TRD.md

고정하기

TRD.md
TRD: 퍼스널 브랜딩 랜딩페이지 빌더
Technical Requirements Document

---

1. 기술 개요

1.1 시스템 아키텍처
```mermaid
graph TB
    subgraph "Client Side"
        A[Next.js App] --> B[React Components]
        B --> C[Zustand Store]
        B --> D[Framer Motion]
        B --> E[Tailwind CSS]
    end

    subgraph "Server Side"
        F[Next.js API Routes]
        G[Edge Functions]
    end

    subgraph "External Services"
        H[Google Sheets API]
        I[AWS S3]
        J[Vercel Analytics]
    end
```

```
    A → F
    F → H
    F → I
    A → G
    A → J
```

1.2 기술 스택 상세

레이어	기술	버전	선택 이유
Framework	Next.js	14.2+	App Router, RSC, 최적화된 번들링
Language	TypeScript	5.0+	타입 안정성, 개발 생산성
Styling	Tailwind CSS	3.4+	유틸리티 우선, 빠른 개발
UI Library	shadcn/ui	latest	커스터마이징 가능, 접근성
Animation	Framer Motion	11.0+	선언적 애니메이션, 성능
State	Zustand	4.5+	간단함, 번들 크기 작음
Form	React Hook Form	7.50+	성능, 유연성
Validation	Zod	3.22+	TypeScript 통합, 스키마 검증
DnD	@dnd-kit	6.1+	접근성, 성능, 터치 지원
Editor	Lexical	0.12+	확장성, Facebook 지원
HTTP	Axios	1.6+	인터셉터, 에러 핸들링
Date	date-fns	3.0+	트리셰이킹, 불변성

---

2. 시스템 요구사항

2.1 개발 환경
```json
{
  "node": ">=20.0.0",
  "npm": ">=10.0.0",
  "os": ["macOS", "Windows 10+", "Ubuntu 20.04+"],
  "ide": "Cursor (recommended) or VS Code",
  "browser": "Chrome (latest) for development"
}
```

2.2 프로덕션 환경
- **Hosting**: Vercel Edge Network
- **CDN**: Vercel Global CDN
- **Domain**: Custom domains with SSL
- **Analytics**: Vercel Analytics + Google Analytics 4

---

3. 프로젝트 구조

3.1 디렉토리 구조
```
personal-brand-builder/
├── app/                    # Next.js App Router
│   ├── (auth)/             # 인증 관련 페이지
│   │   ├── login/
│   │   └── register/
│   ├── (builder)/          # 빌더 관련 페이지
│   │   ├── editor/
│   │   │   └── [id]/
│   │   └── preview/
│   ├── (marketing)/        # 마케팅 페이지
│   │   ├── page.tsx        # 홈페이지
│   │   ├── pricing/
│   │   └── templates/
│   ├── api/                # API Routes
│   │   ├── auth/
│   │   ├── pages/
│   │   └── upload/
│   └── dashboard/          # 대시보드
│
├── components/             # React 컴포넌트
│   ├── builder/            # 빌더 전용 컴포넌트
│   │   ├── Canvas/
│   │   ├── ComponentLibrary/
│   │   ├── PropertyPanel/
│   │   └── Toolbar/
```

```
│   ├── blocks/              # 페이지 블록 컴포넌트
│   │   ├── hero/
│   │   ├── about/
│   │   ├── portfolio/
│   │   └── contact/
│   ├── ui/                  # shadcn/ui 컴포넌트
│   └── shared/              # 공통 컴포넌트
│
├── lib/                     # 유틸리티 및 설정
│   ├── api/                 # API 클라이언트
│   ├── hooks/               # Custom Hooks
│   ├── store/               # Zustand stores
│   ├── utils/               # 유틸리티 함수
│   └── validators/          # Zod 스키마
│
├── styles/                  # 스타일 파일
│   ├── globals.css
│   └── themes/
│
├── types/                   # TypeScript 타입 정의
│   ├── builder.ts
│   ├── components.ts
│   └── api.ts
│
├── public/                  # 정적 파일
│   ├── fonts/
│   ├── images/
│   └── icons/
│
└── tests/                   # 테스트 파일
    ├── unit/
    ├── integration/
    └── e2e/
```

