LG Follow

Mingyu Jung
Dept. of Information Systems
Hanyang Univ.
Seoul, Republic of Korea
alsrb595@hanyang.ac.kr

Taegeon Park
Dept. of Information Systems
Hanyang Univ.
Seoul, Republic of Korea
qkrxorjs@hanyang.ac.kr

Gyudong Kim
Dept. of Information Systems
Hanyang Univ.
Seoul, Republic of Korea
gyudong1594@hanyang.ac.kr

Mingeun Kim
Dept. of Information Systems
Hanyang Univ.
Seoul, Republic of Korea
alsrms0206@hanyang.ac.kr

***Abstract*—Imagine an office worker getting ready for work in the morning, listening to music or the news through an speaker. During the morning routine, they might wash up in the bathroom, make coffee in the kitchen, have breakfast, choose clothes from the closet, and get dressed. For someone who moves between rooms so frequently, it's almost impossible to catch 100% of the audio output from a stationary speaker.**

We are introducing technology that called LG Follow allows sound to follow the user, creating an environment where they can hear audio in any part of the house with LG appliances equipped with speakers.

Using a Raspberry Pi and PIR sensors, the system detects the user's location and seamlessly transfers the sound to the area where the user is currently located. For instance, if they leave the living room and enter the bedroom, the speaker in the living room will stop, and the speaker in the bedroom will automatically take over, seamlessly continuing the audio experience. MQTT protocol manages the communication between Raspberry Pi and the speaker.

Additionally, we provide an app called Sound Sketch that turns children's drawings into songs using generative AI. When a child draws a picture, the Sound Sketch will create a song based on their own picture. Through a trained AI model, the drawings will be transformed into prompts, and those prompts will be turned into music. With LG Follow, kids can enjoy listening to their own music as they move around the house, making each moment truly unique and magical.

TABLE I: : Role Assignments

Roles	**Name**	**Task description and etc.**
Backend Developer	Mingyu Jung	The backend developer would implement the logic to process location data from the Raspberry Pi and PIR sensors, ensuring the user's movement through the house is accurately tracked and relayed to the speaker. Ensure that when the user moves between rooms, the appropriate speaker is activated, and the previous one is turned off, all in real-time. Handle communication between the Raspberry Pi, PIR sensors, and speaker using the MQTT protocol, ensuring compatibility between the smart home appliances and devices.
Frontend Developer	Taegeon Park	The role of a frontend developer includes designing the frontend architecture and creating user interfaces and experiences using Figma. It involves collaborating with the backend team to discuss and implement required features while ensuring seamless integration with the server. The responsibilities also extend to reviewing designs and functionalities, keeping the user and consumer in focus to deliver an optimal user experience.
Backend Developer	Gyudong Kim	AI developer utilizes openai models BLIP to generate prompts from input images. To ensure the safety of children, AI developers fine-tune the BLIP model so that inappropriate prompts are not created. Once the prompts are generated, AI developers use the Suno API to create songs from these prompts. Also AI developers are responsible for designing and building databases to store and manage various types of data, such as user information, home appliances, and the generated drawings and songs, ensuring efficient integration and management within the application. Implement a database to store user preferences, drawings, and song files generated by Sound Sketch.
Frontend Developer	Mingeun Kim	The frontend developer would design the interface for the LG Follow app, ensuring that both LG Follow and Sound Sketch are intuitive and user friendly. This includes creating buttons and layouts for turning speakers on/off and uploading drawings to generate songs. The developer will focus on helping LG Follow and Sound Sketch features, enabling users to control speakers in different rooms and upload or manage children's drawings in the app.

I. Introduction

A. Motivation

1) *We wanted to create an experience where the sound follows the user, ensuring uninterrupted audio no matter where they are in the house. With LG appliances equipped with speakers, the user can now enjoy continuous sound as they move from room to room, eliminating the frustration of missing important parts of the music or news. Our motivation goes beyond just convenience, it's about creating an immersive and seamless audio experience that adapts to the user's movement. By integrating AI and smart home appliances, we aim to make daily life smoother and more enjoyable.*

2) *Our motivation for making Sound Sketch from a desire to bring children's creativity to life in a magical way and to strengthen the bond between parents and their children. Children often express their imagination through drawing, and we wanted to elevate that experience by transforming their artwork into personalized music. With the power of generative AI, a simple drawing becomes a unique, original song, giving children a new way to connect with their creations. What makes this experience even more special is the opportunity for parents and children to collaborate. By drawing together and hearing their joint artwork turned into a song, they can build memories and strengthen their connection through a shared creative process. With LG Follow, these songs can accompany them throughout the house, creating a sense of togetherness and joy, wherever they go. Our goal is to seamlessly blend creativity and technology, offering families a fun and interactive way to bond while engaging with music that feels personal and meaningful. Every moment becomes truly unique and magical as they hear their imagination come to life.*

3) *We've combined the LG Follow and Sound Sketch features into a single app, allowing users to experience both functionalities seamlessly in one place. With LG Follow, users can easily turn off the speaker in any appliance if they prefer not to have sound in a specific room. Meanwhile, Sound Sketch transforms children's drawings into songs, bringing their creativity to life in a fun and engaging way. The app ensures everything is accessible and simple to control, offering a smooth, personalized, and user-friendly experience.*

B. Problem Statement (client's needs)

In today's homes, speakers are commonly used to play music or news, but they are limited by their stationary nature. For someone who moves frequently between rooms during their morning routine—such as washing up in the bathroom, making coffee in the kitchen, or getting dressed in the bedroom—it becomes nearly impossible to catch all the audio from a single fixed speaker. This leads to a frustrating, interrupted experience where important parts of the audio are missed.

LG Follow solves this problem by allowing sound to follow the user throughout the house. As users move from room to room, LG appliances equipped with speakers provide continuous audio, eliminating gaps and ensuring they never miss a moment of music or news. This technology is designed to deliver a seamless and immersive audio experience, adapting to the user's movements to enhance their daily routine.

Additionally, we identified a need for families to engage creatively, leading to the development of Sound Sketch, an app that transforms children's drawings into personalized songs using generative AI. Children often express their imagination through drawings, and this feature elevates that creativity by turning their artwork into unique songs. It also offers parents and children the opportunity to bond through collaboration, creating memories as they hear their joint artwork come to life as music. By integrating LG Follow, these songs can follow the family throughout the house, adding a layer of joy and togetherness.

Both features, LG Follow and Sound Sketch, are integrated into the single app for ease of use, allowing users to control audio and manage drawings in one convenient place.

C. Research on Any Related Software

1) *Contrastive Language-Image Pre-training (CLIP): CLIP, developed by OpenAI, is a model trained on a large dataset of text and images to understand and associate visual and textual inputs. It enables users to perform tasks such as image search or classification using natural language prompts without task-specific fine-tuning, utilizing zero-shot capabilities. CLIP works by encoding images and text into a shared high-dimensional space where related pairs are closer together and unrelated pairs are further apart through a contrastive learning approach. While it demonstrates strong generalization capabilities, CLIP can exhibit biases present in its training data and may face challenges with nuanced or context-specific tasks.*

2) *Sonos (Multi-Room Speaker System): Sonos is a leading brand in multi-room audio systems. Sonos allows users to control audio in various rooms through a smartphone app, letting users sync music in different areas of the house. However, unlike LG Follow, Sonos requires manual control for changing rooms or selecting where to play audio. It does not automatically follow the user based on their movement.*

3) *NVIDIA Fugatto: NVIDIA Fugatto is an AI technology that generates audio from text prompts. By inputting a text description, Fugatto can create audio content that matches the provided prompt. For example, if you input "calm and serene morning," Fugatto will generate soft and peaceful music or nature sounds. Fugatto focuses on converting text into music, creating audio that matches the emotional tone or atmosphere described in the text.*

4) *Google Nest and Amazon Echo: Google Nest and Amazon Echo provide smart home automation, including voice-activated music playback. They can control music in various rooms and offer smart integrations with other appliances. However, users need to manually control playback across different speakers, and the sound doesn't seamlessly follow the user.*

5) *Melobytes: Melobytes is one tool where users can upload an image, and the system generates music from it using algorithms tailored to the visual data. It transforms the picture into unique sound compositions that reflect the visual input.*

6) *Img2Prompt (Anakin.ai): This tool uses AI to analyze an image and generate a text prompt that encapsulates its key visual features. The generated prompt can then be used for various creative projects, like digital art or content creation.*

7) *GoEnhance AI: This platform allows users to upload an image, and its AI algorithms automatically generate a text prompt based on the visual content. These prompts can be used with tools like DALL-E or Midjourney to generate new AI-created images based on the original photo's characteristics.*

II. Requirements

A. Log In

1) *Initial Screen: When the app launches, the initial screen displayed to users is the login screen, providing access for existing users. This screen serves as the primary interface for user access, where users can input their credentials (ID and password) to authenticate.*

2) *User Input: The user is required to enter their credentials, including:*

- ID Entry: The user must enter a unique identifier associated with their account. This ID serves as the primary authentication element linked to their account access.

- Password Entry: The password is a private passphrase known only to the user, ensuring account protection. The password field masks the entered characters to prevent visibility, and only the user is aware of this passphrase.

- ID and Password Format Validation: When the user enters the ID and password, basic format validation is conducted. For instance, if the ID is in email format, it verifies the presence of required elements like '@' and '.com'. The password field may also enforce minimum length and include special characters or numbers to strengthen security.

3) *Validation and Authentication*

- Password Hashing: Once the password is entered, it is hashed using the SHA-256 hashing algorithm upon reaching the server. This process converts the entered password into a unique hash value, ensuring that the original plaintext password is neither transmitted nor stored, thus enhancing security. SHA-256 provides a secure hash that is difficult to reverse-engineer, protecting the password from unauthorized access even in the event of data leaks.

- ID Verification: After hashing, the system checks if the entered ID exists in the database.
  - If the ID exists: The system compares the hash of the entered password with the stored hash in the database.
    * If the hashes match: A 'Login Successful' message is displayed, indicating successful authentication and granting access to the user, who is then directed to the home or main interface.

    * If the hashes do not match: An 'Incorrect password' message is displayed, prompting the user to re-enter their password. This message helps prevent unauthorized access by notifying the user of a password mismatch and requiring a new attempt.

  - If the ID does not exist: The system displays a message such as 'The entered ID does not exist,' informing the user of an incorrect entry. This allows the user to correct their entry and retry the login process.

B. Fine-tuning the BLIP Model

1) *Dataset Collection*

- Utilize the Flickr30k dataset available on Hugging Face, which includes images paired with their corresponding captions. Download the dataset using the Hugging Face Datasets library and prepare it for training preprocessing.

2) *Data Processing*

- Preprocess the dataset to ensure it works smoothly with the BLIP(Bootstrapped Language-Image Pretraining) model. Convert all inputs into a format the model can understand. Adjust text data so all sequences are the same length and create markers to identify real data versus padding. Standardize the image data format and organize captions into a list to prepare for training in batches.

3) Training Process

- Initialize the BLIP model and processor. Use a DataLoader to batch the preprocessed data and train the model. Compute and backpropagate the loss for each batch, updating the model parameters. Save the trained model and processor after completing the specified number of epochs.

C. Prompt Generation from Image

1) User Input

- After successful login, users can provide images through the app. These images can either be photos from their gallery or drawings created by the users.

2) Image Storage

- The images are stored in Amazon S3, and a URL is retrieved. The URL is securely stored in the RDS along with information about the user who uploaded the image.

3) Generating Prompt

- The backend server sends a request to the AI model's server to generate a prompt using the trained BLIP model. The generated prompt is stored in the RDS along with information about the image.

D. Music Generation from Prompt

1) Prompt Transfer

- The backend server sends the stored prompt to the Suno API, requesting the generation of basic music.

2) Generating Music

- Suno API generates music using the provided prompt. It sends information about the generated music to the backend server. This information includes the music URL, playback duration, and creation date.

3) Song Storage

- The backend server stores the received information in the RDS along with the prompt that was used to generate the music.

4) View Music

- When a user requests to view music, backend server provides information about the music. This includes the music URL, the URL of the image that was used to generate the prompt, and the playback duration.

5) Playing Music

- Users can play the generated music in real-time using the app. The duration of the music track is displayed in seconds.

E. Speaker-to-Speaker Connection

- System Configuration
  - The Raspberry Pi plays a central role in controlling music playback on speakers by utilizing a motion detection sensor. The Spring Boot server acts as the central control hub, collecting motion detection information and managing music playback commands for the speakers.

