**UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HA NOI**

**ADVANCED PROGRAMMING WITH PYTHON**

**PROJECT REPORT**

**TOPIC 2**

***Restaurant Information Management System***

**GROUP 10**

**MEMBERS:**

*\_ Đặng Thái Sơn BA9-053*

*\_Nguyễn Minh Hiếu BA9-020*

*\_ Đỗ Công Hòa BA9-022*

*\_Đào Hải Long BA9-041*

*\_Đoàn Văn Chương BA9-008*

**OVERALL CONTENTS OF REPORT**

[I/ Introduction 3](#_Toc73219122)

[II/ The purpose 4](#_Toc73219123)

[III/ The need 5](#_Toc73219124)

[IV/ The creation 6](#_Toc73219125)

[1. Database schema 6](#_Toc73219126)

[*a) Relationship between tables* 7](#_Toc73219127)

[*b) Tables’ functions* 7](#_Toc73219128)

[2. Python modules, classes and packages 8](#_Toc73219129)

[3. UI structure 9](#_Toc73219130)

[*a) Adding* 9](#_Toc73219131)

[*b) Receipt calculating* 10](#_Toc73219132)

[*c) Feature buttons* 11](#_Toc73219133)

[*d) Additional features* 11](#_Toc73219134)

[V/ Conclusion 12](#_Toc73219135)

# I/ Introduction

This is group 10’s project report which makes up 40% of our final results of the “Advanced programming with Python” course. This course, including the most fundamental parts that we should know about Python language namely: OOP, Modules and Packages…, was led by the enthusiasm of Dr. Tran Giang Son.

In this report, we will explain carefully about the program that we showed you in the presentation and demo part on Thursday 27/5/2021: What our program does; Why people should use it; and How we could manage to create the Restaurant Information Management System.

Thank you, Dr. Tran Giang Son, for giving us the chance to access so much valuable knowledge and precious practical sessions about the Python language.

# II/ The purpose

In this project, our group created a fundamental and primitive kind of Python application which is a management tool for a Sushi restaurant. Therefore, it can meet the most basic demands of every restaurant owners.

This program can manage the list of customers, the information of all the dishes by CRUD functions (Create, Read, Update and Delete).

Moreover, it can deal with the receipt calculating task for each customer having meal at the restaurant.

In our opinion, we think that a Sushi business can apply this program into their management system for better efficiency.

# III/ The need

Restaurant is one of the most popular business all over the world, especially in Viet Nam. From the urban areas to the countryside, you can always find a restaurant, either with poor quality or luxury one.

Sushi restaurants are not exceptions. Although so far they have appeared mostly in the cities, the love that customers have for them makes them become more and more trendy in every parts in Viet Nam.

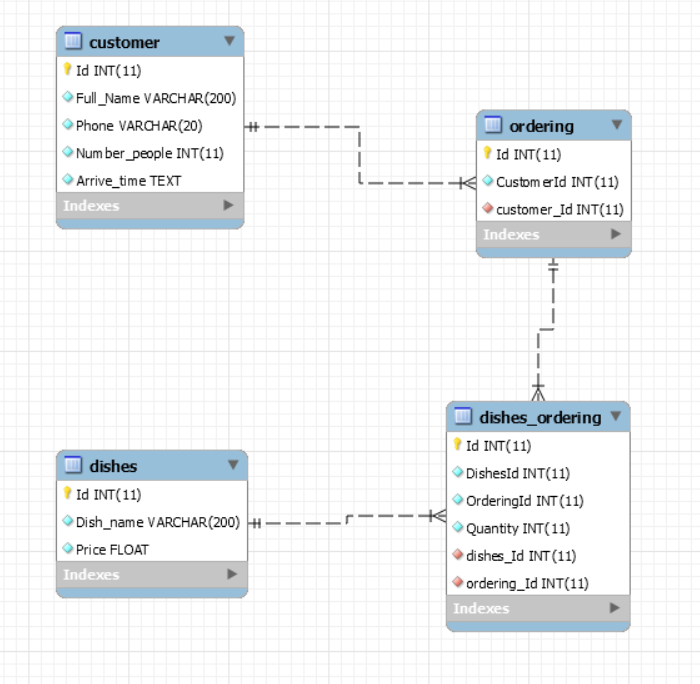
Noticing that not like traditional restaurants focusing one 1 or 2 dishes, a Sushi restaurant usually has an abundant menu with several kinds of food. Therefore, without a proper management system, that restaurant’s owner must have been paid lots of time, money and effort on handling all the information about the business.

