DOCUMENTATION

ASSIGNMENT 4

NAME: Muntean Cristian

GROUP: 30424

Contents

[1. Objective 3](#_Toc100861146)

[2. Analysis, modelling, scenarios and use cases 3](#_Toc100861147)

[3. Design 3](#_Toc100861148)

[4. Implementation 3](#_Toc100861149)

[5. Results 3](#_Toc100861150)

[6. Conclusions 3](#_Toc100861151)

[7. Bibliography 3](#_Toc100861152)

# Objective

Design and implement a food delivery management system for a catering company. The system should have three types of users that log in using a username and a password: administrator, regular employee, and client. The client can order products from the company’s menu and can search for products by keyword, number of calories, rating, number of proteins, number of fats, number of sodium or price. A client can also register a new account and log in his/her account. The admin can import an initial set of products from a .csv file, manage products: add, edit, remove. When adding a new product, that product can either be a composite product (such as a menu containing a steak, a garnish and a dessert). Also, the admin can generate reports based on some given inputs. The employee is notified when a new order is created by a client.

# Analysis, modelling, scenarios and use cases

1. Analysis

This application simulates a food delivery management system, allowing the user to perform actions as a client, an admin or an employee. For the client operations, the user is able to search for products and create new orders. In order to search for products, the user must first log in the account, then click on the “Search products” button, fill in however many fields he/her wants then press the “Submit” button. A table containing all the products will appear in a new window. To create a new order, the user must select at least a product from the products table, then click “Order”.

For admin operations, the user is able import a set of products, manage a product, and generate reports. In order to import the products, the user must click on the “Import products” button. In order to manage a product, the user must click on the “Manage products” button, then select one operation out of “Add”, ”Delete” or “Edit”. In order for the “Delete” and “Edit” to work, a single row must be selected from the products table.

The employee doesn’t have any operations. It is only able to see the orders in real-time.

1. Modelling

For client register operation, the user must input a username and a password. In order for the register operation to succeed, the username must not already exist.

For the login operation, the user must input a correct username-password combination that exists. According to the access level of the account, the user will be shown the Client/Admin/Employee window.

After being logged in, if the user is a client, he can perform the “Search” function. In order to perform the “Search” function, the user must input however many filters he/she wants with the restriction that all of the inputs besides the keyword must be a number. From those number inputs, all of them besides the rating must be an integer. Besides the “Search” operation, the client can add a new “Order”, by selecting at least a row from the table, then click the “Order” button.

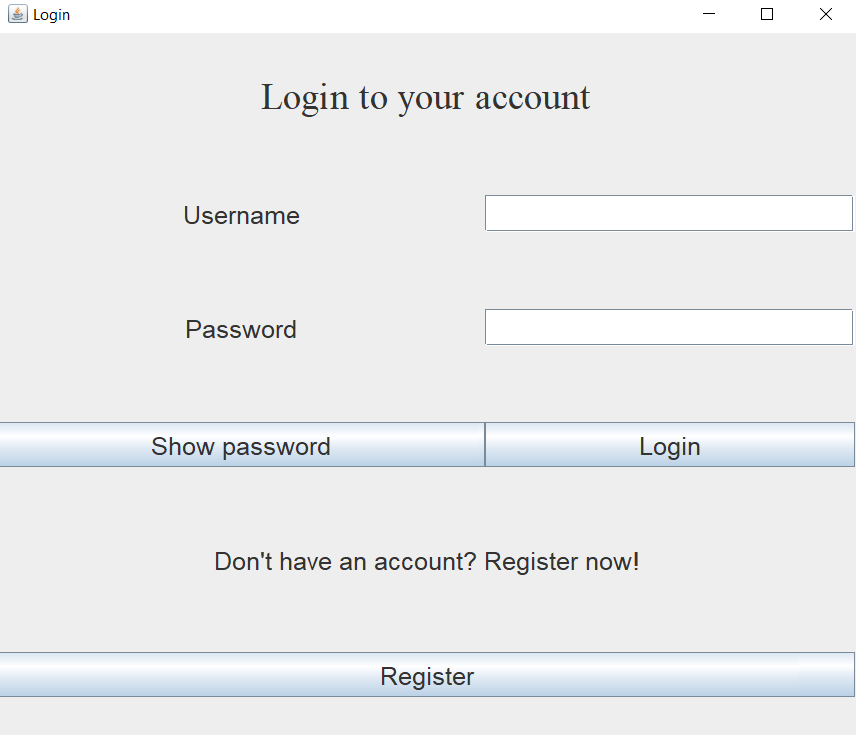
If the user is an admin, he can Import the products by clicking on the button, manage products by clicking on the “Manage products” button. If said button is clicked, a new window will pop up and the user can choose from “Add”/”Edit”/”Delete”. If the “Add” operation is selected, the user can choose between adding a composite product or a base product. If the user selects a composite product, he/she will have to select at least an element from the table, add a title for the product and click “Submit”. If the user selects a normal product, he/she will have to input all the fields (title, rating, calories, proteins, fats, sodium, price) with the following restrictions: the rating must be a double and calories, proteins, fats, sodium, price must be integers. The last operation an admin can perform is the “Generate reports” one. If the user selects this operation, he/she will have to input some values in order for the reports to be generated. The restrictions of these inputs are: the hours must be integers in the value 0 – 24, the minimum number of orders, the minimum number of products and the minimum order value must be greater than 0 and the day must be in the interval 1 – 31.

