GSDS Graduation Requirements Check & Completion History Sharing System

Chaeyun Kim, Hyeryung Son, Younghun Kim, Jihoo Park, Minwoo Park

SNU Graduate School of Data Science

Table of contents

Motivation of the Project

Motivation of the Project : customized graduation requirement checking system

Expected Target : GSDS Students

- Expected graduates, along with those who want to plan their graduation in advance

User Request

Graduation Requirement Checking

User / Content-based Course Recommendations

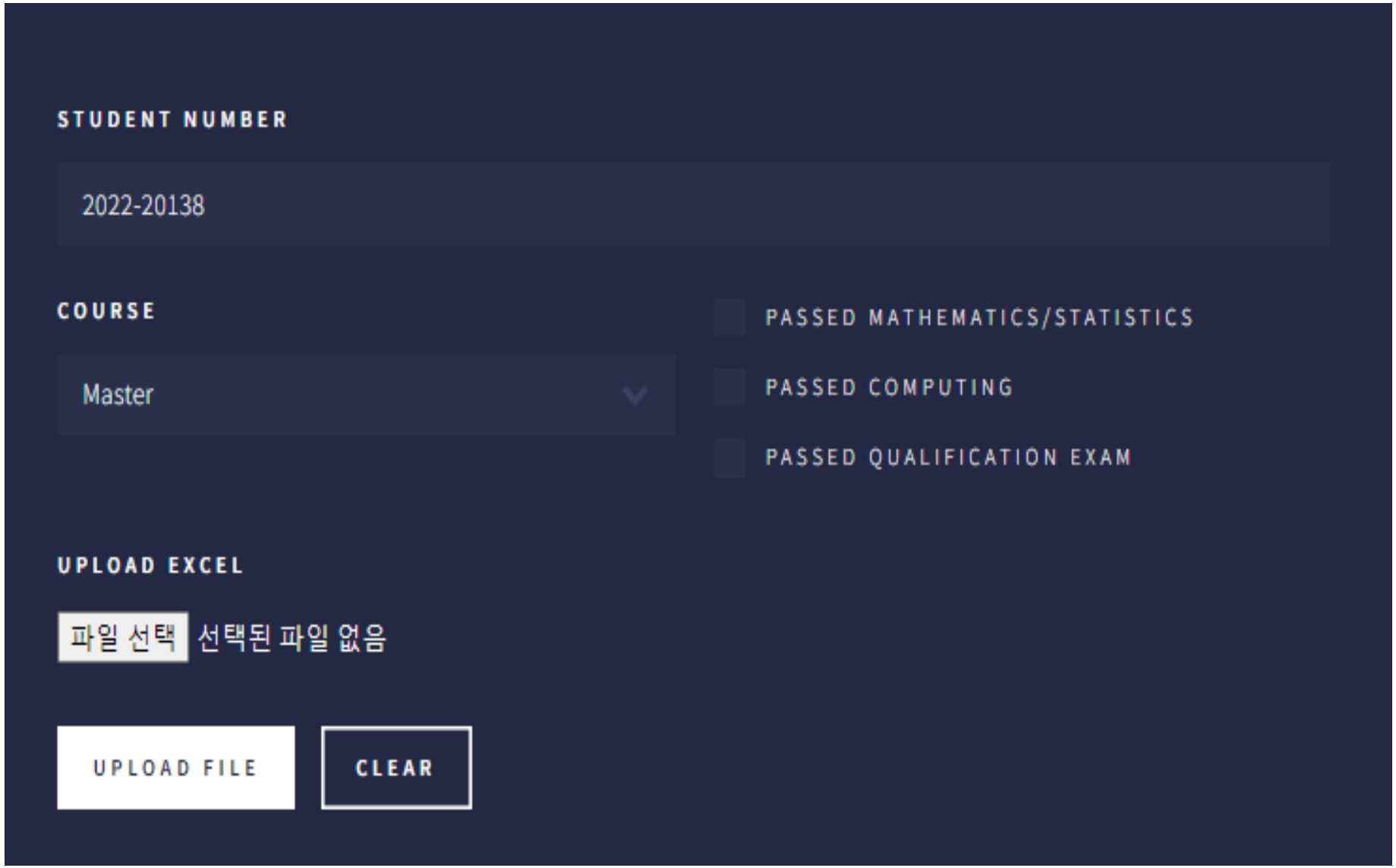
View the course history of other (agreed) classmates

vs SNU portal

vs SNU Genie

User Scenario

Graduation Requirement Checker



My Page : Check the current result

My Page

- Name : 김채윤 / 학생 / 데이터사이언스학과
- Student Number : 2022-20138
- Course : 석사

☑ INCLUDE UNDERGRADUATE SUBJECTS ☑ DO YOU WANT TO SHARE YOUR TIMETABLE?

This is your academic completion status as of the year 2022. (This information is basic information based on the completion standards. For detailed completion requirements, please contact each department office.)

Earned Credit information (Current Simulation Date : 2022/12/04)

Passed Conditions

수료 학점	세미나 검사	공통 과목 검사	논문 연구 검사	선택 과목 검사	수료 학점 검사	석사 전필과목 검사	공통 과목 검사(세미나 제외)
29	Failed	Passed	Passed	Passed	Passed	Failed	Passed

Course Recommendation result

Option : View Timetable of others

Shared Timetables

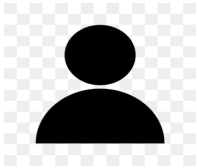
Timetables of Anonymous User

Year 2020	Year 2021
데이터사이언스를 위한 머신러닝 및 딥러닝	빅데이터 및 지식 관리 시스템
데이터사이언스 세미나 특강	데이터사이언스 캡스톤 프로젝트
데이터 사이언스를 위한 소프트웨어 플랫폼	데이터사이언스와 강화학습
시각적 이해를 위한 기계학습	대학원 논문연구
대학원 논문연구	
텍스트 및 자연어 빅데이터 분석 방법론	
데이터사이언스 세미나 특강	

Assumption of Workloads : Graduation Checker

OLTP (Data Input)

Excel file format



GSDS student

Direct user-input

OLAP (Query Processing)

- **Conditional checks such as total credits and whether the minimum required courses for each category have been met are executed through a query.**

- **Analysis of student course information data is performed, and the results of the analysis can be utilized in a similar user-based recommendation system.**

Assumption of Workloads : Recommendation

Requirements

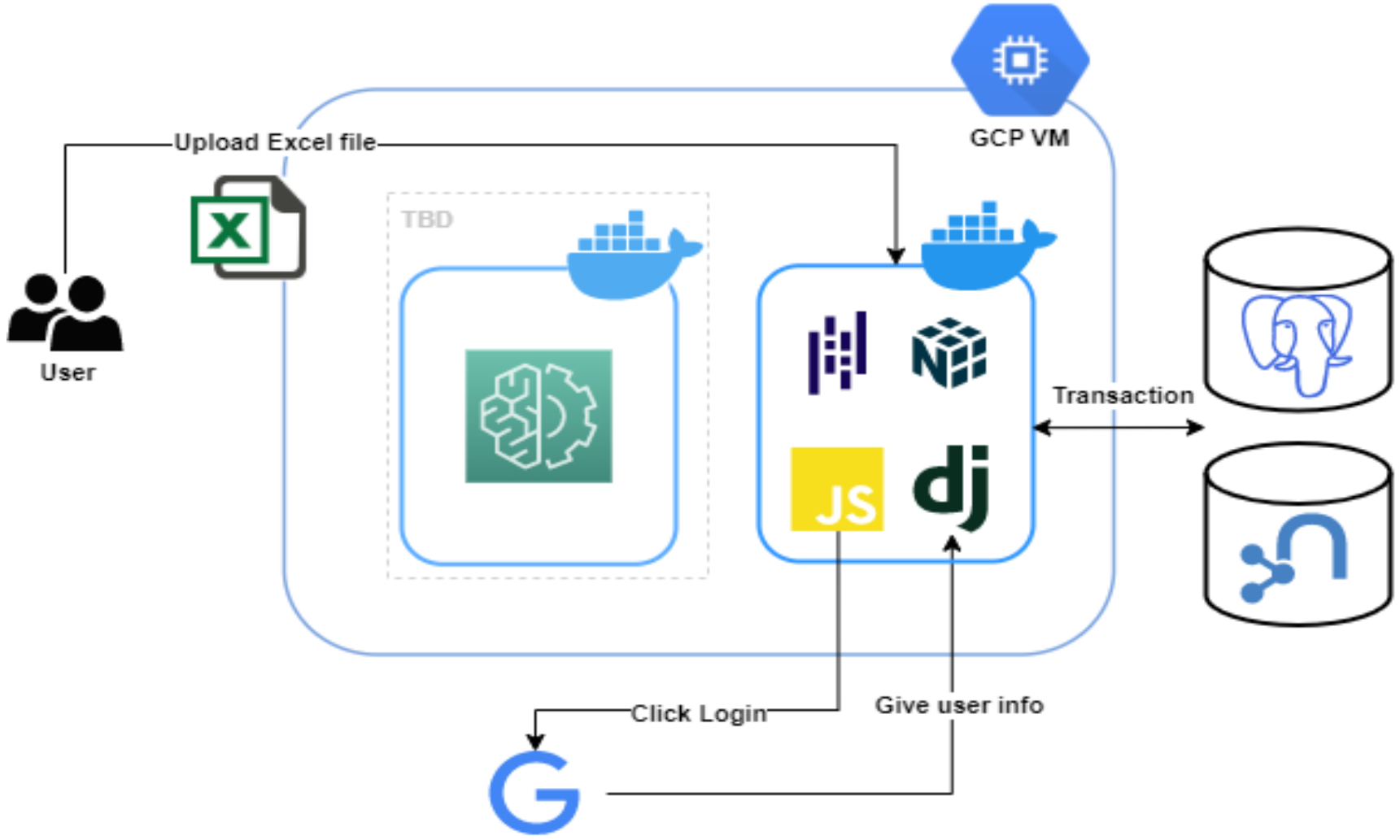
- User wants to get recommendations on which class to take in the future
- Using user course history and course information, we could provide the recommendation.

Assumption Details

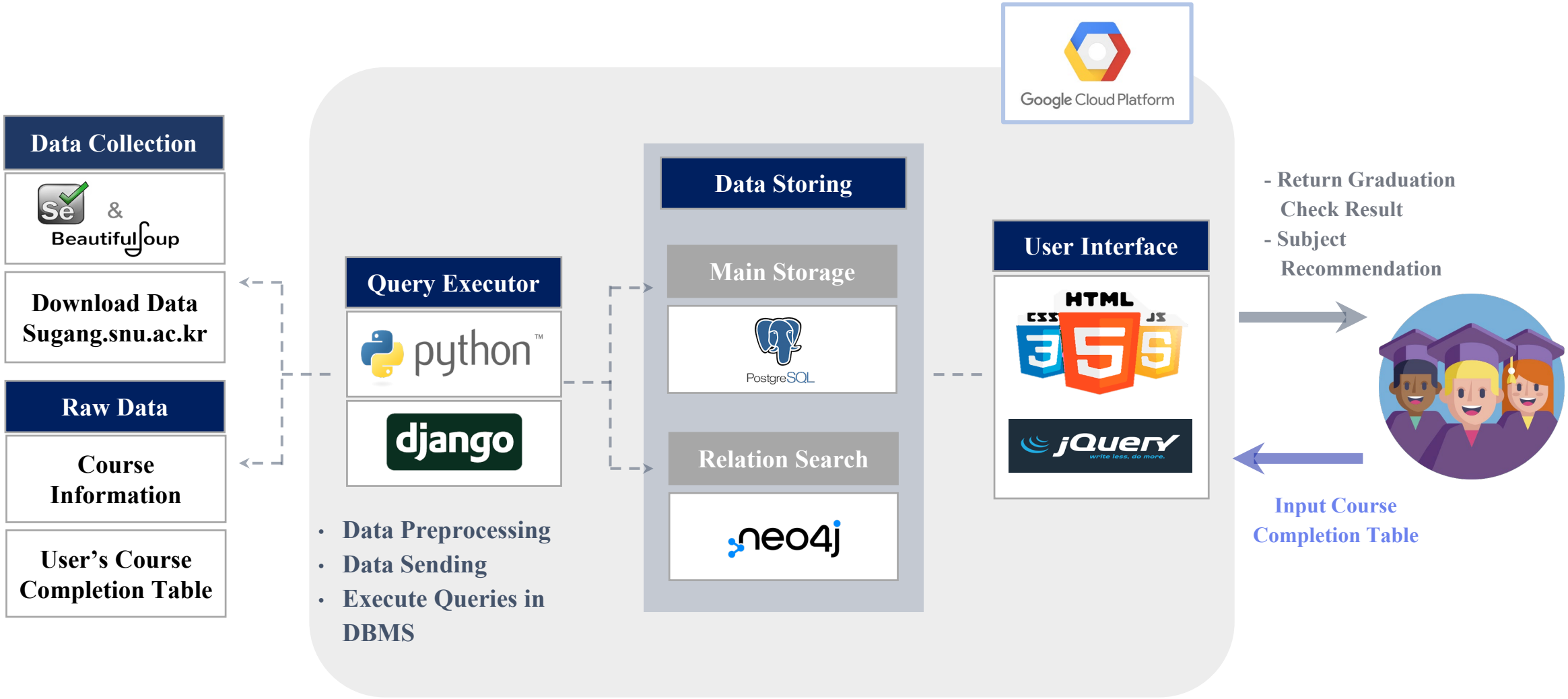
- User prefers courses that are relevant to their interests.
- These interests are reflected in previously taken courses.

Architectural Design

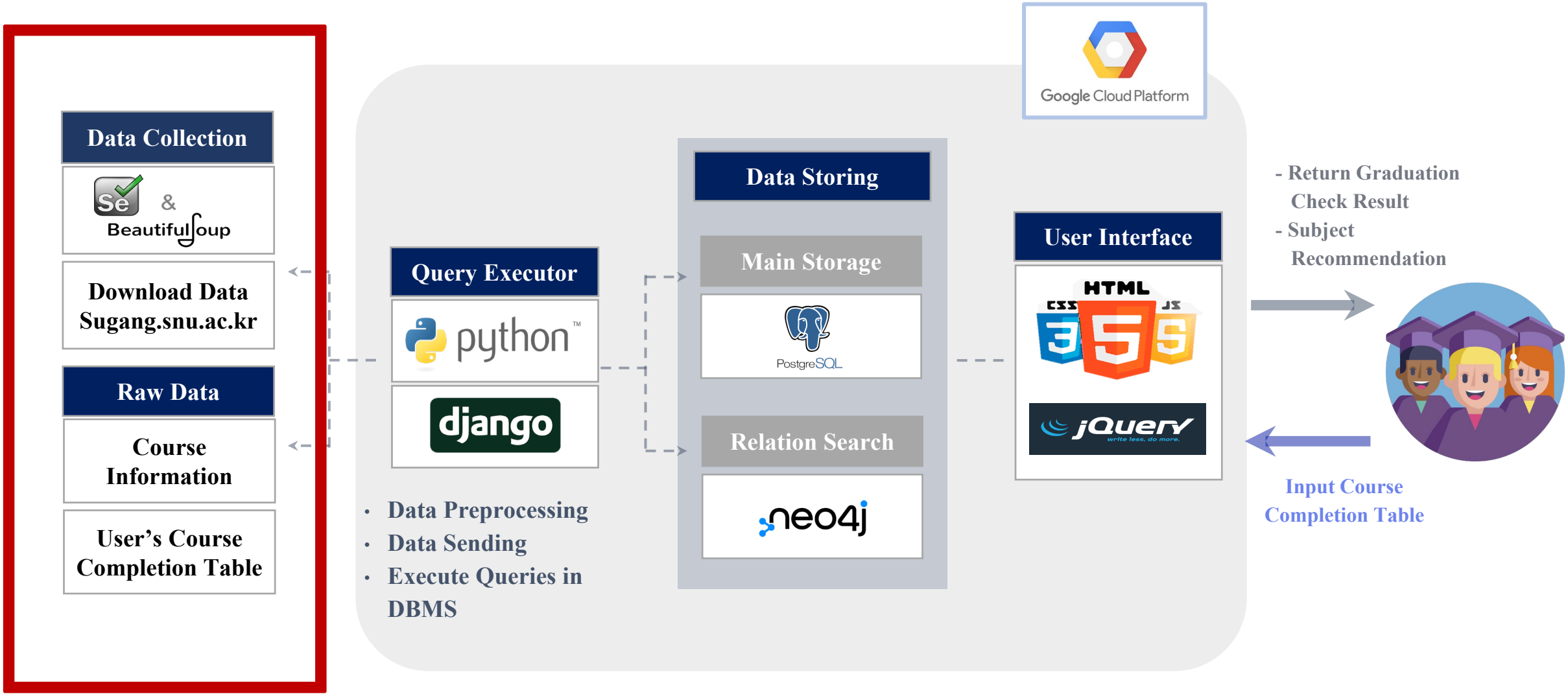
(Review) Preliminary Architectural Design



