TimeCodi: Time Coordination Service

Team B
Geunyeong Park, Hyunyoung Lee, Soeyeong Kim, Seoyeon Cho, Sihyeon Hong

Sungkyunkwan University,
Capstone Design Project

Abstract. This document describes TimeCodi, a service that allows users to coordinate time with others and set appointments. TimeCodi simplifies the process of coordinating schedules with multiple people, as it can be cumbersome and time-consuming to do so manually. TimeCodi has three main functions: registering a personal schedule, creating a group meeting, and checking a friend's calendar. First of all, personal schedules can be registered and managed in the calendar. Second, you can create a group to check the time when you can make an appointment based on each member's schedule and set up a meeting. Finally, you can check your friend's calendar by adding him to the friends list. The development process can be divided into two parts: front-end and back-end. The front-end used React to create pages. Also, the back-end used MariaDB and FastAPI to store data and manage users' personal information. By using this application, users can reduce the burden of sharing their schedules every time they need to coordinate with someone, especially when dealing with a large group.

Keywords: schedule coordination · multiple people · calendar · meeting

1 Introduction

We share our daily lives with others by going to school, work, hanging out with friends, or collaborating with colleagues. What do these things have in common? The commonality is that they require schedule coordination with multiple people. Even when we have to get together to do team projects, like us, we need to coordinate our schedules. We are constantly sharing our time with others. We have to coordinate with multiple people often every month, and it can be very cumbersome to ask each person for their availability and repeat the process until we find a good time. Although there are many applications that can help with this process, a service that can solve the repetitive task of coordinating schedules with multiple people has not yet emerged.

To solve this problem, we have created an application called "TimeCodi", which gets the schedules of multiple people and generates a list of available times for all members. By using this application, we can reduce the burden of sharing our schedules every time we need to coordinate with someone, especially when dealing with a large group.

The development process of TimeCodi can be divided into two parts: front-end and back-end. In the front-end part, we used React, which is one of the JavaScript libraries for UI, to create pages. In the back-end part, we used MariaDB DBMS(Database Management System) to store users' data and used FastAPI to build a server and to create RESTful APIs.

The TimeCodi app allows users 1) to get their Google Calendar schedule and register it in their calendar or register the schedule themselves, 2) to check the group list they belong to and they are invited to, and 3) create a group and invite members. On the group page, 4) it displays the available times based on the members' schedules, 5) an administrator of group can generate a vote for a meeting time, and 6) the administrator can enter meeting information. Lastly, 7) users can check their friend's timetable by adding another user as a friend.

Consequently, the final features are: 1) Register as a member 2) Login 3) My Groups - the group list you belong to or are invited to 4) My Timetable - manage personal schedule 5) Withdrawal 6) Group Bookmarks - you can check members' schedule and make a group meeting 7) Friends 8) How-to that explains how to use TimeCodi.

2 Design

2.1 Overall Architecture

TimeCodi's system can be divided into three parts: front-end(web browser), back-end(server) that handles requests from the front-end, and a database that stores the system's data. A request according to interaction with the user at the front-end enters the server. The server takes the appropriate response from the database for the type of request and returns it to the front-end. The overall architecture of TimeCodi is shown in the following diagram.

2.2 Core Skill

Front-end We used React, a JavaScript library used to create a user interface. It can be used to develop single-page applications or mobile applications. Also, we used Google Map API to input location information into group meeting information.

Back-end We have built our server using the FastAPI web framework and managed data storage with the MariaDB DBMS. For the server, we utilized the FastAPI web framework to create RESTful APIs based on the required functionalities. As for the database, we employed SQLAlchemy, one of the Python libraries, to establish a connection with the MariaDB database. Through this connection, we performed indirect manipulation of the database.

