

---

# 백엔드 개발 지원

## PORTFOLIO

### 김광훈



Contact

010-8668-9294

[hosookkh@gmail.com](mailto:hosookkh@gmail.com)

<https://github.com/kwanghunk>

---

---

# CONTENT

**1** 자기 소개

**2** 기술 스택

**3** 프로젝트

**4** 핵심 역량

## 1. 자기소개 - 1분 자기 소개

### 1분 자기소개

신입 풀스택 개발자 김광훈입니다.

8년간의 군 복무를 통해 책임감과 팀워크를 익혔으며,

IT 기술이 만들어내는 변화와 가능성에 끌려 개발을 시작했습니다.

책임감과 팀워크 능력을 바탕으로 팀에 기여하는 개발자로 성장하고 싶습니다.

풀스택 과정과 프로젝트 경험을 통해 백엔드와 프론트엔드를 모두 경험했으며,

협업과 코드 품질을 중요하게 생각합니다.

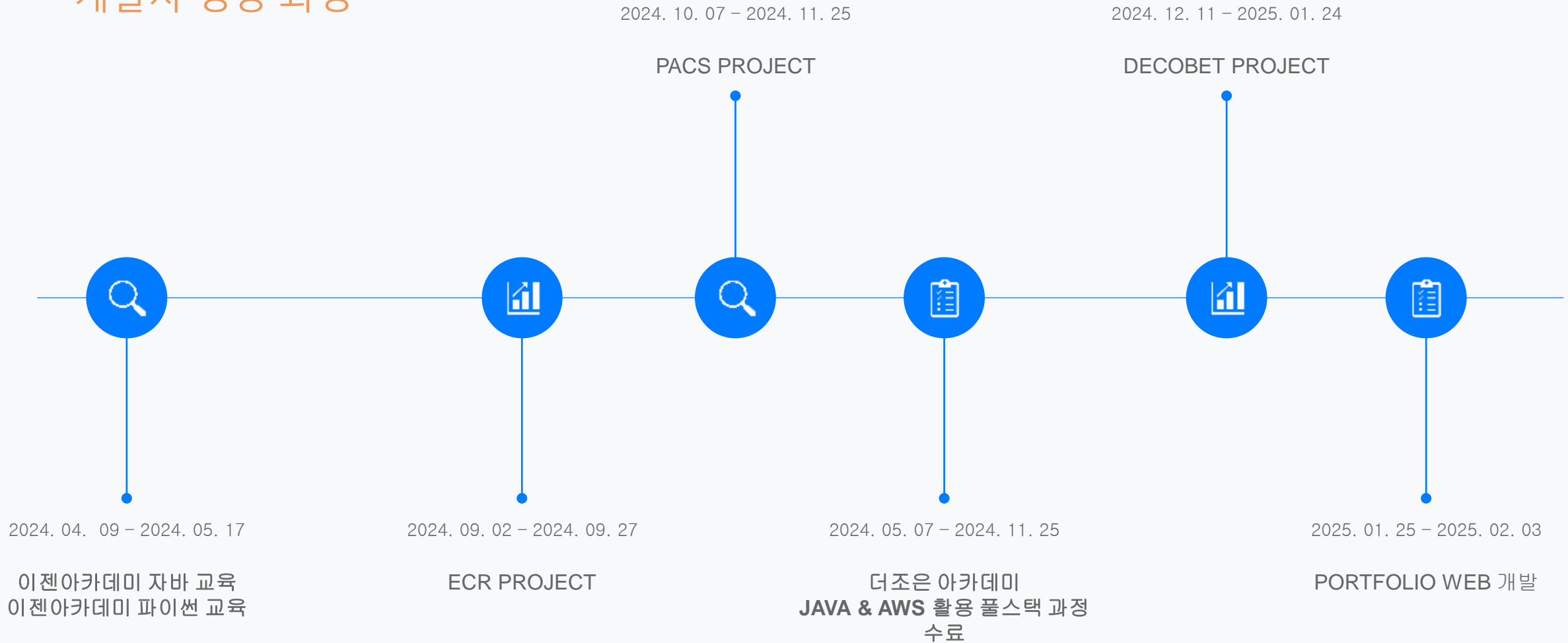
특히, 아키텍처 설계와 관심사 분리를 적용하여 유지보수성과 확장성을 고려한 개발을 지향합니다.

입사 후에도 실무에서 필요한 기술과 협업 역량을 끊임없이 발전시키며,

팀과 함께 성장하는 개발자가 되고 싶습니다.

## 1. 자기소개 – 개발자 성장 과정

### 개발자 성장 과정



## 2. 기술스택

### 기술스택

#### 주요 스택

##### [ Backend ]

- Language: Java 17
- Framework: Spring Framework, Spring Boot, Spring Security, JWT
- Hibernate: MyBatis, JPA(+ NativeQuery, JPQL)
- DBMS: Oracle DB, MySQL
- Template Engine: JSP, Thymeleaf
- Build Tool: Maven, Gradle
- Server: Apache, Tomcat
- DVCS: Git
- IDE: Eclipse, STS

##### [ Frontend ]

- HTML, CSS, JavaScript, jQuery, React
- Bootstrap

#### 보유 역량

##### [ Skills ]

- Spring Framework 기반의 RESTful API 설계 및 구현 경험
- Native Query와 JPQL을 활용한 복잡한 데이터 처리 및 최적화
- TDD를 기반으로 JUnit, Mockito를 활용한 테스트 코드 작성 능력
- CI/CD 파이프라인 구축 경험 (GitHub Actions)
- AWS EC2를 활용한 배포 및 운영 경험
- 팀 협업 프로젝트를 통한 효율적인 Git 관리 및 코드 리뷰 경험
- 아키텍처 설계와 관심사 분리를 적용한 유지보수성과 확장성을 고려한 설계 경험
- HTML, CSS, JavaScript, jQuery, Bootstrap을 활용한 프론트엔드 개발 경험

### 3. 프로젝트 – 프로젝트 목록

#### 프로젝트 목록

##### Escape Room(ECR)

2024. 09. 02 – 2024. 09. 27

인원: 백엔드&프론트엔드 5명

사용기술: Java, Spring Boot, JPA, React, Oracle, AWS

설명: 방탈출 카페 스마트 스토어

##### [ 담당역할 ]

- 동시성 제어를 활용한 안정적인 예약 시스템 구축
- 사용자 및 관리자 예약 관리 기능 구현
- 제휴업체 회원 관리 시스템 개발

##### HealScope

2024. 10. 07 – 2024. 11. 25

인원: 백엔드&프론트엔드 5명

사용기술: Java, Spring Security, JPA, Thymeleaf, HTML5, Oracle

설명: PACS

(의료영상정보전달시스템)

##### [ 담당역할 ]

- 의료 영상 조회, 조작, 전송 시스템 개발
- 대규모 데이터 처리 최적화
- 사용자 친화적 UI/UX 구현

##### DECOBET

2024. 12. 11 – 2025. 01. 24

인원: 백엔드&프론트엔드 3명

사용기술: Java, Spring Security, JWT, JPA, React, Redis, MySQL

설명: 주니어 개발자 코드 번역기

##### [ 담당역할 ]

- 아키텍처 설계 및 데이터베이스 설계
- Spring Security 설정 및 JWT 인증 구현
- 코드 번역 및 검색 API 개발
- 번역 히스토리 및 1:1 문의 관리 기능 개발

ESCAPE ROOM  
[ ECR ] Web Dev

## What is this Project?

다양한 방탈출 카페 데이터를 통합 관리하며,  
검색, 예약, 리뷰 기능을 제공하는 종합 플랫폼입니다.

