

# 이수 교육 내역서

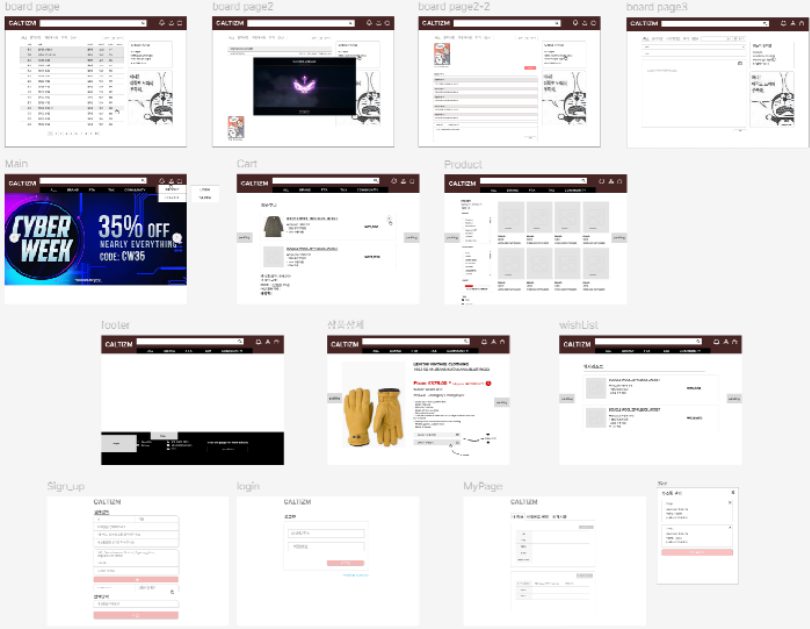
성명	이상민	교육과정명	자바(JAVA) 웹개발자 양성과정(클라우드)
교육기간	2024-07-01 ~ 2025-01-03 (6개월,920시간)		

전공교과 이수내역	
교육과목	주요 교육 내용
HTML5	개발환경 설정 및 HTML 개요   HTML Basic   Firefox, Chrome 등 최신 웹 브라우저 설치   명확한 정보 구조 설계   HTML 5의 요소 성격의 구분 섹션 요소들과 아웃라인 알고리즘: section, nav 등 텍스트 시멘틱요소: mark, time 등   멀티미디어 관련 요소: video, audio   사용자의 입력을 검증하고 도와주는 서식 요소: form
CSS3	CSS의 개요 및 적용방법   CSS3 선택자(Selector) 배치에 관한 속성 (float vs position)   Border 관련 속성, Background 관련 속성 Text Effects   User Interface 관련 속성, Animation, Transition 속성 Media Query   Multicolumn layout   Web font
자바스크립트	자바 스크립트 기초   자바 스크립트 반복문, 제어문   함수 생성 방법   클래스 기초   객체생성하기 Ajax 프로그래밍
React	컴포넌트만들기   JSX 및 JSX 스타일링   이벤트처리하기   props 와 state 관리   useRef   useEffect(마운트,언마운트,업데이트)   useReducer   context 사용하기   페이지 라우팅
SQL	오라클 11g 설치   데이터베이스 개요   정규화   데이터 모델링   ERM (객체관계 모델링)   DML, DDL, DCL 문의 이해   테이블 생성 시 제약 조건 Sequence, index, default   Select 로 자료 추출, 내장함수, group by Join, subquery,   Transaction 관리   View 파일 사용   사용자 계정 관리 - 권한과 보안
Java	자바 기반 프로그래밍에 대한 개괄적 설명, 환경설정 식별자와 데이터형, 연산자, 제어문   배열 선언 및 사용, 클래스와 객체, 클래스 간의 관계   인터페이스와 추상클래스 자바의 유용한 클래스   예외처리, 다중 데이터 처리   자바 입출력   컬렉션 스레드 - 동기화, 자원 공유   네트워크의 이해, Socket
JDBC	JDBC의 이해   Connection, Statement, Prepared Statement, Result Set DAO 및 DTO 패턴   Database Metadata 와 ResultSetMetaData, 자바 미니 프로젝트 작성
JSP/Servlet	웹 프로그래밍의 이해, 자바와 웹 프로그래밍 웹 개발 환경 구축   Servlet 프로그래밍 HTTP 각 방식 별 구현 메서드   JSP 페이지의 구성 요소 웹 어플리케이션 디렉터리 구성과 URL 매핑 웹 어플리케이션의 배포 - war 파일을 이용 JSP 기본 객체의 속성(Attribute) 사용하기 페이지 모듈화와 요청 흐름 제어 - JSP Action tag 에러 처리, 쿠키와 세션   자바빈(JavaBean) 웹 어플리케이션에서 DB 처리   회원관리 등의 프로그램 작성 표현 언어(Expression Language)   표준 태그 라이브러리(JSTL) MVC 패턴 구현   파일 업로드 구현   필터의 구현 미니 프로젝트 작성 ex) 쇼핑몰, 물류관리 등
Spring Framework	스프링 프레임워크의 개요   스프링 환경설정, sts 설치 및 사용 스프링으로 객체 조립하기: 스프링 설정 만들기 스프링 DI를 이용한 객체 생성   애노테이션 기반 의존 자동 연결 위한 설정   빈 라이프사이클과 Bean 범위   스프링 AOP Weaving 방식   스프링 MVC   맨드 객 체 값 검증과 에러 메시지   HTTP 세션 사용하기   Exception 처리   스프링 JDBC 지원 MyBatis 연동 지원   웹 MVC에서 AOP 적용하기 로그인, Transaction 등
Spring Data Jpa	Entity 만들기 JPA CRUD

