

FOOD DELIVERY MANAGEMENT SYSTEM

DOCUMENTATION - ASSIGNMENT 4

Programming Techniques

STUDENT: DOMȘA-STÎNĂ ANDA

GROUP: 30423

# TABLE OF CONTENTS

[1. Objectives 4](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analizing the requirements 4](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Design 7](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementation 11](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Results 14](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Conclusions 14](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliography 15](file:///C:\Users\User\Downloads\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# Objectives

1. Main objective

*Project specification:*

Design and implement a food delivery management system for a catering company. The client can order products from the company’s menu. The system should have three types of users that log in using a username and a password: administrator, regular employee, and client.

The purpose of the project is the development of a food delivery application, in which some products are given, and there are three actors that can perform certain operations regarding those products.

The administrator can:

• Import the initial set of products which will populate the menu from a .csv file.

• Manage the products from the menu: add/delete/modify products and create new products composed of several products from the menu (an example of composed product could be named “daily menu 1” composed of a soup, a steak, a garnish, and a dessert).

• Generate reports about the performed orders considering the following criteria:

o time interval of the orders – a report should be generated with the orders performed between a given start hour and a given end hour regardless the date.

o the products ordered more than a specified number of times so far.

o the clients that have ordered more than a specified number of times so far and the value of the order was higher than a specified amount.

o the products ordered within a specified day with the number of times they have been ordered.

The client can:

• Register and use the registered username and password to log in within the system.

• View the list of products from the menu.

• Search for products based on one or multiple criteria such as keyword, rating, number of calories/proteins/fats/sodium/price.

• Create an order consisting of several products – for each order the date and time will be persisted and a bill will be generated that will list the ordered products and the total price of the order. The employee is notified each time a new order is performed by a client so that it can prepare the delivery of the ordered dishes.

Javadocs were also used for this project, because they describe what we do in a method. Inside are some example tags: @param param, and then specify because it is the param parameter in that method.

Lambda expressions are a new and important feature in Java that has been included in Java SE 8. They provide a clear and concise way to represent an interface of a method using an expression. They are very useful in the Collections library and help to repeat, filter and extract data from the collection.

*Analyze the problem and identify requirements*

The first step in implementing the program is to understand the problem, to identify the tasks it should perform and to model its functionalities using an object-oriented approach. This will be forward detailed in chapter 2.

*Design the simulation application*

The next step in solving this project’s requirements is to design the queues management application by dividing it into smaller components using structural diagrams. This will be further discussed in chapter 3.

*Implement the simulation application*

Now that the requirements are understood and modeled, the next step is to implement the actual java program by writing code for the needed classes in Intellij. This step will be covered in chapter 4.

*Test the simulation application*

The last step of the project development is to test the application, a process which I have described in chapter 5 of the documentation.

