Contribution Title*

Bo-Kyung Seong, Jeong-Yeon Mun, Kyu-Yong Kim, and Yoo-Ho Song

Sungkyunkwan University, Capstone Design Project
Project D

Abstract. Walking is something people always experience in their daily lives. Walking is the easiest and safest exercise and has many advantages, but it can be difficult to find pleasure in it as it is a daily routine for them. We thought it would be a more meaningful activity if a companion was added to this common walking, and we would like to provide people with a meeting space in this project to provide them with a special experience that is different from usual. Existing companion applications have limited situations for special events such as travel. So, this project includes the ability to find a walking mate so that people can find companies not only special events that occur rarely, but also in daily life. In addition, all of the existing neighborhood mate matching apps had a strong characteristic of dating apps, so we wanted to create apps with a strong characteristic of "walking." Therefore, our ultimate goal was to focus on the activity of "walking" and develop an application that could find mate without harming this purpose. Since offline encounters are the main focus, various function have been configured to ensure reliability and include mate filtering functions that make it easier to find mates. Also, it has motivational challenges function to encourages steady walking and increase loyalty to the app.

Keywords: Reliable· walk· trip· mate· feed

1 Introduction

1.1 Background and Goal

Enough personal time is very important for humans. But fundamentally, humans are interdependent. There is a big difference between having and not having someone to do something. Accompanying people of the same mind can be a great pleasure for someone, and sometimes just being next to someone without saying anything is comforting. This project created the application to provide people who want this pleasure or comfort with a safe space to find a walking mate and the pleasure of being with someone. In addition, many daily recording apps have been released recently, and no application has been found to record walking. It has added a walking tracking recording function to increase the differentiation from other applications related to walking and to allow users to use the application more routinely in addition to simply finding an e-mate. Through

* Supported by organization x.

these functions, we want to add meaning to the behavior of trite walking to users.

1.2 Brief

We used Firebase to store and manage user, post, and chat data. Firebase stores data in each online cloud for each project. We selected Naver Login API allowing us to receive user data to user the app from Naver. And APIs provided by Naver, Kakao, and T-map were used. There are four Naver APIs used: Mobile Dynamic map API is for using real-time map, Geocoding API is for coordinating value through address name, Reverse-Geocoding API is for getting corresponding location address name through coordinate value, and finally user Login API is for getting data. Among the APIs provided by Kakao, the address search API is used, and the pedestrian route search API is used in T-map. Other than that, MaterialCalendarview provided with various custom functions, viewpager2 capable of sequentially turning screens with horizontal scrolling, NavigationView acting as a sidebar, and NavigationBarView acting as a bottom bar were used.

The project mainly focuses on finding mates, but this is aimed at helping users find mates to "accompanied" by focusing on walking, not making friends within the app. To revive the purpose of this application, the project designed the application by adding other functions in addition to the Mate Finding Bulletin and walking tracking records. The project was largely divided into front-end and back-end. Front-end was in charge of UI/UX design, implementation, and functional implementation using Android Studio, and Back-end was in charge of database access, management, and security rules using Firebase. All the functions planned at the beginning of the project were implemented normally. As a result of receiving reviews through external acquaintances, there was an opinion that UI/UX was intuitive and convenient, but it was a concern about security aspects or malicious users.

2 Design

2.1 Application

Walking Mate's main functions are the Mate search bulletin board, walking tracking, feed writing, and chatting, and the bulletin board is divided into a walking bulletin board and a travel bulletin board. Each bulletin board has a post creation function, a post filtering function, a post display function, and an application and acceptance function for a post. Users can use the path tracking function they walked through the walking tracking function, the desired location marking function, and conveniently access the walking tracking content, stop function, and place marking outside the application through the Notification bar. In the feed, you can write emotions, weather, writings, and photos of the day, and check the route, the marked place, the number of steps, and the distance. The detailed design is as follows.