3.2 주요 설정 파일

```typescript
// next.config.js
```

```
const nextConfig = {
  experimental: {
    optimizeCss: true,
    optimizePackageImports: ['framer-motion', '@dnd-kit/sortable']
  },
  images: {
    domains: ['res.cloudinary.com', 's3.amazonaws.com'],
    formats: ['image/avif', 'image/webp']
  },
  compiler: {
    removeConsole: process.env.NODE_ENV === 'production'
  }
};

// tsconfig.json
{
  "compilerOptions": {
    "target": "ES2022",
    "lib": ["dom", "dom.iterable", "esnext"],
    "strict": true,
    "paths": {
      "@/*": ["./src/*"],
      "@/components/*": ["./src/components/*"],
      "@/lib/*": ["./src/lib/*"]
    }
  }
}
```

---

4. 핵심 기능 구현

4.1 드래그 앤 드롭 시스템

4.1.1 기술 선택
- **라이브러리**: @dnd-kit/sortable
- **이유**: 접근성, 성능, 모바일 지원, 애니메이션

4.1.2 구현 구조
```typescript
// types/builder.ts
interface DraggableComponent {
  id: string;
  type: ComponentType;
  props: Record<string, any>;
  children?: DraggableComponent[];
}

interface DropZone {
  id: string;
  accepts: ComponentType[];
  maxItems?: number;
}

// components/builder/DragDropContext.tsx
export const DragDropProvider: React.FC = ({ children }) => {
  const sensors = useSensors(
    useSensor(PointerSensor, {
      activationConstraint: {
        distance: 8,
      },
    }),
    useSensor(KeyboardSensor, {
      coordinateGetter: sortableKeyboardCoordinates,
    })
  );

  return (
    <DndContext sensors={sensors} onDragEnd={handleDragEnd}>
      {children}
    </DndContext>
  );
};
```

4.2 상태 관리

4.2.1 Zustand Store 구조
```typescript
// lib/store/builder.store.ts
interface BuilderState {
  // State
  components: DraggableComponent[];
  selectedComponentId: string | null;
  isDirty: boolean;
  history: HistoryState;

  // Actions
  addComponent: (component: DraggableComponent) => void;
  updateComponent: (id: string, updates: Partial<DraggableComponent>) =>
void;
  deleteComponent: (id: string) => void;
  selectComponent: (id: string | null) => void;
  undo: () => void;
  redo: () => void;
  save: () => Promise<void>;
}

export const useBuilderStore = create<BuilderState>()(
  devtools(
    persist(
      immer((set, get) => ({
        // Implementation
      })),
      {
        name: 'builder-storage',
        partialize: (state) => ({ components: state.components })
      }
    )
  )
);
```

4.3 컴포넌트 시스템

4.3.1 컴포넌트 레지스트리
```typescript
// lib/components/registry.ts
export const ComponentRegistry = {
  hero: {
    component: dynamic(() => import('@/components/blocks/hero/HeroBloc
k')),
    icon: LayoutIcon,
    category: 'layout',
    defaultProps: {
      title: 'Welcome to my site',
      subtitle: 'This is a subtitle',
      backgroundType: 'gradient'
    },
    schema: HeroPropsSchema
  },
  // ... other components
} as const;

// 타입 안전성을 위한 유틸리티
export type ComponentType = keyof typeof ComponentRegistry;
export type ComponentProps<T extends ComponentType> =
  z.infer<typeof ComponentRegistry[T]['schema']>;
```

4.4 실시간 미리보기

4.4.1 Canvas 구현
```typescript
// components/builder/Canvas/Canvas.tsx
export const Canvas: React.FC = () => {
  const { components } = useBuilderStore();
  const [deviceView, setDeviceView] = useState<'desktop' | 'tablet' | 'mobil
e'>('desktop');

  return (
```

```
    <div className={cn(
      "relative bg-gray-50 overflow-auto",
      deviceView === 'mobile' && "max-w-[375px] mx-auto",
      deviceView === 'tablet' && "max-w-[768px] mx-auto"
    )}>
      <iframe
        src="/preview"
        className="w-full h-full border-0"
        sandbox="allow-scripts allow-same-origin"
      />
    </div>
  );
};
```

