E- Voting System

Software Architecture: Requirement & Design

Submitted by:

Mohannad Alsofyani

E-VOTING SYSTEM

Introduction:

The word “vote” means to choose from a list, to elect or to determine. The main goal of voting is to come up with leaders of the people’s choice.
E - Voting System is an electronic way of choosing leaders via a web driven application. The advantage of E-Voting system over the common “queue method” is that the voters have the choice of voting at their own free time and there is reduced congestion. It also minimizes on errors of vote counting. The individual votes are submitted in a database which can be queried to find out who of the aspirants for a given post has the highest number of votes.

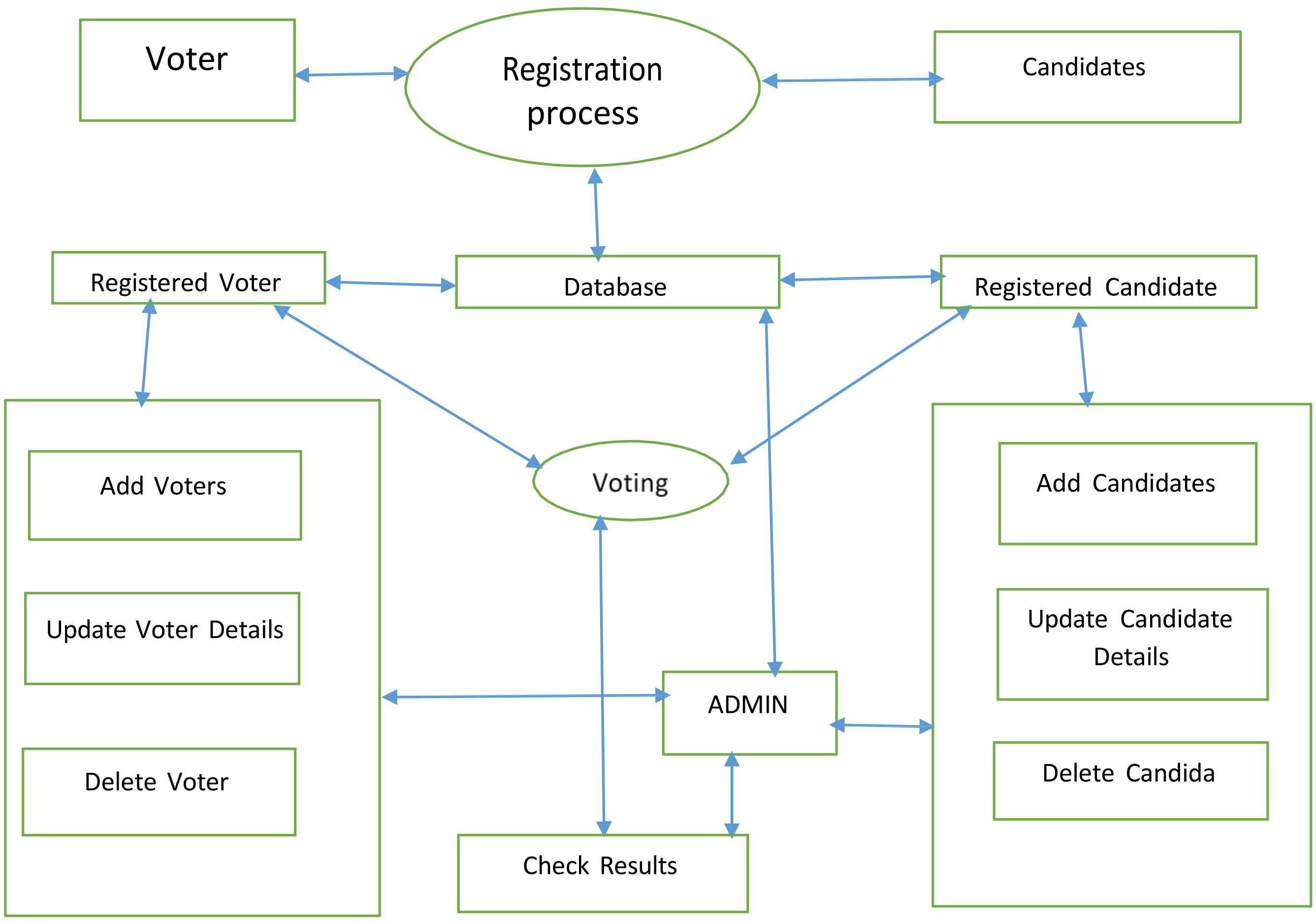
Project justification:

For one to participate in the elections, he\she must have the requirements.
For instance, he\she must be a registered citizen, must be 18 and above old, and any sex can give his\her vote online without going to any physical polling station. E-voting system shall reduce the time spend making long queues at the polling stations during voting. It shall also enable the voters to vote from any part of the globe since this is an online application available on the internet.

Requirements:

1- Registration of the voter is done by the system administrator.
2- System administrator can change the information any time if required.
3- Registration of the voter depends upon the information filled by the user.
4- Voter is given a unique ID and Password.
5- In the database information of every voter is stored.
6- Database shows the information of every user.

Architecture Diagram:



In E- VOTING SYSTEM a voter can use his\her voting right online without any difficulty. He\she has to be registered first for him\her to vote. Registration is mainly done by the system Administrator registers the voters on a special site of the system visited by him only by simply filling a registration form to register voter. After the validity of them being citizens of the country has been confirmed by the system administrator by comparing their details submitted with those in existing databases such as those as the Registrar of Persons, then the citizen is registered as voter. The registration should be done prior the voting date to enable data update in in the database.
After registration, the voter is assigned a secret Voter ID with which he\she can use to log into the system and enjoy services provided by the system.

Components of Architecture Diagram:

Voters: All Persons who votes.

Registration: Registers the voters and candidates participating in the voting system.

Candidate: All persons who gets voted.

Database: Contains all person who are registered voters and candidates.

Registered Voter: This Entity Has Add Voter, Update Voter Details, Delete Voter functionality.

Registered Candidate: This entity has add candidate, update candidate details, and delete candidate.

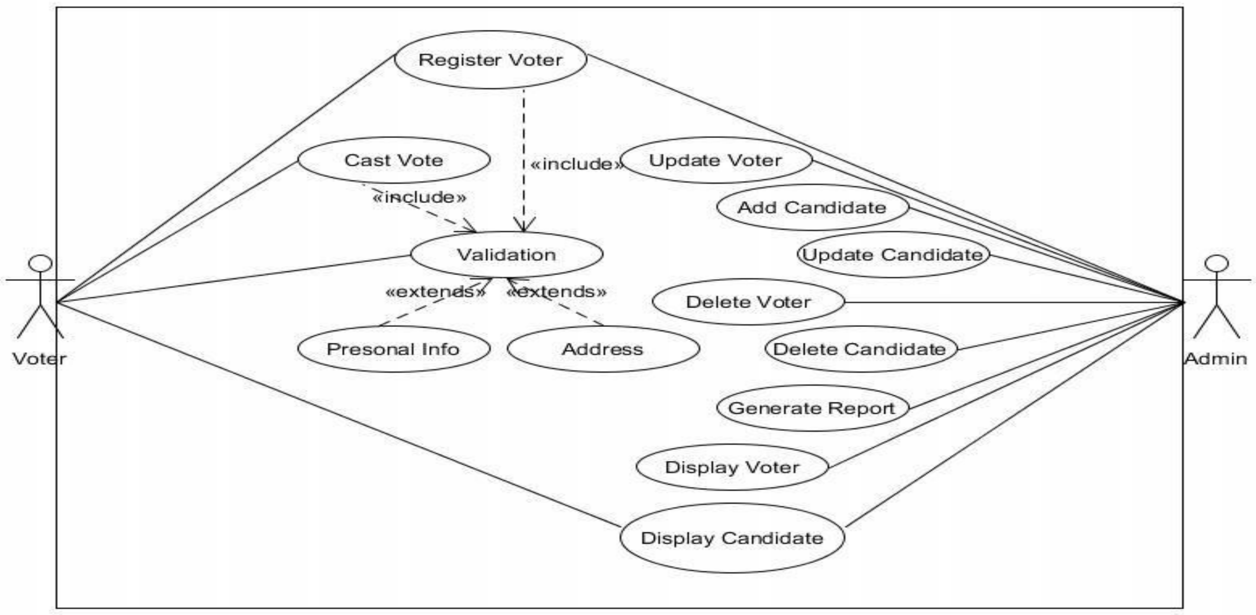
Voting: Register voter has to login with his credentials and votes to the Register Candidate.

