



Wait Management System

COMP3900 22T3 Computer Science Project

Final Project Report

Due: 18 November 2022 9pm AEDT

Submitted: 18 November 2022

Group: 3900-M18A-Dragon

Jinyu Pan (z5191403) jinyu.pan@student.unsw.edu.au Scrum Team

Mason Pun (z5316520) m.pun@student.unsw.edu.au Scrum Master

Zhi Liu (z5261614) zhi.liu@student.unsw.edu.au Scrum Team

Zhenwei Peng (z5237147) zhenwei.peng@student.unsw.edu.au Scrum Team

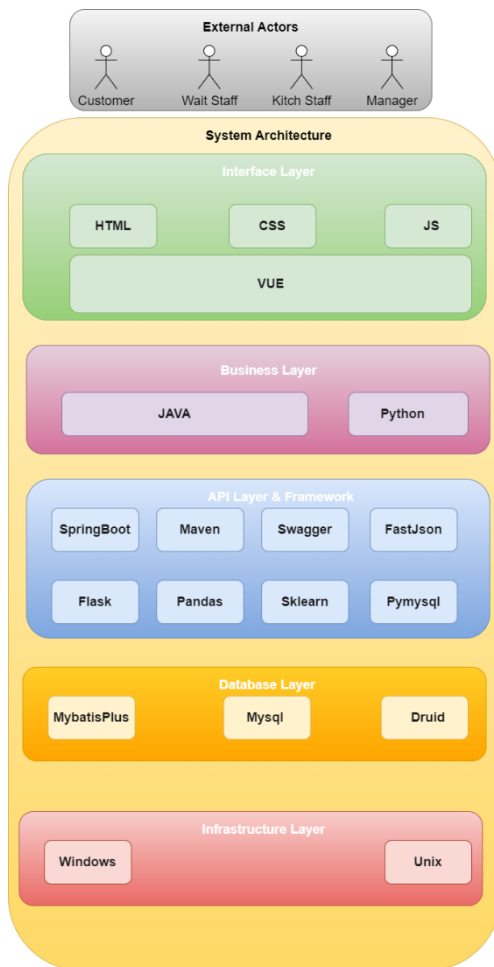
Zhao Zhang (z5355933) zhao.zhang1@student.unsw.edu.au Scrum Team

Contents

1. Overview:	3
2. Objectives and Functionalities:	7
3. Third Party Functionalities:	21
4. Implementation Challenges:	24
5. User Documentation/Manual	25
References.....	27

1. Overview:

1.1 System Architecture Diagram



1.2 External Actors

Following the project proposal, a key factor of this project is to meet the needs of the customers/consumers, our system has kept the originally planned four external actors (Customer/Consumer, Wait staff, Kitchen Staff, Manager) in order to allow customer/consumers to place orders without the need for the staff to serve them.

A customer/consumer is classified as an individual who pick a table, order food, request service from the staff. They also have access to the recommendation system in case they need a menu item recommendation.

The Wait staff will have the ability to login and access the notifications page to see any orders that need serving or customers that need help, labelled in chronological order. Once a job is completed Wait staff will also be able to “Complete” that job, removing it from the list of notifications. They are also able to see their profile page and reset their password if required.

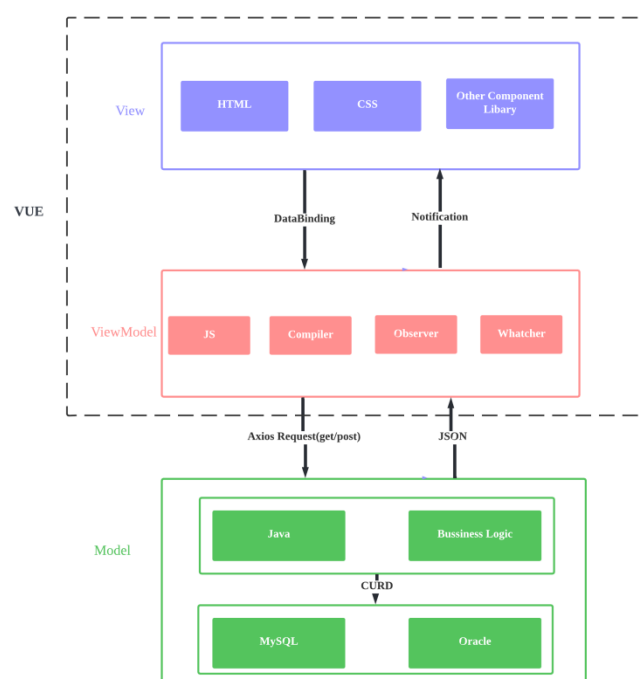
The Kitchen staff will have the ability to see the orders page, allowing them to prepare the customer's orders. Similarly, to Wait staff, they will have the ability to "Complete" a order, which creates a notification for Wait staff that they need to serve that order. They are also able to see their profile page and reset their password if required.

The manager will have the ability to add new menu items, change current menu items, add new categories, sort the order of categories and sort menu items. They also have access to a business recommendation system that will assist in the business plans for their store.

1.3 Project Architecture:

The architecture of our project uses Three-Tier Architecture which separates the application into three logical and computing levels, a Presentation tier, Application tier and Data tier.

Presentation Tier (Interface Layer):



The presentation tier uses the vue.js framework, a progressive JavaScript Framework. It is lightweight and powerful and allows great flexibility to the frontend developers. It has been separated to a VIEW component and a ViewModel component.

The View component primarily uses 2 languages, HTML as the backbone of the webpage and CSS to structure the page layout and for decorations of the components. Element UI, a Vue 3 based component library is also used to allow website design and formatting more efficiently.

The ViewModel component is located between the View and the Application tier layer. This is where objects that the consumer/costumer can interactable with are for the frontend are located. These interactables are bound to the UI elements in the View component.

Finally, Axios request, a promise-based HTTPS library for node.js is used to send responses and receive from the Application Layer. The primary language used for this component is JavaScript. Finally, this layer also consists of Observers, Compilers and Watchers which are used to for interactivity.

The “model” component on the diagram on the previous page is our software’s backend and consists of the Application Tier and Data Tier.

Application Tier (Business Layer):

The application tier is written in two languages, Java and Python. The majority of the code and functionalities are written in Java with our Business Decision Novelty written in Python and connected to the Java codebase via a Flask server.

The application tier is also separated into three layers, Controller, Service and Mapper.

The Controller layer serves as the highest layer that interacts with the frontend through HTTPS requests, its job is to parse the information from the frontend, call Service functions and return information with HTTPS requests to the frontend.

The Service layer manages logic, algorithms of the functionalities. It serves to call the mapper function and return any process results to the Controller layer.

The Mapper layer manages reading and writing to and from the SQL database. It primarily functions as the interactive layer that the Service layer calls when it requires information or needs to update parts of the database.

Separating these three layers allows the Application Tier to say manageable, scalable and more straightforward for backend developers to build on.

The Business Decision novelty is a part of the application layer and uses Python to read from the database and process that data with a Random Forest machine learning model. It avoids the Mapper layer because that creates too much complexity with switching between Java to Python to Java for the Mapper, Service and Controller respectively. Instead, the Controller communicates with a Flask Server to notify the Python code to pull data from the SQL database and process the database, hence the Python code works as a read-only mapper and service layer for the Business Decision Novelty.

