for spring project based on tomcat server

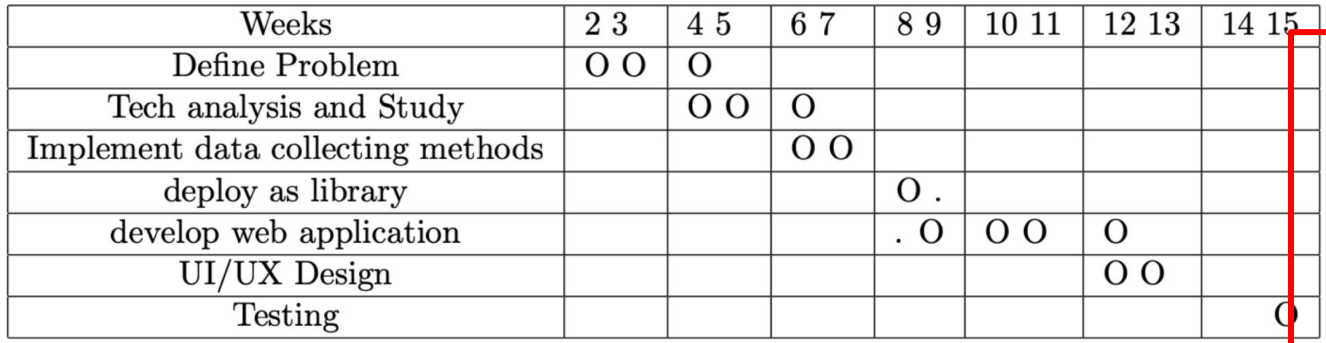
Visualize Web Thread Pool

Contents

Part 1

Milestones

Roll	Work
Backend	Complete implementation
Frontend	finished studying react, implementation is 90% done.



Weeks	2 3	4 5	6 7	8 9	10 11	12 13	14 15
Define Problem	O O	O					
Tech analysis and Study		O O	O				
Implement data collecting methods			O O				
deploy as library				O .			
develop web application				. O	O O	O	
UI/UX Design						O O	
Testing							O

Table 5: Weekly Schedule

Each Member's Role

Kim junyong

- complete backend development and testing api
- handle issues

Yoon Jaehwan

- complete frontend development
- design main views

Han Yongjun

- complete frontend development
- design main views
- testing

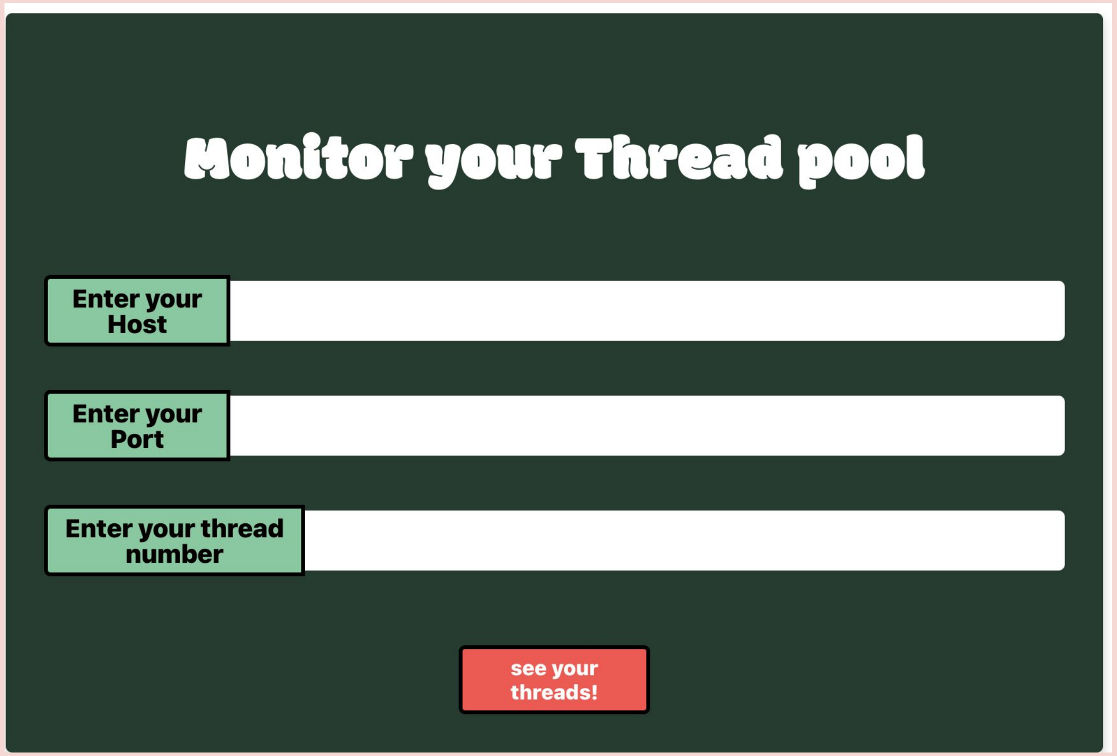
Kim kyeong hyeon

- complete frontend development
- design main views

Part 2

Implementation

제목을 입력하세요


Monitor your Thread pool

Enter your Host

Enter your Port

Enter your thread number

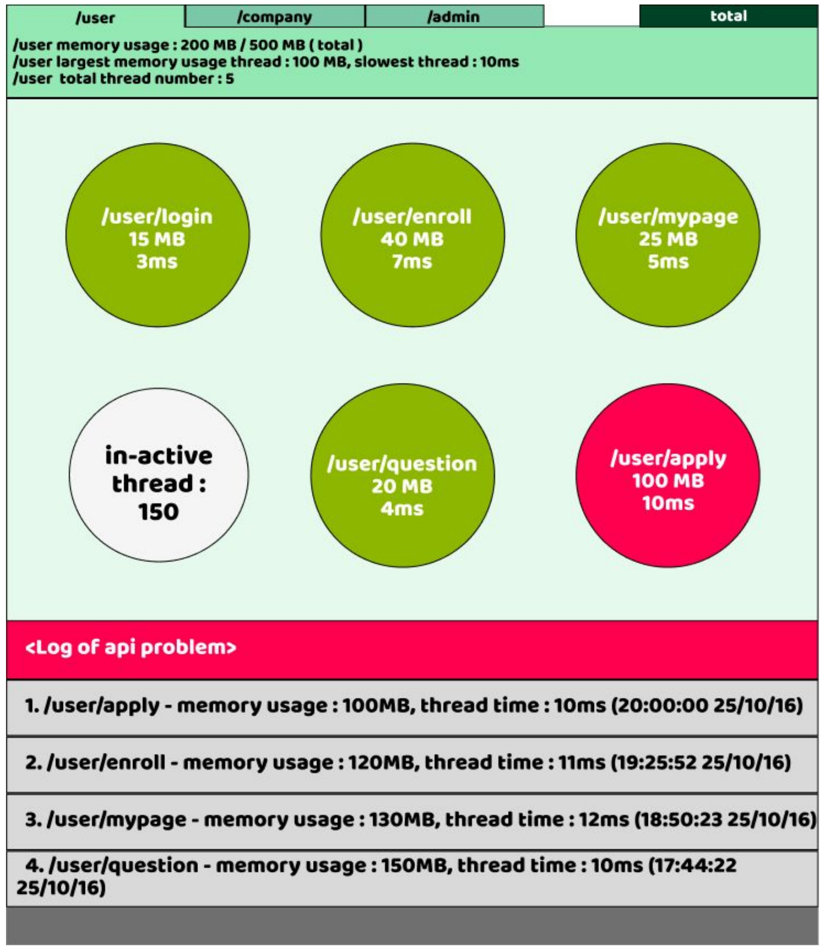
see your threads!

First Page

ıt

Part 2

제목을 입력하세요



/user memory usage : 200 MB / 500 MB (total)
/user largest memory usage thread : 100 MB, slowest thread : 10ms
/user total thread number : 5

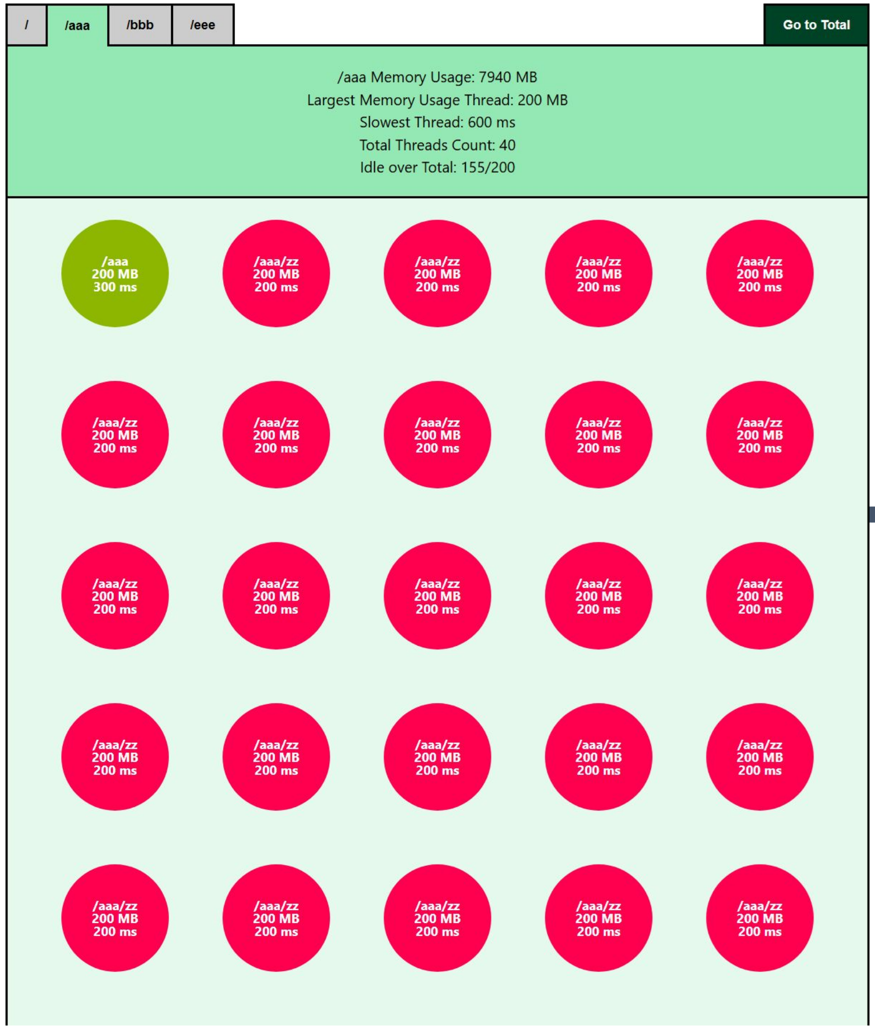
1. /user/apply - memory usage : 100MB, thread time : 10ms (20:00:00 25/10/16)
2. /user/enroll - memory usage : 120MB, thread time : 11ms (19:25:52 25/10/16)
3. /user/mypage - memory usage : 130MB, thread time : 12ms (18:50:23 25/10/16)
4. /user/question - memory usage : 150MB, thread time : 10ms (17:44:22 25/10/16)