1. Scenarios and use cases

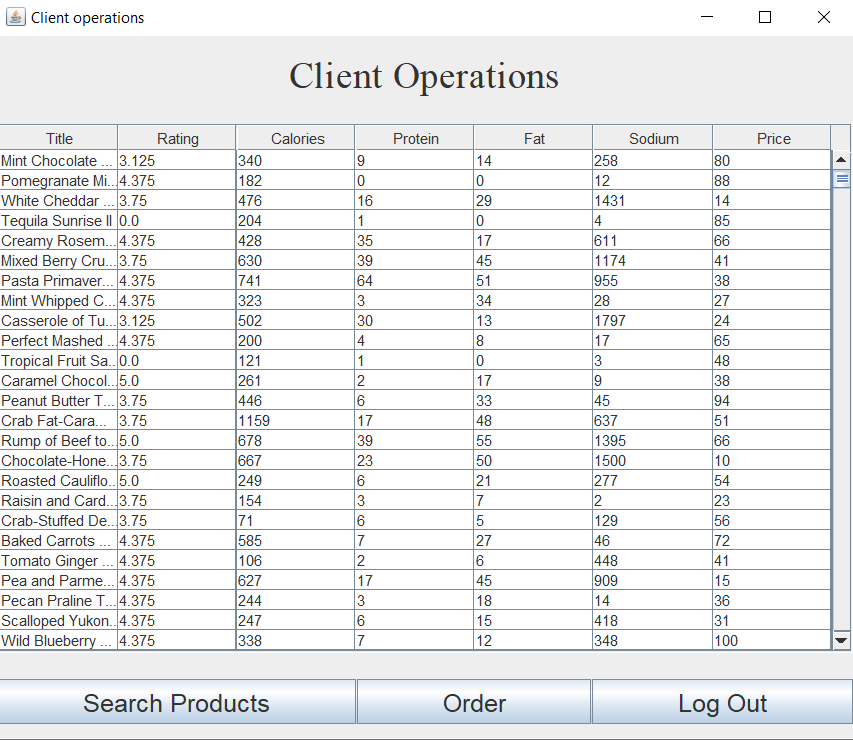
A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

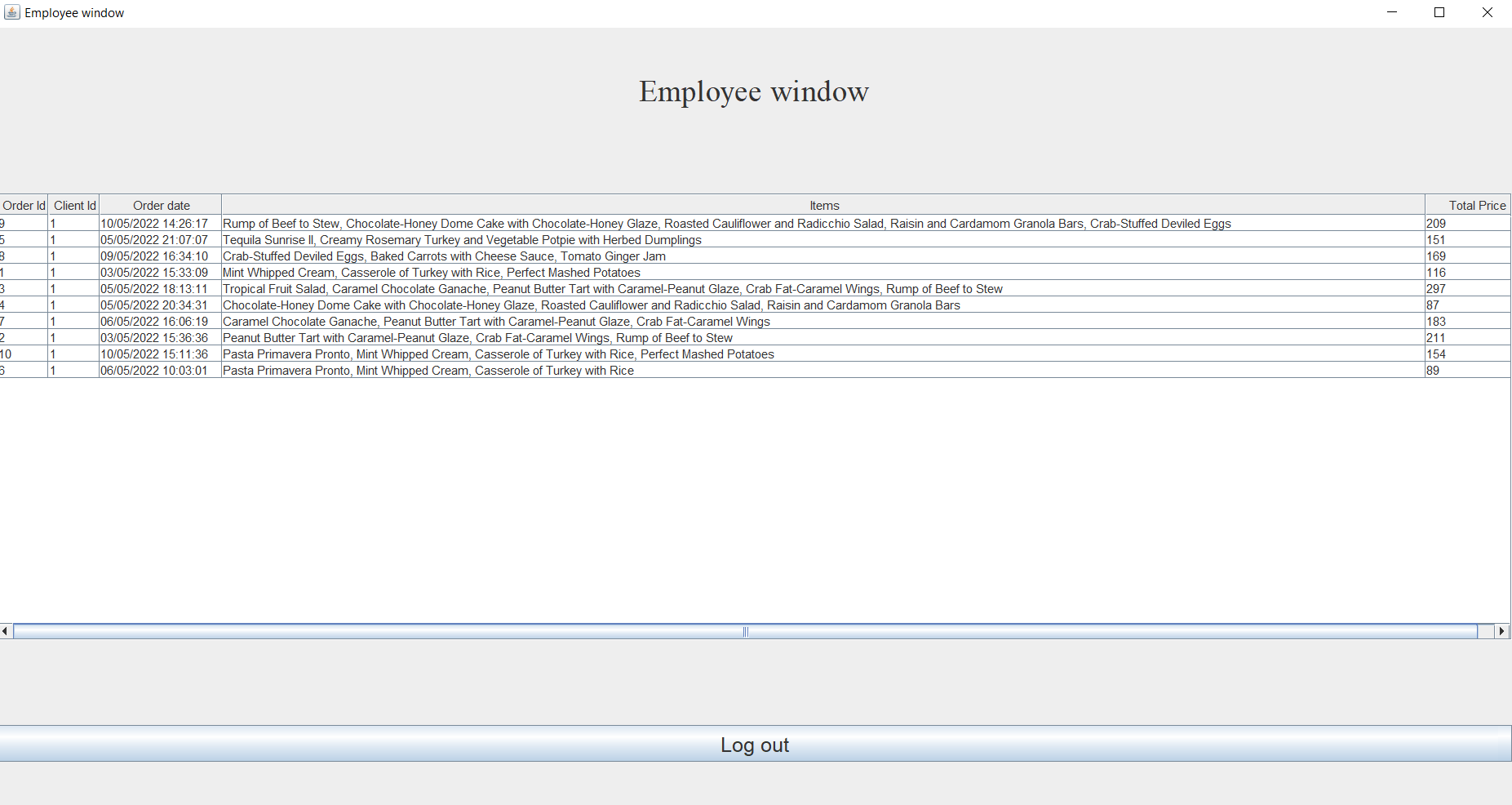
The use cases are strongly connected with the steps the user has to make, so I created the user interface with this in mind, resulting in a straight forward design.