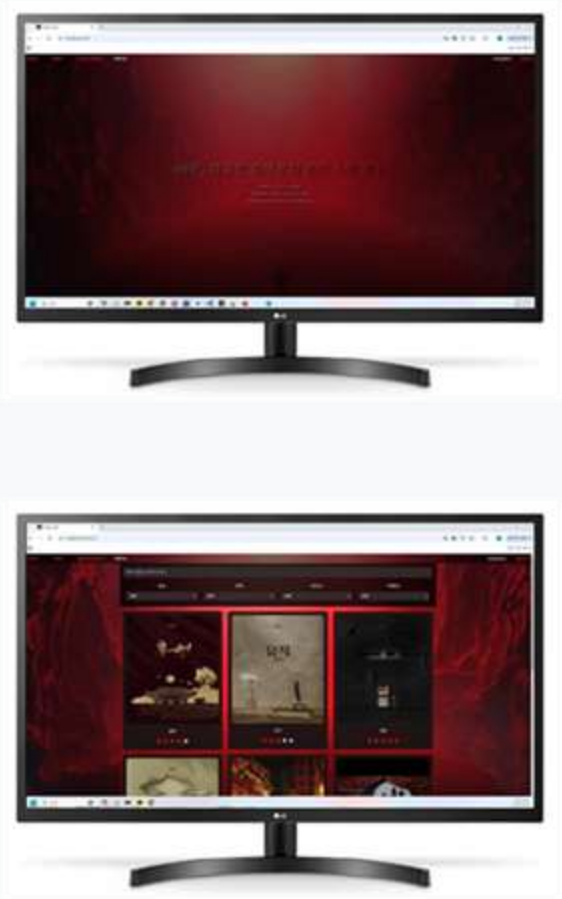
## What are the main features?

- 방탈출 카페 검색 및 필터링
- 예약 가능한 시간대 실시간 확인 및 예약
- 사용자 리뷰 관리 및 평가



### 3-1. 프로젝트 – 프로젝트 소개

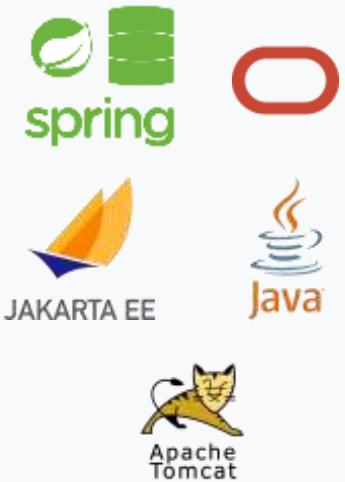
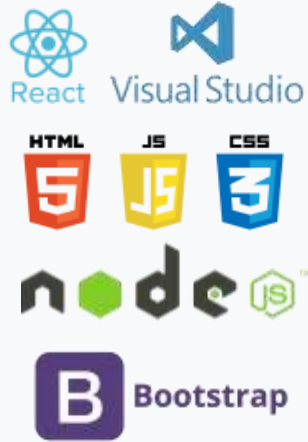
#### 프로젝트 소개 (ECR)



#### [ Escape Room ]

- 방탈출 스마트 스토어 웹 서비스 개발 ( 백엔드 & 프론트엔드 개발)
- 팀 프로젝트 총 5명
- 2024. 09.02 – 2024. 09. 27
- [GitHub - https://github.com/kwanghunk/ECR-WEB-Project](https://github.com/kwanghunk/ECR-WEB-Project)

#### [ Tech Stack ]

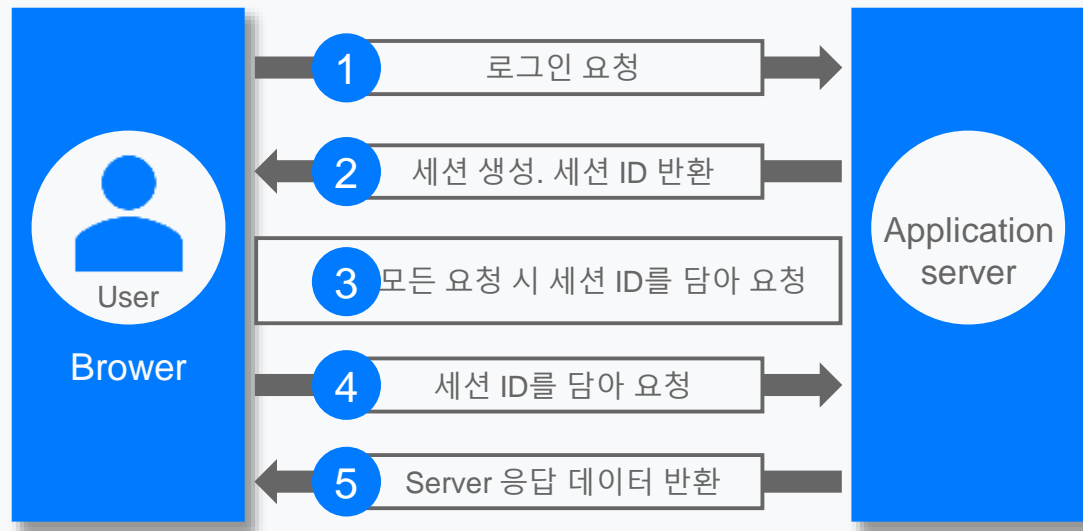


| 공통                               | 사용자                           | 관계자      | 관리자   |
|----------------------------------|-------------------------------|----------|---|
| 회원가입, 로그인<br>상품 목록 및<br>상품 상세 조회 | 상품 예약 관리<br>예약 내역 관리<br>리뷰 관리 | 상품 등록/삭제 | 사용자/관계자 회원관리<br>예약 상품 관리<br>상품, 리뷰, 공지사항 관리 |

