HOLME

An Application for Migrating Smart Home Configuration Using Matter

Kang Museong
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
bbibbi4808@hanyang.ac.kr

Kwon Hyuktae
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
kwon0111@hanyang.ac.kr

Lim Kyumin
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
mycheesepasta@gmail.com

Ha Seongwu
College of Engineering
Hanyang University
Dept.of Information Systems
Seoul, Korea
rockey6865@hanyang.ac.kr

***Abstract*—Our team is strivingly working on developing a MATTER-based smart home configuration maintenance application called 'HOLME.' The aim of this application is to make it convenient to bring the existing smart home configuration to locations other than one's own home. Traditional smart home configuration have been constrained to specific applications under the umbrella of a single manufacturer. However, with the emergence of the MATTER protocol, it has become possible to integrate various IoT devices of different manufacturers seamlessly. Based on the MATTER protocol, we try to utilize the protocol to upload and download one's smart home settings to the server in any circumstances. Our core objectives revolve around the following two points: (1) Convenience: We aim to enable users to utilize our app to easily import their well-configured and time-consuming smart home settings to different places via QR code at once. (2) Replaceability: When users go to a new place which do not have device(s) in their homes, we aim to align the environment to the pre-configured environment. We provide notifications and reports after the replacement process. Through a B2B business model, our application allows users to conveniently migrate their smart home environments using QR codes, not only from one home to another but also to places like hotels. Through this project, we aspire to expand the scope of traditional smart homes and develop a service that will play a pivotal role in the emerging sharing economy, particularly in shared housing scenarios in the near future.**

***Index Terms*—HOLME, MATTER, Convenience, Replaceability, B2B, Sharing Economy**

TABLE I
ROLE ASSIGNMENTS

Roles	Name	Task description and etc.
User, Customer, Development manager	Kang Museong	User/Customer consider what features should be added from the perspective of users or customers. Development manager oversees the overall aspects of the project, such as project scheduling and planning, product and service quality. Additionally, they accurately grasp user requirements and manage and supervise the entire software engineering process, including software design, development, and testing.
AI Developer	Kwon Hyuktae	AI developers create programs that adapt to the business's needs based on collected and analyzed data. In this service, they develop AI that makes recommendations based on users' past data. They design, develop, implement, and monitor AI systems and focus on data collection and data transformation architecture.
Software Developer (Back-end)	Lim Kyumin	Backend developers consider the backend systems required for project development, utilizing databases and SQL queries. They manage the server-side and databases related to websites, web applications, or mobile solutions. As backend developers, they can work with various programming languages such as Python, Java, Node.js, and JavaScript.
Software Developer (Front-end)	Ha Seongwu	Front-end developers design the UI/UX of applications to enhance the user experience, considering user convenience while implementing application designs. To accomplish this, proficiency in React-Native, TypeScript, and CSS is necessary.

I. INTRODUCTION

A. Motivation

In 2003, when the global population was 6.3 billion, there were 500 million IoT devices. However, it is anticipated that by 2025, there will be a distribution of 1 trillion IoT devices among a global population of 8.1 billion. As the proliferation of IoT devices has rapidly progressed, ICT companies have individually developed their IoT platforms. However, an issue arose as each company created their own applications.

The existing smart home environment relied on devices from specific manufacturers and their proprietary applications. This resulted in users being tied to specific brands or apps when setting up their smart home environments. For example, a user with Samsung smart home devices who wanted to add

LG smart home devices had to reset their existing Samsung smart home environment and download the LG smart home app, which was quite inconvenient for users.

MATTER was created with the aim of unifying these 'fragmented' IoT platforms, making it possible to control IoT devices from various manufacturers using a single protocol.

With the introduction of the new MATTER protocol, it has become possible to effectively integrate various IoT devices. Therefore, we intend to develop an application that leverages cloud servers and generative AI to enable users to easily retrieve the smart home environment they configured at home from different locations.

Our team's goal is to provide a feature that allows users to set up their initial smart home environment just once. This environment should remain easily maintainable when they travel, move, or go on business trips to different locations. This application is not only applicable to individual users but also to B2B scenarios, such as hotels and other accommodation providers. Furthermore, it is anticipated that it will play a pivotal role in shared economy models, like shared housing, in the future.

B. Problem Statement

1 In the modern world, where the penetration rate of IoT (Internet of Things) devices is steadily increasing, the need for services that manage these devices and operate the environment efficiently is becoming more important.

2 As the percentage of international travel and business trips has increased since the end of COVID-19 and this has resulted in many people moving to other places, the need for services that can easily relocate and set up the environment of smart home devices is becoming more pronounced.

3 When moving environments from an existing smart home, compatibility issues arise due to differences between the manufacturer's starting smart device and the target smart device, which makes it difficult for users to move their current convenient smart home settings to a new location. In this situation, there is a need for services and capabilities to easily transfer and set the environment beyond the manufacturer's constraints between the starting and target smart devices.

4 When you move away from the existing place where your smart home is configured and move to another place, the new configuration feels cumbersome. This increases the need for solutions that make it easier to move around and to facilitate the smooth transfer of smart home devices.

5 There are no applications to date that take into account the possibility of automatic replacement for devices that are not in a new location when you move your smart home environment to another location. This is where users experience inconvenience in relocating their environment, and there is a growing demand for new solutions and capabilities.

6 Traditional voice notification services often tend to provide information in a way that lists hard and unnecessary content, making it difficult for users to effectively accept information and understand the situation. These limitations increase the need for improved user experience and more effective voice notification services.

C. Target Customer

1 Smart Home Owners
The main target audience for this project is individual smart home owners. They are people who want to effectively manage their smart home environment and migrate to other places, including smart home owners who are interested in moving their residences or bringing them up from other places.

2 Hotels and properties
HOLME can also be delivered to hotels and properties through the B2B business model. This may include tourists and business travelers who want to experience a home-like smart environment at a hotel or property.

3 IoT device maker
HOLME opens up the possibility of working with IoT device maker's products by providing solutions that integrate and maintain compatibility with various IoT devices. Also, companies looking for new business opportunities are potential target customers for HOLME.

4 Sharing Economy Participants
Sharing economy participants will be very interested in solutions that can easily share smart home environments and move to other places. These solutions will provide shared and rental home owners and users with the opportunity to move freely and enjoy similar smart environments in new places while sharing the convenience of smart

D. Research on Related Software

1 ThinQ
ThinQ is LG Electronics' brand for smart devices and appliances, providing users with a more convenient smart home experience through features like smart control, AI integration, voice commands, home automation, and smart routines. This technology and brand are utilized in various LG products. Additionally, users can invite others to their registered spaces using QR codes and have the capability to register MATTER-supported devices.

2 SmartThings
SmartThings is a smart home automation and control

platform developed by Samsung Electronics, allowing users to centrally control and connect smart devices and appliances. This platform offers features such as convenient remote control via a smartphone app, automation and routine settings, compatibility with various devices, interconnectivity, and integration with third-party systems, providing users with an integrated smart home experience.

3 NUGU Smart Home
NUGU Smart Home is SK Telecom's smart home platform that allows users to control household appliances and IoT devices through voice commands. Additionally, this platform provides apartment management services, including apartment news updates, filing complaints, access to shared entrances, and parking information services, among others.

4 GIGA Genie Home IoT
Giga Genie Smart Home is KT's smart home platform, enabling control of household appliances and IoT devices through voice commands. This platform collaborates with various companies to manage not only appliances like refrigerators but also devices such as boilers and cars.

5 Google Cloud IoT Core
Developed by Google, this platform provides a fully managed service and allows easy and secure connection, management, and data ingestion from globally dispersed devices.

6 AWS IoT Device Management
Provided by Amazon Web Services (AWS), this platform aims to facilitate the secure and efficient management of Internet of Things (IoT) devices. It offers tools and features to simplify the onboarding, organization, monitoring, and updating of IoT devices at scale.

7 Azure IoT Hub
Azure IoT Hub is a versatile and scalable cloud platform (IoT PaaS) that caters to multiple tenants. It comprises an IoT device registry, data storage, and robust security features. It also offers a service interface to facilitate IoT application development.

8 Nabu Casa
Nabu Casa is the company behind Home Assistant, a smart home automation platform that integrates and manages smart home devices and services. They offer cloud services for remote management and expansion of smart homes and provide a subscription-based service for storing IoT device settings and routines in the cloud, enabling remote management.

9 Hubitat
Hubitat is a home automation hub that supports Z-Wave and Zigbee protocols, providing local control and processing, customisability, multiple smart home device compatibility, and cost-effective features. In addition, Hubitat is a hub service that manages the smart home hub itself, allowing seamless routine and rule-setting between devices from different manufacturers.

E. Expectation Effectiveness

1 Expanding the range of smart homes
In addition to breaking away from traditional smart home environments (single manufacturer, tied to specific applications), HOLME extends the reach of smart home technology by expanding existing smart home environments to other places, not just at home. This allows users to easily bring up and manage their smart home environment outside of home.

2 Technological innovation and Competitive advantage
Based on the MATTER protocol, HOLME integrates various IoT devices into smart home applications and combines generative AI with the cloud. This provides an opportunity to lead technological innovation and gain a competitive edge in the existing smart home market.

3 Forward-looking
HOLME will be able to lead smart home technology to the wider market through cooperation with various accommodations such as hotels. In addition, it will play a key role in the shared housing platform in the upcoming trend of the shared economy.

F. Key Definitions

1 Virtual space
1) Basic Assumption: Save the configuration settings and routines I'm currently using to 'virtual space.'

2) If IoT devices set at home are set in a 'virtual space' called "My House", the setting of a 'virtual space' called "My House" can be imported into another space.

2 Logical hub
Backbone SW that gives arbitrary IoT devices the ability to act as a matter hub.

3 synchronization
It refers to the process of calling the settings of 'virtual space' by scanning a QR code.

4 allocation
The function of connecting the IoT devices present in each room to their respective rooms within the hotel.

G. Scenario

1) A hotel entered into an agreement with HOLME and 'allocated' the existing IoT devices to their respective rooms.

2) User (Customer) uses the HOLME app at home and villa to store and use smart home environments and routines in the form of each 'virtual space.'

3) The user has a business trip coming up, so he browsed a hotel reservation app and made a reservation at a hotel with the HOLME mark.

4) When the user arrived at the hotel room, he scanned the QR code in the room, thereby 'synchronization' that room with the user's 'virtual space.' During the 'sync' process, the user can choose which 'virtual space' to 'synchronization' with.

5) Then, a new 'virtual space' that can manage the hotel room was copied into the HOLME app, and the existing settings were applied.

H. Profit Structure

1 Certification and Hotel Partnership
HOLME can partner with the hotel to provide certification for the hotel's smart home environment. This certification is responsible for ensuring the quality and stability of the hotel's smart home service to the customer. The hotel may pay a fee to obtain and maintain this certification.

2 Marketing and Public relations agreements
Once the hotel is certified by HOLME, it can be used for marketing and promotion purposes.HOLME promotes the hotel in its own application, and the hotel can promote HOLME to mutual benefit.

3 Custom Solutions and Consulting
Providing customized smart home solutions for properties such as hotels, and providing consulting and integrated services for this purpose can generate revenue.

II. REQUIREMENT ANALYSIS

A. Common Features

1 Tutorial
The tutorial should be the first screens where HOLME is downloaded and shown. Users should be able to select the following options.
1) Skip the tutorial and sign up directly
2) View next page. If a user has reached the last page of the tutorial, the next step should be sign-up

2 Sign-Up
HOLME needs four types of information to sign up for membership. These are phone numbers, passwords, name, and birth dates.
1) Enter phone numbers
The phone number must be entered, and the phone number is verified through the carrier's authentication system to confirm whether the phone number is valid for membership registration. The phone number serves as an ID in the subsequent login process.
2) Enter passwords
Passwords must be entered and must be at least 8 characters long in combination of 3 or more of English uppercase/English lowercase/number/special characters. When the user enters the desired password, it is displayed in the form of '****' on the screen, expressing information about it as [Unavailable/Safe/Dangerous].
3) Enter a name
The name must be entered, and subsequently set to the default nickname at first login. The name is also used in ID search.
4) Enter birth dates
The birth dates must be entered, and a pop-up window is displayed every year to celebrate the birthday of the user. The date of birth is also used in ID search.

3 Log in
There are two types of logins: 1) Local logins through HOLME membership, 2) SNS logins through SNS linkage.
1) Local logins through HOLME membership
(1) Local logins through HOLME membership The system checks whether the ID and password entered by the user have filled the digits. If the number of digits is not filled, a warning message is shown in red.
(2) When the ID and password input by the user exist in the member database, the user succeeds in logging in. After that, it moves to the main page.
(3) If the phone number and password entered by the user do not exist in the member database, the user fails to log in and displays "Non-existent member" in the pop-up window.
2) SNS logins through SNS linkage
(1) Utilizes the Google, Apple, Facebook, Amazon, Naver, Kakao sign-up APIs.
(2) If the SNS login link is successful, the system must receive the user's name and date of birth. Then, go to the main page.

4 Find ID
It is a function that exists for people who have lost their ID. HOLME's ID is based on the phone number, but you can add an email. Therefore, when you forget your phone number, you find your phone number using e-mail, and when you forget your e-mail, you find an e-mail based on the phone number. In case you don't remember both, you can find your ID by the name and date of birth registered at the time of membership registration.
1) The system receives an e-mail or telephone number. If the corresponding information exists in the user DB, a phone number or email is notified based on the corresponding information.
2) If the user does not know either e-mail or phone number, the system receives the name and date of birth and teaches the ID if the information exists in the user DB

5 Resetting password
1) The system receives an ID to reset the password.
2) If the input ID does not exist in the user DB, a warning message will be displayed saying, "Please enter your email or phone number correctly."
3) When the input ID exists in the user DB, the system receives the user's name, date of birth, and phone number, and goes through the process of verifying whether the user is correct through authentication by the carrier.
4) When the user succeeds in identifying himself/herself, the system receives a password so that the user can reset the password. At this time, the password must have a length of at least 8 characters in a combination of at least three of the English uppercase/English lowercase/number/special characters. When the user enters the desired password, it is displayed in the form of '****' on the screen, expressing information about it as [unavailable/safe/dangerous].

6 Language change
This is a function that should be presented in the upper right corner of the login window, and the initial default value is Korean. You should be able to change this into another language.

B. User-Specific

1 Main page
The main page is responsible for 'Virtual space'. The main page consists of Virtual space management, Virtual space setting, device addition, device operation, and routine execution.
1) Virtual space management
The user should be able to add Virtual space, name, modify, and delete each Virtual space. In addition, the color should be set for each place so that the top color of the main page can be changed together.
2) Virtual space settings
Among the virtual spaces added by the user, the desired Virtual space is set as the 'current place' and set as the main page. In addition, it should be possible to show a list of Virtual spaces created by users, and to change the 'current place' into another Virtual space.
3) Device connection
Users should be able to call up all devices at once through QR code recognition.
4) Device manipulation
Users should be able to manipulate IoT devices located on the main page.
5) Run Routine
The user should be able to execute routines located on the main page.

2 Menu bar
The menu bar is responsible for 'my settings'. The menu bar consists of the following five types.
1) Device Settings
Detailed settings can be stored in advance for each type of IoT device. Here, the devices are virtual devices, and detailed settings may be set even for devices that do not have them in reality. After connecting to the space, the actual devices are covered with pre-set settings and the device setting is performed with a connection.
2) Routine Settings
(1) Search Routine
The user should be able to search for pre-set routines.
(2) Add Routine
Users should be able to generate the desired routine by adding routines. Here, the routine means the operation of several actions. The user should be able to set the routine name and then, when this routine starts, set which actions should be executed. And when adding each action, it should be possible to add the desired action of the desired device through 'device and action search'.
(2) Edit Routine
The user should be able to edit the routine. This refers to changing the name of the routine, adding actions, deleting actions, and changing the order of actions.
3) Home
The user should be able to go to the main screen of the space where he is currently located by pressing the home button.
4) Report
Reports are generated when the user connects preset device settings and routine settings to a location.

This is a specific report of what settings have been applied to automatically connected devices, which are replaceable and which are irreplaceable. It also generates a report when the routine is executed. This should inform the user that a device has performed a certain action in the course of executing the routine execution of the routine.

5) My HOLME
My HOLME plays the role of 'set-up' in other applications These include profile changes, nickname changes, notification settings, network connections, IoT service connections, application for accommodation manager, language changes, one-to-one inquiries, email ID changes, and password changes.

3 Import settings
In conjunction with other IoT management applications, HOLME should be able to load a list or routine of devices previously used by other applications to HOLME.

4 Considering Replaceability
When a user loads their smart home environment, considering the likelihood that the configurations of all IoT devices may not be identical, interchangeability is taken into account.
(1) When function replication is possible: automatic connection and function execution.
(2) When function replication is not possible: submit a report.

C. Hotel-Specific

1 Log in for Hotel Administrator
Hotel administrator register as members with ordinary members, but if they apply for hotel administrator authority through "application for accommodation manager", and certify it at HOLME, a new hotel management menu will be opened.

2 Menu bar for Hotel Administrator
The menu bar for hotel managers consists of the following. Room management, device management by room, QR code management, inquiry

1) Room management
The user should be able to add plave, name, modify, and delete each place. In addition, the color should be set for each place so that the top color of the main page can be changed together.

