GROUP 2
Đinh Ngọc Cầm 20226016
Hoàng Mạnh Cường 20226018
Nguyễn Ngọc Quốc Khánh 20226047

[Final Report]
Cafe Store Database Project

31/12/2023

Project Overview

This is a software that helps the owner manage and run the Cafe shop systematically. The owner can have a comprehensive view of the whole system by observing stored data . We will provide that it can be used by employees to manage all orders and a customer can receive a bill that includes the order's details. On top of that, the owner can manage the staff, products, track the list of loyal customers, and revenue...

Project Scope

To make the project of a coffee shop database feasible, we've decided to limit the scope of the project as follows:

The Product

- Beverages: Having 34 types of drinks and 30 types of juices (current data)
- 4 type of Ice Cream, 13 type of Cake (current data)
- Pricing: The initial cost ranges from 5000 VND to 50000 VND, and the selling price will increase by 30% compared to the original value. We do not individually count each

ingredient for every product; instead, we only track the quantity of each item in stock. (current data)

The Staff

Have 3 role with corresponding certificate or award

- Chef
- Bartender
- Waiter

The rating for a staff made by customers is varied from 1 - 5 stars, then from the total stars and the number they are rated, we can calculate the average rating of them. We also limit the maximum award of a chef or a bartender to 1 for easy implementation. However, the certificate must be multivalued - attribute because they may need to update their skill frequently.

The Sitting

- Facilities have 3 floors
- Each floor contains 26 tables
- Tables are marked from 1 to 78

Team Organization

Hoang Manh Cuong	Nguyen Ngoc Quoc Khanh	Dinh Ngoc Cam
• Create indexes, view • Create constraints and triggers • Provide testing data	• Make UI demo • Create basic SQL function • Set project scope	• Create trigger, function • Design ERD model and RS • Write project reports

Database Design

Design Process

- Upon the task of designing the database, our team have spent a considerable amount of time to:
- Brainstorming what's needed.
- Discussing with each other.
- Getting feedback from the instructor.

- Referencing existing systems and resources from the website.

Entity-Relationship Diagram

To design the diagram, we followed a top-down approach. Using the top-down approach for ERD design can help ensure that the resulting database accurately represents the system being modeled, while also making it easier to maintain and update in the future.

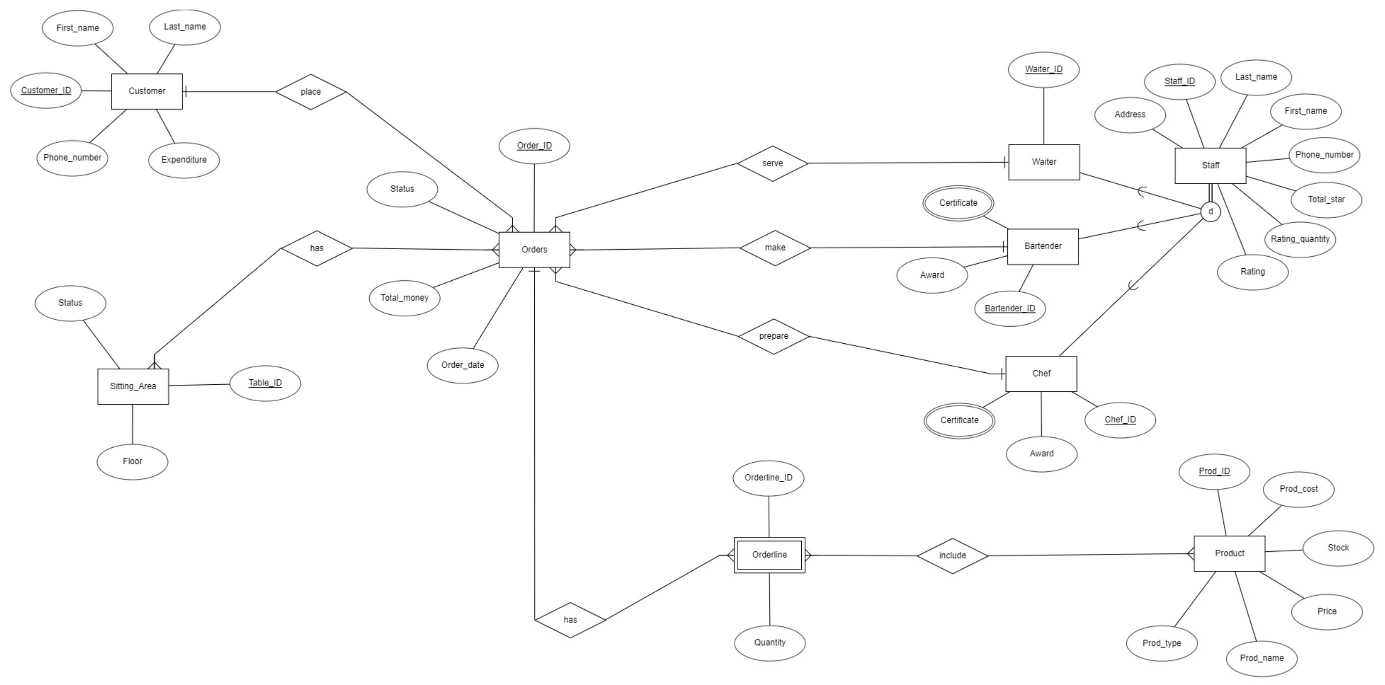
1. **Identify the system**: The first step in the process is to identify the system that needs to be modeled. It is important to define how the system looks when we finish. The system also needs to be realistic and can be applied to solve business problems. To do that, we've based on some basic demands such as managing orders, managing staff, computing revenue,...
2. **Identify the entities**: Once the system has been identified, the next step is to identify the entities that are part of the system. We've chosen the following entities that meet the needs of the system:
    a. Customer - storing information of customers for preferential price later
    b. Sitting_Area - including status and detail of table on each floor
    c. Orders - recording information of orders
    d. Orderline - where products chosen by customers are stored. Because it depend on Orders, so it must be a weak entity
    e. Product - detail of products like name, price, stock,...
    f. Waiter - who serve an order, including basic information like name, address, phone number,...
    g. Bartender - who prepare drink for an order, the same information with the former role but this role have additional fields to store certificate, award
    h. Chef - who make food for an order, fields are similar to Bartender
    i. Staff - keeping all information of 3 roles above

    ➔ Since we divide the staff into 3 types, we need to use a structure including Supertype and Subtype.

