AI 챗봇 프로젝트

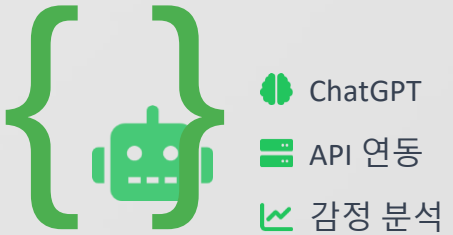
감정 분석 기반 AI 상담 챗봇 시스템

Emotion Analysis-based AI Counseling Chatbot System

팀명 및 팀원

C조

20252364 강한서 20251236 박주언



프로젝트 개요 및 팀 구성

프로젝트 개요

감정 분석 기반 AI 상담 챗봇 시스템

사용자의 발화를 분석하여 감정 상태를 파악하고, 해당 감정에 적합한 상담을 제공하는 AI 챗봇 시스템 개발

- 사회적 요구에 부응한 심리 상담 접근법
- 최신 AI 기술을 활용한 개인화 상담 서비스

프로젝트 동기

- 심리적 문제를 겪고 있는 사람들이 전문가에게 쉽게 접근하지 못하는 문제 해결
- 인공지능을 활용한 접근성 향상과 상담의 편의성 제공
- 감정 분석을 통한 사용자 상태 파악과 적절한 상담 제공

팀 구성

팀장
팀장 역할 담당

개발자
AI 모델 개발 및 시스템 구현

디자이너
UI/UX 디자인 및 사용자 경험 향상

기획자
프로젝트 기획 및 요구사항 분석

배경 및 필요성

온라인 텍스트 데이터의 감정 분석 필요성

대량의 감정 표현 데이터

온라인 채팅, SNS, 포럼 등에서 사용자들은 다양한 감정 상태를 텍스트로 표현하고 있습니다.

감정 인식 부족

사용자들의 감정 표현을 자동으로 파악하고 분석하는 기술적 도구의 부족.

가치 창출 가능성

감정 분석을 통해 사용자의 감정 상태를 파악하고, 이에 맞는 적절한 조치를 제안할 수 있습니다.

AI 기반 상담 서비스의 사회적 요구

감정 데이터 분석 결과

불안/걱정
가장 빈번한 감정 표현

의문/확인
지식적 필요성 높음

건강/체력
신체적 증상에 대한 걱정

관계/상호작용
인간적 상황에 대한 질문

사회적 가치

- 전문가 리소스 부족으로 인한 상담 접근성 문제 해결
- 개인의 감정 건강에 대한 조기 발견 및 개입 기회 제공
- AI 기술을 활용한 개인화된 감정 조절 도구로의 가치 창출

목표 및 범위

프로젝트 목표

감정 분석 모델 개발
사용자의 발화를 분석하여 감정 상태를 정확히 파악

맞춤형 상담 기능 구현
사용자의 감정 상태에 따른 맞춤형 상담 제공

지속적인 학습 및 개선
사용자 피드백을 통해 시스템의 성능 향상

프로젝트 범위

발화 분석
사용자의 언어, 톤, 문맥 분석

감정 인식
불안, 슬픔, 분노 등 감정 상태 파악

상담 제공
맞춤형 상담 내용 및 권장사항 제안

사용자 관리
사용자 프로필 관리 및 기록 보관

기대 성과

사용자 측면

- 감정 상태를 정확히 파악하고 이해하는 데 도움
- 맞춤형 상담으로 문제 해결에 도움

개발자/연구자 측면

- 효율적인 감정 분석 모델 개발
- 사용자 중심의 AI 챗봇 개발 경험

주요 기술 개요

ChatGPT API

OpenAI의 대형 언어 모델

사용자의 발화를 분석하고, 감정 상태에 따라 적합한 답변을 생성합니다. 챗봇의 대화 흐름을 이어가며, 사용자와의 자연어 대화를 가능하게 합니다.

- 대화식 인터페이스 구현

자연어 처리

Natural Language Processing

사용자의 발화를 구조화된 정보로 분석하고, 의미를 파악합니다. 문맥을 이해하고, 사용자의 의도를 캡쳐하여 적절한 답변을 생성합니다.

- 의도 인식 및 문맥 이해

감정 분석 알고리즘

Emotion Analysis Algorithm

사용자의 발화에서 감정 상태를 감지하고 분석합니다. 텍스트, 톤, 단어 선택 등 다양한 요소를 분석하여 사용자의 감정을 파악합니다.

- 감정 상태 파악 및 분류

프롬프트 엔지니어링

Prompt Engineering

AI 모델에 주입되는 질문과 지시문을 전문적으로 설계합니다. 사용자의 감정 상태에 따라 적절한 프롬프트를 선택하여, 사용자와의 효과적인 대화를 촉진합니다.