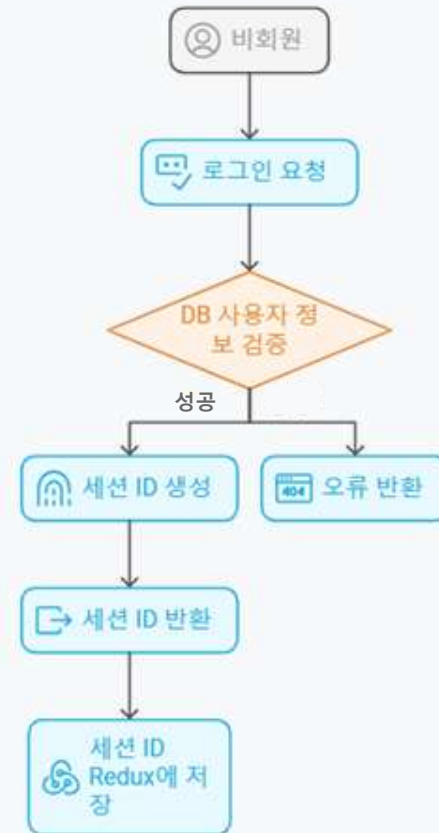
### 3-1. 프로젝트 - 시스템 흐름도

#### 시스템 흐름도

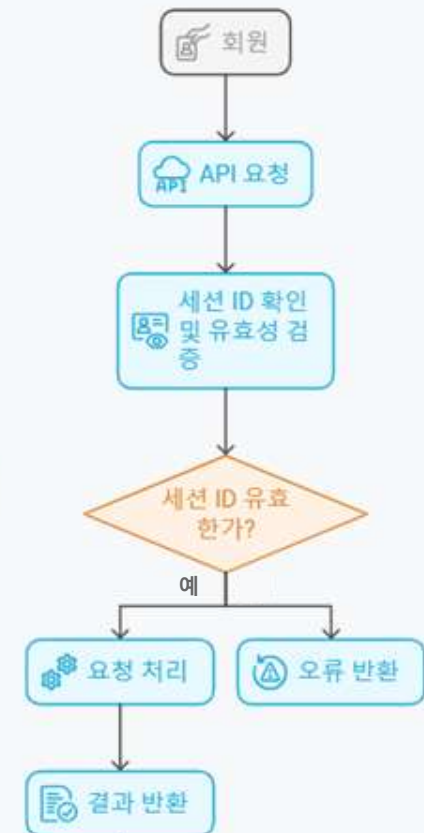
- 사용자 로그인 및 API 요청 처리 과정을 시각화 한 것입니다.
- 로그인 시 세션 ID를 생성하고, Redux에 저장하여 인증 상태를 관리합니다.
- API 요청은 세션 ID 유효성을 검증한 뒤 처리 결과를 반환합니다.
- 효율적인 세션 관리와 데이터 일관성을 위해 설계되었습니다.
- 확장성과 보안을 고려한 구조입니다.



로그인 흐름도



API 요청 처리 흐름도



### 3-1. 프로젝트 - 담당역할

## 담당역할 - 예약시스템

#### 1. 주요 기능

- “동시성 제어를 활용한 예약 시스템 구축”

#### 2. 문제 정의

- 다중 사용자가 동시에 예약할 경우의 데이터 충돌을 방지하고 데이터 무결성을 보장하기 위해 동시성 제어를 적용했습니다.

#### 3. 핵심 코드

- @Transactional을 사용하여 예약 도중 오류가 발생하면 롤백되도록 설정
- 예약 가능 여부를 먼저 확인하고, 같은 시간대에 중복 예약이 있으면 예약을 차단

#### 4. 결과

- 동시 요청 발생 상황에서도 데이터 무결성 유지와 안정적인 예약 시스템이 작동.



```
@Transactional
public boolean reserve(Reservation reservation) {
    Optional<Reservation> existingReservation =
        reservationRepository.findByTema_TemaNoAndUseDateAndUseTime(
            reservation.getTema().getTemaNo(),
            reservation.getUseDate(),
            reservation.getUseTime()
        );

    if (existingReservation.isPresent()) return false;

    reservation.setPaymentStatus("결제 대기");
    reservationRepository.save(reservation);
    return true;
}
```

### 3-1. 프로젝트 – 담당역할

#### 담당역할 – 예약관리시스

##### 1. 주요 기능

- 사용자 예약 내역 조회 및 취소 요청
- 관리자 예약 승인 및 취소 처리

##### 2. 문제 정의

- 다중 요청으로 인한 데이터 무결성 문제 발생
- 인증되지 않은 사용자의 예약 접근 제한 필요

##### 3. 핵심 코드

- 트랜잭션 적용으로 데이터 무결성 보장
- 인증 로직 추가로 보안 강화

##### 4. 결과

- 트랜잭션과 인증 로직으로 안정적인  
예약 관리 구현



```
@Transactional
public Reservation approveReservation(Long reservationCode) {
    Optional<Reservation> reservation =
        reservationRepository.findById(reservationCode);
    if(reservation.isPresent()) {
        Reservation res = reservation.get();
        res.setPaymentStatus("예약확정");
        return reservationRepository.save(res);
    }
    return null;
}
```

```
@GetMapping("/findUserReserveAll")
public ResponseEntity<?> findUserReserveAll(
    @RequestParam(name = "page") int page,
    @RequestParam(name = "size") int size,
    HttpServletRequest request) {
    try {
        Member sessionMember = AuthenticationUtils.authClient(request);
        String userId = sessionMember.getMemberId();

        List<Reservation> result =
            reservationService.findUserReservations(userId, page, size);
        return result != null && !result.isEmpty()
            ? ResponseEntity.ok(result)
            : ResponseEntity.status(204).body("예약 내역이 없습니다.");
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    } catch (Exception e) {
        return ResponseEntity.status(500).body("서버 내부 오류 발생: " + e.getMessage());
    }
}
```

## 3-1. 프로젝트 – 담당역할

### 담당역할 – 회원관리시스템

#### 1. 주요 기능

- 관리자: 일반회원 및 제휴업체 회원 관리
- 회원 조회(페이징 처리) 및 회원 삭제 기능 제공

#### 2. 문제 정의

- 대량의 데이터를 처리할 때 조회 성능 저하 문제
- 회원 삭제 시 무결성 원칙 위배 오류 발생 문제

#### 3. 핵심 코드

- 페이징 처리로 데이터 조회 성능 최적화
- 연관된 데이터 삭제를 위해 각 레포지토리를 순차적 호출

#### 4. 결과

- 페이징 처리로 대량 데이터 처리 성능 개선
- @Transactional로 데이터를 안전하게 삭제



```
@Transactional
public void deleteMemberAndReviewsAndTemas(String memberId) {
    if(memberRepository.existsById(memberId)) {
        reviewRepository.deleteByUserId(memberId);
        temaRepository.deleteByMemberId(memberId);
        memberRepository.deleteById(memberId);
    } else {
        throw new IllegalArgumentException("해당 회원을 찾을 수 없습니다");
    }
}
```

```
@GetMapping("/findClientAll")
public ResponseEntity<?> getAllClients(
    @RequestParam(name = "loginType") int loginType,
    @RequestParam(name = "page") int page,
    @RequestParam(name = "size") int size,
    HttpServletRequest request) {
    try {
        AuthenticationUtils.authClient(request);
        List<Member> list = memberService.getAllClients(loginType, page, size);
        return ResponseEntity.ok(list);
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    }
}
```

HealScope  
[ PACS ] Web Dev

## What is this Project?

의료 기관에서 DICOM 의료 데이터를 조회, 분석, 저장할 수 있도록 개발했습니다.  
의료진이 방대한 영상 데이터를 빠르게 검색하고 전달할 수 있도록  
고속 조회 최적화 및 멀티 뷰어 기능을 제공합니다.