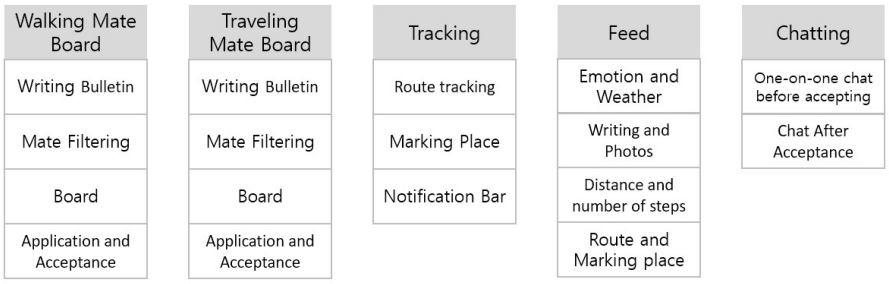

Fig. 1. The main function is a bulletin board for searching for Mate, tracking on foot, and writing feeds, and the bulletin board is divided into a walking bulletin board and a travel bulletin board. Each bulletin board has a post creation function, a post filtering function, a post display function, and an application and acceptance function for a post. In walking tracking, users can use the path tracking function they walked, the desired location marking function, and the Notification bar, and in feeds, emotions, weather, writings and photos can be written, and routes, marked places, steps, and distances are recorded.

Chatting : Walkingmates have completely relied on appointments to prevent them from deviating from the purpose of the application called 'walking'. There are only two ways to enter the chat. When someone sends an application to my post, you can open a personal chat room where you can talk to the applicant in advance before accepting it. Alternatively, when accepted, a group chat room of the publisher and the accepted applicants will be opened automatically. By limiting the chat function, we tried to prevent abuse of chat for socializing.

Reliability : Safety and reliability for strangers are important because Walkingmates leads offline meetings. It has reporting, Blocking, and review functions for this purpose. A user can report a mate who is not well-mannered, and if a total of three reports from another user overlap, the user is blocked by the system. In addition, if users simply grade an appointment partner, the score is converted into a reliability score and applied to the user's reliability. In addition, feed posts and challenge achievement symbols provide clues to understand the other person.

Motivation to user app : Features configured to increase users' loyalty to the app include path tracking, feeds, and challenges. Even without mate matching, users can leave a walking record when they take a walk or run alone. For this record, they can write a feed in the form of a diary, and see at a glance how consistently they have written the feed through the calendar. In addition, the challenges scores that naturally accumulate while utilizing the various functions of the working mate lead to the acquisition of a step-by-step title depending on the achievement.

2.2 API

Naver API

a. Login : This is the login function activation API provided by NAVER. It allows to receive user data to use the app from Naver, and select the necessary part of the data as either essential or additional. In addition, since the ID has already been authenticated and created by NAVER, there is no need for additional procedures to verify the user's identity.

b. Mobile dynamic Map : It is a map API that is updated in real time, and was used to display all map screens in the application. it makes application float markers on the map and draw routes based on a collection of location coordinates. In addition, by registering the location source, the user's location can be displayed on the map in real time, and the map range and location shown on the camera can be set through camera settings. Among the domestic map apps for walking tours, Naver map app was selected because it is the best in terms of information-related aspects and map accuracy, and is easy to develop and use.

c. Geocoding : When a destination is searched through address search when writing posts on a walk or travel, geocoding is used to obtain location coordinates based on the searched address name and display it on a map.

d. Reverse-Geocoding :When writing a walk or travel post, click the map screen to select a destination or starting point. At this time, in order to know the address name of the selected point, Reverse Geocoding was used to return the address name of the location after taking the location coordinate value.

Kakao API - Address Search : This API is used in combination with a file composed of JavaScript. When the html file is uploaded to the web server, the url of the file is called and executed, and the address search window is opened through the WebView in the application to search for the address in the search window. At this time, the hosting function of Firebase was used as a server to upload the html file. Kakao Address Search API can be used without the need for separate user registration, and was selected because it is the best among address search APIs in Korea.

