1. Introduction
   1.1 Purpose of the system
      - Count votes and handle vote weighting
   1.2 Scope of the system
      - Handle the counting and weighting of up to 20,000 votes, as one instance of the tabulation system should be able to process all the votes even in elections of several candidates in one ballot.
   1.3 Objectives and success criteria of the project
      - The objective of our role in the project is to tabulate votes and ensure the process is secure and accurate. We succeed by presenting election results to the proper people in a structured, understandable format.
   1.4 References
      - Object-Oriented Software Engineering using UML, Patterns, and Java Third Edition *Bernd Bruegge & Allen H. Dutoit*
      - Google Forms
      - Qualtrics
   1.5 Overview
      - In this section, we have identified the purpose of the system - to count and weigh votes - and the scope of the system - handling up to 20,000 voters at once. We have identified our success criteria as creating a system that is secure and accurate, that presents results in a structured way to only the proper people. being the security and accuracy of the tabulation process and the structured exportation of the data to those who require it.
2. Current system
      - There is no centralized system for voting on campus, so we are creating one to be used in the student body and club elections.
      - When not done by pen and paper, most voting is done by Google Forms and Qualtrics.
3. Proposed system
   3.1 Overview
      - We are going to make a Visual Studio application that will receive votes, verify that we have as many ballots as voters interacting with the system, and display the winner to the screen. The system will be able to handle a number of votes equal to double the number of the student population to account for the case of a popular election of many candidates at once. The system will also have a simple UI designed so the end user can operate the program without the need to undergo extensive training on system operation.
   3.2 Functional requirements - when users use this system what happens
      - The system collects data from the voting database and shows total votes for each candidate
      - The system shows candidates' vote totals and percentages
      - The system can access any database created by the voting teams
   3.3 Nonfunctional requirements
      3.3.1 Usability

- The vote arbiters must be able to navigate the system - request election results and read election results - without prior training or familiarity with any specific systems.

3.3.2 Reliability

- Should the system experience a failure that results in a crash, it should be equipped with a back up that preserves data.
- If the system crashes on one device it must not affect the results of another.

3.3.3 Performance

- The system must be able to conclude vote tallying, validation, and redundant repetitions of those in under an hour.

3.3.4 Supportability

- The system can access new specified vote databases without changing source code.
- The system must be able to handle FPTP and RBV systems.

3.3.5 Implementation

- The tabulation system will be written in C++ on Visual Studio

3.3.6 Interface

- There is no legacy interface or standards imposing restrictions upon the interface of the new tabulation system.
- The interface must be simple and clear to any new users in both requesting results and showing information to users.

3.3.7 Packaging

- A flash drive with a .zip folder that contains the program.

3.3.8 Legal

- Consent to collect personal information form
- FERPA
- Data use form

3.4 System models

3.4.1 Scenarios

-

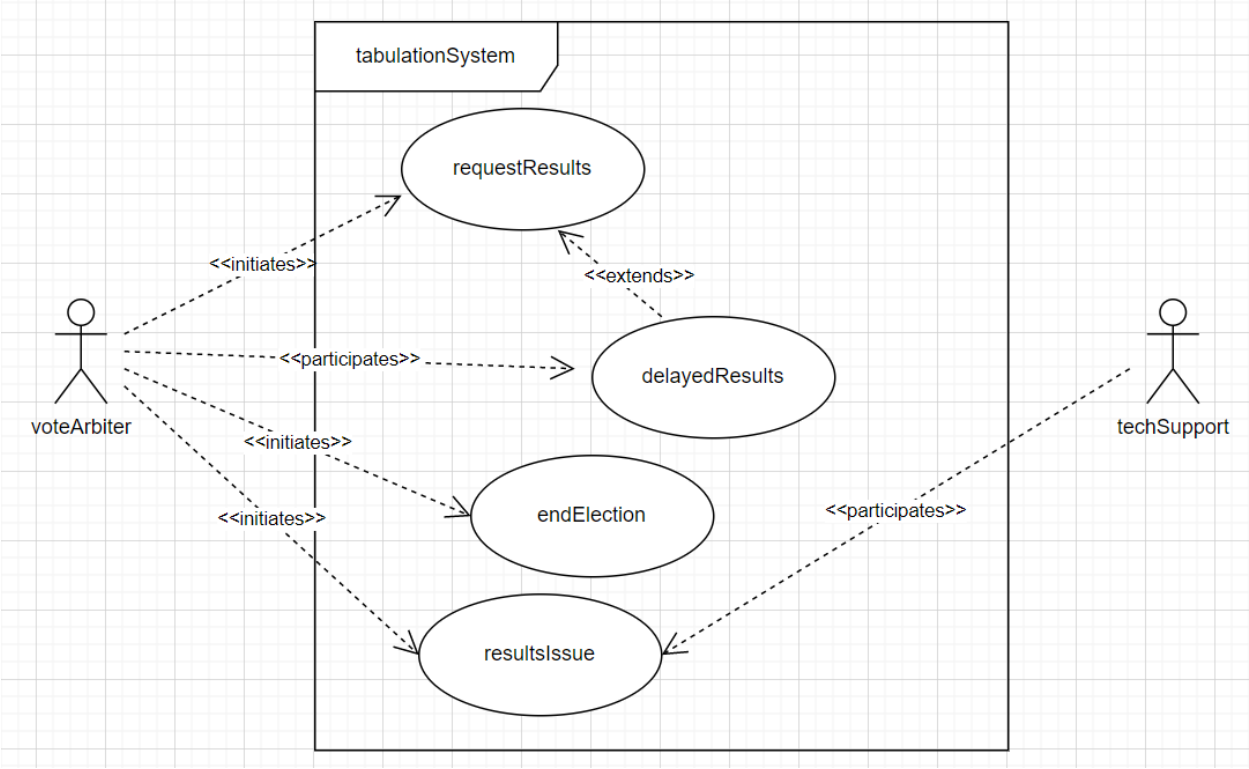
Scenario Name	getElectionResults
Participating actor instances	voteArbiter
Flow of events	1. The voteArbiter uses the tabulationSystem UI to request election results. 2. The tabulationSystem shows a visual acknowledgement of the request and checks to see if the counting system has finished counting votes. 3. The tabulationSystem receives confirmation and displays the

