Purpose of the System

This system was designed with the aim of providing easy, credible and fair elections for users who want to use a vote to choose a person with the highest number of voters thereby declaring the person the winner. It makes the whole process an easy affair without too much hassle as all the users have to do is sign up and follow a few easy steps. It should also be available when there is an ongoing election. The system must be able to support or perform the following tasks/ functions (functional requirements (F)):

- allow voter or candidate to vote
- enable site users to become candidates
- have a single controller of the election processes(election admin)
- provide candidates with the option of opting out of elections
- allow voters to view detailed profile of candidate
- provide candidates with live updates of ongoing election that they are participating in
- give supreme control to election admin, actions such as add candidate, ban voter, disqualify candidate, create new election etc.

As mentioned earlier, the system aims to make the commonly used voting system less of a hassle in few steps. Also the system should be user friendly so even novices can be comfortable in using it. For these reasons, a person who needs to vote should make fourteen clicks to register that is including clicks made for login, approve and start registration so as to satisfy the usability non-functional (NF)requirement. The system must be running 100% of the time when there is an election so users enjoy smooth, hitch-free experience. Obviously there will be multiple users logged in at the same time trying to achieve their various goals so the system must allow least 500 parallel users so this goes in line with our performance NF.

References

The current election system is used generally around the world is on paper election method that requires voters to come to a specific place to vote and besides regional elections for examples in schools, this can be a really big problem.For example last year there was an election in our schools which get %40-50 participation percentage which is very very low. If we want to continue with school example the paperwork needs to be done by the school and candidates and when the election is going through there are a lot of people working in the process to keep the election safe and fast as possible it. After the election is finished there is a calculating votes process and you can't be sure if it's accurate or not fully. When the election is happening voters need to go to a specific place that is mostly not close to their house or they don't want to enter the queue for voting. Moreover, current election system is functional but not optimal. In this modern age, the voting process ought to be computerised and this will definitely ensure a greater voters' turnout and the person in charge of the elections will save precious time and money.

Overview

We have decided to choose the client/server architectural style because our system involves managing large amounts of data (as mentioned earlier we want to support having a minimum of 500 users logged in simultaneously) and we obviously want to grow a large community that use our voting system. Client/server architectural styles are well suited for such systems. Basically in a client/server architecture setting, there is a server which is responsible for providing service to the clients. These services are the resources or data stored on the DBSM. The clients are technically subsystems and they are responsible for communicating with the user. A browser is a type of client which communicates with the user and sends a request service to the server, normally HTTP. The client/server style also allows us to have a large number of users which is imperative in implementing our system and design goals. We want a large community to be engaged in using our voting system. The flexibility of the user interface of the client is also remarkable as it can operate on numerous platforms such as laptops,

PDAs, wearable computers etc. Using the client/server architecture also makes it better to have a central system which controls and stores all the resources (e.g Usernames and passwords) while also managing/limiting the level of access available to individuals. As stated in our reports, limiting access is a key component of our system. Also, since our users will be able to access the voting system from more than one computer or platform, limiting the access to resources comes into greater play because on each computer or platform that they access the the site, they have to be verified/configured every time they log on. The constraint on our system for using client/server is using HTML or PHP as the the programming language, this is the implementation constraint which affects the choice of our software architecture because we are dealing with databases.

Functionalities of subsystems:

1. *Validator*: This subsystem is going to be designed to ensure that voters can not vote more than once, it constantly communicates with the server and other subsystems and checks that the count of vote per election of every voter remains one. This makes the elections fair.
2. *Notification*: This subsystem handles the messaging and alerts of the users of the system. When a there is a change in event or a new event, this subsystem take note of that and updates the concerned users like during an election win, the candidate is notified.
3. *Electoral List*: This contains the list of voters who are eligible to vote for a particular election. By authenticating the voter with their ID and password, this ensures that only eligible voters can vote.
4. *Encryption*: This subsystem is responsible for preventing unauthorised access to sensitive or valuable information/resources stored on the database. It is going to function virtually in-between the server and the client. HTTP encryption can also be used so as to prevent access by third parties and this provides grounds for safe election and anonymity.
5. *Ballot Box:* This subsystem monitors the results, changes etc. of the ongoing elections. Voters place votes for candidates and this subsystem is responsible for counting them.

Definitions, Acronyms, and Abbreviations

Abbreviations/Acronyms:

1. HTTP- Hypertext Transfer Protocol, which is the transfer protocol used on the web
2. DBSM- Database Management System, which is used to store and access user data
3. NF- Non-functional requirements
4. F- functional requirements
5. PHP- Hypertext Preprocessor, is a programming language used to design websites and databases.
6. HTML- Hypertext Markup Language, a widely used programming language to design websites and webpages.
7. PDA- Personal Digital Assistant, devices that function as mini organisers with email and internet capabilities.

Definitions:

- Election admin- Election administrator who is responsible for overseeing the elections. He also creates and manages the elections, all the control in the election belongs to him.
- Novices- users who are not very proficient with operating computers
- Usability- a NF that is the measurement for the ease of use of a particular application or system.
- Parallel Users- used to identify multiple users who are accessing the service independently and simultaneously at the same time.
- Architectural Style- the setting that defines how a system will be developed.
- Constraints- limiting factors to the final output of the product.
- Database- set of stored information such as usernames and passwords.
- Server- the component which manages access to a centralised storage.
- Client- the part responsible for communicating with the server to access information or resources.

- Encryption: converting sensitive information into unreadable format or into code so as to prevent unwanted persons from accessing it.
- Subsystem- an extract from a larger system.