

Alexandria University
Faculty of Engineering
CSED 2025



OOP Lab 5

Report

Submitted to:

Eng. Ismail El-Yamany

Faculty of Engineering Alex. University

Submitted by:

Islam Yasser Mahmoud 20010312

Ehab Yasser Mahmoud 20010382

Marwan Yasser Sabry 20011870

Mkario Michel Azer 20011982

Faculty of Engineering Alex. University

Jan. 2023

Table of Contents

Problem Statement:	3
Features in the program:	4
Required steps to run the program:	4
User guide:	5
Snapshots of the UI:	12
UML diagram: "Class diagram":	14
How the design patterns are applied:	14
• Producer consumer + Observer:	14
• Memento:	14
• Façade:	14
• Singelton:	14
Design decisions and assumptions:	15

Table of Figures

Figure 0-1: Start-Screen	5
Figure 0-2: Machine command.....	5
Figure 0-3: New Method.....	6
Figure 0-4: Queue command	6
Figure 0-5: New Queue.....	7
Figure 0-6: Join Command	7
Figure 0-7: New Link.....	8
Figure 0-8: Add New Product Command	8
Figure 0-9: New Products	9
Figure 0-10: Play Command.....	9
Figure 0-11: Simulation-Screen.....	10
Figure 0-12: Replay Command.....	10
Figure 0-13: Clear Canvas Command.....	11
Figure 0-14: The Cleared Screen	11

Problem Statement:

A simulation program to simulate an assembly line that produces different products consists of different processing machines Ms that are responsible for processing the product at different stages and queue Qs to handle product movement between different processing stages as a queuing network.

Features in the program:

- Build the production line by:
 - graphically add Qs.
 - graphically add Ms.
 - connect Qs and Ms via UI arbitrarily.
- Simulate the production line.
- Save memento and replay it.

Required steps to run the program:

1. Extract the compressed program folder.
2. For back-end part:
 - Open the **Simulation_Backend** folder using IntelliJ IDE (recommended) or any other java IDE or open it by simply running the pom.xml file.
 - Run the **LastAssignmentApplication** on: src/main/java/com.example.demo.
3. For front-end part:
 - You should have NodeJS and Angular-CLI if you haven't downloaded them.
 - Open the **Simulation_Frontend** folder using VS code IDE (recommended) then copy this to your terminal `npm install` and press Enter.
 - Copy this to your terminal `ng serve` and press Enter. Then the Simulation program is going to run on `localhost:4200` copy it to your MS edge end press Enter, the interface of the program is going to show up and enjoy 😊. (You can replace MS edge by Chrome but first you have to open launch.json in the **Simulation_Frontend** from inside the VS code IDE and replace the `2 msedge` written in the page by `pwa-chrome` and `chrome` respectively.)
 - **Note:** it's better to open the **tsconfig.json** then replace the assignment of strict to false instead of true to avoid any unwanted errors.

User guide:

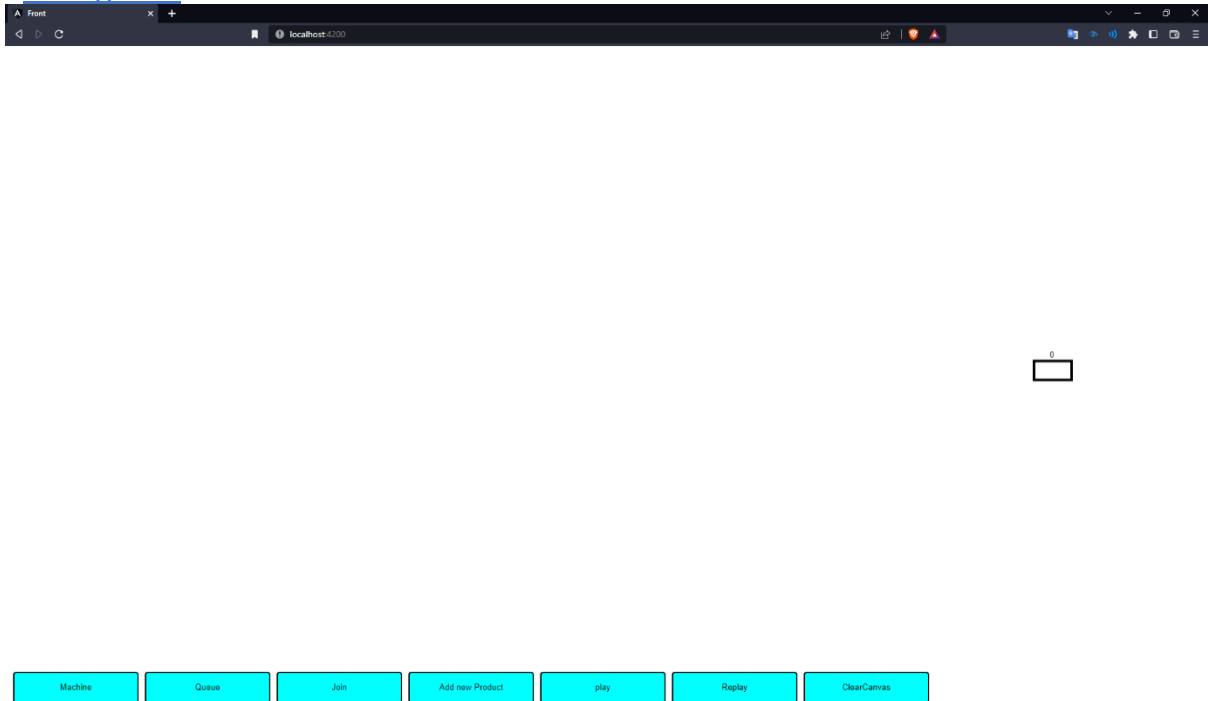


Figure 0-1: Start-Screen

In Fig 0-1: you can see the start screen of the program; you can see that by default Q0 is there.

