

Report 1

Work Integrated Learning – MonicIT

Francesco Ferraioli – n8323143

EXECUTIVE SUMMARY

This report will give a short overview of my first thirty days of industry-based work as a software engineer. In the first thirty days of placement, I found myself working for MonicIT, a small software company in Brisbane CBD. At MonicIT I was able to move quite quickly up in the company, starting off as a tester for their android app to one of the main developers of the company. This meant that I found myself participating in a variety of different tasks. Being a small company I was able to get a lot of mentoring from the senior developers. I gained a lot of experience in my first thirty days of my placement at MonicIT, not only technical skills, but I also learned the everyday duties and tasks of professionals in my field.

TABLE OF CONTENTS

1.0 Work Place.....	4
1.1 Introduction	4
1.2 Employer Organisation	4
1.3 Work Environment Conditions	5
2.0 Work Activities.....	7
2.1 Major Activity and Reflection 1	7
2.2 Major Activity and Reflection 2.....	9
3.0 Conclusion	11
References	11
Appendices	12
Appendix A: Work Log	12
Appendix B: Reflective Notes.....	15
Appendix C: Sample Invoice	19
Appendix D: LMS Login Page.....	19
Appendix E: Failed Login	20
Appendix F: Successful Login	20
Appendix G: Authorization.....	20

1.0 WORK PLACE

1.1 INTRODUCTION

Before commencing my work at MonicIT, I had little to no experience besides my studies at QUT in my Bachelor of Software Engineering. Midway through my fifth semester of the degree I saw an opportunity for a tester position at MonicIT and I decided to take it. At that stage I had completed more than half of my degree and I felt that I had already gained many vital skills and this provided me with the confidence of fulfilling the position. Many of the subjects I undertook at QUT involved working in teams as well as an individual. These subjects gave me the experience I needed to know what is required when working in a team and taught me the crucial skills of teamwork. They taught me not only that communication is key, but also how to communicate effectively. The individual aspects to these subjects helped me to become a stronger problem solver, which was certainly something I needed if I wanted to be a member of a team in the industry. By that time I had received outstanding grades for those subjects, with a GPA of 6.6875.

1.2 EMPLOYER ORGANISATION

My first thirty days of industry-based work were undertaken at MonicIT. MonicIT was founded in 2008. They started off by providing customized software to companies but later began focusing on software for limousine companies as the founders saw some real potential in the market. After a limousine company owner approached them with interest for software to assist them to run their business, they did some more research and found that many limousine companies were in need of this kind of software. After that, they decided to spend all their resources on creating this software for limousine companies. Just like all their other software, Limousine Management System (LMS), was written as a Ruby on Rails web app. MonicIT decided to take this software a step further and began creating an android app for limousine drivers to use that would communicate with the LMS.

My main responsibilities and duties at MonicIT during my first thirty days of placement revolved around ensuring that the android app worked as expected. This involved running tests on the android app against the desired functionality and if bugs were found, to report

them in a professional manner to the developers to fix. As the app and LMS were closely linked, this also meant that the functionality of LMS was also tested and sometimes bugs were found there too. Initially the testing involved only manual testing, but then moved to automated tests which involved me writing scripts to run the User Interface.

1.3 WORK ENVIRONMENT CONDITIONS

Professional Indemnity insurance was designed to protect workers against legal costs and claims for damages to third parties, which may arise out of an act, omission or breach of professional duty in the course of your business (Aon, 2014). Damage can include but is not limited to, any loss of data that belongs to the third party, and certainly includes any financial costs that may arise for the company. A majority of the web applications that are being implemented can be considered e-commerce applications (Schafer, Konstan, & Riedl, 1999). E-commerce applications deal very heavily with financial transactions. These applications have to be not only impeccably correct, but also heavily secure. Any breach of these can result in damage to the third party (clients). A breach could be not securing a transaction well enough that it can be seen or even altered, causing confidentiality and possibly even integrity breaches. It is vital for software engineers to have professional indemnity insurance, especially those working on e-commerce applications. The application MonicIT was developing, LMS, frequently dealt with financial transactions, paying for trips through the application and is certainly classed as an e-commerce application. For my placement at MonicIT, it was crucial for an inexperienced professional as myself to have this.