## What are the main features?

- 대량의 의료 영상 데이터 저장 및 관리
- DICOM 이미지 검색 및 필터링
- 빠른 데이터 조회
- 멀티 뷰어 및 시리즈 비교 기능
- DICOM 이미지 다운로드



## 3-2. 프로젝트 – 프로젝트 소개

### 프로젝트 소개 (HealScope)



#### [ HealScope ]

PACS(의료영상정보전달시스템) ( 백엔드 & 프론트엔드 개발)

팀 프로젝트 총 5명

프로젝트 기간: 2024. 10.07 – 2024. 11. 25

[GitHub - https://github.com/kwanghunk/PACS-WEB-Project](https://github.com/kwanghunk/PACS-WEB-Project)

#### [ Tech Stack ]



#### [ 핵심 구현기능 ]

|                 |                              |
|-----------------|------------------------------|
| 빠른 이미지 검색 및 필터링 | 사용자 입력 조건에 따른 고속 검색 지원       |
| 빠른 데이터 조회 최적화   | Native Query + 페이징을 통한 성능 개선 |
| 멀티 뷰어 및 비교 모드   | 다중 영상 비교 및 시리즈 뷰 지원          |
| DICOM 다운로드 기능   | 단일/시리즈 ZIP 압축 다운로드 제공        |



## 3-2. 프로젝트 - 시스템 흐름도

### 시스템 흐름도

- HealScope는 의료 영상 데이터를 관리하고,
- DICOM 뷰어와 이미지 조작 기능을 제공합니다.
- 백엔드는 빠른 데이터 조회와 다운로드를 처리하며,
- DB는 영상 메타정보를 저장 및 관리합니다.
- 시스템은 사용자 요청을 효율적으로 처리합니다.

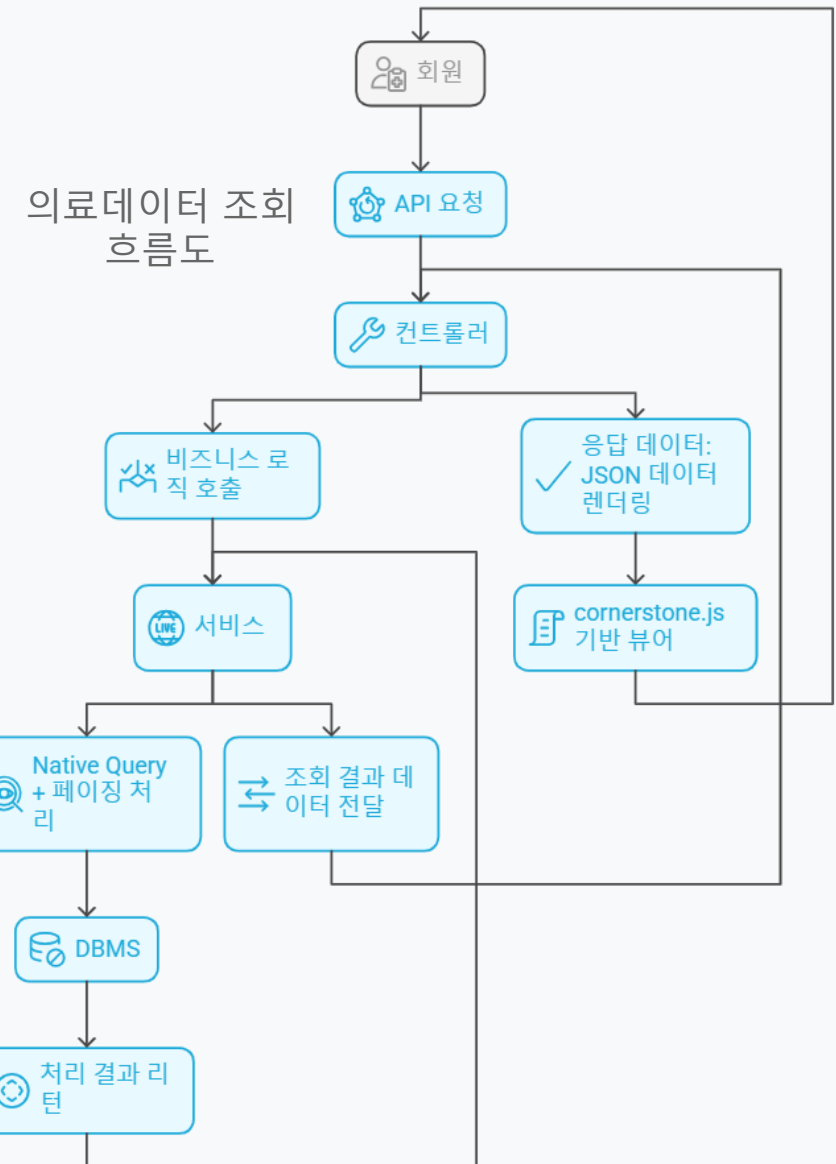
#### [ 프로젝트 미니맵 ]

| ★ External Libraries               |
|------------------------------------|
| Cornerstone.js:                    |
| • Cornerstone-core (DICOM 이미지 렌더링) |
| • cornerstone-tools (뷰어 도구)        |
| • Dicom-parser (DICOM 데이터 파싱)      |
| • Hammer.js (터치 및 제스처 지원)          |

| ★ Modes             |
|---------------------|
| Modes:              |
| • DICOM 이미지 조회 모드   |
| • DICOM 파일 저장 모드    |
| • 시리즈 비교 모드 (다중 뷰어) |

| ★ App Configuration                   |
|---------------------------------------|
| Data Source:                          |
| • DICOMWeb                            |
| • Local Storage 설정: ImgConfig.java    |
| Templates:                            |
| • ImageList.html (목록 화면)              |
| • ImageDetail.html (상세 화면)            |
| Core Features:                        |
| • cornerstone.js 기반 DICOM 뷰어          |
| • 다중 뷰어 및 시리즈 비교 모드 지원                |
| • 이미지 확대, 축소, 회전 등 19개 기능 지원          |
| Routes:                               |
| • /ImageTabList (이미지 목록 조회)           |
| • /ImageDetail (이미지 상세 조회)            |
| • /file/download-series (Zip 파일 다운로드) |

| ★ Backend Configuration                         |
|---|
| 주요 컨트롤러:  |
| • ImageTabController (RESTful API로 데이터 제공)      |
| • DownloadController (DICOM ZIP 압축 다운로드 처리)     |
| 주요 서비스:   |
| • ImageTabService (Native Query + 페이징 → 조회 최적화) |
| 데이터 요청 흐름:                                      |
| • 사용자 요청 → 컨트롤러 → 서비스 → DBMS → 응답 반환            |



