협업방법

프로젝트 진행 절차 상세 가이드

0단계: 프로젝트 사전 준비

0-1. 아이디어 방향성 논의

☐ FigJam 브레인스토밍 세션 개최

- 기본 주제 선정
- 팀원 초대 및 화상회의 병행
- ChatGPT를 활용한 유사 서비스 및 기능 참고
- 주요 타겟 유저 정의 및 서비스 차별점 도출

0-2. 레퍼런스 조사 및 MVP 기능 도출

☐ 유사 서비스 벤치마킹 (3~5개)

☐ 핵심 MVP 기능 정의

- 로그인, 사진 업로드, 분석 결과 표시, 기록 저장 등
- 필수 기능만 선별하여 명세화

1단계: 킥오프 & 환경 구축

1-1. 킥오프 미팅 및 커뮤니케이션 환경 구축

☐ 킥오프 화상회의 진행

☐ 이메일 주소 공유 후 Slack 초대

☐ Slack 채널 구성 (#frontend, #backend, #design, #general 등)

1-2. GitHub 환경 세팅

☐ GitHub 오거나이제이션 생성 및 팀원 초대

☐ 레포지토리 구성 (`frontend`, `backend`, `docs`, `design`) - chatgpt활용해서 초기 구조 잡도록 터미널 명령어 달라고 하고 붙여넣기

- [ ] Git 브랜치 전략 수립
  - 기본 브랜치 구조: `main`, `dev`, `feature/*`, `fix/*`
  - 브랜치 보호 규칙 설정 (리뷰 필수, squash merge 적용)

1-3. 프로젝트 관리 도구 준비

- [ ] GitHub Projects Kanban 보드 구성 or Notion보드 사용
- [ ] 마일스톤 설정: MVP / 중간 데모 / 최종 제출
- [ ] Issue 템플릿 작성
- [ ] Notion 워크스페이스 개설 (규칙, FigJam, Figma 링크, 기능명세 공유 등)
- [ ] pr, 브랜치, commit message, code convention정리하기

2단계: 역할 분담 & 기술 스택 확정

2-1. 기술 스택 결정 미팅

- 프론트엔드: React / React Native / Next.js 중 선택
- 백엔드: FastAPI / Express / NestJS 중 선택
- 데이터베이스: MongoDB / Firebase / PostgreSQL 중 선택
- 배포 환경: Vercel / Railway / AWS 등 선택

2-2. 역할 분담 명확화

- PM: 일정관리, 회의 진행, 커뮤니케이션 관리
- 프론트엔드 리드: UI 설계, API 연결, 상태 관리
- 백엔드 리드: API 설계 및 구현, DB 관리
- AI/모델 담당: 모델 학습 및 API 연동 설계
- QA/DevOps 담당: 테스트 계획 수립 및 CI/CD 관리

3단계: 기능 명세 및 설계

3-1. 기능 명세 문서 작성

- [ ] Notion에 기능 정의

3-2. Figma UI/UX 설계

- [ ] 로그인, 메인, 분석결과, 기록 페이지 상세 설계
- [ ] 디자인 시스템 (색상, 폰트, 컴포넌트) 정의

3-3. API 및 데이터베이스 설계

- [ ] ERD 작성 (dbdiagram.io / Notion 활용)
- [ ] API 명세서 작성 (Swagger or Postman 기반)
    - 각 API의 요청/응답 예시 JSON 포함

4단계: 개발 및 통합

4-1. 개발 환경 세팅

- 프론트엔드: CRA, Vite 또는 Expo로 초기화
- 백엔드: FastAPI 또는 Express 초기화
- 환경 변수 설정 공유 (`.env.example`)
- Docker 도입 시 `Dockerfile`, `docker-compose.yml` 구성

4-2. 코딩 컨벤션 통일

- ESLint, Prettier 설정
- Commit Message 규칙 (`feat:`, `fix:`, `refactor:` 등)

4-3. 기능 개발 및 협업 프로세스

- 기능별 Issue 생성 → 개발 → PR → 코드 리뷰 → merge
- 백엔드: 기능 단위 개발 및 Swagger 문서 실시간 공유
- 프론트엔드: Mock API 기반 초기 UI 구성 후 실제 API 연결
- JSON 형태의 데이터 예시 미리 공유하여 개발 속도 향상

5단계: 통합 테스트 및 배포 준비

5-1. 중간 통합 테스트

- 중간 통합 테스트 기간 지정

- 프론트엔드: Mock 서버 활용하여 테스트
- 백엔드: Swagger UI, curl 활용하여 테스트
- Postman 활용 시나리오 공유

5-2. CI/CD 환경 구성

- GitHub Actions로 배포 자동화 구성
  - 프론트엔드: Vercel / Netlify 연동
  - 백엔드: Railway / Render 연동
- 빌드 실패 시 Slack 알림 연동

6단계: 마무리 및 문서화

6-1. 코드 정리 및 리팩토링

- 불필요한 코드 제거, 예외 처리 강화
- UI/UX 최종 점검 및 오류 처리 추가

6-2. 문서화 및 발표자료 제작

- 최종 README.md 작성 (설치법, 기술 스택, 기능 요약, 스크린샷)
- 발표자료 제작 (기술 블로그 글, 발표 PPT 작성)
- 시연용 영상 또는 GIF 제작 후 공유

📌 주요 참고사항

항목	도구/방법
실시간 협업	Discord, Zoom, Notion
기능 명세 공유	`docs/feature-specs.md`, Notion DB
API 공유	Swagger URL, Postman 링크
회의록 관리	Notion
일정 관리	GitHub Project, Notion Timeline

🚨 잠재 문제 상황 및 대응 방안

통합 테스트 지연 문제

- 문제 상황: 각자 기능 개발로 인해 통합 테스트 지연 및 버그 발생
- 대응 방안:
  - [ ] 중간 통합 테스트 일정을 별도로 지정 (프로젝트 중반부)
  - [ ] 프론트엔드는 Mock API 활용하여 독립적 개발 진행
  - [ ] 백엔드는 Swagger UI와 curl 활용한 자체 테스트 수행
  - [ ] 데이터 예시 JSON 파일 사전 공유로 불일치 최소화