

# AWS 이용 카카오맵 기반 정보 제공 여행 일정 만들기 웹 프로그램

웹 개발 포토폴리오

AWS(클라우드) 기반 공공 · 빅데이터 활용 웹서비스 개발자 양성 과정

4조

팀장: 최해혁

팀원: 손민재,

임나영,

이현수





# 목차

1. 프로그램 개요
2. 팀원 소개 및 팀원 기술 스택
3. 개발환경
4. 메뉴 구조도
5. API 명세서
6. ERD 명세서
7. 화면 구현
8. 주요 기능 및 코드 리뷰
9. 프로젝트 수행 일정
10. 고찰
11. 마지막 Q&A



# 프로그램 개요

- ◆ **과제명** : AWS 이용 카카오맵 기반 정보 제공 여행 일정 만들기 웹 프로그램
- ◆ **과제 목적**: AWS(클라우드) 기반 공공·빅데이터 활용 웹서비스 개발자 양성 과정에 대한 이해와 적용
- ◆ **과제 범위**
  - **공간적 범위**: 대한민국
  - **내용적 범위**:
    - Front – 메인, 로그인, 여행 일정 관리, 리뷰 관리, MyPage
    - Admin – 지역 정보 관리
    - Back – Front 입력 정보를 AWS DB로 연결
- ◆ **과제 내용**
  - 요구 사항 정의
  - 요구 기능 정의
  - ERD 명세서
  - API 명세서
  - 서버, DB 구축
  - Front, Back 데이터 연결



# 팀원 소개 및 팀원 기술 스택



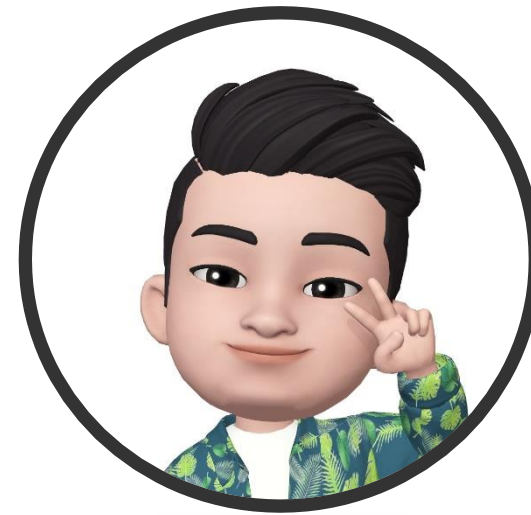
이름: 최해혁

프론트: 권한 router 설정,  
리뷰 생성 서버 통신,  
리뷰 디테일 서버 통신,  
리뷰 수정 서버 통신,  
회원 수정 서버 통신,

백엔드:  
회원수정, 일정 CRUD,  
리뷰 CRUD, OAuth2 로그인, 회원가입,  
DB설계

문서작업:  
클래스 다이어그램

사용한 기술



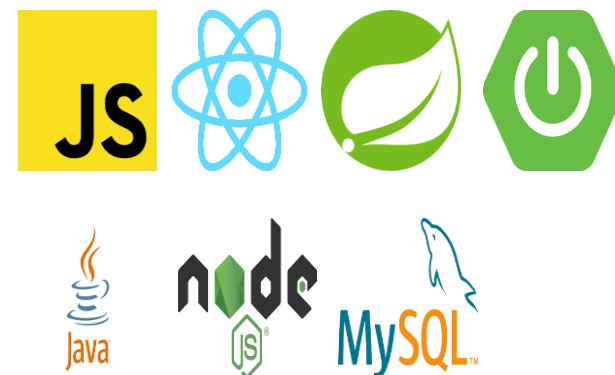
이름: 이현수

프론트: 여행 일정 생성 페이지  
및 통신,  
다른유저 여행가져오기 페이지  
및 통신

백엔드:  
여행 일정, 지역정보 Read,  
다른 유저의  
여행가져오기  
create

문서작업:  
팀 ppt 작성

사용한 기술



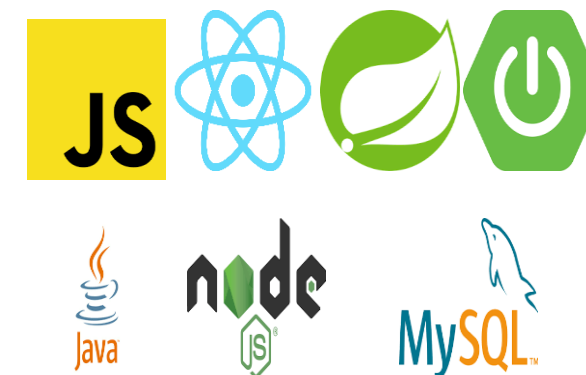
이름: 임나영

프론트:  
로그인, 회원가입, 비밀번호 변경,  
메인등 페이지 작업 및  
상세 디자인 작업

백엔드:  
회원정보 CRUD,  
비밀번호 update,  
지역정보 Create,  
메인페이지 Read

문서작업:  
ERD 작성

사용한 기술



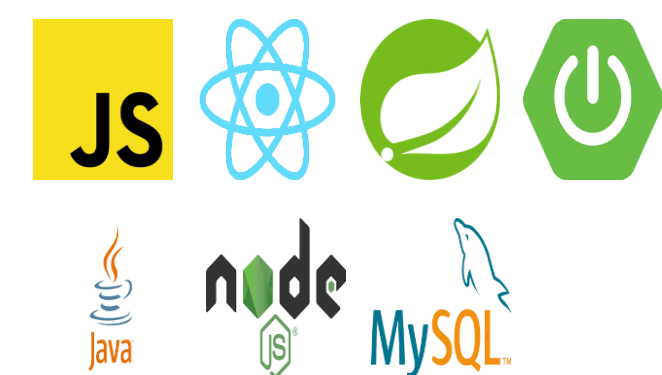
이름: 손민재

프론트:  
여행 생성 중 kakao Map,  
리뷰페이지 생성,

백엔드:  
일반 로그인 관련  
Jwt 토큰 발급

문서작업:  
API 명세서 작성

사용한 기술



# 개발환경

## Front

### 라이브러리

- node js 18.15.0
- react
- emotion
- react-router-dom
- Material-UI
- React-Meterial-UI-Carousel

### 개발 도구

- Visual Studio Code

## App

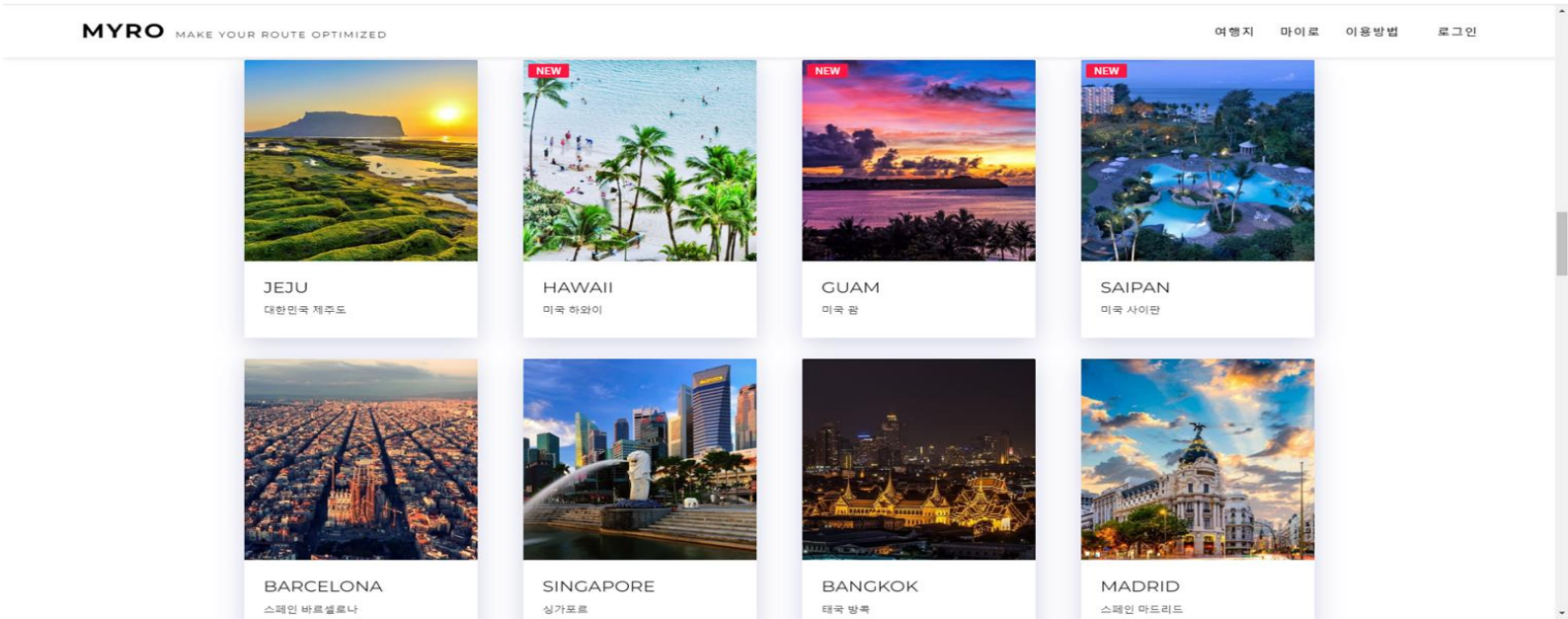
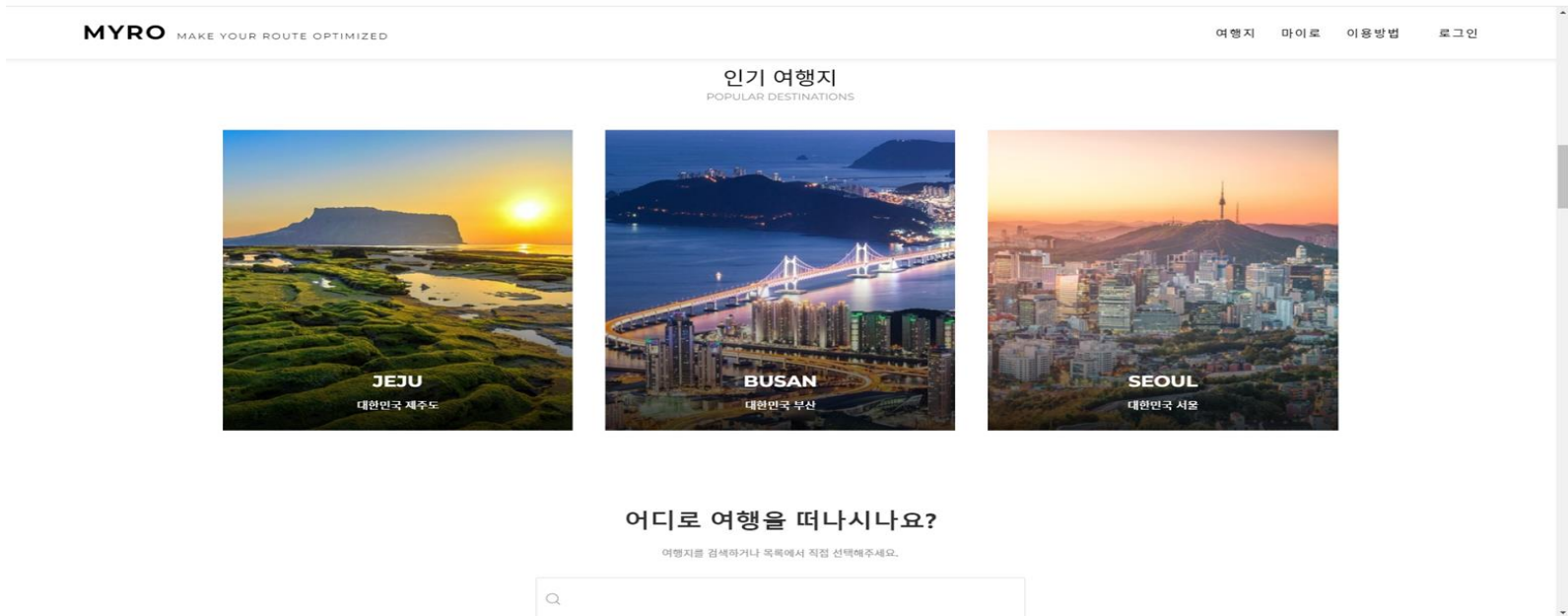
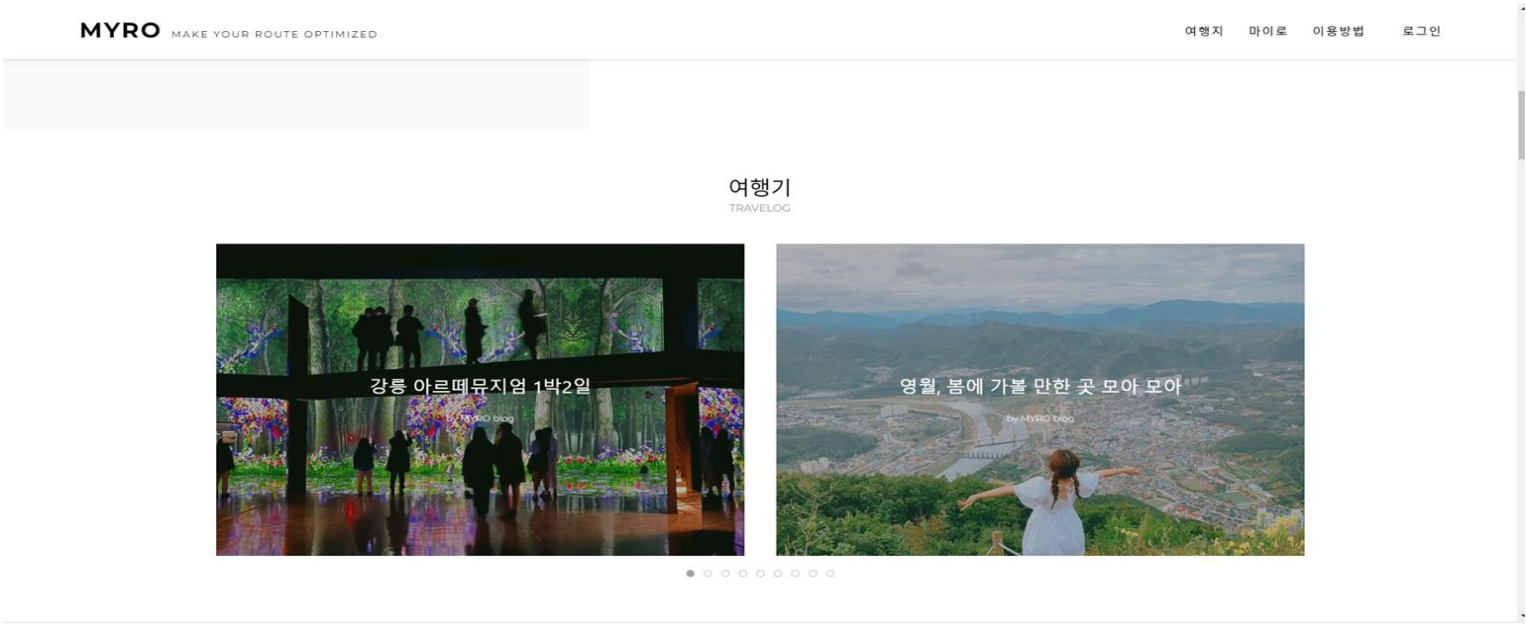
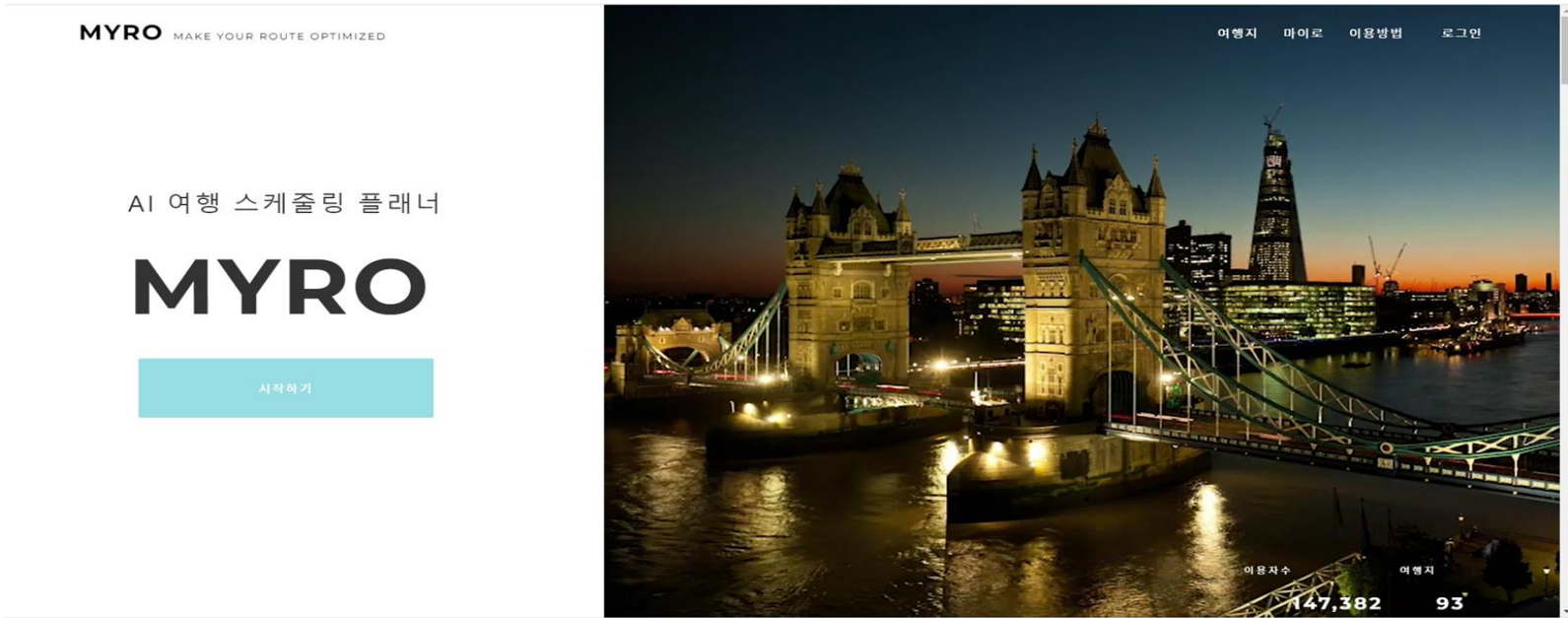
### 라이브러리