  - The Spring Boot server sends audio playback commands to the Raspberry Pi, which uses VLC Media Player to play music through the speakers. During this process, the server transmits data including the playback URL, starting point (currentTime), and additional information to ensure synchronized audio playback across all speakers.

- Expanding Device Compatibility
  - The Raspberry Pi is designed to synchronize audio playback across various types of speakers. When a user enters a specific room, the PIR sensor detects the motion and informs the Spring Boot server, which then sends a music playback command to the corresponding Raspberry Pi in that room.

  - All speakers receive the same music URL and playback starting point, allowing seamless audio playback even when the user moves between rooms. Music playback is automatically controlled based on the Raspberry Pi's detected actions. When the user leaves the room, the server is notified, and music playback is stopped.

F. User Location Tracking for Audio Control

- Location Data Collection via PIR Sensors
  - PIR sensors in each room detect user movement and transmit data to Raspberry Pi in real-time, allowing accurate location tracking.

  - Accurate Location Tracking: The PIR sensors provide accurate, room-level tracking, allowing the system to respond dynamically to the user's movements and adjust audio playback accordingly. This setup enables the audio to follow the user throughout different rooms, ensuring that the music or audio content is always accessible in the user's immediate vicinity.

- Location-Based Speaker Control
  - Automatic Speaker Activation and Deactivation: When a user moves from one room to another, the Raspberry Pi's PIR sensor detects the movement and sends a command to the central Spring Boot server. Based on this command, the server deactivates the speaker in the previous room and activates the speaker in the current room, continuing music playback from the same playback position. This ensures that music only plays in the room where the user is present, providing a seamless audio experience without interruptions.

- Speaker Control and Audio Transition
  - Real-Time Audio Switching Using the MQTT Protocol: The Raspberry Pi communicates motion detection events and playback control commands with the Spring Boot server using the MQTT protocol. When motion is detected, the Raspberry Pi in the active room adjusts the playback position and resumes music without restarting the process. This minimizes latency and allows smooth audio transitions between rooms.

  - Seamless Multi-Speaker Coordination: As the user moves between rooms, music playback seamlessly switches between speakers. Each room's Raspberry Pi synchronizes the playback position using VLC, ensuring consistent audio output. This adjustment allows music to dynamically follow the user, providing a continuous and immersive listening experience.

G. Location-Based Sound Transition

1) Real-time Streaming with MQTT Protocol

- MQTT Configuration for Each Room's Speaker
  - Each room's speaker is configured to receive audio data via the MQTT (Message Queuing Telemetry Transport) protocol. As the user moves between rooms, the Raspberry Pi utilizes MQTT to seamlessly switch audio playback to the appropriate speaker based on the user's location. The MQTT broker serves as the intermediary, ensuring smooth audio data transfer between speakers. Each speaker subscribes to specific topics, and as the user moves, the Raspberry Pi publishes the audio data to the relevant topic, enabling the transition of audio between rooms without interruption.

- Seamless Streaming
  - MQTT, being a lightweight message protocol, is optimized for real-time data transmission, allowing audio to be delivered rapidly and reliably. As the user transitions from one room to another, the MQTT broker ensures that the audio data is routed to the correct speaker based on the user's location. Using MQTT's QoS (Quality of Service) levels, the protocol guarantees that data is delivered reliably, ensuring a continuous streaming experience without interruption.

2) Audio Data Transfer

- Communication Management Using the MQTT Protocol
  - Raspberry Pi devices are directly connected to speakers, and all Raspberry Pis are configured to subscribe to specific topics on the MQTT broker. The Spring Boot server also subscribes to the same MQTT topics, enabling bidirectional communication between the server, Raspberry Pis, and speakers.

  - When user movement is detected, the Raspberry Pi sends an MQTT message, which is received by the Spring Boot server. The server then sends music playback information back to the relevant Raspberry Pi, instructing its connected speaker to play the music. This setup enables real-time communication between devices, ensuring efficient and synchronized audio data transfer.

- Enhanced Audio Experience
  - With MQTT's QoS features, audio data is transmitted reliably and without loss. The MQTT broker plays a central role in managing and propagating messages between speakers. As the user moves, the broker ensures that the audio stream is transitioned smoothly from one speaker to the next, delivering an uninterrupted listening experience.

H. Speaker On/Off Functionality

- User Preference Storage
  - Centralized User Preference Management: User preferences, such as enabling or disabling music in specific rooms, are stored on a central server and communicated to the Raspberry Pi via MQTT protocol.

- Speaker Control
  - Raspberry Pi uses the Matter protocol to control music playback according to user preferences, turning music on or off in designated rooms.

I. Music CRUD

- The Spring Boot server processes the request and ensures real-time streaming of the music, allowing users to listen to the song as soon as it's generated.

- For storage and long-term management of the generated music, the metadata and music links are handled by

Amazon RDS, which communicates with a MySQL database to store information such as the song title, date of creation.

- Users can perform CRUD operations on the generated music.
  - Create: Saving the generated music to the Amazon RDS
  - Read: Streaming the saved music from RDS and retrieving metadata from the database for display in the app.
  - Update: Modifying the song's metadata or regenerating the music by submitting a new drawing.
  - Delete: Removing the music file from RDS and its metadata from the MySQL database.

III. Development Environment

A. Hardware Development Environment

1) Raspberry Pi

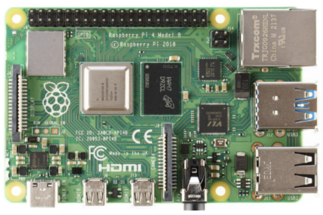


Fig. 1: Rasberry Pi

The Raspberry Pi is a small computer commonly used in various programming and IoT projects. Its efficient and cost-effective design makes it popular for home automation and IoT environments.

Role:

- IoT Device Control: The Raspberry Pi serves as the main device for controlling IoT devices in this project. It connects with multiple sensors and actuators (e.g., PIR sensor, temperature sensor), collects data, processes it, and sends it to the system.
- Using MQTT Protocol: The Raspberry Pi communicates with other devices or servers through the MQTT protocol, allowing real-time monitoring and control of IoT device states.
- Serving as a Matter Bridge: The Raspberry Pi is configured as a Matter bridge to facilitate communication with devices compatible with the Matter Protocol. This setup enables it to communicate with various Matter-compliant IoT devices, allowing centralized control and monitoring of each device via the Spring Boot server. The Matter bridge acts as a communication hub for IoT devices, ensuring interoperability among devices within the same network.
- Network Management and Processing: Through network connectivity, the Raspberry Pi connects to the Spring Boot server or the Matter Protocol to transmit and receive data and to process various commands.

Technical Features:

- Compact Size and Low Power Consumption: Ideal for environments with space and power constraints.
- Expandability: The GPIO pins allow easy connection to various sensors and actuators, providing flexibility for various IoT projects.
- Matter Bridge Integration: Acting as a Matter bridge, the Raspberry Pi operates as an integration point for interacting with Matter-supported devices. This enables other Matter-compatible devices to connect through the Raspberry Pi, ensuring smooth communication.
- Supports Various Languages: Raspberry Pi can be controlled using multiple programming languages such as Python, C++, and Java, offering developers a high degree of flexibility.

2) Matter Protocol

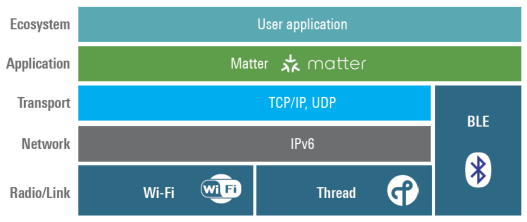


Fig. 2: Matter

The Matter Protocol is a communication protocol developed to enhance interoperability among smart home and IoT devices. Supported by major IoT device manufacturers, Matter ensures compatibility between devices from different manufacturers within a smart home ecosystem.

Role:

- Provides Interoperability among IoT Devices: With the Matter Protocol, various IoT devices within the same network can communicate seamlessly. This enables the devices used in the project to interact smoothly with one another.
- Standardized Data Transmission: The Matter Protocol offers a standardized method for data transmission, simplifying data management and transfer for developers.

- Enhanced Security Features: Matter strengthens security by employing encryption for data transmission between devices and an authentication system that only allows approved devices to connect.

Technical Features:

- Multi-Platform Support: Matter supports various network technologies such as Wi-Fi, Ethernet, Thread, and BLE, allowing flexible configurations suited to the environment.

- Simplified Development Process: Matter streamlines communication protocols, making integration easier for developers and ensuring compatibility across a wide range of IoT devices.
- High-Performance Security: It provides security based on authentication, ensuring that data transmission between IoT devices is securely protected.

3) PIR Sensor

Fig. 3: PIR Sensor

A PIR (Passive Infrared) sensor is used to detect motion and is commonly employed in security and home automation systems. It senses infrared signals emitted by the human body to determine motion and can monitor user presence within a specific area.

Role:

- User Detection and Music Control: The PIR sensor detects users within a room and triggers actions like playing or stopping music when certain conditions are met. When a user is detected, it sends a signal to the Spring Boot server to turn on the music. If the user leaves the detection range or after a certain period, it can be configured to stop the music automatically.

- Energy Efficiency Maintenance: The PIR sensor helps in reducing unnecessary power consumption by detecting activity in a particular area. When no user is detected, it automatically turns off devices like music or lighting to save energy.

- Real-time Status Transmission: It sends real-time movement data to the Spring Boot server, which can then use this information to control other IoT devices based on user location.

Technical Features:

- Low Power Consumption: The PIR sensor consumes minimal power as it only sends a signal when the detection state changes.

- Easy Installation and Connection: It can be easily connected to the GPIO pins of control devices like Raspberry Pi and can be handled with simple signal processing.

- Instant Response Time: The PIR sensor is suitable for applications that require immediate feedback as it quickly detects human movement.

- Wide Detection Range: It generally provides a broad detection angle, allowing it to sense movement throughout an entire room.

B. Software Develepment Platform

1) Linux

Fig. 4: Linux

Linux, an operating system based on UNIX, is known for supporting multiple users, multi-tasking, and multi-threading, making it highly suitable for development environments. As an open-source OS, it has been widely distributed and modified for various needs, with popular distributions like Ubuntu. Linux excels in server environments, desktop applications, and embedded systems development due to its flexibility, performance, and robust security features. Its extensive command-line utilities make it particularly adept for working with development boards and IoT devices, allowing seamless integration with hardware. Additionally, Linux's strong community support ensures quick troubleshooting and continuous enhancements, making it an excellent choice for IoT development and large-scale software projects.

Fig. 5: Windows

2) *Windows*

Windows, developed by Microsoft, is one of the most widely used operating systems globally, known for its compatibility with a vast range of software applications and hardware devices. Its graphical user interface and accessibility make it versatile for personal, business, and development use. Windows supports multi-tasking and multi-threading, which enhances its capability for running complex applications and handling high-demand development tasks. With tools like Visual Studio, Windows provides comprehensive support for app development across multiple platforms, including desktop, web, and mobile applications. Windows Subsystem for Linux (WSL) allows developers to use Linux command-line tools and utilities directly on Windows, making it an adaptable choice for cross-platform development. Furthermore, Windows offers strong support for gaming and multimedia applications due to DirectX and hardware acceleration capabilities. Its enterprise-level security options, combined with Microsoft's extensive documentation and community support, make Windows a reliable operating system for both general-purpose computing and advanced development environments.

3) *macOS*

Fig. 6: macOS

macOS, developed by Apple and based on UNIX, is widely recognized for its user-friendly interface and stability, making it a preferred operating system for developers, particularly in design, media, and software development environments. Its robust integration with Apple hardware ensures optimized performance and energy efficiency, which is essential for seamless application testing and development. macOS includes powerful command-line utilities and developer tools such as Xcode, which supports app development for the Apple ecosystem, including iOS, macOS, watchOS, and tvOS. Additionally, the macOS environment provides strong support for multi-tasking and multi-threading, and its UNIX-based architecture offers compatibility with various development frameworks and programming languages, making it highly adaptable to web, mobile, and cloud applications. macOS's extensive developer community, combined with Apple's regular updates, ensures high security standards, efficient troubleshooting, and consistent performance, making it an ideal choice for both individual developers and large-scale development teams.

4) *MQTT Protocol*

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed primarily for IoT (Internet of Things) and M2M (Machine to Machine) communication. It is optimized for environments with limited bandwidth and unreliable networks, making it ideal for real-time data transmission between small devices and servers.

