Project team 17 - University Cafeteria Management

Team Members:
Adwait Tathe
Mohit Saini
Srikanth Narayana Murthy
Vineetha Kagitha

Objective -
A software solution to manage a University Cafeteria system.

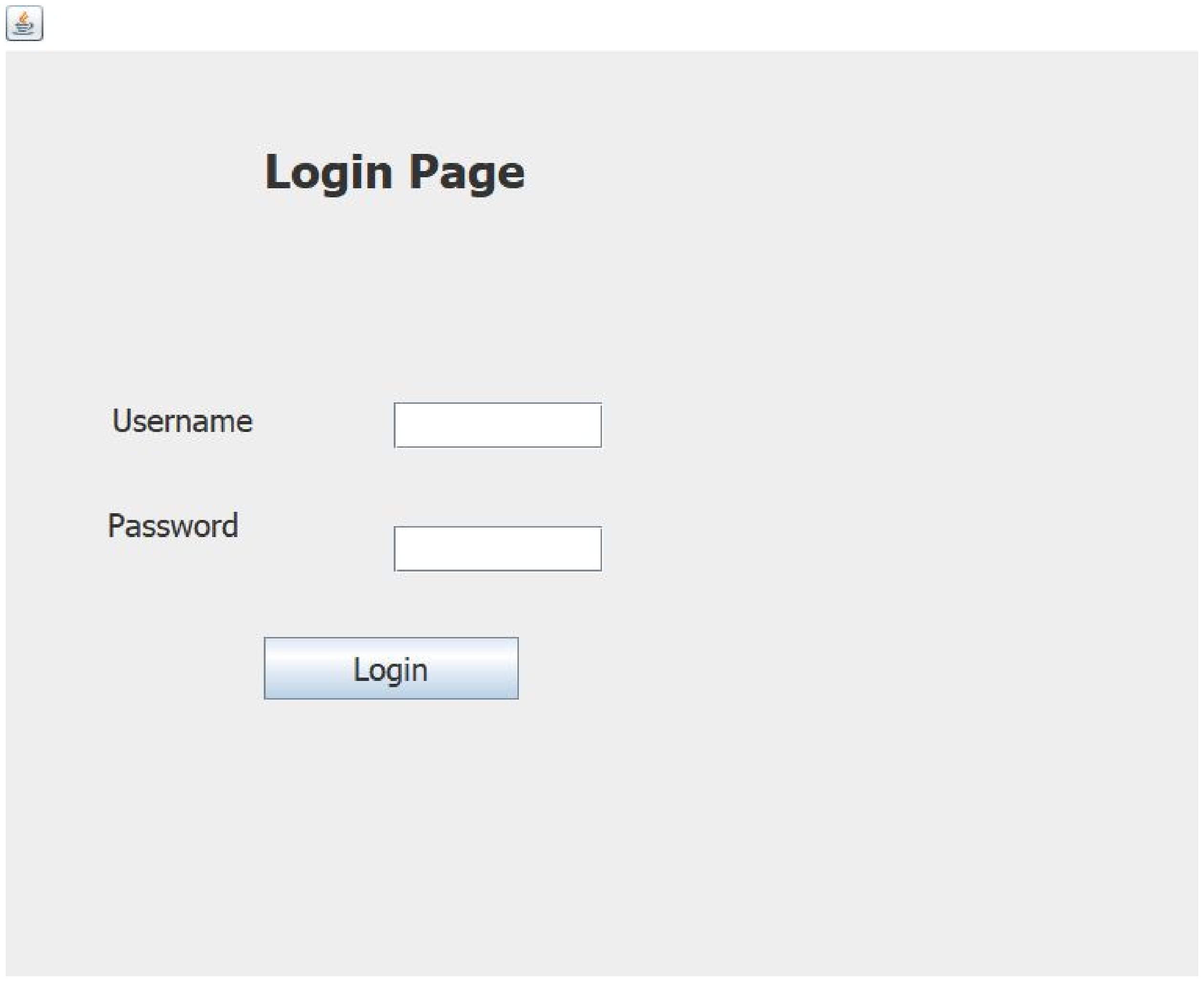
Usually People have to go to cafeteria and order the food and they have to wait in queue for a long time to get the orders.

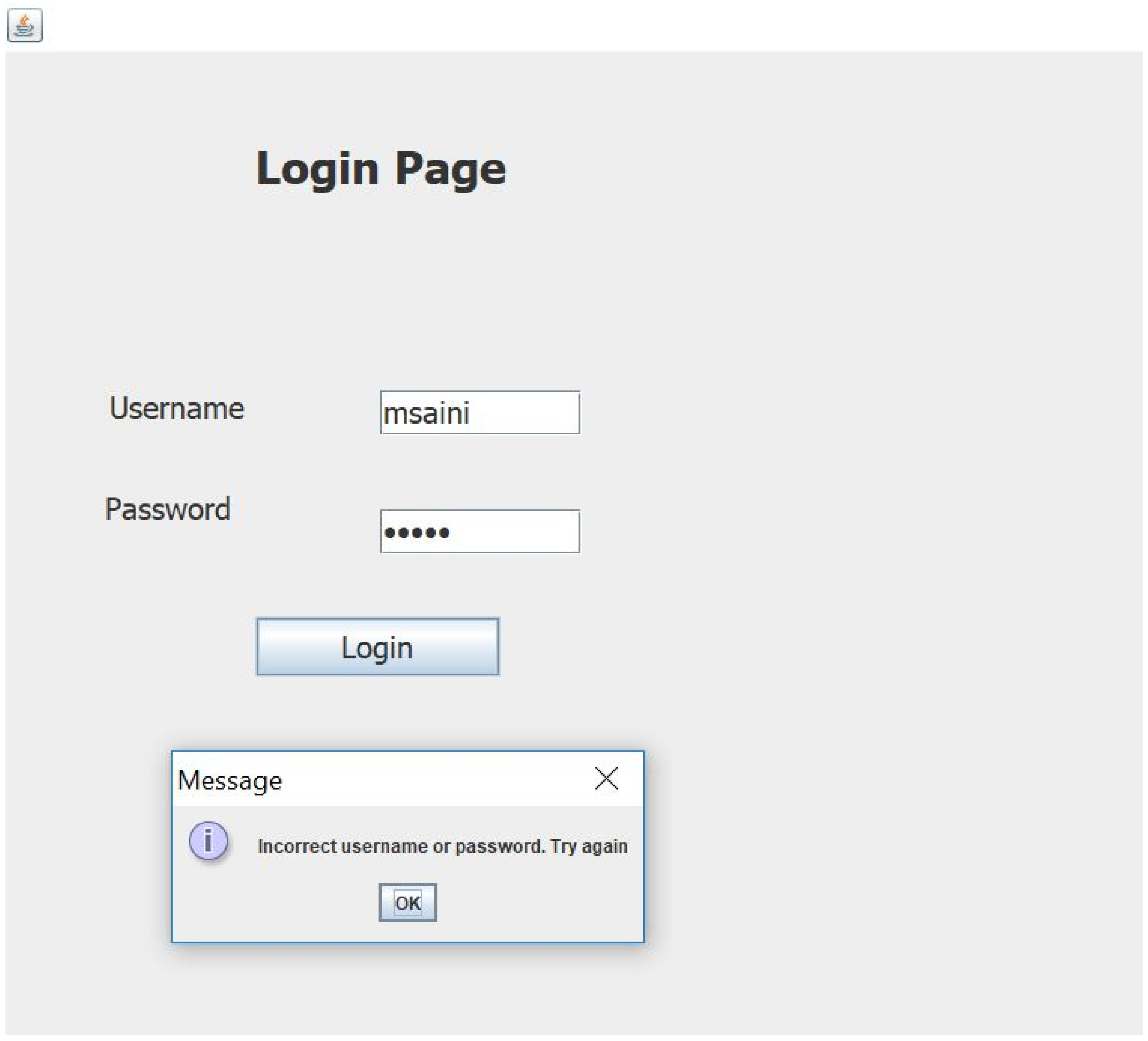
This project is a web application, with the help of this users (Students/Employee/Staff of university) will follow a very simple process to order food stuffs without waiting in the long queue and can pick the food once the order is ready.

UNC charlotte Students/Employee/Staff can make the payment from Niner wallet. We will fetch niner wallet data from university dummy database.

PROJECT ENVIRONMENT

Name of component	**Specification**
Operating System	Windows 10
Language	Java(Java 8 Runtime Environment)
Database	MySQL Workbench 8.0.12
Browser	Any of Mozilla, Opera, Chrome etc.
Web Server	Tomcat 7
Software Development Kit	Java JDK 1.8.0_181 or Above
Scripting Language	For sprint 0 environment setup - Java Swing, AngularJS
Database .jar connector	Mysql-connector