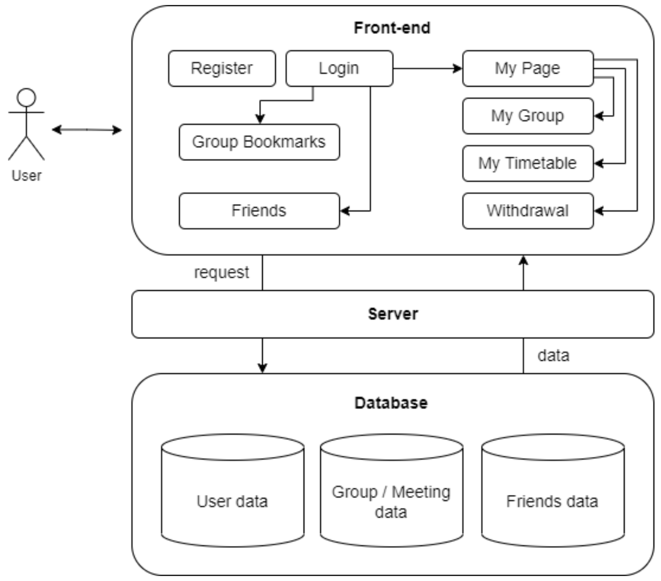


Fig. 1. Overall Architecture

2.3 Reasoning

Front-end React can be used to develop single-page applications or mobile applications. Another function of the React is the use of a virtual document object model, that is, a virtual DOM. React creates an in-memory data structure cache, calculates the result difference, and effectively updates the DOM displayed in the browser. This allows us to write code as if every page is rendered every time it changes.

Back-end The reasons for creating a RESTful API and utilizing SQLAlchemy, an ORM, are as follows.

Creating a RESTful API offers several advantages. Firstly, it improves readability. By simply reading the messages of a REST API, one can clearly understand their intentions. Secondly, it provides scalability and flexibility. RESTful APIs are designed based on a resource-oriented architecture, which allows for excellent scalability and flexibility. New functionalities can be added or modified without the need to alter existing endpoints. Thirdly, it ensures consistency. RESTful APIs offer a consistent interface. By using HTTP methods (GET, POST, PUT, DELETE, etc.) and resource paths, developers can make API calls in a standardized manner. Fourthly, it enables clear representation. REST APIs can express necessary data in the request body by specifying the URI processing method in the header section. Lastly, RESTful APIs support strict separation between clients and servers. As a result, clients and servers can evolve independently without impacting each other's operations.

By using an ORM, there is no need to write SQL queries separately, as the database can be manipulated indirectly through objects. This provides a simpler and more convenient environment.

2.4 Challenges

Front-end There are some challenges that we have faced in implementing the front-end.

The first is the version conflict issue of node.js. React is developed using node.js. We had a hard time sharing the code because the version of node.js used by the team members in charge of developing the front-end was different. We solve this version conflict issue by unifying the development environment as well as node.js.

The second challenge is the use of the Google Map API. Google Map API key is required to use Google Map API. We used github when sharing code, API key should not be shared in public places such as github. So we personally shared the API key and developed front-end with care not to upload it to github.

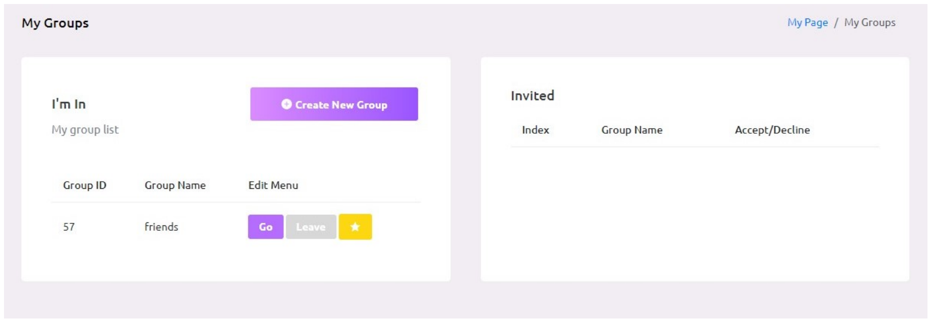
Back-end There are also some challenges that we have faced in implementing the back-end. When attempting to retrieve and save a user's Google Calendar events into the database, an error occurred on the server due to an excessive number of event results. When implementing a function to retrieve user's schedule information from the database, I encountered an error on the server when attempting to retrieve all events for a specific date. This was due to sending a large number of requests to the server when fetching all events for the user. To address this issue, I attempted a different approach where the function retrieves all events in a single request and then filters out the necessary information.

