2025 오픈소스프로그래밍 프로젝트 최종보고서

프로젝트명	어르신을 위한 주민센터 음성인식 AI 키오스크			
팀명	voice4seniors	팀장	학번	이름
			2015869	이예림
수행기간	2025. 05. 15 ~ 2025. 06. 22 (학기말까지)	팀원	2016257	김은서
			2015012	이소영

가. 목표 및 기대효과

- 기술개발 배경

  - 디지털 기기의 빠른 발전에도 불구하고 고령층은 키오스크, 무인 단말기 등 최신 기술에 대한 접근성이 낮아 공공 서비스 이용에 큰 어려움을 겪고 있음. 특히 병원, 관공서, 무인 편의시설에서의 키오스크 사용은 글자 크기, 조작 난이도, 언어 이해도 등에서 진입 장벽이 큼.
  - 이에 따라 본 프로젝트는 음성인식 기반의 키오스크 서비스를 개발하여 고령자의 정보 접근성과 서비스 이용 편의성을 높이고자 함

- 기술개발 목표

  - 음성 명령을 통한 키오스크 서비스 조작 기능 구현 (터치 없이 이용 가능)
  - 딥러닝 기반 한국어 음성 인식 및 질의 응답 처리 (STT + NLP)
  - 간단한 TTS(Text-to-Speech) 응답 기능 포함
  - 사용자의 발화를 분석하여 **문맥과 의도(Intent)**를 파악하고 적절한 UI로 안내
  - 실제 서비스 시나리오(병원, 민원센터 등)를 반영한 예제 구성

나. 프로젝트 개발 내용

1. 음성 인식 모듈 (whisper_infer.py)

```
# Whisper 모델 최적화 설정
model = whisper.load_model("tiny")  # 경량화 모델 선택
result = model.transcribe("kiosk_input.wav", language="ko")
```

- 모델 선택: 실시간 처리를 위한 'tiny' 모델 채택
- 언어 설정: 한국어 특화 음성 인식 정확도 향상
- 성능 최적화: CPU 환경에서도 2-3초 내 처리 가능

2. 의도 분류 시스템 (intent_predictor.py)

```
# TF-IDF + SVM 파이프라인 구현
vectorizer = TfidfVectorizer(max_features=1000, ngram_range=(1,2))
classifier = SVC(kernel='linear', probability=True)
```

- 특징 추출: TF-IDF 벡터화로 텍스트 수치화
- 분류 알고리즘: SVM 선형 커널로 5가지 의도 분류
- 신뢰도 산출: 확률 기반 예측 신뢰도 제공

3. 음성 데이터 처리 (voice_saver.py)

```
# 실시간 음성 녹음 및 저장
def record_audio(filename="kiosk_input.wav", duration=5, samplerate=16000):
    audio = sd.rec(int(duration * samplerate), samplerate=samplerate, channels=
1)
    sd.wait()
    sf.write(filename, audio, samplerate
```

- 샘플링 설정: 16kHz 모노 채널로 Whisper 최적화
- 실시간 처리: 5초 녹음 후 즉시 분석 파이프라인

1. **메인 API 서버 (main.py)**

Python 복사

```
@app.post("/voice-to-intent", response_model=VoiceResponse)
async def voice_to_intent(audio: UploadFile = File(...)):
    # 음성 파일 → 텍스트 → 의도 분류 파이프라인
    with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as temp_file:
        content = await audio.read()
        temp_file.write(content)

    result = whisper_model.transcribe(temp_file_path)
    predicted_intent, confidence = predict_intent_with_confidence(result["tex
t"])
```

- 비동기 처리: async/await를 활용한 비동기 처리
- 파일 업로드: 멀티파트 폼 데이터로 음성 파일 수신
- 임시 파일 관리: 안전한 파일 처리 및 자동 정리

2. **환경 검증 시스템 (check_environment.py)**

```
def check_packages():
    required_packages = [
        ('fastapi', 'FastAPI 웹 프레임워크'),
        ('whisper', 'OpenAI Whisper 음성인식'),
        ('sklearn', 'Scikit-learn 머신러닝'),
    ]
    for package, description in required_packages:
        try:
            __import__(package)
            print(f"✅ {package:12} - {description}")
        except ImportError:
            print(f"❌ {package:12} - {description} (설치 필요)")
```

