

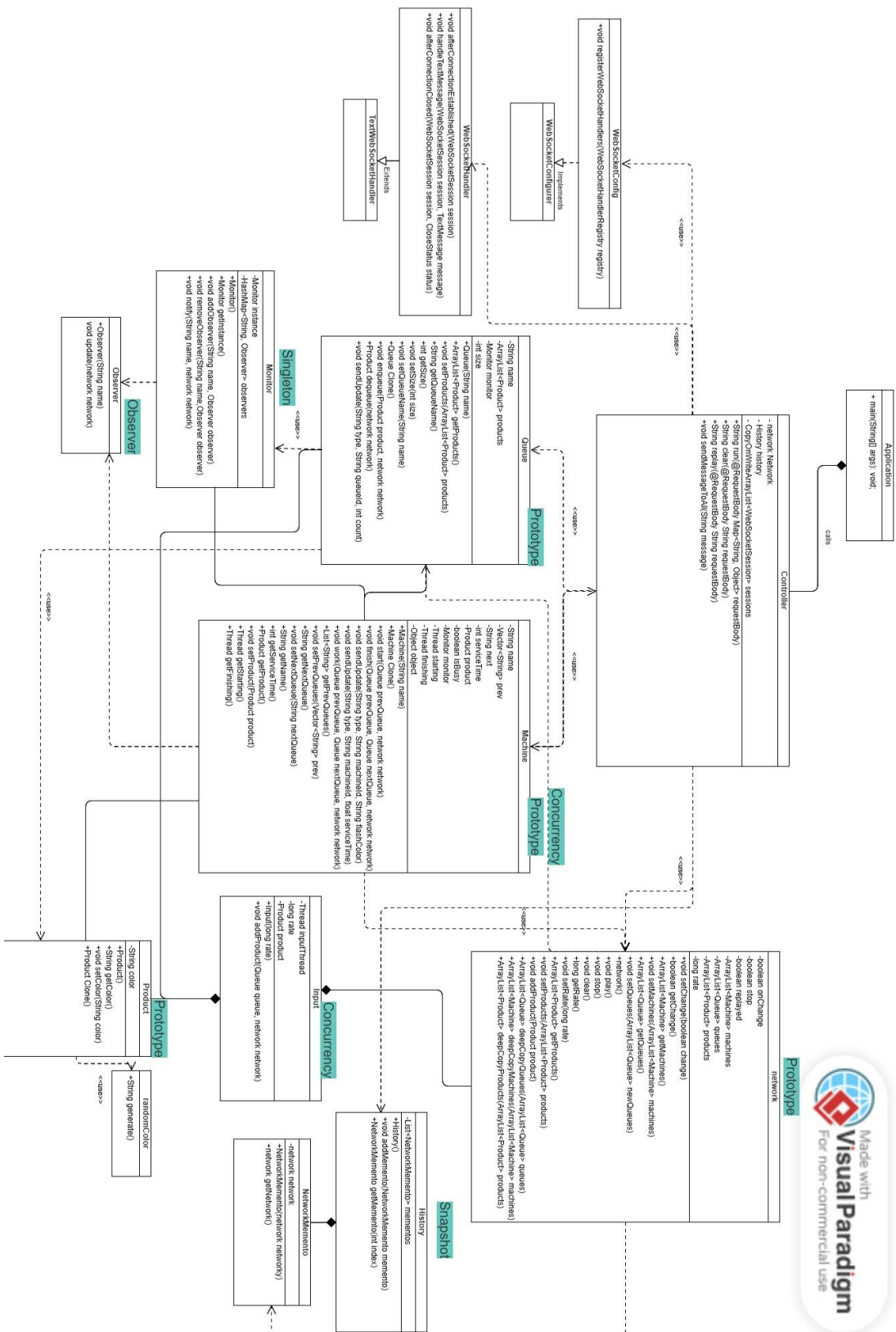
❖ Names and IDs:

- Badr Elsayed - **22010664**
- Adham Anas - **22010601**
- Nour Khaled - **22011319**
- Ali El-Deen Maher – **22010934**

❖ Steps required to run code:

1. Backend:
 - Open the Backend folder using IntelliJ IDE or any other IDE, run the Application.java class.
2. Frontend:
 - Open the Frontend folder using visual studio IDE, then open the terminal of the IDE, and write “npm install” in the terminal.
 - Then write “npm run dev” in the terminal to open the project, usually on port “http://localhost:5173/”
3. Then you can use the application.

