VISQUIT - Voice IS a Quite Universal Interactive Tool (Kiosk with Vioce Recognition)

Kim Jung Mo
2014005414
Dept. of Information System
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email: cadenzah93@gmail.com

Lee Hyo Sik
2014026208
Dept. of Business Management
School of Business,
Hanyang University
Seoul, Rep. of Korea
Email: lhslhs9394@gmail.com

Lee Ha Min
2017029134
Dept. of Information System
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email: ggamini7@gmail.com

Hwang Sung Woo
2016026599
Dept. of Information System
College of Engineering,
Hanyang University
Seoul, Rep. of Korea
Email: hsw0194@gmail.com

***Abstract*—VISQUIT is a Kiosk service powered by voice recognition technology based on artificial intelligence. Nowadays, many stores adopt using Kiosk system, by which the cost of hiring clerks and the time spent to make an individual order reduce. People who are accustomed to digital-based system has no much difficulty using this type of new method in store, but for those people who are digitally illiterate, even ordering their meal in a restaurant is a hard task. To solve the problem with current Kiosk system's poor user experience, we suggest using voice recognition technology to compose the ordering system's UI. This can effectively replace the previous "select-based" ordering system with "speech-based". Based on this approach, we will develop and propose new paradigm of designing UI of Kiosk system.**

1. Introduction

1.1. Motivation

Kiosks are unmanned machine that can take orders on behalf of a clerk. Recently, the installation and operation of kiosks is increasing, especially in franchise restaurants. This saves labor costs for the personnel required to place an order and allows more orders to be processed in unit time.

However, some people suffer from the appearance of kiosks. That people are who are not familiar with these Digital technology. In particular, for older people, they are very new to digital technology. According to the National Information Society Agency (NIA) '2018 Digital Information Gap Survey,' the overall 'digital information level' for the elderly aged 55 or older is only 63.1% of the general public. The contrary situation arises where the marginalized class exists due to the emergence of technology to make people convenient.

1.2. Problem Analysis

Among the problems of kiosks, we will focus on improving the ordering / payment software installed inside the kiosks. In the conventional method, as all the menus are displayed on the screen, the amount of information increases, which causes inconvenience to those who are not familiar with the UI. In addition, in order to maximize the amount of information displayed, the size of text and images is small, and the screen resolution is limited, so the display is not clear. In addition, the sensitivity of the touch screen is not as accurate as that of the physical buttons, providing a confusing UX to the user.

1.3. Solution

Using the artificial intelligence speech recognition technology of SKTelecom's NUGU Platform, the user's speech is recognized and ordered based on it. This provides a familiar interface to users who are not familiar with digital technology, as if they are talking to a real person. It can also handle multiple orders without the need for a complex UI.

In addition, the UI is configured to maximize the clarity of the content by using a large image and text, and to better understand the content by users with low vision. As the size

of the content increases, the amount of information that can be contained on one screen decreases, so the UI is designed to implement the same business logic as the existing despite the reduced amount of information.

2. Requirements

2.1. User side

2.1.1. View Menu List (ID 101).

1) Provide menu list with paper or tablet

2.1.2. Order by Voice (ID 102).

1) SKT NUGU Speaker handles this
2) Accept orders according to Intent and Entity

2.1.3. Check Menu (ID 103).

1) Send an order from the Nugu speaker to the back-end proxy
2) Total orders in Backend proxy based on Entity
3) Transfer the aggregated order back to the Nugu speaker to confirm the order using the TTS function
4) Send user confirmation to backend proxy

2.1.4. Payment (ID 104).

1) Customer could pay with card

2.2. Client Side

2.2.1. Log-in (ID 201).

1) To identify clients, require client to log in

2.2.2. Menu update (ID 202).

1) Client could add new menu
2) Client could delete menu
3) Client could modify menu

2.2.3. View Order History (ID 203).

1) Client could see order lists customers ordered
2) Client could see order history already served

2.3. Server Side

2.3.1. Check order (ID 301).

1) Sum menu input from NUGU
2) Response to NUGU
    a) Receive orders from the Nugu speaker with Intent and Entity and send them to server
    b) aggregate orders based on Entity in the Backend proxy
    c) Send aggregated orders on the Nugu speakers
    d) Order confirmed by user with TTS function in NUGU speaker
    e) If the order is correct, backend proxy stores the order in DB

2.3.2. Send order to database (ID 302).

1) Server get orders from NUGU speaker
2) Server keep these orders in database

2.3.3. Make json for NUGU (ID 303).

1) Server should make json file for learning of NUGU speaker

3. Development Environment

3.1. Software Development Platforms

We chose Web environment to develop our project. Many applications can be run on the Web. Web technology(Node.js) will be used to show that voice kiosks work properly. In addition, we will use RDS to apply what we have learned in class, and additionally use AWS commercial cloud platforms such as EC2, S3, etc. SKT's NUGU API will be used to analyze users' voice commands and translate them into text.

