요구사항 정의서

1. 시스템 개요

1-1. 배경 및 필요성

최근 대한민국에서는 현대인의 우울감 경험 비율이 지속적으로 증가하고 있으며, 이에 따라 심리 상담에 대한 관심과 이용률도 점차 높아지고 있다. 그러나 많은 사람들이 우울증 진단을 받을 정도는 아니더라도, 일상 속에서 겪는 가벼운 우울감이나 정서적 불안정을 경험한다.
이러한 감정적 어려움은 개인이 스스로 정확히 인지하지 못하거나, 전문가에게 상담받는 것에 부담을 느껴 방치되는 경우가 많다. 또한 감정을 솔직하게 털어놓기 어려운 사회·정서적 환경 속에서 개인은 고립되기 쉽고, 이는 정서 악화 및 정신건강 문제로 이어질 위험이 있다.
따라서 전문 상담 이전 단계에서 누구나 부담 없이 자신의 감정을 확인하고, 필요 시 즉각적인 도움(자가진단·병원 안내 등)을 받을 수 있는 도구가 필요하다. 이러한 서비스는 정서적 어려움을 조기에 해소하고, 심리적 위기로 발전하는 것을 예방하는 데 큰 역할을 할 수 있다.

1-2. 목표

- 감정 인식 촉진: 사용자가 자신의 감정 상태를 쉽게 확인하고 인지할 수 있도록 지원한다.
- 심리 상담 진입 장벽 완화: 전문가 상담 이전 단계에서 부담 없이 감정을 다룰 수 있는 창구 제공한다.
- 심리적 위기 대응 유도: 위험 감정군 사용자는 자가진단 결과 및 병원 안내 팝업을 통해 빠르게 전문 기관으로 연결될 수 있도록 한다.

1-3. 시스템 개요

주요 기능

1. 감정분석
- 사용자의 텍스트 입력을 기반으로 6가지 감정(기쁨, 분노, 슬픔, 상처, 당황, 불안)을 분류하고, 각 감정의 강도를 추정한다.

2. 감정 기반 콘텐츠 추천
- 분석된 감정에 따라 위로·공감·활력 회복 등 목적에 맞는 음악 및 영상 콘텐츠를 추천한다.

3. 병원 추천 서비스
- 사용자 현재 위치 기반으로 가까운 정신건강의료기관을 안내한다.
- 또한 사용자가 관심 지역 주소를 입력하면 해당 위치와 가까운 병원을 순서대로 제공한다.

4. 자가진단 서비스
- 자가진단 설문을 통해 개인의 감정 상태를 분석한다.
- 고위험 감정(우울, 불안, 중독) 감지 시 병원 안내 팝업을 제공해 즉시 대응할 수 있도록 한다.

2. 요구사항 분류

2-1. 기능 요구사항 **(Functional Requirements)**

1. 감정 분석: 사용자 텍스트를 6가지 감정으로 분류 및 강도 추정
2. 콘텐츠 추천: 감정 결과에 따라 음악, 영상 콘텐츠 제공
3. 병원 추천: 현재 위치/주소 기반 가까운 병원 안내
4. 자가진단 서비스: 설문 및 고위험 감정 감지 → 병원 안내 팝업 제공
5. 사용자 대시보드: 개인 감정 변화 추이 그래프, 자가진단 결과 리포트 확인
6. 관리자 대시보드: 사용자 로그/통계 모니터링, 데이터 관리(삭제·백업)

2-2. 비기능 요구사항 **(Non-Functional Requirements)**

1. 성능 요구사항

- 감정 분석 요청 시 평균 **20**초 이내 응답 제공 (CPU 환경, KoBERT 고려)
- 동시 사용자 **3**명 이내 처리 가능 (시연 환경 기준)
- 일일 데이터 처리량: 수백 건 이하

2. 보안 요구사항

- 사용자 비밀번호는 해시 암호화 후 저장
- 최소한의 개인정보(이메일, 닉네임)만 수집
  외부 API Key는 `.env` 파일 관리 및 외부 노출 금지

3. 신뢰성 및 가용성

- 발표/시연 시간 동안 안정적 구동 목표
- DB/로그는 하루 1회 수동·자동 백업
- 오류 발생 시 로그 기록 및 관리자 확인 가능

4. 확장성

- 초기 DB는 SQLite/MySQL, 추후 클라우드 RDS로 확장 가능
- 감정 카테고리는 별도 테이블 구조로 설계 (6종 → 8~10종 확장 가능)
- 추후 Docker Compose 기반 다중 컨테이너로 확장 가능