	영속성 관리하기 ( 변경감지) JPA 쿼리매서드 작성하기 JPQL 작성하기
Mybatis Framework	SqlSession 사용하기 mapper 작성하기 동적쿼리 사용하기 Spring @Transactional 트랜잭션 사용하기
Thymeleaf	text, utext, url 링크, 반복, 조건부평가, 연산, 리터럴, SpringEL, 자바스크립트 인라인
Git/GitHub	Git 개요/설치   Git 특징   로컬 저장소/원격 저장소   Git bash 이용   Git 기본 명령 어(init, add, commit 등)   브랜치 관리 및 병합 충돌 해결   GitHub 이용한 원격 저장소 관리 (clone, push, pull) Eclipse 에서 GitHub 사용법
DOM 프로그래밍	DOM 객체 이해하기   이벤트 처리방법   DOM 객체의 속성, 스타일 다루기 노드 순회하기 및 조작하기   계산기 예제 만들기   갤러리 예제 만들기 버블링과 캡처링   트리거   Ajax 다루기   애니메이션 처리 게시글 관리 예제
jQuery	DOM 객체를 jQuery 객체로 래핑하기   노드 Selectors jQuery 객체를 이용한 속성, 스타일 변경하기 노드 순회하기   노드 조작하기   Ajax 다루기   Effect 다루기 이벤트 객체 사용하기   트리거 이용하기
AWS	AWS 에 EC2 서버 만들기 RDS 만들고 연결하기 IAM 사용자 만들기 S3 버킷사용하기 도커 - 컨테이너 생성하기 도커 -Docker 파일만들기 GithubActions 을 활용한 CICD
프로젝트	프로젝트 준비 프로젝트 기획 및 UI 설계 요구사항분석, Git 활용 프로젝트관리 및 형상관리 UI 구현 DB 모델링설계 프레임워크활용 풀스택 프로젝트 프로젝트 발표 및 평가

# 프로젝트 수행이력

프로젝트 명: CALTIZM (온라인 쇼핑몰 프로젝트)

프로젝트 구성원		https://github.com/Sbh-Acorn/Final-Project
이상민 (본인)	프론트단(Thymeleaf) 설계 및 구축, 메인 페이지 구현	
소병학 (팀장)	검색기능 구현, 웹 배포, 환율 데이터 추출	
송병화	DB 설계 및 구축, 게시글 작성 기능 구현, 로그인 / 회원가입 기능 구현	
백바울	웹 크롤링 및 데이터 추출, 계산 기능 구현, 상품리스트 필터링 기능 구현	
용지민	DB 설계 및 구축, 위시리스트 기능 구현, 알림 기능 구현	
송수빈	게시글 기능 구현, 게시판 카테고리별 목록 조회 기능 구현	
수행기간		2024.12.01 ~ 2025.01.03

개발 PLATFORM	프로젝트 개요
▶ <b>Server</b> : Tomcat v10.1.33 Server	<ul style="list-style-type: none"> <li>▶ 해외 직구 쇼핑몰 상품 예상가 도출 페이지</li> <li>▶ 다양한 기능 구현               <ul style="list-style-type: none"> <li>- 온라인 쇼핑몰 페이지 데이터 크롤링</li> <li>- 상품 조회 및 필터링</li> <li>- 로그인, 회원가입, 마이 페이지</li> <li>- 상품 상세 페이지</li> <li>- 위시리스트, 장바구니, 알림</li> <li>- 게시판 페이지</li> </ul> </li> </ul>
▶ <b>DB</b> : mySQL	
▶ <b>Client</b> : HTML5, CSS, Vanilla Js	
▶ <b>H/W</b> : Windows 11	

