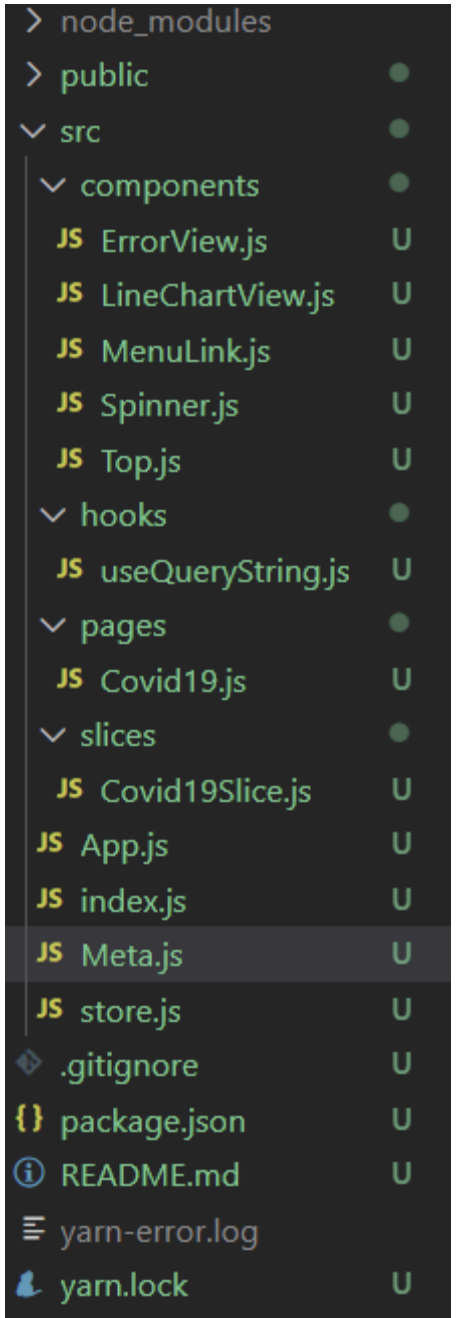
리액트_프로그래밍_신지섭_보고서

1. 파일 구조


node_modules
public
src
components
ErrorView.js U
LineChartView.js U
MenuLink.js U
Spinner.js U
Top.js U
hooks
useQueryString.js U
pages
Covid19.js U
slices
Covid19Slice.js U
App.js U
index.js U
Meta.js U
store.js U
.gitignore U
package.json U
README.md U
yarn-error.log
yarn.lock U

2. 소스코드(index.js)

```
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import App from './App';
4   import { BrowserRouter } from 'react-router-dom';
5
6   import { Provider } from 'react-redux';
7   import Meta from "./Meta"
8   import store from './store';
9
10  const root = ReactDOM.createRoot(document.getElementById('root'));
11  root.render(
12    <React.StrictMode>
13      <Provider store={store}>
14        <Meta/>
15        <BrowserRouter>
16          <App />
17        </BrowserRouter>
18      </Provider>
19    </React.StrictMode>
20  );
```

2. 소스코드(App.js)

```
1   import React, { memo } from 'react';
2   import {Routes, Route } from 'react-router-dom';
3
4   import Top from './components/Top';
5   import Covid19 from './pages/Covid19';
6   const App = memo(() => {
7     return (
8       <div>
9         <Top/>
10        <Routes>
11          <Route path="/:field" element={<Covid19/>}/>
12        </Routes>
13      </div>
14    );
15  })
16
17  export default App;
18
```

2. 소스코드(Meta.js)

```
/**
 * @filename: Meta.js
 * @description: <head>태그 내의 SEO 처리 및 기본 참조 리소스 명시
 * @author: 신지섭
 */

/** 패키지 참조 */
// 기본 참조 객체
import React from 'react';
// SEO 처리 기능 패키지
import { Helmet, HelmetProvider } from 'react-helmet-async';
/**
 * SEO 처리 컴포넌트
 * @param props
 * @returns {JSX.Element}
 */
const Meta = (props) => {
    return (
        <HelmetProvider>
            <Helmet>
                <meta charSet='utf-8' />
                {/* SEO 태그 */}
                <title>{props.title}</title>
                <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"/>
                {/* <meta property='og:image' content={props.image} /> */}
            </Helmet>
        </HelmetProvider>
    );
};

/**
 * props에 대한 기본값 설정
 * @type {{keywords: string, author: string, description: string, title: string, url: string}}
 */
Meta.defaultProps = {
    title: 'React 시험',
    description: 'React.js로 만든 covid19 data Redux활용 react 시험 입니다.' ,
    keywords: 'React,Redux,covid19',
    author: '신지섭',
    url: window.location.href
};

export default Meta;
```

