

Information System Development

Semetral Project

Information System of Selling Glasses

Instructor: RNDr. Ing. Martin Radvanský, Ph.D.

Student: Tuan Phong HUYNH

Student's ID: HUY0018

Aritifact 1:

We are designing an information system to manage the entire lifecycle of selling branded glasses online. The system will cater to **four main categories of users**.

The customer, who wants to purchase branded glasses, interacts with the system via our website. The customer can create an account, browse available glasses by brand, style, lens type, and other filters like size or color. Customers can view product details, read reviews, and check availability. They can also place orders, make payments.

The production team(staff) manages the stock of branded glasses from various suppliers. They use a desktop interface to update product availability, and place reorders when necessary. The system tracks which brands and models are selling well, allowing the team to adjust inventory and stock popular items

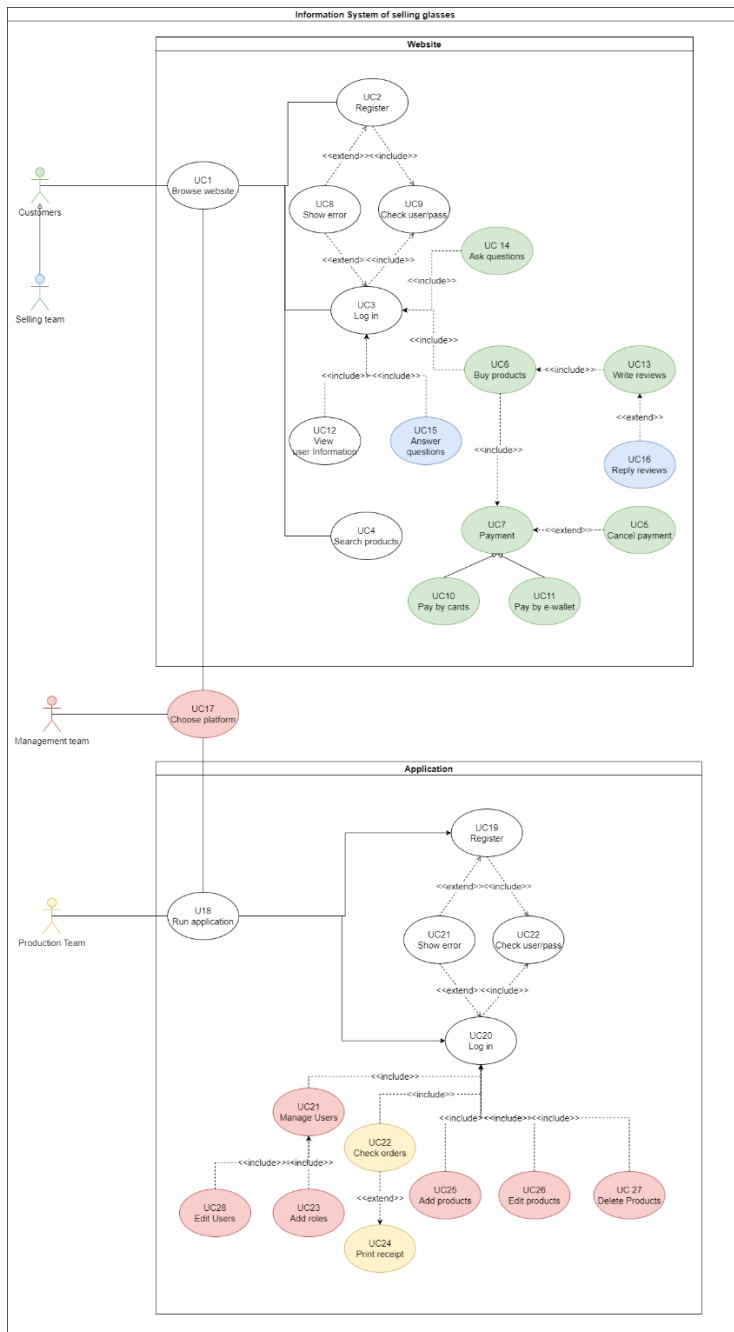
The selling team(staff) uses their account on website interface to interact with review of customers. Answering questions of customers on the website interface.

The management team has the highest authority oversees overall operations of **selling team, production team** and **customer** satisfaction through the website interface. They check the sales trend, stock levels and supplier performance through the desktop system.

The information system for selling glasses helps improve efficiency in bussiness. It monitores all aspects from our users. For customers, the IS provides a shopping experience with basic features. For staff, the IS provides a basic system that allows them to easily manage customers' orders and inventory in real time. For management team, with full access to the information system, they can easily manage their stores and businesses.

The information system will operate 24 hours a day if there is internet.

Aritifact 2:

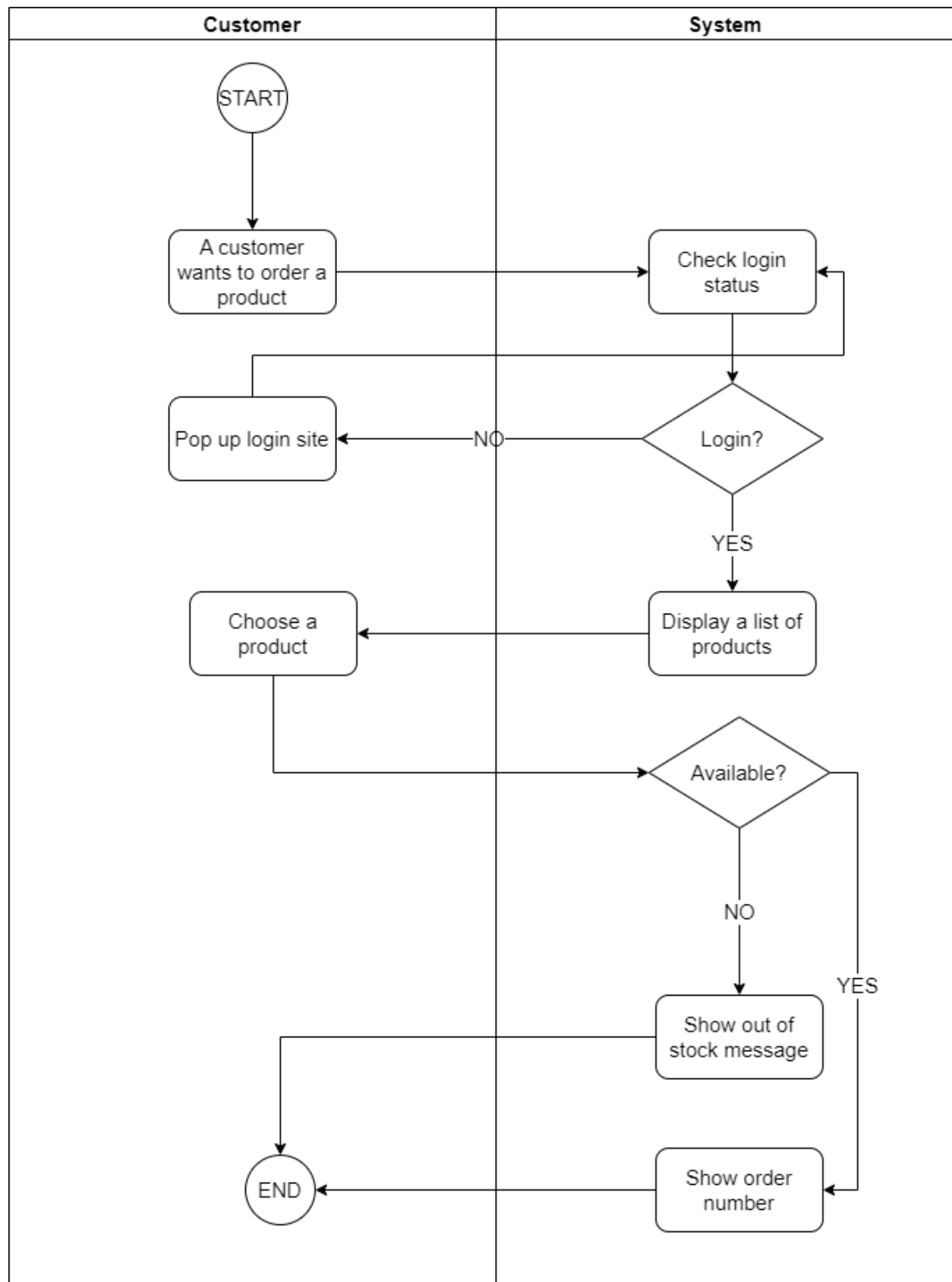


Note:

- Green use cases are for Customers only
- Blue use cases are for Selling team only
- Yellow use cases are for Production team only
- Red use cases are for Management team only

Use case scenario and Activity diagram (**UC6**)

Use case:	UC6 – Buy Products
Goal:	Allow customers to order a product.
Actor:	Customers
Another actors:	none
Pre-conditions:	<ul style="list-style-type: none">• Customers have to login into the system (UC3 - log in)• Customers choose a product in a list and order.
Post-conditions:	<ul style="list-style-type: none">• Customer order will be saved in the system and display in both UI.
Main flow:	<ol style="list-style-type: none">1. The customers have to login into the main page.2. The system displays the list of products.3. The customers select a product in the list.4. The system confirms the order and display the order number
Alternative flow:	<ul style="list-style-type: none">• AF1: If the customer is not logged in:<ol style="list-style-type: none">1. The system requires the customer to log in to order a product.2. The customer performs UC3 – log in.3. After successfully logging in, the customer returns to step 1 of the Main Flow.• AF2: If the product is out of stock:<ol style="list-style-type: none">1. The system notifies that to customers and display the menu to ask customers want to continue shopping or not.2. After successfully selecting countinue shopping, the system return to the step 2 in the main flow.



