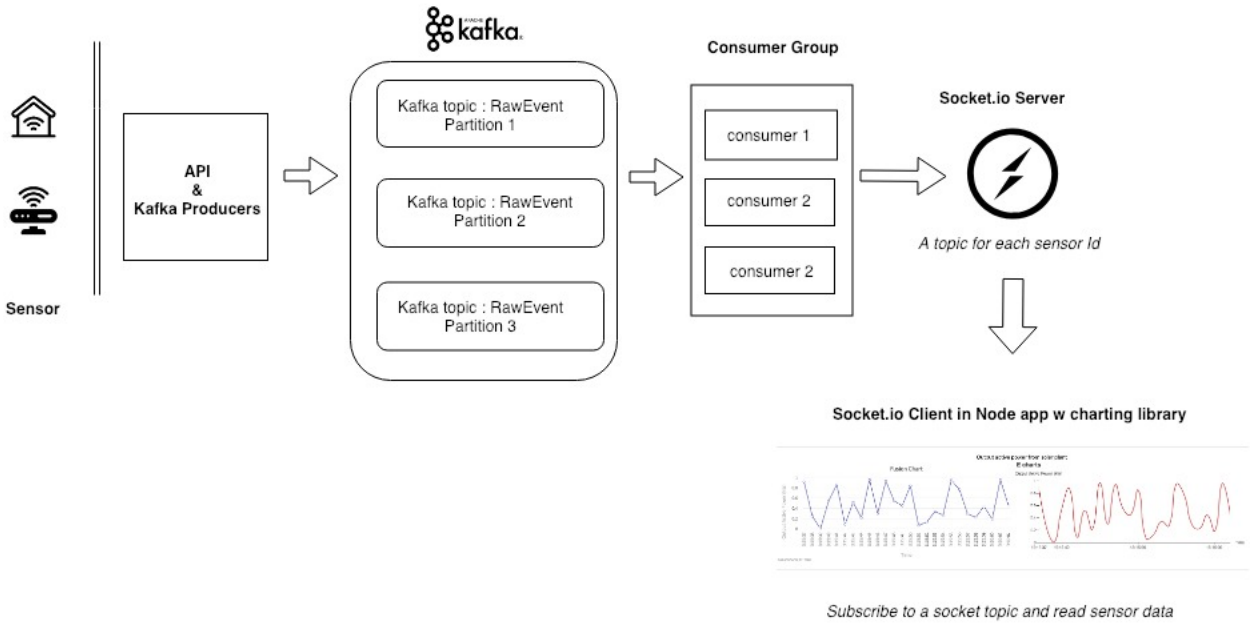
Technical Challenge Discussion (Enrico Gianoglio)

I developed the project using NodeJS and Express on the Backend and React on the Frontend. These are the technologies I am most familiar with, in particular Nodejs has a built-in feature called Stream that has been very helpful in solving the reading of the CSV document to extract data.
Stream allows us to process data chunk-by-chunk, pass it further and then forget it immediately.
To implement real-time communication between Frontend and Backend I used Socket.io, a JavaScript library that enables real-time, bidirectional and event-based communications.
Socket.io uses WebSocket protocol under the hood, to enable real-time communication.
In my proposed solution, the Nodejs backend read data using Streams from a CSV file containing all the data collected by the sensors. Then It sends data to the Client with WebSocket (Socket.io) so that the Client can get new data in real-time and display them in a table.
To persist data we could use both a NoSQL DB or SQL Databases.
NoSQL allows for high-performance and agile processing of information at massive scale.
For unstructured data like texts or geographical It is better to use NoSQL DB.
For structured data and if the data fits in spreadsheet then it is better suited for SQL-type database.
A good solution in our case is to use Realtime Databases like the Firebase Realtime Database to store streaming data. Another possible choice is to use Amazon S3 database with Amazon Kinesis Data Streaming.
For what concerns streams of data, there are numerous ways to handle it, some of them are : Apache Kafka, Confluent's Cloud, AWS Kinesis, Azure Event Hub.
In particular Kafka is a great choice for sensors streaming thanks to its real low latency and fast data transmission.
Below you can see a proposal of a possible improved solution:



1) The source of the data are sensors on tires, which send data to server via API calls.
2) These APIs act as a Kafka producer and insert data into the Kafka topics.
3) A group of Kafka consumers then reads data from the topic, filters it (optional) and sends it to a socket which act as a publisher of the data to the browser.
4) At the subscriber end, we have NodeJS and React app which subscribe to the socket topic of interest and plots event as they are received