3.1.1. Node.js. Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.

3.1.2. React.js. React (also known as React.js or ReactJS) is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded.

3.1.3. Amazon Web Service EC2. Configure an environment where applications can be developed and distributed quickly without the need for physical hardware equipment. Deploy as many virtual servers as you want, configure your security, network, and manage your storage. A sudden change in server spacs can quickly scale up or down depending on the changes, reducing the need for forecasting server traffic.

3.1.4. Amazon Web Service S3. Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network. Amazon S3 can be employed to store any type of object which allows for uses like storage for Internet applications, backup and recovery, disaster recovery,

data archives, data lakes for analytics, and hybrid cloud storage. In its service-level agreement, Amazon S3 guarantees 99.9% uptime, which works out to less than 43 minutes of downtime per month.

3.1.5. Amazon Web Service RDS. Amazon Relational Database Service (or Amazon RDS) is a distributed relational database service by Amazon Web Services (AWS). It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications. Administration processes like patching the database software, backing up databases and enabling point-in-time recovery are managed automatically. Scaling storage and compute resources can be performed by a single API call as AWS does not offer an ssh connection to RDS instances.

3.1.6. SKTelecom NUGU API. Based on SK Telecom's technical skills such as voice recognition, voice synthesis, and understanding of natural language through NUGU developers, the company can develop new functions and provide NUGU's various functions through voice command in devices or applications owned by its affiliates. We will recognize and categorize the user's voice commands through the NUGU API and send output results to the user via voice.

3.1.7. Docker. Docker is a set of platform-as-a-service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.

3.2. Programming Languages

3.2.1. Javascript. Javascript is a high-level, interpreted scripting language that conforms to the ECMAScript specification. Javascript has flexible grammars: freedom from indentation, loose type checks. Also, it adopts modern progamming padigms and has convenient and great features: function programming, reactive programming. By using this language we can learn various modern progamming paradigms. Javascript is used in web browsers, which means it does not require any special working environment to run program written by Javascript.

3.3. Cost Estimation

This project heavily rely on Amazon Web Service. The cost estimation is in Table 1. This is calculated by Amazon Web Service Cost Calulator.

TABLE 1. ROLE ASSIGNMENT FOR EACH PARTICIPANTS OF THIS PROJECT

Service	Region	Cost(Monthly)
Amazon EC2	Asia Pacific (Seoul)	USD($) 11.65
Amazon RDS	Asia Pacific (Seoul)	USD($) 19.64

3.4. Development Environment Description

Used development environment tools information is described in Table 2.

TABLE 2. ROLE ASSIGNMENT FOR EACH PARTICIPANTS OF THIS PROJECT

Name	Version	Description
Windows	10 Pro	Operating System made by Microsoft
macOS	Catalina(10.15)	Operating System made by Apple, used in Macbook
Ubuntu	16.04.6	Operating System developed by Linux Foundation
Visual Studio Code	1.39.1	Text editor and Integrated Development Editor made by Microsoft
Atom.io	1.40.1	Text editor and Integrated Development Editor made by GitHub
iTerm2	3.3.6	Terminal Emulator used in macOS
Z shell(zsh)	5.7.1	Unix shell for CLI

3.5. Market Research & Software in Use

3.5.1. Kiosk Market in Korea. Unmanned stores are spreading all over the world. The unmanned store are actively being made due to artificial intelligence, the Internet of Things, the development of sensor technology, the lack of labor force and the minimum wage hike. South Korea has been burdened with labor costs following a 10-percent minimum wage hike for two consecutive years, and the unmanned era is in full swing due to the spread of an "untact" culture that is reluctant to contact. In restaurants, customers order on their own through KIOSK. And unmanned convenience stores have begun to sprout up everywhere. Unmanned store is spreading not only in the retail industry but in almost all areas, regardless of industry. Demand for kiosks is also on the rise as unmanned stores have emerged. A market for unmanned payments using kiosks is emerging. As of 2017, the size of South Korea's kiosk market is estimated to be around 250 billion won, with a high annual growth rate of 14 percent. The market, which stood at only 60 billion won in 2006, rose to 180 billion won in 2013 and rose to 250 billion won in 2017.

