



COMP704 Research and Development Project

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

# Technical Specification

**Supervisor:**

Dr Nhan Le Thi

**Team Members:**

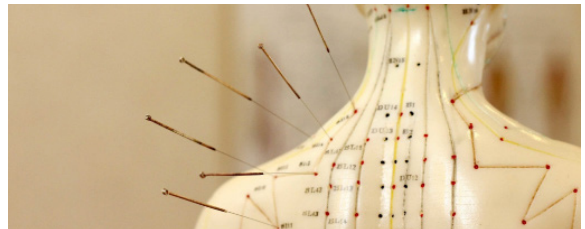
21142643	Chuong Pham Dinh
21142377	Nhan Nguyen Cao
21142355	Tan Le Tran Ba
21142358	Trang Ho Ngoc Thao

**Version:**

1.0

**Date:**

20<sup>th</sup> April 2023



# TABLE OF CONTENTS

<b>DOCUMENT VERSION CONTROL .....</b>	<b>3</b>
<b>1. DOCUMENT INFORMATION .....</b>	<b>3</b>
<b>2. DOCUMENT SIGN-OFF .....</b>	<b>3</b>
<b>3. DOCUMENT VERSIONS .....</b>	<b>3</b>
<b>I. EXECUTIVE SUMMARY .....</b>	<b>4</b>
<b>II. CONFIGURATION MANAGEMENT.....</b>	<b>5</b>
<b>II.1. FOLDER STRUCTURE.....</b>	<b>5</b>
II.1.1. FRONT-END CODE BASE .....	5
II.1.1. BACK-END CODE BASE .....	6
<b>II.2. CODE BASE MANAGEMENT .....</b>	<b>7</b>
II.2.1. FRONT-END .....	7
II.2.2. BACK-END.....	7
<b>III. VERSION CONTROL MANAGEMENT .....</b>	<b>8</b>
<b>III.1. GIT FLOW .....</b>	<b>8</b>
<b>III.2. GIT COMMIT CONVENTION .....</b>	<b>9</b>
<b>IV. ENVIRONMENT SET UP MANUAL.....</b>	<b>10</b>
<b>IV.1. FRONT-END CODE BASE .....</b>	<b>10</b>
<b>IV.2. BACK-END CODE BASE .....</b>	<b>10</b>
<b>IV.3. CI/CD FLOW.....</b>	<b>11</b>
<b>IV.4. PROCEDURE FOR BUILDING A NEW FEATURE.....</b>	<b>11</b>
<b>V. SECRET MANAGER .....</b>	<b>12</b>
<b>V.1. FIREBASE WORKSPACE .....</b>	<b>12</b>
<b>V.2. MONGODB INSTANCE.....</b>	<b>12</b>
<b>V.3. CIRCLECI .....</b>	<b>12</b>

<b>V.4. CODE BASE ENVIRONMENT VARIABLES.....</b>	<b>12</b>
V.4.1. FRONT-END .....	12
V.4.2. BACK-END .....	13

## DOCUMENT VERSION CONTROL

### 1. DOCUMENT INFORMATION

Document code     **TS**

Document title     **Technical Specification**

Version             **1.0**


Authors             **Nhan Nguyen Cao, Trang Ho Ngoc Thao**

Distributed by       **Project VN01 team**

File name           **TS\_Technical Specification\_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		21 Apr 2023 09:45

### 3. DOCUMENT VERSIONS

Version	Timestamp	Description	Responsible members
<b>1.0</b>	20 Apr 2023 00:55	First version of the basic Technical Specification for the project, in order to provide reader basic understanding about the Technical perspective of the project and some basic steps required for hand-overing the project.	<b>Nhan Nguyen Cao</b> (21142377) <b>Trang Ho Ngoc Thao</b> (21142358)

## **I. EXECUTIVE SUMMARY**

This document contains a basic summary of information on the technical requirements for the project's code base, including details about the structure for managing the project's code base and the management of the project's configuration. The document also includes details on the guidelines and practices that the team established and adopted to manage the project's technical development flows.

