

CNG 352 PRACTICAL ASSIGNMENT

Step 5: User Interface

TEAM:

Omar R.A. Ammar 2456010
Mohamad Said Al Kudaimi 2414571
Wuda F.K. Abbasi 2414555

Abstract

Our proposed application for this semester's term project is a shop delivery application database, the shop delivery application is called '**Tarteeb**'. We would like the customer to have an easy way of viewing what they want to order from the restaurants, supermarkets, and pharmacies of their choice. Our application will have a contract with several shops and cafes. The features provided will be as follows; we'll have riders with contacts assigned with specific orders to certain addresses and restaurant, the company will provide vehicles to its deliveries that can be either a car, a cycle, or a motorcycle. The customer will have to make an account by providing their information. In restaurant ordering you can view the most ordered dish. They can also use the option where the application will choose a meal to order for them when a user isn't sure about what to eat.

Data Requirement Analysis

Users

Users of this application will be able to register using their names, surnames, email address and password to login. Users can register multiple addresses and multiple credit card information for different payment methods. This system can store points for each user that they achieve from each order, these points represent the discount they can get on future orders. The points can be earned by users when buying meals. As if the meal was for 40 TL then the user would earn 40 points. Each 10 points can be used as 1 TL discount on their next purchase on the application. Users will also have a profile type; there will be a standard account, a student account, or a premium account. An account can be considered as a student and still be upgraded to premium. A certain fee would be paid by users to be able to upgrade their accounts to premium accounts. With a monthly subscription.

Shops

Shops can register to the platform by supplying their name, location, manager number and a certain contract fee to be on the platform. The shops can get reviews from users using the application. The shop can be a type of supermarket for groceries, a restaurant. These shops are where the users will order from, groceries, food using this app, Tarteeb can deliver from any of these places. restaurants can have the option to specify the cuisine of the restaurant.

Menu

Each shop has an assigned menu that includes all the products the shop offers. The menu has a unique ID and a brief description about the products (i.e., ingredients of the food, brand of the grocery item). Two shops can share the same menu.

Product

Each product of the menu is assigned a unique name combined with the menu it is included in. it also has a price, a description, and reviews.

Riders

Our platform will have rides of its own as well as part time riders. The riders can sign up with their name, surname, email, password, and their vehicle number. Each rider will have a unique ID assigned. The riders can be working full-time, or part time. The platform's own riders that will be working full-time will have a base salary as well as a bonus salary based on the number of deliveries. The riders working part-time will get their salaries and they are able to earn bonuses based on the number of deliveries.

Orders

The platform will have different kinds of orders for different kinds of shops, for example restaurants can have reservations or pick-ups, while pharmacies and grocery stores have access to only delivery method. Each order will be tied with a unique order Id, the total price, the points ordered from this delivery. It will also hold information of the user like the user's phone number and location. As well as restaurant information like the restaurant name and location. Every order delivered will be assigned to a driver By holding his id.

Transaction Requirements (Sample)

Data entry

- Enter the details of a new user.
- Enter the details of a new shop.
- Enter the details of a new order (delivery / Pickup/ Reservation)
- Enter the details of a new rider (Full Time/ Part time)
- Enter the details of a new menu
- Enter the details of a new product on a certain menu

Data update/deletion

- Update / Delete the details of a user
- Update / Delete the details of a shop
- Update / Delete the details of a rider
- Update / Delete the details of a menu
- Update / Delete the details of a product on a certain menu

Data queries

Example of queries a user might expect:

- (a) List the details of all the users who are (students/ Premium / standard) in the database.
- (b) List the details of all the restaurants registered on the platform.
- (c) List the most popular order on the platform
- (d) List the details of all the (part-time/ full-time) riders.
- (e) List all the details of orders delivered by a certain driver.
- (f) List the most popular meal ordered in a certain restaurant.
- (g) List all the products in a certain restaurant menu.
- (h) List the rider with the most deliveries in a certain week.
- (i) List all the orders given by a certain shop.
- (j) List how many orders a restaurant prepares in a month
- (k) Find the number of deliveries by for every part time driver in a month
- (l) Find restaurants with a certain cuisine in a certain area
- (m) List the users who earned more than 1000 points.
- (n) Find the most popular meal in a certain area
- (o) Find the average spending in a certain area
- (p) Find the most popular restaurant in a certain area
- (q) Find the grocery store with best reviews
- (r) Find the driver with the best reviews.
- (s) Calculate the average profit for the platform during a month

Rider

Rider is an entity to store details of the delivery persons, who will deliver goods to users at specific addresses. RiderId will be uniquely used as the primary for this entity. There will be two types of drivers; full time company drivers that will have a fixed base salary and a bonus salary on the number of deliveries made, and a part time driver who will be paid based on the number of deliveries made. So these two attributes will have a disjoint total relation with the Rider entity.

Menu

We have an entity for menu with the menuId as the primary key. This entity is to store details of all the products of the shop, like meals from a restaurant, products from a grocery store. Each menu will be unique to its shop but menus can be shared between the same restaurants or pharmacies in different locations.

Order

Order is an entity with a unique orderId as its primary key, this entity will store all details about the order, where it is coming from and where it is going to, the address of the shop and the address of the delivery point. I will store all details about the product being delivered as well as the name of the shop it is being delivered from. Each order also contains a number of points based on the price of the delivery that will be used to give discounts to the customer if enough point saree collected. An order also has a disjoint total relation with three entities. First entity is the Order_reservation which will be used for restaurants, when customers want to reserve a table, the second entity is Order_delivery which will be a rider picking up the order from the shop and delivering it to the address of the customer, the third entity is Order_pickUp which will be for customers who want to place an order, the app will give the customer an estimated time that will take for the product or meal to get ready, once the good is ready, the customer will come themselves to pick up the order from the location of the shop.

Product

Product entity stores the productId as the primary key, containing price and description of the product in the menu of a given shop. This is so the customer can see each product (i.e. fried chicken from a restaurant, or Vitamin D capsules from a grocery store) on the menu and decide easily what to order.

Relations:

User Entity:

The User entity has a Review relation with the Shop entity, each users will be allowed to rate the shop and give a review in terms of how accurate the products were to the given description or if the products were faulty, etc. A user can have many reviews for a shop and a certain shop can be reviewed by many users so the relation is many to many.

This entity also has a Review relation with the Product entity, with the same logic. User can review each product offered by a shop. A user can review multiple products and a product can be reviewed by multiple users.

The User entity has a Review relation with the Rider entity for reviewing the riders' behavior or not meeting the given estimated time of delivery, etc.

This entity has a Place relation with the Order entity as a user will be allowed to place an order to be delivered from a shop. A single user can place many orders but a certain order will like to a single user (two users can have the same order i.e. fried chicken, but each order will be unique to a user, so one order will link to only that user).

User entity has a Pick relationship with Order_pickUp entity in which a user is going to pick up the order from the shop. One user can pick up many orders from many locations but that specific order will link back to a single customer, so the relationship is one to many.

Shop:

Shop entity has a relationship with Menu entity as each shop will have a menu which will list all the products, etc. Many shop can have the same menu for example the same restaurant in different locations can have the same order (i.e. kfc will have the same menu in Lefkosa and Girne). But one shop will only have one menu.

The sub categories of shops; Restaurant and Grocery will each have a Has relation with Order entity. Restaurant will have a one-to-many relation with Order entity, in which a user can reserve order in that restaurant or pick up the order or get it delivered to their address. Grocery entity will have a one-to-many relation with Order_delivery to get their products delivered.