C. Language & Framework

1) *Spring Boot*

Fig. 7: Spring Boot

Spring Boot is a lightweight Java-based backend framework that simplifies the process of quickly developing standalone web applications. It reduces the complexity of configuring the Spring Framework by including embedded web servers, such as Tomcat or Jetty, enabling the application to run with minimal additional setup in both development and production environments.

Role:

- Web Application Development: Facilitates the rapid development of standalone web applications.

- Auto-configuration: Provides automatic configuration for various Spring features without needing XML or Java Config classes.

- Dependency Management: Simplifies dependency configuration with Spring Boot Starter, boosting development efficiency.

- Security and Authentication: Easily integrates with Spring Security, allowing for streamlined implementation of authentication and authorization.

- Asynchronous Processing and Scheduling: Ideal for IoT, mobile, and cloud applications, enabling easy development of REST APIs, asynchronous processing, and scheduling functionalities.

Technical Features:

- Embedded Server: Bundles web servers such as Tomcat, Jetty, and Undertow, eliminating the need for separate configurations.
- Automatic Configuration: Automatically sets up necessary features, streamlining application setup.
- Support for Various Environments: Easily adaptable from local development to cloud deployment.
- MQTT Support: Enables MQTT messaging with libraries like Eclipse Paho MQTT Client, allowing Spring Boot applications to publish and subscribe to MQTT messages.

2) *Java*

Fig. 8: Java

Java is an object-oriented programming language known for its platform independence, allowing applications to run consistently across different operating systems. With the motto Write Once, Run Anywhere, Java achieves cross-platform compatibility through the JVM (Java Virtual Machine), enabling stable and efficient application development.

Role:

- Server-side Logic Development: Java's reliability makes it widely used for server applications.
- Multithreading and Asynchronous Processing: Java's multithreading and asynchronous processing capabilities are ideal for IoT systems that require real-time data handling.
- Large-scale System Operation: Thread management, garbage collection, and robust libraries enable Java to handle high-traffic server applications effectively.

Technical Features:

- Object-Oriented Programming: High reusability and scalability, making maintenance straightforward.
- Platform Independence with JVM: Allows consistent code execution across various platforms via the JVM.
- Extensive Standard Libraries: Provides libraries for networking, databases, file handling, and more, supporting efficient application development.
- Reliability and Performance: Ensures stability in large-scale systems and guarantees performance in multithreaded environments.

3) *Python*

Fig. 9: Python

Python is a widely used interpreted language known for its simplicity, readability, and versatility across various applications and platforms. Its ease of use and concise syntax significantly reduce development time and minimize errors, making it a popular choice for developers.

Role:

- Cross-Platform Compatibility: Python is a cross-platform language that can run on various operating systems (Linux, Windows, Mac) with minimal modifications, providing flexibility and efficient deployment.
- Data Processing and Manipulation: Its strengths in string processing and data manipulation make Python ideal for applications involving text and image processing, data analysis, and scientific computing.
- Automation and Scripting: Python's readability and flexibility make it suitable for automation tasks, enabling the development of scripts and tools for repetitive tasks and efficient workflows.

Technical Features:

- Concise and Readable Syntax: Python's straightforward syntax enhances code readability and reduces development time, contributing to rapid prototyping and easier debugging.
- Extensive Libraries: Python offers a rich ecosystem of libraries, including Pandas, NumPy, and OpenCV, which simplify complex operations such as data analysis, machine learning, and image processing.
- Community and Documentation: Python boasts a large and active community, providing a wealth of resources,

support, and libraries that streamline development.

4) Flask

Fig. 10: Flask

Fig. 11: Flutter

Flask is a lightweight web framework for Python, known for its simplicity, flexibility, and suitability for small to medium-sized web applications and APIs. Flask provides a straightforward foundation for building web applications with minimal setup.

Role:

- RESTful API Development: Flask is optimized for creating RESTful APIs, providing tools for handling HTTP requests and responses, which is ideal for applications requiring data exchange in formats like JSON.
- Integration with Python Models: Flask can easily integrate with Python-based machine learning models, making it simple to serve model predictions and data processing results to client applications.
- Customizable and Modular: Flask allows developers to include only the components needed for a project, creating a lean and efficient development environment.

Technical Features:

- Lightweight and Flexible: Designed to be minimalistic, Flask allows developers to build web applications without unnecessary features, ensuring a flexible and tailored environment.
- Powerful Routing and Request Handling: Flask's built-in support for routing and handling HTTP methods makes it suitable for building APIs and web services.
- Rich Ecosystem of Extensions: Flask supports numerous extensions, allowing for easy integration of features like authentication, database connections, and caching.

5) Flutter

Flutter is an open-source UI toolkit developed by Google, allowing developers to create natively compiled applications for mobile, web, and desktop from a single codebase. It is known for its expressive UI, fast development cycles, and ability to create visually appealing applications. By using Dart, a language optimized for building mobile applications, Flutter offers a seamless and efficient development experience with extensive community support and a rich library of widgets.

Role:

- Cross-platform Development: Flutter provides a seamless way to create applications that work consistently across multiple platforms, including iOS, Android, web, and desktop, from one codebase. This greatly simplifies development and reduces costs.
- Rich and Customizable UI Design: Flutter's widget-based framework allows developers to create highly customizable UIs with complex animations, ensuring a native-like experience and high performance for both simple and intricate designs.
- Component-based Architecture: Flutter's widget-centric architecture encourages modular code that is easy to reuse, manage, and scale. Developers can create complex UIs by composing and nesting widgets.
- Expressive Declarative UI: Flutter uses a declarative approach to building UIs, making the code more readable, intuitive, and responsive to changes in state.

Technical Features:

- Dart Language: Flutter leverages Dart, a powerful, easy-to-learn language optimized for mobile and desktop apps with strong performance and efficient memory management.
- Extensive Widget Library: Flutter offers a vast collection of built-in widgets that are fully customizable, enabling developers to create UIs that conform to platform-specific designs or have a unique, custom look.
- Hot Reloading: Allows developers to make changes to the code and see the results instantly without losing application state, enhancing productivity and speeding up development cycles.
- Native Integration and Flexibility: Flutter allows seamless integration with platform-specific APIs and services, offering flexibility to access native features and deliver

a native-like user experience.

- IoT-related System Development: Flutter is well-suited for use alongside the latest technologies. Consider using Flutter when developing IoT (Internet of Things) solutions. It enables efficient integration of smart devices with user interfaces, facilitating the creation of high-performance IoT systems.

D. *Software & AI model In Use*

1) BRIP

BLIP (Bootstrapping Language-Image Pre-training) is a cutting-edge AI model that enables unified vision-language understanding and generation. By training on a large dataset of image-text pairs, BLIP can perform diverse tasks such as image captioning, visual question answering, and text-guided image generation.

Role:

- Image Captioning: BLIP generates descriptive captions for images, providing a textual representation of visual content, which is useful for accessibility and content categorization.

- Visual Question Answering: BLIP interprets visual data to answer questions about images, offering real-time assistance and context comprehension for visual-based tasks.

- Text-Guided Image Generation: BLIP creates or modifies images based on textual prompts, bridging the gap between visual creativity and text-based control.

Technical Features:

- Zero-shot Learning: BLIP's zero-shot capabilities allow it to adapt to new tasks without additional training data, making it versatile across various applications.

- Specialized Language Training: By training on datasets with images and Korean captions, BLIP can generate Korean prompts, making it suitable for localized applications and non-English language tasks.

- Enhanced Vision-Language Bridging: BLIP effectively integrates textual and visual data, enabling it to process and produce outputs that combine both modalities seamlessly.

2) Suno API

Suno API is an innovative AI model that enables the automatic generation of songs based on text prompts. It integrates natural language processing and music generation technologies to transform text into fully realized songs, including lyrics, melodies, and vocals. By interpreting the text prompt, Suno API generates custom songs that align with the user's specific requirements in terms of genre, emotion, and style.

Suno

Fig. 12: Suno API

Role:

- Song Lyrics Generation: Based on a text prompt, Suno API generates song lyrics automatically. For example, users can provide a theme or emotion, and the system will create relevant lyrics for a song.

- Music Composition: Suno API can generate accompanying music for the lyrics by interpreting the text prompt. It produces melodies, chords, and harmonies that align with the requested genre or mood.

- Vocal Synthesis: Once the song's lyrics and music are generated, Suno API synthesizes vocals, delivering high-quality, human-like singing. The vocals are crafted to match the mood, tone, and style defined in the prompt.

Technical Features:

- Advanced Text-to-Music Translation: Suno API combines natural language processing (NLP) and music generation technologies to convert text-based prompts into full-fledged songs, including lyrics, melodies, and vocals.

- Emotion and Mood Reflection: The system can analyze the emotion conveyed in the prompt (e.g., sadness, joy, love) and generate music that reflects this emotion, adjusting the melody, tempo, and vocal tone accordingly.

- Zero-shot Learning: Suno API leverages zero-shot learning capabilities to generate songs in various genres and styles without requiring additional training data. It can adapt to new prompts and create custom songs immediately.

3) Amazon S3

Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service known for its industry-leading data availability, security, and performance. Amazon S3 provides a robust infrastructure for storing and managing large volumes of data, particularly useful for media files such as images and music.

Role:

Fig. 13: Amazon S3

- File Storage: Stores generated music files and uploaded image files in S3, with Flask generating music files that the Spring Boot server saves to S3, providing clients with URLs for file download or streaming.

- Cloud Storage: Manages large data volumes efficiently, including user-related images and audio files.

- Data Management: Used for organizing temporary or outdated data, as well as for backups. Access permissions can be configured to restrict file access to specific users.

Technical Features:
- Scalability: S3 automatically scales to accommodate large volumes of data as file counts increase.

- Reliability: Offers high durability, ensuring data is securely stored.

- Flexible Access Management: Access Control Lists (ACLs) and bucket policies allow for detailed permission settings.

4) *MySQL*

Fig. 14: MySQL

MySQL is a widely used open-source relational database management system (RDBMS) known for its stability, performance, and ability to handle large datasets efficiently. Its robust functionality makes it ideal for managing data-intensive applications, including those that track user locations, manage media files, and facilitate device interactions.

Role:
- Data Storage: Stores and manages user information, generated music metadata (title, creation date, etc.), and music request data. This enables the system to store a wide variety of data permanently for retrieval as needed.

- Data Integrity Assurance: Ensures data consistency and integrity through transactions and foreign keys, maintaining stability even when multiple users access data simultaneously.

- CRUD Operations: Supports Create, Read, Update, and Delete operations through integration with the Spring Boot application.

Technical Features:
- SQL-Based: Uses standard SQL queries, making data management straightforward.

- Scalability: Efficiently manages large transactions and data with scalable architecture.

- Reliability: Provides strong backup and recovery capabilities to ensure data security.

5) *Amazon RDS (Relational Database Service)*

Fig. 15: Amazon RDS

Amazon RDS is AWS's managed relational database service, handling database server management and maintenance, allowing developers to focus on data-related tasks rather than setup and maintenance.

Role:
- MySQL Server Hosting: Supports MySQL and other database engines, making it easy to set up and manage MySQL servers. In this project, RDS manages user information, music metadata, and music requests.

- Automated Management: Handles database backups, security patches, and software updates automatically, reducing database management burdens.

- Scalability: Supports scale-up/down as traffic or storage needs change. For high read performance, Read Replicas can be added.

Specific Functions:

- Persistent Data Storage: The RDS MySQL server stores user-related data, music metadata, and music requests, ensuring reliable data management even with high request volumes.

- Data Backup and Recovery: Performs regular backups automatically, with automatic recovery capabilities to ensure data integrity and reliability.

- High Availability: Multi-AZ deployment replicates databases across availability zones, providing high availability and fast recovery in case of failures.

Technical Features:

- Automatic Backups: Regular backups enable point-in-time recovery.

- Security: Supports VPC isolation, AWS IAM integration, and data encryption via AWS Key Management Service (KMS).

- Monitoring and Alerts: Integrates with Amazon CloudWatch for real-time database performance monitoring and notifications for any issues.

Integration with RDS:

- Spring Boot: The Spring Boot project connects directly with AWS RDS MySQL, leveraging Spring Data JPA for efficient database interactions. Database URL, username, and password are configured using the RDS endpoint.

- RDS MySQL Setup: After creating a MySQL instance in the RDS console, its endpoint and credentials are used to connect the Spring Boot application.