Additionally, the guide would include some fundamental instructions for setting up the environments and having everything ready to start working on the code base, whether for expanding the project's functionality or introducing new features.

## II. CONFIGURATION MANAGEMENT

### II.1. FOLDER STRUCTURE

#### II.1.1. FRONT-END CODE BASE

The folder structure tree for Front-end code repository is as follows:

```

/
|-- .circleci/: Store the configuration file for CI/CD flow in CircleCI
|   |-- config.yml: Configuration file for CI/CD flow in CircleCI
|-- .firebase/: Auto-generated directory for storing Firebase deployment cache
|-- node_modules/: Local only, store the libraries files for the application
|-- public/: Store the public assets for running the application
|-- src/: Main working directory, store the source code for the application
|   |-- @types/: Store the global file types for different source files of the code base
|   |-- api/: Store the configuration and tests, mocks for all API calls
|   |-- assets/: Store the assets to be used in the source files, including media (videos,
|       |         images, sounds, etc.), fonts, etc.
|   |-- components/: Store the source files for every single component
|   |-- configs/: Store the source files for defining the compulsory configurations for
|       |         the application
|   |-- e2e/: Store the end-to-end test specifications for the application
|   |-- helpers/: Store some files exporting helper functions for the entire source code
|       |         files
|   |-- layouts/: Store the layout wrapper for different pages of the application
|   |-- pages/: Store the source files for every single page of the application
|   |-- redux/: Store all state management source code files and mocks, tests
|   |-- routers/: Store source files for handling the navigation within the application
|   |-- i18n.ts: Store the translation configuration (Vietnamese and English) for the
|       |         entire application
|   |-- App.[tsx,test.tsx,scss] + index.[tsx,test.tsx,.scss]: Default files for running
|       |         React application
|   |-- responsive.scss: Store the configuration stylings for responsive viewports
|-- .env: Local only, store the environment variables
|-- .env.example: Store the required variable schema
|-- .firebaseconfig: Store the configuration for Firebase deployment
|-- firebase.json: Store the configuration for Firebase command
|-- package.json: Store the configuration for commands and dependencies of the
|       |         application.
|-- README.md: Default README file for the code base
|-- SkippedLinesOfTest.md: Define the lines of code files that should be ignored by
|       |         running unit tests
|-- tailwind.config.cjs: Store the configuration for integrating TailwindCSS
|-- tsconfig.json: Store the configuration for TypeScript project

```

### II.1.1. BACK-END CODE BASE

The folder structure tree for Back-end code repository is as follows:

```

/
|-- .circleci/: Store the configuration file for CI/CD flow in CircleCI
|   |-- config.yml: Configuration file for CI/CD flow in CircleCI
|-- node_modules/: Local only, store the libraries files for the application
|-- src/: Main working directory, store the source code for the application
|   |-- <module>/: Each directory within the src/ directory corresponds to one
|   |   |           module of the project
|   |   |-- dtos/: Store the configuration for Data Transfer Object of the current
|   |   |           module
|   |   |-- entities/: Store the schema for entity that the current modules is managing
|   |   |-- services/: Store the configuration for the service(s) that are handling by
|   |   |           the current module
|   |   |-- <module>.controller.ts: Defining all the controllers for handling
|   |   |           endpoints of the current module
|   |   |-- <module>.module.ts: Defining the module for Mongoose to communicate
|   |   |           with the MongoDB database
|   |-- app.controller.spec.ts: App-level file for defining the unit tests
|   |-- app.controller.ts: App-level file for defining the controller of root endpoints
|   |-- app.module.ts: App-level file for defining the general module for the whole app
|   |-- app.service.ts: App-level file for defining the service
|   |-- main.ts: Bootstrap every module configuration and get it available to run within
|   |           the application
|-- test/: Main directory for storing the configuration of E2E tests (if available)
|-- .env: Local only, store the environment variables
|-- .env.example: Store the required variable schema
|-- nest-cli.json: Store the configuration for using Nest CLI
|-- firebase.json: Store the configuration for Firebase command
|-- package.json: Store the configuration for commands and dependencies of the
|   application.
|-- README.md: Default README file for the code base
|-- tsconfig.json: Store the configuration for TypeScript project

```

## II.2. CODE BASE MANAGEMENT

### II.2.1. FRONT-END

Since the Front-end code base is built using React.js library, we follow the component-first design for building new feature or upgrading parts of the current code.

