MY-TODOLIST 프로젝트

Vibe Coding으로 완성한 풀스택 웹 애플리케이션

발표자: Emily

📋 목차

1. ✨ Vibe Coding이란?
2. 🎯 프로젝트 개요
3. 📐 기획 및 설계
4. ⚙️ 개발 환경 설정
5. 🔧 백엔드 개발
6. 🎨 프론트엔드 개발
7. 🧪 테스트
8. 🚀 배포
9. 📊 결과 및 데모
10. 💡 회고 및 배운 점

1 Vibe Coding이란?

Vibe Coding의 핵심

AI와 함께하는 새로운 개발 방식

- **AI를 코딩 파트너로 활용**하는 협업 개발 방식
- **"무엇을"** 만들지에 집중, **"어떻게"**는 AI와 함께 해결
- 자연어로 의도를 전달하면 AI가 구현 제안

기존 개발 vs Vibe Coding

구분	기존 개발	Vibe Coding
초점	코드 작성 방법	문제 정의와 요구사항
속도	느림 (모든 코드 직접 작성)	빠름 (AI가 보일러플레이트 생성)
학습 곡선	가파름 (문법, 라이브러리)	완만함 (의도 전달이 핵심)
역할	개발자가 모든 것 담당	개발자는 설계자이자 검토자

프롬프트 엔지니어링

AI에게 정확한 의도를 전달하는 방법

✅ 명확한 컨텍스트 제공
✅ 구체적인 요구사항 전달
✅ 제약조건 명시

> "팀원에게 업무를 명확하게 전달하는 것처럼"

2 프로젝트 개요

MY-TODOLIST

비전

> 이메일 인증 기반으로 개인의 할 일을 안전하게 관리할 수 있는 웹 애플리케이션

핵심 기능

- ✅ **회원가입 / 로그인** (JWT + Refresh Token)
- ✅ **할 일 CRUD** (생성, 조회, 수정, 삭제)
- ✅ **마감일 설정** (날짜 + 시간)
- ✅ **상태별 필터** (전체 / 진행중 / 완료)
- ✅ **마감 초과 알림**
- ✅ **다크모드 / 다국어 지원** (한/영/일)

기술 스택

구분	기술
Frontend	React 19, TypeScript, Vite
Backend	Node.js, Express, TypeScript
Database	PostgreSQL 17 (Supabase)
인증	JWT + Refresh Token
배포	Vercel
테스트	Jest (BE), Vitest (FE), Playwright (E2E)

개발 기간

총 4일 (2026-02-10 ~ 2026-02-13)

- **M1: 백엔드 개발** (1일)
- **M2: 프론트엔드 개발** (2일)
- **M3: 테스트 및 배포** (1일)

3 기획 및 설계

도메인 정의

핵심 엔티티

1. Member (회원)

- 이메일 인증 기반
- 비밀번호 bcrypt 암호화
- 닉네임

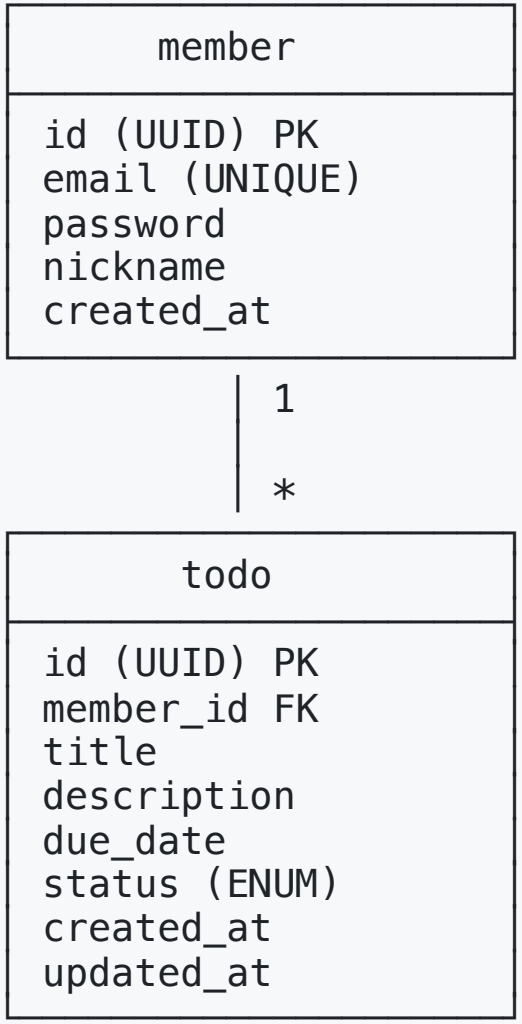
2. Todo (할 일)

