MulMeokNyang

Intelligent cat watering machine that Recognizes individual cats with AI

Ann Jukyung
Dept. Information Systems
Hanyang University
Seoul, South Korea
email@hanyang.ac.kr

Choi Chansol
Dept. Information Systems
Hanyang University
Seoul, South Korea
email@hanyang.ac.kr

Lee Yunsun
Dept. Information Systems
Hanyang University
Seoul, South Korea
email@hanyang.ac.kr

Hong JunGgi
Dept. Information Systems
Hanyang University
Seoul, South Korea
sentorino@hanyang.ac.kr

***Abstract*—Cats are known for their discerning palates and can be quite selective, particularly when it comes to wet food, especially if they haven't been exposed to a variety of flavors previously. This dietary preference, coupled with a reluctance to consume adequate water, can lead to dehydration and pose a significant threat to feline health. To address this issue, smart cat water dispensing systems aim to offer a solution through the identification and in-depth analysis of individual cats' unique preferences and hydration needs.**

***Index Terms*—identification, detection, classification, opencv, cats**

I. Role Assignment

Roles are assigned to improve software development process and increase team productivity. Group name is called Nyangporter.

Name	***Role***	***Responsibilities***
Choi Chansol	Development manager	The role involves task allocation, project progression assessment, translating requirements into practical functionality, and discerning the most suitable frameworks for the project. This position also encompasses the execution of software features and active collaboration with co-workning Software Developers to deliver essential functionalities.
Ann Jukyung	Software developer	This role primarily involves implementing software features and collaborating with fellow developers to meet feature requirements. Additionally, it includes the vital task of ensuring that the minimum viable product remains on schedule. This position is also responsible for engaging with users and customers to gather and integrate feedback into the product's development process. Code maintenance and overseeing the coordination of pull requests are also integral aspects of this role.
Lee Yunsun	User	Tasked with app testing and identifying any deficiencies that require enhancement, this role also involves offering goals, expectations, and deliverables for improving these weaknesses and ensuring ongoing compliance with requirements. Should the requirements fall short of expectations, the individual is responsible for communicating actionable feedback and evaluating implemented features.

Hong JunGgi	Customer	This role is responsible for thoroughly testing the application, not only for functionality but also for usability, design, and overall user experience. This testing process allows for a comprehensive assessment, providing valuable insights that can guide improvements and refinements in various aspects of the application, ensuring it meets the user's distinct needs and requirements more effectively.

II. INTRODUCTION

A. Motivation

Cats tend to consume only the least amount of water, and due to this lack of drinking water, they often have health problems and have to go to the hospital regularly. Since animals are not applied by insurance, the burden of owners is considerable. In addition, in order to increase the amount of water consumed, the owners make them drink wet feed or force them to drink water through injections, but some cats suffer from allergic reactions to wet feed, and they show severe rejection to forced water intake by injection.
So we need a natural way to increase the amount of water consumed. Also, the number of households raising companion animals is increasing these days due to the increase in single or two-person households.

Therefore, we will proceed this project so that households with cats can manage the amount of water consumed by cat through smart cat water supply machine and mobile applications.

B. Problem Statement

Though there are some exceptions, most cats are particularly susceptible to becoming dehydrated as they abhor drinking water like other animals do. Smart cat water supply machine is a project aiming to recognize individual cats and analyze their water consumption data in order to make sure they are consuming appropriate amounts of food and water to avoid dehydration. We envision Smart cat water supply machines to allow clients, especially with more than two or more cats, to have access to cats' water consumption data easily and notify them if they may be in a threatening condition.

In the existing pet water supply system, only one animal per water supply system could be managed because there was no individual identification function in case of raising multiple cats and dogs in one household. With this in mind, our team will recognize the faces of multiple pets to enable differentiated negative number management for each individual.

Our goal is to provide clients with an application that can monitor their cats' water consumption data individually and hopefully give clients information on cats' health conditions. Identification will be done by embedding a camera on the feeder product and capturing cats' faces to analyze their facial features through Machine Learning. This solution will be able to help both clients and cats themselves by being able to monitor their water intake easily and providing them with necessary information regarding cats' health condition.

C. Research on any related software

The AI cat feeding product landscape is indeed populated with various existing solutions, reflecting the strong interest of pet owners in this field. However, it's important to note that the majority of these solutions fall into two categories: not AI, commercially-driven products, and non-open source projects. It was either too simple to be called Artificial Intelligence or AI part was very confidential.

VaraemPet's Welli Smart Hydration Care: In response to the central proposition concerning the daily water intake recommendation for humans, this innovative AI-driven pet hydration monitoring system seeks to address the vital question of how to ensure our pets' proper hydration. This system offers comprehensive hydration monitoring, enabling pet owners to precisely gauge their pets' hydration levels in comparison to recommended averages. It meticulously records and evaluates crucial data, including food intake, water consumption, and weight, all consolidated within a comprehensive health record. Furthermore, it provides real-time notifications via the Baraem app, keeping pet owners updated each time their pets drink and offering valuable insights into their hydration patterns. To facilitate a deeper understanding of their pet's hydration trends, the app displays detailed daily, weekly, and monthly hydration graphs, along with comparisons against peer averages. Additional features encompass monitoring remaining water levels and issuing timely cleaning notifications. The system's adjustable height feature caters to pets of various sizes, ensuring a comfortable drinking experience. Moreover, it follows a standard hydration formula based on the pet's weight, recommending a daily water intake of (Weight x 50 ml). It's important to note that the product's limitation lies in its inability to provide individual identification for multi-pet households, which means it cannot differentiate or tailor services for each pet separately.

III. Requirements Analysis

A. Sign up

- *Basic Information Input Screen:*
  - Method 1) Quick Sign-Up Feature
    - Utilizes the Kakao, Google, and Naver sign-up APIs.
    - If a quick sign-up is done, it directly proceeds to the user profile setup screen.
  - Method 2) Local Sign-Up Feature
    - Email Input: The system checks and displays whether the input is in the correct email format or if the email has already been registered.
    - Password and Password Confirmation Input: The system checks if both values match and displays the result on the screen.
  - If all the fields are correctly filled out and authentication is complete, the "Next" button is activated, leading the user to the user profile setup screen.

- *User Profile Setup Screen:*
  - Profile Picture Registration (optional)
    - Clicking the "Add Photo" button allows the user to either fetch a photo from the album or take one instantly with the camera. The chosen photo can then be registered.
    - If no photo is registered separately, a default image is displayed. - Nickname Input (Required)
  - Nickname Input (Required)
    - The system checks the format and checks for duplicates, then displays the result on the screen.
  - Self-Introduction Input (optional)
    - A maximum character count is specified.
  - If the nickname is correctly entered, the "Complete Sign-Up" button is activated. After sign-up, the user starts from the ¡water dispenser device registration¿ screen in a logged-in state and proceeds through the cat hydration management space creation procedure.

B. Login

- *Method 1) Quick Login:*
- *Method 2) Local Login*
  - The system checks whether the email address and password have been entered.
  - If entered, the system verifies if there's a matching user in the database.
  - If no match is found, the user is prompted to re-enter. If a match is found, a cookie is generated to maintain the logged-in state continuously.
  - If the auto-login checkbox is checked, a session is generated to remember the login details, and upon logging out, the login details are automatically filled in.
  - After successful login, if the user has their own space or is a part of any, they are directed to the cat hydration management space.
  - Otherwise, they begin from the water dispenser device registration screen and proceed through the cat hydration management space creation procedure.

C. Find Email

- Upon successful authentication, the system searches the database using the phone number and displays the user's email on the screen.

D. Forgot Password

- Enter the email address for which you want to retrieve the password.
- After authentication, the system searches the database using the email and phone number, and then sends the user's password to the specified email.

E. Water Dispenser Device Registration

(Step 1 of Creating a Cat Hydration Management Space)

- Instruct the user to put the water dispenser device into pairing mode.
  - Provide a picture indicating the location of the pairing button on the dispenser.
- Search for available water dispenser devices and display them in a list.
- The user selects the desired device from the list.
- Once paired successfully, register the device information in the application and then navigate to the cat profile registration screen.

F. Cat Profile Registration

(Step 2 of Creating a Cat Hydration Management Space)

- *Basic Information Entry Screen*
  - Register a profile picture of the pet cat, and enter its name, weight, and age.
  - Once all details are entered, the next button is activated, allowing the user to proceed to the Nose Print Photo Registration screen.
- *Nose Print Photo Registration Screen*
  - Uses Nose Print Recognition AI model.
  - To identify the individual cat, submit multiple photos of the cat's nose from various angles.
  - After uploading the photos, you can press the next button to proceed to the Wet Food Intake Information Registration screen.
- *Wet Food Intake Information Entry Screen*
  - Select 'Yes/No' for consumption status.
  - If 'No' is selected, activate the "Next" button.

    - If 'Yes' is selected, input the daily intake amount in grams and the moisture content of the food. Once all details are entered, activate the "Next" button.
        - If the moisture content is unknown, a default value of 70% is set.
    - Clicking the "Next" button leads to the hydration setting screen.
- *Hydration Setting Screen*
    - Choose between 'Automatic setting/Manual setting'.
    - If 'Automatic setting' is chosen, the recommended hydration formula is applied, and the "Next" button gets activated.
        - Daily basis formula: '(Weight(kg) * 50ml) - (Wet food amount (g) * Moisture content(%))'
    - If 'Manual setting' is chosen, users input the daily target hydration amount, and then the "Next" button gets activated.
    - Add another cat button:
        - Redirects to the second cat profile registration screen.
    - Complete profile registration button:
        - Directs to the cat hydration management space.

