Approach and Design

After reading the specification I decided to approach the problem developing a simple web application that can be used as a voting machine.
The application is represented by a single web page divided in two sections:

1. The first section (User View) represents the voting machine in which the user can register his/her name and vote a candidate;
2. The second section (Presenter View) shows to the television presenter the voting result. The presenter can call the "Count me up" function at any moment to display the current number of votes for each candidate and the current winner.

In order to design the application according with the specs, I extracted the following main criteria and some questions I would have asked to the client. For each question I made an assumption that would make sense in a real scenario.

Criteria:

- The algorithm should be accurate;
- The function "Count me up" should not count a vote if the same user already exceeded the maximum number of votes;
- The function should respond in less than 1 sec;
- It should run on a single machine

Questions:

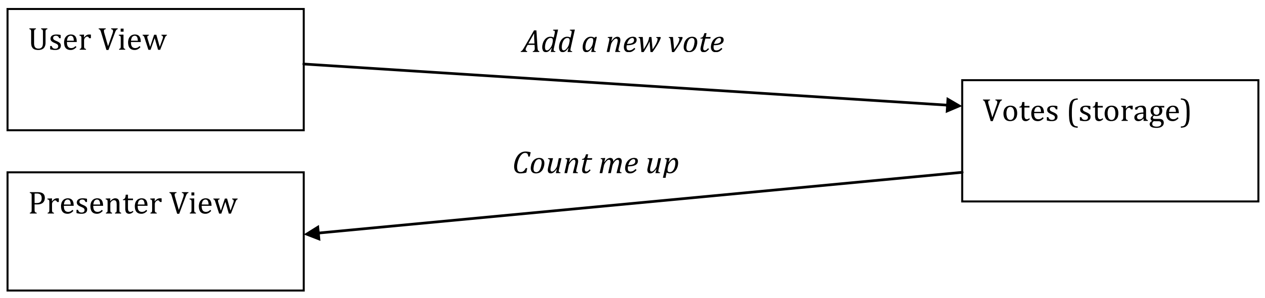
- Number of candidates? Is it fixed or can change?
- Maximum number of potential users?
- Is there a limit in the maximum number of total votes submitted to the system?
- Why the user can vote more than 3 times and exceed the maximum number allowed?
- Where the given scenario is stored? Do I need to reproduce / simulate it?

Assumptions:

- Given N number of users and M number of candidates, we assume M << N. Furthermore I decided that the number of candidates is fixed. In a real case scenario M could be equal to 10 and N could be 1 < N < 10000000
- I assumed that the total number of votes received is 10000000. Anyway I didn't put any limit on that.
- In a real scenario I would forbid the user to vote more than 3 times (for example by putting a validation rule in the interface and disable the "Vote" button after 3 submission for the same registered user). But I deliberately let the possibility to submit more than 3 votes for a single user.

I decided to approach the problem by starting from the way the votes are submitted and where and how are stored. The solution should initially work for one single vote and then with a millions of votes.

The following schema represents the structure of the voting machine:



The solution has been developed using JavaScript language and the following frameworks:

- jQuery 3.1.1
- Knockout JS 3.4.2
- Taffy DB

Knockout JS is a JavaScript framework that facilitates the implementation of the MVVM design patter. I used this pattern to create a clean separation between the interface and the business logic.
Taffy DB is a JavaScript SQL Lite DB that represents the back end storage in our implementation. In a real case I would implement some REST webservices to communicate with the database.

The storage is composed by 3 collections:

- Total number of votes submitted;
- Total number of votes submitted for each user;
- Total number of candidates and related votes.

At the beginning I made the application works for 1 single vote and I tested every function in this basic scenario. The underlying approach can be explained by the following steps:

1. The user register his/her name and makes a vote for a single candidate;
2. We increment the total amount of votes;
3. We check if the user has already voted 3 times;
4. If the check is negative we increment the total number of votes received by the selected candidate along with the real number of votes counted in the final statistic;
5. If the check is positive we just increment the total number of votes received by the candidate;
6. When the presenter clicks on the Count me up, the application will retrieve from the backend all the candidates with the total and real votes.

Basically the routine should be very quick because the real calculation is made after the vote submission and not by the "Count me up" function.
Once it was perfectly tuned for that I thought a way to simulate multiple votes submitted to the application.
A trivial solution could be a loop in which we could call the "Add Vote" function 10 million times. But this solution will slow down the browser to a crawl and can crash it.
The solution I came up to, was to simulate the vote submission from a separate Web Worker thread.

During the simulation the presenter can run the "Count me up" function and see in real-time the current voting along with the current winner.