	election results on the UI

-

Scenario Name	resultsIssue
Participating actor instances	voteArbiter, techSupport
Flow of events	1. The voteArbiter sees the election results and something seems wrong with them, so they report an issue from the results screen. 2. tabulationSystem notifies techSupport 3. techSupport sees the issue and communicates with the voteArbiter to resolve the issue.

3.4.2 Use case model



Use Case name	requestResults

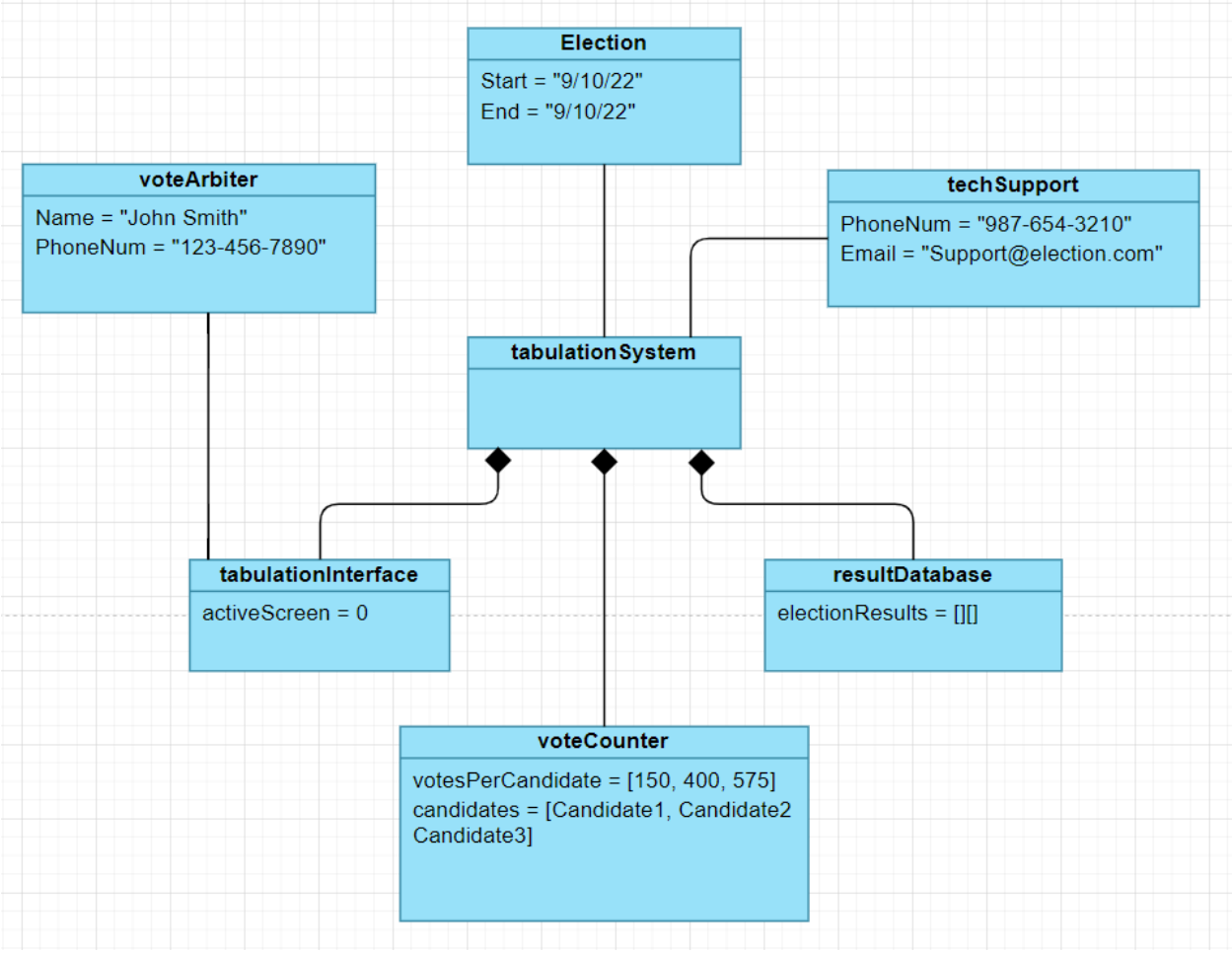
Participating Actor instances	Initiated by voteArbiter
Flow of events	1. The voteArbiter uses the tabulationSystem UI to request election results. 2. The tabulationSystem displays a visual acknowledgement of the request and queries the counting system for completion. 3. The tabulationSystem receives confirmation and displays the election results to the voteArbiter on the UI.
Entry conditions	1. The voteArbiter has verified they are responsible for having created the election in the ballot creation system.
Exit conditions	1. The voteArbiter sees the election results.

