Producer/Consumer Simulation Program

Video Link : https://youtu.be/vz-01DHz4NE

Name 1 :Abdelrahman Ashraf Elmeniawy
ID 1 : 18012538
email : es-abdelrahman.hassan2023@alexu.edu.eg

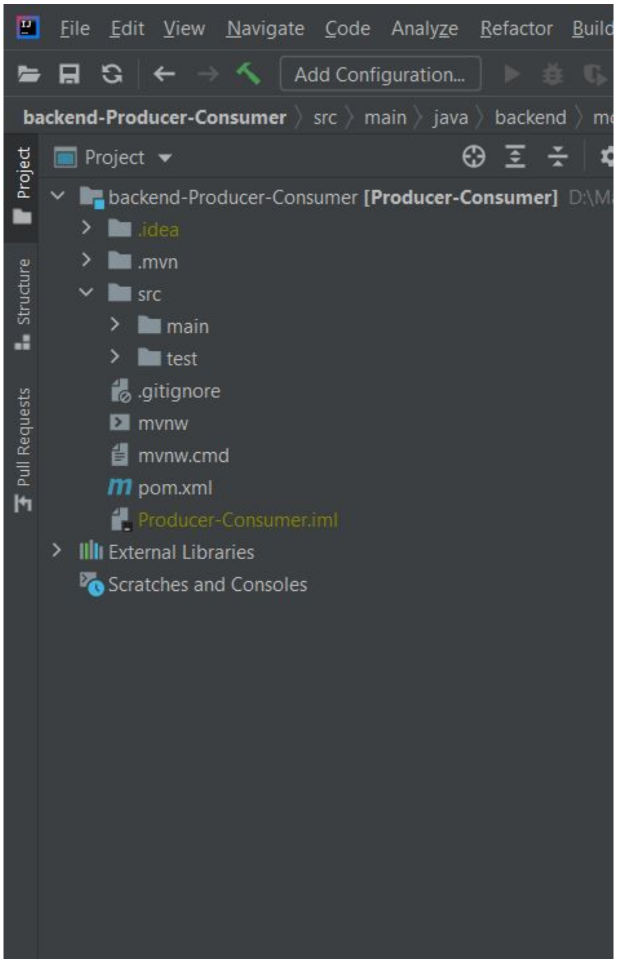
Name 2 : Mina Henen Shafik
ID 2 : 18011939
email : es-MinaHenen2023@alexu.edu.eg

Name 3 : Mark Magdy Nasr
ID 3 : 18011304
email : es-mark.magdy2019@alexu.edu.eg

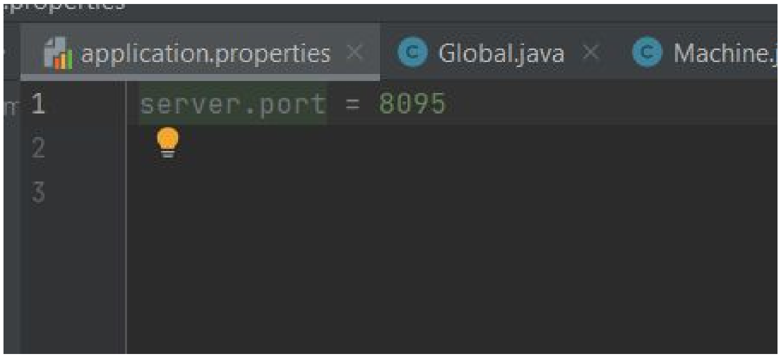
Name 4 : Mark Nader Fathy
ID 4 : 18011305
email : es-mark.nader2018@alexu.edu.eg

Steps Required To Run The Code

- **Import backend-paint folder to IDE (like intelliJ)**

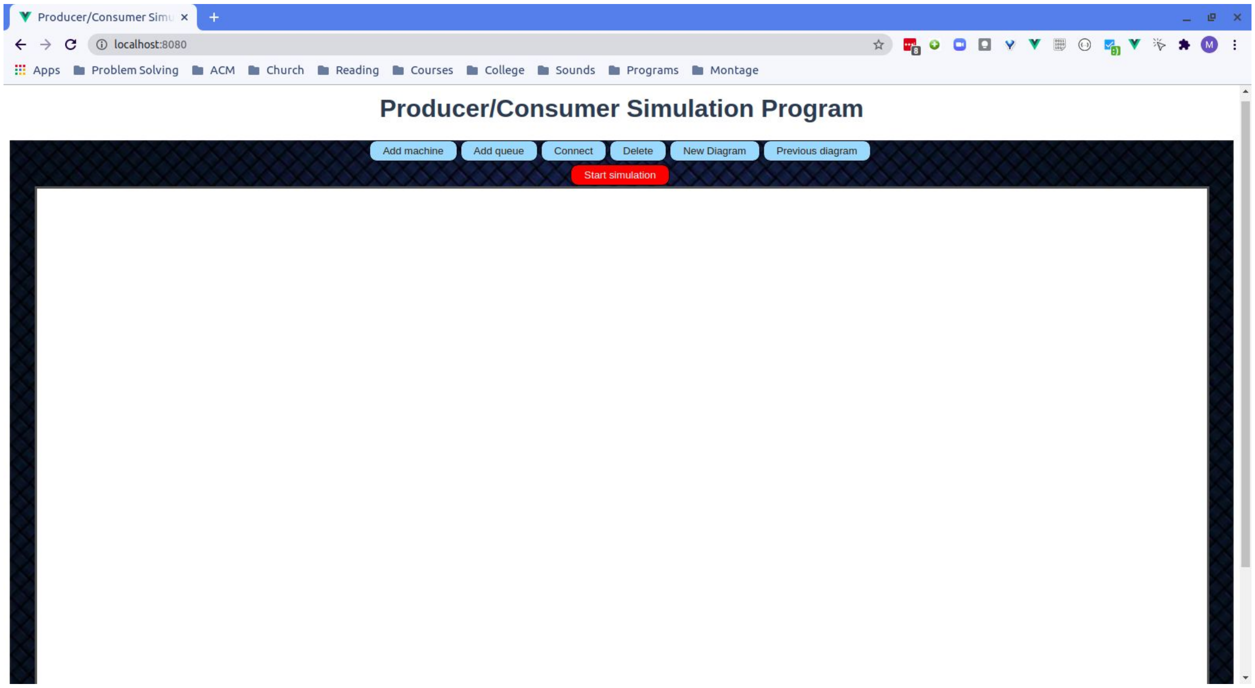

File Edit View Navigate Code Analyze Refactor Build
Add Configuration...
backend-Producer-Consumer 〉 src 〉 main 〉 java 〉 backend 〉 mo
Project
Structure
Pull Requests
backend-Producer-Consumer [Producer-Consumer] D:\M
.idea
.mvn
src
main
test
.gitignore
mvnw
mvnw.cmd
pom.xml
Producer-Consumer.iml
External Libraries
Scratches and Consoles

- **Run the backend**
- **It will run on server port 8095**



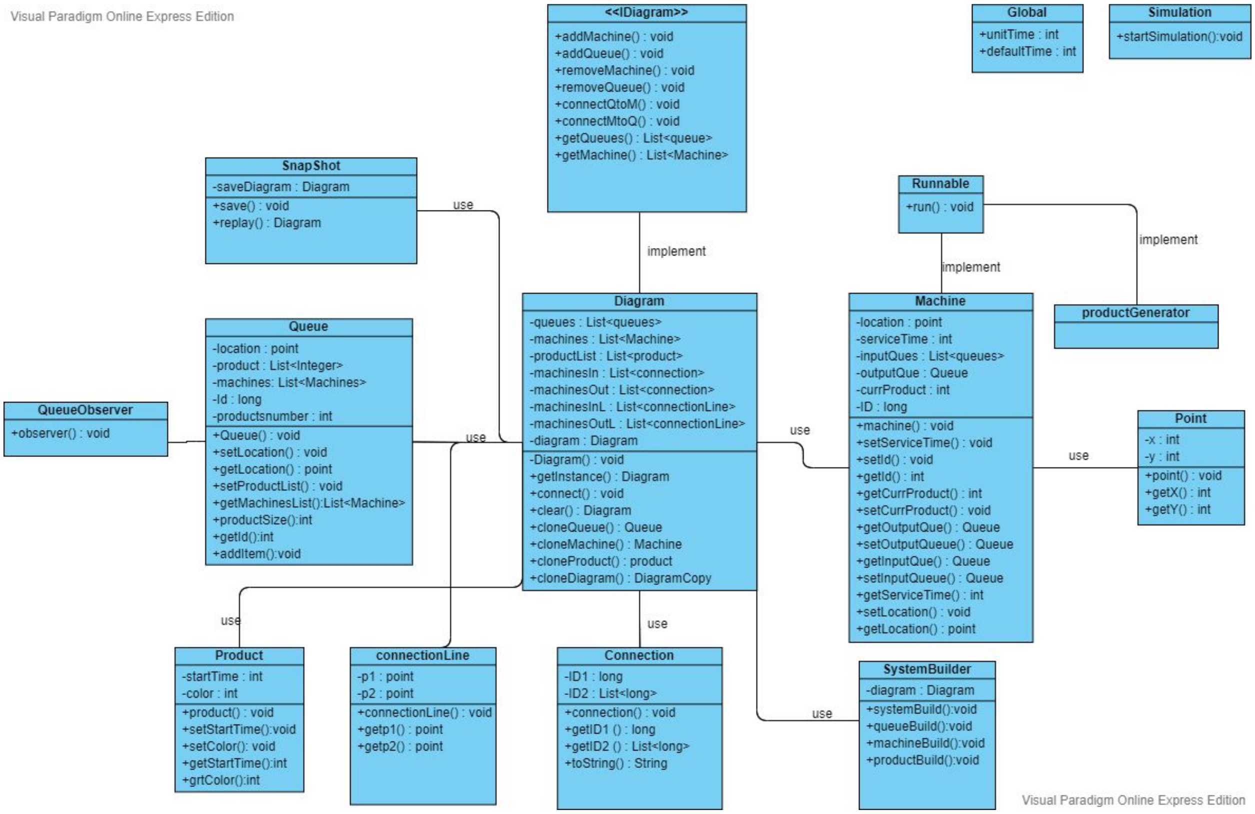
```
server.port = 8095
```

- **Import frontend-paint to visual studio code**
- **Write in terminal**
    - Yarn install
    - Yarn serve



