SW상세설계서(SDS)_7팀

1. 서론 (Introduction)

1.1 목적 (Purpose)

Okestro Meeting Coach의 상세 설계 문서로, 회의 내용 분석, 참여도 평가 및 피드백 기능을 구현하기 위한 구조 및 기술적 사양을 정의합니다.

1.2 범위 (Scope)

본 문서는 회의 참여자 음성 인식, 회의 참여도 분석, 회의 내용 요약, 제스처 및 표정 인식 등 기능을 포함한 백엔드 및 프론트엔드 모듈의 설계를 다룹니다.

1.3 참조 문서 (Reference Documents)

- 프로젝트 계획서
- ERD

2. 시스템 개요 (System Overview)

2.1 기능 설명 (Functional Overview)

본 시스템은 회의 중 음성 인식, 발언 분석, 표정 및 제스처 인식을 통해 회의 참여도와 참여태도를 평가하고 회의록을 자동 작성하는 미팅 코칭 시스템입니다. 회의 중 발언 기회를 균등하게 제공하고, 참여자의 발언을 감정 분석하여 회의의 생산성을 높이는 것이 목적입니다.

2.2 시스템 아키텍처 (System Architecture)

2.2.1 프레젠테이션 레이어 (Presentation Layer)

사용자 인터페이스(UI) - 사용자가 회의에 참여할 수 있는 웹사이트를 제공하며, React로 구현합니다.

2.2.2 비즈니스 로직 레이어 (Business Logic Layer)

- 음성 인식 모듈: Google Cloud Speech-to-Text API를 활용해 회의 음성을 실시간으로 텍스트로 변환합니다.
- 발언 및 감정 분석: OpenCV와 MediaPipe를 활용한 표정 인식과, Librosa 라이브러리 및 머신러닝을 활용한 음성 주파수 및 감정 분석을 수행합니다.
- 회의 참여도 평가 및 회의록 요약: Okestro LLM을 통해 발언 빈도, 감정 분석, 주제 집중도를 평가하고 회의록을 요약합니다.

2.2.3 데이터 레이어 (Data Layer)

MySQL DBMS를 활용해 회의 참여자별 음성 인식 텍스트, 회의 요약본, 참여도 및 태도 평가 데이터를 저장합니다.

3. 모듈 설계 (Module Design)

3.1. 미팅 모듈 (Meeting Module)

3.1.1 기능 설명 (Functional Overview)

회의 정보를 관리하는 기능을 제공합니다. 회의 제목, 날짜, 참가자, 비디오 URL, 회의 요약을 저장하고 관리합니다.

3.1.2 입력 (Input)

회의 제목, 날짜, 참가자 정보, 비디오 URL, 회의 요약

3.1.3 출력 (Output)

저장된 회의 정보

3.1.4 프로세스 흐름 (Process Flow)

1. 회의 정보를 입력하여 새로운 회의 데이터를 생성
2. 데이터베이스에 저장된 회의 정보를 조회

3.1.5 주요 파일 및 함수 (Key Files and Functions)

- **models.py**

```
from django.db import models

class Meeting(models.Model):
    title = models.CharField(max_length=255)
    meeting_date = models.DateTimeField()
    participants = models.TextField()
    video_url = models.CharField(max_length=255)
    summary = models.TextField()
```

- **views.py**

```
from django.http import JsonResponse
from .models import Meeting

def create_meeting(request):
    meeting = Meeting.objects.create(
        title=request.POST['title'],
        meeting_date=request.POST['meeting_date'],
        participants=request.POST['participants'],
        video_url=request.POST['video_url'],
```

```
        summary=request.POST['summary']
    )
    return JsonResponse({'meeting_id': meeting.id})

def get_meeting(request, meeting_id):
    meeting = Meeting.objects.get(id=meeting_id)
    return JsonResponse({
        'title': meeting.title,
        'meeting_date': meeting.meeting_date,
        'participants': meeting.participants,
        'video_url': meeting.video_url,
        'summary': meeting.summary
    })
```

3.2. 피드백 모듈 (Feedback Module)

3.2.1 기능 설명 (Functional Overview)

회의 참여자에 대한 피드백 정보를 관리하는 기능을 제공합니다. 모션 인식, 제스처 인식, 음성 주파수 분석 결과와 피드백 내용을 저장하고 관리합니다.