2. 소스코드(store.js)

```
import { configureStore } from "@reduxjs/toolkit";
import Covid19Slice from "./slices/Covid19Slice";
const store = configureStore({
    reducer: {
        covidData: Covid19Slice
    },
    middleware: (getDefaultMiddleware) => getDefaultMiddleware({serializableCheck: false}),
    devTools: true
})

export default store;
```

2. 소스코드(slices > Covid19Slice.js)

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

export const getCovid19 = createAsyncThunk("Covid19Slice/getCovid19", async(payload, {
rejectWithValue}) => {
    let result = null;
    try {
        result = await axios.get("http://localhost:3001/covid19", {
            params: {
                date_gte: payload.gte ? payload.gte : null,
                date_lte: payload.lte ? payload.lte : null,
            }
        });
    } catch(err) {
        result = rejectWithValue(err.response);
    }
    return result;
});

const Covid19Slice = createSlice({
    name: 'covidData',
    initialState: {
        data: null,
        loading: false,
        error: null
    },
    reducers: {},
    extraReducers: {
        [getCovid19.pending]: (state, { payload }) => {
            return { ...state, loading: true}
        },
        [getCovid19.fulfilled]: (state, { payload }) => {
            return {
                data: payload?.data,
                loading: false,
                error: null
            }
        },
        [getCovid19.rejected]: (state, {payload}) => {
            return {
                data: payload?.data,
                loading: false,
                error: {
                    code: payload?.status ? payload.status : 500,
                    messsage: payload?.statusText ? payload.statusText : 'Server Error'
                }
            }
        }
    }
})

export default Covid19Slice.reducer
```

2. 소스코드(pages > Covid19.js)

```python
import React, { memo } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { getCovid19 } from '../slices/Covid19Slice';
import { useQueryString } from '../hooks/useQueryString';
import { useParams } from 'react-router-dom';
//로딩바 컴포넌트
import Spinner from "../components/Spinner"
//에러정보를 표시하기 위한 컴포넌트
import ErrorView from "../components/ErrorView"
import MenuLink from '../components/MenuLink';
import dayjs from 'dayjs';
import LineChartView from '../components/LineChartView';

const Covid19 = memo(() => {
    const dispatch = useDispatch();
    //필드명 가져오기
    const {field} = useParams()
    // gte 값 가져오기
    const {date_gte} = useQueryString();
    // lte 값 가져오기
    const {date_lte} = useQueryString();
    // Covid19Slice에서 받아온 데이터, 로딩, 에러 가져오기
    const {data, loading, error} = useSelector(state => state.covidData)
    //useState를 통해 chartData 관리
    const [chartData, setChartData] = React.useState();
    React.useEffect(() => {
        dispatch(getCovid19({
            gte:date_gte,
            lte: dayjs(date_lte).add(+1, "d").format("YYYY-MM-DD") // 1일 전 값을 가져오므로 +1
        }))
    },[dispatch, date_gte, date_lte])
```

```python
    React.useEffect(() => {
        if (data) {
            const newData = {
                date: [],
                fieldData: []
            }
            data.map((v,i) => {
                newData.date.push(dayjs(v.date).format("MM-DD"));
                // field에 따라 fieldData 에 넣는 값을 다르게 한다.
                if (field === "confirmed") {
                    newData.fieldData.push(v.confirmed)
                } else if (field === "confirmed_acc") {
                    newData.fieldData.push(v.confirmed_acc)
                } else if (field === "active") {
                    newData.fieldData.push(v.active)
                } else if (field === "released") {
                    newData.fieldData.push(v.released)
                } else if (field === "released_acc") {
                    newData.fieldData.push(v.released_acc)
                } else if (field === "death") {
                    newData.fieldData.push(v.death)
                } else if (field === "death_acc") {
                    newData.fieldData.push(v.death_acc)
                }
            })
            setChartData(newData)
        };
    },[field,data])
    return (
        <div>
            <Spinner visible={loading}/>
            {error ? (<ErrorView/>) : (
                <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>
                    <LineChartView chartData={chartData}/>
                </div>
            )}
        </div>
    );
});

export default Covid19;
```

2. 소스코드(components > Top.js)

```
import React, { memo, useCallback } from 'react';
import { useNavigate } from 'react-router-dom';
import { useQueryString } from '../hooks/useQueryString';
import styled from 'styled-components';
import dayjs from 'dayjs';


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;

`


const Top = memo(() => {
    const navigate = useNavigate();
    /**
    const qs = useQueryString();
    console.log(qs);
    const query = qs.query;
    /*/
    const {date_gte} = useQueryString();
    const {date_lte} = useQueryString();
    /**/

    const onSearchSubmit = useCallback((e) => {
        e.preventDefault();
        navigate(`confirmed?date_gte=${e.target.gte.value}&date_lte=${e.target.lte.value}`);
    }, [navigate]);
    return (
        <div>
            <h1>Covid 19 현황</h1>
            <hr/>
            <Form onSubmit={onSearchSubmit}>
                <input type='date' name='gte' defaultValue={date_gte}/>
                ~
                <input type='date' name='lte' defaultValue={date_lte ? date_lte : dayjs().format(
"YYYY-MM-DD")}/>
                <button type='submit'>검색</button>
            </Form>