6) *Visual Studio Code*

Fig. 16: VS Code

Visual Studio Code (VS Code) is a lightweight and highly customizable code editor developed by Microsoft. Known for its versatility and extensive extension ecosystem, VS Code supports a wide range of programming languages and development environments, making it a popular choice for developers across various fields.

Role:

- Code Editing and Development: VS Code provides a streamlined, efficient environment for writing, editing, and managing code in multiple languages. It offers a powerful, customizable interface that supports development workflows from simple scripts to large projects.

- Extension Support: With thousands of extensions available, VS Code allows developers to add language support, frameworks, linters, debuggers, and tools specific to their project needs, enhancing productivity and flexibility.

- Remote Development: VS Code's remote development feature allows users to work on projects hosted on remote servers or in containers, enabling efficient collaboration and resource utilization.

Technical Features:

- Lightweight and Fast: Designed for speed and efficiency, VS Code is a lightweight editor that quickly adapts to various development environments without significant resource usage.

- Integrated Debugging and Git: VS Code includes built-in debugging and Git support, streamlining the development and version control process within a single interface.

- Highly Customizable: Through JSON configuration files, settings, and an extensive extension library, VS Code allows for deep customization to meet individual development needs.

7) *Android Studio*

Fig. 17: Android Studio

Android Studio is Google's official integrated development environment for Android app development, offering a comprehensive suite of tools and resources tailored specifically for Android platforms.

Role:

- Emulator for Testing: It provides a fast and feature-rich emulator, simulating a wide range of Android devices and configurations for thorough testing without needing physical devices.

- Extensive Testing Tools: Android Studio includes a variety of testing tools and frameworks that support automated and manual testing, enhancing app quality and stability.

- Easy SDK Access: The IDE includes Android SDK and essential tools out-of-the-box, allowing developers to start building Android applications immediately.

Technical Features:

- Integrated Emulator: Fast, powerful emulator to test app functionality across device configurations.
- Comprehensive Testing Support: Tools for unit, UI, and integration testing.
- Code Inspection and Lint: Automates code quality checks and flags potential issues.

8) *Xcode*

Fig. 18: Xcode

Xcode is Apple's official IDE for developing applications on iOS, macOS, watchOS, and tvOS, offering a complete suite of tools and resources to streamline development for Apple's platforms.

Role:

- Apple Platform Development: Xcode is essential for creating applications for Apple's ecosystem, providing all necessary tools to develop iOS and macOS applications quickly and efficiently.
- Built-In Simulator: The simulator allows for testing applications without needing a physical device, supporting various iOS versions and device types.
- Storyboard and App Structure Visualization: The storyboard feature enables developers to see the structure of the app at a glance, aiding in the systematic design of complex applications.

Technical Features:

- Drag-and-Drop Interface Builder: Simplifies UI design with visual placement and Auto Layout.
- Comprehensive Preview Options: Preview and adjust for multiple devices and screen sizes.

9) *Figma*

Figma is a cloud-based design and prototyping tool used for creating interactive and collaborative design experiences.

Fig. 19: Figma

Known for its flexibility and real-time collaboration features, Figma allows designers and teams to work together seamlessly from anywhere.

Role:

- UI/UX Design: Figma provides a robust platform for designing user interfaces and experiences, with tools to create detailed wireframes, mockups, and high-fidelity designs.
- Prototyping: Enables the creation of interactive prototypes that allow designers to simulate user flows and test interactions, providing a clear vision of the final product.
- Collaboration: Real-time collaboration allows multiple team members to work simultaneously on the same file, streamlining the feedback process and reducing delays.

Technical Features:

- Developer-Friendly Inspect Tool: Developers can access design specifications, CSS code, and assets directly, ensuring a smooth design-to-code transition.
- Interactive Prototyping and Animation: Offers tools to create interactive animations and transitions, making it easy to visualize and test user interactions.
- Cloud-Based and Cross-Platform: Figma operates in the cloud, accessible from any browser and supporting both Windows and macOS, making it highly versatile and collaborative.

10) *Docker*

Fig. 20: Docker

Docker is an open-source platform that allows applications to be containerized, enabling consistent management of environments across development and production. It helps developers and operations teams build and deploy applications more efficiently. By running applications in isolated containers, Docker minimizes issues that can arise during the deployment process.

Role:

- Application Deployment: Docker reduces the gap between development and production environments, making it easy to deploy and run applications. This ensures applications can be consistently run across different environments.
- Environment Isolation: Docker runs each application in an isolated container, resolving dependency issues between systems. It ensures that multiple applications do not conflict with each other.
- CI/CD Integration: Docker is useful in continuous integration and deployment (CI/CD) environments, enabling fast and stable deployment of code changes.

Technical Features:

- Containerization: Docker packages the application and its environment into a single unit, allowing it to run consistently across different platforms.
- Lightweight: Docker is more lightweight than virtual machines, using fewer system resources and allowing for faster startup and shutdown.
- Networking and Storage: Docker makes it easy to manage networking and storage settings, facilitating smooth communication between containers.
- Scalability: Docker can run hundreds of containers simultaneously, making it easy to scale and manage large applications.

Integration with Docker:

- Spring Boot: Docker can containerize Spring Boot applications, automating deployment and minimizing the differences between environments. Spring Boot applications run within Docker containers, ensuring consistency between development and production environments.
- Flask: Flask applications can also be containerized using Docker. This enables consistent execution across different environments, improving efficiency in deployment and maintenance.

11) Apache Kafka

Apache Kafka is a high-performance, distributed streaming platform used to handle large-scale, real-time data streams.

Fig. 21: Apache Kafka

Kafka is widely used for messaging systems, event sourcing, log aggregation, and building data pipelines. Built on a distributed architecture, Kafka provides high scalability and durability.

Role:

- Data Streaming: Kafka is used to process real-time data streams. It efficiently transmits data from producers to consumers in real-time, handling large volumes of data flow.
- Event Sourcing: Kafka plays a crucial role in event-driven systems by sequentially recording and storing events, enabling subsequent processing of these events.
- Data Pipelines: Kafka provides a pipeline for managing data flow between different systems, allowing seamless integration between multiple systems exchanging data.
- Real-Time Analytics: Kafka is useful for real-time data analysis, enabling quick processing and extraction of insights from continuous data streams.

Technical Features:

- Distributed Architecture: Kafka is designed with a clustered setup, providing high availability and fault tolerance. Data is distributed across multiple servers, ensuring continuous operation even in case of failures.
- Scalability: Kafka can handle thousands of partitions, allowing it to scale efficiently even as data volume grows.
- Durability: Kafka ensures data durability by continuously storing data on disk, preventing data loss and allowing for recovery in case of failures.
- High Throughput: Kafka is known for its high throughput, capable of processing millions of messages per second.

Integration with Kafka:

- Spring Boot: Kafka integrates well with Spring Boot applications, enabling the building of asynchronous, message-driven systems. The Spring Kafka library simplifies interactions with Kafka within Spring Boot projects.

- Microservices: Kafka is ideal for service-to-service communication in microservices architectures. It allows services to communicate by sending and receiving events through Kafka, reducing tight coupling between services.

- Real-Time Processing: Kafka can be used in combination with real-time processing frameworks like Apache Spark or Apache Flink to enable real-time data analytics and processing.

12) Pytorch

Fig. 22: PyTorch

PyTorch is an open-source deep learning framework that provides a flexible and efficient platform for building and training machine learning models, widely used in both academic research and production environments.

Role:

- Deep Learning Framework: PyTorch is essential for developing machine learning and deep learning models, offering tools that support both research and production. It provides efficient and flexible mechanisms for building complex models, from neural networks to reinforcement learning agents.

- GPU Acceleration: With native support for CUDA, PyTorch accelerates training using GPUs, making it well-suited for working with large-scale datasets and deep neural networks.

Technical Features:

- Tensor Operations: PyTorch offers a powerful tensor library for multi-dimensional arrays and matrices, similar to NumPy, with GPU acceleration for efficient computations on large datasets.

- Pretrained Models and Ecosystem: PyTorch provides access to a wide range of pretrained models and related libraries, such as torchvision for computer vision, torchaudio for audio processing, and torchtext for NLP tasks, enabling rapid deployment of state-of-the-art solutions.

IV. Specification

A. Initial Screen

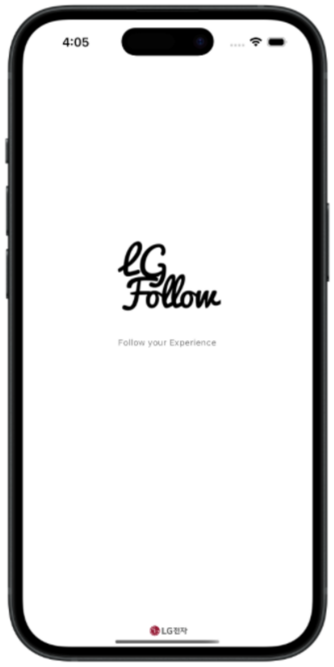


Fig. 23: Initial Screen

When the user launches the app, the initial screen displays the 'LG Follow' logo and the 'Follow your Experience' tagline for 2–3 seconds. After a brief delay, the screen transitions smoothly to the login page.

B. Log In

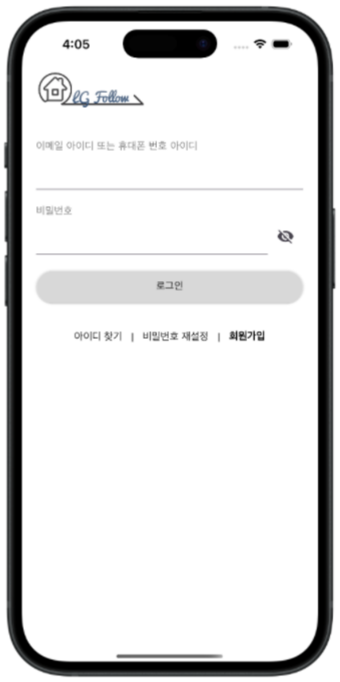

4:05
LG Follow
이메일 아이디 또는 휴대폰 번호 아이디
비밀번호
로그인
아이디 찾기 | 비밀번호 재설정 | 회원가입

Fig. 24: Log In

1) ID Input

- Explanation: 'Email ID or Phone number ID' to guide the user on the input type.
- Input Format: Accepts either an email address or phone number.
- Validation: Checks if the input follows the correct format
  - Email: Format like user@example.com
  - Phone number: Format like 01012345678

2) Password Input

- Explanation: Displays 'Password' as guidance for the user.
- Input Format: Accepts text for password entry, displayed as '•' symbols to conceal the input.

3) Log In

- Action: When clicked, the button attempts to log in with the provided ID and password.
  - On Success: Redirects to the main screen.
  - On Failure: Shows an error message ('The entered ID or password is incorrect').

4) Log In Process

- Input Check: When the log in button is clicked, the ID and password are validated against database records.
- Password Hashing: The entered password is hashed using the SHA-256 algorithm before being compared to the stored hash in the database.
- Authentication and Response
  - Success: On successful log in, the user is redirected to the main screen.
  - Failure: If log in fails, an error message ('The entered ID or password is incorrect') is displayed.

C. Main Page

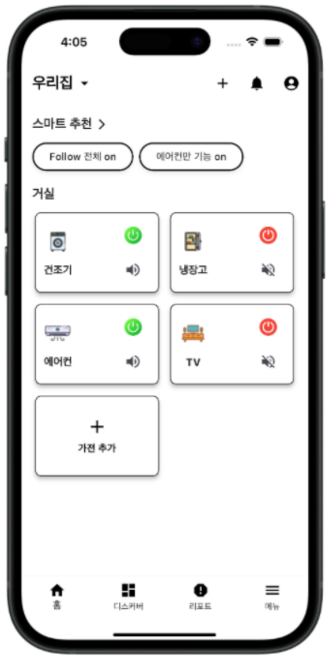


Fig. 25: Main Page

1) Home Application Power On/Off

- Power Button: Turns individual appliances On/Off. (In the On state, the power button appears green, and in the Off state, it appears as a red button.
- When the user clicks the power button for a specific appliance, the application records the device's status (On/Off) in the database or local storage.
- Upon status change, the application sends a command to the appliance to toggle its power state.
- If the device is currently on, clicking the button will turn it off, and if it is off, clicking the button will turn it on.

2) LG Follow Function On/Off

- Speaker Button: Controls the LG Follow function for audio playback in the room where the appliance is located.(urns individual appliances On/Off. (In the On state, the speaker button is displayed, and in the Off state, it appears on the speaker button with a strikethrough.)