- JDK 11
- Spring Boot 2.6.6
- Spring Security
- Spring Aop
- MySQL 8
- Mybatis 2.2.2
- Lombok
- Maven

### 개발 도구

- IntelliJ IDEA
- Spring tool suite 4
- MySQL Workbench
- Postman
- Git
- SourceTree
- GitKraken

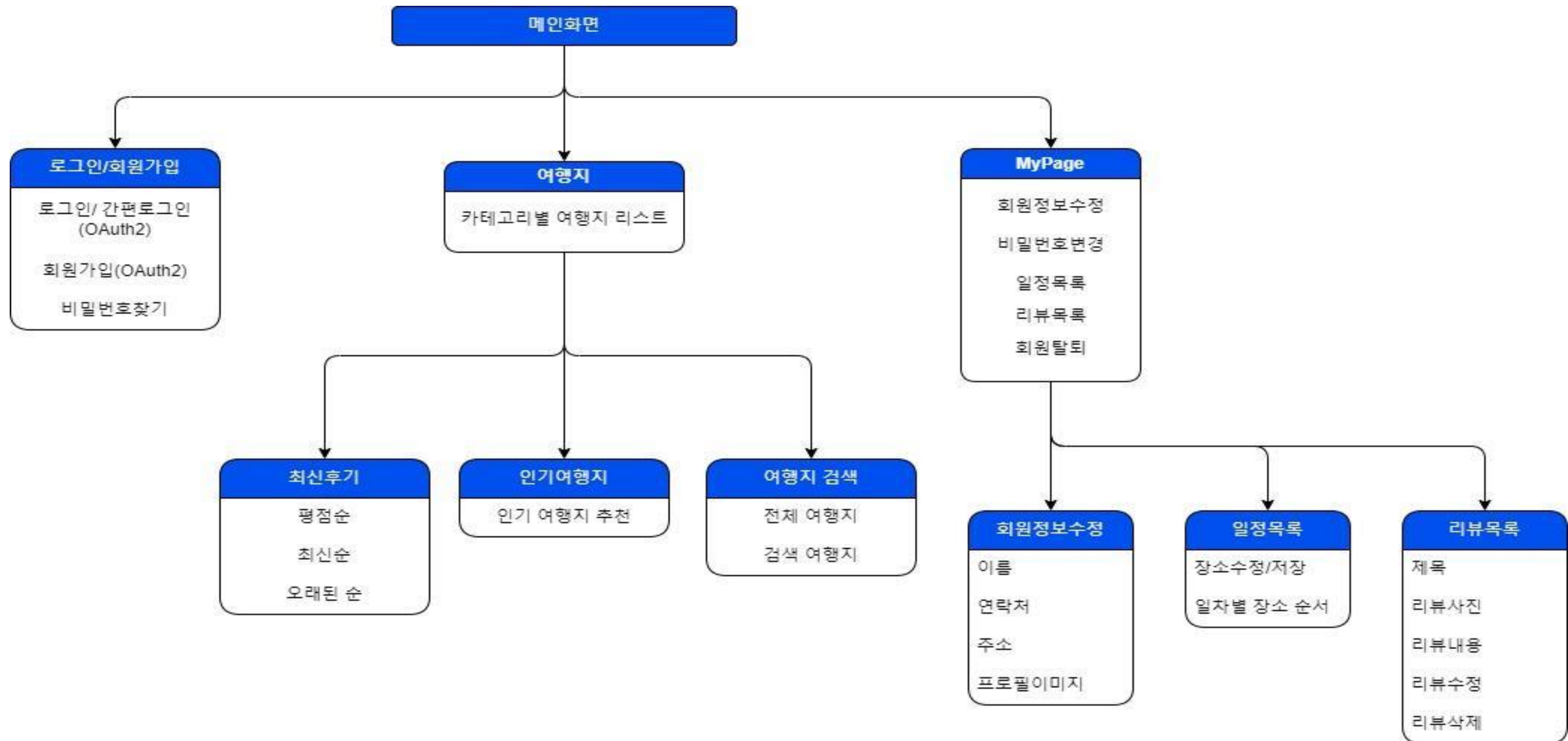
# 참고 사이트



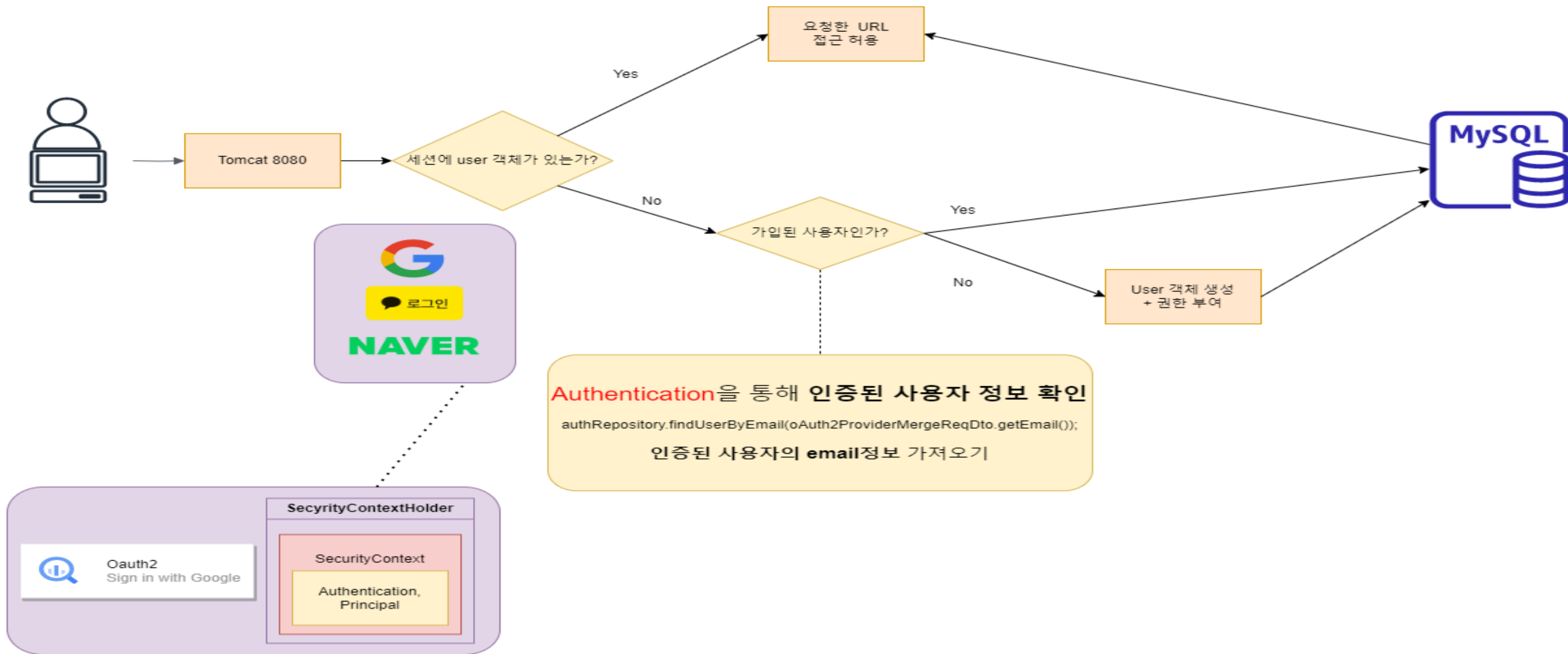
참조 사이트: MYRO



# Front-End – 메인 화면 구조도

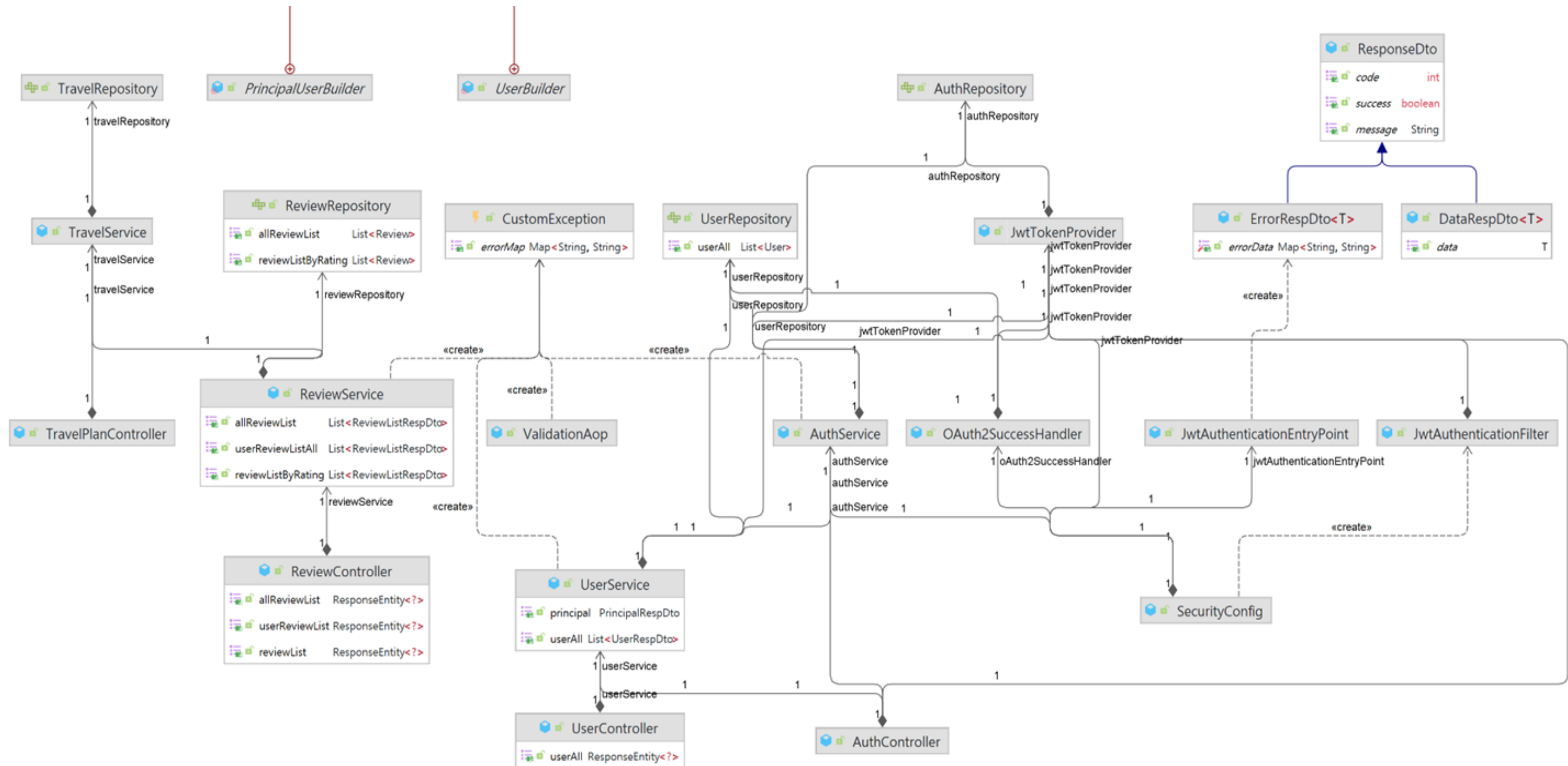


# Back-End - 사용자 인증 과정 구조도





# Back-End Travel Class 구조도



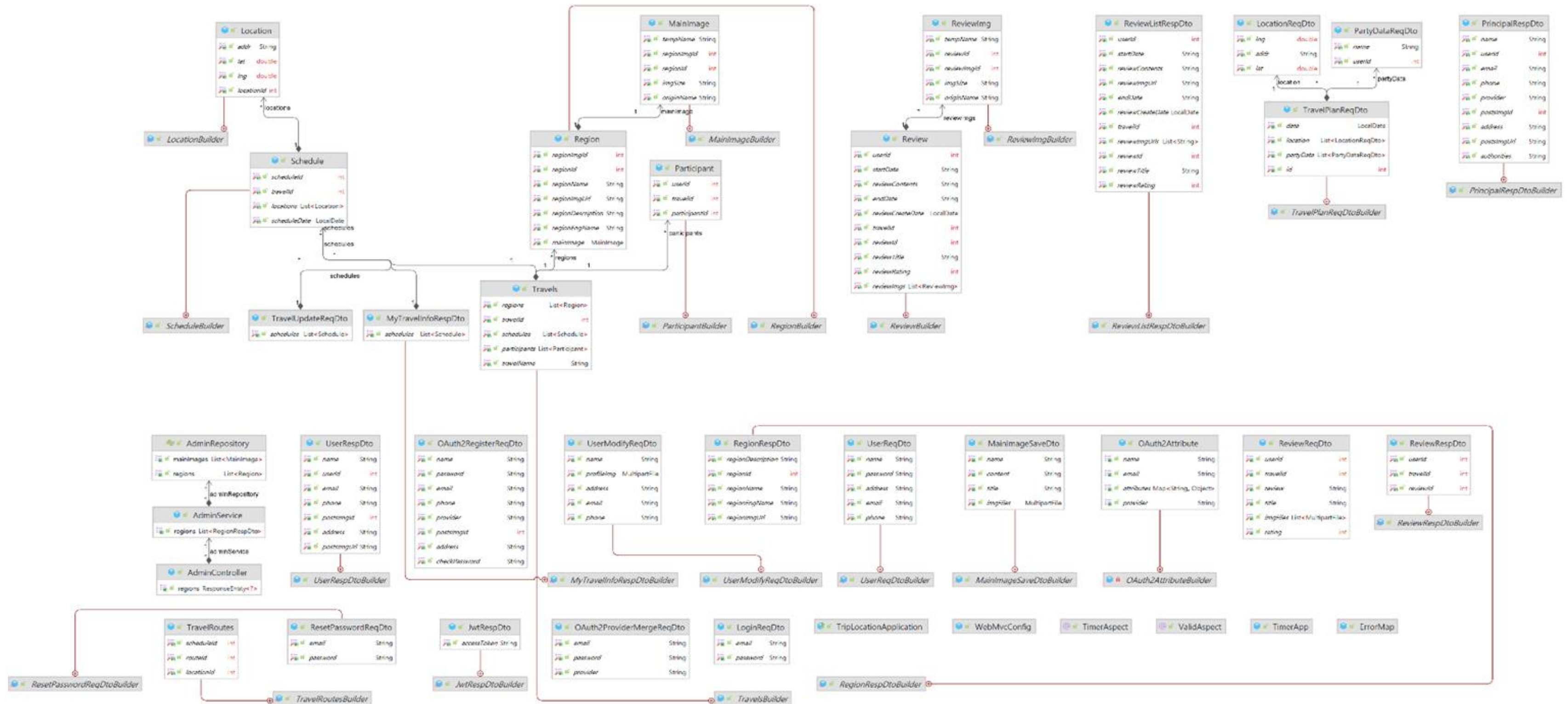


# User Class 구조도



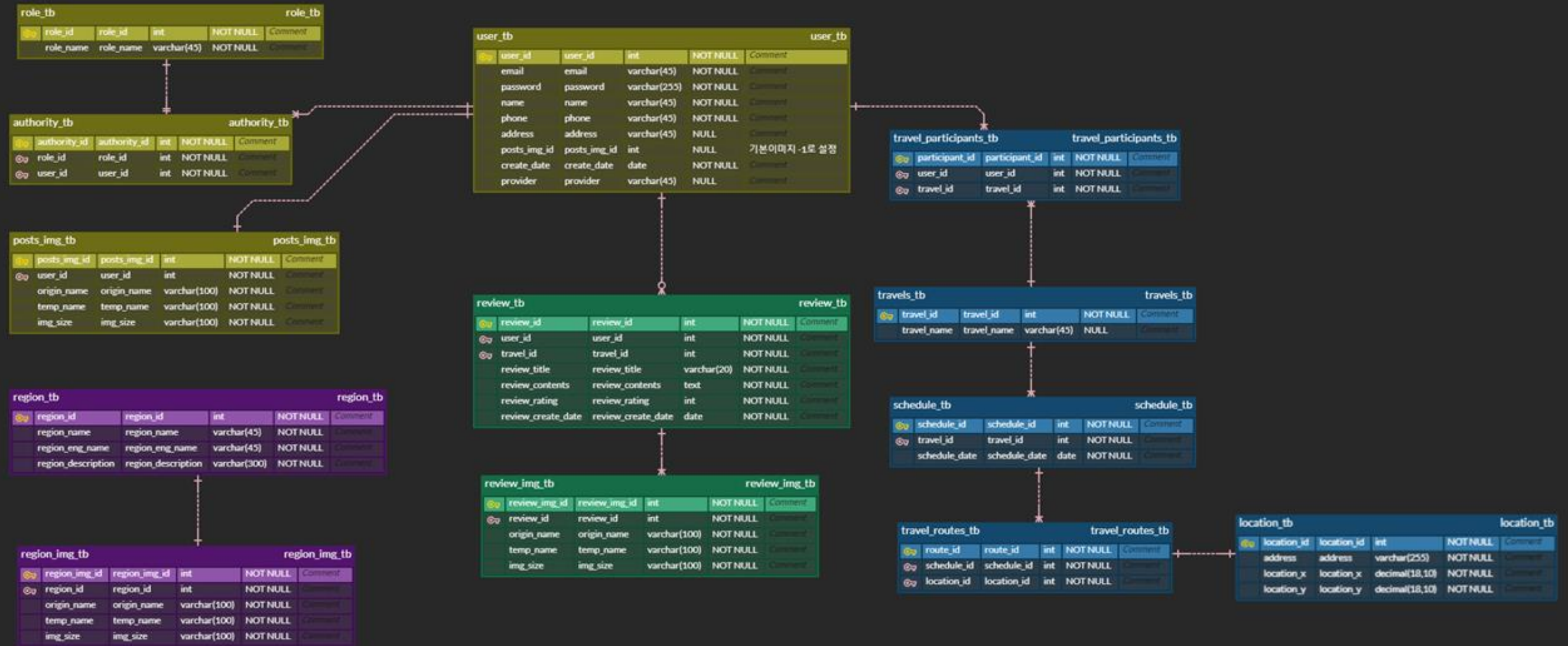


# Back-End Class 구조도





# ERD





# API 명세서

## ▼ Admin API

- 📄 [POST /post/register](#)
- 📄 [GET /post](#)
- 📄 [GET /locations/popoular](#)

## ▼ Auth API ([/api/v1/auth](#))

- 📄 [POST /user](#)
- 📄 [POST /login](#)
- 📄 [PUT /password/reset](#)
- 📄 [GET /authenticated](#)
- 📄 [POST /oauth2/register](#)
- 📄 [PUT /oauth2/merge](#)

## ▼ Review API ([/api/v1/review](#))

- 📄 [GET /mylist](#)
- 📄 [GET /list/{reviewId}](#)
- 📄 [GET /list](#)
- 📄 [GET /list/all](#)
- 📄 [POST /save](#)
- 📄 [PUT /{reviewId}](#)
- 📄 [DELETE /{reviewId}](#)

## ▼ [TravelPlan](#) API ([/api/v1/travel/plan](#))

- 📄 [POST /save](#)
- 📄 [GET /list](#)
- 📄 [GET /info](#)
- 📄 [GET /info/copy](#)
- 📄 [GET /info/review](#)
- 📄 [PUT /update/{travelId}](#)
- 📄 [DELETE /{travelId}](#)

## ▼ User API ([/api/v1/user](#))

- 📄 [GET /all](#)
- 📄 [GET /search](#)
- 📄 [PUT /{userId}](#)
- 📄 [PUT /password/reset](#)
- 📄 [DELETE /{userId}](#)
- 📄 [GET /principal](#)