3. **Identify the Relationship**: After identifying the entities, the next step is to identify the relationships between these entities. In our system, we decided the relationship between each entity as following:
    - A **customer** can place many **orders** but an **orders** only belong to a **customer** ➔ One - Many
    - At first, we decided an **order** could have just only a **table** but a problem occurred: what would happen if the number of corresponding customer was larger than the number of chair ➔ Many - Many.

- In general, an **order** can be served by **one** staff for each role but a **staff** can serve many different **orders** ➜ One - Many
- An **order** may include many **products** but the reverse is wrong ➜ One - Many
- The last relation is orderline with product. Of course, it must be Many - Many

4. **Create the ERD**: With the entities and relationships identified, the next step is to create the ERD. The ERD is a visual representation of the system that shows the entities and their relationships. We created the ERD using a free online tool called ERDPLUS.
5. **Refine the ERD**: Once the initial ERD has been created, it is important to refine it to ensure that it accurately represents the system. This involves identifying any missing entities or relationships, as well as ensuring that the relationships are properly defined.
   ➜ This process took up a lot of time since things can get confusing with natural language and initially everyone has a different view and expectation of what the model should be.
6. **Implement the Database**: The final step in the process is to implement the database based on the ERD. This involves creating tables for each entity and defining the relationships between them ➜ Relational Schema.

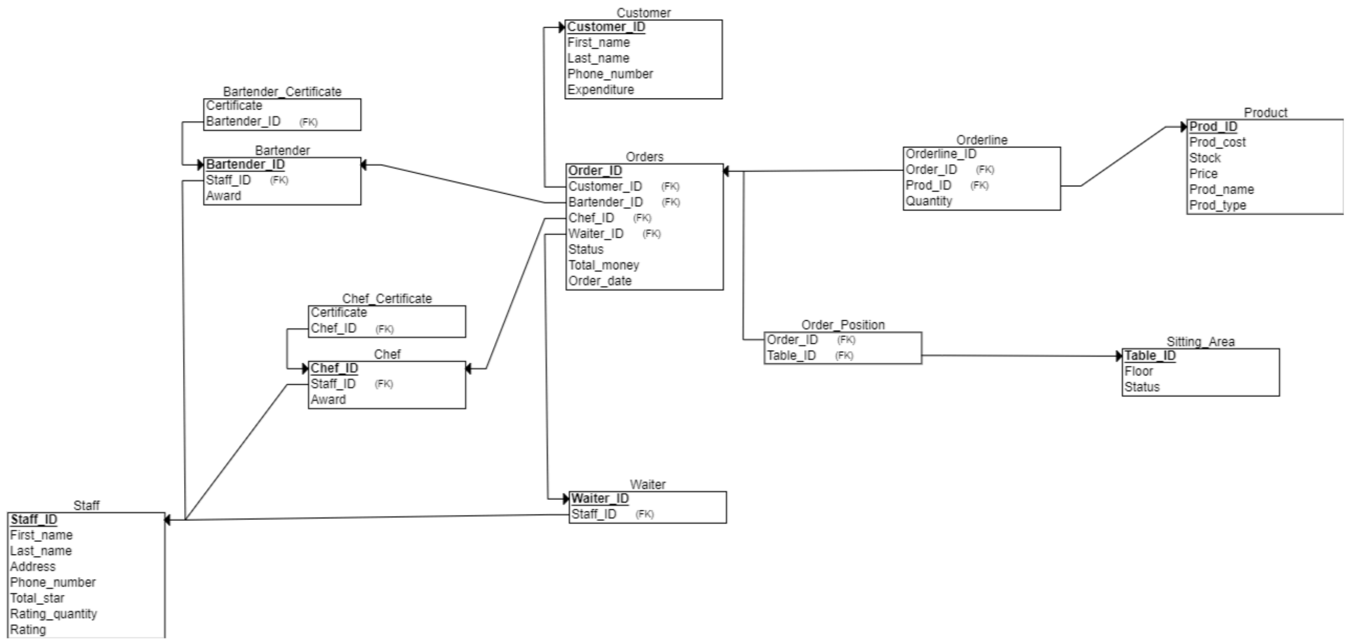


Relational Schema

The process to create a relational schema involves identifying the entities that need to be represented in the database, defining their attributes, and creating tables that represent each entity. We created the relations by following these steps:

1. **Identify the entities:** Each entity or relationship will correspond to a table in the relational schema, and each attribute will correspond to a column within that table. ➜ Based on the ERD above, we can ensure that there are at most 16 tables for this systems
2. **Create tables for each entity and relationship:** Create table identified from previous step. An important point is that the table created from relationships must include just 2 fields (2 Primary Key with corresponding adjacency entities) except weak entities.
3. **Normalize the schema:** After creating tables, we need to consider what tables can be merged to reduce redundancy. For more detail, we've done as following:
    - Customer - Orders: Their relationship is One - Many and we know that the order_id in Place table is unique. So order_id will be considered as **primary key** of Place table and since it shares the same **key** with Orders table, we can merge them into one table (Add a column as **foreign key** customer_id in Orders table and drop Place table). Customer table is kept original. The same process was applied to relationships between Orders and 3 different roles of staff (Chef, Bartender, Waiter).
    - Sitting_area - Orders: Since their relationship is Many - Many so in the relationship table Has, no field can be considered as **key**. But the combination of Table_ID and Order_ID is unique, so we can not merge any tables. We just renamed the Has table ➜ Order_Position.
    - Orderline - Product: Because Orderline is a weak entity, so Order_ID must be included in this table. On top of that, the relationship table Include has no field with unique value even if we combine all fields. Therefore, our solution is to drop the relationship table Include and add 2 **foreign keys** to the Orderline table. They are Order_ID and Prod_ID.
    - Multivalued Attributes such as **certificate** hold a separate table with corresponding **foreign key.**
    - It is important to notice that, in this system, we've applied Inheritance. It means when a staff with a particular role is inserted to one of three subtype tables (chef, bartender, waiter), the system auto updates information to the supertype table (staff). By default, the **primary key** and **foreign key** are **not** inherited from supertype. Therefore, for easy query, we've decided to choose staff_id from the Staff table as **primary key** and in 3 subtype tables, **staff_id** exists as a foreign key. Another reason is when operating the system, the user may encounter a problem that a

staff_id may occur many times in the Staff table since we just insert new staff into the subtype table. We've provided a trigger solution for this problem.



Introduce Triggers & Constraint

The process of identifying needed constraints and triggers involves analyzing the database requirements and determining what rules need to be enforced and what tasks need to be automated.