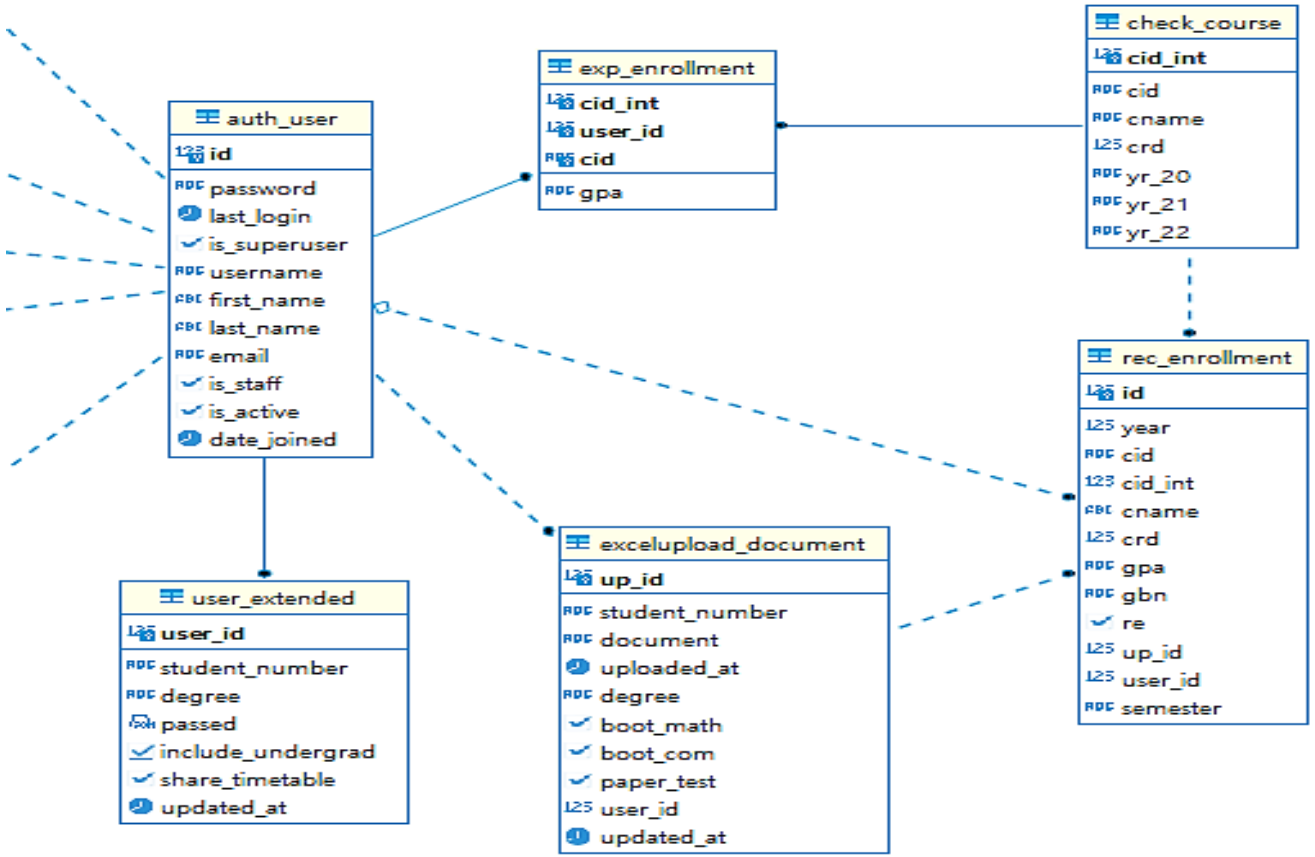
Architectural Design (Demo Version)



Part 1. Dataset and Data Collection



01. Dataset : ER Diagram of User-Course Data



02. Data Collection : Graduation Checker

1) User's Course Completion Dataset

	year	semester	cid	cid_int	cname	crd	gpa	gbn	re
0	2020	1학기	M3239.000200	9563	데이터사이언스를 위한 머신러닝 및 딥러닝	3	A+	계산	N
1	2020	1학기	M3239.000900	9568	데이터사이언스 세미나 특강	1	S	공통	N
2	2020	1학기	M3239.000300	9564	데이터 사이언스를 위한 소프트웨어 플랫폼	3	A0	계산	N
3	2020	2학기	M3224.000100	9522	시각적 이해를 위한 기계학습	3	A+	계산	N
4	2020	2학기	M3239.002000	9571	대학원 논문연구	3	S	논문	N
5	2020	2학기	M3239.001100	9570	텍스트 및 자연어 빅데이터 분석 방법론	3	A+	계산	N
6	2020	2학기	M3239.000900	9568	데이터사이언스 세미나 특강	1	S	공통	N
7	2021	1학기	M3239.000100	9562	빅데이터 및 지식 관리 시스템	3	A+	계산	Y
8	2021	1학기	M3239.001000	9569	데이터사이언스 캡스톤 프로젝트	3	S	공통	N
9	2021	1학기	M3239.004100	9585	데이터사이언스와 강화학습	3	A+	계산	N
10	2021	2학기	M3239.002000	9571	대학원 논문연구	3	S	논문	N

2) Course Information (2020-2022)

	cid_int	cid	cname	crd	yr_20	yr_21	yr_22
9834	9594	M3239.005300	데이터사이언스를 위한 머신러닝 및 딥러닝 1	3	계산	계산	석사전필
9835	9595	M3239.005400	데이터사이언스를 위한 컴퓨팅 2	3	계산	계산	석사전필
9836	9596	M3239.005500	데이터사이언스를 위한 컴퓨팅 1	3	계산	계산	석사전필
9837	9597	M3239.005600	데이터사이언스를 위한 컴퓨팅의 기초	3	기초	기초	기초
9838	9598	M3239.005700	빅데이터 및 지식 관리 시스템 1	3	계산	계산	석사전필
9839	9599	M3239.005800	데이터사이언스를 위한 수학과 통계의 기초	3	기초	기초	기초
9840	9600	M3239.005900	앰비언트 인공지능 플랫폼 및 실습	3	계산	계산	선택
9841	9601	M3239.006500	머신러닝을 위한 연속형 수리적 방법	3	응용	응용	선택
9842	9602	M3239.006700	데이터사이언스 프로젝트	3	응용	응용	석사전필
9843	9603	M3239.006800	고차원데이터 분석을 위한 기하학적 방법	3	응용	응용	선택

02. Data Collection : Recommendation System

1) Meta-Data of Course Information

	cid	program	cname	latest_classification	department	language	latest_year_semester
0	031.001	학사	글쓰기의 기초	교양	인문대학_국어국문학과	한국어	2022-2학기
1	031.002	학사	인문학글쓰기	교양	기초교육원	한국어	2022-1학기
2	031.004	학사	과학과 기술 글쓰기	교양	기초교육원	한국어	2022-2학기
3	031.031	학사	말하기와 토론	교양	기초교육원	한국어	2022-2학기
4	031.032	학사	창의적 사고와 표현	교양	기초교육원	한국어	2022-2학기
...	...	...	...	...	...	...	...
9677	252.505	대학원	금융기관경영론연구	전선	경영대학_경영학과	영어	2022-2학기
9678	M1338.002800	학사	재무와 기계학습	전선	경영대학_경영학과	한국어	2022-2학기
9679	M1338.003800	대학원	생산서비스운영세미나	전선	경영대학_경영학과	영어	2022-2학기
9680	251.564A	대학원	촉진관리론세미나	전선	경영대학_경영학과	한국어	2022-2학기
9681	806.524A	대학원	인체질환과 후성유전체학	전선	의과대학_의과학과	영어	2022-2학기

02. Data Collection : Recommendation System

2) Course Abstracts

	cid	abstract
0	M2183.001200	4차 산업혁명이라 불리는 인공지능시대를 맞아 음악대학에서는 학생들...
1	M2194.002800	머신러닝 등 빅데이터를 활용한 인공지능 기술의 급속한 발전 및 사...
2	M2177.004900	본 강좌는 학부 3학년 교과목으로 전기·정보공학 및 컴퓨터공학 비...
3	654.592	대학원 과목이다. 국악과의 석사과정 기악 및 성악, 작곡전공 학생...
4	353.726A	상품유통에 대한 최근의 이론과 연구결과를 살펴보고 의류상품에서 특...
...	...	...
1933	430.523	확률제어와 추정론 및 진보된 현대제어 이론의 기초가 되는 Stoc...
1934	205.730A	본 강좌에서는 사회학적 연구가 기초한 다양한 설명의 논리들에 대해...
1935	5321.5604	바이오모듈레이션 전공 대학원생이 현재 수행하고 있는 연구 및 이를...
1936	5321.5404	이 교과목에서는 곤충생태학의 주요 개념들을 배운다. 예를 들어, ...
1937	941.589B	일상적으로 네트워크 시설의 계획과 운영관리에 치중해온 전통적 교통...

Crawled using Selenium

Dynamically crawled abstracts by searching courses with certain conditions. Limited search terms to crawl only relevant courses.

AI, 데이터(Data), 통계(Statistics), 딥러닝(Deep Learning), 머신러닝 (Machine Learning), 시각화(Visualization), 강화학습(Reinforcement Learning), 자연어처리(Natural Language Processing), 영상처리(Computer Vision), 인과추론(Causal Inference)

SNU CRS
서울대학교 수강신청 시스템

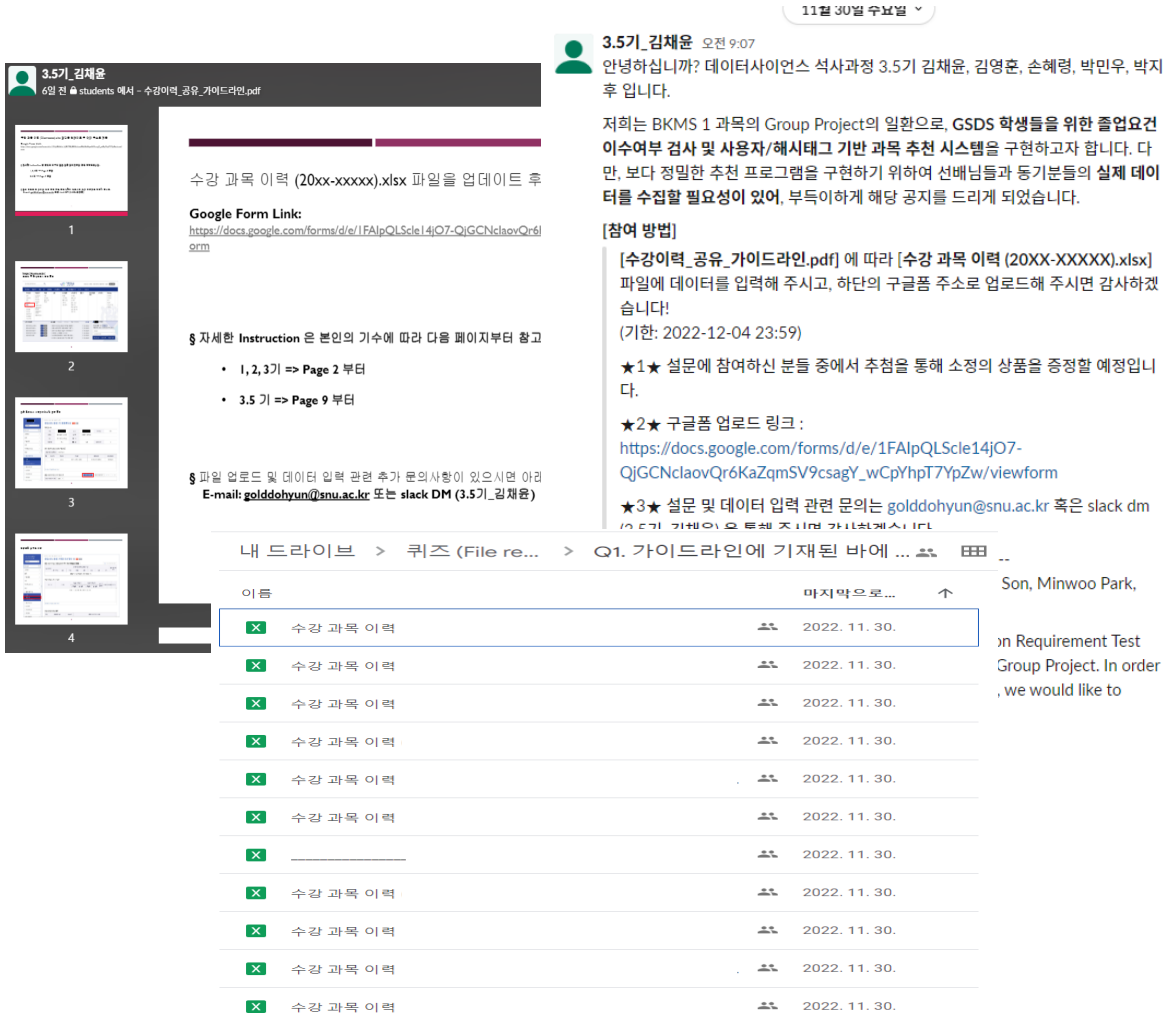
```
for query_keyword in query_keywords:
    for year in years:
        for semester in  semesters:
            driver=wd.Chrome(executable_path='chromedriver', options=opti
            driver.implicitly_wait(3)
            driver.get(url)
            driver.implicitly_wait(5)
            frame = driver.find_element(by=By.XPATH, value = '//*[@id="main"]')
            driver.switch_to.frame(frame)
            driver.implicitly_wait(3)

            # 돋보기 아이콘 클릭
            driver.find_element(by=By.XPATH, value='//*[@id="header"]/div[2]/div[1]/button[1]').click()
            driver.implicitly_wait(4)
            # 상세 조건 버튼 클릭
            driver.find_element(by=By.XPATH, value='//*[@id="HD100"]/div/div/div[1]/fieldset/div[3]/button[2]').click()
            driver.implicitly_wait(10)


            # 이전학기 조건 보기 클릭
            driver.find_element(by=By.XPATH, value='//*[@id="HD100"]/div/div/div[3]/div[2]/div/fieldset/button').click()
            driver.implicitly_wait(2)

            # 년도 변경
            select = Select(driver.find_element(by=By.XPATH, value='//*[@id="hSrchOpenSchyy"]'))
            select.select_by_visible_text(year)

            # 학기 변경
            select = Select(driver.find_element(by=By.XPATH, value='//*[@id="hSrchOpenShtm"]'))
            select.select_by_visible_text(semester)

            # 검색어 입력
            search = driver.find_element(By.XPATH, value='//*[@id="hMobileTotalSearch"]')
            search.clear()
            search.send_keys(query_keyword)
```

02. Data Collection : Recommendation System

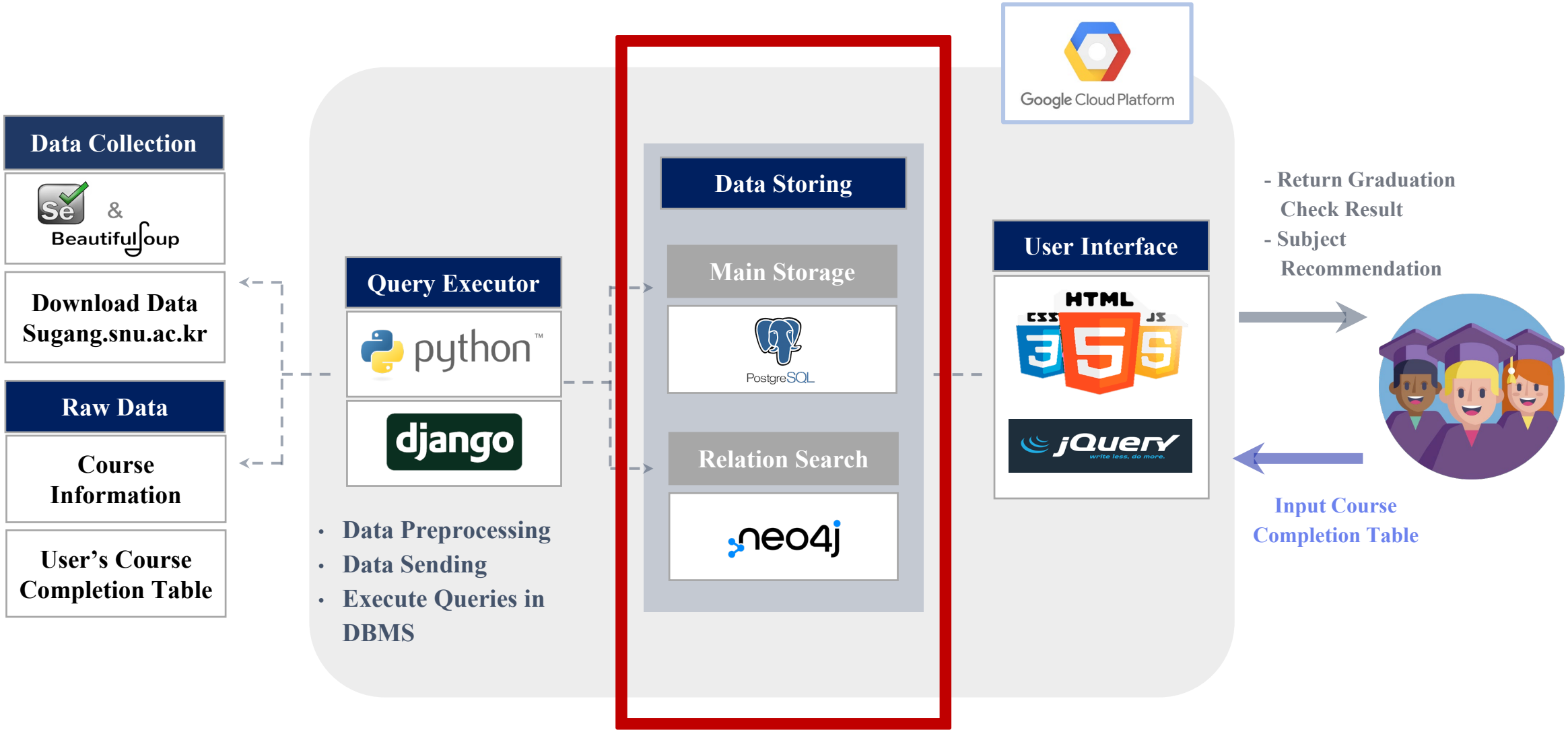
Gathered real data from GSDS students via survey!

