PBI 1	Ballot Re-assignment for STV
Task Description	Ensure correct ballot assignment for candidate election or elimination in STV.
Testing Number	PBI1_System_tests_01
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/stv_ballots_4_candidates.csv number of seats: 2 Select algorithm (STV or Plurality): STV Enable ballot shuffle? (true/false): false audit file name: audit_stv.txt
Tests	1. Test when a candidate is elected and surplus votes are transferred correctly. 2. Test when a candidate is eliminated and ballots are reassigned to next preferred candidates.
Outputs	Final winners reflect the correct application of STV rules. All ballots are counted and reassigned properly. Correct selection of the top 2 candidates (those meeting the quota). Output shown below: ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10 Number of Seats: 2 Number of Candidates: 4 Droop Quota: 4 votes Winners: C | Votes: 4 | Percentage: 40.00% | Met Quota: Yes B | Votes: 5 | Percentage: 50.00% | Met Quota: Yes Losers: A | Votes: 3 | Percentage: 30.00% D | Votes: 0 | Percentage: 0.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass
Date	4/18/2025

PBI 1	Ballot Re-assignment for STV
Task Description	Validate STV election behavior for finding candidate with fewest votes.
Testing Number	PBI1_Unit_tests_01
Author	Anwesha Samaddar
Inputs	Automated testing suite using test function: TEST_F(STVTests, FindLowestCandidateTest)
Tests	1. Tests if function findLowestCandidate() correctly identifies the candidate with lowest votes 2. Tests if it returns non-null pointer for valid input 3. Tests if it correctly identifies candidate C as having fewest votes (0 votes)
Outputs	6: Note: Google Test filter = STVTests.FindLowestCandidateTest 6: [==========] Running 1 test from 1 test suite. 6: [----------] Global test environment set-up. 6: [----------] 1 test from STVTests 6: [RUN] STVTests.FindLowestCandidateTest 6: [OK] STVTests.FindLowestCandidateTest (0 ms) 6: [----------] 1 test from STVTests (0 ms total) 6: 6: [----------] Global test environment tear-down 6: [==========] 1 test from 1 test suite ran. (0 ms total) 6: [PASSED] 1 test. 6/8 Test #6: STVTests.FindLowestCandidateTest ... Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/28/2025

PBI 1	Ballot Re-assignment for STV
Task Description	Validate correct ballot re-assignment behavior in STV elections when a candidate exceeds the Droop quota.
Testing Number	PBI1_Unit_tests_02
Author	Anwesha Samaddar
Inputs	Automated testing suite using test function: TEST_F(STVTests, SurplusRedistributionTest) in stv_UT.cc .
Tests	1. Tests if Candidate A meets the Droop quota at exactly 3 votes. 2. Tests if Candidate B inherits votes via later preferences.
Outputs	test 9 Start 9: STVTests.SurplusRedistributionTest 9: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 9: Test timeout computed to be: 9999879 9: Note: Google Test filter = STVTests.SurplusRedistributionTest 9: [==========] Running 1 test from 1 test suite. 9: [----------] Global test environment set-up. 9: [----------] 1 test from STVTests 9: [RUN] STVTests.SurplusRedistributionTest 9: [OK] STVTests.SurplusRedistributionTest (0 ms) 9: [----------] 1 test from STVTests (0 ms total) 9: 9: [----------] Global test environment tear-down 9: [==========] 1 test from 1 test suite ran. (0 ms total) 9: [PASSED] 1 test. 9/10 Test #9: STVTests.SurplusRedistributionTest ... Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/28/2025

PBI 2	One Candidate in STV
Task Description	Validate STV election behavior with a single candidate and one ballot.
Testing Number	PBI2_Unit_tests_01
Author	Anwesha Samaddar
Inputs	1. Automated testing suite using test function: TEST_F(STVTests, SingleCandidateTest) in stv_UT.cc. 2. One ballot with a single ranked preference 3. Number of Candidates = 1, Candidate name = “D” 4. Number of seats = 1 5. shuffle=false
Tests	1. Tests if only a single candidate gets elected as winner. 2. Test if the elected candidate is “D” 3. Test if no losers are present in the single-candidate election
Outputs	test 2 Start 2: STVTests.CandidateNameTest 2: Test timeout computed to be: 9999879 2: Note: Google Test filter = STVTests.CandidateNameTest 7: Note: Google Test filter = STVTests.SingleCandidateTest 7: [==========] Running 1 test from 1 test suite. 7: [----------] Global test environment set-up. 7: [----------] 1 test from STVTests 7: [RUN] STVTests.SingleCandidateTest 7: [OK] STVTests.SingleCandidateTest (0 ms) 7: [----------] 1 test from STVTests (0 ms total) 7: 7: [----------] Global test environment tear-down 7: [==========] 1 test from 1 test suite ran. (0 ms total) 7: [PASSED] 1 test. 7/8 Test #7: STVTests.SingleCandidateTest ..... Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/28/2025

PBI 2	One Candidate in STV
Task Description	Test for a single candidate in STV
Testing Number	PBI2_System_tests_01
Author	Zoe Sepersky
Inputs	CSV File: ../testing/single_candidate_stv.csv Number of seats: 1 Select algorithm: STV Enable shuffle?: True Audit file name: audit.txt
Tests	1. Test when there is a single candidate and they have the majority of votes 2. Test when there is a single candidate and they have received no votes
Outputs	Outputs election statistics, including names of winners and losers, the amount of votes received, and if winners reached the droop quota. Output shown below: ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 5 Number of Seats: 2 Number of Candidates: 1 Droop Quota: 2 votes Winners: A | Votes: 2 | Percentage: 40.00% | Met Quota: Yes Losers: === End of Results ===
Pass/Fail	1. Pass 2. Pass
Date	4/18/2025

PBI 3	STV Results Output
Task Description	Ensure no candidate is omitted from the winners/losers list in the STV results display.
Testing Number	PBI3_System_tests_01
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/stv_ballots_regular.csv number of seats: 3 Select algorithm (STV or Plurality): STV Enable ballot shuffle? (true/false): false audit file name: audit_stv.txt
Tests	1. Test if all 6 candidates (A–F) appear in the final election result output. 2. Test if candidates of both winners and losers lists are accurately listed with vote counts and percentages. 3. Test if output formatting is clean and consistent.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 5 Number of Seats: 3 Number of Candidates: 6 Droop Quota: 2 votes Winners: A | Votes: 2 | Percentage: 40.00% | Met Quota: Yes C | Votes: 1 | Percentage: 20.00% | Met Quota: No E | Votes: 1 | Percentage: 20.00% | Met Quota: No Losers: B | Votes: 1 | Percentage: 20.00% F | Votes: 0 | Percentage: 0.00% D | Votes: 0 | Percentage: 0.00% === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/18/2025

PBI 3	STV Results Output
Task Description	Test how the system handles invalid CSV file entries or file path errors.
Testing Number	PBI3_System_tests_02
Author	Anwesha Samaddar
Inputs	Invalid CSV file name: ../testing/stv_ballots_4.csv
Tests	1. Test if the system correctly identifies missing or unreadable CSV files. 2. Test that after too many invalid attempts, the system exits cleanly.
Outputs	The system correctly identifies the invalid CSV path, displays error messages after each failed attempt, and exits after three tries.
Pass/Fail	1. Pass 2. Pass
Date	4/18/2025

PBI 3	STV Results Output
Task Description	Run STV with seats greater than the candidate count.
Testing Number	PBI3_System_tests_03
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/stv_ballots_regular.csv number of seats: 7 Select algorithm (STV or Plurality): STV Enable ballot shuffle? (true/false): true Enter audit file name: audit.txt
Tests	1. Test if all candidates are correctly listed as winners with accurate vote counts when seats ≥ candidates.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 5 Number of Seats: 8 Number of Candidates: 6 Droop Quota: 1 votes Winners: A | Votes: 1 | Percentage: 20.00% | Met Quota: Yes B | Votes: 1 | Percentage: 20.00% | Met Quota: Yes C | Votes: 1 | Percentage: 20.00% | Met Quota: Yes E | Votes: 1 | Percentage: 20.00% | Met Quota: Yes D | Votes: 0 | Percentage: 0.00% | Met Quota: No F | Votes: 0 | Percentage: 0.00% | Met Quota: No Losers: === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt
Pass/Fail	1. Pass
Date	4/18/2025

PBI 3	STV Results Output
Task Description	Run the STV election with invalid ballots.
Testing Number	PBI3_System_tests_04
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/invalid_stv_ballots.csv number of seats: 4 Select algorithm (STV or Plurality): stv Enable ballot shuffle?: true audit file name: audit.txt
Tests	1. Test system behavior when all ballots are invalid (less than half of the candidates ranked per ballot).
Outputs	The system correctly identifies all invalid ballots and displays a message stating that the election is aborted due to 0 valid ballots.
Pass/Fail	1. Pass
Date	4/18/2025

PBI 3	STV Results Output
Task Description	Automated testing suite function TEST_F(STVTests, DroopQuotaTest) in stv_UT.cc, used to validate Droop quota calculation for different seat and ballot counts in STV.
Testing Number	PBI3_Unit_tests_01
Author	Anwesha Samaddar
Inputs	1.3 ballots and 2 seats – expects quota: floor(3/(2+1)) + 1 = 2 2.3 ballots and 1 seat – expects quota: floor(3/1+1) = 2 3.0 ballots and 1 seat – expects base quota of 1
Tests	1.Tests if Droop quota is correctly computed as 2 when there are 3 ballots and 2 seats 2. Tests if Droop quota is 2 with 3 ballots and 1 seat 3. Tests if Droop quota defaults to 1 when there are 0 ballots and 1 seat
Outputs	test 3 Start 3: STVTests.DroopQuotaTest 3: Test timeout computed to be: 9999879 3: Note: Google Test filter = STVTests.DroopQuotaTest 3: [==========] Running 1 test from 1 test suite. 3: [----------] Global test environment set-up. 3: [----------] 1 test from STVTests 3: [RUN] STVTests.DroopQuotaTest 3: [OK] STVTests.DroopQuotaTest (0 ms) 3: [----------] 1 test from STVTests (0 ms total) 3: 3: [----------] Global test environment tear-down 3: [==========] 1 test from 1 test suite ran. (0 ms total) 3: [PASSED] 1 test. 3/8 Test #3: STVTests.DroopQuotaTest ..... Passed 0.01 sec test 4
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/28/2025

PBI 3	STV Results Output
Task Description	Automated testing suite function TEST_F(STVTests, runElectionTest) in stv_UT.cc, used to validate correctness of the full STV election algorithm.
Testing Number	PBI3_Unit_tests_02
Author	Anwesha Samaddar
Inputs	1. 3 STVBallots with ranked preferences 2. 3 Candidates (A, B, C) 3. Number of seats: 2 4. Shuffle = false
Tests	1. Tests if number of winners returned is exactly 2 2. Tests if number of losers is exactly 1 3. Tests if candidates B and C are selected as winners based on preference 4. Tests if candidate A is correctly marked as the loser
Outputs	8: Note: Google Test filter = STVTests.runElectionTest 8: [==========] Running 1 test from 1 test suite. 8: [----------] Global test environment set-up. 8: [----------] 1 test from STVTests 8: [RUN] STVTests.runElectionTest 8: 8: DEBUG - Final Classification: 8: Winners (2): 8: B: 2 votes 8: C: 1 votes 8: Losers (1): 8: A: 0 votes 8: [OK] STVTests.runElectionTest (0 ms) 8: [----------] 1 test from STVTests (0 ms total) 8: 8: [----------] Global test environment tear-down 8: [==========] 1 test from 1 test suite ran. (0 ms total) 8: [PASSED] 1 test. 8/8 Test #8: STVTests.runElectionTest ..... Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	4/30/2025

PBI 3	STV Results Output
Task Description	Automated testing suite function TEST_F(STVBallotTest, InvalidSTVBallotConstructor) in STVBallot_UT.cc verifies that the STVBallot class constructor throws an exception for invalid ballots that do not meet the minimum ranking requirement.
Testing Number	PBI3_Unit_tests_03
Author	Anwesha Samaddar
Inputs	1.Ballot votes: {1, 0, 0, 0, 0, 2} → Only two candidates ranked out of six 2.Ballot ID = 1
Tests	1. Tests if the constructor of STVBallot throws a std::invalid_argument when fewer than half the candidates are ranked. 2. Tests if the thrown exception contains this specific error message: "Invalid STV Ballot 1: Only 2 candidates ranked (minimum 3 required)."
Outputs	test 2 Start 2: STVBallotTest.InvalidSTVBallotConstructor 2: Working Directory: C:/Users/ASUS/CLionProjects/GIT2/repo-Team4/Project2/testing/build 2: Test timeout computed to be: 9999879 2: Note: Google Test filter = STVBallotTest.InvalidSTVBallotConstructor 2: [==========] Running 1 test from 1 test suite. 2: [----------] Global test environment set-up. 2: [----------] 1 test from STVBallotTest 2: [RUN] STVBallotTest.InvalidSTVBallotConstructor 2: [OK] STVBallotTest.InvalidSTVBallotConstructor (0 ms) 2: [----------] 1 test from STVBallotTest (0 ms total) 2: 2: [----------] Global test environment tear-down 2: [==========] 1 test from 1 test suite ran. (0 ms total) 2: [PASSED] 1 test. 2/3 Test #2: STVBallotTest.InvalidSTVBallotConstructor ...Passed 0.02 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 4	Plurality Results Output
Task Description	A unit test to verify that the Plurality election results include every candidate's name.
Testing Number	PBI4_Unit_tests_01
Author	Annabelle Coler
Inputs	1. The winners of a regular plurality election, and the expected winners 2. The losers of a regular plurality election, and the expected losers 3. The winners of a tied plurality election, and the expected winners 4. The losers of a tied plurality election, and the expected losers
Tests	1. Test that all winners of a regular plurality election appear in a winner's list 2. Test that all losers of a regular plurality election appear in a loser's list 3. Test that all winners of a tied plurality election appear in a winner's list 4. Test that all losers of a tied plurality election appear in a loser's list
Outputs	No outputs
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	4/18/2025

PBI 4	Plurality Results Output
Task Description	Debug the Plurality election results to print out every candidate's name, so that we can see each candidate and the number of votes they received.
Testing Number	PBI4_System_tests_01
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/plurality_ballots_regular.csv number of seats: 4 Select algorithm (STV or Plurality): plurality audit file name: audit_plurality.txt
Tests	1. Test if all 6 candidates (A–F) appear in the final election result output. 2. Test if candidates of both winners and losers lists are accurately listed with vote counts and percentages. 3. Test if output formatting is clean and consistent.
Outputs	=== Election Results === Election Type: Plurality ####### Plurality Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 3 Number of Seats: 4 Number of Candidates: 6 Winners: A | Votes: 2 | Percentage: 66.67% C | Votes: 1 | Percentage: 33.33% B | Votes: 0 | Percentage: 0.00% D | Votes: 0 | Percentage: 0.00% Losers: E | Votes: 0 | Percentage: 0.00% F | Votes: 0 | Percentage: 0.00% === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt Press Enter to exit...
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/18/2025

PBI 4	Plurality Results Output
Task Description	Run Plurality with seats greater than the candidate count taking user inputs.
Testing Number	PBI4_System_tests_02
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/plurality_ballots_regular.csv number of seats: 8 Select algorithm (STV or Plurality): plurality audit file name: audit_plurality.txt
Tests	1. Test if all candidates are correctly listed as winners with accurate vote counts when seats ≥ candidates.
Outputs	=== Election Results === Election Type: Plurality ####### Plurality Results ####### Number of Ballots: 3 Number of Seats: 8 Number of Candidates: 6 Winners: A | Votes: 2 | Percentage: 66.67% C | Votes: 1 | Percentage: 33.33% B | Votes: 0 | Percentage: 0.00% D | Votes: 0 | Percentage: 0.00% E | Votes: 0 | Percentage: 0.00% F | Votes: 0 | Percentage: 0.00% Losers: === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt
Pass/Fail	1. Pass
Date	4/18/2025

PBI 4	Plurality Results Output
Task Description	Run Plurality with only invalid ballots.
Testing Number	PBI4_System_tests_03
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/invalid_plurality_ballot.csv Number of seats: 3 Select algorithm (STV or Plurality): plurality Audit file name: audit.txt
Tests	1. Test if the user is notified of invalid ballots present. 2. Test if system removes invalid ballots and lists them with ballot IDs 3. Test if the system aborts the election with a clear error message.
Outputs	Invalid Plurality ballot 1: Exactly one '1' is required. Invalid Plurality ballot 2: Exactly one '1' is required. Invalid Plurality ballot 3: Non-zero/non-one value found. Invalid Plurality ballot 4: Exactly one '1' is required. Invalid Plurality ballot 5: Exactly one '1' is required. === Election Results === Election Type: Plurality ####### Plurality Results ####### Number of Invalid Ballots: 5 === List of Removed Ballots === ID 1: 1 1 0 0 0 ID 2: 0 0 0 0 0 ID 3: 2 0 0 0 0 ID 4: 0 0 1 1 0 ID 5: 0 1 0 1 0 Number of Valid Ballots: 0 Number of Seats: 3 Number of Candidates: 5 ERROR: Election aborted. No valid ballots to process.

	Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt Press Enter to exit...
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/21/2025

PBI 4	Plurality Results Output
Task Description	Run Plurality with mixed ballots (both valid and invalid ballots).
Testing Number	PBI4_System_tests_04
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/mixed_plurality_ballots.csv Number of seats: 3 Select algorithm (STV or Plurality): plurality Audit file name: audit.txt
Tests	1. Test if invalid ballots are listed with ballot IDs. 2. Test if the system removes invalid ballots and proceeds with the remaining valid ballots.
Outputs	Invalid Plurality ballot 4: Exactly one '1' is required. Invalid Plurality ballot 5: Non-zero/non-one value found. Invalid Plurality ballot 7: Non-zero/non-one value found. Invalid Plurality ballot 8: Non-zero/non-one value found. === Election Results === Election Type: PV ####### PV Results ####### Number of Invalid Ballots: 4 === List of Removed Ballots === ID 4: 0 0 1 1 1 1 ID 5: 2 3 0 0 0 1 ID 7: 0 0 0 0 0 2 ID 8: 4 0 0 1 0 0 Number of Valid Ballots: 6 Number of Seats: 3 Number of Candidates: 6 Winners: A | Votes: 2 | Percentage: 33.33% D | Votes: 1 | Percentage: 16.67% E | Votes: 1 | Percentage: 16.67%

	Losers: C | Votes: 1 | Percentage: 16.67% B | Votes: 1 | Percentage: 16.67% F | Votes: 0 | Percentage: 0.00% === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit.txt Press Enter to exit...
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/21/2025

PBI 5	Audit File Directory
Task Description	Ensure that the audit file gets stored in the correct directory.
Testing Number	PBI5_System_tests_01
Author	Zoe Sepersky
Inputs	CSV file name: ../testing/plurality_ballots_regular.csv number of seats: 3 Select algorithm (STV or Plurality): plurality audit file name: audit_plurality.txt
Tests	When running an election, ensure the audit file is generated in the correct directory.
Outputs	While printing election results, the audit file path would be displayed like: Audit log written to: C:\Users\ASUS\CLionProjects\GIT7\repo-Team4\Project2\src\audit_plurality.txt Press Enter to exit...
Pass/Fail	Pass
Date	4/18/2025

PBI 6	Shuffle Functionality for STV
Task Description	Verify whether enabling shuffle randomizes the order of ballots prior to STV processing.
Testing Number	PB6_System_tests_01
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/stv_ballots_regular.csv Number of seats: 3 Select algorithm (STV or Plurality): STV Enable ballot shuffle? (true/false): true Enter audit file name: audit.txt
Tests	1. Test that ballots are visibly shuffled before the STV algorithm runs, and the election still processes correctly.
Outputs	Ballots were successfully shuffled with a new randomized order displayed. The election ran without errors, results were generated accurately, and the audit log was created in the /src/ directory. Ballot Allocation: Ballot ID: 1, Votes: 1 0 2 0 3 0 Ballot ID: 2, Votes: 3 2 1 4 6 5 Ballot ID: 3, Votes: 1 2 0 0 3 4 Ballot ID: 4, Votes: 4 1 0 2 0 3 Ballot ID: 5, Votes: 3 0 2 0 1 0 After Shuffle: Ballot ID: 4 -> 4 1 0 2 0 3 Ballot ID: 3 -> 1 2 0 0 3 4 Ballot ID: 2 -> 3 2 1 4 6 5 Ballot ID: 1 -> 1 0 2 0 3 0 Ballot ID: 5 -> 3 0 2 0 1 0
Pass/Fail	1. Pass
Date	4/18/2025

PBI 6	Shuffle Functionality for STV
Task Description	Verify whether enabling shuffle randomizes the order of ballots in STV.
Testing Number	PB6_Unit_tests_01
Author	Anwesha Samaddar
Inputs	1. Nine ballots with different ranked candidate preferences, passed into the STVTests class object.
Tests	1. Automated testing suite using TEST_F(STVTests, BallotShufflingTest) test function in stv_UT.cpp
Outputs	Start 4: STVTests.BallotShufflingTest 4: Test command: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\testing\build\stv_UT.exe "--gtest_filter=STVTests.BallotShufflingTest" "--gtest_also_run_disabled_tests" 4: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 4: Test timeout computed to be: 9999879 4: Note: Google Test filter = STVTests.BallotShufflingTest 4: [==========] Running 1 test from 1 test suite. 4: [----------] Global test environment set-up. 4: [----------] 1 test from STVTests 4: [RUN] STVTests.BallotShufflingTest 4: [OK] STVTests.BallotShufflingTest (2 ms) 4: [----------] 1 test from STVTests (2 ms total) 4: 4: [----------] Global test environment tear-down 4: [==========] 1 test from 1 test suite ran. (2 ms total) 4: [PASSED] 1 test. 4/8 Test #4: STVTests.BallotShufflingTest ..... Passed 0.02 sec
Pass/Fail	1. Pass
Date	4/30/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(electionUnitTests, setBallotsTest) validates that setBallots() correctly reads and stores the contents of a valid STV ballot file.
Testing Number	PB7_Unit_tests_01
Author	Anwesha Samaddar
Inputs	Valid STV ballot file path: "../../testing/stv_ballots.csv"
Tests	1. Tests if setBallots() reads the file correctly and creates ballot objects 2. Verifies that the first ballot loaded has vote values matching {1, 0, 2, 0, 3} and ID = 8
Outputs	test 6 Start 6: electionUnitTests.setBallotsTest 6: Test command: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\testing\build\Election_UT.exe "--gtest_filter=electionUnitTests.setBallotsTest" "--gtest_also_run_disabled_tests" 6: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 6: Test timeout computed to be: 9999879 6: Note: Google Test filter = electionUnitTests.setBallotsTest 6: [==========] Running 1 test from 1 test suite. 6: [----------] Global test environment set-up. 6: [----------] 1 test from electionUnitTests 6: [RUN] electionUnitTests.setBallotsTest 6: [OK] electionUnitTests.setBallotsTest (0 ms) 6: [----------] 1 test from electionUnitTests (0 ms total) 6: 6: [----------] Global test environment tear-down 6: [==========] 1 test from 1 test suite ran. (0 ms total) 6: [PASSED] 1 test. 6/9 Test #6: electionUnitTests.setBallotsTest ..... Passed 0.02 sec
Pass/Fail	1. Pass 2. Pass
Date	5/01/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(electionUnitTests, SetBallotsWithInvalidFile) checks if the program throws an error when a non-existent CSV file is passed to setBallots().
Testing Number	PB7_Unit_tests_02
Author	Anwesha Samaddar
Inputs	Invalid ballot file path: "nonexistent_file.csv"
Tests	1. Tests if calling setBallots() on a non-existent file throws a std::runtime_error
Outputs	test 5 Start 5: electionUnitTests.SetBallotsWithInvalidFile 5: Test command: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\testing\build\Election_UT.exe "--gtest_filter=electionUnitTests.SetBallotsWithInvalidFile" "--gtest_also_run_disabled_tests" 5: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 5: Test timeout computed to be: 9999879 5: Note: Google Test filter = electionUnitTests.SetBallotsWithInvalidFile 5: [==========] Running 1 test from 1 test suite. 5: [----------] Global test environment set-up. 5: [----------] 1 test from electionUnitTests 5: [RUN] electionUnitTests.SetBallotsWithInvalidFile 5: [OK] electionUnitTests.SetBallotsWithInvalidFile (0 ms) 5: [----------] 1 test from electionUnitTests (0 ms total) 5: 5: [----------] Global test environment tear-down 5: [==========] 1 test from 1 test suite ran. (0 ms total) 5: [PASSED] 1 test. 5/9 Test #5: electionUnitTests.SetBallotsWithInvalidFile ..... Passed 0.01 sec
Pass/Fail	1. Pass
Date	5/01/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(electionUnitTests, GetCandidatesTest) in Election_UT.cc checks that the Election class function getCandidates() reads candidates data correctly from the input CSV file.
Testing Number	PB7_Unit_tests_04
Author	Anwesha Samaddar
Inputs	1. A test CSV file located at ../../testing/plurality_all_inputs_2.csv 2. CSV file has Election type: "PV", Number of seats: 3
Tests	1. Tests if the Election class successfully loads ballots using setBallots() 2. Tests if the list of candidates is not empty after loading. 3. Tests if exactly 6 candidates are extracted from the file. 4. Tests if the names of the first three candidates are "A", "B", and "C".
Outputs	test 8 Start 8: electionUnitTests.GetCandidatesTest 8: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 8: Test timeout computed to be: 9999879 8: Note: Google Test filter = electionUnitTests.GetCandidatesTest 8: [==========] Running 1 test from 1 test suite. 8: [----------] Global test environment set-up. 8: [----------] 1 test from electionUnitTests 8: [RUN] electionUnitTests.GetCandidatesTest 8: [OK] electionUnitTests.GetCandidatesTest (0 ms) 8: [----------] 1 test from electionUnitTests (0 ms total) 8: 8: [----------] Global test environment tear-down 8: [==========] 1 test from 1 test suite ran. (0 ms total) 8: [PASSED] 1 test. 8/10 Test #8: electionUnitTests.GetCandidatesTest ..... Passed 0.02 sec
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	5/02/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(UserInterfaceTest, getNumSeatsFromCsv) in UserInterface_UT.cc checks if the number of seats is correctly extracted from the input CSV.
Testing Number	PB7_Unit_tests_05
Author	Anwesha Samaddar
Inputs	Simulated user input that has: Number of files: 1 File path: "../../testing/plurality_all_inputs_3.csv" Audit file name: "audit.txt"
Tests	1.Tests if UserInterface function getInfo() correctly reads the number of seats from the specified CSV file. 2.Tests if the retrieved number of seats equals 3 as per the CSV contents.
Outputs	test 2 Start 2: UserInterfaceTest.getNumSeatsFromCsv 2: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 2: Test timeout computed to be: 9999879 2: Note: Google Test filter = UserInterfaceTest.getNumSeatsFromCsv 2: [==========] Running 1 test from 1 test suite. 2: [----------] Global test environment set-up. 2: [----------] 1 test from UserInterfaceTest 2: [RUN] UserInterfaceTest.getNumSeatsFromCsv 2: How many CSV files do you want to input? Enter CSV file #1: Enter audit file name: [OK] UserInterfaceTest.getNumSeatsFromCsv (0 ms) 2: [----------] 1 test from UserInterfaceTest (0 ms total) 2: 2: [----------] Global test environment tear-down 2: [==========] 1 test from 1 test suite ran. (0 ms total) 2: [PASSED] 1 test. 2/8 Test #2: UserInterfaceTest.getNumSeatsFromCsv .....Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(UserInterfaceTest, getAlgorithmFromCsv) in UserInterface_UT.cc checks if the election type is correctly extracted from the input CSV.
Testing Number	PB7_Unit_tests_06
Author	Anwesha Samaddar
Inputs	Simulated user input that has: Number of files: 1 File path: "../../testing/plurality_all_inputs_3.csv" Audit file name: "audit.txt"
Tests	1.Tests if getInfo() in UserInterface class correctly reads the election algorithm type from the specified CSV file. 2.Tests if the retrieved algorithm type equals "PV" as per the CSV contents.
Outputs	test 3 Start 3: UserInterfaceTest.getAlgorithmFromCsv 3: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 3: Test timeout computed to be: 9999879 3: Note: Google Test filter = UserInterfaceTest.getAlgorithmFromCsv 3: [==========] Running 1 test from 1 test suite. 3: [----------] Global test environment set-up. 3: [----------] 1 test from UserInterfaceTest 3: [RUN] UserInterfaceTest.getAlgorithmFromCsv 3: How many CSV files do you want to input? Enter CSV file #1: Enter audit file name: [OK] UserInterfaceTest.getAlgorithmFromCsv (0 ms) 3: [----------] 1 test from UserInterfaceTest (0 ms total) 3: 3: [----------] Global test environment tear-down 3: [==========] 1 test from 1 test suite ran. (0 ms total) 3: [PASSED] 1 test. 3/8 Test #3: UserInterfaceTest.getAlgorithmFromCsv .....Passed 0.02 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Automated testing suite function TEST_F(UserInterfaceTest, AuditFileNameNotEmpty) in UserInterface_UT.cc checks whether the audit file name entered by the user is successfully captured and stored by the UserInterface class.
Testing Number	PB7_Unit_tests_06
Author	Anwesha Samaddar
Inputs	Simulated user input that has: Number of files: 1 File path: "../../testing/plurality_all_inputs_3.csv" Audit file name: "audit_file.txt"
Tests	1.Tests if the audit file name captured via getAuditFileName() is not empty. 2.Tests if the captured audit file name exactly matches the input "audit_file.txt".
Outputs	test 6 Start 6: UserInterfaceTest.AuditFileNameNotEmpty 6: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 6: Test timeout computed to be: 9999879 6: Note: Google Test filter = UserInterfaceTest.AuditFileNameNotEmpty 6: [==========] Running 1 test from 1 test suite. 6: [----------] Global test environment set-up. 6: [----------] 1 test from UserInterfaceTest 6: [RUN] UserInterfaceTest.AuditFileNameNotEmpty 6: How many CSV files do you want to input? Enter CSV file #1: Enter audit file name: [OK] UserInterfaceTest.AuditFileNameNotEmpty (0 ms) 6: [----------] 1 test from UserInterfaceTest (0 ms total) 6: 6: [----------] Global test environment tear-down 6: [==========] 1 test from 1 test suite ran. (0 ms total) 6: [PASSED] 1 test. 6/8 Test #6: UserInterfaceTest.AuditFileNameNotEmpty .....Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 7	Election takes in all input from the CSV file.
Task Description	Verify that an election takes in all necessary information from a CSV file with no input from the user other than the CSV file name and the audit file name for Plurality.
Testing Number	PB7_System_tests_01
Author	Zoe Sepersky
Inputs	CSV file name: ../testing/plurality_all_inputs.csv Enter audit file name: audit.txt
Tests	1. Tests that the Plurality election runs as expected with all the information from the CSV file (for all valid plurality ballots). 2. Tests if there are exactly 1 winner and 5 losers.
Outputs	=== Election Results === Election Type: PV ####### PV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 3 Number of Seats: 1 Number of Candidates: 6 Winners: A | Votes: 2 | Percentage: 66.67% Losers: C | Votes: 1 | Percentage: 33.33% B | Votes: 0 | Percentage: 0.00% D | Votes: 0 | Percentage: 0.00% E | Votes: 0 | Percentage: 0.00% F | Votes: 0 | Percentage: 0.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass
Date	4/22/2025