# Analyzing the requirements

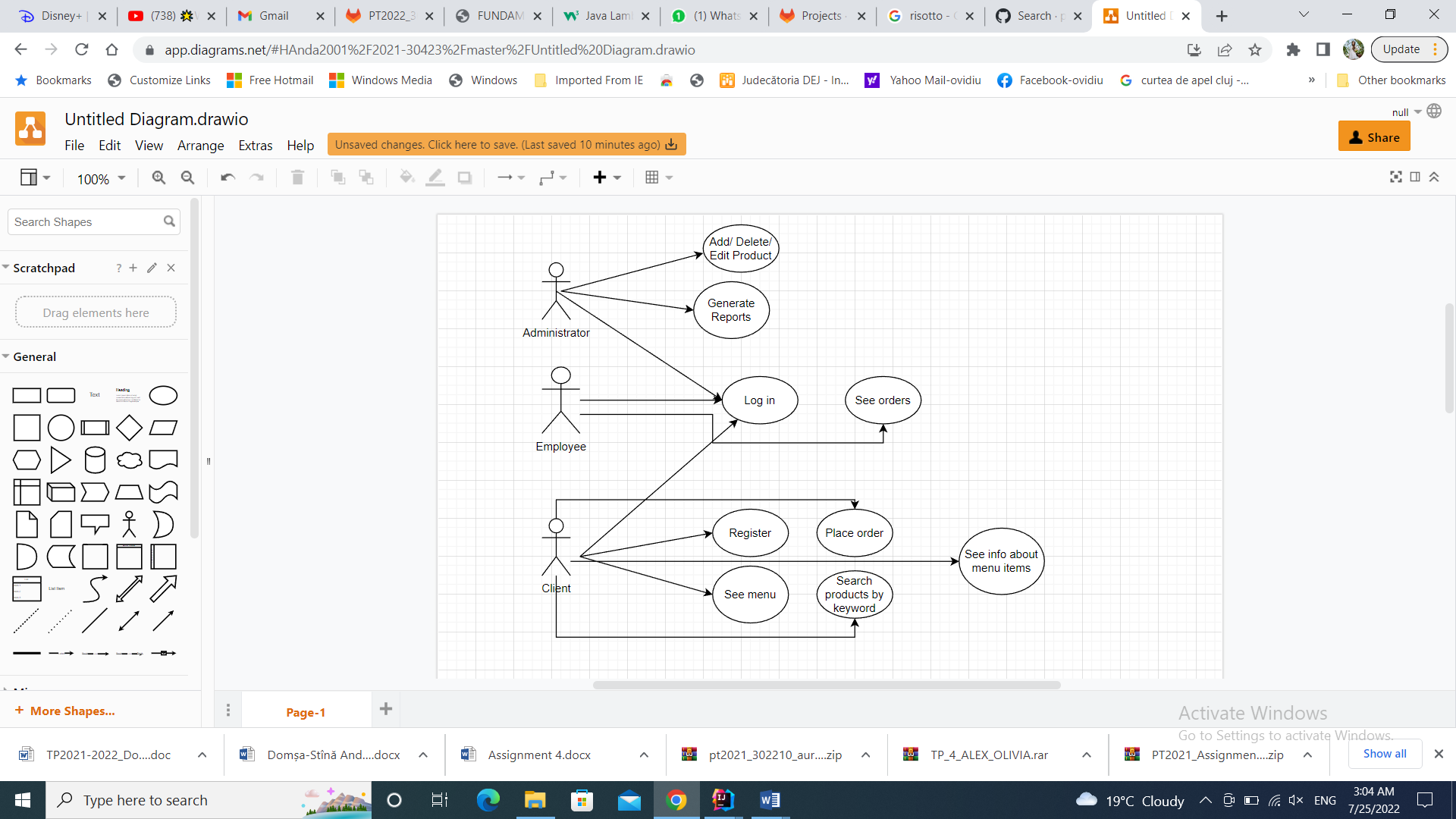
The first step in implementing the program is to understand the problem, to identify the tasks it should perform and to model its functionalities using an object-oriented approach.

The requirement of the project is to implement an application that simulates a food delivery application.

However, the analysis of this problem is not an easy one, because for this application to work, it is based on the import of products from the products.csv file, which is not very easy. This is a complex application that requires a lot of attention, knowledge and even imagination. The way in which each customer, order or product is must be very well thought out so that there are no problems during the implementation.

The user interacting can have one of the following three roles: client, administrator or employee. They each have different ways to interact with the program, so they will present different possible use cases.

**Use cases:**



Use case example: **Create a new account**

Actor: The client

Scenario:

The user has to introduce a username and a password and click the button “LogIn”. If the credidentials were before registered in the system, the application will open the client’s specific interface, so he can see the menu and place an order. In case the user doesn’t have an account yet, he must first register to create a new account and then log into the application.

Use case example: **Log in as administrator or employee**

Actor: The administrator or the employee

Scenario:

The user has to introduce a username and a password and click the button “LogIn”. If the data wes before registered in the system, the application will open the admin or employee specific interface, so the admin can make changes in the menu or generate reports and the employee can see the orders that were before placed. In case the username or password were wrongly introduced, an error message will be displayed to inform the user that the data is not correct.

Use case example: **Deleting a product as admin**

Actor: The administrator

Scenario:

The admin the user presses the delete button of the previously selected product form the table. The product is actually deleted. To see the product being gone, the user must press the “Refresh Menu” button.

Use case example: **Creating a product as admin**

Actor: The administrator

Scenario:

The admin presses the create base button and another window opens. The admin must now introduce the data for the product he wants to add to the menu and then press the add button. To see the product being added in the menu, the user must press the “Refresh Menu” button.

