



고병진, 김정수, 임예은, 최은호(PL)

3WORK : WWW HEALTH CARE

WELBING LIFE WITH EVERY WORK OUT

Contents

01

주제

02

개발 환경

03

역할 분담

04

ERD
Wire-Frame

05

프로젝트 수행 일정

06

UI 디자인

07

웹 사이트 시현

08

프로젝트를 마치며

Topic

3WORK:WWW (Wellbeing Life With Every Workout)

공동체 참여와 업무 수행, 그리고 정기적인 운동이 조화롭게 결합되어
현대인의 건강한 라이프스타일을 촉진하고자 한다.

Wellbeing Life

즐겁고 건강한 삶을 위해
운동과 올바른 식생활에
관련된 차별화된 솔루션 제공

With

공동체 의식을 바탕으로
'함께' 하는 문화를 추구

Every Workout

효과적이고 올바른
운동 방식으로 바쁜 삶 속
맞춤형 트레이닝 방식을 제공
하여 건강생활에 기여

Development Environment

- OS

Windows 10 | Mac OS

- Front-End

HTML5 | CSS | JavaScript ES5 | JQuery 3.7.1 | Thymeleaf

- Back-End

MySQL | JAVA 17 | Spring Boot | JPA

- Tool

MySQL Workbench | SpringToolSuite4 | Figma | Git



Character



최은호 Project Leader

프로젝트 총괄 디렉터 & 팀 회의 주최
주소 API, 날씨 API 연동
회원가입 페이지 DB 작업 진행 & ERD 제작



김정수 Team Member

UI 디자인 & 페이지 별 DB 작업 & 캘린더 API 연동
유저 홈페이지, 프로필 수정 페이지 제작 (HTML, CSS)
회원가입 유효성 검사 진행



고병진 Team Member

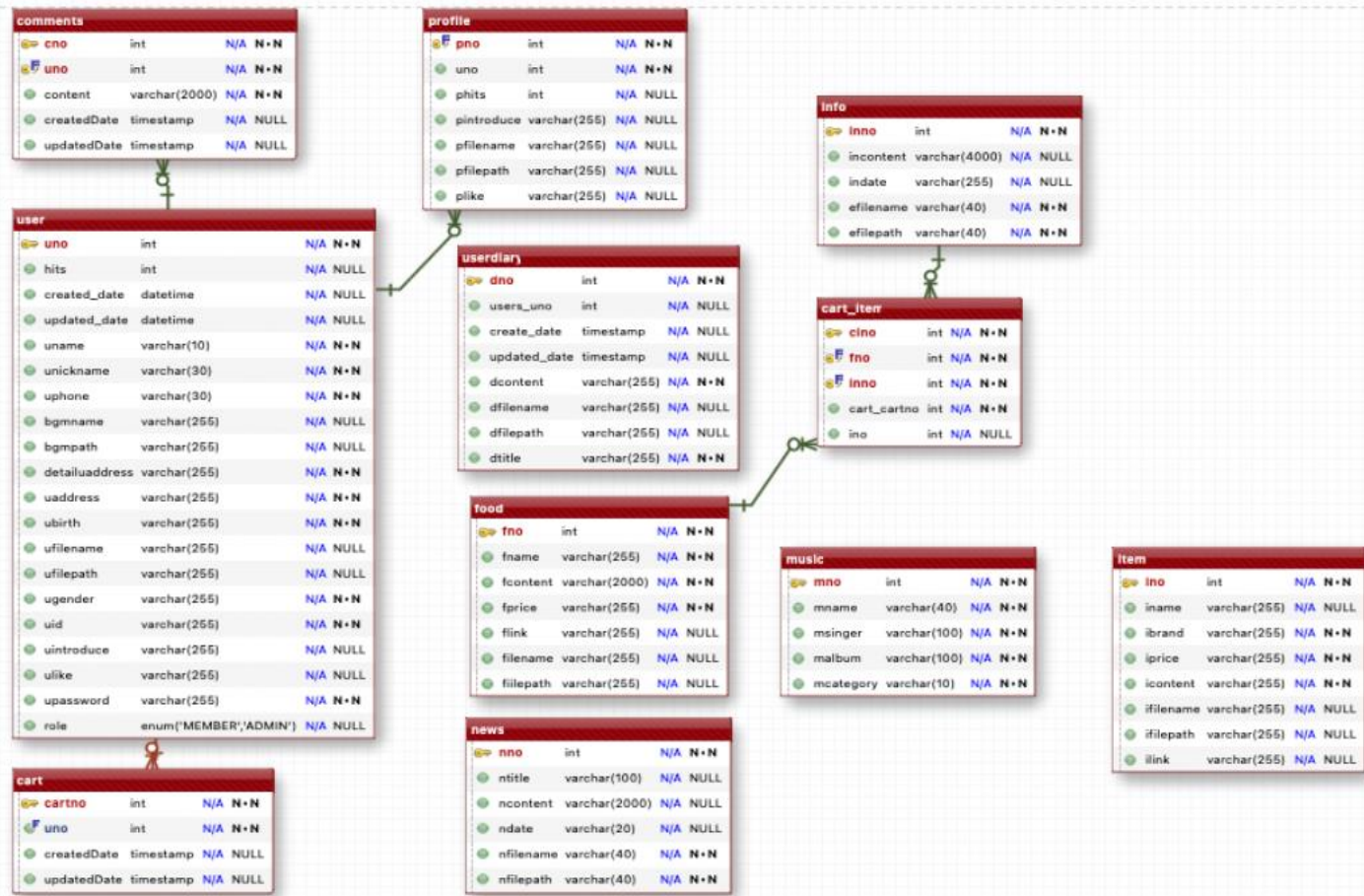
지도 API 연동, 장바구니 & 운동 DB 작업 진행
노래, 뉴스, 아이템 등 페이지 제작 (HTML, CSS)
서브메뉴 페이지 제작 (HTML, CSS)



임예은 Team Member

프로젝트 계획서, 회의록 작성 및 발표자료 제작
대문, 메인, 로그인 등 페이지 제작 (HTML, CSS)
음악, 뉴스, 아이템, 장바구니 등 DB 작업 진행