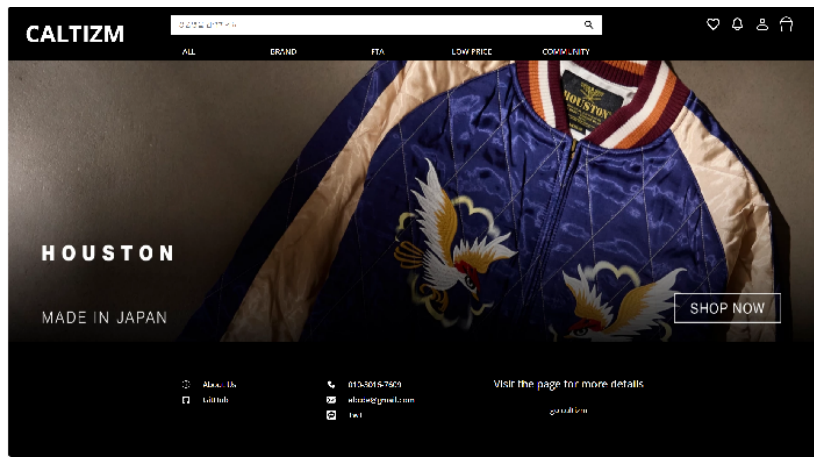
# 프로젝트 특징

## 온라인 쇼핑몰 페이지 데이터 크롤링

- ▶ 실제 온라인 쇼핑몰 CULTIZM 으로부터 데이터를 크롤링
- ▶ 브랜드, 상품, FTA URL 데이터 수집

## 메인 페이지

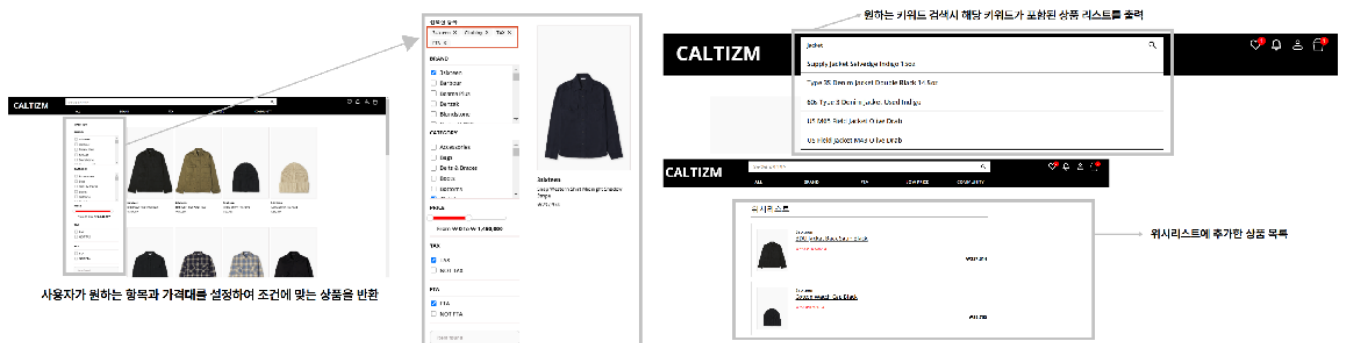
- ▶ 루트 접속시 메인 페이지로 이동
- ▶ 자동 및 클릭 슬라이드 배너



( 메인 화면 )

## 상품 조회 및 필터링

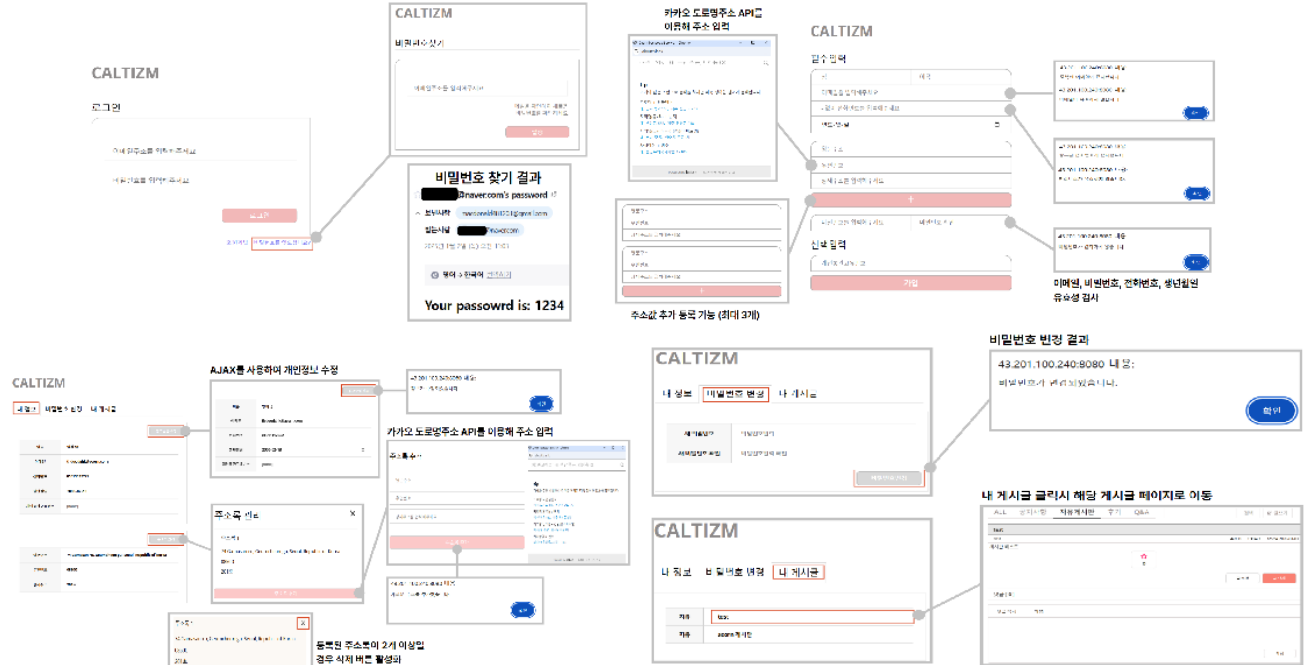
- ▶ 크롤링한 데이터를 바탕으로 상품 조회
- ▶ 키워드 입력을 통한 필터링 기능



