VOICE

A Multimodal Routine Recommender for Any Circumstances

Song Jaewhi
dept. Information System
Hanyang Univ.
Seoul, Republic of Korea
wotns0319@naver.com

Yu Hyeonseo
dept. Information System
Hanyang Univ.
Seoul, Republic of Korea
daina192939@gmail.com

Lee Seungjin
dept. Information System
Hanyang Univ.
Seoul, Republic of Korea
cookjin@hanyang.ac.kr

Choi Hyungrak
dept. Information System
Hanyang Univ.
Seoul, Republic of Korea
hrgr0711@naver.com

***Abstract*—Our team introduces 'VOICE', a multimodal smart home configuration and management application that leverages Speech-to-Intent (STI) technology and text-based interactions. VOICE aims to revolutionize the daily lives of busy modern individuals by making smart home systems more intuitive and efficient.**

VOICE enables users to easily register and manage home appliances through a user-friendly interface. The 'AI's Pick' feature, in particular, provides optimal routines tailored to the user's situation. This feature utilizes a model trained on ten thousands of routines generated by state-of-the-art generative AI model APIs, analyzing user input and context to suggest customized routines. Furthermore, through integration with AI speakers, VOICE offers the ability to recommend and execute routines via voice commands. It also includes an innovative approach where the speaker continuously monitors conversations to proactively suggest routines appropriate to the situation. VOICE's multimodal analysis technology not only understands words but also analyzes voice tone and speed to comprehend hidden emotions and context when recommending routines.

By combining a text-based generative AI model with voice recognition technology, VOICE recommends optimal routines for a wide variety of situations, allowing smart home technology to blend more naturally into users' daily lives. Our goal is to increase the accessibility and utilization of smart home technology across all age groups through VOICE. This project aims to create smart home systems using an intuitive user interface and natural voice interactions, leveraging technology to enhance accessibility and ease of use. In particular, the AI-based personalized routine recommendation feature provides an optimized smart home experience tailored to users' living patterns without complex settings. This will make daily life more convenient for a wide range of users, from the elderly to younger generations.

***Index Terms*—Speech-to-Intent, Multimodal, Smart Home, Personalization, Generative AI**

TABLE I
ROLE ASSIGNMENTS

Roles	Name	Task description and etc.
User/Customer	Lee Seungjin	User/Customer plays the role of an individual of any age seeking to enhance his daily life through intuitive smart home technology. He seeks to interact with his living space through voice and text commands, with the system recognizing his emotions and situational context. He desires a service that not only simplifies control over multiple devices with intelligent, context-aware support, but also creates a responsive living environment that enhances his comfort and efficiency by predicting and meeting his needs with minimal manual input.
Development Manager	Yu Hyeonseo	A Development Manager oversees the entire software development lifecycle, setting goals and establishing methodologies for timely, quality deliverables. She communicates with clients and developers, coordinating efforts across functions and adapting strategies as needed. Through comprehensive oversight of software design, development, and testing processes, she ensures the delivery of high-quality products that effectively fulfill user requirements and adhere to best practices in the field.
AI Developer	Song Jaehwi	An AI Developer develops a multimodal AI service that generates optimal smart home routines based on user circumstances. He is responsible for designing and implementing the AI architecture, from dataset creation to deployment. This involves text-based generation initially, followed by speech recognition with emotion and context analysis. He then utilizes transfer learning techniques for model training and ensures seamless integration with necessary libraries. Additionally, he monitors and fine-tunes the AI system to continually improve its recommendation accuracy and adaptability to diverse user scenarios.
Software Developer	Choi Hyungrak	A Software Development is responsible for transforming project requirements into functional, high-quality software applications. Along with the development team, he designs scalable architectures and writes clean, maintainable code. His responsibilities extend beyond coding to include testing, debugging, and quality assurance to ensure robust and error-free products. Additionally, he documents development processes and software functionalities, enabling future modifications and ensuring project continuity.

I. Introduction

A. Motivation

The recommendation engine market size is projected to hit $12 billion by 2025, highlighting a growing need for personalized, data-driven solutions in our daily lives. This trend underscores the potential for innovative approaches to home automation and personal assistance. In today's fast-paced world, managing daily routines and making decisions about household activities can be overwhelming for many people.

Our project aims to address these challenges by creating a more intuitive and responsive smart home experience. We envision a system that goes beyond the current limitations of smart home technology, offering users a seamless and natural way to interact with their living spaces. This approach allows users to communicate with their home systems using natural language. By enabling this type of interaction, we're making it possible for users to have more nuanced control over their smart home environment. Users can easily express their needs, preferences, and even complex instructions in a way that feels intuitive and effortless.

Smart home voice interaction systems typically require specific trigger words or phrases, resulting in an experience that users often find unnatural and constraining.. Our project seeks to implement advanced voice-to-intent technology, creating a more fluid and conversational experience. This level of natural language processing will make the interaction between users and their smart homes more intuitive and less burdensome.

Our project is also motivated by the desire to make smart homes truly 'smart' – capable of learning and adapting to users' routines and preferences over time. While existing systems handle common situations well, we're pushing further by incorporating highly unusual cases into our AI's training, ensuring our system can respond effectively to any situation. This level of automation and personalization will significantly reduce the cognitive load associated with home management. We believe that by pushing the boundaries of what's possible in home automation, we can make a significant positive impact on people's daily lives, helping them to manage their homes more efficiently and freeing up time and mental energy for the things that matter most to them.

B. Problem Statement

1 Current smart home systems require users to manually create routines through existing UI, which presents some challenges:

1) Accessibility

   The setup process can be complex for some users, particularly those less familiar with technology. There's an opportunity to make these systems more intuitive for all users, including elderly individuals or those with less interest in technology.

2) Feature Discovery

   Users sometimes utilize only a portion of the system's capabilities. For instance, with air conditioning control, many might use basic on/off functions without exploring more advanced features like temperature adjustment or fan speed control. We could explore ways to encourage users to discover and use more features.

3) User Experience

   There's potential to enhance the setup process to make it more enjoyable and less frustrating for users, which could improve overall perceptions of smart home technology.

4) Adaptability

   As user needs change over time, adjusting the system can be challenging. We could look into making ongoing management and adjustments more straightforward and user-friendly.

2 This manual process is time-consuming and may not account for all potentially useful routines. We see opportunities to address this:

1) Time Efficiency

   In today's fast-paced world, users often don't have much time to optimize their smart home systems. We could explore ways to make the optimization process quicker and easier.

2) Maximizing Potential

   Due to time constraints, users might not fully utilize all the capabilities of their smart home systems. For example, they might use simple on/off schedules instead of more complex, energy-saving routines. There's potential to help users leverage more advanced features without requiring significant time investment.

3) Ease of Updates

   As users add new devices or change their living patterns, updating routines can be inconvenient. We could look into ways to make these updates more seamless and automatic.

4) Enhancing Satisfaction

   By making system setup and management less time-intensive, we could potentially increase user satisfaction and encourage wider adoption of smart home technology.

3 Users may miss out on beneficial routines they haven't thought to create themselves. There's room to broaden the range of possibilities:

1) Inspiring Creativity

   Users might not always think of all the potential routines that could benefit them. For instance, while many

might set up basic work-from-home routines, they might not consider more novel ideas like automated end-of-workday celebrations. We could explore ways to inspire users with creative routine suggestions.

2) Unlocking Value

By helping users discover routines they might not have thought of themselves, we could enhance the overall value and appeal of smart home systems.

4 Current AI systems offer excellent recommendations for common scenarios in smart homes, but adapting to a broader spectrum of situations could be advantageous:

1) Accommodating Non-Standard Scenarios

There's an opportunity to better serve users with non-typical schedules or needs, such as shift workers who might leave for work at unusual hours.

2) Contextual Intelligence

While current systems often work with time or date-based routines, we could explore ways to incorporate more contextual information, such as user behavior, location, or other relevant factors, to make the system more responsive and intuitive.

C. Research on Related Software

1 ThinQ

LG Electronics' ThinQ is an AI-driven smart home platform. This technology is embedded in a range of household devices. ThinQ enables remote management through mobile applications or tracks power usage, fine-tunes operations by studying user habits, and offers self-troubleshooting capabilities The platform strives to deliver an enhanced, streamlined smart living environment by harnessing machine learning and IoT innovations, surpassing basic appliance manipulation.

2 NUGU Smart Home

NUGU Smart Home is SK Telecom's AI-powered smart home platform for the South Korean market, built around the Korean-language NUGU AI assistant. Users can control various smart devices through voice commands or a smartphone app, with features including customized routines, energy management, and security integration. The platform learns user preferences over time and leverages SK Telecom's infrastructure while expanding its ecosystem through third-party partnerships to compete in Korea's smart home market.

3 SmartThings

SmartThings is Samsung Electronics' open IoT platform that connects and controls various smart home devices from both Samsung and third-party manufacturers. Users can manage devices through the SmartThings mobile app or voice commands via assistants like Bixby. The platform features automation through custom routines, device monitoring, energy management, and home security integration. Supporting multiple protocols (Zigbee, Z-Wave, Wi-Fi), SmartThings continues to expand its ecosystem through manufacturer partnerships to provide a seamless smart home experience.

4 Apple Home Kit

Apple HomeKit is Apple's smart home platform that enables control of various smart devices from both Apple and third-party manufacturers through Siri voice commands, the Home app, and Control Center. It offers key features like scenes (predefined configurations), automations (time or location-based actions), remote access, and end-to-end encryption for security. The system requires an Apple device running recent iOS/iPadOS/macOS, compatible smart devices, and a home hub (Apple TV, HomePod, or iPad), while seamlessly integrating with other Apple services like CarPlay and Apple Watch.

5 Amazon Alexa

Amazon Alexa is Amazon's cloud-based voice assistant that uses natural language processing and machine learning to respond to user commands. First introduced in 2014 with the Echo speaker, Alexa can perform tasks like playing music, providing information, controlling smart home devices, and setting reminders. Available on Amazon devices and third-party products, Alexa's capabilities can be extended through developer-created "Skills," with Amazon continuously improving its AI and features for enhanced smart home functionality.

6 Amazon Echo (with Alexa Guard)

The Amazon Echo, equipped with Alexa Guard, is a smart speaker that enhances home security by monitoring ambient sounds. It can detect breaking glass, alarms, and even human voices, providing real-time alerts to users' smartphones when unusual sounds are detected. Users can enable this feature when leaving home, allowing Echo to actively listen for potential intrusions. However, its accuracy can be affected by background noise, leading to false alarms. Additionally, privacy concerns may arise since the device continuously listens for sounds. Alexa Guard is also dependent on the Echo device, limiting its integration with other smart home systems.

7 Josh.ai

Josh.ai is a premium smart home automation system that uses advanced natural language processing to enable intuitive voice control of smart devices. It understands natural language commands, allowing users to issue complex instructions without needing to memorize specific phrases. Josh.ai emphasizes privacy, processing voice

commands locally to keep user data security. While it can automate tasks and adjust multiple devices based on user commands, it currently does not offer proactive recommendations based on user situations. Instead, it excels in executing pre-defined routines and responding to direct commands. Josh.ai integrates seamlessly with a wide range of high-end devices, providing a sophisticated and user-friendly smart home experience.

8 Ring

Ring is a smart home security camera system designed to enhance safety through video surveillance and sound detection. It can recognize specific sounds, such as breaking glass or alarms, sending immediate alerts to users when these noises are detected. Ring cameras provide live video feeds, allowing users to monitor their property in real-time and respond to potential threats. They also store recorded incidents in the cloud for later review, ensuring users have access to important footage. However, the sound detection feature may sometimes result in false alarms due to non-threatening noises, like barking dogs. Privacy concerns are also prevalent, as continuous monitoring might invade personal space. Additionally, some Ring models require a subscription fee for cloud storage, adding to the overall cost of ownership.

II. Requirement Analysis

A. Common Features

1 Sign Up

1) Required information include: Email (ID), password, phone number, and name.
2) Password requirements include: minimum 8 characters with combination of letters, numbers, or special characters.
3) Name becomes default nickname and cannot be changed.
4) Phone number is used for account verification and recovery.

2 Log In

1) The system supports local login using email and password credentials.
2) Users can also log in using the SNS platform integration with Google.
3) Invalid login attempts trigger appropriate error messages.
4) Upon successful login, users are redirected to the main page.

3 Account Recovery

1) Find ID

   Users can find their ID through verification of their registered phone number.

2) Reset Password
   a) For password reset, users must verify their identity through email and phone authentication.
   b) The new password must meet the same requirements established during signup.
   c) For security purposes, the password input is masked with asterisks.

B. In Application

1 Splash Screen

Users are redirected to a landing page that serves as a splash screen after successful authentication. This landing page features a clean, user-friendly design with VOICE's logo.