Constraints can be used to enforce rules such as unique values, referential integrity, and check constraints,... We defined the constraints as follows:

- Primary key - Foreign key.
- The phone number of staffs and customers must be NOT NULL and UNIQUE.
- The "Bartender," "Chef," and "Waiter" tables inherit from the "Staff" table. This means they automatically include all attributes of the "Staff" table. Additionally, each role-specific table can have its own unique attributes. This setup ensures a clear and concise representation of information, allowing for both sharing and role-specific details.
- Staff_id in Chef, Bartender, Waiter table are set as UNIQUE, we just need to add an arbitrary number for this field and trigger will handle duplication issues.
- Other constraints were also added to make sure that the system can run without error and appropriate with reality.

Triggers can be used to handle automatic tasks. Our trigger include:

- When selecting a product for an order, check its availability. Afterward, deduct the corresponding quantity from the product's stock.
- Calculate the total price of each order
- When an order is completed, add the total_money to the expenditure in Customer relation for further preference.
- It also auto calculates the rating of a staff when they are rated by customers after an order (The logic we using is Rating = Total_star / Rating_quantity)
- As I mentioned above, when we add a new staff with a particular role, the staff_id in the **subtype** table will automatically update in the **supertype** table. It seems to be an advantage but staff_id can occur duplication in the staff table. So we've created a trigger to handle that. It's simply update the staff_id in **subtype** table (like **chef**) by MAX(staff_id) + 1 from **staff** table

When the identify process has been done, constraints and triggers are created using SQL statements. We tested them to ensure that they are functioning properly and not causing any errors when using.

Create Functions

When writing functions for a database, we need a good strategy to ensure that these functions can perform efficiently and are easy to maintain.

1. **Identify the requirements**: In this project, we've chosen to serve the store owner (or managers). Some main function are:
    - Calculate the total revenue for the previous n months at a given time.
      - This function, named 'calculate_revenue_custom_period,' enables the manager to assess the shop's revenue over a specified period of n months. By providing the start date and n months previous, the function outputs the total revenue accumulated during this timeframe.

    - Display the transaction history of a customer
      - This function 'get_trading_history_for_person' shows all the history transactions of 1 customer so that if a customer has a query or concern about a specific transaction, having access to their complete transaction history allows customer support representatives to provide more accurate and efficient assistance. It enables them to understand the context of the customer's inquiry and resolve issues more effectively. By providing the id of the customer, the function outputs all details of the customer's history transaction.
    - Calculate accrued interest all the time
        - Calculating accrued interest is a fundamental accounting practice that supports accurate financial reporting, ensures compliance with accounting

standards, and assists in various financial decision-making processes. In function ‘calculate_total_profit’ outputs the cafe shop’s total profit.

- Calculate revenue from [start date] to [end date].
    - Function ‘calculate_revenue_within_time_range’ will help the customer to calculate the revenue taken from start_date end end_date. Then the output will print out the money.
- Display employees have the highest rating.
    - A function ’get_highest_rated_staff’ to print out employees with high ratings so that managers can identify those who receive positive feedback from customers and consider them for possible rewards. The output will print top 10 employees having the highest rating.
- Add product to the menu.
    - Sometimes, the manager wants to add new items or seasonal dishes to the menu, so this function add_product will help accomplish that. This function will input some basic information of 1 dish like name, type, product_cost, stock, price, then if inserting is successful, it will print out the new product_id.
- Add the new customer.
    - This function takes basic customer information such as name and phone number as input, and upon successful execution, returns the customer_id corresponding.
- Delete customer.
    - This function deletes all information associated with a single customer, including any orders made by that customer.
- Find the items in the cafe shop by its name.
    - This function ‘find_items_by_name’ will help the customer to find out the dishes in the menu of the cafe shop by its name. Input the name of the dish then output will show all details of this item so that the manager easily manages.
- Show order bill for customer:
    - Function generate_bill takes an input parameter and will display some basic detail about order such as order_id, customer_id, list of products. We hope that this function can give the customer apparent detail.

2. **Design function**: Determine the logic that the function will use to give the desired output and try to optimize the syntax.
3. **Write Syntax:** Our syntax was save in a file, you can see it in the file **FUNCTION.sql**
4. **Test Function:** Once the syntax has been written, we test the function to ensure that it will produce desired output and perform efficiently. We also test the function with a variety of data to ensure that it will work precisely in all cases.
5. **Optimize Function:** Optimizing is a crucial step to increase the efficiency of the function when a whole data was collected. To do it, we’ve decided to create index for some

function; details of the index are represented in the next part. The main method that we use is B-tree and it can speed up the function by about **19.5%** faster.

6. **Document Function:** We've decided that all necessary functions we've worked on step (2) will be stored.

Index Analyze

To enhance the efficiency of calculating total revenue, profit, trading history, and total price of orders, we utilize BTREE INDEX for several columns. These columns are involved in numerous join operations, and B-trees contribute to improved performance in such scenarios. Indexing facilitates quicker location of relevant rows, therefore reducing the time and resources needed for join operations.

BTREE INDEX on ORDER_ID in ORDERS table.
BTREE INDEX on CUSTOMER_ID in CUSTOMER table
BTREE INDEX on PROD_ID in PRODUCT table.
BTREE INDEX on STAFF_ID in STAFF table.

	QUERY	Without INDEX	WITH INDEX
1	Top 10 best seller items in cafe shop	0.061s	0.051s
2	Top 10 customer having the highest spending	0.062s	0.053s
3	Get trading history of one customer	0.088s	0.045s
4	Find all bills in 5 months	0.065s	0.063s
5	Find all customer have bill in 5 months	0.064s	0.053s
6	Calculate the price of an order	0.142s	0.118s

This result is relative because it also depends on your CPU, MEMORY,...

View

Based on some basic needs that often used by user, we've created the following views for easy operation:

1. **View best_seller:** displays the menu's top-selling items to assist management in effortlessly reviewing and adapting subsequent strategies for boosting sales, along with corresponding vouchers.

2. **Customer having highest expenditure**: Display the top most loyal customers to help management create exclusive offers for them.