Login Page
Username
Password
Login



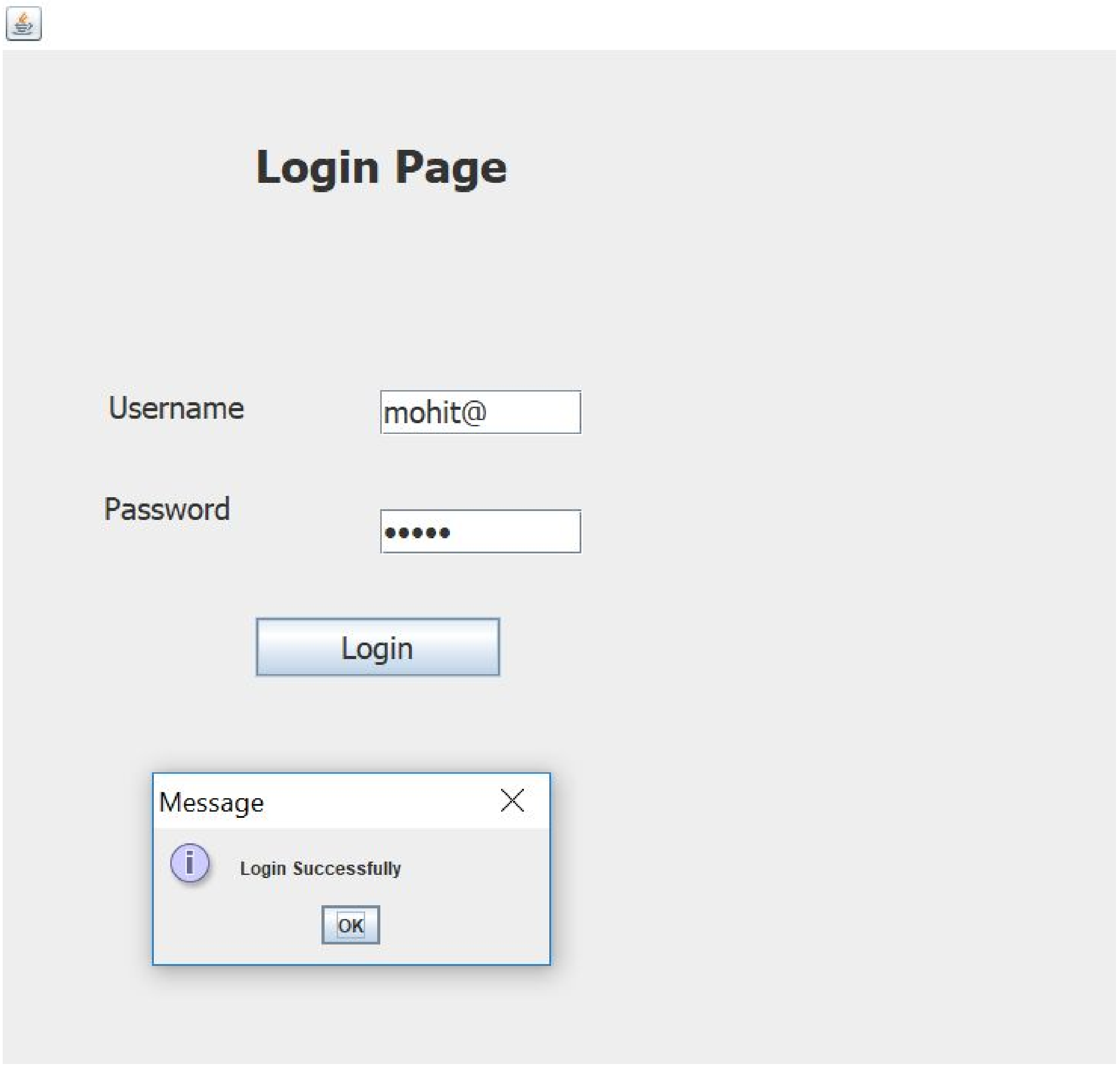
Login Page

Username msaini

Password •••••

Message

Incorrect username or password. Try again



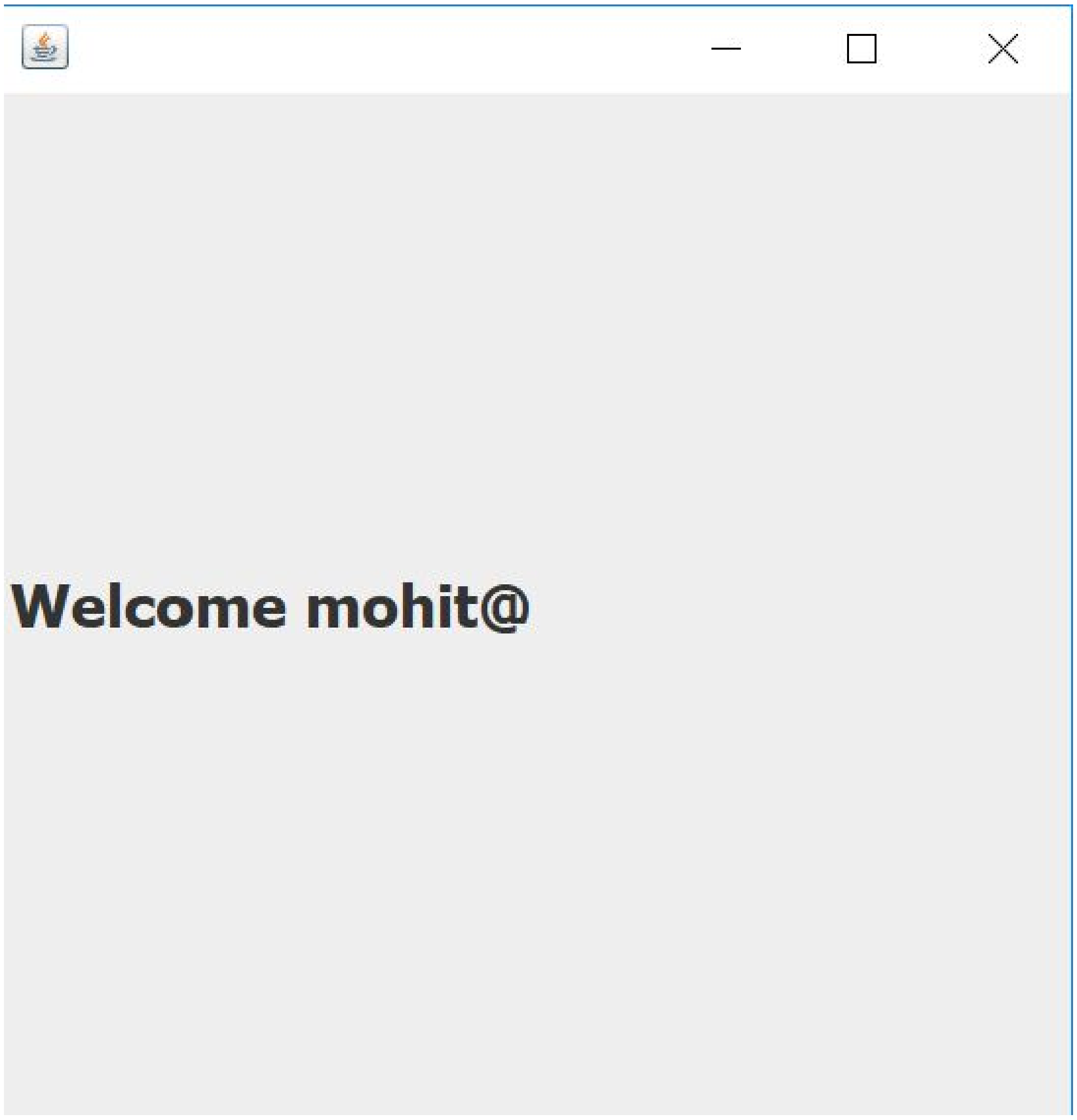
Login Page

Username mohit@

Password •••••

Message

Login Successfully


Welcome mohit@

HIGH LEVEL REQUIREMENTS

Initial user roles

User Role	**Description**
User(Students/ Employee/Staff)	Users can login to system with NINER credentials and can order food from the web-application.

	1. Student/Faculty/Staff can login using University ID validated with University Student database. 2. After Login, They can browse the food menu and add items to the cart. 3. Users can search for any food item or any restaurant using the search bar on their home screen. 4. They can checkout and pay using their Niner Wallet. 5. Order ID will be generated for the successful orders.
Administrator	Administrator can perform CRUD operations on Vendor’s data. Administrator can provide or modify the access for both existing and new vendors.
Vendor*	Service vendors can create a free account with the system and can manage their food orders.There is one Vendor for each shop. 1. Vendors can add food items, can delete items or can update prices or descriptions of food items 2. Vendors can login with their credentials, un-authorize users will get user-friendly error messages. 3. Vendors will receive user orders on homepage , which auto refreshes after a given interval of time. 4. Vendors will mark the order as “Ready” which will notify the user.

* There will be 3 to 4 different Vendors having same functionalities as described above.

Initial user story descriptions

Story ID	Story description
US1	As a User*, I want to log in/log out to/from application by using university Niner account.
US2	As a User*, I want to search for service vendors available in the application.
US3	As a User*, I want to list food items provided by service vendors.
US4	As a User*, I want to see image, price and description/ingredients of listed food items.
US5	As a User, I want to place food order.
US6	As a User, I want to get approx. food preparation time and order id
US7	As u User, I want to get food prepared notification.
US8	As a Service Vendor*, I want to log in/log out to/from application.
US9	As a Service Vendor*, I want to search/add/update/delete food items in my store.
US10	As a Service Vendor*, I can notify users whenever the food order is ready.
US11	As an Administrator, I can add/update/delete any service vendor.

HIGH LEVEL CONCEPTUAL DESIGN

Entity 1 : **User**

Entity 2 : **Food**

Entity 3 : **Order**

Entity 4: **Vendor**

Entity 5: **Administrator**

Relationships

1. **User** gives **Order**
   Cardinality : One to Many
   Participation : User has Partial participation
   Order has Total participation

2. **Vendor** manages **Food**
   Cardinality : One to Many
   Participation: Vendor has Total participation
   Food has Total participation

3. **Order** contains **Food**
   Cardinality : Many to Many
   Participation: Order has Total Participation
   Food has Partial Participation

4. **Administrator** manages **Vendors**
   Cardinality: Many to Many
   Participation: Administrator has Partial Participation
   Vendors has Total Participation

SCOPE -

Typical system users will include thousands of university members who can use this web application to easily order their food online.
It will also include Vendors (such as Bojangles, Papa John, Starbucks, etc.) who can manage their sales through this web application.

Sprint 1

REQUIREMENTS

Story ID	**Story description**
US1	As a User*, I want to login to application by using university Niner account.
US2	As a User*, I want to search for service vendors available in the application.
US3	As a User*, I want to list food items provided by service vendors.
US4	As a Service Vendor*, I want to login to application.
US5	As a Service Vendor*, I want to search/add/update/delete food items in my store.
US6	As a User*, I want to see image, price and description/ingredients of listed food items.
US7	As a User, I want to place food order.
US8	As a User, I want to get approx. food preparation time and order id
US9	As a User, I want to get food prepared notification.
US10	As a Service Vendor*, I can notify users whenever the food order is ready.

US11	As an Administrator, I can add/update/delete any service vendor.

