**Electronic Voting Machine**

**Software Requirements Specification V3**

**Pratik Basyal**

**Date: 11/05/2024**

**CS 460 Software Engineering**

Contents

[1 Introduction 3](#_Toc14366)

[2. General Description 4](#_Toc14367)

[2.1 Block Diagram 4](#_Toc14368)

[3. Specific Requirements 5](#_Toc14369)

[3.1 External Interfaces 6](#_Toc14370)

[3.1.1 Input Events 6](#_Toc14371)

[3.1.2 Output Events 7](#_Toc14372)

[3.2 Control Logic 8](#_Toc14373)

[3.2.1. System State 8](#_Toc14374)

[3.2.2. Transition State 9](#_Toc14375)

[4. Design Constraints 13](#_Toc14376)

[5. Definition of Terms 13](#_Toc14377)

# 1 Introduction

This is the next generation electronic voting machine SRS document that defines a development of an electronic voting system for New Mexico. The purpose of this project is to develop an effective and efficient machine that is secure, and easily accessible means of voting, and that conforms to the democratic principles using the best technologies. Therefore, our focus will be to create an easy to use and accessible voting system so that all eligible voters can participate in it, while at the same time making sure that the electronic voting system is secure and transparent to make the voters have trust in the electoral process. This machine will be designed to produce precise, nontampered readings together with strict security requirements to ensure that its output cannot be accessed or changed by an unauthorized person.

The goal of this SRS document is to provide a roadmap for everyone engaged in the creation and implementation of the electronic voting machine. It identifies the functional and non-functional requirements, intent, and limitations for the guidance of every phase, in terms of the defined goals and regulations.

This SRS document is divided into five sections: Introduction outlines the project and document objectives; General Description provides a high-level system diagram and architecture; Specific Requirements details system flow, including External Interfaces (input events and hardware connections) and Control Logic (state transition model); Design Constraints covers limitations like hardware, legal, and environmental factors; and Definition of Terms lists relevant terms used throughout the document.

# 2. General Description

This section covers a broad spectrum of physical design related to our electronic voting machine. It gives an elaboration on their operations, with regards to the system's operation.

## 2.1 Block Diagram

The block diagram illustrates the physical interface of the components:

A diagram of voting machine

Description automatically generated

*Figure 1: Block Diagram*

Touchscreen: Acts as the core control point through which voters engage with the process, read the options, and make choices from the ballot.

Card Reader: Used in reading voter/official ID or authentication cards to confirm the identity of each voter. This is the initial stage of voting where only those who meet the laid down guidelines are allowed to vote.

Power On: Brings and maintains the voting machine on while providing constant power supply to the machine as the voting goes on.

SD Card (Read Only): Stores pre-loaded ballot information in a secure, non-editable format. This card is accessed only for reading data, ensuring that the ballot information remains unchanged.

SD Card 2, SD Card 3 (Write): Used as temporary storage for the voting’s and it safely stores every vote that has been deposited. This card presents the outcome of the voting as it is being conducted.

Latches: The Latches in the electronic voting machine act as Security Features that are meant to regulate access to the inner parts of this machine. Latches are open when the official open voting and closed when tampering happens, or voting is closed.

Printer: The printer prints out a record of each voter’s votes which is then stored in an internal compartment within the voting machine.

Tamper Sensor: Security component integrated within the voting machine to detect any unauthorized access or tampering attempts. It continuously monitors the physical integrity of the machine, and if any suspicious activity or forced access is detected, it triggers a security alert, or notifying election officials to ensure the integrity of the voting process is maintained.

# 3. Specific Requirements

This section analyses the technical and functional requirements needed for the proper functioning of the electronic voting machine, and their logical flow and interactions, and control flow. It uses a logical diagram to illustrate the components and how they are arranged in terms of the overall system flow and offers simple elaborations of a subset of the functional requirements that define how the system interfaces and interacts with external devices, as well as with its own internal control logic.

