Deliverable 3

Team Members: Levi Hagan, Richard Marshall and Hannah Posch

Vision

Revision History

Version	Date	Description	Authors
Deliverable 1	Sep 09, 2018	First draft.	Hannah Posch Levi Hagan Richard Marshall
Deliverable 2	Sep 30, 2018	Second Draft	Hannah Posch Levi Hagan Richard Marshall
Deliverable 3	Oct 18, 2018	Third Draft	Hannah Posch Levi Hagan Richard Marshall
Deliverable 4	November 5, 2018	Fourth Draft	Hannah Posch Levi Hagan Richard Marshall

Introduction

We envision an electronic voting system with the reliability to cast and store citizens' votes, the usability to ensure those votes can be accounted for, while remaining secure and the software has the ability to register voters.

Positioning

Business Opportunity

Current voting systems are not entirely reliable in the storage of votes and the security of the system as seen with many recent elections that may have been hacked. This ineffective, non-democratic way of voting has created a need for a system that is reliable and accurate and provides the honest results of the election. Our electronic voting system is needed to rectify these problems previously mentioned as the system is built to remain secure and provide precise results.

Problem Statement

Traditional voting systems are subject to voter tampering, inaccuracy and corruption. This causes problems in voting results that can have alternative effects of government and local affairs

depending on the candidate elected. These issues do not reflect the will of the people and affect voters, staff, candidates, and even administrators.

Product Position Statement

The system is for any voter to be able to complete their civic duty and cast their vote, as well as, provide a secure, reliable system that prevents over-voting creating a much better experience while supporting its purpose: democracy.

Alternatives and Competition

One major alternative to our system is the traditional paper ballot system.

Stakeholder Descriptions

Market Demographics

Any citizen wishing to fulfill their civic duty in a democratic manner.

Stakeholder(Non-User) Summary

- Government: This stakeholder includes the state officials (or another level of official) that oversees the entire voting process and promotes the use of a democratic election.
- Voting Service: This stakeholder is the main business in charge of providing the government with a voting system and would be hired to complete the voting poll setup.
- Candidate: This stakeholder is the political candidate that is included on the electronic ballot.

User Summary

- Voter: This user includes the everyday citizen who wishes to cast their vote in the election.
- Administrator: This user is the designated person that has access to the unofficial results from the election.
- Staff: This user includes those who have been assigned the duty of helping voters and may need to operate the system depending on the scenario.

Key High-Level Goals

High-Level Goal	Priority	Problems and Concerns	Current Solutions
Fast, reliable secure vote submission	High	-Secure system not susceptible to hacking and voter fraud -System bugs that prevent accurate voting results or cause inability to vote -Reduced submission speed as more systems come online -Inability to authorize voter	Current voting systems supply basic voting needs, but do not prevent all problems described.
Easy to use interface,	High	-User interface is too complicated	Current voting

and able to locate voting results		and user is unable to complete task -Voting results are not provided when requested -Administrator not able to authenticate identity to access results	systems do meet most of these needs, however, if we could make our system more efficient and easy to use, then that is our current goal.

User-Level Goals

The users require a system that can meet these needs:

- Voter: cast vote, submit vote, authenticate identity
- Administrator: check unofficial vote tally
- Staff: Start up, shut down, restart system
- Electronic voting system: cast and count votes electronically

User Environment

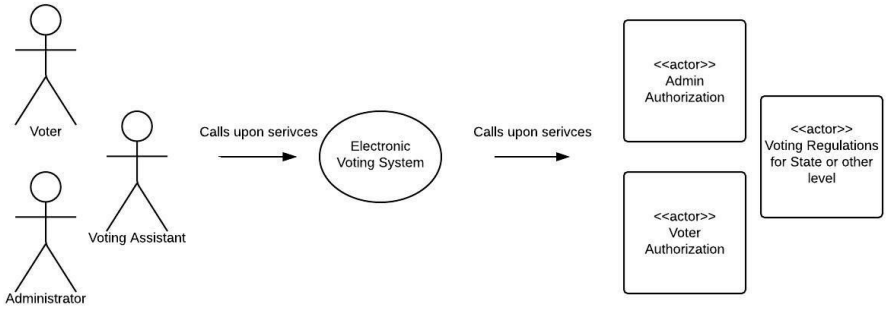
Voting poll location which will include machines with voting system installed.

Product Overview

Product Perspective

The electronic voting system will be located in a voting poll location. Due to the individual machines being used by the voter, the machines will be in close proximity to the network, if not hard wired to the network. It will provide a service to voters, and administrators as indicated by Figure Vision-1.

Figure Vision-1. Voting system context diagram.



Summary of Benefits

Supporting Feature	Stakeholder Benefit

The system will supply the necessary services to complete a vote, including submission of vote and unofficial tally.	Automated, and easy to use voting services.
Automatic voter and admin authorization will be required for all functionality.	Secure system that prevents over-voting and vote tampering.
Real-time voting submissions using state or specific voting level regulations	Reliable and regulated system that is comparable to any paper ballot system.

Summary of System Features

- Vote submission
- Voter authorization
- Admin authorization
- Real-time unofficial voting tally

Other Requirements and Constraints

See the Supplemental Specification and two use cases for additional details on design constraints, reliability, usability, performance, documentation, packaging, supportability and more.

Use Case UC1: Cast Vote

Scope: Voting Application

Level: User-goal

Primary Actor: Voter

Stakeholders and Interests:

-Voter: Wants accurate, fast voting experience with no system errors, and vote to be submitted correctly.

-Government: Wants to calculate votes correctly from every voting period. May include multiple agencies, such as national, state and local officials.

-Candidates: Want accurate, secure voting system with no errors in calculations.

-Voting service: Want reliable, fast voting system that works efficiently and is fair in results.

-Admin: Want an easy to use system that is secure and easy to locate results.

Preconditions: System is up and running with a voting period in place.

Success Guarantee (or Postconditions):

- Individual vote is saved, vote tally is correctly calculated

- System resets to handle next voter

Main Success Scenario (or Basic Flow):

1. Voter arrives at voting booth.
2. Voter logs in/enters identification.
3. System validates identification.
4. Voter chooses candidate.
5. Voter verified candidate selection.
6. System saves vote.
7. System verifies vote submission.
8. System saves voter identification.
9. System returns to main menu and is ready for next voter.

Extensions (or Alternative Flows):

*a. At any time, System fails:

1. System displays an error.

2. Voting attendant attempts to restarts system.
3. If system unable to start, voter completes vote on paper.

2a. Vote fails to input valid input identification information:

1. System Displays Error
2. Voter is found unable to vote

3a. Voter identification not valid:

1. System Displays Error
2. Voter is found unable to vote

3b. Voter has already voted:

1. System Displays Error
2. Voter is found unable to vote

4a. Voter chooses to write in a candidate:

1. Systems displays field to enter the name of the candidate.
2. Voter enters name of candidate.

5a. System unable to verify candidate selection:

1. System displays error.
2. Staff assists voter in filling out a paper ballot.

6a. System unable to save vote:

1. System displays error
2. Voting assistant assist voter in filling out paper ballot.

7a. System unable to save voter identification:

1. Voting attendants write down voter's identification by hand.

Special Requirements:

- UI on a large flat panel monitor. Text must be visible from 1 meter.
- UI color pallet must be color blind friendly
- Voter authorization response within 1 minute 90% of the time.
- System vote submission confirmation response within 1 minute 90% of the time.
- System must be secure.

Technology and Data Variations List:

- System must support keyboard and mouse or touch input.

Frequency of Occurrence:

- Nearly continuous during voting period.

Open Issues:

- Variations in Local, State, and Federal Elections?
- How to deal with a large number of candidates?
- How to handle voter verification?