3.5.2. Voice Recognition ARS. The ARS allowed us to perform the tasks we needed without waiting time. However, the initial ARS was a one-sided way of listening and

pressing buttons. As people become accustomed to ARS, they can't wait to get the number of services they need. In addition, the service used was mostly concentrated in some services. Based on this, ARS through voice recognition was able to connect the necessary services quickly by listening to the voice or words. Considering these voice recognition ARS, it is thought that it will help us solve the problem.

3.5.3. Voice Recognition AI. There are services in the market called artificial intelligence voice assistant such as Bixby, Siri, Nugu and Gigagenie. Despite providing convenient services, many people are reluctant to use these services. Understanding these reasons is thought to help us envision the services we will provide in the future.

3.5.4. Delivery Application. We think the delivery app is the same situation as the kiosk we are currently thinking about. Orders, which were previously made through voice-to-speech conversations over the phone, have been made possible through the application with several touch points. There are many advantages, such as the availability of personalized orders through delivery apps, but it is still a far from those who are not familiar with digital culture. If Kiosk's problem is solved, it is believed that the delivery app will be able to solve the problem in the same way later.

3.5.5. Starbucks Siren Order with Voice. T-Map x NUGU introduced a voice service for Starbucks' "Siren Order". The combination of "T-Map x Nugu" and "Siren Order" makes it possible for users to order for various Starbucks products with voice, even while driving. Orders are based on voice while driving, followed by beverage selection, neighborhood store selection, order product confirmation, order receipt and payment. In addition, linking T-map, by forwarding the order to the store within five minutes of arrival at the designated store, users were able to improve the inconvenience of waiting for the ordered product at the store for a long period of time, while considering the freshness of the product.

3.6. Task Distribution

Task Distribution is shown in Table 3. Note that each project participants periodically switched their roles with the others to apply different perspectives and improve their expriences.

TABLE 3. TASK DISTRIBUTION FOR EACH PARTICIPANTS OF THIS PROJECT

Task	Assignee	Description
Project Manager	Alice	Schedule overall development plan and assign proper jobs to team members
Consumer	Bob	Test and try out a prototype application, gather the potential improvement
User	Chris	Gather basic feature requirement for this project by asking questionnaires the elderly near the university.
Developer	Daniel	Prepare a development environment for this project and build up the application

4. Specification

4.1. User side

4.1.1. View Menu List (ID 101).

1) Client should provide menu

4.1.2. Order by Voice (ID 102).

1) Quantity of a menu in a order cannot pass 10
2) End ordering when customers say '8]'

4.1.3. Check Menu (ID 103).

1) NUGU speaker asks customers whether order is right

4.2. Client Side

4.2.1. Log-in (ID 201).

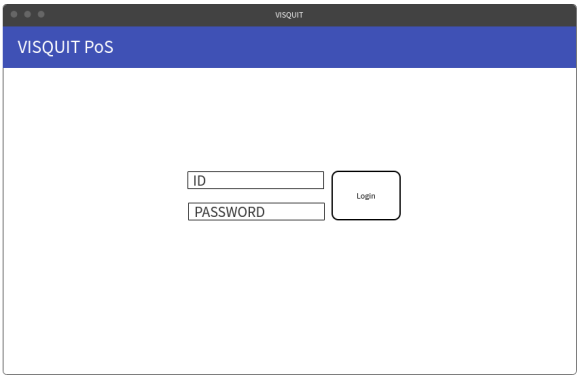

FIGURE 1. LOGIN

1) Client could login with SKT ID
2) After log-in clients could choose menu management or check order list

4.2.2. Menu update (ID 202).

1) Client could see menu list of the store
2) Clients should enter menu name and price to add menu
3) Clients should check all the synonyms for learning of NUGU speaker

4.2.3. View Order History (ID 203).

1) Clients could see order lists in a page
2) When orders are served client could see orders in order history

