

**Store Management System**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**SUBMITTED TO**

MA’AM FATIMA GILLANI

**SUBMITTED BY**

IFRA RAZZAQ (SP-21005)

HIRA FATIMA (SP-21007)

**COURSE**

SOFTWARE CONSTRUCTION AND DEVELOPMENT

# Contents

[1. Contents 2](#_Toc137288751)

[1. Introduction: 3](#_Toc137288752)

[2. What problem does it solve? 3](#_Toc137288753)

[3. Functional Requirements: 4](#_Toc137288754)

[4. Benefits: 5](#_Toc137288755)

[5. Diagrams: 6](#_Toc137288756)

[Use Case: 6](#_Toc137288757)

[Sequence Diagram: 7](#_Toc137288758)

[Package Diagram: 8](#_Toc137288759)

[Object Diagram: 8](#_Toc137288760)

[Petrinet Diagram (firing, matrices): 9](#_Toc137288761)

[Communication diagram: 14](#_Toc137288762)

# Introduction:

The Store Management System aims to assist store owners or managers in managing their store's inventory and sales. It allows users to perform various operations such as adding and removing items from the inventory, processing sales, updating item details, and generating reports.

The code is structured using object-oriented programming principles. It defines two classes: Item and StoreManagementSystem.

The Item class represents a single item in the store's inventory. It has three member variables: name (string), price (integer), and quantity (integer). The class provides a constructor to initialize these variables and a display method to print the item's details.

The StoreManagementSystem class is the main class that controls the overall functionality of the system. It has private member variables: inventory (a vector of pointers to Item objects) and totalSales (integer) to keep track of the inventory and total sales amount, respectively. The class defines several private methods to handle specific tasks like login validation, adding and removing items from the inventory, processing sales, and updating item details. It also provides public methods like run to start the system and interact with the user.

In the run method, the system prompts the user to enter their login credentials. If the login is successful, the main menu is displayed, and the user can choose different options by entering the corresponding numbers. The system then performs the selected operation and repeats the process until the user chooses to exit.

The Store Management System aims to simplify the process of managing a store's inventory and sales by providing a user-friendly interface and essential functionalities. It serves as a starting point that can be further extended and customized based on specific business requirements.

# What problem does it solve?

Store management system solves the problem of managing inventory and sales in a store by providing various features such as: For example, adding items to inventory, removing items from inventory, processing sales, viewing inventory, viewing total sales, searching for an item, and updating item information (price and quantity) and calculating the total inventory value.

The system allows users to perform these operations through a command line interface, providing a menu-based approach to interacting with the system.

As a whole, Store Management System helps store owners or managers to keep track of their inventory, manage sales and get important information about the items in stock.

# Functional Requirements:

The store management system has the following functional requirements:

**Login:** The system should prompt users to enter a username and password to access the system. The current implementation requires admin credentials for both the username and password. However, for a real application, you would need to implement proper login validation logic.

**Add Item to Inventory:** Users should be able to add a new item to inventory. You should provide the item name, price and quantity.

**Remove item from inventory:** Users should be able to remove an item from inventory by providing the item name.

**View Stock Level:** Users should be able to view the current stock level, which shows the details of each item, including its name, price, and quantity.

**Process a sale:** Users should be able to process a sale by entering the item name and quantity sold. The system should update the quantity of the item and calculate the total sale amount.

**View total sales:** Users should be able to view total sales.

**Search for an item:** Users should be able to search for an item in inventory by typing its name. The system should display the details of the item if it is in stock.

**Update an item:** Users should be able to update the price or quantity of an item in inventory. You should select an option and provide the new value.

**Calculate Total Inventory Value:** Users should be able to calculate the total value of inventory. The system should multiply each item's price by its quantity and sum the values.

**Exit:** Users should be able to exit the program.

These are the main functional requirements of the store management system based on the provided code. However, note that this is a basic implementation and there may be additional requirements depending on the specific needs of the system or application.

