

식물추천 홈페이지 개발

팀 명 : 한 번 해보조
지도 교수 : 양환석 교수님
팀 장 : 손현지
팀 원 : 변유빈
유은지
김민우

2023. 11.

중부대학교 정보보호학과

목 차

1. 서 론	
1.1 연구 배경	3
1.2 연구 목적 및 주제선정	3
2. 본 론	
2.1 FRONTEND 구성	4
2.1.1 식물추천 웹페이지	4
2.1.2 로그인/회원가입 기능 페이지	5
2.1.3 추천 시스템 페이지	11
2.1.4 식물 상점 페이지	13
2.1.5 팀원 소개 페이지	15
2.2 BACKEND 구성	16
2.2.1 MySQL DB 구축	16
2.2.2 추천 시스템 구축	17
2.2.3 식물 상점 구축	20
2.3 배포	22
2.3.1 FRONTEND 배포	22
2.3.2 BACKEND 배포	23
3. 결 론	
3.1 결 론	24
3.2 기대 효과	24
4. 별 첨	
4.1 참고 문헌	25
4.2 팀원 소개	25
4.3 소스 코드	25
4.4 발표 자료	26
4.5 소개 자료	32

1. 서론

1.1 연구배경

현대 사회에서는 도시화와 고밀도 주거 공간 증가로 인해 사람들은 작은 공간에서 생활하고 있다. 이러한 환경에서 사람들은 다양한 스트레스를 받으며 살아간다. 각종 스트레스에 시달리는 사람들이 식물을 키움으로써 스트레스를 완화하는 데 도움을 받을 수 있다. 『실내조경에 의한 스트레스 해소효과』라는 논문을 통해 ‘정서순화를 위해서 환경조건이나 상황설정이 매우 중요한 역할을 하는데, 실내 식물을 도입함으로써 정서 순화 환경이 유도된다’는 것을 알 수 있다. 따라서 실내 식물을 키우는 것은 실내 공기를 정화하는 데 도움을 줄 뿐만 아니라 식물을 관리하는 과정에서 스트레스를 감소시키는 데 도움이 된다는 점을 알 수 있다. 또한, 일상에서 자연과의 연결감을 높여줄 수 있다. 이때, 어떤 식물을 선택하고 어떻게 키우는지에 대한 정보를 얻는 것은 식물을 키우는 데 중요하다고 할 수 있다. 그러나 많은 사람은 어떤 식물을 선택해야 할지, 그리고 그 식물을 어떻게 키워야 하는지에 대한 적절한 지침을 얻기 어렵다고 느낄 수 있다. 본 연구에서는 이러한 문제를 해결하기 위해 실생활에서 키울 수 있는 식물을 추천하고, 해당 식물을 구매할 수 있는 온라인 사이트를 추천하는 서비스를 제공한다. 이를 통해 사람들은 자신의 생활환경에 맞는 식물을 쉽게 선택하고, 키우는 방법을 습득할 수 있다. 또한, 온라인으로 간편하게 해당 식물을 구매할 수 있는 경로를 제공함으로써 더 나은 식물 관리와 관심 있는 사람들 간의 유용한 상호작용을 촉진 시킬 것이다.

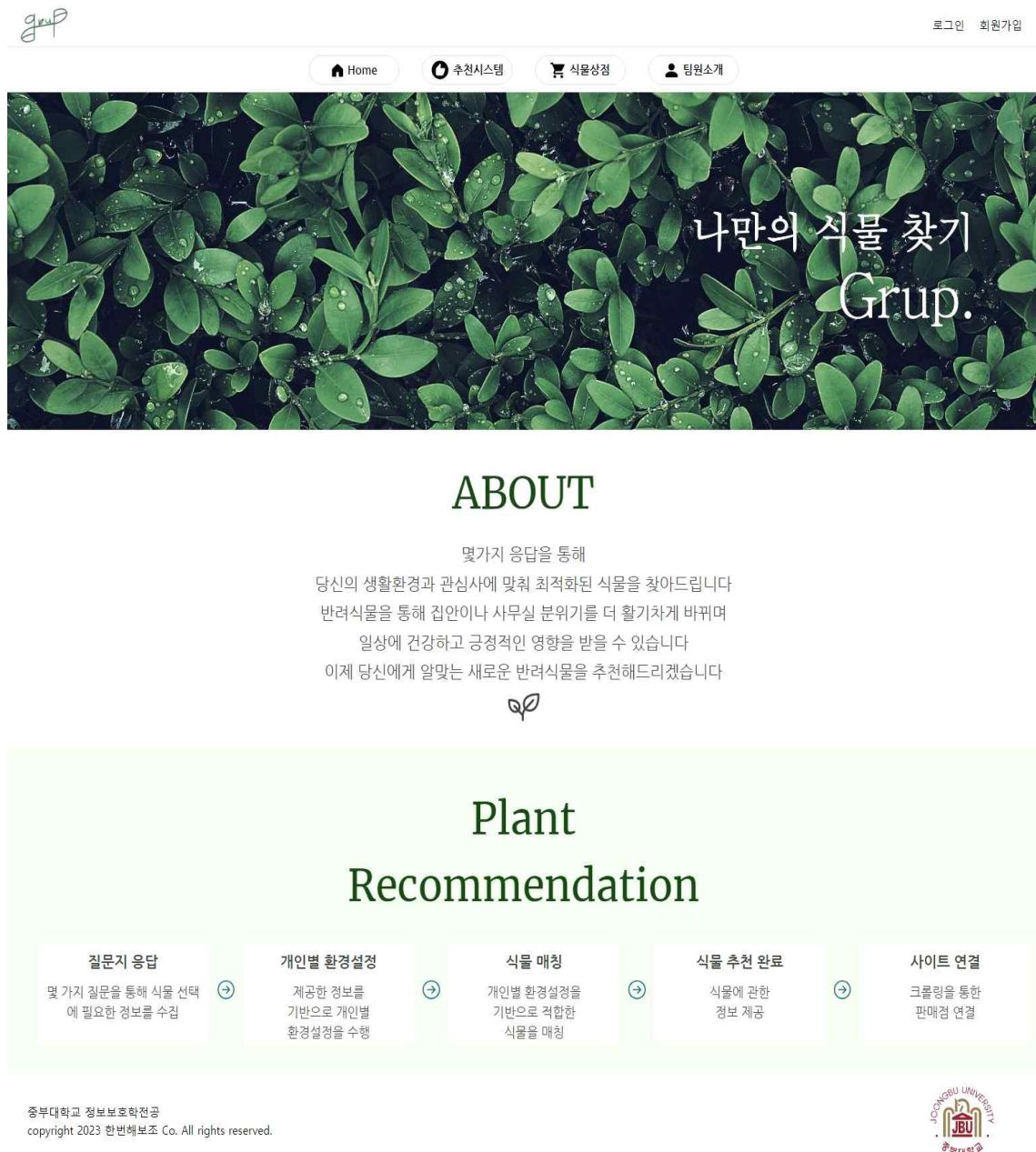
1.2 연구 목적 및 주제선정

요즘 일상 속 나만의 작은 정원을 갖고 싶어 하는 사람이 늘어나고 있다. 여행을 떠나는 대신 집에서 식물을 키움으로써 스트레스를 해소하고자 하는 것이다. 이를 위해 자신에게 어울리는 식물을 찾거나 원하는 식물을 어디서 구매할 수 있는지 찾아보는 것이 여간 번거로운 일이 아니다. 또한, 식물에 대한 지식이 전혀 없다면 어떤 식물을 키워야 하는지에 대해서도 잘 모르는 것이 당연하다. 우리 집의 환경은 어떤지, 내가 식물에 맞춰줄 수 있는 정도는 어느 정도인지 등 자신의 정보를 토대로 이른바 ‘퍼스널 식물’을 찾을 수 있게 하고자 하였고, 그와 동시에 해당 식물을 구매할 수 있는 사이트를 연결함으로써 사용자가 원하는 식물을 보다 편하게 키울 수 있도록 간편성을 높이고자 하였다. 보통의 식물 판매 사이트에는 사용자가 원하는 식물이 무엇인지 추천해 주는 것이 아니라 판매량 순이나 후기가 좋은 순으로 사용자에게 추천하고 있으며 사용자가 어느 수준의 지식을 가졌는지, 어떤 환경을 가졌는지 알지 못하고 추천하기 때문에 처음 식물에 관심을 갖게 된 사용자가 난처할 수 있으므로 ‘퍼스널 식물 찾기’ 및 ‘식물 상점 검색’을 융합한 사이트로 주제를 정하게 되었다.