G. Cat Hydration Management Space (Main)

1) Daily Hydration Info for Each Cat:

- Cat Profile Selection Top Bar
    - By tapping the profile picture, users can access the hydration info screen for that specific cat.
- Cat Profile
    - Displays the cat's profile picture, name, age, and weight.
    - An edit button lets users access a screen where they can modify that cat's profile. (Reuses the Cat Profile Registration screen).
    - Tapping the right arrow displays the daily hydration info for the next cat. The last item is a plus button, which leads to a screen to add a new cat profile. (Reuses the Cat Profile Registration screen).
- Daily Hydration Gauge
    - Displays how much of the daily hydration goal the cat has achieved, as a percentage(%).
    - Different colors indicate hydration ranges:
        - 0%   29%: Red
        - 30%   59%: Yellow2
        - 60%   89%: Green
        - 90%   Upper Limit: Blue
        - Upper Limit   200%: Blue up to the upper limit, then red
    - The upper limit is set to address excessive hydration, calculated using the recommended formula.
    - Push notifications are sent to the user if the upper limit is exceeded.
- Daily Hydration Evaluation & Advice
    - Based on the cat's water intake for the day, an evaluation is given, advising if more water intake is needed.
- Hydration Statistics Button
    - Leads to the periodical hydration statistics screen for that cat.
- Watering Attempt Button
    - Tapping this button prompts the water dispenser to make a cat-calling sound, enticing the cat to approach and drink.
    - If the cat drinks within 30 minutes, a push notification is sent to the user.
- Navigation Bar
    - Tapping the NavigationBarIcon slides out the navigation bar from the left.

2) Periodical Hydration Statistics Screen:

- Cat Profile Selection Top Bar
    - If accessed from a specific cat's daily hydration info screen, that cat is selected.
    - Otherwise, the first cat is selected by default.
    - Users can view statistics for other cats by selecting their profiles.
- By default, displays a bar graph of the past week's hydration data.
- Users can choose between 'One Week/One Month/One Year' durations.
- A calendar button allows users to select specific 'Week/-Month/Year'.
- Tapping a bar displays hydration data for that specific 'Day/Week/Month'.
    - If the graph is set for one week, it displays data for that day.
    - If set for one month, it displays data for that week.
    - If set for one year, it displays data for that month.
- At the bottom, an evaluation and advice on hydration are provided.
    - If 'Red' or 'Yellow' days make up more than half, a message suggests increasing water intake.
    - If 'Green' or 'Blue' days make up more than half, a message confirms adequate hydration.
    - If days with 'Blue + Red' make up more than half, a message advises consulting with a veterinarian due to excessive water intake.

H. Navigation

To quickly transition to the desired screen, navigation links are provided in the navigation bar. The navigation bar is applied to screens where necessary, providing relevant links for each screen.

- User Profile: Displays the user's nickname, email, profile picture, and self-introduction.
- Menu Link List:
  - Profile Management: Takes the user to a screen where they can modify their profile.
    - (Reuses the User Profile Setup screen from the sign-up process)
  - Cat Profile Management: Takes the user to a screen where they can modify the cat's profile.
    - (Reuses the Cat Profile Registration screen)
  - Today's Hydration Info: Takes the user to the screen that displays the day's hydration information.
  - Hydration Statistics: Takes the user to the screen that displays periodic hydration statistics.
  - Co-admin Management: Takes the user to the Co-admin Management screen.
- Logout Button:
  - Upon selection, the app processes the logout and redirects the user to the initial screen (Sign-up and Login).

I. *Co-admin Management*

To enable the use of the Cat Hydration Management Space at the family level, a co-admin feature is provided.

- Co-admin Profile Card List
  - Each card displays the co-admin's profile picture, nickname, and self-introduction.
  - There is a delete button on the card, which, when pressed, removes the co-admin rights for that user.
- Add Co-admin Button
  - Pressing this button brings up a dialog box prompting the user to enter the nickname of the co-admin they wish to add.
  - If the entered nickname exists in the system, that user will be added as a co-admin. If the nickname does not exist, the user will be prompted to re-enter a valid nickname.
  - Once a co-admin is added, a push notification is sent to that user.
  - When the newly added co-admin logs into the app, if auto-login is set, they are immediately directed to the Cat Hydration Management Space to which they belong.

IV. Development Environment

A. *Choice of software development platform*

We are using windows 11 environment, we are likely to use bash terminal or WSL most of the time. Other tools like vscode or jupyter notebook will be based on windows 11. Windows might not be the best choice for software development environment but it is considered one of the good option and we don't have linux machine so we had no choice. In addition, Visual Studio Code will be used as IDE, and Typescript, Python, and MySQL will be used as languages.

Tool and language	***Reason***
Typescript	Typescript is an open-source programming language that is a super-set of JavaScript. By specifying the static type, type errors can be prevented, high code readability, and errors can be checked during compilation. It is also a class-based object-oriented programming language that can support inheritance, encapsulation, and generator. Since all the team members are familiar with JavaScript, and Typescript is a complementary language to JavaScript, we chose to learn it through this opportunity
React Native	React Native can develop platform applications and is partially compatible native. It was judged that developing Android and IOS applications in native languages was less efficient, and the accessibility of developing React Native was high due to the knowledge of React. In addition, it is a framework that is widely used in the field, so we chose it as the front-end language.
Expo	Expo is a React Native cross platform for development. It was judged that Xcode was not available because we do not have a Mac OS device, and it was inefficient because it was not suitable for the project size to develop using both Android studio and Xcode. With Expo, IOS applications can be developed in Windows as well, and real-time tests can be made with mobile devices held through the Expo application. In addition, it was used because the initial setting was simple and there were many modules needed to develop the Expo SDK.

Node.js	Node.js is commonly used by back-end development beginners because it can quickly process data with an asynchronous event-based architecture and provides most of what is needed in package manager. Most of the team members does not have back-end development experience and have knowledge of JavaScript, so we decide to use Node.js. Using Nest.js that supports Typescript as a framework, we hope to facilitate co-working with front-end developers who use React Native.
MySQL	We need a database to store the user information, user's cat information, list of co-manager, and water consumption data. MySQL is an open-source relational database system language, we decide to use because all team members have used it in the "Database System" class.
Amazon Web Services(AWS)	To back up and manage databases by building and hosting cloud-based servers, we intend to use Amazon Relational Database Service (RDS) and Amazon SageMaker to manage machine learning services. With SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. We can get a 12-month Free Tier also a reason for choice.
Python 3.11	Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Because Python is simple to use and it can be used in Machine Learning development, its popularity in AI and ML development is very high. We will be using libraries like matplotlib for visualization, tensorflow/pytorch for image classification, scikit-learn for data analysis, and all the rest of the necessary libraries like pandas or numpy etc.

Cost Estimation
We will use open source if possible, and servers will also use AWS Free Tier plans to avoid incurring development costs.

B. *Software in use*

a. *Visual Studio Code(1.83.1)*
Visual Studio Code (VSCode) is an open-source IDE and can be used in various environments such as Windows, macOS, and Linux. In addition to code editing, it supports various development tasks such as debugging, version management, and terminal access, also there are many extension plug-ins that can increase development productivity with plug-ins such as Prettiers. We chose this IDE because the OS of laptops used by team members is different, also it would be easy for everyone to use the same IDE for smooth cooperative working.

b. *SKT NUGU*
SKT NUGU is an artificial intelligence(AI) service based on voice recognition. It was intended to increase the user's convenience by notifying the amount of water consumption of cats through the voice of the water supplier even without accessing the application. SKT NUGU has the advantage of providing guidelines for developers to utilize, allowing them to mount artificial intelligence-based functions easily and quickly, and being able to test without NUGU devices in a GUI environment called Play Builder.

c. *Git hub*
A version management system based on Git that allows multiple developers to work simultaneously, track changes, and avoid conflicts. The remote storage provided enables collaboration online, and facilitates communication between team members, code review, and issue management. We hope to use GitHub to prevent version problems during collaboration and to check and develop each other's code.

d. *Database Management System*
A database management system is essential to manage the information of the user, the user's cats, and cats' water consumption data. To easily and securely store the information associated with it, the database needs to be designed and built.

e. *Figma*
Figma is a web-based UI/UX design tool. It supports many designers and developers to collaborate in real-time, and does not require installation. It is easy to build and manage reusable components, making it possible to maintain a consistent design pattern, which also contributes to increasing developer productivity. In addition, it is available for free and is not difficult to use, so we chose it as a software design tool.

f. *Open CV*
This is a library of programming functions mainly for real-time computer vision. We found other alternatives like YOLOv8. However, it is unclear whether it could be used or not due to my limitation on knowledge and time. For now, we are looking to use built-in Haar Cascade alorithm for object detection.

C. Task distribution

Name	***Role***	***Responsibilities***
Choi Chansol	Front-end	The role involves developing mobile apps for Android and iOS using React Native. It includes UI/UX design, feature integration, and communication with the backend server via RESTful API for seamless functionality.
Ann Jukyung & Lee Yunsun	Back-end	The responsibilities for this role encompass various aspects of software development and database management. This includes database design and management, which involves overseeing and organizing user information, device information, and statistical data. Additionally, this position involves server-client communication. This multifaceted role entails not only creating and managing the database structure but also maintaining efficient server-client communication for a seamless and responsive user experience.
Hong Jun Ggi	AI	The development environment for this project comprises Python 3.11 within a WSL (Windows Subsystem for Linux) setup. The core objective involves individual cat recognition based on unique nose prints. To achieve this, various libraries are employed, including OpenCV 4 for image processing and the likely integration of scikit-learn (sklearn) for machine learning capabilities. Additionally, for data visualization and analysis, visualization libraries like Matplotlib are utilized. Furthermore, the project explores the realm of generative AI (OpenAI API), possibly in collaboration with SKT NUGU, to enhance its capabilities and further its objectives.

V. SPECIFICATIONS

A. Database Structure

The list below is the tables and schemes for MulMeokNyang database. When creating tables that need to be created in advance and cat's drink measurement management spaces, there are tables that need to be created dynamically.

Tables that need to be generated in advance include a user table to store data for membership users and a session table to store data related to the automatic login function.

Tables that need to be dynamically generated include a management_space table to store basic data related to management space, a cat_in_management_space table to store basic data for cats to manage in management space, and a cat_hydration_statistics table to store data related to the drinking volume of a specific cat. Tables that need to be created dynamically are created together sequentially, and the table name includes the values of the spaceId and catId variables. This allows you to dynamically create and associate tables by management space and individual cat.

