AI-JANSORI

*Software Engineering, Artificial Intelligence and Application

1st Kim Nayoung
Department of Information Systems
Hanyang University
Seoul, South Korea
nykongkong@hanyang.ac.kr

2nd Heo Yunseo
Department of Information Systems
Hanyang University
Seoul, South Korea
zizizic103@gmail.com

3rd Hwang Yuna
Department of Information Systems
Hanyang University
Souel, South Korea
hyn4008@naver.com

***Abstract*—"AI Manager" is a nagging service for single-person household for better and safer life. Our service has an advantage to smoothly connect to the home appliance. The service generates nagging content based on user-recorded data for a day. Nagging categories are largely composed of outside and inside nagging. In outside nagging contents, speaker informs odd weather conditions or congestion information about user's destination. In inside nagging contents, service generates nagging based on the process of user's goal related to the excercise, meal and cleaning.**

***Index Terms*—AI Speaker, Mobile Application**

Role Assignment

Role	***Name***	***Task Description***
User/Customer	All	Since it is a software developed for personal use, there is a very high probability that the user and the costumer are the same. They buy products and actually use them in their daily lives.Therefore, it is necessary to accurately understand what their needs are, and analyze what can stimulate their desire to buy, market conditions, and competitive products. You should also receive user feedback to update the product periodically and consider new products.
Server Developer	Nayoug Kim, Yunseo Heo	Server development is a technology that manages servers or databases, an area that users of applications cannot see, and actually manages data or operates servers so that clients can provide the information they want. Therefore, server developers engage in various development activities such as system component work, API creation, library creation, and database integration.
Client Developer	Yuna Hwang	Client developers should implement a simple, intuitive interface to make our services more convenient for users. In particular, in developing an app, it is very important to accurately recognize what gestures the user makes because not only the screen shown to the user but also all gestures are related to the functions in the app.
AI Developer	Nayoug Kim, Yunseo Heo	AI developer program systems that will change to suit the business's needs based on the data collected and analyzed. In this service, they will make AI to recommend based on users' past data. They design, develop, implement, and monitor AI systems and put efforts on data ingest and data transformation architecture.
Development Manager	Yuna Hwang	The development manager manages overall project. Initially, he sets goals for this project, and plans accordingly. Then, he supervises to lead the project to success, such as whether the project is moving well toward its goal and whether each team member is performing his or her duties well. From the beginning to the end of the project, the development manager should continue to consider and strive to meet the needs of clients. In addition, development managers should help ensure smooth communication within the team.

I. Introduction

A. Motivation

Recently, the number of senior single-person households is increasing rapidly. In the past, the elderly received support from their children who grew up. There were many large families, but as they became more and more small families, many parents were left alone after their children became independent. Also as the elderly divorce became more frequent, The number of elderly people living alone is increasing not only for bereavement but also for other reasons.
See you. Right before I left the door after saying goodbye, my mom said, "It's raining today. Take your umbrella,". It's sunny outside now and it doesn't look like it's going to rain at all. But suddenly, the sky becomes cloudy and it starts to rain. I regret that I should have listened to my mother. Everyone will have a similar experience at least once. Our team wants to create an AI Mom that nags single-person households instead of their mothers with the motif of conversation with their mothers in front of the door before going out.
In addition, most people who live alone do not eat well because they are lazy. AI Mom, which links SK's AI speaker NUGU with LG's various smart home appliances aims to improve the

healthy life of single-person households by managing meals, health, and safety like a mother.

B. Problem statement(user's needs)

- According to Statistics Korea, the proportion of single-person households is steadily increasing from 27.2

- As of 2021, the number of deaths without relatives was 3,488 and the number has been continuously increased. As a result, the lonely death of the elderly and young people is emerging as another social problem.

- Lack of social infrastructure to care for single-person households that have no relatives or disconnected from society

- People who live alone often feel lonely

- People who live alone need someone to take care of them.

- Users can feel less pressure if there is someone to help them while they have to deal with everything by themselves.

C. Research on related software

1) NAVER Clova Application & Skill Store
   Clova is an Ai technology developed by Naver, and Clova-based software includes skills that can be added to the 'Clova application' and artificial intelligence speakers. The Clova app has 'Life Assistant' and 'Smart Home Control' as its main functions. Life Assistant has detailed functions such as schedule management, music selection recommendation, and voice search, and it also reads fairy tales or plays children's songs. In the smart home function, various IoT products are linked with Clova to control lighting, reserve home appliances such as vacuum cleaners and air conditioners. Among the skills that can be added to smart speakers, there are nine skills in Health & Fitness, which are related to "J.Ansol" and all of them are parenting-related skills except for 119 reports. In addition, in the lifestyle & art category, all of the impractical functions that companies put for marketing.

2) GiGA Genie application
   GiGA Genie is the most commonly used AI speaker in Korea that developed by KT. Compared to Clova, there are many similar functions that we want to implement. There are relatively various functions such as home training assistance, delivery food information, voice home appliance control, and routine setting. However, think about "whether the user should make a request first" and "whether the product makes a suggestion to me first," the different points is there.

3) Loop - habit maker
   Loop Maker is an app that records people's habits on their phones and notifies them if they don't. It is used to make good habits, or to get rid of bad habits. This is similar to the nagging app's daily nagging service. Similar functions include users turning on their cell phones for recording and receiving notifications when passing the goals they set. However, since the ai nagging service actually tells you what to do through voice, it is expected to be more enforceable. In this app, there is a function that allows users to see if they have followed their routines well in the past, and it would be better to borrow this part and add similar functions to the nagging app.

4) Weather applications
   There are various weather notification apps on the market. Compared to these applications, the advantages of Jansori include voice notification, picking out only unique things(like Cold wave, heat wave, fine dust, dryness warning, etc) and informing them. The above applications also use a lot of information from the Meteorological Agency operated by the country, not their own data, and the data processing mechanism will be similar.

II. REQUIREMENT ANALYSIS

A. Login

Users who have accounts log in by entering their IDs and passwords. Users who do not have an account should sign up for membership.

B. Sign Up

Receive a name, ID, password, and address from the user. When the user presses the sign up button, the entered data are stored in the database through the server. It is then used in the application or for NUGU to provide nagging.

C. Exercise

Get input from the user whether he/she worked out today. It is stored in the database through the server and is then used by NUGU to provide nagging in everyday life. The application offers a variety of exercise videos. When the user selects one video, it is connected to the TV and plays the video on the TV.

D. Meals

The user records through the application every time he/she eat. This is delivered to the server, and the server calculates and stores the time he/she eats and number of eating in the database based on the input data. Using these data, NUGU provides nagging related to meals. In addition, various food images are provided within the application. It aims to stimulate the user's appetite so that the user does not skip a meal.

E. Clean

The user first sets his or her cleaning cycle. And record whether or not the user cleaned through the application every day. All of these data are stored in the database. NUGU provides nagging to clean up when cleaning is not done more than the cleaning cycle set by the user. The application also provides a cleaning list. The user may use this to determine what kind of cleaning he/she has done and what kind of cleaning he/she should do.

F. Appliance Control

In some cases, the home appliance may turn on or off through conversation with NUGU. The user can check the on-off status of his/her home appliances through the application.

G. Safety

1) Weather
   When a weather warning or warning is issued, it nags accordingly. If multiple warnings overlap, they tell you everything. Weather-related data is taken from the Data List — Public Data Portal (data.go.kr).
   Heat wave warning - "We recommend you to stay indoors as much as possible because heat waves are expected."
   Day with UV index above 6 - "Put on sunscreen and go out"
   Cold wave warning - "It's cold wave warning, so dress warmly."
   Typhoon Warning - "Close the windows and refrain from going out as much as possible"
   Fine dust warning - "Please wear a mask because fine dust is severe"

2) Congestion
   pre-designated nagging is required for Christmas, Halloween, New Year's Day, and famous festivals that can attract crowds. Representatively, it tells the place where the crowd gathers every year. Myeong-dong on Christmas, Itaewon on Halloween, and Bosingak on New Year's Day were set as dangerous areas. The festival, which will attract crowds, was set up as Yeouido Fireworks Festival and Water Bomb, and the date and location information can be changed every year.

H. NUGU API

1) Physical Care
   Ask at the set time that if user exercised or not. And if user give an answer that didn't do the exercise, tell user to do it. If it can be linked to smart home appliances such as smart TVs or ThinQ application, it can turn on the connected TV to show the stored YouTube home training videos, and turn on air conditioners or music to set up in a comfortable state to exercise.

2) Eating Life
   when there was a person at home and there was no opening and closing the refrigerator on time, J.Ansol will asks "Did you have a meal?". If the answer comes back that user didn't eat yet, it will encourage users to eat at on time. And if you have difficulty choosing a menu, she will recommend a menu. Menu recommendations are made according to general preferences, weather, and learned user's tastes.

III. DEVELOPMENT ENVIRONMENT

A. Choice of software development platform

1) Which platform and why?
   Our team is planning to use both Linux and Windows. When developing, Linux is usually preferred to Windows as a development environment. Linux provides a rich software development environment first. Linux provides most programming languages, so that user can develop programs regardless of tools. In addition, user can download the program comfortably with the "sudo apt-get install". That's why updates and management are so useful. User does not have to install programs like Windows. Finally, it is a free operating system that has a cost advantage. Nevertheless, we will also use Windows because it is a more familiar OS for our team members. We think there will be no problem because various tools are provided for Windows. If there is a problem, we will build and use a Linux environment using a virtual machine.

2) Which programming language and why?

