DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

Food Ordering System

by

2018510089 Onur İltaş

2018510079 Bülent Çivan

CHAPTER ONE

INTRODUCTION

This is the phase 1(One) project progress report of the Food Ordering System project developed by Bülent Çivan and Onur İltaş for Dokuz Eylül University Department of Computer Engineering. The period of the project 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 menus from contracted restaurants and coffee shops and allows the users to order food and drinks to their specified address without going to the restaurants themselves.

With the advancing technology, many solutions to make peoples lives easier are being provided even as this report is written. Within these solutions, there are ordering service providers that are already operating in Turkey. But some problemss effect both consumers and businesses in these systems. In one of them, the commision 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 get pushed to bankruptcy and little by little, the Turkish economy weakens.

In another food system, due to operational and software mistakes, the main application sometimes crashes and the B2B(Business to Business) part of it has many bugs. This causes orders from the consumers to not be visible to restaurants or show up late. When this happens the consumers become upset 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 unable to order food and drinks take-away or sit in the restaurants.Our goal is to help people order food easier and let businesses reach to more customers without risking bankruptcy.

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.

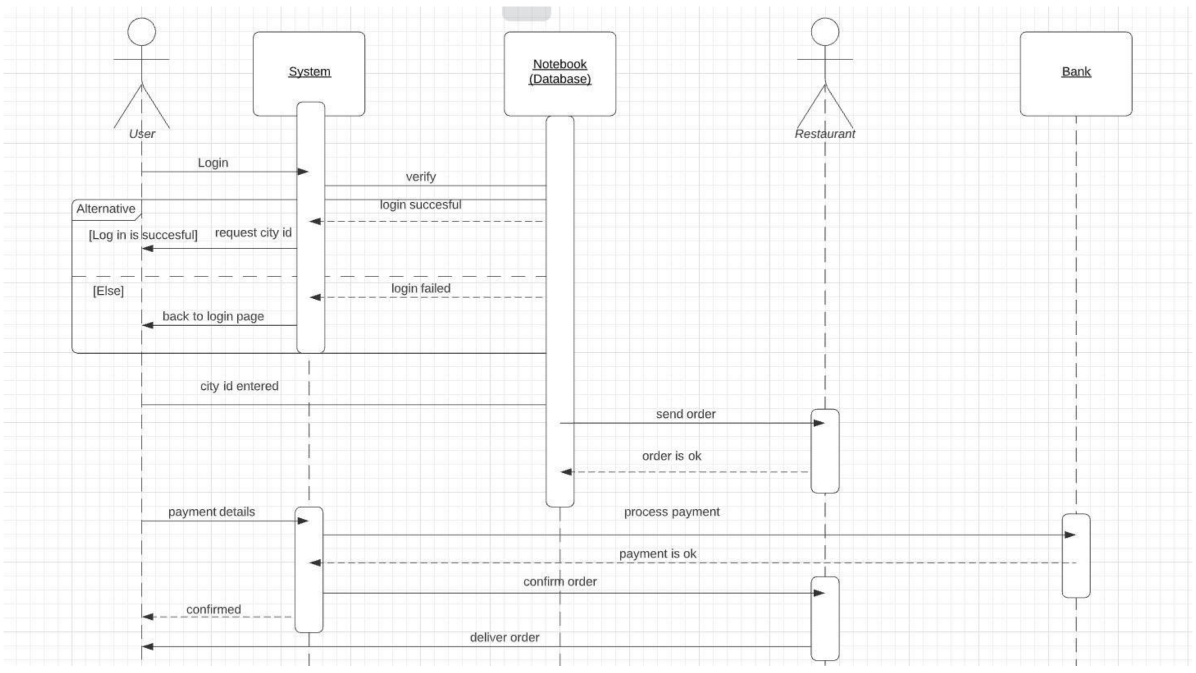
- **confirmCreditCard()** ➔ controls the credit card in **"Payment Page"** and returns true or false (boolean type).