Use Case UC2: Tallying Votes

Scope: Voting Application

Level: User-goal

Primary Actor: Admin

Stakeholders and Interests:

-Voter: Wants accurate, fast voting experience with no system errors, and vote to be submitted correctly.

-Government: Want to calculate votes correctly from every voting period. May include multiple agencies, such as national, state and local officials.

-Candidates: Want accurate, secure voting system with no errors in calculations.

-Voting service: Want reliable, fast voting system that works efficiently and is fair in results.

-Admin: Want an easy to use system that is secure and easy to locate results.

Preconditions:

- Voting period is closed.
- System is still running.
- Votes have been saved.

Success Guarantee (or Postconditions):

- System displays vote totals.

Main Success Scenario (or Basic Flow):

1. Administrator logs into system.
2. System verifies administrator's identity
3. Administrator selects voting results.
4. System displays voting results.

Extensions (or Alternative Flows):

*a. If at any time system fails:

1. System displays an error.

1a. Administrator is not able to log into system.

1. System displays an error.

2a. System unable to verify administrator identity.

1. System displays an error.

4a. System unable to display voter results.

1. System displays an error.

Special Requirements:

- UI on a large flat panel monitor. Text must be visible from 1 meter.
- UI color pallet must be color blind friendly
- Voting results response within 1 minute 90% of the time.
- Administrator confirmation response within 1 minute 90% of the time.
- System must be secure.

Technology and Data Variations List:

- System must support keyboard and mouse or touch input.

Frequency of Occurrence:

- At least once or more per voting period.

Open Issues:

- How to handle administration log in?
- How to address incorrect results?

Use Case UC3: Register

Scope: Voting Application

Level: User-goal

Primary Actor: Voter

Stakeholders and Interests:

-Voter: Wants accurate, fast registration experience with no system errors, and forms to be submitted correctly.

-Government: Wants to calculate the total number of voters. May include multiple agencies, such as national, state and local officials.

-Candidates: Want accurate, secure voting registration system with no errors in calculations.

-Voting registration service: Want reliable, fast voting system that works efficiently and is fair in results.

Preconditions: System is up and running with a registration period in place.

Success Guarantee (or Postconditions):

- Individual voter is saved, voter list is correctly calculated, and identity is verified.
- System resets to handle next voter

Main Success Scenario (or Basic Flow):

1. Voter arrives at registration booth.
2. Voter logs in/enters identification.
3. System validates identification.
4. Voter enters personal information.
5. Voter verifies personal information.
6. Voter submits personal information.
7. System saves voter information.
8. System returns to main menu and is ready for next voter.

Extensions (or Alternative Flows):

*a. At any time, System fails:

1. System displays an error.
2. Staff attempts to restarts system.

3. If system unable to start, voter completes vote on paper.

2a. Vote fails to input valid input identification information:

1. System Displays Error
2. Voter is found unable to register

3a. Voter identification not valid:

1. System Displays Error
2. Voter is found unable to register

3b. Voter has already registered:

1. System Displays Error
2. Voter is found unable to register

6a. Voter unable to submit personal information:

1. System displays error.
2. Staff assists voter in filling out an alternate registration

7a. System unable to save voter:

1. System displays error
2. Staff assist voter in filling out an alternate registration

Special Requirements:

- UI on a large flat panel monitor. Text must be visible from 1 meter.
- UI color pallet must be color blind friendly
- Voter verification response within 1 minute 90% of the time.
- System voter submission confirmation response within 1 minute 90% of the time.
- System must be secure.

Technology and Data Variations List:

- System must support keyboard and mouse or touch input.

Frequency of Occurrence:

- Nearly continuous during registration period.

Open Issues:

- Variations in Local, State, and Federal Elections?
- How to switch between registering and voting?
- How to handle voter verification?

Use Case UC4: Voter Identification

Scope: Voting Application

Level: User-goal

Primary Actor: Voter

Stakeholders and Interests:

-Voter: Wants accurate, fast voting experience with no system errors, and vote to be submitted correctly.

-Government: Wants to calculate votes correctly from every voting period. May include multiple agencies, such as national, state and local officials.

-Candidates: Want accurate, secure voting system with no errors in calculations.

-Voting service: Want reliable, fast voting system that works efficiently and is fair in results.

-Admin: Want an easy to use system that is secure and easy to locate results.

Preconditions: System is up and running with a voting period in place.

Success Guarantee (or Postconditions):

- Voter confirmation message displayed.

- System displays ballot or registration portal to voter.

Main Success Scenario (or Basic Flow):

1. Voter arrives at voting booth.
2. Voter logs in/enters identification.
3. System validates identification.
4. System displays confirmation window.
5. System displays next screen.

Extensions (or Alternative Flows):

*a. At any time, System fails:

1. System displays an error.
2. Staff attempts to restarts system.
3. If system unable to start, staff verifies voter.

2a. Voter fails to input valid input identification information:

1. System Displays Error

2. Voter is unable to verify identity.

3a. Voter identification not valid:

1. System Displays Error
2. Voter is found unable to log in

Special Requirements:

- UI on a large flat panel monitor. Text must be visible from 1 meter.
- UI color pallet must be color blind friendly
- Voter authorization response within 1 minute 90% of the time.
- System must be secure.

Technology and Data Variations List:

- System must support keyboard and mouse or touch input.

Frequency of Occurrence:

- Nearly continuous during voting period.

Open Issues:

- Variations in Local, State, and Federal Elections?
- How to handle voter verification?

Use Case UC5: Admin Identification

Scope: Voting Application

Level: User-goal

Primary Actor: Admin

Stakeholders and Interests:

-Voter: Wants accurate, fast voting experience with no system errors, and vote to be submitted correctly.

-Government: Wants to calculate votes correctly from every voting period. May include multiple agencies, such as national, state and local officials.

-Candidates: Want accurate, secure voting system with no errors in calculations.

-Voting service: Want reliable, fast voting system that works efficiently and is fair in results.

-Admin: Want an easy to use system that is secure and easy to locate results.

Preconditions: System is up and running with a voting period in place.

Success Guarantee (or Postconditions):

- Admin confirmation message displayed.

- Admin portal displayed

Main Success Scenario (or Basic Flow):

1. Admin arrives at system.
2. Admin selects the Admin login button.
3. Admin logs in/enters identification.
4. System validates identification.
5. System displays confirmation window.
6. System displays admin portal.

Extensions (or Alternative Flows):

*a. At any time, System fails:

1. System displays an error.
2. Staff attempts to restarts system.
3. If system unable to start, staff verifies voter.

3a. Admin fails to input valid input identification information:

1. System Displays Error
2. Admin is unable to verify identity.

4a. Admin identification not valid:

1. System Displays Error
2. Admin is found unable to log in

Special Requirements:

- UI on a large flat panel monitor. Text must be visible from 1 meter.
- UI color pallet must be color blind friendly
- Admin authorization response within 1 minute 90% of the time.
- System must be secure.

Technology and Data Variations List:

- System must support keyboard and mouse or touch input.

Frequency of Occurrence:

- As frequently as needed during voting period.

Open Issues:

- Variations in Local, State, and Federal Elections?
- How to handle Admin verification?

Supplemental Specification

Introduction

This document is the repository of all voting system requirements not captured in the use cases.

Functionality

- ***Error Handling:*** All errors will be followed by a message to the voter, staff or administrator.
- ***Security:*** All usage requires identification and authorization to use the system.

Usability

- ***Human Factors:*** The voter will be able to see a large-monitor display of the voting system within the voting booth.
    - Avoid a color palette associated with most forms of color blindness.
    - System text should also be visible from 1 meter away.
- Speed, accuracy, and reliability are monumental in a voting system, as the voter wants to cast their vote quickly and ensure that is was submitted correctly.
- Message boxes signaling errors should be used to ensure that if a voter or administrator is using the system, they can be made aware of the error.