3) User's Course Completion History

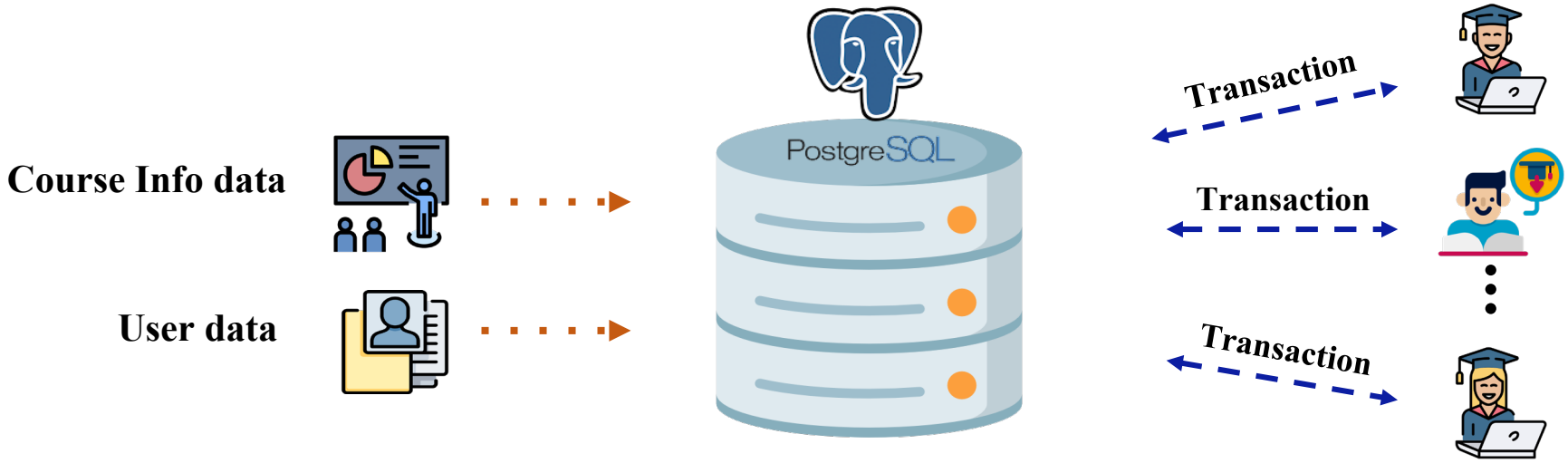
year	cid	cid_int	cname	crd	gpa	gbn	re	up_id	user_id
2,022	M3239.005700	9,598	빅데이터 및 지식 관리 시스템 1	3	NaN	NaN	[ ]	2	1
2,022	M3239.005400	9,595	데이터사이언스를 위한 컴퓨팅 2	3	NaN	NaN	[ ]	1	1
2,022	M3239.005300	9,594	데이터사이언스를 위한 머신러닝 및 딥러닝 1	3	NaN	NaN	[ ]	2	1
2,022	M3239.003300	9,579	데이터사이언스 세미나	1	NaN	NaN	[ ]	1	1
2,022	M3239.005000	9,591	데이터사이언스를 위한 머신러닝 및 딥러닝 2	3	NaN	NaN	[ ]	1	1
2,022	M3239.005700	9,598	빅데이터 및 지식 관리 시스템 1	3	NaN	NaN	[ ]	1	1
2,022	M3239.003300	9,579	데이터사이언스 세미나	1	NaN	NaN	[ ]	1	1
2,022	M3239.005500	9,596	데이터사이언스를 위한 컴퓨팅 1	3	NaN	NaN	[ ]	1	1
2,022	M3239.003300	9,579	데이터사이언스 세미나	1	NaN	NaN	[ ]	2	1
2,022	M3239.005000	9,591	데이터사이언스를 위한 머신러닝 및 딥러닝 2	3	NaN	NaN	[ ]	2	1
2,022	M3239.005100	9,592	빅데이터 및 지식 관리 시스템 2	3	NaN	NaN	[ ]	1	1
2,022	M3239.005500	9,596	데이터사이언스를 위한 컴퓨팅 1	3	NaN	NaN	[ ]	3	1
2,022	M3239.003300	9,579	데이터사이언스 세미나	1	NaN	NaN	[ ]	3	1
2,022	M3239.005700	9,598	빅데이터 및 지식 관리 시스템 1	3	NaN	NaN	[ ]	3	1
2,022	M3239.005400	9,595	데이터사이언스를 위한 컴퓨팅 2	3	NaN	NaN	[ ]	2	1
2,022	M3239.005000	9,591	데이터사이언스를 위한 머신러닝 및 딥러닝 2	3	NaN	NaN	[ ]	3	1
2,022	M3239.003300	9,579	데이터사이언스 세미나	1	NaN	NaN	[ ]	3	1
2,022	M3239.005100	9,592	빅데이터 및 지식 관리 시스템 2	3	NaN	NaN	[ ]	2	1
2,022	M3239.005300	9,594	데이터사이언스를 위한 머신러닝 및 딥러닝 1	3	NaN	NaN	[ ]	3	1
2,022	M3239.005100	9,592	빅데이터 및 지식 관리 시스템 2	3	NaN	NaN	[ ]	3	1
2,022	M3239.005400	9,595	데이터사이언스를 위한 컴퓨팅 2	3	NaN	NaN	[ ]	3	1
2,022	M3239.005500	9,596	데이터사이언스를 위한 컴퓨팅 1	3	NaN	NaN	[ ]	2	1
2,022	M3239.005300	9,594	데이터사이언스를 위한 머신러닝 및 딥러닝 1	3	NaN	NaN	[ ]	1	1



Part 2. Data Storing – Brief overview



03. Main Storage : PostgreSQL



Why PostgreSQL?

1. Concurrency Control

PostgreSQL boasts sophisticated features (compared to MySQL!) such as **Multi-Version Concurrency Control (MVCC)**, point in time recovery, **nested transactions, online/hot backups, a sophisticated query optimizer,** and write-ahead logging for fault tolerance.

2. Compatible with Django web framework tools

There are many ways to connect the PostgreSQL to different web framework tools.
Particularly, **PostgreSQL has the richest set of features that are supported by Django**.

04. Relational Search : Neo4j

Raw Data

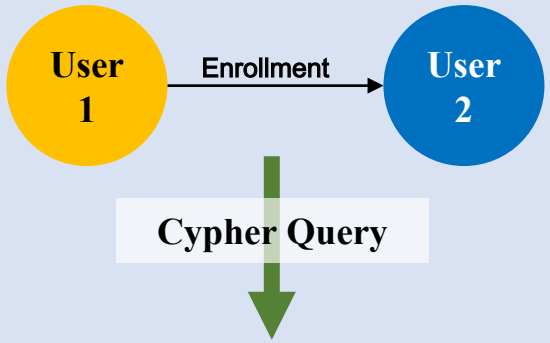
User Enrollment Record

Column Name	#	Data type	Identity	Collation	Not Null
year	1	int4			[ ]
cid	2	varchar(255)		default	[v]
cid_int	3	int4			[v]
cname	4	varchar(255)		default	[ ]
crd	5	int4			[ ]
gpa	6	varchar(50)		default	[ ]
gbn	7	varchar(50)		default	[ ]
re	8	bool			[v]
up_id	9	int4			[v]
user_id	10	int4			[ ]
id	11	int4	Always		[v]
semester	12	varchar(5)		default	[ ]

Courses Table

Column Name	#	Data type	Identity	Collation	Not Null
cid	1	varchar(255)		default	[v]
program	2	varchar(5)		default	[ ]
cname	3	varchar(100)		default	[ ]
avg_quota	4	float4			[ ]
classification	6	varchar(5)		default	[ ]
dept	7	varchar(255)		default	[ ]
lang	8	varchar(50)		default	[ ]
yr_sem	9	varchar(255)		default	[ ]
last_yr_sem	10	varchar(10)		default	[ ]
notes	11	text		default	[ ]

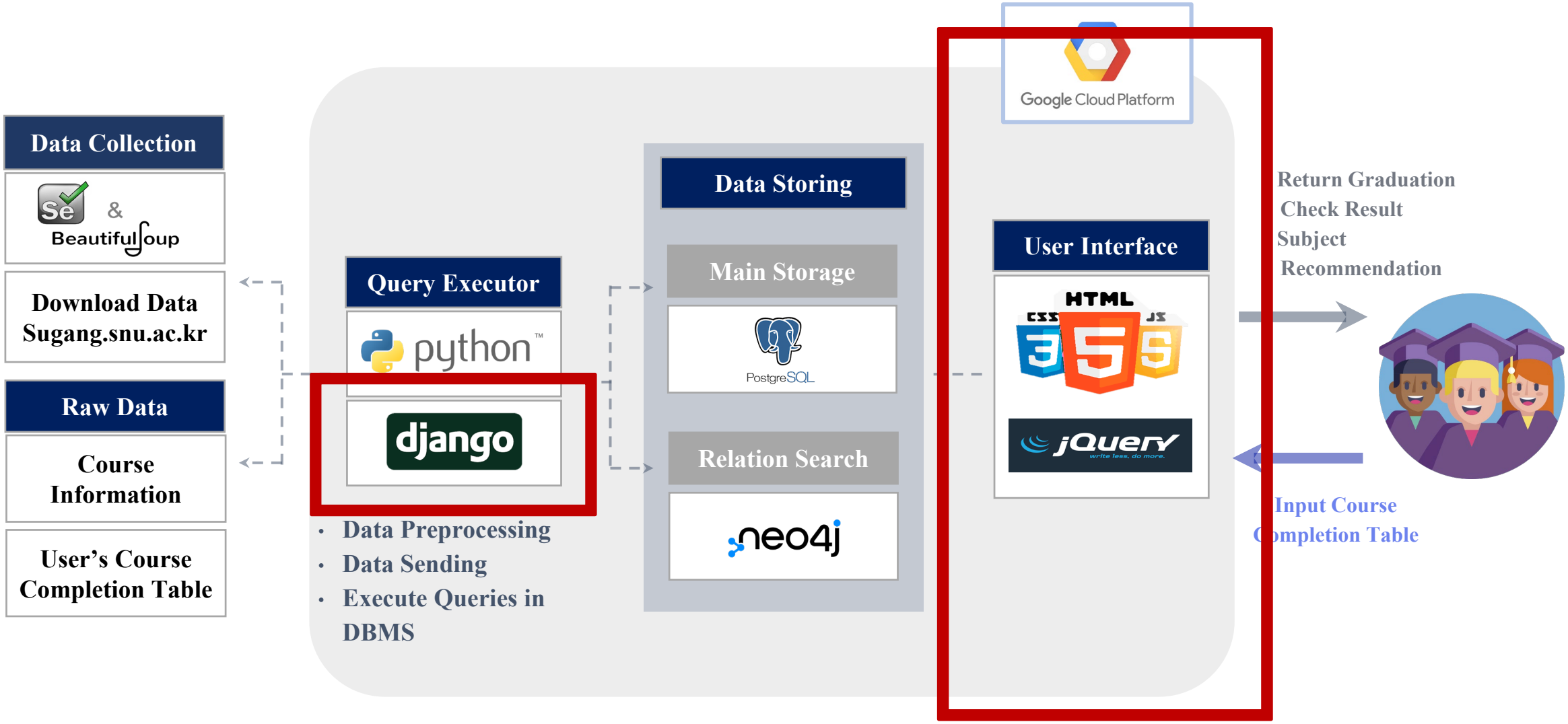
neo4j



Find "Candidate User"

Returns the ids of 'candidate users' who are similar in particular user's course history

Part 3. User Interface and Backend



05. Ui&Backend : Django

Django

Effective ML Model linkage !

Uses the Python web framework.
In particular, **basic foundation of the web architecture itself is firmly established**.

Why use Django?

Since the decision to implement the Recommendation System was made, a Python web framework was needed. Only one of the team members had experience in developing Java-based Spring Boot – thus the manpower base for the web framework using Python was relatively insufficient. However, there are many functions provided by Django, so it was the most suitable among python-based frameworks to compensate for and offset this shortcoming. Django's Form, Model, and ORM were used to create and implement database tables as objects.

How was it used?

Form(https://docs.djangoproject.com/en/4.1/topics/forms/):

Model Form has implemented an area where data is input from users. A model called "Document" was created so that data input from users could be uploaded to the server. The user-input data could be validated at the backend. After reading the uploaded Excel file, the table was converted into DataFrame, loaded into the Enrollment table of the database, and handed over to the Condition Checker and Recommendation View.
In addition, there are models that Django offers by default, one of which is **auth_user**. As this model was implemented as a table in PostgreSQL, it could be utilized in the database.

05. Ui&Backend : jQuery

JQuery

Simple Manipulation!

Jquery **simplifies** document object model (DOM) manipulation, event processing, and **all basic tasks without complicating HTML code**.

Why use jQuery?

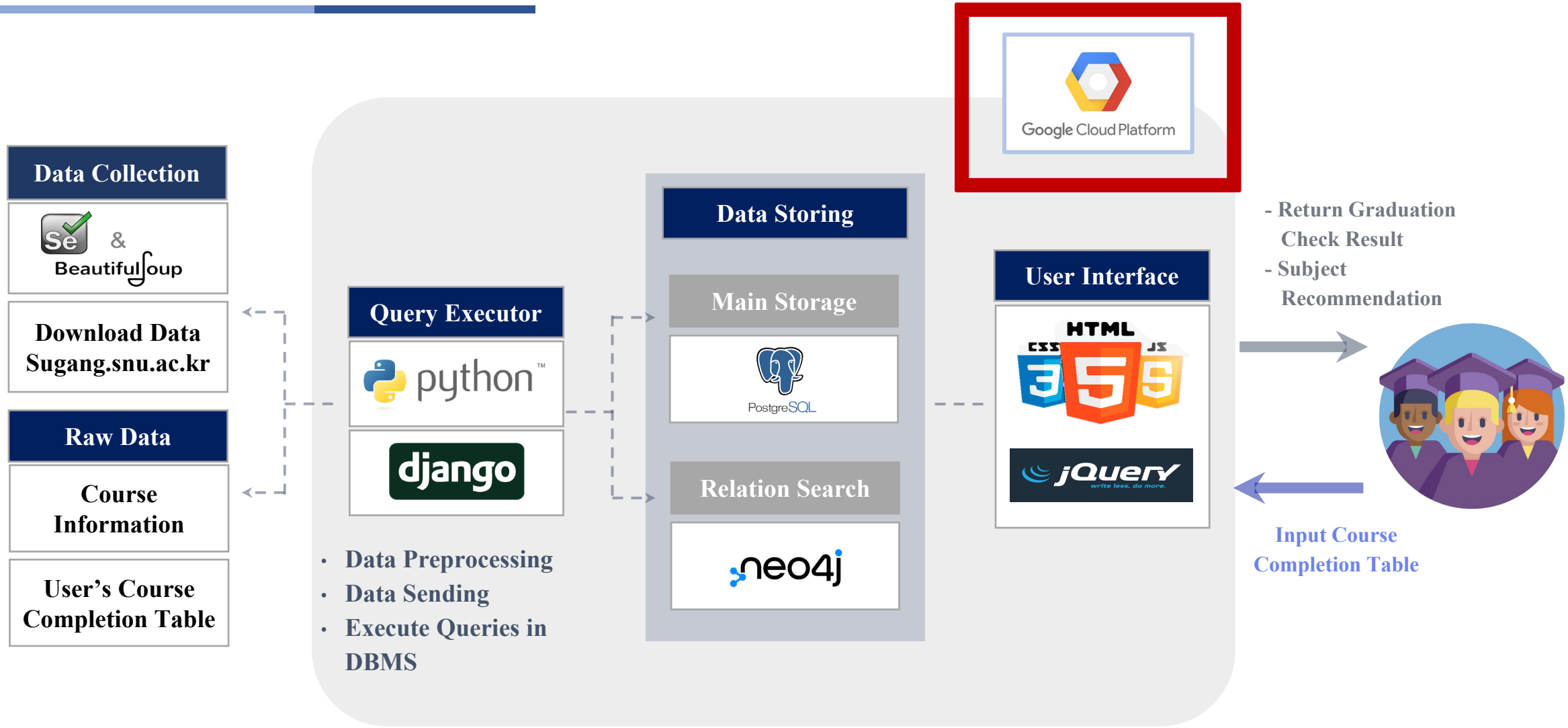
Currently, front-end frameworks such as React and Vue are mainly used to create websites, but they have learning curves. Since the current team members have little experience in developing the framework, jQuery was used - considering that the Learning Curve is shorter than other front-end services. Another advantage is that the query can be applied immediately by adding only jQuery CDN in the Javascript stage. Therefore, it is possible to shorten the frontend development time by applying jQuery directly from Django's template.

How was it used?

Form(https://docs.djangoproject.com/en/4.1/topics/forms/):

Most of the web framework was formed by HTML using Django's template, but jQuery was used to transmit the REST API immediately when the check box related to recommendation and sharing was changed on My Page. In addition, the subject card of the recommendation system and the timetable of the timetable sharing system are examples shown in HTML using Javascript and jQuery.

Part 4. CI/CD : Google cloud Platform



06. CI/CD : Google Cloud Platform

Why Google Cloud Service? (Using Google OAuth 2)

The current Seoul National University graduation requirement inspection system is implemented using the web service on the Seoul National University website. A cloud account logged in with the Seoul National University account is needed because it has the characteristic that it does not log in without logging in with the Seoul National University account. A cloud project was created with a team member's personal SNU account (jihoopark@snu.ac.kr) an OAuth authority was applied.

Why Cloud Run?

