MoodFinance AI: 제품 요구 명세서 **(PRD)**

문서 버전: 1.0
작성일: 2025-08-05
프로젝트명: MoodFinance AI (삼인성AI 팀)

1. 프로젝트 개요

1.1. 프로젝트 목표

사용자의 감정(일기)과 소비(가계부) 데이터를 통합 분석하여, 개인의 금융 습관에 숨겨진 패턴을 발견하고, 데이터 기반의 맞춤형 코칭과 보상을 통해 사용자의 건강한 금융 생활과 주체적인 의사결정을 돕는 지능형 자산관리 플랫폼을 구축한다.

1.2. 대상 사용자

- 주요 타겟**:** 감정적 소비로 인해 재정 관리에 어려움을 겪는 20-30대 MZ세대.
- 확장 타겟**:** 자신의 소비 습관을 깊이 있게 이해하고 개선하고자 하는 모든 금융 소비자.

1.3. 핵심 가치 제안 **(CVP)**

- 더 쉬운 기록**:** 사진 한 장, 음성 한마디로 나의 하루와 소비를 기록.
- 더 깊은 이해**:** 내가 몰랐던 나의 감정-소비 연결고리를 AI 리포트로 발견.
- 더 나은 결정**:** '지름신'이 오기 직전, AI의 개입으로 충동구매를 예방하고 합리적 판단을 지원.
- 지속 가능한 성장**:** 게임처럼 즐거운 챌린지와 실질적인 보상으로 건강한 습관 형성을 유도.

2. 시스템 아키텍처

2.1. 기술 스택 요약

구분	기술	역할 및 책임
프론트엔드	React	사용자 인터페이스(UI/UX) 구축, Spring Boot API와 통신
백엔드 **(Core)**	Spring Boot	사용자 인증, 핵심 비즈니스 로직(CRUD), 데이터 연동, AI 서버와 통신
백엔드 **(AI)**	Python (FastAPI)	NLP, ML/DL 모델 서빙, 이미지 분석, Gemini Agent 로직 수행
데이터베이스	MariaDB	정형 데이터 (사용자 정보, 일기 텍스트, 소비 내역 등)
	MongoDB	비정형/반정형 데이터 (AI 분석 결과, 이미지 메타데이터, 로그)

	Redis	캐싱, 세션 관리, 실시간 순위/점수 관리
	SupaBase	파일 스토리지 (사용자 업로드 이미지, 음성 파일)
메시지 큐	Apache Kafka	서버 간 비동기 통신, 서비스 디커플링 및 안정성 확보
AI Agent	Google Gemini API	멀티 에이전트 분석, 자연어 생성(NLG), 대화형 코칭 등 고차원 AI 기능
실시간 통신	WebSocket, SSE	서버 -> 클라이언트 간 실시간 푸시 (알림, 피드백)
인프라	Docker, Nginx	컨테이너화, 리버스 프록시, 로드 밸런싱
환경변수 관리	.env, application.yml	통합 및 분산 환경변수 관리

2.2. 환경변수 관리 전략

- 통합 관리**:** Docker 환경을 기준으로, 프로젝트 루트에 .env 파일을 두어 Nginx, Python, DB 등 공통 환경변수를 관리한다.
- 분산 관리**:** Spring Boot는 고유의 application.yml 파일을 사용하여 프로파일(dev, prod)별 설정을 관리한다. 이는 Spring Boot의 강력한 설정 관리 기능을 최대한 활용하기 위함이다.
- 주입**:** Docker Compose는 .env 파일의 변수들을 각 컨테이너에 주입하고, Spring Boot 컨테이너는 application.yml을 우선적으로 사용한다.

3. 기능 명세 **(Functional Specifications)**

3.1. 감정**-**소비 통합 다이어리

- 목표**:** 사용자가 최소한의 노력으로 자신의 감정과 소비를 기록하고, 한눈에 파악할 수 있는 기반을 마련한다.
- 사용자 스토리**:** "나는 그날의 기분과 사건, 그리고 지출 내역을 쉽고 빠르게 기록하고 싶다. 그리고 캘린더에서 특정 날짜를 누르면 그날의 일기와 소비 내역을 함께 보고 싶다."

기능	상세 설명	기술 구현
일기 작성	- 입력 방식**:** 텍스트, 음성(STT), 사진, 5단계 감정 이모티콘(매우나쁨-나쁨-보통-좋음-매우좋음)을 조합하여 일기 작성. - 기능**:** 임시저장, 수정/삭제 기능 제공.	- **FE:** React 기반의 에디터 컴포넌트, 마이크/카메라 접근 권한 요청. - **BE(Spring):** 일기 데이터 CRUD를 위한 REST API (/api/v1/diaries). - **AI(Python):** 음성 파일을 받아 텍스트로 변환하는 STT API (/api/ai/stt). 음성 파일은 SupaBase에 임시 저장 후 처리.

