Furniture Deals

Analysis and Design Document

Student:Urcan Miruna - Stefania

**Group:30238**

Table of Contents

[**1. Requirements Analysis**](#_qciaqh5gxo9t) **2**

[**Assignment Specification**](#_tyjcwt) **3**

[**Functional Requirements**](#_3dy6vkm) **3**

[**Non-functional Requirements**](#_1t3h5sf) **3**

[**2. Use-Case Model**](#_wjffpxe4o1l3) **3**

[**3. System Architectural Design**](#_4sbmqyx0k1sv) **5**

[**4. UML Sequence Diagrams**](#_17dp8vu) **6**

[**5. Class Design**](#_37aj2sxz686c) **6**

[5.1 Design Patterns Description](#_i5tzg8liao28) 6

[5.2 UML Class Diagram](#_3zp950cyo3eq) 6

[**6. Data Model**](#_b14llhtnb4f1) **9**

[**7. System Testing**](#_lnxbz9) **9**

[**8. Bibliography**](#_dsn5qcxmfg6d) **10**

# 1. Requirements Analysis

## Assignment Specification

The task is about building a search engine for furniture products. The client ( or user) is able to login or create an account in case he does not have one. He can search for different products added by a staff member. For an easier search, he can filter the products by their type, price or name. In this case, the product list will change, in order to have a better vision for what the user is looking for.

A deal can be proceeded only if available, which means there are enough products to cover the order for user. The payment process is automatically requested when the user validate the order.

The payment is validated by the staff. In this case, the order will remain in the system until a staff member validate it. This process can be seen by the user via Invoice and Order History: if the payment is not validated yet, the order will be seen in the Invoice category. On the other hand, if the payment if validated, the order details will appear in the Order History section.

## Functional Requirements

There are two kinds of clients for this application: users and staff .

The users operations required are: login, addOrder, filter by name, type ( kitchen, office, livingroom, bedroom) or price, invoice, order history, logout.

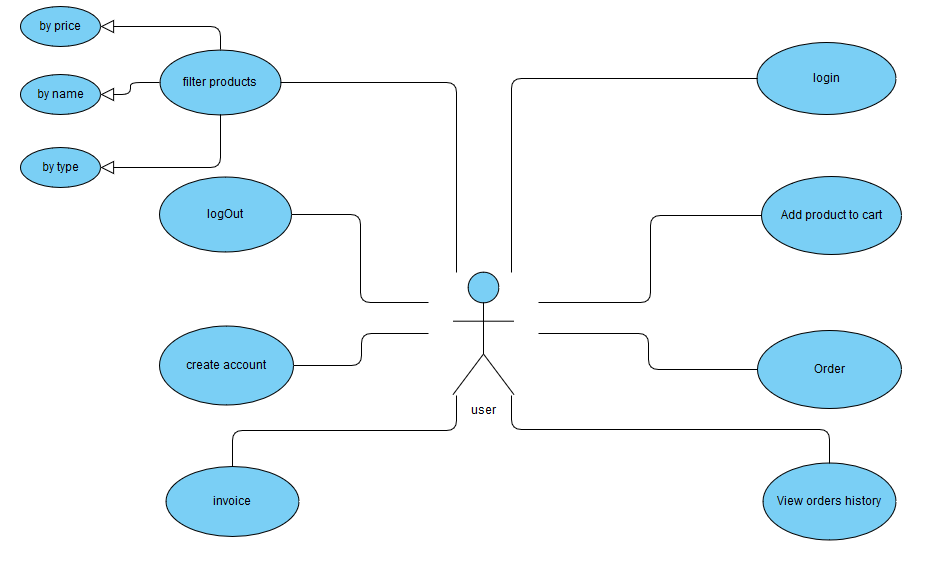
The staff operations required are: login, validate order, delete product, add product, update product, apply discount, logout.

Both categories of clients are able to logIn. The logout is made automatically when the client closes the window.

## Non-functional Requirements

We were asked to use an OOP language (Java), implement and test the application. Another requirement was to use a layered architecture and to use a database for storage. All the inputs of the application has to be validated. The next request was to build a feature for the staff member, in order to apply discounts. For this requirement, we were asked to use Factory Method. This feature can change the overall price.

