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.

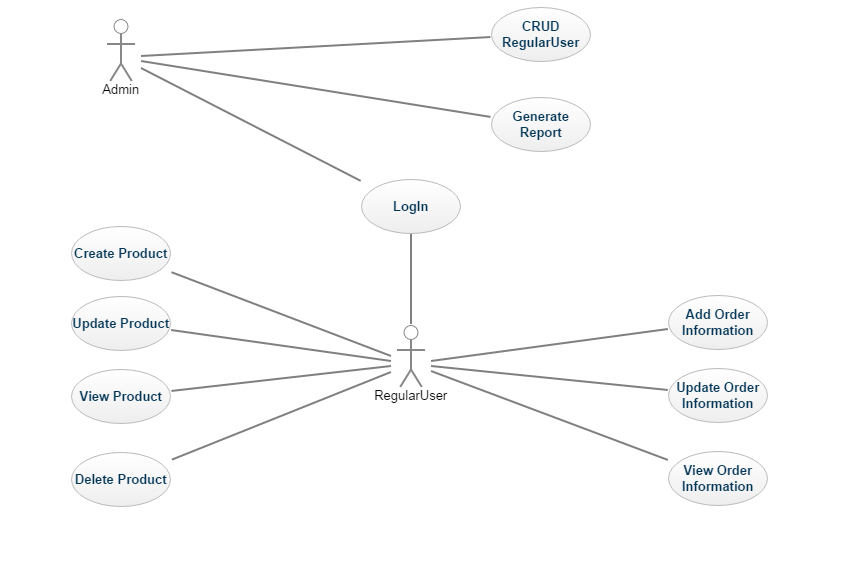
The administrator user can perform the following operations:

* 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)
* [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

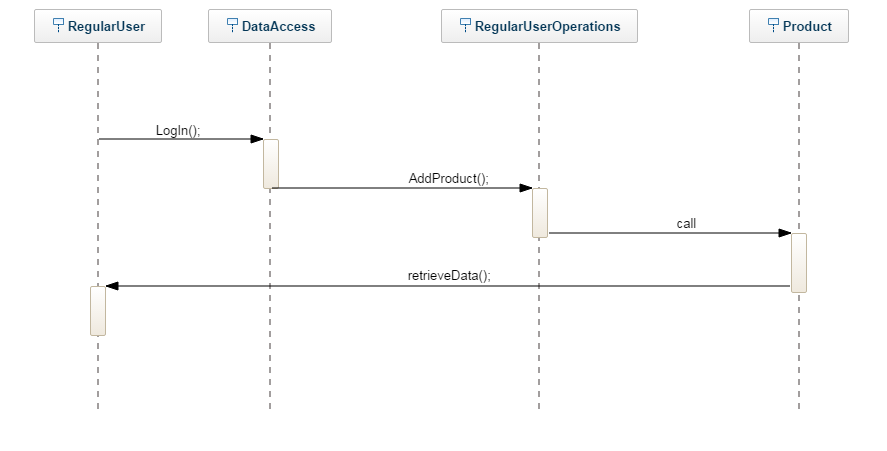
**3.1 Architectural Pattern Description**

MVP is a user interface [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern_(computer_science)) engineered to facilitate [automated](https://en.wikipedia.org/wiki/Test_automation) [unit testing](https://en.wikipedia.org/wiki/Unit_testing) and improve the [separation of concerns](https://en.wikipedia.org/wiki/Separation_of_concerns) in presentation logic:

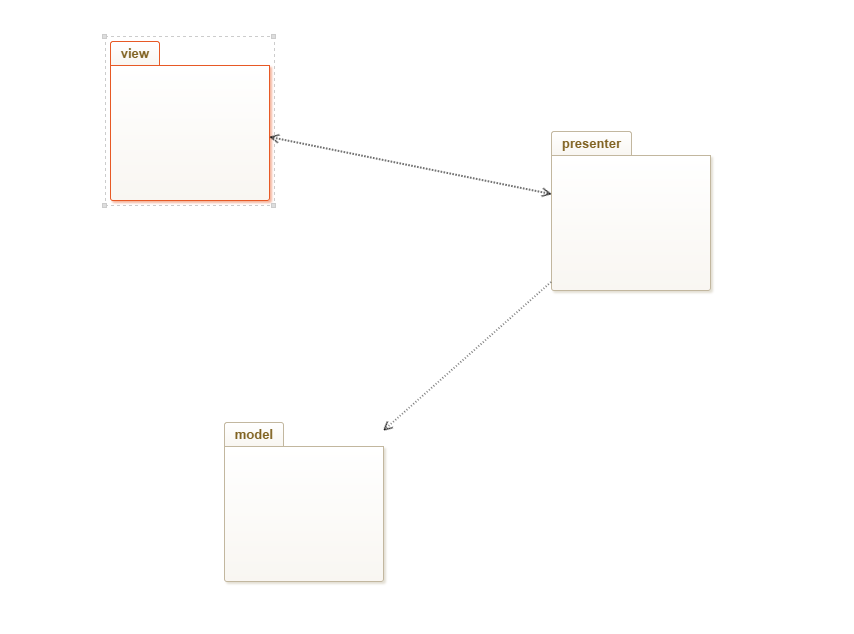
* The *model* is an interface defining the data to be displayed or otherwise acted upon in the user interface.
* The *presenter* acts upon the model and the view. It retrieves data from repositories (the model), and formats it for display in the view.
* The *view* is a passive interface that displays data (the model) and routes user commands ([events](https://en.wikipedia.org/wiki/Event_(computing))) to the presenter to act upon that data.

**3.2 Diagrams**

4. UML Sequence Diagrams



5. Class Design

**5.1 UML Package Diagram**

**5.2 UML Class Diagram**