- User Location Detection
  - PIR Sensors: PIR (Passive Infrared) sensors installed in each room detect the user's movement. When the user enters a room, the PIR sensor in that room activates and sends the location data to the Raspberry Pi.

- Data Collection and Transmission by Raspberry Pi
  - Raspberry Pi: Acts as a central hub that collects location data from all room PIR sensors. As the user moves from room to room, the Raspberry Pi updates the user's location in real time.
  - MQTT Protocol: Raspberry Pi communicates with other room speakers via the Matter protocol to control audio output based on the user's current location.

- Audio Transition
  - When the user enters a new room, the Raspberry Pi sends a command to stop audio playback in the previous room's speaker and switch the audio to the speaker in the current room.
  - This setup allows the user to hear audio naturally in each new room as they move throughout the house.

- User Control Options for LG Follow
  - LG Follow On/Off for Specific Devices: Users can enable or disable the Follow feature for specific appliances in particular rooms, allowing more customized control over where audio follows them.

3) Add Application Button ('+' Button)

- Opens a QR code scanning screen to add a new appliance.
- When the user clicks the appliance button, the application navigates to the QR code scanning screen.
- The user can scan a QR code to register a new appliance, which will then be added to the home screen for control.

D. Device Registration

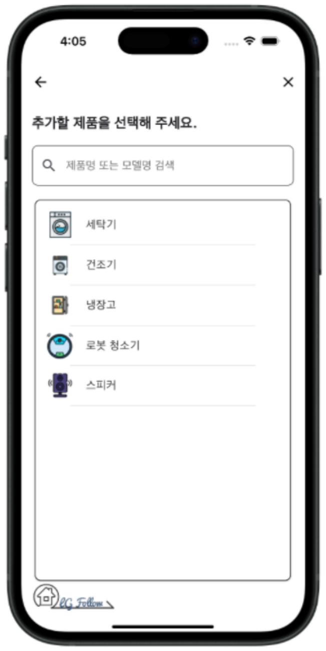


Fig. 26: Device Registration

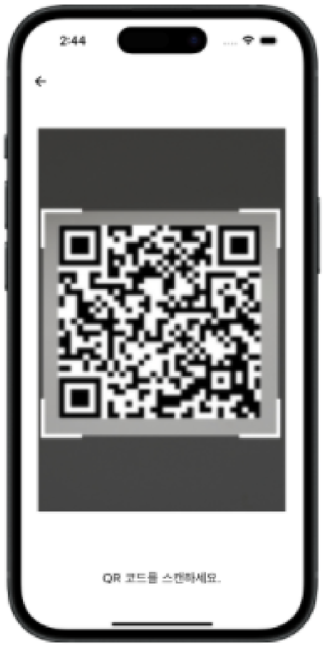


Fig. 27: QR Scan

- QR Scanning Frame: A square frame located in the center of the screen where users align the QR code.
- QR Code Recognition and Registration
  - Automatic Scanning: When a QR code is aligned within the scanning frame, the app automatically recognizes it without requiring a

separate confirmation button.

  - Validation: Once the QR code is scanned, the app verifies if it corresponds to a valid LG appliance model and matches the format stored in the database.
  - Registration: If validated, the appliance is registered to the user's account and linked to the LG Follow feature.

- Navigation Upon Successful Scan
  - Automatic Transition: Once the QR code is successfully recognized and the appliance is registered, the app automatically navigates the user back to the main screen.
  - Confirmation Message: Upon returning to the main screen, a brief confirmation message ('Appliance registered successfully') may appear to inform the user of successful registration.
- Error Handling: If the QR code is invalid or unrecognized, an error message appears ('Invalid QR code. Please try again').
- Through these features, users can easily control appliance power, manage room-specific LG Follow audio settings, and add new appliances.

E. Menu Page

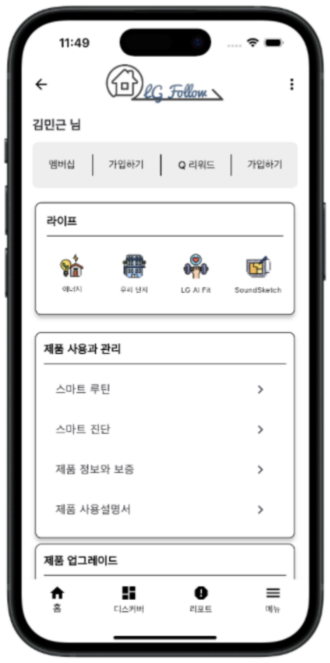


Fig. 28: Menu Page

User starts on the home screen and presses the menu button located on the bottom navigation bar, they open the menu page. From there, if the user navigates to the Life section and selects the Sound Sketch button, the app transitions to the Sound Sketch screen, providing them access to that feature.

F. Sound Sketch Initial Screen

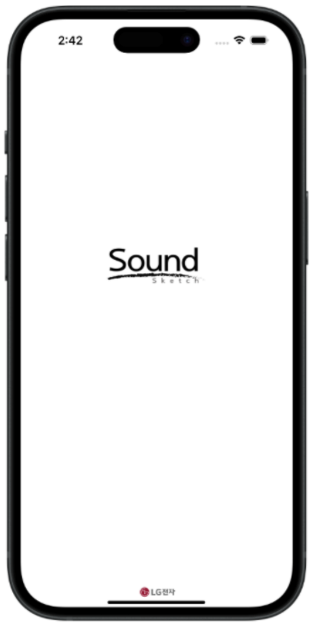


Fig. 29: Enter Sound Sketch

When the user navigates to the menu page and presses the Sound Sketch button, the screen transitions smoothly to the Sound Sketch initial screen. The initial screen displays the 'Sound Sketch' logo at the center along with a clean design. After a brief moment, the screen transitions seamlessly to the main functionality of the Sound Sketch feature.

G. Sound Sketch Main Page

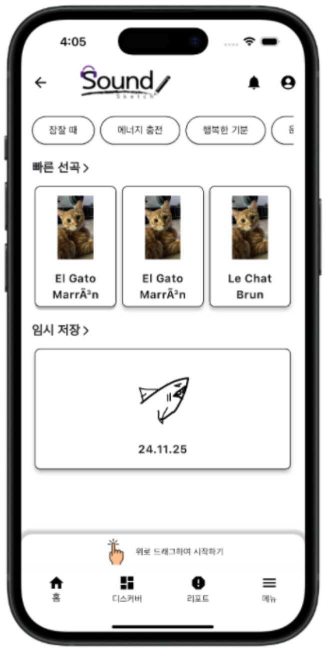


Fig. 30: Sound Sketch Main Page

1) Quick Playlist Section

- The Quick Playlist section, located at the top, displays a list of previously created drawings along with their associated music.
  - Thumbnail: Each record is shown as a thumbnail that displays the user's actual drawing.
  - Label: A label is displayed below each thumbnail, showing the song title and creation date to help users easily identify each record.
- Functionality: Users can select a previous drawing to view the related music information or to play the music.
- Saved Playlist: When a user completes and saves a drawing, it is added to the record section. The actual drawing is displayed as a thumbnail in the record section, and the metadata for the generated music is stored in the MySQL database via Spring Boot.
- Music Streaming: When a user selects a specific record thumbnail, the associated music is streamed in real-time, allowing the user to listen to it immediately.

2) Temporary Storage Section

- The Temporary Storage section is located at the middle and shows the drawings that the user started but hasn't finished yet.
  - Thumbnail: The thumbnail displays the user's current drawing in a large square, making it easy to recognize.
- Functionality: Even if the app is closed or the user switches tasks, the ongoing drawing is saved. When the user taps this thumbnail, they can resume working on the saved drawing.
- Image Temporary Storage: When a user temporarily saves a drawing, the image data is stored in a fast-access storage system. This ensures quick access and enhances the user experience by allowing seamless continuation during the drawing process.
- Temporary Data Recovery: When the user selects the saved drawing from the Temporary Storage section, the data is retrieved, allowing the user to resume their work without interruption.

3) New Start Section

- When the user drags the 'Drag Up to Start' section at the bottom of the screen, they can begin a new drawing.
- Functionality
  - Transition to Drawing Page: By dragging the button up, the user is taken to a new page where they can start a new drawing.
  - Temporary Data Reset: When a new project is started, the existing temporary data is either reset or updated to prepare for saving the new drawing.

H. *Sketch Page*

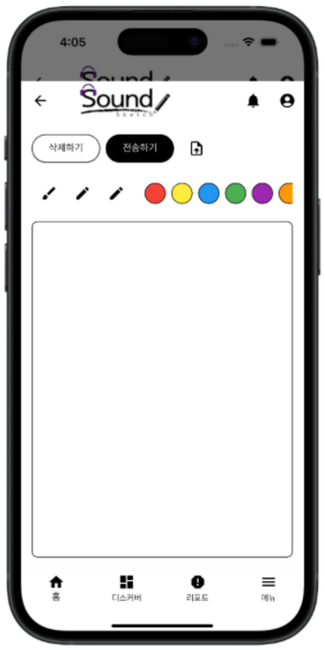


Fig. 31: Sketch Page

1) *Canvas Section*

- The white rectangle area in the center of the screen serves as a canvas where users can draw freely. This section is dedicated to allowing users to create and edit their drawings.
- Drawing Canvas: An interactive canvas where users can draw using various lines and colors.

2) *Submit Button*

- Allows users to submit the completed drawing to the server for music generation.

3) *Delete Button*

- Provides an option to delete the current drawing on the canvas.

4) *File Icon*

- Sound Sketch allows you to create songs by attaching image files.

5) *Back Icon*

- Allows users to return to the Sound Sketch main page.
- When clicked, the current drawing is temporarily saved, ensuring it is securely stored before navigating back to the Sound Sketch main page.

V. Architecture Design

A. *Overall Architecture*

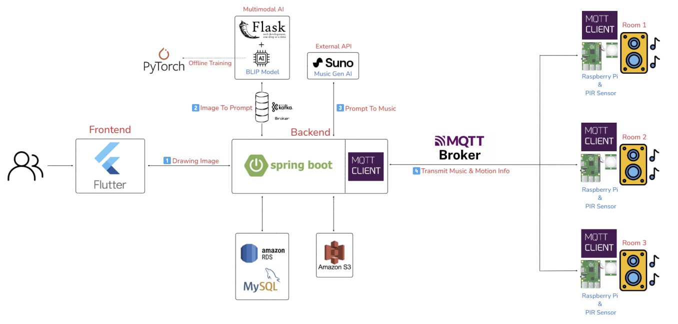


Fig. 32: Overall Architecture

The overall software architecture consists of three parts; frontend, backend and AI.

1) *Frontend*

- Framework: Flutter
- Design tool: Figma
- Summary of features:
  - Provides a canvas screen that allows users to draw and send their drawings to the backend.
  - Offers toggle buttons to turn LG appliances power and speakers On/Off.
  - Enables QR scanning to register LG appliances in the LG Follow app.
  - Provides a user interface to access previously created songs.
  - Displays an interface to view unfinished drawings.

2) *Backend*

- Web Framework: Spring Boot
- Database: My SQL. RDS
- Data storage: AWS S3
- API: Suno API
- Application tool: Apache Kafka, Docker
- Summary of features:
  - Generates songs based on prompts.
  - Stores user-drawn illustrations and the generated songs.

– It enables sound transmission between speakers.

3) AI

- Framework: PyTorch
- Web Framework: Flask
- AI Model: BLIP (Bootstrapping Language-Image Pre-training)
- Summary of features: Extracts prompts from user-drawn illustrations.

B. *Overall Directory Organization*

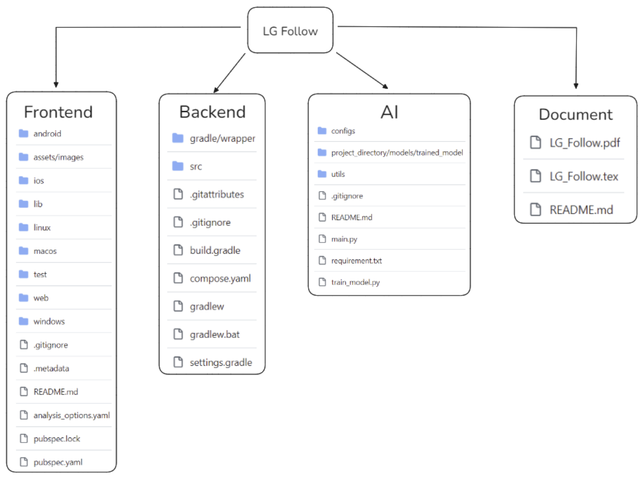


Fig. 33: Directory Organization

The LG Follow project consists of four GitHub repositories: Frontend, Backend, AI, and Document. Each repository plays a distinct role in the project's architecture and development.