Use case scenario and Activity diagram (**UC28**)

Use case: **UC28** – Edit Users

Goal: Allow managers to edit roles and information of a user.

Actor: Management Team

Another actors: None

Pre-conditions:

- Actors have to login into the system (**UC3** - log in)
- Actors have to a user of management team

Post-conditions:

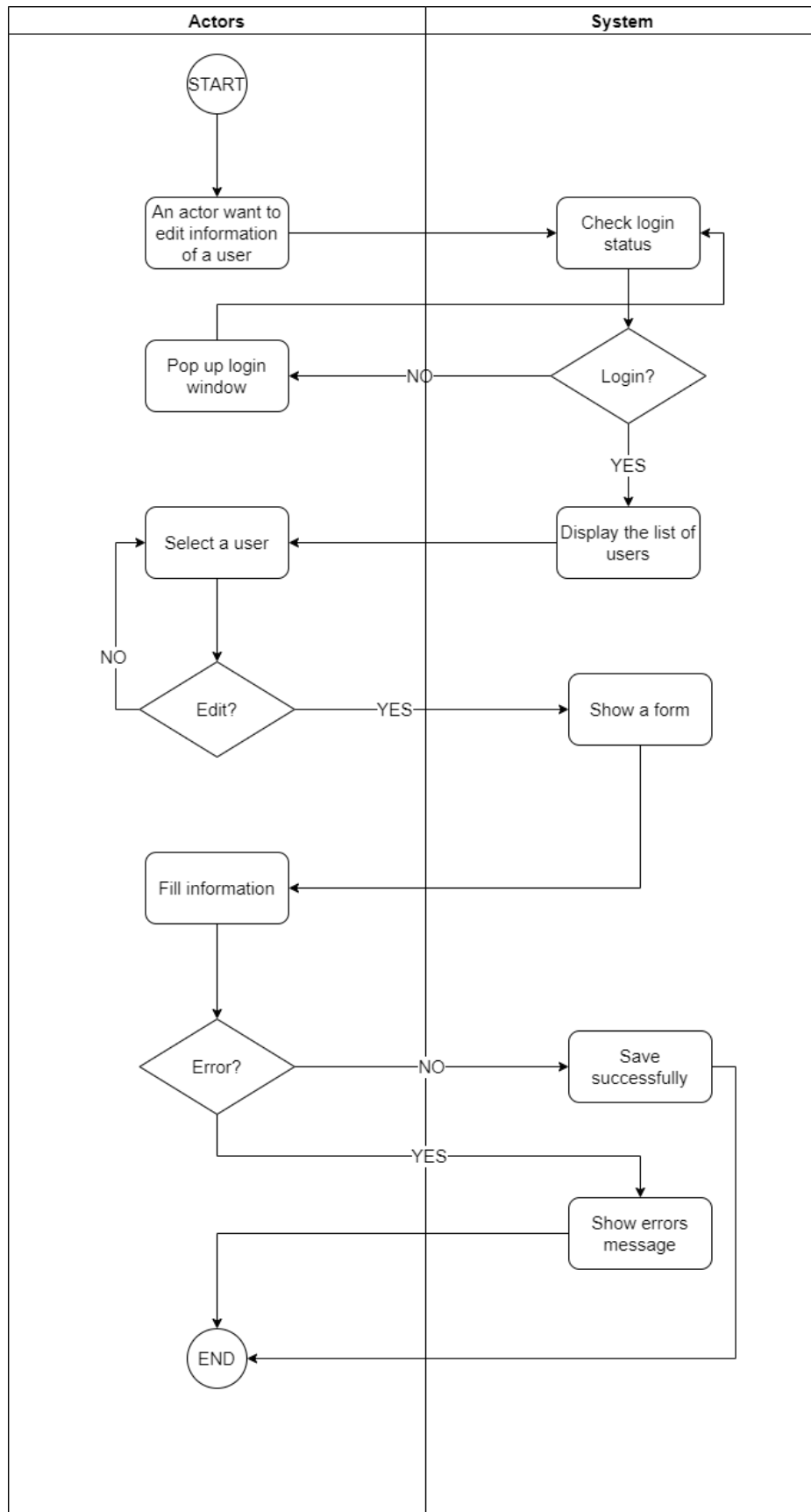
- Edit information for a user successfully

Main flow:

1. Actors accesses the manage page in the application(**UC21** – Manage Users).
2. Actors select a user row in the list and press the button edit.
3. The system displays a form of full detail.
4. Actors fill into the form and press the save button.
5. The system confirms and change the user information into the database.

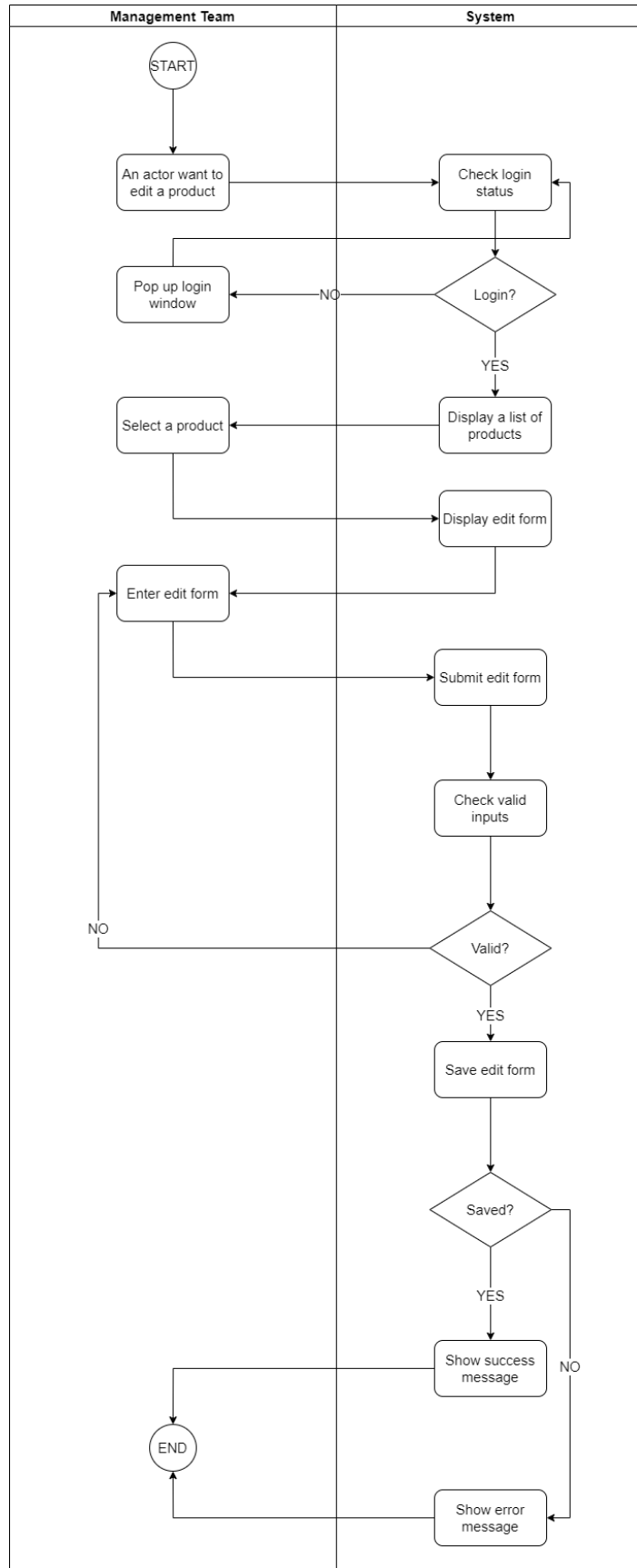
Alternative flow:

- **AF1:** If the customer is not logged in:
 1. The system requires the actor to login first.
 2. The actor performs **UC3** – log in
 3. After successfully logging in, the actor returns to step 1 of the Main Flow.
- **AF2:** If the actor does not fill all the form:
 1. The system will pop-up the message to notify the actor that need to fill all the form.
 2. The actor has to fill the form and return to the step 4 in the main flow.