        </div>
    );
});

export default Top;
```

2. 소스코드(components > Spinner.js)

```
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;
    right: 0;
    z-index: 9999;
    background-color: #0003;
    width: 100%;
    height: 100%;
`

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

/**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;
```

2. 소스코드(components > MenuLink.js)

```
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;
    /* css의 가상 클래스 hover*/
    &:hover {
        color: #22b9cf;
    }

    &: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. (가림효과)*/
            border-bottom: 4px solid #fff !important;
        }
    }
`
const MenuLink = ({to, children}) => <MenuLinkContainer to={to}>{children}</MenuLinkContainer>

export default MenuLink;
```

2. 소스코드(components > LineChartView.js)

```
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';
ChartJS.register(
    CategoryScale,
    LinearScale,
    PointElement,
    LineElement,
    Title,
    Tooltip,
    Legend
);
const LineChartView = memo(({chartData}) => {
    /** 그래프 옵션 */
    const options = {
        responsive: true,
        plugins: {
          legend: {
            position: 'top'
          },
        },
    }
    const data = {
        // x축에 나타날 항목들
        labels: chartData.date,
        // y축의 값을 비롯한 기타 옵션들
        datasets: [{
            //그래프 제목
            label: '명',
            backgroundColor: '#0066ff44',
            borderColor: '#0066ff',
            borderWidth: 1,
            //그래프 각 항목별 y축 수치값
            data: chartData.fieldData,
        }]
    };
    return (
        <Line
            data={data}
            options={options}
        />

    );
});
LineChartView.defaultProps = {
    chartData: {
        movieNm: [], audiCnt: []
    }
}

export default LineChartView;
```

2. 소스코드(components > ErrorView.js)

```
1   import React, { memo } from 'react';
2
3   const ErrorView = memo(({error}) => {
4       return (
5           <div>
6               <h1>Oops~!!! {error.code} Error.</h1>
7               <hr/>
8               <p>{error.message}</p>
9           </div>
10      );
11  });
12
13  export default ErrorView;
```

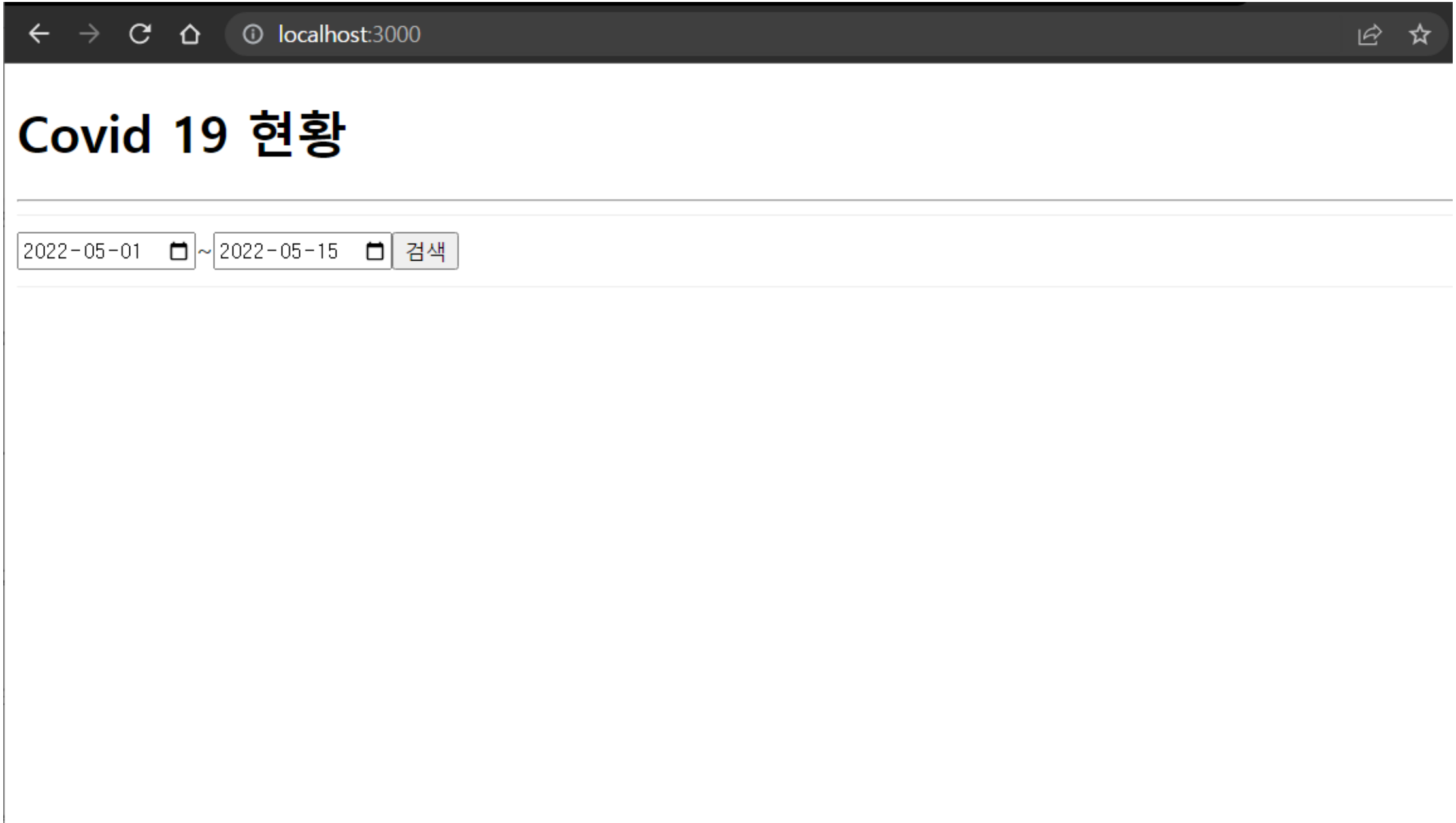
2. 소스코드(components > ErrorView.js)