1) Lists of tables that must be created in advance:

- user table
  - user_email : primary key, not null, unique
  - user_pw : not null
  - user_name : not null
  - user_phonenum : not null, unique
  - user_profile_photo
  - user_nickname : not null, unique
  - user_introduction
  - management_space_id : default ''(Empty String)
- session table
  - session_id : primary key, not null, unique
  - user_email : foreign key, not null, unique

2) Lists of tables that are going to be created statically:

- management_space_${spaceId} table
  - management_space_id : primary key, default ${spaceId}
  - main_manager_user_email : foreign key, not null
  - co_managers_user_email : JSON, default '{}'
- cat_in_management_space_${spaceId} table
  - session_id : primary key, not null, unique
  - cat_id : primary key, auto_increment, default 0
  - cat_profile_photo : not null
  - cat_name : not null
  - cat_age : not null
  - cat_weight : not null
  - cat_photos : JSON, default '{}'
  - cat_feedstuff_daily_consumption : not null
  - cat_feedstuff_moisture_content : not null88
  - is_hydration_auto : not null
  - cat_goal_hydration : not null

- cat_hydration_statistics_${catId} table
  - date : primary key, CURDATE()
  - day : default DAYOFWEEK(NOW())
  - goal_hydration : not null
  - actual_hydration : default 0
  - hydration_guage : default 0

B. Navigation

In our application, it uses the Navigation module from the react-navigation package.

1) Main Navigation:

- Route List
  - Start
  - Login
  - LocalSignUp
  - UserProfileRegistration
  - Find
    - FindEmail
    - FindEmailResult
    - FindPw
  - HowToGoSpace
  - PendingCoManagerAddition
  - DeviceRegistration
  - CatInfoRegistration
    - CatProfileRegistration
    - CatPhotosRegistration
    - CatFeedStuffRegistration
    - CatHydrationRegistration
  - Main
  - HydrationStatistics

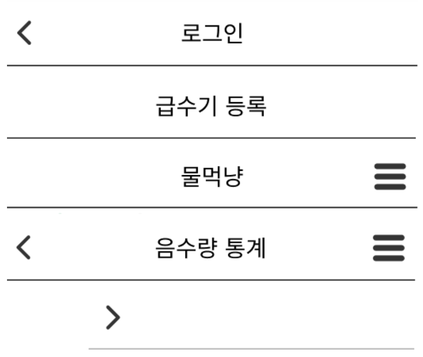
2) Drawer Navigation:

- Route List
  - UserProfileModification
  - CatInfoRegistration
    - CatProfileRegistration
    - CatPhotosRegistration
    - CatFeedStuffRegistration
    - CatHydrationRegistration
  - CatInfoModification
    - CatProfileModification
    - CatFeedStuffModification
    - CatHydrationModification
  - CoManagerSet

C. Frequently Used Custom Component

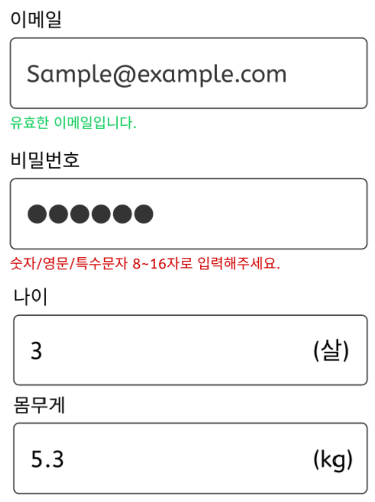
The following five Component are Custom Component used throughout our application.

1) TopBar Component:



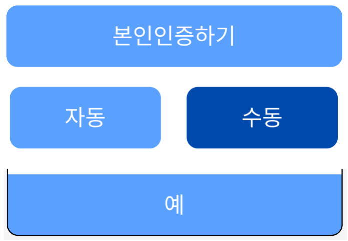
- The title of the current screen appears in the center, and depending on what the current screen is, go back icon and menu icon are visible.
- It's also used in Drawer Navigation, where it turns into a close icon rather than a backward one.

2) InputContainer Component:



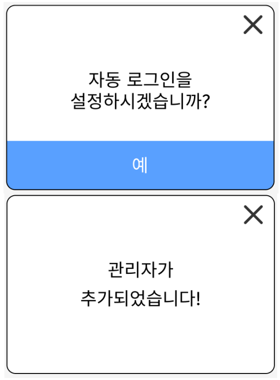
- Used on the Login, Sign Up, and Information Registration page.
- The title on the top of the page represents what that input page is for and if input requirement is not satisfied, msg will deliver information on how to satisfy those input requirements.
- Input values such as password and password verification are made invisible.
- If the input window requires a unit, the unit is shown on the right.
- On each screen, it passes the required validation function to properly proceed with the validation. If the validation fails, the form submit button is not activated.

3) Button Component:



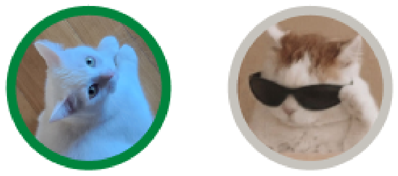
- Title appear and indicate which button this is inside the button.
- Pressing the button changes the color to navy.
- When it is triggered by Alert Component, it replaces the border raidus to 0.
- The button type includes a button to move to the specific page and a button to select a value. This is distinguished by passing the goRoute for screen movement in method property and the selectOpt value for value selection.

4) Alert Component:



- Before setting up auto login or submitting a form, it appears in the form with the button.
- Once the data is registered or modified in the database after submitting the form, the database notifies you that the processing is complete.
- If you need to move to the specific page, when you press the button or close the icon, it passes the route name of the destination page to the goRoute property.

5) CatProfileList Component:

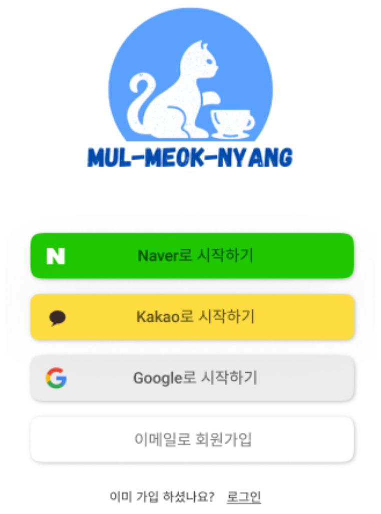


- When you select a cat to display drink amount information on the Main Screen and Hydration Statistics Screen or when you select a cat to modify or delete information, CatProfileList appears.
- When the number of cats increases that the page overflows, scroll feature is available.
- When a cat is selected, the catId value of that cat is stored in the global variable currentSelectedCat.

D. Components and Configuration of each pages and Flow

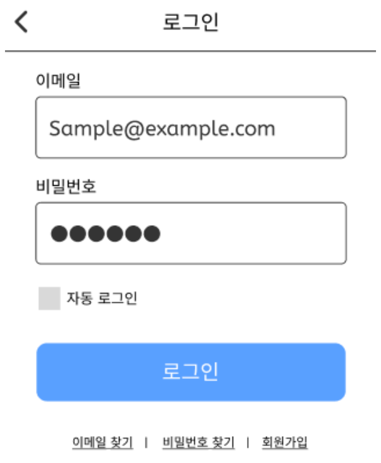
The API that requires a detailed description of the API's call and processing process is described in detail in Section E.

1) Start Screen:



- Components Structure of Screen
  - Image Component
  - SignUpButton Component
    - ∗ Quick SignUp and Local SignUp is distinguished with method property.
  - Text Component
  - UnderlineTextButton Component
- Flow of Screen
  - When the app starts, it makes sure that there is a cookie in the React Native App storage.
  - If cookie exists, it calls the AutoLogin API because user has auto-login enabled.
  - When the button '_____로 시작하기' is pressed, QuickSignUp API is called.
  - When the button '이메일로 회원가입' is pressed, it moves to the LocalSignUp Screen.
  - When the button '로그인' is pressed, it moves to the Login Screen.

2) Login Screen:

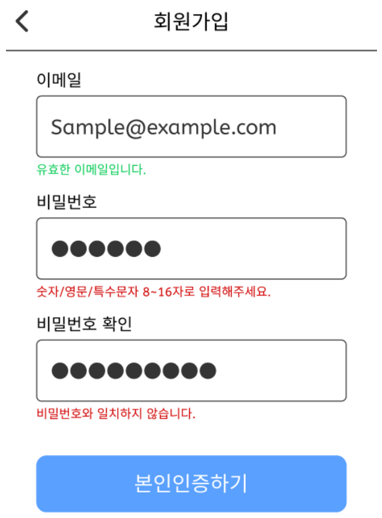


Sample@example.com

●●●●●●

- Components Structure of Screen
  - TopBar Component
  - InputContainer Component
    - ∗ On the Login screen, only checkEmpty validation is performed, and the results of the check is not displayed below the input box.
  - AutoLoginCheckbox Component
  - Button Component
  - UnderlineTextButton Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Enter your email and password, select whether to log-in automatically, and press the 'Log in' button to call the Login API.
  - When the button '<u>이메일 찾기</u>' is pressed, moves to the FindEmail Screen.
  - When the button '<u>비밀번호 찾기</u>' is pressed, moves to the FindPw Screen.
  - When the button '<u>회원가입</u>' is pressed, moves to the LocalSignUp Screen.

3) LocalSignUp Screen:



Sample@example.com

유효한 이메일입니다.

●●●●●●

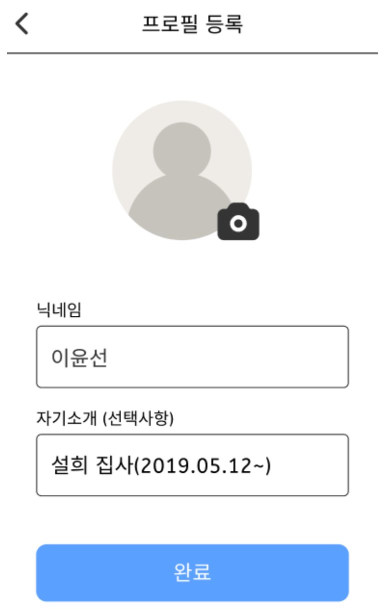
숫자/영문/특수문자 8~16자로 입력해주세요.

●●●●●●●●●

비밀번호와 일치하지 않습니다.

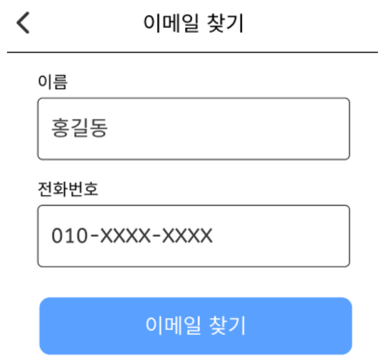
- Components Structure of Screen
  - TopBar Component
  - InputContainer Component
    - ∗ On the LocalSignUp Screen, checkEmail, checkPw, checkConrefmPw validation is performed, and the results are displayed below the input box.
  - Button Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - After entering e-mail, password, and password verification, press the '본인인증하기' button to call the LocalSignUp API.