# 2. Use-Case Model



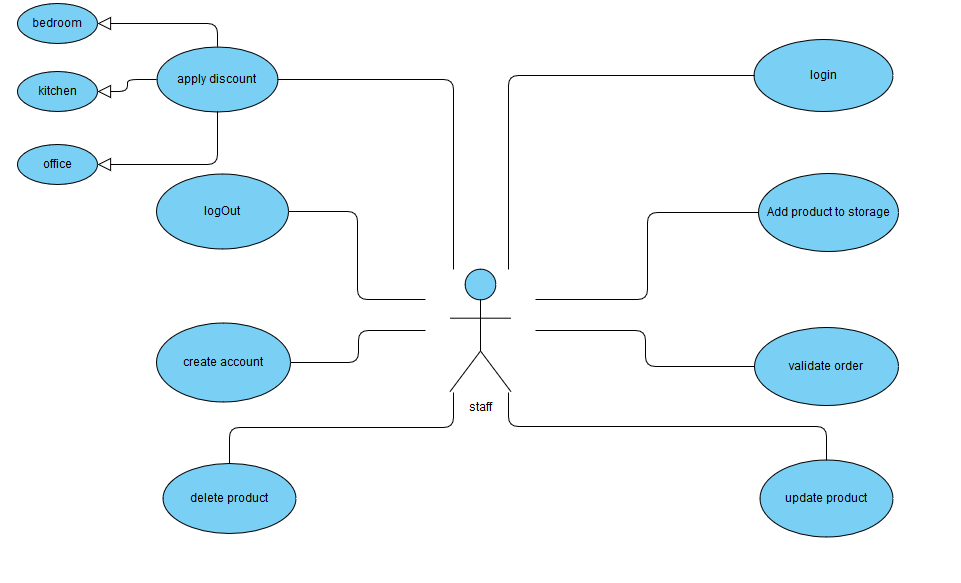
Use case: order

Level : user - goal level

Primary actor: user

Main success scenario: logIn successfully, choose product and make order.

Extensions: there are no deals



Use case: validate order

Level : staff-goal level

Primary actor: staff

Main success scenario: login, choose order from user and validate data

Extensions: there are no orders

# 3. System Architectural Design

**3.1 Architectural Pattern Description**

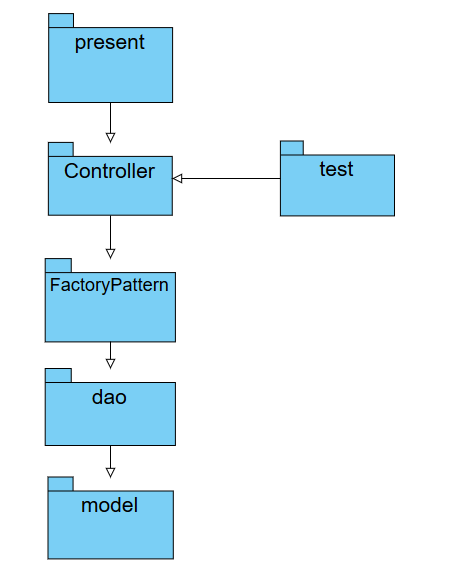
An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture. he most common pattern is the layered pattern. Each layer has a specific responsibility.

For this assignment, I organized the system using five layers. The first layer, model, is describing the model of data, dao, where the connection with the database is made, Controller, in which are implemented the functionalities for this assignment. FactoryPattern describes the pattern implemented for discounts. In the next layer, present, I created the graphical interface.

**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

Package diagram:



# 4. UML Sequence Diagrams

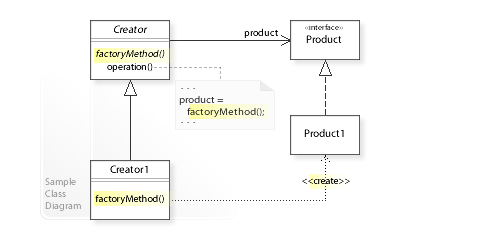
*[Create a sequence diagram for a relevant scenario.]*

# 5. Class Design

## 5.1 Design Patterns Description

For this assignment, I used layered architecture: categorized after their role in the application, all classes are organized in packages. Therefor, every package has a specific role in the implementation.

