HILLSTATE PARK

PORTFOLIO

한길수 안세홍 유희원 김하엘

TEAM PROJECT

CONTENTS

01
기획
주제선정
페르소나
시나리오

02
UI
와이어 프레임
프로토 타입

03
구현
코드 구현

04
마무리
평가 및 소감

00 타임라인

DAY 1

주제 선정 및 사용자 분석

- '힐스테이트 아파트 주차 관리 사이트'로 주제 선정
- 페르소나와 시나리오, 와이어프레임 제작

DAY 2

프로토타입 구현

- 와이어프레임을 토대로 프로토타입 구현

DAY 3

프로토타입 테스트

- 내비게이션/게시판/차량 등록/차량 조회 등의 기능 테스트
- 화면 너비에 따른 반응형 테스트

DAY 4

최종 웹사이트 완성

- 오류 수정 및 UI 리디자인
- 컨테이너, 폰트, 버튼 모서리의 사이즈 통일

DAY 5

문서 작업

- 깃허브에 배포 및 페이지 생성
- 프로젝트 과정 문서화

HILLSTATE PARK

아파트 주차 관리 사이트를 주제로 선정한 이유

- 공공기관에서 운영하는 공용 주차장 관리 사이트, 민간 기업에서 제공하는 실시간 주차 관제 서비스 등은 이미 많이 구축되어 있으나 아파트 단지 내에서 주민들의 차량 출입, 차량 등록, 주차 공간 등을 관리할 수 있는 웹사이트는 많지 않습니다.

- 또한, 이 주제를 통해 기존의 일반적인 주차 시스템과 차별화된 방향성을 가질 수 있다고 생각했습니다. 단순히 차량 위치를 보여주거나 주차 가능 여부만 제공하는 것이 아니라, 세대 정보와 연동된 입주민 전용 시스템, 방문 차량의 사전 등록, 관리자가 주차 내역을 통계적으로 확인할 수 있는 기능 등 아파트만의 특수한 구조와 요구 사항을 담을 수 있다는 점에서 구현 난이도는 다소 높지만 도전해 볼 만한 가치가 있다고 판단하였습니다.

01 기획 | 페르소나 (1)

이름 : 이수현 나이 : 31

직업 : 회사원 성별 : 여성

- 배경

매주 주말마다 혼자 계신 어머니를 찾아 아파트로 차량을 이용해 방문함.
어머니는 연세가 있어 스마트폰이나 인터넷 사용에 익숙하지 않아,
차량 등록은 항상 본인이 직접 처리함.

- 동기

과거에는 매번 관리실에 전화하거나
방문자 차량 등록지를 손으로 작성해야 해서 번거로웠음.
로그인이나 인증 절차 없이 빠르게 등록할 수 있는 웹사이트 기반 시스템을 원함.

- 사용 습관

금요일 저녁 또는 당일 아침에 모바일로 등록하는 경우가 많음
사무실에서도 데스크탑 PC를 이용한 등록 가능성이 있음
최소 입력, 직관적인 화면, 빠른 응답 속도를 중요하게 생각함

- 인용 문구

" 매주 가는데도 매번 전화하거나 서류 쓰는 건 너무 번거롭죠.
그냥 웹으로 몇 번 클릭해서 끝낼 수 있으면 정말 편할 것 같아요."

01 기획 | 페르소나 (2)

이름 : 황태욱 나이 : 50

직업 : 사업가 성별 : 남성

- 배경

최근 입주민들의 신뢰를 받아 동대표로 선출됨.
주차 질서와 등록 차량 관리에 꾸준히 관심을 가져옴.
세대별 차량 정보 확인을 통해 단지 내 주차 운영을 개선하고자 함.

- 동기

세대별 차량 등록 현황을 한눈에 보고 누락된 차량을 확인하고 싶고
한 세대에서 차량이 너무 많거나, 등록 정보가 오래된 경우 조정을
하고 싶어하며,
차량 번호, 등록일, 세대 번호 등으로 필터링해서 효율적으로 체크하고 싶어함

- 사용 습관

주로 아침에 출근해 사무실 PC로 웹사이트 접속
회의 전 주말에 간단히 체크하기 위해 모바일에서도 확인하는 경우 있음

- 인용 문구

"차량 정보를 한눈에 파악할 수 있어야
입주민 불편 없이 효율적인 관리가 가능합니다."

01 기획 | 시나리오 (1)

이름 : 이수현 나이 : 31

직업 : 회사원 성별 : 여성

- 상황

금요일 저녁, 이수현은 다음 날 어머니의 아파트를 방문하기 위해 방문 차량 등록을 하려 한다. 업무를 마치고 퇴근길 지하철 안에서 스마트폰으로 빠르게 처리하고자 한다.