1) Non-disruptive deployment, integration with GitHub:

- Google has deployment tools such as compute engine and App Engine in addition to Cloud Run. Assuming that they push a new version through GitHub, there are ways to send build commands directly to the Google Cloud console or automate them using CI/CD tools such as Jenkins and Travis CI. So if the version changes, it takes a lot of time to roll back to the previous version. However, when distributed to Cloud Run, it is distributed as soon as it merges into the master branch of GitHub, and the previous versions are also backed up, so if there is an error in a hurry, a quick rollback is possible.

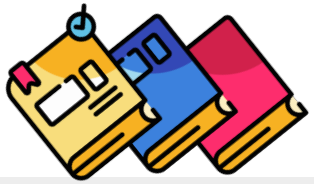
2) Simple server setup with Dockerfile:

If web deployment is made within the Docker container, the developer's local environment and the web environment will be the same. Therefore, it has the advantage that errors that may occur due to differences in the Python version do not occur.

Query Executor

Graduation Requirement Condition Checking

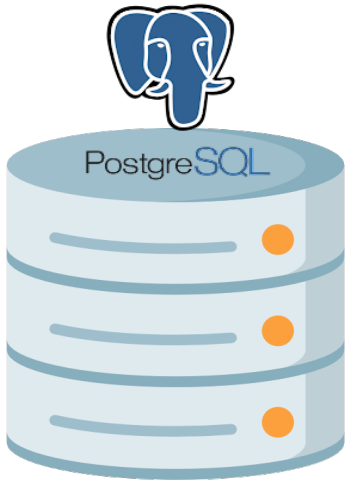
03. Query Executor



Information on all subjects by year since 2020

Gather Info in Excel

Database Normalization



cid → **cid_int**

34	2019	여름학기	M2170.004600	10003	유럽연합(EU)의 이해	3	A0	외부 교과	N

1. Convert the data type of course ID [cid] : VARCHAR to INT
2. **Purpose of introducing [cid_int]** : Process [cid] that emerges for the first time on the user's report card
   (For Exception Processing)

03. Query Executor : Handling Exceptions

9565	M3239.00(	데이터사이언스를 위한 통계
9576	M3239.00:	데이터사이언스를 위한 확률과 통계
9599	M3239.00!	데이터사이언스를 위한 수학과 통계의 기초

01. Subject categories applied by class are different

A column for each class was created for the Data Science subject table

cname	crd	yr_20	yr_21	yr_22
데이터사이언스를 위한 머신러닝 및 딥러닝 1	3	계산	계산	석사전필
데이터사이언스를 위한 컴퓨팅 2	3	계산	계산	석사전필
데이터사이언스를 위한 컴퓨팅 1	3	계산	계산	석사전필
데이터사이언스를 위한 컴퓨팅의 기초	3	기초	기초	기초
빅데이터 및 지식 관리 시스템 1	3	계산	계산	석사전필
데이터사이언스를 위한 수학과 통계의 기초	3	기초	기초	기초
앰비언트 인공지능 플랫폼 및 실습	3	계산	계산	선택
머신러닝을 위한 연속형 수리적 방법	3	응용	응용	선택
데이터사이언스 프로젝트	3	응용	응용	석사전필
고차원데이터 분석을 위한 기하학적 방법	3	응용	응용	선택

02. Name and subject code are different by year, but they are practically the same subject

Category classification conditions were applied by setting the same subject list.

```
def delete_redundant(course_checker:dict #입학 연도별 수강한 과목들을 분류한 dict -> category : cid_int
                     ,course_list:list #[cid_int] ex) computing1 = [9596,9654]
                     ,category='계산',
                    ):
    #같과다코 = 같은과목다른코드
    #같과다코가 해당 카테고리에 존재할경우
    if course_list[0] in course_checker[category] and course_list[1] in course_checker[category]:
        if dic_grade_df[course_list[0]]['re'].iloc[0]=='N': #그중 하나가 재수강이 아니면
            course_checker[category].remove(course_list[0])#그 것을 지운다 (재수강인 녀석이 유의하므로)
        elif dic_grade_df[course_list[1]]['re'].iloc[0]=='N':
            course_checker[category].remove(course_list[1])
        else : #만약 둘다 재수강일 경우 -> 둘다 여러번 들은 경우
            grade1=dic_grade_df[course_list[0]]['gpa'].iloc[0]
            grade2=dic_grade_df[course_list[1]]['gpa'].iloc[0]
            if grade_alph_float[grade1]>grade_alph_float[grade2]: #성적이 높은녀석을 살린다.
                course_checker[category].remove(course_list[1])
            else:
                course_checker[category].remove(course_list[0])
```

01. Graduation Condition Checker

Convert existing course id to primary key(cid_int) + Exception handling

```
def translate(cid:str): # cid -> cid_int transaction (모르는 cid 는 -1 내뱉기)
    cur = conn.cursor()
    # conn.commit()
    try:
        query="select * from course_key where cid='"
        cur.execute(query+cid+"'")
        result = cur.fetchall() # result에 결과 저장

        # 쿼리문 실행 결과를 pandas dataframe 형식으로 만들기
        my_df = pd.DataFrame(result)
        my_df.columns = [desc[0] for desc in cur.description]
        cid_int=my_df['cid_int'].iloc[0]
        return cid_int
    except:
        return -1

translate('M3507.000800')
```

01. Graduation Condition Checker

Access information related to the subject through cid_int

```
#colname in [cid_int,cid,cname,crd,yr_20,yr_21,yr_22]
def search(cid_int:int,colname=None)->pd.DataFrame:
    conn = psycopg2.connect(conn_string)
    cur = conn.cursor()
    # conn.commit()
    query="select * from check_course where cid_int="
    cur.execute(query+str(cid_int))
    result = cur.fetchall() # result에 결과 저장

    # 쿼리문 실행 결과를 pandas dataframe 형식으로 만들기
    my_df = pd.DataFrame(result)
    my_df.columns = [desc[0] for desc in cur.description]
    if not colname:
        return my_df
    else:
        return my_df[colname].iloc[0]
print(search(9840,'cname'))
search(9840)
```

	cid_int	cid	cname	crd	yr_20	yr_21	yr_22
0	9840	T2184.001900	교과-소프트웨어 융합역량 교육론 I	2	외부교과	외부교과	외부교과

01. Graduation Condition Checker

Converting report card (in Excel form) files into DataFrame form

```
#해당 성적표를 우리가 원하는 양식으로 고친다.
def changer(excel_name:str,admission:int):
    grade_xlsx=grade_sample=pd.read_excel(excel_name,engine='openpyxl')
    size_row=len(grade_xlsx)
    info=[] #성적표가 담길 리스트(이 중 리스트 후 DataFrame 으로 변경)

    for i in range(size_row): #모든 row 에 대하여 검사를 한다.
        one_row=list(grade_xlsx.iloc[i,:]) #각 로우를 볼건데
        if pd.notna(one_row[1]): #그 로우의 1번째 인덱스가 수업연도여야 볼거야.
            try:
                int(one_row[1])
            except :
                continue
            #유의미한 row 필터링 완료.
            new_info=[]
            new_info.append(one_row[1]) #연도
            new_info.append(one_row[2]) #학기
            new_info.append(one_row[3]) #교과목 번호
            course_key=translate(one_row[3]) #교과목 번호
            if course_key==-1:
                if one_row[5]==3 or one_row[5]=='3':
                    course_key=10003
                elif one_row[5]==2 or one_row[5]=='2':
                    course_key=10002
                else :
                    course_key=10001
            new_info.append(course_key)
            new_info.append(one_row[14]) #과목명
            new_info.append(int(one_row[5])) #학점
            new_info.append(one_row[6]) #성적
            new_info.append(search(course_key,dic_year[admission])) #교과구분
            new_info.append(one_row[10]) #재수강
            info.append(new_info)

    grade=pd.DataFrame(info)
    grade.columns=['year','semester','cid','cid_int','cname','crd','gpa','gbn','re']
    return grade

#테스트
grade_df=changer("GradeOfChaeyoon.xlsx",2020)
grade_df
```

	year	semester	cid	cid_int	cname	crd	gpa	gbn	re
0	2017	1학기	031.001	0	글쓰기의 기초	3	A+	외부교과	N
1	2017	1학기	042.044	204	한국음악의 이해	3	A+	외부교과	N
2	2017	1학기	044.023	267	북한학개론	3	A0	외부교과	N
3	2017	1학기	200.103	1261	정치학원론	3	A0	외부교과	N
4	2017	여름학기	045.013	280	인간관계의 심리학	3	A+	외부교과	N
5	2017	여름학기	051.025	351	배드민턴초급	1	S	외부교과	N
6	2017	2학기	032.049	54	초급스페인어 2	3	A+	외부교과	N
7	2017	2학기	043.004	207	근·현대 한국민족주의	3	A+	외부교과	N
8	2017	2학기	043.019	213	인물로 본 한국사	2	A+	외부교과	N

01. Graduation Condition Checker

Exception handling: Subjects that are the same subject but have different code numbers by year

```
def delete_redundant(course_checker:dict #입학 연도별 수강한 과목들을 분류한 dict -> category :
                     ,course_list:list #[cid_int] ex) computing1 = [9596,9654]
                     ,category='계산',
                    ):
    #같과다코 = 같은과목다른코드
    if course_list[0] in course_checker[category] and course_list[1] in course_checker[
        #같과다코가 해당 카테고리에 존재할경우
        if dic_grade_df[course_list[0]]['re'].iloc[0]=='N': #그중 하나가 재수강이 아니면
            course_checker[category].remove(course_list[0])#그 것을 지운다 (재수강인 녀석이 유
        elif dic_grade_df[course_list[1]]['re'].iloc[0]=='N':
            course_checker[category].remove(course_list[1])
        else : #만약 둘다 재수강일 경우 -> 둘다 여러번 들은 경우
            grade1=dic_grade_df[course_list[0]]['gpa'].iloc[0]
            grade2=dic_grade_df[course_list[1]]['gpa'].iloc[0]
            if grade_alph_float[grade1]>grade_alph_float[grade2]: #성적이 높은녀석을 살린다.
                course_checker[category].remove(course_list[1])
            else:
                course_checker[category].remove(course_list[0])
```

01. Graduation Condition Checker

Graduation Requirement Checker

```
def please_graduate(grade_df:pd.DataFrame, #성적표
                    admission:int, #입학년도
                    degree:str, #학위 (석사,박사,석박통합)
                    flag_basicComputing=True, #기초 컴퓨팅 통과 여부
                    flag_basicMath=True,# 논자시 통과 여부
                    nonjasee=True
                   ): #기초 수학 통과 여부

    # 중복되는 과목에 대해서 합쳐지므로 세미나 과목등 여러번 듣는 과목에 대해서 조심하기.
    dic_grade_df=dict() # 성적표상의 cid_int 로 성적표상의 row 로 접근하기 위한 dic
    row=0
    for i in grade_df['cid_int']:
        dic_grade_df[i]=grade_df.iloc[row:row+1,:][['year','cid','cname','crd','gpa','g
        row+=1

    grade_alph_float={'A+':4.3,'A0':4.,'A-':3.7,'B+':3.3,'B0':3,'B-':2.7,'C+':2.3,'C0':

    #개별 학생이 가질 정보
    course_checker=dict()
    credit_certificated=0 # 수료학점


    #기초 (컴기,확통) : 이수하여 B0 이상 혹은 개별 통과(청강통과-성적표기제 x, bootcamp)해야함.

    basic_math=[9599,9576,9565]
    # 데이터사이언스를 위한 수학과 통계의 기초, 데이터사이언스를 위한 확률과 통계, 데이터사이언스를 위한 통계
    basic_computing=[9597,9577,9566]
    # 데이터사이언스를 위한 컴퓨팅의 기초, 데이터사이언스를 위한 컴퓨팅, 데이터사이언스를 위한 프로그래밍

    #석사전필과목(계산) + 프로젝트

    a1=[9563,9594] # 데이터사이언스를 위한 머신러닝 및 딥러닝, 데이터사이언스를 위한 머신러닝 및 딥러닝 1
    a2=[9591] # 데이터사이언스를 위한 머신러닝 및 딥러닝 2

    b1=[9562,9598] # 빅데이터 및 지식 관리 시스템, 빅데이터 및 지식 관리 시스템 1
    b2=[9592] # 빅데이터 및 지식 관리 시스템 2

    c1=[9596,9564] # 데이터사이언스를 위한 컴퓨팅 1, 데이터 사이언스를 위한 소프트웨어 플랫폼
    c2=[9595,9582] # 데이터사이언스를 위한 컴퓨팅 2, 확장형 고성능 컴퓨팅


    # 세미나,세미나 특강
    seminar=[9579,9568]
    # 논문연구
    paper=[9571]
```

- **Through queries, information such as the report card data frame on the database, admission, academic classification, boot camp passing, and thesis qualification test passing is delivered on the back end.**

- **The information is stored in the form of a code number list of the subjects for processing subjects that require exception processing.**

01. Graduation Condition Checker

Conditions common to all students regardless of year: basic subjects

```
##########<모든학생에게 적용되는 부분>

#<기초 과목에 대한 테스트> : 모든 학생들에게 동일하게 적용됨.

#기초과목 통과여부 확인하기
basic_math_checker=flag_basicMath
if not flag_basicMath: #통과 못했으면
    for i in basic_math: #베이직 과목에 속하는 과목들에 대하여
        if i in set_course and grade_alph_float[dic_grade_df[i]['gpa'].iloc[0]]>2.99:
            # 그 과목을 수강했고 수강 학점이 3이상이면 (B0)
            basic_math_checker=True #체크
            break

basic_computing_checker=flag_basicComputing
if not flag_basicComputing:
    for i in basic_computing:
        if i in set_course and grade_alph_float[dic_grade_df[i]['gpa'].iloc[0]]>2.99:
            basic_computing_checker=True
            break
#같은과목이지만 코드번호가 다른 과목을 동시 수강하였을 경우 (둘중 하나는 재수강이다.) 재수강이 아닌것을 제거해야

print("기초 수학통계 검사 결과 :",basic_math_checker)
print("기초 컴퓨팅 검사 결과 :",basic_computing_checker)
print("논자시 통과 여부 :",nonjasee)
if basic_math_checker+basic_computing_checker<2 and nonjasee:
    print("기초 과목 통과 안하면 논자시 통과 못하는데? 확인 다시해보셈.")
```

01. Graduation Condition Checker

Sample code for 2020 graduation requirement test

```
########################################<2020>#############################################
if admission==2020:
    course_checker={'기초':[],'공통':[],'계산':[],'분석':[],'응용':[],'논문':[],'외부교과':[]}

    for row in range(len(grade_df)):
        if grade_df.loc[row,'gpa'] !='F' and grade_df.loc[row,'gpa'] !='U': # 성적이 F 
            course_checker[grade_df.loc[row,'gbn']].append(grade_df.loc[row,'cid_int'])

        #아직 논문연구 과목을 얼마나 들었는지에 대한 평가를 하지 않았기 때문에 모든 과목이 유효하다고 보장할 수
        #아직 계산과목에서 중복되는 것이 있을 수 있다.

    #중복제거
    delete_redundant(course_checker,a1)
    delete_redundant(course_checker,b1)
    delete_redundant(course_checker,c1)
    delete_redundant(course_checker,c2)

    #여러가지 졸업요건을위한 변수들
    checker_paper=True #논문
    valid_paper=None
    count_paper=len(course_checker['논문'])

    checker_calculate=True #계산
    count_calculate=len(course_checker['계산'])

    checker_seminar=True #세미나
    count_seminar=0

    checker_nonSeminar=True #공통-세미나
    count_nonSeminar=0

    checker_select1=True #분석
    count_select1=len(course_checker['분석'])

    checker_select2=True #응용
    count_select2=len(course_checker['응용'])
```

01. Graduation Condition Checker

Sample code for 2020 graduation requirement test (continued)