T-map API - Pedestrian route search : When route search is requested with the location of the starting point, destination, and waypoint, the pedestrian route connecting the areas is brought in JSON file format. Among domestic map APIs, T-map is the only API that provides pedestrian route search service, so this API was selected.

2.3 View Functions

MaterialCalendarView : A widget for displaying a calendar. We can customize the text representing the year, month, day, date, etc., and the color that composes the calendar. It can also decorate the design of each date. In addition to initially setting the calendar design, we can create classes that implement DayViewDecorator to dynamically select and decorate the corresponding dates according to conditions. This widget was used to implement a complex feed calendar designed in the app conception stage, and to dynamically display walking tracking records and created feeds on the calendar.

ViewPager2 : A widget that enables sequential page turning through horizontal scrolling. Each page is composed of fragments, which can be added or deleted dynamically. It is implemented in a way to apply the configuration of fragments to appear on each page to ViewPager2 by customizing an Adapter that extends FragmentStateAdapter. This widget, which is used as a function to view attached images in feed creation and viewing, was selected because it has the most appropriate horizontal image flipping animation when displaying feed images and is easy to modify dynamically.

NavigationView :Depending on the set layout-gravity, it is a widget that can be hidden and displayed in the form of a sidebar on the left or right side. It consists of header and menu. In the case of Header, the layout is configured by customizing, and in the case of menu, it is composed of items in the xml file created in the menu folder. Header can be dynamically changed within the code. In the case of a menu button, we can override the onNavigationItemSelected() function in the code to set which action will be executed when each item is clicked. When designing the My Page sidebar, it consisted of the user profile at the top and the menu buttons below it, and NavigationView was selected because it had the most appropriate form to implement this design.

NavigationBarView : As a bar-type widget, the place where the widget is located is determined according to the layout-gravity value, and the item inside the bar is composed of the menu property value. At this time, the menu consists of items in the xml file created in the menu folder. In the case of menu buttons, we can override the onNavigationItemSelected() function in the code to configure which actions are executed when each menu button is clicked. At this time, the selected item is displayed on the screen. In the case of this widget, it was selected because it is easy to customize the design and easy to implement, as it is a widget that is widely used as the bottom navigation bar in applications such as Baedal Minjok and YouTube.

2.4 Data

Firebase

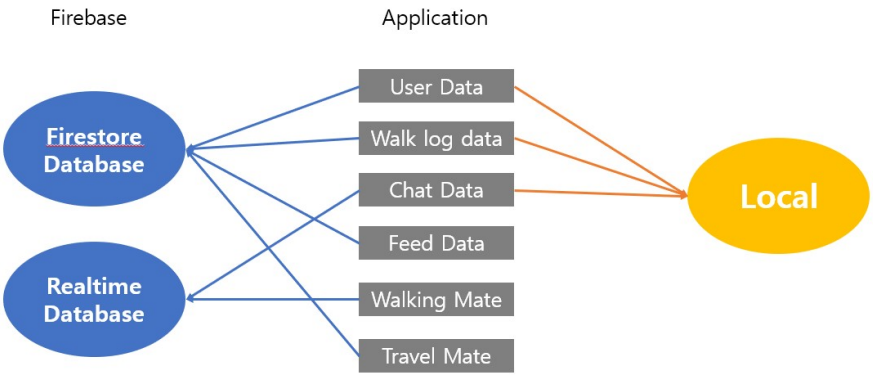

Fig. 2. The data generated and used in the application is stored in the Firestore Database and Realtime Database for each purpose. Among the data, User, Walking log, and Chat data are also stored in Local.

Since Firebase is a non-SQL database, we chose it because it is easy to handle.

a. Firestore database is a static database. Store data about user information, posts, reports, blocklists, feeds, and challenges. After creating a collection for each data category, the data is saved as a document within the collection. When a user sends data to be stored or requests a desired document through Query, it is processed in firestore. In the case of the above data, Firestore was used because it can be used by requesting and importing only when necessary.

b. Realtime database used to manage chat information. In addition to providing data at the first request, ValueEventlistner allows users to receive chat room and chat data by detecting data changes in real time when chat room data is requested once. Unlike the data used in Firestore, in the case of chatting, a real-time database was used because the data needs to be updated while exchanging conversations with chat room participants in real time.

