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 tens of thousands of routines generated by ChatGPT, 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 Chat-GPT

ChatGPT demonstrates remarkable capabilities in generating human-like text across various topics, allowing for the creation of extensive datasets of hypothetical home routines and activities, from major daily tasks to minor habits. This generated data can be categorized by attributes like activity type, frequency, duration, and complexity, then filtered through automated analysis and expert review. This method offers advantages in uncovering patterns and opportunities for technological integration in home environments, particularly benefiting smart home development projects by providing rich datasets that would be challenging to obtain through traditional research. This data-driven approach ensures developments are grounded in realistic domestic scenarios, revealing new opportunities for smart home innovation.

6 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.

7 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.

8 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.

9 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 Tutorial

First-time users are shown introduction screens about key features.

1) Users should be able to select the following options: Skip to sign up or continue tutorial.
2) Tutorial ends with account creation prompt.

2 Sign Up

1) Required information include: Email (ID), password, phone number, name, birth date.
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.

3 Log In

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

4 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 Main Page

The main page should be expressed with maximum simplicity. It must feature the VOICE logo, accompanied by a welcome message stating "Welcome, [User's Name]." 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.

2 AI's Pick

This page utilizes a model trained on tens of thousands of routines generated by an existing generative AI model, ChatGPT, to recommend a single routine tailored to the user's specific situation.

1) Situation Input

a) When a user inputs their situation into a text field, the system preprocesses this input by combining it with contextual information, such as the list of smart devices registered in the VOICE system, into a structured template.

b) This template is then tokenized and formatted as a query suitable for the trained generative AI model. The model processes this input and generates a routine recommendation based on the provided context and user situation.

c) Finally, the system receives the AI-generated response and processes it according to a predefined format, ensuring efficient backend handling and user presentation. This structured approach facilitates consistent input formatting for the AI model and streamlined output processing, ultimately enabling effective information management and response delivery to the user.

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 execute the recommended routine, cancel it, or request a different recommendation.

b) If the user chooses to execute or cancel, they are redirected to the main page.

c) If the user opts for a different recommendation, the process reverts to the recommendation retrieval stage.

d) When a routine is executed, its details are stored in the database.

3 Appliance Management

This page serves as the central hub for managing all appliances within the VOICE system.

1) Users can register and remove 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 related functions.

4 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

User can view or modify their personal data.

a) Personal Data Lookup

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

b) Modify Password

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

2) Routine Collection

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

Upon selecting a routine, a modal window appears with execute and cancel buttons. Pressing execute initiates the routine and returns to the situation search, while cancel simply returns to the situation search.

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 captures nuances such as voice tone and speech rate during this interaction.

1) The user addresses the speaker with a phrase like "Hey [Speaker Name], I'm going to [Action].".
2) The speaker converts speech into text and detects signals like voice tone and speech rate.
3) The model, trained on tens of thousands of routines, recognizes the situation and requests a routine recommendation via script, then receives a response.
4) The AI speaker vocalizes the response.
5) If the user responds positively, the routine is executed; otherwise, it is not.
6) When a routine is executed, its details are stored in the database.

2 Continuous Conversation Tracking by AI Speaker

The speaker continuously monitors user conversations to capture additional information such as context, keywords, urgency, and tone, enabling it to provide appropriate smart home routines for any situation.

1) In daily life, the AI speaker employs 'Speech to intent' technology to continuously listen to ambient sounds.
2) When a particular situation arises, the speaker proactively asks, "I've noticed [Specific Situation]. Would you like me to [Suggested Action]?".
3) If the user responds positively, the routine is executed; otherwise, it is not.
4) Once a routine is executed, its details are stored in the database.

D. AI Model

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

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

This will utilize the ChatGPT API to generate tens of thousands of routine data points. To recommend appropriate routines for users in all situations, a comprehensive dataset will be constructed, including a wide range of scenarios, even those of minimal significance. Subsequently, a transformer-based model such as T5 or Flan-T5 (to be finalized later) will be employed to train the model through transfer learning.

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

Existing libraries will be utilized to implement voice recognition that includes linguistic details such as emotion, tone, and speech rate. 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, using this data to provide the most suitable routines.

