

# NANYANG TECHNOLOGICAL UNIVERSITY

---

## SINGAPORE

AN6007

ADVANCED PROGRAMMING

Group Project

## Non-technical Report : User Guide for Bank Queueing System



**Group B -Team 6**  
LYU, JIAXIN (MISAKI)  
YANG, YUXIN  
ZHAO, YUECHEN  
PENG, LI  
GAN, JIUJUN

Nanyang Business School  
12/02/2023



# Readme

## Build the program runtime environment

This project is based on **node.js** development of back-end applications, which requires the configuration of the computer environment before running the program.

### Step1:

Go to the <https://nodejs.org> installation site to download and install the corresponding version of node.js.

### Step2:

Open terminal

### Step3:

Unzip the program.zip file and enter the command in terminal to access the path where the unzipped file is located.

### Step4:

Enter the command: 'npm install express'

**So far the program runtime environment has been set up, the next just need to run and test the program**

The program can be run by typing the command: '**npm run dev**' in the current path

### Test each web page

<http://localhost:3000/branches/display/1>

<http://localhost:3000/CRO/branches/1>

<http://localhost:3000/branches/1/getQueueNum>

<http://localhost:3000/branches/1/counterCallPage/1>

<http://localhost:3000/reschedule/branches/1>

### \*If you have problems with bodyparser as:

*body-parser deprecated bodyParser: use individual json/urlencoded middlewares app.js:7:9*

please go to **app.js Line 7** and find: *app.use(bodyParser())*

delete this line and replace by *app.use(bodyParser.json())*

and *app.use(bodyParser.urlencoded({extended: true}))*

**Content**

- 1. Introduction..... 2**
- 2. Queue System Overview ..... 2**
- 3. Business Flow Chart..... 2**
- 4. Queue System Operation Set Up..... 3**
- 5. Single Page Application Main Pages Overview ..... 4**
- 6. Main Page User Guide ..... 6**
  - 6.1 Domain Clients Support User Guide ..... 6
  - 6.2 Counter Staff Support User Guide..... 8
  - 6.3 Reschedule Use Case Diagram and Illustration ..... 10
  - 6.4 Customer Relationship Officer (CRO) User Guide ..... 11
  - 6.5 Queue Management System Main Display ..... 12
- 7. Limitation and Improvement..... 14**
- 8. Conclusion..... 14**

### 1. Introduction

NoMoney Bank queue management systems was designed to manage customer traffic and improve the customer experience. NoMoney Banks always have many clients visiting branches for various services. Without a utilized queueing system, customers may have to wait in long lines, leading to frustration and a poor experience. A well-organized queue system allows customers to take a number and wait comfortably until their turn arrives which helps to manage customer traffic, reduce wait times, and improve the overall customer experience.

### 2. Queue System Overview

The queue management system combines backend server and single page web application to support domain client, counter staff, and customer relationship officer to achieve their operational goal. The system is easy to use and intuitive, with a user-friendly interface that makes it simple to record a customer queue request based on the category, call for the customer to the respective counter, ‘re-schedule’ ‘missed queue number’ to be served after next 2, view the entire queue list for the respective categories and stop/re-initiate the taking of queue number for either queue category. Overall, this queue management system provides a powerful and flexible solution for the bank service, enabling users to be served without dissatisfaction in an efficient and streamlined manner.

### 3. Business Flow Chart

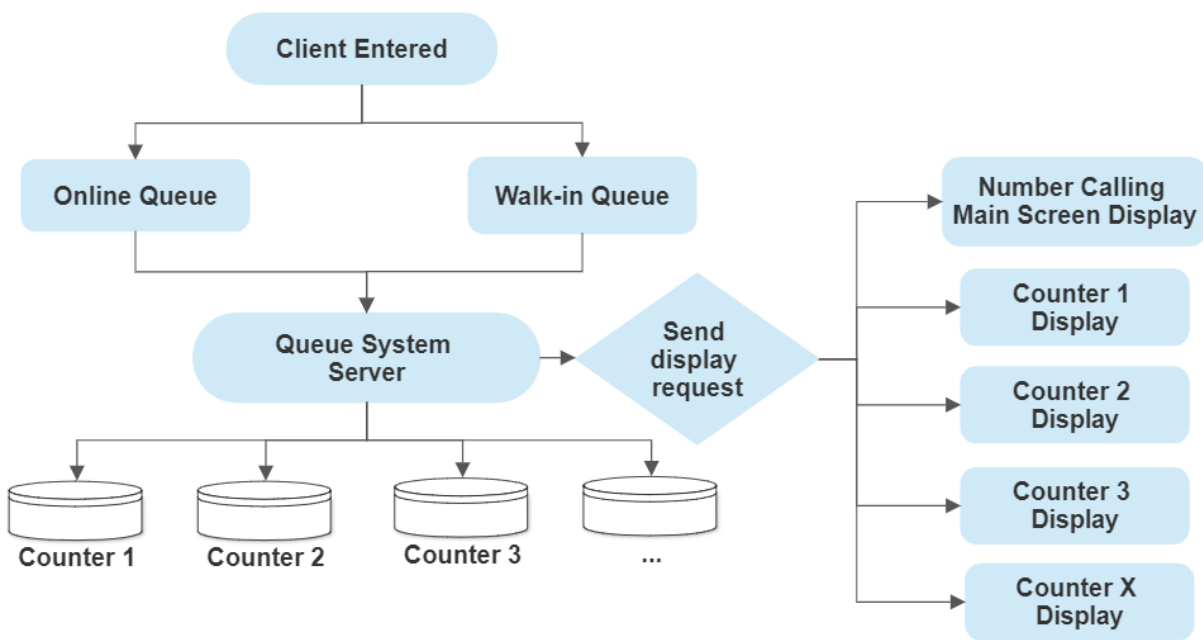


Figure 1 Queue Management System Business Flow Chart

### 4. Queue System Operation Set Up

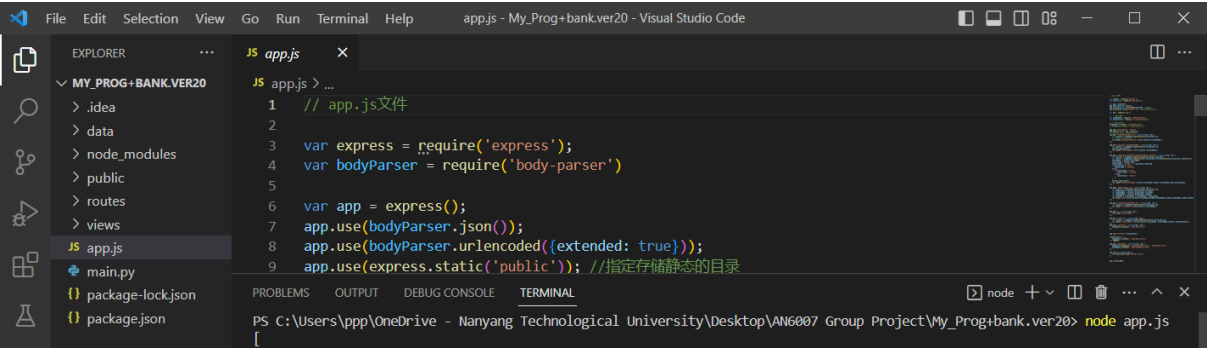


Figure 2 Visual Studio Package Installation and Set Up

Use Visual Studio Code (VS code) as a tool to build up the entire system and issue the command **node app.js** in terminal. Once running successfully, open the browser and enter following command to display the 5 application pages established for the queue system:

	Application Programming Interface (API) Address	Details of the application
1	<a href="http://localhost:3000/branches/:branchid/getQueueNum">http://localhost:3000/branches/:branchid/getQueueNum</a> <a href="http://localhost:3000/branches/1/getQueueNum">http://localhost:3000/branches/1/getQueueNum</a>	Record a customer queue request based on the category <i>:branchid is the id of the branch</i>
2	<a href="http://localhost:3000/branches/:branchid/counterCallPage/:counterid">http://localhost:3000/branches/:branchid/counterCallPage/:counterid</a> <a href="http://localhost:3000/branches/1/counterCallPage/1">http://localhost:3000/branches/1/counterCallPage/1</a>	Call for the customer to the respective counter <i>:branchid is the id of the branch</i> <i>:counterid is the id of counter</i>
3	<a href="http://localhost:3000/reschedule/branches/:branchid">http://localhost:3000/reschedule/branches/:branchid</a> <a href="http://localhost:3000/reschedule/branches/1">http://localhost:3000/reschedule/branches/1</a>	Re-schedule missed queue number to be served after next 2 <i>:branchid is the id of the branch</i>
4	<a href="http://localhost:3000/CRO/branches/:branchid">http://localhost:3000/CRO/branches/:branchid</a> <a href="http://localhost:3000/CRO/branches/1">http://localhost:3000/CRO/branches/1</a>	View the entire queue list for the respective categories and stop/re-initiate the taking of queue number <i>:branchid is the id of the branch</i>
5	<a href="http://localhost:3000/branches/display/:branchid">http://localhost:3000/branches/display/:branchid</a> <a href="http://localhost:3000/branches/display/1">http://localhost:3000/branches/display/1</a>	Main display page to show the queue serving status <i>:branchid is the id of the branch</i>

