

# Project 2

SNUSearch 서비스 설계 및 개발

# Background

때는 2001년. 당신은 천재 정현규 박사와 함께 SNUSearch를 설립하였다.

정 박사가 개발한 검색 엔진을 사용해서 다양한 검색 옵션을 제공하는 검색 서비스를 출시해야 한다. 또한 해킹 공격에 대비하기 위한 보안 장치를 마련하고, 사용자 정보를 최대한 수집하여 훗날 AI 시대를 대비하고자 한다.

장기간 회의를 통해 확정한 요구사항과 정 박사가 제공한 Search API를 활용하여, “요즘 가장 주목받고 있는 Java” 기반 Backend 서비스를 개발해보자.

# Related Terminology

## RESTful API (Representational State Transfer API)

- Client/Server 사이의 통신 지침을 제공하는 개념 (참고문서: <https://namu.wiki/w/REST>)
- RESTful API를 참고하여 잘 구성된 HTTP Request를 보내고 그에 따른 HTTP Response 받기 가능
- [ChatGPT](#), [Bard](#)와 같은 생성형 AI를 비롯하여, 대부분의 주요 웹 서비스에서 RESTful API를 제공함
- [CURL](#) 커맨드를 통해 모든 OS의 터미널 환경에서 Restful API를 간단하게 테스트 해볼 수 있음

### ChatGPT API

```
curl -X POST https://api.openai.com/v1/chat/completions -H "Authorization: Bearer <YOUR_API_KEY>"  
-H "Content-Type: application/json" -d '{"model": "gpt-3.5-turbo", "messages": [{"role": "user", "content": "서울대학교 컴퓨터공학부 2학년 수업 제목 5개를 알려주세요."}]}'
```

### Bard API

```
curl -X POST 'https://api.bardapi.dev/chat' -H 'Authorization: Bearer <YOUR_API_KEY>' -H 'Content-Type: text/plain' -d '{"input": "서울대학교 컴퓨터공학부 2학년 수업 제목 5개를 알려주세요."}'
```



교통

활용사례(갤러리) 등록 URL 복사 목록 이동

### 서울시 지하철 실시간 도착정보

서울시 교통정보과(TOPIS)에서 제공하는 정보를 이용한 지하철 실시간 도착정보 조회 API 입니다.  
\* 역조회시 참조: 용암 -> 용암순환(상선), 공릉 -> 공릉(서울산업대입구), 준의 -> 준의, 남한산성입구 -> 남한산성입구(성남법원, 검찰청), 대모산입구 -> 대모산, 천호 -> 천호(풍납토성), 몽촌토성 -> 몽촌토성(평화역문)  
\* 서울시 이외의 역구간은 미제공 됩니다.(예, 광명, 서동탄, 춘천 등)

<http://data.seoul.go.kr/dataList/OA-12764/A/1/datasetView.do>

### > 기본 정보

서비스명	기상청 단기예보 조회서비스(2.0)	첨부문서	기상청41_단기예보 조회서비스_오픈API활용가이드_최종.zip 
데이터포맷	JSON+XML	API유형	REST
등록일	2021-07-01	수정일	2023-04-11
서비스설명	초단기실황, 초단기예보, 단기예보, 예보마전 정보를 조회하는 서비스입니다. 초단기실황정보는 동네예보 구역에 대한 대표 AWS 관측값을, 초단기예보는 예보시점부터 6시간까지의 예보를, 동네예보는 예보기간을 끝까지 확장 및 예보단위를 상세화(3시간→1시간)하여 시간간격으로 세분화한 예보를 제공합니다.		

<https://data.kma.go.kr/api/selectApiList.do?pgmNo=42>

# Related Terminology

## Regular Expression (정규 표현식)

- 프로그래밍에서 문자열을 다룰 때, 문자열의 일정한 패턴을 표현하는 일종의 형식 언어
- 이메일 포맷 등 특정 포맷 준수 여부 검사 등 다양한 용도로 활용 가능
- 자바에서는 String 객체 내 여러 메소드에서 RE 문법을 지원 (matches, replaceAll, split)

영문자

`^[a-zA-Z]*$`

이메일 주소

`WWW+@WWW+WW.WWW+(WW.WWW+)?`

```
1 // matches (일치하는지 확인)
2 String txt = "123456";
3 boolean result1 = txt.matches("[0-9]+"); // 숫자로 이루어져 있는지
4 System.out.println(result1); // true
```

