

IVR (INTERACTIVE VOICE RESPONSE) FOR PIZZERIAS

Submitted by:

EDWARD LE

CWID: 886654797



California State University, Fullerton

CPSC-597

Fall 2021

Advisor: Dr. Sampson Akwafuo

Department of Computer Science

California State University Fullerton



CALIFORNIA STATE UNIVERSITY
FULLERTONTM

Department of Computer Science

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the award of Master of Science degree in Computer Science.

IVR (Interactive Voice Response) For Pizzerias

Project Title

Edward Le

Student Name

Dr. Sampson Akwafuo

Advisor's Name

Advisor's signature

Date

None

Reviewer's name

Reviewer's signature

Date

Table of Contents

<i>Abstract</i>	6
Introduction	1
1.1 Motivation	1
1.2 Objectives	1
Literature Review	2
Methodology	5
Technologies	5
Single Sign-On (SSO)	5
Phone Gateway API	5
Order Online (Website)	6
Kitchen Screen	6
Admin Dashboard	6
System Block Diagram	7
Sequence Diagrams	8
Workflow of Single Sign-On	8
Workflow of Phone Gateway	9
Workflow of Website (Order Online)	11
Workflow of Kitchen Screen	11
Workflow of Admin Dashboard	12
Use cases	12
ER Diagram	13
Results and Discussion	13
Results	13
Installation	13
Database	15
GUI/Interfaces/Functions	20
Single Sign-On	20
Phone Gateway	22
Website	27
Kitchen Screen	30
Admin Dashboard	31
Discussion	33

Conclusion	35
Future Work	35
Limitations	35
<i>References</i>	36

Table of Figures

Figure 3.2.1: System block diagram	7
Figure 3.3.1.1: Workflow of Single Sign-On	8
Figure 3.3.2.1: Workflow of Phone Gateway (Overview)	9
Figure 3.3.2.2: Workflow of Phone Gateway (Place Order)	9
Figure 3.3.2.3: Workflow of Phone Gateway (Check Order)	10
Figure 3.3.2.4: Workflow of Phone Gateway (Cancel Order)	10
Figure 3.3.2.5: Workflow of Phone Gateway (Voice Recognition vs. Keypad)	11
Figure 3.3.3.1: Workflow of Website (Order Online)	11
Figure 3.3.4.1: Workflow of Kitchen Screen	11
Figure 3.3.5.1: Workflow of Admin Dashboard	12
Figure 3.4.1: User cases	12
Figure 3.5.1: ER Diagram	13
Figure 4.1.1.1: System Configuration	13
Figure 4.1.2.1: Database - Table Credentials	15
Figure 4.1.2.2: Database - Table Crust	16
Figure 4.1.2.3: Database - Table Hours	16
Figure 4.1.2.4: Database - Table Orders	17
Figure 4.1.2.5: Database - Table Pizzas	17
Figure 4.1.2.6: Database - Table Sauces	18
Figure 4.1.2.7: Database - Table Sizes	18
Figure 4.1.2.8: Database - Table Toppings	19
Figure 4.1.4.1.1: Interface of Single Sign-On (Sign Up Form)	20
Figure 4.1.4.1.2: Interface of Single Sign-On (Login Form)	21
Figure 4.1.4.1.3: Interface of Single Sign-On (Navigator)	22
Figure 4.1.4.2.1: Phone Gateway - Webhooks	22
Figure 4.1.4.2.2: Phone Gateway Console	25
Figure 4.1.4.2.3: Sample of Voice Recognition	25
Figure 4.1.4.2.4: Dictionary for Voice Recognition	26
Figure 4.1.4.3.1: Interface of Website (Homepage)	27
Figure 4.1.4.3.2: Interface of Website (Order page)	28
Figure 4.1.4.4.1: Interface of Kitchen Screen	30
Figure 4.1.4.5.1: Interface of Dashboard (Order details)	32
Figure 4.1.4.5.2: Interface of Dashboard (Working hours)	33

Abstract

IVR For Pizzerias is a customer helpline phone system for pizza restaurants. This system can handle up to 50 customers at the same time, aiding them in placing orders, checking order status as well as handling payments.

This system is a set of five synchronized applications that can assist pizzeria owners in maintaining communication closely with customers:

- 1. Single Sign-On (SSO)**

Single Sign-on (SSO) is an authentication application that enables users to log in once and access services without re-entering authentication factors.

- 2. Phone Gateway API**

Phone Gateway is used to transmit and receive voice communications and is capable of receiving hundreds of calls at the same time without interruption or waiting. This is the main component of this project.

- 3. Order Online (Website)**

A simple website built with the purpose of allowing customers to order pizzas online.

- 4. Kitchen Screen**

Kitchen Screen is a monitor located inside the production area that only displays received and queued orders from customers.

- 5. Admin Dashboard**

Admin Dashboard is reserved for administrators. It provides necessary reports and interfaces so that they are able to view the details of the orders, and to enable configuration of the entire system.

The five apps are built with different technologies. The specialization of each app will allow them to be developed independently. In addition, having each application operating separately increases the security of the entire system. This also stems from the practical needs of large operating systems. Particularly, a large system will require many features. A group of related features will be sent to a department for implementation. Breaking down a large system into many small systems will clearly define tasks for each department. This implementation is clearly described in the Agile architecture and is widely used. The benefits of this are increased project completion velocity, reduced maintenance costs, and more efficient staff turnover.

This project was made based on the real need from small restaurant owners during the Covid-19 pandemic, when the vast majority of stores switched to serving takeaway food in observation of laws and regulations. This project attempts to solve two problems for small restaurants: to make up for the shortage of workers, and to reduce the load on serving customers, thereby eliminating human errors.

The project has managed to put together a working system. The main and important functions are all successfully built and in working condition: Users can login with Single Sign On throughout the system. IVR in Phone Gateway API app can flexibly communicate with customers by voice and keypad. The website was able to allow customers to order online. The Kitchen Screen is expected to automatically update every time a new order is placed. In case of network connection problems, it is also set to refresh every two minutes to update orders. Admin Dashboard allows administrators to view detailed orders and configure the system when there is a need to. In order to scale, this dashboard will require more reports and analyses in the near future.

In addition, there are two successes resulting from this project. The first being the successful application of Google Cloud Voice Recognition technology to the IVR system. Google Cloud Voice Recognition is the leader in speech recognition technologies. It allows the IVR system to speak with customers in 125 languages with natural human voice. The other success is the synchronization of data across multiple applications. When applications use different platform technologies, it becomes difficult for them to communicate with each other. In order for them to exchange data with the lowest cost possible, we need to have a set of standards and build a solid common structure from the beginning.

Limited time, manpower and test subjects caused this project to fall short in terms of minor optimization issues. That is, this project is ready to be rolled out in restaurants, but for the lack of a few convenience features such as improved loading times and user freedom in customizing their flow of processes. However, the most important component of the project, IVR, has been completed. As the next step, if the POS/payment system is integrated, this IVR can be used at restaurants across the United States.

Introduction

1.1 Motivation

The main variable of interest in this thesis is small pizzerias. Most of the time, as small businesses, they hardly focus on improving the capabilities of order receipts by phone as there were plentiful customers making orders in person. This should not have been a major problem if the pandemic had not hit so suddenly. To ensure public safety, most governments decided to force restaurants to shut down in-person capability, which, for many small businesses, is their main lifeline. Instead, they now find themselves stranded with having to fulfill orders for distant customers, and to make it worse, with a limited amount of staff in compliance with social distancing guidelines. As a result, their outdated phone lines were overloaded with customer calls only a few days after the new directives came into effect. Unfortunately, those without the capabilities to support such a large volume will suffer as most calls will be dropped, leading to lost opportunities for revenues for already struggling establishments. With this in mind, this thesis attempts to propose a cost-effective solution for these small businesses to capture all the potential orders and make the best use of their facilities in this trying time and most likely other disasters that are to come unexpectedly in the future.

The second motivation for this project is to harness the advances in technology to improve business operations efficiency. Particularly, in recent years, Google AI has made incredible progress in trying to understand human language and can be taught to accomplish objectives when left on its own. It is with the highest hope that the implementation of this project will utilize all the benefits that are freely available from Google to forge a comprehensive tool that can help small businesses survive and continue to grow in the form of an automated system that organizes the receipt of orders to free up more manpower, reduce human errors, present them to the departments in charge in a timely manner, and properly archive such orders afterwards for management purposes.

1.2 Objectives

The core objective of this thesis is the implementation of an example simple set of information technology applications that enables small pizzerias to reduce workers from 30 up to 50% and help minimize the chance of missed orders during peak hours. The following is a list of components of the system:

1. Website: Helping customers place orders online
2. Phone gateway + IVR: Receiving orders over the phone
3. Kitchen screen: Automatically sending orders to the kitchen
4. Admin dashboard: Allowing users to view order details and manage system configuration

Literature Review

In a 2013 study about factors influencing customer acceptance of kiosks at quick service restaurants, it was revealed that more and more companies are using self-service technologies (SSTs), leading to rapid growths within the industry and that it would soon become a standard in the near future (Kim, 2013). Also in the study, it was concluded that fewer young people are considering interactions with employees important, and SSTs in general are more of a convenience and time saver (Kim, 2013). Thus, a successful SST must capture the customer's psychology. Ideally, it should bring about shorter wait times, the sense of control, and absolute privacy. The goal of using an SST is the freedom of personalization based on customer needs and satisfaction. Another vital role of using SST is to improve customer service by collecting feedback from customers at anywhere and anytime.

Robin B. DiPietro describes the third generation of restaurants in his research titled "The Use of Social Networking Sites in the Restaurant Industry: Best Practices" as using technology for customer profiling, online reservations, labor and inventory management (DiPietro, 2012). These are examples of how technology can be used for analyzing menu trends and menu items forecast to determine items to add or remove from the menu. IVR and websites are the main sales channel of this generation. Along with text messages and emails, they are used as a marketing tool for communication with customers through coupons and menu updates. Based on these findings, a complete system would need to include websites, IVR, and POS (point of sale) as payment systems (Sehgal, 2018).

The two experiments of James C. Mundt in the article "Psychological performance assessment via interactive voice response systems" demonstrated that customers are calm and patient when talking with IVR, and the system in turn helps restaurant owners increase in confidence while making customers feel they are being treated in a professional manner using cutting edge technology (Mundt, 2018). In the same article, Mundt also points out that the use of IVR as well as the application of technology will help restaurants and startups shape their operating models in a methodical manner (Mundt, 2018). By taking orders step by step through IVR, the system helps avoid errors when making orders, allowing restaurant owners to streamline the production process within one quarter, which is faster than the overall average timeline of six to twelve months.