Main Page
Figma

ıt

Part 2

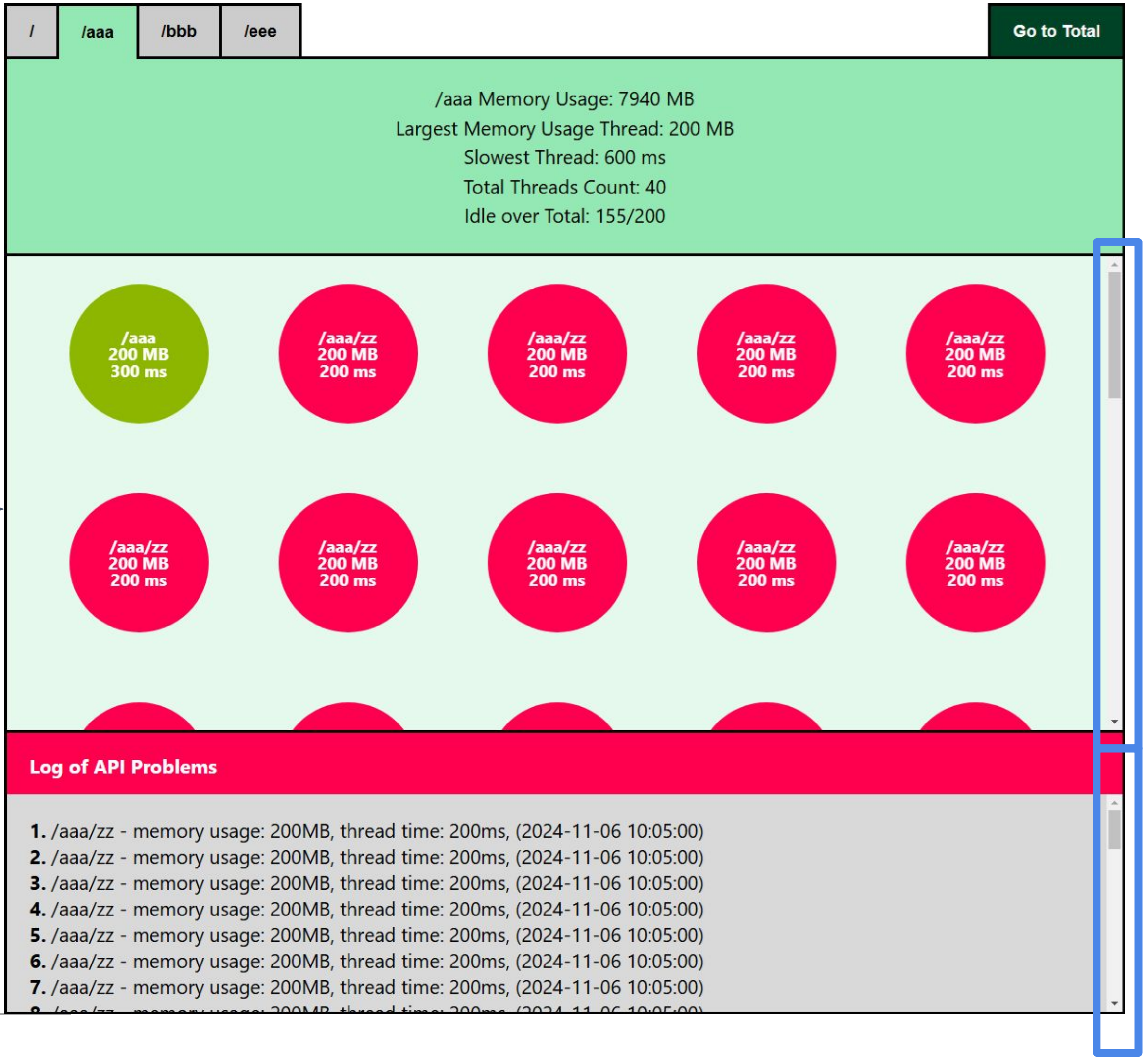
제목을 입력하세요



/aaa Memory Usage: 7940 MB
Largest Memory Usage Thread: 200 MB
Slowest Thread: 600 ms
Total Threads Count: 40
Idle over Total: 155/200

```
height: 300px;
overflow-y: auto;
```

```
height: 124px;
overflow-y: auto;
```



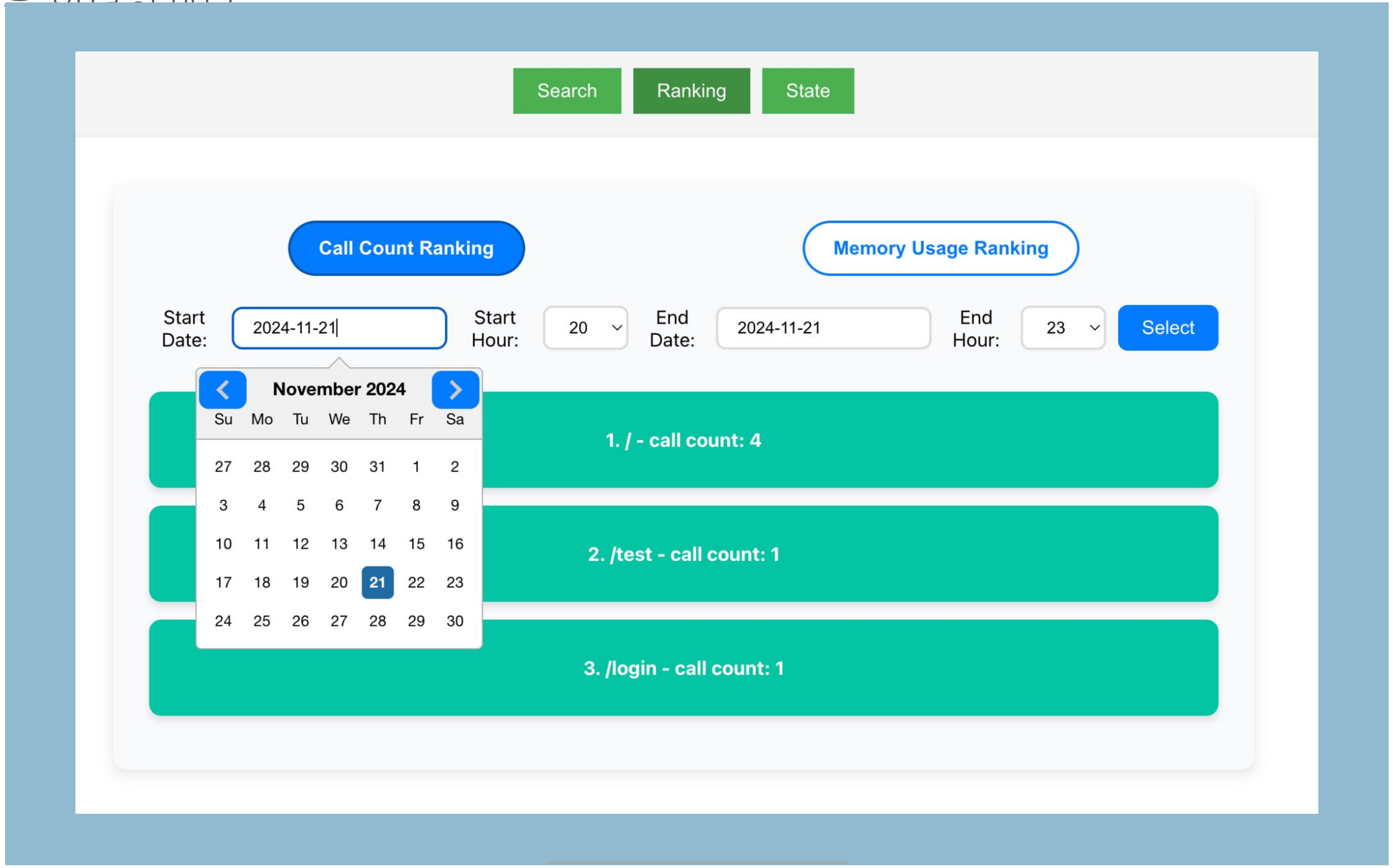
/aaa Memory Usage: 7940 MB
Largest Memory Usage Thread: 200 MB
Slowest Thread: 600 ms
Total Threads Count: 40
Idle over Total: 155/200

Log of API Problems

1. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
2. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
3. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
4. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
5. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
6. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
7. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)
8. /aaa/zz - memory usage: 200MB, thread time: 200ms, (2024-11-06 10:05:00)

Main Page
Current (with Scroll Implementation)