Tool and Language	***Reason***
Java Script XML	JavaScript is an object-based language often used in web development. We decided to use JavaScript XML, an extended grammar of JavaScript, to create the app. It is convenient to use HTML as well as things that can be done with existing JavaScript. In addition, when working on UI, the code can be understood more intuitively, so it is easy to work. In addition, the length of the code can be reduced compared to when using the existing js.
Python	Python was ranked No. 1 as the most popular programming language as of October 2022, according to the TIOBE Index. It is a highly accessible programming language because of the simple code and grammar. It also has rich machine learning libraries and frameworks such as pandas, scikit-learn, and TensorFlow. It also performs detailed memory management for programmers. In addition, the programmer community using Python is also active, making it easy to share code or solve problems.

3) Provide clear information about your development environment:
   - Windows 10
     - 2.90GHz Intel Core i7
     - 16GB RAM
     - 2.30GHz Intel Pentium
     - 4GB RAM
   - Windows 11 Home
     - 1.80GHz Intel Core i5
     - 8GM RAM
   - Visual Studio Code 1.72.2
   - Git 2.30.0

B. *Software in use*

1) *React Native*: react is a library for web development originally developed by Facebook. react native is Facebook's open-source framework that extends such a reaction approach to mobile. It features a cross platform that can produce both iOS and Android apps. Mobile JavaScript was a method of building an interface through webview, but React Native displays it as a native widget. It uses JavaScript or JavaScript xml.

Fig. 1. React Native

2) *Expo*: Expo is a framework that allows programs being developed to be executed on real devices. If a user writes code based on React-native, it's easy and fast to test it. Both Android and iOS can be built using Expo.

Fig. 2. Expo

3) *DJANGO*: Django is an open-source web application framework written in Python, based on certain rules according to MTV (Model, Template, View). There are many users, so there are various problem solutions on the Internet, and it is easy for beginners to handle. Since django is a full-stack framework, we developed a model and view that handles the control flow and logic of the application, except for the template part that is in charge of the user's view.

Fig. 3. Django

4) *POSTMAN*: Postman is a framework for testing developed APIs. If you enter the HTTP method and url and send a request, you can check the response you received immediately. You can put your APIs in writing and share it with others as well as test it.

Fig. 4. Postman

5) *Git, GitHub*: GitHub is Microsoft's web service that hosts source code and provides collaborative supports based on Git, a distributed version control software. In addition, Git provides remote storage management and issue trackers for collaboration as well as code hosting service. GitHub is a necessary tool for development collaboration with fork function and full request function.

Fig. 5. Git, GitHub

6) *LaTeX*: LaTeX is a document creation tool, and has the advantage of being easy to enter page numbers, footnotes, formulas, graphs, and tables. Also, even if the size of the document grows, it uses computer resources rather than words or '한글'. Version management is possible because it is written as a text file, not a binary.

Fig. 6. LaTeX

7) *OVERLEAF*: Overleaf is a tool that allows you to share and jointly edit documents. You can check the results immediately through compilation while writing a document, and it automatically finds errors. LaTex

documents can be easily written through Overleaf.

Fig. 7. Overleaf

8) *NOTION*: NOTION is one of the workspace for the team. It has the advantage of being able to share the progress of the project or manage documents scattered in several places in one place. So, we can immediately understand the workflow of each team member. In addition, it is a fun service to use because users can customize it as they want. Various templates are also being shared.

Fig. 8. NOTION

9) *ZOOM*: Since it's difficult to meet all together, we plan to hold meetings online using zoom. Zoom has the advantage of being able to have meetings whenever and wherever the Internet is connected and therefore we can also use spare time to talk.

Fig. 9. ZOOM

10) *Kakao Oven*: Kakao Oven is a prototyping tool that allows users to design UI and UX before actually developing apps. By visualizing the app in advance, we can develop the app more specifically and systematically. Unlike other prototyping tools, Kakao Oven is free and it is simple and all in Korean, so we thought it would be suitable for our team members who have no experience in development.

11) *Google Docs*: Google Docs is a cloud service that allows multiple people to work with MS Words on the Internet. It can be useful for writing minutes or recording details of ideas and services we want to share while working on a team project.

Fig. 10. Google Docs

12) *ngrok*: ngrock is a platform that enables NAT and its in-house local servers behind the firewall to be exposed to the public Internet through a secure tunnel. In the firewall, servers running locally can be accessed from the outside without changing network environment settings such as port forwarding. We used to this program to generate backend proxy url at NUGU Developers.

Fig. 11. ngrok

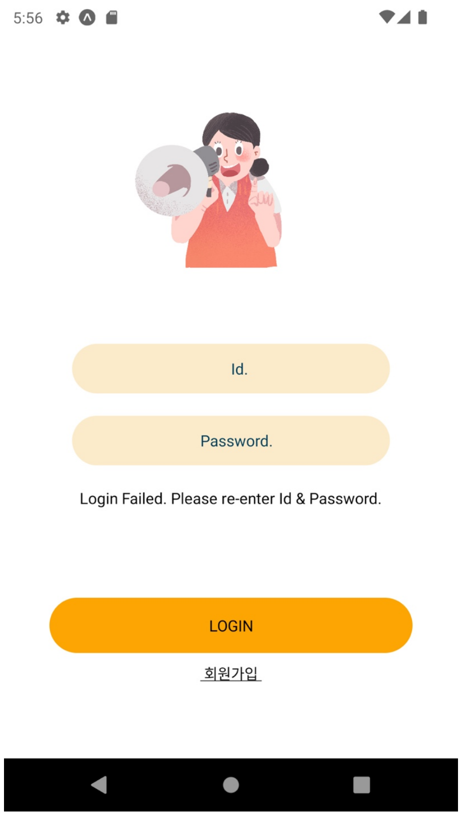
C. Task distribution

Name	***Task Distribution***
Kim Nayoung	Designed NUGU communication Process and NUGU Backend Proxy. Get user data from database. Connect to the NUGU Candle.
Heo Yunseo	Get data from public API, Design mobile application server with Django Rest API, Manage DB
Hwang Yuna	Design and build the User Interface. Send data generated on the mobile application to the server.

IV. SPECIFICATIONS

A. Login page

Log in after entering the ID and password. If the input ID and password match the data in the database when the user presses the login button, move to the main page automatically. Otherwise, if ID and password do not match, the user will be followed by the line 'Login failed. Please re-enter.' Below the login button, there is a sign up button that allows the user to go to the membership page.

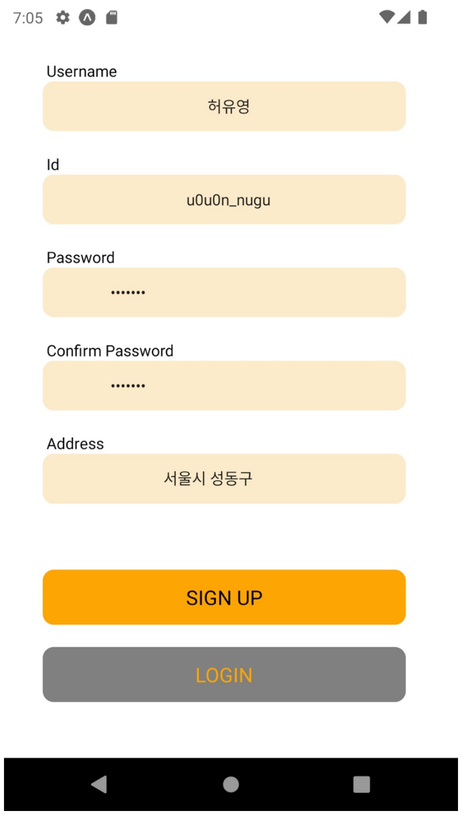


Login Failed. Please re-enter Id & Password.

Fig. 12. Login Page

B. Sign Up

The user enter his/her name, ID, password, password confirmation, and address in order. The ID must consist of at least two characters, including English and numbers. The password must consist of at least eight characters, including English, numbers, and special characters. The value entered in password and password confirmation must be the same. The entered address value is designated as OO시, XX구, **동 so that receives the same type of value from all users. If the user succeeds in signing up when pressing the sign up button, the user is followed by the line 'welcome for becoming a member of Jansori.' At this time, the input data are stored in the database. The name entered when signing up becomes the nickname that the application refers to the user, and the ID and password are used to log in on the application. In addition, when data is delivered to the database on each page, the ID is delivered together to distinguish which of the various users in the database pressed the button. Address is used when NUGU provides nagging about the weather. The user can return to the login page by pressing the login button below.



Username: 허유영
Id: u0u0n_nugu
Password: •••••••
Confirm Password: •••••••
Address: 서울시 성동구

Fig. 13. SignUp page

C. Main Page

At the top, the phrase ”OOO, please record your day!” is displayed. In the middle, there are three yes/no buttons for meal, exercise, and cleaning to receive a value from the user. Each time the user presses a button, deliver the time to the server. If the value has been delivered to the server normally, the line ”saved” is followed. Using these data, NUGU can nag more specifically related to daily life. In the case of meals, if the user doesn't eat properly, NUGU nag. In the case of exercise, if the user has not exercised for more than three days, exercise video is played automatically on TV. In the case of cleaning, nagging is provided when cleaning is not performed more than the cleaning cycle input from the user.

