DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

FINAL REPORT

PHASE-III

FOOD ORDERING SYSTEM

By

Sadullah CİHAN

Özcan ELMACI

Onur İLTAŞ

June, 2021

İZMİR

CHAPTER ONE

INTRODUCTION

The report is the Food Ordering System project progress report of Dokuz Eylul University Department of Computer Engineering. The project period is one semester. The objective of this project is to develop a food ordering system using object-oriented concepts.

The food ordering system is mainly a service that contains contracted restaurants and coffee shops with their menus inside and allows users to order food and drinks to their located address without going outside.

The pandemic has risen to a whole other level this year and Turkey is currently the country with the most daily COVID-19 cases. In the current situation of the country, people older than 65 and children younger than 20 can not go outside without special cases permitted by local authorities. This becomes a problem for families that has old people and children to eat outside.

There are currently two main food ordering systems the operates actively in Turkey. But some problems affect both the consumers and the businesses in these systems. In one of them, the commission fee to sign a contract as a restaurant is too high so businesses increase the price of their food products and this effects the consumers' budget to buy more products. As a domino effect, the consumers do not want to risk trying new restaurants because of the high prices, instead, they mostly choose corporate restaurants. This leads to low endorsement and almost little to none profit to Turkish tradesmen. Due to this, businesses slowly gets pushed to bankruptcy and little by little, the Turkish economy weakens.

In the other food system, due to operational and software mistakes, the main application keeps crashing and this makes the users(consumers) unhappy because they can't order food easily or their orders don't show up on time. The unhappiness leads to low consumer demand on the app and thus, the contracted businesses don't gain profit from this app. The result will be the same as the other app in the close future.

The app developed by our team will give Turkish tradesmen more space for-profit and the R&D department will constantly push the app's limits to block it from crashing.
Our target users are people who are against going outside to get food during the pandemic. Our goal is to help people eat food safe from the virus and buy them for cheaper prices than our rivals.

CHAPTER TWO

REQUIREMENTS

1) CLASSES

The required classes are as follows:

- **Address**
- **Phone**
- **User**
    - String name
    - Phone phone
    - Address address
    - String email
    - String password
    - DateTime birthday
    - Int creditCardId
    - Int [] oldOrderId
- **Food**
    - String name
    - Int foodID

    - String kitchenType
    - Double price

- **Cart**
    - Food[] order

- **Order**
    - Food[] order
    - DateTime deliveryTime
    - String discountCode
    - String orderNote
    - Int ID

- **Restaurant**
    - String name
    - Int ID
    - Int cityID
    - Address address
    - Phone phone
    - Food[] menu
    - Inventory [] inventories

- **Inventory**
    - int foodId
    - int unitInStock
- **Payment**
    - Int paymentTypeID
- **Authorization**
    - Login()
    - Register()

2) METHODS

- **readTextFile(String fileName)** ➔ reads a text file.
- **creatingRestaurants()** ➔ assigns the values that were read from text files.
- **filter(int selection)** ➔ applies the filter was chosen by the user.
- **creatingCities()** ➔ assigns the values that were read from text files.
- **creatingUsers()** ➔ assigns the values that were read from text files.

- **creatingFoods()** ➔ assigns the values that were read from text files.

- **addToCart(Food food)** ➔ adds the food to the cart.

- **deleteFromCart(int foodId)** ➔ deletes the food from the cart.

- **dischargeCart()** ➔ empties the cart.

- **changeTheAmountOfProduct()** ➔ increases or decreases the amount of the added product to the cart.

- **confirmCart()** ➔ if the user is not a member or not log in before, directs the user to the **" Login-Registration Page"**. Otherwise, directs the user to the **"Payment Page"**.

- **payment(int paymentTypeID)** ➔ according to the parameter, directs the other necessary payment options.

- **confirmCreditCard()** ➔ controls the credit card in **"Payment Page"** and returns true or false (boolean type).