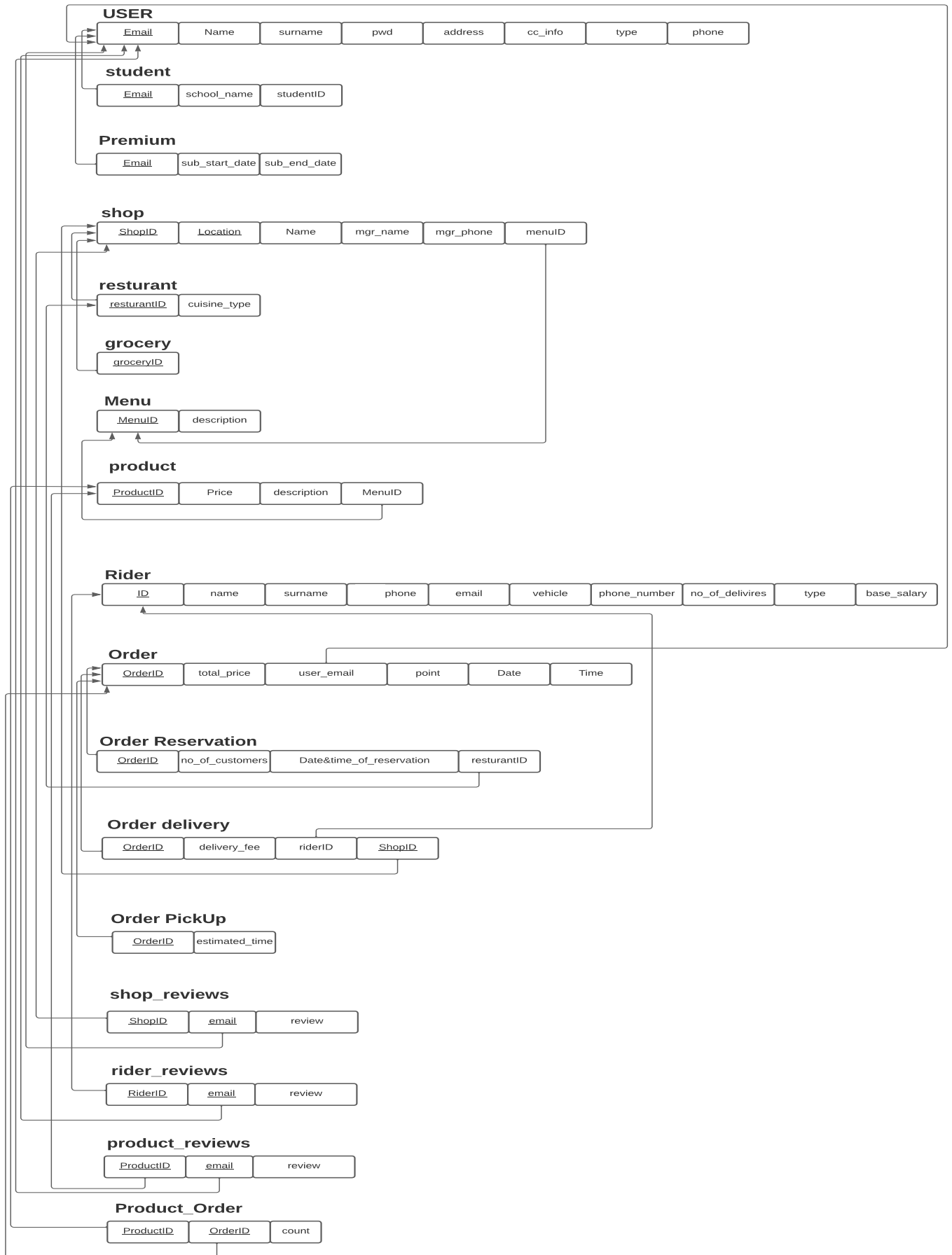
Order:

Order entity has a relationship with product as an order will contain a product. This relation is many-to-many as a single order can have many orders and a single product can be in many different order.

Rider:

The Rider entity had a Takes relation with Order_delivery entity as a rider will take the order and deliver it, one rider can have many orders to deliver at different places but a single order will use only one rider to deliver it (i.e. two riders will not deliver the same order to the same place, etc).

EER MAPPING



Functional Dependencies

The following functional dependencies are taken from our assumptions about the data as well as the EER Mapping above, for every table in the EER mapping its functional dependencies are written below it. We will use these functional dependencies as a guide while normalizing the data base.

USER:

FD1 : {Email} -> {name, surname, pwd, address, cc_info, type, phone}

Student:

FD3 : {Email} -> {School_name, StudentID}

Premium:

FD4: {Email} -> {sub_start_date, sub_end_date}

Shop:

FD5: {ShopID, Location} -> {Name, mgr_name, mgr_phone, menuID}

FD6: {mgr_phone} -> {mgr_name}

Restaurant:

FD7: {ResturantID} -> {cuisine_type}

Menu:

FD8: {MenuID} -> {Description}

Product

FD9: {ProductID} -> {Price, Description, menuID}

Order

FD10: {OrderID} -> {date, Time, Total_price, Points}

Reservation

FD11: {OrderID} -> { NO_of_customer, Date&time, ResturantID}

Pick_up

FD12: {OrderId} -> {EST, RestaurantId}

Delivery

FD13: {OrderId} -> {Rider Id, delivery_fee, grocerry_id, restaurant_id}

Shop_reviews

FD14: {ShopId, user_email} -> {review}

Rider_reviews

FD15: {RiderID, user_email} -> {review}

Product_reviews

FD16: {ProductID, user_email} -> {review}

Product Order

FD17: {ProductID, OrderID} -> {count}

Rider

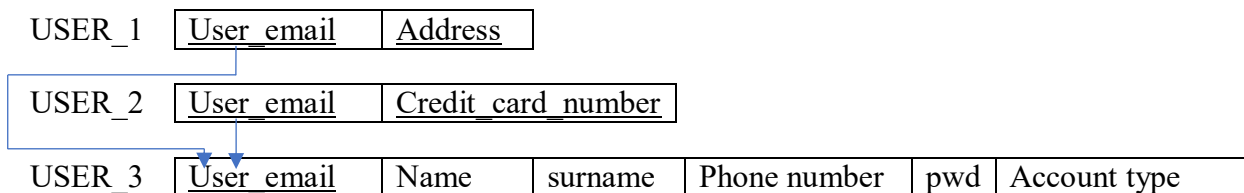
FD18 : {RiderID} -> {Name, surname, email, vehicle, phone, no of deliveries, type, base_salary}

FD19: {email} -> {phone number, name , surname}

Normalization

1NF

The only relation that breaks the first normal form is **USER** since the user table include values that could be repeated like the addresses and credit card information as each user is allowed to store multiple credit card information and addresses in the data base. Thus we have to normalize the **USER** table.



There is no other relations / functional dependencies which violates the 1NF rule. Thus we move to 2NF.

2NF

The first step in normalizing to the second normal form is picking the super key of every relation.

For user -> **email**

For premuim -> **email**

For student -> **email**

For rider -> **riderID**

For shop -> **ShopID, Location**

For Restaurant -> **ResturantID**

For Order -> **OrderID**

For Reservation -> **OrderID**

For Pick_up -> **OrderID**

For Delivery -> **OrderID**

Product -> **ProductID**

Menu -> **MenuID**

Order_Product -> OrderID ,ProductID

As we can see from the RIDER relation the email could give us the name surname and phone number (FD19) . Thus, we must normalize the relation.

RIDER1	<u>email</u>	Vehicle	No_of_deliveries	type	Base_salary
RIDER2	<u>email</u>	Name	surname	Phone	

There is no other relations that violates 2NF.

3NF

In the shop relation FD6 violates the 3NF Thus we have to normalize.

