****

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT071-3-3-DDAC**

**Designing and Developing Applications on the Cloud**

**UC3F1706SE**

**Name: Yeo Kai Wen**

**Student ID: TP038173**

**Lecturer: DR. KALAI ANAND A/L RATNAM**

**ASSIGNMENT HANG IN DATE : 13th April 2018**

**Acknowledgement**

First of all, I would like to thank Dr. Kalai for the guidance and expertise in cloud computing, it helps when come to publishing the project along with the documentation.Next, I would like to thank all my classmate for guiding me whenever I need their help, and last, I would like to thank my parents for supporting and allowing me to study in APU.

Table of Contents

[1.0 Introduction 1](#_Toc511376016)

[1.1 Objectives 1](#_Toc511376017)

[1.2 Scopes 2](#_Toc511376018)

[2.0 Project Plan 1](#_Toc511376019)

[3.0 Design 2](#_Toc511376020)

[3.1 Modelling 2](#_Toc511376021)

[3.1.1 Use Case Diagram 2](#_Toc511376022)

[3.1.2 Use Case Description 4](#_Toc511376023)

[3.1.3 Entity Relationship Diagram 11](#_Toc511376024)

[3.1.4 Sequence diagram 12](#_Toc511376025)

[3.1.5 Cloud Architecture 20](#_Toc511376026)

[4.0 Implementation 21](#_Toc511376027)

[4.1 Application development 21](#_Toc511376028)

[4.2 Azure Publishing 23](#_Toc511376029)

[4.3 Application Scaling 27](#_Toc511376030)

[4.3.1 Reliability & Performance 29](#_Toc511376031)

[5.0 Test Plan & Testing Discussion 31](#_Toc511376032)

[5.1 Unit Testing 31](#_Toc511376033)

[5.1.1 Login Function Test Plan 31](#_Toc511376034)

[5.1.2 Logout Function Test Plan 31](#_Toc511376035)

[5.1.3 View Schedule Function Test Plan 32](#_Toc511376036)

[5.1.4 Register New Agent Function Test Plan 32](#_Toc511376037)

[5.1.5 Define New Schedule Function Test Plan 32](#_Toc511376038)

[5.1.6 Accept Agent Booking Function Test Plan 33](#_Toc511376039)

[5.1.7 Book Schedule Function Test Plan 33](#_Toc511376040)

[5.1.8 Register Customer Function Test Plan 33](#_Toc511376041)

[5.2 Performance 35](#_Toc511376042)

[5.3 Implementation & Discussion of Managed Databases 38](#_Toc511376043)

[6.0 Conclusion 41](#_Toc511376044)

[References 42](#_Toc511376045)

# 1.0 Introduction

Maersk Line is the global container division and the largest operating unit of the A.P. Moller – Maersk Group, a Danish business conglomerate. It is the world’s largest container shipping company having customers through 374 offices in 116 countries. It employs approximately 7,000 sea farers and approximately 25,000 land-based people. Maersk Line operates over 600 vessels and has a capacity of 2.6 million TEU. The company was founded in 1928.

Operating in 100 countries and transporting goods around the globe, at first glance it would appear Danish shipping company Maersk Line is already handling all the cargo it can manage. But when Maersk determined that the volume of most of the goods it was shipping had grown to full capacity, the company decided that cloud powered solutions would be a crucial part of rectifying the situation.

“There was a ‘mind-opener’ where Maersk said, ‘How can we support the overall business strategy, and also from an IT perspective,” says Soeren Lorenzen, an account general manager with Hewlett-Packard company who is involved first-hand with Maersk’s ITO efforts. “There was a new CIO who wanted to outsource every part of IT, but without [negatively] impacting shipping.”

In an effort to support further business growth and increase organizational flexibility, Maersk decided to consolidate all of its data centers and server rooms operating worldwide onto a virtualized platform. Microsoft Azure was already hosting some of Maersk’s IT environment, and in March 2016 Maersk initially approached Microsoft about expanding the scope of the relationship. Moving forward, Lorenzen says Maersk is currently changing over its IT setup based on Microsoft Azure, starting with the desktop environment up to container management.