제목을 입력하세요



1. / - call count: 4

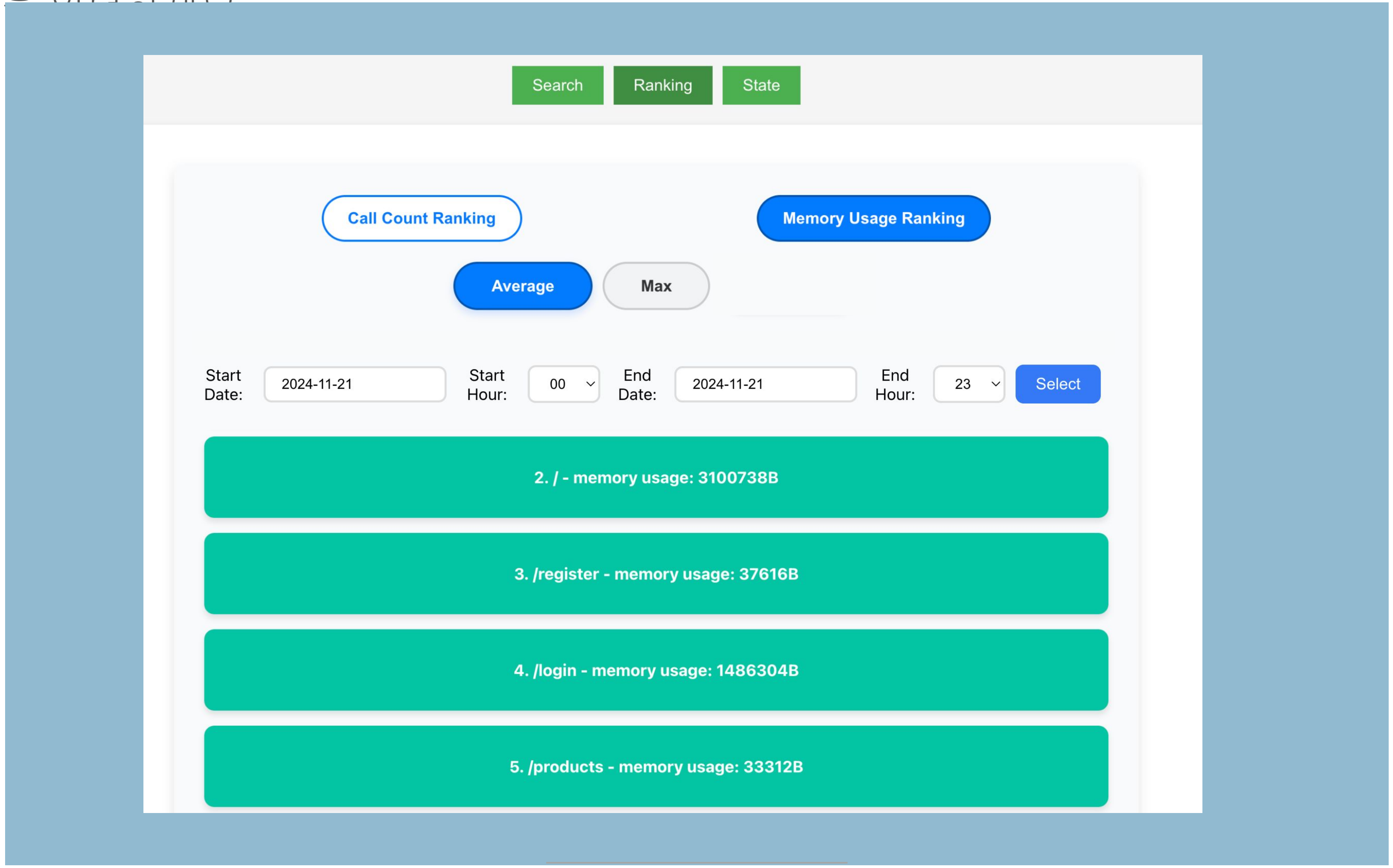
2. /test - call count: 1

3. /login - call count: 1

Summary Page

Part 2

제목을 입력하세요



Search Ranking State

Call Count Ranking Memory Usage Ranking

Average Max

Start Date: 2024-11-21 Start Hour: 00 End Date: 2024-11-21 End Hour: 23 Select

2. / - memory usage: 3100738B

3. /register - memory usage: 37616B

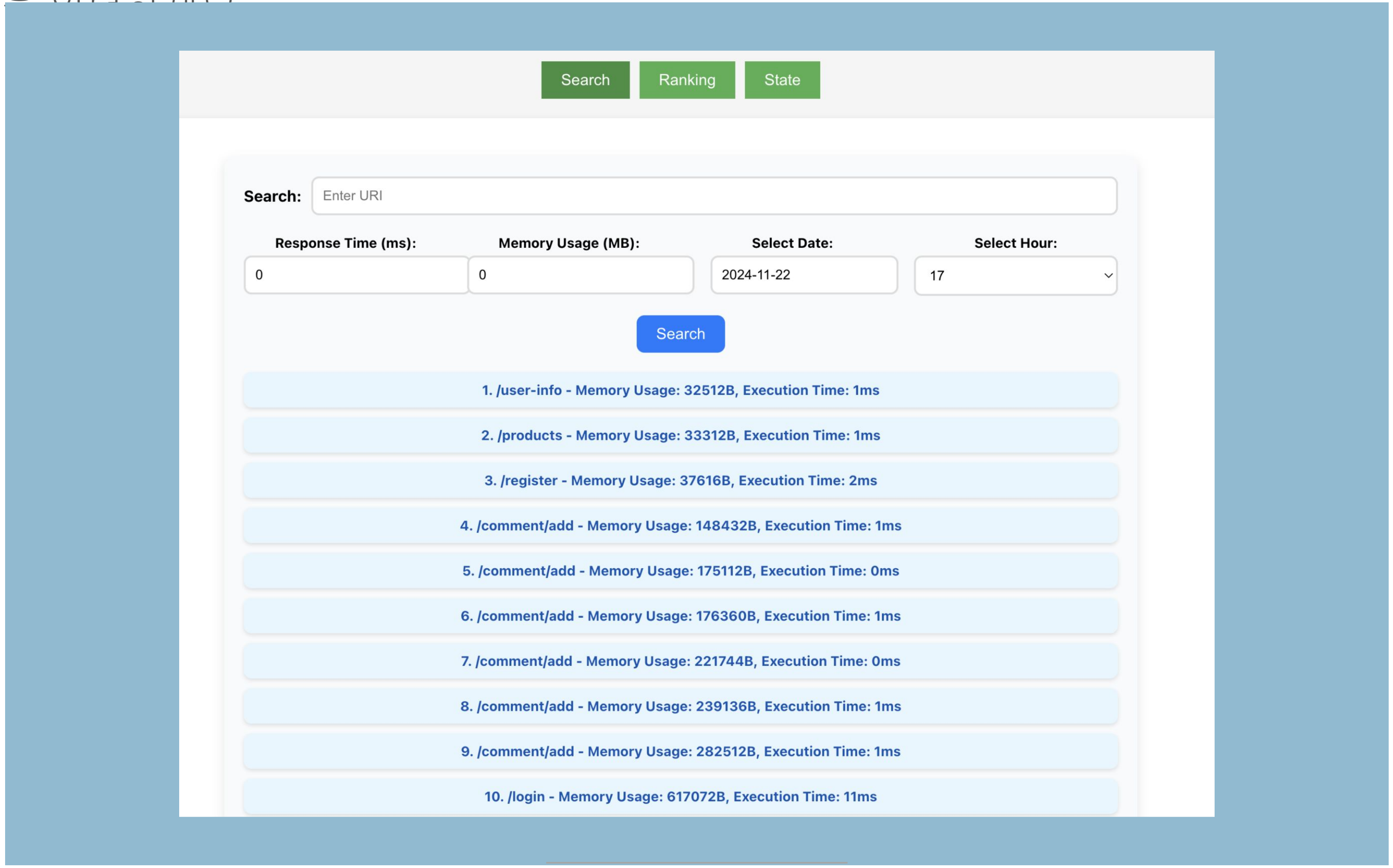
4. /login - memory usage: 1486304B

5. /products - memory usage: 33312B

Summary Page

nt

제목



Search: Enter URI

Response Time (ms): 0
Memory Usage (MB): 0
Select Date: 2024-11-22
Select Hour: 17

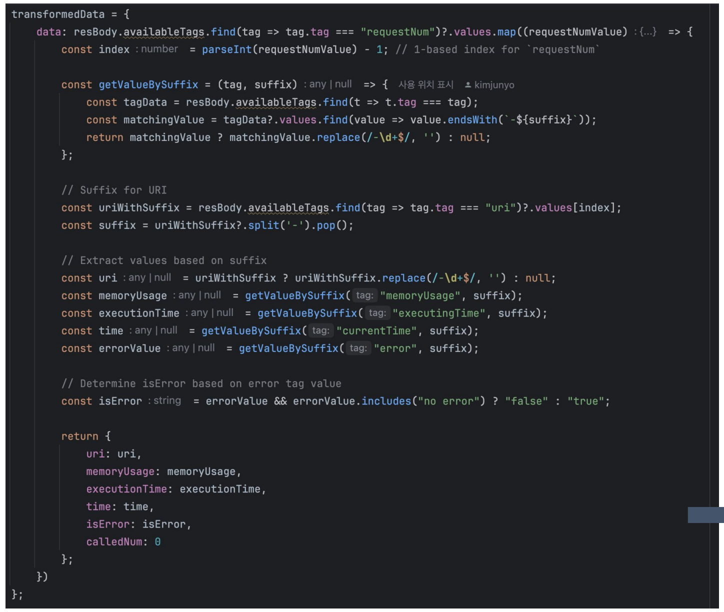
1. /user-info - Memory Usage: 32512B, Execution Time: 1ms
2. /products - Memory Usage: 33312B, Execution Time: 1ms
3. /register - Memory Usage: 37616B, Execution Time: 2ms
4. /comment/add - Memory Usage: 148432B, Execution Time: 1ms
5. /comment/add - Memory Usage: 175112B, Execution Time: 0ms
6. /comment/add - Memory Usage: 176360B, Execution Time: 1ms
7. /comment/add - Memory Usage: 221744B, Execution Time: 0ms
8. /comment/add - Memory Usage: 239136B, Execution Time: 1ms
9. /comment/add - Memory Usage: 282512B, Execution Time: 1ms
10. /login - Memory Usage: 617072B, Execution Time: 11ms

Summary Page

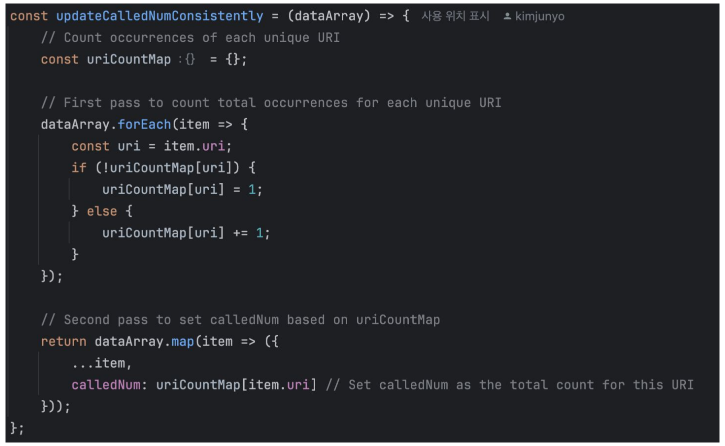
Backend Implementation

“memory usage”
{
    “request1”
    “request2”
    “request3”
}
“executionTime”
{
    “request1”
    “request2”
    “request3”
} ….

To transform actuator metrics into useful data



```
transformedData = {
    data: resBody.availableTags.find(tag => tag.tag === "requestNum")?.values.map((requestNumValue) :{...}  => {
        const index :number  = parseInt(requestNumValue) - 1; // 1-based index for `requestNum`

        const getValueBySuffix = (tag, suffix) :any | null  => {  사용 위치 표시  kimjunyo
            const tagData = resBody.availableTags.find(t => t.tag === tag);
            const matchingValue = tagData?.values.find(value => value.endsWith(`-${suffix}`));
            return matchingValue ? matchingValue.replace(/-\d+$/, '') : null;
        };

        // Suffix for URI
        const uriWithSuffix = resBody.availableTags.find(tag => tag.tag === "uri")?.values[index];
        const suffix = uriWithSuffix?.split('-').pop();

        // Extract values based on suffix
        const uri :any | null  = uriWithSuffix ? uriWithSuffix.replace(/-\d+$/, '') : null;
        const memoryUsage :any | null  = getValueBySuffix( tag: "memoryUsage", suffix);
        const executionTime :any | null  = getValueBySuffix( tag: "executingTime", suffix);
        const time :any | null  = getValueBySuffix( tag: "currentTime", suffix);
        const errorValue :any | null  = getValueBySuffix( tag: "error", suffix);

        // Determine isError based on error tag value
        const isError :string  = errorValue && errorValue.includes("no error") ? "false" : "true";

        return {
            uri: uri,
            memoryUsage: memoryUsage,
            executionTime: executionTime,
            time: time,
            isError: isError,
            calledNum: 0
        };
    })
};
```

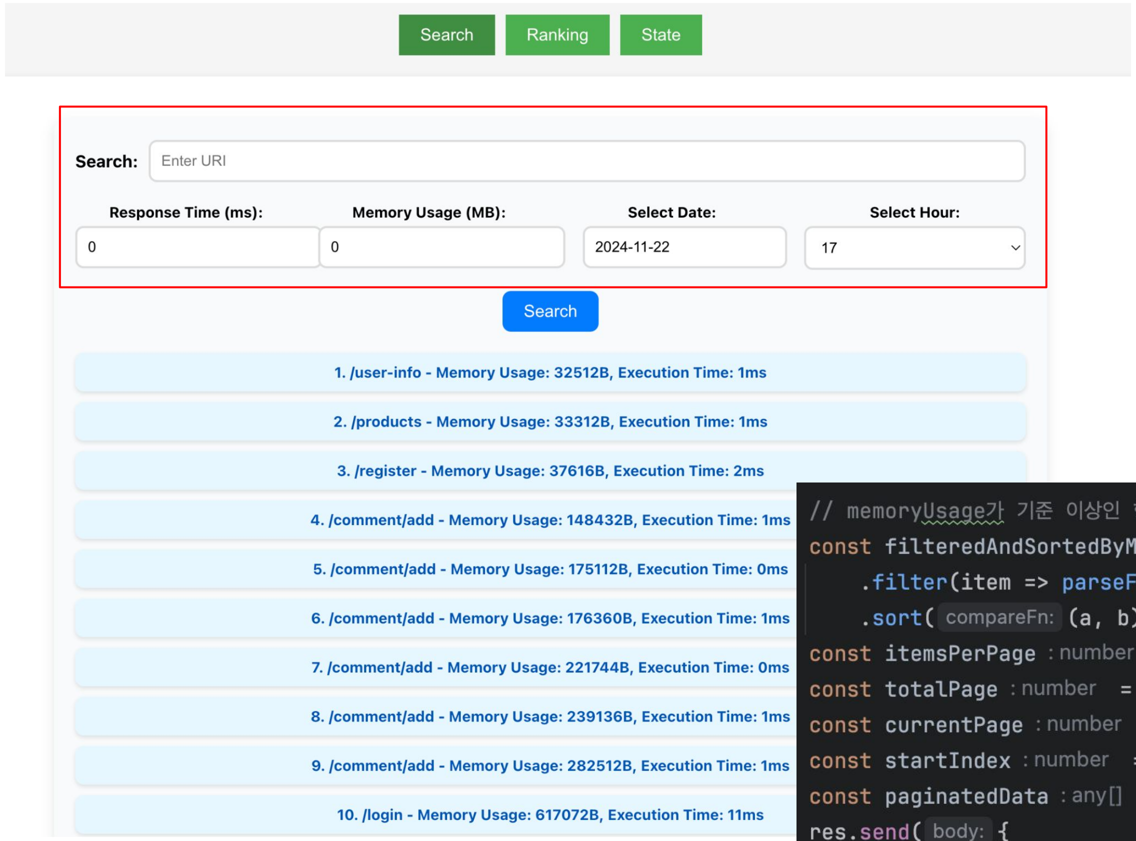


```
const updateCalledNumConsistently = (dataArray) => {  사용 위치 표시  kimjunyo
    // Count occurrences of each unique URI
    const uriCountMap :{}  = {};

    // First pass to count total occurrences for each unique URI
    dataArray.forEach(item => {
        const uri = item.uri;
        if (!uriCountMap[uri]) {
            uriCountMap[uri] = 1;
        } else {
            uriCountMap[uri] += 1;
        }
    });

    // Second pass to set calledNum based on uriCountMap
    return dataArray.map(item => ({
        ...item,
        calledNum: uriCountMap[item.uri] // Set calledNum as the total count for this URI
    }));
};
```

```
[
  {
    "uri": "/test",
    "memoryUsage": "272880",
    "executionTime": "2ms",
    "time":
"2024-11-22T00:13:48.989957",
    "isError": "false",
    "calledNum": 1
  },
  {
    "uri": "/",
    "memoryUsage": "398872",
    "executionTime": "10020ms",
    "time":
"2024-11-22T00:13:39.688466",
    "isError": "false",
    "calledNum": 1
  }
]
```