SHOP1	<u>shopID</u>	Location	MenuID	Mgr_phone
SHOP2	<u>Mgr_phone</u>	Mgr_name		

BCNF

Since BCNF is related to prime attributes. It can be observed that there is no prime attributes in the database thus , our database is normalized to the 4NF or BCNF

Final notes on normalization

Above you can see the finalized version (BCNF normalized version of our database schema) . we can see that no relations could be combined without introducing redundancy to our database. Thus, no relations will be combined.

Final relations table

USER

Email	Name	surname	pwd	type	phone
-------	------	---------	-----	------	-------

USER_Addresses

Email [FK:USER_EMAIL]	address
-----------------------	---------

USER_CC_Info

Email [FK:USER_EMAIL]	cc_info
-----------------------	---------

student

Email [FK:USER_EMAIL]	school_name	studentID
-----------------------	-------------	-----------

Premium

Email [FK:USER_EMAIL]	sub_start_date	sub_end_date
-----------------------	----------------	--------------

shop

ShopID	Location	Name	menuID [FK: Menu_MenuID]	mgr_phone [FK: Manager_data_MgrPhone]
--------	----------	------	--------------------------	---------------------------------------

resturant

restaurantID [FK: Shop_ShopID]	cuisine_type
--------------------------------	--------------

grocery

groceryID [FK: Shop_ShopID]

Menu

MenuID	description
--------	-------------

Manager_Data

mgr_phone	mgr_name
-----------	----------

product

ProductID	Price	description	MenuID
-----------	-------	-------------	--------

Rider_Data

email	name	surname	phone_number	vehicle
-------	------	---------	--------------	---------

Rider

ID	email [FK:Rider_data_email]	no_of_delivires	type	base_salary
----	-----------------------------	-----------------	------	-------------

Order

OrderID	total_price	user_email [FK: USER_email]	point	Date	Time
---------	-------------	-----------------------------	-------	------	------

Order Reservation

OrderID [FK:Order_orderId]	no_of_customers	Date&time_of_reservation	restaurantID [FK: Restaurant_Restaurant_id]
----------------------------	-----------------	--------------------------	---

Order delivery

OrderID [FK:Order_orderId]	delivery_fee	riderID [FK:RIDER_RIDERID]	ShopID [FK: SHOP_ShopID]
----------------------------	--------------	----------------------------	--------------------------

Order Pickup

OrderID [FK:Order_orderId]	estimated_time
----------------------------	----------------

shop_reviews

ShopID [FK:Shop_ShopID]	email [FK: USER_Email]	review
-------------------------	------------------------	--------

rider_reviews

RiderID [FK:Rider_RiderID]	email [FK: USER_Email]	review
----------------------------	------------------------	--------

product_reviews

ProductID [FK:Product_ProductID]	email [FK: USER_Email]	review
----------------------------------	------------------------	--------

Product_Order

ProductID [FK:Product_ProductID]	OrderID [FK: Order_OrderID]	count
----------------------------------	-----------------------------	-------

Database Physical Design and Tunning

In the attached code, you can see the database dump for initializing and inserting dummy data into our data base to be able to test some of our queries. After the physical design of the database was done. We considered adding indexes for some of our data like the type of the account in USERS table as well as the type of the rider in RIDERS table.

Index Creation

```
-- Index: USER_TYPE_INDEX

-- DROP INDEX IF EXISTS public."USER_TYPE_INDEX";

CREATE INDEX IF NOT EXISTS "USER_TYPE_INDEX"
ON public."USER" USING btree
("accountType" COLLATE pg_catalog."default" varchar_ops ASC NULLS LAST)
INCLUDE("accountType")
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."USER"
CLUSTER ON "USER_TYPE_INDEX";
```

Riders_Type:

```
-- Index: RIDERS_TYPE

-- DROP INDEX IF EXISTS public."RIDERS_TYPE";

CREATE INDEX IF NOT EXISTS "RIDERS_TYPE"
ON public."RIDERS" USING btree
(type COLLATE pg_catalog."POSIX" bpchar_pattern_ops ASC NULLS LAST)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."RIDERS"
CLUSTER ON "RIDERS_TYPE";
```

Workload of the database

We expect our database to be able to handle at least a thousand concurrent users at one time in the primary stage of the project. However, in later stages it might expand to support a couple thousand users at the same time. The Order table will be the most table with accesses and additions as we expect orders to be added all the time. We don't expect the riders or the restaurants tables to be modified that often.

Frequency of Queries and Updates:

We expect our database to hold its durability and integrity and withstand high-workload , as orders could be made every 10 s or even less which will redirect updates to at least 3 relations at a single time. We approximate the number of tuples in high-load relations like orders to reach 10K or more while on smaller relations like Riders up to 100 max. of course we should always consider the expansion of the database.

User views Design & Security

We also added a couple of user views for users of the system. For example, business owners, can have access to their deliveries as well as the riders that worked for them before. End-Users will only have access to the Shops and their menus. while HR would have access to the rider's data as well as all the shops and restaurants on the platform currently.

Example Scripts for Insert , Delete and Update

Delete :

```
DELETE FROM public."SHOP"  
WHERE <condition>;
```

```
DELETE FROM public."USER"  
WHERE <condition>;
```

```
DELETE FROM public."PRODUCT_ORDER"  
WHERE <condition>;
```

Update :

```
UPDATE public."PRODUCT"  
SET "productID"=?, price=?, description=?, "menuID"=?  
WHERE <condition>;
```

```
UPDATE public."RIDERS"  
SET "ID"=?, email=?, type=?, base_salary=?  
WHERE <condition>;
```

Insert:

```
INSERT INTO public."USER"(  
    email, uname, surname, pwd, "accountType", phone)  
VALUES (?, ?, ?, ?, ?, ?);
```

```
INSERT INTO public."RIDER_DATA"(  
    email, name, surname, phone_number, vehicle)  
VALUES (?, ?, ?, ?, ?);
```

Queries

List the most popular order on the platform:

```
SELECT
    O."productID",
    description
FROM
    "PRODUCT_ORDER" O
INNER JOIN "PRODUCT" P
    ON P."productID" = O."productID"
GROUP BY O."productID",description
HAVING COUNT(O."productID") = (SELECT MAX("countitems") FROM (
    (SELECT pd."productID",COUNT(*) AS COUNTitems
    FROM "PRODUCT_ORDER" pd
    GROUP BY(pd."productID"))) AS ItemsOrderCount);
```

postgres/postgres@CNG352

Query Editor Query History

```
1 SELECT
2     O."productID",
3     description
4 FROM
5     "PRODUCT_ORDER" O
6 INNER JOIN "PRODUCT" P
7     ON P."productID" = O."productID"
8 GROUP BY O."productID",description
9 HAVING COUNT(O."productID") = (SELECT MAX("countitems") FROM (
10     (SELECT pd."productID",COUNT(*) AS COUNTitems
11     FROM "PRODUCT_ORDER" pd
12     GROUP BY(pd."productID"))) AS ItemsOrderCount);
13
```

Data Output Explain Messages Notifications

	productID character varying (20)	description character varying (100)
1	PG10	Product 10

Find the number of deliveries by for every part time driver in a month

```
SELECT D."riderID", COUNT(O."orderID") AS OrdersDelivered
FROM "ORDER" O
INNER JOIN "ORDER_DELIVERY" D ON O."orderID" = D."orderID"
INNER JOIN "RIDERS" R ON R."ID" = D."riderID"
WHERE R."type" = 'PART' AND "date" >= '2022-10-01' and "date" <= '2022-11-01'
GROUP BY D."riderID";
```