- 효과적인 프롬프트 설계

ChatGPT API | NLP | 감정 분석 | 프롬프트 | 상담 답변

요구사항 분석

기능 요구사항

발화 분석
사용자의 발화를 분석하여 감정 상태를 파악

감정 인식
화, 슬픔, 불안, 행복 등 다양한 감정 상태 인식

응답 생성
인식된 감정에 적합한 상담 내용 및 조언 생성

대화 기록 관리
사용자의 대화 내용을 저장하고 분석

비기능 요구사항

성능
빠른 응답 시간과 정확한 감정 인식 정확도

보안성
사용자 대화 내용 보호 및 개인정보 보안

확장성
사용자 증가에 따른 시스템 확장성

안정성
지속적인 서비스 제공과 오류 처리

타겟 사용자 정의

대학생
학교 생활과 관련된 스트레스와 걱정을 가진 사용자

직장인
업무 스트레스와 생활 균형을 맞추는 사용자

가족관계에 어려움
가족, 연인, 친구와의 관계에서 고충을 겪는 사용자

건강에 대한 걱정
건강에 대한 불확실성과 걱정을 가진 사용자

시스템 아키텍처

시스템 구조 개요

사용자 인터페이스
웹/모바일 챗봇

대화 관리 모듈
질의/응답 처리

감정 분석 엔진
발화 분석/감정 추출

데이터 관리 모듈
사용자/대화 데이터 저장

AI 챗봇 엔진
응답 생성/상담 제공

데이터 흐름도

사용자 발화
텍스트/음성 입력

→ NLP 처리
의미/의도 파악

→ 감정 분석
감정 상태 추출

→ 응답 생성
맞춤형 콘텐츠/답변

→ 사용자 응답
사용자 인터페이스 출력

API 연동 구조

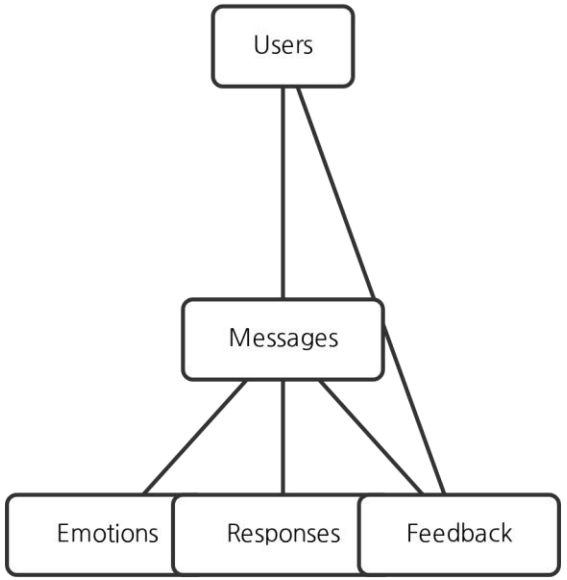
ChatGPT API
대화 생성 및 관리
지속적인 챗봇 대화 흐름 관리

감정 분석 API
사용자 발화 감정 추출
텍스트/음성 기반 감정 분석

외부 데이터 API
사용자 프로필/기록
사용자 맞춤화된 콘텐츠 제공

데이터베이스 및 기술 스택

ER 다이어그램



테이블 구조

Users
사용자 정보를 저장하는 테이블
id, name, age, gender, created_at

Messages
대화 내용을 저장하는 테이블
id, user_id, content, created_at

Emotions
감정 분석 결과를 저장하는 테이블
id, message_id, emotion, score, created_at

Responses
시스템의 답변을 저장하는 테이블
id, message_id, response, created_at

Feedback
사용자의 피드백을 저장하는 테이블
id, user_id, message_id, rating, created_at

개발 환경

Python
백엔드 로직 및 AI 모델 연동

JavaScript
프론트엔드 개발

PostgreSQL
관계형 데이터베이스

ChatGPT API
지능형 대화 생성

Emotion API
감정 분석 서비스

Flask
웹 서버 프레임워크

UI/UX 설계 개요 및 목표 사용자 정의

사용자 중심 설계 방향과 페르소나 정의 (9/14)

설계 개요

사용자 중심 설계 원칙

감정 분석 기반 AI 상담 시스템에서 사용자의 감정적 상태와 필요를 최우선으로 고려

- 감정 상태에 따른 반응성 있는 인터페이스
- 지나치게 복잡하지 않은 직관적 대화 흐름
- 사용자 프라이버시와 감정적 안전 보장

목표 사용자 그룹

20-30대 대학생/직장인 : 일상 스트레스와 정서적 지원 필요

40-50대 중년층 : 가족/직장 관계 및 건강 관련 상담

심리상담 전문가 : 보조 도구로서 활용 및 모니터링

페르소나 1

김지현, 28세

IT 기업 마케팅 담당자

배경 및 상황:

업무 스트레스와 경쟁 환경에서 번아웃을 경험 중. 대면 상담은 시간적 여유가 없어 어려움.

목표:

- 일상 스트레스 관리와 감정 조절
- 직장 내 인간관계 개선을 위한 조언
- 자기 효능감 향상 및 업무 능률 개선

니즈:

- 언제든지 접근 가능한 심리 상담 서비스
- 개인정보 보호와 비밀 유지
- 직관적이고 빠른 응답

페르소나 2

박민수, 45세

중학교 교사

배경 및 상황:

학생 지도와 가족 부양에 따른 이중 부담으로 불안과 우울감 경험. 주변에 감정 표현을 어려워함.

