1. AboutPage.js

```jsx
import React from "react";
import NavBar from "../components/NavBar";
import { Container } from "reactstrap";
import { Jumbotron } from "reactstrap";

const AboutPage = (props) => {
  return (
    <>
      <NavBar />
      <div
        style={{
          display: "flex",
          width: "100%",
          height: "100%",
          textAlign: "center",
          alignItems: "center",
        }}
      >
        <Container>
          <Jumbotron style={{ backgroundColor: "#fff" }}>
            <h1 className="display-3">Enjoy your meal!</h1>
            <br />
            <p className="lead">
            <mark><em>고민사거리</em></mark>는 숭실대학생들의 식사고민을 덜어주기 위해 제작된
              웹사이트입니다.
              <br />
              본래 "고민사거리"라는 단어는 숭실대 근처에 있는 식당가를 의미합니다.
              <br />
              학생들이 사거리에 서서 무엇을 먹을지 고민한다고 하여, "고민사거리"라는 이름을 붙였습니다.
              <br />
              본 웹페이지 <mark><em>고민사거리</em></mark>는 숭실대 일대 및 일명 "고민사거리"에 있는 식
              카테고리화하였습니다.
              <br />
              친구들과 메뉴를 정할 때 랜덤추천을 사용해보세요!
            </p>
            <br />
            <hr className="my-2" />
            <br />
            <p>
              식당추가 등 문의사항은 언제든지 환영입니다! choijh9182@naver.com로 연락주세요
              <br />20180402 최주형
              <br />20170401 최정민
              <br />20183423 최한나
              <br />20170403 최혜원
            </p>
            {/* <br/><br/><br/> */}
          </Jumbotron>
        </Container>
      </div>
    </>
  );
};

export default AboutPage;
```

- components 디렉토리의 NavBar.js 응용 (창 맨 위에 네비게이션 바 삽입)

- 페이지에 출력할 텍스트

- Container와 JumboTron 활용

- 배경: 하양

1.1 NavBar.js

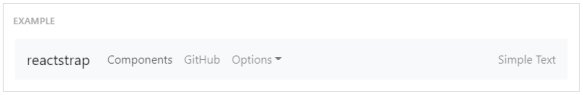
```jsx
import React, { useState } from "react";
import { Container, NavbarText } from "reactstrap";
import {
  Collapse,
  Navbar,
  NavbarToggler,
  NavbarBrand,
  Nav,
  NavItem,
  NavLink,
} from "reactstrap";
```

- NavBar.js가 사용하는 라이브러리의 파일 형식

- reactstrap의 Navbar 형식을 참조하여 구현

1.1.1 Reactstrap의 Navbar

Navbar




```
import React, { useState } from 'react';
import {
  Collapse,
  Navbar,
  NavbarToggler,
  NavbarBrand,
  Nav,
  NavItem,
  NavLink,
  UncontrolledDropdown,
  DropdownToggle,
  DropdownMenu,
  DropdownItem,
  NavbarText
} from 'reactstrap';

const Example = (props) => {
  const [isOpen, setIsOpen] = useState(false);

  const toggle = () => setIsOpen(!isOpen);

  return (
    <div>
      <Navbar color="light" light expand="md">
        <NavbarBrand href="/">reactstrap</NavbarBrand>
        <NavbarToggler onClick={toggle} />
        <Collapse isOpen={isOpen} navbar>
          <Nav className="mr-auto" navbar>
            <NavItem>
              <NavLink href="/components/">Components</NavLink>
            </NavItem>
            <NavItem>
              <NavLink href="https://github.com/reactstrap/reactstrap">GitHub</NavLink>
            </NavItem>
            <UncontrolledDropdown nav inNavbar>
              <DropdownToggle nav caret>
                Options
              </DropdownToggle>
              <DropdownMenu right>
                <DropdownItem>
                  Option 1
                </DropdownItem>
                <DropdownItem>
                  Option 2
                </DropdownItem>
                <DropdownItem divider />
```