API 명세서 주소: <https://www.notion.so/API-a2ec865fea9b46b787f548dc4fe201e5>

# 주요기능 코드리뷰

- 유저 정보 관리, 여행 일정 관리, 리뷰 일정 관리

## 유저 정보 관리

C: 회원가입  
R: 로그인한 회원 정보 조회  
- 이메일로 조회  
U: 비밀번호 변경, 이름 변경  
- 프로필 이미지 변경, 주소 변경  
D: 회원 탈퇴

## 여행 일정 관리

C: 일정 생성  
R: 다른 회원 정보 조회  
- 이메일 조회  
- 전화번호 조회  
U: 일정 장소 수정, 함께  
갈 인원 수정  
D: 일정 삭제

## 리뷰 일정 관리

C: 리뷰 생성  
R: 리뷰 조회  
- 최신순 조회  
- 오래된순 조회  
- 평점순 조회  
- 제목순 조회  
U: 리뷰 수정  
- 이미지 변경  
- 여행기 수정  
D: 리뷰 삭제

## Git Hub Repository

- FrontEnd : <https://github.com/KORIT-LSLC/KORIT-LSLC-portfolio-front>
- BackEnd: <https://github.com/KORIT-LSLC/KORIT-LSLC-portfolio-app>



# 주요기능 코드리뷰 – 회원가입(Front-End)

## C: 회원가입

2023 Trip Together

SIGN UP SIGN IN

### Sign Up

이메일\*

비밀번호\*

이름\*

연락처\*

주소

SIGN UP

```
const signupHandleSubmit = async () => {
  try {
    const option = {
      headers: {
        "Content-Type": "application/json",
      },
    };

    const response = await axios.post(
      `http://localhost:8080/api/v1/auth/user`,
      signupUser,
      option
    );

    setMessages({
      profileImgPath: "",
      email: "",
      password: "",
      name: "",
      phone: "",
      address: "",
    });

    const accessToken =
      response.data.grantType + " " + response.data.accessToken;
    localStorage.setItem("accessToken", accessToken);
    alert("회원가입 완료");
    window.location.replace("/auth/login");
    return response;
  } catch (error) {
    setMessages(error.response.data);
  }
};
```

# 주요기능 코드리뷰 - 로그인, 회원가입(Back-End)

- **Domain.User.Entity**
  - user\_tb, role\_tb, authotity\_tb , post\_img\_tb
- **Repository**
  - UserRepository
- **Exception**
  - CustomException, ErrorMap
- **Security**
  - PrincipalUser,
  - jwt(JwtAuthenticationEntryPoint, JwtAuthenticationFilter, JwtTokenProvider),
  - oauth2(OAuth2Attribute, OAuth2SuccessHandler)
- **Service**
  - UserService
- **API .Controller**
  - UserController
- **API.Dto.Request**
  - UserReqDto, UserModifyReqDto,
  - OAuth2RegisterReqDto, OAuth2ProviderMergeReqDto,
  - LoginReqDto,
  - ResetPasswordReqDto
- **API.Dto.Response**
- **Mapper**
  - UserMapper.xml

```
<insert id="saveUser" parameterType="com.korea.triplocation.domain.user.entity.User" useGeneratedKeys="true" keyProperty="userId">
    <choose>
        <when test="provider != null">
            조건에 따라 회원가입 진행 하는 로직입니다 provider가 null 값이 아니면 oauth2의 회원가입에 대한 쿼리
            insert into user_tb (email, password, name, phone, address, posts_img_id, create_date, provider)
            values (#{email}, #{password}, #{name}, #{phone}, #{address}, #{postsImgId}, now(), #{provider})
        </when>
        <otherwise>
            insert into user_tb (email, password, name, phone, address, posts_img_id, create_date)
            values (#{email}, #{password}, #{name}, #{phone}, #{address}, #{postsImgId}, now())
        </otherwise>
    </choose>
</insert>
```

```
public JwtRespDto signin(LoginReqDto loginReqDto) {

    UsernamePasswordAuthenticationToken authenticationToken =
        new UsernamePasswordAuthenticationToken(loginReqDto.getEmail(), loginReqDto.getPassword());
    Authentication authentication =
        authenticationManagerBuilder.getObject().authenticate(authenticationToken);

    return JwtRespDto.builder().accessToken(jwtTokenProvider.generateAccessToken(authentication)).build();
}
```

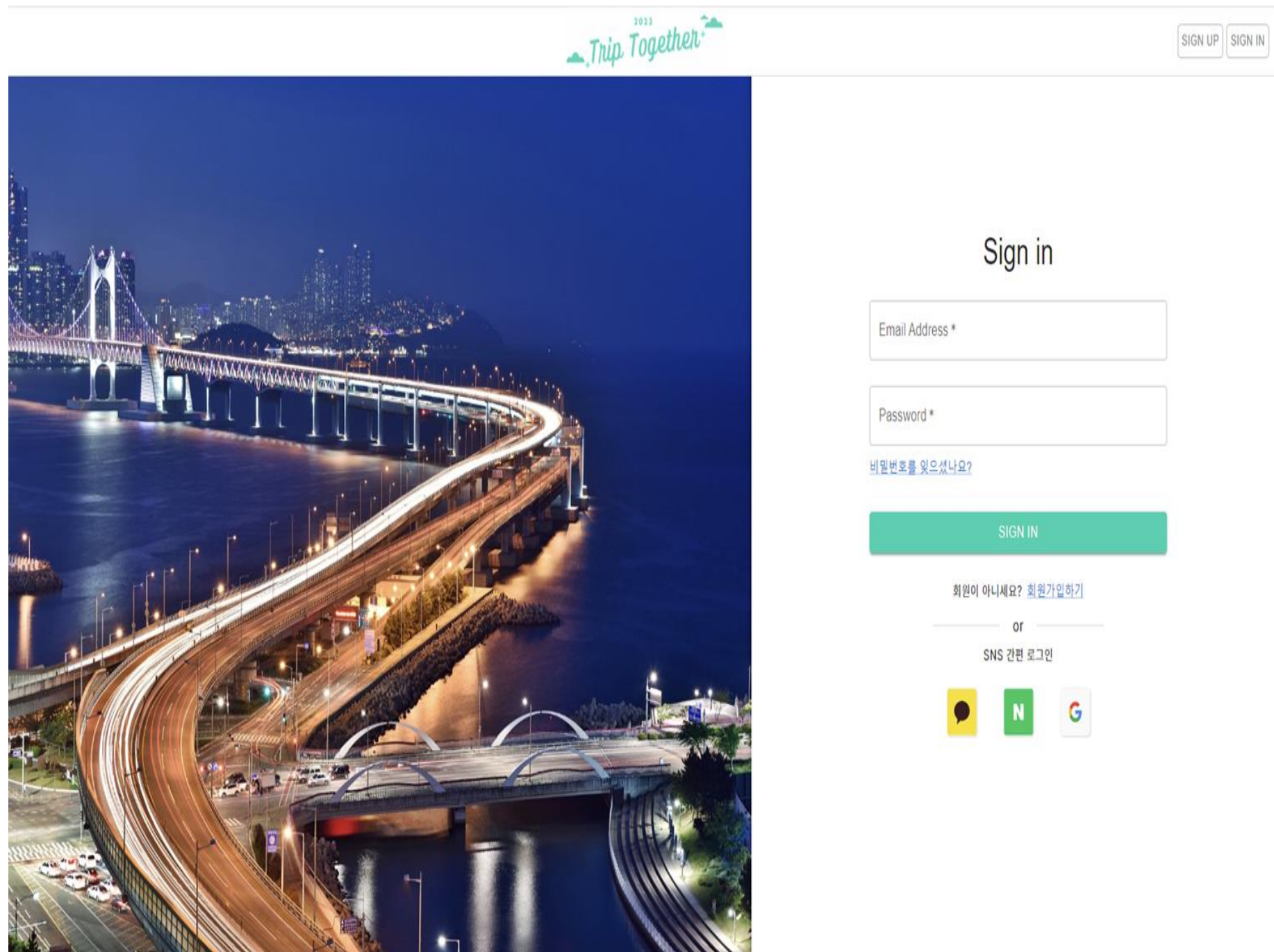
```
@Override
public UserDetails loadUserByUsername (String username) throws UsernameNotFoundException{
    User userEntity = authRepository.findUserByEmail(username);

    if(userEntity == null) {
        throw new CustomException("로그인 실패",
            ErrorMap.builder()
                .put("email", "사용자 정보를 확인하세요.")
                .build());
    }

    PrincipalUser principalUser = userEntity.toPrincipal();
    System.out.println("loadUserByUsername returns: " + principalUser.getClass().getName());
    return principalUser;
}
```



# 주요기능 코드리뷰 – 로그인(Front-End)



```
const signInUser : UseMutationResult<unknown, unknown, void, unknown> = useMutation( options: async (loginData) : Promise<void> => {
  try {
    const option : {headers: {...}} = {
      headers: {
        'Content-Type': 'application/json'
      }
    }

    const response : AxiosResponse<any> = await axios.post( url: 'http://localhost:8080/api/v1/auth/login', loginData, option);


    const accessToken = response.data.accessToken;
    localStorage.setItem('accessToken', accessToken);
    setAuthState( valOrUpdater: {
      isAuthenticated: true,
    });
    navigate('/home');

  }catch (error) {
    alert('아이디 또는 비밀번호를 잘못 입력했습니다. \n입력하신 내용을 다시 확인해주세요. ');
  }
})
```

# 주요기능 코드리뷰 – 회원정보 수정 (Back-End, Front-End)

U: 이름 변경, 프로필 이미지 변경, 주소 변경

Edit Member Information



이메일

aaa@gmail.com

이름

홍길동

Edit

연락처

010-1597-5362

Edit

주소

부산광역시

Edit

MODIFY MEMBER

[회원탈퇴](#)

```
const modifyUser = useMutation(async (modifyData) => {
  try {
    const formData = new FormData();

    formData.append("email", modifyData.email)
    formData.append("name", modifyData.name)
    formData.append("phone", modifyData.phone)
    formData.append("address", modifyData.address)

    if(imgFiles) {
      formData.append("profileImg", imgFiles)
    }

    const option = {
      headers: {
        'Content-Type': 'multipart/form-data',
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    }

    const response = await axios.put(`http://localhost:8080/api/v1/user/${principal.data.data.userId}`, formData, option);

    setErrorMessage({
      profileImg: '',
      email: '',
      name: '',
      phone: '',
      address: ''
    });

    return response
  } catch (error) {
    alert('입력값을 확인해주세요');
  }
}, {
  onSuccess: (response) => {
    alert('회원정보 수정 완료');
    window.location.replace('/');
  }
})
```

```
public boolean modifyUser(UserModifyReqDto userModifyReqDto) {
  PrincipalUser principal = (PrincipalUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
  User user = userRepository.getUserById(principal.getUserId());

  if (user == null) {
    throw new UsernameNotFoundException("User not found");
  }
  if (userModifyReqDto.getEmail() != null) {
    user.setEmail(userModifyReqDto.getEmail());
  }
  if (userModifyReqDto.getName() != null) {
    user.setName(userModifyReqDto.getName());
  }
  if (userModifyReqDto.getPhone() != null) {
    user.setPhone(userModifyReqDto.getPhone());
  }
  if (userModifyReqDto.getAddress() != null) {
    user.setAddress(userModifyReqDto.getAddress());
  }
  if (userModifyReqDto.getProfileImg() != null) {
    System.out.println(userId);
    PostsImg currentPostsImg = userRepository.getPostsImgByUserId(principal.getUserId());
    // If user has a profile image, delete it
    if (currentPostsImg != null) {
      try {
        deleteFile(currentPostsImg.getTempName());
      } catch (IOException e) {
        throw new RuntimeException(e);
      }
    }
    userRepository.deletePostsImg(currentPostsImg.getPostsImgId());

    PostsImg postsImg = uploadFile(principal.getUserId(), userModifyReqDto.getProfileImg());
    userRepository.postsImg(postsImg);
    user.setPostsImgId(postsImg.getPostsImgId());
  }

  if (currentPostsImg == null && user.getPostsImgId() == -1) {
    PostsImg postsImg = uploadFile(principal.getUserId(), userModifyReqDto.getProfileImg());
    userRepository.postsImg(postsImg);
    user.setPostsImgId(postsImg.getPostsImgId());
  }

  return userRepository.modifyUser(user) != 0;
}
```



# 주요기능 코드리뷰 – 회원정보 수정 (Back-End, Front-End)

R: 로그인한 회원 정보 조회  
U: 비밀번호 변경

New Password

이메일

aaa@gmail.com

비밀번호 \*

새로운 비밀번호를 입력해주세요.