Data Tier (Data Layer):

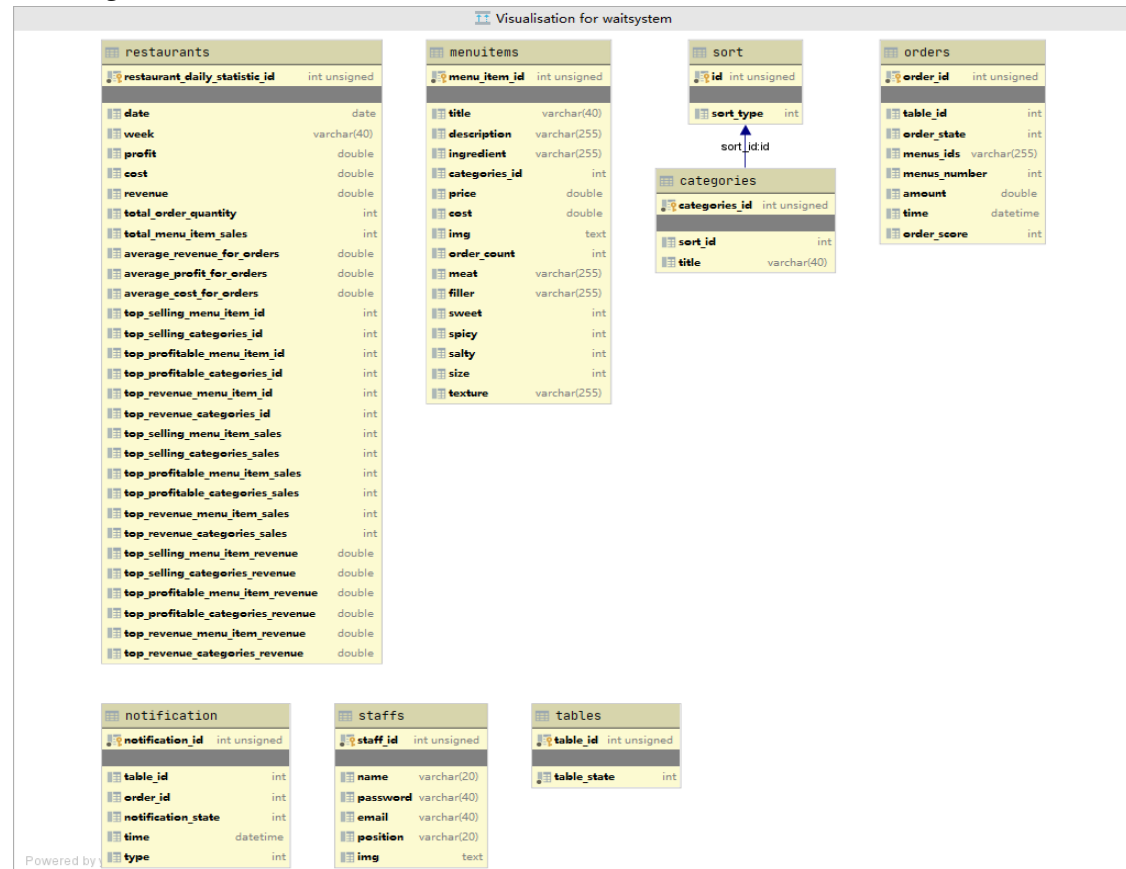
MYSQL is used for the database with the Mapper classes in our java codebase interacting with it. MybatisPlus is the primary library that is used to talk to the database (Please see section 4 “Database & Entities” for more info). MYSQL is used to store the user data and host a local server in where our backend connects to it. Throughout development, we needed a simple way to access and manipulate our SQL database hence we used DataGrip to view and change our database and DataGrip allowed to us to manually prepare a menu and export it as SQL. This is how we kept our

database consistent across the team near the end of development.

Below is an ER-Diagram of our full database, showing all the attributes and any relations.

As mentioned before, Java Mapper classes are the one of the two ways our backend accesses our database, while the Business Decision Python code also accesses the database for read-only purposes

ER-Diagram



2. Objectives and Functionalities:

Note: Due to some of the data being static on the frontend or potential glitches/errors with browser versions, information may not immediately update on the UI. Make sure to refresh the page to get the most up-to-date text.

2.1 Basic Functionalities:

The functionalities of the wait management system are as follows:

1. Staff can register, log in, log out, change and password
2. Customers can select their table numbers when ordering
3. Customers can view dishes on the menu with information such as ingredients, descriptions and cost.
4. Users are able to scroll to view the menu items, and can also quickly locate different dishes through menu categories
5. Customers are able to check their current order list (cart)
6. Customers are able to request assistance from staff
7. Customers are able to request a bill to confirm their order
8. The system will generate customer order lists for kitchen staff
9. Waiter staff can receive notifications from kitchen staff and customers as well as manage notifications (remove them when completed)
10. Managers are able to modify the menu content (such as dishes, descriptions, ingredients, categories, category orders, menu order)

2.2 Novel Functionalities:

1. The manager can use a Business Prediction System that will use machine learning algorithms to predict the venue's future operations in order to support the manager's decision making.
2. Customers have an option to use a recommendation system when ordering.

2.3 Customer functionalities:

Customers come to a restaurant with the sole purpose of ordering their meal and eating, so our goal is to provide the customers with functionalities that create the best customer experience such as the ability to filter dishes by category, add to a cart and place an order.

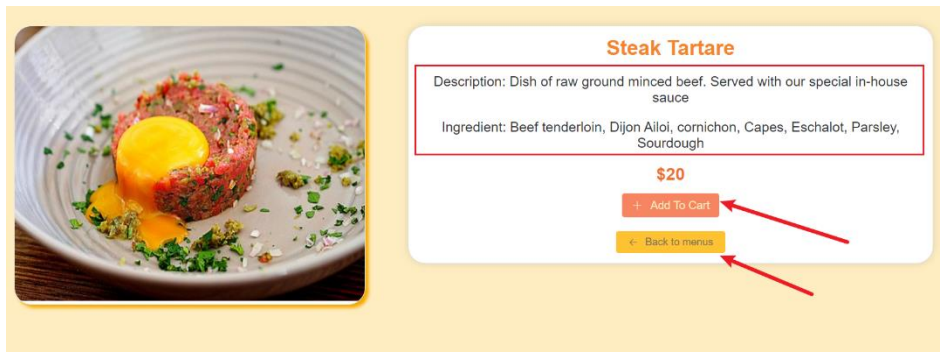
When a customer first enters the UI, they must pick which table they are at.



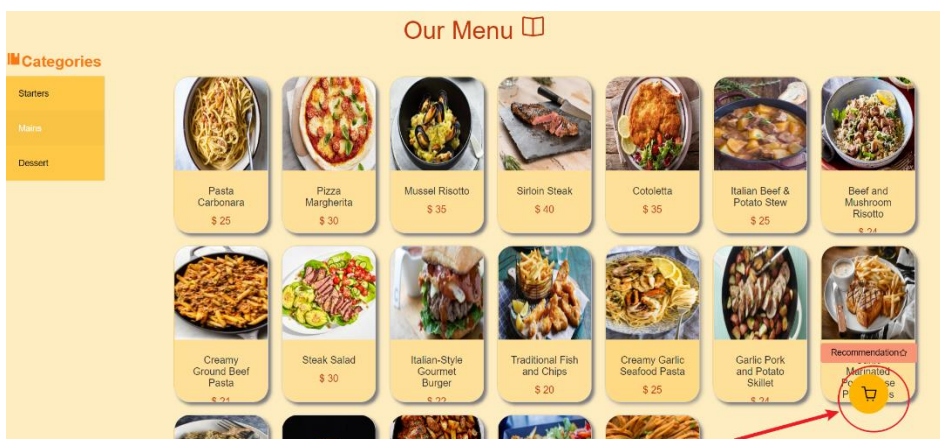
After selecting a table, the customer will be placed in our menu section. On the left there is the Categories, where customers can pick the categories to filter the menu. In the center there is our menu items, with their pictures, menu item names and prices. On the bottom right we can see the cart button, which takes the customer to view their cart. Finally, above the cart button, there is the Recommendation button which takes the customer to a page where they can answer some questions to get a menu item recommendation.



In order to provide the customer with more information to assist in their decision on what to eat, customers can click onto an item to view the menu item details. In the image below, we can see the description of the item, a large rendering of the image of the item as well as the ingredients. In this page is where the customer can add this item to the cart with the 'Add To Cart' button, or return to the menu page with "Back to menus".

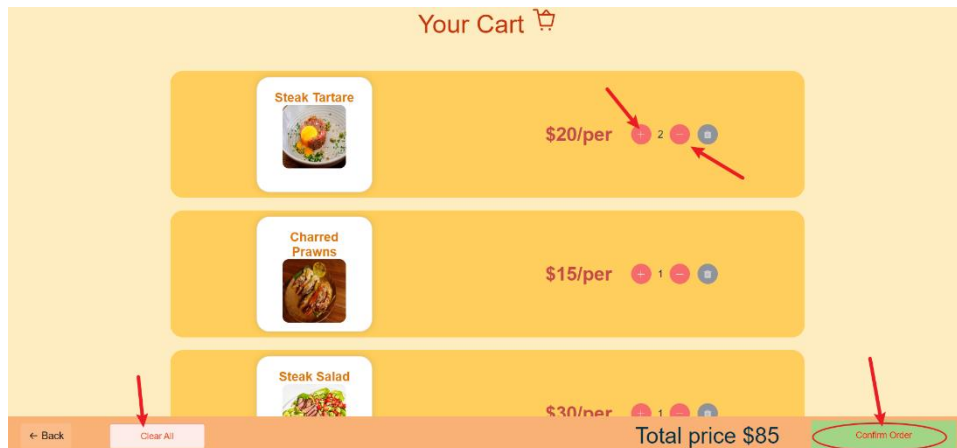


Now after the customer has ordered some menu items, we can click on the Cart button to see what their order is looking like so far.

