2025년 새싹 해커톤(SeSAC Hackathon) AI 서비스 기획서

팀명	200OK
팀 구성원 성명	soyeondubu, kkdo, kongsemin, tyson123

1 AI 서비스 명칭

ㅇ 수어·표정·제스처를 포함한 실시간 AI 기반 비언어 커뮤니케이션 플랫폼 - CueCode

2 활용 인공지능 학습용 데이터

	활용 데이터명	분야	출처
1	위급상황 음성/음향 (고도화) - 119 지능형 신고접수 음성 인식 데이터	재난안전환경	AI Hub
2			

3 핵심내용

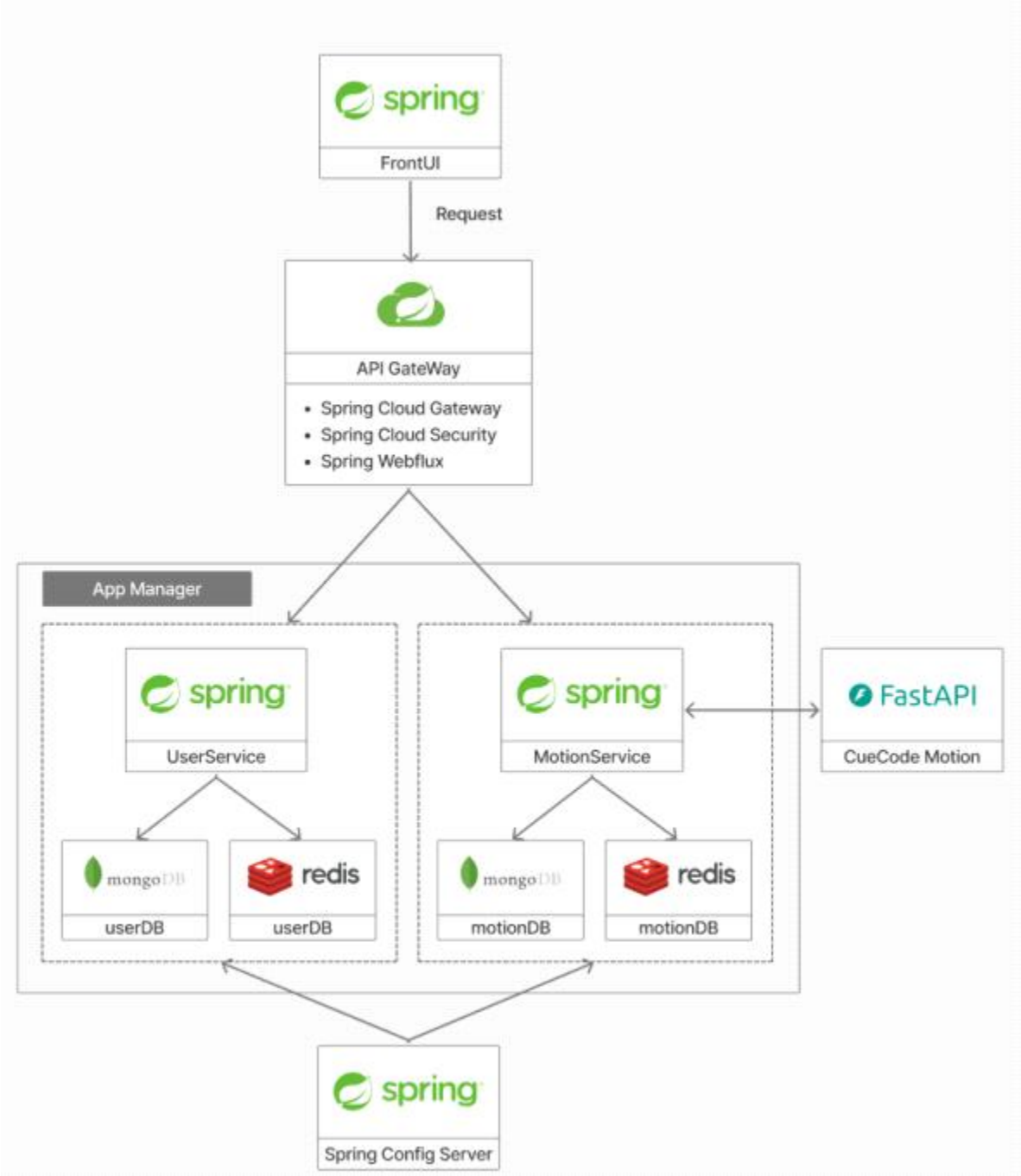
ㅇ 작품 소개
- **CueCode는 얼굴 표정과 손 동작을 AI가 실시간 인식하여 문장 또는 음성으로 변환하는 비언어 커뮤니케이션 플랫폼임**
- 사용자가 직접 동작과 의미를 등록할 수 있어 개인 맞춤형 소통 환경 구성이 가능함
- **웹캠만으로 이용할 수 있어 별도 장비 없이 누구나 쉽게 접근**할 수 있음

ㅇ 주요 기능
- MediaPipe와 DTW 기반 동작 분석 기술로 **얼굴·손 제스처를 실시간 인식함**
- 사용자 정의 제스처-문장 매핑을 통해 **개인 맞춤형 소통 사전 구축**이 가능함
- WebSocket 기반 구조로 인식된 내용을 텍스트·음성으로 즉시 출력함
- **LangGraph 파이프라인을 활용해 인식된 단어를 문맥 기반 완성 문장**으로 자동 생성함
- 응급 데이터를 분석해 위험 문장을 감지하고 관리자 화면에 실시간 알림을 제공함
- **MSA 구조를 적용하여 확장성과 유지보수성을 확보**하고 경량 웹 환경에서 동작 가능함

ㅇ 활용 가능성 및 기대효과
- 청각, 언어, 뇌병변, 고령 장애인 등 **의사소통 취약계층의 실질적 커뮤니케이션 자율성을 높임**
- 돌봄 인력 부족 문제를 보완하고 의료·요양·교육 현장에서 즉각적 의사소통을 지원하는 도구로 활용 가능함

ㅇ 개발 환경

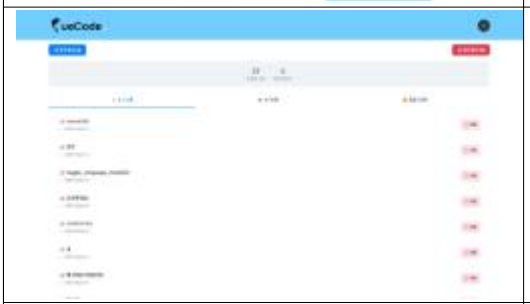
- [그림 1]은 CueCode Inner Architecture을 표현함



[그림 1] CueCode Inner Architecture

- Spring Boot와 FastAPI 기반의 마이크로서비스 구조로 서비스 간 독립 개발 및 유지보수가 용이함
- Kubernetes 및 Docker 기반 컨테이너 환경에서 배포되어 자동 확장 및 롤백이 가능함
  - 구축한 CI/CD 파이프라인을 통해 빌드·배포 자동화 및 운영 효율성이 확보됨
- 트래픽 변화에 따라 서비스 단위로 스케일 조정이 가능해 고가용성과 안정성을 보장함
- API Gateway와 실시간 메시징 구조를 적용하여 사용자 요청을 고속·저지연으로 처리함

ㅇ 주요 기능 개발 완료 결과물

화면	설명
	ㅇ 메인 페이지 - 사용자 계정별 대시보드 - 마이페이지(개인 정보 수정)
	ㅇ 동작 정의 - 감지 범위 선택 - 사용자 정의 의미 설정 - 사용자 메모 설정
	ㅇ 동작 촬영 - 3초간 영상 촬영 - 3회 촬영
	ㅇ 실시간 동작 인식 - 감지 범위 선택 - 실시간 인식 시작 / 중지
	ㅇ 사용자 동작 사전 - 감지 범위별 동작 사전 - 의미별 삭제 - 사전 일괄 삭제
	ㅇ 관리자 대시보드 - 아파요 등 긴급 알림 실시간 시각화 - 사용자 최근 메세지

4 제안배경 및 목적

ㅇ 제안배경
- 현대 사회는 디지털화가 가속화되고 있지만, 여전히 의사 표현이 어려운 사람들은 소통과 정보에서 소외되고 있음
- 의사 표현이 어려운 사람들은 자신의 의사를 표현할 방법이 제한되어 가족, 보호자, 의료진과의 소통이 원활하지 못함
- 이러한 소통의 어려움은 **단순히 개인의 불편을 넘어 의료, 복지, 가족 관계 전반에 영향을 미치는 사회적 과제**로 떠오르고 있음
- 제안하는 프로젝트는 언어가 아닌 **"얼굴과 손"이라는 인간의 가장 본질적인 표현 수단을 활용하여 의사소통의 새로운 가능성**을 열고자 함

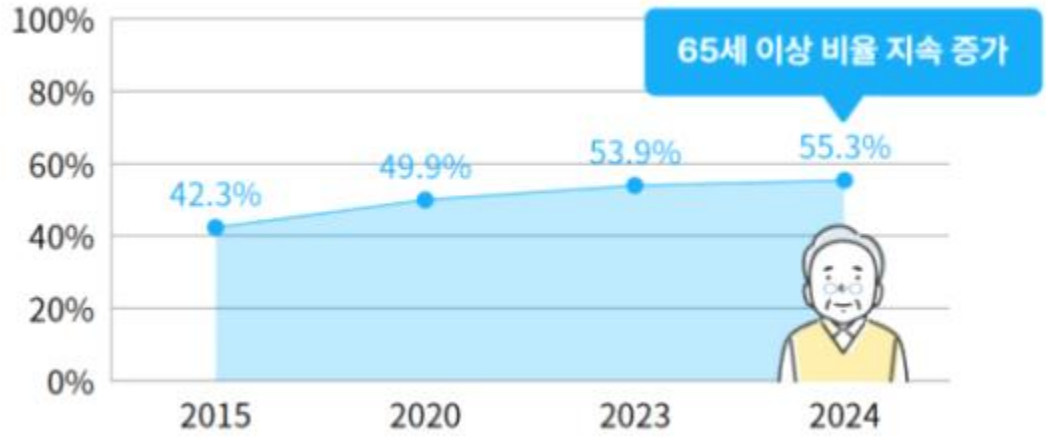
ㅇ 의사소통 취약계층 현황과 기존 Augmentative and Alternative Communication 기술의 한계 분석
- 의사소통 격차의 문제
  - [그림2]은 국내 등록 장애인 수의 추이를 나타냄

출처 : 국가데이터처

[그림 2] 국내 연도별 등록 장애인 추이

- 장애인복지법에서 규정한 15개 장애 유형 중 지체, 뇌병변, 청각, 언어, 지적, 자폐성, 호흡기, 뇌전증 장애인은 의사소통에 어려움을 겪는 경우가 많음
- 이들 장애 유형은 Augmentative and Alternative Communication(보완·대체 의사소통, 이하 AAC)의 직접적인 적용 대상군으로 분류됨
- 국내 등록장애인 수는 2019년 이후 약 260만 명 수준으로 지속되고 있으며, 이 중 청각·언어 장애인의 비중은 매년 꾸준히 존재함
- 이러한 수치는 언어 표현이 어려운 이들의 커뮤니케이션 문제와 정보 접근성 격차가 구조적 문제임을 시사함

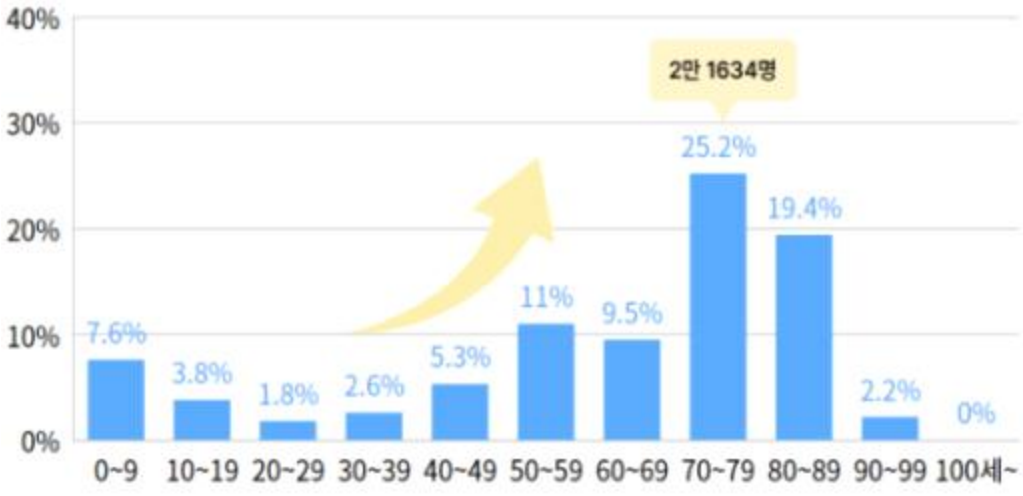
- 고령 장애인 비율과 후천적 장애 증가
  - [그림3]은 65세 이상 등록 장애인 비율을 나타냄



출처 : 국가데이터처

[그림 3] 65세 이상 등록 장애인 비율

- 그래프에 따르면 등록장애인 중 65세 이상 고령 장애인의 비율은 2015년 42.3%에서 2024년 55.3%로 지속적으로 증가하고 있음
- 고령화와 장애가 중첩되는 경우에는 신체적 기능 저하 및 의사소통 취약 등이 복합적으로 발생하여 맞춤형 돌봄·소통 지원의 필요성이 커지고 있음
- 따라서 고령 장애인을 위한 의료·돌봄·소통 지원 체계 강화는 향후 장애인 복지의 핵심 과제로 부상하고 있음
- 고령 장애인은 일반 성인 장애인에 비해 언어 표현력 및 반응속도가 더 크게 저하될 수 있어, 신속하고 직관적인 의사소통 수단이 반드시 필요함
- 돌봄 인력 부족과 1:1 관리의 한계로 인해, 기술 기반의 커뮤니케이션 지원 솔루션은 고령 장애인의 돌봄 공백을 줄이고, 보호자 및 의료진의 부담을 완화하는 대안으로 주목받고 있음
- [그림 4]은 2024년 신규 등록장애인 통계임



출처 : 국가데이터처

[그림 4] 신규 등록 장애인 연령대별 비중

- 신규 등록장애인의 연령대 중 70~79세는 25.2%로 가장 높은 비율을 차지하고 있으며, 60대 후반(21.1%)과 80대(19.4%)에서도 높은 비중을 보임
- 이러한 통계는 장애가 중·고령기에 집중적으로 발생하고 있음을 의미하며, 노화에 따른 신체 기능 저하가 장애 발생의 주요 요인으로 작용하고 있음을 시사함
- 또한 중·고령기 신규 장애인은 기존 생활 방식 변화, 의사소통 및 정보 접근의 어려움을 동시에 겪기 쉬워 맞춤형 접근이 필요함
- 특히 후천적 장애를 겪은 고령자의 경우, 언어 표현력 저하와 심리적 위축이 동반되기 쉬워 즉각적이고 직관적인 소통 수단에 대한 수요가 더욱 높음
- 이에 따라 고령 신규 장애인을 위한 AI 기반 커뮤니케이션 보조 기술은 단순한 소통 도구를 넘어 정서적 안정 지원과 자율성 회복을 위한 핵심 수단으로 기능할 수 있음
- [그림 5]은 연령별 주요 장애 유형의 차이를 나타낸 그래프임