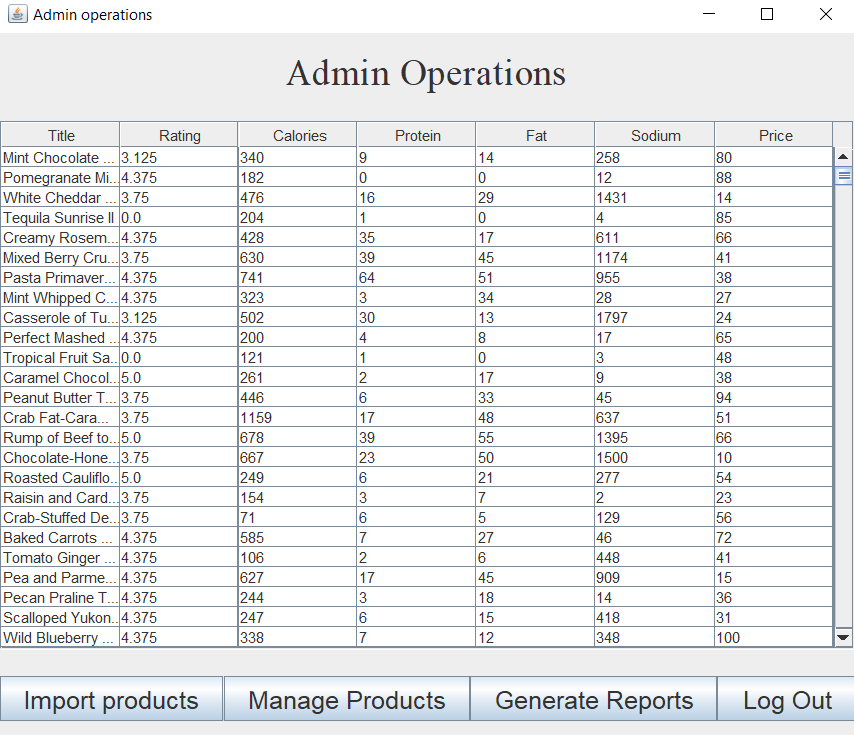
The first window that the user will see is the login window, where he/her has to input a username and a password and then click the login or the register button.