PBI 7	Election takes in all inputs from the CSV file.
Task Description	Verify that an election takes in all necessary information from a CSV file with no input from the user other than the CSV file name and the audit file name for STV..
Testing Number	PB7_System_tests_02
Author	Anwesha Samaddar
Inputs	CSV file name: ../testing/stv_all_inputs.csv Enter audit file name: audit.txt
Tests	1. Tests that the STV election runs as expected with all the information from the CSV file (for all valid STV ballots) 2. Tests that STV assumes shuffle as true by default (no user input for turning ballot shuffle on).
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10 Number of Seats: 3 Number of Candidates: 5 Droop Quota: 3 votes Winners: A | Votes: 3 | Percentage: 30.00% | Met Quota: Yes C | Votes: 3 | Percentage: 30.00% | Met Quota: Yes B | Votes: 2 | Percentage: 20.00% | Met Quota: No Losers: E | Votes: 1 | Percentage: 10.00% D | Votes: 0 | Percentage: 0.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass
Date	4/24/2025

PBI 7	Election takes in all inputs from the CSV file.
Task Description	Verify that an election takes in all necessary information from a CSV file with no input from the user other than the CSV file name and the audit file name for MV election.
Testing Number	PB7_System_tests_03
Author	Anwesha Samaddar
Inputs	Number of csv files: 1 CSV file name: ../testing/mv_all_valid_ballots_100.csv Enter audit file name: audit_mv.txt
Tests	1. Tests that the MV election runs as expected with all the information from the CSV file 2. Tests that the number of valid ballots are 100 in the output
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 100 Number of Seats: 4 Number of Candidates: 6 Winners: A | Votes: 50 | Percentage: 50.00% F | Votes: 40 | Percentage: 40.00% D | Votes: 40 | Percentage: 40.00% E | Votes: 40 | Percentage: 40.00% Losers: C | Votes: 40 | Percentage: 40.00% B | Votes: 40 | Percentage: 40.00% === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\src\audit_mv.txt
Pass/Fail	1. Pass 2. Pass
Date	4/24/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Verify that MV ballot methods work as intended
Testing Number	PBI8_Unit_tests_01
Author	Zoe Sepersky
Inputs	Automated testing suite ran on file mvballot_UT.cc
Tests	1. Test that the constructor does not throw any exceptions when initialized with correct data 2. Test that getPreference() correctly returns the candidate ID numbers 3. Test that isValid() returns the correct value and throws an exception when a ballot is instantiated with invalid data
Outputs	Start 1: MVBallotTests.ConstructorTest 1/3 Test #1: MVBallotTests.ConstructorTest ..... Passed 0.00 sec Start 2: MVBallotTests.GetPreferencesTest 2/3 Test #2: MVBallotTests.GetPreferencesTest ... Passed 0.00 sec Start 3: MVBallotTests.isValidTest 3/3 Test #3: MVBallotTests.isValidTest ..... Passed 0.00 sec 100% tests passed, 0 tests failed out of 3
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/30/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Verify that MV logic individual methods are working correctly
Testing Number	PBI9_Unit_tests_02
Author	Zoe Sepersky
Inputs	Automated testing suite ran on file mv_UT.cc
Tests	1. Test that when an election is run with correct data, no exception is thrown 2. Test that when an election is run, the winners of said election are correct 3. Test that when a tie happens, the winner is one of the candidates that are tied
Outputs	Start 1: MVTests.runElectionTest 1/3 Test #1: MVTests.runElectionTest ..... Passed 0.00 sec Start 2: MVTests.getWinnersTest 2/3 Test #2: MVTests.getWinnersTest ..... Passed 0.00 sec Start 3: MVTests.tieTests 3/3 Test #3: MVTests.tieTests ..... Passed 0.00 sec 100% tests passed, 0 tests failed out of 3
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	4/30/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Verify that a MV election runs correctly with correct output and correct winners/losers on a valid CSV file.
Testing Number	PBI8_System_Tests_01
Author	Zoe Sepersky
Inputs	Number of CSV Files: 1 CSV File name: ../testing/mv_all_valid_ballots_10.csv Audit File name: audit.txt
Tests	When running the election, confirm that the decided winners and losers are correct when manually tallying votes.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10 Number of Seats: 4 Number of Candidates: 6 Winners: F | Votes: 5 | Percentage: 50.00% A | Votes: 4 | Percentage: 40.00% D | Votes: 4 | Percentage: 40.00% E | Votes: 4 | Percentage: 40.00% Losers: B | Votes: 3 | Percentage: 30.00% C | Votes: 3 | Percentage: 30.00% === End of Results ===
Pass/Fail	Pass
Date	4/30/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Verify that a MV election correctly handles a CSV file with only invalid ballots
Testing Number	PBI8_System_Tests_02
Author	Zoe Sepersky
Inputs	Number of CSV Files: 1 CSV File name: ../testing/mv_all_invalid_ballots.csv Audit File name: audit.txt
Tests	When running the election, confirm that the system handles invalid ballots correctly and exits gracefully.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 9 === List of Removed Ballots === ID 1: 1 1 1 2 0 0 ID 2: 1 0 4 1 0 1 ID 3: 0 1 3 2 1 0 ID 4: 5 0 1 0 0 1 ID 5: 0 2 3 4 0 1 ID 6: 1 0 2 1 1 0 ID 7: 0 4 1 1 0 3 ID 8: 1 2 3 1 1 0 ID 9: 0 1 2 3 1 1 Number of Valid Ballots: 0 Number of Seats: 4 Number of Candidates: 6 ERROR: Election aborted. No valid ballots to process.
Pass/Fail	Pass
Date	4/30/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Verify that a MV election correctly handles a CSV file with mixed (valid and invalid) ballots
Testing Number	PBI8_System_Tests_03
Author	Anwesha Samaddar
Inputs	Number of CSV Files: 1 CSV File name: ../testing/mv_mixed_ballots.csv Audit File name: audit.txt
Tests	1. When running the election, confirm that the system handles invalid ballots correctly 2. Test if MV election results show the correct number of valid and invalid ballots 3. Test if list of removed ballots are displayed and recorded in audit file 4. Test if list of winners and losers are displayed as expected
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 4 === List of Removed Ballots === ID 5: 2 3 0 0 4 1 ID 6: 1 0 4 1 1 2 ID 7: 3 0 1 1 0 0 ID 8: 1 0 4 1 1 2 Number of Valid Ballots: 6 Number of Seats: 4 Number of Candidates: 6 Winners: F | Votes: 4 | Percentage: 66.67% A | Votes: 3 | Percentage: 50.00% B | Votes: 3 | Percentage: 50.00% C | Votes: 2 | Percentage: 33.33% Losers: D | Votes: 2 | Percentage: 33.33% E | Votes: 2 | Percentage: 33.33% === End of Results ===

	Audit log written to: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\src\audit_mv.txt Press Enter to exit...
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	4/30/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Test for a single candidate in MV
Testing Number	PBI8_System_tests_04
Author	Anwesha Samaddar
Inputs	Number of CSV inputs: 1 CSV file #1: ../testing/mv_single_candidate.csv Audit file name: audit_mv.txt
Tests	1. Test if the single candidate present is declared as winner. 2. Test if the losers list is empty. 3. Test if the number of valid ballots reported is correct.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10 Number of Seats: 4 Number of Candidates: 1 Winners: A | Votes: 6 | Percentage: 60.00% Losers: === End of Results === Audit log written to: C:\Users\ASUS\CLionProjects\GIT10\repo-Team4\Project2\src\audit_mv.txt Press Enter to exit...
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	5/02/2025

PBI 8	Implement Municipal voting (MV) algorithm
Task Description	Test MV election behaviour when number of seats is greater than number of candidates in CSV file.
Testing Number	PBI8_System_tests_05
Author	Anwesha Samaddar
Inputs	Number of CSV inputs: 1 CSV file #1: ../testing/mv_seats_more_than_candidates.csv Audit file name: audit_mv.txt
Tests	1. Test if the all candidates present in csv are declared as winners. 2. Test if the losers list is empty. 3. Test if the number of seats and valid ballots reported is correct.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 9 Number of Seats: 10 Number of Candidates: 6 Winners: A | Votes: 4 | Percentage: 44.44% D | Votes: 4 | Percentage: 44.44% E | Votes: 4 | Percentage: 44.44% F | Votes: 4 | Percentage: 44.44% B | Votes: 3 | Percentage: 33.33% C | Votes: 3 | Percentage: 33.33% Losers: === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verify that Election can take in information from multiple CSV files.
Testing Number	PB9_Unit_tests_01
Author	Annabelle Coler
Inputs	CSV file names vector: {“plurality_ballots_1.csv”, “plurality_ballots_2.csv”, “plurality_ballots_3.csv”} nameP = test.getCSVFileNames()
Tests	Tests that Election’s getCSVFileNames() method returns a concatenated string containing the names of all input files.
Outputs	csvFileNames vector and nameP are equal, so the test passes.
Pass/Fail	Pass
Date	4/27/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verify that UserInterface can take in information from multiple CSV files.
Testing Number	PB9_Unit_tests_02
Author	Annabelle Coler
Inputs	CSV file names vector: {"../../testing/stv_ballots.csv","../../testing/stv_ballots2.csv"} ui.getCSVFileNames()
Tests	Tests that UserInterface's getCSVFileNames() method returns a vector of strings containing the names of all input CSV files
Outputs	Csv_files and ui.getCSVFileNames() are equal, so the test passes. Start 1: UserInterfaceTest.GetCsvFileNameReturnsCorrectValue 1/5 Test #1: UserInterfaceTest.GetCsvFileNameReturnsCorrectValue ..... Passed 0.01 sec
Pass/Fail	Pass
Date	4/27/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verify that the Election class function setBallots() successfully parses and loads ballots from multiple valid CSV files
Testing Number	PB9_Unit_tests_04
Author	Anwesha Samaddar
Inputs	Automatic testing suit function TEST_F(electionUnitTests, SetBallotsWithMultipleFiles) in file Election_UT.cc has the following input: Vector of valid ballot file paths:{ "../../testing/stv_all_inputs_regular.csv", "../../testing/stv_all_inputs_regular_2.csv", "../../testing/stv_all_inputs_regular_3.csv"}
Tests	1. Tests whether setBallots() executes without throwing errors for multiple input files 2. Tests if the total number of ballots loaded equals 80
Outputs	test 7 Start 7: electionUnitTests.SetBallotsWithMultipleFiles "--gtest_also_run_disabled_tests" 7: Working Directory: C:/Users/ASUS/CLionProjects/GIT2/repo-Team4/Project2/testing/build 7: Test timeout computed to be: 9999879 7: Note: Google Test filter = electionUnitTests.SetBallotsWithMultipleFiles 7: [==========] Running 1 test from 1 test suite. 7: [----------] Global test environment set-up. 7: [----------] 1 test from electionUnitTests 7: [RUN] electionUnitTests.SetBallotsWithMultipleFiles 7: [OK] electionUnitTests.SetBallotsWithMultipleFiles (0 ms) 7: [----------] 1 test from electionUnitTests (0 ms total) 7: 7: [----------] Global test environment tear-down 7: [==========] 1 test from 1 test suite ran. (1 ms total) 7: [PASSED] 1 test. 7/10 Test #7:electionUnitTests.SetBallotsWithMultipleFiles ...Passed 0.02 sec
Pass/Fail	1.Pass 2.Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verifies that the shuffle flag (shuffle_stv) is correctly set to “true” when the STV algorithm is specified in the CSV file.
Testing Number	PB9_Unit_tests_05
Author	Anwesha Samaddar

Inputs	Automated testing suite function TEST_F(UserInterfaceTest, GetShuffleStvReturnsCorrectValue) validates if shuffle is true.
	Default csv file names used:
	csvFileNames = {
	"../../testing/stv_ballots.csv",
	"../../testing/stv_ballots2.csv",
	};

Tests	1. Confirms default behavior where shuffle_stv is automatically enabled for STV elections.
Outputs	test 5 Start 5: UserInterfaceTest.GetShuffleStvReturnsCorrectValue 5: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 5: Test timeout computed to be: 9999879 5: Note: Google Test filter = UserInterfaceTest.GetShuffleStvReturnsCorrectValue 5: [==========] Running 1 test from 1 test suite. 5: [----------] Global test environment set-up. 5: [----------] 1 test from UserInterfaceTest 5: [RUN] UserInterfaceTest.GetShuffleStvReturnsCorrectValue 5: [OK] UserInterfaceTest.GetShuffleStvReturnsCorrectValue (0 ms) 5: [----------] 1 test from UserInterfaceTest (0 ms total) 5: 5: [----------] Global test environment tear-down 5: [==========] 1 test from 1 test suite ran. (0 ms total) 5: [PASSED] 1 test. 5/5 Test #5: UserInterfaceTest.GetShuffleStvReturnsCorrectValue ..... Passed 0.01 sec
Pass/Fail	1.Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verifies that the audit file name input by the user is successfully captured and is not left empty.
Testing Number	PB9_Unit_tests_06
Author	Anwesha Samaddar
Inputs	Automated testing suite function TEST_F(UserInterfaceTest, AuditFileNameNotEmpty) uses the default audit file name input: "audit.txt"
Tests	1. Tests if getAuditFileName() returns a non-empty string after valid user input. 2. Verifies that the audit file name assignment from cin to the auditFileName member variable is functioning as expected.
Outputs	test 6 Start 6: UserInterfaceTest.AuditFileNameNotEmpty 6: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 6: Test timeout computed to be: 9999879 6: Note: Google Test filter = UserInterfaceTest.AuditFileNameNotEmpty 6: [==========] Running 1 test from 1 test suite. 6: [----------] Global test environment set-up. 6: [----------] 1 test from UserInterfaceTest 6: [RUN] UserInterfaceTest.AuditFileNameNotEmpty 6: [OK] UserInterfaceTest.AuditFileNameNotEmpty (0 ms) 6: [----------] 1 test from UserInterfaceTest (0 ms total) 6: 6: [----------] Global test environment tear-down 6: [==========] 1 test from 1 test suite ran. (0 ms total) 6: [PASSED] 1 test. 6/8 Test #6: UserInterfaceTest.AuditFileNameNotEmpty ..... Passed 0.02 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Ensures that all input file names collected via UserInterface::getInfo() have a valid .csv extension.
Testing Number	PB9_Unit_tests_07
Author	Anwesha Samaddar

Inputs	Automated testing suite function TEST_F(UserInterfaceTest, AllCsvFilesHaveCorrectExtension) uses the default csv file names as input:
	csvFileNames = {
	"../../testing/stv_ballots.csv",
	"../../testing/stv_ballots2.csv",
	};

Tests	1.Validates the input constraint logic that was implemented in UserInterface to enforce .csv file inputs. 2.Iterates over all values returned by getCsvFileNames() and checks that each filename ends with .csv using EXPECT_EQ(file.substr(file.size() - 4), ".csv").
Outputs	test 7 Start 7: UserInterfaceTest.AllCsvFilesHaveCorrectExtension 7: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 7: Test timeout computed to be: 9999879 7: Note: Google Test filter = UserInterfaceTest.AllCsvFilesHaveCorrectExtension 7: [==========] Running 1 test from 1 test suite. 7: [----------] Global test environment set-up. 7: [----------] 1 test from UserInterfaceTest 7: [RUN] UserInterfaceTest.AllCsvFilesHaveCorrectExtension 7: [OK] UserInterfaceTest.AllCsvFilesHaveCorrectExtension (0 ms) 7: [----------] 1 test from UserInterfaceTest (0 ms total) 7: [----------] Global test environment tear-down 7: [==========] 1 test from 1 test suite ran. (0 ms total) 7: [PASSED] 1 test. 7/8 Test #7: UserInterfaceTest.AllCsvFilesHaveCorrectExtension ..... Passed 0.01 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Verifies that the User Interface system prompts the user upto 3 times for entering a valid input for number of csv files and gracefully exits.
Testing Number	PB9_Unit_tests_08
Author	Anwesha Samaddar
Inputs	Automated testing suite function TEST_F(UserInterfaceTest, HandlesInvalidInputs) in userinterface_UT.cc file has an input stream simulating 3 invalid values for number of input files: -10, abc, 0
Tests	1.Tests that the User Interface gives the user up to 3 chances to enter a valid (positive) number of .csv files. 2. Tests if the program gracefully exits if the user continues to enter invalid values.
Outputs	test 8 Start 8: UserInterfaceSimpleTest.HandlesInvalidInputs 8: Working Directory: C:/Users/ASUS/CLionProjects/GIT10/repo-Team4/Project2/testing/build 8: Test timeout computed to be: 9999879 8: Note: Google Test filter = UserInterfaceSimpleTest.HandlesInvalidInputs 8: [==========] Running 1 test from 1 test suite. 8: [----------] Global test environment set-up. 8: [----------] 1 test from UserInterfaceSimpleTest 8: [RUN] UserInterfaceSimpleTest.HandlesInvalidInputs 8: [OK] UserInterfaceSimpleTest.HandlesInvalidInputs (20 ms) 8: [----------] 1 test from UserInterfaceSimpleTest (20 ms total) 8: [----------] Global test environment tear-down 8: [==========] 1 test from 1 test suite ran. (20 ms total) 8: [PASSED] 1 test. 8/8 Test #8: UserInterfaceSimpleTest.HandlesInvalidInputs ..... Passed 0.04 sec
Pass/Fail	1. Pass 2. Pass
Date	5/02/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing STV election system behaviour when 3 CSV files with all valid ballots are given as input.
Testing Number	PB9_System_tests_01
Author	Anwesha Samaddar
Inputs	Number of csv files: 3 Enter CSV file #1: ../testing/stv_all_inputs_regular.csv Enter CSV file #2: ../testing/stv_all_inputs_regular_2.csv Enter CSV file #3: ../testing/stv_all_inputs_regular_3.csv Enter audit file name: audit_stv.txt # seats = 3, # candidates = 5, # total valid ballots = 100, # total invalid ballots = 0
Tests	1.Tests if the STV system reports number of valid ballots = 100 2. Tests if the STV system reports number of invalid ballots = 0 3. Tests if the droop quota reported is accurate = 26 4. Tests if the winners and losers lists are as expected.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 100 Number of Seats: 3 Number of Candidates: 5 Droop Quota: 26 votes Winners: A | Votes: 26 | Percentage: 26.00% | Met Quota: Yes C | Votes: 26 | Percentage: 26.00% | Met Quota: Yes B | Votes: 19 | Percentage: 19.00% | Met Quota: No Losers: D | Votes: 1 | Percentage: 1.00% E | Votes: 11 | Percentage: 11.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing STV election system behaviour when 2 CSV files are given as input with mixed ballots (valid and invalid ballots).
Testing Number	PB9_System_tests_02
Author	Anwesha Samaddar
Inputs	Number of csv files: 2 Enter CSV file #1: ../testing/stv_mixed_ballots_200.csv Enter CSV file #2: ../testing/stv_mixed_ballots_300.csv Enter audit file name: audit_stv.txt # seats = 3, # candidates = 5, # total ballots =500, # total invalid ballots = 15, #total valid ballots = 485
Tests	1.Tests if the STV system reports the correct number of valid ballots = 485 2. Tests if the STV system reports number of invalid ballots = 15 3. Tests if the droop quota reported is accurate = 122 4. Tests if 15 invalid ballots are removed and reported in results. 5. Tests if the winners and losers lists are as expected.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 15 === List of Removed Ballots === ID 123: 1 0 0 0 0 ID 133: 1 0 0 0 0 ID 137: 0 0 0 1 0 ID 147: 1 0 0 0 0 ID 153: 0 0 1 0 0 ID 168: 1 0 0 0 0 ID 198: 1 0 0 0 0 ID 271: 1 0 0 0 0 ID 281: 1 0 0 0 0 ID 285: 0 0 1 0 0 ID 301: 1 0 0 0 0 ID 346: 0 0 0 1 0 ID 472: 1 0 0 0 0 ID 486: 0 0 0 0 1 ID 496: 1 0 0 0 0 Number of Valid Ballots: 485 Number of Seats: 3

	Number of Candidates: 5 Droop Quota: 122 votes Winners: A | Votes: 122 | Percentage: 25.15% | Met Quota: Yes B | Votes: 124 | Percentage: 25.57% | Met Quota: Yes C | Votes: 133 | Percentage: 27.42% | Met Quota: Yes Losers: E | Votes: 62 | Percentage: 12.78% D | Votes: 99 | Percentage: 20.41% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass 5. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing STV election system behaviour when 2 CSV files are given as input with mixed ballots (valid and invalid ballots) and having only 1 seat.
Testing Number	PB9_System_tests_03
Author	Anwesha Samaddar
Inputs	Number of csv files: 2 Enter CSV file #1: ../testing/stv_mixed_ballots_200_seat_1.csv Enter CSV file #2: ../testing/stv_mixed_ballots_300_seat_1.csv Enter audit file name: audit_stv.txt # seats = 1, # candidates = 5, # total ballots =500, # total invalid ballots = 15, #total valid ballots = 485
Tests	1.Tests if the STV system reports the correct number of valid ballots = 485 2. Tests if the STV system reports number of invalid ballots = 15 3. Tests if the droop quota reported is accurate = 243 4. Tests if 15 invalid ballots are removed and reported in results. 5. Tests if there is exactly 1 winner and 4 losers.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 15 === List of Removed Ballots === ID 71: 1 0 0 0 0 ID 81: 1 0 0 0 0 ID 85: 0 0 1 0 0 ID 101: 1 0 0 0 0 ID 146: 0 0 0 1 0 ID 272: 1 0 0 0 0 ID 286: 0 0 0 0 1 ID 296: 1 0 0 0 0 ID 423: 1 0 0 0 0 ID 433: 1 0 0 0 0 ID 437: 0 0 0 1 0 ID 447: 1 0 0 0 0 ID 453: 0 0 1 0 0 ID 468: 1 0 0 0 0 ID 498: 1 0 0 0 0 Number of Valid Ballots: 485 Number of Seats: 1 Number of Candidates: 5

	Droop Quota: 243 votes Winners: B | Votes: 367 | Percentage: 75.67% | Met Quota: Yes Losers: E | Votes: 62 | Percentage: 12.78% D | Votes: 99 | Percentage: 20.41% C | Votes: 133 | Percentage: 27.42% A | Votes: 172 | Percentage: 35.46% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass 5. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing STV election system behaviour when 1 CSV file is given as input with 10,000 valid ballots and named candidates.
Testing Number	PB9_System_tests_04
Author	Anwesha Samaddar
Inputs	Number of csv files: 1 Enter CSV file #1: ../testing/stv_ballots_named_10000.csv Enter audit file name: audit_stv.txt # seats = 3, # candidates = 5, # total invalid ballots = 0, # total valid ballots = 10000
Tests	1.Tests if the STV system reports the correct number of valid ballots = 10000 2. Tests if the STV system reports number of invalid ballots = 0 3. Tests if the droop quota reported is accurate = 2501 4. Tests if there are exactly 3 winner and 2 losers.
Outputs	=== Election Results === Election Type: STV ####### STV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10000 Number of Seats: 3 Number of Candidates: 5 Droop Quota: 2501 votes Winners: Bill Jones | Votes: 2501 | Percentage: 25.01% | Met Quota: Yes Alice Mix | Votes: 994 | Percentage: 9.94% | Met Quota: No Sally Ride | Votes: 1098 | Percentage: 10.98% | Met Quota: No Losers: Ahmed Mohamed | Votes: 110 | Percentage: 1.10% Siyang Xiong | Votes: 221 | Percentage: 2.21% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing MV election system behaviour when 3 CSV files with all valid ballots are given as input.
Testing Number	PB9_System_tests_05
Author	Anwesha Samaddar
Inputs	Number of csv files: 3 Enter CSV file #1: ../testing/mv_all_valid_ballots_100.csv Enter CSV file #2: ../testing/mv_all_valid_ballots_200.csv Enter CSV file #3: ../testing/mv_all_valid_ballots_300.csv Enter audit file name: audit_mv.txt # seats = 4, # candidates = 6, # total valid ballots = 600, # total invalid ballots = 0
Tests	1. Tests if the MV system reports number of valid ballots = 600 2. Tests if the MV system reports number of invalid ballots = 0 3. Tests if the reported number of seats = 4 and number of candidates = 6 4. Tests if there are exactly 4 winners (A,F,C,D) and 2 losers (E,B).
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 600 Number of Seats: 4 Number of Candidates: 6 Winners: A | Votes: 299 | Percentage: 49.83% F | Votes: 243 | Percentage: 40.50% C | Votes: 241 | Percentage: 40.17% D | Votes: 240 | Percentage: 40.00% Losers: E | Votes: 238 | Percentage: 39.67% B | Votes: 236 | Percentage: 39.33% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing the MV election system behaviour when 2 CSV files with mixed ballots (valid and invalid ballots) are given as input.
Testing Number	PB9_System_tests_06
Author	Anwesha Samaddar
Inputs	Number of csv files: 2 Enter CSV file #1: ../testing/mv_mixed_ballots_100.csv Enter CSV file #2: ../testing/mv_mixed_ballots_200.csv Enter audit file name: audit_mv.txt # seats = 4, # candidates = 6, # total ballots = 300, # total invalid ballots = 14, # total valid ballots = 286
Tests	1. Tests if the MV system reports number of valid ballots = 286 2. Tests if the MV system reports number of invalid ballots = 14 3. Tests if 14 invalid ballots are removed and reported in results. 4. Tests if the reported number of seats = 4 and number of candidates = 6 5. Tests if there are exactly 4 winners and 2 losers.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 14 === List of Removed Ballots === ID 6: 1 0 4 1 1 2 ID 8: 1 0 4 1 1 2 ID 37: 3 0 1 1 0 0 ID 38: 1 0 4 1 1 2 ID 48: 1 0 4 1 1 2 ID 76: 3 0 1 1 0 0 ID 92: 3 0 1 1 0 0 ID 108: 1 0 4 1 1 2 ID 148: 1 0 4 1 1 2 ID 203: 2 1 4 3 1 0 ID 239: 2 0 1 1 0 3 ID 273: 3 1 1 0 2 0 ID 287: 0 1 4 5 0 1 ID 299: 2 0 3 1 4 0 Number of Valid Ballots: 286 Number of Seats: 4

	Number of Candidates: 6 Winners: F | Votes: 149 | Percentage: 52.10% A | Votes: 143 | Percentage: 50.00% D | Votes: 139 | Percentage: 48.60% E | Votes: 113 | Percentage: 39.51% Losers: B | Votes: 97 | Percentage: 33.92% C | Votes: 84 | Percentage: 29.37% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass 5. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing the MV election system behaviour when 2 CSV files are given as input with mixed ballots (valid and invalid ballots) and having only 1 seat.
Testing Number	PB9_System_tests_07
Author	Anwesha Samaddar
Inputs	Number of csv files: 2 Enter CSV file #1: ../testing/mv_mixed_ballots_200_seat_1.csv Enter CSV file #2: ../testing/mv_mixed_ballots_100_seat_1.csv Enter audit file name: audit_mv.txt # seats = 1, # candidates = 5, # total ballots = 300, # total invalid ballots = 10, # total valid ballots = 290
Tests	1. Tests if the MV system reports number of valid ballots = 290 2. Tests if the MV system reports number of invalid ballots = 10 3. Tests if 10 invalid ballots are removed and reported in results. 4. Tests if the reported number of seats = 1 and number of candidates = 5 5. Tests if there are exactly 1 winner and 4 losers.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 10 === List of Removed Ballots === ID 103: 2 1 4 3 1 ID 139: 2 0 1 1 0 ID 173: 3 1 1 0 2 ID 187: 0 1 4 5 0 ID 208: 1 0 4 1 1 ID 237: 3 0 1 1 0 ID 238: 1 0 4 1 1 ID 248: 1 0 4 1 1 ID 276: 3 0 1 1 0 ID 292: 3 0 1 1 0 Number of Valid Ballots: 290 Number of Seats: 1 Number of Candidates: 5 Winners: A | Votes: 147 | Percentage: 50.69%

	Losers: D | Votes: 143 | Percentage: 49.31% E | Votes: 116 | Percentage: 40.00% B | Votes: 97 | Percentage: 33.45% C | Votes: 87 | Percentage: 30.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass 5. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Testing the MV election system behaviour when 1 CSV file is given as input with 10,000 valid ballots and named candidates.
Testing Number	PB9_System_tests_08
Author	Anwesha Samaddar
Inputs	Number of csv files: 1 Enter CSV file #1: ../testing/mv_ballots_named_10000.csv Enter audit file name: audit_mv.txt # seats = 3, # candidates = 5, # total invalid ballots = 0, # total valid ballots = 10000
Tests	1.Tests if the MV system reports the correct number of valid ballots = 10000 2. Tests if the MV system reports number of invalid ballots = 0 3. Tests if the reported number of seats = 3 and number of candidates = 5 4. Tests if there are exactly 3 winner and 2 losers.
Outputs	=== Election Results === Election Type: MV ####### MV Results ####### Number of Invalid Ballots: 0 Number of Valid Ballots: 10000 Number of Seats: 3 Number of Candidates: 5 Winners: Bill Jones | Votes: 4910 | Percentage: 49.10% Siyang Xiong | Votes: 4267 | Percentage: 42.67% Ahmed Mohamed | Votes: 4104 | Percentage: 41.04% Losers: Sally Ride | Votes: 3905 | Percentage: 39.05% Alice Mix | Votes: 3661 | Percentage: 36.61% === End of Results ===
Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Test that the system can handle multiple Plurality CSV files with large numbers.
Testing Number	PB9_System_tests_09
Author	Zoe Sepersky
Inputs	Number of CSV Files: 5 CSV File #1: ../testing/plurality_all_inputs_100.csv CSV File #2 ../testing/plurality_all_inputs_2.csv CSV File #3 ../testing/plurality_all_inputs_200.csv CSV File #4 ../testing/plurality_all_inputs_300.csv CSV File #5 ../testing/plurality_all_inputs_3.csv Audit file name: audit.txt
Tests	1. Tests that the system runs correctly with a Plurality election and a large amount of ballots 2. Tests that the system has the correct output
Outputs	=== Election Results === Election Type: PV ####### PV Results ####### Number of Invalid Ballots: 84 === List of Removed Ballots === ID 405: 0 0 1 1 0 0 ID 412: 0 0 1 2 0 0 ID 413: 1 2 3 0 0 0 ID 417: 0 1 0 1 0 0 ID 418: 0 0 0 1 1 0 ID 421: 1 0 3 0 0 0 ID 422: 0 1 1 0 0 0 ID 428: 0 0 0 2 1 0 ID 430: 3 1 0 3 0 0 ID 436: 2 3 1 0 0 0 ID 443: 1 0 4 5 0 0 ID 449: 0 3 0 0 2 1 ID 455: 0 0 1 1 0 0 ID 462: 0 0 1 2 0 0 ID 463: 1 2 3 0 0 0 ID 467: 0 1 0 1 0 0 ID 468: 0 0 0 1 1 0 ID 471: 1 0 3 0 0 0

	ID 472: 0 1 1 0 0 0 ID 478: 0 0 0 2 1 0 ID 480: 3 1 0 3 0 0 ID 486: 2 3 1 0 0 0 ID 493: 1 0 4 5 0 0 ID 499: 0 3 0 0 2 1 ID 505: 0 0 1 1 0 0 ID 512: 0 0 1 2 0 0 ID 513: 1 2 3 0 0 0 ID 517: 0 1 0 1 0 0 ID 518: 0 0 0 1 1 0 ID 521: 1 0 3 0 0 0 ID 522: 0 1 1 0 0 0 ID 528: 0 0 0 2 1 0 ID 530: 3 1 0 3 0 0 ID 536: 2 3 1 0 0 0 ID 543: 1 0 4 5 0 0 ID 549: 0 3 0 0 2 1 ID 555: 0 0 1 1 0 0 ID 562: 0 0 1 2 0 0 ID 563: 1 2 3 0 0 0 ID 567: 0 1 0 1 0 0 ID 568: 0 0 0 1 1 0 ID 571: 1 0 3 0 0 0 ID 572: 0 1 1 0 0 0 ID 578: 0 0 0 2 1 0 ID 580: 3 1 0 3 0 0 ID 586: 2 3 1 0 0 0 ID 593: 1 0 4 5 0 0 ID 599: 0 3 0 0 2 1 ID 605: 0 0 1 1 0 0 ID 612: 0 0 1 2 0 0 ID 613: 1 2 3 0 0 0 ID 617: 0 1 0 1 0 0 ID 618: 0 0 0 1 1 0 ID 621: 1 0 3 0 0 0 ID 622: 0 1 1 0 0 0 ID 628: 0 0 0 2 1 0 ID 630: 3 1 0 3 0 0 ID 636: 2 3 1 0 0 0 ID 643: 1 0 4 5 0 0 ID 649: 0 3 0 0 2 1 ID 655: 0 0 1 1 0 0 ID 662: 0 0 1 2 0 0 ID 663: 1 2 3 0 0 0 ID 667: 0 1 0 1 0 0 ID 668: 0 0 0 1 1 0 ID 671: 1 0 3 0 0 0

	ID 672: 0 1 1 0 0 0 ID 678: 0 0 0 2 1 0 ID 680: 3 1 0 3 0 0 ID 686: 2 3 1 0 0 0 ID 693: 1 0 4 5 0 0 ID 699: 0 3 0 0 2 1 ID 705: 0 0 1 1 0 0 ID 712: 0 0 1 2 0 0 ID 713: 1 2 3 0 0 0 ID 717: 0 1 0 1 0 0 ID 718: 0 0 0 1 1 0 ID 721: 1 0 3 0 0 0 ID 722: 0 1 1 0 0 0 ID 728: 0 0 0 2 1 0 ID 730: 3 1 0 3 0 0 ID 736: 2 3 1 0 0 0 ID 743: 1 0 4 5 0 0 ID 749: 0 3 0 0 2 1 Number of Valid Ballots: 666 Number of Seats: 3 Number of Candidates: 6 Winners: A | Votes: 272 | Percentage: 40.84% B | Votes: 136 | Percentage: 20.42% C | Votes: 129 | Percentage: 19.37% Losers: F | Votes: 68 | Percentage: 10.21% E | Votes: 61 | Percentage: 9.16% D | Votes: 0 | Percentage: 0.00% === End of Results ===
Pass/Fail	1. Pass 2. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Validate that an election exits gracefully and reports correct information when taking in multiple CSV files with only invalid ballots
Testing Number	PB9_System_tests_10
Author	Zoe Sepersky
Inputs	Number of CSV Files: 2 CSV File #1: ../testing/plurality_all_invalid_5.csv CSV File #2 ../testing/plurality_all_invalid_10.csv Audit file name: audit.txt
Tests	1. Test that the system exits gracefully 2. Test that the system catches all invalid ballots
Outputs	=== Election Results === Election Type: PV ####### PV Results ####### Number of Invalid Ballots: 15 === List of Removed Ballots === ID 1: 1 2 3 1 ID 2: 2 1 3 1 ID 3: 0 0 0 0 ID 4: 1 1 2 3 ID 5: 3 4 1 2 ID 6: 1 2 3 1 ID 7: 2 1 3 1 ID 8: 0 0 0 0 ID 9: 1 1 2 3 ID 10: 3 4 1 2 ID 11: 1 2 3 1 ID 12: 2 1 3 1 ID 13: 0 0 0 0 ID 14: 1 1 2 3 ID 15: 3 4 1 2 Number of Valid Ballots: 0 Number of Seats: 2 Number of Candidates: 4 ERROR: Election aborted. No valid ballots to process.
Pass/Fail	1. Pass 2. Pass
Date	5/03/2025

PBI 9	Election can take in multiple CSV files
Task Description	Validate that the PV election system reports correct information when taking in a single CSV with 10000 ballots and named candidates.
Testing Number	PB9_System_tests_11
Author	Anwesha Samaddar
Inputs	Number of CSV Files: 1 CSV File #1: ../testing/plurality_ballots_named_10000.csv Audit file name: audit.txt
Tests	1.Tests that the system reports the correct number of invalid ballots = 604 2.Tests that the system reports the correct number of valid ballots = 9396 3.Test that the system lists and removes all invalid ballots 4.Tests that PV election system reports the election stats correctly: exactly with 3 winners and 2 losers
Outputs	=== Election Results === Election Type: PV ####### PV Results ####### Number of Invalid Ballots: 604 === List of Removed Ballots === ID 55: 0 0 1 1 0 ID 59: 0 0 0 0 0 ID 72: 0 1 1 0 0 ID 158: 0 0 1 1 0 ID 162: 0 0 0 0 0 ID 771: 0 0 0 1 1 ID 772: 0 0 0 0 0 ID 789: 0 1 1 0 0 ID 796: 0 0 0 0 0 ID 806: 0 0 0 0 0 ID 822: 0 0 1 1 0 ID 826: 0 0 0 0 0 ID 834: 0 1 0 1 0 ID 835: 0 0 0 1 1 ID 836: 0 0 0 0 0 ID 839: 0 1 1 0 0 ID 846: 0 0 0 0 0 ID 856: 0 0 0 0 0 …… ** continued list upto 604 invalid ballots

	Number of Valid Ballots: 9396
	Number of Seats: 3
	Number of Candidates: 5
	
	Winners:
	  Bill Jones | Votes: 4118 | Percentage: 43.83%
	  Sally Ride | Votes: 2400 | Percentage: 25.54%
	  Alice Mix | Votes: 1678 | Percentage: 17.86%
	
	Losers:
	  Siyang Xiong | Votes: 841 | Percentage: 8.95%
	  Ahmed Mohamed | Votes: 359 | Percentage: 3.82%
	
	=== End of Results ===

Pass/Fail	1. Pass 2. Pass 3. Pass 4. Pass
Date	5/03/2025

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_001 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's constructor. This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, CandidateConstructorTests)

Automated: yes _X__ **no** ___

Results: Pass ___X___ **Fail** _____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A Candidate instance is initialized with a negative ID number	candidate("Name", -1, 0, false, false)	An invalid_agument exception should be thrown.	An invalid_argument was thrown and caught by the test.	
3	A candidate instance is initialized with an invalid amount of votes	candidate("Name", 0, -5, false, false)	An invalid_argument exception should be thrown.	An invalid_argument was thrown and caught by the test.	
4	A candidate instance is initialized with correct data	candidate("Name", 0, 0, false, false)	There should be no exceptions thrown.	There were no exceptions thrown.	
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and no candidates initialized with incorrect or invalid data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_002 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's getName() method. This test was defined in src/candidate_UT.cc

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, DisplayCandidateName)

Automated: yes X **no**

Results: Pass ___X__ **Fail**_____

Preconditions for Test: A candidate instance with correct data has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
3	The getName() method is then run on the testing data	std::string name = test_candidate1.getName()	getName() should return the same name that was initialized with the candidate	The names were equal.	
4					
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and the getVote() method returns the correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_003 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's getCandidateID() method.
This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, GetCandidateIDTest)

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: A candidate instance with correct data has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
2	The getCandidateID() method is then run on the testing data	int id = test_candidate1.getCandidateID()	getCandidateID() should return the correct integer value.	The integer returned by getCandidateID() was correct.	
3					
4					
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and the getCandidateID() method returns the correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_004 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's isWinner() method. This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, GetWinnerTest)

Automated: yes _X__ **no** ___

Results: Pass ___X__ **Fail**_____

Preconditions for Test: A candidate instance with correct data has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
3	The isWinner() method is then run on the testing data	bool win = test_candidate1.isWinner()	isWinner() should return the correct boolean value.	The boolean value returned by isWinner() was correct.	
4					
5					

Post condition(s) for Test:

There is a candidate instance initialized with correct data, and the isWinner() method returns the correct data.

Project Name: Project 1: Voting System Team# 4

Test Stage: Unit _X_ System __

Test Date: Mar 23, 2025

Test Case ID#: candidate_ut_005

Name(s) of Testers: Zoe Sepersky

Test Description:

This is a test of the Candidate class's isLoser() method. This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, GetLoserTest)

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: A candidate instance with correct data has been initialized.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
3	The isLoser() method is then run on the testing data	bool lose = test_candidate1.isLoser()	isLoser() should return the correct boolean value.	The boolean value returned by isLoser() was correct.	
4					
5					

Post condition(s) for Test:

There is a candidate instance initialized with correct data, and the isLoser() method returns the correct data.

Project Name: Project 1: Voting System Team# 4

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_006 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's getNumVotes() method.
This test was defined in src/candidate_UT.cc

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, GetVotesTest)

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: A candidate instance with correct data has been initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
3	The getNumVotes() method is then run on the testing data	int v = test_candidate1.getNumVotes()	getNumVotes() should return the correct integer value.	The integer returned by getNumVotes() was correct.	
4					
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and the getCandidateID() method returns the correct data.

Project Name: Project 1: Voting System Team# 4

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_007 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's updateVotes() method. This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, UpdateVotesTest)

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: A candidate instance with correct data has been initialized, and getNumVotes() works as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1		test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
2	A candidate instance is initialized with correct data				
3	The updateVotes() method is run on the testing data	test_candidate1.updateVotes(5)	The votes of test_candidate1 is updated by the specified value, in this case it is 5, for a total of 10	The votes returned by getNumVotes() after updateVotes() was correct.	
4	updateVotes() throws an exception when attempting to update votes by a negative integer.	test_candidate1.updateVotes(-1)	An std::out_of_range error is thrown.	An std::out_of_range error was thrown and caught by the test.	
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and the candidate's votes are updated by the correct amount.

Project Name: Project 1: Voting System Team# 4

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: candidate_ut_008 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Candidate class's setWinner() method. This test was defined in src/candidate_UT.cc

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, setWinnerTest)

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: A candidate instance with correct data has been initialized, and isWinner() works as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
2	The candidate's winner attribute is updated to true using setWinner().	test_candidate1.setWinner(true)	isWinner() should return the new updated value.	The boolean returned by isWinner() was correct.	
3	The candidate's winner attribute is updated back to false using setWinner().	test_candidate1.setWinner(false)	isWinner() should return the new updated value.	The boolean returned by isWinner() was correct.	
4	A candidate's winner attribute is attempted to be set to true when their loser attribute is already set to true.	Candidate loser = Candidate("Name", 0, 0, false, true)	std::invalid_argument should be thrown.	std::invalid_argument was thrown and caught by the test.	
5					

Post condition(s) for Test:
There is a candidate instance initialized with correct data, and the setWinner() attribute correctly updates the winner attribute of Candidate.

Project Name: Project 1: Voting System Team# 4

Test Stage: Unit _X_ System __

Test Date: Mar 23, 2025

Test Case ID#: candidate_ut_009

Name(s) of Testers: Zoe Sepersky

Test Description:

This is a test of the Candidate class's setWinner() method. This test was defined in src/candidate_UT.cc.

File: src/candidate_UT.cc
Function: TEST_F(candidateUnitTests, setLoserTest)

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: A candidate instance with correct data has been initialized, and isWinner() works as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A candidate instance is initialized with correct data	test_candidate1 = Candidate("Buffy Summers", 1, 5, false, false)	There should be no exceptions thrown	There were no exceptions thrown.	
2	The candidate's loser attribute is updated to true using setLoser().	test_candidate1.setLoser(true)	isLoser() should return the new updated value.	The boolean returned by isLoser() was correct.	
3	The candidate's loser attribute is updated back to false using setLoser().	test_candidate1.setLoserr(false)	isLoser() should return the new updated value.	The boolean returned by isLoser() was correct.	
4	A candidate's loser attribute is attempted to be set to true when their winner attribute is already set to true.	Candidate winner = Candidate("Name", 0, 0, true, loser)	std::invalid_argument should be thrown.	std::invalid_argument was thrown and caught by the test.	
5					

Post condition(s) for Test:

There is a candidate instance initialized with correct data, and the setLoser() attribute correctly updates the loser attribute of Candidate.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_001 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the Ballot class's constructor. Test code is stored in ballot_UT.cc in the src folder.

File: src/ballot_UT.cc
Function:TEST_F(BallotTest, BallotConstructor)

Automated: yes X no ___

Results: Pass X Fail

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A Ballot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 0} ballotID = 1	A Ballot object is created with the appropriate data.	A Ballot object is created with the appropriate data.	
3	The getVotes() method is used to compare the value of test_ballot.votes to the value of the input votes.	votes = {1, 0, 0, 0, 0, 0}	The output of getVotes() and votes are equal.	The output of getVotes() and votes are equal.	
4	The getIDs() method is used to compare the value of test_ballot.ballotID to the value of the input ballotID.	ballotID = 1	The output of getID() and ballotID are equal.	The output of getID() and ballotID are equal.	

Post condition(s) for Test:
There is a Ballot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 0} and test_ballot.ballotID = 1.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_002 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the Ballot class's getVotes() method.
Test code is stored in ballot_UT.cc in the src folder.

File: src/ballot_UT.cc
Function: TEST_F(BallotTest, GetVotes)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A Ballot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 0} ballotID = 1	A Ballot object is created with the appropriate data.	A Ballot object is created with the appropriate data.	
3	The getVotes() method is used to compare the value of test_ballot.votes to the value of the input votes.	votes = {1, 0, 0, 0, 0, 0}	The output of getVotes() and votes are equal.	The output of getVotes() and votes are equal.	

Post condition(s) for Test:
There is a Ballot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 0} and test_ballot.ballotID = 1.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_003 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the Ballot class's getID() method. Test code is stored in ballot_UT.cc in the src folder.

File: src/ballot_UT.cc
Function: TEST_F(BallotTest, GetBallotID)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A Ballot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 0} ballotID = 1	A Ballot object is created with the appropriate data.	A Ballot object is created with the appropriate data.	
3	The getID() method is used to compare the value of test_ballot.ballotID to the value of the input ballotID.	ballotID = 1	The output of getID() and ballotID are equal.	The output of getID() and ballotID are equal.	

Post condition(s) for Test:
There is a Ballot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 0} and test_ballot.ballotID = 1.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_004 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the PluralityBallot class's constructor. This test uses a standard plurality ballot format. Test code is stored in pluralityballot_UT.cc in the src folder.

File: src/pluralityballot_UT.cc
Function: TEST_F(PluralityBallotTest, NormalPBallotConstructor)

Automated: yes_X__ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A PluralityBallot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 0} ballotID = 1	A PluralityBallot object is created with the appropriate data.	A PluralityBallot object is created with the appropriate data.	
3	The getVotes() method is used to compare the value of test_ballot.votes to the value of the input votes.	votes = {1, 0, 0, 0, 0, 0}	The output of getVotes() and votes are equal.	The output of getVotes() and votes are equal.	
4	The getID() method is used to compare the value of test_ballot.ballotID to the value of the input ballotID.	ballotID = 1	The output of getID() and ballotID are equal.	The output of getID() and ballotID are equal.	
5	The getPreference() method is used to compare the value of test_ballot.prefernce to the correct preference value, which is 0.	preference = 0	The output of getPreference() and preference are equal.	The output of getPreference() and preference are equal.	

