AUT
UNIVERSITY
NEW ZEALAND

**COMP704** Research and Development Project

**VN01** **3D acupuncture healthcare data management and treatment system**

# Technical Testing Results Report

**Supervisor:**                    Dr Nhan Le Thi

**Team Members:**     21142643     Chuong Pham Dinh
                      21142377     Nhan Nguyen Cao
                      21142355      Tan Le Tran Ba
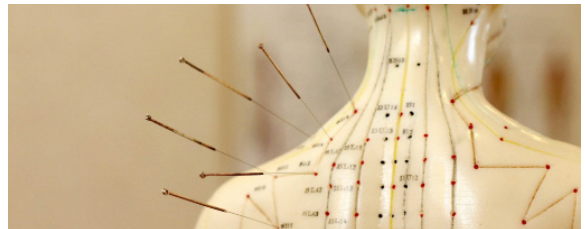                      21142358   Trang Ho Ngoc Thao



**Version:**                           1.0

**Date:**                    28th April 2023

# TABLE OF CONTENTS

# DOCUMENT VERSION CONTROL

## 1. DOCUMENT INFORMATION

| | |
|---|---|
| Document code | **TechTR** |
| Document title | **Technical Testing Results Report** |
| Version | **1.0** |
| Authors | **Nhan Nguyen Cao** |
| Distributed by | **Project VN01 team** |
| File name | **TechTR_Technical Testing Results Report_1.0.pdf** |
| Release definition | **Only released as a finished document** |

## 2. DOCUMENT SIGN-OFF

| ID | Member | Role | Signature | Timestamp |
|---|---|---|---|---|
| 21142355 | Tan Le Tran Ba | Project Manager |  | 30 Apr 2023 23:55 |

## 3. DOCUMENT VERSIONS

| Version | Timestamp | Description | Responsible members |
|---|---|---|---|
| **1.0** | 28 Apr 2023 22:30 | Summarize the results of all Technical Testing results involved during the Testing phase of the project. | **Nhan Nguyen Cao** *(21142377)* |

# I. FRONT-END TECHNICAL TESTING RESULTS

## I.1. UNIT TESTING + INTEGRATION TESTING

For Unit testing and Integration testing, a list of files to be ignored in covering test was defined based on some following criteria:

- The files involving the use of Three.js library for rendering out the 3D model. This was due to the mechanism the library used to render the model (render by script in third-party environment and just delivered back the 3D environment as a <canvas>, so no details inside the canvas can be reached using HTML code).

- The files involving the use of Swal Alert modal for renderring out the styled modals in different flows of the project. This was due to some un-resolvable bugs in mocking the libraries our team met during the implementation of the tests.

- The files involving the use of Firebase Authentication mechanism. This This was due to some difficulties in mocking the complex authentication mechanism done by Firebase for handling the steps. We assumed the library provided by Google is well-tested, and do not perform further unit tests on this.

Our main approach in covering the unit tests and integration tests were just to start from the components with least dependencies first, and cover all basic rendering and functions of that components. The test coverage was maintained throughout the implementation of unit tests to identify which other components, layouts or pages should be picked up for covering testing.

For the covering of integration tests, we mocked the API calls to cover different types of HTTP requests available for each endpoints. For all endpoints, we examined the happy cases (with results returned and status of 200) and failed cases (including both cases with empty results or failed request cases). We used the library "axios-mock-adapter" (https://www.npmjs.com/package/axios-mock-adapter) to mock the API server for each endpoint.

Some libraries were also mocked during the implementation of the test cases to get the expected behaviors, include:

- react-router-dom: Used for navigation between pages within the website.
- react-i18next: Used for the handling of language preference for the website.
- react-chartjs-2: Used for rendering the charts and graphs in Personal Learning Progress dashboard.

