System Test Logs

Testing Manually with Terminal and Audit File

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ **System** ___✓___

Test Date: Mar 22, 2024

Test Case ID#: S_0001

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run the program & inspect manually

Automated: yes ______ **no** ___✓___

File: ../src/main.cpp
Methods: System test on CPL Election

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make run”				User is prompted to enter file
2	Enter file name	../testing/test_cpl.csv			
3	Program checks that file exists, counts votes, allocates seats, handles ties between parties for the seats, and chooses the winning candidates.				For an tie that occurs, information is printed on the terminal about the process and outcome.
4	Check that text is displayed to the terminal, and that the audit file is created.				Terminal table should match the table in the audit file.

Post condition(s) for Test:
Results are printed to the terminal, and an audit file is generated.

Testing Manually with Terminal and Audit File

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ **System** ___✓___

Test Date: Mar 22, 2024

Test Case ID#: S_0002

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run the program & inspect manually

Automated: yes ______ **no** ___✓___

File: ../src/main.cpp
Methods: System test on CPL Class

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make run FILE=../testing/test_cpl.csv”	../testing/test_cpl.csv			User gives filename as CMD line argument
2	Program checks that file exists, counts votes, allocates seats, handles ties between parties for the seats, and chooses the winning candidates.				For any tie that occurs, information is printed on the terminal about the process and outcome.
3	Check that text is displayed to the terminal, and that the audit file is created.				Terminal table should match the table in the audit file.

Post condition(s) for Test:
Results are printed to the terminal, and an audit file is generated.

Testing Manually with Terminal and Audit File

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ System ___✓___ **Test Date:** Mar 22, 2024

Test Case ID#: S_0003

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run the program & inspect manually

Automated: yes ______ no ___✓___

File: ../src/main.cpp
Methods: System test on OPL Class

Results: Pass ___✓___ Fail ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make run”				User is prompted to enter file
2	Enter file name	../testing/test_opl.csv			
3	Program checks that file exists, counts votes, allocates seats, handles ties between parties & candidates for the seats.				For any tie that occurs, information is printed on the terminal about the process and outcome.
4	Check that text is displayed to the terminal, and that the audit file is created.				Terminal table should match the table in the audit file.

Post condition(s) for Test:
Results are printed to the terminal, and an audit file is generated.

Testing Manually with Terminal and Audit File

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ **System** ___✓___

Test Date: Mar 22, 2024

Test Case ID#: S_0004

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run the program & inspect manually

Automated: yes ______ **no** ___✓___

File: ../src/main.cpp
Methods: System test on OPL Class

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make run FILE=../testing/test_opl.csv”	../testing/test_opl.csv			User gives filename as CMD line argument
2	Program checks that file exists, counts votes, allocates seats, handles ties between parties & candidates for the seats.				For any tie that occurs, information is printed on the terminal about the process and outcome.
3	Check that text is displayed to the terminal, and that the audit file is created.				Terminal table should match the table in the audit file.

Post condition(s) for Test:
Results are printed to the terminal, and an audit file is generated.

System Testing with Google Tests from now on, unless otherwise specified above the table

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ System ___✓___

Test Date: Mar 22, 2024

Test Case ID#: S_0005

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run system test on OPL

Automated: yes ______ no ___✓___

File: ../testing/_opl.csv
Methods: System test on OPL Election

Results: Pass ___✓___ Fail ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make systemtest”	_opl.csv			Testing OPL no tie
3	Checks if display results matches expected output		TRUE	TRUE	

Post condition(s) for Test:
Test results are printed to the terminal, all tests pass

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ **System** ___✓___

Test Date: Mar 22, 2024

Test Case ID#: S_0006

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run system test on OPL (tie case)

Automated: yes ______ **no** ___✓___

File: ../testing/_opl5.csv
Methods: System test on OPL Election

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make systemtest”	_opl5.csv			Test checks if tie is handled successfully
3	Checks if display results matches expected output		TRUE	TRUE	

Post condition(s) for Test:
Test results are printed to the terminal, all tests pass

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ System ___✓___ **Test Date:** Mar 22, 2024

Test Case ID#: S_0007

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run system test for CPL

Automated: yes ______ no ___✓___

File: ../testing/test_cpl.csv
Methods: System test on CPL Election

Results: Pass ___✓___ Fail ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make systemtest”	test_cpl.csv			Testing CPL no tie
3	Checks if display results matches expected output		TRUE	TRUE	

Post condition(s) for Test:
Test results are printed to the terminal, all tests pass.