```
1   import React, { memo } from 'react';
2
3   const ErrorView = memo(({error}) => {
4       return (
5           <div>
6               <h1>Oops~!!! {error.code} Error.</h1>
7               <hr/>
8               <p>{error.message}</p>
9           </div>
10      );
11  });
12
13  export default ErrorView;
```

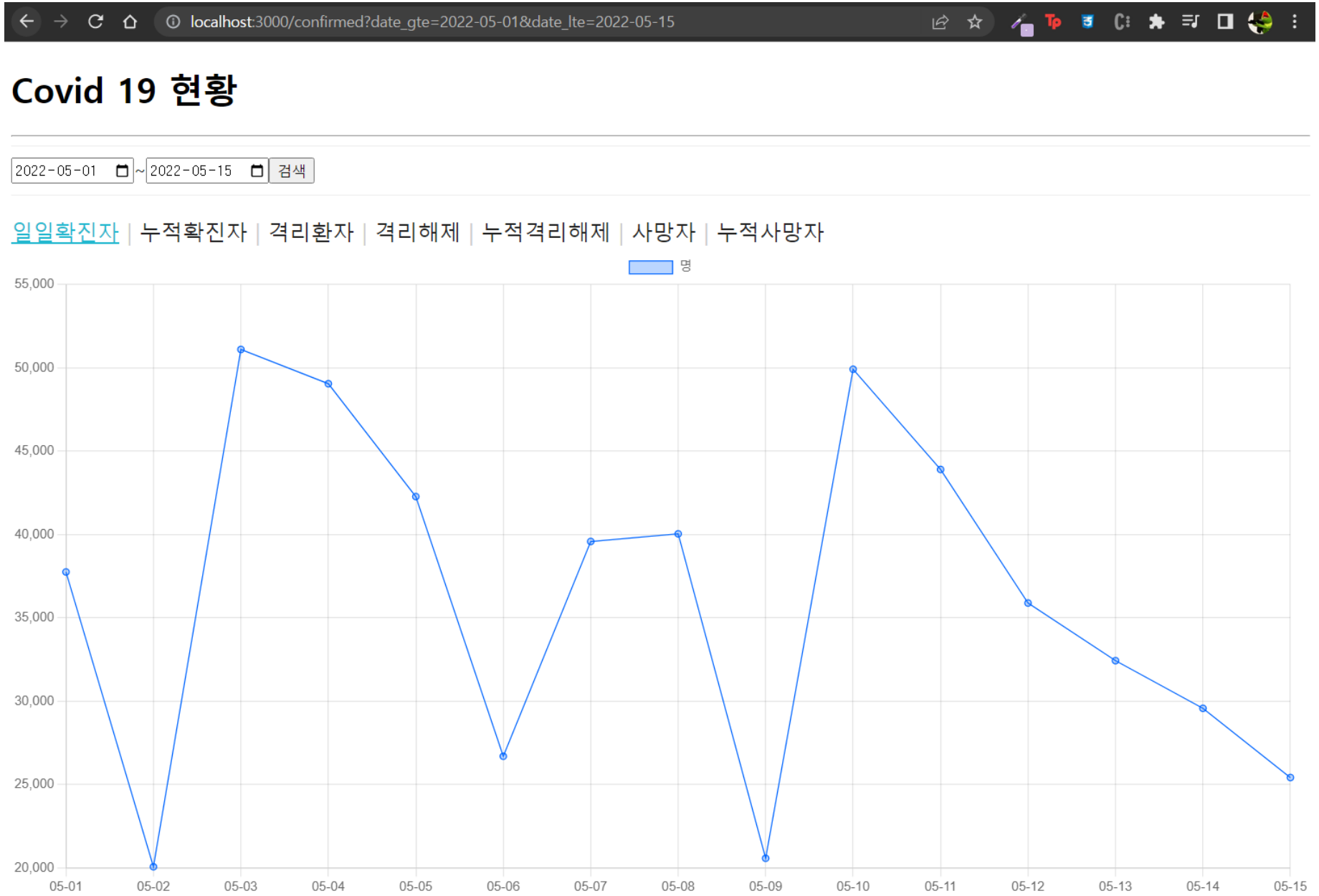
2. 소스코드(hooks > useQueryString.js)