## 1.1 Objectives

* To develop web application to support further business growth and increase organizational flexibility.
* Able to cater and manage the containers.
* Able to reduce overall supply chain costs.
* Provide an efficient way to manage logistics.

## 1.2 Scopes

* Able to manage import, export, transhipment processing, and gate operations.
* Able to scale the system based on 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.
* Provide 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 entire booking process from schedule search to booking confirmation.

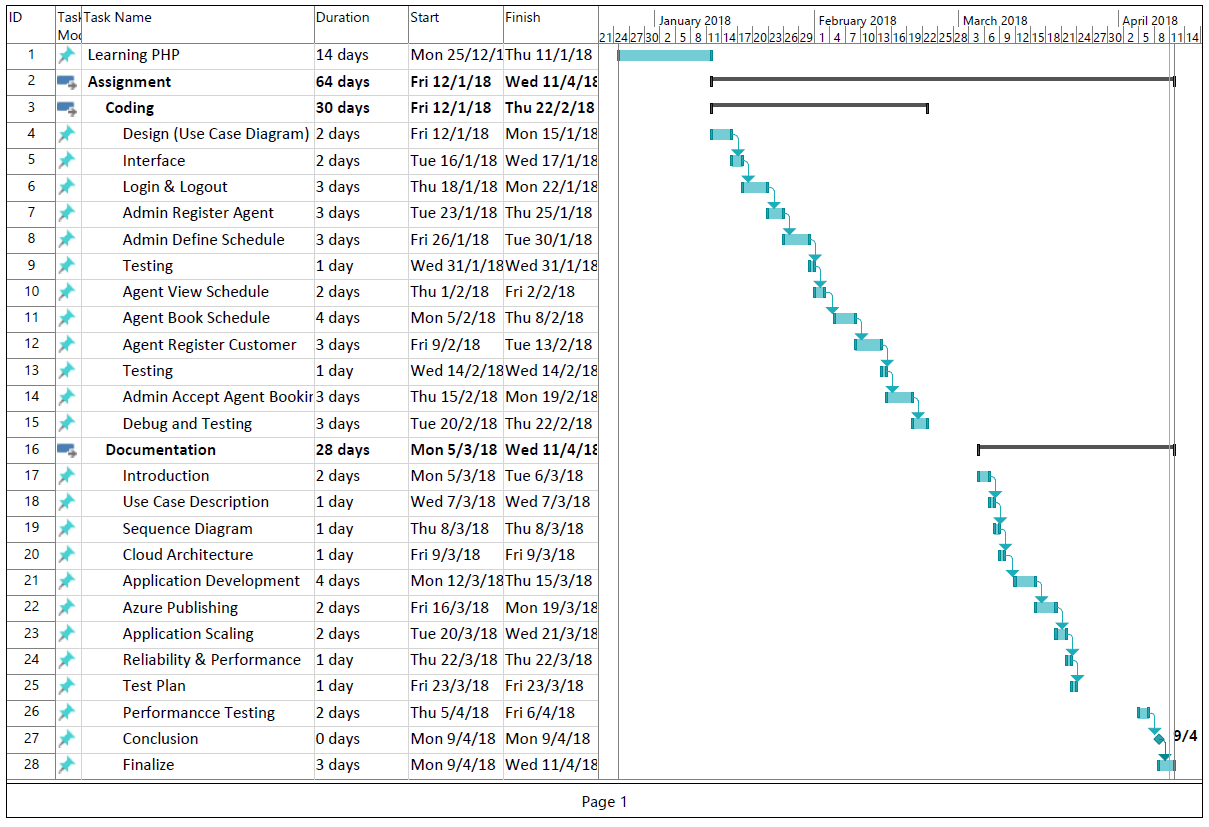
**Administrator functionalities**

* + Manage agent accounts.
  + Manage ship details.
  + Manage shipment schedules.
  + Manage shipment schedule booking requests.

**Agent functionalities**

* + Manage customer records.
  + Manage customer’s item details.
  + View booking done by the agent.
  + View available shipment schedule for booking.
  + Book shipment schedule for customer’s item.

