TEST PLAN

VIRTUAL VOTING MACHINE

ChangeLog *:*

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made
001	20.11.2023	Surya Pratap Singh Utkarsh Mishra	Initial Draft

1 Introduction

In an era characterized by unprecedented technological advancements, the traditional methods of conducting elections have come under increasing scrutiny. The digital age has ushered in the potential for innovation in the electoral process, promising greater transparency, security, and accessibility. One groundbreaking development that has garnered significant attention is the integration of blockchain technology into the realm of voting, giving rise to what is commonly referred to as "virtual voting." Blockchain, originally conceived as the underlying technology for cryptocurrencies like Bitcoin, has demonstrated its ability to revolutionize various industries, and elections are no exception.

The concept of virtual voting using blockchain technology is both revolutionary and disruptive. It offers the promise of a democratic system that is more secure, transparent, and efficient. This novel approach combines the security features of blockchain, such as decentralization and cryptographic protection, with the accessibility and convenience of a virtual voting platform. As a result, it has the potential to address longstanding concerns related to electoral integrity, voter fraud, and accessibility.

1.1 Scope

1.1.1 In Scope

Scope defines the features, functional or non-functional requirements of the software that **will be** tested. Features of the Project:

1. Candidates registration: the number of candidates registered for the election should be between 2 to 9 less than that or greater than that is not allowed.

2. Address mapping: Every registered voter, candidates and election commission have their address mapped corresponding to every action taken by them to check the security and other aspects required for virtual voting machines.

3. User interface: The virtual voting machine would need to provide a user-friendly interface for users and candidates to register and casting of vote should be smooth.

4. Security: The machine uses blockchain and smart contracts which is highly secure and cannot be tampered in any circumstances.

1.1.2 Out of Scope

Out Of Scope defines the features, functional or non-functional requirements of the software that **will NOT be** tested :

1. Scalability:: Load testing to ensure the platform can handle increased user loads

efficiently.

1.2 Quality Objective

Here make a mention of the overall objective that you plan to achieve without your testing Some objectives of your testing project could be
Ensure the Application Under Test conforms to functional and nonfunctional requirements.
Ensure the AUT meets the quality specifications defined by the client.
Bugs/issues are identified and fixed before go live

1.3 Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

- QA Analyst : Surya Pratap singh
- Test Manager : Prof. Shreela Pareek
- Configuration Manager: Prof. Neha Shukla
- Developers : Utkarsh Mishra, Surya Pratap Singh, Adrika Tripathi
- Installation Team :Prof. Shreela Pareek, Prof. Neha Shukla,Utkarsh Mishra, Surya Pratap Singh, Adrika Tripathi

2 Test Methodology

2.1 Overview

We are using an iterative testing approach to make sure our project works well. This means we test it in small steps, starting with checking if each part works on its own. Then, we see how different parts work together.

We keep testing as we make changes and add new things. This way, we make sure our project is always working well, even after modification.

2.2 Test Levels

Test Levels define the Types of Testing to be executed on the Application Under Test (AUT).
We aim to test our project at the following levels :

1) Unit Testing: This is the lowest level of testing and focuses on individual components or functions within the software. Developers often perform unit tests to verify that specific parts of the code work correctly.

2) Integration Testing: This level of testing checks how different components or modules of the software work together. It ensures that integrated parts of the software function as intended.

3) System Testing: At this level, the entire system is tested as a whole. It verifies that the software meets its specified requirements and functions properly in its intended environment.

2.3 Test Completeness

Here you define the criterias that will deem your testing complete.
For instance, a few criteria to check Test Completeness would be

- 100% test coverage
- All Manual & Automated Test cases executed
- All open bugs are fixed or will be fixed in next release