Use case example **Editing a product as admin**

Actor: The administrator

Scenario:

The admin the user presses the edit button after he selected a product from the table and another window opens. The admin must now introduce the data he wants to change for the product he selected from the menu and then press the edit button. To see the product being edited in the menu, the user must press the “Refresh Menu” button.

# Design

The next step in solving this project’s requirements is to design the queues simulator by dividing it into smaller components using structural diagrams. In this chapter, we will also present the OOP design of the application, the UML diagram, as well as the data structures and algorithms used.

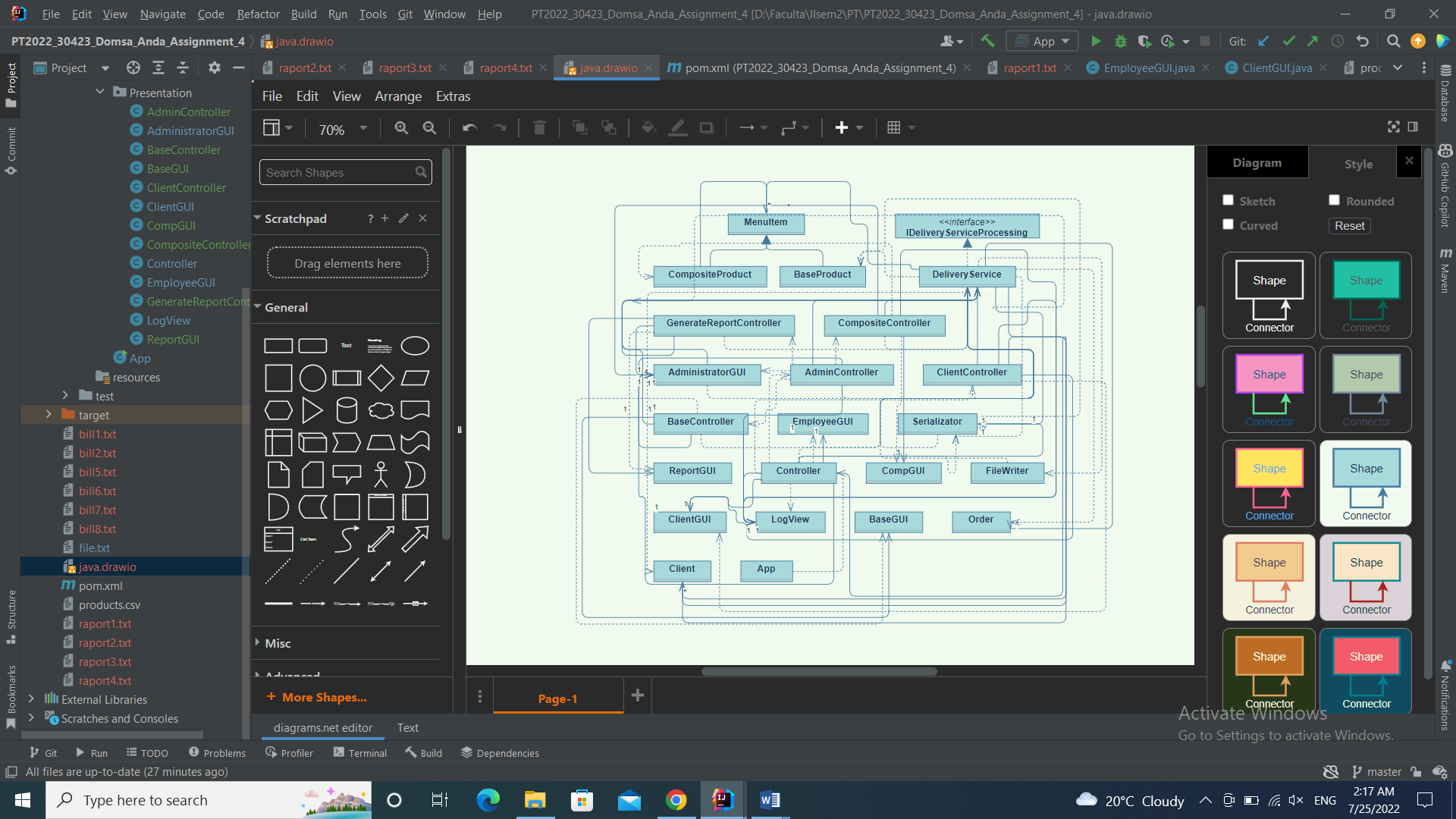
**Packages:**

The application is organized in three packages: BussinessLogic, Presentation and DataAccess.