Local Data Management

a. User data : It is stored and managed locally as well as Firebase in order to check conditions for automatic login and to quickly import user data when using it in each activity. It is stored in the userData folder within the dedicated directory of the application. Not only the text file in which user data is stored, but also the user profile image is stored. At this time, two types of user profile images are stored: a high-resolution version and a low-resolution version. In the case of an image file, it takes a lot of time to load after requesting it to Firebase, but if it is stored locally, it can be loaded quickly, preventing the problem of delay in loading profile images in Mypage.

Class UserData is used for managing user data. In the case of Field, it is configured in the same way as the document where user data is stored in Firebase. One difference is that in the case of UserData class, there is an additional field to store userid used as document id in firestore. The implemented method is getHashmap() which converts user data into a Hashmap to send to Firebase. scanUserData() to scan a file where user data is stored locally. saveData() to save the UserData object locally, and encode() to convert the UserData object to a string at this time. loadData() to load user data stored locally, and decodeData() to create UserData objects with data stored as strings when loaded. saveBitmapToJpeg() to save user profile image in bitmap format locally. loadImageToBitmap() to load a locally stored user profile image. getResizedImage(), which lowers the resolution to an appropriate target value, to solve the problem of taking too long to send and receive image files selected when recording feeds or setting user profiles with the server. Finally, when receiving user data from Firebase, there is setdouble() that checks the variable type of the received numeric value and converts it appropriately.

b. Walk log data : In the case of walking record data, there is no reason to upload and store it on the server before the user creates a feed, so the data is managed locally. In the case of walking record data, it is stored in the walkingmate folder in the directory dedicated to the application.

Class FeedData is used for managing walking record data. The field consists of a collection of coordinates constituting the route the user has passed, a collection of markers added by the user, start and end times, number of steps, and distance traveled. The implemented method is scanFeedList() to scan the walking record file stored locally. savefeed() to save the walking record data locally and encodeFeed() to convert the FeedData object into a string at this time. loadfeed() to load the walking record file stored locally and decodeFeed() to convert the string stored as a file into a FeedData object at this time. Finally, there is deletefeed() to delete the walk log file.

c. Chat data : Chat rooms and chat data are loaded while the application is running. If application try to load the entire chat room and chat data from Firebase every time users open the application, it will take too long. Because of this, chat data is stored locally, and when the application is turned on, the chat data stored locally is loaded, and then the chat data from that point onwards is loaded from Firebase, reducing the time to load the chat room. At this time, if the chat data is stored in the messages folder in the directory dedicated to the app, files containing information about the chat room with participating users and files containing exchanged messages are saved for each chat room. In addition, the profile images of the chatting participants are also stored locally, so that the profile images of each user can be displayed quickly in the chat room.

Class ChatRoom composes a chat room. The fields of this class include roomid to identify the chat room, roomname, the name of the chat room, userids in

the form of hashmap with participant id as key, participation status as value, comments in hashmap form with arbitrary key for identification and Comment object as value. there is Comment is a class created inside Chatroom to organize message information. It consists of three fields: msg (message content), time (message writing time), and userid (id of message creator). Chatting information is stored in two files for each chat room, with participants, roomid, and roomname, which are chat room information, in one file, and exchanged messages in one file. If the chat room information is changed as the chat data is received from Firebase, the local chat room data is changed through overwriting, and the message data is overwritten with the chat data received from the basic message file using FileWriter's overwriting function.

2.5 Challenges

a. Walking Tracking : Recording the route and calculating the distance traveled: In the case of the NAVER Map API, there is no function to record the route traveled and calculate the distance traveled. As a solution, the movement distance was calculated by collecting the corresponding coordinates whenever the user's location coordinates change, calculating the distance from the immediately preceding coordinate each time, and adding it to the total movement distance so far. In the case of the moved route, the collected coordinates were linked to PathOverLay, an object responsible for drawing the route line, on Naver Map.

b. The problem of walking recording stopping when user moves out of the walking recording screen : If the user starts recording walking and move away from the screen, the life cycle of the recording activity passes Onpause() and pauses. This is a problem that must be solved because it is not always possible to turn on the walk record during the walk record. As a solution, we created a service that can perform the walking recording function in the background. As soon as the walk record activity goes to Onpause(), a service that can continue to perform the walk record function is executed, and the activity is restarted with OnResume() to start the walk record. made to keep For convenience of development, some functions were implemented to be always executed as services, and the collection of location coordinates and calculation of movement distance were implemented so that services and activities complement each other.

c. The problem that the Decorator of MaterialCalendarView is not updated : Depending on the walking record in the feed or the created feed, the decorator is applied for each date in the calendar. However, there was a problem that it was not properly applied when the date collection to be decorated with Decorator was updated after modifying the data in the feed list or walking record list in the calendar. As a solution, the activity lifecycle was used. As soon as we enter the list to edit the feed or walking record data, the activity where the calendar is located goes to OnPause(), and when we return to the

calendar, OnResume() is executed. Using this, in OnResume(), all Decorators applied to the calendar are released and re-applied again so that the updated data is normally applied to the calendar.

d. The problem that the fragments that make up the main screen are refreshed every time : When we click the item of the navigation bar located at the bottom of the activity corresponding to the first screen when the application is running, a fragment from the container that will open the screen instead of starting a new activity It is implemented in a way that replaces and floats. The problem is that when a new fragment is created and floated in the container, the fragment is refreshed every time. As a solution, the problem of being refreshed every time was solved by creating a new fragment only when it is executed for the first time, showing only the fragment to be shown after that with show() and hiding the rest using hide().

3 Implementation

3.1 UI/UX Design

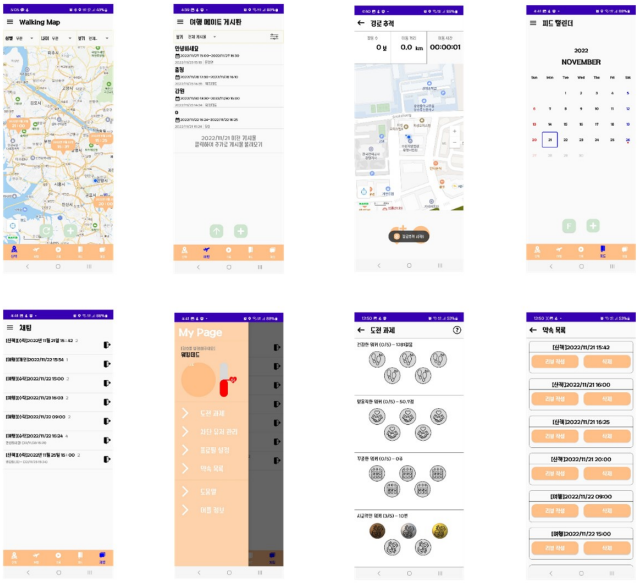

Fig. 3. application test

The bottom navigation bar is a screen that is fixed when using the walking, traveling, feed, and chat menus, except for the record button in the middle. The list and map screen by menu are organized in the Fragment method and are separated from the navigation bar. In the case of My Page and setting options, it was implemented as a side bar without leaving a separate screen, and it consisted of personal data and setting menus such as achievements and blocked user management. And it is set to appear when you click the icon at the top left of the bulletin board, chat list, and feed calendar screen.