- Navbar 기본 형태 및 예시 코드

- 전체 바는 〈Navbar〉로 감싸져 있으며, 메인으로 연결되는 리다이렉트는 〈NavbarBrand〉, 〈NavarTrigger〉로 구성

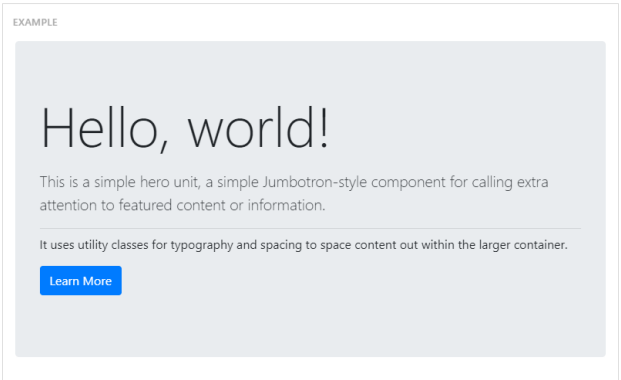
- 각 리다이렉트 성분은 〈NavItem〉과 〈NavLink〉로 구성

- Drop 성분으로 index를 넣을 수도 있으나, 본 프로젝트에선 활용하지 않음

1.2 Container & Jumbotron - reactstrap 라이브러리의 프론트엔드 스타일

Container

Jumbotron



- Container는 JumboTron을 담는 박스 역할

- JumboTron은 두 개의 클래스로 이루어져 있으며, 중간선으로 두 클래스가 나뉘어지도록 디자인되어 있음(AboutPage.js코드 참조)

2. LandingPage.js

```
import React, { useState, useEffect } from "react";
import NavBar from "../components/NavBar";
import LandingMap from "../components/LandingMap";
import { CustomInput } from "reactstrap";
import { Container, Row, Col, Button, Jumbotron } from "reactstrap";
import axios from "axios";
import Loading from "../components/Loading";
import MealCard from "../components/MealCard";

const LandingPage = (props) => {
  const [datas, setDatas] = useState([]);
  const [filteredDatas, setFilteredDatas] = useState([]);
  const [RandomCards, setRandomCards] = useState([]);
  const [all, setAll] = useState(false);
  const [Kfood, setKfood] = useState(false);
  const [Wfood, setWfood] = useState(false);
  const [Cfood, setCfood] = useState(false);
  const [Jfood, setJfood] = useState(false);
  const [meat, setMeat] = useState(false);
  const [snackfood, setSnackfood] = useState(false);
  const [pub, setPub] = useState(false);
  const [fastfood, setFastfood] = useState(false);
  const [cafe, setCafe] = useState(false);
  const [etc, setEtc] = useState(false);
  const [isRandom, setIsRandom] = useState(0);
```

- 필요한 외부 파일형식 및 변수 선언 (line 112 까지 변수 및 식당 종류에 대한 정의문)
- component 디렉토리의 *NavBar.js, LandingMap.js, Loading.js, MealCard.js 활용
- reactstrap 라이브러리 CustomInput, *Container,Row, Col, Button, *JumboTron 활용

*: 1에서 설명

```
  const randomHandler = () => {
    if (filteredDatas.length === 0) {
      alert("메뉴를 선택하세요");
      return;
    } else {
      setIsRandom(1);
      setTimeout(() => {
        let x = getRandomInt(0, filteredDatas.length);
        let y = -1;
        while (1) {
          y = getRandomInt(0, filteredDatas.length);
          if (x !== y) break;
        }
        setRandomCards([filteredDatas[x], filteredDatas[y]]);
        setIsRandom(2);
      }, 2500);
    }
  };
```

- 랜덤선택을 해 주는 기능 코드
- 아무 데이터도 선택이 되어있지 않다면 동작하지 않음

```
  const getRandomInt = (min, max) => {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min)) + min; //최댓값은 제외, 최솟값은 포함
  };

  return (
    <>
      <NavBar />
      <Container
        style={{
          paddingTop: "1.5rem",
        }}
      >
        <div style={{ display: "flex", width: "100%", height: "100%" }}>
          <Container>
            <Jumbotron
              style={{
                backgroundColor: "#fff",
                paddingTop: "1.5rem",
                boxShadow:
                  "0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19)",
              }}
```