2 Main Page

The main page should be expressed with maximum simplicity. It must feature the VOICE background image, accompanied by a welcome message stating "[User Name]'s Home" and include a menu bar. There are four options:

1) Home

   This button allows users to navigate from other pages to the main page.

2) AI's Pick

   This button enables users to navigate to the AI's Pick page.

3) Application Management

   This button enables users to navigate to the appliance management page.

4) Mypage

   This button enables users to navigate to the mypage.

3 AI's Pick

This page utilizes a model trained on tens of thousands of routines generated by state-of-the-art generative AI model APIs to recommend a single routine tailored to the user's specific situation.

1) Situation Input
   a) When a user inputs their situation as plain text, the system directly preprocesses this input by appending it to an existing predefined template. This approach ensures the input is structured in a way that aligns with the requirements of the fine-tuned AI model.

b) The combined input is then formatted as a query and sent to the AI model, which processes the input to generate a routine recommendation. This recommendation reflects the model's understanding of the user's situation within the provided context.

c) The generated output is further processed to enhance its usability. The system parses the output, breaking it down into specific instructions for each smart device, and organizes this data into a structured JSON format. This format allows for easy integration with smart home systems, enabling seamless control and execution of the recommended routines.

2) Recommendation Retrieval Process

Upon receiving the response, the system processes the data to display it on the screen. During this retrieval period, a loading screen is presented to the user.

3) Display

The system presents the routine, detailing how each appliance should operate.

a) Users are given options to accept the recommended routine, reject it, or request a different recommendation.

b) If users choose to accept the recommendation, the system executes the routine and redirects the AI's Pick initial screen.

c) Routines that have been accepted and executed are stored in the Routine History database.

d) If users choose to reject the recommendation, the system redirects to the AI's Pick initial screen without any additional processing.

4 Appliance Management

This page serves as the central hub for managing registered smart home appliances within the VOICE system.

1) Users can register and remove smart home appliances they wish to manage through VOICE. Any registration or removal action should be instantly reflected in the database in real-time.

2) This page allows for manual control of appliances. In instances where manual operation is necessary, users can perform actions such as turning appliances on or off, adjusting temperature settings, and other complementary controls.

5 Mypage

The Mypage allows users to view their personal information and routines they have executed based on recommendations. It also enables AI speaker integration.

1) Profile Page

User can view or modify their personal data.

a) Personal Data Lookup

Users can view their personal details (specifically name, email, and phone number) on this page.

b) Modify Password

After verifying identity through mobile phone authentication, allow the user to modify their password.

2) Routine History Page

Users can review routines they chose to execute based on recommendations.

a) Situation Search

When searching for the situation in which a routine was recommended, relevant routines are displayed in a viewable format.

b) Routine Selection

When users click on a specific routine, the system displays detailed information about the situation and routine, along with execute and delete buttons. Pressing execute initiates the routine, while the delete button removes the routine from the routine history collection.

3) AI Speaker Integration

Speaker integration for receiving routine recommendations via voice.

a) Press 'Find AI Speaker'.

b) When VOICE detects an AI speaker, select the device from the list.

c) Follow VOICE's instructions and press 'Connect'.

d) Upon successful connection, a message "Connection Complete" is displayed.

C. With AI Speaker

The AI speaker integrated with VOICE will recognize voice commands and provide corresponding routines. This service offers two methods of use:

1 User-Initiated Interaction

Users initiate dialogue with the speaker to receive routine recommendations. The system processes various aspects of the interaction through multiple steps:

1) Initial Interaction The interaction begins when a user triggers the system with "[Speaker Name] start", to which the speaker responds with a greeting "Hello, I'm [Speaker Name]"

2) Voice Processing and Analysis

The user's voice data is converted to WAV format and processed through STT and emotional analysis. The emotional analysis results are combined with the converted situation text to create the AI model input.

3) AI Model Processing

The fine-tuned model generates recommendations by considering the user's command and situation, emotional context analysis, previous routine history, and the user's registered smart devices.

4) Response Processing

The model's recommendations are parsed into device-specific instructions and structured for smart home system control.

5) Voice Response

The processed recommendations are converted to speech using TTS, allowing the speaker to announce the selected routine to the user, after which the routine is executed.

6) Data Recording

The system stores each interaction's details including request information, emotional context translated into corresponding emojis, and executed device actions for future optimization.

2 Continuous Conversation Tracking by AI Speaker

The continuous monitoring mode enables AI speakers to autonomously process ambient conversations without requiring explicit user triggers, allowing users to receive tailored routine recommendations for any situation that may arise.

1) Ambient Monitoring

The system continuously processes ambient audio through Speech-to-Intent technology, actively listening for potential routine situations and contextual cues.

2) Voice Processing and Analysis

When a routine-applicable situation is detected, the captured audio is processed through STT and emotional analysis.

3) Further Processing

From this point, the system follows the same process as steps 3-6 of user-initiated interaction.

D. AI Model

To create a multimodal AI service that recommends suitable smart home routines for users in diverse situations, VOICE has divided its service into three components:

1 Dataset Creation for Smart Home Routine Recommendations

We developed a comprehensive dataset of 10,000 unique smart home routine recommendations using state-of-the-art generative AI model APIs. Rather than using the APIs directly for real-time recommendations, which often produced duplicate and unrealistic routines, we created and validated an extensive dataset. Through detailed prompt engineering, we carefully curated unique and specialized situations. The structured JSON dataset, containing contextual situations and device actions, underwent rigorous deduplication and has proven invaluable for understanding smart home automation patterns. Our multi-model approach enhanced dataset diversity while providing insights into AI systems' interpretation of complex automation scenarios.

2 Development of a Text-based Generative AI Model Within the Application

This will utilize state-of-the-art generative AI model APIs to generate ten thousands of routine data points. To recommend appropriate routines for users in diverse situations, a comprehensive dataset will be constructed, including a wide range of scenarios, even those of minimal significance. Subsequently, paust/pko-chat-t5-large model, a transformer-based variant of Google Flan-T5 fine-tuned with Korean chat data, will be fine-tuned to align it with the unique requirements of our smart home routine recommendation service.

3 Expansion into Voice Domain Using SKT NUGU (AI Speaker)

In order to expand our service domain from text to voice, we combined existing strong AI services, Google STT and Hume AI, to not only convert the voice audio to text but also extract additional verbal information like emotions and urgency by analyzing the voice audio signal. Those verbal information will then be tagged with converted text to better understand users' situations. The system will then recommend optimal routines based on users' voice requests. Furthermore, it will continuously track user conversations to capture additional information such as context, keywords, and urgency in everyday life, and provide the most suitable routines.

III. Development Environment

A. Choice of Software Development Platform

1 Development Platform

1) Windows

Windows is Microsoft's operating system that provides a comprehensive development environment through its extensive tools and frameworks. Its core IDE Visual Studio enables development across multiple languages, while the .NET framework supports diverse application development. The platform features Windows Terminal, Package Manager, and Windows Subsystem for Linux (WSL) for modern development workflows.

Windows offers strong enterprise integration, PowerShell automation, DirectX support for gaming, and robust security features. Its Azure cloud integration and widespread market presence ensure broad compatibility and testing capabilities, making it a versatile development platform for various applications.

2) macOS

macOS is Apple's operating system providing a robust development environment centered around Xcode for Apple ecosystem development. Its Unix foundation offers powerful command-line capabilities, while package managers like Homebrew facilitate tool management. The platform excels in native development through Swift and Objective-C support, and includes comprehensive debugging and performance optimization tools. macOS integrates smoothly with Apple services like iCloud and TestFlight for deployment, while maintaining security through features like Gatekeeper. Its UNIX certification and virtual machine support enable versatility across different development scenarios.

2 Language and Framework

1) Python

Fig. 1. Python

Python is a versatile and widely used high-level programming language, praised for its simplicity and readability. This makes it particularly attractive for both beginners and experienced developers. Python's extensive standard library and rich ecosystem of third-party libraries provide powerful tools for various tasks, including web development, data analysis, and artificial intelligence. The language's strong support for object-oriented, imperative, and functional programming paradigms allows developers to choose the style that best fits their needs. Furthermore, Python is heavily utilized in the AI community due to its robust frameworks and libraries that facilitate tasks such as data preprocessing, model building, and evaluation.

2) JavaScript

Fig. 2. Javascript

JavaScript has been chosen as the primary programming language for our project. As the most widely used language in modern web and mobile application development, we particularly utilize various cutting-edge features of the ES6+ version. The selection of JavaScript enables effective implementation of essential elements in frontend development and offers perfect compatibility with React Native. We can fully utilize JavaScript's powerful features in implementing asynchronous programming, modular development, and component-based architecture. Furthermore, its rich ecosystem and support for various libraries make the development process more efficient.

3) Expo & React Native

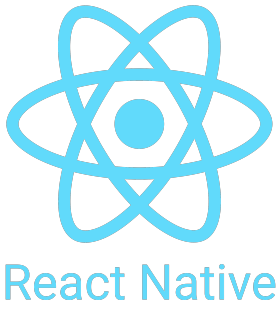

Fig. 3. React Native

Expo and React Native serve as the cornerstone frameworks for our team's mobile application development. React Native offers the significant advantage of enabling cross-platform development that supports both iOS and Android platforms with a single codebase, while Expo provides a comprehensive toolkit that streamlines React Native development. Particularly, Expo's powerful build tools and straightforward development environment configuration significantly enhance development productivity and enable easy implementation of various native functionalities. Additionally, through extensive community support, it provides an environment where issues arising during development can be swiftly resolved.

4) Node.js

Fig. 4. Node.js

Node.js is a server-side runtime environment designed to build fast and scalable network applications. This open-source platform utilizes an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js is particularly effective in handling real-time data processing and concurrent connections through its single-threaded event loop architecture. The platform enables efficient implementation of RESTful APIs, manages database interactions effectively, and provides robust support for various server-side operations. Its architecture naturally supports processing multiple simultaneous requests and maintaining real-time communication through WebSocket protocols, making it ideal for modern web applications that require high-performance backend operations.

5) Sequelize

Fig. 5. Sequelize

Sequelize is a robust Object-Relational Mapping (ORM) framework for Node.js that supports multiple SQL databases. As a promise-based ORM, it simplifies database operations by allowing developers to interact with databases using JavaScript objects instead of raw SQL queries. The framework provides comprehensive features for data modeling, relationships management, and migrations. Its architecture supports complex data relationships through associations, while also providing transaction support and connection pooling for optimal performance. Sequelize's validation and hooks system enables data integrity maintenance and custom business logic implementation at the model level.

6) FastAPI

Fig. 6. FastAPI

FastAPI is a modern, fast, and efficient web framework for building APIs with Python. It is designed to simplify API development by leveraging Python 3.7+ type hints for better code validation and readability. Built on Starlette and Pydantic, FastAPI provides high performance that rivals frameworks like Node.js and Go. One of its standout features is its ability to automatically generate interactive API documentation using OpenAPI standards, with tools like Swagger UI and ReDoc included by default. FastAPI also supports asynchronous programming with async and await, making it well-suited for high-concurrency applications. Developers benefit from built-in dependency injection, automatic data validation, and serialization, reducing boilerplate code and minimizing potential bugs. With strong type checking, FastAPI ensures that request and response data match the defined structure, enhancing reliability and maintainability. This framework is particularly popular for creating RESTful APIs, deploying machine learning models, and developing microservices due to its speed, simplicity, and robust features.

7) Flask

Fig. 7. Flask

Flask is a Python web framework that serves as a crucial bridge between NUGU PlayKit and backend services. In NUGU-integrated applications, Flask handles

incoming webhook requests from the NUGU platform, processes intent and utterance data, and returns appropriate responses through REST APIs. Its routing capabilities enable developers to create distinct endpoints for different NUGU actions, while its request handling makes it ideal for parsing JSON payloads from NUGU cloud services. Flask's ability to maintain session states and handle asynchronous requests makes it particularly suitable for managing ongoing conversations between users and NUGU speakers, while its extensibility allows for easy integration with databases and external services needed for NUGU skill development.

3 Libraries

1) Hugging Face Transformer Library

Hugging Face Transformer Library is a powerful and widely used library for natural language processing (NLP) tasks. It provides a comprehensive suite of pre-trained transformer models, including popular architectures like BERT, GPT, and T5, which are designed to handle tasks such as text classification, translation, summarization, question answering, and more. One of the key strengths of the library is its user-friendly API, which allows developers to easily integrate these models into applications with minimal setup. Additionally, it supports fine-tuning, enabling users to adapt pre-trained models to their specific datasets and tasks for improved performance. The library is highly versatile, offering compatibility with popular deep learning frameworks like PyTorch and TensorFlow, as well as extensive documentation and community support. Hugging Face has become a cornerstone for both research and industry applications, simplifying the implementation of state-of-the-art NLP solutions.