Part 2

Backend Implementation

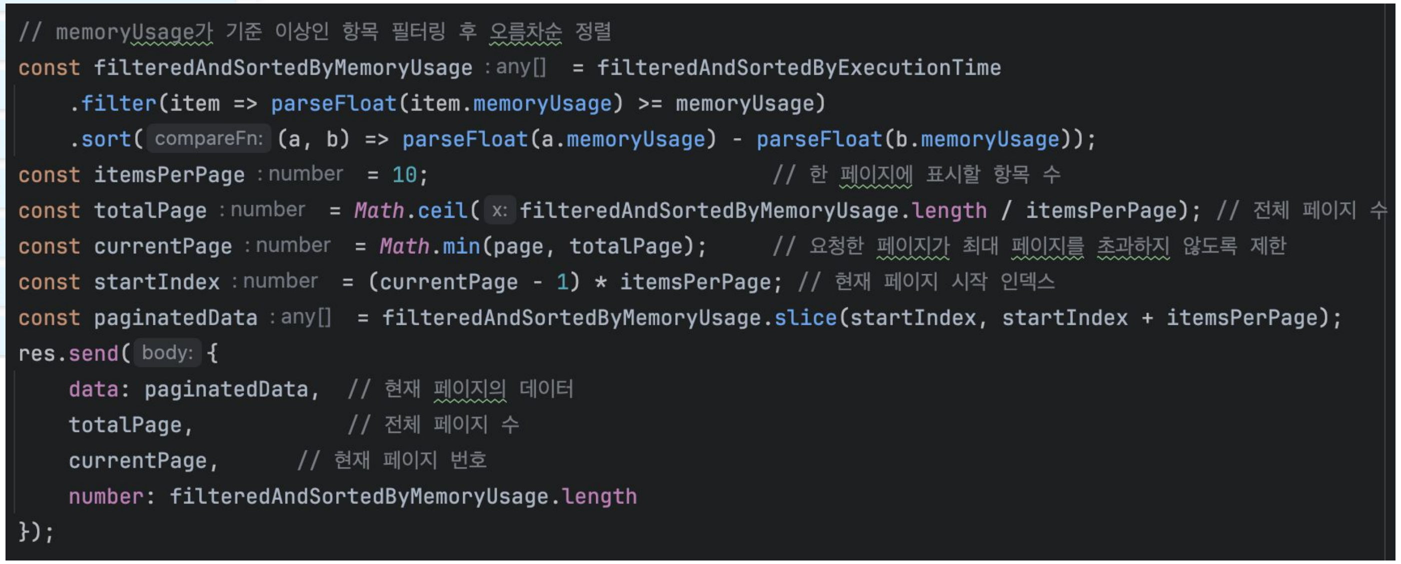


Search: Enter URI

Response Time (ms):	Memory Usage (MB):	Select Date:	Select Hour:
0	0	2024-11-22	17

1. /user-info - Memory Usage: 32512B, Execution Time: 1ms
2. /products - Memory Usage: 33312B, Execution Time: 1ms
3. /register - Memory Usage: 37616B, Execution Time: 2ms
4. /comment/add - Memory Usage: 148432B, Execution Time: 1ms
5. /comment/add - Memory Usage: 175112B, Execution Time: 0ms
6. /comment/add - Memory Usage: 176360B, Execution Time: 1ms
7. /comment/add - Memory Usage: 221744B, Execution Time: 0ms
8. /comment/add - Memory Usage: 239136B, Execution Time: 1ms
9. /comment/add - Memory Usage: 282512B, Execution Time: 1ms
10. /login - Memory Usage: 617072B, Execution Time: 11ms

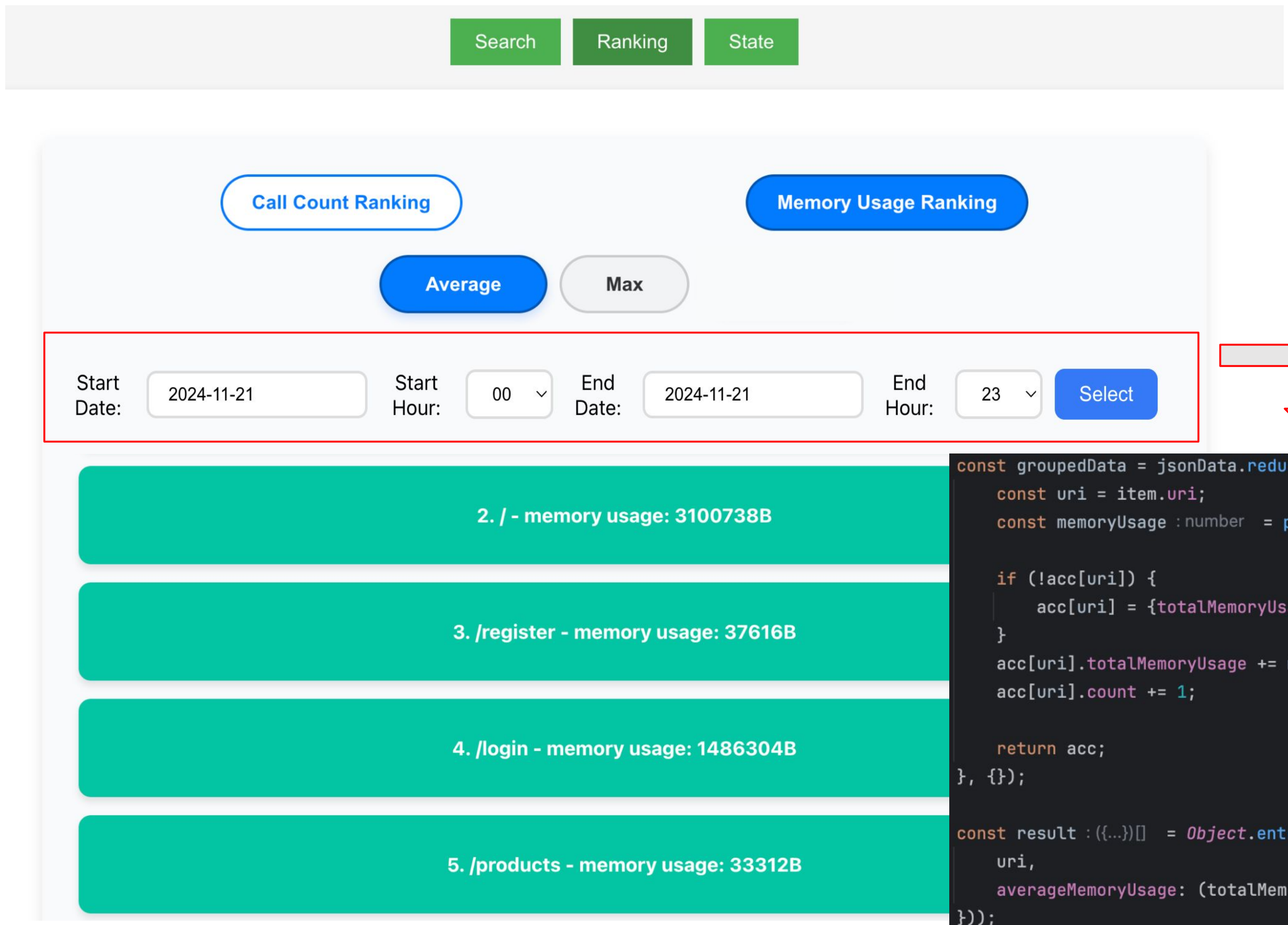
To show the api data that matches the item you are searching for



```
// memoryUsage가 기준 이상인 항목 필터링 후 오름차순 정렬
const filteredAndSortedByMemoryUsage :any[]  = filteredAndSortedByExecutionTime
    .filter(item => parseFloat(item.memoryUsage) >= memoryUsage)
    .sort( compareFn: (a, b) => parseFloat(a.memoryUsage) - parseFloat(b.memoryUsage));
const itemsPerPage :number  = 10;                          // 한 페이지에 표시할 항목 수
const totalPage :number  = Math.ceil( x: filteredAndSortedByMemoryUsage.length / itemsPerPage); // 전체 페이지 수
const currentPage :number  = Math.min(page, totalPage);      // 요청한 페이지가 최대 페이지를 초과하지 않도록 제한
const startIndex :number  = (currentPage - 1) * itemsPerPage; // 현재 페이지 시작 인덱스
const paginatedData :any[]  = filteredAndSortedByMemoryUsage.slice(startIndex, startIndex + itemsPerPage);
res.send( body: {
    data: paginatedData,  // 현재 페이지의 데이터
    totalPage,            // 전체 페이지 수
    currentPage,       // 현재 페이지 번호
    number: filteredAndSortedByMemoryUsage.length
});
```