4.5 성능 최적화

4.5.1 코드 스플리팅
```typescript
// 동적 임포트를 통한 번들 크기 최적화
const HeroBlock = dynamic(
  () => import('@/components/blocks/hero/HeroBlock'),
  {
    loading: () => <BlockSkeleton />,
    ssr: false
  }
);
```

4.5.2 메모이제이션
```typescript
// components/builder/PropertyPanel/PropertyField.tsx
export const PropertyField = memo(({
  field,
  value,
  onChange
}: PropertyFieldProps) => {
  const handleChange = useCallback((newValue: any) => {
```

```
    onChange(field.name, newValue);
  }, [field.name, onChange]);

  // Render logic
}, (prevProps, nextProps) => {
  return prevProps.value === nextProps.value &&
         prevProps.field.name === nextProps.field.name;
});
```

---

5. API 설계

5.1 RESTful API 엔드포인트

```typescript
// API 라우트 구조
/api/auth/
  POST   /login      # 로그인
  POST   /logout     # 로그아웃
  POST   /register   # 회원가입
  GET    /me         # 현재 사용자 정보

/api/pages/
  GET    /           # 페이지 목록
  POST   /           # 페이지 생성
  GET    /:id        # 페이지 상세
  PUT    /:id        # 페이지 수정
  DELETE /:id        # 페이지 삭제
  POST   /:id/publish # 페이지 퍼블리시

/api/upload/
  POST   /image      # 이미지 업로드
  DELETE /image/:id  # 이미지 삭제

/api/templates/
  GET    /           # 템플릿 목록
```

```
  GET    /:id        # 템플릿 상세
```

5.2 API 응답 포맷
```typescript
// types/api.ts
interface ApiResponse<T = any> {
  success: boolean;
  data?: T;
  error?: {
    code: string;
    message: string;
    details?: any;
  };
  meta?: {
    page?: number;
    limit?: number;
    total?: number;
  };
}

// 에러 코드
enum ErrorCode {
  UNAUTHORIZED = 'UNAUTHORIZED',
  FORBIDDEN = 'FORBIDDEN',
  NOT_FOUND = 'NOT_FOUND',
  VALIDATION_ERROR = 'VALIDATION_ERROR',
  INTERNAL_ERROR = 'INTERNAL_ERROR'
}
```

---

6. 데이터 모델

6.1 주요 스키마

```typescript
```

```typescript
// lib/validators/page.schema.ts
export const PageSchema = z.object({
  id: z.string().uuid(),
  userId: z.string().uuid(),
  title: z.string().min(1).max(100),
  slug: z.string().regex(/^[a-z0-9-]+$/),
  components: z.array(ComponentSchema),
  theme: ThemeSchema,
  seo: SeoSchema,
  customDomain: z.string().optional(),
  isPublished: z.boolean(),
  publishedAt: z.date().optional(),
  createdAt: z.date(),
  updatedAt: z.date()
});

// lib/validators/component.schema.ts
export const ComponentSchema = z.object({
  id: z.string(),
  type: z.enum(['hero', 'about', 'portfolio', 'contact']),
  props: z.record(z.any()),
  order: z.number(),
  isVisible: z.boolean().default(true),
  animations: AnimationSchema.optional()
});
```

6.2 로컬 스토리지 구조

```typescript
// localStorage 키 구조
interface LocalStorageKeys {
  'builder:current-page': PageData;
  'builder:preferences': UserPreferences;
  'builder:history': HistoryState;
  'builder:autosave': AutoSaveData;
}
```

````
---

## 7. 보안 요구사항

### 7.1 인증 및 인가
```typescript
// lib/auth/middleware.ts
export const authMiddleware = async (req: NextRequest) => {
  const token = req.cookies.get('auth-token');

