UI 구현 및 정보 구조(IA) 명세서

본 문서는 프로젝트의 디렉토리 구조가 실제 서비스의 정보 구조와 어떻게 매핑되는지를 정의한 명세서입니다.

1. 프로젝트 디렉토리 구조(Directory Tree)

```
UI구현
├─ index.html
├─ search.html
├─ home.html
├─ cart.html
├─ lounge.html
├─ my_page.html
├─ product_list.html
├─ login.html
├─ events.html
├─ signup.html
├─ css
│  ├─ cart.css
│  ├─ home.css
│  ├─ index.css
│  ├─ login.css
│  ├─ my_page.css
│  ├─ product_list.css
│  ├─ reset.css
│  └─ style.css
├─ images
│  ├─ cart_in1.jpg
│  ├─ cart_in2.jpg
│  ├─ cart_in3.jpg
│  ├─ cart_in4.jpg
│  ├─ cart_rec1.jpg
│  ├─ cart_rec2.jpg
│  ├─ cart_rec3.jpg
│  ├─ cart_rec4.jpg
│  ├─ cart_rec5.jpg
│  ├─ cart_rec6.jpg
│  ├─ Favicon.png
│  ├─ Frame 77.jpg
│  ├─ home1.jpg
│  ├─ home2.jpg
│  ├─ home3.jpg
│  ├─ home4.jpg
│  ├─ home5.jpg
│  ├─ home6.jpg
│  ├─ home_Mbanner1.jpg
│  ├─ home_Mbanner10.jpg
│  ├─ home_Mbanner11.jpg
│  ├─ home_Mbanner2.jpg
│  ├─ home_Mbanner3.jpg
```

```
│   ├─ home_Mbanner4.jpg
│   ├─ home_Mbanner5.jpg
│   ├─ home_Mbanner6.jpg
│   ├─ home_Mbanner7.jpg
│   ├─ home_Mbanner8.jpg
│   ├─ home_Mbanner9.jpg
│   ├─ kakao.jpg
│   ├─ logo.jpg
│   ├─ logo.svg
│   ├─ naver.jpg
│   ├─ pro1.jpg
│   ├─ pro10.jpg
│   ├─ pro2.jpg
│   ├─ pro3.jpg
│   ├─ pro4.jpg
│   ├─ pro5.jpg
│   ├─ pro6.jpg
│   ├─ pro7.jpg
│   ├─ pro8.jpg
│   └─ pro9.jpg
├─ js
   ├─ script_all.js
   └─ script_home.js
```

2. 정보 구조(IA) 및 기능 매핑

파일/폴더명	정보 구조상 역할	주요 구현 기능
home.html	메인 홈 화면	전체 서비스의 입구이자 핵심 세션 나열
index.html	메뉴 페이지	전체 메뉴를 확인하고 접근 할 수 있도록 반영
login.html	로그인 페이지	회원가입 페이지로 접근, 로그인
product_list.html	상품 페이지	판매 상품을 진열
cart.html	장바구니 페이지	카트에 넣은 상품을 확인, 결제
my_page.html	마이 페이지	유저의 정보 확인, 변경
search.html	검색 페이지	상품을 검색 할 수 있도록 반영
lounge.html	라운지 페이지	유저간의 소통 게시판
event.html	이벤트 페이지	모든 이벤트를 한번에 확인
signup.html	회원가입 페이지	서비스에 회원가입
css/	디자인 시스템	UI의 시각적 일관성 및 레이아웃 정의
js/	상호작용 로직	사용자 조작에 따른 동적 반응(이벤트) 제어
image/	시각적 콘텐츠	브랜드 이미지 및 아이콘 정보 제공

3. 구조 설계 원칙

- **자원 분리**
    - HTML(구조), CSS(표현), JS(행위)를 폴더별로 엄격히 분리
    - 유지보수 용이, 기능별 코드 충돌 최소화
    - 프로젝트 확장 시 새로운 페이지/기능 추가 용이
- **직관적 위계 및 명명 규칙 준수**
    - 폴더와 파일명을 통해 자원의 역할을 즉시 파악 가능
    - 표준 명명 규칙과 일관된 구조 유지
- **논리적 매핑**
    - 물리적 파일 시스템의 위계가 기획된 정보 구조(IA)와 1:1로 대응
    - 서비스 구조와 개발 구조 일치 → 유지보수 및 디버깅 효율성 향상
- **모듈화와 재사용성 고려**
    - CSS, JS를 기능별로 모듈화하여 공통 기능 재사용 가능
    - 페이지별 스크립트/스타일과 공통 모듈을 분리 → 코드 중복 최소화
- **확장성과 유연성 확보**
    - 신규 기능이나 페이지 추가 시 기존 구조를 크게 변경하지 않고 확장 가능
    - CSS 변수, 공통 클래스, JS 함수 재사용으로 프로젝트 규모 확대 용이

4. 기술적 특징

- jQuery를 활용하여 효율적인 DOM 조작 및 인터랙션을 구현했습니다.
    - 클릭, 마우스오버, 스크롤 등 사용자 이벤트에 따라 페이지 요소를 동적으로 제어
    - 코드가 간결하고 재사용 가능하도록 모듈화
- 외부 라이브러리와 사용자 정의 스크립트를 분리하여 관리 효율성을 높였습니다.
    - 라이브러리 업데이트 시 사용자 코드에 영향 최소화
    - 기능별 파일 분리로 유지보수 용이
- reset.css와 layout.css를 분리하여 스타일 수정 시 효율성을 극대화함
    - 브라우저 기본 스타일 초기화와 페이지 레이아웃 스타일을 분리
    - 스타일 충돌 최소화 및 수정 범위 명확화
- 이벤트 중심의 상호작용 로직 구현
    - 사용자 입력 및 페이지 상태 변화에 따라 동적 컨텐츠 렌더링
    - 불필요한 DOM 재렌더링 최소화로 성능 최적화
- 코드 재사용성과 유지보수를 고려한 모듈화
    - 공통 기능(JS 함수, CSS 클래스)을 별도 파일로 분리

- 페이지별 스크립트와 공통 스크립트를 명확히 구분