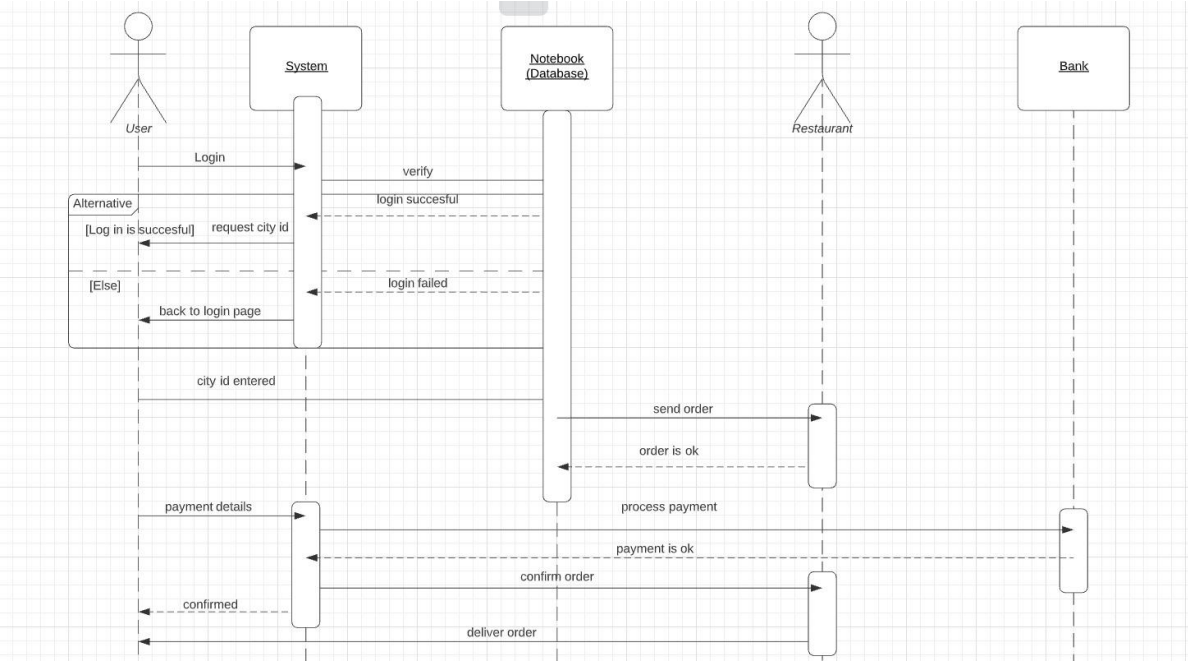
- **checkCouponCode()** ➔ controls the coupon code.

- When the user enters the system for the first time, the city selection is asked. It will be solved by keeping the choice of the user. Then, the choice will be assigned to **"getRestaurantByCityId(int cityId)"**. After that, the method will receive the city and return the restaurant list in that city. The list type is **"Restaurant"**.