5. 사용성

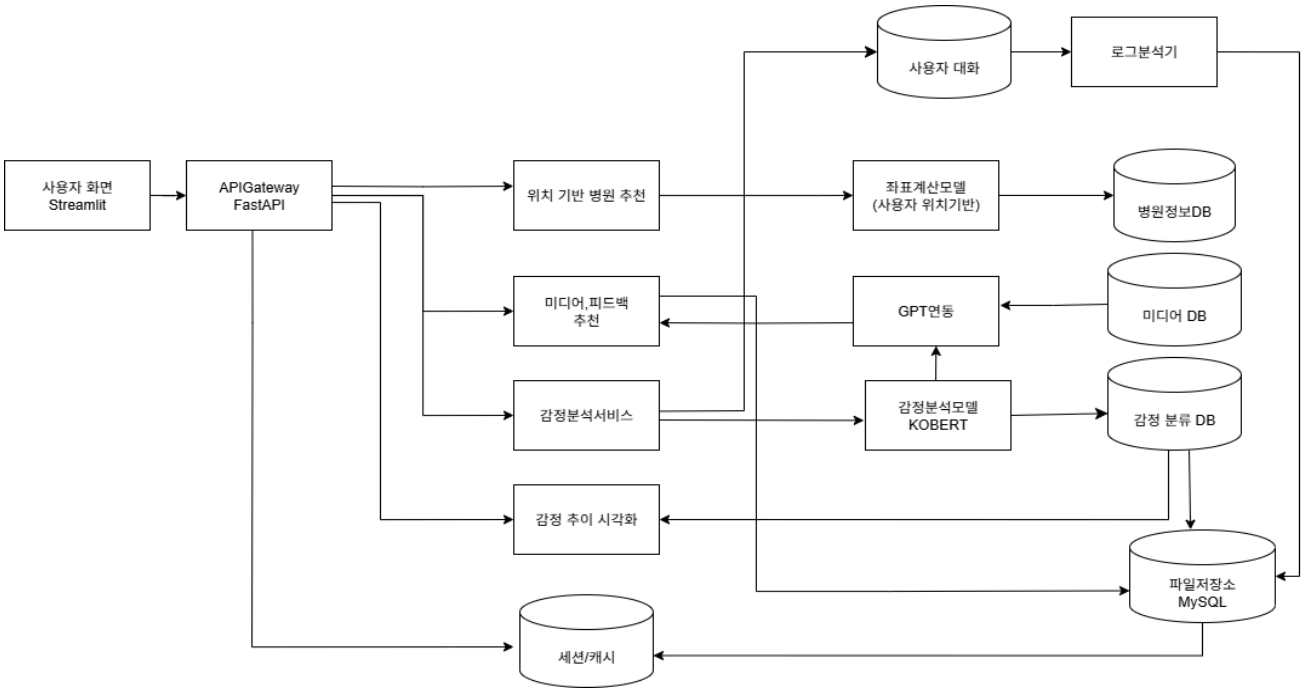
- PC 웹 브라우저에서 접근 가능
- 회원가입 → 대화 입력 → 결과 확인까지 **3~4**번 클릭 이내 제공
- 감정 변화 추이를 그래프로 직관적으로 시각화

6. 운영 및 유지보수

- 사용자 요청/오류 로그는 DB + CSV 기록
- 모델 성능은 **F1-score ≥ 0.2** 수준을 유지 목표 (향후 개선 목표 ≥ 0.7)
- 관리자는 대시보드를 통해 로그 및 사용자 활동을 모니터링하고, 필요 시 데이터 관리(삭제·백업) 수행 가능

3. 시스템 아키텍처

3-1. 시스템 구성도



3-2. 구성 요소 설명

1. **Users(**사용자**)** 서비스 이용자
   - 서비스 최종 이용자
   - 텍스트 입력을 통해 감정 분석 및 추천 결과를 확인
   - 웹 브라우저(PC/모바일) 환경에서 접근
2. **Streamlit App (UI, Processing, Visualization, Data Layer)**
   - UI: 사용자 입력 인터페이스 제공 (대화 입력창, 그래프 출력 등)
   - Processing: 입력 텍스트를 감정 분석 API로 전달하고, 결과를 가공
   - Visualization: 감정 변화 추이, 분석 결과를 그래프로표시
   - Data Layer: 분석 결과, 사용자 로그를 MySQL DB에 저장/조회

