Software Design Document

OurGovernment

Names: Itai Lashover, Liav Weiss,
Amichai Kafka, Shoshana Levin

Date: (11/12/2021)

TABLE OF CONTENTS:

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of OurGovernment System.

1.2 Scope

The executive branch, or government, is typically not elected directly by the people, but rather formed by another elected body or person such as the parliament or the president. As a result, its members are not directly accountable to the people, individually or as a group.
Indeed, in coalition systems, following coalition negotiations, each party in the coalition gets a group of ministries which, in turn, have to be populated by ministers.
We consider a scenario in which the members of the government are elected directly by the people, and wish to achieve proportional representation while doing so.
The project includes a system for electing ministers in the government, a research paper that analyzing the results of the algorithm and an application that will present the algorithm.

1.3 References Material

'Electing the executive branch' wrote by Rutvik Page, Ehud Shapiro & Nimrod Talmon.

1.4 Definitions and Acronyms

GreedyPAV - The main algorithm of the system, this algorithm ensures a proportionate and fair division of ministers into offices

2. SYSTEM OVERVIEW

A system for electing ministers in the government, while implementing an innovative algorithm that ensures that the ministers represent the general public.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

Our system is divided into 3 main subsystems.
The Algorithm - GreedyPAV An algorithm.
Implementation: Python using ABCVoteing open-source library.
A website - an interactive and simple system for the user that supports two prominent functions: running the algorithm on a real government and running the algorithm for fair distribution on a variety of topics for the user to choose
Implementation: HTML, CSS & JS framework - React.
Research paper - an article to be written by us whose main purpose is to examine the results of the algorithm against the current division in the government today.
Note: The connection between the website and the algorithm will be through Django open-source web framework

3.2 Decomposition Description

The following diagram shows the decomposition of the website's subsystems in the architectural design.

3.3 Design Rationale

Our project deals mainly with the implementation of the algorithm in order to allocate offices to government ministers but we have identified the potential of the algorithm through which a fair and proportionate distribution can be obtained on a variety of similar topics (like division into committees in different organizations).
Therefore, we chose to divide the system into these 2 main components.

4. DATA DESIGN

The file that can be uploaded to the system will be in xlsx format.
The template file can be seen in the appendix.

5. COMPONENT DESIGN

Our system is divided into 3 parts (algorithm, website and research article).
In this section we will expand on the design of the algorithm because the 2 components listed in section 3.3 are built mainly on the basis of the algorithm.

Algorithm design:
GreedyPAV is used for multiwinner elections and is known to be proportional for that setting. In those settings, it works as follows: Initially, each voter has a weight of 1; the rule works in k iterations (as the task in standard multiwinner elections is to select a set of k alternatives), where in each iteration one alternative will be added to the initially-empty committee.
In particular, in each iteration, the alternative with the highest total weight from voters approving it is selected, and then the weight of all voters who approve this alternative is reduced; the reduction follows the harmonic series, so that a voter whose weight is reduced i times will have a weight of $1/(i+1)$ (e.g., initially the weight is 1; then, a voter reduced once would have a weight of $1/2$, then of $1/3$, and so on).
In the proposed adaptation of GreedyPAV to our setting of electing an executive branch, in each iteration, we again select the alternative with the highest weight from approving voters; say this is some $a_j \in A_j$. Now, we fix the jth office to be populated by a_j; then, as it is fixed, we remove all other $a_i \in A_j$ from further consideration (as the jth office is already populated) and reweight approving voters as described above (in the description of GreedyPAV for the standard setting of multiwinner elections).