3.2 Application

Android app has UI presets based on the user's accumulated experiences and is implemented to ask the user for consent when installing unauthorized apps or using certain sensors and privacy-related parts. For example, when installing an app for the first time, check whether the app has been verified in Google Protect first, and if it is not verified, a window will appear asking if you agree to the installation. And all apps ask for the user's consent for what needs permission, such as alarms and file system usage after installation. We'll replace the example UI preset with a photo. So, our app requires consent to measure physical activity, collect location information, and media files, and supports step counting, distance traveled, upload feed posts, create appointment posts, and chat appointments.

3.3 Database

Our app is a service that requires users' personal data, we have adopted NAVER Login to authenticate users. Therefore, we simplified the process of verifying individuals and set their profile picture and nickname within the app so that they can use different values from the data of Naver account. As a system that handles data, we used NoSQL-based Firebase instead of a database using SQL. Therefore, we processed the posts on the board, the data of the chat system, and the data of My Page in real time. However, if you communicate with Firebase every time you process data, there is a problem that you cannot receive new data when Firebase stops, and there is a disadvantage that the speed slows down when proceeding at the same time, so in the case of user profile data, we first saved it as data in the app and sent it to Firebase. In addition, in the case of the bulletin board screen, it takes a lot of time to be printed on the screen because all the data stored in the collection is read from the online storage, so in the case of appointment posts that are outdated based on the current time, the data is not read unless the user requests it.

Due to the nature of Firebase, it is easy to access data with only permissions, and to compensate for this, each database store has a system called security rules. This system can subdivide the access rights of the data not only in READ/WRITE units, but also in detail such as GET/CREATE/UPDATE/DELETE and subdivide the authority to handle data. For example, in the case of travel

posts, since all authenticated users must write and read, you can make CREATE and READ permissions available only to authorized users, and you can set UPDATE and DELETE permissions to only the post author so that only the post author can edit and delete. In the case of such a security system, there is one caveat: you must use the authentication function among the functions provided by Firebase to fully utilize it. Currently, the authentication function provided by Firebase can be easily used by linking with login media commonly used around the world such as Google, Facebook, GitHub, and text authentication, but in the case of login functions that are used in Korea such as Naver and Kakao, additional work must be done to use them in conjunction with the authentication function.

4 Evaluation

4.1 Alpha Test

We conducted alpha and beta tests to test the function of 'Walking Mate'. In the alpha test, as shown in Table 1, the following functions were checked.

Looking at the functions that failed, in item 21, GPS works bad indoors, so it was not accurately tracked. And, in item 23, GPS works bad indoors, so it was not accurately recorded. Lastly, in item 34, the application ends unintentionally in 1 out of 4 attempts.

4.2 Beta Test

The tester is one male and one female in their 20s. The test was conducted in the form of free use of the application without prior explanation of the application and receiving feedback. Based on the feedback, the direction of development is summarized. We studied the direction of development focusing on new perspectives, except for the parts that have already been said and organized in the planning stage.

UI improvements : There have been some feedbacks that there is a confusion in UI, which can be easily solved through UI improvement.

Dealing with malicious users : The male tester was worried about malicious users, but this part was difficult to solve functionally, so we decided to actively promote the reporting system. Also, it is necessary to prevent acts such as applying for a mate as a mischief or writing a post in a completely different area. This can be solved by developing a function that limits writing and application based on location data only for walking map.

Table 1. Evaluation Table. This is a table showing the results of the functional test of the application. In the column 'Result' , 'P' means pass and 'F' means failure.