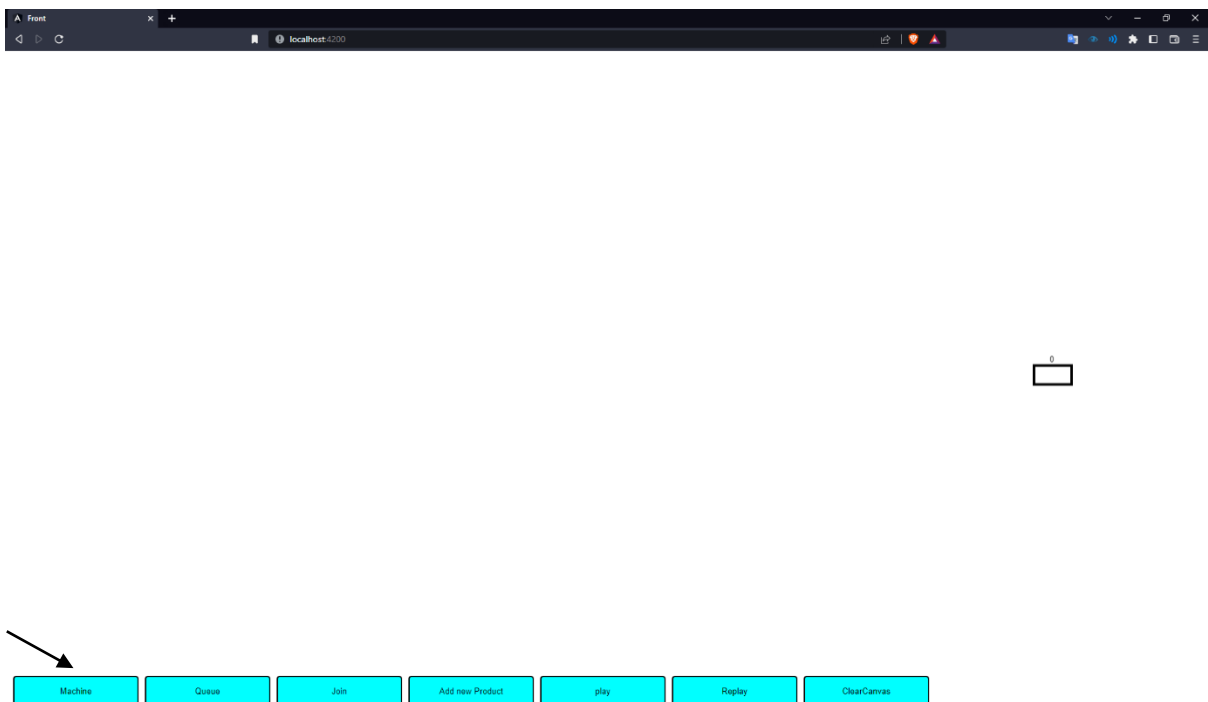


Figure 0-2: Machine command

In Fig 0-2: you can see the arrow referring to the Machine command which will insert a Machine like the following Fig:

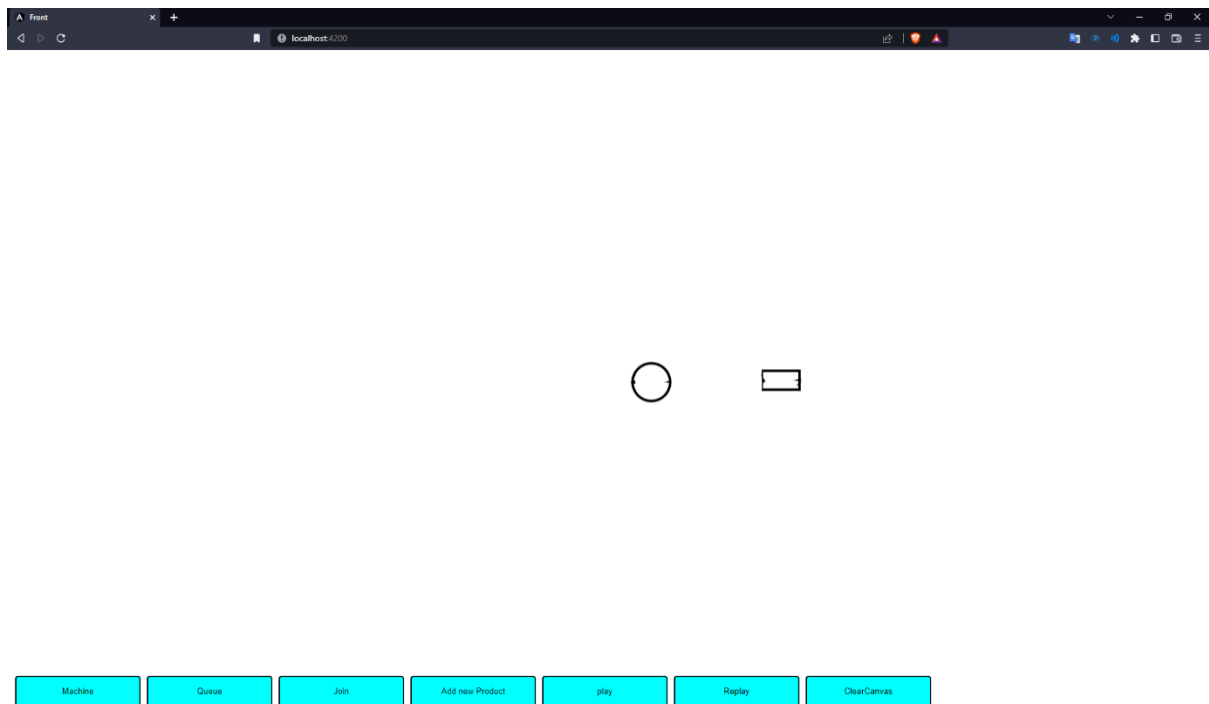


Figure 0-3: New Method



Figure 0-4: Queue command

In Fig 0-4: you can see the arrow referring to the Queue command which will insert a Queue like the following Fig:

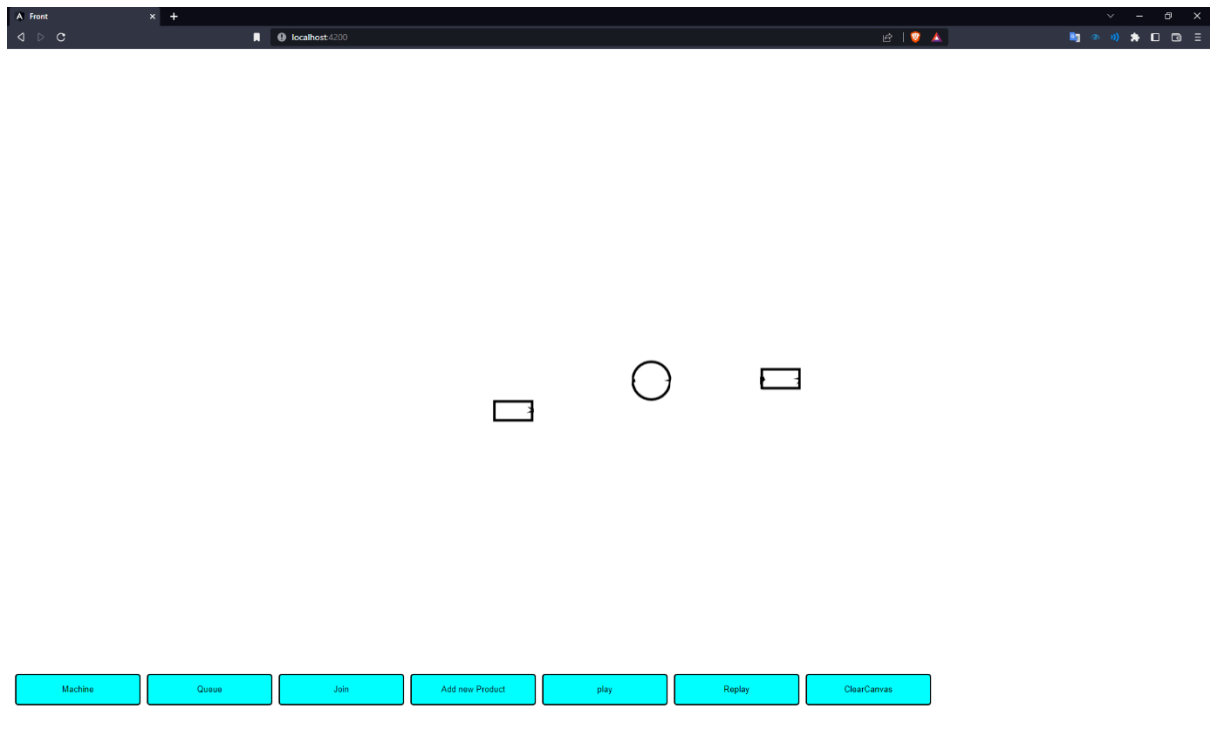


Figure 0-5: New Queue

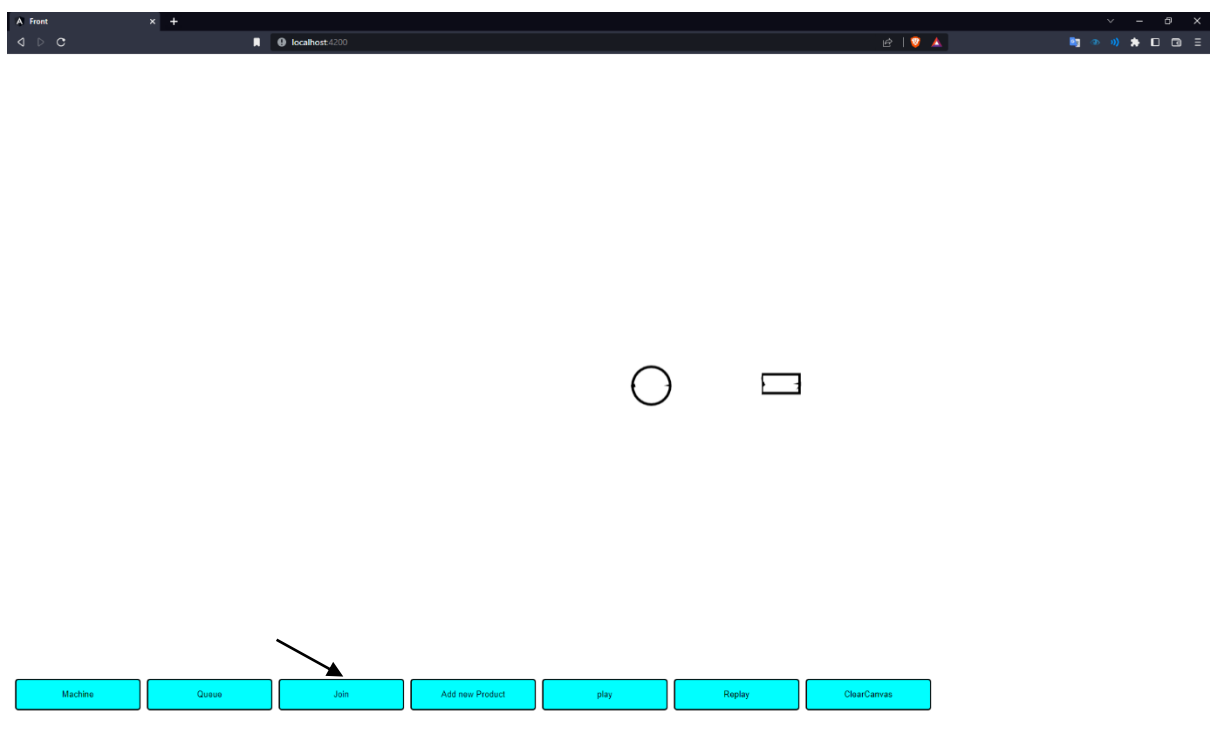


Figure 0-6: Join Command

In Fig 0-6: you can see the arrow referring to the Join command which will Join the two selected component like the following Fig:

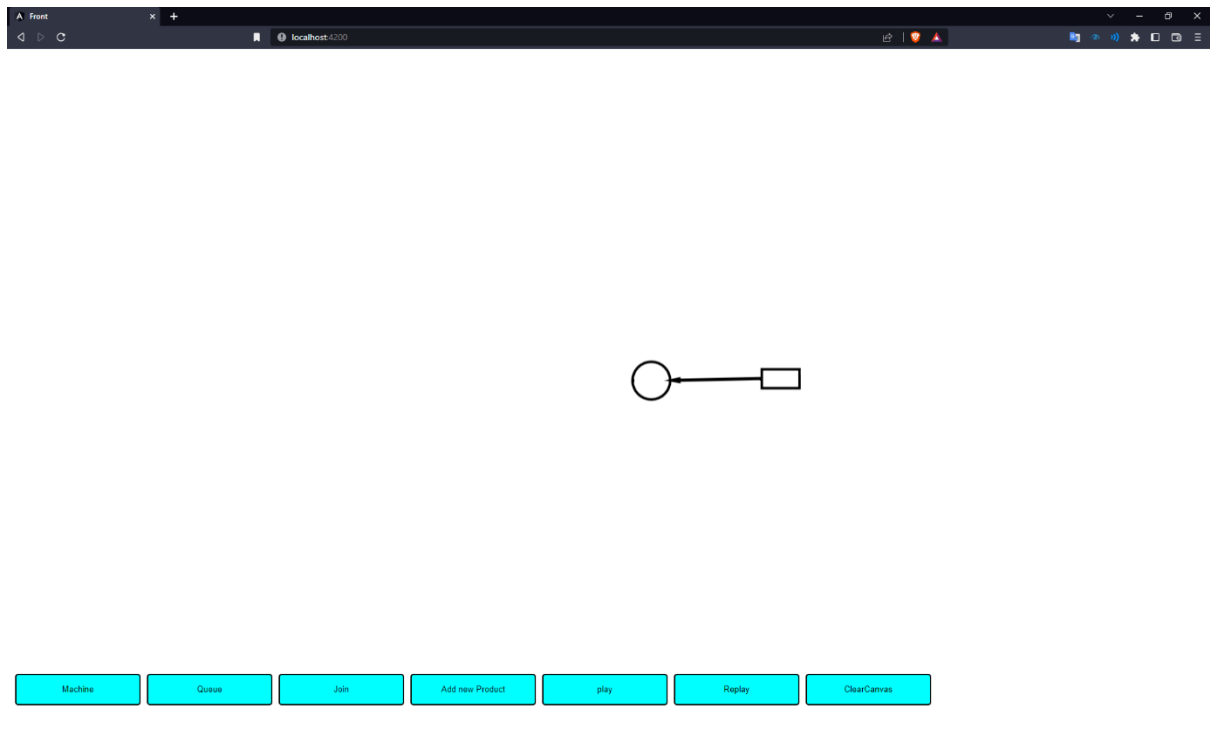


Figure 0-7: New Link

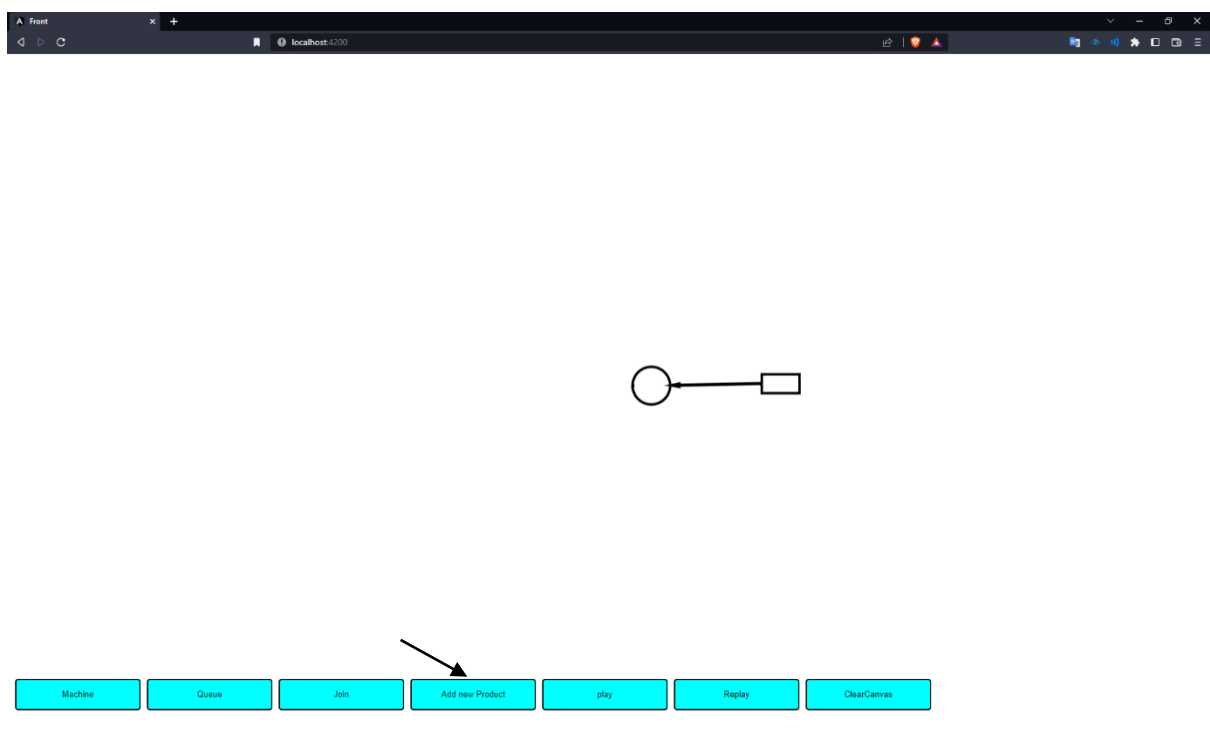


Figure 0-8: Add New Product Command

In Fig 0-8: you can see the arrow referring to the Add new product command which will Add a new product to Q0 like the following Fig:

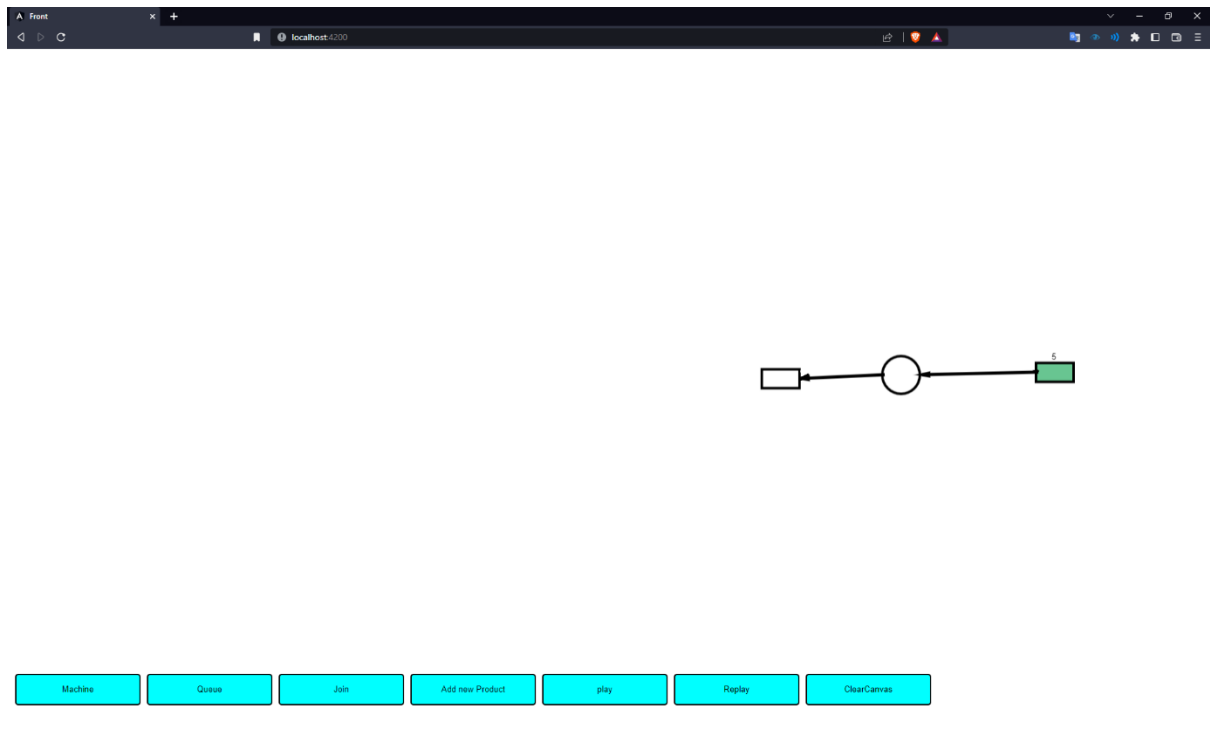


Figure 0-9: New Products

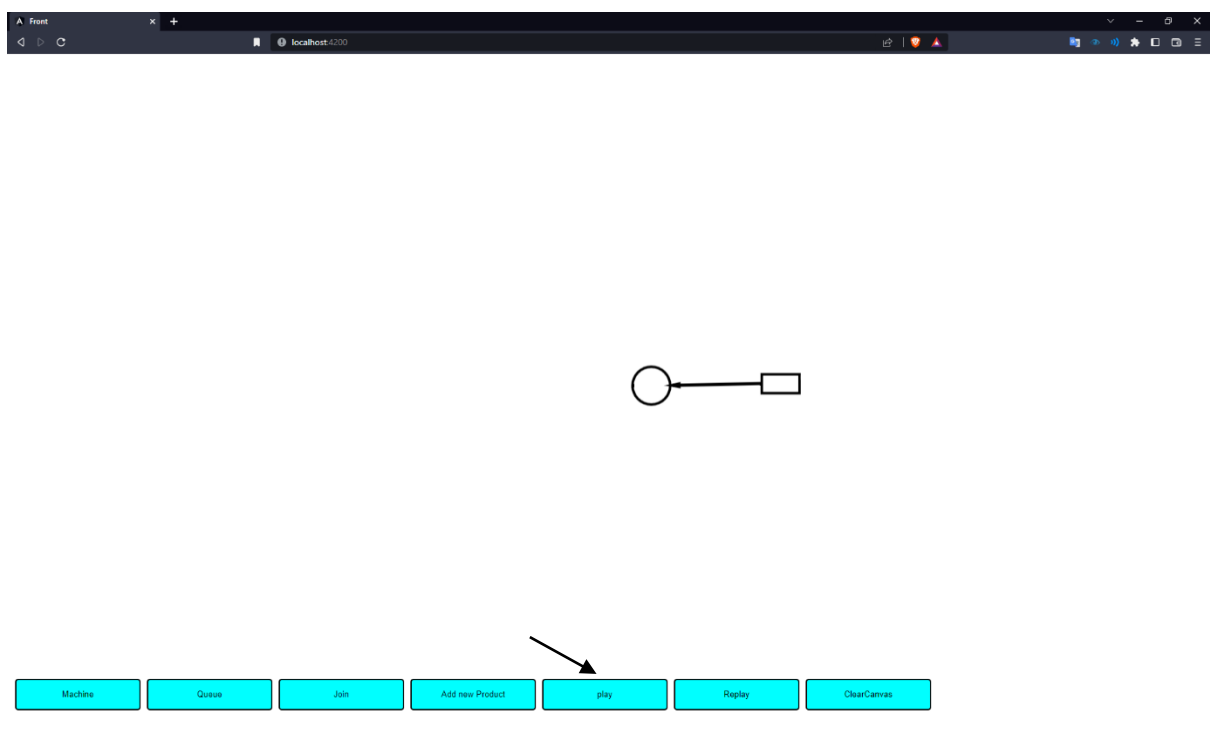


Figure 0-10: Play Command

In Fig 0-10: you can see the arrow referring to the play command which will start the simulation of the product like the following Fig:

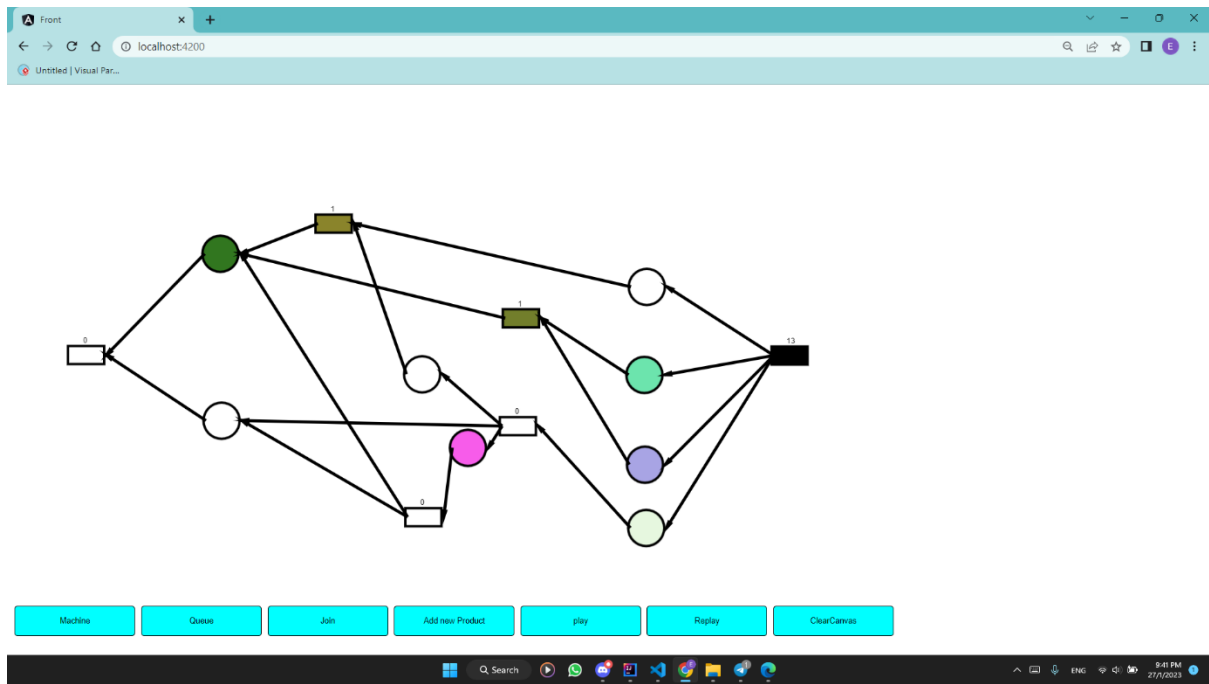


Figure 0-11: Simulation-Screen

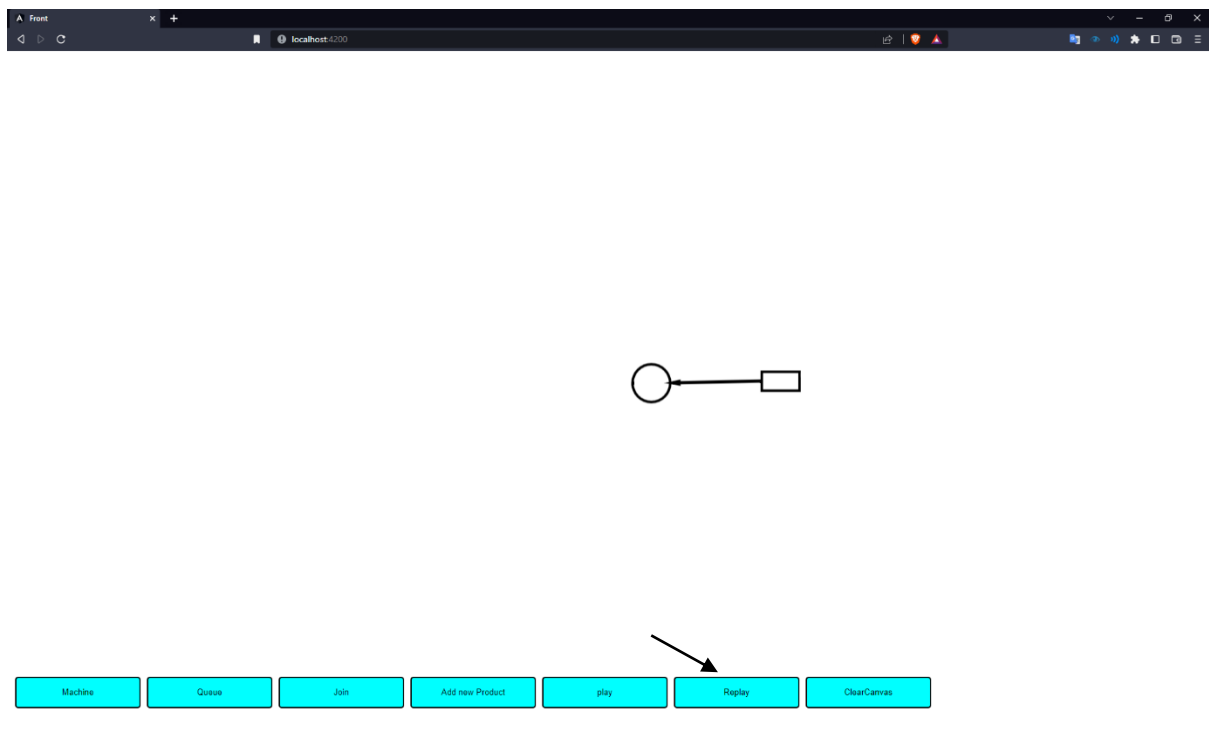


Figure 0-12: Replay Command

In Fig 0-12: you can see the arrow referring to the replay command which will replay the simulation of the product from the beginning.