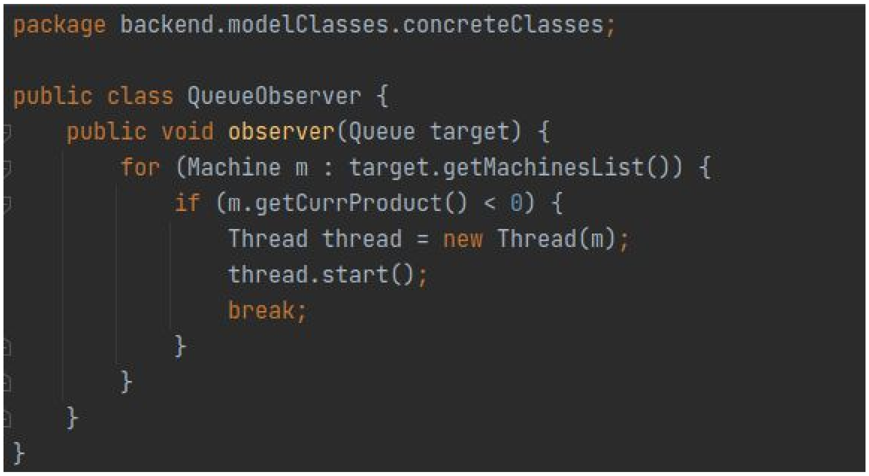
Producer/Consumer Simulation Program

UML Class Diagram



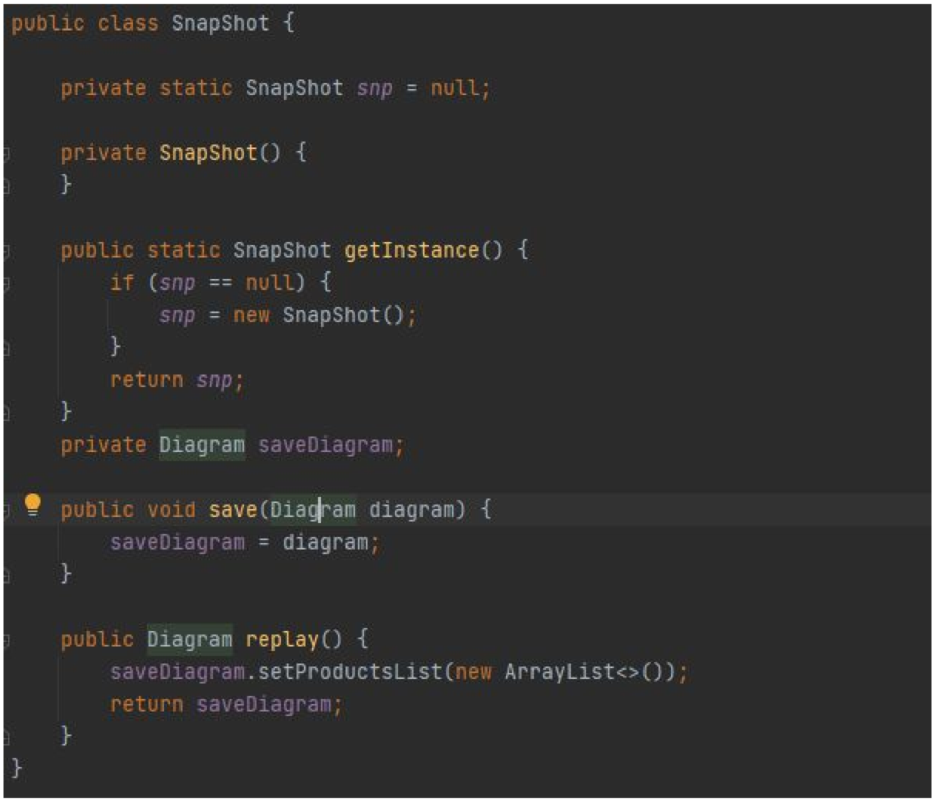
Design Patterns Applied

- Observer design pattern
  - It was applied in the QueueObserver Class to update the changes happens in the queue



```
package backend.modelClasses.concreteClasses;

public class QueueObserver {
    public void observer(Queue target) {
        for (Machine m : target.getMachinesList()) {
            if (m.getCurrProduct() < 0) {
                Thread thread = new Thread(m);
                thread.start();
                break;
            }
        }
    }
}
```

- Memento (snapshot) design pattern
  - It's applied in the SnapShot class for re simulating a previous diagram.

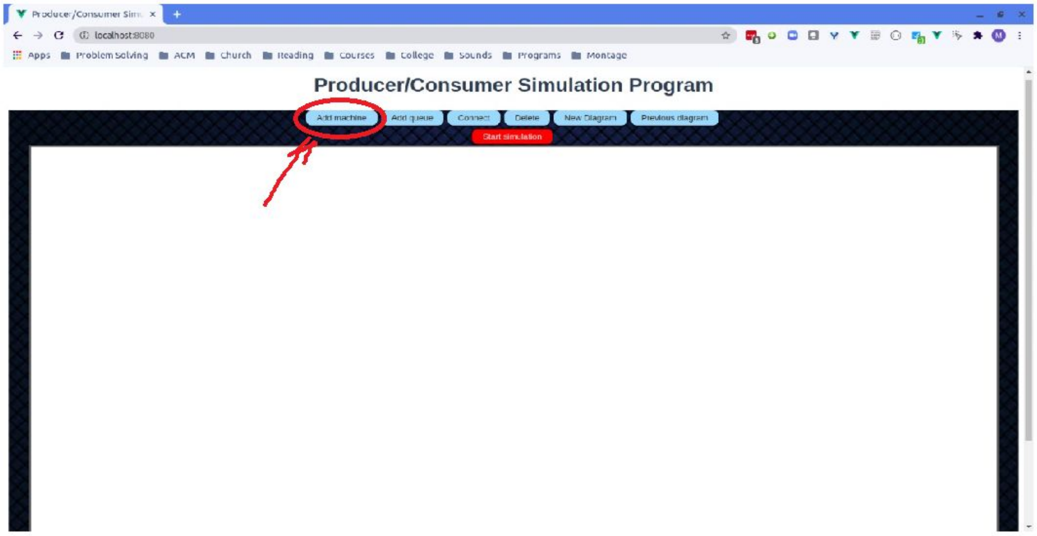


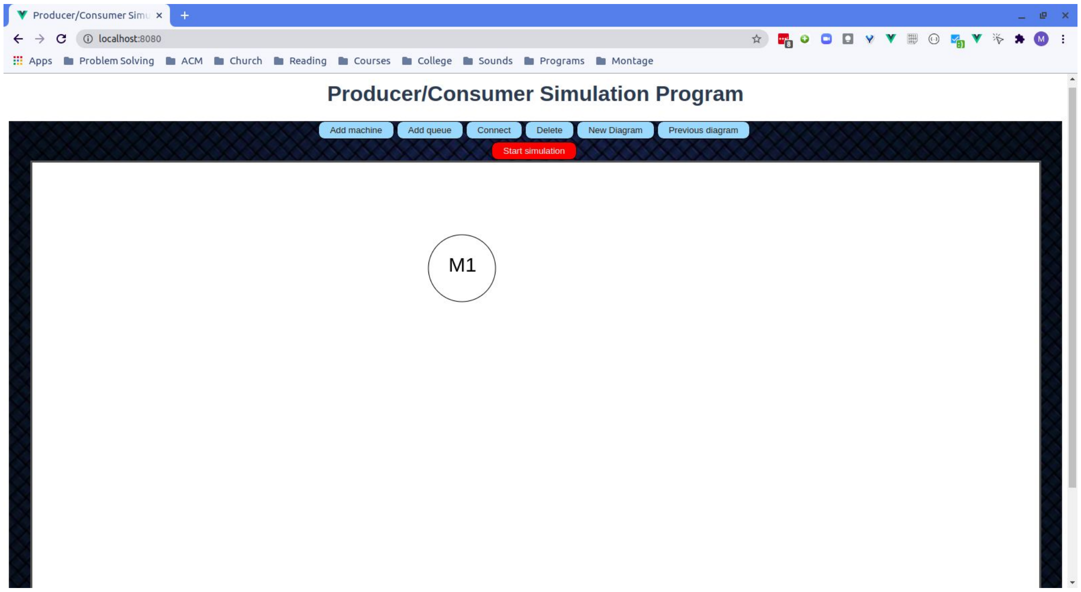
```
public class SnapShot {

    private static SnapShot snp = null;

    private SnapShot() {
    }

    public static SnapShot getInstance() {
        if (snp == null) {
            snp = new SnapShot();
        }
        return snp;
    }
    private Diagram saveDiagram;

    public void save(Diagram diagram) {
        saveDiagram = diagram;
    }

    public Diagram replay() {
        saveDiagram.setProductsList(new ArrayList<>());
        return saveDiagram;
    }
}
```

- concurrency design pattern
  - All the application is a simulation of the producer-consumer design pattern.