## 3-2. 프로젝트 – 담당역할

### 담당역할 – DICOM 데이터 조회

#### 1. 주요 기능

- 대용량 DICOM 데이터를 빠르게 조회하고 필터링할 수 있는 기능 제공
- cornerstone.js 이미지 시각화

#### 2. 문제 정의

- 대량 데이터 조회 시 속도 저하 및 서버 부담
- 페이징 및 Native Query를 활용하여 성능 최적화 필요

#### 3. 핵심 코드

- Native Query 및 페이징 처리 구현
- JSON 데이터를 뷰어로 전달하는 API 작성

#### 4. 결과

- JPA ORM 사용 시 대용량 데이터 조회에 10초 이상 소요되던 문제 해결
- Native Query와 페이징 적용 후 2초 이내 처리
- 대량 데이터 처리 효율성 5배 이상 향상



```
//imageList 가져오기
@Query(value = "SELECT CONCAT(REPLACE(i.PATH, '\\', '/'), i.FNAME) " +
              "FROM IMAGETAB i " +
              "WHERE i.STUDYKEY = :studyKey AND i.SERIESKEY = :seriesKey " +
              "ORDER BY i.CURSEQNUM",nativeQuery = true)
ArrayList<String> findByIdStudyKeyAndIdSeriesKeyOrderByIdImageKey(
    @Param("studyKey") Long studyKey,
    @Param("seriesKey") Long seriesKey);
```

```
// 데이터 페이징 처리 및 Native Query 구현
Page<Long> seriesList = imageTabService.seriesList(PageRequest.of(nowPage, 4),studyKey);
for(Long s : seriesList) {
    System.out.println("seriesList : "+s);
    ArrayList<String> images = imageTabService.list(studyKey,s);
    imagesList.add(images);
}
// 응답 데이터 생성 및 반환
int totalPages =seriesList.getTotalPages()-1;
model.addAttribute("imageList",imagesList);
model.addAttribute("nowPage",nowPage);
model.addAttribute("totalPages",totalPages);
```

## 3-2. 프로젝트 – 담당역할

### 담당역할 – 시리즈 비교 모드

#### 1. 주요 기능

- 선택한 시리즈 데이터를 받아 다중 비교 지원
- Native Query + 페이징으로 빠른 조회 지원

#### 2. 문제 정의

- 대량 데이터 동시 조회 시 성능 저하 발생
- DB 부하 감소를 위한 쿼리 최적화 필요

#### 3. 핵심 코드

- Native Query로 시리즈 데이터 조회 최적화
- 페이징 적용으로 과도한 데이터 로딩 방지

#### 4. 결과

- 기존 JPA 대비 조회 속도 3배 향상
- API 응답 평균 1.5초 이내 유지



```
@Query(value = "SELECT DISTINCT s.SERIESKEY " +  
              "FROM IMAGETAB s " +  
              "WHERE s.STUDYKEY = :studyKey " +  
              "ORDER BY s.SERIESKEY", nativeQuery = true)  
ArrayList<Long> findBySeriesKey(@Param("studyKey") Long studyKey);  
  
@Query(value = "SELECT CONCAT(REPLACE(i.PATH, '\\', '/'), i.FNAME) " +  
              "FROM IMAGETAB i " +  
              "WHERE i.STUDYKEY = :studyKey AND i.SERIESKEY = :seriesKey " +  
              "ORDER BY i.CURSEQNUM", nativeQuery = true)  
ArrayList<String> findByIdStudyKeyAndIdSeriesKeyOrderByImageKey(  
    @Param("studyKey") Long studyKey, @Param("seriesKey") Long seriesKey);
```

## 3-2. 프로젝트 – 담당역할

### 담당역할 – DICOM 다운로드 기능

#### 1. 주요 기능

- 사용자가 선택한 DICOM 이미지를 ZIP 압축하여 다운로드
- 백엔드에서 파일 경로를 기반으로 압축 파일을 생성 및 반환

#### 2. 문제 정의

- 대량 파일 다운로드 시 네트워크 부하 및 응답 속도 저하 발생
- 다운로드 시 파일 경로 검증 및 예외 처리 필요

#### 3. 핵심 코드

- DownloadController에서 ZIP 파일 생성 및 전송
- download.js에서 axios로 zip 요청 후 파일 저장

#### 4. 결과

- 대량 파일 다운로드 속도 30% 향상

```
if (folderPath == null || fileNames == null || fileNames.isEmpty()) {
    throw new IllegalArgumentException("folderPath 및 fileNames가 필요합니다.");
}
// 절대 경로 생성
String baseDir = "C:" + folderPath;
List<String> absolutePaths = new ArrayList<>();
for (String fileName : fileNames) {
    String absolutePath = baseDir + "/" + fileName;
    absolutePaths.add(absolutePath); // 절대 경로 생성
    System.out.println("Resolved absolute path: " + absolutePath);
}

// ZIP 파일 생성
String tempDir = Files.createTempDirectory("dicom-series").toString();
String zipFilePath = tempDir + File.separator + "series_images.zip";

// 파일 경로 리스트를 ZIP 파일로 압축
createZipFile(absolutePaths, zipFilePath);

// 클라이언트로 ZIP 파일 전송
File zipFile = new File(zipFilePath);
InputStreamResource resource = new InputStreamResource(new FileInputStream(zipFile));

HttpHeaders headers = new HttpHeaders();
headers.add(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=series_images.zip");

return ResponseEntity.ok()
    .headers(headers)
    .contentTypeLength(zipFile.length())
    .contentType(MediaType.APPLICATION_OCTET_STREAM)
    .body(resource);
```

DECOBET  
[ Easy To Translate Coding ] Web Dev

## What is this Project?

- 주니어 개발자를 위한 코드 번역 사이트
- 소스코드의 기본 자료형, 메서드, 상수 등을 이해하기 쉽게 변환
- 사용자가 입력 또는 업로드한 코드의 형식을 유지한 채 한글 설명 추가

## What are the main features?

- 소스코드의 기본 자료형, 메서드, 상수 등을 한글로 변환
- 소스코드 파일 업로드를 통한 사용자 편의성 제공
- 이전 번역 결과를 조회 기능 제공

### 3-3. 프로젝트 – 프로젝트 소개

#### 프로젝트 소개 (DECOBET)



##### [ DECOBET ]

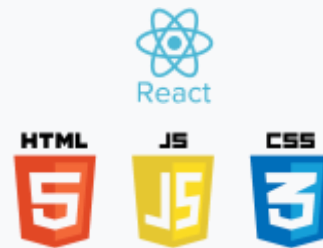
주니어 개발자를 위한 코드 번역 웹 사이트 ( 백엔드 & 프론트엔드 개발)

팀 프로젝트 총 3명

프로젝트 기간: 2024. 12.23 – 2025. 01. 24 (조기 종료)

[GitHub - https://github.com/kwanghunk/Third-WEB-Project](https://github.com/kwanghunk/Third-WEB-Project)

