CSCE606 Software Engineering Final Report

TAMU Risk Assessment Tool (TAMURAT)

Team: TreeNewBee

**Team Members:**

Yixu Chen

Feiyan Yu

Boquan Tao

Shibo Wang

Zheyong Sun

Lijiu Liang

**Introduction**

The purpose of the TAMURAT project is to build an evaluation tool for a large company to evaluate different scenarios in which different companies are chosen as contractors and subcontractors with different weights. This tool helps stakeholders make decisions by charting the Business, Security, and Financial scores of the evaluated company and the final scores of different scenarios.

The stakeholders of this project are British Telecom. According to the stakeholder requirements, the users of this tool is divided into four roles, which are administrator, decision maker, validator, and company representative. The administrator is responsible for managing all users and building new scenarios. It can delete or add a user and assign privilege of scenarios to this user. It can also set evaluation criteria for company representatives and add new questions. The company representative can sign up and ask some questions and also needs to upload some evidence, for example, PDF files. The validator can validate a company representative based on it answers and evidence it provides and finalize a decision so that the company representative cannot change its answers and evidence anymore. Finally, the decision maker can build new scenarios and assign different weights to different company representatives. It will choose the best scenario based on the final scores of all scenarios.

**User stories and Lo-fi UI**

|  |  |
| --- | --- |
| Admin Feature:Feature 1: Admin could approve subcontractor invitation and send log in id & default password through email (3 pts)  * As an admin * So that I could decide whether subcontractor could register or not * I want to approve subcontractor invitation and send user id and default password via email  Feature 2: Admin edit questions (decision maker cannot) (2 pts)  * As an admin * So that I can provide my desired questions * I want to edit questions.  Feature 3: Admin sets certain scenarios for the decision maker (3 pts)  * As an admin * So that I want different decision makers to evaluate differently * I want to restrict the scenarios decision makers can see |  |

|  |  |
| --- | --- |
| Decision Maker Feature:Feature 4: Decision Maker and Admin could set up scenario (2 pts)  * As a Decision Maker or Admin * So that I could set up scenario * I want to give each scenario a name and description  Feature 5: Decision Maker change companies’ weight at different scenarios (2 pts)  * As a Decision Maker * So that I could evaluate companies weight depends on scenarios * I want to give or change companies weight according to different scenarios  Feature 6: Decision Maker could change question weights (2 pts)  * As a Decision Maker * So that I could evaluate each question * I want to give or change questions weights |  |

|  |  |
| --- | --- |
| Validator Feature:Feature 7: Evaluate the company  * As a validator * So that I could evaluate subcontractors and contractor * I want to give level (drop down), comments (text box), the weight of a company (text box), the weight of a question (text box)  Feature 8: More than one validator  * As a decision maker or validator * So that I could see other validation of questions * I want to see who validates which question  Feature 9: Re-validates a company when it changes the answers  * As a validator * So that I can validate a company more accurately * I want to re-validate a company |  |

|  |  |
| --- | --- |
| Contractor & Subcontractor Feature:Feature 10: Company user could upload pdf (3 pts)  * As a company user * I want to upload several pdf documents as my company evidence * So that validator can see or download them  Feature 11: Company could be evaluated by answering questions (3 pts)  * As a company user * So that I could be admitted * I want to fill the questions (CSV file) to get evaluated  Feature 12: Contractors invite subcontractors via email (3 pts)  * As a contractor * So that I need to add a subcontractor * I want to have a separate page to invite subcontractor via email |  |

|  |  |
| --- | --- |
| Function FeatureFeature 13: Users could register an account (3 pts)  * As a contractor or decision-maker or validator * So that I could register an account * I want to fill my email, nickname (company name/validator name), password and role and wait for the login id sent by admin via email.  Feature 14: Users could contract admin for support (2 pts)  * As a user * So that I may meet problems and need help * I want to have a button to contract admin for support  Feature 15: Forget and reset password (3 pts)  * As a user or admin * So that I may forget or want to change password * I want to reset the password via email  Feature 16: Notice if password incorrect (2 pts)  * As a user * So that I want to get noticed when password incorrect * a flash will jump up when password incorrect to note user or admin |  |

**Role of each Iteration**

|  |  |
| --- | --- |
| Product owner | *Yixu Chen* |
| Scrum master | *Feiyan Yu (Iteration 0-2)* |
| *Boquan Tao (Iteration 3)* |
| *Shibo Wang (Iteration 4)* |

**Description of each Iteration**

|  |  |  |
| --- | --- | --- |
|  | Features | Points |
| Iteration 1 |  |  |
|  |  |
| Iteration 2 |  |  |
|  |  |
| Iteration 3 |  |  |
|  |  |
| Iteration 4 |  |  |
|  |  |
|  |  |

**Test Report**

In this project, we use rspec for TDD and cucumber for BDD, the overall coverage up to now is 91%. In the project, we use rspec here mainly for unit test and cucumber for the integration test.

## Test with Rspec

### Overview

In this project, we use rspec for TDD. Since there are many functions are implemented in the project, such as creating accounts, uploading files, sending emails and activating account via email, we use many other helpful packages to improve the quality of testing. Here are all the packages we used in rspec:

* 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 emails and personal name for convenience
* shoulda\_matchers: simplifying the Model test
* email\_sepc: email testingUp to now, the coverage of rspec is about 86%, only a few branches of functions are not covered.

### 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 contain the test cases for controllers, the model folder contains the test cases for models, the mailers contain the test cases for mailers. Besides these, folder factories have 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 wants 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 the 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 for testing. The relationships between each model generally contain belongs to, foreign key and dependent. Definition of each model instance variables includes validates, presence, length, format and etc. There are many private functions defined in each model but we do not test them directly but will call them in controllers test.  
we cover all the models implemented in the project:

* scenario: The whole task has different work assignments to access. We test its relationship and validation of presence and length of some variables.
* scenario weight: the importance of each scenario. We test its relationship and validation of variable.
* category: questions have different classes and we classify them into categories. We test its relationship and the validation of certain attributes, like presence and length for several inner variables.
* scale: the criterion of answers for each category. We test its relationship and validation of presence of some variables.
* category score: the quality of overall questions answered by one company in categories. as an auxiliary model, we test its relationship of belongs\_to and has\_many
* subcategory: category may be too vague and subcategories could help validators and companies. We test its relationship and validation of presence and length of some variables.
* subcategory\_score: the quality of overall questions answered by one company in subcategories and we just test its relationships with other models.
* question: questions for validation. We test its relationships with other models but its instance function import is tested in the controller.
* answer: answers to each question in each subcategory, we test its relationship of belongs\_to and has\_many
* evidence: file of answers. as an auxiliary model, we test its relationship of belongs\_to and has\_many
* user: user account with different roles. It is the biggest model here, relationships, validation and instance function are tested here.
* company: the main object of this project. We test its relationship and the validation of certain attributes like presence and uniqueness, besides these we also test its instance function.
* privilege: the mid-model between users(admin) and scenario. as an auxiliary model, we test its relationship of belongs\_to and has\_many.
* relationship: The auxiliary model between answer and evidence and we just test its relationship.

### Controller Test

Controller test is the main work in rspec. The most controllers have before\_action to check the permission of current\_user so we have to deploy the test for the same action with a different result under various current\_user such as Admin, Validator, Decision Maker and Company Representative. To build the environment of login users with various roles, we need to create an instance user first using Factory\_Bot and assign its id to session[:id] and then the system could recognize the current user.

Generally, we use expect(response) to test the workflow of each action to make sure they render or redirect to the correct views. This kind of test occupies most of the overall test cases, every action under various roles may have a different response and we try to cover them all. In the latest version, we need to use params to pass key-value pairs when launch a new HTTP request and those values are built by Factory\_Bot as well.

For some actions like creating and updating, 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

Test cases in Mailer are simple but important, email\_spec is used to test the sender, recipient, subject or context of email and etc. The functions in Helper are not tested directly but they are called frequently in the test cases of controllers and they 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 find the documents on the Internet. We have to know the mechanism of rspec and controllers very well so that we could solve the problem.

While we take time to get familiar with these tools but they provide us a trustworthy backing since we could find the hidden bugs immediately. In each iteration after a weekly meeting we changed or modified the structure of our project, if no support of testing, we may suffer a lot of time of debugging and got frustrated.