Part 2

Backend Implementation



Start Date: 2024-11-21 Start Hour: 00 End Date: 2024-11-21 End Hour: 23

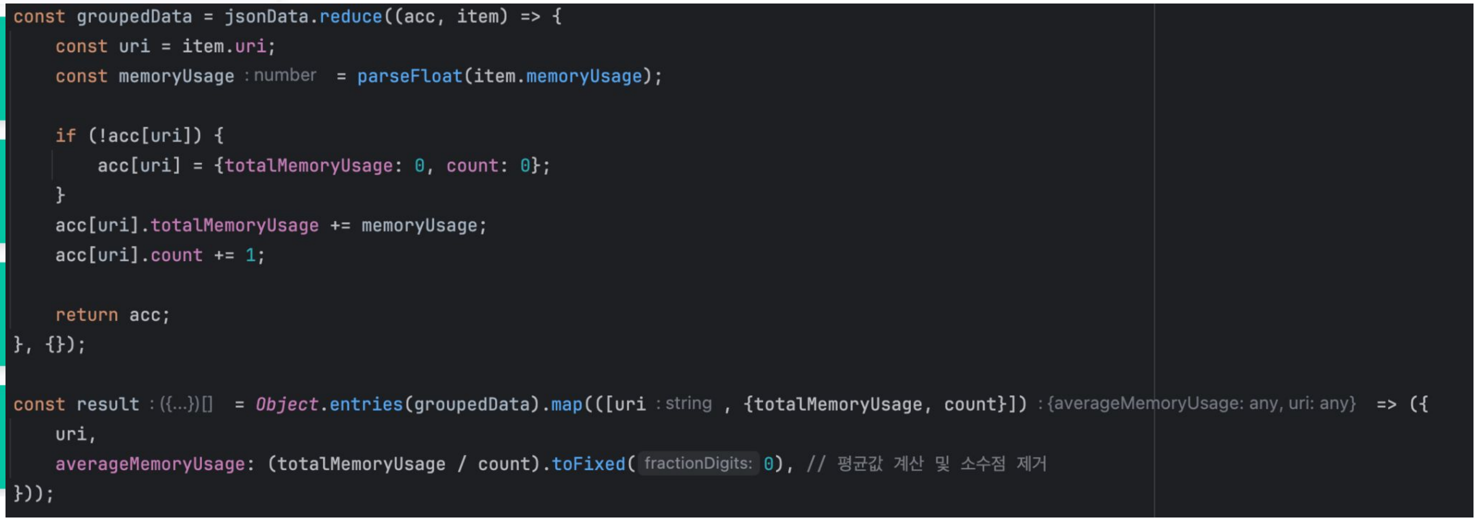
2. / - memory usage: 3100738B

3. /register - memory usage: 37616B

4. /login - memory usage: 1486304B

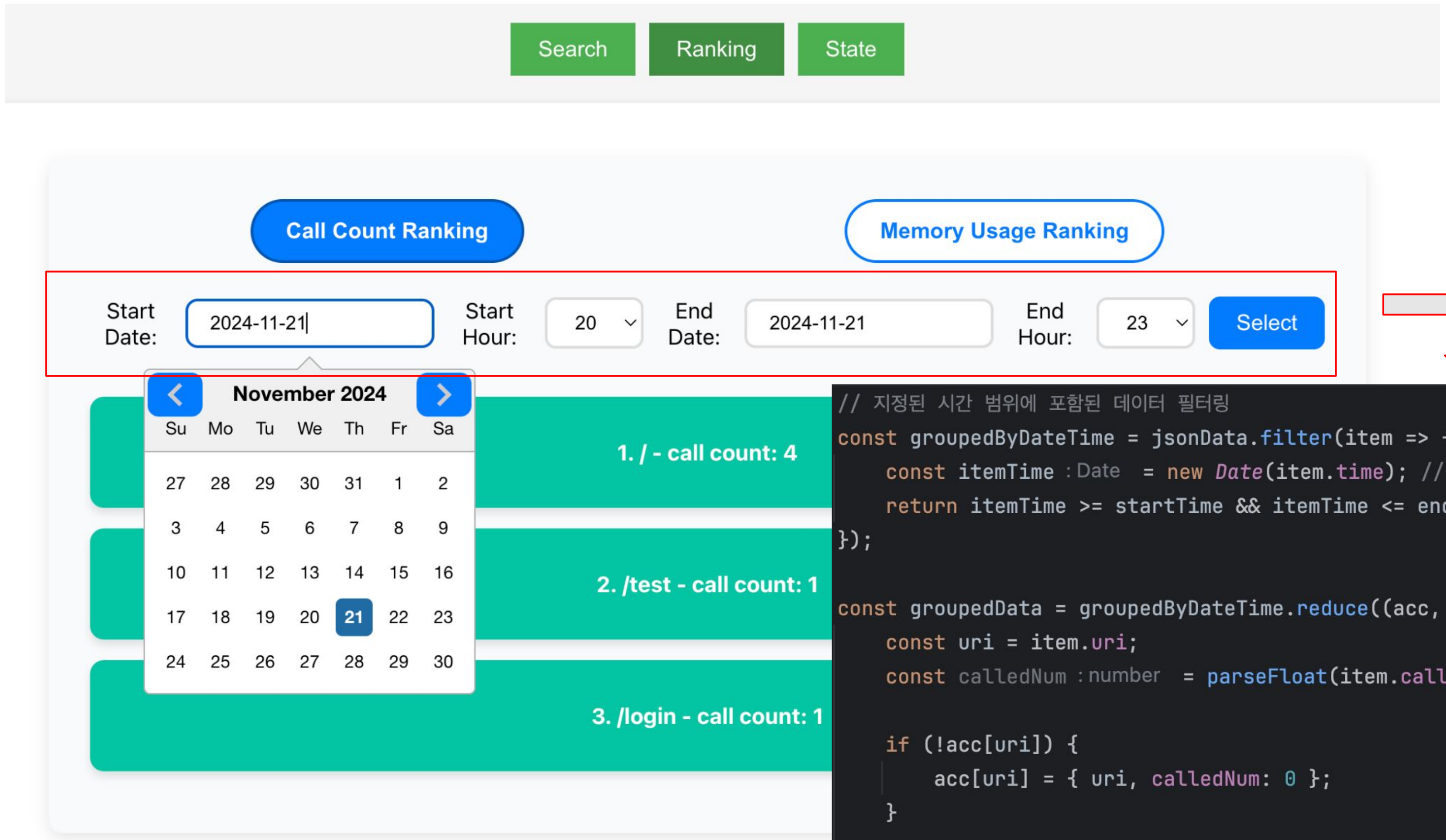
5. /products - memory usage: 33312B

To show the ranking of data based on the memory usage over a given period of time



```
const groupedData = jsonData.reduce((acc, item) => {
    const uri = item.uri;
    const memoryUsage :number  = parseFloat(item.memoryUsage);

    if (!acc[uri]) {
        acc[uri] = {totalMemoryUsage: 0, count: 0};
    }
    acc[uri].totalMemoryUsage += memoryUsage;
    acc[uri].count += 1;

    return acc;
}, {});

const result :({...})[]  = Object.entries(groupedData).map(([uri :string , {totalMemoryUsage, count}]) :{averageMemoryUsage: any, uri: any}  => ({
    uri,
    averageMemoryUsage: (totalMemoryUsage / count).toFixed( fractionDigits: 0), // 평균값 계산 및 소수점 제거
}));
```

Backend Implementation



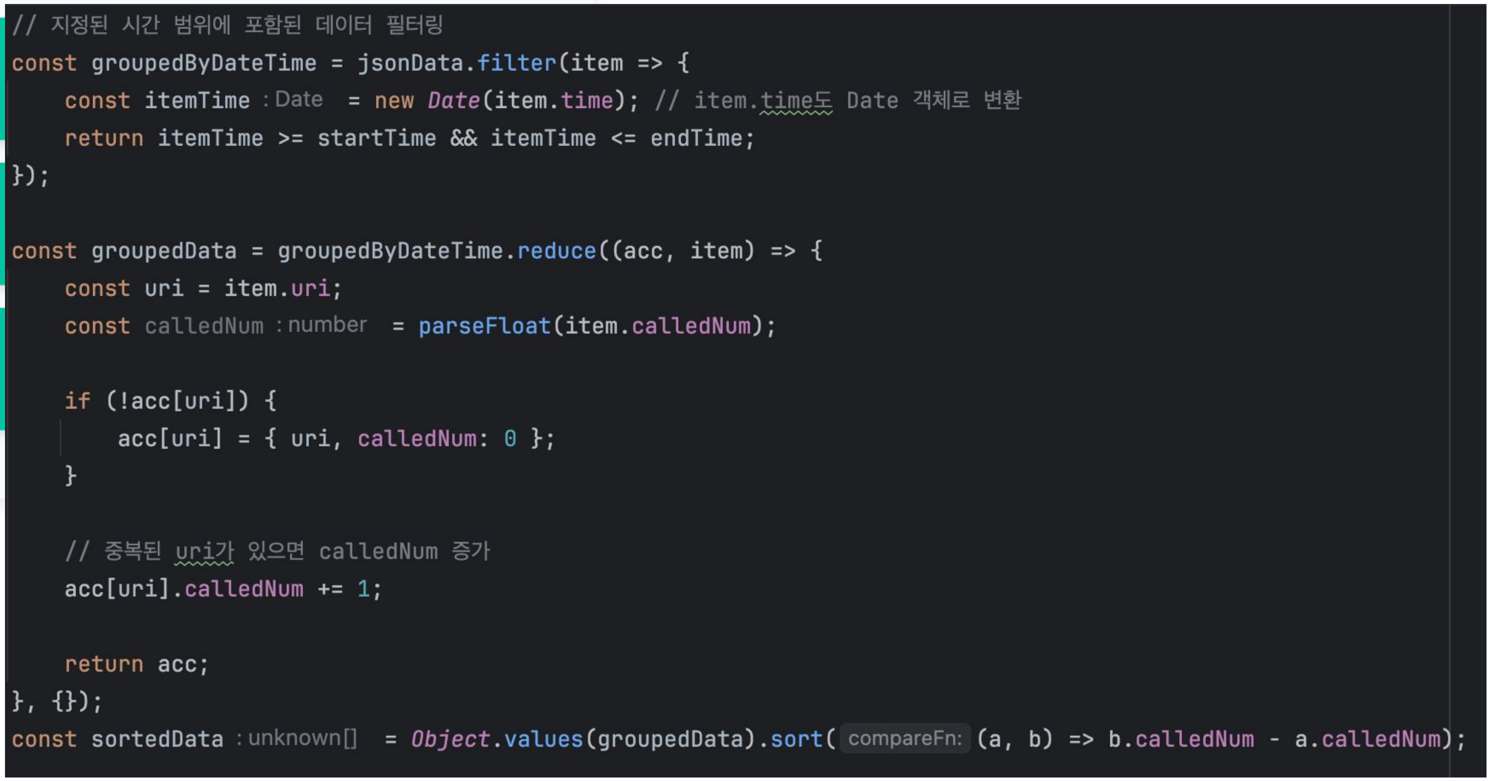
Start Date: 2024-11-21 Start Hour: 20 End Date: 2024-11-21 End Hour: 23

1. / - call count: 4

2. /test - call count: 1

3. /login - call count: 1

To show the ranking of data based on the callCount over a given period of time



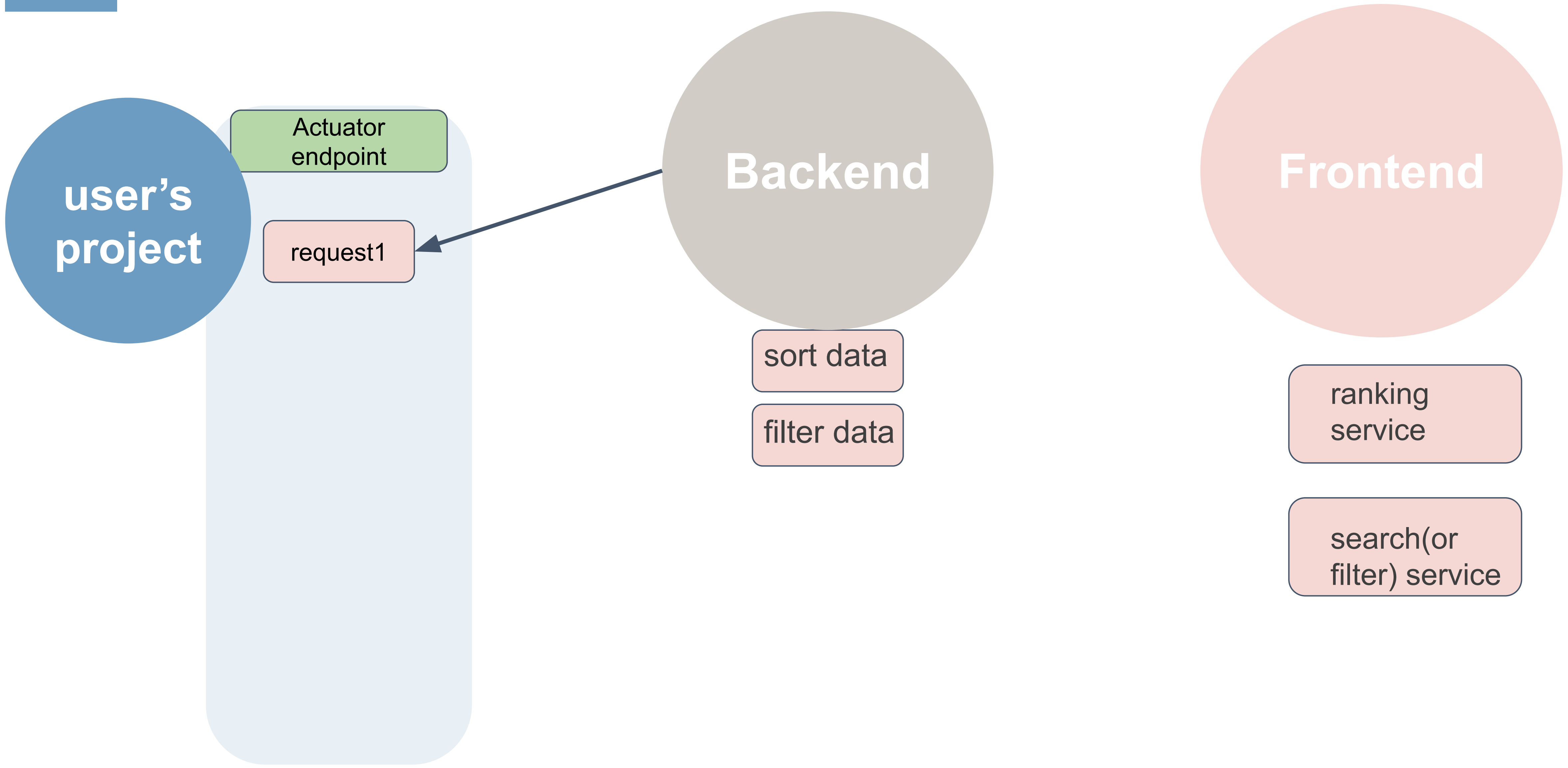
```
// 지정된 시간 범위에 포함된 데이터 필터링
const groupedByDateTime = jsonData.filter(item => {
    const itemTime :Date  = new Date(item.time); // item.time도 Date 객체로 변환
    return itemTime >= startTime && itemTime <= endTime;
});

const groupedData = groupedByDateTime.reduce((acc, item) => {
    const uri = item.uri;
    const calledNum :number  = parseFloat(item.calledNum);

    if (!acc[uri]) {
        acc[uri] = { uri, calledNum: 0 };
    }

    // 중복된 uri가 있으면 calledNum 증가
    acc[uri].calledNum += 1;

    return acc;
}, {});
const sortedData :unknown[]  = Object.values(groupedData).sort( compareFn: (a, b) => b.calledNum - a.calledNum);
```