Table 1. API Address for Main Pages

## 5. Single Page Application Main Pages Overview

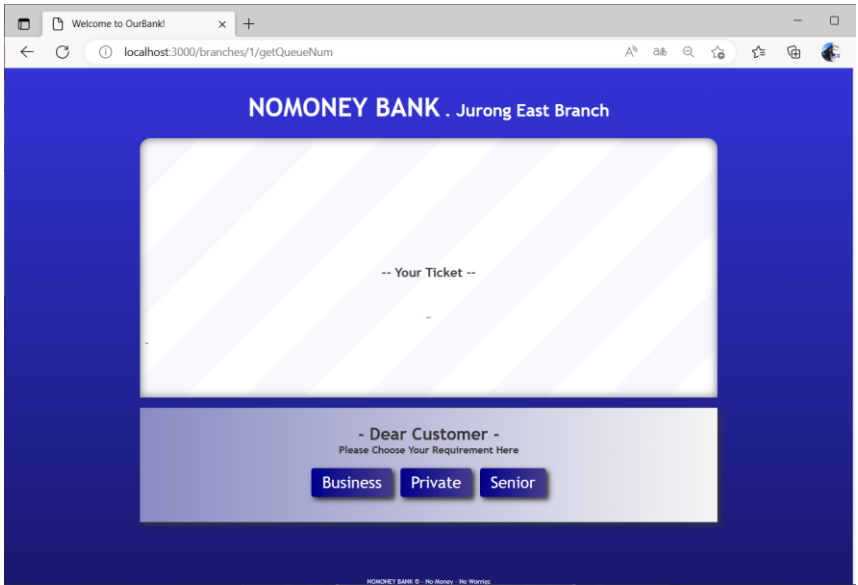


Figure 3 Domain Client Support Application

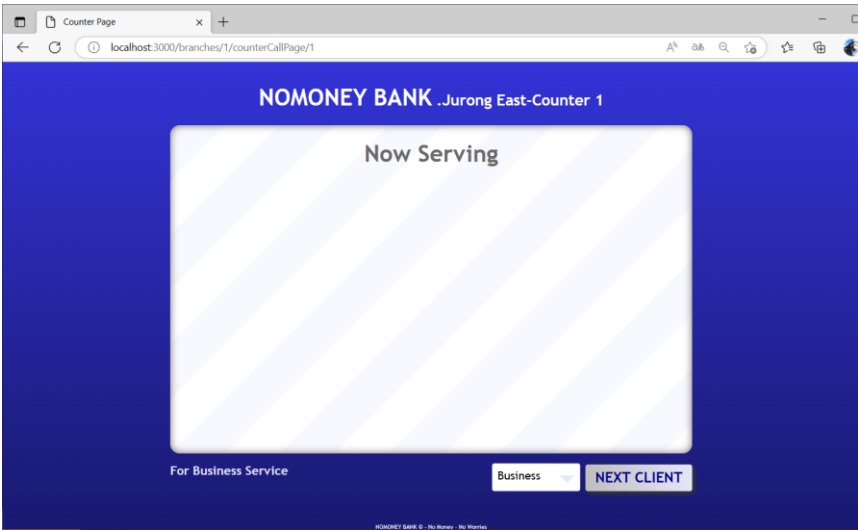


Figure 4 Counter Staff Support Application

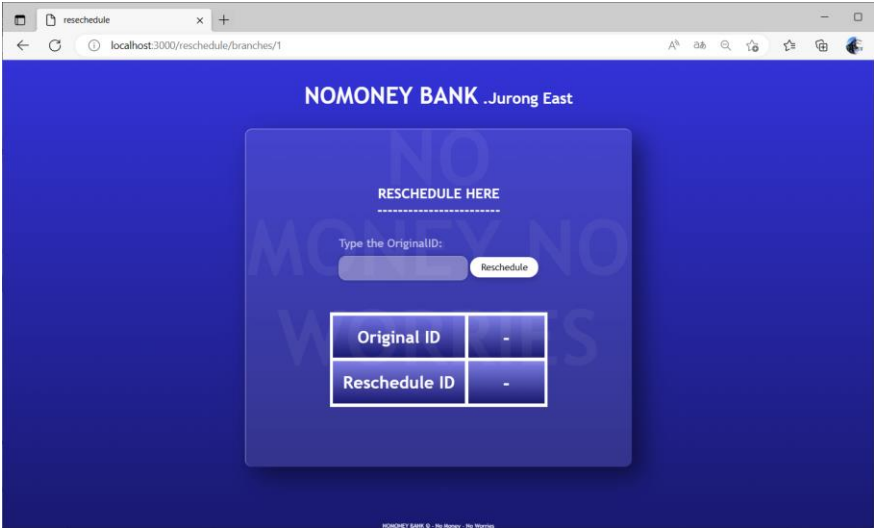


Figure 5 Reschedule Support Application

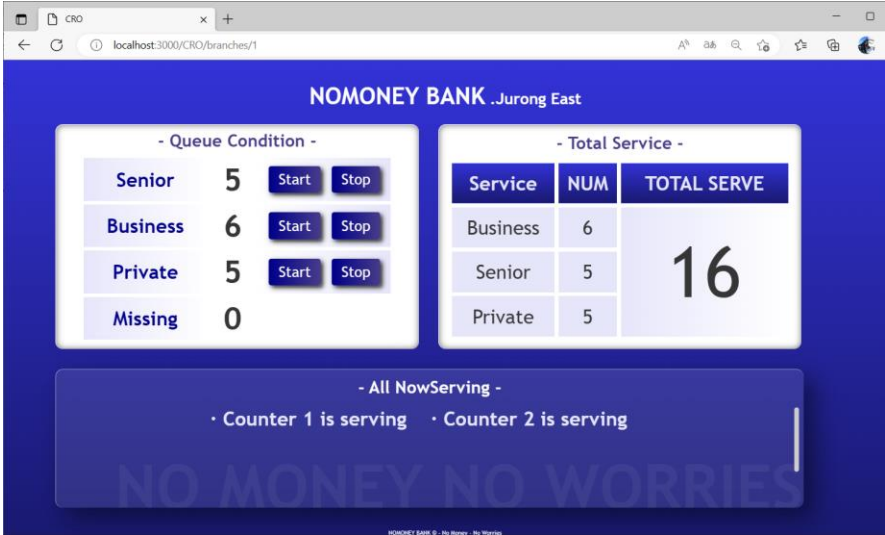


Figure 6 Customer Relationship Officer Control Application

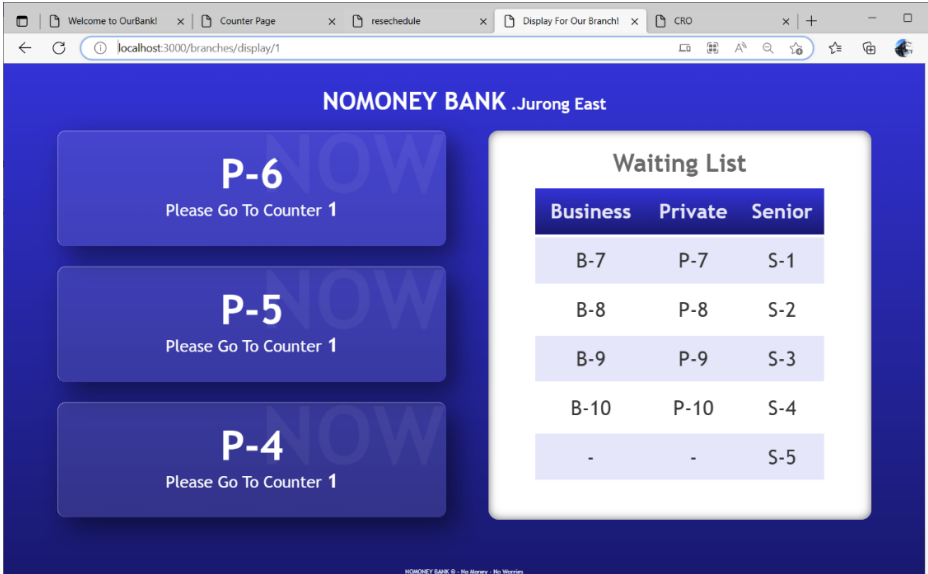


Figure 7 Serving Status Main Display

6. Main Page User Guide

6.1 Domain Clients Support User Guide

Use Case Diagram and Illustration

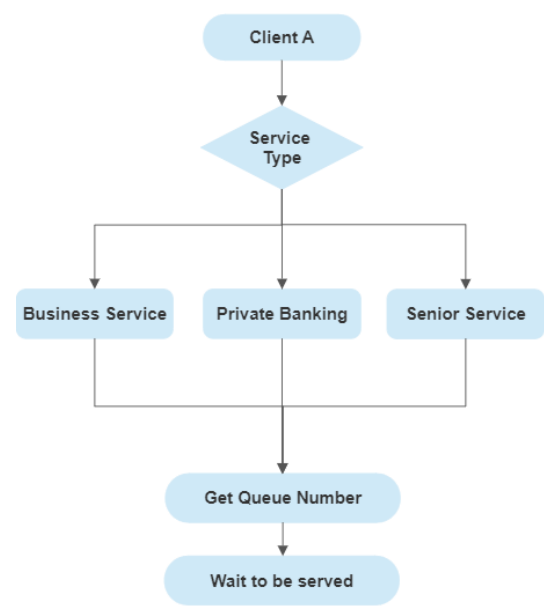


Figure 8 Domain Client Support Use Case Diagram

A client can simply access the queue management system and get the queue number by clicking the service type button – Business, Private Banking or Senior Priority Service. The webpage will display the client’s queue number once successful as shown below.

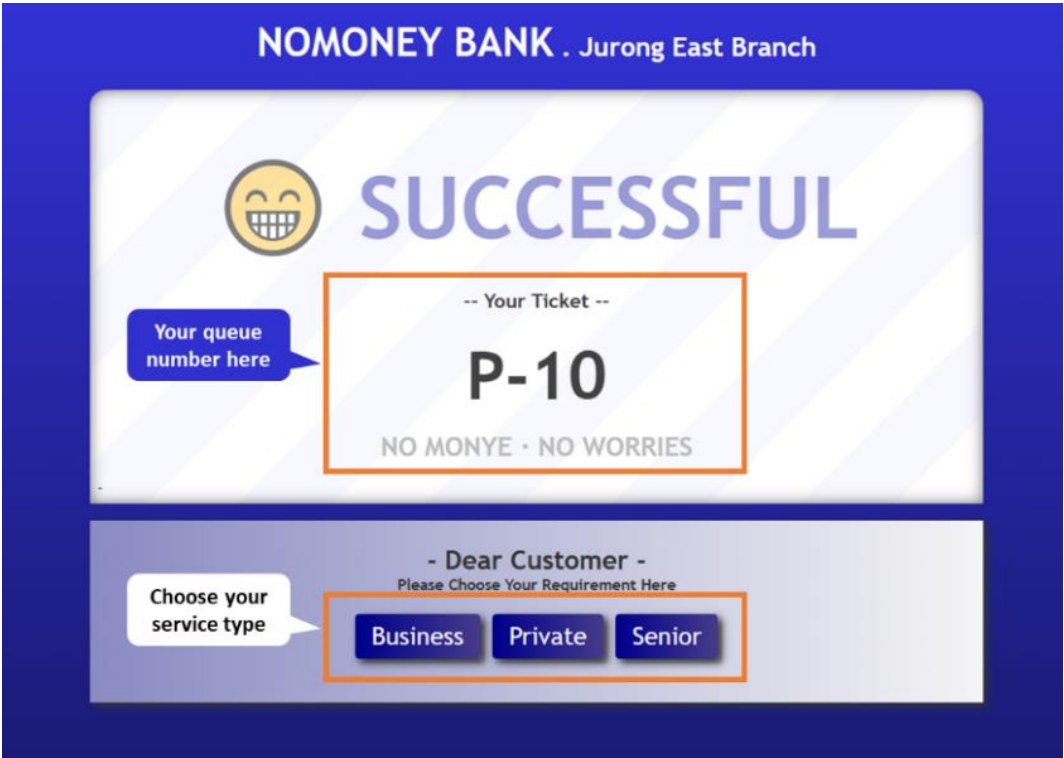
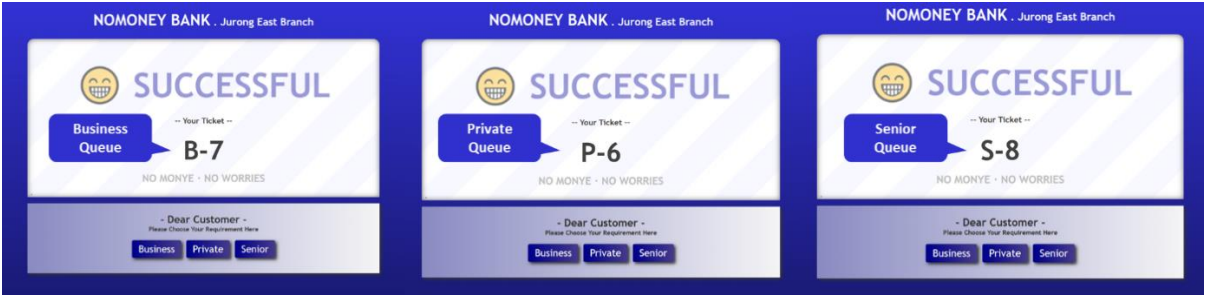


Figure 9 Domain Client Support Application



The client will get the queue number immediately under the chosen service type. “**B-X**” for business service, “**P-X**” for private service and “**S-X**” represented senior service.



