

Technical University of Cluj-Napoca

Programming Techniques

Laboratory – Assignment 4

Food Delivery Management System



TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

Teacher: prof. Ioan Salomie

Teacher Assistant: Dr. Cristina Pop

Student: Rusu David

Group: E_30424

Contents

1.	Assignment Objective	3
2.	Problem analysis, modeling, scenarios and use cases	3
I.	Problem Analysis.....	3
II.	Modeling	3
III.	Use Cases and Scenarios.....	4
3.	Design.....	8
I.	Class Design Decisions	8
II.	UML Diagrams.....	11
III.	Data Structures	12
IV.	Packages.....	12
V.	User Interface	13
4.	Implementation	26
5.	Results.....	27
6.	Conclusion.....	27
7.	Bibliography	27

1. Assignment Objective

The scope of this assignment is to 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.

2. Problem analysis, modeling, scenarios and use cases

I. Problem Analysis

Managing a food delivery service can be rather challenging especially when it is done not using modern technologies, it can quickly turn into a messy nightmare where no one knows what is going on.

To solve such a logistics problem, we use modern day technology to create a software application that enables the delivery service to run in an organized manner.

Now that we know the problem that we need to solve, we have to come up with a solution. A typical delivery service has 3 main moving parts to it: Client, Employes, Managers. Each of those three moving parts support a critical role in the big picture, thus an application will be divided into those 3 parts. A client for example, creates an order, after, an employee takes the order and services the customer. Above these two actors, we have a manager that overlooks what the customer can order, and how well the employee provided the service. Now that we know what the main parts in our application are, we can move on to modeling.

II. Modeling

Now that we analyzed the problem, we must model a solution. In this case, it will have three main sections:

a) The client part

From here a client should be able to create an account, login, and view the available products from which he will then create an order. There should also be a checkout page that shows the items and total of the order.

b) The Employee part

From here an employee should be able to create an account and login. Once logged in, a page with all the active orders should be presented. The employee should have the option to mark an order as completed.

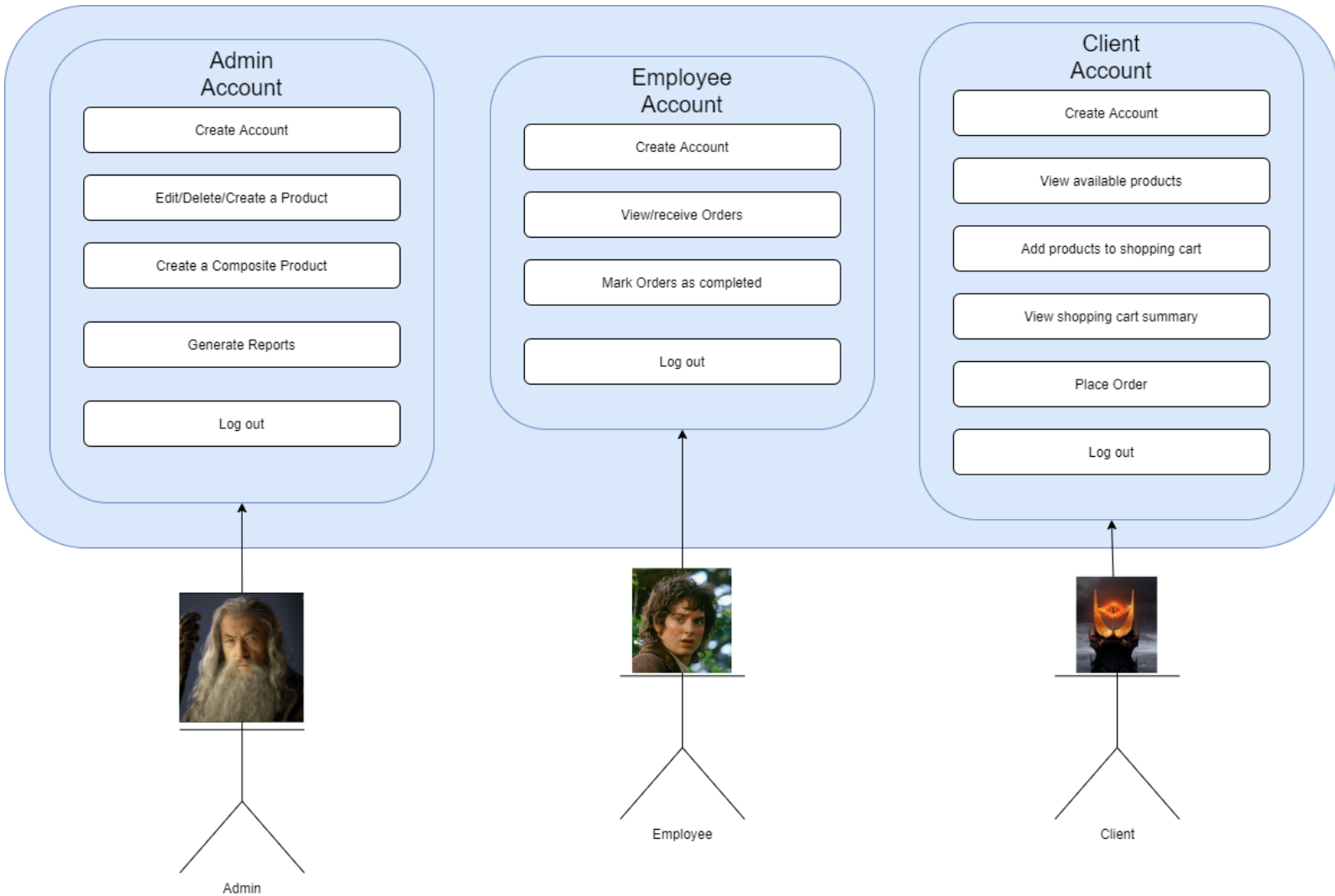
c) Admin part

The managerial part of the service happens here. A manager should be able to view all the products that are offered, edit them or even create new ones. A special page that presents reports should also be available so that the manager can see how well the service is doing.

III. Use Cases and Scenarios

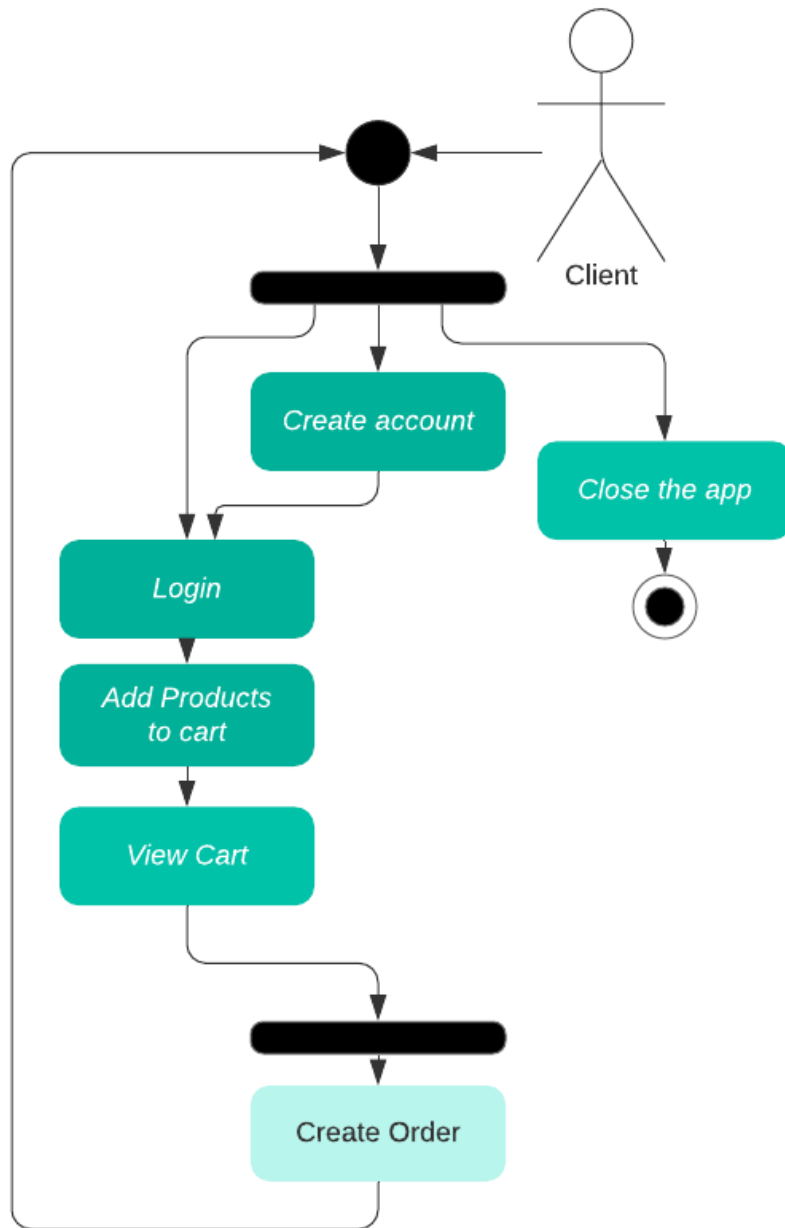
Below we can find the use case and scenarios diagrams