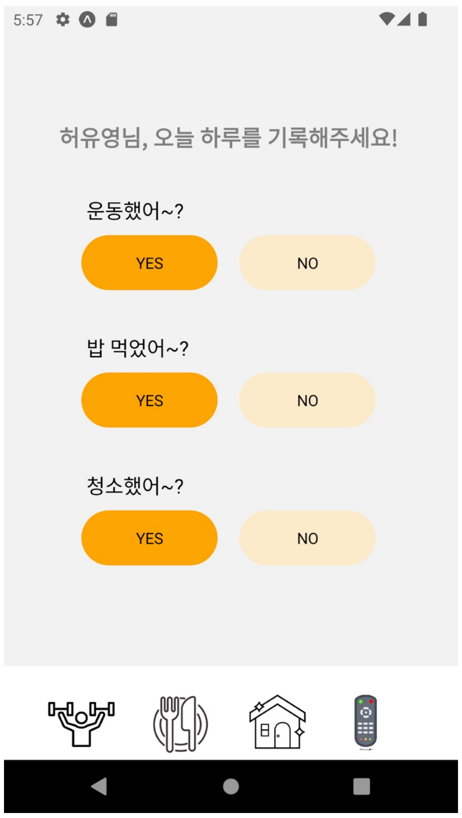


Fig. 14. Main Page

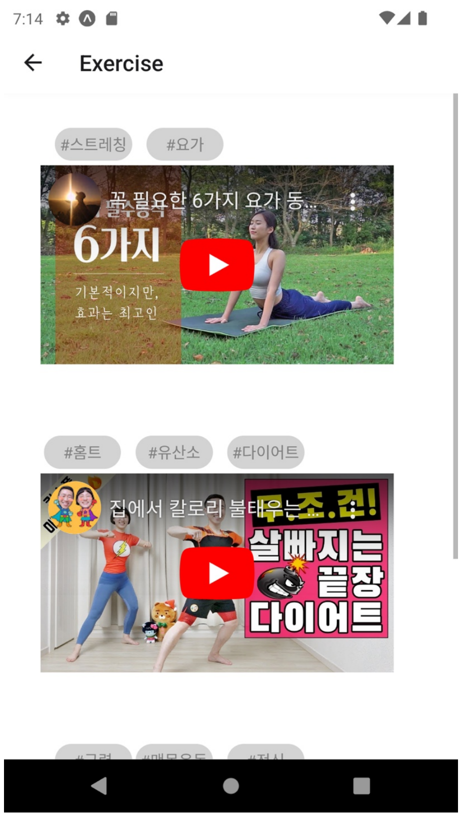


Fig. 15. Exercise Tab

D. Exercise

There are various exercise videos in the application. Most of the functions available on YouTube, such as playing videos, stopping, going forward, going back, speed control, and sound control, are also available within the application. A brief description of the exercise is provided through two to three tags above each video. Through this, the user can more easily select the exercise he/she wants to do. When the user touches the specific exercise video, it is connected to TV and plays it on TV.

E. Meals

There are various food images in the application. Underneath the food, it tells the name of the food. This is to stimulate the user's appetite. It may help the user choose today's menu.

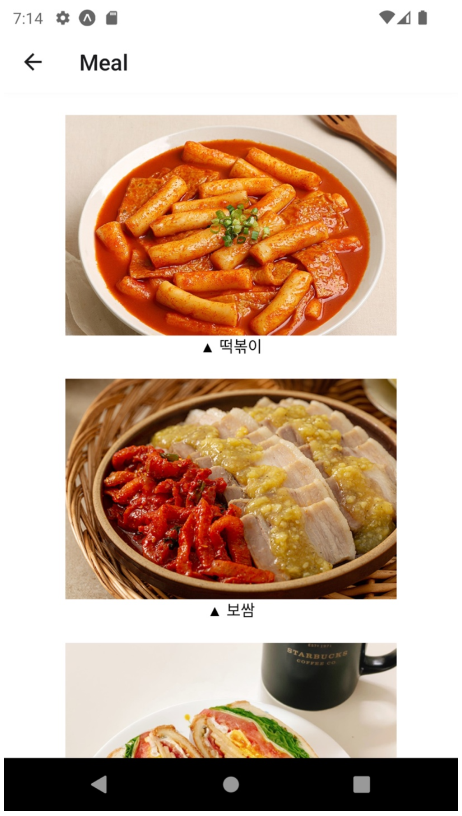
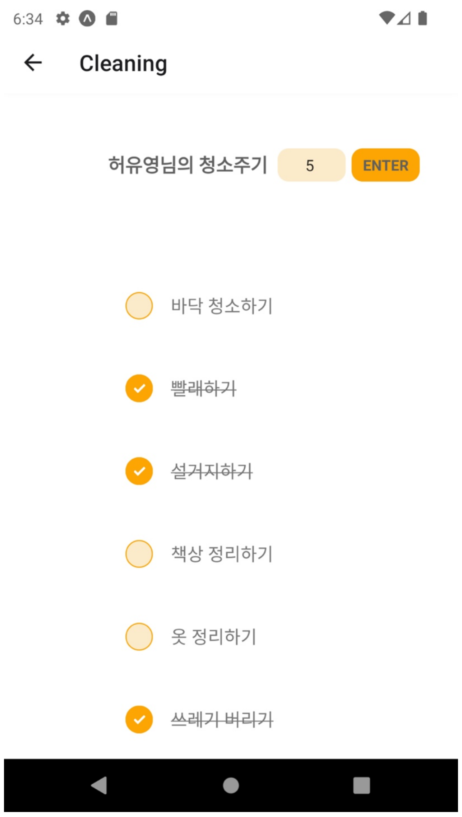


Fig. 16. Meal Tab

F. Cleaning

At the top, the user inputs his/her cleaning cycle. If the user enters a number and press the enter button next to it, the value is passed to the server. When the value is normally delivered to the server, the line "saved" is followed. At the middle is a cleaning list. The user can check the list after cleaning. Through this, the user can check what kind of cleaning he/she needs to do.



바닥 청소하기
✓ ~~빨래하기~~
✓ ~~설거지하기~~
책상 정리하기
옷 정리하기
✓ ~~쓰레기 버리기~~

Fig. 17. Cleaning Tab

G. Appliance Control

The application also connects home appliances connected to NUGU. By displaying the serial number of the home appliances connected under each control button, the user can check whether the home appliance is connected. Conversations with NUGU can cause connected appliances to turn on or off. At this time, the user can also check the on-off status of the home appliance in the application.

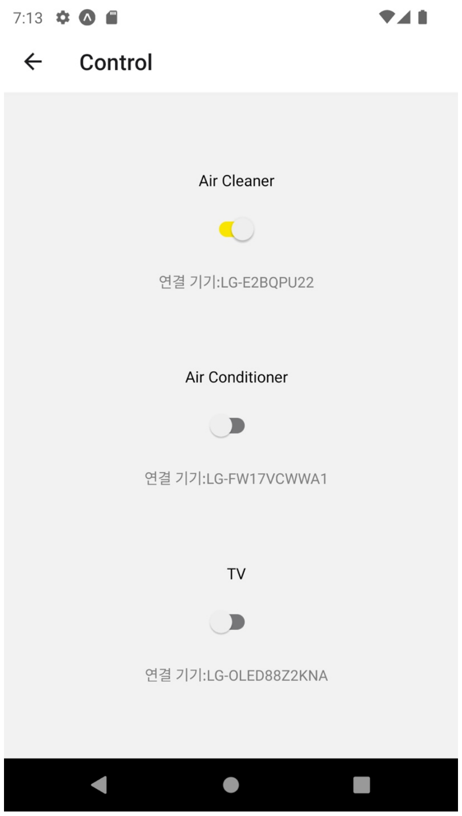


Fig. 18. Appliance Control Tab

H. NUGU API

NUGU Speaker distinguishes services by the name of the play call. So if user wants to call the service, user has to say the play call name with the intent associated with the action that user wants to use. The service's utterance can begin with ”SoRiiJjang.” The reason for ”SoriJjang” is that the service concept of nagging can be expected, and it conforms to the name rule of NUGU play, which states that the stronger the pronunciation of three or more syllables, the better speaker understand. The NUGU Play Builder sets the user's utterance in intents. Therefore, when the user makes an utterance corresponding to the defined intent, speaker answers based on action which is associated with the utterance.
User speech identifies intentions through voice recognition and natural language understanding AI technology provided by NUGU platform, generates responses through actions that fit the intent, and delivers them as synthetic sounds through voice synthesis modules. NUGU device sends a request to ”BASE_URL/”action_name””. For example, if you send a request from answer.congestion, the request url will be https://host:port/nugu/views/answer.congestion. In the backend, the path of the function corresponding to the action must be set as the action name. Below is an image of the architecture when NUGU speaks. When the user speaks, a request is sent to the back-end proxy server in an action that meets the intention of the utterance, and when the response is received, the answer is delivered according to the action.

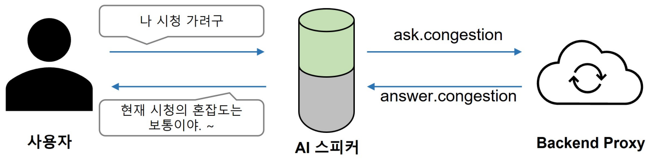


Fig. 19. nugu rest api architecture

→ User's Voice Command

→ NUGU find right intent

→ Send request to Backend Parameter with Utterance Parameter

→ Backend Proxy send REST API response to NUGU

→ NUGU responds back to user

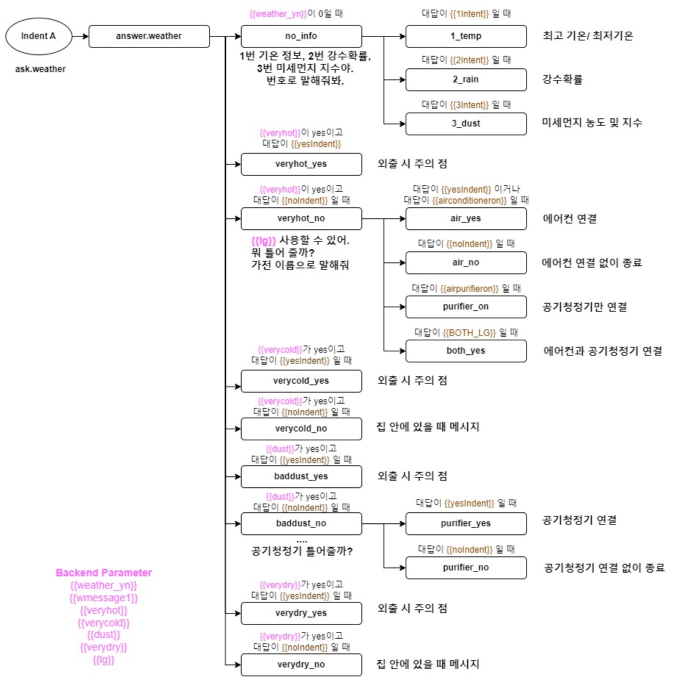
Intent/Action	***User Utterance***	***Response***
ask.weather-answer.weather	오늘 밖에 어때	오늘 날씨 괜찮네. 밖에 나갈거야?
ask.congestion-answer.congestion	나 왕십리 가려구	현재 왕십리역의 혼잡 정도느 여유(이)야. “사람이 몰려 있을 가능성이 낮고 붐빔은 거의 느껴지지 않아요. 도보 이용이 자유로워요.”라고 하네.
ask.athome-answer.athome	나 심심해	운동을 안 한지 벌써 5일째야. 운동 추천 영상 tv에 틀었어. 다른 영상을 보고 싶으면 앱에서 선택할 수 있어. 우리 매일 스트레칭 1분이라도 하자!
jansoriend-answer.end	고마워	좋은 하루 보내. 잔소리 끝!

1) answer.weather
The representative speech to use the service is ”How is the outside?” The resulting message is delivered through the wmessage1 parameter which is pass in json format. This scenario gives an overview of weather-specific information, and the format of the answer varies depending on whether you go out or not. The home appliances available in this scenario are air conditioners and air purifiers.
The types and contents of the backend parameters set

in this scenario can be found in the table below.

Backend parameter	***Parameter contents***
weather_yn	Get an number of special or standout weather condition today
wmessage1	first message for ask.weather
veryhot	“yes” if it’s heat wave
verycold	“yes” if its’ cold wave
dust	“yes” if the fine dust index is bad and very bad
verydry	“yes” if the atmosphere is very dry
lg	LG home appliances that can be used according to weather specifications

It is a logic tree that allows you to check the weather scenario at once.



Indent A
ask.weather
answer.weather
{{weather_yn}}이 0일 때
no_info
1번 기온 정보, 2번 강수확률, 3번 미세먼지 지수야. 번호로 말해줘봐.
대답이 {{1Intent}} 일 때
1_temp
최고 기온/ 최저기온
대답이 {{2Intent}} 일 때
2_rain
강수확률
대답이 {{3Intent}} 일 때
3_dust
미세먼지 농도 및 지수
{{veryhot}}이 yes이고 대답이 {{yesIndent}}
veryhot_yes
외출 시 주의 점
{{veryhot}}이 yes이고 대답이 {{noIndent}} 일 때
veryhot_no
{{lg}} 사용할 수 있어. 뭐 틀어 줄까? 가전 이름으로 말해줘
대답이 {{yesIndent}} 이거나 대답이 {{airconditioneron}} 일 때
air_yes
에어컨 연결
대답이 {{noIndent}} 일 때
air_no
에어컨 연결 없이 종료
대답이 {{airpurifieron}} 일 때
purifier_on
공기청정기만 연결
대답이 {{BOTH_LG}} 일 때
both_yes
에어컨과 공기청정기 연결
{{verycold}}가 yes이고 대답이 {{yesIndent}} 일 때
verycold_yes
외출 시 주의 점
{{verycold}}가 yes이고 대답이 {{noIndent}} 일 때
verycold_no
집 안에 있을 때 메시지
{{dust}}가 yes이고 대답이 {{yesIndent}} 일 때
baddust_yes
외출 시 주의 점
{{dust}}가 yes이고 대답이 {{noIndent}} 일 때
baddust_no
.... 공기청정기 틀어줄까?
대답이 {{yesIndent}} 일 때
purifier_yes
공기청정기 연결
대답이 {{noIndent}} 일 때
purifier_no
공기청정기 연결 없이 종료
{{verydry}}가 yes이고 대답이 {{yesIndent}} 일 때
verydry_yes
외출 시 주의 점
{{verydry}}가 yes이고 대답이 {{noIndent}} 일 때
verydry_no
집 안에 있을 때 메시지
Backend Parameter
{{weather_yn}}
{{wmessage1}}
{{veryhot}}
{{verycold}}
{{dust}}
{{verydry}}
{{lg}}

Fig. 20. answer.weather logic tree

It is also a service that includes several weather specifications. If there are several things to be aware of, it delivers all messages that match the corresponding weather conditions. In addition, if there are several LG appliances available, it checks which appliances the user wants to use and delivers a message that the related appliances are connected.

2) answer.congestion
The representative speech to use the action is ”I’m going to DESTINATION”. Since the user’s destination information is essential for an action to inform the congestion level, so if there is no DESTINATION information, destination will be received through additional questions. The Utterance parameter is DESTINATION and the backend parameter is cmessage.

The congestion level public data for this service is real-time public data in Seoul. Since this data only supports congestion in 50 districts in Seoul, entities that store 50 locations should be created separately. Therefore, I made a custom ENTITY called SEOUL_DATA_PLACE and a synonym corresponding to the representative word was also designated. For example, if the user’s DESTINATION is ”Hongdae,” the representative word ”Hongdae Entrance” is included in the request and delivered, and the server delivers the ”Hongdae Special Tourist Zone” matched to the Hongdae Entrance to the public data API. Invoking the API and receiving congestion information and messages is implemented in get_congets$dest$.

3) answer.athome
The representative utterance of this action is ”I’m bored.” The reason why I decided with this speech was obtained from the experience of using AI speakers. There are quite a few cases where users talk to speakers for no reason, such as ”I’m bored” and ”Play with me,” and this utterance means that the user has nothing to do. When this utterance comes in, the user can be nagged for not cleaning, eating, or exercising. When user did everything, service gives a compliment. The backend parameters required for the action are as follows.

Backend parameter	***Parameter contents***
hemessage	main message for answer.athome
ex_not_day	term that user didn’t exercise

The answer.athome action requires a multi-turn conversation to use the TV. If you haven’t exercised for more than three days, service will automatically connect to TV and show exercise recommendation videos. And less than that, service asks users whether user wants NUGU to play recommendation videos on TV or not. It is a logic tree that can check the answer.athome scenario at once.

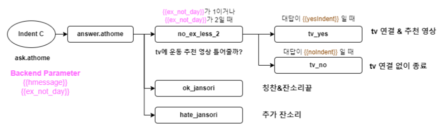


Fig. 21. answer.athome logic tree

4) ask.end
The answer.end action is an action that ends when the user no longer wants to hear nagging. Representative utterances include ”Stop,” ”Thank you,” and ”Okay.” It responds, ”Okay. Stop nagging! Have a nice day.” and informs you that the service has been terminated.

V. Architecture Design & Implementation

A. Overall architecture

Users can use NUGU speaker and app. The NUGU service communicates with the NUGU backend proxy and receives the information needed for the response. Information input by the user through the app communicates with the backend server through REST API. Both the speaker and the app get the information they need from the SQL database. And it can be connected to LG home appliances through the API provided by LG ThinQ.

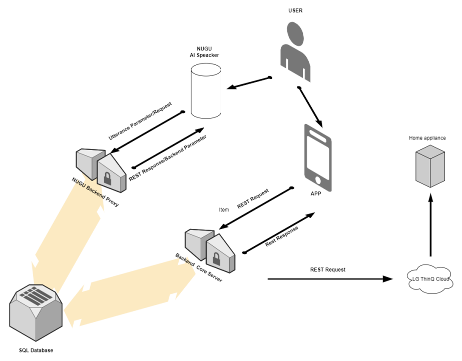


Fig. 22. Overall Architecture

B. Directory organization

Directory	***File Name***	***Module Name***
backend/jansori/jansori	settings.py urls.py	Django - jansori
backend/jansori/app	migrations admin.py apps.py models.py reauirements.txt serializers.py tests.py urls.py views.py	Django - app
backend/jansori/nugu	urls.py base_views.py weather_views.py athome_views.py	Django - nugu
Frontend	app.js app.json	React Native
Frontend/src/bottomTab	Cleaning.js Control.js Exercise.js Meal.js	React Native
Frontend/src/home	Home.js	React Native
Frontend/src/login	Login.js SignUp.js	React Native
Document	jansori_doc.tex	LaTex

C. Module1: Django - jansori

1) Purpose
The server construction and back-end development framework were considered among options such as django, flask, and spring. Considering that the project period is not long and it is the first time to develop the back-end, I tried to choose among Python-based frameworks that I have encountered more often than Java, and I chose django considering that there are many materials to refer to and that I can put various applications in one project. Django is an open-source web application framework written in Python, based on certain rules according to MTV (Model, Template, View). Since django is a full-stack framework, we developed a model and view that handles the control flow and logic of the application, except for the template part that is in charge of the user's view. The purpose of this module is to control the entire nagging project. In Django, a project can be seen as a big service. An app is a unit that divides elements such as functions in a project into a certain standard.