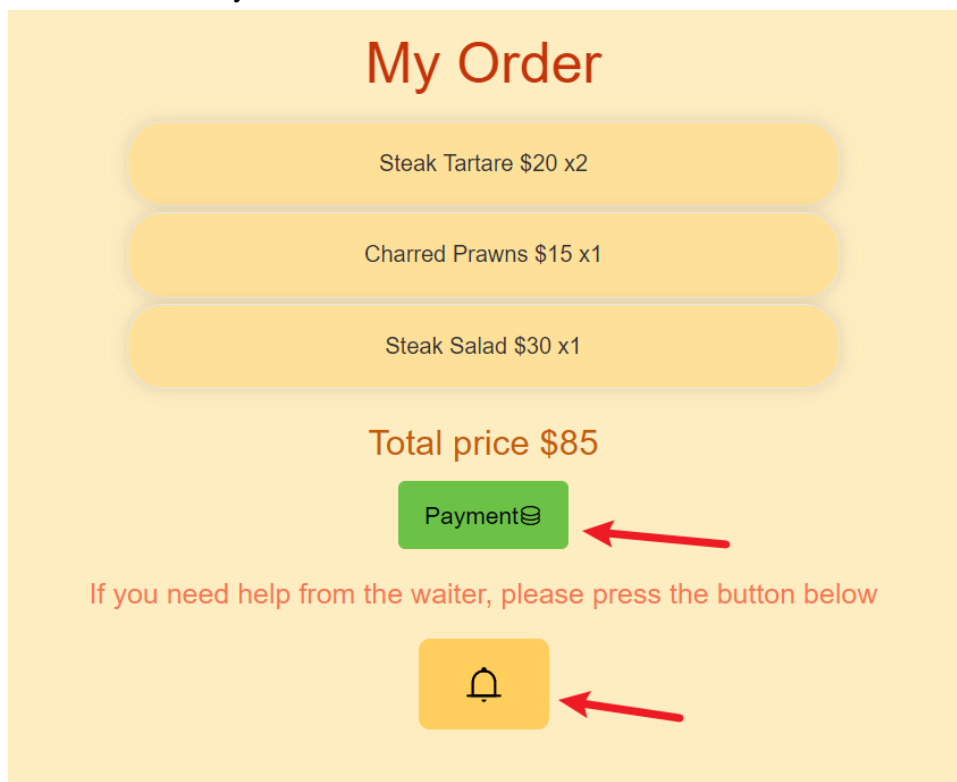


For each item in the cart, there are two buttons that will increase (+) and decrease (-) the quantity of any particular menu item. The total price in the bottom right is also dynamically refreshed whenever items are added and removed from the Cart.

To simplify the user experience, if the quantity of a menu item is decreased to zero, that item is directly removed from the customer's cart. Finally, in the bottom left, there is a "Clear All" button, what will reset the button to an empty cart when pressed.



In the bottom right, the customer can confirm their order with the "Confirm Order" button. Once this is done, the kitchen staff will receive the customer's order and be able to start preparing it. This brings the customer to the page below, where they have the option to pay (once they have finished eating) or request help by pressing the bell button to notify a waiter.



2.4 Kitchen and staff functionalities:

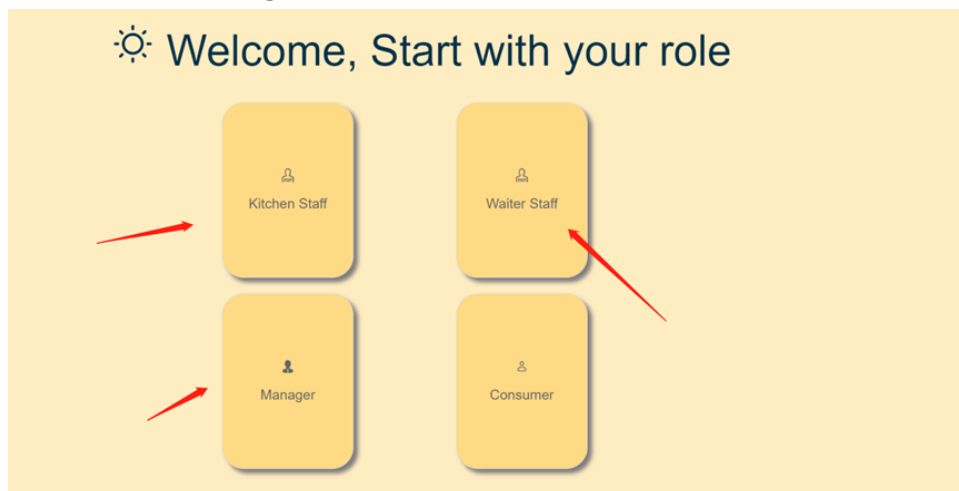
In order to ensure there is no misuse of the staff functionalities no unauthorized parties, we require staff to register and log in whenever they wish to use staff features. There are 3 options of Staff roles (Manager, Waiter Staff, Kitchen Staff, see **Picture 1**) and when the user wants to use these pages, they will be prompted to log in (**See Picture 2**).

In the staff for a new staff member, they have the option to register with the “Sign Up” button (**See Picture 2**) which will bring them to the registration page (**See Picture 3**).

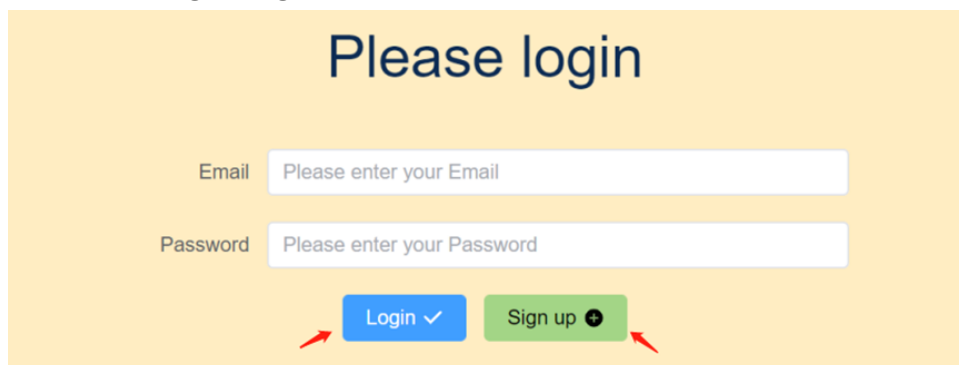
The bonus feature when you log in, is that the system will check which page it should bring you too depending on your role (Waiter, Kitchen, Manager) and forward you to that page when you click the log in button after typing the correct credentials.

On the main page for staff members, they all contain view Profile and Logout buttons (**Picture 4**). In the user profile page, staff can view their details and reset their password when it may be necessary. Once a password is successfully changed, the software will automatically log them out to ensure account security (**Picture 5**). This is all a part of the *Basic Functionality No.1*.

