西北工業大學
Northwestern Polytechnical University

Object Oriented Programming

(U10M12004)

Group Project Report

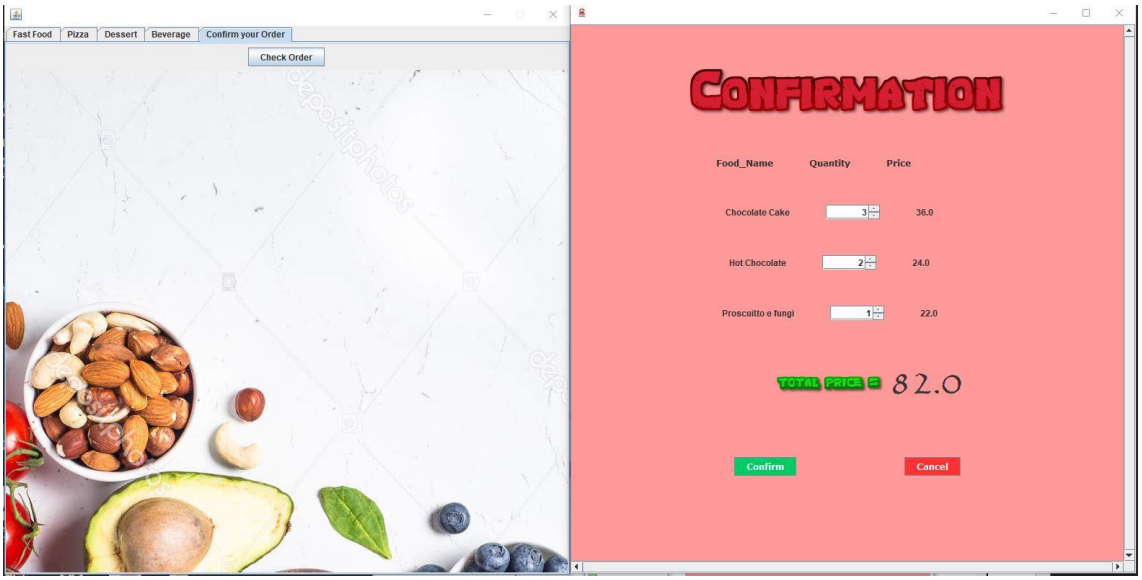
Title	Waiters and Waitresses
Group No.	06
Leader	2018380038 Amirbek Raimov
Member(s)	2018380130 Khan Md Shahedul Islam 2018380115 Wong Mann Theng, 2018380039 Dikshya Kafle

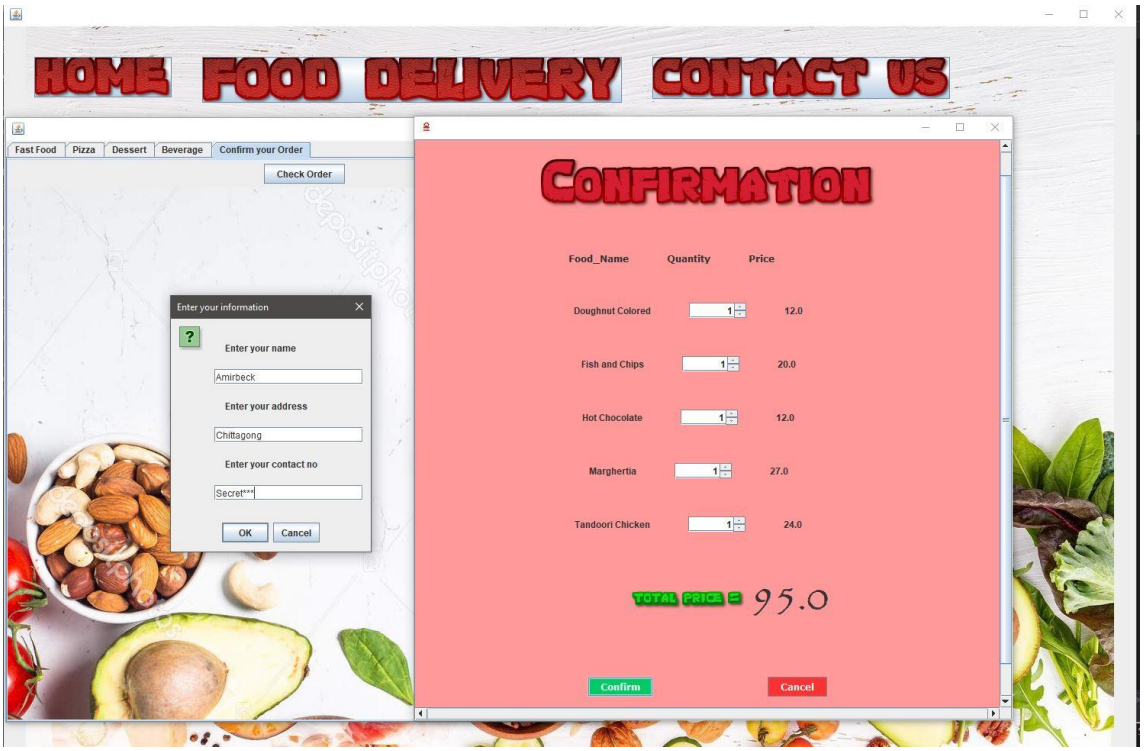
May 19 2021

I. Introduction

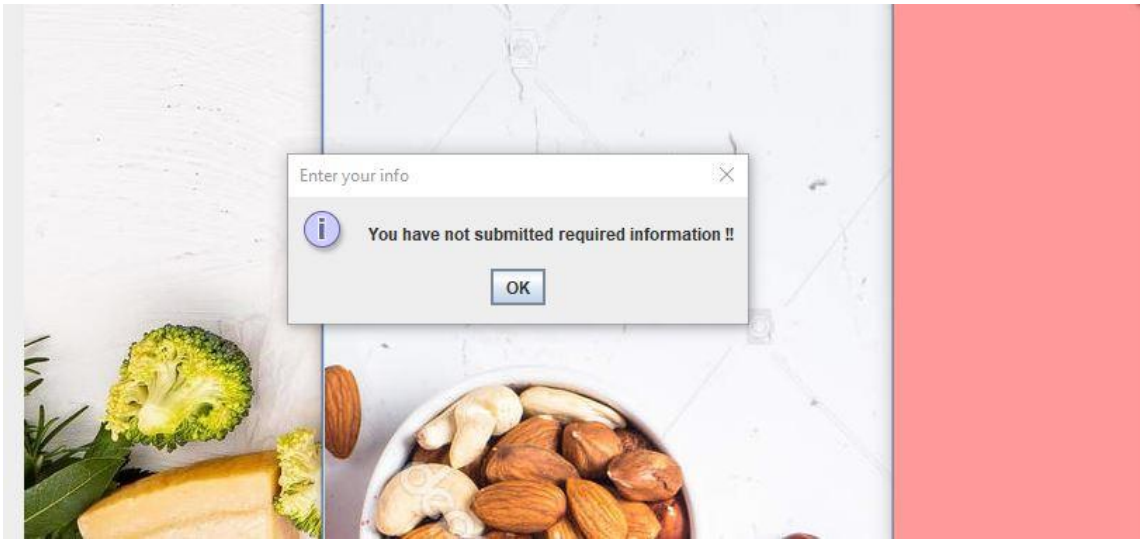
We are bringing our new ideas putting in a core java app to deliver an online food parceling system. We have designed and implemented unique and easier user interfaces for faster access. With our app anyone can surf and order all of our foods. The checkout procedure is easy and done within a click. We have included our full restaurant menu and admin support as an extension to our app. Functionalities our app offers:

- A user can access our food gallery from the app home page and can select any of the items we offer. Later moving on to check order option, user can add/drop, change quantity of the items and confirm the order after providing the user credentials (name, address, contact info).