Reliability

- ***Recoverability:*** If the system begins to malfunction, try to solve it with a paper solution (cast vote on paper), or alert staff to check the system. It is imperative that the staff is given instruction via the system of how to check and reboot system if needed. More analysis will be needed in this portion as the system is developed.

Performance

- Voters want to submit their votes quickly, however, one issue that may slow this down is voter authorization. Our goal as mentioned in the use cases is to have the voter authorization within 1 minute, 90% of the time.
- The administrator also want to access the voting results quickly, as well as, ensure that the votes have been counted accurately. Another goal we have is to have the administrator authorization within 1 minute, 90% of the time. We would also like the system to display the voting results to the administrator within 1 minute, 90% of the time.

Supportability

- ***Adaptability:*** Although most of the voting scenarios should be similar because each voter is simply casting our vote, we do want our system to remain adaptable to any issues that may arise. Also we want our system to allow write in candidates which will require the system to adapt as it counts votes.
- ***Configurability:*** Our voting system will need to be configurable to the voting election type it is being used for (local, state, federal). This support mechanism allows the system to be useful in a variety of scenarios, and apply the correct voting rules and vote calculations when necessary. This portion will need more analysis as we begin the design and development of our system.

Implementation Constraints (Hardware and Software)

- Our team will be using Java and JavaFX as the base of our system, as our team is familiar with these chosen software constraints which will help the development of our project run smoother.

Purchased Components

- During this phase of development, we do not have any purchased components for our system.

Free Open Source Components

- At this time, we don't have definite components in mind, but we would also like to use our own developed components as much as possible due to the nature of the project.
- Since the security of this project remains at the forefront, we would also be weary to use free open source code with known security issues as it could lead to an unsecure system.

Documentation

- ***Voter Instructions:*** Voters will be provided with instructions on how to verify their identity before continuing and whether their ability to continue voting is valid. Voters will also be given clear instruction on how to cast their vote throughout the system. The voter will also be asked to confirm their vote before submitting.
- ***Administrator Instructions:*** The administrator will be given instruction on how to login to the system, as well as, complete an unofficial tally at the specific precinct.
- ***Help:*** In the event of system failure, instructions will be provided to the staff on ways to troubleshoot or reboot the system if necessary.

Internationalization concerns

- Due to the nature of the voting system, we remain confident that our system is capable of being internationalized. To ensure the system is effective in another company, the voting rules would need to be changed and the written instruction throughout the UI would need to be translated. However these changes would not be extremely difficult and allow the developer to transition our system to another location.

Interfaces

Hardware Interfaces

- Touch screen monitor

Software Interfaces

- At the time we are not using any software interfaces (this is subject to change)

Risks

Hardware

- Software may be deployed on a variety of hardware depending on the client which may lead to compatibility and security issues

Time

- Production time
- Run time

- Time constraints based on the election

Labor

- Voter assistant and Government Administrators are required for this system operate as intended

Conversions

- Different types of elections require different numbers of candidates and election types

Legal Issues

- Inaccurate results could cause severe issues
- Software must be compliant with all voting laws which vary from place to place

Application-Specific Domain(Business) Rules

Rule Number	**Rule**	**Changeability**	**Source**
Rule 1	Proper Identification must be used and be able to be verified	States voter ID laws change as newer forms of identification are produced such as SC's new "REAL ID"	Law
Rule 2	Time constraints	Some States Require voting booths to shut off after the election period is over and not allow new voters.	Law

Information in Domains of Interest

Voter identification

- With recent events voter fraud is a big issue so our team must have an effective system to identify voters and prevent voter fraud. Because different states require different levels of identification to vote our system must be able to recognize all of them, such as driver's license numbers, social security numbers, or green cards for California.

Reports

- Inception document created.
- UI mockup created

Concerns (Environment and Operational)

- Must use JavaFX (requirement)
- Must use Java (team familiarity)