3 Implementation

3.1 UX/UI Viewpoints

Followings are the representative UI of TimeCodi.

First, when a user log in after registering in TimeCodi, the My group list screen appears. This page lists the groups the user belong to and the groups the user is invited to.

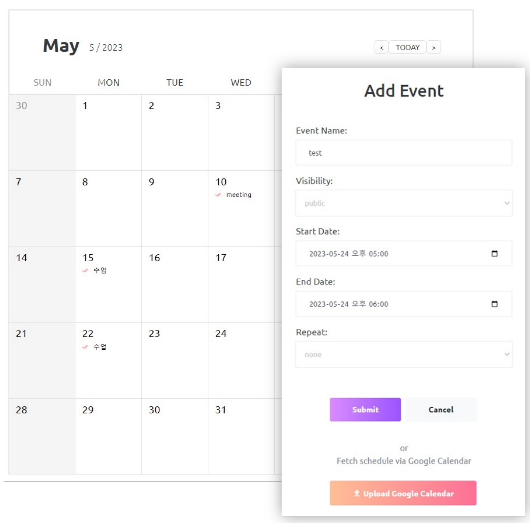


Group ID	Group Name	Edit Menu
57	friends	Go Leave ★

Index	Group Name	Accept/Decline

Fig. 2. My group list

Clicking the ‘My Timetable’ menu takes the user to a page with a UI where the user can add a calendar and a schedule. When the user add a schedule through the Add Event UI, it appears in the calendar. Double-click a specific date in the calendar to show the schedule that belongs to that date. The user can modify and delete the schedule by clicking the Edit button on the schedule.



Event Name: test

Visibility: public

Start Date: 2023-05-24 오후 05:00

End Date: 2023-05-24 오후 06:00

Repeat: none

Fig. 3. My timetable

On the group page, the group calendar has a pink button next to each week. When the user click the pink button, the available time zone of the members of the week is displayed in the timetable. The administrator of the group creates a vote by clicking the 'Generate Vote' button, and the members can vote to set the meeting time.

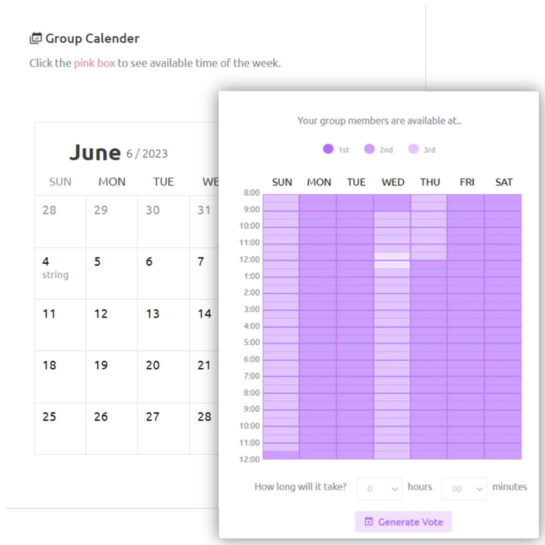


Fig. 4. Group calendar

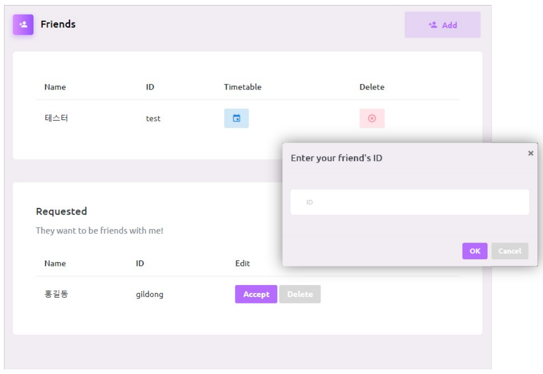
The Friends page shows the user's friend list. The user can send a friend request to the friend by entering the friend's ID. When the friend accepts a request, it is added to the friend list. The user can check friend's schedule by clicking the timetable button and delete it from the list by clicking the delete button.