Use Case:

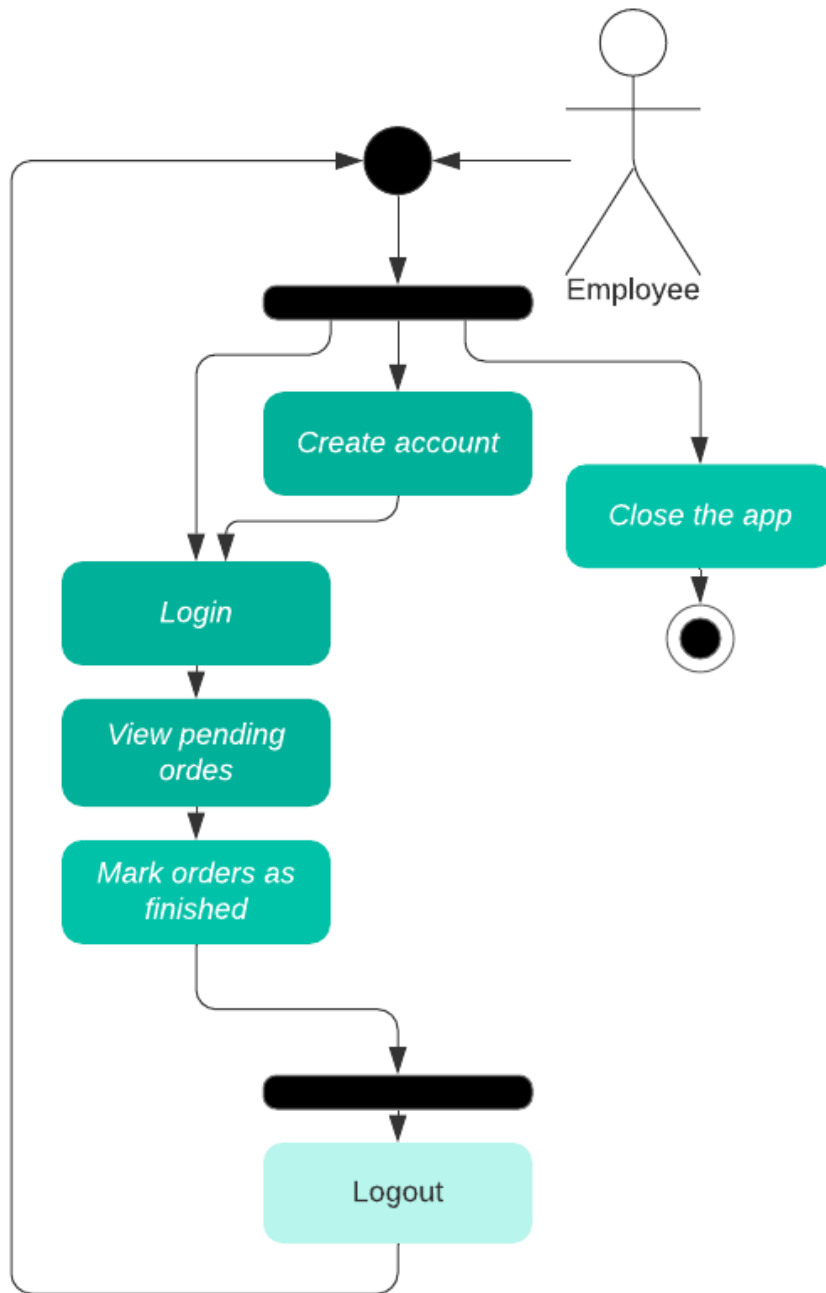


Scenario Diagram's:

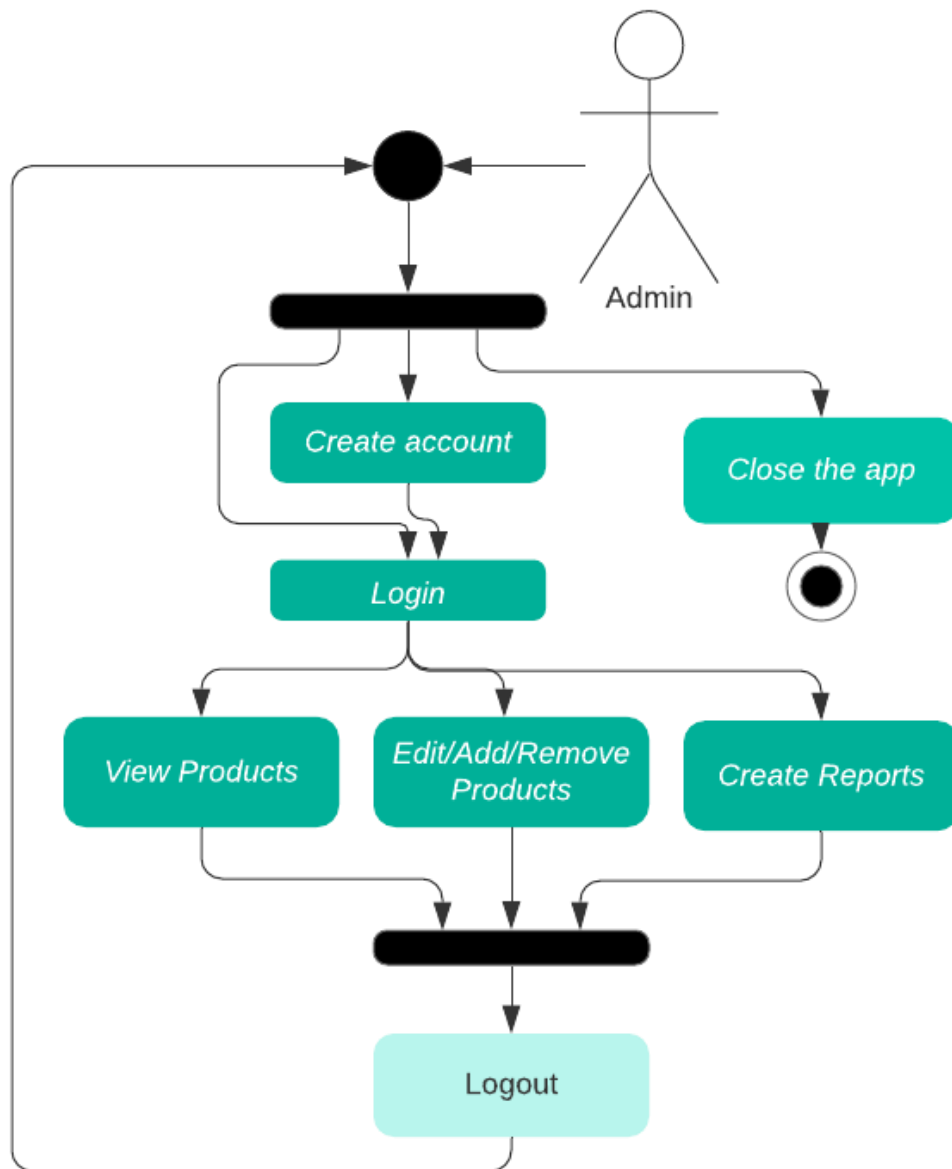
Client:



Employee:



Admin:

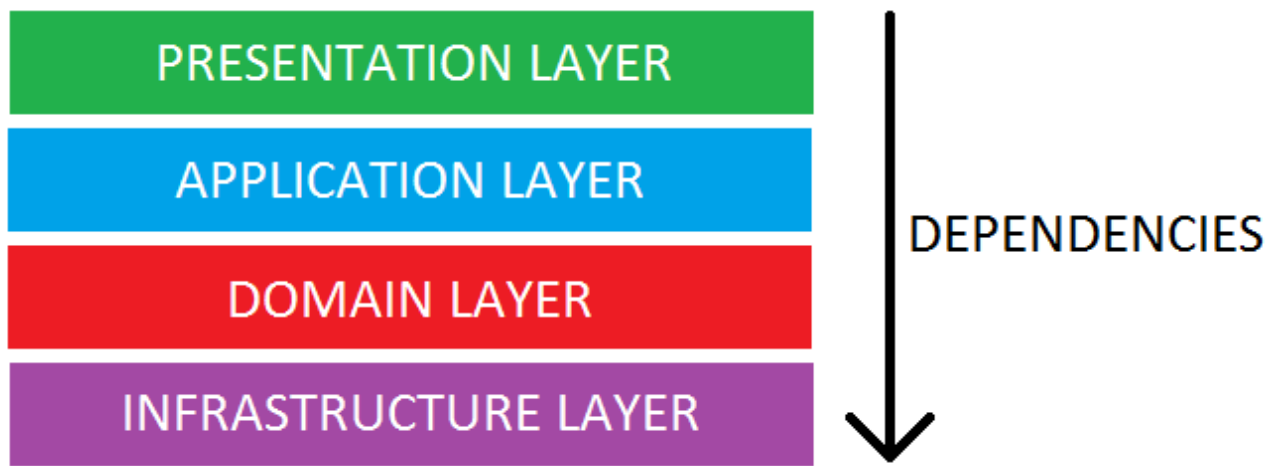


3. Design

I. Class Design Decisions

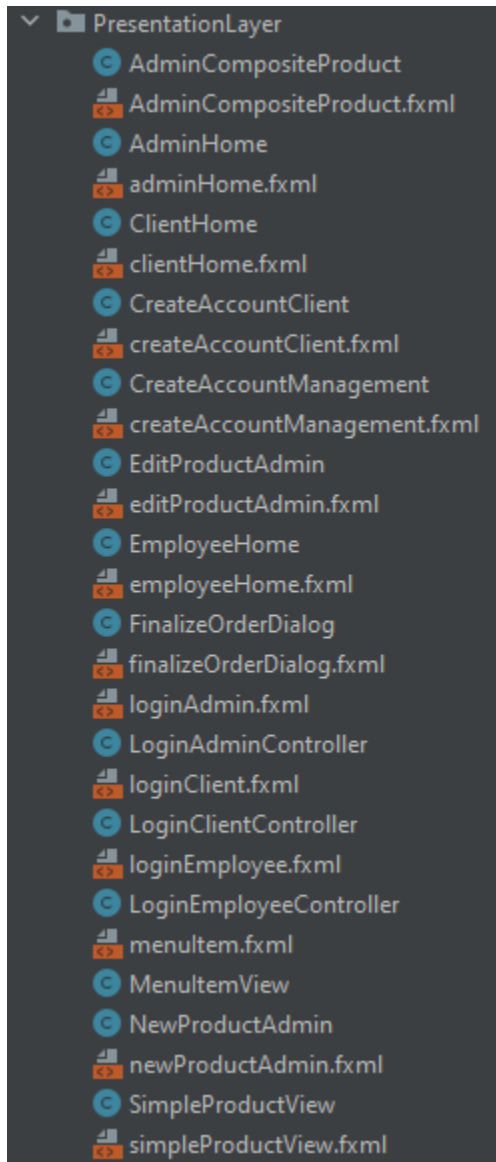
One of the most important parts in designing our app is to choose a structure. We have 2 options here, either a custom structure or an already existing one, better known as a design pattern. In this case we will go with a preexisting design pattern called Layered Architecture. A Layered Architecture, as I understand it, is the organization of the project structure into four main categories: presentation, model, Business layer, and Data Access Layer. Each of the layers contains objects related to the particular concern it represents.

- **The presentation layer** contains all of the classes responsible for presenting the UI to the end-user or sending the response back to the client (in case we're operating deep in the back end).
- **The Business layer** contains all the logic that is required by the application to meet its functional requirements and, at the same time. In most systems, the application layer consisted of services orchestrating the domain objects to fulfill a use case scenario.
- **The Data Access Layer** contains all the classes responsible for doing the technical stuff, like persisting the data in the database, like DAOs.
- **The Model Layer** represents the underlying domain, mostly consisting of domain entities and, in some cases, services.



In our case, the **model layer** is merged into the Business layer.

The **presentation layer** will be the JavaFX app which is made up of multiple FXML files and their controllers:



The **Business layer** will be made up of the following classes:

- BaseProduct
- CompositeProduct
- DeliveryService
- IDeliveryServiceProcessing
- Menuitem
- Order

- User
- UserManagement

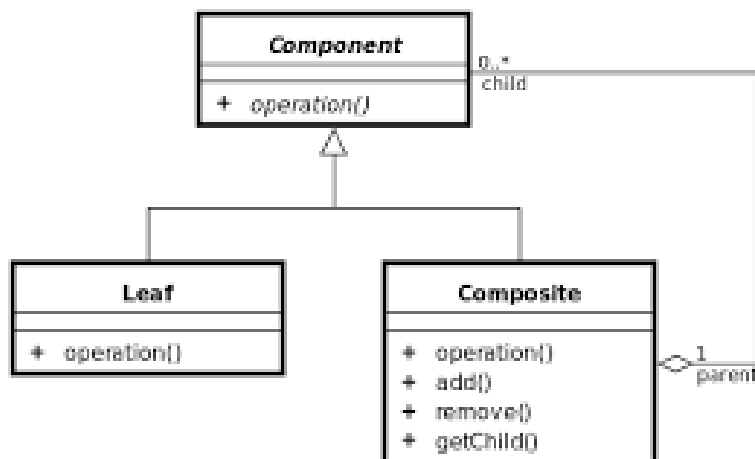
The **Data Access Layer** contains the following classes:

- FileWriter
- Serializator

Besides the Layer architecture, we also use the composite design pattern. The composite pattern is meant to allow treating individual objects and compositions of objects, or “composites” in the same way. It can be viewed as a tree structure made up of types that inherit a base type, and it can represent a single part or a whole hierarchy of objects.

We can break the pattern down into:

- **component** – is the base interface for all the objects in the composition.
- **composite** – has leaf elements. It implements the base component methods and defines the child-related operations.
- **leaf** – implements the default behavior of the base component. It doesn't contain a reference to the other objects.

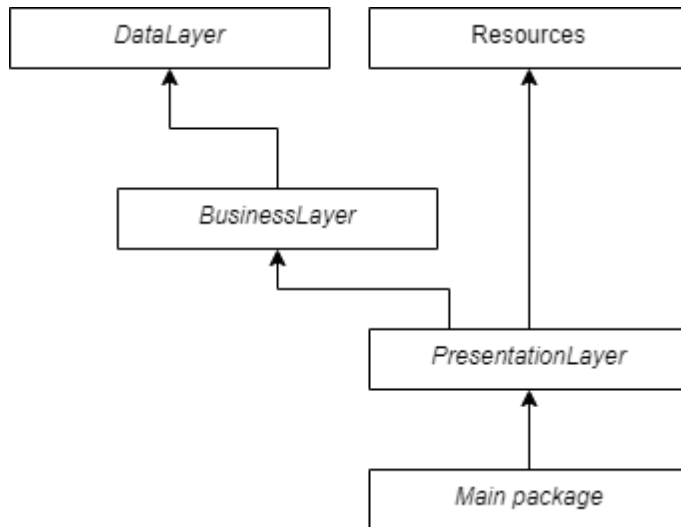


In our case, the design pattern is constructed as follows:

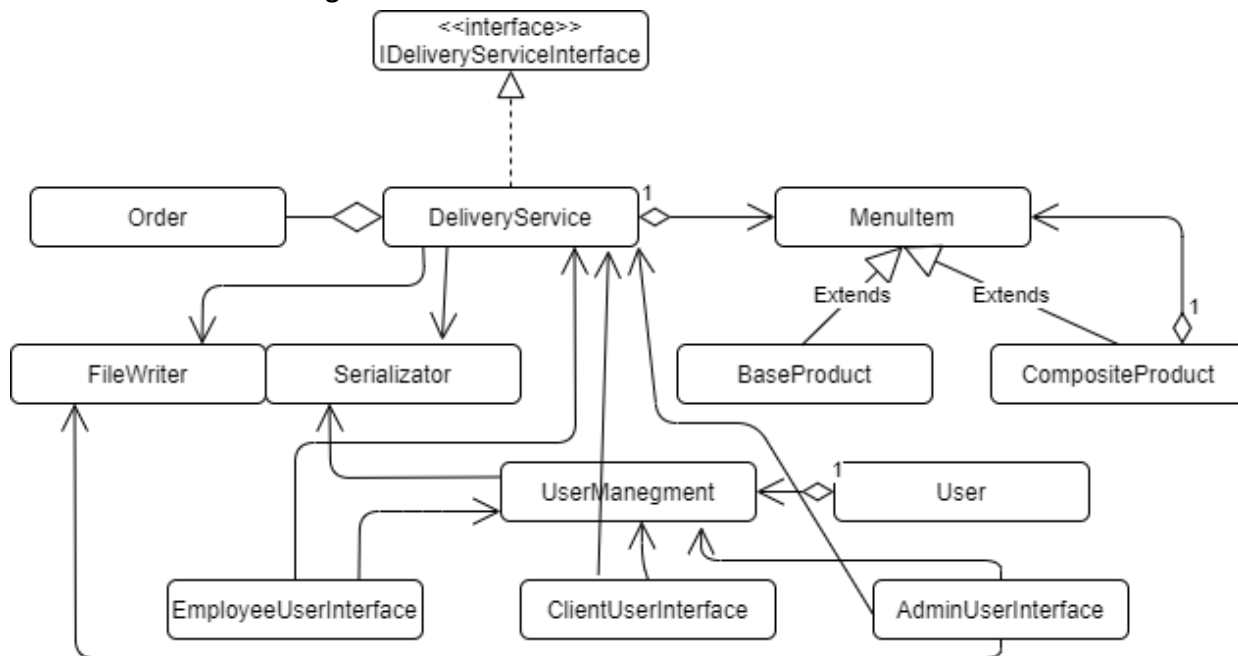
- **Component** is made of MenuItem
- **Leaf** is made of BaseProduct
- **Composite** is made of CompositeProduct

II. UML Diagrams

Package wise UML diagram:



Class wise UML diagram:



III. Data Structures

All the application specific data structures can be found in the BusinessLayer package. In that package we have 5 classes that represent a data type:

- a) User, this holds a user and has the following data:
 - Name
 - Password
 - Rights
- b) Order, this holds a order for a specific client and holds the following data:
 - orderID
 - ClientID
 - date
- c) BaseProduct, this holds a simple product from which we can derive other composite products. This data structure is an extension of the MenuItem data structure.
- d) CompositProduct, this data structure holds a list of MenuItem, thus forming a Composite product.
 - List<MenuItem> item
- e) MenuItem, this data structure is the base for the BaseProduct and CompositProduct data structures, it holds the following data:
 - Price
 - Rating
 - Calories
 - Protein
 - Fat
 - Sodium
 - Title

IV. Packages

The application is split into multiple packages in order to obey the Layered Architecture design pattern. The application has 5 main packages:

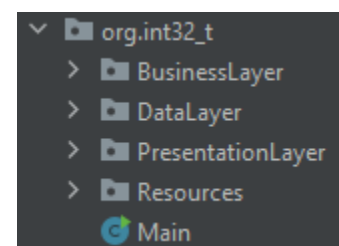
-**org.int32_t**: contains all the application

-**BussinessLayer**: Contains the backend logic for the user interface

-**DataLayer**: This package manages data manipulation with external files, such as writing orders, or serializing objects.

-**PresentationLayer**: In this package, the user interface is found.

-**Resources**: Holds extra data that the UI uses, for example images.



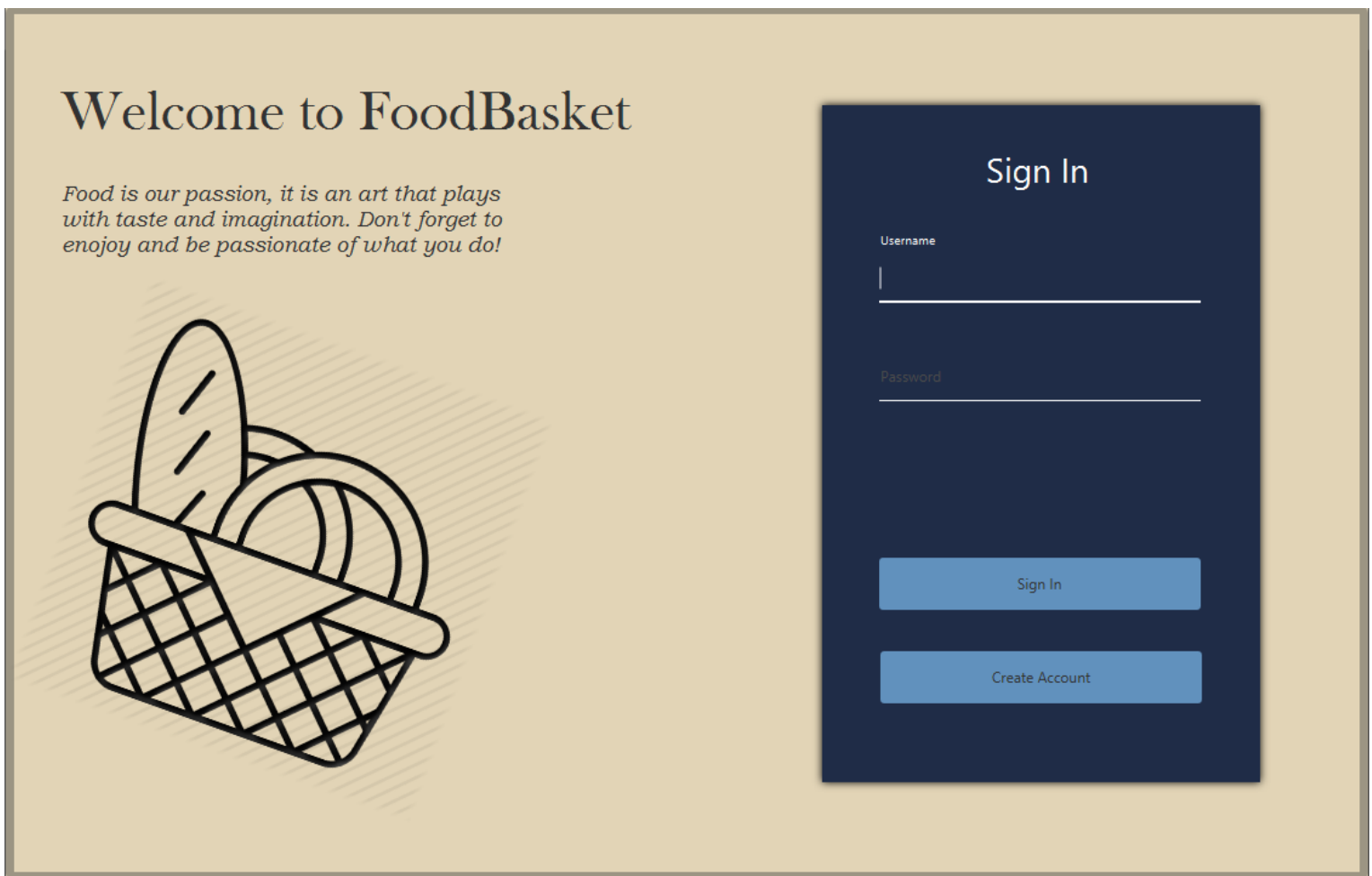
V. User Interface

The UI represents a large portion of this app. Its main purpose is to offer the user an easier method to interact with the backend and organize delivery service. The UI is made up of 3 main windows. Each window is meant to be used by one user with a certain role. For example the client window will only be used by clients, where as the admin window will only be used by admins. Each window offers all the necessary elements such that all the required operations can be completed successfully.