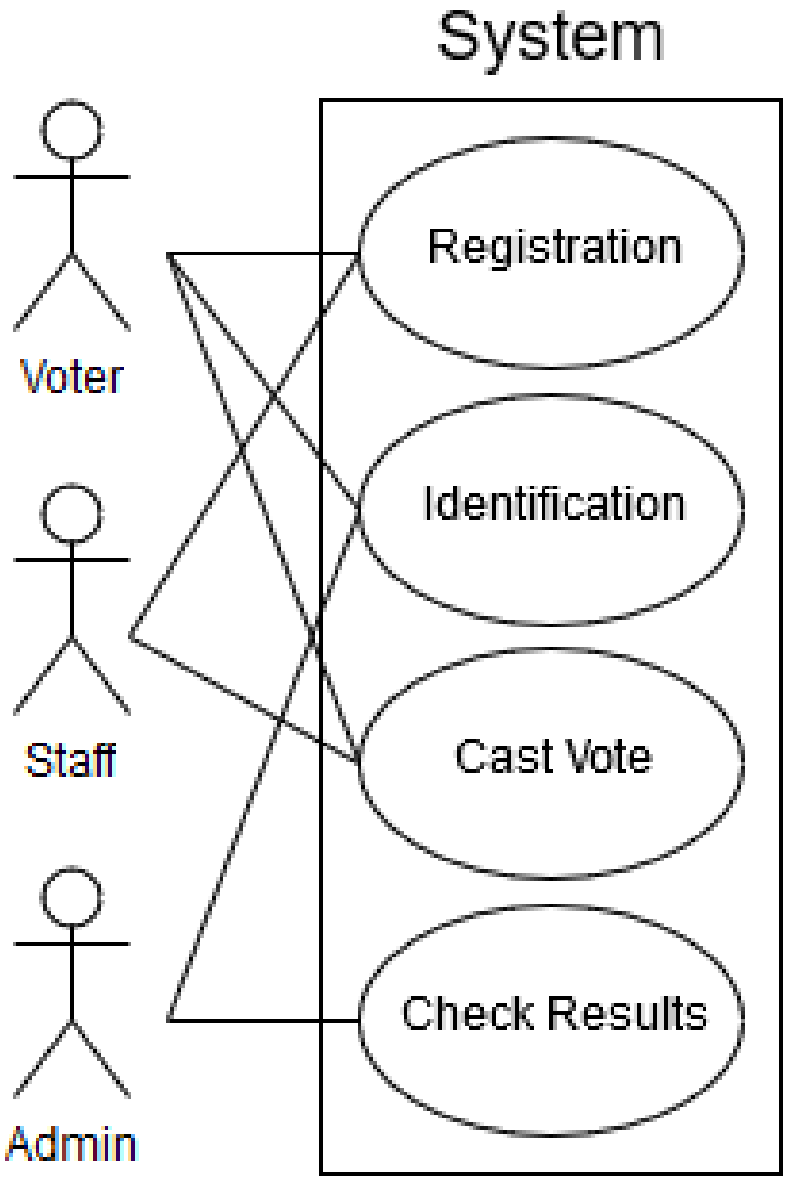
Glossary

Definitions

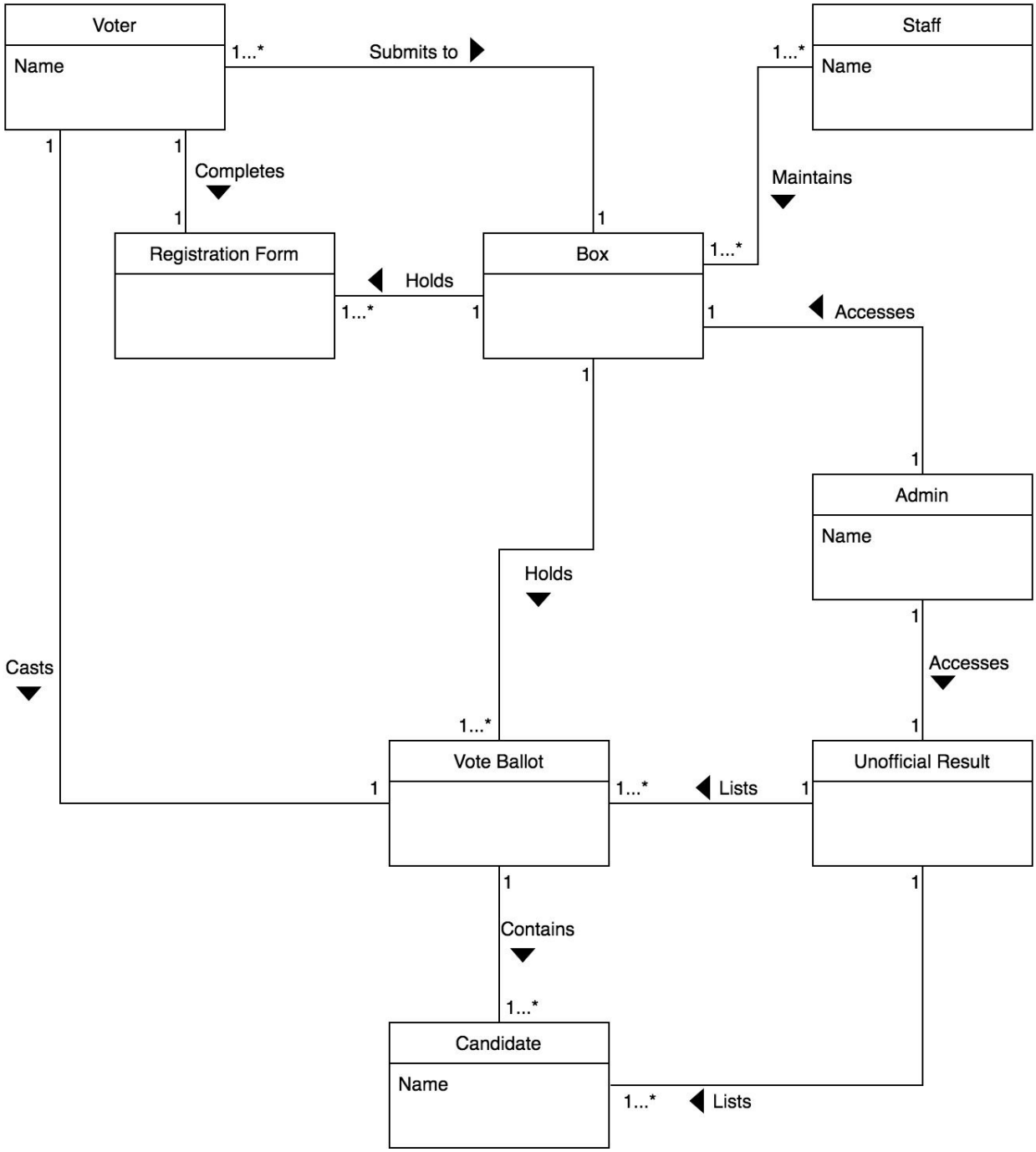
Term	Definition and Information	Format	Validation Rules	Aliases
System	**The voting application.**	**N/A**	**N/A**	**None**
Voter	**Person casting a vote that will have a voter account tied to their identification which allows them to cast their vote for a specific candidate once in each voting period. Elements include a name, voter ID, and social security number.**	**N/A**	**N/A**	**None**
Staff	**Staff at the election facility who assists the voter.**	**N/A**	**N/A**	**None**
Administrator	**The person in charge of checking the results from the system, with a special admin account which allows them to check the results of a voting period on the machine, as well as to begin and end a voting period.**	**N/A**	**N/A**	**Admin**
Voting Period	**Time in which the election occurs and the system takes votes.**	**N/A**	**N/A**	**None**
Administrator Authorization	**Validation by an external system to ensure that the admin**	**N/A**	**N/A**	**Admin Authorization**

	has permission to view the voting results.			
Voter Authorization	**Validation within the system to ensure the voter is not a duplicate voter.**	**N/A**	**N/A**	**None**

Use Case UML Diagram



Domain Model Diagram



Domain Model Diagram Justification

For this domain model, we focused on the main objects that would be needed to create an electronic voting system. So the hub of the design is the box object, as this is the primary object all actors will interact with and where most of the actions take place. We isolated the voter, staff, and admin as individual components as they each have their own unique associations. The voter will login in, complete a registration form, as well as, cast their vote. The staff is also connected in our model, as they provide assist with voting and use of box. The admin will also access the unofficial result. The vote ballot, registration form and candidate were also isolated in our model and highlight the main features of our system which include the ability to register, and also cast a single vote. The domain model design choices were made on simplicity, usability and highlighting the main scope of the project.

Software Architecture Document

Architectural Representation

This Software Architecture Document (SAD) shows the overall architecture of our voting system from multiple views. These include:

- Sequence view
- System Sequence view
- Operation Contracts view

In addition, this Software Architecture Document includes architecturally significant requirements recorded in a table. We have also included technical memos to summarize key architectural decisions and motivations.

Architectural Factors

Factor	**Measures and quality scenarios**	**Variability (current flexibility and future evolution)**	**Impact of factor(and its variability) on stakeholders, architecture, and other factors**	**Priority for Success**	**Difficulty or Risk**
Reliability - Recoverability					
Recovery from a remote voting machine service fault	When a voting machine fails, re-establish connectivity with it within 3 minutes of its detected re-availability, under normal voting session environment	Current flexibility- our current voting services are acceptable and will be able to restart when necessary. Evolution- within 4 years, some voting services may invest in more reliable remote machines Probability? High.	Large impact on voting system design and could affect vote count if failure occurs. Voters and voting service officials do not like when a remote voting machine fails, as they are not able to submit or count votes.	H	M
Supportability-Adaptability					
Support	When a new	Current flexibility-	Required for system	H	M

different types of elections and adapt based on qualities.	election is created, the parameters will also be added and a new election should be generated within a few minutes.	Election type must remain flexible and our current system will need to adapt. Evolution - none	functional requirements and system acceptance. Medium impact on design.		
Other - Legal					
Current election rules must apply	When the government and admin evaluate the voting results, 100% compliance is required.	Current flexibility- The use of current election rules will need to remain inflexible. Evolution - noneq	Failure to meet standard could cause incorrect election results or cause an unfair election.	H	M

Architectural Decisions (Technical Memos)

Technical Memo

Issue: Reliability-Recovery from a remote voting machine service fault

Solution Summary: Restart voting machine and voting system software, and set system back online after restart.

Factors

- Robust recovery from a voting machine failure.

Solution

Voting staff must have the ability to restart system either remotely or through the physical voting machine. Once restarted, the staff will need the ability to reset the voting service on that particular machine to start allowing the submission of votes again. This restart, however, will need to still allow the machine to keep track of votes submitted already on system and ensure that any future votes submitted are talloyed correctly.

Motivation

Voters on election day need to be able to submit their vote quickly and accurately. Voting service failure is not an option as election periods are short and require efficiency. Voting service organizations also want votes to be counted correctly to ensure proper voting results are

recorded. Without all machines working correctly, the election results may not include all votes in the results which could cause additional problems.

Unresolved Issues - none

Alternatives Considered

The only additional solution to this problem could include casting votes on another machine or having the voters complete paper ballots.

Technical Memo

Issue: Supportability-Support different types of elections and adapt based on qualities

Solution Summary: Create a system that can adapt and build an election based on certain rules and parameters given.

Factors

- Robust adaptability to form an election from standards provided.

