풀스택 자바(JAVA)웹개발자(프론트엔드&백엔드)_A

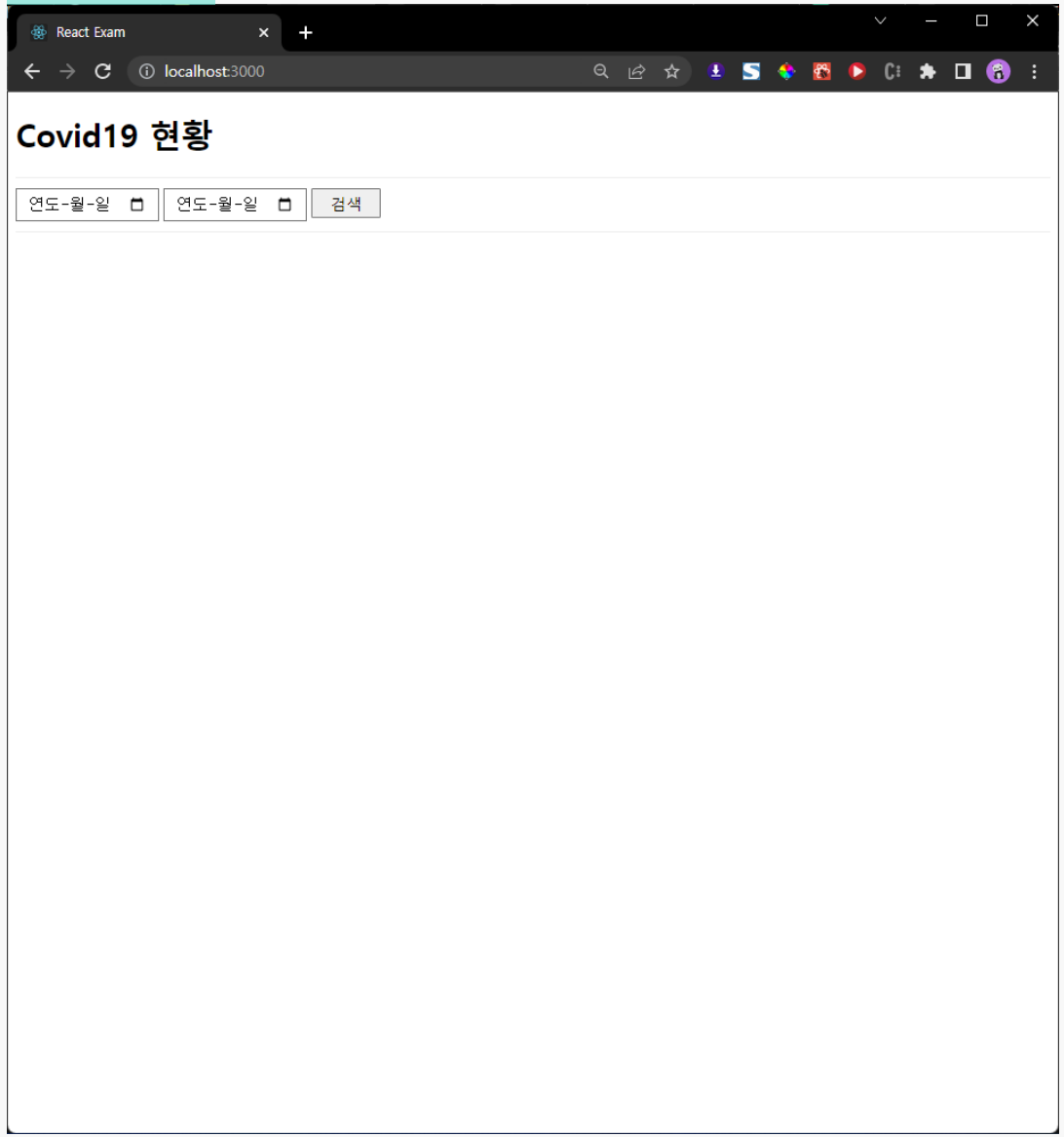
React 프로그래밍 보고서

박정모

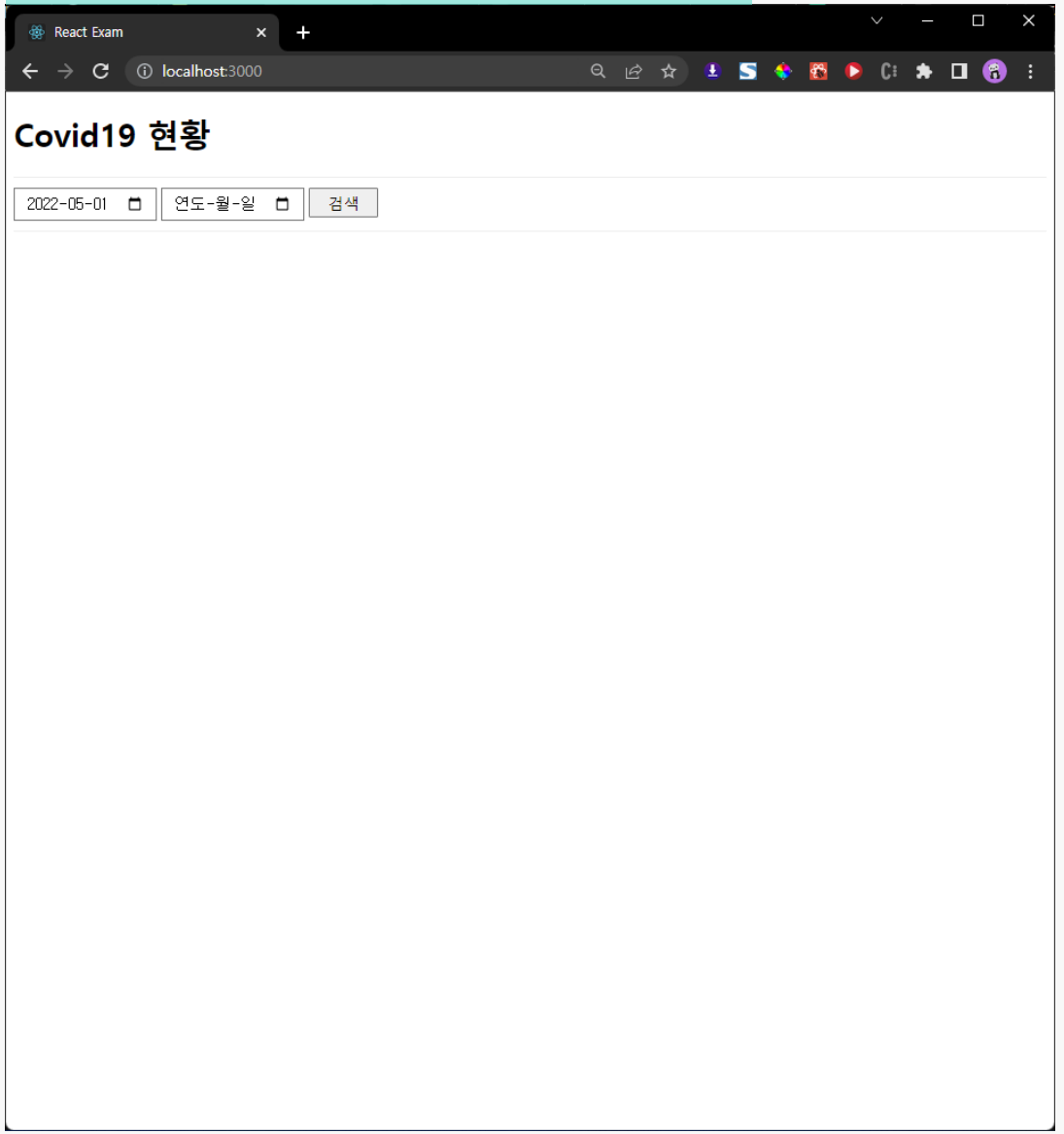
00 목차

01 구현 결과 스크린샷

첫 페이지



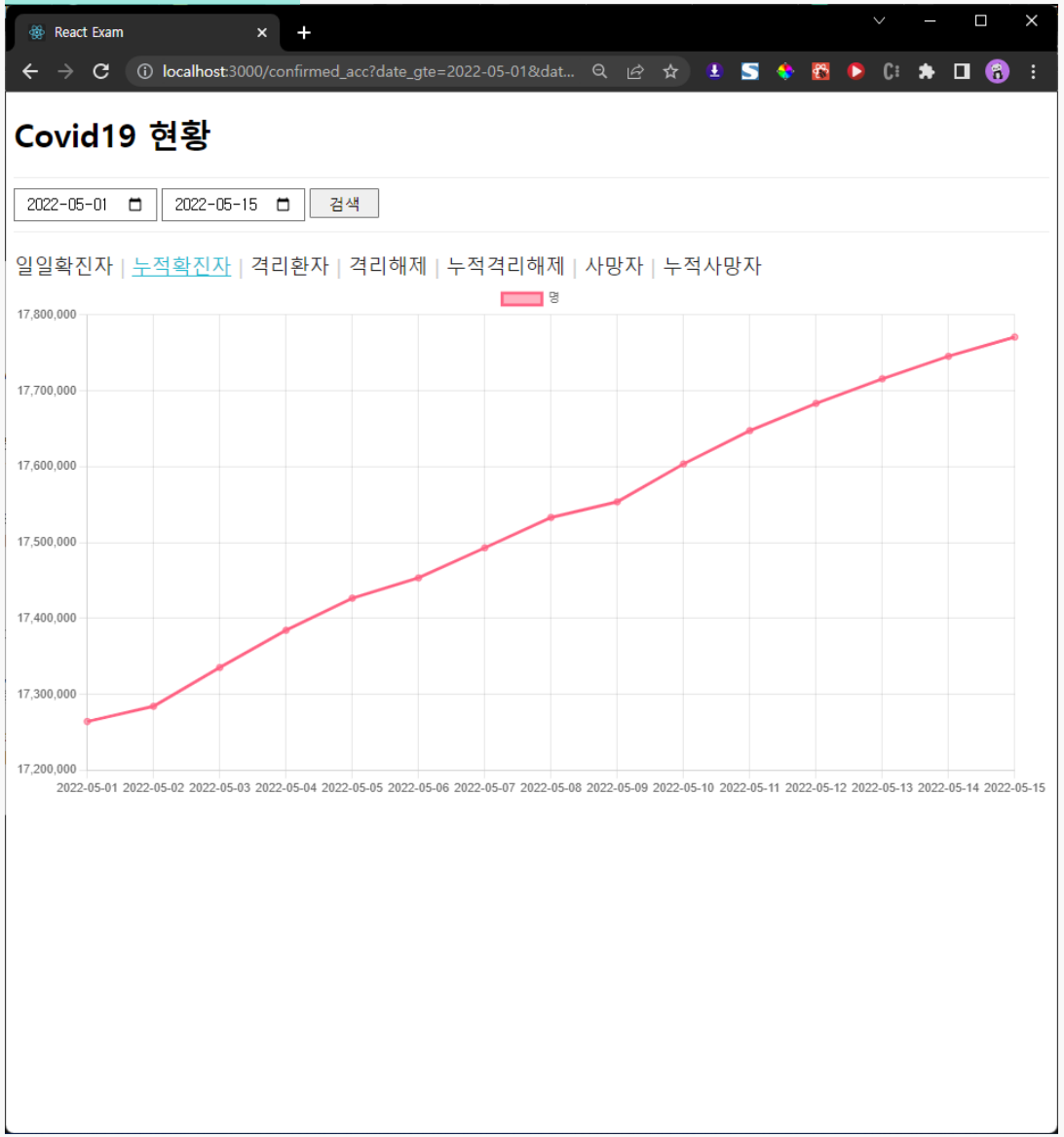
날짜를 하나만 입력하고 검색을 했을 때



01 구현 결과 스크린샷

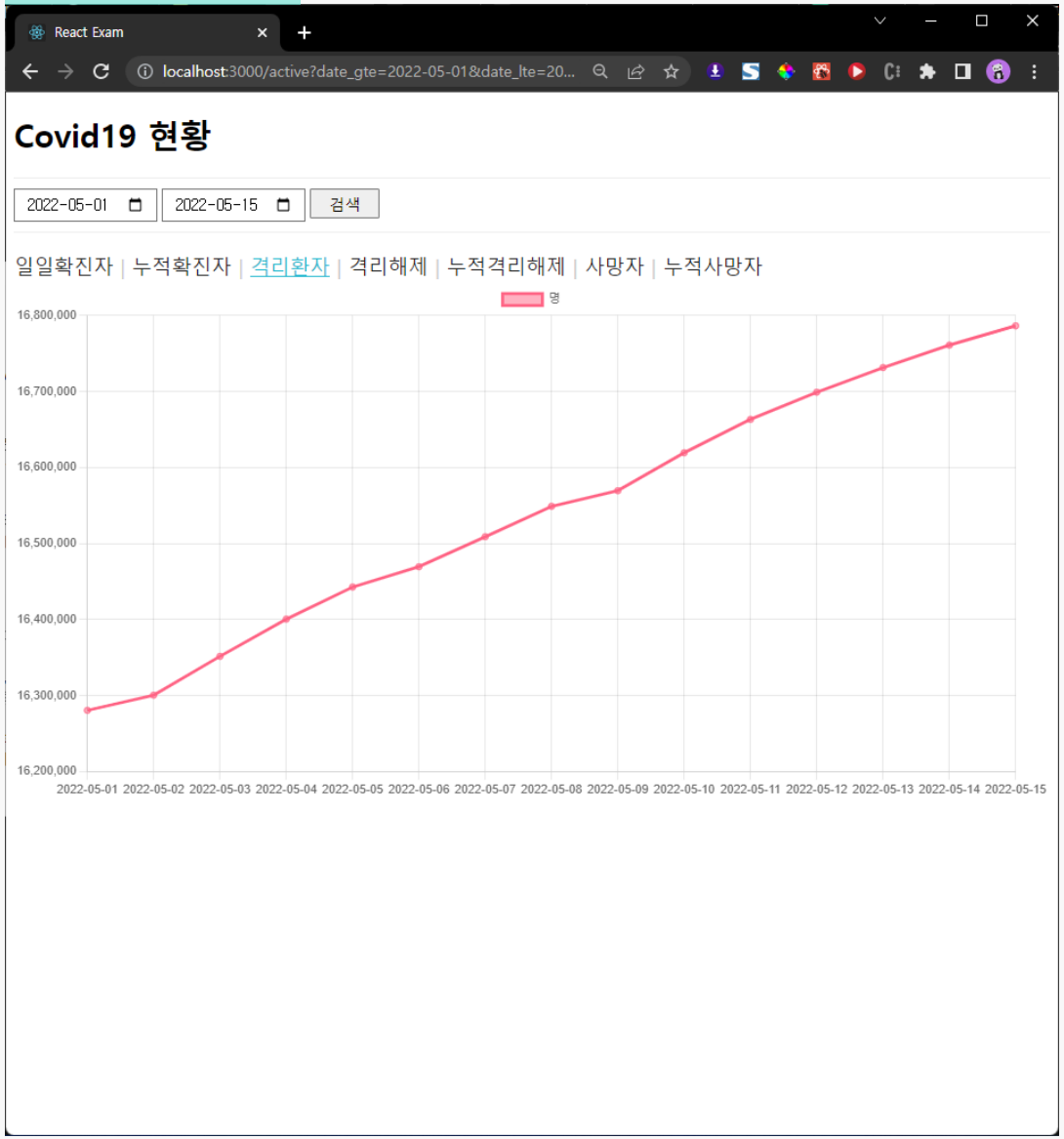
각 링크별 결과

각 링크별 결과



01 구현 결과 스크린샷

각 링크별 결과

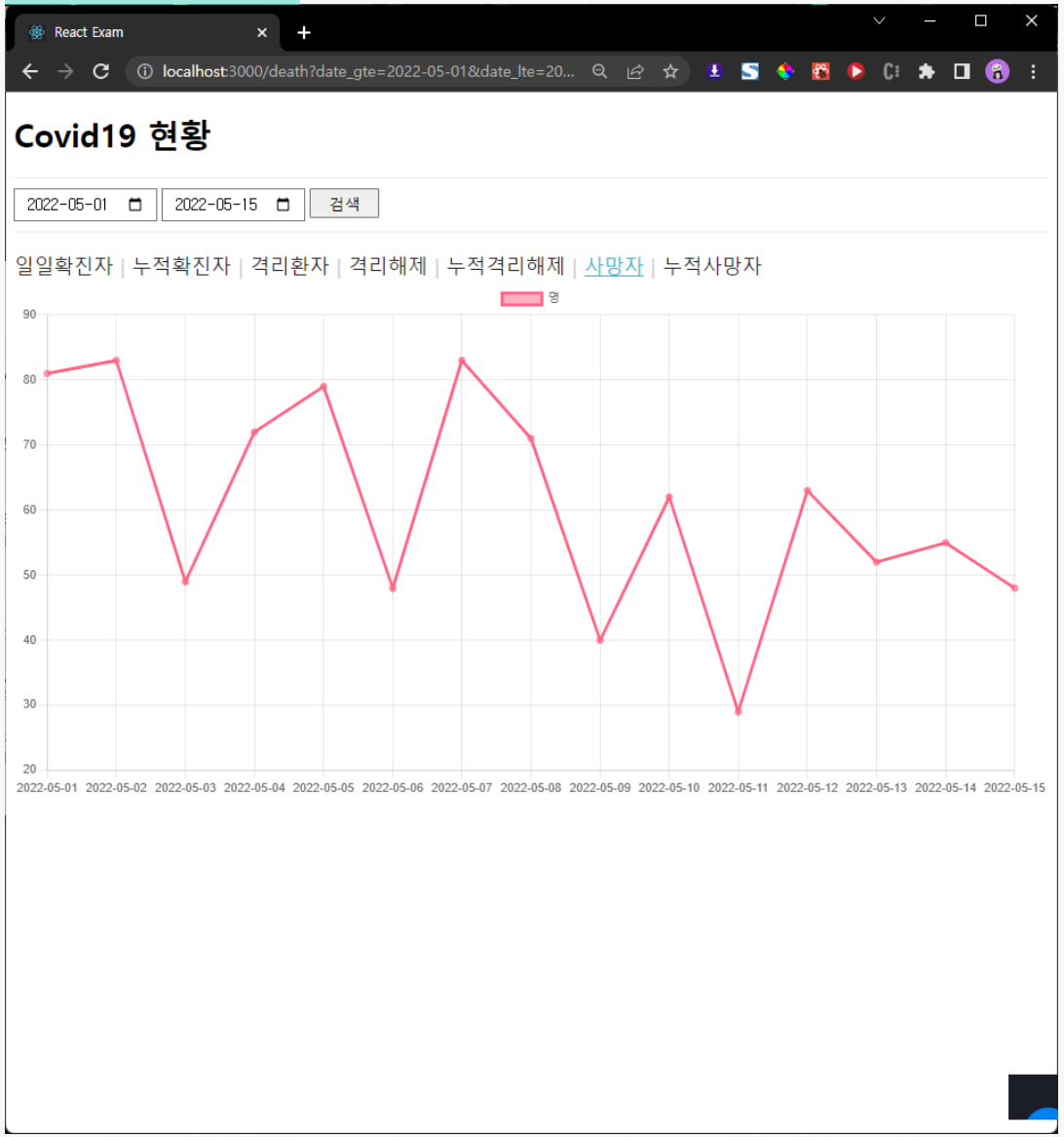


각 링크별 결과

01 구현 결과 스크린샷

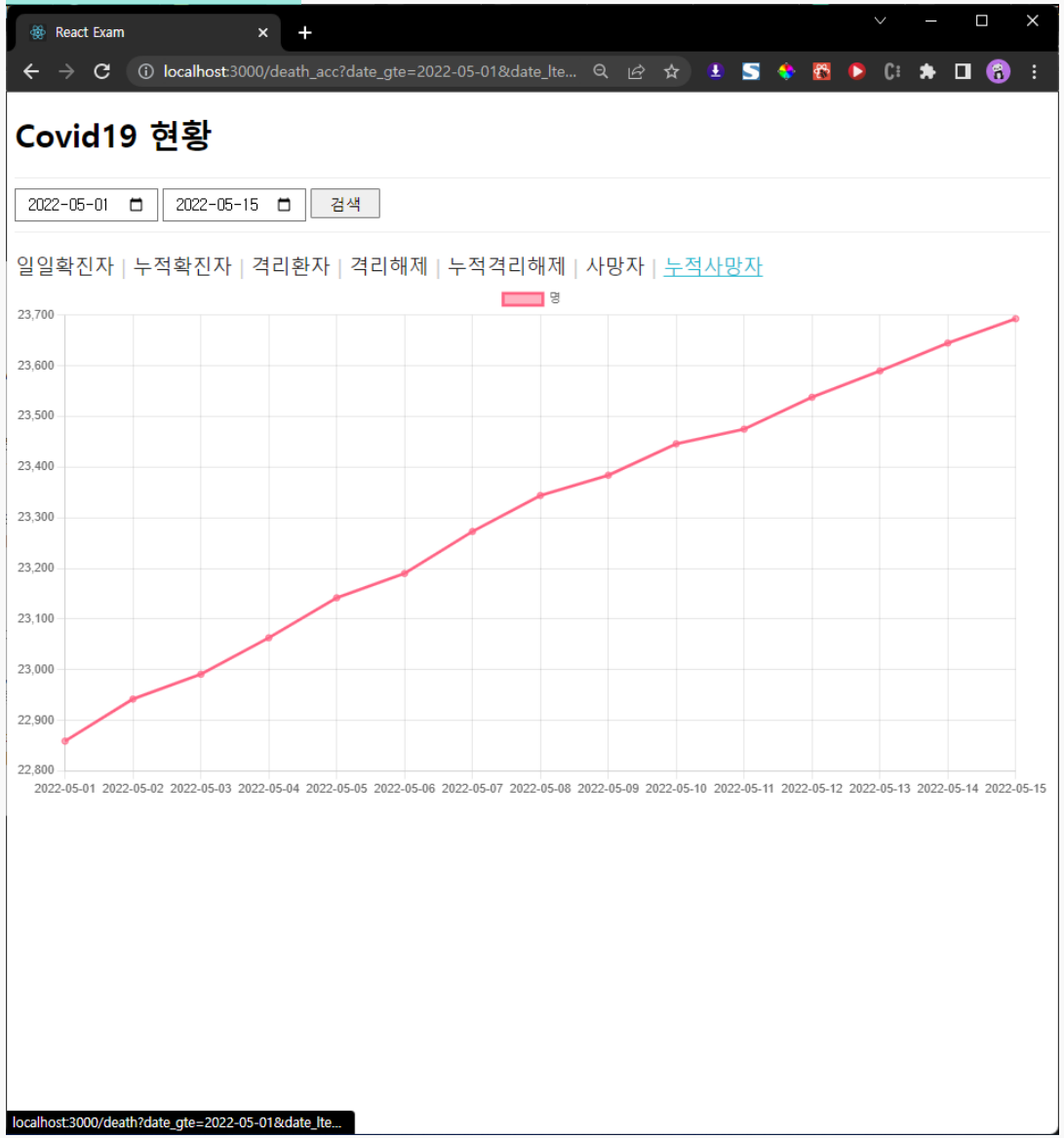
각 링크별 결과

각 링크별 결과

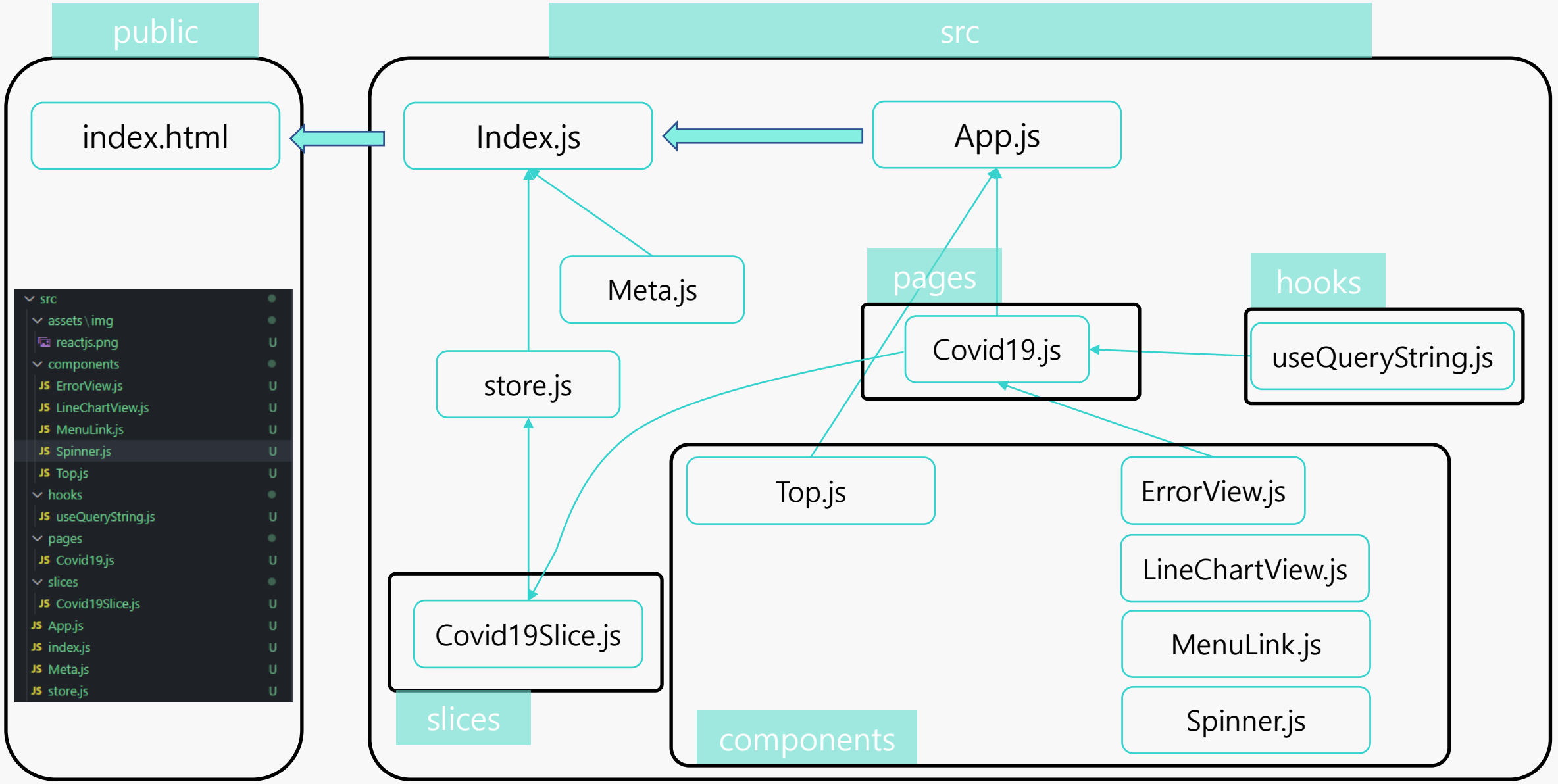


01 구현 결과 스크린샷

각 링크별 결과



02 파일구조


public
src
index.html
Index.js
App.js
Meta.js
store.js
pages
Covid19.js
hooks
useQueryString.js
Top.js
ErrorView.js
LineChartView.js
MenuLink.js
Spinner.js
Covid19Slice.js
slices
components
src
assets \ img
reactjs.png
components
ErrorView.js
LineChartView.js
MenuLink.js
Spinner.js
Top.js
hooks
useQueryString.js
pages
Covid19.js
slices
Covid19Slice.js
App.js
index.js
Meta.js
store.js

03 상세코드

index.js

```
/**
 * @filename: index.js
 * @author: 박정모(oooperbjm@gmail.com)
 * @description: React 초기화 파일
 */

import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./store";