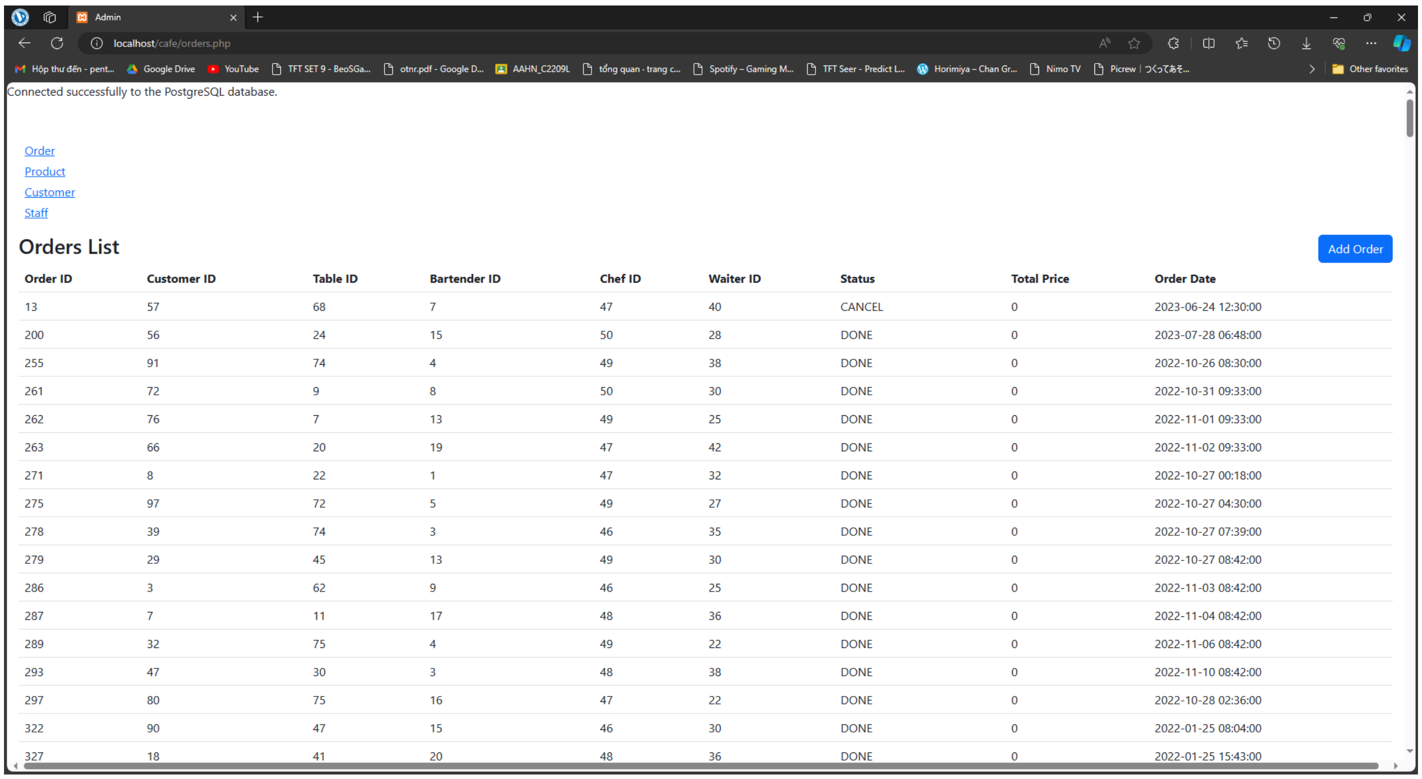
User Interface Demo

Our User Interface is written in PHP and JavaScript and linked to PostgreSQL.

How to use:

- Install XAMPP
- Copy project folder to xampp/htdocs
- Run Apache service
- Go to php file at localhost/[ProjectName]/[filename.php]

Our Order page demo:



Connected successfully to the PostgreSQL database.