( 필터링 및 검색 화면 )

## 로그인, 회원가입, 마이 페이지

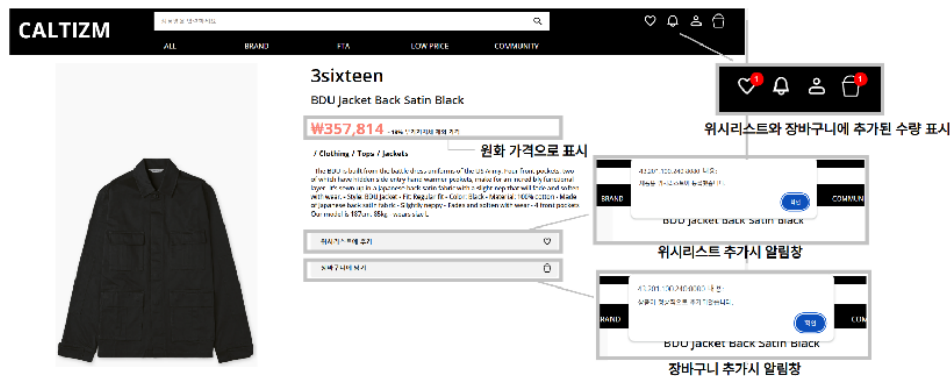
- ▶ 로그인 기능 및 비밀번호 찾기 기능
- ▶ 입력 정보 유효성 검사 및 주소 API 를 사용한 회원가입 기능
- ▶ 정보 수정 및 비밀번호 변경이 가능한 마이페이지



( 로그인, 회원가입, 마이페이지 화면 )

## 상품 상세 페이지

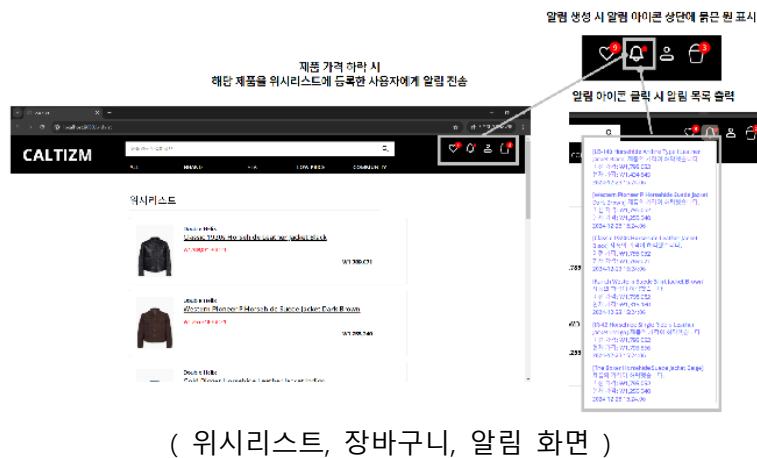
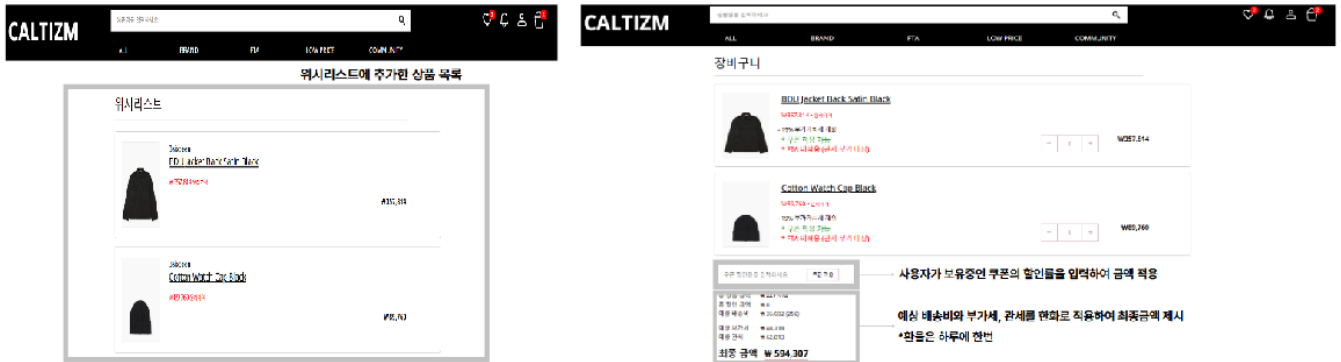
- ▶ 환율 계산을 통한 원화 가격 표시
- ▶ 위시리스트 및 장바구니 상품 담기 기능



( 상품 상세 페이지 화면 )

