SKKULAR : Scholarship Recommendation Application Using BERT, Django, and REACT NATIVE

Kyujin Kim[2019310741], Byeongjun Kang[2017314035], Jinah Park[2020314944], Ijun Jang[2019313491], and Jaehyun Joo[2017314245]

Sungkyunkwan University, 2066,
Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, Republic of Korea
{kyujin0115,freudbj2,pja9362,ijun0824,xhzjtm117}@g.skku.edu

Abstract. Since lots of students do not know or search for scholarship information, the number of scholarship applications may not be met. Therefore, The SKKULAR app allows students at Sungkyunkwan University to see scholarships in schools and can also search for scholarships. Also, the user's personal information and the appropriate scholarship are recommended, and a notice is sent. The development process can be divided into three parts: AI, backend, and frontend. First of all, the AI part collected scholarship information from the school and extracted the desired information from the scholarship announcement using the BERT machine reading model. The backend used Django and MySQL to store data and manage users' personal information. It also used AWS to build servers and RESTAPI to transmit appropriate APIs. Finally, the front end used the Expo CLI of React-Native to create pages and build notification functions. In particular, SKKULAR application will provide useful information about scholarship to students of Sungkyunkwan University by automatically analyzing scholarships and filtering scholarships suitable for students.

Keywords: Scholarship · BERT · Django · Reactive Native

1 Introduction

The SKKULAR app provides scholarship information, recommends scholarships that can be applied for, therfore helps many students to easily know and apply for scholarships. The majority of students are unable to apply without knowing the scholarship information, and many scholarships are waiting students to apply, but the number of applicants is not sufficient fully. In fact, many newspapers say that students cannot apply because they do not know about scholarships. In order for students to apply for a scholarship, they must search each scholarship by the school's announcement and the department's announcement every day or every week to confirm their eligibility. Students are too tired to search before applying for a scholarship, and in the end, they often fail to apply for a scholarship.

In order to solve this problem, apps such as "Every time" and "Uniview" actually created a function to collect public announcements, and Dream Phone collected various scholarships at a glance. However, such apps only provide the basic information related to scholarships to students, and the problem is that they cannot recommend scholarships that students can apply for directly. Therefore, it is necessary to have an app that informs students about scholarship information and selects scholarships that can be applied for.

The SKKULAR app allows students of Sungkyunkwan University 1) to take a look at the scholarships listed in the school and department announcements, 2) to search for scholarships listed, and 3) set up scholarships of interest. Also, if there is a scholarship that the user can apply for using the user's personal information, 4) they recommend it and 5) send a notification. The SKKULAR app would recommend students apply for scholarships and provides direct convenience. Furthermore, it would increase the number of scholarship applications, which will help scholarship support organizations introduce more scholarships to SungKyunKwan university.

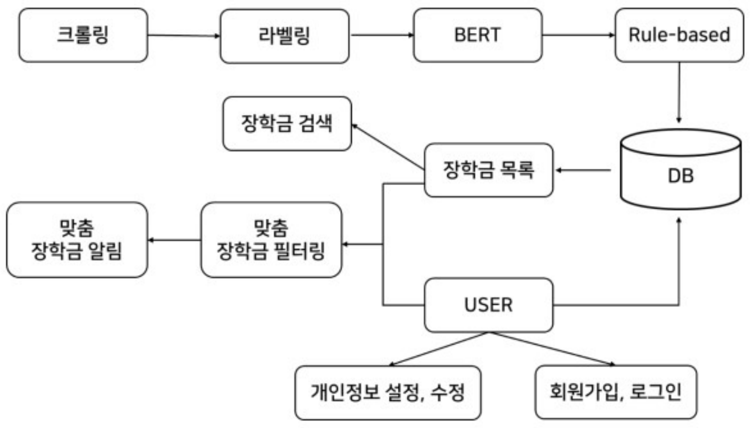
The development process of the SKKULAR app is divided into three parts: AI, backend, and frontend. In the AI part, we crawled the contents of announcements from schools and colleges. After that, the BERT model is used to find answers to each question from the scholarship. The result is used to collect scholarship application qualifications such as grades, departments, and regions that are required for students to apply. The EM score of the learned BERT model is 76 %, and the F1 score is 78 %. In the backend part, Django was used as a web framework to communicate with the database, and Rest FrameWork was used to configure the RESTFUL API. The server was also distributed to AWS, and the API test proceeded through POSTMAN. Finally, the front-end uses React Native, a low degree of freedom but easy to initialize and relatively convenient to develop, creating a page provided by the SKKULAR app, as well as scholarship filtering and notification.