출처 : 국가데이터처

[그림 5] 연령별 주요 장애 유형 차이

- 65세 이상 등록장애인의 주요 장애 유형은 지체장애(46.6%)와 청각장애(25.3%)가 압도적으로 높은 비중을 차지함
- 이러한 분포는 노화에 따른 신체 기능 저하 및 청력 손실이 고령층 장애 발생의 주요 요인임을 보여줌
- 따라서 고령층 장애인에게는 감각 보조 중심의 신체 기능 지원이 우선되어야 함을 시사함
- 특히 지체 및 청각 중심의 장애 유형은 의사소통이 어려운 경우가 많으므로, 연령 맞춤형 소통 보조 기술 및 AI 기반 커뮤니케이션 솔루션의 도입 필요성이 강조됨
- 고령 장애인은 후천적 장애 경험으로 인해 심리적 위축과 사회적 고립을 겪기 쉬우며, 이로 인해 의사소통 회피 및 돌봄 의존성이 증가하는 경향을 보임
- 또한 돌봄 인력 부족과 보호자의 1:1 지속 대응이 어려운 현실은 기술 기반 소통 보조 솔루션의 도입 필요성을 더욱 가중시킴
- 이러한 환경적 요인들은 얼굴·손동작 등 비언어적 표현을 즉시 해석할 수 있는 CueCode와 같은 실시간 AI 기반 커뮤니케이션 기술의 필요성을 더욱 부각시킴
- CueCode는 고령층에서도 부담 없이 사용할 수 있는 웹캠 기반 인터페이스를 제공하며, 맞춤형 동작 등록 기능을 통해 개인의 신체 기능 저하 정도에 맞춘 유연한 소통 방식을 지원함

- 기존 Augmentative and Alternative Communication(AAC)의 한계
  - [그림6]는 AAC에 대한 정의임

출처 : 제주특별자치도 장애인 종합 복지관

[그림 6] AAC의 정의

- 보완대체의사소통(AAC) 체계는 말이나 글로 의사표현이 어려운 사람을 위해 다양한 기호, 그림, 동작, 음성 출력 장치 등을 활용하여 의사소통을 지원하거나 대체하는 모든 형태의 방식을 의미함
- 의사소통 보조도구는 개인의 잔존 의사표현 능력을 보완하거나, 표현 수단이 전혀 없는 경우 이를 대체하는 역할을 수행함
- 의사표현 지원 기술은 그림카드·보드와 같은 저기술(Lo-Tech) 방식부터 전자기기 및 AI 기반 음성 출력형 고기술(Hi-Tech) 방식까지 폭넓은 형태를 포함함
- 의사소통 지원 시스템은 청각·언어 장애뿐 아니라 지체장애, 뇌병변장애 등 신체적 표현이 어려운 다양한 대상에게 적용될 수 있음
- 의사소통 보조체계의 활용은 사용자의 의사표현 자율성을 높이고, 보호자, 의료진, 교사 등과의 상호작용을 원활하게 함으로써 삶의 질 향상에 기여함
- 디지털 기반 보완의사소통 기술은 사용자 데이터와 결합될 경우, 맞춤형 의사소통 환경을 구축할 수 있어 기술 발전과 함께 활용 가능성이 더욱 확대되고 있음
- [그림7]은 기존 AAC 서비스의 특징을 정리한 표임

서비스명	사용자 입력 방식	상징의 개인화	관리자기능
MYAAC	클릭	가능	있음
Smart AAC	터치, 스위치	불가능	없음
	터치	불가능	없음

[그림 7] 기존 AAC의 특징

- 기존 AAC는 사전 정의된 문장과 아이콘 중심으로 구성되어 있어 개인의 고유한 표현 방식이나 변화하는 소통 패턴을 반영하기 어려움
- 정형화된 구성 방식은 장기적인 맞춤형 커뮤니케이션 지원 체계 마련의 필요성을 제기함
- 문장 전달 중심 구조는 긴급성, 감정 등 사용자의 의도적 메시지를 충분히 표현하기 어렵다는 한계를 보임
- [그림 8]은 기존의 AAC에서 제공하는 입력방식임

SmartAAC에서 제공하는 입력 방식　　CueCode의 영상 인식 입력

[그림 8] 기존 AAC와 CueCode의 입력 방식 차이

- 많은 AAC 서비스는 클릭 또는 터치 중심 입력 방식에 과도하게 의존하는 경향을 보임
- 이러한 구조는 손 사용이 어렵거나 복잡한 UI 접근에 제한이 있는 사용자가 실질적인 의사소통 도구로 활용하기 어려운 문제를 초래함
- 단일 입력 방식으로 구성된 기존 AAC 구조는 돌발 상황이나 긴급한 도움 요청 시 즉시 전달 가능한 직관적 상호작용을 제공하지 못함
- 현행 시스템은 보호자·교사·치료사가 사용자 상태를 분석하거나 사용 패턴에 기반한 맞춤 지원 전략을 수립할 수 있는 관리 기능이 부재함
- 관리 기능의 결핍은 돌봄 및 재활 과정과 연계 가능한 통합형 AAC 관리 시스템의 필요성을 시사함
- 정적 표현 방식에 머무르는 기존 구조는 표정이나 손짓 등 일상 속 비언어적 표현을 반영하지 못해 감정 전달과 긴급 상황 표현이 제한됨
- 자연스러운 의사전달을 지원할 수 있는 비언어 기반 소통 기술의 도입은 필수적 과제로 대두됨
- 고정형 구조를 가진 기존 AAC 플랫폼은 영상 인식, 시선 추적, 모션 인식 등 차세대 입력 기술과의 연동에 제약이 존재함
- 누적된 기술적·표현적 제약을 가진 기존 시스템은 사용자의 환경 변화에 능동적으로 대응하고 개인 소통 패턴을 학습할 수 있는 AI 기반 적응형 비언어 커뮤니케이션 시스템의 도입 필요성을 명확히 보여줌
- 이러한 한계점은 얼굴·손동작·표정 등 자연스러운 비언어 신호를 실시간으로 해석할 수 있는 CueCode와 같은 멀티모달 기반 솔루션의 도입 필요성을 더욱 부각함
- CueCode는 사용자가 직접 동작·제스처를 등록하여 자신의 고유한 표현 방식을 반영할 수 있으며, 응급 상황 감지 기능을 통해 기존 AAC에서 부족했던 긴급 대응 기능까지 보완함

- 기존 AAC의 한계를 기반으로 분석한 사용자의 요구사항
  - [그림 9] 기존 AAC의 한계와 사용자 요구사항을 나타냄

[그림 9] 기존 AAC의 한계와 사용자 요구사항

- 사용자는 개인별 표현 방식을 등록하고 학습할 수 있는 맞춤형 커뮤니케이션 시스템 제공을 요구함
- 사용자는 위기 상황을 즉시 알릴 수 있는 직관적·감정적 입력 방식과 감정 기반 표현 기능을 요구함
- 사용자는 손을 사용하지 않고도 활용 가능한 모션·표정 등 멀티모달 입력 방식을 요구함
- 사용자는 보호자·치료사를 위한 사용 현황 분석 및 패턴 리포트 기능을 요구함
- 사용자는 대체 입력 기술과 유연하게 연동되며 환경 변화에 따라 학습·적응하는 AI 기반 커뮤니케이션 시스템을 요구함
- 이러한 요구는 단순한 UI 확장이 아니라, 입력 기술·동작 인식·문장 생성이 유기적으로 연결된 고도화된 AI 커뮤니케이션 엔진의 필요성을 의미함
- 특히 실시간 처리·응급 상황 감지·개인화 모델링 등 기술적 요구 수준이 높아짐에 따라, 시스템 전반에 AI 기반 적응형 구조를 적용하는 것이 필수 요건으로 부상하고 있음
- 사용자 맞춤형 표현을 지속적으로 학습하기 위해서는 모델 업데이트와 데이터 관리가 자동화된 구조가 필요함
- 또한 다양한 디바이스·네트워크 환경에서도 안정적으로 동작할 수 있는 경량화·최적화 기술이 필수 요소로 요구됨

5 세부내용

ㅇ 활용 데이터 및 AI 모델

- **활용 데이터 : 위급상황 음성/음향 (고도화) - 119 지능형 신고접수 음성 인식 데이터**
  - AI-Hub 「위급상황 음성-음향(119 지능형 신고접수)」 데이터셋의 라벨링 JSON 데이터 총 127,178개를 EmergencyData 폴더에 저장하여 활용함
  - 각 JSON에서 urgencyLevel(상/중/하)과 신고자 speaker=1의 발화만 추출하여 응급도별 코퍼스를 구축함
  - JSON 기반 응급 문장 추출 파이프라인
    ① EmergencyData/*.json을 순회하며 신고자 발화를 수집하고, 응급도(상/중/하)별로 데이터셋을 분리함
    ② 2~4글자 한글 토큰 기준으로 단어 빈도를 계산하고, 불용어·환경 단어를 제거한 뒤 TF-IDF 방식으로 응급도별 특징 키워드 Top 50을 산출함
    ③ 특징 키워드와 신체 증상 키워드를 모두 포함하는 문장만 필터링하고, 길이(짧음/중간/긴)별로 분류함
    ④ 응급도 간 중복을 제거하고 다양성을 고려해 각 응급도별로 대표 문장 50개씩을 최종 선정함
    ⑤ 선정된 문장을 TXT, CSV, JSON, 학습 데이터(문장[TAB]응급도) 형식으로 저장하여 이후 AI 모델 학습 및 룰 엔진에 활용 가능하도록 구성함
  - JSON 기반 응급 문장 정규화 파이프라인
  - 규칙 기반 정규표현식을 활용하여 원본 문장을 "의식이 없어요", "숨을 못 쉬어요" 등 표준 응급 문장 템플릿으로 정규화함
  - 응급도(상/중/하)별로 중복 없는 정규화 문장을 최대 50개까지 구축하고, TXT·CSV·JSON·학습 데이터 형태로 다시 저장하여 CueCode의 응급 알림 로직 및 LLM 프롬프트에 직접 활용할 수 있도록 함
  - [그림 10]는 위급상황 음성-음향(119 지능형 신고접수)」 데이터셋의 활용 코드임

```
    print("학습 데이터 저장: emergency_json_based_training_data.txt")

    # 3. 키워드 분석 결과
    with open('emergency_distinctive_keywords.txt', 'w', encoding='utf-8') as f:
        f.write("="*100 + "\n")
        f.write("응급도별 특징 키워드 분석 결과\n")
        f.write("="*100 + "\n\n")

        for urgency in ['상', '중', '하']:
            f.write(f"\n{'='*100}\n")
            f.write(f"[{urgency}] 응급도 - 상위 50개 특징 키워드\n")
            f.write(f"{'='*100}\n\n")

            for i, (word, count, score) in enumerate(distinctive_keywords[urgency][:50], 1):
                f.write(f"{i:2d}. {word} (빈도: {count}회, 차별화 점수: {score:.2f})\n")
            f.write("\n")

    print("키워드 분석 저장: emergency_distinctive_keywords.txt")

def show_samples(selected_sentences):  1 usage  new *
    """샘플 미리보기"""
    print("\n" + "="*100)
    print("샘플 미리보기")
    print("="*100)

    for urgency in ['상', '중', '하']:
        print(f"\n[{urgency}] 응급도 (상위 10개):")
        for i, sentence in enumerate(selected_sentences[urgency][:10], 1):
            print(f"  {i:2d}. {sentence}")
```

[그림 10] 위급상황 음성-음향(119 지능형 신고접수)」 데이터셋의 활용

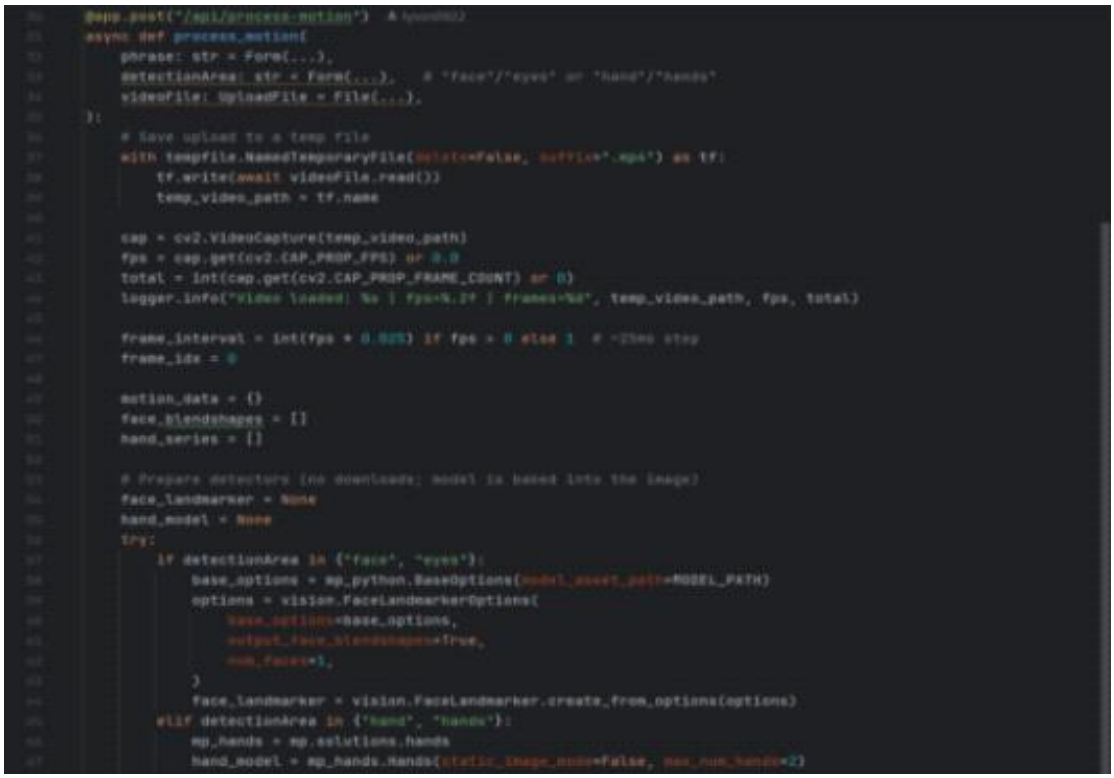
- **AI 모델 : Google Mediapipe**
  - [그림 11]는 CueCode의 동작 데이터 저장 및 처리 흐름을 나타냄

[그림 11] CueCode의 동작 데이터 저장 및 처리

① 사용자가 웹 화면을 통해 동작 영상을 촬영하면, API Gateway를 통해 Motion Service(Spring Boot)로 업로드됨
② Motion Service는 촬영된 영상을 FastAPI 서버로 전송하며, 분석 구역(예: "face", "hand")을 함께 전달함
③ FastAPI는 Google MediaPipe의 사전 학습된 Face/Hand 모델을 활용하여 프레임별 표정 계수(Blendshapes) 또는 손 관절 좌표(21개 Joint 기준)를 추출함
④ 분석 과정에서는 원본 영상 전체를 처리하지 않고 일정 주기로 프레임을 샘플링하여 경량화된 데이터를 생성함
⑤ Motion Service는 반환된 JSON 형태의 결과를 MongoDB Document 형식에 맞게 변환 후 저장함
⑥ 변환된 JSON은 사용자 ID, 문장(phrase), 동작 타입, 시점별 동작 변화 등 분석에 필요한 데이터만 포함하여 저장됨

- CueCode는 Google Mediapipe 기반의 경량 추출 방식을 사용함
- 영상 전체 저장 대비 데이터 크기를 1/100 이상으로 줄여 저장 성능 및 조회 속도 향상시킴
- 표정·손동작 등의 세부 정보만 추출하기 때문에 이후 인식/딥러닝 학습에 유리함
- 프레임별 좌표값만 저장되므로 환자/사용자의 민감한 영상 정보를 별도로 보관할 필요가 없음
- 사전 학습된 MediaPipe 모델 활용을 통해 빠른 실시간 추론이 가능하며 GPU 없이도 경량 환경에서 동작 가능