import Meta from './Meta';

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
    <React.StrictMode>
        <Provider store={store}>
          <Meta />
            <BrowserRouter>
                <App />
            </BrowserRouter>
        </Provider>
    </React.StrictMode>
);

```

03 상세코드

store.js

```
/**
 * @filename: store.js
 * @description: reducer와 state 관리
 * @author: 박정모(oooperbjm@gmail.com)
 */

import {configureStore} from '@reduxjs/toolkit';
import Covid19Slice from './slices/Covid19Slice';

const store = configureStore({
    reducer: {
        // 직접 작성한 reducer
        covid19: Covid19Slice
    },
    middleware: (getDefaultMiddleware) => getDefaultMiddleware({serializableCheck: false}),
    devTools: true,
});

export default store;
```

03 상세코드

App.js

```
/**
 * @filename: App.js
 * @description: 페이지 시작점
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React, { memo } from 'react';
import Top from './components/Top';
import Covid19 from './pages/Covid19';
import {Routes, Route} from 'react-router-dom';

// 컴포넌트 최적화를 위한 memo
const App = memo(() => {
  return (
    <div>
      <Top />
      <Routes>
        {/* path파라미터로 각 링크별 field를 설정 */}
        <Route path="/:field/*" element={<Covid19 />} />
      </Routes>
    </div>
  );
});

export default App;
```

03 상세코드

Covid19Slice.js

```js
/**
 * @filename: Covid19Slice.js
 * @description: json서버와 통신하는 비동기함수 + slice(Action함수 + Reducers)
 * @author: 박정모(oooperbjm@gmail.com)
 */

import {createSlice, createAsyncThunk} from '@reduxjs/toolkit';
import axios from 'axios';

/** 비동기 처리 함수 구현 */
export const covidGet = createAsyncThunk("covid19/getList", async (payload, {rejectWithValue}) => {
    let result = null;

    try {
        result = await axios.get('http://localhost:3001/covid19', {
            params: {
                date_gte: payload ? payload.date_gte : null,
                date_lte: payload ? payload.date_lte : null,
            }
        });
    } catch (e) {
        result = rejectWithValue(e.response);
    }

    return result;
});
```

```js
/** Slice 정의 */
const Covid19Slice = createSlice({
    name: 'covid19',
    initialState: {
        data: null,
        loading: false,