Made with
Visual Paradigm
For non-commercial use



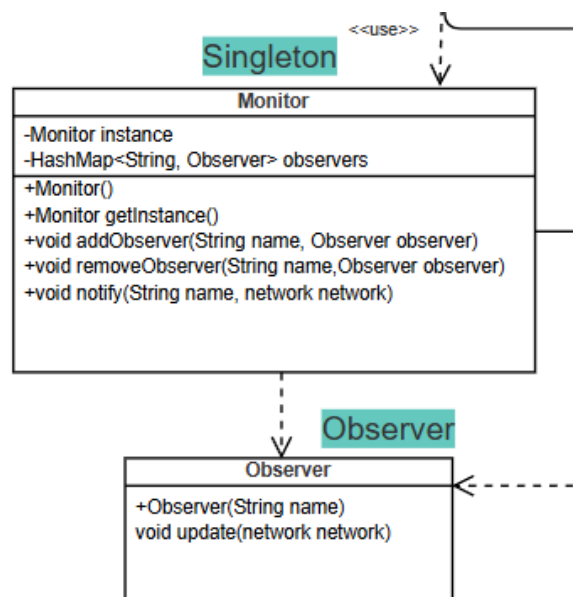
❖ Design Patterns:

1. Singleton design pattern

- This design pattern ensures that there is only one instance of the **Monitor** class throughout the application. This central **Monitor** instance manages the **observers**. Singleton helps conserve memory and enables reusability.

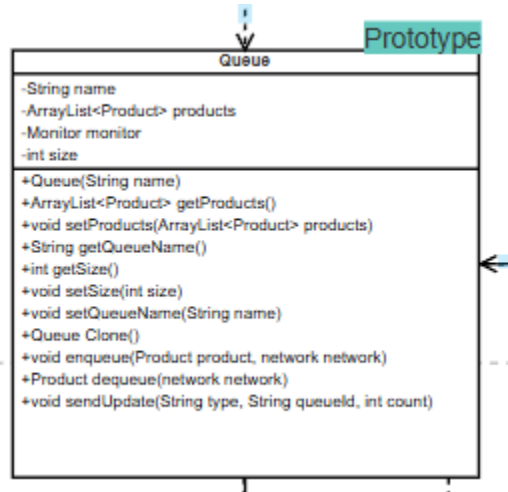
2. Observer design pattern

- This design pattern was used to implement a system where objects (**observers**) are notified of changes in the **network**. The **Monitor** class maintains a list of **observers**, and when the state of the **network** changes, all registered **observers** are updated accordingly.



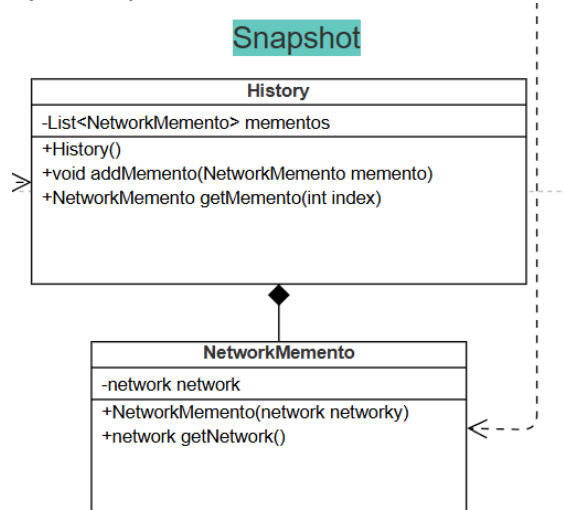
3. Prototype design pattern

- This design pattern enables the cloning of **Machine, Queue, Product and Network** objects. It help maximize the efficiency of object creation by cloning from an existing template.