# 2.0 Project Plan



# 3.0 Design

## 3.1 Modelling

### 3.1.1 Use Case Diagram

**Admin module**

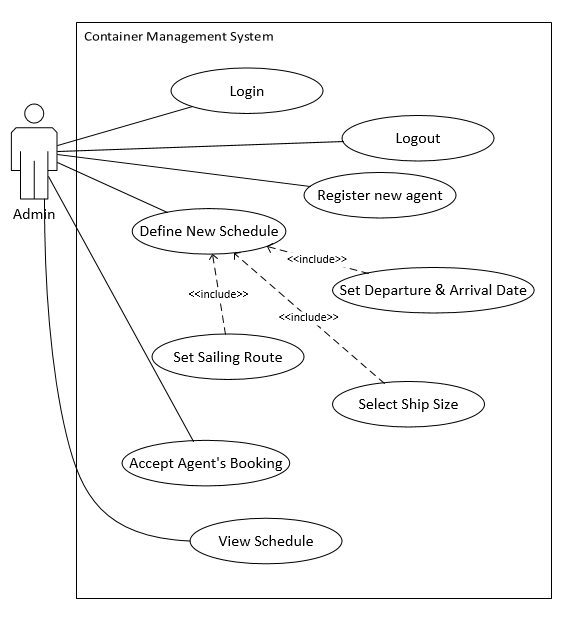


Figure 1 Admin module use case diagram

**Agent module**

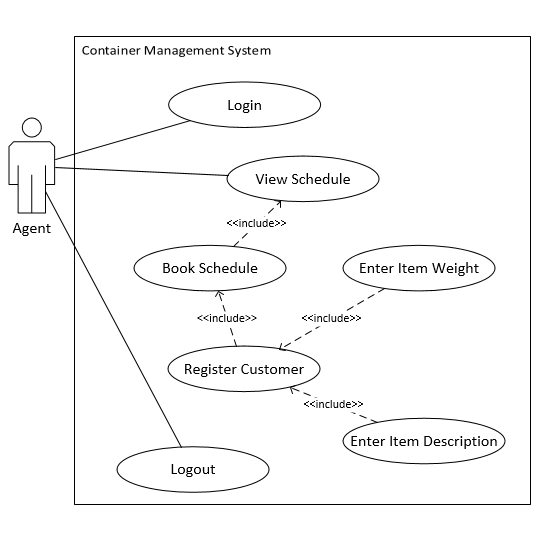


Figure 2 Agent module use case diagram

### 3.1.2 Use Case Description

|  |  |
| --- | --- |
| Use Case: | Login |
| Summary: | An admin or agent enter their unique username and password in order to access more function of the system. |
| Dependency: | - |
| Actors: | Admin and Agent |
| Pre-condition: | Actor must be an admin or agent. |
| Main Flow: | 1. Actor enter their username and password. 2. System will search through database whether the username and password are existed and matched. 3. Actor may enter the website if they entered the correct username and password. |
| Alternative Flow: | **Step 2**: System could not be login and ask for the actor to enter the correct username and password. |
| Post Condition: | Actor has login into the system. |

|  |  |
| --- | --- |
| Use Case: | Logout |
| Summary: | When a admin or agent has done their task or done using the system, they have to logout to avoid other people use to their account. |
| Dependency: | - |
| Actors: | Admin or Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Actor finish using with the system. 3. Actor click the logout button in the system. 4. System will show the page before the actor login to their account if logout successfully. |
| Alternative Flow: | **-** |
| Post Condition: | The actor has logout. |

|  |  |
| --- | --- |
| Use Case: | View schedule |
| Summary: | Actor can use the function to view the available schedule. |
| Dependency: | - |
| Actors: | Admin or Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Actor select the view schedule button. 3. The available schedule showed. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The available schedule showed. |

|  |  |
| --- | --- |
| Use Case: | Register new agent |
| Summary: | Admin can create an account for agent if agent request to. |
| Dependency: | - |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. The form is showed in the front page. 3. Fill in the required information. 4. Submit the form. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The agent account has been created. |