Consequently, The final features are: 1) Register as a member 2) Login 3) Scholarship List 4) Scholarship Search 5) Scholarship Details Page - You can check your scholarship eligibility is correct based on the information you filled out on registration page 6) Quick Links - You can see scholarship directly 7) Interesting Scholarship Tab 8) Customized Scholarship Tab 9) Notice that the announcement you can apply is registered on the school website.

2 Design

2.1 Overall Architecture

The overall development process is shown in the following diagram. First, in the AI process, we crawled, labeled, BERT learning, and rule-based processing of scholarship announcements. Then we saved each scholarship support qualification obtained through BERT learning in the database. After that, at the front end and back end, it realized the function of displaying and searching for scholarships stored in the list and saving scholarships of interest. It also implemented a filtering function by checking whether user information matches scholarship qualification and an alarm when the new scholarship matches user information. Furthermore, the user's registration and personal information correction functions are also realized. As explained in the Introduction, the development process can be divided into AI, Backend, and Frontend, but it can be seen that the Backend and Frontend implemented the same functionality while communicating with APIs. Therefore, the future development process will be explained in two components: AI and the App.



2.2 Core Skill

2.2.1 AI: BERT Q&A model

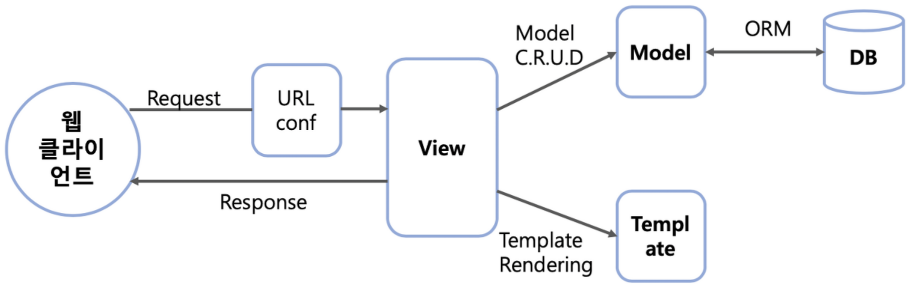
BERT corresponds to a natural language understanding model in which the encoder of the transformer is stacked in a total of 12 layers. Historically, language models could only read text input sequentially (either left-to-right or right-to-left) but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality. If pre-trained Bert model existed, it is better to train new model by fine-tuning(changing fraction of the weights rather than training all weights in the model) instead of doing pre-training because of resources. BERT can be used in various tasks by fine-tuning such as Question Answering, Abstract summarization, sentence prediction and etc.

2.2.2 Frontend : React Native

We used React Native, an open-source mobile application framework created by Facebook to develop frontend. When developing applications, it is common to develop Android as Kotlin and Java, and IOS as Objective-C and Swift. However, in order to develop into each language, developers must understand both the language and the operating system. The development process is more complicated and more expensive to maintain. Because of these problems, we used React Native. React Native is a cross-platform that can be developed with JavaScript, and can simultaneously develop IOS and Android mobile apps in one programming language.

2.2.3 Backend

We designed the server with following architecture.
1. Using Django to implement the functionality of our applications.
2. Using mysql(RDB) for storing data.
3. AWS EC2(Electric Compute) for global public access[2].
4. Django Rest Framework to build API



Django communicates with the server and client in the form shown in the figure above. When a client connects to a specific URL, Django searches for a function mapped to the URL in the view and performs CRUD and rendering work.If the structure of the model, template, and view corresponding to the MTV format is understood, the developer can easily provide planned items to the client. Let's take a quick look at the roles of models, templates, and views.