##### [ Tech Stack ]



##### [ 핵심 구현기능 ]

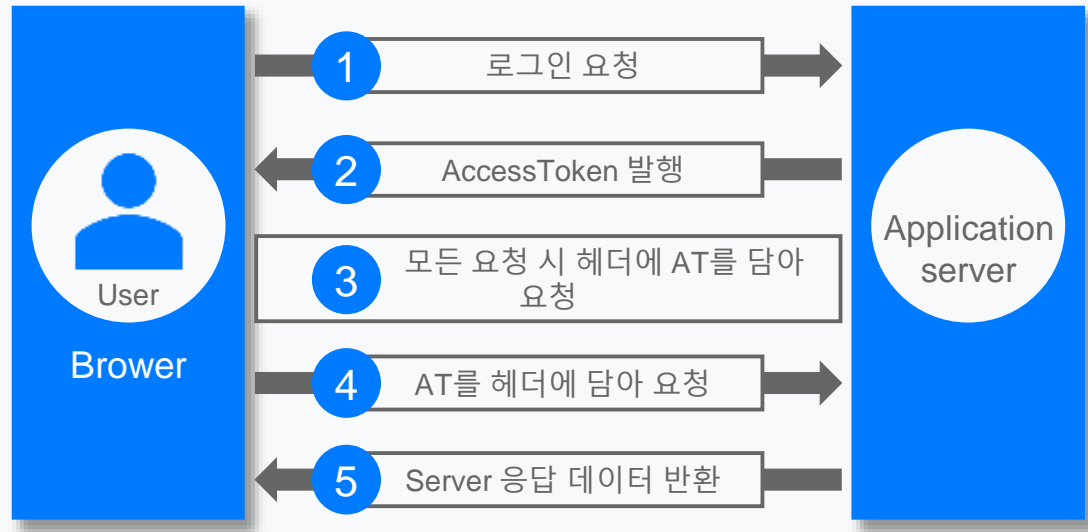
|                |                                 |
|----------------|---------------------------------|
| 코드 번역 api      | 자료형, 메서드 및 변환 기능 개발             |
| 번역 히스토리 api    | 이전 번역 기록 조회 및 저장 기능 개발          |
| 번역 결과 다운로드 api | 원본 소스코드와 번역된 소스코드를 txt 파일로 다운로드 |
| 1대1 문의 api     | 문의 관련 CRUD 개발                   |



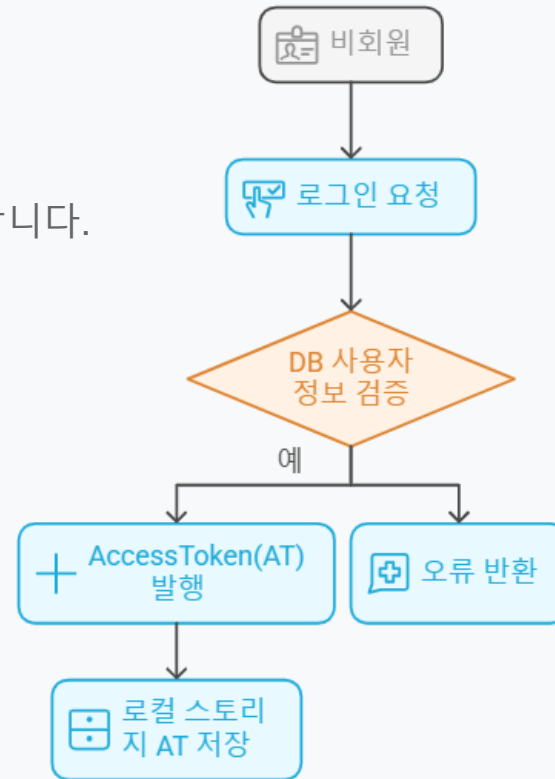
### 3-3. 프로젝트 - 시스템 흐름도

#### 시스템 흐름도

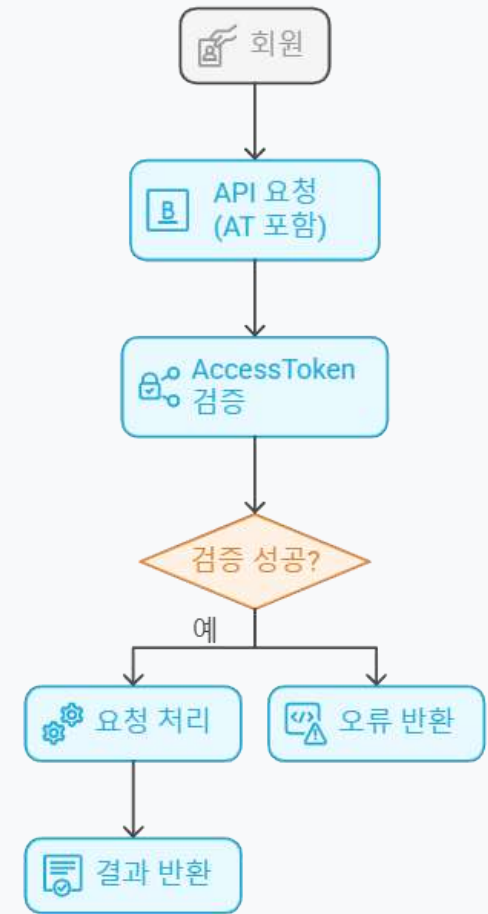
- 사용자 로그인 및 API 요청 처리 과정을 시각화 한 것입니다.
- 로그인 시 AccessToken을 발행하고, 로컬 스토리지에 저장하여 API 요청 시 헤더에 포함하여 요청합니다.
- API 요청은 AccessToken 유효성을 검증한 뒤 처리 결과를 반환합니다.
- AccessToken 검증 실패 시 오류를 반환합니다.



로그인 흐름도



API 요청 처리 흐름도





# DECOBET v2

## [ Easy To Translate Coding ] Web Dev

## What is this Project?

- DECOBET 프로젝트에서 보완이 필요한 점을 개선한 리빌딩 프로젝트입니다.
- JWT 인증 시스템을 고도화로 보안을 강화, 안정적인 서비스 운영을 목표로 합니다.
- HttpOnly, Secure, Samesite 등의 보안 설정을 적용하여 각종 보안을 강화합니다.

## What are the main features?

- JWT RTR (Refresh Token Rotation) 전략을 활용한 인증 시스템 고도화
- IP 기반 배회원 사용횟수 제한 기능으로 트래픽 관리
- 1:1문의 시스템의 계층형 댓글 구조 적용으로 서비스 관리 품질 향상
- Junit & Mockito를 활용한 TDD(테스트 주도 개발) 적용으로 코드 안정성 확보
- React-Bootstrap을 활용한 UI 개선 및 사용자 경험 향상

### 3-4. 프로젝트 – 프로젝트 소개

#### 프로젝트 소개 (DECOBET v2)



##### [ DECOBET v2 ]

보안 및 성능을 대폭 개선한 코드 번역 웹사이트 ( 백엔드 & 프론트엔드 개발)

개인 프로젝트

프로젝트 기간: 2025. 02.07 – 2025. 02. 28(진행 중)

[GitHub - https://github.com/kwanghunk/TEC-WEB-Project](https://github.com/kwanghunk/TEC-WEB-Project)

##### [ Tech Stack ]



Bootstrap



##### [ 핵심 구현기능 ]

보안 강화(JWT RTR 적용)