2) Functionality

a) REST API
Rest api was mainly developed using Django. The API (application programming interface) provides features and rules that allow interaction and communication between different applications. REST is an architectural style for designing APIs over the HTTP protocol. The key benefit is greater flexibility. Developers use the REST API wherever they need to provide data to users of web applications or sites directly from the server.

b) ORM
ORM stands for Object-Relational Mapping, which means mapping data between an object and a relational database. It automatically generates SQL based on the relationship between objects, allowing data in the database to be handled without sql query statements. It is a technology that allows tables in a database to be used like a class commonly used in object-oriented programming. This technology is designed to create code that is more efficient and suitable for readability and maintenance beyond manipulating the database by writing existing query statements. We used mySQL and backend servers through ORM.

3) Location of Souce Code
backend/jansori

4) Class Components

a) Jansori
In Jansori, the apps in the project are set up to be included as urls, and the default settings are also saved.

b) app

app is one of the Jansori project's apps, which is related to the back-end server of a mobile application which is for AI speaker's Jansori control and user informations.

c) nugu
nugu is one of the Jansori project's apps which is related to the back-end server of NUGU speaker. In this app, Jansori's response dialog is written based on the information entered by the user in the mobile app.

d) manage.py
The manage.py file is a file used to write various commands related to the Django during the Django project. This file takes a function called 'execute from command line' from the django.core.management module and processes commands that you enter.

D. Module2: Django - app

a) Purpose
This module mainly stores the user's information in a database and allows the user to take the necessary data from nugu's backend server and use it. When a user subscribes to a membership, the information is stored in a database. When you try to log in, make sure your ID and password match. In addition, if you save your target exercise, cleaning, and meal cycle in a mobile application, and save when you did it recently, you can tell me how long you haven't done your job based on it.

b) Functionality
Servers to API endpoint /app/user/
Servers to API endpoint /app/login/
Servers to API endpoint /app/goal/
Servers to API endpoint /app/meal/
Servers to API endpoint /app/exercise/
Servers to API endpoint /app/clean/
Servers to API endpoint /app/connect/
Servers to API endpoint /app/turnon/

c) Location of Source Code
backend/jansori/app

d) Class Components

i) app/migrations
In this repository, files are automatically created to reflect the change in mySQL when a model change occurs as a part of the database connection. When you first finish modeling and do python manage.py makemigrations, an init file is created to create an initial database table. After that, a file that reflects the modification is generated. When python manage.py migrate, the generated file is applied to the DB. It is important to note that you should not modify the table or column on your own in mySQL. Since there is no way to perform a reverse migrate in mySQL, it is difficult to recognize each other and an error occurs. Therefore, it is recommended to modify the mySQL query statement in the warehouse without using it on the workbench.

ii) app/urls.py
It is a file needed for url mapping. You can organize urls in jansori's urls file, but it becomes too complicated to add all the app's functions that way. Therefore, you have urls files for each app, and the front parts you share are added to urls files in the jansori folder. And the other parts in the back are added to urls in the app folder. In this project, /app and /nugu are added to the urls of jansori, and the api which is produces in views.py of the app are mapped in the urls of the app file. Import include to urls.py and create the back of the requested address within urlpatterns = [ ].(In this project's app, 'app/') and include('app.urls') specify the path to the urls file. Using include, it allows you to move quickly by attaching only the path of the app to the front path that is common every time and the path of the app at the back.

iii) app/models.py
This file creates tables for the database. It is a major component of the MTV model, which is a big feature of Django, and can be used without entering the workbench and writing query statements. I wrote a User model that stores basic user information (ID, name, password, location), a Goal model that stores life goals (number of meals per day, exercise cycle, cleaning cycle), a Recent model that stores recent life information (when was the most recent meal, cleaning, exercise) and a HomeConnect model that stores connectivity of appliances. Each table has an id-connected OneToOne structure, and the location information of the User model is used to receive weather information. Goal and Recent are used to check and nag users to make sure they are doing well, exercising, cleaning, and eating. These information is stored in the user's mobile application.

iv) app/serializers.py

The Django application that provides the REST API should be able to exchange JSON data with the application that requested the API. To this end, it is necessary to serialize the DB instance as JSON data, or vice versa. For this purpose, the class provided by Django Rest Framework is Serializer (similar to the form class of the device), which inherits it to define a serializer corresponding to a specific model. Using such a serializer, DB instances can be expressed as JSON data, and DB instances can be created or modified based on JSON data. In the serializer, fields of the model to be serialized or decialized are defined. All fields of the corresponding model may be defined, or only some fields may be defined. The values of the fields defined here are used to represent JSON data during serialization and to create or modify DB instances during serialization. App/serializers.py serializes APIs such as receiving user information and receiving recent life data. Eating time, cleaning, and exercise are individual implementations, so they belong to one table, but they are characterized by using different serializers.

v) app/views.py
The view file is a place to create a restapi that can be requested for DB instances based on the serializer. A basic form of REST API view refers to a view for listing (List), Create, Retrieve, Update, and Delete of a DB instance. App/views.py includes an API that posts user information to DB, an API that stores whether home appliances are connected, and an API that updates user's meal, exercise, and cleaning information.

vi) requirments.txt
It is a collection of name and version information such as modules and libraries needed to produce apps for the project to distribute services with collaborating team members. I organized the lists using freeze.

E. Module3: Django - nugu

1) Purpose
This module is intended to generate nagging to the user. Overall, it generates nagging when there is or is not a special weather condition, how crowded where the user wants to go and what user needs to do to have a better day. This module is used when the user speaks on a speaker and the corresponding action requires a backend.

2) Functionality
Serves to API endpoint /nugu/views/answer.weather
Serves to API endpoint /nugu/views/answer.congestion
Serves to API endpoint /nugu/views/answer.athome

3) Location of Souce Code
backend/jansori/nugu

4) Class Components

a) nugu/urls.py
This code maps the path to find the correct function in the request of the AI speaker. There are many functions that need to be processed by NUGU backend proxy, so views files are divided by function. Therefore, three files must be imported from the views file. NUGU passes the path by attaching an action name to a given backend proxy url. Therefore, the path of each function must be an action name regardless of the function name.

b) nugu/views/base_veiws.py
NUGU Play Builder communicates with the backend server via the Rest API. To respond to the NUGU service, it must be delivered in the form of "version, resultCode, and output". The output includes backend parameters to which the NUGU service should be delivered. Backend parameters required for each NUGU action is directly set by developers. If there is no parameter information in the output received from the backend, it is recognized as Null. When distributing back-end servers, we used an "ngrok" service that temporarily creates url for 1 hour. NUGU requires a health() function for basic status verification. If the health function is called, send "OK" message.

c) nugu/views/weather_views.py
Weather data received API provided by the Korea Meteorological Administration's short-term forecast inquiry service and used the current lowest, highest temperature, precipitation probability, humidity, fine dust concentration and index to create messages. The backend function related to ask.weather can be found in views/weather_views.py. The answer_weather function sends backend parameter values related to answer.weather, which is the most basic in the weather-related scenario. Read weather data in the form of a dictionary to check if there are any weather specifications, and if applicable, specify the appropriate parameter value. Weather data is received by calling weather_data and fine_dust functions. If there is nothing unusual about the weather today, you can ask additional questions about the weather conditions. The information

provided at this time is the highest/lowest temperature, precipitation probability, and fine dust index. Each data is passed in the give_temp function, the give_rain_probability function, and the give_dust function.

d) nugu/views/congestion_views.py
The congestion level data used real-time public data in Seoul. This data provides data on population, road communication, public transportation, environment, and COVID-19 in 50 major visiting areas in Seoul, derived from the Seoul Metropolitan Government's analysis of floating population and demand surveys for related agencies. Real-time population data in Seoul is provided by expanding and correcting the entire real-time population based on signals from LTE and 5G users among those who use KT mobile phones. The main 50 places include 3 ancient palaces and cultural heritages, 7 special tourist zones, 12 parks, 13 developmental commercial districts, and 15 densely populated areas. Since real-time public data APIs can only call one place at a time, it was important to put the area where users go in the form stored in the API. Therefore, exceptions were made so that data could be accessed with a total of 73 words by adding surrounding stations. First, in the answer_congestion(request) function, when a request is received, it first receives a UTERANCE PARAMETER delivered by NUGU and calls get_congets(dest). get_congestion(dest) takes the DESTINATION parameter delivered by NUGU as an argument, calls the API, and returns congestion information and congestion messages. If the user wants to go to places that is not included in 50 places and the surrounding stations, the message "It's a place that can't be retrieved from real-time data" is delivered. Or it generates a message that fits the current congestion. In addition, the action designates Children's Day, Halloween, Christmas, and New Year as special days, providing a warning about places where people may flock on the day regardless of DESTINATION if they ask for congestion on the day. A message that fits a special day is generated by the special_date function.

e) nugu/views/athome_views.py
The most recent data of exercise, eating, and cleaning are received from the app to generate nagging at home. Each data is a datatime type value, and in the case of meals, data on the number of meals per day have also been received. The time when the user exercised or ate or clean, and number of meals user had can be obtained from the app's DB. Then, check if user did it today through the what_user_did function. If user did, value will set as yes. And if not value will be set as no. The msg_for_ex function generates a message for the number of days of the week without exercise. The msg_for_clean function generates a mesh around a cleaning cycle preset by the user. When answer.athome sends a request to the backend server, it receives a request from the answer_athome function. This function comprehensively generates messages according to the situation when the user does all of them, when there is one thing user did not do, when there is two things user did not do, and when there are three things user did not do. And it sends a comprehensive message to NUGU by putting it in hmessage. The longer the user does not clean, exercise, or eat, the longer the message is.

F. *Module4: React Native*

1) Purpose
We used React Native to implement the interface of the app that users encounter. First of all, React Native uses JavaScript as a development language, so you can start developing right away without learning special language for app development. In addition, if you modify the code slightly, the modifications are immediately reflected in the application without rebuilding, making it quick and convenient. Finally, both Android and iOS can be executed with one code.

2) Functionality
In general, React Native is a framework for implementing an app to be displayed on the user's screen. It implements designed apps using various modules and tags. The form and specific shapes such as location, size, and color to be shown on the screen are implemented through tags similar to HTML and style attributes similar to CSS respectively. It is possible to implement not only the app displayed on the screen, but also various functions within the app. For example, you can move between pages using the modules '@react-navigation/native' and '@react-navigation/stack'. In addition, YouTube videos can be played within the app using the 'react-native-youtube-iframe' module.

3) Location of Souce Code
Frontend
Frontend/src/bottomTab
Frontend/src/home
Frontend/src/login

4) Class Components

a) app.js
The application is executed with the execution of this code. The first screen of the application is designated as a login page. In addition to the Login page, pages to be shown in the application are all included. This file allows you to understand the overall structure of the application.

b) app.json
This is a code that describes the setting of Expo, such as name and version.

c) Cleaning.js
This is a code that implements a kind of checklist that can record what kind of cleaning the user has done or not. The cleaning list to be done with the empty circle was listed vertically. If you touch the empty circle, it changes to an orange circle, creating a check mark, and a cancellation line drawn on the cleaning list. Touch again to return to the original empty circle state, which is an indication that cleaning has not been done.

d) Control.js
This is a code that implements an on-off button that shows whether the connected LG home appliances are currently turned on or off. First, when you connect the home appliance with the app, the model name of the connected device appears under the button. When the round button is on the left and its color is gray, it indicates that the home appliance is turned off. Conversely, when the round button is on the right and orange, it indicates that the home appliance is on.

e) Exercise.js
This is a code that implements a list of various exercise videos. It shows you the videos taken from YouTube. Functions such as video play, pause, sound control, and speed control that can be used within YouTube can also be used on the app. On top of the video, a simple description of the exercise is provided through various tags.

f) Meal.js
This is a code that implements a screen showing various food images. Below the image is the name of the food.

g) Home.js
It is a code that implements a screen that can record today's state. Specifically, record the answers to whether a user exercised, ate, or cleaned. Touch the appropriate button among the YES/NO buttons to record the answer. Each time the YES button of each is clicked, a message is displayed to the user that the answer has been saved.

h) Login.js
This is the code that implements the login page. First, the logo is displayed in the middle, and the ID and password are entered from the user through two boxes below it. After the user press the login button, if the ID and password do not match, a message that the login failed is displayed. Under the login button, the word "Sign Up" is written. Press this to go to the page where you can sign up.

i) SignUp.js
Each of the five boxes receives a user's name, ID, password and password confirmation, and address. On top of each box, the name of what value to enter is written And in the box, specific descriptions such as the number of characters and the type of input are written. Press the Sign Up button at the bottom to return to the login page.

VI. Use Cases

A. Turn On The Application

Open the Expo Go and run the AI-Jansori program. AI-Jansori is launched.

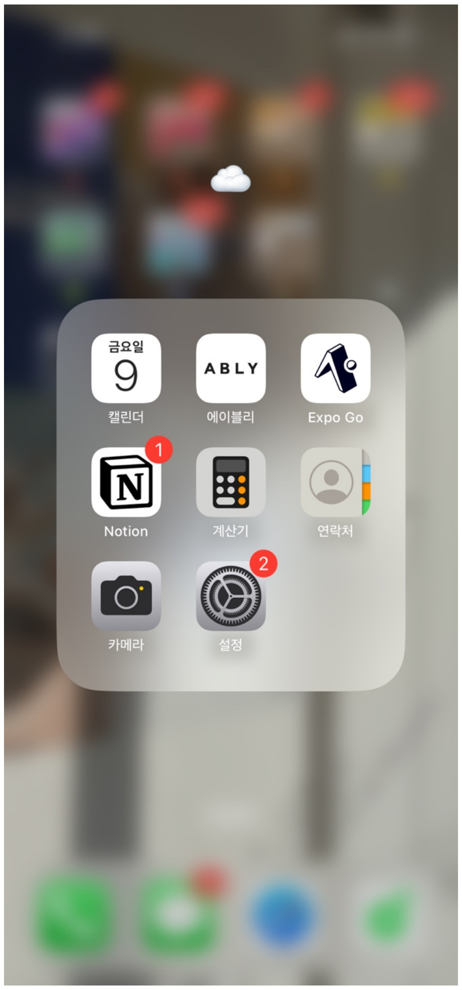
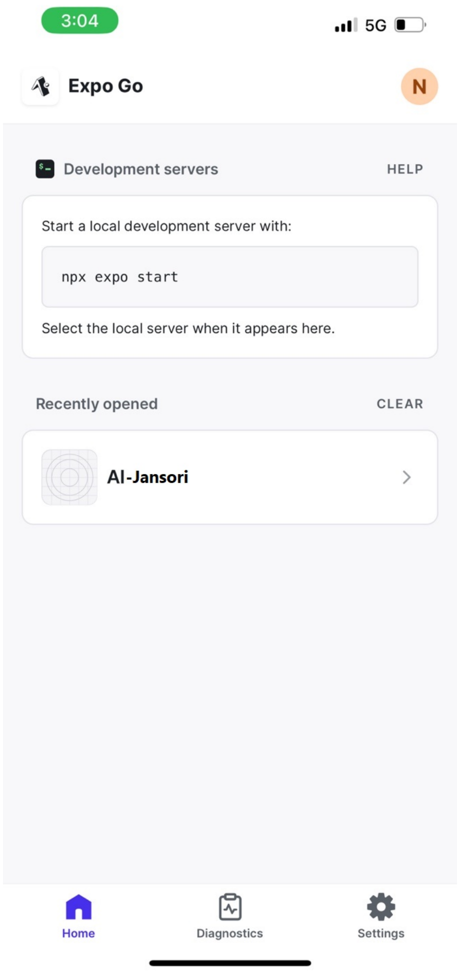


Fig. 23. Expo Go



Development servers HELP

Start a local development server with:

```
npx expo start
```

Select the local server when it appears here.

Recently opened CLEAR

AI-Jansori

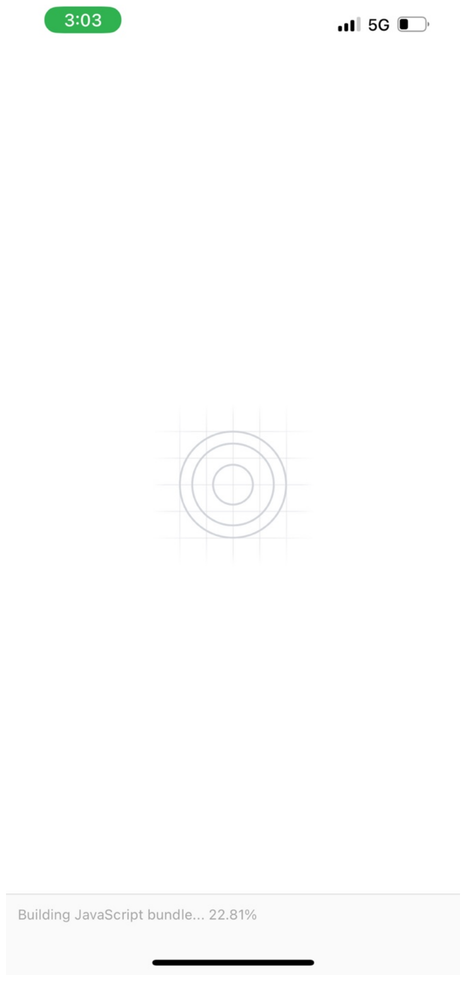
Fig. 24. Program running

B. Loading

When the user launches AI-Jansori, the program is loaded. During loading, the splash icon appears on a white background, which is the default setting of Expo. You can check how much loading has progressed through the bar at the bottom.

1) Wait On The Loading Page When the user waits on the loading page, AI-Jansori's loading ends and all preparations for the execution of AI-Jansori are completed. First, login page appears.

2) Leave The Loading Page If the user leaves the loading page without waiting, the execution of AI-Jansori is terminated. When the user turns on the app again, the program's loading starts again from the beginning.



Building JavaScript bundle... 22.81%

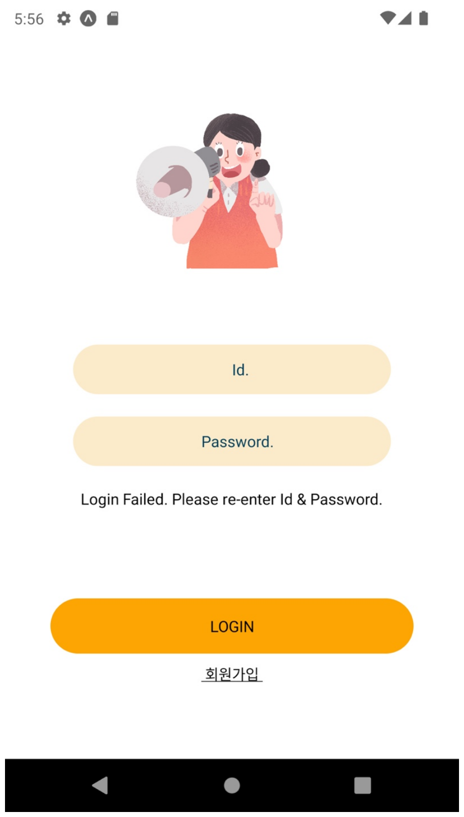
Fig. 25. Loading

C. Login

There is a logo in the middle of the screen and below that, there is a space in which you can enter your ID and password, respectively. Below that, there are a login button and a sign up button.