*Figure 10 Queue number for different service type*

6.2 Counter Staff Support User Guide

Use Case Diagram and Illustration

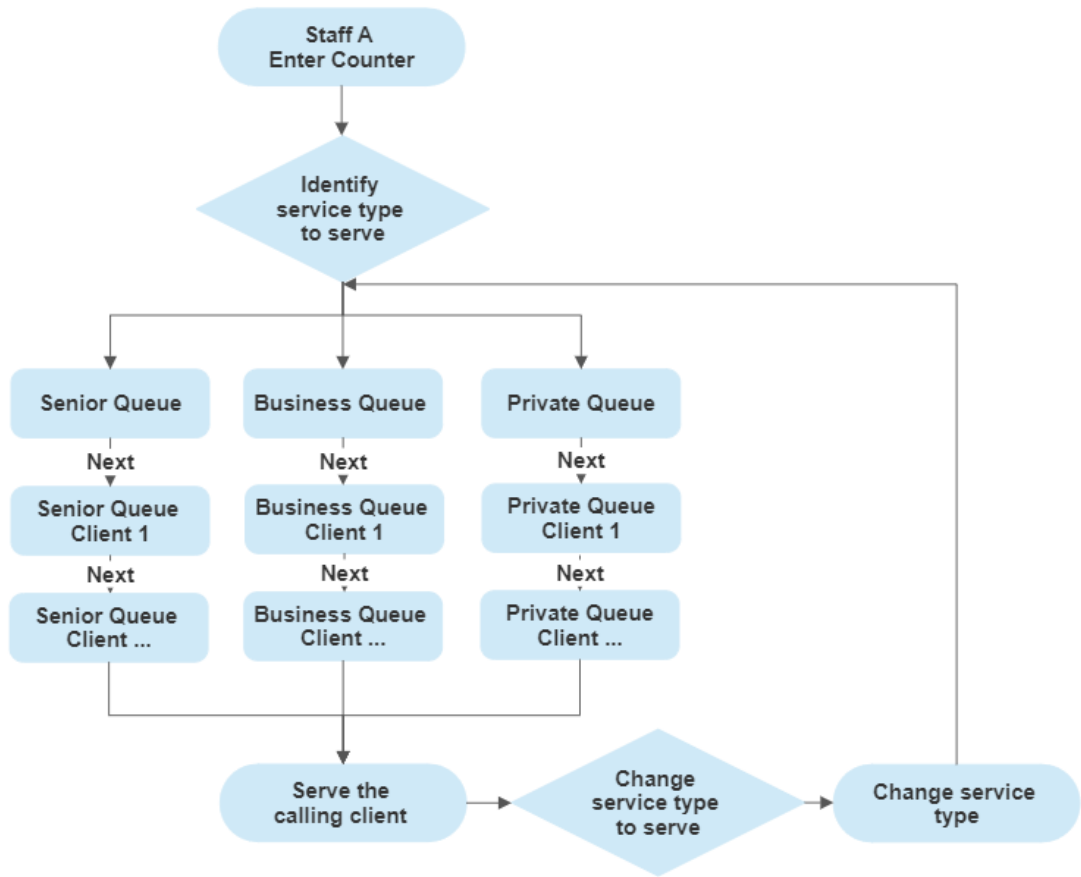


Figure 11 Counter Staff Support Use Case Diagram

Each counter will have their own user interface to communicate with the backend queue management system. Counter staff can call for the customer to the respective counter with one click on “**Next Client**” and the serving number will be shown on the page.

Besides, if there is no more client in the queue, the counter can choose to halt until the next client arrival or help up in other service queues. Once the original service queue is empty, it will automatically switch to other queues to support temporary. For example, counter X is for business service, however now business queue is empty. When the counter staff click “Next Client” it will shift to senior queue and help to take the number for senior queue as illustrated in figure 12. When there is new client comes in, the counter staff will switch back to serve the original type.

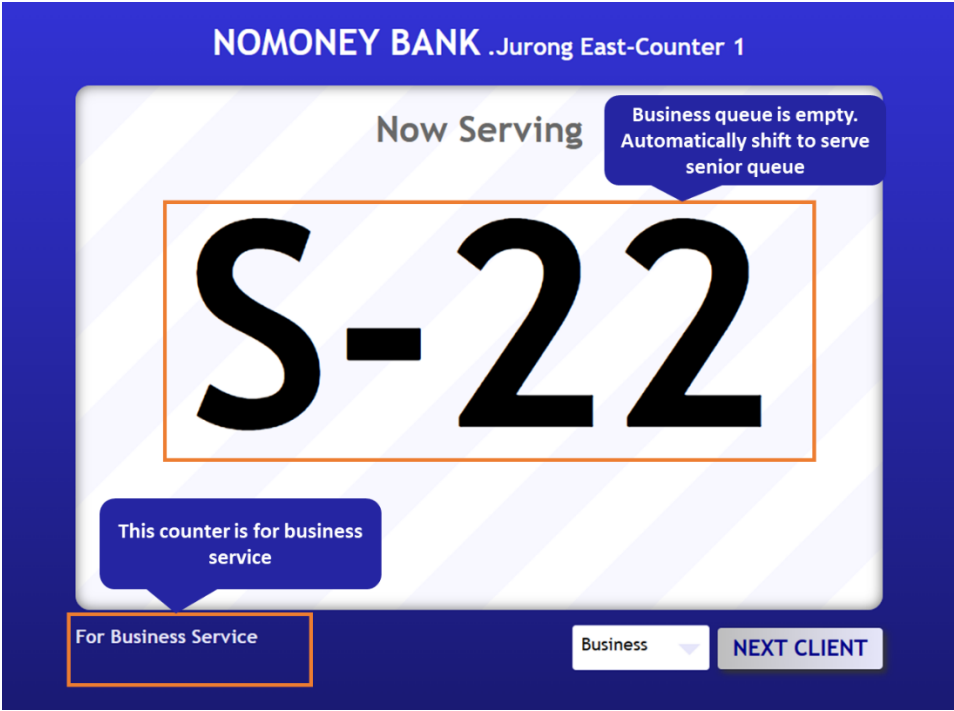


Figure 12 Service change even the original queue is empty

The counter might need to serve different types of the queue within the day or the week, for example, staff A is serving senior service on Saturday morning and shift to business after 1pm. In this case, staff A can simply change the service based on the needs from the drop-down list which shows in figure 13 &14.

