MCI project Testing plan

Team number: 04
Project Title: User-friendly experimentation platform for blockchain

1. Introduction

1.1. Project Overview

NVAL is a framework for designing a new private blockchain network, but the current utility of NVAL framework is lower than expected. The goal of the project is to improve the usability and capability of NVAL framework and directly start the blockchain network design with a browser, no specific software is needed. The project will develop a graphical user interface to manage the deployment and evaluation of private blockchain networks and provide graphical editors for the designing of private blockchain networks.

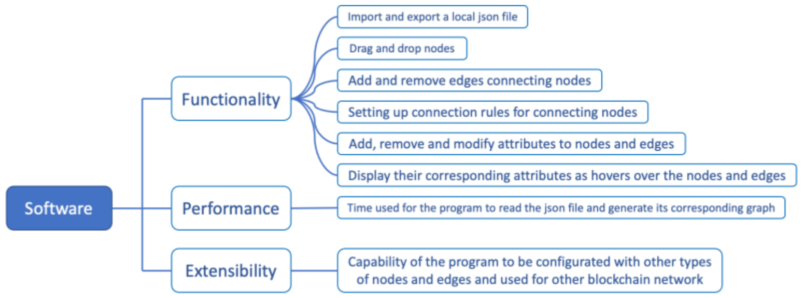
1.2. Scope of the test plan document

The test plan document describes the test strategy, test execution(including defect testing, unit testing and integration and release testing), testing plan schedule, identified risks and examples of test report.

2. Test strategy

2.1. Test scope: what will be tested and why

The objective of testing is to verify the developed software if it functions as expected and validate to see if it satisfies the requirement of the client as listed in the features of software which is illustrated in Figure 1. In short, there are 3 levels of features including functionality, performance and extensibility, which requires three types of testing that are specifically targeting each level, i.e. unit test, validation test and defect test. Unit tests will test for individual functions, validation tests will test all the functionalities and performance of the developed software, while defect tests will test if an error is raised during the whole development process.


Software
Functionality
Import and export a local json file
Drag and drop nodes
Add and remove edges connecting nodes
Setting up connection rules for connecting nodes
Add, remove and modify attributes to nodes and edges
Display their corresponding attributes as hovers over the nodes and edges
Performance
Time used for the program to read the json file and generate its corresponding graph
Extensibility
Capability of the program to be configurated with other types of nodes and edges and used for other blockchain network

2.2. Testing report

Details of the testing plan are in the Appendix A.

2.3. Test assumptions

2.3.1. Mock Data

We assume that our mock functions or services will provide the necessary data for testing. For instance, we assume that a mock API will return correct dummy data when provided with correct calls.

2.3.2. Environment

We assume that certain environment variables or configurations are set.

2.3.3. Data Consistency
The data representing the nodes and their connections remains consistent before and after operations.

3. Test execution

3.1. Defect testing (what will be tested and how)

In our projects, defect testing is designed to find and identify defects and problems in the project. It involves executing test cases and then detecting possible defects in the project by comparing the differences between the expected and actual results. The team will perform defect testing in three areas, nodes, attributes and generating Json files. Defect testing of nodes will involve node generation and deletion, connection rules and connection logic of nodes. For example, checking whether the program has followed the node connection rules correctly by trying to connect different nodes at random. In terms of attributes, the team tests the binding of attributes to nodes, the limit of the number of attributes added. In this aspect of generating Json files, the format of the generated Json files will also be involved in defect testing. Defect testing helps to help the team identify problems that were overlooked during the design and coding process, while enhancing the reliability and stability of the project.

3.2. Unit testing (what will be tested and how)

Unit testing includes the following functions:

- Add / drag and drop / remove nodes
- Add / remove edges connecting nodes
- Add / remove / modify / save attributes to nodes and edges
- Display attributes of a node or an edge when the mouse hovers over the node or edge
- Disallow certain types of node connections according to the connection rules
- Load a local json file to generate a connection graph of nodes and edges
- Export a json file based on the graph

3.3. Integration and release testing (what will be tested and how)

The team’s integration testing will focus on the two main functions of importing Json files and generating Json files. The integration test of import will verify that the team's project can generate the corresponding node relationship graph from the externally imported Json file, moreover the attributes and interconnections of each node are in accordance with the JSON file.The integration test for generating JSON files is then divided into two aspects, generation and import. The first point is that the Json file generated from the node relationship diagram should meet the basic JSON file format. The second point is that the generated Json file can be imported into the Json file integration test to ensure the integrity of the file.
In week 11 the team will do the final release testing of the entire project. This is to verify that the project meets the requirements and has value for delivery. The team will use black box testing to ensure the integrity, stability and compatibility of the project. Because the project

is not complex, black box testing is used to verify that the project works correctly according to the specifications, to detect functional defects and logical errors, and to ensure that the project is consistent with the requirements.

4. Testing plan schedule

The team will do unit tests to check the completeness of the features after they are completed, and the team will meet with the supervisor every Thursday and invite the supervisor to help test the features that have been implemented and those that are still under development. Since some of the features have not been developed yet, the team plans to implement the remaining features and start defect testing and integration testing in week 10, and release testing in week 11 when it will be delivered.

Testing Phase	Start time	End time
Unit test	March 9 (every Thursday)	May 18
Detect test and integration testing	May 19	May 25
Release test	May 26	Jun 1

5. Risks

Risk Description	Impact	Mitigation Plan
External imported Json files cannot generate node diagrams	Critical	Fix immediately
Unable to generate the Json file corresponding to the node diagram	Critical	Fix immediately
Problem with the format of the generated Json file	Critical	Fix immediately
Excessive coupling	Medium	Fix when have time

Appendix A

Test description	Input	Expected output	Current output	Responsible	Reviewer
Add node	Drag one type of node into the panel	One node is added into the panel	Same as expected	Yiwei Shen	Ziheng Jing
Connect nodes	Drag one node to another and make one connection	Two nodes are connected with each other	Same as expected	Hei Nga Chan	Yiwei Shen
Edit node attributes and save	Change node attributes on the content component	Node attribute is changed	Same as expected	Yizhi Xing	Hei Nga Chan
Edit egde attributions and save	Change edge attributes on the content component	Edge attribute is changed	Same as expected	Ziheng Jing	Yizhi Xing
Load JSON file and generate corresponding graph	Start this react server	Server is started and generate the correct graph	Same as expected	Yiwei Shen	Ziheng Jing
Generate JSON file according to current graph	Click the generate JSON button	One JSON file that include graph content is generated	Same as expected	Yizhi Xing	Ziheng Jing
Check if current graph apply to the connection rules	Node attribute is changed or connected to another node	If the connection is not allowed, the connection will not be made.	Same as expected	Hei Nga Chan	Yiwei Shen