# Benefits:

The Store Management System offers several advantages and motivations for implementation in a retail or inventory management environment:

**Efficient Inventory Management:** The system allows for easy inventory tracking and management. Users can add new items, remove items, update prices and quantities, and view the current inventory status. This helps to provide accurate and up-to-date information about available products.

**Optimized sales processing:** The system simplifies the process of sales control. Users can enter the name and quantity of the items sold, and the system will automatically update the stock quantity and calculate the total sale amount. This streamlines the sales process and reduces manual effort.

**Improved decision-making:** With access to real-time inventory information and total sales data, users can make informed decisions regarding restocking, pricing strategies, and overall store performance. The system provides insights into inventory value and overall sales, allowing users to optimize their business operations.

**Improved Customer Service:** A comprehensive inventory management system allows retailers to provide better customer service. You can quickly check item availability, provide accurate product information, and process sales efficiently. This helps improve customer satisfaction and loyalty.

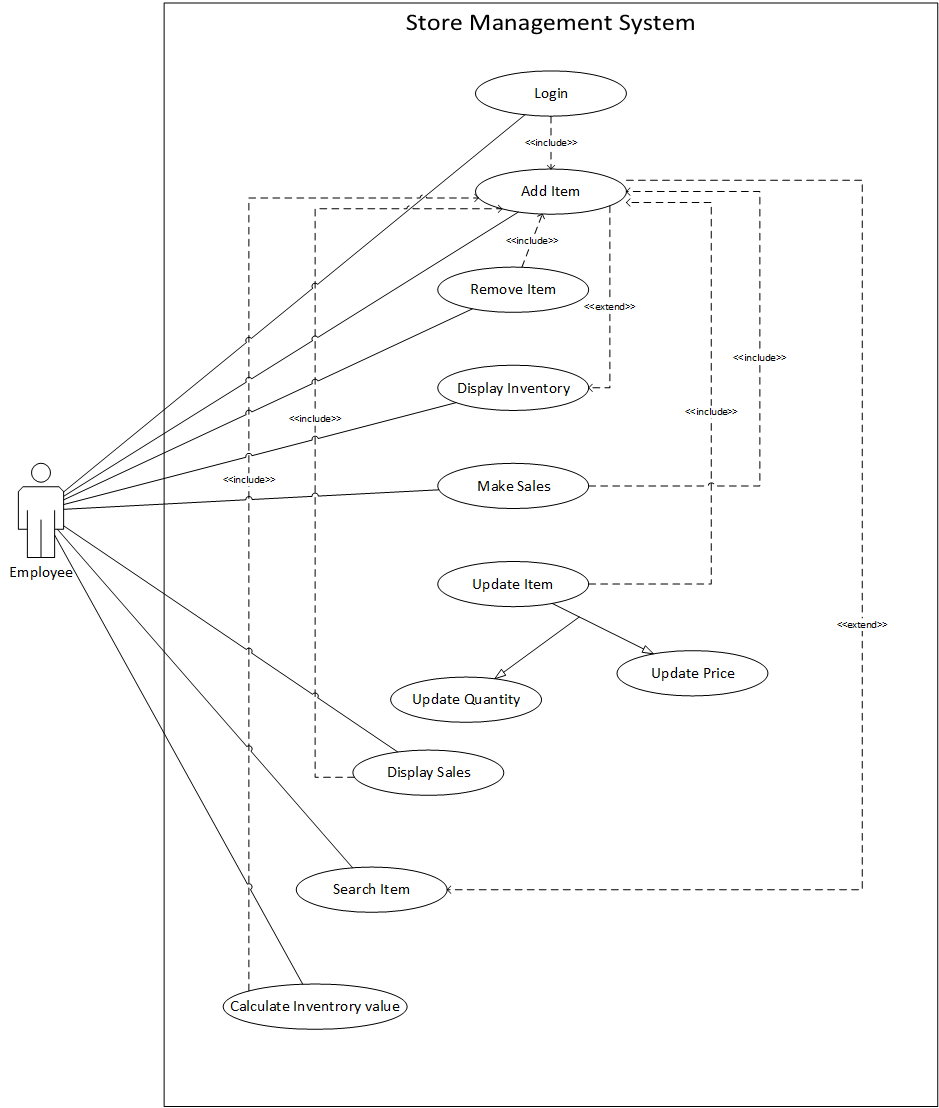
**Reduced Errors and Losses**: The system minimizes the chances of errors and losses due to manual inventory management. By automating processes such as quantity updates and sales calculations, the system reduces the risk of data entry errors and provides accurate information.