3.2 Front-end Detail

This introduces the detail functions of the group page that have not been explained before.

We implemented vote UI to setting meeting time. The administrator of the group creates a vote by clicking the 'Generate Vote' button, and the members can vote to set the meeting time they want. The administrator can end the vote and when the vote is finished, the result of the vote is displayed in the UI on the right.

After the vote, when the administrator presses the 'Make Meeting' button, a pop-up window appears to enter meeting information. We used Google Map API



Friends

Add

Name	ID	Timetable	Delete
테스터	test		

Enter your friend's ID

ID

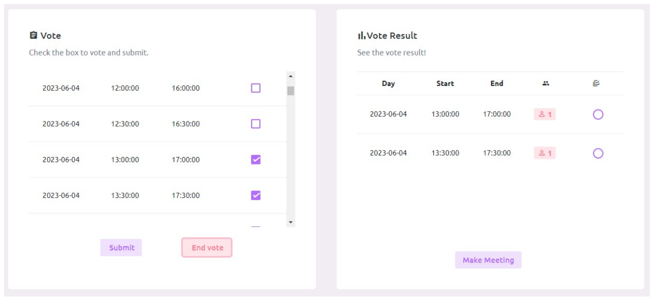
OK Cancel

Requested

They want to be friends with me!

Name	ID	Edit
홍길동	gildong	Accept Delete

Fig. 5. Friends list



Vote

Check the box to vote and submit.

2023-06-04	12:00:00	16:00:00	☐
2023-06-04	12:30:00	16:30:00	☐
2023-06-04	13:00:00	17:00:00	☑
2023-06-04	13:30:00	17:30:00	☑

Submit End vote

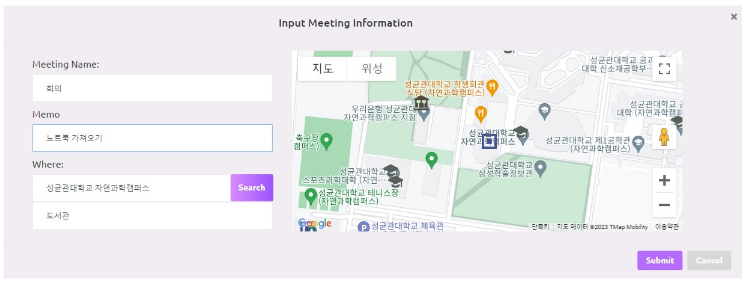
Vote Result

See the vote result!

Day	Start	End		
2023-06-04	13:00:00	17:00:00	1	○
2023-06-04	13:30:00	17:30:00	1	○

Make Meeting

Fig. 6. Vote UI



Input Meeting Information

Meeting Name: 회의

Memo: 노트북 가져오기

Where: 성균관대학교 자연과학캠퍼스

도서관

Search

Submit Cancel

지도
위성
성균관대학교 학생회관 식당 (자연과학캠퍼스)
우리은행 성균관대 자연과학캠퍼스 지점
성균관대학교 자연과학캠퍼스
성균관대학교 제1공학관 (자연과학캠퍼스)
성균관대학교 삼성학술정보관
성균관대학교 스포츠과학대학 (자연…
성균관대학교 테니스장 (자연과학캠퍼스)
성균관대학교 체육관
Google
단축키
지도 데이터 ©2023 TMap Mobility
이용약관

Fig. 7. Meeting information

to input the location in the meeting information. When the administrator enter the location in the search box and the map will be displayed on the right screen. If the administrator press the OK button, the entered meeting information is registered in the calendar.

3.3 Back-end Detail

Creating a database We created a MariaDB database using HeidiSQL. We deployed it to ensure that it is connected properly not only in the server and local environment but also in other environments.