2. 본 론

2.1 FRONTEND 구성

2.1.1 식물추천 웹페이지



[그림1] 메인페이지

[메인페이지 주 기능]

1. 로그인/회원가입 페이지
2. 상단 메뉴바
3. 웹페이지 소개

[상단 메뉴바 기능]

1. Home : 메인페이지로 이동
2. 추천 시스템 : 개인의 환경 분석을 통해 식물을 추천해주는 설문페이지로 이동
3. 식물 상점 : 크롤링을 통해 식물을 구매 할 수 있는 상점으로 연결해주는 페이지로 이동
4. 팀원 소개 : 웹페이지를 개발한 개발자의 정보를 보여주는 페이지 이동

※각 페이지 간 이동은 react-router-dom을 사용해 구현하였다.

```
<Link to="/" className="link-style">
  <button className="nav-button">
    <div className="NavIconContainer">
      <GoHomeFill size={22} />
    </div>
    <li>Home</li>
  </button>
</Link>
```

[그림2] react-router-dom 사용예시

2.1.2 로그인/회원가입 기능 페이지

```
export function AuthProvider({ children }) {
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [loggedInEmail, setLoggedInEmail] = useState('');

  useEffect(() => {
    // 로그인 상태 초기화 및 업데이트 로직
    const fetchLoginStatus = async () => {
      try {
        const response = await axios.get('/PlantsPlanet/loginCheck');
        console.log(response);
        if (response.data.includes('@')) {
          setIsLoggedIn(true);
          setLoggedInEmail(response.data);
          console.log("response.data=null");
        } else {
          setIsLoggedIn(false);
          console.log("reponse.data=null");
        }
      } catch (error) {
        console.error(error);
        setIsLoggedIn(false);
      }
    };

    fetchLoginStatus();
  }, []);

  return (
    <AuthContext.Provider value={{ isLoggedIn, setIsLoggedIn, loggedInEmail, setLoggedInEmail }}>
      {children}
    </AuthContext.Provider>
  );
}
```

[그림3] 로그인 상태 초기화 및 업데이트 로직

```
import { useAuth } from "../Header/AuthContext";
```

[그림4] 로그인 상태 초기화 및 업데이트 로직

```
// 로그인 실패시
if (response.data.startsWith("login")) {
  setMemberEmail(memberEmail);
  console.log(response.data);
  setIsLoggedIn(false);
  setLoggedInSuccess(false);
  // openModal()
} else {
  // 로그인 성공
  setMemberEmail(response.data);
  setLoggedInEmail(response.data);
  console.log(response.data);
  setIsLoggedIn(true);
  navigate("/");
}
```

[그림5] 로그인 상태 초기화 및 업데이트 로직

로그인 상태를 AuthContext를 통해 전역변수로 관리하고 로그인/로그아웃 상태에 따라 Header에 바로 반영된다.

로그인 회원가입

[그림6] 로그인하기 전의 헤더

qqq@gmail.com 로그아웃

[그림7] 로그인 된 상태의 헤더

로그인 시 yup 라이브러리를 통해 객체 스키마 유효성 검사를 실행하였다. 이 라이브러리를 통해 간편하게 데이터의 유효성을 확인하고 검증할 수 있다.

```
import { object, string, number, date, InferType } from 'yup';

let userSchema = object({
  name: string().required(),
  age: number().required().positive().integer(),
  email: string().email(),
  website: string().url().nullable(),
  createdOn: date().default(() => new Date()),
});

// parse and assert validity
const user = await userSchema.validate(await fetchUser());

type User = InferType<typeof userSchema>;
/* {
  name: string;
  age: number;
  email?: string | undefined
  website?: string | null | undefined
  createdOn: Date
}*/
```

[그림8] 공식 문서에서 보여주는 yup 예시 코드

출처 : <https://www.npmjs.com/package/yup>

```
const formSchema = yup.object({
  memberEmail: yup
    .string()
    .required("이메일을 입력해주세요")
    .email("이메일 형식이 아닙니다."),
  memberPassword: yup.string(),
});

const {
  register,
  handleSubmit,
  formState: { errors },
} = useForm({
  mode: "onChange",
  resolver: yupResolver(formSchema),
});
```

[그림9] yup 적용 코드

```
<input
  name="memberEmail"
  placeholder="이메일"
  {...register("memberEmail")}
  value={memberEmail}
  onChange={(e) => setMemberEmail(e.target.value)} // 이메일 입력 상태 업데이트
/>
{errors.memberEmail && <p>{errors.memberEmail.message}</p>}
<input
  name="memberPassword"
  type="password"
  placeholder="비밀번호"
  {...register("memberPassword")}
  value={memberPassword}
  onChange={(e) => setMemberPassword(e.target.value)} // 비밀번호 입력 상태 업데이트
/>
{errors.memberPassword && <p>{errors.memberPassword.message}</p>}
```

[그림10] yup 적용 코드



a

이메일 형식이 아닙니다.

...

로그인

회원가입

[그림11] 로그인페이지 yup



asd@asdf.com

...

로그인

회원가입

[그림12] 로그인페이지 yup

지정된 형식이 지켜지지 않으면 오류 메시지와 함께 로그인이 이뤄지지 않는 것을 알 수 있다.

로그인과 마찬가지로 회원가입 시 yup 라이브러리를 통해 객체 스키마 유효성 검사를 실행한다. 이 라이브러리를 통해 간편하게 데이터의 유효성을 확인하고 검증할 수 있다.

```
const formSchema = yup.object({
  memberEmail: yup
    .string()
    .required("이메일을 입력해주세요")
    .email("이메일 형식이 아닙니다."),
  memberPassword: yup
    .string()
    .required("영문, 숫자포함 8자리를 입력해주세요.")
    .min(8, "최소 8자 이상 가능합니다")
    .max(15, "최대 15자 까지만 가능합니다")
    .matches(
      /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,15}$/,
      "영문 숫자포함 8자리를 입력해주세요."
    ),
  passwordConfirm: yup
    .string()
    .oneOf([yup.ref("memberPassword")], "비밀번호가 다릅니다."),
});


const {
  register,
  handleSubmit,
  watch,
  formState: { errors },
} = useForm({
  mode: "onChange",
  resolver: yupResolver(formSchema),
});
```

[그림13] yup 적용 코드

```
<input
  name="memberEmail"
  placeholder="이메일"
  {...register("memberEmail")}
  value={memberEmail}
  onChange={(e) => setMemberEmail(e.target.value)}
  // register에 포함된 구문임 onChange={(e)=> setUserEmail(e.target.value)}
/>
{errors.memberEmail && <p>{errors.memberEmail.message}</p>}
<input
  name="memberPassword"
  type="password"
  placeholder="비밀번호"
  value={memberPassword}
  {...register("memberPassword")}
  onChange={(e) => setMemberPassword(e.target.value)}
/>
{errors.memberPassword && <p>{errors.memberPassword.message}</p>}

<input
  type="password"
  name="passwordConfirm"
  placeholder="비밀번호 확인"
  {...register("passwordConfirm")}
/>
```

[그림14] yup 적용 코드



a

이메일 형식이 아닙니다.

• •

최소 8자 이상 가능합니다

•


비밀번호가 다릅니다.

010101

01011111111

회원가입

[그림 15] 회원가입 페이지 yup



asd@asdf.com

• • • • •

• • • • •

010101

01011111111

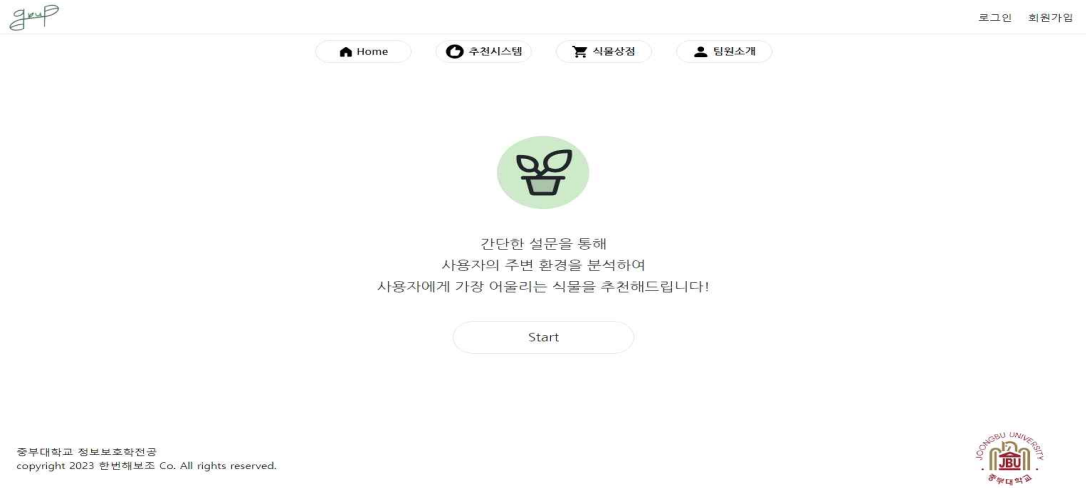
회원가입

[그림 16] 회원가입 페이지 yup

지정된 형식이 지켜지지 않으면 오류 메시지와 함께 회원가입이 이뤄지지 않는 것을 알 수 있다.

