WorkLife Dashboard - Phase 1 제품 요구사항 문서 (PRD)

버전: 1.0
날짜: 2025-01-15
작성자: 개발팀
상태: 초안

1. 프로젝트 개요

1.1 제품 소개

WorkLife Dashboard는 직장인을 위한 종합 생산성 도구 플랫폼입니다. Phase 1에서는 핵심 금융 관리 도구(가계부, 급여계산기)와 기본 인증 시스템을 구축합니다.

1.2 프로젝트 목표

- 직장인의 일상적인 금융 관리를 돕는 웹 애플리케이션 개발
- 안전한 사용자 인증 및 데이터 관리 시스템 구축
- 모던한 다크 테마 UI/UX 제공
- 확장 가능한 아키텍처 설계로 향후 기능 추가 용이성 확보

1.3 성공 지표

- [ ] 사용자 100명 이상 가입
- [ ] 일일 활성 사용자(DAU) 30명 이상
- [ ] 페이지 로딩 시간 3초 이내
- [ ] 모바일 반응형 디자인 100% 구현
- [ ] 핵심 기능 버그 0건

2. 기술 스택

2.1 기술 구성

프론트엔드

- **프레임워크**: React 18.x
- **언어**: TypeScript (권장) / JavaScript
- **UI 라이브러리**: Mantine 7.x
- **스타일링**: Mantine 내장 스타일 시스템 + CSS Modules
- **상태 관리**: Redux Toolkit
- **차트**: Mantine Charts (Recharts 기반)

- **폼 관리**: Mantine Form + Zod
- **HTTP 클라이언트**: Axios
- **빌드 도구**: Vite
- **아이콘**: Tabler Icons

백엔드

- **런타임**: Node.js 20.x LTS
- **프레임워크**: Express.js
- **데이터베이스**: PostgreSQL 15.x
- **ORM**: Prisma
- **인증**: JWT + bcrypt
- **검증**: Zod

개발 도구 및 배포

- **버전 관리**: Git
- **개발 환경**: VSCode + Claude Code
- **테스트**: Jest + React Testing Library
- **API 테스트**: Postman/Thunder Client
- **배포**: Vercel (프론트) + Railway/Render (백엔드)

2.2 프로젝트 구조

```
worklife-dashboard/
├── client/                 # 프론트엔드 React 애플리케이션
│   ├── public/
│   ├── src/
│   │   ├── components/     # 재사용 가능한 UI 컴포넌트
│   │   │   ├── common/     # 공통 컴포넌트
│   │   │   ├── layout/     # 레이아웃 컴포넌트
│   │   │   └── charts/     # 차트 컴포넌트
│   │   ├── features/       # 기능별 컴포넌트
│   │   │   ├── auth/       # 인증 관련
│   │   │   ├── accountBook/ # 가계부
│   │   │   └── salary/     # 급여계산기
│   │   ├── hooks/          # 커스텀 React 훅
│   │   ├── pages/          # 페이지 컴포넌트
│   │   ├── services/       # API 서비스 함수
│   │   ├── store/          # Redux 스토어 설정
```

```
│   │   ├── theme/        # Mantine 테마 설정
│   │   └── utils/        # 유틸리티 함수
│   └── package.json
│
├── server/               # 백엔드 Node.js 애플리케이션
│   ├── src/
│   │   ├── controllers/  # 라우트 컨트롤러
│   │   ├── middlewares/  # Express 미들웨어
│   │   ├── models/       # 데이터베이스 모델
│   │   ├── routes/       # API 라우트
│   │   ├── services/     # 비즈니스 로직
│   │   ├── utils/        # 유틸리티 함수
│   │   └── validators/   # 입력 검증
│   ├── prisma/
│   │   └── schema.prisma
│   └── package.json
│
├── docs/                 # 문서
├── scripts/              # 유틸리티 스크립트
└── README.md
```

3. UI/UX 디자인 시스템

3.1 Mantine 테마 설정

javascript

```tsx
// 다크 테마 기본 설정
const theme = {
  colorScheme: 'dark',
  primaryColor: 'blue',
  defaultRadius: 'md',

  colors: {
    dark: [
      '#C9C9C9',
      '#B8B8B8',
      '#828282',
      '#696969',
      '#424242',
      '#3B3B3B',
      '#2E2E2E',  // 메인 배경
      '#252525',
      '#1F1F1F',
      '#141414',
    ],
  },

  components: {
    Card: {
      defaultProps: {
        withBorder: true,
        radius: 'md',
        padding: 'lg',
      },
    },
    Paper: {
      defaultProps: {
        radius: 'md',
        p: 'md',
      },
    },
  },
};
```

