

**Designing & Developing Cloud Applications**

**CT071-3-5-3-DDAC**

**DDAC Individual Assignment**

**UC3F1706SE**

Lecturer Name : Dr. Kalai Anand Ratnam

Student Name : Lee Kar Mun

TP Number : TP034870

Date Assigned : 11th December 2017

Due Date : 13th April 2018

# Acknowledgment

With this opportunity, I would like to first thank my parents, for their never-ending support, in all aspects, to which I am grateful and thankful of. Next, I would like to thank Furthermore, Dr.Kalai Anand A/L Ratnam, the course’s lecturer, for him who educated and guided the class on cloud computing. On top of that, through tutorial practices, a better understanding was delivered. Besides that, I would like to show my gratitude to my friends and classmates for their patience, tolerance and their support for always being kind and willing to help, sincerely thank you.

Contents

1. [Acknowledgment 2](#_Toc511399612)
2. [Introduction 4](#_Toc511399613)

[Project Background 4](#_Toc511399614)

[Objectives and Scopes 4](#_Toc511399615)

[Requirement Specification 5](#_Toc511399616)

[Summary of Major Functions/ Solution Contents 5](#_Toc511399617)

1. [Project Plan 6](#_Toc511399618)
2. [Design 8](#_Toc511399619)

[Implementation Architectural Diagrams 8](#_Toc511399620)

[Cloud Architectural Diagram 8](#_Toc511399621)

[Design Consideration 10](#_Toc511399622)

[Modelling 10](#_Toc511399623)

[Use Case Diagram 10](#_Toc511399624)

[Sequence Diagram 15](#_Toc511399625)

1. [Implementation 19](#_Toc511399626)

[Application Development 19](#_Toc511399627)

[Azure Publishing 22](#_Toc511399628)

[Application Scaling 28](#_Toc511399629)

[Reliability and Performance 30](#_Toc511399630)

1. [Testing 31](#_Toc511399631)

[Unit Testing 31](#_Toc511399632)

[Login 31](#_Toc511399633)

[Registration 31](#_Toc511399634)

[Schedule 32](#_Toc511399635)

[Vessel Booking 32](#_Toc511399636)

[Performance 34](#_Toc511399637)

[Analysis 36](#_Toc511399638)

1. [Managed Database 37](#_Toc511399639)
2. [Conclusion 39](#_Toc511399640)
3. [References 40](#_Toc511399641)

# Introduction

## Project Background

Found in almost a century back, with just a decade short, the 90 years old company Maersk Line is the largest operating unit, for the Danish conglomerate, A.P. Moller-Maersk Group. Maersk is a global container division, that consists of 374 office operating in over 116 countries. The company is ran by a big amount of employees, around 32,000 employees, to be precise, with 25,000 people working in offices and another seven thousand people as sea farers. Around six hundred vessels with a capacity of 2.6 million TEU, are operated under the company.

Throughout the years, Maersk has expanded to a few time big in term of its business, with its cargo doing well. The business is operated over hundreds of countries, with goods being shipped worldwide. However, the company has come to know most of the goods shipped is nearing to a full capacity, where the company tries on cloud powered solutions, which was believed to would help in the situation. For this, the company has come to think, how could the situation to turn for a better, taking in consideration the IT perspective as well.

Therefore, Maersk has then came to a decision on virtualizing the data centers and servers. The company has already approach the virtual environment in the recent years, through Microsoft Azure. The company has now wishes to set up the desktop environment to the container management, for all to the cloud.

## Objectives and Scopes

1. To look into a few perspectives, for how the web application developed to fit with the cloud service provider, the cloud environment, the connection and the software developing environment.
2. Design and implement Maersk’s web application.
3. Deploy Maersk’s web application on cloud, Microsoft Azure.

## Requirement Specification

* From import, export and transshipment processing to gate operations.
* To be able to scale the solution to meet the needs of demands during peak seasons.
* Improves profitability, reduce costs, increases productivity, eradicates errors and optimizes resources to future-proof your cargo handling business for high performance.
* Assurance & reliability through Failover Management.
* Accurately allocates inbound containers to yard locations and plan outbound containers to individual haulier vehicles, delivering an exceptional level of automation and removing human error.
* Manage your entire booking process from schedule search to booking confirmation.

## Summary of Major Functions/ Solution Contents

* Design & Develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App) or on AWS Elastic Beanstalk.
* Consume Relational Database
* Consist of 5 - 10 interlinked pages.
* Provide quality content and design.
* Analyze web application performance with monitoring tools.
* To be able to scale the solution to meet the needs of demands during peak seasons.
* Source code to place in source control management service.

# Project Plan





# Design

## Implementation Architectural Diagrams

### Cloud Architectural Diagram



Figure 4.1.1: Cloud Architecture Diagram

As Maersk has already previously handed over some IT environment hosting with Azure, the Maersk Line Container Management System will be topping up on that, to bring in more business opportunity with Azure. The container management system that has business connection over half of the globe, due to its business nature, will emerge and improvise the system better through the cloud.

With Maersk, already have planted most of its business root on most of the countries, therefore a few servers will be rooted at a few different regions, to ensure the availability of the system. This could be achieved if a server of a specific region is down, or experience some unexpected traffic, the servers from another region could offer instance support.

MSSQL will be a package coming along with the implementation to support the database side, by retrieving data, The choice behind the decision, is accounted to a few factors, its function, budget and availability. One of the reason Maersk intended to go for cloud was to cut down some budget on the operation cost, therefore, MSSQL fufills pretty much of the requirements, as it stores data, low budget and is available.