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

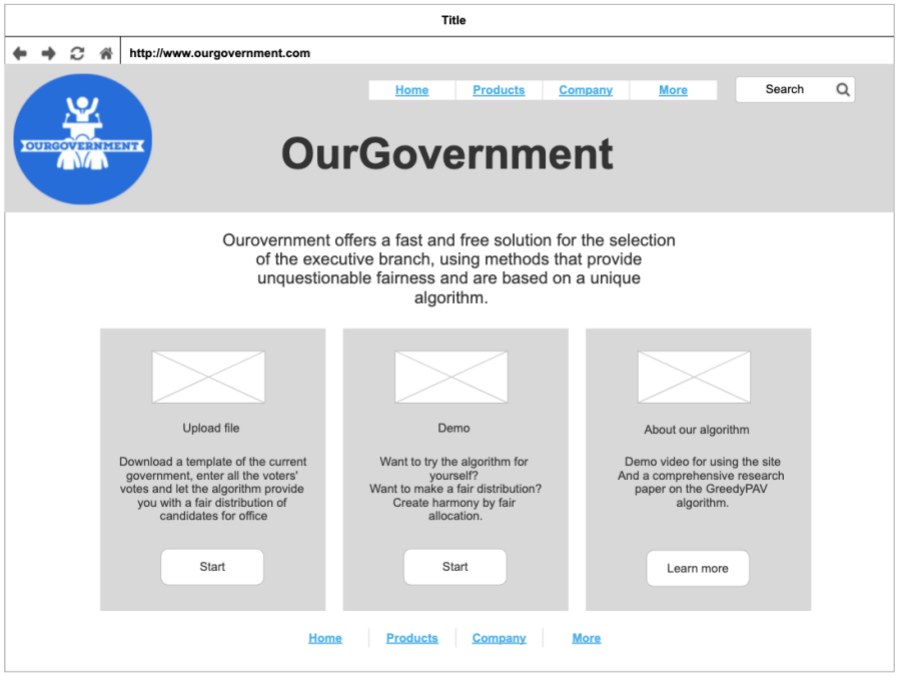
In this section we will describe the functionality of the system from the user's point of view by different scenarios for using the system:

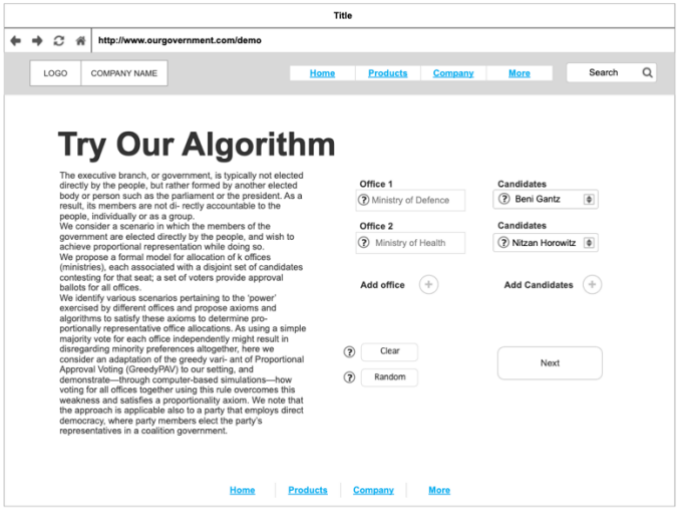
Scenario 1: On the Home screen, the user clicks ‘Upload File’, downloaded the template file and uploaded it to the system.
If the format is correct and the user clicks ‘Run the algorithm’, the algorithm will run on the server side and output the result.
During the calculation a loading bar will be displayed to the user.
The result of the algorithm will then be converted to the system interface and displayed to the user with a detailed explanation of its correctness.
In addition, the user can choose whether to download one of the files by clicking ‘Download results’ or ‘Download full statistics’.

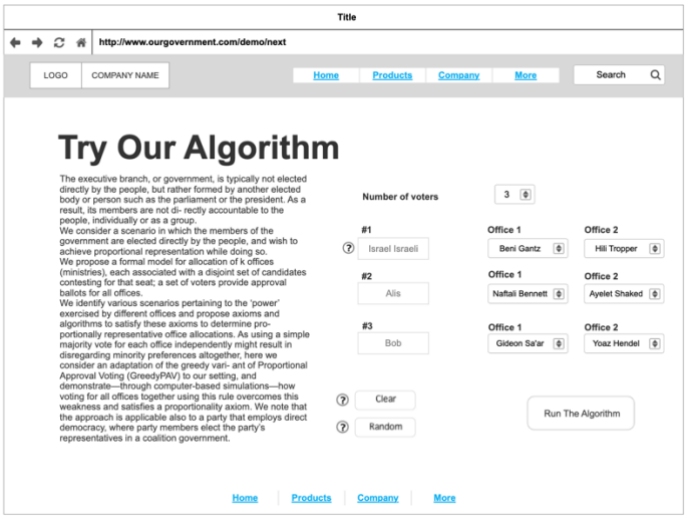
Scenario 2: On the Home screen, the user clicks ‘Demo’, a screen will open for the user where he can fill in the names of the offices and candidates the user will be able to draw a random placement of offices and candidates by clicking on 'Random' - the system will draw offices and candidates from the current government.
In addition, the user will be able to enter the names of the offices and candidates himself.
At the end, the user will click on ‘Next’, a new screen will open where he will fill in the voters' votes.
Then the user clicks ‘Run the algorithm’, the algorithm will run on the server side and output the result.
The result of the algorithm will then be converted to the system interface and displayed to the user.
In addition, the user can choose whether to download one of the files by clicking ‘Download results’ or ‘Download full statistics’.