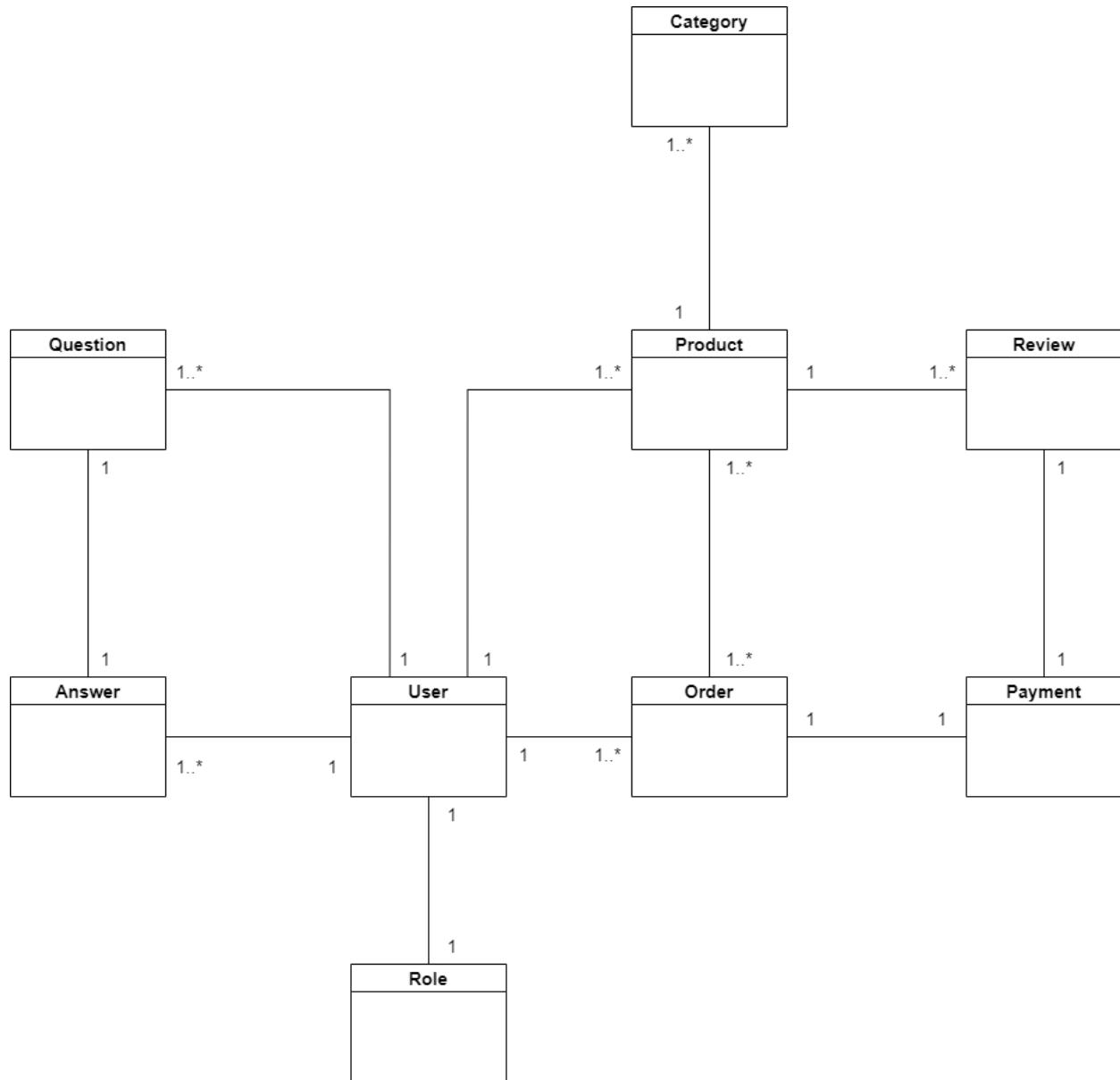
Use case scenario and Activity diagram (**UC26**)

Use case:	UC26 – Edit Product
Goal:	Allows actors to edit existing product information on the system.
Actor:	Management Team
Another actors:	None
Pre-conditions:	<ul style="list-style-type: none">• Actors have to login into the system (UC3 - log in).• The products have to exist in the system.
Post-conditions:	<ul style="list-style-type: none">• Product information is successfully updated in the system.• Product changes are saved and updated immediately on the web page for customers.
Main flow:	<ol style="list-style-type: none">1. Management Team accesses the product list in the system.2. Management Team selects the product to edit.3. The system displays the product editing form, including fields such as name, description, price, image, and other information.4. Management Team makes the necessary changes and clicks "Save" to save.5. The system updates the product information and displays a confirmation message that the changes have been saved successfully.
Alternative flow:	<ul style="list-style-type: none">• AF1: If there is an error in the input data:<ol style="list-style-type: none">1. The system displays an error message and asks the Management Team to check and re-enter the correct data.2. The Management Team edits the data and clicks "Save" again to save the changes.• AF2: If the Management Team cancels the edit:<ol style="list-style-type: none">1. The Management Team can select the "Cancel" option instead of "Save" to cancel the changes.2. The system does not save any changes and returns to the product list page.



Artifact 3:

First domain model



Technological decisions – estimations

Class	Storage	Estimated Entity size in kB	Estimated number of Entities after one year	Estimated Entities size after one year in kB
User	MS-SQL	2	1200	2400
Role	XML	0.5	4	2
Order	MS-SQL	1	10000	10000
Payment	MS-SQL	1	10000	10000
Review	MS-SQL	1	5000	5000
Product	MS-SQL	3	500	1500
Category	MS-SQL	1	10	10
Question	MS-SQL	1	1000	1000
Answer	MS-SQL	1	1000	1000

Estimated database size after one year is 30.13 MB.

Estimating growing factor is 1.2 year.

MS-SQL was chosen for reasons:

- Easy to install
- Scalable with high performance
- Safe with high security
- Good integration

Role was chosen for XML file because this class has basic structures, do not have too many attributes, and do not require complex queries.

Types of user interactions with the system

Users access to the information system based on roles. Customers and Selling Team mainly use the system on the website. In the opposite, Management Team and Production Team use the system via the desktop application. All roles have their own account to access the system.

Webiste interface is provided for Customers, Selling Team and Management team, with a user-friendly interface, customers will easily access the website in the fastest and easiest way to use, as well as for the selling team and managment team, it is very easy to use as well as to train new users.

Appllication interface is provided for Management Team and Production Team, the desktop application will try to be as simple as possible so that the management team can manage employees as well as products, while the application is also easy to use for the production team manage orders as well.

For the design information system, a three-layer architecture with two different UIs (web and desktop) and two different data stores are used.

For the repository Microsoft SQL Server and XML Files were chosen because:

- High performance and stability
- Well-structured data storage
- Easy to maintain
- Scalable in the future

The information system will be deployed on .NET 4.8 because:

- Support cross-platform
- Improved Performance
- Support various libraries

Choice of platform

1. .NET framework:

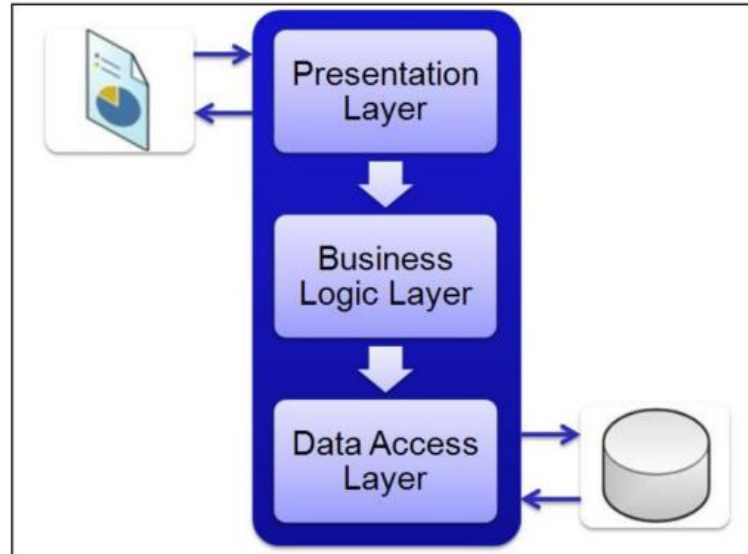
- Microsoft Visual Studio Community 2022 Version 17.9.5
- Microsoft .NET Framework Version 4.8

2. System requirements:

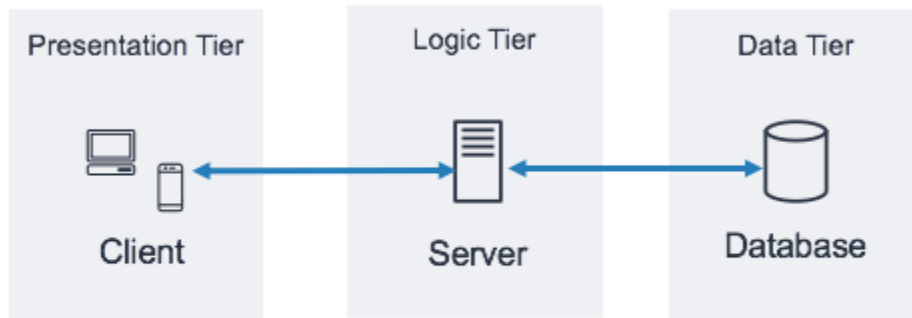
- Server:
 - Processor:
 - Minimum: 4-core CPU
 - Recommended: 8-core or higher
 - Memory (RAM):
 - Minimum: 8GB
 - Recommended: 16GB or higher
 - Storage:
 - Minimum: 256GB
 - Recommended: SSD storage and 512GB or higher
 - Database: Microsoft SQL server for remote and XML files
 - Operating system: Windows 10 or higher or Linux
- Client:
 - Processor: 4-core CPU, 1.6GHz or higher
 - Memory (RAM): 4GB or higher
 - Storage: HDD or SSD, at least 100GB free for storage
 - Browser: Google Chrome, Microsoft Edge, Mozilla Firefox, or Safari
 - Operating system: Windows 10 or later

System layout

For information system architecture, this project will use 3 layer architectures which are User interface layer, Business logic layer and Data access layer.



For information system layout, this project will use 3 tier architecture which are presentation tier, business tier and data tier.

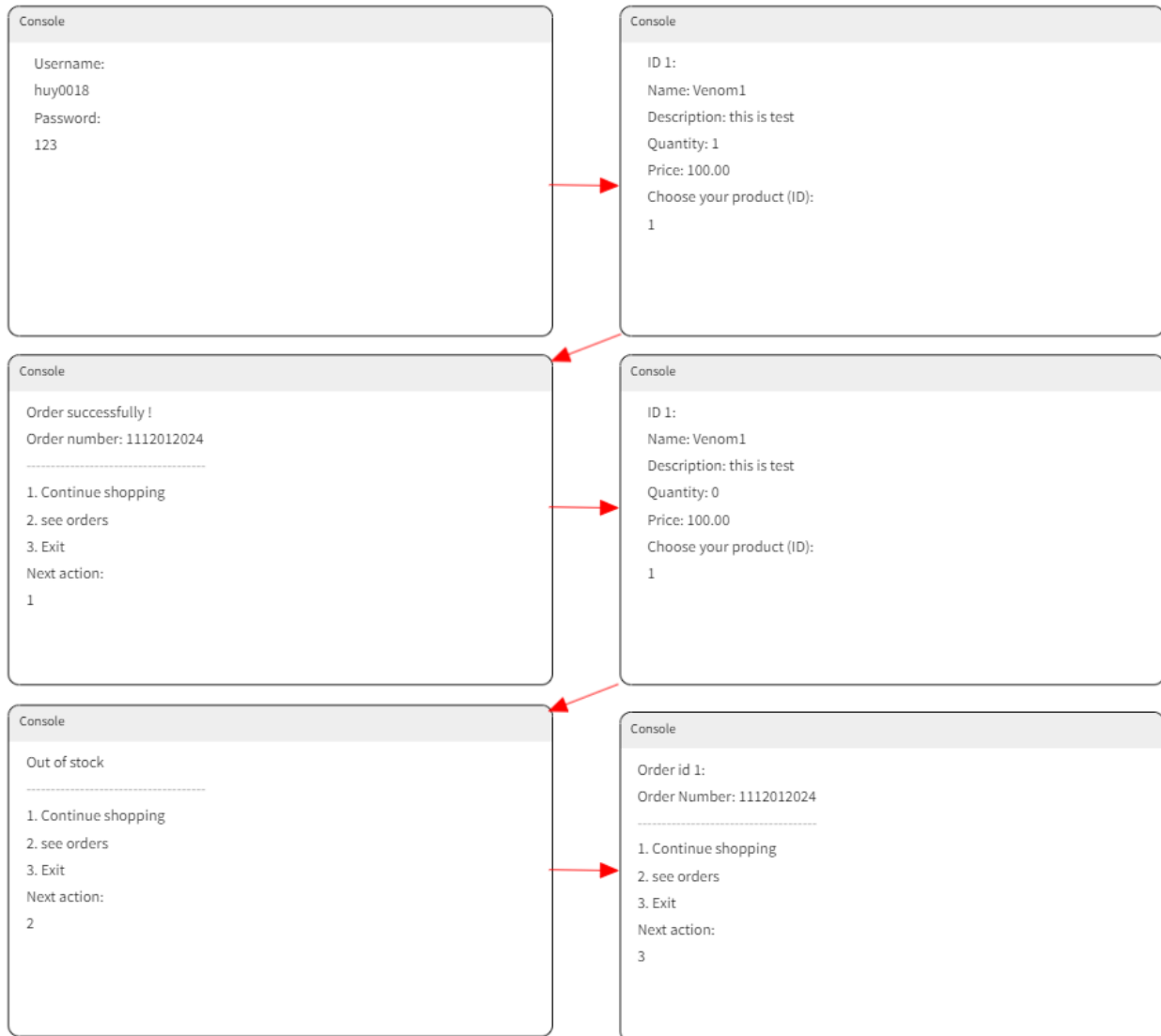


Patterns and architecture

- Design patterns use:
 - Table data gateway pattern: Used in data access layer to create, update and delete data easily.
 - Transaction Script pattern: to hides the complexities of the system and provides an interface to the client using which the client can access the system
 - Singleton pattern: provides a static method to get its static instance to outside world, to create an object while making sure that only single object gets created.
- Architectural patterns:
 - Three-layered (three-tier) architecture:
 - Presentation layer: for website interface and desktop application interface which is UIs. This layer is for users interact with information system.
 - Business logic layer: All logics is working in this layer, it contains functions such as to create new users, create new orders, helps users do the logics.
 - Data Access Layer: Used to store data such as users, orders, products, ... in microsoft SQL Server and XML files.

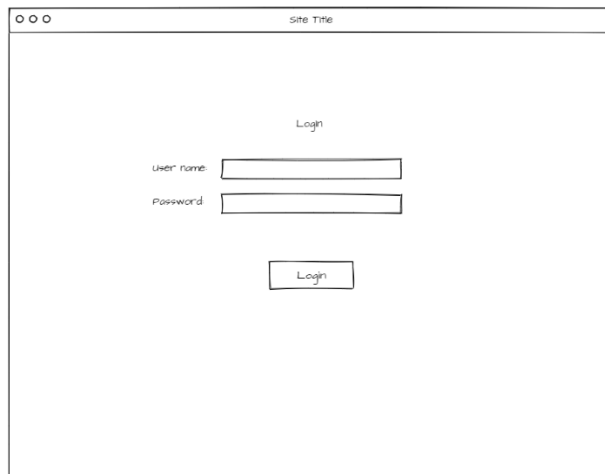
Artifact 4:

Use case scenario (UC6) – Console UI:



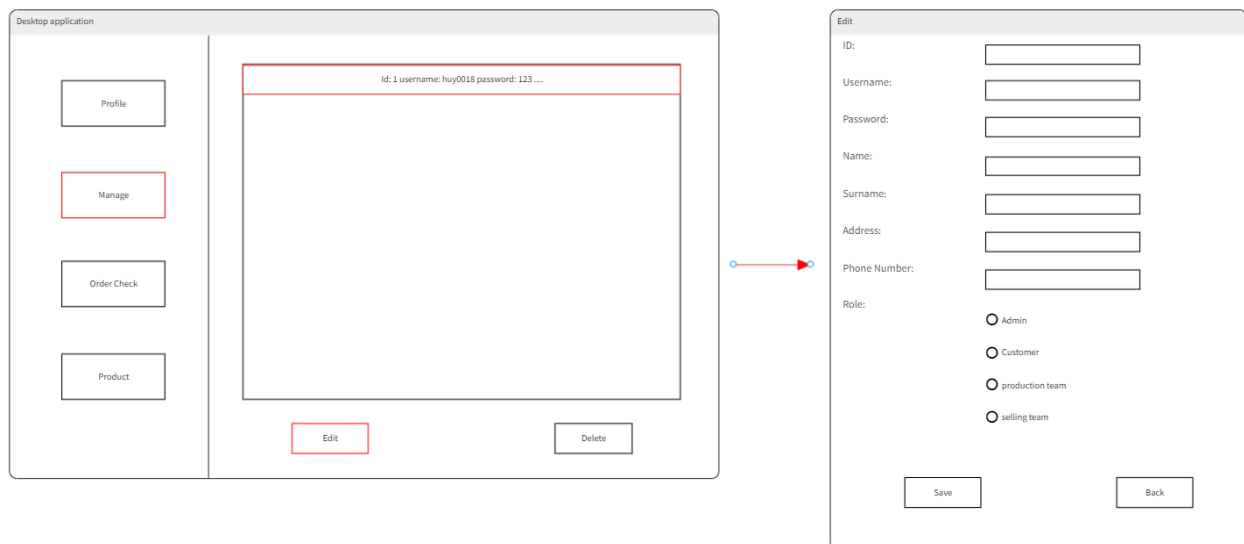
Use case scenario (UC28) – desktop application UI:

Users need to login first:



A login form window titled "site Time". It contains a "Login" label at the top center. Below it are two input fields: "User name:" and "Password:". At the bottom center is a "Login" button.

After selecting a user:



The "Desktop application" window shows a sidebar with "Profile", "Manage", "Order Check", and "Product". The "Manage" button is highlighted with a red border. The main area displays a table with one row: "Id: 1 username: huy0018 password: 123 ...". Below the table are "Edit" and "Delete" buttons. A red arrow points from the "Edit" button to the "Edit" form on the right.

The "Edit" form contains the following fields:

- ID:
- Username:
- Password:
- Name:
- Surname:
- Address:
- Phone Number:
- Role:
 - ☐ Admin
 - ☐ Customer
 - ☐ production team
 - ☐ selling team

At the bottom are "Save" and "Back" buttons.

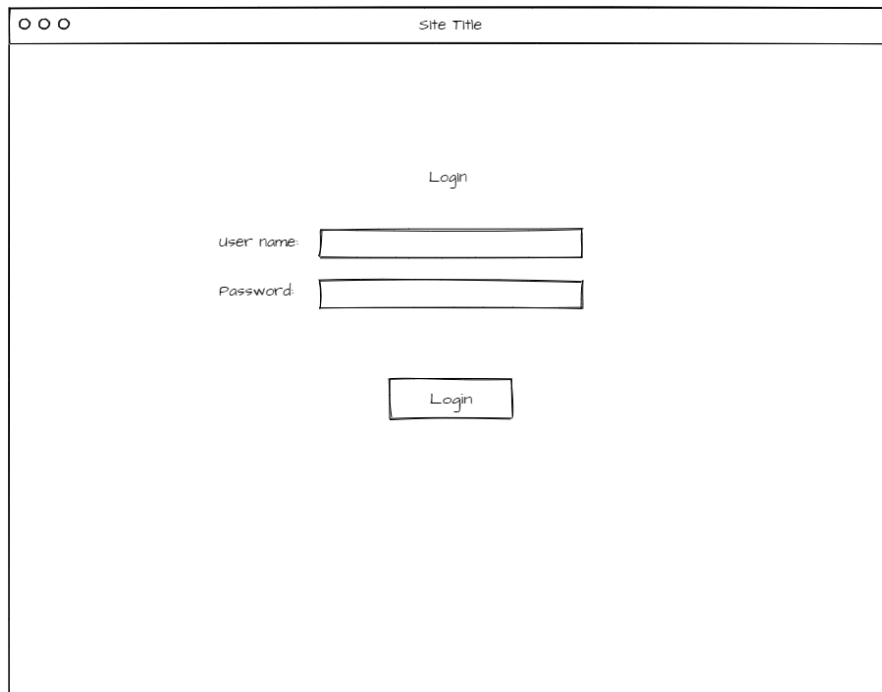
If actors do not fill all details:



An "Error" dialog box with the message "Must fill all detail" and a "Back" button.

Use case scenario (UC26) – desktop application UI:

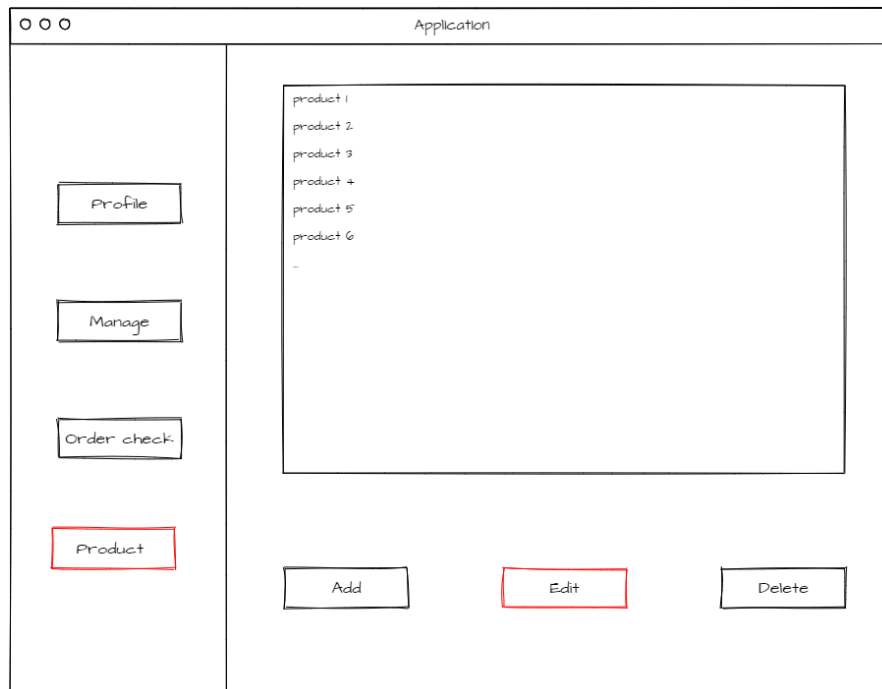
Users need to login first:



A hand-drawn sketch of a login form within a window titled "Site Title". The window has three small circles in the top-left corner. The form is centered and contains the following elements:

- The word "Login" centered at the top of the form area.
- A label "User name:" followed by a rectangular input field.
- A label "Password:" followed by a rectangular input field.
- A rectangular button labeled "Login" centered below the input fields.

Selecting product section:



A hand-drawn sketch of a product management interface within a window titled "Application". The window has three small circles in the top-left corner. The interface is divided into a left sidebar and a main content area.

Left Sidebar:

- Four buttons stacked vertically: "Profile", "Manage", "Order check", and "Product". The "Product" button is highlighted with a red border.

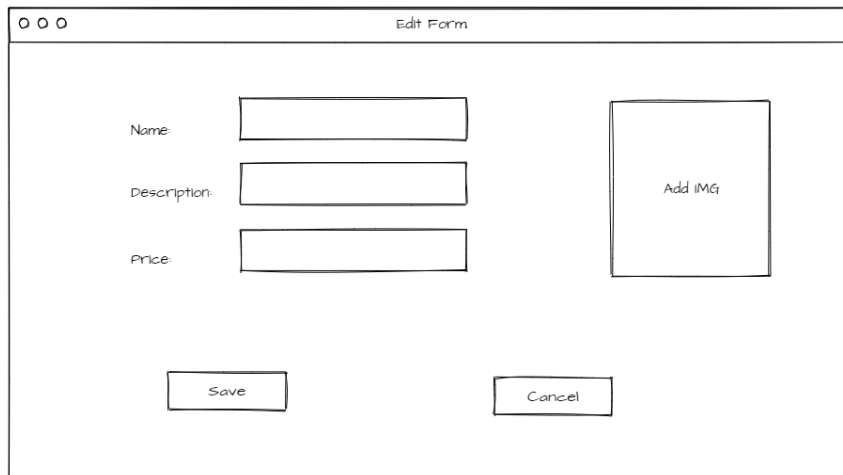
Main Content Area:

- A large rectangular box containing a list of products:
 - product 1
 - product 2
 - product 3
 - product 4
 - product 5
 - product 6
 -

Bottom Action Bar:

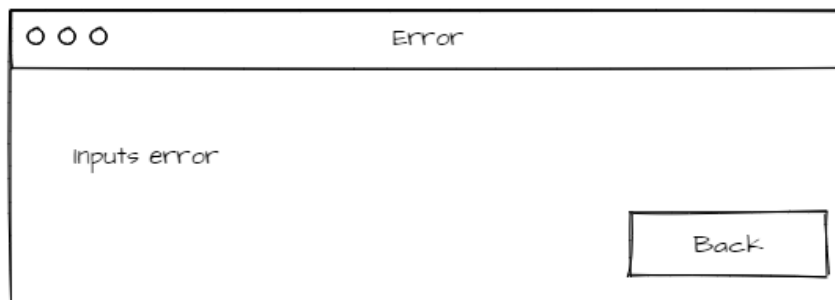
- Three buttons: "Add", "Edit", and "Delete". The "Edit" button is highlighted with a red border.

Edit form:



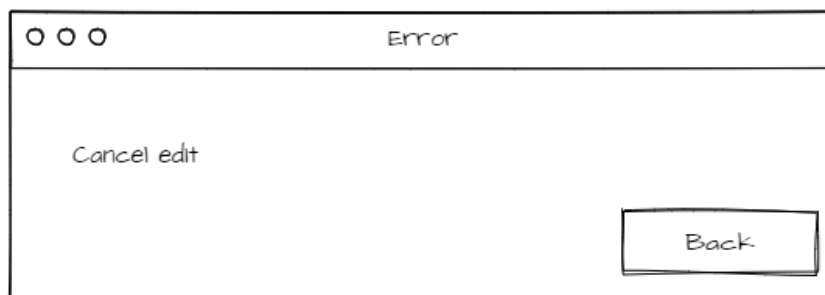
A hand-drawn sketch of an 'Edit Form' window. The title bar contains three small circles on the left and the text 'Edit Form' on the right. The main area contains three input fields on the left, each with a label to its left: 'Name', 'Description', and 'Price'. To the right of these fields is a large square button labeled 'Add IMG'. At the bottom of the window are two buttons: 'Save' on the left and 'Cancel' on the right.

If the inputs are error:



A hand-drawn sketch of an 'Error' window. The title bar contains three small circles on the left and the text 'Error' on the right. The main area contains the text 'Inputs error' on the left. On the right side, there is a button labeled 'Back'.

If users cancel edit product:



A hand-drawn sketch of an 'Error' window. The title bar contains three small circles on the left and the text 'Error' on the right. The main area contains the text 'Cancel edit' on the left. On the right side, there is a button labeled 'Back'.

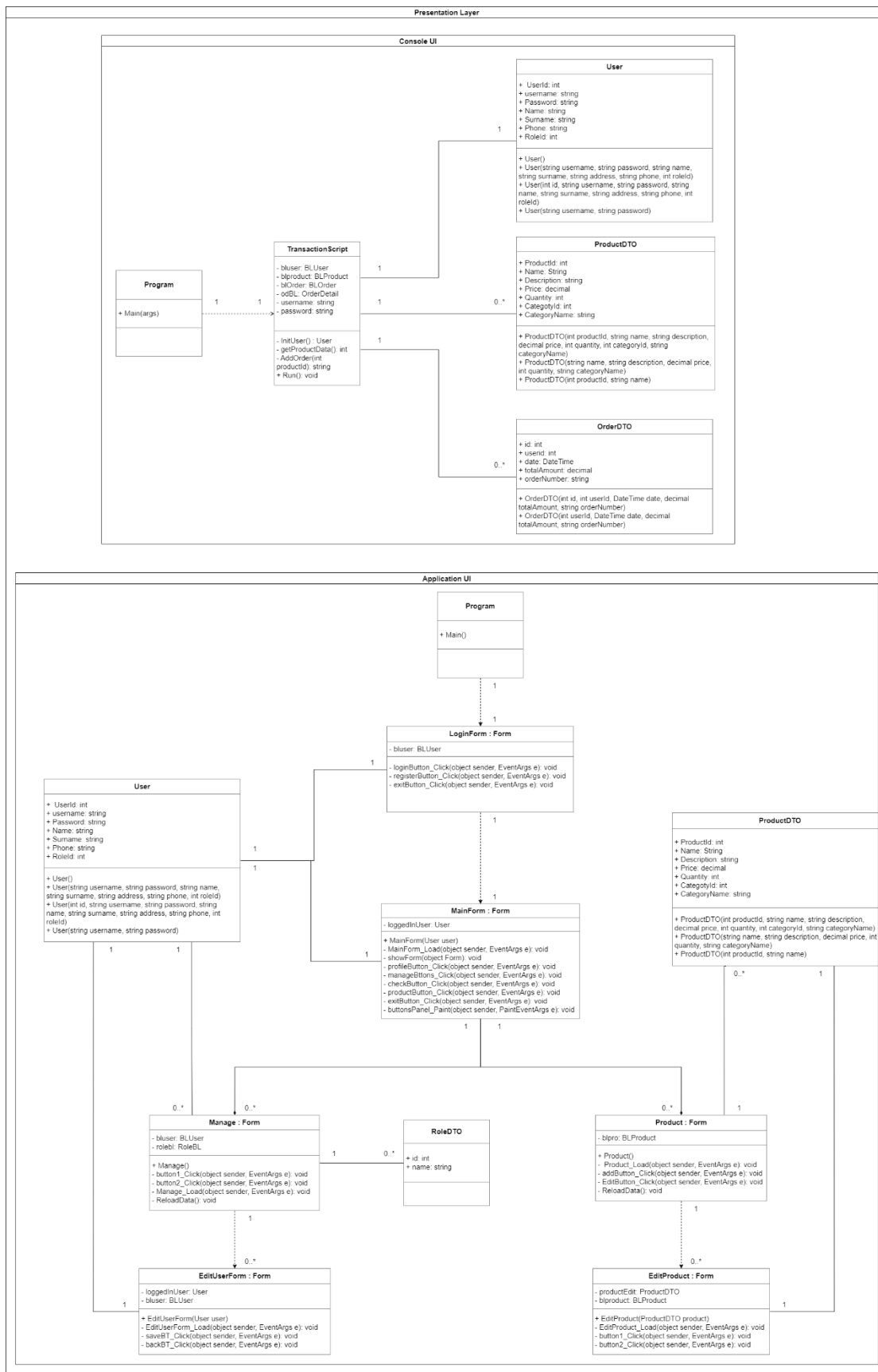
Artifact 5:

Static Class Diagram :

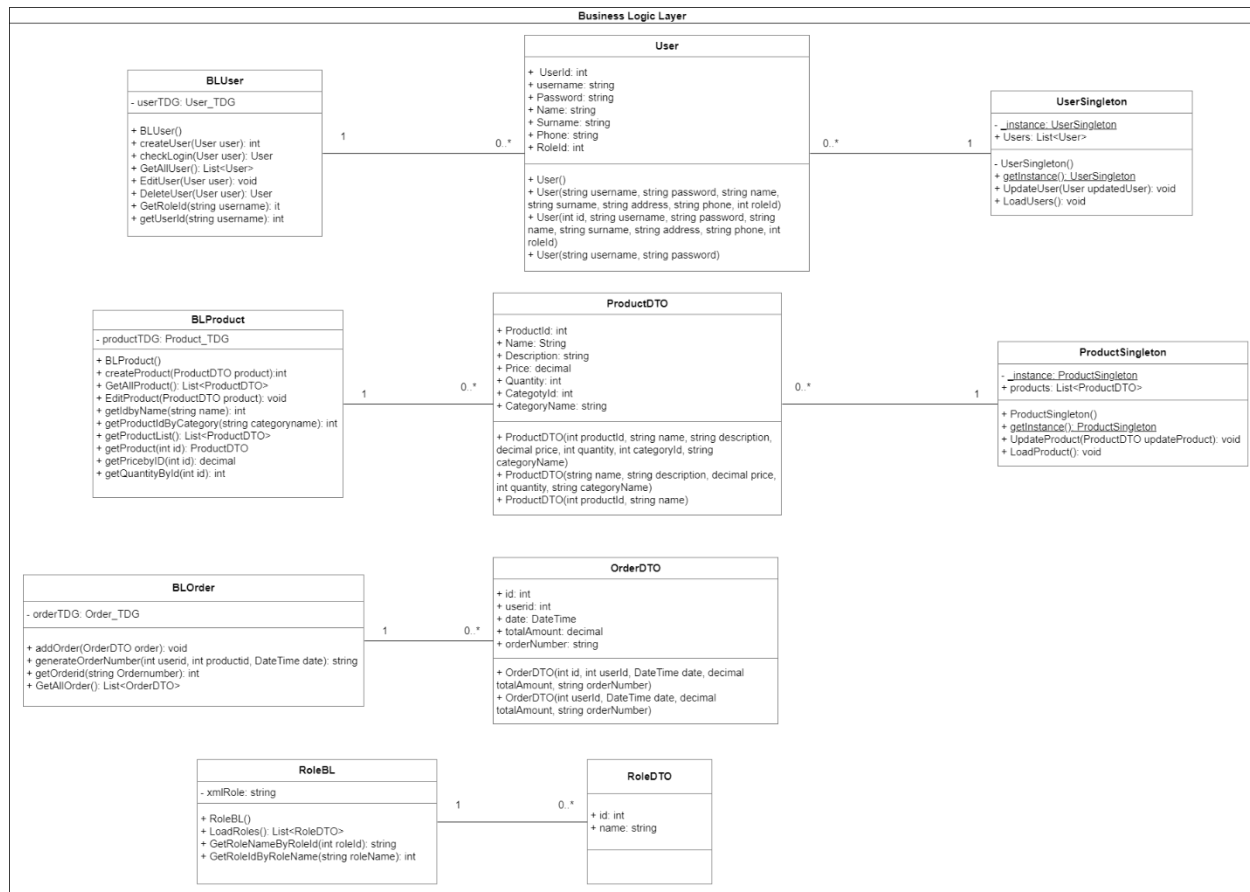
Presentation Layer:

In presentation layer, I use 3 patterns which are:

- Transactionscript: to reduce the compexity in program classes
- DTO: to transfer data between layers in the system
- Singleton: to create an instance (list of an DTO) and use it globaly.



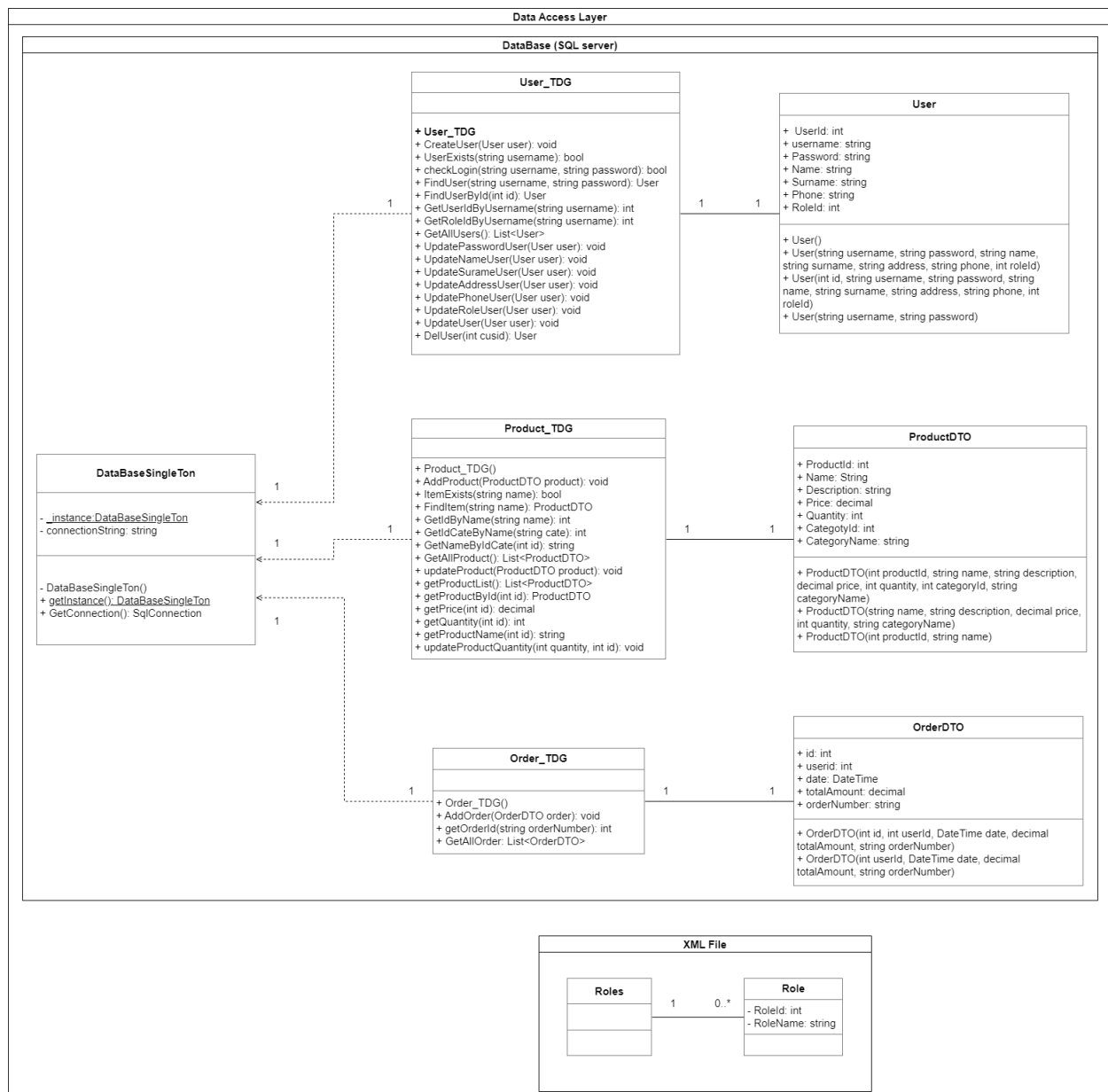
Business Logic Layer:



In Business Logic Layer, 3 patterns were used which are:

- **DTO**: to transfer data between layers in the system
- **Table Data Gateway**: to use queries in the database related to a DTO table
- **Singleton**: to create an instance (list of an DTO) and use it globally.

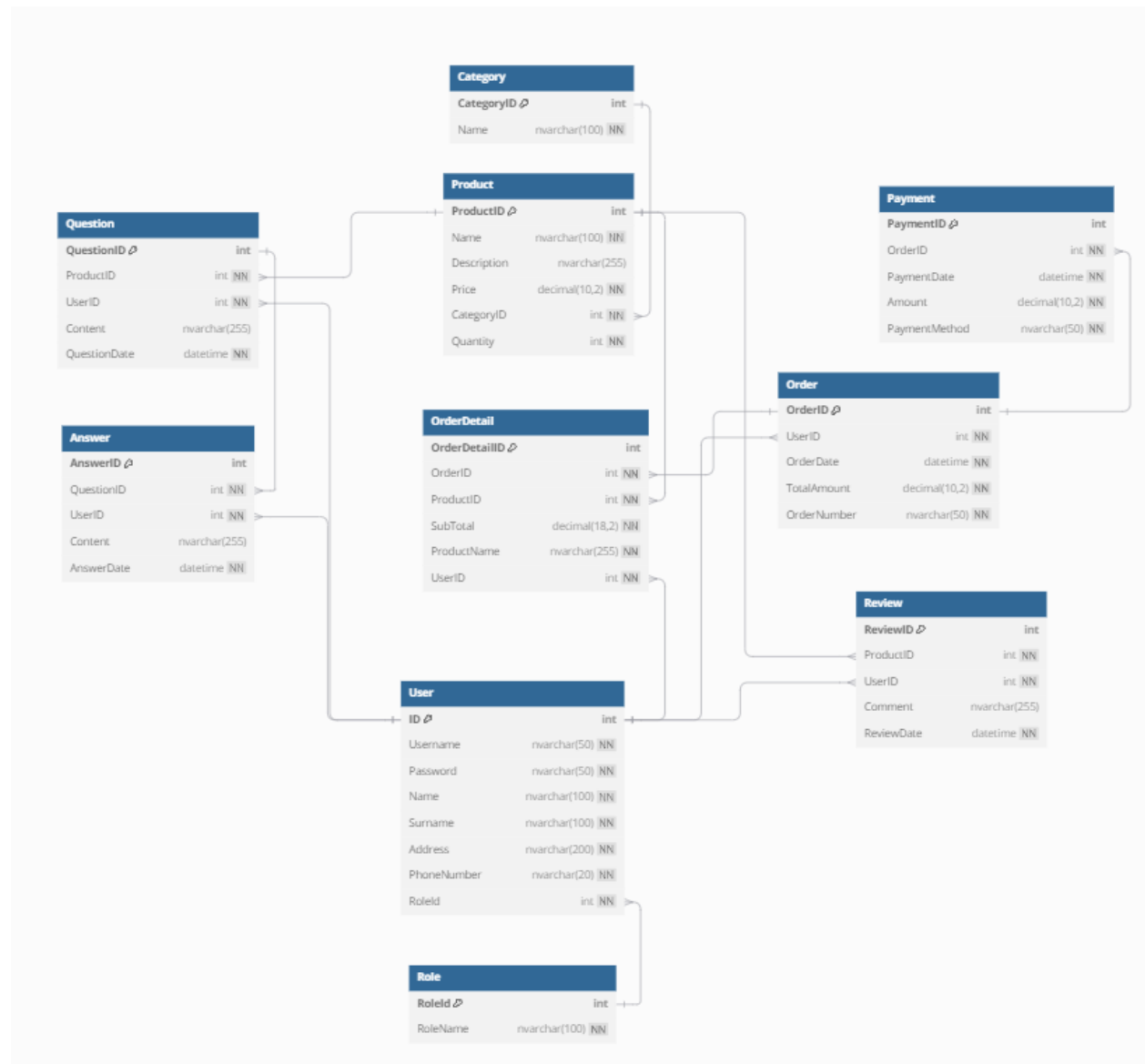
Data Access Layer:



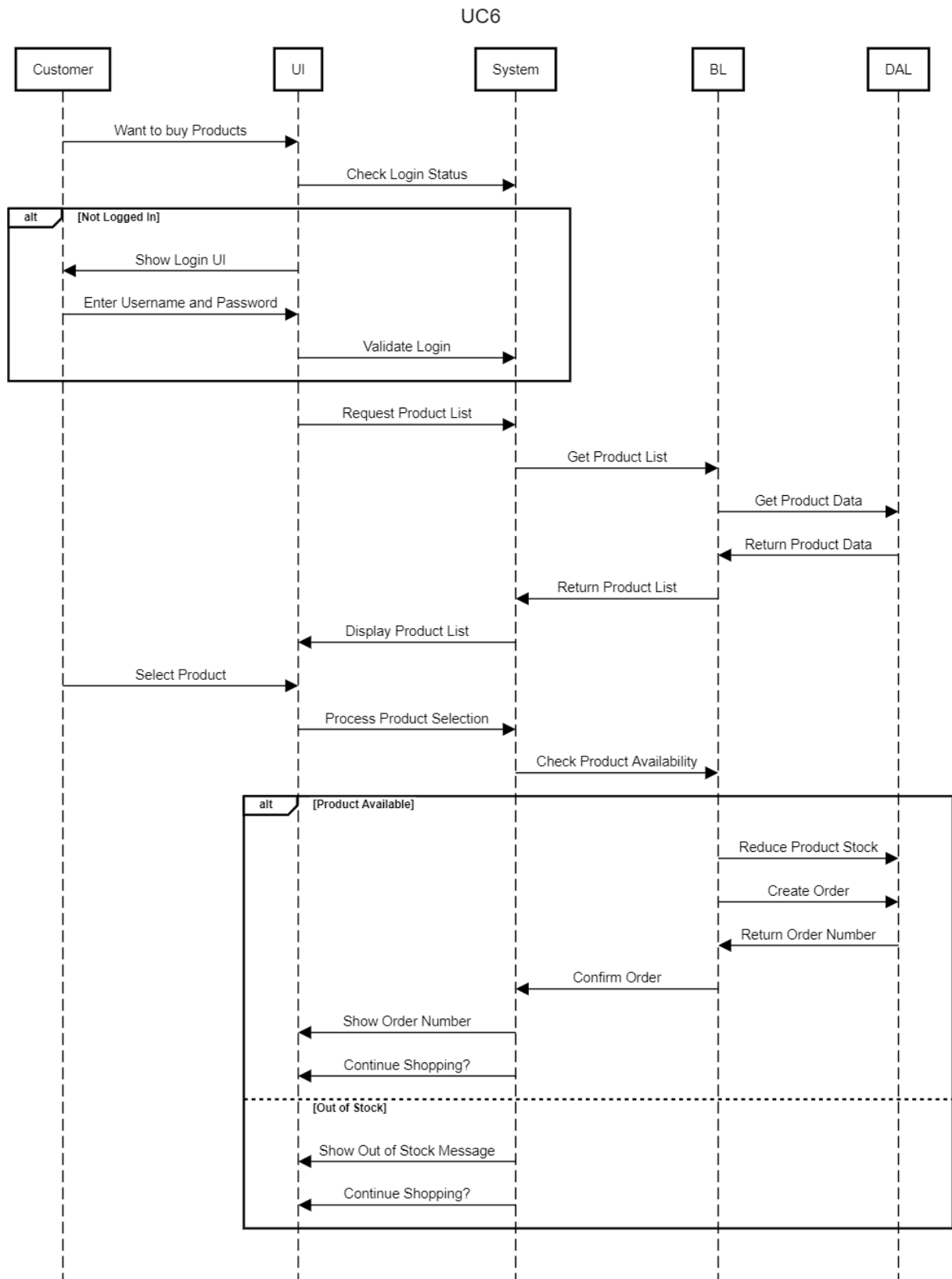
In Data Access Layer, I use 3 patterns which are:

- Singleton: to create an instance (connection) and use it globally.
- Table Data Gateway: to use queries in the database related to a DTO table
- DTO: to transfer data between layers in the system

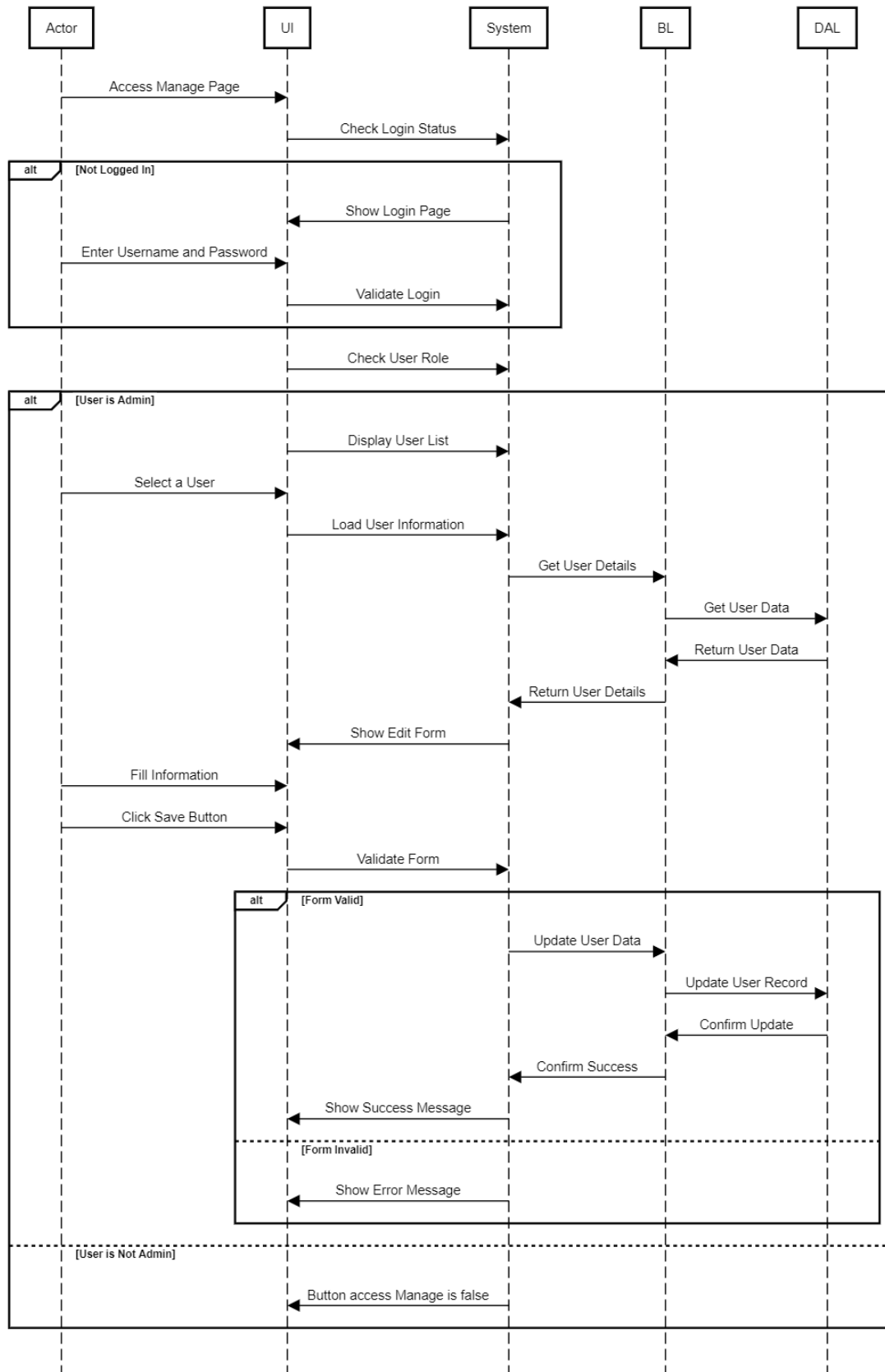
Data model:



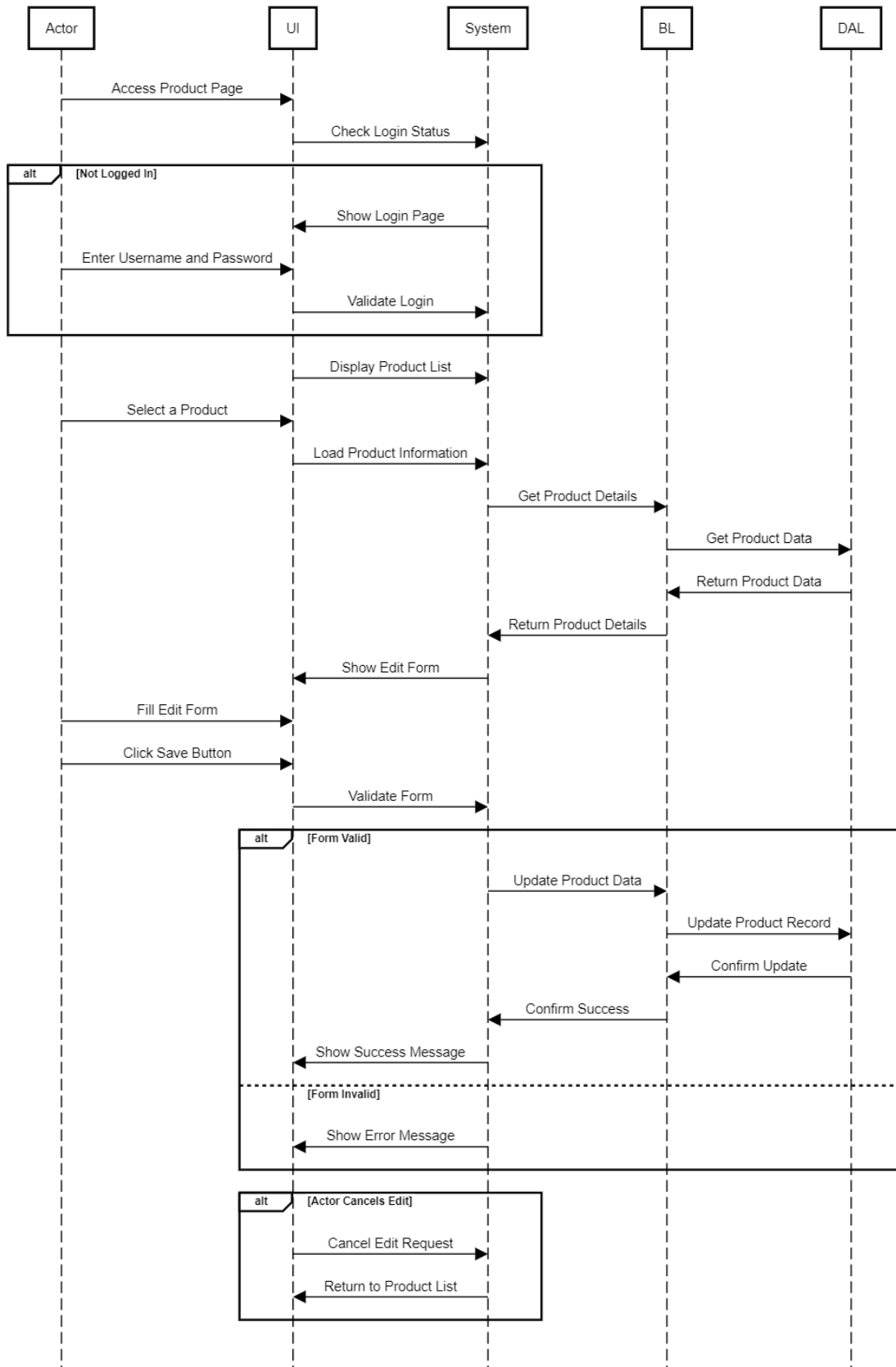
Sequence diagrams:



UC28

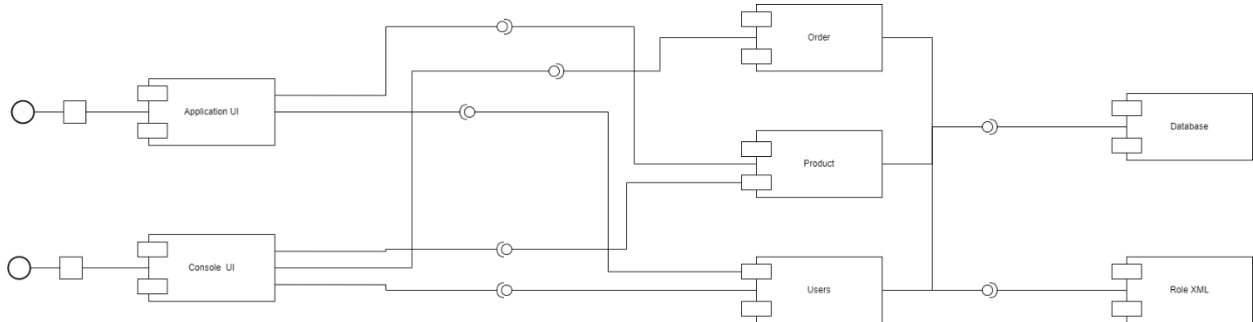


UC26



Artifact 6:

Component diagram:



Deployment diagram:

