Design Document

## 

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author(s) | Changes | Version |
| 03/05/2021 | Özgün Şen  Berfu Anıl | initial document | v1.0 |
| 31/05/2021 | Özgün Şen  Berfu Anıl | Review document | v1.1 |

# Design structure

The design of the software should follow the decisions defined in the Architecture notebook document. The system’s structure and the model classes are shown below. This view is the *class diagram* view of Visual Studio 2019.

# 

# Patterns

## Controller Pattern

#### **Overview**

A controller handles all incoming requests from the user interface and controls the data flow.

#### **Structure**

For example, in our project, the HomeController is the connection between the UserData model and the Index and Login views. It listens for events prompted by the Actor from views and responds appropriately. Login process occurs by getting data from view to controller then sending it properly by the model to the Data Access layer.

## Abstract Factory Pattern

**Overview**

Abstract Factory Pattern says that just define an interface or abstract class for creating families of related

objects but without specifying their concrete sub-classes.

**Structure**

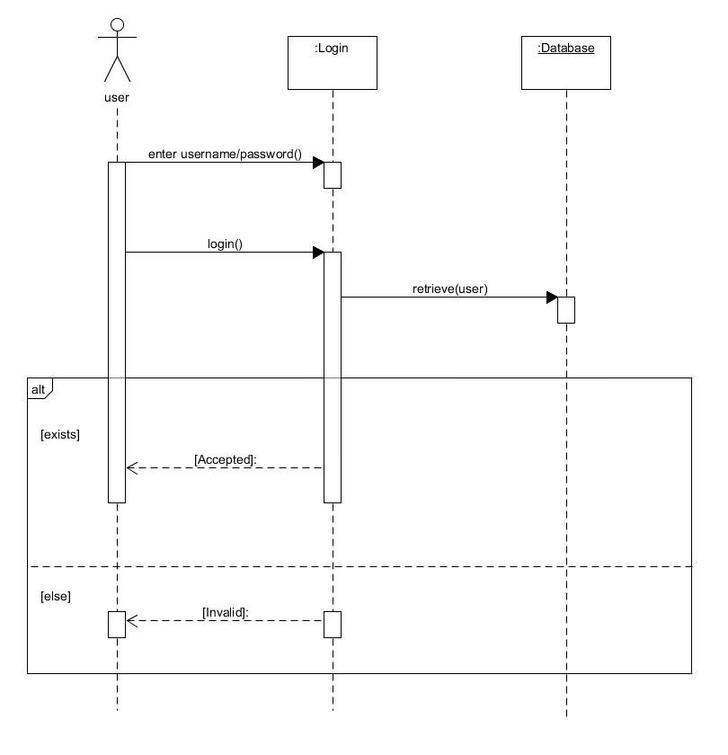
In our project, we create ISqlDataAccess and IUserData as abstract interfaces. SqlDataAccess and UserData classes are the concrete classes.

# Requirement realizations

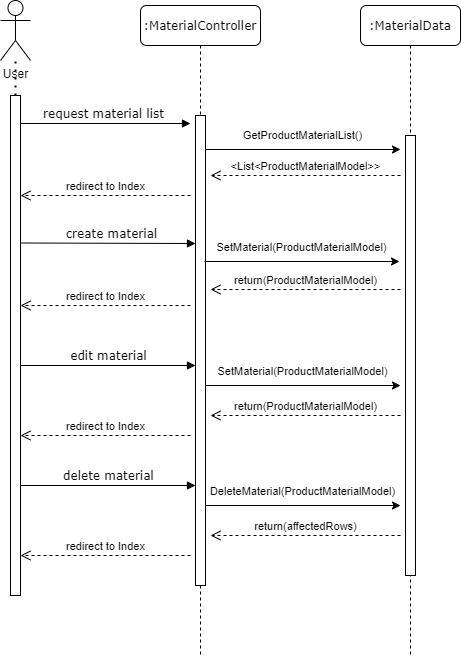
Verification of all input data will be made with the data annotations (property attributes written on the head

of each property of each class) written on the domain classes.

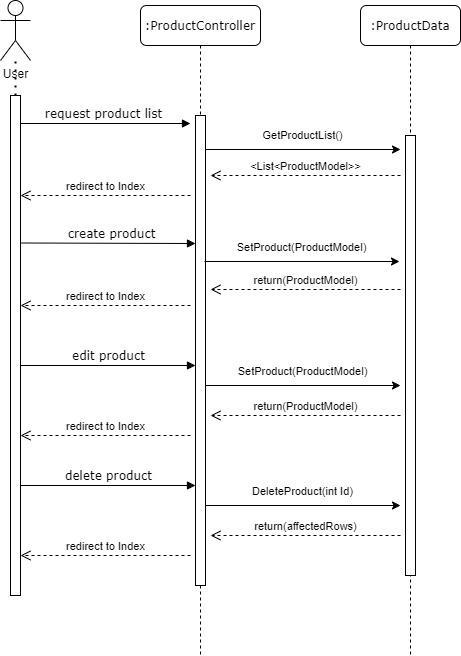
## **UC1.2:** Use Case Realizaiton Diagram



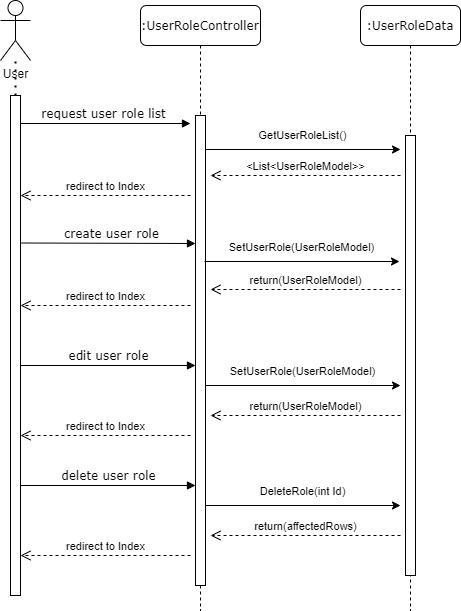
## UC2.1: Use Case Realizaiton Diagram



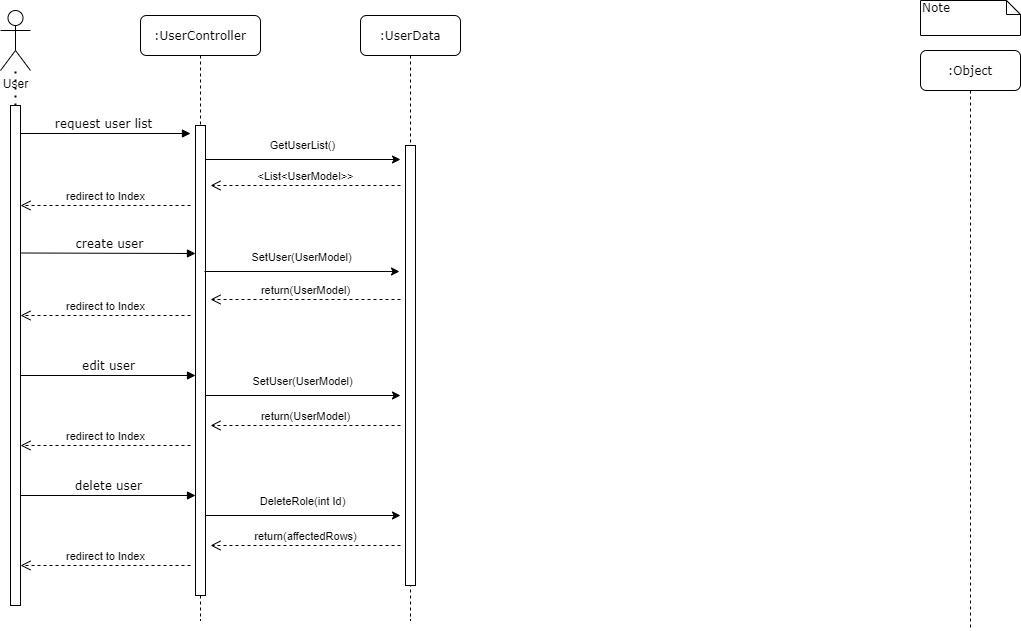
## UC2.2: Use Case Realizaiton Diagram



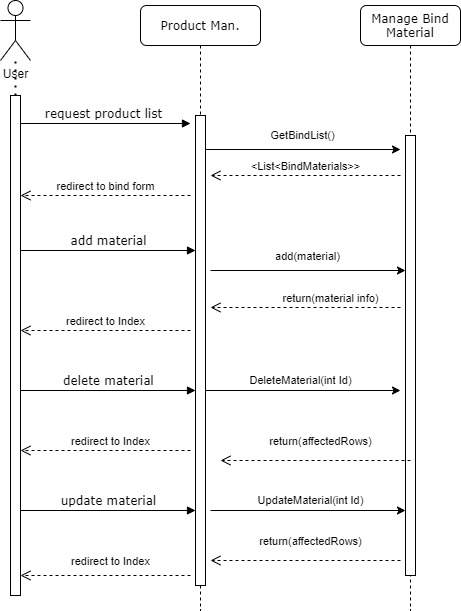
## UC1.3: Use Case Realizaiton Diagram



## UC1.4: Use Case Realizaiton Diagram



## UC2.3: Use Case Realizaiton Diagram



## UC3.1: Use Case Realizaiton Diagram