1) Existing User Log in on the current page. Enter your ID and password and click the login button. Once the login is successfully completed, move to the main page.

2) New User Click sign up under the login button to go to the sign up page. Register for membership on the sign up page.

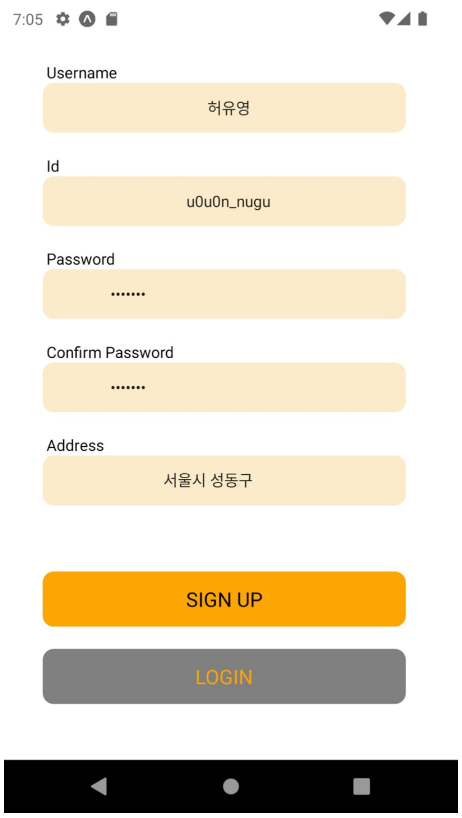


Login Failed. Please re-enter Id & Password.

Fig. 26. Login

D. Sign Up

Enter your name, ID, password, password confirmation, and addresses in order and press the sign up button. If membership registration is successfully completed, you are followed by the line ”Welcome to Jansori”. After that, press the login button under the sign up button to go back to the login page. Log in on the login page.


7:05

Username
허유영

Id
u0u0n_nugu

Password
•••••••

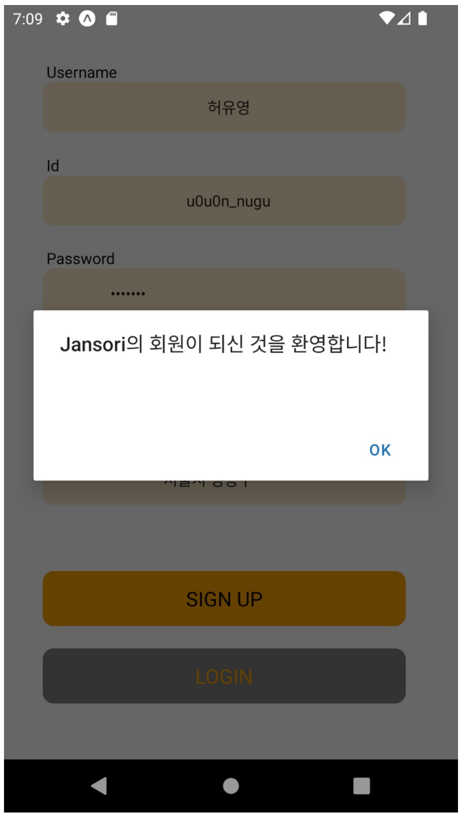
Confirm Password
•••••••

Address
서울시 성동구

SIGN UP

LOGIN

Fig. 27. Sign Up


7:09

Username
허유영

Id
u0u0n_nugu

Password
•••••••

Jansori의 회원이 되신 것을 환영합니다!

OK

SIGN UP

LOGIN

Fig. 28. Sign Up Success

E. Main Page

Record your daily life. This data will be used by NUGU to provide nagging.

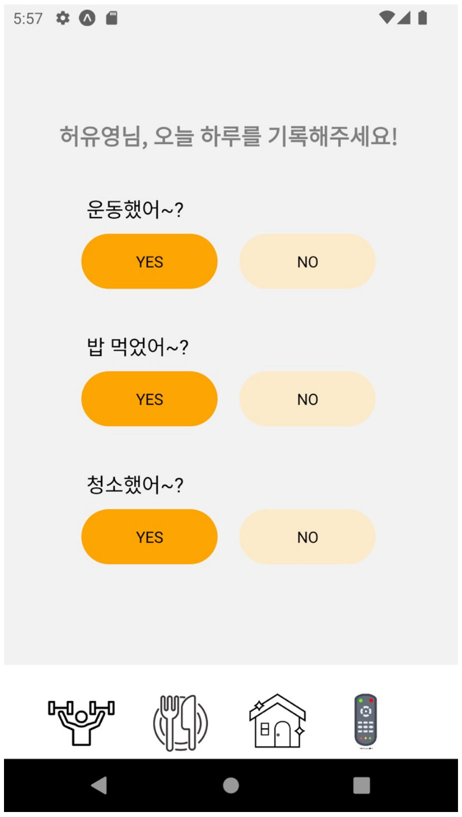


Fig. 29. Main Page

1) Exercise If you exercised when asked if you exercised today, click the yes button. If not, click the no button.
2) Meals Click the yes button every time you eat.
3) Cleaning If you cleaned today, click the yes button when asked if you cleaned. If not, click the no button. If you enter your cleaning cycle on the cleaning tab, you will be nagged to clean.

You can move to the exercise, meal, cleaning, or home appliance control page through the navigation tab at the bottom.

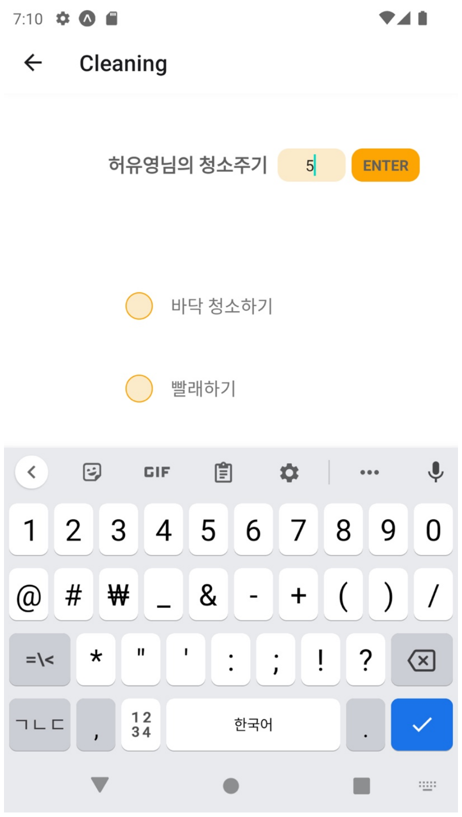
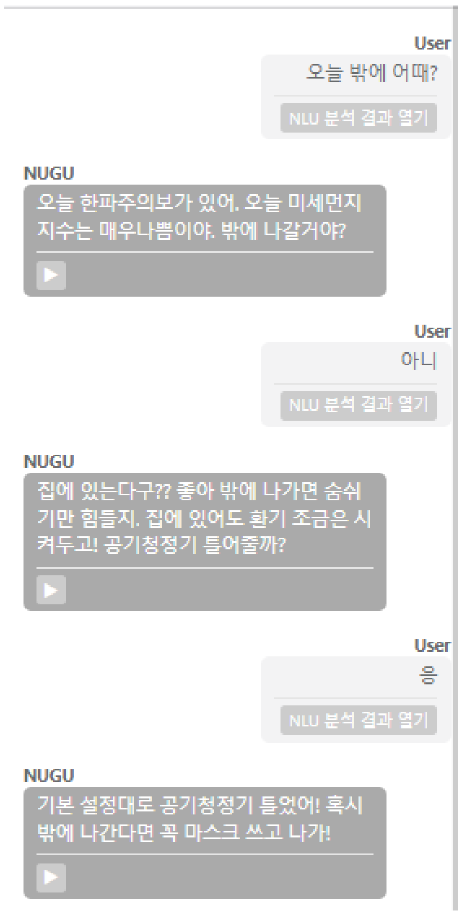


Fig. 30. Set Clean Cycle

F. Ask NUGU "오늘 밖에 어때?"



User
오늘 밖에 어때?

NUGU
오늘 한파주의보가 있어. 오늘 미세먼지 지수는 매우나쁨이야. 밖에 나갈거야?

User
아니

NUGU
집에 있는다구?? 좋아 밖에 나가면 숨쉬기만 힘들지. 집에 있어도 환기 조금은 시켜두고! 공기청정기 틀어줄까?

User
응

NUGU
기본 설정대로 공기청정기 틀었어! 혹시 밖에 나간다면 꼭 마스크 쓰고 나가!

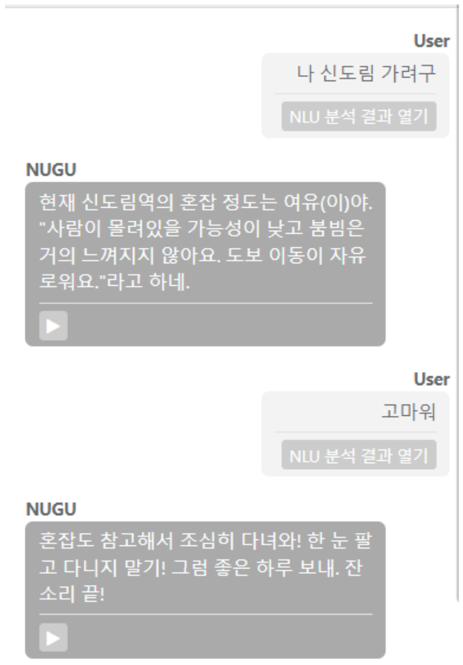
Fig. 31. test screen of answer.weather

It checks whether today's weather includes heat wave warnings, cold wave warnings, dry weather warnings, or fine dust warnings, and delivers appropriate messages. If there is nothing, it say "It's good to go outside," and if there is any special weather condition, it delivers all the corresponding singularities. And ask if you want to go outside in common.