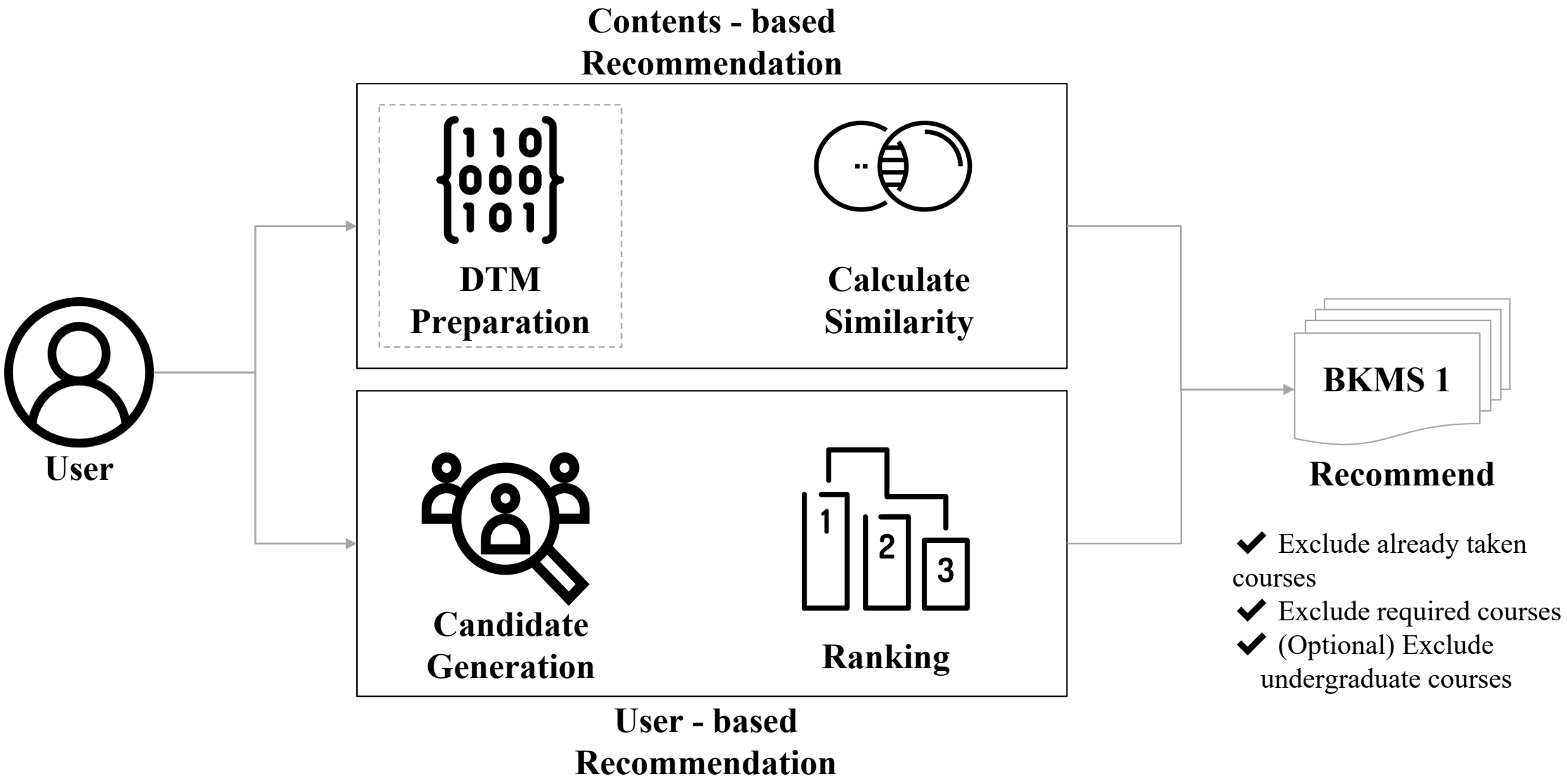
```
###공통 판단 : 공통에서 세미나와 그 외 과목을 나눠서 생각해야 한다.
for course in course_checker['공통']:
    if course in seminar: #세미나인가?
        count_seminar+=1 #세미나 갯수 추가

    else :
        count_nonSeminar+=1 #아니면 논 세미나 추가

if count_seminar<2: # 세미나 두번 미만으로 들었다면.
    checker_seminar=False

if count_nonSeminar==0: # 세미나가 아닌 공통과목을 한번도 안들었다면
    checker_nonSeminar=False

if degree=='석사': #석사라면
    if count_paper==0: # 논문과목을 한번도 안들었다면.
        checker_paper=False
    valid_paper=min(count_paper,2) #2과목을 넘어서면 의미없다.

    if count_calculate<3: # 계산과목을 3번 미만 들었다면
        checker_calculate=False

    if count_select1+count_select2==0: # 분석+응용을 한번도 안들었다면
        checker_select1=False
        checker_select2=False


else : #박사라면
    if count_paper<2: # 논문과목을 두번 미만으로 들었다면.
        checker_paper=False
    valid_paper=min(count_paper,4) #4과목을 넘어서면 의미없다.

    if count_calculate<5: # 계산과목을 5번 미만 들었다면
        checker_calculate=False

    if count_select1<2: # 분석을 2번 미만 들었다면
        checker_select1=False

    if count_select2<2: # 응용을 2번 미만 들었다면
        checker_select2=False
```

```
    credit_certificated=0

    for course in course_checker['기초']:
        if course in basic_math or course in basic_computing: #기초과목중 기초수학통계, 컴퓨팅
            continue
        credit_certificated+=search(course,'crd')

    credit_certificated+=valid_paper*3 #유효 논문연구강의 수강 횟수 * 학점(3)
    credit_certificated+=count_calculate*3 #계산과목 수강 횟수 * 학점(3)

    for course in course_checker['공통']:
        credit_certificated+=search(course,'crd')
    credit_certificated+=count_select1 #분석과목 수강 횟수 * 학점(3)
    credit_certificated+=count_select2 #응용과목 수강 횟수 * 학점(3)

    print_list=['논문 연구 검사','계산 과목 검사','공통 과목 검사(세미나 제외)','세미나 검사','분석 과목
    checkers=[checker_paper,
              checker_calculate,
              checker_nonSeminar,
              checker_seminar,
              checker_select1,
              checker_select2]
    for i in range(len(checkers)):
        print(print_list[i],checkers[i])
```

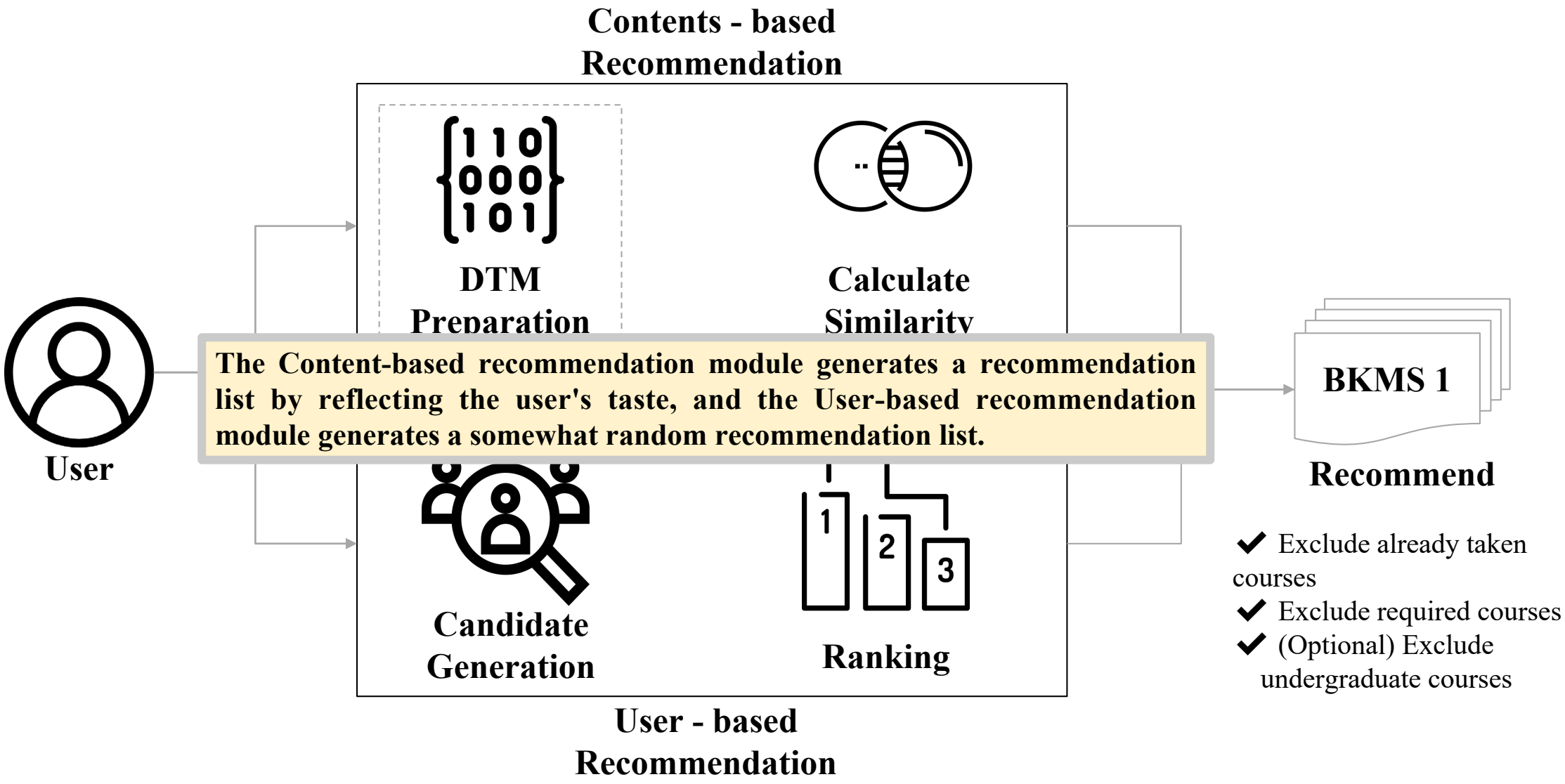
Recommendation System

Graduation Requirement Condition Checking

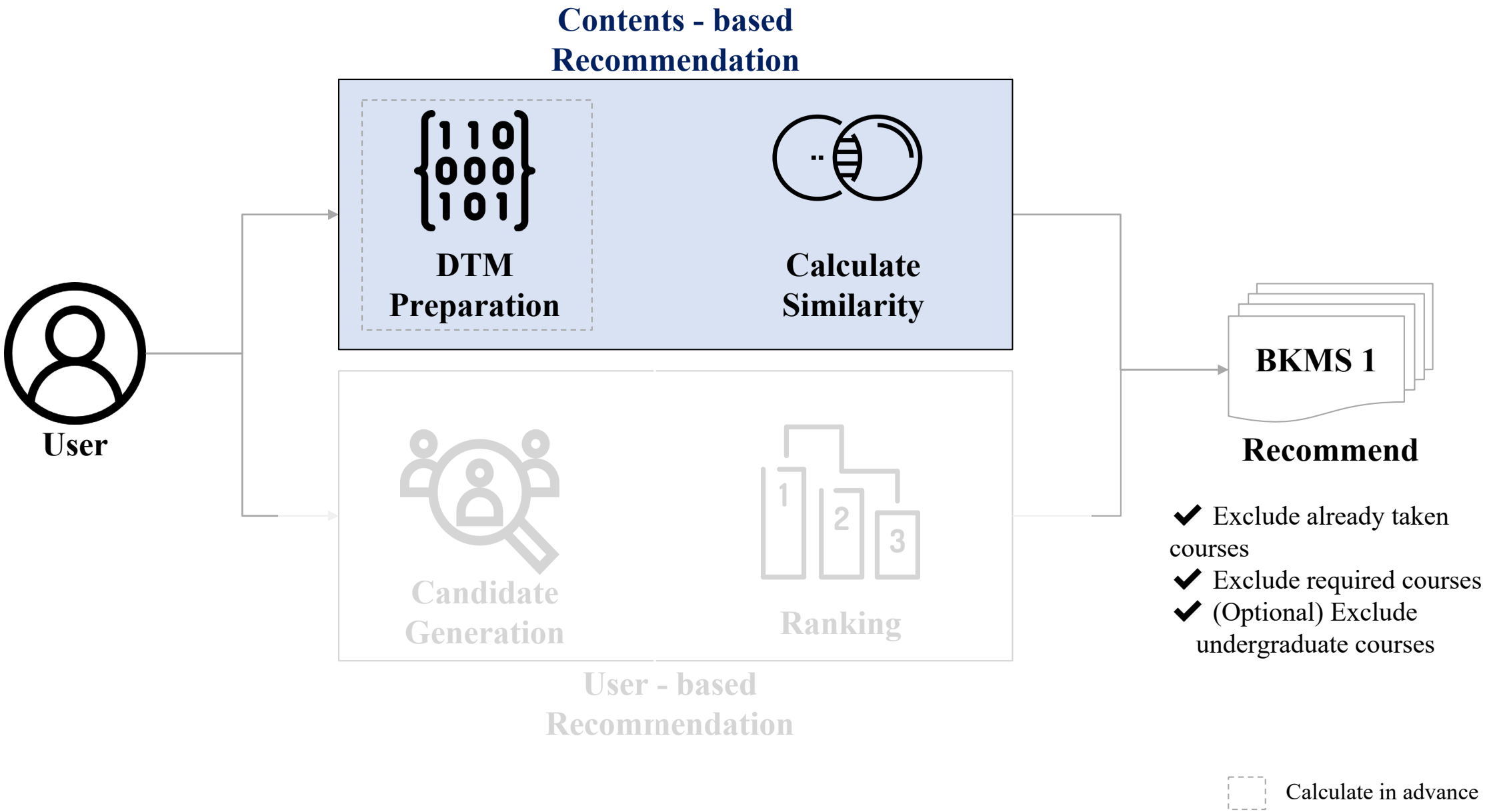
Overall Process



Overall Process

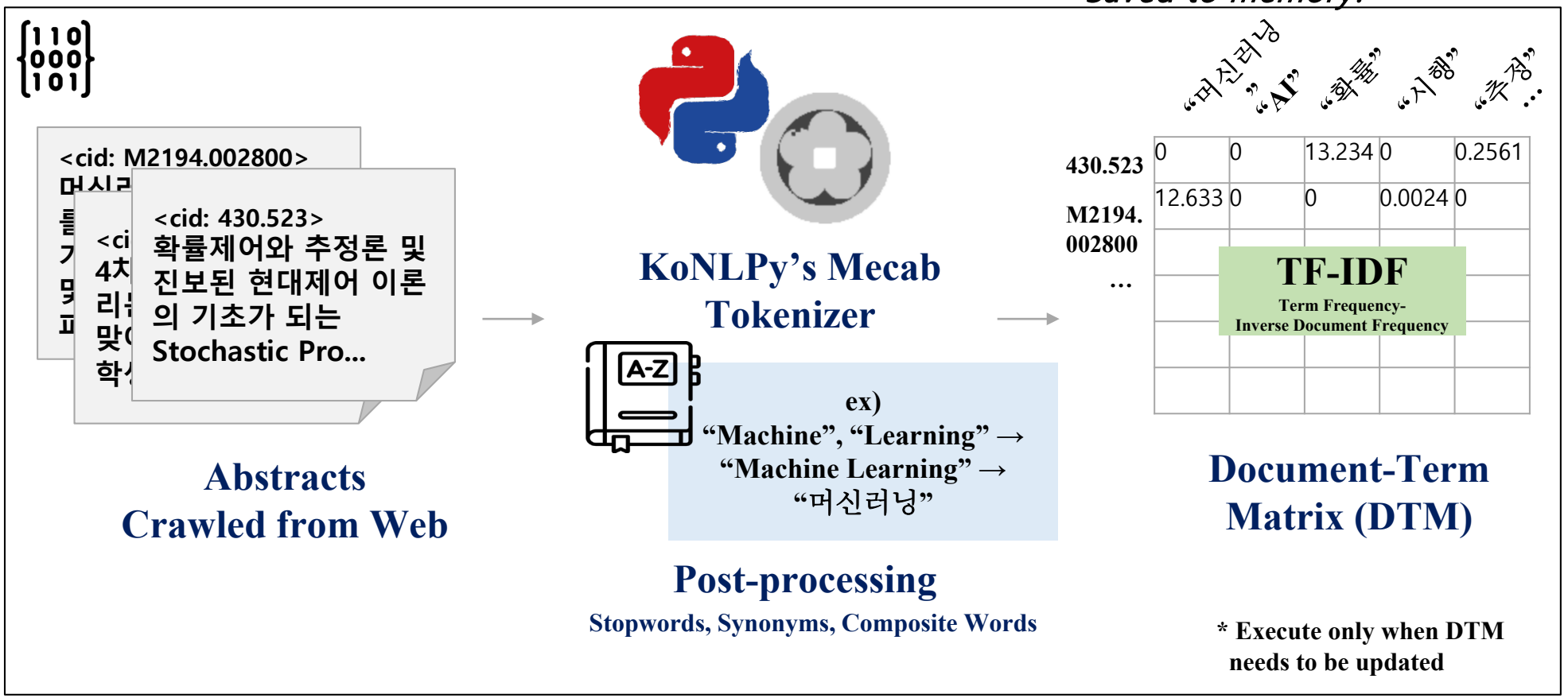


01. Content-Based Recommendation



01. Content-Based Recommendation

DTM Preparation



01. Content-Based Recommendation

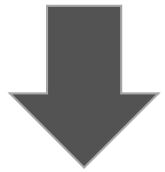
Tokenizing - Python code

```
class NLProcessor():
  def __init__(self, tagger, stopwords=None, synonyms=None, composites=None):
    self.tagger = tagger
    self.stopwords = stopwords
    self.synonyms = synonyms
    self.composites = composites # assumption: 2 words
    self.wordcount = Counter(

  def get_nouns_and_sl(self,
    pos = self.tagger.pos(tex
    nouns = [x[0] for x in pc
    nouns = [x.lower() for x
    # merge two consecutive t
    if self.composites is not
      for i in range(len(nour
        if tuple(nouns[i:i+2]
          repl = self.composi
          nouns = nouns[:i] +
          nouns.insert(i, rep
    # replace tokens with syn
    if self.synonyms is not N
      nouns = [self.synonyms[
    # remove stopwords
    if self.stopwords is not
      nouns = [x for x in nou
    nouns = [x for x in nouns
    return nouns

  def update_stopwords(self,
    if self.stopwords is None
      self.stopwords = stopwc
    else:
      self.stopwords = list(s

  def update_wordcount(self,
    for nouns in nouns_list:
        self.wordcount.update
```

```
stopwords_list = stopwords.words('english')
stopwords_list += "과목 학습 분야 집중 때문 강좌 목표 학생 배양 중요 요구 대학 \
역량 개설 수업 주요 내용 다양 최근 능력 마련 \
등장 자신 교과목 목적 최대한 강의 강좌 이해 시도 수강 한편 강조 관련 우리".split(" ")
composites_dict = {('머신', '러닝'): '머신러닝',
                   ('machine', 'learning'): 'machine learning',
                   ('artificial', 'intelligence'): 'ai',
                   ('neural', 'network'): 'neural network',
                   ('neural', 'networks'): 'neural network',
                   ('internet', 'things'): 'iot',
                   ('기계', '학습'): '기계학습',
                   ('데이터', '사이언스'): '데이터사이언스',
                   ('data', 'science'): 'data science',
                   ('빅', '데이터'): '빅데이터',
                   ('big', 'data'): 'big data',
                   ('컴퓨터', '비전'): '컴퓨터비전',
                   ('컴퓨터', '비젼'): '컴퓨터비전',
                   ('computer', 'vision'): 'computer vision',
                   ('인과', '추론'): '인과추론',
                   ('강화', '학습'): '강화학습',
                   ('이상', '탐지'): '이상탐지',
                   ('anomaly', 'detection'): 'anomaly detection',
                   ('reinforcement', 'learning'): 'reinforcement learning',
                   ('causal', 'inference'): 'causal inference'}
synonyms_dict = {'ai': '인공지능',
                 'accountability': '책임',
                 'data': '데이터',
                 'machine learning': '머신러닝',
                 'optimization': '최적화',
                 'computer': '컴퓨터',
                 'autoencoderes': 'autoencoder',
                 'autoencoders': 'autoencoder',
                 'perceptron': 'neural network',
                 'probability': '확률',
                 'data science': '데이터사이언스',
                 'computer vision': '컴퓨터비전',
                 '파이썬': 'python',
                 '기계학습': '머신러닝',
                 '베이즈': '베이지안',
                 'big data': '빅데이터',
                 'causal inference': '인과추론',
                 'reinforcement learning': '강화학습',
                 'anomaly detection': '이상감지',
                 'technology': '기술'}
```

Result

본 과목은 빠른 속도로 발전하고 있는 인공지능 분야에서 최근 중요 기술 및 논문들 중 선별하여 강의를 제공한다. 주요 논문과 최신기술에 관한 학습 및 논의, 팀별로 국제적 발표가 가능한 수준의 연구 프로젝트의 진행, 등을 목표로 한다. 주제는 알고리즘 분야 (active learning, classification, clustering, multitask and transfer learning, stochastic methods, unsupervised learning, self-supervised learning) 및 딥러닝 분야 (adversarial networks, deep autoencoderes, generative models, optimization for deep networks, recurrent networks, supervised deep networks)를 중심으로 다루며, 각 학기마다 중점 내용이 조정된다.



