Use TypeScript, C#, Java, or any other object oriented programming language of your choice to complete the following tasks

A. Build Queuing classes

a. Write 2 classes to implement the following `IQueuable` interface

```
Interface IQueuable {
//adds value to queue and returns new queue
enqueue(value: string): string[];

//removes item from queue, and returns the item removed
dequeue(): string;

//returns a list of all the items in the queue
getQueue(): string[];

//returns the number of items in the queue
size():number;
}
```

Build your queues ontop of arrays; call your classes `FIFOQUEUE` and `LIFOQUEUE` (first-in first-out & last-in first-out) queues, or `QUEUE` and `STACK`, whichever names you prefer.

b. If you've written your enqueue and dequeue using array methods, how would you rewrite the same functions without using any array methods?

c. Show how you would improve the design of your classes, apply various design patterns and techniques as you see fit

B. Build a simple queue ticketing application

Design a ticket counter website with the following specifications with two separate views, one for a counter manager and the other for customers. **See image below**

a. The customer View
   i. Take a Number: Allows the customer to take a ticket. When the button is clicked, the system generates a ticket number and displays to the customer
   ii. Now Serving: Shows the latest number to be removed from the waiting queue and added to a counter
   iii. Last Number: Shows the latest ticket number to be issued
   iv. Counters:
      1. The green dots: shows serving status – green is counter is online but not currently serving any customer, red if counter is online and currently serving a customer
      2. Current number: shows the number of the current ticket being served. If the counter is offline, the value of <cur_num> will be “Offline”, the status color changed to grey, and the whole counter will be greyed out (disabled)

b. Counter Management
   i. “Go Offline”: This button shows “Go Offline” if the counter is online, and shows “Go Online” if the counter is offline. Toggles counter status when clicked. When offline, counter status on customer view will be grey, and the counter will be disabled in the customer view only
   ii. “Complete Current”: Marks the current ticket being served as complete. When clicked, the current counter status becomes available, and the status on the Customer View will turn to green
   iii. Call Next: Will
      1. Pick up the next first-in ticket from the waiting queue,
      2. Update <cur_num> in the customer view to the same ticket number (on the corresponding counter)
      3. Change the counter status (customer view) to red
      4. Display a message “No tickets in the waiting queue” if there are no more tickets to serve.

Note:

1. Build a backend application to manage the queues, so that both views can be run simultaneously on separate browsers
2. Do NOT bother with login or authentication. Assume only one user
3. Host the application using any free hosting option for demo purposes
4. Host your source code on any public repository for easy sharing.

Customer View

Now Serving: <latest_serving_num>

Last Number: <last_issued_num>

Take a Number

counter 1	counter 2	counter 3	counter 4
<cur_num>	<cur_num>	<cur_num>	<cur_num>

Counter Management

counter 1	counter 2	counter 3	counter 4
Go offline	Go offline	Go offline	Go offline
Comp curr	comp curr	comp curr	comp curr
Call Next	Call Next	Call Next	Call Next