# Test Report of TAMURAT

In this project, we use `rpsec` for TDD and `cucubmer` for BDD, the overall coverage upto now is 91%.

## Test with Rspec

### Overview

In this project, we use Rsepc for TDD. Since there are many functions are implemented in the project, such as create account, upload file, send email and activate account via email, we use many other helpful packages to improve the quality of testing. Here are all the package we used:

\* rspec-rails : bone of TDD

\* simplecov : show the overall coverage of testing

\* byebug : debugging tool

\* fixture : mocking uploaded file

\* factory\_bot\_rails : mocking variables for testing

\* faker: generating email and personal name for convenience

\* shoulda\_matchers : simplifing the Model test

\* email\_sepc : email testing

Up to now, the overall coverage of rspec is about 86%, only few branches of functions are not covered yet . We use rspec here here mainly for unit test and cucumber for integration test.

### Guide On Usage

In part of Rspec, the test files are all in the folder named with `spec`. In the `spec` folder, there are 5 sub-folders, the `controllers` contains the test cases for controllers, the `model` folder contains the test cases for model, the `mailers` contains the test cases for mailer. Besides these, folder `factories` has the data file of `Factory\_Bot`, which provides the mocked data for testing. All the data we used in the rspec test could be found here. `fixtures` is the folder for another package called `fixture`, which has a very similar function with `Factory\_Bot` but we only take it for test file uploading, because it has a function could mock the uploaded file.

To run the test, just type `rspec` in the terminal. If someone want to run a specific test file, just type `rspec [path of file]`. Sometimes we need to empty the test database with the command `rails db:test:prepare` so that we could have a clean environment to run the test.

### Model Test

In model test, we focus on the relationships between each model and the definition of each model variables. `shoulda\_matchers` is widely used here because it has many simplified matchers.

The relationships between each model generally contains `belongs\_to`, `foreign\_key` and `dependent`. And definition of each model variables includes `validates`,`presence`,`length`,`format` and etc. There are many private functions definied in each model but we do not test them directly but merged into controller test.

we cover all the Model implemented in the project:

\* scenario : The whole task has different work assignments to access

\* scenario\_weight : the importance of each scenario

\* category : questions have different classes and we classify them into categories

\* scale : the criterian of answers for each category

\* category\_score : the quality of overall questions answered by one company in category

\* subcategory : category may be too vague and subcategories could help validators and companies

\* subcategory\_score: the quality of overall questions answered by one company in subcategoriy

\* question : questions for validation

\* answer : answers of each question in each subcategory

\* evidence : file of answer

\* user : user account with different roles

\* company : the main object of this project

\* privilege : the mid-model between user(admin) and scenario

### Controller Test

Controller test is the main work in rspec. Most controller have `before\_action` to check the permission of current\_user so we have to deploy the test for the same action but under different current\_user such as Admin, Validator, Decision Maker and Company Representative.

Generally, we use `expect(response)` to test the workflow of each action to make sure they render or redirect to the correct views. For some actions like create and update, we use `expect(assigns)` to check the details like variables and parameters. Factory\_Bot is very important here to mock different valid or invalid variables so that the test could go on.

### Helper and Mailer

Implement in Mailer is simple but important, email\_sepc is used to test the sender, recipient, subject or context of email and etc. The functions in Helper are not test directly but they are called frequently in the test of controller and them are covered in the end.

### Challenge

We use the latest version of Rspec and take a period of time to get used to it. The test of activation via email cost us a lot of time because we did not found the documents in the Internet. We have to know the mechanism of rpsec and controllers very well so that we could solve the problem.

&nbsp;

---

## Test with Cucumber/Capybara

### Overview

The BDD tests are mainly implemented by Cucumber and Capybara. Pakages support BDD test are listed below:

\* cucumber-rails: run automated tests written in plain language

\* capybara: simulate user interactions with app

\* webdrivers: apply simulations of user operation on a browser

\* selenium-webdriver: offer some methods to deal with Javascript

\* guard: run the tests automatically

In this project, BDD tests focus on features and scenarios of different roles: Admin, Decision Maker, Validator and Company Representative. Most of scenarios are included in `\*.feature` files in `/feature`. Some fake users are created `in seed.rb` for test.

To implement BDD test, prepare the test database first and then run cucumber:

```

$ rails db:test:prepare

$ cucumber

```

### Sign up/ Log in

\* User can `sign up` to create a new account

\* User receive an email after signing up, which informs the login id

\* Account is available after activation

\* User can `log in` with their login id and password

\* User can `log out`

Up to now, all the features of sign up and log in are tested and pass the tests. except email features.

### All users features

\* User can see `Home page`

\* User can see `Help page`

\* User can see `Profile`

\* User can change account inofrmation (name, password, gravatar) in `Settings`

All the users features are tested and pass.

### Admin features

\* Admin can see all users information in `Users` (gravatar, name, login id, email, role, activated)

\* Admin can manage users in `Users` (search, filter, delete, activate)

\* Admin can see all questions in `Questions`

\* Admin can manage categories, subcategories and questions in `Questions`(create, edit, delete)

\* Admin can import questions by uploading `.cvs` files in `Questions`

\* Admin can see all scales in 'Scales'

\* Admin can manage all scales in `Scales`(filter, create, edit, delete)

\* Admin can see all scenarios in `Scenarios`

\* Admin can see the detail information of a scenario by clicking name in `Scenarios`

\* Admin can manage scenarios in `Scenarios`(create, edit, delete, assign to Decision Makers)

\* Admin can see all companies and their dependency in a JS-tree in `Output`

\* Admin can see task achievement of each company by clicking its name in the JS-tree in `Output`

\* Admin can invite other people to be an Admin in `Invite`. The person who is invited will receive an email

Up to now, all the features of Admin are tested and pass, except scenarios relate to email, JS-tree or deleting.

### Decision maker features

\* Decision maker can see all questions in `Questions`

\* Decision maker cannot manage questions

\* Decision maker can see scenarios of his/her own in `Questions`

\* Decision maker can manage scenarios in `Questions`(create, edit)

\* Decision maker can see all companies and their dependency in a JS-tree in `Output`

\* Decision maker can see task achievement of each company by clicking its name in the JS-tree in `Output`

Up to now, all the features of Decision maker are tested and pass, except scenarios relate to JS-tree or deleting.

### Validator features

\* Validator can see all scales in `Scales`

\* Validator can only filter scales in `Scales`

\* Validator can see all companies in `Companies`

\* Validator can validate a company's questionnaire in `Companies`(question details, validate the answer, view evidence file)

\* Validator can finalize validation in `Companies`. After finalization, Company representative cannot change the answer of questions

Up to now, all the features of Validator are tested and pass, except finalization or scenarios relate to files.

### Company Representative features

\* Company Representative can see all scales in `Scales`

\* Company Representative can only filter scales in `Scales`

\* Company Representative can see questionnaire in `Questionnaire`

\* Company Representative can answer questions in `Questionnaire`

\* Company Representative can edit their answer, see the details of answers and upload evidences after answering a question in `Questionnaire`

\* Company Representative can upload files by clicking `Manage evidences` in `Questionnaire`

\* Company Representative can invite other people to be a Company Representative in `Invite`. The person who is invited will receive an email

Company Representative features have not been fully tested in BDD yet.

### Result

The overall coverage of cucumber is 61.62%.