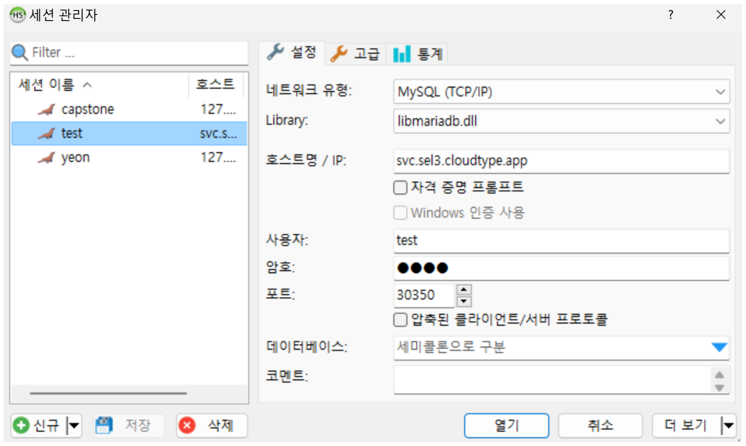
Connecting to a database In order for the web server to provide appropriate data to the client, it needs to have access to the database. Additionally, since we are using SQLAlchemy, establishing a connection with the database is essential. We created a function to connect to the database and manage the data, providing the necessary values to access the database on the server.

Creating API We have created RESTful APIs using the FastAPI web framework. The functions that return information are designed to handle GET requests, the functions that register new information handle POST requests, the functions that update existing information handle PUT requests, and the functions that delete existing information handle DELETE requests. Moreover, all these functions are implemented as asynchronous functions. By implementing these functions as asynchronous, we ensure that unexpected situations, such as returning information before processing it, are avoided. We have utilized the async/await keywords to create asynchronous functions. Each function connects to the database and manages and returns data based on its specific functionality. In particular, for implementing the login function, we utilized the JWT token authentication method. Along with that, we leveraged the authentication library provided by Python. Almost all requests include user information and are sent with the user's authentication token. If the information provided is not valid, we raise an HTTP exception to handle the error.

OPEN API We utilized the open APIs of KakaoTalk and Google Calendar.

We used the KakaoTalk Open API to enable users to invite other users to a group by sending KakaoTalk messages to their KakaoTalk friends. This integration allowed us to facilitate the invitation process through KakaoTalk messaging.

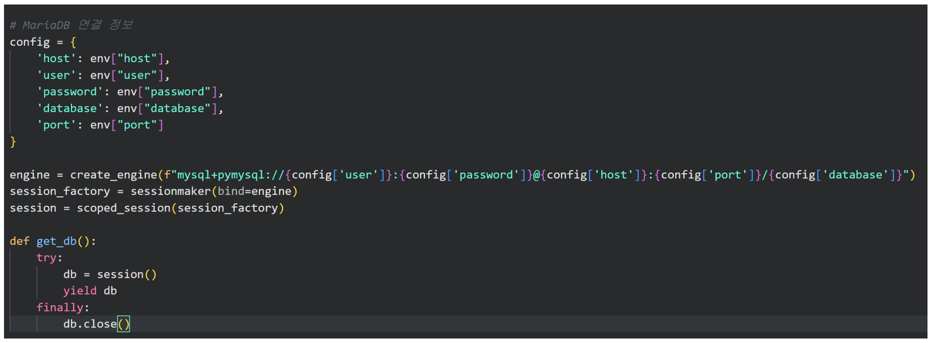
We utilized the Google Calendar Open API to retrieve and store information from users who have already registered their schedules in Google Calendar. This integration enabled us to fetch and store their calendar data efficiently.


세션 관리자

세션 이름	호스트
capstone	127....
test	svc.s...
yeon	127....