ERD



[T] user : 회원
[T] comments : 댓글
[T] profile : 프로필
[T] userdiary : 건강일지
[T] music : 노래
[T] food : 음식
[T] item : 아이템
[T] news : 뉴스
[T] info : 정보
[T] cart_item : 장바구니 상품
[T] cart : 장바구니

11개

Wire-Frame



3WORK_메인

아보카도 아이콘		Header	
검색 창			Body
슬라이드 이미지	로그인 / 로그아웃		
	날씨 정보		
3WORK 추천 메뉴			
3WORK 정보 메뉴			
기타 메뉴			Footer

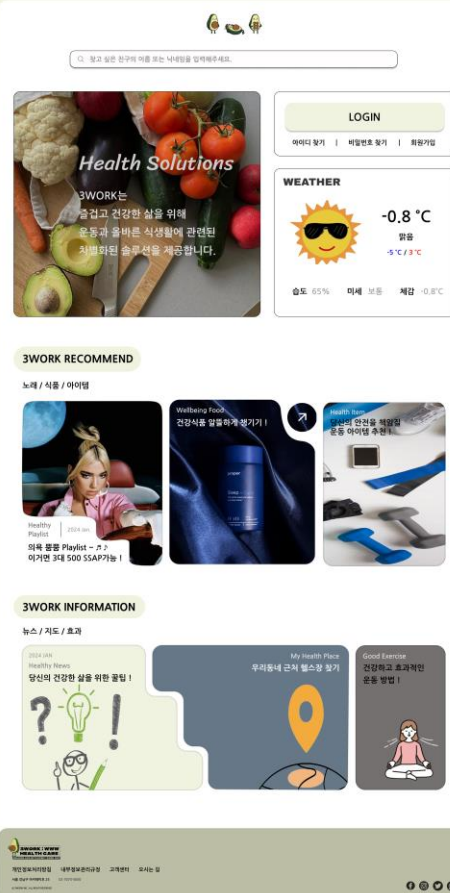
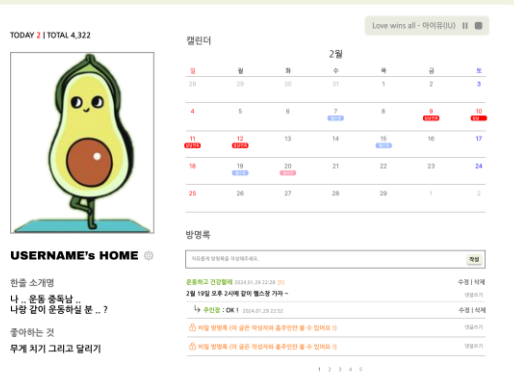
3WORK_유저_홈

아보카도 아이콘		Header
방문객 수	배경음악	Body
프로필 이미지	구글 캘린더	
프로필 정보	방명록	

Schedule

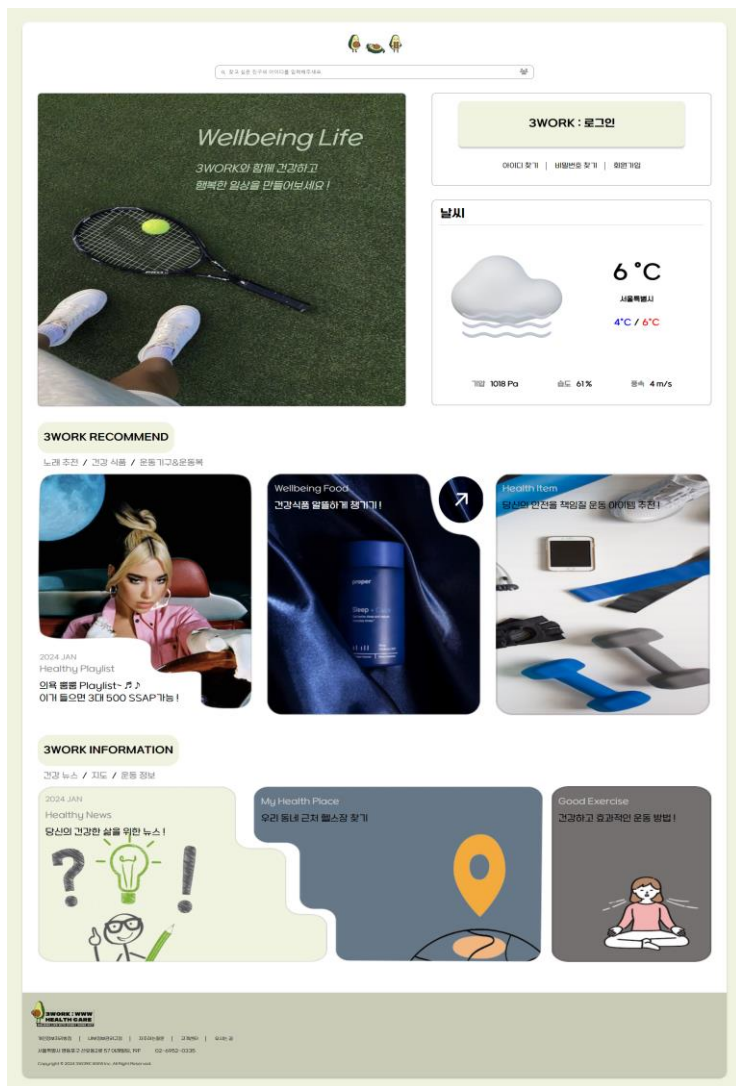
	1주차					2주차					3주차					4주차	
	29	30	31	1	2	5	6	7	8	9	12	13	14	15	16	19	20
주제 선정																	
요구사항 정리																	
업무 분담																	
개발환경 구축																	
UI (디자인) 구현																	
기능 구현																	
기능 테스트																	
발표자료 제작																	

UI Design



주요 기능 구현화면 및 코드

1. 메인페이지



- 날씨 API 구현

```

<div id="weather_box">
  <div id="weather_title">
    <p>
      <!-- <span class="nowtime"></span> -->
      <span>날씨</span>
    </p>
  </div>
  <div id="weather_info">
    <h3 class="SeoulIcon"></h3>
    <div id="weather_info_right">
      <h3 class="SeoulNowtemp"></h3>
      <h3 class="Seoul">서울특별시</h3>
    </div>
    <span class="SeoulLowtemp"></span>
    <span class="tempslice">&nbsp;&nbsp;&nbsp;</span>
    <span class="SeoulHightemp"></span>
    </h3>
  </div>
</div>
<div id="weather_info_bottom">
  <h3 class="pressure"></h3>
  <h3 class="humidity"></h3>
  <h3 class="wind"></h3>
</div>
</div>