|  |  |
| --- | --- |
| Use Case: | Define new schedule |
| Summary: | Admin can create new schedule for agent to book it which include departure and arrival date, ship size and sailing route. |
| Dependency: | <<include>> Set departure & arrival date  <<include>> Set sailing route  <<include>> Set ship size |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select define new schedule. 3. Fill in the required information. 4. Submit the form. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | New schedule has been created. |

|  |  |
| --- | --- |
| Use Case: | Set departure and arrival date |
| Summary: | While admin creating a new schedule, admin can enter the departure and arrival date for the new schedule. |
| Dependency: | <<include>> Define new schedule |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select define new schedule. 3. Fill in the departure and arrival date of the new schedule. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The date has been set. |

|  |  |
| --- | --- |
| Use Case: | Select ship size |
| Summary: | While admin creating a new schedule, admin can select the ship size for the schedule |
| Dependency: | <<include>> Define new schedule |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select define new schedule. 3. Select the ship size for the schedule. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The ship size has been set. |

|  |  |
| --- | --- |
| Use Case: | Set sailing route |
| Summary: | While admin creating a new schedule, admin can enter the sailing route for the schedule. |
| Dependency: | <<include>> Define new schedule |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select define new schedule. 3. Fill in the sailing route for the schedule. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The sailing route has been set. |

|  |  |
| --- | --- |
| Use Case: | Accept agent’s booking |
| Summary: | When an agent books the schedule, the admin has to accept the booking to allow agent to book it. |
| Dependency: | - |
| Actors: | Admin |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select accept agent booking button at the home page. 3. Click on ‘Accept’ button on which the admin want to accept the booking. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The agent’s booking has been accepted. |

|  |  |
| --- | --- |
| Use Case: | Book schedule |
| Summary: | Agent can book the schedule and register the customer and the item for the schedule. |
| Dependency: | <<include>> View schedule  <<include>> Register Customer |
| Actors: | Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select view schedule function. 3. Select the desired available schedule. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The schedule has been selected. |

|  |  |
| --- | --- |
| Use Case: | Register customer |
| Summary: | After select the schedule, agent can enter the customer information in the form. |
| Dependency: | <<include>> Book schedule  <<include>> Set item weight  <<include>> Enter item description |
| Actors: | Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select view schedule function. 3. Select the desired available schedule. 4. Enter all the required information. 5. Submit the form. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The customer has been registered for booking. |

|  |  |
| --- | --- |
| Use Case: | Enter item weight |
| Summary: | After select the schedule, agent can enter the customer information and the item’s weight. |
| Dependency: | <<include>> Register Customer |
| Actors: | Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select view schedule function. 3. Select the desired available schedule. 4. Enter the item weight for the booking. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The item weight has been filled. |

|  |  |
| --- | --- |
| Use Case: | Enter item description |
| Summary: | After select the schedule, agent can enter the customer information and the item’s description. |
| Dependency: | <<include>> Register Customer |
| Actors: | Agent |
| Pre-condition: | The actor has to login their account. |
| Main Flow: | 1. Actor login to their account. 2. Select view schedule function. 3. Select the desired available schedule. 4. Enter the item description for the booking. |
| Alternative Flow: | **Step 1**: Login failed. |
| Post Condition: | The item description has been filled. |