- [그림 12]은 CueCode의 Google MediaPipe 적용 코드 중 일부임



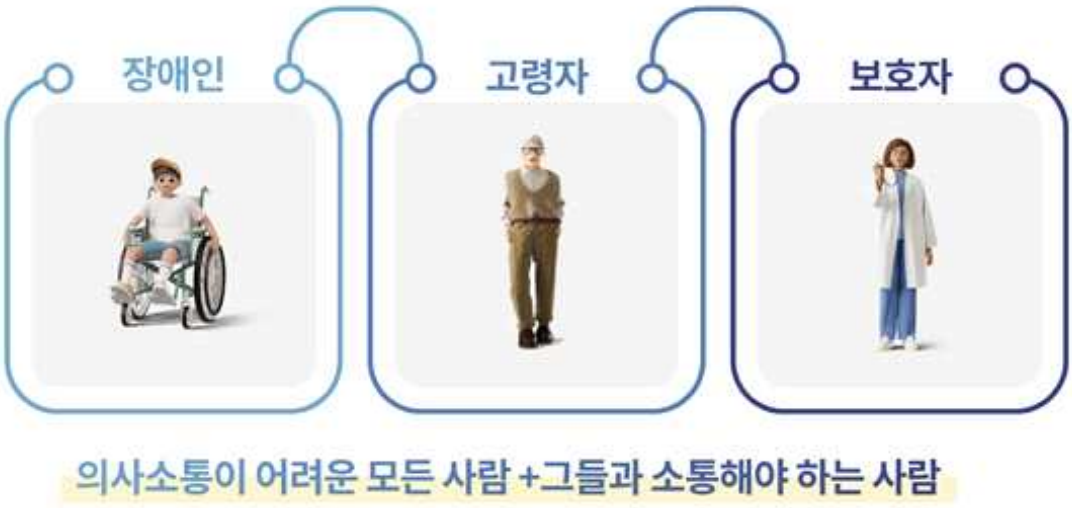
[그림 12] CueCode의 Google MediaPipe 적용 코드

- Google MediaPipe는 얼굴·손·신체 등의 동작을 실시간으로 인식하고 좌표 기반 데이터로 반환할 수 있는 경량 AI 비전 프레임워크임
- FastAPI 서버에서는 MediaPipe의 FaceLandmarker, Hands 모듈을 사용하여 동작 프레임별 특징값(좌표/표정 계수)을 자동 추출함
- face_landmarker_v2_with_blendshapes.task 모델을 사전에 Docker 이미지에 포함하여, 실행 시 외부 다운로드 없이 즉시 사용 가능하도록 구성함

- **다중 LLM 오케스트레이션 기반 LangGraph 커뮤니케이션 구조 설계 (예상)**
  - CueCode의 문장 생성 파이프라인은 Llama(로컬), GPT(OpenAI), Gemini 세 가지 LLM을 LangGraph 상에서 역할 분리하여 오케스트레이션하는 구조로 설계함
  - Llama는 로컬에서 동작하는 무료 모델로, 사용자 입력을 EMERGENCY / REQUEST / STATUS / OTHER로 분류하는 Intent Classifier 역할을 수행함
  - Llama는 너무 긴 문장·비속어·비정상 패턴을 1차 필터링하고, GPT 장애 시 간단한 Fallback 문장을 생성하는 보조 엔진으로 활용됨
  - GPT는 한국어 기반 응급 문장과 일반 요청 문장을 생성하는 메인 Sentence Generator로 사용됨
  - GPT는 불완전한 표현을 "물 좀 주세요."와 같은 공손한 단문으로 다듬는 Refiner 역할도 수행하며, 기본 모델은 gpt-4.1-mini를 사용하고 필요 시 gpt-4.1을 선택적으로 적용함
  - Gemini는 생성된 문장을 대상으로 길이·문장 수·공손한 표현 여부·응급 표현 강도 등을 점검하는 Rule/Safety Checker 역할을 수행함
  - Gemini는 "OK", "TOO_LONG", "NOT_POLITE", "UNCLEAR" 같은 판정 태그를 반환하고, LangGraph는 해당 결과에 따라 GPT 재생성 노드 또는 최종 출력 노드로 흐름을 전환함

ㅇ 서비스 내용

- 서비스 제공 대상
  - [그림 13]은 CueCode의 서비스 제공 대상자들을 나타냄



[그림 13] CueCode 서비스 제공 대상

- 광의(廣義)의 의미에서 CueCode는 단순히 장애인이나 고령자만을 위한 보조 도구가 아니라, 의사소통이 어려운 모든 사람과 그들과 소통해야 하는 사람을 연결하는 포용적 커뮤니케이션 플랫폼을 의미함
- 즉, 발화나 청각, 인지 기능의 한계를 넘어 누구나 자신만의 방식으로 의사를 표현하고 이해받을 수 있는 확장된 개념의 커뮤니케이션 생태계를 지향함

- 다음은 핵심 서비스 제공 대상에 대한 설명임
- **청각장애인** : 음성 의사소통이 어렵고, 수어 통역 지원이 제한된 환경에서도 손동작 인식 기능을 통해 자신의 의사 표현이 가능함
- **언어장애인** : 뇌손상, 발달장애, 질환 등으로 말하기 어려운 경우 표정·몸짓으로 의사소통 가능함
- **지체장애인** : 손 동작이 일부 제한된 사용자도, MediaPipe 기반의 미세 동작 인식 기능을 활용해 커뮤니케이션 지원이 가능함
- **인지 저하자** : 치매 초기나 경도 인지장애로 언어 표현이 감소한 노인에게 손동작·표정 기반의 단순한 명령 표현 기능 제공됨
- **거동 불편자 / 회복기 노인** : 병원이나 요양시설 등에서 말하기 어려운 상황에서 의사 전달 수단으로 활용 가능함
- **가정 내 보호자** : 환자나 가족의 상태를 정확히 이해하고 돌봄 효율을 높이기 위한 소통 보조 도구로 활용함
- **의료진 / 간병인** : 발화가 어려운 환자와의 커뮤니케이션 도구로, 응급 상황에서 신속한 판단 가능함
- **교사 및 사회복지사** : 특수교육 또는 사회복지 기관에서 장애학생·노인과의 상호작용을 보조하는 학습·상담 도구로 활용 가능함

- 주요 기능
  - [그림 14]은 CueCode가 제안하는 서비스의 주요 기능을 나타냄

[그림 14] CueCode가 제안하는 서비스의 주요기능

- 얼굴 표정 인식 : MediaPipe를 활용해 표정 변화를 실시간 분석하고 의미 매칭을 수행함
- 손동작 인식 : 손 모양과 움직임을 인식하여 사용자의 동작을 식별하고 의미 매핑을 수행함
- 동작-문장 변환 : DTW 알고리즘을 통해 연속 동작을 문장단위로 해석함
- 사용자 맞춤 사전 : 자주 사용하는 동작과 문장을 등록·저장하여 개인화된 표현 지원함
- 텍스트 및 음성 출력 : 인식된 동작을 텍스트와 음성으로 변환하여 상대방에게 전달함
- 보호자 공유 : 인식된 내용과 사용자의 상태를 보호자에게 실시간으로 공유함
- 긴급 상황 알림 : 위급 상황 감지 시 보호자 또는 의료진에게 즉시 알림 전송

- 서비스 제공 절차
  - [그림 15]은 CueCode가 서비스 제공 절차를 나타냄

[그림 15] CueCode의 서비스 제공 절차

① 사용자는 역할에 따라 환자용 / 관리자용 계정으로 가입함
② 환자는 이메일을 포함한 기본정보와 감지 범위를 설정함
③ 관리자는 의료기관 또는 돌봄시설 단위로 등록되어 여러 환자를 관리할 수 있음
④ 환자는 영상 촬영을 통해 자신만의 동작을 등록함
⑤ 각 동작은 사용자가 직접 의미를 부여하며, 시스템이 이를 개인 사전으로 저장함
⑥ 기본 데이터셋(예: "예 / 아니오 / 도움요청") 외에 사용자가 자주 쓰는 문장이나 표현을 커스터마이징할 수 있음
⑦ 등록된 영상은 MediaPipe → DTW 순으로 분석되어 동작과 문장의 매핑 정확도를 향상시킴
⑧ 환자가 카메라를 통해 손동작이나 표정을 수행 함
⑨ 시스템이 이를 MediaPipe로 인식 → DTW로 유사도 비교로 문맥 분석함
⑩ 분석 결과는 텍스트·음성으로 출력됨
⑪ 관리자는 환자를 조회 및 추가하여 관리할 수 있음
⑥ 상대방(보호자, 의료진 등)은 환자의 최근 메시지를 즉시 확인 가능함

- 서비스 구성도
  - [그림 16]은 CueCode가 제안하는 서비스 구성도를 나타냄

[그림 16] CueCode가 제안하는 서비스 구성도

- 본 시스템은 사용자 웹 애플리케이션을 중심으로, 사용자 관리, 개인 맞춤 모션 등록, 실시간 분석, 긴급 동작 인식 서비스를 통합적으로 제공하는 구조로 설계됨
- 사용자 서비스 : 회원 정보 및 프로필 데이터를 관리하며, 인증/인가를 담당함
- 개인 맞춤 모션 등록 서비스 : 사용자가 직접 영상을 업로드하고 동작에 의미를 태깅하여 개인 맞춤 모션 사전에 저장하도록 지원함
- 실시간 분석 서비스 : MediaPipe-DTW 기반의 파이프라인을 통해 사용자의 동작을 실시간 분석하고, 결과를 텍스트 또는 음성 형태로 출력함
- 관리자 기능 서비스 : 사용자의 행동을 지속적으로 감지하며 긴급상황을 나타내는 동작이 판별될 경우 즉시 알림을 제공함
- 데이터 저장 구조는 MongoDB를 통해 사용자 정보 및 개인화된 모션 데이터와 분석 결과를 저장함
- RedisDB를 활용해 분석과정 중 상태 값과 자주 요청되는 데이터를 캐싱함으로써 실시간 응답성과 성능을 보장함

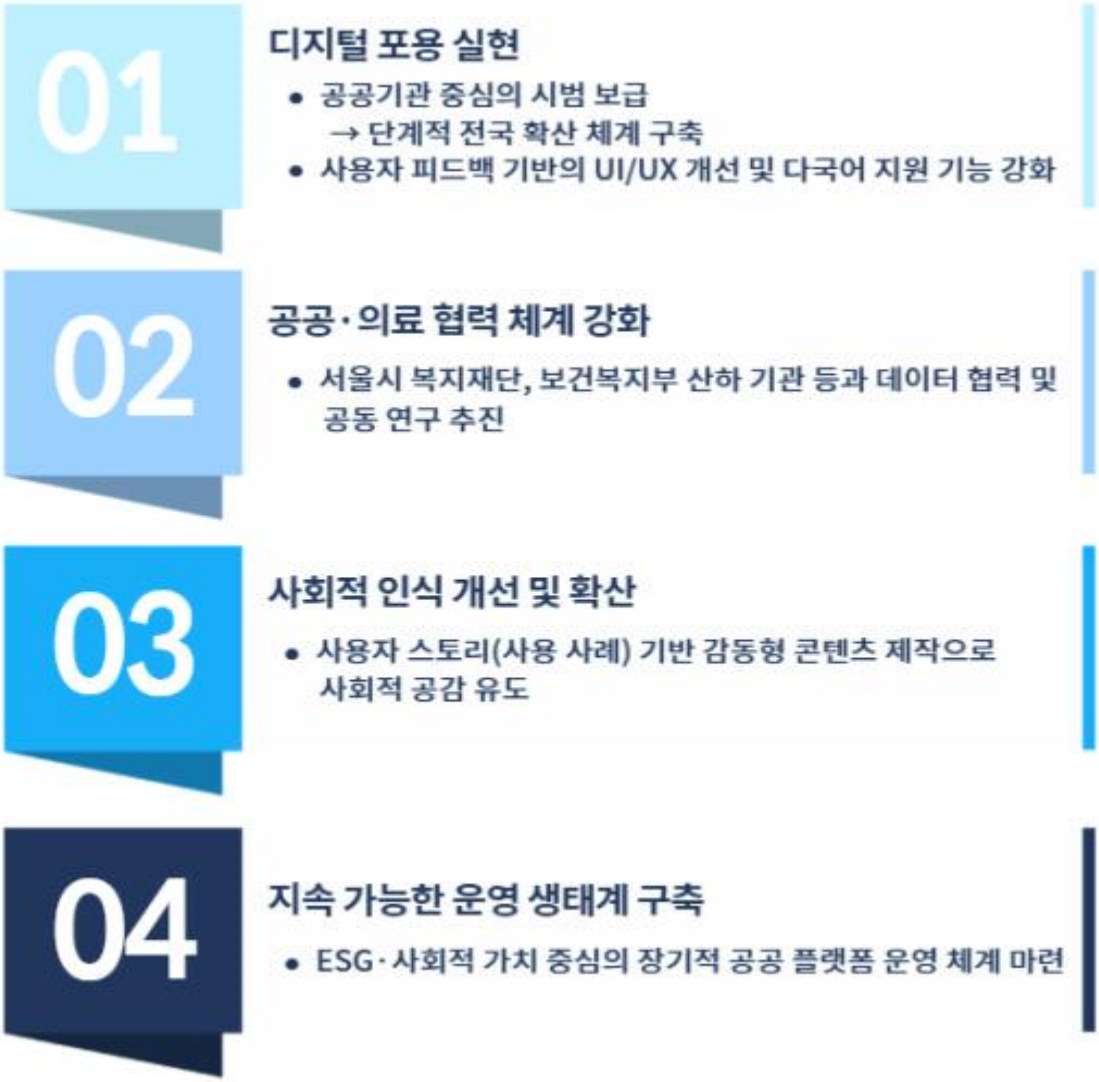
o 추진 전략 및 방안

- CueCode 추진 목표
  - [그림 17]는 CueCode의 추진 목표임

[그림 17] CueCode의 추진 목표

- 아래는 추진 목표별 세부 실행 방향 내용임
- 디지털 포용 실현 : 2027년까지 3,000명 이상 사용자 확보 및 20개 기관 도입 목표
- 공공·의료 협력 체계 강화 : 서울시 복지재단, 보건복지부 산하 기관 등과 MOU 체결 및 공동 실증사업 추진
- 사회적 인식 개선 및 확산 : SNS, 유튜브, 블로그 등 디지털 홍보 채널을 통한 사회적 가치 확산
- 지속 가능한 운영 생태계 구축 : Microsoft Azure 기반 클라우드 인프라를 활용한 안정적 클라우드 운영 환경 조성
- CueCode는 디지털 포용 실현, 공공·의료 협력 강화, 사회적 인식 개선, 지속 가능한 운영 생태계 구축을 핵심 축으로 삼고 있음
- 네 가지 방향은 고령층·장애인의 의사소통 격차를 해소하고 접근성을 높이기 위한 통합적 전략임
- AI 기반 비언어 인식 기술을 고도화하여 개인 맞춤형 소통 환경을 확대하고자 함
- 기관·가정·사회 전반을 연결하는 소통 생태계를 구축해 지속 가능한 서비스 운영을 목표로 함
- 데이터 표준화, 글로벌 확장성 확보, 보안·신뢰 기반 클라우드 운영 체계 정착을 단계적으로 추진함
- 오픈 API 중심의 개방형 플랫폼으로 확장하여 다양한 보조공학 기술 및 외부 서비스와 연동 가능한 구조를 마련함