- 의존성 검증: 필수 패키지 설치 상태 자동 확인
- 모델 파일 검증: AI 모델 파일 존재 및 크기 확인
- 포트 및 권한 확인: 서버 실행 환경 사전 검증

3. AI/프론트엔드 시스템 전체 통합

```python
@app.on_event("startup")
async def startup_event():
    """서버 시작시 AI 모델들 로드 및 시스템 초기화"""
    global whisper_model, intent_classifier, vectorizer

    whisper_model = whisper.load_model("tiny")
    intent_classifier = joblib.load(f"{ai_module_path}/intent_model.pkl")
    vectorizer = joblib.load(f"{ai_module_path}/vectorizer.pkl")

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

- 전체 시스템 아키텍처 설계: AI 모듈, 백엔드 서버, 프론트엔드 간의 데이터 흐름 설계
- API 명세서 작성: 프론트엔드 개발자가 참고할 수 있는 상세한 API 문서화
- 에러 핸들링 체계: 음성 인식 실패, 모델 로드 오류 등 전체 시스템 에러 처리
- 성능 최적화: 음성 파일 처리 속도 향상을 위한 비동기 처리 및 메모리 관리
- 개발환경 구축: 팀원들이 쉽게 개발할 수 있는 환경 설정 및 문서화

1. 메인 키오스크 컴포넌트 (App.jsx)

```javascript
const handleMicClick = async () => {
  if (!audioSupported) {
    const errorMsg = '이 브라우저에서는 음성 입력이 지원되지 않습니다.';
    alert(`🎤 ${errorMsg}`);
    return;
  }

  const audioBlob = await audioRecorderRef.current.startRecording(5);
  const result = await kioskAPI.uploadVoice(audioBlob);

  if (result.success) {
    setRecognizedText(result.transcribed_text);
    await handleIntentResult(result);
  }
};
```

- 상태 관리: useState/useRef를 활용한 복잡한 키오스크 상태 관리
- 음성 처리 플로우: 녹음 → 업로드 → 결과 처리의 완전한 파이프라인
- 에러 처리: 브라우저 호환성 및 권한 문제 대응

2. 음성 녹음 유틸리티 (audioRecorder.js)

```
class AudioRecorder {
  async initialize() {
    this.stream = await navigator.mediaDevices.getUserMedia({
      audio: {
        echoCancellation: true,
        noiseSuppression: true,
        autoGainControl: true,
        sampleRate: 16000,
      }
    });

    this.mediaRecorder = new MediaRecorder(this.stream, {
      mimeType: this.getSupportedMimeType()
    });
  }
}
```

- 음성 품질 최적화: 에코 제거, 노이즈 억제 설정
- 브라우저 호환성: 다양한 오디오 코덱 지원
- 리소스 관리: 마이크 스트림 자동 해제

3. 음성 출력 시스템 (textToSpeech.js)

```
class TextToSpeech {
  speak(text, options = {}) {
    return new Promise((resolve, reject) => {
      const utterance = new SpeechSynthesisUtterance(this.cleanText(text));
      utterance.lang = 'ko-KR';
      utterance.rate = 0.9;  // 어르신을 위한 느린 속도
      utterance.pitch = 1.1; // 명확한 음성