The use of IVR and other automated systems also has a great value to society, helping to reduce elder mistreatment, racism, and discrimination. Scott R. Beach has indicated that in one out of every sixteen phone conversations, an elderly person is treated with disrespect, and calls are even hung up when young people lose their patience (Beach, 2010). The target audience for this study was adults 60 years of age or older who have landlines and live within the Allegheny County (Pittsburgh), Pennsylvania. The same situations apply when the victims are immigrants with thick accents. Particularly, employees with accents often receive criticism from unfriendly customers when taking phone calls. The rate is one out of twenty-five. With this in mind, IVR and automated systems will eliminate the possibility of people sharing the same classifications or backgrounds being maltreated.

Keith Kirkpatrick analyzes recent advances in artificial intelligence and appraises them for creating significant benefits for IVR systems in call centers (Kirkpatrick, 2017). In previous years, rule-based decision matrices (such as "press 1 for sales, press 2 for technical support") were popular systems for customer service providers. However, in recent years, artificial intelligence can do more than just recognize speech by patterns. In fact, artificial intelligence technologies, including machine learning, natural language processing, and even sentiment analysis are being strategically integrated into IVR systems to improve overall user experience (Kirkpatrick, 2017).

Kirkpatrick states that the trend is to create products that can replace the time-consuming or costly manual work. In the article "AI in Contact Centers," the author alleges that "AI is being used by customer contact centers as a contextual knowledge management system," so the system can learn the most appropriate responses to questions, for use in its automated bots and to train other agents (Kirkpatrick, 2017). In addition, as artificial intelligence technologies for personal devices become widespread (such as Siri, Alexa and Google Home Assistant, etc.), voice systems would need to evolve as well.

In a technological paper called "Speech to text and text to speech recognition systems - A review," Ayushi Trivedi comprehensively identified the keys of Speech Recognition Systems being keywords and vocabulary (Trivedi, 2018). From the beginning, speech can be converted into text. Text is then sent to pattern-matched models for data extraction. The recognition accuracy of speech depends on how large the sample dataset is. Also, there are two established and emerging technologies in speech recognition: Text-to-Speech and Speech-to-Text. Trivedi confirms that the former technology works better than the latter (Trivedi, 2018). With many techniques being applied to creating robot voices, they are becoming very accurate and similar to the natural human voice. Humans and robots differ in emotions. When robots can recognize human emotions or bring emotions into conversations with humans, then the distance between humans and robots is shortened (Yacoub, 2008).

Automatic Speech Recognition has been developed since 1952. However, this technology was not made popular until 2010 from the two apps Google Voice Search and Siri. As a pioneer, Google provides Speech Recognition API for the community to use. In 2019, Bogdan Iancu conducted tests on 3000 simple English phrases. The results of the experiment described in the paper "Evaluating Google Speech-to-Text API's Performance for Romanian e-Learning Resources" are as follows: mean accuracy of 89.4 for native speakers and 65.7 for non-native, which is considered the leader of the industry (Iancu, 2019). Interestingly, Google Cloud Speech-to-Text API supports 125 languages. This might be a huge benefit for programmers, as developers do not necessarily need to know the languages, yet they can still write cross-language applications.

Single Sign On (SSO) protocols have become popular and widely used today. SSO is not only reserved for private applications, it is used to connect together systems which are not from the same developers. Third-party identity providers trusted by many big companies can authenticate end users and allow them to access multiple services. JSON Web Token is one of the outstanding technologies that can be used in Single Sign On solutions. In a 2018 study by Pratama, it was revealed that SSO with JWT is a robust option from the fact that "by exploiting

specialty of JWT that can contain entities, roles of users can be sent simultaneously with the authorization in a single token. Beside that, JWTs can represent a set of claims between 2 parties , hence a JWT token has an ability to authenticate the sender of the request on the receiver. That is in line with the architecture of token-based SSO. We compare performance the proposed SSO to a non-JWT based SSO that usually uses an encryption token. We also compare a token of JWT containing roles to a token of JWT without roles and a token of non-JWT. Therefore, we show performance of the tokens and observe effect of inserting roles of users in a token of JWT” (Pratama, 2018). The same research also pointed out that SSO with JWT has been consistently utilized by government agencies such as voting commissions alongside e- and c-government contractors, which stand for electronic and cloud governments respectively (Pratama, 2018). Ultimately, if such an option is trusted on a national security level, its inclusion will simplify the verification process of the entire system, as users will only need to log in once across all four independent components of the application while at the same time reducing the amount of extra procedure put into authenticating already trusted sources. Although its sole purpose for the time being is to simplify log in procedures for a relatively small environment, it is with a firm belief that SSO embedded with JWT will prove to be even more useful in the future if there are plans to incorporate other functionalities into the pipeline such as food delivery services (i.e. Uber Eats, Grubhub, Postmates and so on) as it ensures data flows smoothly across all the channels involved.

Methodology

1. Technologies

1.1. Single Sign-On (SSO)

Single Sign-on (SSO) is an authentication application that enables users to log in once and access services without re-entering authentication factors.

Technical information:

- Login form: HTML, CSS3, Javascript
- Backend: PHP, JSON Web Tokens (JWT)
- Database: MySQL

SSO validates sets of credentials by using PHP and MySQL, just like other normal authenticators. After a user has a successful login, JSON Web Token (JWT) generates a token for this user. This token contains all the user's information, including IP and expiration.

When users access our services, such as Admin dashboard, Kitchen screen, Phone Gateway API, and website, which require authentication, the current system sends a request to SSO to confirm access authority. If the user is logged in, SSO will return a trusted token. The current system will create a session and grant permission to this user.

1.2. Phone Gateway API

Technical information:

- Vonage Nexmo APIs
- Google Text-to-Speech
- Google Speech-to-Text
- NodeJS, ExpressJS
- Database: MySQL

Phone Gateway is used to receive information from a service provider about calls, and Vonage is the telecommunications service provider (TSP) for this project. The information that can be received from Vonage is: caller phone number, date and time, tracking number (uuid), voice and keys interactions, call records. Actions that we can send to a call: play a sound, forward call, interrupt call, request interaction, and disconnect. These services of Vonage allow programmers to get deep into a call. These services are sufficient to build a queue for incoming calls, navigate and store all the information about the call.

Developers have the capability to build a call center from scratch by using Vonage Nexmo APIs, that allows us to create unique experiences for our customers. Most of the similar services, from either Amazon or Microsoft, have a

ready-to-use call center that programmers don't have full access to to customize the existing system. Plus, its fee and cost are the cheapest. It means this project is a highly practical system.

An IVR system must need speech recognition services, from Google Cloud: Text-to-Speech, Speech-to-Text. These services allow programmers to analyze voice audio and navigate calls. Google Cloud Voice Recognition services provide a speech recognition system that supports 125 languages with a variety of voice patterns. It makes this project possible to deploy to many countries around the world, not just the US.

1.3. Order Online (Website)

Technical information:

- Front-end: HTML, Bootstrap 5
- Back-end: PHP
- Database: MySQL

Although ordering pizzas over the phone is the traditional way of Americans, ordering through a website is still an option used by many young people today. The need to have a website for a business is inevitable, but the main part of this project is not creating a website for the pizza restaurant. The website was created simply as another pizza ordering option to demonstrate the system's data synchronization.

The website consists of two simple pages: the main page and the order page. The main page will simply be a static web page. The ordering page is modeled after the action of ordering pizza over the phone.

1.4. Kitchen Screen

Technical information:

- Front-end: Javascript
- Back-end: NodeJS, SocketIO
- Database: MySQL

Kitchen Screen is a display of unprepared orders. Its orders are displayed in real time. Its interface design must be simple and only display the necessary information about the order. The only action for this function is for the chef to set an order as cooked and ready for delivery.

1.5. Admin Dashboard

Technical information:

- Front-end: HTML, Bootstrap 5
- Back-end: PHP
- Database: MySQL

Admin Dashboard is for administrators, the only interface that can be configured for the whole system. Admin dashboard is used to set up business hours for the system. It is also used to view order details and modify orders.