  if (!token) {
    return NextResponse.redirect('/login');
  }

  try {
    const payload = await verifyJWT(token);
    req.headers.set('x-user-id', payload.userId);
    return NextResponse.next();
  } catch (error) {
    return NextResponse.redirect('/login');
  }
};
```

### 7.2 입력 검증
```typescript
// 모든 사용자 입력에 대한 검증
export const sanitizeInput = (input: string): string => {
  return DOMPurify.sanitize(input, {
    ALLOWED_TAGS: ['b', 'i', 'em', 'strong', 'a'],
    ALLOWED_ATTR: ['href', 'target']
  });
};
```

### 7.3 CORS 설정
```typescript
// next.config.js
````
````

```
async headers() {
  return [
    {
      source: '/api/:path*',
      headers: [
        { key: 'Access-Control-Allow-Credentials', value: 'true' },
        { key: 'Access-Control-Allow-Origin', value: process.env.ALLOWED_OR
IGIN },
        { key: 'Access-Control-Allow-Methods', value: 'GET,POST,PUT,DELET
E,OPTIONS' },
      ],
    },
  ];
}
```

---

8. 성능 최적화

8.1 프론트엔드 최적화

8.1.1 번들 최적화
```typescript
// 컴포넌트 레이지 로딩
const ComponentLibrary = dynamic(
  () => import('@/components/builder/ComponentLibrary'),
  {
    loading: () => <ComponentLibrarySkeleton />,
    ssr: false
  }
);

// 트리 셰이킹을 위한 배럴 파일 제거
// ❌ import { Button, Input, Card } from '@/components/ui';
// ✅ import { Button } from '@/components/ui/button';
```

8.1.2 이미지 최적화
```typescript
// components/common/OptimizedImage.tsx
export const OptimizedImage: React.FC<ImageProps> = ({
  src,
  alt,
  priority = false,
  ...props
}) => {
  return (
    <Image
      src={src}
      alt={alt}
      loading={priority ? 'eager' : 'lazy'}
      placeholder="blur"
      blurDataURL={generateBlurDataURL(src)}
      quality={85}
      {...props}
    />
  );
};
```

8.1.3 렌더링 최적화
```typescript
// Virtual scrolling for component library
import { VirtuosoGrid } from 'react-virtuoso';

export const ComponentGrid = () => {
  return (
    <VirtuosoGrid
      data={components}
      itemContent={(index, component) => (
        <ComponentCard key={component.id} {...component} />
      )}
      overscan={2}
    />
  );
```

```
};
```

8.2 백엔드 최적화

8.2.1 캐싱 전략
```typescript
// lib/cache/redis.ts
export const cacheStrategy = {
  pages: {
    ttl: 3600, // 1 hour
    tags: ['pages'],
  },
  templates: {
    ttl: 86400, // 24 hours
    tags: ['templates'],
  },
  user: {
    ttl: 1800, // 30 minutes
    tags: ['user'],
  }
};
```

8.2.2 데이터베이스 쿼리 최적화
```typescript
// Prisma query optimization
const getPageWithComponents = async (pageId: string) => {
  return prisma.page.findUnique({
    where: { id: pageId },
    include: {
      components: {
        orderBy: { order: 'asc' },
        select: {
          id: true,
          type: true,
          props: true,
          order: true
```

```
      }
    }
  }
});
};
```

---

9. 테스트 전략

9.1 테스트 구조
```typescript
// 테스트 파일 구조
tests/
├── unit/              # 단위 테스트
│   ├── components/
│   ├── hooks/
│   └── utils/
├── integration/       # 통합 테스트
│   ├── api/
│   └── features/
└── e2e/               # E2E 테스트
    ├── auth.spec.ts
    ├── builder.spec.ts
    └── publish.spec.ts
```

9.2 테스트 예시

9.2.1 컴포넌트 테스트
```typescript
// tests/unit/components/HeroBlock.test.tsx
describe('HeroBlock', () => {
  it('renders with default props', () => {
    render(<HeroBlock {...defaultProps} />);
    expect(screen.getByText(defaultProps.title)).toBeInTheDocument();
  });
```

```
  it('applies animations when enabled', () => {
    const { container } = render(
      <HeroBlock {...defaultProps} enableAnimation />
    );
    expect(container.firstChild).toHaveClass('motion-safe:animate-fadeIn');
  });
});
```