   (코드 예시)

```
# streamlit_app.py

import streamlit as st

import requests

import mysql.connector

import plotly.express as px


st.title("감정 분석 기반 심리 케어 서비스")
```

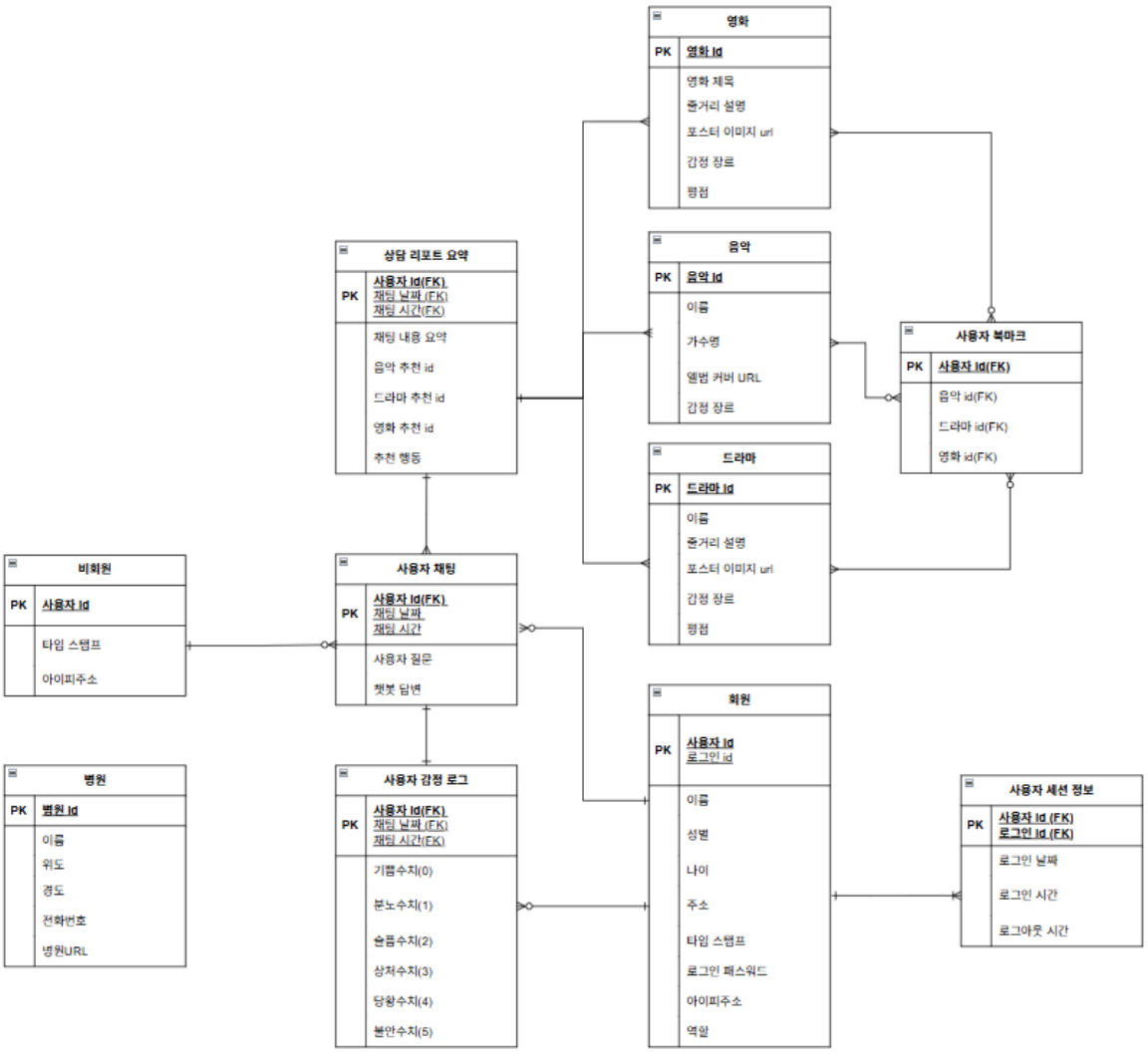
```
# 사용자 입력
user_text = st.text_area("오늘 기분은 어떤가요?")

if st.button("분석하기"):
    # 감정 분석 API 호출
    response = requests.post("http://localhost:8000/predict", json={"text": user_text})
    result = response.json()

    st.write("감정 분석 결과:", result["emotion"], "(강도:", result["score"], ")")

    # DB 저장
    conn = mysql.connector.connect(host="localhost", user="root", password="1234", database="emotiondb")
    cur = conn.cursor()
    cur.execute(
        "INSERT INTO EmotionResult (user_id, text, emotion, score) VALUES (%s, %s, %s, %s)",
        (1, user_text, result["emotion"], result["score"])
    )
    conn.commit()
    conn.close()
```