**Time and Cost Savings:** The store management system saves time by automating repetitive tasks such as inventory tracking, sales processing, and inventory value calculations. It eliminates the need for manual record-keeping and reduces the time spent on inventory management tasks. This leads to cost savings by improving operational efficiency.

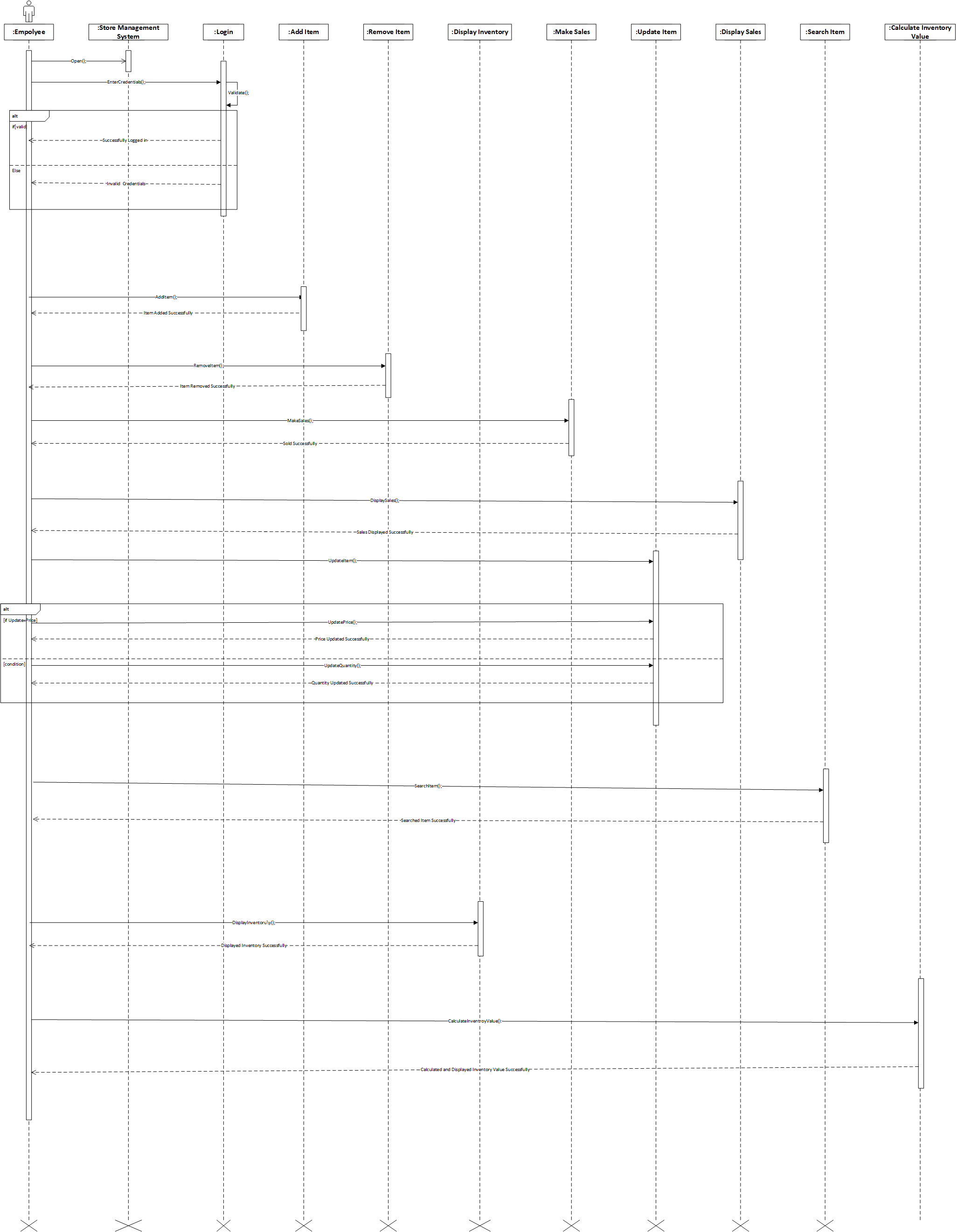
**Scalability and growth:** The system can handle a growing volume of storage and sales. As the business grows, the system can process more items and sales transactions without sacrificing performance. It offers a scalable solution to handle increasing inventory levels and sales demands.