Query Editor	Query History						
<pre> 1 SELECT D."riderID" , COUNT(O."orderID") AS OrdersDelivered 2 FROM "ORDER" O 3 INNER JOIN "ORDER_DELIVERY" D ON O."orderID" = D."orderID" 4 INNER JOIN "RIDERS" R ON R."ID" = D."riderID" 5 WHERE R."type" = 'PART' AND "date" >= '2022-10-01' and "date" <= '2022-11-01' 6 GROUP BY D."riderID"; 7 8 9 </pre>							
Data Output	Explain Messages Notifications						
<table> <tr> <th>riderID</th><th>ordersdelivered</th></tr> <tr> <td>integer</td><td>bigint</td></tr> <tr> <td>1</td><td>1</td></tr> </table>	riderID	ordersdelivered	integer	bigint	1	1	
riderID	ordersdelivered						
integer	bigint						
1	1						

List the users who earned more than 10 points.

```

SELECT U."uname" , U."surname" , SUM(O."points") AS POINTS
FROM public."ORDER" O
INNER JOIN
public."USER" U ON U."email" = O."user_email"
WHERE POINTS > 10
GROUP BY U."uname" ,U."surname",O."user_email";

```

Query Editor

Query History

```

1 SELECT U."uname" , U."surname" , SUM(O."points") AS POINTS
2 FROM public."ORDER" O
3 INNER JOIN
4 public."USER" U ON U."email" = O."user_email"
5 WHERE POINTS > 10
6 GROUP BY U."uname" ,U."surname",O."user_email";
7
8

```

Data Output

Explain

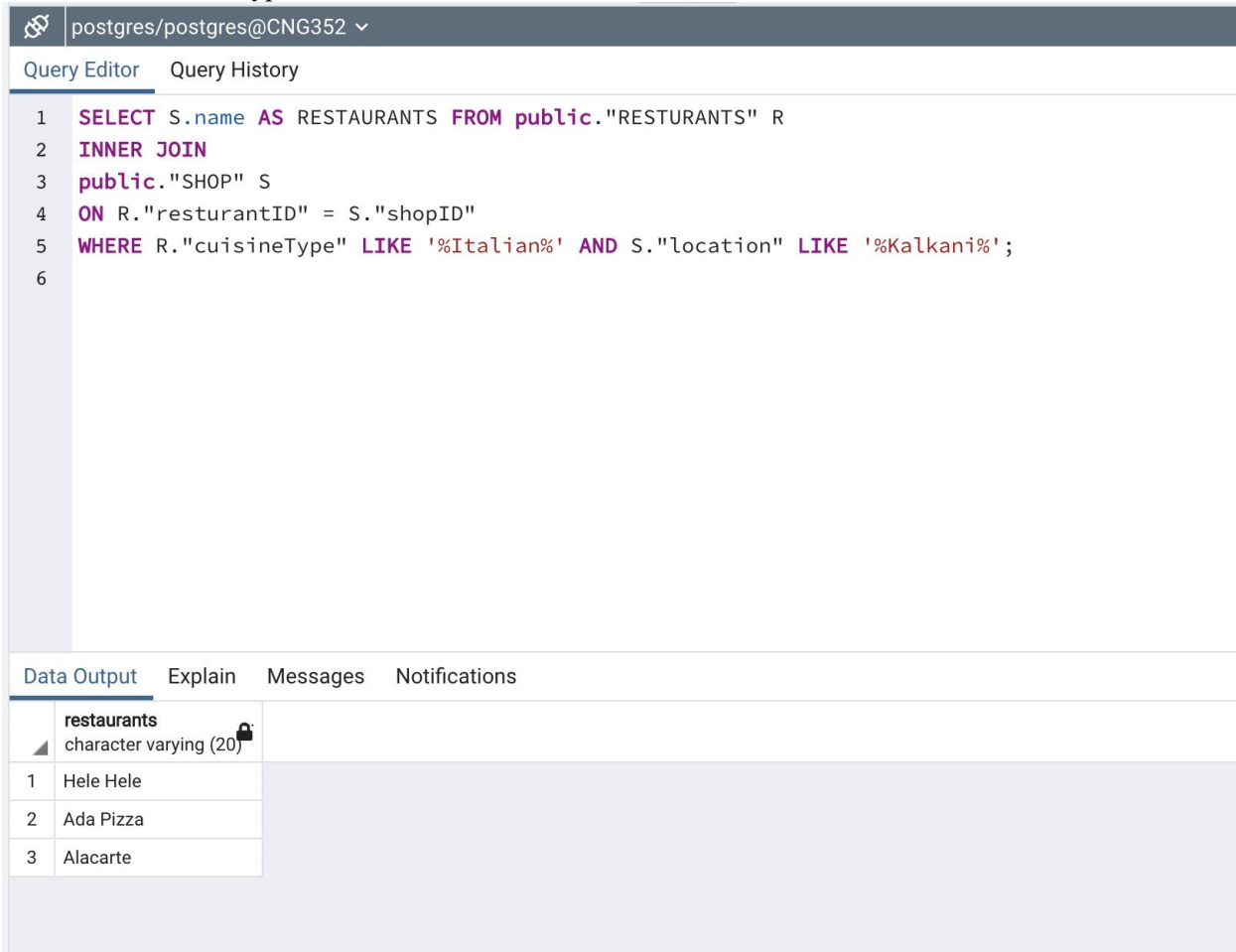
Messages

Notifications

	uname character varying (30)	surname character varying (30)	points bigint
1	Saeed	Alkudaimi	20
2	Omar	Ammar	12
3	Wuda	Abassi	15
4	James	Jones	35

Find restaurants with a certain cuisine in a certain area:

```
SELECT S.name AS RESTAURANTS FROM public."RESTURANTS" R
INNER JOIN
public."SHOP" S
ON R."resturantID" = S."shopID"
WHERE R."cuisineType" LIKE '%Italian%' AND S."location" LIKE '%Kalkani%';
```



The screenshot shows a PostgreSQL query editor interface. At the top, the database connection is set to 'postgres/postgres@CNG352'. Below this, there are tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab is active, displaying the SQL query from the previous block. Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing the results of the query. The results are displayed in a table with the following data:

	restaurants
1	Hele Hele
2	Ada Pizza
3	Alacarte

Calculate the average profit for the platform during a month:

```
SELECT (SUM(O."totalPrice") -SUM(R."base_salary") ) AS Profit
FROM public."ORDER" O
INNER JOIN public."ORDER_DELIVERY" D ON O."orderId" = D."orderId"
INNER JOIN public."RIDERS" R ON R."ID" = D."riderID"
WHERE "date" >= '2022-09-01' and "date" <= '2022-10-01';
```

Query Editor Query History

```
1 SELECT (SUM(O."totalPrice") -SUM(R."base_salary") ) AS Profit
2 FROM public."ORDER" O
3 INNER JOIN public."ORDER_DELIVERY" D ON O."orderId" = D."orderId"
4 INNER JOIN public."RIDERS" R ON R."ID" = D."riderID"
5 WHERE "date" >= '2022-09-01' and "date" <= '2022-10-01';
```

Data Output Explain Messages Notifications

	profit bigint
1	[null]