2) PyTorch

PyTorch is an open-source machine learning library that has gained immense popularity among researchers and developers. Its dynamic computational graph allows for more intuitive and flexible model building, enabling developers to write complex architectures without sacrificing speed or performance. PyTorch's ecosystem includes a wide range of tools and libraries that support deep learning workflows, from model training and evaluation to deployment in production environments. The strong community support and active development ensure that PyTorch remains at the forefront of deep learning innovation.

3) PEFT (Parameter-Efficient Fine-Tuning) Library

PEFT (Parameter-Efficient Fine-Tuning) Library is a specialized library designed to make the process of fine-tuning large pre-trained models more efficient and cost-effective. It provides a framework to fine-tune only a subset of a model's parameters, significantly reducing computational requirements and memory usage compared to traditional fine-tuning methods. This approach is particularly useful when working with resource-constrained environments or when handling massive models. PEFT supports techniques like LoRA (Low-Rank Adaptation), Prefix-Tuning, and Adapter Tuning, enabling users to adapt models to specific tasks without needing to update the entire set of model parameters. By focusing on task-specific adjustments and retaining most of the pre-trained model's knowledge, PEFT allows for faster and more scalable model deployment. The library is designed to integrate seamlessly with the Hugging Face ecosystem, making it easy to combine with popular transformer models for a wide range of applications such as text generation, classification, and summarization. With PEFT, developers and researchers can achieve competitive results with significantly lower resource demands, making it an essential tool for modern NLP workflows.

4) Google Cloud Speech-to-Text

Google Cloud Speech-to-Text is a powerful speech recognition service that converts audio to text by applying neural network models. The service supports real-time streaming or pre-recorded audio processing across multiple languages and variants. It provides features like automatic language detection, speaker diarization, and profanity filtering. The service achieves high accuracy through its machine learning models trained on a vast array of audio data, making it particularly effective for handling different accents and background noise. Its API integration allows developers to easily incorporate voice recognition capabilities into their applications while maintaining scalability and reliability.

5) Hume AI Sentiment Analysis

Hume AI Sentiment Analysis is an advanced emotion recognition service that provides detailed analysis of human expressions and emotions. The service uses sophisticated AI models to analyze voice recordings and identify various emotional characteristics including valence, arousal, and specific emotional states. Its comprehensive API offers granular insights into emotional patterns and intensities, enabling applications to understand and respond to user emotions effectively. The service supports both real-time analysis and batch processing, making it versatile for various use cases from user experience research to interactive applications.

6) NUGU Play Kit

NUGU PlayKit is SKT's development kit for their AI platform NUGU. This SDK enables developers to create custom services that interact with NUGU speakers through voice commands. It provides voice

command processing, natural language understanding, and dialogue management capabilities, communicating with the NUGU cloud via REST APIs. The PlayKit allows seamless integration of third-party services into the NUGU ecosystem without requiring complex voice interface implementations.

4 Cloud Platform

1) Google Cloud Platform (GCP) for Speech-To-Text

Google Cloud Platform (GCP) provides speech recognition capabilities through its Speech-to-Text service, which operates in a scalable cloud environment. The service leverages GCP's infrastructure to handle audio processing tasks efficiently, offering features like real-time transcription and multiple language support. The speech recognition capabilities are accessible through robust APIs and deliver reliable performance. The platform's security features and compliance standards ensure the protection of audio data during processing.

2) AWS EC2 (Elastic Compute Cloud)

AWS EC2 is a core component of Amazon's cloud computing platform that provides scalable computing resources in the cloud. This web service enables flexible deployment of virtual servers, offering secure and resizable compute capacity for running applications and services. EC2 allows users to select from various instance types optimized for different use cases, including compute-optimized, memory-optimized, and general-purpose instances. It provides robust features such as auto-scaling to handle varying workloads, load balancing for distributed traffic, and security through Amazon Virtual Private Cloud (VPC). The service operates on a pay-as-you-go pricing model, allowing cost optimization by paying only for the compute capacity used. With its high reliability and extensive global infrastructure, EC2 maintains consistent performance while offering the flexibility to scale resources up or down based on demand.

3) AWS SageMaker

Amazon SageMaker is a comprehensive, fully managed service offered by AWS that streamlines the process of building, training, and deploying machine learning models at scale. It provides an integrated environment with tools for data preparation, model development, and evaluation, supporting popular ML frameworks like TensorFlow, PyTorch, and Scikit-learn. The platform includes features such as automatic model tuning and distributed training for efficient development, while offering flexible deployment options through real-time endpoints and batch inference. SageMaker integrates with other AWS services for storage, security, and data pipelines, allowing organizations to focus on building ML models while benefiting from a secure, scalable platform.

5 Development Environment

1) On Local Machine

TABLE II
ON LOCAL MACHINE

Name	***Computer Resource***
Song Jaehwi (AI)	Apple M2 16 GB RAM
Yu Hyeonseo (Back-end)	Apple M1 Pro 16 GB RAM
Lee Seungjin (Data-Engineering)	Windows Intel i5-1240P 12th Gen 16GB RAM
Choi Hyungrak (Front-end)	Apple M2 8 GB RAM

2) Cloud Platform

TABLE III
CLOUD PLATFORM

Purpose	***Computer Resource***
AI Model training	AWS SageMaker (ml.g5n.xlarge) NVIDIA A10G GPU 24 GB / 4 vCPUs 16 GB RAM
AI server deployment	AWS EC2 (g4dn.xlarge) NVIDIA T4 GPU 16 GB / 4 vCPUs 16 GB RAM Linux Ubuntu 20.04
Back-end server deployment	AWS EC2 (t2.micro) 1 vCPU 1GB RAM Ubuntu 20.04

B. Software in use

1 Visual Studio Code

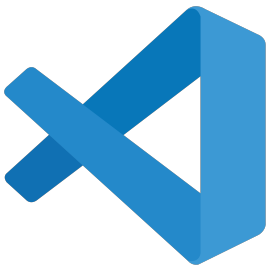

Fig. 8. Visual Studio Code

Visual Studio Code (VS Code) is a powerful, open-source source code editor developed by Microsoft. It supports a wide array of programming languages and is highly extensible through its rich marketplace of plugins and extensions. VS Code offers an integrated terminal, debugging tools, and Git version control capabilities, making it an ideal environment for collaborative projects.

Features like IntelliSense provide smart code completion and suggestions, which enhance productivity. The user-friendly interface and customizable settings allow developers to tailor their workspace according to their preferences, thereby facilitating efficient coding practices.

2 Vim

Fig. 9. Vim

Vim is a highly efficient, keyboard-centric text editor that is favored by many developers for its speed and flexibility. Vim's modal editing system, which distinguishes between different modes (such as insert and command mode), allows for rapid text manipulation and navigation. While it has a steeper learning curve compared to traditional editors, many developers find that it significantly enhances their coding speed and efficiency once mastered. Vim is particularly useful for making quick edits across multiple files or when working in a terminal-based environment, making it a valuable tool for any developer's toolkit.

3 Figma

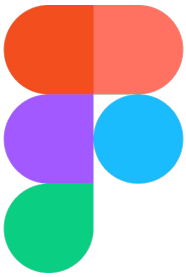

Fig. 10. Figma

Figma serves as our project's core tool for UI/UX design. Using the latest web-based version, it features real-time collaboration capabilities. It offers the advantage of managing the entire design process, from wireframe creation to detailed prototype production, on a single platform. Notably, through its design system management functionality, we can efficiently manage consistent UI elements and easily extract CSS values and assets during the transition from design to development. Additionally, its component-based design system aligns well with React Native's component structure, creating a seamless workflow from design to development.

4 Xcode

Fig. 11. Xcode

Xcode 15.2 is an essential development environment for iOS application development. This integrated development environment provided by Apple offers real-time testing capabilities through iOS simulator and provides a complete toolkit for building and deploying iOS applications. It plays a crucial role particularly in the iOS build generation process for React Native applications and enables iOS-specific debugging tasks. Through built-in performance analysis tools, we can optimize application performance, and it provides testing environments for various iOS devices, enabling stable application development.

5 MySQL

Fig. 12. MySQL

MySQL is a widely-used open-source relational database management system, implemented with version 8.0. It offers powerful data management capabilities including complex queries, transaction processing, and stored procedures with strong ACID compliance. The system features built-in replication support, high availability, and robust security mechanisms including encrypted connections and role-based access control. MySQL's efficient indexing, caching, and query optimization features enable high-performance data operations at scale, making it a standard choice for applications requiring reliable data persistence.

6 Jira

Fig. 13. Jira

Jira is Atlassian's project management and issue tracking platform designed for agile software development teams. It provides customizable workflows, sprint planning tools, and real-time reporting dashboards to track project progress. Teams can create and assign tasks, track bugs, manage product backlogs, and monitor development cycles through its intuitive interface. Jira's integration capabilities with other development tools like GitHub, Bitbucket, and Confluence enhance its functionality, while its customizable Kanban and Scrum boards help visualize work progress. The platform also offers advanced features for time tracking, version management, and automated reporting to support efficient project delivery.

7 Github

Fig. 14. Github

GitHub is a web-based Git version control platform that enables code hosting, sharing, and collaboration. It provides essential features like pull requests, issue tracking, and code reviews for team coordination. The platform includes automated workflows through GitHub Actions, documentation tools, and project management capabilities. Developers can work simultaneously using branches and forks, then merge changes via pull requests. GitHub offers extensive integration options with development tools and security features like dependency scanning. Its widespread adoption makes it fundamental to modern software development, serving individual developers to large enterprises.

8 Overleaf

Fig. 15. Overleaf

Overleaf is an online LaTeX editor providing real-time document creation and collaborative editing features. It offers extensive LaTeX templates and instant PDF preview functionality. The platform includes syntax highlighting, auto-completion, and error checking tools alongside version control capabilities. Overleaf integrates with reference management tools and provides a complete LaTeX environment with built-in compilation, eliminating local installation needs. The service includes comprehensive documentation and tutorials to support users in LaTeX document creation.

9 Notion

Fig. 16. Notion

Notion is a collaborative workspace that combines document management, note-taking, and project organization tools. Its block-based structure allows users to create and arrange various content types within a hierarchical page system. The platform offers templates for different purposes and real-time collaboration features. Notion includes database capabilities, version history, and integration options with external tools. The flexible interface lets users customize their workspace organization and layout to match specific workflows.

10 ChatGPT

Fig. 17. ChatGPT

ChatGPT is an OpenAI language model that functions as a conversational AI assistant and customizable model platform. Based on transformer architecture and extensive training data, it handles diverse tasks from answering questions to coding assistance. The system maintains contextual awareness in conversations and generates responses according to its training. It features multilingual support, adaptable communication styles, and capabilities in summarization, translation, and creative writing. ChatGPT includes safety measures and clearly indicates its knowledge limitations through its training cutoff date.

11 Claude

Fig. 18. Claude

Claude is an Anthropic language model serving as an AI assistant with strong analytical capabilities. Built on transformer architecture, it excels in writing, coding, analysis, and logical reasoning while maintaining conversational context. The model handles complex tasks from document analysis to mathematical problem-solving and coding across various languages. Claude delivers thoughtful responses with appropriate uncertainty acknowledgment and includes safety measures while clearly stating its limitations and knowledge cutoff date.

C. Task Distribution

TABLE IV
ROLE ASSIGNMENTS

Roles	**Name**	**Task description and etc.**
Data Engineer	Lee Seungjin	A Data Engineer designs and implements data collection strategies for smart home automation systems. He specializes in creating comprehensive datasets for AI model training, focusing on diverse situational contexts and device control patterns. His work involves data validation, structuring, and preprocessing to ensure high-quality training data. Through careful curation and organization of smart home routine data, he enables the development of accurate and reliable AI recommendation systems while providing valuable insights into user behavior patterns and automation scenarios.
Back-end Developer	Yu Hyeonseo	A Back-end Developer designs and implements the server-side architecture that powers smart home systems. She builds robust APIs and database structures while establishing seamless integration with AI services for intelligent routine recommendations and automation. Her role involves creating secure endpoints for device control, managing data persistence, and facilitating communication between the front-end and AI components. Through careful system design and implementation of industry best practices, she ensures reliable performance and scalability of the entire backend infrastructure.
AI Developer	Song Jaehwi	An AI Developer develops a multimodal AI service that generates optimal smart home routines based on user circumstances. He is responsible for designing and implementing the AI architecture, from dataset creation to deployment. This involves text-based generation initially, followed by speech recognition with emotion and context analysis. He then utilizes transfer learning techniques for model training and ensures seamless integration with necessary libraries. Additionally, he monitors and fine-tunes the AI system to continually improve its recommendation accuracy and adaptability to diverse user scenarios.
Front-end Developer	Choi Hyungrak	A Front-end Developer creates the user interfaces and interactive elements that enable users to control their smart home environment. He designs intuitive web interfaces using modern frameworks while ensuring seamless communication with back-end services. His work emphasizes responsive design and optimal performance across devices, transforming complex system functionalities into accessible user experiences. Through close collaboration with the development team, he implements features that allow users to effectively monitor and control their connected devices.