Fast Food
Pizza
Dessert
Beverage
Confirm your Order
Check Order
CONFIRMATION
Food_Name
Quantity
Price
Chocolate Cake
3
36.0
Hot Chocolate
2
24.0
Proscuitto e fungi
1
22.0
TOTAL PRICE = 82.0
Confirm
Cancel

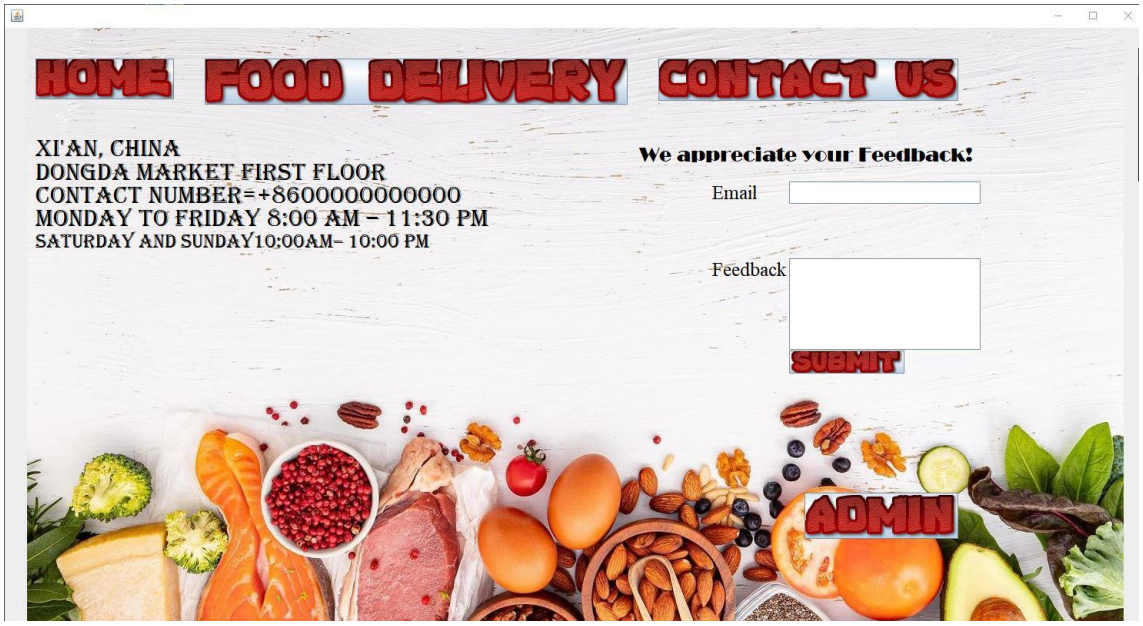

HOME
FOOD DELIVERY
CONTACT US
Fast Food
Pizza
Dessert
Beverage
Confirm your Order
Check Order
Enter your information
Enter your name
Amirbeck
Enter your address
Chittagong
Enter your contact no
Secret***
OK
Cancel
CONFIRMATION
Food_Name
Quantity
Price
Doughnut Colored
1
12.0
Fish and Chips
1
20.0
Hot Chocolate
1
12.0
Marghertia
1
27.0
Tandoori Chicken
1
24.0
TOTAL PRICE = 95.0
Confirm
Cancel

- If user does not input all the required information the system will give a pop-up reminding that.