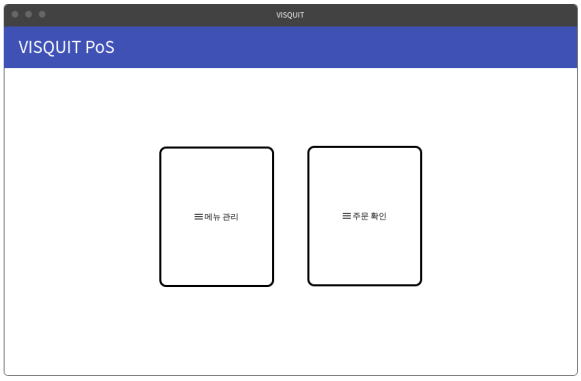

FIGURE 2. MAIN

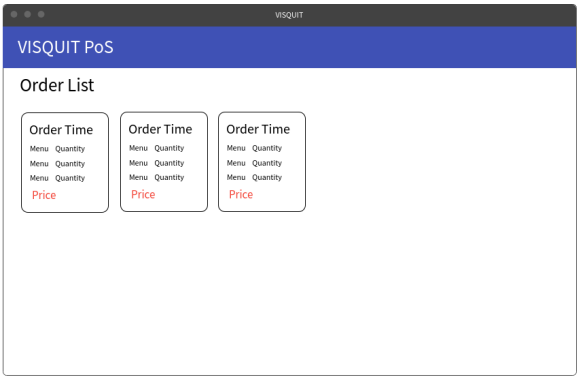

FIGURE 5. ORDER LIST

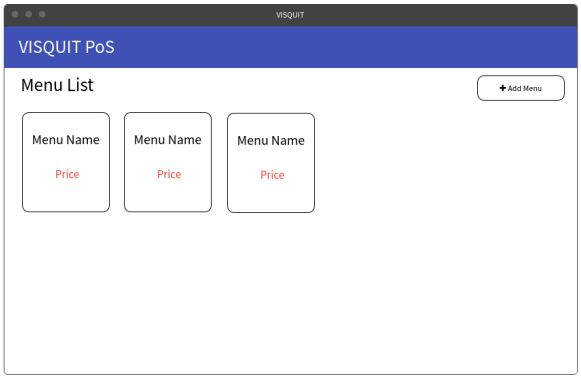

FIGURE 3. MENU MANAGEMENT

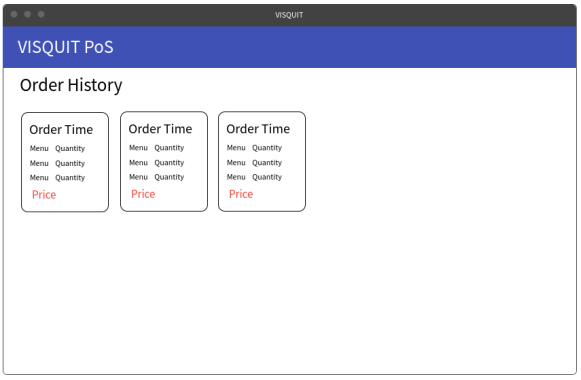

FIGURE 6. ORDER HISTORY

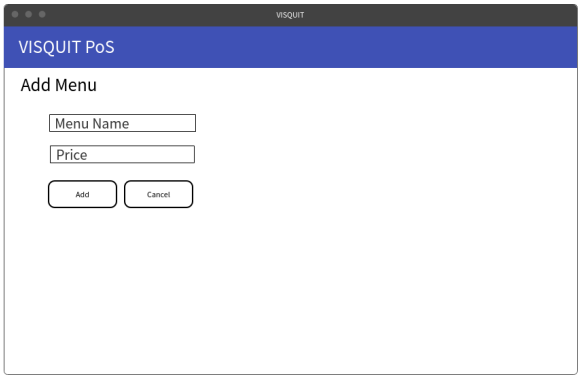

FIGURE 4. ADD MENU

4.3. Server Side

4.3.1. Check order (ID 301).

1) When customer order, NUGU speaker check each menus
2) Customer should confirm what they ordered
3) Server give customers order number
4) Customer can check what they ordered with order number

4.3.2. Send order to database (ID 302).

1) When customer confirm all the orders, server confirm the orders and keep in database

4.3.3. Make json for NUGU (ID 303).

1) When clients update menu, server should make json file

5. Architecture Design

5.1. Overall Architecture

Our service is web-based application and heavily depends on Amazon Web Service for deployment. Our application's UI is implemented with React.js. Our server is implemented by Node.js with Express.js web server on it to serve

service data to frontend UI as REST API, and interact with SKTelecom NUGU's API and speaker. (DESCRIPTION FOR INFRA AND CLOUD ARCHITECTURE SHOULD BE ADDED). Figure 1 shows overall architecture of the service.