### 3.1.3 Entity Relationship Diagram

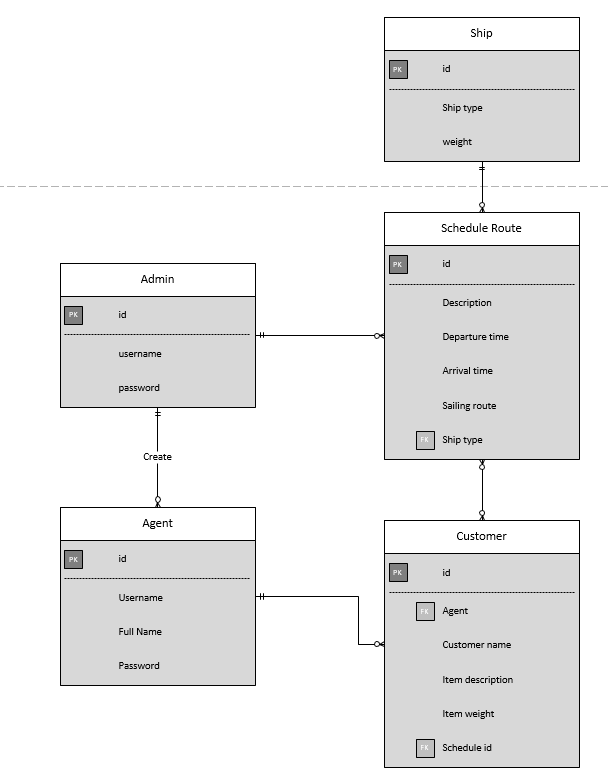


Figure 3 ERD

### 3.1.4 Sequence diagram

**Login sequence diagram**

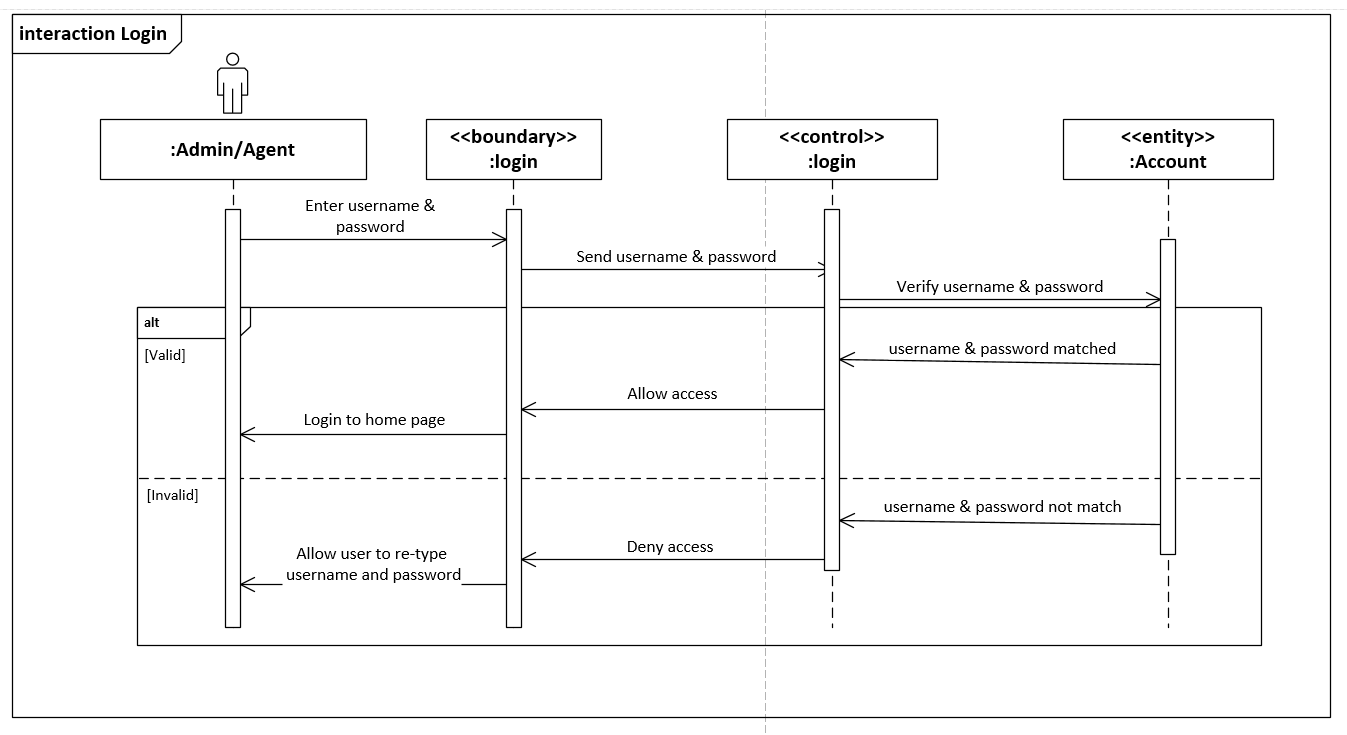


Figure 4 Login sequence diagram