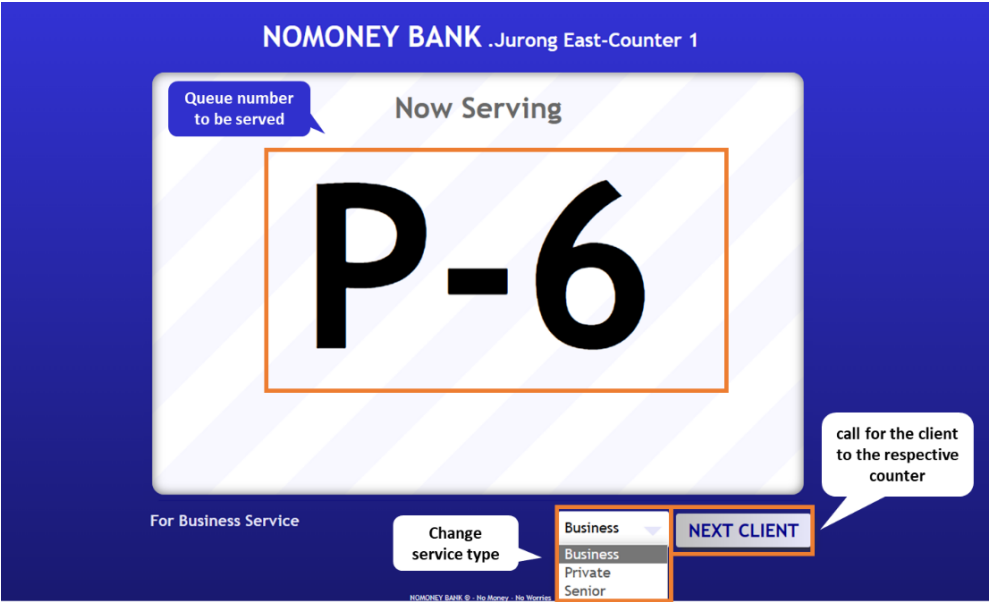


Figure 13 Counter Staff Support Application

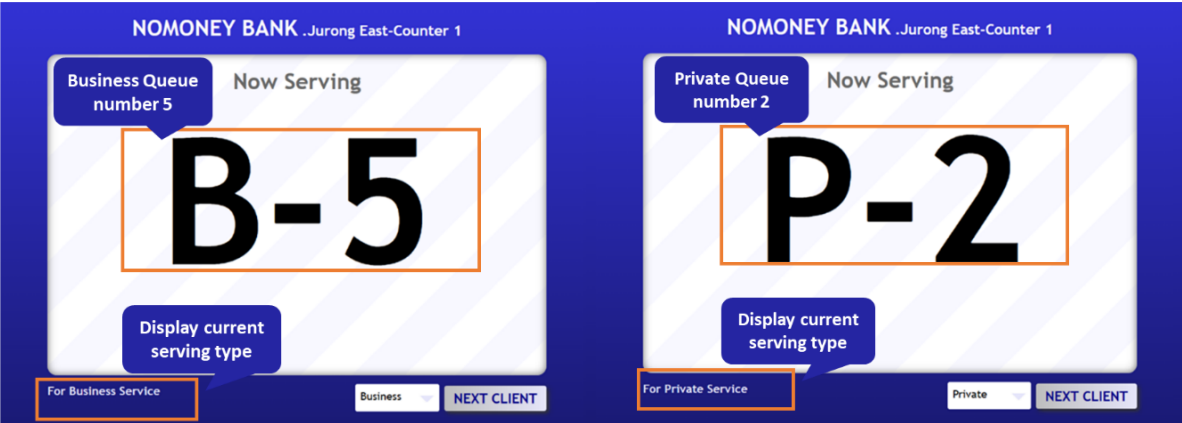


Figure 14 Change Service Type

6.3 Reschedule Use Case Diagram and Illustration

What if the client missed the queue? The queue management system has implemented reschedule function to help the client re-enter the queue. Counter staff will help to enter the original ID and click “Reschedule” to reschedule the missed queue number to the next 2 of the current serving.

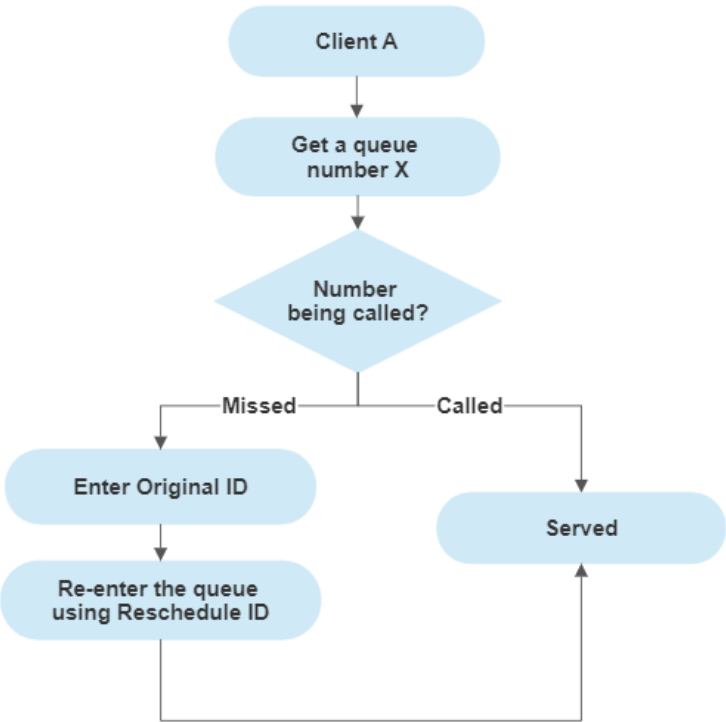


Figure 15 Reschedule Use Case Diagram

Figure 15 is the reschedule user interface display. When reschedule successfully, the original number will be displayed as “\*” + Original number as shown in figure 16 and scheduled before the counter staff calling reschedule ID.