3. 외부 **API (GPT / OpenAI)**
   - 텍스트 감정 분석 및 추천 문장 생성을 위한 API 활용
   - 감정 분류, 상담 요약, 콘텐츠 추천에 활용
4. 저장소 **(MySQL)**
   - 사용자 입력 로그, 감정 분석 결과, 추천 내역을 저장
   - ERD에 따라 User / EmotionResult / Log / Content / Hospital 등 테이블로 구성
5. 관리 도구
   - GitHub: 코드 버전 관리 및 협업

- Notion: 프로젝트 산출물, 문서, 일정 관리
- 웹 브라우저(PC/모바일) 환경에서 접근

6. 배포 **(Docker)**
   - Streamlit App을 Docker 컨테이너로 패키징
   - 환경 의존성을 줄이고, 발표/시연 시 동일 환경에서 안정적 실행 보장

4. 데이터 설계



4-1. 데이터 흐름도 **(Data Flow Diagram)**

1. 비회원: 비회원 전용의 임의 사용자 계정 생성
2. 회원**(**사용자**)**: 텍스트 입력을 기반으로 감정 분석 요청 → 감정 결과 로그 저장
3. 사용자 채팅: 사용자의 입력 문장 → 감정 분석 모듈 처리 → 상담 리포트 요약 및 감정 결과 생성
   콘텐츠 추천: 감정 결과를 기반으로 영화, 드라마, 음악 등 추천 제공

4. 병원 추천: 사용자 위치 정보 기반으로 병원 정보 제공
5. 사용자 세션 정보: 로그인 세션과 연결되어 개인화된 감정 기록 및 추천 관리
6. 사용자 북마크: 추천받은 콘텐츠(영화, 드라마, 음악 등)를 북마크 저장 가능

4-2. 저장 구조 (Data Storage Structure)

1. 사용자 계정**(User)**
    - 회원가입 정보(이메일, 닉네임, 비밀번호 해시 등) 저장
    - 로그인 세션과 연결
2. 사용자 감정 로그**(Emotion Log)**
    - 사용자 입력 문장, 분석된 감정 결과, 감정 강도 기록
    - 상담 리포트 요약 포함
3. 추천 콘텐츠**(Content)**
    - 감정별 추천 미디어 정보 저장 (영화, 드라마, 음악 등)
    - 사용자 북마크와 연결
4. 병원 정보**(Hospital)**
    - 사용자 위치 기반으로 제공할 수 있는 병원 정보 저장 (병원명, 주소, 위도/경도)
5. 세션**(Session)**
    - 사용자 로그인 상태, 최근 활동, 감정 히스토리 연결

4-3. ERD (Entity Relationship Diagram)

1. Member (회원**)**

- 핵심 속성: user_id, login_id, name, password, ip_address, latitude, longitude
- 역할: 서비스에 가입한 사용자의 기본 정보 관리
- 관계:
    - UserChat, EmotionLog, CounselingSummary, UserSession, UserBookmark와 1:N 관계

2. Guest (비회원**)**

- 핵심 속성: user_id, ip_address, created_at
  역할: 회원 가입하지 않은 사용자의 최소한의 접속 정보 관리
- 관계: 현재 다른 테이블과 직접 FK는 없음 (단순 기록용)

3. UserSession (사용자 세션**)**

- 핵심 속성: session_id, user_id, login_time, logout_time
- 역할: 회원의 로그인/로그아웃 기록 관리
- 관계: Member와 N:1 관계

4. UserChat (사용자 채팅 로그**)**

- 핵심 속성: chat_id, user_id, question, answer, chat_date, chat_time

- 역할: 사용자가 입력한 문장과 챗봇 응답 기록 관리
- 관계:
    - Member와 N:1 관계
    - EmotionLog, CounselingSummary와 1:1 또는 1:N 관계 (채팅 1건 → 감정분석/리포트 결과 여러 개 가능)