```

```
//제이쿼리 사용
$.getJSON('https://api.openweathermap.org/data/2.5/weather?q=seoul&appid=a8501159eb9e48f3bb16a140c20e9c6f&units=metric', function(WeatherResult) {
    // 기온 출력용 위해 Math.round()를 사용하여 온도 값을 정수로 변환함
    var nowTemp = Math.round(WeatherResult.main.temp);
    var lowTemp = Math.round(WeatherResult.main.temp_min);
    var highTemp = Math.round(WeatherResult.main.temp_max);
    var pressure = Math.round(WeatherResult.main.pressure);
    var humidity = Math.round(WeatherResult.main.humidity);
    var wind = Math.round(WeatherResult.wind.speed);

    $('#.SeoulNowtemp').html(nowTemp + ' °C');
    $('#.SeoulLowtemp').html(lowTemp + '°C');
    $('#.SeoulHightemp').html(highTemp + '°C');
    $('#.pressure').html('<span class="texts">기압</span> &nbsp;&nbsp;&nbsp;' + pressure + ' Pa');
    $('#.humidity').html('<span class="texts">습도</span> &nbsp;&nbsp;&nbsp;' + humidity + ' %');
    $('#.wind').html('<span class="texts">풍속</span> &nbsp;&nbsp;&nbsp;' + wind + ' m/s');

// 날씨 아이콘 설정
// 날씨 상황에 따른 아이콘 추가하기
var iconPath = "/images/weather_icon/"; //아이콘 파일경로

var customWeatherIcons = [
    // 맑음
    "01d": iconPath + "clearSky_d.png",
    "01n": iconPath + "clearSky_n.png",
    // 구름 낀 맑음
    "02d": iconPath + "fewClouds_d.png",
    "02n": iconPath + "fewClouds_n.png",
    // 구름이 많은 날(맑은 하늘)
    "03d": iconPath + "scattered_dn.png",
    "03n": iconPath + "scattered_dn.png",
    // 구름이 많은 날(흐림)
    "04d": iconPath + "brokenClouds-dn.png",
    "04n": iconPath + "brokenClouds-dn.png",
    // 소나기가 오는 날
    "09d": iconPath + "showerRain_dn.png",
    "09n": iconPath + "showerRain_dn.png",
    // 비가 오는 날
    "10d": iconPath + "rain_d.png",
    "10n": iconPath + "rain_n.png",
    // 천둥번개 치는 날
    "11d": iconPath + "thunderstorm_dn.png",
    "11n": iconPath + "thunderstorm_dn.png",
    // 눈
    "13d": iconPath + "snow_dn.png",
    "13n": iconPath + "snow_dn.png",
    // 안개
    "50d": iconPath + "mist_dn.png",
    "50n": iconPath + "mist_dn.png",
];

var weatherIconCode = WeatherResult.weather[0].icon;
var weatherIconAlt = WeatherResult.weather[0].description;
var weatherIconUrl = '';

$('#.SeoulIcon').html(weatherIconUrl);
```

* 그 외 기능

- 비 로그인시 : 로그인, 아이디 및 비밀번호 찾기 버튼 출력
- 로그인시 : 미니홈페이지로 이동, 위시리스트, 회원정보 수정, 로그아웃 버튼 출력
- 회원 리스트 및 검색 기능 구현

주요 기능 구현화면 및 코드

2. 아이디 찾기



```
// 아이디 찾기 페이지
@GetMapping("/user/id_search")
public String idSearchForm() {
    return "user/id_search"; // 아이디 찾기 폼 페이지 반환
}


//아이디 찾기 (이름과 전화번호)
@PostMapping("/user/id_search")
public String idSearch(@RequestParam("uname") String name, @RequestParam("uphone") String phone, Model model) {
    Optional<Users> usersOptional = usersRepository.findByNameAndUphone(name, phone);
    if(usersOptional.isPresent()) {
        Users users = usersOptional.get();
        model.addAttribute("uid", users.getId());
        model.addAttribute("joinDate", users.getCreatedAt()); // 가입 날짜, BaseEntity에서 상속받은 createdAt 사용
        return "user/id_result"; // 결과 페이지로 이동
    } else {
        model.addAttribute("message", "일치하는 사용자 정보가 없습니다.");
        return "user/id_search"; // 정보가 없는 경우 다시 아이디 찾기 페이지로 이동
    }
}
```

- <Optional>을 활용하여 이름과 연락처 DB에서 검색된 아이디, 가입일 출력

- 일치하는 아이디가 있는 경우 Model을 활용하여 출력화면 으로 보냄

주요 기능 구현화면 및 코드

3. 비밀번호 찾기(임시 비밀번호 발급)



임시 비밀번호 발급

임시 비밀번호는 가입 시 입력한 이메일, 이름, 전화번호를 통해 발급이 가능하며, 로그인 후 수정해주세요.

임시 비밀번호 발송

```
// 비밀번호 찾기 요청 페이지
@GetMapping("/user/resetpassword")
public String pwSearchForm() {
    return "user/resetpassword";
}

@PostMapping("/user/resetpassword")
public String resetPassword(@ModelAttribute UsersDTO usersDTO, Model model) {
    boolean isMatch = userService.isUidAndUphoneAndUnameMatch(usersDTO.getUid(), usersDTO.getUname(), usersDTO.getUphone());
    if(isMatch) {
        String temporaryPassword = userService.generateTemporaryPassword();
        emailService.sendTemporaryPassword(usersDTO.getUid(), temporaryPassword);
        // 임시비밀번호를 db에 저장
        UsersDTO usersDT02 = userService.findByUid2(usersDTO.getUid());
        usersDT02.setUpassword(temporaryPassword);
        userService.update(usersDT02);
        return "user/resetpassword2";
    } else {
        model.addAttribute("error", "일치하는 정보가 없습니다.");
        return "user/resetpassword";
    }
}
```

임시 비밀번호가 메일에 전송되었습니다.

발급된 비밀번호로 로그인이 가능하며, 로그인 후 수정해주세요.

