Mediger - A medication alarm manager on NUGU speaker

*Group name: LightIsLED

1st Son Myeongji
dept. Information System
Hanyang Univ.
Seoul, South Korea
mjnh0915@gmail.com

2nd Yang Jiwon
dept. Information System
Hanyang Univ.
Seoul, South Korea
jiwonyang.tech@gmail.com

3rd Lee SeungEun
dept. Information System
Hanyang Univ.
Seoul, South Korea
a01072783383@gmail.com

4th Chung Yewon
dept. Information System
Hanyang Univ.
Seoul, South Korea
kellychung730@gmail.com

***Abstract*—As our society enters an aging society, social problems related to the elderly are getting serious. We focused on the problem about them and we got to know older people often suffer from improper medication. Because of chronic diseases, the elderly usually take many medicines. In the case, they could confuse the medicines and take them in inappropriate ways. Therefore, we will develop 'Mediger' by using NUGU speaker, an AI speaker, and provide patients with a service to recieve information about proper medication and alarm when to take a medicine.**

***Index Terms*—Medication, Alarm, NUGU speaker, Application**

I. Introduction

A. Motivation

As our society enters an aging society, we started with the idea that services for the elderly would be practical. While searching for problems of elderly, we noticed that there are many elderly who take more than 10 medicines due to chronic diseases, and that they can be hospitalized for inappropriate use of drugs, or even die in severe cases. In order to solve this problem, expert emphasized that there has to be the way to figure out which specific medicine the elderly take hour to hour, which was difficult in reality. Based on this, our team came up with the idea of 'Mediger' as a software that could solve the problem. We abbreviated 'A medication alarm manager'.

At every time they have to take medicine, NUGU alerts and tells them how and what to take, like a secretary. And NUGU checks whether they took the right medication and records the result. By providing day-routine medicine information to the NUGU speaker and mobile application, we suggest a way to solve improper medicine taking problem. It is also likely that SK Telecom, the client, will assess its effectiveness positively. Currently, SK Telecom sets importance on artificial intelligence welfare services for senior citizens. To be specific, in April 2019, SK Telecom launched an artificial intelligence care service to take care of elderly people living alone. In their point of view, the software that we developed is also fancy in that it can enrich their currently in place artificial intelligence care services.

B. Problem statement (client's needs)

We regarded that the client is in need of the development of user-friendly service and the expansion of NUGU speaker service. Thus, we decided to develop a software that can be used in the daily lives of older people who take medicine and that can utilize the NUGU service.

Even though it is daily routine for the elderly to take medicine due to chronic diseases, it is difficult to memorize all of them properly. It is because since many medications are taken with different dosages, the medication is often not taken in the right amount at the right time. If the improper medicine intake lasts longer, it can lead to the severe side effect.

Therefore, to meet the client's requirement and solve the social issue, we figured out our software.

C. Research on any related software

Currently, Korea University Medical Center cooperated with SK Telecom is developing an AI-based voice recognition medical treatment system. The goal is to allowing medical staff to simply enter the cure chart in voice directly and automatically by using AI speaker 'NUGU', which relieves the current medical staff from the cumbersome typing into the chart. However, the point is that this research is on the physician's perspective, not the patient's. Therefore, our team developed a service focused on the patients' daily lives.

As we researched for the background, there were already medicine alarms on the mobile application market. For example, the 'Medicine Time' application is alarmed when the time, medicine type, medicine usage is set, and checked whether the user took it or not. However, we differentiated ourselves by utilizing AI speaker NUGU. We also have created NUGU Play 'Mediger' with the mobile application. By 'Mediger' play, user could set medication alarm in voice via NUGU speaker and Mediger can notice the medication to users in voice. NUGU is an artificial intelligence speaker and allows for variety of input structures. Because there is a self-training process, a variety

of words can be interpreted to suit the user's intentions. Thus, the service is more flexible than the other mobile application services, which requires users to set alarm in a set order and format. This would help older people, who are the main users of our services, to make it easier to use our services.

TABLE I
ROLE ASSIGNMENTS

Role	Name	Task description and etc.
User	Myeongji Son	Assumes which specific services would be popular and needed in the user's point of view. Also searches for the background of the actual services.
Customer	Jiwon Yang	Predicts which requirements could be needed to raise purchasing desire in the customer's point of view. Also when the software development is done, checks if the requirements are sufficient or not.
Software Developer	Seungeun Lee	Draws out a list of software features to satisfy the customer's requirements and works on the actual software development. Tries best to reflect the customer's and user's needs.
Development Manager	Yewon Chung	Totally manages the project schedule and checks the deadline of each role. Helps other roles to communicate with each other smoothly and evaluates the software features.

II. Requirements

A. Functional Requirement

A-1. Mobile application

The application serves to visually show the input and output of data. In addition, it is connected to an external server and passes various data to the speakers via proxy server.

1) The mobile application will be powered by Android and iOS.
2) The application must be able to receive information on medication from the user.
   - The application should be able to process information about the user-inputted 'medicine name', 'dosage', and 'time to take medicine'.
   - The application should be able to execute if the user wants to enter drug information through the photograph. Applications should be able to extract and process medicine information from photographs.
3) The application should be developed for user's convenience.
   - The application should have the text prediction service while typing the medication name.
   - The application should suggest the personal optimized time for taking the medication according to the accumulated personal data.
4) The application will have a simple and intuitive UI for older people to use.
5) The application will use a medication API that contains additional information such as the medicine's effects/ side effects.
6) The application must be able to store the information received from the user in the database. The database will have information about user, medication name and medication time (e.g. - 30 minutes after meals)

A-2. NUGU Speaker

1) NUGU speaker should be able to start 'Mediger' play.
2) NUGU speaker should be able to recognize the voice and read the medicine information from the user's various speeches.
3) The request entered through NUGU speaker should be reflected in 'Mediger' application.
4) NUGU speaker will allow users to check the medication information by voice through 'Mediger' play.
5) NUGU speaker should be able to end 'Mediger' play.
6) NUGU speaker will inform the user that it is time to take certain medicine.
7) NUGU speaker must be able to automatically terminates the alarm when the medication period is over

A-3. Backend Proxy Server

1) The Backend Proxy Server is connected to the NUGU speaker. It receives the user's voice request through the speaker and forwards it to the application server.
2) The Backend Proxy Server is connected to the application server. The Backend Proxy Server receives a response to the request from application server and forwards it to the NUGU speaker.