5. EmotionLog (사용자 감정 로그**)**

- 핵심 속성: emotion_id, chat_id, joy_score, sadness_score, anxiety_score, dominant_emotion
- 역할: 사용자의 채팅을 감정 분석한 결과 저장 (점수 및 대표 감정)
- 관계:
    - UserChat과 N:1 관계
    - Member와도 N:1 관계

6. CounselingSummary (상담 요약**/**추천**)**

- 핵심 속성: summary_id, user_id, chat_id, summary_text, action_plan, music_rec_id, drama_rec_id
- 역할: 한 세션/채팅에 대한 요약 및 추천 미디어 결과 저장
- 관계:
    - UserChat과 N:1 관계
    - Member와 N:1 관계
    - Music, Drama와 N:1 관계 (추천 결과 연결)

7. Hospital (병원**)**

- 핵심 속성: hospital_id, name, latitude, longitude, phone
- 역할: 위치 기반으로 추천할 수 있는 병원 정보 저장
- 관계: 현재 다른 테이블과 직접 FK는 없지만, Member의 위도/경도와 거리 계산을 통해 연계

8. Movie / Music / Drama (추천 컨텐츠**)**

- 핵심 속성: title, poster_url, emotion_genre (불안, 우울, 기쁨 등)
- 역할: 감정 기반 추천이 가능한 미디어 컨텐츠 저장
- 관계:
    - CounselingSummary와 연결되어 추천 결과로 활용
    - UserBookmark와 N:1 관계

9. UserBookmark (사용자 북마크**)**

- 핵심 속성: bookmark_id, user_id, movie_id, music_id, drama_id
- 역할: 사용자가 마음에 들어 한 컨텐츠(영화/음악/드라마)를 저장
- 관계:
    - Member와 N:1 관계
    - Movie, Music, Drama와 각각 N:1 관계

10. ERD의 일부 예시 코드 일부

```
CREATE DATABASE Churo_me
    DEFAULT CHARACTER SET utf8mb4
    COLLATE utf8mb4_general_ci;

USE Churo_me;

CREATE TABLE emotion (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE   -- 예: 슬픔, 기쁨, 분노
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE user_info (
    id INT AUTO_INCREMENT PRIMARY KEY,     -- 사용자 고유 ID
    name VARCHAR(50) NOT NULL,             -- 이름
    gender ENUM('M', 'F', 'Other'),        -- 성별 (남/여/기타)
    age INT CHECK (age >= 0),              -- 나이
    residence VARCHAR(100),                -- 거주지
    session_info TEXT                      -- 세션 정보 (로그인 기록 등)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE user_post (
    id INT AUTO_INCREMENT PRIMARY KEY,     -- 게시글 고유 ID
    user_id INT NOT NULL,                  -- 작성자 (FK)
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP, -- 등록일자
    content TEXT NOT NULL,                 -- 게시글 내용
    review VARCHAR(255),                   -- 리뷰 요약/한줄평
```

```
    FOREIGN KEY (user_id) REFERENCES user_info(id)
        ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

5. 모듈 설계

시스템의 기능적 구조 **(Layer** 기반**)**

1. **UI Layer**
    - 사용자 입력창 (텍스트 입력)
    - 결과창 (감정 분석 결과/추천 콘텐츠 표시)
    - 차트 영역 (시각화 결과)
2. **Processing Layer**
    - KoBERT: 감정 분류 모델 (다중 클래스 분류)
    - GPT API (OpenAI): 상담 요약 문장, 자연스러운 대화 응답 생성
    - 추천 모델: 감정 기반 콘텐츠 추천 (Scikit-learn 활용)

● 코드 예시

```
# fastapi_server.py

from fastapi import FastAPI

from pydantic import BaseModel

import torch

from kobert import get_pytorch_kobert_model

from transformers import BertTokenizer


app = FastAPI()


# KoBERT 모델 로드 (샘플)

bertmodel, vocab = get_pytorch_kobert_model()

tokenizer = BertTokenizer.from_pretrained('skt/kobert-base-v1')