## Design Consideration

To design the application, a few considerations needs to be taken into account, as well some assumptions. These assumptions and design considerations includes:

1. Maersk’s web application will be managing Maersk major operating system, like booking and scheduling shipments.
2. Microsoft Azure is already hosting some of Maersk’s IT environment, however Maersk’s web application will be requiring more than the available ones, therefore, a budget of RM400 will be given per month to host the application.
3. Available in both version, desktop and mobile.

## Modelling

### Use Case Diagram



Figure 4.3.1: Maersk Line Container System Use Case

#### Use Case Specification

|  |  |
| --- | --- |
| Use case ID | 001 |
| Use case | Register |
| Summary | This use case allows the actor, user to register a new account to access the system |
| Dependency | - |
| Actors | Agent, Administrator |
| Preconditions | User should first access the system. |
| Description of main sequences | 1. User should select ‘Register’. 2. User should fill in the required fields. 3. After filling in the details, user should select the ‘submit’ button. |
| Description of alternative sequences | 1. If the email is already registered in the system, a message indicating ‘user is already registered’ will be prompted. 2. If email doesn’t meet the email standard, upon data validation, an error message will be prompted. 3. Generally, if the other information, like address’s postcodes and contact number is against the data type prefixed, an error message will pop out indicating a wrong type of data. 4. In the confirm password field, if the password entered does not match the password, an error message will be generated. |
| Post conditions | Agent is registered to the system. |

|  |  |
| --- | --- |
| Use case ID | 002 |
| Use case | Login |
| Summary | This use case allows the actors to login to the system. |
| Dependency | - |
| Actors | Agent and Administrator |
| Preconditions | User should first access the system. |
| Description of main sequences | 1. User should select ‘Login’. 2. User should fill in the required fields. 3. After filling in the details, user should select the ‘Login’ button. |
| Description of alternative sequences | 1. If user enters an invalid email, basically a record that is not found in the database, a message indicating ‘invalid email’ will be prompt. 2. If password does not match the one registered with the system, an error message of invalid password will be prompt. |
| Post conditions | User login success. |

|  |  |
| --- | --- |
| Use case ID | 003 |
| Use case | Logout |
| Summary | This use case allows the actor, user to logout of the system. |
| Dependency | - |
| Actors | Agent and Administrator |
| Preconditions | User should be logged in to the system. |
| Description of main sequences | 1. User should select ‘Logout’. |
| Description of alternative sequences | - |
| Post conditions | User logout successful. |

|  |  |
| --- | --- |
| Use case ID | 004 |
| Use case | Manage Schedule |
| Summary | This use case allows the actor, to manage the schedules. |
| Dependency | Extend <<Create Schedule>>, Extend <<Modify Schedule>>, Extend <<Delete Schedule>> |
| Actors | Agent and Administrator |
| Preconditions | User should first be logged in, to the system. |
| Description of main sequences | 1. User should select ‘Schedule’. |
| Description of alternative sequences | 1. Users that are not logged in will not be able to access this part of the system. 2. Users are allowed to create, modify and delete schedules. |
| Post conditions | Available schedule displays, with other functions option for user to modify, create and delete. |

|  |  |
| --- | --- |
| Use case ID | 005 |
| Use case | View Schedule |
| Summary | This use case allows the actor, to create a view schedule. |
| Dependency | - |
| Actors | Agent and Administrator |
| Preconditions | User should first be logged into the system. |
| Description of main sequences | 1. User should select ‘View Schedule’. |
| Description of alternative sequences | 1. If there are no schedules available, no record will be showed. |
| Post conditions | Schedule displayed. |

|  |  |
| --- | --- |
| Use case ID | 006 |
| Use case | Manage Vessel Booking |
| Summary | This use case allows the actor, to manage the vessels’ bookings. |
| Dependency | Extend <<Create Vessel Booking >>, Extend <<Modify Vessel Booking>>, Extend <<Delete Vessel Booking >> |
| Actors | Agent and Administrator |
| Preconditions | User should first be logged in, to the system. |
| Description of main sequences | 1. User should select ‘Vessels’. |
| Description of alternative sequences | 1. Users that are not logged in will not be able to access this part of the system. 2. Users are allowed to create, modify and delete schedules. |
| Post conditions | Bookings will be displayed, with other functions option for user to modify, create and delete. |

|  |  |
| --- | --- |
| Use case ID | 007 |
| Use case | View Booking |
| Summary | This use case allows the actor, to create a view booking. |
| Dependency | - |
| Actors | Agent and Administrator |
| Preconditions | User should first be logged into the system. |
| Description of main sequences | 1. User should select ‘View Booking’. |
| Description of alternative sequences | 1. If there are no Booking available, no record will be showed. |
| Post conditions | Booking displayed. |

### Sequence Diagram

#### Register



#### Login



#### View Schedule



#### Manage Schedule



#### Manage Vessel Booking

#### 

# Implementation

## Application Development

Maersk Container Line was developed through Microsoft Visual Studio 2017 IDE. The system was built with the help of MVC framework, which structures the program through modelling, views and controlling.

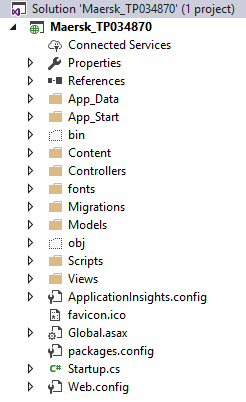


Figure 5.1.1: File Structure

The developed page and folders are arranged and presented as shown in figure (5.1.1). As seen, there are three folders among the list of it, and these 3, makes up MVC, which is Models, Views and Controllers. At normal time, a model will be first created, in the form of a class, where objects and variable are defined. Below, in figure (5.1.2) is a example of Vessel model in MVC.