III. Development Environment

A. Choice of Software Development Platform

1 Development Platform

1) Windows

Windows is the dominant operating system for software development, providing developers with a comprehensive suite of tools and frameworks through Microsoft's ecosystem. The platform's flagship Integrated Development Environment (IDE), Visual Studio, offers powerful features for coding, debugging, and deploying applications across various programming languages and platforms. Windows' robust NET framework empowers developers to create flexible applications ranging from desktop software to web services using languages like C# and Visual Basic. The platform excels in enterprise development through seamless integration with Microsoft's business services and Active Directory, while tools like PowerShell enable efficient automation of development tasks. Recent modernizations including Windows Terminal, Windows Package Manager (winget), and Windows Subsystem for Linux (WSL) have enhanced the development experience by bridging the gap between Windows and Unix-based workflows. The platform's extensive third-party tool compatibility, combined with DirectX support for game development and strong security features like Windows Defender, makes it a comprehensive choice for developers. Additionally, Windows' integration with cloud services such as Azure and DevOps tools streamlines the development pipeline, while its dominant market share ensures broad application compatibility and testing capabilities across diverse user environments.

2) MacOS

MacOS is a widely used operating system developed by Apple, providing developers with an efficient and streamlined environment for software development. The platform's native development environment,

Xcode, provides a comprehensive set of tools for creating applications across Apple's ecosystem, including iOS, macOS, watchOS, and tvOS. The Unix-based foundation of macOS enables powerful command-line capabilities and seamless integration with common development tools, making it particularly appealing for web and cross-platform development. The platform's built-in terminal application and package managers like Homebrew simplify the installation and management of development tools, while native support for programming languages such as Swift and Objective-C enables fluid development of Apple-centric applications. macOS emphasizes performance optimization and debugging features, offering developers detailed insights into application performance and behavior. The platform's focus on user experience extends to its development tools, providing a clear interface that streamlines the development workflow. Integration with Apple's broader ecosystem, including services like iCloud and TestFlight, facilitates seamless deployment and testing processes. The platform's security features, including Gatekeeper and XProtect, ensure a protected development environment, while its UNIX certification guarantees compatibility with a wide range of development tools and frameworks. Additionally, macOS's support for industry-standard development technologies and its ability to run virtual machines make it an adaptable platform for various development scenarios, from mobile applications to enterprise software solutions.

2 Language and Framework

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

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

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

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) Flask (for Routine Recommendation Model API Server)

Flask is a lightweight and flexible web framework for Python, making it an excellent choice for developing web applications and RESTful APIs. Its minimalistic design allows developers to get started quickly and build applications incrementally. Flask's simplicity and scalability are particularly beneficial when creating an API to serve your AI model, as it allows for easy integration of machine learning models into web services. In your case, Flask will handle incoming requests, invoke the recommendation model, and return results in a user-friendly format. Additionally, Flask's extensive ecosystem of extensions facilitates the implementation

of various features such as authentication, database interactions, and input validation.

3 Libraries

1) Hugging Face Transformer Library (T5/Flan-T5 Model)

Hugging Face Transformer Library is a state-of-the-art library designed for natural language processing tasks. It provides a wide selection of pre-trained models that can be easily integrated into applications. The T5 (Text-to-Text Transfer Transformer) and Flan-T5 models are particularly notable for their ability to handle a variety of tasks through a unified text-to-text framework. This flexibility allows you to fine-tune these models on your specific datasets, enabling them to perform tasks like translation, summarization, and text classification effectively. The library's user-friendly API and comprehensive documentation make it accessible for developers at all levels.

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.

4 Cloud Platform

1) Google Cloud Platform (GCP)

Google Cloud Platform (GCP) provides a robust and scalable cloud environment for developing, training, and deploying machine learning models. GCP offers a variety of services that are particularly beneficial for AI applications, including Google Cloud Storage for data storage, BigQuery for large-scale data analysis, and AI Platform for training and deploying machine learning models. With GCP, you can take advantage of powerful computing resources such as GPUs and TPUs to accelerate the training of your models. The integration of GCP with tools like TensorBoard for visualization and monitoring helps streamline the development process. Additionally, GCP's security and compliance features ensure that your data and applications are protected.