- 랜덤 박스 코드, Container & JumboTron 으로 구현
- 각 메뉴 텍스트를 작성할 박스는 Row & Col로 구현

```
                <Row xs="1" sm="1" md="2">
                  <Col style={{}}>
                    <h3 className="text-center">
                      <span className="font-weight-bold">랜덤추천 : </span>메뉴를
                      선택하세요
                    </h3>
                    <span
                      style={{
                        paddingLeft: "0.5rem",
                      }}
                    >
                      {/* <Button color="warning">Random!</Button> */}
                    </span>
```

```
170                 <Container>
171                   <Row>
172                     <Col>
173                       <CustomInput
174                         type="switch"
175                         id="all"
176                         label="전체"
177                         checked={all}
178                         onChange={() => setAll(!all)}
179                       />
180                     </Col>
181                   </Row>
```

```
182                   <Row xs="3" sm="3" md="4">
183                     <Col>
184                       <CustomInput
185                         type="checkbox"
186                         id="Kfood"
187                         label="한식"
188                         checked={Kfood}
189                         onChange={() => setKfood(!Kfood)}
190                       />
191                     </Col>
```

- CustomInput으로 각 메뉴를 선택할 수 있는 체크박스 구현 (line 283까지 반복)

```
285                   <Button
286                     onClick={randomHandler}
287                     size="lg"
288                     style={{
289                       margin: "0 auto",
290                     }}
291                     color="danger"
292                   >
293                     Random!
294                   </Button>
```

- 랜덤 추첨 동작을 실행할 버튼 구현 코드

- Button으로 구현

```
298                 <Col>
299                   {isRandom === 0 ? (
300                     ""
301                   ) : isRandom === 1 ? (
302                     <span style={{ padding: "1.5rem" }}>
303                       <Loading value="추첨중.." />
304                     </span>
305                   ) : (
306                     <Container>
307                       <Row>
308                         {RandomCards.map((data, index) => (
309                           <Col>
310                             <MealCard
311                               key={index}
312                               id={data.id}
313                               name={data.name}
314                               address={data.address}
315                               latitude={data.latitude}
316                               longitude={data.longitude}
```

- 추첨을 진행할 때 로딩 중인 회전 이미지 구현

- 추첨 후 해당하는 식당의 MealCard 정보를 받아와 출력

2.1 LandingMap.js

- 카카오 맵 API를 통해 지도를 출력하는 코드

2.2 Loading.js

- 페이지 기능 중 랜덤으로 식당을 뽑는 기능이 있다. 이 때 추첨 시 로딩 동작을 표현하는 코드

- reactstrap 라이브러리에서 Spinner를 활용

2.3 MealCard.js

- 각 식당의 정보를 표기하는 박스 코드

2.4 reactstrap의 파일들 (CustomInput & Row & Col & Button)

- Row, Col은 Container를 기본형으로 하는 텍스트 박스의 역할

  Row는 일정한 크기로 나뉘어져 있음

  Col은 다양한 크기 활용 가능

- CustomInput은 체크박스 및 스위치를 보유함(본문에선 랜덤선택을 위해 메뉴를 선택할 때 활용)

- Button은 말 그대로 버튼(본문에서는 랜덤 추첨 동작 수행버튼으로 활용)

2.5 axios

- http 통신 라이브러리, 본 코드에선 axios로 모든 식당의 정보들을 받아옴

3. MenuPage.js

- 사용자가 선택한 메뉴에 해당하는 모든 식당의 리스트(with MealCard)를 보여주는 페이지

```
import React, { useState, useEffect } from "react";
import NavBar from "../components/NavBar";
import MealCard from "../components/MealCard";
import { CustomInput } from "reactstrap";
import { Container, Row, Col } from "reactstrap";
import axios from "axios";
import Loading from "../components/Loading";

const MenuPage = (props) => {
  const [datas, setDatas] = useState([]);
  const [filteredDatas, setFilteredDatas] = useState([]);
  const [isLoading, setIsLoading] = useState(false);
  const [all, setAll] = useState(false);
  const [Kfood, setKfood] = useState(false);
  const [Wfood, setWfood] = useState(false);
  const [Cfood, setCfood] = useState(false);
  const [Jfood, setJfood] = useState(false);
  const [meat, setMeat] = useState(false);
  const [snackfood, setSnackfood] = useState(false);
  const [pub, setPub] = useState(false);
  const [fastfood, setFastfood] = useState(false);
  const [cafe, setCafe] = useState(false);
  const [etc, setEtc] = useState(false);
```

- 필요한 외부 파일형식 및 변수 선언
(line 115까지 변수 및 식당 종류에 대한 정의문)

- component 디렉토리의 NavBar.js, Loading.js, MealCard.js 활용

- reactstrap 라이브러리 CustomInput, Container, Row, Col 활용

(모두 위에서 활용한 파일 형식들이므로 추가설명X)

```
  return (
    <>
      <NavBar />
      <Container
        style={{
          paddingTop: "1.5rem",
        }}
      >
        <Row>
          <Col>
            <CustomInput
              type="switch"
              id="all"
              label="전체"
              checked={all}
              onChange={() => setAll(!all)}
            />
          </Col>
        </Row>
        <Row xs="3" sm="3" md="5">
          <Col>
            <CustomInput
              type="checkbox"
              id="Kfood"
              label="한식"
              checked={Kfood}
              onChange={() => setKfood(!Kfood)}
            />
          </Col>
```

- LandingPage에서 사용했던 각 메뉴 체크박스 코드를 그대로 사용

```
      <Container></Container>
      {isLoading ? (
        <Container style={{ paddingTop: "1.2rem" }}>
          <Row xs="2" sm="2" md="4">
            {filteredDatas.map((data, index) => (
              <Col>
                <MealCard
                  key={data.id}
                  id={data.id}
                  name={data.name}
                  address={data.address}
                  latitude={data.latitude}
                  longitude={data.longitude}
                  type={data.type}
                  menu={data.menu}
                  img={data.img}
                  img_source={data.img_source}
                />
              </Col>
            ))}
          </Row>
        </Container>
```

- 선택한 메뉴에 해당하는 모든 식당의 데이터를 받아 MealCard형식으로 화면에 출력

4. MypickPage.js

- 현재 로그인한 유저가 선택한 mypick들을 보여주는 페이지

```
import React, { useState, useEffect } from 'react';
import NavBar from '../components/NavBar';
import LoginLink from '../components/LoginLink';
import UserCards from '../components/UserCards';
import { Container } from 'reactstrap';

```

- components 디렉토리의 *NavBar.js, LoginLink.js, UserCards.js 사용

- reactstrap 라이브러리의 *Container 사용 (*: 위에 있던 파일들이므로 추가설명X)

```
const MypickPage = () => {
    const [isLogin, setIsLogin] = useState(false);
    const [userName, setUserName] = useState('');
    const authApi = () => {
        const user = JSON.parse(localStorage.getItem('user'));
        return fetch('/api/auth', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'authorization': user,
                'Accept': 'application/json'

            }
        }).then(response => response.json())
        .then(result => {
            if(result.message === 'valid token') {
                setIsLogin(true);
                setUserName(result.username);
            } else if(result.message === 'expired token') {
                // alert('토큰이 만료되었습니다. 로그인 해주세요.');
                setIsLogin(false);
            } else if(result.message === 'invalid token') {
                setIsLogin(false);
            }
        });
    }

```

- 로그인 상태를 확인하기 위해 json 및 localstorage에서 토큰을 받아 옴

- 토큰 속 메시지를 통해 현재 유저의 로그인/비로그인 상태 확인

```
    useEffect(() => {
        authApi();
    }, [isLogin]);

    return (
        <>
        <Container>
            <NavBar/>
            {
            isLogin ?
            (<>
            <UserCards username={userName} isLogin={setIsLogin}/>
            </>)
            :
            (<>
            <LoginLink/>
            </>)
            }
        </Container>
        </>
    )
}
```

- 로그인이 되어 있다면? = UserCard.js 출력

- 비로그인 상태라면? = LoginLink.js 출력

4.1 LoginLink.js

```
import React from 'react'
import {Container, Jumbotron, Button} from 'reactstrap';
const LoginLink = () => {
    return (
        <>
            <div style={{paddingTop:'8rem', 'display':'flex', 'width':'100%', "
                <Container style={{}}>
                    <Jumbotron
                    style={{'backgroundColor':'#fff'}}
                    >
                        <h1 className="display-3">My pick</h1><br/>
                        <p className="lead">로그인이 필요합니다.
                        </p>
                        <br/>
                        <hr className="my-2" />
                        <br/>
                        <Button color="link" href='/signin'>로그인</Button>
                        <Button color="link" href='/signup'>회원가입</Button>
                    </Jumbotron>
                </Container>
            </div>
        </>
    )
}

export default LoginLink;
```

- MypickPage에 들어갔을 때, 비로그인 상태라면 로그인이 필요함을 보여주는 페이지

- Button을 통해 로그인 또는 회원가입 창으로 넘어갈 수 있도록 함

4.2 UserCards.js

- 로그인이 되어 있을 때 유저가 픽한 식당들을 보여주는 페이지 코드

```
import React, { useState, useEffect } from "react";
import { Container, Row, Col, Button } from "reactstrap";
import PickedCard from "../components/PickedCard";

```

- components의 PickedCard.js, reactstrap 디렉토리에서 Container, Row, Col, Button 사용

```
const UserCards = (props) => {
  const [username, setUsername] = useState("User");
  const [picks, setPicks] = useState([]);

  const authApi = () => {
    const user = JSON.parse(localStorage.getItem("user"));

    return fetch("/api/mypicks", {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
        authorization: user,
        Accept: "application/json",
      },
    })
      .then((response) => response.json())
      .then((result) => {
        setPicks(result.datas);
      });
  };
```

- MypickPage에서의 로그인 정보를 받아오는 부분의 코드를 그대로 차용(localStorage 사용)

- mypicks에 저장되어 있는 유저가 픽 한 식당의 정보들을 받는 코드 추가적으로 구현

```
26      useEffect(() => {
27        setUsername(props.username);
28        authApi();
29      }, [username, props]);
30
31      const LogoutHandler = (e) => {
32        e.preventDefault();
33        localStorage.removeItem("user");
34        props.isLogin(false);
35        return;
36      };
```

- 로그아웃 동작 함수 구현

- localStorage에서 해당 유저의 데이터를 제거하는 것으로 구현

```
38    return (
39      <>
40        <h1 style={{ paddingTop: "3rem" }} className="text-center">
41          <div className="font-weight-bold">{username}'s Pick</div>
42          <Button onClick={LogoutHandler} color="link" className="float-right">
43            <h4>Logout</h4>
44          </Button>
45        </h1>
46        <br />
47        <hr className="my-2" />
48        <br />
49        <Container style={{ paddingTop: "1.2rem" }}>
50          <Row xs="2" sm="2" md="4">
51            {picks &&
52              picks.map((data, index) => (
53                <Col key={index}>
54                  <PickedCard
55                    key={index}
56                    id={data.id}
57                    name={data.name}
58                    address={data.address}
59                    latitude={data.latitude}
60                    longitude={data.longitude}
61                    type={data.type}
62                    menu={data.menu}
```

- 로그아웃 버튼 구현 (위의 로그아웃 동작 실행)

- PickedCard로 유저가 픽한 식당의 리스트 출력

4.2.1 PickedCard.js

- MealCard.js 와 구조 상 차이점은 없음

- MealCard에서는 mypick으로 저장할 수 있었던 버튼이 PickedCard에서는 mypick에서 제거할 수 있는 버튼으로 바뀐 점이 유일한 차이

```
43              }).then(response => response.json())
44              .then(result => {
45                if(result.message === 'delete success') {
46                  alert('삭제 완료');
47                  window.location.href = "/mypick";
48                } else if(result.message === 'delete error') {
49                  alert('delete error');
50                } else {
51                  alert('error');
52                }
53              });
54              } else {
55                alert('토큰이 만료되었습니다.');
56                window.location.href = "/mypick";
57              }
58          });
59      }
60
61
62      const deleteHandler = (e) => {
63        e.preventDefault();
64        authApi();
65      }
```

- mypick 삭제하는 코드 파트

- 서버에서 보내주는 메시지에 따라 동작이 되었는지 확인

5. SigninPage.js

- 로그인 페이지

```
import React, { useState } from "react";
import cookie from "react-cookies";
import { Button, Form, FormGroup, Label, Input } from "reactstrap";
import { FacebookLoginButton } from "react-social-login-buttons";
```

- cookie를 사용하여 이전에 로그인했던 아이디의 기록을 입력창에 출력할 수 있게 함

- Facebook 연동 로그인 구현 (FacebookLoginButton)

- 그 외의 요소들: Form (FormGroup, Label, Input 모두 Form 형식에 포함)

```
const SigninPage = (props) => {
  const [userName, setUserName] = useState("");
  const [userPw, setUserPw] = useState("");

  const signinApi = (user) => {
    return fetch("/api/signin", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
      },
      body: JSON.stringify(user),
    }).then((response) => response.json());
  };
```

- 서버 및 데이터베이스 연동을 위한 코드

```
const handleSubmit = async (e) => {
  e.preventDefault();
  if (!userName || !userPw) {
    return;
  }
  try {
    const response = await signinApi({
      username: userName,
      password: userPw,
    });

    if (response.message === "Token issue") {
      localStorage.setItem("user", JSON.stringify(response.token));
      const expires = new Date();
      expires.setDate(expires.getDate() + 60);
      cookie.save("username", response.username, {
        expires,
      });
      props.history.push("/mypick");
    } else if (response.message === "user does not exist") {
      alert("User does not exist");
      props.history.push("/signin");
      setUserName("");
      setUserPw("");
```

- localStorage에 접근하여 DB에 있는 회원 정보들과 현재 페이지에서 유저가 입력한 정보를 비교

- 맞는 정보가 입력되었다면 my pick page 로 전환

- 존재하지 않는 정보가 입력되었다면 그에 맞는 메시지 전달 및 Signup 화면 초기화

```
    } else if (response.message === "invalid password") {
      alert("Invalid Password");
      props.history.push("/signin");
      setUserName("");
      setUserPw("");
    } else if (response.message === "server error") {
      alert("Server Error");
      props.history.push("/signin");
      setUserName("");
      setUserPw("");
    }
  } catch (err) {
    alert("Signin Failed");
    setUserName("");
    setUserPw("");
    props.history.push("/signin");
  }
};
```

- 틀린 pw/서버 에러 등의 문제 발생마다 해당하는 메시지를 리턴하고 페이지 초기화 동작

```
<Form
  style={{
    width: "100%",
    maxWidth: "330px",
    padding: "15px",
    margin: "auto",
    height: "100%",
  }}
>
  <a href="/" style={{ color: "#000000", textDecoration: "none" }}>
    <h1 className="text-center">
      <span className="font-weight-bold">고민사거리</span>.com
    </h1>
  </a>
  <h2 className="text-center">
    <br />
    Sign In
  </h2>
  <FormGroup>
    <Label>Username</Label>
    <Input
      required="required"
      value={userName}
      onChange={onChangeUsername}
      type="name"
      placeholder="Enter your name"
    ></Input>
```

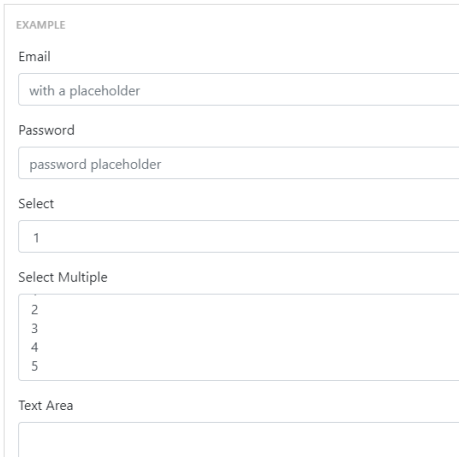
-Id와 Password 입력란 구현 코드

-Form 형식에 맞게 코딩 (5.1참조)

5.1 Form

- 아이디/이메일 및 패스워드를 입력하는 란이 포함된 디자인 형태

Form



- 디자인 형태

```
import React from 'react';
import { Button, Form, FormGroup, Label, Input, FormText } from 'reactstrap';

const Example = (props) => {
  return (
    <Form>
      <FormGroup>
        <Label for="exampleEmail">Email</Label>
        <Input type="email" name="email" id="exampleEmail" placeholder="with a
      </FormGroup>
      <FormGroup>
        <Label for="examplePassword">Password</Label>
        <Input type="password" name="password" id="examplePassword" placeholde
      </FormGroup>
      <FormGroup>
        <Label for="exampleSelect">Select</Label>
        <Input type="select" name="select" id="exampleSelect">
          <option>1</option>
          <option>2</option>
          <option>3</option>
          <option>4</option>
          <option>5</option>
```

- 코드 형식

6. SignupPage.js

- 회원가입 페이지

- 외부 파일은 로그인 페이지에서 사용한 reactstrap 파일들과 같음

```
const SigninPage = (props) => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [passwordCheck, setPasswordCheck] = useState("");

  const onSubmit = (e) => {
    e.preventDefault();

    // 비밀번호가 6자이상인지 검사
    if (password.length < 6 || passwordCheck.length < 6) {
      return alert("비밀번호는 6자이상이어야 합니다.");
    }

    // 비밀번호와 비밀번호 체크가 다를 경우를 검사
    if (password !== passwordCheck) {
      return alert("비밀번호가 일치하지 않습니다.");
    }

    const signupInfo = {
      username: username,
      password: password,
    };
```

- username과 password, passwordcheck 변수 선언
- 회원가입 시 비밀번호의 길이 체크(6자 이상)와 password와 passwordcheck간 일치 체크 코드 구현

```
    const signup_info = {
      method: "POST",
      body: JSON.stringify(signupInfo),
      headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
      },
    };

    if (username && password) {
      fetch("/api/signup", signup_info)
        .then((response) => response.json())
        .then((json) => {
          if (json.message === "success") {
            alert("회원가입에 성공했습니다.");
            props.history.push("/signin");
          } else if (json.message === "user exist") {
            alert("이미 존재하는 유저입니다");
            setUsername("");
            setPassword("");
          } else {
            alert("회원가입에 실패했습니다.");
            setUsername("");
            setPassword("");
          }
        });
    }
  };
```

- 서버 및 데이터베이스(api/signup) 와의 연동으로 회원가입 id/pw 정보가 정상적으로 입력되었을 때 경우에 따라 성공/존재하는 유저/실패로 경우 나눔
- 서버에서 받아온 메시지에 따라 다음 동작 수행

```
  const onChangeUsername = (e) => {
    setUsername(e.target.value);
  };

  const onChangePassword = (e) => {
    setPassword(e.target.value);
  };

  const onChangePasswordCheck = (e) => {
    setPasswordCheck(e.target.value);
  };
```

- 각 정보 입력란에 넣어줄 코드, 위의 Submit 파트와 연동

```
return (
  <>
    <Form
      onSubmit={onSubmit}
      style={{
        width: "100%",
        maxWidth: "330px",
        padding: "15px",
        margin: "auto",
        height: "100%",
      }}
    >
      <a href="/" style={{ color: "#000000", textDecoration: "none" }}>
        <h1 className="text-center">
          <span className="font-weight-bold">고민사거리</span>.com
        </h1>
      </a>
      <h2 className="text-center">
        <br />
        Sign Up
      </h2>
```

- 페이지 디자인 파트, “고민사거리.com”에는 메인 화면으로 리다이렉트 걸어놓음

```
<FormGroup>
  <Label>Username</Label>
  <Input
    required="required"
    value={username}
    onChange={onChangeUsername}
    name="username"
    type="name"
    placeholder="Enter your name"
  ></Input>
</FormGroup>
<FormGroup>
  <Label>Password</Label>
  <Input
    required="required"
    type="password"
    value={password}
    onChange={onChangePassword}
```

- Form을 활용하여 회원가입 id/pw를 입력하는 Input 파트 구현