The Frontend repository is responsible for developing the user interface and application for LG Follow. It includes code and resources to support multiple platforms, such as Android and iOS. The lib directory contains the core business logic and UI components of the frontend application, handling essential functionalities like screen navigation, state management, and user interactions. Additionally, configuration and dependency management files, such as pubspec.yaml and analysis options.yaml, are included to ensure efficient project maintenance.

The Backend repository is responsible for server-side logic in LG Follow. This repository includes all the necessary files to handle user requests, data processing, and other backend operations. The main implementation is located in the src directory, while Gradle build scripts (build.gradle, settings.gradle) and Docker configuration files (compose.yaml) support deployment and build automation.

The AI repository is dedicated to developing and managing LG Follow's artificial intelligence functionalities. It includes trained models (project_directory/models/trained_model), core scripts like main.py for executing AI processes, and train_model.py for model training. Dependencies for AI tasks are managed through the requirement.txt file. Additionally, the configs and utils directories provide configuration files and utility scripts, ensuring a modular and well-structured AI system.

The Document repository is responsible for managing the documentation of the LG Follow project. It includes the final project document in PDF format, LG_Follow.pdf, and the LaTeX source file, LG_Follow.tex, used to create it.

C. *Module 1: Backend*

1) Purpose

To implement the backend, we used Spring Boot to build an application, streamlining the development process. For database management, we adopted Amazon RDS, which provided a centralized and scalable solution. This setup enabled the team to collaborate effectively and handle data efficiently throughout the project. The centralized database allowed shared access and simplified maintenance tasks, significantly improving overall productivity. Additionally, we integrated the MQTT protocol to facilitate real-time communication with Raspberry Pi. This implementation allowed for efficient data exchange and control between the backend and the device.

2) Functionality

The LG Follow server executes tasks requested by users through the frontend and Raspberry Pi. It processes data and stores the corresponding values in the database. When a user requests song creation, the server generates the song, saves it to the database, and delivers it to the frontend. Additionally, it ensures smooth sound transmission between the Raspberry Pi devices using the MQTT protocol.

3) Location of source code:
https://github.com/LG-Follow/Back_End

TABLE II: Backend Directory Organization

Directory	**File Name**
Root	.gitattributes .gitignore build.gradle compose.yaml gradlew gradlew.bat settings.gradle
gradle/wrapper	gradle-wrapper.jar gradle-wrapper.properties
src/main/java/com/example /lgfollow_server	LgFollowServer Application.java
src/main/java/com/example /lgfollow_server/component	MqttMessageListener.java
src/main/java/com/example /lgfollow_server/config	KafkaConsumerConfig.java KafkaProducerConfig.java MqttConfig.java RestConfig.java S3Config.java
src/main/java/com/example /lgfollow_server/controller	ImageController.java SongController.java UserDeviceController.java UsersController.java
src/main/java/com/example /lgfollow_server/dto	ImageSendDto.java PromptGetDto.java SongDto.java SunoApiResponse.java UserDto.java
src/main/java/com/example /lgfollow_server/gateway	MqttGateway.java
src/main/java/com/example /lgfollow_server/model	Device.java Image.java Prompt.java Song.java SongRequest.java UserDevice.java UserDeviceId.java Users.java
src/main/java/com/example /lgfollow_server/repository	ImageRepository.java PromptRepository.java SongRepository.java UserDeviceRepository.java UsersRepository.java
src/main/java/com/example /lgfollow_server/security	SecurityConfig.java WebConfig.java
src/main/java/com/example /lgfollow_server/service	ImageService.java MqttPublisherService.java SongSendService.java SongService.java UserDeviceService.java UsersService.java
src/main/resources	application.properties application.yml
src/test/java/com/example /lgfollow_server	LgFollowServer ApplicationTests.java

4) *Class Component*

- LgFollowServerApplication.java: This is a file for Spring Boot Application entrypoint.

- lgfollow_server/component: This package contains components that handle reusable logic.
  - MqttMessageListener.java: A component that listens for incoming MQTT messages and processes them.
- lgfollow_server/config: This package contains configurations for external systems and application settings.
  - KafkaConsumerConfig.java: Configuration for Kafka consumers, defining properties for consuming messages from Kafka topics.

  - KafkaProducerConfig.java: Configuration for Kafka producers, specifying properties for sending messages to Kafka topics.

  - MqttConfig.java
    - Configuration for connecting to the MQTT broker, including client settings and connection parameters.

    - Dependency: org.springframework.integration:spring-integration-mqtt:6.3.5

    - Broker Setup: A local Mosquitto broker is used for managing MQTT message exchanges.

  - RestConfig.java: Configuration for REST API clients, including base URLs and request/response customization.

  - S3Config.java: Configuration for connecting to AWS S3, including bucket names, credentials, and storage options.

- lgfollow_server/controller: This package contains controllers that manage API endpoints for handling user requests.
  - ImageController.java: Handles API endpoints related to image upload, retrieval, and processing.

  - SongController.java: Manages API requests for song creation, storage, and retrieval.

  - UserDeviceController.java: Handles device-related actions, such as registration, updates, and communication.

  - UsersController.java: Manages user-related endpoints for profile management and data access.

- lgfollow_server/dto: This package contains Data Transfer Objects (DTOs) for exchanging structured data between layers.

  - ImageSendDto.java: Data Transfer Object (DTO) for sending image-related data between layers.

  - PromptGetDto.java: DTO for receiving prompt-related data.

  - SunoApiResponse.java: DTO representing the response structure from external Suno API.

  - UserDto.java: DTO for transferring user data, such as authentication or profile information.

- lgfollow_server/gateway : This package contains gateway interfaces for external communication.
  - MqttGateway.java: A gateway interface for sending messages through the MQTT broker.

- lgfollow_server/model: This package contains entity classes that represent the database structure and application models.
  - Device.java: Entity representing a device registered by a user, including its attributes.

  - Image.java: Entity representing images uploaded in the application.

  - Prompt.java: Entity for storing prompts submitted by users for various images.

  - Song.java: Entity for storing metadata and content related to songs created or retrieved.

  - SongRequest.java: Entity representing a user's request for song creation.

  - UserDevice.java: Entity representing the relationship between users and their devices.

  - UserDeviceId.java: Composite key entity for UserDevice relationships.

  - Users.java: Entity for storing user information, such as credentials and profile data.

- lgfollow_server/repository : This package contains repository interfaces for interacting with the database.
  - ImageRepository.java: Repository interface for database operations related to Image entities.

  - PromptRepository.java: Repository interface for database operations related to Prompt entities.

  - SongRepository.java: Repository interface for CRUD operations related to Song entities.

  - UserDeviceRepository.java: Repository interface for managing UserDevice relationships in the database.

  - UsersRepository.java: Repository interface for managing Users entities and user-related database operations.

- lgfollow_server/security: This package contains security configurations for authentication, authorization, and endpoint access control.
  - SecurityConfig.java: Configuration for securing the application, including authentication, authorization, and endpoint access.

  - WebConfig.java: Configuration for web-related settings, such as CORS policies and request/response customizations.

- lgfollow_server/service: This package contains business logic and services for the application.
  - ImageService.java: Service for handling image-related logic, such as upload, processing, and retrieval.

  - MqttPublisherService.java: Service for publishing messages to the MQTT broker.

  - SongSendService.java: Service for sending song-related data to external APIs or systems.

  - SongService.java: Service for managing songs, including creation, retrieval, and updates.

  - UserDeviceService.java: Service for managing user-device relationships, including registration and communication.

  - UsersService.java: Service for managing user-related logic, such as authentication, profile updates, and data access.

D. Module 2: AI

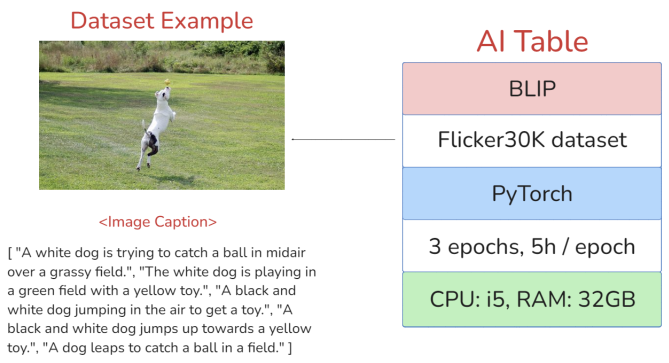
1) Purpose

Our team has developed a multimodal deep learning model based on the BLIP architecture to generate prompts from images. BLIP is a model that connects images and text, making it possible to create descriptive and meaningful prompts from images. To enhance the accuracy of BLIP and produce richer prompts, we fine-tuned the model using the Flickr30k dataset via the Hugging Face framework. Flickr30k is a comprehensive dataset containing everyday images with detailed annotations. This fine-tuning process has improved the model's ability to interpret and describe various types of visual input, including real-world images and hand-drawn illustrations.

2) Fuctionality

The model takes an input image and generates a descriptive prompt based on its visual features. This functionality allows the user to easily extract meaningful textual descriptions from various types of images, such as real-world photos or hand-drawn sketches. The model was trained using the Flickr30k dataset, enabling it to recognize and describe a wide range of visual elements effectively.

3) AI Table



["A white dog is trying to catch a ball in midair over a grassy field.", "The white dog is playing in a green field with a yellow toy.", "A black and white dog jumping in the air to get a toy.", "A black and white dog jumps up towards a yellow toy.", "A dog leaps to catch a ball in a field."]

Fig. 34: AI Table

- BLIP (Bootstrapping Language-Image Pre-training)
  - The BLIP model is a multimodal model designed to connect images and text. BLIP can generate descriptive and meaningful prompts based on images.
  - It is used to analyze input images and generate text prompts based on their visual features.
- Flickr30K dataset
  - The Flickr30K dataset is a comprehensive dataset containing a variety of everyday images, each annotated with detailed captions.
  - It was used to fine-tune the BLIP model, improving its accuracy and enhancing its ability to generate richer text prompts. By training on this dataset, the model has learned to better interpret visual elements and describe them effectively.
- PyTorch: PyTorch was the framework used for implementing and training the deep learning model. Its integration with Hugging Face facilitated the fine-tuning process using the Flickr30K dataset.
- Training Details
  - Epochs: The model was trained on the entire dataset for 3 epochs. This allows the model to sufficiently learn the patterns in the data and improve its generalization capabilities.
  - Time per Epoch: Each epoch took approximately 5 hours to complete, reflecting the model's complexity and the large size of the dataset.
  - Effect: These training configurations helped the model improve its ability to understand fine-grained visual details and convert them into meaningful text descriptions.
- System Configuration
  - CPU: Intel Core i5 processor
  - RAM: 32GB, enabling efficient processing of large datasets and fast training times.

4) Result of Fine-turned BLIP Model

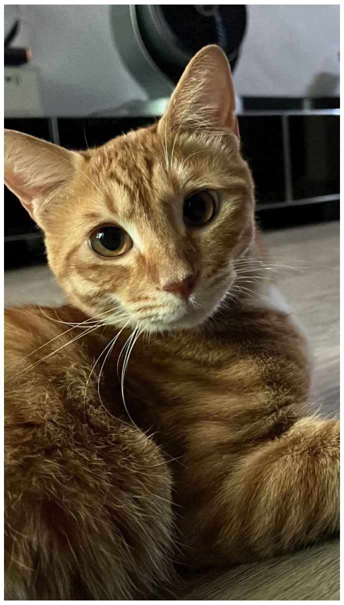


Fig. 35: Example picture

- The output caption of the original BLIP model: a cat laying on the floor.

- The output caption of the fine-tuned BLIP model: a brown cat is laying on the floor with its eyes wide open.

5) Location of source code:
https://github.com/LG-Follow/AI

TABLE III: AI Directory Organization

Directory	File Name
Root	.gitignore README.md requirement.txt train_model.py main.py
configs	blip_config.json
project_directory/models /trained_model	config.json generation_config.json preprocessor_config.json special_tokens_map.json tokenizer.json tokenizer_config.json vocab.txt
utils/__pycache__	blip_prompting.cpython-312.pyc blip_training.cpython-312.pyc blip_utils.cpython-312.pyc collate.cpython-312.pyc data_processing.cpython-312.pyc
utils	blip_prompting.py blip_training.py collate.py data_processing.py

6) Class Component
- main.py: The main entry point of the project.