Solution

The voting system will need to include parameters for the voting service to adjust as needed per given election. These parameters may include election timeline, number of candidates or type of candidates for the given election. Once selected, a new election will need to be generated with these parameters applied to ensure a proper election takes place.

Motivation

Not every election is the same which requires the system to be able to adapt upon the given conditions. Without this support, the system would not be very useful in the scope of current elections.

Unresolved Issues - none

Alternatives Considered - none

Technical Memo

Issue: Legal-Current election rules must apply

Solution Summary: Create a system that maintains integrity and follows a set of defined rules from the voting service and government.

Factors

- Accuracy and integrity in maintaining authorized election rules.

Solution

The voting system will need to include standards for the voting service to adjust as needed per given election. These standards may include the current state and national rules and laws for a given election to prevent an illegitimate election.

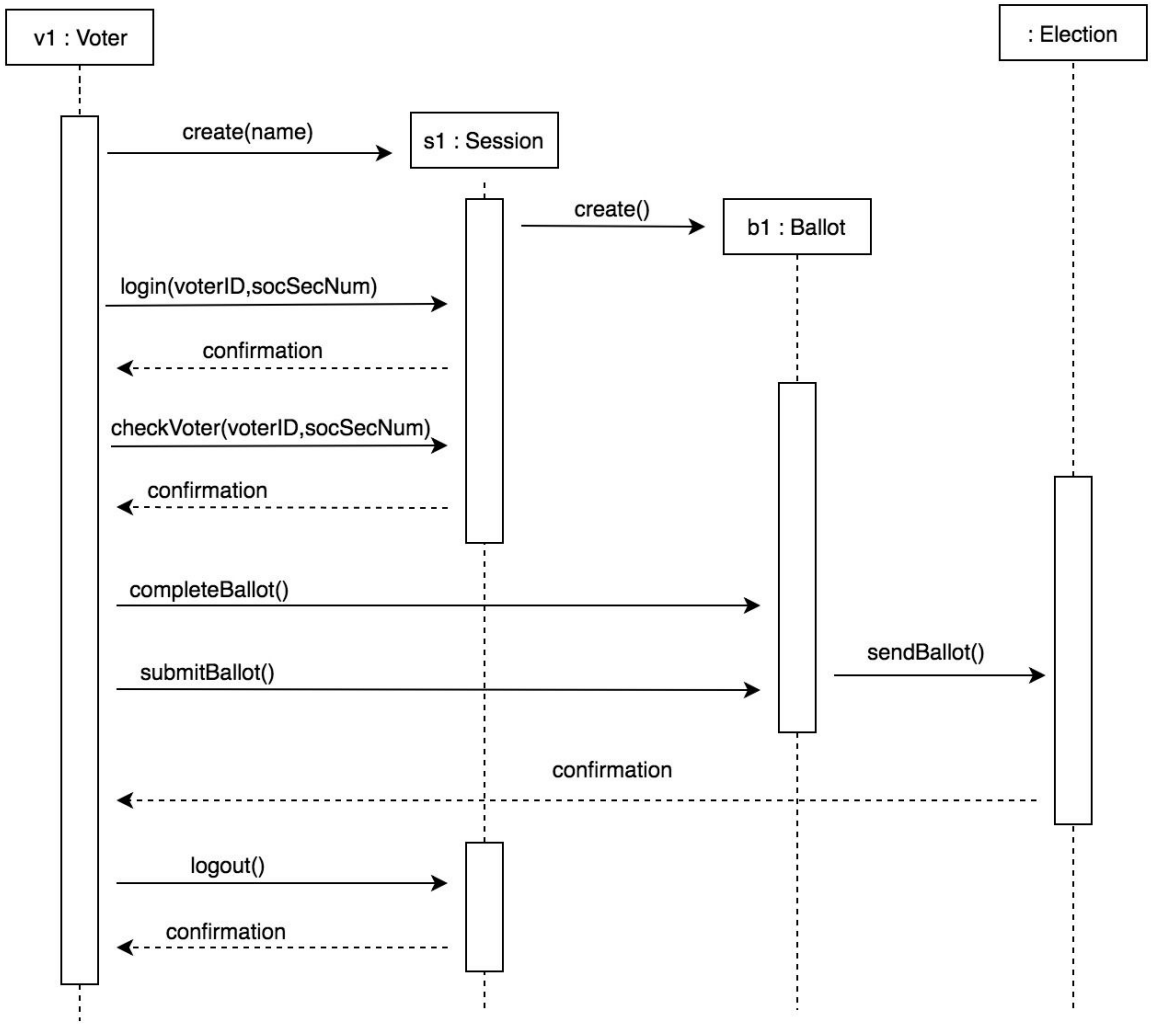
Motivation

Voting is a right and thus needs to be respected and maintained by the standards of the voting system. Certain rules applied to an election allow fairness and a democratic election to take place.