The BussinessLogic package contains several classes such as: Client which contains the username and password fields and where the password and username of the clients are saved.

The DataAcces package contains the classes that are of importance for the simulation, like “Serializator” or “FileWriter”.

The Presentation package contains the graphical user interfaces and the controller associated.



# Implementation

I started the implementation by working on the “Client” and “Serializator” classes.

Package: BussinessLogic

**Classes:**

* *BaseProduct –* This class contains the 7 fields that are originally given: title, rating, calories, protein, fat, sodium, price, and the super constructor for them. It also extends the class MenuItem which is the patern for the products added in the menu.
* *Client -* This class models a serializable object, being used to keep the information of a client and to help to keep the clients in a list for further use. The client object is described using the parameters: username, password and client id. For the other actors needed in this simulation I didn’t need to create separate classes, as you can’t add more administrators or employees, but the clients are needed for placing and describing the orders.
* *CompositeProduct –* This class, as does the BaseProduct class, contains the 7 fields that are originally given: title, rating, calories, protein, fat, sodium, price, and the super constructor for them. A composite product represents a product that is composed of a list of multiple base products, for example a daily menu containing a soup and a main course. When a composite product is added the values of the parameters is recalculated. It also extends the class MenuItem which is the patern for the products added in the menu.
* *IDeliveryServiceProcessing* – This is an interface that represents a pattern for the main functionalities of the application*.* It contains the methods that must be implemented in the Deliveryservice class and also the java doc pre and post conditions.
* *Deliveryservice* – This class implements the *IDeliveryServiceProcessing* interface and it implements the most important methods that are used for the good working flow of the application. Here are implemented for example the “Register” and “LogIn” methods.
* *MenuItem –* This is an abstract class that represents the pattern for the base product and the composite product. It containg the getters and setters for the menu item’s attributes.
* *Order -* This class implements the Serializable interface. This class is used to model the order and keep the item list, date, and client id for each order.

Package: DataAccess

**Classes:**

* *FileWriter -* Interface containing the addTask method. It is implemented by both the ShortestQueueStrategy class and the TimeStrategy Class.
* *Serializator –* Thisn is the class that links the bussniss logic classes to the GUI classes and sets the action listeners for the user interface.