Part 3

Challenge

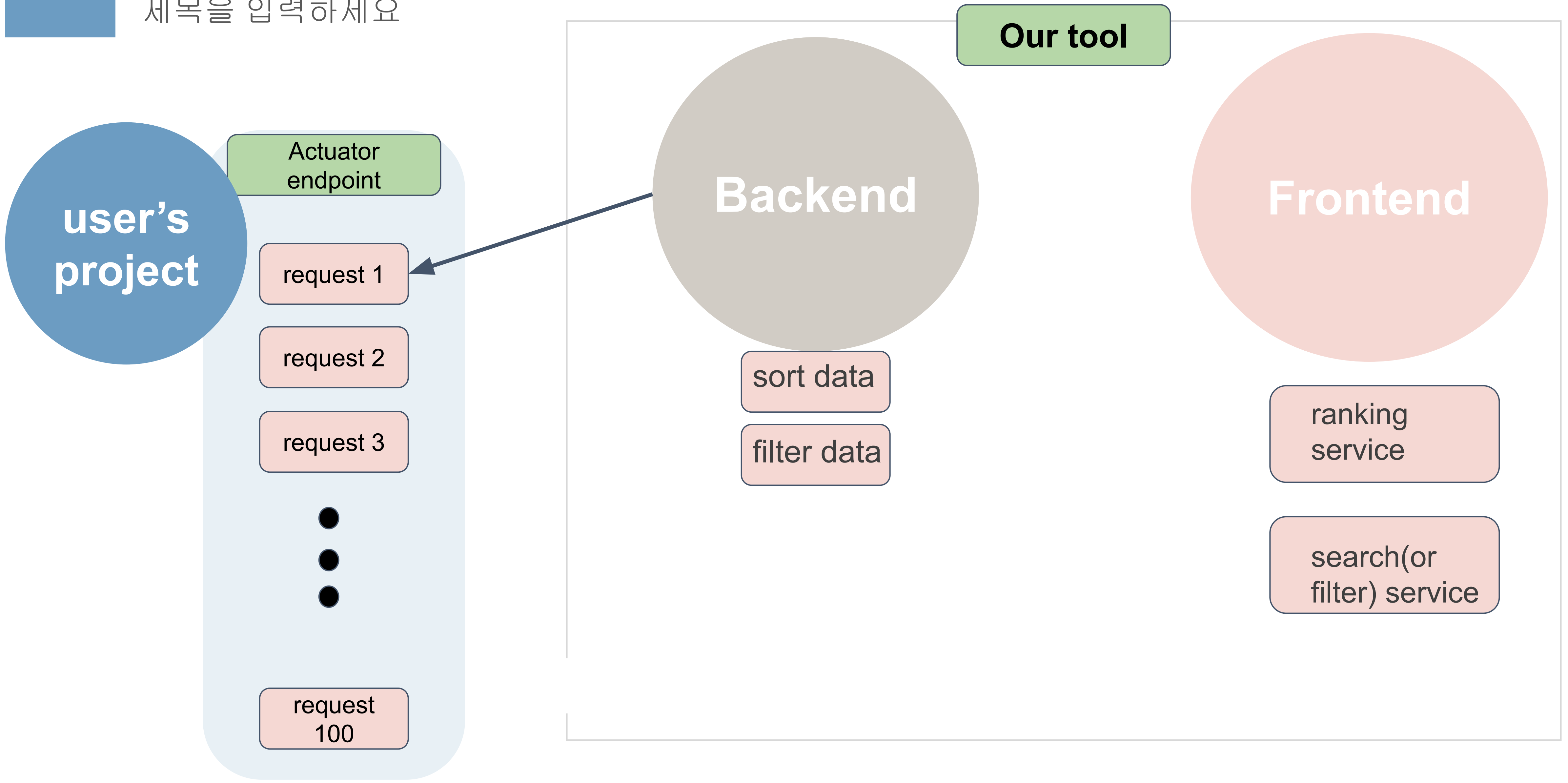
Part 3

제목을 입력하세요

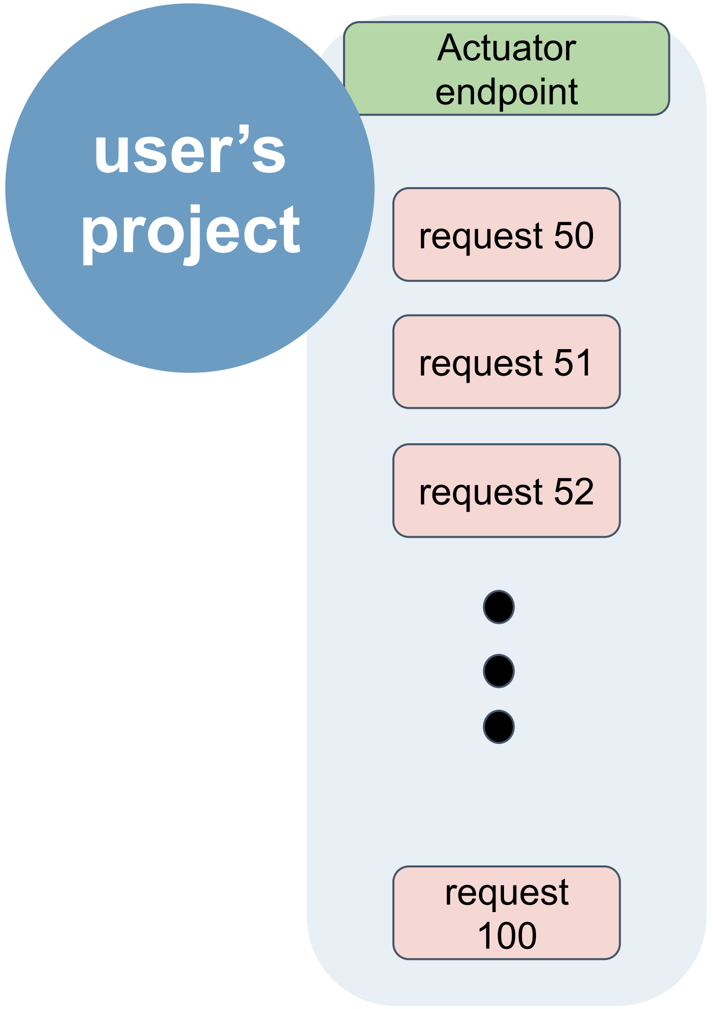


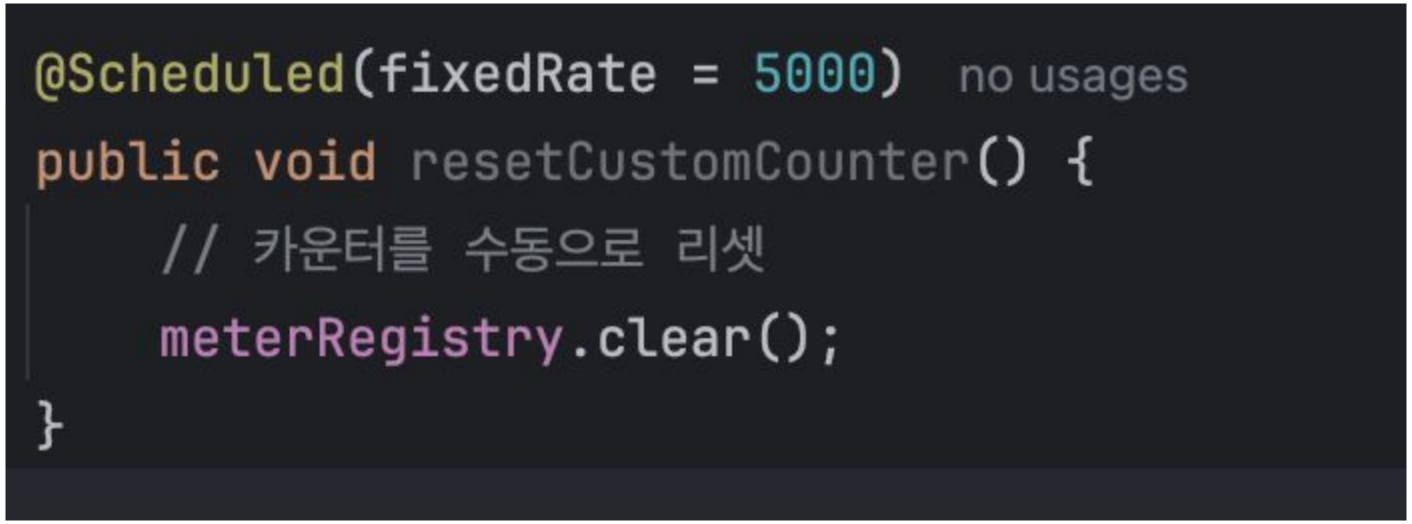
Challenging Situation : user wants to use this tools for his/her project for long time

제목을 입력하세요


Our tool
user's project
Actuator endpoint
request 1
request 2
request 3
●
●
●
request 100
Backend
sort data
filter data
Frontend
ranking service
search(or filter) service

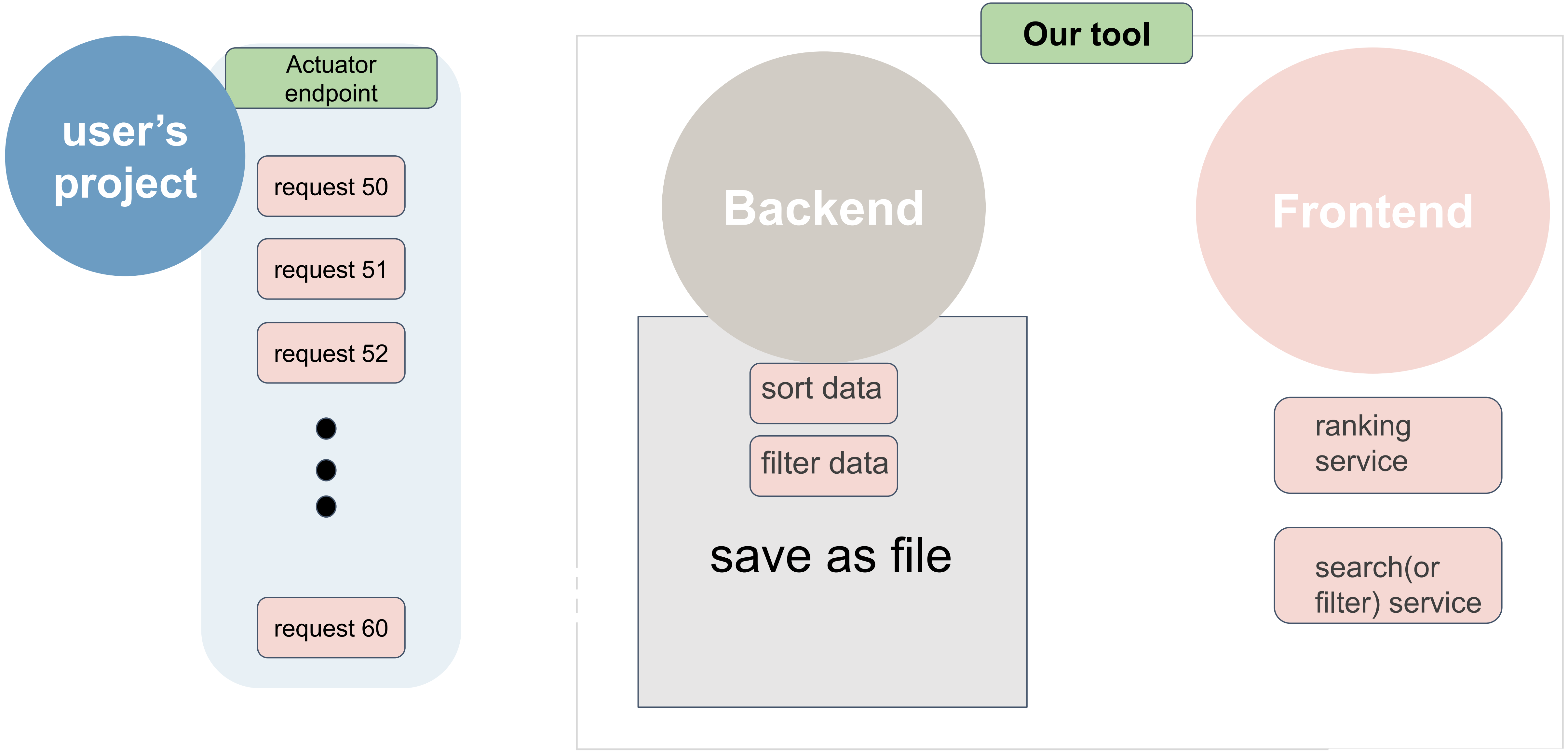
On user's projet, reset data per 5 seconds (or user can change duration)