Model means data stored in DB. A model is defined as a class, and one class is one DB Table. Without direct query writing, using Django's model function, you can configure the DB with only Python code. SKKULAR used MYSQL, a relational database, and performed DB CRUD by linking MYSQL with Django. Template means the screen shown to the user. Django processes the logic in the view and then renders the html file along with the context. The html file at this time is called a template. However, since SKKULAR uses REST API for the convenience of development, the template function was not used separately.

Django supports its own Django Template syntax, and thanks to this syntax, you can use data received as context within an html file. View responds by rendering the result as a template by performing appropriate logic according to the request. However, it doesn't always render the template, and in some cases it only sends and receives data from the backend.

When exchanging data between server and client, they exchange data using HTTP protocol. REST API (Representational State Transfer) is a design guide designed to explicitly exchange data between client and server. Our team was able to develop the API easily using Django REST FRAMEWORK. The development sequence is as follows. First, a model is constructed, and an object is converted into a series of data through a serializer.The serializer is responsible for converting data inside Django into json/xml files. After that, configure logic through view and apply the method of mapping work through URL. API testing was done through Postman, an API platform for designing, building, testing, and iterating APIs.

2.3 Reasoning

2.3.1 AI

The important task of this project is to automatically write data into the Database after extracting key information (ex. eligibility for application (income quantile, grade, credit), deadline and etc) even if any scholarship posts are given as input. This is because such formalized data is required to recommend scholarships that meet the conditions of users. At this time, the NLP model that was used as a core in the service is BERT model, especially BERT QA model. It is expected that the desired information can be extracted by asking questions that can extract key information from the scholarship post. For example, BERT QA model can extract grade requirement of posts by giving a question like "What grade can I apply for?". For this question, BERT model will return a answer such as "over 3.00".

2.3.2 Frontend

There are two ways we can set up a React Native for application.

The first is React Native CLI. React Native CLI is a method of developing only using React Native without using any tools. It has the ability to control what developers want or all elements, and the ability to use many libraries. To summarize the pros and cons, the degree of freedom is high, but the initial setup is difficult and the development process is relatively complicated.

The second is Expo CLI. Expo is a framework and platform that helps facilitate the development of react native. There are many convenient utility features that simplify application development, basic settings, and the native file is hidden and automatically managed. To summarize the pros and cons, the degree of freedom is low, but the initial setting is simple and relatively convenient to develop. In addition, it is a great advantage that any device such as Android or IOS can conveniently operate the simulator without any complicated settings.

We chose Expo among the two methods. Even if only the functions provided by Expo are used, it is not a big problem in implementing the SKKULAR, and it is the first time for all team members to develop the react native, we developed using Expo, which is relatively convenient and fast.

2.3.3 Backend

Crawling, BERT learning, and the process of transmitting the collected results to the DB were all done in Python. The Django library also uses the Python language, and considering compatibility with other tasks, the Django library was used. In addition, the convenience of API development through Django Rest Framework and compatibility with MYSQL, the database used by SKKULAR, were also considered.

Also, the reasons for using Mysql and EC2 for the database and server are as follows. Mysql is a relational database management system, which requires a clearly defined schema and guarantees data integrity. Skkular decided that what would be stored in dB was uncomplicated and had a lot of room for change. Therefore, we chose Mysql because we believe that a clear schema is important to users and their data. The reason why EC2 was used as the deployment server was that it was possible to configure an independent server. This is because it provides stability and security, and can provide servers optimized for various purposes.

2.4 Challenges

2.4.1 AI

The most important challenge in AI part is "how to define answer that doesn't exist in posts". In KorQuad Dataset, it marked "no answer" as any random text in the posts. As we should print "no answer" into a unified answer, we made datasets that mark "no answer" as "." which appears frequently in the sentences.