Order
Product
Customer
Staff

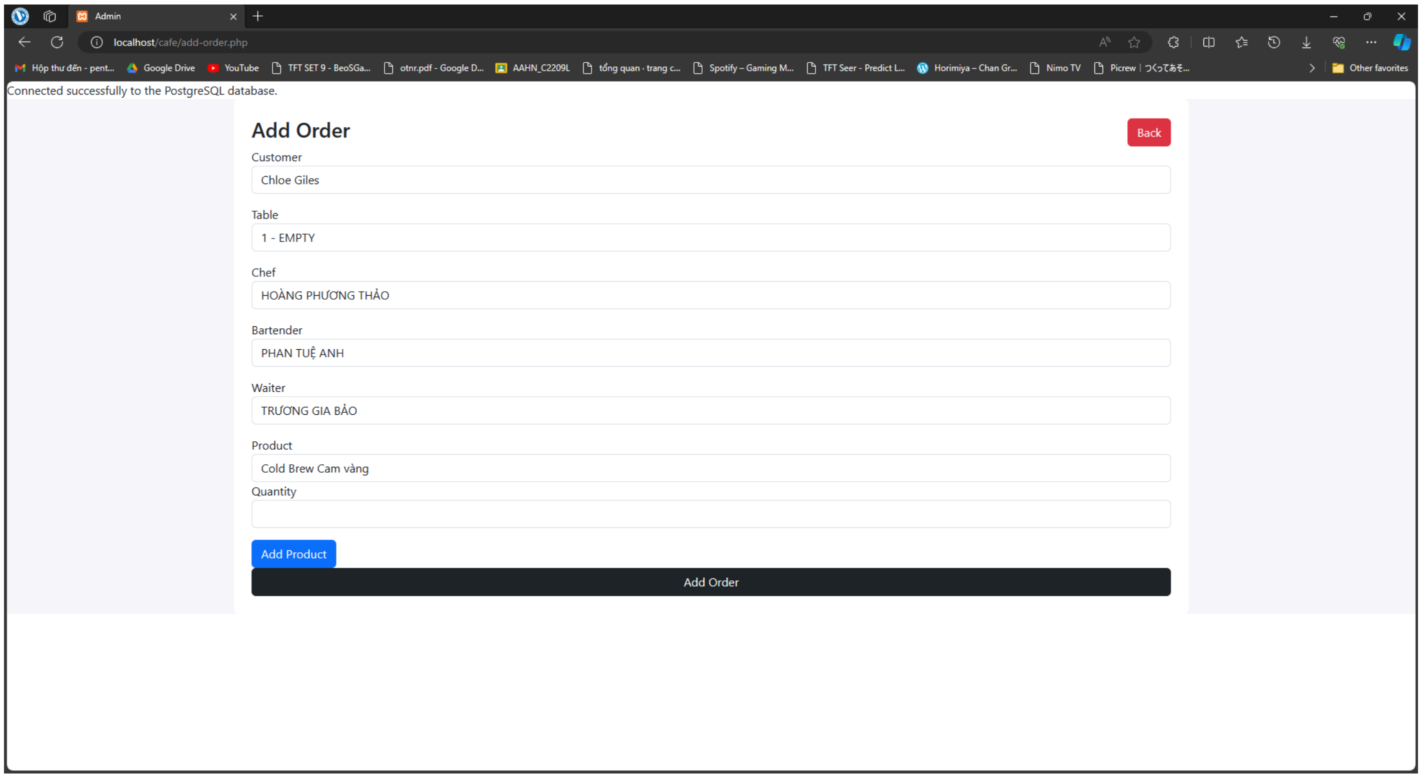
Orders List

Add Order

Order ID	Customer ID	Table ID	Bartender ID	Chef ID	Waiter ID	Status	Total Price	Order Date
13	57	68	7	47	40	CANCEL	0	2023-06-24 12:30:00
200	56	24	15	50	28	DONE	0	2023-07-28 06:48:00
255	91	74	4	49	38	DONE	0	2022-10-26 08:30:00
261	72	9	8	50	30	DONE	0	2022-10-31 09:33:00
262	76	7	13	49	25	DONE	0	2022-11-01 09:33:00
263	66	20	19	47	42	DONE	0	2022-11-02 09:33:00
271	8	22	1	47	32	DONE	0	2022-10-27 00:18:00
275	97	72	5	49	27	DONE	0	2022-10-27 04:30:00
278	39	74	3	46	35	DONE	0	2022-10-27 07:39:00
279	29	45	13	49	30	DONE	0	2022-10-27 08:42:00
286	3	62	9	46	25	DONE	0	2022-11-03 08:42:00
287	7	11	17	48	36	DONE	0	2022-11-04 08:42:00
289	32	75	4	49	22	DONE	0	2022-11-06 08:42:00
293	47	30	3	48	38	DONE	0	2022-11-10 08:42:00
297	80	75	16	47	22	DONE	0	2022-10-28 02:36:00
322	90	47	15	46	30	DONE	0	2022-01-25 08:04:00
327	18	41	20	48	36	DONE	0	2022-01-25 15:43:00

On the Order page, you can click the button Add Order to add a new order to the list.

Add Order page demo:

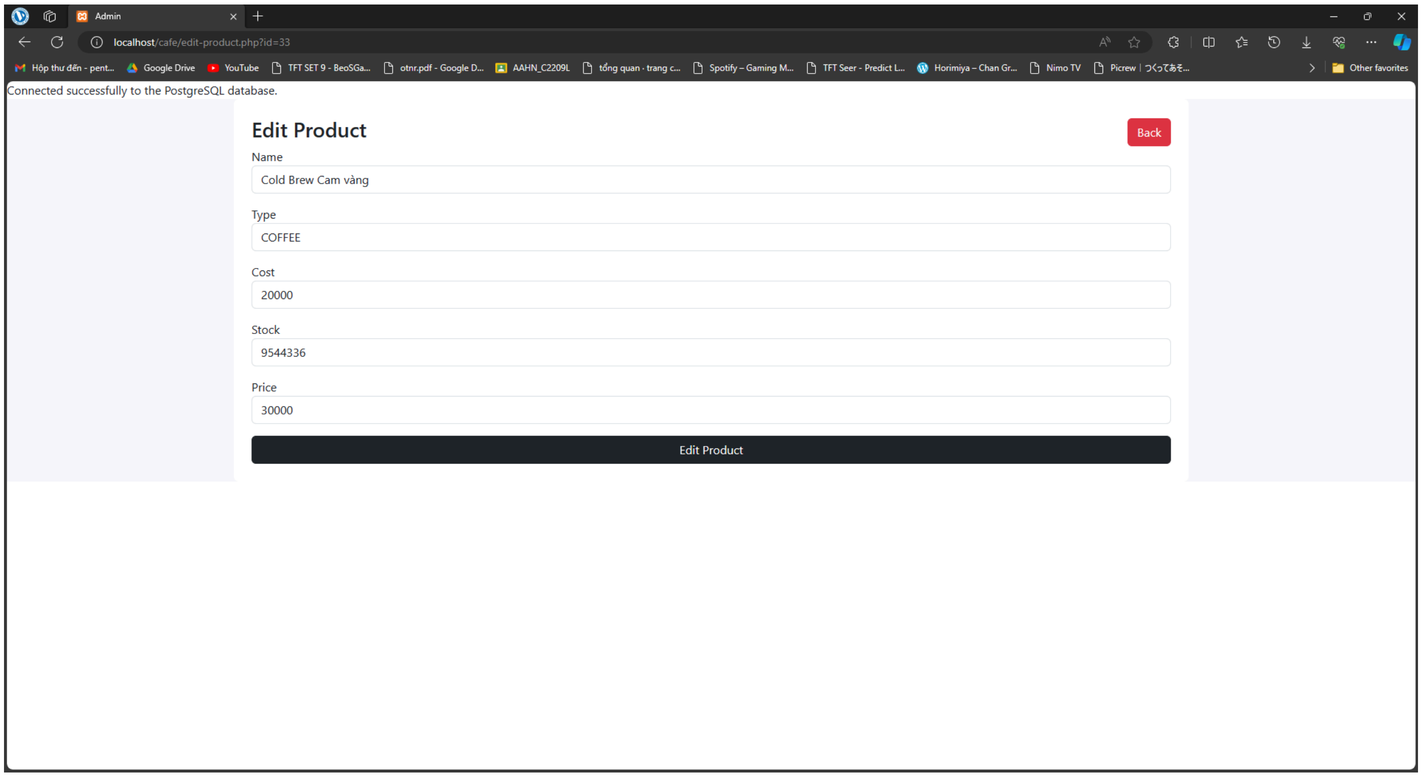


Connected successfully to the PostgreSQL database.

Add Order

Customer	Chloe Giles
Table	1 - EMPTY
Chef	HOÀNG PHƯƠNG THẢO
Bartender	PHAN TUỆ ANH
Waiter	TRƯƠNG GIA BẢO
Product	Cold Brew Cam vàng
Quantity	

In the Product List page, you can add, edit or delete products.



localhost/cafe/edit-product.php?id=33

Connected successfully to the PostgreSQL database.