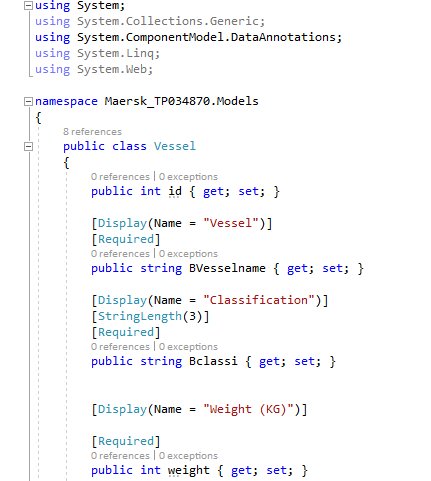


Figure 5.1.2: Model

After creating a model, in the package manager console, input “enable-migrations” to enable migrations and after complete, enter “add-migration ‘addVesselTable’” to create a table in the database for it. After connections are done, the controller file would be new added, scaffold-ly. Figure (5.1.3) is a example of a part of the Vessel controller, while the View file would also be generated at the same time.

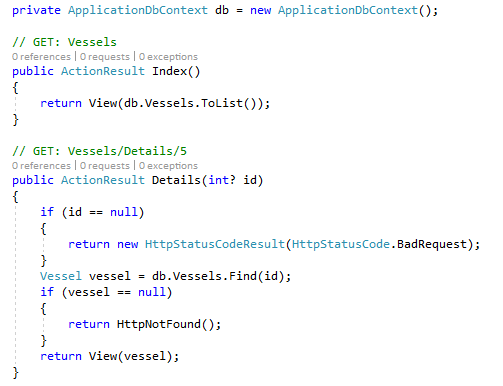


Figure 5.1.3: Controller

The end product is shown as in the figure below, (5.1.4). This could be found on: (https://github.com/carmennnnnl96/DDAC\_TP034870/).

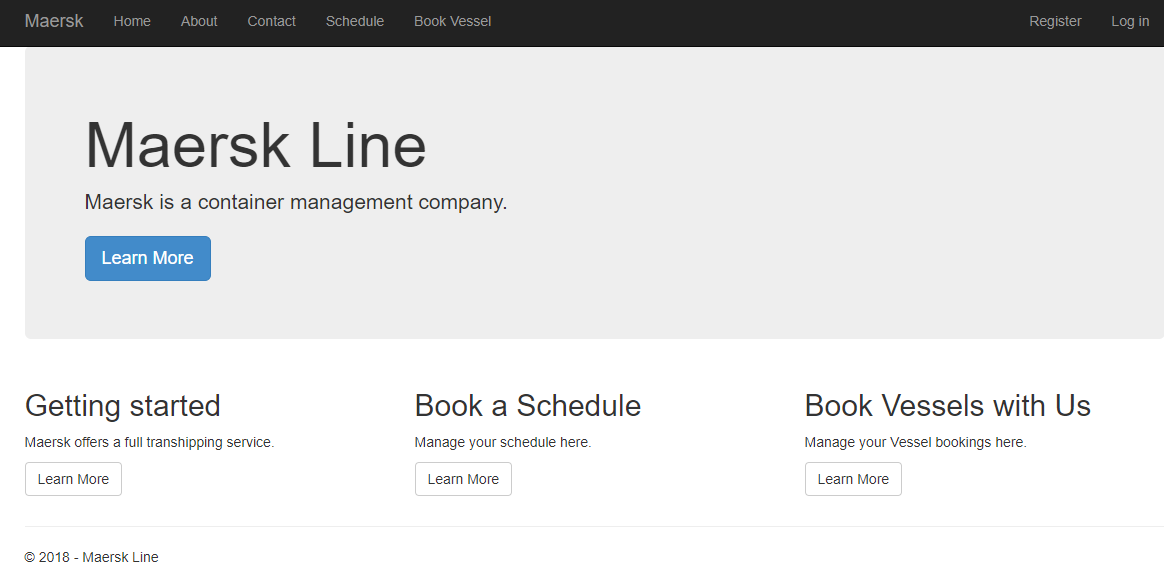


Figure 5.1.4: Homepage

## Azure Publishing

There are a few steps to go thorough, before realizing the publish of the web application to cloud. The process starts off with:

1. Right clicking the file, in the solution explorer, and select publish.

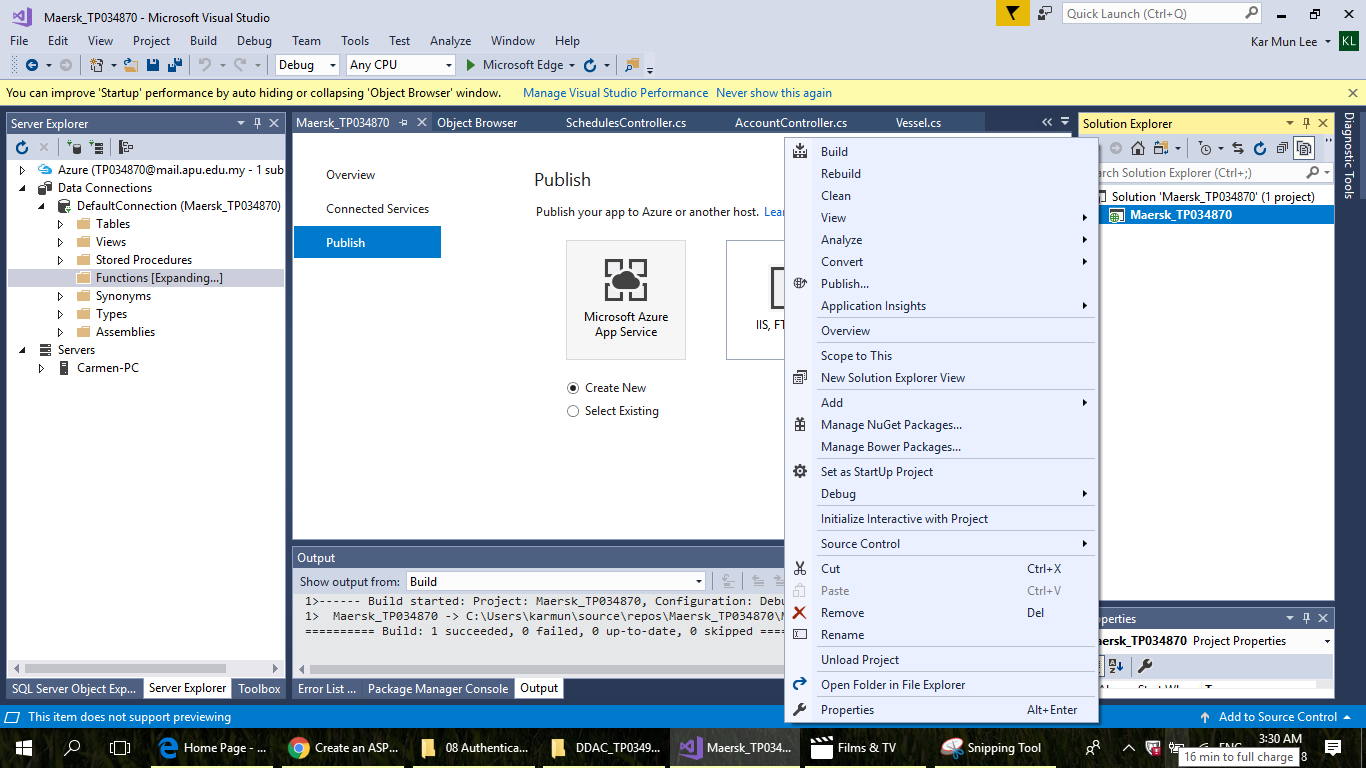


Figure 5.2.1: Step 1

1. Then, select the Microsoft Azure app Service, and Publish.

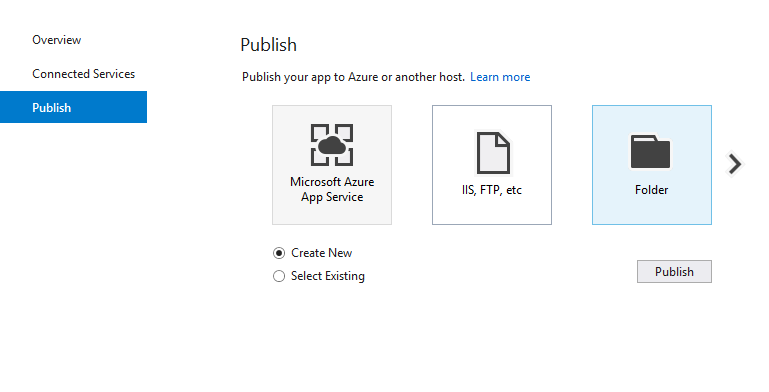


Figure 5.2.2: Step 2

1. Next, prefix the unique names for the following field and sign in at the right top corner with the account that is desired.