Major Classification	No.	Content	Result
Start & My page	1	Register and log in.	P
	2	Delete the app cache and log in.	P
	3	Change my profile.	P
	4	Check the achievement.	P
	5	Check the appointment list.	P
	6	Check the block list.	P
Walking Map	7	Create a new pin.	P
	8	Send requests to each pin.	P
	9	Accept requests and check the appointment list and the chat room.	P
	10	Test if pins are visible after blocking.	P
	11	Test if pins are visible after unblocking.	P
Travel List	12	Check various filtering.	P
	13	Check my posts.	P
	14	Send requests to each post.	P
	15	Accept requests and check the appointment list and the chat room.	P
	16	Test if posts are visible after blocking.	P
	17	Test if posts are visible after unblocking	P
	18	Delete a post and check the list.	P
	19	Delete a post while someone is in the post.	P
Tracing	20	Trace the route outdoors.	P
	21	Trace the route indoors.	F
	22	Record the walking distance outdoors.	P
	23	Record the walking distance indoors.	F
	24	Record the number of walks.	P
	25	Record the time.	P
Feed	26	Write the tracing record.	P
	27	Check if a user can see other users' published feeds.	P
	28	Check if a user can see other users' private feeds.	P
	29	Change the published/private mode.	P
	30	Check if a user can see other users' published feeds.	P
	31	Check if a user can see other users' private feeds.	P
	32	Click on deleted feeds.	P
Server Connection	33	Check status after 30 minutes with the app running in the background.	P
	34	Check status after 30 minutes with the tracing function on.	F

Add comment function in walking map : In the case of the walking map, the map is complex already, so UI was made as simple as possible. But it would be good to have such a comment function. Therefore, adding comments with a short character limit will be better.

Hide the tracing record : This is the most serious issue. It will be a huge burden for users to show the feed containing the walking records to anyone. Walking tracking is likely to start from the actual house, so it must be fixed for security. To solve this problem, the tester mentioned that it would be nice to have a friend add-on, but we intentionally deleted the friend add-on because we were worried that the purpose of the application would change. Instead, giving users the option to hide the map in their feed will make the app safer for them.

5 Limitation and Discussion

Login : In the initial plan for the login function, it was said that it will use not only Naver Login, but also social login, which is used in many apps such as Google and Kakao. However, because the age-related part of personal data is mandatory, Google was excluded, which does not collect this part at all, and Kakao, which provides age-related personal data, was excluded because there were too many materials that it wanted to receive as required data. And NAVER Login also has the disadvantage that only people registered in the NAVER Login system can log in. we submitted the necessary documents to Naver to resolve this part, but we was rejected and could not solve it.

Security Rules : Security Rules Feature When we learned about Firebase security rules, we planned to set permissions o block access to my personal data before deploying it as a test. However, during intra-team testing, all access was blocked due to the permissions we set, so we reset it, and we found that for the security rules to work properly, they need to be linked to the Firebase authentication feature instead of the method they applied. we've tried to solve this part in various ways, but we only need to add a few lines in the code we applied to the current app, and we couldn't find a solution, so we have the permission open now.

Explore Walking Paths : In the 'Walking Mate', Tmap was used for the walking path search function. The regions that support path search in Tmap are limited. Our app, which is dependent of Tmap, is also capable of path search only for limited regions.

6 Conclusion

This application combines the services of the existing neighborhood mate matching application and travel mate finding application with "walking" to provide services for those who want to accompany. In addition, you can remember

or recall your experiences through walking tracking records, a new diary format that has not existed before, and include various functions for safety and motivation to use the application.

References

1. [Firebase] https://firebase.google.com/docs/. Last accessed 27 Nov 2022
2. [Naver Dynamic Map API] https://navermaps.github.io/android-map-sdk/guide-ko/0.html. Last accessed 27 Nov 2022
3. [Naver Geocoding API] https://api.ncloud-docs.com/docs/ai-naver-mapsgeocoding. Last accessed 27 Nov 2022
4. [Naver Reverse-Geocoding API] https://api.ncloud-docs.com/docs/ai-naver-mapsreversegeocoding. Last accessed 27 Nov 2022
5. [kakao REST API] https://developers.kakao.com/docs/latest/ko/local/dev-guide. Last accessed 27 Nov 2022
6. [SK open API] https://skopenapi.readme.io/reference/. Last accessed 27 Nov 2022
7. [Material-Calendar-View] https://github.com/Applandeo/Material-Calendar-View. Last accessed 27 Nov 2022
8. [Navigation-bar-view]https://github.com/S64/android-navigation-bar-view. Last accessed 27 Nov 2022