- 제목 (필수)
- 설명 (선택)
- 마감일 (TIMESTAMPTZ)
- 상태 (PENDING / DONE)

비즈니스 규칙

- BR-M-01: 이메일 중복 불가
- BR-M-02: 비밀번호 최소 8자, 영문+숫자 필수
- BR-T-02: 본인 소유 할 일만 접근 가능
- BR-T-07: dueDate < 현재시각 AND status = PENDING → overdue

ERD


member
id (UUID) PK
email (UNIQUE)
password
nickname
created_at
1
*
todo
id (UUID) PK
member_id FK
title
description
due_date
status (ENUM)
created_at
updated_at

아키텍처

3-Tier 아키텍처

Presentation → React 19 + TypeScript (Vite, React Router)

⬇️

Application → Node.js + Express (JWT 미들웨어)

⬇️

Data → PostgreSQL 17 (pg 드라이버)

프로젝트 구조

```
my-todolist/
├── backend/              # Node.js + Express
│   ├── src/
│   │   ├── controllers/
│   │   ├── services/
│   │   ├── repositories/
│   │   ├── middlewares/
│   │   ├── routes/
│   │   └── config/
│   └── tests/
├── frontend/             # React + Vite
│   ├── src/
│   │   ├── pages/
│   │   ├── components/
│   │   ├── hooks/
│   │   ├── services/
│   │   └── i18n/
│   └── tests/
└── docs/                 # 설계 문서
```

4 개발 환경 설정

Vibe Coding 도구

1. Claude Code (CLI)

- 터미널 기반 AI 코딩 어시스턴트
- 파일 읽기/쓰기, 코드 생성/수정
- Git 통합 (커밋, PR 생성)

2. 프롬프트 전략

계층적 컨텍스트 관리

CLAUDE.md → 프로젝트 메모리 → 도메인 정의서 → PRD → 설계 원칙 → 실행 계획

개발 환경

- Node.js 18+
- PostgreSQL 17
- VS Code + TypeScript
- Git + GitHub

초기 세팅 프롬프트 예시

```
"React 19 + TypeScript + Vite로 프론트엔드 프로젝트를 초기화해줘.
- strict 모드 활성화
- react-router-dom 설치
- API 서비스 레이어 구조 포함
- ESLint 설정"
```

결과: 완전한 프로젝트 구조가 자동 생성됨 ✨

5 백엔드 개발

백엔드 태스크 (BE-01 ~ BE-07)

BE-01: 프로젝트 초기화

BE-02: DB 연결

BE-03: Member Repository

BE-04: 인증 API

BE-05: JWT 미들웨어

BE-06: Todo Repository

BE-07: Todo API

BE-01: 프로젝트 초기화

- Express + TypeScript 설정
- `tsconfig.json` (strict mode)
- 전역 에러 핸들러
- Health check 엔드포인트

BE-04: 인증 API

POST /api/auth/signup

- 이메일 중복 검사 (409)
- 비밀번호 정책 검증 (400)
- bcrypt 해싱 (cost 10)

POST /api/auth/login

- 이메일 조회 + bcrypt 비교
- Access Token (15분) + Refresh Token (7일) 발급

BE-05: JWT 미들웨어

```
// Authorization: Bearer <token>
middleware(req, res, next) {
  - 토큰 파싱
  - 서명 검증
  - 만료 검증
  - req.memberId 주입
}
```

