Development Rules

- **Keep It Sample And Stupid** - Easy to read and understand, never over-design the code structure.
- **Convention Over Configuration** - Following the official suggestion, I don't think we are smarter than Elon Musk.
- **Don't Repeat Yourself** - Never repeat your codes, reuse them.

Auxiliary Verb Level

- ➔ Must
  - ◆ You must follow the rules with no option, just do it.
- ➔ Must Not
  - ◆ You must not do that with no option, never do that.
- ➔ Never
  - ◆ Don't do that or we will die together.
- ➔ Should
  - ◆ You should do that most of the time, and we suggest you do that.
- ➔ Should Not
  - ◆ You should not do that most of the time, and we suggest you not to do that.
- ➔ May
  - ◆ You may do that as u like.

Ref: RFC 2119

Development Environment

1. **VScode** (Must) - Don't tell me use webstorm or vim or balabala, only VScode since fast loading speed, fast search speed, and nice development support (plugin, theme etc .)
2. **Google Chrome** (Must), **Safari** (Must) - I think I don't need to explain right?
3. **Vagrant** (Must) - Docker helps us a lot, but in most cases, we still need a developed environment that is 90% similar to the **Linux** system.
4. **VScode Plugin** (Must) - Just download all of them.
5.

Color Highlight	Highlight Display Web Color

Document This	Auto Markdown
ESLint	Frontend Code Style Checker
GitLens	Git Manage Tools
Live Server	Static Doc Service
Material Icon Theme	Good Looking Icon Theme
Material Theme	Good Looking Theme
SonarLint	Code Quality Checker
TODO Highlight	Highlight Your TO-DO List
StandardJS - JavaScript Standard Style	Backend Code Style Checker

---

Markdown

1. For all markdown, follow the rules of 中文文案排版指北.
2. All markdown **must** include
    - **Project Overview** - What is this project about, such as simple project description, simple feature description, etc.
    - **Production Environment** - System Requirements, Production Environment Required, etc.
    - **Development Environment Setup Guide** - How to set up the development environment including all required sources, step by step guidelines **must** be included.
    - **Server Structures** - Show your server structure graph (If it is possible, show your caching, RAM using, Third Party connection too)
    - **Code Deployment** - How to deploy your code into a testing environment and production environment, show your branch architecture too.
    - **PlugIn List** - List all of your plugins in package.json including **what is it**, **where is it** and **why using it**.
    - **API List** - For the frontend, list out all required API connections of each page, for the backend, list out all non-secure level API sources including **Postman Collection**. (Including Production and Testing Environment)

Tools

- Code Editor: VSCode
- Code Formatter: ESLint, StandardJS
- Code Quality Checker: SonarLint
- VM: Virtualbox
- MySql Query Tool: Sequel Pro
- Redis Mange Tool: RDM
- MongoDB Mange Tool: MongoDB
- Design Tool: Figma

Backend

- Backend code style must follow Standard JavaScript Style

Backend Router

- Keep the router simple, only router configuration is available in any router file.
- A maximum number of each router is 2. - e.g.「 /user/create 」
- Router must be single - e.g.「 /user/create 」
- All router files must name with pre file name - e.g.「 user.create.js」

Backend Data Model

- All **data model** must be placed into「/model」files
- All **data model** must extend from「Base Model」
- All **data model** must be single - e.g. 「models/User」
- All files must be single - e.g. 「models/User.js」
- All table name must be single with snake case - e.g. 「user, user_detail」
- **Database primary key** must be「id」.
- **Database reference key** must be「resource_id」. - e.g. 「user_id, post_id」
- **Database creation script** must be single. - e.g. 「2020_02_12_create_user_table.js 」
- **Database data insert script** must be single. - e.g. 「UserTableSeeder.js」

- Never using **SQL manage tool** or **SQL statements** directly to create tables, must use **database creation script** to create tables.
- Never manually insert data into the database, must use **database data insert script** to insert testing data.

Backend Controller

- All controller files must be single. - e.g.「UserController, UserController.js」
- Keep minimum and clear the code inside the controller file.
- **Should Not** write any comments for the name of the controller function.
- **Should** delete all useless / unuse controller routers.

```
├── app
│   ├── controller
│   │    ├── authController
│   │    │    └── authController.ts
│   │    ├── adminController
│   │    │    └── adminController.ts
│   │    └── userController
│   │         └── userController.ts
│   ├── dto
│   │    ├── authDto
│   │    │    └── authDto.ts
│   │    ├── adminDto
│   │    │    └── adminDto.ts
│   │    └── userDto
│   │         └── userDto.ts
│   ├── extend
│   │    └── helper.ts
│   ├── middleware
│   │    ├── error-handler.ts
│   │    ├── jwt-auth.ts
│   │    └── request-id.ts
│   ├── model
│   │    ├── authModel
│   │    │    └── authModel.ts
│   │    ├── adminModel
│   │    │    └── authModel.ts
│   │    └── userModel
│   │         └── userModel.ts
│   ├── service
│   │    ├── authService
│   │    │    └── authService.ts
│   │    ├── adminService
│   │    │    └── adminService.ts
│   │    └── userService
│   │         └── userService.ts
│   └── util
│        └── my-error.ts
├── config
```

```
|    ├── config.default.ts
|    ├── config.local.ts
|    ├── config.types.ts
|    ├── config.unittest.ts
|    └── plugin.ts
├── database
├── typings
├── app.ts
├── configuration.ts
└── interface.ts
```

Error Code

- 200: Server returns data successfully.
- 201: Create or update data successfully.
- 202: New request to enter MQ.
- 204: Delete data successful.
- 400: Request error, server did nothing.
- 401: Permission denied.
- 403: Premium accept but request denied.
- 404: No record and server did nothing.
- 406: Wrong request format.
- 410: Resource deleted.
- 422: Token validation fails during creation of new record.
- 500: Server error, please check the server status.
- 502: Network error.
- 503: Server not usable since under maintenance or gateway timeout.
- 504: Gateway timeout.

Frontend Folder Structure

```
├── config
├── mock
├── public
|    ├── favicon.png
|    └── index.html
├── src
|    ├── assets # Static Resource Such as Icon
|    ├── components # Basic Components
|    |    ├── globalComponent
|    |    |    ├── header
|    |    |    ├── footer
|    |    |    ├── dialog
|    |    |     ...
|    |    ├── homePageComponent
```

```
|     |     └── myAccountPageComponent
|     ├── e2e # Testing Use Cases
|     ├── layouts # Basic Layouts
|     ├── models # DVA Model
|     ├── pages # Frontend Page
|     |     ├── homePage
|     |     └── myAccountPage
|     ├── apis # API Request Handler
|     ├── utils # Tools
|     ├── locales # Localization
|     └── styles # Global css
├── tests
├── readme.md
└── package.json
```

UI Request Flow

1. Users interact with UI components.
2. Call **model** effect.
3. **Service** to call function.
4. Using Packed **request.js** to call API.
5. Receive data response from server and call **reducer** to update **state**.
6. Update **model**.

Git Branch Management

Master	Develop	Design	Hotfix
Master branch is the unique branch with master permission which is the production environment branch.	Develop branch is the development branch for engineer, new feature, function will under this branch. Both testing and local environment will under this branch.	Design branch is the development branch for designer, only non-functional related update (UI/UX) will be approved and merge to master immediately.	Hotfix branch is the development branch for engineer fix the problem which was discovery in the production environment, no permission required but we should avoid to use it