2. System Block Diagram

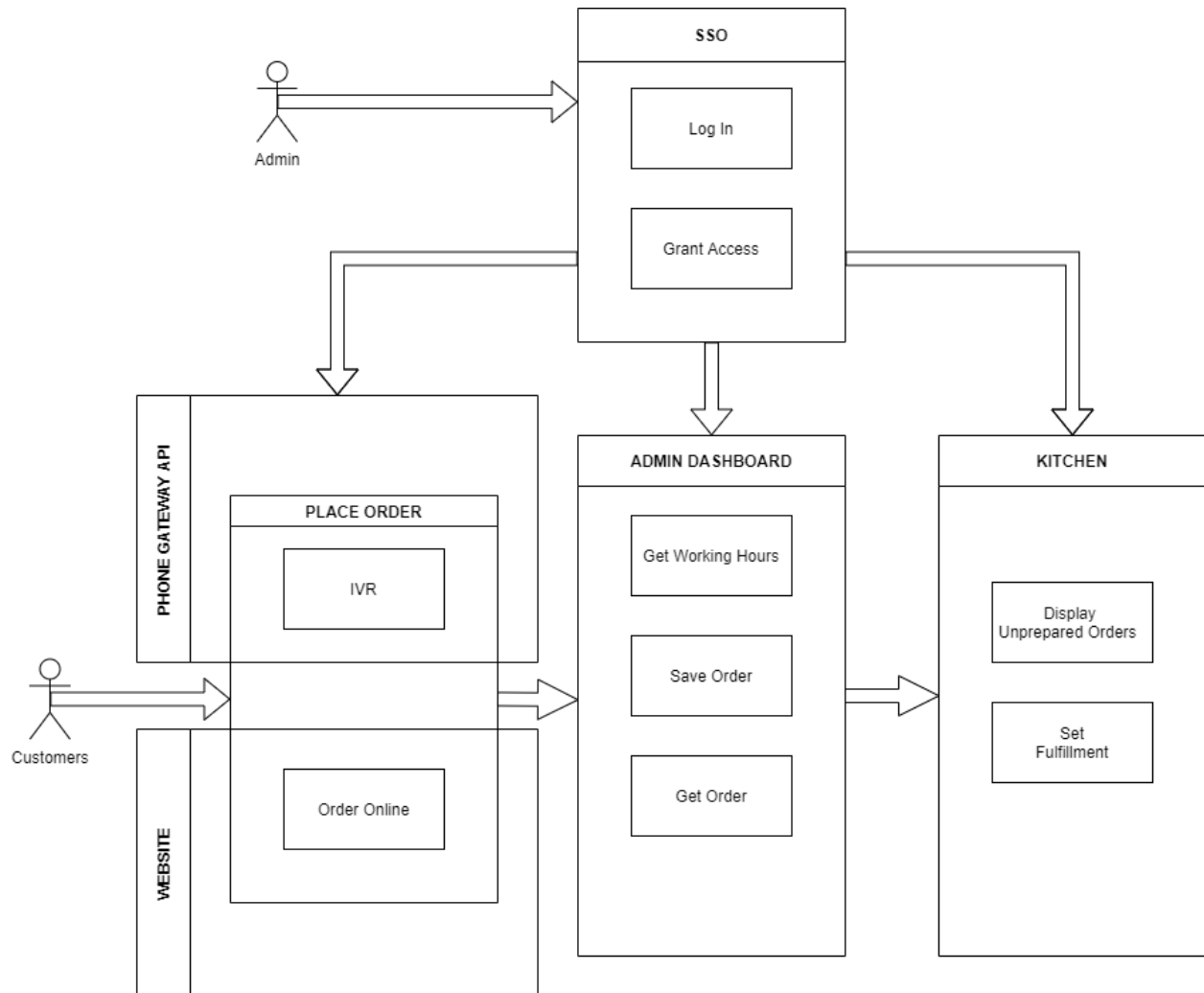


Figure 3.2.1: System block diagram

SSO: It will have two main functions: login and grant permissions. Since each application requires authorization, the need for Single Sign On to grant shared access permission for each application. Whenever an application needs the user to be authorized, the user is sent back to SSO for login. And if the user has successfully logged in, SSO already has the user's credentials. The system will automatically grant permissions to those users in the new application.

Place Order: Orders can be created through the website or the phone system (Phone Gateway). The IVR in Phone Gateway will be able to take orders by voice or by keypad. Artificial intelligence helps improve the user experience and make the system easier to use. The website

will be just a simple shopping cart where users can create their own pizzas. Users interact directly with the website and Phone Gateway.

Admin Dashboard: It includes two main functions: order modifying, and configuration for Phone Gateway. An order will be stored in the database and displayed on the Admin Dashboard and the screen in the Kitchen. Phone Gateway needs information to know the opening and closing hours of the restaurant, as well as customer information from incoming calls.

Kitchen: It is short for a monitor placed in the kitchen. There are two functions: displaying unprepared orders in real time and setting the status of orders by the chefs.

3. Sequence Diagrams

3.1. Workflow of Single Sign-On

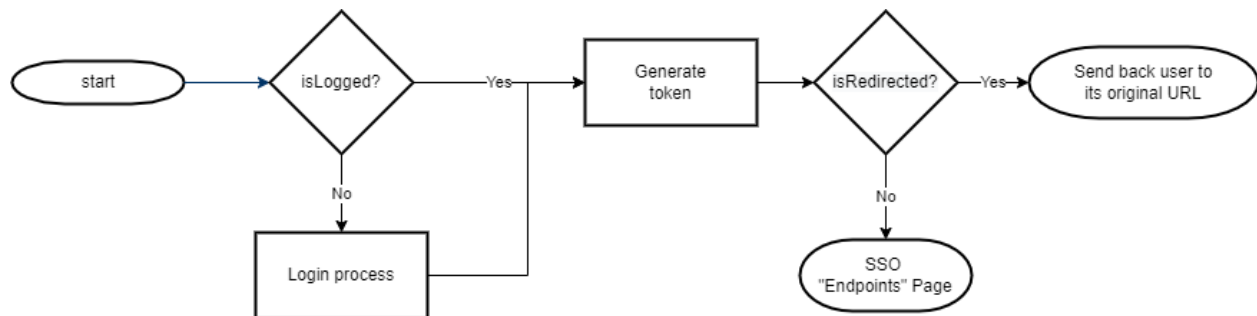


Figure 3.3.1.1: Workflow of Single Sign-On

Users have to log in at SSO. User's credentials will be stored in SSO. The data stored includes user id, email, password, IP, expiration time, and a token is generated with authorization information. SSO authorizes an application by sending a token to that application.

3.2. Workflow of Phone Gateway

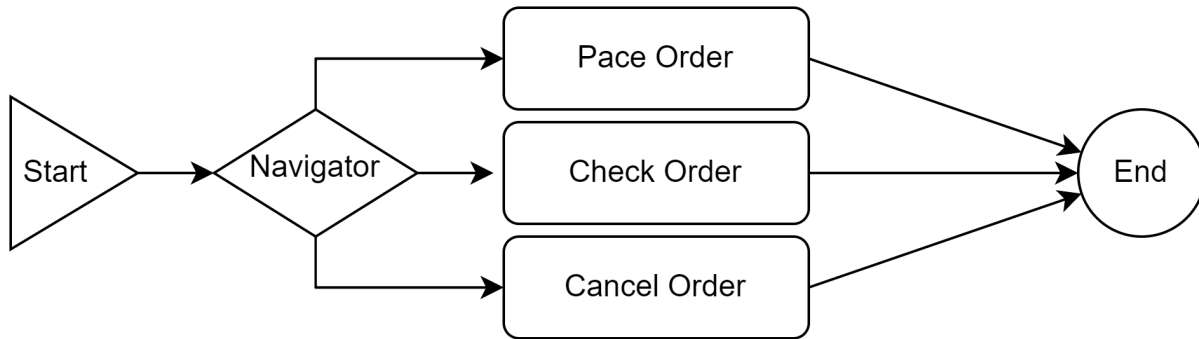


Figure 3.3.2.1: Workflow of Phone Gateway (Overview)

Phone Gateway only works with incoming calls. The IVR will start the call with a greeting. It also checks the current time. If the current time is in the business hours of the restaurant, the call is allowed to continue. If it is not, the call will be disconnected after a polite apology. Phone Gateway accesses the database to make sure that there are no orders associated with the current call. We understand that customers who do not have an order will want to order pizzas. Customers who are associated with an order will want to check or cancel the order.

In addition, this module has an option that allows customers to talk with real people as a representative; however, this project is a demo and there is no operator to take calls.

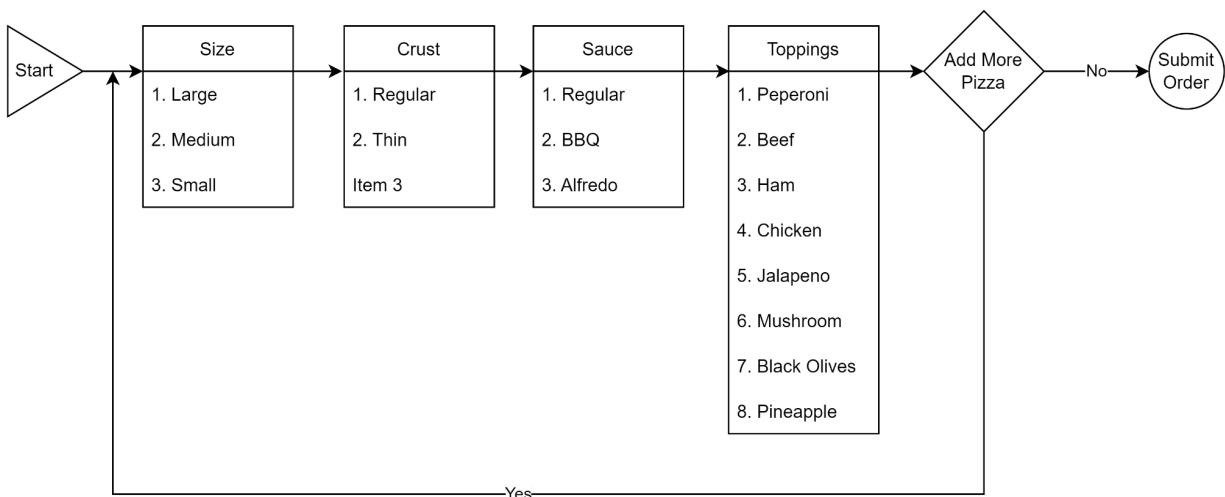


Figure 3.3.2.2: Workflow of Phone Gateway (Place Order)

Ordering pizza must be done step by step. One order will include one pizza or more. There should be a loop for the customer to make each pizza. There are four steps to customizing a pizza: choose the size, choose the crust, choose the sauce, and choose

the toppings. Choosing toppings is the most difficult part of this function. Customers can choose from 1 to 8 toppings on the menu. Our system will listen and save the toppings as a list. The IVR will take one customized pizza at a time, and when the user does not order more pizzas, the order will be considered complete.

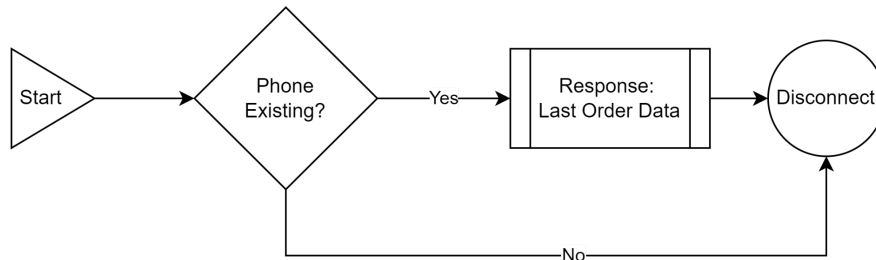


Figure 3.3.2.3: Workflow of Phone Gateway (Check Order)

To check the status of an order, it is simple to check if the phone number of an incoming call is associated with an order in our database. Then IVR can answer the search results to the call.

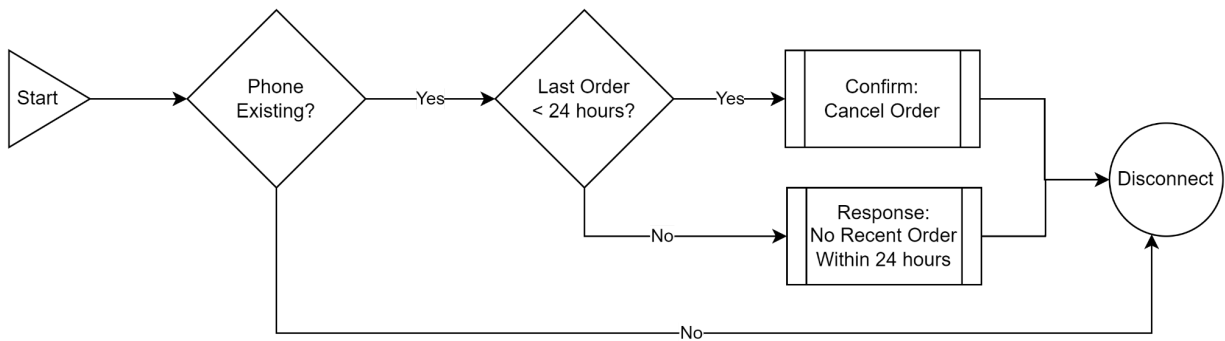


Figure 3.3.2.4: Workflow of Phone Gateway (Cancel Order)

Phone number of the incoming call must be associated with a valid order for cancellation. A valid order for canceling is an order that must be created within 24 hours. In fact, most restaurants are only allowed to cancel orders within hours or less.