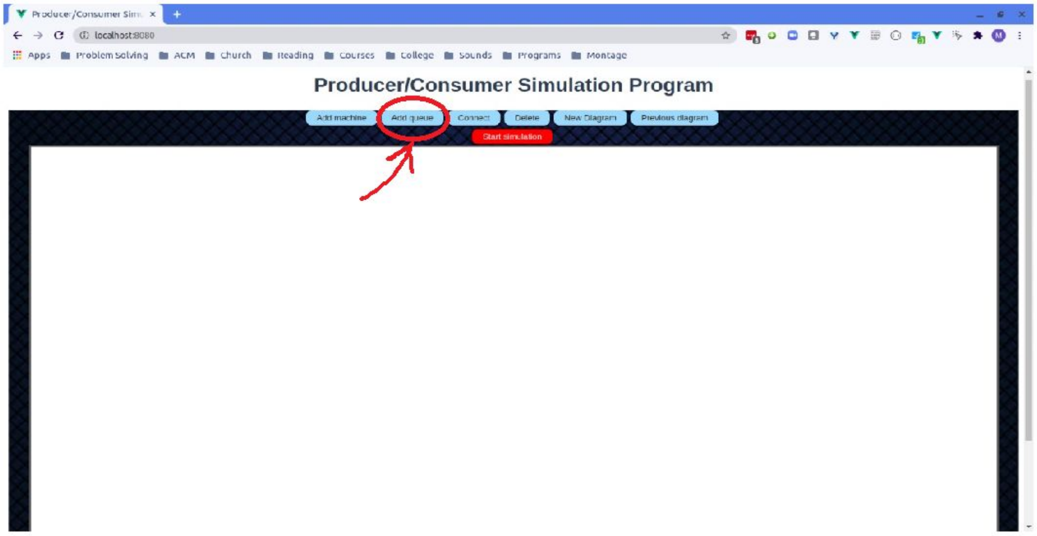
User Guide

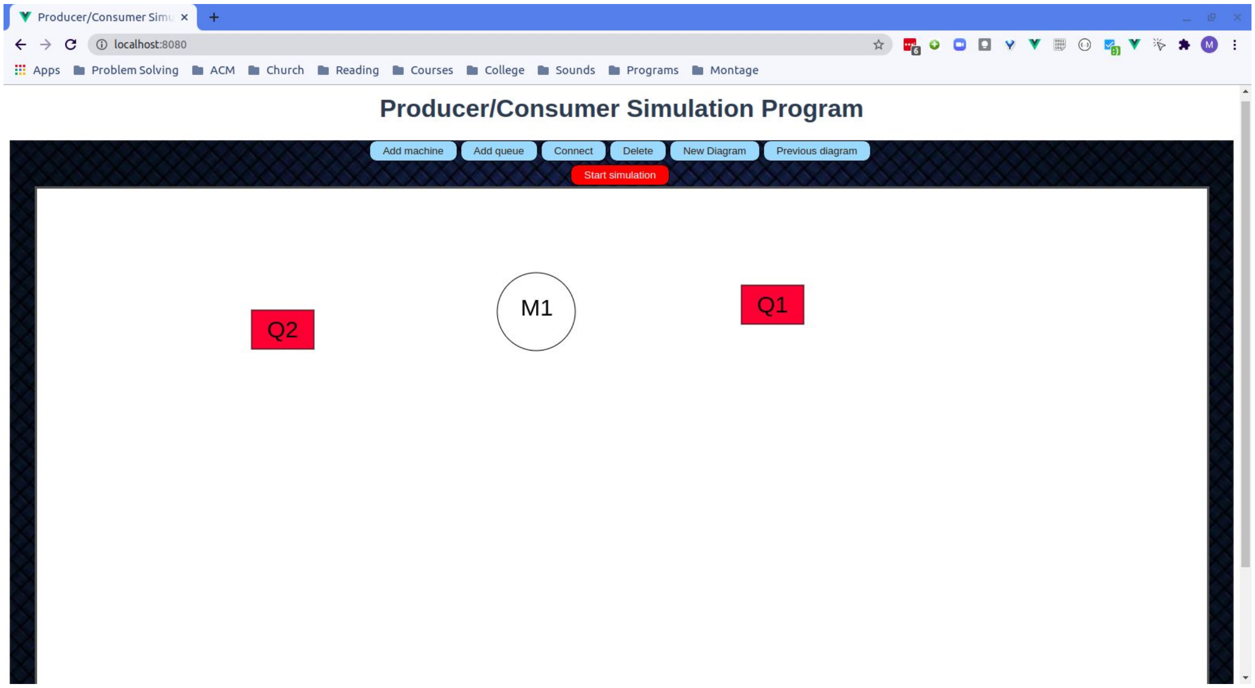
- To add machine
  - Click add machine button then click where you want to draw the machine



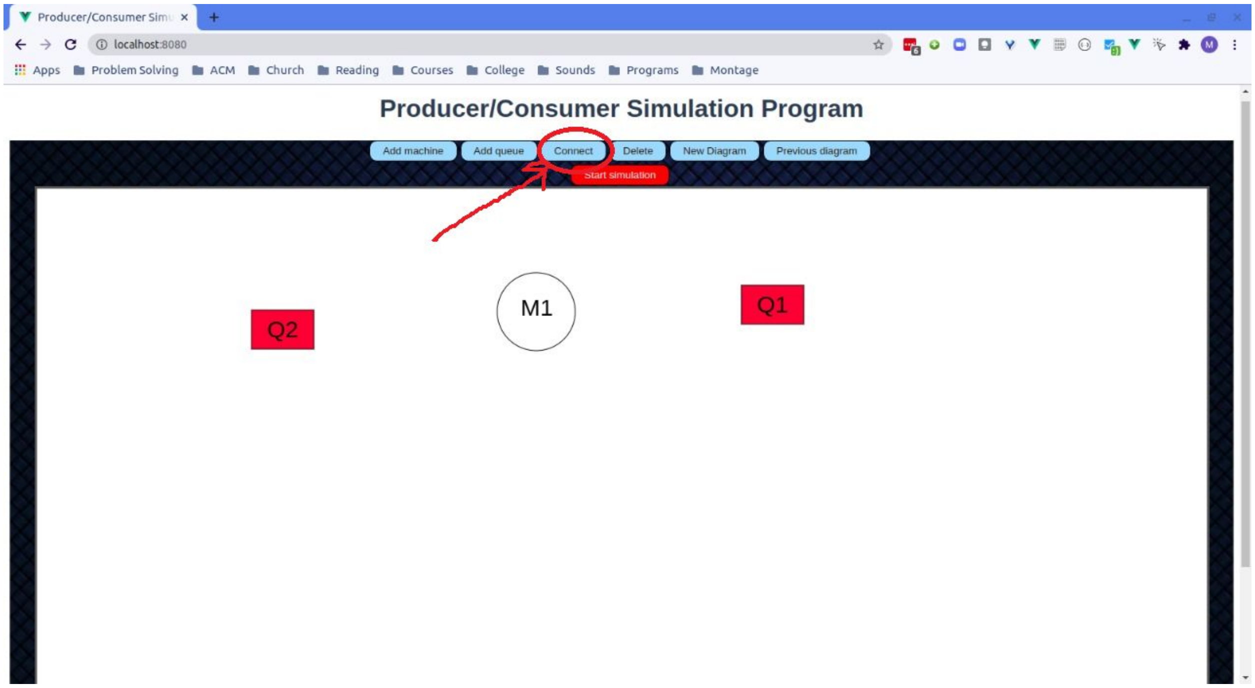


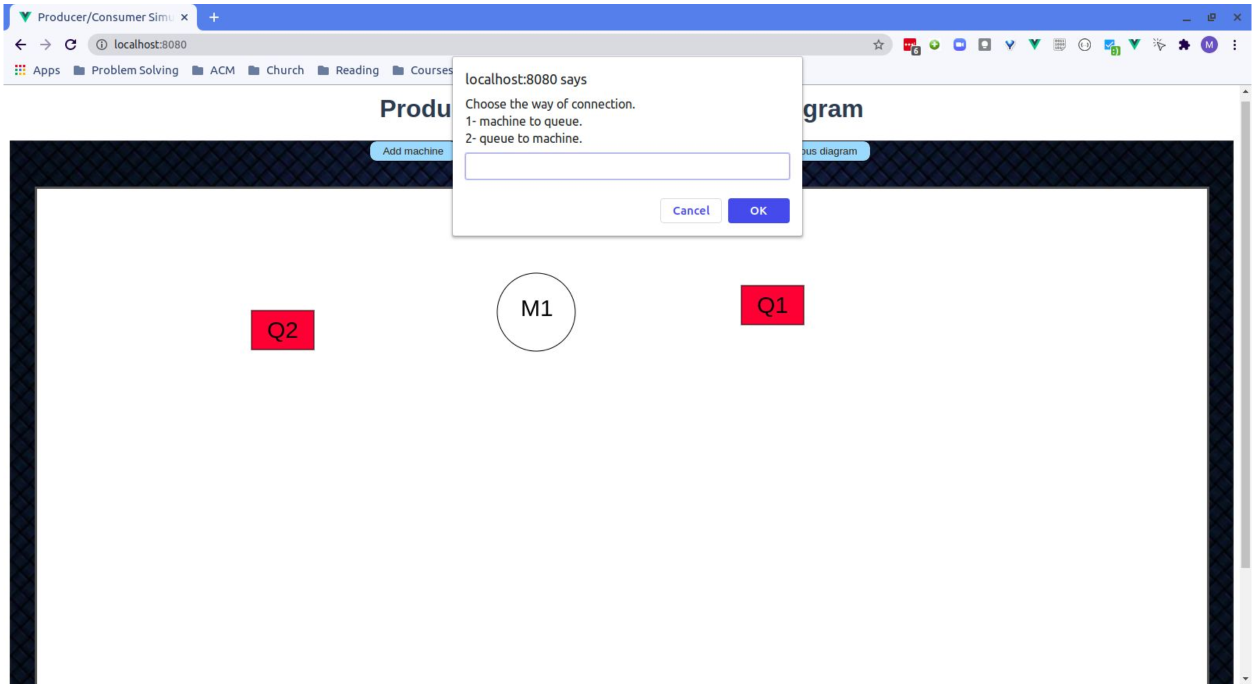
- To add queue
    - Click add queue button then click where you want to draw the queue





- To connect between queue and machine or machine and queue
  - Click connect button and choose by type 1 or 2