Post condition(s) for Test:
There is a PluralityBallot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 0}, test_ballot.ballotID = 1, and test_ballot.preference = 0.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_005 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the PluralityBallot class's constructor. This test uses an incorrect plurality ballot format (all votes are 0). Test code is stored in pluralityballot_UT.cc in the src folder.

File: src/pluralityballot_UT.cc
Function: TEST_F(PluralityBallotTest, AllZeroesPBallotConstructor)

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A PluralityBallot object (test_ballot) is created.	votes = {0, 0, 0, 0, 0, 0} ballotID = 1	The PluralityBallot constructor exits with the error: invalid_argument.	The PluralityBallot constructor exits with the error: invalid_argument.	
3					
4					
5					

Post condition(s) for Test:
There is no post condition.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_006 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the PluralityBallot class's constructor. This test uses an incorrect plurality ballot format (multiple 1 votes). Test code is stored in pluralityballot_UT.cc in the src folder.

File:src/pluralityballot_UT.cc
Function:TEST_F(PluralityBallotTest, MultipleVotesPBallotConstructor)

Automated: yes X **no**

Results: Pass X **Fail**

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A PluralityBallot object (test_ballot) is created.	votes = {1, 0, 1, 0, 1, 0} ballotID = 1	The PluralityBallot constructor exits with the error: invalid_argument.	The PluralityBallot constructor exits with the error: invalid_argument.	
3					
4					
5					

Post condition(s) for Test:
There is no post condition.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: **Unit** _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_007 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the PluralityBallot class's constructor. This test uses an incorrect plurality ballot format (a vote > 1). Test code is stored in pluralityballot_UT.cc in the src folder.

File:src/pluralityballot_UT.cc
Function:TEST_F(PluralityBallotTest, MultipleVotesPBallotConstructor2)

Automated: **yes** _X__ **no** ___

Results: **Pass** ___X___ **Fail** _____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A PluralityBallot object (test_ballot) is created.	votes = {0, 0, 3, 0, 0, 0} ballotID = 1	The PluralityBallot constructor exits with the error: invalid_argument.	The PluralityBallot constructor exits with the error: invalid_argument.	
3					
4					
5					

Post condition(s) for Test:
There is no post condition.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_008 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the PluralityBallot's getPreference() method. Test code is stored in pluralityballot_UT.cc in the src folder.

File:src/pluralityballot_UT.cc
Function:TEST_F(PluralityBallotTest, GetPreferenceTest)

Automated: yes _X__ **no** ___

Results: Pass ___X___ **Fail** _____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A PluralityBallot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 0} ballotID = 1	A PluralityBallot object is created with the appropriate data	A PluralityBallot object is created with the appropriate data	
3	The getPreference() method is used to compare the value of test_ballot.preference to the correct preference value, which is 0.	preference = 0	The output of getPreference() and preference are equal.	The output of getPreference() and preference are equal.	
4					
5					

Post condition(s) for Test:
There is a PluralityBallot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 0}, test_ballot.ballotID = 1, and test_ballot.preference = 0.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ **System** __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_009 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the STVBallot class's constructor. This test uses a standard STV ballot format. Test code is stored in stvballot_UT.cc in the src folder.

File:src/stvballot_UT.cc
Function: TEST_F(STVBallotTest, GetPreferenceTest)

Automated: yes_X__ **no** ___

Results: Pass ___X__ **Fail**_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A STVBallot object (test_ballot) is created.	votes = {1, 0, 2, 0, 0, 3} ballotID = 1	A STVBallot object is created with the appropriate data.	A STVBallot object is created with the appropriate data.	
3	The getVotes() method is used to compare the value of test_ballot.votes to the value of the input votes.	votes = {1, 0, 2, 0, 0, 3}	The output of getVotes() and votes are equal.	The output of getVotes() and votes are equal.	
4	The getID() method is used to compare the value of test_ballot.ballotID to the value of the input ballotID.	ballotID = 1	The output of getID() and ballotID are equal.	The output of getID() and ballotID are equal.	
5	The getPreference() method is used to compare the value of test_ballot.prefernce to the correct preference value, which is 0.	preference = 0	The output of getPreference() and preference are equal.	The output of getPreference() and preference are equal.	

Post condition(s) for Test:
There is a STVBallot object test_ballot with the variables test_ballot.votes = {1, 0, 2, 0, 0, 3}, test_ballot.ballotID = 1, and test_ballot.preference = 0.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_010 **Name(s) of Testers:** Annabelle Coler, Zoe Sepersky

Test Description:

This is a test of the STVBallot's getPreference() method.
Test code is stored in stvballot_UT.cc in the src folder.

File:src/stvballot_UT.cc
Function: TEST_F(STVBallotTest, GetPreferenceTest)

Automated: _yes_X___ no ___

Results: Pass X Fail

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A STVBallot object (test_ballot) is created.	votes = {1, 0, 2, 0, 0, 3} ballotID = 1	A STVBallot object is created with the appropriate data.	A STVBallot object is created with the appropriate data.	
2	The getPreference() method is used to compare the value of test_ballot.preference to the correct preference value, which is 0.	preference = 0	The output of getPreference() and preference are equal.	The output of getPreference() and preference are equal.	
3					
4					

Post condition(s) for Test:
There is a STVBallot object test_ballot with the variables test_ballot.votes = {1, 0, 2, 0, 0, 3}, test_ballot.ballotID = 1, and test_ballot.preference = 0.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 23, 2025

Test Case ID#: Ballot_UT_011 **Name(s) of Testers:** Anwesha Samaddar

Test Description:

This is a test of the STVBallot's getVotes() method to check if it throws an exception for an invalid ballot with less than half of the candidates ranked. Test code is stored in stvballot_UT.cc in the src folder.

File:src/stvballot_UT.cc
Function: TEST_F(STVBallotTest, InvalidSTVBallotConstructor)

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A STVBallot object (test_ballot) is created.	votes = {1, 0, 0, 0, 0, 3} ballotID = 1	A STVBallot object is created with the appropriate data.	A STVBallot object is created with the appropriate data.	
2	We check if the getVotes() method is throwing an exception for this invalid ballot with less than half of the candidates ranked.		The STVBallot constructor exits with the error: invalid_argument.	The STVBallot constructor exits with the error: invalid_argument.	
3					
4					

Post condition(s) for Test:
There is a STVBallot object test_ballot with the variables test_ballot.votes = {1, 0, 0, 0, 0, 3} and test_ballot.ballotID = 1

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __ **Test Date:** April 22, 2025

Test Case ID#: UserInterface_UT_001 **Name(s) of Testers:** Annabelle Coler

Test Description:
Verifies if getCsvFileName() returns the correct file location in the UserInterface class. Test code is stored in userinterface_UT.cc in the src folder.

Test file: src/userinterface_UT.cc
Function: TEST_F(UserInterfaceTest, GetCsvFileName)

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a UserInterface object (ui).		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getCsvFileName() method returns the same value as csv_file.	csv_file = "../testing/stv_ballots.csv"	getCsvFileName() method should return "../testing/stv_ballots.csv"	getCsvFileName() method returns "../testing/stv_ballots.csv".	
3					
4					

Post condition(s) for Test:

The UserInterface object ui has its CSV filename set to "../testing/stv_ballots.csv".

ui.csvFileName = "../testing/stv_ballots.csv"

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __

Test Date: April 22, 2025

Test Case ID#: UserInterface_UT_002

Name(s) of Testers: Annabelle Coler

Test Description:
This test verifies if the UserInterface class correctly sets the number of seats using getNumSeats(). Test code is stored in userinterface_UT.cc in the src folder.

Test file: src/userinterface_UT.cc
Function: TEST_F(UserInterfaceTest, GetNumSeatsTest)

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A UserInterface object (ui) is created.		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getNumSeats() method returns the same value as num_seats.	num_seats = 3	getNumSeats() should return 3.	getNumSeats() returns 3.	
3					
4					

Post condition(s) for Test:
The UserInterface object ui has the number of seats set to 3.

ui.numSeats = 3

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __

Test Date: April 22, 2025

Name(s) of Testers:
Annabelle Coler

Test Case ID#: UserInterface_UT_003

Test Description:
This test verifies if the UserInterface class correctly sets the audit file name using getAuditFileName() method. Test code is stored in userinterface_UT.cc in the src folder.

Test file:
src/userinterface_UT.cc
Function:
TEST_F(UserInterfaceTest, GetAuditFileNameTest)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A UserInterface object (ui) is created.		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getAuditFileName() method returns the same value as audit_file.	audit_file = "audit.txt"	getAuditFileName() Should return "audit.txt".	getAuditFileName() returns "audit.txt".	
3					
4					

Post condition(s) for Test:

The UserInterface object ui the audit filename set to "audit.txt".

ui.auditFileName = “audit.txt”

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __

Test Date: April 22, 2025

Name(s) of Testers:
Annabelle Coler

Test Case ID#: UserInterface_UT_004

Test Description:
This test verifies if the UserInterface class correctly sets the algorithm using getAlgorithm() method. Test code is stored in userinterface_UT.cc in the src folder.

Test file:
src/userinterface_UT.cc
Function:
TEST_F(UserInterfaceTest, GetAlgorithmTest)

Automated: yes X no ___

Results: Pass ___X__ Fail _____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A UserInterface object (ui) is created.		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getAlgorithm() method returns the same value as algorithm.	algorithm = "STV"	getAlgorithm() should return “STV”.	getAlgorithm() returns “STV”..	
3					
4					

Post condition(s) for Test:

The UserInterface object ui has the algorithm set to "STV".

ui.algorithm = “STV”

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __

Test Date: April 22, 2025

Name(s) of Testers:
Annabelle Coler

Test Case ID#: UserInterface_UT_005

Test Description:
This test verifies if the UserInterface class correctly sets the audit file name using getAuditFileName() method. Test code is stored in userinterface_UT.cc in the src folder.

Test file:
src/userinterface_UT.cc
Function:
TEST_F(UserInterfaceTest, GetAuditFileNameTest)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A UserInterface object (ui) is created.		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getAuditFileName() method returns the same value as audit_file.	audit_file = "audit.txt"	getAuditFileName() Should return "audit.txt".	getAuditFileName() returns "audit.txt".	
3					
4					

Post condition(s) for Test:

The UserInterface object ui the audit filename set to "audit.txt".

ui.auditFileName = “audit.txt”

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ **System** __

Test Date: April 22, 2025

Name(s) of Testers:
Annabelle Coler

Test Case ID#: UserInterface_UT_006

Test Description:
This test verifies if the UserInterface class correctly sets the shuffle option using getShuffleStv()method. Test code is stored in userinterface_UT.cc in the src folder.

Test file:
src/userinterface_UT.cc
Function:TEST_F(UserInterfaceTest, GetShuffleStvTest)

Automated: yes _X__ **no** ___

Results: Pass ___X__ **Fail**_____

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	A UserInterface object (ui) is created.		A UserInterface object is created with default values.	A UserInterface object is created with default values.	
2	Testing if the getShuffleStv() method returns the same value as shuffle.	shuffle = true	getShuffleSTV() should return true.	getShuffleSTV() returns true.	
3					
4					

Post condition(s) for Test:

The UserInterface object ui has the shuffle set to true.
ui.shuffle_stv = true

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit __ System _X_

Test Date: Mar 25, 2025
Name(s) of Testers:
Anwesha Samaddar

Test Case ID#: UserInterface_ST_001
Test Description:
This test validates if the user interface works as expected.

File: src/userinterface.cpp
within src folder
Function: getInfo()

Automated: yes___ no X___

Results: Pass X **Fail**

Preconditions for Test: No precondition

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Start election_app program through command prompt..		election_app is launched.	election_app is launched.	
2	Display user interface		User Interface is displayed and the system asks for location of ballots file.	User Interface is displayed and the system asks for location of ballots file.	
3					
4					

Post condition(s) for Test:
Program proceeds to take input from user for location of ballots file in csv format.

Project Name: Project 1: Voting System **Team #4**

Test Stage: Unit _X_ System __

Test Date: Mar 25, 2025
Name(s) of Testers: Anwesha Samaddar

Test Case ID#: UserInterface_ST_005
Test Description:
This test validates if the user interface handles the shuffle input correctly for the STV algorithm.

Test file: src/userinterface.cpp within src folder
Function: getInfo()

Automated: yes no X

Results: Pass X Fail

Preconditions for Test: The election_app program is running and the user has selected STV as the voting algorithm.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	System prompts the user to choose if they want to disable ballot shuffle or not.		User interface has this prompt displayed: "Disable ballot shuffle? (true/false):"	User interface has this prompt displayed: "Disable ballot shuffle? (true/false):"	
2	Enter invalid input (non-boolean value).	"abc"	System displays error message: "Invalid input. Please enter true or false."	System displays error message: "Invalid input. Please enter true or false."	
3	Enter valid input.	true	Program accepts input and proceeds.	Program accepts input and proceeds.	
4					

Post condition(s) for Test:

The program proceeds with the valid shuffle option or exits after 3 failed attempts with an error message.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit __ System _X_ **Test Date:** Mar 25, 2025

Name(s) of Testers:
Anwesha Samaddar

Test Case ID#: UserInterface_ST_006

Test Description:
This test validates if the user interface accepts the input for the audit file name.