Graphical Application Interface

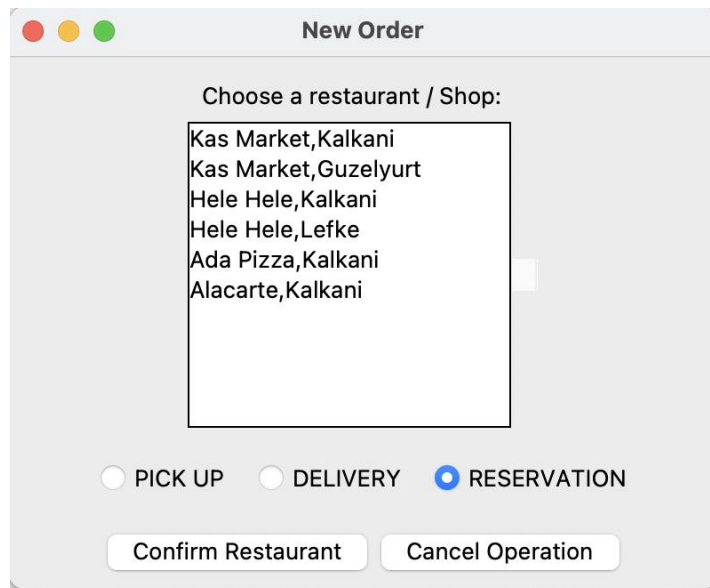
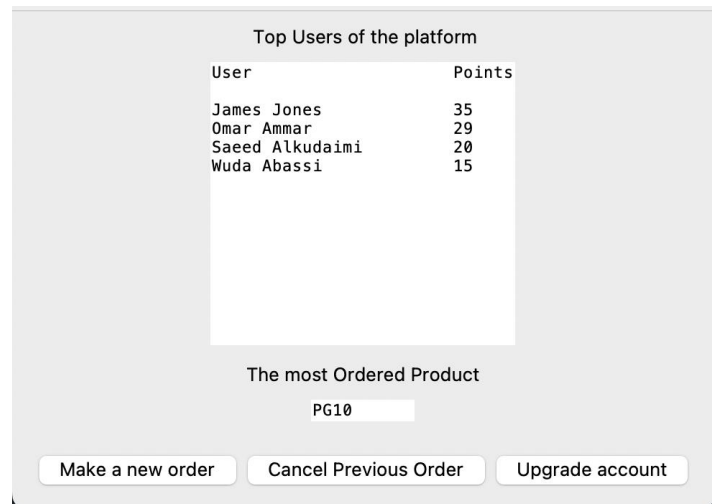
In the last step of our assignment we have created a sample user interface to represent the functionalities of our proposed application, in the following pages you can check out the interface and how is it supposed to be used;

Graphical User Interface

Once the user signs in , he will Be welcomed with this page which shows him the top 10 users of the platform and the most ordered product.

It will also give him the option to make a new order , cancel a previous order or upgrade his account.

As it can be observed there's a lot of improvement space for the user interface but due to tight timing this was built.



Once the user clicks on the new order, it shows him a list of all the shops/ restaurants on the platform alongside their locations.

It also gives him the choice to choose between Pick up , Delivery and Reservation.

Products:	Price	Description
<input type="checkbox"/> PG10	2	Product 10
<input checked="" type="checkbox"/> PG19	12	Product 9
<input type="checkbox"/> PG18	18	Product 8
<input type="checkbox"/> PG17	12	Product 7
<input checked="" type="checkbox"/> PG15	30	Product 5
<input type="checkbox"/> PG14	35	Product 4
<input checked="" type="checkbox"/> PG13	32	Product 3
<input type="checkbox"/> PG12	20	Product 2
<input type="checkbox"/> PG11	10	Juice

Confirm Items

After picking the restaurant and the method of receiving the order the user is presented with a list of all the products alongside their prices and description. Confirming the items would send the order to the restaurant in the delivery option. Otherwise it would take the user to a confirmation page to add extra data like the number of customers.

To cancel orders the user faces another login screen to confirm his login details, if his details are correct he will be forwarded to the next screen.

Cancel Order

Enter Email: user1@gmail.com

Enter Password:

Confirm Login Cancel Operation

Cancel Order

Choose a restaurant / Shop:

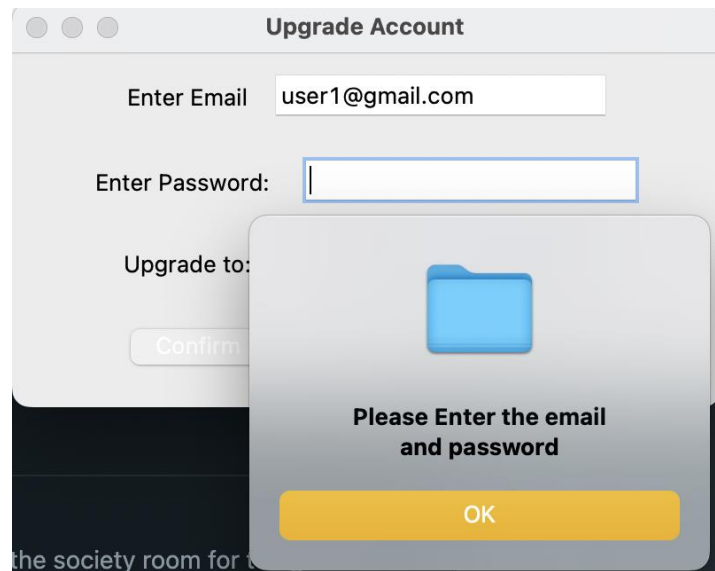
OrderID	Total Price	Points	Date	Time
1	100	10	2022-10-10	17:06:22
2	123	12	2022-12-02	05:49:22
3	75	7	2022-04-12	18:07:12

Cancel

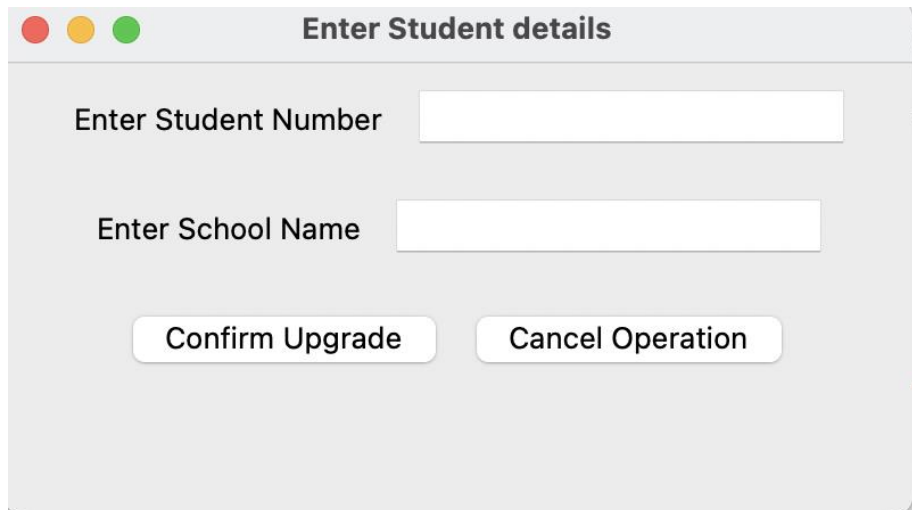
Cancelling the order window show the user a list of all of his previous orders with its details. He can select a certain order and cancel it, which would remove it from the list of orders.

if a user decides to upgrade his account he is faced with a window that asks him to confirm his login details. On wrong entry an error message would appear.

If the user decides to upgrade his account to a premium account no more information is required. On the other hand, upgrading to a student account asks more details of the user to complete his registration.



The screenshot shows a window titled "Upgrade Account". It contains two input fields: "Enter Email" with the text "user1@gmail.com" and "Enter Password:" with an empty field. Below these is a "Confirm" button. An error message dialog is overlaid on top of the window, featuring a blue folder icon and the text "Please Enter the email and password" with an "OK" button.