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.

5 Development Environment

1) On Local Machine

Apple M2 16 GB RAM (if this is not enough for fine-tuning T5 model, use second method)

2) Cloud Platform

GCP (Google Cloud Platform) T4 GPU or V100 with 16GB RAM Linux Ubuntu 20.04

B. Software in use

1 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

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

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

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 Android Studio

Android Studio is an essential development environment for Android application development. This IDE provided by Google offers real-time testing capabilities through Android emulator and provides a complete toolkit for building and deploying Android applications developed with React Native. It particularly provides powerful tools for performing Android-specific debugging tasks and supports testing environments for various Android devices. The built-in performance profiling tools enable application performance optimization, and its Gradle-based build system provides an efficient build process.

6 MongoDB

MongoDB serves as our project's primary database management system, utilizing version 7.0. As a NoSQL database, it provides flexible document-based data storage that aligns perfectly with JavaScript's JSON-like data structure, making it particularly suitable for React Native applications. Its schema-less nature allows for agile development by enabling dynamic data modeling without rigid structure constraints. Through MongoDB Atlas, our cloud-hosted solution, we benefit from automated scaling, backup systems, and global data distribution capabilities. The platform's robust aggregation framework and indexing capabilities ensure efficient query performance, while its native drivers for Node.js provide seamless integration with our backend services. Additionally, the built-in security features, including field-level encryption and role-based access control, help maintain our application's data integrity and confidentiality.

7 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.

8 Github

GitHub is a web-based hosting service for version control using Git. It provides a centralized system where developers can store, share, and collaborate on code repositories. The service offers various features including pull requests, issue tracking, and code review tools which help teams coordinate their work effectively. GitHub's actions provide automated workflow capabilities for continuous integration and deployment processes. The platform includes documentation tools, wiki pages, and project management boards to help teams organize their development process. Through branches and forks, developers can work on code simultaneously without affecting the main project, then merge their changes through pull requests. GitHub supports integration with numerous development tools and services, extending its functionality beyond basic version control. The service also includes security features such as dependency alerts and secret scanning to help maintain code security. Its enterprise version provides additional tools for larger organizations, including advanced authentication and access controls. The platform's widespread adoption has made it a standard tool in modern software development, used by individual developers, open-source projects, and large corporations alike.

9 Overleaf

Overleaf is an online LaTeX editor that enables users to create and edit documents in real-time. The service combines a LaTeX editor with collaboration features, allowing multiple users to work on documents simultaneously. Overleaf includes a comprehensive collection of LaTeX templates for various document types including academic papers, presentations, and theses. The editor

provides instant PDF preview of the document as users write, making it easier to visualize changes. Overleaf's interface includes syntax highlighting, auto-completion, and error checking to assist in writing LaTeX code. The platform maintains document version history, enabling users to track changes and revert to previous versions when needed. It integrates with reference management tools and other academic services, streamlining the document creation process. The editor includes a complete LaTeX environment with built-in compilation, eliminating the need for local LaTeX installation. Overleaf also provides extensive documentation and tutorials to help users learn and effectively use LaTeX.

10 Notion

Notion is a collaborative workspace and document management system that combines note-taking, project management, and organization tools. The platform uses a block-based structure where users can create and arrange different types of content including text, tables, lists, and embedded files. Notion documents can be organized hierarchically with nested pages, making it efficient for managing complex projects and information. The service includes templates for various purposes such as project tracking, meeting notes, and documentation. Notion's collaboration features allow team members to work on documents simultaneously, with changes syncing in real-time across devices. The platform supports integration with other tools and services, enabling users to embed content from external sources. Notion includes version history for tracking changes and database features for organizing structured information. The platform's flexible interface allows users to customize their workspace layout and organize content according to their specific needs.