1. 이수현은 아파트 방문 차량 등록을 위해 힐스테이트 주차 관리 사이트에 접속한다.
2. 로그인이나 복잡한 절차 없이, 방문 날짜나 목적 등의 간단한 정보만 입력하면 등록 가능한 화면을 본다.
3. 방문 일자를 토요일로 설정하고 차량 번호를 입력한 후, 등록 버튼을 눌러 완료한다.
4. 등록이 완료되었음을 나타내는 메시지를 확인하고 안심한다.
5. 추가 등록 없이, 1분 이내에 등록이 완료된다.

- 인용 문구

"이제 전화 안 해도 돼서 정말 편하네. 간단한 정보를 입력만 하면 끝이야!"

01 기획 | 시나리오 (2)

이름 : 황태욱 나이 : 50

직업 : 사업가 성별 : 남성

- 상황

월요일 오전, 황태욱 동대표는 주간 회의 전에 세대별 차량 등록 현황을 사무실 PC에서 확인하고자 한다. 그는 누락 차량이나 다량 등록 세대를 중심으로 확인하고 싶다.

1. 황태욱은 세대별 차량 등록 현황을 보기 위해 힐스테이트 주차 관리 사이트에 접속한다.

2. 차량 번호나 세대 번호에 따른 조회, 세대주 이름에 따른 조회 등 여러 옵션이 제공된다.

3. 세대 번호에 따른 조회를 선택해 한 세대당 차량 수를 확인한다.

4. 문제가 있는 세대 정보를 확인하고 출력하거나 스크린샷을 저장한다.

5. 이후 관리사무소에 자료를 전달한다.

- 인용 문구

"이렇게 쉽게 원하는 정보를 찾을 수 있다니, 앞으로도 이 사이트를 이용하면 되겠네."

02 UI | 와이어프레임

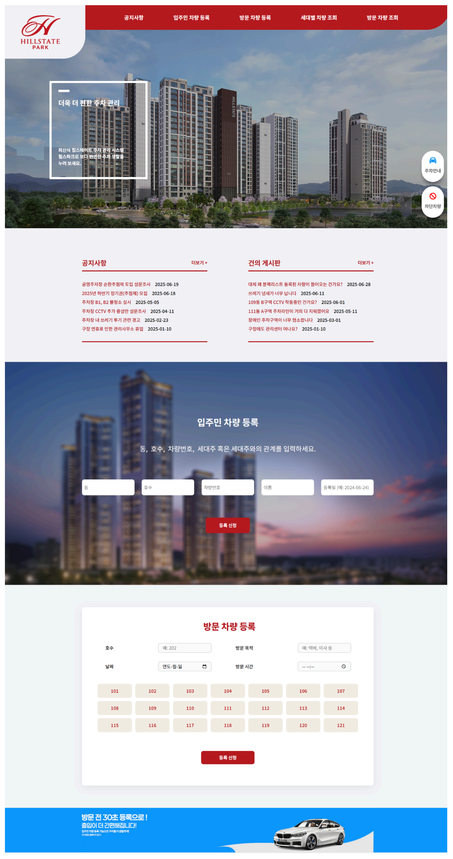
HILLSTATE PARK
더욱 더 편한 주차 관리
공지사항
건의게시판
공명주차장 순환주첨제 도입 설문조사 안내 2025-06-19
2025년 하반기 정기권(주첨제) 모집 안내 2025-06-18
주차장 B1 , B2 물청소 실시 안내 2025-05-05
주차장 CCTV 추가 증설안 설문조사 안내 2025-04-11
주차장 내 쓰레기 투기 관련 경고 안내 2025-02-23
구정 연휴로 인한 관리사무소 휴일 안내 2025-01-10
더 많은 내용 +
대체 왜 블랙리스트 등록된 차량이 왜 들어오... 2025-06-28
쓰레기 냄새가 너무 납니다 2025-06-11
109동 B구역 CCTV 작동중인 건가요? 2025-06-01
111동 A구역 주차라인이 거의 다 지워져... 2025-05-11
장애인 주차구역이 너무 협소합니다 2025-03-01
구정에도 관리센터 여나요? 2025-01-10
더 많은 내용 +
입주민 차량 조회
입주민 차량 조회
- 차량 번호 조회
- 차량 번호 조회
- 차량 번호 조회
차량 번호 조회
입력

방문 차량 등록/신청
세부 사항을 선택해주세요.
* 호수
* 방문 목적
* 방문 날찌
* 방문 시간
101 102 103 104 105 106
107 108 109 110 111 112
113 114 115 116 117 118
119 120 121
선택
등록 신청
입주민 차량 등록
동 , 호수 , 차종 , 세대주 혹은 세대주 와의 관계를 입력하세요.
동
호수
차량번호
관계
등록 신청