비밀번호 확인 \*

NEXT

```
const modifyUserPassword = useMutation(async (modifyData) => {
  try {
    const modifyUserData = {
      email: principal.data.data.email,
      password: updateUserPassword.password
    };

    const option = {
      headers: {
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    };

    const response = await axios.put('http://localhost:8080/api/v1/user/password/reset', modifyUserData, option);

    return response
  } catch (error) {
    alert('ERROR');
  }
}, {
  onSuccess: (response) => {
    if (response.status === 200) {
      alert('비밀번호가 변경되었습니다.');
      navigate('/user/:id');
    }
  }
})
```

```
public boolean resetPassword(ResetPasswordReqDto resetPasswordReqDto) {
  User user = userRepository.searchUserByEmail(resetPasswordReqDto.getEmail());

  if (user == null) {
    throw new UsernameNotFoundException("User not found");
  }
  if (resetPasswordReqDto.getPassword() != null) {
    user.setPassword(new BCryptPasswordEncoder().encode(resetPasswordReqDto.getPassword()));
  }
  return userRepository.resetPassword(user) != 0;
}
```

# 주요기능 코드리뷰 – 회원정보 탈퇴 (Back-End, Front-End)

R: 로그인한 회원 정보 조회 체크

D: 회원 탈퇴

회원정보 확인

NEXT

```
const checkUser = useMutation(async (userCheck) => {
  try {
    const userData = {
      email: userCheck.email,
      password: userCheck.password,
    };

    const response = await axios.delete(`http://localhost:8080/api/v1/user/${principal.data.data.userId}`, {
      headers: {
        Authorization: `${localStorage.getItem("accessToken")}`,
      },
      data: userData,
    });

    return response;
  } catch (error) {
    alert("로그인된 이메일과 비밀번호를 확인해주세요.");
  }
}, {
  onSuccess: (response) => {
    if(response.status === 200) {
      alert('지금까지 Trip Together를 이용해주셔서 감사합니다. ');
      localStorage.removeItem("accessToken");
      setAuthState({
        isAuthenticated: false,
      });
      navigate("/auth/login");
    }
  }
});
```

```
public boolean deleteUser( LoginReqDto loginReqDto) {
  boolean flag = false;

  // 한 번더 로그인하는 것으로 본인 확인
  if(authService.signin(loginReqDto) != null) {
    PrincipalUser principal = (PrincipalUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    User user = userRepository.getUserById(principal.getUserId());
    int postsImgId = user.getPostsImgId();

    if (postsImgId != -1) {
      PostsImg currentPostsImg = userRepository.getPostsImgById(postsImgId);

      if (currentPostsImg != null) {
        try {
          deleteFile(currentPostsImg.getTemplateName());
        } catch (IOException e) {
          throw new RuntimeException(e);
        }
      }
      userRepository.deleteUser(principal.getUserId());
      flag = true;
    } else {
      throw new CustomException("이메일 또는 비밀번호를 확인해주세요");
    }
  }

  return flag;
}
```



# 주요기능 코드리뷰 - 여행 생성

- **Domain.User.Entity**
  - Travel, TravelRoutes, Schedule, Region, Participant, MainImage, Location
- **Repository**
  - TravelRepository
- **Exception**
  - CustomException, ErrorMap
- **Service**
  - TravelService
- **API.Controller**
  - TravelPlanController
- **API.Dto.Request**
  - TravelPlanReqDto, TravelUpdateReqDto,
  - PartyDataReqDto, MainImageSaveDto, LocationReqDto
- **API.Dto.Response**
  - PopularLocationsRespDto
  - RegionRespDto,
- **Mapper**
  - TravelMapper.xml

<insert id="callInsertTravelData">

{CALL InsertTravelData(# {travelName}, # {addr}, # {lat}, # {lng}, # {userId}, # {scheduleDate})}

</insert>

```
CREATE DEFINER='admin'@'%' PROCEDURE `InsertTravelData`(IN travelName VARCHAR(255), IN addr VARCHAR(255), IN lat decimal(18,10), IN lng decimal(18,10), IN userId integer, IN scheduleDate date)
BEGIN
    DECLARE v_travelId INT DEFAULT (SELECT MAX(travel_id) FROM travels_tb);
    DECLARE v_locationId INT;
    DECLARE v_scheduleId INT;

    -- Check if the travel name is NULL
    IF travelName IS NOT NULL THEN
        -- Insert into travels_tb and get the generated ID
        INSERT INTO travels_tb(travel_name) VALUES (travelName);
        SET v_travelId = LAST_INSERT_ID();
    END IF;

    -- Check if the same dateId and visitDate already exist in schedule_tb
    IF NOT EXISTS (SELECT 1 FROM schedule_tb WHERE travel_id = v_travelId AND schedule_date = scheduleDate) THEN
        -- Insert into schedule_tb and get the generated ID
        INSERT INTO schedule_tb(travel_id, schedule_date) VALUES (v_travelId, scheduleDate);
        SET v_scheduleId = LAST_INSERT_ID();
    ELSE
        -- If record already exists, get the schedule_id
        SET v_scheduleId = (SELECT schedule_id FROM schedule_tb WHERE travel_id = v_travelId AND schedule_date = scheduleDate);
    END IF;

    -- Insert into location_tb and get the generated ID
    IF NOT EXISTS (SELECT 1 FROM location_tb WHERE address = addr) THEN
        INSERT INTO location_tb(address, location_x, location_y) VALUES (addr, lat, lng);
        SET v_locationId = LAST_INSERT_ID();
    ELSE
        SET v_locationId = (SELECT location_id FROM location_tb WHERE address = addr);
    END IF;

    -- Insert into travel_participants_tb, ignore if the pair (travel_id, user_id) already exists
    IF NOT EXISTS (SELECT 1 FROM travel_participants_tb WHERE travel_id = v_travelId AND user_id = userId) THEN
        INSERT INTO travel_participants_tb(travel_id, user_id) VALUES (v_travelId, userId);
    END IF;

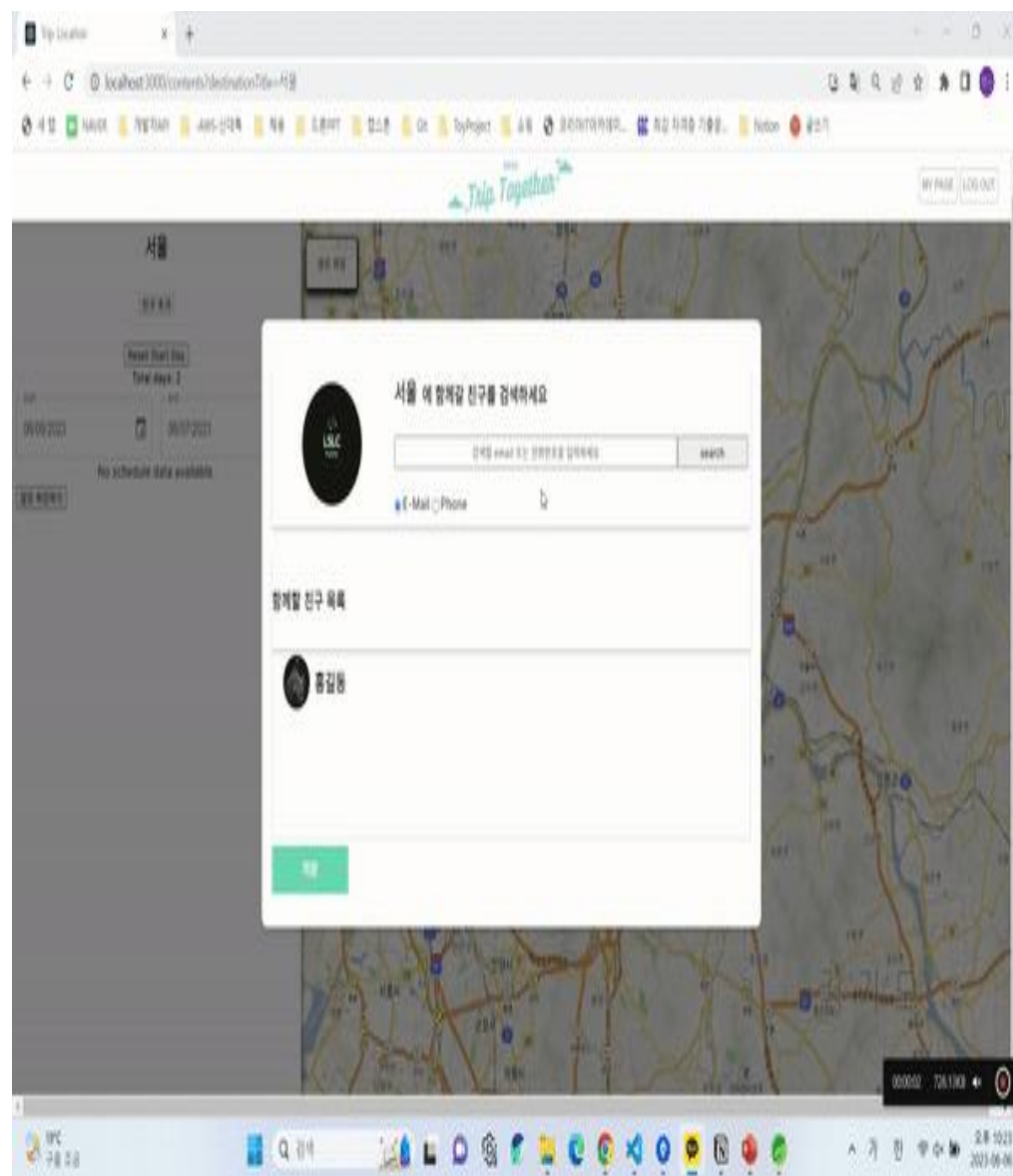
    -- Insert into travel_routes_tb
    IF NOT EXISTS (SELECT 1 FROM travel_routes_tb WHERE schedule_id = v_scheduleId AND location_id = v_locationId) THEN
        INSERT INTO travel_routes_tb(schedule_id, location_id) VALUES (v_scheduleId, v_locationId);
    END IF;

    SELECT v_travelId, v_locationId, v_scheduleId;
END
```