IV. Requirement Specification

A. Common Features

1 Signup

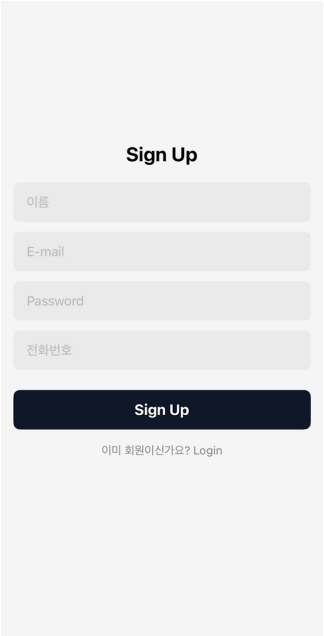

Fig. 19. ID:001, VOICE-Signup-Page

ID	*Name*	**Description**
001	VOICE-Signup-Page	VOICE requires five use information to sign up for membership: E-mail, password, phone number, and name.

ID	*Name*	**Description**
002	VOICE-Signup-Email	Email address is a mandatory field that serves as the user ID for account creation and system access. The input must include '@' and domain, with the system performing an immediate availability check for duplicate emails. Users will receive error messages such as "Please enter a valid email address" for format errors or "This email is already in use" for duplicate entries.
003	VOICE-Signup-Password	The password must consist of a minimum of 8 characters and be a combination of at least 2 types from the following: alphabetic letters, numbers, and special characters. During input, the password should be displayed on the screen in the format of asterisks '****'.
004	VOICE-Signup-Phone number	The mobile phone number is a required field for user verification purposes. If users forget their ID or password, they can regain access to their account using mobile phone authentication.
005	VOICE-Signup-Name	The name must be entered, and will automatically be set as the user's default nickname upon first login. The name cannot be changed after it is entered.

2 Login

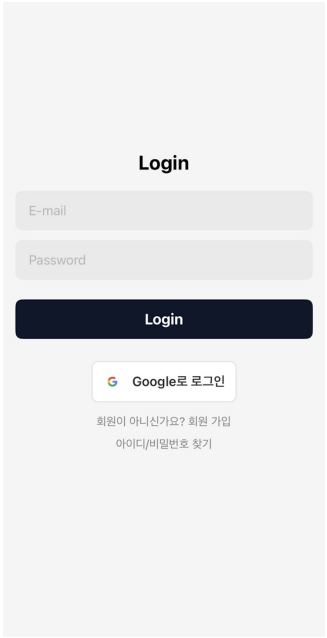

Fig. 20. ID:006, VOICE-Login-Page

ID	*Name*	**Description**
006	VOICE-Login-Page	The login page provides users with essential account management features including Sign Up for new accounts, Log In for existing users, ID Recovery, Password Reset, and Social Media Login options.

ID	*Name*	**Description**
007	VOICE-Login-Types	VOICE offers two login options: (1) Local login through VOICE membership, (2) Google SNS (Social Networking Service) login.

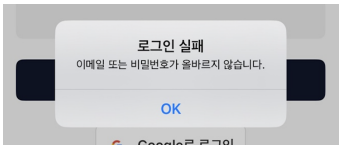

Fig. 21. ID:010, VOICE-Login-Local Failure 1

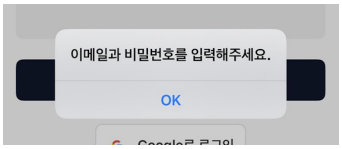

Fig. 22. ID:010, VOICE-Login-Login Failure 2

ID	*Name*	**Description**
008	VOICE-Login-Local login	The system will verify that both the ID and password fields are filled. If either field is left blank, a warning message will be displayed.
009	VOICE-Login-Local Success	When both ID and password are correctly entered and match an existing entry in the user database, the login is successful. The user is then redirected to the main homepage.
010	VOICE-Login-Local Failure	If the entered login information does not match any existing record in the user database, the system will deny access and display a "User not found" message. Additionally, an alert appears when either field is empty.

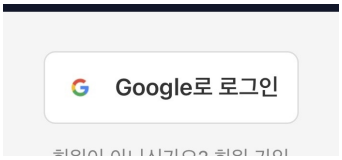

Fig. 23. ID:011, VOICE-SNS-Login

ID	*Name*	**Description**
011	VOICE-Login-SNS Login	This option uses the authentication APIs provided by Google.
012	VOICE-Login-SNS Success	Upon successful authentication through social media, the system must receive the user's name and phone number. Following this data retrieval, the user is directed to the main page.

3 Account Recovery

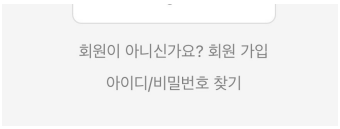

Fig. 24. ID:013, VOICE-FindID-Page

ID	*Name*	**Description**
013	VOICE-FindID-Page	This feature assists users who have forgotten their account ID (email address). In VOICE's system, the email address serves as the user ID for account login and identification. In case of a forgotten ID, account recovery is possible through entering the registered phone number as an alternative verification method.

ID	*Name*	**Description**
014	VOICE-FindID-Phone Number	Users who have forgotten their ID (email) are asked to provide their registered phone number. The system accepts it as an input for ID recovery.
015	VOICE-FindID-Verification	The system then checks if the provided phone number exists in the user database. If the input phone number does not exist in the user database, an error message will appear: "Please double-check your phone number and try again".
016	VOICE-FindID-Recovery	If the phone number is verified, the system retrieves the associated user ID (email address). Subsequently, the system displays or sends the email address (ID) to the user through a secure method.

4 Reset Password

ID	*Name*	**Description**
017	VOICE-Reset Password-Page	To initiate a password reset, the user must provide their ID, which is the email address they designated as their user ID during the registration process.

ID	*Name*	**Description**
018	VOICE-Reset Password-Verification	The system then checks if the input user ID exists in the user database. If the input ID does not exist in the user database, an error message will appear: “Please double-check your email and try again”.
019	VOICE-Reset Password-Verification Success	When the input ID exists in the user database, the system receives the user’s name, date of birth, and phone number, and goes through the process of verifying whether the user is correct through authentication by the carrier.
020	VOICE-Reset Password-New Password	Upon successful verification, users must enter a new password with minimum 8 characters combining at least 2 types from letters, numbers, and special characters. The password input is displayed as asterisks for security.

B. In Application

1 Entry

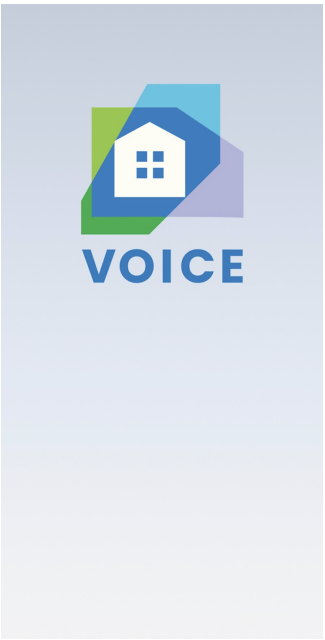

Fig. 25. ID:021, VOICE-Entry-Splash Screen

ID	*Name*	**Description**
021	VOICE-Entry-Splash Screen	A splash screen is displayed for 1.5 seconds when launching the app, featuring the VOICE logo while loading essential resources and providing visual feedback to users.
022	VOICE-Entry-First Time Users	When users visit the app for the first time, they are directed to the registration page. After registering and logging in, they see the splash screen. After 1.5 seconds, the main page appears.
023	VOICE-Entry-Existing Users	When users open the app, the splash screen appears first. After 1.5 seconds, the main page appears.

2 Mainpage

Fig. 26. ID:024, VOICE-Mainpage

ID	*Name*	**Description**
024	VOICE-Mainpage	The main page should be expressed with maximum simplicity. It must feature the VOICE background image, accompanied by a welcome message stating “[User Name]’s Home” and include a menu bar. The menu bar consists of (1) Home, (2) AI’s Pick, (3) Application Management, and (4) Mypage. Clicking on each menu item navigates to that corresponding page.

3 AI's Pick

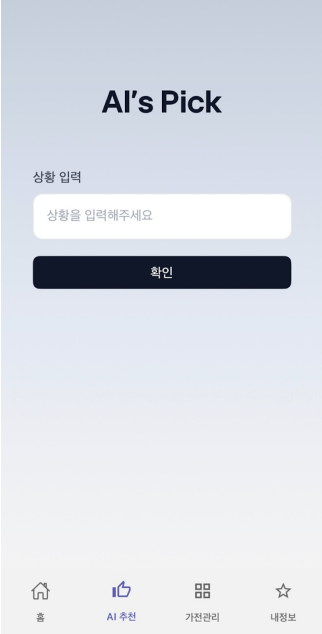

Fig. 27. ID:025, VOICE-AI's Pick-Page

ID	*Name*	**Description**
025	VOICE-AI's Pick-Page	When 'AI's Pick' is selected from the menu bar, a field for entering the current situation appears. Users must input their current situation as text.

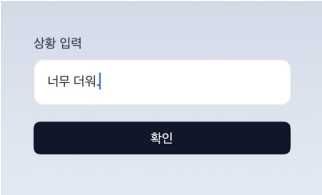

Fig. 28. ID:026, VOICE-AI's Pick-Format Conversion

ID	*Name*	**Description**
026	VOICE-AI's Pick-Format Conversion	The system processes the input by appending it to a predefined template, which includes contextual data like the user's registered smart devices. This structured approach ensures consistency in input formatting for the AI model.

ID	*Name*	**Description**
027	VOICE-AI's Pick-Generative AI Model	The model, trained on tens of thousands of routines from state-of-the-art generative AI APIs, processes the formatted input to generate personalized routine recommendations based on the user's specific situation.
028	VOICE-AI's Pick-Backend Processing	The system processes the AI-generated response into a structured JSON format that details specific instructions for each smart home device. The structured data enables both seamless control of the smart home system and efficient storage in the database, while maintaining consistency in how routines are executed across different device types.

Fig. 29. ID:029, VOICE-AI's Pick-Loading

ID	*Name*	**Description**
029	VOICE-AI's Pick-Loading	While the system processes the response data for display, a loading screen is shown to the user.

Fig. 30. ID:030, VOICE-AI's Pick-Display

ID	*Name*	**Description**
030	VOICE-AI's Pick-Display	The system then presents a comprehensive routine detailing specific operations for each appliance. Users are given options to accept, reject, or request a different recommendation.

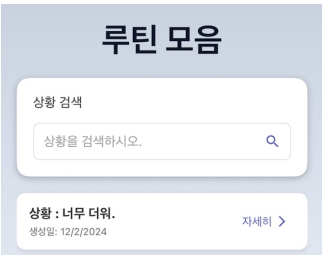

Fig. 31. ID:031, VOICE-AI's Pick-Accept

ID	*Name*	**Description**
031	VOICE-AI's Pick-Accept	When users accept the recommendation, the system executes the routine by sending control signals to the appropriate devices, stores the successfully executed routine in the Routine History database, and redirects to the AI's Pick initial screen.

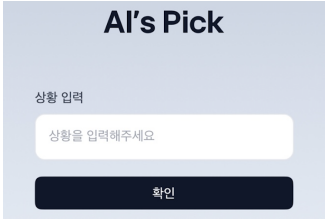

Fig. 32. ID:032, VOICE-AI's Pick-Reject

ID	*Name*	**Description**
032	VOICE-AI's Pick-Reject	If users reject the recommendation, the system returns to the AI's Pick initial screen without storing any data.

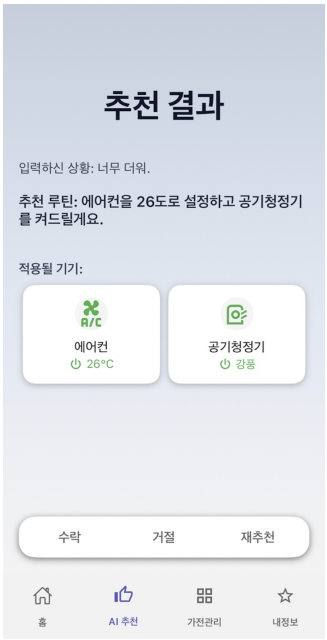

Fig. 33. ID:033, VOICE-AI's Pick-New Request

ID	*Name*	**Description**
033	VOICE-AI's Pick-New Request	When users request a different recommendation, the system maintains the original situation input but generates a new routine recommendation through the AI model, showing the loading screen while processing.