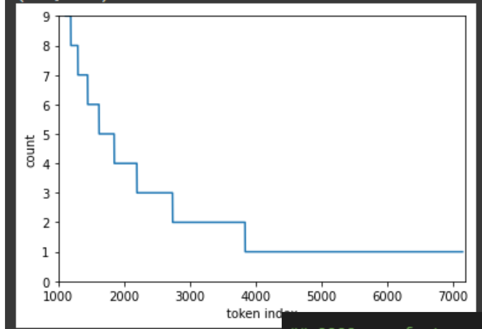
{'학기', 'unsupervised', 'self', 'adversarial', '선별', 'stochastic', 'deep', '중점', 'transfer', '중심', 'supervised', 'methods', 'models', 'active', 'clustering', '프로젝트', '최신', '국제', '제공', '인공지능', '연구', 'multitask', '발전', 'recurrent', '알고리즘', 'generative', '최적화', '기술', '진행', '논문', '발표', '가능', 'classification', '주제', '속도', '러닝', '논의', 'autoencoder', '조정', 'learning', '수준', 'networks'}

01. Content-Based Recommendation

TF-IDF Calculation - Python code

```python
# plot distribution of counts to find how many tokens to use for vectorizing

plt.plot(sorted(processor.wordcount.values(), reverse=True))
plt.xlabel('token index')
plt.ylabel('count')
plt.xlim([1000,7200])
plt.ylim([0, 9])
```

```
(0.0, 9.0)
```




```python
## 2200 max_features to include tokens that occurred at least 4 times (e.g. iot)
cv = CountVectorizer(max_features=2200, tokenizer=processor.get_nouns_and_sl)
tdm = cv.fit_transform(abstr['abstract'])

tfidf = TfidfTransformer()
tdm2 = tfidf.fit_transform(tdm)

words = cv.get_feature_names_out()
```

```python
# 벡터라이저 등 저장
import joblib
joblib.dump(tfidf, 'tfidf_transformer_tok2200.pkl')
joblib.dump(cv, 'count_vectorizer_tok2200.pkl')
joblib.dump(tdm, 'count_tdm.pkl')
joblib.dump(tdm2, 'tfidf_tdm.pkl')
joblib.dump(abstr['cid'].values, 'cid_list.pkl')
```

```
['cid_list.pkl']
```

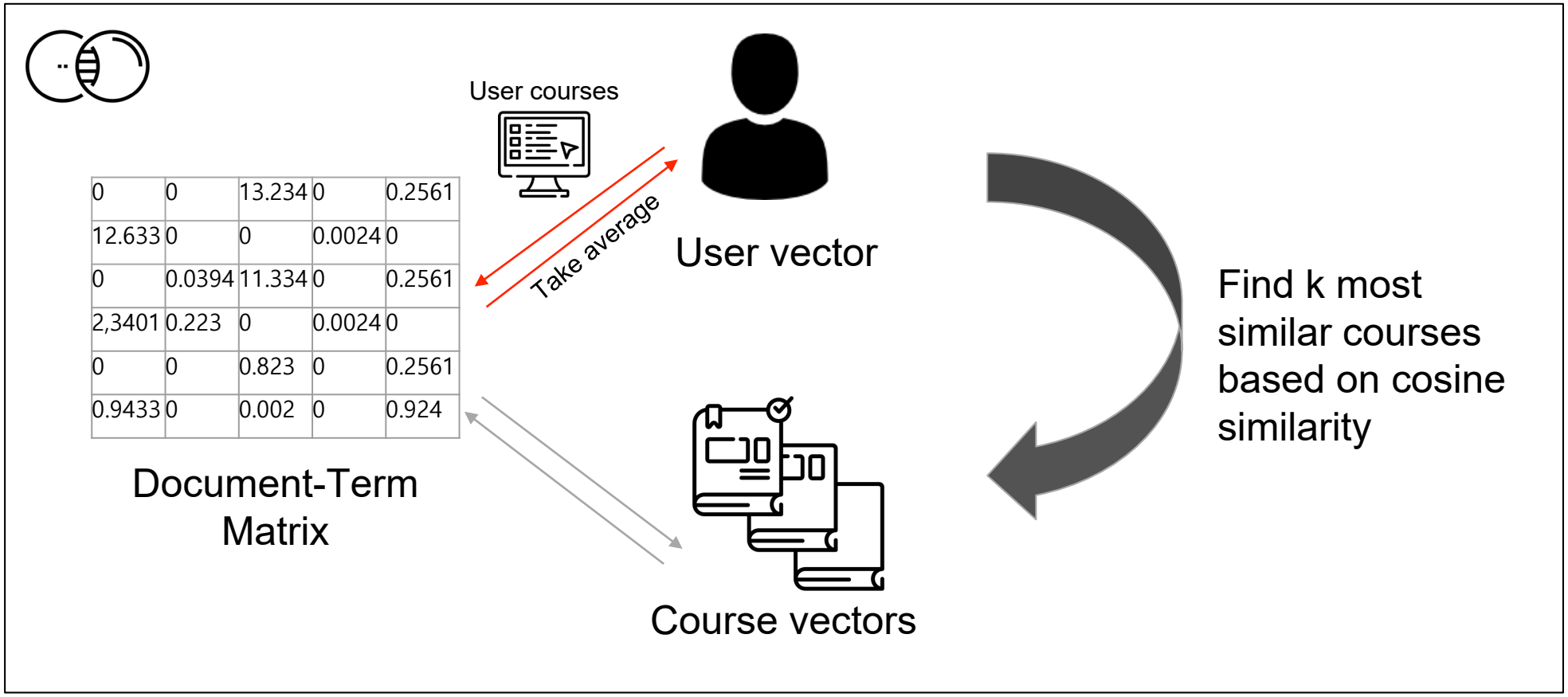
Result

{'학기', 'unsupervised', 'self', 'adversarial', '선별', 'stochastic', 'deep', '중점', 'transfer', '중심', 'supervised', 'methods', 'models', 'active', 'clustering', '프로젝트', '최신', '국제', '제공', '인공지능', '연구', 'multitask', '발전', 'recurrent', '알고리즘', 'generative', '최적화', '기술', '진행', '논문', '발표', '가능', 'classification', '주제', '속도', '러닝', '논의', 'autoencoder', '조정', 'learning', '수준', 'networks'}

```
array([[0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       ...,
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.132, 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.243, 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , ..., 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ]])
```

