Online Book System

Analysis and Design Document

Student: Biris Alexandra

**Group: 30432**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 19/04/18 | 1.0 | First version | Biris Alexandra |
| 25/04/18 | 1.1 | Added iteration 1.2 | Biris Alexandra |
| 21/05/18 | 1.2 | Changed iteration 1.1 and 1.2 | Biris Alexandra |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

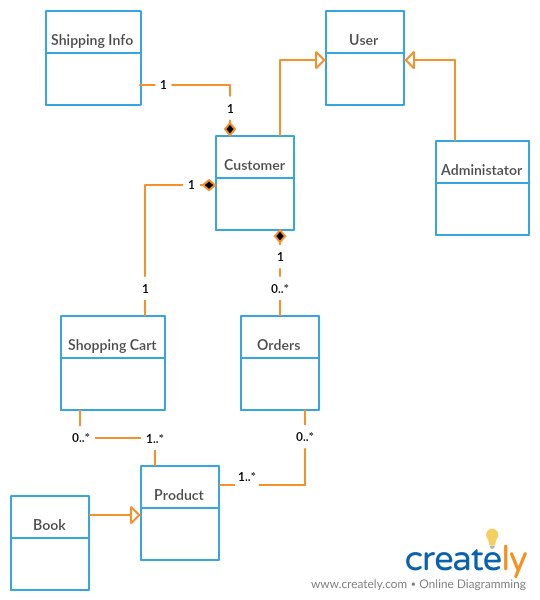
V. Bibliography 5

# Project Specification

The Online book selling application is a tool that helps customers to buy the books online. The advantages of such a system include the stock (physical stores are restricted by space limitations and budgets), the convenience (one can easily order books without the restrictions of specific store timings and long queues while sitting in the comfort of your home) and the cost (customers can hunt for great deals, big discounts and free shipping).

# Elaboration – Iteration 1.1

# Domain Model



# Architectural Design

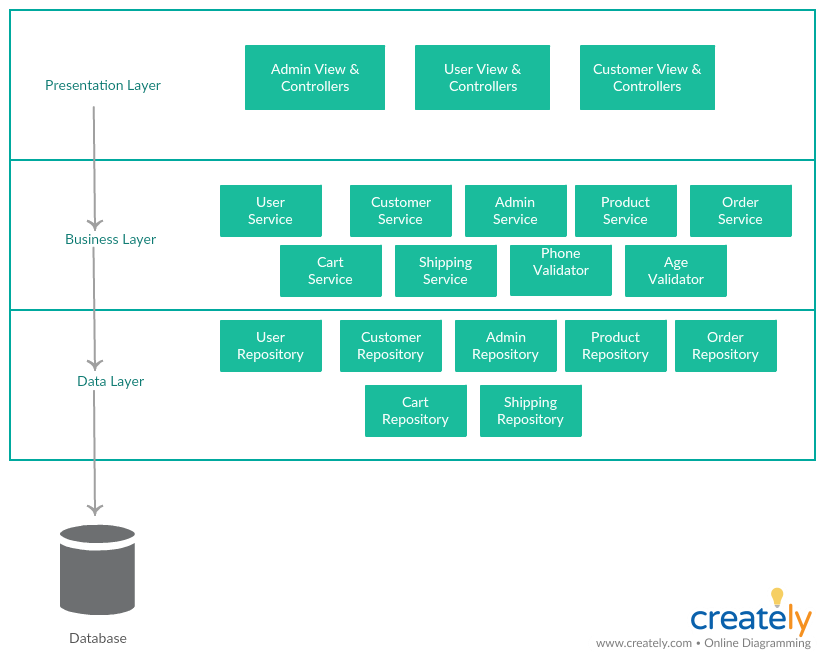
## Conceptual Architecture

The architectural pattern used is Layers. Components within this pattern are organized into horizontal layers, each layer performing a specific role within the application. Although it does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of three standard layers: presentation, business and database. Data layer will also be divided into entity and repository.