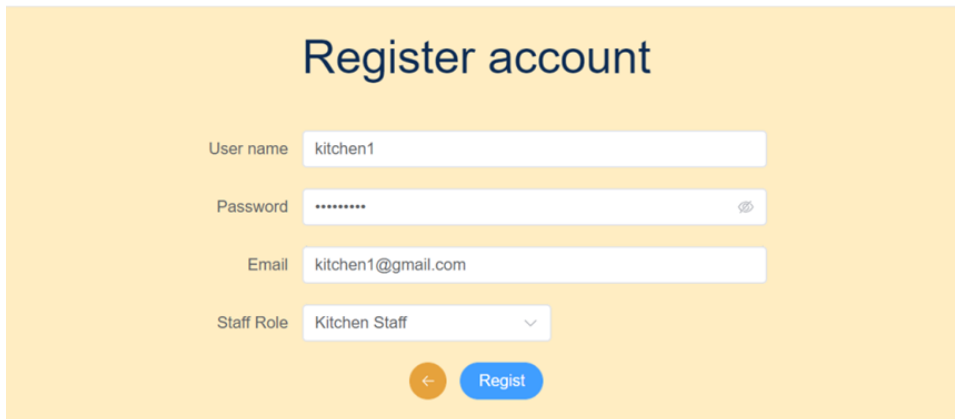
Picture 1: Main Page where we can view the 3 staff roles



Picture 2: Login Page

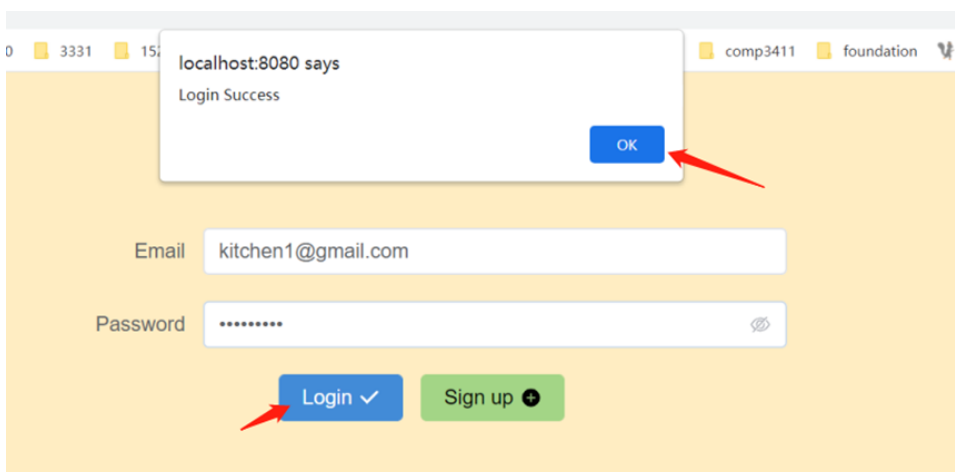


Picture 3: Registration Page



The screenshot shows a 'Register account' form on a light orange background. The form includes fields for 'User name' (filled with 'kitchen1'), 'Password' (filled with seven dots), 'Email' (filled with 'kitchen1@gmail.com'), and 'Staff Role' (a dropdown menu showing 'Kitchen Staff'). At the bottom, there is a blue 'Regist' button and a circular orange button with a left-pointing arrow.

Picture 3.1: After Successful Registration



The screenshot shows a login page with a modal dialog box in the center. The dialog box has the text 'localhost:8080 says Login Success' and an 'OK' button. Below the dialog, the login form has 'Email' (filled with 'kitchen1@gmail.com') and 'Password' (filled with seven dots) fields. At the bottom, there are 'Login ✓' and 'Sign up +' buttons. Red arrows point to the 'OK' button and the 'Login ✓' button.

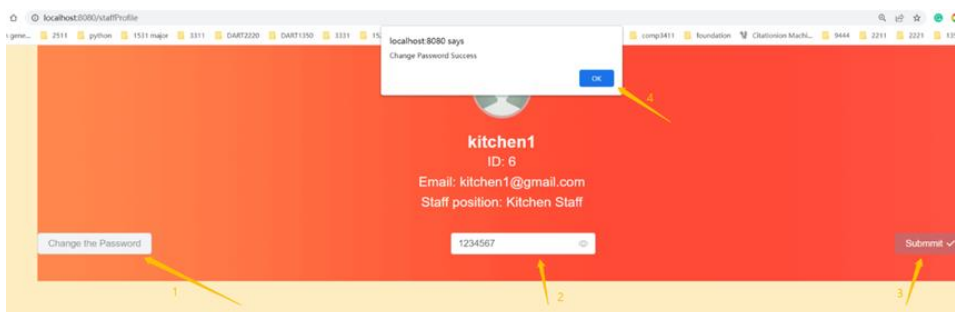
Picture 4: Staff Page (View Profile and Log Out buttons)



The screenshot shows the 'Kitchen Staff Page' with a header bar containing a 'Log Out' button. Below the header is a table with one data row. The table has columns: orderid, tableid, menusids, menusNumber, amount, time, and Action.

orderid	tableid	menusids	menusNumber	amount	time	Action
6	6	.2,9,11	3	60	2022-11-18 11:09:37	Finish

Picture 5: View Profile Page (Prompt is for successfully changing user's password)

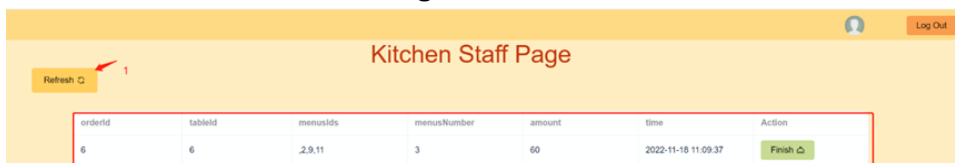


The screenshot shows the 'View Profile Page' for user 'kitchen1'. The page has a red background. At the top, there is a modal dialog box with the text 'localhost:8080 says Change Password Success' and an 'OK' button. Below the dialog, the user's profile information is displayed: 'kitchen1', 'ID: 6', 'Email: kitchen1@gmail.com', and 'Staff position: Kitchen Staff'. At the bottom, there is a 'Change the Password' section with a text input field containing '1234567' and a 'Submit ✓' button. Yellow arrows point to the 'Change the Password' label, the input field, the 'Submit' button, and the 'OK' button in the modal.

When a customer submits their order with the “Confirm Order” button, the order’s details are immediately sent to the Kitchen Staff’s main page. These details include the orderId, tableId, menuItemId and the menuNumber (**See Picture 6**). All orders are displayed by order time, to ensure that the staff’s main page is sorted by chronological order, so that earlier customers get their order’s completed first. Whenever a kitchen staff has completed all current orders, they can click the refresh button to see if any new orders have popped up.

Once an order has been completed, Kitchen staff can click the “Finish” button to complete this order. This will notify the waiter staff that this order has been finished cooking and requires delivery to the table that ordered it (**Picture 7**).

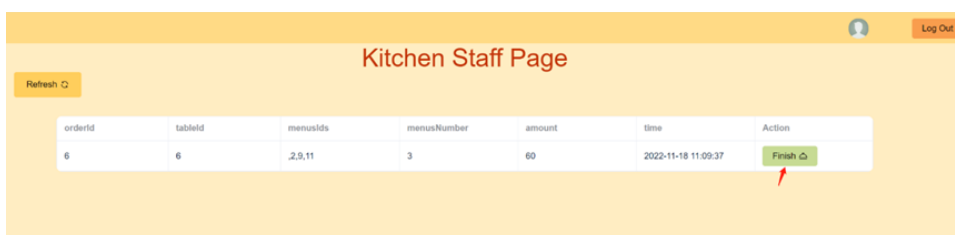
Picture 6: Kitchen Staff’s Main Page



The screenshot shows the 'Kitchen Staff Page' with a yellow header. A 'Refresh' button is on the left, and a 'Log Out' button is on the right. A table displays one order with the following data:

orderId	tableId	menuIds	menuNumber	amount	time	Action
6	6	2,9,11	3	60	2022-11-18 11:09:37	Finish

Picture 7: Kitchen Staff’s “Finish” order button



This screenshot is identical to Picture 6, but a red arrow points to the 'Finish' button in the 'Action' column of the table.

orderId	tableId	menuIds	menuNumber	amount	time	Action
6	6	2,9,11	3	60	2022-11-18 11:09:37	Finish

Picture 7.1: After completing the order it vanishes from this page



The screenshot shows the 'Kitchen Staff Page' after the order has been completed. The table is now empty and displays 'No Data'.

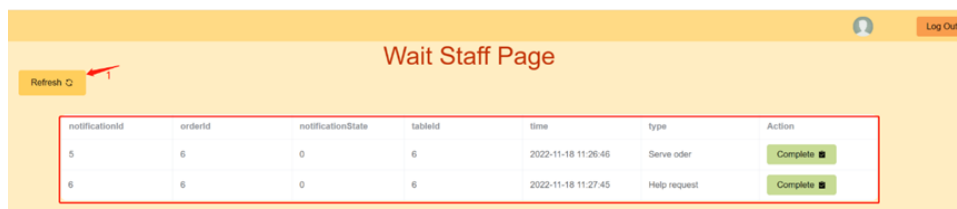
orderId	tableId	menuIds	menuNumber	amount	time	Action
No Data						

Next is the Waiter Staff's Main page. On this page, waiter staff will get notifications whenever the Kitchen staff has clicked the Finish button, as when the kitchen staff is finished cooking, that means it is time for the Waiter staff to delivery this order to the customer's table. Similar to the Kitchen Staff's Main Page, the Waiter Staff's Main Page is also ordered in chronological order to ensure the customers are served as soon as their food is ready. When a waiter completes the act of delivering their order, they will click the "Complete" button to remove this entry from the Wait Staff Page.