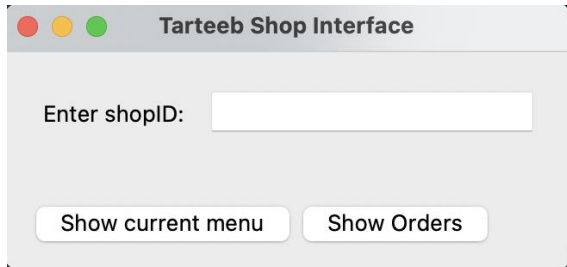


The screenshot shows a window titled "Enter Student details". It contains two input fields: "Enter Student Number" and "Enter School Name". Below these fields are two buttons: "Confirm Upgrade" and "Cancel Operation".

The next screen asks the user for the extra details to complete his account upgrade.

Shop Graphical Interface

We decided to provide shops with a specific system view that will allow them to access the database and modify their restaurants details or view their platform transactions.

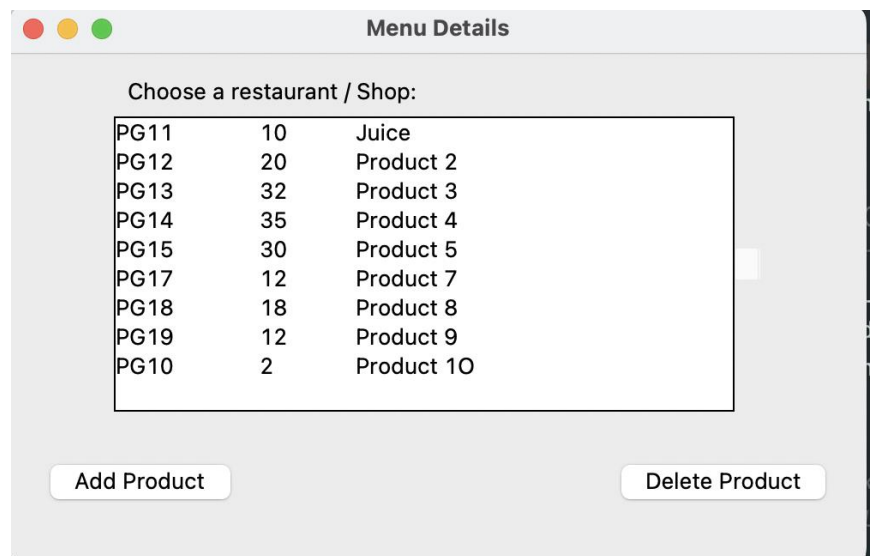


The screenshot shows a window titled "Tarteeb Shop Interface". It contains a text input field labeled "Enter shopID:". Below the input field are two buttons: "Show current menu" and "Show Orders".

On asking to show the current menu , a new prompt opens.

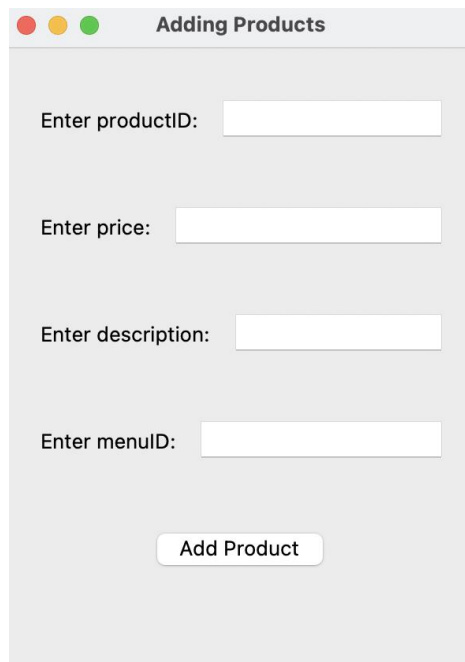
The new windows show a list of all the current product in the restaurant's menu.

If a product was selected then delete product button was pressed the product would be removed from the database.



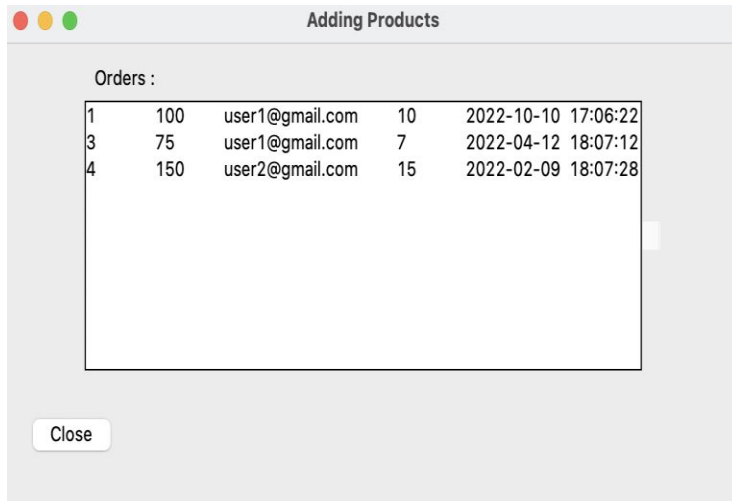
The screenshot shows a window titled "Menu Details". It contains a text input field labeled "Choose a restaurant / Shop:". Below the input field is a table with three columns: Product ID, Price, and Product Name. The table lists 10 products. Below the table are two buttons: "Add Product" and "Delete Product".

Product ID	Price	Product Name
PG11	10	Juice
PG12	20	Product 2
PG13	32	Product 3
PG14	35	Product 4
PG15	30	Product 5
PG17	12	Product 7
PG18	18	Product 8
PG19	12	Product 9
PG10	2	Product 10



The screenshot shows a window titled "Adding Products". It contains four text input fields: "Enter productID:", "Enter price:", "Enter description:", and "Enter menuID:". Below the input fields is a button labeled "Add Product".

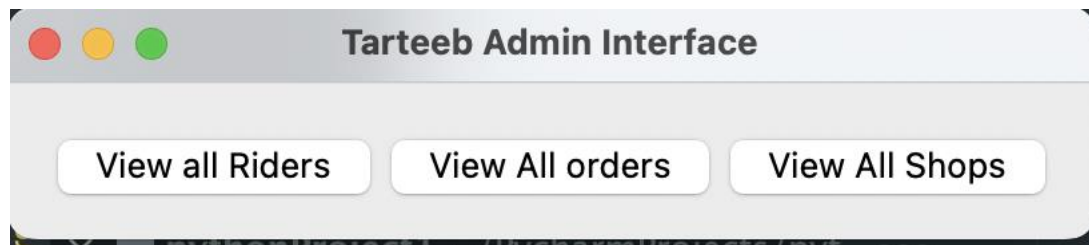
On clicking the add product button a new window opens where the shop manager can add the details of his new product then this product would be added to the database.



Show Product window shows all the orders in a certain restaurant with their details for the shop manager to view.

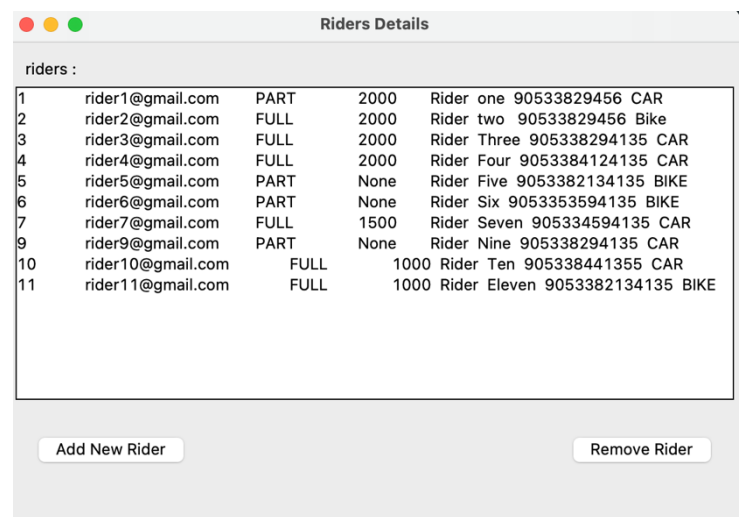
Admin Graphical Interface

We also decided to give the admin some authority over the platform. The admin interface below shows the functionalities of the admin.