Use Case name	resultsIssue
Participating Actor instances	Initiated by voteArbiter, techSupport participates
Flow of events	1. The voteArbiter perceives any error in the election results, so they report an issue using the UI. 2. tabulationSystem notifies techSupport with a message containing which election has an issue, and who the voteArbiter is. 3. techSupport sees the issue and communicates with the voteArbiter to resolve the issue.
Entry conditions	1. The voteArbiter raises the issue using the tabulationSystem UI.
Exit conditions	1. The issue is resolved.

Use Case name	delayedResults
Participating Actor instances	voteArbiter participates
Flow of events	1. The tabulationSystem displays a temporary message to the voteArbiter communicating that the election results are still being calculated.
Entry conditions	1. This use case extends the requestResults use case. It is triggered when the tabulationSystem's counting system does not confirm that votes have been counted.
Exit conditions	1. The temporary message fades from the screen.

Use Case name	endElection
Participating Actor instances	voteArbiter initiates
Flow of events	1. voteArbiter is satisfied by the results of the election, so they press the End Result button 2. The system archives the results and the UI closes.
Entry conditions	1. Election results screen showing.
Exit conditions	1. Election results are archived by the system

3.4.3 Object model

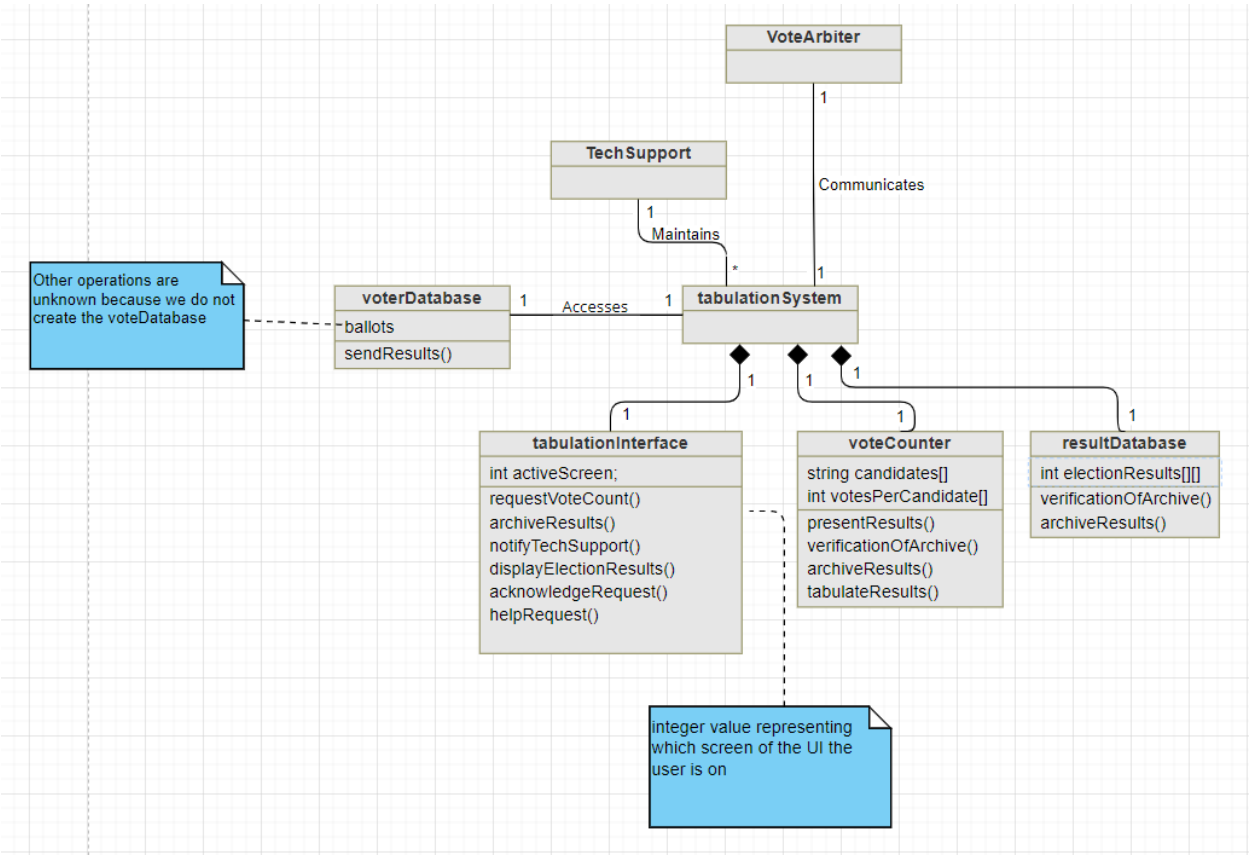


3.4.3.1 Data dictionary

- tabulationSystem - the name of the whole of the system
- techSupport - human responsible for maintaining the system and dealing with issues from parties concerned with the election results and the tabulationSystem

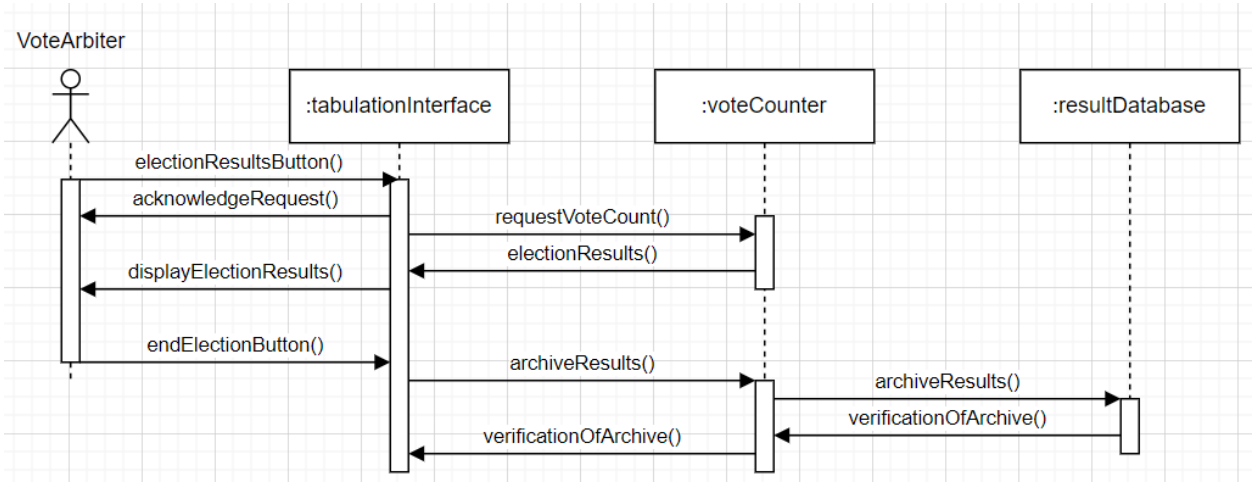
- voteArbiter - the main user of the system; will elicit and receive the results of the election
- voterDatabase - the database that stores votes, created by the vote collection team
- tabulationInterface - the interface boundary object of tabulationSystem
- voteCounter - the subsystem control object responsible for counting votes and accessing the voteDatabase
- resultDatabase - database created by tabulationSystem that stores the results of the election