## 위시리스트, 장바구니, 알림

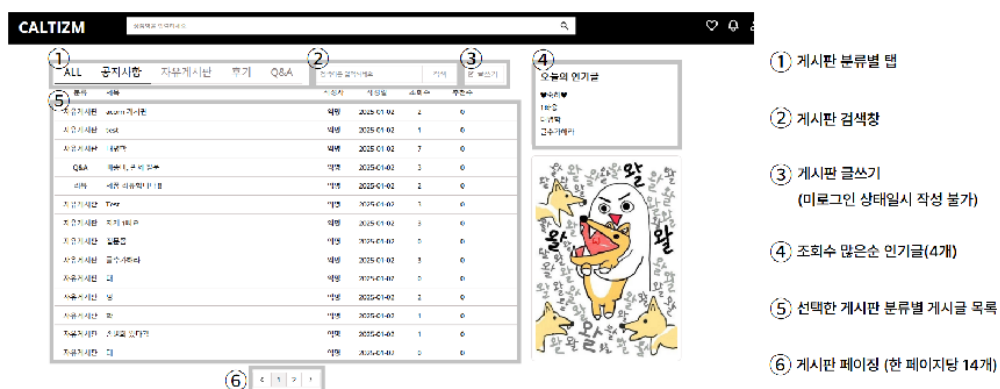
- ▶ 위시리스트 페이지를 통해 관심 상품 목록 확인
- ▶ 할인율 수치를 입력해 예상 최종 금액을 장바구니 페이지를 통해 확인
- ▶ 위시리스트 제품의 가격 및 재고 변동을 알림 기능을 통해 사용자에게 전송

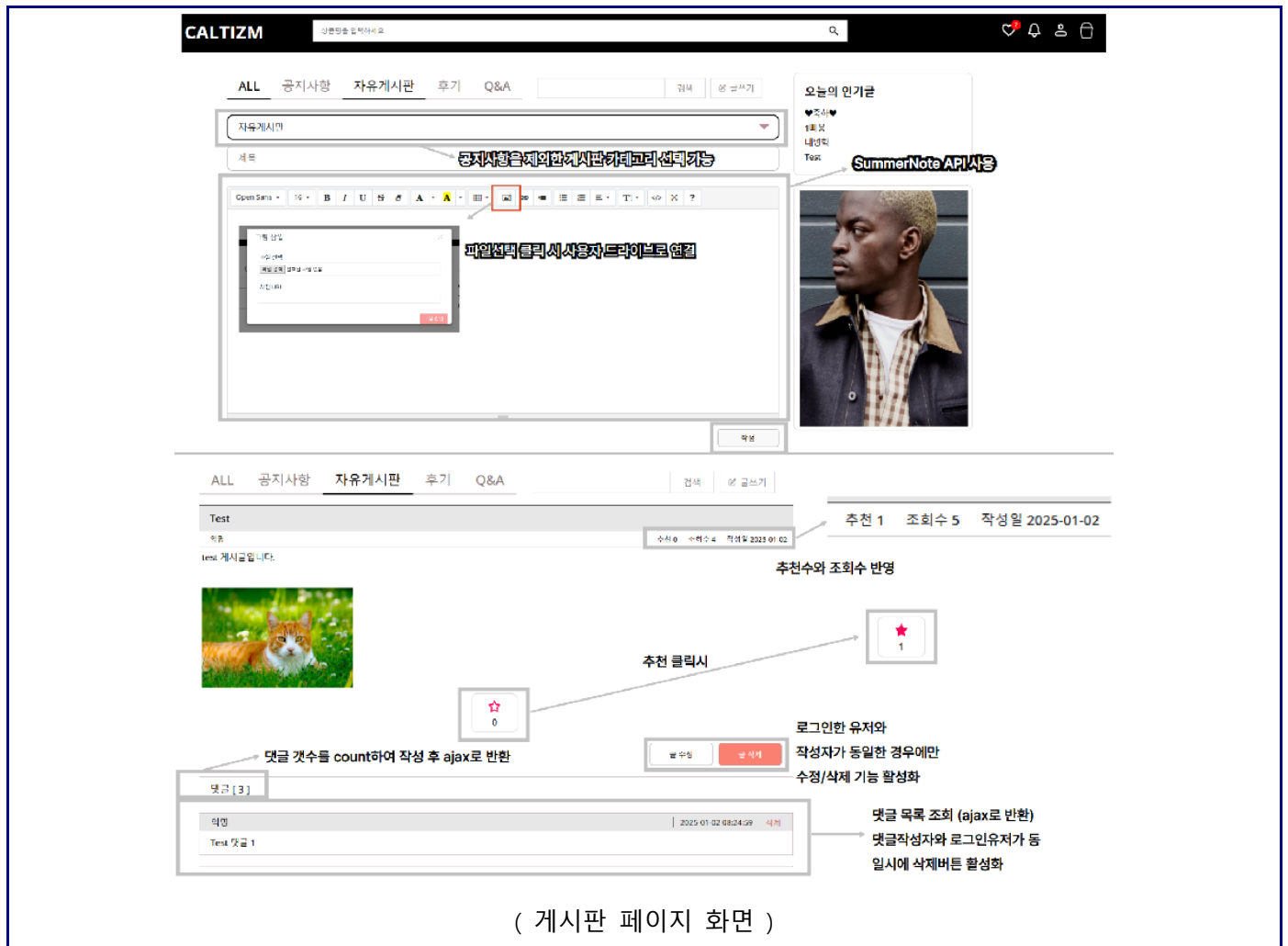


( 위시리스트, 장바구니, 알림 화면 )