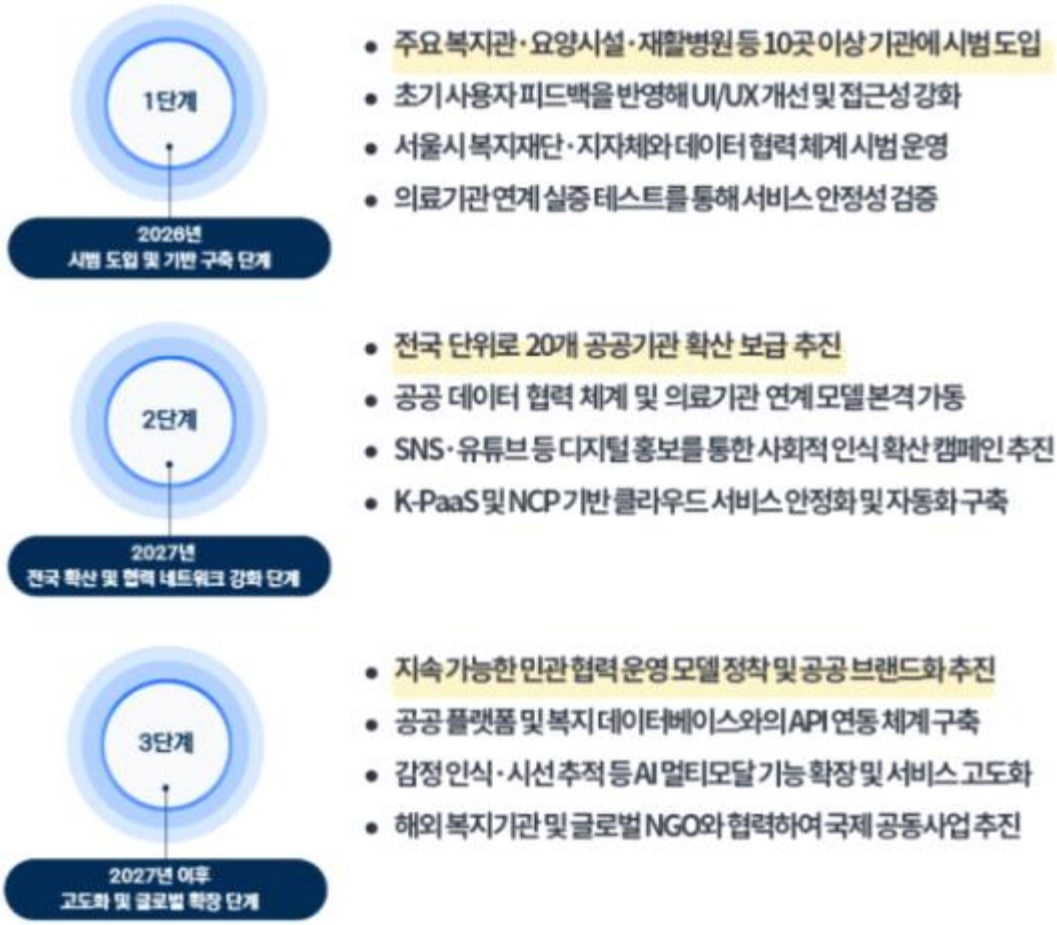
- CueCode 추진 전략
  - [그림 18]은 CueCode의 세부 추진 전략임



[그림 18] CueCode 세부 추진 전략

- CueCode는 디지털 포용 실현, 협력 체계 강화, 사회적 확산, 지속 가능성 확보의 네 축을 중심으로 공공성과 기술혁신이 결합된 AI 기반 커뮤니케이션 복지 플랫폼으로 발전을 추진하고자 함
- 공공기관 중심의 시범 보급을 통해 초기 사용자 기반을 확보함
- 단계적 전국 확산 체계를 마련하여 서비스 접근성을 확대함
- 사용자 피드백을 반영한 UI/UX 개선 및 다국어·배리어프리 기능을 강화함
- 서울시 복지재단, 보건복지부 산하기관 등과 연계해 데이터 협업을 추진함
- 협력 기관과 공동 연구·실증을 통해 서비스 신뢰성과 공공성을 확보함
- 사회적 인식 개선 및 확산
- 사용자 스토리와 사례 기반 감동형 콘텐츠를 제작해 사회적 공감대를 유도함
- 다양한 미디어 플랫폼을 활용해 비언어 커뮤니케이션의 필요성을 대중에게 확산함
- 지속 가능한 운영 생태계 구축
- ESG·사회적 가치 중심의 장기적 공공 플랫폼 운영 체계를 마련함

- 추진 방안
  - [그림 19]은 CueCode의 추진 방안임



[그림 19] CueCode의 추진 방안

- CueCode는 시범 도입 → 전국 확산 → 글로벌 확장의 3단계 전략을 통해 공공성과 지속 가능성을 갖춘 AI 기반 커뮤니케이션 복지 플랫폼으로 발전하고자 함
- CueCode는 단계적 확산 전략을 통해 사회적 약자의 의사소통 격차를 해소하고, 공공·의료 협력 기반의 지속 가능한 커뮤니케이션 생태계를 구축할 예정
- 더불어 멀티모달 인식 기술·다중 LLM 기반 문장 생성·사용자 패턴 분석 등 핵심 AI 역량을 고도화하여 다양한 환경과 장애 유형에 대응할 수 있는 차세대 보완대체의사소통(AAC) 기술을 제공할 계획임
- 장기적으로는 지역사회 돌봄, 재활 의료, 공공 복지 서비스와 연계함으로써 사회적 약자 중심의 포용적 디지털 환경 조성에 기여하고, 글로벌 접근성 표준을 충족하는 국제형 커뮤니케이션 플랫폼으로 확장하고자 함

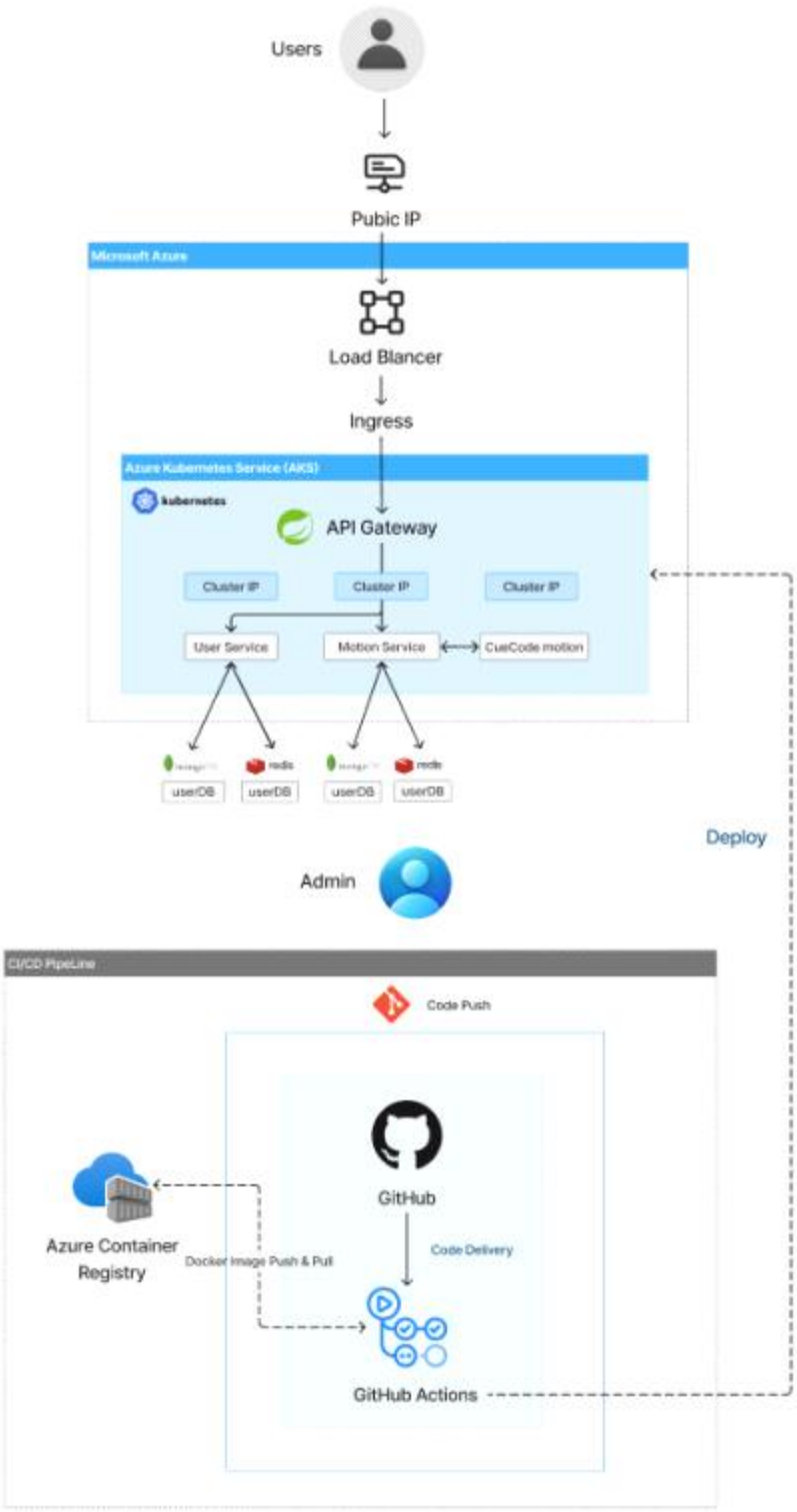
- 성과 지표
  - <표 1>는 CueCode 성과 지표임

<표 1> CueCode의 성과 지표

구분	정량 지표	정량 지표
보급 성과	- 전국 복지관·병원 등 50개 기관 도입 달성 - 3,000~5,000명 이상 사용자 확보	- 사용자 만족도 80% 이상 달성 - 서비스 접근성·UI 개선 만족도 향상
협력 성과	- 공공·의료기관 10곳 이상 MOU 체결 - 복지·의료 데이터 5종 이상 연계 구축	- 기관 간 협력 체계 안정화 - 공공 데이터 활용 모델 구축
사회적 가치 성과	- 비언어 사용자 의사소통 시간 30% 이상 단축 - 체험 캠페인 10회 이상 운영	- 디지털 포용성 강화 - 커뮤니케이션 복지 실현 기여
운영 지속성 성과	- Microsoft Azure 기반 클라우드 인프라 활용 안정화율 99% 이상 - 민관 협력 운영 모델 구축	- 장기적 서비스 유지·운영 역량 확보 - 민관 파트너십을 통한 지속성 강화

o 아키텍처

- 개발 환경 구성도
  - [그림 20]은 CueCode 개발 환경 구성도임

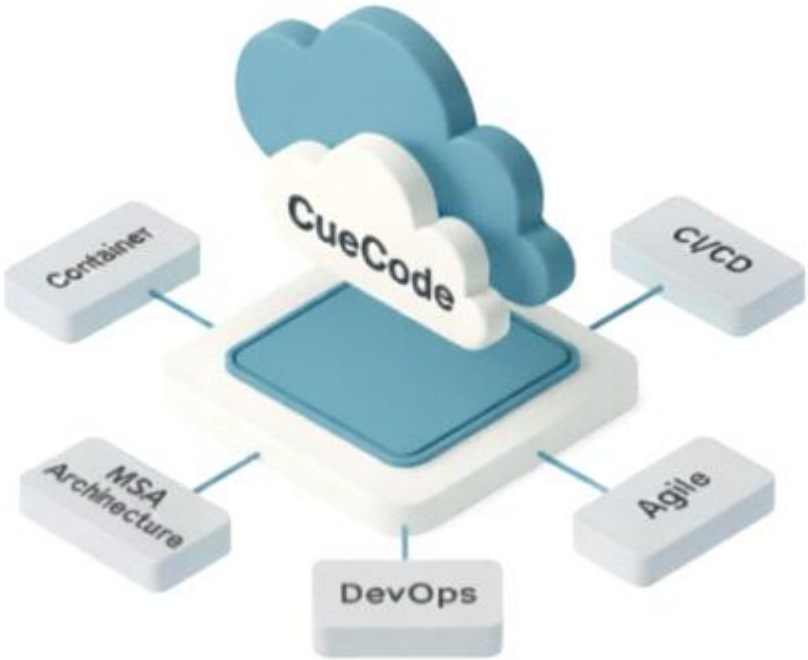


[그림 20] 개발 환경 구성도

구분	상세 내용
웹 배포 서버	- Microsoft Azure 기반 클라우드 인프라
개발 언어	- Java, Python, JavaScript, HTML5
개발 툴	- IntelliJ, PyCharm
주요 개발 프레임워크	- MSA에 적합한 개발 및 운영 환경을 제공하는 Spring Boot3 Framework 적용 - AI 분석 서비스 제공을 위한 Python FastAPI 적용 - API Gateway 구현을 위한 Spring Cloud Gateway 적용 - 비동기 요청 처리 및 실시간 응답을 위한 Spring WebFlux 적용 - 사용자 접근 제어 및 인증 관리를 위한 Spring Security와 JWT(Json Web Token) 적용 - 환경설정의 중앙 집중 관리를 위한 Spring Cloud Config Server 적용
AI 및 인식 기술	- 얼굴 및 손동작 인식을 위한 MediaPipe 적용 - 유사도 분석을 위한 DTW 알고리즘 적용 - AI 기반 동작-문장 매핑 기능 구현
데이터 베이스	- 비정형 데이터 저장 및 사용자 정보 관리를 위한 MongoDB 적용 - 동작 매핑 속도 향상을 위한 Redis 적용 - 서비스 간 데이터 연동을 고려한 Hybrid 구조 설계
라이브러리	- Java : Spring Cloud Gateway, Spring Boot Starter Web, Spring Boot Starter Security, Spring Boot Starter, Thymeleaf 등 - Python : FastAPI, Uvicorn, OpenCV, MediaPipe, NumPy 등
CI/CD	- Microsoft Azure사용 - 컨테이너 기반 서비스 운영을 위한 Docker 사용 - 서비스 모니터링 및 상태 제어를 위한 Spring Actuator 사용
형상 관리 도구	- Git
협업 및 설계 도구	- Swagger (API 명세) - Figma (UI/UX 설계) - Notion (작업 일정 관리)

ㅇ 개발 환경

- Cloud Native Application에 맞춘 개발
  - [그림 21]은 Cloud Native Application에 맞춘 개발을 표현함



[그림 21] CueCode의 Cloud Native Application

(1) Container

- 애플리케이션과 종속 요소를 하나의 패키지로 묶어, 실행 환경 간 일관성 유지함
- 개발·테스트·운영 환경 간 일관성을 유지하여 배포 오류를 줄이고 신속한 이전(Portability)을 지원함

(2) MSA Architecture

- 시스템을 기능 단위로 세분화하여 독립적인 마이크로서비스로 설계함
- 각 서비스가 독립적으로 배포·확장·장애 복구될 수 있어 운영 안정성 및 확장성이 강화됨

(3) DevOps

- 개발(Dev)과 운영(Ops) 간의 조직적 경계를 허물고 협업 중심의 개발·운영 문화를 구축함
- 코드 변경 사항을 빠르게 테스트 및 배포하는 자동화 프로세스를 통해 개발 사이클을 단축함

(4) Agile

- 대규모 개발을 한 번에 수행하는 방식이 아니라 반복적·단계적(Iteration) 방식으로 개발을 진행함
- 주 단위 또는 스프린트 단위로 결과물을 제공하여 리스크를 최소화하고 고객 만족도를 높임