4) UserProfileRegistration Screen:



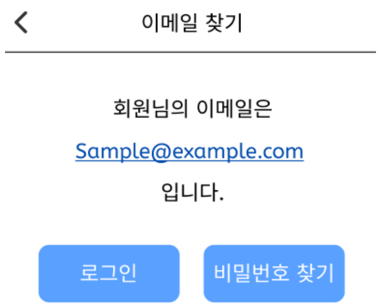
- Components Structure of Screen
  - TopBar Component
  - PhotoRegist Component
    - ∗ Uses react-native-image-picker package.
  - InputContainer Component
    - ∗ On the UserProfileRegistration Screen, only the nickname performs a checkEmpty validation, and the result of the check is not displayed below the input box.
  - Button Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Press the camera button to select a picture from the gallery, or to register a profile picture by taking a picture with the camera.
  - Enter the nickname, which is mandatory, and press the '완료' button to call the RegisterUserProfile API.

5) FindEmail Screen:



- Components Structure of Screen
  - TopBar Component
  - InputContainer Component
    - ∗ On the FindEmail screen, checkEmpty, checkPhoneNum validation is performed, and the results of the inspection are not displayed below the input box.
  - Button Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Enter your name and phone number, and press the '이메일 찾기' button to call the FindEmail API.

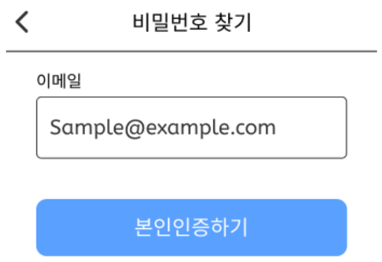
6) FindEmailResult Screen:



회원님의 이메일은
Sample@example.com
입니다.

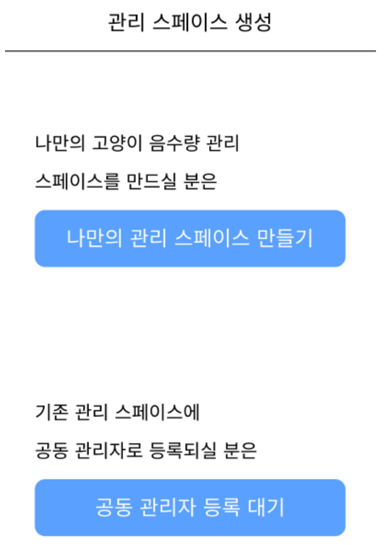
- Components Structure of Screen
  - TopBar Component
  - Text Component
    - ∗ Displays property userEmail value.
  - Button Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Press the '로그인' button to go to the Login Screen.
  - Press the '비밀번호 찾기' button to go to the FindPw Screen.

7) FindPw Screen:



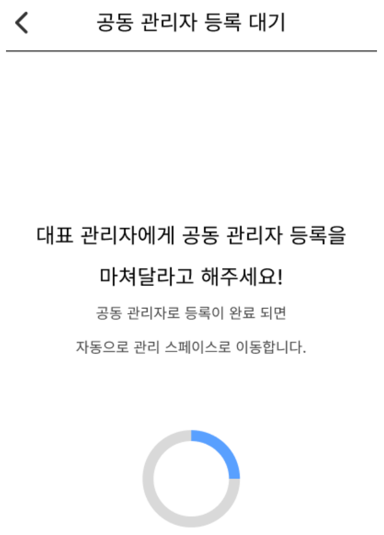
- Components Structure of Screen
  - TopBar Component
  - InputContainer Component
    - ∗ On the FindPw screen, a checkEmail validation is performed, and the results are displayed below the input box.
  - Button Component
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Enter an email and press the '본인인증하기' button to call the FindPw API.

8) HowToGoSpace Screen:



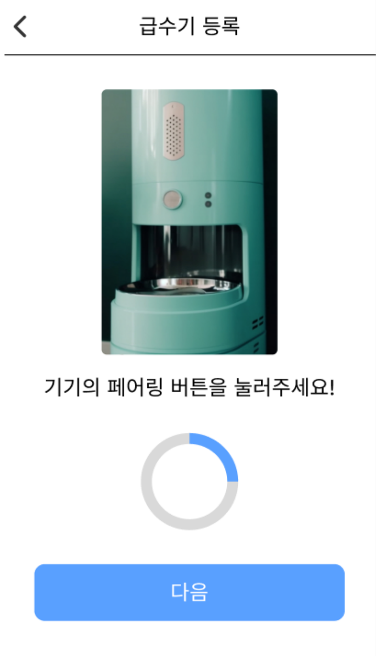
- Components Structure of Screen
  - TopBar Component
    - If there was a go-back button, the go-back button will not appear on the HowToGoSpace screen because it will be moved to the Start, Login screen while logged in.
  - Text Component
  - Button Component
- Flow of Screen
  - Press the '나만의 관리 스페이스 만들기' button to go to the DeviceRegistration Screen.
  - Press the '공동 관리자 등록 대기' button to go to the PendingCoManagerAddition Screen.

9) PendingCoManagerAddition Screen:

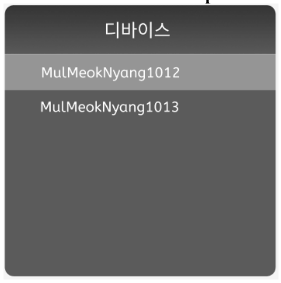

공동 관리자 등록 대기
대표 관리자에게 공동 관리자 등록을
마쳐달라고 해주세요!
공동 관리자로 등록이 완료 되면
자동으로 관리 스페이스로 이동합니다.

- Components Structure of Screen
  - TopBar Component
  - Text Component
  - Loading Component
- Flow of Screen
  - Press the Back button to call the clearInterval function and return to the previous Screen.
  - Use the setInterval function to call the GetManagementSpaceId API every 5 seconds until you are registered as a co-administrator.

10) DeviceRegistration Screen:

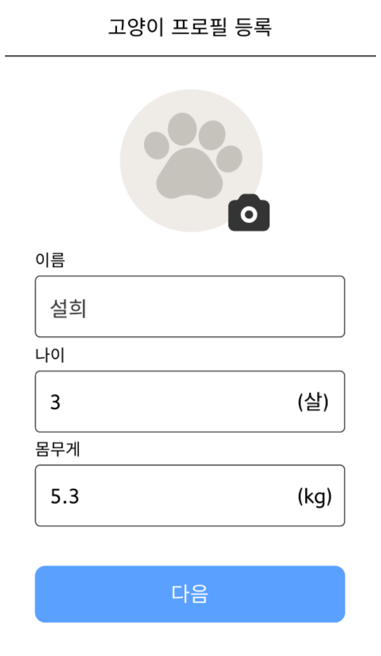


- Components Structure of Screen
  - TopBar Component
  - Image Component
  - Text Component
  - Loading Component
  - Button Component
  - DeviceSelect Component



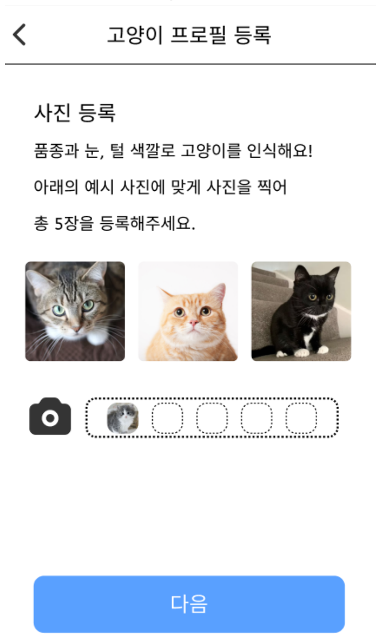
- Flow of Screen
  - Press the Back button to call the clearInterval function and return to the previous Screen.
- Instead of Using API
  - Since the device cannot actually be registered, it shows an animation during loading for 3 seconds, assuming that the pairing button is being pressed.
  - Then, the Device Select Component is displayed and when the user selects Device, the property value of the Loading Component is changed to false and shows the pairing completed animation. Then it goes to the CatProfileRegistration Screen.

11) CatProfileRegistration Screen:



- Components Structure of Screen
  - TopBar Component
  - PhotoRegist Component
  - InputContainer Component
    - ∗ On the CatProfileRegistration screen, checkEmpty, checkNumber validation is performed, and the results of the check are displayed below the input box.
  - Button Component
- Flow of Screen
  - If the user does not have a management space yet, it displays Alert Component that says, "고양이 프로필을 최소한 한 마리 이상 등록 해야 관리 스페이스가 생성됩니다"
  - Press the camera button to select a picture from the gallery, or take a picture with the camera to register a cat profile picture.
  - Enter a name, age, and weight, and press the Next button to save the catName, catAge, and catWeight global variables with the Context API and moves to CatPhotosRegistration Screen.

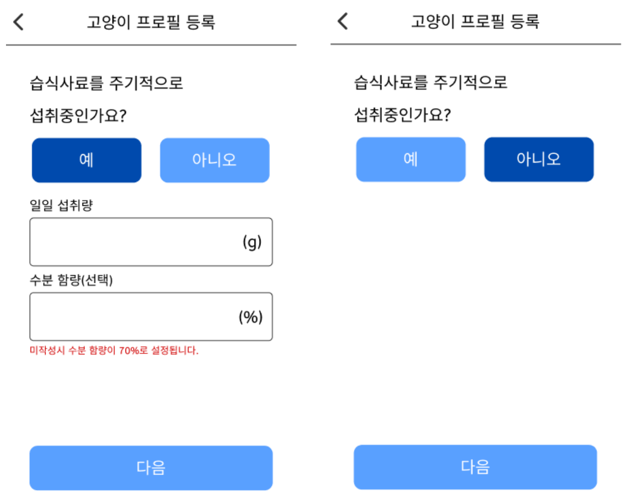
12) CatPhotosRegistration Screen:



품종과 눈, 털 색깔로 고양이를 인식해요!
아래의 예시 사진에 맞게 사진을 찍어
총 5장을 등록해주세요.