2.4.2 Frontend

There are some challenges that we have faced in implementing the frontend.

The first part is login. We initially managed the user information using Async-storage provided by React Native. We temporarily stored sign-up information in asyncstorage, and when logging in, the stored information was retrieved and used through 'getItem'. Most of cases worked normally, but there were cases where they did not work normally due to async-await delay problems, such as not being able to log in with valid user information as soon as the login screen was loaded. Most of cases worked normally, but there were cases where they did not work normally due to async-await delay problems, such as not being able to log in with valid user information as soon as the login screen was loaded. The problem could be solved using the Provider component provided by React. The Provider component accepts a value prop to be passed to consuming components that are descendants of this Provider. Using this, it can be implemented to manage user information without using asyncstorage and to enable login function without exception situations.

The second challenge is related to updating real-time scholarship information. In order to provide a customized scholarship notification function, which is a key function of SKKULAR, it is necessary to receive real-time updated scholarship information. When using useEffect only, unlike the features that we wanted to implement, the user had to reload screen to get updated information. To solve this problem, we used setInterval, useInterval functions. These functions used to fetch the scholarship information API every specific cycle. This allows users to automatically receive updated information in real time and notifications when customized scholarships are newly updated.

2.4.3 Backend

There were several challenges along the way. The first challenge was the integration of MYSQL and Django. Django saves model changes to python files located in the migrations directory. However, there have been occasional situations where django is not properly recognizing changes to the model. When testing the data locally, the migration file was deleted, but when the DB was uploaded to the server, this solution could cause a big problem, so the above method was considered dangerous. In the meantime, I came to know that migration is a convenience feature of Django and is not required but optional. After modifying the Django model and directly modifying the table contents without migration through MYSQL WORKBENCH, Django recognized that there were no errors and found that the server was running well.

The second challenge was to determine the user's login status. Each user should be able to perform certain tasks only within the privileges they are allowed. For example, writing on the notice board only for users with administrator rights, and functions such as exam timetables only for users of the Department of Computer Science and Engineering. There is login decorator function provided by Django by default, but for some reason the decorator did not work well. So, it took a bit of repetitive work, but we used a method to verify user information through the process of acquiring and decrypting tokens from the header part delivered when requesting http.

3 Implementation

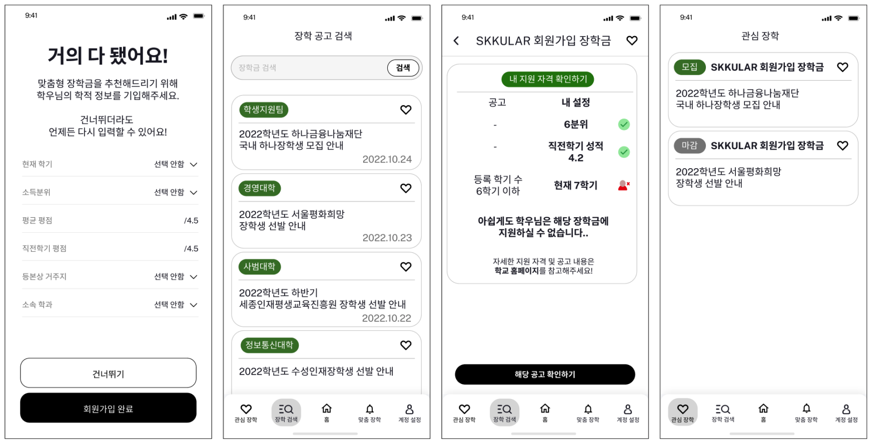
3.1 Dataset

The dataset consists of announcements, questions, and answers to questions. First of all, the notice was crawled in the notice of schools and each department. Crawled items are the notification number, notification creation date, notification title, notification content, attachment URL, attachment URL, and notification department. The attachments were crawled only when there were attachments. Also, the questions consisted of eight questions: "What grade can I apply for?" "What is my last semester's score?" "What is my income grade?" "What is my related department?" "When is the deadline?" and "What is the specificity?". The dataset is 256 contents, and 2048 (256*8) questions and answers.