      this.synthesis.speak(utterance);
    });
  }
}
```

- 한국어 TTS: 브라우저 내장 음성 합성 API 활용
- 음성 최적화: 어르신에게 적합한 속도와 음조 설정
- 텍스트 정제: HTML 태그 제거 및 특수문자 처리

다. 추진일정 및 업무분장

1) 역할 분담 계획

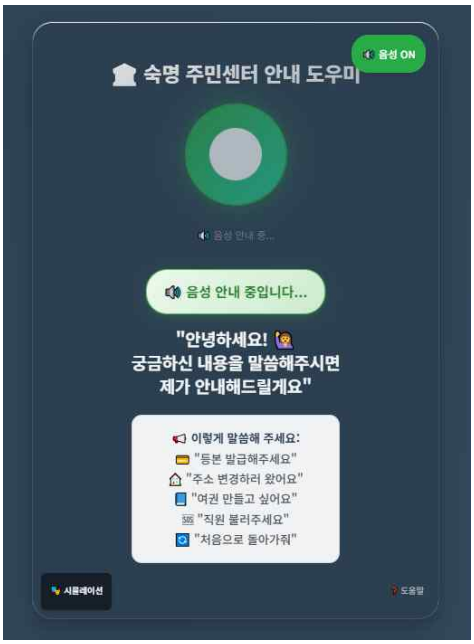
이소영 : 음성 인식 및 의도 분류
주요 담당 업무
- 음성 인식 시스템 구축: OpenAI Whisper 모델 최적화
- 의도 분류 모델 개발: 사용자 발화의 민원 의도 자동 분류
- 데이터 전처리 및 모델 훈련: 한국어 민원 데이터셋 구축 및 학습

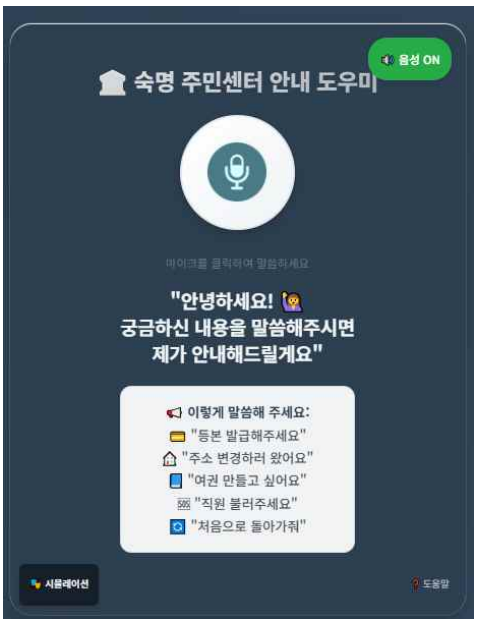
이예림 : API 설계, 서버 구축, 시스템 통합, 문서 작성
주요 담당 업무
- FastAPI 서버 구축: RESTful API 설계 및 구현
- AI 모델 통합: 음성 인식과 의도 분류 파이프라인 연결
- 시스템 안정성: 에러 처리, 로깅, 환경 검증 시스템 구축

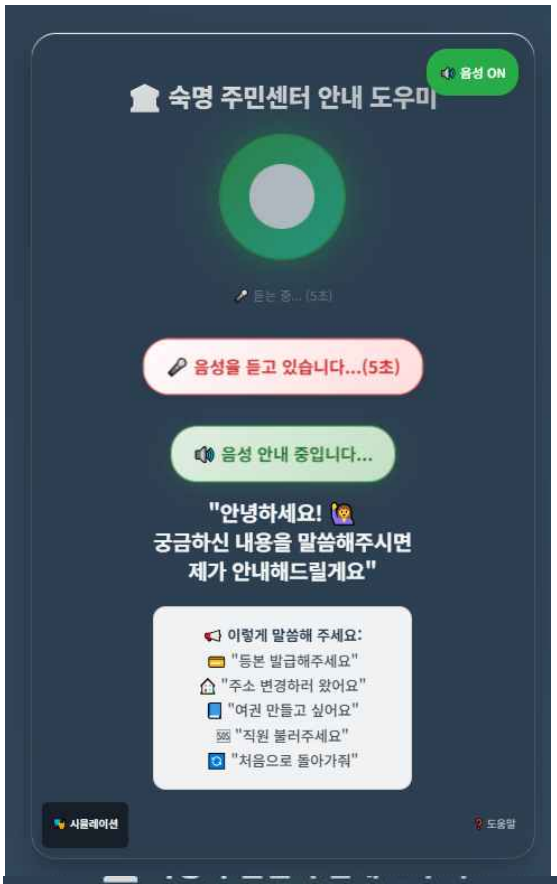
김은서 : 사용자 인터페이스 및 음성 처리
주요 담당 업무
- React 기반 키오스크 UI: 어르신 친화적 인터페이스 설계
- 음성 입출력 처리: 브라우저 음성 API 통합 및 최적화
- 사용자 경험 향상: 직관적 네비게이션 및 접근성 구현

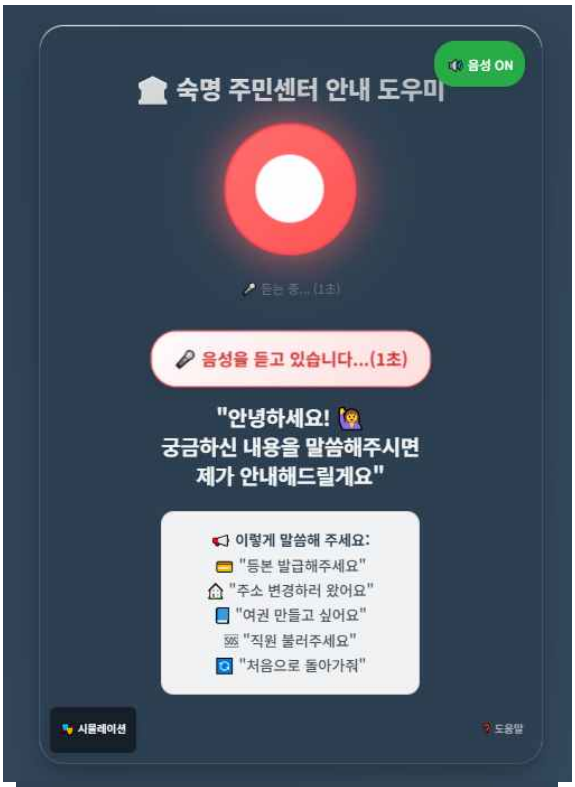
다. 프로젝트 개발 결과

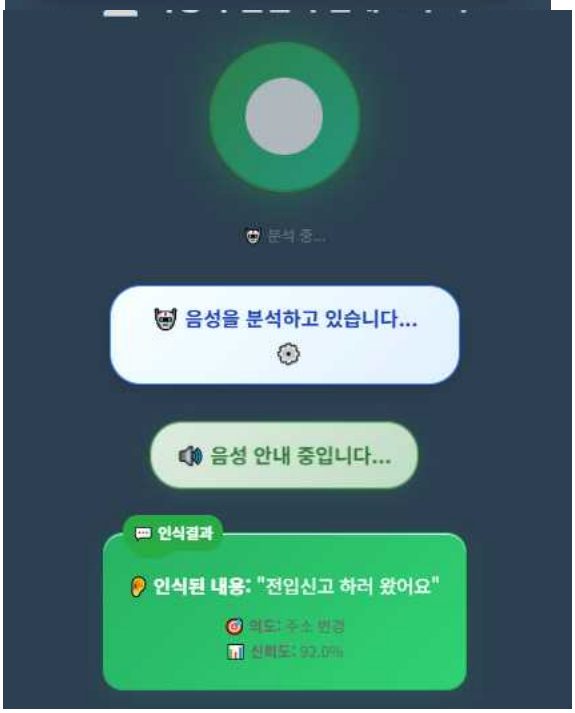
- 최종결과물 내용
  - 결과물 예시

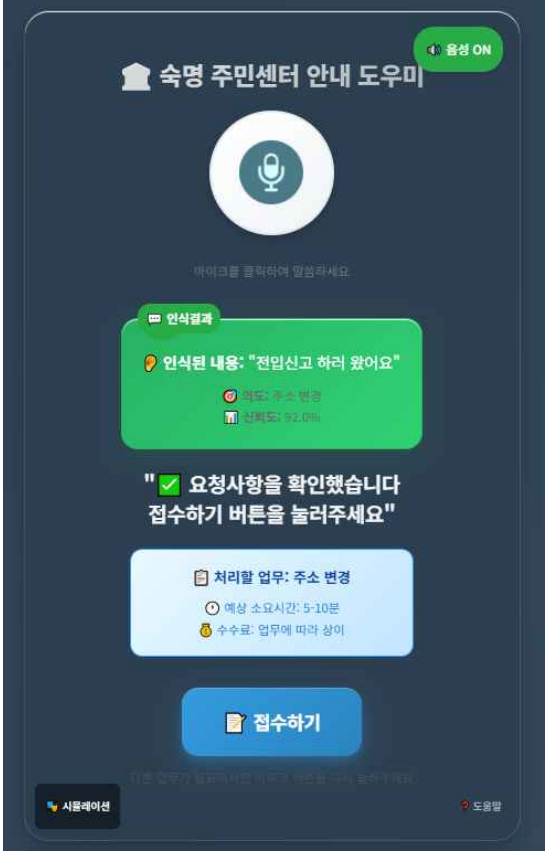