목표:

- 불안과 우울 증상 완화
- 일과 가정 사이의 균형 찾기
- 학생 지도에 대한 새로운 접근법 모색

니즈:

- 익명성이 보장된 상담 환경
- 단계별 감정 관리 가이드
- 실용적이고 구체적인 조언

화면 구조 설계

주요 화면 구성도 및 정보 구조(IA) (10/14)

앱 화면 구조도

로그인/가입

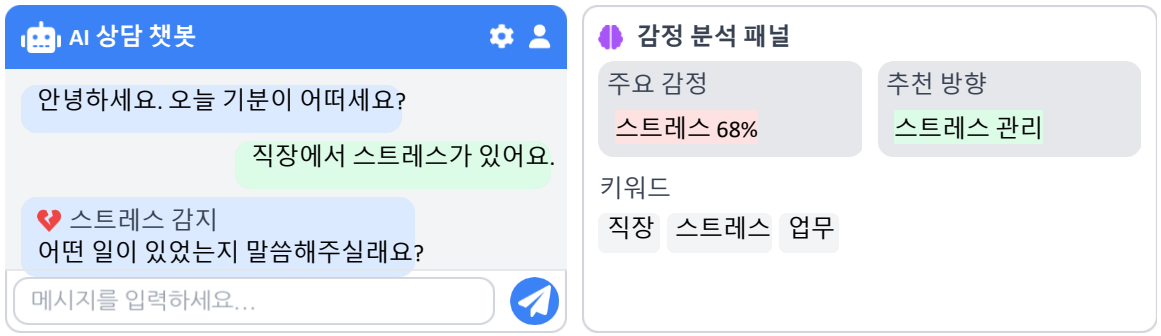
홈 대시보드

상담 채팅

감정 분석

상담 보고서

챗봇 인터페이스



정보 구조(IA)

서비스 IA 구조

V1.0

사용자 관리	상담 기능	데이터 & 설정
로그인	감정 평가	상담 기록
회원가입	일상 대화	분석 보고서
프로필 설정	집중 상담	알림 설정

앱/웹 화면 흐름도

- 사용자 인증 후 홈 대시보드로 이동
- 감정 상태에 따른 맞춤형 상담 화면 구성
- 대화 흐름에 따른 동적 UI 업데이트
- 감정 분석 결과를 시각적으로 표시
- 상담 히스토리와 통계 데이터 접근성 확보

화면 간 자연스러운 전환으로 사용자 경험 최적화

정보 구조 특징

접근성
직관적 내비게이션으로 필요한 기능에 빠르게 접근

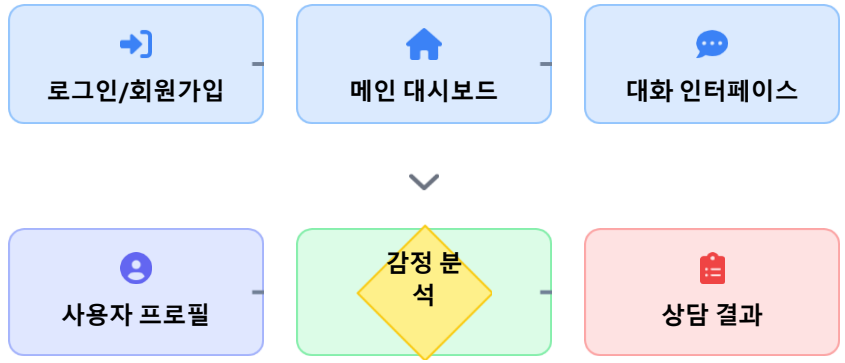
일관성
모든 화면에서 일관된 UI 요소와 상호작용 패턴 유지

확장성
새로운 기능 추가를 고려한 모듈식 구조 설계

사용자 흐름(Flow) 설계

화면 간 이동 흐름도 및 주요 동작 시나리오 (11/13)

주요 화면 간 이동 흐름도



흐름도 범례:

기본 화면 사용자 관련 결정 분기점 결과 화면

주요 화면 이동 흐름 설명

1. 로그인/인증 → 메인 대시보드: 사용자는 로그인 후 자신의 감정 상태와 이전 상담 요약이 표시된 대시보드로 이동
2. 대시보드 → 대화 인터페이스: 사용자는 상담 시작 버튼을 통해 AI 상담사와의 대화 인터페이스로 이동
3. 대화 중 → 감정 분석: 사용자의 입력을 실시간으로 분석하여 감정 상태를 파악하고 적절한 응답 생성
4. 감정 분석 → 상담 결과: 상담 종료 시 대화 요약과 맞춤형 조언이 포함된 결과 화면 제공
5. 상담 결과 → 프로필: 상담 내역과 감정 변화 추이가 사용자 프로필에 저장되어 장기적인 상태 관찰 가능

