Student:Lecu Alexandru-Călin

**Group:30234**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use Swing/C# API to design and implement an application for the order managers of a furniture manufacturer. The application should have two types of users (a regular user represented by the  order manager and an administrator user) which have to provide a username and a password in order to use the application.

# Functional Requirements

The regular user can perform the following operations:

* Add/update/view order information (customer, shipping address, identification number, delivery date, status.).
* Create/update/delete/view product information (title, description, color, size, price, stock etc).
* Add products to order and update order value and stock accordingly.

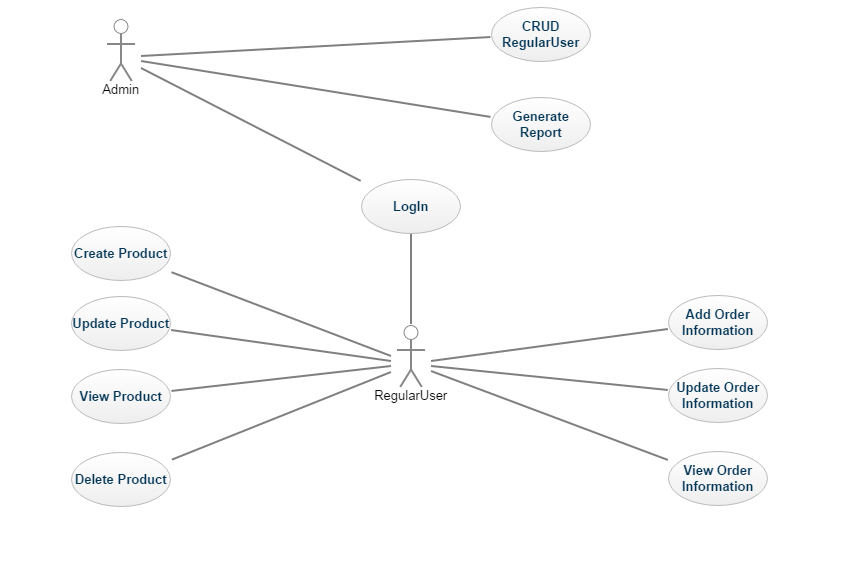
The administrator user can perform the following operations:

* CRUD on employees’ information.
* Generate reports for a particular period containing the activities performed by an employee.

# Non-functional Requirements

* [Open source](https://en.wikipedia.org/wiki/Open_source" \o "Open source)
* [Operability](https://en.wikipedia.org/wiki/Operability)
* [Performance](https://en.wikipedia.org/wiki/Computer_performance) / response time ([performance engineering](https://en.wikipedia.org/wiki/Performance_engineering))
* [Platform](https://en.wikipedia.org/wiki/Platform_(computing)) compatibility
* [Price](https://en.wikipedia.org/wiki/Price)
* [Privacy](https://en.wikipedia.org/wiki/Privacy)
* [Portability](https://en.wikipedia.org/wiki/Software_portability)
* [Quality](https://en.wikipedia.org/wiki/Quality_(business)) (e.g. faults discovered, faults delivered, fault removal [efficacy](https://en.wikipedia.org/wiki/Efficacy))
* [Readability](https://en.wikipedia.org/wiki/Computer_programming#Readability_of_source_code)
* [Stability](https://en.wikipedia.org/wiki/Stability_Model)
* [Supportability](https://en.wikipedia.org/wiki/Serviceability_(computer))
* [Testability](https://en.wikipedia.org/wiki/Software_testability)
* [Transparency](https://en.wikipedia.org/wiki/Transparency_(behavior))

2. Use-Case Model



3. System Architectural Design

**3.1 Architectural Pattern Description**

Three-tier architecture:

* **Presentation tier**

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

* **Application tier (**[**business logic**](https://en.wikipedia.org/wiki/Business_logic)**, logic tier, or middle tier)**

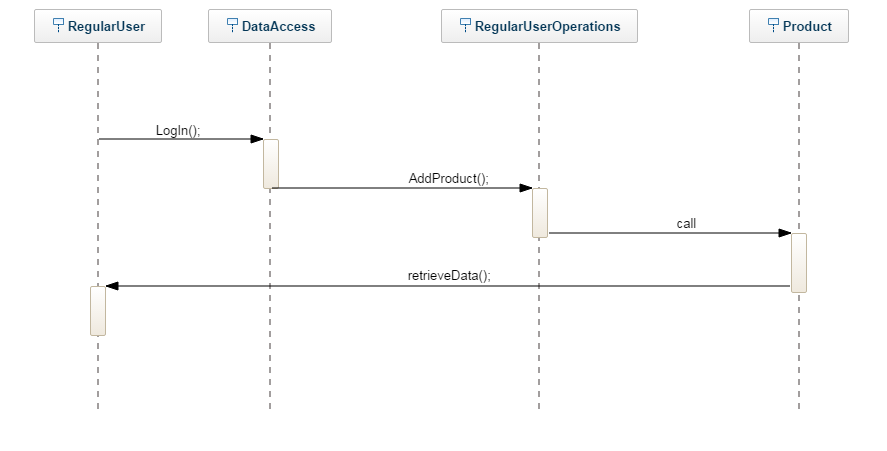
The logical tier is pulled out from the presentation tier and, as its own layer, it controls an application’s functionality by performing detailed processing.