○ Github 사이트

- https://github.com/voice4seniors

- 프로젝트 수행 결과의 한계점
  ○ 개발 기술의 한계점

1. AI 모델 성능 한계

- 음성 인식 한계

- 배경 소음 민감성: 키오스크가 설치될 주민센터 환경의 다양한 소음(대화, 발걸음, 안내 방송 등)에 대한 강건성 부족
- 방언 및 발음 다양성: 지역별 방언이나 개인별 발음 차이에 대한 인식 정확도 저하
- 연령대별 음성 특성: 어르신들의 낮은 음성, 불명확한 발음에 대한 최적화 부족

- 의도 분류 한계

- 훈련 데이터 부족: 실제 주민센터 민원 데이터의 한계로 인한 일반화 성능 제약
- 복합 의도 처리: "등본도 뽑고 주소변경도 하고 싶어요"와 같은 다중 의도 처리 불가
- 맥락 이해 부족: 이전 대화 맥락을 고려하지 못하는 단발성 분류

2. 기술적 제약사항

- 실시간 처리 한계

현재 처리 시간: 평균 4초
음성 녹음(5초) → Whisper 처리(2-3초) → 의도 분류(0.5초) → 응답 생성(0.5초)
목표: 2초 이내 처리 (현실적으로 어려움)

- 메모리 및 성능 제약

- Whisper 모델 크기: 'tiny' 모델 사용으로 인한 정확도와 성능의 트레이드오프
- 모바일 브라우저 한계: 일부 모바일 환경에서의 음성 API 지원 불완전

  ○ 프로젝트 수행의 어려움 및 느낀 점

- 기술적 어려움

도전 과제:

- Whisper 모델의 한국어 방언 대응 부족
- 제한된 민원 데이터로 인한 의도 분류 성능 한계
- 실시간 처리와 정확도 간의 균형점 찾기

해결 과정:

- 다양한 Whisper 모델 크기별 성능 테스트 실시
- 키워드 기반 데모 모드로 모델 없이도 동작하는 폴백 시스템 구현
- 신뢰도 기반 재확인 시스템으로 오인식 대응