주요 사용자 시나리오

1 직장 스트레스 상담 시나리오

목표: 업무 스트레스 관리 및 직장 내 인간관계 개선

사용자: 28세 직장인 김지현(페르소나 1)

- → 대화 시작:퇴근 후 앱에 접속하여 오늘 있었던 상사와의 갈등 상황에 대해 대화 시작
- → 감정 인식:AI가 사용자의 텍스트에서 스트레스와 분노 감정을 감지하고 표시
- → 맞춤형 질문:상황에 대한 구체적인 감정과 생각을 끌어내는 질문으로 심층 대화 유도
- → 해결책 제시:스트레스 관리 기법 및 직장 내 갈등 해소를 위한 커뮤니케이션 방법 안내
- → 세션 종료:대화 요약 및 내일 실천할 수 있는 구체적인 행동 제안

2 일-가정 균형 찾기 시나리오

목표: 직장과 가정 사이의 균형 유지 및 불안감 해소

사용자: 45세 교사 박민수(페르소나 2)

- → 일정 확인:사용자는 업무 및 가족 일정을 시스템에 등록하여 과부하 지점 확인
- → 감정 추적:정기적으로 감정 상태를 기록하고 일정과의 연관성 분석
- → 갈등 상황 토론:학생 지도와 가족 행사가 겹치는 특정 상황에 대해 AI와 대화
- → 우선순위 설정:가치 기반 의사결정을 돕는 질문을 통해 우선순위 정리
- → 행동 계획:균형 잡힌 생활을 위한 주간 계획표와 스트레스 관리 전략 수립

감정 추적

이번 주 감정

안정적 개선 중

시나리오 결과

사용자는 감정 추적과 맞춤형 조언을 통해 일과 가정의 균형을 점차 찾아가고, 우선순위 설정을 통해 불필요한 스트레스 요인을 줄이며 자기 효능감 향상.

사용성 테스트 계획

테스트 방법 및 지표 (12/13)

테스트 방법론

사용자 테스트

실제 사용자들이 직접 시스템을 사용하며 과제 수행. 대화 인터페이스와 감정 인식 기능에 대한 사용성 평가

20-50대 사용자 그룹별 10명씩

전문가 평가

UI/UX 전문가와 심리상담 전문가가 인터페이스와 상담 접근 방식을 검토. 휴리스틱 평가 적용

UX 전문가 5명, 심리상담사 3명

A/B 테스트

다양한 UI 디자인과 상호작용 패턴을 비교 테스트하여 최적의 대화 방식과 감정 피드백 UI 발견

베타 테스터 그룹 100명 대상

종단 연구

장기간(4주) 시스템 사용을 통해 적응도, 지속적 활용도, 그리고 실제 감정 상태 개선 효과 측정

스트레스 관리 필요 그룹 30명

테스트 프로세스

1단계	2단계	3단계	4단계	5단계
사전 준비 및 계획 수립	파일럿 테스트	본 테스트 실시	데이터 분석	결과 반영

핵심 테스트 시나리오

시나리오 1: 초기 감정 상태 평가

사용자가 시스템에 처음 접속하여 감정 상태를 파악하는 초기 대화 진행 테스트

중점: 대화 자연스러움, 감정 인식

소요: 약 5분

시나리오 2: 스트레스 관리 세션

직장 스트레스 상황을 제시하고, 사용자가 AI와 대화하며 해결책을 모색하는 과정 테스트

중점: 응답 관련성, 사용자 만족도

소요: 약 10분

시나리오 3: 감정 변화 추적

여러 상담 세션 후, 사용자가 감정 상태 변화와 진행 상황을 확인하는 기능 테스트

중점: 정보 가시성, 통찰 유용성

소요: 약 7분

1단계: 전문가 평가	2단계: 사용자 테스트	3단계: A/B 테스트
2023년 1월 1~2주	2023년 1월 3주~2월	2023년 2월

사용성 평가 지표

효율성 지표

- 작업 완료 시간: 주요 시나리오 수행 시간
- 클릭/터치 수: 목표 달성까지 필요한 상호작용
- 오류 발생률: 사용자 경험 오류 비율

효과성 지표

- 작업 성공률: 과제 완료 비율
- 감정 인식 정확도: 감정 파악 정확도
- 상담 관련성: 제공된 조언의 적합도

만족도 지표

- SUS 점수: 표준화된 사용성 만족도
- NPS: 추천 의향 측정 지수
- 감정적 반응: 사용 중 감정 상태

효과 측정 지표

- 감정 상태 변화: 사용 전후 개선도
- 지속 사용률: 4주간 활성 사용 빈도
- 행동 실천율: 조언 실천 비율

전문가 평가 기준

1. 가시성 (Visibility)
   시스템 상태 및 감정 분석 결과의 명확한 표시
2. 자연스러움 (Naturalness)
   대화 인터페이스의 자연스러운 흐름과 응답 품질
3. 일관성 및 신뢰성 (Consistency)
   UI 요소와 상호작용 패턴의 일관된 적용