2) Device management by room
Hotel managers should be able to add, modify, and delete IoT devices that will be placed in rooms created through 'room management'. This can be applied collectively to multiple rooms.

3) QR code management
For rooms where device management for each room has been completed, a QR code that can bring all devices connected to the room to HOLME at once should be generated. In addition, the administrator must be able to expire the QR code if desired.

4) Inquiry
Hotels should be able to proceed remotely by contacting HOLME for all processes, including room management, room-specific device management, and QR code management. It also serves as a consultation channel for hotel managers and HOLME

III. DEVELOPMENT ENVIRONMENT

A. Choice of Software Development Platform

1 Development Platform

1) Windows
Windows provides a wide range of development tools and integrated development environments (IDEs) for creating various types of applications, including web applications, desktop applications, mobile apps, and games. This supports effective code editing, debugging, testing, deployment, and collaboration, ultimately enhancing developers' productivity. Furthermore, Windows supports multiple programming languages and frameworks, allowing developers to choose their preferred languages and technologies to flexibly meet project requirements. Windows offers a user-friendly and intuitive interface, making it easy for developers to configure and manage their development environments. A robust community and support system enable developers to share experiences and receive assistance. Lastly, Windows continuously updates and improves, ensuring access to the latest technologies and tools, empowering developers to stay current and modernize their applications. Windows is recognized as a versatile platform suitable for various software development fields, playing a crucial role in turning developers' ideas into reality.

2) macOS
macOS is a highly regarded operating system in the field of software development, known for its user-friendly interface and exceptional versatility. This operating system offers several advantages to developers, and let's explore some of them. Firstly, macOS provides essential development tools and an integrated development environment (IDE) for creating a wide range of applications, including web applications, desktop applications, mobile apps, and games. Official IDEs like Xcode are powerful tools for application development across various platforms such as macOS, iOS, watchOS,

and tvOS. They support tasks like code writing, debugging, testing, deployment, and collaboration, significantly enhancing developer productivity. Additionally, macOS supports a variety of programming languages and frameworks, allowing developers to choose their preferred languages and technologies, making it flexible to adapt to project requirements. macOS offers an intuitive and user-friendly interface that simplifies development environment setup and project management. The active macOS developer community provides a platform for sharing experiences and collaboration among developers. Finally, macOS ensures access to the latest technologies and tools through continuous updates and improvements. Apple's dedication to innovation provides developers with the necessary features to leverage the latest technologies and modernize their applications. For these reasons, macOS is recognized as an essential platform for software development, playing a significant role in turning ideas into reality.

2 Language / Framework

1) Programmin Languages

(1) Typescript [1]

Fig. 1. TypeScript

TypeScript is a powerful tool developed by Microsoft, which is a superset of JavaScript. It provides static type checking, enhancing the development of robust and scalable applications. Introduced in 2012, TypeScript allows developers to detect errors at compile time, resulting in fewer bugs and improved code quality. Additionally, TypeScript offers advanced features such as interfaces, generics, and decorators, strengthening code organization and maintenance. It maintains full compatibility with JavaScript, enabling a seamless migration of existing JavaScript code to TypeScript, providing web developers with various options. TypeScript's advantages have led many enterprises to adopt it over JavaScript, regardless of project size, enhancing development productivity and code reliability. In summary, TypeScript is a valuable tool for modern web development, providing developers with better code quality and efficiency while simplifying project management.

(2) Kotlin [2]

Fig. 2. Kotlin

Kotlin is a modern programming language developed by JetBrains and widely used as an alternative to Java. Kotlin provides developers with concise syntax and a stable type system, making it easier to write and maintain code efficiently. Moreover, it is extensively employed in Android app development and offers seamless interoperability with existing Java code. The succinct syntax enhances project productivity and fosters collaboration by simplifying code writing and comprehension. The robust type system detects errors at compile time, boosting code reliability and reducing runtime errors. In the realm of Android app development, Kotlin enables more efficient application development and improved user experiences. Additionally, Kotlin seamlessly integrates with existing Java code, facilitating the modernization of legacy projects. In summary, Kotlin, as a contemporary programming language, offers numerous advantages, providing developers with improved code quality and efficiency while simplifying project management.

(3) GO lang [3]

Fig. 3. GO lang

Go language, developed by Google, is a programming language that offers a combination

of simplicity and powerful features. It's a compiled language known for its fast execution speed and efficient memory management. Go's concise syntax makes it easy to write and maintain code, and it comes with a rich standard library that supports various tasks. Furthermore, Go language emphasizes concurrency and supports parallel programming through lightweight threads known as goroutines. It provides a module system for simplified dependency management, enhancing project management and collaboration. Go is utilized in a wide range of fields, from server development to cloud computing, mobile apps, games, data analysis, and artificial intelligence. Finally, Go language's fast compilation speed and small executable file sizes enable efficient development and deployment. As a result, many developers and companies choose Go to develop efficient and stable software.

2) Frameworks

(1) React Native [4]

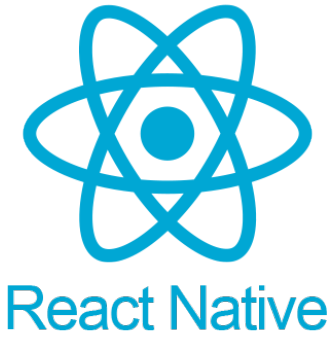

Fig. 4. React Native

React Native is a framework that utilizes JavaScript and the React library to develop mobile apps for both iOS and Android platforms. It offers the advantage of cross-platform app development while delivering performance and user experiences similar to native apps. React Native employs a component-based structure to build apps with modular components and supports hot reloading for quickly verifying code changes. Additionally, it integrates native modules for accessing hardware features and interacting with external services. Furthermore, it benefits from an active community and a wealth of open-source packages, providing extensive support and resources to developers. React Native stands as a powerful tool for efficiently developing mobile apps.

(2) Spring Boot [5]

Fig. 5. Spring boot

Spring Boot is a framework for easily developing Java-based web applications and microservices. This framework offers convenient configuration, an embedded web server, automatic setup, starter dependencies, monitoring and management capabilities, support for microservices, and access to a rich ecosystem of libraries and tools. Using Spring Boot, developers can rapidly build applications, reduce the complexity of configuration, and enhance productivity.

(3) Hibernate [6]

Hibernate is an open-source Object-Relational

Fig. 6. HIBERNATE

Mapping (ORM) framework for Java. It enables seamless interaction between Java objects and relational databases. Key highlights of Hibernate include database agnosticism, automatic table generation, an Object-Oriented Query Language (HQL), caching, built-in transaction management, and a strong community and ecosystem. In essence, Hibernate simplifies database operations in Java applications, offering flexibility, performance, and portability.

(4) gRPC [7]

Fig. 7. gRPC

gRPC is a high-performance Remote Procedure Call (RPC) framework developed by Google,

designed to facilitate communication between services in various environments. gRPC uses Protocol Buffers for data exchange, offering an efficient binary format that allows message definitions to be shared across multiple programming languages. The framework supports multiple programming languages, making it possible for clients and servers written in different languages to communicate seamlessly. Operating based on the HTTP/2 protocol, gRPC provides efficient and fast communication, featuring features such as multiplexing, header compression, bidirectional communication, and more. Additionally, gRPC includes automatic code generation, simplifying developer tasks and ensuring type safety. Widely used in cloud and microservices architectures, gRPC supports efficient service-to-service communication.

(5) Flask [8]

Fig. 8. Flask

Flask is a lightweight and extensible Python web framework used for developing web applications. It provides developers with a concise yet flexible structure for efficient work. With features such as URL routing, template engine, and session management, Flask offers scalability to add necessary functionalities flexibly. Known for its simple syntax and high flexibility, Flask is favored by many developers and can be employed in a variety of projects, ranging from small-scale initiatives to large-scale web applications.

3 Cost Estimation

To implement HOLME, it was necessary to obtain data from the database or obtain real-time information from the server while communicating with the server in real-time. Therefore, real-time server hosting or multiple APIs were required. However, during the development process, we initially made efforts to utilize open APIs, free modules, and free servers.

4 Development Environment

Name	Computer Resource	Version of OS, SW
Kang Museong	Intel Core i5 16GB RAM memory	Windows 10 TexLive 2022
Kwon Hyuktae	Apple M1 Chip 16GB RAM memory	MacOS Ventura 13.5 Visual Studio Code 1.82.0 IntelliJ (LTS) spring boot (3.1.1)
Lim Kyumin	Apple M2 Chip 16GB RAM memory	MacOS Ventura 13.4 GoLand (LTS) WebStorm (LTS) IntelliJ (LTS) React-Native (10.1.3) spring boot (3.1.1)
Ha Seongwu	Intel Core i5 8GB RAM memory	Windows 11 Home Visual Studio Code 1.82.0 React-Native (10.1.3) Android studio (LTS) Visual Studio Code 1.82.0

5 Cloud Platform

We plan to use GCP (Google Cloud Platform) instead of AWS (Amazon Web Services) for a specific reason. We believe that GCP is more suitable for our needs because we are looking for instance types that are lightweight for hosting our servers, and in such cases, GCP is a better fit compared to AWS.

B. *Software in use*

1 visual Studio Code

Fig. 9. Visual Studio Code

Visual Studio Code (VS Code) is a highly popular integrated development environment (IDE) among developers. This convenient code editor is available for free and is known for its speed and lightweight nature, making it a preferred choice among users. VS Code supports various programming languages and offers excellent extensibility, allowing users to add the necessary features through extensions. Additionally, it provides intelligent code completion, debugging, Git integration, and a range of development tools to simplify coding tasks. With a user-friendly and intuitive interface, it offers an environment for programmers to work efficiently. For these reasons, VS Code stands as one of the most favored development tools among developers.

2 IntelliJ

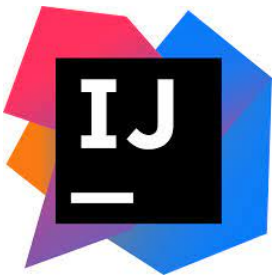

Fig. 10. Visual Studio Code

IntelliJ IDEA is a renowned integrated development environment (IDE) designed for Java developers. It is developed by JetBrains and is known for its robust features and user-friendly interface. IntelliJ IDEA offers a wide range of tools and functionalities to enhance Java application development. With intelligent code completion, comprehensive coding assistance, and advanced refactorings, developers can write high-quality code more efficiently. The IDE also supports a variety of programming languages and frameworks, making it a versatile choice for different projects. Furthermore, it provides excellent integration with popular version control systems and build tools, streamlining the development process. Overall, IntelliJ IDEA is a powerful and versatile IDE that caters to the needs of Java developers and beyond, making it a top choice in the development community.

3 Android Studio

Fig. 11. Android Studio

Android Studio is an integrated development environment (IDE) developed by Google for Android application development. This IDE serves as a core tool for developing Android apps, providing a user-friendly interface and a rich ecosystem of plugins to make Android app development more accessible. Integrated with the Android SDK tools, Android Studio allows quick access to the latest Android APIs and features. It also offers an emulator for simulating and testing apps on various Android devices, along with robust debugging and performance profiling tools to support the development process. Android Studio provides tools and resources for app deployment and assists developers in building Android apps and publishing them on app stores like Google Play Store.

4 WebStorm

Fig. 12. WebStorm

WebStorm is a popular integrated development environment (IDE) designed specifically for web development. Developed by JetBrains, it offers a comprehensive set of tools for building modern web applications using web technologies such as HTML, CSS, and JavaScript. WebStorm provides a rich and intuitive coding environment with features like code completion, navigation, and refactoring, making web development more efficient and productive. It also offers built-in support for popular web frameworks and libraries, real-time code analysis, and debugging capabilities to help developers create high-quality web applications. With its extensive set of features and continuous updates, WebStorm is a go-to choice for web developers looking to streamline their workflow and build web applications with ease.

5 GoLand

Fig. 13. GoLand

GoLand is an integrated development environment (IDE) developed by JetBrains, designed specifically for the Go programming language. It offers powerful tools for developers and programmers working with Go, enhancing their productivity and facilitating efficient code development. This IDE provides various features and tools tailored to the Go language's specific characteristics. It includes robust code completion, refactoring, debugging, testing, and module support,

making code writing easier and more efficient. Additionally, GoLand integrates project management and version control tools, simplifying complex tasks for developers. GoLand also offers features such as static analysis, code inspections, and auto-completion to enhance code quality, supporting safe and efficient Go language development. It serves as a comprehensive tool for all developers and teams working with the Go language, aiding in managing and developing Go language projects effectively.

6 PostgreSQL

Fig. 14. PostgreSQL

PostgreSQL, often known as Postgres, is a versatile and open-source relational database management system. Its adaptability stands out, allowing developers to customize data types and functions to meet specific project needs. PostgreSQL excels in data integrity and supports advanced concurrency control, ensuring data consistency in multi-user scenarios. With an active and supportive community, it receives regular updates and improvements, making it a reliable and high-performance choice for businesses and developers seeking an open-source RDBMS.

7 LaTeX

Fig. 15. LaTeX

LaTeX is a free typesetting system designed for creating professional documents, spanning various academic fields such as science, mathematics, and technology. It utilizes text files with commands to define the structure, formatting, tables, graphics, equations, references, and more in a document. Using these commands, you can compose your document, and a compiler is used to produce an output in the form of a PDF or other document formats. In contrast to word processors, LaTeX offers professionalism and consistency in formatting, making it ideal for creating documents like research papers, academic theses, books, presentations, and more. LaTeX reduces the need to worry about layout, fonts, and paragraph divisions. It excels in typesetting mathematical equations, allowing you to beautifully represent complex mathematical notations. Furthermore, LaTeX is well-known for its strong community support and various packages and styles available, enabling users to customize documents to meet specific requirements. Due to these features, it is widely used among researchers, students, writers, and engineers, facilitating the creation of professional, high-quality documents.

8 GitHub

Fig. 16. GitHub

GitHub is a web-based platform and service for version control and collaboration. It is widely used by developers and teams to manage and track changes in their code, making it an essential tool for software development. GitHub provides a centralized platform where developers can store, manage, and collaborate on their source code, as well as track any modifications or issues related to their projects. One of GitHub's core features is Git, a distributed version control system. Git enables developers to track changes, work on different aspects of a project simultaneously, and merge their work efficiently. GitHub adds a collaborative layer on top of Git, allowing multiple team members to work on a project collaboratively, making it easier to handle pull requests and code reviews. GitHub hosts millions of public repositories, making it a valuable resource for open-source projects. It offers features like issues tracking, project management boards, and wikis, helping teams streamline their development processes. Moreover, GitHub Actions allows for automated workflows, further enhancing productivity.

9 Notion

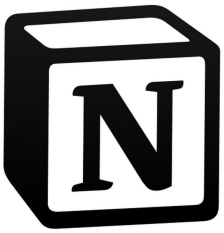

Fig. 17. Notion

Notion is an all-in-one productivity platform for collaboration, note-taking, project management, and knowledge-based tasks. It caters to both personal note-taking and collaborative work, allowing users to scale up for larger projects and team-based tasks. Notion employs a block-based approach to organizing information, providing users with creative flexibility to structure content using various block types like text, images, videos, checklists, tables, calendars, and databases. This versatility extends its utility beyond simple text documents, making it suitable for diverse tasks such as project planning, task tracking, and knowledge base creation. Notion excels in supporting team collaboration with features like real-time editing, comments, to-do lists, calendar management, and other collaboration tools. Its diverse features and high levels of customization make it widely adopted as an effective productivity tool for personal and business use.

10 Zoom

Fig. 18. Zoom

Zoom is an online video conferencing and collaboration software used for remote communication. This platform offers features like video meetings, webinars, screen sharing, group chat, and file sharing, facilitating efficient work and communication. Widely used to support remote work and education internationally.

IV. REQUIREMENT SPECIFICATION

A. Entry

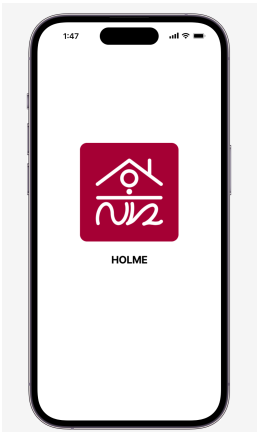

Fig. 19. ID:001, HOLME-Entry-splashing

ID	Name	Description
001	HOLME-Entry-splashing	When the application is launched, the startup page should be displayed for a duration of 1 to 2 seconds to prevent an empty page from being shown while the application is loading its data. This ensures a smooth and visually appealing user experience during the app's startup process.

B. Entry

Fig. 20. ID:002, HOLME-Tutorial

ID	Name	Description
002	HOLME-Tutorial	Upon the initial download of the application, a tutorial screen must be displayed to the user. The tutorial screen should consist of approximately 4 pages, providing information that explains the key features and usage of the application.

Fig. 21. ID:003, HOLME-Tutorial-Skip

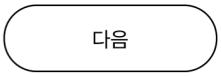

Fig. 22. ID:004, HOLME-Tutorial-Navigate

ID	Name	Description
003	HOLME-Tutorial-Skip	Users should have the option to skip the tutorial. If the user chooses to skip the tutorial, they must be able to proceed directly to the registration process.
004	HOLME-Tutorial-Navigate	Within the tutorial screen, users should be provided with an option to navigate to the next page of the tutorial. When users reach the last page of the tutorial, they should be presented with the option to proceed to the registration steps.

