

Report 2

Work Integrated Learning – SEB701

Francesco Ferraioli – n8323143 – francesco.ferraioli@connect.qut.edu.au

Software Engineering (EN40)

MonicIT (60 Days Placement)

Executive Summary

Executive Summary

Table of Contents

1.0 Work Place Background 3

2.0 Work Activities 4

2.1 Major Activity and Reflection 1 4

2.2 Major Activity and Reflection 2 4

2.3 Major Activity and Reflection 3 4

2.4 Major Activity and Reflection 4 4

2.5 Major Activity and Reflection 5 4

3.0 Conclusion 4

References 4

Appendicies 4

Appendix A: Work Log 4

# 1.0 Work Place Background

Work Place Background

# 2.0 Work Activities

## 2.1 Major Activity and Reflection 1

Major Activity and Reflection 1 – Airport model

Situation:

One of LMS’s main customer facing features is the booking form. The booking form allows a customer to book one or more trips with a limousine company. The booking board allows for automatic trip quotes, but it requires a number of things to be able to calculate it. This includes: trip distance and time, the pick up time, the payment method and the vehicle they wish to book for the trip. Using Google API, calculating the total distance and total time of the trip is a simple task, as it just requires the longitude and latitude of the pick up and drop off locations as inputs. However, many of the customers who wish to book a trip with a limousine, wish to either be picked up from the airport, or dropped off from the airport.

Task:

Instead of the customers having to look up the address for their departure or arrival airports, supplying the customer with a list of airports with prepopulated addresses was certainly the better option. My task was to implement a way for our clients to manage the airports they wish to display on their booking form. To allow this feature, each client must be able to manage their own list of airports. This means that a new entity must be introduced into the system, the entity of airports.

Action:

After speaking to both the boss and the lead engineer for the project, we settled on the schema for the airport table. The model will simply be a class that can be used to access the database table and records. The views will simply need a form to create and update airports and an index to list all the airport. Rails has a built-in scaffolding command to automatically generate a generic template for all this functionality.

After running the scaffolding command I did a bit of customization of the generic templates to match the current system look and feel. Furthermore, I finalized the CRUD operations for the airport model. The last implementation task was to add the airport functionality to the booking form, displaying a combo box, which lists all the available airports for the client. I then moved on to writing unit tests for the model functions as well as end to end tests to ensure the UI side of things acted as expected.

Results:

The client could now easy create, read, update and delete their airports, through the views available for managing this entity. Furthermore, the customer facing booking form would list out the airports on the combo box. After a bit of tweaking, the unit tests and the end to end tests all passed.

Learnt:

I learnt a lot of things during this task as this was the first rails entity I was implemented. I learnt about some important rails commands, including rails generate scaffold command which assisted me a lot in this task. I learnt a lot about the rails framework and how it functions with the database. Furthermore, I learnt how to write unit tests for rails models.

## 2.2 Major Activity and Reflection 2

Major Activity and Reflection 2 – Booking board validation

## 2.3 Major Activity and Reflection 3

Major Activity and Reflection 3 – ID vs Access Token

## 2.4 Major Activity and Reflection 4

Major Activity and Reflection 4 – Generic Template

## 2.5 Major Activity and Reflection 5

Major Activity and Reflection 5 – Best In Place

# 3.0 Conclusion

Conclusion

# References

**There are no sources in the current document.**

# Appendicies

## Appendix A: Work Log

## Appendix B: Certificate Of Time Worked

## Appendix C: Reflective Notes

|  |  |
| --- | --- |
| Situation | Booking board needed to support Airport selection |
| Task | Implement Airport model view and controller |
| Action | 1. Discussed with boss and lead engineer about the schema for the airport table 2. Implemented Airport model 3. Implemented Airport views and controller 4. Finalized Airport CRUD 5. Integrated Airport selection into booking form 6. Wrote unit test for Airport model functions |
| Result | * Airport CRUD was very easy to implement with Rails assisted scaffolding command * A client could view, create, edit and delete airports which they want to support as pick up or drop of locations * A customer using the booking form could select airports that the client supports * Airport model Unit tests pass |
| Learnt | * A model is equivalent to a table in the database * Rails makes implementing new CRUD very easy for developer with all the supporting commands * Leant a lot of new powerful rails command * Updating the schema for a model (table) is also easily done * Controllers work closely with the views as they respond to requests |

|  |  |
| --- | --- |
| Situation | Validation was needed for booking form |
| Task | Implement Validation using JQuery for booking form |
| Action | 1. Did some research on JQuery validator 2. Began implementing validation for inputs for one trip 3. Enhanced the validation to incorporate validation for multiple trips in one booking 4. Wrote UI tests for validation |
| Result | * JQuery validator was not documented heavily * Was helped out by senior developer * Trip validation was in place in booking form * UI tests were a bit flaky especially due to Google Maps issues |
| Learnt | * Sometimes useable documentation isn’t available and experience is much more valuable * JQuery makes it easy to enhance features * Leant a lot of JQuery functionality, including JQuery DOM manipulation * UI tests are difficult to write at times, especially when replying on third party tools |

|  |  |
| --- | --- |
| Situation | Booking form was using incrementing ID to access DB – Security Issue |
| Task | Modify booking form to use Access Tokens instead |
| Action | 1. Investigated use of ID throughout the booking form 2. Added access token both to trip and booking model 3. Implemented functions to calculate unique access tokens 4. Refactored booking form to work with access token instead of ID 5. Fixed failing tests |
| Result | * ID was used heavily in booking form * Refactoring took a long time * A lot of tests began to fail due to element ID mismatch * Fixing tests also took a long time |
| Learnt | * Exposing incrementing ID to customer facing tools is a big security vulnerability as attackers can easily increment or decrement the ID stored in the page to alter unauthorized resources * Refactoring a big page takes a long time * Using element ID for testing is not the best approach and tests will fail upon changes. |

|  |  |
| --- | --- |
| Situation | A generic template was needed for CRUD operations of many models for consistency |
| Task | Create a generic template that all models can use to implement CRUD operations |
| Action | 1. Discuss with boss, what the general UI should look like 2. Began implementing generic and modular HTML templates for views 3. Began implementing generic and modular JavaScript to run the views 4. Began refactoring model CRUD operations using the generic and modular templates and scripts |
| Result | * My work resulted in a lot of code and mark up to be removed as generic templates took care of it * Code was much more maintainable, as a lot of the models were using the same code for simple CRUD operations * CRUD operations was enhanced and made simpler for the user due to the changes |
| Learnt | * Generic, modular and reusable code is very important in large software systems * Rails makes this easy by the use of partials * How to use partials * How to write generic and modular code and how to use it correctly * Making generic code makes it much easier and quicker to implement enhancements and will be applied to all models |

|  |  |
| --- | --- |
| Situation | Functionality for updating models needed enhancements |
| Task | Integrating ruby gem ‘Best In Place’ to views |
| Action | 1. Read up on ‘Best In Place’ documentation 2. Began integrating ‘Best In Place’ into application 3. Added ‘Best In Place’ features to views for easy updating of model fields 4. Wrote helper functions to test ‘Best In Place’ functionality |
| Result | * Best In Place needed extra configuration than expected to work with current version of Rails * Users can now update model properties by a simple click now thanks to ‘Best In Place’ instead of having to open a form for the whole model * Tests were added to test the newly added ‘Best In Place’ functionality |
| Learnt | * Ruby gems are very powerful and easy to add to an application, even if extra configuration is needed * Learnt how to use the ‘Best In Place’ gem to provide the user with a simply way of updating model properties * New functionality always means new tests and adding helper functions can really help in test cleanliness and maintainability |