B. User Management

1) Create a user account
2) Register the information of medication that the user is taking
   - Enter the information through voice via NUGU speakers
   - Type the information in the application
   - Register a photograph about the medication
3) Store information in Database
   - Store the information received through NUGU speaker
   - Store the information entered from users via application
   - If entered as a picture, extract information from the picture and store it to database
4) Notification of medication
   - Alarm that user should take the medication.
   - Notify that the alarm is over when the medication period is over

- Visually delivers effects/side-effects of medicine through application

5) Confirmation of medication
   - Consider the termination of the alarm as taking medicine well
   - User can check through app whether he/she has taken the medicine or not
6) Saving information about whether to take medicine or not

III. Development Environment

A. Choice of software development platform

1) Platform used form
   We will go to use Windows 10 and macOS. Windows 10 is a series of personal computer operating systems produced by Microsoft. According to the data onto 'Usage share of operating systems ', In the area of desktop and laptop computers , Microsoft Windows is generally above 70% in most markets and at 77% globally. Then, it could be familiar to both of users and developers and decided to use. On the other hand, macOS is a Unix-based operating system and is a popular choice since lots of people prefer using Mac so we also choose it for developing.
2) Programming language
   - Javascript : We program using JavaScript because it enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities.
     To be more specific, we use react for front-end and use Node.js for back-end. React is a Javascript library for building user interfaces. It is optimal for fetching rapidly changing data that needs to be recorded such as information about medication. Node.js is an open-source, JavaScript runtime environment that executes JavaScript code outside of a browser. Node.js lets we use JavaScript to write command line tools and produce dynamic mobile app.
   - HTML& CSS : We use HTML (Hypertext Markup Language) for front-end of our mobile app. It is the standard markup language for documents designed to be displayed. We also use CSS (Cascading Style Sheets) to describe the presentation of a document.
   - SQL : We use SQL(Structured Query Language) for managing data. It is a domain-specific language used in programming and designed for managing data held in a relational database management system. It is particularly useful in handling structured data so this makes it possible to process user accounts and medicine-related information.
3) Cost estimation

TABLE II
COST ESTIMATION

Software	Task Description	Cost
Visual Studio Code	source code editor	0
Github	Remote repository	0
MySQL	Data management	0
Overleaf	Document typesetting program	$8/month
Adobe XD	UI design tool	0
Visual Paradigm	UML tool	0
NUGU Play Builder	0	

4) Information of development environment
   - Visual Studio Code : We choose Visual Studio Code as source code editor which could run on desktop and is available for Windows and macOS. Also, it comes with built-in support for JavaScript and Node.js that we would use for programming.
   - Github : We decided to use Github because it provides the necessary management functions for software development including the basic functions of the Git such as bug tracking, functional requests, task management and etc. Also, we can easily share and develop program sources together with our team members.
   - MySQL : We would use MySQL for data management. It is an open-source relational database management system which is using SQL language to add, view or modify data in database. It is used to store account information and information about medicine.
   - Overleaf : We decided to use Overleaf for writing the document. It is an online collaborative writing and publishing tool with real-time collaboration and the fully compiled output produced automatically.
   - Adobe XD(UI design) : We use Adobe XD for user interface design. Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for Windows and macOS, although there are versions for iOS and Android to help preview the result of work directly on mobile devices. We decided to use it because it reflects all the platforms we plan to use.
   - Visual paradigm(diagram tool) : We use Visual Paradigm for modeling UML diagram(Unified Modeling Language diagram). This helps us envision simple and easy to understand the software we want to develop.
   - NUGU Play Builder : Play Builder is a development tool provided by the NUGU play kit that enables easy creation of conversation-based AI voice service

with simple code definition and input of example sentences. The play created can be serviced in conjunction with the NUGU application and device.

B. Software In Use

- Existing Software Medication alarm services(ex. Medisafe, MyTheraphy, ROUNDhealth and etc) that can be operated on mobile apps already exist in many markets. It alerts a user when the user should take medicine. However, it is difficult for the elderly, who are information-weak, to use it. This is mainly because the way it is used is by hand to type information and set alarms. That is why we would utilize NUGU speaker to make medication-related services available to users through voice.

C. Task Distribution

TABLE III
TASK DISTRIBUTION

Name	Task Description
Son Myeongji	App Backend, App Frontend
Yang Jiwon	App Backend, App Frontend
Lee Seungeun	Frontend, Playbuilder
Chung Yewon	Frontend, Playbuilder

IV. Specifications

A. Mobile Application

Fig. 1. app icon

a) Sign up
Sign-up page is a page after seeing logo when users enter to our 'Mediger' app . They can sign up our app by entering their name , the date of their birth and gender . By doing this, we make their accounts and store it to database for identification.By storing the user's session id, once they sign up, they don't have log in again.

b) Calendar Screen
Users can check their medication schedule on the calendar screen after sign up.

a) Horizontal scroll calendar
The calendar is divided into two main parts: At the top, the horizontal scrolling calendar displays only the weekly unit, which can be flipped horizontally to increase user convenience. At the bottom, it tells you the medication schedule for the day you selected on the calendar above. In particular, it is easy to distinguish between medicine taken(green) and those not taken(gray)

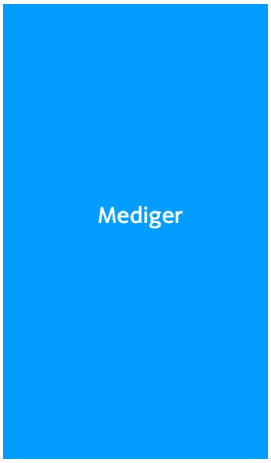

Fig. 2. Splash

Fig. 3. Sign up

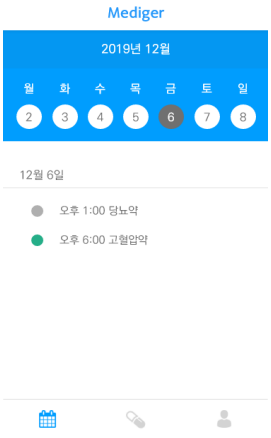

Fig. 4. Calendar Screen

by color. Clicking on the medicine takes users to the Alarm function.

b) Alarm

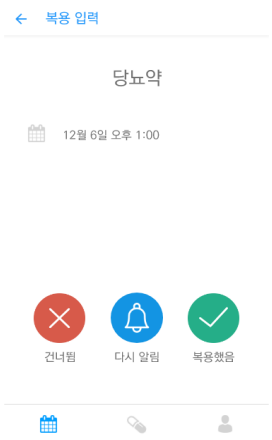

Fig. 5. alarm

When the alarm sounds that you need to take the medicine, the following page pops up. The user can respond 3 ways. You can tell them that you have taken it, ask for a re-alarm, or skip not to take it. The user can easily send his request with the intuitive button design.

c) Medicine List

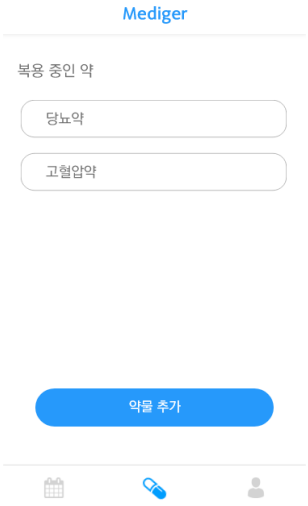

Fig. 6. Medicine list

The user are served a series of services to register the medications, to view a list of medicines, and to view detailed information about individual medicines.

a) Add medicine
The users can add medicine by touching the 'Add medicine' button. Six types of information about medication is required.