로그인

☆ 임시 비밀번호 발급 안내

보낸사람 foreunho@gmail.com

받는사람 yddk920@naver.com

2024년 3월 7일 (목) 오후 1:04

임시 비밀번호 :UpZtIEHEtO

```
//임시 비밀번호 발송
public boolean isUidAndUphoneAndUnameMatch(String uid, String uname, String uphone) {
    Users users = usersRepository.findByUidAndUnameAndUphone(uid, uname, uphone);
    return users != null;
}

//임시 비밀번호 생성, 이메일 전송
public void sendTemporaryPassword(String uid){
    String temporaryPassword = generateTemporaryPassword();
    emailService.sendTemporaryPassword(uid, temporaryPassword);
}

public String generateTemporaryPassword(){
    String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    int length = 10;
    StringBuilder temporaryPassword = new StringBuilder(length);
    SecureRandom random = new SecureRandom();
    for(int i=0; i<length; i++){
        int randomIndex = random.nextInt(characters.length());
        temporaryPassword.append(characters.charAt(randomIndex));
    }
    return temporaryPassword.toString();
}

//임시비밀번호 저장
public UsersDTO findByUid2(String uid) {
    Users users = usersRepository.findByUid(uid).get();
    UsersDTO usersDTO = UsersDTO.toSaveDTO(users);
    return usersDTO;
}
```

- 아이디, 이름, 연락처로 일치하는 회원 확인 후 isMatch가 true인 경우 임시 비밀번호를 생성하여 회원의 이메일로 전송

- 발급된 임시 비밀번호를 데이터에 저장하여 비밀번호 업데이트 및 로그인 가능하도록 구현

주요 기능 구현화면 및 코드

4. 회원 가입(JS를 활용한 유효성 검사)

Daum Postcode Service - Chrome

about:blank

강남구

검색결과가 많습니다. 검색어에 아래와 같은 조합을 이용하시면 더욱 정확한 결과가 검색됩니다. '도로명+건물번호', '지역명+지번', '지역명+건물명(아파트명)', '사서화명+번호'

도로명 전체

지역명 전체

06035

영문보기

지도

도로명

서울 강남구 가로수길 5

지번

서울 강남구 신사동 537-5

외 주변주소 1건

더보기

06035

영문보기

지도

도로명

서울 강남구 가로수길 9

지번

서울 강남구 신사동 536-9

06036

영문보기

지도

도로명

서울 강남구 가로수길 10

회원가입

아이디

yddk9202@naver.com

중복확인

사용 가능한 이메일입니다.

비밀번호

.....

비밀번호 확인

.....

비밀번호가 일치합니다.

이름

테스트

닉네임

테스트

중복확인

사용 가능한 닉네임입니다.

성별

남

주소

[06035] 서울 강남구 가로수길 5 (신사동)

우편번호 찾기

B동 102호

연락처

010-2321

생년월일

2024-03-01

☐ 이용약관과 개인정보 수집 및 활용에 동의합니다.

가입

```
//아이디 중복 검사
const checkUserIdOnClick = () => {
  let inputId = document.getElementById("uid").value;
  let checkResult = document.getElementById("result");

  let header = $("meta[name='_csrf_header']").attr('content');
  let token = $("meta[name='_csrf']").attr('content');

  if(uid != "") {
    $.ajax({
      type: "POST",
      beforeSend: function(xhr){
        xhr.setRequestHeader(header, token);
      },
      url: "/user/check-email",
      data: {uid: inputId},
      success: function(res) {

        if (res == "OK") {
          checkResult.innerHTML = "사용 가능한 이메일입니다."
          checkResult.style.color = "blue"
          checkResult.style.fontFamily = "SBAggroM";
        } else {
          checkResult.innerHTML = "이미 등록된 이메일입니다."
          checkResult.style.color = "red"
          checkResult.style.fontFamily = "SBAggroM";
        }
        checkResult.style.fontSize = "0.9em";
      },
      error: function(err) {
        console.log("실패: ", err);
      }
    });
  } else {
    checkResult.innerHTML = "";
  }
};
```

```
//비밀번호 유효성 검사
function validatePassword() {
  let passwordInput = document.getElementById("upassword");
  let checkMessageDiv = document.getElementById("pucheck-message");
  let passwordError = checkMessageDiv.querySelector("p"); // 'check-message' div 안의 p 태그 선택

  let passwordRegex = /^(?=.*[a-zA-Z])(?=.*[!@#$%^&*()_~?{}|;':"<>])(?=.*[0-9])(?=.*[0-9]).{8,20}$/;
  if (!passwordRegex.test(passwordInput.value)) {
    passwordError.innerHTML = "특수문자, 대소문자를 포함한 8~20자여야 합니다.";
    passwordError.style.color = "blue";
    passwordError.style.fontFamily = "SBAggroM";
    passwordError.style.fontSize = "0.9em";
  } else {
    passwordError.innerHTML = "";
  }
};

//비밀번호 확인
function checkPasswordMatch() {
  var password = document.getElementById("upassword").value;
  var confirmPassword = document.getElementById("confirmPassword").value;
  var message = document.getElementById("puconfirmResult");

  if (password === confirmPassword) {
    message.innerHTML = "비밀번호가 일치합니다.";
    message.style.color = "blue";
  } else {
    message.innerHTML = "비밀번호가 일치하지 않습니다.";
    message.style.color = "red";
  }
  message.style.fontSize = "0.9em";
  message.style.fontFamily = "SBAggroM";
}
```

- ajax를 활용한 아이디, 닉네임 중복검사 진행

- 비밀번호 일치 여부 및 각 입력 데이터 유효성 검사 진행

- 우편번호 및 주소 찾기 API 구현

주요 기능 구현화면 및 코드

4-1. 회원 가입(DTO를 활용한 유효성 검사)

localhost:8080 내용:
이용약관과 개인정보 수집 및 활용에 동의해야 합니다.