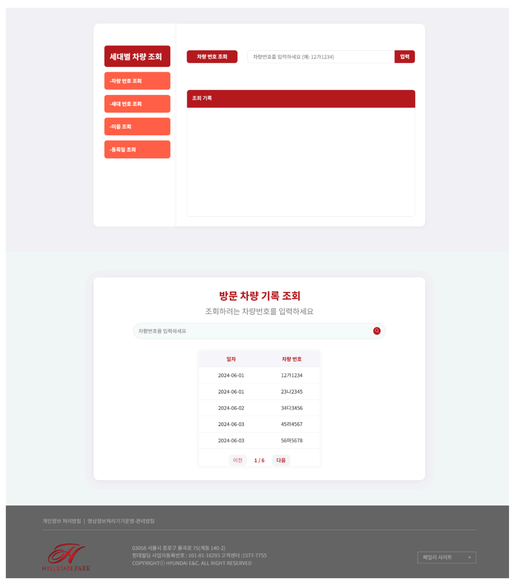
세대 방문 차량 기록 조회
조회 하시려는 세대를 입력해 주세요
일자
차량 번호
블랙리스트 차량 안내
차량 번호
사유
차종
등록 번호
주차장 안내도
B1
장애인 주차구역
여성 전용 주차구역
전기차 충전소
개인정보 처리방침 영상정보처리기기운영-관리방침

02 UI | 프로토타입

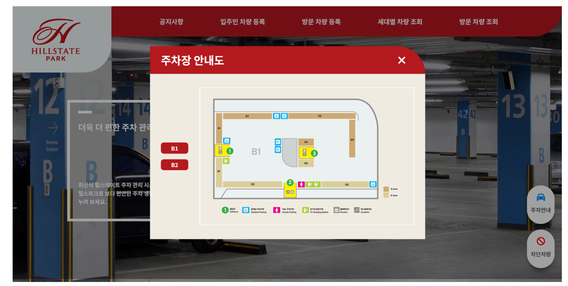
<main 1>

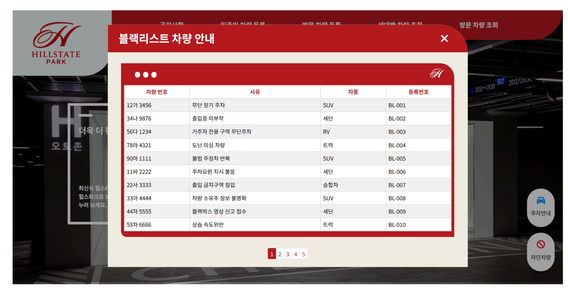


<main 2>

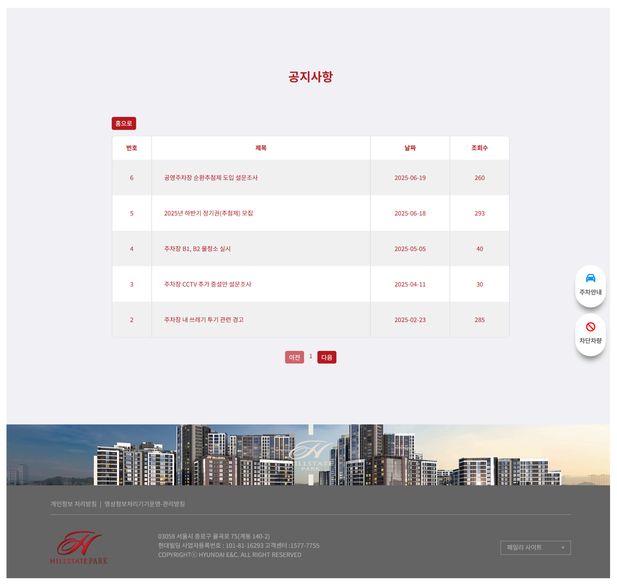


<modal>

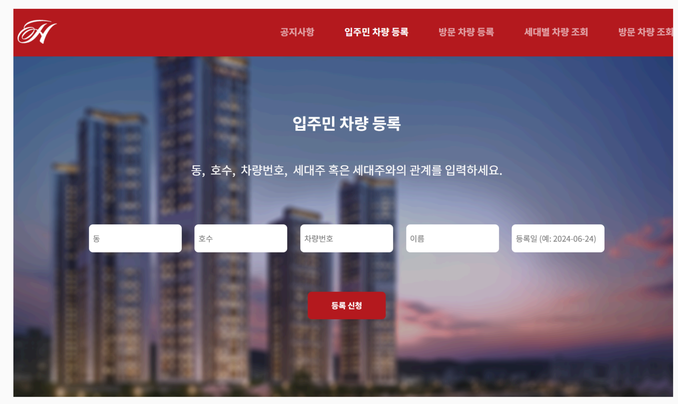




<sub>



메인 페이지 코드 구현 (1)