        error: null,
    },
    reducers: {},
    extraReducers: {
        [covidGet.pending]: (state, {payload}) => {
            return { ...state, loading: true };
        },
        [covidGet.fulfilled]: (state, {payload}) => {
            return {
                data: payload?.data,
                loading: false,
                error: null,
            }
        },
        [covidGet.rejected]: (state, {payload}) => {
            return {
                data: payload?.data,
                loading: false,
                error: {
                    code: payload?.status ? payload.status : 500,
                    message: payload?.statusText ? payload.statusText : 'Server Error',
                }
            }
        }
    }
});

// 리듀서 객체 내보내기
export default Covid19Slice.reducer;
```

03 상세코드

Covid19.js

```jsx
/**
 * @filename: Covid19.js
 * @description: 메뉴링크와 차트를 렌더링하는 페이지
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React, { memo } from "react";
import { useSelector, useDispatch } from "react-redux";
import { covidGet } from "../slices/Covid19Slice";
import { useQueryString } from "../hooks/useQueryString";
import LineChartView from "../components/LineChartView";
import Spinner from "../components/Spinner";
import ErrorView from "../components/ErrorView";
import MenuLink from "../components/MenuLink";
import dayjs from "dayjs";

// 컴포넌트 최적화를 위한 memo
const Covid19 = memo(() => {
    // queryString에서 date_gte와 date_lte값을 받아옴
    const { date_gte, date_lte } = useQueryString();
    const { data, loading, error } = useSelector((state) => state.covid19);
    const dispatch = useDispatch();

    // action 파라미터로 date_gte는 그대로, date_lte는 하루를 더해서 설정함
    React.useEffect(() => {
        dispatch(covidGet({ date_gte: date_gte, date_lte: dayjs(date_lte).add(1, "d").format("YYYY-MM-DD") }))
    }, [dispatch, date_gte, date_lte]);

    return (
        <div>
            <Spinner visible={loading} />
            {error ? (
                <ErrorView />
            ) : (
                // 데이터가 있다면, 링크를 띄우고 받아온 데이터를 LineChartView태그에 props로 전달함
                data && (
                    <>
                        <div>
                            <nav>
                                <MenuLink to={`/confirmed?date_gte=${date_gte}&date_lte=${date_lte}`}>일일확진자</MenuLink>
                                <MenuLink to={`/confirmed_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적확진자</MenuLink>
                                <MenuLink to={`/active?date_gte=${date_gte}&date_lte=${date_lte}`}>격리환자</MenuLink>
                                <MenuLink to={`/released?date_gte=${date_gte}&date_lte=${date_lte}`}>격리해제</MenuLink>
                                <MenuLink to={`/released_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적격리해제</MenuLink>
                                <MenuLink to={`/death?date_gte=${date_gte}&date_lte=${date_lte}`}>사망자</MenuLink>
                                <MenuLink to={`/death_acc?date_gte=${date_gte}&date_lte=${date_lte}`}>누적사망자</MenuLink>
                            </nav>
                        </div>

                        <div>
                            <LineChartView covid_data={data} />
                        </div>
                    </>
                )
            )}
        </div>
    );
});

export default Covid19;
```