CONCEPTUAL DESIGN

Entity : **User**

Attributes:

- Name [Composite]
  - firstName
  - lastName
- userId
- contactNo[multi-valued]
- ninerWallet
- emailId
- password

Entity : **Food**

Attributes

- foodName
- foodCategory: (Veg/Non Veg)
- price
- ingredients
- image
- vendorId

Entity : **Orders**

Attributes

- vendorId:
- prepTime:
- quantity:

Entity : **Vendors**

Attributes

- Name(Composite)
  - firstName

lastName
username:
password:
shopName

Entity : **Administrator**
Attributes
Name(Composite)
firstName
lastName
username:
password:
date:

Relationship:

1.**User** gives **Order**
Cardinality : One to Many
Participation : User has Partial participation
Order has Total participation

2. **Vendor** manages **Food**
Cardinality : One to Many
Participation: Vendor has Total participation
Food has Total participation

3. **Order** contains **Food**
Cardinality : Many to Many
Participation: Order has Total Participation
Food has Partial Participation

4. **Administrator** manages **Vendors**
Cardinality: Many to Many
Participation: Administrator has Partial Participation
Vendors has Total Participation

LOGICAL DESIGN

Entity : **User**

Columns:

- <u>userId</u>(Auto-generated)
- password
- firstName
- lastName
- emailID
- contactNo1
- contactNo2
- ninerWallet

Entity : **Food**

Columns:

- <u>foodId</u>(Auto Generated)
- foodName
- foodCategory
- price
- ingredients
- image
- availability
- vendorId[foreign key; references **vendorId** of **Vendor**]

Entity : **Order**

Columns:

- <u>orderId</u>(Auto Generated)
- vendorId[foreign key;references **vendorId** of **Vendor**]
- userId[foreign key; references **userId** of **Users**]
- preparationTime
- quantity

Entity : **Vendor**
Columns:

vendorId(Auto Generated)
vendorUserName
shopName
vendorFirstName
vendorLastName
Password
vendorEmail

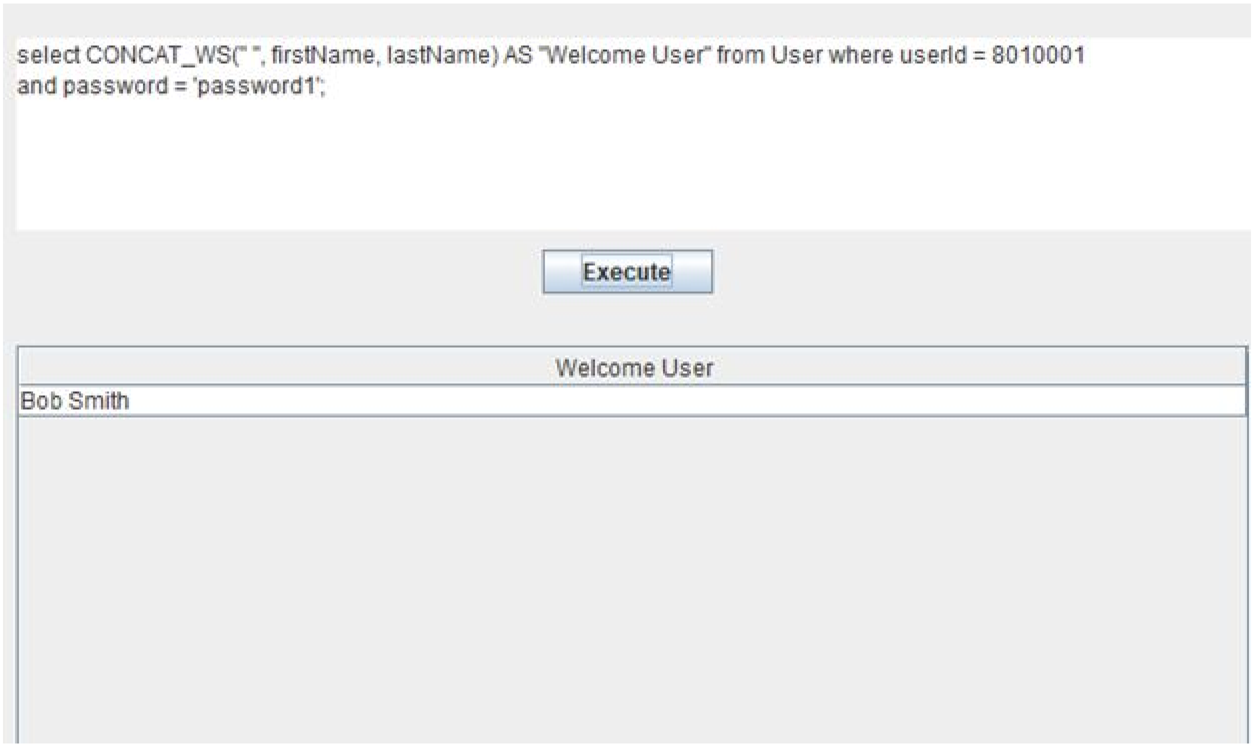
Entity : **Administrator**
Columns:

adminstratorId(Auto Generated)
password
firstName
lastName
adminUserName

SQL QUERIES

As a User*, I want to login to application by using university Niner account.

```
select CONCAT_WS(" ", firstName, lastName) AS "Welcome User" from User
where userId = 8010001
and password = 'password1';
```

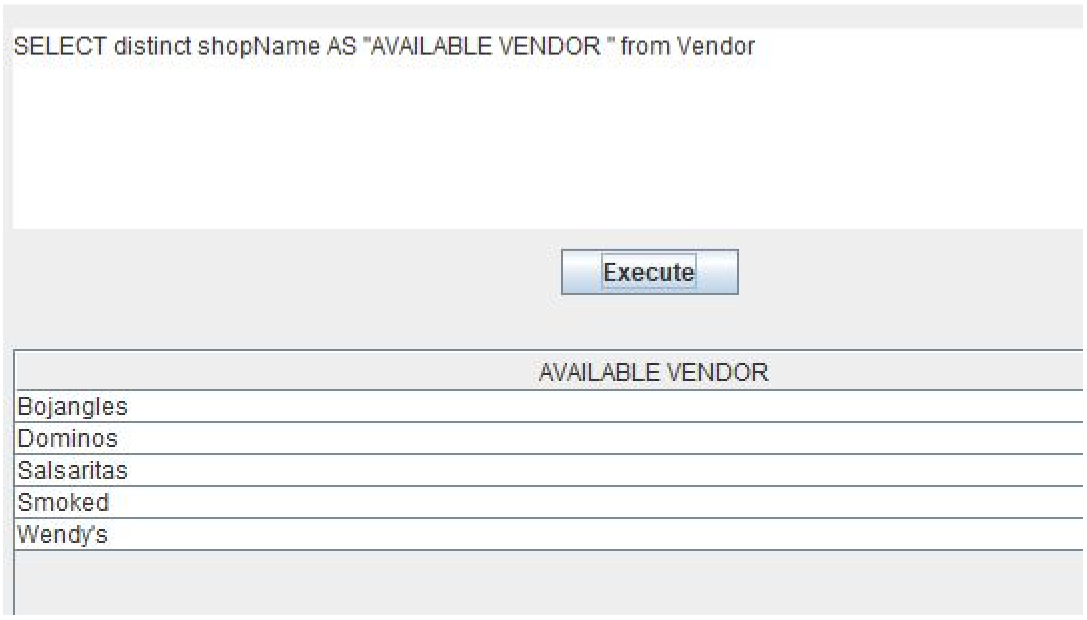


```
select CONCAT_WS(" ", firstName, lastName) AS "Welcome User" from User where userId = 8010001
and password = 'password1';
```

Welcome User
Bob Smith

As a User*, I want to search for service vendors available in the application.

SELECT distinct shopName AS "AVAILABLE VENDOR " from Vendor

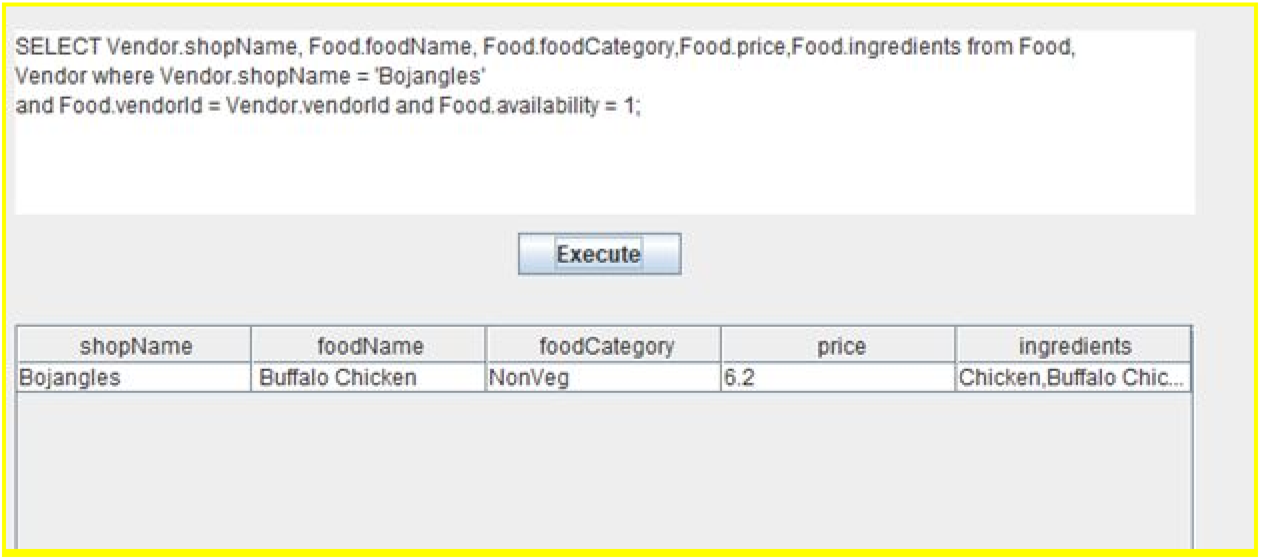


```
SELECT distinct shopName AS "AVAILABLE VENDOR " from Vendor
```

AVAILABLE VENDOR
Bojangles
Dominos
Salsaritas
Smoked
Wendy's

As a User*, I want to list food items provided by service vendors.

```
SELECT Vendor.shopName, Food.foodName,
Food.foodCategory,Food.price,Food.ingredients from Food,
Vendor where Vendor.shopName = 'Bojangles'
and Food.vendorId = Vendor.vendorId and Food.availability = 1;
```