Package: Presentation

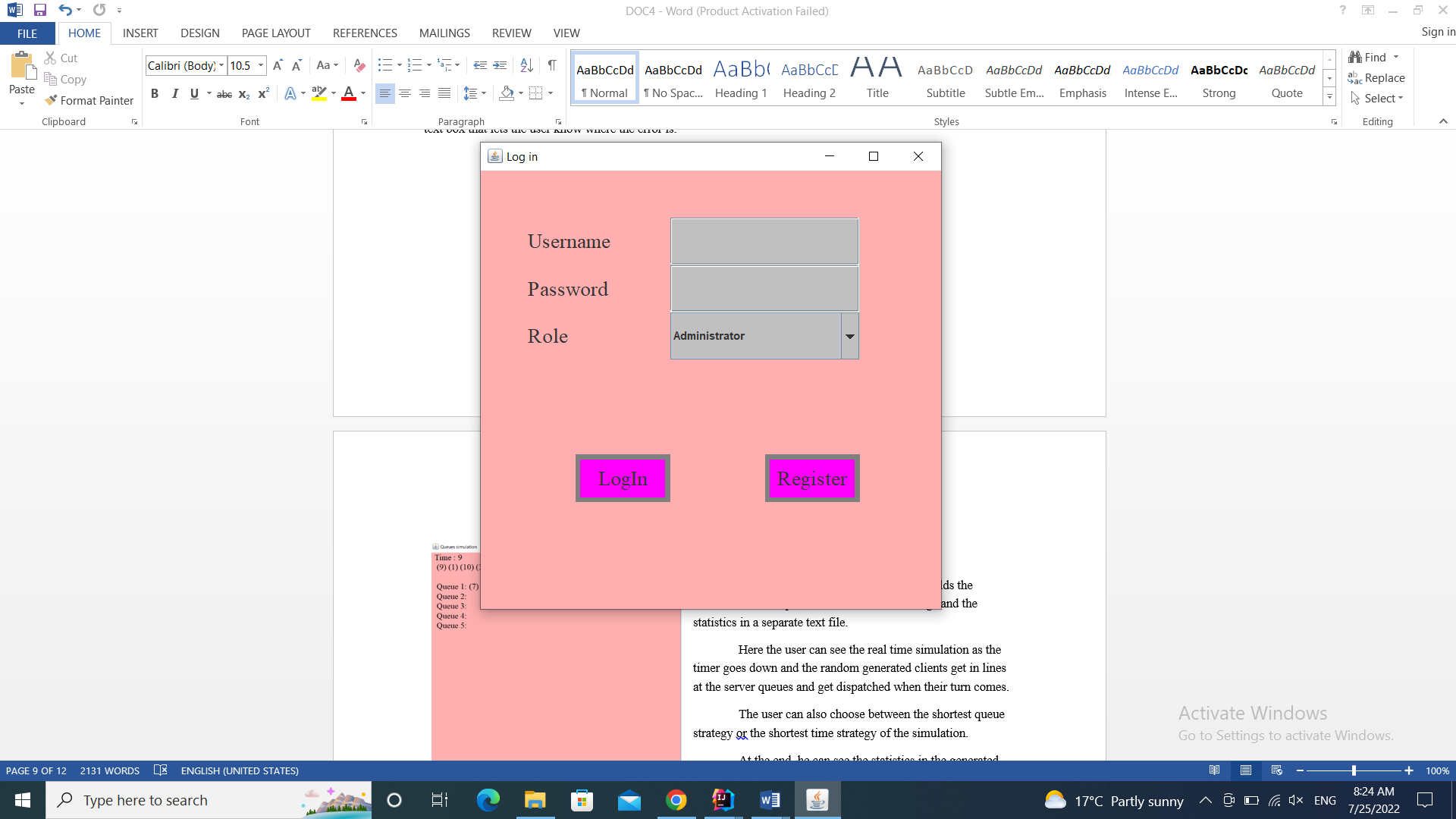
**Classes:**

* *Views –* These classes implement graphical user interfaces for the application, that are user friendly and easy to understand: *AdministratorGUI, BaseGUI, ClientGUI, CompGUI, EmployeeGUI, LogView, ReportGUI.*
* *Controllers –* These classes implement the functionalities of the graphical user interfaces and make the connection between the GUIs and the rest of the application: *AdminController, BaseController, ClientController, CompositeController, Controller, GenerateReportController.*