11 ChatGPT

ChatGPT is a large language model developed by OpenAI that serves as both a conversational AI system and a foundation for creating custom AI models through fine-tuning. The model uses transformer architecture trained on extensive text data to engage in conversational interactions, and developers can adapt it for specific applications or domains. ChatGPT can handle various tasks including answering questions, writing content, explaining concepts, and assisting with code-related queries. The system processes input in context, maintaining relevance throughout conversations while generating responses based on its training data. ChatGPT supports multiple languages and can adapt its communication style based on the user's needs. The model can help with tasks such as text summarization, translation, code explanation, and creative writing. It includes built-in safeguards and limitations to promote responsible use, and its responses are based on its training data cutoff date.

12 Zoom

Zoom is a cloud-based video communications platform that provides video conferencing, online meetings, and chat services. The platform offers high-quality video and audio streaming capabilities with features like screen sharing, recording, and virtual backgrounds. Zoom supports various meeting formats from one-on-one calls to large webinars, with features for participant management and meeting security. The service includes collaboration tools such as breakout rooms, polls, and whiteboarding capabilities to enhance virtual interactions. Zoom integrates with calendar systems and other productivity tools, streamlining meeting scheduling and organization. The platform works across different devices and operating systems, making it accessible for users on desktop and mobile devices. Zoom's architecture enables stable connections across different network conditions and includes features for bandwidth management and audio optimization.

C. Task Distribution

TBD

IV. Requirement Specification

A. Common Features

1 Entry

1) VOICE-Tutorial-Page

The tutorial should be the first screens where VOICE is downloaded and shown. This guide provides new users with an overview of the app's essential features.

2) VOICE-Tutorial-Skip

Skip the tutorial and proceed directly to the signup page.

3) VOICE-Tutorial-Navigate

View next page. If a user has reached the final tutorial page, the next step should be to create their account and begin using VOICE.

2 Signup

1) VOICE-Signup-Page

VOICE requires five use information to sign up for membership: E-mail, password, phone number, name, and birth date.

2) 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.

3) 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 ‘****’.

4) 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.

5) 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.

6) VOICE-Signup-Birth Date

The user will be presented with a calendar in a modal window upon clicking the birth data field, allowing the user to select the date of birth from this interface.

3 Login

1) 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, Social Media Login options, and Language Selection.

2) VOICE-Login-Types

VOICE offers two login options: (1) Local login through VOICE membership, (2) SNS (Social Networking Service) login.

3) 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.

4) 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.

5) 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.

6) VOICE-Login-SNS login

This option uses the authentication APIs provided by Google, Kakao, and Naver.

7) VOICE-Login-SNS Success

Upon successful authentication through social media, the system must receive the user's name, date of birth, and phone number. Following this data retrieval, the user is directed to the main page.

4 Account Recovery

1) 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.

2) 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.

3) 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”.

4) 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.

5 Reset Password

1) 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.

2) 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”.

3) 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.

4) VOICE-Reset Password-New Password

Upon successful verification, the user enters a new 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 is displayed on the screen in the format of asterisks '****' for security purposes.

B. In Application

1 Mainpage

The main page should be expressed with maximum simplicity. It must feature the VOICE logo, accompanied by a welcome message stating "Welcome, [User's Name]." and include a menu bar. The menu bar consists of 'Home, AI's Pick, Application Management, Mypage', and clicking on each menu item navigates to that corresponding page.

2 AI's Pick

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.

1) VOICE-AI's Pick-Format Conversion

The system processes this input by integrating it with relevant contextual data, such as the list of smart devices registered in the VOICE system, into a structured format.

2) VOICE-AI's Pick-Generative AI Model

This converted template is tokenized and formatted into a query suitable for the trained generative AI model. The model processes this input and generates routine recommendations based on the provided context and user situation.

3) VOICE-AI's Pick-Backend Processing

The system receives the AI-generated response and processes it according to a predefined format, ensuring efficient backend handling and user presentation. This approach facilitates consistent input formatting for the trained AI model and streamlined output processing, ultimately enabling effective information management and response delivery to users.

4) VOICE-AI's Pick-Loading

While the system processes the response data for display, a loading screen is shown to the user.

5) VOICE-AI's Pick-Display

A routine detailing how each appliance should operate is displayed on the screen. The screen includes buttons for accepting, canceling, or requesting different recommendations.