BE-07: Todo API

메서드	경로	기능
GET	/api/todos	목록 조회 (overdue 계산)
POST	/api/todos	생성
PATCH	/api/todos/:id	수정 (소유권 확인)
DELETE	/api/todos/:id	삭제 (소유권 확인)
PATCH	/api/todos/:id/complete	완료 처리
PATCH	/api/todos/:id/revert	완료 취소

Vibe Coding 활용 예시

프롬프트:

```
"BE-04 태스크를 구현해줘.
- 회원가입 API: POST /api/auth/signup
  - 이메일 중복 시 409 반환
  - 비밀번호 정책: 8자 이상, 영문+숫자
  - bcrypt cost 10으로 해싱
- 로그인 API: POST /api/auth/login
  - Access Token 15분, Refresh Token 7일
  - JWT_SECRET 환경변수 사용"
```

결과

✅ `auth.service.ts`
✅ `auth.controller.ts`
✅ `auth.routes.ts`
✅ 테스트 케이스 10개

단 하나의 프롬프트로 완전한 기능 구현! 🚀

6 프론트엔드 개발

프론트엔드 태스크 (FE-01 ~ FE-07)

FE-01: 프로젝트 초기화

FE-02: API Service 레이어

FE-03: 인증 페이지

FE-04: 할 일 목록

FE-05: 생성/수정 페이지

FE-06: Overdue 시각화

FE-07: 반응형 UI

FE-02: API Service 레이어

```
// authService.ts
export const authService = {
  signup(email, password, nickname),
  login(email, password),
  logout(),
  refreshToken(),
};

// todoService.ts
export const todoService = {
  getAll(), create(dto), update(id, dto),
  remove(id), complete(id), revert(id),
};
```

책임 분리로 컴포넌트가 간결해집니다

FE-04: 할 일 목록

- `TodoListPage.tsx` - 목록 렌더링
- `TodoItem.tsx` - 개별 아이템 컴포넌트
- `useTodos` 훅 - 상태 관리
- Protected Route (미인증 시 리다이렉트)

FE-06: Overdue 시각화

```
{todo.overdue && (
  <span className="overdue-badge">
    ⚠️ 마감 초과
  </span>
)}
```

마감일이 지난 항목을 시각적으로 강조

FE-07: 반응형 UI

- 📱 모바일: 360px+
- 💻 데스크탑: 1280px+
- 👆 터치 타겟: 최소 44px
- 🌓 **다크모드** 지원
- 🌍 **다국어** 지원 (한/영/일)

상태 관리 예시

```
const useTodos = () => {
  const [todos, setTodos] = useState<Todo[]>([]);
  const [filter, setFilter] = useState<Filter>('all');

  const filteredTodos = useMemo(() => {
    switch (filter) {
      case 'pending': return todos.filter(t => t.status === 'PENDING');
      case 'done': return todos.filter(t => t.status === 'DONE');
      default: return todos;
    }
  }, [todos, filter]);

  return { filteredTodos, filter, setFilter, ... };
};
```

7 테스트

테스트 전략

총 119개 자동화 테스트

- 🔧 백엔드: **35개** (커버리지 94%)
- 🎨 프론트엔드: **84개** (커버리지 95%)
- 🔄 E2E: **25개 시나리오**

백엔드 테스트 (Jest + Supertest)

```
backend/tests/
├── unit/
│   └── app.test.ts           # 5개
└── integration/
    ├── auth.test.ts          # 10개
    └── todo.test.ts          # 20개
```

총 35개, 커버리지 94%

프론트엔드 테스트 (Vitest + Testing Library)

```
frontend/src/test/
├── services/                    # 19개
├── hooks/                       # 12개
├── components/                  # 12개
└── pages/                       # 42개
```

총 84개, 커버리지 95%

E2E 테스트 (Playwright)

SC-E2E-01: 직장인 김민준

1. ✅ 회원가입 완료
2. ✅ 로그인하여 할 일 3개 생성
   - "프로젝트 기획서" (마감: 내일)
   - "회의 준비" (마감: 오늘 오후)
   - "주간 보고서" (마감: 어제) → overdue
3. ✅ "주간 보고서"에 빨간 배지 표시 확인
4. ✅ "회의 준비" 완료 처리 → 취소선 확인
5. ✅ 완료 탭 선택 → "회의 준비"만 표시