```
1   import { useLocation } from 'react-router-dom';
2
3   const useQueryString = () => {
4       //QueryString 문자열 추출함
5       const { search } = useLocation();
6       //QueryString 문자열을 객체로 변환
7       const params = new URLSearchParams(search);
8       // 모든 key와 value의 쌍을 for ...in 반복문으로 처리 가능함 [key, value] 쌍의 배열로 반환함.
9       const entries = params.entries();
10
11      //리턴할 빈 객체
12      const result = {}
13
14      //추출한 배열을 반복문으로 처리하여 JSON객체로 변환함
15      for (const [key, value] of entries) {
16          result[key] = value;
17      }
18
19      return result;
20  }
21
22  export { useQueryString };
```

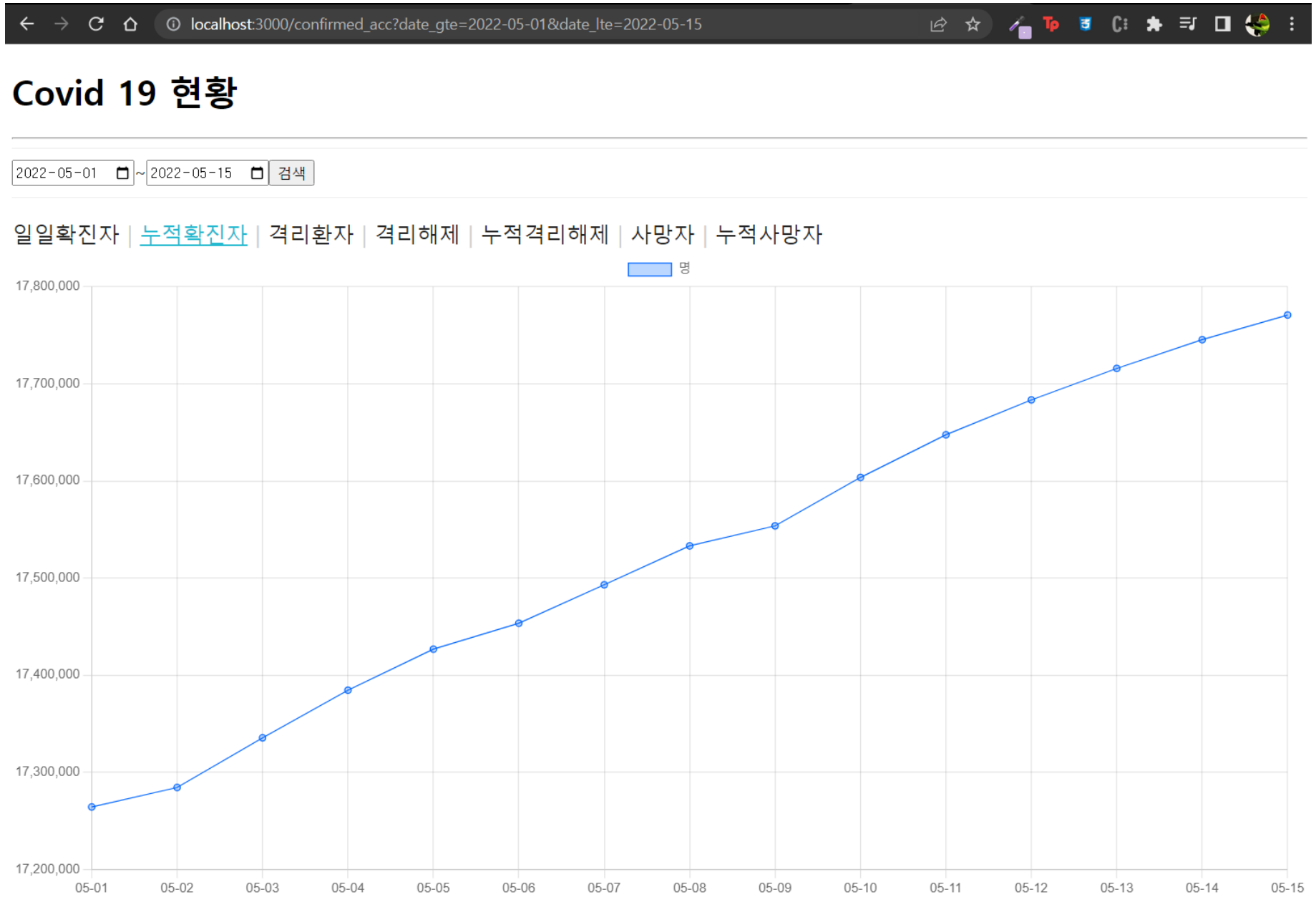