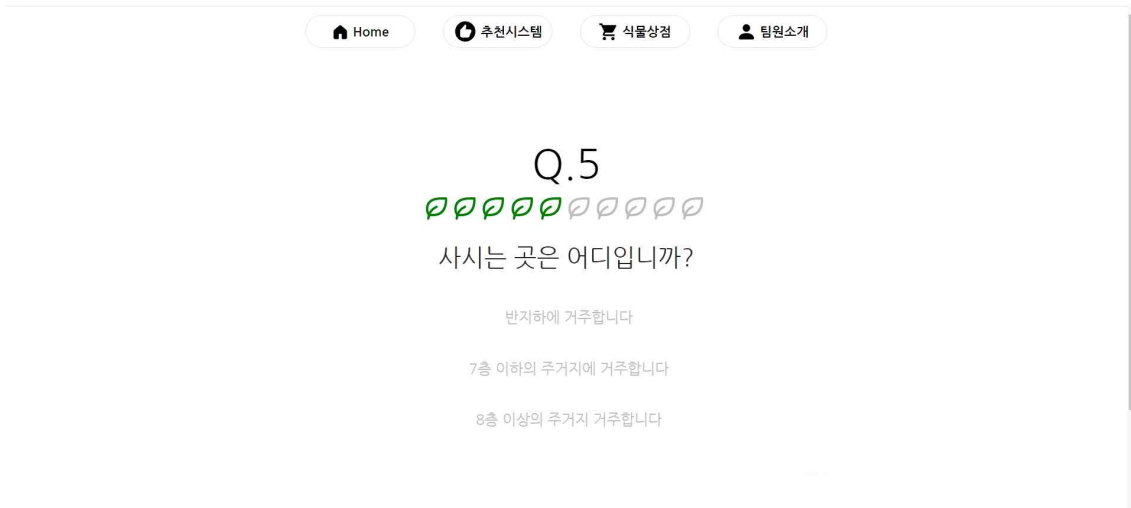
2.1.3 추천 시스템 페이지

10개의 질문을 통해 사용자의 개인별 환경설정을 진행하여, 개인별 맞춤 식물을 추천해준다. 그리고 10개의 질문은 식물의 빛, 물, 온도, 난이도에 관한 질문들이며 질문마다 가중치가 부여된다.



[그림17] 추천 시스템 페이지(설문 시작)

간단한 설명과 함께 버튼을 눌러 식물 추천이 시작될 수 있게 하였다.



[그림18] 추천 시스템 페이지(설문 진행)

나뭇잎 아이콘을 통해 직관적으로 설문 진행도를 알 수 있게 하였다.

나의 반려식물은?



셀로움

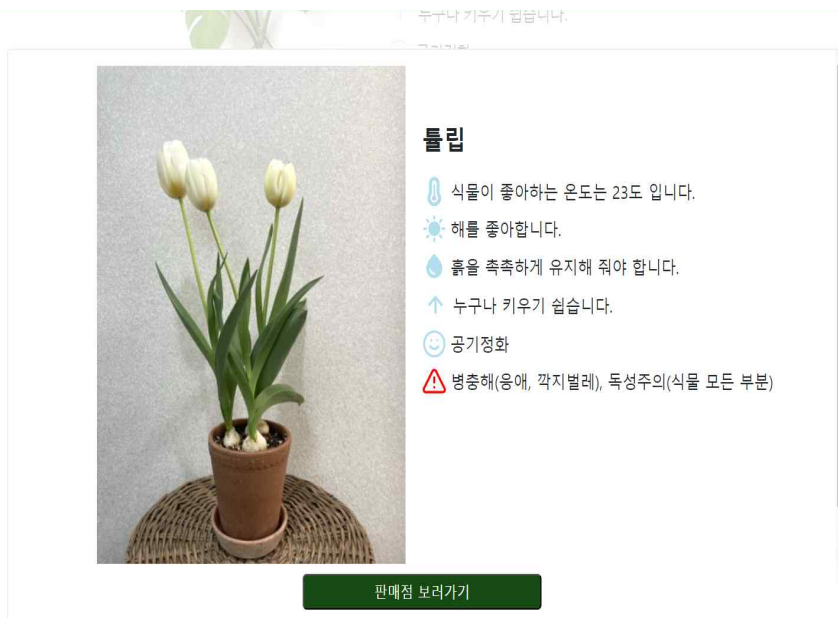
- 식물이 좋아하는 온도는 18.5도 입니다.
- 직사광선을 좋아하지 않습니다.
- 흙을 촉촉하게 유지해 줘야 합니다.
- ↑ 누구나 키우기 쉽습니다.
- 공기정화: 포름알데히드, 일산화탄소 제거, 이산화탄소 감소, 상대습도 증가, 음이온 발생, 공기정화
- ⚠ 병충해(진딧물, 깍지벌레), 독성주의(수액에 독성 함유)

판매점 보러가기

[그림19] 추천 시스템 페이지(결과)

사용자에게 가장 잘 어울리는 식물을 상단에 배치하여 바로 보이게 하였다. 식물에 대한 정보를 직관적인 디자인과 함께 그래픽, 이미지, 그림 등을 활용하여 사용자가 쉽게 이해할 수 있도록 정보를 같이 제시하였다.

판매점 보러가기를 통해 바로 식물 상품페이지로 이동하여 식물 구매를 편리하게 진행 할 수 있도록 하였다.



튤립

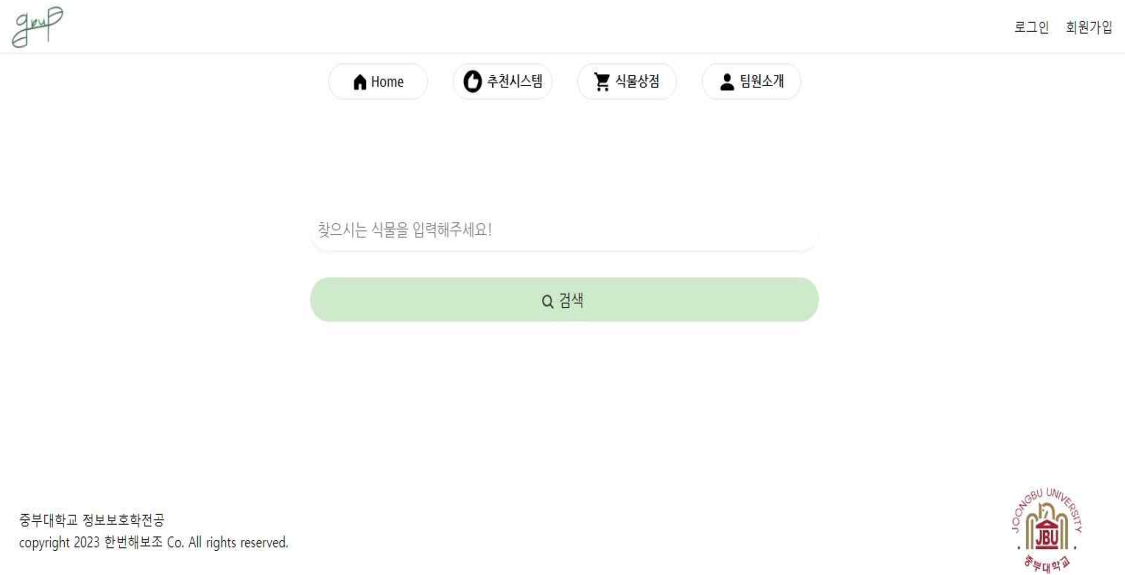
- 식물이 좋아하는 온도는 23도 입니다.
- ☀ 해를 좋아합니다.
- 흙을 촉촉하게 유지해 줘야 합니다.
- ↑ 누구나 키우기 쉽습니다.
- 공기정화
- ⚠ 병충해(응애, 깍지벌레), 독성주의(식물 모든 부분)