3.2 BERT hyperparameters

Hyperparameters of BERT model are learning rate, epoch and Batch size. we fixed hyperparameters as epoch: 5, learning-rate: 3e-5, batch size: 4.

3.3 UX/UI viewpoints



거의 다 됐어요!

맞춤형 장학금을 추천해드리기 위해
학우님의 학적 정보를 기입해주세요.

건너뛰더라도
언제든 다시 입력할 수 있어요!

현재 학기	선택 안함
소득분위	선택 안함
평균 평점	/4.5
직전학기 평점	/4.5
등본상 거주지	선택 안함
소속 학과	선택 안함

학생지원팀
2022학년도 하나금융나눔재단
국내 하나장학생 모집 안내
2022.10.24

경영대학
2022학년도 서울평화희망
장학생 선발 안내
2022.10.23

사범대학
2022학년도 하반기
세종인재평생교육진흥원 장학생 선발 안내
2022.10.22

정보통신대학
2022학년도 수성인재장학생 선발 안내

SKKULAR 회원가입 장학금

공고	내 설정
-	6분위
-	직전학기 성적 4.2
등록 학기 수 6학기 이하	현재 7학기

아쉽게도 학우님은 해당 장학금에
지원하실 수 없습니다..

자세한 지원 자격 및 공고 내용은
학교 홈페이지를 참고해주세요!

관심 장학

모집 **SKKULAR 회원가입 장학금**
2022학년도 하나금융나눔재단
국내 하나장학생 모집 안내

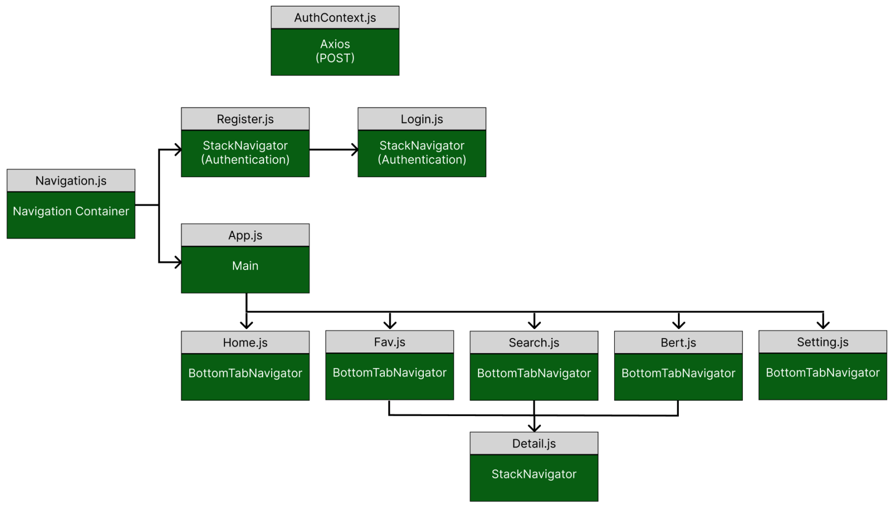
마감 **SKKULAR 회원가입 장학금**
2022학년도 서울평화희망
장학생 선발 안내

To explain the UI for the core pages of the app, first of all, the page for writing academic information when signing up can be selected. When signing up, the identity authentication process was simplified, and the reasons are as follows. First of all, there is less possibility of an abuser's intervention. Since it is based on the scholarship notice published in the notice of Sungkyunkwan University, the entry factor of outsiders is not large, and the application for scholarship is also judged to be less likely to be intervened by the outsider because it is applied through the mail address announced on the website. In addition, since sensitive information such as income quintile and credits may be included in filling in academic information, the certification process was simplified so that students could not be specified, reducing the burden of disclosing information. When entering academic information, credits was entered in numbers, but information such as department, residence, and semester was used in the drop-down select box.