3.4.3.2 Class diagrams

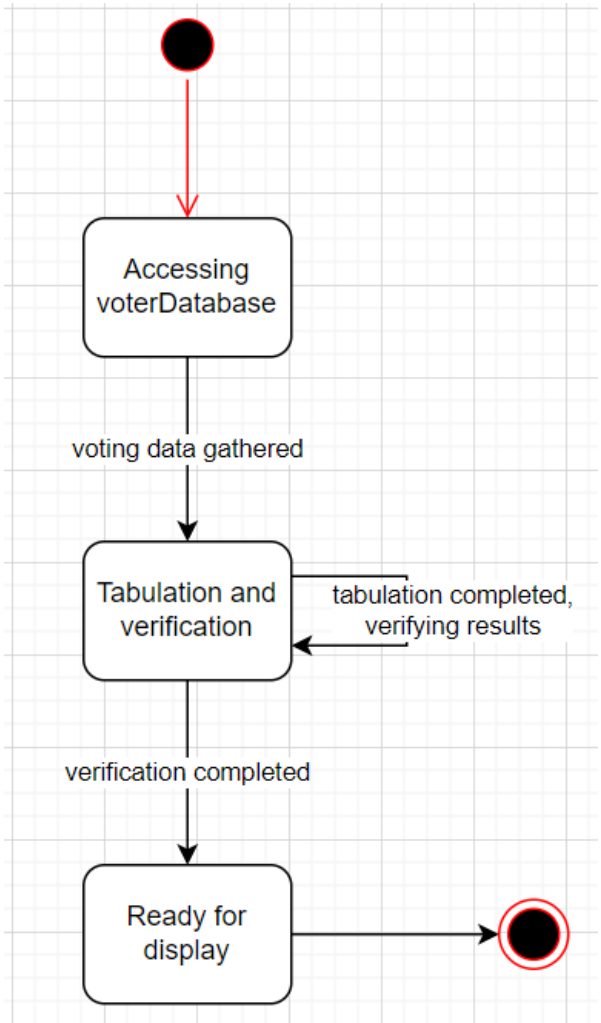


3.4.4 Dynamic models

Sequence Model:

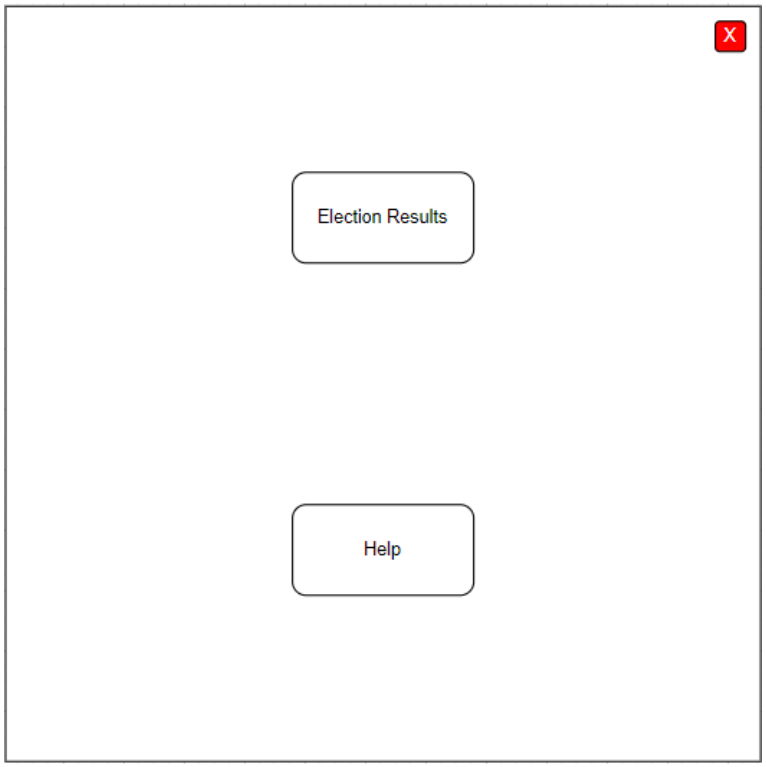


State machine model:

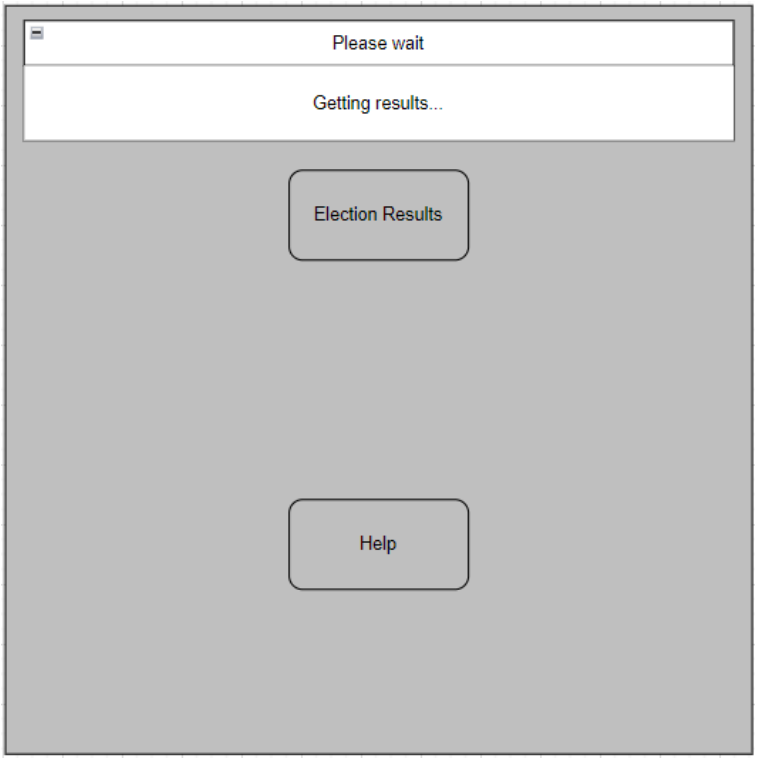


3.4.5 User interface

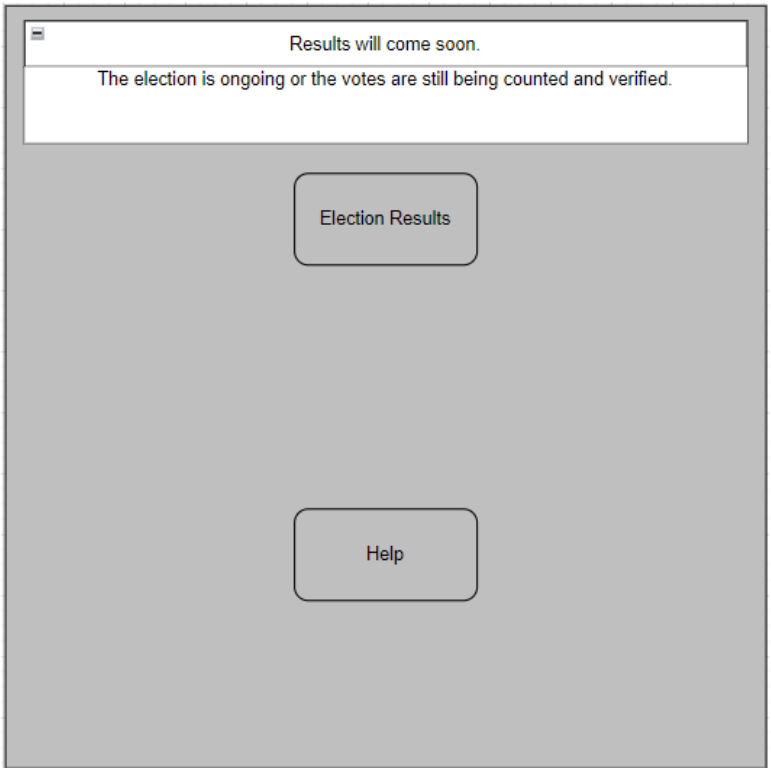
Default



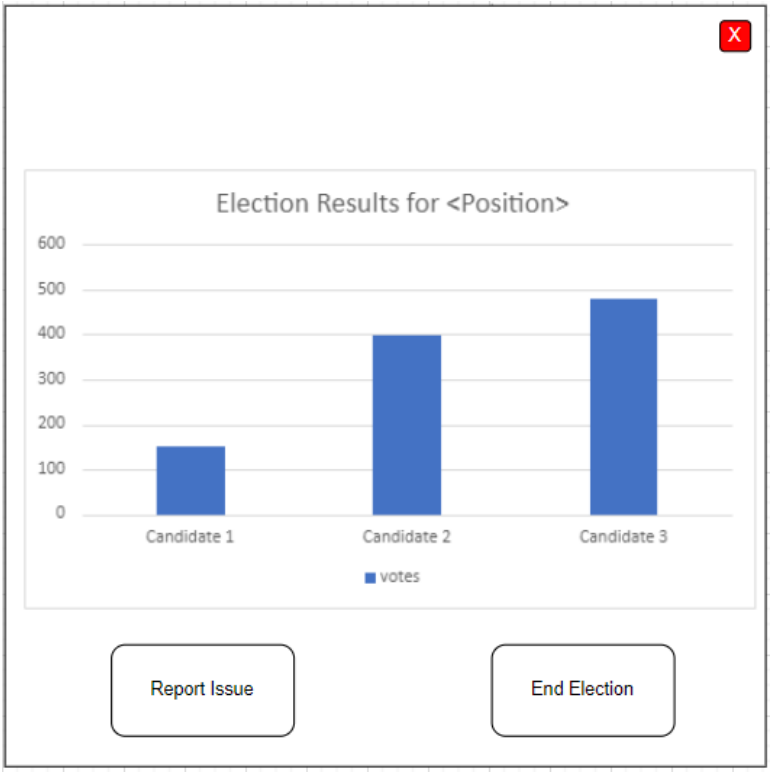