# Diagrams:

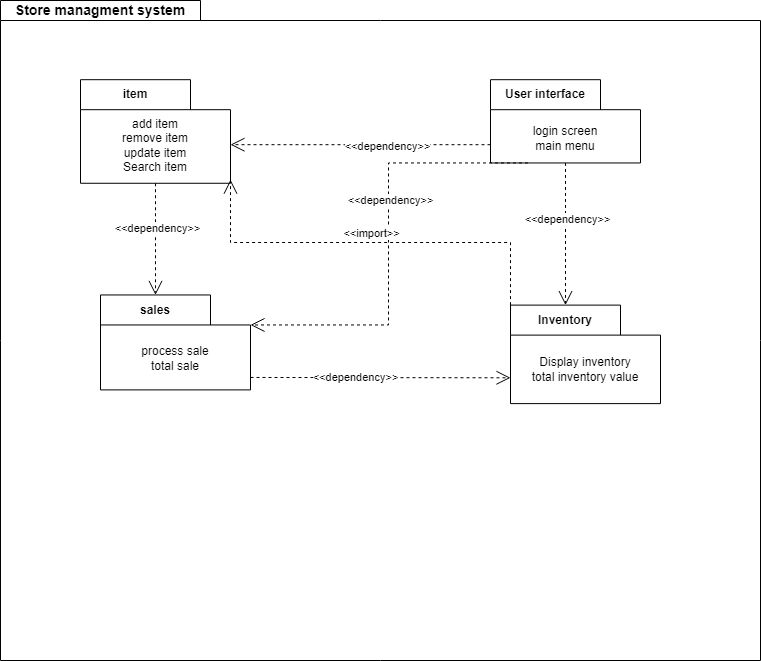
## Use Case:



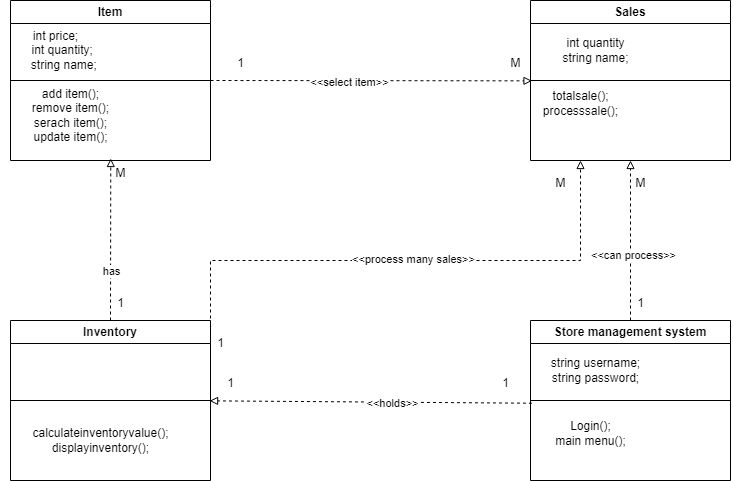
## Sequence Diagram:



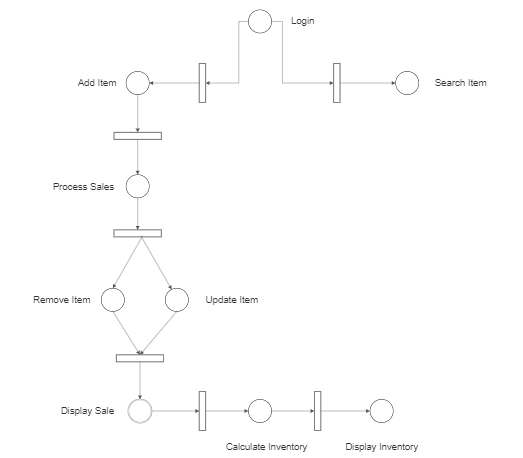
## Package Diagram:



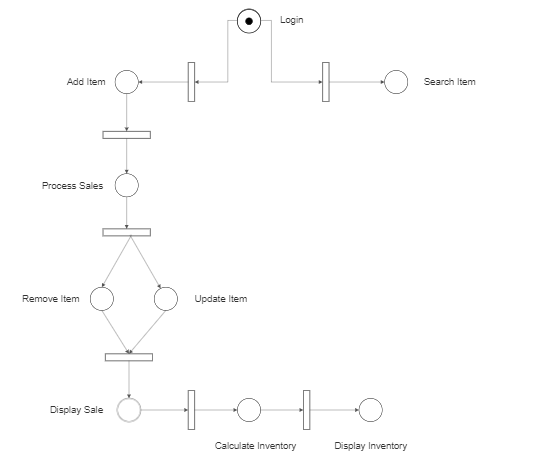
## Object Diagram:

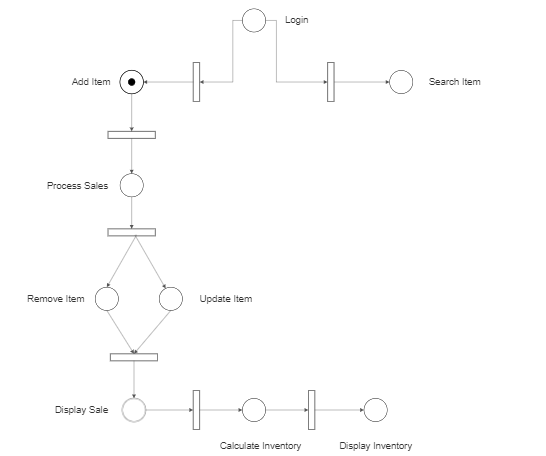


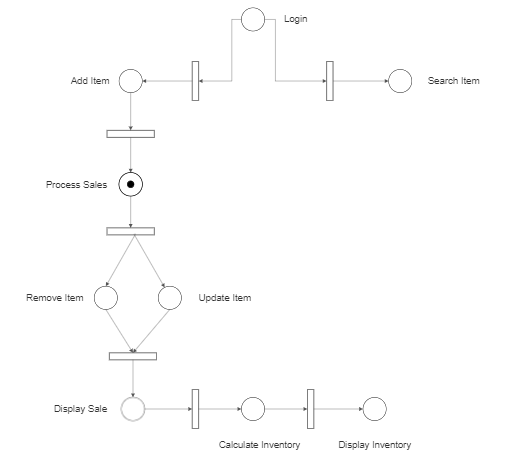
## Petrinet Diagram (firing, matrices):