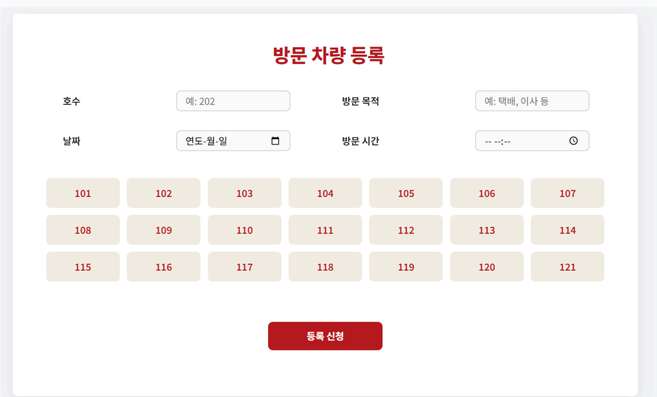

```
useEffect(() => {
    const handleScroll = () => {
        const scrollY = window.scrollY;

        const triggerPoint1 = window.innerWidth * 0.468;
        const triggerPoint2 = window.innerWidth * 0.788;
        const triggerPoint3 = window.innerWidth * 1.268;
        const triggerPoint4 = window.innerWidth * 1.868;
        const triggerPoint5 = window.innerWidth * 2.368;

        setIsFixed(scrollY > triggerPoint1);

        if (scrollY < triggerPoint1) {
            setActiveIndex(-1);
        } else {
            if(scrollY > triggerPoint1 && scrollY < triggerPoint2){
                setActiveIndex(0)
            }else if (scrollY > triggerPoint2 && scrollY < triggerPoint3){
                setActiveIndex(1)
            }else if (scrollY > triggerPoint3 && scrollY < triggerPoint4){
                setActiveIndex(2)
            }else if (scrollY > triggerPoint4 && scrollY < triggerPoint5){
                setActiveIndex(3)
            }else if (scrollY > triggerPoint5){
                setActiveIndex(4)
            }
```

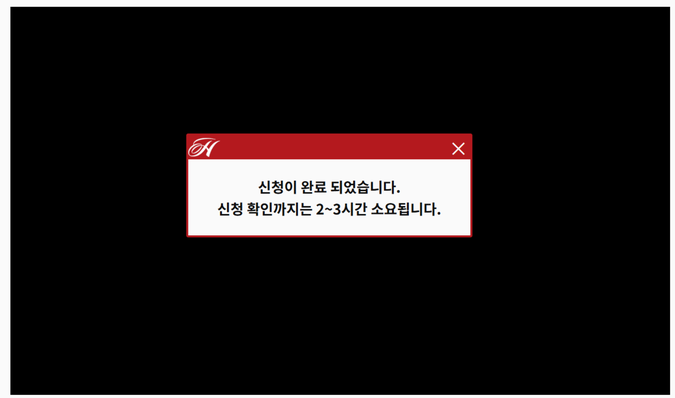
```
return (
    <div className='scrollNav'>
        <Navbar className={`nav ${isFixed ? 'fixed-nav' : ''}`} style={{ background:
        '#B81C22', borderBottom: '2px solid #B81C22' }}>
            <Container>
                <Navbar.Brand><img src={process.env.PUBLIC_URL + "/img/h_white.png"} onClick=
                {() => window.scrollTo({ top: 0, behavior: 'smooth' })} alt="Logo"
                className='logo' /></Navbar.Brand>
                <Nav className='scrollnav-text'>
                    <Nav.Link className={activeIndex === 0 ? 'active' : ''} onClick={() =>
                    window.scrollTo({ top: window.innerWidth * 0.468, behavior: 'smooth' })}
                    >공지사항</Nav.Link>
                    <Nav.Link className={activeIndex === 1 ? 'active' : ''} onClick={() =>
                    window.scrollTo({ top: window.innerWidth * 0.798, behavior: 'smooth' })}
                    >입주민 차량 등록</Nav.Link>
                    <Nav.Link className={activeIndex === 2 ? 'active' : ''} onClick={() =>
                    window.scrollTo({ top: window.innerWidth * 1.268, behavior: 'smooth' })}
                    >방문 차량 등록</Nav.Link>
                    <Nav.Link className={activeIndex === 3 ? 'active' : ''} onClick={() =>
                    window.scrollTo({ top: window.innerWidth * 1.868, behavior: 'smooth' })}
                    >세대별 차량 조회</Nav.Link>
                    <Nav.Link className={activeIndex === 4 ? 'active' : ''} onClick={() =>
                    window.scrollTo({ top: window.innerWidth * 2.368, behavior: 'smooth' })}
                    >방문 차량 조회</Nav.Link>
                </Nav>
            </Container>
        </Navbar>
    </div>
```

홈 화면에서 아래로 스크롤하면 화면 최상단에 고정된 내비가 나와 각각의 컴포넌트가 어떤 페이지인지 사용자에게 알려줍니다. 스크롤에 따라 페이지가 바뀌면서 그에 맞는 글씨가 활성화되기도 하고, 사용자가 글씨를 클릭하면 그에 맞는 페이지로 이동하기도 합니다. 로고를 클릭하면 스크롤이 올라가며 홈으로 이동하고, 고정된 내비는 사라집니다.

<React(JSX)>

| 메인 페이지 코드 구현 (2)






```jsx
const handleSubmit = (e) => {
  e.preventDefault();
  if (!hosu || !purpose || !date || !time || !selectedDong) {
    alert("모든 항목을 입력 해주세요");
    return;
  }
  setModalOpen(true);
};
```

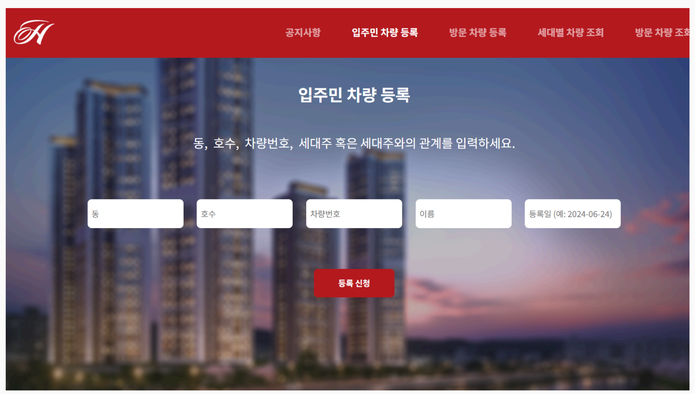
```jsx
<div className="visit-dong-area">
  <div className="visit-dong-list">
    {dongList.map((dong) => (
      <button
        key={dong}
        type="button"
        className={`visit-dong-btn${selectedDong === dong ? " selected" : ""}`}
        onClick={() => setSelectedDong(dong)}
      >
        {dong}
      </button>
    ))}
  </div>
```

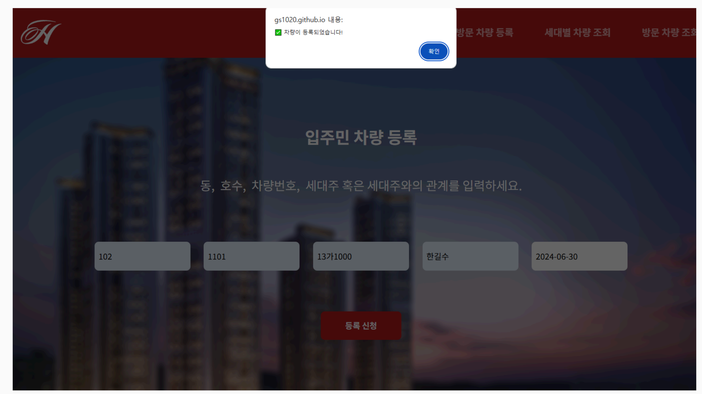
```jsx
    </button>
  </form>
  <Modal open={modalOpen} onClose={() => setModalOpen(false)}>
    신청이 완료 되었습니다.<br />
    신청 확인까지는 2~3시간 소요됩니다.
  </Modal>
</div>
div>
```

<React(JSX)>

방문 차량 등록 페이지는 모든 항목 입력 시에만 등록 버튼이 활성화되고, 누락된 경우에는 alert로 즉시 피드백이 제공되도록 구현했습니다.
동 선택은 버튼 배열에 selected 클래스를 적용해 시각적으로 명확히 표시되며, 등록 시 모달 팝업을 통해 간편한 UX를 제공합니다.

| 메인 페이지 코드 구현 (3)



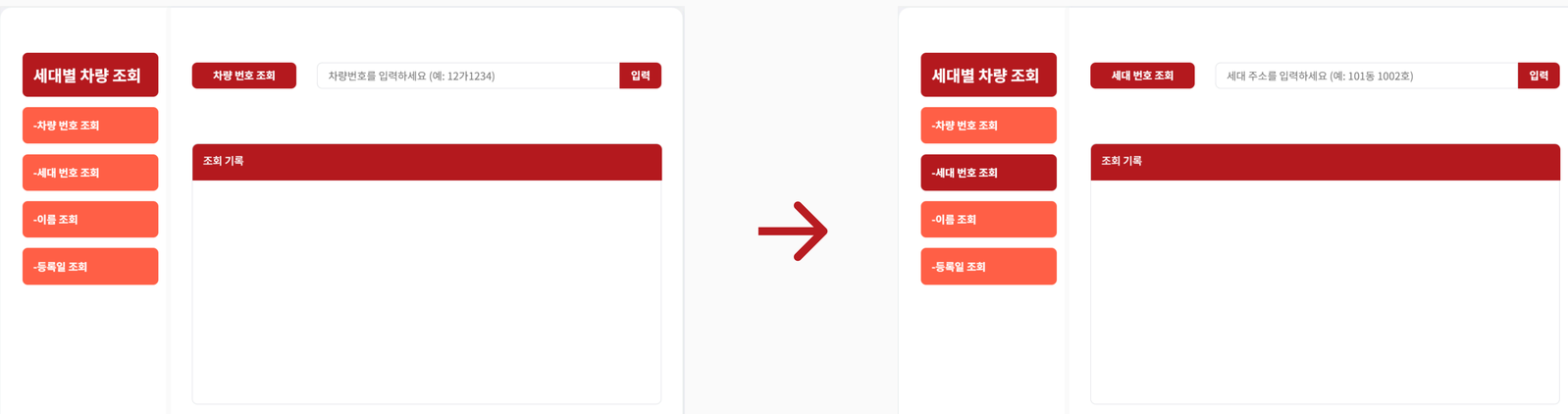



```react
const Addcar = ({ onRegister }) => {
  const [form, setForm] = useState({
    dong: '',
    hosu: '',
    vehicleNumber: '',
    name: '',
    registrationDate: '',
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setForm(prev => ({ ...prev, [name]: value }));
  };

  const handleSubmit = () => {
    const { dong, hosu, vehicleNumber, name } = form;
    if (!dong || !hosu || !vehicleNumber || !name) {
      alert('동, 호수, 차량번호, 이름은 필수입니다.');
      return;
    }

    const newVehicle = {
      unit: `${dong}동 ${hosu}호`,
      vehicleNumber,
      name,
      registrationDate: form.registrationDate || new Date().toISOString().split('T')[0],
    };

    onRegister(newVehicle);
    alert('✅ 차량이 등록되었습니다!');

    setForm({
      dong: '',
      hosu: '',
      vehicleNumber: '',
      name: '',
      registrationDate: '',
    });
```

<React(JSX)>

handleChange()로 각 input의 변경사항을 업데이트
상태 관리를 통해 입력값을 제어하고, 등록일이 없으면 현재 날짜를 자동으로 지정해주며, 등록 후 폼을 초기화합니다

메인 페이지 코드 구현 (4)



카테고리를 클릭할 때마다 타이틀 텍스트가
해당 항목에 맞게 변경되도록 구현했습니다.
또한, 입력값이 예시에 맞지 않으면 alert로 경고 문구가 뜨도록 처리하였고, 조건문은 if 대신 switch 문을 사용해 구현해 보았습니다.

```
const Carsearch = ({ data }) => {
  const handleSearch = () => {
    const keyword = search.trim()
    if (!keyword) {
      alert('검색어를 입력해주세요.')
      return
    }

    const found = data.filter((item) => {
      switch (searchType) {
        case 'vehicle':
          return item.vehicleNumber?.includes(keyword)
        case 'unit':
          return item.unit?.includes(keyword)
        case 'name':
          return item.name?.includes(keyword)
        case 'registrationDate':
          return item.registrationDate?.includes(keyword)
        default:
          return false
      }
    })


    setResults(found.length > 0 ? found : [{ notFound: true }])
  }

  const getSearchTitle = () => {
    switch (searchType) {
      case 'vehicle':
        return '차량 번호 조회 '
      case 'unit':
        return '세대 번호 조회'
      case 'name':
        return '이름 조회'
      case 'registrationDate':
        return '등록일 조회'
      default:
        return ''
    }
  }
```