ADMIN: Admin manages the whole system. He can Add, Update, Delete voters and candidates and can check the winner.

Check Result: Has count of number of votes each candidate got and check the winner.

Use Case diagram:

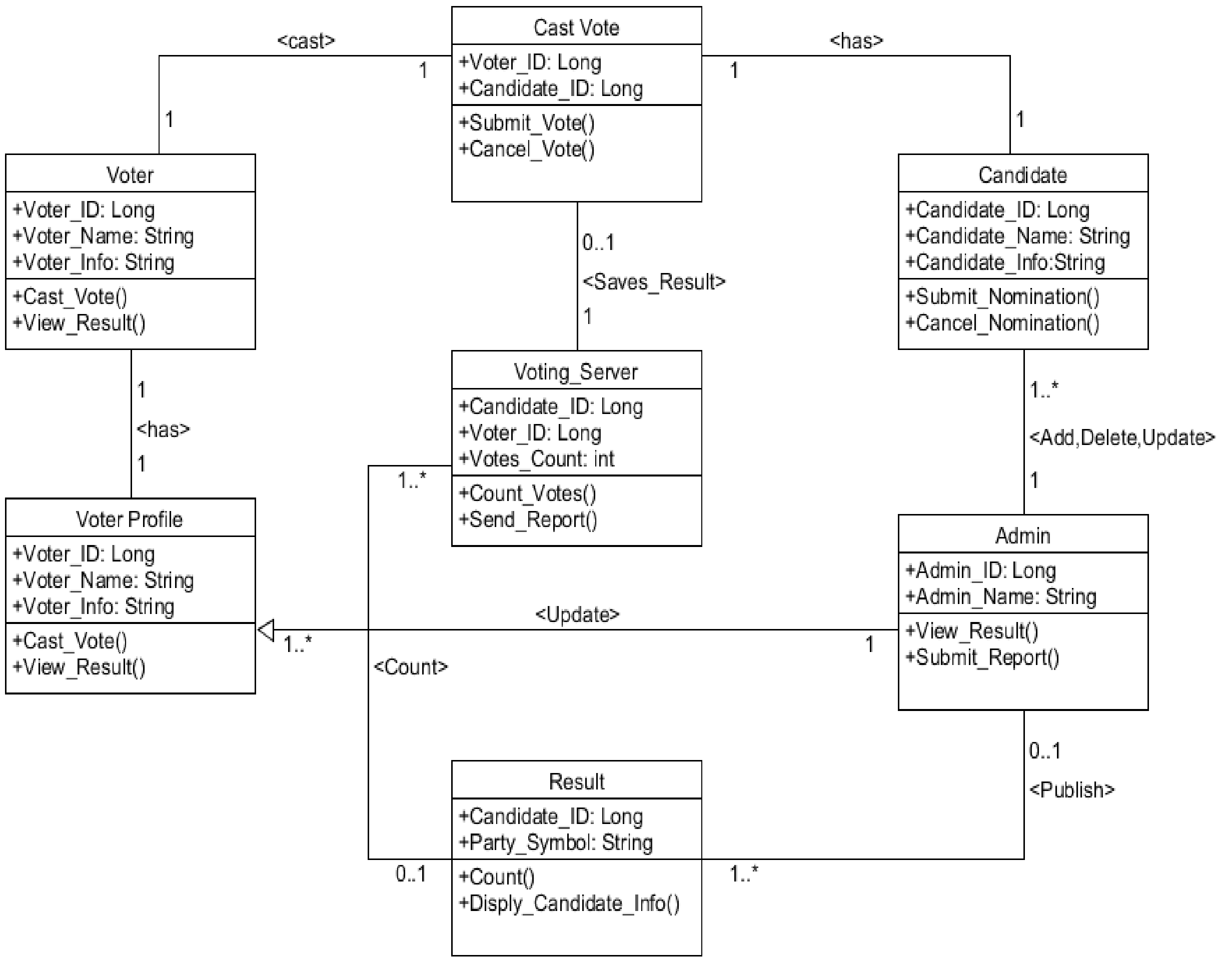
Consists of two actors Voter, Admin and a number of use cases.
Shows relationships between actors and use cases and relationships between use cases.
Actor
• A role that a user plays ‖ith respeĐt to the systeŵ
• RepreseŶted ďy a stiĐk figure.
Use Case
• RepreseŶted ďy aŶ o|al



<u>Class Diagram:</u>

• DefiŶitioŶ: a Đlass diagraŵ desĐriďes the types of objects in the system and the various kinds of static relationships that exist among them.

- Essential elements
- Classes
- Attributes
- Operations
- Associations
- Generalization
- Aggregation
- Composition



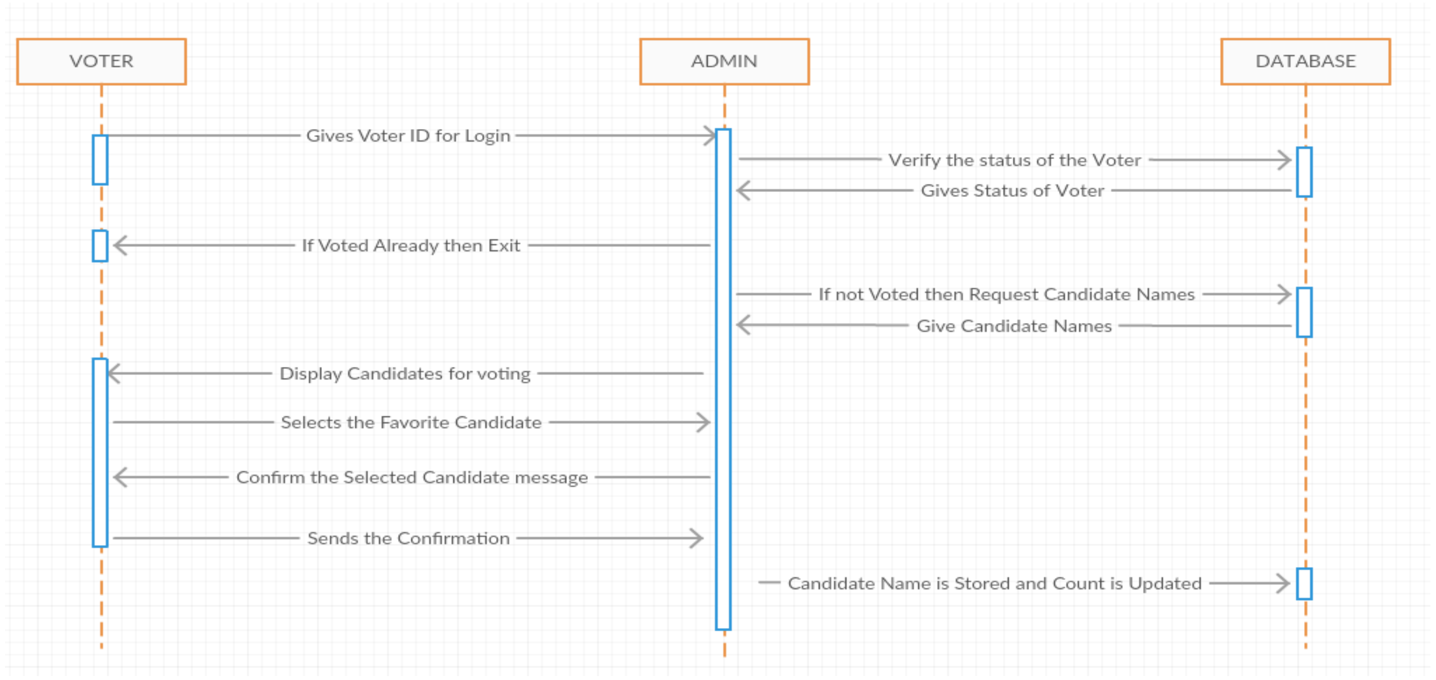
-

Sequence Diagram:

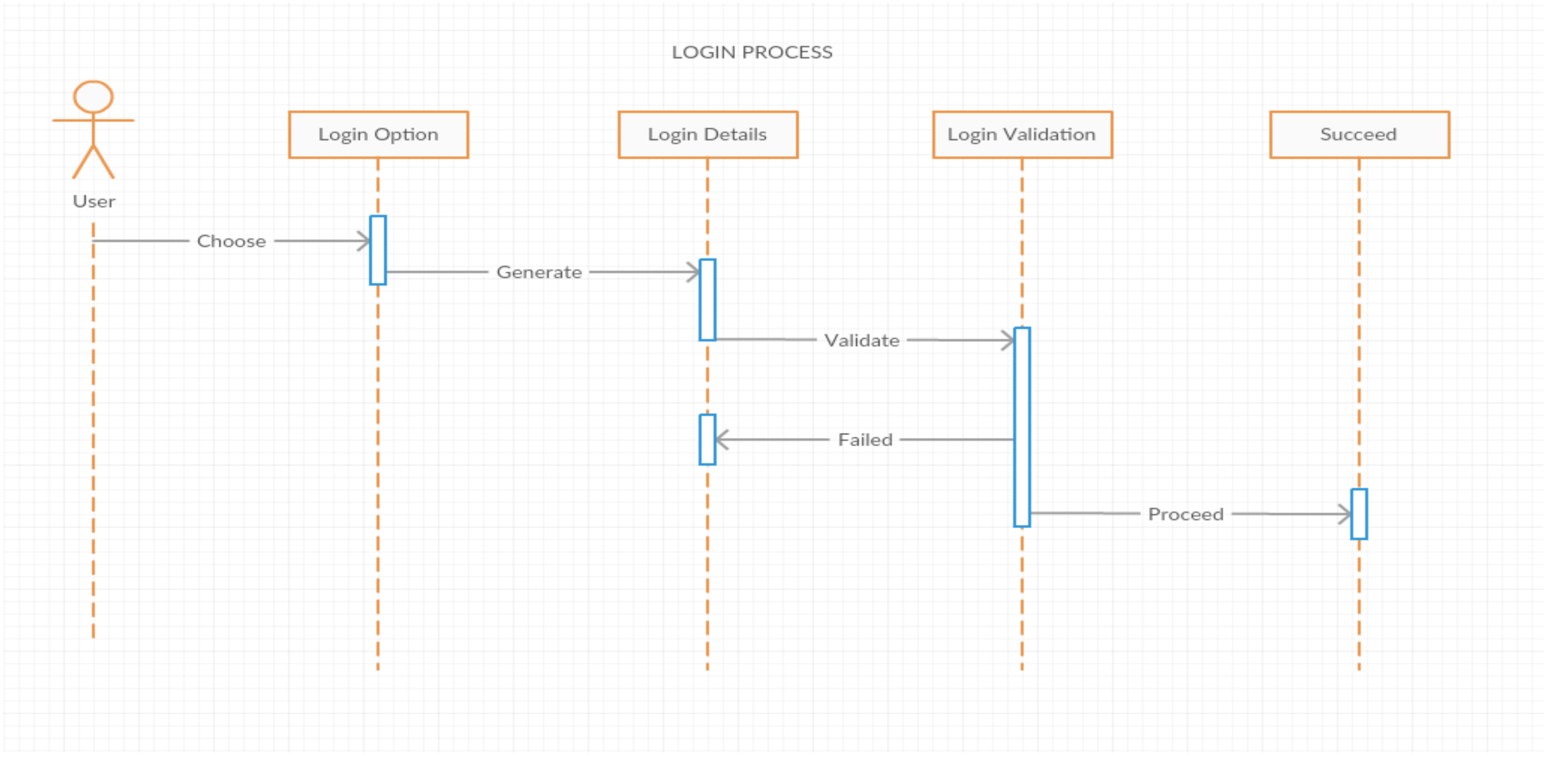
• A sequence diagram captures the behavior of single scenario.
• The diagraŵ sho‖s a Ŷuŵďer of partiĐipatiŶg objects and the messages that are passed between these objects within the scenario.

• The diagram conveys information along the horizontal and vertical dimension:
  - The vertical dimension shows, top down, the time sequence of messages/calls as they occur.
  - The horizontal dimension shows, left to right, the object instances that the messages are sent to.

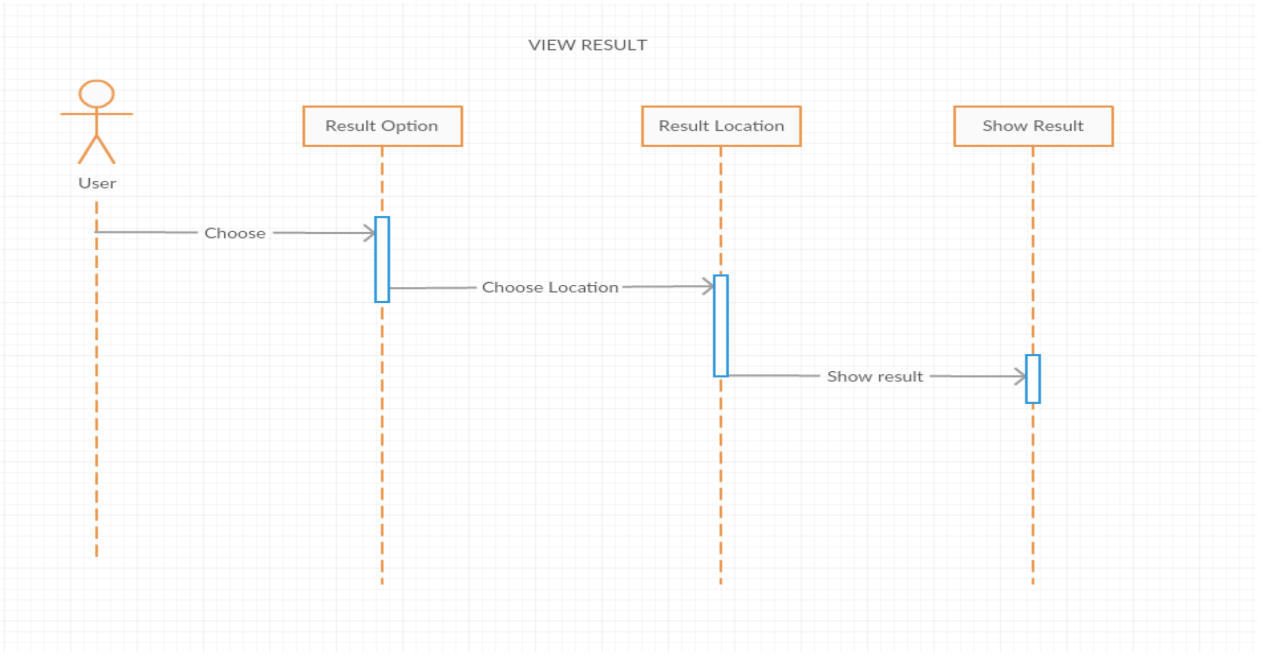
Voting whole process


VOTER
ADMIN
DATABASE
Gives Voter ID for Login
Verify the status of the Voter
Gives Status of Voter
If Voted Already then Exit
If not Voted then Request Candidate Names
Give Candidate Names
Display Candidates for voting
Selects the Favorite Candidate
Confirm the Selected Candidate message
Sends the Confirmation
Candidate Name is Stored and Count is Updated