3.2 주요 UI 컴포넌트 구성

대시보드 레이아웃

- **AppShell**: Mantine의 레이아웃 시스템
- **Navbar**: 사이드바 네비게이션 (데스크톱)
- **Header**: 상단 헤더

- **Drawer**: 모바일 메뉴

통계 및 데이터 시각화

- **StatsGrid**: 통계 카드 그리드
- **AreaChart**: 추이 차트
- **DonutChart**: 카테고리별 비중
- **BarChart**: 월별 비교

폼 및 입력

- **NumberInput**: 금액 입력 (한국 원화 포맷)
- **DatePicker**: 날짜 선택 (한국어 지원)
- **Select**: 카테고리 선택
- **SegmentedControl**: 수입/지출 토글

3.3 반응형 디자인

```javascript
// Mantine 브레이크포인트
{
  xs: '36em',  // 576px
  sm: '48em',  // 768px
  md: '62em',  // 992px
  lg: '75em',  // 1200px
  xl: '88em',  // 1408px
}
```

4. 기능 요구사항

4.1 인증 시스템

4.1.1 회원가입

우선순위: P0 (필수)

기능 요구사항:

- 이메일/비밀번호 기반 회원가입
- 이메일 유효성 검증 (형식 및 중복 체크)
- 비밀번호 강도 요구사항 (최소 8자, 대문자 1개, 숫자 1개)
- 약관 동의

- Mantine의 PasswordInput 컴포넌트 사용

UI 컴포넌트:

```
javascript

// Mantine 컴포넌트 사용
- TextInput (이메일)
- PasswordInput (비밀번호, 강도 표시기 포함)
- Checkbox (약관 동의)
- Button (회원가입)
- Notification (성공/실패 메시지)
```

4.1.2 로그인

우선순위: P0 (필수)

기능 요구사항:

- 이메일/비밀번호 인증
- JWT 토큰 생성 (액세스: 15분, 리프레시: 7일)
- 자동 로그인 옵션
- 로그인 실패 시 에러 메시지
- 5회 실패 시 계정 잠금

UI 컴포넌트:

```
javascript

- Paper (로그인 폼 컨테이너)
- TextInput (이메일)
- PasswordInput (비밀번호)
- Checkbox (자동 로그인)
- Button (로그인)
- Alert (에러 메시지)
```

4.2 가계부

4.2.1 거래 관리

우선순위: P0 (필수)

기능 요구사항:

- 수입/지출 거래 추가

- 거래 수정/삭제
- 거래 검색 및 필터링
- 페이지네이션 (20개씩)
- CSV 가져오기/내보내기

Mantine 컴포넌트 구성:

```
javascript

// 거래 입력 폼
- SegmentedControl (수입/지출 선택)
- NumberInput (금액, 한국 원화 포맷)
- Select (카테고리)
- DatePickerInput (날짜)
- Textarea (메모)
- Switch (반복 거래)

// 거래 목록
- Table (데스크톱)
- Card + Stack (모바일)
- Pagination (페이지네이션)
- ActionIcon (수정/삭제)
```

4.2.2 통계 대시보드

우선순위: P1 (높음)

기능 요구사항:

- 월별 수입/지출 요약
- 카테고리별 지출 분석
- 일별 지출 추이
- 전월 대비 비교

차트 구성:

```
javascript

// Mantine Charts 사용
- StatsGrid (요약 통계)
- DonutChart (카테고리별 비중)
- AreaChart (일별 추이)
- BarChart (월별 비교)
```

4.3 급여계산기

4.3.1 급여 입력

우선순위: P0 (필수)

기능 요구사항:

- 연봉/월급 선택 입력
- 부양가족 수 선택
- 비과세액 입력
- 상여금 계산 지원

Mantine 컴포넌트:

```
javascript

- SegmentedControl (연봉/월급)
- NumberInput (급여액, 천 단위 구분)
- Slider + NumberInput (부양가족 수)
- NumberInput (비과세액)
- Button (계산하기)
```

4.3.2 계산 결과

우선순위: P1 (높음)

표시 항목:

- 월 실수령액
- 공제 내역 상세
- 연간 예상 실수령액
- PDF 다운로드

UI 구성:

```
javascript

- Card (결과 컨테이너)
- StatsRing (실수령액 시각화)
- Table (공제 내역)
- Button.Group (PDF/공유)
```