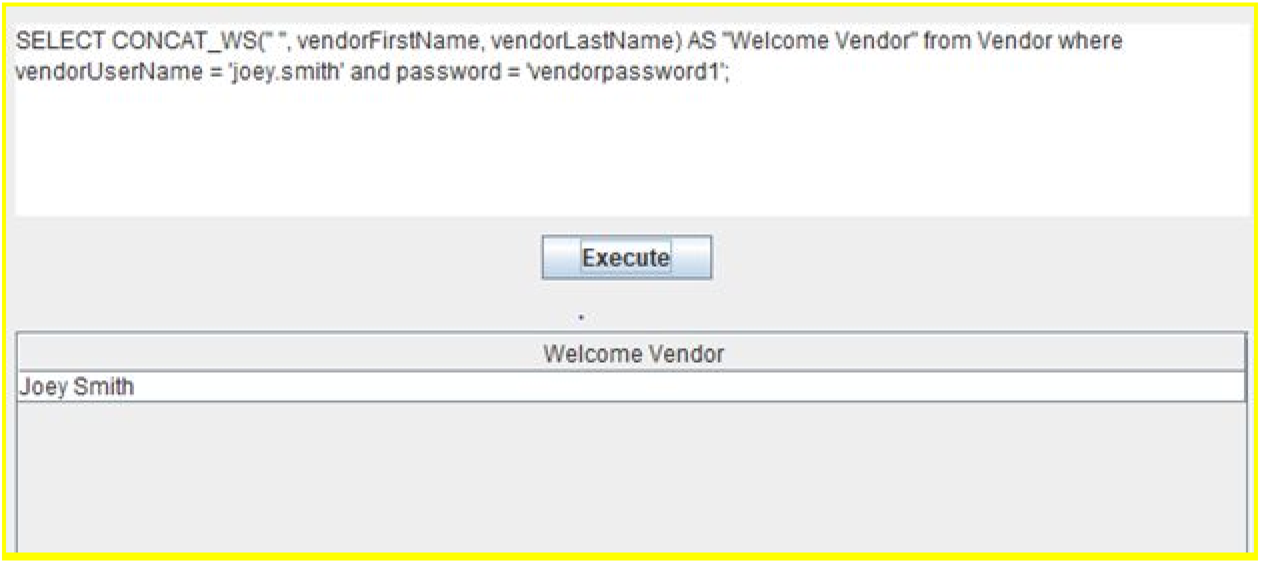


```
SELECT Vendor.shopName, Food.foodName, Food.foodCategory,Food.price,Food.ingredients from Food,
Vendor where Vendor.shopName = 'Bojangles'
and Food.vendorId = Vendor.vendorId and Food.availability = 1;
```

shopName	foodName	foodCategory	price	ingredients
Bojangles	Buffalo Chicken	NonVeg	6.2	Chicken,Buffalo Chic...

As a Service Vendor*, I want to login to application.

SELECT CONCAT_WS(" ", vendorFirstName, vendorLastName) AS "Welcome Vednor" from Vendor where vendorUserName = 'joey.smith'
and password = 'vendorpassword1';

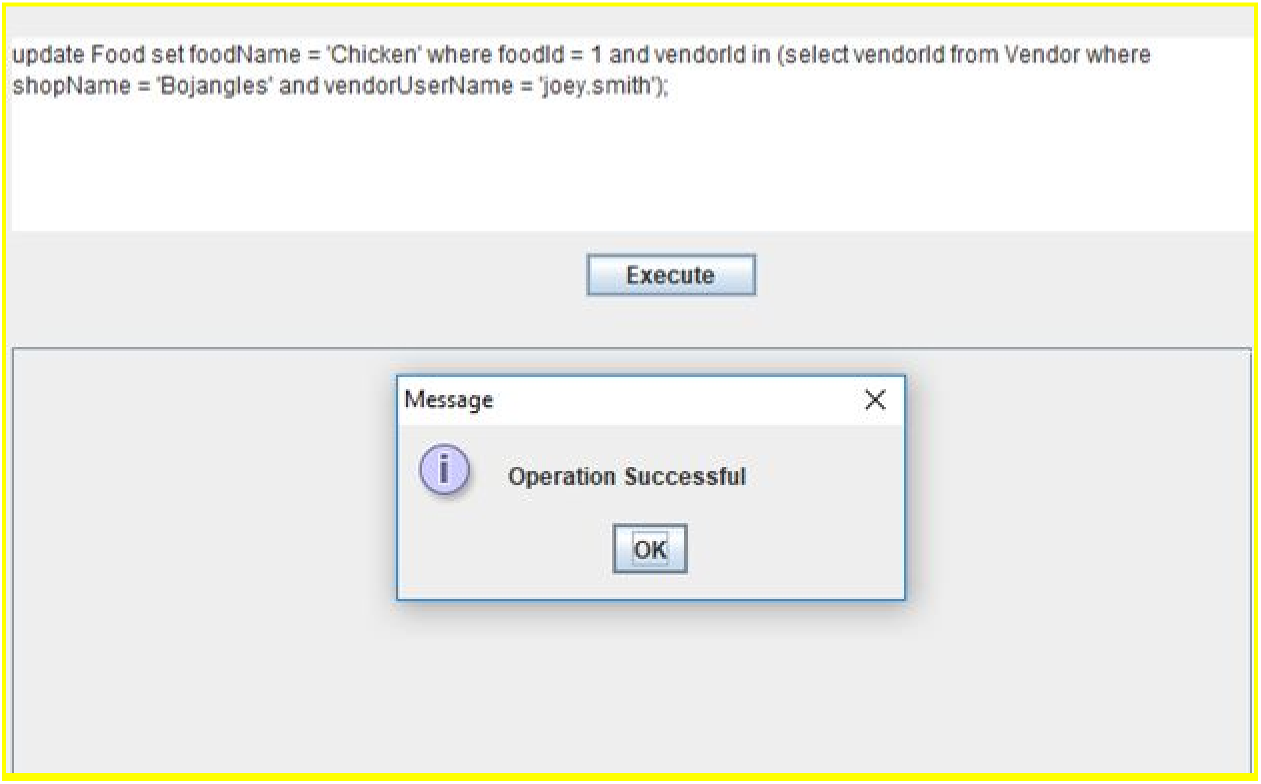


```
SELECT CONCAT_WS(" ", vendorFirstName, vendorLastName) AS "Welcome Vendor" from Vendor where
vendorUserName = 'joey.smith' and password = 'vendorpassword1';
```

Welcome Vendor
Joey Smith

*As a Service Vendor, I want to update food items in my store.

update Food set foodName = 'Chicken' where foodId = 1 and vendorId in (select vendorId from Vendor where
shopName = 'Bojangles' and vendorUserName = 'joey.smith');



```
update Food set foodName = 'Chicken' where foodId = 1 and vendorId in (select vendorId from Vendor where
shopName = 'Bojangles' and vendorUserName = 'joey.smith');
```

* We are passing fooId in query because we have safe update setting in MySQL, so we can update Food table only with primary key.
OR we can use below statement
SET SQL_SAFE_UPDATES = 0;

Sprint 2

REQUIREMENTS

Story ID	**Story description**
US1	As a User*, I want to login to application by using university Niner account.
US2	As a User*, I want to search for service vendors available in the application.
US3	As a User*, I want to list food items provided by service vendors.
US4	As a Service Vendor*, I want to login to application.
US5	As a Service Vendor*, I want to search/add/update/delete food items in my store.
US6	As a User*, I want to see image, price and description/ingredients of listed food items.
US7	As a User, I want to place food order.
US8	As a User, I want to get approx. food preparation time and order id
US9	As a user, I want to give feedback to vendors.
US10	As a user, I want to get food delivered to my location.
US11	As an Administrator, I can add/update/delete any service vendor.
US12	As a User, I want to get food prepared notification.
US13	As a Service Vendor*, I can notify users whenever the food order is ready.
US14	As a user, I want to get the invoice of my order.
US15	As a vendor, I want to include combo packs, Offer prices etc.

CONCEPTUAL DESIGN

Entity : **User**

Attributes:

- name [Composite]
  - firstName
- lastName
- userId
- contactNo[multi-valued]
- ninerWallet
- emailId
- password

Entity : **Food**

Attributes

- foodName
- foodCategory: (Veg/Non Veg)
- price
- ingredients
- image

Entity : **Vendor**

Attributes

- name[Composite]
  - firstName
  - lastName
- username
- password
- shopName

Relationship:

1. **User** Orders **Food**
   Cardinality : Many to Many
   Participation : User has Partial participation
   Food has Partial participation

2. **Vendor** manages **Food**
   Cardinality : One to Many
   Participation: Vendor has Total participation
   Food has Total participation

3. **User** gives feedback to **Vendor**
   Cardinality: One to Many
   Participation: User has Partial Participation
   Vendor has Partial Participation

LOGICAL DESIGN WITH NORMAL FORM IDENTIFICATION

Table : **User**

Columns:

userId(Auto-generated)
password
firstName
lastName
emailID
contactNo1
contactNo2
ninerWallet

Highest normalization level : 4NF

Table : **Food**

Columns:

foodId(Auto Generated)
foodName
foodCategory

price
ingredients
image
availability
preparationTime
vendorId[foreign key; references **vendorId** of **Vendor**]

Highest normalization level : 4NF

Table : **Order**
Columns:
<u>orderId</u>(Auto Generated)
vendorId[foreign key;references **vendorId** of **Vendor**]
userId[foreign key; references **userId** of **Users**]
takeAwayType ENUM(Inperson,CampusDelivery)
totalAmount (derived attribute)

Highest normalization level : 4NF

Table : **Vendor**
Columns:
<u>vendorId</u>(Auto Generated)
vendorUserName
shopName
vendorFirstName
vendorLastName
password
vendorEmail

Highest normalization level : 4NF

Table: **OrderedFood**
Columns:
<u>Id</u>(Auto Generated)
orderId[foreign key; references **orderId** of **Order**]
foodId[foreign key; references **foodId** of **Food**]
quantity

Justification: Relation between User and Food is Many to Many, hence we came up with new tables ‘order’ and ‘OrderedFood’ based on cross-reference approach.