On the "장학 검색" page, users can check crawled scholarship information, such as title, date, and information of the department which the notice was posted, and also provide a search function. When you click on a specific notice, it organizes scholarship requirements easily for students to determine whether they can apply based on academic information entered, and provides quick link button so that they can easily check the announcement. In addition, when you click the heart-shaped icon, the notice will go to the "관심 장학" tab so that students can see the notices they are interested in.

3.4 Frontend Detail

3.4.1 Frontend structure

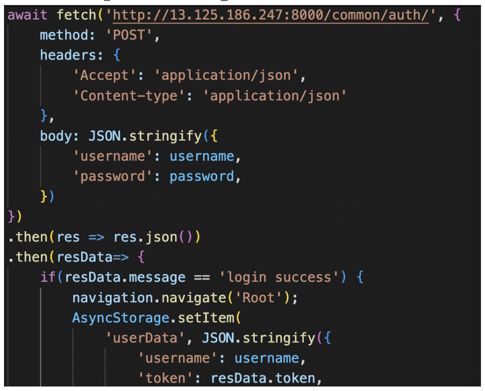


App SKKULAR was developed using React-Native and consists of the following framework. The authentication process was performed using Stack Navigator in the entire Navigation container. As a result, users can access the main page after logging in, and when signing up, information can be transferred to DB using Axios POST in AuthContext.js file. After login process, user can finally access the main page, App.js, which consists of five Bottom Tab Navigator. It moves to Home.js with the Default value, and various functions can be used using the bottom navigation bar. At this time, if you click on the scholarship notice listed in Fav.js, Search.js, Bert.js, you will go to Detail.js with the key value of the post and show detailed information related to the scholarship to the student based on it.

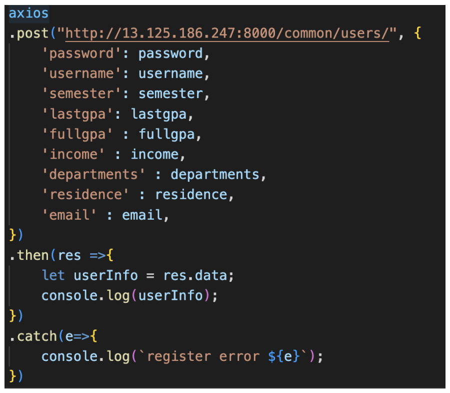
3.4.2 Connect with backend

When accessing the API, we used fetch and axios. There are some differences in code when implementing, such as the transmitted factor, but both produce the same results. It doesn't matter if we use any method, but React Native is not yet a fully stabilized framework with continuous version updates, so we mainly used fetch that is a built-in library, for stability. Axios was used only to receive user information and send a POST request in sign-up page, and fetch was used in all other cases.

The following is the part that sends a post request using fetch and axios, when implementing SKKULAR.



```
await fetch('http://13.125.186.247:8000/common/auth/', {
    method: 'POST',
    headers: {
        'Accept': 'application/json',
        'Content-type': 'application/json'
    },
    body: JSON.stringify({
        'username': username,
        'password': password,
    })
})
.then(res => res.json())
.then(resData=> {
    if(resData.message == 'login success') {
        navigation.navigate('Root');
        AsyncStorage.setItem(
            'userData', JSON.stringify({
                'username': username,
                'token': resData.token,
```



```
axios
.post("http://13.125.186.247:8000/common/users/", {
    'password': password,
    'username': username,
    'semester': semester,
    'lastgpa': lastgpa,
    'fullgpa' : fullgpa,
    'income' : income,
    'departments' : departments,
    'residence' : residence,
    'email' : email,
})
.then(res =>{
    let userInfo = res.data;
    console.log(userInfo);
})
.catch(e=>{
    console.log(`register error ${e}`);
})
```

3.5 Backend Detail

The API specification provided by the backend is as follows.