(5) CI/CD

- 코드 커밋 이후 자동 빌드, 자동 테스트를 수행하여 통합 과정에서 발생하는 오류를 조기에 발견 및 수정함

- <표 2> 은 Cloud Native Application에 맞춘 개발을 요약한 표임

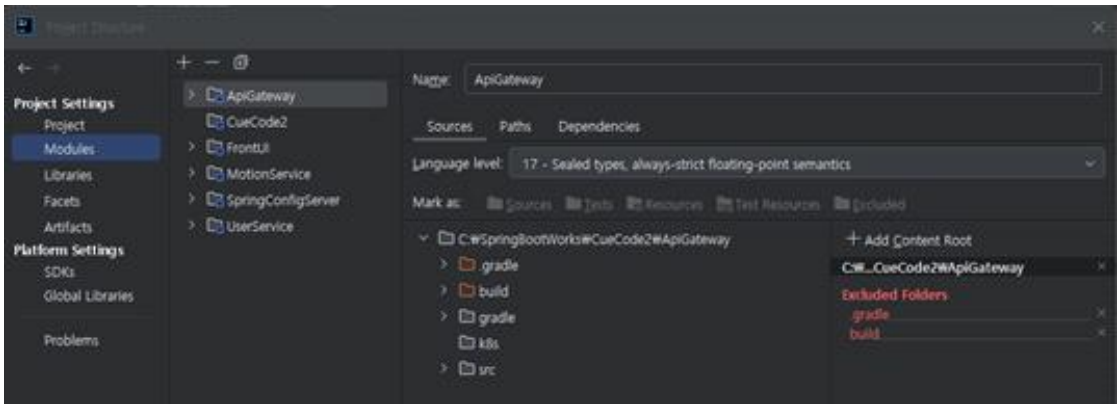
<표 2> CueCode의 Cloud Native Application 적용

구분	핵심 개념	상세 내용
Container	경량 실행 환경	- 애플리케이션과 필요한 요소를 하나로 묶어 어디서든 동일하게 실행 - 빠른 배포·확장 및 자원 효율성을 확보함
MSA	서비스 분리 구조	- 시스템을 독립적인 기능 단위로 분리해 각각 독립 배포·확장 가능 - 서비스 간 결합도를 낮춰 유연한 유지보수가 가능함
DevOps	개발-운영 통합 문화	- 개발팀과 운영팀 간 협업을 강화해 빌드 및 배포를 자동화 - 모니터링 기반 운영 최적화를 통해 지속적인 개선을 지원함
Agile	반복 및 단계적 개발 방식	- 짧은 주기로 기능을 점진적으로 개발 - 변경 요구에 유연하게 대응하여고객 중심의 개발 프로세스를 실현
CI/CD	자동 통합 및 지속적 배포	- 코드를 저장소에 반영한 후 자동 빌드, 테스트, 배포가 이루어짐 - 오류 조기 발견 및 안정적인 반복 배포를 가능하게 함

ㅇ 클라우드 네이티브 원칙을 적용한 확장형 MSA 구조로 개발

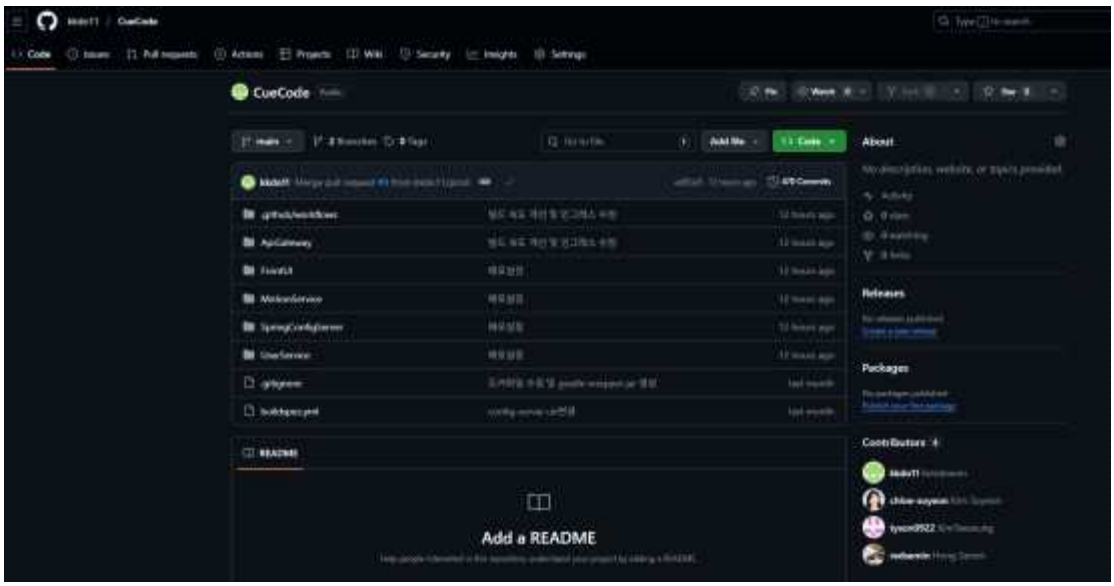
(1) CodeBase (코드 베이스)

- [그림 22]은 CueCode의 기능별 모듈관리임



[그림 22] CueCode의 기능별 모듈관리

- 12-Factor 원칙 중 Codebase에 따라 각 마이크로서비스는 독립된 코드베이스로 관리됨
- 이를 통해 서비스 간 의존성을 최소화하고, 각각의 서비스가 별도 저장소에서 독립적으로 개발 및 유지보수가 가능함
- 코드 분리를 기반으로 서비스별 기능 변경 시 전체 시스템에 영향을 주지 않으며, 빠른 수정 및 배포가 가능함
- [그림 23]는 CueCode의 형상관리에 대한 내용임



[그림 23] CueCode의 형상 관리

- 프로젝트 변경 사항을 커밋 단위로 기록 및 관리함
- 각 커밋 메시지를 통해 코드 변경 목적 및 수정 내역을 명확히 설명하여 추적이 용이함

(2) Dependencies (종속성)

- 각 마이크로서비스는 Gradle 기반으로 필요한 외부 라이브러리와 패키지를 명시적으로 선언하여 종속성을 관리함
- 모든 의존 요소는 중앙 저장소에서 자동으로 다운로드되며, 수동 설치 없이도 동일한 구성으로 실행 가능함
- [그림 24]은 CueCode의 종속성 관리임

```
dependencies { Add Starters...
    // WebFlux + Security (버전은 Boot가 관리)
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'org.springframework.boot:spring-boot-starter-security'

    // Spring Cloud (버전 명시 금지: BOM이 잡힐)
    implementation 'org.springframework.cloud:spring-cloud-starter-gateway'
    implementation 'org.springframework.cloud:spring-cloud-starter-config'

    // JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    runtimeOnly   'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly   'io.jsonwebtoken:jjwt-jackson:0.11.5'

    // Actuator
    implementation 'org.springframework.boot:spring-boot-starter-actuator'

    // Lombok
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    // Test
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.security:spring-security-test'
    testImplementation 'io.projectreactor:reactor-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}
```

[그림 24] CueCode의 종속성 관리

(3) Config (설정)

- 환경 설정정보는 앱에 포함하지 않고 분리하여 관리함
- [그림 25]은 CueCode의 Config파일에 대한 내용임

```
spring:
  config:
    activate:
      on-profile: dev
  import: "optional:configserver:http://localhost:8888/"
```

```
package kopo.springconfigserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.config.server.EnableConfigServer;

@EnableConfigServer
@SpringBootApplication
public class SpringConfigServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringConfigServerApplication.class, args);
    }

}
```

[그림 25] CueCode의 Config 설정

- @EnableConfigServer 어노테이션으로 Config Server 기능만을 담당하는 독립적인 서비스 구성됨
- 모든 마이크로서비스의 설정을 중앙에서 관리하여 코드와 설정의 완전한 분리를 유지함
- Spring Actuator를 활용해 설정 변경 사항을 실시간으로 반영할 수 있어, 재배포 없이 환경별 설정 수정이 가능함

(4) Backing Services (백엔드 서비스)

- 모든 서비스는 RESTful 방식으로 연결된 리소스로 취급하여 구현됨
- 코드 수정 없이 데이터베이스(MongoDB ↔ RedisDB) 변경이 가능함
- [그림 26]은 CueCode의 Spring Data MongoDB코드임

```java
package kopo.userservice.repository.document;

import lombok.*;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Getter  14 usages  chloe-soyeon
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Document(collection = "detection_area")
public class DetectionAreaDocument {
    @Id
    private String patientId; // patientId가 MongoDB의 _id로 저장됨
    private boolean hand;
    private boolean face;
    private boolean both;
}
```

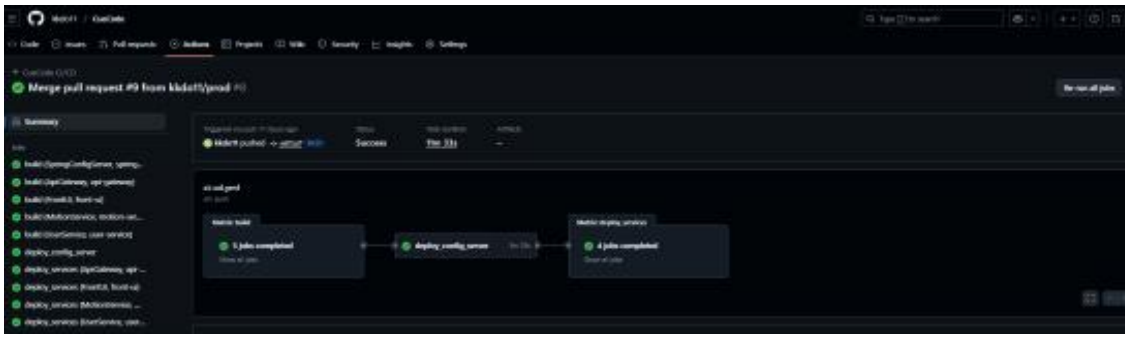
```java
package kopo.userservice.repository;

import kopo.userservice.repository.document.DetectionAreaDocument;
import org.springframework.data.mongodb.repository.MongoRepository;

import java.util.Optional;

public interface DetectionAreaRepository extends MongoRepository<DetectionAreaDocument, String> {  4 usages  chloe-soyeon
    Optional<DetectionAreaDocument> findByPatientId(String patientId);  2 usages  chloe-soyeon
}
```

[그림 26] CueCode의 Spring Data MongoDB 코드

- @Document 어노테이션을 활용하여 DetectionAreaDocument 클래스를 MongoDB 컬렉션(detection_area)과 매핑함으로써 도메인 객체를 문서 형태로 저장할 수 있도록 구성함
- 데이터 접근 로직이 Repository 인터페이스 내에 캡슐화되어 있어 코드 수정 없이도 DB 전환이 용이함
- 이와 같은 Repository 추상화 방식은 비즈니스 로직과 데이터 저장소를 분리하여 유지보수성과 확장성을 향상시킴

(5) Build, Release, Run (빌드, 릴리즈, 실행)

- 환경 설정정보는 앱에 포함하지 않고 분리하여 관리함
- [그림 27]은 CueCode의 Build에 대한 내용임



[그림 27] CueCode의 Build 활용 내역

- Gradle 기반으로 애플리케이션을 빌드하고, 실행 가능한 JAR 파일을 생성함
- 버전별로 업로드하여 릴리즈를 관리하며, 필요 시 특정 태그 기준으로 롤백이 가능함

(6) Process (Stateless Process)

- 서비스는 각 요청이 이전 상태에 의존하지 않도록 상태를 서버에 저장하지 않는 Stateless 구조로 설계되어, 프로세스 간 완전한 분리가 가능함
- 사용자는 매 요청 시 JWT(JSON Web Token)를 통해 인증 정보를 전달하며, 서버는 별도의 세션 저장 없이 토큰만으로 사용자 식별을 수행함
- Spring WebFlux 기반 반응형 아키텍처를 적용하여 필터 체인 내에서 요청을 비동기적으로 처리하도록 구성함
- JWT 필터는 수신된 요청에서 토큰을 추출하고, 유효성 검증 이후에만 다음 필터 단계로 요청을 전달함
- 사용자의 인증 상태는 서버 세션에 저장되지 않고, 토큰에 캡슐화되어 클라이언트 측에서 전송되므로 완전한 상태 비저장(Stateless) 기반의 보안 구조를 구현함
- [그림 28]은 CueCode의 JWT 토큰 사용 내용임

[그림 28] CueCode의 JWT 토큰 사용 내용

- [그림 29]은 CueCode의 UserService모듈의 Swagger 사용 내역임

[그림 29] CueCode의 Swagger 사용

- 서비스 간 결합도를 낮추기 위해 RESTful API 기반 통신 구조를 적용함
- 각 마이크로서비스는 이전 요청의 상태에 의존하지 않는 Stateless 방식으로 독립적으로 요청을 처리할 수 있음
- JWT 필터는 수신된 요청에서 토큰을 추출하고, 유효성 검증 이후에만 다음 필터 단계로 요청을 전달함
- Swagger를 활용하여 각 마이크로서비스의 API 명세서를 자동화된 방식으로 관리하고, 서비스 간 통신 규격이 정확하게 준수되는지를 검증할 수 있음

(7) Port Binding (포트 바인딩)

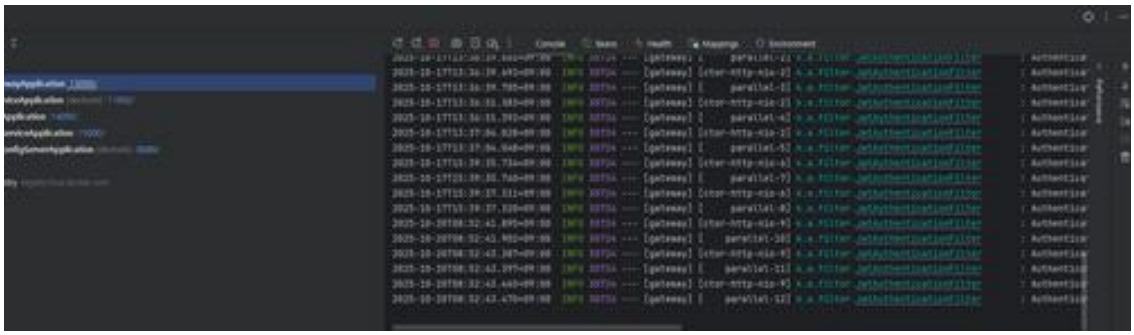
- 컨테이너 기반 웹 서비스는 Kubernetes 환경에서 동작하기 위해 포트가 동적으로 할당될 수 있도록 설계되어야 함
- [그림 30]은 CueCode의 Kubernetes 배포 및 포트 바인딩 내용임

```yaml
containers:
  - name: api-gateway-app
    image: cuecode.kr.ncr.ntruss.com/cuecode:api-gateway-latest
    imagePullPolicy: Always # 항상 최신 이미지를 가져오도록 설정
    ports:
      - name: http
        containerPort: 13000
      - name: management
        containerPort: 13001
```

[그림 30] CueCode의 Kubernetes 배포 및 포트 바인딩

(8) Concurrency (동시성)

- 클라우드 네이티브 어플리케이션 병렬처리를 지원해야함
- [그림 31]은 CueCode의 Kubernetes 배포 및 포트 바인딩 내용임



[그림 31] CueCode의 Spring Boot 병렬 처리 사용

(9) Disposability (폐기 가능)

- 서비스는 빠르게 시작되고 필요 시 즉시 종료될 수 있도록 설계되어야 함
- [그림 32]은 CueCode의 Actuator와 Graceful Shutdown 적용 내용임