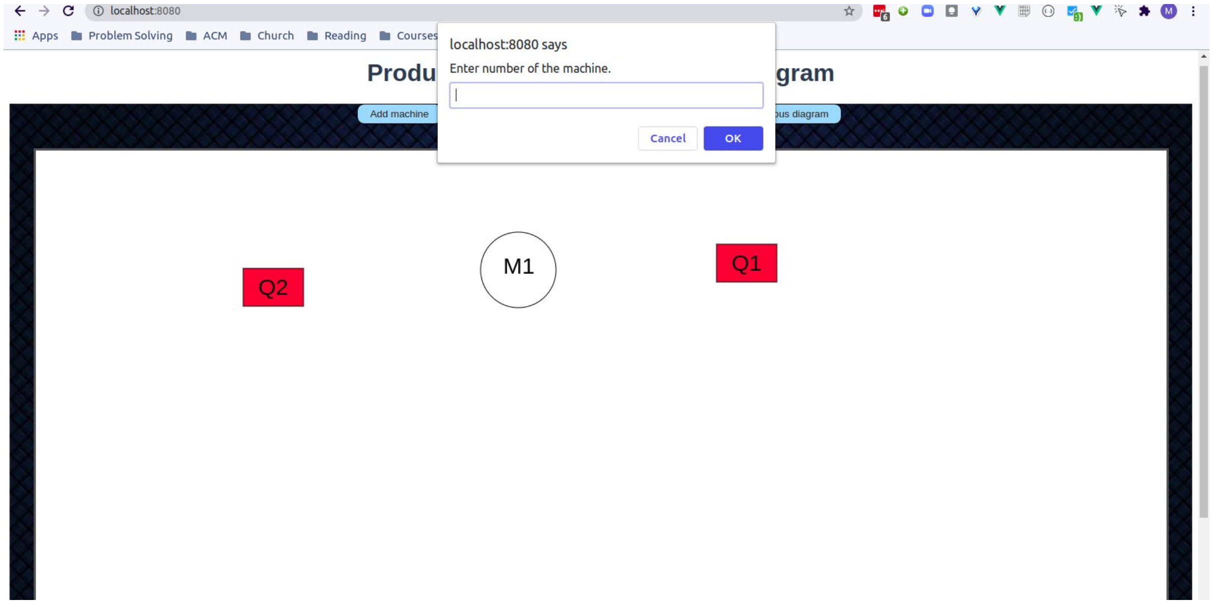




localhost:8080 says

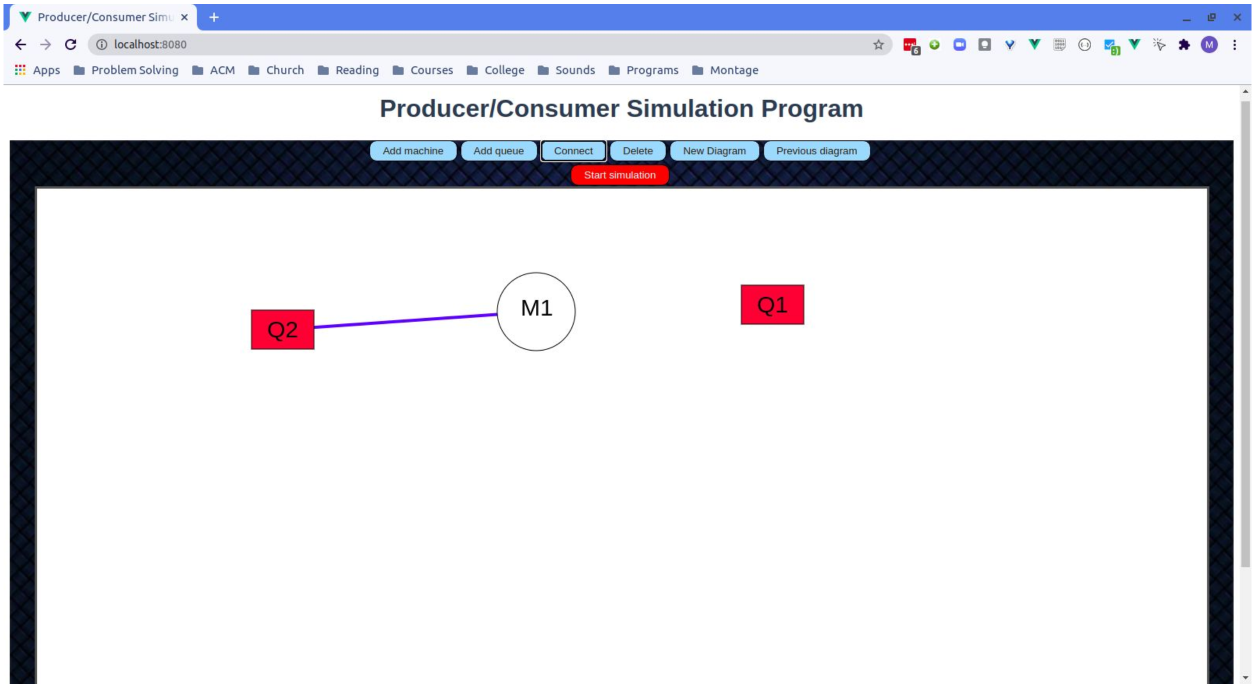
Choose the way of connection.
1- machine to queue.
2- queue to machine.

- Then enter the number of queue and the number of machine that you want to connect them

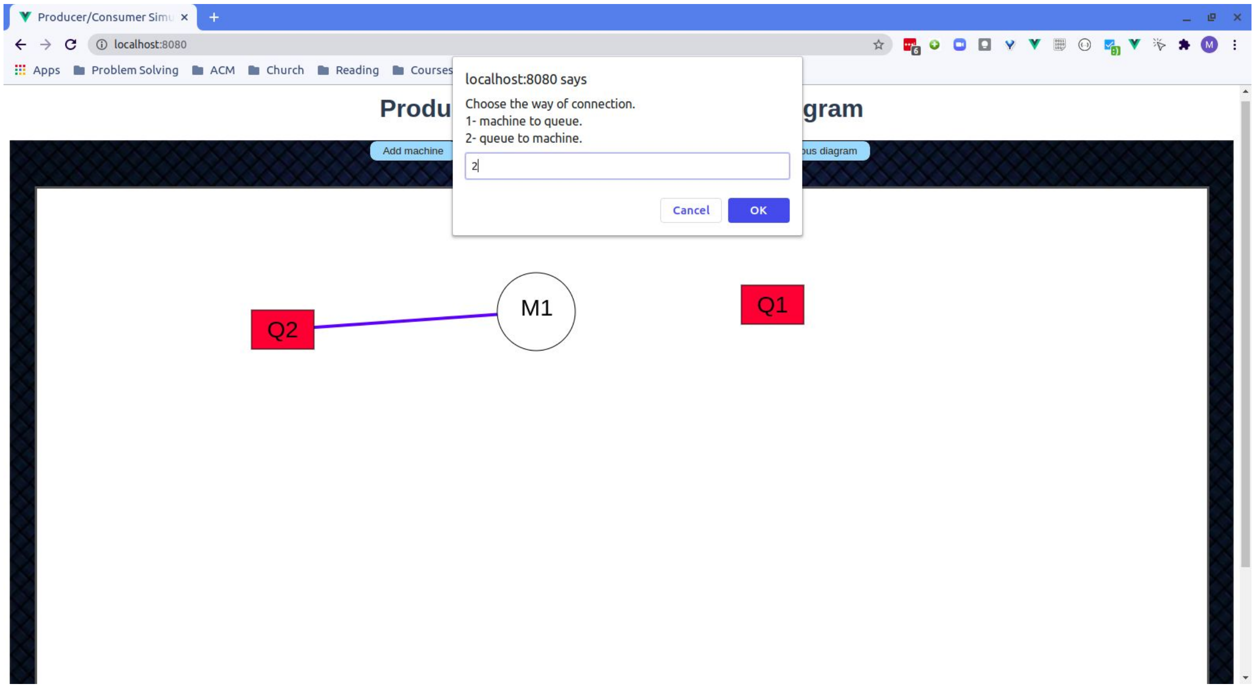


localhost:8080 says

Enter number of the machine.



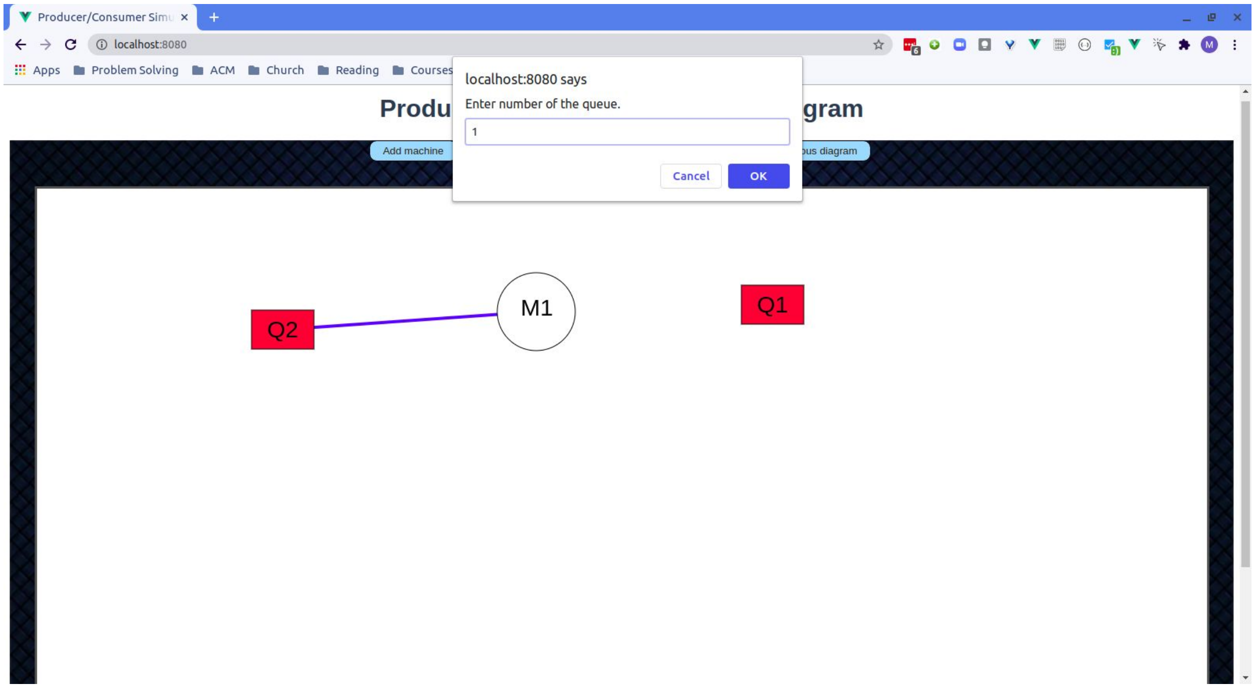
- ○ Let’s make another connection (Queue 1 is the input of Machine 1)



localhost:8080 says

Choose the way of connection.
1- machine to queue.
2- queue to machine.