C. Sign-Up and Log-In

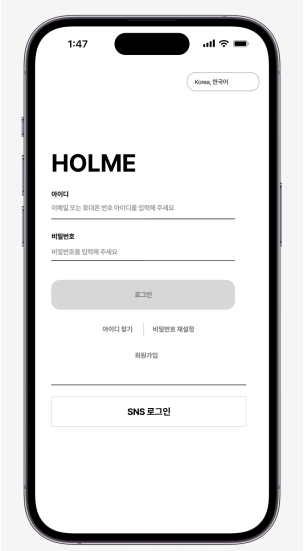

Fig. 23. ID:005, HOLME-Login-Page

ID	Name	Description
005	HOLME-Login-Page	Users should be able to use the following features in the login page: Sign Up, Log In, Find ID, Reset Password, SNS Login, and Language Change.

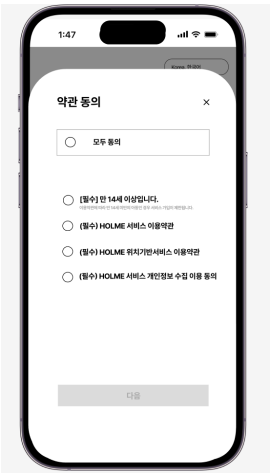

Fig. 24. ID:006, HOLME-SignUp-Terms and Conditions Agreement

ID	Name	Description
006	HOLME-SignUp-Terms and Conditions Agreement	Users must agree to HOLME's terms and conditions to sign up.

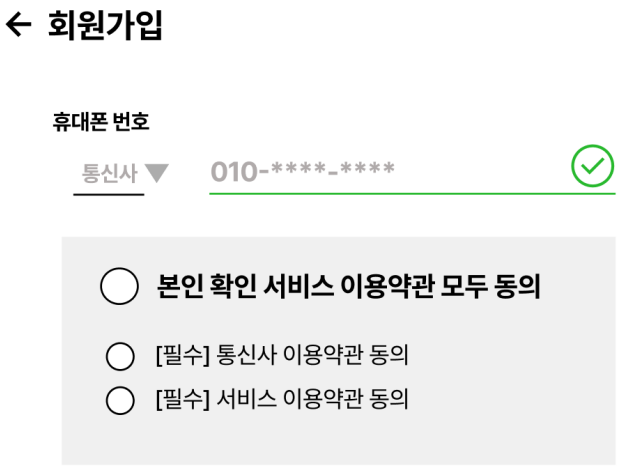

Fig. 25. ID:007, HOLME-SignUp-PhoneNumber

ID	Name	Description
007	HOLME-SignUp-Phone Number	Users are required to enter their phone number. The phone number will serve as the user's ID during the login process after registration. The validity of the phone number should be verified through the authentication system of the mobile service provider.

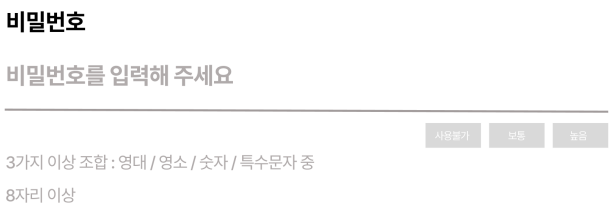

Fig. 26. ID:008, HOLME-SignUp-Password

ID	Name	Description
008	HOLME-SignUp-Password	Users must enter a password. The password should be at least 8 characters long and must contain a combination of at least 3 of the following: uppercase letters, lowercase letters, numbers, and special characters. When the user enters their desired password, it should be displayed on the screen as '****', and its security level should be indicated as "[Unavailable/Safe/Dangerous]" depending on the password's strength.

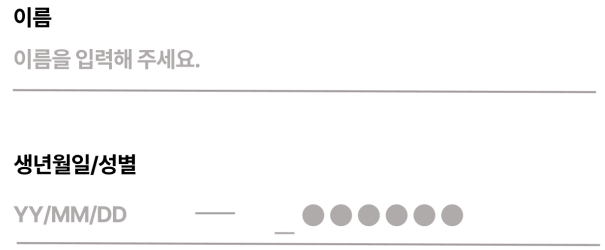

Fig. 27. ID:009, HOLME-SignUp-Name and Birthdate

ID	Name	Description
009	HOLME-SignUp-Name and Birthdate	Users must enter their name and date of birth. This information will be used for 'ID retrieval' purposes. The date of birth should be entered in the 'YY/MM/DD' format, and gender will be verified based on the first digit of the resident registration number. This information is utilized for the 'Find ID' functionality.

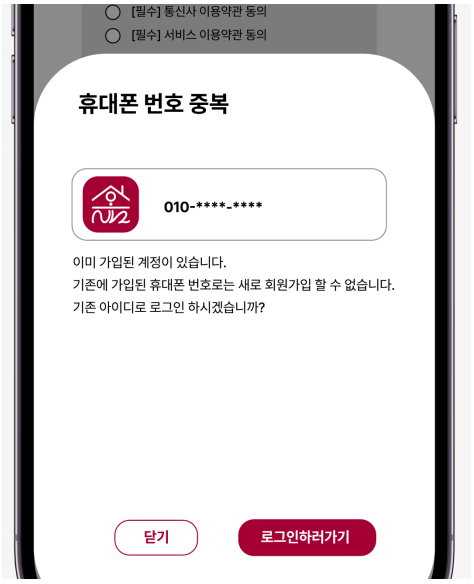

Fig. 28. ID:010, HOLME-SignUp-PreventingDuplicate-Phonenumber

ID	Name	Description
010	HOLME-SignUp-Preventing Duplacte Phonenumber	Registration with duplicate phone numbers must be prevented. Attempting to register with a phone number that is already in use should not be allowed.

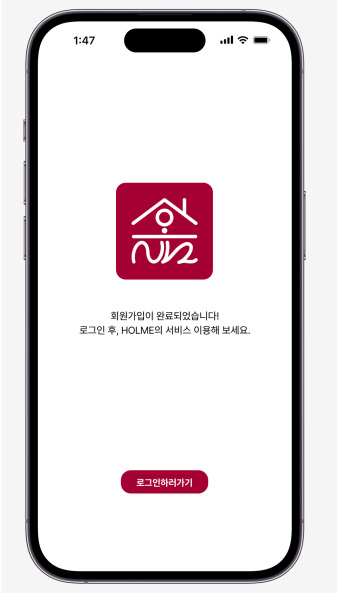

Fig. 29. ID:011, HOLME-SignUp-RegistrationCompleted

ID	Name	Description
011	HOLME-SignUp-Registration Completed	Upon successful registration, a notification should be displayed to the user, and they should be automatically redirected to the login process.

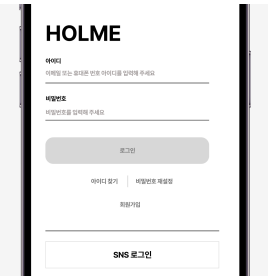

Fig. 30. ID:012, HOLME-Login-Types

ID	Name	Description
012	HOLME-Login-Types	Users should be able to log in using two types of login methods: (1) Local login via HOLME membership, (2) SNS login via social media integration.

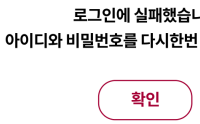

Fig. 31. ID:014, HOLME-Login-Local Failed(1)

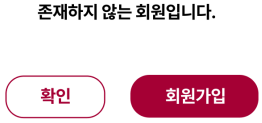

Fig. 32. ID:015, HOLME-Login-Local Failed(2)

ID	Name	Description
013	HOLME-Login-Local Success	If the ID and password entered by the user exist in the member database, the user will successfully log in and be directed to the main page.
014	HOLME-Login-Local Failed(1)	If the user enters their ID and password, but either the ID or the password is incorrect, a popup window will request the user to check their ID and password again.
015	HOLME-Login-Local Failed(2)	If the phone number and password entered by the user do not exist in the member database, the user will fail to log in, and a popup window will display the message 'Non-existent Member'.

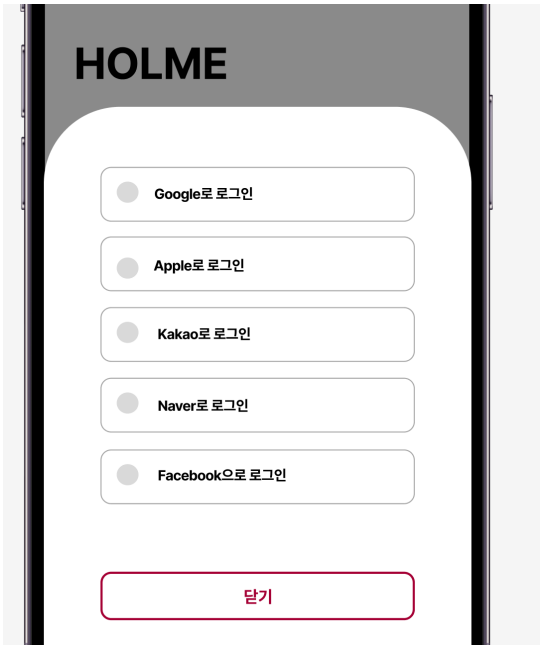

Fig. 33. ID:016, HOLME-Login-Types

ID	Name	Description
016	HOLME-Login-SNS login	The system utilizes SNS registration APIs such as Google, Apple, Facebook, Amazon, Naver, Kakao, and more. Users should be able to conveniently log in through these platforms.

Fig. 34. ID:017, HOLME-Login-FindID

ID	Name	Description
017	HOLME-Login-FindID	If a user forgets their username (phone number or email), they should be able to navigate to the "Find ID" page via the "Find ID" button.

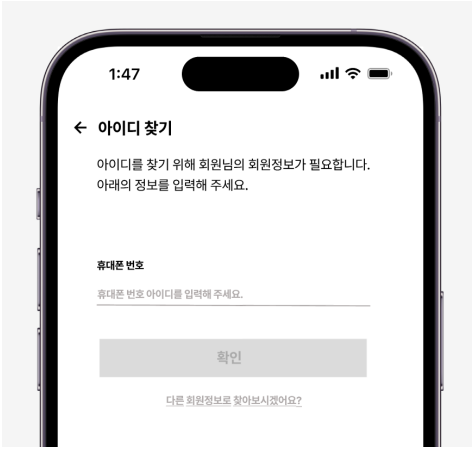

Fig. 35. ID:018, HOLME-Login-FindID-Phonenumber

ID	Name	Description
018	HOLME-Login-FindID-Phonenumber	Users should be able to find their userID by entering their registered phone number.

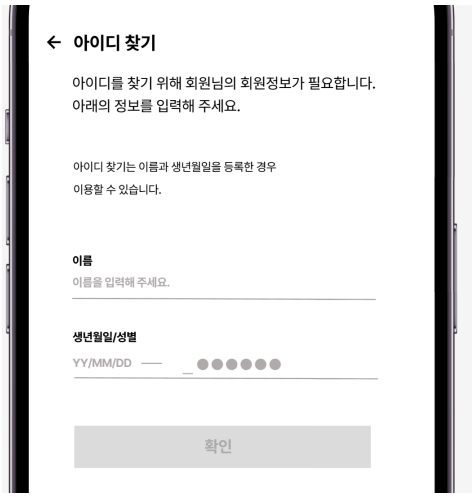

Fig. 36. ID:019, HOLME-Login-FindID-name/birthdate

ID	Name	Description
019	HOLME-Login-FindID-name/birthdate	If a user has forgotten their phone number as well, they should be able to find their username using other member information such as name, date of birth, and gender.

Fig. 37. ID:020, HOLME-Login-Password Reset

ID	Name	Description
020	HOLME-Login-Password Reset	If a user has forgotten their password, they should be able to access the “Password Reset” page via the “Password Reset” button.

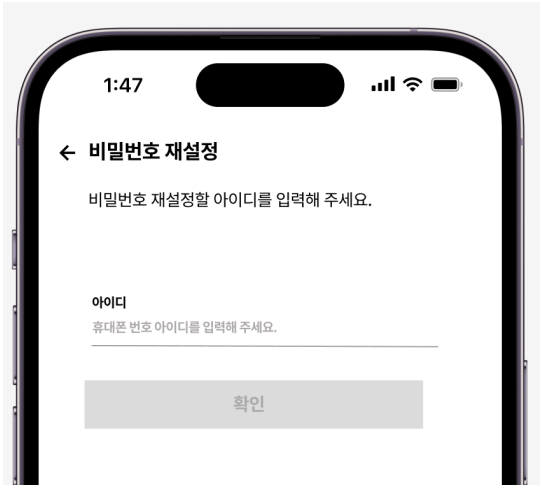

Fig. 38. ID:021, HOLME-Login-Password Reset Page

ID	Name	Description
021	HOLME-Login-Password Reset Page	Users should enter their registered phone number. The entered phone number should be verified to match the information in the user database.

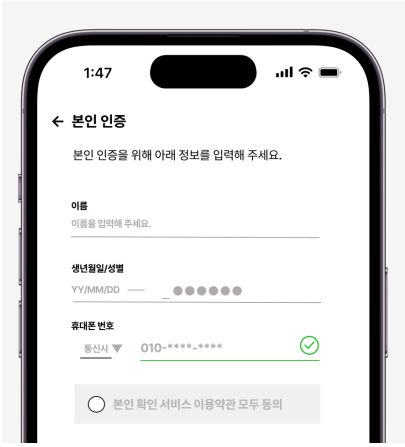

Fig. 39. ID:022, HOLME-Login-Password Reset-Authentication

ID	Name	Description
022	HOLME-Login-Password Reset-Authentication	The system should offer users guidance on setting a new password through carrier-based verification. Upon successful carrier authentication, users will be directed to the 'Password Reset' page; in case of failure, they will return to this page.

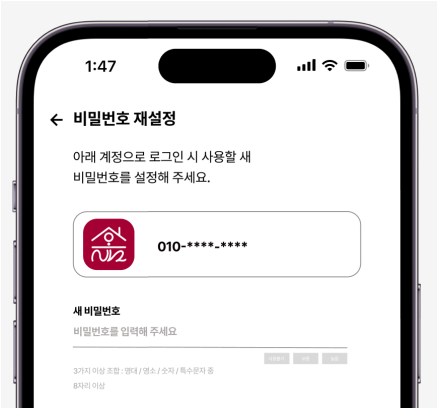

Fig. 40. ID:023, HOLME-Login-New Password

ID	Name	Description
023	HOLME-Login-New Password	When setting a new password, users should be provided with appropriate security requirements, such as a combination of at least 8 characters, including uppercase letters, lowercase letters, numbers, and special characters.

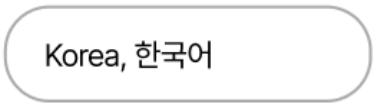

Fig. 41. ID:024, HOLME-Login-Language Setting

ID	Name	Description
024	HOLME-Login-Language Setting	The language setting feature should be located in the top right corner of the login window. The initial default language should be set to Korean. Users should be able to change to a different language by clicking on this button.

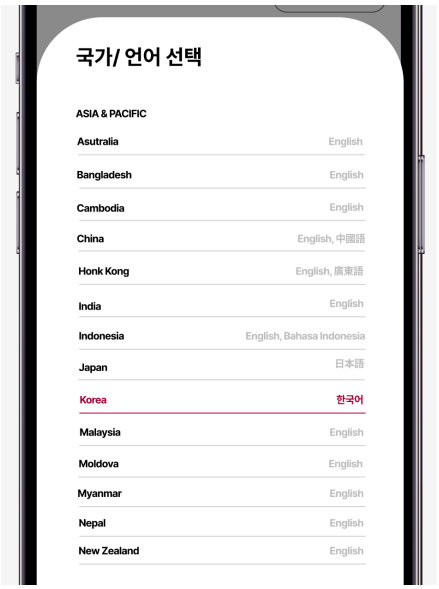

Fig. 42. ID:025, HOLME-Login-Language select

ID	Name	Description
025	HOLME-Login-Language Select	Users should have the capability to choose their preferred language from the available options. Upon changing the language, both the interface and text content should seamlessly switch to the selected language. The user's language preference must be retained even if they exit and restart the application. To facilitate language selection, countries worldwide should be listed continent-wise and sorted alphabetically by country code, allowing users to easily switch to the language associated with their chosen country.