## JSON (JavaScript Object Notation)

- 서버에서 클라이언트로 데이터를 보낼 때 사용하는 포맷 (참고문서: <https://namu.wiki/w/JSON>)
- JSON 파일을 간편하게 Java의 ArrayList로 변환하여 활용 가능

# Related Terminology

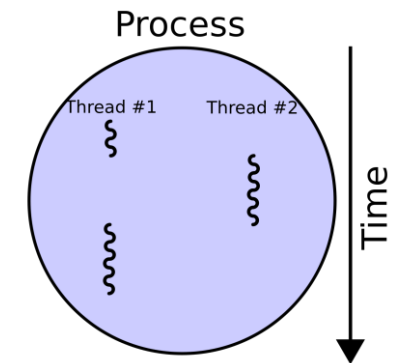
## Socket (소켓)

- 전기/전자 부품에 연결하는 접점 부분의 작은 부품 (전구는 소켓을 통해 전력과 연결)
- 네트워크 통신 관점에서는 통신 양쪽 Endpoint를 의미 (PC는 소켓을 통해 네트워크와 연결)
- Client와 Server간의 통신 과정에서 양쪽에 위치한 Socket이 해당 역할을 중재함 (port open 등)



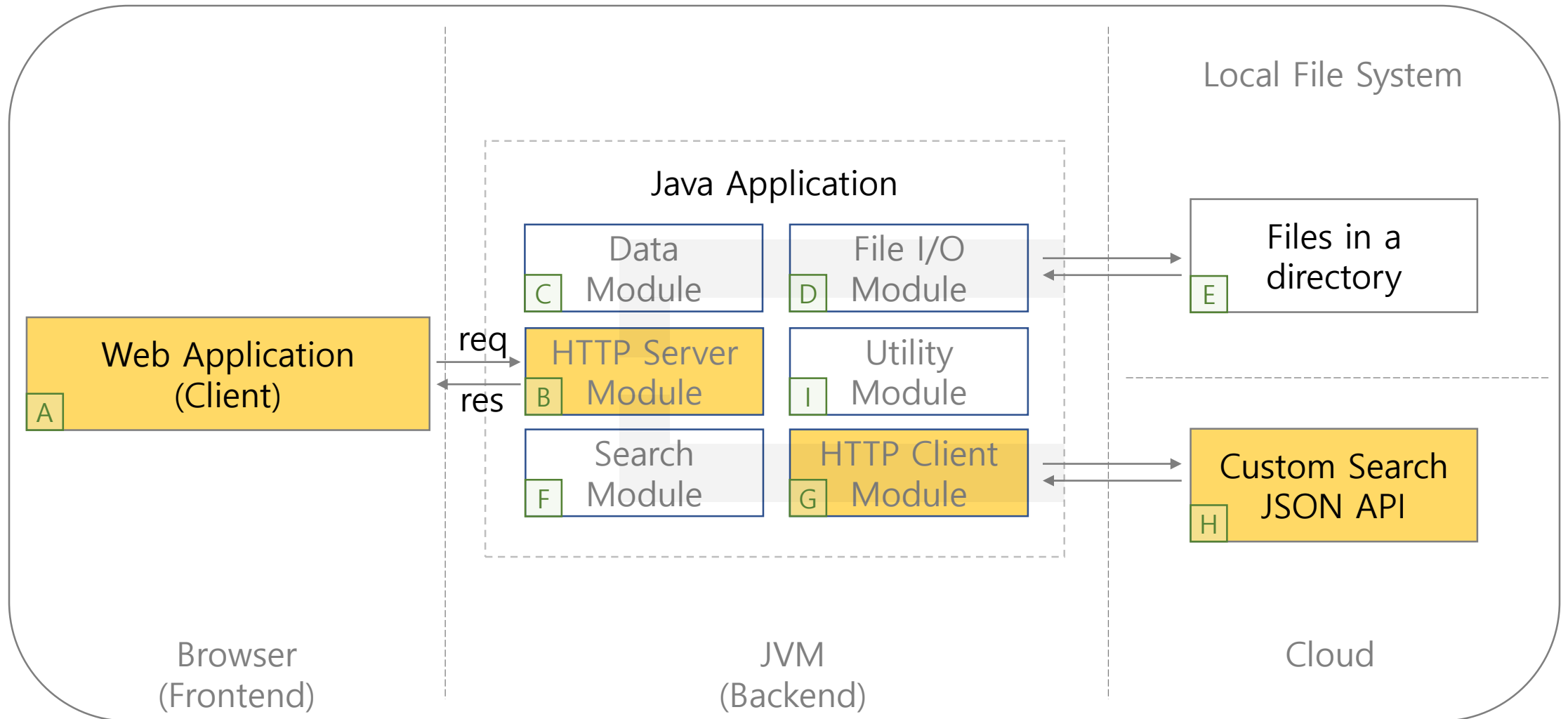
## Thread (스레드)