확인

아이디	yddk9202@naver.com	중복확인
사용 가능한 이메일입니다.		
비밀번호	*****	
비밀번호 확인	*****	
비밀번호가 일치합니다.		
이름	홍길동	
닉네임	테스트	중복확인
사용 가능한 닉네임입니다.		
성별	남	
주소	[06035] 서울 강남구 가트수길 5 (신사동)	우편번호 찾기
	B동 102호	
연락처	연락처 (- 포함 13자리입력)	
핸드폰입력은 필수 항목입니다.		
생년월일	연도-월-일	
생년월일은 필수 항목입니다.		
<input type="checkbox"/> 이용약관과 개인정보 수집 및 활용에 동의합니다.		
가입		

```
@Builder
@Data
public class UsersDTO {

    private Integer uno;

    //아이디는 4자~20자로 입력
    @NotEmpty(message = "이메일(아이디)은 필수 항목입니다.")
    @Pattern(regexp = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$", message = "")
    private String uid;

    @NotEmpty(message = "비밀번호는 필수 항목입니다.")
    @Pattern(regexp = "(?=.*[a-zA-Z])(?=.*[!@#$%^&*()_~:|{}<>])(?=.*[0-9]).{8,20}$", message = "")
    private String upassword;

    @NotEmpty(message = "이름은 필수 항목입니다.")
    @Pattern(regexp = "^[가-힣]*$", message = "")
    private String uname;

    @NotEmpty(message = "닉네임은 필수 항목입니다.")
    private String unickname;

    @NotEmpty(message = "성별은 필수 항목입니다.")
    private String ugender;

    @NotEmpty(message = "주소는 필수 항목입니다.")
    private String uaddress;

    @NotEmpty(message = "상세주소는 필수 항목입니다.")
    private String detailuaddress;

    @NotEmpty(message = "핸드폰입력은 필수 항목입니다.")
    @Pattern(regexp = "^[0-9]{3}-[0-9]{4}-[0-9]{4}$", message = "")
    private String uphone;

    @NotEmpty(message = "생년월일은 필수 항목입니다.")
    private String ubirth;
}
```

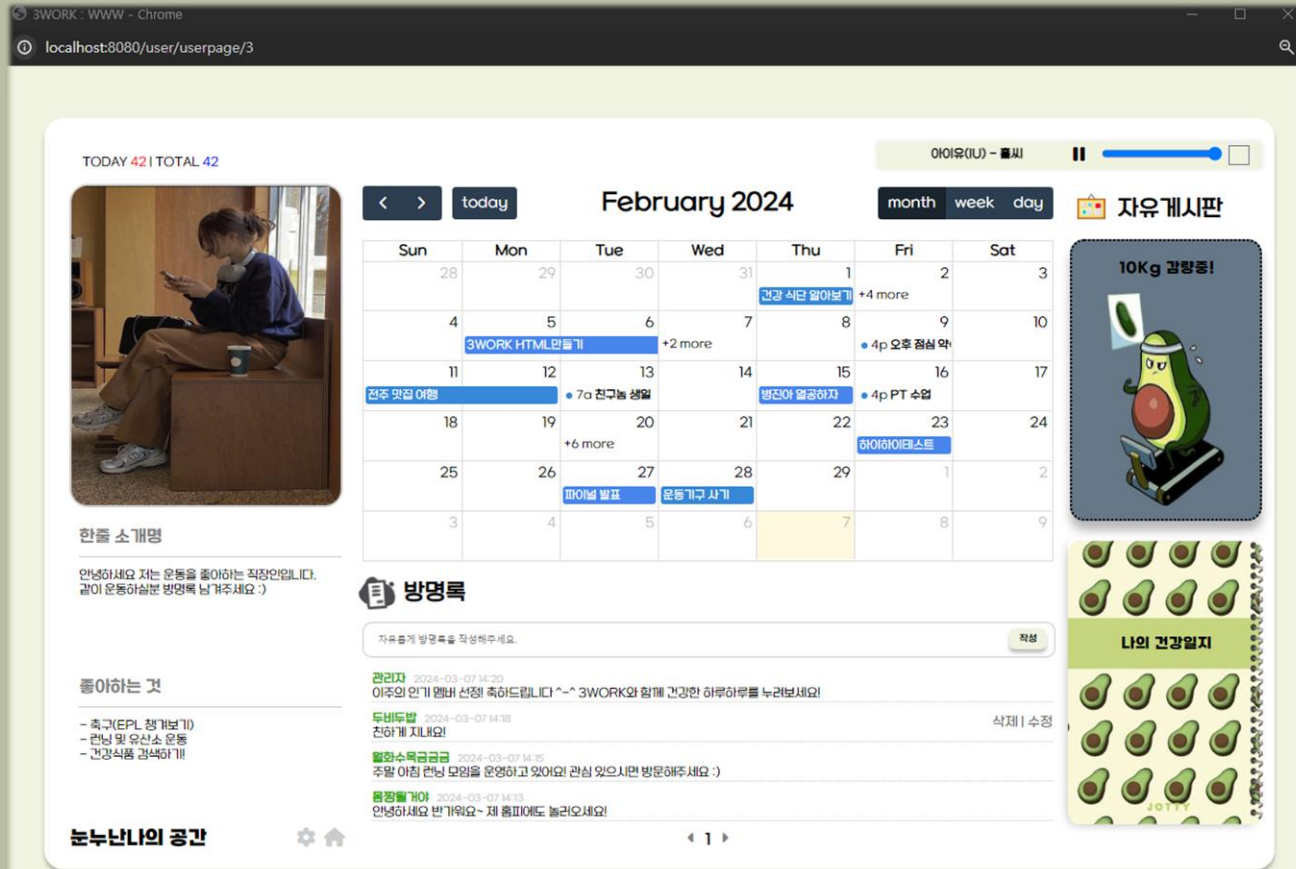
```
// 회원 가입 처리
// @Valid : 필드의 유효성 검사
// BindingResult: 에러 처리 클래스
@PostMapping("/user/join")
public String join(@Valid UsersDTO usersDTO, BindingResult bindingResult) {
    if (bindingResult.hasErrors()) {
        // 에러가 있으면 회원 가입 페이지에 머무름
        return "user/join";
    }
    //방문자수 0으로 설정
    usersDTO.setHits(0);
    userService.save(usersDTO);
    return "user/join_wel";
}
```

- 입력하지 않은 항목이 있을시 DTO의 설정한 유효성 검사 메시지가 출력되도록 구현

- @valid와 BindingResult기능을 활용하여 에러메시지 출력

주요 기능 구현화면 및 코드

5. 유저 홈페이지(팝업 형태로 구현)



1. 홈페이지 방문시 방문자수 증가
2. 프로필(이미지, 소개글) 구현
3. 회원 정보 수정 가능(첨부파일 저장기능 포함)
4. BGM 재생 기능 구현
5. 방명록 작성, 수정, 삭제 기능 구현
6. 방명록 삭제 및 수정은 회원당 자신이 작성한 방명록만 가능하도록 구현
7. 방명록 페이지 처리 구현
8. 게시판 '나의 건강일지' 구현
- 게시글의 CRUD 구현
9. 'FullCalendar' 라이브러리를 활용하여 구글 API를 포함하여 캘린더 구현

주요 기능 구현화면 및 코드

5-1. 유저 홈페이지(게시판) : 나의 건강일지

나의 건강일지

※ 당신의 건강계획을 위한 자유로운 게시판

의지로 바꿀 수 있는 게 내 몸뿐임을 알게 되었다.