<React(JSX)>

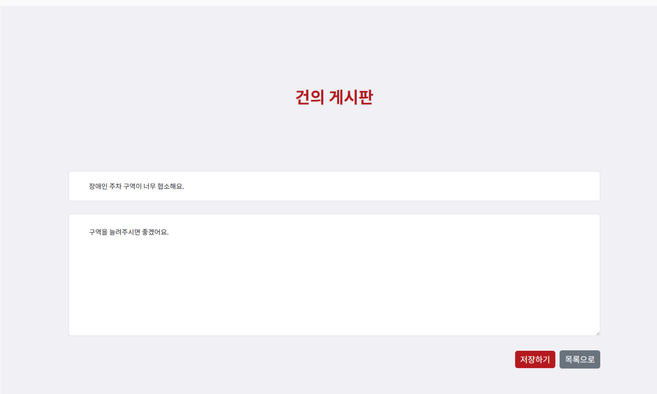
서브 페이지 코드 구현 (1)

```react
useEffect(() => {
    console.log(blacks.length)
    setLastPage(Math.ceil(blacks.length / 10))
}, [page])

<tbody>
    {
        blacks.slice((page-1)*10, page*10).map((black, i) =>
            <tr key={i}>
                <td>{black.vehicleNumber}</td>
                <td>{black.reason}</td>
                <td>{black.vehicleType}</td>
                <td>{black.registrationNumber}</td>
            </tr>
        )
    }
</tbody>
```

<React(JSX)>

페이지 버튼 클릭 시 slice()로 해당 구간의 데이터만 출력되며,
데이터 수에 따라 버튼 개수도 자동으로 계산되도록 하였습니다.
전체 레이아웃은 Container, Row, Col 구조로 반응형 대응이 가능하도록 구현했으며, JSON 형태의 블랙리스트 데이터를 useState로 불러와 테이블 형태로 화면에 출력하였습니다.

서브 페이지 코드 구현 (2)



```
listOk, setListOk] = useState(true);
readOk, setReadOk] = useState(false);
writeOk, setWriteOk] = useState(false);
editOk, setEditOk] = useState(false);

boardInfo, setBoardInfo] = useState({});

today = new Date().toISOString().split('T')[0];

title, setTitle] = useState('');
content, setContent] = useState('');

editId, setEditId] = useState(null);
editTitle, setEditTitle] = useState('');
editContent, setEditContent] = useState('');

errorMessage, setErrorMessage] = useState('');
```

```
( className="board_suggestion">
<div className="wrap">
    <h3 style={{ fontWeight: "bold" }}>건의 게시판</h3>

    {listOk && (
        <div>
            <div style={{ display: "flex" }}>
                <Button style={{ backgroundColor: "#B81C22", border: 0, marginBottom: "1vw"
                fontSize: "1vw" }} onClick={() => navigate('/')}>홈으로</Button>
                <Button onClick={boardWrite} style={{ backgroundColor: "#B81C22", border: 0
                marginBottom: "1vw", fontSize: "1vw", marginLeft: "0.5vw" }}>건의글 작성하기</
            </div>

            <Table striped bordered hover className='table_suggestion'>
                <thead>
                    <tr style={{ lineHeight: '3vw' }}>
                        <th style={{ width: "10%" }}>번호</th>
                        <th style={{ width: "55%" }}>제목</th>
                        <th style={{ width: "20%" }}>날짜</th>
                        <th style={{ width: "15%" }}>조회수</th>
                    </tr>
                </thead>
```

<React(JSX)>

건의 게시판에서 글을 작성하면 목록에 업데이트 됩니다. 글을 작성한 시점의 날짜가 자동으로 입력되고, 다른 글과 똑같이 조회수가 카운트 됩니다. 사용자가 직접 작성한 글은 삭제하거나 수정할 수 있습니다. 그렇지 않은 경우에는 삭제나 수정이 불가합니다.

| 평가 및 소감

항상 혼자서 작업만 하다가 이번에 팀원들과 팀 프로젝트를 처음 해봤는데 색다르고 좋은 경험이었던 거 같습니다.
이번에 팀장으로 뽑혀서 팀을 이끌게 되었고 처음엔 잘할 수 있을까 하는 고민도 있었지만 팀원들이 각자 해야 할 역할을 잘 수행해서 팀장으로서 많이 부족했지만 트러블도 없었고 잘 마치게 할 수 있었던 거 같아서 다행이었다고 생각이 들었습니다. 이번 프로젝트를 하면서 마치 작은 조직 안에서 함께 일하는 것 같은 느낌을 받았고,
실무와 비슷한 상황에서의 협업 경험을 해볼 수 있었던 점이 특히 좋았습니다. 단순히 코드를 작성하는 것뿐 아니라, 일정 조율, 역할 분배, 진행 상황 공유 같은 것들도 전부 팀장으로서 신경 써야 했기에, 평소에는 잘 느끼지 못했던 책임감도 자연스럽게 생겼습니다. 무엇보다 이번 경험을 통해 제 자신을 돌아볼 수 있었던 계기가 되었고,
어떤 부분에서 부족한지, 어떤 역량을 더 키워야 하는지도 알 수 있었습니다.
여전히 부족한 점이 많다는 것도 느꼈고, 앞으로 더 성장하기 위해 어떤 방향으로 노력해야 할지에 대해 고민하게 만든 아주 값진 시간이었다고 생각합니다.