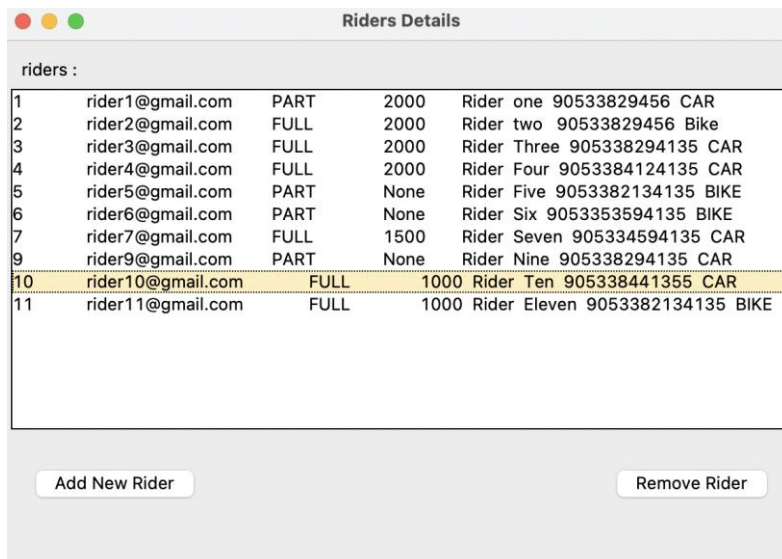
The admin interface allows the admin to either view all the rider, view all the orders and/or view all the shops.

Firstly viewing all the riders, the window Shows different options where we can either add a new rider or remove a current rider, it also shows the details of every ride including their phone number, their car and email.



The window shown allows the admin to enter the rider data to add a new rider.

A rider could be removed by selecting them and then clicking on remove rider button like shown bellow

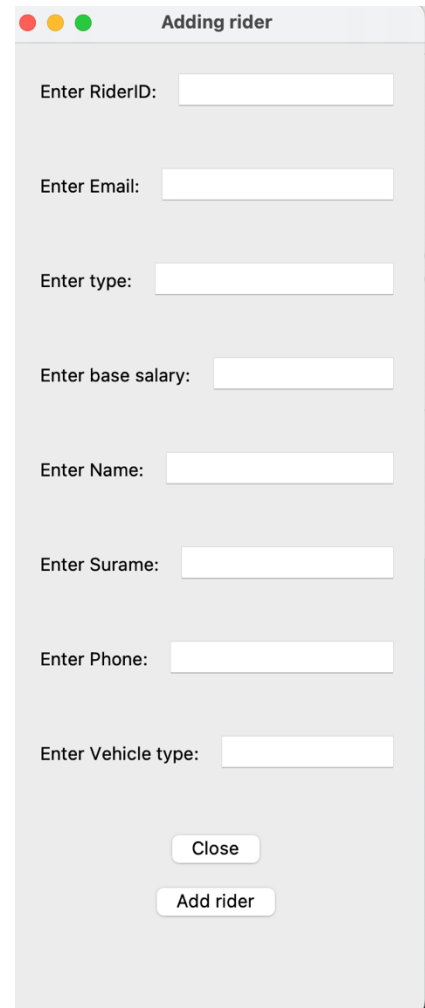


Riders Details

riders :

1	rider1@gmail.com	PART	2000	Rider one	90533829456	CAR
2	rider2@gmail.com	FULL	2000	Rider two	90533829456	BIKE
3	rider3@gmail.com	FULL	2000	Rider Three	905338294135	CAR
4	rider4@gmail.com	FULL	2000	Rider Four	9053384124135	CAR
5	rider5@gmail.com	PART	None	Rider Five	9053382134135	BIKE
6	rider6@gmail.com	PART	None	Rider Six	9053353594135	BIKE
7	rider7@gmail.com	FULL	1500	Rider Seven	905334594135	CAR
9	rider9@gmail.com	PART	None	Rider Nine	905338294135	CAR
10	rider10@gmail.com	FULL	1000	Rider Ten	905338441355	CAR
11	rider11@gmail.com	FULL	1000	Rider Eleven	9053382134135	BIKE

Add New Rider Remove Rider



Adding rider

Enter RiderID:

Enter Email:

Enter type:

Enter base salary:

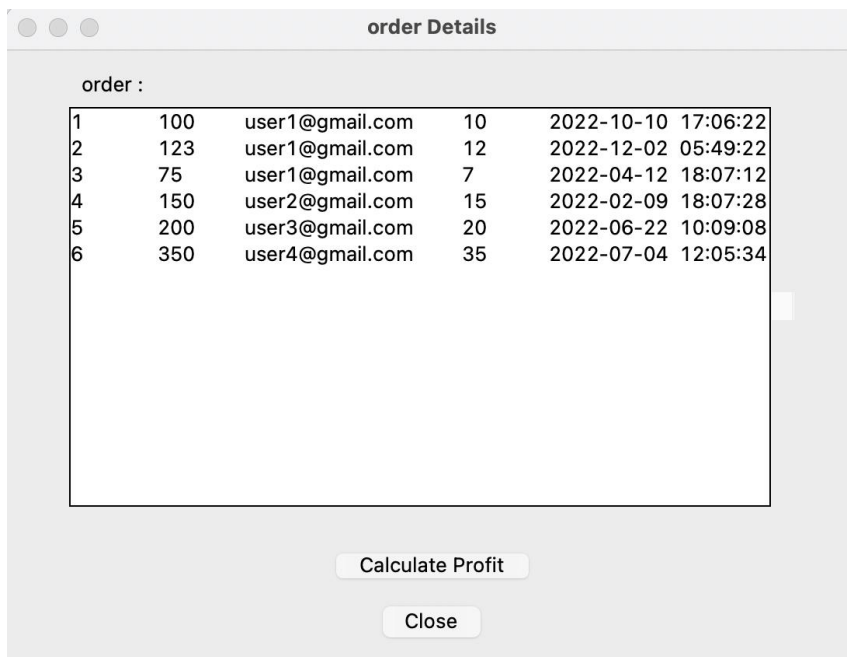
Enter Name:

Enter Surname:

Enter Phone:

Enter Vehicle type:

Close Add rider



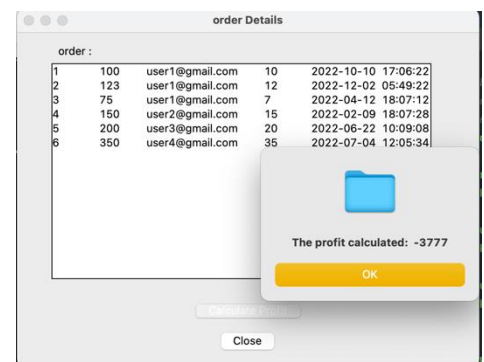
order Details

order :

1	100	user1@gmail.com	10	2022-10-10	17:06:22
2	123	user1@gmail.com	12	2022-12-02	05:49:22
3	75	user1@gmail.com	7	2022-04-12	18:07:12
4	150	user2@gmail.com	15	2022-02-09	18:07:28
5	200	user3@gmail.com	20	2022-06-22	10:09:08
6	350	user4@gmail.com	35	2022-07-04	12:05:34

Calculate Profit Close

The next window shows all the orders on the platform and how much money we earned from it, we can also calculate profit using the button provided.