아이의 방학 그리고 개학까지 거의 세 달 만에 만난 친구의 눈에 놀라움이 가득. 12월에 만났을 때 이뻐서 아찔한 했지..

2024-03-07 15:08



이젠 더 이상 물러설 수 없다. 계단운동 1주차

가는 날이 장날이다 정말 운동을 해야겠다고 마음먹고 운동을 시작하고 보니 며칠 뒤면 설날이다.

2024-03-07 15:07



< 1 >

글쓰기 돌아가기

- CRUD를 포함한 게시판 구현

- Controller, Service, Repository를 활용한 MVC
패턴 적용

```
//유저 건강일지 페이지
@GetMapping("/user/userdiary/{uno}")
public String userDiary(@PathVariable Integer uno,
    Model model,
    @PageableDefault(page = 1) Pageable pageable) {
    Page<UserDiaryDTO> diaryList = diaryService.findAll(pageable);

    //하단에 페이지 영역 만들기
    int blockLimit = 10; //하단에 보여줄 페이지 개수
    //시작페이지
    int startPage = ((int)(Math.ceil(((double)pageable.getPageNumber() / blockLimit)) - 1)* blockLimit + 1;
    //마지막 페이지
    int endPage = (startPage + blockLimit - 1) > diaryList.getTotalPages() ?
        diaryList.getTotalPages() : startPage + blockLimit - 1;

    model.addAttribute("uno", uno);
    model.addAttribute("diary", diaryList);
    model.addAttribute("startPage", startPage);
    model.addAttribute("endPage", endPage);
    return "user/userdiary";
}

//유저 건강일지 글쓰기페이지
@GetMapping("/user/diarywrite")
public String writeDiary(@AuthenticationPrincipal SecurityUser principal,
    Model model) {
    Integer uno = principal.getUsers().getUno();
    model.addAttribute("uno", uno);
    return "user/diarywrite";
}

//글쓰기
@PostMapping("/user/diarywrite/{uno}")
public String write(@ModelAttribute UserDiaryDTO userDiaryDTO,
    @PathVariable Integer uno,
    MultipartFile boardFile, Model model,
    @AuthenticationPrincipal SecurityUser principal) throws Exception {
    userDiaryDTO.setUsers(principal.getUsers());
    diaryService.save(userDiaryDTO, boardFile);
    model.addAttribute("diary", boardFile);
    return "redirect:/user/userdiary/" + uno;
}

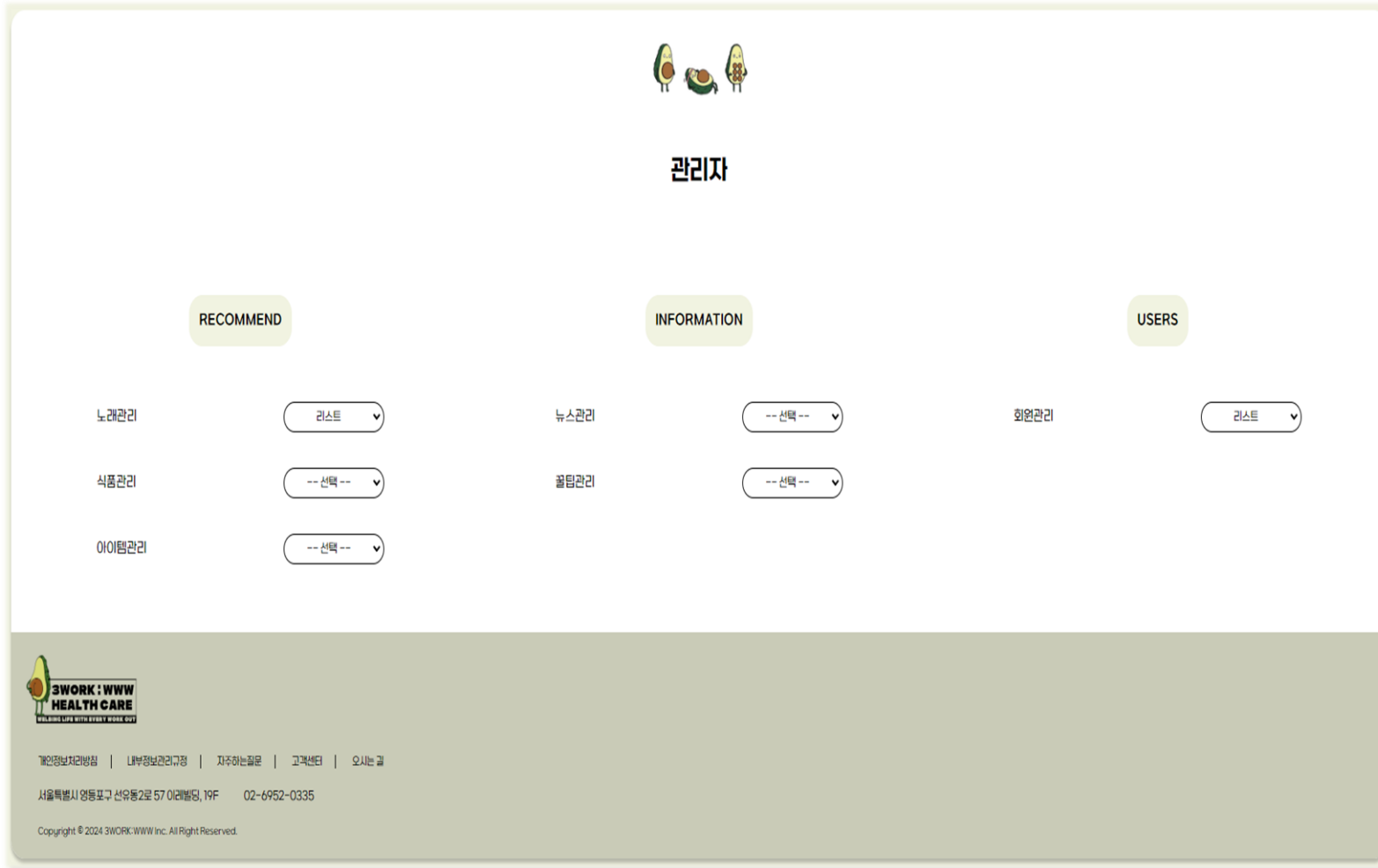
//글 삭제
@GetMapping("/delete/{dno}")
public String delete(@PathVariable Integer dno) {
    UserDiary userDiary = diaryService.findById(dno);
    diaryService.deleteByDno(dno);
    return "redirect:/user/userdiary/" + userDiary.getUsers().getUno();
}

//글 상세보기
@GetMapping("/userdiary/{dno}")
public String getDiary(@PathVariable Integer dno,
    Model model) {
    UserDiary diaryDetail = diaryService.findById(dno);
    model.addAttribute("diaryDetail", diaryDetail);
    model.addAttribute("dno", dno);
    return "user/diarydetail";
}

//글 수정페이지
@GetMapping("/user/diaryupdate/{dno}")
public String diaryUpdate(@PathVariable Integer dno,
    Model model) {
    //해당 아이디의 수정할 게시글 가져오기
    UserDiary userDiary = diaryService.findById(dno);
    model.addAttribute("diary", userDiary);
    model.addAttribute("dno", dno);
    return "user/diaryupdate";
}
```