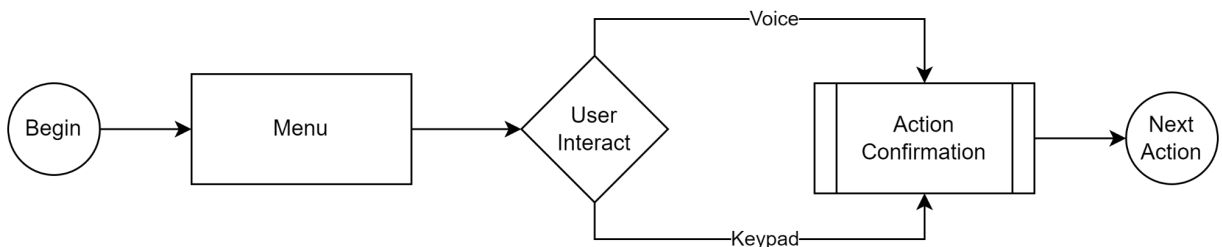


Figure 3.3.2.5: Workflow of Phone Gateway (Voice Recognition vs. Keypad)

For convenience and to encourage users to use voice recognition, interactions with the IVR at all steps can be done using voice or keypad. In this project, the IVR will

use sentences that encourage users to respond by voice. When the IVR cannot recognize the voice response, the IVR will give instructions to use the keypad to users. During the conversation with the IVR, users will always have the option to respond by voice or using the keypad.

3.3. Workflow of Website (Order Online)

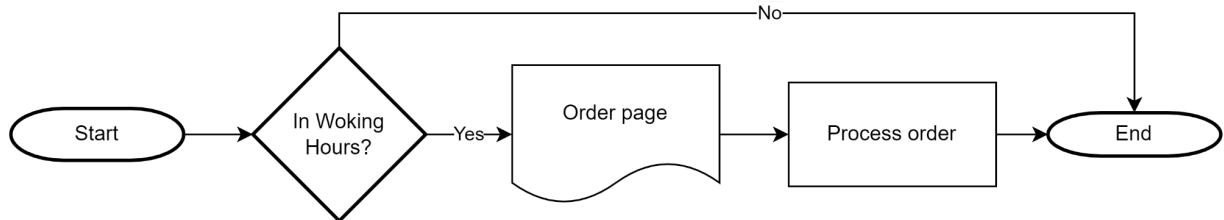


Figure 3.3.3.1: Workflow of Website (Order Online)

Since pizza can't last long, it's essential to check the restaurant's operating hours before taking an order. Similar to the IVR, customers can easily customize their pizzas on the website by choosing toppings and sauce. After submitting an order of pizzas, the order will be saved to the database and return the Order Number to the user.

3.4. Workflow of Kitchen Screen

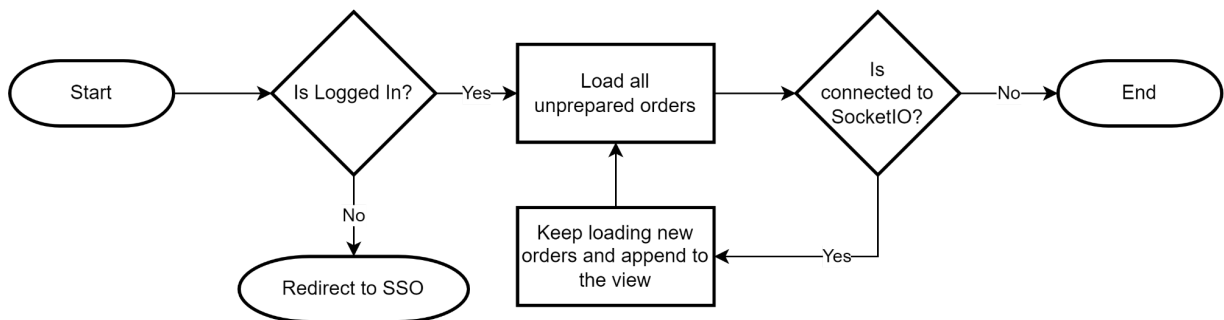


Figure 3.3.4.1: Workflow of Kitchen Screen

Kitchen Screen is a real time application. New orders from the database will be displayed in real time so chefs can prepare orders. This application is written in javascript for the front-end and uses SocketIO for the back-end. SocketIO sets up dedicated rooms for each department. Each monitor connected to the same room will display the same data.

3.5. Workflow of Admin Dashboard

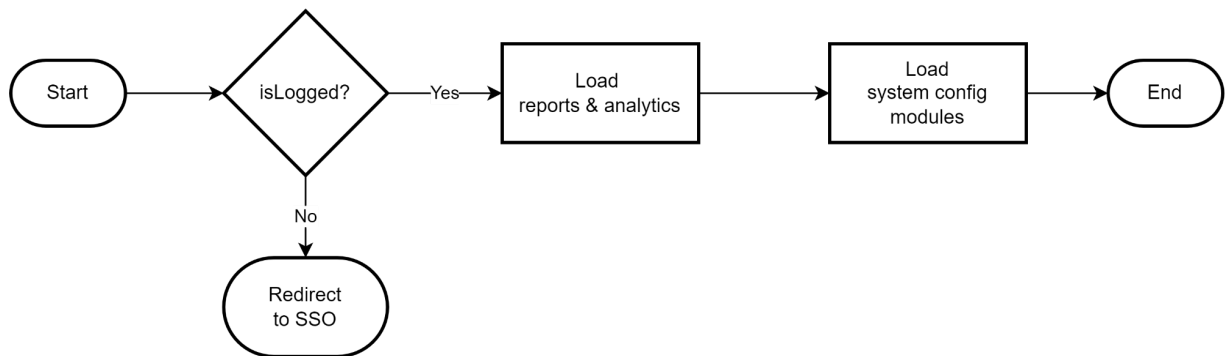


Figure 3.3.5.1: Workflow of Admin Dashboard

Admin Dashboard is an extra tool for management of the system. Admin can view order details and modify orders. Admin can also set up business hours of the store.

4. Use cases

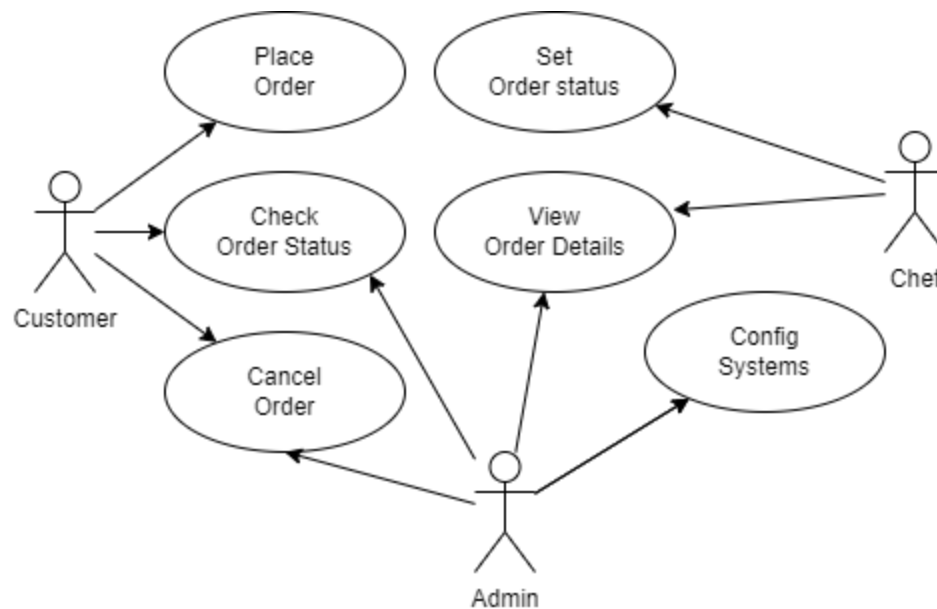


Figure 3.4.1: User cases

There are three actors in this system:

- **Customer:** Customers will place orders, check status of their orders, and even cancel orders. They interact with the IVR through the Phone Gateway or can also tune in to the website to order online.
- **Chef:** Chefs need to see unprepared orders in real time and will also need to change the status of the orders as they become available for pick-up.
- **Admin:** Administrators can view all order details, modify and even cancel them. Administrators are also in charge of configuring this system.

5. ER Diagram

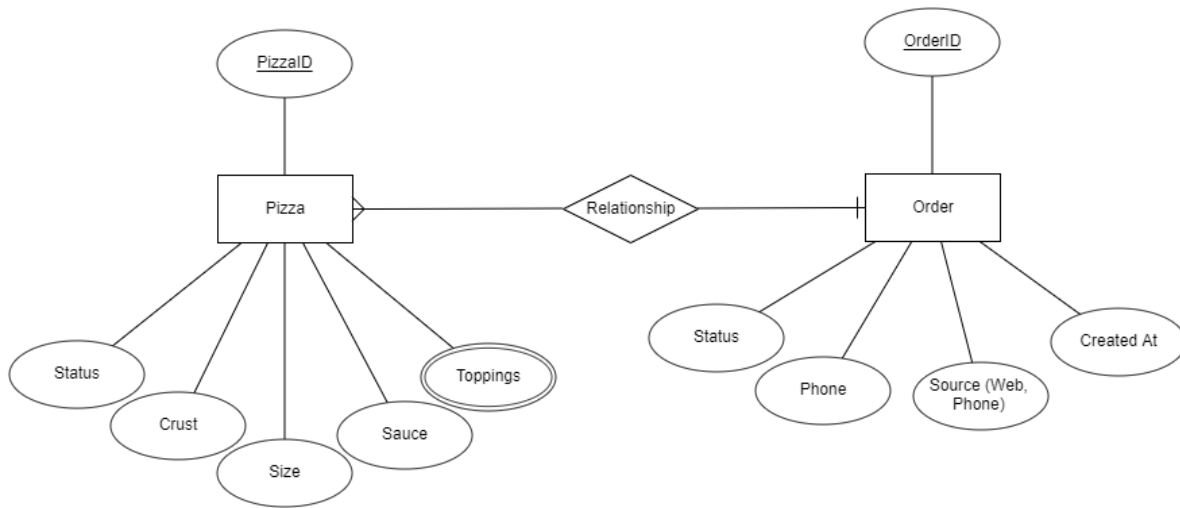


Figure 3.5.1: ER Diagram

The relationship between order and pizza is one-to-many. An order can have one or more pizzas. Each pizza belongs to only one order. Order can only be canceled if all pizzas in the order are cancelled. When canceling an order, all pizzas belonging to that order are cancelled.