Highest normalization level : 4NF

Table: **Feedback**

Columns:

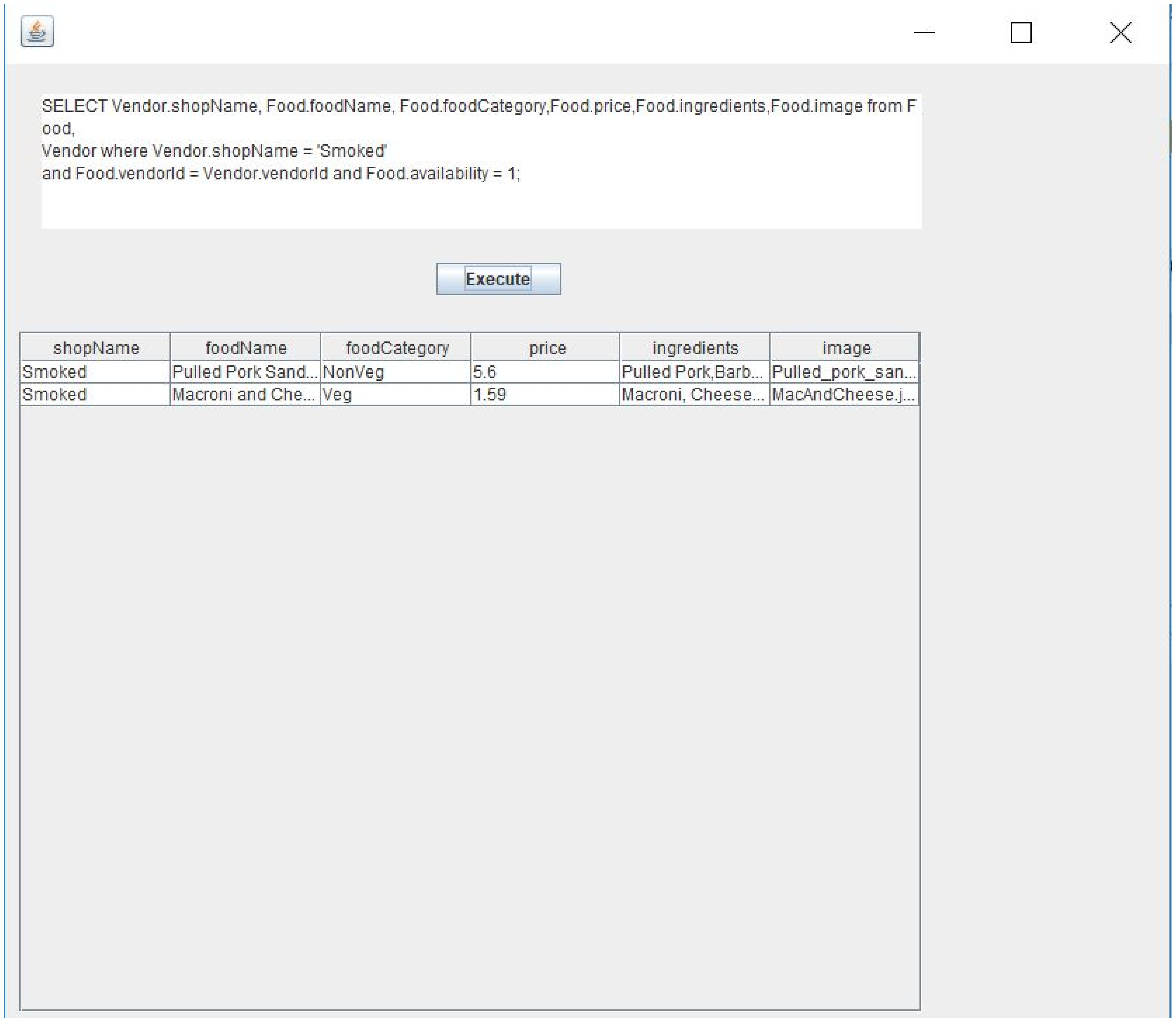
feedbackId(Auto Generated)
userId[foreign key; references **userId** of **User**]
vendorId[foreign key; references **vendorId** of **Vendor**]
suggestion
rating
Date

Highest normalization level : 4NF

SQL QUERIES

As a User*, I want to see image, price and description/ingredients of listed food items.

```
SELECT Vendor.shopName, Food.foodName,
Food.foodCategory,Food.price,Food.ingredients,Food.image from Food,
Vendor where Vendor.shopName = 'Smoked'
and Food.vendorId = Vendor.vendorId and Food.availability = 1;
```

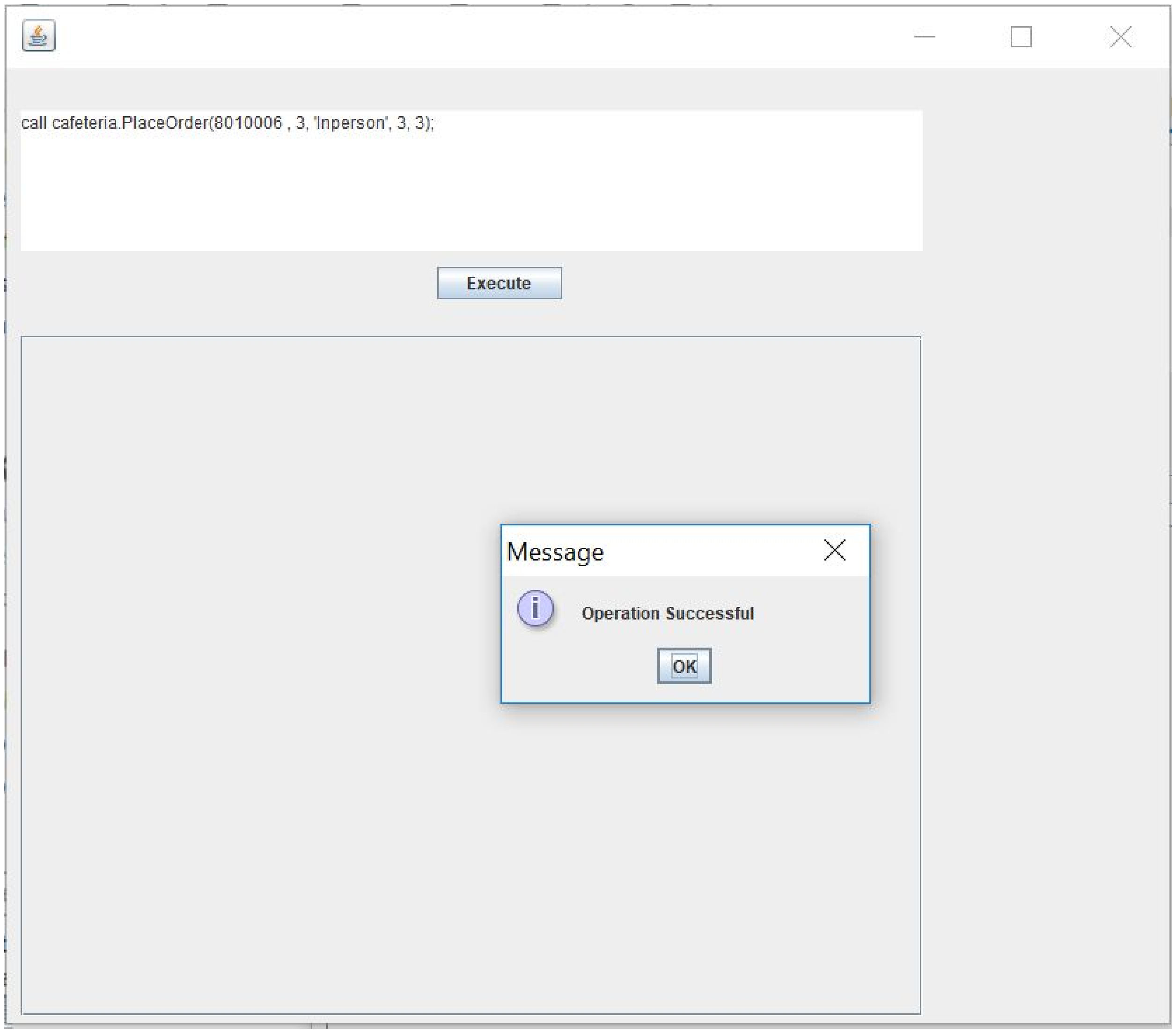


```
SELECT Vendor.shopName, Food.foodName, Food.foodCategory,Food.price,Food.ingredients,Food.image from F
ood,
Vendor where Vendor.shopName = 'Smoked'
and Food.vendorId = Vendor.vendorId and Food.availability = 1;
```

shopName	foodName	foodCategory	price	ingredients	image
Smoked	Pulled Pork Sand...	NonVeg	5.6	Pulled Pork,Barb...	Pulled_pork_san...
Smoked	Macroni and Che...	Veg	1.59	Macroni, Cheese...	MacAndCheese.j...

As a User, I want to place food order.

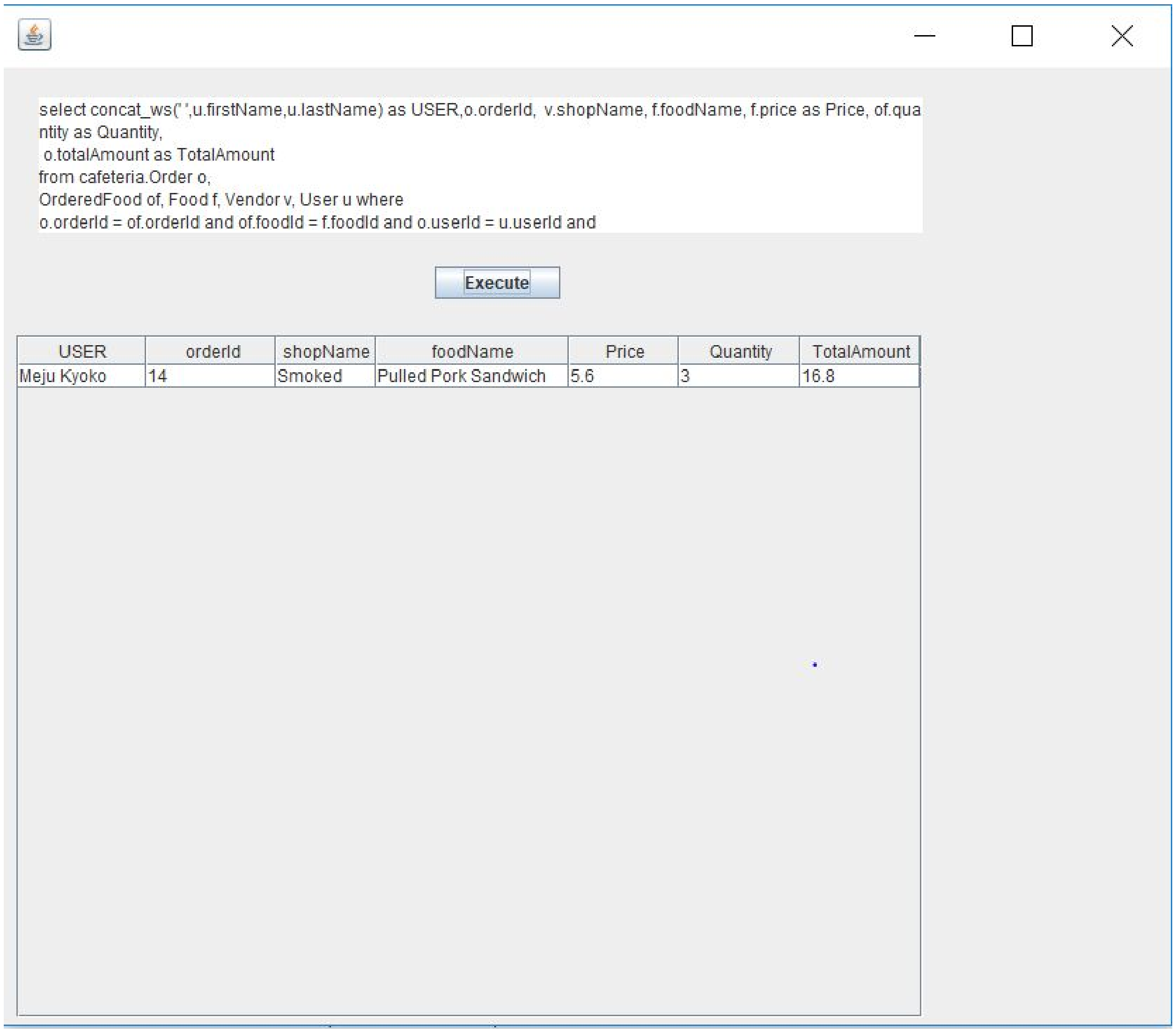
call cafeteria.PlaceOrder(8010006 , 3, 'Inperson', 3, 3);



```
call cafeteria.PlaceOrder(8010006 , 3, 'Inperson', 3, 3);
```