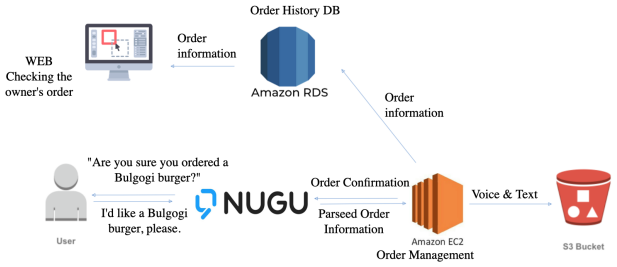

FIGURE 7. OVERALL SYSTEM ARCHITECTURE DIAGRAM

5.2. Directory Organization - Frontend

Table 4 shows the directory organization of React.js frontend application's project.

TABLE 4. DIRECTORY ORGANIZATION FOR FRONTEND APPLICATION PROJECT

Directory	File name(s)	Modules used
/src/components	Component.js	Storybook
/src/containers	ComponentContainer.js	Redux.js, axios
/src/pages	App.js, Page.js	React Router
/redux	module.js	Redux.js
.storybook	config.js, addon.js	Storybook

5.2.1. /components. React codes implementing each UI assets used to compose the Frontend UI. Basic design concept is based on Material-UI and we modified it according to our purpose.

5.2.2. /containers. React codes including states, event listeners, and functions used in each pages. The structure and workflow are based on Redux. States and functions are initialized in this code and passed into each UI components.

5.2.3. /pages. Index codes where each routings start. Each routings are provided via React Router. Our service's frontend UI is rendered by React-based client-side rendering, where page routing executes in frontend, not backend.

5.2.4. .storybook. Storybook is a test framework that eases developers by allowing to check each design assets independently on the screen rapidly. With storybook, we can check if a design asset is correctly implemented without running entire service, improving efficiency of development process and easing the communication with designers and planners.

5.2.5. Axios. Axios is Promise based HTTP client for the browser and Node.js. It is used to request data from API server.

5.2.6. Webpack. Webpack is a module bundler used to combine all codes and resources in a project into a single file, also providing size minifying. Our service is optimized with Webpack to rapidly serve our web page to users.

5.2.7. Babel. Babel is a JavaScript transpiler that converts React syntax into plain JavaScript codes, downgrades ES6+ JavaScript syntax into ES5 syntax. This is a required module to execute React-based project.

5.3. Directory Organization - Backend

TABLE 5. DIRECTORY ORGANIZATION FOR BACKEND APPLICATION PROJECT

Directory	File name(s)	Modules used
/api/store	index.js, store.controller.js	Express, Sequelize
/api/menu	index.js, menu.controller.js	Express, Sequelize
/api/common	baseResult.js,	-
/config	sequelize.json	-
/models	MENU-TB.js	Sequelize
/bin	www.js	Express
.	app.js, Dockerfile	Express

5.3.1. /api/store. Node.js codes implementing API that related to store information and 'make order' business logic. In API, Express and Sequelize modules are used.

5.3.2. /api/menu. Node.js codes implementing API that related to menu-information business logic. In API, Express and Sequelize modules are used.

5.3.3. /api/common. Node.js codes including basic API response format, function that checks whether HTTP Request sender is legit or not and event-handler on some specific HTTP connection event.

5.3.4. /config. In order to use sequelize module in Node.js file, configuration about DB connection needed. sequelize.json files contain database name,address of host ,password and type of database.

5.3.5. /models. Querying database using sequelize needs certain model definition or configuration about table in that database. In models folders, all table in database are defined in JavaScript syntax. Then Index.js file imports that *table-definition JavaScript file* and sets connection between Node.js and Sequelize module.

5.3.6. /bin. Entry point of visquit-backend service. In www.js file, using express module,listen specific port.

5.3.7. app.js,Dockerfile. app.js routes incoming HTTP request into proper API handler. We use Docker , so docker configuration file,Dockerfile,needed.