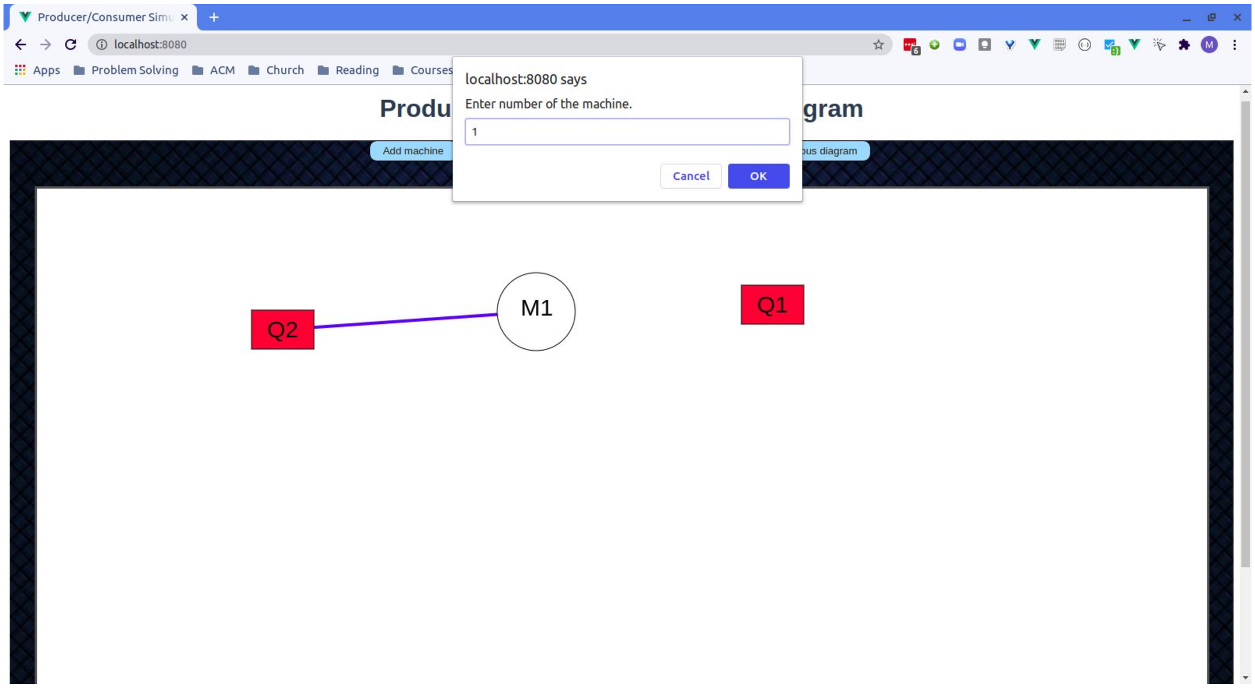
2



localhost:8080 says

Enter number of the queue.

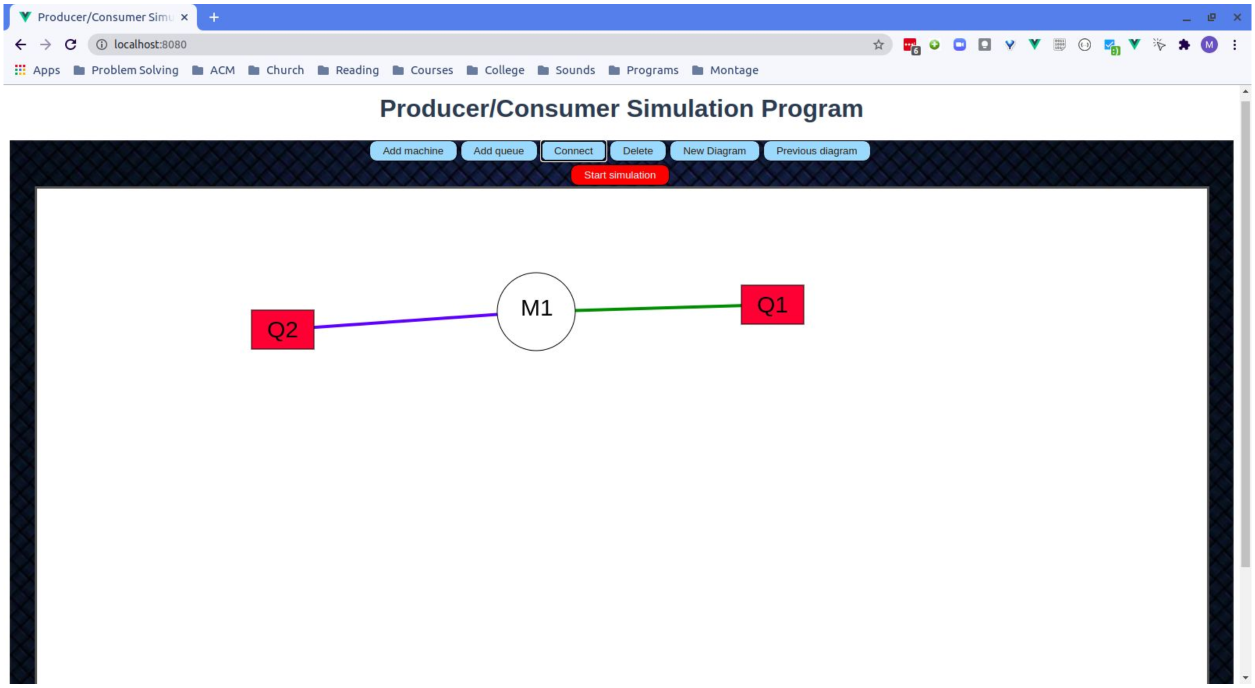
1



localhost:8080 says

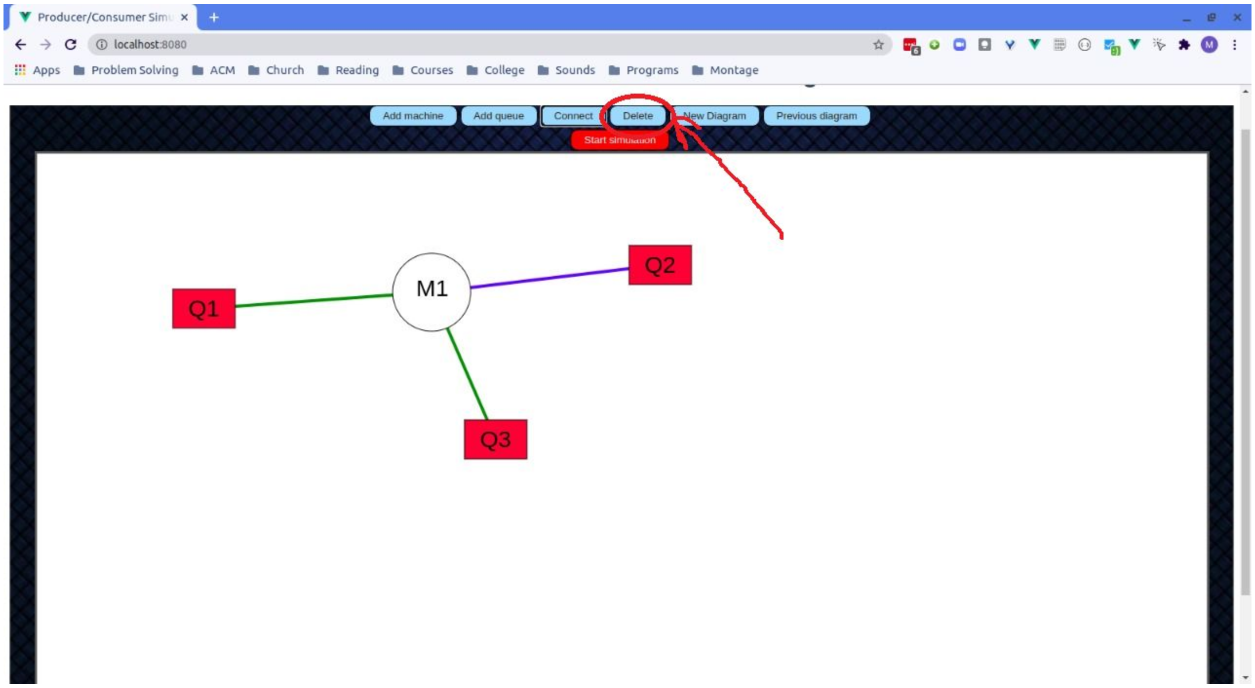
Enter number of the machine.