Election Results button pressed



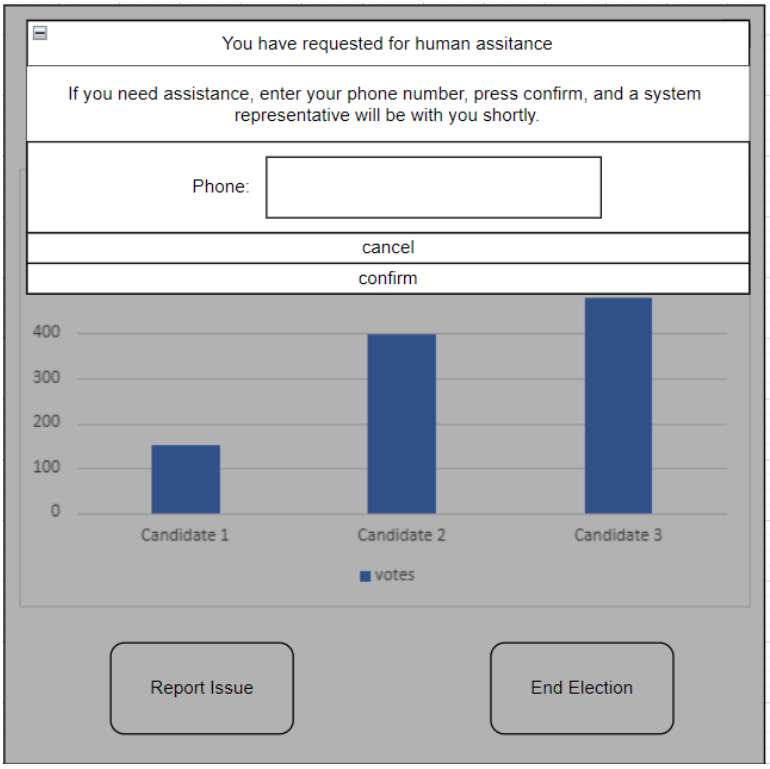
Loading finished, but voteCount or election is still ongoing.