A diagram of a voting machine

Description automatically generated

*Figure 2: Logical Diagram*

## 3.1 External Interfaces

#### 3.1.1 Input Events

1. **Voter** 
   * newVoter(id):
     + 1. Uses the **card reader** for ID verification and the **touchscreen** to display the ballot after verification.
       2. Starts a new voting after verifying the voter through the card reader. When the voter is cleared, the voting machine electronically pulls up and presents to the voter the ballot on the touchscreen.

* + filledBallot(id, B):
    - 1. Uses the **touchscreen** for selections and **SD card 2 and SD card 3**for saving the choices with the voter’s ID.
      2. Records the voter’s choices made from the ballot. The choice is linked with the voter’s ID and preserved within the internal state.

1. **Official** 
   * newOfficial(id):
     1. Linked to the **card reader** for official ID verification.
     2. An official enters the system using a given Identification number.

* + machineSetup(id, BB):
    1. Involves the control unit for powering on and initializing, and **storage** for loading the blank ballot.
    2. The official starts the machine. The machine is not shut down (shutDown == false).
    3. The blank ballot (BB) is loaded in the system for the voter to complete.

* + openVoting(id):
    1. Uses the control panel to activate voting capabilities.
    2. Election official opens the voting to enable voting by the voters.

* + enableVotingSession(id):
    1. Connected to the control panel to initiate the voting session timer.
    2. Official opens the voting session for a certain period.

* + disableVotingSession(id):
    1. Connected to the control panel to pause or suspend the session.
    2. Official terminates the voting session for a short time.

* + closeVoting(id):
    1. Uses the control panel to end the voting process.
    2. Official brings the voting to an end to prevent further voting.
  + shutdown(id):
    1. Linked to the **power** to turn off the machine.
    2. Official shuts down the machine and is powered off.

1. **Device** 
   * tamperAlarm:
     1. Tied to the **tamper sensor** that activates if tampering is detected, locking the
     2. Comes on when the tamper sensor is triggered meaning that somebody has tampered with the machine and machine gets locked.
   * failureAlarm:
     1. Tied to Printer that sends alarm if there is any paper jam.

#### 3.1.2 Output Events

1. **Notification** 
   * tamperAlert:
     1. Linked to the notification system to alert authorities if tampering is detected.
     2. Authorities will know if an act of interfering is observed, thus, the integrity of the election is preserved.
   * failureAlert:
     1. Linked to the notification system to alert authorities if a critical failure occurs, paper jam.
   * voteComplete
     1. Linked to the notification system to confirm that a voter has successfully completed their voting process.

1. **Latches** 
   * openLatches: 
     1. Related to the **latch** mechanism to unlock and allow access.
     2. This event triggers the latches to open.

* + closeLatches: 
    1. Connected to the **latch** mechanism to lock and secure the machine.
    2. This event triggers latches to close

## 3.2 Control Logic

The control logic of the electronic voting machine is modeled using transition state which is an effective way of handling the various states of an event – in this case voting. In essence, this model defines when a transition should occur so that the system could respond to user action, like selecting the candidate, casting vote, or canceling the session.

### 3.2.1. System State

The system state consists of various variables that determine the status and operational flow of the voting machine:

* currentOfficialID = null
* setUpComplete = False
* latchStatus = “opened”
* openVoting = False
* enableVotingSession = False

* currentVoterId = null
* blankBallot = null
* voteCount = 0
* ballotSequence = {}
* voteConfirmed = false

* tampered = false
* failure = false
* shutdown = false

**currentOfficialID [String, Initial: null]**

Represents the ID of the current official operating the machine. It is set to null when no official is logged in.

**setUpComplete [Boolean, Initial: False]**

Indicates whether the machine setup is complete. Initially set to False and becomes True once the setup is finished and the machine is ready for voting.

**latchStatus [String, Initial: "opened"]**

Represents the status of the machine’s latch. Set to "opened" when the machine setup is not complete and changes to "closed" once the machine is ready for voting.

**openVoting [Boolean, Initial: False]**

Indicates whether voting is open. Initially set to False and becomes True when voting officially starts.

**enableVotingSession [Boolean, Initial: False]**

Controls whether a voting session can be started. Initially set to False, and it becomes True when the machine is ready for voting. **currentVoterId [String, Initial: null]**

Represents the ID of the current voter. It is null when no voter is interacting with the machine. **blankBallot [String, Initial: null]**

Stores the ballot template in markup language (e.g., HTML) given to the voter. It is null initially and gets populated with a blank template once the system setup is completed and ready for voting.

