Debate Culture

-web application-

1. Overview

The Mathias Corvinus Collegium (MCC) is developing Debate Culture, a cutting-edge application that helps to manage formal debates. This application provides features such as creating debates, managing and scheduling debates, and tracking debates.

Debate Culture is a great tool for the MCC to streamline the process of organizing and running formal debates. Formal debates are competitive discussions between two or more teams, each of which is trying to prove that their argument is the most accurate and valid. Debate Culture makes it easier to manage these debates, which can often be complex and time consuming.

2. How it works?

An admin can create a debate and provide an entry code for everyone to join. Participants can then choose their roles, whether it be judge, debater, or spectator. The judge has control over the debate, including starting, pausing, and restarting the countdown timer.

Once the debate begins, the judge will have the ability to listen to both sides and make a judgement on which team won the debate. At the end of the debate, the judge will select the winning team and close the debate.

With the help of an admin, debates can be organized quickly and easily. Participants can join, select their roles, and the judge can manage the debate until it reaches its conclusion. When the debate is finished, the judge will make a judgement and close the debate, and everyone can leave feeling satisfied with the outcome.

3. Main problems

The current approach for keeping the timer synchronized in real-time across all clients is extremely inefficient, as it requires each client to make an individual request to the server every second. This leads to an excessive number of requests, which can put a strain on the server.

A possible solution to this problem is to have the server send out the end time of the clock instead of the remaining seconds. However, this solution is not valid, since the judge can change the end time of the clock.

A better solution would be to have the client send a single request to the server, and then wait until something changes in the server's database. Once something changes, the server would respond with the new end time or remaining seconds, and the client can update the timer accordingly. This approach would significantly reduce the number of requests made to the server, while still keeping the timer synchronized.