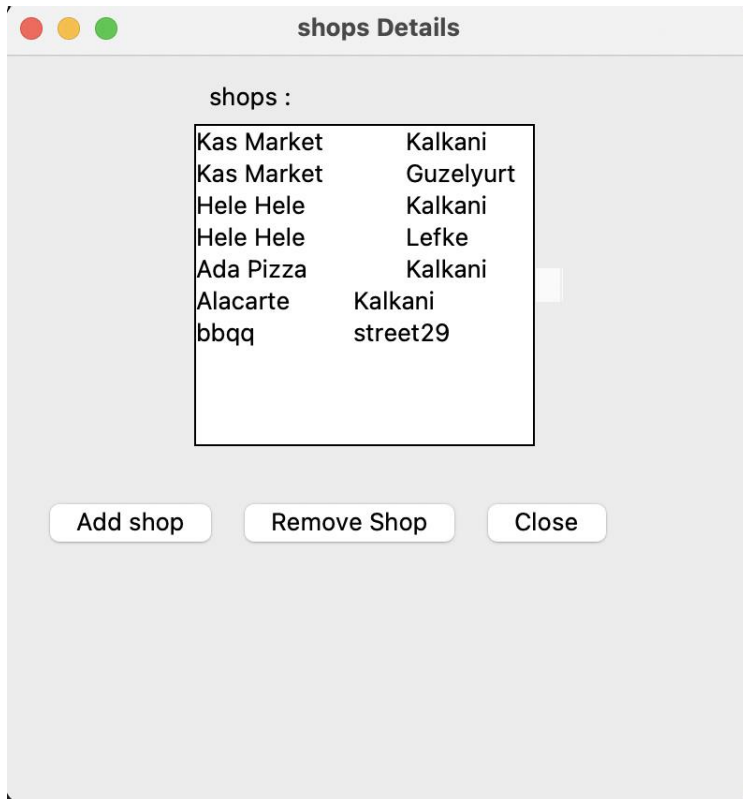
order Details

order :

1	100	user1@gmail.com	10	2022-10-10	17:06:22
2	123	user1@gmail.com	12	2022-12-02	05:49:22
3	75	user1@gmail.com	7	2022-04-12	18:07:12
4	150	user2@gmail.com	15	2022-02-09	18:07:28
5	200	user3@gmail.com	20	2022-06-22	10:09:08
6	350	user4@gmail.com	35	2022-07-04	12:05:34

The profit calculated: -3777

OK Close



shops :

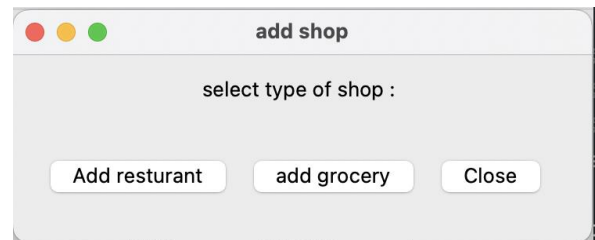
Kas Market	Kalkani
Kas Market	Guzelyurt
Hele Hele	Kalkani
Hele Hele	Lefke
Ada Pizza	Kalkani
Alacarte	Kalkani
bbqq	street29

Add shop Remove Shop Close

The last window provided for the admin interface is the shop details window where it gives him the information about all the restaurants on the platform.

It allows the admin to remove shops the same exact way we remove shops in other parts of this application.

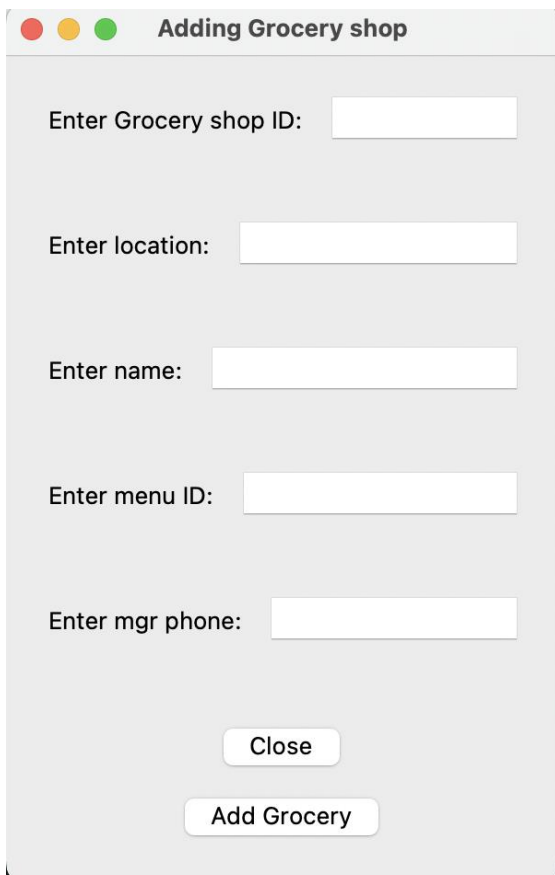
It also allows the user to add another shop providing the menu.



add shop

select type of shop :

Add resturant add grocery Close



Adding Grocery shop

Enter Grocery shop ID:

Enter location:

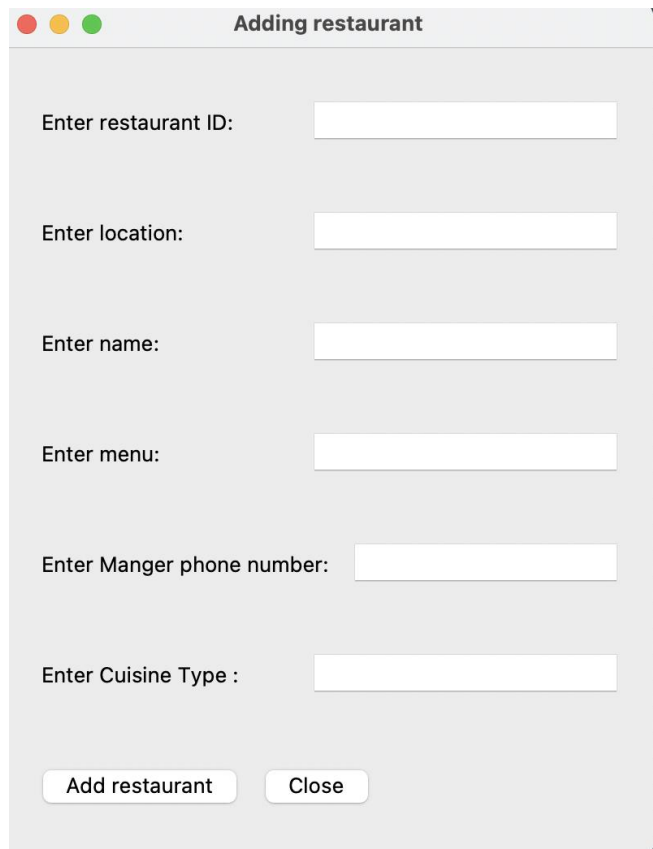
Enter name:

Enter menu ID:

Enter mgr phone:

Close

Add Grocery



Adding restaurant

Enter restaurant ID:

Enter location:

Enter name:

Enter menu:

Enter Manger phone number:

Enter Cuisine Type :

Add restaurant Close

Modification and changes to the database system:

After implementing the interface we had to add some queries to our previous working model for checking, adding , updating and deleting items in the database. However no major changes were done to the database structure. Attached is our code for checking our interface code and our database dump.