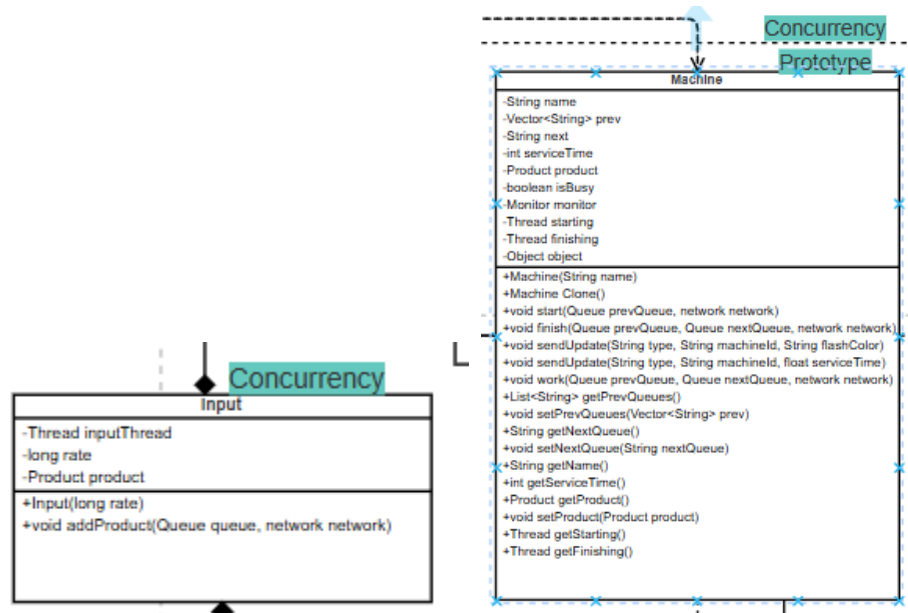
4. Snapshot design pattern

- This design pattern was implemented in the **NetworkMemento** and **History** classes. It allows capturing and restoring the state of the **network** at specific points in time.



5. Concurrency design pattern

- This design pattern was implemented in **Machine** and **Input** to handle parallel processing. The classes uses concurrency by using threads to manage and perform tasks asynchronously, allowing multiple machines to operate simultaneously.

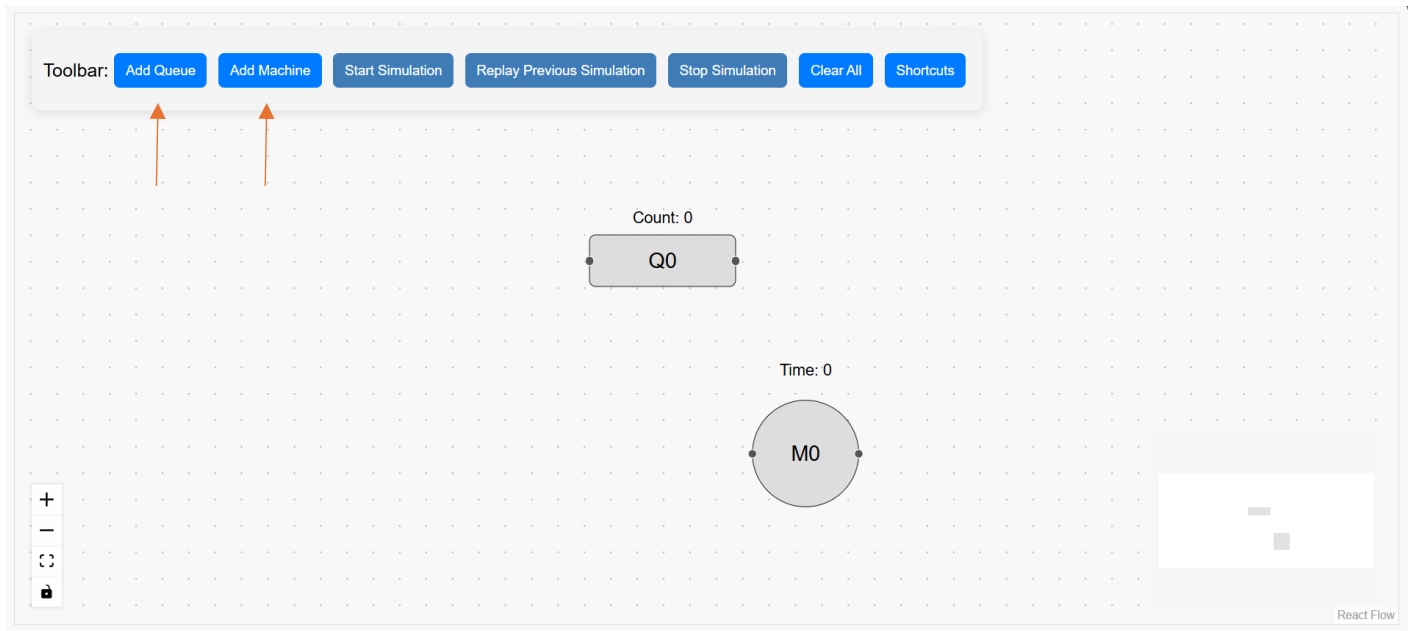


❖ Design decisions:

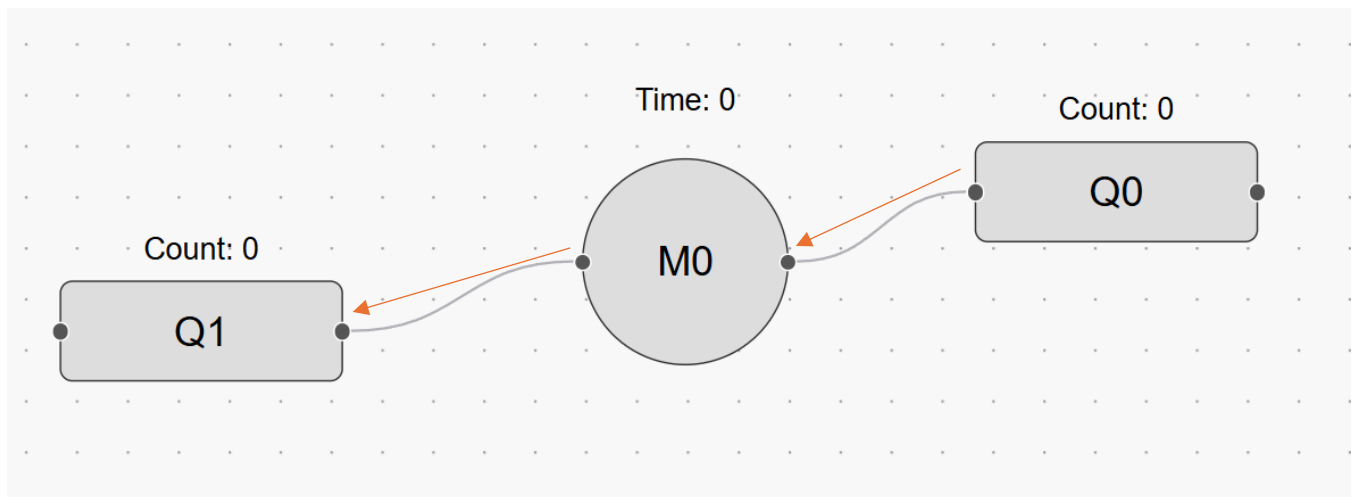
- Input (First) Queue must be Q0
- User can stop the simulation completely but can't pause/resume it.
- A machine can only output in a single Queue.

❖ Snapshots UI and User guide:

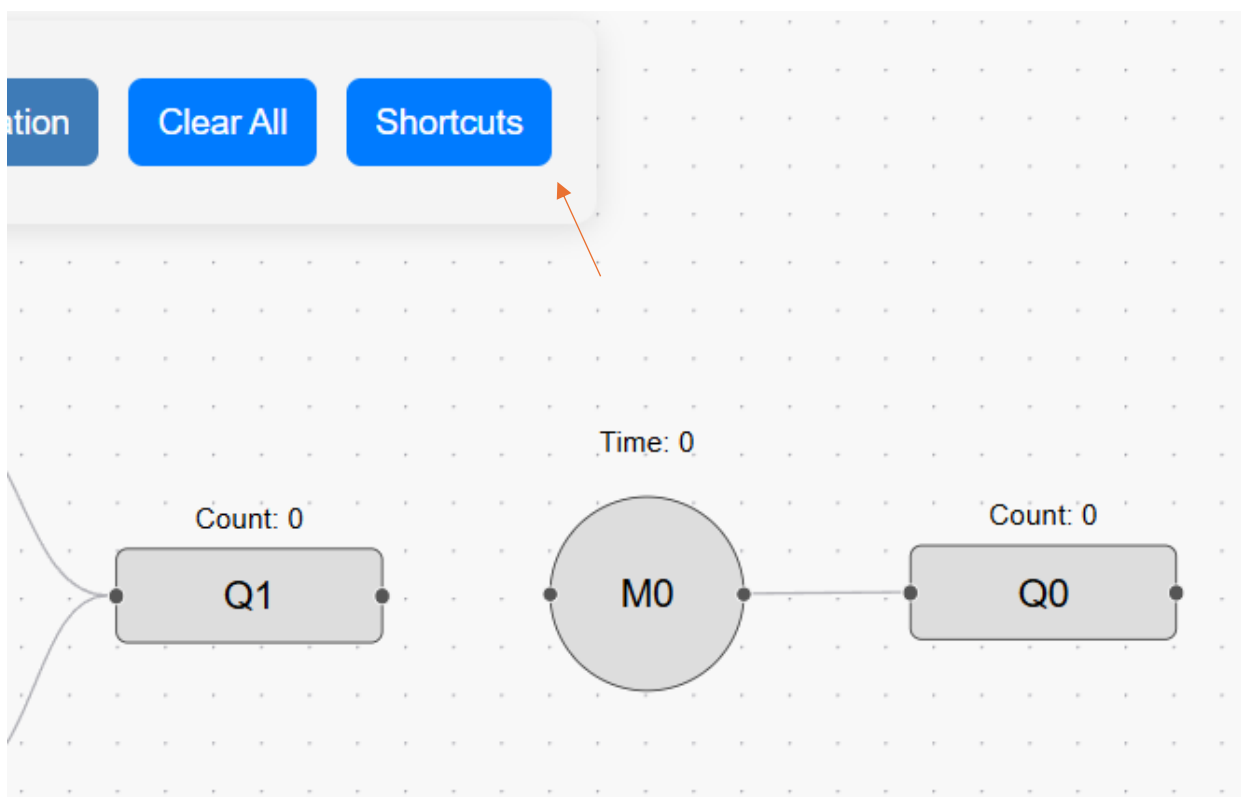
1. Start by **adding** the Queues and Machines that you want, when you click an “Add” button, a Queue/Machine will appear on the board.



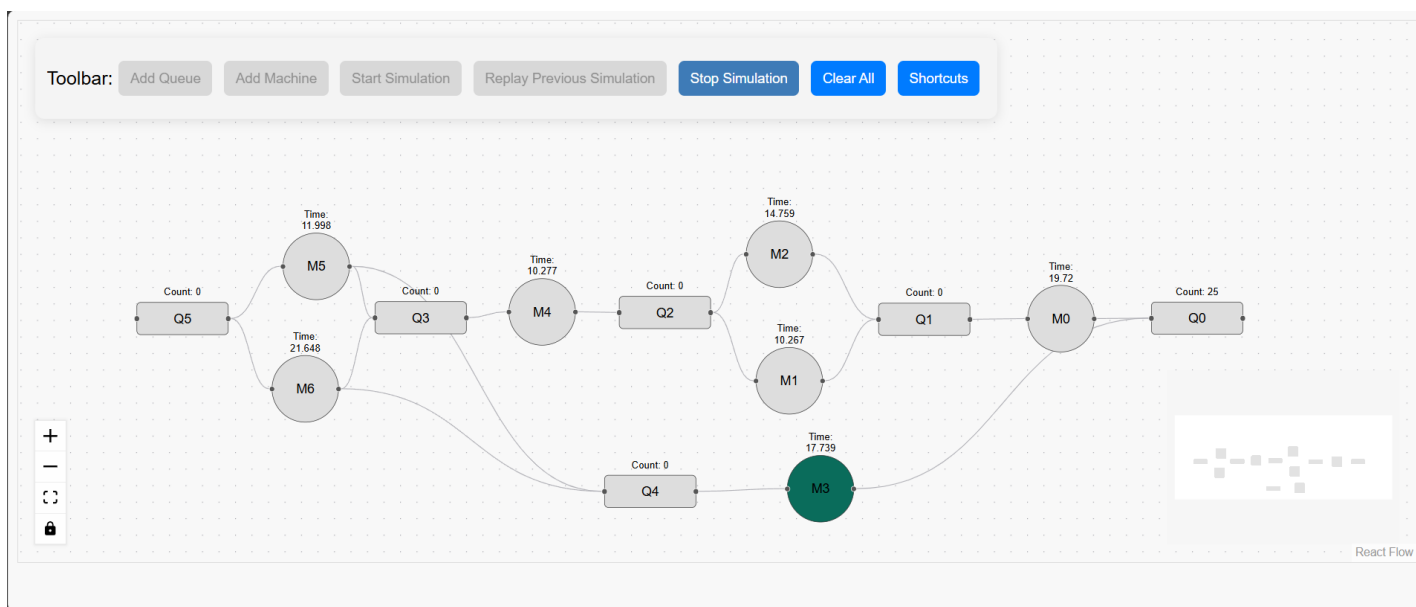
2. Rearrange the Queues and Machines by dragging them, then connect them by grabbing and dragging one end to the other.
Note: The flow of simulation is from right to left, meaning that Q0 is the input and Q1 is the final output, so keep that in mind!