3. 구현결과(버튼 클릭 전)



3. 구현결과(버튼 클릭 후 일일확진자)



3. 구현결과(누적확진자)


localhost:3000/confirmed_acc?date_gte=2022-05-01&date_lte=2022-05-15
Covid 19 현황
2022-05-01
~
2022-05-15
검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
명
17,800,000
17,700,000
17,600,000
17,500,000
17,400,000
17,300,000
17,200,000
05-01
05-02
05-03
05-04
05-05
05-06
05-07
05-08
05-09
05-10
05-11
05-12
05-13
05-14
05-15

3. 구현결과(격리환자)

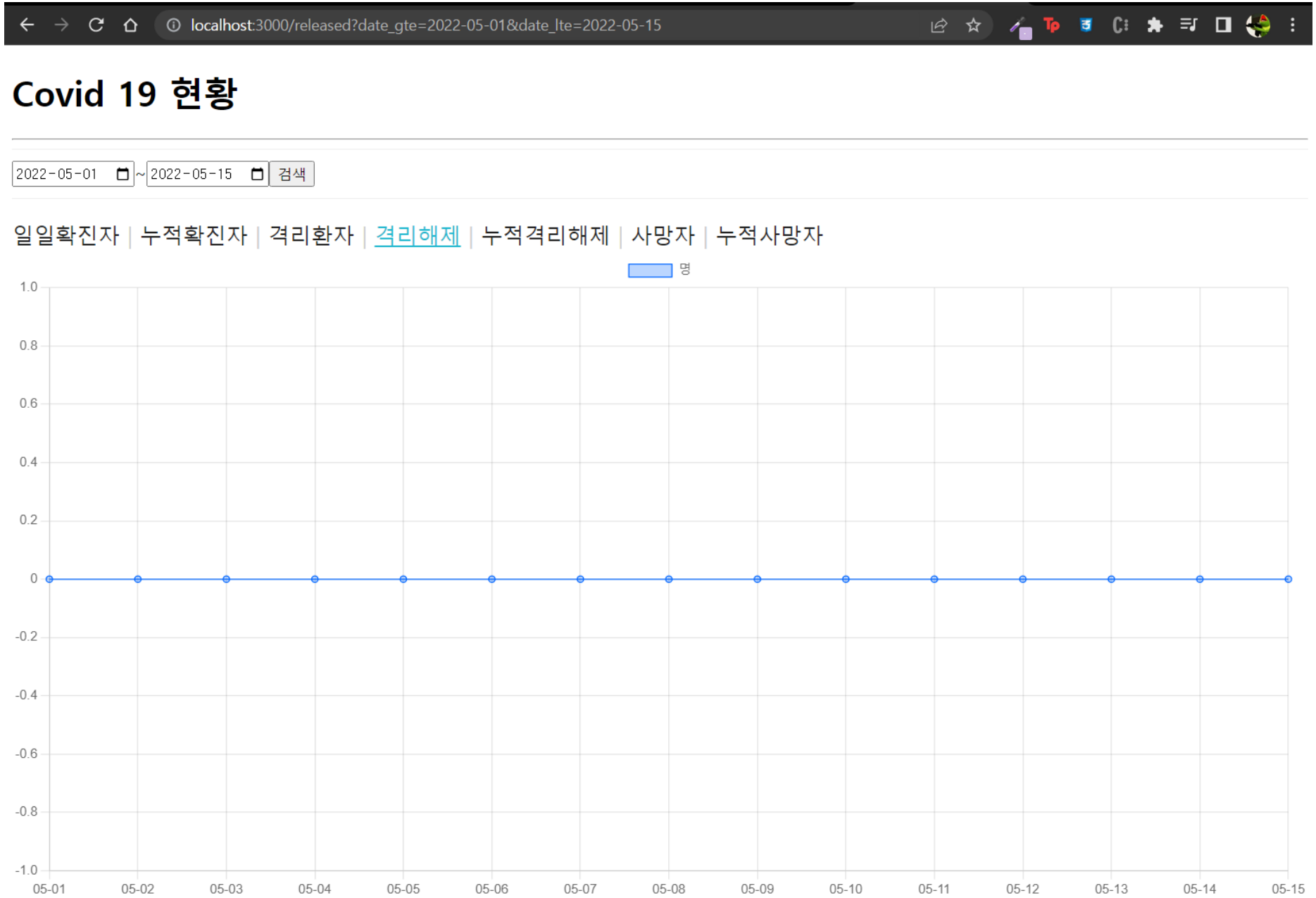
localhost:3000/active?date_gte=2022-05-01&date_lte=2022-05-15

Covid 19 현황

2022-05-01 ~ 2022-05-15 검색

일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자

3. 구현결과(격리해제)


localhost:3000/released?date_gte=2022-05-01&date_lte=2022-05-15