주요 기능 구현화면 및 코드

6. 관리자 페이지



- 회원이 해당 페이지 접근시 "접근 불가 페이지입니다." 문구 출력 및 메인으로 이동하도록 구현
- 각종 콘텐츠 페이지 (음악추천, 건강 뉴스 및 정보) 데이터 저장/삭제 가능
- 각 콘텐츠 화면 내 페이지징처리 구현
- 회원 리스트 및 회원 정보 상세 보기, 탈퇴 기능 구현

활용한 API

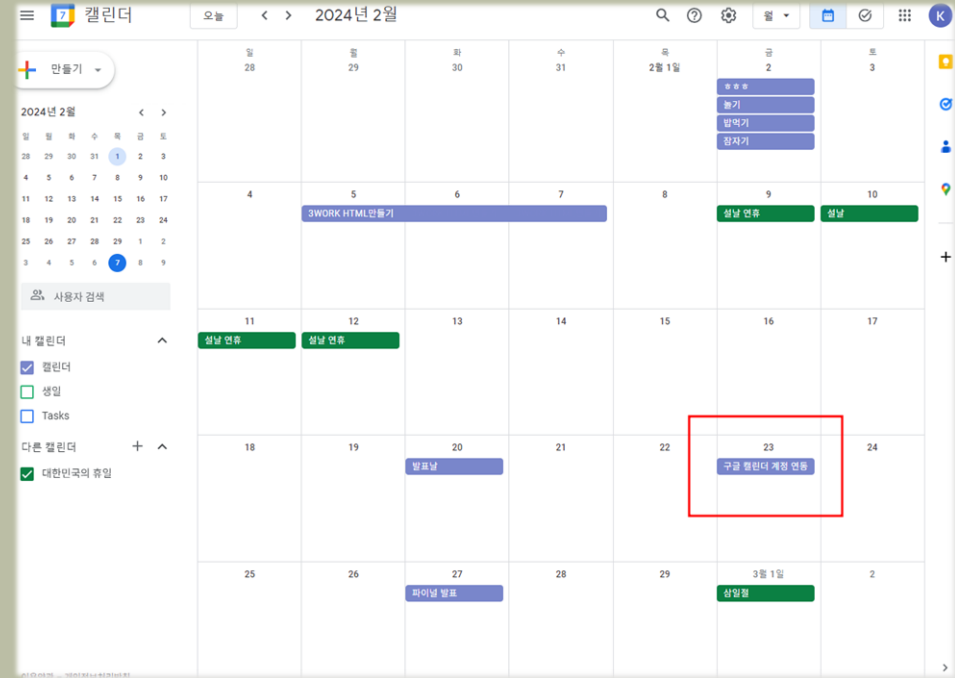
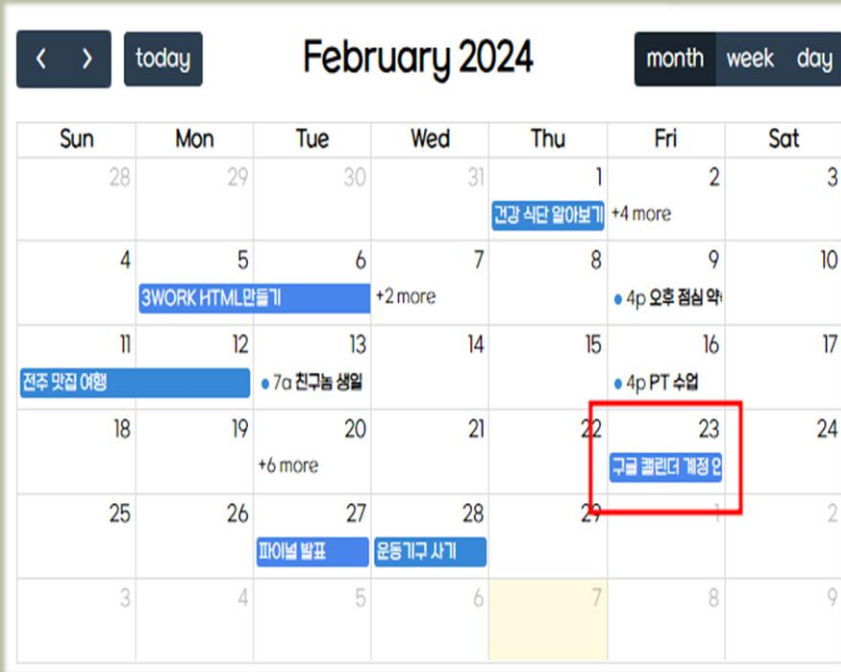
1. 카카오 지도