5. 데이터베이스 스키마

5.1 주요 테이블

```sql
```

```sql
-- 사용자 테이블
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  name VARCHAR(100),
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW(),
  last_login TIMESTAMP,
  is_active BOOLEAN DEFAULT true,
  theme_preference VARCHAR(20) DEFAULT 'dark'
);

-- 카테고리 테이블
CREATE TABLE categories (
  id SERIAL PRIMARY KEY,
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  name VARCHAR(50) NOT NULL,
  type ENUM('income', 'expense') NOT NULL,
  icon VARCHAR(50) DEFAULT 'IconWallet',
  color VARCHAR(20) DEFAULT 'blue',
  is_default BOOLEAN DEFAULT false,
  created_at TIMESTAMP DEFAULT NOW()
);

-- 거래 테이블
CREATE TABLE transactions (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  category_id INTEGER REFERENCES categories(id),
  type ENUM('income', 'expense') NOT NULL,
  amount DECIMAL(12, 2) NOT NULL,
  date DATE NOT NULL,
  description TEXT,
  is_recurring BOOLEAN DEFAULT false,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- 예산 테이블
CREATE TABLE budgets (
  id SERIAL PRIMARY KEY,
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  category_id INTEGER REFERENCES categories(id),
  amount DECIMAL(12, 2) NOT NULL,
  month DATE NOT NULL,
```

```
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- 급여 계산 기록 테이블
CREATE TABLE salary_calculations (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  gross_salary DECIMAL(12, 2) NOT NULL,
  net_salary DECIMAL(12, 2) NOT NULL,
  calculation_data JSONB NOT NULL,
  created_at TIMESTAMP DEFAULT NOW()
);
```

6. API 명세

6.1 기본 설정

- 기본 URL: `https://api.worklife-dashboard.com`
- 버전: `/v1`
- 형식: JSON
- 인증: Bearer Token

6.2 주요 엔드포인트

인증 API

```
POST   /api/auth/register     # 회원가입
POST   /api/auth/login        # 로그인
POST   /api/auth/logout       # 로그아웃
POST   /api/auth/refresh      # 토큰 갱신
GET    /api/auth/me           # 현재 사용자
PUT    /api/auth/profile      # 프로필 수정
```

가계부 API

```
# 거래
GET    /api/transactions      # 거래 목록
POST   /api/transactions      # 거래 추가
PUT    /api/transactions/:id  # 거래 수정
DELETE /api/transactions/:id  # 거래 삭제
GET    /api/transactions/stats # 통계

# 카테고리
GET    /api/categories        # 카테고리 목록
POST   /api/categories        # 카테고리 추가
PUT    /api/categories/:id    # 카테고리 수정
DELETE /api/categories/:id    # 카테고리 삭제

# 예산
GET    /api/budgets           # 예산 목록
POST   /api/budgets           # 예산 설정
PUT    /api/budgets/:id       # 예산 수정
GET    /api/budgets/status    # 예산 현황
```

급여계산기 API

```
POST   /api/salary/calculate  # 급여 계산
GET    /api/salary/history    # 계산 기록
POST   /api/salary/save       # 계산 저장
GET    /api/salary/export/:id # PDF 다운로드
```

7. Mantine 컴포넌트 활용 전략

7.1 레이아웃 구성

```
javascript
```

```
// AppShell을 활용한 메인 레이아웃
const MainLayout = () => {
  return (
    <AppShell
      header={{ height: 60 }}
      navbar={{ width: 300, breakpoint: 'sm' }}
      padding="md"
    >
      <AppShell.Header>
        {/* 헤더 컨텐츠 */}
      </AppShell.Header>

      <AppShell.Navbar>
        {/* 네비게이션 메뉴 */}
      </AppShell.Navbar>

      <AppShell.Main>
        {/* 메인 컨텐츠 */}
      </AppShell.Main>
    </AppShell>
  );
};
```

7.2 다크 테마 설정

javascript

```
// MantineProvider로 다크 테마 적용
import { MantineProvider, createTheme } from '@mantine/core';

const theme = createTheme({
  colorScheme: 'dark',
  fontFamily: 'Pretendard, -apple-system, BlinkMacSystemFont, sans-serif',
  primaryColor: 'blue',
  defaultRadius: 'md',
});

function App() {
  return (
    <MantineProvider theme={theme} defaultColorScheme="dark">
      {/* 앱 컨텐츠 */}
    </MantineProvider>
  );
}
```