There are three main levels of components included in the code base, organized in the directories within the src/ folder, including:

- **Layout:** can be understood as the outermost container for a whole page. Currently, the application only contain one type of layout, the FullLayout, which span all width and height fitting the content of the page, not shorter than the height of the screen.
- **Page:** can be understood as the screens of the application that can be accessed through the sub-path in URL.
- **Component:** can be understood as each small sub-parts of any screen, that has its own logic and flow for performing its feature. Please also take in mind the reusability of the components, and put those that can be generalized and reused in the common/ sub-directory.

### II.2.2. BACK-END

Since the Back-end code base is built using NestJS framework, we follow the module-first design for building any new feature. Each module can be understood as one collection stored in the database, and each module directory should handle all the endpoints for communicating with that collection in the database. Consider those things while working with any module within the Back-end code base of the project:

- **dtos** and **module:** make sure they are defined compatible with the structure of the documents inside each collections of the database.
- **controller:** should define all the endpoints in using, including the HTTP method for reaching the endpoint and whether there are any decorator wrapped outside for adding layers before reaching the core of the endpoints.
- **service:** should define the basic CRUD operation, but prioritize those already provided by NestJS.



### III. VERSION CONTROL MANAGEMENT

#### III.1. GIT FLOW

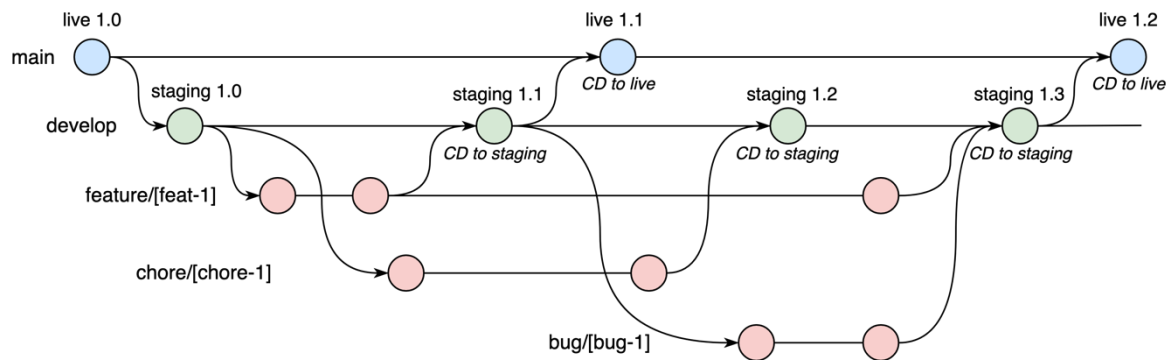


Figure 1 - Git flow for the project

The git flow for all repositories of the project include the following branches:

- **main**: The code stored in this branch is the one available in live server.
- **develop**: The code stored in this branch is the one available in staging server.
- **feature, chore or bug**: These branches are created to handle each small part of the code, normally corresponding to one ticket.

The git flow for the project code base can be described as follow:

- For every new ticket started, create a new branch from develop following the naming convention: **[feature/chore/bug]/(<ticket-number>)-<name-of-the-ticket>**. You have this branch to freely develop your code.
- When the code is ready and the tests are passed, create a new Pull Request to develop, following the convention: **<ticket-number> - <name-of-the-ticket>**. When the Pull Request is approved, the code is merged into develop and CI/CD will handle the flow for deploying the feature into Staging Server for testing.
- When there are official releases to live server, the group leader create a new Pull Request from develop to main, and resolve any conflict. When the Pull Request is ready (no need to wait for approval), the code is merged into main and CI/CD will handle the flow for deploying the feature into Live Server.