## 게시판 페이지

- ▶ 게시물 검색, 분류, 페이징 기능
- ▶ 게시물 작성, 조회, 댓글 기능





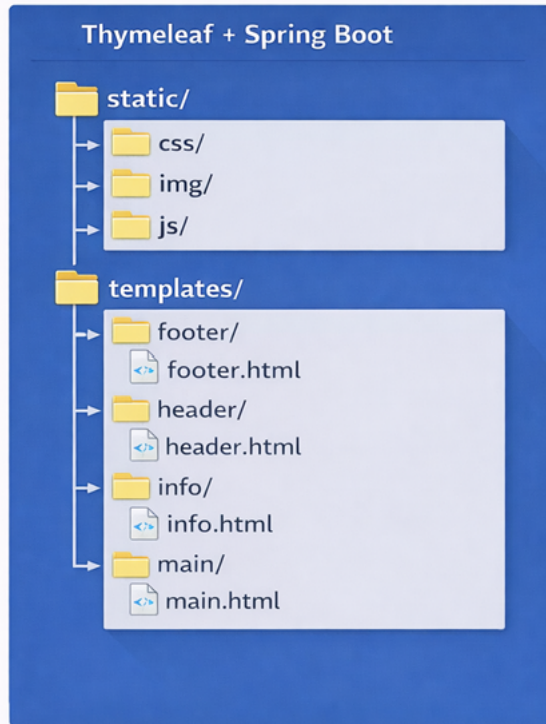
<p>프로젝트 후기</p>	<p>한 달간 프로젝트에서 프론트엔드 전체와 메인 페이지 구현을 담당하며, 하루 단위로 세분화된 일정 관리 방식으로 개발을 진행하였다.</p> <p>프로젝트 초기, 팀원과의 기능 구현과 관련된 충분한 사전 협의가 이루어지지 않아 일부 화면을 추가 구현해야 하는 상황이 발생하였다. 이 일을 계기로 팀원들과 기능 구현에 대한 협의 과정을 가졌다.</p> <p>UI 는 전체적으로 초기 설계안에 맞게 일관되게 구현되었으며, 헤더와 같은 중복 템플릿의 경우 Fragment 로 분리하여 중복을 최소화하고 유지보수성을 개선하였다. 그리고 메인 페이지의 배너 이미지는 본래 프론트단에서 정적으로 처리하였으나, 후의 배포 및 확장을 고려하여 백단으로부터 이미지 목록을 모델에 담아 밀어주는 방식으로 구조를 변경하였다.</p> <p>이번 프로젝트를 통해 기존에 부족하다고 느꼈던 프론트단의 경험을 많이 쌓았으며, 협업 과정에서 팀원들과의 소통의 중요성을 배웠다.</p>
----------------	---

# 프로젝트 수행이력

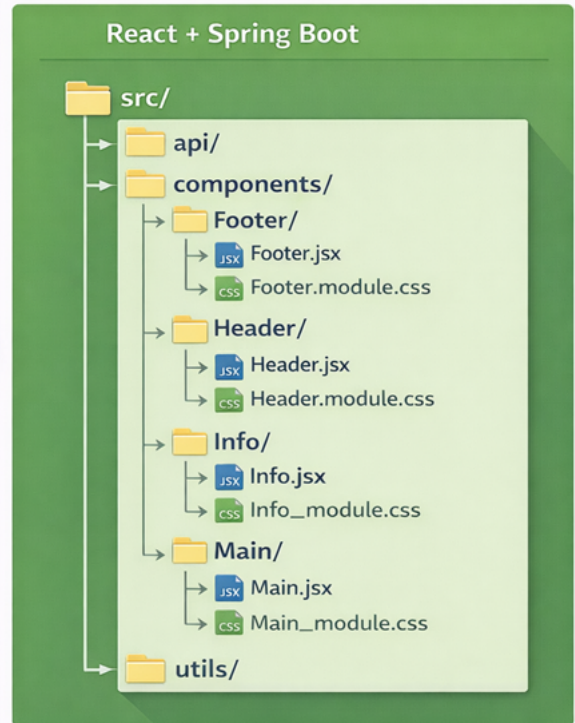
프로젝트 명: CALTIZM SPA 프로젝트

git address : <https://github.com/tkdals1144/React-migration-project>

**Before**



**After**



수행기간

2025.11.25 ~ 2025.12.24

개발 PLATFORM	프로젝트 개요
▶ <b>Server</b> : Tomcat v10.1.33 Server	<p>▶ 기존 CALTIZM 프로젝트 아키텍처 리팩터링 ( SSR -&gt; CSR )</p> <p>기존 Thymeleaf + Spring Boot 기반의 SSR 구조로 구현된 해외 직구 쇼핑몰 CALTIZM 을 React + Spring Boot 기반의 CSR(SPA) 구조로 전환한 프로젝트입니다. 데이터 흐름 개선, 유지보수성 향상, React 실전 감각 향상, 그리고 무더진 Spring Boot 코딩 감각 개선을 목표로 진행한 개인 프로젝트입니다.</p>
▶ <b>DB</b> : mySQL	
▶ <b>Client</b> : HTML5, CSS, React Js	
▶ <b>H/W</b> : Windows 11	