판매점 보러가기

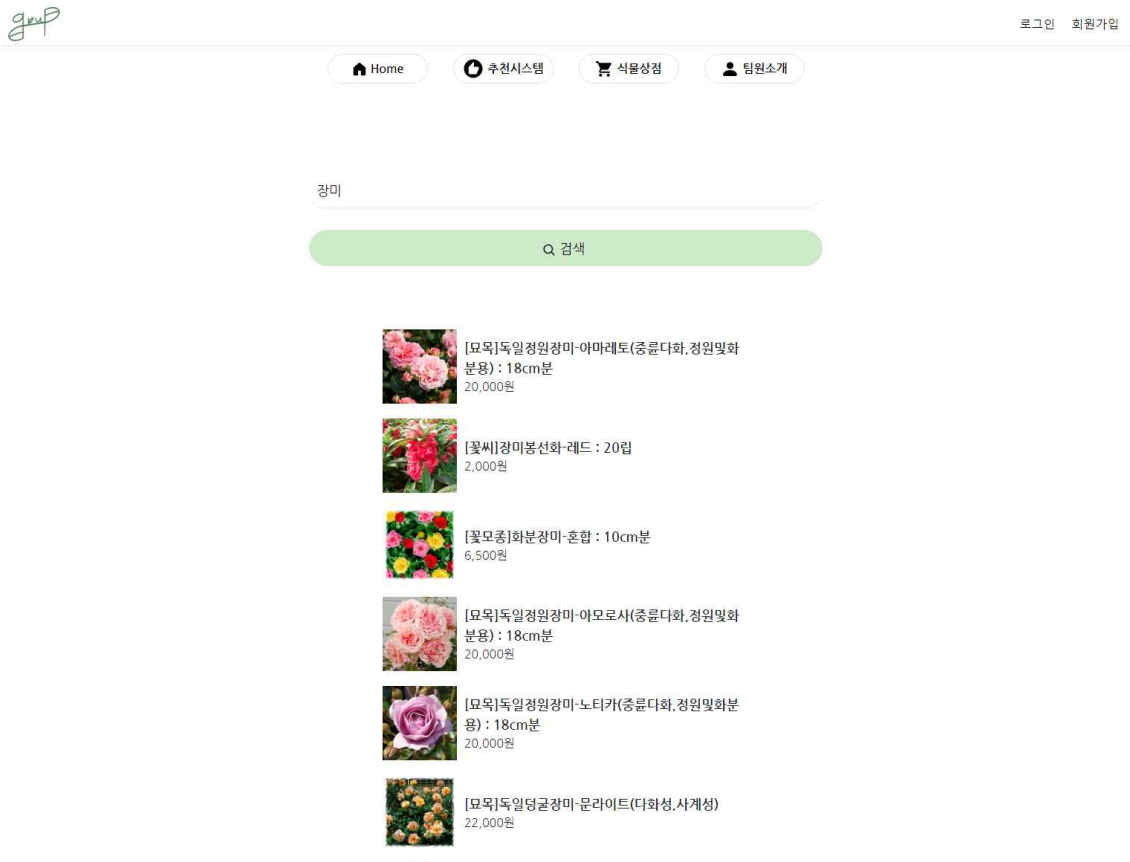
[그림20] 추천 시스템 페이지(모달창)

2, 3순위의 식물들은 'react-modal'을 통해 확인할 수 있도록 구현하였다. 'react-modal'은 렌더링할 때, 필요한 props만 설정하면 되어 간단한 설정이 가능하다. 또한, 모달을 통해 사용자에게 더 의미 있는 정보를 제공할 수 있게 도와준다.

2.1.4 식물 상점 페이지



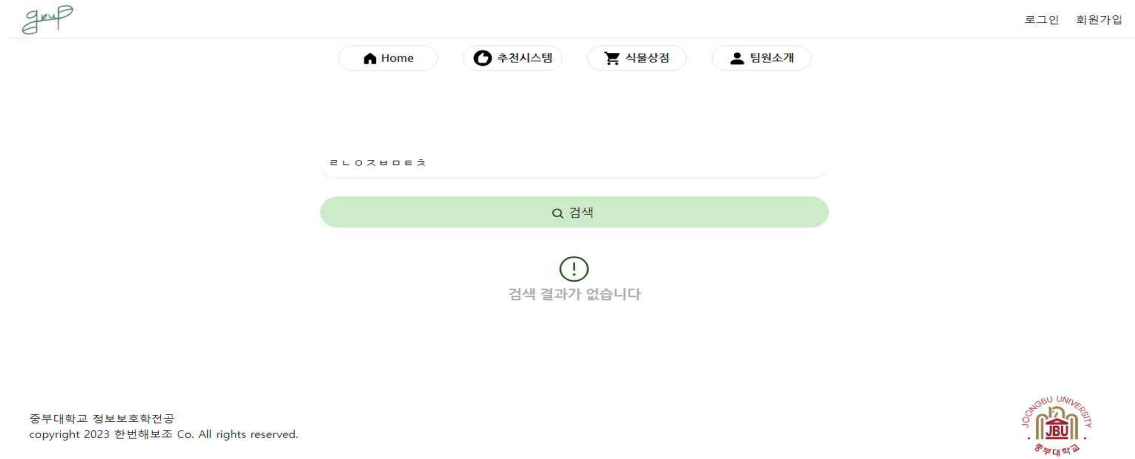
[그림21] 식물 상점 페이지(검색 전)



[그림22] 식물 상점 페이지(검색 후)

식물 상점 페이지는 크롤링을 통해 구현하였다. 크롤링을 통해 다양한 온라인 식물 판매 사

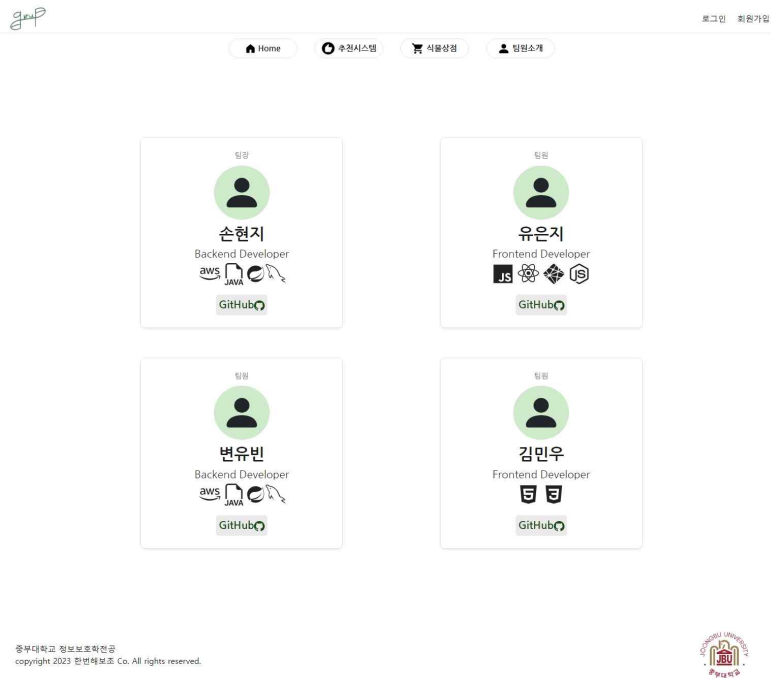
이트에서 해당 식물의 가격, 판매자 정보 등을 가져와 활용하였다. 식물 상점 페이지를 통해 사고 싶거나 궁금한 식물의 판매 가격, 판매처 등을 알 수 있다.



[그림23] 식물 이름이 부정확하거나 없을 때

식물 이름이 부정확하거나 없을 때는 ‘검색 결과가 없습니다’라는 설명으로 알려준다.

2.1.5 팀원 소개 페이지



[그림24] 팀원 소개 페이지

식물추천 웹페이지를 개발한 개발자 정보가 담긴 페이지로 개인별 깃허브 페이지에도 접속할 수 있게 하여 개인별 소스코드를 확인할 수 있다.

```

const onSubmit = async () => {
  try {
    const response = await axios.post(
      "/PlantsPlanet/save",
      {
        memberEmail: memberEmail, // memberEmail 상태를 서버에 전달
        memberPassword: memberPassword, // memberPasswd 상태를 서버에 전달
        memberBirth: memberBirth,
        memberNum: memberNum,
      },
      {
        headers: {
          "Content-Type": "application/json",
        },
      }
    );
    setMemberData(response.data);
    console.log(response.data);
    if (response.data == "success") {
      setTimeout(() => {
        navigate("/LoginPage");
      }, 2000);
    }
  } catch (error) {
    console.error(error);
  }
};

```

[그림25] axios 활용 코드

백엔드와의 소통은 axios를 통해 진행하였다. axios는 promise 기반의 API를 사용하여 비동기 코드를 작성하는데 편리하고, 요청과 응답을 자동으로 JSON으로 변환하는 장점이 있다.

2.2 BACKEND 구성

2.2.1 MySQL DB 구축

Table: answerplants

Columns:

answerId	bigint AI PK
answerLevel	decimal(4,2)
answerLight	decimal(4,2)
answerTemperature	decimal(4,2)
answerWater	decimal(4,2)

[그림26] 사용자의 가상식물을 추가하는 DB

사용자의 가상식물을 저장하는 DB를 추가하였다.

Table: member

Columns:

memberEmail	varchar(255) PK
memberBirth	varchar(255)
memberPhone	varchar(255)
memberPassword	varchar(255)

[그림27] 회원 정보를 추가하는 DB

	memberEmail	memberBirth	memberPhone	memberPassword
...	qqq@gmail.com	001122	00011112222	\$2a\$10\$mL3ly.OweGliOWrcOz8HqeFpZD7n1hs...