Operation Successful

```
select concat_ws(' ',u.firstName,u.lastName) as USER,o.orderId,
v.shopName, f.foodName, f.price as Price, of.quantity as Quantity,
 o.totalAmount as TotalAmount
from cafeteria.Order o,
OrderedFood of, Food f, Vendor v, User u where
o.orderId = of.orderId and of.foodId = f.foodId and o.userId = u.userId and
o.vendorId = v.vendorId and u.userId = 8010006;
```

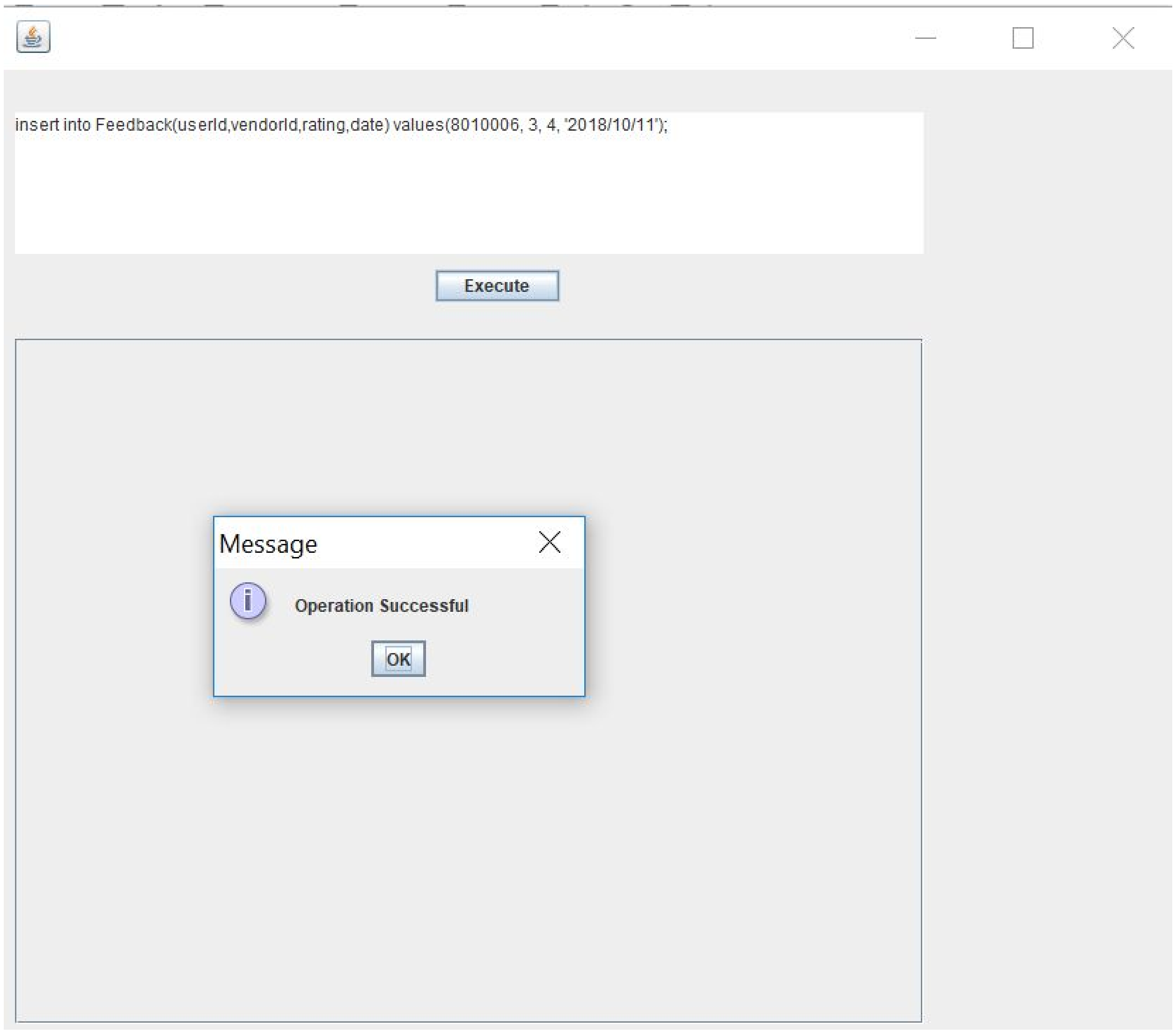


```
select concat_ws(' ',u.firstName,u.lastName) as USER,o.orderId,  v.shopName, f.foodName, f.price as Price, of.qua
ntity as Quantity,
 o.totalAmount as TotalAmount
from cafeteria.Order o,
OrderedFood of, Food f, Vendor v, User u where
o.orderId = of.orderId and of.foodId = f.foodId and o.userId = u.userId and
```

USER	orderId	shopName	foodName	Price	Quantity	TotalAmount
Meju Kyoko	14	Smoked	Pulled Pork Sandwich	5.6	3	16.8

As a user, I want to give feedback to vendors.

```
insert into Feedback(userId,vendorId,rating,date) values(8010006, 3, 4,
'2018/10/11');
```

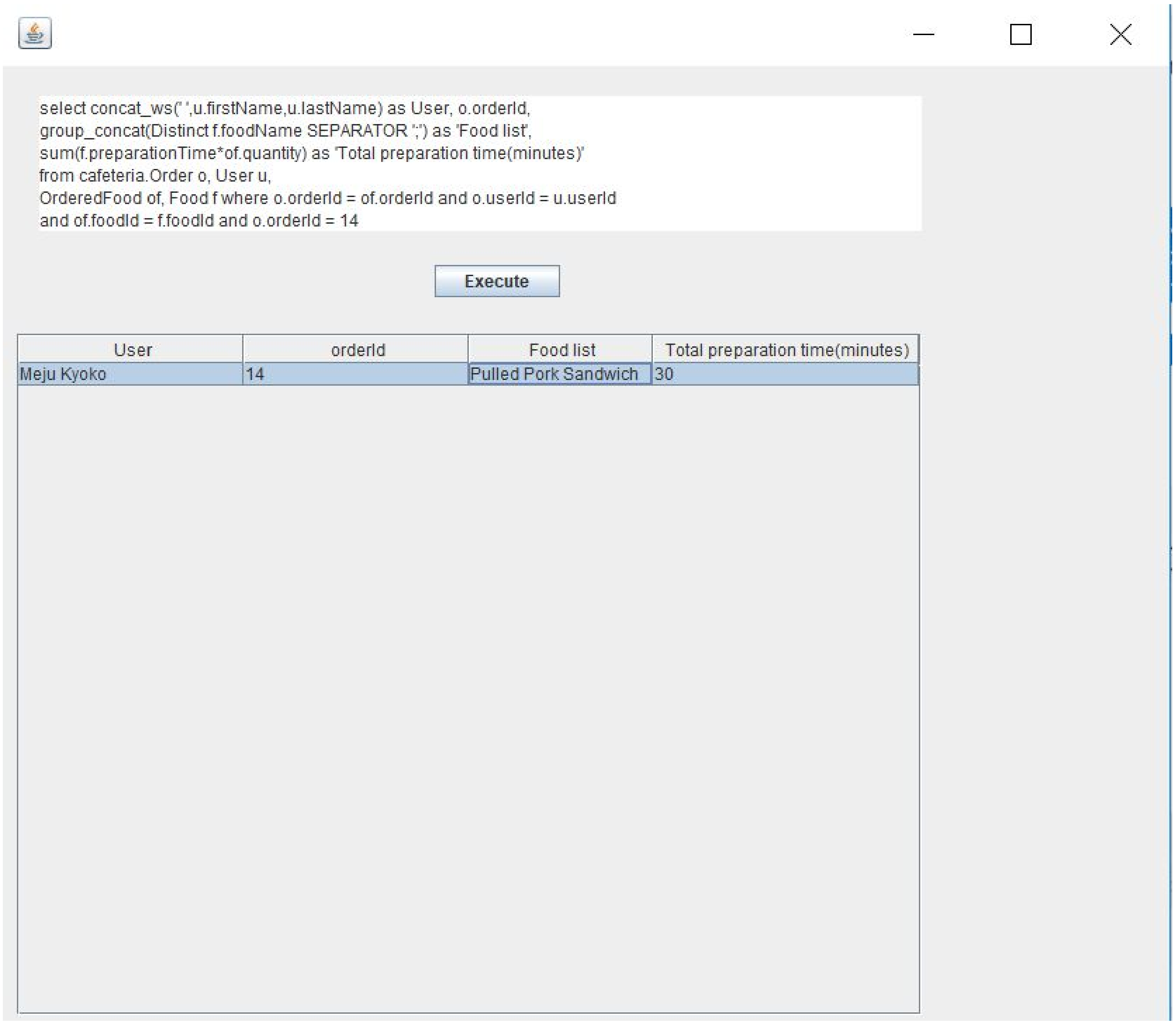


```
insert into Feedback(userId,vendorId,rating,date) values(8010006, 3, 4, '2018/10/11');
```

