Question:

1.Express: Find Recipe Step

The request to the route `/recipes/step/:id?elapsedTime` returns the index of the current step of the recipe based on the elapsed time in the format `{"index: currentIndex}`.
If the ID passed in the URL is missing or is not a valid number, the server should respond with the status code 400 and the string message `NOT_FOUND` in the body of the response. If the elapsedTime query parameter is not present in the URL, it should default to 0.
Note: Elapsed time should be interpreted as a Number, and the unit of measurement is the same as the unit of measurement for the timers (i.e., minutes).

Each Recipe object has a list of steps, stored in the steps property, and the corresponding timers, stored in the timers property. The steps and timers are two different arrays, but they are interpreted in combination with each other. The time required to finish step 1 (index 0) of the steps array is stored in the 0th index of the timer array.


```
{
  "id": 2,
  "name": "Roasted Asparagus",
  "steps": [
    "Preheat oven to 425°F.",
    "Cut off the woody bottom part of the asparagus spears and discard.",
    "With a vegetable peeler, peel off the skin on the bottom 2-3 inches of the
spears (this keeps the asparagus from being all and if you eat asparagus you know
what I mean by that).",
    "Place asparagus on foil-lined baking sheet and drizzle with olive oil.",
    "Sprinkle with salt.",
    "With your hands, roll the asparagus around until they are evenly coated with oil
and salt.",
    "Roast for 10-15 minutes, depending on the thickness of your stalks and how
tender you like them.",
    "They should be tender when pierced with the tip of a knife.",
    "The tips of the spears will get very brown but watch them to prevent burning.",
    "They are great plain, but sometimes I serve them with a light vinaigrette if we
need something acidic to balance out our meal."
  ],
  "timers": [
    0,
    0,
    0,
    0,
    0,
    0,
    10,
    0,
    0,
    0
```

```
  ]
},
```

Routes

- `/recipes/step/:id?elapsedTime` - The id parameter should be used to filter the results based on the ID of the object provided. If the ID is not valid or is not a valid number, the server should send a status code 400 with `NOT_FOUND` in the body of the response.

Examples

- `/recipes/step/4?elapsedTime=11`

```
Outptut - {index: 0}
```

Explanation

The ID of the recipe in the URL is 4. The recipe with ID 4 has the following properties:

```
{
    "id": 4,
    "name": "Big Night Pizza",
    "steps": [
      "Add hot water to yeast in a large bowl and let sit for 15 minutes.",
      "Mix in oil, sugar, salt, and flour and let sit for 1 hour.",
      ...
    ],
    "timers": [
      15,
      60,
      ...
    ]
}
```

The elapsed time sent as query param is 11. This roughly translates to, "Which step will be active for this recipe when 11 minutes have passed after starting the recipe?"

The step at index 0 has a timer property of 15. So after 11 minutes, step 1 will be the current step, and its index is 0, which is the final output.

- `/recipes/step/wqw`

```
Status Code - 400
Body - NOT_FOUND
```

Project Specifications

Read-Only Paths

- test
- bin
- recipes.json

Commands

- run: `npm start`
- install: `npm install`
- test: `npm test`

1.

2. Node JS Order Processing

You have been given the task to write an Order-Processing Script that takes the order data as input, performs a few pre-order checks/validations, and returns the result. If any line-item in the order data does not pass the pre-order checks, the order should fail, and the corresponding failure event should be fired.

Class Definition

Create a module, processor.js, and implement the following set of functionalities:

- The file should export a class OrderProcessor as the default export of the module, and it should inherit from the EventEmitter class.

Methods:

- `placeOrder`: Accepts the order data as an argument and immediately starts performing the validations on it. The orderData passed to the function should have the following properties/shape:

```
[
 {
    "orderNumber": "OD2323", // Order Number of the order
    "lineItems": [ // Array containing the lineItems in the order
        {
            "itemId": 3, // ID of the line Item.
            "quantity": 4 // The quantity requested in the order
        },
```

```
        {
            "itemId": 5,
            "quantity": 4
        }
    ]
 }
]
```

Events:

The class should emit the following events

- `PROCESSING_STARTED`: Should be fired just before the validations are started for an order. The order number of the current order should be passed to the callback of the event handler.
- `PROCESSING_FAILED`: Should be fired if, for any reason, the pre-order checks do not pass. The callback of the event handler will accept an object containing the following properties:
    - `orderNumber`: The orderNumber of the order that failed to process.
    - `itemId`: The ID of the first matching item that failed to process.
    - `reason`: The reason why the processing failed. It can be one of `INSUFFICIENT_STOCK` or `LINEITEMS_EMPTY`.
        - If the lineItems property is a blank array, the check should fail with the reason `LINEITEMS_EMPTY`. The itemId property need not be passed to the errorObject in this scenario.
        - If any of the lineItems' requested quantity is more than the matching stock property in the stockData array, the check should fail with the reason `INSUFFICIENT_STOCK`.
- `PROCESSING_SUCCESS`: Should be fired if all the pre-order checks pass. The order number of the current order should be passed to the callback of the event handler.

The file containing the current stock information, stock-list.json, is added to the root of the project and can be used for validation. An explanation of an entry in the JSON array is as follows:

- `id`: The ID of the item for which stock information is stored.
- `stock`: The current stock count of the item.

```
[
  {
```

```
    "id": 0,
    "stock": 4
  },
  {
    "id": 1,
    "stock": 12
  },
...
]
```

Note: It can be assumed that lineitems passed to the placeOrder function will always be an array, and the ID of the lineItem passed will always be present in the stock list. No extra code is required for these conditions.

Example Implementation

Please open the file `index.js` for example implementation.

Project Specifications

Read-Only Paths

- test

Commands

- run: `npm start`
- install: `npm install`
- test: `npm test`