Fig. 7. Add medicine

i) Convenient data input
For the elderly, who will be the major users of our service, we add another convenient functions for entering data. By writing an example description in each input field in advance, users will not have to worry about what to write. Start date, end date, medicines, and time fields are more convenient to select via popup. The selected day and time will appear on the screen immediately and can be deleted by clicking it again.

b) Read medicine detail

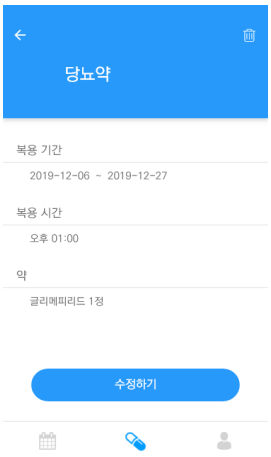

Fig. 8. Medicine detail

Other medication applications that exist today simply function as alarms on time. However, Mediger shows the user's medicine alarm data including start date, end date, and the medicines. The name of medicines alarm is usually the disease of the name like 'diabetes'. The medicines included in alarm can be many medicines that a user take because of the disease.

c) Update/Delete medicine

Fig. 9. Update medicine

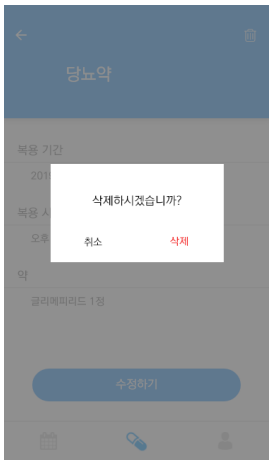

Fig. 10. Delete medicine

The user can change or delete the information of medication on the 'medicine detail' page whenever he needs it.

d) Profile
The users can see his personal information in your profile. In particular, you can edit the information by touching the Edit Profile button. In addition to user's own information, he can add the protector's information by saving the his name and phone number.

B. Nugu Play

1) Setting an alarm
This action sets the alarm by receiving alarm information related to medicine. First, check the user ID and connect to the database. It then receives the alarm name, medication name, dosage, end date, and alarm time from the user and stores them.
2) Deleting the alarm
This action deletes the previously set alarm. It checks

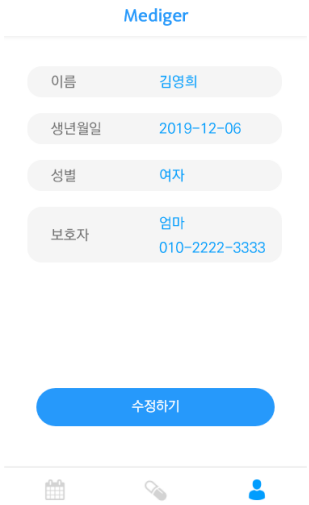

Fig. 11. Profile

the user ID and connects to the database. Then, after receiving the alarm name, ask once again if a user really want to delete the alarm. If the user says yes, it deletes the alarm.
3) Checking medicine taken
This action tells a user the medicines the user took and will take today. When a user asks, "Mediger, what should I take today?" or "Mediger, what did I take today?" this action works. This action responds differently in three cases. If the user has taken all the medication he or she needs today, it would tell him what he has taken. If some of the medicines the user needs to take today exists, mediger would tell him what he has taken and what he has not taken. In this case, mediger asks if the user has taken the medicines that was recorded as an untaken medication. them This is because it is possible that the records are wrong. (If the alarm is not checked, but the user has taken medication for himself.) If the user responds that he or she has taken the medicines, it corrects the records of the database. If the user says he or she didn't take it, it do not make any changes. If no medication is taken yet, mediger gives you a list of medications to take. This action also checks the user ID and connect to the database at first.
4) Changing the alarm
This action changes the information of the alarm that a user set. It checks the user ID and connects to the database. Then, it asks the name of the alarm that the user wants to modify and asks what they want to modify among end date, alarm time, and medicine name. Then, it asks how the user would like to modify it, and then modify what is recorded in the database by using the response of the user as a parameter.
5) Asking what to take
This action is an added action instead of an alarm

function. Currently, the NUGU play builder cannot implement the alarm function, but it will be able to implement it later this year, so we have temporarily made the equivalent of an alarm function into action. Because it is currently impossible to act on the NUGU speaker without a trigger, the user first asked about the medication he should take now. Then it informs the user of the medication if the current time is 1 hour different from the alarm time. After informing, it asks the user if he or she is taking the medication and record whether the user has taken or not.

6) Asking medicine list
This action tells a user a list of medicines that users are taking. It checks the user ID and connects it to the database. It then responds with a list of today's dosages, a list of certain dates of medications, and a complete list of medications, depending on how the user asks.

V. Architecture Design & Implementation

A. Overall architecture

The structure of our service consists of three modules. The first module used javascript and React.js for the front of the app. The Frontend receives the user's request, forwards it to the server, processes the response received from the server and provides the user with an easy-to-see form. Users can effectively view their medication information through the calendar UI. In addition, users can easily enter their medication information with the intuitive color and icon design of the front. The second module is an app server. We used Javascript, Node.js, and express.js. The server receives a request from the user and provides the appropriate response. It also interacts with the Database in this process to find, create, update and destroy data. The last module is the NUGU Play. NUGU Play is a service available on NUGU, an artificial intelligence speaker, and is developed using Play Builder provided by SKT. Our goals are to develop a mediger play through the Play builder and provide alarm services like the mediger app in the NUGU speakers.

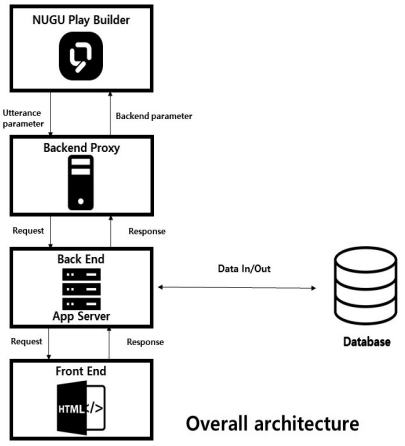

Fig. 12. Overall architecture

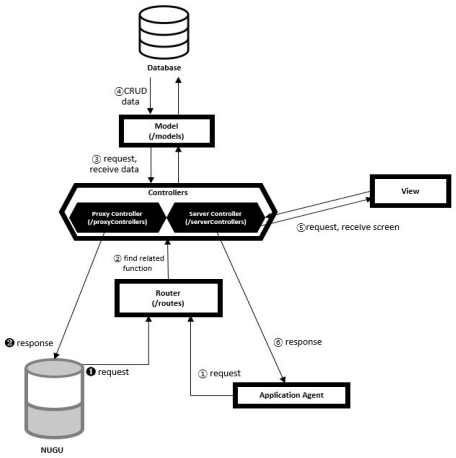

Fig. 13. MVC

B. Directory Organization

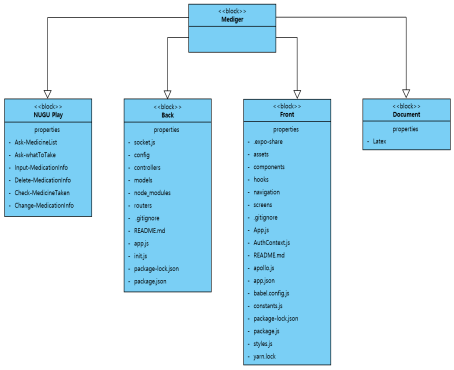

Fig. 14. Directory Organization

Our Mediger can be largely divided into four directories. First, the 'NUGU Play' directory consists of several action trees. And there are files and subdirectories associated with the server in 'Back'. 'Frontend' consists of files and subdirectories related to UI/UX. Finally, the 'Document' directory contains a Latex document.