The UI was implemented in JavaFX, and style with CSS as well JFoenix for certain UI elements, such as buttons or fields. JavaFX was chosen over Swing because it offers a much more modern way of writing it, FXML compared to java objects.

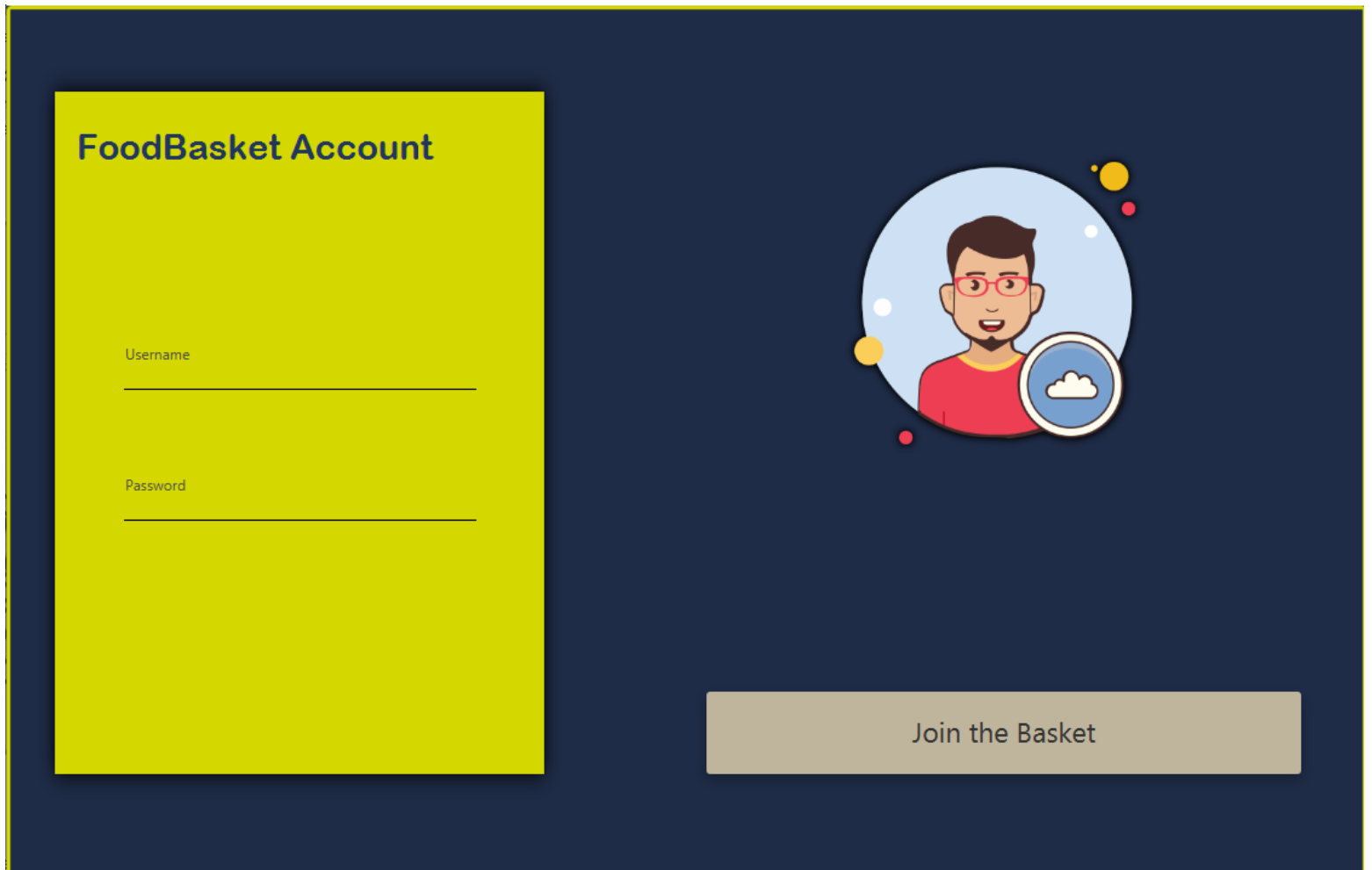
We will present the UI by roles (client, admin, employee) starting with the client.

I. The login menu



From the login screen the user can select to create a new account, or if an account is present, the user can input his credentials into the username and password fields and sign in.

II. Create Account menu



The image shows a user interface for creating a new account. It features a dark blue background. On the left, there is a yellow rectangular box containing the title "FoodBasket Account" in bold dark blue text. Below the title are two input fields: "Username" and "Password", each with a horizontal line for text entry. To the right of the yellow box is a circular illustration of a man with brown hair, a beard, and red-rimmed glasses, wearing a red shirt. He is surrounded by several small colored circles (yellow, red, white). A magnifying glass icon with a white cloud inside is positioned over the bottom right of the man's head. Below the illustration is a light beige rectangular button with the text "Join the Basket" in dark blue.

FoodBasket Account

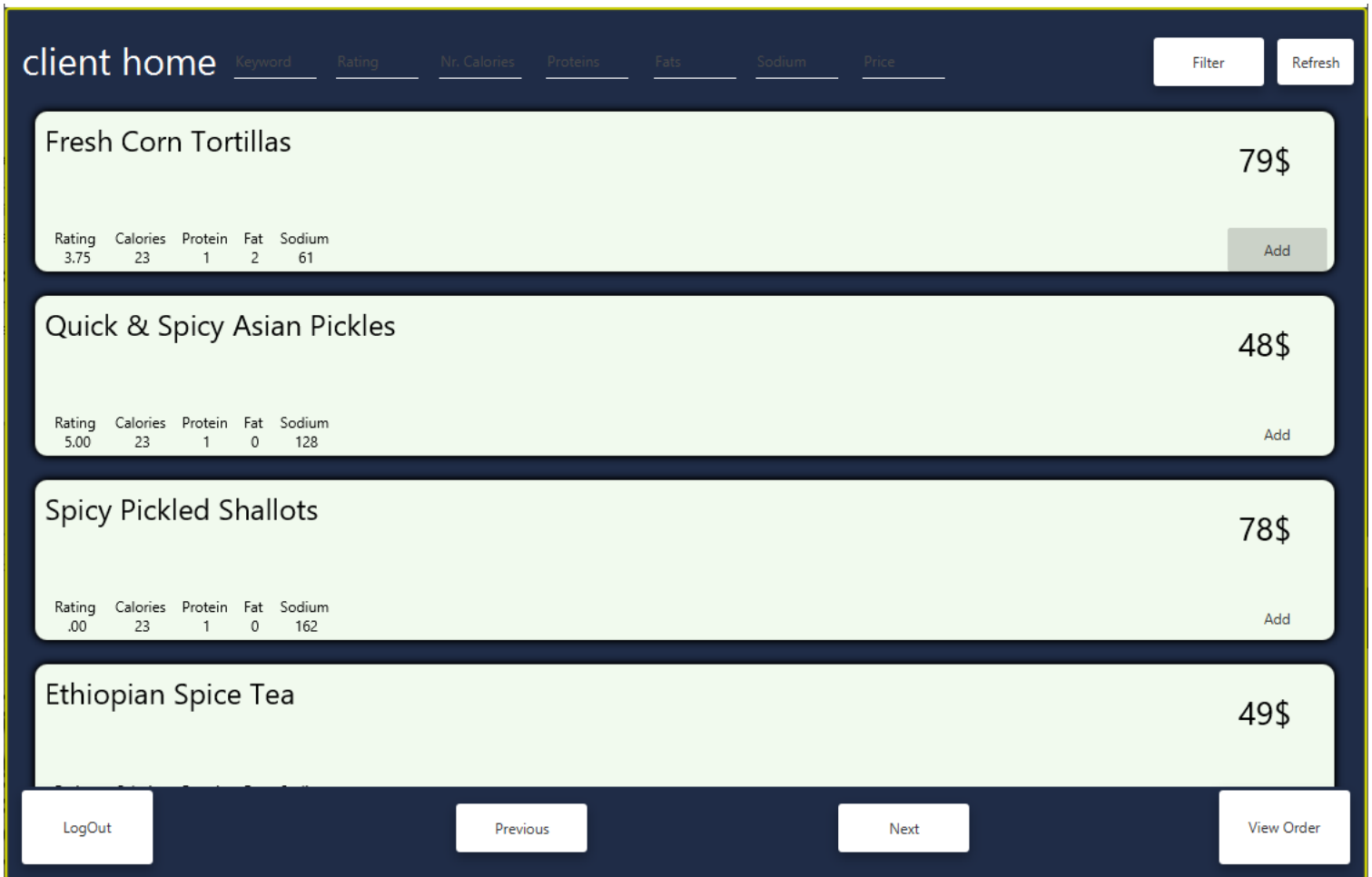
Username

Password

Join the Basket

Here the user can create a new account. If an account with the desired username already exists, the **“Join the Basket”** button will do nothing, only when valid credentials are supplied will the button work.