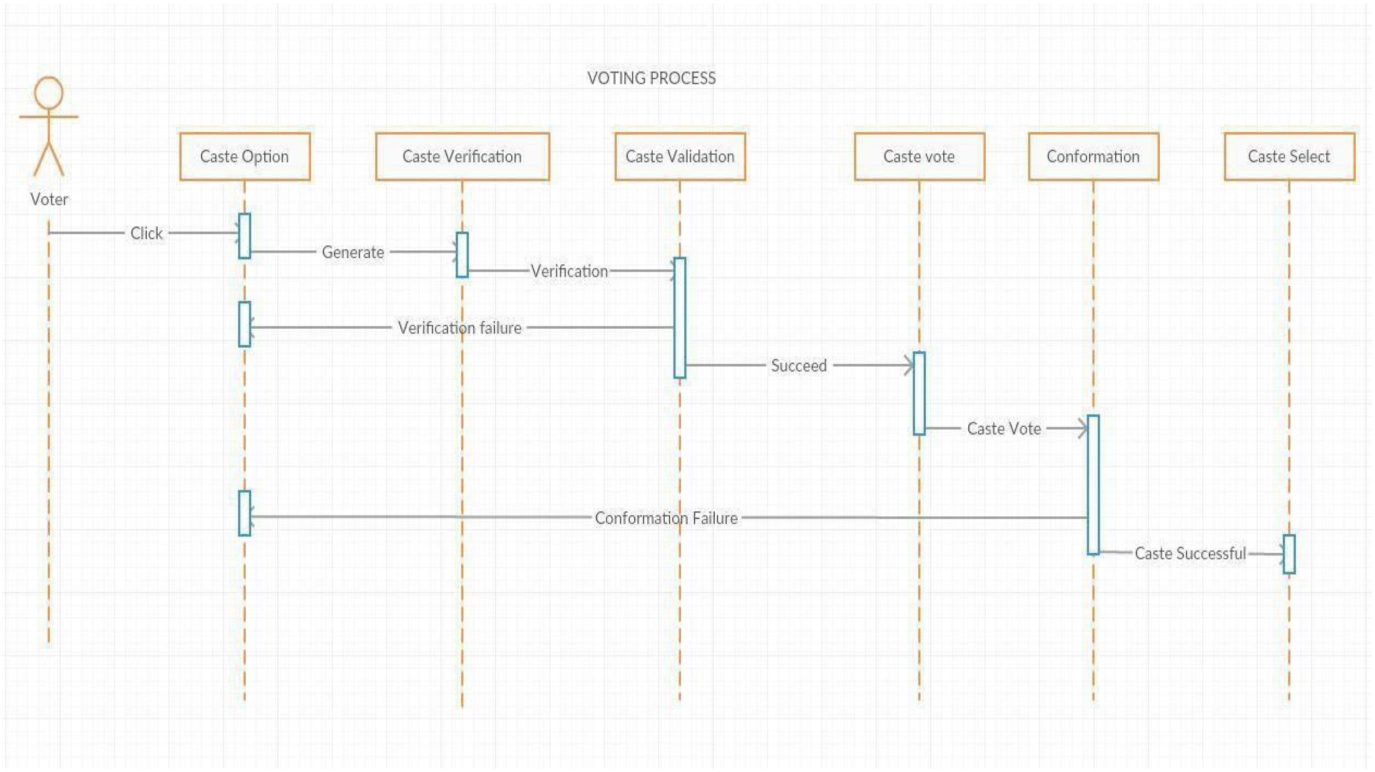
Login process



View Result



Voting Process



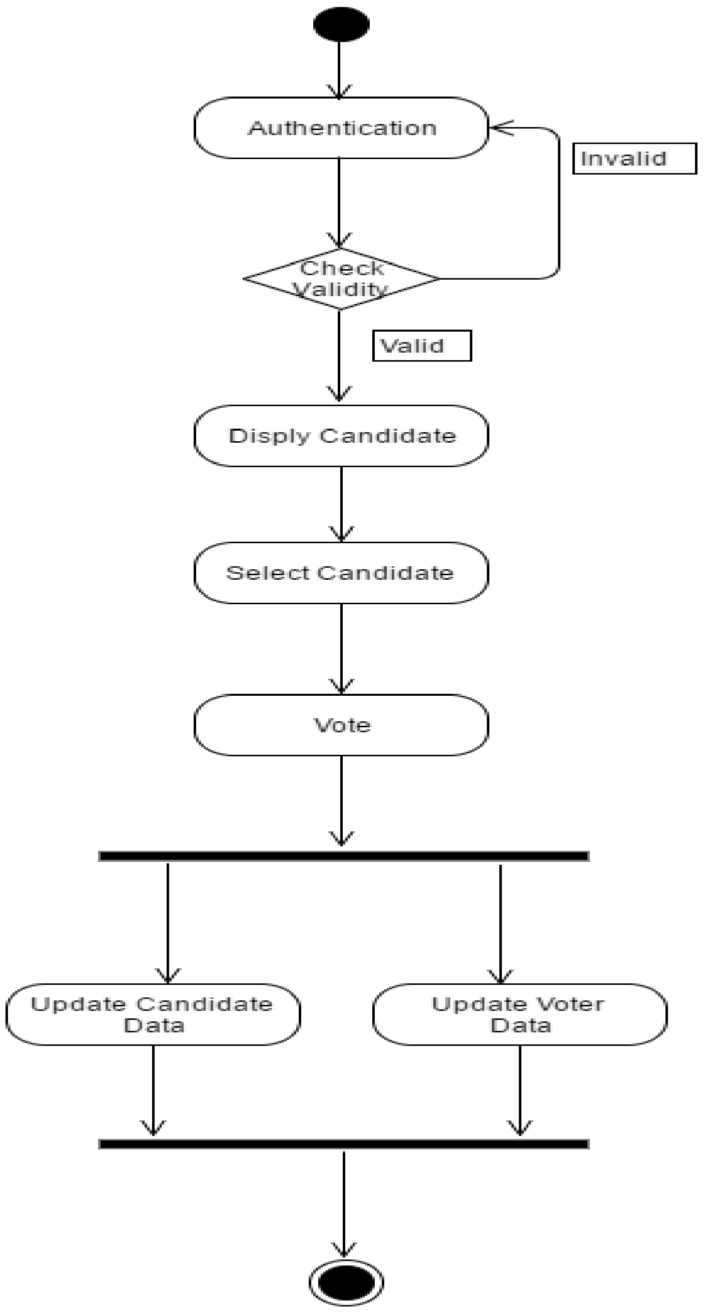
Activity Diagram:

An activity diagram describes the sequencing of activities, with support for both conditional and parallel behavior.