C. Module 1 - NUGU

1) Purpose
The purpose of the module is providing Mediger service on a NUGU speaker. Through this module, the user could run a Mediger play on NUGU speaker and alert the user to take medicine.

2) Functionality
The functions of this module include setting alarm, changing alarms, checking alarm information, and checking whether or not users have taken medicines, deleting alarm. Unlike the Mediger mobile application

TABLE IV
DIRECTORY ORGANIZATION

Directory	File names	Module names in use
/NUGU	Ask-Medicine List Ask-whatToTake Input-MedicationInfo Delete-MedicationInfo Check-MedicineTaken Change-MedicationInfo	Actions
/Back	app.js init.js socket.js	Initial setup of server using Express framework functions of server-side socket events
/Back/config	onfig.json	Setup of Database
/Back/controllers /serverControllers	askAlarmList.js askWhatToTake.js changeMedicationInfo.js checkMedicineTaken.js deleteMedicationInfo.js inputMedicationInfo.js responseController.js	Functions of proxy server
/Back/models	index.js medicines.js mediSchedules.js schedules.js users.js	Setup of Database structure
/Back /node$_modules$	...	Import modules
/Back/routes	index.js	Major route settings
/Back/routes /serverRoutes	authRouter.js calendarRouter.js medicineRouter.js userRouter.js	Setup of server detailed route specifying functions for each route
/Front	App.js AuthContext.js apollo.js babel.config.js constants.js styles.js	Configuration files App component Define global constants
/Front/screens	Auth/ Tabs/ MedicineDetail.js AlarmDetail.js	Front screens
/Front/components	Alarm.js AuthButton.js AuthInput.js Calendar.js Loader.js Medicine.js NavController.js NavIcon.js Schedule.js ScheduleList.js UserProfile.js	Separated components
/Front/assets	icon.png left-arrow.png logo.png right-arrow.png splash.png	Public static image files
/Front/navigation	AuthNavigation.js MainNavigation.js	Divide into 4 navigations Configure the navigations
/Front/hooks	useInput.js	Component with react hooks

that interacts through UI, interacting and operating with users through voice is a marked difference

3) Play component
It consists of trigger intents and actions connected to them.

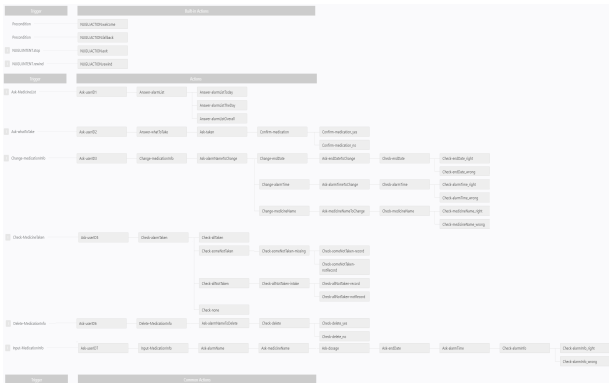

Fig. 15. Mediger Play Action tree

- Ask-MedicineList
This is the trigger intent that activates the checking alarm information function. If the user says, "Mediger, tell me what medicines I am taking." Or "Mediger, tell me what medicines I am taking today."or "Mediger, tell me what medicines I taking January 1st(specific date)", mediger play take action to tell the user the full list of medicines taking or the list of medicines the user need to take today or the list of medicines the user need to take on a specific date.
- Ask-whatToTake
SKT announced that it will enable alarm function later this year so alarm function could not be implemented to our play yet. Originally, the alarm function should be activated, but instead of the alarm function, this is temporarily set aside. This trigger intent, which asks the user what medication he or she should take now, informs the alarm information instead of the alarm. However, it will be replaced by an alarm action when it is time to take.
- Change-medicationInfo
This is the trigger intent that activates the changing alarm function. When the user says "Mediger, I want to change the alarm,", the action is taken to change the end date, drug name, and alarm time stored in the alarm information.
- Check-MedicineTaken
This is the trigger intent that activates the checking either or not users have taken media function. If the user said something like, "Mediger, what should I eat?" "Madiger, what medicine did you take today?", the action that tells you the list of medicines you have taken or the list of medicines you should take is taken.
- Delete-MedicationInfo
This is the trigger intent that activates the deleting alarm function. When the user says "Mediger, I want to delete the alarm," mediger play asks the alarm name and takes action to delete it.

- Input-MedicationInfo
  This is the trigger intent that activates the setting alarm function. When the user says ”Mediger, I want to set an alarm”, mediger play takes the action of receiving the alarm name, medication name, dosage, end date, and alarm time and transmits it to the backend proxy server as a parameter.

4) Where it is taken from
   It is used to use Mediger service on the NUGU speaker.
5) How and Why we use it
   Using the mobile application is difficult and burdensome to older people who are in the information-weak age group. Therefore, we decided that providing a medication alarm service on NUGU, an artificial intelligence speaker along with the Mediger mobile application, would be more useful for older people. The Mediger service can be used through voice instead of UI and the artificial intelligence NUGU speaker has the advantage of training intent and entity of speech. This advantage enables our Mediger play to understand user's speech and operate to suit user's purpose. For this reason, the 'Mediger' NUGU play was developed.

D. Module 2 - Frontend

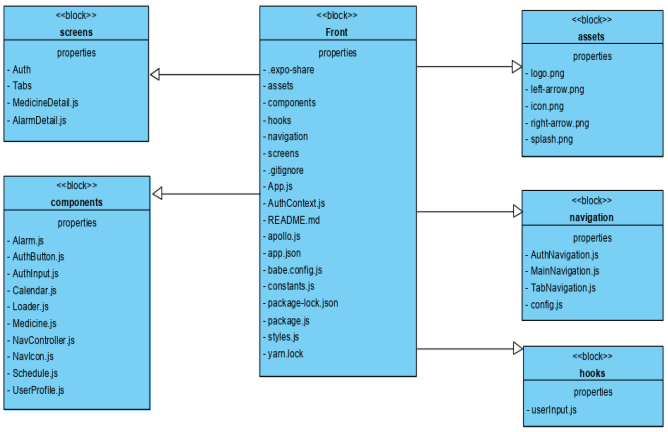

Fig. 16. Front Directory Organization

1) Purpose
   The purpose of this module is to provide UI / UX in ‘Mediger’ mobile applications. The front application firstly provides services and information visually with the user. Secondly, it receives the request from the user and delivers it to the server, and sends the response from the server back to the user.
2) Functionality
   Front application is the media between user and server. The user makes various requests to the front for creating, modifying, and printing out the medication data and the user data. The front sends this request to the server and sends out a server response.
3) Location of Source Code