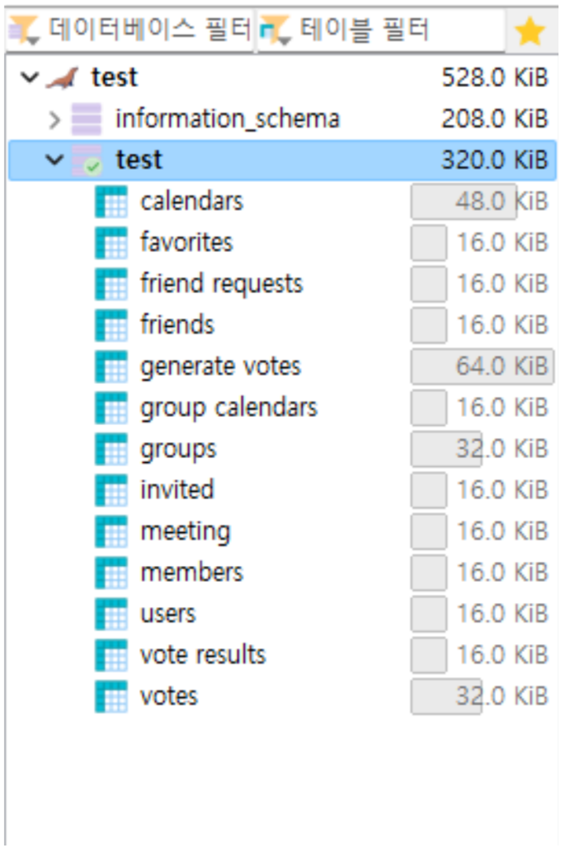
네트워크 유형: MySQL (TCP/IP)
Library: libmariadb.dll
호스트명 / IP: svc.sel3.cloudtype.app
☐ 자격 증명 프롬프트
☐ Windows 인증 사용
사용자: test
암호: ●●●●
포트: 30350
☐ 압축된 클라이언트/서버 프로토콜
데이터베이스: 세미콜론으로 구분
코멘트:

Fig. 8. Creation of database



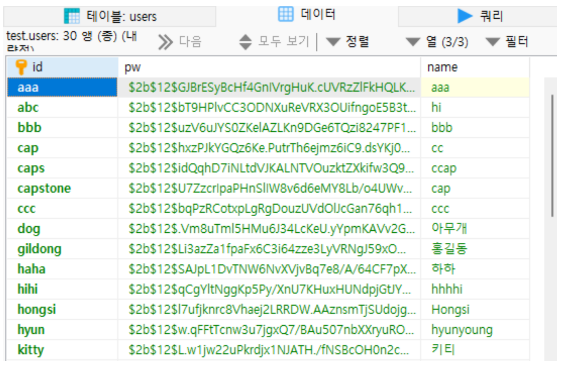
```
# MariaDB 연결 정보
config = {
    'host': env["host"],
    'user': env["user"],
    'password': env["password"],
    'database': env["database"],
    'port': env["port"]
}

engine = create_engine(f"mysql+pymysql://{config['user']}:{config['password']}@{config['host']}:{config['port']}/{config['database']}")
session_factory = sessionmaker(bind=engine)
session = scoped_session(session_factory)

def get_db():
    try:
        db = session()
        yield db
    finally:
        db.close()
```

Fig. 9. Connecting to database



test	528.0 KiB
information_schema	208.0 KiB
test	320.0 KiB
calendars	48.0 KiB
favorites	16.0 KiB
friend requests	16.0 KiB
friends	16.0 KiB
generate votes	64.0 KiB
group calendars	16.0 KiB
groups	32.0 KiB
invited	16.0 KiB
meeting	16.0 KiB
members	16.0 KiB
users	16.0 KiB
vote results	16.0 KiB
votes	32.0 KiB

Fig. 10. Database Entity



test.users: 30 행 (총) (대략적)

id	pw	name
aaa	$2b$12$GJBrESyBcHf4GnlVrgHuK.cUVRzZlFkHQLK...	aaa
abc	$2b$12$bT9HPlvCC3ODNXuReVRX3OUifngoE5B3t...	hi
bbb	$2b$12$uzV6uJYS0ZKelAZLKn9DGe6TQzi8247PF1...	bbb
cap	$2b$12$hxzPJkYGQz6Ke.PutrTh6ejmz6iC9.dsYKj0...	cc
caps	$2b$12$idQqhD7iNLtdVJKALNTVOuzktZXkifw3Q9...	ccap
capstone	$2b$12$U7ZzcrIpaPHnSlIW8v6d6eMY8Lb/o4UWv...	cap
ccc	$2b$12$bqPzRCotxpLgRgDouzUVdOlJcGan76qh1...	ccc
dog	$2b$12$.Vm8uTml5HMu6J34LcKeU.yYpmKAVv2G...	아무개
gildong	$2b$12$Li3azZa1fpaFx6C3i64zze3LyVRNgJ59xO...	홍길동
haha	$2b$12$SAJpL1DvTNW6NvXVjvBq7e8/A/64CF7pX...	하하
hihi	$2b$12$qCgYltNggKp5Py/XnU7KHuxHUNdpjGtJY...	hhhhi
hongsi	$2b$12$l7ufjknrc8Vhaej2LRRDW.AAznsmTjSUdojg...	Hongsi
hyun	$2b$12$w.qFFtTcnw3u7jgxQ7/BAu507nbXXryuRO...	hyunyoung
kitty	$2b$12$L.w1jw22uPkrdjx1NJATH./fNSBcOH0n2c...	키티