[그림 32] CueCode의 Actuator와 Graceful Shutdown 사용

(10) Dev / Prod Parity(개발 및 운영 환경 일치)

- 애플리케이션은 동일한 코드베이스(Build 결과물)를 기준으로 배포되며, 개발 환경과 운영 환경의 구조가 일관되도록 유지됨
- [그림 33]은 CueCode의 기능별 환경 설정 내용임

[그림 33] CueCode의 기능별 환경 설정

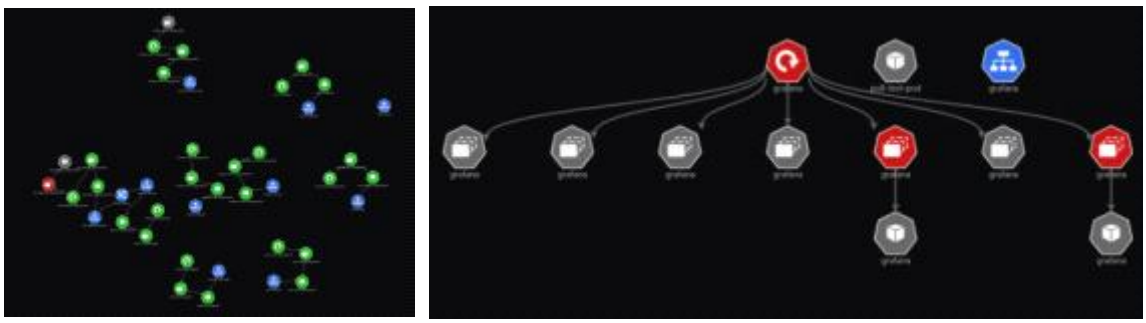
(11) Logging (로깅)

- 로깅은 애플리케이션 내부에서 파일 시스템에 기록하는 방식이 아닌, 모든 로그를 이벤트 스트림 형태로 표준 출력(STDOUT)으로 전달함으로써 운영 환경과 분리된 형태로 처리됨
- Fluent Bit은 각 마이크로서비스에서 표준 출력으로 발생한 로그를 수집하여 중앙 로깅 시스템으로 전달하는 경량 로그 포워더로 사용됨
- Grafana는 Fluent Bit을 통해 적재된 로그 데이터를 기반으로 서비스 동작 상태를 실시간 모니터링하며, 장애 발생 시 원인 추적 및 성능 분석 대시보드를 제공함
- [그림 34]은 CueCode의 Grafana 대시보드임

[그림 34] CueCode의 Grafana 대시보드

(12) Admin Process (관리 프로세스)

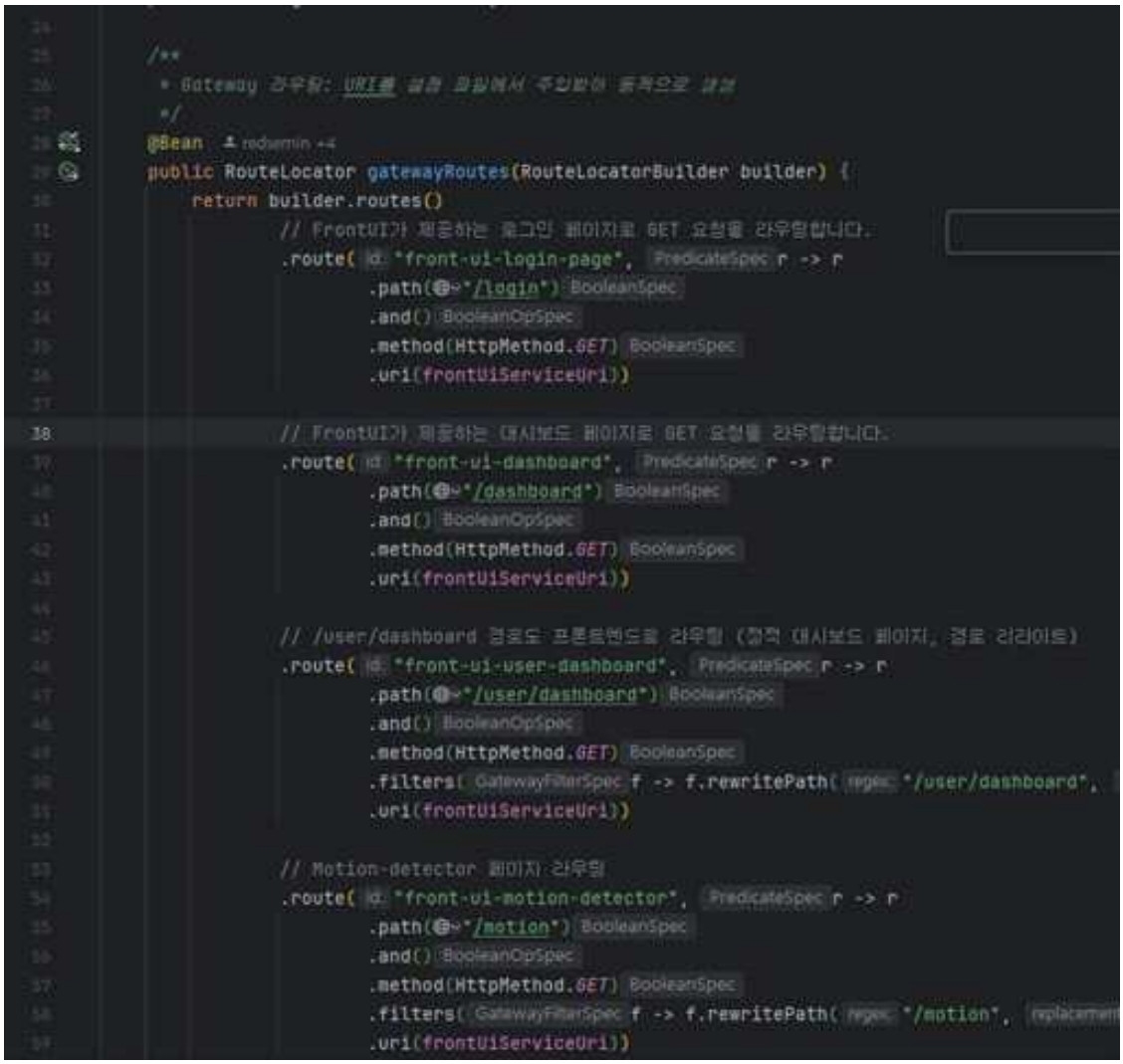
- 관리 작업은 애플리케이션 내부 로직과 분리된 상태에서 수행되며, Kubernetes 환경에서 독립적인 관리 명령(kubectl 명령어, HPA, Pod 재기동 등)을 통해 운영됨
- Kubernetes 관리 도구를 통해 Pod 상태 확인, 로그 조회, 컨테이너 재기동 등을 수행할 수 있음
- 관리자는 이를 통해 서비스별 리소스 사용량(CPU, 메모리), 네임스페이스별 배포 현황, Pod의 상태를 실시간으로 모니터링할 수 있음
- 또한 KubeView는 Pod 간 트래픽 흐름을 그래프 형태로 표시하여 서비스 간 의존 관계를 직관적으로 파악할 수 있음
- [그림 35]는 Kubenetes를 기반으로 관리되고 있는 CueCode 프로젝트를 나타냄



[그림 35] CueCode의 Kube View 대시보드

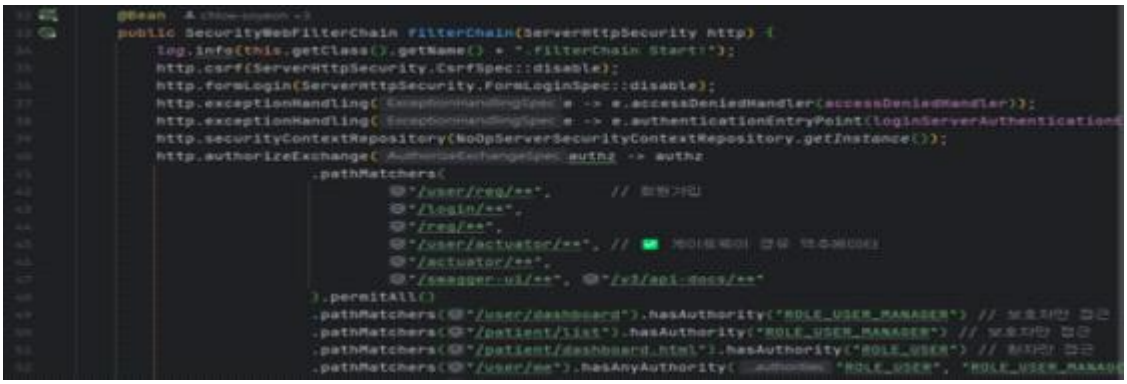
ㅇ 주요 기술 개발

- Spring Cloud Gateway 적용
  - API Gateway 구현을 위해 논블로킹 기반의 Spring Cloud Gateway 적용함
  - [그림 36]는 단일 진입점에서 각 마이크로서비스로 라우팅 처리하는 내역을 보여줌



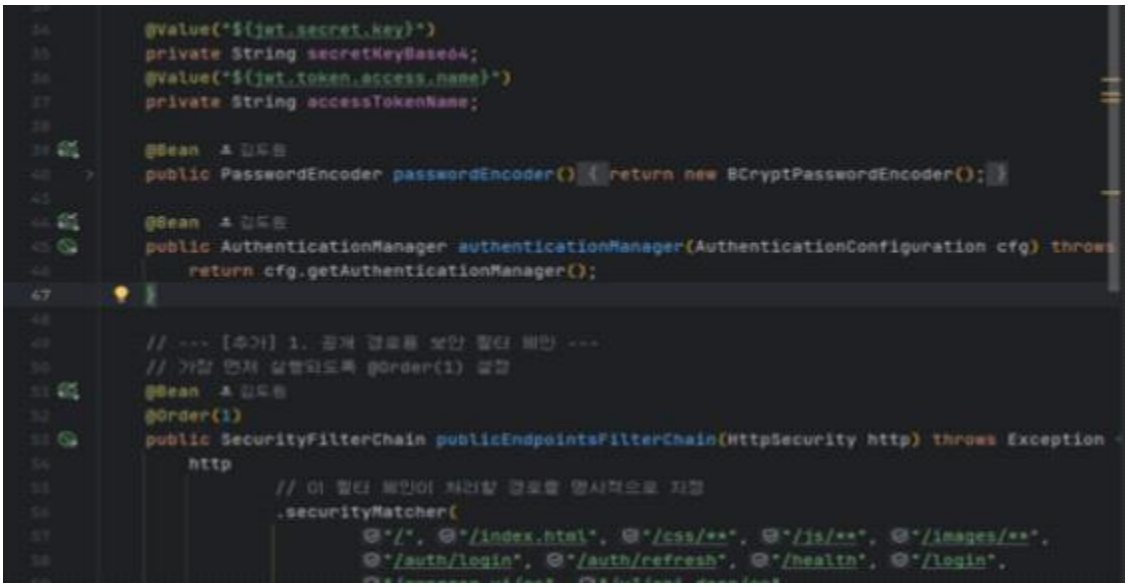
[그림 36] API Gateway 구현을 위한
논 블로킹 기반 Spring Cloud Gateway 적용 코드

- [그림 37]은 Security Config가 적용된 코드임



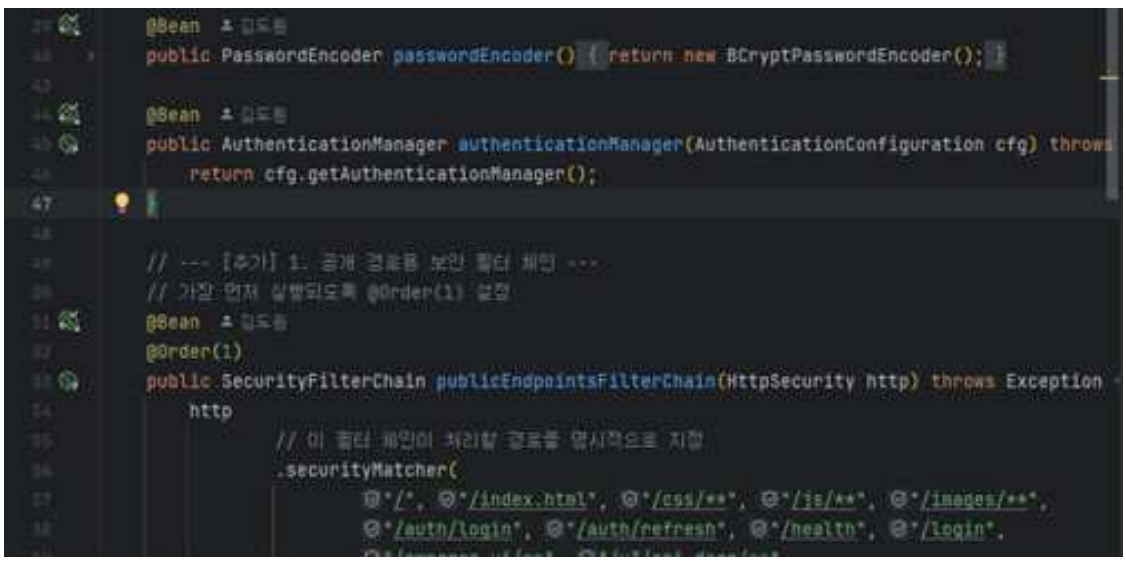
[그림 37] 각 서비스 접근 제어하는 Security Config 적용 코드

- Spring Security 적용
  - [그림 38]는 Gateway 레벨과 개별 서비스 레벨에서 인증·인가 정책을 일관되게 적용하여 권한 분리 및 보안 정책 시행을 용이하게 함



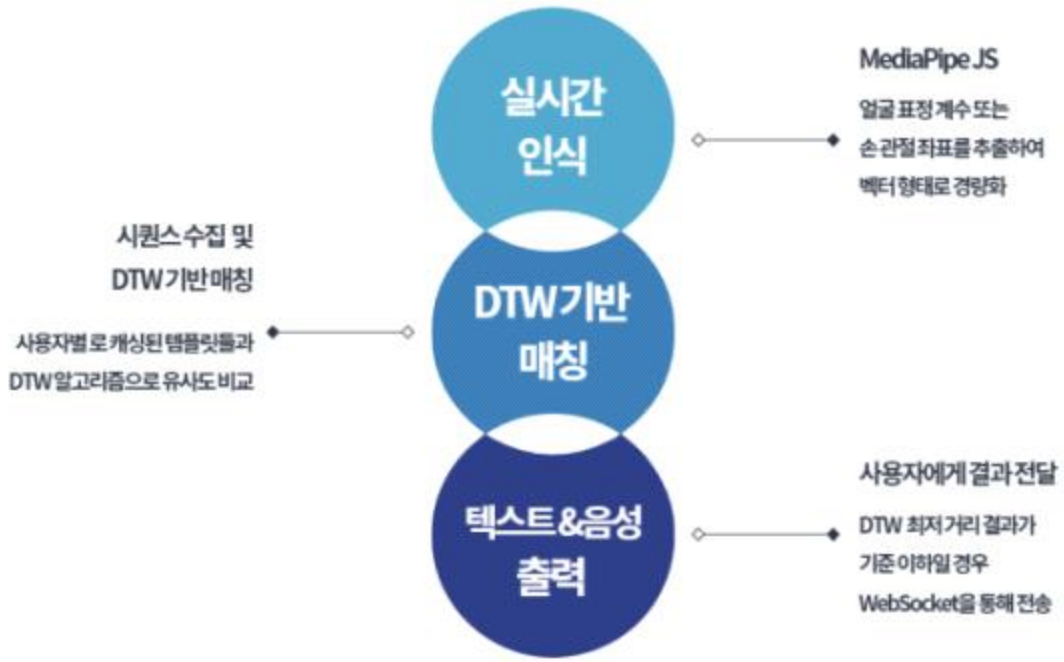
[그림 38] Spring Security 적용 코드

- JWT 기반 인증 처리
  - [그림 39]는 로그인 성공 시 사용자 정보를 기반으로 JWT를 생성하고, 요청 시 토큰 유효성을 검증함



[그림 39] JWT 기반 인증 처리

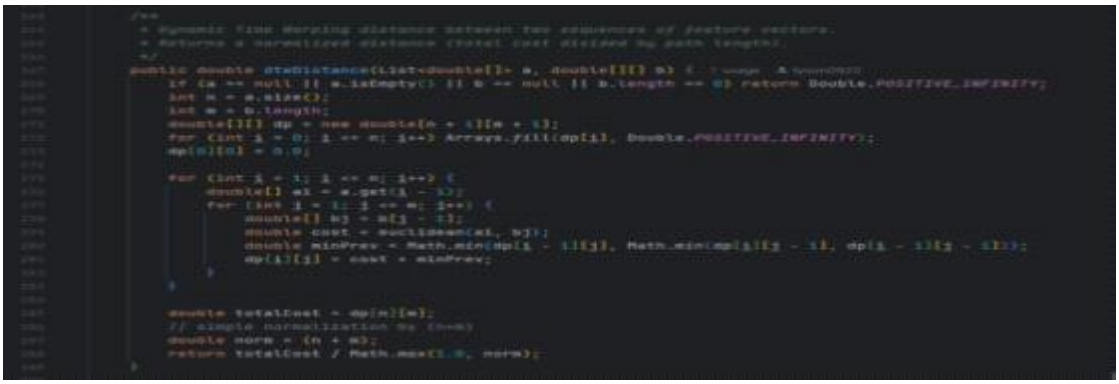
- 실시간 동작 감지 흐름
  - [그림 40]은 CueCode의 동작 데이터 실시간 처리 흐름을 나타냄