1

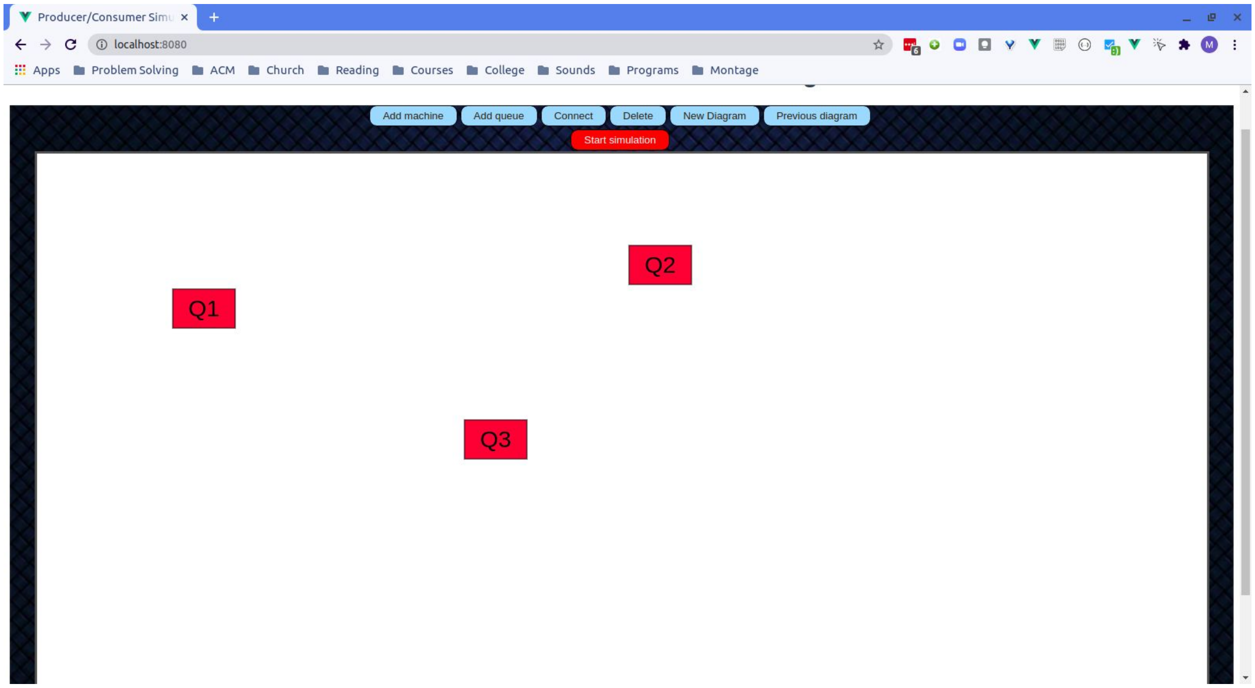


Producer/Consumer Simulation Program

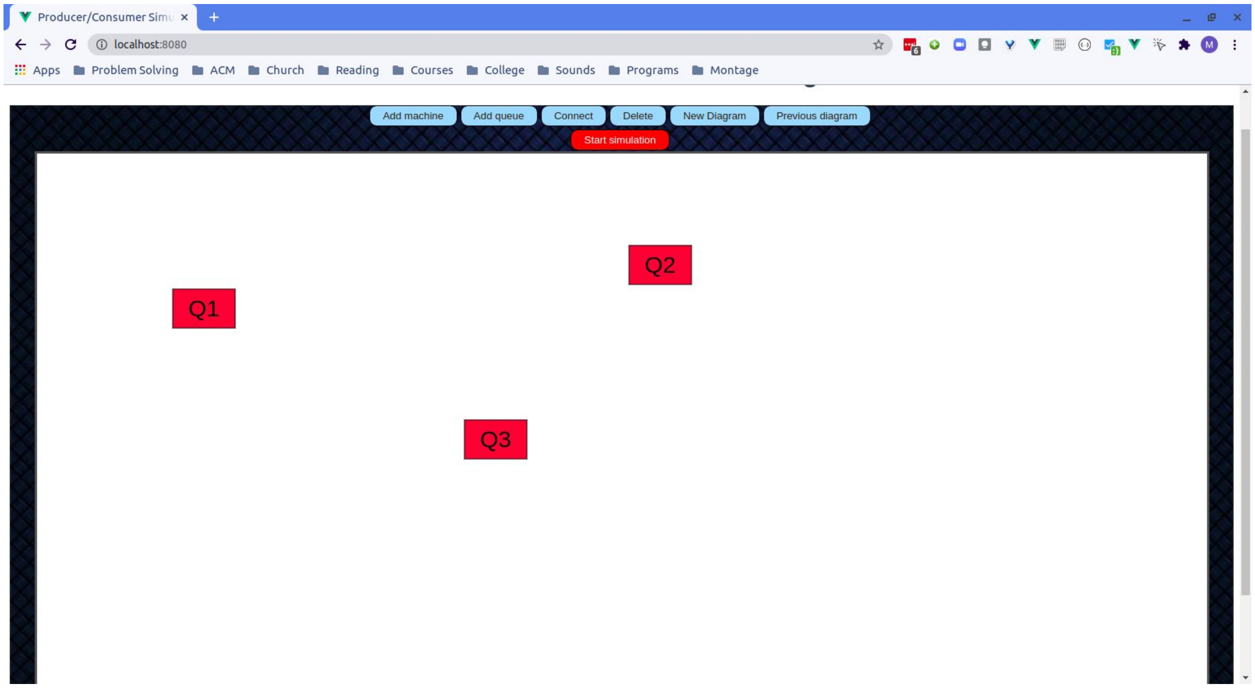
- To delete an item in your diagram
  - Click on delete button