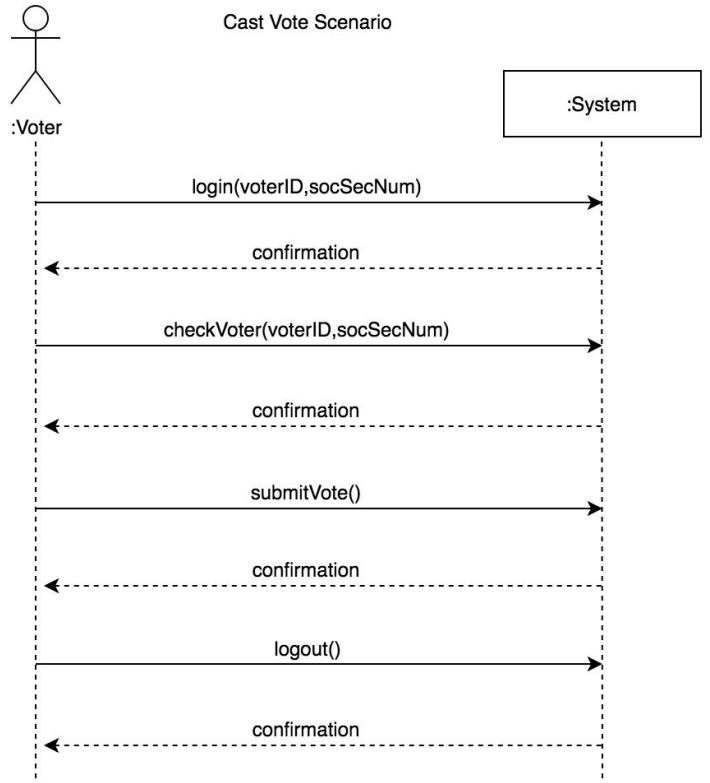
Unresolved Issues - none

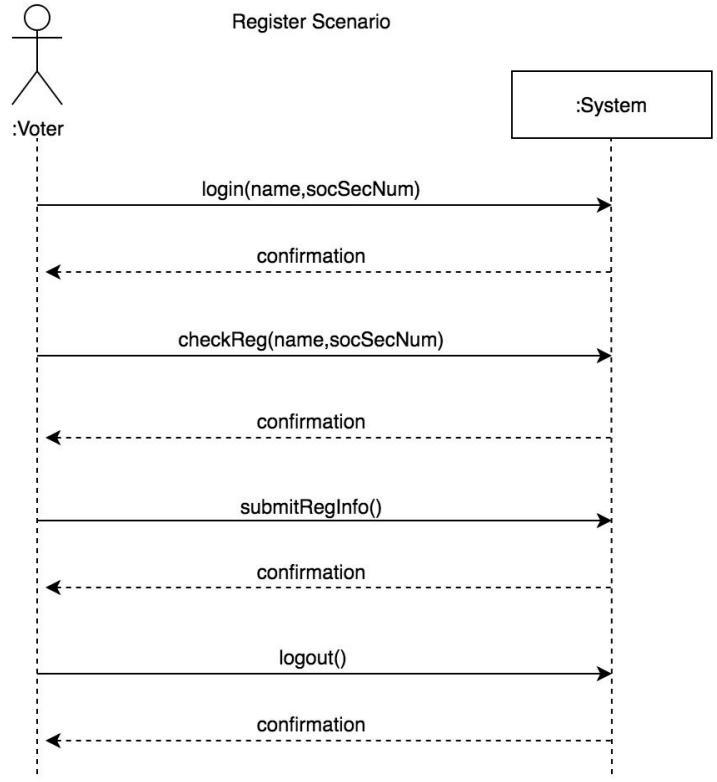
Alternatives Considered - none

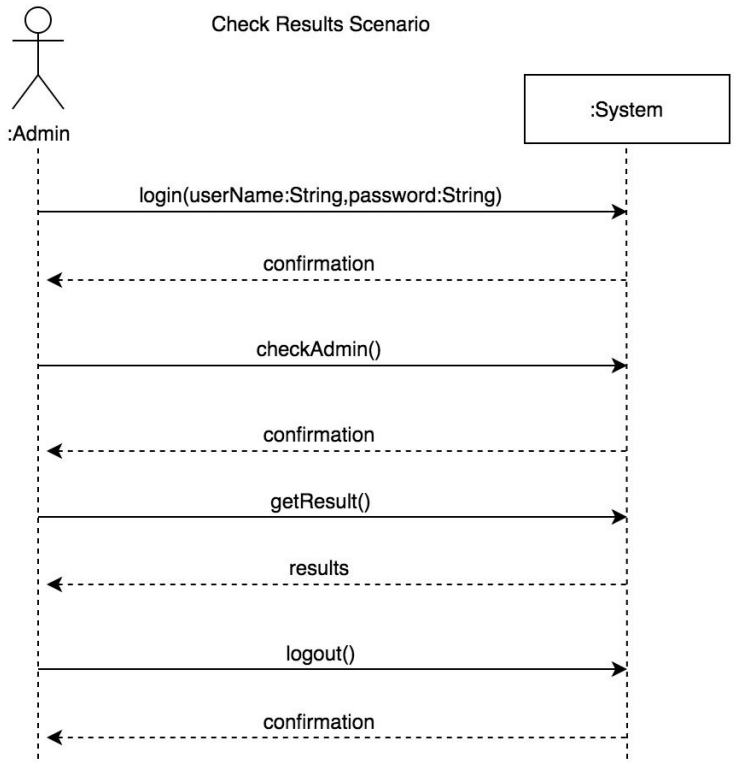
Sequence View



System Sequence View






Check Results Scenario
:Admin
:System
login(userName:String,password:String)
confirmation
checkAdmin()
confirmation
getResult()
results
logout()
confirmation

Operation Contracts View

Contract CO1: checkVoter

Operation:	checkVoter(voterID: String, socSecNum: String)
Cross References:	Use Cases: Identification
Preconditions:	There is an election underway.
Postconditions:	- A voter instance v1 was created (instance creation). - v1 was verified by searching voterList array. - Attributes of v1 were initialized. - v1 was associated with the current Election (association formed).

Contract CO2: checkAdmin

Operation:	checkAdmin(name: String, adminID: String)
Cross References:	Use Cases: Identification
Preconditions:	There is an election underway.
Postconditions:	- An admin instance a1 was created (instance creation). - a1 was verified by searching adminList array. - a1 was associated with the current Election (association formed).

Contract CO3: checkReg

Operation:	checkReg(name: String, socSecNum: String)
Cross References:	Use Cases: Registration
Preconditions:	There is a registration period underway.
Postconditions:	- A registration instance r1 was created (instance creation). - r1 was verified by searching regList array. - Attributes of r1 were initialized.

Contract CO4: addCandidate

Operation:	addCandidate(name: String, party: String)
Cross References:	Use Cases: Cast Vote
Preconditions:	There is an election underway.
Postconditions:	- A candidate instance c1 was created (instance creation). - c1 was added to votesCounted array. - c1 was associated with the current Election (association formed).

Contract CO5: getResult

Operation:	getResult()
Cross References:	Use Cases: Check Results
Preconditions:	The election is either underway or complete.
Postconditions:	-A Result instance res1 was created (instance creation). -res1 was associated with the current Election (association formed).

Contract CO6: endElection

Operation:	endElection()
Cross References:	Use Cases: Check Results
Preconditions:	The election is complete.
Postconditions:	-Election.isComplete became true (attribute modification).

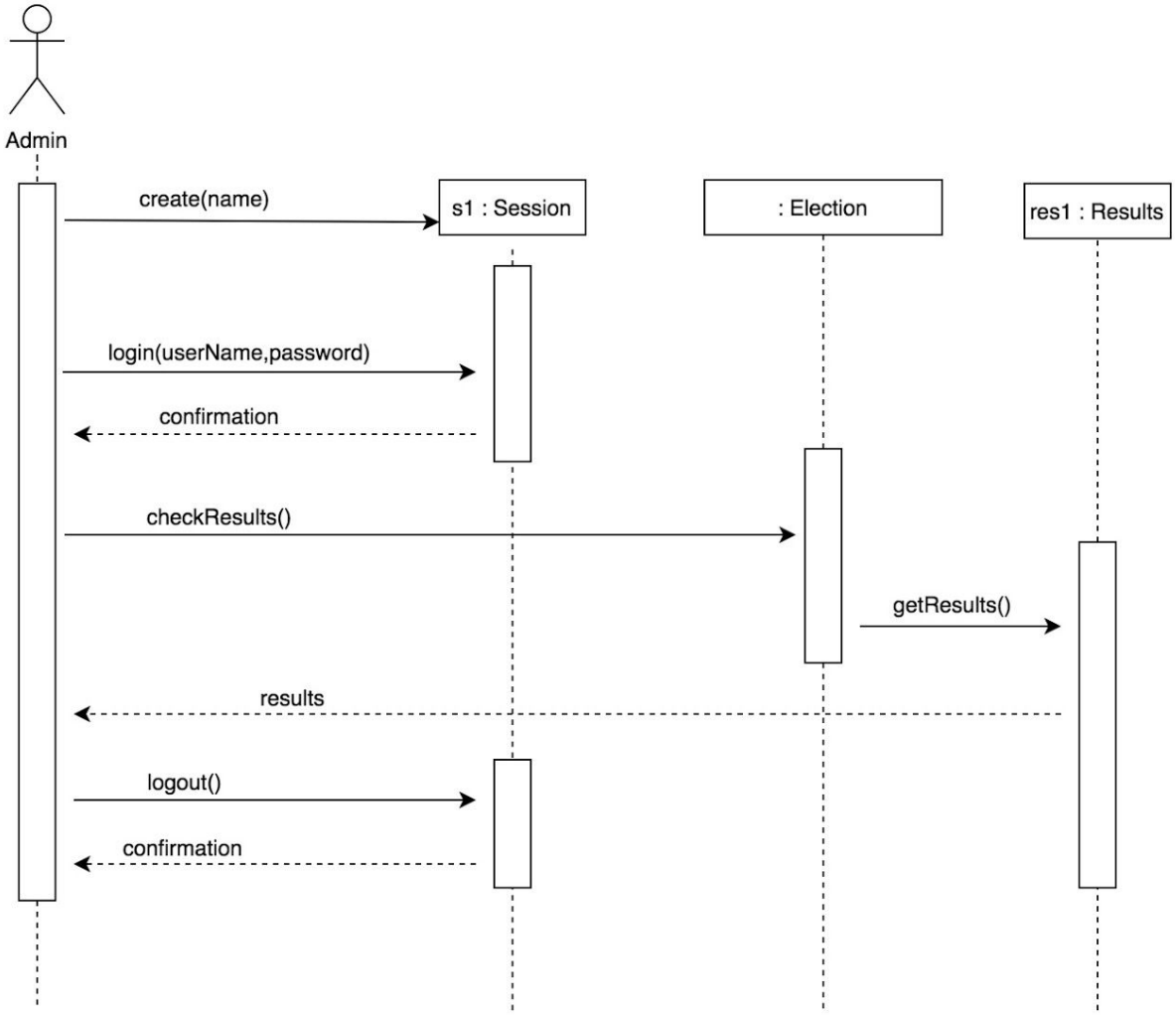
Contract CO7: login

Operation:	login(voterID:String, socSecNum: String)
Cross References:	Use Cases: Cast Vote
Preconditions:	The election is underway.
Postconditions:	- A voter instance v1 was created (instance creation).