Finally, whenever a customer clicks the Bell button to request help from a waiter, this will also create a notification on the Waiter Staff's Main Page.

These features map to the Basic Functionalities Number Eight and Nine.

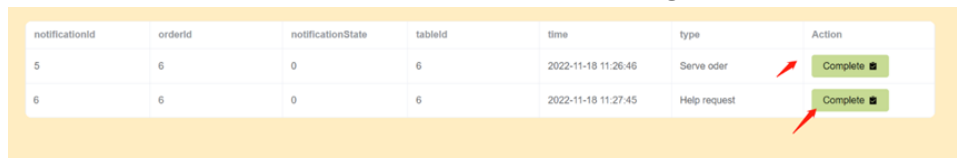
Picture 8: Waiter Staff's Main Page



The screenshot shows the 'Wait Staff Page' interface. At the top right, there is a 'Log Out' button. On the left, there is a 'Refresh' button with a circular arrow icon. A red arrow points to this button. Below the buttons is a table with the following data:

notificationid	orderid	notificationState	tableid	time	type	Action
5	6	0	6	2022-11-18 11:26:46	Serve oder	Complete
6	6	0	6	2022-11-18 11:27:45	Help request	Complete

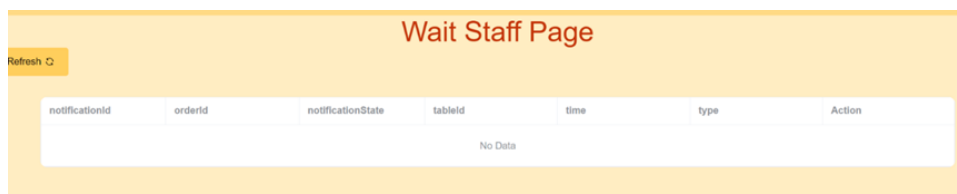
Picture 9: Help Request Notification and Completing orders



This is a close-up of the table from the previous screenshot. Two red arrows point to the 'Complete' buttons in the 'Action' column for the two rows.

notificationid	orderid	notificationState	tableid	time	type	Action
5	6	0	6	2022-11-18 11:26:46	Serve oder	Complete
6	6	0	6	2022-11-18 11:27:45	Help request	Complete

Picture 9.1: After clicking complete button on both orders

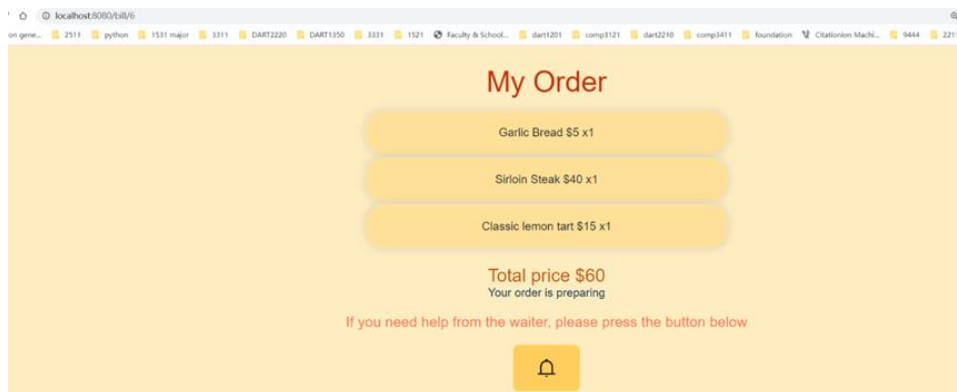


The screenshot shows the 'Wait Staff Page' after the notifications have been completed. The table is now empty, and the text 'No Data' is displayed in the center. The 'Refresh' button is still present on the left.

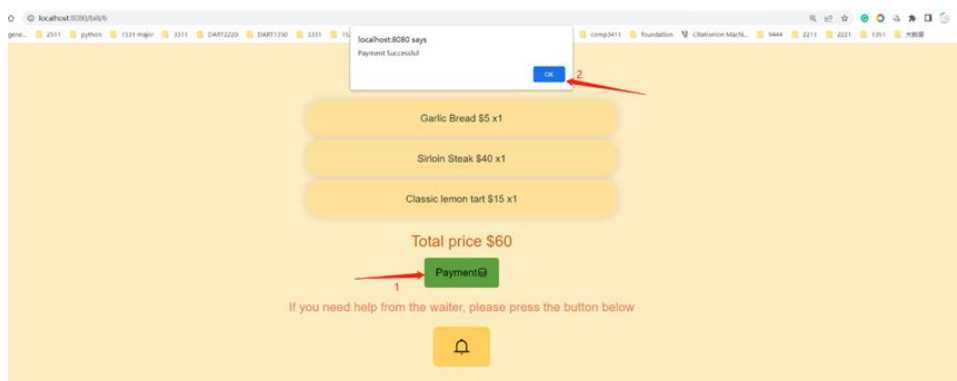
notificationid	orderid	notificationState	tableid	time	type	Action
No Data						

After a waiter staff has completed a customer's order, the customer's "My Order" page will update to allow them to click the "Payment" button. Once the Payment is completed, the customer's particular table will be freed for future customers and they have completed the user cycle for our Wait System. This maps to Basic Functionalities Two and Seven.

Picture 10: Customer's My Order page



Picture 10.1: After Order and Payment has been completed



Finally, the last feature for customers/consumers is the recommendation functionality. The recommendation feature takes in a short survey of responses from the customer and uses those to rank each menu item with a score. Then the items with the top three scores are provided for the customer as a recommendation. This allows the customers to gain some suggestions without the need for a waiter or staff's opinion when the restaurant may be busy. Furthermore, this recommendation functionality is weighted to recommend more popular foods.

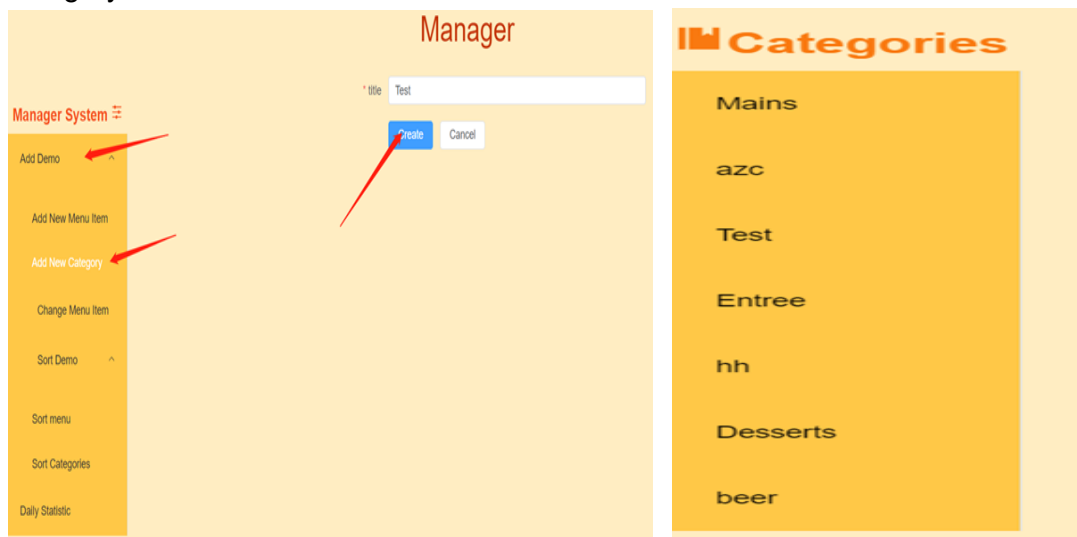
Manager Functionalities:

The manager has the options to Add New Menu items, create new categories, edit existing menu items, sort the menu and reorder the categories.