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ______ System ___✓___ **Test Date:** Mar 22, 2024

Test Case ID#: S_0008

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Run system test for cpl tie case

Automated: yes ______ no ___✓___

File: ../testing/_cpl.csv
Methods: System test on CPL Election

Results: Pass ___✓___ Fail ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make systemtest”	_opl.csv			Test checks if tie is handled successfully
3	Checks if display results matches expected output		TRUE	TRUE	

Post condition(s) for Test:
Test results are printed to the terminal, all tests pass

Unit Testing Below

Pages left empty on purpose

Unit Test Logs

Candidate Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_001

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Candidate default constructor and getName() function

Automated: yes ______ **no** ___✓___

File: ../src/Candidate.cpp
Methods: Candidate()
getName()
getID()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
None

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Candidate object instantiated				
3	Checks that this Candidate’s name is “No Name”		True	True	
4	Checks that this Candidate’s IDis 0		True	True	

Post condition(s) for Test:
Candidate object initiated with no name.

Candidate Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: __✓____ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_002

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Candidate overridden constructor and getName() function

Automated: yes ______ **no** ___✓___

File: ../src/Candidate.cpp
Methods: getName()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Candidate object instantiated with name and ID	“John Doe”, 123			
3	Checks that this Candidate’s name is “John Doe”		True	True	
4	Checks that this Candidate’s IDis 123		True	True	

Post condition(s) for Test:
Candidate object initiated with a name.

Candidate Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: __✓____ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_003

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Candidate overridden constructor and getName() function

Automated: yes ______ **no** ___✓___

File: ../src/Candidate.cpp
Methods: getName()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Candidate object instantiated with name and an ID	“Jane Smith”, 123			
3	Checks that this Candidate’s name is “Jane Smith”		True	True	

Post condition(s) for Test:
Candidate object initiated with a name and an ID.

Candidate Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: __✓____ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_004

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Candidate overridden constructor and getID() function

Automated: yes ______ **no** ___✓___

File: ../src/Candidate.cpp
Methods: getID()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Candidate object instantiated with name and an ID	“Jane Smith”, 123			
3	Checks that this Candidate’s ID is “123”		True	True	

Post condition(s) for Test:
Candidate object initiated with a name and an ID.

Party Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_005

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Default Constructor

Automated: yes ______ **no** ___✓___

File: ../src/Party.cpp
Methods: Party Class Default Constructor

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File is in the correct directory

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create Party object with no parameter				
3	Use method getName() and expect value “No Name”		True	True	

Post condition(s) for Test:
Result produced is expect to be the same as expected result

Party Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_006

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Constructor

Automated: yes ______ **no** ___✓___

File: ../src/Party.cpp
Methods: Party Class Constructor with Parameter

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File is in the correct directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create Party object with a parameter name	“Democrat Party”			
3	Use method getName() and expect with value of parameter		True	True	

Post condition(s) for Test:
Result produced is expect to be the same as expected result

Party Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_007

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Constructor

Automated: yes ______ **no** ___✓___

File: ../src/Party.cpp
Methods: Party Class Constructor with Parameter

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File is in the correct directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create Party object with parameter	“Republican Party”			
3	Use method getName() and expect with value of parameter		True	True	

Post condition(s) for Test:
Result produced is expect to be the same as expected result

Party Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_008

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Multiple Parties

Automated: yes ______ **no** ___✓___

File: ../src/Party.cpp
Methods: Party Class Constructor with Parameter

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File is in the correct directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create Parties object with parameter (have the same names)	"Republican Party" "Republican Party"			
3	Use method getName() and expect with value of parameter for both object (string) are equal		True	True	

Post condition(s) for Test:
Result produced is expect to be the same as expected result

Party Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_009

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test Multiple Parties

Automated: yes ______ **no** ___✓___

File: ../src/Party.cpp
Methods: Party Class Constructor with Parameter

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File is in the correct directory

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create Parties object with parameter (have different names)	"Republican Party" "Democratic Party"			
3	Use method getName() and expect with value of parameter for both object (string) are equal		False	Flase	

Post condition(s) for Test:
Result produced is expect to be the same as expected result

Tie Breaker Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_010

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test verifies the default constructor of the TieBreaker class.

Automated: yes ______ **no** ___✓___

File: ../src/TieBreaker.cpp
Methods: TieBreaker() (Default Constructor)

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create an instance using default constructor				
3	Returned value should be empty		True	True	

Post condition(s) for Test:
TieBreaker object is successfully created

Tie Breaker Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_011

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test case verifies the breakTies method of the tiebreaker class when there's only one candidate and seat available

Automated: yes ______ **no** ___✓___

File: ../src/TieBreaker.cpp
Methods: TieBreaker()
breakTies(numPeople, numPeople)

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run "make unittest"				
2	Calls breakTies method with arguments '1'candidate and '1' seat	1, 1			
3	Check if the tieBreakerResult is 0		0	0	

Post condition(s) for Test:
The tieBreakerResult attribute is 0

Tie Breaker Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_012

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test case verifies the BreakTies method of TieBreaker class when there are multiple candidates and seats available

Automated: yes ______ **no** ___✓___

File: ../src/TieBreaker.cpp
Methods: TieBreaker()
breakTies(numSeats, numPeople)

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
File must be formatted correctly.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Calls breakTies method on ‘5’ candidates and ‘2’ seats	5, 2			
3	Checks if tieBreakerResult is not Empty		Not Empty	Filled	A string is returned, but the winners are randomly selected

Post condition(s) for Test:
The tieBreakerResult is populated with the indices of the winning candidate. It should not be empty

Result Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_013

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test verifies the setters of the class work with dummy data

Automated: yes ______ **no** ___✓___

File: ../src/Results.cpp
Methods: setDisplayInfo(string)
setAuditInfo(string)
getDisplayInfo()
getAuditInfo()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
Results class object should be instantiated.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create an instance using default constructor				
3	Pass random string into the setters	Random chars			
4	Get the random string from the Results class				
5	Compare the passed string to returned string		True	True	

Post condition(s) for Test:
None

Result Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_014

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test verifies the setters of the class work with empty data

Automated: yes ______ **no** ___✓___

File: ../src/Results.cpp
Methods: setDisplayInfo(string)
setAuditInfo(string)
getDisplayInfo()
getAuditInfo()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
Results class object should be instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create an instance using default constructor				
3	Pass random string into the setters	Empty string			
4	Get the random string from the Results class				
5	Compare the passed string to returned string		True	True	

Post condition(s) for Test:
None

CPL Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ System ______

Test Date: Mar 22, 2024

Test Case ID#: U_015

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
Test CPLSystem constructor and displayResults() method when no file is given

Automated: yes ______ no ___✓___

File: ../src/CPLSystem.cpp
Methods: CPLSystem(), displayResults()

Results: Pass ___✓___ Fail ______

Preconditions for Test:
None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create CPLSystem object with default constructor	0 Seats & 0 Votes	Empty Table	Empty Table	
3	Call displayResults() method and check the output				
4					
5					

Post condition(s) for Test:
CPLSystem object is successfully created with default values

CPL Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_016

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:

Automated: yes ______ **no** ___✓___

File: ../testing/CPLSystem.cpp
Methods: CPLSystem(), auditResults()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
None

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create CPLSystem object with default constructor	0 Parties & 0 Seats	Empty Table	Empty Table	
3	Call auditResults() method and check the output. There were 0 parties				
4					
5					

Post condition(s) for Test:
CPLSystem object is succesfully created with default values

CPL Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_017

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test verifies the setters of the class work with empty data

Automated: yes ______ **no** ___✓___

File: ../testing/CPLSystem.cpp
Methods: CPLSystem(),
countVotes(),
allocateSeats(),
displayResults(),
auditResults(),

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
Test CPLSystem class with an empty file.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”	0 Seats			
2	Check that program reads file and initializes CPLSystem Object	0 Votes			
3	Call countVotes() and allocateSeats() methods				
4	Call displayResults() and auditResults() methods and		Empty Table	Empty Table	
5	Check outputs		Empty Table	Empty Table	

Post condition(s) for Test:
Results are printed to terminal and audit file is generated

CPL Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_018

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test verifies the setters of the class work with empty data

Automated: yes ______ **no** ___✓___

File: ../src/Results.cpp
Methods: setDisplayInfo(string)
setAuditInfo(string)
getDisplayInfo()
getAuditInfo()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
Results class object should be instantiated.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Run “make unittest”				
2	Create an instance using default constructor				
3	Pass random string into the setters	Empty string			
4	Get the random string from the Results class				
5	Compare the passed string to returned string		True	True	

Post condition(s) for Test:
None

OPL System Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_019

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:

This test checks that an OPL object has a basic terminal output, and the values it has access to are equal to 0.

Automated: yes ______ **no** ___✓___

File: ../src/OPLSystem.cpp
Methods: displayResults()
getSeats()
getVotes()
getCandidates()
getParties()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
OPLSystem object should be instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run "make unittest"				
2	Make a default string & table that would be displayed to the terminal if there was no election data	None			
3	Check that the string and display results match.		TRUE	TRUE	
4	Check that the values OPL has access to are 0		TRUE	TRUE	

Post condition(s) for Test:
None

OPL System Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_020

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:

This test checks that an OPL object has a basic audit file output, and the values it has access to are equal to 0.

Automated: yes ______ **no** ___✓___

File: ../src/OPLSystem.cpp
Methods: auditResults()
getSeats()
getVotes()
getCandidates()
getParties()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
OPLSystem object should be instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Make a default string & table that would be written to the audit file if there was no election data	None			
3	Check that the string and audit content match.		TRUE	TRUE	
4	Check that the values OPL has access to are 0		TRUE	TRUE	

Post condition(s) for Test:
None

OPL System Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_021

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test checks that the OPL system would produce empty tables for the terminal and the audit file. It should also not change any of the variables that it has access to.

Automated: yes ______ **no** ___✓___

File: ../src/OPLSystem.cpp
Methods: OPLSystem(filename)
displayResults()
auditResults()
getSeats()
getVotes()
getCandidates()
getParties()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
OPLSystem object should be instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run “make unittest”				
2	Create an OPL class instance, and give it this file to read.	../testing/_opl0.csv			This file has only 0 values, no parties/candidates, and no votes.
3	Make a default string & table that would be displayed to the terminal if there was no election data				
4	Make a default string & table that would be written to the				

	audit file if there was no election data				
5	Check that the strings mentioned above match the terminal output and audit file strings produced.		TRUE	TRUE	
6	Check that the values OPL has access to are 0		TRUE	TRUE	
Post condition(s) for Test: None					

OPL System Class Testing

Project Name: Project 1: Voting System **Team07**

Test Stage: Unit: ___✓___ **System** ______

Test Date: Mar 22, 2024

Test Case ID#: U_022

Name(s) of Testers:
Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali (ali00429)

Test Description:
This test checks that the OPL system would produce empty tables for the terminal and the audit file. It should also not change any of the variables that it has access to.

Automated: yes ______ **no** ___✓___

File: ../src/OPLSystem.cpp
Methods: OPLSystem(filename)
displayResults()
auditResults()
getSeats()
getVotes()
getCandidates()
getParties()

Results: Pass ___✓___ **Fail** ______

Preconditions for Test:
OPLSystem object should be instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run "make unittest"				
2	Create an OPL class instance, and give it this file to read.	../testing/_opl.csv			This file has only 0 values, no parties/candidates, and no votes.
3	Make a string & table that would be displayed to the terminal after this file is processed				
4	Make a default string & table that would be written to the				

	audit file after this file is processed				
5	Check that the strings mentioned above match the terminal output and audit file strings produced.		TRUE	TRUE	
6	Check that the values OPL has access to are 0		TRUE	TRUE	

Post condition(s) for Test:
None