Workplace Health and Safety is the discipline that deals with protecting the health and safety of all stakeholders in the workplace (Davitt, 2012). Workplace health and safety is not a major issue for software companies but certainly needs to be considered. MonicIT's office is located on the top floor of one of the highest building in the Brisbane CBD. Workplace health and safety was certainly a part of the building and MonicIT, as a customer of the building, needed to abide by any of defined rules regarding workplace health and safety that were set for the building. The building had stairs to use in case of a fire and in every room was a diagram detailing the meeting place in case of a fire and how to direct yourself there. It was important for workers at MonicIT to be aware of the procedures in case of emergencies. Fire detectors were present in every room of the building and were tested regularly.

Intellectual Property is used to describe the results of creative and innovative endeavours (Dutfield, 2008). Intellectual Property is a vital part of any company and certainly crucial for software companies. Without Intellectual Property a developer could take any work that they does to be their own. At MonicIT it was made clear from the start that any work that is done using MonicIT's resources were the property of MonicIT. MonicIT ensured that Intellectual Property was part of every contract that an employee signs with them. For developers, especially it is important to know and understand this and MonicIT made it very clear in the contract. Even for a position like mine, the contract stated very clearly the agreed terms regarding Intellectual Property.

Quality Assurance is any systematic process of checking to see whether a product or service being developed is meeting specified requirements (Rouse, 2014). It is crucial for any company to have quality assurance when creating a product for customers. MonicIT valued quality assurance highly, as most companies do (Rouse, 2014). My main duties during my first thirty days of placement at MonicIT revolved around ensuring that the product was meeting the requirements. This is vital for customer satisfaction and thus revenue for the company.

MonicIT code of practice revolved heavily around respect for one's self and others. Respect for one's self included being honest and truthful not only about your work but also about the action you undertake as a MonicIT employee. Harassment in the workplace was unacceptable at MonicIT. Drinking alcohol while at work was also very frowned upon at MonicIT, as the founders wanted to ensure that all employees could always give 100% of their efforts whilst at work. Furthermore, working in a very classy building in the Brisbane CBD, it was important to dress and act professionally and MonicIT was sure to follow those expectations.

2.0 WORK ACTIVITIES

2.1 MAJOR ACTIVITY AND REFLECTION 1

The sending of invoices was a major part of the android application. The functionality of sending invoices had been implemented but was yet to be tested. The android application would be used by drivers who owned one limousine but who did not belong to a limousine company. If a company is out of vehicles or drivers, they could use LMS to offload jobs to drivers who use this application. The company however would want a share of the earning. Thus, depending on the payment type, an invoice would need to be sent between the company and the driver. If the payment type was Cash, then it would be the driver who would receive the cash and the limousine company who offloaded the job would need to send an invoice to the driver requesting their part of the earnings. On the other hand, if the payment type was credit card and the trip was already paid for, then the company would have the money and the driver would then need to send an invoice to the company requesting to be paid for the job that they completed.

On top of this, drivers could decide with the company if invoices were to be sent on a per job basis or a weekly basis. When the configuration is set to per job basis, invoices are to be sent every time a driver finishes a job. As previously stated, the payment type determines the direction of the invoice. On the other hand, if the configuration is set to weekly basis, the total jobs for that week are all included in the one invoice. Invoices that contain the full week's work will need to sum up the total of jobs, taking into account who owes whom money and sending the invoice to the entity who, once the calculations are done, owes the money. A sample invoice that was sent from the phone is shown in Appendix C. The invoice is billed to GetHummered, which was a limousine company that used LMS, and Test Driver 1, which was the driver I was using for my testing. Test Driver 1 had completed two trips in which the payment method was credit card and thus the invoice was sent to GetHummered from Test Driver 1 containing the two trips.