- train_model.py: Script for training the BLIP model using provided configurations and datasets.

- configs: This is a directory for configuration of BLIP model training.
  - blip_config.json: Contains epoch number, architecture settings, and training configurations for BLIP.

- project_directory/models/trained_model: Directory containing the trained model and associated configuration files.
  - config.json: Metadata about the model, including architecture and dataset details.

  - generation_config.json: Settings for generating outputs.

  - preprocessor_config.json: Configuration for preprocessing images and input data for compatibility with the model.

- utils: A directory for utility scripts used during training, preprocessing, and inference.
  - blip_prompting.py: Script for generating prompts

from images using the BLIP model.

– blip_training.py: Script for handling the BLIP model's training pipeline.

– collate.py: Utility for batching and collating input data during training or inference.

– data_processing.py: Functions for processing and augmenting datasets for model training.

E. Module 3: Frontend

1) Purpose

The Frontend leverages the Flutter framework to provide an interface that supports user interactions. This allows users to control devices, view generated content, and process data visually. It focuses on enhancing the user experience by transmitting user inputs (e.g., drawings, settings) to the backend and visually presenting the processed data returned from the backend.

2) Functionality

Users can draw on a canvas screen and send their drawings to the backend. Additionally, it offers toggle buttons to turn LG appliances' power and speakers On/Off. The application also allows users to register LG appliances in the LG Follow app via QR code scanning. Furthermore, it provides a user interface to access previously created songs and a screen to view unfinished drawings.

3) Using MVVM Pattern (Model, View, ViewModel)

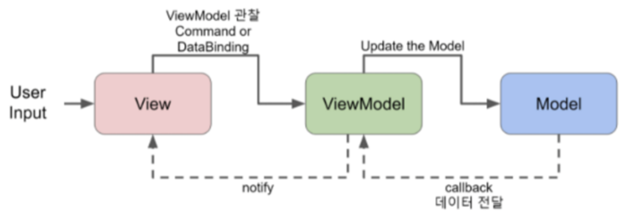


Fig. 36: MVVM Pattern

The View delivers user input to the ViewModel, which then processes the data or requests further information from the Model based on the input. Subsequently, the View subscribes to the ViewModel using tools like Providers, Streams, or state management techniques and updates the display to reflect the updated data state.

- Advantages of MVVM Pattern
  – Separation of Concerns: The application is divided into distinct areas: View, ViewModel, and Model, each with specific responsibilities. This separation significantly reduces the complexity of the program.

  – Ease of Testing: The clear interface between the View and ViewModel makes it easier to independently test the logic and functionality of the ViewModel.

  – Structured Project Organization: The MVVM pattern organizes the application into three components: View, ViewModel, and Model, which can be developed and managed independently. This structure improves maintainability and collaboration efficiency.

  – Parallel Development of UI: The separation of View and ViewModel enables simultaneous development of the UI and the ViewModel's logic, enhancing team productivity.

  – Abstraction of View: The View does not directly handle data but communicates with the ViewModel to send and receive data. The ViewModel serves as an intermediary between the UI and the business logic, reducing the coupling between them and improving maintainability.

4) Location of source code:

https://github.com/LG-Follow/Front-End

TABLE IV: Frontend Directory Organization

Directory	File Name
Root	.gitignore .metadata README.md analysis_options.yaml pubspec.lock pubspec.yaml
android	.gitignore build.gradle gradle.properties settings.gradle
android/app	build.gradle
android/app/src/debug	AndroidManifest.xml
android/app/src/main	AndroidManifest.xml MainActivity.kt
android/app/src/main/res /drawable-v21	launch_background.xml
android/app/src/main/res /drawable	launch_background.xml
android/app/src/main/res/mipmap-hdpi	ic_launcher.png
android/app/src/main/res/mipmap-mdpi	ic_launcher.png
android/app/src/main/res/mipmap-xhdpi	ic_launcher.png
android/app/src/main/res/mipmap-xxhdpi	ic_launcher.png
android/app/src/main/res/mipmap-xxxhdpi	ic_launcher.png
android/app/src/main/res/values-night	styles.xml
android/app/src/main/res/values	styles.xml
android/app/src/profile	AndroidManifest.xml
assets/images	Follow.png Follow_logo.png SoundSketch.png ai_fit.png aircon.png dryer.png energy.png finger 1.png finger 2.png finger.png lg_follow_logo.png lg_logo.png off.png on.png our_apt.png ref.png robot.png sound.png sound_sketch_logo.png speaker.png tv.png washing.png
ios	.gitignore Podfile Podfile.lock
ios/Flutter	AppFrameworkInfo.plist Debug.xcconfig Release.xcconfig

TABLE V: Frontend Directory Organization

Directory	File Name
ios/Runner	AppDelegate.swift Info.plist Runner-Bridging-Header.h
ios/Runner/Assets.xcassets /AppIcon.appiconset	Contents.json Icon-App-1024x1024@1x.png Icon-App-20x20@1x.png Icon-App-20x20@2x.png Icon-App-20x20@3x.png Icon-App-29x29@1x.png Icon-App-29x29@2x.png Icon-App-29x29@3x.png Icon-App-40x40@1x.png Icon-App-40x40@2x.png Icon-App-40x40@3x.png Icon-App-60x60@2x.png Icon-App-60x60@3x.png Icon-App-76x76@1x.png Icon-App-76x76@2x.png Icon-App-83.5x83.5@2x.png
ios/Runner/Assets.xcassets /LaunchImage.imageset	Contents.json LaunchImage.png LaunchImage@2x.png LaunchImage@3x.png README.md
lib	main.dart
lib/model	Drawing.dart Login.dart Point.dart Temp.dart device.dart drawing_painter.dart home.dart menu.dart product.dart sketch_home.dart song.dart
lib/view	Drawing_view.dart Follow_view.dart Login_view.dart Sound.dart Temp_view.dart home_view.dart menu_view.dart product_view.dart qr_scan_view.dart sketch_home_view.dart song_list_view.dart
lib/viewModel	Drawing_view_model.dart Login_view_model.dart Temp_view_model.dart home_view_model.dart menu_view_model.dart product_view_model.dart qr_scan_view_model.dart sketch_home_view_model.dart song_view_model.dart
linux	.gitignore CMakeLists.txt
linux/flutter	CMakeLists.txt generated_plugin_registrant.cc generated_plugin_registrant.h generated_plugins.cmake

TABLE VI: Frontend Directory Organization

Directory	File Name
macos	.gitignore Podfile
macos/Flutter	Flutter-Debug.xcconfig Flutter-Release.xcconfig GeneratedPluginRegistrant.swift
macos/Runner	AppDelegate.swift Info.plist
macos/Runner/Assets.xcassets /AppIcon.appiconset	Contents.json app_icon_1024.png app_icon_128.png app_icon_16.png app_icon_256.png app_icon_32.png app_icon_512.png app_icon_64.png
pubspec	pubspec.lock pubspec.yaml
test	widget_test.dart
web	favicon.png index.html manifest.json
web/icons	Icon-192.png Icon-512.png Icon-maskable-192.png Icon-maskable-512.png
windows	.gitignore CMakeLists.txt
windows/flutter	CMakeLists.txt generated_plugin_registrant.cc generated_plugin_registrant.h generated_plugins.cmake

5) *Class Component*

- .metadata: Contains project metadata.
- analysis_options.yaml: Configuration for static code analysis.
- android: Contains Android-specific files and configurations.
  - .gitignore: Specifies ignored files in the Android directory.
  - build.gradle: Defines the Android project's build configurations.
  - gradle.properties: Configuration properties for Gradle.
  - gradle-wrapper.properties: Gradle wrapper configuration file.
  - settings.gradle: Settings for the Android project.
  - src/debug/AndroidManifest.xml: Android manifest for the debug build.
  - src/main/AndroidManifest.xml: Main Android manifest defining app configurations.
  - src/main/kotlin/com/example/lg_follow/MainActivity.kt: Entry point for the Android app.
  - src/main/res/drawable-v21/launch_background.xml: Background for the launch screen for API level 21+.
  - src/main/res/drawable/launch_background.xml: Background for the launch screen for older APIs.
  - src/main/res/mipmap-/ic_launcher.png: Launcher icons in different resolutions.
  - src/main/res/values-night/styles.xml: Styles for night mode.
  - src/main/res/values/styles.xml: Default styles for the app.
- assets/images: Contains image assets used in the application.
  - Follow.png, Follow_logo.png: Images for branding and follow features.
  - SoundSketch.png, ai_fit.png, aircon.png, dryer.png, etc.: Miscellaneous assets representing app features or functionalities.
- ios: Contains iOS-specific files and configurations.
  - .gitignore: Specifies ignored files in the iOS directory.
  - Flutter/AppFrameworkInfo.plist: Info for the Flutter framework.
  - Flutter/Debug.xcconfig, Release.xcconfig: Debug and release configurations.
  - Podfile, Podfile.lock: CocoaPods dependencies and lock file.
  - Runner.xcodeproj: Xcode project file for the iOS app.
  - Runner/AppDelegate.swift: Entry point for the iOS app.
  - Runner/Assets.xcassets: Contains app icons and launch images.

  - Runner/Base.lproj: Contains storyboard files for iOS UI.

- lib: Contains the main Dart code for the application, including business logic, UI, and state management.
  - model: Contains data models representing the app's structure and behavior.
    - Drawing.dart: Defines a class for storing user-drawn sketches.
    - Login.dart: Manages user login information.
    - Point.dart: Represents point data in a sketch.
    - Temp.dart: Manages temporarily saved data.
    - device.dart: Stores information about IoT devices (e.g., speakers) to support appliance-related functionality.
    - drawing_painter.dart: Implements rendering and painting functionality for sketches.
    - home.dart: Manages data related to the home screen (registered appliances).
    - menu.dart: Manages menu information (e.g., option lists, icons).
    - product.dart: Manages appliance addition (e.g., selecting which appliance to add).
    - sketch_home.dart: Manages data needed for the Sound Sketch home screen.
    - song.dart: Manages song data such as title, URL, and creation date.
  - view: Contains UI components and widgets.
    - Drawing_view.dart: Provides a UI for creating or editing sketches.
    - Follow_view.dart: Sets up the app's initial screen (loading screen).
    - Login_view.dart: User login screen.
    - Sound.dart: Loading screen for navigating to the Sound Sketch feature.
    - home_view.dart: Implements the UI for the home screen.
    - menu_view.dart: Displays the app's menu.
    - product_view.dart: Displays a list of appliances to add.
    - qr_scan_view.dart: QR code scanning screen.
    - sketch_home_view.dart: Displays key features (e.g., sketches, temporary data) on the Sound Sketch home screen.
    - song_list_view.dart: Displays a list of songs and allows users to select one.
  - viewModel: Contains state management and logic.
    - Drawing_view_model.dart: Handles and manages sketch data, connecting the View and Model, and converts sketches to FormData or PNG format.
    - Login_view_model.dart: Handles login logic and user authentication.
    - Temp_view_model.dart: Manages logic for temporarily saved data.
    - home_view_model.dart: Manages data logic for the home screen.
    - menu_view_model.dart: Manages logic related to menu selection.
    - product_view_model.dart: Handles logic for adding appliances.
    - qr_scan_view_model.dart: Manages QR code scanning logic and processing.
    - sketch_home_view_model.dart: Manages data and actions for the Sound Sketch home screen.
    - song_view_model.dart: Manages the song list and playback state, connecting the UI and Model.
- pubspec.yaml: Lists dependencies and assets for the project.
- test: Contains unit tests for the application.
- web: Contains web-specific files for deploying the app as a web application.
- windows: Contains Windows-specific files and configurations.