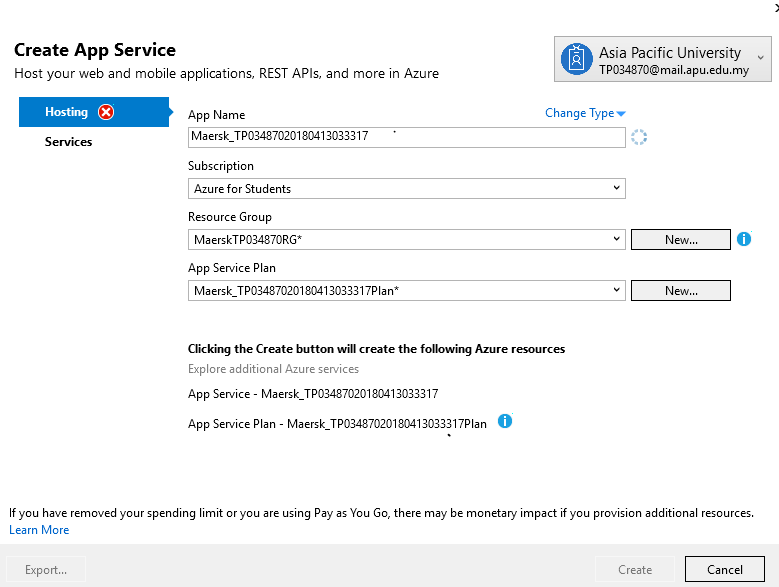


Figure 5.2.3: Step 3

1. Select create, after done. Then select publish.

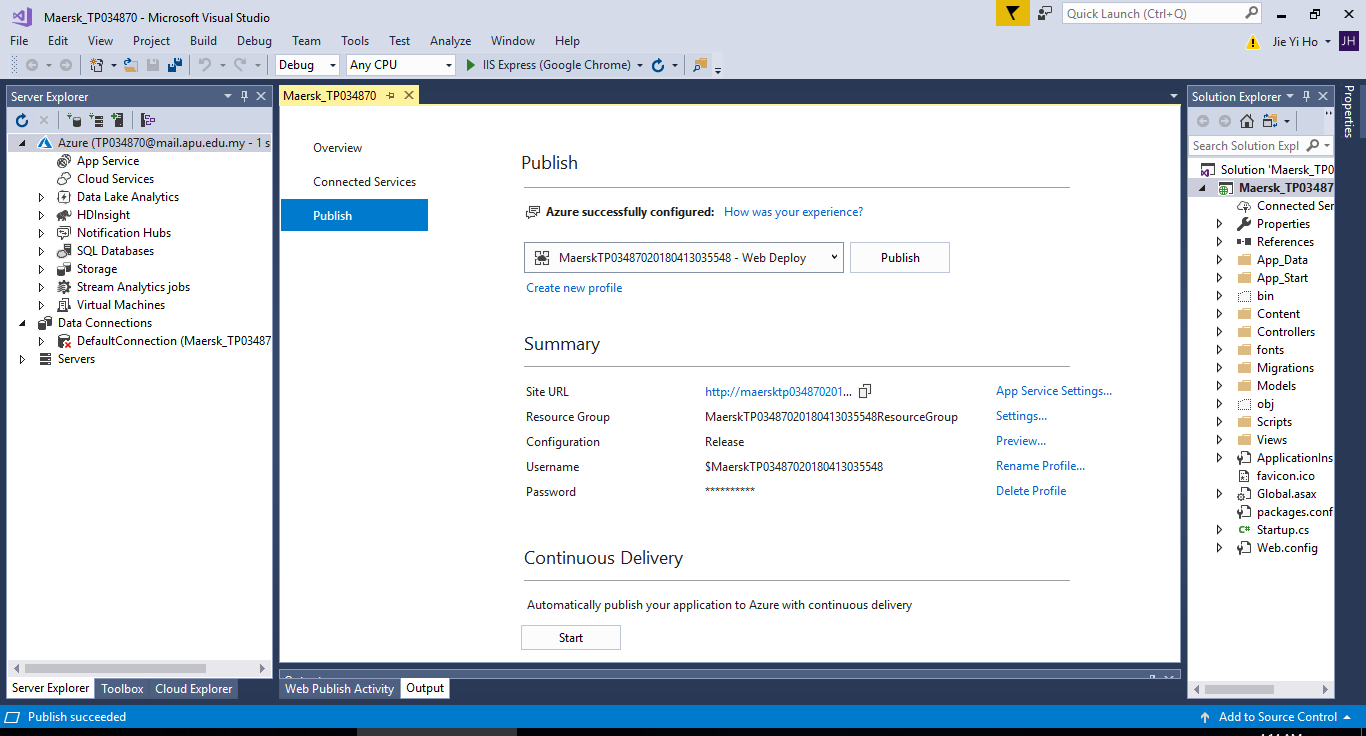


Figure 5.2.4: Step 4

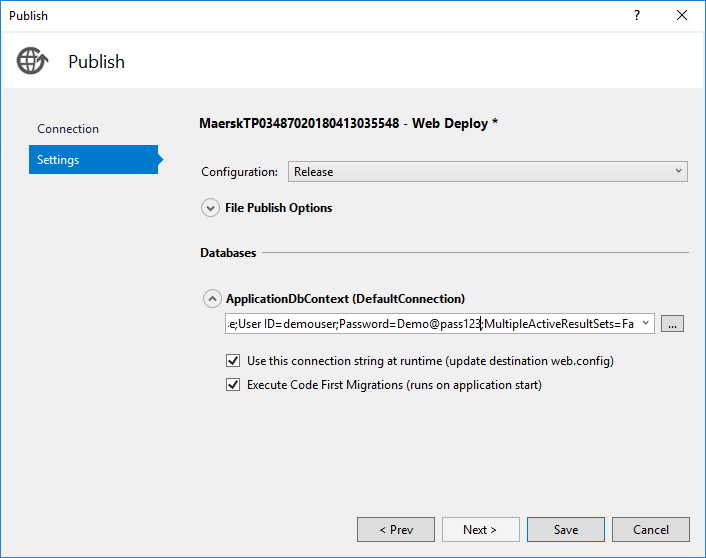


Figure 5.2.5: Step 5

1. After done with all naming, configuring, the web app is set to be deploy.

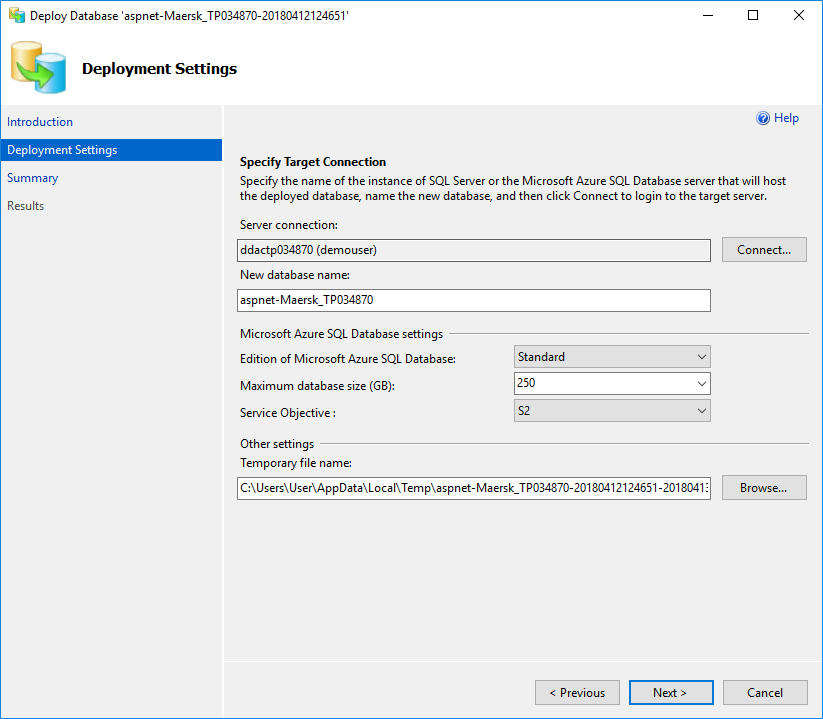


Figure 5.2.6: Step 6

1. The final step here is to connect the database, and bring it up to Azure’s database so it is on cloud.

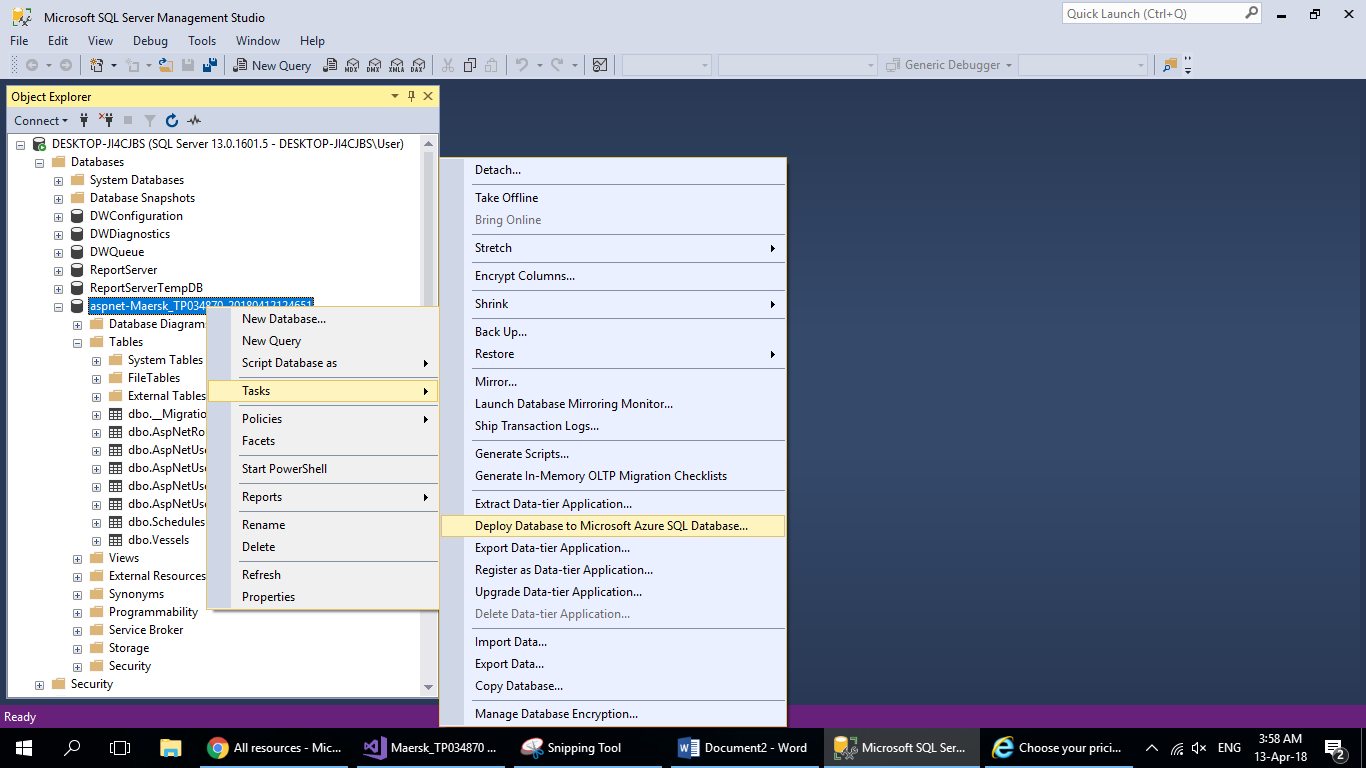


Figure 5.2.7: Step 7

## Application Scaling

As for now, Maersk Container Management system is on Standard 1. According to the requirements, S2 standard looks fine with meeting it however, budget wise, S1 tier is actually below the given, which actually offers a better price for what was plan. This offers a bigger quota for spare, in case of exceeding the usage. Although, Maersk has a wide relation of business worldwide, where 1.75GB RAM would be too intimidating to support the high traffic, however, as the application has just launched, it would be more than enough to support current users. In the future, S3 would be taken into consideration, as an upgrade, as it supports more, which meets Maersk’s business nature.

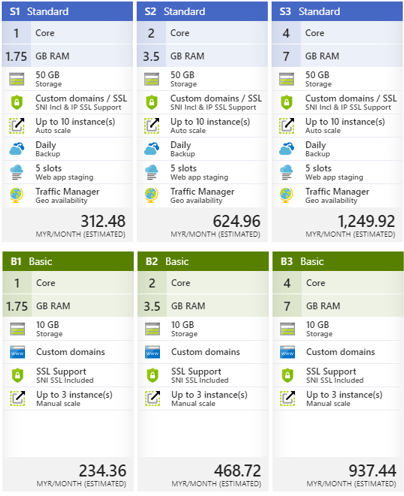


Figure 5.3.1: Azure Price Tier

Web applications these days are more to standard plans, however, it is not by mean of any trend or stuff, but because of the service it could offer. Talking about storage, standard plans offer a storage of 50GB. However, in figure (5.3.1), it is clearly shown, at the same tier, tier 1, with around RM80 differences, there is 40 GB more, in terms of storage. Comparing the service provided, it is clearly shown, even by that little increase on budget, there is more to offer. Looking at the Traffic Manager, which is not available for the Basic plans, geo availability is offered. Geo availability allows user to build their application at just anywhere, any region that Azure covers, which is just basically any place (Foulds, et al., 2018). Domain wise, Standard plan offers an additional IP SSL Support, apart from the SNI Incl. Automatic could never be said is better than manual, but is definitely an easier way to perform something. Same situation here, with the scaling function, Standard plan offers 7 instance more than Basic plan, with a base of 3 instance, however, it is not just that. Standard plan offers auto scaling where users would not need to scale in and out themselves, especially on non-peak and peak hours.

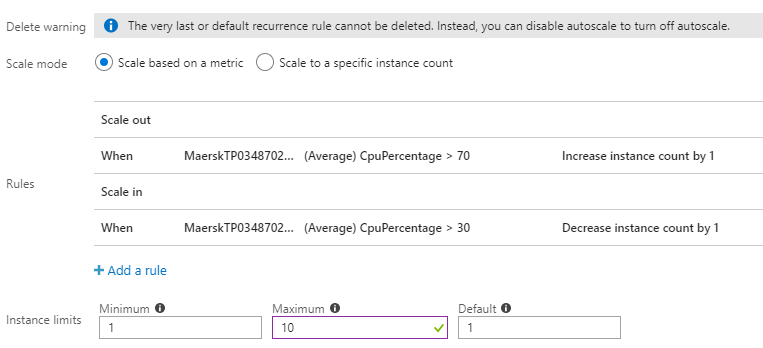


Figure 5.3.2: Azure Scale

Figure shows a self prefixed standard on scales, where if CPU utilization is more than 70%, an instance will be added, at the same time, if it is less than 30%, an instance will drop.

## Reliability and Performance

As mentioned earlier, the web application will be deployed to several regions, due to its business nature, therefore, high reliability is definitely not much of a concern, as the application will be available in a few different servers, in a few different regions. Just in case if any one of the server is down, the back up in the other servers could still assure the application cold still run. Performance wise, a performance test will be conducted to monitor the progress. The performance test could be brought out in Azure itself.

# Testing

## Unit Testing

### Login

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Testcase ID** | **Page** | **Test Function** | **Description** | **Expected Result** | **Actual Result** | **Status** | **Priority** |
| L1 | Login | Login | - Enter correct username  - Enter correct password  **[email:** [**g@g.com**](mailto:g@g.com)  **Password:**  **g@g.comG1]** | Login successful | Login success | Pass | High |
| L2 | Login | Login | - Either one or both data does not match the one in database | Return “Username or password not found. Please try again or register to proceed. ” | “Username or password not found. Please try again or register to proceed. ” | Pass | High |

### Registration

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Testcase ID** | **Page** | **Test Function** | **Description** | **Expected Result** | **Actual Result** | **Status** | **Priority** |
| R1 | Registration | Register button | - All fields filled, with data meeting the data validations | Register successful;  Return Login Page | Register successful;  Return Login Page | Pass | High |
| R2 | Registration | Register button // Password field | - The password field is tested to see if validations were presented | An error message due to data validation should be prompted. | System: “Passwords must have at least one non letter or digit character. Passwords must have at least one digit ('0'-'9'). Passwords must have at least one uppercase ('A'-'Z').” | Pass | Low |
| R3 | Registration | Register button//  Email validation | - Email meets the email standard | An error message due to data validation should be prompted. | System: “The Email field is not a valid e-mail address.” | Pass | Low |

### Schedule

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Testcase ID** | **Page** | **Test Function** | **Description** | **Expected Result** | **Actual Result** | **Status** | **Priority** |
| CS1 | Create Schedule | Create | - Users shall be able to make a schedule | - schedule created | schedule created | Pass | High |
| MS1 | Modify Schedule | Edit button | - Data shall edit the record. | Modifying Successful | Edited | Pass | High |
| D1 | Delete Schedule | Delete button | - Shall direct back to Login page | Record deleted! | Directed to Login page | Pass | Low |

### Vessel Booking

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Testcase ID** | **Page** | **Test Function** | **Description** | **Expected Result** | **Actual Result** | **Status** | **Priority** |
| CS2 | Create Vessel Booking | Create Vessel Booking | - Users shall be able to make a Vessel Booking | - Vessel Booking created | Vessel Booking created | Pass | High |
| MS2 | Modify Vessel Booking | Edit button | - Data shall edit the record. | Modifying Successful | Edited | Pass | High |
| D2 | Delete Vessel Booking | Delete button | - Shall direct back to Login page | Record deleted! | Directed to Login page | Pass | Low |

## Performance

In Azure, performance test is availble and provided, in the cloud service. In figure (), the preformance test was carried out to measure the peformance of the web application. The test is conducted on the Southeast Asia region, with 250 users, within a duration of 5 minutes. The test is performed on Standard 1 plan. The test results are as shown.