D. Mainpage

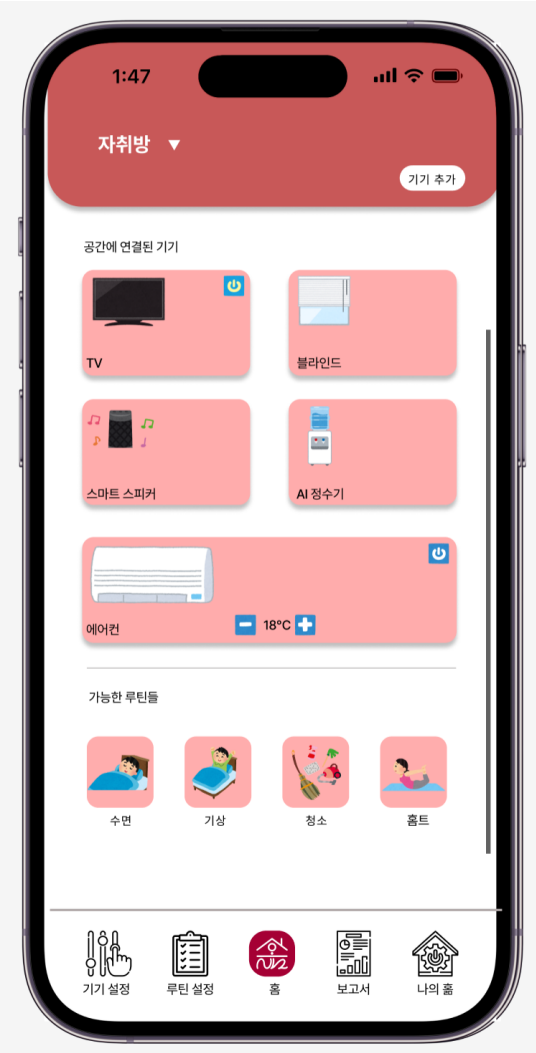

Fig. 43. ID:026, HOLME-Mainpage

ID	Name	Description
026	HOLME-Mainpage	The main page is responsible for managing 'Virtual space' and controlling various smart home functions. The main page should include the following key functions: 'Virtual space management settings,' 'Remote device control,' 'Routine execution,' and 'Device addition.'
027	HOLME-Mainpage-display	When the user selects the current space, the IoT devices and routines associated with that virtual space will be displayed.

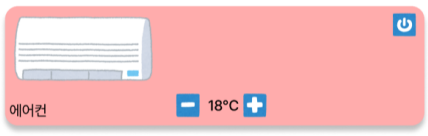

Fig. 44. ID:028, HOLME-Mainpage-Simple Control

ID	Name	Description
028	HOLME-Mainpage-Simple Control	On the main page, designated as the current space, users should have the ability to effortlessly manage connected IoT devices remotely. This includes performing simple functions such as temperature control and power on/off.

Fig. 45. ID:029, HOLME-Mainpage-Detailed Control

ID	Name	Description
029	HOLME-Mainpage-Detailed Control	Users should be able to operate all functions of the device from this page. Furthermore, users should be able to utilize these features through an artificial intelligence speaker.

(+) Instances

(1) AirConditioner

ID	Name	Description
030	HOLME-Instances-AirConditioner-Check Temperature	The user must be able to check the temperature. The temperature should be represented as an integer in degrees Celsius (°C). Additionally, it should be capable of displaying temperatures ranging from -20°C to 50°C.
031	HOLME-Instances-AirConditioner-Check Humidity	The user must be able to check the humidity. Humidity should be represented as an integer in percentage, and it should be capable of displaying values ranging from 0 percentage to 100 percentage.
032	HOLME-Instances-AirConditioner-Adjust Temperature	The user should have the ability to adjust the temperature, which is divided into temperature increase and temperature decrease. Each press results in a one-degree Celsius change. Temperature adjustment is limited to the range of 18 to 28 degrees Celsius.
033	HOLME-Instances-AirConditioner-Control Power	The user must be able to control the power, indicating the ability to turn the air conditioner on and off.
034	HOLME-Instances-AirConditioner-Select Opertating mode	The user must be able to select the operating mode, cycling through the modes with each button press. This should be done in a circular manner, meaning that once the list of modes is traversed, it should return to the initial mode.
035	HOLME-Instances-AirConditioner-Adjust direction of the airflow	User must be able to adjust the direction of the airflow (up and down).
036	HOLME-Instances-AirConditioner-Adjust Fan Speed	The user must be able to adjust the fan speed, which is set to low/medium/high. It should change by pressing the buttons.
037	HOLME-Instances-AirConditioner-Run SmartCare	The user can run Smart Care for numerical feedback and detailed information on the system's health and performance, allowing the air conditioner to conduct a comprehensive self-assessment with specific diagnostic data and suggested actions.
038	HOLME-Instances-AirConditioner-Adjust Screen brightness	The user must be able to adjust the screen brightness. The screen brightness is divided into dim/normal/bright, and it should be switchable by pressing the buttons.
039	HOLME-Instances-AirConditioner-Activate Power saving modes	User must be able to activate and deactivate the power-saving mode.
040	HOLME-Instances-AirConditioner-Activate Drying mode	User must be able to activate and deactivate the automatic drying mode.
041	HOLME-Instances-AirConditioner-Activate Tropical nigth sleep mode	User must be able to activate and deactivate the tropical night sleep mode.
042	HOLME-Instances-AirConditioner-Set On-time reservation by the hour	The user must be able to set on-time reservations by the hour. Each press of the button increases the reservation time by one hour, with a maximum reservation of up to 24 hours.
043	HOLME-Instances-AirConditioner-Set Off-time reservation by the hour	User must be able to set off-time reservations by the hour. Each press of the button increases the reservation time by one hour, with a maximum reservation of up to 24 hours.
044	HOLME-Instances-AirConditioner-Cancel specified reservations	User must be able to cancel specified reservations.

(2) LightBulb [9]

(3) Refrigerator [10]

(5) WaterDispensor [11]

(6) Television [12]

(7) Soundbar [13]

(8) MassageChair [14]

(9) Blind [15]

E. Mainpage(.cont)

Fig. 46. ID:045, HOLME-Mainpage-QR Code Button

ID	Name	Description
045	HOLME-Mainpage-QR Code Button	Users should be able to touch the “QR Code” button on the main page to transition to the QR code recognition screen.

Fig. 47. ID:046, HOLME-Mainpage-QRcode

ID	Name	Description
046	HOLME-Mainpage-QRcode recognition	Users should be able to easily add devices through QR code recognition. QR code recognition should operate swiftly and prioritize user convenience.

Fig. 48. ID:047, HOLME-Mainpage-RoutineExcution

ID	Name	Description
047	HOLME-Mainpage-Routine Excution	Users should be able to execute pre-set routines on the main page. While only 4 are visible initially, users should be able to slide horizontally to view other routines.

Fig. 49. ID:048, HOLME-Mainpage-CurrentSpace

ID	Name	Description
048	HOLME-Mainpage-CurrentSpace	Add a label at the top of the main page to indicate “Current Space,” clearly displaying the currently selected virtual space to the user.

Fig. 50. ID:049, HOLME-Mainpage-Space List Button

ID	Name	Description
049	HOLME-Mainpage-Space List Button	Users should be able to check the list of virtual spaces by clicking a button.

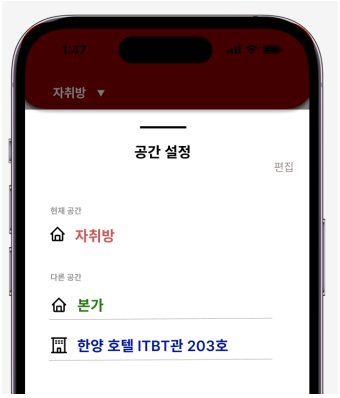

Fig. 51. ID:050, HOLME-Mainpage-Virtual Space List

ID	Name	Description
050	HOLME-Mainpage-Virtual Space List	Users should be able to check the list of virtual spaces and, at any point, designate a different virtual space as the current space. Note that the maximum number of displayed spaces is limited to 10.

Fig. 52. ID:051, HOLME-Mainpage-Space Change

ID	Name	Description
051	HOLME-Mainpage-Space Change	Users should be able to set a different virtual space as the current space at any time.

Fig. 53. ID:052, HOLME-Mainpage-Space Edit Button

ID	Name	Description
052	HOLME-Mainpage-Space Edit Button	Users should be able to add, edit, and delete spaces by pressing the edit button in the space list.

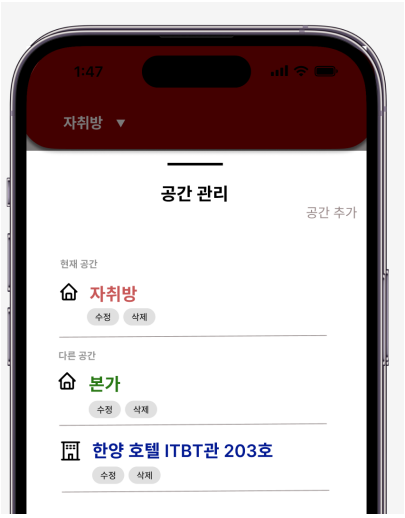

Fig. 54. ID:053, HOLME-Mainpage-Space Management

ID	Name	Description
053	HOLME-Mainpage-Space Management	Users should be able to add new virtual spaces, as well as edit or delete existing ones, on the ‘Space Management’ page. Each virtual space is represented as an independent space where users can configure and manage different smart home environments.

Fig. 55. ID:054, HOLME-Mainpage-Add Space Button

ID	Name	Description
054	HOLME-Mainpage-ADD Space Button	Users should be able to click the ‘Add Space’ button to navigate to the ‘Add Space’ page.

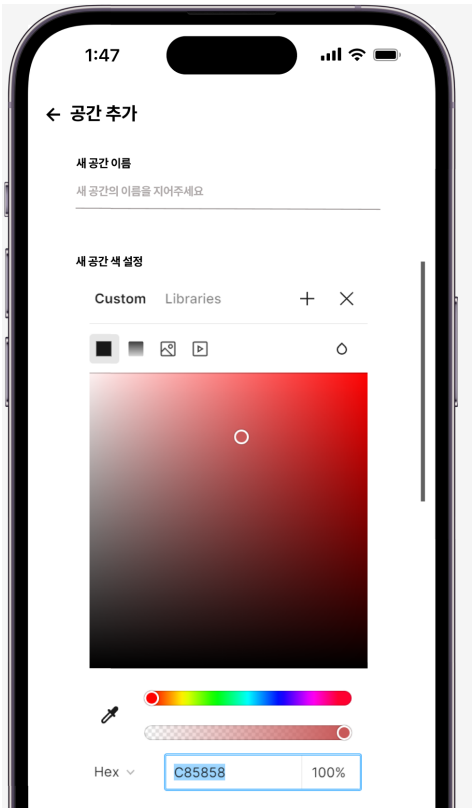

Fig. 56. ID:055, HOLME-Mainpage-Add Space Page

ID	Name	Description
055	HOLME-Mainpage-ADD Space Page	Users should be able to specify the name and color of the space when adding it on the ‘Add Space’ page. The name should be limited to 50 characters, and for the color, users can either input RGB values or choose from a provided palette.

Fig. 57. ID:056, HOLME-Mainpage-Edit Space Button

ID	Name	Description
056	HOLME-Mainpage-Edit Space Button	Users should be able to click the ‘Edit Space’ button to navigate to the ‘Edit Space’ page.

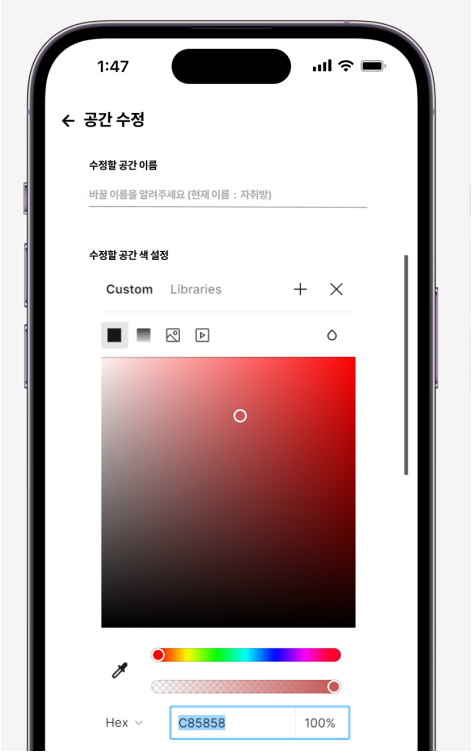

Fig. 58. ID:057, HOLME-Mainpage-Edit Space Page

ID	Name	Description
057	HOLME-Mainpage-Edit Space Page	On the ‘Space Edit’ page, users should be able to see the current space name and have the option to set a new name and color for the space they intend to modify. The name should not exceed 50 characters, and for the color, users can either input RGB values or choose from a provided palette.

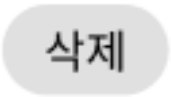

Fig. 59. ID:058, HOLME-Mainpage-Delete Space Button

ID	Name	Description
058	HOLME-Mainpage-Delete Space Button	Users should be able to see a popup window that provides a cautionary message when they press the 'Delete Space' button.

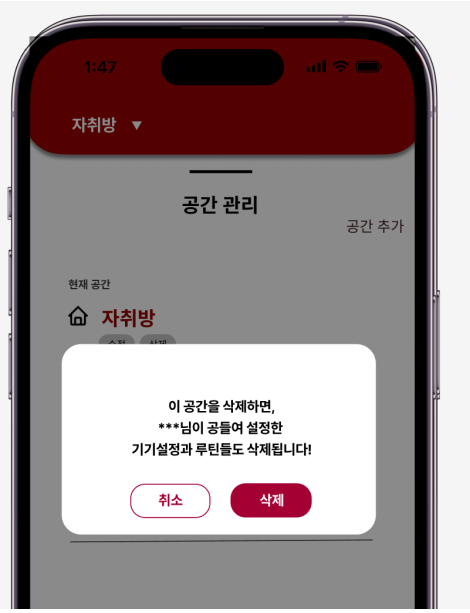

Fig. 60. ID:059, HOLME-Mainpage-Delete Space Popup

ID	Name	Description
059	HOLME-Mainpage-Delete Space Popup	In a pop-up window, when deleting a space, the user must be informed that settings and routines will also be deleted, and the user should be able to choose between 'Cancel' or 'Delete' buttons to perform their desired action.

F. Menubar

Fig. 61. ID:060, HOLME-Menubar

ID	Name	Description
060	HOLME-Menubar	The menu bar is responsible for managing 'My settings.' The menu bar should consist of the following five items: 'Device Settings', 'Routine Settings', 'Home', 'Report', 'Settings'.

G. Device Setting

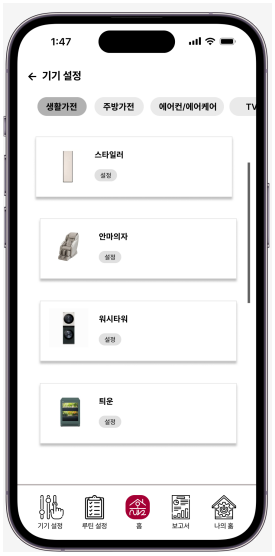

Fig. 62. ID:061, HOLME-Devices Setting

ID	Name	Description
061	HOLME-Device Setting	Predefined detailed settings for various types of IoT devices must be storable. Here, the term "devices" refers to virtual devices, and settings should be configurable for devices that may not have these settings in reality. After users connect a device to a virtual space, the actual device settings should be overridden with the predefined settings.

Fig. 63. ID:062, HOLME-Devices Setting-Category

ID	Name	Description
062	HOLME-Device Setting Category	Users should be able to view various pre-categorized types of home appliances in the 'Device Settings' menu. Here are examples of device types that users can configure:Household Appliances(washing machine,refrigerator), Kitchen Appliances(microwave,coffee machine) and so on.

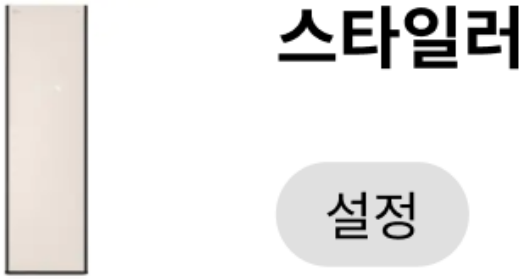

Fig. 64. ID:063, HOLME-Devices Setting Button

ID	Name	Description
063	HOLME-Device Setting Button	When users press the 'Setting' button for the respective device type, they should be taken to a page where they can configure detailed settings.

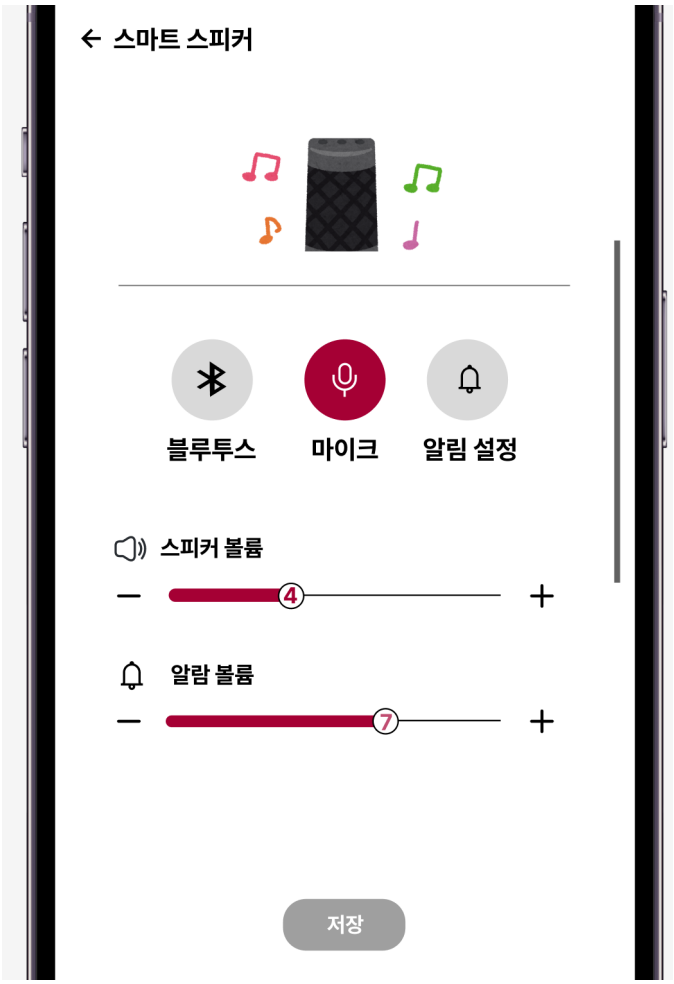

Fig. 65. ID:064, HOLME-Devices Setting Page

ID	Name	Description
064	HOLME-Device Setting Page	The settings screen for each device should enable users to save and modify detailed settings for that particular device. Additionally, users should be able to make adjustments to specific settings, with modifications working in increments of one for positive integers or performing On/Off functions.
065	HOLME-Device Setting Save	The saved settings should be applied when the device is controlled through the app.