[그림28] DB 비밀번호 암호화

```
public void save(MemberDTO memberDTO){
    memberDTO.setMemberPassword(BCrypt.hashpw(memberDTO.getMemberPassword(), BCrypt.gensalt()));
    memberRepository.save(MemberEntity.toMemberEntity(memberDTO));
}
```

[그림29] DB 비밀번호 암호화

회원가입할 때 비밀번호를 BCrypt로 암호화해서 저장한다. 따라서 관리자라도 이용자의 비밀번호를 알고 접근할 수 없다.

[식물 DB]

순서대로 식물 고유 id, 이름, 사진, 온도, 광량, 습도, 난이도, 선택 경험, 효과, 주의할 점 순서로 구성되어있다. 실제 사진은 프론트 서버에 저장하고 DB에는 그 경로를 저장한다.

Table: plants

Columns:

plantsId	int PK
plantsName	varchar(255)
plantsPic	varchar(255)
plantsTemperature	decimal(4,2)
plantsLight	decimal(4,2)
plantsWater	decimal(4,2)
plantsLevel	decimal(4,2)
plantsSelected	int
plantsEffect	varchar(255)
plantsNotice	varchar(255)

[그림30] 크롤링 DB

Table: product

Columns:

id	int AI PK
storeTitle	varchar(255)
storeName	varchar(255)
price	varchar(255)
storeLink	varchar(255)
imgUrl	varchar(255)
searchName	varchar(255)

[그림31] 크롤링 DB

2.2.2 추천 시스템 구축

```
q: '식물을 어느 장소에서 키울 예정입니까?',
a: [
  { answer: '빛이 잘 드는 베란다에서 키울 예정입니다', 'answerTemperature' : -1.5, 'answerLight': 4.5, type: [] },
  { answer: '빛이 잘 안드는 베란다에서 키울 예정입니다', 'answerTemperature' : -3, 'answerLight': -3.5, type: [] },
  { answer: '빛이 잘 드는 실내에서 키울 예정입니다', 'answerTemperature' : 2, 'answerLight': 4, type: [] },
  { answer: '빛이 잘 안드는 실내에서 키울 예정입니다', 'answerTemperature' : -4, 'answerLight': -5, type: [] },
]
```

[그림32] data.js

설문조사로 각 항목(광량, 습도, 난이도, 온도)을 계산해서 사용자가 원하는 가상의 식물을 생성한다.

```
public RecommendDTO question(RecommendDTO recommendDTO) {
    RecommendEntity recommendEntity = RecommendEntity.toRecommendEntity(recommendDTO);
    recommendRepository.save(recommendEntity);
    RecommendDTO dto = RecommendDTO.toRecommendDTO(recommendEntity);
    return dto;
}
```

[그림33] RecommendService

설문조사로 나온 사용자가 원하는 가상의 식물을 저장한다.

```

@Query(value = "SELECT p.plantsId " +
    "FROM plants p, answerplants a " +
    "WHERE a.answerId = :userAnswerId " +
    "AND ABS(p.plantsLevel - a.answerLevel) = (" +
    "SELECT ABS(p1.plantsLevel - a1.answerLevel) " +
    "FROM plants p1, answerplants a1 " +
    "WHERE a1.answerId = :userAnswerId " +
    "ORDER BY ABS(p1.plantsLevel - a1.answerLevel) " +
    "LIMIT 1)" +
    "ORDER BY RAND() " +
    "LIMIT 1",
    nativeQuery = true)
Long findLevelABS(@Param("userAnswerId") Long userAnswerId);

```

[그림34] PlantsRepository

사용자의 가상식물의 난이도와 가장 가까운 난이도를 가진 무작위 식물 하나 선택한다.

```

Long levelABSId = recommendService.findLevelABS(userAnswerId);
model.addAttribute("attributeName: "levelABSId", levelABSId);
plantsDTO = recommendService.getPlantsById(levelABSId);
if (!containsPlantsId(resultingPlantsList, levelABSId)) {
    Map<Long, Float> resultingMap = new HashMap<>();
    getResulting(levelABSId, recommendDTO, plantsDTO, resultingMap, resultingPlantsList);
}

Long temperatureABSId = recommendService.findTemperatureABS(userAnswerId);
model.addAttribute("attributeName: "temperatureABSId", temperatureABSId);
plantsDTO = recommendService.getPlantsById(temperatureABSId);
if (!containsPlantsId(resultingPlantsList, temperatureABSId)) {
    Map<Long, Float> resultingMap = new HashMap<>();
    getResulting(temperatureABSId, recommendDTO, plantsDTO, resultingMap, resultingPlantsList);
}

Long lightABSId = recommendService.findLightABS(userAnswerId);
model.addAttribute("attributeName: "lightABSId", lightABSId);
plantsDTO = recommendService.getPlantsById(lightABSId);
if (!containsPlantsId(resultingPlantsList, lightABSId)) {
    Map<Long, Float> resultingMap = new HashMap<>();
    getResulting(lightABSId, recommendDTO, plantsDTO, resultingMap, resultingPlantsList);
}

Long waterABSId = recommendService.findWaterABS(userAnswerId);
model.addAttribute("attributeName: "waterABSId", waterABSId);
plantsDTO = recommendService.getPlantsById(waterABSId);
if (!containsPlantsId(resultingPlantsList, waterABSId)) {
    Map<Long, Float> resultingMap = new HashMap<>();
    getResulting(waterABSId, recommendDTO, plantsDTO, resultingMap, resultingPlantsList);
}

```

[그림35] levelABS

```

private void getResulting(Long plantsId, RecommendDTO recommendDTO, PlantsDTO plantsDTO, Map resultingMap, List<Map<Long, Float>> resultingPlantsList) {
    Float similarity;

    BigDecimal temperatureABS = plantsDTO.getPlantsTemperature().subtract(recommendDTO.getAnswerTemperature());
    BigDecimal lightABS = plantsDTO.getPlantsLight().subtract(recommendDTO.getAnswerLight());
    BigDecimal waterABS = plantsDTO.getPlantsWater().subtract(recommendDTO.getAnswerWater());
    BigDecimal levelABS = plantsDTO.getPlantsLevel().subtract(recommendDTO.getAnswerLevel());

    if (plantsDTO.getPlantsSelected() == 0) {
        similarity = ((temperatureABS.add(lightABS.add(waterABS.add(levelABS))))).divide(BigDecimal.valueOf(4), scale: 3, RoundingMode.HALF_EVEN).floatValue();
    } else {
        similarity = ((temperatureABS.add(lightABS.add(waterABS.add(levelABS))))).divide(BigDecimal.valueOf(4), scale: 3, RoundingMode.HALF_EVEN)
            .subtract(BigDecimal.valueOf(plantsDTO.getPlantsSelected() / 10)).floatValue();
    }

    resultingMap.put(plantsId, similarity);
    resultingPlantsList.add(resultingMap);
}

```

[그림36] 항목별 4개의 식물 구하기

이렇게 항목별로 총 4개의 식물을 구한다.

식물 1은 온도 10 광량 10 습도 10 난이도 10
 식물 2는 온도 20 광량 20 습도 20 난이도 20이고
 사용자의 가상식물은 온도 12 광량 16 습도 21 난이도 5일 때,
 각 식물의 항목과 비교해 차이를 구한다.

식물 1은 온도 2 광량 6 습도 11 난이도 5
 식물 2는 온도 8 광량 4 습도 1 난이도 15

그리고 식물별로 각 항목의 평균치를 구한다. 이 값이 작을수록 사용자의 가상식물과 비슷하다는 뜻이다. 만약 식물 1이 이전에 다른 사용자에게 선택된 경험이 있다면, 이 또한 계산에 포함해 평균치에서 값을 뺀다. 이전에 선택된 경험이 있으면 다른 사람들도 관심을 가지는 인기 많은 식물로 간주하고 평균치 값을 줄인다. 평균치가 작을수록 우리가 추천하는 식물과 가깝다는 것을 의미한다. 마지막으로 이 값을 작은 순서대로 나열하고, 앞에서부터 1, 2, 3등 추천 식물이 된다

```

@PostMapping("/api/resultPlantsSelected")
public void setPlantsSelected(@RequestBody String plantsName) throws IOException {
    String replacedPlantsName=plantsName.replace(target: "\\", replacement: "");
    System.out.println("set Selected plantsName: " + replacedPlantsName);
    recommendService.setPlantsSelected(replacedPlantsName);
}

```

[그림37] RecommendController

추천 시스템 페이지에서 선택 경험을 추가할 수 있게 하였다.