느낀 점

- 실제 서비스 환경에서는 깨끗한 실험 환경과 달리 예상치 못한 변수들이 많다는 것을 깨달았습니다. 특히 어르신들의 다양한 발화 패턴을 모두 커버하기 위해서는 훨씬 더 많은 데이터와 모델 튜닝이 필요함을 느꼈습니다.

개선 방향

- 실제 주민센터 환경에서의 데이터 수집 및 모델 재훈련

- 온디바이스 모델 최적화를 통한 응답 속도 향상
- 대화형 AI로 발전하여 맥락을 이해하는 시스템 구축

향후 개선 계획

1. 음성 인식 정확도 향상
  - 지역별 방언 데이터 수집 및 파인튜닝
  - 배경 소음 필터링 알고리즘 개선

2. 사용자 경험 개선
  - 음성 안내 메시지 다국어 지원
  - 터치스크린 대체 옵션 제공

기술적 성과
- AI 기술을 실제 서비스에 적용
- 음성 인터페이스를 통한 디지털 격차 해소
- 프론트엔드, 백엔드, AI 모델의 완전한 통합

사회적 가치
- 어르신들의 공공서비스 접근성 향상
- 민원 담당 직원의 업무 부담 경감
- 복잡한 기술을 누구나 쉽게 사용할 수 있도록 구현

본 프로젝트는 **어르신을 위한 음성인식 키오스크**라는 사회적 가치가 높은 목표를 설정하고, 이를 실현하기 위해 최신 AI 기술과 웹 기술을 통합한 의미 있는 결과물을 만들어냈습니다.

기술적 한계와 개선점이 존재하지만, 이는 향후 발전 방향을 명확히 제시해주는 값진 경험이었습니다. 특히 **사용자 중심의 기술 개발**이 얼마나 중요한지를 깨달았으며, AI 기술의 실용화를 위해서는 단순한 성능 향상을 넘어 실제 사용 환경과 사용자 특성을 깊이 이해해야 한다는 교훈을 얻었습니다.

앞으로 이 프로젝트가 디지털 접근성 향상과 포용적 기술 발전의 시초가 되기를 기대하며, 지속적인 개선을 통해 더 많은 사람들이 혜택을 받을 수 있는 서비스로 발전시켜 나가겠습니다.