[개인 과제] 프론트엔드 기초

훈련과정명	[신한투자증권] 프로디지털아카데미 4기
강의과목명	프론트엔드 개발 기초 (JavaScript)
강사명	신윤수
과제명	실전 자바스크립트 종합 과제 - 웹 분석하여 데이터 수집 및 저장하기 (HTML) - 웹 분석하여 데이터 수집 및 저장하기 고급 (JSON) - Bootstrap 이용하여 UI 구성하기. - React에서 통신 후 UI에 Rendering하기 - React에서 복잡한 Component 구성하기
제출기한	5월 27일 오후 23:59 까지 제출 **[**과제 제출시 주의사항**]** - 첨부파일을 다운받고 작업하시길 추천드립니다**.** - 패키지 설치는 자유롭게 하셔도 되지만, 제출시에는 **node_modules** 디렉토리는 모두 제거하여 주십시요. - 과제 제출시 **homework** 디렉토리**(**가장 상위 디렉토리**)**의 이름을 제출자 이름으로 변경 후 압축하여 제출바랍니다**.** - 원활한 과제평가를 위하여 **React UI(**과제 그룹 **2)**는 산출 결과물을 캡쳐하여 이미지파일로 저장하여 주시기 바랍니다. - 결과물 캡처 이미지 파일은 가장 상위 폴더에 위치하며**(**제출자 이름**)** 파일이름은 세부과제번호로 해주십시요. (예: 신윤수/*2.1.1.jpg*)

과제 설명

자바스크립트 종합 과제로서 총 두개의 파트로 나뉘어집니다.

1. Node.js환경에서의 데이터 수집
2. React를 활용한 UI실습 및 통신

배운 것을 모두 활용하시고, 배우지 않은 것을 찾아서 활용하셔도 좋습니다.

[총 세부 과제 및 과제별 배점**]**
세부 과제의 총 개수는 **7**개이며**,** 각 세부과제별 내용과 배점은 다음과 같습니다**.**

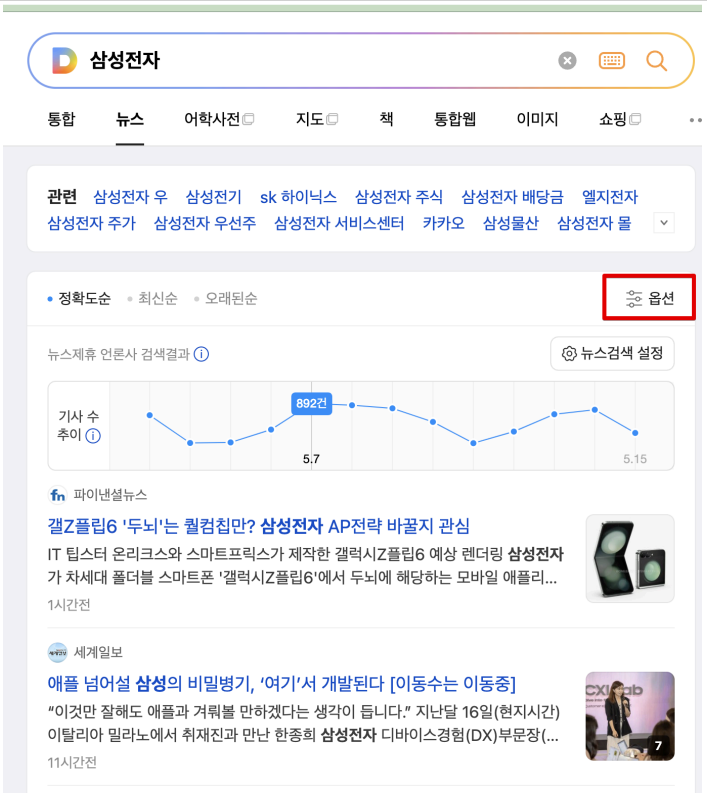
- 1.1: 다음 검색(뉴스탭)에서 뉴스 데이터 수집하기 (15점)
- 1.2: 다음 주가차트에서 데이터 수집하기 (15점)
- 2.1.1. StockApp UI 완성하기 (20점)
- 2.1.2. StockApp 데이터 통신하여 Rendering하기 (10점)
- 2.2.1. UserApp UI완성하기 (20점)
- 2.2.2. UserApp 데이터 통신 및 Rendering (10점)
- 2.2.3. UserApp 데이터 통신 및 고급 응용 (10점)