Results and Discussion

1. Results

1.1. Installation

Software	Version
NodeJS	10.18.0
NPM	8.0.0
PHP	7.4.23
MySQL	14.14 Distrib 5.6.51
SocketIO	4.4.0

Figure 4.1.1.1: System Configuration

This system uses NodeJS, PHP, MySQL, and SocketIO. Make sure Node and NPM are installed and their PATHs defined. You also need a web server for

running PHP programs like WAMP & XAMPP. There are 5 applications in this system, located in 5 folders: *admin*, *gate*, *kitchen*, *shop*, *sso*. It also requires a *cs597.sql* file.

Step 1: Import database to MySQL

- Create a database *cs597*.
- Note your *username* and *password* for later use.
- Import file *cs597.sql* to your database *cs597*.
- You can check the sample data in Chapter 4.1.2 of this document.

Step 2: Setup Admin Dashboard

- Move the folder *admin* to your PHP server. Usually, the destination will be inside folder *htdocs* (XAMPP) or folder *www* (WAMP or Linux server).
- Configure database credentials in file *connection.php*

Step 3: Setup Phone Gateway

This step is a bit difficult, please pay attention to follow the instructions.

- First of all, you need a Vonage developer account. Sign up for a free account at the link:
<https://dashboard.nexmo.com/>
- After logging in, you need to buy a phone number for taking calls. Then, you must create an application to use Vonage APIs for your system. Let's choose a random name for the application. Note that you don't have any webhooks yet, but we will set up some URLs in the next step. Now you can assign the phone number to the application.
- To run Phone Gateway, you need a live NodeJS server or you can use Ngrok as a testing server. Learn more about Ngrok at:
<https://ngrok.com/>
- Move the folder *gate* to your NodeJS server, then you will have some webhooks to copy into your Vonage application:
 - Answer URL: https://your_domain/answer
 - Event URL: https://your_domain/event
 - Fallback URL: https://your_domain/fallback
- Remember to update your database credentials in file *connection.js*
- To run Phone Gateway, use command *node index.js* in your directory.

Step 4: Setup Kitchen Screen

- Move the folder *kitchen* to your NodeJS server.
- Configure database credentials in file *connection.js*
- To run your web server, using command *node index.js*

Step 5: Setup Website

- Move the folder *shop* to your PHP server. Usually, the destination will be inside folder *htdocs* (XAMPP) or folder *www* (WAMP or Linux server).
- Configure database credentials in file *connection.php*

Step 6: Setup SSO

- Move the folder *sso* to your PHP server. Usually, the destination will be inside folder *htdocs* (XAMPP) or folder *www* (WAMP or Linux server).
- Configure database credentials in file *dbConnect.php*

Now, your system is ready when the five applications are running and ready for business. Your system will have five addresses (local) or five domain's names (live servers). Note that, if you don't do any configuration, your two NodeJS applications will run on port 3000.

1.2. Database

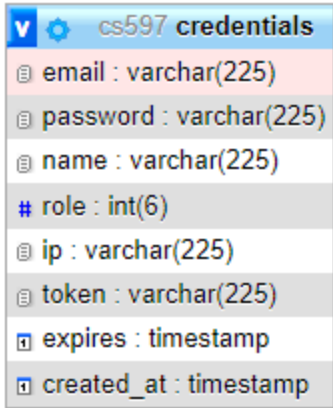
Table	Sample data																
	<table> <tr> <td>email</td><td>admin@cs597.com</td></tr> <tr> <td>password</td><td>\$2y\$10\$2pMKNR...</td></tr> <tr> <td>name</td><td>Ed Le</td></tr> <tr> <td>role</td><td>1</td></tr> <tr> <td>ip</td><td>45.51.23.144</td></tr> <tr> <td>token</td><td>eyJhbGciOiJIUzI1NiJ...</td></tr> <tr> <td>expires</td><td>2021-09-28 00:37:49</td></tr> <tr> <td>created_at</td><td>2021-09-28 00:37:49</td></tr> </table>	email	admin@cs597.com	password	\$2y\$10\$2pMKNR...	name	Ed Le	role	1	ip	45.51.23.144	token	eyJhbGciOiJIUzI1NiJ...	expires	2021-09-28 00:37:49	created_at	2021-09-28 00:37:49
email	admin@cs597.com																
password	\$2y\$10\$2pMKNR...																
name	Ed Le																
role	1																
ip	45.51.23.144																
token	eyJhbGciOiJIUzI1NiJ...																
expires	2021-09-28 00:37:49																
created_at	2021-09-28 00:37:49																

Figure 4.1.2.1: Table Credentials

This data table “*credentials*” will store the login information of the whole system. *Username* is email. The *password* is encrypted. The *role* will correspond to 1 as developer, 2 as admin, 3 as mods, and 0 as normal user. We also save the last *IP* of users, as well as the *token* for authorization. The *expires* is expiration of user's session.

Table	Sample data

Figure 4.1.2.2: Table Crust

This data table "crust" will store information on pizza crust types. There are only two types of crust: regular and thin.

Table	Sample data

Figure 4.1.2.3: Table Hours

This data table "hours" will store information on business hours. Customers are only allowed to order pizza during store hours. Currently only 7 rows for 7 days of the week, but if you can change as you like. For example, on Monday, the store hours are 7am to 12pm and 5pm to 9pm. You will need two rows of Monday, the first row has timein at 7:00 and timeout at 12:00, and the second row has timein at 17:00 and timeout at 21:00.

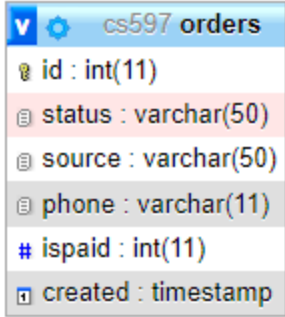
Table	Sample data												
	<table> <tr><td>id</td><td>1639027459</td></tr> <tr><td>status</td><td>active</td></tr> <tr><td>source</td><td>ivr</td></tr> <tr><td>phone</td><td>7146035555</td></tr> <tr><td>ispaid</td><td>1</td></tr> <tr><td>created_at</td><td>2021-12-08 21:14:34</td></tr> </table>	id	1639027459	status	active	source	ivr	phone	7146035555	ispaid	1	created_at	2021-12-08 21:14:34
id	1639027459												
status	active												
source	ivr												
phone	7146035555												
ispaid	1												
created_at	2021-12-08 21:14:34												

Figure 4.1.2.4: Table Orders

This data table "orders" will store information about orders. The order *id* will generate a special way. It is a string of Unix timestamps in seconds. The *status* is *active/canceled/ready/delivered*. The *source* is *ivr/online*. The *phone* is the customer's phone number.

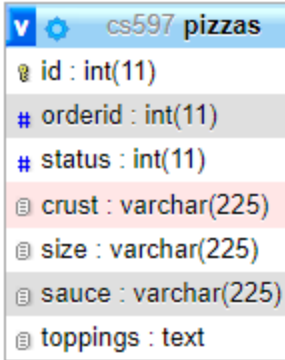
Table	Sample data														
	<table> <tr><td>id</td><td>1</td></tr> <tr><td>orderid</td><td>1639027459</td></tr> <tr><td>status</td><td>unprepared</td></tr> <tr><td>crust</td><td>regular</td></tr> <tr><td>size</td><td>medium</td></tr> <tr><td>sauce</td><td>regular</td></tr> <tr><td>toppings</td><td>peperoni, mushroom</td></tr> </table>	id	1	orderid	1639027459	status	unprepared	crust	regular	size	medium	sauce	regular	toppings	peperoni, mushroom
id	1														
orderid	1639027459														
status	unprepared														
crust	regular														
size	medium														
sauce	regular														
toppings	peperoni, mushroom														

Figure 4.1.2.5: Table Pizzas

This data table "pizzas" will store information about all pizzas of an order. This is a one-to-many relationship, an order will contain one or more pizzas. The *orderid* is a foreign key referenced from the data table *orders*. The *status* is

unprepared/cooked/canceled. Since we cannot change pizza's attributes, we do not need to create foreign keys for the attributes *crust*, *size*, *sauce*, and *toppings*.

Table	Sample data												
<div><div><div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div>cs597</div><div>sauc</div></div></div><div><div><div>🔑</div><div>id : int(11)</div></div><div><div>📄</div><div>name : varchar(225)</div></div><div><div>#</div><div>price : double(10,2)</div></div></div></div>	<table><tr><th>id</th><th>name</th><th>price</th></tr><tr><td>1</td><td>Regular Sauce</td><td>0.00</td></tr><tr><td>2</td><td>BBQ Sauce</td><td>1.00</td></tr><tr><td>3</td><td>Alfredo Sauce</td><td>1.30</td></tr></table>	id	name	price	1	Regular Sauce	0.00	2	BBQ Sauce	1.00	3	Alfredo Sauce	1.30
id	name	price											
1	Regular Sauce	0.00											
2	BBQ Sauce	1.00											
3	Alfredo Sauce	1.30											

Figure 4.1.2.6: Table Sauces

The data table “*sauces*” stores the sauces for a pizza. There are three types of sauces: regular, barbeque, and alfredo.

Table	Sample data												
<div><div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div>cs597 sizes</div></div></div><div><div><div>🔑 id : int(11)</div><div>📄 name : varchar(225)</div><div># price : double(10,2)</div></div></div></div>	<table><tr><th>id</th><th>name</th><th>price</th></tr><tr><td>1</td><td>small</td><td>2.00</td></tr><tr><td>2</td><td>medium</td><td>4.00</td></tr><tr><td>3</td><td>large</td><td>6.00</td></tr></table>	id	name	price	1	small	2.00	2	medium	4.00	3	large	6.00
id	name	price											
1	small	2.00											
2	medium	4.00											
3	large	6.00											

Figure 4.1.2.7: Table Sizes

The data table “*sauces*” stores the sizes of a pizza. There are three sizes: small, medium and large.