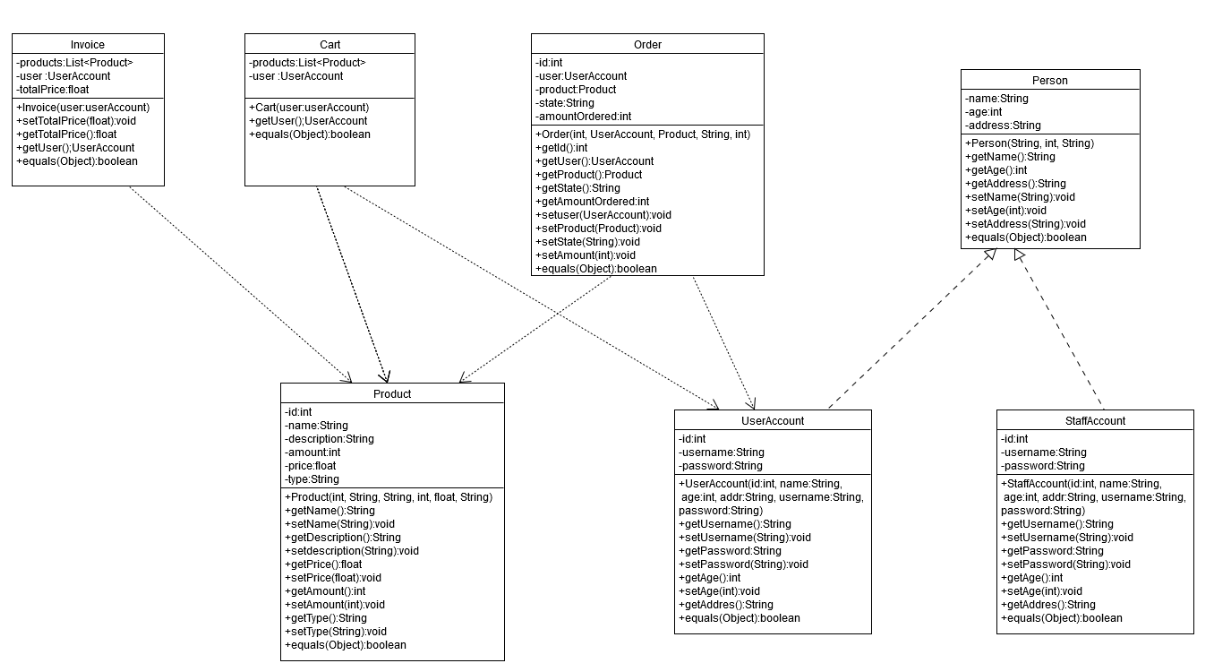
Factory pattern is a creational pattern that uses factory methods to deal with the problem of [creating objects](https://en.wikipedia.org/wiki/Object_creation) without having to specify the exact [class](https://en.wikipedia.org/wiki/Class_(computer_programming)) of the object that will be created. This pattern defines an interface for creating an object, but let the subclases decide which class to instantiate.