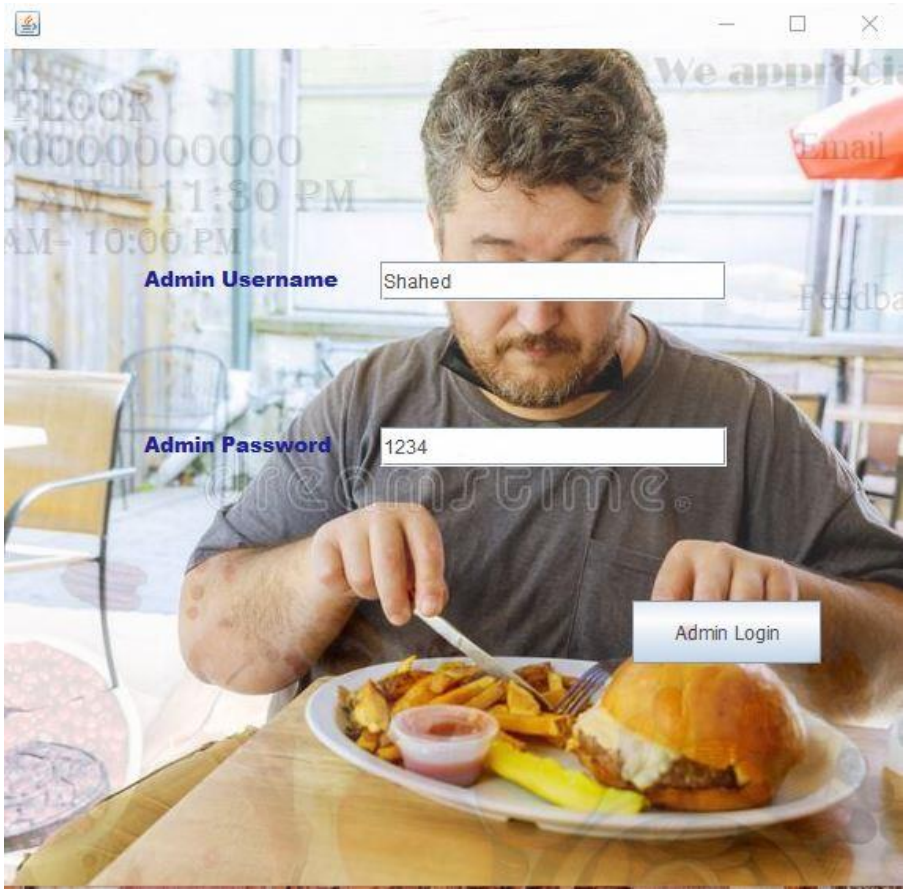


- One click order cancellation

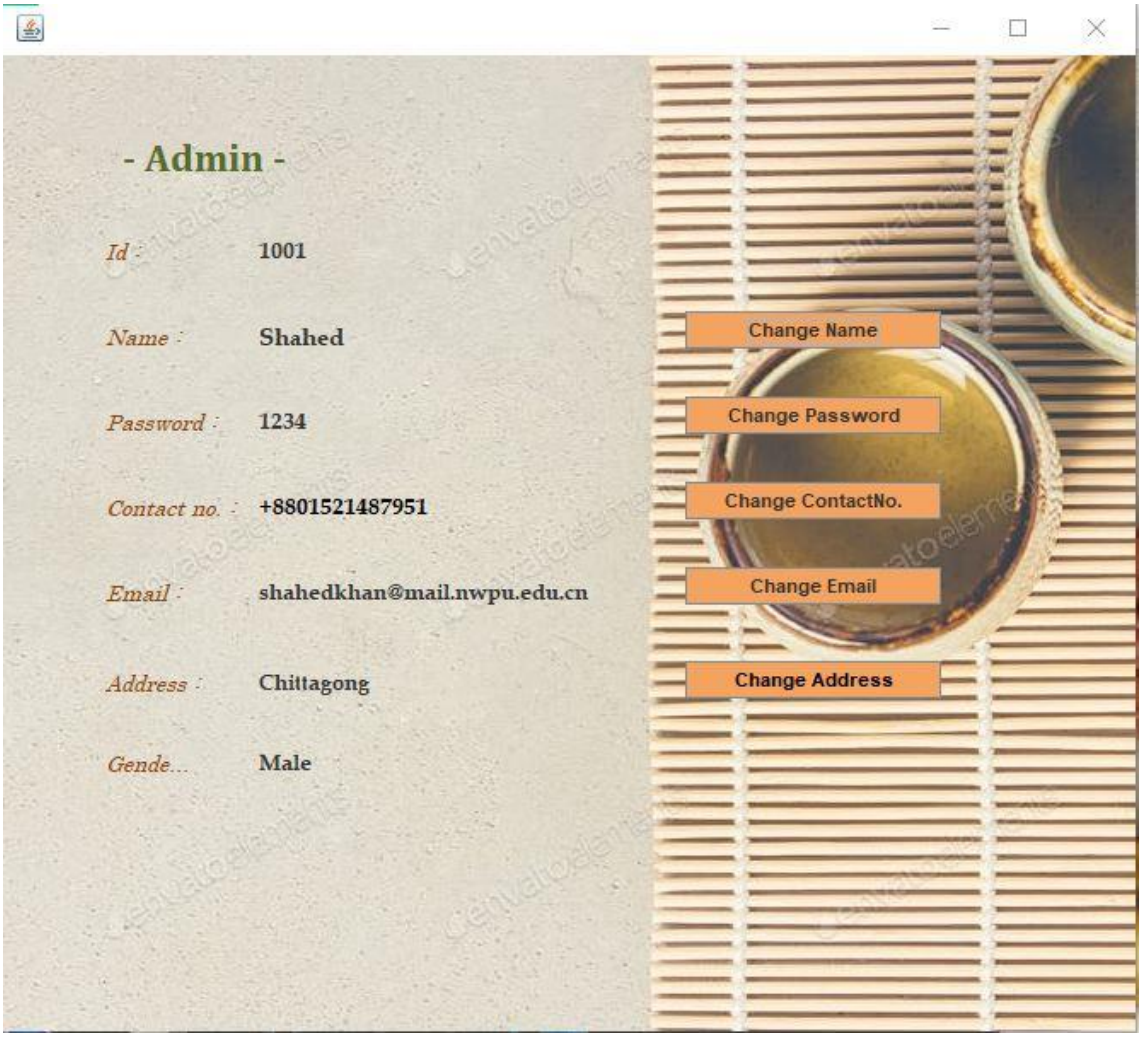
- We have provided easy to interact system between system managers and users to help us grow and new customers to decide on foods. We have a contact us functionality embedded in the home page of our app for that.



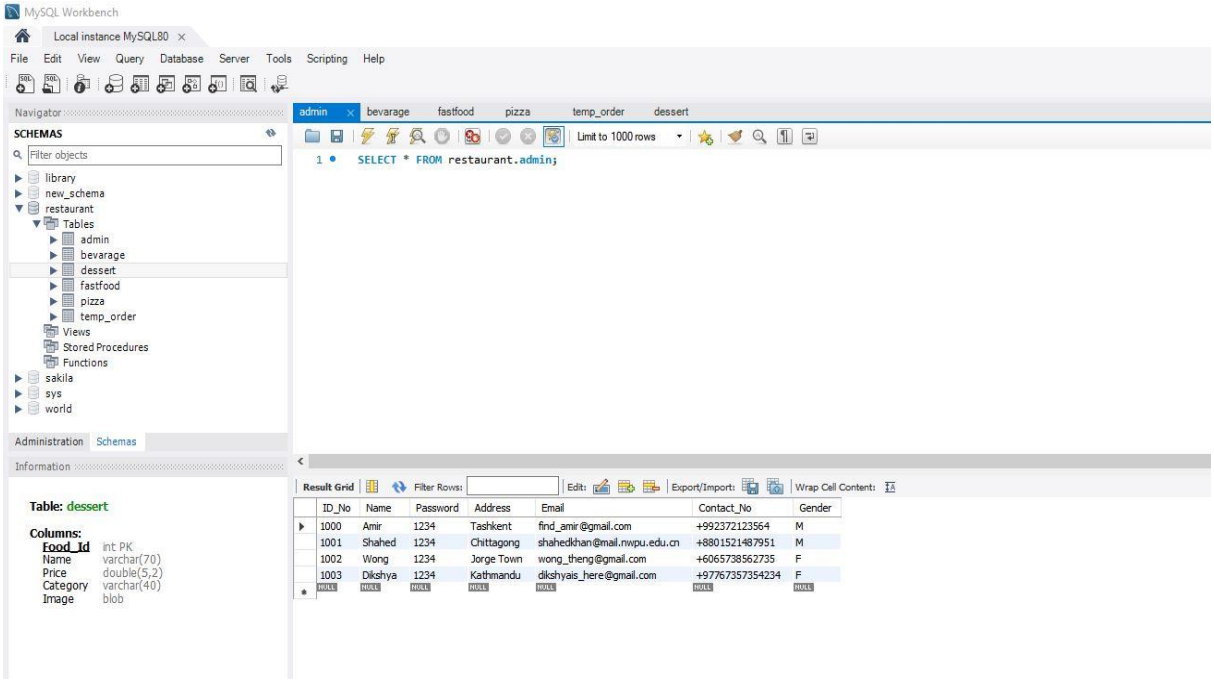
- App managers/admins can easily access the database to change admin credentials (all the credentials), update foods (price, name, category, Id in the database)