Fig. 11. User Info

4 Evaluation

We have implemented all the functions that we had planned in the beginning. We created an account and tested adding, editing, and deleting schedules to personal calendar. We also created several more accounts to test creating groups and inviting members. We checked the available time zones of the members, generated a meeting time vote, selected the result of the vote, entered the meeting information, and registered it in the group calendar. Finally, we sent a friend request to see if we could make friends. Therefore, the functions of our web page implemented are expected to be very useful in real life.

5 Limitations and Discussions

Users need to register their personal schedule to coordinate time with others using TimeCodi. It will be easy to upload to TimeCodi's calendar using Google Calendar, but there is a hassle of having to register the schedule directly if the users are not using it. In addition, it will be difficult to set a meeting time reflecting their schedule unless several members in the group have set up personal calendars.

Also, when mapping the available time zone on the group page, we set the time unit to 30 minutes. So, it was impossible to recommend a more accurate meeting time. If we had mapped with a smaller unit of time, like 10 minutes, we could have recommended a more efficient and accurate meeting time.

It took a long time to develop, so we haven't tested it on many people, but we are planning to refine and test the UI to make it easy to use with users' mobile phone.

6 Related Work

6.1 Related Services

When2meet When2Meet is a service that allows multiple people invited to a link to register their schedules, and based on the stored schedule data, everyone can see the available time slots at a glance. In addition, the entire timetable is displayed, and the number of people available at each time slot is represented by the intensity of the color, making it highly visible, and you can also see who is available at that time slot if you select a specific time. However, a fatal drawback of this service is that you have to enter your schedule every time you create a new group. Also, you cannot check a specific person's entire schedule.

Everytime Everytime is an application that allows college students to save their schedules and share them with friends. It has the advantage of reducing the hassle of entering information every time by storing user schedule information in the users' account, and allowing one to check the schedules of specific people. However, this application only supports the function of storing and checking users' schedules, and it does not provide a function to find a time slot that multiple members agrees on, so it is highly inefficient for scheduling purposes.

Doeneun-sigan Doeneun-sigan supports various functions related to team meetings, and because users can save personal schedule information in their accounts, they do not have to enter information every time they coordinate their schedules. However, this application is inefficient for groups with multiple people because it is based on the form of making reservations for the available time of the other person one-to-one, without providing a function to find a time slot that everyone can agree on.

6.2 Difference between our app and others

Existing scheduling services, as mentioned above, each have a fatal flaw. To summarize, if they provide a function to save schedule data in user accounts, they do not have a function to come up with time slots that are available to multiple users, and if they provide the latter function, they do not have the former. Ultimately, any service taken to schedule with multiple people will result in some form of inconvenience. Therefore, a new scheduling service that provides both of these important functions is needed. In response to this need, we propose the new service called TimeCodi. TimeCodi provides both of these important functions simultaneously, as well as additional features, making it a more efficient service than any existing scheduling application.

7 Conclusion

While there are existing services that store user schedules and create group events, our service offers unique advantages compared to others. We integrate with Google Calendar and KakaoTalk, providing seamless connectivity. Additionally, our service stands out by automatically identifying available time slots and creating instant polls. These distinct features set us apart from other services. We believe that these advantages offer significant value to users and make our service worth exploring.

8 References

1. React Homepage, https://ko.legacy.reactjs.org/. Last accessed 9 June 2023
2. Node.js Wikipedia, https://ko.wikipedia.org/wiki/Node.js. Last accessed 9 June 2023