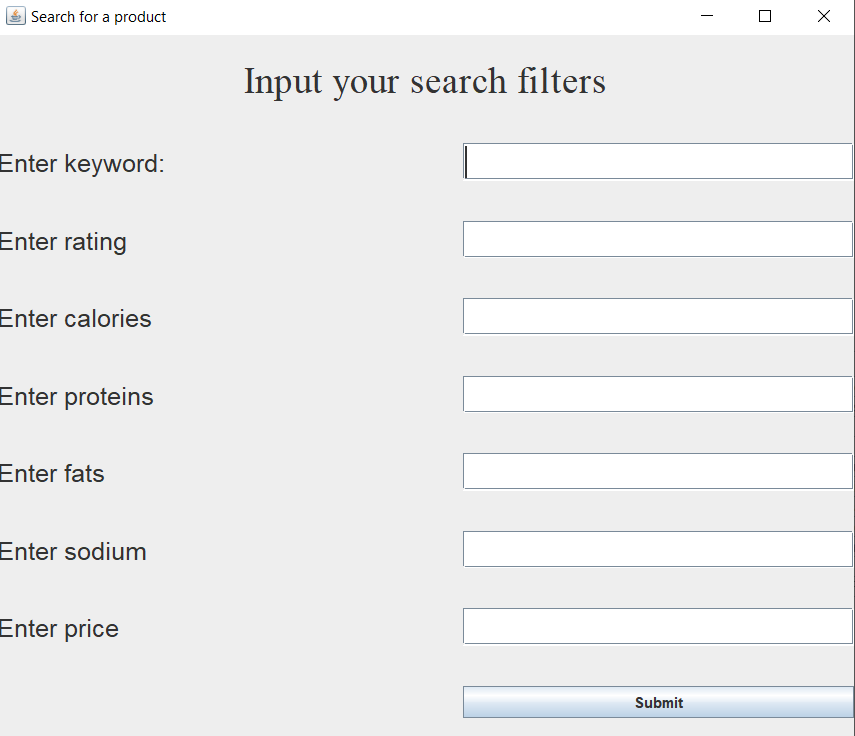
The following windows correspond to the type of account the user has. A user can have one of three types of accounts: client, admin and employee. After the login/register a window corresponding to the type of account will replace the login window.



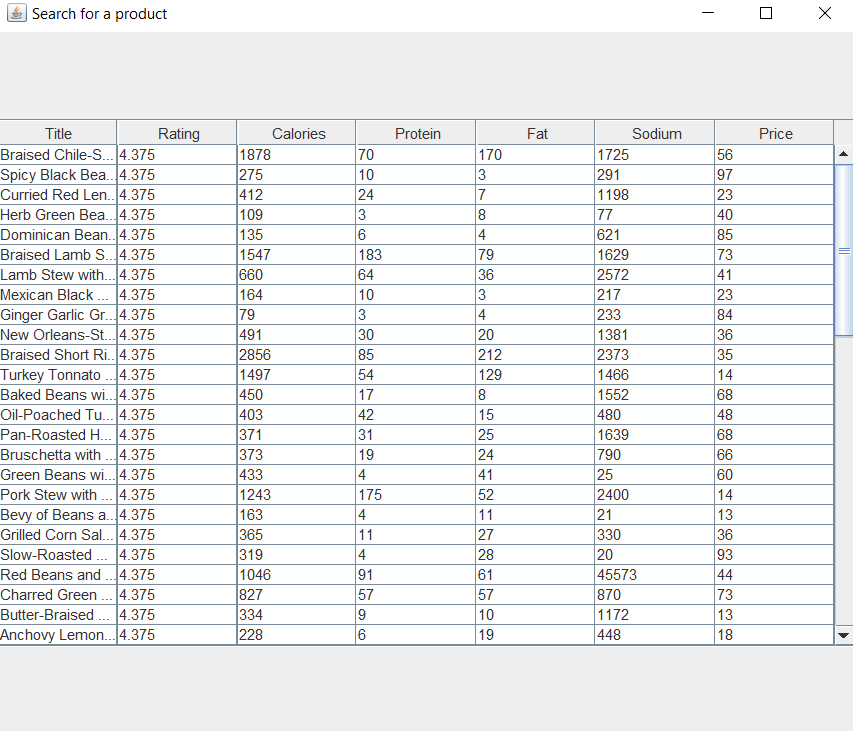




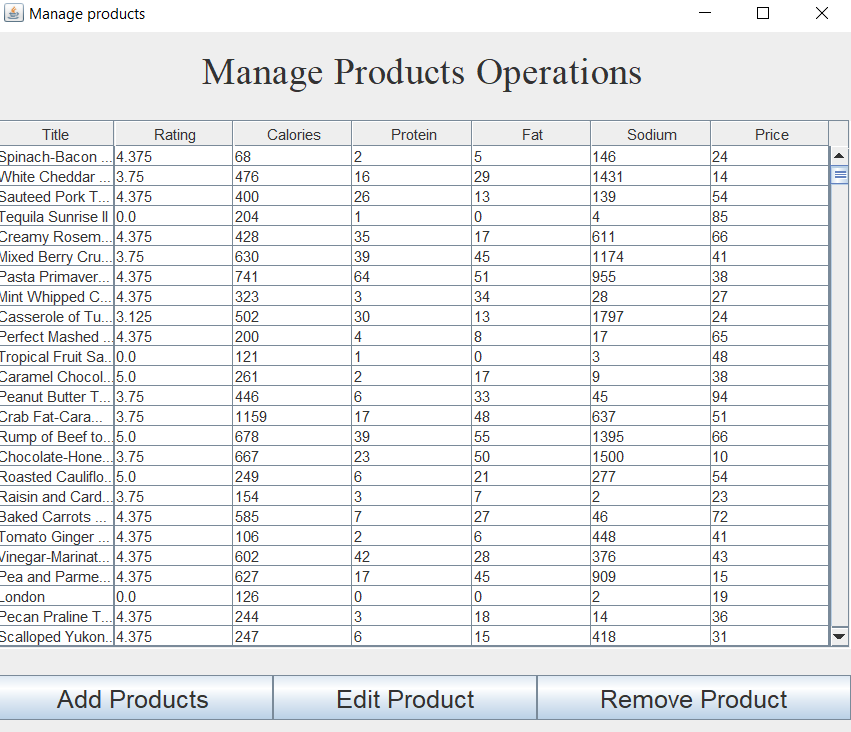
If the user is a client and selects the “Search Procuts” operation, the following window will pop up:



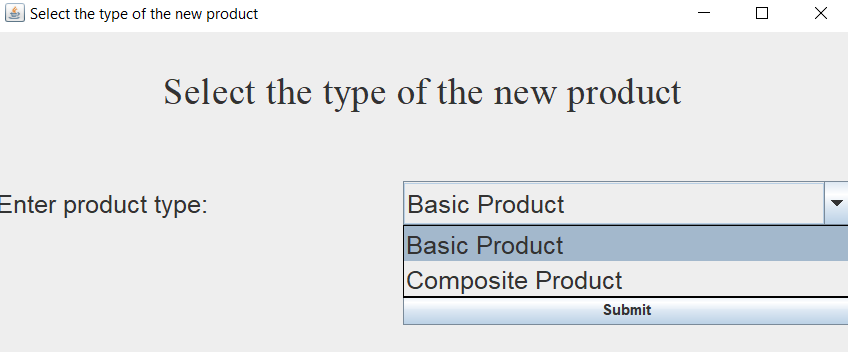
After the user has inputted the filters he/she wants, the products that correspond to the filters will be shown as a table. In the next example, the products that contain the word “Beans” and have a rating of 4.375 are displayed as a result of the “Search” operation.



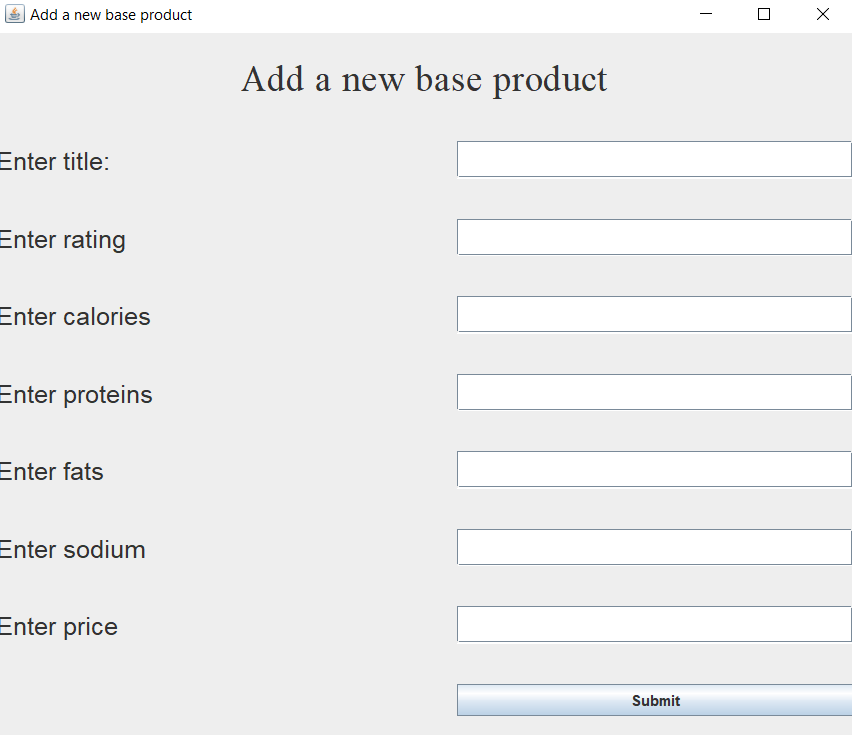
If the user is an admin and selects the “Manage products” operation, the following window will appear:

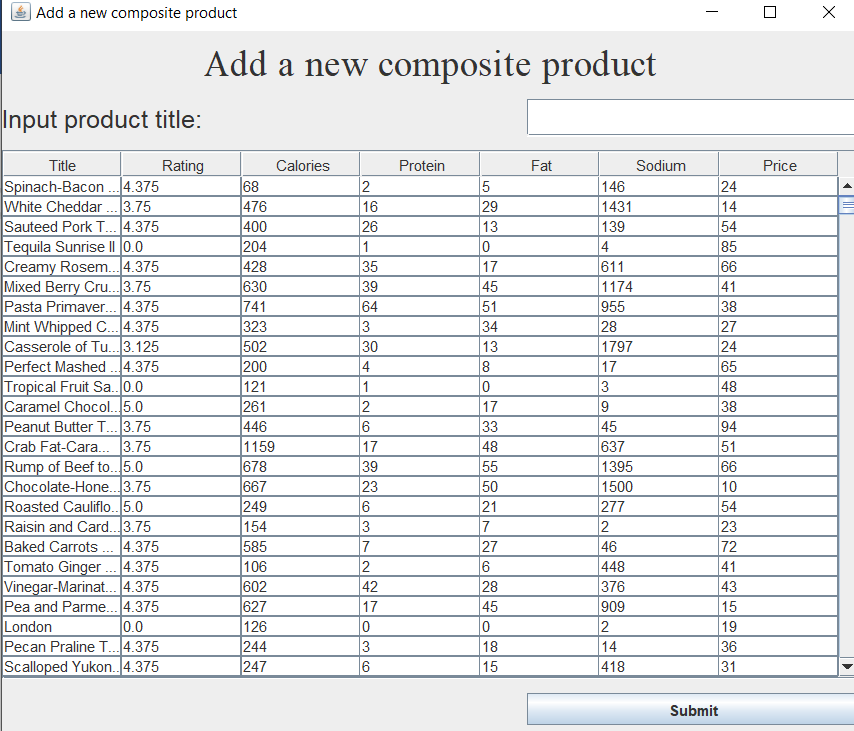


If the user selects the “Add products” operation, a window will pop up that asks the user what type of product he/she would like to add.



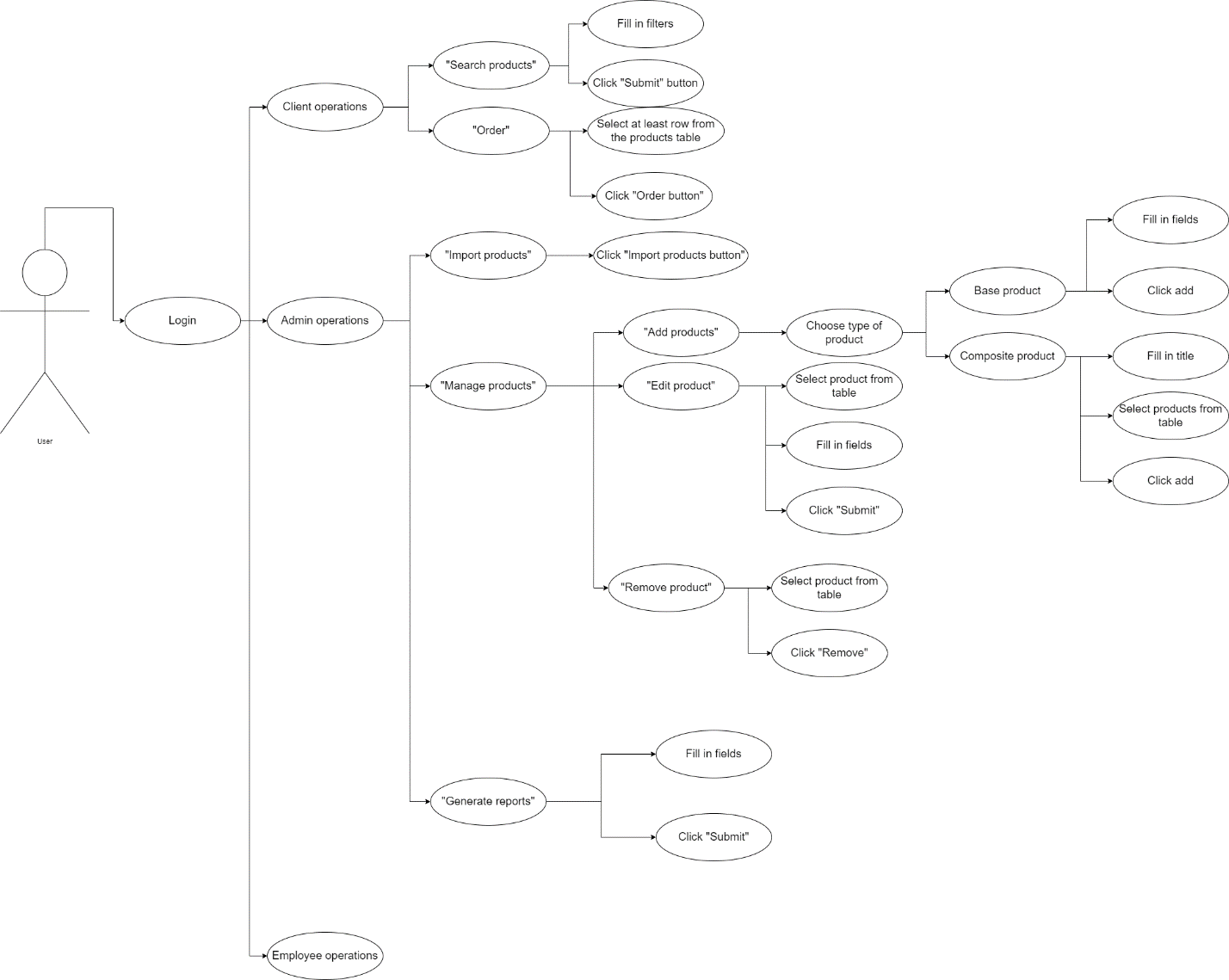
The “Basic Product” and “Composite Product” selections will decide which window will pop up. The two separate windows will pop up.