- Components Structure of Screen
  - TopBar Component
  - Text Component
  - CatPhotosRegist Component
  - Button Component
    - ∗ User must register 5 photos to activate the button.
- Flow of Screen
  - Press the camera button to select a picture from the gallery or take a picture with the camera to register a cat photo.
  - Press the Go Back button to return to the previous Screen.
  - Register 5 photos, press the ’Next’ button to save the catPhotos global variable with the Context API and navigate to the CatFeedStuffRegistration Screen.

13) CatFeedStuffRegistration Screen:

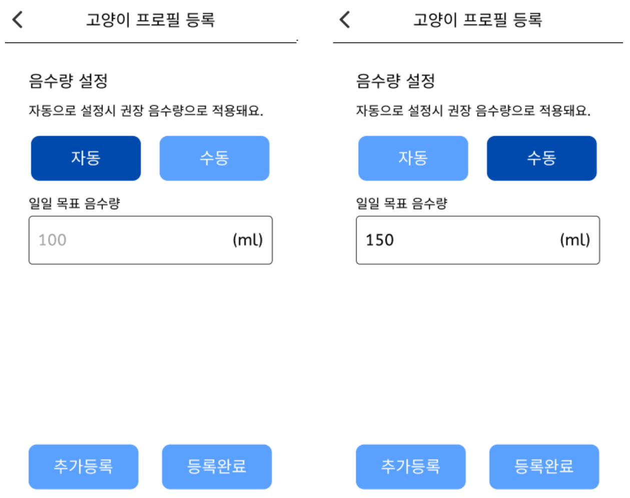


습식사료를 주기적으로
섭취중인가요?
미작성시 수분 함량이 70%로 설정됩니다.

습식사료를 주기적으로
섭취중인가요?

- Components Structure of Screen
  - TopBar Component
  - Text Component
  - Button Component
    - ∗ Press ’예’ to display the daily intake and water intake window.
  - InputContainer Component
    - ∗ On the CatFeedStuffRegistration screen, checkEmpty, checkNumber validation is performed only when ’Yes’ is pressed, and the results are displayed below the input box.
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Press ‘예’ on the intake button and enter daily intake, water content. Then, press Next to save the catFeedStuffDailyConsumption, catFeedStuffMoistureContent global variables in the Context API.
  - Press ’No’ and then press ’Next’ to store 0 in the catFeedStuffDailyConsumption, catFeedStuffMoistureContent global variable.
  - Next, it goes to the CatHydrationRegistration Screen.

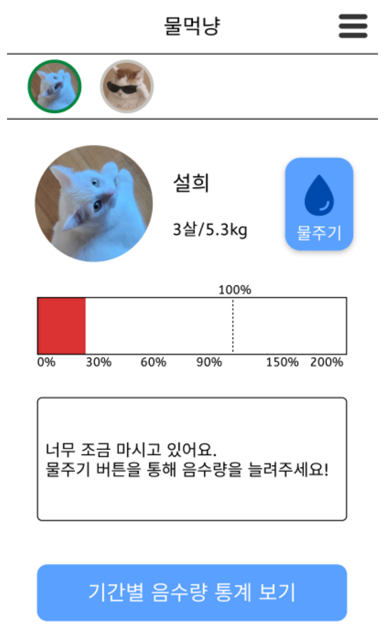
14) CatHydrationRegistration Screen:



자동으로 설정시 권장 음수량으로 적용돼요.

- Components Structure of Screen
  - TopBar Component
  - Text Component
  - Button Component
  - InputContainer Component
    * Press 'Auto' to automatically calculate the recommended water intake and automatically enter it into the input box.
    * On the CatHyditionRegistration screen, checkEmpty, checkNumber validation is performed, and the results of the inspection are displayed below the input box.
- Flow of Screen
  - Press the Go Back button to return to the previous Screen.
  - Press '자동' on how to set the water intake and then press the 'Additional Registration' or 'Registration Completed' button to store true in the isHyditionAuto global variable with the Context API and substitute the values of the global variable catFeedStuffDailyConsumption, catFeedStuffMoistureContent into the recommended drinking volume formula and store the calculated values in the catGoalHydition global variable.
  - Press '수동' and enter the daily target water intake, then press the 'Additional Registration' or 'Registration Completed' button to store false in the global variable isHydrationAuto and save the input in the global variable catGoalHydration.
  - Then, call the CatInfoRegist API.
  - Once you have pressed the '추가 등록' button, it goes back to the CatProfileRegistration Screen.
  - If you press the '등록 완료' button, delete the global variables' value related to cat information and navigate to the Main screen.

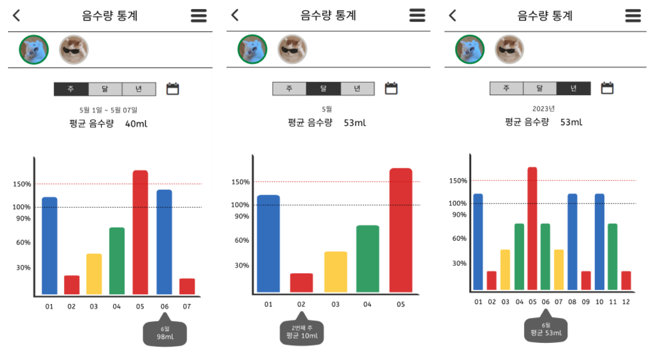
15) Main Screen:



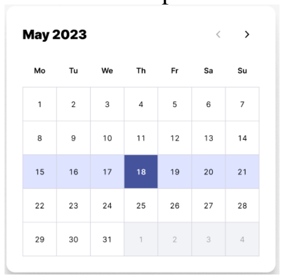
너무 조금 마시고 있어요.
물주기 버튼을 통해 음수량을 늘려주세요!

- Components Structure of Screen
  - TopBar Component
  - CatProfileList Component
  - CatProfile Component
  - HydrationButton Component
  - HydrationGauge Component
    * 0% 2̃9% : Red
    * 30% 5̃9% : Yellow
    * 60% 8̃9% : Green
    * 90% 1̃49% : Blue
    * 150% 2̃00% : Red
  - EvaluationText Component
  - Button Component
- Flow of Screen
  - Call the GetCatProfileList API in the Mount step, process the response, and call the GetCatMainInfo API.
  - Press the menu button to open the Drawer.
  - In the cat profile list at the top, call the GetCatMainInfo API whenever user press a picture of another cat profile that is not currently selected.
  - When you press the '물주기' button, a cat-calling sound comes out from the built-in speaker, and water comes out when the cat approaches the water supply.
  - You will receive a push notification whether the cat drank water or not within 10 minutes.
  - A water intake gauge shows how much the cat has intaken today so far.
  - Different assessments appear depending on the gauge.
  - Press the '기간별 음수량 통계 보기' button to go to the Hydration Statistics Screen.

16) HydrationStatistics Screen:

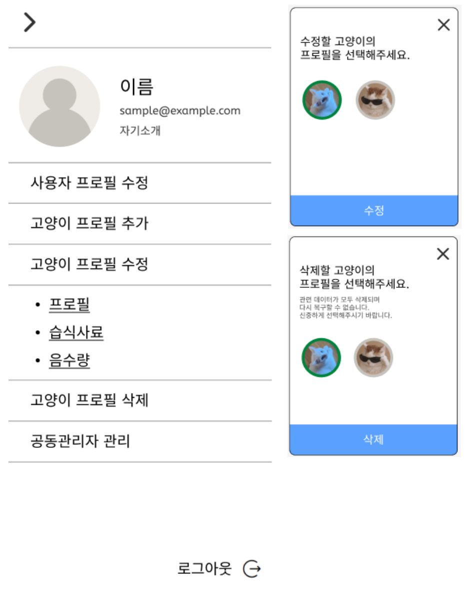


- Components Structure of Screen
  - TopBar Component
  - CatProfileList Component
  - RangeSet Component
  - Calender Component



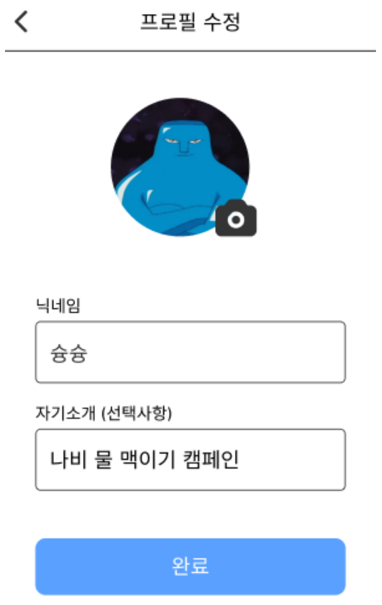
  - AvgText Component
  - BarGraph Component
    - ∗ Bar Component
    - ∗ BarDetail Component
- Flow of Screen
  - GetCatStatistics API is called in the Mount step.
  - Press the Go Back button to return to the previous Screen.
  - Press the menu button to open the Drawer.
  - From the cat profile list at the top, whenever you click cat's picture that is not currently selected, change the statistical period, or select a specific week/month/year in the calendar, GetCatStatistics API is called.
  - For '주', if you select a specific date in the calendar, the week containing that date is selected.
  - Touch a particular bar to display detailed water intake amount.

17) Drawer:


이름
sample@example.com
자기소개
사용자 프로필 수정
고양이 프로필 추가
고양이 프로필 수정
프로필
습식사료
음수량
고양이 프로필 삭제
공동관리자 관리
로그아웃
수정할 고양이의 프로필을 선택해주세요.
수정
삭제할 고양이의 프로필을 선택해주세요.
관련 데이터가 모두 삭제되며 다시 복구할 수 없습니다. 신중하게 선택해주시기 바랍니다.
삭제

- Components Structure of Screen
  - TopBar Component
  - UserProfile Component
  - TextContainerButton Component
    - ∗ For users with only one cat registered, the Delete Cat Profile button does not appear.
  - UnderlineTextButton Component
  - LogoutButton Component
  - LongAlert Component
    - ∗ IconButton Component
    - ∗ Title Component
    - ∗ Text Component
    - ∗ CatProfileList Component
    - ∗ Button Component
- Flow of Screen
  - Press the Close button to close the Drawer.
  - Press the '사용자 프로필 수정' button to go to the UserProfileModification Screen.
  - Pressing the '프로필', '습식사료', and '음수량' buttons under '고양이 프로필 수정' displays LongAlert Component to select the cat to modify the information.
  - Select the cat you want to modify and press the '수정' button to go to the Cat_____Modification Screen.
  - When you press the 'Delete Cat Profile' button, a LongAlert Component will appear to select the cat to delete all the information.
  - Select the cat you want to delete and press the '삭제' button to call the DeleteCatInfo API.
  - Press the '공동관리자 관리' button to navigate to the CoManagerSet Screen.
  - If user presses the Logout button, the Alert Component that says 'Do you really want to log out?' apeears and when user proceed with 'Yes', the Logout API is called.