근처 운동할 수 있는 곳 찾기



활용한 API

2. FullCalendar 라이브러리를 활용한 달력 구현(구글 API로 계정 연동 가능)



```
//Google Calendar 이벤트 소스
eventSources: [
  {
    googleCalendarApiKey: 'AIzaSyA...',
    googleCalendarId: 'yddk866@gmail.com',
    className: '캘린더',
    color: '#4885ed', // 이벤트 색상 (옵션)
    textColor: '#ffffff' // 텍스트 색상 (옵션)
  }
]
```

- 구글 API로 계정을 연동하여 구글 캘린더에서 저장한 일정이 FullCalendar에도 표시되도록 구현

Implementation Failed(보완 진행중)

노래 추천

- 카테고리(요가, 수영, 헬스) 선택 시, 페이지 처리 실패

회원 검색

- 회원 검색 시, 순위 고정으로 인한 순위 변동 불가
- 페이지 처리 실패

회원 미니홈피

- 총 방문자 수 구현 실패
- 페이지 새로고침 시, 방문자 수 증가 해결 실패
- 방명록 작성 시, 회원 별 방명록 목록 불러오기 실패
- 회원 별 구글 캘린더 연동 실패
- 방명록 작성 시, 대댓글 작성 불가
- 회원 수정 시, 비밀번호 변경 필수 (안했을 시, 비밀번호가 암호화되어 다시 저장됨)

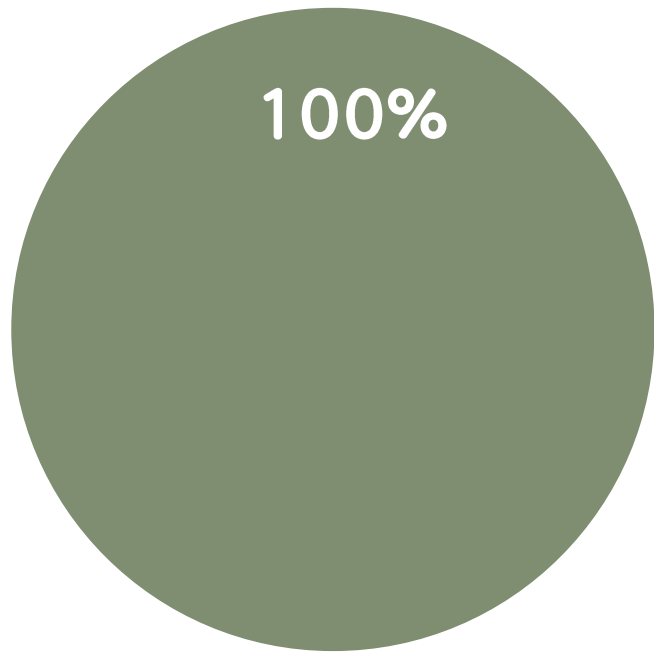
아이디 찾기

- 핸드폰 번호가 겹치는 회원 존재 시, 아이디 찾기 불가

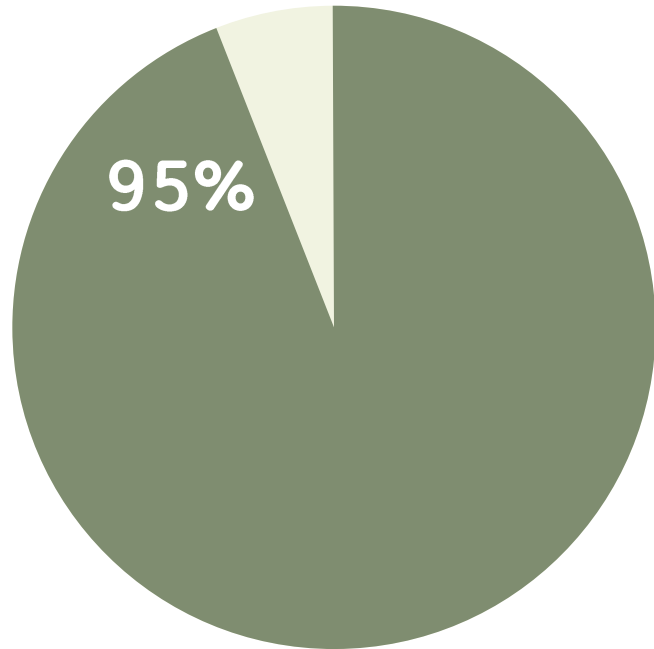
메인

- SNS 계정 연동해 로그인 불가
- CSS 반응형 일부 미흡

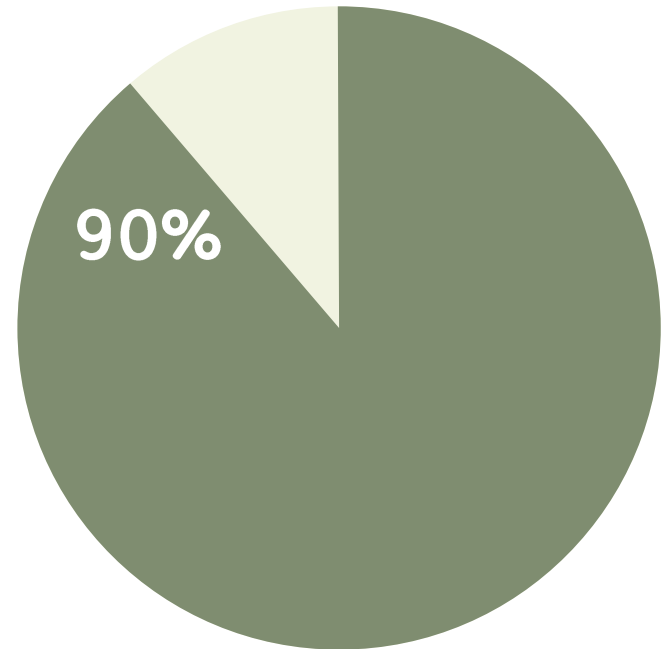
Summary



ERD & Wire-Frame, UI 디자인



기능 구현 (HTML, CSS)



기능 구현 (DB, API)

Thank You ~ ☺