Refresh Token Rotation을 적용하여 보안성 향상

비회원 사용 제한 기능

IP 기반 요청 제한을 추가하여 트래픽 관리

1:1 문의 시스템 개선

계층형 댓글 구조 도입으로 서비스 관리 품질 향상

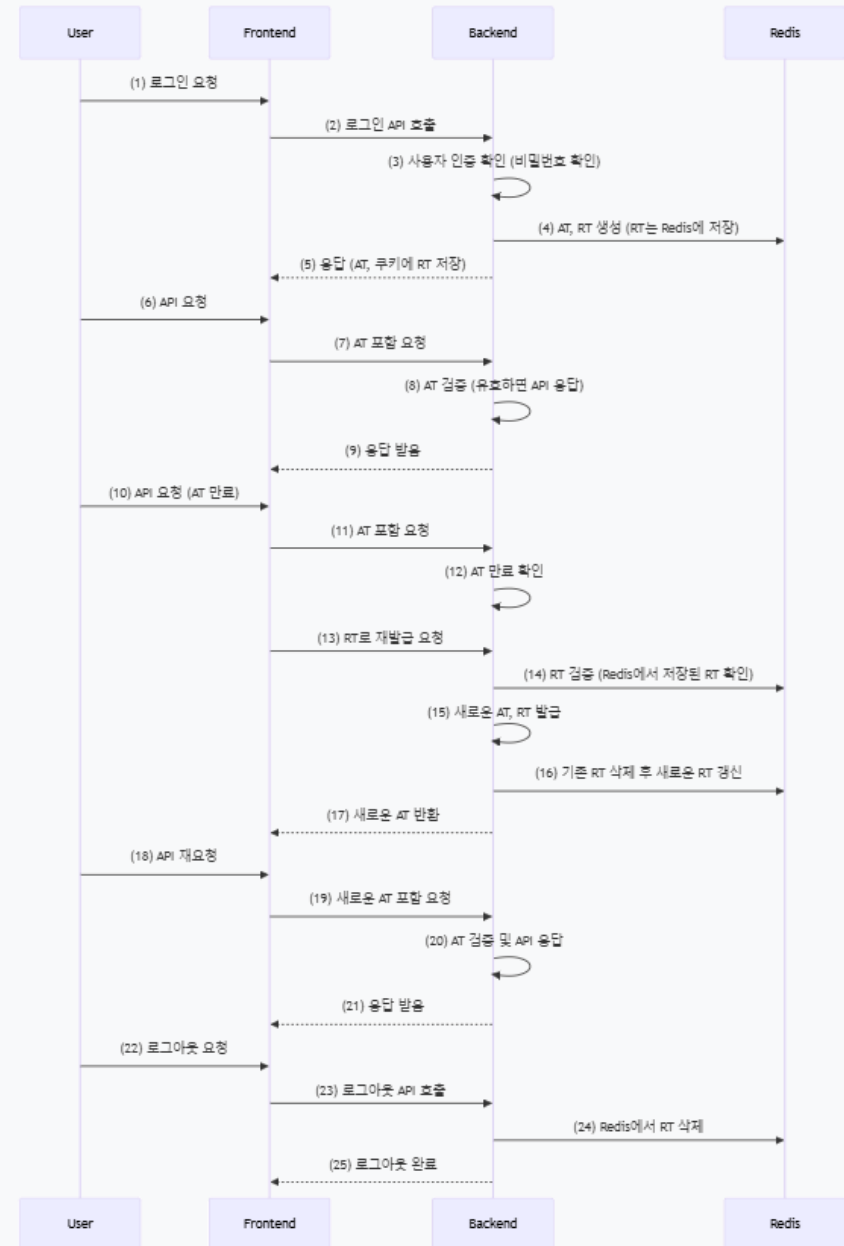
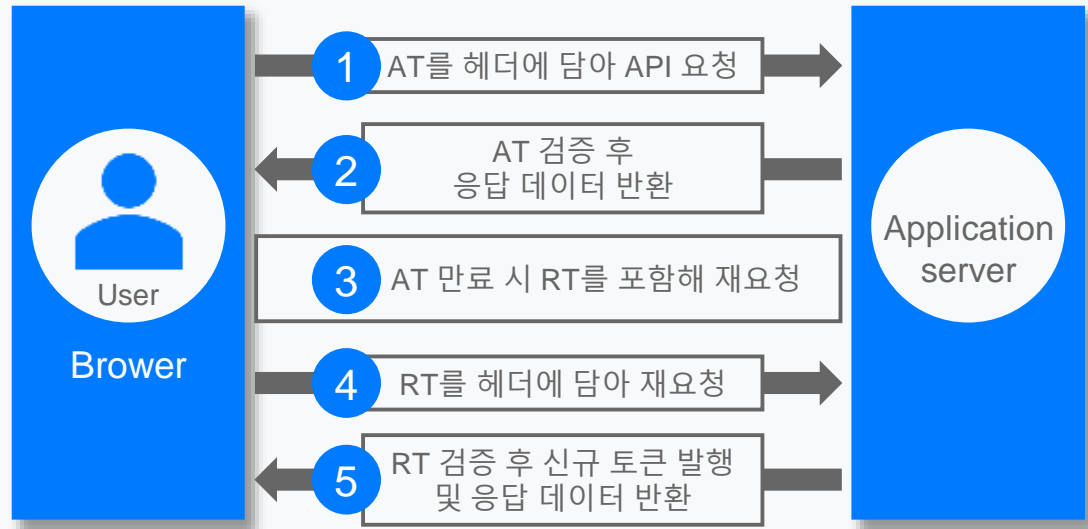
TDD(테스트 주도 개발) 적용

Junit & Mockito를 활용하여 안정적인 API 제공

### 3-4. 프로젝트 - 시스템 흐름도

## 시스템 흐름도

- 사용자 로그인 및 API 요청 처리 과정을 시각화 한 것입니다.
- 로그인 시 AT와 RT를 함께 발급하며, RT는 Redis에 저장 보안을 강화합니다.
- API 요청 시 AT를 헤더에 포함하여 요청하며, AT가 만료되었을 경우 저장된 RT를 이용하여 신규 AT, RT를 자동 발급합니다.
- RTR적용으로 RT가 재사용될 경우 기존 RT는 폐기하고 사용자의 모든 세션을 강제 종료하여 탈취 방지를 강화합니다.



## 3-4. 프로젝트 – 담당역할

### 담당역할 – JWTFilter 동작

#### 1. 주요 기능

- API 요청 시 JWTFilter를 통해 인증 및 권한 검사를 수행
- AccessToken(AT) 검증 및 갱신(RTR 전략 적용)

#### 2. 문제 정의

- 기존 JWT 인증 방식에서는 AT 만료 시 로그인 필요
- 토큰이 만료될 경우 자동 갱신 기능 부재

#### 3. 핵심 코드

- JWTFilter를 통해 AT 검증 및 만료된 경우 RT를 활용하여 자동 갱신 처리
- Redis를 활용하여 RT 저장 및 관리

#### 4. 결과

- RT를 활용한 자동갱신으로 사용자 경험 개선
- 인증 강화 및 API 요청 시 불필요한 로그인 감소

```
String authorization= request.getHeader("Authorization");
if (authorization == null || !authorization.startsWith("Bearer ")) {
    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
    response.getWriter().write("Unauthorized - No Access Token");
    return;
}

String token = authorization.replace("Bearer ", "");

if (jwtUtil.parseTokenExpirationCheck(token)) {
    String refreshToken = jwtUtil.getRefreshTokenFromCookie(request);
    if (refreshToken == null || !jwtUtil.isRefreshTokenValid(refreshToken)) {
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        response.getWriter().write("Unauthorized - Invalid Refresh Token");
        return;
    }
}
```

```
Claims refreshClaims = jwtUtil.parseToken(refreshToken);
String username = refreshClaims.get("username", String.class);
String userType = refreshClaims.get("userType", String.class);
String tokenFamily = refreshClaims.get("tokenFamily", String.class);

jwtUtil.revokeRefreshToken(refreshToken);

String newAccessToken = jwtUtil.createAccessToken(username, userType, 1000L * 60 * 15);
String newRefreshToken = jwtUtil.createRefreshToken(username, userType, tokenFamily);

Cookie newRefreshCookie = new Cookie("refreshToken", newRefreshToken);
newRefreshCookie.setHttpOnly(true);
newRefreshCookie.setSecure(true);
newRefreshCookie.setAttribute("SameSite", "Strict");
newRefreshCookie.setPath("/");
response.addCookie(newRefreshCookie);

response.setHeader("Authorization", "Bearer " + newAccessToken);

token = newAccessToken.replace("Bearer ", "");
```