| 평가 및 소감

프론트엔드 팀 프로젝트는 처음이라 걱정이 많았습니다. 주제가 주차 관리 사이트인데 저는 이런 사이트를 이용해본 적도 없고, 팀원들과 잘 어울릴 수 있을지도 미지수였기 때문입니다.

그런데 막상 작업에 들어가니, 서로 약한 부분을 도와주거나 의견을 공유하거나 하며 문제를 해결할 수 있어 혼자 작업할 때와 비교해 훨씬 수월하게 과제를 끝낼 수 있었습니다. 한 명이 와이어프레임을 만들기 시작하면, 페이지 하나가 끝나는 대로 다른 사람이 코드를 구현하고, 또 다른 사람은 검색이나 게시판 기능을 구현하는 데에 필요한 데이터를 작성하고, 누구는 홍보용 배너나 로고 등의 이미지를 만들어내는 식으로 말입니다.

여럿이서 하나의 사이트를 만들다 보니 작업 방식도 다 다르고, 폰트나 여백의 크기 등이 중구난방이라 맞춰가는 과정이 필요했는데, 이것도 팀 프로젝트를 하지 않았다면 몰랐을 것 같습니다. 실제로는 어떨지 모르겠지만, 회사에 들어가 프로젝트를 맡으면 이런 식으로 일하게 될까 조금이나마 상상할 수 있어 좋았습니다. 다음에 또 팀 프로젝트를 한다면 어떠한 순서대로 일을 처리해야 할지, 어떤 문제가 있을지 조금은 예상할 수 있게 된 것 같습니다.

감사합니다!

| 평가 및 소감

이번 팀 프로젝트는 다섯 번째 작업이자, 처음으로 경험한 팀 단위 프로젝트였습니다. 시작 전에는 걱정과 부담이 컸지만, 막상 진행해보니 예상보다 의견 충돌도 적었고, 각자의 역할에 충실하면서도 유기적으로 협업이 이루어져 좋은 인상을 받았습니다.
특히 팀워크의 중요성과 효율적인 소통 방식, 협업을 통해 얻어지는 시너지 효과를 직접 체감할 수 있었던 점은 가장 큰 수확이었습니다. 디자이너와의 의견 조율 과정에서 실무와 커뮤니케이션의 균형 감각도 배울 수 있었고, 일정 관리나 작업 분배 측면에서도 협업의 흐름을 이해하게 되었습니다.
완성도 면에서는 아직 아쉬운 점이 있지만, 하나의 결과물을 팀과 함께 끝까지 완성해냈다는 점에서 큰 의미가 있었고, 그 과정 자체가 소중한 배움이 되었다고 느꼈습니다. 이번 팀 프로젝트로 쌓은 경험은 앞으로 진행할 다양한 프로젝트들에도 긍정적인 영향을 줄 것이라 기대하고 있으며, 부족했던 점들을 보완해 더욱 성장한 모습으로 다음 작업을 이어가고 싶습니다.

| 평가 및 소감

이번 프로젝트에서는 디자인 전반을 주로 담당하며,
팀원들과 자연스럽게 의견이 이어지도록 조율하며 작업을 진행하였습니다.
디자인 관련 의견을 공유할 때 팀원들이 긍정적으로 반응하고 신뢰해 주는 모습을 보며 감사했습니다.
협업 속에서 각자의 역할을 나누고 조화롭게 완성해 가는 과정 자체가 매우 의미 있었습니다.
처음 팀 프로젝트를 경험하면서 원활한 소통과 조율의 중요성을 크게 깨달았고,
효율적인 역할 수행을 위해 예상보다 많은 대화와 협의가 필요하다는 것도 알게 되었습니다.
 그 과정에서 자연스럽게 협업 능력과 책임감도 함께 성장할 수 있었습니다.
또한 디자인이나 기능에 관한 의견 차이를 서로 존중하며 해결책을 찾아가는 값진 경험을 하였으며,
 이번 프로젝트를 통해 기술적인 성장뿐 아니라 팀워크와 커뮤니케이션 능력의 중요성도
깊이 체감할 수 있는 시간이 되었다고 생각합니다.

Thank you!

team 3 | 한길수 안세홍 유희원 김하엘

https://gs1020.github.io/gs1020team3/