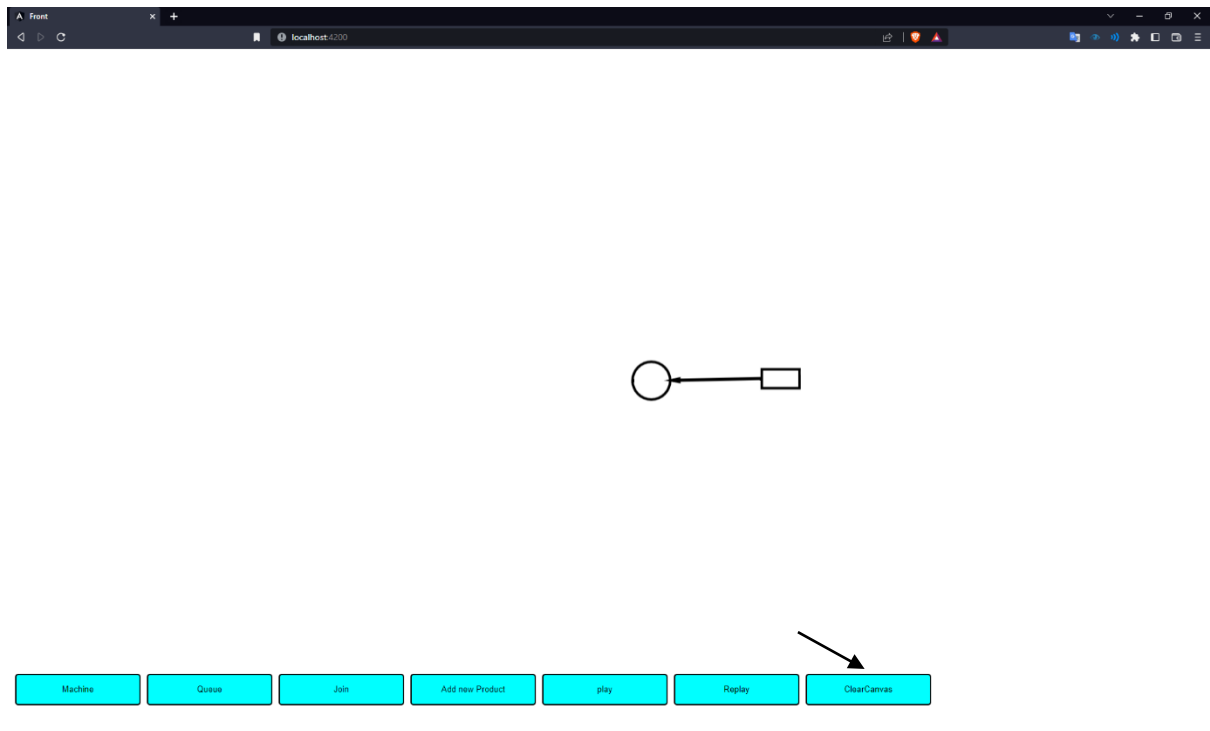


Figure 0-13: Clear Canvas Command

In Fig 0-13: you can see the arrow referring to the Clear Canvas command which will clear the screen and return to the starting screen like the following Fig:

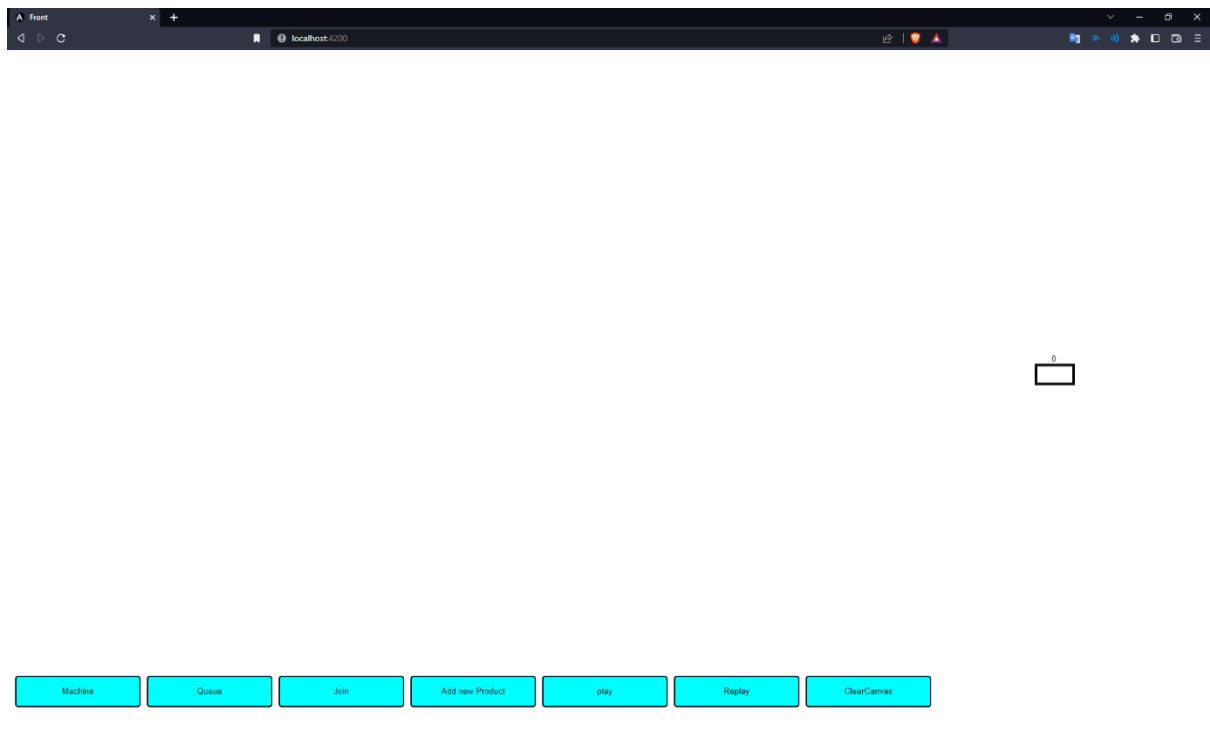
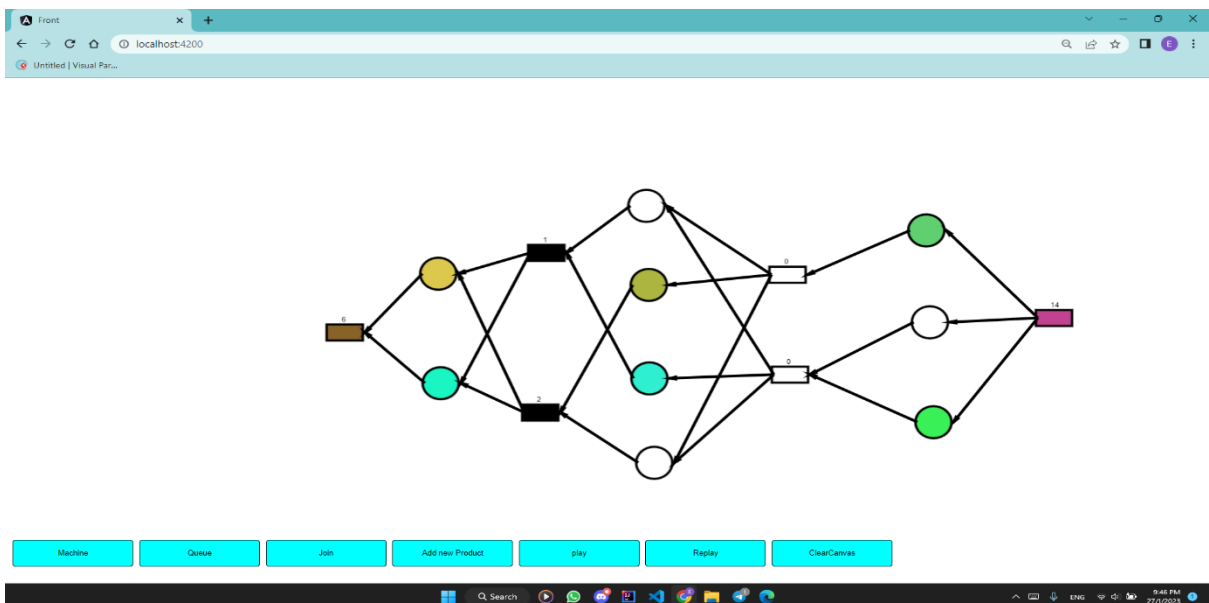
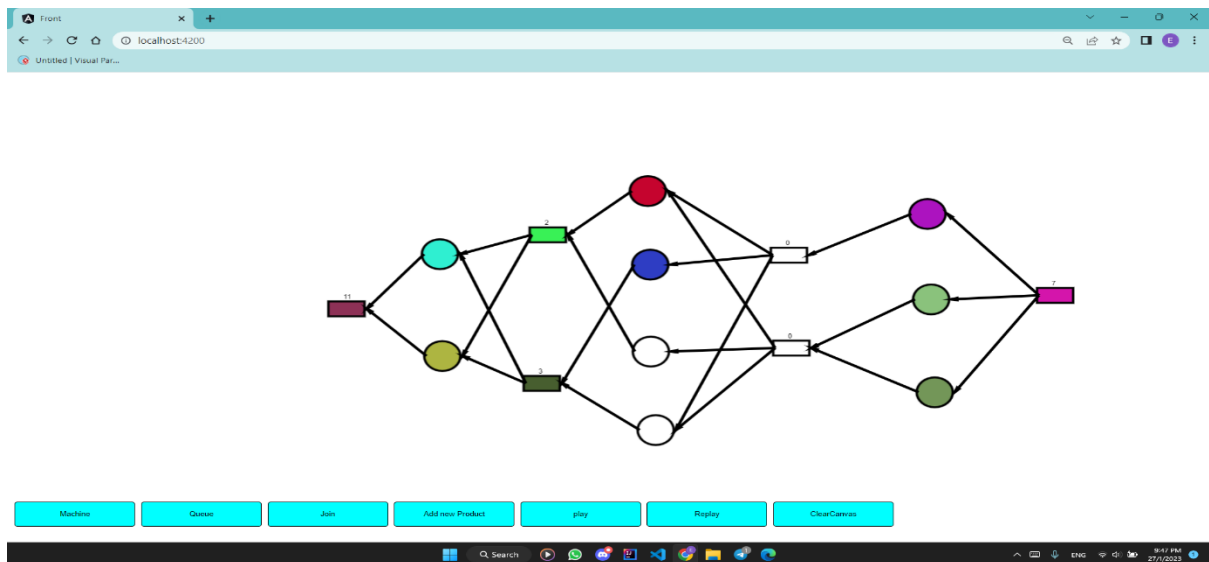
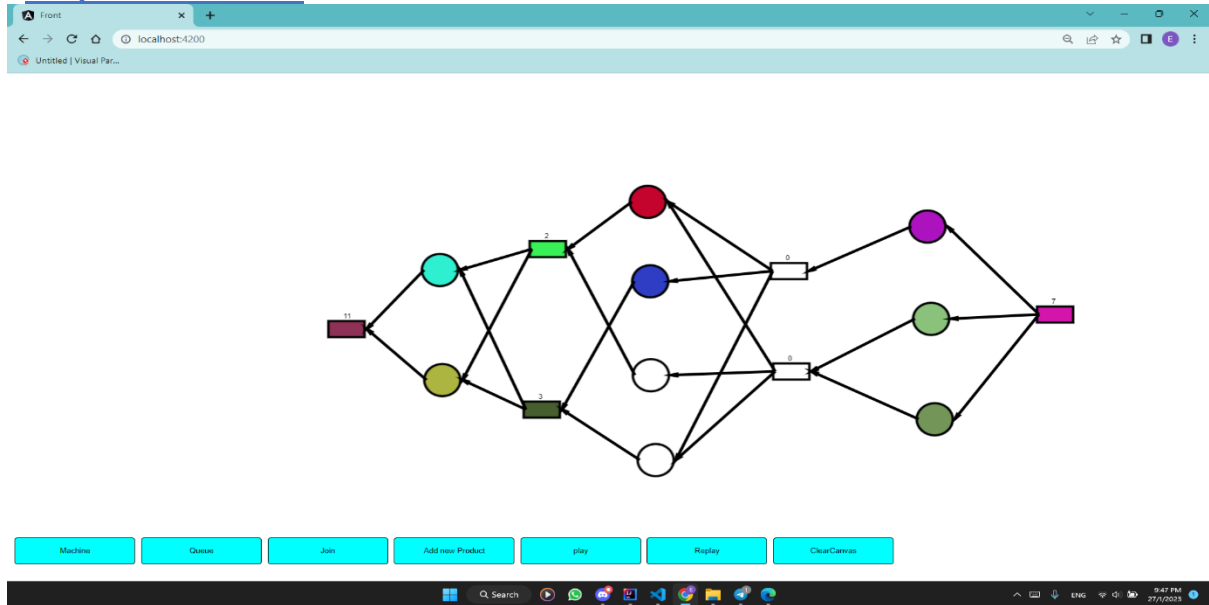
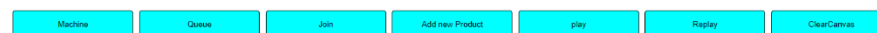
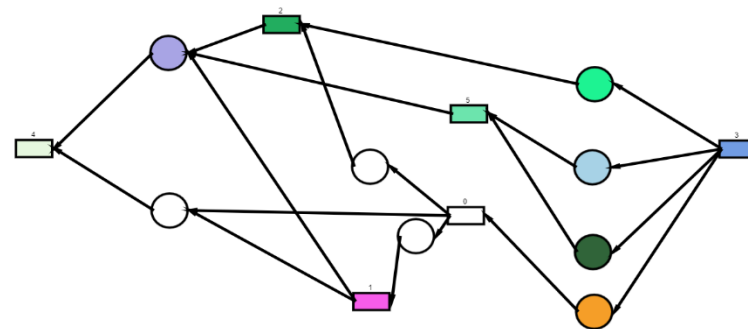
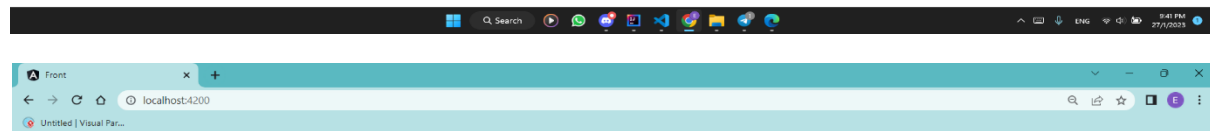
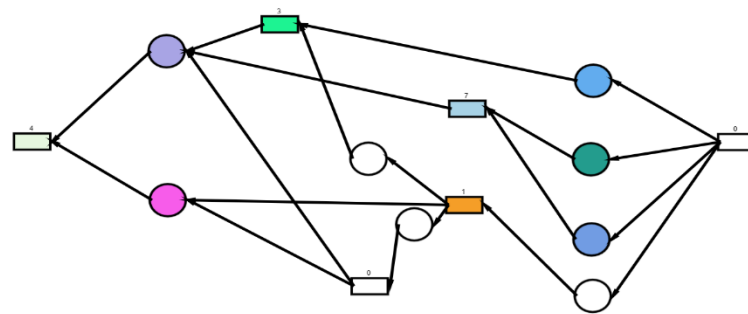
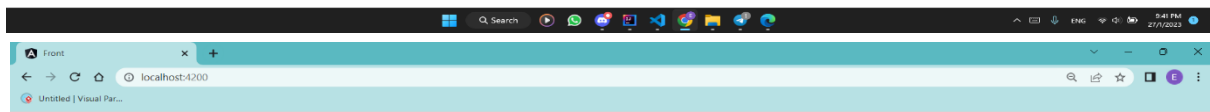
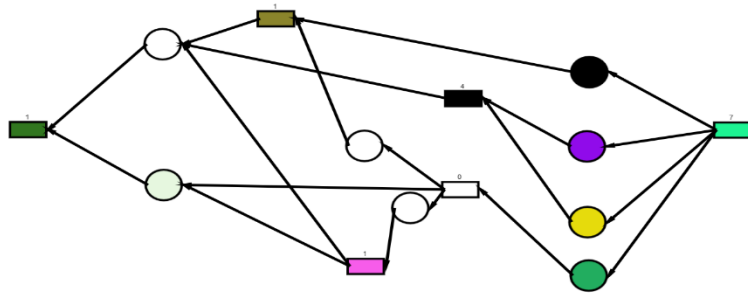


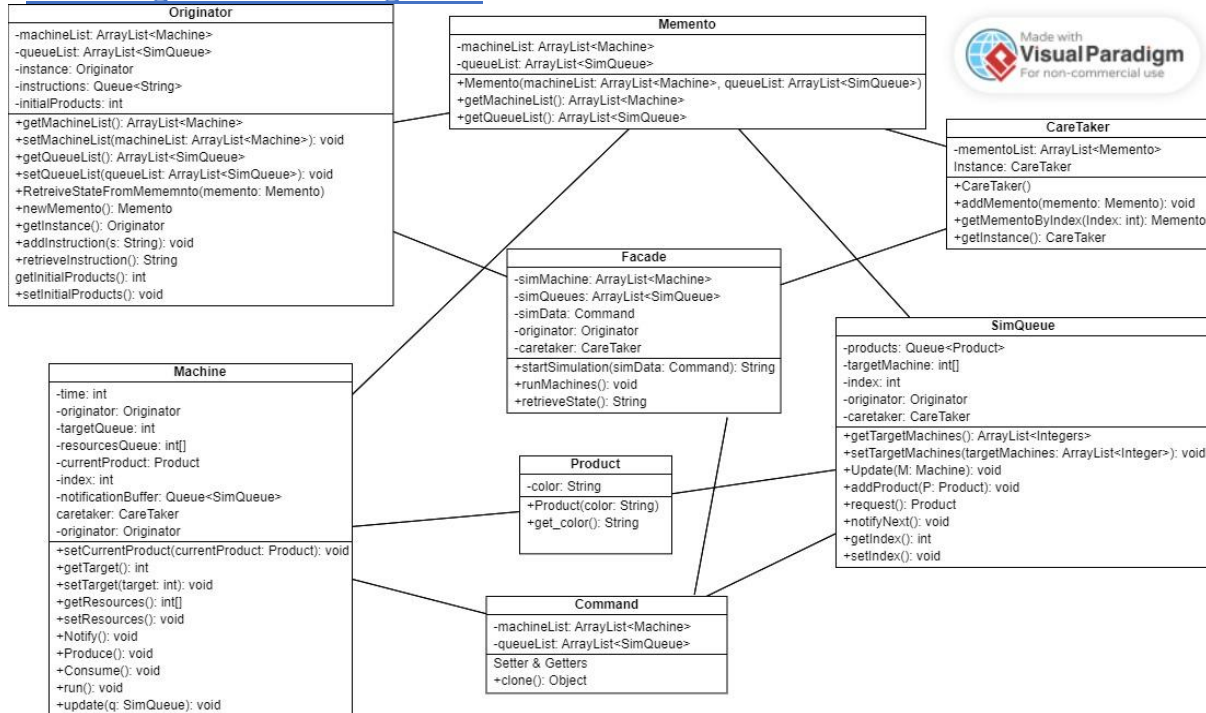
Figure 0-14: The Cleared Screen

Snapshots of the UI:





UML diagram: "Class diagram":



How the design patterns are applied:

- **Producer consumer + Observer:**

Each machine can act as both producer and consumer, so whenever the producer puts a product in the queue it instructs the queue to notify all the machines that are going to be consuming from it, then it returns to its own operation.

If the machine was busy at the time, it keeps the notification in a buffer to respond to it when it finishes its current process even if some other machine responds to the queue and takes the product from it.

If that happens it will simply check if it still has a product to be consumed, if not then it waits for a notification. In this situation the queues act as subject and the consuming machines act as observers

- **Memento:**

we used snapshot design pattern to store the machines and queues in an Originator Object and CareTaker class to store several mementos of objects in an ArrayList.

- **Façade:**

The facade class communicates with all classes in order to begin the simulation, once the simulation has begun, each machine starts its own thread and begins the process.

The class initializes the originator for the snapshot design pattern and readies the machines and queues.

- **Singleton:**

the originator class is a singleton class.

Design decisions and assumptions:

- We used Angular for Front-End coding.
- We used Spring boot for the Back-end coding.
- In front End we have used:
 - npm i konva
 - npm i net -S
- The Color of the Queues is the color of the last product equipped.
- The production line must start with a Queue.
- The production line must end with a Queue.
- The terminal Queue must be the last Queue equipped.