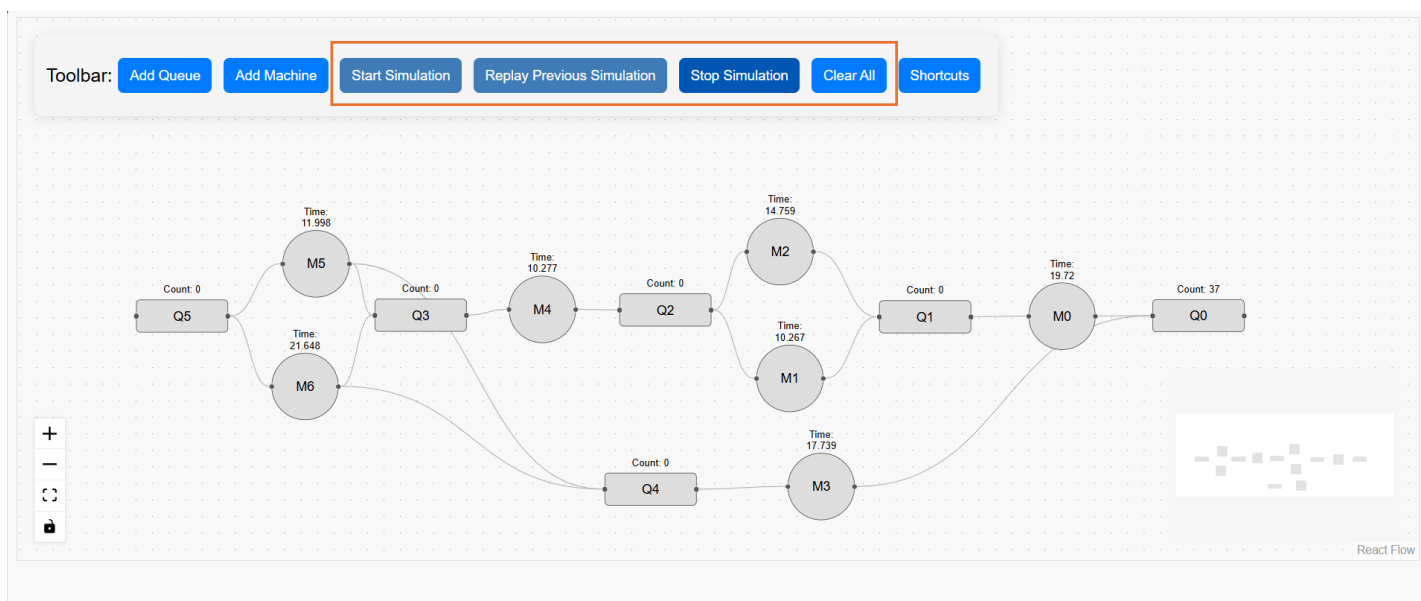
3. Made a mistake? Delete a Queue / Machine or an edge by selecting it and clicking **"Backspace"** on the keyboard, you can also see this info in the **"Shortcuts"** button in case you forgot what to press.



4. When all good, press “**Start Simulation**” to begin. You’ll see the current number of products in each Queue and the time a Machine takes to service/process a product.
- Each machine will flash the color of the product it’s inside when it finishes servicing the product,
- Each product from Q0 up till Q5 will have a unique random color.



5. Press **“Stop Simulation”** to stop the simulation.
- Press **“Clear All”** to start fresh (will obviously stop simulation).
- You can start simulation again with new random values and colors when you click **“Start Simulation”**.
- Or click **“Replay Previous Simulation”** to replay the last played simulation with the same values and colors!



6. Lastly, you may have wondered what's in the bottom left.

It's a control panel! You can

- 1- Zoom in
- 2- Zoom out
- 3- Fit view
- 4- Toggle interactivity, which includes (Select, Delete, Drag)

The bottom right is just a mini-map (overview) of the whole board.