테스트 도구 및 방법

- 시선 추적 (Eye Tracking)
  UI 요소 주목도 및 정보 탐색 패턴 분석
- 발성 사고법 (Think Aloud)
  사용자가 생각을 소리내어 표현하며 테스트
- 행동 관찰 (Observation)
  상호작용 행동 및 비언어적 반응 녹화 분석

데이터 분석 및 활용 방안

정량적 분석

작업 시간, 성공률, SUS 점수의 통계적 분석으로 객관적 성과 측정

정성적 분석

사용자 피드백 및 행동 관찰을 통한 심층적 인사이트 도출

반영 프로세스

발견된 문제점 우선순위화 후 디자인 개선에 반복적 적용

기대 효과 및 향후 계획

프로젝트 완료 후 기대 효과와 서비스 확장 및 개선 방안 (13/17)

기대 효과

접근성 향상

AI 챗봇을 통해 전문가와의 상담 접근성을 높이고, 사용자들의 감정 문제를 조기에 발견합니다.

맞춤형 상담

사용자의 발화를 분석하여 개개인의 감정 상태와 특성에 맞춘 상담을 제공합니다.

마케팅 및 교육

AI 챗봇 시스템을 통해 정신건강의 중요성에 대한 인식을 높이고, 건강한 감정 관리 교육을 제공합니다.

데이터 기반 분석

사용자 데이터를 통한 감정 건강 통계적 분석을 수행하고, 사회적 문제점 발견 및 정책 제안에 기여합니다.

향후 계획

서비스 확장 기획

- 다양한 채널(웹, 모바일, SNS)에서의 접근성 향상 및 통합 서비스 제공
- 다국어 및 지역화된 콘텐츠 제공 전문화 및 텍스트 기반 성공률 개선
- 엔터테인먼트 및 교육적 콘텐츠 확장으로 사용자 간극 단절 방지

기술 개선 주안점

- 감정 상태 세분화 및 다양한 감정 인식 알고리즘 개선
- 대화맥락을 고려한 지수화된 감정 지원 및 반응 글머리 기법
- 사용자 피드백을 통한 지속적으로 발전하는 인공지능 모델 생성

AI 챗봇 프로젝트

API 연동 구조 개선

필요 API 및 데이터 처리 방향

감정 분류 API

텍스트 기반 사용자 발화를 분석하여 감정 분류 결과 도출

패턴 분석 API

대화 내 사용자 언어 풍성도 분석 및 감정 패턴 감지

대화 맥락 분석 API

AI 대화 환경 및 맥락 기반 응답 발생 조건을 파악

개인화 추천 API

가입자별 개인화된 챗봇 상담응답 및 서비스 추천

API 연동 최적화 방안

임베딩 모델 통합

단어/문장 임베딩을 사용한 유사 표현 및 의미 색인 기반 고성능 구현 실현

인퍼런스 최적화

함수 레벨의 코드 인프런싱을 통한 배포 환경 제로 네트워크 인프라 최적화 실현

APM/모니터링 조합

API 응답 지연 관리/감정 분석 범위 최적화를 통한 실시간 HTTP 비동기 호출 효율화

AI 챗봇 프로젝트

주차별 일정표

1주차

- 프로젝트 주제 선택 및 조사
- 프로젝트 특징 및 목표

2주차

- 기획서 작성 및 역할 분담 최
- 종 기획서 및 개발 계획 완성

3주차

- 공개 분석 데이터셋 정도

4주차

- 감정 분석 모델 학습 및 ChatGPT API 캐스트 시작
- 감정 평가 + 모델링 작성

5주차

- 감정 분석 + 대화 설정
- 감정(Chart.js) 및 UI
- 통합 봇 MVP

6주차

- 사용자 테스트 및 이에 대한 응답을 작성하고 발표
- 최종 시연 및 의견

AI 챗봇 프로젝트

개발 현황 및 개선점

현재 완성된 기능

- 완료 기본적인 챗봇 프레임워크 구현 (text based, 일부 API 연동)
- 완료 텍스트 기반 사용자 질문 응답 처리
- 완료 Emotion Classification을 위한 공유 원형 UI 구성

개선이 필요한 부분

- 개선 감정 분석 API 연동 실패 (standard speech_to_text & emotion_classifier API)
- 개선 실시간 감정 모니터링 기능 구현 필요성 높음
- 개선 다층 감정 분류 필요 (phrase 단위, persistence based 감정)