III. Home screen



From the main menu, a client can view all the products by navigating through them using the **Previous** and **Next** buttons. To add a item to the cart, the user must press the **Add** button. In order to simplify the shopping experience, **filters** are also available. Products can be filtered by the following criteria: Keyword, Rating, Calories, Proteins, Fats, Sodium, Price. To go to checkout, the **View Order** button must be pressed.

IV. Checkout menu

Order Summary

Fresh Corn Tortillas

79\$

Rating	Calories	Protein	Fat	Sodium
3.75	23	1	2	61

Remove

Quick & Spicy Asian Pickles

48\$

Rating	Calories	Protein	Fat	Sodium
5.00	23	1	0	128

Remove

Spicy Pickled Shallots

78\$

Rating	Calories	Protein	Fat	Sodium
.00	23	1	0	162

Remove

Close

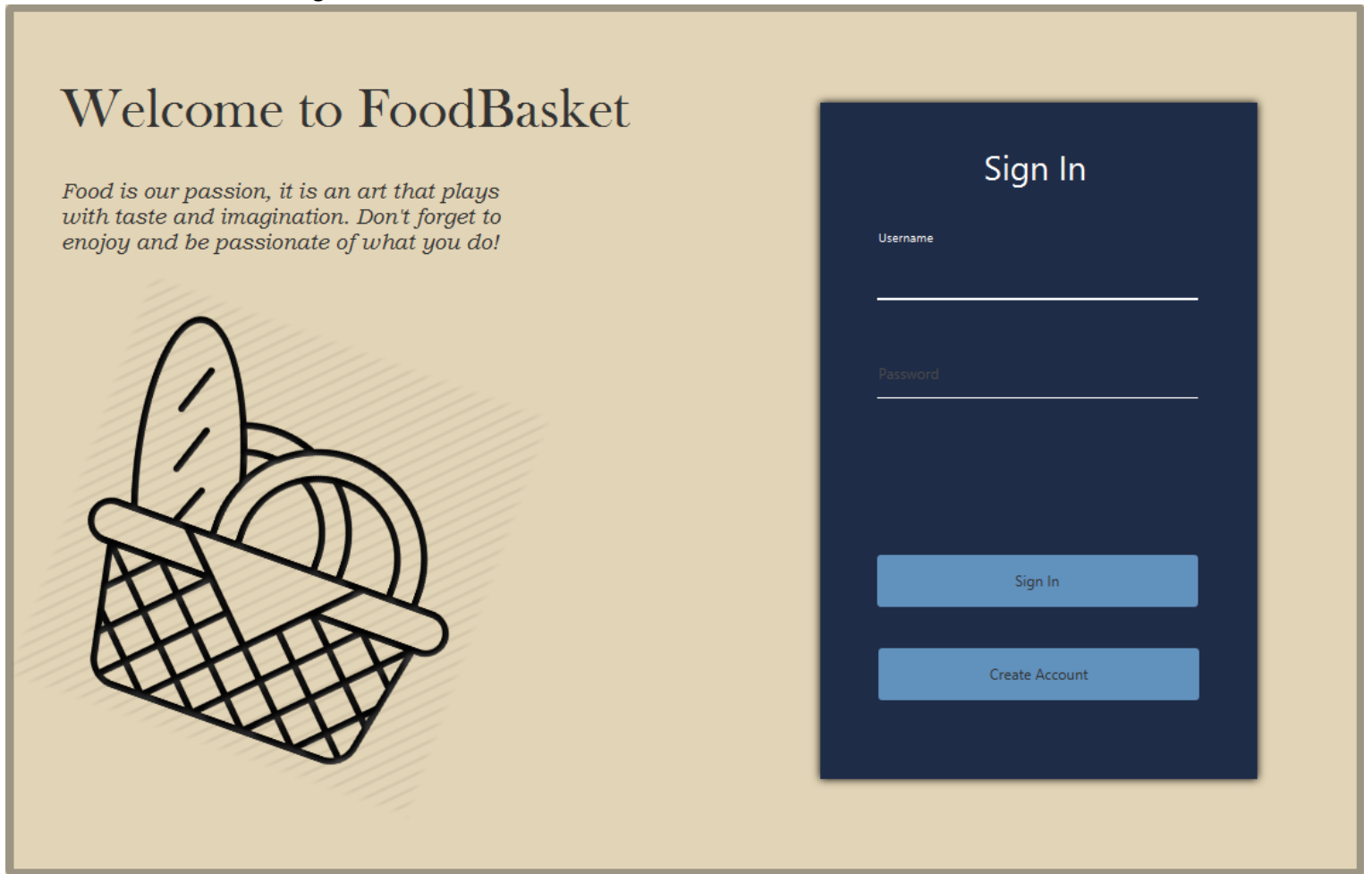
Order Total: \$205

Confirm

From the check out screen, the client can view a summary of his order. To place the order the **Confirm** button must be pressed.

We will continue with the employee role.