CHAPTER THREE

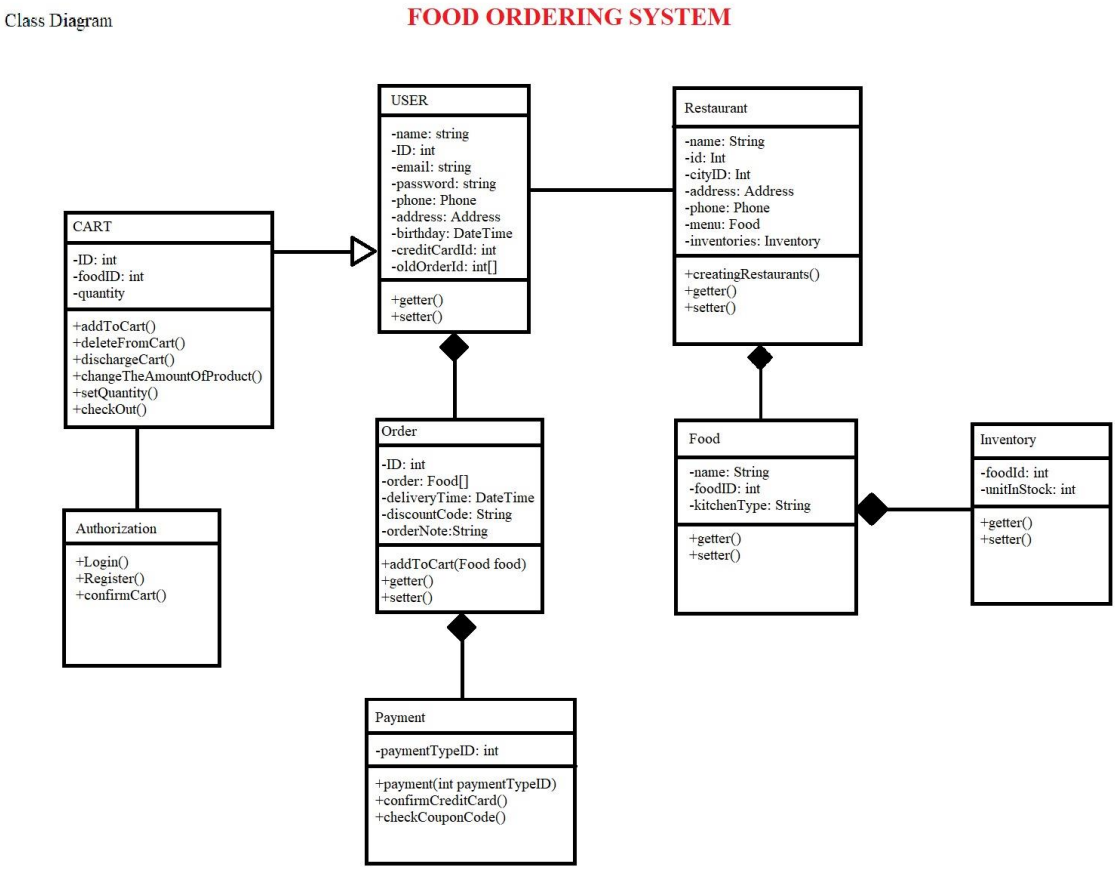
UML DIAGRAMS

1) SEQUENCE DIAGRAM



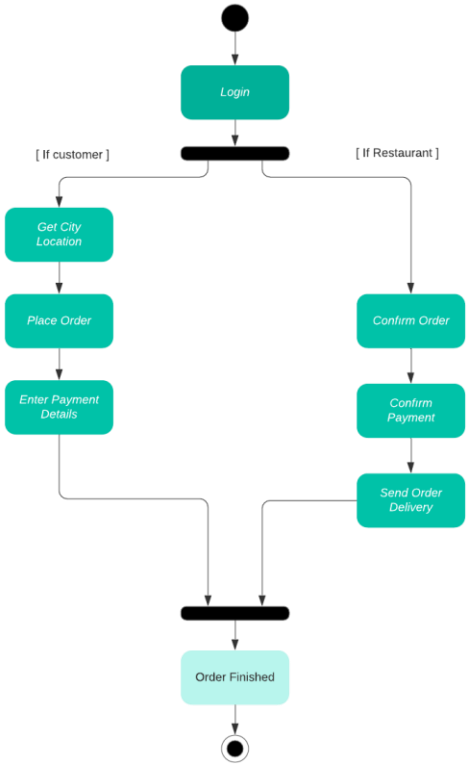
As you see above, firstly user should log in to the system. Then the system has to verify the login and returns two options. One of them is a successful return and the other is failed return. If the return is failed, the system redirects you to the login page. If the return is successful, the system requests a city id. The city id is entered and the restaurants are taken from the database(notebook-text files). After that, the order is sent to the restaurant. If the restaurant accepts the order, the user is directed to the payment system. Then payment operations occur.

2) CLASS DIAGRAM



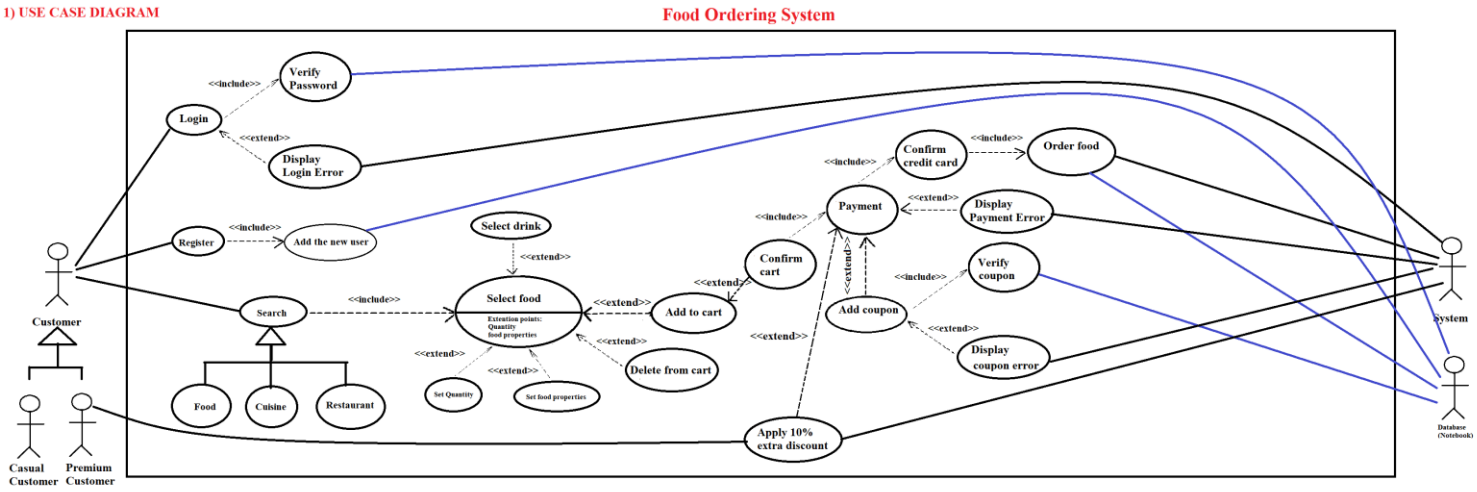
We have a Cart class and by inheritance, there are the User and Restaurant classes. The user has a sub-class named Order and it is the main class for placing an order. Under the order, there belongs the Payment class where all the money business is done. The Restaurant class has a sub-class named Food where their sold goods belong. To count how much they can make or deliver, We have the Inventory sub-class under Food.