4 Appliance Management

Fig. 34. ID:034, VOICE-Appliance Management-Page

ID	*Name*	**Description**
034	VOICE-Appliance Management-Page	Upon entering the appliance management page, users can view all appliances managed in the VOICE system. This page includes buttons for registering new appliances and accessing detailed pages for managing each appliance. When a recommended routine is accepted, the device status should be automatically updated and the devices should operate. The same applies when executing a past routine from the routine collection.

Fig. 35. ID:035, VOICE-Appliance Management-Registration

ID	*Name*	**Description**
035	VOICE-Appliance Management-Registration	Users can register appliances here, with all changes reflected in the database in real time.

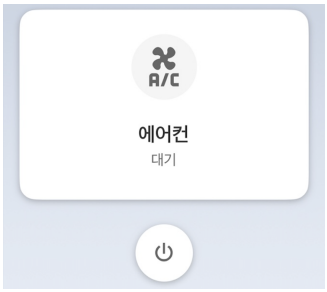

Fig. 36. ID:036, VOICE-Appliance Management-Control

ID	*Name*	**Description**
036	VOICE-Appliance Management-Control	Users can manually control appliances with options such as turning them on or off. These controls provide flexibility for real-time adjustments to suit user needs.

5 Mypage

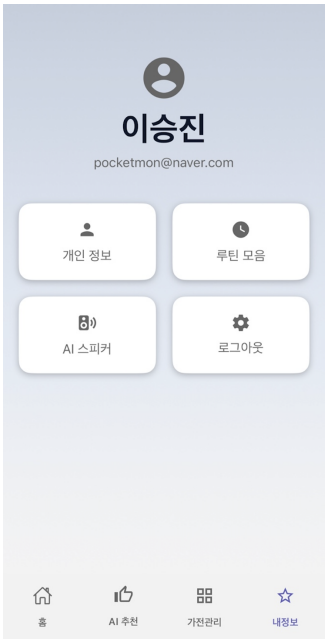

Fig. 37. ID:037, VOICE-Mypage

ID	*Name*	**Description**
037	VOICE-Mypage	This page serves as the user's personal dashboard. The Mypage screen shows the user's name and email address at the top, with four buttons: (1) Personal Information (2) Routine Collection (3) AI Speaker (4) Logout

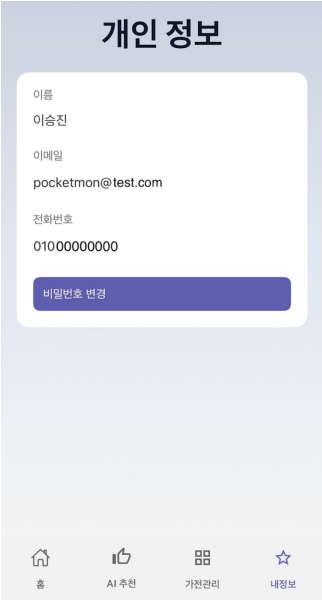

Fig. 38. ID:038, VOICE-Mypage-Profile-Page

ID	*Name*	**Description**
038	VOICE-Mypage-Profile-Page	This page allows users to view their personal data. It displays the user's name, email, phone number, and includes a change password button.

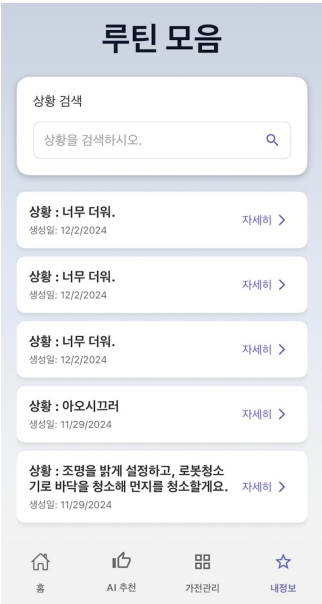

Fig. 39. ID:039, VOICE-Mypage-Routine History-Page

ID	*Name*	**Description**
039	VOICE-Mypage-Routine History-Page	When entering the Routine History page, there's a search field at the top where users can search for situations. Below this, all routines that the user has previously accepted are displayed in a list format showing 'Situation, Creation Date'. Each routine has a 'View Details' button, which when clicked, allows users to see a detailed view of the executed routine.

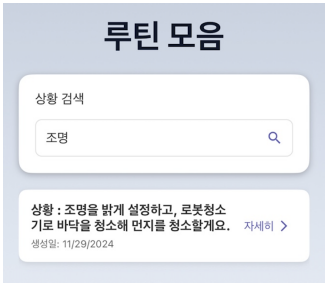

Fig. 40. ID:040, VOICE-Mypage-Routine History-Search

ID	*Name*	**Description**
040	VOICE-Mypage-Routine History-Search	Users can search for specific routines by entering situation keywords. The search results will only display routines whose situations match two or more characters of the search text.

Fig. 41. ID:041, VOICE-Mypage-Routine History-Select

ID	*Name*	**Description**
041	VOICE-Mypage-Routine History-Select	Clicking the 'View Details' button expands the view to display selected routine's information: situation, routine description, list of devices to be modified, and buttons for execution and deletion.

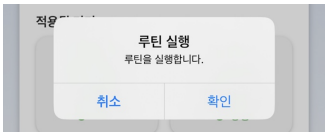

Fig. 42. ID:042, VOICE-Mypage-Routine Collection-Select-Execute

ID	*Name*	**Description**
042	VOICE-Mypage-Routine Collection-Select-Execute	When the execution button is pressed, an alert will appear notifying that the routine will be executed. Pressing the execute button will initiate the routine and return to the routine history page.

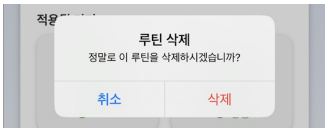

Fig. 43. ID:043, VOICE-Mypage-Routine Collection-Select-Delete

ID	*Name*	**Description**
043	VOICE-Mypage-Routine Collection-Select-Delete	When the delete button is pressed, an alert will appear confirming the deletion of the routine from the routine history collection. The routine will also be removed from the database. After deletion, the user is redirected to the routine history page with the routine no longer displayed.

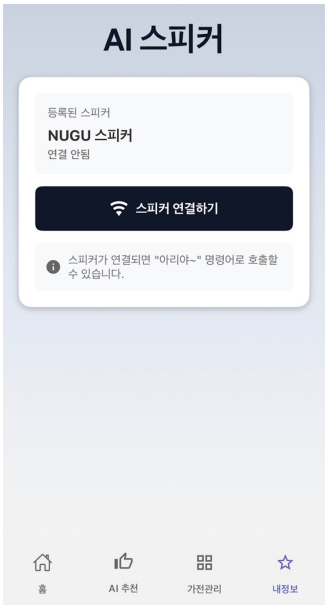

Fig. 44. ID:044, VOICE-Mypage-AI Speaker-Page

ID	*Name*	**Description**
044	VOICE-Mypage-AI Speaker-Page	When the 'Find AI Speaker' button is pressed, the system detects available AI speakers. Users can select a detected device and press the 'Connect' button to establish a connection.

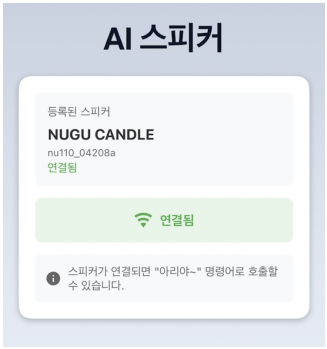

Fig. 45. ID:045, VOICE-Mypage-AI Speaker-Connect

ID	*Name*	**Description**
045	VOICE-Mypage-AI Speaker-Connect	Upon successful speaker integration, the system displays the AI speaker's serial number along with a "Connection Complete" message.

C. With AI Speaker

1 User-Initiated Interaction

1) Initial Interaction

When a user says "[Speaker Name] start," the speaker responds with, "Hello, I'm [Speaker Name]." Following this, users can describe their situation or intended action, and the system leverages natural language processing to facilitate seamless, conversational interactions instead of relying on fixed commands. This approach enables users to maintain a natural flow of dialogue after the initial prompt.

2) Voice Processing and Analysis

The user's voice is converted into a WAV file and sent to the AI server. On the server, this file undergoes multi-layered voice analysis: Google STT handles speech-to-text conversion, while Hume AI evaluates the user's emotional state. These analyses are then integrated into a unified input format for the AI model, combining the textual content with the emotional context to provide comprehensive understanding of the user's request.

3) AI Model Processing

The AI model, trained on ten thousands of 'situation': 'routine' pairs, processes the integrated information to determine the most appropriate routine. The system evaluates multiple contextual factors simultaneously, referencing the user's previous preferences, routine history, and detailed status of appliances currently registered in VOICE. Through this comprehensive analysis, the system selects a routine that best aligns with the user's current situation.

4) Response Processing

After analysis and routine selection, the system structures device-specific commands for each applicable smart home device. These commands are organized in a format optimized for the smart home control system, ensuring precise execution of the selected routine.

5) Voice Response

The system uses Text-to-Speech (TTS) technology to generate natural language responses that clearly communicate the suggested routine while maintaining conversational flow. The system first announces the selected routine through the speaker, and only after confirmation or a brief pause, executes the structured device commands.

6) Data Recording

After routine execution, the system records the initial request along with the emotional analysis (converted into corresponding emojis for efficient storage and processing), the system-suggested routine content, and detailed information about the actions executed on each applicable appliance in the database. This systematic data collection and storage process continuously enhances the system's machine learning capabilities, enabling more accurate and personalized routine recommendations in future interactions.

2 Continuous Conversation Tracking by AI Speaker

Beyond standard user-triggered interactions, this advanced mode leverages continuous audio processing to anticipate user needs and provide proactive routine suggestions, creating a more intuitive and responsive smart home experience.

1) Ambient Monitoring

Through Speech-to-Intent technology, the system maintains constant awareness of surrounding conversations and sounds. It analyzes speech patterns, contextual cues, and environmental signals to identify situations where a routine suggestion might enhance the user's experience.

2) Voice Processing and Analysis

Upon detecting and understanding specific user situations in real-time, the system initiates comprehensive audio processing. The audio undergoes dual analysis - converting speech to text via Google STT while simultaneously extracting emotional nuances using Hume AI. Both analyses are synthesized into a single comprehensive input format for further processing.

3) AI Model Processing

The unified analysis feeds into the AI model, which correlates the detected scenario with potential routines. Drawing from historical preferences, past interactions, and current device configurations in VOICE, the model determines the most contextually appropriate routine recommendation.

4) Response Processing

Following routine selection, the system translates the chosen actions into a series of precise instructions. These are formatted according to each smart device's specific requirements while maintaining the cohesive execution of the entire routine.

5) Voice Response

Leveraging TTS technology, the system generates contextually appropriate responses, framing suggestions in a conversational manner such as "Would you like me to [suggested action]?" or "I'll [suggested action] for you." Before initiating any actions, it clearly communicates the intended routine through voice output.

6) Data Recording

Each interaction enriches the system's knowledge base, storing detected scenarios, emotional contexts (represented as emojis), recommended routines, and device

execution details. This accumulated data drives continuous improvements in situation recognition and routine suggestions.

D. AI Model

1 Dataset Acquisition Using Generative AI Model APIs

1) Dataset Acquisition Using ChatGPT API

A comprehensive dataset of 10,000 unique entries was constructed using multiple state-of-the-art generative AI models (ChatGPT, Microsoft Copilot, Google Bard, and Claude) to document diverse smart home scenarios and their corresponding routine recommendations. The dataset entries were structured with systematic situation descriptions and associated automation suggestions, ensuring organized data categorization. A rigorous validation protocol was implemented to eliminate duplicate responses and unrealistic routines, maintaining high quality and accuracy standards. For specialized or rare scenarios, detailed prompt engineering techniques were employed to ensure comprehensive coverage. This methodical curation process, leveraging multiple AI models, provided diverse perspectives and interpretations of home automation scenarios, establishing a robust foundation for generating accurate and practical recommendations. The development of this extensive, validated dataset of 10,000 entries demonstrates a systematic approach to ensuring high-quality, reliable recommendations across a wide spectrum of user scenarios.