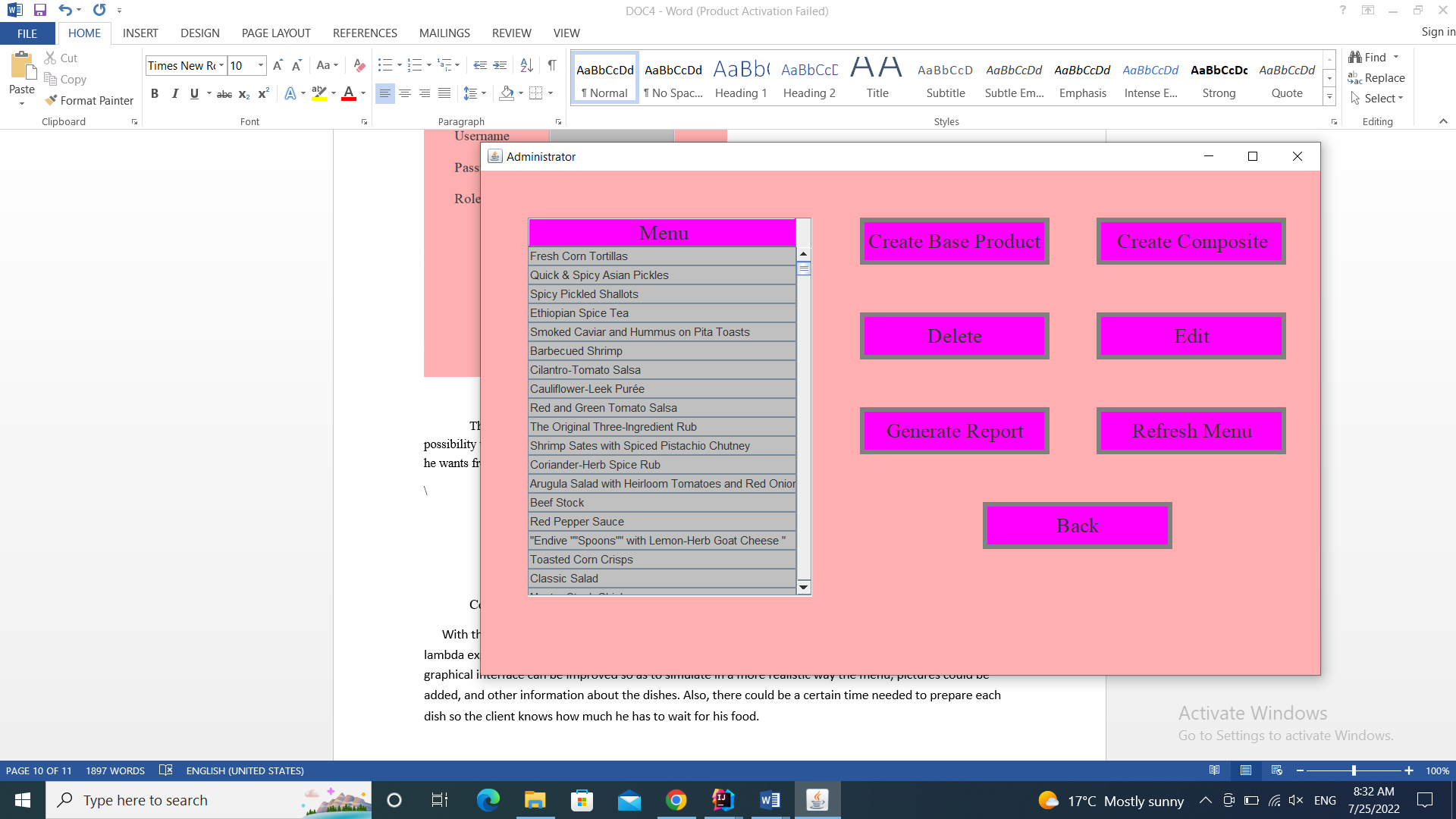
*App –* This class contains the main() method which starts the controller and so, the application of simulating the delivery service.

# Results

As presented, the application is a user-friendly one, as it provides the user an interface where he or she can input the data for the simulation and it signals if any errors occurred while introducing the details through a dialog text box that lets the user know where the error is.



The logIn View is the window that start along with the application and it provides the user with the possibility to either register as a client or log in as a client, employee or administrator. The user can choose the role he wants from the role combo box. Depending on the role he chooses, another window will open.