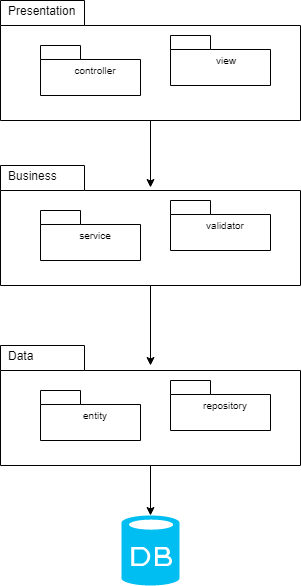
1. Presentation layer: responsible for handling all user interface and browser communication logic
2. Business layer: responsible for executing specific business rules associated with the request
3. Database layer: responsible for executing SQL statements to retrieve the corresponding data and pass it back up in the business layer.

MVC Pattern stands for Model-View-Controller Pattern and will be used to create the presentation of the project. This pattern is used to separate application's concerns as follows:

1. Model - This part of the framework is to store the data of the application, such as databases, text data, files and/or other web resources.
2. View - This is the graphical user interface of the application. That would contain different buttons, text boxes and other controls to let the user interact with the application to complete his projects depending on the sort of the software he is using.
3. Controller - The actual back-end code constitutes the controller of the framework. A controller controls the data coming from the users, or going to the user from a model.