File: src/userinterface.cpp within src folder

Automated: yes no X **Function: getInfo()**

Results: Pass X **Fail**

Preconditions for Test: The election_app program is running and user has given a valid input for voting algorithm and/ or ballot shuffle.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	System prompts the user to input audit file name.		User interface prompts: "Enter audit file name: "	User interface prompts: "Enter audit file name: "	
2	User enters the audit file name.	"audit.txt"	System accepts the file name and proceeds.	System accepts the file name and proceeds.	
3					
4					

Post condition(s) for Test:
The program proceeds to run the selected voting algorithm using the inputs entered by the user.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: election_UT_001
Test Description:

This is a test of the Election class's constructor. This test was defined in src/election_UT.cc

File: src/election_UT.cc
Function: Election()

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized.	Election test = Election("plurality_ballots.csv", "plurality", 3)	There should be no exceptions thrown	There were no exceptions thrown.	
3					
4					
5					

Post condition(s) for Test:
There is an Election instance initialized with correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __ **Test Date:** Mar 26, 2025

Test Case ID#: election_UT_002 **Name(s) of Testers:** Zoe Sepersky

Test Description:

This is a test of the Election class's getCSVFIleName() method. This test was defined in src/election_UT.cc

File: src/election_UT.cc
Function: getCSVFileName()

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: There is an election instance initialized with correct data.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized.	Election test = Election("plurality_ballots.csv", "plurality", 3)	There should be no exceptions thrown	There were no exceptions thrown.	
3	getCSVFileName() is then run on the initialized data.	std::string nameP = test.getCSVFileName()	getCSVFileName() should return the correct name of the csv file.	The correct name was returned.	
4					
5					

Post condition(s) for Test:
There is an Election instance initialized with correct data, and getCSVFileName() returns the correct name.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: election_UT_003
Test Description:

This is a test of the Election class's getNumSeats(). This test was defined in src/election_UT.cc

File: src/election_UT.cc
Function: getNumSeats()

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: There is an election instance initialized with correct data.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized.	Election pluralityTest = Election("plurality_ballots.csv", "plurality", 6)	There should be no exceptions thrown	There were no exceptions thrown.	
3	Another candidate instance is initialized with a different number of seats.	Election stvTest = Election("stv_ballots.csv", "stv", 5)	There should be no exceptions thrown.	There were no exceptions thrown.	
4	getNumSeats() is then run on the testing data.	int seats = stvTest.getNumSeats(); int seatP = pluralityTest.getNumSeats();	getNumSeats() returns the correct number of seats.	The correct number of seats were returned.	
5					

Post condition(s) for Test:
There is an Election instance initialized with correct data, and getNumSeats() returns the correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: election_UT_004
Test Description:

This is a test of the Election class's getAlgorithm() method. This test was defined in src/election_UT.cc

File: src/election_UT.cc
Function: getAlgorithm()

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: An Election instance is initialized with correct data.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized.	Election stvTest = Election("stv_ballots.csv", "plurality", 5)	There should be no exceptions thrown	There were no exceptions thrown.	
3	A candidate instance is initialized.	Election pluralityTest = Election("plurality_ballots.csv", "plurality", 6)	There should be no exceptions thrown	There were no exceptions thrown.	
4	getAlgorithm() is then run on the testing data.	std::string name1 = stvTest.getAlgorithm(); std::string name2 = pluralityTest.getAlgorithm();	getAlgorithm() should return the correct names for each algorithm.	The correct names were returned.	
5					

Post condition(s) for Test:
There is an Election instance initialized with correct data, and getAlgorithm() returned the correct values.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: election_UT_005
Test Description:

This is a test of the Election class's setBallots() method. This test was defined in src/election_UT.cc

File: src/election_UT.cc
Function: Election()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: There is a valid CSV file that can be accessed, and an Election instance initialized with correct data.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	A candidate instance is initialized.	Election stvTtest = Election("../testing/stv_ballots.csv", "stv", 3)	There should be no exceptions thrown.	There were no exceptions thrown.	
3	setBallots() is run on the correct data.	stvTest.setBallots();	Ballots in stv_ballots.csv are read in and set accordingly.	../../testing/stv_ballots.csv was successfully opened.	
4					
5					

Post condition(s) for Test:
There is an Election instance initialized with correct data, and ballots have been read in from the csv file.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: stv_UT_001
Test Description:

This is a test of the STV class's constructor. This test was defined in src/stv_UT.cc

File: src/stv_UT.cc
Function: STV()

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a vector of Candidate pointers, Ballots pointers, and the number of seats up for election is specified.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	An STV instance is initialized with correct data.	STV test(ballots, candidates, seats)	There should be no exceptions thrown	There were no exceptions thrown.	
3	getNumSeats() is then run on the testing data.	test.getNumSeats()	The correct number of seats should be returned.	The correct number of seats was returned.	
4	getBallots() is then run on the testing data.	test.getBallots()	The Ballot* vector should be returned.	The Ballot* vector was returned.	
5	getCandidates() is then run on the testing data.	test.getCandidates()	The Candidate* vector should be returned.	The Candidate* vector was returned.	

Post condition(s) for Test:
There is an STV instance initialized with correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: stv_UT_002
Test Description:

This is a test of the STV/Election class's getNames(). This test was defined in src/stv_UT.cc

File: src/stv_UT.cc
Function: TEST_F(STVTests, CandidateNameTest)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a vector of Candidate pointers, Ballots pointers, and the number of seats up for election is specified.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	An STV instance is initialized with correct data.	STV test(ballots, candidates, seats)	There should be no exceptions thrown	There were no exceptions thrown.	
3	getNames() is then run on the testing data.	test.getNames()	The names returned from getName() should be in the correct order of the Candidate* vector.	The names were in the correct order.	
4					
5					

Post condition(s) for Test:
There is an STV instance initialized with correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: stv_UT_003
Test Description:

This is a test of the STV/Election class's getNumBallots(). This test was defined in src/stv_UT.cc

File: src/stv_UT.cc
Function: TEST_F(STVTests, GetNumBallotsTest)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a vector of Candidate pointers, Ballots pointers, and the number of seats up for election is specified.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	An STV instance is initialized with correct data.	STV test(ballots, candidates, seats)	There should be no exceptions thrown	There were no exceptions thrown.	
3	getNumBallots() is then run on the testing data.	test.getNumBallots()	getNumBallots() should return the correct number of ballots..	getNumBallots() returned the correct number of ballots.	
4					
5					

Post condition(s) for Test:
There is an STV instance initialized with correct data.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _X_ System __

Test Date: Mar 26, 2025
Name(s) of Testers: Zoe Sepersky, Anwesha Samaddar

Test Case ID#: stv_UT_004
Test Description:

This is a test of the STV class's runElection() method. This test was defined in src/stv_UT.cc

File: src/stv_UT.cc
Function: TEST_F(STVTests, runElectionTest)

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a vector of Candidate pointers, Ballots pointers, and the number of seats up for election is specified.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	An STV instance is initialized with correct data.	STV test(ballots, candidates, seats)	There should be no exceptions thrown	There were no exceptions thrown.	
3	Empty vectors for the winners and losers are initialized and passed into runElection()	std::vector<Candidate*> winners; std::vector<Candidate*> losers; test.runElection(winners, losers)	runElection() should function normally and the list of winners and losers should match as expected.	The election ran normally and the lists of winners and losers matched the expected lists.	This test segfaulted when ran on Zoe's local laptop. but ran without complication when ran on Anwesha's laptop.
4					
5					

Post condition(s) for Test:
There is an STV instance initialized with correct data, and the election has been run.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit __ System _X_

Test Date: Mar 28, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: stv_sys_001
Test Description:

This is a test of the STV election when ran normally on normal data, with two seats up for election

Automated: yes ___ no _X_

Results: Pass X Fail

Preconditions for Test: There exists a CSV file with no data errors, and the user interface and election functionalities are in place and can run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	The election application is compiled using `make` and then ran using ./election_app	none	There should be no errors.	There were no errors during compilation.	There are warnings, but it does not affect functionality.
3	The testing CSV files are inputted, as well as the number of seats (2), STV algorithm is entered, shuffle is kept on, and audit file name is specified.	CSV file = ../testing/stv_ballots.csv Number of seats = 2 Algorithm = STV Disable shuffle = false Audit file name = audit.txt	There should be no errors when inputting data. The UI should recognize all input.	The user interface recognized all data and the election was run successfully.	
4	Inspect the election data outputted to the terminal	none	The two candidates elected should be A, C as tested by an outside STV calculator.	A and C were elected.	To determine which candidates would win the election, I used an outside calculator and input the same data as the CSV file passed into election_app. The calculator can be found here: https://paul-lockett.co.uk/stv.html audit.txt was generated in the root directory of the project.
5					

Post condition(s) for Test:
There is an STV instance initialized with correct data, and the election has been run. The correct winning candidates were outputted and election data was written to audit.txt.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit __ System _X_

Test Date: Mar 28, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: stv_sys_002
Test Description:

This is a test of the STV election when run on data where there is a tie and only one seat up for election.

Automated: yes___ no _X__

Results: Pass ___X___ Fail ___

Preconditions for Test: There exists a CSV file with no data errors, and the user interface and election functionalities are in place and can run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	The election application is compiled using `make` and then ran using ./election_app	none	There should be no errors.	There were no errors during compilation.	There are warnings, but it does not affect functionality.
3	The testing CSV files are inputted, as well as the number of seats, STV algorithm is entered, shuffle is kept on, and audit file name is specified.	CSV file = ../testing/stv_tie_ballots.csv Number of seats = 1 Algorithm = STV Disable shuffle = false Audit file name = audit.txt	There should be no errors when inputting data. The UI should recognize all input.	The user interface recognized all data and the election was run successfully.	
4	Inspect the election data outputted to the terminal	none	B and C should be in a tie throughout the election, with C ultimately winning the tiebreaker.	C was elected as the winner.	To determine which candidates would win the election, I used an outside calculator and input the same data as the CSV file passed into election_app. The calculator can be found here: https://paul-lockett.co.uk/stv.html
5					

Post condition(s) for Test:
There is an STV instance initialized with correct data, and the election has been run. The correct winning candidates were outputted and election data was written to audit.txt.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit __ System _X_

Test Date: Mar 28, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: plurality_sys_001
Test Description:

This is a test of a Plurality election when run on data where there is no tie and the election data is normal.

Automated: yes___ no _X__

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a CSV file with no data errors, and the user interface and election functionalities are in place and can run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	The election application is compiled using `make` and then ran using ./election_app	none	There should be no errors.	There were no errors during compilation.	There are warnings, but it does not affect functionality.
3	The testing CSV files are inputted, as well as the number of seats, plurality algorithm is entered, and audit file name is specified.	CSV file = ../testing/plurality_ballots_regular.csv Number of seats = 1 Algorithm = Plurality Audit file name = audit.txt	There should be no errors when inputting data. The UI should recognize all input.	The user interface recognized all data and the election was run successfully.	
4	Inspect the election data outputted to the terminal	none	Candidate A should be the elected winner.	A was elected as the winner.	To determine the winner of a plurality election beforehand, the votes were tallied by hand and those winners were compared to the application's output.
5	The election is ran again on the same data except the number of seats is 2	Same as step 3 but number of seats = 2	Candidates A and C should be the elected winners	A and C were the elected winners.	

Post condition(s) for Test:
There is a Plurality election instance initialized, the election was run and information about the election was outputted to the terminal and the specified audit file.

Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit __ System _X_

Test Date: Mar 28, 2025
Name(s) of Testers: Zoe Sepersky

Test Case ID#: plurality_sys_002
Test Description:

This is a test of a Plurality election when run on data where there is a tie.

Automated: yes___ no _X__

Results: Pass ___X__ Fail_____

Preconditions for Test: There exists a CSV file with no data errors, and the user interface and election functionalities are in place and can run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	The election application is compiled using `make` and then ran using ./election_app	none	There should be no errors.	There were no errors during compilation.	There are warnings, but it does not affect functionality.
3	The testing CSV files are inputted, as well as the number of seats, plurality algorithm is entered, and audit file name is specified.	CSV file = ../testing/plurality_ballots_tie_2.csv Number of seats = 1 Algorithm = Plurality Audit file name = audit.txt	There should be no errors when inputting data. The UI should recognize all input.	The user interface recognized all data and the election was run successfully.	
4	Inspect the election data outputted to the terminal	none	Candidate A should be the elected winner, according to the tiebreaking algorithm..	A was elected as the winner.	To determine the winner of a plurality election beforehand, the votes were tallied by hand and those winners were compared to the application's output.
5	The election is ran again on the same data except the number of seats is 2	Same as step 3 but number of seats = 2	Candidates A and B should be the elected winners	A and B were the elected winners.	

Post condition(s) for Test:
There is a Plurality election instance initialized, the election was run and information about the election was outputted to the terminal and the specified audit file.