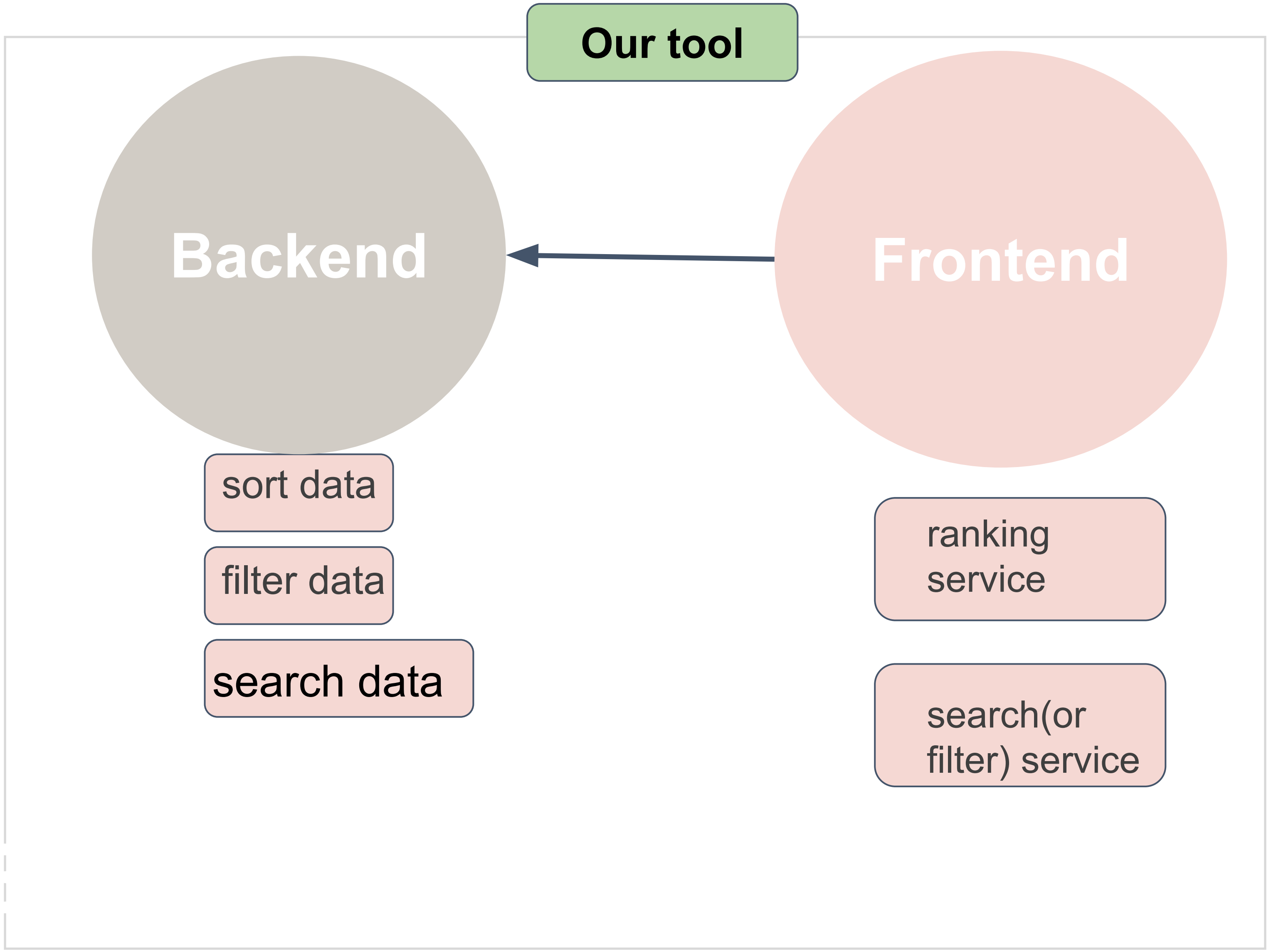


```
@Scheduled(fixedRate = 5000)  no usages
public void resetCustomCounter() {
    // 카운터를 수동으로 리셋
    meterRegistry.clear();
}
```

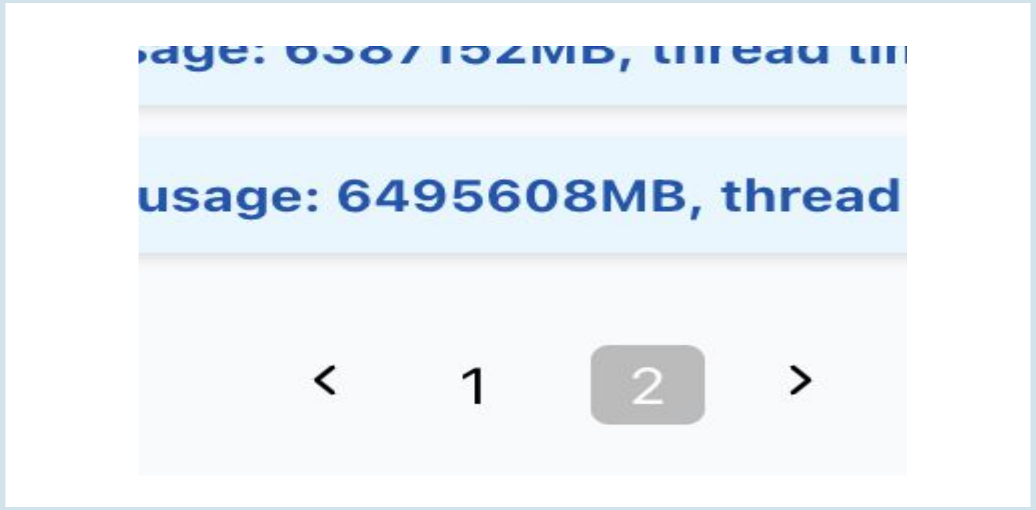
What about refreshed data?(in this example, request 1 ~ 49)



Challenging Situation : if Frontend needs too much data?


Our tool
Backend
Frontend
sort data
filter data
search data
ranking service
search(or filter) service

Solved by pagination - used react-js-pagination library

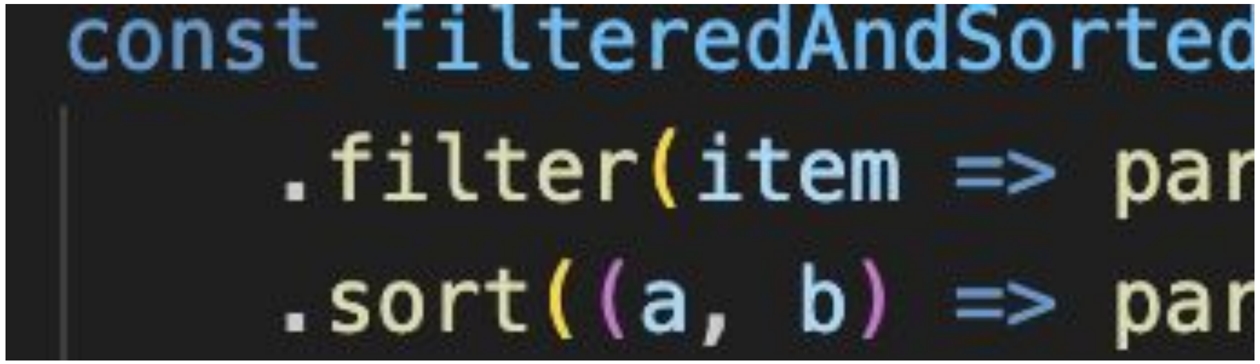

usage: 6495608MB, thread

After

Challenging Situation : Sorting for ranking service Backend

Solved by pagination

Ranking Api → Need SORT!!!



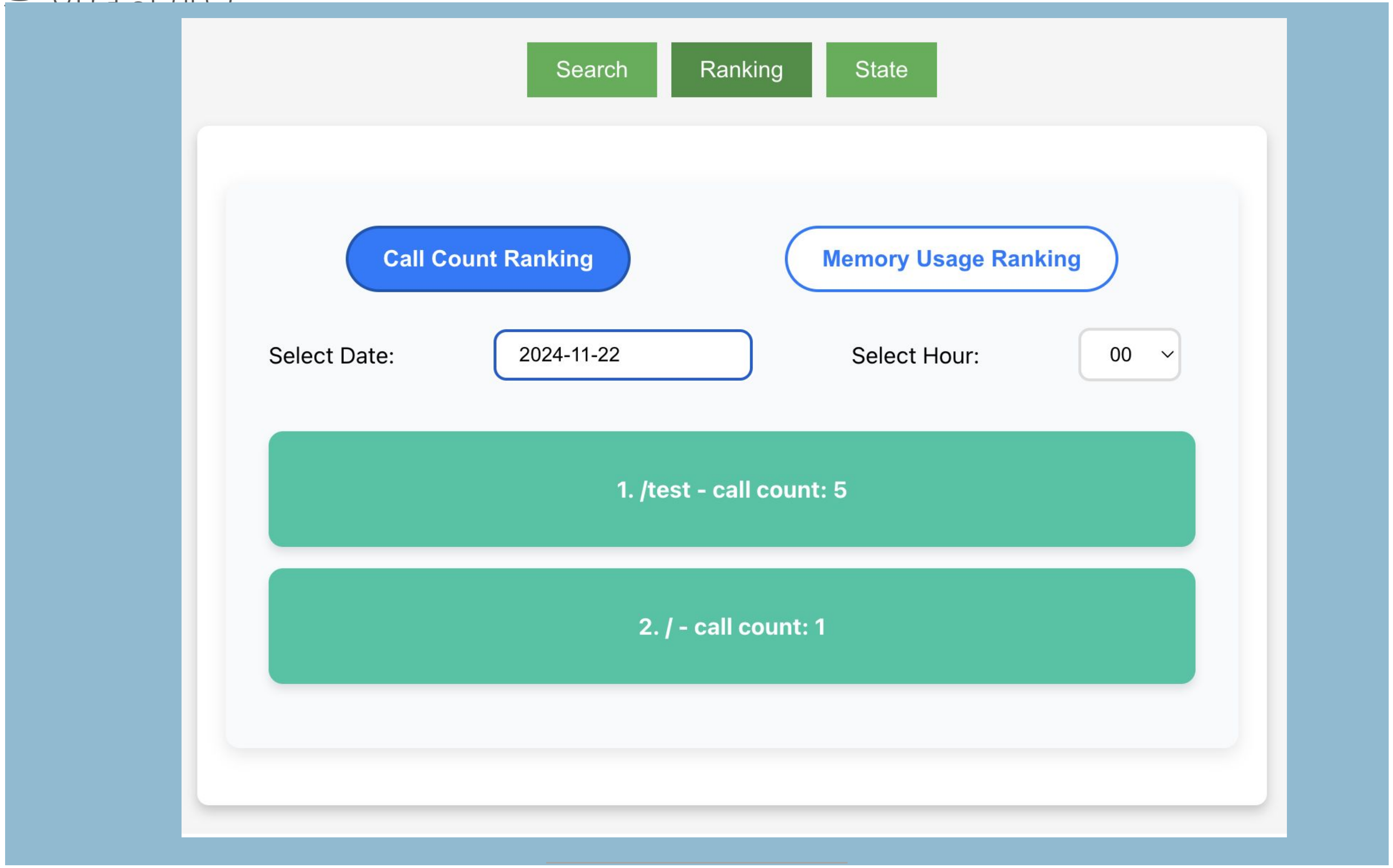
```
const filteredAndSorted
    .filter(item => par
    .sort((a, b) => par
```