Table

cs597 toppings

🔑 id : int(11)

📄 type : varchar(50)

📄 name : varchar(225)

price : decimal(10,2)

Sample data

id	type	name	price
1	meat	pepperoni	0.20
2	meat	beef	0.20
3	meat	ham	0.20
4	meat	chicken	0.20
5	veggie	jalapeno peppers	0.20
6	veggie	mushroom	0.20
7	veggie	black olives	0.20
8	veggie	pineapple	0.20

Figure 4.1.2.8: Table Toppings

The data table “toppings” stores all the toppings of a pizza. There are two categories for toppings: meat and veggie. There are eight toppings: pepperoni, beef, ham, chicken, jalapeno, mushroom, black olives, and pineapple.

1.3. Create Users

A user can be created using the Sign-Up form (*Single Sign-Up*) at the Single Sign-On application. Note that the password will be “one-way” encrypted and a JWT token will be generated. The user created will be assigned a default role 0 as a regular user. Developers are role 1 and administrators have a role of 2.

1.4. GUI/Interfaces/Functions

1.4.1. Single Sign-On

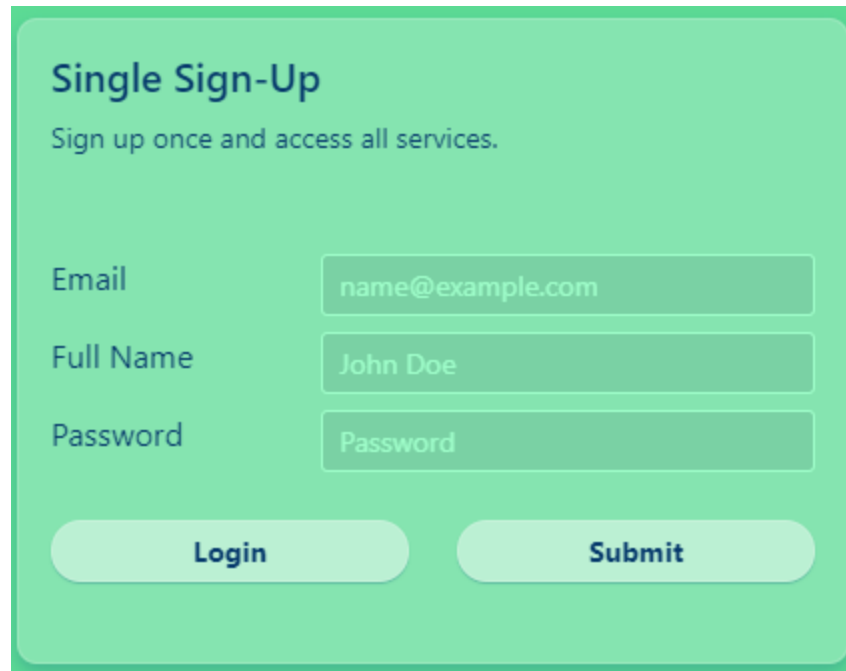
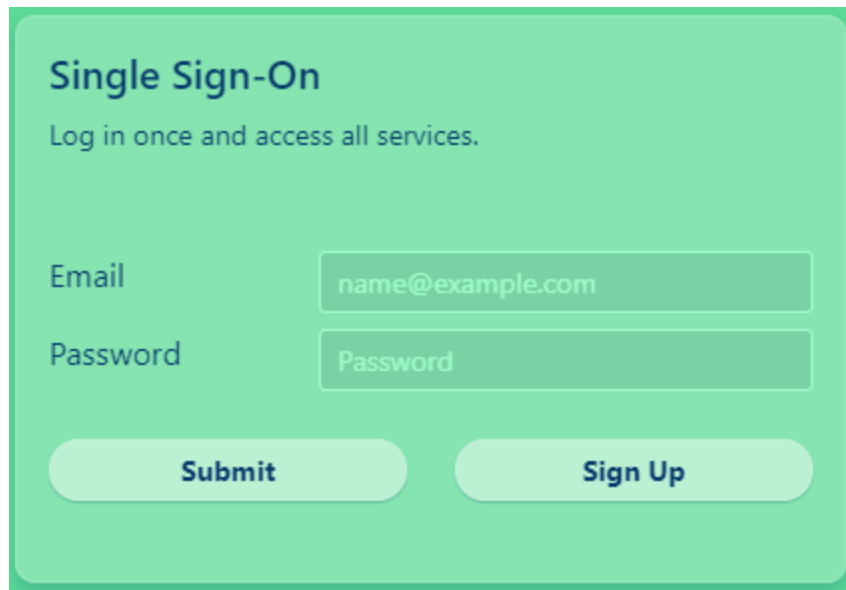
A screenshot of a 'Single Sign-Up' form. The form has a light blue header with the title 'Single Sign-Up' and the subtitle 'Sign up once and access all services.' Below the header, there are three input fields: 'Email' with the placeholder 'name@example.com', 'Full Name' with the placeholder 'John Doe', and 'Password' with the placeholder 'Password'. At the bottom of the form, there are two buttons: 'Login' and 'Submit'.

Figure 4.1.4.1.1: Interface of Single Sign-On (Sign Up Form)

The Single Sign-On has three interfaces. This Sign-Up Form requires an email address as username. The password is encrypted by using a strong one-way hashing algorithm:

```
$user->name = $data['name'];  
$user->email = $data['email'];  
$password = $data['password'];  
$user->password = password_hash($password, PASSWORD_DEFAULT);
```

The second interface of this application is the Log-In form:



The image shows a 'Single Sign-On' login form with a light green background. At the top, the title 'Single Sign-On' is in bold, followed by the subtitle 'Log in once and access all services.' Below this are two input fields: 'Email' with the placeholder 'name@example.com' and 'Password' with the placeholder 'Password'. At the bottom, there are two buttons: 'Submit' and 'Sign Up'.

Figure 4.1.4.1.2: Interface of Single Sign-On (Login Form)

This interface is for the login page of Single Sign-On application. Upon successful login, the user will be redirected to the original application, which requires login permission from the user.

```
if(isset($row->email)){
    // echo("Email match");
    if (password_verify($this->password,$row->password)){
        $issueAt= time();
        $expirationTime = $issueAt + 60 * 60 * 24;
        $key = "myKey"; // private key
        $payload = array(
            'user_id' => $row->id,
            'name' => $row->name,
            'exp'=> $expirationTime,
        );
        $jwt = JWT::encode($payload,$key);
        return array(
            "message" => "Successful login",
            "jwt" => $jwt,
            "status"=> true,
        );
    }
}
```

This snippet is from the login function. After checking that the email address exists and the password is matched, this snippet generates

a JWT token for the user. The expiration time (*\$expirationTime*) is 24 hours from the time of login. After the expiration, the user must log in again.

The private key is *myKey*. This private key is used for the JWT (JSON Web Tokens) to generate a token. This private key must be the same in all applications for the JWT to verify a token is valid.

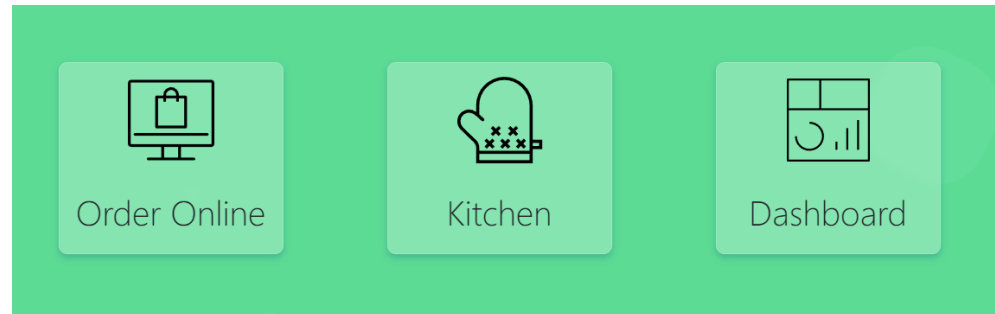


Figure 4.1.4.1.3: Interface of Single Sign-On (Navigator)

The last interface of SSO is Navigator. This interface will be displayed if the user is not sent to other applications. There are three buttons on this interface that will take users to the website, kitchen screen, and admin dashboard.

1.4.2. Phone Gateway

Phone Gateway needs to exchange information with the communication service providers Vonage through APIs. Therefore, we need some URLs on our system to send and receive data:

Answer URL ?	<div>HTTP GET ▾</div> <div>https://gate.edwardsle.com/answer</div>
Event URL ?	<div>HTTP POST ▾</div> <div>https://gate.edwardsle.com/event</div>
Fallback URL ?	<div>HTTP GET ▾</div> <div>https://gate.edwardsle.com/fallback</div>

Figure 4.1.4.2.1: Webhooks

- Event URL is used to receive call status information, such as: ringing, answer, busy,... If there are any changes during the call are sent to this URL. This URL is very helpful when you have multi-calls to the system at the same time. Assume that you have a queue of incoming calls and the number of phone agents is limited. When a call is disconnected, its status will be sent to the Event URL, and you can understand that an agent is available to take the next call in the queue.

- Answer URL is used to initial an incoming call. Vonage will send the information of the call to this URL and expect an action from our system in response to this call.
- Fallback URL is used when the Answer URL is not responding to the ping from Vonage.

Vonage APIs use a Call Control Object (NCCO) for call navigation. An NCCO is represented by a JSON array. For example, if we want to connect a call on line 714-660-6060 to an agent on line 714-340-5555, we send an NCCO to Vonage APIs as:

```
const ncco = [{  
  "action": "connect",  
  "from": "17146606060",  
  "endpoint": [  
    {  
      "type": "phone",  
      "number": "17143405555"  
    }  
  ]  
}];
```

The Vonage APIs provide eight actions for developers to control the flow of calls: record, create or join a conversation, connect, talk, stream, input, notify, and pay. An action must be completed before the next action can be performed. An NCCO can include multiple actions and it must be written as a valid JSON object.

```
const isInBusinessHours = () => {  
  const businessHours = getBusinessHours();  
  const todayName = getTodayName();  
  
  let openHour = new Date();  
  openHour.setHours(  
    businessHours[todayName]['openHr'],  
    businessHours[todayName]['openMin'],  
    0);  
  
  let closeHour = new Date();  
  closeHour.setHours(  
    businessHours[todayName]['closeHr'],  
    businessHours[todayName]['closeMin'],  
    0);  
  
  const currentTime = new Date();  
  
  if(currentTime >= openHour && currentTime < closeHour){  
    return true;  
  } else {
```

```

        return false;
    }
}

```

The *isInBusinessHours* function is a boolean function, and it is used to check if the current time is valid to receive a call or not. The *getBusinessHours* will return an array containing the opening and closing times of all days of the week from the database. The *getTodayName* will find out the name of the current day.