Contract CO8: startElection

Operation:	startElection(name:String)
Cross References:	Use Cases: Check Results
Preconditions:	The election is not underway.
Postconditions:	-An election instance e1 was created (instance creation). -Attributes of e1 were initialized.

Sequence Diagram - Check Results Scenario



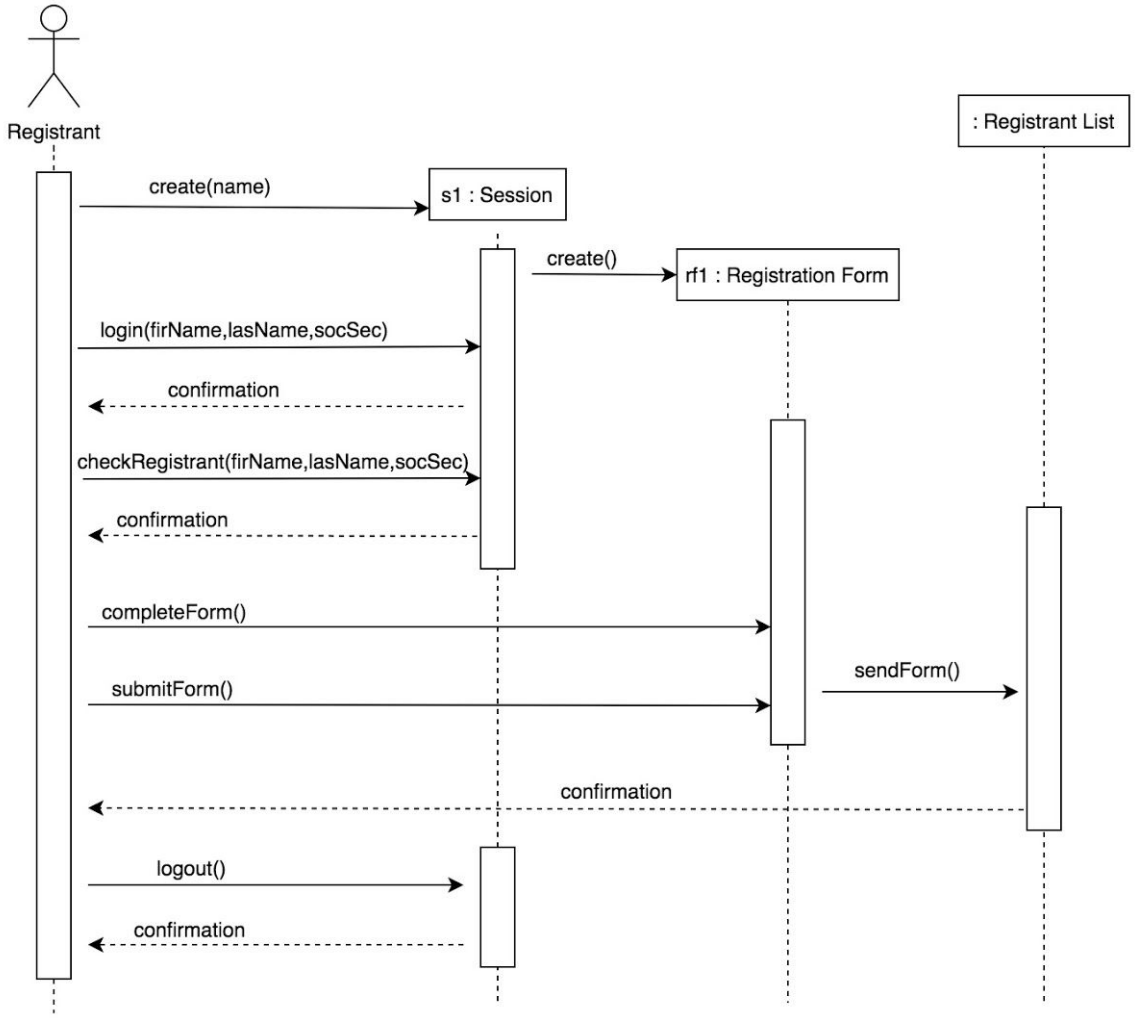
GRASP

- create(name) : Creator
- login(userName,password) : Information Expert
- checkResults() - HighCohesion
- getResults() - Controller
- logout() : High Cohesion

Justification

For our main check results scenario, we divided the actions among 4 main objects the Admin, Session, Election, and Results. The Admin creates a new session to check results. After the Admin logs in, the Admin will check the results. The election will then get the results from the Results class. After the results are returned, the Admin will log out.