3) ACTIVITY DIAGRAM



The users have to firstly log in to the app. If the user is a customer or a restaurant, the process changes but ends up with the same result. When a customer logs in, they first have to select their city. After that, they place an order from their desired restaurant. At the same time when the customer orders from a restaurant, the logged-in restaurant that takes the order confirm the order. Then, the user enters their payment details and the restaurant confirms the payment comes to them. Lastly, the restaurant sends the order to a delivery person and notifies them from the app that they have sent the order. After the order is delivered, the order is finished and the process ends.

4) USE CASE DIAGRAM



This is a use case diagram. As you can see above, there are two types of customers; casual customers, the premium customers. A customer can log in or register at the beginning of the program. Then, a customer can search according to food name, cuisine type, or restaurant name. After searching the restaurant, the user can select foods. Also, he or she can add selected foods to the cart. Then, a customer has to select a payment type. If online payment is selected, the credit card information is checked. Then, an order object is created. If payment at the door option is selected, an order object is created like before. If the customer has a coupon code or has a premium account, the discount is applied and the bill is reduced.

CHAPTER FOUR

IMPLEMENTATION

There is an interface called “Customer”. It’s the factory pattern. It specifies the user if it is premium or not. These are the classes that are used in the source code:

1. Authorization
2. Cart
3. Customer
4. Food
5. Login
6. MainMenu
7. Order
8. Payment
9. PaymentFrame
10. PremiumUser
11. Register
12. Restaurant
13. TestFoodOrderingSystem
14. User

The following classes have been tested within Java.Swing framework.

1. Login
2. MainMenu
3. PaymentFrame
4. Register

Other methods have been used and tested in the Java console. The program should be started from the Login class or the Register class. The explanation of codes indicated in the source code. Several changes have been done in some classes.

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

All in all, the design of GUI has been learned by using Java Swing Framework. Several OOP techniques have been applied. Customer interface merged User class and Premium User class. Also, the reference property of the interface has been implemented. Premium User class inherits the User class.

The source code is appropriate for implementing future works. It can be improved by using several changes. Maintenance can be applied.

CHAPTER FIVE

REFERENCES

1. **"N.D.", "HTTPS://HELP.ECLIPSE.ORG/2021-06/INDEX.JSP?TOPIC=%2FORG.ECLIPSE.WB.DOC.USER%2FHTML%2FINDEX.HTML", RETRIVED ON 21/06/2021**
2. **"N.D.", "HTTPS://WWW.JAVA67.COM/2015/07/HOW-TO-APPEND-TEXT-TO-EXISTING-FILE-IN-JAVA-EXAMPLE.HTML", RETRIVED ON 21/06/2021**
3. **"N.D.", "HTTPS://STACKOVERFLOW.COM/QUESTIONS/19940653/GETTING-THE-ID-FROM-A-ROW/19941344", RETRIVED ON 21/06/2021**
4. **"N.D.", "HTTPS://GITLAB.COM/WSENG92/FOOD-ORDERING-SYSTEM/-/BLOB/MASTER/SRC/GUI/FOODMENU.JAVA", RETRIVED ON 21/06/2021**
5. **"N.D.", "HTTPS://STACKOVERFLOW.COM/QUESTIONS/22066387/HOW-TO-SEARCH-AN-ELEMENT-IN-A-JTABLE-JAVA", RETRIVED ON 21/06/2021**
6. **"N.D.", "HTTPS://STACKOVERFLOW.COM/QUESTIONS/9919230/DISABLE-USER-EDIT-IN-JTABLE", RETRIVED ON 21/06/2021**