3.2.2 입력 (Input)

모션 인식, 제스처 인식, 음성 주파수 분석 결과, 피드백 내용

3.2.3 출력 (Output)

저장된 피드백 정보

3.2.4 프로세스 흐름 (Process Flow)

1. 미팅 ID와 사용자 ID를 기반으로 피드백을 생성
2. 피드백 데이터를 조회하고 업데이트

3.2.5 주요 파일 및 함수 (Key Files and Functions)

- **models.py**

```
from django.db import models

class Feedback(models.Model):
    meeting = models.ForeignKey('Meeting', on_delete=models.CASCADE)
    user = models.ForeignKey('User', on_delete=models.CASCADE)
    motion_recognition = models.TextField()
    gesture_recognition = models.TextField()
    pitch_frequency_analysis = models.TextField()
    feedback = models.TextField()
```

- **views.py**

```
from django.http import JsonResponse
from .models import Feedback

def create_feedback(request):
    feedback = Feedback.objects.create(
        meeting_id=request.POST['meeting_id'],
        user_id=request.POST['user_id'],
        motion_recognition=request.POST['motion_recognition'],
        gesture_recognition=request.POST['gesture_recognition'],
        pitch_frequency_analysis=request.POST['pitch_frequency_analysi
s'],
        feedback=request.POST['feedback']
    )
    return JsonResponse({'feedback_id': feedback.id})

def get_feedback(request, feedback_id):
    feedback = Feedback.objects.get(id=feedback_id)
    return JsonResponse({
        'motion_recognition': feedback.motion_recognition,
        'gesture_recognition': feedback.gesture_recognition,
        'pitch_frequency_analysis': feedback.pitch_frequency_analysis,
        'feedback': feedback.feedback
    })
```

3.3. 퀴즈 모듈 (Quiz Module)

3.3.1 기능 설명 (Functional Overview)

회의 내용을 기반으로 퀴즈를 생성하고 관리하는 기능을 제공합니다. 각 퀴즈에는 질문, 선택지, 정답이 포함됩니다.

3.3.2 입력 (Input)

질문, 선택지, 정답

3.3.3 출력 (Output)

퀴즈 질문과 정답

3.3.4 프로세스 흐름 (Process Flow)

1. 미팅 ID를 기반으로 퀴즈를 생성
2. 퀴즈 데이터를 조회 및 관리

3.3.5 주요 파일 및 함수 (Key Files and Functions)

- **models.py**

```
from django.db import models
```

```
class Quiz(models.Model):
    meeting = models.ForeignKey('Meeting', on_delete=models.CASCADE)
    question = models.TextField()
    options = models.CharField(max_length=255)
    correct_answer = models.IntegerField()
```

- **views.py**

```
from django.http import JsonResponse
from .models import Quiz

def create_quiz(request):
    quiz = Quiz.objects.create(
        meeting_id=request.POST['meeting_id'],
        question=request.POST['question'],
        options=request.POST['options'],
        correct_answer=request.POST['correct_answer']
    )
    return JsonResponse({'quiz_id': quiz.id})

def get_quiz(request, quiz_id):
    quiz = Quiz.objects.get(id=quiz_id)
    return JsonResponse({
        'question': quiz.question,
        'options': quiz.options,
        'correct_answer': quiz.correct_answer
    })
```

3.4. 사용자 관리 모듈 (User Management Module)

3.4.1 기능 설명 (Functional Overview)

사용자 정보를 관리하는 기능을 제공합니다. 사용자 등록, 로그인, 사용자별 피드백 확인을 처리합니다.

3.4.2 입력 (Input)

사용자 이름, 이메일, 비밀번호

3.4.3 출력 (Output)

사용자 정보, 피드백 데이터

3.4.4 프로세스 흐름 (Process Flow)

1. 사용자 정보를 입력하여 회원가입 처리
2. 로그인 및 피드백 데이터를 조회

3.4.5 주요 파일 및 함수 (Key Files and Functions)

- **models.py**

```
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=255)
```