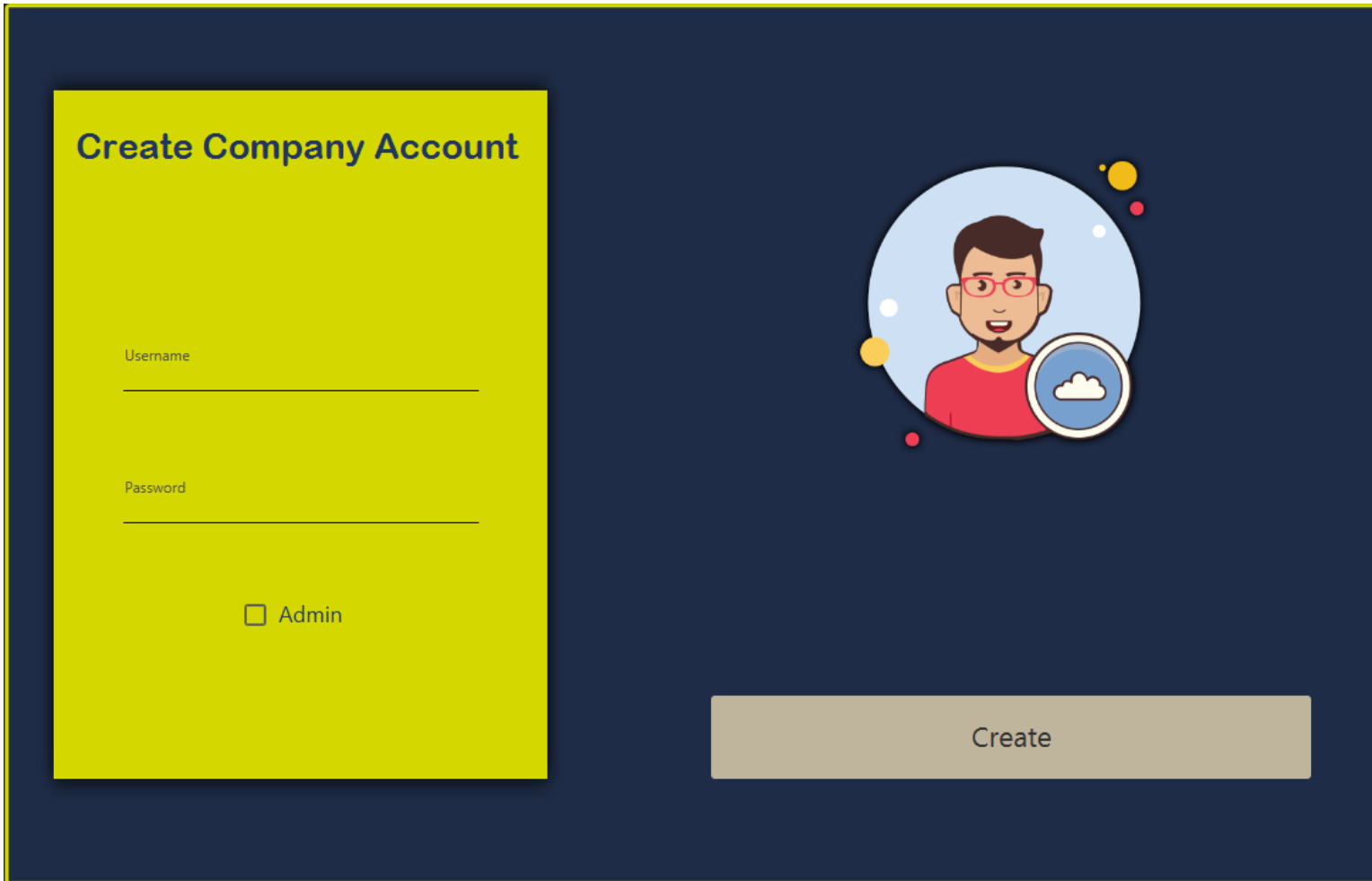
I. The login menu



The image shows a login screen for 'FoodBasket'. The background is a light beige color. On the left side, there is a large, stylized line drawing of a woven basket filled with various breads, including a long loaf and several round loaves. Above the basket, the text 'Welcome to FoodBasket' is written in a large, serif font. Below this, a smaller line of text reads: 'Food is our passion, it is an art that plays with taste and imagination. Don't forget to enjoy and be passionate of what you do!'. On the right side, there is a dark blue rectangular panel. At the top of this panel, the text 'Sign In' is written in a white, sans-serif font. Below this, there are two input fields: the first is labeled 'Username' and the second is labeled 'Password', both in a small, light blue font. Each label is followed by a white horizontal line representing the input field. At the bottom of the dark blue panel, there are two light blue buttons. The top button is labeled 'Sign In' and the bottom button is labeled 'Create Account', both in a dark blue, sans-serif font.

From the login screen the Employee can select to create a new account, or if an account is present, the employee can input his credentials into the username and password fields and sign in.

II. Create Account menu



The image shows a user interface for creating a company account. On the left, a yellow rectangular box contains the title "Create Company Account" in bold black text. Below the title are two input fields: "Username" and "Password", each with a horizontal line for text entry. Under the "Password" field is a checkbox labeled "Admin". To the right of the yellow box is a circular illustration of a man with brown hair, a beard, and red-rimmed glasses, wearing a red shirt. He is holding a blue circular icon with a white cloud inside. The background of the entire interface is dark blue. At the bottom right, there is a light gray rectangular button with the word "Create" in black text.

Create Company Account

Username

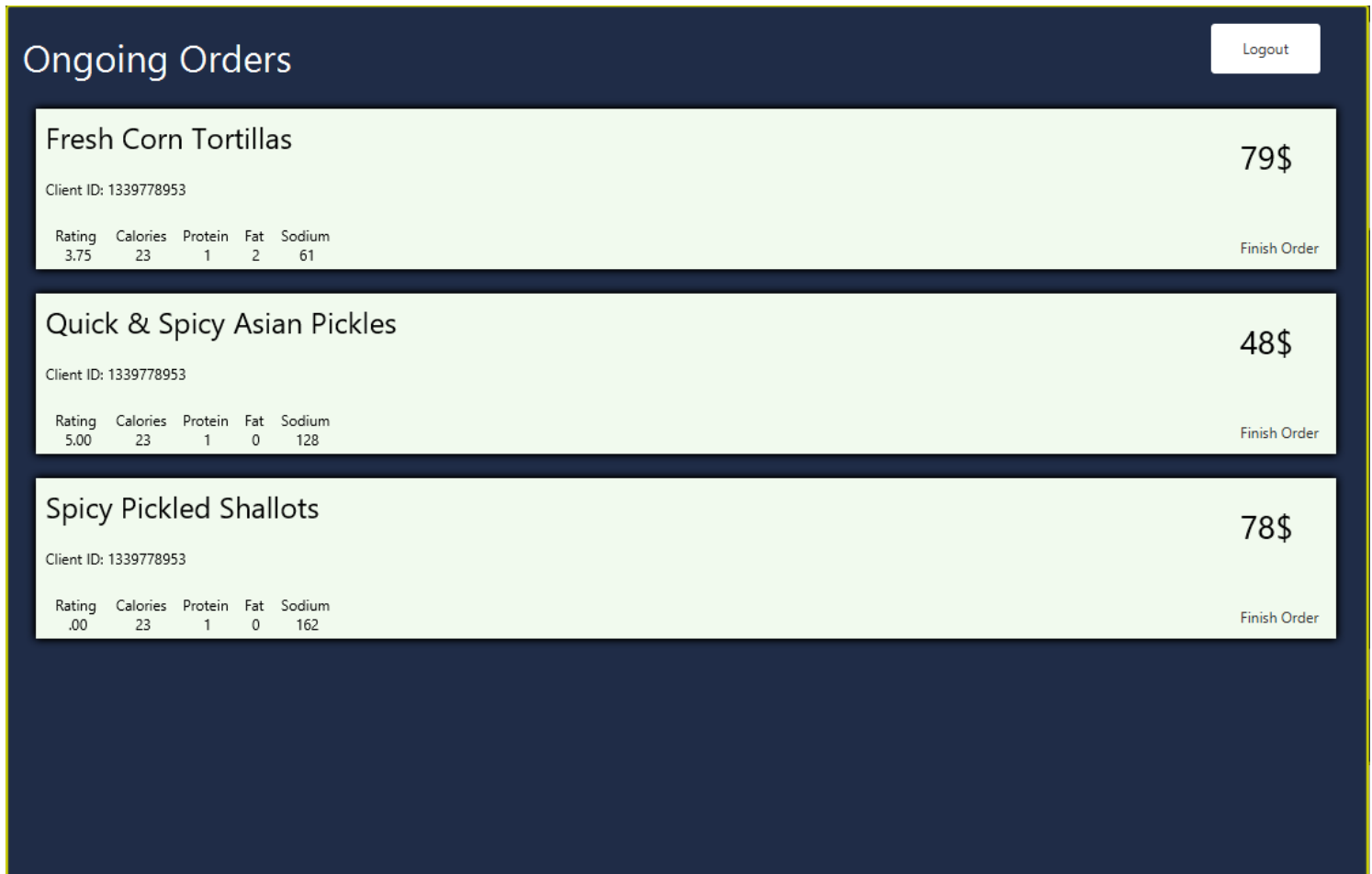
Password

☐ Admin

Create

Here the employee can create a new company account. If an account with the desired username already exists, the “**Create**” button will do nothing, only when valid credentials are supplied will the button work. The role for Admin should not be selected!

III. Home screen




From the home screen, the employee can manage the current orders. Once a order is complete it can by finished by pressing the **Finish Order** button.

Finally, we will finish off with the Admin role.

I. The login menu

FoodBasket Admin

Don't forget, with great power comes great responsibility!