Vibe Coding으로 테스트 작성

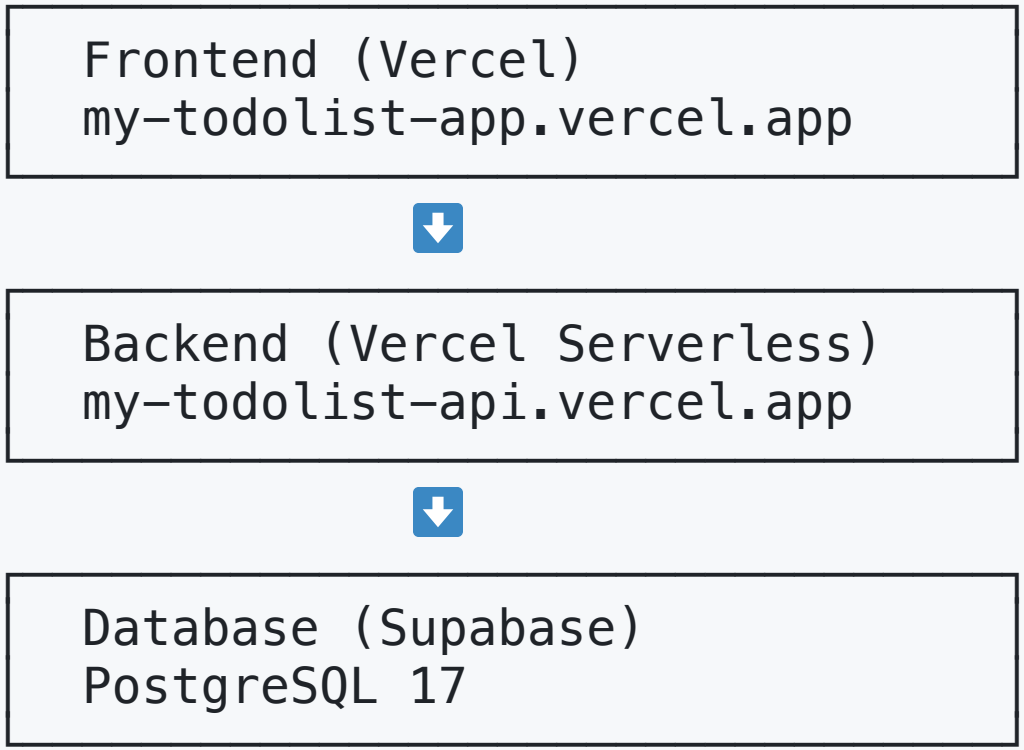
프롬프트:

```
"SC-E2E-01 시나리오를 Playwright로 구현해줘.
- 회원가입 → 로그인 → 할 일 3개 생성
- overdue 배지 확인
- 완료 처리 후 UI 변경 확인
- 탭 필터 동작 확인"
```

결과: 전체 E2E 테스트 코드 자동 생성 ✨

8 배포

Vercel + Supabase 아키텍처



배포 설정

backend/vercel.json

```
{
  "version": 2,
  "builds": [
    { "src": "api/index.ts", "use": "@vercel/node" }
  ],
  "routes": [
    { "src": "/api/(.*)", "dest": "/api/index.ts" },
    { "src": "/docs", "dest": "/api/index.ts" }
  ]
}
```

환경변수 설정

Backend (Vercel)

```
DATABASE_URL=postgresql://...  (Supabase)
JWT_SECRET=...
JWT_REFRESH_SECRET=...
CORS_ORIGIN=https://my-todolist-app.vercel.app
NODE_ENV=production
```

Frontend (Vercel)

```
VITE_API_URL=https://my-todolist-api.vercel.app
```

CI/CD 파이프라인

```
Git Push → GitHub
     ⬇️
Vercel Auto Deploy
     ⬇️
Build & Test
     ⬇️
Production Deploy
     ⬇️
Health Check
```