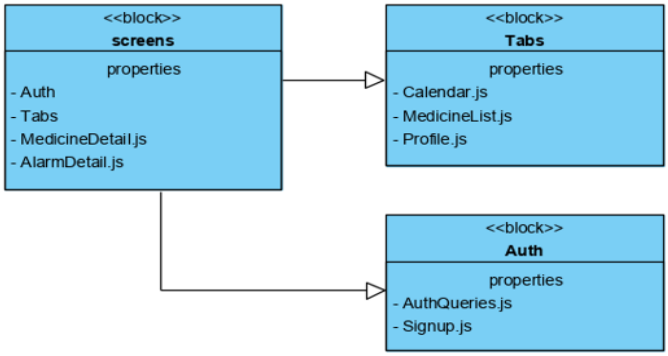

Fig. 17. Front: screens: Directory Organization

- public images: front/assets/
- Navigations: front /navigations/
- Screens: front/screens/
- Components: front/components/

4) Class component
   a) Signup Screen
      This is a screen for registration. As this is the first screen shown to the user, we used blue color and logo that clearly show NUGU and Mediger. This screen consists of three input fields. The first is to enter the name directly, the second is to enter the date of birth directly, and the third is to enter the gender by checking the checkbox. The placeholders increase the user convenience. The front asks the server to create a user account based on the information entered.
   b) Calendar Screen
      The user can very easily request the medication information of the selected date through the horizontal scroll calendar. Colored icons were used to intuitively express medications. If you click on a certain schedule, you will be redirected to the detail screen. This screen also provided more intuitive function to users through colors and icons. When the user clicks on one of the three medication options, the option data is transmitted to the server.
   c) Medicine List Screen
      The user can see a list of medications he is taking on this screen. Each medicine name is contained in a box. To view detailed medicine information, click on the medicine and you will be taken to a screen showing detailed information. On this screen, you can redirect to the update screen. After the update, the user actually requests the server through the update button.
   d) Profile Screen
      The user can check his information on the profile screen. The fields were divided by fields, and the texts of the field names and data were changed to show information more intuitively. The user can also

move from this screen to a screen to modify his profile. In particular, in the update screen, fields for entering accompanier information are shown in toggle form. After the modification, the user actually requests the server through the update button.

5) Where it is taken from
It is used on all screens shown in the three tabs. Each screen provides the appropriate UI / UX to the user.
6) How and Why we use it
In order to implement Mediger's "Medicine Alarm and Configuration" service, we need to provide more effective UI / UX to users. In particular, in consideration of the target users of the elderly, we minimized the input method with the keyboard and maximized the input method using touch and scroll. In addition, it maximizes users' convenience to use easily recognizable icons and outstanding colors. Mediger mobile application front is based on the user's experience.

E. Module 3 - Backend

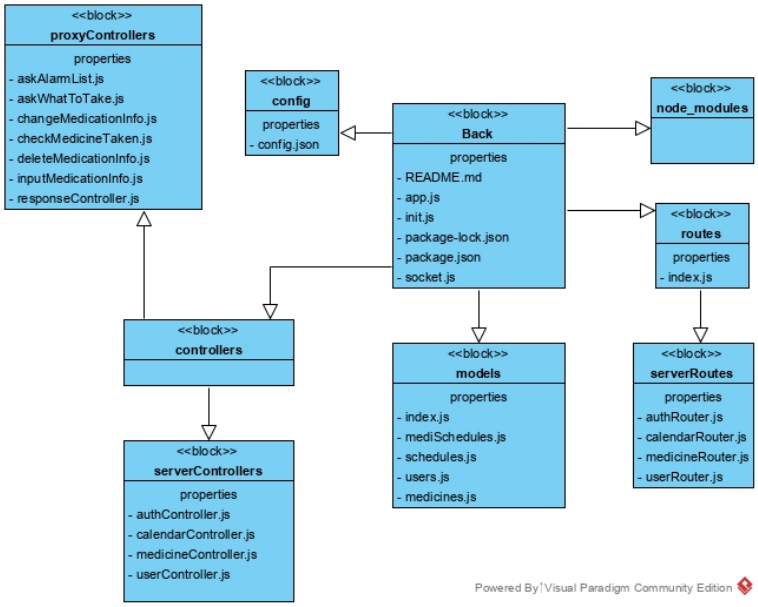

Fig. 18. Back Directory Organization

1) Server
   a) Purpose
   The Backend is intended to provide an appropriate response to requests from users. There are two main ways to respond to a request. Firstly, when the server receives a request, it requests and receives data from the database accordingly. Afterward, the data provided by the database is processed properly and responded to the user. Secondly, when requested, based on it, the server updates the data in the database or creates new data. And this data can be used for another request.
   b) Functionality
   The Backend interacts with the database and provides a variety of functions. The functions can be divided according to the major pages of the Mediger app. First is a function of calendar page. The Calendar page shows a list of alarms that match the specific date that the user chooses. In addition, when a specific alarm is selected, it shows the Intake Information Entry page, where the user can enter whether or not to take medicines. Second is the function of the Medicine page. The main function of the Medicine page is the alarm add-on. This function allows users to enter the name of the alarm, the names of the medicine they take, the intake, and the time to take medicine, and add new alarms. In addition, the user can check the list of alarms and medications users are taking after users enter the input. Third, there is a user page. The User page shows the user's information by default and users can modify their own information in User page. users can add accompanier information through edit function in User page. Not only this, but it also implemented sign up and alarm functions.
   c) Location of Source Code
   : /Mediger/Back
   d) Class component
      (1) Sign Up
      A function is implemented for signup. When users enter to the application, if there is no userID, then function moves users to the Join page. In Join page, the function stores the name of user, birthday and gender information entered by users into the database and then assign value to session.user.userID. Afterward, users are turned over to the Calendar page. If there is userID, immediately users go to the Calendar page.
      (2) Calendar
      In the Calendar page, Firstly, It has the function to extract the alarm list. When a user selects a specific date, the corresponding alarm list is imported from the schedule table of the database, especially the alarm name, time, and intake column. Secondly, if users select a specific alarm, users are moved to the Intake Information Entry page for the alarm. This page also saves the user-entered intake to the schedule table of the database.
      (3) Medicine
      In Medicine Page, Firstly, It extracts alarms and a list of medications users are taking from the database. Secondly, users can add alarms. When adding an alarm, users enter the alarm name, name of the medication users are taking, dosage, and time to take and the function save them into

the database. Then, the function moves a user to the Medicine page again and extracts the list containing the newly added alarms.

(4) User
First of all, the User page gives the user's basic information. This provides information on the user's name, date of birth, and sex and extracts user information from the database. In addition, users can add accompanier information through edit. At that time, the function updates the appropriate row in the database using the accompanier name and phone number entered by the user. And then the function moves user to User page and provides newly added accompanier information.

(5) Alarm
The alarm is provided at the specific time set by the user. the alarm function updates the alarm list if a socket named 'insert' is sent from the client-side from the Medicine page. Afterward, if the user sends a "ready" socket on the calendar page and it is time to take the medicine, inform the user that it is time to take the medicine using a pop-up.

e) Where it is taken from
It is used by where the user's request is handled. The appropriate server functionality is used for requests from each page. It includes the use of the database within the server functionality as needed.

f) How and Why we use it
We want to provide Mediger service which is a medicine alarm service. To implement this, we need to provide users with various functions related to medicine alarms. These functions have been described earlier and we have created and have used a server which includes those functions. Also, in order to provide functions, we need to store a variety of information entered by the user. Therefore, we have built the database. The database can be used to store user-inputted information and to provide the appropriate information to the user as needed.