**Logout sequence diagram**

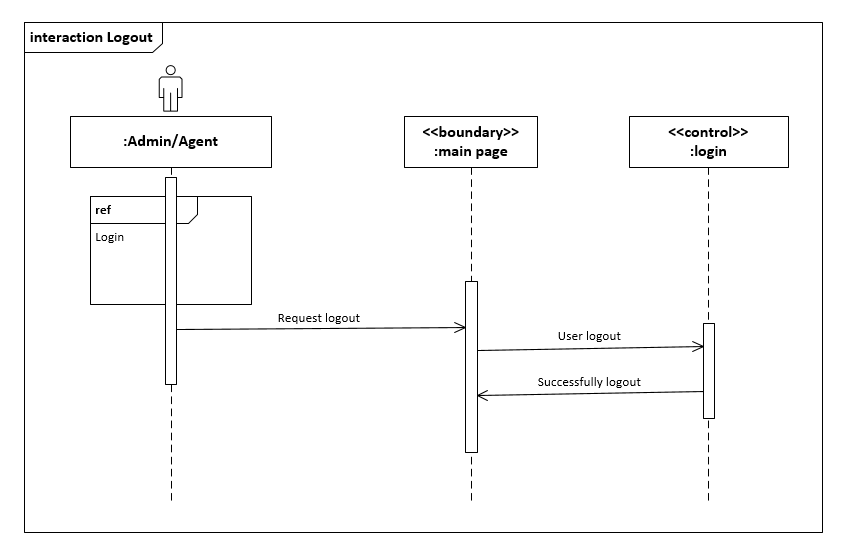


Figure 5 Logout sequence diagram

**View schedule sequence diagram**

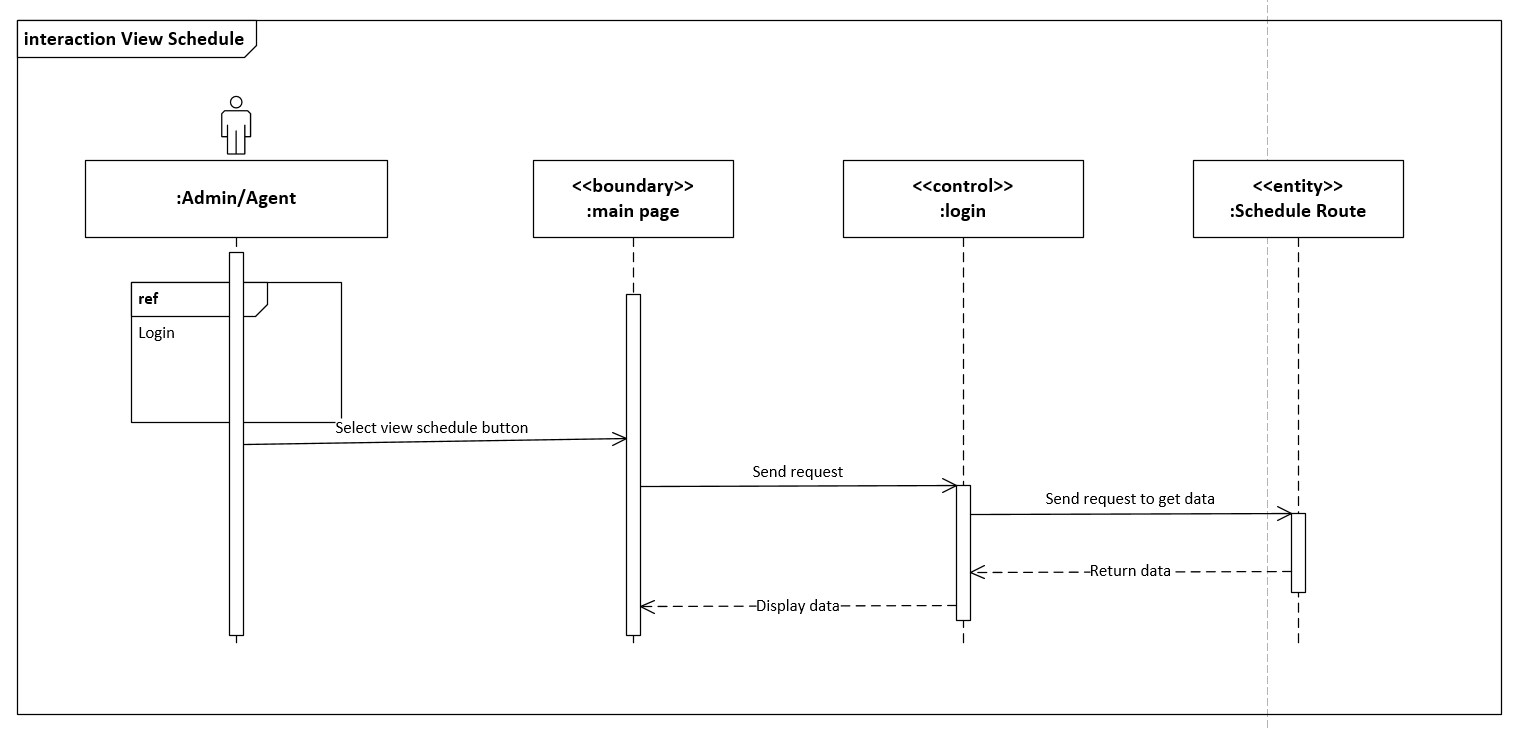