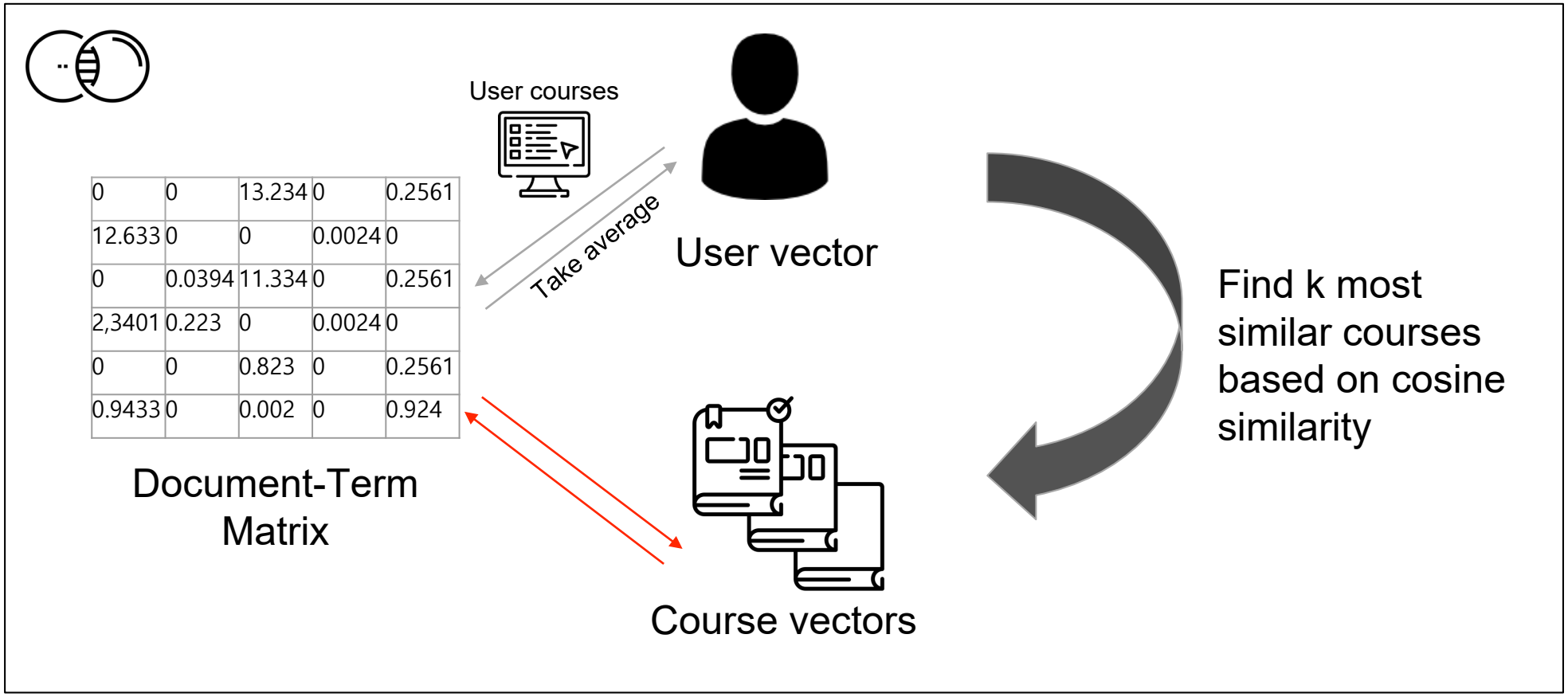
01. Content-Based Recommendation

Inference



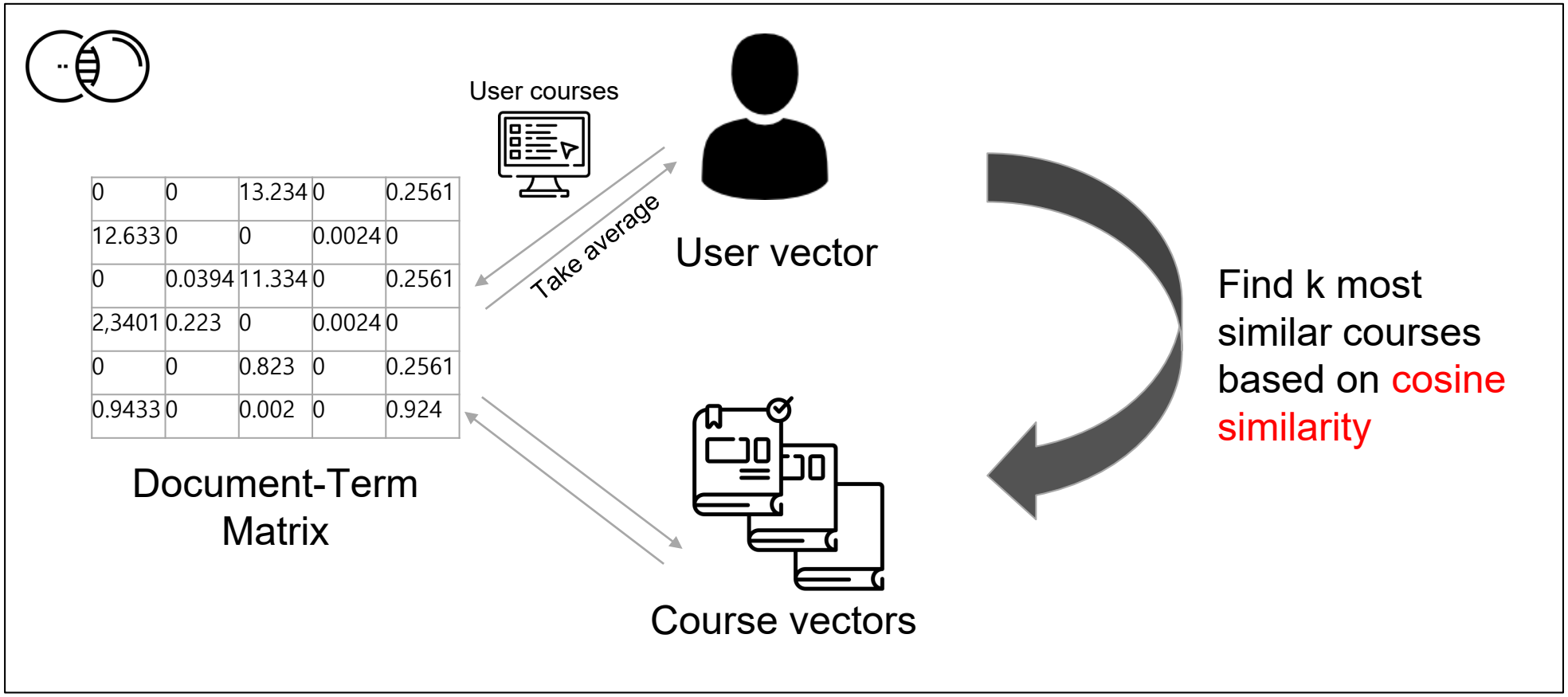
01. Content-Based Recommendation

Inference

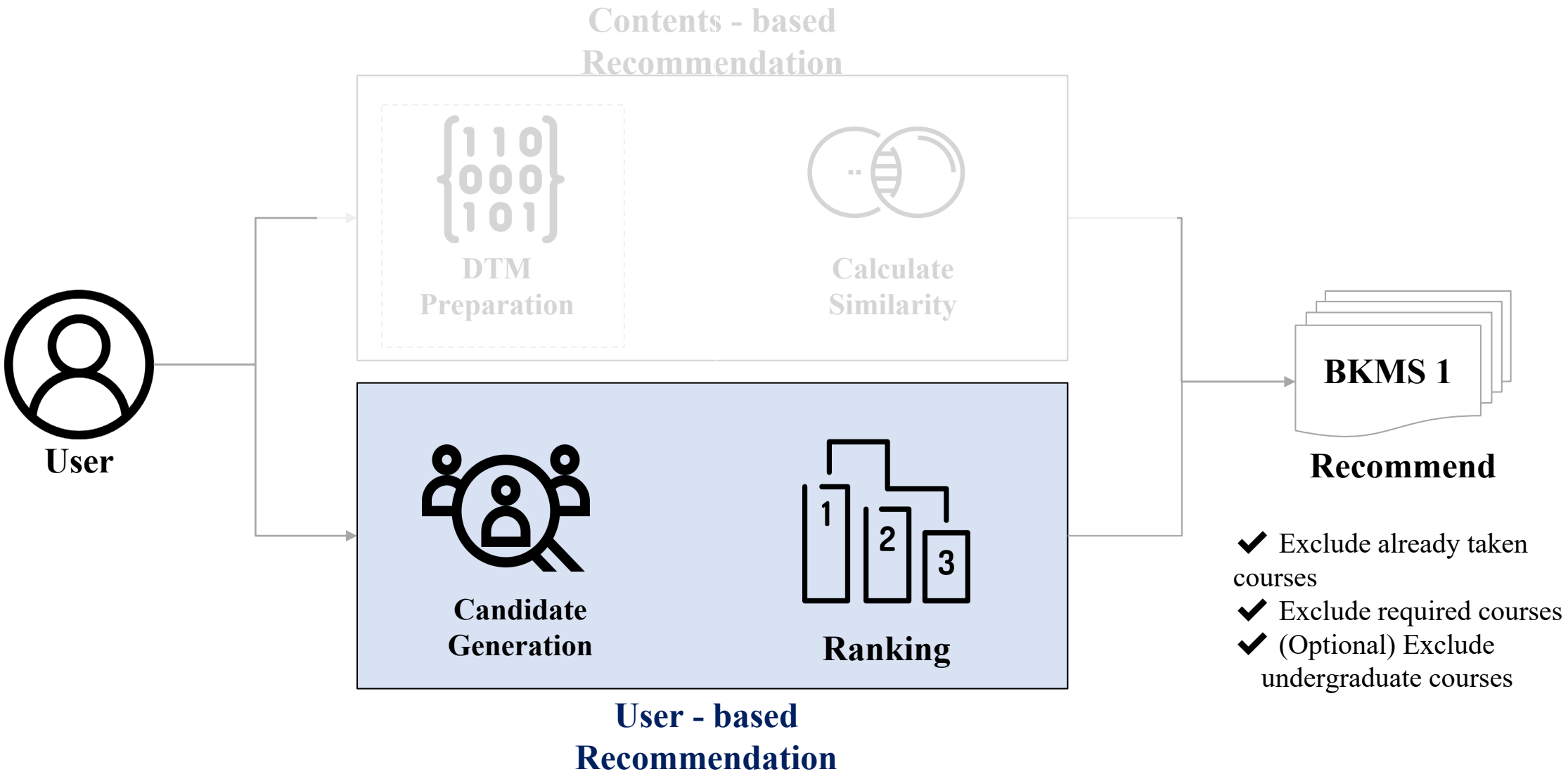


01. Content-Based Recommendation

Inference

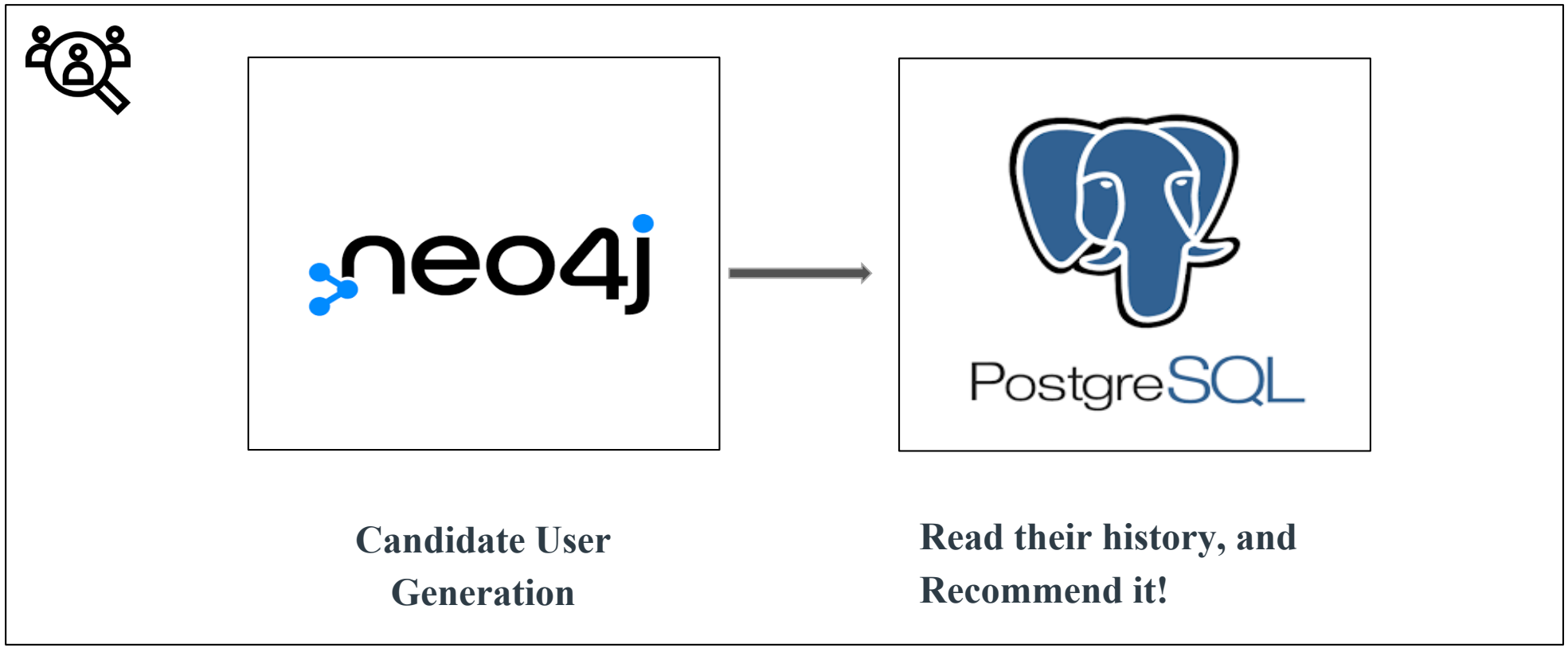


02. User-Based Recommendation



02. User-Based Recommendation

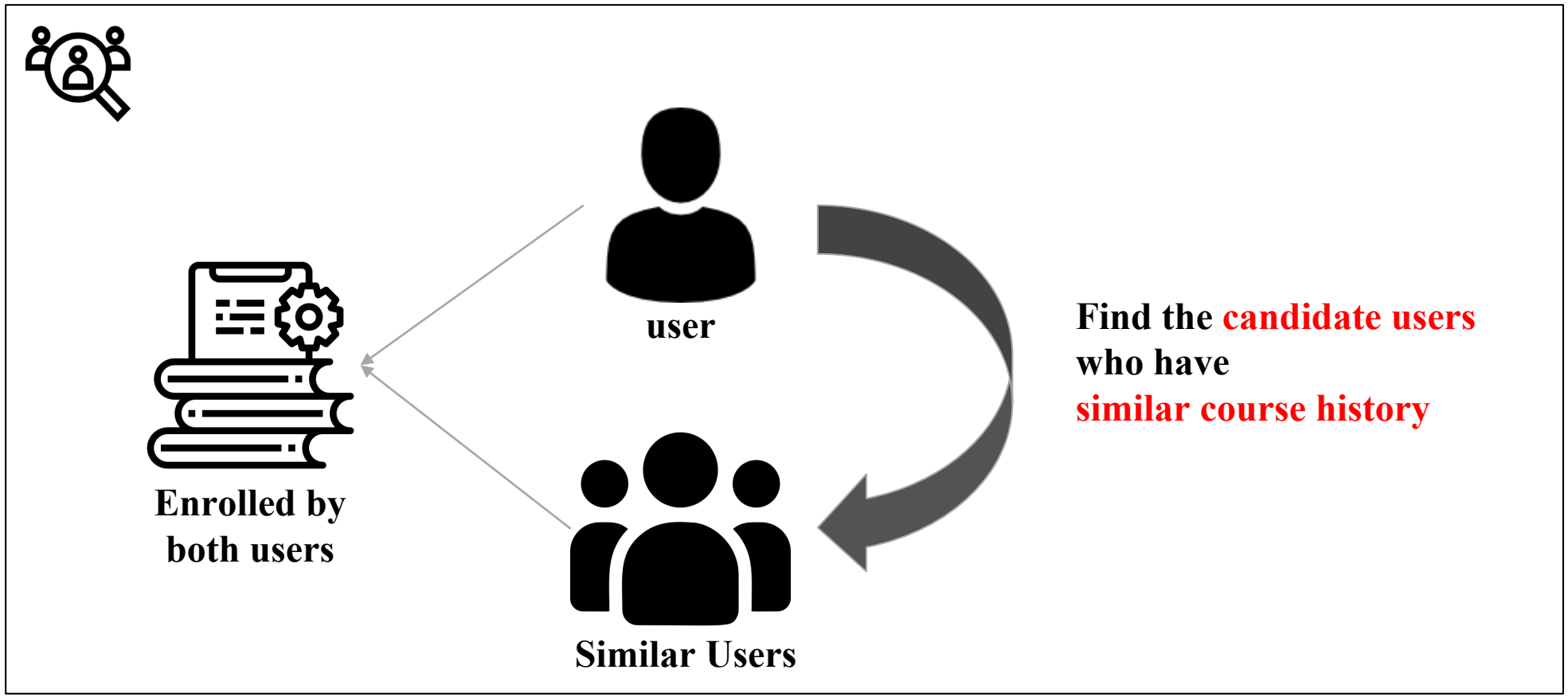
Database Linkage



The user-based recommendation logic reads the user's information from the DB server and finds users with similar course histories. Their courses are then created as a recommendation list.

02. User-Based Recommendation

Detailed Mechanism



02. User-Based Recommendation

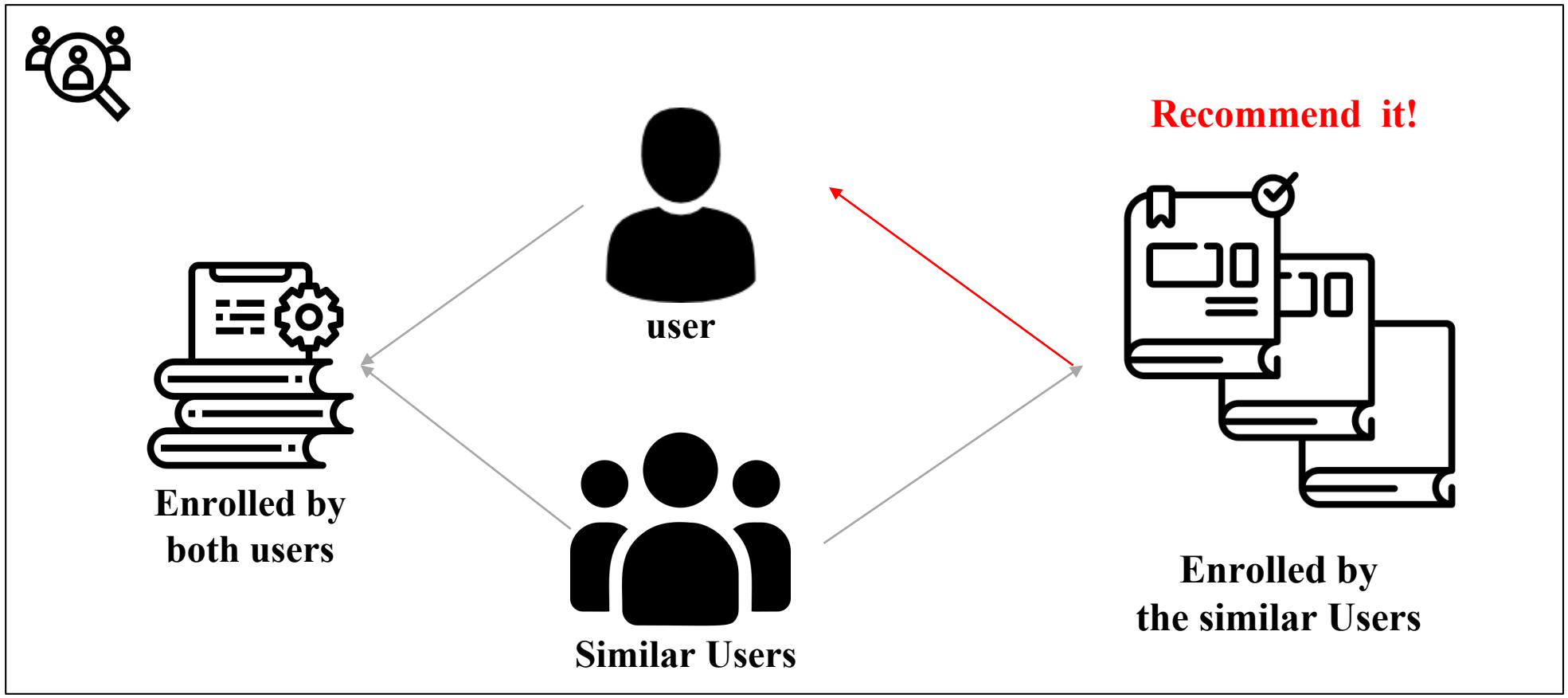
Code Detail

```
class Candidate_user_generate:

  def __init__(self):
      # DB Connection Information
      self.connection_info = "host=147.47.200.145 dbname=teamdb5 user=team5 password=snugraduate port=34543" # PostgreSQL

  def connect_SQL(self): # DB에서 데이터를 불러옴
    # PostgreSQL 연결
    conn = psycopg2.connect(self.connection_info)

    try:
        # 데이블을 Pandas.Dataframe으로 추출 (user table, course table, enrollment table)
        self.user = pd.read_sql('SELECT id, username FROM auth_user ',conn)   # user table
        self.course = pd.read_sql('SELECT cid_int, cid, cname FROM check_course',conn)  # check_course table
    ########################################################################################## 12.02 max_up id 쿼리
        self.enroll = pd.read_sql('SELECT cid_int, user_id, cid from rec_enrollment',
                                  'where up_id in (select max(re.up_id) from rec_enrollment re group by re.user_id)',conn
    ##########################################################################################
        self.enroll_idx = self.enroll.set_index('user_id')

    except psycopg2.Error as e:
        # 데이터베이스 에러 처리
```

PostgreSQL and Neo4j connections were implemented as a method function within the same class. When loading the Enrollment table from the DB, the up_id variable was set in consideration of the case where each user uploads an Excel several times. Each time an Excel is uploaded, an up_id is recorded, and only the row with the largest up_id for each user id is loaded into the enrollment table.

02. User-Based Recommendation

Detailed Mechanism



02. User-Based Recommendation

Code Detail

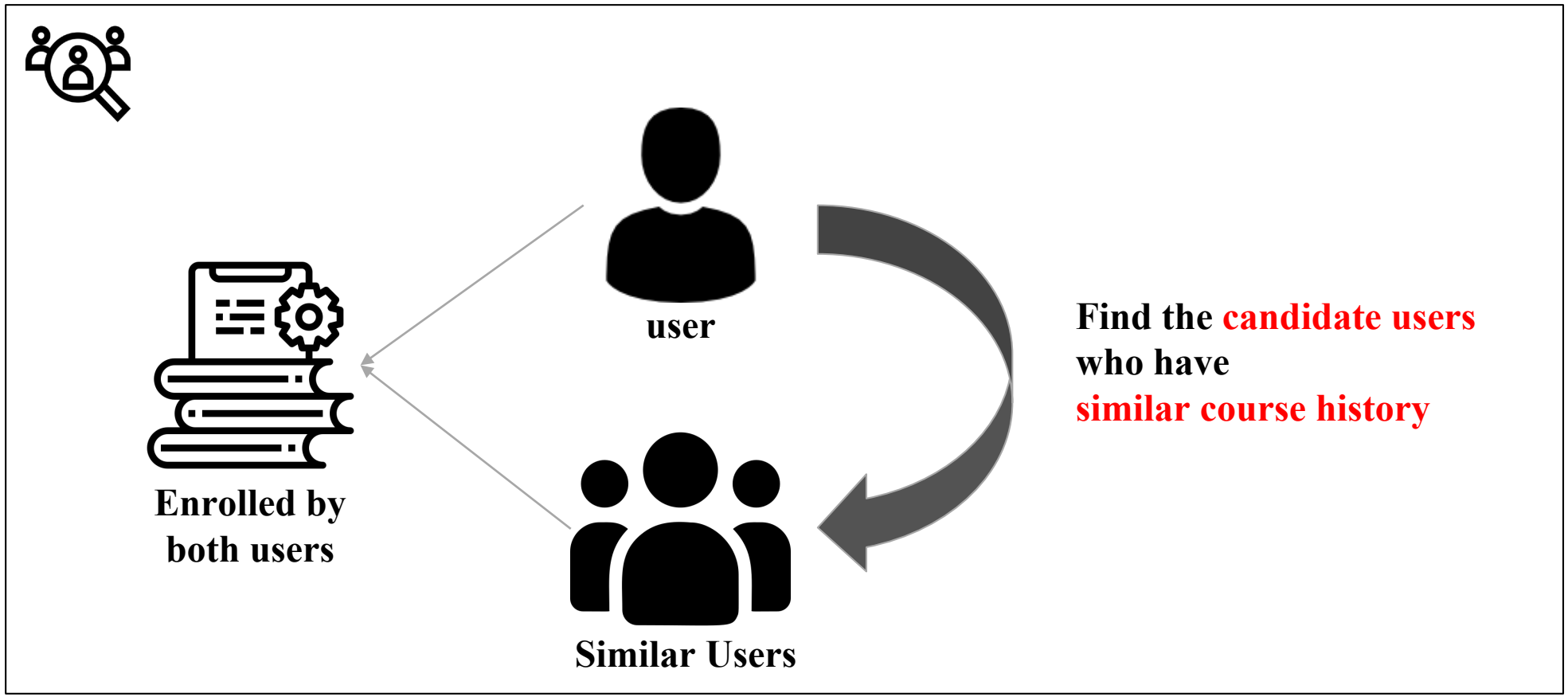
```
def connect_Neo4j(self, id, cnt): # 조건 쿼리, input : id - User ID / cnt : 몇명을 리턴할것인가? (default = 3)
    # DB Connection Information
    dbname = "teamdb5"
    uri_param = "bolt://147.47.200.145:37687"
    user_param = "team5"
    pwd_param = "snugraduate"
    apoc_string = 'CALL apoc.load.jdbc("jdbc:postgresql://localhost:34543/teamdb5?user=team5&password=snugraduate",'

    # Neo4j 연결
    conn = Neo4jConnection(uri=uri_param, user=user_param, pwd=pwd_param)
    # Cypher 쿼리 입력
    cypher = (apoc_string + '"SELECT cid_int, user_id,cid,up_id,gpa from rec_enrollment'
              'where up_id in (select max(re.up_id) from rec_enrollment re group by re.user_id)") YIELD row '
              'MERGE (sd:Student {id: row.user_id}) '                    # User Node 형성
              'MERGE (c:Course {cid_int: toInteger(row.cid_int)}) '  # 과목 Node 형성
              'MERGE (sd)-[e:Enrollment {gpa : row.gpa}]->(c) '  # 수강 Flow 형성
              'with sd,c '
              'Match (sd{id:'+ str(id) + '})-->(c)<--(sd2) '         # User과 들은 과목을 들은 sd2를 찾아라
              'return distinct sd2.id, count(*) as cnt '              # sd2의 user id 반환
              'order by cnt desc '                                    # 겹치는 과목의 순서가 많은 사람부터
              'Limit '+ str(cnt)                                      # cnt명 반환
              )

    # Cypher 쿼리 실행 후 결과를 print
    response = conn.query(cypher, db=dbname)
```

The Neo4j method allows two inputs to be received. One is the ID of the current user, and the other is the parameter 'cnt' about how many similar users to find. Currently, the number of users is not large, so the default cnt is set to 3. The input variable was implemented to be reflected in the Cyper query.