2.2.3 식물 상점 구축

```
@Service
@Transactional
public class JsoupService {

    @usage
    public void searchPlant(String searchKeyword) throws IOException {
        String searchName = searchKeyword;
        String encodedSearchName = URLEncoder.encode(searchName, enc: "UTF-8");

        if(checkIfSearchNameExists(searchName) == true) {
            deleteBySearchName(searchName);
            String flowerPlusUrl = "https://www.flowerseed-mall.com";
            String flowerUrl = "https://www.flowerseed-mall.com/shop/shopbrand.html?search=";
            Document flowerDoc = Jsoup.connect(url: flowerUrl + encodedSearchName + "&refer=https:").get();

            Elements products = flowerDoc.select(cssQuery: "li.item_list.item_list2");

            for (Element product : products) {
                String price = product.select(cssQuery: "div.info > p.prdprice > span").text();
                String storeTitle = product.select(cssQuery: "div.info > p.prdname.clear_fix").text();
                String storeLink = flowerPlusUrl + product.select(cssQuery: "div.tumb > a").attr(attributeKey: "href");
                String imgUrl = flowerPlusUrl + product.select(cssQuery: "div.tumb > a > img").attr(attributeKey: "src");
                String storeName = "꽃씨들";

                saveProductData(storeTitle, storeName, price, storeLink, imgUrl, searchName);
            }
        }
    }
}
```

[그림38] 제품 검색 서비스

Jsoup은 HTML 문서에 저장된 데이터를 구문 분석, 추출 및 조작하도록 설계된 오픈소스 Java 라이브러리로, 제품 검색 서비스에 Jsoup 라이브러리를 이용하여 해당 페이지에서 요청받은 식물 이름을 총 세 개의 식물 판매 사이트에 검색하여 해당 HTML 문서를 분석하고 추출하여 크롤링 DB에 검색 정보를 집어넣는 방식으로 사용하였다.

```
@Transactional(readOnly = true)
public boolean checkIfSearchNameExists(String searchName) {
    Query query = entityManager.createQuery( @String: "SELECT CASE WHEN COUNT(p) > 0 THEN true ELSE false END FROM Product p WHERE p.searchName = :searchName");
    query.setParameter( name: "searchName", searchName);
    return (boolean) query.getSingleResult();
}
```

[그림39] 중복 단어 확인 코드

checkIfSearchNameExists의 경우, 같은 단어를 검색한 기록이 크롤링 DB에 저장되어 있는지 확인하고 검색 결과가 이미 DB에 있다면 True를 반환한다. 이미 해당 검색어가 DB에 존재하는 경우에는 DB에 있는 동일한 검색어를 가지고 있는 튜플들을 지워버리고 새로 검색하여 DB에 저장하는 형식이다.

```

0 usage
public void saveProductData(String storeTitle, String storeName, String price, String storeLink, String imgUrl, String searchName) {
    String sql = "INSERT INTO product (storeTitle, storeName, price, storeLink, imgUrl, searchName) VALUES (?, ?, ?, ?, ?, ?)";
    jdbcTemplate.update(sql, storeTitle, storeName, price, storeLink, imgUrl, searchName);
}

1 usage
public void deleteBySearchName(String searchName) { productRepository.deleteBySearchName(searchName); }

```

[그림40] DB 정보, deleteBySearchName 코드

DB에 정보를 넣는 코드로 식물 판매 제목, 상점 이름, 가격, 상점 링크, 사진 주소, 검색명 등의 정보가 들어가게 된다.

하단의 deleteBySearchName의 경우에는 같은 정보를 검색하면 해당 검색어가 들어간 컬럼을 지우는 메소드이다.

```

@RestController
@Transactional
public class JsoupController {

    @Autowired
    private JsoupService jsoupService;

    @PostMapping("/api/searchSubmit")
    public ResponseEntity<List<Product>> handleSubmit(@RequestBody String searchName) throws IOException {
        String stringWithQuotes = searchName;
        String stringWithoutQuotes = stringWithQuotes.replace(target: "\"", replacement: "");

        jsoupService.searchPlant(stringWithoutQuotes);

        List<Product> entities = jsoupService.getProductsBySearchName(stringWithoutQuotes);

        return ResponseEntity.ok(entities);
    }
}

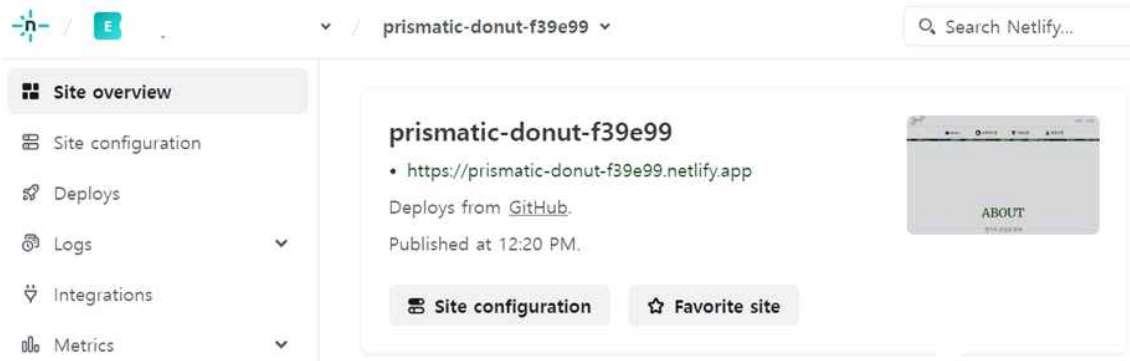
```

[그림41] Jsoup 컨트롤러

사용자가 식물 상점 검색 서비스 페이지에서 식물 정보 검색을 위해 식물명을 입력하고 검색 버튼을 누른다. 이후 검색 정보가 들어오면 “장미”와 같은 “” 형태로 컨트롤러에 들어오기 때문에 장미라는 단어만 추출하여 Jsoup 서비스에서 해당 단어를 가지고 검색을 진행하고 DB에 저장하게끔 요청을 보낸다. 그리고 List<Product>로 검색한 단어로 저장된 튜플들을 불러와서 사용자 화면에 식물 상점 정보를 리스트 형태로 반환해서 줄 수 있도록 한다.

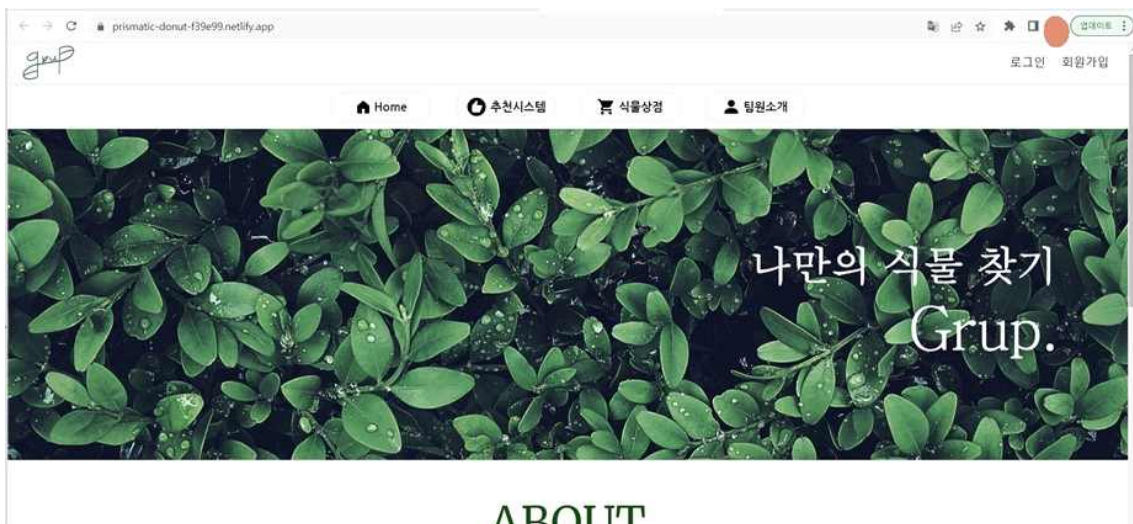
2.3 배포

2.3.1 FRONTEND 배포



[그림42] Netlify 배포

<https://prismatic-donut-f39e99.netlify.app/>



[그림43] Netlify 배포 결과

프론트의 배포는 Netlify로 진행하였다. Netlify는 GitHub, GitLab 등과 계정 연동 및 쉬운 호스팅을 제공한다. CDN, Continuous Deployment(지속적 배포), One-Click HTTPS 제공 등 고성능 사이트나 웹 응용 프로그램을 제작하는데 필요한 다양한 서비스들을 쉽고 빠르게 제공한다.