6) VOICE-AI's Pick-Accept

Upon accepting the AI's recommendations, the system executes the routines, stores their details in the database, and redirects to the main screen.

7) VOICE-AI's Pick-Reject

If the user rejects the recommendations, the system redirects to the main screen.

8) VOICE-AI's Pick-New Request

When a user requests a different recommendation, the 'AI's Pick Loading' screen appears and the above process is executed again.

3 Appliance Management

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.

1) VOICE-Appliance Management-Registration

Users can register appliances here. This must be reflected in the database in real-time. (Registration method to be added later)

2) VOICE-Appliance-Management-Control

Users can manually turn appliances on and off. Additionally, appliances can be removed from VOICE. (Additional features to be added later)

4 Mypage

This page displays user's personal information and previously executed recommendation-based routines. It also enables AI speaker integration. The mypage shows personal information and includes buttons for password modification, viewing routine collection, and AI speaker integration.

1) VOICE-Mypage-Profile-Page

This page allows users to view their personal data. It displays name, email, and date of birth.

2) VOICE-Mypage-Profile-Modification

After identity verification through mobile phone authentication, users can modify their personal information.

3) VOICE-Mypage-Routine Collection-Page

On the routine collection page, when a specific situation is entered in the search bar, relevant routines are

displayed in a viewable format. The routines shown here are those previously executed by the user.

4) VOICE-Mypage-Routine Collection-Selection

When a routine is selected, a modal window appears with execute and cancel buttons.

5) VOICE-Mypage-Routine Collection-Selection Execute

Pressing the execute button will initiate the routine and return to the routine collection page.

6) VOICE-Mypage-Routine Collection-Selection Cancel

Pressing the cancel button will return to the routine collection page without any changes.

7) VOICE-Mypage-AI Speaker-Page

When the 'Find AI Speaker' button is pressed in Mypage, the VOICE system detects AI speakers. Select the detected device and press the 'Connect' button.

8) VOICE-Mypage-AI Speaker-Complete

When integration is successfully completed, a "Connection Complete" message is displayed.

C. With AI Speaker

1 User-Initiated Interaction

1) The user addresses the speaker with a phrase like "Hey [Speaker Name], I'm going to [Action]". The system supports natural language processing, allowing users to speak naturally rather than using fixed commands. The speaker maintains a conversational flow, allowing users to continue the dialogue naturally after the initial interaction.

2) Upon receiving the user's voice input, the speaker employs advanced speech-to-text conversion while simultaneously analyzing various voice parameters for deeper context understanding. The system processes emotional states through tone analysis, monitors speaking rates to detect urgency or relaxation, analyzes volume patterns to understand emphasis points, and evaluates speech clarity to gauge the user's current state. This multi-layered voice analysis enables the system to understand not just what the user is saying, but how they are saying it.

3) The AI model, trained through tens of thousands of routines, processes the analyzed information to determine the most appropriate routine for the situation. The system considers multiple contextual factors simultaneously, evaluating the time of day, referencing the user's previous preferences and routine history, and checking current house status. This comprehensive contextual analysis ensures the selected routine precisely matches the user's current needs and circumstances.

4) Following the analysis and routine selection, the AI speaker generates a natural language response that clearly communicates the suggested routine while maintaining conversational flow. The system articulates its understanding of the situation and presents relevant routine options about proposed actions.

5) The system processes the user's response, recognizing various forms of responses and understanding modifications to the suggested routines. When users provide positive responses, the system proceeds with execution, while negative responses result in routine cancellation. The system's flexibility allows for partial acceptance and modifications, enabling users to customize suggested routines according to their preferences.

6) Following routine execution, the system comprehensively documents the interaction in its database, recording detailed information about the user's initial request, the specific actions executed, any modifications made to the suggested routine, and the success or failure status of each individual action. This systematic data collection and storage process continuously enhances the system's machine learning capabilities, enabling increasingly accurate and personalized routine recommendations for future interactions.

2 Continuous Conversation Tracking by AI Speaker

1) The AI speaker continuously monitors ambient sounds using 'Speech to intent' technology. The system processes audio locally until wake word detection, identifies relevant keywords and phrases, and detects urgent situations through sound pattern recognition.

