Technical Disclosure Commons

Defensive Publications Series

December 12, 2016

Video Conference Voting System

Gabriel Diniz de Faria

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation
de Faria, Gabriel Diniz, "Video Conference Voting System", Technical Disclosure Commons, (December 12, 2016)
http://www.tdcommons.org/dpubs_series/343

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
VIDEO CONFERENCE VOTING SYSTEM

ABSTRACT

A system and method for voting via video conference are disclosed. The system is organized by meeting IDs, voting is done by joining a meeting ID to vote on a matter. The system is implemented via an application interface. When voting, each participant will move from the “Waiting” to the “Voted” section in the current task interface, without revealing what they voted for. Participants can click an eye icon to peek at what other people have voted, after voting. Once voting is done, a checkmark is clicked by a participant, which reveals the scores in bar graph style. The system calculates the winning score (in green) or conflicting scores (in red) if the votes were too far apart. The system is versatile and offers features such as change of voting scale, import and export of voting results, “moderator” and “spectator” users with specified rights, and robot login to allow monitoring.

BACKGROUND

Scoring meetings are very common when doing agile/scrum development to estimate work or to decide bug priorities, etc. Fibonacci and T-Shirt scales are commonly used for this purpose. In remote meetings, it is hard to have secret voting coming from different remote locations. Teams may vote over a videoconference by saying “one... two... three... vote!” and showing fingers to the camera with the score at the same time. Other alternatives include paper cards, a phone or other app that displays a card on screen and collects a vote, or customized software or web pages designed to address the problem of planning collectively.

DESCRIPTION

A system and method for voting via video conference are disclosed. The system is organized by meeting IDs, where everyone joining a meeting ID would vote on the same matter.
The system could be implemented via an application as shown in FIG. 1, where all participants are joining the meeting “Timemachine”.

![FIG. 1: Video conference voting system and application](image)

When voting, each participant will move from the “Waiting” to the “Voted” section in the current task interface, without revealing what they voted for. Participants can click the eye icon to peek at what other people have voted, after voting. Once voting is done, the checkmark is clicked by a single participant, and that reveals the scores in bar graph style. The system calculates the winning score (in green) or conflicting scores (in red) if the votes were too far apart.

Any time during the meeting, the voting scale can be changed. The system offers a choice of Fibonacci (3, 5, 8...), T-Shirt (M, L, XL...), Priorities (P0, P1, P2...), Yes/No, and Yes/Maybe/No scales. The system allows adding more scales, if required. The resulting scores will stay in the page up to 64 scores. After 40 seconds, they will fade to indicate they are
probably not as relevant anymore. Each score card has a timer showing how long ago that score was decided.

The system allows import items from other programs, csv files, or external system APIs via custom plugins. The import of fields can include:

- Title: One liner description of what's being voted on,
- Description: Longer description for hover over or expanding in the UI (optional), and
- Link: A link to the original resource so it can be modified or to show more details (optional).

This would create an agenda for the meeting, and a moderator can go through each item, and participants can vote ahead of time if they already have knowledge of what is being discussed. This would save additional meeting time.

When importing from external systems that provide an API, the system can just have a “search box”, where the results from the search are imported. That way an option can be given to refresh the items. Some participants may change the items in the external system as the meeting progresses, or add items to a queue that would be included in the result set. The application can then auto refresh or manually refresh the agenda, keeping the currently scored items and adding new items by importing them.

After a voting session is over, the application could export the data to Cloud, to csv files, or to external system APIs and save the data directly into other systems via custom plugins. The export can include:

- Title: One liner description of what's being voted on,
- Description: Longer description for hover over or expanding in the UI (optional),
• External System ID: ID of the resource that was voted on in an external system, and

• Result: The final result from the voting (e.g. “P1”, “8”, “Yes”).

The application could have a “moderator” workflow. The first person joining the meeting would receive the moderator crown (indicated by a ♛ next to their name). That will indicate that that is the only person who can import new issues, rename them, see other people's votes, and go to the next item. All other participants will have the option to steal the crown and become the moderator for this session in case they need to take over (moderator had to leave early, got disconnected, etc.) This would reduce the number of options for all other participants, letting them only vote and simplifying their user interface.

Participants could add context on the fly as the meeting progresses. The item being voted on gets a default title (or no title), and as it is discussed participants can add a title to it, some description, and a link to an external system for more information.

A participant can peek at other participants’ scores, but the fact they have looked at what was voted would be displayed to everyone. The current assistant (green when there's a winning score, red when there's conflict between multiple people) can be customized for the teams depending on how they pick their scores. For example, some teams pick the highest result when all votes are 1 apart in their scale. For example, for result set 2, 2, 2, 3 some teams pick 2 (most voted) while others pick 3 (highest). The customized assistant may also provide for a method to handle uncertain “?” or “infinite” votes.

We could have robot accounts join an ongoing web meeting to display the current (and maybe next) item being voted. This would help the team so nobody has to present and use up their time and screen for it. The robot account would show who has/hasn't voted and the results
of the previous item. Participants can set themselves to “Spectator.” They would then not be shown in the “Waiting” list of users, and they can see a live graph of how the voting is unfolding.