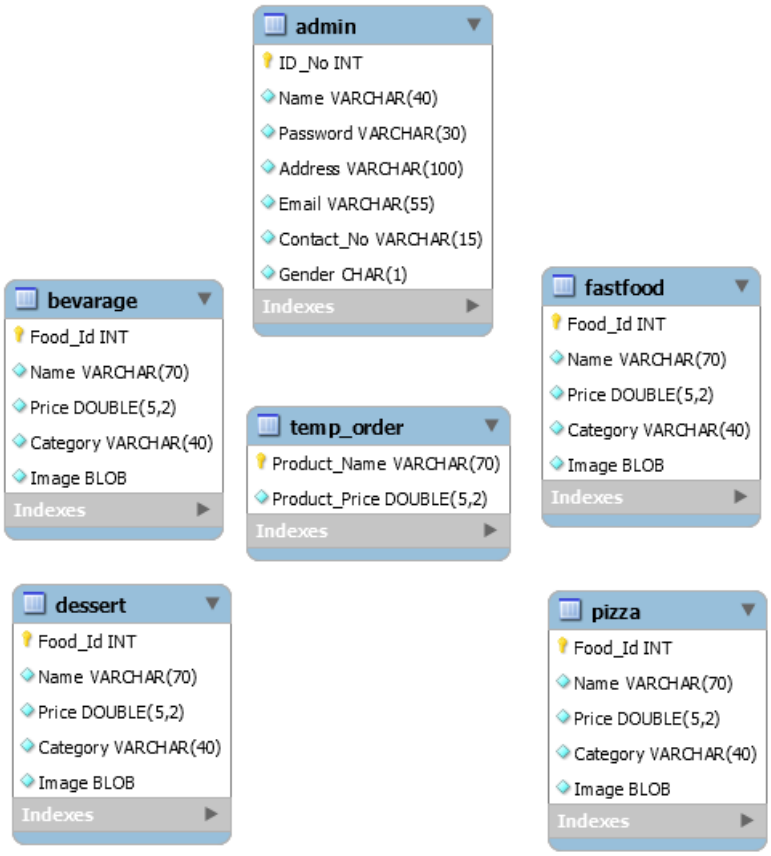

Fast Food
Pizza
Dessert
Beverage
Modify Food Info
Food ID
Name
Price
Category
clear
UPDATE
Food_Id | Name | Price | Category
2000 | Fish and Chips | 20.0 | Fastfood
2001 | Tandoori Chicken | 24.0 | Fastfood
2002 | Chicken Grill | 29.0 | Fastfood
2003 | Chicken Drumsticks | 25.0 | Fastfood
2004 | Butter Lobster | 24.0 | Fastfood
2005 | Tuna Stake | 25.0 | Fastfood
2006 | Onion Pasta | 25.0 | Fastfood
2007 | Fish Finger | 25.0 | Fastfood
2008 | Chicken Supreme | 30.0 | Fastfood
2009 | Whopper | 32.0 | Fastfood
2010 | Chicken Cheese De... | 34.0 | Fastfood
2011 | Mushroom Caramel | 37.0 | Fastfood
2012 | Sushi | 38.0 | Fastfood
2013 | Rabbit Roast | 40.0 | Fastfood
2014 | Half Pounder (Beef) | 53.0 | Fastfood


- Admin -
Id : 1001
Name : Shahed
Password : 1234
Contact no. : +8801521487951
Email : shahedkhan@mail.nwpu.edu.cn
Address : Chittagong
Gende... Male
Change Name
Change Password
Change ContactNo.
Change Email
Change Address

- Affiliated database –



ER diagram



II. Problem Faced while developing the app

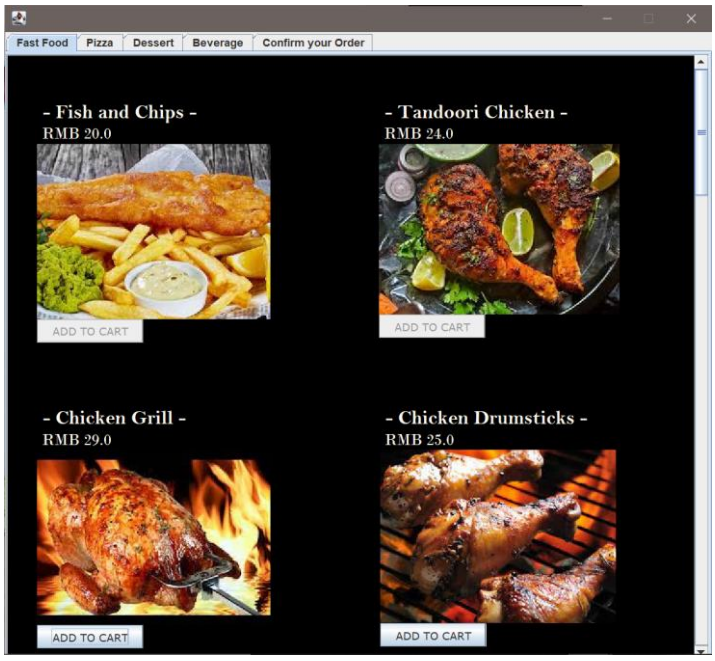
In this day and age of complete mobility, going out to eat is a hard option. But business and consumption have to go on. Thinking to tackle this Covid Pandemic era we have come out with our solution through this app. Users can access the previous reviews on app as well as visual representation of what they are going to get when they place orders as well.

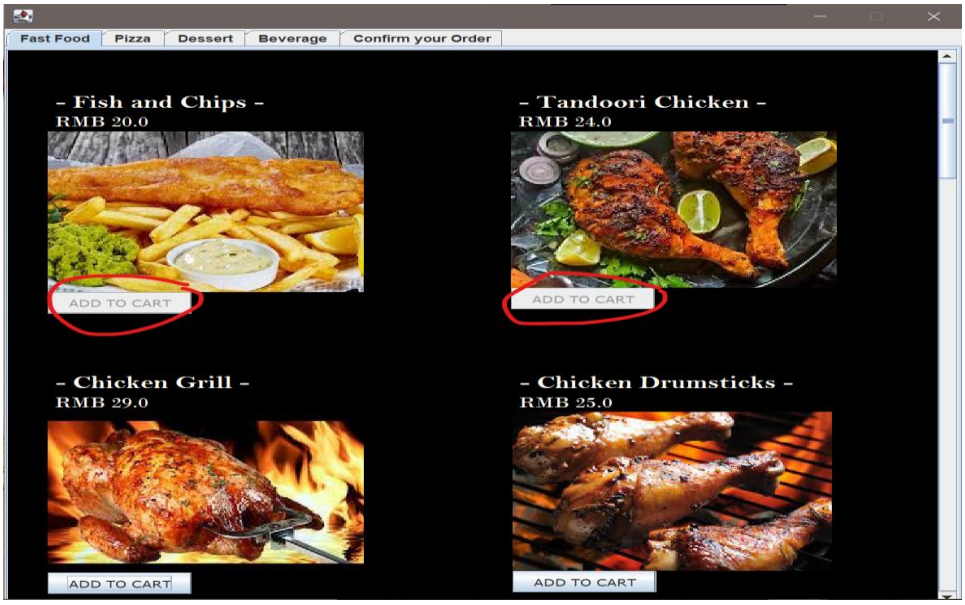
1)Trying to create a desktop core java app we faced problem with GUI: Java Swing, AWT (First we were implementing design part manually, later we found out that there is drag drop option for design, which is more convenient and time saving.)