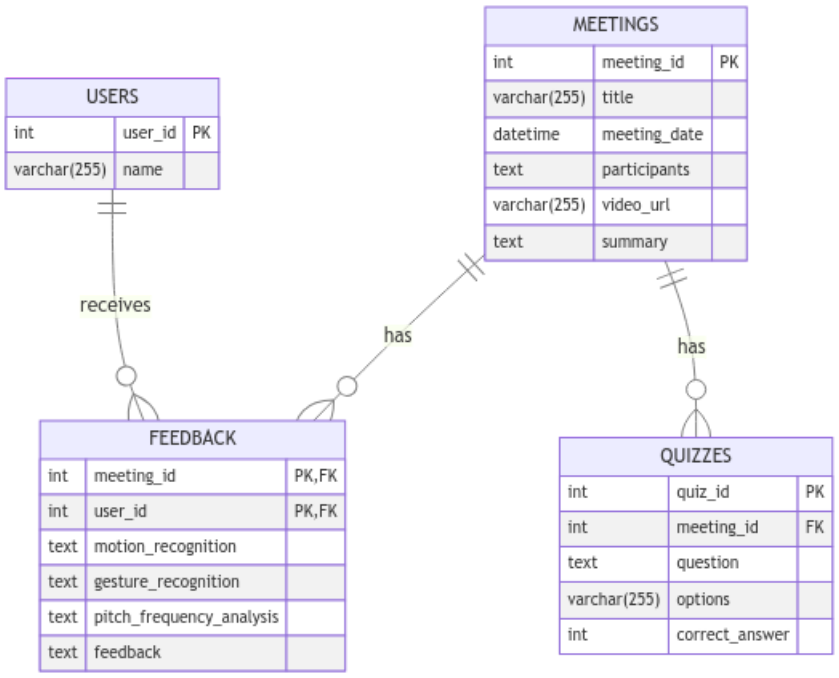
- **views.py**

```
from django.http import JsonResponse
from .models import User

def create_user(request):
    user = User.objects.create(name=request.POST['name'])
    return JsonResponse({'user_id': user.id})

def get_user_feedback(request, user_id):
    feedback = Feedback.objects.filter(user_id=user_id)
    feedback_data = [{'meeting_id': fb.meeting_id, 'feedback': fb.feedba
ck} for fb in feedback]
    return JsonResponse({'feedback': feedback_data})
```

4. 데이터베이스 설계 (Database Design)



4.1 사용자 테이블 (User Table)

- **테이블명:** `users`
- **컬럼:**

- `user_id` (PK, INT) - 사용자 ID (Primary Key)
- `name` (VARCHAR(255)) - 사용자 이름

- **인덱스:** `user_id` 컬럼에 인덱스 설정

4.2 회의 테이블 (Meeting Table)

- **테이블명:** `meetings`
- **컬럼:**
  - `meeting_id` (PK, INT) - 회의 ID (Primary Key)
  - `title` (VARCHAR(255)) - 회의 제목
  - `meeting_date` (DATETIME) - 회의 날짜
  - `participants` (TEXT) - 참가자 정보 (텍스트 형태로 저장)
  - `video_url` (VARCHAR(255)) - 회의 비디오 URL
  - `summary` (TEXT) - 회의 요약
- **인덱스:** `meeting_id` 컬럼에 인덱스 설정

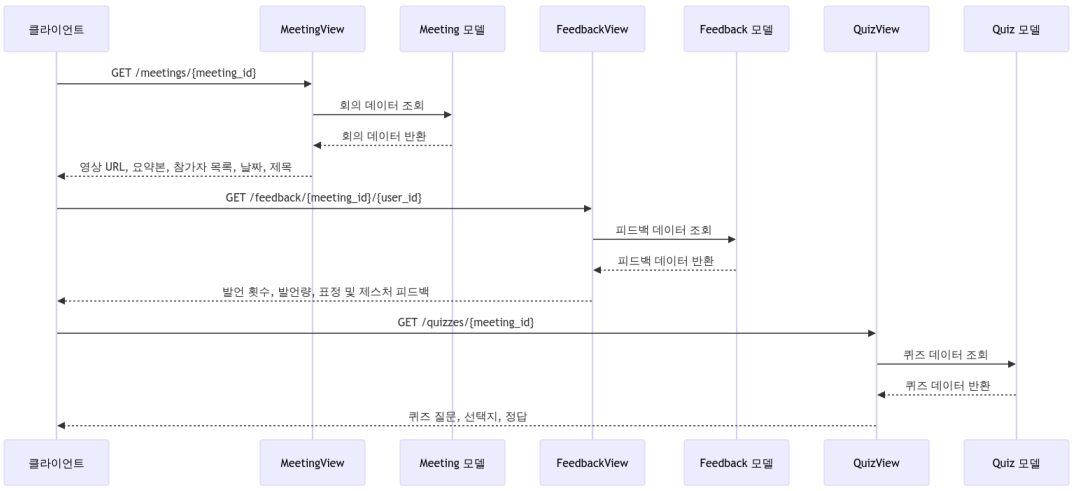
4.3 피드백 테이블 (Feedback Table)