index	Method	URL	Description
1	POST	signup/	Registration
2	POST	login/	Login
3	PUT	user/update/	Edit member information
4	GET	user/{userId}/	Get member information
5	POST	favorscholar/	Add interest scholarship
6	GET	favorscholar/{scholarId}/	Get interest scholarship
7	DELETE	favorscholar/{scholarId}/	Remove interst scholarship
8	GET	scholar/	Get all scholarship information
9	GET	scholar/{scholarId}/	Get certain scholarship information
10	GET	token/refresh/	Refresh JWT token
11	GET	bert/{scholarId}/	Get bert matched scholarship
12	GET	bert/	Get all bert information

4 Evaluation

As performance evaluation indicators, F1-score and EM-score was used to measure the performance of the model. F1-score is the harmonized average value of the precision and reproduction rate between the predicted and actual answers, and EM score is the degree to which the word units of the Korean-based answer exactly match. As a result, EM-score was 76%, and F1-score was 78%(tested by 208 datasets).

5 Limitations and Discussions

5.1 AI

The limitation of AI began with a variety of scholarships. Therefore, exceptions were set for each process. In the crawling process, scholarships that are not eligible for undergraduate students and that are about the results of the scholarship selection were excluded. The second process, the labeling process, excludes health insurance calculations from income calculation criteria. It is calculated according to the payment amount by looking at the health insurance payment statement, but this would be considered a difficult condition to enter from the perspective of a user who is a student user. In addition, questions related to last semester's credits were excluded because they were mainly 12 credits, and questions related to whether the student got another scholarship were also excluded because the announcement is mainly similar in that the student must not have gotten another scholarship this semester. Third Process, when studying the BERT model, if there are multiple scholarships in a scholarship announcement, they were excluded from the study data. Also, if the attached file is an image format or an hwp file, the data extracted by OCR and Textract is not accurate and took a long time to exclude it from the data. Finally, when storing the contents of scholarships and learned data in a database, scholarships that are not required to be applied for are excluded.

5.2 App

Limitation in terms of the front-end may be a D-day notification. Looking at the existing proposal, we tried to implement a function that shows D-Day based on the deadline of submission of scholarship and sends a notification to users a day before the deadline, but this was not implemented. First of all, there was a problem in the process of unifying the data type,. Deadline extracted from the AI model was not output in a unified form, but it was solved through the data preprocessing process. Since then, the front-end has been implemented to calculate and shows D-Day using Date() data type, but it has not been able to implement notification function in time because we were not familiar with React-Native and Expo especially on notification. Therefore, before beginning the actual distribution and beta test stage, additional related functions will be searched and reorganized.

The limitation in the backend was server management. After testing locally, loading the changes directly from the server took a lot of effort. Because of this, we felt that doing the deployment ourselves was unnecessary repetitive work. According to the needs of these developers, recently, there is a library that supports automated distribution such as ZAPPA. So, if I had made better use of this tool, I think I could have focused on other tasks. Additionally, it seems that you are not using the web server properly. Since our project reached the demo version, the load on the server did not occur, but I think that the performance of the program could have been improved by using the web server a little better. Also, it was not possible to secure various users during the debugging process. No matter how well you design your logic, many exceptions can happen. Errors frequently occurred in the process of debugging our project as well. However, I think that there are enough exceptions that have not been considered because the main subject of debugging is the people who make the application.

6 Related Works

There are several services that allow university students to effectively check the notice of scholarships.

6.1 Mobile Application

Uniview is an application that helps students with convenient and smart university life and supports both Android and iOS devices. It shows notices by universities and provides search and alert services by keyword. Key features include: view notices, search and alert by keyword, bookmarks, and quick links to access specific sites directly. However, it does not sort notices about scholarships.

6.2 Website

DreamSpon is a website that provides various scholarship information for university students in real time. It shows the information on 3,000 scholarships, including scholarships by the school, regional, and foundation scholarships. Furthermore, DreamSpon offers its own scholarships in various forms, including goods, meals, and mentoring, as well as financial support to cheer for university students' dreams. It is also actively using SNS communication channels such as KakaoTalk and Instagram for the efficient operation of the website. Currently, the service is provided in a way that provides information collectively. However, it does not recommend the eligibility of each user.