A total of 79 test suites, with 395 test cases are included within the code base from Front-end side. As the CI/CD flow requires all tests to be passed, we maintain 100% passing test cases through the implementation of the project from Front-end side. After ignoring the above files, a total of 64 code files were included in the counting of unit tests and integration tests coverage for the Front-end side of the project. Some statistics of the unit tests and integration tests include:

*Table 1 - Front-end Unit testing + Integration testing results*

| Criteria | Details |
|---|---|
| **Test suites** | 79 |
| **Test cases** | 395 |
| **Passed** | 395 / 395 |
| **Failed** | 0 |
| **Statements coverage** | 98.41% |
| **Branch coverage** | 96.87% |
| **Functions coverage** | 97.89% |
| **Lines coverage** | 98.39% |

Some specific functions or hooks were difficult to mock and test, due to the synchronization execution mechanism, and was ignored from covering unit tests.

## I.2. AUTOMATION TESTING

At first, Automation Testing was planned to cover all components or pages that have not yet been covered by unit tests and integration tests. However, we did have some difficulties executing the API calls when running the tests in Headless Chromium browsers. Following that, our team decided:

- Only use Automation Testing to cover the components that do not require API calls to function, and were not covered by unit tests or integration tests.
- For the remaining untested files, we would hand over the responsibility for Quality Assurance to Manual testing process.

The method defined for performing Automation Testing test cases were to run a simulated browser, entering the site (required running on localhost, so not added to CI/CD flow), wait for the basic loading (especially the 3D model loading) and perform some automation interactions. The results of the test cases are verified by two steps:

- Ensuring the test steps are able to be functioned without any error received.
- Capture a snapshot of the page at the end of the step, and verify the test cases at the end of the tests.

A total of 28 test cases covering 3 test suites (including one for Authentication flows, one for basic Manual Interaction Control on home page with the model, and one for basic examination of the expected UI for case the Advanced Search feature cannot call the API to get data) were written for handling of some other flows in Front-end side of the project. Test cases statistics are as follows:

*Table 2 - Front-end Automation testing results*

| Criteria | Details |
|---|---|
| **Test suites** | 3 |
| **Test cases** | 28 |

| Passed | 25 / 28 |
|:---:|:---:|
| Failed | 3 / 28 |

The failed cases were due to some change in mechanism of displaying Oauth signing in modal of Google and Firebase Authentication. The test cases were passed during the first runs, but failed a few days later. We decided to hand over these cases for manual testing.

# II. BACK-END TECHNICAL TESTING RESULTS

## II.1. UNIT TESTING

Unit testing from the Back-end side was a little bit more complex to do compared to the Front-end side, as there were many middlewares available for each successful calls to the Back-end server of the system. We agreed to use the following method for performing unit testing in the Back-end side:

- Focus the test cases only on the controller and service files, which are the main files to handle the execution and processing of endpoints for each modules.
- Ignore the Authorization module from code coverage calculation due to some difficulties met while mocking the guardian middleware (provided by Passport.js library). This would be handled later on in manual testing.
- Use the library "mongodb-memory-server" to mock the MongoDB database and examine the expected behavior for communication with the database (https://www.npmjs.com/package/mongodb-memory-server).

To cover the basic endpoints and examine whether it works correctly, Back-end side of the project covered 5 test suites, with 25 unit tests in total. Similar to the Front-end side, as the CI/CD flow requires all tests to be passed, we maintain 100% passing test cases through the implementation of the project from Front-end side. Some statistics of the unit tests and integration tests include:

*Table 3 – Back-end Unit testing results*

| Criteria | Details |
|:---:|:---:|
| Test suites | 5 |
| Test cases | 25 |
| Passed | 25 / 25 |
| Failed | 0 |
| Statements coverage | 98.88% |
| Branch coverage | 100% |
| Functions coverage | 93.54% |
| Lines coverage | 98.71% |

For some module, the endpoints for Deletion were available, which were due to the expected scope of Data Management feature at first. However, as we suggested later

on to not allowing inserting and deletion of acupuncture points and meridians items within the system, those endpoints were left unused, but not removed since it could be required for future expansion. Following that, since the Front-end side of the project does not use those deletion endpoints for the present, unit tests were ignored for those flows.