Transactions were going to be determined from this functionality and therefore it needed to work perfectly. MonicIT prided themselves in flawless systems and financial matters are very important to customers and thus to MonicIT. As this functionality was very crucial to the system, it was important for me to test it thoroughly. This meant taking into account many edge cases, as well as the normal test cases for this feature.

Normal test cases involved either a driver set to per job basis or setting the invoice frequency to weekly but only completing a minimal number of jobs, all of the same payment type. Edge cases on the other hand would include a weekly invoice with many jobs of varying payment types. On top of that, a very important test case revolves around changing the invoice configuration from per job basis to weekly basis. The driver is allowed to change this configuration through the android app and this functionality needed to be incorporated in the edge cases.

After brainstorming all the test cases I began writing the steps for each test case in an excel file. Each step had a section to input the expected result and actual result. With the help of my boss, we filled in the expected results. Once this was finished, I began undertaking the tests with the android app. All of the normal test cases were completed without fail. Invoices showed correct amount and were issued to the correct entity depending on the payment type. Even weekly invoices containing numerous trips with the same payment type were generated with the correct amount and issued to the right entity.

Edge cases on the other hand found some bugs in the system. When testing multiple payment types on the one weekly invoice, it was found that even if the total was calculated correctly, it was sent to the incorrect entity. Moreover, when changing the configuration from per job to weekly, all the jobs completed for that week were included in the invoice even if they had already been invoiced before when it was set to per job basis. Furthermore, when changing from weekly to per job, some invoices were not generated at all and thus not sent. I then proceeded to documenting all the bugs found, including the steps to reproduce them. I spoke with the developer, explaining to him the bugs I found.

One of the major things I learnt from this experience was the importance of the initial brainstorming of test cases, both the normal and the edge test cases. Without testing the edge cases, this feature would have passed the Quality Assurance phase and gone to the customers in production, resulting in many problems for them and for MonicIT. Furthermore, this experience really taught me the importance of documentation. Documenting each step for each test was crucial when it came time to informing the developer how to reproduce the bug.

2.2 MAJOR ACTIVITY AND REFLECTION 2

Authentication and Authorization play a major role in many software systems, especially web based applications (Dániel, 2008). Authentication is the process used to determine whether someone is who they claims to be (Gollmann, 2000). Authentication is usually done through the use of a log in and password – especially for web based applications – but there are many other forms of methods of authentication. On the other hand, authorization is the process of granting or denying access to a entity when they request a particular resource (Ahn, 2000). A resource can be data, like records in the database, or features, like a page that allows you to undertake particular functionality that a user does not have access to.

LMS was certainly very dependent on both authentication and authorization. Authentication was needed to ensure that a random user could only log in to the system if they were a certified user of the system. Appendix D shows the log in page for LMS; this will be the centre of most of the authentication test cases. Authorization on the other hand was used for a couple of things. Firstly it was used to ensure that different clients using the database could not, create, read, update or delete anything that did not belong to them. This could be deleting a record in the database that belonged to another client. Another form of authorization which was used in LMS revolved around role based permissions. There are different types of users that can log in to LMS: company owner, manager, driver; and each of them would have different permissions. For example, the manager should not be able to update the available time for the drivers, and similarly, the driver should not be able to update his salary. Different user means different permissions and rules.

I began brainstorming ideas for test cases along with the lead developer of the project. We were able to discuss the very basic normal cases, as well as some important edge cases. Normal cases involve successfully logging in with correct credential and unsuccessfully attempting to log in with incorrect credential. Appendix E and F respectively illustrate what the user would see on an unsuccessful and successful log in. Password changing and testing that new password works and the old password fails was also considered a normal test case. Edge cases were more covered by authorization. A normal test case for authorization is to ensure that a particular user is not provided with a link they shouldn't have access to. A very similar but more particular test case is ensuring that if a user types in a URL that will lead them to a resource they do not have access to, they are denied that access. This would be

considered an edge case as a developer can easily disable a link, but cannot stop them from typing in a URL which will lead to a resource they do not have access to. Appendix G demonstrates what the user would see if they try to access a resource they do not have access to. All these need to be tested and I was given the task to write automated test for this.

Automated test gives the ability to test functionality over and over again with no input from the developer beside some sort of command to start the test. The automated test simply run the UI on the browser performing actions like clicking a button, entering text in a text box and selecting an option from the combo box.

I moved to writing the test scripts for the normal test cases. After having finished I ran through the suite of tests and found that many failed. After investigating the reason for the failure, I found that the reason was due to test bugs. Test bugs are bugs found in the test scripts. Test bugs give out false negatives. Test bugs include two types of issues: referencing issues and rendering speed issues. Referencing issues occur when an element is referenced incorrectly, causing the driver to be unable find the element to interact with. On the other hand, rendering speed issues occur when an element has not been rendered yet and the driver tries to interact with it but cannot find it as it is not yet rendered.

After speaking with the lead developer, he informed me that it is crucial to run test scripts as you are writing them, this way if you find a test bug, you can fix it straight away. After fixing the test bugs all the tests came back green (pass). I then started implementing the edge cases, but this time running the tests as I was writing them. Once I had finished writing the tests, I ran them all again and they all came back green.

A very clear lesson that I learnt from this experience is to run the test scripts as you are writing them, this way you minimize the chances of test bugs down the track. Through this exercise I also gained a deeper understanding of the importance of authentication and authorization in systems like LMS.

3.0 CONCLUSION

The first thirty days of work placement at MonicIT provided me with a wide range on real world professional experience. I was able to familiarize myself with the processes used in software companies as well as gain many professional skills and learnt many valuable lessons that will help me throughout my career as a software engineer. The first thirty days at MonicIT have been a great learning experience and enhanced the learning I had done at university. It allowed me to apply the knowledge I had gained as well as gaining additional knowledge. I was able to apply the knowledge I gained at university to real situations to assist me, not only through the rest of my degree, but also throughout my whole career as a software engineer.

REFERENCES

- Ahn, G. J. (2000). Role-based authorization constraints specification.
- Aon. (2014). *Professional indemnity insurance*. Retrieved August 12, 2014, from AUSDANCE: <http://ausdance.org.au/articles/details/what-is-professional-indemnity-insurance>
- Dutfield, G. (2008). Global intellectual property law.
- Davitt, M. (2012, April). Workplace health and safety act 2012.
- Dániel, T. &. (2008). Authentication and authorization.
- Gollmann, D. (2000, January). What is authentication?
- Schafer, B., Konstan, J., & Riedl, J. (1999, November). E-Commerce Recommendation Applications . Minneapolis.
- Rouse, M. (2014). *Quality Assurance (QA)*. Retrieved August 12, 2014, from Tech Target: <http://searchsoftwarequality.techtarget.com/definition/quality-assurance>

APPENDICIES

APPENDIX A: WORK LOG

Day	Date	Hours	Type of work	Description of work
1	Monday, 11/03/2013	5	Manual Testing / Documenting	Manual Tests on Android App Documenting Results Found Bug and Reported bug: Reset of the icon counters (App)
2	Wednesday, 13/03/2013	8	Manual Testing / Documenting	Manual Tests on Android App Documenting Results Found bug: Date selection Found bug: Found 500 response (Web)
3	Thursday, 14/03/2013	8	Manual Testing / Documenting	Documenting bugs Writing steps to reproduce bugs
4	Monday, 18/03/2013	5	Manual Testing / Documenting	Manual Tests on Android App Documenting results Proposed Enhancement to functionality
5	Wednesday, 20/03/2013	8	Manual Testing / Documenting	Manual Tests on Android App Documenting results Proposed Enhancement for user friendliness
6	Thursday, 21/03/2013	8	Manual Testing / Documenting	Manual Tests on Android App Documenting results Found bug with invoicing
7	Monday, 25/03/2013	5	Manual Testing / Documenting	Cross Platform Testing of Android App
8	Wednesday, 27/03/2013	8	Manual Testing / Documenting	Documenting current features

9	Thursday, 28/03/2013	8	Manual Testing / Documenting	Documenting current features
10	Monday, 01/04/2013	5	Autonomous Testing / Documenting	Began learning how to write autonomous tests Testing Tool: TestWise Testing Language: RWebSpec/Selenium
11	Wednesday, 03/04/2013	8	Autonomous Testing / Documenting	Continued learning how to write autonomous tests
12	Thursday, 04/04/2013	8	Autonomous Testing / Documenting	Continued learning how to write autonomous tests
13	Monday, 08/04/2013	5	Autonomous Testing / Documenting	Continued learning how to write autonomous tests
14	Wednesday, 10/04/2013	8	Autonomous Testing / Documenting	Writing my first autonomous tests
15	Thursday, 11/04/2013	8	Autonomous Testing / Documenting	Writing autonomous tests to run against webapp
16	Monday, 15/04/2013	5	Autonomous Testing / Documenting	Writing more simple autonomous tests
17	Wednesday, 17/04/2013	8	Autonomous Testing / Documenting	Writing autonomous tests to expose bugs
18	Thursday, 18/04/2013	8	Autonomous Testing / Documenting	Writing autonomous tests for validation

19	Monday, 22/04/2013	5	Autonomous Testing / Documenting	Writing more complex autonomous tests Authentication Authorization
20	Wednesday, 24/04/2013	8	Autonomous Testing / Documenting	Exposing known bugs through autonomous tests
21	Thursday, 25/04/2013	8	Autonomous Testing / Documenting	Finding new bugs through autonomous tests
22	Monday, 29/04/2013	5	Autonomous Testing / Documenting	Fixing test issue where test where giving false negatives
23	Wednesday, 01/05/2013	8	Implementation / Autonomous Testing	Began learning Ruby on Rails for the new project
24	Thursday, 02/05/2013	8	Implementation / Autonomous Testing	Wrote HTML for the web app
25	Monday, 06/05/2013	5	Implementation / Autonomous Testing	Wrote HTML for the web app
26	Wednesday, 08/05/2013	8	Implementation / Autonomous Testing	Wrote HTML for the web app
27	Thursday, 09/05/2013	8	Implementation / Autonomous Testing	Began writing JavaScript for interacting with HTML
28	Monday, 13/05/2013	5	Implementation / Autonomous Testing	Wrote my first ajax call used to update fields on the database via the webapp

29	Wednesday, 15/05/2013	8	Implementation / Autonomous Testing	Finished implementing ajax call to update fields on the database via the webapp
30	Thursday, 16/05/2013	8	Implementation / Autonomous Testing	Wrote tests for the ajax call functionality

APPENDIX B: REFLECTIVE NOTES

Situation	The android application has to send invoices
Task	Undertake manual test against the invoice functionality of the android app
Action	<ol style="list-style-type: none"> 1. Brainstormed normal test cases 2. Brainstormed edge test cases 3. Wrote steps for tests 4. Wrote the expected results 5. Undertook the tests 6. Documented results
Result	<p>Some of the tests passed – the outcome was what was expected</p> <ul style="list-style-type: none"> - Invoice had the right amount - Invoice included correct items <p>Some of the tests failed – the outcome was different than expected</p> <ul style="list-style-type: none"> - Some invoices were issued to the incorrect person - Some invoices didn't send at all
Learnt	<ul style="list-style-type: none"> - Testing functionality after implementation is crucial - Write down all the steps taken so that if an error occurs, it can be reproduced - Invoices are extremely important and must be tested thoroughly - Good practice to brainstorm test ideas and test steps before commencing testing - Good practice to write down the expected results next to each step - Good practice to think of a good set of normal as well as many edge cases