# 주요기능 코드리뷰 – 함께 갈 인원 수정 (Back-End, Front-End)

R: 다른 회원 정보 조회

U: 함께 갈 인원 수정



```
const searchUser = useQuery(['searchUser', searchType, searchValue], async() => {
  if (!searchValue) {
    return null;
  }
  const params = {
    type: searchType === 'email' ? 1 : 2,
    value: searchValue,
  }

  const option = {
    params,
    headers: {
      Authorization: `${localStorage.getItem("accessToken")}`
    }
  }
  try {
    const response = await axios.get('http://localhost:8080/api/v1/user/search', option);

    return response;
  } catch (error) {
    return error;
  }
},{
  enabled: userInfo.userId !== '',
  onSuccess: (response) => {
    if (response?.data?.data) {
      const newSearchInfo = {
        userId: response.data.data.userId,
        email: response.data.data.email,
        name: response.data.data.name,
        phone: response.data.data.phone,
        profileImg: response.data.data.postsImgUrl,
      };
      setSearchInfo(newSearchInfo);

      setPartyUsers((prevPartyUsers) => {
        const isAlreadyAdded = prevPartyUsers.some((party) => party.userId === newSearchInfo.userId);
        if (!isAlreadyAdded) {
          return [...prevPartyUsers, newSearchInfo];
        }
        return prevPartyUsers;
      });
    }
  }
});
```

```
public List<Travels> findTravelByUser() {
  PrincipalUser principal = (PrincipalUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
  if(principal.getUserId() == 0) return null;

  return setTravelImages(travelRepository.findTravelAllByUser(principal.getUserId()));
}

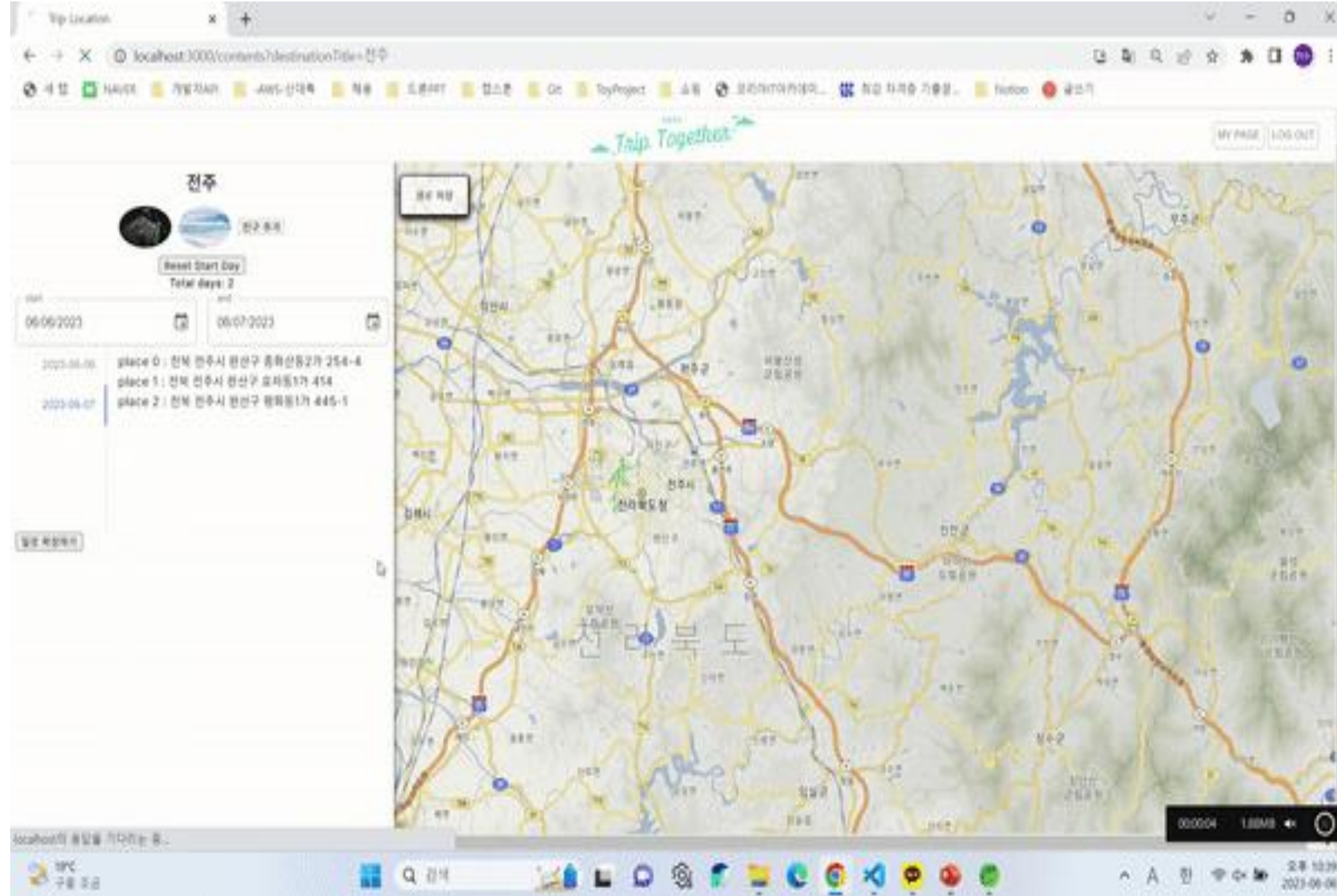
public UserRespDto searchUser(int type, String value) {
  User user = null;
  PostsImg postsImg = null;
  String imgUrl = null;
  int postsImgId = 0;
  if (type == 1) {
    user = userRepository.searchUserByEmail(value);
    if (user != null) {
      postsImgId = user.getPostsImgId();
      if (postsImgId != -1) {
        postsImg = userRepository.getPostsImgById(postsImgId);
        if (postsImg != null) {
          imgUrl = convertFilePathToUrl(postsImg.getTempName());
        } else {
          imgUrl = convertFilePathToUrl("default.png");
        }
      } else {
        imgUrl = convertFilePathToUrl("default.png");
      }
    }
  } else if (type == 2) {
    user = userRepository.searchUserByPhone(value);
    if (user != null) {
      postsImgId = user.getPostsImgId();
      if (postsImgId != -1) {
        postsImg = userRepository.getPostsImgById(postsImgId);
        if (postsImg != null) {
          imgUrl = convertFilePathToUrl(postsImg.getTempName());
        } else {
          imgUrl = convertFilePathToUrl("default.png");
        }
      } else {
        imgUrl = convertFilePathToUrl("default.png");
      }
    }
  } else {
    throw new IllegalArgumentException("Invalid type");
  }

  if (user == null) {
    return null;
  }

  return UserRespDto.builder()
    .userId(user.getUserId())
    .postsImgId(user.getPostsImgId())
    .email(user.getEmail())
    .name(user.getName())
    .phone(user.getPhone())
    .postsImgUrl(imgUrl)
    .build();
}
```



# 주요기능 코드리뷰 – 여행생성(Front)



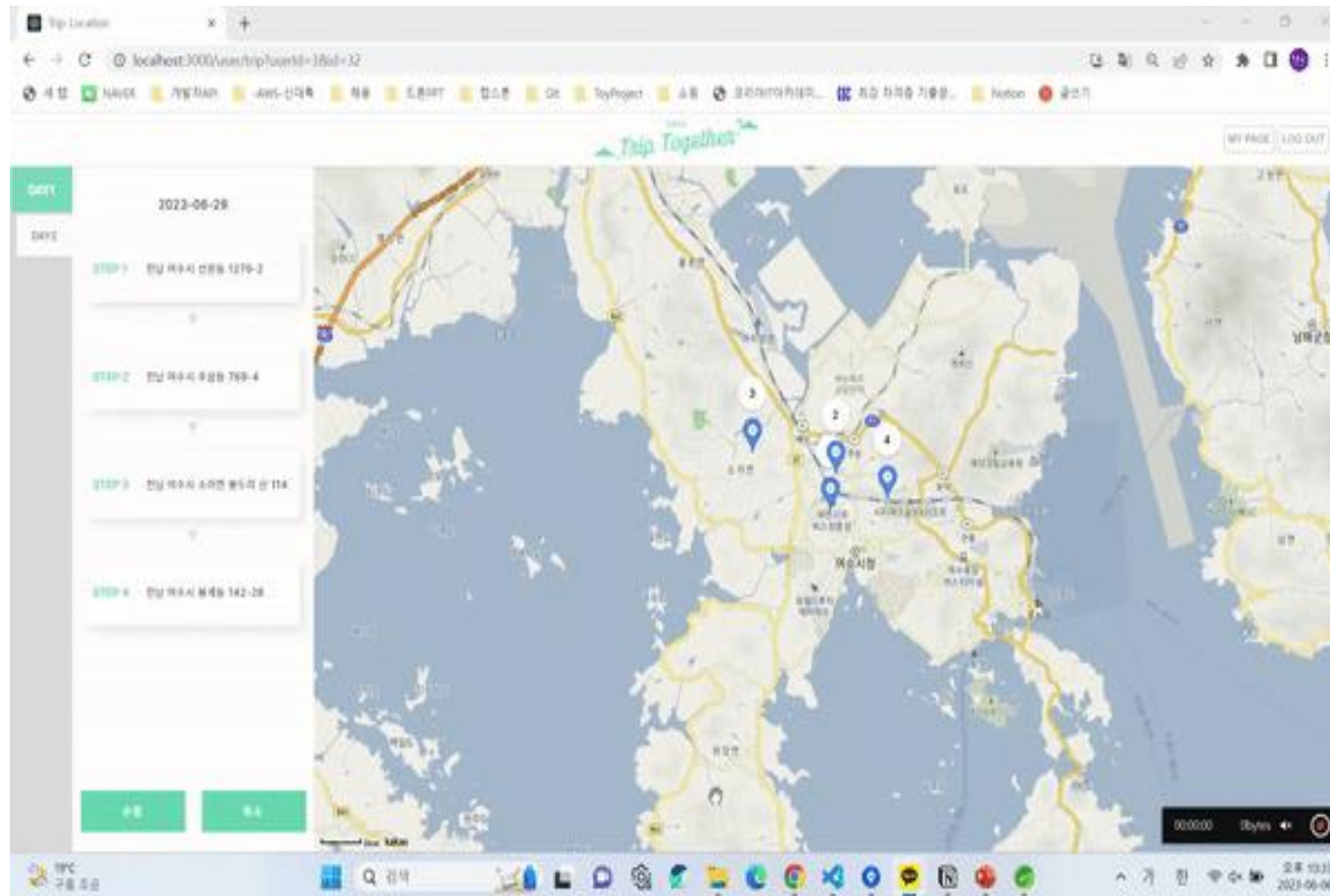
```
const requestData = useMutation(async (updatedScheduleData) => {

  const option = {
    headers: {
      'Content-Type': 'application/json',
      Authorization: `${localStorage.getItem("accessToken")}`
    }
  }
  try {
    const response = await axios.post("http://localhost:8080/api/v1/travel/plan/save", updatedScheduleData, option)

    window.location.replace(`/user/${userInfo.userId}`)
    return response;
  } catch (error) {
    return error;
  }
}, {
  onSuccess: (response) => {
    localStorage.removeItem("scheduleData");
  }
})
```

# 주요기능 코드리뷰 – 여행일정 수정 (Front-End)

R: 여행 ID로 나의 여행 일정 정보 조회  
U: 일정 장소 정보 수정

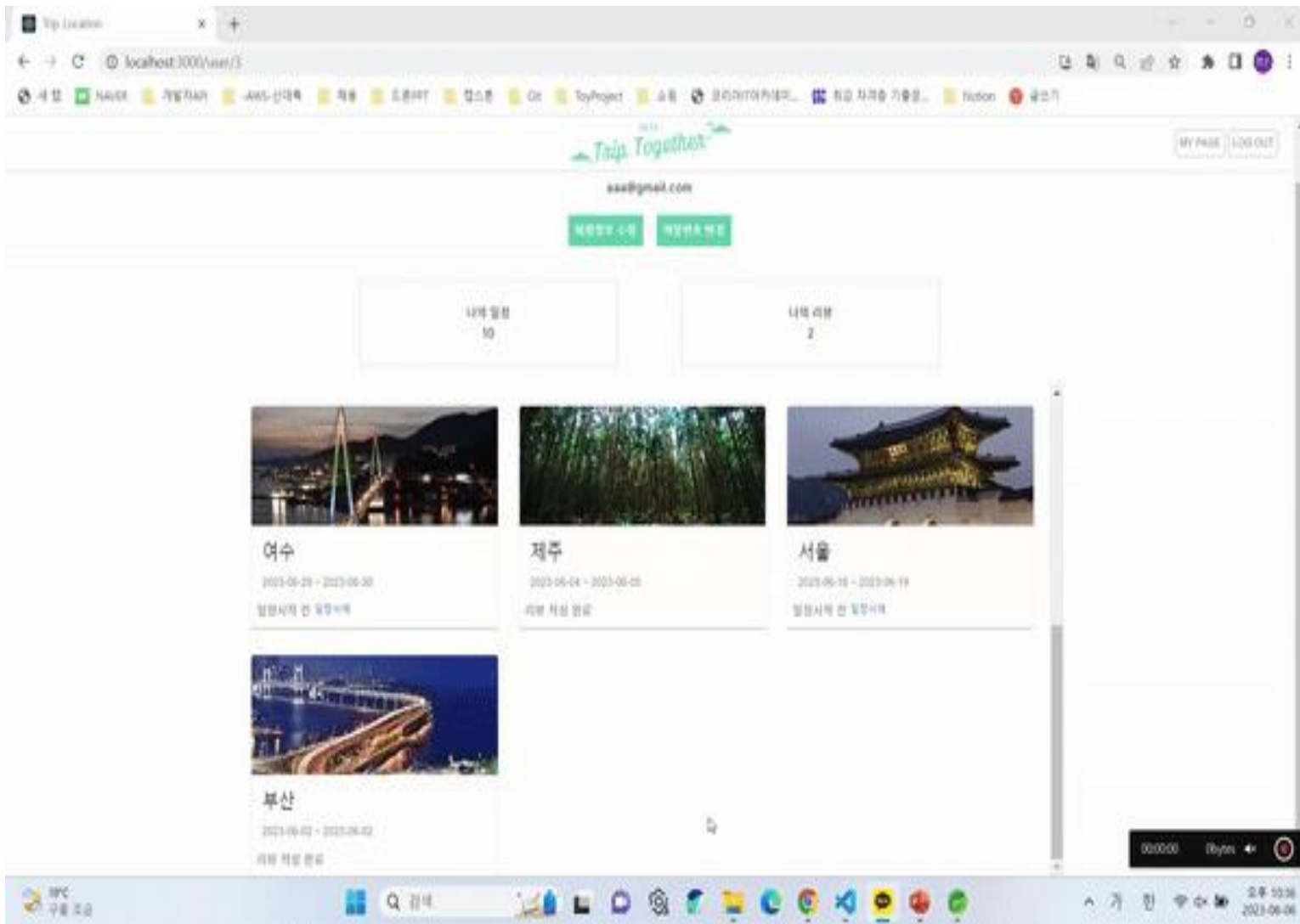


```
const updateTravelInfo = useMutation(async (travelPlan) => {  
  const travelId = parseInt(searchParams.get('id'));  
  const option = {  
    headers: {  
      'Content-Type': 'application/json',  
      Authorization: `${localStorage.getItem('accessToken')}`  
    }  
  }  
  try {  
    const response = await axios.put(`http://localhost:8080/api/v1/travel/plan/update/${travelId}`, travelPlan, option);  
    window.location.replace(`/user/${searchParams.get('userId')}`);  
    return response;  
  } catch (error) {  
  }  
},)
```