Loading has finished and voteCount has finished counting votes

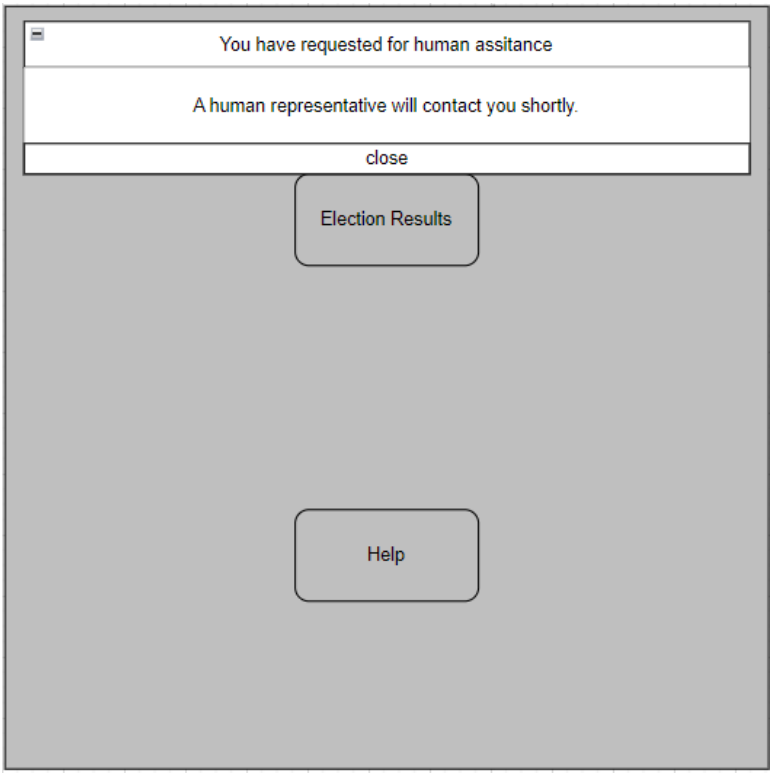


Report Issue from election screen

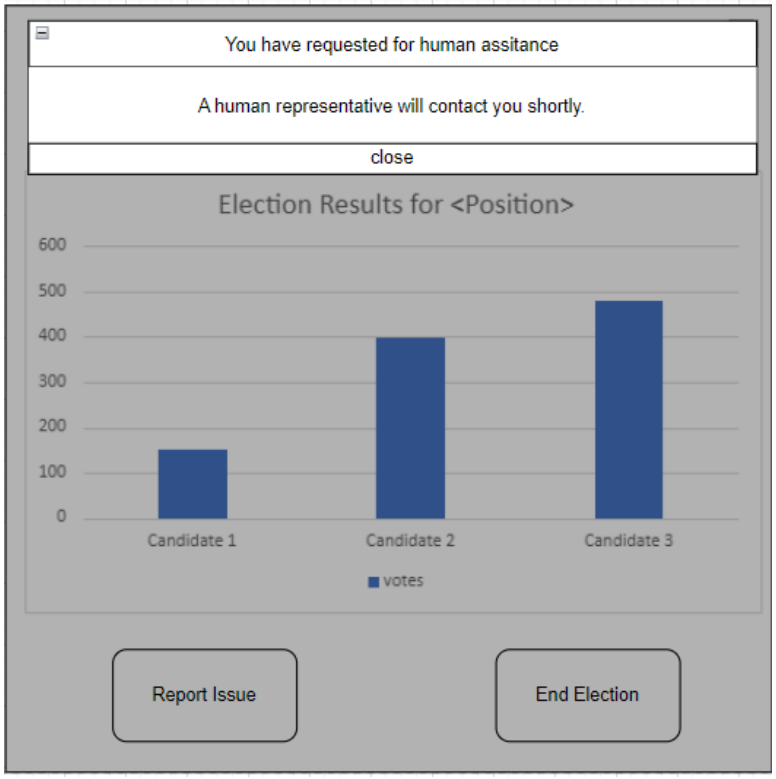


Help from default screen

Confirm on help window from default screen



Confirm on help window from election result screen



4. Glossary

Group 1- Man and the Muppets

1. FERPA - Family Educational Rights and Privacy Act

2. JSVS - Jaguar Secure Voting System

Group 2- Working on Sunday

1. User - Anyone who visits the website/application and is able to log in using their Jag Number and email address.
2. Candidate - User that has an approved candidacy request for an election.
3. Election Admin - User that has the powers of an administrator given by the client when it comes to creating and moderating an election within a department.
4. Jag Number - Unique identifier only available to students and faculty of the University of South Alabama.
5. CompSci - Computer Science.

Group 3- N-The-Process :

1. Voter - Students who are currently enrolled in the university and have a valid Jag ID.
2. Voting Session Dashboard - a user interface that displays all of the sessions the voter is interested in. Voter's interest is determined by majors and/or minors.
3. Voting Session - a user interface screen that shows the candidates and their information.
4. Manager - faculty or member in charge of a particular voting session.
5. Candidate - any option that is available to be selected for a particular voting session. Examples include yes, no, john clark, etc..
6. Registered Voter Database- a database that holds the voters that were registered using the voter registration system.
7. Voting Sessions Database- a database that maintains the information of the ongoing voting sessions.
8. Manager Records Database - a database that records the list of managers of the system according to a password and id.

Group 4 – Dream

1. User - student/faculty member that will be utilizing the system to vote
2. Candidate - individual that is running for a specific position
3. VotingApp - the mobile interface for voting
4. LoginPage- user interface that will display user's register/login information
5. VotingPage - user interface that will display candidates that the user will select from
6. SuccessPage - user interface that will display that the voter's ballot was successfully submitted
7. VoterDatabase - database that will contain the user's casted ballot
8. CandidateDatabase - database that will contain the list of candidates that the user will vote on
9. Connection - establishing authenticated connection to the databases with use of login/password of technician
10. Appstore - app store platform where users can purchase/download voting app

Group 5- ALAP

- tabulationSystem - the name of the whole of the system

- techSupport - human responsible for maintaining the system and dealing with issues from parties concerned with the election results and the tabulationSystem
- voteArbiter - the main user of the system; will elicit and receive the results of the election
- voterDatabase - the database that stores votes, created by the vote collection team
- tabulationInterface - the interface boundary object of tabulationSystem
- voteCounter - the subsystem control object responsible for counting votes and accessing the voteDatabase
- resultDatabase - database created by tabulationSystem that stores the results of the election