가계부 자동 연동	- 연동**:** 오픈뱅킹 API를 통해 사용자가 동의한 금융사의 소비/수입 내역을 주기적으로 자동 스크래핑. - 분류**:** AI가 가맹점명, 업종 코드 등을 분석하여 '식비', '교통', '쇼핑' 등 카테고리를 자동 분류. 분류 정확도가 낮을 시 사용자가 직접 수정 가능.	- **BE(Spring):** 오픈뱅킹 연동 모듈, 스케줄러(@Scheduled)를 통한 주기적 데이터 수집. - **AI(Python):** 소비 내역 텍스트를 받아 카테고리를 분류하는 머신러닝 모델 API (/api/ai/classify/expense).
타임라인 뷰	- 뷰 제공**:** 캘린더 뷰와 리스트 뷰 두 가지 형태로 제공. - 캘린더 뷰**:** 날짜별로 감정 이모티콘과 총 소비 금액을 요약 표시. 날짜 선택 시 상세 내역 표시. - 리스트 뷰**:** 최신순으로 일기와 소비 내역을 시간순으로 나열.	- **FE:** react-calendar 등 라이브러리를 커스텀하여 뷰 구현. - **BE(Spring):** 특정 기간(월별/일별)의 일기 및 소비 데이터를 조회하는 API (/api/v1/timeline?start_date=&end_date=).

3.2. AI 패턴 분석 및 통합 피드백

- 목표**:** 축적된 데이터를 분석하여 사용자 고유의 '감정-소비 패턴'을 발견하고, 시각적 리포트를 통해 객관적인 자기 성찰의 기회를 제공한다.
- 사용자 스토리**:** "나는 내 소비 습관에 어떤 패턴이 있는지 궁금하다. 특히 스트레스를 받거나 우울할 때 돈을 어떻게 쓰는지 AI가 분석해서 알려주고, 그 의미를 여러 관점에서 해석해주면 좋겠다."

기능	상세 설명	기술 구현
패턴 발견 **(**백엔드**)**	- 분석 주기**:** 매주 일요일 밤, 월말에 배치(Batch) 작업으로 실행. - 분석 로직**:** 시계열 분석, 연관 규칙 분석(Apriori) 등 알고리즘을 사용하여 '특정 감정'과 '특정 소비 카테고리/금액' 간의 통계적 유의미성을 분석. (예: "슬픔" 감정 발생 시 24시간 내 "온라인 쇼핑" 지출액 5만원 이상 발생 확률 60% 증가)	- **BE(Spring):** Kafka에 분석 요청 메시지(UserID, 분석 기간)를 전송. - **AI(Python):** Kafka로부터 메시지를 받아 분석 수행. MariaDB에서 정형 데이터, MongoDB에서 감정 분석 결과 등을 가져와 종합 분석. 분석 결과는 MongoDB에 저장. - **Message Queue:** analyze-request (Spring->Python), analyze-response (Python->Spring) Kafka 토픽 사용.
통합 피드백 **(**시각화**)**	- 리포트 생성**:** 매주/매월 AI가	- **FE:** Recharts를 활용한 동적

	발견한 핵심 인사이트를 리포트 형식으로 제공. - 시각화**:** "지난달, OO님은 '지침'을 느낀 날 배달 음식에 가장 많은 돈을 쓰셨어요." 와 같은 텍스트 요약과 함께 Recharts 라이브러리를 활용한 인터랙티브 차트(파이, 바, 라인) 제공.	차트 컴포넌트 구현. - **BE(Spring):** MongoDB에 저장된 분석 리포트를 조회하는 API (/api/v1/reports/{userId}?type=weekly).
멀티 에이전트 분석	- 다각적 해석**:** 발견된 단일 패턴(예: 스트레스 시 쇼핑 증가)에 대해 3가지 페르소나를 가진 AI 에이전트가 해석을 제공. 1. 합리적 분석가**:** "이 소비는 장기적인 재정 목표에 부정적 영향을 줄 수 있습니다." 2. 공감하는 친구**:** "힘들 때 쇼핑으로 기분 전환하고 싶으셨군요. 그럴 수 있어요." 3. 미래 설계사**:** "이 비용을 아꼈다면, OO님의 미래 목표에 한 걸음 더 다가갈 수 있었을 거예요."	- **AI(Python):** 패턴 분석 후, 해당 결과를 바탕으로 Gemini API에 3가지 다른 역할(Role)과 지침(Instruction)을 부여하여 순차적으로 호출 (Context Switching). - **Prompt Engineering:** 각 에이전트의 페르소나를 명확히 정의하는 프롬프트 템플릿을 설계.