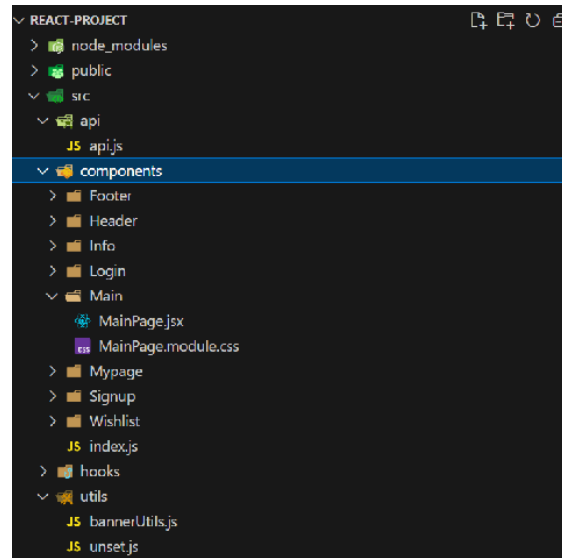
# 프로젝트 특징

## 렌더링 구조 전환

- ▶ **Before** : 서버에서 HTML 을 생성하여 반환 (Thymeleaf SSR)
- ▶ **After** : 화면 전환을 클라이언트 라우팅으로 처리 (React CSR)

## 기능 단위 컴포넌트 설계

- ▶ 역할별로 프론트 컴포넌트를 분리
- ▶ 컴포넌트 단위로 모듈화



( 디렉토리 구조 )

## 프록시 설정

- ▶ CORS 에러 방지를 위한 프록시 설정
- ▶ 프론트엔드와 백엔드 간의 통신을 설정

```
1 import { defineConfig } from "vite";
2 import react from "@vitejs/plugin-react";
3
4 // https://vite.dev/config/
5 export default defineConfig({
6   plugins: [react()],
7   server: {
8     proxy: {
9       "/api": {
10         target: "http://localhost:8080",
11         changeOrigin: false,
12       },
13     },
14   },
15 });
```

( 프록시 설정 화면 )

## 백엔드 역할 재정립

- ▶ Controller 책임 변경 (HTML 반환 -> JSON 데이터 반환)
- ▶ DTO 중심 데이터 전달 구조 확립

```
@Controller no usages tkdals1144
@ResponseBody
public class SignupController {

    @Autowired 4 usages
    SignupService service;
    @Autowired 1 usage
    PasswordEncoder passwordEncoder;

    public record AuthResponse(boolean success, String message) { } 4 usages tkdals1144

    @PostMapping("/signup") no usages tkdals1144
    public ResponseEntity<AuthResponse> register(@RequestBody SignupRequestDTO user) {

        // 0 메일 중복 검사
        if(service.userCheck(user.getEmail())) {
            return ResponseEntity.badRequest().body(new AuthResponse( success: false, message: "중복된 이메일이 존재합니다."));
        }
        // 전화번호 중복 검사
        if(!service.telDupCheck(user.getPhone_number())) {
            return ResponseEntity.badRequest().body(new AuthResponse( success: false, message: "중복된 전화번호가 존재합니다."));
        }
        String encodePassword = passwordEncoder.encode(user.getPassword());
        user.setPassword(encodePassword);

        service.registUser(user);
        service.registUserAddr(user);

        return ResponseEntity.ok().body(new AuthResponse( success: true, message: "회원가입이 완료되었습니다."));
    }
}
```

( 변경된 회원가입 Controller )

## 보안 및 인증 구조 강화

- ▶ Spring Security 적용
- ▶ SecurityConfig 를 생성하여 들어오는 요청에 대한 권한 확인

```
@Configuration no usages tkdals1144
@EnableWebSecurity
public class SecurityConfig {

    @Bean no usages tkdals1144
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf( CsrfConfigurer<HttpSecurity> csrf -> csrf.disable() ) // REST API로 연결 시 CSRF 보호
            .authorizeHttpRequests( AuthorizationManagerRequestMatcher auth -> auth
                .requestMatchers( ...patterns: "/error" ).permitAll()
                .requestMatchers( ...patterns: "/api/**" ).permitAll()
                .anyRequest().authenticated()
            );
        return http.build();
    }
}
```

```

@Bean no usages tkdals1144
public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.addAllowedOrigin("http://localhost:5173"); // 프론트엔드 주소
    configuration.addAllowedMethod("*");
    configuration.addAllowedHeader("*");
    configuration.setAllowCredentials(true); // 세션/쿠키 허용

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration(pattern: "/*", configuration);

    return source;
}