If the “Edit product” operation is selected, the window that pops up looks the same as the “Add basic product” window.

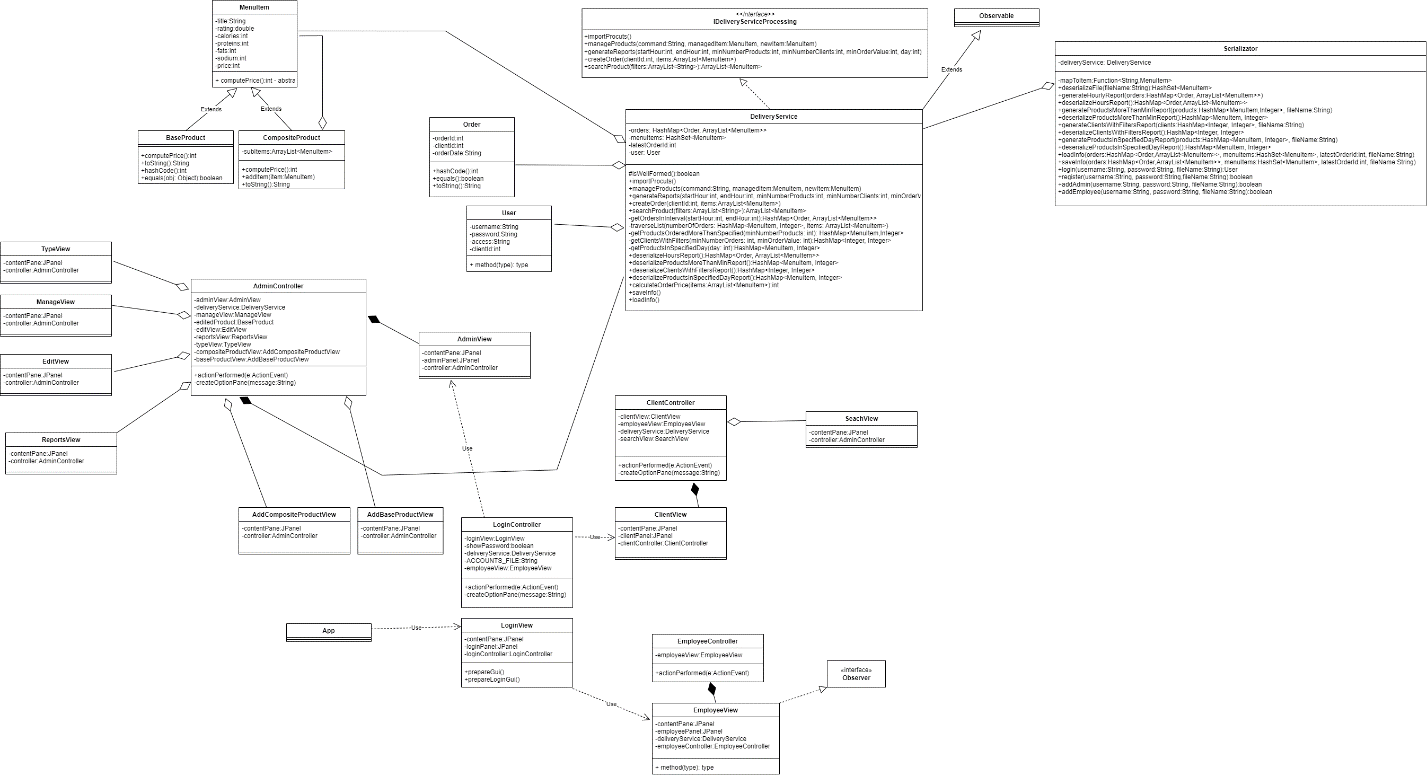
The use case diagram for these processes is presented below:



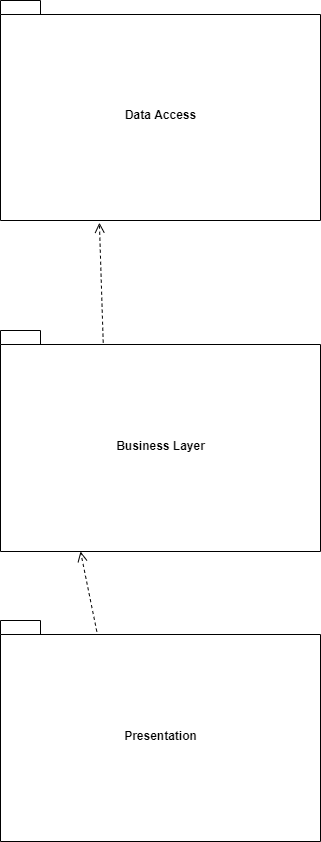
# Design

In the design of my application, I used the layered architecture that consists of the following layers:

* Presentation Layer – contains the classes responsible with the user interface and connecting the user interface to the business logic layer
* Business Logic Layer – contains the classes that encapsulate the application logic
* Data Access Layer – contains the classes that are responsible with saving and loading information from files.
* Class diagram



* Package diagram



* Data structures

The data structures I have used in this project are either primitives (such as int, boolean, double) or more complex data structures (such as ArrayLists, HashMaps, HashSets). I used ArrayLists in order to store the results of the search operation because the size is unknown at the beginning of the search. I used HashMaps to store pairs of items (such as a Order - ArrayList<MenuItem> pair, or MenuItem – Integer pair). I used HashSets in order to not have products that are duplicates in the “import from procuts.csv operation”.

# Implementation

Because I used a layered architecture, my application has 3 main parts:

1. The Business Layer

This part contains the classes that are responsible with the logic of the application. It contains 4 classes, 1 interface and 1 abstract class:

1. The “IDeliveryServiceProcessing” interface

This interface is the blueprint of for the “DeliveryService” class and it contains the methods that are responsible with the operations that a user can perform (importProducts, manageProducts, generateReports, createOrder, searchProducts)

1. The “MenuItem” abstract class

This abstract class represents a product that will later be in the the list of products. It contains 7 fields: title, rating, calories, proteins, fats, sodium and price. Because it is an abstract class, it can not can not be instatiated.

1. The “BaseProduct” class

This class extends the MenuItem abstract class and represents a simple product. It has the hashCode and equals methods overridden in order to remove the title duplicates from the products.csv file.

1. The “CompositeProduct” class

This class extends the MenuItem abstract class and represents a composite product (such as a menu that contains a steak, a garnish and a dessert).

1. The “Order” class

This class represents an order that a client has placed. It only stores the orderId, the clientId and the orderDate. A full order (that contains the products) consists of an object of type Order and an ArrayList of MenuItem objects, which are stored in a HashMap.

1. The “DeliveryService” class

This class implements the “IDeliveryServiceProcessing” interface and it’s methods. It contains a HashMap that stores orders, a HashSet that stores the products, an int that stores the latest order id and an object of type User that contains the user that performs the operation.

1. The Data access Layer

This part contains the classes that are responsible with accessing data from the serialized files (storing and restoring information when the app closes or when the app opens). It contains 2 classes:

1. The “Serializator” class

This class is responsible with everything that communicates with the files that store the information about accounts, orders information and products. Some of the most important methods are “saveInfo”, “loadInfo”, “desererializaeFile”, “login” and “register”. There are 2 methods that are able to add an admin and an employee named “addAdmin” and “addEmployee”.

1. The “User” class

This class contains information about the user such as the clientId, username, password and access level. The access level can only be “Client”, “Admin”, “Employee”.

1. The Presentation Layer

This part contains the classes that are responsible with the user interface and the classes that connect the user interface to the logic of the application. The most important classes of this layer are:

1. The “AdminView” class

This class extents JFrame and is the frame that shows when a user with “Admin” access levels logs in.

1. The “AdminController” class

This class controls anything that happens within the “AdminView” frame and the subframes that appear when the user wants to perform an operation.

1. The “ClientView” class

This class extents JFrame and is the frame that shows when a user with “Client” access levels logs in. Along with the “ClientView”, the “EmployeeView” window appears in order for the notifications to be visible whenever a client places an order.

1. The “ClientController” class

This class controls anything that happens within the “ClientView” frame and the subframes that appear when the user wants to perform an operation.

1. The “EmployeeView” class

This class extents JFrame and is the frame that shows when a user with “Employee” access levels logs in.

1. The “EmployeeController” class

This class is only responsible with logging out of the employee account.

# Results

For testing the application I created the invariant method in the “DeliveryService” class, and implemented pre conditions and post conditions within the method of said class by using asserts.

Moreover, I also tested cases manually, by inputting some products, adding orders, creating new accounts, using bad username – password combinations.

# Conclusions

By doing this project, I managed to learn about serialization, about storing information in files and retrieving it when the application launches, about invariant methods and pre conditions and post conditions. Also, I learned about lambda functions and working with streams. A good example of using lambda functions and streams is the “search product” method, in which I started from an Array List of items and filtered said list using the “.filter()” method from streams together with a lambda function.

# Bibliography

* <https://docs.oracle.com/> - for learning how to work with various layouts and graphical user interface components
* <https://stackoverflow.com/> - for various bugs that I encountered during the development
* Programming Techniques in Java – Lectures of prof Cristina Bianca Pop – for familiarizing myself with the serialization and the invariants