3.3. 이미지 분석 기능

- 목표**:** 이미지(사진, 영수증)를 통해 사용자의 기록 편의성을 극대화하고, 텍스트만으로는 파악하기 힘든 심층적인 데이터를 추출하여 분석 정확도를 높인다.
- 사용자 스토리**:** "글 쓰기 귀찮을 땐 그냥 내 표정 사진이나 음식 사진을 올리고 싶다. 영수증을 찍으면 알아서 가계부에 입력되면 좋겠다."

기능	상세 설명	기술 구현
이미지 자동 분류	- 사용자가 이미지를 업로드하면, AI가 먼저 이미지의 종류를 판단하여 후속 처리 모드를 결정. - 분류 기준**:** 사람 얼굴, 텍스트(영수증), 특정 사물, 일반 풍경	- **AI(Python):** ResNet, MobileNet 등 경량화된 CNN 기반 분류 모델을 사용하여 1차 분류 API (/api/ai/classify/image) 구현.
표정 분석	- 처리**:** 얼굴 영역을 탐지하고, 미세 근육 움직임을 분석하여 7가지 주요 감정(기쁨, 슬픔, 분노, 놀람, 두려움, 혐오,	- **AI(Python):** OpenCV, Dlib으로 얼굴 탐지. FER(Facial Emotion Recognition) 모델을 활용한 감정 분석 API

	무표정) 및 '피곤함'의 확률값을 추출.	(/api/ai/analyze/face).
영수증 분석 **(OCR)**	- 처리**:** 이미지에서 텍스트를 추출(OCR)하고, 정규표현식과 패턴 매칭을 통해 가맹점명, 결제 금액, 날짜, 품목을 인식하여 구조화된 JSON 형태로 반환.	- **AI(Python):** EasyOCR, Tesseract 등 OCR 라이브러리 사용. 추출된 텍스트를 파싱하는 로직 구현. (/api/ai/ocr/receipt).
객체**/**상황 분석	- 객체 인식**:** 사진 속 주요 사물(명품, 음식, 주류 등)을 인식하여 태그(#)로 추출. - 상황**/**분위기 분석**:** 이미지의 전체적인 색감, 밝기, 구도를 분석하여 '활기참', '차분함', '우울함' 등의 분위기 지표를 추출.	- **AI(Python):** YOLO, SSD 등 객체 탐지 모델 사용 (/api/ai/detect/object). - **AI(Gemini):** 이미지의 전반적인 분위기나 맥락 분석은 Gemini의 멀티모달 기능을 활용하여 더 풍부한 해석 생성.
데이터 통합	- 이미지 분석으로 추출된 모든 데이터(감정, 품목, 분위기 등)는 해당 날짜의 일기/가계부 데이터와 자동으로 연결되어 통합 분석에 활용.	- **BE(Spring):** Python AI 서버로부터 분석 결과를 받아 MariaDB와 MongoDB에 저장하고, 기존 데이터와 연결.

3.4. 사전 예방 알림 및 개인화 코칭

- 목표**:** 감정적 소비가 발생하기 직전에 개입하여 충동구매를 막고, 개인화된 챌린지와 긍정적 상호작용을 통해 실질적인 행동 변화를 유도한다.
- 사용자 스토리**:** "내가 스트레스를 받아 쇼핑 앱을 켤 때, 누군가 옆에서 '잠깐만!' 하고 말려주면 좋겠다. 그리고 나쁜 습관을 고칠 수 있도록 구체적인 미션을 주고 성공하면 칭찬해주면 좋겠다."

기능	상세 설명	기술 구현
위험 감지 및 알림	- 감지**:** '슬픔/지침' 일기 작성 직후, 특정 시간대(밤 10시 이후)에 쇼핑 앱 사용 패턴 감지 등 과거 데이터 기반 고위험 행동이 포착되면 위험 신호 발생. - 알림**:** "지금 스트레스가 높으시네요! 구매 전에 5분만 산책 어떠세요?" 와 같이 구체적인 대안을 제시하는 푸시 알림 발송.	- **BE(Spring):** 실시간 위험 감지 로직 구현. 위험 감지 시 WebSocket/SSE를 통해 클라이언트에 알림 이벤트 전송. - **FE(React):** WebSocket/SSE 리스너를 통해 알림 수신 및 화면에 표시. - **AI(Python):** Gemini API(NLG)를 활용하여 개인의 상황에 맞는 알림 메시지를 동적으로 생성하는 API 제공.
맞춤형 챌린지	- AI 분석 결과를 바탕으로	- **AI(Python):** 사용자 데이터