핵심 기능 개선점

실시간 감정 모니터링
실시간으로 사용자 발화의 감정 변화를 추적하고 분석

다층 감정 분류
감정의 심층 분석 및 위험 신호 조기 감지 기능

위험 신호 감지
안전과 적절한 대응을 위한 위험 신호 감지 알고리즘

개인화된 응답 생성
사용자 특성이 반영된 맞춤형 상담 응답 제공

```
import http.server
import socketserver
import json
import os
from http import HTTPStatus
from typing import List, Tuple

# --- [0] 안전 및 공감 답변 목록 ---
SENSITIVE_KEYWORDS = ["자살", "자해", "죽고 싶다", "살기 싫다", "뛰어내리", "목숨을 끊", "스스로 해치"]
SAFETY_RESPONSE = (
    "현재 당신이 매우 힘든 상황에 있다는 것을 이해합니다. 하지만 당신은 혼자가 아닙니다.<br>"
    "긴급한 도움이 필요할 경우, 즉시 전문 기관에 연락하세요.<br>"
    "<br>"
    "<strong>[자살 예방 상담 전화]</strong><br>"
    "국가번호 없이 **109** (24시간)<br>"
    "<strong>[정신건강 상담 전화]</strong><br>"
    "국가번호 없이 **1577-0199** (24시간)<br>"
    "저희 AI 상담사는 전문 치료를 대체할 수 없습니다. 안전이 최우선입니다."
)

EMOTION_RESPONSES = {
    "우울": "지금 많이 우울하고 힘드시죠. 괜찮아요, 제가 당신의 이야기를 들어드릴 준비가 되어 있어요. 무엇이 당신을 그렇게 힘들게 하나요?",
    "힘들어": "힘들다는 그 마음, 제가 충분히 이해합니다. 잠시 멈추고 쉬어가도 괜찮아요. 어떤 점이 당신을 가장 지치게 하는지
말씀해주시겠어요?",
    "짜증": "짜증나고 답답한 마음이 크시군요. 그 감정은 지극히 자연스러운 것입니다. 그 감정의 원인에 대해 조금 더 이야기해 주시겠어요?",
    "걱정": "걱정 때문에 마음이 편치 않으시군요. 어떤 일에 대해 걱정하고 계신가요? 함께 이야기하면서 걱정을 덜어봐요.",
    "불안": "불안한 마음이 당신을 짓누르고 있군요. 불안함을 느끼는 것은 정상입니다. 지금 가장 불안하게 만드는 것은 무엇인가요?",
    "기타": "당신의 감정을 이해합니다. 좀 더 자세히 이야기해 주시겠어요?"
}
```

```python
# --- [1] 데이터 로드 및 전처리 ---
DATASET_FILE = 'total_kor_counsel_bot.jsonl'
script_dir = os.path.dirname(os.path.abspath(__file__))
dataset_path = os.path.join(script_dir, 'dataset', DATASET_FILE)

counseling_data: List[Tuple[str, str]] = []

try:
    with open(dataset_path, 'r', encoding='utf-8') as f:
        for line in f:
            data = json.loads(line)
            counseling_data.append((data.get('input', '').strip(), data.get('output', '').strip()))

    print(f"[서버] 데이터셋 로드 성공! 총 {len(counseling_data)}쌍의 발화 데이터 로드됨.")

except FileNotFoundError:
    print(f"[오류] 데이터셋 파일({DATASET_FILE})을 찾을 수 없습니다. 경로를 확인해주세요: {dataset_path}")
    print("서버는 시작되지만 상담 기능을 사용할 수 없습니다.")
except Exception as e:
    print(f"[오류] 데이터 로드 오류: {e}")
```

```python
# --- [2] 핵심: 단순 문자열 검색 로직 (안전 기능 강화) ---
def get_simple_response(user_message: str) -> str:
    """사용자 메시지를 데이터셋의 'input'과 단순 부분 문자열 일치로 비교하여 응답을 반환합니다."""
    if not counseling_data:
        return "데이터셋 오류로 상담을 진행할 수 없습니다."

    user_message_lower = user_message.strip().lower()

    if not user_message_lower:
        return "메시지를 입력해주세요."

    # --- [2-1] 안전 기능 (최우선 예외 처리) ---
    if any(keyword in user_message_lower for keyword in SENSITIVE_KEYWORDS):
        return SAFETY_RESPONSE

    ai_response = None

    # 1. 정확히 일치하는 경우만 우선적으로 처리
    for input_text, output_text in counseling_data:
        input_text_lower = input_text.lower()

        if user_message_lower == input_text_lower:
            ai_response = output_text
            break

    # 2. 일치하는 답변이 없을 경우 일상 인사말 및 키워드 검색 진행
    if ai_response is None:
```

```
# 일상 인사말 처리
        if user_message_lower in ["안녕", "안녕하세요", "hi", "반가워"]:
            ai_response = "안녕하세요! 저는 당신의 마음에 귀 기울이는 AI 상담사입니다. 어떤 고민이 있으신가요?"

        else:
            # --- [2-2] 짧은 감정 표현 처리 (데이터셋 검색 우회) ---
            keywords = ["우울", "힘들어", "짜증", "걱정", "불안"]
            detected_keyword = next((k for k in keywords if k in user_message_lower), None)

            # 짧은 입력(10자 이내)이고 감정 키워드가 포함된 경우, 데이터셋 검색 없이 바로 범용 응답 제공
            if detected_keyword and len(user_message_lower) < 10:
                ai_response = EMOTION_RESPONSES.get(detected_keyword, EMOTION_RESPONSES["기타"])
            # --- [2-2] 끝 ---

            # 긴 문장이나 특이한 문장이라면 (ai_response가 여전히 None일 경우), 데이터셋 검색 시도
            if ai_response is None:
                for input_text, output_text in counseling_data:
                    input_text_lower = input_text.lower()

                    if user_message_lower in input_text_lower:
                        ai_response = output_text
                        break

            # 3. 모든 검색 실패 시 폴백 응답
            if ai_response is None:
                ai_response = "제가 아직 부족해서 정확히 이해를 못 했어요. 혹시 다른 표현으로 말씀해 주실 수 있나요? 듣고 싶어요."
```

```
 # --- [4] 최종 필터링: '사우님' 단어 제거 및 가독성 개선 ---
    if ai_response:
        # 1. '사우님' 단어 제거 및 '당신'으로 대체
        ai_response = ai_response.replace('사우님', '당신')

        # 2. 가독성 개선: 줄바꿈 문자('\n')를 HTML <br> 태그로 변환
        ai_response = ai_response.replace('\n', '<br>')

    return ai_response


# --- [3] Python 기본 웹 서버 핸들러 ---
class PurePythonChatServer(http.server.SimpleHTTPRequestHandler):

    def end_headers(self):
        self.send_header('Access-Control-Allow-Origin', '*')
        self.send_header('Access-Control-Allow-Methods', 'POST, OPTIONS')
        self.send_header('Access-Control-Allow-Headers', 'Content-Type')
        self.send_header('Cache-Control', 'no-store, no-cache, must-revalidate')
        super().end_headers()

    def do_OPTIONS(self):
        self.send_response(HTTPStatus.OK)
        self.end_headers()
```

```
def do_POST(self):
        if self.path == '/chat':
            content_length = int(self.headers['Content-Length'])
            post_data = self.rfile.read(content_length).decode('utf-8')

            try:
                request_data = json.loads(post_data)
                user_message = request_data.get('message', '')

                ai_response_text = get_simple_response(user_message)

                response_json = json.dumps({
                    "response": "SUCCESS",
                    "emotion_label": ai_response_text
                })

                self.send_response(HTTPStatus.OK)
                self.send_header('Content-type', 'application/json')
                self.end_headers()
                self.wfile.write(response_json.encode('utf-8'))

            except Exception as e:
                print(f"[오류] 서버 처리 오류: {e}")
                self.send_error(HTTPStatus.INTERNAL_SERVER_ERROR, f"서버 오류: {e}")
        else:
            self.send_error(HTTPStatus.NOT_FOUND, "Not Found")
```

```python
# --- [4] 서버 실행 ---
PORT = 8000
Handler = PurePythonChatServer

if __name__ == "__main__":
    server_address = ("", PORT)

    with socketserver.TCPServer(server_address, Handler) as httpd:
        print(f"================================================")
        print(f"| 순수 Python 웹 서버 시작: http://127.0.0.1:{PORT} |")
        print(f"| 파일 위치: test1/server.py                    |")
        print(f"================================================")
        print("Ctrl + C를 눌러 서버를 종료하세요.")
        httpd.serve_forever()
```