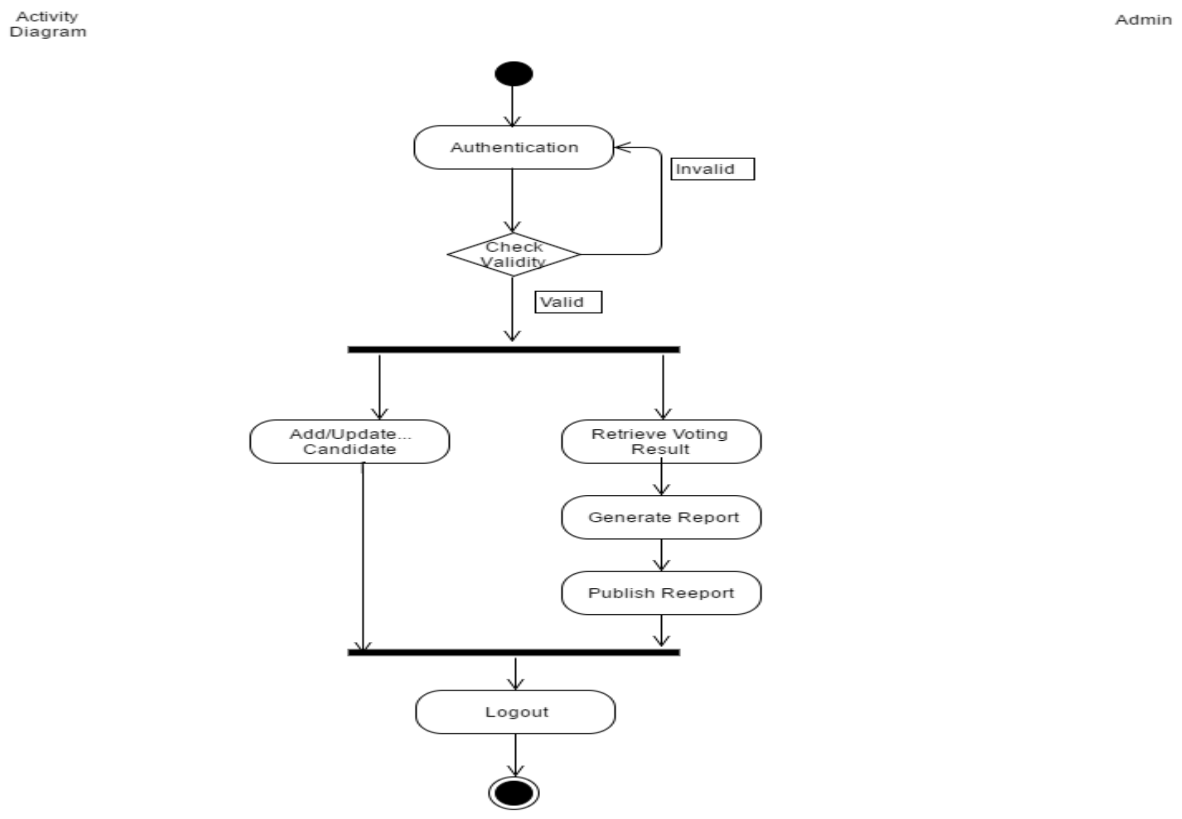
Conditional behavior:

- Branch
  - When the incoming transition is triggered, only one of the outgoing transitions can be taken.
- Merge
  - Marks the end of conditional behavior started by a branch.

Voter Activity



Admin activity



Architectural Style Used:

Client/server: architectural style suits best for this scenario.

Reason:

Because I am using online voting system, that means it should have a voting server to save the number of votes and client systems where voters vote to their favorite candidate.
Here Client should be aware of the server whether the server is authentic or not.

Components:

Server 1: Authentication server. Used to verify |oters' identity, and allow them to register if they are eligible, and also allow only one vote per voter.

Server 2: Voting server. Used to collect votes from voters, count number of votes for each candidate.

Client: the voter. Can only vote one time for one candidate. Once voters voted for a candidate, they cannot modify the vote.

Proxy: To prevent direct interaction between client and server.

Model-View-Controller Design Pattern:

Components:

- The *model* directly manages the data, logic, and rules of the application.
- A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- The third part, the *controller*, accepts input and converts it to commands for the model or view.

Interactions:

- A *model* stores data that is retrieved according to commands from the controller and displayed in the view.
- A *view* generates new output to the user based on changes in the model.
- A *controller* can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).

slides:

E-voting System

Traditional Voting System:

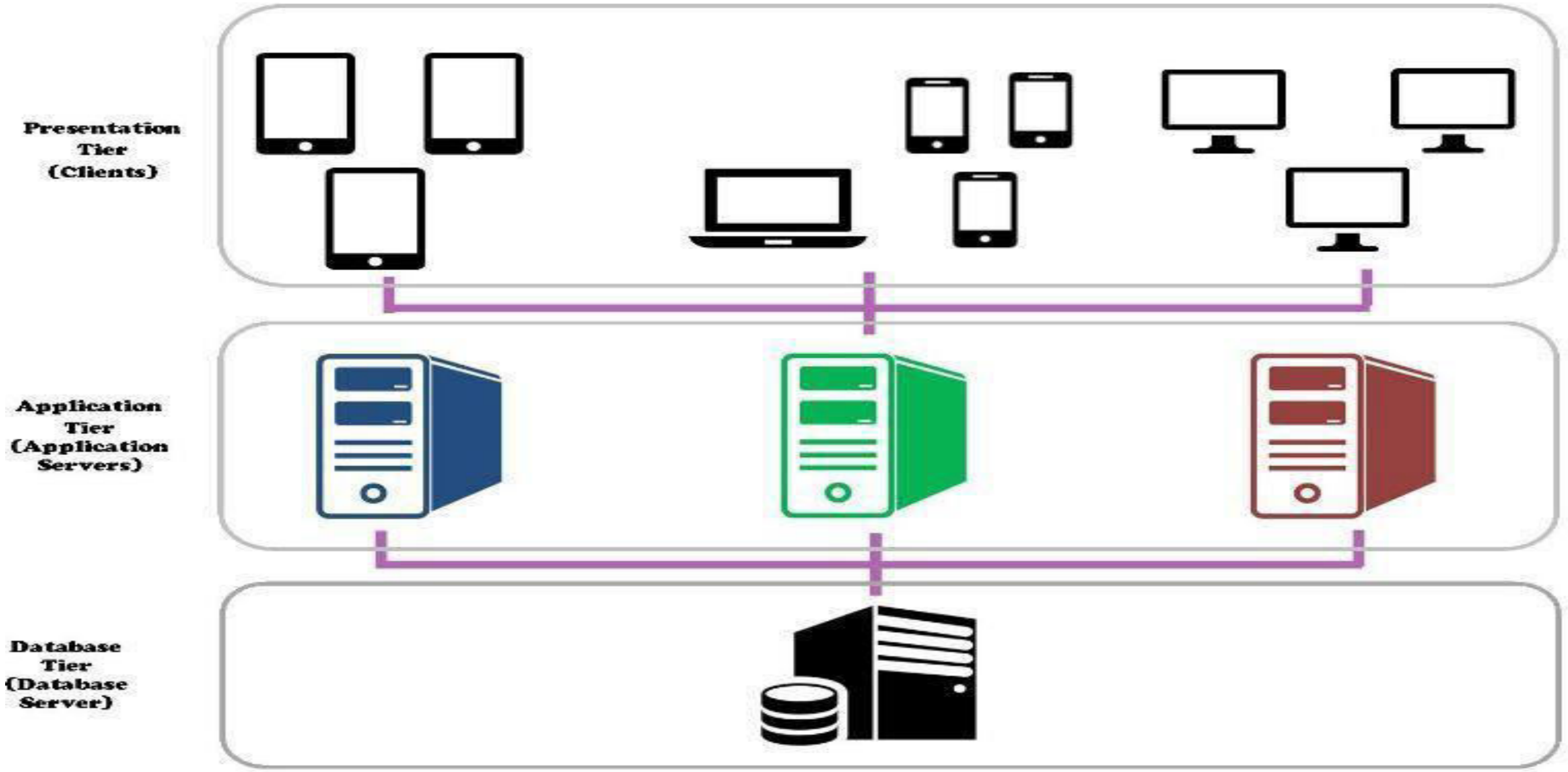
- Inefficient.
- Takes time and human resources.
- Does not give an instant poll result.
- Hard to track who voted and who don’t.

Our Style for E-voting

- Why this style ?
- High security.
- Ease of maintenance.
- Centralized data access.
- The architecture is separated into ordered layers.
- Client-to-client communication prohibited.