7.3 모바일 반응형 처리

```javascript
// useMediaQuery 훅 활용
import { useMediaQuery } from '@mantine/hooks';

const isMobile = useMediaQuery('(max-width: 48em)');

// 조건부 렌더링
return isMobile ? (
  <Stack>{/* 모바일 레이아웃 */}</Stack>
) : (
  <Grid>{/* 데스크톱 레이아웃 */}</Grid>
);
```

8. 개발 일정

8주 개발 계획

1-2주차: 프로젝트 설정 및 기반 구축

- 개발 환경 설정
- Mantine UI 설정 및 테마 구성
- 데이터베이스 설계 및 구축
- 기본 라우팅 설정

3-4주차: 인증 시스템

- 백엔드 인증 API 구현
- 로그인/회원가입 UI (Mantine)
- JWT 토큰 관리
- 보호된 라우트 구현

5-6주차: 가계부 개발

- 거래 CRUD API 구현
- 가계부 UI 컴포넌트 개발
- 통계 대시보드 구현
- Mantine Charts 적용

7주차: 급여계산기

- 급여계산 로직 구현
- 계산기 UI 개발
- 결과 표시 및 PDF 생성

8주차: 테스트 및 배포

- 통합 테스트
- 버그 수정 및 최적화
- 배포 환경 구축
- 문서화

9. 비기능 요구사항

9.1 성능

- 페이지 로드 시간: < 3초
- API 응답 시간: < 500ms (95% 요청)
- 동시 사용자 1000명 이상 지원
- Mantine 컴포넌트 lazy loading

9.2 보안

- HTTPS 강제 적용
- 비밀번호 해싱 (bcrypt, 10+ rounds)
- SQL 인젝션 방지
- XSS 보호
- CSRF 토큰 구현
- Rate limiting (분당 100 요청)

9.3 사용성

- 모바일 반응형 디자인 (320px - 1920px)
- 크로스 브라우저 호환성
- 웹 접근성 (WCAG 2.1 Level AA)
- 한국어 지원
- 다크/라이트 테마 전환

9.4 신뢰성

- 99.5% 가동률 목표
- 자동 백업 (일 단위)
- 에러 로깅 및 모니터링
- 우아한 에러 처리

10. 테스트 전략

10.1 단위 테스트

- 커버리지 목표: 80%
- 유틸리티 함수 테스트
- 계산 로직 테스트
- API 엔드포인트 테스트

10.2 통합 테스트

- API 통합 테스트
- 데이터베이스 작업 테스트
- 인증 플로우 테스트

10.3 E2E 테스트

- 핵심 사용자 경로
- 크로스 브라우저 테스트
- 모바일 반응형 테스트

11. 배포 계획

11.1 환경 구성

- 개발: 로컬
- 스테이징: staging.worklife-dashboard.com
- 프로덕션: worklife-dashboard.com

11.2 CI/CD 파이프라인

- GitHub Actions 자동화 테스트
- main 브랜치 자동 배포
- 데이터베이스 마이그레이션

- 환경 변수 관리

12. 모니터링 및 분석

12.1 애플리케이션 모니터링

- 에러 추적 (Sentry)
- 성능 모니터링
- API 응답 시간
- 데이터베이스 쿼리 성능

12.2 사용자 분석

- Google Analytics 4
- 사용자 참여도 측정
- 기능 사용 추적
- 전환 퍼널 분석

13. 위험 평가

높음

- 데이터 유출 또는 보안 취약점
- 데이터베이스 손상 또는 데이터 손실
- 높은 부하에서 성능 저하

중간

- Mantine 업데이트로 인한 호환성 문제
- 브라우저 호환성 이슈
- 계산 정확도 오류

낮음

- UI/UX 개선 필요
- 기능 범위 확대
- 문서화 부족

14. 성공 지표 (KPI)

Phase 1 KPI

- 사용자 가입률: 주 50명 이상
- 일일 활성 사용자: 30명 이상
- 거래 입력: 월 1000건 이상
- 급여 계산: 월 500건 이상
- 7일 사용자 유지율: 40% 이상
- 시스템 가동률: 99.5% 이상

15. 향후 계획 (Phase 2+)

- 소셜 로그인 (구글, 네이버, 카카오)
- 모바일 앱 개발 (React Native)
- 시간 관리 도구 추가
- 파일 변환 도구 추가
- AI 기반 인사이트 제공
- 디바이스 간 데이터 동기화
- 협업 기능
- 프리미엄 구독 모델

문서 버전 이력

- v1.0 - 2025-01-15 - 초기 작성 (Mantine 기반)