18) UserProfileModification Screen:


프로필 수정

닉네임

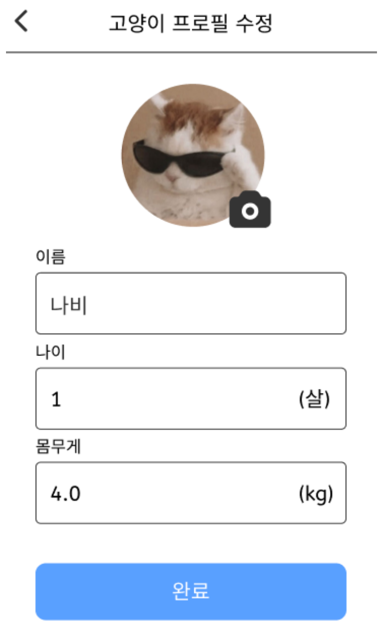
슝슝

자기소개 (선택사항)

나비 물 맥이기 캠페인

- Components Structure of Screen
  - Identical to UserProfileRegistration Screen.
- Flow of Screen
  - Call the GetUserProfile API in the Mount step.
  - Call the ModifyUserProfile API when the '완료' button is pressed.
  - The rest is the same as the UserProfileRegistration Screen.

19) CatProfileModification Screen:


고양이 프로필 수정

이름

나비

나이

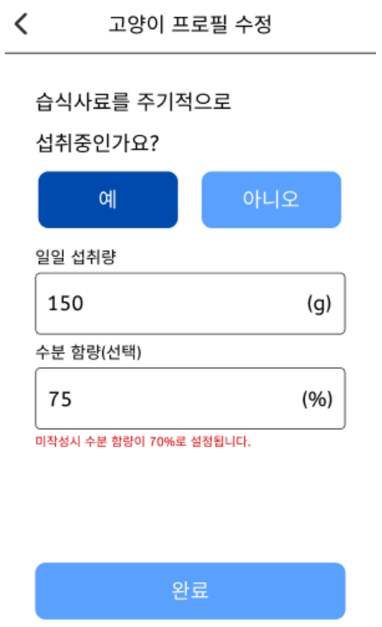
1 (살)

몸무게

4.0 (kg)

- Components Structure of Screen
  - Identical to CatProfileRegistration Screen.
- Flow of Screen
  - Call the GetCatProfile API in the Mount step.
  - Call the ModifyUserProfile API when the '완료' button is pressed.
  - The rest is the same as the CatProfileRegistration Screen.

20) CatFeedStuffModification Screen:



습식사료를 주기적으로 섭취중인가요?

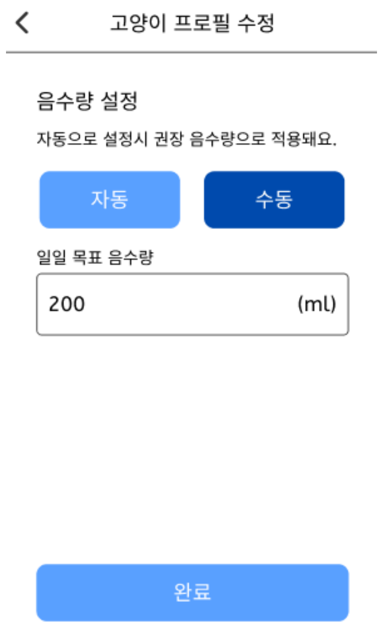
일일 섭취량: 150 (g)

수분 함량(선택): 75 (%)

미작성시 수분 함량이 70%로 설정됩니다.

- Components Structure of Screen
  - Identical to CatFeedStuffRegistration Screen.
- Flow of Screen
  - Call the GetCatFeedStuffAPI in the Mount step.
  - Call the ModifyCatFeedStuff API when the '완료' button is pressed.
  - The rest is the same as the CatFeedStuffRegistration Screen.

21) CatHydrationModification Screen:



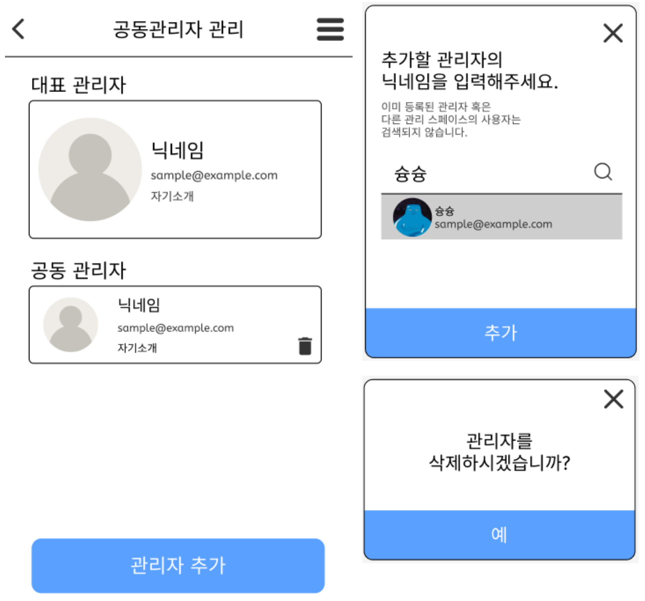
음수량 설정

자동으로 설정시 권장 음수량으로 적용돼요.

일일 목표 음수량: 200 (ml)

- Components Structure of Screen
  - Identical to CatHydrationRegistration Screen.
- Flow of Screen
  - Call the GetCatHydration in the Mount step.
  - Call the ModifyCatHydration API when the '완료' button is pressed.
  - The rest is the same as the CatHydrationRegistration Screen.

22) CoManagerSet Screen:



- Components Structure of Screen
  - TopBar Component
  - MainManagerCard Component
  - CoManagersCardList Component
    * CoManagerCard Component
  - Button Component
  - LongAlert Component
    * IconButton Component
    * Title Component
    * Text Component
    * Search Component
    * SearchedUser Component
    * Button Component
- Flow of Screen
  - Call the GetManagerList API in the Mount step.
  - Pressing the Add Administrator button to displays the LongAlert Component. After entering a nickname and when you press the search icon, it calls the UserSearch API.
  - Select the administrator user wants to add and press the Add button to call the AddCoManager API.
  - On the Administrator Screen, the Trash icon appears in the Co Manager Card, and when you press the '예'' button in the Manager Delete Alert Component, the DeleteCoManager API is called.

E. API Usage Logic

1) AutoLogin API:

- Client send HTTP Request to Server
  - Case 1) Users with auto login disabled
    * Set the userEmail value to HTTP Request and send the Request to the Server.
  - Case 2) Users with auto login enabled
    * Set the cookie that contains the Session ID in the React Native app storage to HTTP Request and send the request to the Server.

- HTTP Request process in the sever, HTTP Response from server to client
  - Case 1) Users with auto login disabled
    * If the value set in the HTTP Request is userEmail, create a session and add a record along with userEmail to the session table.
    * Configure Session ID and generate a cookie.
    * When querying the record in the user table with userEmail, if there is a value for management_space_id, it is saved in the variable managementSpaceId, and if not, it is saved as null.
    * Set userEmail, managementSpaceId, and Cookie in HTTP Response to send a response to the Client.
  - Case 2) Users with auto login enabled
    * The session ID stored in the cookie set in the HTTP Request inquires the record in the session table and stores the user_email value in the userEmail variable.
    * When querying the record in the user table with userEmail, if there is a value for management_space_id, it is saved in the variable managementSpaceId, and if not, it is saved as null.
    * Set userEmail, managementSpaceId to HTTP Response and send a response to the Client.

- HTTP Response Processing in Client side
  - The userEmail value set in the HTTP Response is stored as a global variable userEmail using the Context API, and if the managementSpaceId value is not null, it is also stored as a global variable managementSpaceId.
  - If the cookie is set in HTTP Response, it saves the cookie in the React Native App storage before moving into the screen.
  - If the user have a managementSpaceId value, go to the Main screen, and if the value is null, go to the HowToGoSpace screen.

2) QuickSignUp APIs : NaverSignUp, KaKaoSignUp, GoogleSignUp:

- HTTP Response for Server → Client
  - Using Naver, Kakao, and Google's simple login APIs and with the user profile information provided, userEmail values are first stored in variables.
  - Check the record in the user table with the userEmail value, store true in the userExists variable if there is a registered user, and store false if not.
  - Case 1) Registered Users → Easy Login
    the userExists value is true, check the record in the user table with the userEmail value,

and if there are user_nickname and management_space_id values, store them in the userNickname and managementSpaceId variables, and if not, store null.When login is successful, store true to the variable loginSuccess. Set loginSucess, userEmail, userNickname, managementSpaceId to HTTP Response and send it to the Client.

  - Case 2) Unregistered users → Easy membership registration
    - If the userExists value is false, the remaining profile information is also stored in the variable, and add record containing userEmail, userPw, userName, and userPhoneNum values to the user table.
    - When registeration is successful, store true to the variable registerSuccess
    - Set registerSuccess, userEmail to HTTP Response and send it to Client.

- HTTP Response Processing in Client side
  - The userEmail value set in HTTP Response is stored as a global variable userEmail using the Context API.
  - Case 1) Registered Users → Easy Login
    - If loginSuccess is set in HTTP Response, check whether there is a value stored in userNickname or null is stored.
    - The existence of the userNickname value means that the user profile has been registered.
    - Case 1-1) Users that have profile registration completed
      - Display an Alert Component that says, '자동로그인을 설정하시겠습니까?'.
      - If '예', follow case 1-1-1 below.
      - Then, if the managementSpaceId value set in HTTP Response is not null, it is stored as a global variable managementSpaceId using the Context API.
      - If the user have a managementSpaceId value, go to the Main screen, and if the value is null, go to the HowToGoSpace screen.
      - Case 1-1-1) User who wants auto login
      - Set the userEmail value to HTTP Request and send the request to the Auto Login API in the server.
      - This corresponds to Case 1 of the AutoLogin API.
      - Store the cookie set in the HTTP Response sent from the AutoLogin API in the React Native App Storage.
      - Case 1-2) User who needs to register for a user profile
      - If the userNickname value is null, go to the UserProfileRegistration screen.
  - Case 2) Unregistered users → Easy membership registration
    - If registerSuccess is set in HTTP Response, go to the UserProfileRegistration screen.