**voteCount [Integer, Initial: 0]**

Keeps track of the number of votes recorded. Initially set to 0, and it increments after each vote is completed.

**ballotSequence [HashMap, Initial: {}]**

Represents key-value pairs, where the key is the voter ID and the value is the filled ballot. A new pair is added after each completed vote.

**voteConfirmed [Boolean, Initial: False]**

Indicates if the voter has confirmed their vote. Initially False, becomes True once the vote is confirmed.

**tampered [Boolean, Initial: False]**

Indicates whether the machine has been tampered with. Set to False initially and changes to True if tampering is detected, shutting down, and preventing further use.

**failure [Boolean, Initial: False]**

Set to True if the machine encounters an issue, such as: A paper jam in the printer, A malfunction in the card reader, Power failure or system crash. It prevents further use until the issue is resolved.

**shutdown [Boolean, Initial: False]**

Indicates whether the machine is shut down. Initially set to False and becomes True when the machine is turned off.

### 3.2.2. Transition State

This transition state explains the structured way of managing the interactions within the electronic voting machine.

**Official Transitions: An official can only perform one event at a time.**

1. **newOfficial(id)**: **Checks:**
   * + **If currentVoterId == null** (no voter is logged in, so official can log in).
     + **If currentOfficialId == null** (no official is logged in yet, to prevent duplicates).
     + **If shutdown == false** (machine is not shut down).
     + **If failure == false** (machine is not in a failure state, ensuring it's functional).
     + **If tampered == false** (machine has not been tampered with).

**Action:**

* + - If all checks pass, assigns **id** to **currentOfficialId**, indicating the official is now logged in, and prepares the machine for the voting process.

1. **machineSetup(id, BB, shutDown)**: **Checks:**
   * + **If shutdown == false** (machine is not already shut down).
     + **If id == currentOfficialId** (only the logged-in official can perform the setup).

**Action:**

* + - If all conditions are met, assigns the blank ballot (**blankBallot** = **BB**) and sets **latchStatus** = "**closed**", indicating the setup phase is complete and the machine is ready for voting.

1. **openVoting(id)**: **Checks:**
   * + - **If id == currentOfficialId** (ensures only the logged-in official can open voting).
       - **If setUpComplete == true** (ensures that the setup is completed before opening voting).  **Action:**
       - If all checks pass, sets **openVoting** = **true**, allowing the machine to start accepting votes.
2. **enableVotingSession(id)**: **Checks:**
   * + - **If id == currentOfficialId** (only the logged-in official can enable the voting session).
       - **If openVoting == true** (ensures voting has been opened before enabling the session).  **Action:**
       - If all checks pass, sets **enableVotingSession** = **true**, allowing voters to start casting votes.
3. **disableVotingSession(id)**: **Checks:**
   * + - If **id == currentOfficialId** (only the logged-in official can disable the session).
       - If **enableVotingSession == true** (session must be active to disable it).

**Action:**

* + - * If all checks pass, sets **enableVotingSession** = **true**, allowing voters to start casting votes.

1. **closeVoting(id)**: **Checks**:
   * + - * If **id == currentOfficialId** (only the logged-in official can close voting).
         * If **openVoting == true** (ensures voting is currently open before closing it).

**Action**:

* + - * + If all checks pass, sets **openVoting** = **false**, signaling the end of the voting process.

1. **shutdown(id)**: **Checks**:
   * + - * If **id == currentOfficialId** (only the logged-in official can shut down the machine). **Action**:
         * If all checks pass, sets **shutdown** = **true**, turning off the machine.

**Voter Transitions:**

* 1. **newVoter(id)**: **Checks**:
     + - * If **currentVoterId == null** (no voter is logged in, ensuring only one voter can log in at a time).
         * If **currentOfficialId ==** null (ensures no official is logged in when the voter logs in).
         * If **shutdown == false** (machine is not shut down).
         * If **failure == false** (machine is not in a failure state).
         * If **tampered == false** (machine has not been tampered with).
         * If **voteConfirmed == false** (ensures the voter has not confirmed their vote before).
         * If **blankBallot != null** (ensures that a blank ballot is available to provide to the voter).

**Action**:

* + - * + If all checks pass, assigns id to **currentVoterId** and provides the blank ballot (**blankBallot**(id, B)) for the voter to fill out.
  1. **filledBallot(id, B)**: **Checks**:
     + - * If **currentVoterId == id** (ensures only the logged-in voter can submit their ballot).
         * If **voteConfirmed == false** (ensures the voter has not yet confirmed their vote).

**Action:**

* + - * + If all checks pass, adds the key-value pair (id, B) to ballotSequence.
        + voteCount is increased by 1.
        + voteConfirmed = true
        + voteComplete notification is send and all system variables goes back to system state.

**Device Transition**

1. **tamperAlarm Checks**:

* + - * If **shutdown == false** (machine is not already shut down due to another issue).
      * If **tampered == false** (ensures the machine has not already been marked as tampered).

**Action:**

* + - * If tampering is detected, sets tampered = true, **shutDown = True** indicating that the machine has been tampered with.
      * A tamperAlert event is triggered, notifying authorities about the tampering.
      * The machine is shut down to prevent further operations until cleared.

2. **failureAlarm Checks**:

* + - * If **failure == false** (machine is not in failure state yet).
      * If **tampered == false** (ensures the failure is not caused by tampering, distinguishing the two).
      * If **shutdown == false** (ensures the machine is not already shut down).

**Action:**

* + - * If a critical failure is detected, it sets **failure** = **True**.
      * Triggers failureAlarm event notifying the authorities about the failure, allowing them to take appropriate action like fix paper jam.

# 4. Design Constraints

The design requirements of the electronic voting machine describe parameters and guidelines that must be followed to correctly build a working, secure and regulatory-compliant system. Key constraints include:

**Regulatory Compliance**: Being an election system, the electronic voting machine must be compliant to federal as well as state election protocols such as the Help America Vote Act (HAVA, accessibility standards which include the Americans with Disabilities Act (ADA). In the same way, we still guarantee that the system is also legal, and every voter can easily follow and understand regulations.

**Security Requirements**: The machine must meet strict security standards of physical access, data input modification and data leakage. This involves protocols to ensure that the votes are safely stored, the communication between nodes in the system uses secure lines and there is protection of physical hardware from alteration.

**Usability:** The UI should make voting as easy as possible for every user regardless of their IT literacy.

**Hardware Limitations:** The electronic voting machine must work as per the chosen hardware components to the best of their abilities. These factors involve aspects such as processor speed and capacity, total storage space and devices for input and output together with their efficiency in performing at normal usage.

**Scalability:** The design should be scalable to be made bigger in the future or during the time of elections when demand increases drastically. For implement ability, this entails possibilities of combining the new technologies with the existing one, or enlargement for more electorate within the existing environment.

# 5. Definition of Terms

**State Transition Model:** A set of states of the voting process and interactions between these states upon user actions and system events that have been defined rationally.

**User Interface (UI):** A component of the electronic voting machine that takes one of the active interfaces, which are touchscreens, buttons, or visual surfaces, through which the voters accomplish their voting mission.

**Input Events:** Activities that are performed independently by the voters or the election officers that lead to changes in the status of the electronic voting machine including a selection of the candidate.

**Output Actions:** The types of output events produced by the electronic voting machine, for instance feedback on the display, audio feed back and voting record.

**Regulatory Compliance:** Compliance with federal, state, and local law and regulations required for the design and use of electronic voting systems as well as, accessibility and security requirements.

**Voting Process:** Procedures that a voter goes through to vote electronically, these are authentication, voting, and submitting the vote.

**Help America Vote Act (HAVA):** Mentioned federal legislation signed in 2002 is generally designed to enhance the management of elections in the United States and provides for the subsequent sections of voting systems, including accessibility.

**Americans with Disabilities Act (ADA):** An ember of civil rights that mandates that those with disabilities cannot be discriminated against and that electronic voting machines be created to allow the physically challenged to vote easily.