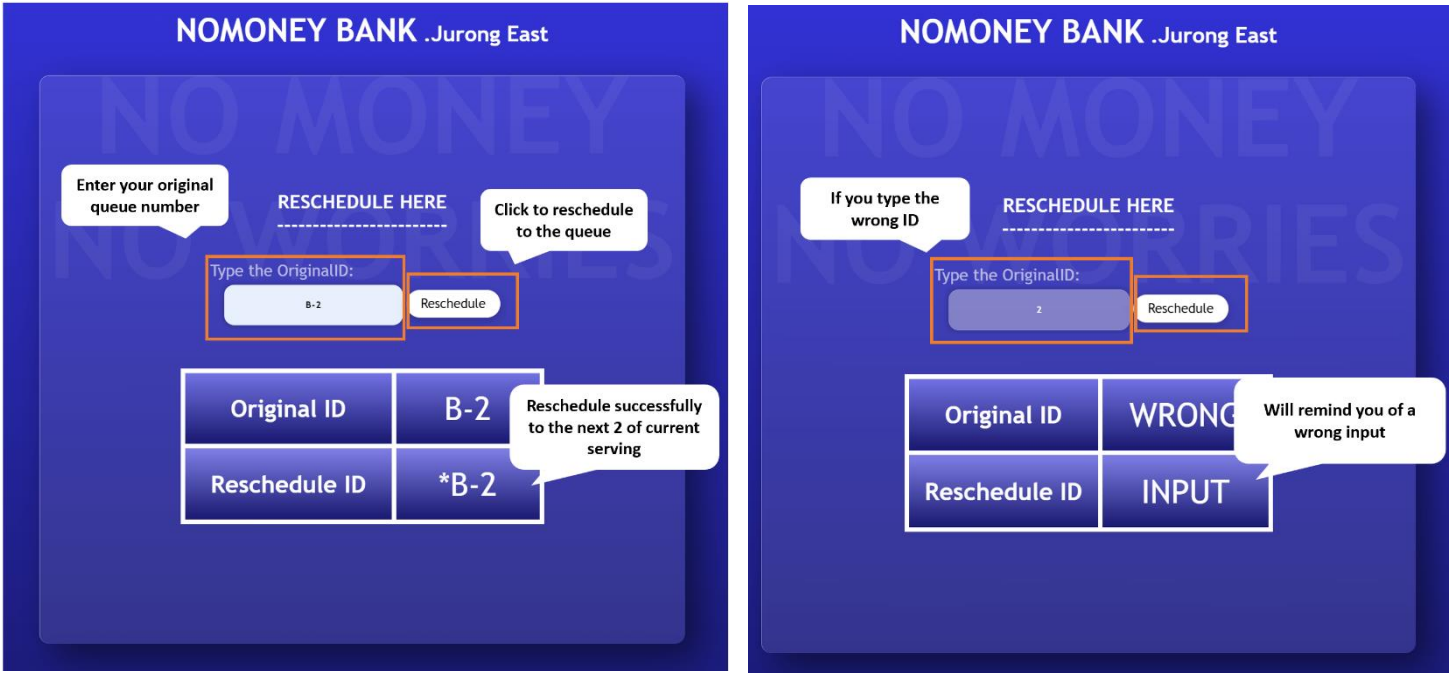


Figure 16 Reschedule Application

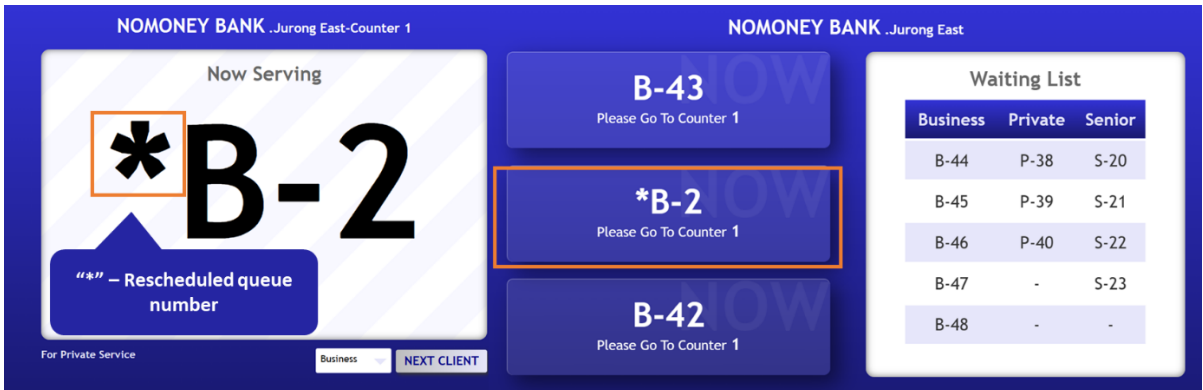


Figure 17 Reschedule Working Scheme

6.4 Customer Relationship Officer (CRO) User Guide

Customer Relation Officer will have separate user interface to communicate with the backend queue management system. There are two key functions under this section – Overall view the entire queue list for the respective categories and stop/re-initiate the taking of queue number for either queue category.

The officer can re-initiate the taking of queue number by clicking “Start” button and the webpage will pop up the message “Queue has been started”.

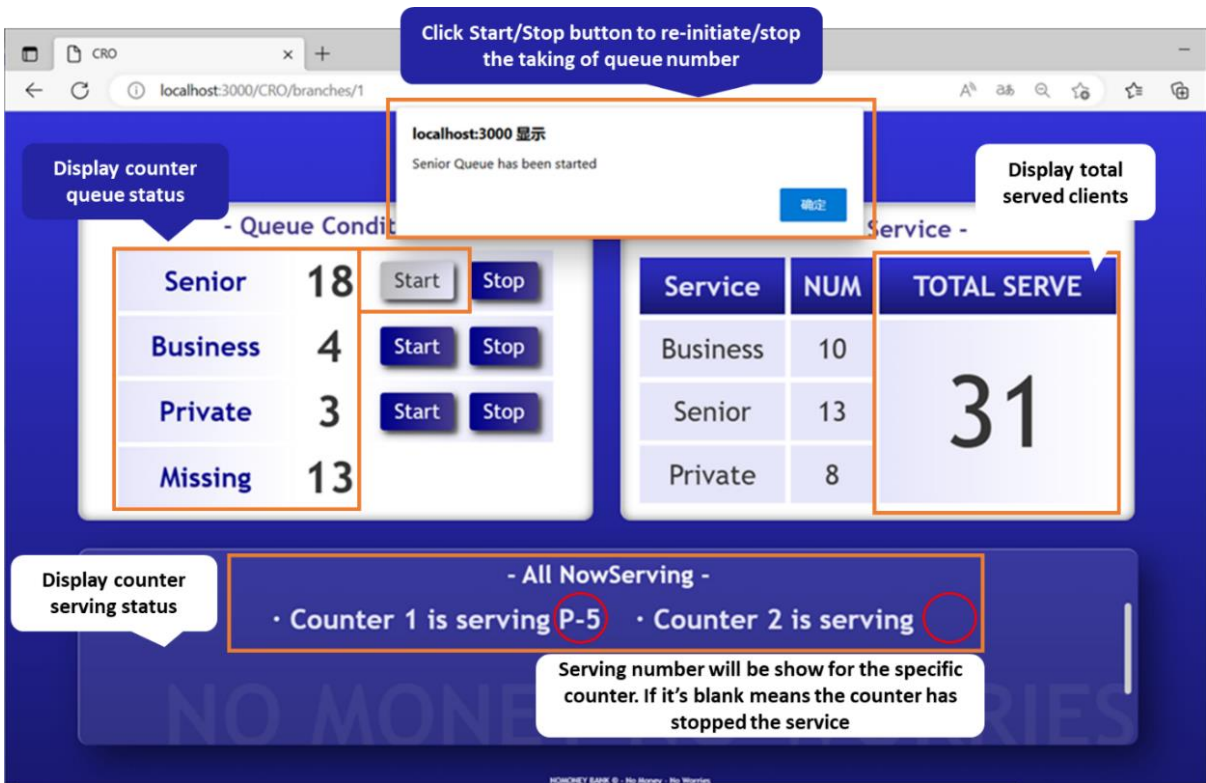


Figure 18 Customer Relationship Officer Application User Interface

When the counter is close or pause, the officer has to stop the number taking for that queue. It can be simply achieved by clicking the “**Stop**” button beside the queue that need to be stopped. Once the queue stopped, the “GetQueneNum” application stop the queue ticket taking which shown in Figure 19.

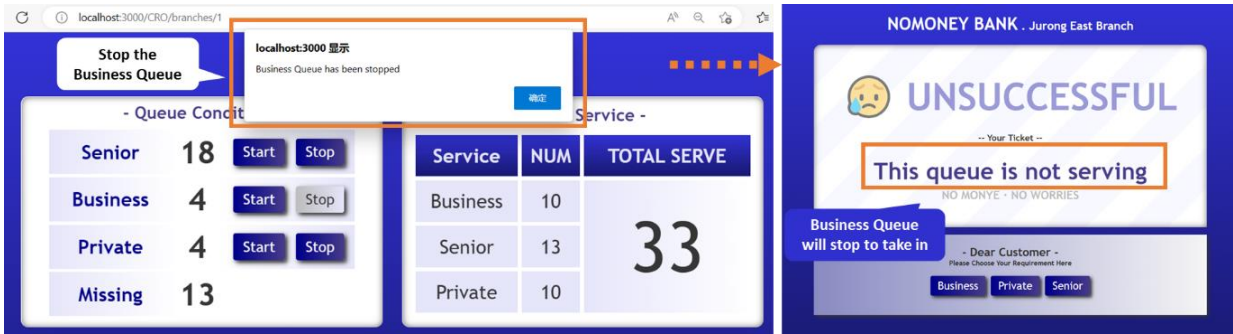


Figure 19 Stop to taking queue number for a particular service

6.5 Queue Management System Main Display

When the client’s number being called out by the counter it will be display on this main page. The client will then follow the instruction shown to go to the counter. Meanwhile, this also shows the queue waiting list for each service type so that the client can clearly see how many people in front of him/her.