- **테이블명:** `feedback`
- **컬럼:**
  - `meeting_id` (PK, FK, INT) - 회의 ID (Foreign Key)
  - `user_id` (PK, FK, INT) - 사용자 ID (Foreign Key)
  - `motion_recognition` (TEXT) - 모션 인식 결과
  - `gesture_recognition` (TEXT) - 제스처 인식 결과
  - `pitch_frequency_analysis` (TEXT) - 음성 주파수 분석 결과
  - `feedback` (TEXT) - 피드백 내용
- **인덱스:** `meeting_id`, `user_id` 컬럼에 인덱스 설정

4.4 퀴즈 테이블 (Quiz Table)

- **테이블명:** `quizzes`
- **컬럼:**
  - `quiz_id` (PK, INT) - 퀴즈 ID (Primary Key)
  - `meeting_id` (FK, INT) - 회의 ID (Foreign Key)
  - `question` (TEXT) - 퀴즈 질문
  - `options` (VARCHAR(255)) - 선택지 (쉼표로 구분된 선택지 리스트)
  - `correct_answer` (INT) - 정답 번호
- **인덱스:** `quiz_id` 컬럼에 인덱스 설정

5. 시퀀스 다이어그램 (Sequence Diagrams)



5.1 회의 데이터 요청 (Meeting Data Request)

1. 클라이언트 → 서버: `/meetings/{meeting_id}` GET 요청
2. **MeetingView**: 요청 수신 후 회의 데이터를 조회
3. **Meeting 모델**: 회의 ID를 기반으로 데이터베이스에서 조회
4. 서버 → 클라이언트: 영상 URL, 요약본, 참가자 목록, 날짜, 제목 반환

5.2 사용자 피드백 요청 (User Feedback Request)

1. 클라이언트 → 서버: `/feedback/{meeting_id}/{user_id}` GET 요청
2. **FeedbackView**: 요청 수신 후 피드백 데이터를 조회
3. **Feedback 모델**: 사용자 ID와 회의 ID를 기반으로 데이터베이스에서 피드백 조회
4. 서버 → 클라이언트: 발언 횟수, 발언량, 표정 및 제스처 피드백 반환

5.3 퀴즈 정보 요청 (Quiz Information Request)

1. 클라이언트 → 서버: `/quizzes/{meeting_id}` GET 요청
2. **QuizView**: 요청 수신 후 퀴즈 데이터를 조회
3. **Quiz 모델**: 회의 ID를 기반으로 데이터베이스에서 퀴즈 조회
4. 서버 → 클라이언트: 퀴즈 질문, 선택지, 정답 반환

6. API 설계 (API Design)

6.1 회의 데이터 요청 API

- **URL:** `/api/meetings/{meeting_id}`
- **Method:** GET

- **Request Parameters:**
  - `meeting_id` (URL parameter, INT): 요청할 회의의 ID
- **Response:**
  - **성공 시 (200 OK):**

```
{
  "meeting_id": 1,
  "title": "회의 제목",
  "meeting_date": "2024-10-15T10:30:00Z",
  "participants": "홍길동, 이순신",
  "video_url": "https://example.com/video",
  "summary": "회의 내용 요약"
}
```

  - **에러 처리:**
    - **404 Not Found:** 요청한 회의 ID가 존재하지 않는 경우

```
{
  "error": "Meeting not found"
}
```

6.2 사용자 피드백 요청 API

- **URL:** `/api/feedback/{meeting_id}/{user_id}`
- **Method:** GET
- **Request Parameters:**
  - `meeting_id` (URL parameter, INT): 요청할 회의의 ID
  - `user_id` (URL parameter, INT): 피드백을 요청할 사용자의 ID
- **Response:**
  - **성공 시 (200 OK):**

```
{
  "meeting_id": 1,
  "user_id": 2,
  "motion_recognition": "손동작: 나쁨",
  "gesture_recognition": "몸짓: 좋음",
  "pitch_frequency_analysis": "주파수 분석: 안정적",
  "feedback": "사용자의 참여 태도가 우수했습니다."
}
```

  - **에러 처리:**
    - **404 Not Found:** 해당 회의 또는 사용자에 대한 피드백이 없는 경우

```
{
  "error": "Feedback not found"
}
```