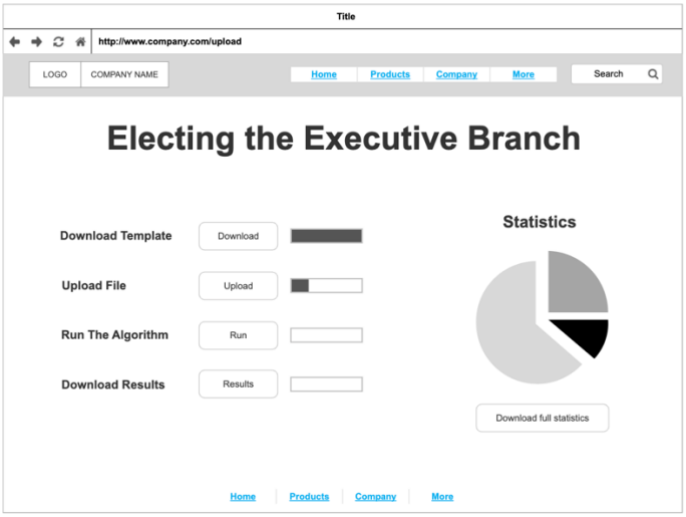
Scenario 3: On the Home screen, the user clicks ‘About the algorithm’, a screen will be displayed where he can watch a video explaining the use of the system and in addition the user will be able to read the research paper written about the algorithm.

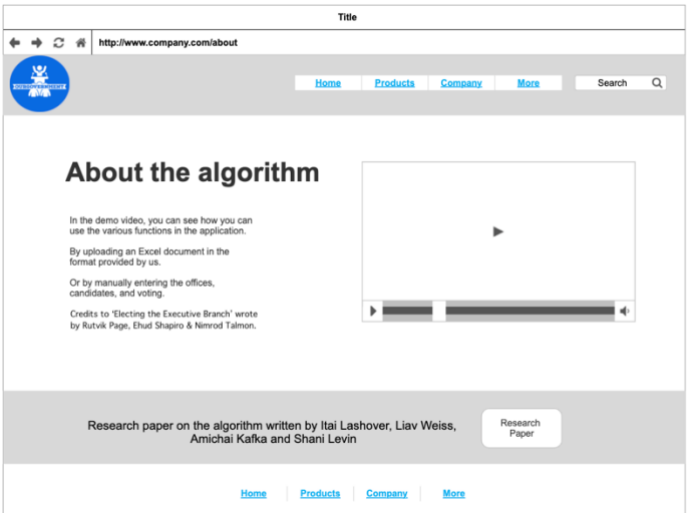
6.2 Screen Images











7. REQUIREMENTS MATRIX

Requirement	Which component meets the requirement
An algorithm for a fair and proportionate division of government ministers into ministries	GreedyPAV algorithm
UI & UX	Website (AboutOurAlgorithm component)
File upload support	Upload File & Download Template components in the website
Manual data entry support	Demo component in the website
The user will be able to use the system for a variety of purposes	Upload File & Demo components in the website
An explanation of the algorithm results will be displayed to the user	Description of results component
Comprehensive statistics on the results will be presented to the user	Download Full Statistics components in the website

8. APPENDICES

[1] Rutvik Page, Ehud Shapiro & Nimrod Talmon. *Electing the executive branch*: 2009.09734, 21 September 2020.

[2] template.xlsx