Electronic Voting

Unit Test Plan and Report

Andreas du Preez 12207871

Azhar Mohungoo 12239799

Gift Sefako 12231097

Stakeholders

Epi-Use Advance Roelof Nuade

Contents

1 Introduction

Software testing is performed to verify that the completed software package functions according to the expectations defined by our requirements/specifications.
EVoting consists of different components, each designed to do a specific seperate task. It is important to test each component individually to insure that they do what is expected from them before building the project as a whole and executing it. Tests are therefore essential to our system to prevent bugs, compile time and run time errors.

1.1 Purpose

EVoting is an electronic voting system, designed specifically for government elections, but it can also be used in other scenarios where reliable electronic voting is needed. Since it is crutial that the voting procesess cannot be tampered with in any way, both reliability and security is of main importance. It is there extremely important to test the crutial parts of the system to ensure the system executes as expected and eliminate unexpected security and reliability threats.

1.2 Scope

The scope of this document is structured as follows:
In this document, we elaborate on the purpose as well as the specific details to why we are testing certain funtionality. The functionality that are considered for testing are listed in section 2. Tests that have been identied from the requirements are discussed in detail in later section. Furthermore, this document outlines the test environment and the risks involved in the testing approaches that will be followed. Assumptions and dependencies of this test plan will also be mentioned. **??** outlines, discusses and concludes on the results of the tests.

1.3 Test Environment

Our testing environment is as follows:

1.3.1 Programming Languages:

Java version 1.8+

1.3.2 Testing Framework:

JUnit Testing framework.
Maven - Testing phase. Spring Boot.

1.3.3 Coding Environment:

IntelliJ Ultimate Edition 2016.1

1.3.4 Operating Systems:

Windows 10
Mac OS
Android version 4.4+

1.3.5 Internet Browsers:

HTML5 supported web browsers.
Firefox
Chrome
Safari

1.3.6 Database:

PostgreSQL

1.4 Assumptions and Dependencies

Assumptions:

- Server-side, client-side and the Blockchain are running on the same network.
- Each OS doesn't have a firewall blocking any of the incoming and outgoning messages as well as ports used by the system.
- The Blockchain is running (at least 1 Admin Node).
- The database is reachable by the system.

Dependencies:

- The Blockchain is reachable by the system.
- Network IP range and mask: 192.168.1.0/255.255.255.0.
- Each Blockchain node and the webservice all have static IP addresses.

Unit Test Plan

2 Functional Features to be Tested

The aspects that will be tested and evaluated in these tests are described in the following sections. The items that are tested is the crutial fundamental use cases that are required of the system to function at a core level. These use cases are specified in the section below.

The following is a list of features to be tested:

- Blockchain Module:
  - Connection to the blockchain and general module configuration.
  - Be able to send a vote from and to different Nodes.
  - Be able to get the vote balance of a Node.
- Database Module:
  - Validate a voter's login credentials via the webserver.
  - Validate a political party's login credentials via the webserver.
  - A voter's active state after being activated via the webserver.
  - Add an Admin into the database via the webserver.
  - Add an Activater into the database via the webserver.
  - Add a Voter into the database via the webserver.
- Voter Module
  - Register a new voter via the webservice.

3 Test Cases

3.1 Blockchain Module

3.1.1 Test Case 1: ConfigurationAndConnection

3.1.1.1 Condition 1: InvalidIPandPort

3.1.1.1.1 Objective: The purpose of this test is to see if the system can detect if the specified Node IP address and port are invalid or empty.

3.1.1.1.2 Input:

- An incorrect IP address string and/or
- An incorrect port address string.
- A correct RPC username string.
- A correct RPC password string.
- A correct Node address (to address).
- A positive interger amount.

3.1.1.1.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<BlockchainErrorMessages.InvalidIP> or "result":<BlockchainErrorM
  respectively where the values in brackets are strings defined in the Blockchain-ErrorMessages enumerate.

3.1.1.2 Condition 2: InvalidRPCCredentials

3.1.1.2.1 Objective: The purpose of this test is to see if the system can detect if the specified RPC username and/or password are incorrect.

3.1.1.2.2 Input:

- A correct IP address string.
- A correct port address string.
- An incorrect RPC username string and/or
- An incorrect RPC password string.
- A correct Node address (to address).
- A positive interger amount.

3.1.1.2.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<BlockchainErrorMessages.InvalidRPCCredentials> where the value in brackets is a string defined in the BlockchainErrorMessages enumerate.

3.1.2 Test Case 2: sendVote

3.1.2.1 Condition 1: InvalidToAddress

3.1.2.1.1 Objective: The purpose of this test is to see if the system can detect wether votes are trying to be sent to an invalid Node address (i.e. an address that doesn't exist in the Blockchain).

3.1.2.1.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or
- A correct RPC password string.
- An incorrect Node address (to address).
- A positive interger amount.

3.1.2.1.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<BlockchainErrorMessages.InvalidToAddress> where the value in brackets is a string defined in the BlockchainErrorMessages enumerate.

3.1.2.2 Condition 2: InvalidVotesLeft

3.1.2.2.1 Objective: The purpose of this test is to see if the system can detect wether a voting Node is trying to send a vote when its vote balance is 0.

3.1.2.2.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or
- A correct RPC password string.
- A correct Node address (to address).
- A positive interger amount.

Existing Entity: The voting Node's vote balance is 0.

3.1.2.2.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<BlockchainErrorMessages.InvalidVotesLeft> where the value in brackets is a string defined in the BlockchainErrorMessages enumerate.

3.1.2.3 Condition 3: SendVoteFromAdminToNode

3.1.2.3.1 Objective: The purpose of this test is to see if the system can successfully send 1 vote from the Admin Node to any other Node.

3.1.2.3.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or
- A correct RPC password string.
- A correct Node address (to address).
- A positive interger amount.

3.1.2.3.3 Outcome:

- JSON Object returned with a key-value pair of "success":"true".
- Also a key-value pair of "result":<TransactionHash> where the value in brackets is a hash value to identify the transaction.

3.1.2.4 Condition 4: SendVoteFromNodeToAdmin

3.1.2.4.1 Objective: The purpose of this test is to see if the system can successfully send 1 vote from a Node other than the Admin to the Admin Node.

3.1.2.4.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or

- A correct RPC password string.
- A correct Node address (to address).
- A positive interger amount.

3.1.2.4.3 Outcome:

- JSON Object returned with a key-value pair of "success":"true".
- Also a key-value pair of "result":<TransactionHash> where the value in brackets is a hash value to identify the transaction.

3.1.3 Test Case 3: GetBalance

3.1.3.1 Condition 1: GetAdminBalance

3.1.3.1.1 Objective: The purpose of this test is to see if the system can successfully retrieve the vote balance of the Admin Node.

3.1.3.1.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or
- A correct RPC password string.

3.1.3.1.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<Balance> where the value in brackets is an interger of the balance.

3.1.3.2 Condition 2: GetNodeBalance

3.1.3.2.1 Objective: The purpose of this test is to see if the system can successfully retrieve the vote balance of any other Node other than the Admin Node.

3.1.3.2.2 Input:

- A correct IP address string.
- A correct port address string.
- A correct RPC username string and/or
- A correct RPC password string.

3.1.3.2.3 Outcome:

- JSON Object returned with a key-value pair of "success":"false".
- Also a key-value pair of "result":<Balance> where the value in brackets is an interger of the balance.

3.2 Database Module

3.2.1 Test Case 1: ValidateUser

3.2.1.1 Condition 1: InvalidInput

3.2.1.1.1 Objective: The purpose of this test is to see if the database module's validateUser process can detect if invalid credentials have been entered.

3.2.1.1.2 Input:

- An ID number string.
- An invalid password.

3.2.1.1.3 Outcome:

- A boolean "False" value returned from database module.

3.2.1.2 Condition 2: ValidInput

3.2.1.2.1 Objective: The purpose of this test is to see if the database module's validateUser process can detect if valid credentials have been entered.

3.2.1.2.2 Input:

- An ID number string.
- A valid password.

3.2.1.2.3 Outcome:

- A boolean "True" value returned from database module.

3.2.2 Test Case 2: ValidateParty

3.2.2.1 Condition 1: InvalidInput

3.2.2.1.1 Objective: The purpose of this test is to see if the Database Module's validateParty process can detect if invalid credentials have been entered.

3.2.2.1.2 Input:

- An ID number string.
- An invalid password.

3.2.2.1.3 Outcome:

- A boolean "False" value returned from database module.

3.2.2.2 Condition 2: ValidInput

3.2.2.2.1 Objective: The purpose of this test is to see if the Database Module's validateParty process can detect if valid credentials have been entered.

3.2.2.2.2 Input:

- An ID number string.
- A valid password.

3.2.2.2.3 Outcome:

- A boolean "True" value returned from database module.

3.2.3 Test Case 2: ActivateVoter

3.2.3.1 Condition 1: InvalidInput

3.2.3.1.1 Objective: The purpose of this test is to see if the Database Module's activateVoter process can successfully change the active state of a voter.

3.2.3.1.2 Input:

- An ID number string.
- An invalid password.

3.2.3.1.3 Outcome:

- A boolean "True" value returned from database module.

3.2.4 Test Case 3: AddVoter

3.2.4.1 Condition 1: ValidInput

3.2.4.1.1 Objective: The purpose of this test is to see if the Database Module's addVoter process can successfully add a new voter into the database.

3.2.4.1.2 Input:

- An ID number string.
- A password string.
- A name string.
- A surname string.
- A location string.
- A mobile number string.
- An email string.
- An integer with their available votes.
- A boolean value indicating if they have already voted for national election.
- A boolean value indicating if they have already voted for provintial election.
- A boolean value indicating if the user is active.

3.2.4.1.3 Outcome:

- A boolean "True" value returned from database module.