2) The AI speaker initiates conversation when it recognizes specific situations. Based on the detected context, keywords, and user's tone, the system formulates a natural query like "I've noticed [Specific Situation]. Would you like me to [Suggested Action]?". The suggestion focuses on situational relevance and user convenience, presenting appropriate routines that match the current context.

3) The system evaluates the user's response through voice tone analysis and keyword recognition to determine approval or rejection. For positive responses, the speaker proceeds with the suggested routine, while negative responses result in immediate cancellation without further prompts.

4) After each routine execution, its details are stored in the database. This includes the contextual situation that triggered the suggestion, the specific routine executed, the user's response pattern, and the execution timestamp. This stored information helps the system refine future suggestions and improve the accuracy of its situational awareness.

D. AI Model

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

1) Dataset Acquisition Using ChatGPT API

The first step involves leveraging the ChatGPT 4o API to gather a comprehensive dataset of 100,000 entries that encapsulate a wide range of situations and their corresponding smart home routine recommendations. The dataset will be structured in pairs, where each entry consists of situation information and the related smart home routine recommendation. This format ensures that the model can learn to associate specific situations with appropriate responses, enabling it to generate context-aware recommendations. The goal is to ensure that the dataset covers diverse scenarios, providing a robust foundation for training the generative AI.

2) Model Fine-Tuning with the Transformer Library

Once the dataset is established, the next step is to utilize the Transformer library to implement the T5 (Text-to-Text Transfer Transformer) model. This model will be fine-tuned through transfer learning based on the previously collected dataset. The fine-tuning process will enhance the model's ability to understand the nuances of the input situations and generate appropriate smart home routine recommendations. By training the model on this specific dataset, it will develop a more refined understanding of context and improve its accuracy in predicting optimal routines.

3) API Development with Flask

After fine-tuning the model, the final step is to integrate it into a Flask backend server. This server will be responsible for handling incoming text inputs that describe the user's situation. Upon receiving an input, the server will invoke the trained model to generate and return the most suitable smart home routine recommendations. By encapsulating the model within a Flask API server, the application will facilitate seamless interactions, allowing 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

Collect human voice data using Speech-To-Text service of SKT NUGU SDK to access the microphone and convert spoken input into text.

2) Extract Emotion and Context Information

To add more contextual information such as user's emotional state and urgency, analyze the voice data for emotional tone, pitch, and speed. To accomplish this task, use a Sentiment and Emotion Analysis Libraries such as nltk (for sentiment analysis), or more specialized emotion-detection APIs (like IBM Watson Tone Analyzer). After extracting verbal features, use the ChatGPT API to enrich the text with additional context derived from voice tone and urgency, providing more comprehensive understanding of the user's current state by adding context to the statement.

3) Model Processing and Recommendation

Send the modified text, including emotional and urgency cues, as input to the model for generating personalized routine recommendations. Based on the processed text, the model outputs routines. Ensure the output format aligns with how the NUGU system can execute or present these routines.

4) Response Delivery

The model's output then be converted back to speech using NUGU's TTS (Text-to-Speech) and provide the user with auditory feedback on the recommended routines.

V. Architecture Design

A. Overall Architecture

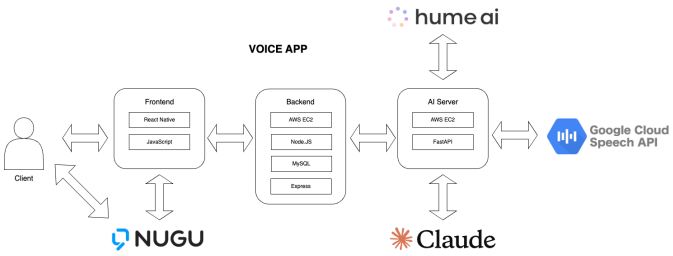

Fig. 1. Overall Architecture

VOICE users interact with our system through the Frontend application or NUGU interface. To achieve this, the system consists of 3 main modules: Frontend, Backend, and AI server.

The first module is "Frontend," developed with React Native and JavaScript. It provides the main user interface and integrates NUGU for voice command capabilities, allowing users to interact through both text and voice inputs.