# 주요기능 코드리뷰 – 여행일정 삭제 (Front-End, Back-End)

R: 여행 ID로 나의 여행 일정 정보 조회  
D: 여행일정 삭제



```
const deletePlan = useMutation(async (travelId) => {
  try {
    const option = {
      headers: {
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    }
    const response = await axios.delete(`http://localhost:8080/api/v1/travel/plan/${travelId}`, option);

    return response
  } catch (error) {
  }
}, {
  onSuccess: (response) => {
    if(response.status === 200) {
      navigate('/home');
    }
  }
})

public int deleteTravelPlan(int travelId) {
  PrincipalUser principal = (PrincipalUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
  return deleteTravelPlanForUser(principal, travelId);
}

private int deleteTravelPlanForUser(PrincipalUser principal, int travelId){
  for (Travels travels : travelRepository.findTravelAllByUser(principal.getUserId())) {
    if(deleteTravelPlanForParticipant(travels, principal, travelId)) return 1;
  }
  return -1;
}

private boolean deleteTravelPlanForParticipant(Travels travels, PrincipalUser principal, int travelId){
  for (Participant participant: travels.getParticipants()) {
    if(participant.getTravelId() == travelId && participant.getUserId() == principal.getUserId()) {
      Participant participantIdByUserIdAndTravelId = travelRepository.findParticipantIdByUserIdAndTravelId(principal.getUserId(), travelId);
      if (participantIdByUserIdAndTravelId != null) {
        travelRepository.deleteTravelPlanByParty(participantIdByUserIdAndTravelId.getParticipantId());
        return true;
      }
    }
  }
  return false;
}
```

# 주요기능 코드리뷰 - 리뷰 작성

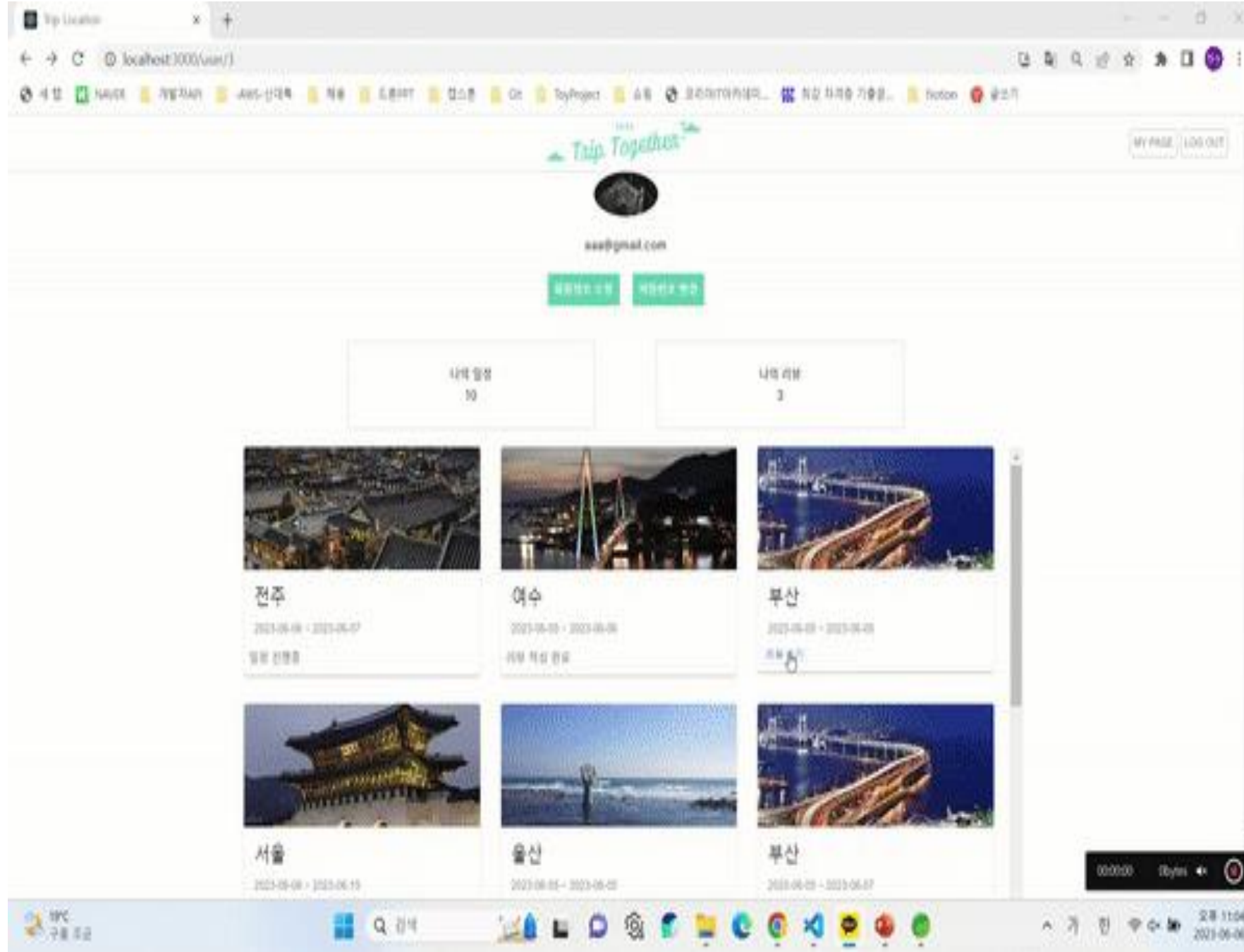
- **Domain.User.Entity**
  - Review, ReviewImg
- **Repository**
  - ReviewRepository
- **Exception**
  - CustomException, ErrorMap
- **Service**
  - ReviewService
- **API.Controller**
  - ReviewController
- **API.Dto.Request**
  - ReviewReqDto,
- **API.Dto.Response**
  - ReviewListRespDto
  - ReviewRespDto
- **Mapper**
  - ReviewMapper.xml

```
<insert id="registerReviews" parameterType="com.korea.triplocation.domain.review.entity.Review"
        useGeneratedKeys="true" keyProperty="reviewId">
    insert into review_tb
    values (0, #{userId}, #{travelId}, #{reviewContents}, #{reviewTitle}, now(), #{reviewRating} )
</insert>

<insert id="registerReviewImgs" parameterType="list">
    insert into review_img_tb
    values
    <foreach collection="list" item="reviewImg" separator=",">
        (0, #{reviewImg.reviewId}, #{reviewImg.originName}, #{reviewImg.tempName}, #{reviewImg.imgSize})
    </foreach>
</insert>
```



# 주요기능 코드리뷰 – 리뷰작성(Front-End)



```
const saveReview = useMutation(async (reviewData) => {
  try{
    const formData = new FormData();
    formData.append('review', reviewData.review);
    formData.append('title', reviewData.title);
    formData.append('travelId', reviewData.travelId);
    formData.append('userId', reviewData.userId);
    formData.append('rating', value);

    imgFiles.forEach(imgFile => {
      formData.append('imgFiles', imgFile.file);
    })

    const option = {
      headers: {
        'Content-Type': 'multipart/form-data',
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    }

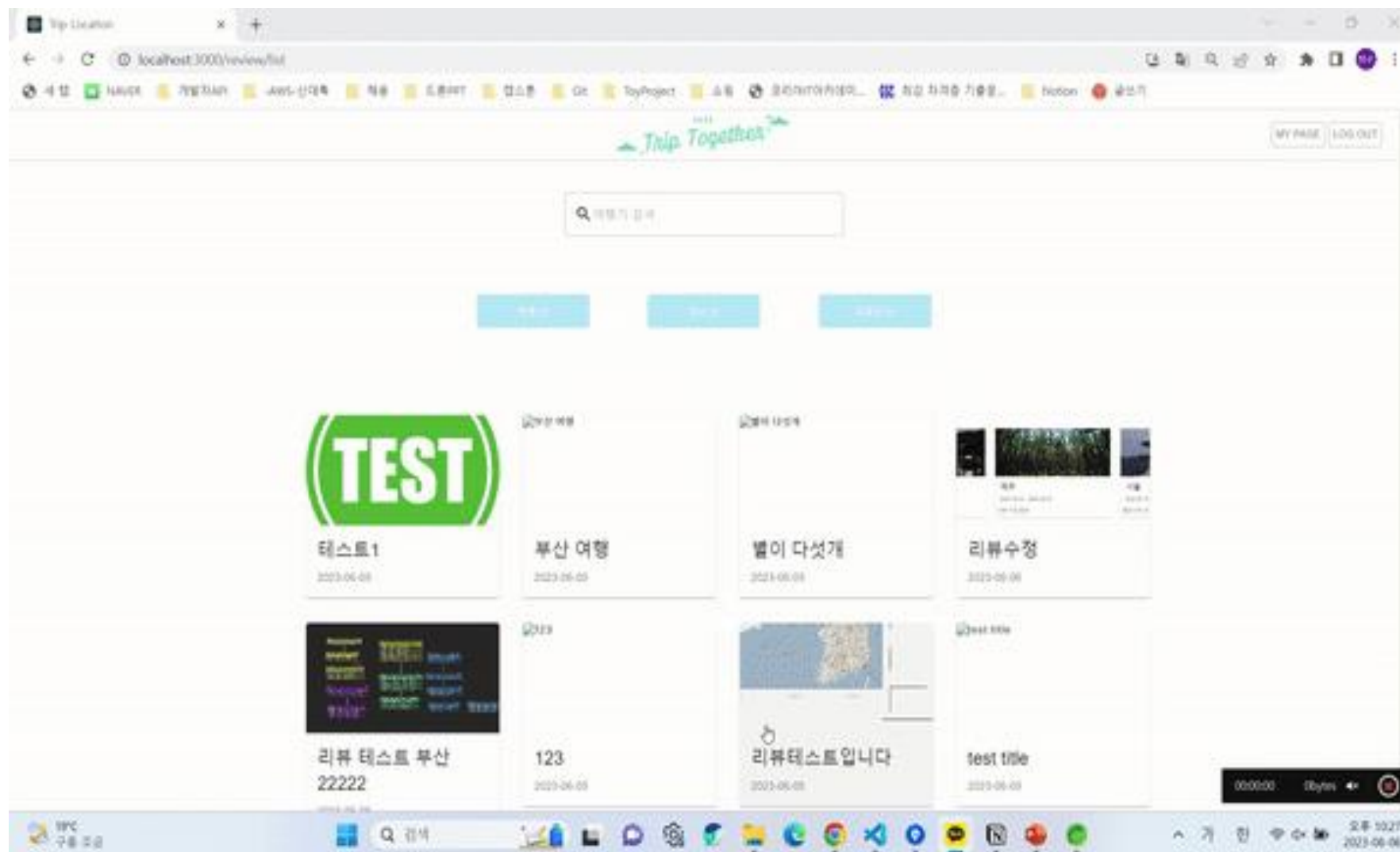
    const response = await axios.post('http://localhost:8080/api/v1/review/save', formData, option);
    return response
  }catch(error) {

  }
}, {
  onSuccess: (response) => {
    if(response.status === 200) {
      navigate(`/user/${searchParams.get('userId')}`);
    }
  }
})
})
```