H. Routine Setting

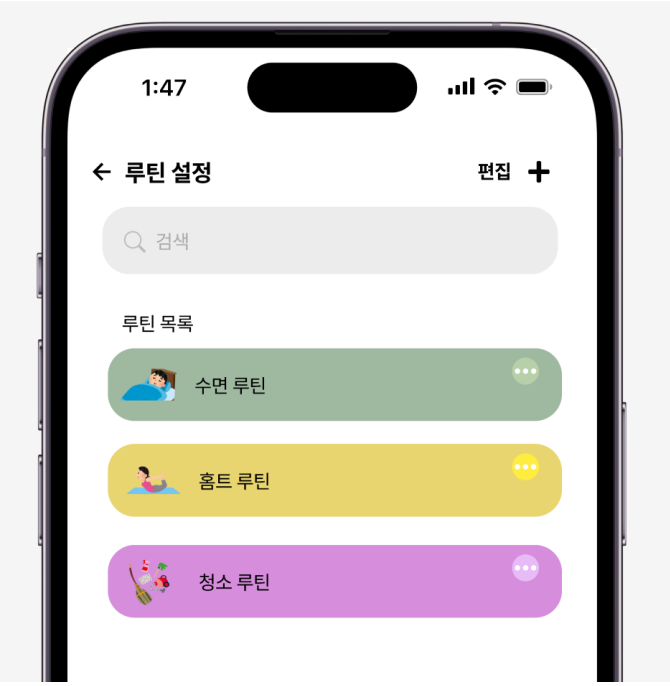

Fig. 66. ID:066, HOLME-Routine Setting

ID	Name	Description
066	HOLME-Routine Setting	Users should be able to view routines on this page, and they should also have the ability to search, edit, and add routines. The maximum number of displayed routines is 50, and users can scroll down to see more.

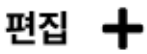

Fig. 67. ID:067, 068 HOLME-Routine Setting-Edit/Add Buttons

ID	Name	Description
067	HOLME-Routine Setting-EditButton	Users should be able to go to the 'Routine Edit' page by clicking on this button.
068	HOLME-Routine Setting-AddButton	Users should be able to go to the 'Add Routine' page by clicking on this button.

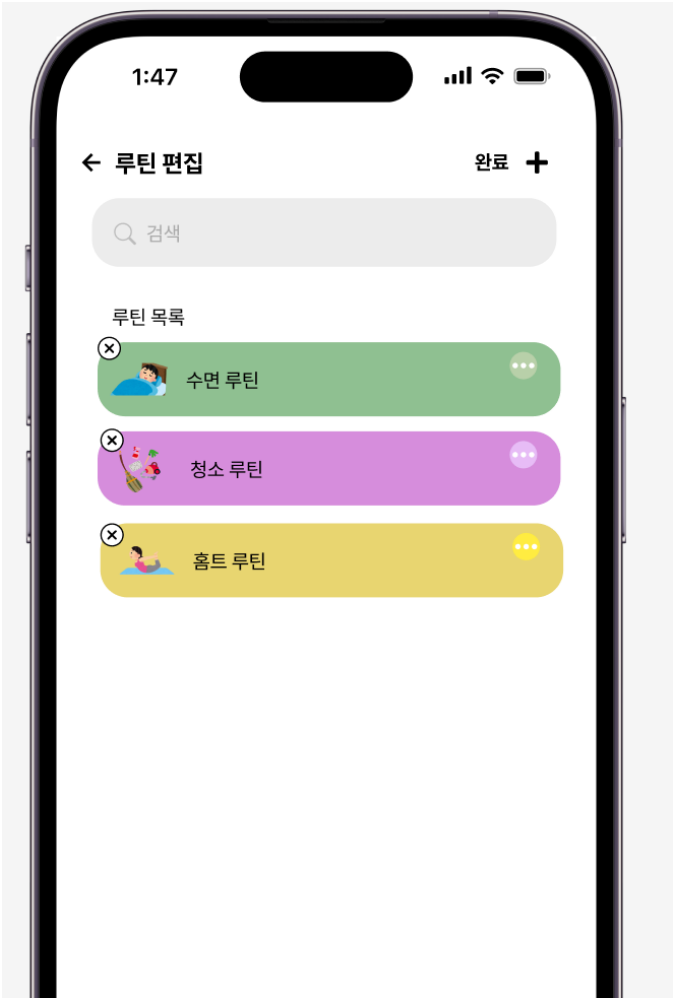

Fig. 68. ID:069, 070, 071 HOLME-Routine Setting-Routin Edit Page

ID	Name	Description
069	HOLME-Routine Setting-Rearrange Routines	Users should be able to long-press a routine to make it floating, allowing them to rearrange routines in their desired order.
070	HOLME-Routine Setting-Delete Rotuine	Users should be able to delete the desired routine by clicking the 'X' button in the top left corner of the routine.
071	HOLME-Routine Setting-Done Button	Users should be able to return to the previous page by clicking the 'Done' button once they have finished editing the routine.

Fig. 69. ID:072, HOLME-Routine Setting-Searching

ID	Name	Description
072	HOLME-Routine Setting-Searching	Users should be able to search for their desired routine using the search bar.

Fig. 70. ID:073, HOLME-Routine Setting-edit button for that specific routine

ID	Name	Description
073	HOLME-Routine Setting-edit button for specific routine.	Users should be able to go to a page where they can edit the specific routine by clicking this button.

Fig. 71. ID:074, 075, 076 HOLME-Routine Setting-specific routine Edit Page

ID	Name	Description
074	HOLME-Routine Setting-Specific Routine Edit-Appearance Edit Button	Users should be able to navigate to the 'Edit Appearance' page by clicking the icon to the right of the routine name.
075	HOLME-Routine Setting-Specific Routine Edit-Rearrange Routines	Users should be able to long-press a routine to make it floating, enabling them to rearrange routines in their desired order. The maximum number of displayed actions is 50, and users can scroll down to see more.
076	HOLME-Routine Setting-Specific Routine Edit-Add Action Button	Users should be able to go to a page where they can browse and select the desired actions by clicking the 'Add Action' button.

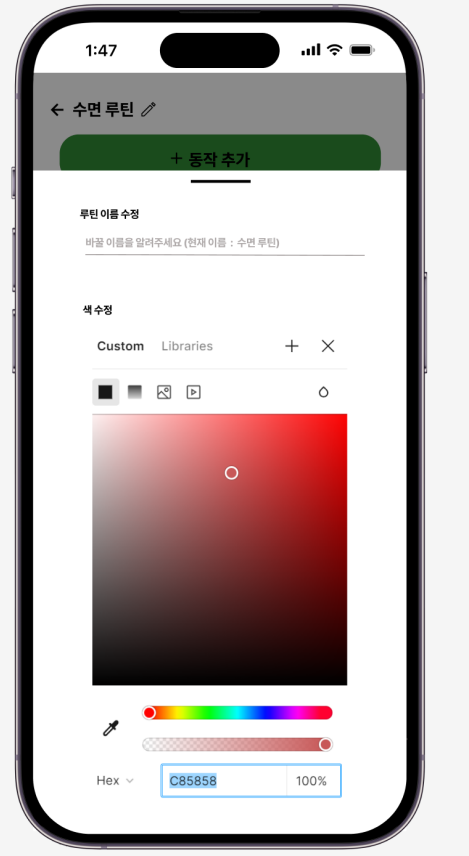

Fig. 72. ID:077, 078 HOLME-Routine Setting-specific routine-Edit Appearance

ID	Name	Description
077	HOLME-Routine Setting-Specific Routine Edit-Appearance Edit-Name	Users should be able to see the current name of the routine on this page and have the option to change it to a new desired name. The name should be limited to a maximum of 50 characters.
078	HOLME-Routine Setting-Specific Routine Edit-Appearance Edit-Color	Users should be able to view the current color of the routine on this page and have the option to change it to a new desired color. Users can either input RGB values or choose from a provided palette to select the color they prefer.

Fig. 73. ID:079 HOLME-Routine Setting-specific routine-Add Action

ID	Name	Description
079	HOLME-Routine Setting-Specific Routine Edit-Add Action	Users should be able to add the desired actions of their preferred devices on the 'Add Action' page.

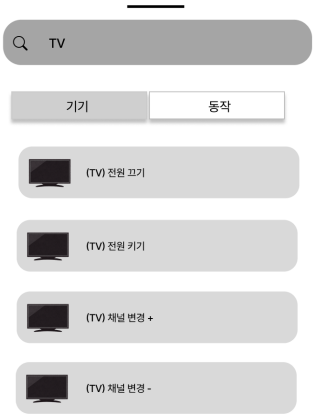

Fig. 74. ID:080 HOLME-Routine Setting-specific routine-Add Action-Search

ID	Name	Description
080	HOLME-Routine Setting-Specific Routine Edit-Add Action-Search	Users should be able to search for the desired content on the 'Add Action' page.

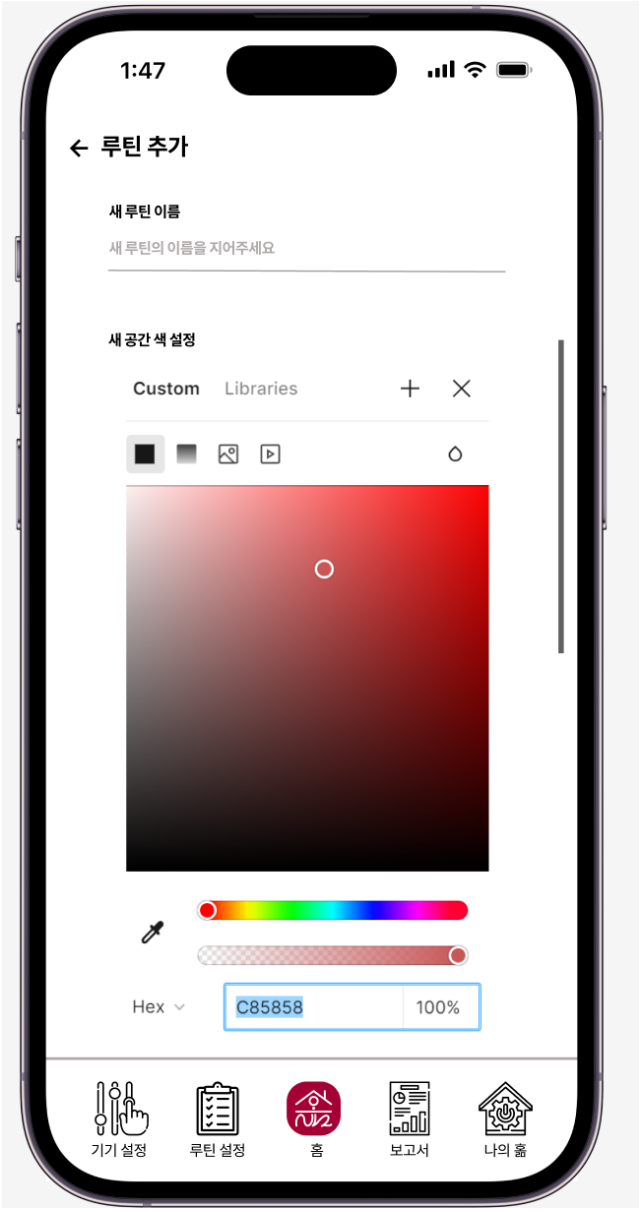

Fig. 75. ID:081 HOLME-Routine Setting-Add Routin

ID	Name	Description
081	HOLME-Routine Setting-Add Routine	Users should be able to create a new routine with the desired name and color. The name should be limited to a maximum of 50 characters, and for the color, users can either input RGB values or choose from a provided palette to select the color they prefer.

I. HOLME Button

Fig. 76. ID:082 HOLME-Routine Setting-Add Routin

ID	Name	Description
082	HOLME-Menubar-HOLME Button	Users should be able to return to the main screen by pressing the HOLME button.

J. Report

Fig. 77. ID:083 HOLME-Report Page

ID	Name	Description
083	HOLME-Report Page	Users should be able to access reports on their completed activities and events. The report should include AI-driven substitutions for items that can be replaced and those that cannot, taking into consideration their replaceability. The maximum number of displayed reports is limited to 50.

K. Settings

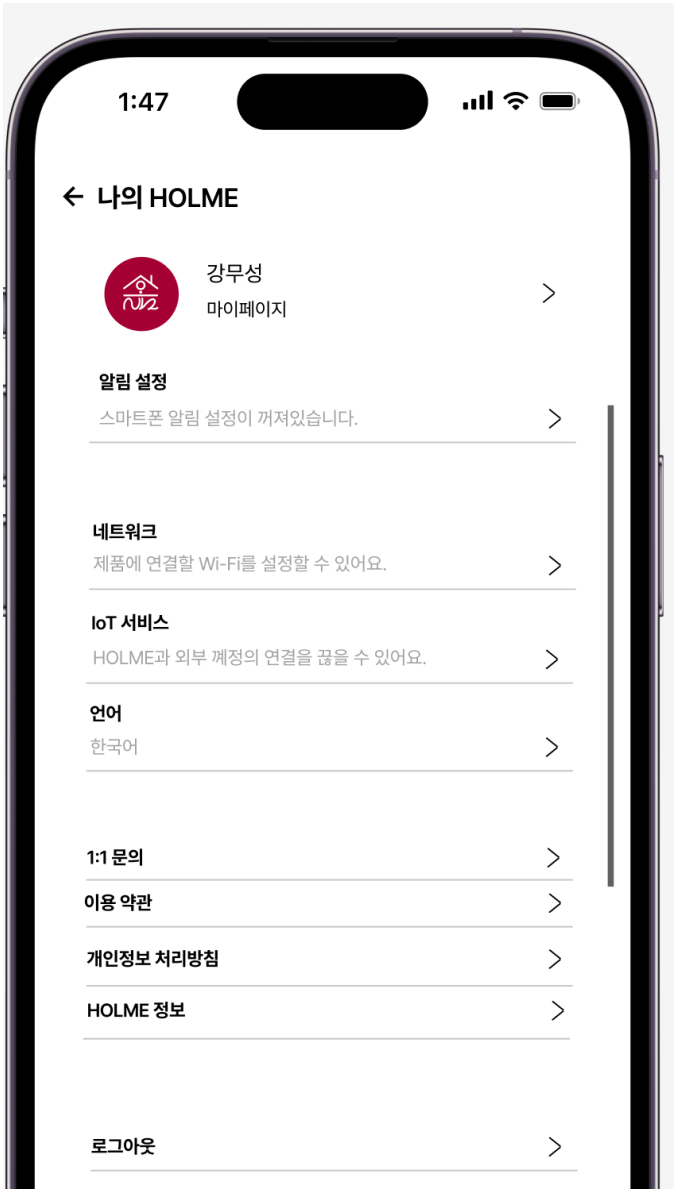

Fig. 78. ID:084 HOLME-Setting Page

ID	Name	Description
084	HOLME-Settig Page	Users should be able to navigate from the 'Settings' page to 'My Page' or access options like 'Notification Settings,' 'Network,' 'IoT Services,' 'Change Language,' 'Contact Us,' 'View Terms of Service,' 'View Privacy Policy,' 'View HOLME Information,' and 'Log Out.'