Edit Product

Back

Name: Cold Brew Cam vàng

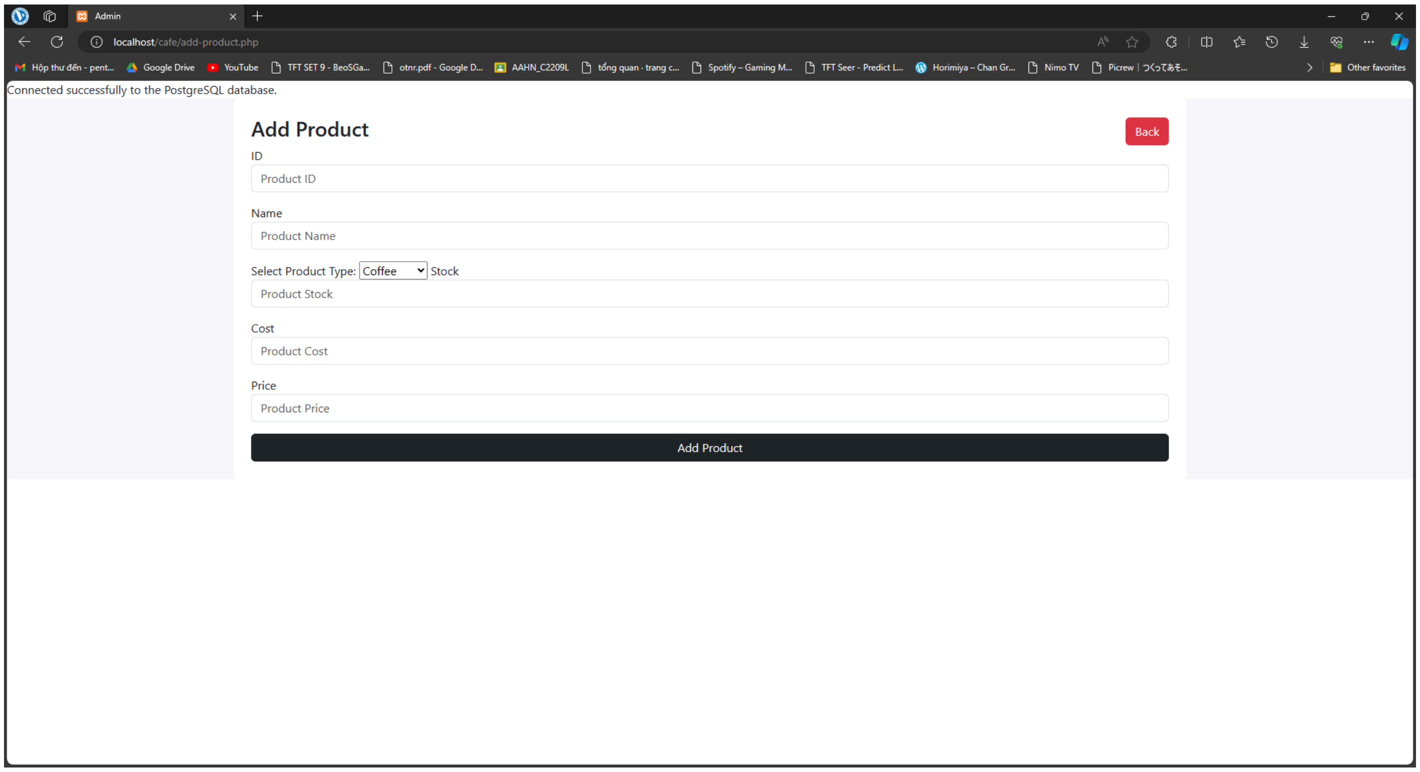
Type: COFFEE

Cost: 20000

Stock: 9544336

Price: 30000

Edit Product



localhost/cafe/add-product.php

Connected successfully to the PostgreSQL database.

Add Product

Back

ID: Product ID

Name: Product Name

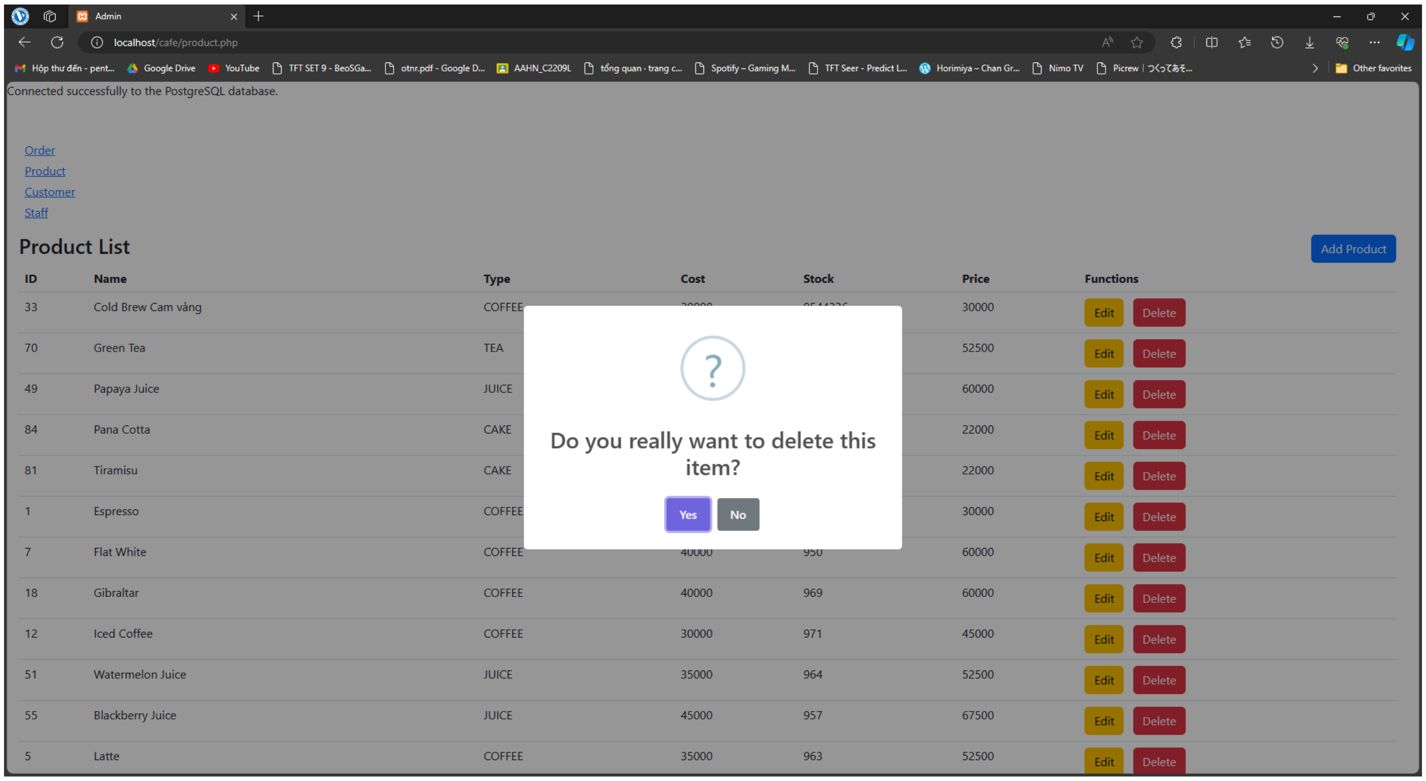
Select Product Type: Coffee Stock

Product Stock

Cost: Product Cost

Price: Product Price

Add Product



Connected successfully to the PostgreSQL database.

Order
Product
Customer
Staff

Product List

Add Product

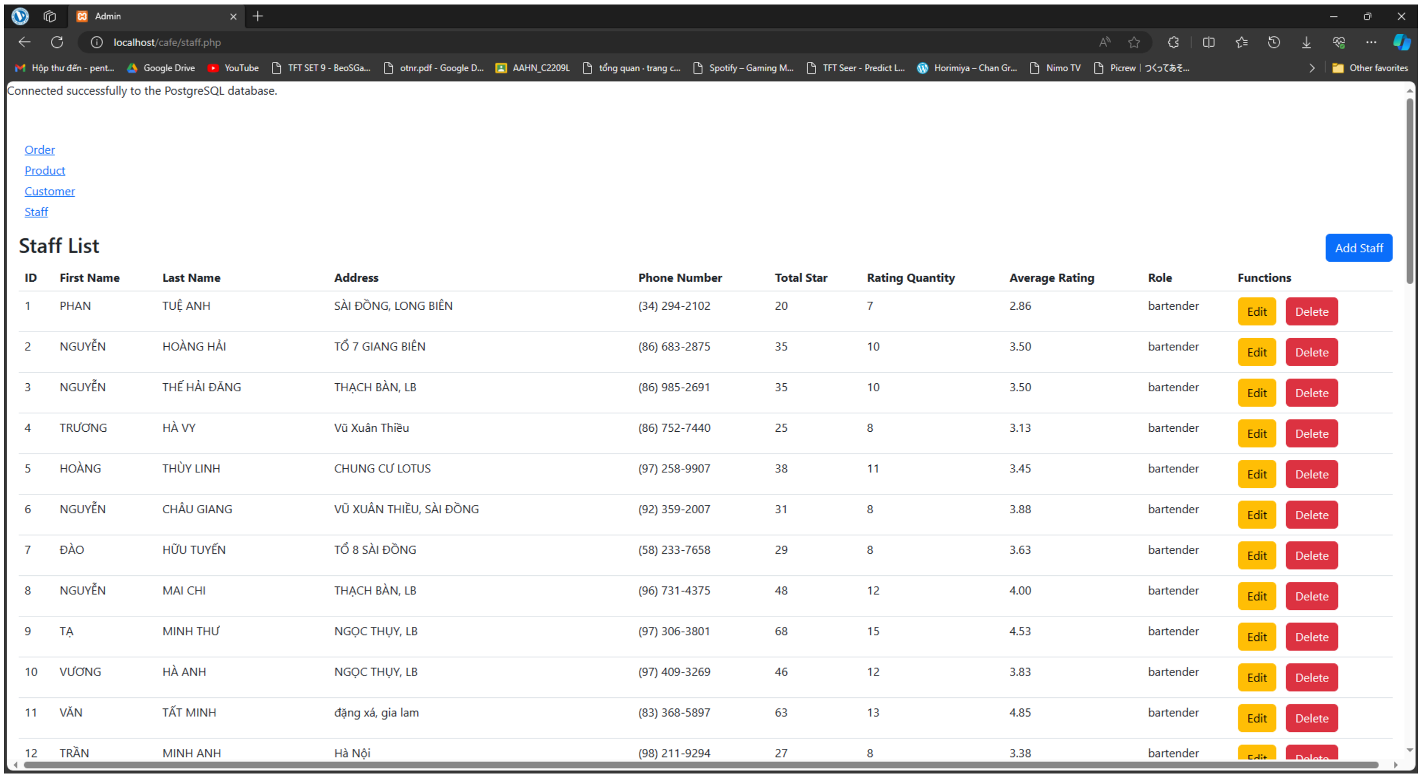
ID	Name	Type	Cost	Stock	Price	Functions
33	Cold Brew Cam vàng	COFFEE	[illegible]	[illegible]	30000	Edit Delete
70	Green Tea	TEA			52500	Edit Delete
49	Papaya Juice	JUICE			60000	Edit Delete
84	Pana Cotta	CAKE			22000	Edit Delete
81	Tiramisu	CAKE			22000	Edit Delete
1	Espresso	COFFEE			30000	Edit Delete
7	Flat White	COFFEE	40000	950	60000	Edit Delete
18	Gibraltar	COFFEE	40000	969	60000	Edit Delete
12	Iced Coffee	COFFEE	30000	971	45000	Edit Delete
51	Watermelon Juice	JUICE	35000	964	52500	Edit Delete
55	Blackberry Juice	JUICE	45000	957	67500	Edit Delete
5	Latte	COFFEE	35000	963	52500	Edit Delete

Do you really want to delete this item?

Yes No

Add, edit, and Delete Product demo.