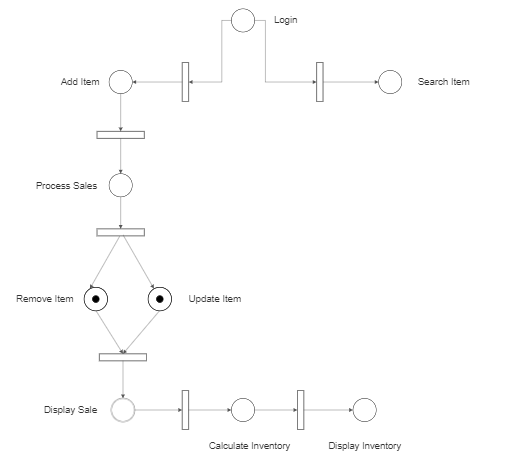


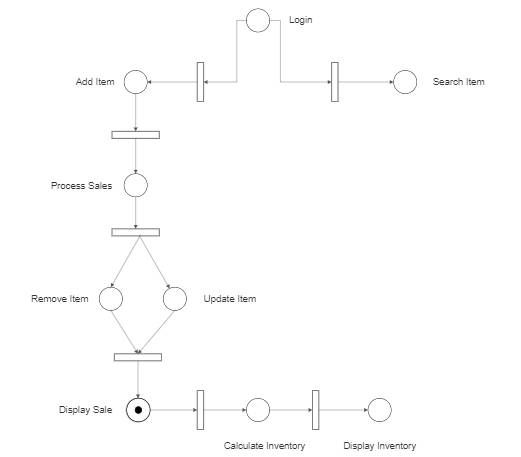
**Firing:**

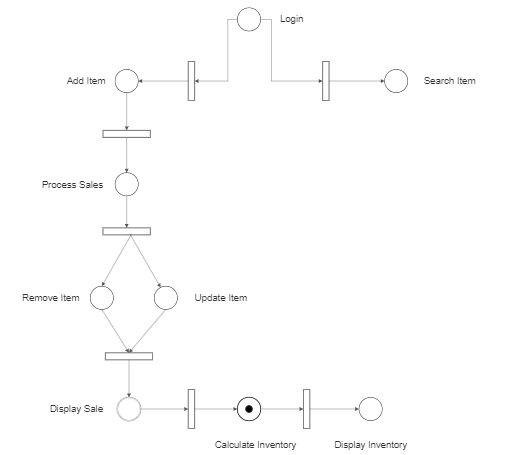


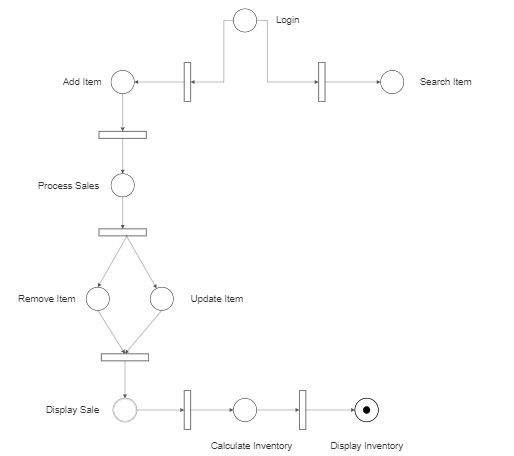




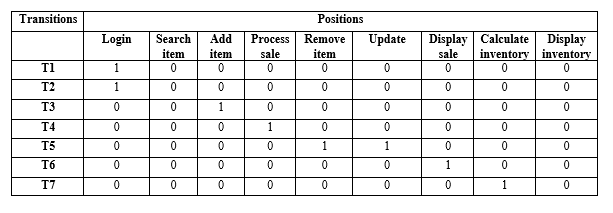




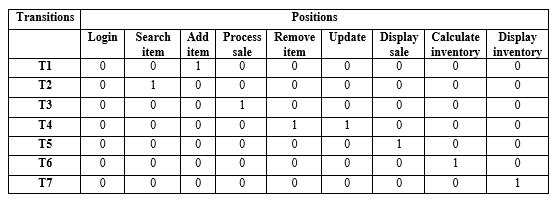




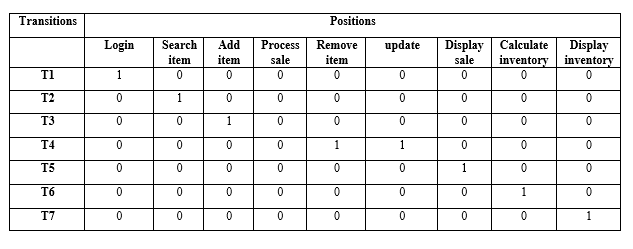
**Input metrices:**

****

**Output metrices:**

****

**Firing Table:**

****

# **D:\Software-Engineering(NUML)\5th Semester\SCD Lab\Project\WhatsApp Image 2023-06-01 at 7.21.05 PM (1).jpegCommunication diagram**: