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 relationship among these four roles and the database diagram are shown below.

After the contractor's answering the questions and upload evidence, the validators log in and validate all the answers and evidence and finalize their decision so that the contractor can't change their answers and evidence.

Validated by the validator, a score will be calculated following the formulae of weighted sum for each subcategory and category of questions.

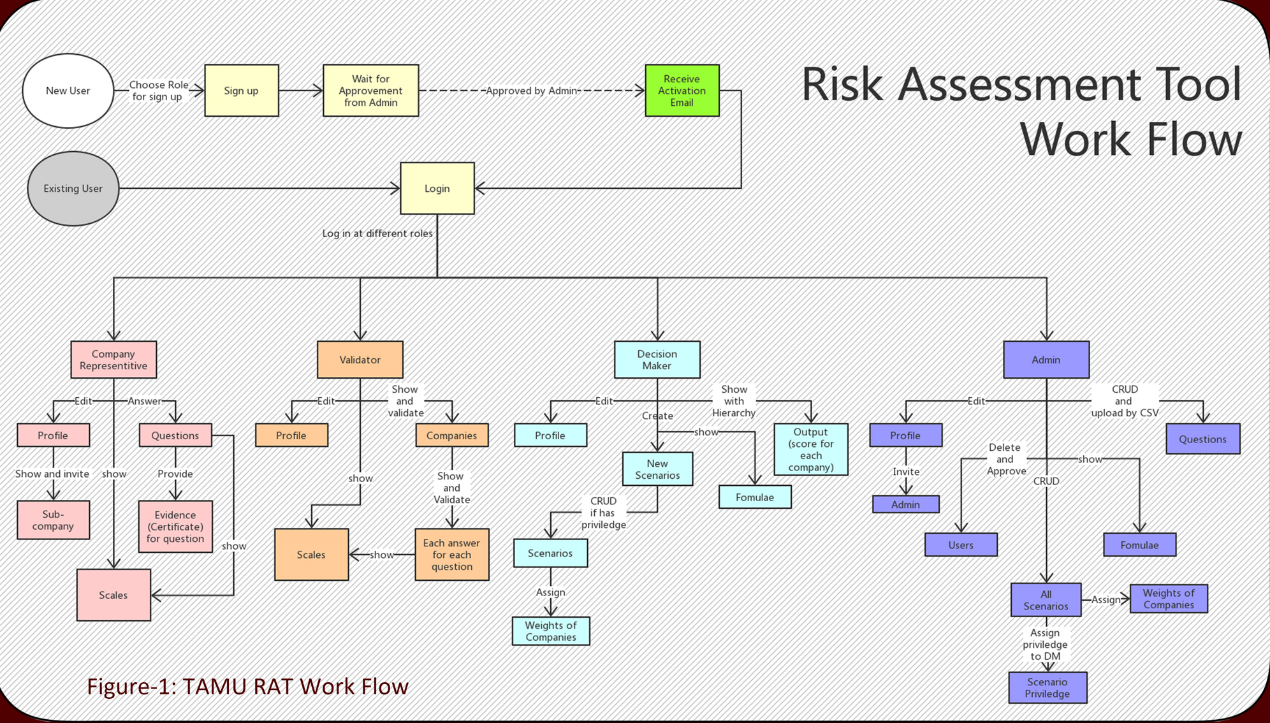
The Admin and the Decision Maker can both create scenarios to weight company on a different scale. The Admin decides which Decision Maker can see and edit which scenario.

Then the Decision Maker can see the output of each company and the score of each scenario to select the best solution for their project.