### 3-4. 프로젝트 – 담당역할

#### 담당역할 – 번역 기록 관리

##### 1. 주요 기능

- 사용자의 번역 기록을 관리하며 이전 번역 기록의 조회, 저장, 다운로드 기능 제공
- 사용자가 요청한 기록만 데이터베이스(DB)에 저장하여 불필요한 데이터 증가 방지

##### 2. 문제 정의

- 사용자의 모든 번역 기록을 DB에 저장할 경우 저장공간 낭비 및 조회 성능 저하 발생 우려

##### 3. 핵심 코드

- 기본적으로 클라이언트 측에서 세션을 활용하여 최근 10개 기록만 유지
- 사용자가 명시적으로 요청한 기록만 DB에 저장

##### 4. 결과

- 세션 기록과 DB 기록을 분리하여 저장공간 최적화



```
SaveHistory entity = SaveHistory.builder()
    .username(username)
    .requestCode(saveDTO.getRequestCode())
    .responseCode(saveDTO.getResponseCode())
    .typeCode(saveDTO.getTypeCode())
    .historyTitle(saveDTO.getHistoryTitle())
    .saveTime(LocalDateTime.now())
    .build();
saveHistoryRepository.save(entity);
```

```
public String generateFileContentFromDB(int saveHistoryNo, String username) {
    Optional<SaveHistory> optionalHistory =
        saveHistoryRepository.findBySaveHistoryNoAndUsername(saveHistoryNo, username);
    if (optionalHistory.isPresent()) {
        SaveHistory history = optionalHistory.get();

        StringBuilder fileContent = new StringBuilder();
        fileContent.append(LocalDateTime.now()).append("\n")
            .append("Saved On: ").append(history.getSaveTime()).append("\n\n")
            .append("Language: ").append(history.getTypeCode()).append("\n\n")
            .append("OriginCode:\n").append(history.getRequestCode()).append("\n\n")
            .append("TranslationCode:\n").append(history.getResponseCode()).append("\n\n");
        return fileContent.toString();
    } else {
        throw new IllegalArgumentException("데이터를 찾을 수 없습니다.");
    }
}
```



### 3-4. 프로젝트 – 담당역할

## 담당역할 – 비회원 기능 제한(IP)

#### 1. 주요 기능

- 비회원 사용자의 API 요청 횟수 제한
- IP 기반 요청 카운트 관리 및 일정 주기로 초기화

#### 2. 문제 정의

- 무분별한 API 요청으로 서버 부하 발생 가능
- 비회원과 회원 간의 서비스 사용 차별성 미비

#### 3. 핵심 코드

- IpService를 활용하여 IP 기반 요청 횟수 관리
- IpResetService를 활용하여 매일 자정마다 요청 횟수 초기화

#### 4. 결과

- 비회원의 과도한 API요청을 방지하여 서비스 안정성 향상

```
// 비회원 요청 가능여부 확인 및 요청 카운트 증가
@Transactional
public boolean isRequestAllowed(HttpServletRequest request) {
    String ipAddress = ipUtil.getClientIp(request);

    Optional<Ip> ipRecord = ipRepository.findByIpAddress(ipAddress);
    Ip ipData = ipRecord.orElseGet(() -> createNewIp(ipAddress));

    if (ipData.getRequestCount() >= GUEST_LIMIT) return false; // 요청 제한 초과

    // 요청이 가능하면 카운트 증가 후 DB 저장
    ipData.setRequestCount(ipData.getRequestCount() + 1);
    ipData.setLastRequest(LocalDateTime.now());
    ipRepository.save(ipData);

    return true;
}
```

```
public String getClientIp(HttpServletRequest request) {
    String[] headers = {
        "X-Forwarded-For", // 프록시 서버 뒤의 실제 클라이언트 IP
        "Proxy-Client-IP", // Proxy-Client-IP 헤더 확인
        "WL-Proxy-Client-IP", // WebLogic Proxy-Client-IP
        "HTTP_X_FORWARDED_FOR", // 일부 프록시 환경
        "HTTP_CLIENT_IP" // 일부 프록시 호나영
    };
    String ipAddress = null;
    for (String header : headers) {
        String ipList = request.getHeader(header);
        if (ipList != null && !ipList.isEmpty() && !"unknown".equalsIgnoreCase(ipList)) {
            ipAddress = ipList.split(",")[0].trim();
            break;
        }
    }
    if (ipAddress == null || ipAddress.isEmpty() || "unknown".equalsIgnoreCase(ipAddress)) {
        ipAddress = request.getRemoteAddr();
    }
    return ipAddress;
}
```

### 3-4. 프로젝트 – 담당역할

#### 담당역할 – 1:1 문의

##### 1. 주요 기능

- 사용자와 관리자가 1:1 문의로 직접 소통 가능
- 계층형 댓글 구조로 서비스 품질 개선 향상

##### 2. 문제 정의

- 단순 1:1 문의 방식에서는 추가 답변이 어려움
- 만족할 만한 문의 답변을 받기 위해 반복적으로 새로운 문의를 작성해야 함

##### 3. 핵심 코드

- SupportService에서 문의 데이터를 계층 구조로 저장하여 다중 답변 지원
- 부모-자식 관계를 이용하여 답변이 이어질 수 있도록 설계

##### 4. 결과

- 다중 답변을 지원하여 효율적인 소통 가능



```
UserSupport parentInquiry = parentInquiryOpt.get();
UserSupport reply = UserSupport.builder()
    .username(username)
    .title("RE: " + parentInquiry.getTitle())
    .content(replyContent)
    .status(InquiryStatus.IN_PROGRESS)
    .category(parentInquiry.getCategory())
    .isDeleted("N")
    .createdDate(LocalDateTime.now())
    .parentInquiry(parentInquiry) // 부모 문의 연결
    .build();
```

```
supportRepository.save(reply);
```

```
// 재귀적으로 모든 답변 조회
private List<UserSupportDetailDTO> getRepliesRecursively(UserSupport parent) {
    List<UserSupport> replies = supportRepository.findAllRepliesByParentInquiryNo(parent.getInquiryNo());

    return replies.stream()
        .filter(reply -> "N".equals(reply.getIsDeleted())) // 삭제되지 않은 데이터만 필터링
        .map(reply -> {
            UserSupportDetailDTO dto = UserSupportDetailDTO.fromEntity(reply);
            dto.setReplies(getRepliesRecursively(reply)); // 재귀 호출하여 하위 답변까지 가져오기
            return dto;
        })
        .collect(Collectors.toList());
}
```



Thank you for visting my portfolio

김 광 훈