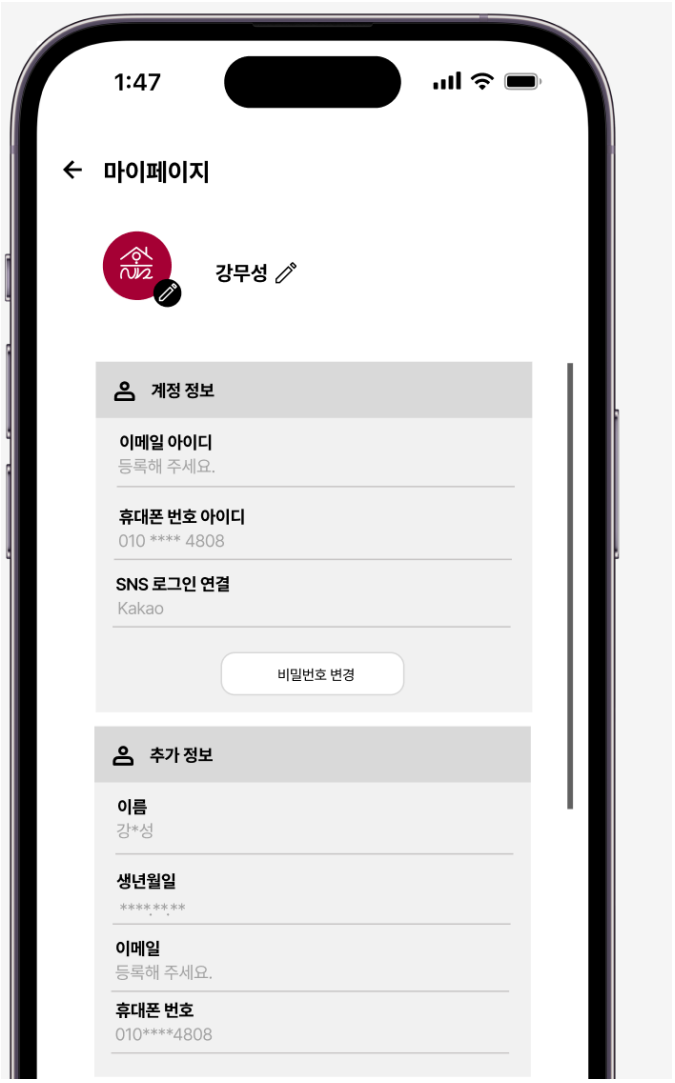

Fig. 79. ID:085 HOLME-Setting Page-MyPage

ID	Name	Description
085	HOLME-Settig Page-MyPage	Users should be able to change their nickname and profile picture on the 'My Page' and review their account information and additional details. Additionally, they should have the ability to change their password.

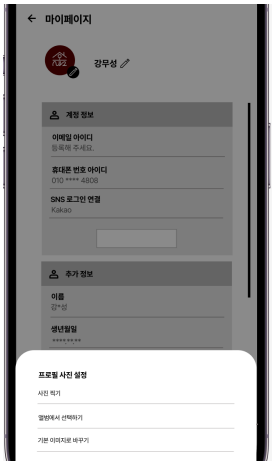

Fig. 80. ID:086 HOLME-Setting Page-MyPage-ProfileEdit

ID	Name	Description
086	HOLME-Settig Page-MyPage-Profile Edit	Users should be able to change their profile picture with three options: (1) Take a photo, (2) Select from the album, and (3) Change to a default image.

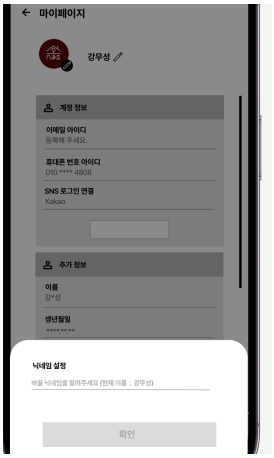

Fig. 81. ID:087 HOLME-Setting Page-MyPage-NicknameEdit

ID	Name	Description
087	HOLME-Settig Page-MyPage-Nickname Edit	Users should have the ability to change their nickname, and they should be able to view their current name while doing so.

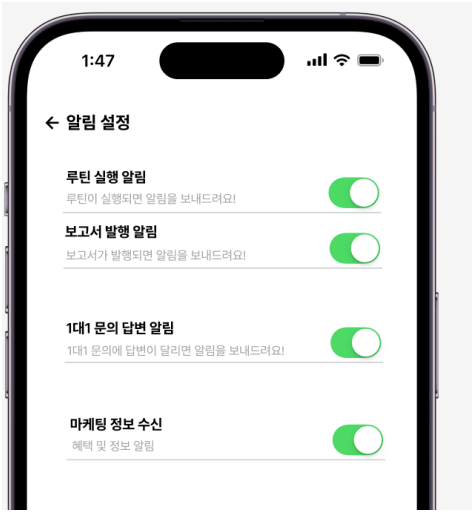

Fig. 82. ID:088 HOLME-Setting-Notification

ID	Name	Description
088	HOLME-Settig Page-Notification	Users should be able to toggle notification options on and off for different types of notifications within the notification settings.

Fig. 83. ID:089 HOLME-Setting-Network

ID	Name	Description
089	HOLME-Settig Page-Network	Users should be able to add new networks in the network settings.

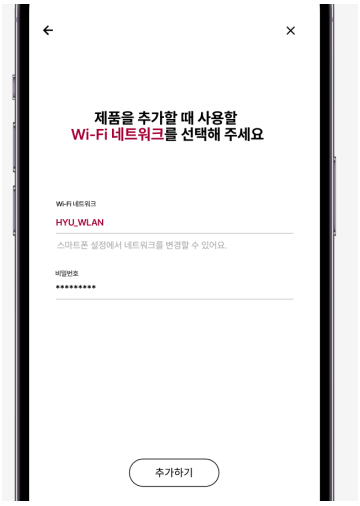

Fig. 84. ID:090 HOLME-Setting-Add Network

ID	Name	Description
090	HOLME-Settig Page-Add Network	Users should be able to add new networks in the network settings.

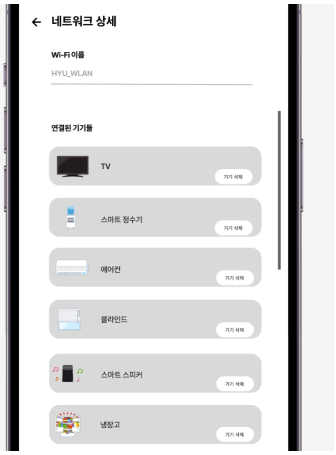

Fig. 85. ID:091 HOLME-Setting-Network Details

ID	Name	Description
091	HOLME-Settig Page-Network Details	Users should be able to view and remove devices connected to a specific Wi-Fi network from the 'Network Details' section.

Fig. 86. ID:092 HOLME-Setting-IoT Services

ID	Name	Description
092	HOLME-Settig Page-IoT Services	Users should be able to view the connected IoT services from the 'IoT Services' section. If connected IoT services exist, users should be able to import device lists or routines used in those services into HOLME.

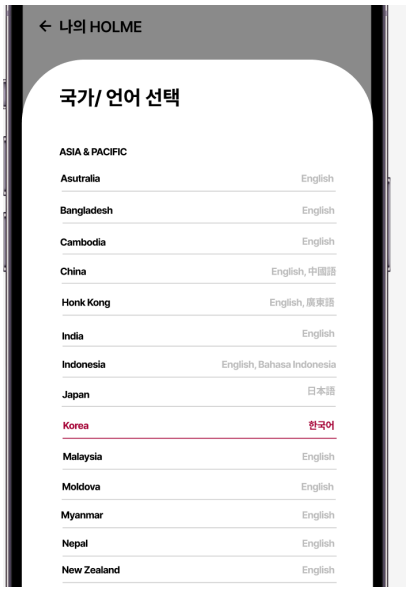

Fig. 87. ID:093 HOLME-Setting-Change Language

ID	Name	Description
093	HOLME-Settig Page-Change Language	Users should have the capability to choose their preferred language from the available options. Upon changing the language, both the interface and text content should seamlessly switch to the selected language. The user's language preference must be retained even if they exit and restart the application. To facilitate language selection, countries worldwide should be listed continent-wise and sorted alphabetically by country code, allowing users to easily switch to the language associated with their chosen country..

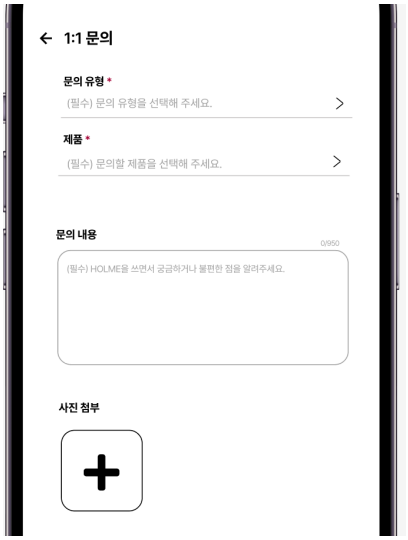

Fig. 88. ID:094 HOLME-Setting-1:1 Inquiry

ID	Name	Description
094	HOLME-Settig Page-1:1 Inquiry	Users should be able to reach out to HOLME through the '1:1 Inquiry' section, and they should also have the option to attach photos when submitting their inquiries. The title should be limited to 50 characters, and the content should not exceed 950 characters. Additionally, the size of attached files should be within 10MB.

Fig. 89. ID:095 HOLME-Setting-Terms of Service

ID	Name	Description
095	HOLME-Settig Page-Terms of Service	Users should be able to review the terms of service through the 'Terms of Service' section.

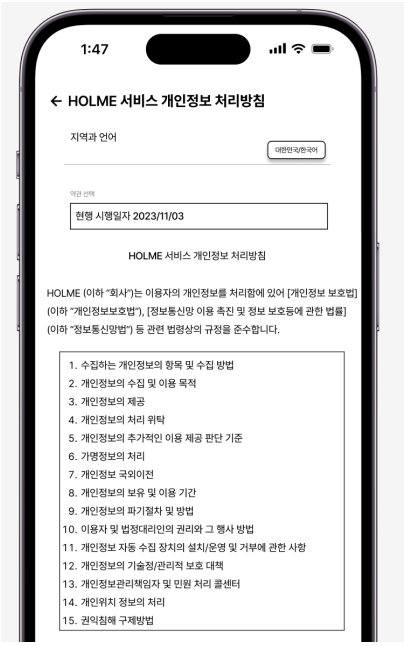

Fig. 90. ID:096 HOLME-Setting-Privacy Policy

ID	Name	Description
096	HOLME-Settig Page-Privacy Policy	Users should be able to review HOLME's privacy policy through the 'Privacy Policy' section, and they should also be able to set their region and language preferences within this section.

Fig. 91. ID:097 HOLME-Setting-HOLME Information

ID	Name	Description
097	HOLME-Settig Page-HOLME Information	Users should be able to check HOLME's version and open-source licenses through the 'HOLME Information' section.

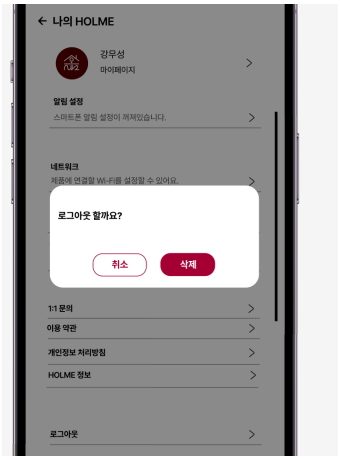

Fig. 92. ID:098 HOLME-Setting-Log Out

ID	Name	Description
098	HOLME-Settig Page-Log Out	Users should be able to log out through the 'Log Out' option.

V. ARCHITECTURE DESIGN

A. Overall Architecture

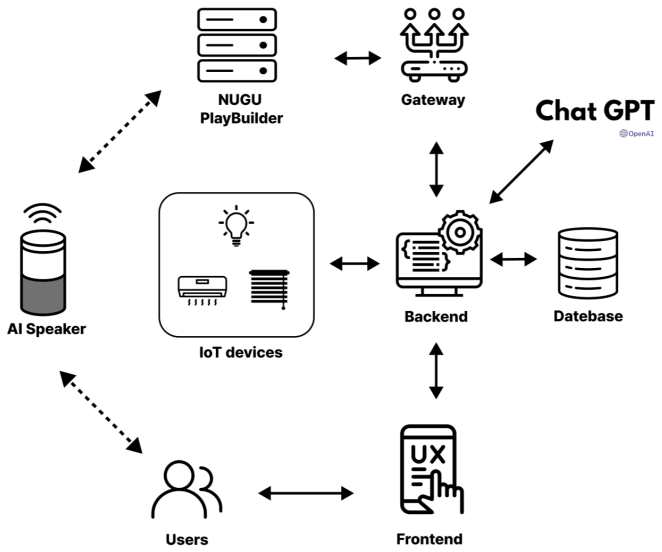

Fig. 93. Overall Architecture

HOLME users should be able to interact with HOLME's backend using either the frontend or NUGU. To achieve this, we have 4 modules: Frontend, Backend, Instances, and NUGU Playbuilder.

The first module is the "Frontend." We designed the HOLME application using React Native to enable users to directly interact with the app. Users can recognize QR codes on the main page, manipulate devices/routines, and customize various settings according to their preferences.

The second module is the "Backend." We utilized Spring Boot to build the application server, with Hibernate serving as the connected database. The backend performs tasks requested by users. Our application includes functions such as user registration/login, space configuration, QR code recognition, device control, and routine execution. The database contains tables for instance configuration, users, and reports. When information is received from users, the server stores it in the database.

The third module is the "Instances." During the progress of our project, we faced challenges in finding devices that actively support the MATTER protocol. As a solution, we created virtual instances of IoT devices. These instances play the role of devices supporting the MATTER protocol. Additionally, we established a hub in this module to connect these virtual instances to our application.

The fourth module is "NUGU Playbuilder." This service, provided by SKT, allows developers to create services interacting with NUGU. NUGU plays a role in enabling users to use HOLME through voice commands. Users can manipulate HOLME's main features through NUGU. To facilitate communication between NUGU Playbuilder and the backend, we added a gateway structure built with Flask.

B. Directory Organization

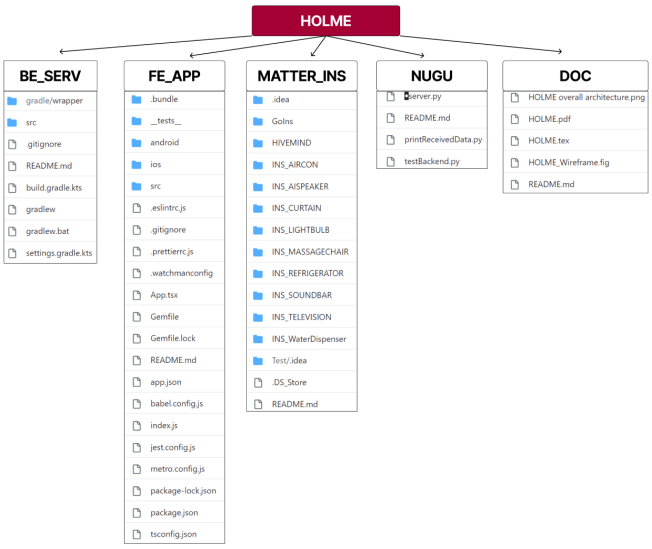

Fig. 94. Overall Architecture

HOLME consists of five GitHub repositories: FE_APP, BE_SERV, MATTER_INS, NUGU, and DOC.

The FE_APP repository is used for developing the frontend application of the HOLME project. This repository contains code and files responsible for implementing the overall design and functionality required for users to interact with the HOLME application. The primary focus is on enhancing HOLME's user

interface and user experience.

The BE_SERV repository is utilized for implementing the backend services of the HOLME project. It contains code and files needed to handle various tasks requested by users of the HOLME application. The server-side logic responsible for HOLME's core functionalities is primarily implemented in this repository.

The MATTER_INS repository contains code for virtual instances that simulate the role of actual MATTER devices in the HOLME project. The NUGU repository includes code for integrating and utilizing NUGU in the HOLME project.

The DOC repository is dedicated to documenting the project and includes Latex code and PDF files for this purpose.