	"이번 주, 배달 음식 1번만 시키기", "월급날, 쇼핑 대신 운동하기" 등 성공 가능성이 높은 작은 목표(챌린지)를 제안.	기반으로 가장 효과적인 챌린지를 추천하는 모델 (강화학습 고려). - **BE(Spring):** 챌린지 제안, 수락, 진행 상태, 성공/실패 여부를 관리하는 로직 및 API.
긍정 피드백 루프	- 챌린지 성공, 긍정적 소비(예: 저축) 발생 시 "정말 멋져요! 어려운 도전에 성공하셨네요!" 와 같은 격려 메시지를 챗봇 또는 알림 형태로 제공. - 조언/챌린지에 대해 "도움돼요/안돼요" 피드백을 받아 코칭 전략을 지속적으로 개선.	- **FE:** 챗봇 형태의 대화형 UI, 피드백 버튼. - **AI(Python):** 사용자 피드백을 학습 데이터로 활용하여 강화학습 기반의 코칭 모델을 점진적으로 최적화.

3.5. 보상 시스템 및 개인 설정

- 목표**:** 게이미피케이션과 실질적 보상으로 서비스의 지속 사용을 유도하고, 사용자에게 데이터 통제권을 부여하여 신뢰를 확보한다.
- 사용자 스토리**:** "내가 노력해서 돈을 아끼고 습관을 개선하면, 칭찬뿐만 아니라 커피 쿠폰 같은 작은 보상이라도 받으면 더 신날 것 같다. 그리고 내 민감한 정보는 내가 원할 때 지우거나 알림을 조절하고 싶다."

기능	상세 설명	기술 구현
금융 건강 점수	- 산정**:** [일기(긍정 감정 비율)], [가계부(계획 소비 비율, 문제 소비 감소율)], [조언 수용도(챌린지 성공률)]에 동적 가중치를 부여하여 매월 '금융 건강 점수' 산출.	- **BE(Spring):** 각 항목의 데이터를 종합하여 점수를 산정하는 스케줄링 배치 작업. 가중치는 AI 서버에서 받아올 수 있음.
포인트 및 쿠폰	- 적립**:** 전월 대비 '금융 건강 점수'가 목표치 이상 상승하면 '이행 성공'으로 판단하고 차등 포인트 적립. - 교환**:** 적립된 포인트로 제휴사의 모바일 쿠폰(예: 스타벅스 아메리카노)으로 교환.	- **BE(Spring):** 안전한 포인트 적립/사용/소멸 로직 구현. 외부 기프티콘 API 연동 모듈.
개인 설정	- 알림 설정**:** 푸시 알림 빈도(낮음/보통/높음), 방해금지 시간대 설정. - 데이터 관리**:** 금융사 연동 해지, 데이터 백업(암호화된 JSON	- **BE(Spring):** 사용자 설정 관리 API. 개인정보보호법(PIPA)을 준수하는 데이터 처리 정책 수립 및 구현. 데이터 삭제 요청

	파일), 서비스 탈퇴 시 모든 데이터 영구 삭제(Hard Delete) 기능 제공.	시 관련 데이터를 모두 찾아 삭제하는 로직 필요.

4. 비기능적 요구사항

항목	요구사항
성능 **(Performance)**	- 모든 API의 평균 응답 시간은 200ms 이내를 목표로 한다. - 이미지 분석, STT 등 AI 처리 시간은 사용자 인지 가능한 지연 시간인 3초 이내 완료를 목표로 한다. (필요시 비동기 처리)
보안 **(Security)**	- 개인정보보호법(PIPA) 및 관련 법규를 철저히 준수한다. - 모든 사용자 데이터는 전송(TLS 1.2 이상) 및 저장(AES-256) 시 암호화한다. - SQL Injection, XSS 등 주요 웹 취약점에 대한 시큐어 코딩을 적용한다. - OAuth 2.0 기반의 안전한 인증/인가 체계를 구축한다.
확장성 **(Scalability)**	- 모든 백엔드 서비스는 Stateless하게 설계하여 수평 확장이 용이하도록 한다. - Docker와 컨테이너 오케스트레이션 도구(예: K8s)를 활용하여 트래픽 증가에 유연하게 대응할 수 있는 구조를 지향한다.
안정성 **(Reliability)**	- Kafka를 통한 비동기 처리로 특정 서비스의 장애가 전체 시스템의 장애로 이어지지 않도록 격리한다. - 주요 기능에 대한 로깅 및 모니터링 체계를 구축하여 오류 발생 시 신속하게 추적하고 대응할 수 있도록 한다.