class InputText(BaseModel):
```

```
    text: str


@app.post("/predict")
def predict(input: InputText):
    # 여기서는 샘플로 랜덤 감정 반환
    import random
    emotions = ["기쁨", "분노", "슬픔", "상처", "당황", "불안"]
    emotion = random.choice(emotions)
    score = round(random.uniform(0.5, 0.9), 2)
    return {"emotion": emotion, "score": score}
```

3. **Data Layer**
    - SQL 저장 (MySQL DB)
    - 사용자 로그, 감정 결과, 추천 기록 저장
4. **Visualization Layer**
    - 감정 추세 그래프 (시계열 기반 변화 시각화, Plotly)
    - 병원 지도 (Folium 기반 위치 표시)

6. 기술 스택

실제 사용하는 기술명**/**라이브러리

1. **Frontend / Visualization**
    - Streamlit (웹 UI/UX)
    - Plotly (차트, 시각화)
    - Folium (지도 시각화)
2. **ML / NLP**
    - KoBERT (감정 분류) ***[***모델 변경 가능성 있음***]***
    - GPT API (OpenAI, 상담 요약 및 응답)
    - Scikit-learn (간단한 추천 모델, 예: 콘텐츠 기반 추천)
3. **Storage**
    - MySQL (사용자/로그/분석결과 저장소)
4. **DevOps /** 협업 도구
    - Docker (배포 환경 통합)
    - GitHub (코드 관리)

- Notion (산출물 관리, 일정 관리)

5. 코드 예시

```
# Dockerfile

FROM python:3.9-slim


WORKDIR /app


COPY requirements.txt .

RUN pip install -r requirements.txt


COPY . .


EXPOSE 8501

CMD ["streamlit", "run", "streamlit_app.py", "--server.port=8501",
"--server.address=0.0.0.0"]
```

7. 배포 설계

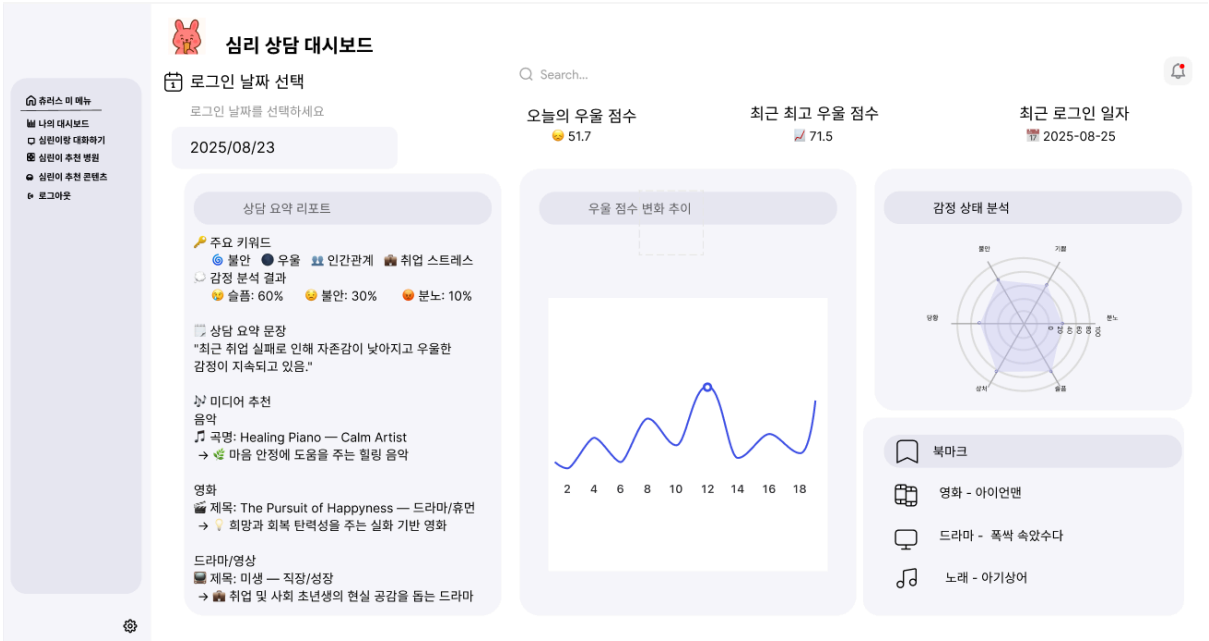
1. 기본 설계
   - Docker 기반 단일 컨테이너 실행
   - Streamlit + 모델(KoBERT/GPT API) + MySQL DB 포함

2. 확장 설계
   - Docker Compose를 이용해 다중 컨테이너 구조로 분리
     - Frontend (Streamlit)
     - Backend (FastAPI, 모델 서버)
     - DB (MySQL)
   - 확장 시 서비스 안정성 및 관리 용이성 확보 가능

3. UI/UX 요구사항

3-1. 관리자 대시보드

3-2. 사용자 대시보드



3-3. 메인화면, 챗봇 **UI (**업데이트 예정**)**