It is Client/Server style

Client/Serve Diagram:

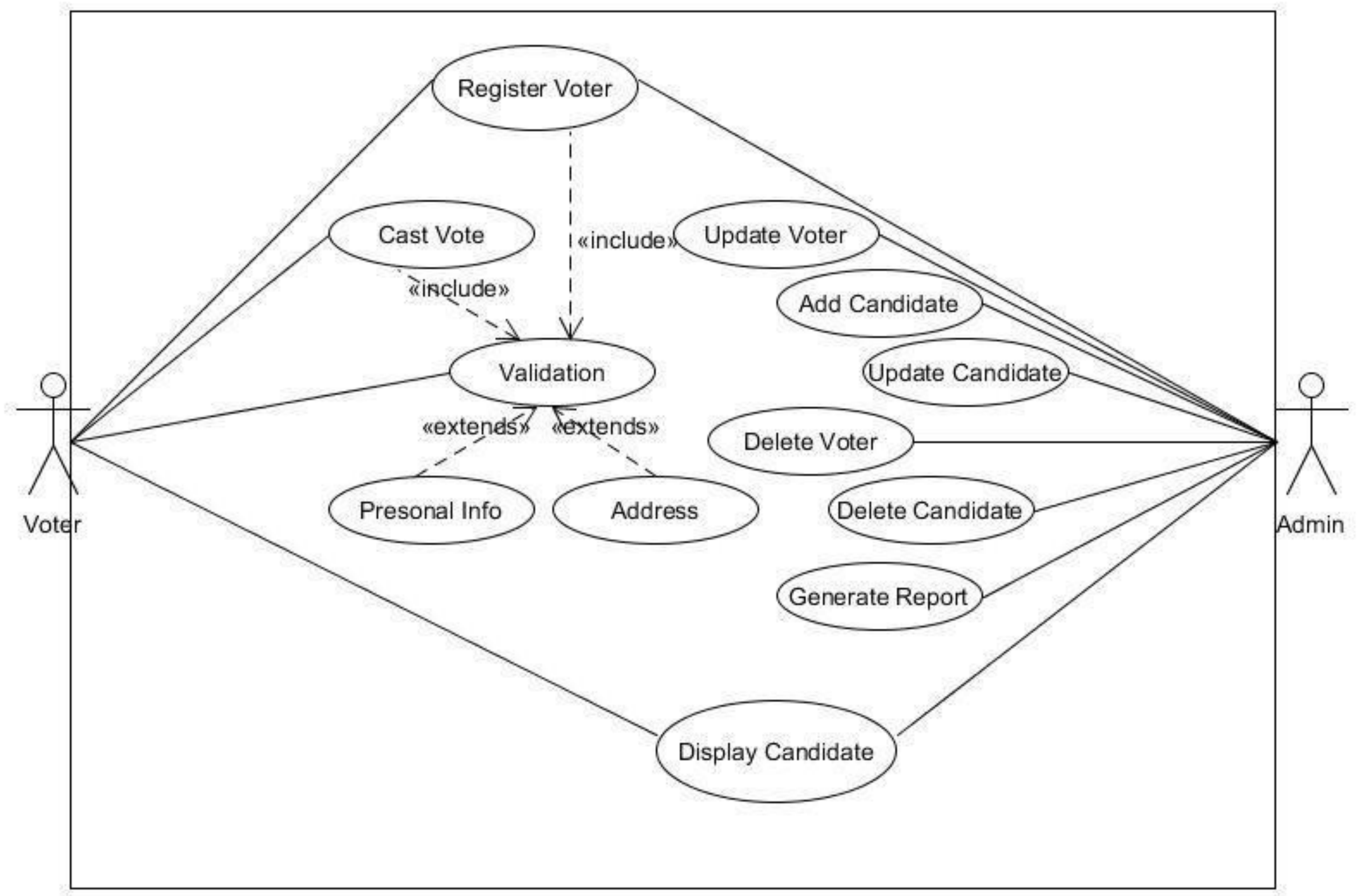

Presentation Tier (Clients)
Application Tier (Application Servers)
Database Tier (Database Server)

Considerations in Client/Server style?

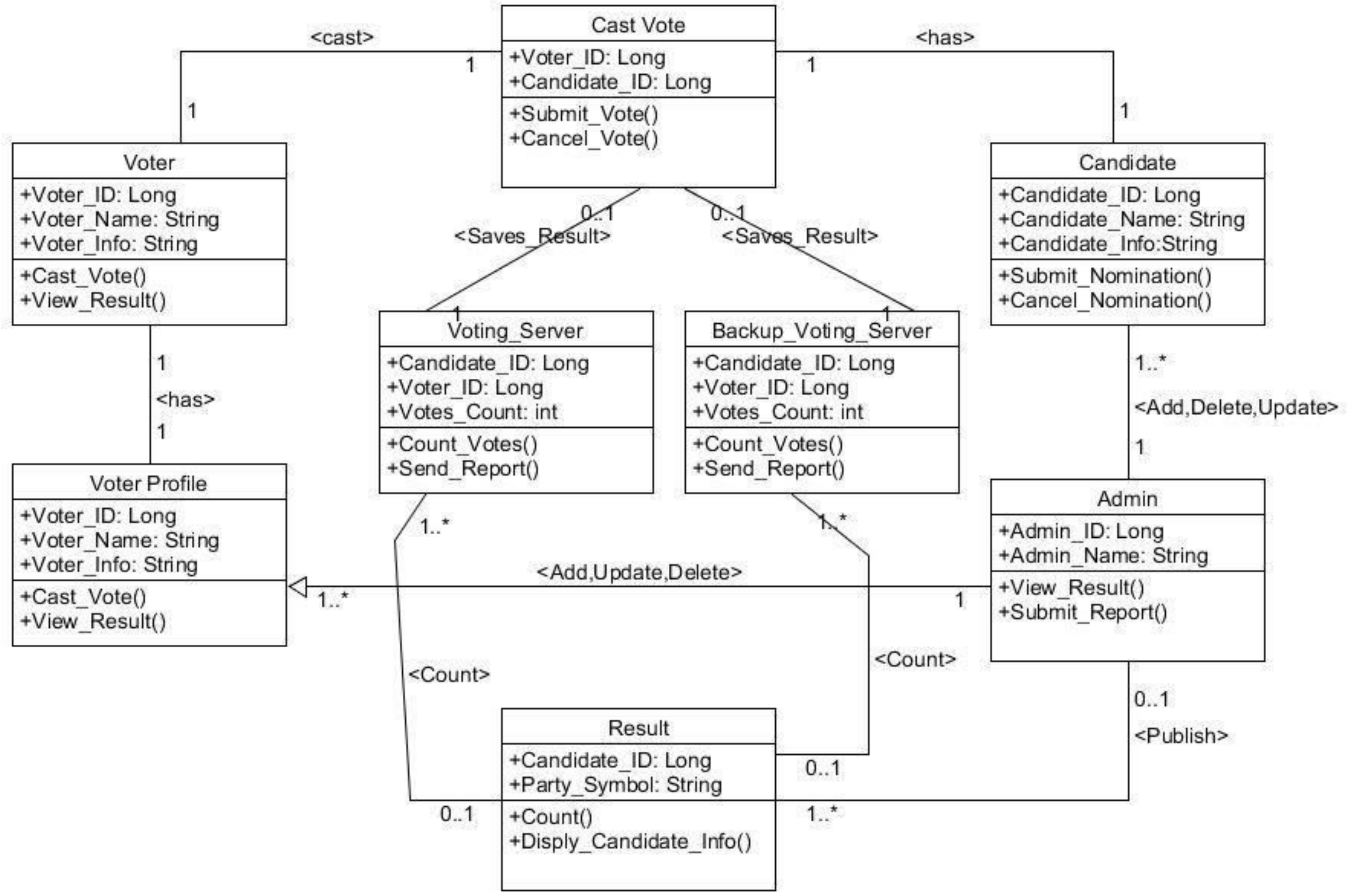
- A server application access by multiple clients.
- Create Web-based applications.
- Support different client types and different devices.
- Two problems need to think about:

1. Performance.
2. Dependence on a central server.

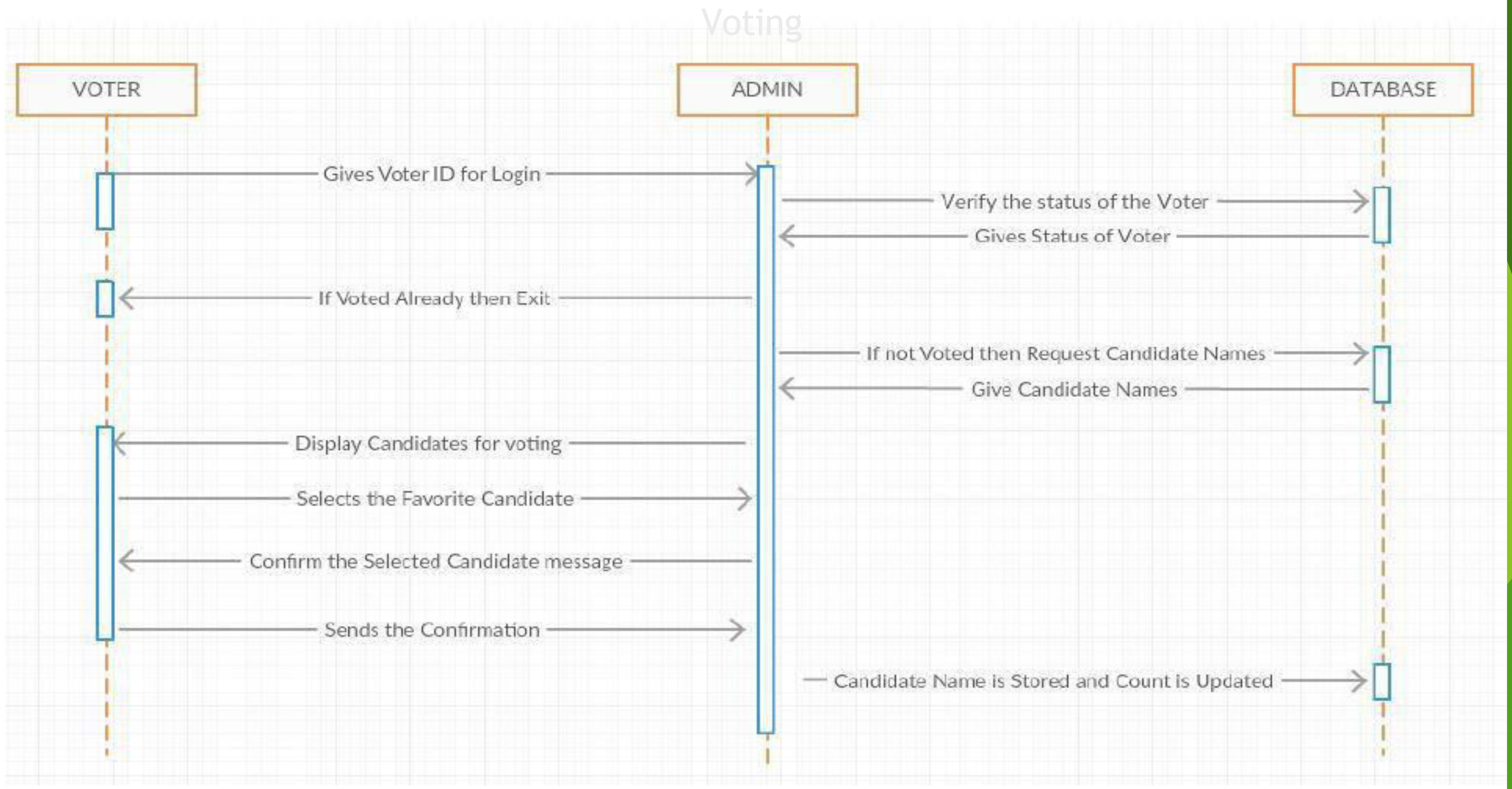
Use Case Diagram



Class Diagram

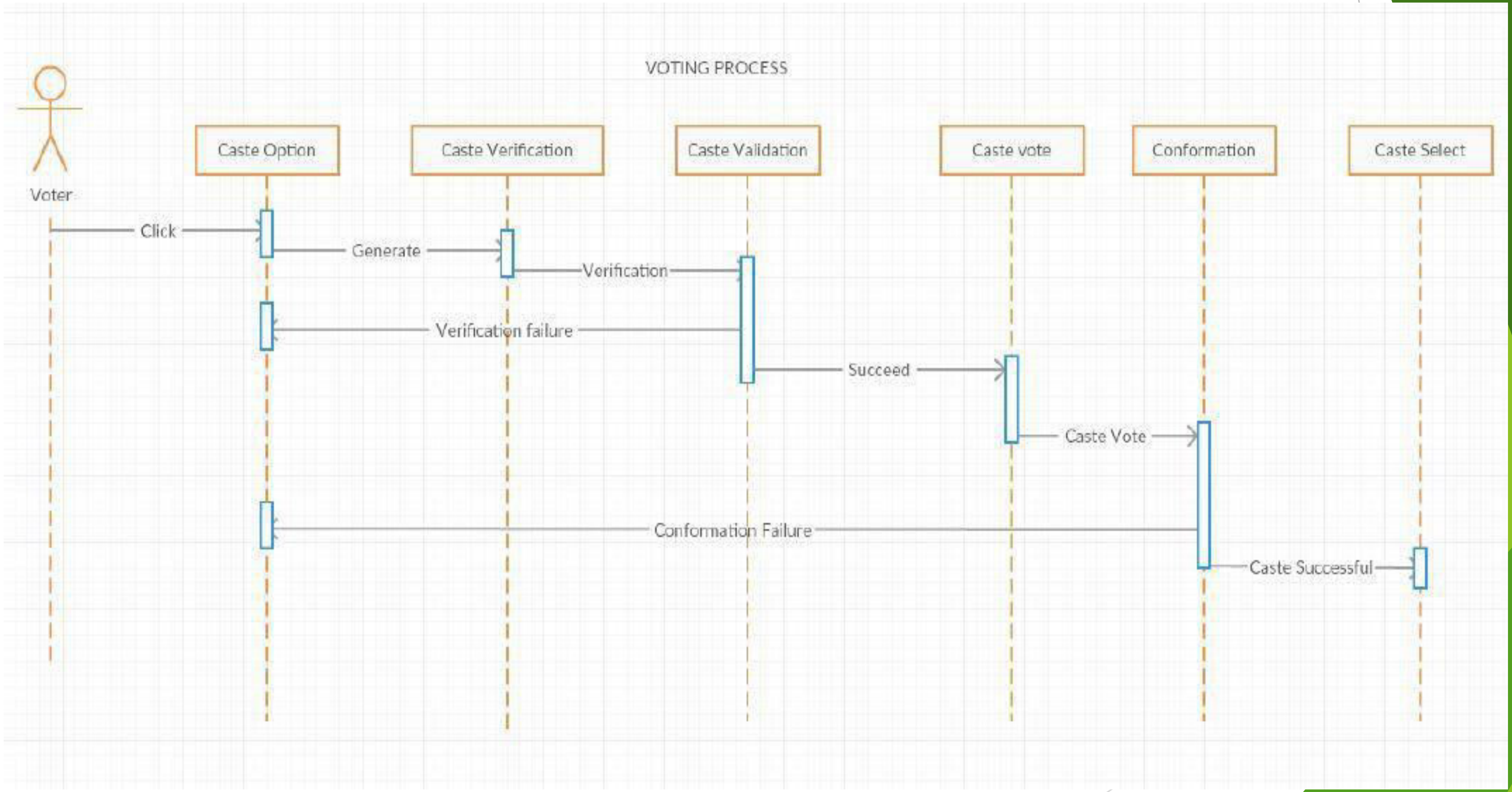


Sequence Diagram:

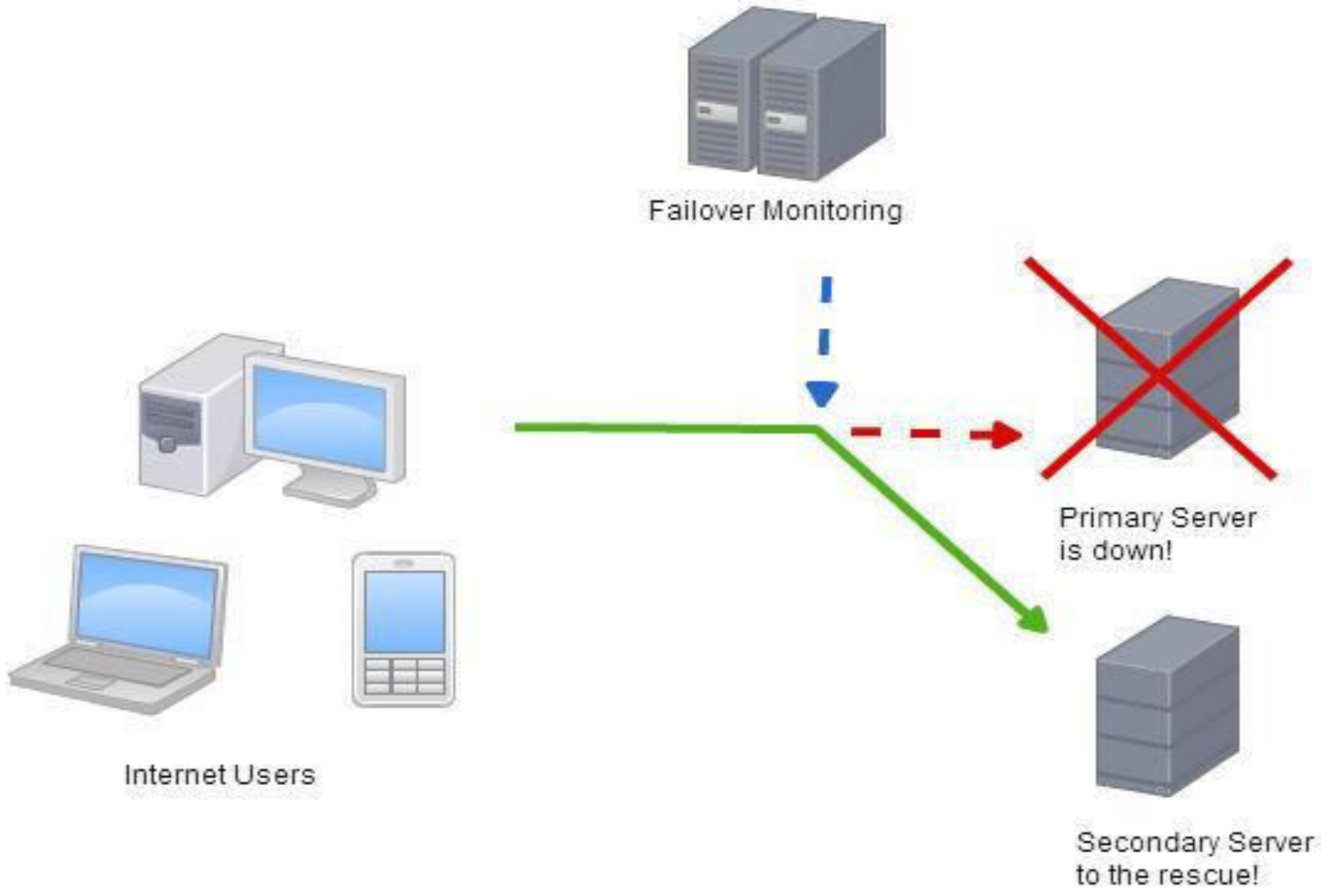


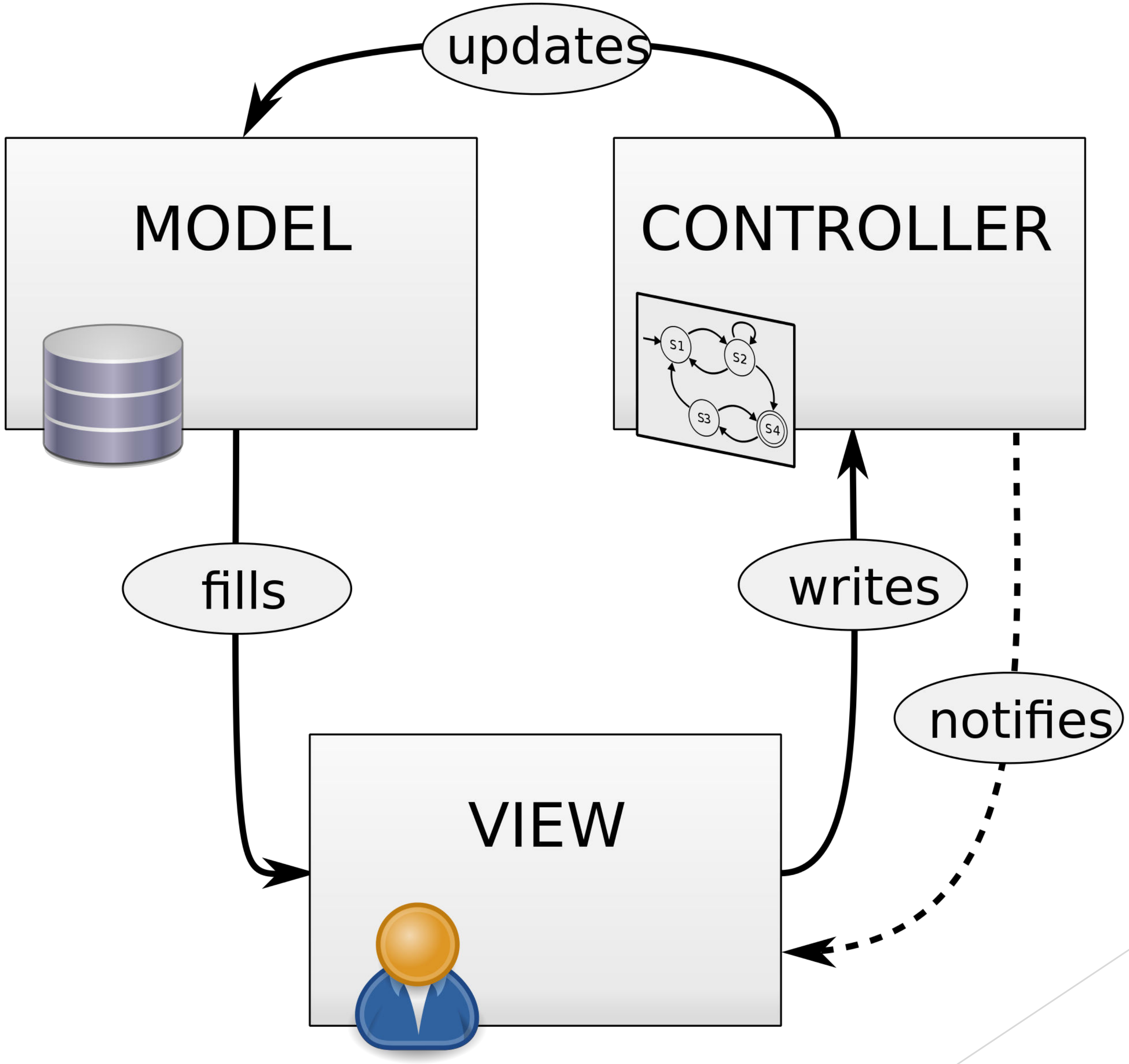
Voting
VOTER
ADMIN
DATABASE
Gives Voter ID for Login
Verify the status of the Voter
Gives Status of Voter
If Voted Already then Exit
If not Voted then Request Candidate Names
Give Candidate Names
Display Candidates for voting
Selects the Favorite Candidate
Confirm the Selected Candidate message
Sends the Confirmation
Candidate Name is Stored and Count is Updated

Sequence Diagram:



Fail Safe Plan:




updates
MODEL
CONTROLLER
S1
S2
S3
S4
fills
writes
notifies
VIEW

Thank you