03 상세코드

Top.js

```styled.js
/**
 * @filename: Top.js
 * @description: 상단의 페이지 타이틀과 날짜 선택 Form
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React, { memo, useCallback } from "react";
import { useNavigate } from "react-router-dom";
import styled from "styled-components";


const Form = styled.form`
    position: sticky;
    display: flex;
    top: 0;
    background-color: #fff;
    border-top: 1px solid #eee;
    border-bottom: 1px solid #eee;
    padding: 10px 0;
    margin: 0;
    margin-bottom: 20px;

    input,
    button {
        display: block;
        margin-right: 5px;
        font-size: 16px;
        padding: 0 10px;
        height: 30px;
    }

    button {
        width: 70px;
        flex: none;
    }
`;
```

```Top.js
// 컴포넌트 최적화를 위한 memo
const Top = memo(() => {
    const navigate = useNavigate();

    // 컴포넌트 최적화를 위한 useCallback
    const onSearchSubmit = useCallback(
        (e) => {
            e.preventDefault();
            // input 태그에 입력된 값을 받아온다.
            const date_gte = e.target.date_gte.value;
            const date_lte = e.target.date_lte.value;

            // 시작날과 마지막날이 하나라도 결정되지 않았을 때는 이동하지 않는다.
            // 둘 다 존재할 경우 path파라미터는 첫 번째 탭인 일일확진자 confirmed를, queryString은 날짜값을 전달한다.
            date_gte && date_lte && navigate(`/confirmed?date_gte=${date_gte}&date_lte=${date_lte}`);
        },
        [navigate]
    );

    return (
        <div>
            <h1>
                <a href="http://localhost:3000" style={{ textDecoration: "none", color: "black" }}>
                    Covid19 현황
                </a>
            </h1>
            <Form onSubmit={onSearchSubmit}>
                <input type="date" name="date_gte" />
                <input type="date" name="date_lte" />
                <button type="submit">검색</button>
            </Form>
        </div>
    );
});

export default Top;
```

03 상세코드

LineChartView.js

```jsx
/**
 * @filename: LineChartView.js
 * @description: 선 그래프 표시를 위한 컴포넌트
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React, { memo } from "react";
import { Chart as ChartJS, CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend } from "chart.js";
import { Line } from "react-chartjs-2";
import { useParams } from "react-router-dom";

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend);

const LineChartView = memo(({ covid_data }) => {

    // path파라미터를 받아옴
    const { field } = useParams();

    /** 그래프 옵션 */
    const options = {
        responsive: true,
        plugins: {
            legend: {
                position: "top",
            },
            title: {
                display: false,
                text: "Chart.js Line Chart",
            },
        },
    };

    const labels =
        covid_data &&
        covid_data.map((v) => {
            // json 데이터의 날짜값을 잘라서 차트에 사용
            return v.date.substring(0, 10);
        });

    /** chart에 표시될 데이터 (막대그래프용) */
    const data = {
        labels,
        datasets: [
            {
                label: "명",
                // path파라미터로 받아온 키워드를 이용하여 map함수로 해당하는 값을 받아옴
                data: covid_data && covid_data.map((v) => v[field]),
                borderColor: "rgb(255, 99, 132)",
                backgroundColor: "rgba(255, 99, 132, 0.5)",
            },
        ],
    };

    return <Line data={data} options={options} />;
});

export default LineChartView;

```

03 상세코드

ErrorView.js

```jsx
/**
 * @filename: ErrorView.js
 * @description: 에러메시지 표시를 위한 컴포넌트
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React, { memo } from 'react';

const ErrorView = memo(({error}) => {
    return (
        <div>
            <h1>Oops~!!! {error.code} Error.</h1>
            <hr />
            <p>{error.message}</p>
        </div>
    );
});

export default ErrorView;
```

03 상세코드

MenuLink.js

```jsx
/**
 * @filename: MenuLink.js
 * @description: 메뉴링크를 구성하기 위한 styledComponent
 * @author: 박정모(oooperbjm@gmail.com)
 */


import React from "react";
import styled from "styled-components";
import { NavLink } from "react-router-dom";

/** 메뉴링크 --> NavLink: 현재 머물고 있는 페이지와 관련된 링크에 CSS 적용 */

const MenuLinkContainer = styled(NavLink)`
    font-size: 20px;
    cursor: pointer;
    text-decoration: none;
    padding-bottom: 2px;
    color: #222;

    &:hover {
        color: #22b8cf;
    }

    &:after {
        content: '|';
        display: inline-block;
        padding: 0 7px;
        color: #ccc;
    }

    &:last-child {
        &:after {
            // 글자색을 흰색으로 지정하여 화면에서 숨긴다.
            color: #fff;
        }
    }

    /*
        URL이 현재 메뉴를 가르키는 경우(콜론이 아닌 점에 주의)
        활성 메뉴에 적용되는 기본 클래스 이름이 'active'이다.
    */

    &.active {
        text-decoration: underline;
        color: #22b8cf;

        &:after {
            // 흰색 선을 추가하여 .active에서 지정한 border를 덮을 수 있도록 지정한다.(가림 효과)
            border-bottom: 4px solid #fff !important;
        }
    }
`;

const MenuLink = ({to, children}) => <MenuLinkContainer to={to}>{children}</MenuLinkContainer>;

export default MenuLink;
```

03 상세코드

Meta.js

```jsx
/**
 * @filename: Meta.js
 * @description: SEO처리
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React from "react";
import { Helmet, HelmetProvider } from "react-helmet-async";
import sample from './assets/img/reactjs.png';

const Meta = (props) => {
    return (
        <HelmetProvider>
            <Helmet>
                <meta charSet="utf-8" />
                <title>{props.title}</title>
                {/* SEO 태그 */}
                <meta name="description" content={props.description} />
                <meta name="keywords" content={props.keywords} />
                <meta name="author" content={props.author} />
                <meta property="og:type" content="website" />
                <meta property="og:title" content={props.title} />
                <meta property="og:description" content={props.description} />
                <meta property = 'og:image' content = {props.image} />
                <meta property="og:url" content={props.url} />

                <link rel='shortcut icon' href={props.image} type="image/png" />
                <link rel='icon' href={props.image} type="image/png" />
                <link rel="preconnect" href="https://fonts.googleapis.com" />
                <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
                <link href="https://fonts.googleapis.com/css2?family=Gugi&family=Noto+Sans+KR:wght@100;300;400;500&display=swap" rel="stylesheet" />

            </Helmet>
        </HelmetProvider>
    );
};

Meta.defaultProps = {
    title: "React Exam",
    description: "React.js 시험 답안입니다.",
    keywords: "React",
    author: "박정모(oooperbjm@gmail.com)",
    img: sample,
    url: window.location.href,
};

export default Meta;
```

03 상세코드

Spinner.js

```jsx
/**
 * @filename: Spinner.js
 * @description: 로딩바 컴포넌트
 * @author: 박정모(oooperbjm@gmail.com)
 */

import React from "react";
import PropTypes from "prop-types";
import styled from "styled-components";

/** 로딩바 컴포넌트 */
// --> https://mhnpd.github.io/react-loader-spinner/
import { Bars } from "react-loader-spinner";

/** 로딩바 뒤에 표시될 반투명 막 */
const TransLayer = styled.div`
    position: fixed;
    left: 0;
    top: 0;
    z-index: 9999;
    background-color: #0003;
    width: 100%;
    height: 100%;
`;
```

```jsx
const Spinner = ({ visible, color, width, height }) => {
    return (
        <>
            {visible && (
                <TransLayer>
                    <Bars
                        color={color}
                        height={width}
                        width={height}
                        wrapperStyle={{
                            position: "absolute",
                            zIndex: 10000,
                            left: "50%",
                            top: "50%",
                            marginLeft: -width / 2 + "px",
                            marginTop: -height / 2 + "px",
                        }}
                    />
                </TransLayer>
            )}
        </>
    );
};

/** 기본값 정의 */

Spinner.defaultProps = {
    visible: false,
    color: '#06f',
    width: 100,
    height: 100
}

/** 데이터 타입 설정 */
Spinner.propTypes = {
    visible: PropTypes.bool.isRequired,
    color: PropTypes.string,
    width: PropTypes.number,
    height: PropTypes.number,
}

export default Spinner;
```

03 상세코드

useQueryString.js

```js
/**
 * @filename: useQueryString.js
 * @description: 쿼리스트링을 추출하여 json객체로 반환하는 hook
 * @author: 박정모(oooperbjm@gmail.com)
 */

import { useLocation } from 'react-router-dom';

const useQueryString = () => {
    // QueryString 문자열 추출함
    const {search} = useLocation();

    // QueryString 문자열을 객체로 변환
    const params = new URLSearchParams(search);

    // 모든 key와 value의 쌍을 for...in 반복문으로 처리 가능한 [key, value] 쌍의 배열로 반환함
    const entries = params.entries();

    // 리턴할 빈 객체
    const result = {};

    // 추출한 배열을 반복문으로 처리하여 JSON 객체로 변환함
    for (const [key, value] of entries) {
        result[key] = value;
    }

    return result;
};

export { useQueryString };
```

04 후기

소요시간	4시간
후기	날짜 선택 input으로 날짜를 받아서, 그걸 queryString으로 보내고, action에 담아 dispatch하는 것 까지는 금방 했는데, Link걸고 데이터를 핸들링하는 부분에서 시간이 조금 지체되었습니다. queryString과 path파라미터를 동시에 다루려고 하다보니 버벅이는 부분이 조금 있었지만, 평소에 살짝 헷갈리던 부분인데 시험을 통해서 오히려 확실하게 사용법을 익힌 것 같습니다. Chart.js같은 경우는 워낙 친절하게 설명이 쓰여있어서 데이터만 갈무리하면 바로 해결되었습니다.