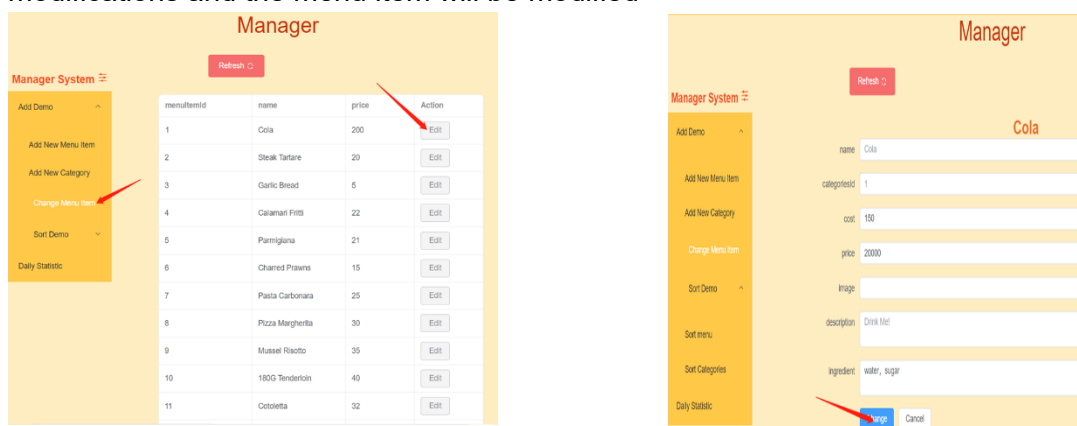
To add a new menu item, click “add Demo”, select the first item 'Add New Menu Item'. After you have entered all the relevant information, click the “create button” to add this item into the menu. This item will then appear in the Menu under the particular category based on the entered “CategoriesId”.

The screenshot displays the 'Manager System' interface. On the left, a sidebar menu lists various actions: 'Add Demo', 'Add New Menu Item', 'Add New Category', 'Change Menu Item', 'Sort Demo', 'Sort menu', 'Sort Categories', and 'Daily Statistics'. Red arrows point to 'Add Demo' and 'Add New Menu Item'. The main area is titled 'Manager' and contains a form for adding a new menu item. The form fields are: Name (test), CategoriesId (1), Cost (20), Image (no image), Price (200), Description (Test), Ingredient (test1, test2), Meat Type (Select), The Side (Select), Texture (Select), Sweet level (Select), Spice level (Select), Salty level (Select), and Size (Select). A red arrow points to the 'Create' button. To the right of the form, a preview of the menu item is shown: a yellow card with the title 'test' and price '\$ 200'. Below the form, a larger preview shows the item in the context of a menu, with the title 'test', description 'Description: Test', ingredient 'Ingredient: test1, test2', price '\$200', and buttons for 'Add To Cart' and 'Back to menu'.

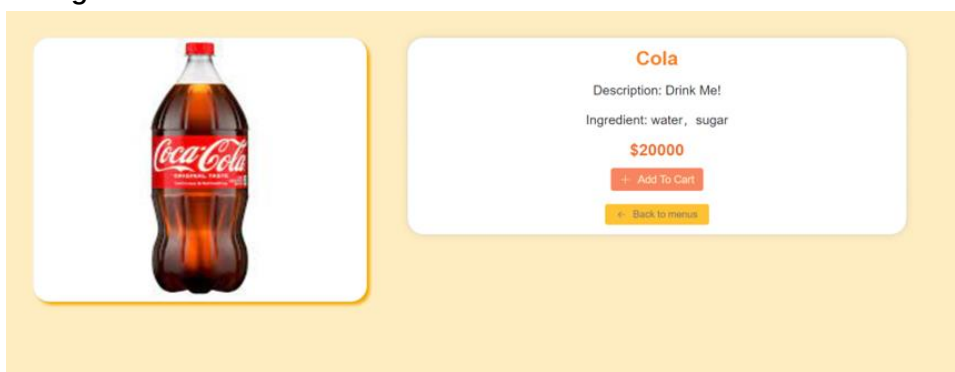
To add a new category, click the “Add New Category” button and enter the name of the new category. Then click “Create”. As shown in the images below, a new category will appear on the menu and the system will automatically assign a category id to the new category.



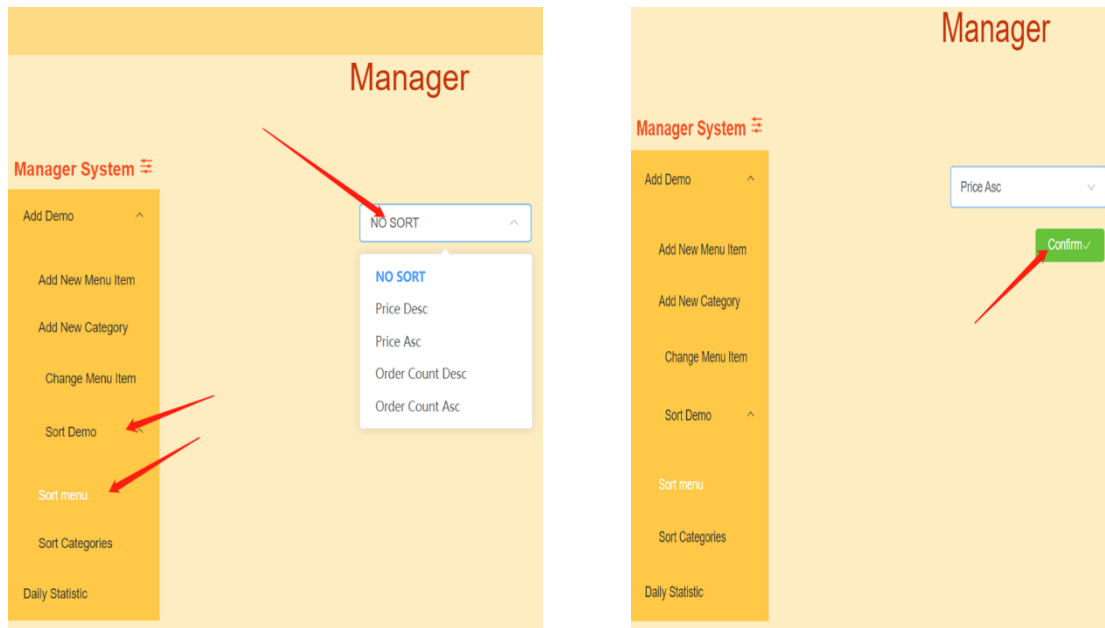
To change existing menu items, click the “Change Menu Item” button and you will be able to view all the menu items. After you have decided which one to change, click the “Edit” button you will be prompted to enter the new details. The previous information will be displayed in translucent text and won't be modified if you do not enter anything into them when you have clicked “Change”. The “Change” button confirms your modifications and the menu item will be modified



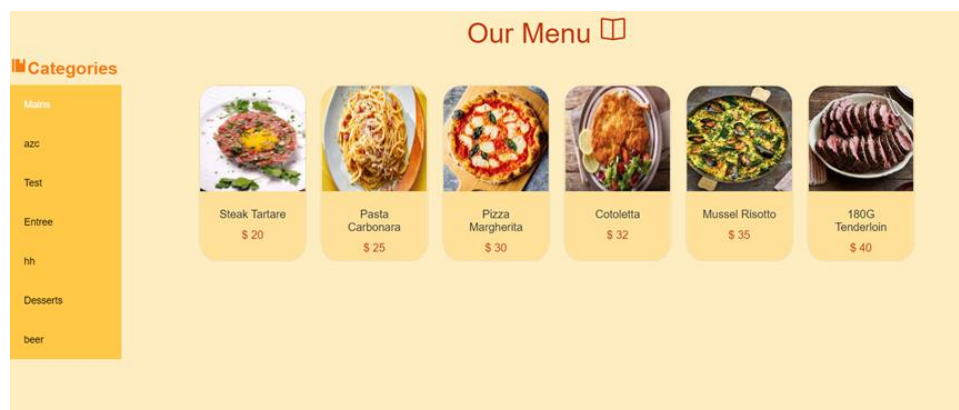
Changed item:



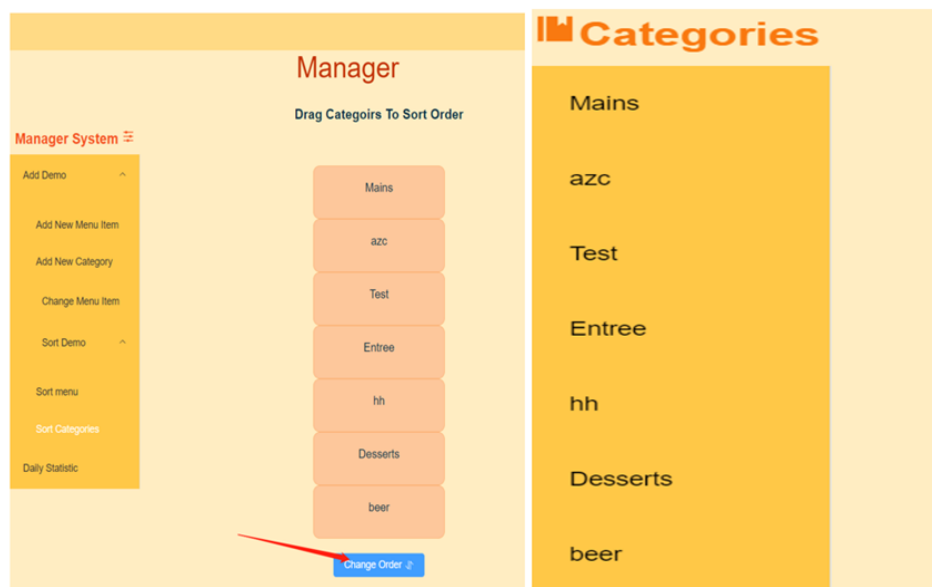
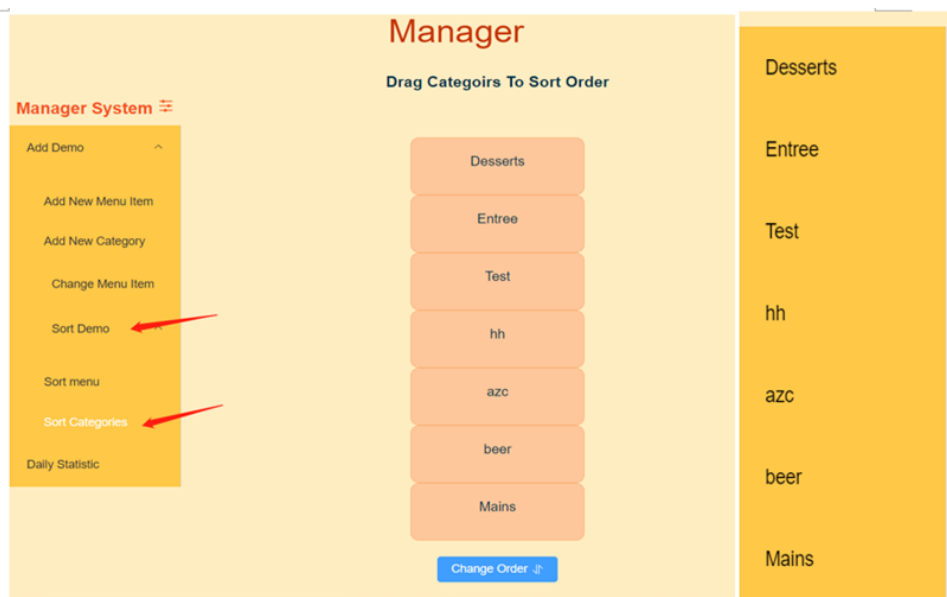
Finally the manager can sort the menu with the “Sort menu” tab. The give options to sort are Default (Sort by menu item ID), Price Descending, Price Ascending, Order Count Descending (from most Popular to least Popular) and Order Count Ascending (from least Popular to most Popular). After clicking confirm the menu will be reordered.



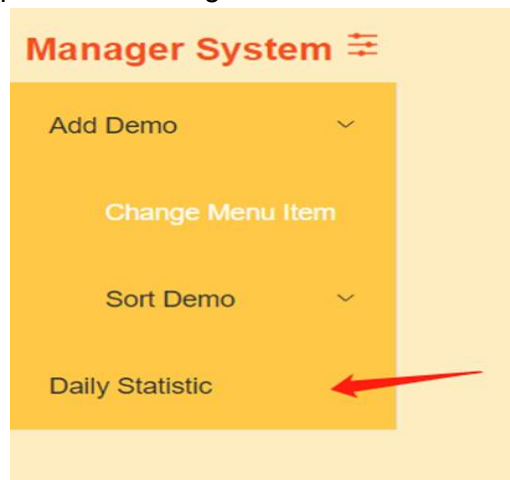
Reordered menu:



To reorder the categories, click “Sort Categories”. On the right you will see all the categories and you can drag and drop them to change the order. After clicking “Change Order” then categories will be rearranged when the menu is refreshed.



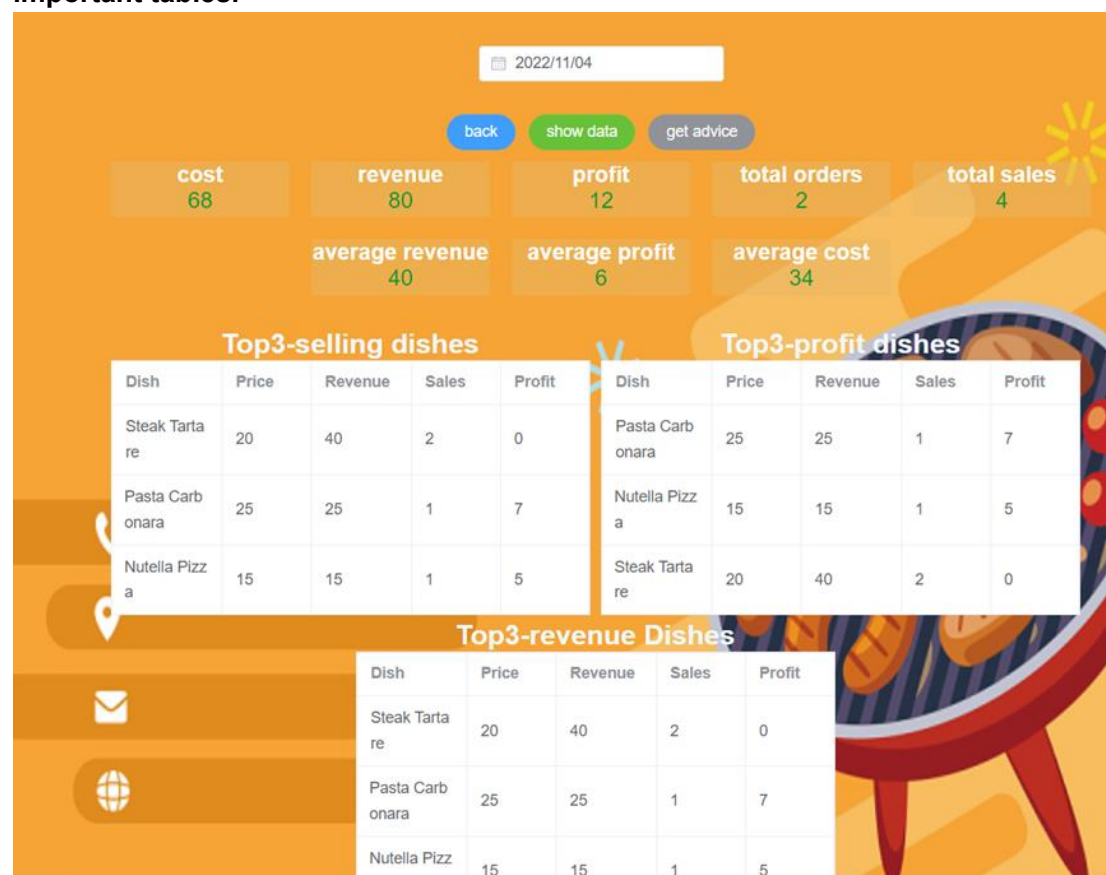
Finally is the Business Decision Functionality. By clicking on Daily Statistic, you can view today's cost, revenue, profit, total orders, total sales, average revenue, average profit and average cost.



At the same time, Managers can view three tables: one that displays the Top 3 Selling menu items, one for the Top 3 most profitable menu items dishes and one for the Top 3 most Revenue-gaining dishes, Top 3 Selling categories, Top 3 Profitable categories and Top 3 Revenue making categories.

With the date at the top of the page, you can select any date to view past sales data. If there is no sales data today, it will display empty. If there are too few orders to have rankings, one or two will be displayed.

Note: In this image below the window shows three tables. When the window size of the frontend is increased, you will see all 6 tables. This is done in order to not clog up the screen real estate and make it challenging for the manager to view the more important tables.



3. Third Party Functionalities:

Spring boot:

In this project we mainly used the Spring boot framework and use Spring boot embedded API such as JDBC, Rest API, Tomcat, Jetty and Servlet etc. Spring boot is an open-source Java-based framework that is used to create a simple micro service, a project. We used MVC pattern for code development for front-end data to reach the server and then to process the request. In general, Spring boot was used to assist with the groundwork of starting any project, allowing us to reduce our code's complexity and smell.