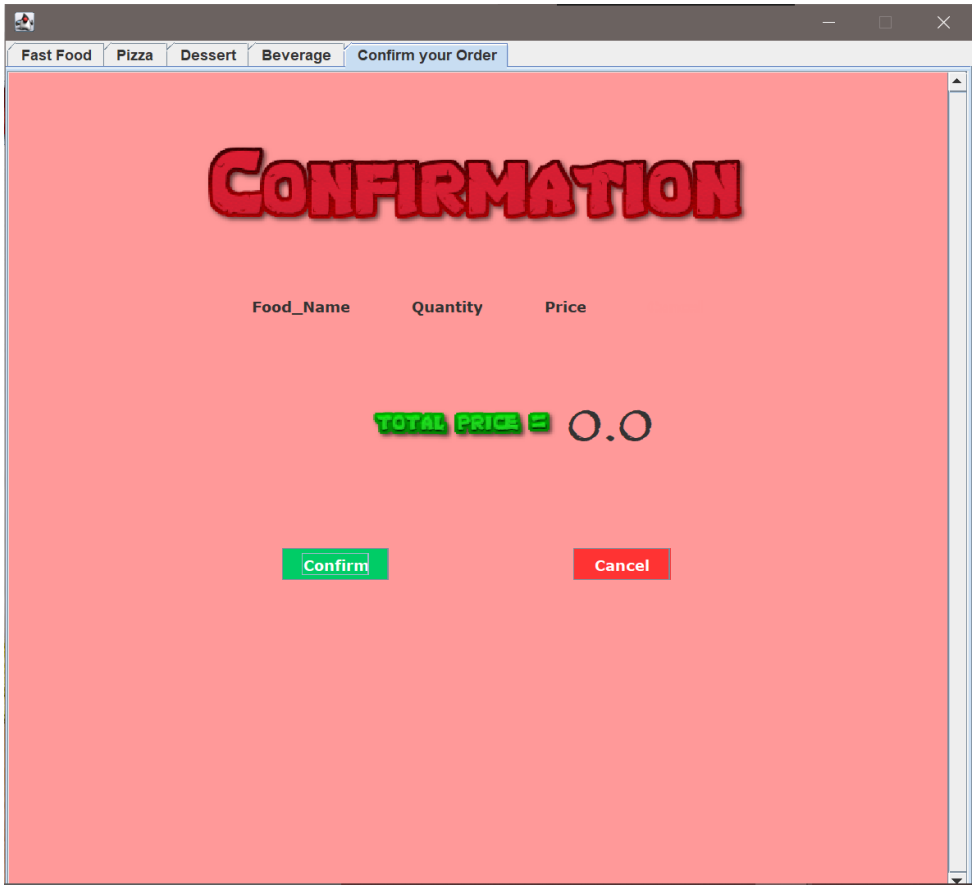
2)In our database we have imported all the images for our available foods. But it was hard to import and later we went with BLOB(64kb limit) as it was the best fit for our combination.

3)**Structure-**

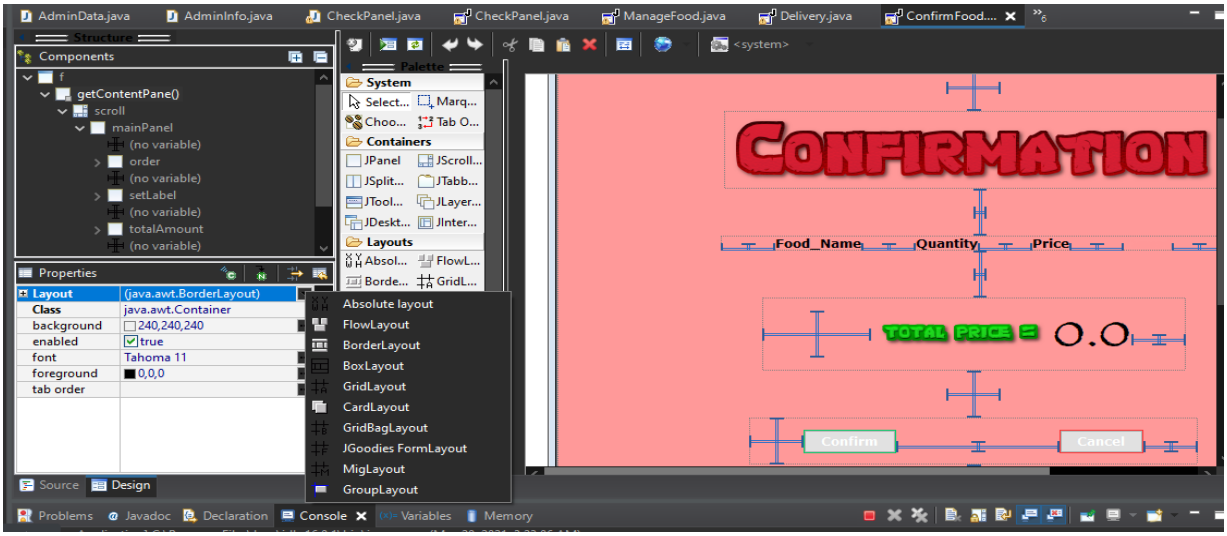
- We encountered some problems to implement structure according to our desired design (app interface and layout)
- We had issues creating admin page for the app and add data manipulation of data base through the app.
- We wanted to make our menu appear like this following picture, but we were having troubles with the synchronization(input) of information in order confirmation page.