3 Test Deliverables

Here are the deliverables

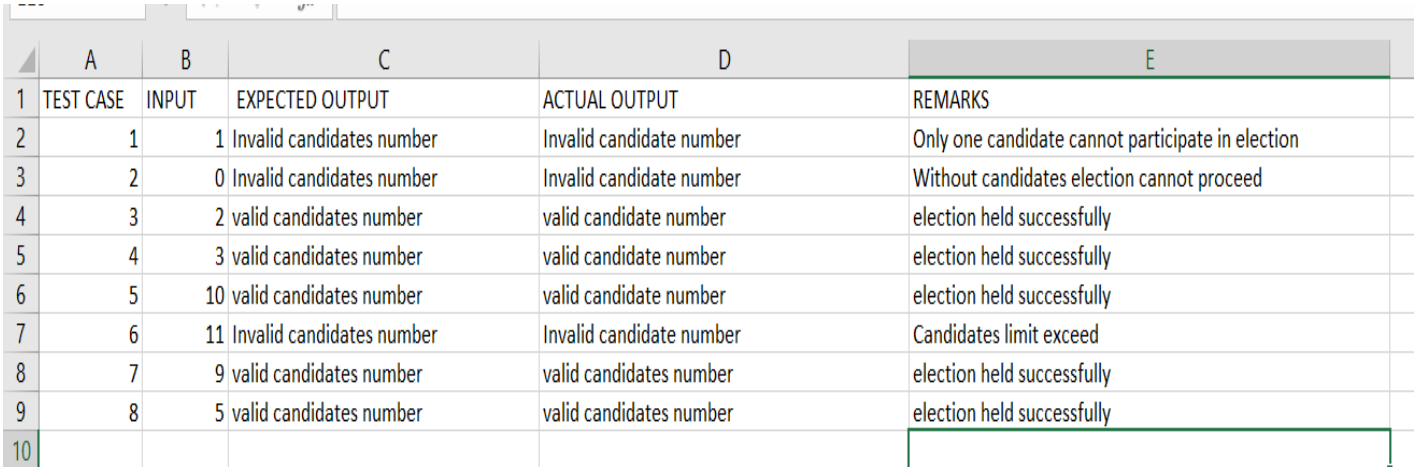
- Test Plan
- Test Cases
- Bug Reports
- Test Strategy

4 Test Cases :

Boundary Value analysis: Interface Capability

Number of candidates = 2 - 10 :

Invalid(min-1)	Valid (min, min+1, nominal, max-1, max)	Invalid (max+1)
0,1	2, 3, 5, 9, 10	11



	A	B	C	D	E
1	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMARKS
2	1	1	Invalid candidates number	Invalid candidate number	Only one candidate cannot participate in election
3	2	0	Invalid candidates number	Invalid candidate number	Without candidates election cannot proceed
4	3	2	valid candidates number	valid candidate number	election held successfully
5	4	3	valid candidates number	valid candidate number	election held successfully
6	5	10	valid candidates number	valid candidate number	election held successfully
7	6	11	Invalid candidates number	Invalid candidate number	Candidates limit exceed
8	7	9	valid candidates number	valid candidates number	election held successfully
9	8	5	valid candidates number	valid candidates number	election held successfully
10					

Equivalence Class Testing: Interface Capability

No of votes acceptable by a voter = 1

Invalid	valid	Invalid
0	1	2,3,4.....

	A	B	C	D	E
1	Test Case	Input	Expected output	Actual Output	REMARKS
2	1	1	Valid output	Valid output	only one vote cast by a voter
3	2	2	Invalid output	Invalid output	multiple votes cannot be cast by a user
4	3	0	No action perform	No action perform	nothing happen at all lead to no result
5	4	10	Invalid output	Invalid output	multiple votes cannot be cast by a user
6	5	-20	Invalid output	Invalid output	Invalid input type
7	6	50	Invalid output	Invalid output	multiple votes cannot be cast by a user

Equivalence Class testing: Input File Verification and Classifiaction

Accepted address	Not Accepted address
20 byte address	other than 20 byte address

Decision Table: Input File Verification, File Classification & Interface capability:

Conditions	Input-1	Input-2	Input-3	Input-4
address size	T	T	F	F
registered person (VOTER)	T	F	T	F
Result	Accepted	Not Accepted	Not Accepted	Not Accepted

Example:

Conditions	Input-1	Input-2	Input-3	Input-4	Input-5	Input-6
address size	20 byte	20 byte	20 byte	>20 byte	<20 byte	<20 byte
registered person (Voter)	same address	Election Commission address	Candidate address	same address	Election Commission address	Candidate address
Result	Accepted	Not Accepted	NOT Accepted	Not Accepted	NOT Accepted	Not Accepted

5 Resource & Environment Needs

5.1 Testing Tools

List of Tools like

- Selenium
- Mentis BT
- Automation BT

5.2 Test Environment

It mentions the minimum **hardware** requirements that will be used to test the

Application. Following **software's** are required in addition to client-specific software.

- Windows 10 and above preferred
- VSCode 2022 or above preferred
- Chrome, Mozilla or Edge Preferred over non-chromium based browsers

6 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test