6.3 퀴즈 정보 요청 API

- **URL:** `/api/quizzes/{meeting_id}`
- **Method:** GET
- **Request Parameters:**
  - `meeting_id` (URL parameter, INT): 퀴즈 정보를 요청할 회의의 ID
- **Response:**
  - **성공 시 (200 OK):**

```
{
  "quiz_id": 1,
  "meeting_id": 1,
  "question": "이 회의에서 논의된 핵심 주제는 무엇인가요?",
  "options": ["A", "B", "C", "D"],
  "correct_answer": 2
}
```

  - **에러 처리:**
    - **404 Not Found:** 해당 회의에 대한 퀴즈가 없는 경우

```
{
  "error": "Quiz not found"
}
```

7. 에러 처리 및 로깅 (Error Handling and Logging)

7.1 에러 처리 방안 (Error Handling)

- **전역 에러 처리**: Django의 `Middleware`를 사용하여 전역적으로 에러를 처리하고, 표준화된 JSON 형식의 에러 응답을 반환합니다.
- **사용자 정의 예외 클래스**: 비즈니스 로직에 특화된 사용자 정의 예외 클래스를 생성하여 특정 예외 상황을 처리합니다.
- **에러 응답 형식 표준화**: HTTP 상태 코드, 에러 메시지, 에러 코드를 포함한 일관된 형식으로 에러 응답을 제공합니다.
- **유효성 검사 에러 처리**: 입력 값의 유효성 검사 실패 시 400 Bad Request 응답을 반환하며, 유효성 검사 오류 메시지를 포함합니다.

7.2 로깅 시스템 (Logging)

- **로깅 시스템 설정**: Django의 `logging` 모듈을 사용하여 애플리케이션의 에러 및 주요 이벤트를 로그 파일로 기록합니다.
- **로그 포맷 정의**: 타임스탬프, 로그 레벨, 에러 발생 위치, 에러 메시지 등을 포함하여 문제 해결에 필요한 정보를 기록합니다.
- **로그 파일 관리**: 로그 파일은 일별로 생성하며, 로테이션 설정을 통해 용량 또는 시간이 초과되면 새로운 로그 파일을 생성하여 관리합니다.
- **주요 이벤트 로깅**: 회의 데이터 조회, 피드백 처리, 퀴즈 제공과 같은 주요 비즈니스 이벤트 발생 시 정보 로그를 기록하여 트랜잭션 흐름을 추적합니다.

8. 보안 설계 (Security Design)

- **인증 및 권한 관리**:

  JWT (JSON Web Token) 토큰을 사용하여 사용자 인증 및 권한 관리를 구현합니다.

  사용자가 로그인하면 서버는 JWT 토큰을 발급하며, 이후 모든 요청에서 이 토큰을 이용해 사용자를 인증하고, 필요한 경우 권한을 확인합니다. 관리자나 특정 역할을 가진 사용자만 접근할 수 있는 API 엔드포인트는 JWT를 검증하여 권한을 부여합니다.

- **비밀번호 암호화**:

  사용자의 비밀번호는 **bcrypt** 알고리즘을 사용하여 해시하여 저장합니다. 이를 통해 비밀번호를 안전하게 저장하고, 인증 시 입력된 비밀번호와 해시된 비밀번호를 비교하여 인증합니다.

9. 테스트 계획 (Test Plan)

- **유닛 테스트 (Unit Test)**:

  Django의 **TestCase** 클래스를 활용하여 각 모델과 뷰에 대한 단위 테스트를 작성합니다.

  주요 기능 (회의 데이터 조회, 피드백 조회, 퀴즈 제공 등)에 대한 유닛 테스트를 구현하여 개별 기능이 정확하게 동작하는지 확인합니다.

- **통합 테스트 (Integration Test)**:

  각 API 엔드포인트에 대한 통합 테스트를 구현합니다. 예를 들어, `/api/meetings/{meeting_id}`, `/api/feedback/{meeting_id}/{user_id}`, `/api/quizzes/{meeting_id}`와 같은 API 요청에 대해 통합 테스트를 수행하여, 전체 서비스가 유기적으로 동작하는지 확인합니다.

- **성능 테스트 (Performance Test)**:

  대규모 동시 사용자 접속 시 시스템의 성능을 테스트합니다. 특히 회의 데이터 조회, 피드백 제공, 퀴즈 조회 시 발생할 수 있는 트래픽 부하를 확인하고, 시스템이 적절히 응답하는지 확인합니다.

  성능 테스트 도구를 사용해 서버의 성능을 모니터링하고 병목 현상을 찾아 최적화합니다.

10. 배포 계획 (Deployment Plan)

- **배포 환경**:

  AWS EC2 인스턴스를 사용하여 Django 애플리케이션을 배포합니다.