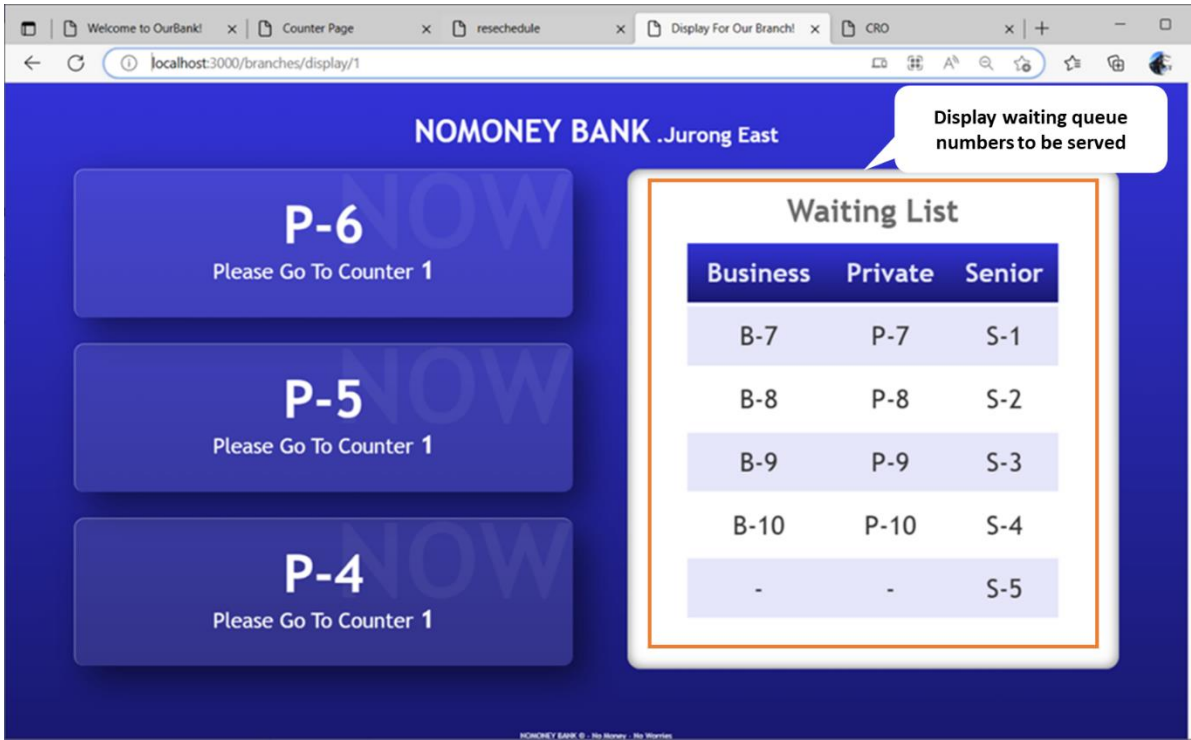


Figure 20 Queue Management System Main Interface

## 7.Limitation and Improvement

In conclusion, while our webpage has achieved significant progress, there are still several areas that can be improved to enhance the user experience and meet the needs of our clients more effectively.

Firstly, we can consider incorporating a SMS text notification system through the use of the Twilio package. This would allow clients to be alerted when they are next in the queue, providing them with greater transparency and convenience.

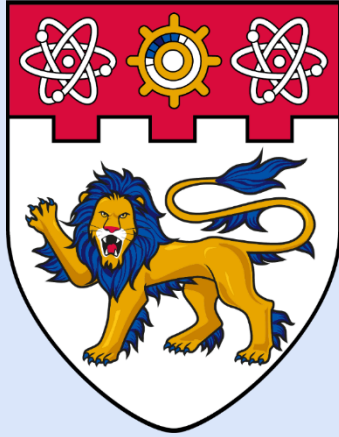
Secondly, we can work to add additional functionalities such as the ability to cancel a waiting number in the queue or provide a "Call Again" button for our counters. These features would improve the overall user experience and ensure a smooth and efficient service.

Lastly, when implementing the solution in a real-world scenario, we can examine the data structure and make improvements to ensure it is clear, meaningful, and tailored to the specific service requirements. By continually evaluating and refining our approach, we can maintain a high level of effectiveness and ensure our solution remains relevant and responsive to the needs of our clients.

## 8.Conclusion

This report provides an in-depth analysis of the Queue Management System, including its various features and functionalities. The aim of this guide is to assist both first-time and experienced users in making the most of the system and meeting their objectives. From getting started and setting up the product to advanced usage and operation, this comprehensive guide covers all the relevant information necessary for a successful implementation of the system. With clear instructions and comprehensive explanations, this guide is designed to ensure a seamless and satisfactory experience for all users.





**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**  
**SINGAPORE**

**AN6007**

**ADVANCED PROGRAMMING**

**Group Project**

## **Technical Report : API Manual for Bank Queueing System**



**Group B -Team 6**  
**LYU, JIAXIN (MISAKI)**  
**YANG, YUXIN**  
**ZHAO, YUECHEN**  
**PENG, LI**  
**GAN, JIUJUN**

**Nanyang Business School**  
**12/02/2023**



# Bank Queuing System API Design Manual

## Introduction

This system aims to establish a queuing system for a bank. Each front-end web page needs to send requests to the server and retrieve queuing information to the server. To enable communication between the front-end pages and the server, our group set up the following APIs based on the requirements of information communication.

## Main HTML

	Application Programming Interface (API) Address	Details of the application
1	<a href="http://localhost:3000/branches/:branchid/getQueueNum">http://localhost:3000/branches/:branchid/getQueueNum</a> <a href="http://localhost:3000/branches/1/getQueueNum">http://localhost:3000/branches/1/getQueueNum</a>	Record a customer queue request based on the category <i>:branchid is the id of the branch</i>
2	<a href="http://localhost:3000/branches/:branchid/counterCallPage/:counterid">http://localhost:3000/branches/:branchid/counterCallPage/:counterid</a> <a href="http://localhost:3000/branches/1/counterCallPage/1">http://localhost:3000/branches/1/counterCallPage/1</a>	Call for the customer to the respective counter <i>:branchid is the id of the branch</i> <i>:counterid is the id of counter</i>
3	<a href="http://localhost:3000/reschedule/branches/:branchid">http://localhost:3000/reschedule/branches/:branchid</a> <a href="http://localhost:3000/reschedule/branches/1">http://localhost:3000/reschedule/branches/1</a>	Re-schedule missed queue number to be served after next 2 <i>:branchid is the id of the branch</i>
4	<a href="http://localhost:3000/CRO/branches/:branchid">http://localhost:3000/CRO/branches/:branchid</a> <a href="http://localhost:3000/CRO/branches/1">http://localhost:3000/CRO/branches/1</a>	View the entire queue list for the respective categories and stop/re-initiate the taking of queue number <i>:branchid is the id of the branch</i>
5	<a href="http://localhost:3000/branches/display/:branchid">http://localhost:3000/branches/display/:branchid</a> <a href="http://localhost:3000/branches/display/1">http://localhost:3000/branches/display/1</a>	Main display page to show the queue serving status <i>:branchid is the id of the branch</i>

## General Operation API

- **Bank branches**

/branches

/branches/:branchID

/branches/:branchID/counters

/branches/:branchID/counters/: countersID

- **change counterType Operation**

/branches/:branchID/setTypeB/counter/:counterID

/branches/:branchID/setTypeP/counter/:counterID

/branches/:branchID/setTypeS/counter/:counterID

- **bizQueue Operation**

```
/branches/:branchID/bizQueue  
/branches/:branchID/bizQueue/push  
/branches/:branchID/bizQueue/insert  
/branches/:branchID/bizQueue/stop  
/branches/:branchID/bizQueue/start
```

- **priQueue Operation**

```
/branches/:branchID/priQueue  
/branches/:branchID/ priQueue /push  
/branches/:branchID/ priQueue /insert  
/branches/:branchID/priQueue/stop  
/branches/:branchID/priQueue/start
```

- **SenQueue Operation**

```
/branches/:branchID/senQueue/branches/:branchID/ senQueue /push  
/branches/:branchID/ senQueue /insert  
/branches/:branchID/senQueue/stop  
/branches/:branchID/senQueue/start
```

- **Reschedule Operation**

```
/branches/:branchID/reschedule/:originalID
```

## Database and Data Demonstration through JSON

Before introducing the functions implemented by each API and the presentation of the functions, this article first introduces the following database used in this project and the data demonstration used for the presentation.