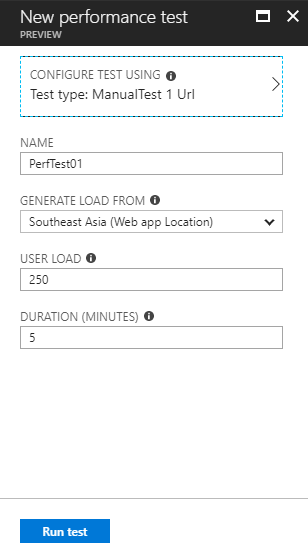


Figure 6.1: Performance Test

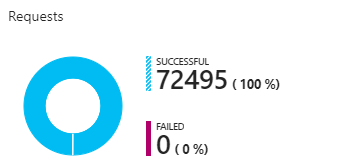


Figure 6.2: Failure Rate

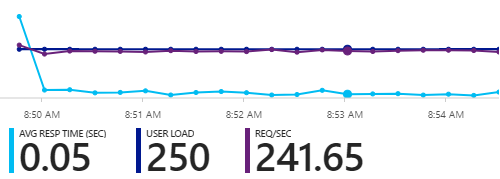


Figure 6.3: Average Response Rate

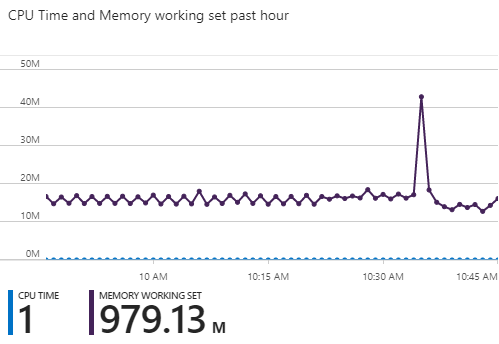


Figure 6.4: CPU Utilization

The test, is then conducted a few more times, and the results are in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Users | 250 | 500 | 750 | 1000 | 2000 |
| Service Plan |
| Standard S1 | Avg Resp | failing rate: 0.05 | 0% | Avg Resp | failing rate:  5.76 | 0% | Avg Resp | failing rate:  5.2 | 0% | Avg Resp | failing rate:  5.53 | 0% | Avg Resp | failing rate:  4.99 | 0% |

## Analysis

Unit Testing is conducted to test units of codes, to whether it is functioning well, and if it capable to handle exceptions. For instance, to test aa login function, user’s input needs to match the data in the database, however what if it does not match, and what are the handling methods that will be adopted to take care of the situation is very important. The most common situation could be email or passwords. Taking email as an example, email has a general standard of [username@emailserviceprovider.com](mailto:username@emailserviceprovider.com)*.* Questions throw in, with what if, there’s a symbol or signs like dashes, brackets and so forth, what if there’s no ‘@’, what if there’s numbers and to answer all these what if(s), data validation is very important to handle these exceptions, where unit testing, checks if they could handle it. Unit testing is important, to assure the system runs smoothly, handles exceptions without deadly crashing.

Performance Test is important to see how fast the system can respond upon a request. In this situation, Azure has already provided the feature, to test the quality attribute. According to the results, in the previous section, failing rate was 0, despite the increase of users in every test, while the average response time shows consistency in most of the tests, except the one with 250 users, with an unexceptional good result. This shows that the current plan is a reliable one, where if there isn’t really anything big happening, or major changes, this could last a long time.

# Managed Database

Platform as Service (PaaS), the cloud’s development platform (Microsoft Azure, 2018). PaaS provides a complete less complicated environment, where users can perform cloud computing easier. The service works as pay-as-you-go where users can perform task of a secure net. PaaS basically provides everything a developer needs, from development tools, database, network and so forth (Microsoft Azure, 2018).

The Azure SQL Database is a cloud based database, works finely like any local database, but on the cloud (Microsoft Azure, 2018). Azure SQL Database allows users to perform migration with SQL server, without modifying the application, through SQL Database Managed Instance. According to (Microsoft Azure, 2018), Managed Instance is best for database migrations, where users can benefit from it, operational and financial wise.



Figure 7.1: SQL Database

Apart from that, SQL Database offers a built-in intelligence that captures patterns and exploits the captured information, in a good way on improving the performance and security. In the security aspect, the cloud database service offers a built-in protection and features, that encrypts the data when it is at rest and in motion (Microsoft Azure, 2018). In addition, the service duplicates 3 versions of the application, to support a 99.99% of availability. The service rules guides and standards within itself, there meeting standards like ISO/IEC 7001/27002, PCI DSS and so forth should not be a problem.

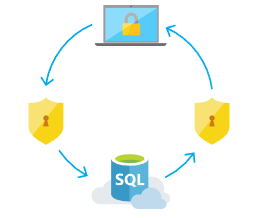
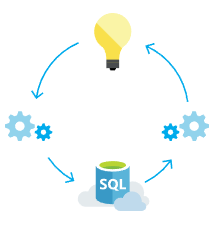


Figure 7.2: SQL Database

For all these reasons itself, it is already much attracting, for it has to offer. For Maersk, a Standard level plan is chose to support back the application.

# Conclusion

Through developing to wrapping up this project, there was a lot of knowledge gained, both theory wise and practical wise. Theoretical wise, the developer was able to look into a deeper in depth of the cloud computing technology. The advantages of cloud computing was more than what it looks, however, to really know about the technology, how it benefits people, practically implementing would improve the experience. From designing the web application to deploying it, experiences and knowledge were gained. This was mostly experienced through designing the web application, realizing it and then pushing it to cloud, where in Maersk Line Container Management System case, it was deployed to Microsoft Azure was used. With the cloud service going strong in this century, it is believed, that acquiring cloud computing knowledge is important in this era.

# References

Foulds, I., Nottingham, C. & Squillace, R., 2018. *Regions and availability for virtual machines in Azure.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/regions-and-availability  
[Accessed 1 April 2018].

Microsoft Azure, 2018. *SQL Database.* [Online]   
Available at: https://azure.microsoft.com/en-us/services/sql-database/  
[Accessed 1 April 2018].

Microsoft Azure, 2018. *What is Paas?.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/  
[Accessed 1 April 2018].

Appendix

GitHub: https://github.com/carmennnnnl96/DDAC\_TP034870/).

Microsoft Stream: <https://web.microsoftstream.com/video/87d98a01-4955-48a1-88c9-dbea4f9d773d>

Website: <https://maersktp03487020180413035548.azurewebsites.net/>