TABLE II
DIRECTORY ORGANIZTION-FRONTEND-1

Directory	File Name
HOLME_FE_APP/src/api/ping	sendPingRequest.ts types.d.ts
HOLME_FE_APP/src/api/synk	sendSyncRequest.ts types.d.ts
HOLME_FE_APP/src/screen /BreifingBoard	BrifingBoard.tsx
HOLME_FE_APP/src/screen /BreifingBoard/api	fetchAllReport.ts
HOLME_FE_APP/src/screen /BreifingBoard/types	types.d.ts
HOLME_FE_APP/src/screen /Loading	LoadingScreen.tsx
HOLME_FE_APP/src/screen /Loading/logic	navigateToScreen.ts
HOLME_FE_APP/src/screen /Login/api	sendLoginRequest.ts
HOLME_FE_APP/src/screen /Login/logic	LoginLogics.ts
HOLME_FE_APP/src/screen /Login/types	types.d.ts
HOLME_FE_APP/src/screen /Login/ui	LoginScreen.tsx
HOLME_FE_APP/src/screen /Login/ui/assets	string.ts
HOLME_FE_APP/src/screen /Login/ui/components	accountUtilsComponent.tsx languageButton.tsx loginButtonComponent.tsx loginFormComponent.tsx snsLoginButtonComponent.tsx
HOLME_FE_APP/src/screen /Mainpage	MainScreen.tsx QRCodeScanner
HOLME_FE_APP/src/screen /Mainpage/api	fetchSettingData.ts sendPingRequest.ts
HOLME_FE_APP/src/screen /Mainpage/assets	line.png
HOLME_FE_APP/src/screen /Mainpage/components/buttons	adddeviceButton.tsx roomSettingButton.tsx
HOLME_FE_APP/src/screen /Mainpage/components/modals	BottomSheet.tsx modalcontent.tsx

TABLE III
DIRECTORY ORGANIZTION-FRONTEND-2

Directory	File Name
HOLME_FE_APP/src/screen /Mainpage/logic	syncLogic.ts
HOLME_FE_APP/src/screen /Mainpage/menubar	menubar.tsx
HOLME_FE_APP/src/screen /Mainpage/types	types.d.ts
HOLME_FE_APP/src/screen /WelcomeScreen/types	types.d.ts
HOLME_FE_APP/src/screen /welcomeScreen/ui	WelcomeScreen.tsx
HOLME_FE_APP/src/screen /welcomeScreen/ui/components	Layer.tsx

TABLE IV
DIRECTORY ORGANIZTION-BACKEND-1

Directory	File Name
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app	BeAppApplication.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/entity	instances.kt MultipleRequrets.kt ServiceRequest.kt SingleRequest.kt MultipleResponses.kt MultipleResponseService.kt ServiceResponse.kt SingleRespones.kt SingleResponseService.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/ping	PingController.kt PingRequest.kt PingResponse.kt instanceMapManage.kt PingService.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/report	ReportController.kt ReportRequest.kt ReportResponse.kt OpenAIService.kt ReportService.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/setting/download	DownloadContoller.kt DownloadRequest.kt DownloadResponse.kt DownloadService.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/setting/upload	UploadContoller.kt UploadRequest.kt UploadResponse.kt UploadService.kt _REST_TEST_DATA.json
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/singin	SigninContoller.kt SigninRequest.kt SigninResponse.kt Signinervice.kt

TABLE V
DIRECTORY ORGANIZTION-BACKEND-2

Directory	File Name
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/singup	SignUpContoller.kt SignUpRequest.kt SignUpResponse.kt SignUpService.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /api/sync	SyncContoller.kt SyncRequest.kt SyncResponse.kt SyncType.kt SyncInstanceTypeFactory.kt SubroutineManager.kt SyncRequestService.kt sendReportService.kt sync_test_data.json
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /utils	HashFunction.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /repository	ReportRepository.kt ServiceUserRepository.kt SettingRepository.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /entity	ServiceUser.kt InstanceSetting.kt Report.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /dto	ServiceUserDto.kt SafeServiceUserDto.kt InstanceSettingDto.kt ReportDto.kt
HOLME_BE_SERVER/src /main/kotlin/com/holme/be_app /config	WebAppConfig.kt GPTConfig.kt

TABLE VI
DIRECTORY ORGANIZTION-MATTER_INS-1

Directory	File Name
HOLME_MATTER_INS /INS_AIRCON	airconTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_AIRCON/src/core	frameHandle.go aircon.pb.go aircon.proto aircon_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_AIRCON/src/features	aircon.go
HOLME_MATTER_INS /INS_AIRCON/src/raw	on.txt
HOLME_MATTER_INS /INS_AIRCON/src/terminal	output.go

TABLE VII
DIRECTORY ORGANIZTION-MATTER_INS-2

Directory	File Name
HOLME_MATTER_INS /INS_AISPEAKER	aiSpeakerTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_AISPEAKER/src/core	frameHandle.go aispeaker.pb.go aispeaker.proto aispeaker_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_AISPEAKER/src/features	aispeaker.go
HOLME_MATTER_INS /INS_AISPEAKER/src/raw	on.txt
HOLME_MATTER_INS /INS_AISPEAKER/src/terminal	output.go
HOLME_MATTER_INS /INS_CURTAIN	curtainTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_CURTAIN/src/core	frameHandle.go curtain.pb.go curtain.proto curtain_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_CURTAIN/src/features	curtain.go
HOLME_MATTER_INS /INS_CURTAIN/src/raw	on.txt
HOLME_MATTER_INS /INS_CURTAIN/src/terminal	output.go
HOLME_MATTER_INS /INS_LIGHTBULB	lightbulbTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_LIGHTBULB/src/core	frameHandle.go lightbulb.pb.go lightbulb.proto lightbulb_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_LIGHTBULB/src/features	lightbulb.go
HOLME_MATTER_INS /INS_LIGHTBULB/src/raw	on.txt
HOLME_MATTER_INS /INS_LIGHTBULB/src/terminal	output.go
HOLME_MATTER_INS /INS_MASSAGECHAIR	massagechairTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_MASSAGECHAIR /src/core	frameHandle.go massagechair.pb.go massagechair.proto massagechair_grpc.pb.go init.sh instance_core.go

TABLE VIII
DIRECTORY ORGANIZTION-MATTER_INS-3

Directory	File Name
HOLME_MATTER_INS /INS_MASSAGECHAIR /src/features	massagechair.go
HOLME_MATTER_INS /INS_MASSAGECHAIR /src/raw	on.txt
HOLME_MATTER_INS /INS_MASSAGECHAIR /src/terminal	output.go
HOLME_MATTER_INS /INS_REFRIGERATOR	refrigeratorTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_REFRIGERATOR /src/core	frameHandle.go refrigerator.pb.go refrigerator.proto refrigerator_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_REFRIGERATOR /src/features	refrigerator.go
HOLME_MATTER_INS /INS_REFRIGERATOR /src/raw	on.txt
HOLME_MATTER_INS /INS_REFRIGERATOR /src/terminal	output.go
HOLME_MATTER_INS /INS_SOUNDBAR	soundbarTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_SOUNDBAR/src/core	frameHandle.go soundbar.pb.go soundbar.proto soundbar_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_SOUNDBAR/src/features	soundbar.go
HOLME_MATTER_INS /INS_SOUNDBAR/src/raw	on.txt
HOLME_MATTER_INS /INS_SOUNDBAR/src/terminal	output.go
HOLME_MATTER_INS /INS_TELEVISION	televisionTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_TELEVISION/src/core	frameHandle.go television.pb.go television.proto television_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_TELEVISION /src/features	television.go
HOLME_MATTER_INS /INS_TELEVISION /src/raw	on.txt

TABLE IX
DIRECTORY ORGANIZTION-MATTER_INS-4

Directory	File Name
HOLME_MATTER_INS /INS_TELEVISION /src/terminal	output.go
HOLME_MATTER_INS /INS_WaterDispensor	waterDispensorTestData.json go.mod go.sum main.go
HOLME_MATTER_INS /INS_WaterDispensor /src/core	frameHandle.go waterDispensor.pb.go waterDispensor.proto waterDispensor_grpc.pb.go init.sh instance_core.go
HOLME_MATTER_INS /INS_WaterDispensor /src/features	waterDispensor.go
HOLME_MATTER_INS /INS_WaterDispensor /src/raw	on.txt
HOLME_MATTER_INS /INS_WaterDispensor /src/terminal	output.go
HOLME_MATTER_INS /HIVEMIND	go.mod go.sum main.go run_hivemind.sh
HOLME_MATTER_INS /HIVEMIND/core	ip.go RESTHandler.go hivemind_core.go intanceHandler.go intances.go logger.go types.go init.sh merged_pb.go merged.proto merged_grpc.pb.go
HOLME_MATTER_INS /HIVEMIND/intances	types.go
HOLME_MATTER_INS /HIVEMIND/utils	utils.go

TABLE X
DIRECTORY ORGANIZTION-NUGU

Directory	File Name
HOLME_NUGU	server.py testBackend.py printRecievedData.py

C. Module 1: Frontend

1) Purpose
To develop HOLME, we utilized React Native. The frontend plays a crucial role in providing an interface to the user. By offering input fields for users to enter information, it establishes a connection between the user and the server. Once this task is completed, the values are transmitted to the backend. Additionally, the frontend retrieves data from the database, transforms it into a format understandable to the user, and then presents it to the user.

2) Functionality
Users can create an account by entering their email, password, and ID, or by using social accounts such as KakaoTalk or Google. Additionally, they have the option to add devices through QR code recognition, operate devices, execute routines, and read reports. The frontend communicates with the backend by sending requests to display information obtained from the database.

3) Location of source code:
https://github.com/PROJECT-HOLME/HOLME_FE_APP

4) Class component
- api folder : This is a folder containing files for sending ping and sync requests.
- deepLinkingConfig.ts : This is a file that configurates links of screens.
- appNavigation.tsx : This is a file that creates native stack navigator, defines root stack screens for navigating screens.
- screen folder: This is a folder containing files for screens and their components.
- images.d.ts : This is a file that, declares modules of image files.
- navigator.d.ts : This is a file that contains navigator parameter list.
- props.d.ts : This file that defines screen props used for navigation between screens.
- sendPingRequest.ts:: This is a file for ping request process.
- types.d.ts : This is a file that defines variables needed in ping request process.
- sendSyncRequest.ts : This is a file for synchronization with backend server.
- types.d.ts : This is a file that defines variables needed in synchronization process.
- fetchAllReport.ts : This is a file that receives the message from backend server.
- types.d.ts : This is a file that defines variables needed in BriefingBoard.
- BriefingBoard.tsx : This file is a typescript file showing the briefing board page in HOLME
- natigateToScreen.ts : This file shows alert after configuration migration is finished.
- LoadingScreen.tsx : This is typescript file for showing the loading page needed when the configuration migration process is running.
- sendLoginRequest.ts : This file defines status variables needed in login function.
- LoginLogics.ts : This is a file that contains variable needed for login function.
- types.d.ts : This contains definition of parameters needed in login function.
- string.ts : This contains text for login system.
- components folder : This is a folder which contains multiple typescript files for components in LoginScreen.
- LoginScreen.tsx: This is typescript file for showing the login page in HOLME.
- fetchSettingData.ts : This is a file that defines status variables needed in configuration migration function.
- sendPingRequest.ts : This is a file file for ping request process.
- buttons folder : This is a folder that contains buttons used in MainScreen.
- devices folder : This is a folder that contains device components used in MainScreen.
- modals folder : This is a folder that contains components for modal contents used in MainScreen.

- routines folder : This is a folder that contains routine components used in MainScreen.

- syncLogic.ts : This is a file that defines logic of synchronization process.

- menubar folder : This is a folder that contains components for menubar used in HOLME screens.

- types.d.ts: This is a file that defines variables needed in MainScreen.

- MainScreen.tsx : This is typescript file for main screen of HOLME.

- QRcodeScanner.tsx : This is a typescript file for QR code scanner using backside camera.

- types.d.ts : This is the contains layer parameters for welcome screen.

- assets folder : This is a folder that contains image files used in welcome screen.

- Layer.tsx : This is a typescript file defining the layer used in welcome screen.

- WelcomeScreen.tsx: This is typescript file for welcome screen of HOLME.

D. Module 2: Backend

1) Purpose
The backend is responsible for managing servers and databases. It stores and manages data and processes actions taken by users on the client side of the application. The backend stores data generated as a result of user actions in the frontend's database and fulfills the role of retrieving data needed by the user from the frontend's database. To implement HOLME's backend, we used Spring. For the database, we employed Hibernate, known for its high compatibility with Spring Boot. Utilizing Hibernate, we stored user information necessary for the application's usage, thus constructing the backend server. Snyk is also used to analyze and manage the possible cyber threats, making the spring-boot server more secure.

2) Functionality
HOLME's server is responsible for executing tasks requested by users through the frontend or NUGU and storing corresponding values in the database. When users load or save their settings, this information is stored in the database. Additionally, after actions such as device manipulation, routine execution, and device connections, HOLME provides overall services, including generating reports.

3) Location of source code:
https://github.com/PROJECT-HOLME/HOLME_BE_SERV

4) Class component
- BeAppApplication.kt : This is a file for Spring Boot Application entrypoint.

- api/entity : This is a folder that contains entities used for api service.

- api/entity/instance/Instances.kt : This ia a file for Entity Definition for Instance (Used for MATTER communication with GoLang IOT Instances.)

- api/request : This is a folder for Entities used for request of each endpoints.

- ServiceRequest.kt : This is a file for Interface for service request.

- SingleRequest.kt : This is Class used for single service request.

- MultipleRequests.kt : This is a f Class used for multiple

service request.

- api/response : This is a folder for Entities used for response of each endpoints

- ServiceResponse.kt : This is a file for Interface for service response.

- SingleResponse.kt , SingleResponseService.kt : These are Class and Service used for single service response.

- MultipleResponses.kt , MultipleResonseService.kt : These are Class and Service used for multiple service response.

- api/ping : This is a folder that contains service to check whether the IOT instance exists in targeting place.

- PingController.kt : : This is Controller for ping service.

- PingRequest.kt, PingResonse.kt : These are Request/Response class definition for Ping Service.

- InstaneMapManager.kt : Manager class for ping response. It handles the IOT instance doesn't exists.

- api/report : This is Report service directory; contains service for report regarding setting substitution and upgrades.

- ReportController.kt : This is a Controller for report service.

- ReportRequest.kt, ReportResonse.kt : These are Request/Response class definition for Report Service

- ReportService.kt : Report service class. Contains business logic for report service.

- OpenAIService : Report service class using Open AI. Contains business logic for report service using Chat GPT.

- api/setting : This is a Setting service directory; it will be divided into uploading service and downloading service.

- setting/upload : This is a Setting upload service directory; contains service to upload IOT instance setting to database.

- UploadController.kt : This is a Controller for upload service.

- UploadRequest.kt, UploadResonse.kt : These are Request/Response class definition for Setting Upload Service.

- UploadService.kt : This is Setting upload service class. Contains business logic for setting upload service.

- REST_TEST_DATA.json : This is a file contains test data for testing setting uploads.

- setting/download : This is Setting download service directory; contains service to download IOT instance setting to database.

- DownloadController.kt : This is a controller for download service.

- DownloadRequest.kt, DownloadResonse.kt : These are Request/Response class definition for Setting download service.

- DownloadService.kt : This is Setting download service class. Contains business logic for setting download service.

- api/signin : Sign in service directory; contains service for signing in.

- SignInController.kt : This is a Controller for sing in service.

- SignInRequest.kt, SignInResonse.kt : : These are Request/Response class definition for Sign in service.

- SignInService.kt : This is sign in service class. Contains business logic for sign in service.
- api/signup : This is a Sign up service directory; contains service for signing in.

- SignUpController.kt : This is a Controller for sing up service.

- SignUpRequest.kt, SignUpResonse.kt : These are Request/Response class definition for Sign up service.

- SignUpService.kt : Sign up service class. Contains business logic for sign up service.

- api/sync : This is Setting synchronization service directory; contains service to synchronize and apply IOT instance settings to targeting place.

- SyncController.kt: This is a Controller for synchronization service.

- SyncRequest.kt, SyncResonse.kt : These are Request/Response class definition for sync Service.

- SyncInstanceTypeFactory.kt : This return a class for

instance. Used a factory pattern to do such task.

- SubroutineManager.kt : This is Manager class for sub-routines; apply subroutines based on the ping result.

- SyncRequestService.kt : This is a Synchronization service class. Contains business logic for synchronization service.

- sync_test_data.json : This contains test data for testing setting synchronization.

- HashFunction.kt : This apply SHA Hash for secret value. Used for universal encryption in backend service.

- ServiceUserRepository.kt : This is Database repository for ServiceUser table.

- SettingReposiory.kt : This is Database repository for InstanceSetting table.

- ReportRepository.kt : This is Database repository for Report table.

- ServiceUser.kt : This is a Database entity for ServiceUser table.

- InstanceSetting.kt : This is Database entity for InstanceSetting table.

- Report.kt : This is Database entity for Report table.

- ServiceUserDto.kt : This is a Dto class for ServiceUser entity.

- SafeServiceUserDto.kt : This is a Safe dto class for ServiceUser entity. (Dto class that doesn't contains password and user secret information.)

- InstanceSettingDto.kt : This is a Dto class for InstanceSetting entity.

- ReportDto.kt : This is a Dto class for Report entity.

- WebAppConfig : This set backend configuration required to be applied. (e.g. CORS configuration)
- GPTConfig : This set OpenAI configuration required for using Chat GPT.

E. Module 3: Instances

1) Purpose
   The instances are divided into two parts: the instance part that creates virtual IoT devices, and the logical hub part that can serve as a hub role even without a Matter hub.
   Firstly, we encountered challenges progressing the project due to the absence of devices that actually support the MATTER protocol. As a solution, we generated virtual IoT devices. These virtual IoT devices were instantiated to simulate the functionalities required for the project.
   Secondly, the logical hub, capable of assuming the role of a hub even in the absence of a Matter hub, is a crucial component of our project.

2) Functionality
   The first part, virtual IoT devices, operates like real IoT devices through the HOLME application or NUGU. The second part, the logical hub, can perform the role of a hub even without a Matter hub.

3) Class component
   Describing an example of an air conditioner within the instances. The file structure for the remaining instances is also identical.

- airconTestData.json : This is a Test file created for JSON-formatted data coming from the backend for testing purposes.

- go.mod : This is a File for managing dependencies in GoLang. All modules used in GoLang are maintained in the go.mod file.

- go.sum : This file lists down the checksum of direct and indirect dependency required along with the version. It is to be mentioned that the go.mod file is enough for a successful build. They why go.sum file is needed?. The checksum present in go.sum file is used to validate the checksum of each of direct and indirect dependency to confirm that none of them has been modified.

- main.go: This is a file containing the main function within the package main. For a web server (an executable), a package main with a main() function is necessary.

- /src : This is a directory where functions, structures, and other elements required for the main function are defined.

- /src/core : Handling the frame of the air conditioner

instance / The format of data exchanged with the backend server / Defining how the instance will behave when initiated and started, a core section of the instance.

- /src/core/handler : This is a Directory containing the frameHandler.

- frameHandler.go: This is a file defining how the air conditioner instance will handle a frame received from the backend server.

- instance_core.go : This is a file defining the actions to be taken when the air conditioner instance is initiated or started.