- 프로세스 내에서 한 개 이상의 존재할 수 있으며 실제 작업을 수행하는 주체
- 두 개 이상의 스레드를 가지는 프로세스를 Multi-threaded process라고 함
- Java에서 Thread를 생성하려면, Runnable Interface를 구현하거나 Thread Class를 상속받아야 함



# Overall Design

\* 코드 또는 가이드라인 제공



# Requirement

## 사용자 기능 리스트 (5개)

기능	동작
회원가입 기능 GET /user/join	아이디와 비밀번호를 입력하고 드롭다운 리스트에서 "Join" 선택 후 버튼 클릭 회원가입 성공 조건 • 아이디가 기존에 등록된 것과 중복되지 않아야 하며, 대소문자를 구분하지 않음 • 비밀번호는 4자 이상이어야 하며 알파벳(영문자)로 시작해야만 하고 특수기호는 @, %만 허용 (# 허용 X)
로그인 기능 GET /user/login	아이디와 비밀번호를 입력하고 드롭다운 리스트에서 "Login" 선택 후 버튼 클릭 로그인 성공 조건: 아이디와 비밀번호가 기존 회원 가입했던 정보와 일치해야 함 아이디와 비밀번호 모두 admin을 입력하고 로그인하면 관리자 모드 진입 (상세 내용은 <a href="#">9 page</a> 참조)
로그아웃 기능 GET /user/logout	아이디와 비밀번호를 입력하고 드롭다운 리스트에서 "Logout" 선택 후 버튼 클릭 로그아웃 성공 조건: 로그인 성공 상태에서 로그인 한 아이디를 정확하게 입력함 (단, 웹 페이지를 새로고침하면 자동 로그아웃이 됨, <b>패스워드는 비워 두거나 아무 값이 와도 OK, 같게 해도 OK</b> )
회원탈퇴 기능 GET /user/leave	아이디와 비밀번호를 입력하고 드롭다운 리스트에서 "Leave" 선택 후 버튼 클릭 회원탈퇴 성공 조건: 아이디와 비밀번호가 기존 회원 가입했던 정보와 일치해야 함
회원복구 기능 GET /user/recover	아이디와 비밀번호를 입력하고 드롭다운 리스트에서 "Recover" 선택 후 버튼 클릭 회원복구 성공 조건: 아이디와 비밀번호가 기존 회원 탈퇴했던 정보와 일치해야 함

# Requirement

## 검색 기능 리스트 (5개)

기능	동작
검색 기능 GET /data/search	로그인 후, 검색어 (+검색 옵션)을 입력하고 "Search" 버튼 클릭 (상세 내용은 <a href="#">16 page</a> 참조) 검색 성공 조건 • Custom Search JSON API의 응답이 정상(200) 검색 결과여야 함
MY 검색어 저장 기능 GET /data/save_data	로그인 후, 검색어를 입력하고 드롭다운 리스트에서 "Save_Data" 선택 후 버튼 클릭 해당 검색어 (+검색 옵션)은 서버로 전달되어 해당 사용자 계정에 한해 저장됨
MY 검색어 로드 기능 GET /data/load_data	로그인 후, 드롭다운 리스트에서 "Load_Data" 선택 후 버튼 클릭 그동안 저장한 모든 MY 검색어를 로드하여 출력창에 보여줌 (정렬 불필요 ,이후 추가 Action 요구되지 않음)
친구 검색어 로드 기능 GET /data/load_fri	로그인 후, 친구 ID를 입력한 후 드롭다운 리스트에서 "Load_Fri" 선택 후 버튼 클릭 해당 친구 계정의 MY 검색어를 로드하여 출력창에 보여줌 (정렬 불필요, 이후 추가 Action은 요구되지 않음)
HOT 검색어 로드 기능 GET /data/load_hot	로그인 후, 드롭다운 리스트에서 "Load_Hot" 선택 후 버튼 클릭 계정과 무관하게 저장된 모든 검색어를 빈도수가 높은 순으로 TOP 10개 출력 (10개 미만이면 그것만 출력)