We choose to use JSON as the data storage file.

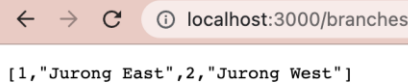
```
[  
  { "branchID":1,"branchName":"Jurong  
East","queuenumber":{"biznumber":3,"prinumber":3,"senmber":3},"branchCounter":[{"cou  
nterID":1,"counterServing":"","type":""},{ "counterID":2,"counterServing":"","type":""}], "busi  
nessQueue":["B-1","B-2","B-3"],"privateQueue":["P-1","P-2","P-3"],"seniorQueue":["S-1","S-  
2","S-3"],"missingQueue":[],"status":[]},  
  { "branchID":2,"branchName":"Jurong  
West","queuenumber":{"biznumber":3,"prinumber":3,"senmber":3},"branchCounter":[{"cou  
nterID":1,"counterServing":"","type":""},{ "counterID":2,"counterServing":"","type":""}], "busi  
nessQueue":["B-1","B-2","B-3"],"privateQueue":["P-1","P-2","P-3"],"seniorQueue":["S-1","S-  
2","S-3"],"missingQueue":[],"status":[]}  
]
```

## API Function Introduction and Display

- Bank branches

**/branches** - This API is used to output all the available branches of the bank.

Webpage output



```
[1, "Jurong East", 2, "Jurong West"]
```

**/branches/:branchID** – This API is used to output all the counters and Queue categories (Business, Private Banking, Priority Queue).

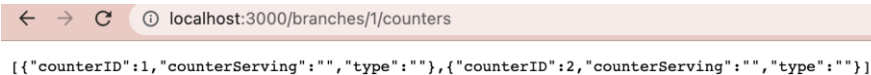
Webpage output



```
{ "result": { "branchID": 1, "branchName": "Jurong East", "queueNumber": { "biznumber": 3, "prnumber": 3, "senber": 3 }, "branchCounter": { "counterID": 1, "counterServing": "", "type": "" }, { "counterID": 2, "counterServing": "", "type": "" }, "businessQueue": { "B-1", "B-2", "B-3" }, "privateQueue": { "P-1", "P-2", "P-3" }, "seniorQueue": { "S-1", "S-2", "S-3" }, "missingQueue": {}, "status": {} } }
```

**/branches/:branchID/counters** – This API is used to output all the counters serving information.

Webpage output

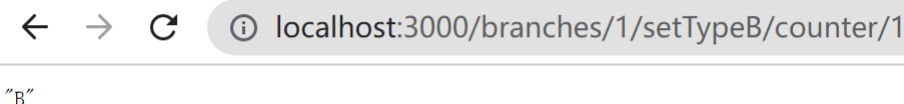


```
[ { "counterID": 1, "counterServing": "", "type": "" }, { "counterID": 2, "counterServing": "", "type": "" } ]
```

- change counterType Operation

**/branches/:branchID/setTypeB/counter/:counterID** – This is to set the counter type to serve a specific queue.  
“B”: Business, “S”: Senior, “P”: Private

Webpage output



```
"B"
```

Same logic applies to the Private and Senior type.

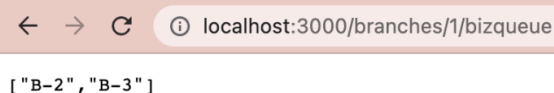
/branches/:branchID/setTypeP/counter/:counterID

/branches/:branchID/setTypeS/counter/:counterID

- bizQueue Operation

**/branches/:branchID/bizQueue** – This is to output the queue number for the bizQueue.

Webpage output



```
[ "B-2", "B-3" ]
```

**[/branches/:branchID/bizQueue/push](#) – This is to push one number into bizQueue. Will show the whole list.**

Webpage output

← → ↻ ⓘ localhost:3000/branches/1/bizQueue/push

["B-13"]

← → ↻ ⓘ localhost:3000/branches/1/bizQueue/push

["B-13", "B-14", "B-15"]

**[/branches/:branchID/bizQueue/insert](#) – This is to shift out a number from bizQueue. Will show the list.**

Webpage output

← → ↻ ⓘ localhost:3000/branches/1/bizQueue/insert

["B-16"]

**[/branches/:branchID/bizQueue/stop](#) – This API will change the status of specific type queue. For example, status of business queue is changed to 0.**

**When clients want to get a number, the function will first identify whether the status of this queue is '1'. Otherwise will tell the clients 'this queue stops'.**

Webpage output

← → ↻ ⓘ localhost:3000/branches/1/bizQueue/stop

{"biz":0,"pri":1,"sen":1}

**[/branches/:branchID/bizQueue/start](#) – This API will change the status of specific type queue. For example, status of business queue is changed to 1. In this situation, the client can take the waiting number.**

Webpage output

← → ↻ ⓘ localhost:3000/branches/1/bizQueue/start

{"biz":1,"pri":1,"sen":1}

**Same logic applies to the Private and Senior queue.**

- **priQueue Operation**

/branches/:branchID/priQueue

/branches/:branchID/priQueue /push

/branches/:branchID/priQueue /insert

/branches/:branchID/priQueue/stop

/branches/:branchID/priQueue/start



- **SenQueue Operation**

/branches/:branchID/senQueue  
 /branches/:branchID/senQueue/isnull (Pending)  
 /branches/:branchID/ senQueue /push  
 /branches/:branchID/ senQueue /insert  
 /branches/:branchID/senQueue/stop  
 /branches/:branchID/senQueue/start

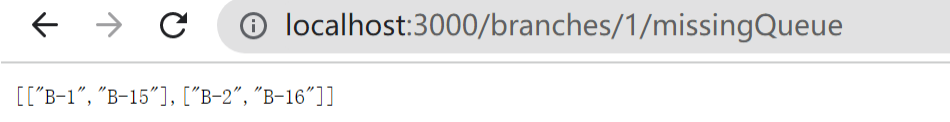
- **Reschedule Operation**

/branches/:branchID/reschedule/:originalID  
 /branches/:branchID/missingQueue

**/branches/:branchID/reschedule/:originalID** – This API help the missing client to reschedule his number for the current next 2 in his original queue type, API need to get *:originalID* parameter from the HTML form. For example, his original number is **B-2**. The next number being served in this queue will be **B-15**. Then, the reschedule-ID for the original number will be B-17 in system, which means he will be served before the real person **B-17**. His reschedule-ID shown to him on the display will be **\*B-2**.

Webpage output	
compare with BizQueue	

**/branches/:branchID/missingQueue display the missingQueue with a list of [originalID, rescheduleID]**

Webpage output	
----------------	---