9.2.2 E2E 테스트
```typescript
// tests/e2e/builder.spec.ts
test('complete page creation flow', async ({ page }) => {
  // 로그인
  await page.goto('/login');
  await page.fill('[name="email"]', 'test@example.com');
  await page.fill('[name="password"]', 'password');
  await page.click('button[type="submit"]');

  // 새 페이지 생성
  await page.click('text="Create New Page"');
  await page.fill('[name="title"]', 'My Portfolio');

  // 컴포넌트 추가
  await page.dragAndDrop(
    '[data-component="hero"]',
    '[data-dropzone="canvas"]'
  );

  // 저장 및 퍼블리시
  await page.click('text="Save"');
  await page.click('text="Publish"');

  // 확인
  await expect(page.locator('.success-toast')).toBeVisible();
});
```

---

10. 배포 및 CI/CD

10.1 CI/CD 파이프라인
```yaml
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '20'
          cache: 'npm'
      - run: npm ci
      - run: npm run test:ci
      - run: npm run lint
      - run: npm run type-check

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: vercel/action@v1
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.ORG_ID }}
          vercel-project-id: ${{ secrets.PROJECT_ID }}
```

```

### 10.2 환경 변수
```bash
# .env.production
NEXT_PUBLIC_APP_URL=https://app.personalbrand.io
NEXT_PUBLIC_API_URL=https://api.personalbrand.io
DATABASE_URL=postgresql://...
REDIS_URL=redis://...
JWT_SECRET=...
GOOGLE_CLIENT_ID=...
GOOGLE_CLIENT_SECRET=...
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...
AWS_S3_BUCKET=...
```

---

11. 모니터링 및 로깅

11.1 에러 트래킹
```typescript
// lib/monitoring/sentry.ts
Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: 1.0,
  integrations: [
    new Sentry.BrowserTracing(),
    new Sentry.Replay()
  ]
});
```

11.2 애널리틱스
```typescript
// lib/analytics/index.ts

```
export const trackEvent = (
  eventName: string,
  properties?: Record<string, any>
) => {
  // Google Analytics
  gtag('event', eventName, properties);

  // Mixpanel
  mixpanel.track(eventName, properties);

  // Custom analytics
  if (process.env.NODE_ENV === 'production') {
    fetch('/api/analytics', {
      method: 'POST',
      body: JSON.stringify({ event: eventName, properties })
    });
  }
};
```

---

12. 개발 가이드라인

12.1 코딩 컨벤션
```typescript
// ESLint 설정
{
  "extends": [
    "next/core-web-vitals",
    "plugin:@typescript-eslint/recommended",
    "prettier"
  ],
  "rules": {
    "@typescript-eslint/no-unused-vars": "error",
    "@typescript-eslint/no-explicit-any": "warn",
    "react-hooks/exhaustive-deps": "error"
  }
```

```
}
```

12.2 커밋 컨벤션
```bash
# Conventional Commits
feat: 새로운 기능 추가
fix: 버그 수정
docs: 문서 수정
style: 코드 포맷팅
refactor: 코드 리팩토링
perf: 성능 개선
test: 테스트 추가/수정
chore: 빌드 업무, 패키지 매니저 설정 등
```

---

13. 부록

13.1 참고 문서
- [Next.js Documentation](https://nextjs.org/docs)
- [Tailwind CSS Documentation](https://tailwindcss.com/docs)
- [Framer Motion Documentation](https://www.framer.com/motion/)
- [@dnd-kit Documentation](https://docs.dndkit.com/)

13.2 도구 및 확장
- **VS Code/Cursor Extensions**:
  - Tailwind CSS IntelliSense
  - ESLint
  - Prettier
  - TypeScript Error Lens
  - GitLens

13.3 문제 해결 가이드
```markdown
## 일반적인 문제 해결
```

```
### 빌드 오류
- `rm -rf .next node_modules package-lock.json`
- `npm install`
- `npm run build`

### TypeScript 오류
- `npm run type-check`
- VS Code: Cmd+Shift+P → "TypeScript: Restart TS Server"

### 성능 이슈
- Chrome DevTools Performance 탭 확인
- React DevTools Profiler 사용
- `npm run analyze` 실행하여 번들 크기 확인
```

문서 버전: 1.0
최종 수정일: 2025-07-31
작성자: Technical Lead
다음 리뷰: 2025-08-07