[그림 40] CueCode의 실시간 동작 데이터 처리

① 사용자가 웹 화면에서 실시간으로 동작(얼굴 표정, 손 동작 등)을 수행하면, 브라우저 내에서 WebAssembly 기반 MediaPipe Tasks (vision_bundle.js + WASM)가 이를 추론하여 특징 벡터를 추출함.
② 추출된 프레임 데이터는 WebSocket을 통해 API Gateway → Motion Service(Spring Boot) 로 전송됨
③ 완성된 시퀀스는 DTW(Dynamic Time Warping) 알고리즘을 통해사용자의 저장된 템플릿(Motion Record) 과 비교됨
④ DTW 매칭 결과가 임계값 이하일 경우, 응답을 WebSocket으로 즉시 클라이언트에 전송함

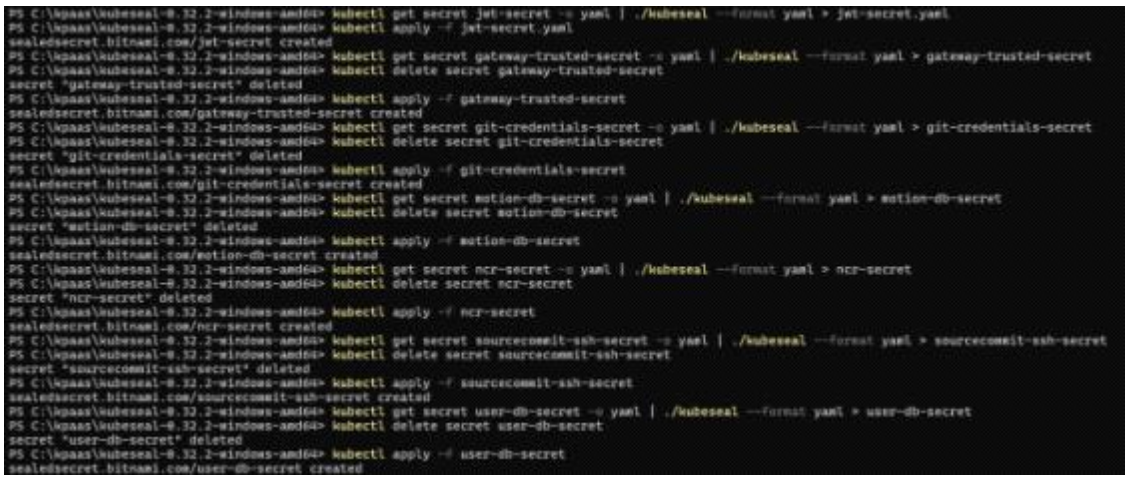
- [그림 41]는 CueCode의 DTW 알고리즘 코드 중 일부임



[그림 41] CueCode의 DTW 알고리즘 활용 코드

- WebSocket을 통해 전달된 실시간 특징 시퀀스는 매칭 프로세스에 진입함
- 매칭 대상은 해당 사용자(userId)의 캐시된 Motion 템플릿이며, 이 캐시는 MongoDB에서 불러와 Lazy-Loading 방식으로 사용자별 구성됨
- 얼굴 템플릿은 저장된 Blendshape Key들을 기준으로 정렬된 벡터 형태로 구성되고, 손 템플릿은 평탄화한 후 정규화하여 프레임 단위 벡터 시퀀스로 구성됨
- DTW(Dynamic Time Warping)는 두 시퀀스 간 프레임 길이가 다르더라도, 시간 왜곡을 허용하면서 최적의 정렬을 찾아 유사도를 비교하는 알고리즘임
- 가장 낮은 DTW score를 가진 템플릿이 매칭 후보가 되며, 특정 기준값 이하일 경우 동작이 인식된 것으로 판단됨

- Kubernetes SealedSecret 생성 및 보안 Secret 관리 자동화 과정
  - [그림 42]는 Secret을 SealedSecret으로 변환하고 배포하는 과정임



[그림 42] Secret을 SealedSecret으로 변환 및 배포

- 이 과정은 Kubernetes 클러스터 내의 민감 정보(Secret)를 Bitnami Sealed Secrets Controller를 통해 암호화(Encryption)하고 보안성을 유지한 상태로 배포하는 절차를 보여줌
- 각 Secret(jwt-secret, gateway-trusted-secret, git-credentials-secret, motion-db-secret, ncr-secret, sourcecommit-ssh-secret, user-db-secret)을 개별적으로 YAML 형식으로 추출함
- kubeseal CLI를 사용해 암호화(./kubeseal --format yaml) 함
- 암호화된 결과물은 SealedSecret 형태로 저장되어 버전 관리(GitOps)가 가능함
- 적용하여 보안 데이터가 자동 복원되도록 구성함이후 기존 Secret을 삭제하고, 새롭게 생성된 SealedSecret을 클러스터에 적용함
- 이러한 과정은 CI/CD 파이프라인에서도 반복적으로 수행되어, 보안 토큰·데이터베이스 자격증명·SSH 키 등 민감 정보가 안전하게 관리되도록 함

ㅇ 설계 산출물

- 메뉴 구성도

[그림 43] CueCode 메뉴구성도

- 간트 차트

[그림 44] CueCode 간트 차트

- Work Breakdown Structure (WBS)

WBS 코드	작업 내용	담당자	시작일	종료일
0.0.0	wbs작성	홍세민	2025-09-25	2025-10-30
Phase 1	공통 인프라 및 기반 설정	김도원	2025-09-25	2025-09-30
1.1	프로젝트 환경 설정	김소연	2025-09-25	2025-09-28
1.1.1	Git 레포지토리 전략 수립 및 브랜치 생성	김도원	2025-09-25	2025-09-28
1.1.2	각 서비스 기본 Spring Boot 프로젝트 생성 (UserService, MotionService 등)	김소연	2025-09-25	2025-09-28
1.2	클라우드 인프라 설계 (NCP)	김도원	2025-09-27	2025-09-30
1.2.1	VPC, Subnet 등 네트워크 환경 설계	김도원	2025-09-27	2025-09-30
1.2.2	NKS(Naver Kubernetes Service) 클러스터 설계	김도원	2025-09-27	2025-09-30
1.2.3	데이터베이스(MySQL, MariaDB 등) 서비스 설계	김도원	2025-09-27	2025-09-30
1.3	MSA 공통 모듈 구축	김소연	2025-09-25	2025-09-30
1.3.1	SpringConfigServer 구축 (설정 중앙 관리 및 Git 연동)	김소연	2025-09-25	2025-09-30
1.3.2	ApiGateway 구축 (라우팅 및 공통 인증/인가 필터 적용)	김소연	2025-09-25	2025-09-30
Phase 2	사용자 기능 개발 (UserService + FrontUI)	김소연,홍세민	2025-10-01	2025-10-10
2.1	백엔드 개발 (UserService)	김소연,홍세민	2025-10-01	2025-10-10
2.1.1	DB 설계 (user 테이블)	김소연	2025-10-01	2025-10-10
2.1.2	User Entity, Repository, DTO 구현	김소연	2025-10-01	2025-10-10
2.1.3	회원가입 API 개발 (POST /users/register, BCrypt 암호화, 중복체크)	김소연	2025-10-01	2025-10-10
2.1.4	로그인 API 개발 (JWT 발급, POST /users/login)	김소연	2025-10-01	2025-10-10
2.1.5	사용자 정보 조회/수정 API 개발 (GET/PUT /users/me)	홍세민	2025-10-01	2025-10-10
2.2	프론트엔드 개발 (FrontUI)	홍세민	2025-10-01	2025-10-10
2.2.1	환자/메니저 대시보드 개발	홍세민	2025-10-01	2025-10-10
2.2.1	환자 상세정보 모듈 개발	홍세민	2025-10-01	2025-10-10
2.2.1	메인/랜딩 페이지 (서비스 소개, 로그인/회원가입 버튼)	김소연	2025-10-01	2025-10-10
2.2.2	공통 레이아웃 (헤더, 푸터)	홍세민	2025-10-01	2025-10-10
2.2.3	회원가입 페이지 (폼 UI, 유효성 검사, API 연동)	김소연	2025-10-01	2025-10-10
2.2.4	로그인 페이지 (폼 UI, JWT 저장)	김소연	2025-10-01	2025-10-10
2.2.5	마이페이지 (정보 조회/수정 UI, API 연동)	김소연	2025-10-01	2025-10-10
2.3	인증/인가 연동	김소연	2025-10-05	2025-10-20
2.3.1	ApiGateway JWT 검증 필터 추가	김소연	2025-10-06	2025-10-20
2.3.2	FrontUI에서 JWT 포함 API 요청 처리	김소연	2025-10-05	2025-10-20
Phase 3	핵심 기능 개발 (MotionService + FrontUI)	김태승,김소연	2025-10-11	2025-10-20
3.1	백엔드 개발 (MotionService)	김태승	2025-10-11	2025-10-20
3.1.1	인식 기술 선정 (OpenCV, CLOVA 등)	김태승	2025-10-11	2025-10-20
3.1.2	DB 설계 (motion_mapping 테이블)	김도원	2025-10-11	2025-10-20
3.1.3	인식 처리 API 개발 (POST /motions/recognize)	김태승	2025-10-11	2025-10-20
3.1.4	커스텀 매핑 관리 API 개발 (CRUD)	김태승	2025-10-21	2025-10-31
3.2	프론트엔드 개발 (FrontUI)	김소연	2025-10-16	2025-10-25
3.2.1	인식 실행 페이지 (웹캠 연동, 인식 결과 표시)	김태승	2025-10-16	2025-10-25
3.2.2	커스텀 매핑 관리 페이지 (목록, 수정, 삭제, API 연동)	김태승	2025-10-16	2025-10-25
Phase 4	배포 및 운영	김도원	2025-10-21	2025-10-30
4.1	컨테이너화	김도원	2025-10-21	2025-10-30
4.1.1	각 서비스별 Dockerfile 작성	김도원	2025-10-21	2025-10-30
4.1.2	Naver Cloud Container Registry(NCR) 빌드 및 푸시	김도원	2025-10-21	2025-10-30
4.2	배포 (NKS)	김도원	2025-10-21	2025-10-30
4.2.1	Kubernetes Deployment, Service YAML 작성	김도원	2025-10-21	2025-10-30
4.2.2	ApiGateway Ingress YAML 작성	김도원	2025-10-21	2025-10-30
4.2.3	kubectl 또는 devspace 배포 자동화	김도원	2025-10-21	2025-10-30
4.3	CI/CD 구축	김도원	2025-10-21	2025-10-30

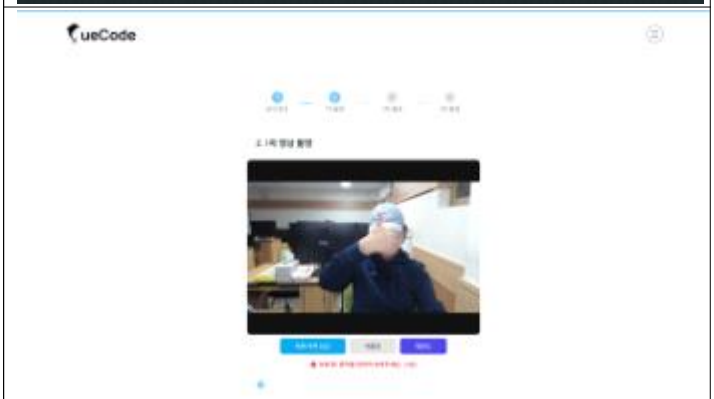
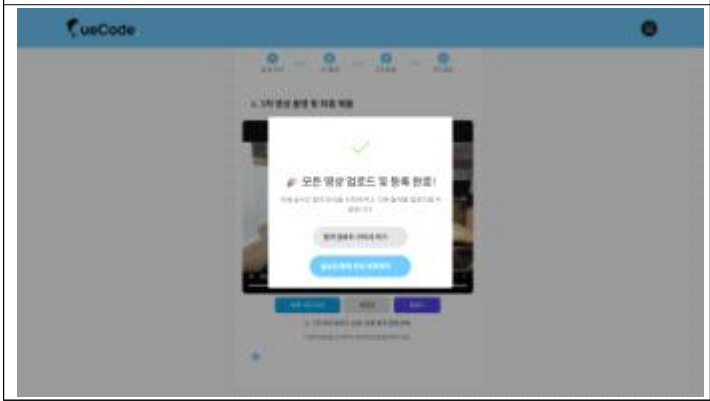
[그림 45] CueCode WBS

