretrained YOLO11n model
     model = YOLO("yolo11n.pt")

     # Train the model on the COCO8 example dataset for 100 epochs
     results = model.train(data='/content/data.yaml', epochs=10, patience=10, batch=32,imgsz=640)

     # Run inference with the YOLO11n model on the 'bus.jpg' image
     #results = model("path/to/bus.jpg")
```

```
Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.99 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
YOLO11n summary (fused): 100 layers, 2,583,322 parameters, 0 gradients, 6.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  4.91it/s]
                   all         28         64     0.0404      0.484      0.348      0.106
                   fog         28         64     0.0404      0.484      0.348      0.106
Speed: 0.1ms preprocess, 1.8ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to runs/detect/train
```

진행 상황

AI - 회사 서버 환경 구축.

2. 39.121.224.150

```
(base) guest@server:~/team_5/Data$ du -h
591G    ./Training/Original_Data
4.9G    ./Training/Label_Data
596G    ./Training
74G     ./Validation/Original_Data
615M    ./Validation/Label_Data
75G     ./Validation
671G    .
(base) guest@server:~/team_5/Data$ find . -type d
.
./Training
./Training/Original_Data
./Training/Label_Data
./Validation
./Validation/Original_Data
./Validation/Label_Data
(base) guest@server:~/team_5/Data$
```

```
(base) guest@server:~$ nvidia-smi
Mon Apr  7 21:37:10 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.14              Driver Version: 550.54.14      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA A10                    Off  |   00000000:4B:00.0 Off |                    0 |
|  0%   27C    P8             19W /  150W |      13MiB /  23028MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A      1245      G   /usr/lib/xorg/Xorg                             4MiB |
+-----------------------------------------------------------------------------------------+
```

OS : Ubuntu 20.04

DATA : 671G

GPU : NVIDIA A10

이슈 사항 및 해결 방안

1. 샘플 데이터 라벨링 오류.

```
[ ]  from PIL import Image, ImageDraw
     import matplotlib.pyplot as plt

     def draw_bounding_box(image_path, x1, y1, x2, y2, color="red"):
         # 이미지 로드
         image = Image.open(image_path)


         # 그리기 객체 생성
         draw = ImageDraw.Draw(image)

         # 바운딩 박스 그리기
         draw.rectangle([x1, y1, x2, y2], outline=color, width=3)

         # 이미지 출력
         plt.imshow(image)
         plt.axis("off")  # 축 없애기
         plt.show()

     # 사용 예시
     #draw_bounding_box("test_image.jpg", 37, 29, 1322, 443)
     draw_bounding_box("test_image.jpg", 483, 216, 727, 444)
```

이슈사항

- 샘플 데이터 라벨링 오류로 인해서 학습 정확도가 낮게 나옴.

해결방안

- 실제 학습 데이터는 라벨링 정상 확인 완료.

이슈 사항 및 해결 방안

2. 초기에 계정 관리자 가 없는 경우.

```
async function createAdminUser() {
  try {
    // 이미 admin 계정이 있는지 확인
    const existingAdmin = await UserInfo.findOne({ ID: 'admin' });

    if (existingAdmin) {
      console.log('이미 admin 계정이 존재합니다.');
      return;
    } else {
      //관리자 생성
      const adminUser = new UserInfo({
        ID: 'admin',
        PASSWORD: 'admin123',
        name: '관리자',
        role: 'admin'
      });

      await adminUser.save();
      console.log('관리자 계정이 성공적으로 생성되었습니다.');
      console.log('ID: admin');
      console.log('Password: admin123');
    }
  } catch (err) {
    console.error('관리자 계정 생성 중 오류 발생:', err);
  } finally {
    // 연결 종료
    mongoose.connection.close();
    console.log('MongoDB 연결 종료');
  }
}

createAdminUser();
```

이슈사항

- 계정 생성의 경우 관리자 계정을 통해서 가능하지만, 초기 관리자 계정이 없는 문제

해결방안

- DB에 초기 관리자를 삽입하는 스크립트를 통해서 해결.

이슈 사항 및 해결 방안

3. Docker 세팅 및 코드 오류

```
54  UserSchema.pre('save', function(next) {
55      if (this.PASSWORD) {
56          this.salt = crypto.randomBytes(128).toString('base64');
57          this.PASSWORD = this.hashPassword(this.PASSWORD);
58      }
59      next();
60  });
```

```
54  UserSchema.pre('save', function(next) {
55      if (this.isNew || this.isModified('PASSWORD')) {
56          this.salt = crypto.randomBytes(128).toString('base64');
57          this.PASSWORD = this.hashPassword(this.PASSWORD);
58      }
59      next();
60  });
```

이슈사항

- Docker 세팅 오류나 Code 실행 및 테스팅 과정에서 오류발생.

해결방안

- Code 디버깅을 통해서 수정.
- Docker세팅 수정.

향후 일정

1. 프런트 엔드 : 세부 디자인 마무리 & 외부 API 및 백 엔드 연동.

2. 백 엔드 : 관련 API & 실시간 스트리밍 구현.

3. AI : 실제 데이터를 통한 모델 학습.

4. 멘토님과 회의.

5. 논문 관련 논의.

감사합니다