5.3.8. Express. Express is Web framework used in our project. Using this module, we don't *reinvent the wheel* about HTTP service. Express provide simple and convenient middle-ware like routing,error-handler,cookie and parser. Many of above middle-ware can be implemented custom way that we want to make.

5.3.9. Sequelize. We use ORM(Object-relational mapping) module named Sequelize for database querying. Using this module, we don't have to write *long and complex* SQL Query statement, just simply use function like findOne,findAll,destroy,create to get data from database.

5.3.10. express-list-endpoints. Simple module that list all API entry point URL and type(GET,POST,PUT,PATCH,DELETE) . We use this module to print out currently implemented API list.

5.3.11. bodyParser. In order to parsing HTTP request body, we use bodyParser module.

6. Use Cases

6.1. Frontend UI

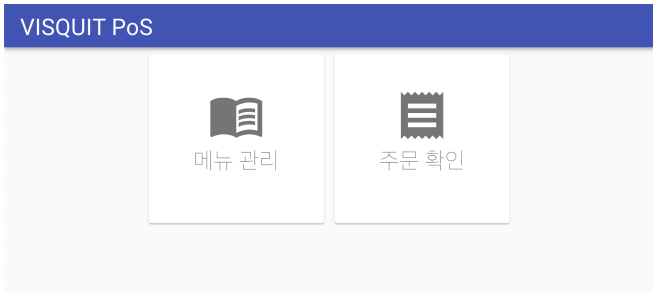

FIGURE 8. INITIAL LANDING PAGE OF FRONTEND UI

6.1.1. Initial Landing. When a user (typically, a restaurant's owner or manager) access our service, he or she will see this initial landing page. Then the user can choose which feature to use.

6.1.2. View Menu List. The user can view all menus registered in the user's context, where only menus served at the user's restaurant are shown. If the user click the arrow button on top-left side, the menu bar can be toggled to hide or be shown.

6.1.3. Add New Menu. When clicking 'Add New Menu' the button on the left, the user will be routed to menu adding page. The user can fill each forms and these values will be sent to the server.

FIGURE 9. VIEWING MENU LIST

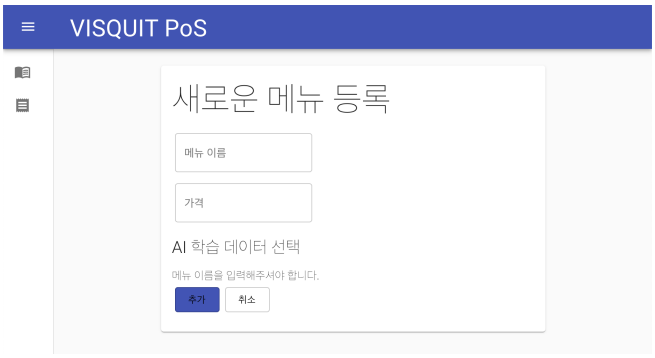

FIGURE 10. PAGE OF ADDING NEW MENU

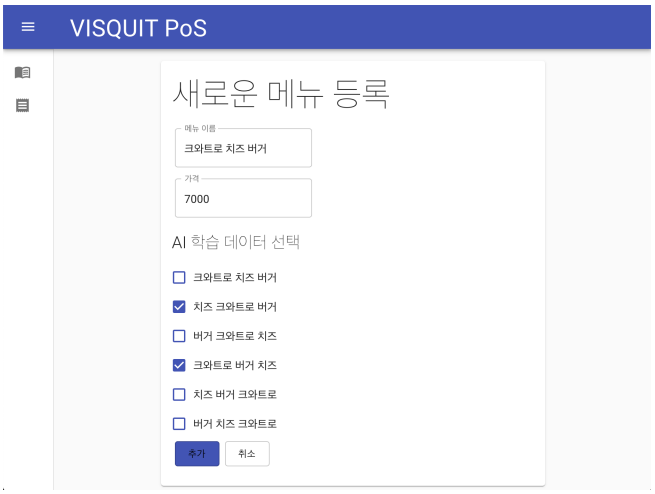

FIGURE 11. GENERATING STRING DATA FOR AI'S LEARNING

6.1.4. Generating String Data for AI's Learning. For SKTelecom's NUGU Speaker's AI to construct model for processing menu data, it is helpful to provide learning data of a new menu from frontend logic. When the user types in a new menu's name, that string will be used for generating all possible permutations of menu names. For example, if the user types in 'Quatro Cheese Burger', then the frontend logic will autometically generate 'Quatro Cheese Burger', 'Cheese Quatro Burger', 'Quatro Burger Cheese', ... etc. After that, the user can choose what to pass to the server

for learning.