Our intention when creating this program was to make it suitable for a medium-size Sushi restaurant. With this application, users can not only deal with the data of all the dishes but they can also keep track of the huge amount of customers having meal every day which are pretty hard to handle by hand. As a result, it can help the restaurants’ managers to save so much time make their work more efficient.

# IV/ The creation

## 1. Database schema

The database is the information source that our whole project bases on. Here, we designed our database to have 4 fundamental tables that every restaurant management system should have, and all of these 4 tables relates to each other:

****

### *a) Relationship between tables*

* Customer – Ordering: one to many relationship (One customer can have many orders but one order can belong to an exact customer)
* Dishes – Ordering: many to many relationship (One dish can belong to many orders and vice versa)

### *b) Tables’ functions*

* Customer: key information of a customer.
* Dishes: data of all dishes in the menu.
* Ordering: contains customers’ ID implementing which order belongs to which customer.
* Dishes\_Ordering: intermediary table between dishes and ordering table which contains the quantity of each dish ordered

## 2. Python modules, classes and packages

Since the amount of code is quite large, we decided so split the whole project into small parts in order to handle everything easily:

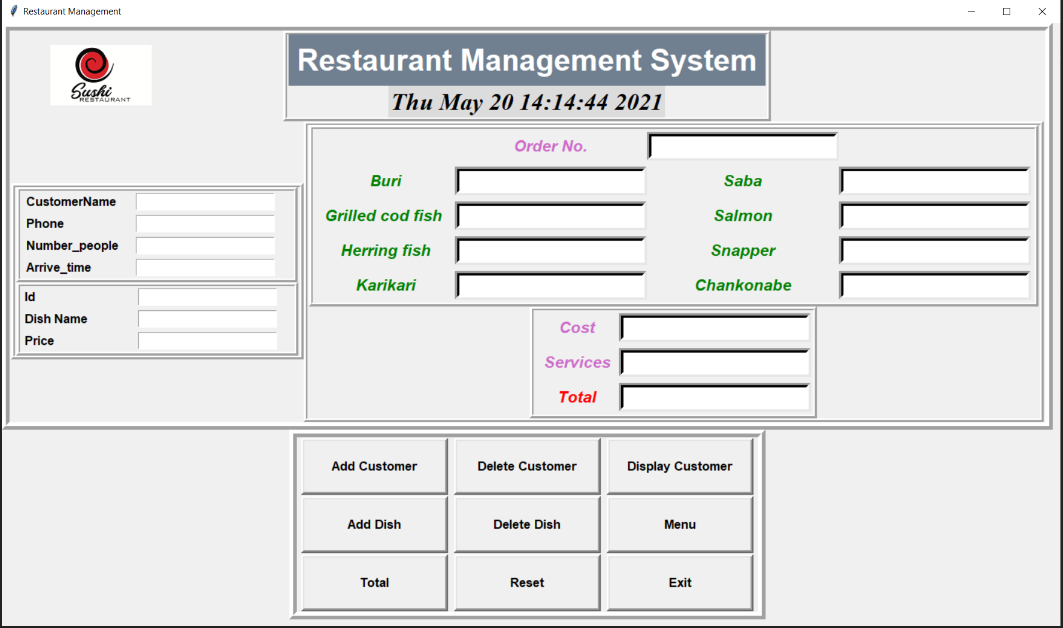
## modul.png

Firstly, we divided the code into 2 packages: **“Domains”, “Images”** and 2 modules **“Interfaces”, “Main”**