1. Node.js환경에서의 데이터 수집

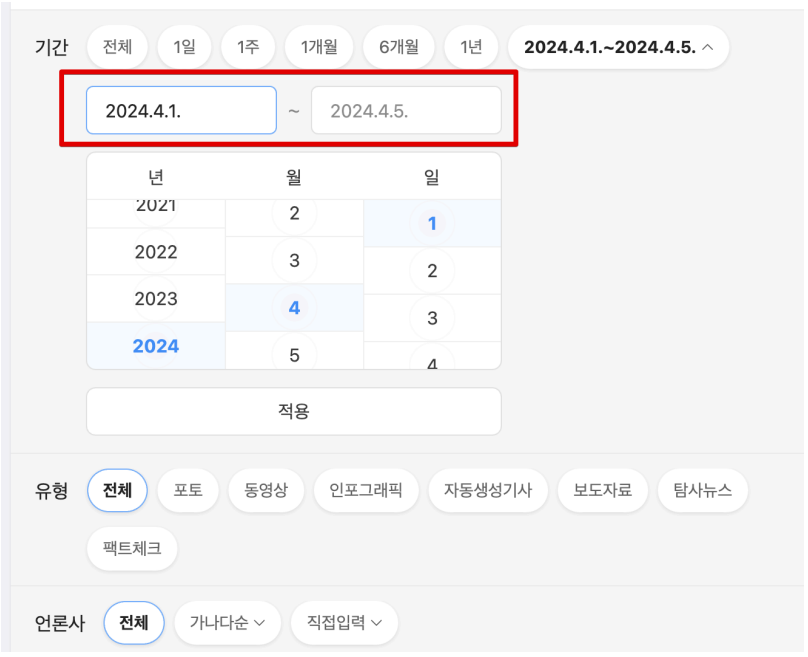
현 과제는 Node.js 환경에서 작업합니다.

1.1 다음 검색(뉴스탭)에서 뉴스 데이터 수집하기 (15점)

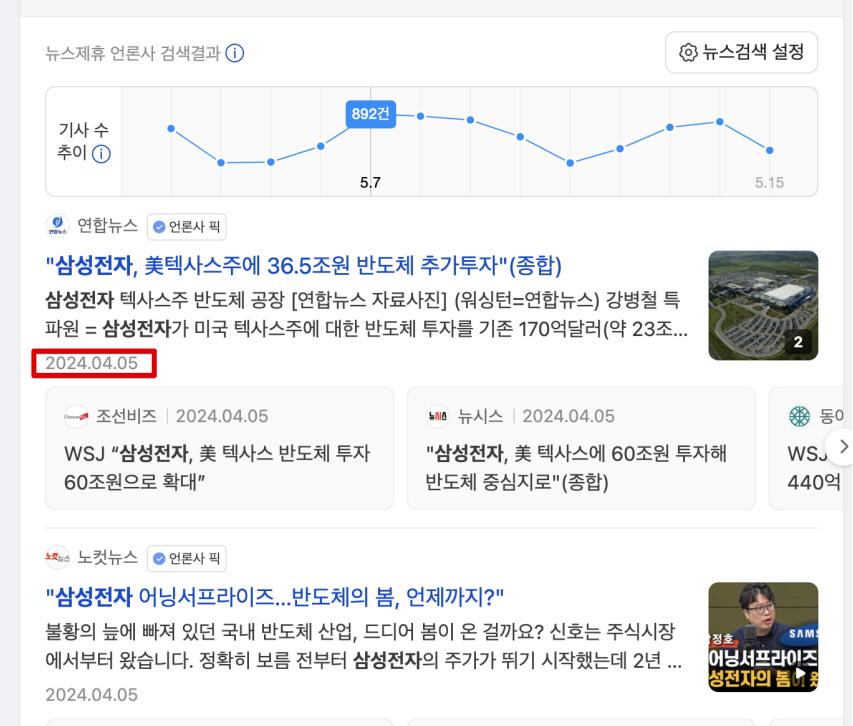
검색 **URL**: https://search.daum.net/search?w=news&cluster=y&q=삼성전자



<그림*1*>



<그림2: 옵션 클릭 후 날짜 설정>



<그림3: 해당하는 날짜에 속하는 뉴스리스트**>**

[과제 내용**]**

1. <그림1: 검색 **URL**: https://search.daum.net/search?w=news&cluster=y&q=삼성전자> 에 접속하면 나오는 페이지에서 옵션 클릭
2. <그림2>에서 볼 수 있듯이 날짜 설정을 할 수 있다.