The second module is "Backend," built with Node.js and Express on AWS EC2, using MySQL for data storage. It manages user authentication, device configurations, and routine data while coordinating component communications. The database stores essential data including user information, real-time device status, and history of accepted/executed routines for pattern analysis and personalization.

The third module is ”AI Server,” using FastAPI on AWS EC2. This server processes voice and emotion analysis by integrating multiple AI services: Hume AI analyzes emotional patterns, Claude structures data into standardized formats for system processing, and Google Cloud Speech API handles voice-to-text conversion. The AI Server manages these services to provide comprehensive voice and emotion analysis capabilities.

To ensure system reliability and performance, we implemented standardized API endpoints with robust error handling, monitoring, and logging capabilities across all components.

B. Directory Organization

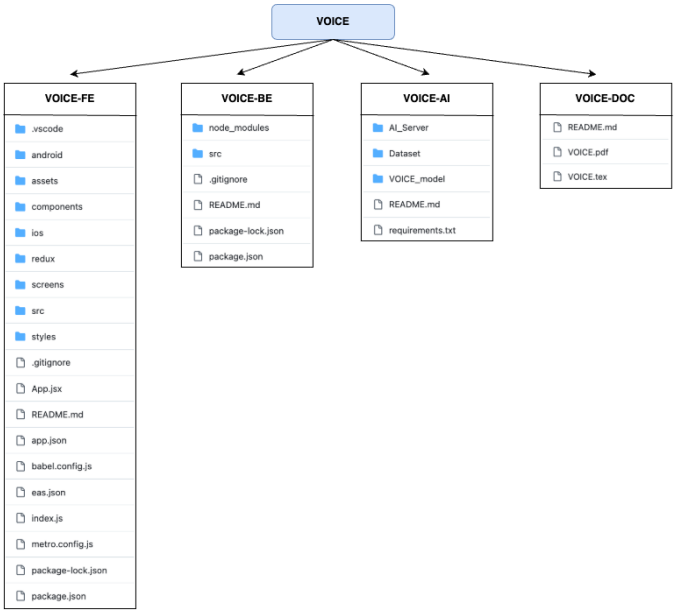

Fig. 2. Directory Organization

TABLE II
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
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

TABLE III
DIRECTORY ORGANIZATION-FRONTEND 2

Directory	*File / Folder Name*	*Modules used*
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 /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 IV
DIRECTORY ORGANIZATION-BACKEND

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
VOICE-BE /src/models	ai-speaker.model.js appliance.model.js index.js routine-history.model.js user.model.js	sequelize dotenv
VOICE-BE /src/routes	ai-pick.routes.js appliance.routes.js auth.routes.js index.js mypage.routes.js voice.routes.js	express multer path fs
VOICE-BE /src	app.js	express sequelize express-session cors morgan dotenv

TABLE V
DIRECTORY ORGANIZATION-AI

Directory	*File / Folder Name*	*Modules used*
VOICE-AI	AI_Server Dataset VOICE_model README.md requirements.txt	
VOICE-AI /AI_Server	__pycache__ main.py parsing_routine.py	fastapi pydantic transformers torch langchain_core.prompts langchain_core.output_ parsers langchain_anthropic dotenv json
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_model	LoRa_train_model.py data_preprocessing.py predict_model.py train_model.py	pandas torch tqdm transformers sklearn os peft

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 and manages hierarchical file storage. Implements error handling for file processing failures.

- 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. Configures multer for file uploads and audio conversion routes.

- 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.

- 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.

- 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.

VI. Use Cases

A. Loading

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.

B. Login

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. Sign Up

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.

D. Main Page

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 new appliances to their system.

1) Users should be able to select the following options: Skip to sign up or continue tutorial.

E. AI's Pick

When users tap the "AI's Pick" section from the Home screen, 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.

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 aren't satisfied with the suggested routine, they can request an alternative recommendation by using the "Recommend Again" button to receive different suggested routines.

F. Device Management

When users access the Device 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 control their registered appliances through basic operations such as turning them on or off.

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

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.

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

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

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

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

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