AI 챗봇 프로젝트

UI/UX 디자인

디자인 컨셉

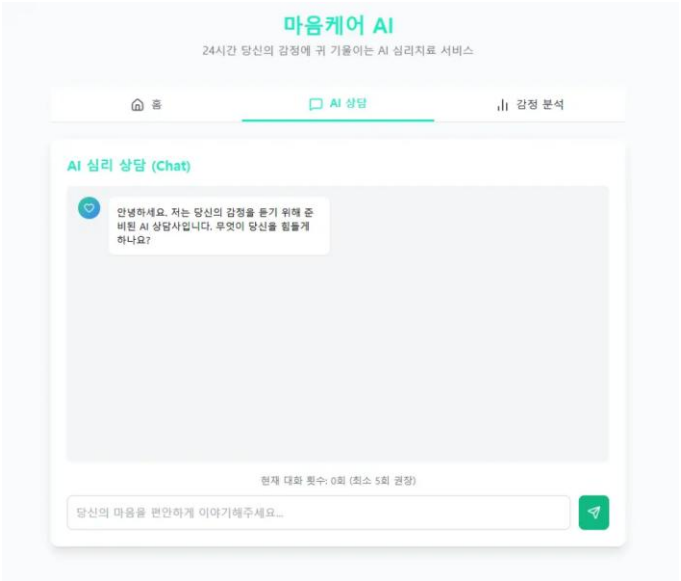
- 밝은 브랜드 이미지 vs 따뜻한 배경 사이 이 조화
- 민감한 주제 다루기 위한 '천칭을 넘지 않는' 기획 방향
- 효율적 UI 디자인을 위한 '간결함'과 '직관성'의 균형