2) Model Fine-Tuning with PEFT and the Transformer Library

Once the dataset is established, the next step is to utilize the Transformer library to implement the paust/pko-chat-t5-large model. Fine-tuning is conducted using PEFT (Parameter-Efficient Fine-Tuning) with the LoRA (Low-Rank Adaptation) technique on AWS SageMaker. This approach allows for efficient fine-tuning of the model on the previously collected dataset by updating only a small subset of parameters, significantly reducing computational overhead while maintaining performance. Through this process, the model is enhanced to better understand the nuances of the input situations and generate more accurate smart home routine recommendations. By leveraging LoRA-based fine-tuning, the model is tailored specifically to the dataset, improving its contextual understanding and prediction accuracy for optimal routines in diverse scenarios.

3) API Development with FastAPI

After fine-tuning the model, the final step is to integrate it into a FastAPI-based AI server deployed on an AWS EC2 instance. This AI server is designed to handle incoming text inputs describing the user's situation. Upon receiving an input through POST request, the server concatenates the input situation with existing input template and processes it through the trained model to generate and return the most suitable smart home routine recommendations via an API. By leveraging FastAPI's high performance and modern features, the application ensures efficient and seamless interactions, enabling users to input their situations in natural language and receive instant, context-aware suggestions for smart home routines.

2 Expansion into Voice Domain Using SKT NUGU (AI Speaker)

In order to increase the modality by expanding domain from text to audio, VOICE adopted AI speaker to convert human voice into text with additional context information extracted from human voice.

1) Collect Voice Data and Convert to Text

When a user speaks to the NUGU device, the spoken input is captured as an audio file and sent to the AI server. The AI server processes the audio file using the Google Speech-to-Text (STT) API to convert the spoken voice into text. This integration enables seamless communication by transforming natural speech into text data that can be further utilized for processing and generating smart home routine recommendations.

2) Extract Emotion and Context Information

After the AI server receives the user's voice audio file, the process doesn't stop at simply converting the audio into text. Instead, the audio file itself is further analyzed to extract additional information about the user's emotional state and urgency. This analysis is performed using Hume AI, which processes the audio file to detect emotional nuances and urgency levels based on factors like tone, pitch, and intensity. The resulting emotion and urgency information are then appended to the text as tags (e.g., "(emotion)"), enhancing the contextual understanding of the user's statement. This enriched input, containing both the textual and emotional context, is then used as input for the model, enabling it to provide smarter and more context-aware routine recommendations.

3) Model Processing and Recommendation

When the situation information, enriched with emotional and urgency cues, is received as input, the server further processes it to enhance the model's understanding. Specifically, the server concatenates the input situation with a predefined input template, structuring the information in a way that aligns with the context the model was trained on. This final, structured input is then fed into the fine-tuned model. The model, now equipped to better understand the user's situation, generates personalized routine recommendations

as output. To provide more actionable insights, the server uses the Claude API to parse the generated routines by appliance type. The routines are then organized into a structured JSON format, detailing the specific actions for each device. The original output and JSON response is then sent to the Node.js backend server, which ensures the results are ready for seamless integration and execution.

4) Response Delivery

The output generated by the model is transformed into speech using NUGU's Text-to-Speech (TTS) functionality. This enables the system to deliver the recommended routines to the user in spoken form, providing auditory feedback on the recommended routines.

V. Architecture Design

A. Overall Architecture

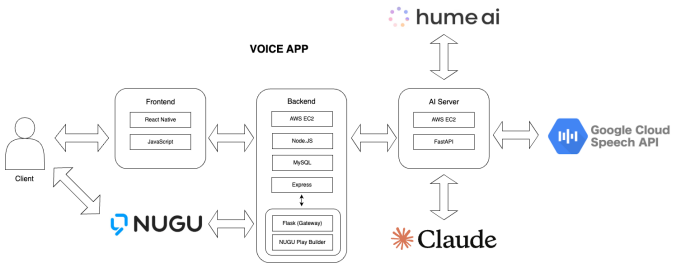

Fig. 46. Overall Architecture

VOICE users interact with our system through the Frontend application or NUGU interface. To achieve this, the system consists of 4 main modules: Frontend, Backend, AI Server, and NUGU Playbuilder.

The first module is "Frontend," developed with React Native and JavaScript. This module serves as the primary user interface for VOICE, providing a sophisticated platform for both text and voice-based smart home control. Built with cross-platform compatibility in mind, it integrates seamlessly with the NUGU SDK for voice recognition while offering an intuitive interface for device management and routine automation. The module implements complex state management through Redux, handles navigation with React Navigation, and maintains persistent data storage using AsyncStorage. Key features include real-time device monitoring, AI-based routine recommendations, comprehensive routine management, AI speaker integration, and personal profile management. Through careful architecture and component design, the Frontend module ensures responsive performance and seamless user experience across all smart home management functions.

The second module is "Backend," built with Node.js and Express on AWS EC2, serving as the central orchestrator of VOICE's functionality. This module manages the critical data flow between user interfaces and AI services while maintaining data integrity through MySQL with Sequelize ORM. It implements sophisticated authentication systems with bcrypt encryption, handles real-time device state management, processes routine recommendations, and manages historical routine tracking. The backend utilizes express-session for state management, incorporates multer for file handling, and employs fluent-ffmpeg for audio processing. Through RESTful API architecture and standardized endpoints, it ensures reliable communication between components while maintaining robust error handling and logging systems. The module's architecture enables efficient scaling and maintenance while providing secure and performant data operations for all system functions.

The third module is the "AI Server," built using FastAPI on AWS EC2. This server integrates multiple AI services to process voice data, analyze emotions, and generate smart home routines. Google Cloud Speech API converts voice input into text, while Hume AI analyzes the audio data to detect emotional patterns and urgency. The detected emotion and urgency information are appended as tags to the converted text, providing richer context for understanding the user's situation. The AI Server then processes this enhanced input using a routine recommendation VOICE model, a fine-tuned paust/pko-chat-t5-large model, which generates personalized smart home routines based on the user's situation. Additionally, the AI Server employs Claude API to parse the generated routines into structured JSON data, categorizing the actions by specific smart devices for seamless execution in the smart home environment. This streamlined approach ensures that all components work in harmony to deliver accurate and actionable recommendations.

The fourth module is "NUGU Playbuilder". This service, developed by SKT, enables voice command interactions for VOICE through NUGU AI assistant. To facilitate communication between NUGU Playbuilder and our backend, we implemented a gateway server using Flask. The server processes various voice commands including routine recommendations, routine execution, and direct device control through defined endpoints and standardized response formats. This integration ensures users can access VOICE's core functionalities through natural voice interactions while maintaining robust error handling and secure data transmission between NUGU and our backend services.

To ensure system reliability and performance, we implemented standardized API endpoints with robust error handling, monitoring, and logging capabilities across all modules - the Frontend interface in React Native, Backend service in Node.js, AI Server in FastAPI, and NUGU Playbuilder integration. This comprehensive approach ensures seamless communication and optimal performance throughout the entire system.

B. Directory Organization

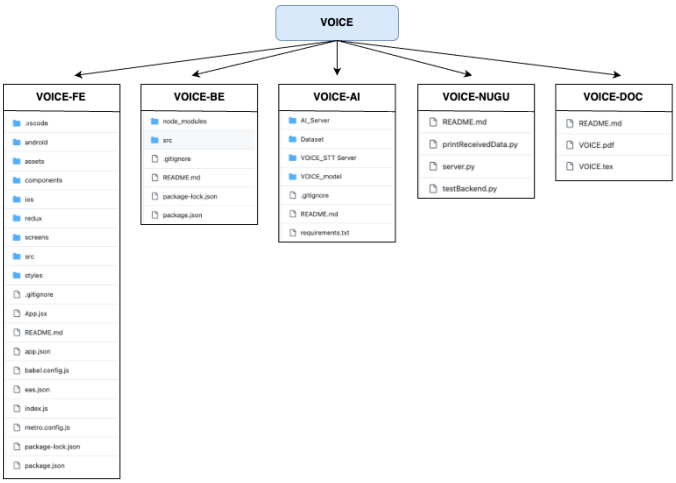

Fig. 47. Directory Organization

TABLE V
DIRECTORY ORGANIZATION-FRONTEND 1

Directory	***File / Folder Name***	***Modules used***
VOICE-FE	.gitignore app.json App.jsx babel.config.js eas.json index.js metro.config.js package-lock.json package.json README.md	expo-font expo-status-bar react react-native react-native-gesture-handler react-redux @expo/vector-icons @react-native-async-storage/async-storage @react-navigation/bottom-tabs @react-navigation/native @react-navigation/stack
VOICE-FE /ios	Podfile	cocoapods expo-modules-autolinking react-native NuguClientKit NuguLoginKit NuguCore NuguAgents

TABLE VI
DIRECTORY ORGANIZATION-FRONTEND 2

Directory	***File / Folder Name***	***Modules used***
VOICE-FE /ios/front	Info.plist nugu-config.plist front-Bridging-Header.h NuguBridge.m NuguBridge.swift NuguConfiguration.swift	OAuthServerUrl OAuthClientId OAuthClientSecret PoCId DeviceTypeCode React/RCTBridgeModule.h React/RCTEventEmitter.h Foundation NuguClientKit NuguLoginKit NuguCore NuguAgents AVFoundation NuguServiceKit Foundation
VOICE-FE /screens	LoadPage.jsx MainPage.jsx	react react-native Platform @expo/vector-icons @react-native-async-storage/async-storage
VOICE-FE /screens AiPick	AiPick.jsx AiPickStack.jsx recommendation.jsx	react react-native react-redux @react-native-async-storage/async-storage @react-navigation/native @react-navigation/stack
VOICE-FE /screens Appliance	Appliance.jsx ApplianceDetail.jsx ApplianceStack.jsx	react react-native react-redux @react-navigation/native @expo/vector-icons @react-navigation/stack
VOICE-FE /screens Auth	Login.jsx SignUp.jsx	expo-auth-session expo-web-browser react react-native react-redux
VOICE-FE /screens /Mypage	ChangePassword.jsx MyPage.jsx MyPageAI.jsx MyPageInfo.jsx MyPageRoutine.jsx MyPageRoutineDetail.jsx MyPageStack.jsx	react react-native react-redux @expo/vector-icons @react-navigation/stack
VOICE-FE /redux	store.jsx	@reduxjs/toolkit
VOICE-FE /redux /slices	applianceSlice.jsx authSlice.jsx routineSlice.jsx speakerSlice.jsx	react-redux @reduxjs/toolkit @react-native-async-storage/async-storage @reduxjs/toolkit
VOICE-FE /components	GradientBackground.jsx	expo-linear-gradient react-native
VOICE-FE /redux	store.jsx	@reduxjs/toolkit
VOICE-FE /redux	store.jsx	@reduxjs/toolkit

TABLE VII
DIRECTORY ORGANIZATION-FRONTEND 2

Directory	*File / Folder Name*	*Modules used*
VOICE-FE /redux /slices	applianceSlice.jsx authSlice.jsx routineSlice.jsx speakerSlice.jsx	react-redux @reduxjs/toolkit @react-native-async- storage/async-storage @reduxjs/toolkit
VOICE-FE /components	GradientBackground.jsx	expo-linear-gradient react-native
VOICE-FE /components /ui	button.jsx DatePickerModal.jsx input.jsx loading-overlay.jsx	react react-native @react-native- community /datetimepicker
VOICE-FE /src	NuguModule.js	react-native
VOICE-FE /src/api	config.js	
VOICE-FE /styles	authStyle.jsx typography.js	react-native
VOICE-FE /assets	logov1.png backgroundimg2.png	
VOICE-FE /assets/fonts /LG-EI_font _all/TTF	LGEIHeadlineTTF LGEITextTTF-Bold LGEITextTTF-Light LGEITextTTF-Regular LGEITextTTF-SemiBold LGEIHeadlineVF LGEITextTTF-Bold	

TABLE VIII
DIRECTORY ORGANIZATION-BACKEND 1

Directory	*File / Folder Name*	*Modules used*
VOICE-BE	node_modules src .gitignore README.md package-lock.json package.json	axios bcryptjs cors dotenv express express-session ffmpeg-static fluent-ffmpeg morgan multer mysql2 sequelize nodemon
VOICE-BE /src/controllers	ai-pick.controller.js appliance.controller.js auth.controller.js index.js mypage.controller.js voice.controller.js	express sequelize axios bcryptjs fluent-ffmpeg ffmpeg-static fs form-data path

TABLE IX
DIRECTORY ORGANIZATION-BACKEND 2

Directory	*File / Folder Name*	*Modules used*
VOICE-BE /src/models	ai-speaker.model.js appliance.model.js index.js routine-history.model.js user.model.js	sequelize dotenv mysql2
VOICE-BE /src/routes	ai-pick.routes.js appliance.routes.js auth.routes.js index.js mypage.routes.js voice.routes.js	express path fs
VOICE-BE /src	app.js	express sequelize express-session cors morgan dotenv mysql2 path