Sequence Diagram - Register Scenario

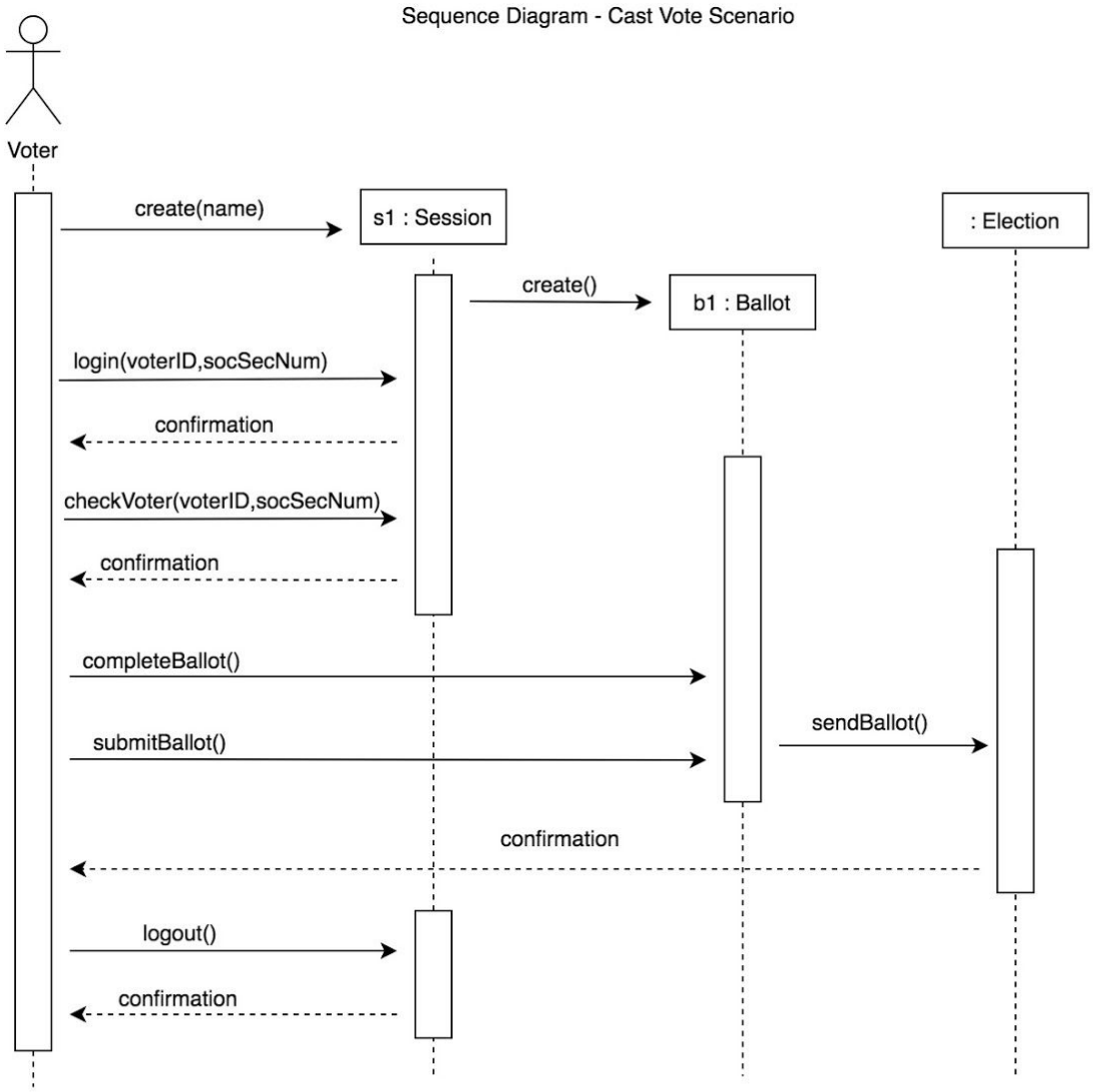


GRASP

- create(name) : Creator
- create() : Creator
- login(firName,lasName,socSecNum) : Information Expert
- checkRegistrant(firName,lasName,socSecNum) : Information Expert
- completeForm() : Low Coupling
- submitForm() : Controller
- sendForm() : Controller
- logout() : High Cohesion

Justification

For our main register scenario, we divided the actions among 4 main objects the Registrant, Session, Registration Form, and Registrant List. The Registrant creates a new session which instantiates a form. After the Registrant logs in and has their identity confirmed, the Registrant completes and submits their form. The form is then sent to the Registrant List and the Registrant will finally log out. These four main classes were chosen based on the principle of Low Coupling and High Cohesion. These classes also allow for integration of objects in our other main scenarios.

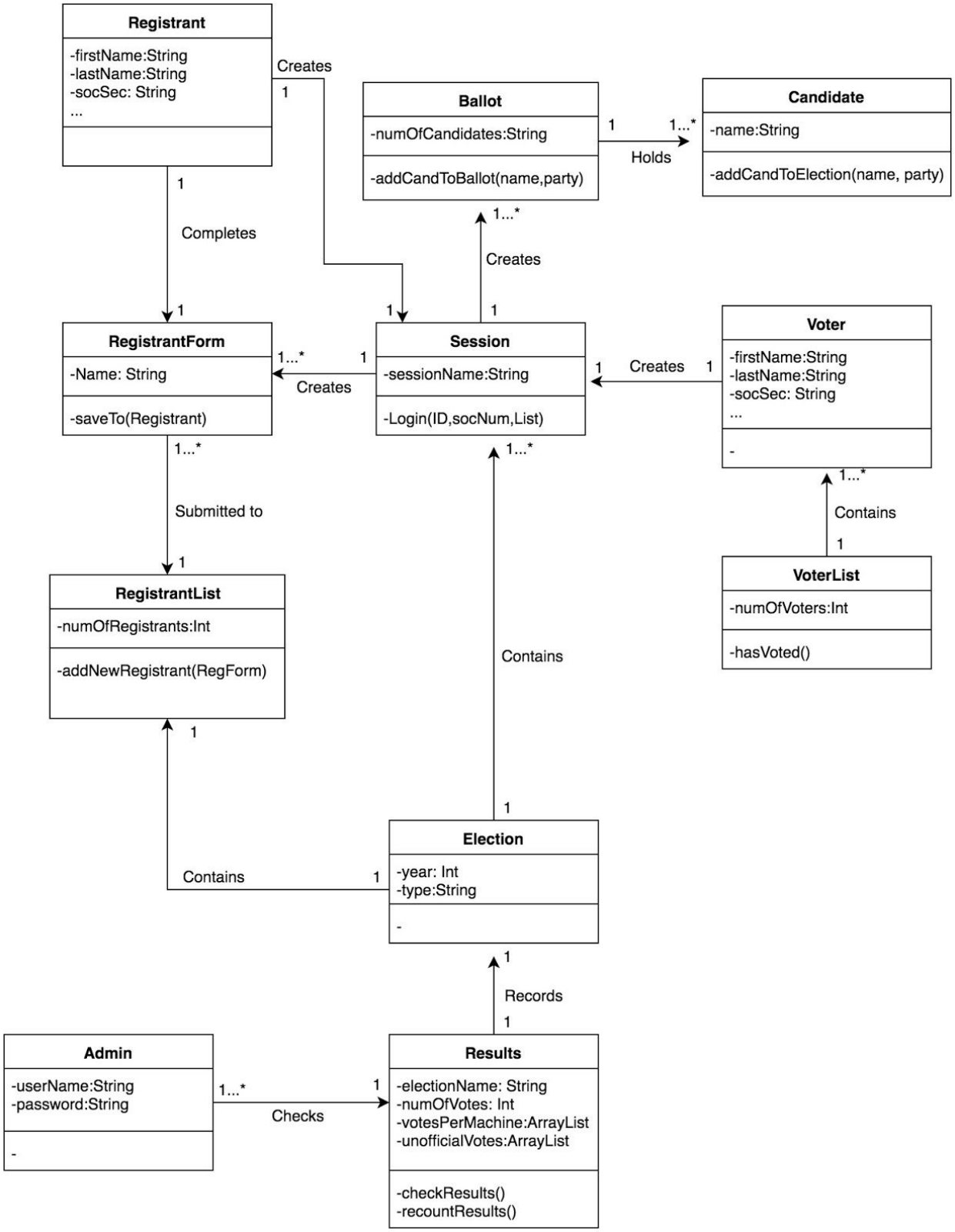


GRASP

- create(name) : Creator
- create() : Creator
- login(voterID,socSecNum) : Information Expert
- checkVoter(voterID, socSecNum) : Information Expert
- completeBallot() : Low Coupling
- submitBallot() : Controller
- sendBallot() : Controller
- logout() : High Cohesion

Justification

For our main cast vote scenario, we divided the actions among 4 main objects the Voter, Session, Ballot, and Election. The Voter creates a new session which instantiates a ballot. After the Voter logs in and has their identity confirmed, the voter completes and submits their ballot. The ballot is then sent to the Election and the voter will finally log out. These four main classes were chosen based on the principle of Low Coupling and High Cohesion. These classes also allow for integration of objects in our other main scenarios.


Registrant
-firstName:String
-lastName:String
-socSec: String
...
Creates
1
Ballot
-numOfCandidates:String
-addCandToBallot(name,party)
1
1...*
Holds
Candidate
-name:String
-addCandToElection(name, party)
1
Completes
1
1...*
Creates
1
1
RegistrantForm
-Name: String
-saveTo(Registrant)
1...*
Creates
1
Session
-sessionName:String
-Login(ID,socNum,List)
1
Creates
1
Voter
-firstName:String
-lastName:String
-socSec: String
...
-
1...*
Submitted to
1
1...*
1...*
Contains
1
RegistrantList
-numOfRegistrants:Int
-addNewRegistrant(RegForm)
VoterList
-numOfVoters:Int
-hasVoted()
1
Contains
1
Election
-year: Int
-type:String
-
Contains
1
1
Records
1
Admin
-userName:String
-password:String
-
1...*
Checks
1
Results
-electionName: String
-numOfVotes: Int
-votesPerMachine:ArrayList
-unofficialVotes:ArrayList
-checkResults()
-recountResults()