완전 자동화된 배포 프로세스 🚀

9 결과 및 데모

서비스 URL

프로덕션 환경

🌐 **프론트엔드:** https://my-todolist-app.vercel.app
🔧 **백엔드 API:** https://my-todolist-api.vercel.app
📚 **API 문서:** https://my-todolist-api.vercel.app/docs

주요 기능

- ✅ **회원가입 / 로그인** - JWT 토큰 발급
- ✅ **할 일 관리** - 생성, 조회, 수정, 삭제
- ✅ **상태별 필터** - 전체, 진행중, 완료 탭
- ✅ **마감 초과 알림** - 빨간 배지 표시
- ✅ **다크모드** - 시스템 테마 연동
- ✅ **다국어 지원** - 한국어, 영어, 일본어

성능 지표

지표	목표	실측값	결과
API 응답 시간	500ms 이하	평균 40ms	✅
할 일 목록 조회	1초 이하	평균 3.6ms	✅
테스트 커버리지	90% 이상	BE 94%, FE 95%	✅
자동화 테스트	100% 통과	119/119 통과	✅

모든 목표 달성! 🎯

10 회고 및 배운 점

Vibe Coding의 장점

1. 개발 속도 향상 🚀

- **4일** 만에 풀스택 애플리케이션 완성
- 보일러플레이트 코드 자동 생성
- 테스트 코드 자동화

2. 품질 향상 ✨

- **119개** 자동화 테스트
- 커버리지 **94~95%**
- TypeScript strict mode

Vibe Coding의 장점 (계속)

3. 학습 효과 📚

- 최신 기술 스택 빠르게 습득
- 베스트 프랙티스 적용
- 아키텍처 패턴 이해

> "AI가 선생님이자 동료 개발자"

주요 도전 과제

1. 프롬프트 엔지니어링

- 명확한 요구사항 정의 필요
- 컨텍스트 관리의 중요성
- 단계별 검증 필요

2. AI 생성 코드 검토

- 보안 취약점 확인 필수
- 비즈니스 로직 검증
- 테스트 코드 신뢰성 확인

주요 도전 과제 (계속)

3. 통합 과정

- 프론트엔드-백엔드 API 연동
- 배포 환경 설정
- CORS 정책 관리

> "AI가 생성한 코드도 반드시 검토가 필요합니다"

현재 MVP vs 향후 개선

✅ 현재 MVP 범위

- 기본 CRUD 기능
- 인증/인가
- 마감 초과 알림

📋 향후 개선 계획

- 할 일 정렬 기능
- 🔔 실시간 알림
- 📊 통계 대시보드
- 👥 협업 기능 (공유 할 일)

Vibe Coding 권장 사항

1. 📐 명확한 설계 먼저

- 도메인 정의서
- PRD (요구사항 문서)
- 아키텍처 다이어그램

2. 🔄 단계별 진행

- 작은 태스크로 분할
- 각 단계 검증
- 점진적 개선

Vibe Coding 권장 사항 (계속)

3. 🧪 테스트 자동화

- TDD 접근
- 높은 커버리지 유지
- E2E 시나리오 작성

4. 📚 문서화

- 프로젝트 메모리 관리
- README 작성
- API 문서화 (Swagger)

핵심 메시지

Vibe Coding은...

✨ **개발 속도**를 높이고
✨ **코드 품질**을 향상시키며
✨ **학습 효과**를 극대화합니다

> "어떻게"보다 "무엇을"에 집중하면,
> 더 나은 제품을 만들 수 있습니다.

감사합니다! 🙏

질문이 있으시면 편하게 해주세요.

참고 자료

📁 프로젝트 문서

- 도메인 정의서, PRD, 아키텍처 다이어그램, 실행 계획서

💻 코드 저장소

- GitHub: github.com/sehee-jeong/my-todolist

🌐 배포 환경

- 프론트엔드: https://my-todolist-app.vercel.app
- 백엔드: https://my-todolist-api.vercel.app
- API 문서: https://my-todolist-api.vercel.app/docs

기술 스택 문서

- **React 19:** react.dev
- **TypeScript:** typescriptlang.org
- **Express:** expressjs.com
- **PostgreSQL:** postgresql.org
- **Vercel:** vercel.com/docs