* **“Domain”** package: contains everything related to the database
  + **“restaurant.SQL”** file: the database of the restaurant in SQL file
  + **“SQL.py”**: a Python module that deals with the database. It has 2 classes:
    - **“class sqll”** is responsible for the CRUD methods;
    - **“class buttons”** connects the database to the interface. To be more specific, we took advantage of *“mysql.connector”* module in *“mysql-connector-python”* package to link them together
* **“Images”** package: has the images that we use in our interface
* **“INTERFACE.py”**: a Python module that created the layout of the user interface by *“tkinter”* toolkit. It has only 1 class named **“class interface”**
* **“main.py**”: another Python module that is able to run the program

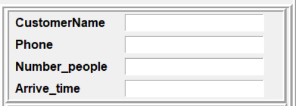
## 3. UI structure

Here is the main user interface that we designed:



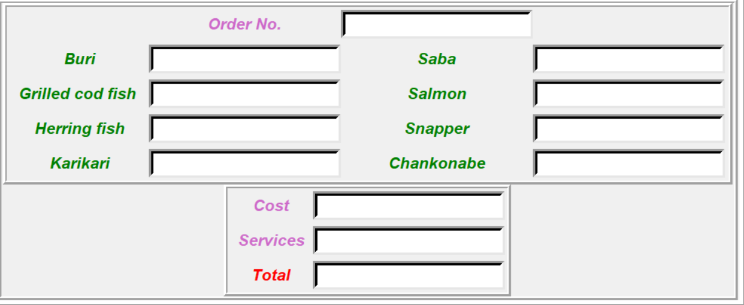
It has three main parts: Adding, Receipt calculating and Feature buttons

### *a) Adding*



In those blanks, user can enter information of either customers or dishes in order to add, delete, update or show by using the feature buttons

### *b) Receipt calculating*



This area is used to show the order of each customer

* + - * “Order no.” part is the ID of the order.
      * 8 blanks under it show all the dishes the restaurant has. User can enter the quantity into it to show how many of each dish a customer ordered.
      * “Cost” is the price the order.
      * “Services” is the price of the service which equals to 10% of the “Cost”
      * “Total” is the total price that a customer must pay (including “Cost” and “Services”)

### *c) Feature buttons*

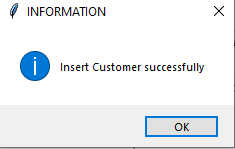
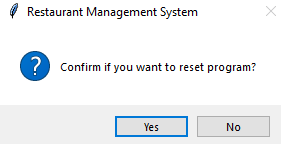
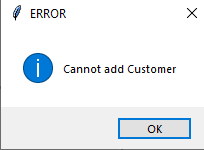


Here is the list of all the buttons that allow users to interact with the program:

* The first row is the CRUD functions for the list of customers.
* The second row is the CRUD functions for the list of all the dishes.
* “Total” button is used to calculate the total price a customer must pay.
* “Reset” button’s aim is for deleting information entered in all the blank.
* “Exit” button is utilized to exit the program.

### *d) Additional features*

To increase the practicality and productivity of the program, we inserted some minor features to support the users. To be more precise, our program can inform people when they use it in the wrong way, validate the manager’s options, or tell the users when the tasks are done.



# V/ Conclusion

To sum up, though there are some weak points in our program such as no ability to add, delete or update several entities at a time; failure to set default of every dishes to “0” in the interface…, we are proud that we were able to make a management system that can not only deal with the primitive CRUD functions but also be capable of doing some improvements like bill calculating.

Again, thanks to your precious lectures and practical sessions, we could manage to apply so much knowledge into our project such as OOP, splitting into modules and packages… to enhance our program.