2) Proxy

a) Purpose
The proxy server acts as a gateway between the NUGU speaker and the server. It is an intermediary server separating NUGU client from the servers.

b) Functionality
It performs actions of NUGU speakers such as setting alarms, deleting alarms, checking user medicine taken, changing alarms, asking what to take, asking medicine list.

c) Location of Source Code
: /Mediger/Back

d) Class component
It is related to actions defined in NUGU Play Builder.

i) Setting an alarm

(1) Action name: Check-alarmInfo
After the user confirms(say "Yes") to set a new alarm, the parameters(user id, alarm name, medicine name, dose, alarm time, end date) is passed. It inserts new data into Schedules, MediSchedules, and Medicines tables in Mediger database. If succeeds, it responds with OK sign. Else, it terminates with specifying error.

ii) Deleting the alarm

(1) Action name: Ask-alarmNameToDelete
After the user mentions the expected alarm name, the parameters(user id, alarm name) is passed. It finds out the alarm's information(alarm time, end date, alarm id) in 'Schedules' table and responds with these backend parameters(alarm time, end date, alarm id) and OK sign. If there is an error, it terminates.

(2) Action name: Check-delete
After the user confirms(say "Yes") to delete the particular alarm, the parameters(user id, alarm id, alarm name, end date) is passed. It deletes all data in 'Schedules' table which satisfy above conditions. The corresponding data in 'Medischedules' table is deleted cascade. If succeeds, it responds with OK sign.

iii) Checking medicine taken

(1) Action name: Check-alarmTaken
If the user wants to check whether he/she has recorded alarm or not today, the parameter(user id) is passed. It selects all alarms in 'Schedules' table where conditions are 'scheDate'=today, 'userID'=user id. Then, it classifies the alarm with recorded taken and not recorded taken. If succeeds, it responds with parameters(alarm recorded, alarm not recorded) and OK sign. Else, it terminates.

(2) Action name: Check-allNotTaken-intake
If the user wants to update the "intake" information, the parameters(alarm id) are passed. Then, it updates the corresponding alarm's 'intake' column in 'Schedules' table. If succeeds, it responds with OK sign. Else, it terminates.

(3) Action name: Check-someNotTaken-missing
It works same as 'Check-allNotTaken-intake'.

iv) Changing the alarm

(1) Action name: Check-endDate
If the user wants to change the end date, the parameters(user id, alarm name, new end date) are passed. If the new end date is greater than the previous end date, firstly, it updates existing data's end date, secondly, it inserts new data as same information above. If the

new end date is less than the previous end date, firstly, it deletes out existing data where ‘scheDate’¿new end date, secondly, it updates end date. If succeeds, it responds with OK sign. Else, it terminates.

(2) Action name: Check-alarmTime
If use wants to change the alarm name, the parameters(user id, previous alarm name, new alarm name) are passed. It updates the alarm name in ‘Schedules’ table where all conditions above are satisfied. If succeeds, it responds with OK sign. Else, it terminates.

(3) Action name: Check-medicineName
It works similar with Check-alarmTime.

v) Asking what to take

(1) Action name: Answer-whatToTake
If user asks which alarm is supposed to ring right now, the parameter(user id) is passed. It selects data where ‘scheDate’=today and ‘scheHour’ in (current hour-1, current hour, current hour+1). If succeeds, it responds with OK sign. Else, it terminates.

(2) Action name: Confirm-medication
It works similar with Check-allNotTaken-intake.

vi) Asking medicine list

(1) Action name: Answer-alarmListToday
It finds out the scheduled alarm on today.

(2) Answer-alarmListTheDay
It finds out the scheduled alarm on a specific day.

e) Where it is taken from

i) Backend proxy server is called by the NUGU platform’s REST API in the following cases

(1) Play Builder needs to get essential information to response

(2) Play Builder needs a judgement to a specific value

(3) An exceptional situation

f) How and Why we use it

i) REST API calls are made only in the actions which is specified to use the Backend proxy, and each action determines its own REST API URL. REST API URL generation rules are as follows

(1) Play Builder - General - External Server Connection Information - Web URL + Play Builder - Actions - Action Name
For example, if you set https://mediger-server.herokuapp.com to the Web URL of the external server connection information, and if you set the action name to ”alarmList”, the REST API URL that handles the action will be https://mediger-server.herokuapp.com/alarmList. For each of the information, see play builder the screen is as follows.

TABLE V
PROXY TERMINOLOGY

Action Name	Used to distinguish which requests need to be processed by the Backend proxy. Only checked actions in the NUGU Play Builder call the Backend proxy.
Parameters	Parameters defined in the NUGU Play Builder are passed. The Utility Parameter passes a normalized value as ”value” for the Entity or its Entity contained in the Play user’s actual firing. Backend Parameter is a parameter that will contain ”value” and pass it to the Play. When it firstly calls the Backend proxy, its value will be passed as ”null”.
Context Information	User identification token, device status information, and so on are passed.
Event Information	Event information from the NUGU device. The Backend proxy server should use the above information to communicate the appropriate information for requests made by a particular user.

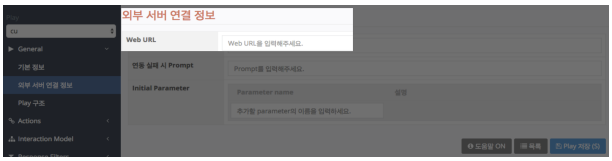

Fig. 19. External Server Connection Information

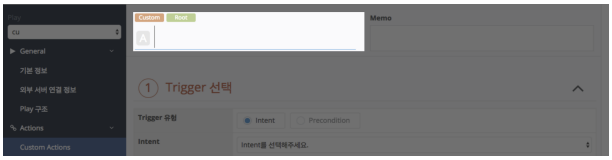

Fig. 20. NUGU Play Builder Action

ii) The Dialog Manager on the NUGU platform requests information using its fixed format below. Therefore, the proxy server is needed to make Mediger Server’s REST API format same as the NUGU platform’s format.

VI. USE CASES

A. Use Cases for NUGU Speaker users

1) Use case 1 : Alarm Setter
This is the case in which the alarm is set. Alarm Setter needs the process of connecting to database via the own user ID. . After connection to database via the user ID, alarm setter could input alarmName, medicineName, dosage, endDate and alarmTime as utterance parameter.

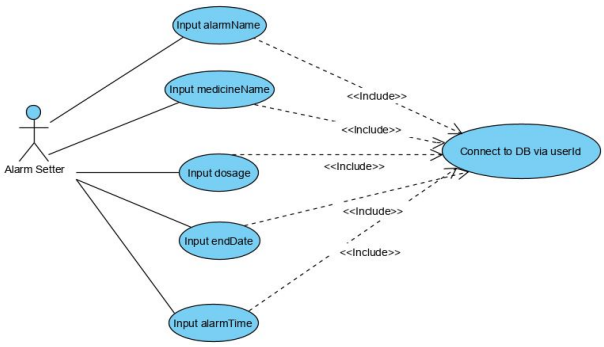

Fig. 21. Use case 1 : Alarm Setter

2) Use case 2 : Alarm Deleter

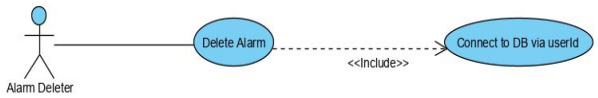

Fig. 22. Use case 2 : Alarm Deleter

This is the case in which the alarm is deleted. Alarm Deleter needs the process of connection to database via the own user ID. After connection, Alarm Deleter could choose the alarm to delete by using alarm name.