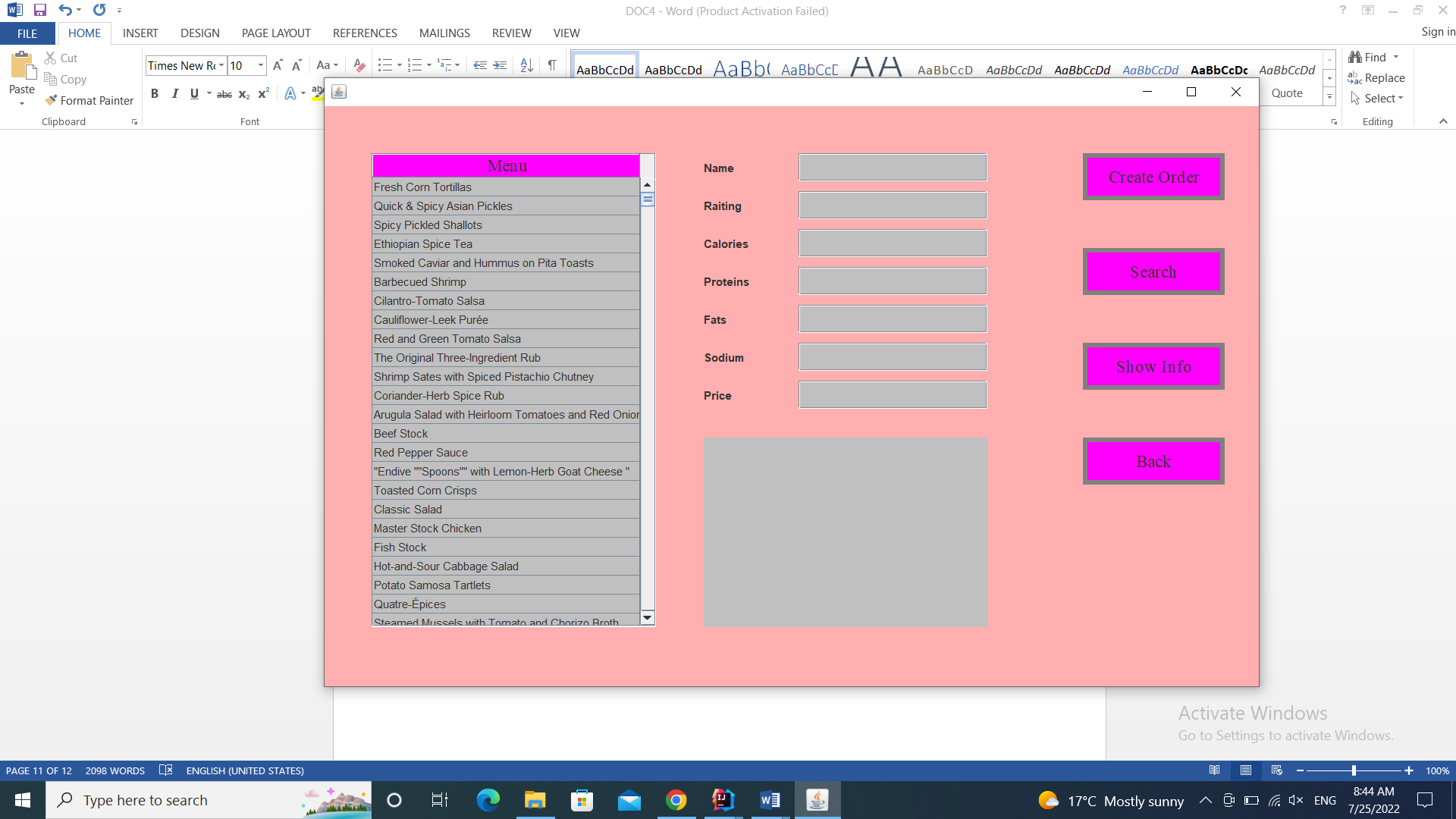


If the right data has been introduced for the administrator (username: “admin”, password: “admin” ), the admin specific window will be opened. The admin can now see the products and decide what he wants to do next.

If the button “Create Base Product” is pressed, a new window in which the admin must introduce the new product’s data will be opened. The same goes for the “Create Composite” button, but here the administrator must select some items from the menu so that they form a new composed menu item. These will be added in the menu in the left and will be visible after pressing the “Refresh Menu” button.

Another thing the admin can do at this point is to delete or edit a product. If the product is to be deleted, it must firstly be selected and then the “Delete” button must be pressed. After pressing the “refresh Menu” button again, the product will disappear from the menu present in the left. If a product must be edited, the “Edit product” button will open another window where the administrator can change the product’s information.

Also, the admin can generate certain reports that are going to be printed in the report files.



The Client window is opened after introducing the right data for the username and password. Here the client can see the menu in the left and choose to select some items and place an order. When the “Create Order” button is pressed, a bill will be printed containing the selected dishes and the total price of the order.

The client can also search for a product using specific keywords or see a product’s details.

Conclusion

With the implementation of this food delivery simulation application, the concepts related to Streams and lambda expressions have been clarified much better. In terms of further development possibilities, the graphical interface can be improved so as to simulate in a more realistic way the menu, pictures could be added, and other information about the dishes. Also, there could be a certain time needed to prepare each dish so the client knows how much he has to wait for his food.

# Bibliography:

* <https://app.diagrams.net/>
* <https://dsrl.eu/courses/pt/materials/A4_Support_Presentation.pdf>
* <https://google.github.io/styleguide/javaguide.html>
* <https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_domains/packagediagram.html>
* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.wikipedia.org](http://www.wikipedia.org)
* <https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_4.pdf>
* <http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>
* <https://www.draw.io/> (for diagrams)