When a call comes in, the Vonage APIs will ping our Phone Gateway and request an NCCO at the Answer URL as our response:

```

// answer inbound calls
const TEXT_WELCOME = "
< speak>
    Welcome!< break time='1s' /> Thank you for calling Pizza Company.
</ speak>";

app.get('/answer', (req, res) => {
    let ncco = [{
        action: 'talk',
        bargeIn: false,
        text: TEXT_WELCOME,
        language: "en-US",
        style: 11
    }];
    ncco = ncco.concat(mainMenu());
    res.json(ncco);
});

```

This is how Text-to-Speech works. The text in tag `< speak>text</ speak>` will be converted to a voice, and we can customize the sound of the voice.

The following sample code snippet demonstrates how users can interact with Phone Gateway using both voice and keypad at the same time:

```

{
    action: 'input',
    type: [ 'speech', 'dtmf' ],
    eventUrl: [ 'https://gate.edwardsle.com/confirmmainmenu' ],
    speech: {
        language: "en-US",
        context: ["order", "check", "cancel"],
        endOnSilence: 1,
        saveAudio: true
    },
    dtmf: {
        maxDigits: 1,
        submitOnHash: true,
        timeout: 1
    },
}

```

There are two types of user interaction for this input. Speech is voice interaction and dtmf is keypad response. User language will be English with a US accent. The context is words that our system expects from the user. In DTMF, the number of

keys we expect from the user is only a single key. If the user has no voice response within one second (*endOnSilence: 1*) or does not press any key for one second. (*timeOut: 1*) then this action will be terminated for the next action to be performed.

Phone Gateway only runs in the background so it does not have any interface. However, for the convenience of debugging and tracking the flow of calls, there are many outputs on the console.

```
=====MAIN MENU was called=====
{ digits: null, timed_out: false }
{ recording_url:
  'https://api-us.nexmo.com/v1/files/42b92ecd-15bd-4b49-9295-6c385ffd598d',
  timeout_reason: 'end_on_silence_timeout',
  results:
    [ { confidence: '0.7254131', text: 'order pizza' },
      { confidence: '0.5563975', text: 'order a pizza' },
      { confidence: '0.51581204', text: 'part of Pisa' },
      { confidence: '0.51581204', text: 'Port of Pisa' },
      { confidence: '0.51581204', text: 'Court of Pisa' },
      { confidence: '0.51581204', text: 'quart of Pisa' } ] }
validOption true
```

Figure 4.1.4.2.2: Phone Gateway Console

This screenshot shows the output of a user's interaction to Phone Gateway. In the image, the function Main Menu was called. There are four options for customers in this step: order pizza, check order, cancel order, and talk to a representative. The user replied "order pizza" to the system to get started ordering a pizza. We can see that the user responded by voice, and there is a url for the record. We also see that the keypad was unused and no digits was pressed.

```
=====CONFIRM PIZZA TOPPINGS was called=====
{ digits: '', timed_out: true }
{ recording_url:
  'https://api-us.nexmo.com/v1/files/1c024a98-6f74-4324-84a0-6e8bf57b6f01',
  timeout_reason: 'end_on_silence_timeout',
  results:
    [ { confidence: '0.7225994', text: 'pepperoni mushroom' },
      { confidence: '0.56447315', text: 'pepperoni mushrooms' },
      { confidence: '0.54825276', text: 'pepperoni must room' },
      { confidence: '0.54825276', text: 'pepperoni most room' } ] }
[ 'pepperoni', 'mushroom' ]
```

Figure 4.1.4.2.3: Sample of Voice Recognition

This second screenshot shows how the Speech-to-Text works. The context in this image is the user telling the system that he wants pepperoni and mushroom for his pizza, and his speech was converted to text with the accuracy rate as confidence. However, we understand that high confidence is not necessarily the text we need. Mostly, we need a dictionary to match the user's voice to current options.

This below table is an example of dictionary for speech recognition:

Option	Keyword
bbq	barbeque, barbie cute, bbq
cancel order	cancel, cancelation, canceling
check order	check, status, what, how, tell
large	large, lar, lord
medium	medium
no	no, nope, nah
order pizza	order, place, buy, purchase
regular	regular
representative	representative, real person, someone
small	small
yes	yes, yeah, yup, yep

Figure 4.1.4.2.4: Dictionary for Voice Recognition

The sensitivity of the speech recognition is dependent on the size of the dictionary. Each option requires multiple keywords for identification. In the example above, I even used some strange words, but the pronunciation of them is similar to the sound of the original word. We can build models and apply machine learning to teach speech recognition more accurately.

1.4.3. Website

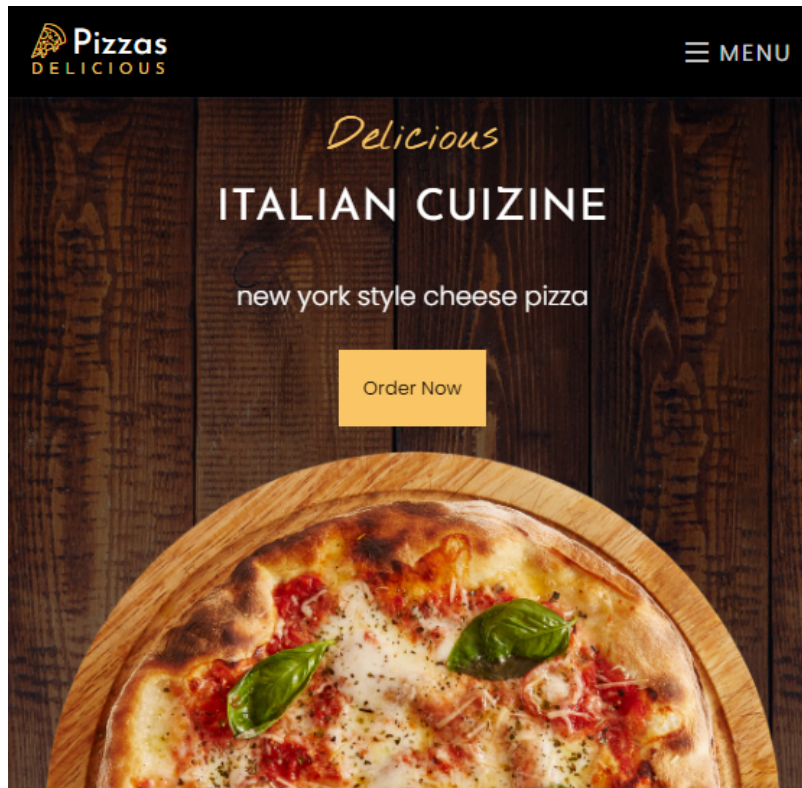


Figure 4.1.4.3.1: Interface of Website (Homepage)

There are many pictures and information about our pizzeria on the homepage of our website. Users can also find phone numbers to order pizza. However, this homepage is a static page. It has a link and button that takes the user to the order page.

The Order page is modeled after the steps of the Place Order module from Phone Gateway. And of course, users need a phone number to start ordering. Ordering steps include: choosing size, choosing crust, choosing sauce, and choosing toppings. Users also can buy many pizzas in one order by hitting the button *Add More Pizza*.

Order Online

Phone

Enter Phone Here To Process

We'll never share your phone with anyone else.

Pizza Builder

1. Size

☐ Large ☐ Medium ☐ Small

2. Crust

Regular

3. Sauce

Regular Sauce

4. Toppings

☐ pepperoni ☐ jalapeno

☐ beef ☐ mushroom

☐ ham ☐ black olives

☐ chicken ☐ pineapple

Add More Pizza Place Order

Figure 4.1.4.3.2: Interface of Website (Order page)

Just like the function Place Orders of Phone Gateway. Users can only access this form when the current time is valid, within the operating hours of our store:

```
function isInBusinessHour()
{
    $today_name = Date('l');
    $current_time = strtotime(Date('H:i:s'));

    $query = "
        SELECT *
        FROM hours
        WHERE
            day=$today_name
            AND timein <= $current_time
            AND timeout >= $current_time
        LIMIT 1";
    $result = $conn->query($query);
    if ($result->num_rows > 0)
    {
        return true;
    }
    return false;
}
```

It is easy to see that implementing this boolean function in PHP is simpler than in NodeJS. Everything needed is available from the PHP libraries and developers can simply call and use it as fast as you can see in the code.

This form is made by pure javascript, and it is optimized for using PHP in the backend. When an order is submitted, all pizzas are stored in an array, one by one. That means there will be no data disturbance.

```
$add_order = "
    INSERT INTO orders (id, status, source, phone, ispaid)
    VALUES ('$order_id','active','$source','$phone','$ispaid')";

if(mysqli_query($conn,$add_order)){
    foreach($data as $pizza){
        $size = $pizza['size'];
        $crust = $pizza['crust'];
        $sauce = $pizza['sauce'];
        $toppings = '';
        foreach($pizza['toppings'] as $topping){
            $toppings .= $topping;
            $toppings .= ", ";
        }
        $add_pizza = "
            INSERT INTO pizzas
            (orderid,status,crust,size,sauce,toppings)
            VALUES
            ('$order_id','$status','$crust',
            '$size','$sauce','$toppings')";

        if(!mysqli_query($conn,$add_pizza)){
            echo "Pizza did not get added to db";
        }
    }
} else {
    echo "Create order failed";
}
```

This is a simple form and there isn't much to talk about. Furthermore, PHP also simplifies the backend implementation. The hard part is implementing this form in pure Javascript.

```
const createBuilder = () => {
  ...
  let card = document.createElement('div');
  let cardHeader = document.createElement('div');
  let divCardTitle = document.createElement('span');
  cardTitle.innerHTML = "Pizza Builder";
  let divRemoveButton = document.createElement('span');
  let removeBtn = document.createElement('a');
  removeBtn.onclick = function() {
    pizzaBuilder.removeChild(column);
  }
  divCardTitle.appendChild(cardTitle);
  divRemoveButton.appendChild(removeBtn);
  cardHeader.appendChild(divCardTitle);
  cardHeader.appendChild(divRemoveButton);
  ...
}

createBuilder();
```

Implementing this form using Javascript will allow users to change the form in real time and give the user the impression that the system works fast and smoothly.