3) Login API:

- HTTP Request From Client to Server
  - Set userEmail, userPw, and autoLogin values to HTTP Request and send a request to the server.

- HTTP requests process in Server, HTTP Response from Server to Client
  - Case 1) Registered User
    - With the userEmail and userPw values set in the HTTP Request, look up the record in the user table. If there is a registered user, store true in the userExists variable and if not, store false.
    - If the record has user_nickname and management_space_id values, store them in the userNickname and managementSpaceId variables, respectively, and if not, store null.
    - Case 1-1) Who checked the Auto Login check box
      - If the userExists value is true and the AutoLogin value set in the HTTP Request is also true, the userEmail value is set in the HTTP Request and the request is sent to the AutoLogin API.
      - This corresponds to Case 1 of the Auto Login API.
      - The cookie set in the HTTP Response transmitted from the Auto Login API is stored in the variable.
      - Set userEmail, userNickname, managementSpaceId, and Cookie in HTTP Response to send a response to the Client.
    - Case 1-2) Users who did not check the Auto Login check box
      - If the AutoLogin value set in the HTTP Request is false, set userEmail, userNickname, and managementSpaceId to HTTP Response and send a response to the Client.
  - Case 2) Unregistered User
    - If the userExists value is false, set the userExists value to HTTP Response and send a response to the Client.

- HTTP Response process in Client side
  - Case 1) Registered User
    - If userEmail is set in HTTP Response, the userEmail value is saved as a global variable userEmail using the Context API.
    - Check whether there is a set userNickname value in HTTP Response or null.

        * The existence of the userNickname value means that the user profile has been registered.
        * Case 1-1) User that has completed profile registration
            · If the managementSpaceId value set in HTTP Response is not null, it is stored as a global variable managementSpaceId.
            · If the cookie is set in HTTP Response, store the cookie in the React Native App storage.
            · Then, if the user has a managementSpaceId value, it moves to the Main screen, and if the value is null, it moves to the HowToGoSpace screen.
        * Case 1-2) User who needs to register user profile
            · If the userNickname value is null, display an Alert Component that says "자동 로그인 설정은 사용자 프로필 등록을 한 다음 사용이 가능합니다." and navigate to the UserProfileRegistration screen.
    - Case 2) Unregistered User
        * If the userExists value set in HTTP Response is false, it displays an Alert Component that says "해당되는 사용자가 없습니다." and keeps the Login screen.

4) LocalSignUp API:

- HTTP Request from Client to Server
    - Set the userEmail, userPw value to HTTP Request and send the request to the server.

- HTTP Request Process in Server side, HTTP Response from Server to Client
    - Retrieve a record from the user table using the userEmail value set in the HTTP Request. If a registered user exists, store true in the userExists variable; otherwise, store false.
    - Case 1) Registered User
        * If the userExists value is true, set userExists in the HTTP Response and send the response to the client.
    - Case 2) Unregistered User
        * If the userExists value is false, store the userEmail and userPw set in the HTTP Request in variables.
        * Next, send a request to the NICE Auth API to proceed with identity verification.
            · Case 2-1) Identity Verification Completed
            · If the successAuth in the HTTP Response sent by the NICE Auth API is true, add a record to the user table with the userName and userPhonenum values along with the userEmail and userPw variables.
            · Store true in the stepOneDone variable.
            · Case 2-2) Identity Verification Failed
            · If the successAuth in the HTTP Response sent by the NICE Auth API is false, store false in the stepOneDone variable.
            · Set stepOneDone and userEmail in the HTTP Response and send the response to the client.

- Client-side HTTP Response Handling
    - Case 1) Registered User
        * If the HTTP Response has userExists set, display an Alert Component with the message "이미 회원 가입된 이메일입니다." and navigate to the Login screen.
    - Case 2) Unregistered User
        * Case 2-1) Identity Verification Completed
            · If the stepOneDone value in the HTTP Response is true, save the userEmail value as a global variable using Context API and navigate to the UserProfileRegistration screen.
        * Case 2-2) Identity Verification Failed
            · If the stepOneDone value in the HTTP Response is false, display an Alert Component with the message "본인 인증에 실패하셨습니다." and stay on the LocalSignUp screen.

5) RegistUserProfile API:

- Client → Server HTTP Request
    - Send userProfilePhoto, userNickname, userIntroduction values, along with the userEmail stored as a global variable, in the HTTP Request from the Client to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
    - Retrieve a record from the user table using the userNickname value set in the HTTP Request. If the nickname already exists, store true in the nicknameExists variable.
    - Case 1) Duplicate Nickname
        * If nicknameExists is true, set nicknameExists in the HTTP Response and send it to the Client.
    - Case 2) Non-duplicate Nickname
        * If nicknameExists is false, retrieve a record from the user table using the userEmail value set in the HTTP Request, and add userProfilePhoto, userNickname, and userIntroduction values.
        * Save true in the stepTwoDone variable.

    * Set stepTwoDone in the HTTP Response and send it to the Client.

- Client-side HTTP Response Handling
  - Case 1) Duplicate Nickname
    * If the HTTP Response has nicknameExists set to true, display an Alert Component with the message "이미 존재하는 닉네임입니다." and stay on the UserProfileRegistration screen.
  - Case 2) Non-duplicate Nickname
    * If the HTTP Response has stepTwoDone set, display an Alert Component with the message "사용자 프로필 등록이 완료되었습니다. 자동로그인을 설정하시겠습니까??".
    * If the user selects '예' follow the instructions in case 2-1.
    * Then, navigate to HowToGoSpace Screen.
    * Case 2-1) User Who Wants Automatic Login
      · Set the userEmail value in the HTTP Request and send a request to the Auto Login API on the Server.
      · This corresponds to Case 1 in the Auto Login API.
      · Store the Cookie received from the HTTP Response of the Auto Login API in the React Native App's storage.

6) FindPw API:

- Client → Server HTTP Request
  - userEmail 값을 HTTP Request에 설정하여 Server에 Request를 전송한다.
  - Set userEmail value in the HTTP Request and send Request to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
  - Retrieve a record from the user table using the userEmail value set in the HTTP Request. If a registered user exists, store true in the userExists variable; otherwise, store false.
  - Case 1) Registered User
    * If userExists is true, send a request to the NICE Auth API for identity verification.
    * Case 1-1) Identity Verification Completed
      · If the successAuth in the HTTP Response received from the NICE Auth API is true, retrieve the user_pw from the user table using userName and userEmail, and store it in the userPw variable.
      · Use the express-email package to send the userPw to the user's email.
      · Once the email has been sent, store true in the sendMail variable.
      · Set sendMail in the HTTP Response and send it to the client.
    * Case 1-2) Identity Verification Failed
      · If the successAuth in the HTTP Response from the NICE Auth API is false, set successAuth in the HTTP Response and send it to the client.
  - Case 2) Unregistered User
    * If userExists is false, set userExists in the HTTP Response and send the response to the client.

- Client-side HTTP Response Handling
  - Case 1) Registered User
    * Case 1-1) Identity Verification Completed
      · If the HTTP Response has sendMail set, display an Alert Component with the message "비밀번호가 이메일로 전송되었습니다" and navigate to the Login screen.
    * Case 1-2) Identity Verification Failed
      · If the successAuth value in the HTTP Response is false, display an Alert Component with the message "본인 인증에 실패하셨습니다." and stay on the FindPw screen.
  - Case 2) Unregistered User
    * If the userExists value in the HTTP Response is false, display an Alert Component with the message "회원 가입된 이메일이 아닙니다." and stay on the FindPw screen.

7) RegistCatInfo API:

- Client → Server HTTP Request
  - Case 1) First-time Cat Registration (For users with no existing management space):
    * Send the userEmail, catProfilePhoto, catName, catAge, catWeight, catPhotos, catFeedStuffDailyConsumption, catFeedStuffMoistureContent, isHydrationAuto, and catGoalHydration values that are saved from global variables in the HTTP Request to the Server.
  - Case 2) Additional Cat Registration (For users with an existing management space):
    * Include the managementSpaceId value from global variables in the HTTP Request and send it to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
  - Case 1) First-time Cat Registration (For users with no existing management space):
    * Step 1) If the managementSpaceId is not set in the HTTP Request, generate a 10-digit random

value using the Math.random() method and store it in the spaceId variable.
    * Step 2) ynamically create the management_space_${spaceId} table.
    * Step 3) Add a record with spaceId and the userEmail value from the HTTP Request.
    * Step 4) Dynamically create cat_in_management_${spaceId} table.
    * (common) Step 5) Add records to the cat_in_management_${spaceId} table with the values of cat information from the HTTP Request.
    * (common) Step 6) Retrieve the cat_id from cat_in_management_${spaceId} table based on the catName value from the HTTP Request and store it in the catId variable.
    * Step 7) Dynamically create cat_hydration_statistics_${catId} table.
    * (common) Step 8) Add a record to the cat_hydration_statistics_${catId} table with the value of catGoalHydration from the HTTP Request.
    * Step 9) Add the spaceId value to the management_space_id column in the user table based on the retrieved information, userEmail.
    * Step 10) Set spaceId in the HTTP Response and send it to the Client.
  - Case 2) Additional Cat Registration (For users with an existing management space):
    * Step 1) If the managementSpaceId is set in the HTTP Request, store it in the spaceId variable.
    * (common) Step 2) Add records to the cat_in_management_${spaceId} table with the values of cat information from the HTTP Request.
    * (common) Step 3) Retrieve the cat_id from the cat_in_management_${spaceId} table based on the catName value from the HTTP Request and store it in the catId variable.
    * (common) Step 4) Add a record to the cat_hydration_statistics_${catId} table with the value of catGoalHydration from the HTTP Request.
    * Step 5) Store true in the addCatInfo variable and set addCatInfo in the HTTP Response to send it to the Client.

- Client-side HTTP Response Handling
  - Case 1) First-time Cat Registration (For users with no existing management space):
    * If spaceId is set in the HTTP Response, store it as the managementSpaceId global variable using the Context API.