Covid 19 현황
2022-05-01
~
2022-05-15
검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
명
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
05-01
05-02
05-03
05-04
05-05
05-06
05-07
05-08
05-09
05-10
05-11
05-12
05-13
05-14
05-15

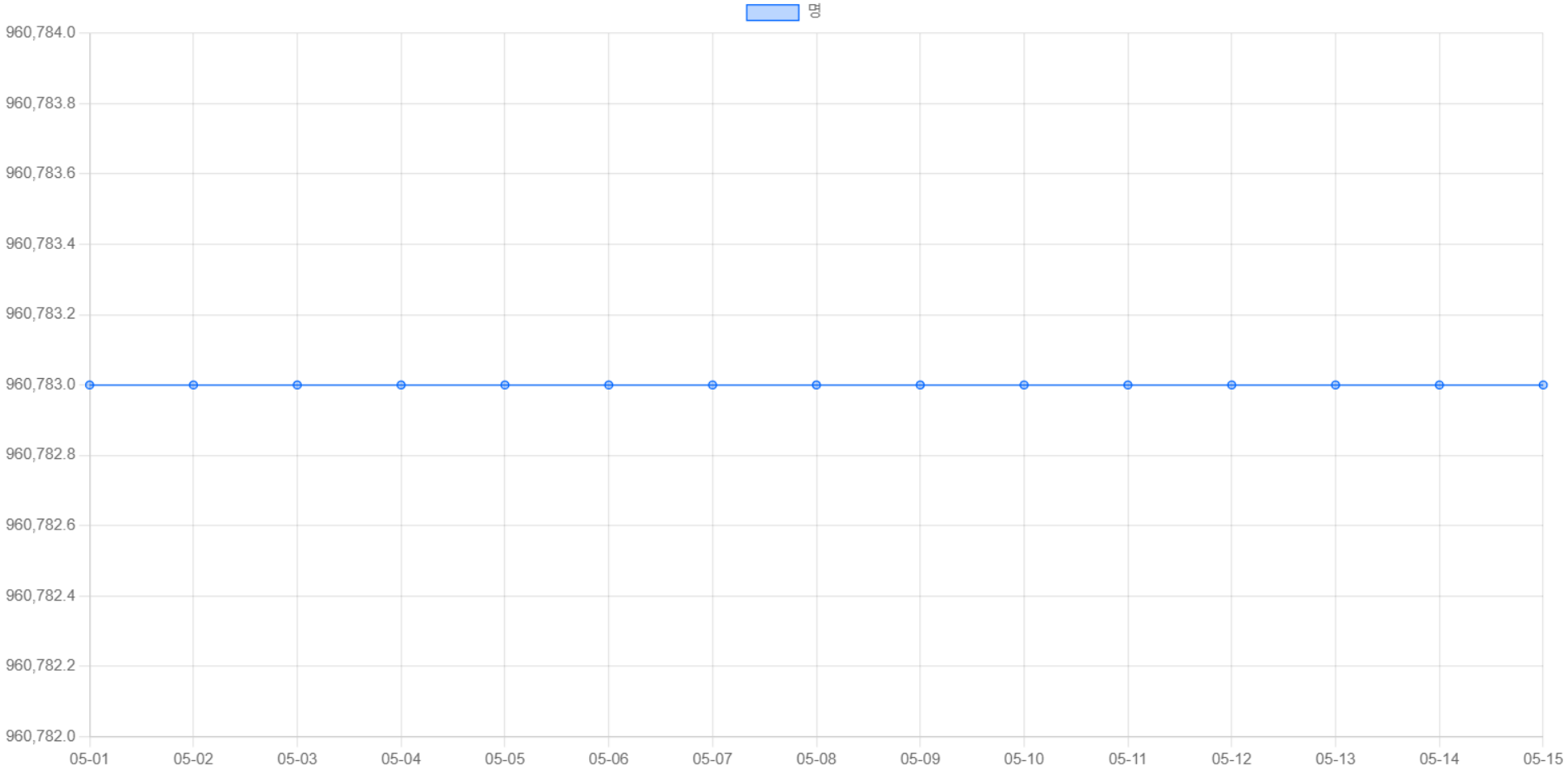
3. 구현결과(누적격리해제)

localhost:3000/released_acc?date_gte=2022-05-01&date_lte=2022-05-15

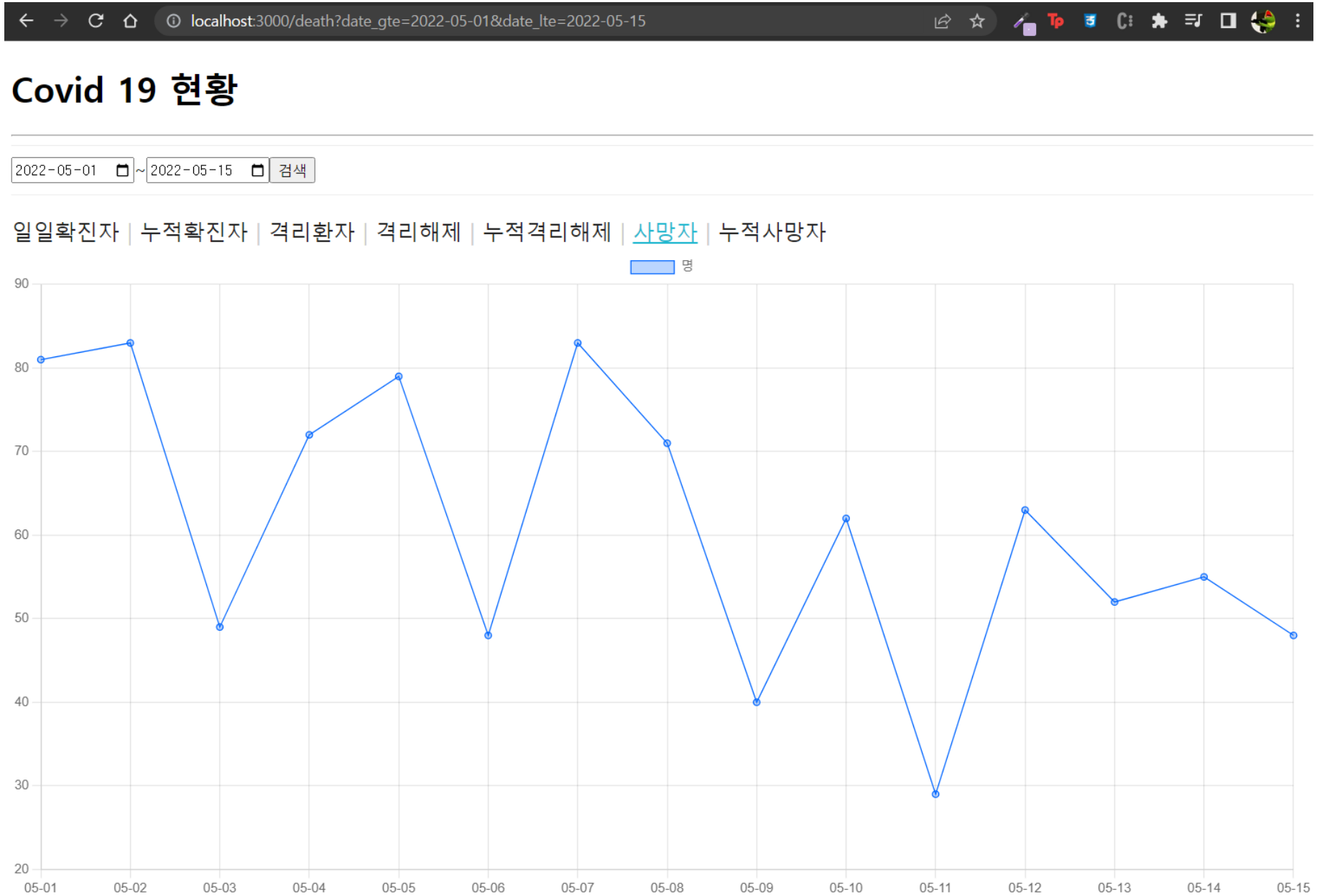
Covid 19 현황

2022-05-01 ~ 2022-05-15 검색

일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자



3. 구현결과(사망자)


localhost:3000/death?date_gte=2022-05-01&date_lte=2022-05-15
Covid 19 현황
2022-05-01
~
2022-05-15
검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
명
90
80
70
60
50
40
30
20
05-01
05-02
05-03
05-04
05-05
05-06
05-07
05-08
05-09
05-10
05-11
05-12
05-13
05-14
05-15

3. 구현결과(누적확진자)


localhost:3000/death_acc?date_gte=2022-05-01&date_lte=2022-05-15
Covid 19 현황
2022-05-01
~
2022-05-15
검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
명
23,700
23,600
23,500
23,400
23,300
23,200
23,100
23,000
22,900
22,800
05-01
05-02
05-03
05-04
05-05
05-06
05-07
05-08
05-09
05-10
05-11
05-12
05-13
05-14
05-15

4. 문제점 및 소감

2시간 동안 데이터가 안 불러와져서 뭐가 문제일까 한참을 고민했습니다. 문제는 dispatch(getCovid19)로 해놓고 2시간을 낭비해 버렸습니다. 그 후로는 이것 저것 테스트 해보면서 문제를 해결했습니다. 수업 시간내에 해결하지 못한 것이 아쉽습니다.

소요시간:

문제풀이: 5시간

Ppt: 40분