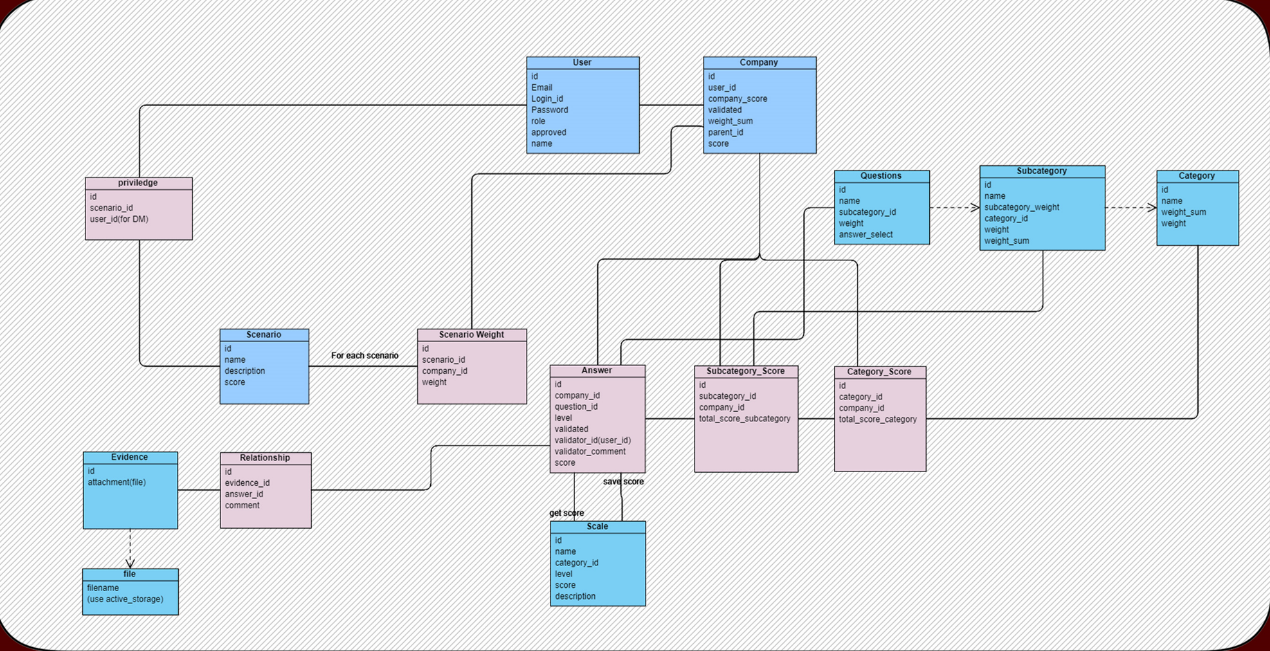
The administrator is in charge of setting up everything in this app, including users, questions, scenarios, and scales. Only the administrator can invite other administrators.

The Company Representative can also invite their sub-company after approved by the Admin, they will receive their activation email and login\_id.

**Work Flow**



**Database Diagram**



**User stories and 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 (3 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 (3 pts)  * 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 (3 pts)  * 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 (3 pts)  * 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 | Implemented feature 1, 3, 4, 5, 6, 13, 14 | 21 |
| Iteration 2 | Implemented feature 2, 7, 8, 9, 11, 15 | 18 |
| Iteration 3 | Implemented feature 10, 12, 16 | 9 |
| Iteration 4 | Improved the UI by JavaScript and made the app adapts to mobile devices. Navigation bar and contents on page can adapt to the window size. Navigation menu will be folded in an icon | 12 |

**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.

**Test with Cucumber/Capybara**

### Overview

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

* cucumber-rails: run automated tests written in plain language
* capybara: simulate user interactions with the 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 the scenarios are included in \*.feature files in /feature. Some fake users are created in seed.rb for the 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 receives an email after signing up, which informs the login id
* The 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 for email features.

### All users features

* User can see Home page
* User can see Help page
* User can see Profile
* User can change account information (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 a 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 for 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 evidence after answering a question in Questionnaire
* Company Representative can upload files by clicking Manage evidence 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%.

### Challenge

Operations based on JavaScript is hard to test, due to different versions of webdrivers' vague syntax. Most of the functions, such as confirm a JavaScript popup or upload files, are failed to implement a Cucumber test. There is no doubt that these features can be tested via Cucumber and Capybara. Further research is still needed on this part.

**Future work of TAMURAT**

1. Concurrent operation: not all the operation has been optimized for concurrent operation. For example, a user might edit one resource while another attempt to delete it.

2. Administrator invitation security issue: currently, the admin and invite another admin whose password is generated randomly. As long as they click the link in the activation email, they will be logged in without any check. If the email was hijacked or send to someone else, the system is at risk.

3. The initial setup: In this app, the initial administrator setup is through `seed` operation from rails. It's better to do a setup page for this system.

4. Better visual hints: We can have a red font for those questions that haven't been answered or those weight of the questions is set to 0.

5. Calculating the score: the score of the unvalidated answer is calculated as long as the `finalize` button is clicked.

6. Import a zip file into the system for answering the questions and all the evidence needed for the system.

* GitHub repo: <https://github.com/teamtreenewbee/TAMUART_new>
* Vimeo link (Client Interview):

<https://vimeo.com/user98199585/review/334522658/cb201fa5c8>

* Vimeo link (Demo):

<https://vimeo.com/user95224829/review/334537378/c084274362>

* Pivotal Tracker: [https://www.pivotaltracker.com/n/projects/2318271](https://www.pivotaltracker.com/n/projects/2319401)
* Heroku: <https://fast-shore-38378.herokuapp.com/>