```

(SecurityConfig 로직)

## UI 밀접 로직 프론트엔드로 이동

- ▶ UI 와 밀접한 단순 계산, 가공, 분기 로직을 프론트로 이동
- ▶ 이동한 로직을 utils / hooks 로 분리하여 책임 명확화

```

5 function useLogin(refreshAuth) {
6     const [error, setError] = useState(null);
7     const [isLoading, setIsLoading] = useState(false);
8     const navigate = useNavigate();
9
10    const login = async (email, password) => {
11        setIsLoading(true);
12        setError(null);
13
14        try {
15            //1. 통신 로직과 await 비동기 처리
16            const res = await axios.post("/api/login", { email, password }, { withCredentials: true });
17
18            // 2. 비즈니스 로직 및 상태 처리
19            if (res.data.success) {
20                await refreshAuth();
21                navigate("/main", { replace: true });
22            } else {
23                setError(res.data.message || "로그인 실패");
24                console.log(error);
25            }
26        } catch (err) {
27            console.error(err);
28            setError("로그인 서버 오류 발생");
29        } finally {
30            setIsLoading(false);
31        }
32    };
33
34    return { login, error, isLoading, setError };
35 }
36
37 export { useLogin };
38

```

( 로그인 Custom Hook )

<p>프로젝트 후기</p>	<p>본 프로젝트는 React 에 대한 실전 감각을 확보하고, 다소 무더진 Spring Boot 백엔드 감각을 되살리기 위해 약 한 달간 진행한 개인 프로젝트이다. 기존 Spring Boot + Thymeleaf 기반의 서버 사이드 렌더링 (SSR) 구조를 React 기반의 클라이언트 사이드 렌더링(CSR) 구조로 전환하는 것을 주요 목표로 삼았다. 학습이 목적이었기 때문에, 가능한 한 AI 의 도움을 최소화하고 직접 설계 · 구현 · 디버깅 하는 방식으로 프로젝트를 진행하였다.</p> <p>프로젝트 초기에는 “Thymeleaf 템플릿을 JSX 로 옮기고, HTML 대신 JSON 을 주고받도록 Controller 만 수정하면 비교적 빠르게 마무리할 수 있겠다” 라는 판단을 했다.</p> <p>그러나 실제로는 렌더링 방식이 바뀌는 것 자체가 단순한 View 변경이 아니라 시스템 전반의 책임 구조를 바꾸는 작업임을 프로젝트를 진행하며 깨닫게 되었다. CSR 구조로 전환하면서 기존에는 서버에서 암묵적으로 처리되던 화면 분기, 데이터 가공, 상태 판단 로직을 프론트엔드로 이전하게 되었고, 해당 과정에서 기존에는 필요 없던 데이터 요구사항들이 새롭게 발생했다. 그 결과 새로운 DTO 를 추가와 기존 Repository, Service, mapper.xml 로직의 확장이 이루어졌다. 이 과정에서 기존 프로젝트의 DTO, Controller, Service 간 책임이 어긋난 부분들을 발견하였고 이로 인해 예상보다 많은 디버깅 시간을 소모하였다. 이를 통하여 <b>초기 설계의 중요성</b>을 매우 강하게 체감하게 되었다.</p> <p>React 전환 이후, 프론트엔드 로직을 utils 와 hooks 로 분리하는 과정에서 props drilling 이 깊어지며 한글 IME 입력 조합이 깨지는 버그를 경험하였다. 이 문제를 통하여 zustand 나 redux 와 같은 전역 상태 관리 도구의 필요성을 느끼게 되었다.</p> <p>프로젝트 후반부에는 보안 강화를 위해 Spring Security 를 도입하였다. 그러나 초기 설정 단계에서 permitAll 처리를 충분히 고려하지 않아 다른 에러 경로에서 403 에러가 발생하는 문제를 겪었다. 이를 해결하며, 보안 역시 기능 구현 이후 덧붙이는 요소가 아니라 <b>초기 설계 단계부터 함께 고려</b>되어야 하는 영역임을 다시 한 번 인식하게 되었다.</p> <p>이 개인 프로젝트를 진행하며 가장 크게 체감한 것은 초기 설계가 프로젝트의 유지보수에 지대한 영향을 끼친다는 점이다. 설계 자체가 미흡하면 작은 변경을 하려고 해도 그것이 연쇄적인 수정으로 이어지고, 디버깅 비용이 기하급수적으로 증가한다. 그렇기에 다음 프로젝트에서는 구현 이전에 설계가 보다 많은 시간을 투자할 계획이다. 구체적으로 팀원들과의 코드 컨벤션을 통하여 네이밍과 디렉터리 규칙을 사전에 정의할 생각이다. 그리고 컴포넌트, 유스케이스 등 다양한 다이어그램을 그려 전체 흐름을 확실히 잡은 상태에서 개발을 진행할 생각이다. 이를 통해 단순히 개발 속도보다는 전체적인 구조의 안정성과 확장성을 우선하는 방향으로 목표로 나아갈 생각이다.</p>
----------------	---

끝까지 읽어 주셔서 감사합니다.