2.3.2 BACKEND 배포

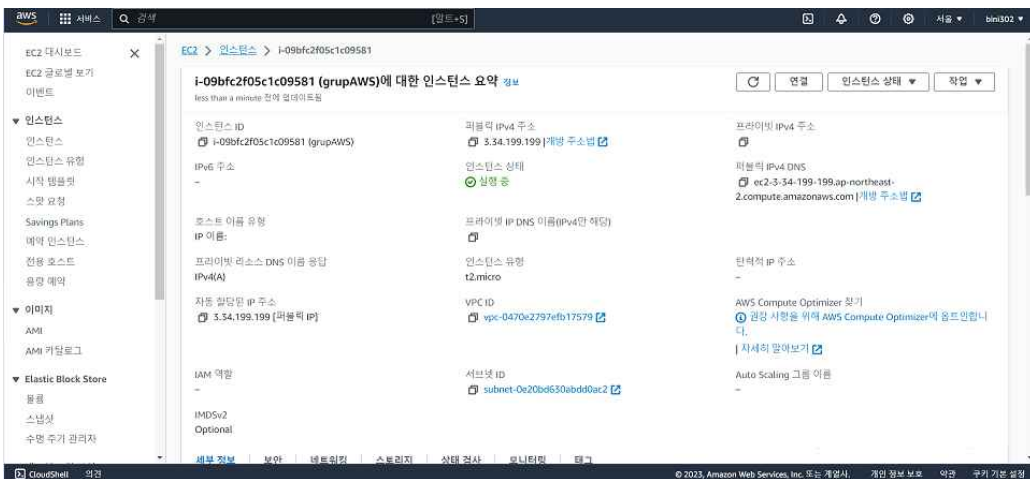
RDS는 AWS에서 제공하는 관계형 데이터베이스 서비스로 간편하게 설정, 운영 및 확장할 수 있다. 하드웨어 프로비저닝, 데이터베이스 설정, 패치 및 백업과 같은 시간 소모적인 관리 작업은 자동화하고, 사용자가 원하는 크기로 조정 가능한 용량을 제공한다. 자동 백업도 가능하고 세 가지 인증 방법, 자체 보안 시스템이 있기 때문에 안전한 작업이 가능하다. RDS 구축이 완료되면, 엔드포인트로 접속해 적절한 인증 방법을 거친 사용자는 DB 작업이 가능하다.



[그림44] AWS RDS

AWS의 RDS를 사용해 DB 서버를 띄우고 백엔드는 각자 MYSQL 워크벤치에서 RDS를 연결해 작업한다.

EC2는 클라우드에 인스턴스(컴퓨터)를 빌리고, 그 컴퓨터의 컴퓨팅 파워를 사용자가 원하는 만큼 조절할 수 있는 서비스다. 일종의 컴퓨터 템플릿이라 볼 수 있는 AMI를 사용해 간편하게 서버를 배포할 컴퓨터를 구성하고, 인스턴스 타입과 볼륨도 사용자 수를 예측하여 적절하게 설정한다. 보안그룹도 사용할 포트만 열어놓고 AWS의 보안 시스템을 이용하면 서버를 안전하게 사용한다. 추가로 탄력적 IP(Elastic IP)를 사용하면 주소를 고정해 더 편리하게 웹 서비스를 이용한다.



[그림45] AWS EC2

AWS의 EC2로 서버를 배포했다. 위의 퍼블릭 주소로 들어가면 어디서든 웹사이트에 연결된다.

3. 결 론

3.1 결 론

사용자들은 이 추천 시스템이 포함된 쉬운 방식의 설문을 통하여 자신의 원하는 식물을 추천받는 것에 쉽게 접근할 수 있게 된다. 그리고 웹사이트를 통해 자신에게 맞는 식물을 쉽게 선택하고 구매할 수 있다. 또한, 추천이 끝나면 각 식물에 대한 상세한 정보와 추가 정보들을 제공하여 사용자가 식물을 더욱 성공적으로 키우는 데 많은 도움을 줄 수 있다. 개별로 제공되는 기능인 식물 상점 페이지에서도 찾고 싶은 식물을 바로 찾을 수 있게 정보를 제공하는 것도 가능하다. 이를 통해 사용자들은 손쉽게 다양한 종류의 식물을 경험하고 자신에게 맞는 식물을 찾아낼 수 있으며, 식물을 키우며 즐거움과 만족감을 느낄 수 있게 해주는 이 웹서비스를 경험하게 될 것이다.

3.2 기대효과

사용자들은 간단하고 직관적인 설문을 통해 자신에게 맞는 식물을 추천받을 수 있어 효율적인 식물 선택이 가능해졌다. 이는 사용자들이 특별한 식물 관련 지식 없이도 손쉽게 식물을 찾고 키울 수 있게 해줄 것이다. 또한, 웹사이트에서 제공하는 상세한 식물 정보와 추가 정보 같은 이런 세부적인 디테일이 포함되어 있기에 사용자들이 식물을 키우고 관리하는 데 큰 도움이 될 것이다. 자신이 선택한 식물에 대한 전반적인 정보뿐만 아니라 특별한 관리 방법 및 관리 팁을 얻으며 키우는 과정에서 느끼는 어려움을 자연스럽게 줄일 수 있다. 또한, 식물 상점 페이지를 통해 원하는 식물을 바로 찾을 수 있는 편리한 검색 기능과 상세한 상품 정보를 통해 사용자들이 만족하는 식물을 찾을 수 있을 것이다. 이 웹서비스를 통해 다양한 종류의 식물을 경험하고 적합한 식물을 선택하여 키우면서 식물에 관한 관심을 더욱 가질 수 있게 되고 더불어 식물을 키우는 경험을 통해 자연과 가까워지며 실내 환경을 더욱 쾌적하게 만들 수 있는 효과를 기대한다.

4. 별첨

4.1 참고 문헌

이진희, 이창래(2001), 『실내조경에 의한 스트레스 해소효과』, 한국조경학회지 = Journal of Korean institute of landscape architecture v.28 no.6

4.2 팀원 소개

손현지 : 백엔드

github 주소 : <https://github.com/sonhyunji>

변유빈 : 백엔드

github 주소 : <https://github.com/bini302>

유은지 : 프론트엔드

github 주소 : <https://github.com/EunjiYoo9932>

김민우 : 프론트엔드

github 주소 : <https://github.com/kmw1122>

4.3 소스 코드

프론트엔드 소스코드 : <https://github.com/EunjiYoo9932/Done>

백엔드 소스코드 : <https://github.com/bini302/grupEC2>

4.4 발표 자료



[그림46] 발표 자료 1



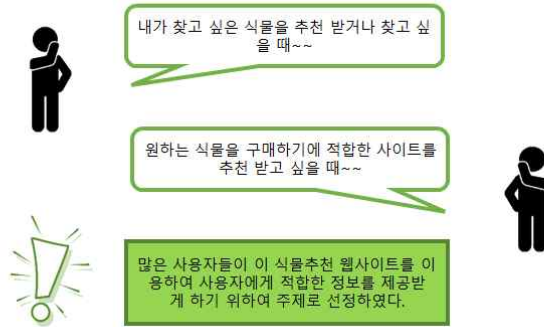
[그림47] 발표 자료 2

조원소개

이름	역할
손현지	백엔드
변유빈	백엔드
유은지	프론트엔드
김민우	프론트엔드

[그림48] 발표 자료 3

주제선정



[그림49] 발표 자료 4

구상도



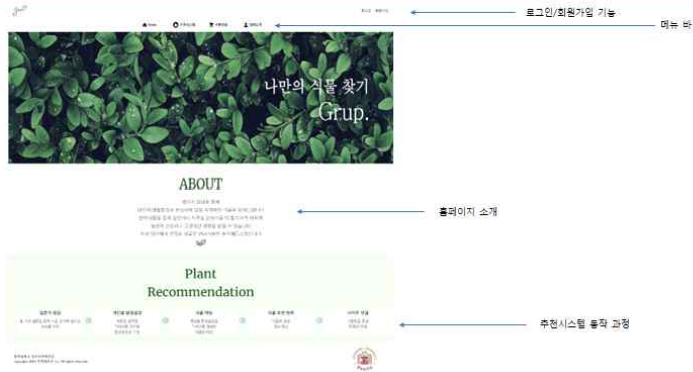
[그림50] 발표 자료 5

개발환경



[그림51] 발표 자료 6

FRONTEND 개발내용 - 홈페이지



[그림52] 발표 자료 7

FRONTEND 개발내용 - 로그인/회원가입 페이지



[그림53] 발표 자료 8

FRONTEND 개발내용 - 추천시스템 페이지



[그림54] 발표 자료 9

FRONTEND 개발내용 - 식물 상점 페이지



크롤링을 통해서 검색어에 대한 정보를 찾아 사고 싶거나 궁금한 식물의 각종 정보를 알려준다.