We also have Staff list and Customer list with information shown on the screen



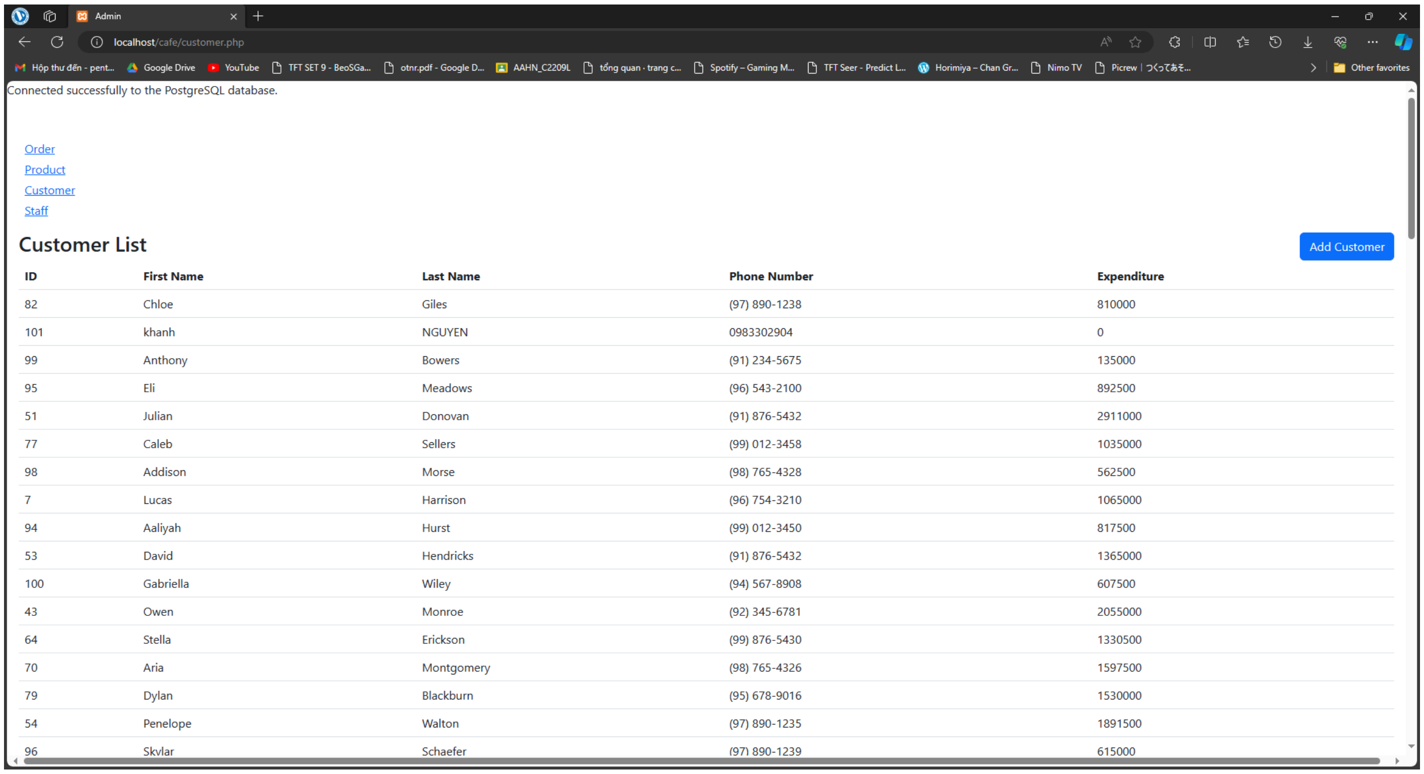
Connected successfully to the PostgreSQL database.

Order
Product
Customer
Staff

Staff List

Add Staff

ID	First Name	Last Name	Address	Phone Number	Total Star	Rating Quantity	Average Rating	Role	Functions
1	PHAN	TUỆ ANH	SÀI ĐỒNG, LONG BIÊN	(34) 294-2102	20	7	2.86	bartender	Edit Delete
2	NGUYỄN	HOÀNG HẢI	TỔ 7 GIANG BIÊN	(86) 683-2875	35	10	3.50	bartender	Edit Delete
3	NGUYỄN	THẾ HẢI ĐĂNG	THẠCH BÀN, LB	(86) 985-2691	35	10	3.50	bartender	Edit Delete
4	TRƯƠNG	HÀ VY	Vũ Xuân Thiều	(86) 752-7440	25	8	3.13	bartender	Edit Delete
5	HOÀNG	THÙY LINH	CHUNG CƯ LOTUS	(97) 258-9907	38	11	3.45	bartender	Edit Delete
6	NGUYỄN	CHÂU GIANG	VŨ XUÂN THIỀU, SÀI ĐỒNG	(92) 359-2007	31	8	3.88	bartender	Edit Delete
7	ĐÀO	HỮU TUYẾN	TỔ 8 SÀI ĐỒNG	(58) 233-7658	29	8	3.63	bartender	Edit Delete
8	NGUYỄN	MAI CHI	THẠCH BÀN, LB	(96) 731-4375	48	12	4.00	bartender	Edit Delete
9	TẠ	MINH THƯ	NGỌC THỤY, LB	(97) 306-3801	68	15	4.53	bartender	Edit Delete
10	VƯƠNG	HÀ ANH	NGỌC THỤY, LB	(97) 409-3269	46	12	3.83	bartender	Edit Delete
11	VĂN	TẤT MINH	đặng xá, gia lam	(83) 368-5897	63	13	4.85	bartender	Edit Delete
12	TRẦN	MINH ANH	Hà Nội	(98) 211-9294	27	8	3.38	bartender	



Connected successfully to the PostgreSQL database.

Order
Product
Customer
Staff

Customer List

Add Customer

ID	First Name	Last Name	Phone Number	Expenditure
82	Chloe	Giles	(97) 890-1238	810000
101	khanh	NGUYEN	0983302904	0
99	Anthony	Bowers	(91) 234-5675	135000
95	Eli	Meadows	(96) 543-2100	892500
51	Julian	Donovan	(91) 876-5432	2911000
77	Caleb	Sellers	(99) 012-3458	1035000
98	Addison	Morse	(98) 765-4328	562500
7	Lucas	Harrison	(96) 754-3210	1065000
94	Aaliyah	Hurst	(99) 012-3450	817500
53	David	Hendricks	(91) 876-5432	1365000
100	Gabriella	Wiley	(94) 567-8908	607500
43	Owen	Monroe	(92) 345-6781	2055000
64	Stella	Erickson	(99) 876-5430	1330500
70	Aria	Montgomery	(98) 765-4326	1597500
79	Dylan	Blackburn	(95) 678-9016	1530000
54	Penelope	Walton	(97) 890-1235	1891500
96	Skylar	Schaefer	(97) 890-1239	615000

User Interface self-comment:

→ Positive points: UI connected with database smoothly, can read - write - edit data from the UI to the database.

→ Negative points: lack of css decoration, not enough requirements and sometimes appear some errors.

Conclusion

Difficulties during Project Execution

1. With Teamwork
    - Not having much time to discuss offline.
    - Different timetables.
    - Hard to express ideas by using natural language.
    - Merge different parts done by each member into one complete project.
2. With Database Project
    - Many problems occur when designing ERD and Relational Schema since they need to have many applications in reality.
    - Collect sample data appropriately.
    - Merge conflicts and inconsistency between components built by different members.
    - Not have enough knowledge to make a complete GUI

Lesson Learned

- Think of requirements before doing something.
- Start doing project earlier to avoid rushed situation.
- Spend more time to design and check the ERD to avoid unnecessary changes during the coding process.
- Concern on performance and apply the lesson taught in class to the project.
- Important steps to develop a project