dev.agaber

Ranked Choice Voting PRD

Contributors: abrahim.gaber@gmail.com
Last Updated: Feb 2023
Self link: https://tinyurl.com/rcv-app-prd
Status: Draft | Review | Final | Obsolete

OVERVIEW

Instant-runoff voting, or ranked-choice voting as it is commonly referred to in the United States[1], is a way of counting votes in single-winner elections. CGP Grey's series of YouTube videos do a great job of explaining the problems with traditional first-past-the-post elections and how other voting algorithms can yield fairer, more democratic results.

The focus of this proposal is to create an application that allows users to create arbitrary straw polls that use an instant-runoff algorithm to determine winners. A generalized API and standalone web application are the initial goals but it is conceivable that this will find utility as an embedded utility inside other systems (i.e., Google Forms, Discord bots, etc.).

GOALS

1. Launch a REST API that let's users define an election, submit votes and determine the winner.
2. Launch a web application that allows users to perform the above functions.
3. Learn or enhance skills in Frontend, API and database design and algorithms.

SPECIFICATIONS

Strawpoll has a great interface that can be referenced as a starting point. The following are MVP requirements and all of the following specifications should be available in a UI and a public REST API.

- Submit an election: Input a text question and a list of choices.
- Share/retrieve an election: The person holding the election should be able to share it.
- Vote in an election: Voters should be able to submit numeric choices for each option:
    - Numbers should start at 1.

[1] Also commonly called Alternate Vote (AV) in the United Kingdom and preferential voting in Australia.

    - Numbers cannot be repeated.
    - Numbers have to increase sequentially for each choice (cannot skip a number).
    - Voters do not have to vote for every choice.
    - Voters do need to make at least one selection though.
    - Users can vote more than once for now.
- Determine a winner
    - Count votes using the instant run-off algorithm.
    - Return the winning choice (don't need to include how many votes or how the winner was determined in this iteration).

MILESTONES

Although these milestones are numbered sequentially, they do not need to be implemented sequentially. There is no implied priority here either.

M1: Define and implement an in-memory REST API

Choose programming languages and frameworks and check in a skeleton project that exposes a RESTful API. Define the API methods and parameters using OpenAPI. Implement something simple that stores everything in memory.

M2: Persist elections

Decide on a database product, define a schema and get it all working locally. Completion of this milestone means there is a fully operational API that persists data somewhere.

M3: Launch a Web UI

Decide on a web framework. Figure out what these words mean: pre-processors, task runners, bundlers, SCSS, LESS. Sub-milestones:

- Create new elections: Build a form that allows users to submit new elections.
- Vote on existing elections: Build a form that renders the title and choices of a defined election and lets users submit votes (numeric choice for answers).
- See results of an election. Just need a winner in this iteration.

M4: Deploy to a cloud provider using Kubernetes.

Kubernetes is an explicit goal for learning purposes (I want to learn about it). So completion of this milestone implies some level of Kubernetes mastery by the developer.

A second and probably more important goal functionally speaking is to host this someplace where other people can actually use this. For this objective some sacrifices may need to mitigate rising expenses.

M5: Carry moonbeams off in a jar

I'm actually not sure if I want to do anything with this after I'm finished experimenting what I wanted to experiment with. I have seen a couple of places where voting is kinda cool and there are a number of disparate tools to do so. After these 4 milestones the API may actually have the most utility since embedding vote functionality into other apps might be more useful to people. Some ideas:

- Live polling or races. Analogy: "Teacher: By a show of hands how many students want to learn math today? And also by a show of hands, how many students want to go outside and play today?" Someone did something like this in a zoom meeting and they were using Mentimeter . You set up an election and visually show the current status as more votes come in.
- Straw polls: Support traditional first-past-the-post elections with single or multiple choice options.
- Draft election configs.
- Ending elections either manually or at a specified datetime.
- Require users to authenticate and restrict one vote per person. Optionally let them change their vote.
- Anonymous and not anonymous voting.
- Just showing results in general. For ranked choice voting, display the breakdown of each round. Show stats for any type of election.
- Embedded UIs. Discord, Slack, Google Chat, Reddit, Facebook, Facebook Messenger.
- Dory. There is actually dory.app now undoubtedly inspired by Xooglers somewhere. Google's Dory is for asking questions in a meeting or presentation. Popular questions get upvoted like reddit. There's a lot of functionality to unpack here (not sure why Google didn't productize this) and I'm not sure if that's an appropriate solution for this.
- Government elections. Did you know Estonia has electronic "i-voting"? There's a lot of content saying why this is a bad idea. It has to be secure, anonymous but still verifiable. Some people have mentioned block-chain to avoid the security concerns. There must be a technical solution to this problem. Seems important given all the recent hand wringing about election fraud in the US and Brazil recently.