Timsort: O(nlogn)
-> merge sort + insertion sort

Challenging Situation : Handling Date and Time Selection Issues in User Inputs - Frontend

Part 2

제목



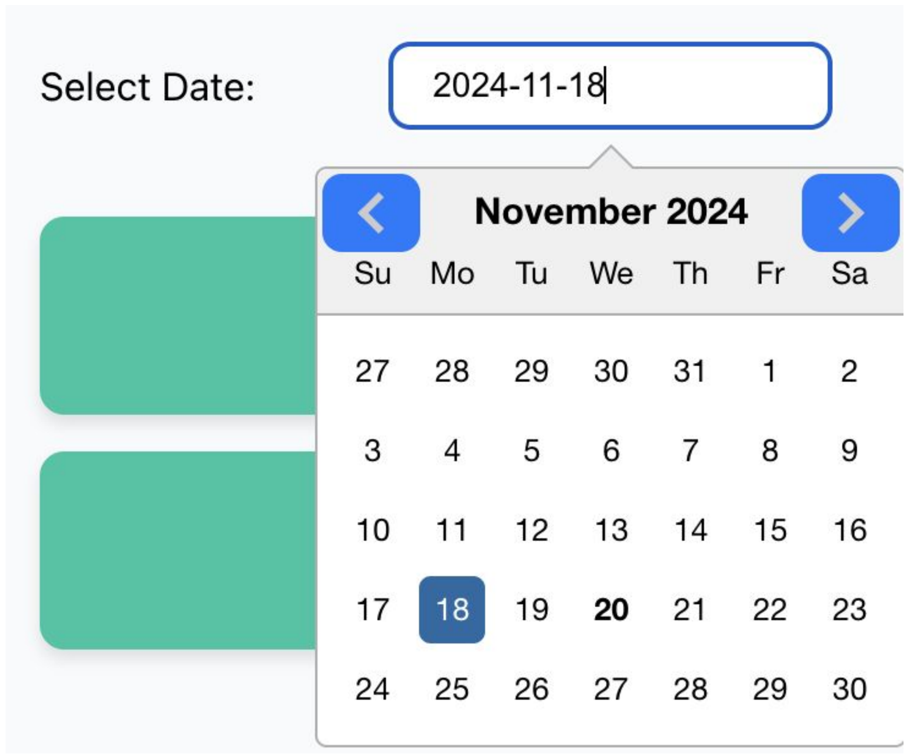
Select Date: 2024-11-22

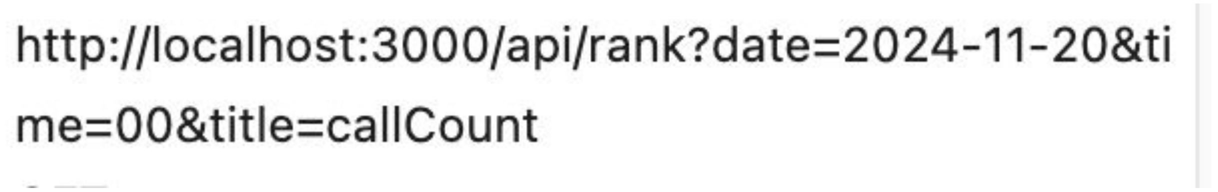
Select Hour: 00

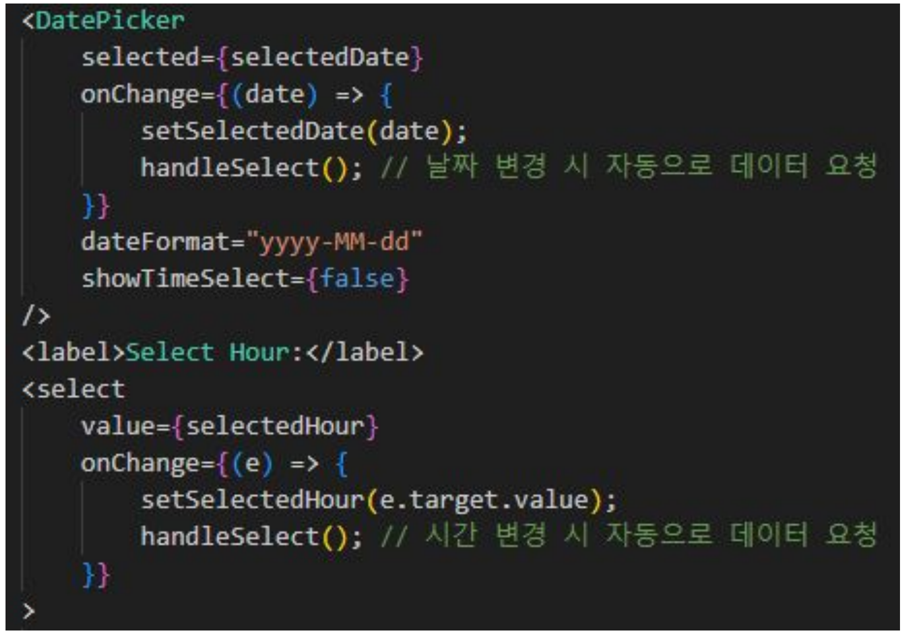
1. /test - call count: 5

2. / - call count: 1

Summary Page

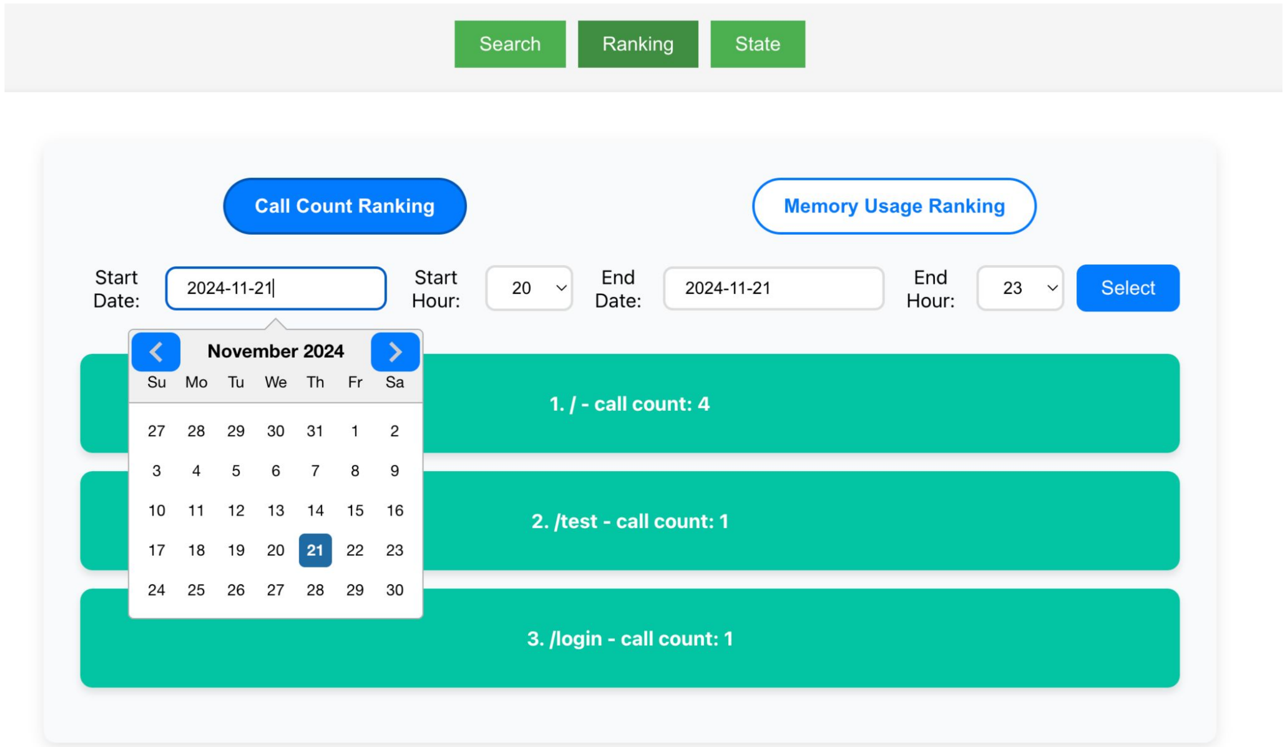

Select Date: 2024-11-18


http://localhost:3000/api/rank?date=2024-11-20&time=00&title=callCount


```
<DatePicker
    selected={selectedDate}
    onChange={(date) => {
        setSelectedDate(date);
        handleSelect(); // 날짜 변경 시 자동으로 데이터 요청
    }}
    dateFormat="yyyy-MM-dd"
    showTimeSelect={false}
/>
<label>Select Hour:</label>
<select
    value={selectedHour}
    onChange={(e) => {
        setSelectedHour(e.target.value);
        handleSelect(); // 시간 변경 시 자동으로 데이터 요청
    }}
>
```

nt

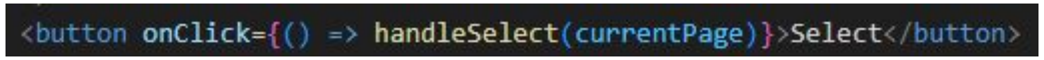
Part 3



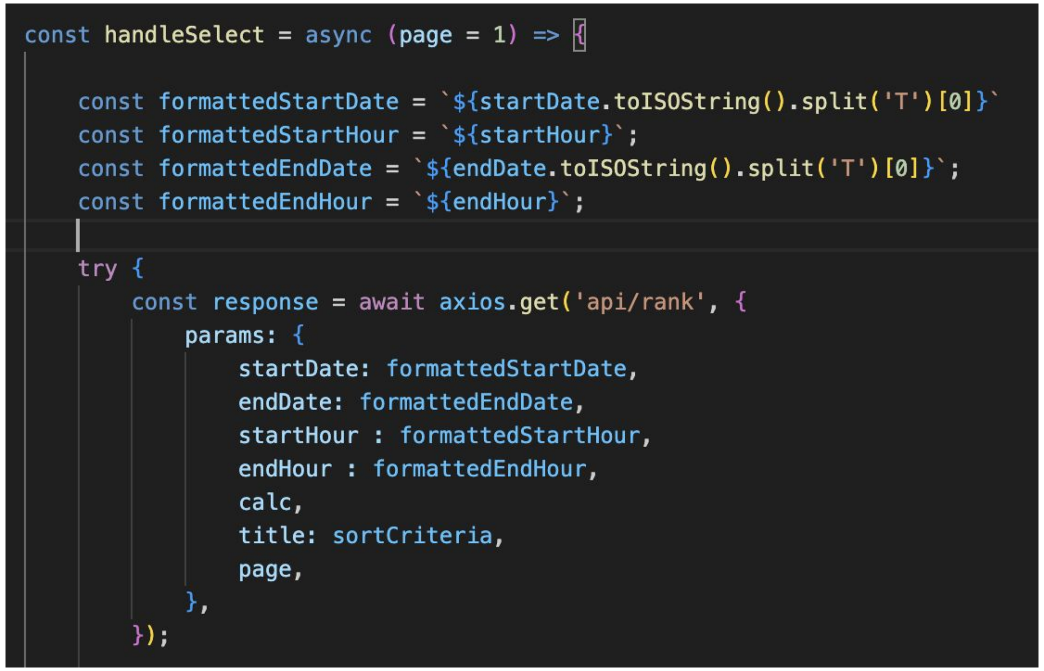
1. / - call count: 4

2. /test - call count: 1

3. /login - call count: 1



```
<button onClick={() => handleSelect(currentPage)}>Select</button>
```



```
const handleSelect = async (page = 1) => {

    const formattedStartDate = `${startDate.toISOString().split('T')[0]}`
    const formattedStartHour = `${startHour}`;
    const formattedEndDate = `${endDate.toISOString().split('T')[0]}`;
    const formattedEndHour = `${endHour}`;

    try {
        const response = await axios.get('api/rank', {
            params: {
                startDate: formattedStartDate,
                endDate: formattedEndDate,
                startHour : formattedStartHour,
                endHour : formattedEndHour,
                calc,
                title: sortCriteria,
                page,
            },
        });
```

THANK YOU