Take home test

Background: sports event timing

The task is about a fictional sports event, that needs an automatic timing system for the finish corridor and finish line.

There are 2 automatic digital timing points/locations in the finish corridor:

1. Entering into finish corridor
2. Crossing the finish line

From both of the timing points, the following information is sent to our server:

1. The code of the chip that is attached to the athlete/sportsmen (identifier)
2. The identifier of the timing point
3. The (wall)clock time with a precision of a fraction of second, when the athlete (the given chip) passed this timing point.

The database, that is prepared before the sports event, has a table with the following information about the athletes participating the sports event:

1. The code/identifier of the chip that is attached to this athlete
2. The start number of the athlete
3. The full name (First name, Surname) of the athlete

The task

Implement the following software:

1. Server/service, that receives the timing information in real-time. Protocol is not known at the moment - just design it yourself in the way you wish it. You can't use the real timing system for testing - create a test-client that sends some dummy data instead.
2. Web user interface, that displays in real-time in table form the athletes who have entered the finish corridor in the following way:
    1. When athlete enters the finish corridor, a corresponding row is added to the table, where the athlete's start number and name is displayed.
    2. When the athlete crosses the finish line, the finish time is added to the athlete's row.
    3. Design the UI in the way, that the athlete's who entered to the finish corridor last, would be visible to the user without any effort from the user - the older rows/records just move out of the visible area, sequentially.
    4. Demonstrate the functioning of the system with the test-client that sends the dummy data.

5. Try to design the user interface in the way, that user don't have to put any effort or do any additional moves in order to see something (for example, no need to "refresh" or scroll the page or do any other annoyances), so that he/she would understand adequately what is happening and won't get confused.

Non-mandatory additional tasks

3. Make sure that the athletes who cross the finish-line would be displayed in the correct order - for example, if athlete A enters the finish corridor before athlete B, then they are displayed in the order of entering the corridor. But if athlete B passes the athlete A in the finish corridor (i.e. the athlete B crosses the finish line before the athlete A), then adjust the displayed order accordingly. Demonstrate it with the dummy test-client.
4. Do so that the web user interface would interact with the server in real-time only, when the browser window is in foreground / active. If user brings some other application window into foreground, then the web user interface has to stop the communication with the server. I the user activates the web browser window again, the real-time communication with the server must be resumed.
5. If the browser window has been deactivated meantime and the user brings it to foreground again, then, depending on the technical solution, there might be situation where there is a "gap" in the information that has been received from the server (because the communication with the server didn't happen and the information was not sent). In that case, think / propose, how it could be handled in the user interface so, that user would understand it adequately and won't get confused.

Solution requirements

1. **Backend requirements**
    1. You can choose the language you prefer: Go, C#, Java, PHP, Python, JavaSript (Node.js)
    2. The entire UI should be rendered by the browser - the back-end should only output JSON data, front-end and backend communication should be implemented using a JSON based (REST and/or WebSocket) API.

2. *Front-end requirements*
    - *Option 1: React*
        1. *Use Redux or MobX (or similar) for state management*
        2. *For the CSS, use your preferred toolset (Sass, inline styling, CSS-modules, styled-components, etc)*
        3. *Write JavaScript using ES6*
    - *Option 2: Angular*
        1. *For the CSS, use your preferred toolset (Sass, inline styling, CSS-modules, styled-components, etc)*
        2. *JavaScript using TypeScript*
    - *Using jQuery is discouraged*

Send us the solution as a ZIP file (or git repository URL) that contains source code together with guidelines for setting it up and observing / running the test.