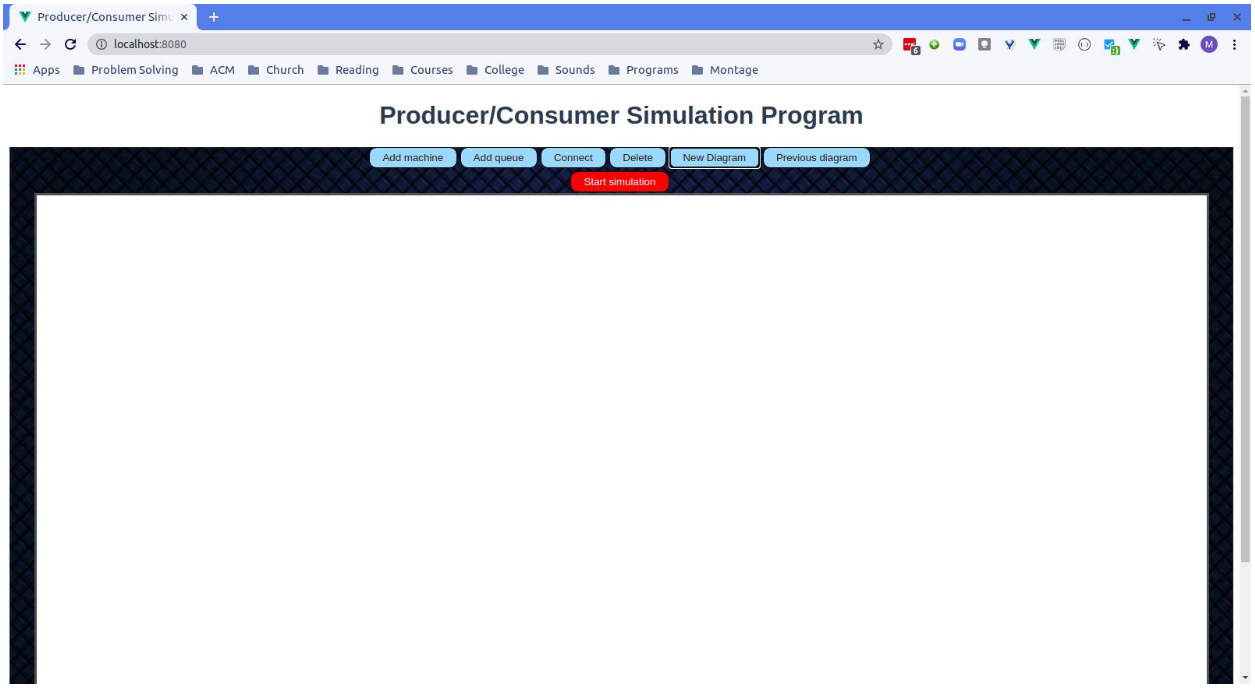


  - Click on the center of the item you want to be deleted

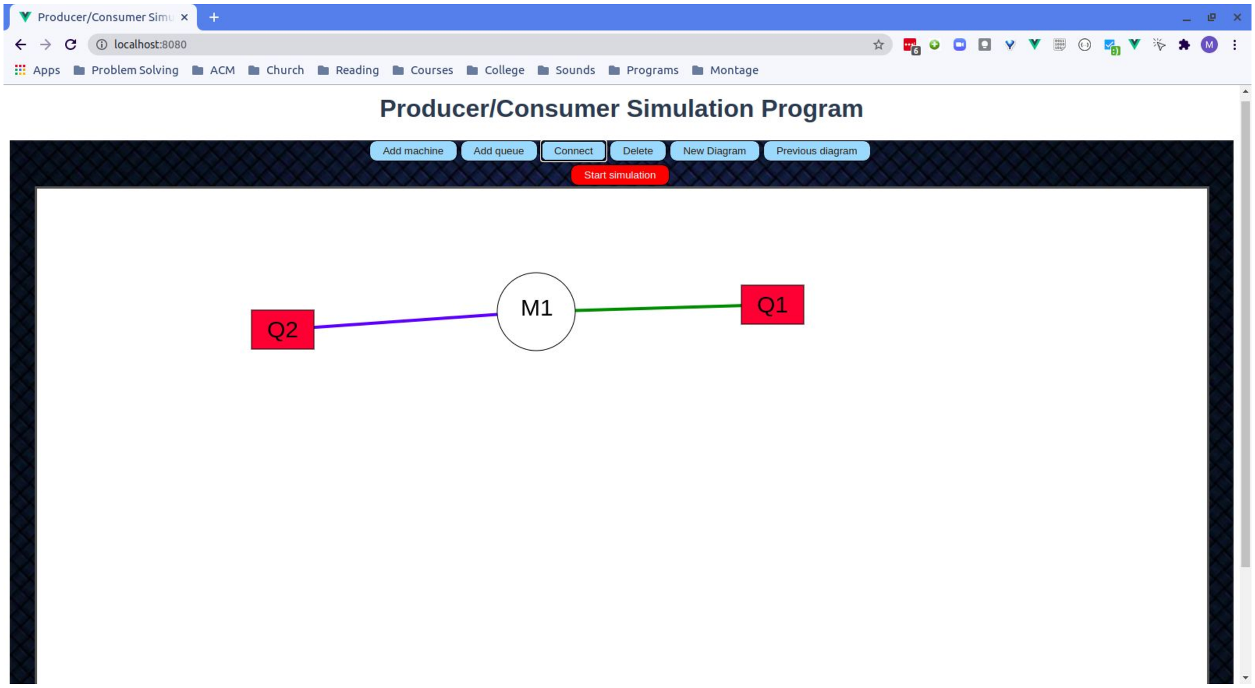


- To create a new diagram
  - Click on new diagram button

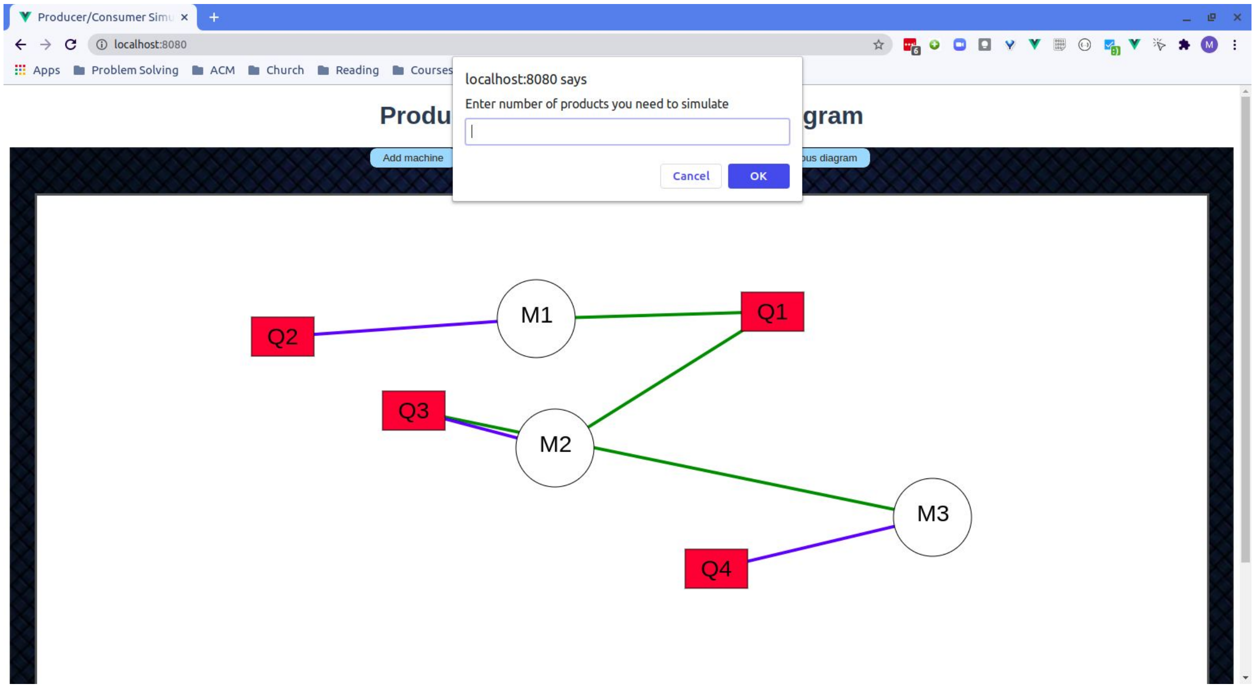




- To start simulation of your diagram
  - Click start simulation button



  - Enter the number of products you need to be simulated

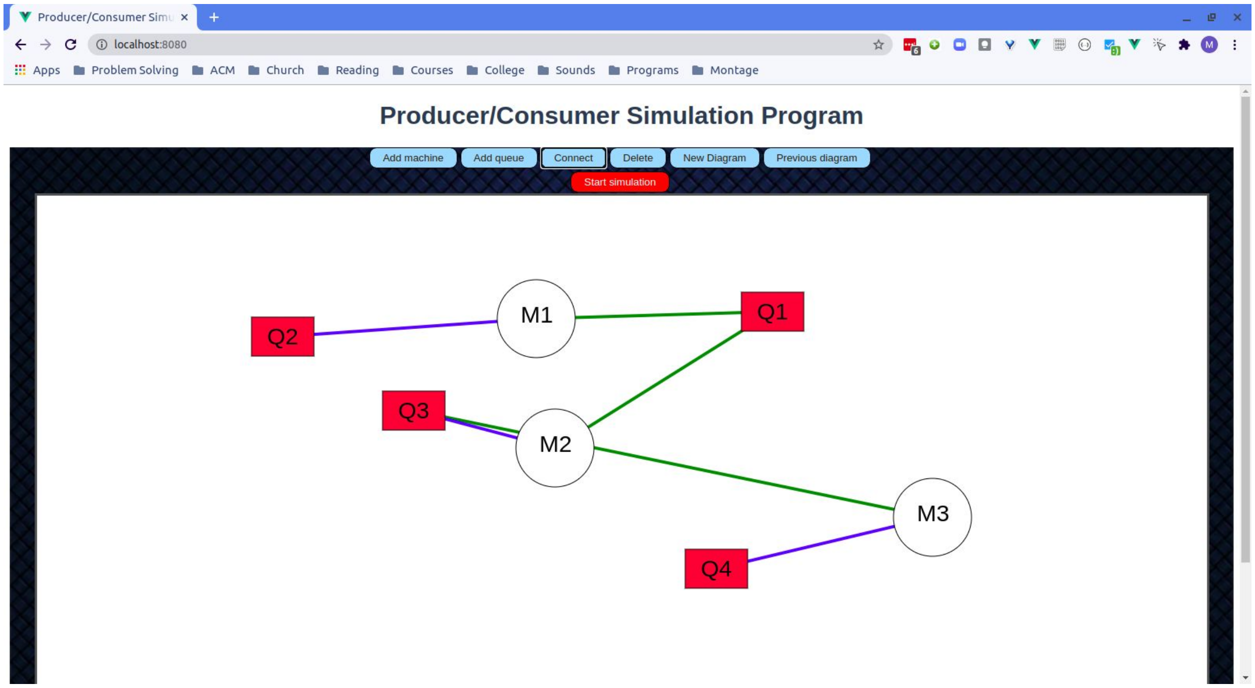

localhost:8080 says

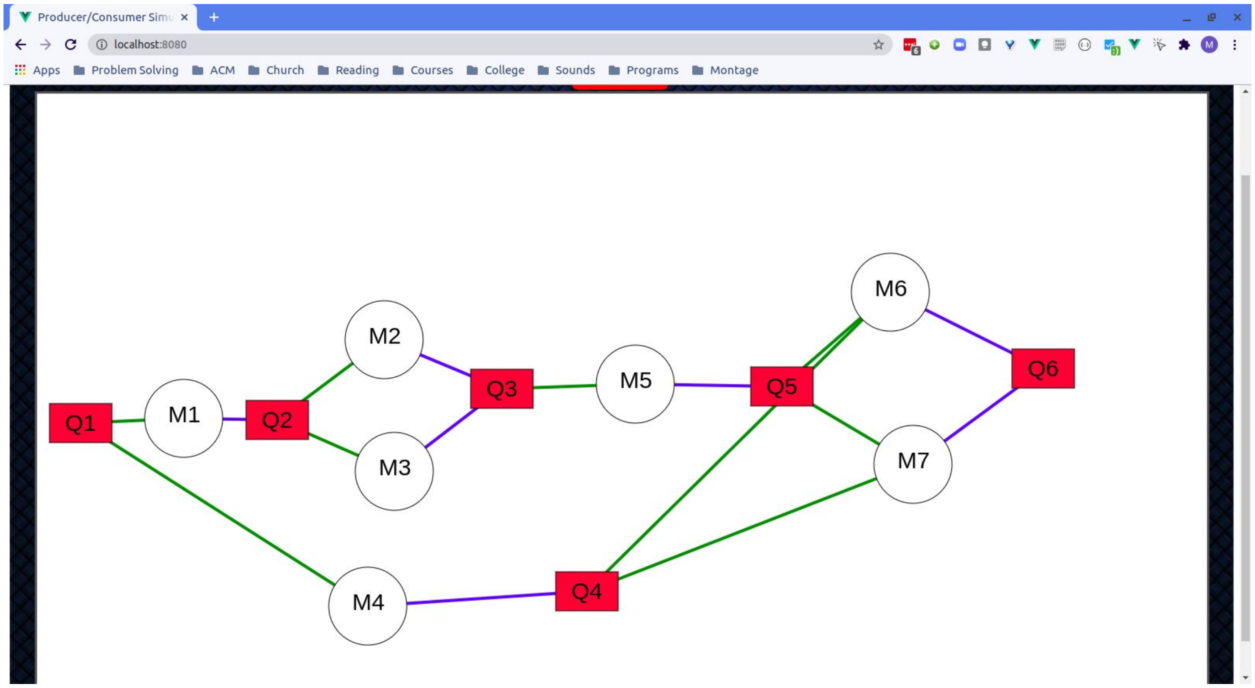
Enter number of products you need to simulate

  - The system then starts to simulate until all the products are executed in machines

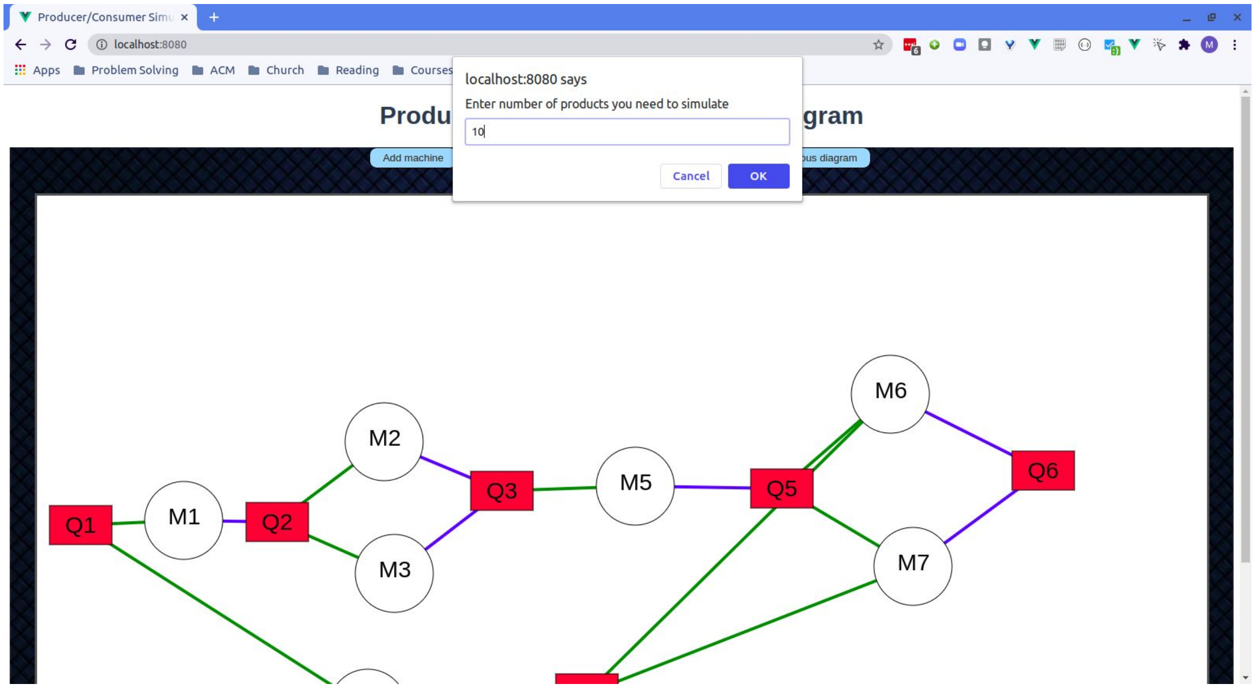
Sample Runs

- More Complex Diagrams examples





- Let’s simulate the previous diagram with 10 products

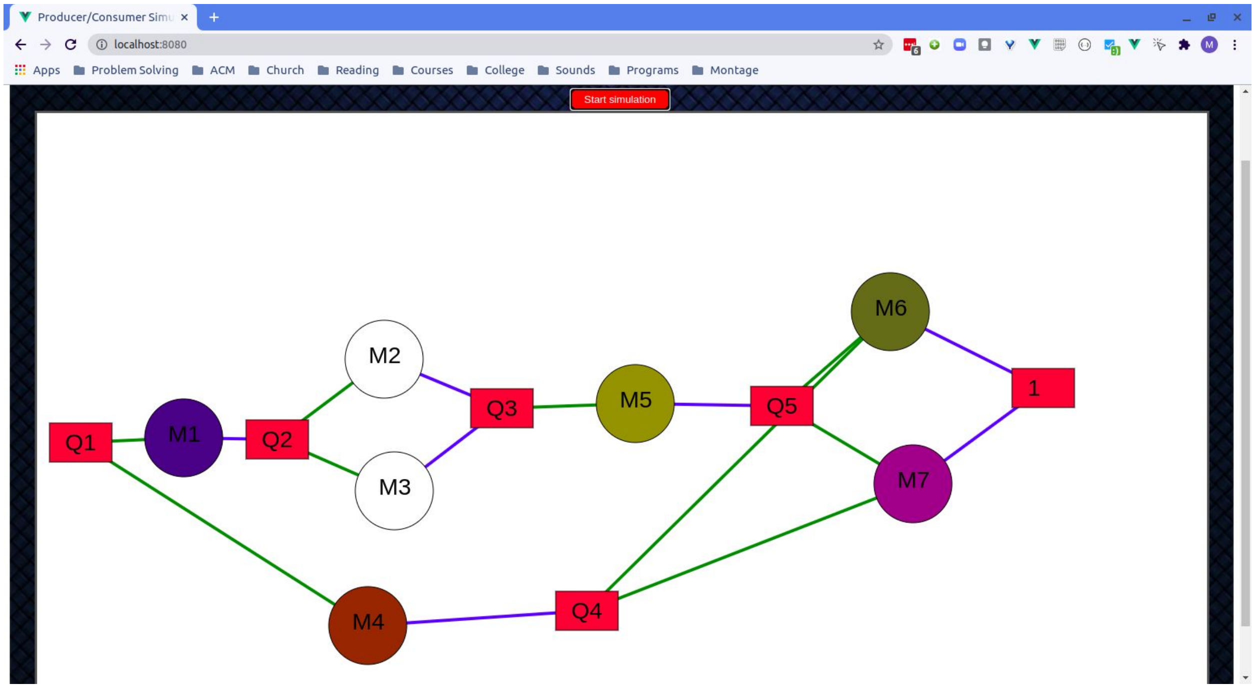


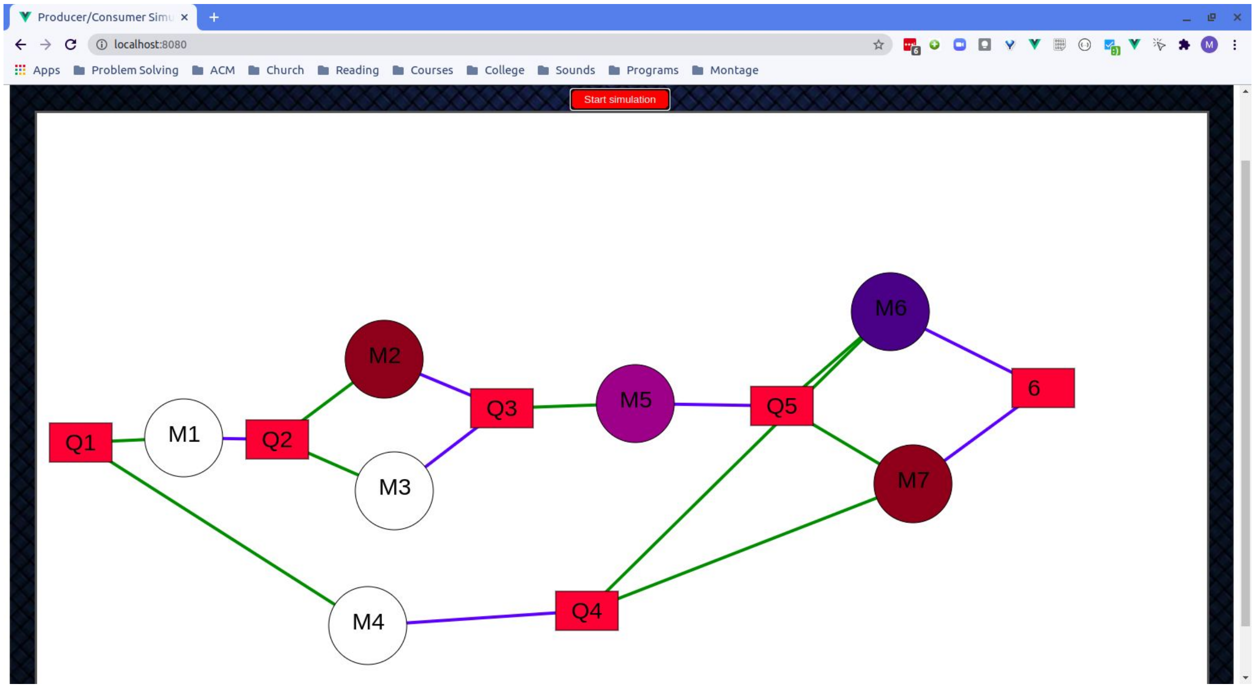
localhost:8080 says

Enter number of products you need to simulate

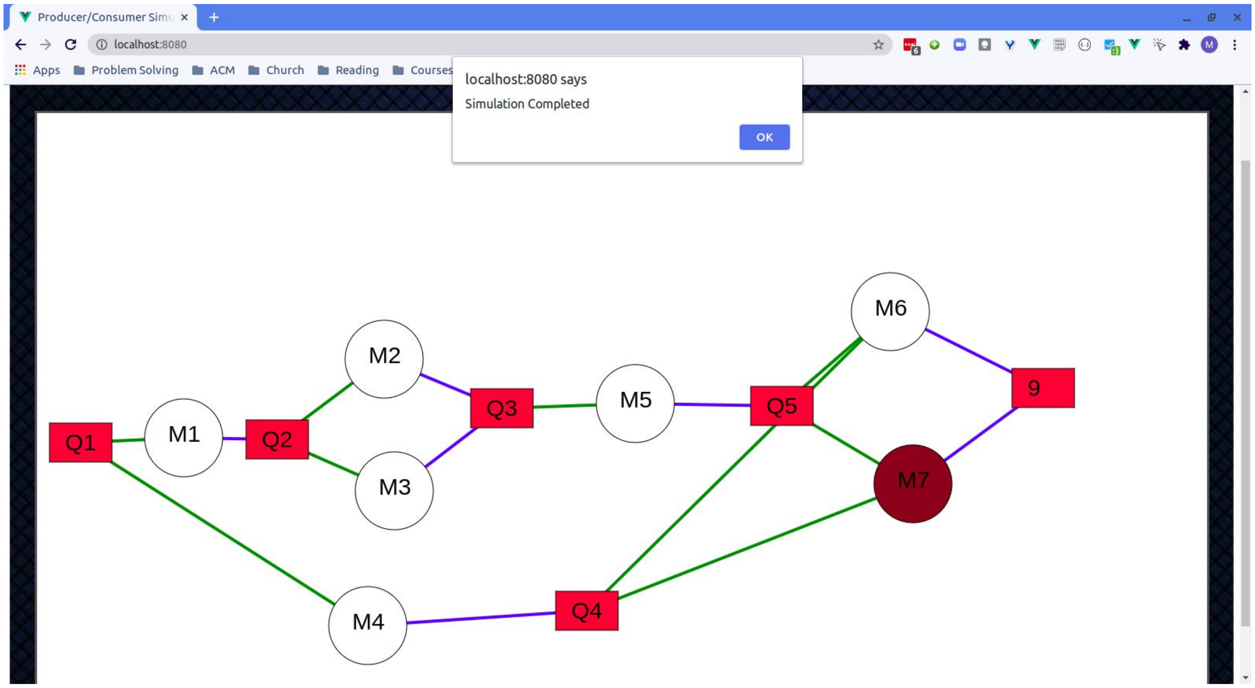
10

- Some screenshots during the simulation



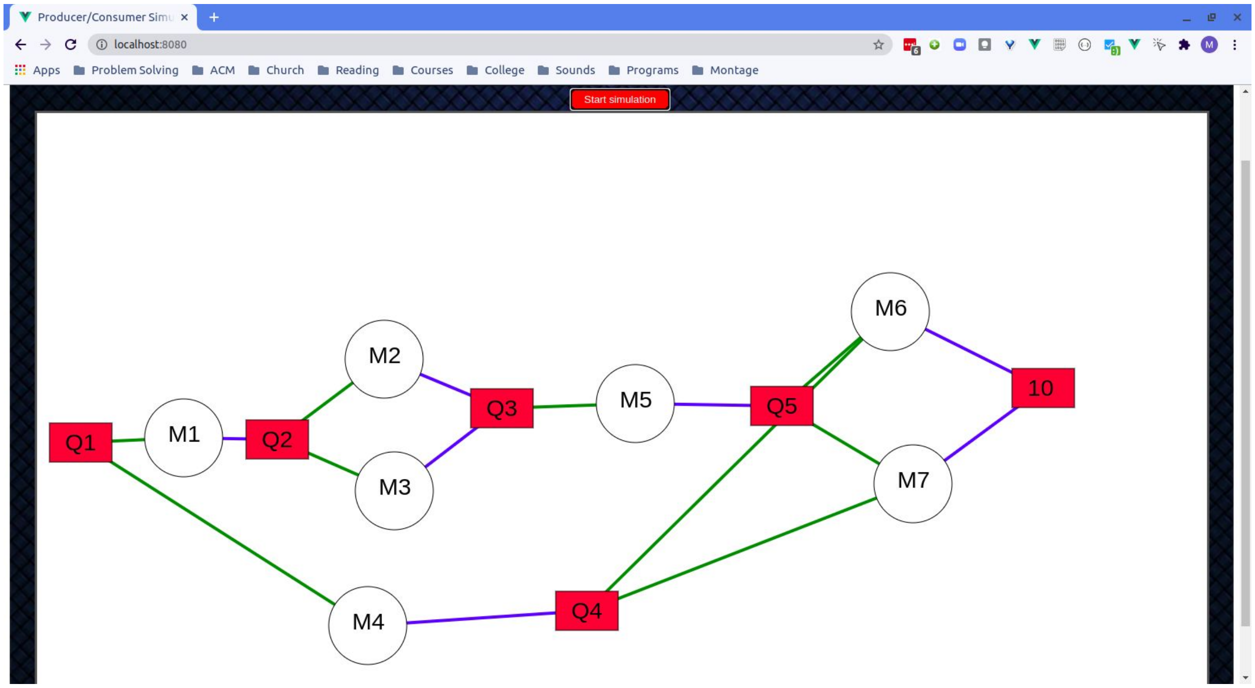


- Simulation is over



localhost:8080 says

Simulation Completed



Design Decisions

- The Starting point (Queue) is the first queue drawn on the canvas labeled by (Q1)
- The Ending point (Queue) is the last queue drawn on the canvas labeled by the maximum index in the queues sequence.
- Lines with green color are the lines connecting the queues to the machine (making queue input of a machine)
- Lines with purple color are the lines connecting the machine to the queue (making queue output of a machine)
- Machine can have multiple input queues but it only has one output queue.
- Machine serving time is generated randomly for each machine.
- Product colour is generated randomly for each product.
- Queue is represented by a label (Q) and a number if it is an empty queue but during simulation if it contains products the label will be changed to a number indicating the number of products inside this queue.