# Requirement

## 관리자 기능 리스트 (2개)

기능	동작
회원정보 로드 기능 GET /data/load_acc	관리자로 로그인 후, 드롭다운 리스트에서 "Load_Acc" 선택 후 버튼 클릭 이 때, 회원정보는 모든 사용자의 ID 정보를 의미 출력 예시: "test \n user11 \n admin \n test123" (웹 앱에서는 한 줄에 ID 1개씩 출력)
시스템 로그정보 로드 기능 GET /data/load_log	관리자로 로그인 후, 드롭다운 리스트에서 "Load_Log" 선택 후 버튼 클릭 이 때, 시스템 로그정보는 모든 사용자의 과거 request 정보 히스토리를 의미 (response 정보는 고려하지 않음) 출력 예시: "[사용자ID] request URL" ([test] http://localhost:8080/user/login?id=test&passwd=abcd)

\* 관리자 기능은 admin 계정으로 로그인해야 사용 가능

\* admin 계정도 초기에 회원가입을 통해 계정을 생성해야 이용 가능  
(비밀번호는 admin으로 해도 되고 임의의 값으로 해도 OK)

# Requirement

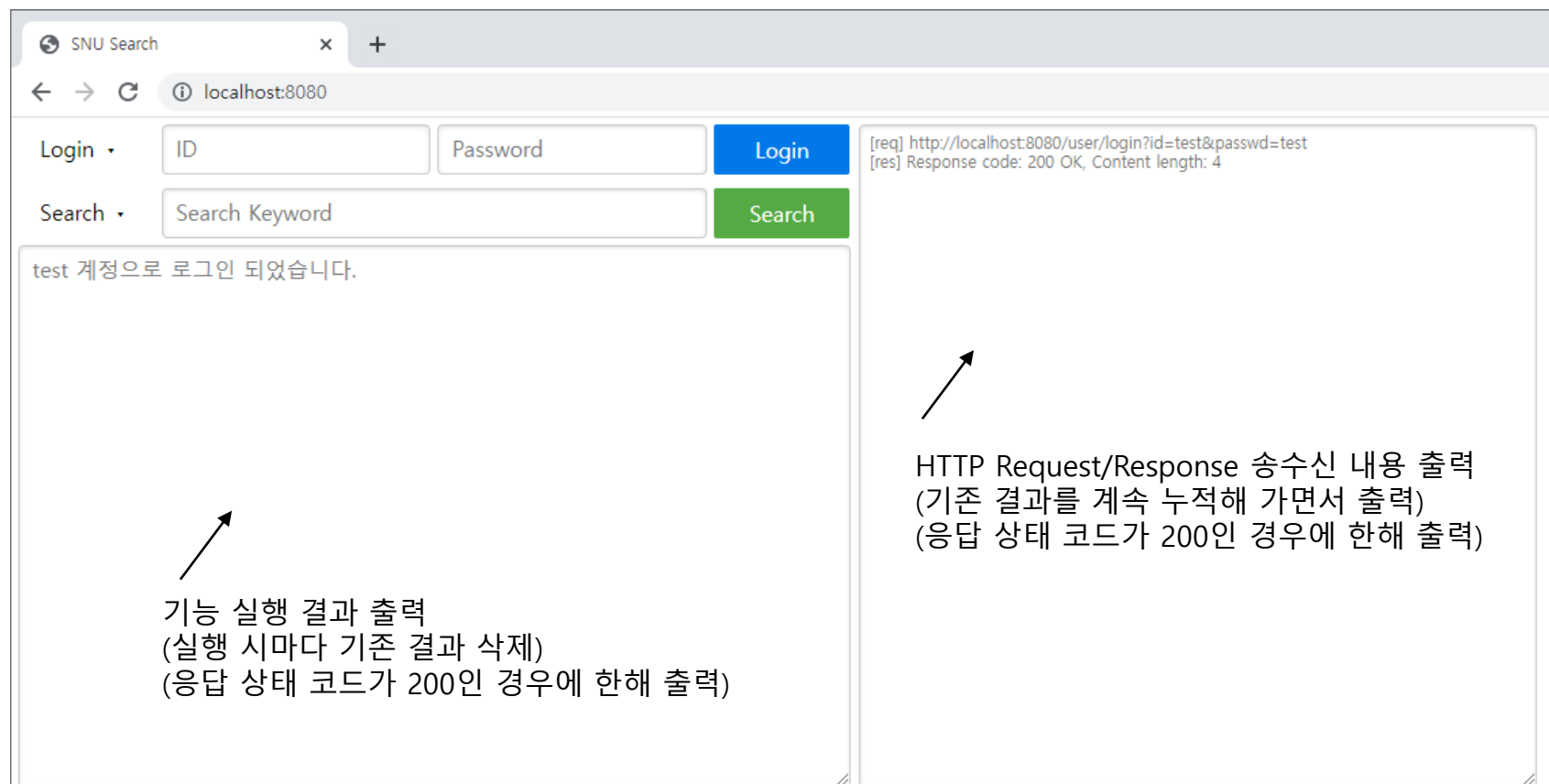
## A: Web Application (제공 코드 그대로 사용)

- 브라우저에서 실행 가능한 index.html 파일 제공 (과제 진행 과정에서 별도 수정 불필요)
- 해당 html 파일을 IntelliJ 프로젝트 내 src에 넣어두면,  
HTTP Server Module(B) 실행 후 <http://localhost:8080>으로 접근 가능
- 사용자가 해당 웹 페이지에서 값을 입력하고 버튼을 클릭하면  
해당 정보가 B (HTTP Server Module)에게 GET 방식으로 전달되어 요청을 수행하고  
처리 결과 응답을 다시 A에게 전달하면 해당 결과를 출력하고 우측에 해당 Response 내용 출력
- 응답 상태 코드 (Response Status Code)가 200인 경우에 한해 출력 진행  
(서버에서 오류 발생 시, 응답 상태 코드를 200으로 보내면 안됨, [MDN 참고링크](#))
- 동시에 여러 사용자가 여러 계정에 로그인 등의 복잡한 상황은 일절 고려하지 않음  
(단순하게 한 번에 한 명만 사용 가정)
- 입력 받은 비밀번호의 보안은 본 프로젝트에서는 Client, Server 모두에서 고려하지 않음

# Requirement

새로고침 (F5) 실행 시 모두 초기화  
(다시 로그인 해야함)

## A: Web Application (제공 코드 그대로 사용)



기능 실행 결과 출력  
(실행 시마다 기존 결과 삭제)  
(응답 상태 코드가 200인 경우에 한해 출력)

HTTP Request/Response 송수신 내용 출력  
(기존 결과를 계속 누적해 가면서 출력)  
(응답 상태 코드가 200인 경우에 한해 출력)

LOGIN ID: test

현재 로그인 사용자 ID 출력  
DCSLAB CSE, SNU

# Requirement

## B: HTTP Server Module (제공 코드 활용하여 확장 또는 재작성)

- index.html 파일을 호스팅하는 웹 서버 역할을 담당하는 Java 샘플 파일 제공 (SimpleHttpServer.java)
- index.html에서 요청하는 GET 방식 HTTP Request에 대한 Response를 보내는 역할 담당
- 해당 HTTP Request와 HTTP Response를 보내고 받는 역할은 run() 메소드에서 담당
- 모든 HTTP Request 메시지는 GET 방식으로 전달되며, 다음 Scheme을 가짐  
(Full URL은 앞 슬라이드 기능 리스트 및 index.html 제공 코드 참조)

기능	동작
사용자 기능	http://localhost:8080/user/...
검색/관리자 기능	http://localhost:8080/data/...

\* 예시: 아이디가 test이고 비밀번호가 abcd인 계정으로 로그인하는 HTTP Request URL  
=> <http://localhost:8080/user/login?id=test&passwd=abcd>

# Requirement

## B: HTTP Server Module (제공 코드 활용하여 확장 또는 재작성)

- 로그인 처리 예시

- Web App에서 아이디/비밀번호 정보를 아래와 같이 서버로 전달
- <http://localhost:8080/user/login?id=test&password=test>
- 그러면 run() 메소드 내에서 해당 정보 식별 및 처리 필요 (text, json 포맷 모두 OK)
- 아래 스크린샷 코드의 content 값은 별도로 작성한 Class를 통해 생성해서 클라이언트에 전달 필요
- 로그인 성공 시 (기존에 없는 ID이며 규칙에 문제가 없음)  
응답코드를 200으로 하고 content 변수에 성공 결과 담기  
(성공 문구 예시: test 계정으로 로그인 되었습니다.)
- 검색의 경우 검색 결과를 content 변수에 담기  
(검색 결과는 [page 16](#)에 명시된 정보만 추려서 보여주기)
- 처리 과정에서 문제(오류) 발생 시  
반드시 응답코드를 200이 아닌 값으로 해야함

```
} else { // CASE 2: It sends a specific data to the client as an HTTP Response.  
    // #####  
    // This #AREA# needs to be revised using an external class.  
    String content = "PASS";  
    // #####  
    out.println("HTTP/1.1 200 OK");  
    out.println("Content-Type: text/plain"); // application/json would be OK!  
    out.println("Content-Length: " + content.length());  
    out.println();  
    out.println(content);  
}
```

# Requirement

## C: Data Module

- 파일 접근이 필요한 모든 요청은 Data Module에서 담당
- 크게 세 가지 유형으로 분류 가능: 1) 사용자 정보, 2) 검색 정보, 3) 관리자 정보
- 요구사항에 대한 명확한 이해를 통해 클래스 및 파일을 적절하게 구성해야 함
- 각 기능에 대한 상세 내용은 앞 슬라이드의 요구사항 내용 확인 (명시되지 않은 내용은 자율 구현)

## D: File I/O Module

- 파일을 생성하고 접근하는 부분을 전담하는 모듈 (C와 통합하여 개발하는 방식도 OK)
- 특정 파일이 없는 경우 바로 신규 생성해야 하며, 있는 경우 그 안에 값을 업데이트 하는 방식으로 동작

## E: Files in a directory

- 모든 파일은 프로젝트 내 data 디렉토리에 두고 활용 (src 디렉토리와 같은 레벨)

# Requirement

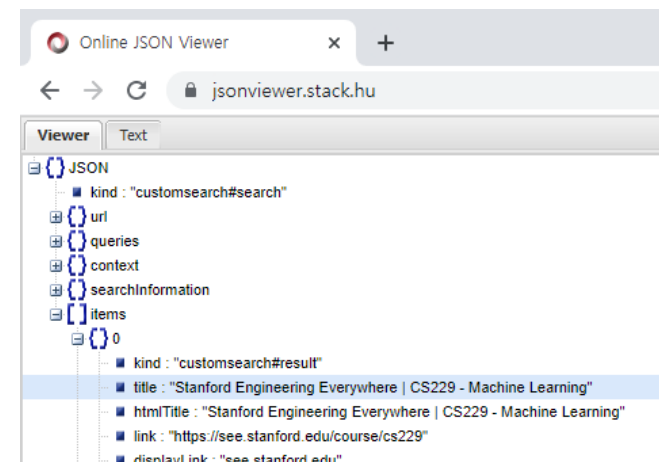
## F: Search Module

- Google Custom Search JSON API를 활용하여 검색어 정보 요청 및 가공
  - 해당 API 활용을 위해서는 구글 계정 로그인 후 API 키 발급 필요 (아래 링크 안내 참조)
  - [https://developers.google.com/custom-search/v1/using\\_rest](https://developers.google.com/custom-search/v1/using_rest)
  - 코드에 해당 **API\_KEY**를 포함시켜서 제출 필요 (채점 시 최소한의 횟수로 활용 및 폐기 예정)
- 이 때, 검색어는 검색어 키워드와 검색 옵션으로 구성 (검색어만 있어도 OK)
  - 검색어에 **한글**과 특수문자는 포함하지 않는다고 가정 (영문자, 숫자만 허용)
  - 검색 옵션은 검색어 뒤에 한 칸 띄고 "?" 표기 후 &을 구분자로 하여 옵션 여러 개 나열 가능
  - 검색 옵션 리스트: <https://developers.google.com/custom-search/v1/reference/rest/v1/cse/list>
- 검색어+검색옵션 예시
  - Seoul National University ?fileType=pdf&lr=lang\_ko

# Requirement

## G: HTTP Client Module (제공 코드 활용하여 확장 또는 재작성)

- Google Custom Search JSON API에 요청을 보내고 받는 Java 샘플 파일 제공 (SimpleHttpClient.java)
- 웹 앱을 통해 전달 받은 검색어를 Google API 규격에 맞게 가공하여 HTTP 요청 발송 필요
- 보통 JSON 포맷의 응답 내용의 길이가 매우 긴 편이므로 아래 정보를 참고하여 해석 필요
  - JSON 텍스트를 ArrayList로 변환하면 원하는 정보에 보다 쉽게 접근 가능 (Gson, JSON-B 등 외부 라이브러리 활용)
  - 내용 확인 목적으로 JSON Viewer 활용 가능 (JSON Viewer 예시: <https://jsonviewer.stack.hu/>)
- 검색 결과는 보통 10개가 전달되는데, 그 중 TOP3 (index 0~2)만 추려서 index.html에 응답으로 전송
- 이 때, HTTP 응답 메시지에 꼭 포함시켜야 하는 데이터는 아래와 같음
  - queries.request[0].searchTerms
  - queries.request[0].totalResults
  - items[0].title, items[0].link (item[1], item[2]의 동일 정보 포함)
  - 그 외의 추가 데이터를 포함시키는 것도 허용 (선택사항, 채점과 무관)







# Requirement

## H: Google Custom Search JSON API

- 구글 계정 로그인 후 아래 링크에서 Key 발급, 검색 엔진 ID 생성 진행 필요
- 계정당 API 사용 한도가 하루 100회로 제한되어 있음 (잡은 테스트 필요 시, API 호출 부분 잠시 주석 처리 등 고려 가능)
- [https://developers.google.com/custom-search/v1/using\\_rest](https://developers.google.com/custom-search/v1/using_rest)

[홈](#) > [제품](#) > [Programmable Search Engine](#) > [가이드](#)

도움이 되었나요?  

## REST를 사용하여 API 호출

의견 보내기

이 문서에서는 Custom Search JSON API를 사용하는 방법을 설명합니다.

### 요청하기

REST([Representational State Transfer](#))는 Custom Search JSON API에서 기존 REST와 약간 다릅니다. API는 리소스에 대한 액세스 권한을 제공하는 대신 서비스에 대한 액세스를 제공합니다. 따라서 API는 서비스 엔드포인트 역할을 하는 단일 URI를 제공합니다.

URI에 HTTP `GET` 요청을 전송하여 특정 검색의 결과를 검색할 수 있습니다. 검색 요청의 세부정보를 쿼리 매개변수로 전달합니다. Custom Search JSON API URI의 형식은 다음과 같습니다.

# Requirement

## I: Utility Module

- 여러 클래스에서 공통적으로 활용하는 메소드 또는 일반적인 쓰임새가 있는 메소드를 제공
- Math 클래스와 유사하게 아래 두 가지 요건 만족해야 함
  - final class를 정의하여 상속받지 못하게 함
  - 모든 메소드 및 필드는 static을 명시하여 객체 생성 없이 활용할 수 있게 함
- 활용 예시
  - `SNUUtility.validateEmailFormat(email_string);` // 클래스명, 메소드명, 변수명은 모두 자율적으로 정하기
- Math 클래스 내용 확인 방법
  - Math.PI 코드 작성 후 마우스 우클릭, Go To -> Implementation 메뉴를 통해 파일 열기
  - Shift 키를 두 번 연달아 누르면 나오는 화면에서 Math.java 파일을 검색하여 열기

# Evaluation (총 20점)

## 요구사항 만족도 (15점)

- 사용자/검색/관리자 기능 총 12개가 버그없이 안정성 있게 동작해야 함 (안정성 의미 예시: 응답코드 200 여부 판정)
- 요구사항에 명시되지 않은 사항은 자율적으로 판단하여 진행 OK (자유도 높은 편)

## 패키지/클래스/파일 설계 구조 (3점)

- 모든 클래스는 상단에 패키지가 명시되어야 하며 패키지명은 해당 파일의 디렉토리 path를 참고하여 정의 (Java 기반 오픈 소스 참조)
- 프로젝트 내 패키지 개수 최소 2개 이상, 클래스 사이의 상속 관계 (**interface 포함**) 최소 3개 이상, 데이터 파일 개수 최소 4개 이상 준수  
\* In other words, 파일 최상단에 명시하는 패키지명 2개 이상, 서브클래스 3개 이상, File I/O가 발생하는 파일 개수 4개 이상

## 코드 주석 및 README.md 문서 (2점, 국문/영문 모두 허용)

- 중요한 코드 또는 복잡한 코드에 이해를 돕는 주석 표기 (전체 코드의 5% 이상은 주석이어야 함. 20줄 코드에 주석 1줄 이상)
- 프로젝트 최상단 디렉토리 내 README.md 텍스트 파일을 생성하고 10-20 lines 분량으로 아래 내용 간략히 기재
  1. 요구사항 14개 중 미구현 항목 명시 및 이유,
  2. 가장 어려웠던 기능 및 접근 방법,
  3. 느낀점 (유익한점, 애로사항, 개선점 등)

# Evaluation (optional)

## Early submission benefit (2점)

- 6/6(화) 23:59까지 프로젝트 결과물을 제출한 사람 중, 우수 결과물 제출자 1명 또는 2명에게 추가 점수 2점 부여
- 기본 점수가 만점인 경우 해당사항 없음 (기본 배점에 감점이 있는 경우 추가 점수 2점이 적용됨)
- 6/7(수) 또는 그 이후 수정 제출이 있는 경우 대상자 후보에서 제외
- 우수 결과물은 사전 개별통보 및 협의 후 6/14(수) 실습 수업시간에 공유 예정 (또는 eTL을 통해 코드 공유)

# Policy

## 제출 기한

- 2023년 6월 13일(화) 23시 59분  
(기한을 넘겨 제출 시 24시간당 -2점 감점, 48시간 이후 받지 않음)

## 제출 방식

- 프로젝트 명 (프로젝트 디렉토리 이름)은 SNUSearch\_[이름]로 정의  
(예시: SNUSearch\_홍길동)
- 프로젝트 디렉토리를 통째로 압축하여 첨부 (\*.java, \*.txt or \*.db, README.md 등 포함)
  - 외부 라이브러리 사용 시, 프로젝트 내 lib 디렉토리에 해당 파일을 넣어서 함께 제출  
(만약 외부 라이브러리 파일 크기가 1MB 초과 시 README.md에 해당 라이브러리 다운로드 가능한 링크 명시 후 첨부하지 않기)
  - IntelliJ 외의 IDE 활용 시 Main 함수가 들어있는 Class 상단에 주석으로 IDE 이름 또는 개발 환경 명시
- 압축 파일명: cp2023project2\_[학번]\_[이름].zip  
(예시: cp2023project2\_20xx-xxxxx\_홍길동.zip)

# Appendix

## 프로젝트 진행 방향 제안 (초보자용)

1. 이해가 가지 않는 키워드나 내용의 인터넷 검색 도움을 받으면서 제공하는 PDF 문서를 정독한다.
2. 제공되는 파일을 직접 로컬 환경에서 실행하여 기본 동작을 확인한다.
  - SimpleHttpServer.java 내 main 메소드를 실행
  - 브라우저에서 <http://localhost:8080>에 접속 후, 입력 박스에 정보 입력 및 버튼 클릭 여러 번 시도하기
  - 이 때, Java 프로그램이 어떤 문구를 출력하는지, 관련 코드는 무엇인지 확인하며 이해하기
  - SimpleHttpClient.java 내 main 메소드를 실행
  - url 변수를 변경하면서 어떤 결과가 출력되는지, 관련 코드는 무엇인지 확인하며 이해하기
3. 간단한 기능부터 하나씩 구현한다.
  1. 메소드명, 변수명이 길어져도 OK, 직관적으로 이해할 수 있으면 좋음
  2. 주요 클래스, 메소드, 필드에 대해 inline 주석을 기재하여 다른 사람이 보아도 이해가 쉽도록 개발 진행
4. 구현된 기능들을 어떻게 구성하고 연결할지 그림을 그려가며 설계한다.
5. 구현이 완성되면 요구사항과 비교해가며 놓친 부분이 없는지 검토한다.