* **Data tier**

The data tier includes the data persistence mechanisms (database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an [API](https://en.wikipedia.org/wiki/Application_programming_interface) to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

**3.2 Diagrams**

4. UML Sequence Diagrams

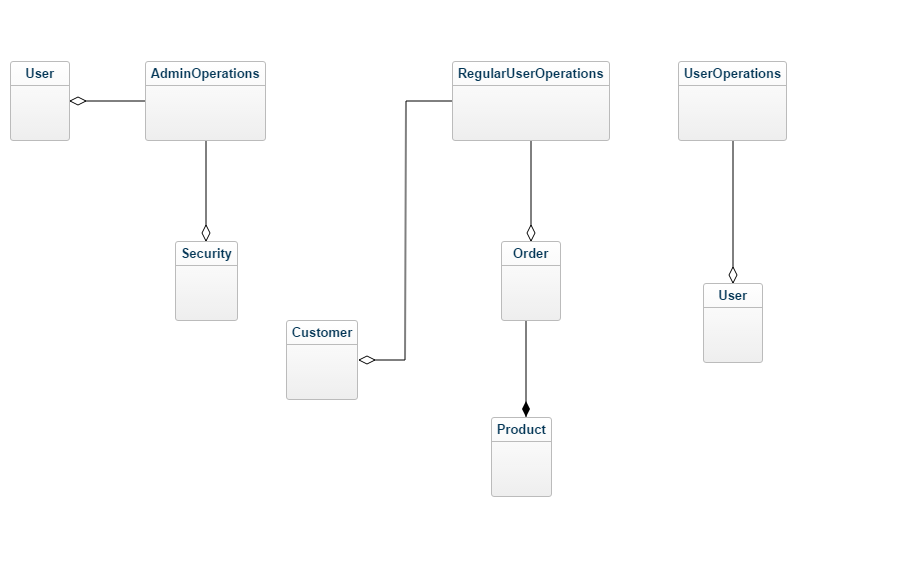


5. Class Design

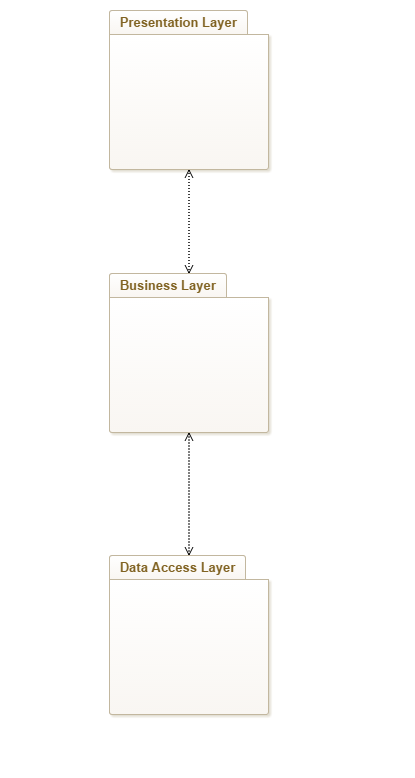
**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

**5.2 UML Class Diagram**



**5.3 UML Package Diagram**



6. Data Model

*[Present the data models used in the system’s implementation.]*

7. System Testing

[TestMethod()]

public void AddProductTest()

{

Product product = new Product();

product.ID = 9;

product.title = "asdf";

product.description = "asdf";

product.color = "asdfafs";

product.size = 12;

product.price = 13;

product.stock = 11;

IList<Product> products = new List<Product>();

DataAccessProduct dal = new DataAccessProduct();

products = dal.RetrieveProducts();

for(int i =0;i<products.Count;i++)

{

if(products.ElementAt(i).ID == product.ID)

{

Assert.Fail();

}

}

dal.AddProduct(product);

Assert.AreNotEqual(products.Count, dal.RetrieveProducts().Count, 0, "Eroare");

}

8. Bibliography