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

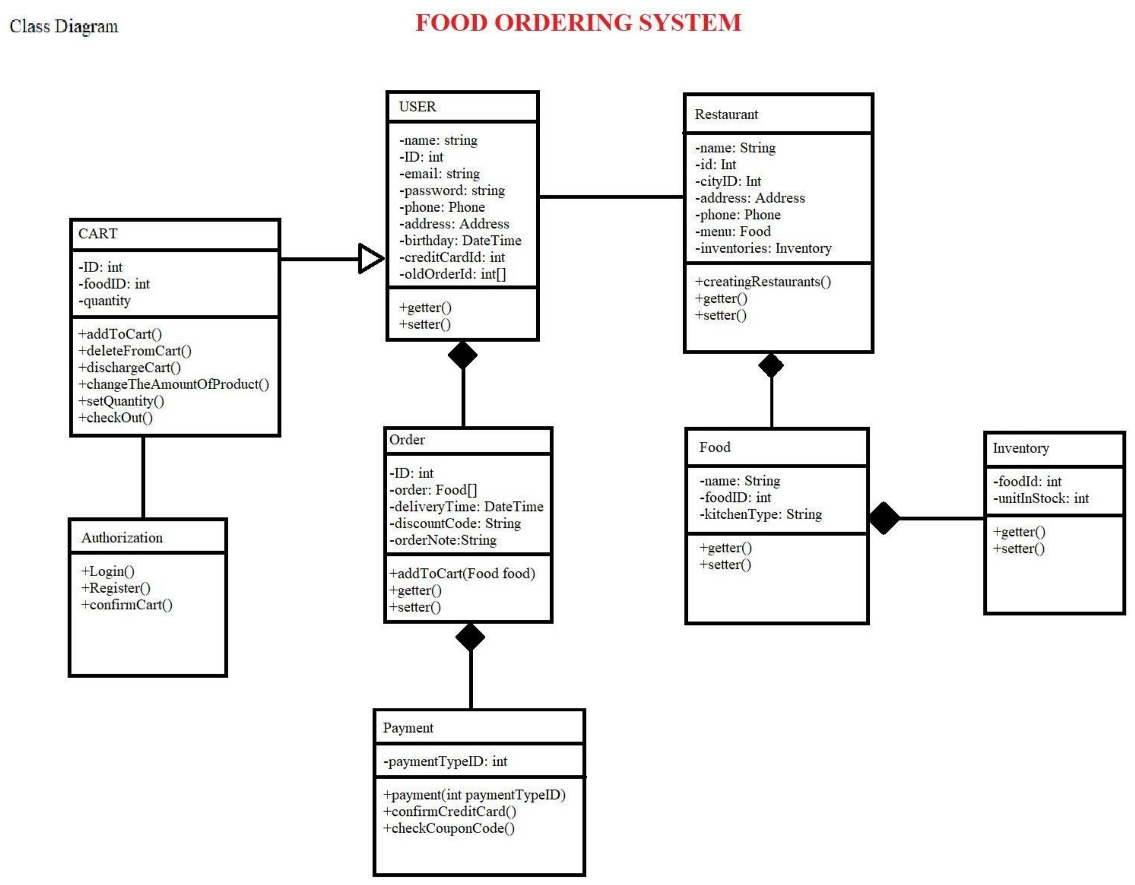
- When the user enters the system for the first time, the city selection is asked. It will be solved by keeping the choice of the user. Then, the choice will be assigned to **"getRestaurantByCityId(int cityId)"**. After that, the method will receive the city and return the restaurant list in that city.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

CHAPTER THREE

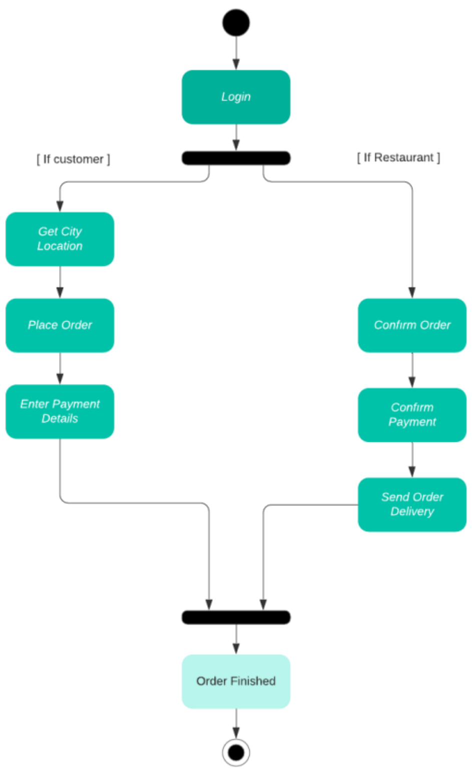
UML DIAGRAMS



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 successful return and 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 occurs.



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 selled goods belong. To count how much they can make or deliver, We have the Inventory sub-class under Food.



The users have to firstly login 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 confirms the order. Then, the user enters their payment details and the restaurant confirms the payment comes to them. Lastly, the restaurant sends the order with a delivery person and notifies from the app that they have sent the order. After the order is delivered, the order is finished and the process ends.

CHAPTER FOUR
IMPLEMENTATION

An understandable and user-friendly design has been chosen for the interface of this project. The team tried to use every property of Object-Oriented Programming while developing this project. For better understanding of interfaces Apache Netbeans was chosen as the IDE.

Completed Tasks

The necessary classes has been created. Some classes implements interfaces and other classes using inheritance and abstraction.

The program opens with the login interface. The users can login with their login credentials or if they don’t have an account, they can register as a new user after clicking on the register button. After the user enters the register interface, the program wants the user to give all credentials for it to save the user in a local memory. If the user already has an account in the local memory, they can easily continue using the program after logging in. Otherwise the program alerts the user if wrong information is given. In the main frame, the user can pick products from a restaurant list and sort the products by multiple ways. A cart is located on the same frame as the restaurant list. The users can see which products they have chosen and see the total price of their cart. They can clear their cart or access the payment page by clicking the buttons next to the cart. In the last step of this ordering process, the user can choose whether to pay by card or cash and if wanted, an order note can be written to give specific details about the order to restaurants. If the user wants to change the order, there is a back button located on the frame so they can start their order with a clean cart.

Incompleted Tasks

Credit card information can not be confirmed so the user can’t pay online by using the program.

Products in cart can not be deleted one by one but it can be cleared completely.

Coupon codes can not be activated because the implementation of coupon codes couldn’t be developed on time.

Because there are a small number of inputs, the team didn’t work on different cities.

CHAPTER FIVE
CONCLUSION AND FUTURE WORKS

Although, we could not do everything that we wanted to do at the beginning of the project, we are happy to have a decent working project. Working as a group while taking different lessons was a bit difficult but we handled it very good. Working on a Java application using Apache Netbeans was an improving experience for us. Now we realize that we can work on different IDEs while learning their own interface as a group.