- /src/core/pbs : This is a directory containing files that define the format of data exchanged with the backend server using gRPC and redefined in the form of GoLang.

- aircon.pb.go : The output obtained by compiling the aircon.proto file using the Protocol Buffer compiler. Here, the structures and functions defined in aircon.proto are redefined in the gRPC format.

- aircon.proto : This is a file necessary for using gRPC. The format of messages exchanged with the backend server is defined here.

- aircon_grpc.pb.go : The output obtained by compiling aircon.proto using the Protocol Buffer compiler. It includes a server/client interface with basic I/O functions.

- init.sh : The shell script containing commands to compile aircon.proto and generate aircon.pb.go and aircon_grpc.pb.go.

- /src/feature : This is a directory containing definitions for functionalities that the air conditioner instance can perform, such as power on/off.

- aircon.go : This is a file containing definitions for functionalities that the air conditioner instance can perform, such as power on/off.

- /src/raw : This is a directory containing ASCII art necessary for visualizing the air conditioner instance in the terminal.

- on.txt: ASCII art of the airconditioner on.

- /src/terminal : This is a directory defining the method to print an air conditioner instance in the terminal.

- output.go : This is a file defining the method to print an air conditioner instance in the terminal.

- main.go : This is an entrypoint of hivemind (a hub).

- run_hivemind.sh : This is a shell code to initiate the hivemind (a hub).

- ip.go : This is IP definitions of IoT instances.

- RESTHandler.go : This handles REST request/response toward the backend server.

- hivemind_core.go : This file includes a core functionalities of hivemind (a hub).

- instanceHandler.go : This file Includes handler functions for instances; it sends dataframe to various arbitrary IoT instances and handle received response.

- instanceRequestHandler.go : This file includes request handler that includes a core functionalities for sending dataframe to various arbitrary IoT instances.

- intances.go : This file includes definitions of possible IoT instances.

- logger.go: This file include functionalities for logging.

- types.go : This file consists of type definitions to handle various requests and responses, as well as system utilities.

- init.sh : This is shell code that updates proto files and instances payloads/status.

- merged.pb.go, merder.proto, merged_grpc.pb.go : These are files that enable the instance availbe to do gRPC communication.

- utils.go: This file includes system utility functions.

F. Module 4: NUGU

1) Purpose
This module is created to enable users to communicate with HOLME's backend using NUGU AI SPEAKER instead of the frontend

2) Functionality
By communicating with HOLME's backend through NUGU AI SPEAKER, users can execute key functionalities of HOLME, such as device control and routine execution.

3) Class component
- server.py : This is a server that is connected to NUGU Play and is responsible for transmitting necessary information to NUGU Play or forwarding commands from NUGU Play to the backend server. It converts requests received from NUGU Play into well-structured JSON format and sends them to the backend server.

- testBackend.py : This is a test server that will function as the backend server connected to server.py.

- printReceivedData.py : This is a test file for testBackend.py that simply prints the received data as is.

VI. USE CASE

A. Loading

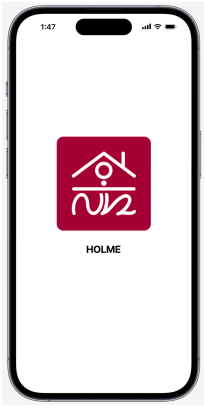

Fig. 95. Loading page

Loading Page is a page that a user will see after turning on the application. This page is turned on only while the application loads the elements needed to operate, and automatically moves on to the next page at the end of loading.

B. Tutorial

Fig. 96. Tutorial page 1

Tutorial Page 1 is the welcoming screen that greets users when they first download the HOLME application

Fig. 97. Tutorial page 2

Tutorial Page 2 explains the convenience of HOLME's feature, which allows users to connect multiple devices at once using QR code recognition.

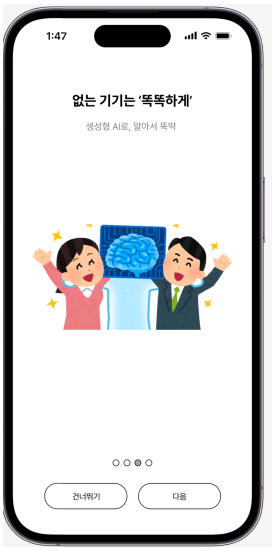

Fig. 98. Tutorial page 1

Tutorial Page 3 provides an explanation of one of HOLME's features, which is 'replaceability'.

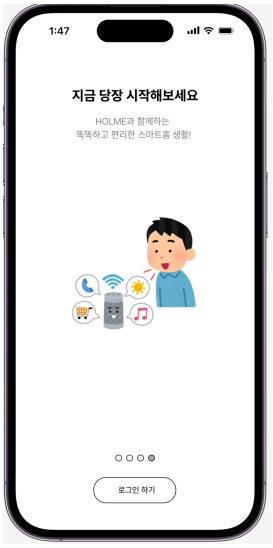

Fig. 99. Tutorial page 1

Tutorial Page 4 prompts the user to log in.

C. Login

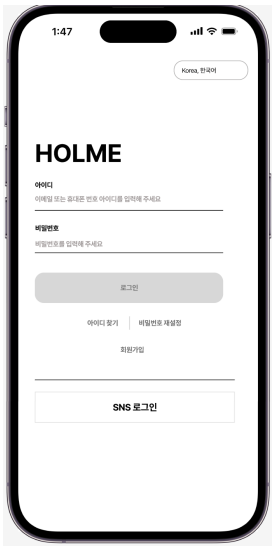

Fig. 100. Login page

Login Page is a page which allows users to log in by entering their ID and password.

Fig. 101. Login Success Pop-up

Login Success Pop-up is a pop-up that acknowledges user that their login attempt at Figure 10: Login Page is successful.

D. Sign Up

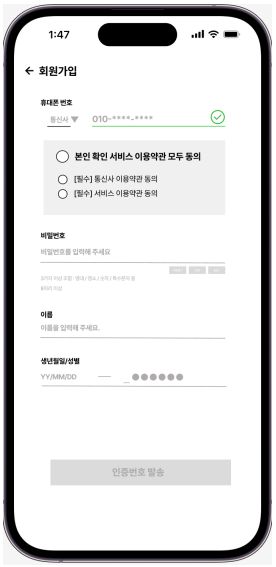

Fig. 102. Sign Up page

The user needs to authenticate their phone number through the telecommunications provider to use it as their ID. After entering the password and additional information, they can proceed to the login screen and attempt to log in with the newly created account.

E. Main page

Fig. 103. Main page

The use cases of the main page are device addition, space configuration, device manipulation, and routine execution.

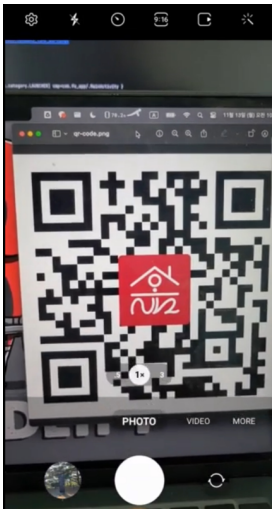

Fig. 104. QR Code recognition page

1) QR Code recognition: On the main page, users can navigate to a QR code recognition page linked to their smartphone's camera by clicking the 'QR Code' button. By scanning QR codes on this page, users can easily connect various devices to HOLME.

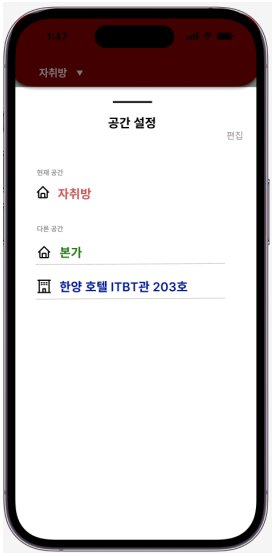

Fig. 105. Space setting page

2) Space setting: On the main page, if the user presses the downward triangle button on the right side of the current location, a list of virtual spaces that the user has set in advance will appear. On this 'Space Settings' page, the user can change the current space to another space by pressing the desired space.

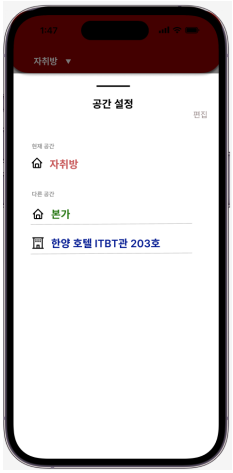

Fig. 106. Space modification page

3) Space modification: On the Space Settings page, you can click the 'Edit' button in the upper right corner to go to the Space Management page, where you can modify or delete each space.

Fig. 107. Simple contol

4) Simple control: On the main page, the user can simply control the connected devices.

Fig. 108. Simple operation

5) Detailed operation: On the main page, the user can move to the 'detailed contol' page, where they can manipulate all the functions of the device by clicking on the device.

Fig. 109. Routine execution

6) Routine execution: On the main page, users can run the routine by clicking on the routine icon they have set up.

F. Device configuration

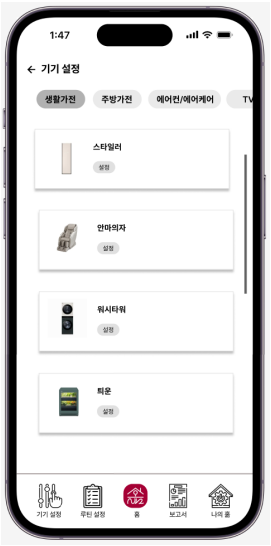

Fig. 110. Device configuration page

The user can navigate to the Device configuration page by clicking 'Device Configuration' from the menu bar. The Device configuration page allows the user to pre-set common settings that belong to a specific device group based on the criteria set out by the MATTER protocol. This supports easy migration of your settings when the user connects to another device. This page categorizes all household appliances, allowing users to preconfigure settings for devices they may not even own yet.

Fig. 111. Specific device setting page

1) Specific device setting: The user can click on a specific appliance on the Device Settings page to navigate to the specific appliance settings page, which allows the user to set the settings that the MATTER protocol can set in common for each appliance type. Upon saving these settings, they will automatically apply the next time the user connects this type of appliance to the HOLME application.

G. Routine configuration

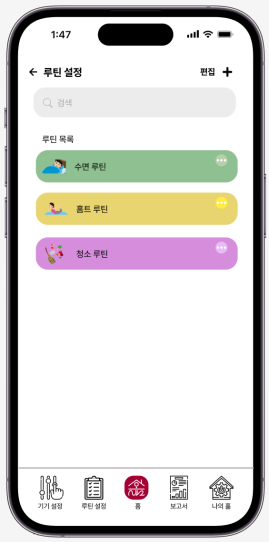

Fig. 112. Routine configuration page

Users can navigate to the Routine configuration page by clicking 'Routine Configuration' from the menu bar. The Routine configuration page lets you search routines, edit routines, add routines, and modify each routine.

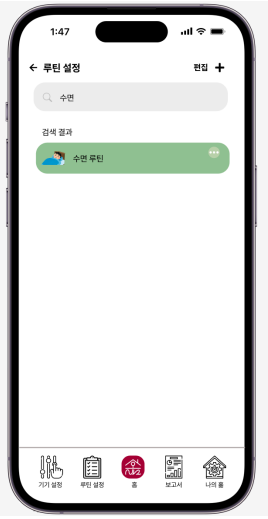

Fig. 113. Search routine

1) Search routine: Users can search for specific routines on the Routine configuration page by entering keywords into the search bar.

Fig. 114. Edit routine

2) Edit routine: The user can modify the routine by clicking the (...) button on the right side of the routine on the Routine Settings page. Features supported at this time include changing the name/color of the routine, adding actions, and reordering the set actions.

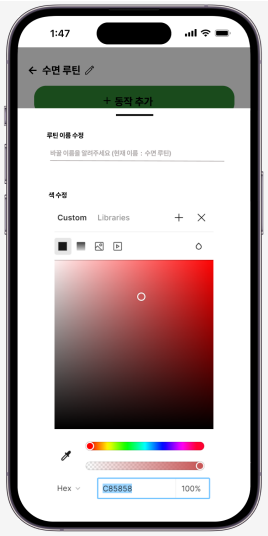

Fig. 115. Edit routine-name/color

3) Edit routine-name/color: Users can change the name and color of a routine on the Routine Edit page by clicking the pencil button to the right of the routine name.

Fig. 116. Edit routine-adding action

4) Edit routine-adding action: Users can add desired actions to a routine on the Routine Edit page by clicking the 'Add Action' button, searching for the desired action, and then adding it to the respective routine.

Fig. 117. Edit routine-search action

5) Edit routine-search action: Users should be able to search for their desired devices or actions on the 'Add Action' page.

H. Report

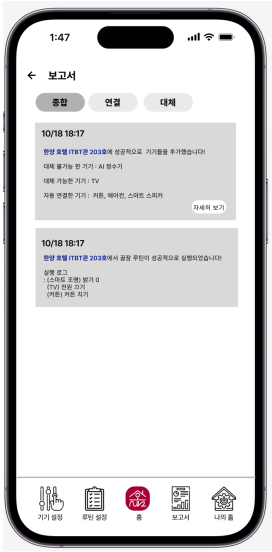

Fig. 118. Report page

Users can navigate to the 'Reports' page by clicking on 'Reports' in the menu bar. On the Reports page, users can receive reports on connecting devices through QR codes or the results of operating devices and executing routines. Detailed information such as 'Upgrade' or 'Replacement Completed' allows users to thoroughly review the outcomes.

I. My page

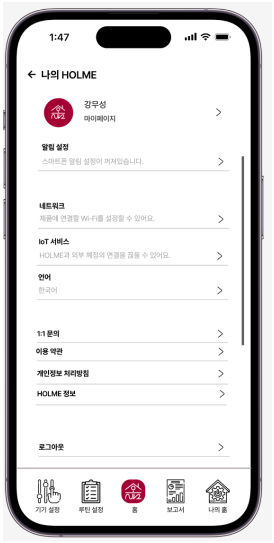

Fig. 119. My HOLME page

Users can access the 'My HOLME' page by clicking on 'My HOLME' in the menu bar. On the My HOLME page, users can navigate to 'My Page' to modify their profile, view account information, and access additional details. Additionally, users can perform functions such as changing the language, and logging out.

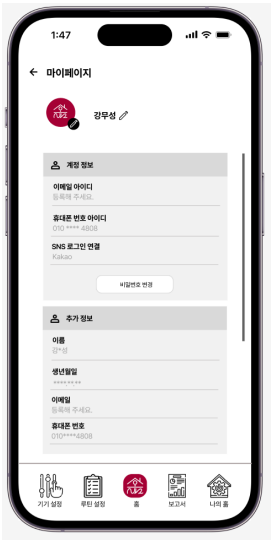

Fig. 120. My page

1) My Page: Users can click on 'My Page' under 'My HOLME' to edit their profile and review account information.

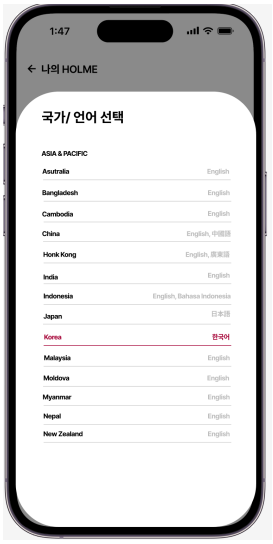

Fig. 121. Language Change page

2) Language Change: Users can change the language of the HOLME application by clicking on the 'Language' button under 'My HOLME.'

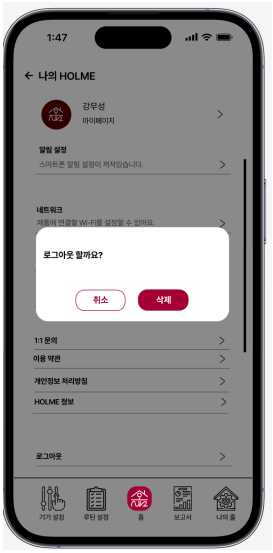

Fig. 122. Logout pop-up

3) Logout: Users can log out of the HOLME application by clicking on the 'Logout' button under 'My HOLME.'

REFERENCES

[1] “Typescript,” https://www.typescriptlang.org/, 2023.
[2] “Kotlin,” https://kotlinlang.org/, 2023.
[3] “GO lang,” https://go.dev/, 2023.
[4] “React Native,” https://reactnative.dev/, 2023.
[5] “SpringBoot,” https://spring.io/projects/spring-boot, 2023.
[6] “Hivernate,” https://hibernate.org/, 2023.
[7] “gRPC,” https://grpc.io/, 2023.
[8] “Flask,” https://flask.palletsprojects.com/en/3.0.x/ 2023.
[9] CSA. ”Matter Device Library Specification.” CSA, 2023, 27 28 pages
[10] CSA. ”Matter Device Library Specification.” CSA, 2023, 69 72 pages
[11] CSA. ”Matter Device Library Specification.” CSA, 2023, 69 72 pages
[12] CSA. ”Matter Device Library Specification.” CSA, 2023, 75 89 pages
[13] CSA. ”Matter Device Library Specification.” CSA, 2023, 84 85 pages
[14] CSA. ”Matter Device Library Specification.” CSA, 2023, 91 pages
[15] CSA. ”Matter Device Library Specification.” CSA, 2023, 66 68 pages