<https://spring.io/projects/spring-boot>

Maven:

We used Maven as the dependency management tool for our project, this will allow us to only configure the pom.xml file to get the jar packages we need. This was crucial to avoid any version conflicts, package conflicts and other errors that may occur from manual package installation. This was a must have, as it allowed the developers to actually work on development, rather than constantly dealing with package conflicts or version conflicts.

<https://maven.apache.org/>

Swagger:

Swagger is an API documentation tool that allows us to generate, execute and visualise HTTP Request formatting (RESTful-style web services) to allows developers to test the return and receive values between the frontend and backend.

This allowed us to confirm the formatting of the JSON Objects that were being sent between these layers, hence allowing frontend developers to quickly understand the needs and functionalities of the backend. Swagger also allows backend developers to simulate a frontend HTTP object to confirm that their controller receives and parses this object correctly and to confirm what their Controller class returns to the frontend.

This package serves a key role in our development to simply the process of communication between the frontend and backend. While allowing us to debug any issues if they may occur with the transfer of data through the HTTP requests.

<https://swagger.io/>

FastJson:

FastJson is a high-performance JSON library written in Java. As our frontend and backend interacts with a JSON data format, we used this API to assist in making our development process smoother. In general, any JSON package could have been used, but we settled on this one because it has been more optimized, allows pre-existing unmodifiable objects to converted to and from JSON and works with custom representations of objects.

<https://github.com/alibaba/fastjson>

Sklearn, Pandas, Flask:

Sklearn, Pandas and Flask are used for the implementation of the Business Decision Functionalities. Sklearn is used for implementing the Random Forest Algorithm while Pandas is used for data processing. Finally, Flask is used to host a lightweight webserver to allow our java backend to communicate with the python Business Decision Program. These APIs provide necessary algorithms and data analysis tools for the Business Decision feature.

<https://scikit-learn.org/stable/>

<https://pandas.pydata.org/>

<https://flask.palletsprojects.com/en/2.2.x/>

MySQL:

The relational database chosen is MySQL because MySQL can operate on almost any platform UNIX, Linux, Windows and most importantly, works with JAVA's JDBC protocol which allows us to use it with Spring boot after simple setup.

<https://www.mysql.com/>

MybatisPlus:

MybatisPlus is an open source, lightweight, data persistence framework that supports custom SQL, stored procedures, and advanced mappings that create mapping relationships between entity classes and SQL statements. This gives us further flexibility in our interactions with our database.

<https://baomidou.com/>

Druid:

Druid is a project on the Alibaba open-source platform that is a database connection pool for monitoring. It provides an efficient, powerful and scalable database connection tool. We are using Druid because database connections are an important but computationally expensive resource. Creating and releasing database connections is a very time-consuming operation, if done frequently it will create a lot of performance overhead, which results in a decrease to the responsiveness of the site, with the potential to lead towards server crashes. Having database connection pooling can save the system a lot of computation overhead.

<https://druid.apache.org/>

Licensing:

Apache License: Spring Boot, Maven, Swagger, FastJson, MyBatis-Plus, Druid

MIT License: Vue

BSD License: Sklearn, Flask, Pandas

GPL License: Mysql

The license agreements for all the third-part APIs are listed above.

All of the above-mentioned APIs are open source software, and we have not modified their source code.

Therefore, even if our project is used for business in the industry, licensing will not have any impact on the Project's publication.

4. Implementation Challenges:

Front-End

Many of the beautiful UI styles are very complicated to derive and to write by hand. During the first two demos we have a very rudimentary and clunky UI but during the later weeks, we focused on learning and utilising the tools in Element-UI. With Element-UI's features we modified our existing frontend to a style that would be more pleasing to the eye and the user's experience.

Database & Entities

In the mapper interface, we use MybatisPlus, a framework based on the idea of "Object Relational Mapping". We map entity classes to database tables, so every time we query and modify a table, we are actually manipulating the properties of an entity and each table stores this entity.

Due to this, we used "select **", because we want the query result as an entity or a list of entities, so that we can use our back-end code to modify the class's instance.

After modification, the entity will replace the data in the database (This process does not require the use of SQL). Due to this framework, despite knowing that the performance of "select *" is poor, we used it as this greatly reduces the amount of SQL writing required, hence reducing the number of connections to the database.

Random forest algorithm (business decision)

For our project we use a flask server (python) in addition to Spring boot. We used machine learning to help managers find the most influential restaurant features. Our backend is in Java, so it became a challenge for us to get the python code to interact with our Java project. This cost is a great amount of development hours, but after the struggles, we Spring Boot's "RestTemplate", alongside a Flask Server, to connect the Java and Python Backends.

The second challenge was to get the dataset for our Python Program. Our data is stored as entities in a table, so I did a lot of calculations on the database data to get enough features. Finally, a dataset with more than twenty features was obtained. These features were calculated based on all the historical orders of the restaurant and other tables. In flask, I accessed our database using python to get these datasets. These datasets were cleaned, filtered and put into our random forest model. For each feature, a different business recommendation was matched. In addition to that, we made a data visualization for each feature, which helps the manager to know which part of the data to focus on (eg: total sales, Total number of orders, Top selling dishes, Highest revenue dishes).

5. User Documentation/Manual

(Lubuntu 20.4.1 LTS)

5.1 Configuration Environment

Before starting the installation, please make sure that the VM can be connected to the Internet.

5.1.1 Install vscode

sudo apt update

sudo snap install --classic code

Type "Visual Studio Code" in the Activities search bar

5.1.2 Clone code from github (Or move the Submission zip file into the Virtual Machine).

Press ctrl+shift+P directly, then enter git, select clone, and enter the git hub account and password

git clone <https://github.com/unsw-cse-comp3900-9900-22T3/capstone-project-3900-m18a-dragon.git>

Rest of the following instructions are placed in the vm-waitssystem in the linux-resource folder)

5.1.3 Install Mysql

sudo apt install mysql-server

5.1.4 Install jdk

sudo apt-get install openjdk-11-jdk

5.1.5 Install pip

sudo apt install python3-pip

5.1.6 Install Node.js & npm

sudo apt install curl

curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -

sudo apt-get install -y nodejs

5.2 Now we Install Project Dependencies

5.2.1 For the Front end

(Using Terminal, change your directory to the “frontend” folder)

```
npm install
```

5.2.2 For machine learning libraries

```
pip install pandas  
pip install flask  
pip install sqlalchemy  
pip install scikit-learn  
pip install pymysql
```

5.3 Now to Set Up the Database

5.3.1 Modify root user password

```
sudo mysql  
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY  
'root';
```

5.3.2 Create Database

```
CREATE DATABASE IF NOT EXISTS waitsystem;  
USE waitsystem;
```

5.3.1 Copy all the contents of FinalDemoDatabase.sql and execute it in mysql to write it to the database

5.4 Now we Run the Project

5.4.1 Run the front-end program

(Using Terminal, with the “frontend” folder as your directory)

```
npm run serve
```

5.4.2 Run the back-end program

(Using Terminal, with the “linux-resource” folder as your directory)

```
Java -jar back-end-0.0.1-SNAPSHOT.jar
```

5.4.3 Run the machine-learning program

(Using Terminal, with the “machine-learning” folder as your directory)

```
python3 app.py
```

References

MyBatis-Plus. (n.d.). Baomidou.com. <https://baomidou.com/>

Druid | Interactive Analytics at Scale. (n.d.). Druid.apache.org.
<https://druid.apache.org/>

Oracle. (2000). MySQL. Mysql.com. <https://www.mysql.com/>

Welcome to Flask — Flask Documentation (2.2.x). (n.d.). Flask.palletsprojects.com.
<https://flask.palletsprojects.com/en/2.2.x/>

Pandas. (2018). Python Data Analysis Library — pandas: Python Data Analysis Library.
Pydata.org. <https://pandas.pydata.org/>

scikit-learn. (2019). scikit-learn: machine learning in Python. Scikit-Learn.org.
<https://scikit-learn.org/stable/>

fastjson. (2022, February 26). GitHub. <https://github.com/alibaba/fastjson>

The Best APIs are Built with Swagger Tools | Swagger. (2019). Swagger.io.
<https://swagger.io/>

Maven – Welcome to Apache Maven. (n.d.). Maven.apache.org.
<https://maven.apache.org/>

Spring Projects. (2019). Spring.io. <https://spring.io/projects/spring-boot>