3) Use case 3 : Medicine taken Checker

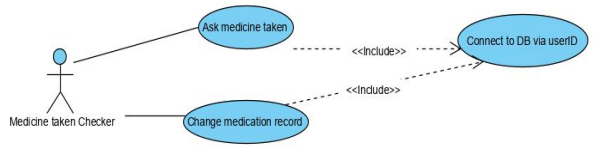

Fig. 23. Use case 3 : Medicine Taken Checker

This is the case in which the user checks the medication a user has taken. Medicine taken Checker needs the process of connection to database via the own user ID. After connection, Medicine taken Checker could ask what medicine he or she has taken or medicines to take.

4) Use Case 4 : Alarm Changer

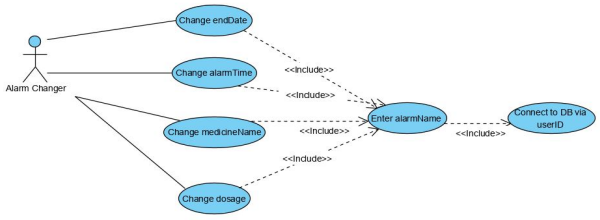

Fig. 24. Use case 4 : Alarm Changer

This is the case in which the user wants to change the alarm information. Alarm Changer needs the process of connection to database via the own user ID. After connection, Alarm Changer could choose which alarm he or she wants to change by entering the alarm name. Then, Alarm Changer would choose the category to change among end date, alarm time or medicine name. If the medication changes, the dosage should be changed as well. So if the user wants to change the name of the medicine, the user could set a new dosage.

5) Use case 5 : Alarm Asker

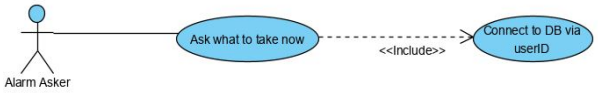

Fig. 25. Use case 5 : Alarm Asker

This is the case in which NUGU notify the user of the medication to be taken. However, there is a limitation for implementing alarm function, the user should ask what medicine he or she would take now. This case also needs the process of connection to database via the own user ID.

6) Use case 6 : Medicine list Asker

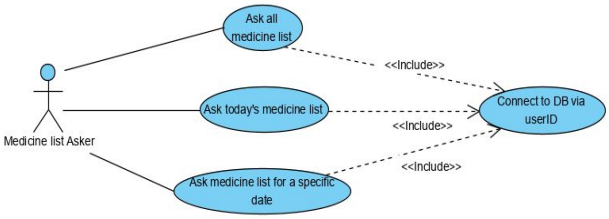

Fig. 26. Use case 6 : Medicine list Asker

This is the case in which the user wants to know the medicine list which he or she has taking now. Medicine list Asker needs the process of connection to database via the own user ID. After connection, it could responds by telling the all medicine list, today's medicine list, or medicine list for a specific date.

B. Use Cases for Mobile Application users

1) Use case 1 : Register
At rst when users enter into our mobile application, login form appears. After users fill in all the fields, users can sign up through button below login form. After entering user information such as name, date of the birth and gender, our service is available then.

2) Use case 2: Calendar Screen
After sign up, users move to the main screen "Calendar Screen." The calendar is displayed this week by default. Users can scroll horizontally to see other weeks. Users can check the medication schedule for the date selected below the calendar. If you have taken medicine, it is indicated by a green circle, and if you have not taken it, it is indicated by a gray circle. When the user clicks on a schedule, the user can be redirected to a screen for checking whether the schedule is to be taken.

Fig. 27. Use case 1-1: Sign up

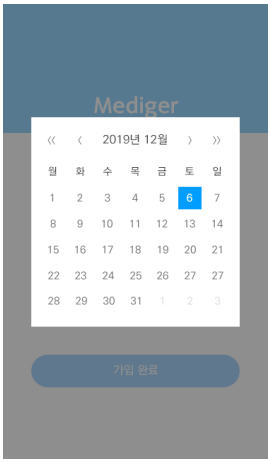

Fig. 28. Use case 1-2: Sign up - Fill the date of birth

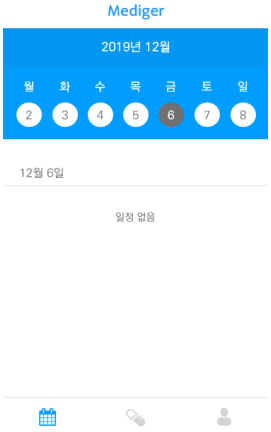

Fig. 29. Use case 2-1: Calendar - Empty schedule

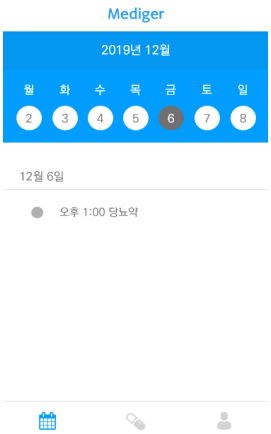

Fig. 30. Use case 2-2: Calendar - Not take medicine

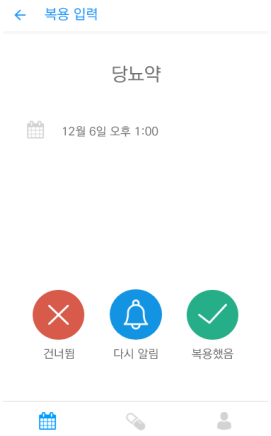

Fig. 31. Use case 3-1: Alarm

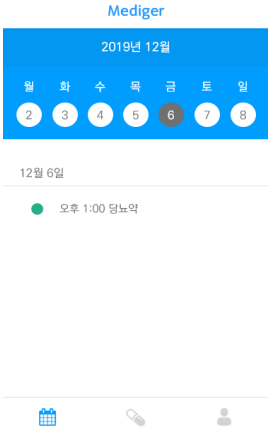

Fig. 32. Use case 3-2: Calendar - Took medicine

3) Use case 3: Check a Medication And Get a Medication Alarm Service
On the screen from use case 2, the user can check whether the medication is on or off. If the user haven't taken it, the user would click the "Skip" button. If the user took it, the user would click on the "take it" button. Based on the medication information entered by the user, an alarm will sound to inform the medication time. This

screen is displayed. After checking for medication, the user is redirected to the calendar screen. And the color of the circle became green.