# 주요기능 코드리뷰 – 리뷰 조회 (Front-End, Back-End)

R: 리뷰 조회

- 최신순 조회, 오래된 순 조회, 평점순 조회, 제목 조회



```
const reviewList = useQuery(['list'], async () => {
  try {
    const response = await axios.get('http://localhost:8080/api/v1/review/list/all');

    return response
  } catch (error) {
    console.error('Failed to fetch reviews', error);
  }
}, {
  onSuccess: (response) => {
    setReview([...response.data]);
  }
}));

const handleSearch = (event) => {
  setSearchTerm(event.target.value);
};

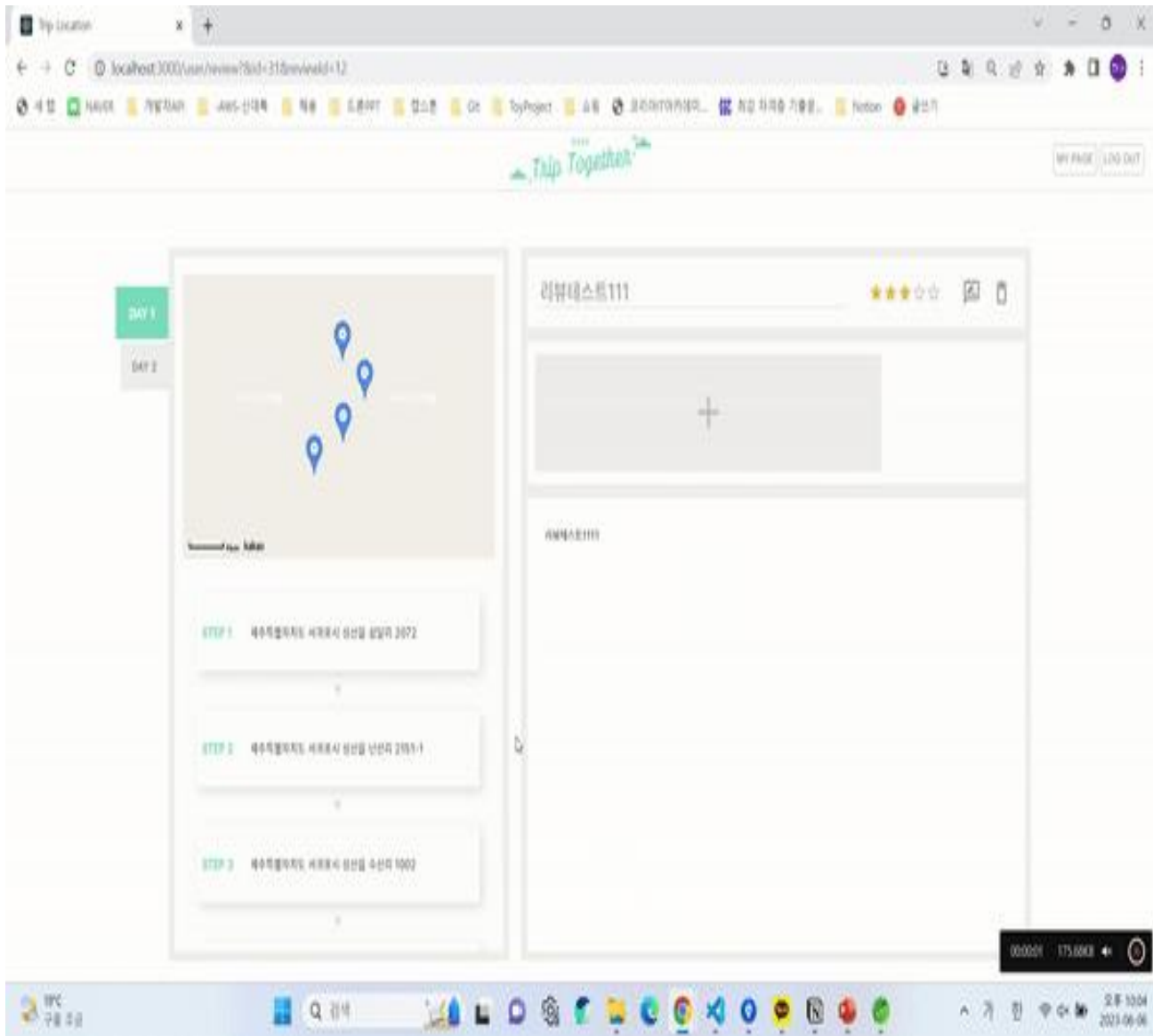
const filteredReviews = review.filter(review =>
  review.reviewTitle.toLowerCase().includes(searchTerm.toLowerCase())
);

const handleCardClick = (reviewId, travelId) => {
  navigate(`/review/list/detail?id=${travelId}&reviewId=${reviewId}`)
};
```



# 주요기능 코드리뷰 – 리뷰 수정 (Front-End, Back-End)

U: 리뷰수정 – 이미지 수정, 여행기 수정



```
const deletePlan = useMutation(async (travelId) => {
  try {
    const option = {
      headers: {
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    }
    const response = await axios.delete(`http://localhost:8080/api/v1/travel/plan/${travelId}`, option);

    return response
  } catch (error) {
  }
}, {
  onSuccess: (response) => {
    if(response.status === 200) {
      navigate('/home');
    }
  }
})

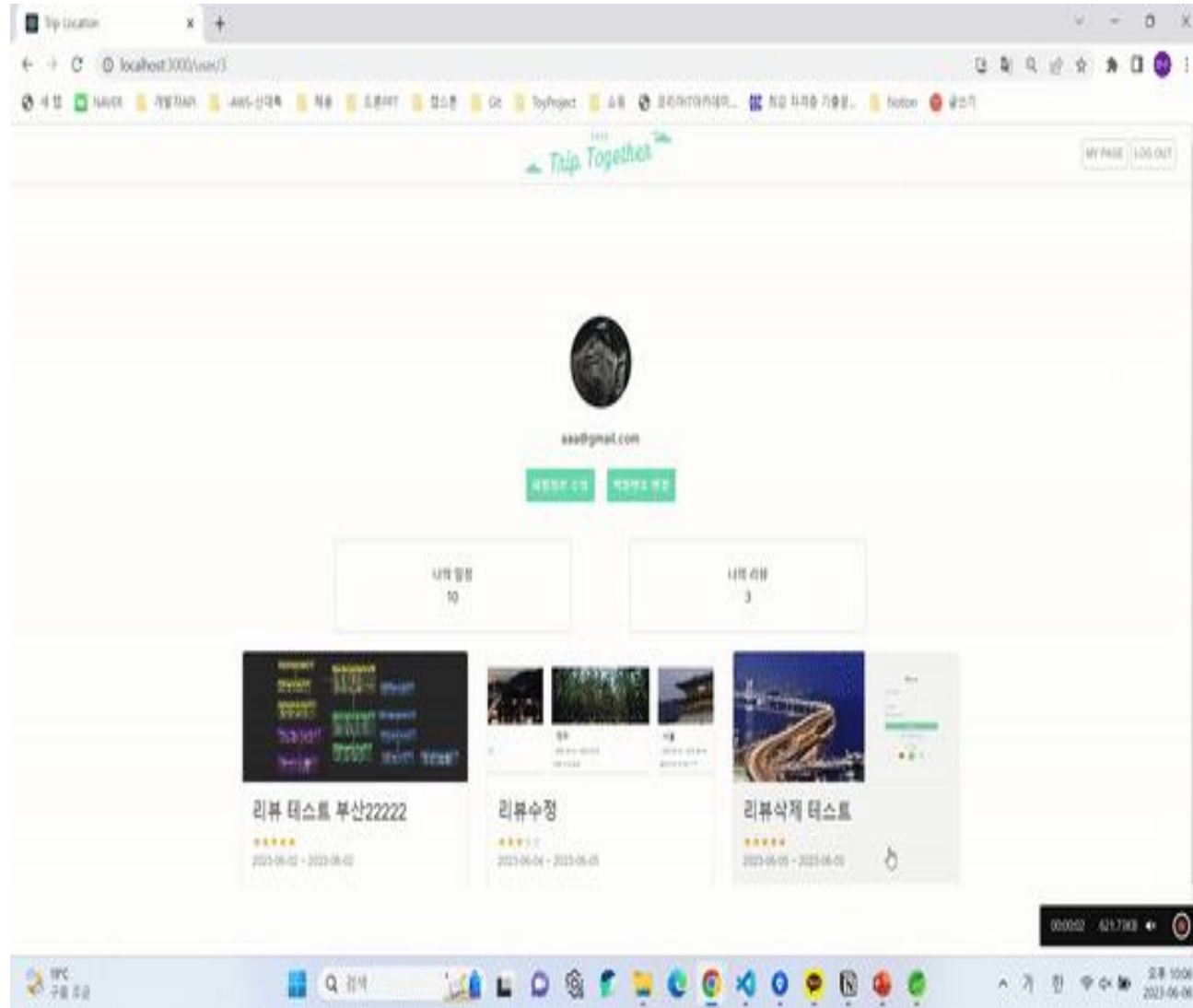
public int deleteTravelPlan(int travelId) {
  PrincipalUser principal = (PrincipalUser) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
  return deleteTravelPlanForUser(principal, travelId);
}

private int deleteTravelPlanForUser(PrincipalUser principal, int travelId){
  for (Travels travels : travelRepository.findTravelAllByUser(principal.getUserId())) {
    if(deleteTravelPlanForParticipant(travels, principal, travelId)) return 1;
  }
  return -1;
}

private boolean deleteTravelPlanForParticipant(Travels travels, PrincipalUser principal, int travelId){
  for (Participant participant: travels.getParticipants()) {
    if(participant.getTravelId() == travelId && participant.getUserId() == principal.getUserId()) {
      Participant participantIdByUserIdAndTravelId = travelRepository.findParticipantIdByUserIdAndTravelId(principal.getUserId(), travelId);
      if (participantIdByUserIdAndTravelId != null) {
        travelRepository.deleteTravelPlanByParty(participantIdByUserIdAndTravelId.getParticipantId());
        return true;
      }
    }
  }
  return false;
}
```

# 주요기능 코드리뷰 – 리뷰 삭제 (Front-End, Back-End)

R: 리뷰 ID로 나의 리뷰 일정 정보 조회  
D: 리뷰 삭제



```
const reviewDelete = useMutation(async (deleteId) => {
  try {
    const option = {
      headers: {
        Authorization: `${localStorage.getItem('accessToken')}`
      }
    }
    const response = await axios.delete(`http://localhost:8080/api/v1/review/${deleteId}`, option);
    return response
  } catch (error) {
  }
}, {
  onSuccess: (response) => {
    if (response.status === 200) {
      navigate('/home', {replace: true});
    }
  }
})

private void deleteFile(String filename) throws IOException {
  Path uploadPath = Paths.get(filePath + "review/" + filename);
  if (Files.exists(uploadPath)) {
    try {
      Files.delete(uploadPath);
    } catch (IOException e) {
      throw new IOException("Failed to delete file: " + uploadPath, e);
    }
  } else {
    throw new FileNotFoundException("File not found: " + uploadPath);
  }
}

@Transactional
public int deleteReview(int reviewId) {
  List<ReviewImg> reviewImgListByReviewId = reviewRepository.getReviewImgListByReviewId(reviewId);
  if (reviewImgListByReviewId != null) {
    try {
      for (ReviewImg reviewImg : reviewImgListByReviewId) {
        reviewRepository.deleteReviewImages(reviewImg.getReviewImgId());
        deleteFile(reviewImg.getTempName());
      }
    } catch (IOException e) {
      throw new RuntimeException(e);
    }
  }
  int deleteReview = reviewRepository.deleteReview(reviewId);
  return deleteReview ;
}
```



# 고찰

## ❖ 결과

- 로그인, 간편로그인
- 여행 일정 생성
- 리뷰 작성

## ❖ 문제점

- 이메일 인증이 안되어 비밀번호 변경 보안에 취약함.
- 리뷰 작성에서 사진 등록시 원하는 순서대로 사진을 넣을 수 없음.

## ❖ 현재 결과에서 보완했으면 하는 점

- 비밀번호 변경을 위해 이메일 인증이 되도록 하는 기능을 추가.
- 일정 생성시 지도에서 검색하여 추천하는 장소를 알려주는 기능을 추가.

# 프로젝트를 마치며...

**최해혁:** 이번 팀 프로젝트는 전체적으로 팀원 간의 업무 분담과 프로젝트 플로우 관리가 잘 진행되었음을 체감하였습니다.

그러나, 사용자 인증 및 로그인 상태 관리가 예상보다 어려운 도전이었습니다.

초기에는 이 부분이 큰 어려움이었으나, recoil과 atom을 활용하여 이 문제를 효과적으로 해결할 수 있었습니다. 결국, 이 프로젝트를 통해 사용자 인증과 상태 관리에 대한 극복 가능한 해결책을 배우게 되었습니다.:

**이현수:** 팀 프로젝트를 시작하면서 처음엔 막연히 ‘지금껏 배웠던 내용을 잘 활용을 할 수 있을까’ 하는 불안감을 가지고 있었습니다. 프로젝트 1~2주 간은 맡은 기능에 대해서 구현하는 방법에 대해 고민도 하고 도움도 받으면서 개발하는데 속도가 더딘 감이 있었습니다. 그리고 혼자서 하는 프로젝트가 아닌 팀 프로젝트이기에 프로그램에 한 부분만 동작하면 끝이 아니고, 서로 유기적으로 연결되어 있어서 협력과 의사소통에 대해서 좀 더 생각을 하게 되는 동기를 가지게 되었습니다.

끝으로 점차 하나씩 생겨나는 기능과 오류 등을 해결해 나가면서 성취감을 느낄 수 있었고, 점차 자신감도 가질 수 있는 동기가 되었습니다.

**손민재:** 팀 프로젝트에 참여하는 건 값진 경험이었지만 저 스스로 개선이 필요한 부분도 많이 알 수 있는 시간이었습니다.

프로젝트를 진행하면서 제가 부족한 부분들을 알고 공부할 수 있었고 팀원과의 협업에 대해서도 배울 수 있었습니다. 또한 프로젝트를 진행하면서 겪은 문제들을 더 유연하게 대처할 수 있는 법을 알게 된 것이 가장 큰 소득이라고 생각합니다.

**임나영:**



감사합니다