TABLE X
DIRECTORY ORGANIZATION-AI 1

Directory	*File / Folder Name*	*Modules used*
VOICE-AI	AI_Server Dataset VOICE_STT Server VOICE_model README.md requirements.txt	
VOICE-AI /AI_Server	__pycache__ src audio_analysis.py emotion_mapping.py main.py parsing_routine.py	fastapi pydantic transformers torch langchain_core.prompts langchain_core.output_ parsers langchain_anthropic dotenv json tempfile os operator time ffmpeg google.cloud
VOICE-AI /AI_Server /src	google_stt hume	
VOICE-AI /AI_Server /src /google_stt	__pycache__ __init__.py analyer.py config.py	google.cloud.speech google.cloud .language_v1 os typing

TABLE XI
DIRECTORY ORGANIZATION-AI 2

Directory	*File / Folder Name*	*Modules used*
VOICE-AI /AI_Server /src /hume	__pycache__ __init__.py cleint.py	os json requests dotenv
VOICE-AI /Dataset	dataset_construction_ Bard.py dataset_construction_ GPT.py dataset_construction_ Llama.py dataset_construction_ claude.py	langchain_core.prompts langchain_core.output_ parsers langchain_anthropic dotenv pandas typing langchain_openai langchain_llama google.cloud hashlib datetime json
VOICE-AI /VOICE_STT Server	src main.py	time argparse pathlib operator
VOICE-AI /VOICE_STT Server /src	google_stt hume	
VOICE-AI /VOICE_STT Server /src /google_stt	__pycache__ __init__.py analyer.py config.py	google.cloud.speech google.cloud .language_v1 os typing
VOICE-AI /VOICE_STT Server /src /hume	__pycache__ __init__.py cleint.py	os json requests dotenv
VOICE-AI /VOICE_model	LoRa_train_model.py data_preprocessing.py predict_model.py train_model.py	pandas torch tqdm transformers sklearn os peft

TABLE XII
DIRECTORY ORGANIZATION-NUGU

Directory	*File / Folder Name*	*Modules used*
VOICE-NUGU	README.md printReceivedData.py server.py testBackend.py	flask json socket threading requests

C. Module 1: Frontend

1 Purpose

To develop VOICE, React Native was chosen as it supports both iOS and Android platforms as a cross-platform framework. It was selected for its ability to develop applications for both platforms with a single codebase, but later the decision was made to target only iOS applications. The main purpose of the frontend is to provide a user-friendly interface for smart home management. It supports both text and voice interaction, offering an intuitive multimodal experience for controlling smart devices and managing routines. Voice interaction includes features such as saving and displaying recommended routines provided by the speaker and adapting device operations based on the routines.

2 Functionality

The main functionalities of the frontend include user authentication and account management (sign-up, login), smart device registration and control interface, real-time device status monitoring and control, an interface for the AI-based recommendation system (AI's Pick), personal profile and routine management, AI speaker registration and management, and support for multimodal interaction.

3 Location of source code:

https://github.com/CSE-VOICE/VOICE-FE

4 Class component

- App.jsx: This file serves as the main entry point for the React Native-based mobile application. Developed using Expo, it utilizes key libraries like React Navigation, Redux, React Native Gesture Handler, and AsyncStorage to set up the application's fundamental structure.
- NuguModule.js: This file acts as a bridge between React Native and the NUGU SDK, enabling voice recognition features on the iOS platform.
- config.js: Simplifies API requests by providing reusable configurations.
- redux folder: Contains files for managing application state.
- applianceSlice.jsx: Manages the state of smart home appliances using Redux Toolkit. Handles actions such as loading device lists, updating device states, and power control.
- authSlice.jsx: Manages the state for user authentication, including sign-up, login, and Google login, using Redux Toolkit.
- routineSlice.jsx: Manages routines and AI recommendations using Redux Toolkit. Handles user routine data

and interactions with AI-recommended routines.

- speakerSlice.jsx: Manages speaker-related functionalities, such as connecting to AI speakers, using Redux Toolkit.
- store.jsx: Configures the Redux store for state management.
- screens folder: Contains files for individual screens and their components.
- LoadPage.jsx: Displays a loading screen after successful login and before navigating to the main page.
- MainPage.jsx: The primary screen of the VOICE application. Displays current air quality if an air purifier is connected.
- AiPick.jsx: Allows users to input situations and send them to the backend for processing.
- AiPickStack.jsx: Defines stack navigation for the AI Pick feature.
- recommendation.jsx: Displays routine recommendations fetched from the backend in JSON format.
- Appliance.jsx: Shows the list of registered devices and their operational status, along with device registration options.
- ApplianceDetail.jsx: Provides control options for individual appliances.
- ApplianceStack.jsx: Defines stack navigation for appliance-related pages.
- Login.jsx: The login page for VOICE.
- SignUp.jsx: The sign-up page for VOICE.
- MyPage.jsx: The My Page section for VOICE users.
- MyPageAI.jsx: Allows users to connect AI speakers.
- MyPageInfo.jsx: Displays the user's personal information.
- MyPageRoutine.jsx: Lists routines accepted by the user.
- MyPageRoutineDetail.jsx: Provides detailed information about a routine and options to re-execute it.
- MyPageStack.jsx: Defines stack navigation for Mypage.
- nugu-config.plist: This is an iOS Property List (.plist) file that contains configuration details and settings required for the application at runtime. It defines NUGU SDK-related authentication and connection information.
- NuguBridge.swift: Implements an iOS native module to integrate NUGU SDK for voice recognition and keyword detection with React Native. It processes voice data and sends voice commands to the React Native application.
- NuguBridge.m: An Objective-C header file that connects React Native with iOS native modules.
- front-Bridging-Header.h: Used for bridging between React Native and iOS native code (Swift). It enables React Native to call iOS native methods.
- NuguConfiguration.swift: Defines the NuguConfiguration struct, which stores authentication and configuration information related to the NUGU SDK. It centralizes settings for communication with the NUGU server and SDK initialization.
- Podfile: Configures iOS platform dependencies using CocoaPods. It ensures proper integration of React Native and NUGU SDK, along with managing necessary libraries and build settings.

D. Module 2: Backend

1 Purpose

The primary purpose of the backend is to enable smart home device control through optimal routine recommendations. It stores user data, appliance status, and executed routines in a MySQL database using Sequelize ORM, while providing secure API endpoints for frontend interactions. Built using Node.js and Express, it processes voice commands, manages device states in real-time, and integrates with AI server for personalized routine recommendations.

2 Functionality

1) Authentication Management

Implements user registration and login with bcrypt password encryption. Processes both local and SNS (Social Networking Service) authentication through session-based security.

2) Device Management

Manages real-time device operations including power control and state monitoring. Provides APIs for device registration, status updates, and synchronization across the system.

3) AI Routine Management

Processes user situations through ML server communication, handles routine recommendations with session storage, and manages user responses (accept/reject/refresh).

4) Routine History

Records and manages user's executed routines with searchable functionality. Enables replay of previous routines with exact device states and configurations.

5) Voice Processing

Converts voice command files from m4a to wav format using ffmpeg. Implements organized file storage with year/month-based directory structure.

3 Location of source code:

https://github.com/CSE-VOICE/VOICE-BE

4 Class component

- controllers folder: Implements core application logic
- ai-pick.controller.js: Manages routine recommendations through ML server integration. Handles session-based recommendation storage, user responses, and ML server interaction via axios for routine generation, acceptance, rejection, and refresh requests.
- appliance.controller.js: Controls device operations through comprehensive CRUD operations. Manages real-time device states, power control, bulk updates, and state synchronization. Implements error handling and validation for device operations.
- auth.controller.js: Handles user authentication with bcrypt password encryption. Processes local login/signup and validates user credentials. Implements email duplication checks and secure password storage.
- mypage.controller.js: Manages user routine history with search functionality. Handles routine execution requests, provides numbering for display order, and implements routine deletion. Uses Sequelize queries for efficient data retrieval.
- voice.controller.js: Processes voice command files using ffmpeg. Converts m4a files to wav format, manages hierarchical file storage, and analyzes emotional context through FastAPI integration. Implements error handling for file processing failures and maintains routine history.
- index.js: Consolidates controller exports for modular access.
- models folder: Database schema and relationship definitions
- user.model.js: Defines user schema with email validation and password encryption. Includes fields for authentication type and social login integration.
- appliance.model.js: Specifies device schema with state tracking fields. Includes status monitoring, power state, and device metadata management.
- routine-history.model.js: Structures routine execution records. Contains fields for situation text, executed routines, and device state changes.
- index.js: Initializes Sequelize models and defines relationships. Sets up MySQL connection and model associations.
- routes folder: API endpoint definitions
- ai-pick.routes.js: Routes for ML routine recommendations. Endpoints for recommendation requests, retrieval, acceptance, rejection, and refresh.
- appliance.routes.js: Device management endpoints. Handles device listing, status updates, power control, and bulk updates.
- auth.routes.js: Authentication route definitions. Manages signup, login, and session handling paths.
- mypage.routes.js: Personal data and routine history routes. Endpoints for routine listing, execution, and deletion.
- voice.routes.js: Voice processing endpoints for scenario-based analysis. Includes routes for processing voice files through FastAPI integration.
- index.js: app.js: Express application setup. Configures middleware, session handling, CORS, error management, and database connection initialization.

E. Module 3: AI

1 Purpose

The VOICE-AI module is designed to provide smart home routine recommendations based on user's input situation. This module utilizes state-of-the-art AI and natural language processing techniques to analyze user situations and generate actionable routines for smart home devices. By integrating cutting-edge AI models and a fine-tuned recommendation engine, this module ensures an intuitive, context-aware user experience.

2 Functionality

1) Routine Parsing and Recommendation

Processes user inputs (via text or voice) to generate and recommend personalized smart home routines.

2) Model Training and Fine-Tuning

Includes scripts to preprocess datasets, and fine-tune the model using LoRA (Low-Rank Adaptation) for efficient performance.

3) Prediction

Generates real-time routine predictions based on user-provided situations and test the performance.

4) Dataset Management

Provides tools for preprocessing and handling datasets used for training and validation.

5) Dataset Generation

Generates a dataset that covers a wide range of situations, from everyday to specific scenarios, providing detailed contexts and corresponding routines using state-of-the-art generative AI models.

3 Location of source code:

https://github.com/CSE-VOICE/VOICE-AI

4 Class component

- AI_Server folder: Handles backend API services for routine generation and parsing.
- audio_analysis.py: Maps English emotion labels to Korean translations. Provides a comprehensive dictionary of emotional states and their Korean equivalents.
- emotion_mapping.py: Implements audio analysis using Google STT and Hume AI APIs. Processes voice data for speech-to-text conversion and emotion detection with confidence scoring.
- main.py: Main FastAPI server script, managing API endpoints for routine generation.
- parsing_routine.py: Handles parsing of user inputs and interacts with the AI model to generate routines.
- google_stt folder: Contains components for Google Speech-to-Text integration and audio processing configuration.
- analyer.py: Implements Google Cloud Speech-to-Text and Natural Language API for voice transcription and sentiment analysis in Korean language.
- config.py: Defines audio configuration constants for voice processing (sample rate, channels, format).
- hume folder: Contains Hume AI integration components for advanced emotion analysis in audio.
- client.py: Implements Hume AI batch API client for audio analysis with job management and result retrieval.
- VOICE_STT Server folder: Contains the main server components for voice analysis and emotion detection using Google STT and Hume AI.
- main.py: Orchestrates voice analysis workflow combining Google STT and Hume AI services with command-line interface.
- VOICE_model folder: Contains training scripts, data preprocessing utilities, and prediction pipelines using transformer-based models.
- LoRA_train_model.py: Implements LoRA-based fine-tuning for efficient model adaptation.
- data_preprocessing.py: Prepares and preprocesses datasets for training.
- predict_model.py: Handles model inference for routine prediction.
- train_model.py: Script for training and validating the AI model.
- Dataset folder: Contains scripts for generating dataset that covers diverse situations and corresponding routines for fine-tuning the model.
- dataset_construction_Bard.py: A Python script for constructing datasets using Bard, designed to generate diverse scenarios and their corresponding routines with reliability.
- dataset_construction_GPT.py: A Python script for constructing datasets using GPT, designed to generate diverse scenarios and their corresponding routines with reliability.
- dataset_construction_Llama.py: A Python script for constructing datasets using Llama, designed to generate diverse scenarios and their corresponding routines with reliability.
- dataset_construction_Claude.py: A Python script for constructing datasets using Claude, designed to generate diverse scenarios and their corresponding routines with reliability.