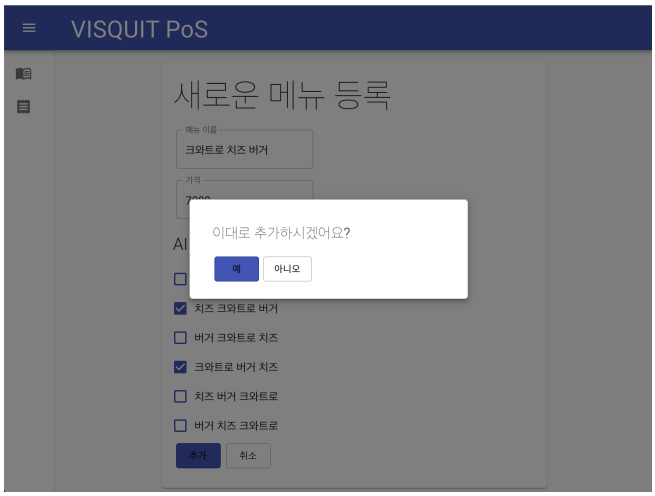

FIGURE 12. PROMPTING MODAL FOR EACH ACTIONS

6.1.5. Prompting Modal for each actions. The user's each actions that triggers interaction with REST API will be double-checked by frontend UI. If the user tries to create, update, or delete a menu, a modal window will be shown on the top of the UI and ask again. When the user clicks 'Yes', the intended action will be executed. Else, the action will be aborted.

FIGURE 13. EDITTING PREVIOUS MENU

6.1.6. Edit Previous Menu. When clicking a item on the 'View Menu List' page, the user will be routed to 'Edit Menu' page. The features provided is identitcal to 'Add New Menu', except the target menu already exists. A modal will appear when an action is triggered as well.

6.1.7. View Current Order List. When clicking 'Current Order List' button on the left-side menu, the user will be routed to the page where the user can view all orders pending.

6.1.8. Process Pending Orders. When clicking a item on the 'Current Order List' page, the frontend UI will double-check whether the particular order is really done. When the user click 'Yes', the order's status for ready will be changed to 'true'.

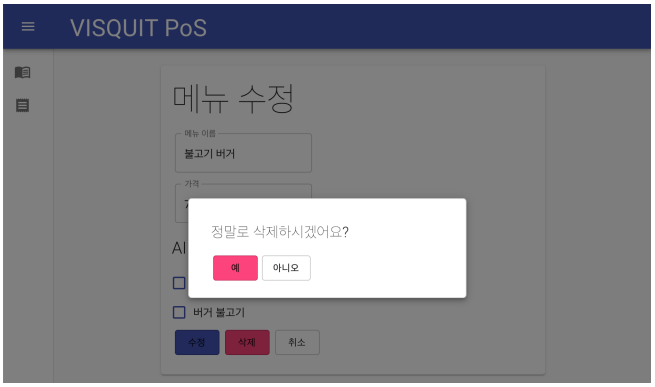

FIGURE 14. PROMPTING MODAL FOR EACH ACTIONS

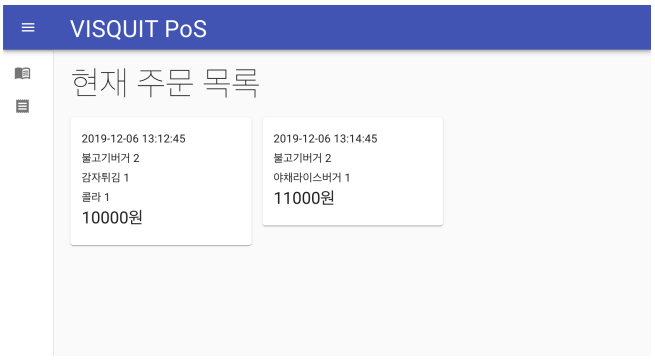

FIGURE 15. VIEW CURRENT ORDER LIST

FIGURE 16. VIEW CURRENT ORDER LIST