8) GetCatProfileList API:

- Client → Server HTTP Request
  - Send the managementSpaceId value from the global variables in the HTTP Request to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
  - Store the managementSpaceId value from the HTTP Request in the spaceId variable.
  - Query the cat_id values from the cat_in_management_space_${spaceId} table and store them in the catIdArr array variable.
  - Retrieve the cat_profile_photo values only from cat_in_management_space_${spaceId} table and store them in the catProfilePhotoArr array variable.
  - Set the catIdArr and catProfilePhotoArr arrays in the HTTP Response to send them to the Client.

- Client-side HTTP Response Handling, Data Binding
  - Store the catIdArr and catProfilePhotoArr arrays set in the HTTP Response as global variables using the Context API.
  - Inside the Cat Profile List Component, use the map method to display Cat Profile List Components. Pass the key prop as 'i,' catId prop as catIdArr[i], and catProfilePhoto prop as catProfilePhotoArr[i].
  - Set the onFocus prop of the Cat Profile Component with i = 0 which means true (indicating the first Cat Profile Component is selected).
  - Use the Context API to store the catIdArr[0] value in the currentSelectedCat global variable.
  - When clicking on a different cat's profile picture, set the onFocus prop of the respective Cat Profile Component to true.
  - Use the key value of the selected Cat Profile Component to store catIdArr[key] in the currentSelectedCat global variable.
  - Display the image of the currently selected cat, which appears to the left of the catName on the main screen, using catProfilePhotoArr[currentSelectedCat].

9) GetCatMainInfo API:

- Client → Server HTTP Request
  - Send the currentSelectedCat value and managementSpaceId from the global variables in the HTTP Request to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
  - Store the currentSelectedCat value from the HTTP Request in the catId variable, and the managementSpaceId value in the spaceId variable.
  - With the catId value, query the cat_name, cat_age, cat_weight from the

cat_in_management_space_${spaceId} table and store them as catName, catAge, and catWeight variables, respectively.
    - Based on the date column, retrieve the most recent record from the cat_hydration_statistics_${catId} table and store the hydration_guage value in the hydrationGuage variable.
    - Set the catName, catAge, catWeight, and hydrationGuage in the HTTP Response to send them to the Client.

- Client-side HTTP Response Handling, Data Binding
    - Bind the values of catName, catAge, catWeight, and hydrationGuage from the HTTP Response to their respective components.

10) GetCatStatistics API:

- Client → Server HTTP Request
    - Case 1) Statistical range is 'week'
        - ∗ Send the currentSelectedCat value, range, startDate, and endDate (in the format 'YYYY-MM-DD') from global variables in the HTTP Request to the Server.
    - Case 2) Statistical range is 'month'
        - ∗ Send the currentSelectedCat value, range, and month (in the format 'YYYY-MM') from global variables in the HTTP Request to the Server.
    - Case 3) Statistical range is 'year'
        - ∗ Send the currentSelectedCat value, range, and year (in the format 'YYYY') from global variables in the HTTP Request to the Server.

- Server-side HTTP Request Handling, Server → Client HTTP Response
    - Store the currentSelectedCat value from the HTTP Request to the catId variable
    - Case 1) Statistical range is 'week'
        - ∗ If the range value in the HTTP Request is 'week', query records from cat_hydration_statistics_${catId} table between the startDate and endDate, retrieving the date and hydration_guage values and storing them in the hydrationGuageArr array.
            - · The hydrationGuageArr is an array of objects.
            - · Ex) [{date: 'YYYY-MM-DD', hydration_guage: n}, {date: 'YYYY-MM-DD', hydration_guage: n}, . . .]
        - ∗ (common) Declare a newHydrationGuageArr array as a constant and initialize it as an empty array.
        - ∗ Using the map method, iterate over the hydrationGuageArr array.
            - · Extract only the DD portion from element.date using the slice method.
            - · Store it in the newHydrationGuageArr array.
            - · Ex) [{date: 'DD', hydration_guage: n}, {date: 'DD', hydration_guage: n}, . . .]
        - ∗ (common) Set newHydrationGuageArr in the HTTP Response to send it to the Client.

    - Case 2) Statistical range is 'month'
        - ∗ Step 1) If the range value in the HTTP Request is 'month,' create a function called getWeeksOfMonth that returns an array containing the total number of weeks in a specific month and the start and end days of each week.

```
function getWeeksOfMonth(year,
    month) {
  let weeksOfMonth = [];
  let date;
  let day;
  let week = [];
  const lastDay = new Date(year,
      month, 0).getDate();

  for (i = 1; i <= lastDay; i++) {
    date = new Date(year, month -
        1, i);
    day = date.getDay();

    week.push(i);

    if (day === 0 || i === lastDay)
        {
      weeksOfMonth.push(week);
      week = [];
    }
  }

  return weeksOfMonth;
}
```

        - ∗ Step 2) In a two-dimensional array returned by the getWeeksOfMonth function, generate a getIndexOfWeekInArr function that returns the specific value to which array it belongs.

```
function getIndexOfWeekInArr(arr,
    targetValue) {
  for (let i = 0; i < arr.length; i
      ++) {
    if (arr[i].includes(targetValue
        )) {
      return i;
    }
  }
}
```

        - ∗ Step 3) Extract the year and month from the month value in the HTTP Request by using the slice method to get 'YYYY' and 'MM' separately. Then, use the parseInt method to convert them into integers and store them in the year and month variables, respectively.
        - ∗ Step 4) Pass the year and month variables as arguments to the getWeeksOfMonth function.

Call the function and store the returned value in the weeksOfMonth array variable.

    * Step 5) Query records from the cat_hydration_statistics_${catId} table where the date column matches the month value from the HTTP Request. Retrieve the date, day, and hydration_guage values and store them in the hydrationGuageArr array.
    * Step 6) Extract only the 'DD' part from the date attribute of the first element in hydrationGuageArr using the slice method. Then, use the parseInt method to convert it into an integer and store it in the dd variable.
    * Step 7) Pass the weeksOfMonth and dd variables as arguments to the getIndexOfWeekInArr function. Call the function and store the returned (value + 1) in the weekNum variable.
    * (common) Step 8) Declare a newHydrationGuageArr array as a constant and initialize it as an empty array.
    * (common) Step 9) Declare weekHydrationGuage as a let variable and initialize it as an empty object.
    * (common) Step 10) Declare numOfDay, sumOfHydrationGuage, and avgOfHydrationGuage as let variables and initialize them all to 0.
    * Step 11) Iterate over the hydrationGuageArr array using the forEach method.

```
hydrationGuageArr.forEach((e, i,
    arr) => {
    numOfDay = numOfDay + 1;
    sumOfHydrationGuage =
        sumOfHydrationGuage +
        e.hydration_guage;
    if(e.day === 1) {
        avgOfHydrationGuage =
            Math.floor(
            sumOfHydrationGuage /
            numOfDay);
        weekHydrationGuage.week = '
            0' + weekNum; // '01'
        weekHydrationGuage.hydration_guage
            =
            avgOfHydrationGuage;
        newHydrationGuageArr.push(
            weekHydrationGuage);
        numOfDay = 0;
        sumOfHydrationGuage = 0;
        weekHydrationGuage = {};
        weekNum = weekNum + 1;
    }
})
```

    * (common) Step 12) Set the newHyditionGuageArr array to HTTP Response to send a response to the Client.

- Case 3) Statistical range is 'year'
    * If the range value in the HTTP Request is 'year', records are retrieved from the cat_hydration_statistics_${catId} tablewhere the date column contains the year value specified in the HTTP Request. The date and hydration_guage values are stored in the hydrationGuageArr array variable.
    * (common) A newHydrationGuageArr array is declared as a constant and initialized as an empty array.
    * A monthHydrationGuage object is declared as a let variable and initialized as an empty object.
    * (common) numOfDay, sumOfHydrationGuage, and avgOfHydrationGuage variables are declared as let variables and all initialized to 0.
    * A currentMonth variable is declared.
    * The hydrationGuageArr array is iterated over using the forEach method with the same logic as in Case 2.
    * (common) The newHydrationGuageArr array is sent in the HTTP Response to the client.

- Client-side HTTP Response Handling, Data Binding
    - Store the newHydrationGuageArr array from the HTTP Response.
    - Calculate the average of hydration_guage values and pass it as the avg property to the Avg Component.
    - Use the map method to loop through the newHydrationGuageArr array. Logics are following:
    - Case 1) Statistical range is 'week'
        * The 'x' property of the Bar Component is set with each element's date property value.
        * (common) The 'y' property is set with the 'hydration_guage' value.
    - Case 2) Statistical range is 'month'
        * The 'x' property of the Bar Component is set with each element's week property value.
        * (common) The 'y' property is set with the 'hydration_guage' value..
    - Case 3) Statistical range is 'year'
        * The 'x' property of the Bar Component is set with each element's month property value.
        * (common) The 'y' property is set with the 'hydration_guage' value.

11) ModifyCatFeedStuff API:

- Client → Server HTTP Request
    - Set the values of managementSpaceId and catId as HTTP Request parameters, where catId is obtained from the key attribute of the selected CatProfile Component in the LongAlert Component.
    - Additionally, set the values of catFeedStuffDailyConsumption and catFeedStuffMoistureContent in the HTTP Request before sending it to the Server.

- Sever-side HTTP Request Handling, Server → Client HTTP Response

- Retrieve spaceId and catId from the HTTP Request and store them in the respective variables.
- Modify the records in the cat_in_management_space_${spaceId} table for the given catId, updating the cat_feedstuff_daily_consumption and cat_feedstuff_moisture_content values.
- Case 1) Automatic Negative Amount Setting
  * If the is_hydration_auto value was true in the record, store the values of cat_weight, cat_feedstuff_daily_consumption, cat_feedstuff_moisture_content to catWeight, catFeedStuffDailyConsumption, and catFeedStuffMoistureContent respectively.
  * Calculate a new value for the newGoalHydration variable as (catWeight * 50) - (catFeedStuffDailyConsumption * catFeedStuffMoistureContent).
  * Retrieve the most recent record from the cat_hydration_statistics_${catId} table and update the goal_hydration with the value of newGoalHydration.+
- Set modifyDone to true.
- Include modifyDone in the HTTP Response to send and inform the Client of the successful modification.

- Client-side HTTP Response Handling
  - If the HTTP Response includes modifyDone, display an Alert Component with the message ”고양이 습식 사료 정보가 수정되었습니다.” and navigate to the previous screen.