Operation Successful

As a User, I want to get approx. food preparation time when I place the order

```
select concat_ws(' ',u.firstName,u.lastName) as User, o.orderId,
group_concat(Distinct f.foodName SEPARATOR ';') as 'Food list',
sum(f.preparationTime*of.quantity) as 'Total preparation time(minutes)'
from cafeteria.Order o, User u,
OrderedFood of, Food f where o.orderId = of.orderId and o.userId =
u.userId
and of.foodId = f.foodId and o.orderId = 14
group by o.orderId;
```



```
select concat_ws(' ',u.firstName,u.lastName) as User, o.orderId,
group_concat(Distinct f.foodName SEPARATOR ';') as 'Food list',
sum(f.preparationTime*of.quantity) as 'Total preparation time(minutes)'
from cafeteria.Order o, User u,
OrderedFood of, Food f where o.orderId = of.orderId and o.userId = u.userId
and of.foodId = f.foodId and o.orderId = 14
```

User	orderId	Food list	Total preparation time(minutes)
Meju Kyoko	14	Pulled Pork Sandwich	30

Sprint 3

REQUIREMENTS

Story ID	**Story description**
US1	As a User*, I want to login to application by using university Niner account.
US2	As a User*, I want to search for service vendors available in the application.
US3	As a User*, I want to list food items provided by service vendors.
US4	As a Service Vendor*, I want to login to application.
US5	As a Service Vendor*, I want to search/add/update/delete food items in my store.
US6	As a User*, I want to see image, price and description/ingredients of listed food items.
US7	As a User, I want to place food order.
US8	As a User, I want to get approx. food preparation time and order id
US9	As a user, I want to give feedback to vendors.
US10	As a user, I want to get food delivered to my location.
US11	As a vendor, I want to update delivery status after food delivered
US12	As a user, I want to get the invoice of my order.

CONCEPTUAL DESIGN

Entity : **User**

Attributes:

name [Composite]

firstName

lastName

userId

contactNo[multi-valued]

ninerWallet

emailId

password

Entity : **Food**

Attributes

foodName

foodCategory: (Veg/Non Veg)

price

ingredients

image

Entity : **Vendor**

Attributes

name[Composite]

firstName

lastName

username

password

shopName

Entity : **DeliveryPerson**

Attributes

name[Composite]

firstName

lastName

workContactNo

Relationship:

1. **User** Orders **Food**
   Cardinality : Many to Many
   Participation : User has Partial participation
   Food has Partial participation

2. **Vendor** manages **Food**
   Cardinality : One to Many
   Participation: Vendor has Total participation
   Food has Total participation

3. **User** gives feedback to **Vendor**
   Cardinality: One to Many
   Participation: User has Partial Participation
   Vendor has Partial Participation

4 . **DeliveryPerson** delivers order to **User**
   Cardinality : One to Many
   Participation: DeliveryPerson has Total Participation
   User has Partial Participation

LOGICAL DESIGN WITH NORMAL FORM IDENTIFICATION

Table : **User**

Columns:

userId(Auto-generated)
password
firstName
lastName
emailID
contactNo1
contactNo2
ninerWallet

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: userId

emailID_UNIQUE: non-clustered
Columns: emailID

Justification : Both indexes will be created by default by Database System.

Table : **Food**
Columns:

<u>foodId</u>(Auto Generated)
foodName
foodCategory
price
ingredients
image
availability
preparationTime
vendorId[foreign key; references **vendorId** of **Vendor**]

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: foodId

order_fk_vendorId_idx: non-clustered
Columns: vendorId

Index idx_food_price : non-clustered
Columns: price

Justification: PRIMARY and order_fk_vendorId_idx indexes will be created by default by Database System.
User will list food items based on price frequently so we are creating index on price attribute.

Table : **Order**
Columns:

orderId(Auto Generated)
vendorId[foreign key;references **vendorId** of **Vendor**]
userId[foreign key; references **userId** of **Users**]
takeAwayType ENUM(Inperson,CampusDelivery)
totalAmount (derived attribute)

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: orderId

order_fk_vendorId_idx: non-clustered
Columns: vendorId

order_fk_userId_idx: non-clustered
Columns: userId

Justification : All indexes will be created by default by Database System.

Table : **Vendor**
Columns:

vendorId(Auto Generated)
vendorUserName
shopName
vendorFirstName
vendorLastName
password

vendorEmail

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: vendorId

vendorUserName_UNIQUE: non-clustered
Columns: VendorUserName

Index idx_Vendor_ShopName : non-clustered
Columns: shopName

Justification: PRIMARY and order_fk_vendorId_idx indexes will be created by default by Database System.

User will list available vendors in system based on shop name frequently so we are creating index on shopName attribute.

Table: **OrderedFood**
Columns:
Id(Auto Generated)
orderId[foreign key; references **orderId** of **Order**]
foodId[foreign key; references **foodId** of **Food**]
quantity

Justification: Relation between User and Food is Many to Many, hence we came up with new tables 'order' and 'OrderedFood' based on cross-reference approach.
Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: id

ordfood_fk_foodId_idx: non-clustered
Columns: foodId

Justification: PRIMARY and ordfood_fk_foodId_idx indexes will be created by default by Database System.

Table: **Feedback**
Columns:
feedbackId(Auto Generated)
userId[foreign key; references **userId** of **User**]
vendorId[foreign key; references **vendorId** of **Vendor**]
suggestion
rating
date

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: feedbackId

feedback_fk_userId_idx: non-clustered
Columns: userId

feedback_fk_vendorId_idx : non-clustered
Columns: vendorId

Justification: All indexes will be created by default by Database System.

Table : **DeliveryPerson**
Columns:

personId(Auto Generated)
firstName
lastName
workContactNo

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: personId
Justification: This index will be created by default by Database System.

Table : **Delivery**
Columns:

orderId[foreign key; references **orderId** of **Order**]
userId[foreign key; references **userId** of **User**]
vendorId[foreign key; references **vendorId** of **Vendor**]
personId[foreign key; references **personId** of **DeliveryPerson**]
locationId[foreign key; references **locationId** of **Location**]

Highest normalization level : 4NF

Indexes:
del_fk_vendorId_idx: non- clustered
Columns: vendorId

del_fk_userId_idx: non-clustered
Columns: userId

del_fk_personId_idx : non-clustered
Columns: personId

del_fk_locationId_idx: non-clustered
Columns: locationId

del_fk_orderId_idx: non-clustered
Columns: orderId

Justification:del_fk_locationId_idx,del_fk_personId_idx, del_fk_userId_idx,del_fk_vendorId_idx indexes will be created by default by Database System.

The delivery table will be frequently searched using orderId so we are creating index on it.

Table: **VendorDeliveryPerson**
Columns:
Id(Auto Generated)
vendorId[foreign key; references **vendorId** of **Vendor**]
personId[foreign key; references **personId** of **DeliveryPerson**]
availability

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: Id

vendor_fk_vendorId_idx: non-clustered
Columns: vendorId

vendor_fk_personalId_idx : non-clustered
Columns: personId

Justification: All indexes will be created by default by Database System.

Table: **Location**

Columns:

<u>locationId</u>(Auto Generated)
locationName

Highest normalization level : 4NF

Indexes:
PRIMARY:clustered
Columns: locationId

Justification: This index will be created by default by Database System.

VIEWS AND STORED PROGRAMS

<u>View</u>:

getAllOrdersOfSmoked,
getAllOrdersOfDominos,
getAllOrdersOfBojangles

Goal: To check all orders placed by user to particular vendor

<u>Stored Procedure</u>

Name: PlaceOrderDelivery
Parameters: IN in_userId INT, IN in_vendorId INT, In in_takeawaytype ENUM('Inperson','CampusDelivery')
Goal: On each call to this stored procedure, a row of order details is inserted in order table and multiple rows of food items in orderedFood table and also delivery details in delivery table if user opts Campus delivery option.

Name: set_availability
Parameters: IN personId INT, IN in_isDelivered ENUM("Yes","No")
Goal: This procedure is used to set the delivery person availability based on assigned delivery task status.

Name: GetOrderDetails
Parameters: IN in_userId INT, IN in_orderId INT
Goal: To get invoice of the order placed.

<u>Trigger</u>

Type: After Insert on **Delivery**
Goal: Trigger to set the assigned delivery person availability to unavailable on placing the order.

Type: After Update on **Delivery**
Goal: Trigger to set the assigned delivery person availability to available after delivering the order