VI. Use Case

A. Initial Screen

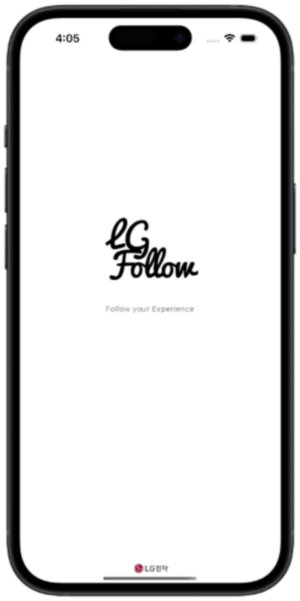
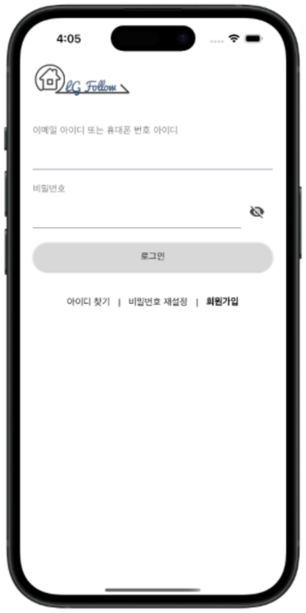


Fig. 37: Initial Screen

The LG Follow loading page briefly appears when users turn on the application. This page is displayed while the application is preparing. Once the loading is complete, it automatically transitions to the next page.

B. Log in



4:05
LG Follow
이메일 아이디 또는 휴대폰 번호 아이디
비밀번호
로그인
아이디 찾기 | 비밀번호 재설정 | 회원가입

Fig. 38: Log In

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

C. Main Page

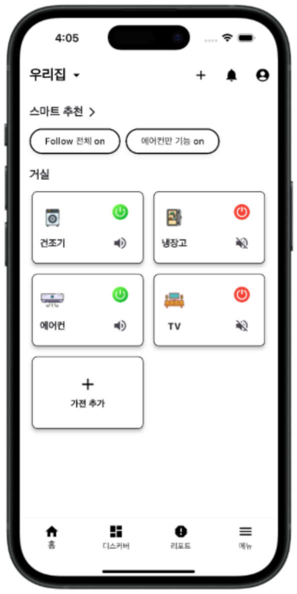


Fig. 39: Main Page

The Main page is where users can view registered devices or register new ones using a QR code. In the center of the page, users can toggle the speaker functionality on or off for each registered device using the button next to it. Below the last listed device, the Add Appliance button enables users to scan a QR code to register a new device. The bottom navigation bar offers options to return to the Main Page or navigate to the Menu Page.

D. Device Registration

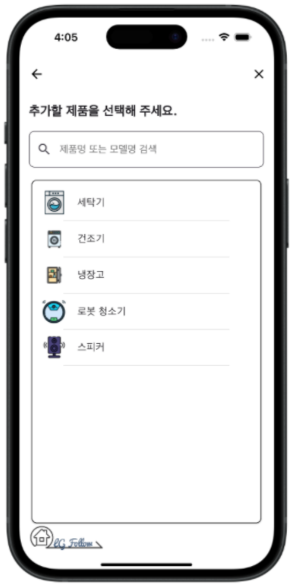


Fig. 40: Divice Registration

Users can tap the 'Add Appliance' button to scan QR codes using their smartphone's camera. By scanning a QR code, users can register their home appliances to the application.

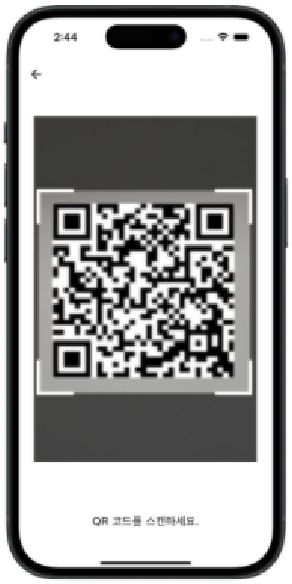


Fig. 41: QR Scan

E. Menu Page

Users can navigate to the menu page by tapping the menu button on the right side of the bottom navigator. In the center of the menu page, there is a 'Sound Sketch' button. Users can tap this button to access the Sound Sketch service.

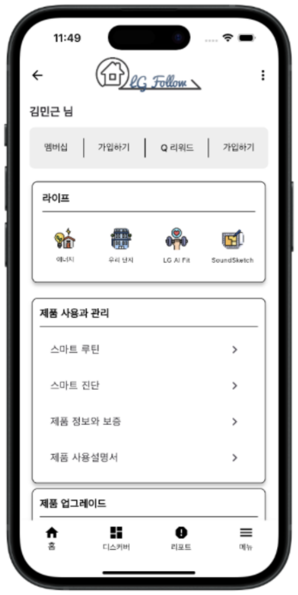


Fig. 42: Menu Page

F. Sound Sketch Initial Screen

Fig. 43: Enter Sound Sketch

The Sound Sketch loading page briefly appears when users click on Sound Sketch from the main page. This page is displayed while the application is preparing. Once the loading is complete, it automatically transitions to the next page.

G. Sound Sketch Main Page

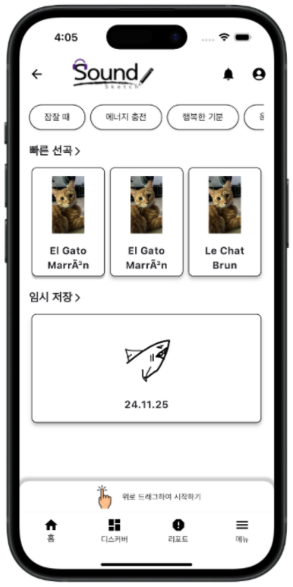


Fig. 44: Sound Sketch Main Page

Sound Sketch main page consists of three main sections. At the top, the Quick Playlist section displays previously created drawings and their associated music as thumbnails with labels, allowing users to view details or play the music. In the center, the Temporary Storage section shows saved but unfinished drawings, which users can resume by tapping the thumbnail. At the bottom, dragging up the 'Drag Up to Start' area opens a canvas where users can create new drawings.

H. Sketch Page

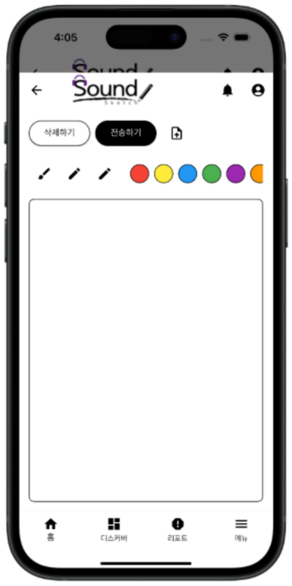


Fig. 45: Sketch Page

The Sketch page, users can draw on the central canvas, selecting from a variety of brush thicknesses and colors to create their artwork. The Delete button at the top lets users erase their current drawing, and the Submit button to its right enables them to generate music from their drawing. Additionally, by clicking the File icon, users can attach an image file, allowing them to import a pre-made image and use it for music generation.

VII. Discussion

1) Technical Difficulties

Our goal was to play a song created with Sound Sketch on the NUGU AI Speaker and then transfer the sound output to a speaker connected to a Raspberry Pi. However, we faced several difficulties in implementing this functionality. Despite our efforts to resolve the issue using NUGU Playbuilder and the NUGU SDK, we were unable to achieve the desired outcome.

One of the primary challenges was the lack of direct support for routing the audio output from the NUGU AI Speaker to an external speaker via Raspberry Pi. Although both NUGU Playbuilder and the NUGU SDK offer robust capabilities for building and managing AI speaker interactions, they did not provide a seamless solution for audio output redirection to another device in this particular setup.

We explored various alternatives and potential workarounds, but the solution remained elusive due to technical limitations in the software and hardware configuration. This experience highlighted the importance of ensuring compatibility and support across all integrated technologies when planning a multi-device setup. Moving forward, we would explore different approaches, such as using alternative software or hardware solutions, to better facilitate this integration.

Ultimately, while the project presented technical challenges, it also offered valuable insights into the complexities of working with multiple technologies simultaneously. This experience will guide future projects where similar integrations are required.

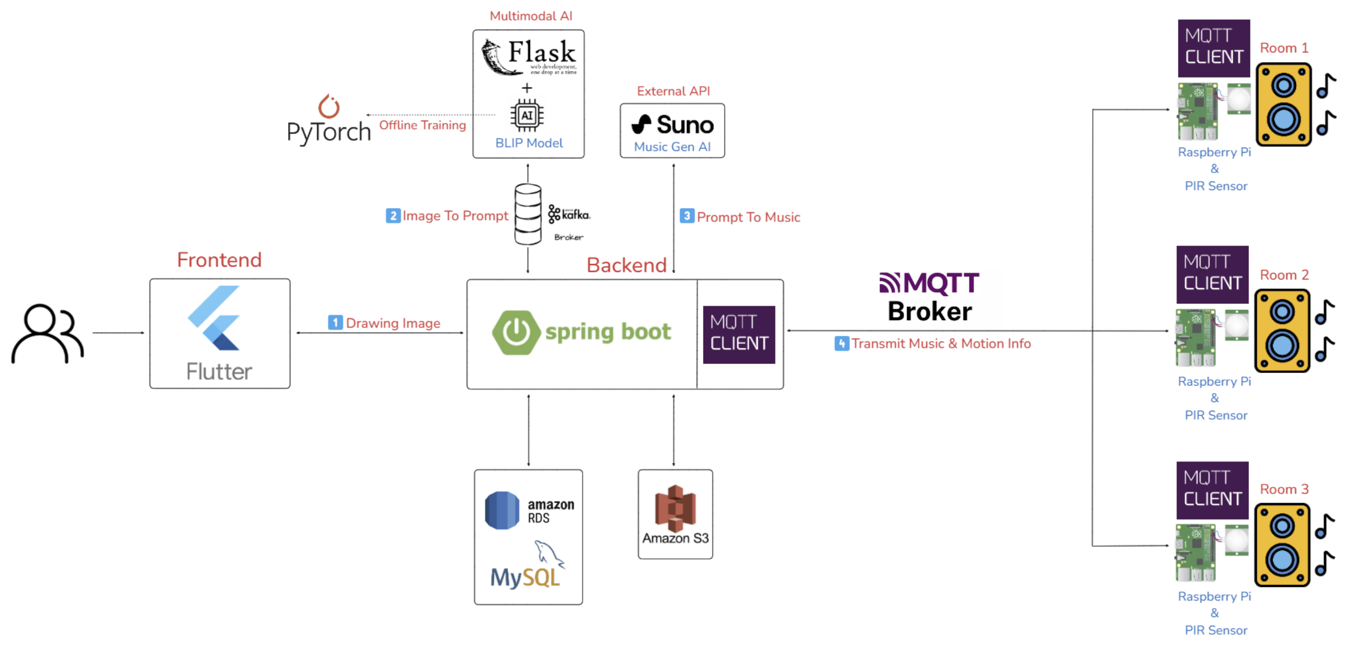
2) Conclusion

Currently, LG Follow is implemented by connecting PIR sensors and speakers to a Raspberry Pi to track the user's location and seamlessly switch the sound to the speaker in the room where the user is present.

However, this technology is just the beginning. If speakers are embedded into home appliances and NFC is utilized, audio will seamlessly blend into our daily lives, creating a more immersive experience.

LG Follow is not just about moving sound from one place to another. This technology has the potential to expand the scope of smart home systems into the realm of sound. We hope that LG Follow will further enhance expectations for the future of smart home technology.

Appendix



References

[1] Li, Junnan et al. (2022). BLIP: Bootstrapped Language-Image Pre-training for Unified Vision-Language Understanding and Generation. Retrieved from https://huggingface.co/Salesforce/blip-image-captioning-base

[2] Plummer, B.A., Wang, L., Cervantes, C.M., et al. (2015). Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. Retrieved from https://huggingface.co/datasets/nlphuji/flickr30k

[3] Amazon RDS Free Tier. (n.d.). Retrieved from https://aws.amazon.com/free/database/

[4] CUDA Toolkit. (n.d.). Retrieved from https://developer.nvidia.com/cuda-toolkit

[5] Figma. (n.d.). Retrieved from https://www.figma.com/

[6] Flask. (2023). Retrieved from https://flask.palletsprojects.com/en/3.0.x/

[7] Flutter. (n.d.). Retrieved from https://flutter.dev

[8] Flow. (n.d.). Retrieved from https://flow.team/kr/index

[9] GitHub. (n.d.). Retrieved from https://github.com/

[10] Kafka. (n.d.). Retrieved from https://kafka.apache.org

[11] LG ThinQ. (n.d.). Retrieved from https://www.lge.co.kr/lg-thinq

[12] MySQL. (n.d.). Retrieved from https://www.mysql.com

[13] Pillow. (n.d.). Retrieved from https://python-pillow.org

[14] PyTorch. (n.d.). Retrieved from https://pytorch.org

[15] scikit-learn. (n.d.). Retrieved from https://scikit-learn.org/stable/

[16] Spring Boot. (2023). Retrieved from https://spring.io/projects/spring-boot

[17] TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org