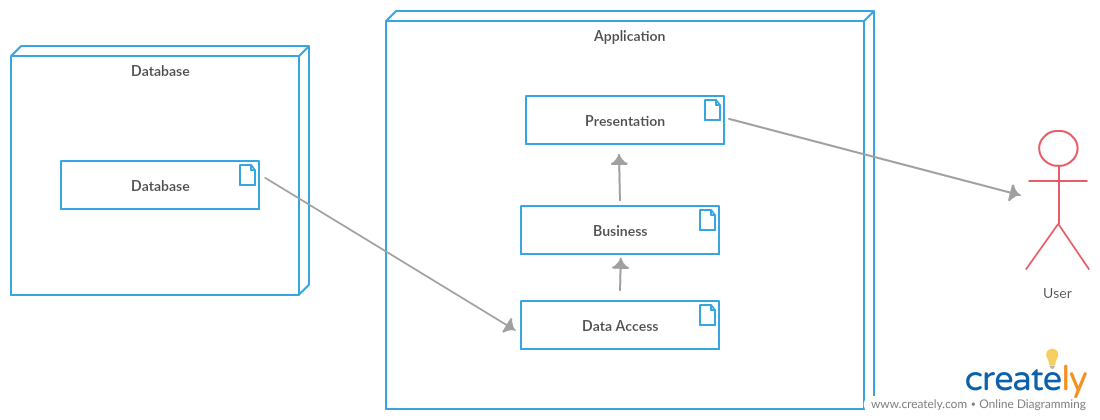


## Package Design

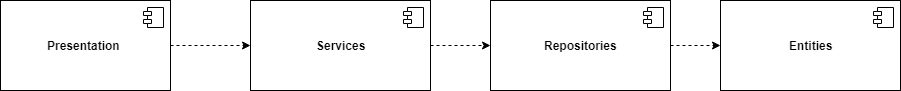


## Component and Deployment Diagrams

# Deployment diagram:



# Component diagram:

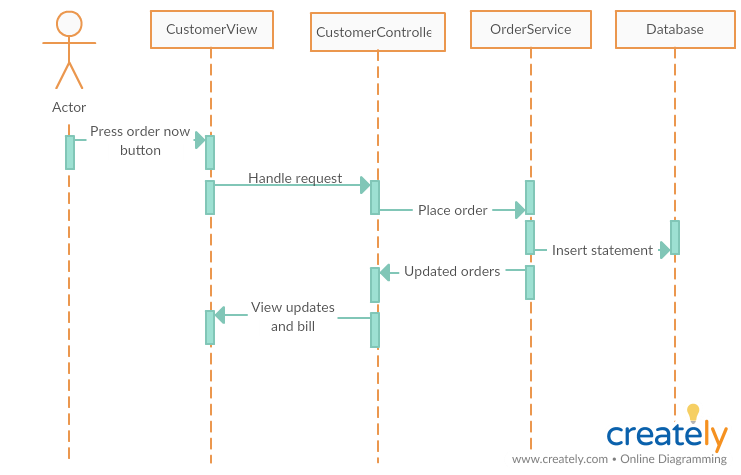


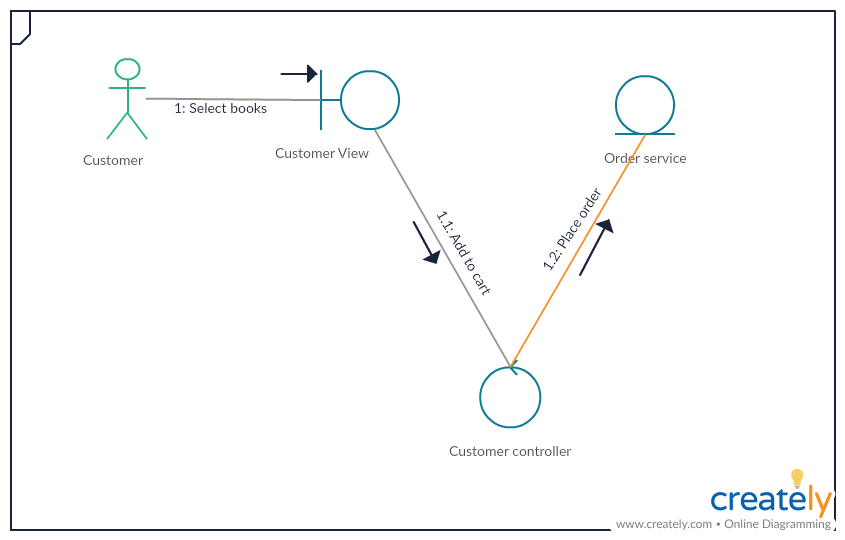
# Elaboration – Iteration 1.2

# Design Model

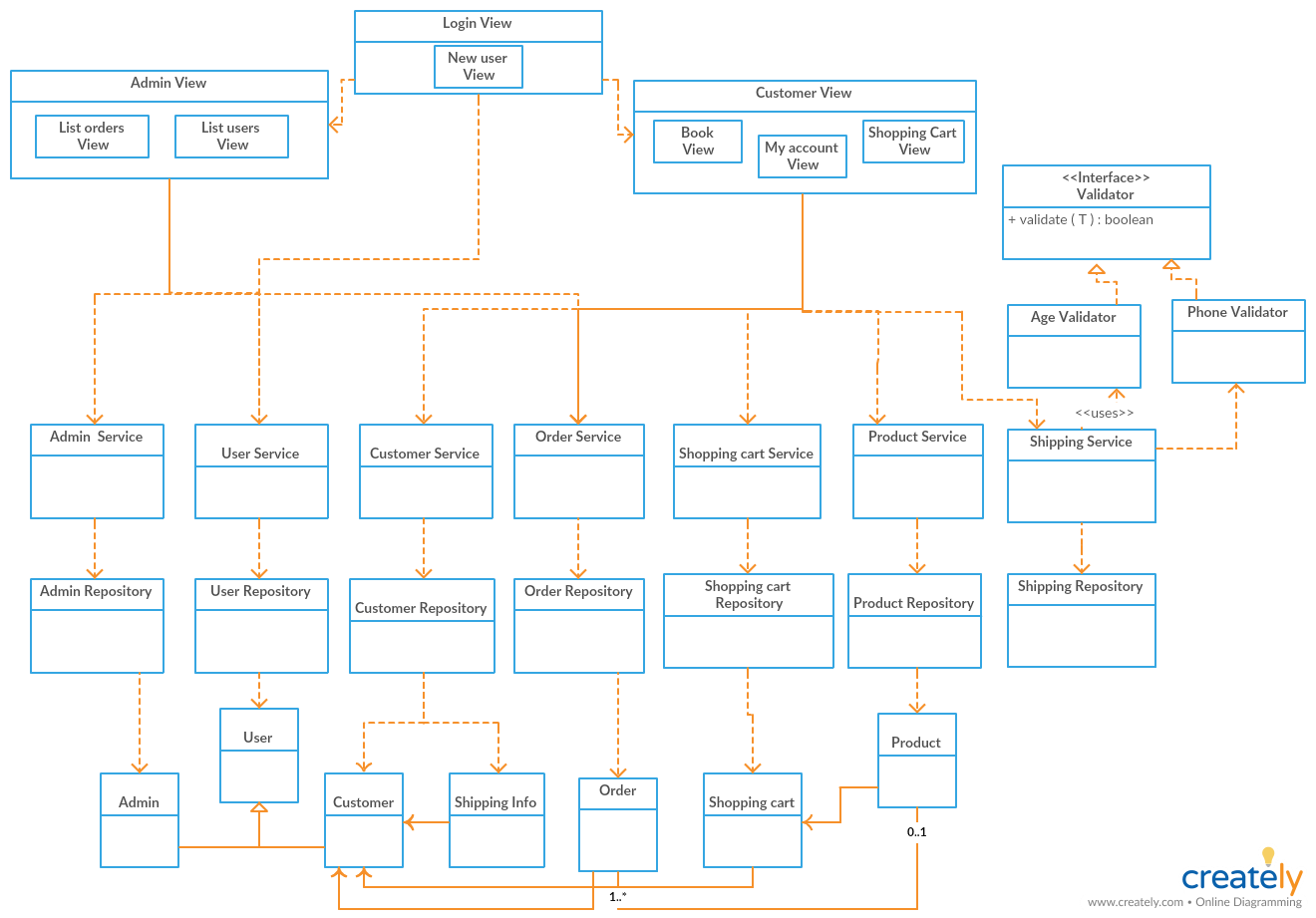
## Dynamic Behavior

Sequence diagram: order the items placed in the shopping cart

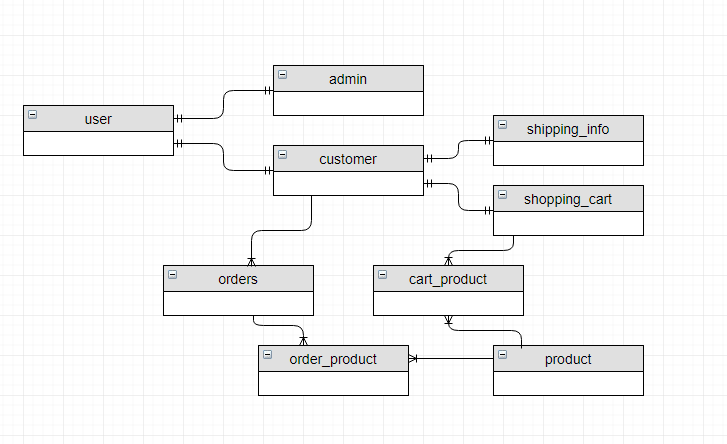


Communication diagram:

## Class Design



# Data Model



# Unit Testing

To test the application I will write some small tests and I will use Mockito. Mockito is a popular mock framework which can be used in conjunction with JUnit. Mockito allows us to create and configure mock objects, by simplifying the development of tests for classes with external dependencies significantly.

# Construction and Transition

# System Testing

As mentioned above, unit tests were written in order to test the application and to increase the certainty that the system is working correctly.

# Future improvements

The system can be improved in the future by adding more functionalities both to the admin and to the customer side (for example, choosing the number of the same books added to the cart). To add flexibility, the application should be converted into a web application and security should be implemented.

# Bibliography

<https://www.tutorialspoint.com/hibernate/hibernate_annotations.html>

<https://spring.io/guides/gs/spring-boot/>

<https://www.tutorialspoint.com/uml/uml_deployment_diagram.htm>

http://www.vogella.com/tutorials/Mockito/article.html