In addition, Neo4j was implemented to load only the latest data for each user in the Enrollment table. Before merging nodes, the function was implemented in SQL statements, which is the same as the part where the Enrollment table is retrieved from PostgresSQL.

02. User-Based Recommendation

Detailed Mechanism



02. User-Based Recommendation

Code Detail

```
# Cypher 쿼리 실행 후 결과를 print
response = conn.query(cypher, db=dbname)

# 연결 종료 필수!
conn.close()

if response == None:
  return list(self.course['cid_int'])

person = pd.DataFrame([dict(record) for record in response])
if person.shape[0] < cnt :  # 특정 인원수가 안되면 그냥 전체 과목을 리턴
  return list(self.course['cid_int'])


cid_list = []
for idx,i in person['sd2.id'].iteritems():
  cid_list+=list(enroll_idx['cid_int'].loc[i]) # 비슷한 과목을 들은 유저들의 과목을 리스트화
cid_set = set(cid_list) # 중복 제거
cid_list = list(cid_set) # 중복 제거

for value in list(enroll_idx['cid_int'].loc[id]): # list에서 user가 들은 과목 제거
  cid_list.remove(value)
return cid_list
```

The results reflected from the query were assigned to the variable response. The response was processed in the form of a Pandas DataFrame, and then the output was returned in the form of a List so that the function result could be used as Python outside the method. This is the list of subjects that similar users have heard.

02. User-Based Recommendation

Step 1. Find the Candidate user using cypher query

```
SELECT cid_int, user_id, cid, up_id, gpa from rec_enrollment
where up_id IN (select max(re.up_id) from rec_enrollment re group by re.user_id)") YIELD
row
MERGE (sd:Student {id: row.user_id, up_id: row.up_id})
MERGE (c:Course {cid: row.cid})
MERGE (sd)-[e:Enrollment {gpa : row.gpa}]->(c)
with sd,c
Match (sd{id:'+ str(id) + '})-->(c)<--(sd2)
return distinct sd2.id, count(*) as cnt
order by cnt desc
'Limit '+ str(cnt)
```

Get the Users' latest course history from the enrollment table in DB

02. User-Based Recommendation

Step 1. Find the Candidate user using cypher query

```
SELECT cid_int, user_id, cid, up_id, gpa from rec_enrollment
where up_id IN (select max(re.up_id) from rec_enrollment re group by
row
MERGE (sd:Student {id: row.user_id, up_id: row.up_id})
MERGE (c:Course {cid: row.cid})
MERGE (sd)-[e:Enrollment {gpa : row.gpa}]->(c)
with sd,c
Match (sd{id:'+ str(id) + '})-->(c)<--(sd2)
return distinct sd2.id, count(*) as cnt
order by cnt desc
'Limit '+ str(cnt)
```

Find the Candidates!

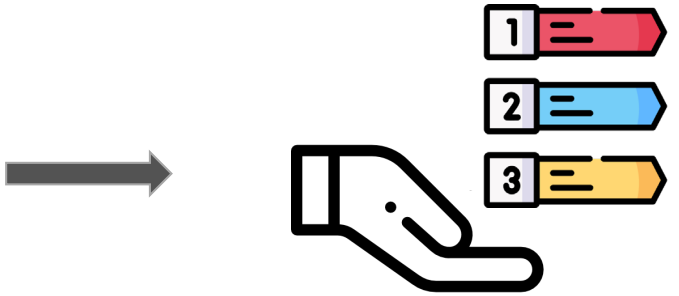
02. User-Based Recommendation

Step 2. Make the list as a pandas dataframe, and add up the courses into a recommend list.

```
person = pd.DataFrame([dict(record) for record in response])
person
```

	sd.id	sd.up_id	c.cid
0	3	11	M3239.005300
1	3	11	M3239.005700
2	3	11	M3239.003300
3	3	11	M3239.005400
4	3	11	M3239.005000
...	...	...	...
81	1	3	M3239.005500
82	6	6	M3239.003000
83	6	6	M3239.003100
84	6	6	M3239.002300
85	6	6	M3239.000300

86 rows × 3 columns



Recommend !

01. User-based recommendations

Exception Handling 1

[Case 1 : No similar User]

```
# 예외 처리 : 유사한 사용자가 없으면 전체 과목 리턴
if response == None:
#########################################################################################
  result_list = list(self.course['cid']) # 전체 과목 id list
  for value in list(self.enroll_idx['cid'].loc[id]): # User가 들은 과목 제외
    if value in result_list: # 조건추가 : user가 들은 과목중에 cid_list에 없는 과목이 있을 수도 있으므로 체크 필요
      result_list.remove(value)
  return result_list
#########################################################################################
```

[Case 2 : Fewer similar users

```
person = pd.DataFrame([dict(record) for record in response])
# print(person) # sd id 확인용
########################################################################################## 12.02 모집 인원수가 적
# 예외 처리 : 특정 인원수가 안되면 그냥 전체 과목을 리턴
if person.shape[0] < cnt :
#########################################################################################
  result_list = list(self.course['cid']) # 전체 과목 id list
  for value in list(self.enroll_idx['cid'].loc[id]): # # 조건추가 : DS 필수과목 제외
    if value in result_list: # 조건추가 : user가 들은 과목중에 cid_list에 없는 과목이 있을 수도 있으므로 체크 필요
      result_list.remove(value)
  return result_list
#########################################################################################
#########################################################################################
```

The Response variable serves to contain data from similar users. However, sometimes there are no similar users, or because of insufficient user data, similar users do not meet the cnt we set. In order to solve the Cold Start Problem for these two exception cases, the entire subject list, not the subject list of similar users, was returned.

In this case, **Content-based recommendation module** acts predominantly.

01. User-based recommendations

Exception Handling 2

```
            cid_fname = 'cid_first.pkl' ):
self.tfidf = joblib.load(os.path.join(file_dir, tfidf_fname))
self.cid_list = joblib.load(os.path.join(file_dir, cid_fname))
#########################################################################################
self.basecourse_list = [   # DS 필수과목 리스트
'M3239.000100',
'M3239.000200',
'M3239.000300',
'M3239.003800',
'M3239.005000',
'M3239.005100',
'M3239.005300',
'M3239.005400',
'M3239.005500',
'M3239.005700',
'M3239.006700',
'M3239.002800',
'M3239.003000',
'M3239.000400',
'M3239.000500',
'M3239.000900',
'M3239.005800',
'M3239.003300'
]
#########################################################################################
```

When recommended, the necessity for mandatory subjects in GSDS was not felt, so the lists were assigned as variables in the class and treated as exceptions.

01. User-based recommendations

Exception Handling 3

```
try:
  generated_list=generator.get_person(user_id, cnt=cnt) # input = user id , output = cid_int list
except psycopg2.Error as e:
  try:
    generated_list=generator.get_person(user_id, cnt=cnt) # input = user id  # 커넥팅 에러 방지차원으로 한번 더 실행
  except psycopg2.Error as e:
    try:
      generated_list=generator.get_person(user_id, cnt=cnt) # input = user id  # 커넥팅 에러 방지차원으로 한번 더 실행
    except psycopg2.Error as e:
      # 데이터베이스 에러 처리
      print("DB error: ", e)
```

Sometimes, connection failures occurred due to unstable interworking with PostgreSQL and Neo4j. Although it was an error that often occurred in practical classes and assignments, separate error processing was conducted because it was a part that could not be overlooked in the actual service.

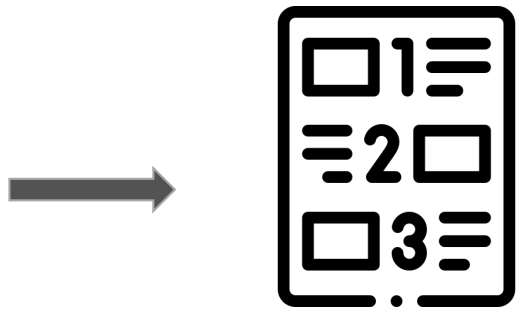
02. User-Based Recommendation

But, it sometimes recommends a course that is too irrelevant.
So, we put a filter on it based on user's course history

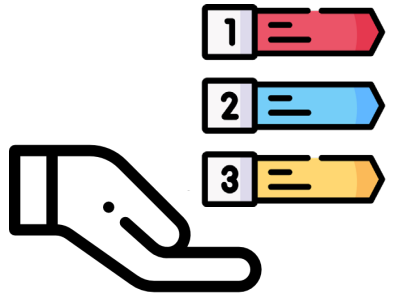
```
person = pd.DataFrame([dict(record) for record in response])
person
```

	sd.id	sd.up_id	c.cid
0	3	11	M3239.005300
1	3	11	M3239.005700
2	3	11	M3239.003300
3	3	11	M3239.005400
4	3	11	M3239.005000
...	...	...	...
81	1	3	M3239.005500
82	6	6	M3239.003000
83	6	6	M3239.003100
84	6	6	M3239.002300
85	6	6	M3239.000300

86 rows × 3 columns

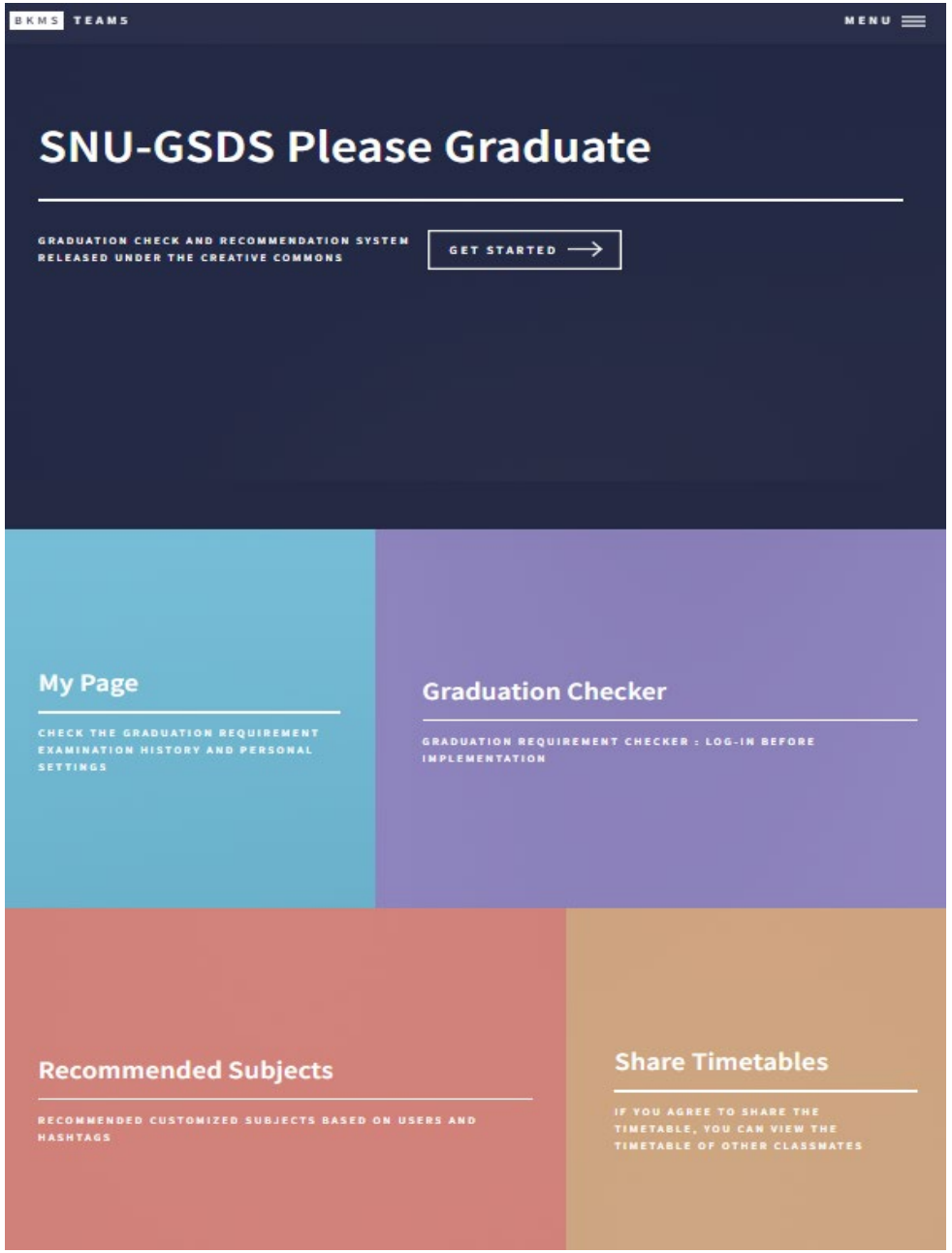


Rank the list based on user's course history



Recommend !

Demonstration



Website Link (Available until the end of Dec 2022)

https://bkms-team5-6nwxvl4rka-du.a.run.app/

Demonstration Recorded Log

https://drive.google.com/file/d/18ujgZka8nQSxjWnjvLCM-4dQ5GhMvmeP/view?usp=sharing

Summary

Merits
Limitations & Possible extensions
Team member contribution

01. Reflections : Technical challenges, Lessons learned

Technical challenges

- Insufficient user course history data
- User might upload their course history multiple times. Need to distinguish and use only the latest uploads
- In the Google OAuth2 environment, you can only get information such as the student's name, department, and snu email address. Therefore, there are many inputs that students receive directly, and we want to synchronize them with the snu account as much as possible. You should be asked to receive information through connection with the school information headquarters or expand the user model to enter additional information such as class number and process when logging in to Google for the first time.

Lessons learned

- Determining ERD together is important. Tables used by different functions might share common columns. In order to reduce redundancy and potential errors, it is important for different parts to come together and design normalized (or sufficiently normalized) tables.
- SQL also helps you import the tables through Cyper query under certain conditions.
- Also able to understand what features SQL and Python each have and where they will be available. Above all, in that both front-end and back-end were implemented, it was a good opportunity to indirectly experience the process of technically implementing actual services.

02. Limitations of the project

[Graduation Condition Check]

- Information that cannot be interpreted on the Course Completion Info (Report Card) should be left to the user input.
- There are a few exceptions: in the second semester of 2021, the '데이터사이언스 특강' designated as a mandatory subject for some students.

[Recommendation]

- We didn’t use collaborative filtering which is used to get user-based recommendations due to lack of user data. Without enough user data, it couldn’t get meaningful results which is called ‘Cold-Start Problem’. Instead, we try to find similar users to overcome the cold-start problem.
- **Students in their first year mostly take required courses** - hard to know their personal interests.
- During preprocessing texts from abstracts, we made our own synonym and stop words dictionary, but the list might not cover all cases.

02. Limitations of the project

[Front and Backend]

- Django provides a guide to distribute using a WAS called ‘gunicorn’, rather than using a version with a ‘runserver’ applied to the debug=True option. This is because debug=True is a mode for developers, not a mode for users. Also, it cannot handle a large amount of traffic when using a runserver.

- Currently, it is not clear where files are saved when uploaded. Google has created a guide to send and receive static files (CSS, javascript, Excel, etc.) using Cloud Storage, but it is not used in the current version. It is future work to apply this part by changing the setting of Django.

- DB connection information was exposed to Django code(settings.py와 views.py), and Django’s secret key was also distributed in an exposed state. Locally, .env files can be used to designate them as environmental variables, and when the web server is deployed, they can be hidden by applying Google Cloud's Secret Manager.

03. Future works and extensions

User personalized course recommendation : Hash-tags or Custom Keywords

- Allowing user to input keywords that they are interested in so the system can incorporate these explicit interests to recommendations

More data to enable rich analytics

- Achieve Apply collaborative filtering (If sufficient data is gathered)
- Add more User nodes to give more diversity to User-based recommendation system

Continuous updates of the Course info and Graduation Requirement

- Amendments and additional exceptions should be directly modified and supplemented by the developer. To this end, it is possible to consistently write a development log and disclose the correction log to the user.

Team Members

Chaeyun Kim

- Leader of Team5
- GSDS Course Data preprocessing
- Proposal & Final presentation speaker
- UI Design and Front -end engineering

Hyeryung Son

- Sub Leader of Team5
- Final presentation speaker
- Crawling the course information from sugang.snu.ac.kr
- Content-Based Recommendation

Younghun Kim

- Crawling the course information from sugang.snu.ac.kr
- User-Based Recommendation

Minwoo Park

- GSDS Course Data preprocessing
- Graduation Requirements Checking Algorithm Development

Jihoo Park

- Django Backend
- Javascript Frontend Development
- Made CI/CD Pipeline (by google cloud)

Thank you

Q&A