## III.2. GIT COMMIT CONVENTION

Corresponding to three types of sub-branches for resolving tickets, there are three types of commit defined for the code base:

- **[f] (feature) commit:** for adding new things, new flows or new layouts to the current version of the application
- **[c] (chore) commit:** for some refactoring or adding small changes to the flows of the application
- **[b] (bug) commit:** for resolving one feedback or identified bug in the current flow of running of the application

All commit should be named following the convention: **[type] – [detail]** (for example: *[f] – Add dropdown for filtering meridians*).

Besides, since there is CI/CD flow set up for the project, all commit should passed unit tests and maintaining the test coverage for the complete test base. It is acceptable by the team to ignore some files that are difficult to test or not supported (especially those related to Three.js integration, since the library render the models using third-party script that is not accessible by the current code), but should not abuse that.

For any Pull Request created, all unit tests must be passed (CI flow will pause if there is any failed test). It is acceptable for commits to have failed tests and fail the CI flow, but for Pull Request, passing all unit tests are required before approved for merging.

## IV. ENVIRONMENT SET UP MANUAL

Prerequisite: It is required to have Node.js installed on your local machine in order to be able to run both the Front-end and Back-end code bases of the project. Feel free to install the latest version of Node.js from the home page of the library: <https://nodejs.org/en/download>.

### IV.1. FRONT-END CODE BASE

The following are some of the steps used for setting up the environment for running the code base from Front-end side:

- Clone the code repository: `git clone https://github.com/rnd-vn01/rnd-vn01-fe.git`
- Install all required libraries: After the code base has been cloned to your local, run the command `npm install` to install all required libraries (Note: It is recommend to use npm over yarn for this project).
- Set up the environment variables: Copy the file `.env.example` to `.env`, and fill in the values for all environment variables. Those values can be achieved from the Secret Manager section below.
- Run tailwindcss: Open one new terminal tab and run the command `npm run tailwind`. Keep this tab opened during your coding session.
- You are ready to go, run `npm run start` to run the application at default port (3000).

Some other commands for specific flows:

- To run all the tests with coverage report provided, use the command: `npm run test:coverage` (Note: this command would only run the unit tests, the end-to-end tests are triggered using a different command).
- To run all end-to-end tests, use the command: `npm run test:e2e`

### IV.2. BACK-END CODE BASE

The following are some of the steps used for setting up the environment for running the code base from Front-end side:

- Clone the code repository: `git clone https://github.com/rnd-vn01/rnd-vn01-be.git`
- Install all required libraries: After the code base has been cloned to your local, run the command `npm install` to install all required libraries (Note: It is recommend to use npm over yarn for this project).
- Set up the environment variables: Copy the file `.env.example` to `.env`, and fill in the values for all environment variables. Those values can be achieved from the Secret Manager section below.

- You are ready to go, run the command `npm run start:dev` or `npm run start:debug` to run your server in watch mode or watch + debug mode, at default port of 3001.

Some other commands for specific flows:

- To run all the tests with coverage report provided, use the command: `npm run test:cov` (Note: this command would only run the unit tests, the end-to-end tests are triggered using a different command).
- To run all end-to-end tests, use the command: `npm run test:e2e`
- The Back-end code base also included one endpoint for accessing the Swagger page of the server. Visit `/api` to interact with the Swagger page.

### IV.3. CI/CD FLOW

The code bases for both Front-end and Back-end side have been set up with flow for Continuous Integration and Continuous Deployment. That is:

- All commits would trigger the check for running all the unit tests. The CI flow is considered passed if all the unit tests are passed successfully.
- All Pull Requests after merged would trigger to flow:
  - Rerun the unit tests to make sure they are good to go
  - Deploy into server: For Front-end side, main branch would be deployed to live server, develop branch would be deployed to staging server, both hosted using Firebase Hosting. For Back-end side, currently, both main and develop branches would be deployed directly to live server, hosted using Heroku.

### IV.4. PROCEDURE FOR BUILDING A NEW FEATURE

- Analyze the feature: Maintain the approach for both sides in designing (Component-first for Front-end and Module-first for Back-end), then define possible components or modules to be included in the project.
- Initialize the code base for new feature:
  - Front-end: Check for existing layouts, pages and components to see whether there are components that are reusable. Then create folders for storing new components, pages. For other configurations, hooks or helper functions, use the existing directories. Each folder should include the `.jsx` file, `.scss` file (if needed) and `.test.jsx` file for storing the unit tests.
  - Back-end: Check to see whether the new feature is belonged to the scope of a new module or an existing module. In case of adding new module to the server, use the commands of NestJS for adding.
- After finish developing the code and execution for the new feature, remember to add unit tests covering at least 80% of the added code and maintaining the total code coverage of more than 80% before creating PR.

## V. SECRET MANAGER

### V.1. FIREBASE WORKSPACE

There are two different Firebase instances for this project, including:

- **Staging server** (Firebase Hosting): Can be signed in using the following Google account:
  - Email: [cycle13.rnd.vn01@gmail.com](mailto:cycle13.rnd.vn01@gmail.com)
  - Password: Acupuncture@VN01
- **Live server** (Firebase Hosting + Firebase Cloud Storage + Firebase Authentication): Can be signed in using the following Google account:
  - Email: [acupuncture.3d.vn01@gmail.com](mailto:acupuncture.3d.vn01@gmail.com)
  - Password: Acupuncture@VN01

### V.2. MONGODB INSTANCE

Use the following connection string to connect: `mongodb+srv://rnd-cycle13-vn01:0HgZUYdcgK4WVORu@cluster0.bqu0t0p.mongodb.net/rnd-cycle13-vn01`

### V.3. CIRCLECI

When joining the project, new members should ask the code owner to be added to the organization of the current code base. Once then, the new member can sign into CircleCI workspace of the project using Github account.

### V.4. CODE BASE ENVIRONMENT VARIABLES

#### V.4.1. FRONT-END

- `FIREBASE_TOKEN=1//0e0x4H6IFIVgLCgYIARAAGA4SNwF-L9IrRuXv-niQDlGjTqT88_MOZvZ9Uzh5EBgSBx-fu1zZwICMa4auWB9QSlyl-nJMfIG8SX0`
- `GENERATE_SOURCEMAP=false`
- `REACT_APP_API_ENDPOINT=https://rnd-vn01-be.herokuapp.com/`
- `PRODUCTION_FIREBASE_TOKEN=1//0em9va7jT-TF1CgYIARAAGA4SNwF-L9Ir9KJVNNTSpq1gWQ-5UwhIzASRCJVPE4221Y4JwJZn0oWtQDOKdkKW2vY0AvO-qYXBSug`
- `REACT_APP_FIREBASE_CONFIG='{ "apiKey": "AIzaSyBhh-rxBo5DHXynUM5H45pTtGxni-um2QA", "authDomain": "acupuncture-3d-vn01.firebaseio.com", "projectId": "acupuncture-3d-vn01", "storageBucket": "acupuncture-3d-vn01.appspot.com", "messagingSenderId": "704168149305", "appId": "1:704168149305:web:881bbf7518e706ebdd68e8" }'`

### V.4.2. BACK-END

- MONGO\_DATABASE\_USER=rnd-cycle13-vn01
- MONGO\_DATABASE\_PASSWORD=0HgZUYdcgK4WVORu
- MONGO\_DATABASE\_CLUSTER=cluster0.bqu0t0p.mongodb.net
- MONGO\_DATABASE\_NAME=rnd-cycle13-vn01
- PORT=3001
- MONGO\_DB=mongodb+srv://rnd-cycle13-vn01:0HgZUYdcgK4WVORu@cluster0.bqu0t0p.mongodb.net/rnd-cycle13-vn01?retryWrites=true&w=majority
- JWT\_SECRET=p2gsk4k8u10xvwpwn853
- JWT\_EXPIRED=30m
- VERCEL\_PROJECT\_NAME=rnd-vn01-be-staging
- VERCEL\_TOKEN=eEi8boGf4WrrsVSgZDn5PGT2
- VERCEL\_SCOPE=cycle13rndvn01-gmailcom