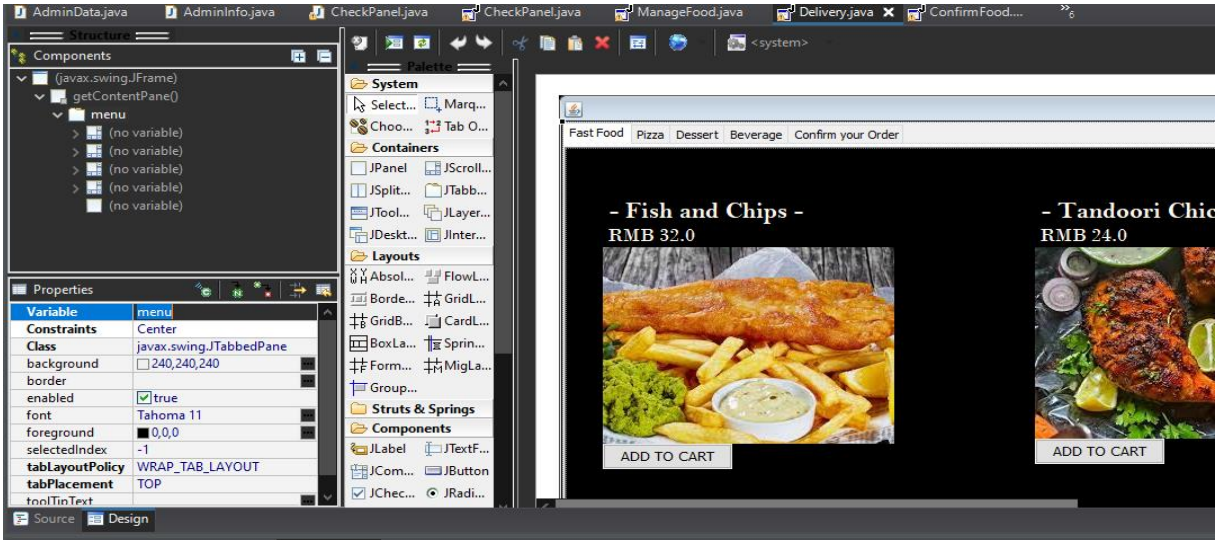


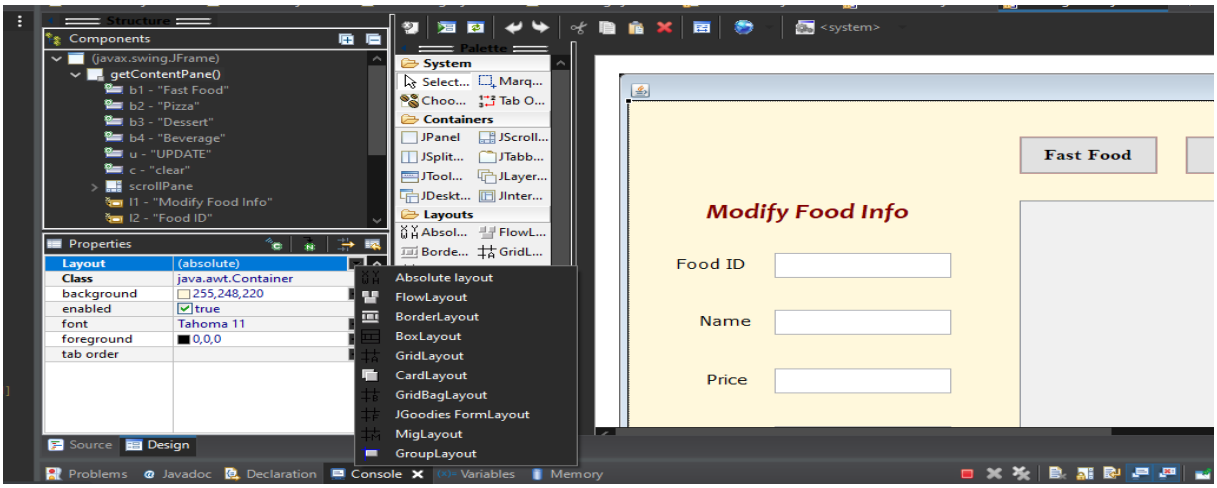

Fast Food
Pizza
Dessert
Beverage
Confirm your Order
- Fish and Chips -
RMB 20.0
ADD TO CART
- Tandoori Chicken -
RMB 24.0
ADD TO CART
- Chicken Grill -
RMB 29.0
ADD TO CART
- Chicken Drumsticks -
RMB 25.0
ADD TO CART


Fast Food
Pizza
Dessert
Beverage
Confirm your Order
CONFIRMATION
Food_Name
Quantity
Price
TOTAL PRICE = 0.0
Confirm
Cancel

- Configuring Layout was a bit hard (we didn't know priorly where to apply which one) as the implication was unknown to us and we had use different layouts for different pages to find the suitable ones we were looking for.







4)**Library-rs2ml.jar:** We had problems as we didn't exactly know which library to use for GUI table manipulation, later after an extensive stressful search session we ended up with rs2ml.jar that provided easy solution for us.

5) **Time Zone problem-** We have worked over 200 hours together for this group from 4 different time zones, while during the holiday it was possible to push 13-15 hours session each day, during weekdays it was really tough on to work more than 3-4 hours/day. Luckily through pushing each other we have managed to pull that off.

III. Problem Statement:

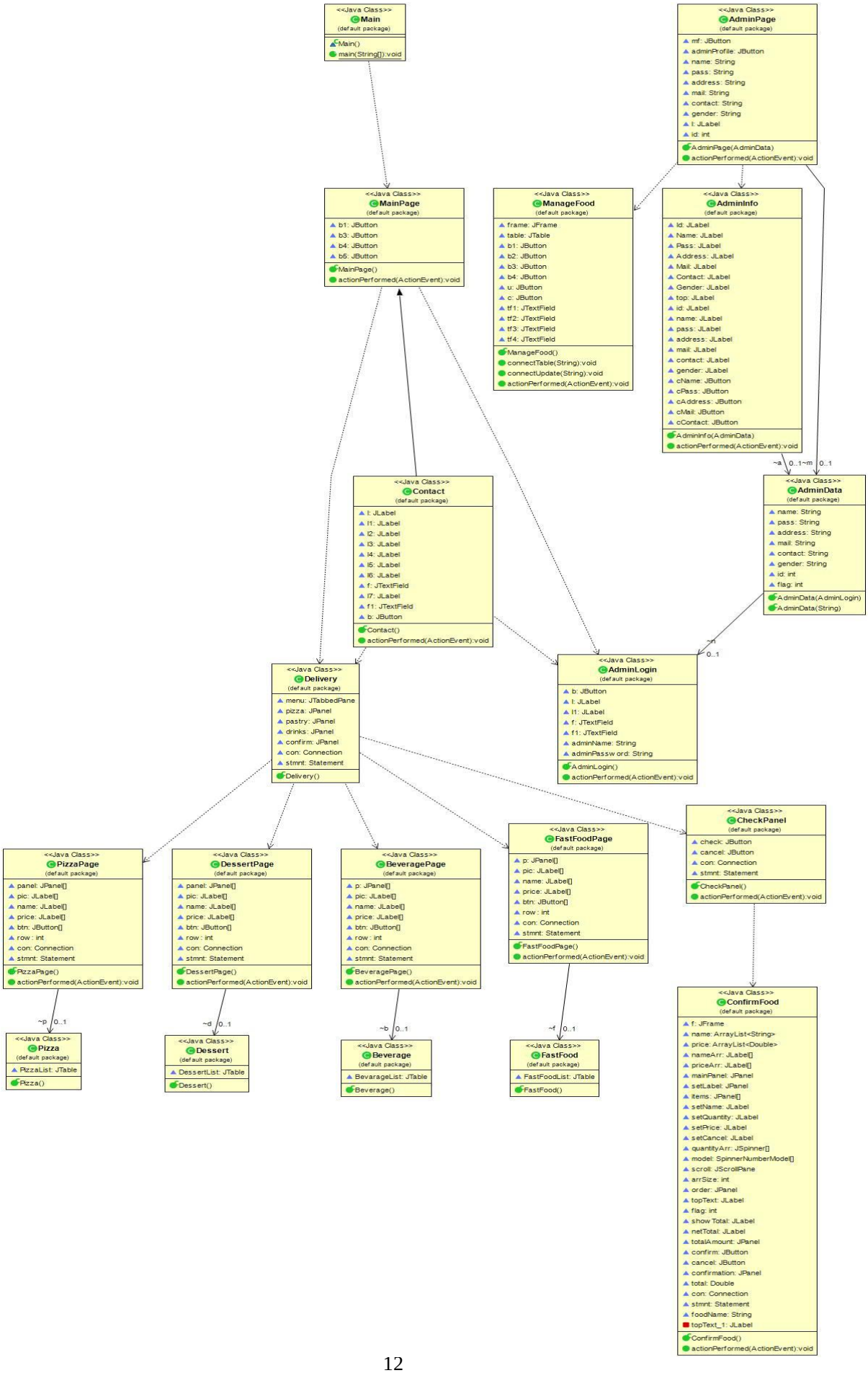
Through our app we have focused on increasing work efficiency of the existing systems. Our app is very user friendly, easy to use and navigate. Multiple admins can access the database according to the need and working time. Modifying food in the database can be done from the app itself without touching database. Same goes for admin credentials.