1.4.4. Kitchen Screen

5	L - R	Regular sauce Pepperoni, Chicken, Jalapeno
5	M - T	BBQ sauce Pepperoni, Beef, Mushroom
6	S - R	Alfredo sauce Pepperoni

Figure 4.1.4.4.1: Interface of Kitchen Screen

Kitchen Screen only shows unprepared orders. Looking at the sample data in the screenshot, we can easily see that the first two pizzas belong to order id 5, and the last pizza belongs to order id 6. The first pizza is a large sized (L), regular crust (R), regular sauce with pepperoni, chicken, and jalapeno pepper. The second pizza is a medium sized (M),

thin crust (*T*), barbeque sauce with pepperoni, beef, and mushroom. The last pizza is a small sized (*S*), regular crust (*R*), alfredo sauce with pepperoni. Colors will help chefs identify faster than read.

```
const reloadData = () => {
  // empty the section before adding new items
  let displaySection =
    document.getElementById('pizzas-display');
  displaySection.innerHTML = "";

  // get unprepared pizzas
  const pizzas = getUnpreparedPizzas();

  // add each pizza to display section
  for (let pizza of pizzas) {
    appendPizzaToDisplay(pizza);
  }
}

// reload data every 2 minutes
let intervalReload = setInterval(reloadData, 120000);
```

This function reloads data every two minutes. Every two minutes, the section will be cleared and new orders updated.

Chefs will interact with this form using the *arrow keys* and select an order with *Enter* key. When a pizza is selected, it is removed from the display and set its status as *cooked*. When all pizzas for an order are set as *cooked*, the order is ready for the customer to pick up. Then, that order is set status as *ready*.

```
// set done to an element
const setFinish = () => {
  //console.log("Entered", currentId, currentOrder);
  if (isRemovalReady) {
    console.log("Ready to remove");

    // == call ajax ==//
    setPizzaAsCooked(currentId);

    // == remove item from list ==//
    list.removeChild(currentOrder);
    isRemovalReady = false;
  } else {
    console.log("Not ready to remove");
  }
}
```

1.4.5. Admin Dashboard

Admin Dashboard has two interfaces. A page for all of the orders, and another page for set business hours of our pizzeria.

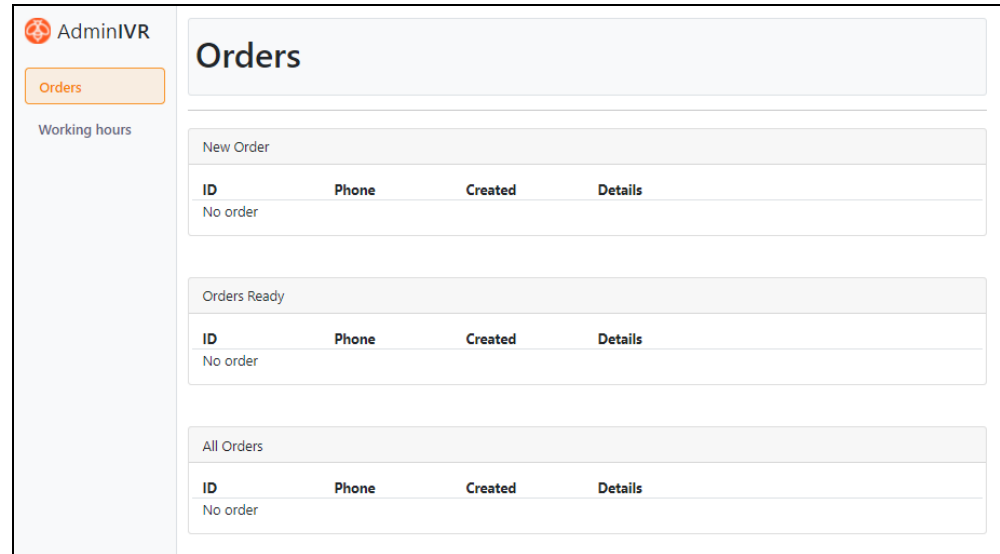


Figure 4.1.4.5.1: Interface of Dashboard (Order details)

The order details page has three sections. The first section, new orders, will show all unprepared orders. The second section, orders ready, will show all orders that have been cooked. The last section, all orders, will show all the orders in the database.

```
// unprepared order
$query_new_order = "SELECT *
  FROM `orders`
  JOIN `pizzas` ON orders.id = pizzas.orderid
  WHERE orders.status = 'unprepared'";
$new_order = mysqli_query($conn,$query_new_order);

// orders ready
$query_ready_order = "SELECT *
  FROM `orders`
  JOIN `pizzas` ON orders.id = pizzas.orderid
  WHERE orders.status = 'ready'";
$ready_order = mysqli_query($conn,$query_ready_order);

// all orders in the database
$query_all_order = "SELECT *
  FROM `orders`
  JOIN `pizzas` ON orders.id = pizzas.orderid";
$orders = mysqli_query($conn,$query_all_order);
```

This code simply queries data from the database. There are three queries for three display sections in the interface. We can also query only once and then filter the data, but the performance of MySQL is always a lot faster than the performance of PHP.

AdminIVR

Orders

Working hours

Day	Open	Close
Monday	08:00 AM	09:00 PM
Tuesday	08:00 AM	09:00 PM
Wednesday	08:00 AM	09:00 PM
Thursday	08:00 AM	09:00 PM
Friday	08:00 AM	09:00 PM
Saturday	08:00 AM	09:00 PM
Sunday	08:00 AM	09:00 PM

Figure 4.1.4.5.2: Interface of Dashboard (Working hours)

This page shows all of the operating hours of the store. It also allows users to make changes directly on this interface. There are seven rows corresponding to the seven days of the week, along with opening and closing hours.

```
// Update Monday
$monday = "UPDATE `hours`
SET hours.timein='$mon_open', hours.timeout='$mon_close'
WHERE hours.id=1";
$monday_input = mysqli_query($conn,$monday);

// Update Tuesday
$tuesday = "UPDATE `hours`
SET hours.timein='$tue_open', hours.timeout='$tue_close'
WHERE hours.id=2";
$tuesday_input = mysqli_query($conn,$tuesday);
// ...
```

There are seven similar pieces of code to update store hours. The screenshot shows two of them. These data are required for the website and Phone Gateway to restrict the uptime for talking orders.

2. Discussion

Vonage Nexmo APIs send the data to our Phone Gateway API for every single call. There are URLs involved:

- HTTP Get: <https://gate.edwardsle.com/answer>
- HTTP Post: <https://gate.edwardsle.com/event>
- HTTP Get: <https://gate.edwardsle.com/fallback>

Phone Gateway API retrieves the Nexmo Call Control Objects (NCCO) about a call at the event URL and sends back information about the call at the answer URL to Vonage to control the call. Vonage acts as a telephone service provider and is fully controlled by our system, Phone Gateway API. The fallback URL is only used when Vonage services cannot connect to the answer URL.

Besides, Google Cloud provides us with perfect powerful services. Google Speech-To-Text is a speech recognition service with 125 languages supported. Based on the keywords we provide, Google Speech-To-Text can return the customer's needs with high

IVR (INTERACTIVE VOICE RESPONSE) FOR PIZZERIAS

accuracy. In addition, WaveNet is an artificial intelligence application to Google Text-To-Speech that makes this service performing more natural. WaveNet may cost you more, but it is useful in other services, such as storytelling, reading.

Conclusion

1. Future Work

This project can improve a lot if machine learning is applied to our system. First, the voice recognition service will work more efficiently. We are using a manual dictionary to match the voices of users. However, we can build models so that the machine learns from the users while using our services and automatically grows the dictionary. Second, machine learning can help the system recognize the user's usage habits, which can automatically give deals and combos that can reduce the duration of a conversation and increase sales. Last, machine learning can also make predictions about food consumption, thereby the owner can estimate the required inventory.

Since this is a sales system, reports and analysis are essential. Clear data will help owners to make the right decisions that can increase sales and increase revenue. Moreover, reports and analysis also help to manage the system more efficiently. The admin dashboard lacks functionality and adding more reports and analytics is what I have to do in the coming days.

2. Limitations

There are five different applications in this project making the handling of communication between them complicated. However, because of the specificity of each system and the limitations of technology, having multiple systems was a must for this project. This is a pioneering project in a new field, so the lack of documentation and guidance is also a difficulty that was encountered during the implementation phase. Applying new technologies to this project is also another difficulty. Testing and experimenting with new technologies will take a long time before they can be integrated into the project.

References

- Beach, Scott R et al. "Using Interactive Voice Response to Measure Elder Mistreatment in Older Adults: Feasibility and Effects." *Journal of official statistics* vol. 26,3 (2010): 507-533.
- DiPietro, Robin B, Tena B. Crews, Cathy Gustafson & Sandy Strick (2012) "The Use of Social Networking Sites in the Restaurant Industry: Best Practices", *Journal of Foodservice Business Research*, 15:3, 265-284, DOI: 10.1080/15378020.2012.706193
- Iancu, Bogdan. "Evaluating Google Speech-to-Text API's Performance for Romanian e-Learning Resources." *Informatica Economica* 23.1 (2019).
- Jain, Shilpi, et al. "Interactive Voice Assistants – Does Brand Credibility Assuage Privacy Risks?" *Journal of Business Research*, vol. 139, 2022, pp. 701–717., <https://doi.org/10.1016/j.jbusres.2021.10.007>.
- Kim, J.(S)., Christodoulidou, N. and Choo, Y.(C). (2013), "Factors influencing customer acceptance of kiosks at quick service restaurants", *Journal of Hospitality and Tourism Technology*, Vol. 4 No. 1, pp. 40-63. <https://doi.org/10.1108/17579881311302347>
- Kirkpatrick, Keith. "AI in Contact Centers." *Communications of the ACM*, vol. 60, no. 8, 2017, pp. 18–19., <https://doi.org/10.1145/3105442>.
- Khullar, Aman, et al. "Costs and Benefits of Conducting Voice-Based Surveys versus Keypress-Based Surveys on Interactive Voice Response Systems." *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS)*, 2021, <https://doi.org/10.1145/3460112.3471963>.
- Mundt, James C., et al. "Psychological Performance Assessment via Interactive Voice Response Systems." *Behavior Research Methods, Instruments, & Computers*, vol. 29, no. 4, 1997, pp. 506–518., <https://doi.org/10.3758/bf03210602>.
- Pratama, I. Putu Arie, and Nyoman Putra Sastra. "Token-based Single Sign-on with JWT as Information System Dashboard for Government." *Telkomnika* 16.4 (2018): 1745-1751.
- Trivedi, Ayushi, et al. "Speech to text and text to speech recognition systems-Areview." *IOSR J. Comput. Eng* 20.2 (2018): 36-43.
- Sehgal, Rohit Raj, and Gaurav Raj., "Interactive Voice Response using Sentiment Analysis in Automatic Speech Recognition Systems," *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2018, pp. 213-218, doi: 10.1109/ICACCE.2018.8441741.
- Shovon, Ahmedur Rahman, et al. "A restful e-governance application framework for people identity verification in cloud." *International Conference on Cloud Computing*. Springer, Cham, 2018.
- Yacoub, Sherif M., et al. "Recognition of emotions in interactive voice response systems." *Interspeech*. 2003.