[그림55] 발표 자료 10

BACKEND 개발내용 - DB구축

DB구축	크롤링 DB	식물 DB
<p>Table: answerplants</p> <p>Columns:</p> <ul style="list-style-type: none"> answerid bigint AI PK answerlevel decimal(4,2) answerlight decimal(4,2) answerTemperature decimal(4,2) answerWater decimal(4,2) <p>Table: member</p> <p>Columns:</p> <ul style="list-style-type: none"> memberEmail varchar(255) PK memberBirth varchar(255) memberPhone varchar(255) memberPassword varchar(255) 	<p>Table: plants</p> <p>Columns:</p> <ul style="list-style-type: none"> plantid int PK plantName varchar(255) plantsTemperature decimal(4,2) plantsLight decimal(4,2) plantsWater decimal(4,2) plantsLevel decimal(4,2) plantsEffect varchar(255) plantsRobtce varchar(255) <p>Table: product</p> <p>Columns:</p> <ul style="list-style-type: none"> id int AI PK storeTitle varchar(255) storeName varchar(255) price varchar(255) storeLink varchar(255) imgUrl varchar(255) searchName varchar(255) 	<ul style="list-style-type: none"> 식물 고유 id 이름 사진 온도 광량 습도 난이도 선택 경험 효과 주의할 점
<p>사용자의 정보를 저장하는 DB를 추가했다</p>		<p>이 순서로 구성된다</p>

[그림56] 발표 자료 11

BACKEND 개발내용 - 추천시스템 구축1

```
data.js
식물을 어느 온도에서 키울 예정입니다.
{
  answer: '물이 잘 드는 온도에서 키울 예정입니다.', answerTemperature: 1.5, answerLight: 4.5, type: 1 },
  answer: '물이 잘 드는 온도에서 키울 예정입니다.', answerTemperature: 3, answerLight: 3.5, type: 1 },
  answer: '물이 잘 드는 온도에서 키울 예정입니다.', answerTemperature: 2, answerLight: 4, type: 1 },
  answer: '물이 잘 드는 온도에서 키울 예정입니다.', answerTemperature: 4, answerLight: 5, type: 1 },
}
```

설문조사로 각 항목(광량, 습도, 난이도, 온도)을 계산해서 사용자가 원하는 가상의 식물을 생성한다.

```
RecommendService
@Valid @RequestParam question @RequestParam recommendDTO {
  RecommendEntity recommendEntity = RecommendEntity.toRecommendEntity(recommendDTO);
  recommendRepository.save(recommendEntity);
  RecommendDTO dto = RecommendDTO.toRecommendDTO(recommendEntity);
  return dto;
}
```

설문조사로 나온 사용자가 원하는 가상의 식물을 저장한다.

[그림57] 발표 자료 12

BACKEND 개발내용 - 추천시스템 구축2

PlantsRepository

```

public boolean isPlantSelected(String plantName) {
    return plants.getSelectedPlantNames().contains(plantName);
}

public void setPlantSelected(String plantName) {
    plants.setSelectedPlantNames(plants.getSelectedPlantNames().add(plantName));
}

public void setPlantsSelected(Collection<String> plantNames) {
    plants.setSelectedPlantNames(new ArrayList<>(plantNames));
}

public void setPlantsSelected(String... plantNames) {
    setPlantsSelected(Arrays.asList(plantNames));
}

public void setPlantsSelected(Collection<String> plantNames, boolean isSelected) {
    if (isSelected) {
        setPlantsSelected(plantNames);
    } else {
        setPlantsNotSelected(plantNames);
    }
}

public void setPlantsNotSelected(Collection<String> plantNames) {
    plants.setSelectedPlantNames(plants.getSelectedPlantNames().removeAll(plantNames));
}

public void setPlantsNotSelected(String... plantNames) {
    setPlantsNotSelected(Arrays.asList(plantNames));
}

public void setPlantsNotSelected(Collection<String> plantNames, boolean isSelected) {
    if (isSelected) {
        setPlantsSelected(plantNames);
    } else {
        setPlantsNotSelected(plantNames);
    }
}

public void setPlantsNotSelected(String... plantNames, boolean isSelected) {
    setPlantsNotSelected(Arrays.asList(plantNames), isSelected);
}
    
```

사용자의 가상식물의 난이도와 가장 가까운 난이도를 가진 무작위 식물을 하나 선택한다.

(levelABS)

```

long levelABSID = recommendService.findNextLevelUserRecommendID(
    user, userLevelID);
levelABS = recommendService.getPlantLevel(levelABSID);
if (levelABS != null) {
    Map<String, Float> resultMap = new HashMap<>();
    getPlantLevel(levelABS, resultMap, resultPlantList);
}

long temperatureABSID = recommendService.findTemperatureUserRecommendID(
    user, userLevelID);
levelABS = recommendService.getPlantLevel(temperatureABSID);
if (levelABS != null) {
    Map<String, Float> resultMap = new HashMap<>();
    getPlantLevel(temperatureABSID, resultMap, resultPlantList);
}

long lightABSID = recommendService.findLightUserRecommendID(
    user, userLevelID);
levelABS = recommendService.getPlantLevel(lightABSID);
if (levelABS != null) {
    Map<String, Float> resultMap = new HashMap<>();
    getPlantLevel(lightABSID, resultMap, resultPlantList);
}

long waterABSID = recommendService.findWaterUserRecommendID(
    user, userLevelID);
levelABS = recommendService.getPlantLevel(waterABSID);
if (levelABS != null) {
    Map<String, Float> resultMap = new HashMap<>();
    getPlantLevel(waterABSID, resultMap, resultPlantList);
}
    
```

이렇게 항목별로 총 4개의 식물을 구한다.

[그림58] 발표 자료 13

BACKEND 개발내용 - 추천시스템 구축3

RecommandController

```

@PostMapping("/api/recommend")
public void setPlantsSelected(@RequestBody String plantName) throws IOException {
    String replacePlantName = plantName.replace(" ", "");
    System.out.println("set Selected plantName: " + replacePlantName);
    recommendService.setPlantsSelected(replacePlantName);
}
    
```

추천 시스템 페이지에서 선택 경험을 추가할 수 있게 하였다.

[그림59] 발표 자료 14

BACKEND 개발내용 - 식물 상점 구축1

제품 검색 서비스

```

public boolean isProductSelected(String productId) {
    return products.getSelectedProductNames().contains(productId);
}

public void setProductSelected(String productId) {
    products.setSelectedProductNames(products.getSelectedProductNames().add(productId));
}

public void setProductsSelected(Collection<String> productNames) {
    products.setSelectedProductNames(new ArrayList<>(productNames));
}

public void setProductsSelected(String... productNames) {
    setProductsSelected(Arrays.asList(productNames));
}

public void setProductsSelected(Collection<String> productNames, boolean isSelected) {
    if (isSelected) {
        setProductsSelected(productNames);
    } else {
        setProductsNotSelected(productNames);
    }
}

public void setProductsNotSelected(Collection<String> productNames) {
    products.setSelectedProductNames(products.getSelectedProductNames().removeAll(productNames));
}

public void setProductsNotSelected(String... productNames) {
    setProductsNotSelected(Arrays.asList(productNames));
}

public void setProductsNotSelected(Collection<String> productNames, boolean isSelected) {
    if (isSelected) {
        setProductsSelected(productNames);
    } else {
        setProductsNotSelected(productNames);
    }
}

public void setProductsNotSelected(String... productNames, boolean isSelected) {
    setProductsNotSelected(Arrays.asList(productNames), isSelected);
}
    
```

HTML 문서에 저장된 데이터를 구문 분석, 추출 및 조작하도록 설계된 오픈 소스 Java 라이브러리인 Jsoup을 사용하여 크롤링 하였다.

이미 같은 단어를 서지한 게 DB에 저장되어 있는지 확인하고 이미 서지 네임이 이미 DB에 있다면 True를 반환한다.

DB에 정보를 넣는 코드로 가게제목, 이름, 가격, 가게 링크, 사진주소, 서치네임 등의 정보가 들어가게 된다.

[그림60] 발표 자료 15

Jsoup 컨트롤러

```
JsoupController
@ExceptionHandler
public class JsoupController {

    @ExceptionHandler
    @RequestMapping(method = RequestMethod.GET)
    public ModelAndView searchProduct(@RequestParam String searchTerm, @RequestParam Integer pageNumber) {
        String url = "http://www.kyobobook.co.kr/search.do?keyword=" + searchTerm + "&page=" + pageNumber;
        Jsoup j = Jsoup.connect(url).get();
        List<Product> articles = j.select("div[id=goodsList]");
        return ModelAndView("articles");
    }
}
```

검색한 단어를 추출할 수 있도록 단어만 추출해서 Jsoup 서비스에 이 단어를 가지고 검색하고 DB에 저장하게끔 요청을 보낸다. 그리고 List<Product>로 검색한 단어로 저장된 튜플들을 불러와서 프론트에 정보를 리스트 형태로 반환해서 줄 수 있도록 한다.

[그림61] 발표 자료 16



결론

- 사용자는 이 추천 시스템이 포함된 쉬운 방식의 설문을 통하여 자신의 원하는 식물을 추천 받을 수 있다.
- 웹사이트를 통해 자신에게 맞는 식물을 쉽게 선택하고 구매할 수 있다.

기대효과

- 사용자는 웹페이지를 통해 효율적으로 식물을 선택하고 키울 수 있다.
- 식물에 대한 더 많은 관심을 가질 것으로 예상된다.

[그림62] 발표 자료 17

4.5 소개 자료

자료 : 식물추천 홈페이지 폼보드



[그림63] 소개 자료