**

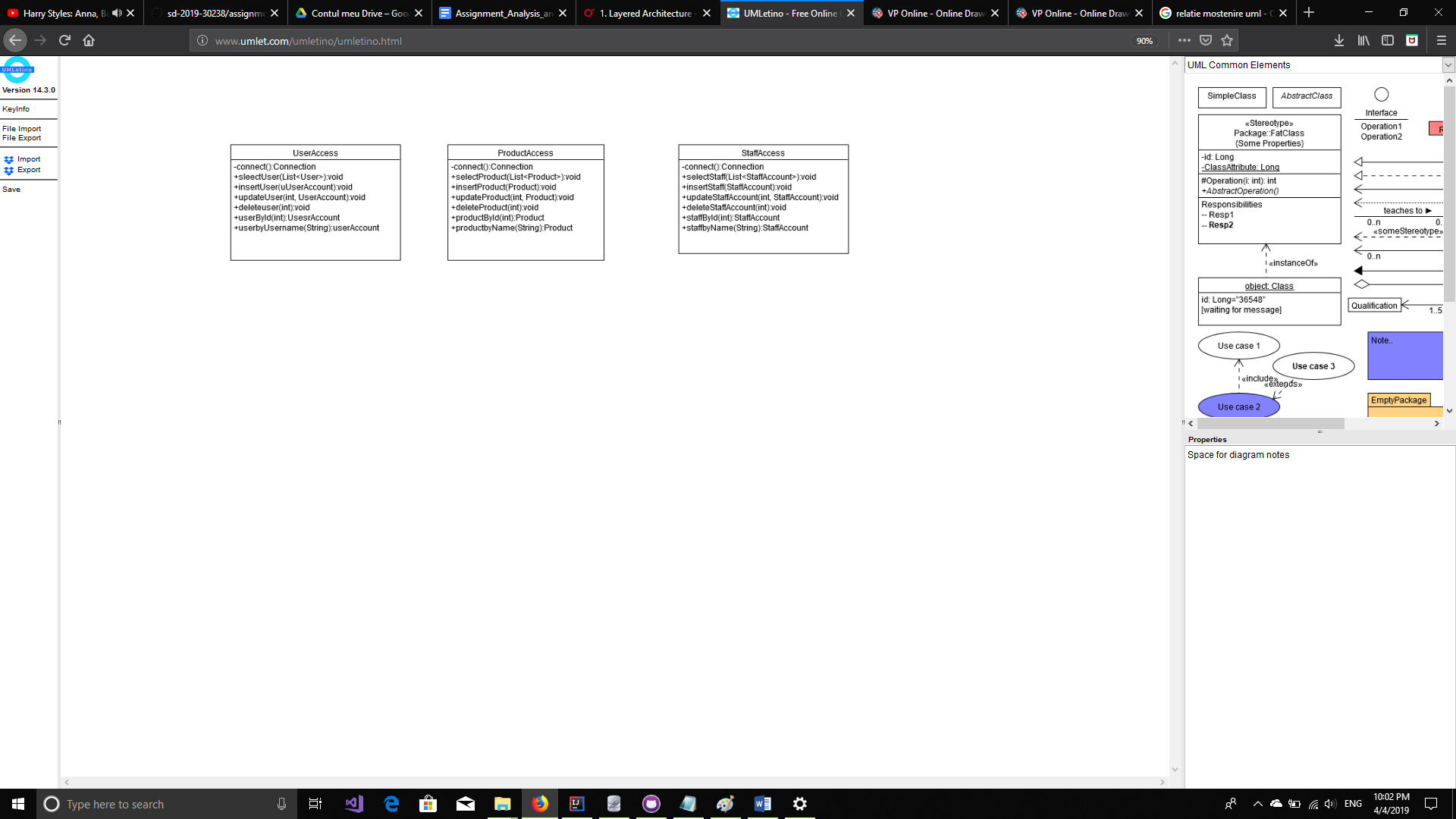
## 5.2 UML Class Diagram

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

Model Diagram



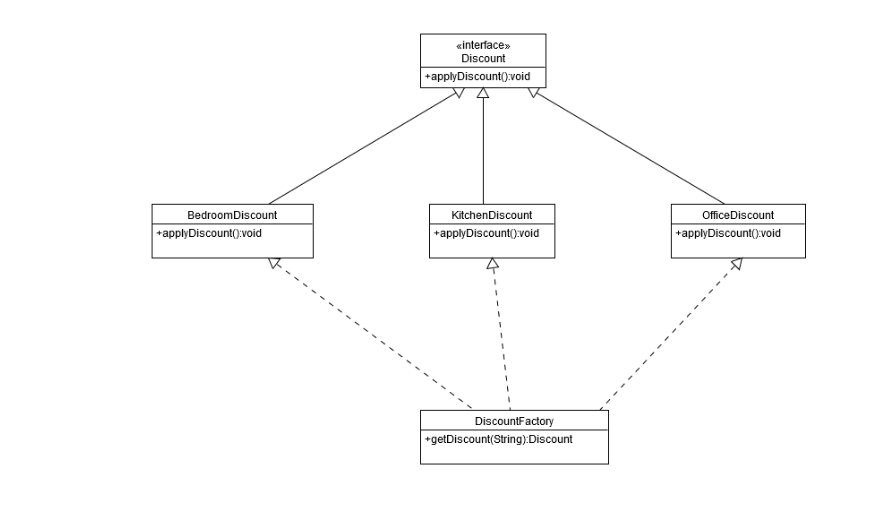
Dao diagram



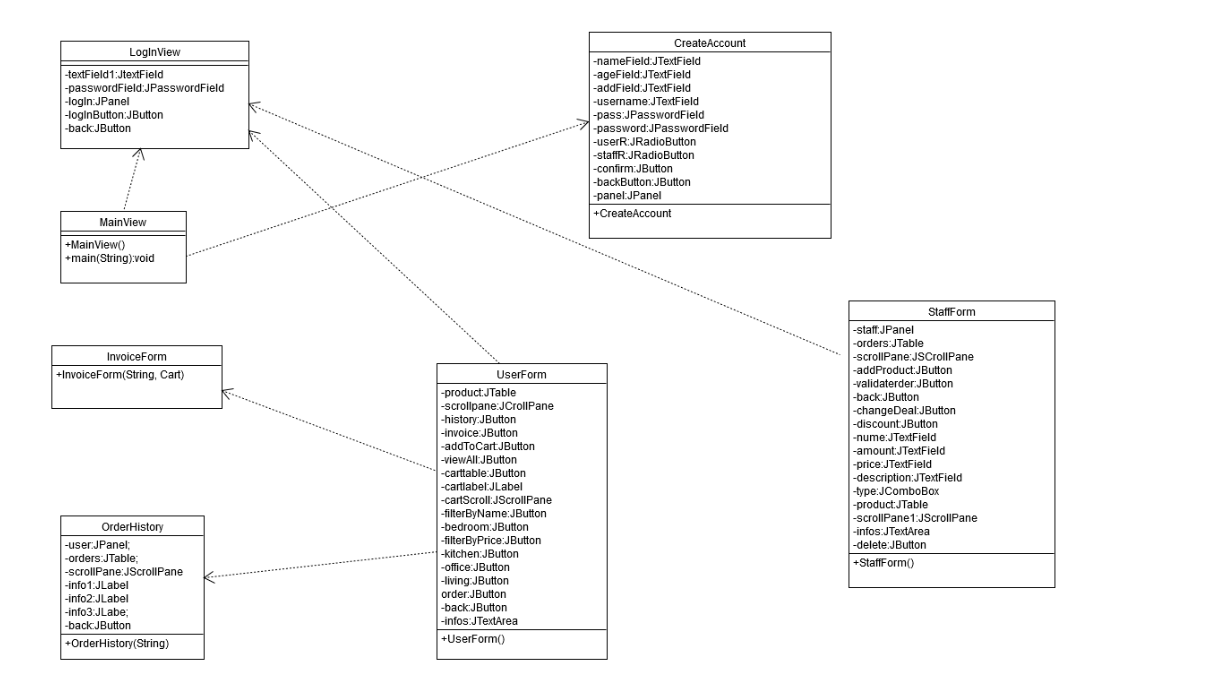
Controller Diagram



Factory Diagram



Presentation Diagram



# 6. Data Model

Data Model is presented in model package. each class(UserAccount, StaffAccount, Product, Invoice, Cart, Person and Order) are the objects used for the implementation of the application. All classes from this package are directly communicating with classes from the dao package, which realize the connection with the database. Each class has more simple attributes needed for representing the data.

In order to create this application, I used a database with five tables, describing the needed attributes to describe data .

# 7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

# 

# 8. Bibliography

Architectural pattern <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

Design Pattern

<https://en.wikipedia.org/wiki/Software_design_pattern>

Factory Method

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>