Sign In

Username

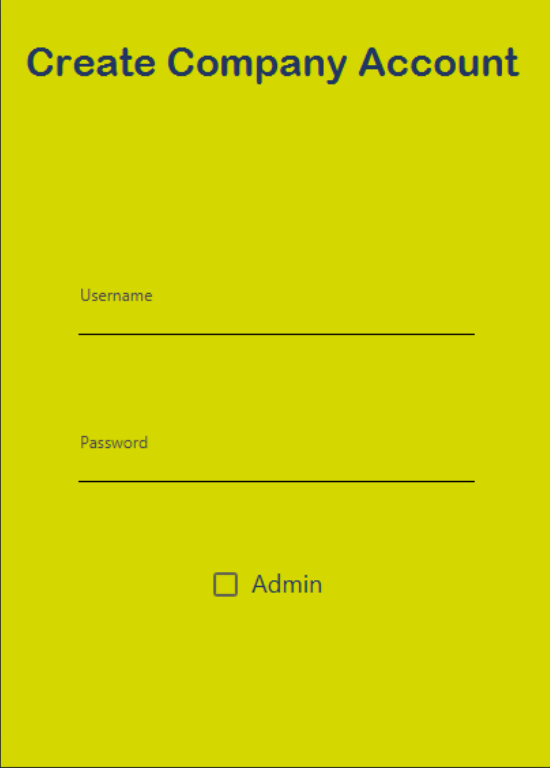
Password

Sign In

Create Account

From the login screen the Admin can select to create a new account, or if an account is present, the admin can input his credentials into the username and password fields and sign in.

II. Create Account menu




Create Company Account

Username

Password

☐ Admin

Create



Here the admin can create a new company account. If an account with the desired username already exists, the **“Create”** button will do nothing, only when valid credentials are supplied will the button work. The role for Admin should be selected!

III. Home Screen

The screenshot displays the Admin Home Screen. At the top, there are tabs for 'Products' and 'Reports'. Below this is the 'Admin' header with a search bar and filters for 'Keyword', 'Rating', 'Nr. Calories', 'Proteins', 'Fats', 'Sodium', and 'Price'. There are 'Filter' and 'Refresh' buttons. The main content area lists four products, each with a name, price, and a table of nutritional information. Each product entry has an 'Edit' button. At the bottom, there are buttons for 'LogOut', 'Previous', 'Load CSV', 'Next', 'New Composite', and 'New Product'.

Product Name	Price	Rating	Calories	Protein	Fat	Sodium
Fresh Corn Tortillas	79\$	3.75	23	1	2	61
Quick & Spicy Asian Pickles	48\$	5.00	23	1	0	128
Spicy Pickled Shallots	78\$.00	23	1	0	162
Ethiopian Spice Tea	49\$					

From the main menu, the admin can perform a multitude of actions. We will start with loading a list of products that are under the .csv format using the **Load CSV** button. After the products are loaded the admin can view them by using the **Previous** and **Next** buttons, he can also filter them. If the admin wants to edit a product, he can press the **Edit** button, a dialog will the appear. If a brand new product is desired, the admin has two options: 1. Create a composite product or 2. Create a new product from the ground up.

IV. Edit/Delete Product

The screenshot shows an Admin interface with a modal dialog for editing a product. The dialog is titled 'Fresh Corn Tortillas' and contains the following fields:

- Title
- Rating
- Calories
- Sodium
- Fat
- Protein
- Price

Below the fields are two buttons: 'Save' (dark blue) and 'Delete' (red).

The background interface shows a list of products with their details and prices:

Product Name	Rating	Calories	Protein	Fat	Sodium	Price
Fresh Corn Tortillas	3.75	23	1	2	61	79\$
Quick & Spicy Asian P	5.00	23	1	0	128	48\$
Spicy Pickled Shallots	.00	23	1	0	162	78\$
Ethiopian Spice Tea						49\$

From this dialog, the admin can alter a product or delete it.

V. New Composite Product

Composite

Keyword

Filter

Beef Stock

Add

Master Stock Chicken

Add

Fish Stock

Add

Hot-and-Sour Cabbage Salad

Add

Potato Samosa Tartlets

Add

Fresh Corn Tortillas

Remove

Arugula Salad with Heirloom Tomatoes and Red Onion

Remove

Red Pepper Sauce

Remove

Previous

Next

Create Product

From this screen the admin can create a new composite product. There are two lists, the one on the left presents all the products to choose from to compose the new product. Once all the desired sub-Products are selected, the Create Product button must be pressed to finish the process.

VI. Report Menu

Products

Reports

Report Management

Time Interval of the Orders – a report that shows the orders performed between a given start hour and a given end hour regardless the date

Start Hour

1

End Hour

12

☒

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

Amount

3

☒

The clients that have ordered more than a specified number of times and the value of the order was higher than the specified amount

Number of Orders


2

Order Value

47

☐

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



< May >

< 2021 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

☒

Generate Report

If the report tab is selected, the admin is presented with the following screen. From here we can generate reports in a dynamic fashion, this is done by check marking the desired report. Once the Generate report button is pressed, a report file will be written.

4. Implementation

The implementation of this project can be split into 4 categories, as per the layered architecture requires. We will work our way up, from the **Data Access Layer** to the **presentation layer**.

a) Data Access Layer

This layer's purpose is to manage external project files.

- FileWriter
This class oversees creating a new file and setting the contents of it.
- Serializator
This class is used to save certain lists to a file so that when the application is closed the app data is not lost. When the application reopened, the data is then loaded back.

b) Business Logic Layer

This layer's purpose is to provide higher level methods that the UI needs. (For example, we do not want the UI to manage the users and their credentials)

- DeliveryService
This class provides the functionality to manage the core delivery services, such as menu and orders management
- IDeliveryServiceProcessing
This interface defines the core functions that are necessary for the delivery service.
- UserManagement
This class manages all the users credentials with their roles. I also offers some helper methods such as validateLogin, this method returns a Boolean that says if certain credentials are valid.
- MenuItem
This class is covered in the Data Structures section.
- BaseProduct
This class is covered in the Data Structures section.
- CompositeProduct
This class is covered in the Data Structures section.
- Order
This class is covered in the Data Structures section.
- User
This class is covered in the Data Structures section.

c) Presentation Layer

This layer's purpose is to provide the user interface. Because the interface is rather large in this project, we will split it into 3 parts, client, employee, admin.

Client:

It is composed of the following pairs of java controllers and .FXML files:

- ClientHome, clientHome.fxml
- CreateAccountClient, createAccountClient.fxml

- FinalizeOrderDialog, finalizeOrderDialog.fxml
- LoginClient, loginClient.fxml
- MenuItemView, menuItemView.fxml

These files compose the Client pages and all their functionality.

Employee:

It is composed of the following pairs of java controllers and .FXML files:

- LoginEmployeeController, loginEmployee.fxml
- MenuItemView, menuItemView.fxml
- EmployeeHome, EmployeeHome.fxml

These files compose the Employee pages and all their functionality.

Admin:

It is composed of the following pairs of java controllers and .FXML files:

- AdminCompositeProduct, adminCompositeProduct.fxml
- AdminHome, adminHome.fxml
- CreateAccountManagement, CreateAccountManagement.fxml
- EditProductAdmin, EditProductAdmin.fxml
- NewProductAdmin, newProductAdmin.fxml
- MenuItemView, menuItemView.fxml
- SimpleProductView, simpleProductView.fxml

These files compose the Admin pages and all their functionality.

5. Results

For testing the application, I wanted to see if all the required operations functioned properly. For this the products were read from the CSV file, multiple orders and products were created, such that we may service the orders and generate the bills and reports. After those operations were carried out, the application was closed and reopened to test out the serialization.

6. Conclusion

In conclusion, this assignment really helped show the possibilities of using an observer pattern as well as using multiple design patterns in the same application. The user interface was also much larger than any previous projects, which helped hone in on my UI/UX skills.

7. Bibliography

<https://www.vogella.com/tutorials/DesignPatternObserver/article.html>

<https://www.geeksforgeeks.org/iterate-map-java/>

<https://github.com/ssshahine/JFoenix/blob/master/demo/src/main/java/demos/components/DatePickerDemo.java>

https://rosettacode.org/wiki/Assertions_in_design_by_contract#Java

<https://stackoverflow.com/questions/35486826/transform-and-filter-a-java-map-with-streams>

<https://stackoverflow.com/questions/907170/java-getminutes-and-gethours>

<http://tutorials.jenkov.com/java/lambda-expressions.html>

<https://coolers.co/>