We have focused on a feasible fast purchasing system. Therefore, and user can do all he/she wants (select, drop, cancel order, change quantity, provide customer info) with 4/5 clicks and the order would be placed.
Our app is visually soothing and app interface is attractive creating a rich user experience for the customers.
So, the main issues we have addressed during designing and developing the app are quick, easy to use, user friendly and foremost accessible.

IV. Core Design

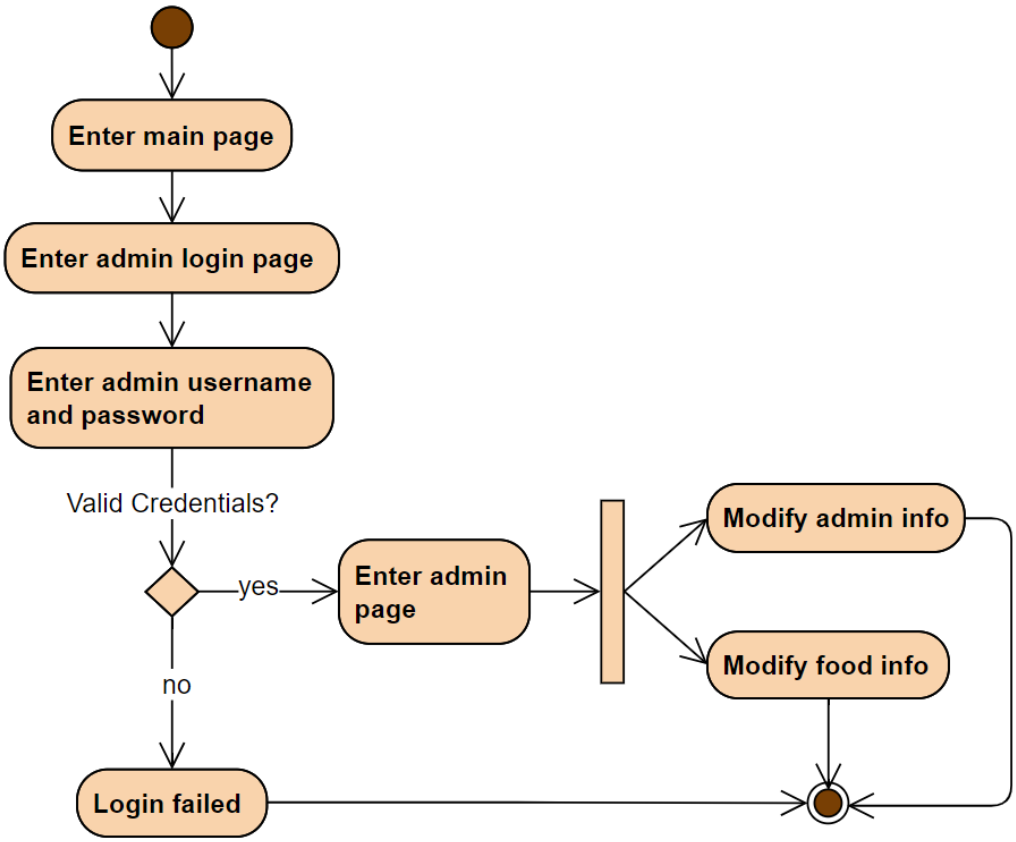
To create our application, we followed **Behavioral pattern** for controlling action events.
Besides that, we also used features like exception handling(to catch SQLException), collections(*Vector*, *ArrayList*), built in interface *ActionListener* and its *actionPerformed*() function for event handling
(When the action event occurs, that object's *actionPerformed* method is invoked.)

The way we have designed and implemented the design it can be best described with the UML diagrams. We have used 19 classes in total developing this application and the required functionalities by the teacher were implemented as a necessity to make this app.