Situation	The web application implements authentication and authorization
Task	Write autonomous tests to test authentication and authorization functionality
Action	<ol style="list-style-type: none"> 1. Brainstorm test cases for authentication and authorization (normal and edge) 2. Write down the steps for each test 3. Write down the expected outcome for each step 4. Turn the worded steps into scripts that can be run against the User Interface 5. Run the tests
Result	<ul style="list-style-type: none"> - After writing the test cases in RWebSpec some of the test passed and some didn't - Some of failing test cases failed due to test bugs. The tests were not written correctly and caused failure. Most of them were due to incorrect ID names for the elements. - Other failing test cases were failed due to browser speeds versus driver speed. The driver would be too fast and would try to interact with an element when the browser has not yet finished loading it.
Learnt	<ul style="list-style-type: none"> - It is good practice to test your test cases as you write them to minimize test bugs - Browsers can be slower than the driver and to be aware of this when implementing tests. This can be done by making the driver wait for something to happen that is known to happen when the page is being loaded, before trying to access elements. - Authentication and Authorization is vital and needs to be tested thoroughly - Authentication includes changing and restoring of passwords

Situation	There are known bugs in the system
Task	Write autonomous tests to expose the bugs
Action	<ol style="list-style-type: none">1. List down the known bugs2. Write down the steps to expose each bug3. Think of any other way that any of the bugs may be exposed and write the steps for those4. Turn the worded steps into scripts that can be run against the User Interface5. Run the tests and ensure that if they fail, they fail because they are exposing the bugs
Result	<ul style="list-style-type: none">- After writing the test cases all of the test failed- Some tests failed because of the bug- Some tests failed because of a different bug in the web app- Some tests failed because of a test bug
Learnt	<ul style="list-style-type: none">- When exposing bugs with autonomous test, it is essential to ensure that they fail because of the bug, not due to a test bug- Exposing bugs with autonomous tests is essential in Agile as these tests will run every build and these can be used to ensure the bug doesn't exist even in the future.

Situation	My colleague had implemented a feature but hadn't had the time to write tests for the feature
Task	Write autonomous tests for the functionality
Action	<ol style="list-style-type: none">1. Take a look at the feature my colleague had implemented2. Undertake manual tests to understand the feature3. Write down normal test cases for the feature4. Write down edge test cases for the feature5. Turn the worded steps into scripts that can be run against the User Interface6. Run the tests
Result	<ul style="list-style-type: none">- The tests came back all green. This occurred because I was careful about the way I implemented the tests and ran the tests as I was implementing them and only once they were complete, would I truly be testing the functionality and not the test.
Learnt	<ul style="list-style-type: none">- In Agile, a feature has not finished to be implemented until there is at least a test for it- Firstly running tests manually is not bad practice when writing test scripts- How important tests are in development- How important it is to run tests as I am implementing them to remove any possible test bugs

APPENDIX C: SAMPLE INVOICE

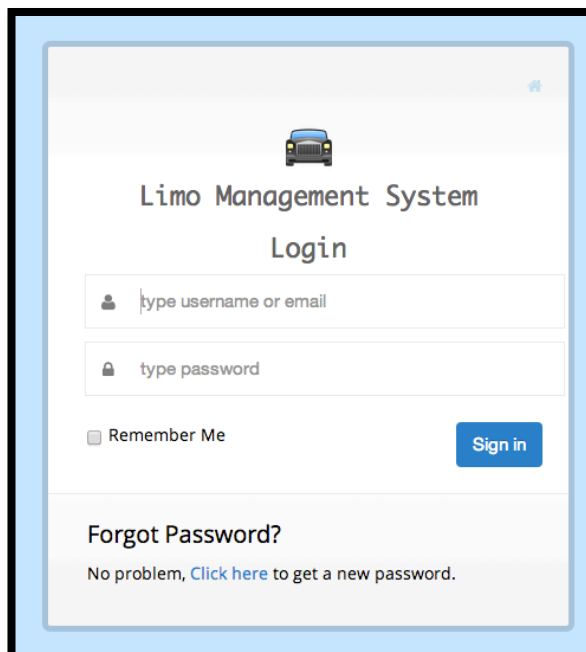
Test Driver 1
111 Eagle Street
BRISBANE QLD 4000


GetHummered
3 Monaco St, Surfers Paradise
GOLDCOAST QLD 4217

Invoice # 0000001
Invoice Date August 15, 2013
Amount Due \$110.00 AUD


Item	Description	Unit Cost	Quantity	Line Total
Trip 1	From: City To: Airport	50.00	1	50.00
Trip 2	From: Airport To: City	60.00	1	60.00
Total				110.00
Amount Paid				-0.00
Amount Due				\$110.00 AUD


APPENDIX D: LMS LOGIN PAGE





**Limo Management System
Login**

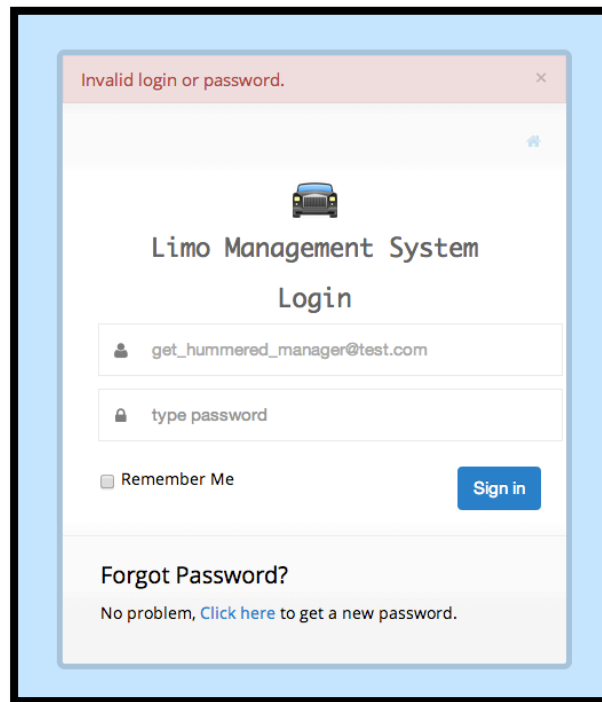
 type username or email

 type password

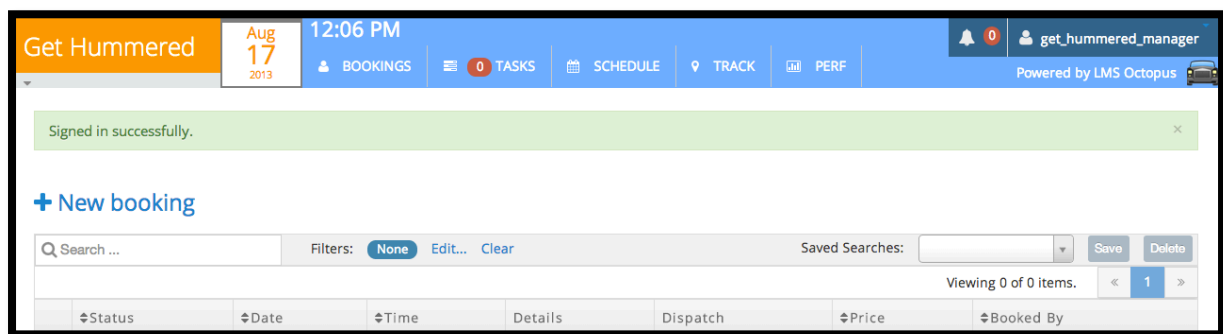
☐ Remember Me **Sign in**

Forgot Password?
No problem, [Click here](#) to get a new password.

APPENDIX E: FAILED LOGIN



APPENDIX F: SUCCESSFUL LOGIN



APPENDIX G: AUTHORIZATION