1) Yes
If the user expresses his/her intention to go out, it mainly tells him/her what to be careful about when going out in relation to each weather feature. If there is nothing unusual, you can receive information on the current maximum and minimum temperatures, the probability of raining, and the concentration of fine dust. In the case of heat waves, it nags at you to put on sunscreen and go out, in the case of cold waves, it nags at you to wear warm clothes, and on dry day it nags to drink a lot of water. Also, it nags you to wear a mask well on days when there is a lot of fine dust. And in conclusion, there is a message to recommend not to go outside now or go outside slightly later.

2) No
If the user expresses his/her intention not to leave and if there is a home appliance that can be connected, NUGU asks if user wants to use home appliance. Currently, the types of appliances available in this service are air conditioners and air purifiers. It can be used on days when the concentration of fine dust is high and in the case of heat waves, respectively. It tells you what you can do inside the house if there is no home appliance to connect to. The same is true of cases where you don't want to connect even though you have a home appliance to connect to. For example, if you don't want to connect an air purifier on a day when fine dust is high, you can get nagging that you need to ventilate even if there is a lot of fine dust. The settings for each appliance follow the LG ThinQ app.

G. Ask NUGU "나 OOO 가려구!"



User
나 신도림 가려구

NUGU
현재 신도림역의 혼잡 정도는 여유(이)야. "사람이 몰려있을 가능성이 낮고 붐빔은 거의 느껴지지 않아요. 도보 이동이 자유로워요."라고 하네.

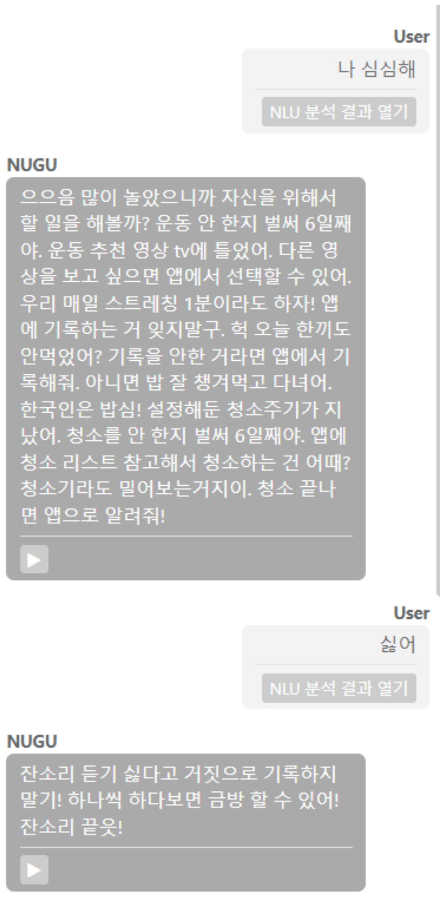
User
고마워

NUGU
혼잡도 참고해서 조심히 다녀와! 한 눈 팔고 다니지 말기! 그럼 좋은 하루 보내. 잔소리 끝!

Fig. 32. test screen of answer.congestion

If you ask a similar question, you can know the real-time congestion level information of the place you want to go. Destination information is unconditionally required to use this service. Therefore, when speaking except for the destination, the destination is received through several questions. If you say an area where real-time information data cannot be received, speaker says that it is an area that cannot be received, and ask them to say it again by the name of the surrounding station or famous place. If it is an area on the list, it nags the current real-time congestion information and corresponding messages. When the user makes a positive speech, it tells you to go safely by referring to the congestion level.

H. *Ask NUGU "나 심심해!"*



User

나 심심해

NUGU

으으음 많이 놀았으니까 자신을 위해서 할 일을 해볼까? 운동 안 한지 벌써 6일째야. 운동 추천 영상 tv에 틀었어. 다른 영상을 보고 싶으면 앱에서 선택할 수 있어. 우리 매일 스트레칭 1분이라도 하자! 앱에 기록하는 거 잊지말구. 헉 오늘 한끼도 안먹었어? 기록을 안한 거라면 앱에서 기록해줘. 아니면 밥 잘 챙겨먹고 다녀어. 한국인은 밥심! 설정해둔 청소주기가 지났어. 청소를 안 한지 벌써 6일째야. 앱에 청소 리스트 참고해서 청소하는 건 어때? 청소기라도 밀어보는거지이. 청소 끝나면 앱으로 알려줘!

User

싫어

NUGU

잔소리 듣기 싫다고 거짓으로 기록하지 말기! 하나씩 하다보면 금방 할 수 있어! 잔소리 끝읏!

Fig. 33. text screen of answer.athome

When a user makes a speech such as "I'm bored" or "Play with me," the server checks what you did today. Based on the data displayed by the user on the app, the information is checked whether he/she exercised, ate, or cleaned. So, each nag corresponding to each data is generated and delivered to the user as one nag. If the user done everything, it gives compliments, and if he/she didn't do at least one thing, it gives the longest nagging of the service.
If the user makes a positive speech when the nagging is over, it delivers a motivational message to create a rewarding day. On the other hand, when a user makes a false utterance, NUGU tells user not to falsely record it on the app, but to try it one by one.

1) Nagging With Exercise
   Check whether you worked out or not today, and if you didn't, check how many days you didn't work out. Nagging related to exercise is divided into more than a day, two days, and three days after not exercising. For each nag, it conveys how much exercise you have not exercised and a nagging that motivates the user to start exercising. The longer you don't exercise, the longer you nag.
   If you haven't exercised for less than two days, ask whether you want to watch an exercise recommendation video on TV. If you give a positive answer, you can watch the recommended video on TV. If it's a negative answer, it says to move your body even a little even if you don't watch the video. If you have exercised for more than three days, you are forced to watch recommended videos on TV.

2) Nagging With Meals
   Meal nagging is a daily base service. In the case of meals, it is made in the process of generating exercise and cleaning nagging. The message is different depending on the state of hunger, the number of meals you had. If you eat more than 3 meals, treat it as normal. If not, NUGU provides nagging for one meal and two meals per day. In addition, nagging is also added to record it on the app, including the case that you did not record it on the app.

3) Nagging With Cleaning
   The cleaning nagging is set based on the cleaning cycle that you set previously. If the cleaning cycle hasn't returned yet, he nags to pick up even the small trash around him. If the cleaning cycle returns, it generates a long nagging along with the number of days it has not cleaned. In addition to the message not to forget the app record, the app checks the cleaning list and sends a message to start cleaning that has not been done.

I. *Click Return to Main Page*

You can return to the main page by clicking Return to Home on each navigation tab.

J. Turn Off The Application

If you want to turn off the application, there are two ways. The first is to press Back continuously. Second, once you swipe up, it returns to the desktop screen. When you swipe up on the desktop screen, the currently running programs are listed. Here, if you swipe up the AI-Jansori, the program terminates.

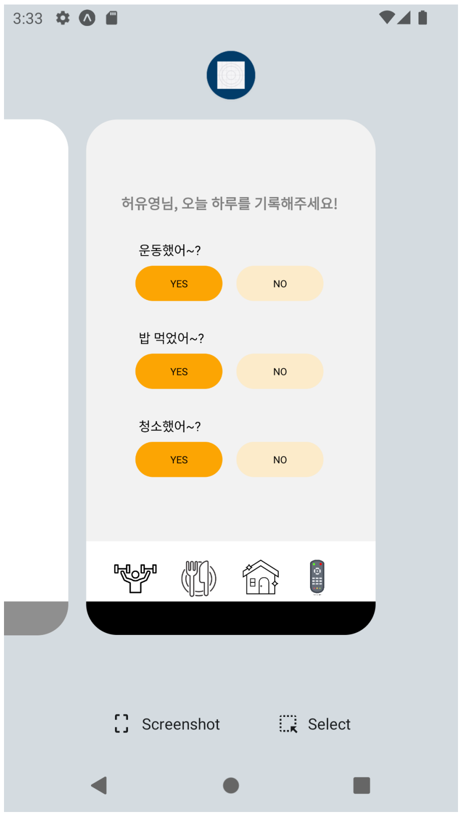


Fig. 34. Program Exit

VII. Discussion

Jansori is a nagging service aimed at improving the healthy life of single-person households. In particular, it focuses on safety and provides various nagging such as weather and congestion. Instead of providing the same nagging to all users, differentiated nagging is provided based on the data entered by each user within the application. Also, it naturally connects to LG home appliances in the process of talking with NUGU.

In addition, Jansori has the following possibilities for expansion.

1) First, our service will be the perfect partner for single households if speaker can initiate the conversation.
2) Second, if congestion data covers much bigger area, congestion function will be the key special point for our service.
3) Last, we can provide more user specific nagging if it combines with machine learning that read user's intention.

References

[1] https://www.tiobe.com/tiobe-index
[2] https://docs.python.org/3/
[3] https://developers-doc.nugu.co.kr/
[4] https://reactnative.dev/docs/getting-started
[5] https://tutorial.djangogirls.org/ko/django/