F. Module 4: NUGU

1 Purpose

The NUGU Voice Interface Module serves as a dedicated bridge between NUGU AI Speaker and VOICE's backend infrastructure, enabling voice-based control without requiring the frontend application. This module transforms spoken commands into structured API requests, allowing users to seamlessly interact with their smart home system through natural voice interactions.

2 Functionality

By communicating with VOICE's backend through NUGU AI SPEAKER, users can execute key functionalities of VOICE, such as appliance control and routine execution.

3 Location of source code:
https://github.com/CSE-VOICE/VOICE-NUGU

4 Class component

- server.py: A bridge server that handles communication between NUGU Play and VOICE's backend. Processes voice commands into JSON format, manages API requests, and ensures proper data flow between NUGU and VOICE systems.
- testBackend.py: A mock backend server for testing the NUGU integration. Simulates VOICE's backend environment to verify command processing and routing without requiring the actual production server.
- printReceivedData.py: A monitoring utility that logs and displays data received by testBackend.py in real-time. Used for development, testing, and debugging the voice command processing pipeline.

VI. Use Cases

A. Sign Up

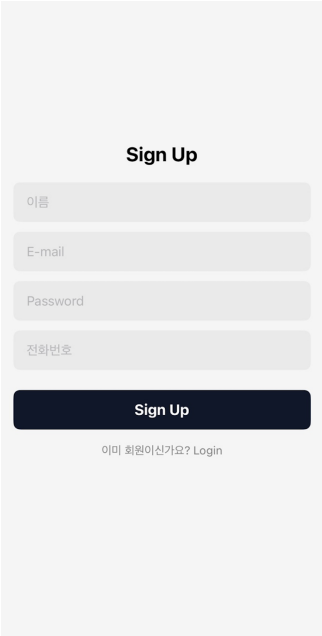

Fig. 48. Signup page

Users can register by entering their basic information: name, email address, phone number, and password. After completing the registration form, they will be taken to the login screen where they can begin using their new account.

B. Login

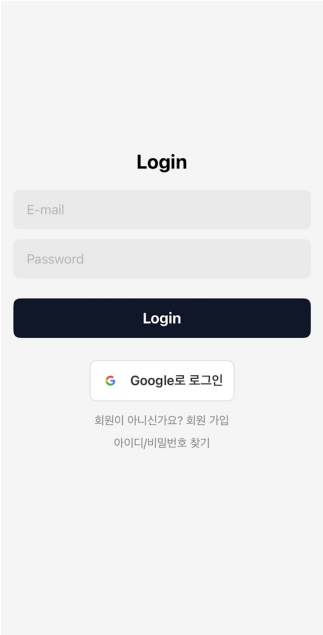

Fig. 49. Login page

A Login Page provides users with access to their accounts by verifying their credentials through ID and password authentication, while also offering the option to sign in using their Google account.

C. Loading

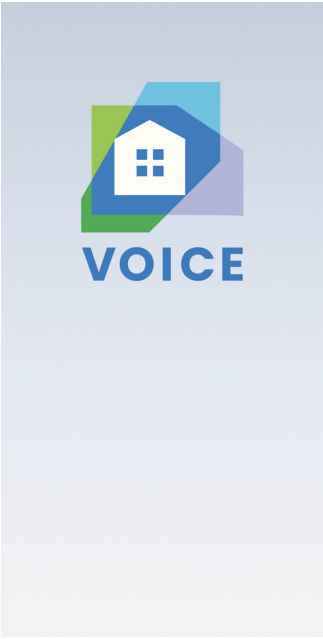

Fig. 50. Loading page

A Loading Page is the initial screen users encounter when launching the app. This screen remains visible only during the preparation of essential components required for the app's operation, and transitions automatically to the subsequent screen once initialization is complete.

D. Main Page

Fig. 51. Mainpage

A Home screen provides essential environmental data from your air purifier, displaying real-time information about your home's temperature, humidity, and fine dust levels. Users can also access the device registration feature to add an air purifier to their system.

E. AI's Pick

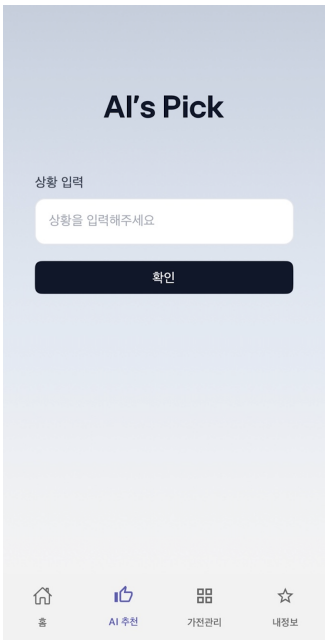

Fig. 52. AI's Pick page

When users tap the "AI's Pick" section from the Mainpage, they are directed to a new page where they can input specific situations or scenarios. After entering their desired conditions, they can proceed by selecting the confirmation button to receive AI-generated recommendations.

Fig. 53. Recommendation result page

1) When users press the confirmation button, they are taken to a recommendation page that displays their inputted situation

along with suggested routines. This page also shows the specific devices that would be involved in the recommended automation. If users are satisfied with the proposed routine, they can accept it. Otherwise, they can reject it or request an alternative recommendation by using the “Recommend again” button.

F. Appliance Management

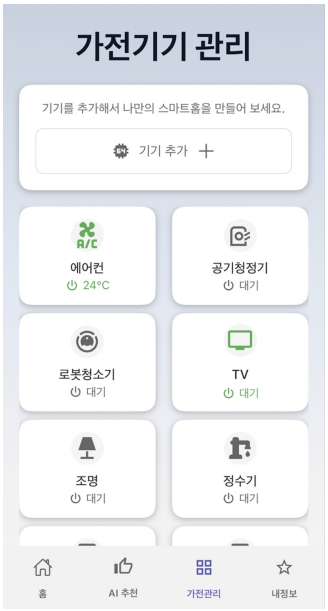

Fig. 54. Appliance management page

When users access the Appliance Management section, they are directed to a control hub where they can add new appliances to their system and manage their existing devices. This interface allows users to both integrate new devices and check their registered appliances.

Fig. 55. Appliance control page

1) Selecting a registered device reveals a status screen that displays its current operational state, complemented by intuitive control buttons that allow users to remotely power the device on or off.

G. Mypage

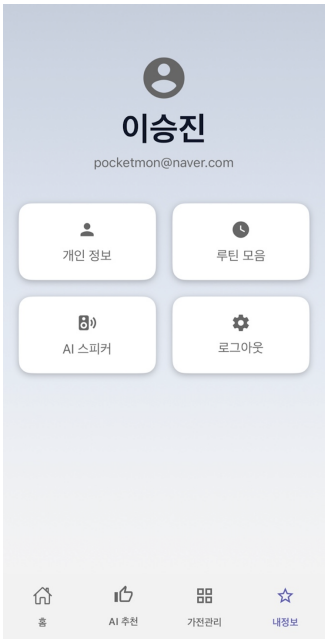

Fig. 56. Mypage

Users accessing the Mypage section are directed to a settings hub where they can view their personal information, browse their saved routines collection, configure AI speaker settings, and access the logout option.

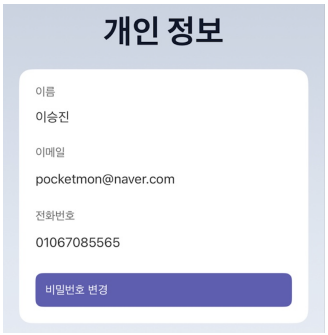

Fig. 57. Personal information page

1) Personal Information: Personal Information displays the user’s name, email address, and phone number, along with a button that enables password modification.

Fig. 58. Routine history page

2) Routine History: Routines history page reveals all created and accepted routines in chronological order, displaying the most recent routine histories first.

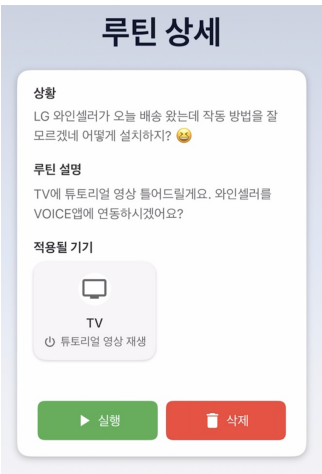

Fig. 59. Routine selection page

3) Routine Selection: Selecting any of the routines in routine history page displays the scenario, routine description, and device adjustments, with buttons available to execute the routine or remove it from the collection.

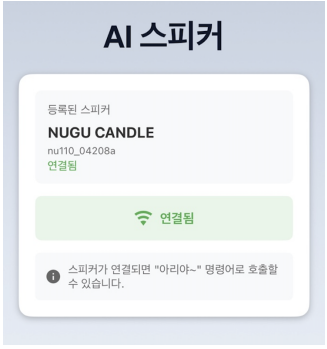

Fig. 60. AI speaker page

4) AI Speaker: Selecting the AI Speaker option shows currently registered speakers and provides a button to connect new speaker devices.

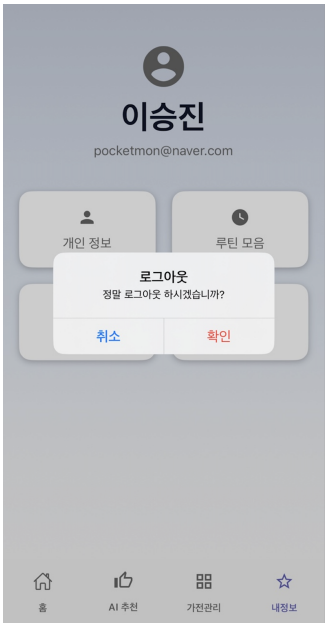

Fig. 61. Logout page

5) Logout: Selecting the logout option enables users to securely sign out of their account.

VII. Discussion

A. Technical Difficulties

The primary challenge in our model training process has been limited computational resources. Although we currently utilize LoRA (Low-Rank Adaptation) to optimize resource usage, constraints still exist regarding model size and complexity. To address this limitation, we are evaluating the implementation of high-performance training environments, specifically considering AWS SageMaker's multi-node training capabilities and enhanced GPU utilization.

To improve training efficiency, we are exploring new computational acceleration techniques while simultaneously examining methods to acquire more diverse training data to enhance model quality. The integration of voice and text data has presented significant challenges. To minimize information loss during speech-to-text conversion, we extract emotional context by analyzing audio signals from user utterances. However, our current capability to capture comprehensive contextual information from utterances remains limited. We are actively investigating methods to more effectively extract situational context from spoken interactions.

B. Conclusion

Our implementation demonstrates the transformative potential of context-based smart home control systems. Moving beyond conventional individual device control, our comprehensive approach to managing multiple devices based on specific situations has demonstrated significant potential for revolutionizing user experience. This approach, combined with increasing demand for personalized recommendations and automated device control, positions it as a potential next-generation core technology in the smart home market.

However, we identified several technical limitations during implementation. Limited computational resources constrained the training of larger, more complex models, and extended epoch duration necessitates further optimization. Key challenges remain in minimizing information loss during voice and text data integration and enhancing the precision of user emotion and context detection. Additionally, continuous improvement through user feedback is essential for enhancing recommendation routine quality.

Addressing these technical challenges will enable the context-based smart home routine recommendation system to evolve into a more sophisticated and personalized solution. Commercialization requires strengthening technical stability, connectivity, and user experience. These improvements will establish our system as a crucial growth driver in the smart home market.

In conclusion, this project has demonstrated both the growth potential and practicality of context-based smart home technology while identifying areas for improvement and future development. Through continued innovation and enhancement, we can create smart home services that deliver greater value to users.

References

[1] "FastAPI," https://fastapi.tiangolo.com/, 2024.
[2] "React Native," https://reactnative.dev/, 2024.
[3] "Node.js," https://nodejs.org/, 2024.
[4] "Sequelize," https://sequelize.org/, 2024.
[5] "Hugging Face Transformers," https://huggingface.co/docs/transformers/, 2024.
[6] "PyTorch," https://pytorch.org/, 2024.
[7] "Parameter-Efficient Fine-Tuning (PEFT)," https://github.com/huggingface/peft, 2024.
[8] "Google Cloud Speech-to-Text," https://cloud.google.com/speech-to-text/, 2024.
[9] "Hume AI," https://hume.ai/, 2024.
[10] "NUGU SDK," https://developers.nugu.co.kr/, 2024.
[11] "Amazon Web Services," https://aws.amazon.com/, 2024.
[12] "Expo," https://expo.dev/, 2024.
[13] "MySQL," https://www.mysql.com/, 2024.
[14] "Flask," https://flask.palletsprojects.com/, 2024.