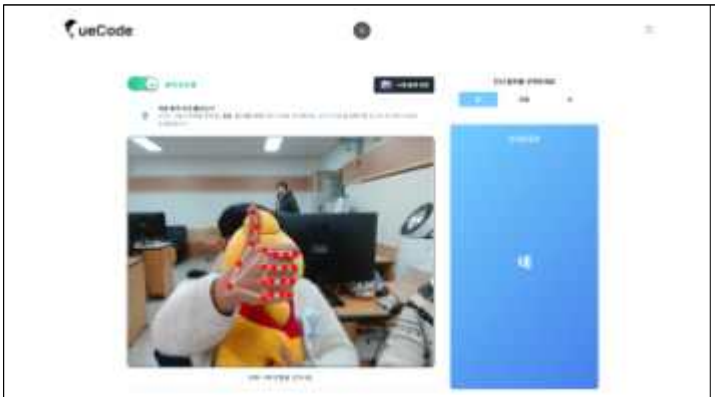
- 컬렉션 정의서

Collection 명	NO	Depth	칼럼ID	칼럼명	Type	Length	Decimal	PK	NOT NULL	UNIQUE
환자 정보	1	1	ID	환자 ID	STRING	50		*	*	*
	2	1	NAME	환자 이름	STRING	50			*	
	3	1	EMAIL	환자 이메일	STRING	255			*	
	4	1	PW	환자 비밀번호	STRING	100			*	
	5	1	MANAGERIDS	관리자 ID 목록	ARRAY (STRING)	-				
	6	1	MEDICALHISTORY	병력	STRING	500				
	7	1	MEDICATIONS	복용 약물	ARRAY (STRING)	-				
	8	1	ALLERGIES	알레르기	ARRAY (STRING)	-				
	9	1	_CLASS	Document 클래스	STRING	-			*	
관리자 정보	1	1	ID	관리자 ID	STRING	50		*	*	*
	2	1	NAME	관리자 이름	STRING	50			*	
	3	1	EMAIL	관리자 이메일	STRING	255			*	
	4	1	PW	관리자 비밀번호	STRING	100			*	
	5	1	PATIENTIDS	환자 ID 목록	ARRAY (STRING)	-				
	6	1	_CLASS	Document 클래스	STRING	-				
감지 영역	1	1	ID	환자 ID	STRING	50		*	*	*
	2	1	HAND	손	BOOLEAN	-			*	
	3	1	FACE	얼굴	BOOLEAN	-			*	
	4	1	BOTH	모두	BOOLEAN	-			*	
	5	1	_CLASS	Document 클래스	STRING	-			*	
동작	1	1	ID	OBJECT ID	STRING	-			*	
	2	1	USER_ID	환자 아이디	STRING	500		*	*	*
	3	1	PHRASE	의미	STRING	500			*	
	4	1	MOTION_TYPE	감지 대상	STRING	500			*	
	5	1	MOTION_DATA	동작 데이터	STRING	500			*	
	6	2	HAND_LANDMARKS	손 랜드마크	ARRAY (DOUBLE)	-				
	7	3	TIMESTAMP_MS	손 촬영 시점	INT64	-				
	8	2	FACE_BLENDSHAPES	얼굴/눈 랜드마크	ARRAY (DOUBLE)	-				
	9	3	TIMESTAMP_MS	얼굴/눈 촬영 시점	INT64	-				
	10	2	CREATED_AT	업로드 시점	DATE	-			*	
	11	1	DESRCIPTION	메모	STRING	500				
	12	1	_CLASS	Document 클래스	STRING	500			*	
위험 단어 감지	1	1	ID	OBJECT ID	STRING	-			*	
	2	1	USERID	환자 아이디	STRING	500		*	*	*
	3	1	PHRASE	의미	STRING	500			*	
	4	1	DETECTEDTIME	감지 시점	DATE	-			*	
	5	1	_CLASS	Document 클래스	STRING	-			*	

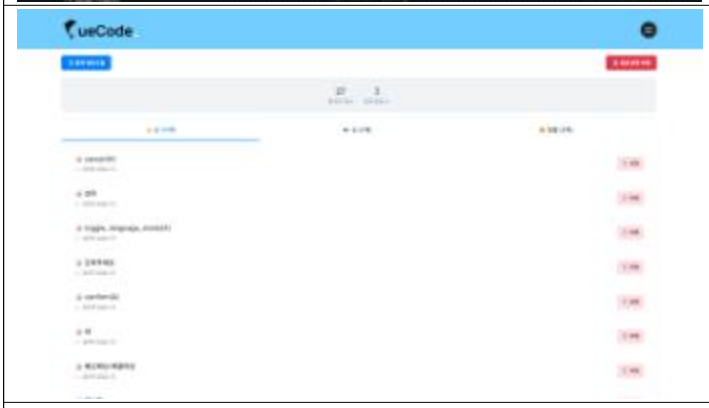
[그림 46] CueCode 컬렉션 정의서

○ 주요 기능 개발 완료 결과물 (화면)

<table>
<tr>
<td> 
</td>
<td>❍ 메인화면
- 로그인
- 회원가입
- 사용자 계정별 대시보드
- 프로젝트 소개
- 고객 소리함</td>
</tr>
<tr>
<td rowspan="2"> 
  </td>
<td>❍ 동작 업로드
- 감지 범위 선택
- 사용자 정의 의미
- 사용자 메모
- 1차 촬영 (3초간 동작 유지)
- 2차 촬영 (3초간 동작 유지)
- 3차 촬영 (3초간 동작 유지)
- 사용자 정의 의미</td>
</tr>
<tr>
<td>  </td>
<td>❍ 동작 업로드 완료
- 동작 업로드 이어서 하기
- 동작 인식 시작하기</td>
</tr>
</table>



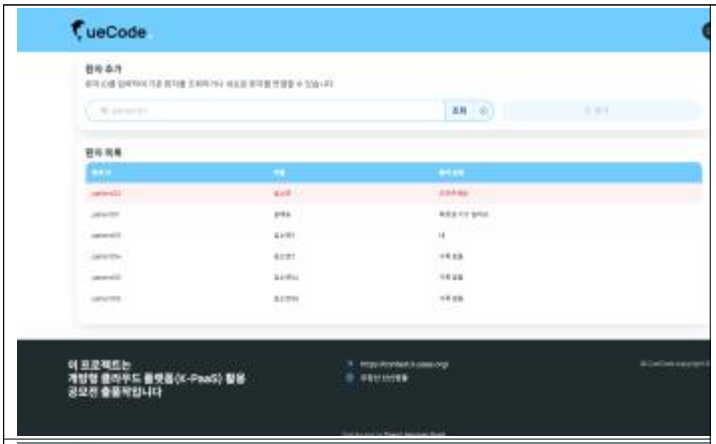
○ 동작 인식
- 동작 인식 시작 / 정지
- 동작 인식 시작하기
- 인식 범위 선택
- 사용자 동작 사전



○ 사용자 동작 사전
- 범위별 동작 조회
- 동작 개별 삭제
- 동작 일괄 삭제

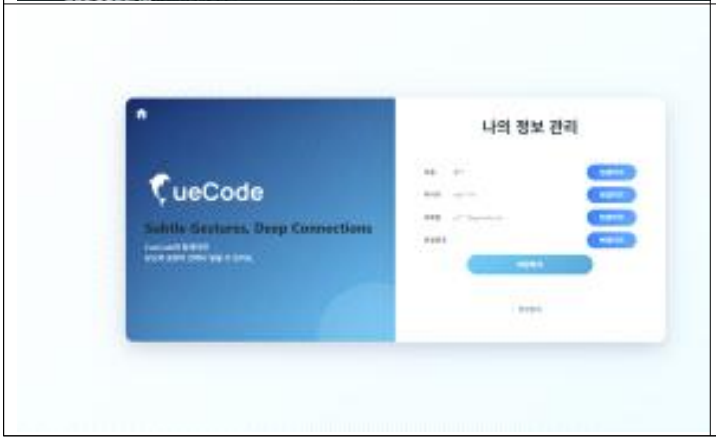
○ 로그인
- 서비스 이용을 위한 사용자 로그인

	❍ 회원가입 - 서비스 이용을 위한 사용자 회원 가입
	❍ 비밀번호 찾기 - 개인정보 보호를 위한 - 비밀번호 재설정
	❍ 고객의 소리함 - 고객 피드백 수용
	❍ 관리자 메인 화면



❍ 관리자 대시보드
- 환자 조회
- 환자 추가
- 환자 실시간 메시지 조회
- 환자 응급 메시지 조회
- 환자 상세 정보 조회

❍ 환자 대시보드
- 튜토리얼
- 실시간 동작 인식
- 동작 업로드
- 환경설정



❍ 나의 정보 관리
- 사용자 이름 수정
- 이메일 수정
- 비밀번호 수정
- 감지 범위 수정

ㅇ 전체 기능목록

구분	기능	설명
회원 관리	회원가입	사이트 사용을 위한 사용자 정보 등록
	로그인	사이트 이용을 위한 사용자 정보 로그인
	비밀번호 찾기	사이트를 사용하기 위한 비밀번호 찾기
	회원정보 수정	회원 정보 수정
	로그아웃	개인정보 보호를 위한 로그아웃
	회원탈퇴	개인정보 보호를 위한 회원 탈퇴
동작 업로드	영상 촬영	사용자의 동작을 촬영
	영상 업로드	영상 업로드 요청
	FastAPI 분석 요청 전송	분석 영역(예: face, hand) 정보를 포함해 FastAPI 서버에 분석 요청
	MediaPipe 기반 좌표 추출	Google MediaPipe Face/Hand 모델을 통해 프레임별 Face Blendshapes 또는 Hand Landmark 좌표를 자동 추출
	경량 데이터 구성	추출된 숫자 좌표 및 표정 계수 기반으로 경량화된 JSON만 반환
	Motion Document 생성	반환된 JSON을 Document 형태로 변환함
	MongoDB 저장	사용자 ID, 문장 구문(phrase), 모션 타입, 시점별 특징값을 포함한 구조화된 문서로 MongoDB에 저장됨

구분	기능	설명
동작 실시간 인식	실시간 동작 감지	MediaPipe JS를 이용해 실시간 Blendshape/관절좌표를 추출함
	특징 벡터 변환	얼굴은 Blendshape Key 정렬 기반, 손은 좌표를 평탄화하여 숫자 벡터로 전송함
	WebSocket 프레임 전송	100ms 간격으로 Motion Service에 실시간 전송됨
	시퀀스 구간화	1초 단위 또는 인식 모드 변경 시 시퀀스 종료를 알림
	사용자별 시퀀스 누적	WebSocket은 시퀀스를 프레임 단위로 수집 후 matchSequence() 호출 준비
	손 좌표 정규화	저장된 템플릿과 일치하도록 중심 이동 + RMS 스케일 정규화 수행
	템플릿 기반 DTW 매칭	실시간 시퀀스와 저장된 시퀀스를 DTW 알고리즘으로 비교
	DTW 점수 계산	최적 정렬 경로 기반 누적 거리 계산 후 최저 DTW 점수 결정
	최적 템플릿 선택	점수가 가장 낮은 phrase가 매칭된 사용자 동작으로 인식됨
	WebSocket 결과 응답	결과가 사용자 화면에 반환됨
	실시간 출력	브라우저는 화면에 문장을 출력하고 SpeechSynthesis로 음성 피드백 제공
	LangGraph 기반 다중 LLM 오케스트레이션 단계	다중 LLM을 통과한 검증된 문장만 최종 출력

6 기대효과

ㅇ 사회적 가치

- 소통 기회 확대를 통한 개인의 자율성 강화
  - 국내 뇌병변 장애인은 약 23만 명(보건복지부, 2024), 뇌졸중 환자는 약 63만 명(건강보험심사평가원, 2022), 파킨슨병 환자는 약 12만 명에 달함
  - 이들 중 상당수는 언어 표현의 제약으로 기본적인 의사소통이 어려워 일상생활 자율성이 제한되는 상황에 있음
  - CueCode는 이러한 문제를 근본적으로 해결하기 위해 환자의 가용한 신체 능력에 맞춘 동작을 알고리즘 기반으로 학습
  - 표준화된 동작을 강요하지 않으며 사용자 고유의 표현 방식을 존중하여 사용자는 의사표현이 보다 원활해지고 도움을 받는 상황에서도 자신의 의사를 명확히 전달할 수 있음

- 돌봄 제공자의 부담 경감
  - 보호자와 의료진은 돌봄 과정에서 환자의 다양한 의사 표현을 빠르고 정확하게 이해하기 위해 상당한 정신적·육체적 노력을 기울임
  - 이들 중 상당수는 언어 표현의 제약으로 기본적인 의사소통이 어려워 일상생활 자율성이 제한되는 상황에 있음
  - CueCode의 실시간 모니터링 대시보드는 사용자의 표현 데이터를 즉시 시각화하여 확인할 수 있도록 지원하여, 돌봄 제공자가 환자의 상태를 신속하게 파악할 수 있게 함
  - 이로써 돌봄 과정의 효율성이 향상되고, 응급상황에 대한 조기 대응이 가능하며, 보호자와 의료진의 정신적·육체적 부담이 경감됨

- 포용적 소통 문화 확산
  - CueCode는 컴퓨터, 태블릿, 스마트폰 등 기본적인 디지털 기기와 웹캠만으로 작동하므로, 특수학교·재활센터·요양시설 등에서 비교적 낮은 비용으로 쉽게 도입 가능함
  - 기술 도입을 넘어 다양한 소통 방식을 인정하고 존중하는 포용적 사회 분위기를 조성하는 데 기여함

ㅇ AI 기반 커뮤니케이션 혁신

- AI 활용형 AAC 혁신 생태계 조성
  - 기존 AAC 도구는 하드웨어와 복잡한 사용 과정으로 인해 보급에 한계가 있었음
  - <표 3>은 기존의 AAC와 비교하여 CueCode에서 제공하는 서비스임

〈표 3〉 CueCode의 차별성

서비스명	사용자 입력 방식	상징의 개인화	관리자 기능
CueCode	영상 인식	가능	O
MYAAC	클릭	가능	O
Smart AAC	터치, 스위치	불가능	X
	터치	불가능	X

- CueCode는 MediaPipe, Dynamic Time Warping(DTW) 모델을 융합한 AI 파이프라인을 순수 소프트웨어로 구현하여 이러한 한계를 극복함
- 특정 플랫폼에 종속되지 않아 스마트폰, 태블릿 등 다양한 디바이스에서 활용 가능함

ㅇ 공공 협력 및 서비스 확산

- 공공적 가치 실현
  - CueCode는 의사소통이 어려운 환자와 의료진 간의 상호 이해를 촉진함으로써, 의료 현장에서의 의사소통 부족으로 인한 오진·치료 지연 등 의료 오류를 예방할 수 있는 기반을 마련함
  - 이를 통해 진료 효율성 향상, 재진료율 감소, 응급 대응 시간 단축 등 의료기관의 운영 품질을 개선하고, 복지시설에서도 돌봄 서비스의 신속성과 정확성을 높일 수 있음
  - 기존의 의료·복지 현장은 청각장애인 등 비언어 사용자들이 필담·손짓·제3자 통역에 의존하는 구조적 한계로 인해 심각한 의사소통 불평등을 겪고 있음
  - 실제로 서울 지역 상급종합병원 14곳 중 수어통역사가 상주한 병원은 단 1곳뿐이며, 긴급 상황 시 즉각 대응이 어려워 환자의 안전과 치료권이 위협받는 실정임
  - [그림 45]는 CueCode의 사용 예시 사진임

[그림 47] CueCode의 사용 예시

- CueCode는 이러한 현실적 제약을 극복하기 위해, 표정·손동작 인식 기반의 AI 모델을 활용해 수어 데이터베이스를 구축하고자 함
- 청각장애인과 언어장애인이 언제 어디서나 자기 표현이 가능한 '휴대형·무제한 확장형 수어 보조 시스템'을 구현함
- 의료·복지·교육 현장 전반에서 활용 가능한 통합형 커뮤니케이션 인프라로 발전할 수 있음
- 나아가, CueCode는 수어 학습 데이터의 개방형 공유(Open Dataset) 를 추진하여 정부·지자체·연구기관이 함께 참여하는 공공 협력 생태계 조성을 목표로 함
- 이를 통해 장애·질병·언어 장벽으로 인한 의사소통 격차를 해소하고, 사회 전반의 디지털 포용성과 접근성 향상을 실현할 수 있음
- 더 나아가 CueCode는 의료·복지 현장에 국한되지 않고, 교육기관·공공청사·지자체 민원센터 등 다양한 공공 서비스 영역으로 확장될 수 있음
- 궁극적으로 CueCode는 기술 혁신을 통한 사회적 약자 보호, 공공 협력 강화, 그리고 포용적 사회 실현이라는 세 가지 축을 중심으로 지속 가능한 디지털 복지 생태계를 만들어갈 예정