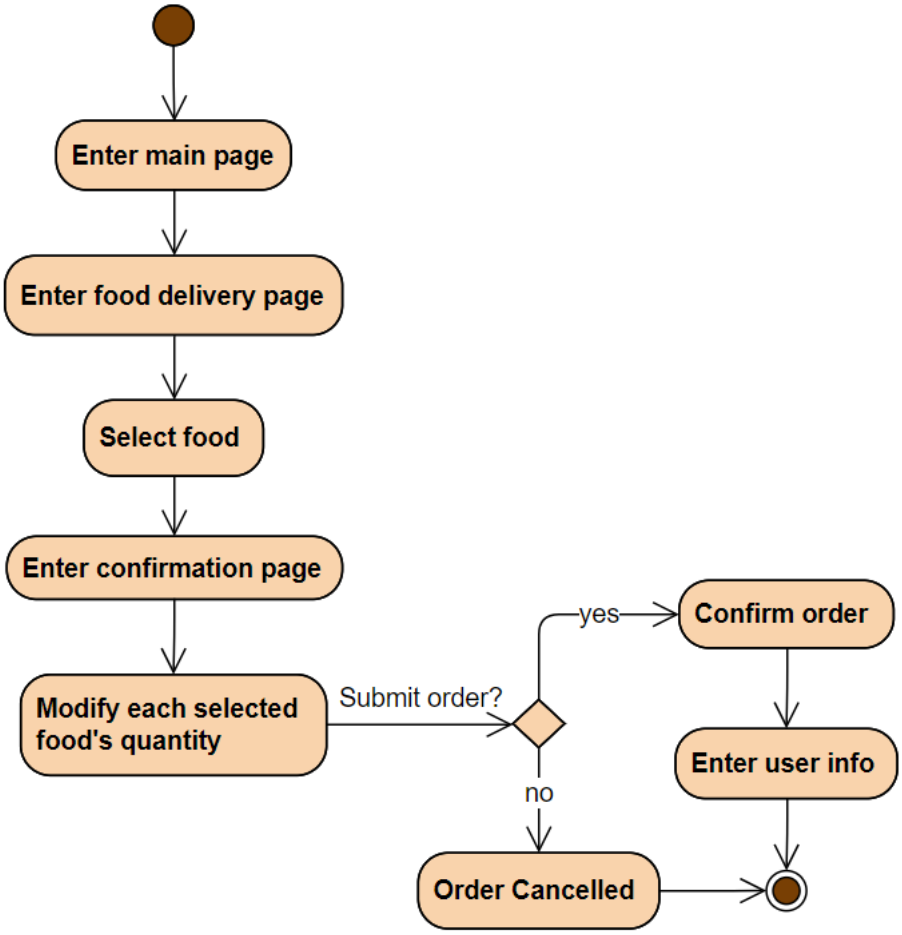

<<Java Class>>
Main
(default package)
Main()
main(String[]):void
<<Java Class>>
AdminPage
(default package)
mf: JButton
adminProfile: JButton
name: String
pass: String
address: String
mail: String
contact: String
gender: String
l: JLabel
id: int
AdminPage(AdminData)
actionPerformed(ActionEvent):void
<<Java Class>>
MainPage
(default package)
b1: JButton
b3: JButton
b4: JButton
b5: JButton
MainPage()
actionPerformed(ActionEvent):void
<<Java Class>>
ManageFood
(default package)
frame: JFrame
table: JTable
b1: JButton
b2: JButton
b3: JButton
b4: JButton
u: JButton
c: JButton
tf1: JTextField
tf2: JTextField
tf3: JTextField
tf4: JTextField
ManageFood()
connectTable(String):void
connectUpdate(String):void
actionPerformed(ActionEvent):void
<<Java Class>>
AdminInfo
(default package)
Id: JLabel
Name: JLabel
Pass: JLabel
Address: JLabel
Mail: JLabel
Contact: JLabel
Gender: JLabel
top: JLabel
id: JLabel
name: JLabel
pass: JLabel
address: JLabel
mail: JLabel
contact: JLabel
gender: JLabel
cName: JButton
cPass: JButton
cAddress: JButton
cMail: JButton
cContact: JButton
AdminInfo(AdminData)
actionPerformed(ActionEvent):void
~a
0..1~m
0..1
<<Java Class>>
Contact
(default package)
l: JLabel
l1: JLabel
l2: JLabel
l3: JLabel
l4: JLabel
l5: JLabel
l6: JLabel
f: JTextField
l7: JLabel
f1: JTextField
b: JButton
Contact()
actionPerformed(ActionEvent):void
<<Java Class>>
AdminData
(default package)
name: String
pass: String
address: String
mail: String
contact: String
gender: String
id: int
flag: int
AdminData(AdminLogin)
AdminData(String)
~n
0..1
<<Java Class>>
Delivery
(default package)
menu: JTabbedPane
pizza: JPanel
pastry: JPanel
drinks: JPanel
confirm: JPanel
con: Connection
stmnt: Statement
Delivery()
<<Java Class>>
AdminLogin
(default package)
b: JButton
l: JLabel
l1: JLabel
f: JTextField
f1: JTextField
adminName: String
adminPassw ord: String
AdminLogin()
actionPerformed(ActionEvent):void
<<Java Class>>
CheckPanel
(default package)
check: JButton
cancel: JButton
con: Connection
stmnt: Statement
CheckPanel()
actionPerformed(ActionEvent):void
<<Java Class>>
PizzaPage
(default package)
panel: JPanel[]
pic: JLabel[]
name: JLabel[]
price: JLabel[]
btn: JButton[]
row : int
con: Connection
stmnt: Statement
PizzaPage()
actionPerformed(ActionEvent):void
<<Java Class>>
DessertPage
(default package)
panel: JPanel[]
pic: JLabel[]
name: JLabel[]
price: JLabel[]
btn: JButton[]
row : int
con: Connection
stmnt: Statement
DessertPage()
actionPerformed(ActionEvent):void
<<Java Class>>
BeveragePage
(default package)
p: JPanel[]
pic: JLabel[]
name: JLabel[]
price: JLabel[]
btn: JButton[]
row : int
con: Connection
stmnt: Statement
BeveragePage()
actionPerformed(ActionEvent):void
<<Java Class>>
FastFoodPage
(default package)
p: JPanel[]
pic: JLabel[]
name: JLabel[]
price: JLabel[]
btn: JButton[]
row : int
con: Connection
stmnt: Statement
FastFoodPage()
actionPerformed(ActionEvent):void
~p
0..1
~d
0..1
~b
0..1
~f
0..1
<<Java Class>>
Pizza
(default package)
PizzaList: JTable
Pizza()
<<Java Class>>
Dessert
(default package)
DessertList: JTable
Dessert()
<<Java Class>>
Beverage
(default package)
BevarageList: JTable
Beverage()
<<Java Class>>
FastFood
(default package)
FastFoodList: JTable
FastFood()
<<Java Class>>
ConfirmFood
(default package)
f: JFrame
name: ArrayList<String>
price: ArrayList<Double>
nameArr: JLabel[]
priceArr: JLabel[]
mainPanel: JPanel
setLabel: JPanel
items: JPanel[]
setName: JLabel
setQuantity: JLabel
setPrice: JLabel
setCancel: JLabel
quantityArr: JSpinner[]
model: SpinnerNumberModel[]
scroll: JScrollPane
arrSize: int
order: JPanel
topText: JLabel
flag: int
show Total: JLabel
netTotal: JLabel
totalAmount: JPanel
confirm: JButton
cancel: JButton
confirmation: JPanel
total: Double
con: Connection
stmnt: Statement
foodName: String
topText_1: JLabel
ConfirmFood()
actionPerformed(ActionEvent):void

Above is given our app's class organization. From the diagram we can see different relations (Inheritance: contact us from main page, dependencies, directed association etc.). All the functions, structures used in each class can also be seen in the diagram.

Activity diagram (User): Shows all the steps users can follow to confirm/cancel order after selection.



Activity Diagram(admin): Shows all the steps admins need to follow to modify admin credential or update/modify food database.



V. Case Study:

This is our first time working on a project to develop desktop app with GUI (Java Swing, AWT) and therefore we really needed all the guidelines we could get. We trained us surfing oracle website and reading oracle documentations. Every time we got stuck, we tried taking help from professor's ppt.

VI. Contribution of Group Members:

Team Lead:
Implementing structure design, working on classes, interface design, Code review, UML diagram design, Bug fixing, creating data base, report writing, preparing ppt…etc.

Member1:
Implementing structure design, working on classes, interface design, Code review, UML diagram design, Bug fixing, creating data base, report writing, preparing ppt…etc.

Member2:
Implementing structure design, working on classes, interface design, Code review, UML diagram design, Bug fixing, creating data base, report writing, preparing ppt…etc.

Member3:
Implementing structure design, working on classes, interface design, Code review, UML diagram design, Bug fixing, creating data base, report writing, preparing ppt…etc.

VII. Conclusion

Working together in a team from 4 different time zones was really hard. But as we had been missing campus time interaction among friends, we really enjoyed the long-stayed nights working together. We have learned a lot from each other. Handled an intermediate level core java project, successfully implemented user defined classes, swing libraries etc. It was great making mistakes and learning from each other.

SOFTWARE TOOL SPECIFICATION:

- Eclipse IDE
- MySQL Workbench:
  MySQL Workbench is a unified visual tool for database architects, developers,
  and DBAs. MySQL Workbench provides data modeling, SQL

development, and
comprehensive administration tools for server configuration, user administration, and
much more. MySQL Workbench is available on Windows, Linux and Mac OS.

- Library-rs2ml.jar
- mysql-connector-java-8.0.22.jar
- Java JDK 16.0.1

HARDWARE REQUIREMENT:

- Processor: 2.00Ghz quad core processor or up (not ARM based)
- Disk Space: 50 MB or more
- 4000 MB or more memory module recommended

SOFTWARE REQUIREMENT:

- Operating System: For development we used windows 10(x64)
- Software: eclipse, MySQL workbench

Programming Language

- Java
- SQL