7 Conclusion

We planned and developed the app SKKULAR to solve the students' indifference to applying for scholarship and the resulting side effects that we identified as problems within Sungkyunkwan University. To fundamentally address the inconvenience of scholarship search, we have made it easier and faster for students to determine whether they can apply for a scholarship by using AI models to extract and compare the application qualifications specified in the scholarship notice, not just crawl the date.

In this process, we tried to solve the problem using tools such as React-Native, Expo, MySQL, and Django framework, along with AI models such as BERT, Rule-based, and QnA, and the final deployment format became a mobile application that students can frequently find and receive notifications easily.

Along with the contents mentioned in the limitation, tasks such as performance improvement and additional functions remain. However, to promote students' interest in scholarships to prevent them from receiving scholarships even though they need them, and to further expand the good influence of the app to other universities, our team aims to accelerate development and debugging and officially distribute them during the next semester.

References

1. 공지사항 게시판목록: 성균관대학교. 성균관대학교, SKKU, 성균관대, 성대, Sungkyunkwan University. (n.d.). Retrieved October 2022, from $https://www.skku.edu/skku/campus/skk_comm/notice06.do$
2. 학부대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from $https://hakbu.skku.edu/hakbu/community/under\_notice.do?mode = list\&amp;srCategoryId1 = 236\&amp;srSearchKey = \&amp;srSearchVal =$
3. 유학대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from $https://scos.skku.edu/scos/community/under\_notice.do?mode = listamp;srCategoryId1 = 236amp;srSearchKey = amp;srSearchVal =$
4. 문과대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from $https://liberalarts.skku.edu/liberal/community/under\_notice.do?mode = listamp;srCategoryId1 = 236amp;srSearchKey = amp;srSearchVal =$
5. 사회과학대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from $https://sscience.skku.edu/sscience/community/under\_notice.do?mode = listamp;srCategoryId1 = 236amp;srSearchKey = amp;srSearchVal =$

6. 경제대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //ecostat.skku.edu/ecostat/community/under_notice.do?mode = listsrCategoryId1 = 236srSearchKey = srSearchVal =*
7. 경영대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //biz.skku.edu/bizskk/community/notice.do?mode = listsrCategoryId1 =* 754*srSearchKey = srSearchVal =*
8. 사범대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //coe.skku.edu/coe/community/under_notice.do?mode = listsrCategoryId1 =* 236*srSearchKey = srSearchVal =*
9. 예술대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //art.skku.edu/art/community/under*$_n$*otice.do?mode = listsrCategoryId1 =* 236*srSearchKey = srSearchVal =*
10. 자연과학대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //cscience.skku.edu/cscience/community/under*$_n$*otice.do?mode = listsrCategoryId1 =* 236*srSearchKey = srSearchVal =*
11. 정보통신대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //ice.skku.edu/ice/community/notice.do?mode = listsrCategoryId1 =* 816*srSearchKey = srSearchVal =*
12. 공과대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //enc.skku.edu/enc/community/under*$_n$*otice.do?mode = listsrCategoryId1 =* 236*srSearchKey = srSearchVal =*
13. 성균융합원. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //icon.skku.edu/icon/community/under*$_n$*otice.domode = listsrCategoryId1 =* 236*srSearchKey = srSearchVal =*
14. 글로벌융합대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //sco.skku.edu/sco/community/notice.do?mode = listsrCategoryId1 =* 267*srSearchKey = srSearchVal =*
15. 생명과학대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //skb.skku.edu/biotech/community/under_notice.do?mode = listsrCategoryId1 = 236srSearchKey = srSearchVal =*
16. 스포츠과학대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //sport.skku.edu/sports/community/under*$_n$*otice.do?mode = listsrCategoryId1 = 236srSearchKey = srSearchVal =*
17. 소프트웨대학. 통합게시판 게시판목록. (n.d.). Retrieved October 2022, from *https : //cs.skku.edu/ko/news/notice/list*