다음 검색에서 날짜 설정을 하여 총 **3**종류의 날짜범위**(**아래 **[**날짜범위**]** 참조**)**에 해당하는 "삼성전자"검색 결과 중에 제목과 뉴스 **URL**을 수집하여 **Json** 형태로 저장하는 코드를 작성하고 실제로 저장하시오. **[**파일이름**: news.json]** (하단 결과예시 확인)

- 저장데이터의 key값은 날짜범위의 종료날짜(%YYYY%MM%DD)로 하고 value는 뉴스내용({title: "~", "url": ~})의 배열로 한다.
- 검색결과는 1페이지만 조회하도록 한다.(뉴스의 개수 10개)

[파일 저장 경로**]:**

- <제출자이름**>/homework-node/news.json** 으로 저장

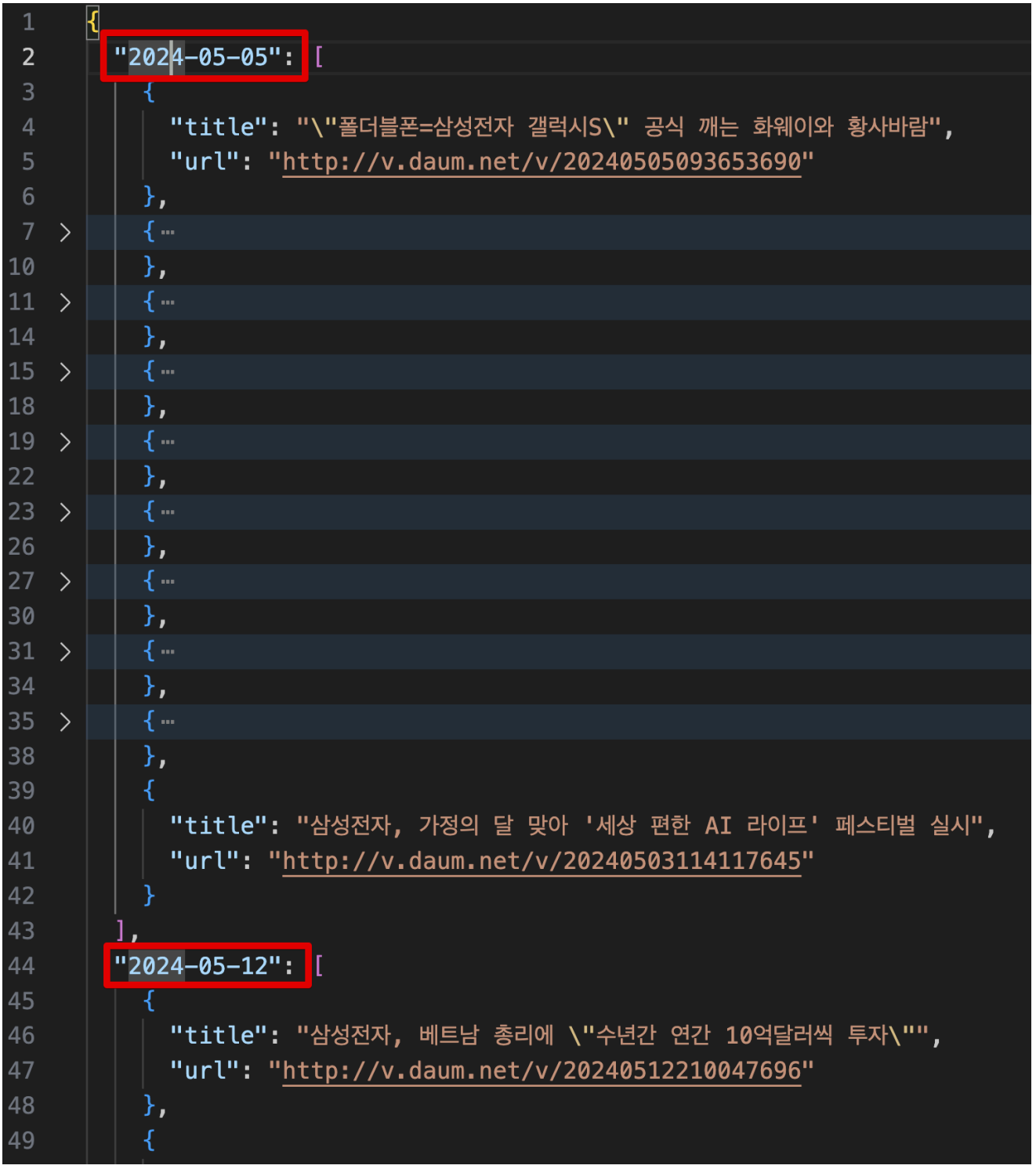
[저장할 **Json Format** 형태**:]**

```
{
  [endDate]:  [{title: "title1…", url: "https://~~url1…" }]
}
```

[날짜범위**]**

1. **20240401** 00시 00분 00초**~20240405** 23시59분59초
2. **20240408** 00시 00분 00초**~20240412** 23시59분59초
3. **20240415** 00시 00분 00초**~20240419** 23시59분59초

[결과 예시 - news.json]

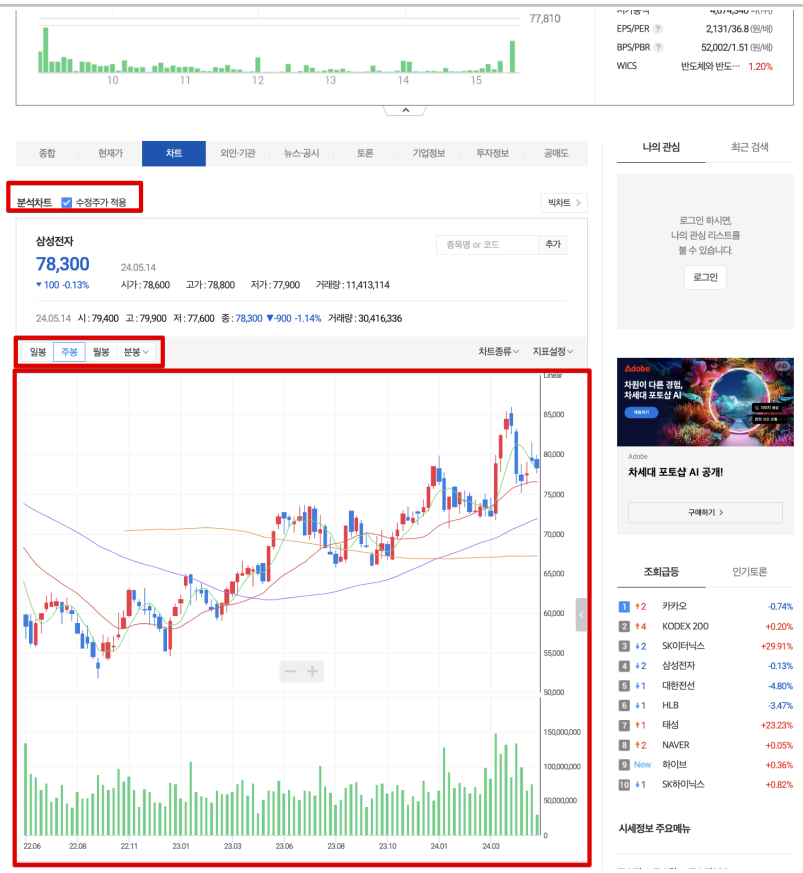


```
{
  "2024-05-05": [
    {
      "title": "\"폴더블폰=삼성전자 갤럭시S\" 공식 깨는 화웨이와 황사바람",
      "url": "http://v.daum.net/v/20240505093653690"
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {
      "title": "삼성전자, 가정의 달 맞아 '세상 편한 AI 라이프' 페스티벌 실시",
      "url": "http://v.daum.net/v/20240503114117645"
    }
  ],
  "2024-05-12": [
    {
      "title": "삼성전자, 베트남 총리에 \"수년간 연간 10억달러씩 투자\"",
      "url": "http://v.daum.net/v/20240512210047696"
    },
    {
```

1.2 다음 주가차트에서 데이터 수집하기 (15점)

삼성전자 코드: **005930**

daum.net 주가차트**URL**: https://finance.daum.net/quotes/A005930#chart



<그림*1*>

[과제 내용**]**

daum.net 주가차트**URL**에 접속하고 스크롤을 조금 내리면 **<**그림**1>** 과 같은 페이지를 확인할 수 있다.

현재 **<**그림**1>**은 수정주가 적용이 체크되어 있는 상태이고, 삼성전자에 대한 주봉에 대한 내역을 볼 수 있는 상태이다.

2024년 **4**월 **1**일부터 **2024**년 **4**월 **19**일까지의 주봉 **[**총 **3**개의 데이터**]** 를 수집하여 Array<StockData>형태의 json포맷으로 저장하는 코드를 작성하고 실제로 저장하여라. **[**파일이름**: stock.json]** (하단 결과예시 확인)

StockData에는 반드시 [date, tradePrice, openingPrice, highPrice, lowPrice, candleAccTradePrice] key는 들어가야 하며,

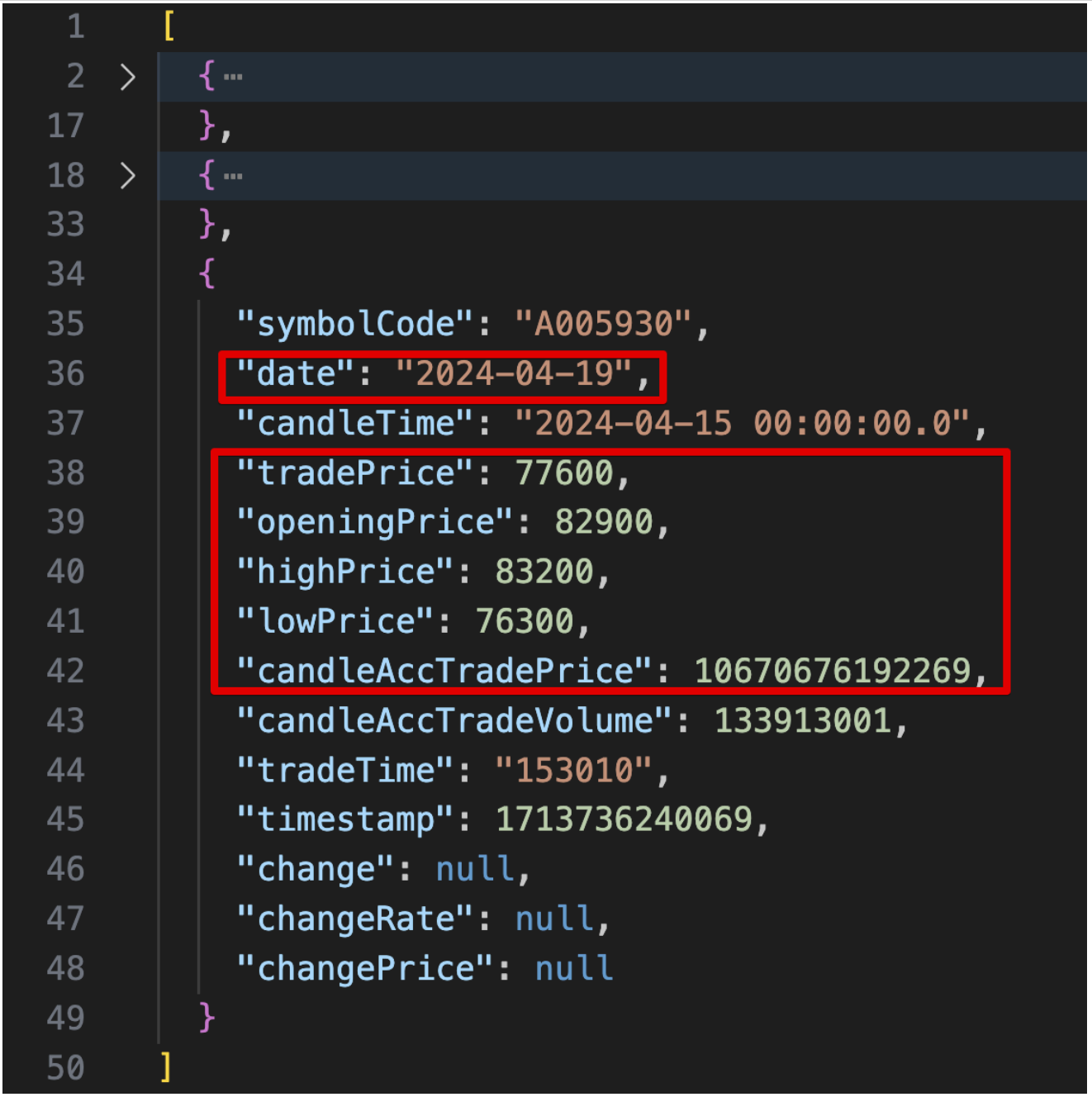
해당 key가 의미하는 바는 다음과 같다.

(date: 종가 날짜, tradePrice: 종가, openingPrice: 시가, highPrice: 고가, lowPrice: 저가, candleAccTradePrice: 거래대금)

[파일 저장 경로**]:**

- <제출자이름**>/homework-node/stock.json** 으로 저장

[결과예시 **- stock.json]**



위 빨간 박스에 해당하는 *key*는 필수, 그 외는 자유*(*넣으셔도 되고 빼도 됨*)*

2. React 응용

본 과제는 React를 자유롭게 응용하시면 됩니다. 컴포넌트 분리는 하셔도 되고 하지 않으셔도 됩니다. 그에 따른 점수에 대한 영향은 없습니다. 요구사항만 충족시켜주시면 됩니다.

작업디렉토리: **homework-react**
현재 프로젝트는 react-bootstrap이 설치되어있는 환경이다.

- src/components/Root.jsx 에는 UserApp과 StockApp이 각각 Rendering되고 있다.
- public/data 디렉토리에는 각각 stock.json과 news.json이 있다.

다음 조건들에 맞게 UserApp과 StockApp을 완성하시오.

** 사전지식 (**react**개발 서버를 실행한 후***<npm run start* 후>**)

- 해당 url(예:localhost:3000)에 public에 위치한 경로를 입력하면 public에 위치한 static file을 반환
  ⇒ **localhost:3000/data/stock.json**에 **HTTP Request**시 stock.json 내용 반환

2.1.1. StockApp UI 완성하기 (20점)

작업 Component: src/components/**StockApp.jsx**

- **react-bootstrap table Component:**
  https://react-bootstrap.netlify.app/docs/components/table

[과제 내용**]**
react-bootstrap table을 사용하여 다음 <그림1>에 해당하는 UI를 완성하시오.
부트스트랩 Table Component에 다음 속성에 해당하는 props만 true로 설정하시고 나머지 스타일링은 자유입니다.

<true로 설정할 **props>**

- **striped**
- **bordered**
- **hover**

<그림**1>**

[이미지 캡쳐 저장 경로**]:**
- <제출자 이름**>/2.1.1.jpg** 로 저장 (jpg, png 등 확장자는 상관없음)

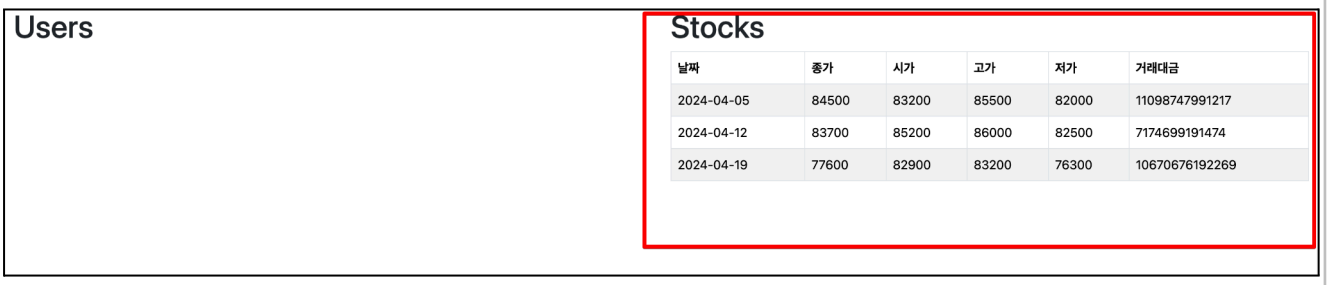
2.1.2. StockApp 데이터 통신하여 Rendering하기 (10점)

작업 Component: src/components/**StockApp.jsx**

[과제 내용**]**

- /data/stock.json 에 요청을 보내어 주별 **stock Data**를 받아오고**,** 해당 내용을 **table-row**로 **Rendering**하시오**.**
- 테이블에서 th(테이블 헤더)에 대응되는 stock.json 의 key는 다음과 같습니다.
    - [날짜: date, 종가: tradePrice, 시가: openingPrice, 고가: highPrice, 저가: lowPrice, 거래대금: candleAccTradePrice]
- 결과는 아래 *[결과 예시]* 확인

[결과 예시**]**



[이미지 캡쳐 저장 경로**]:**
- <제출자 이름**>/2.1.2.jpg** 로 저장 (jpg, png 등 확장자는 상관없음)

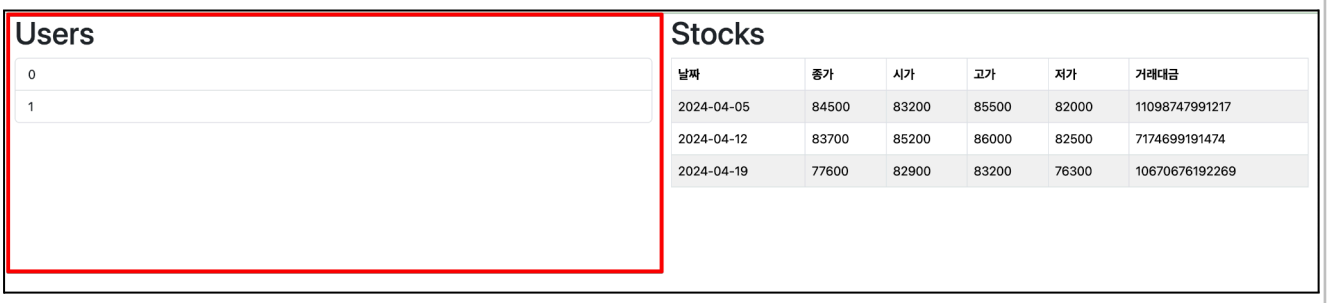
2.2.1. UserApp UI완성하기 (20점)

작업 Component: src/components/**UserApp.jsx**

- **react-bootstrap List Group Component:**
  https://react-bootstrap.netlify.app/docs/components/list-group#actionable-items

[과제 내용**]**
1. react-bootstrap ListGroup 컴포넌트를 사용하여 다음 **[**결과 예시**]**에 해당하는 UI를 완성하시오.
    - 해당하는 데이터는 임의로 하셔도 무방합니다.
      (ListGroup과 ListGroup.Item 만 사용하시면, 세부 스타일링은 과제점수에 반영 되지 않습니다.)

[결과 예시**]**



[이미지 캡쳐 저장 경로**]:**
- <제출자 이름**>/2.2.1.jpg** 로 저장 (jpg, png 등 확장자는 상관없음)

2.2.2. UserApp 데이터 통신 및 Rendering (10점)

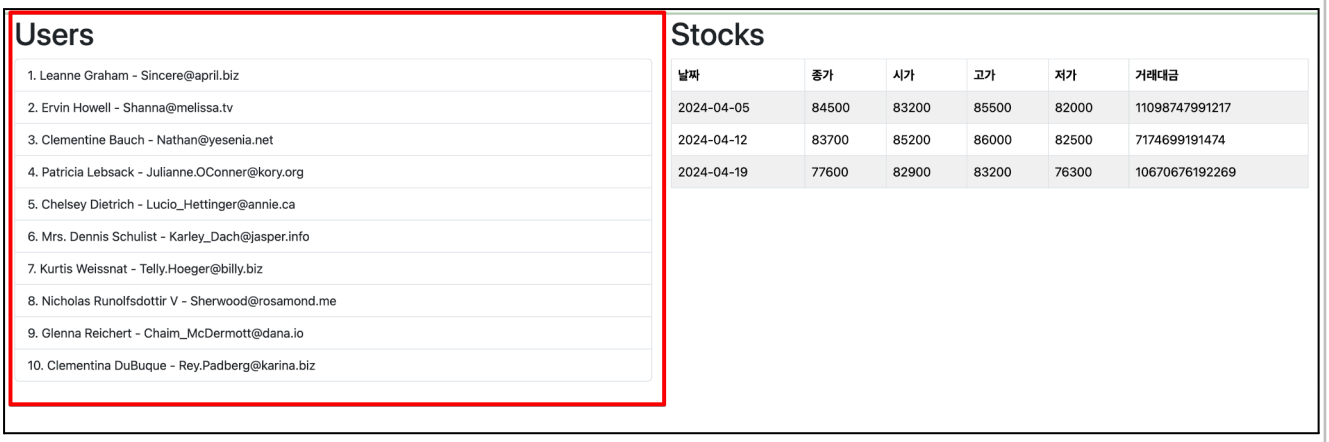
작업 Component: src/components/**UserApp.jsx**

- 통신할 user URL: https://jsonplaceholder.typicode.com/users

[과제 내용**]**

- 위 링크(https://jsonplaceholder.typicode.com/users)에 요청을 보내면, 총 10명의 user에 대한 data를 불러온다.
  위 링크에 HTTP요청을 하고 state에 저장한 후,
  각각 <ListGroup.Item> 컴포넌트에서 다음형태로 Rendering하시오. ({user.id}. {user.name} - {user.email})
  ⇒ 데이터에서 [user.id, user.name, user.email] 속성이 필요
  ⇒ **[**결과 예시**]** 참조

[결과 예시**]**



[이미지 캡쳐 저장 경로**]:**
- <제출자 이름**>/2.2.2.jpg** 로 저장 (jpg, png 등 확장자는 상관없음)

2.2.3. UserApp 데이터 통신 및 고급 응용 (10점)

작업 Component: src/components/**UserApp.jsx**

- **React Bootstrap Modal Component:**
  https://react-bootstrap.netlify.app/docs/components/modal
- 통신할 URL:
  - https://jsonplaceholder.typicode.com/todos
  - https://jsonplaceholder.typicode.com/posts

[사전 지식**]**

- 위 링크는 모두 url parameter(query string을 입력하여 세부 조건을 검색할 수 있다.
  - userId가 2에 해당하는 user의 todos정보를 가져오려면 다음과 같이 get 요청을 보내면 된다.
    https://jsonplaceholder.typicode.com/todos?userId=2
  - userId가 8에 해당하는 user의 posts정보를 가져오려면 다음과 같이 get 요청을 보내면 된다.
    https://jsonplaceholder.typicode.com/posts?userId=8

[과제 내용**]**

- (2.4)에서 작업한 **User(<ListGroup.Item>)**를 클릭하면 **Modal**창이 보이도록 작업하시오.
- 해당 **Modal**의 **Title**은 클릭한 **user**의 이름이 보이도록 하시오**.**

예: *<Modal.Header><Modal.Title> {클릭한 user의 정보} </Modal.Title></Modal.Header>*

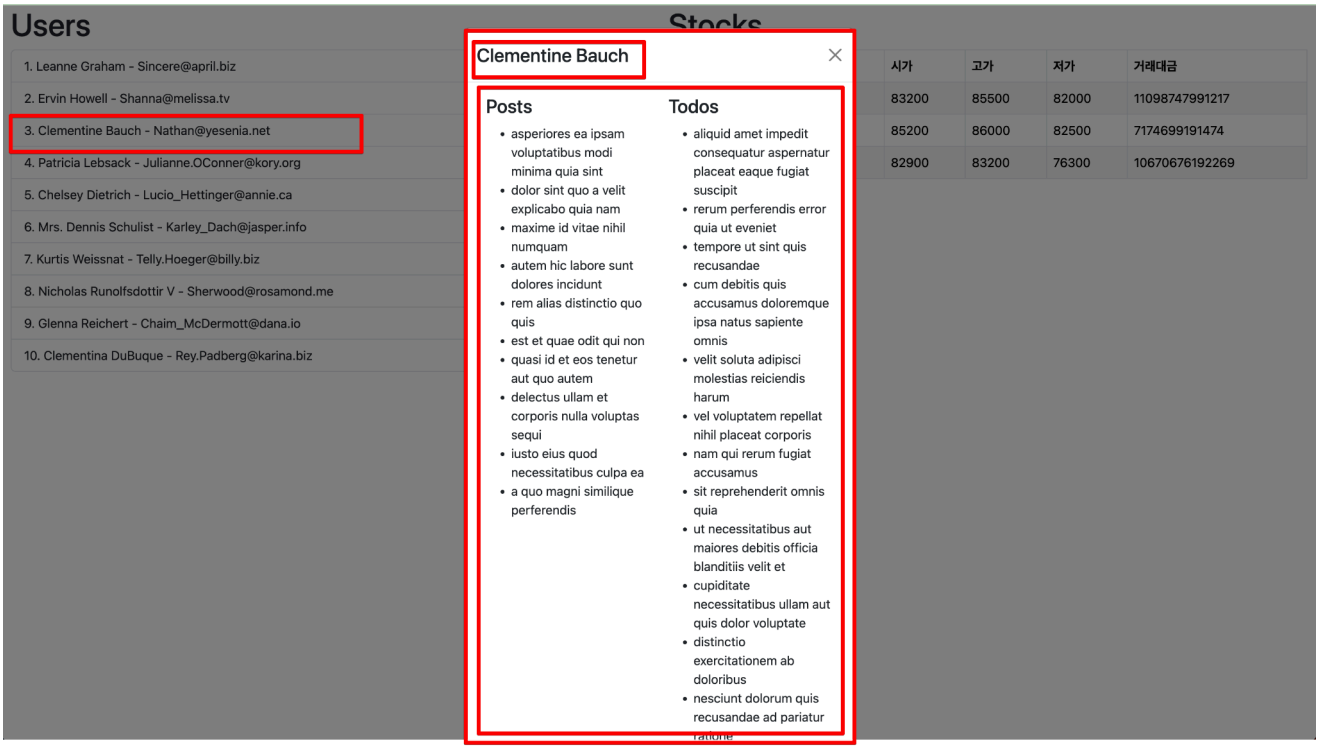
- 본문에는 **user**가 작성한 **post**와 **user**의 **todo** 가 나열되도록 하시오**. (**스타일링은 신경쓰지 않으셔도 됩니다**.)**

  예: (<Modal.Body>{본문내용}}</Modal.Body>

- **Modal** 우상단 **X** 버튼을 클릭하면 **Modal**이 보이지 않도록 하시오**.**
- **[**결과 예시**]**를 참조하시오.

[결과 예시**]**

현재 3번 유저 클릭시 보이는 modal 창입니다.



[이미지 캡쳐 저장 경로**]:**

- <제출자 이름**>/2.2.3.jpg** 로 저장 (jpg, png 등 확장자는 상관없음)