4) Use case 4 : Medicine List Screen Register an Alarm

Fig. 33. Use case 4-1 : Medicine List – empty

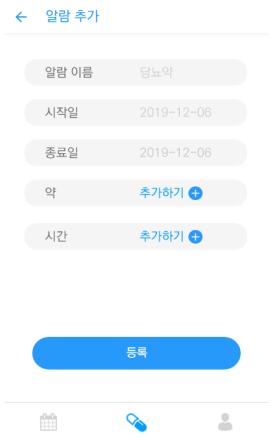

Fig. 34. Use case 4-2 : Add medicine

The user can go to the Medicine list screen by clicking on the medicine icon in the bottom tab. This screen allows the user to see the name of the alarm you are taking. By clicking on the ‘Add’ button at the bottom, the user will be taken to a screen to register an alarm. The user can click the alarm name field and enter it directly. When the user clicks on the start date and end date fields, they can be entered by clicking on the datetime pop-up. The user can type in the input pop-up by clicking on the medicine field. The medicine entered is created as a box at the bottom of the field. If the user want to delete the medicine the user entered, the user can click on the box containing the medicine. When the user clicks the time field, the user can click AM and PM in the time pop-up, and enter a time. The entered time is created as a box at the bottom of the field. If you

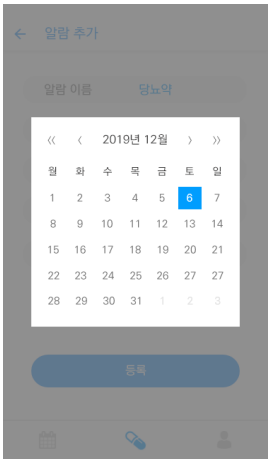

Fig. 35. Use case 4-3 : Datetime input popup

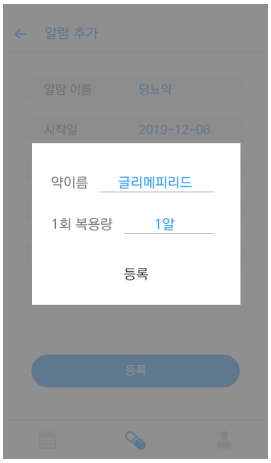

Fig. 36. Use case 4-4 : Medicine input popup

Fig. 37. Use case 4-4 : Medicine input popup (result)

want to delete the entered time, the user can click on the box containing the time. After filling in all the fields, the user can register the medicine by clicking the ’Register’ button. After registration, the user is redirected to the

Fig. 38. Use case 4-5 : Time input popup

Fig. 39. Use case 4-5 : Time input popup (result)

medicine list screen back.

5) Use case 5: Get an Alarm Information

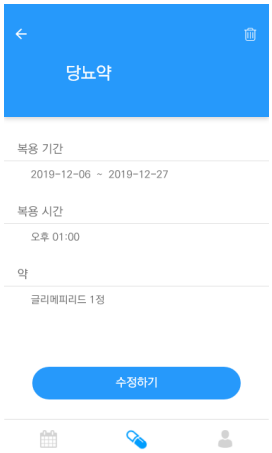

Fig. 40. Use case 5 : Medicine Detail

The user can click the box with the alarm name on the medicine list screen to view detailed alarm information. The user can check the name, start date, end date, name of medicine and alarm time. If the user wants to edit this information, the user can click the ”Update” button at the bottom. When the button is clicked, the user is redirected to the update screen.

6) Use case 6 : Update an Alarm Information

Fig. 41. Use case 6 : Update Medicine

Modifying alarms is almost identical to registering alarms in Use Case 4. The user can click the alarm name field and update it directly. When the user clicks on the start date and end date fields, they can be updated by clicking on the datetime pop-up. The user can type in the input pop-up by clicking on the medicine field. The medicine entered is created as a box at the bottom of the field. If the user want to delete the medicine the user entered, the user can click on the box containing the medicine. When the user clicks the time field, the user can click AM and PM in the time pop-up, and enter a time. The entered time is created as a box at the bottom of the field. If you want to delete the entered time, the user can click on the box containing the time. After filling in all the fields, the user can update the medicine by clicking the ‘Update’ button. After registration, the user is redirected to the detail alarm screen back.

7) Use case 7 : Get a User Profile
The user can go to the User profile screen by clicking the person icon in the tab at the bottom. The user can check his name, date of birth, and gender that he entered when signing up. The user can click the ‘Update’ button to add the protector.

8) Use case 8: Update a User Profile
The user can click the name and gender fields to edit the name directly. When the user clicks on the date of birth field, he can click on it in the datetime pop-up. The user can register in the input field at the bottom by clicking ’Add’ in the protector field. If user already have a protector registered, user can edit it in the same way by clicking ‘Update’. After filling in all the fields,

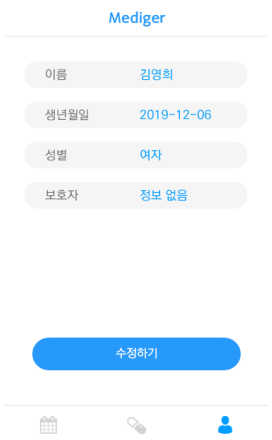

Fig. 42. Use case 7 : Profile

Fig. 43. Use case 8-1 : Update Profile

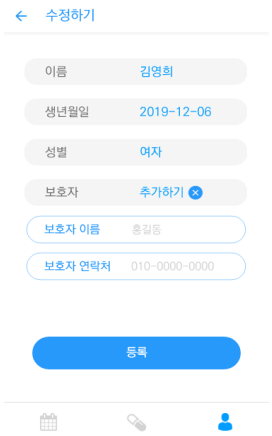

Fig. 44. Use case 8-2 : Update a protector information

the user can edit the profile by clicking the 'Register' button.

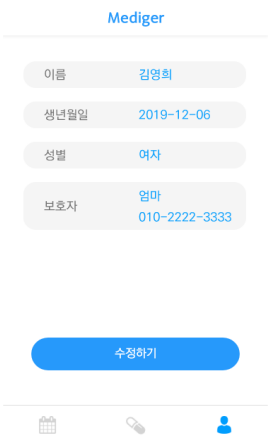

Fig. 45. Use case 8-3 : Updated Profile

VII. Discussion Write one or more paragraph(s)

Our team chose the waterfall model among the software development methods. Waterfall Model is a sequential model that divides software development into different phases. It has 4 steps such as software requirements, software design, software implementation, software verification, and software maintenance. It suited for smaller projects where requirements are well defined. Before the next phase of development, each phase must be completed. However, there was some disadvantages of the waterfall projects. First, documentation occupies a lot of time of developers and testers. Secondly, small changes or errors that arise in the completed software may cause a lot of problems. Thirdly, testing period comes quite late in the developmental process. Because of these disadvantages, our development was slow. There were some problems in the second half, resulting in an emergency. Also, we have developed backend and frontend at the same time, we had some issues in communication. There was a communication conflicts with the API sharing due to the lack of documentation. The development speed beteween backend and frontend was different. It occurs a disagreement about the data type in the API. We had to change the format of data from the server. We have experienced both the advantages and disadvantages of this waterfall model. We tried to make up for the shortcoming of the method. Especially, we increased the number of communications. Overcommunication not only reduced conflicts in communication, but also made development faster. Also, We realized that it is important for backend developers and frontend developers to discuss the API before development. Later, we also want to try the agile software development. Based on pragmatism, agile software development is a development method that promotes iterative development throughout the life cycle of a project. This method is less document-oriented and

code-oriented. Thanks to the software engineering class, we had a good opportunity to write a technical document and to experience software development.

- github address : https://github.com/LightIsLED/Document