핵심 UI 컴포넌트

편집	감정 선택 원형 UI
개선	대화 채널 개선
추가	감정 일기 기능

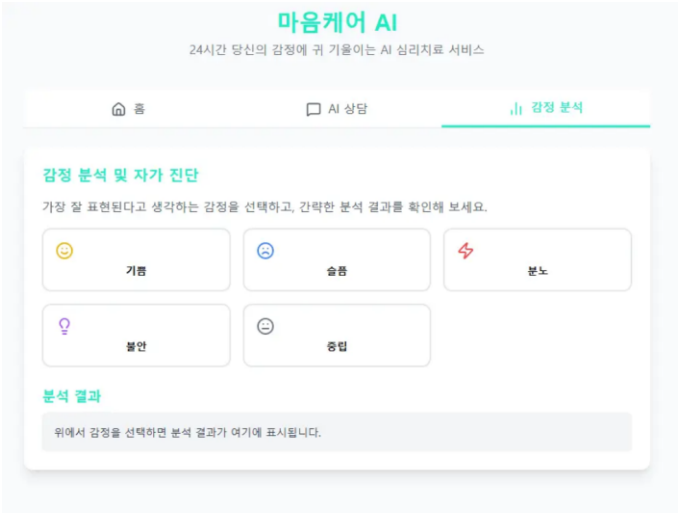
핵심 디자인 요소

감정 분석 및 자가 진단



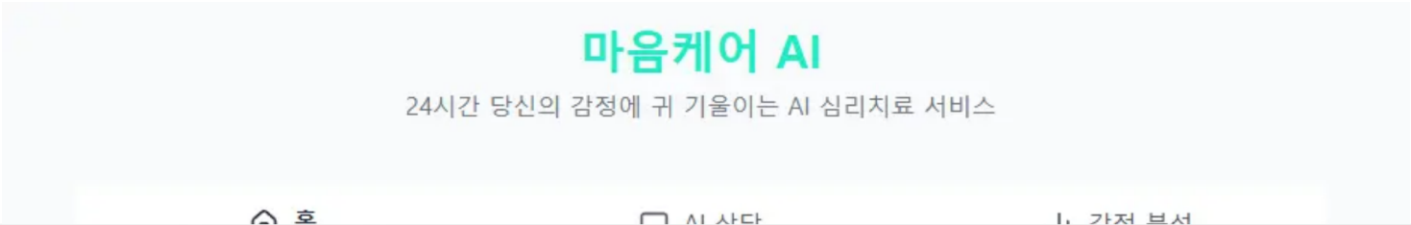
감정 분석 시 사용자가 효과적으로 감정 상태를 선택할 수 있는 직관적인 원형 인터페이스로, 사용자가 자신의 현재 감정 상태를 식별하는 데 도움을 줍니다.

AI 상담 대화 채널



자체 감정 프로필 전환 및 간식에서 효과적으로 원하는 감정 상태를 선택할 수 있으며, 상담 중 일부 다른 화면 전환이 가능한 구조로 설계되었습니다.

서비스 소개 및 컨테스트



AI 챗봇 프로젝트

향후 계획

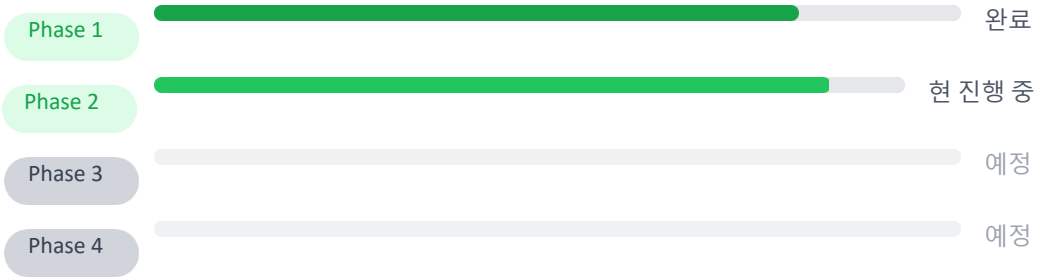
핵심 개선 방향

데이터 기반 분석
사용자 데이터를 활용한 감정 분석 알고리듬 고도화

대화 맥락 인지
AI가 사용자 대화 내용과 맥락에 기반한 맥락 반응 구현

개선된 AI 응답 생성
ChatGPT API 및 트랜스포머 기반 자신감 높은 답변 생성

발전 계획



기대 효과

사용자 경험 개선
사용자의 감정 인식 및 응답 맞춤화를 통한 향상된 상담 서비스 제공

면담임상 접근성 향상
AI 기반 심층 감정 분석 시스템을 통한 기존 상담 인력 자원 대체/보완 방안

사회적 영향력 확대
하루 24시간 상담 서비스 제공을 통한 심리 상담 접근성 및 혼란 효율적 매트릭스로 확장 기여

트리거 키워드 관리 정책

위험 신호 반응
위험 신호 감지 시 바로 사람 문의로 전환, 관련 전문가 단순 유지 방식

인력/인공지능 유스구조
사람 상담사와 AI의 협업을 통한 신뢰도 향상된 이주시 짧은 연속 반응 실현

연령/인종 구분 접근법
사용자 연령/학력/경우별 맞춤화 감성적 언어 패턴 인식 구조로 위기 대응 특화