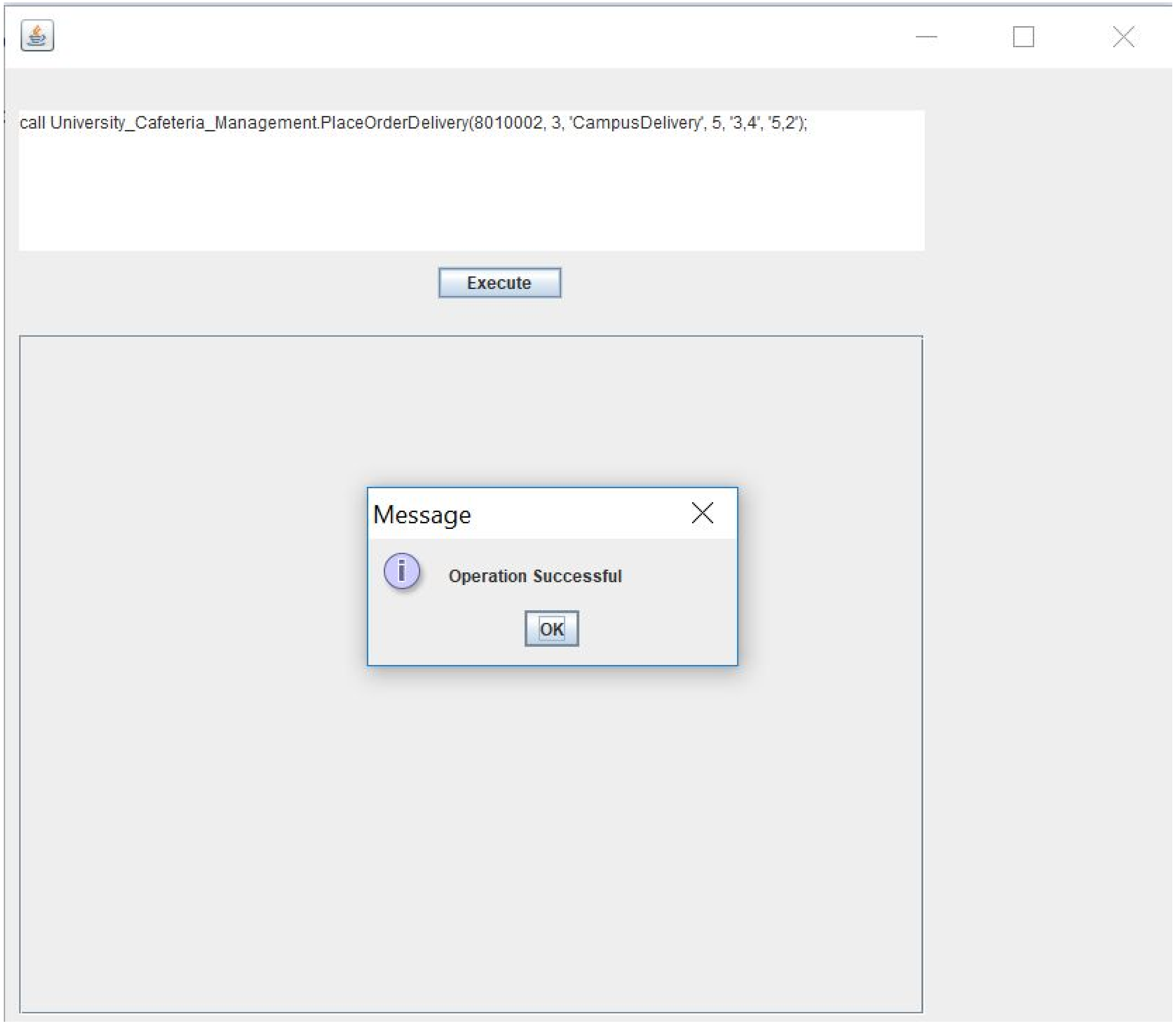
<u>Event</u>

Name: vendorDeliveryPerson_event
Goal: We introduced availability column in VendorDeliveryPerson table, So we have created this event to set it to **yes**.

SQL QUERIES

As a user, I want to get food delivered to my location.

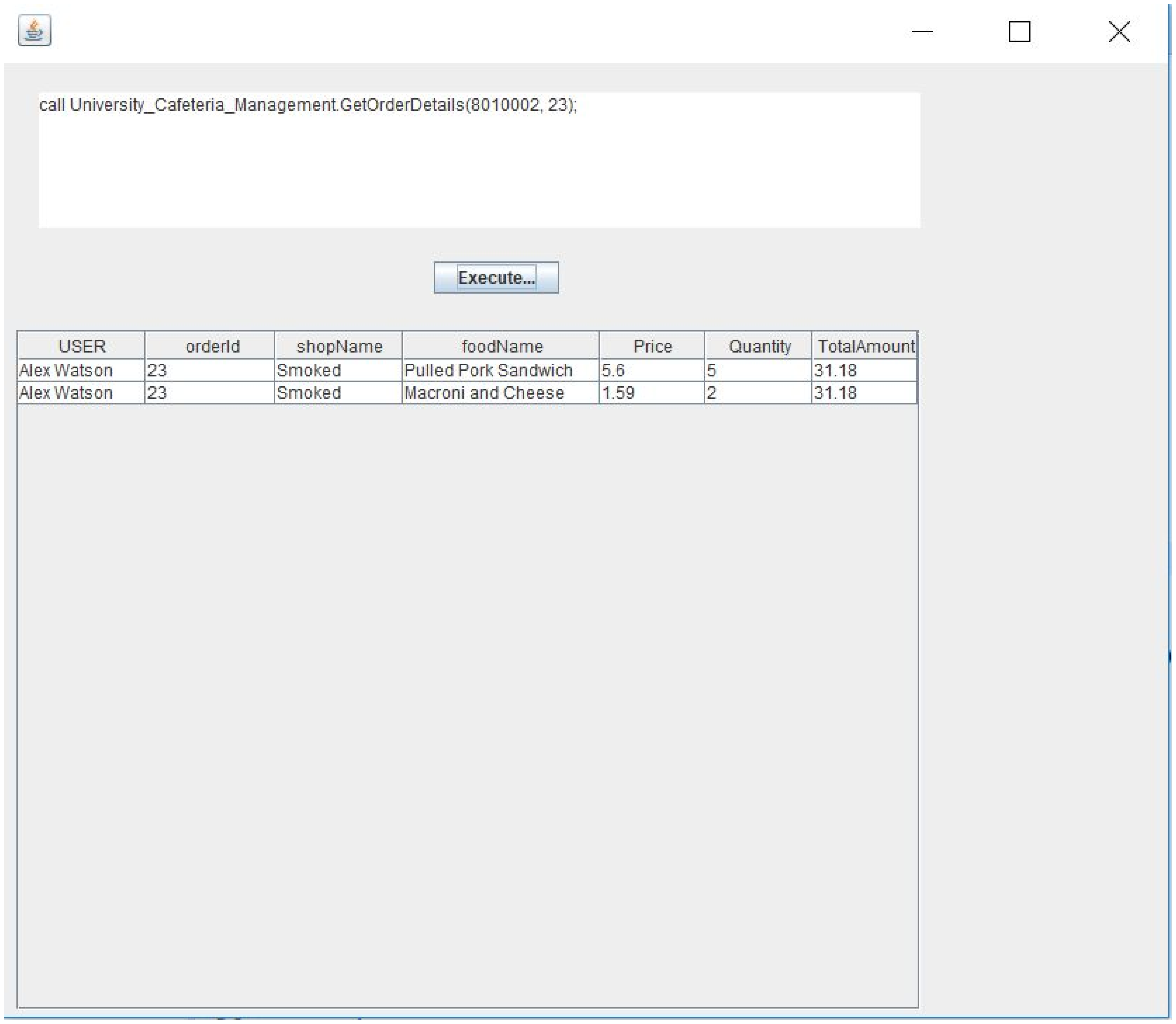
```
call University_Cafeteria_Management.PlaceOrderDelivery(8010002, 3,
'CampusDelivery', 5, '3,4', '5,2');
```



```
call University_Cafeteria_Management.PlaceOrderDelivery(8010002, 3, 'CampusDelivery', 5, '3,4', '5,2');
```

As a user, I want to get the invoice of my order.

```
call University_Cafeteria_Management.GetOrderDetails(8010002, 23);
```

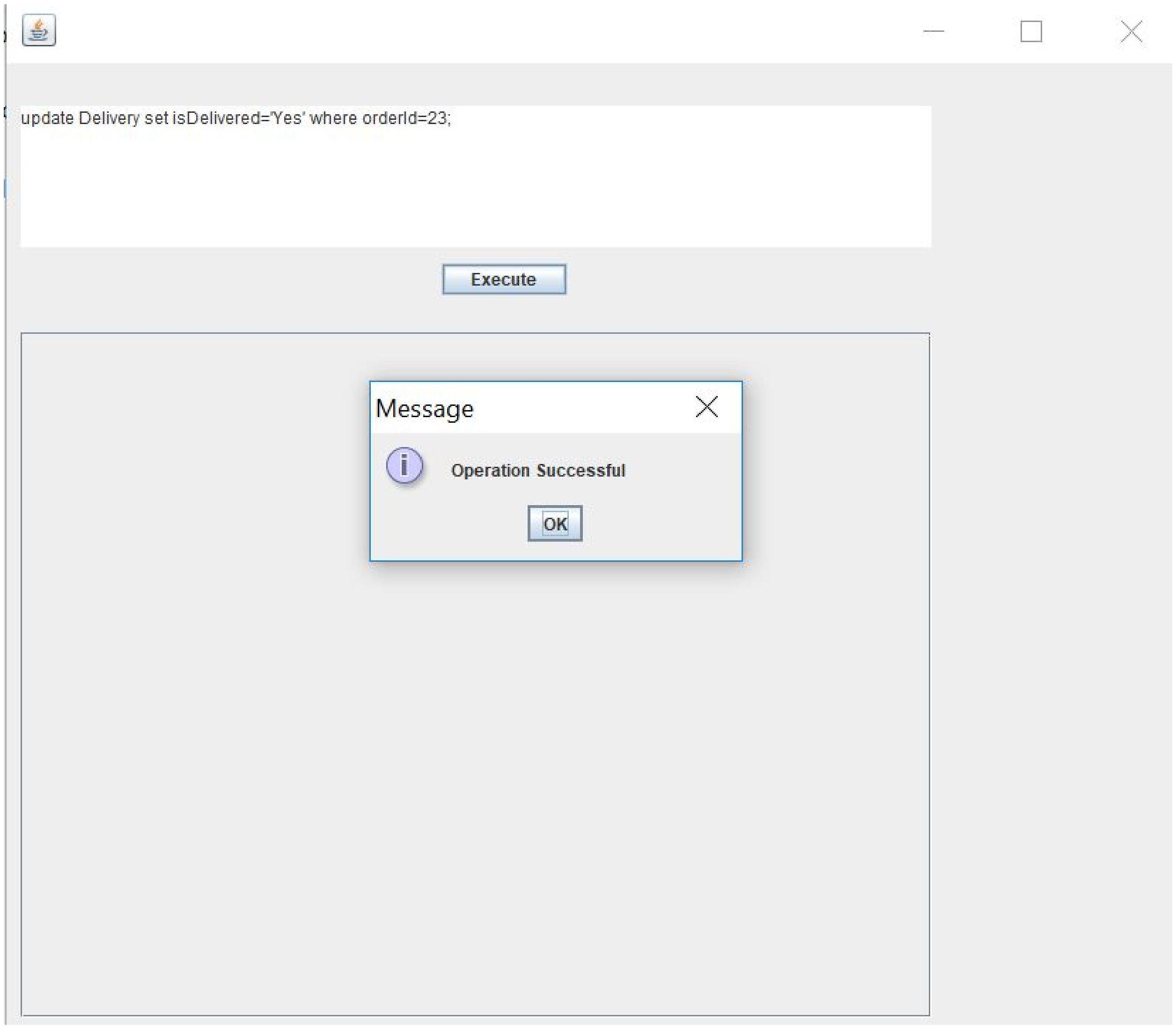


```
call University_Cafeteria_Management.GetOrderDetails(8010002, 23);
```

USER	orderId	shopName	foodName	Price	Quantity	TotalAmount
Alex Watson	23	Smoked	Pulled Pork Sandwich	5.6	5	31.18
Alex Watson	23	Smoked	Macroni and Cheese	1.59	2	31.18

As a vendor, I want to update delivery status after food delivered

update Delivery set isDelivered='Yes' where orderId=23;



```
update Delivery set isDelivered='Yes' where orderId=23;
```

Operation Successful