Figure 6 View schedule sequence diagram

**Admin Register new agent sequence diagram**

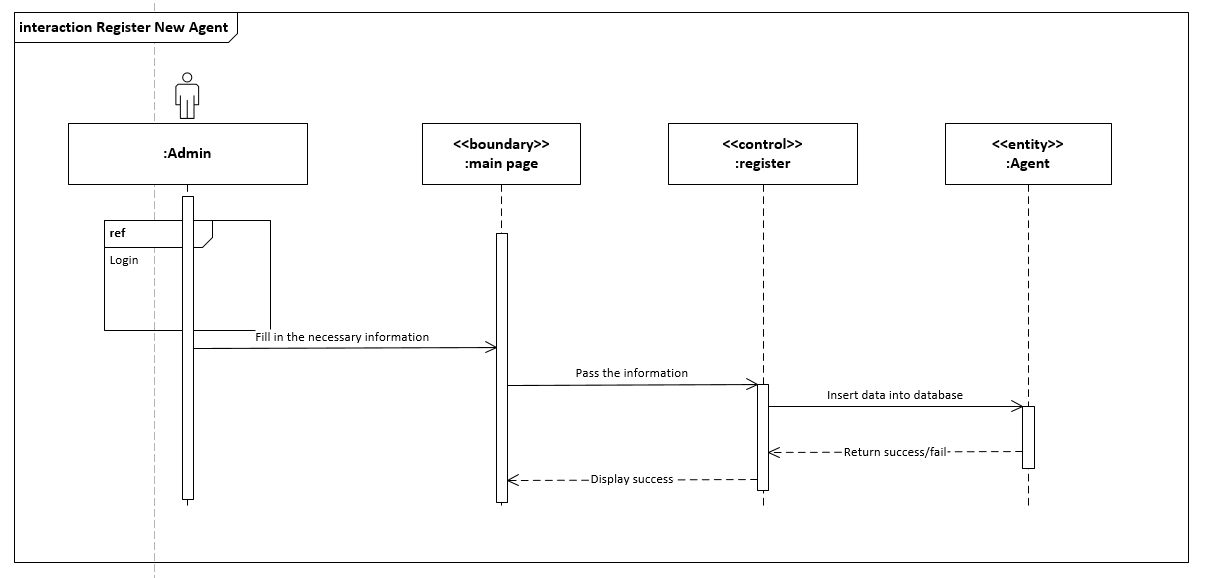


Figure 7 Admin Register new agent sequence diagram

**Admin Define new schedule sequence diagram**

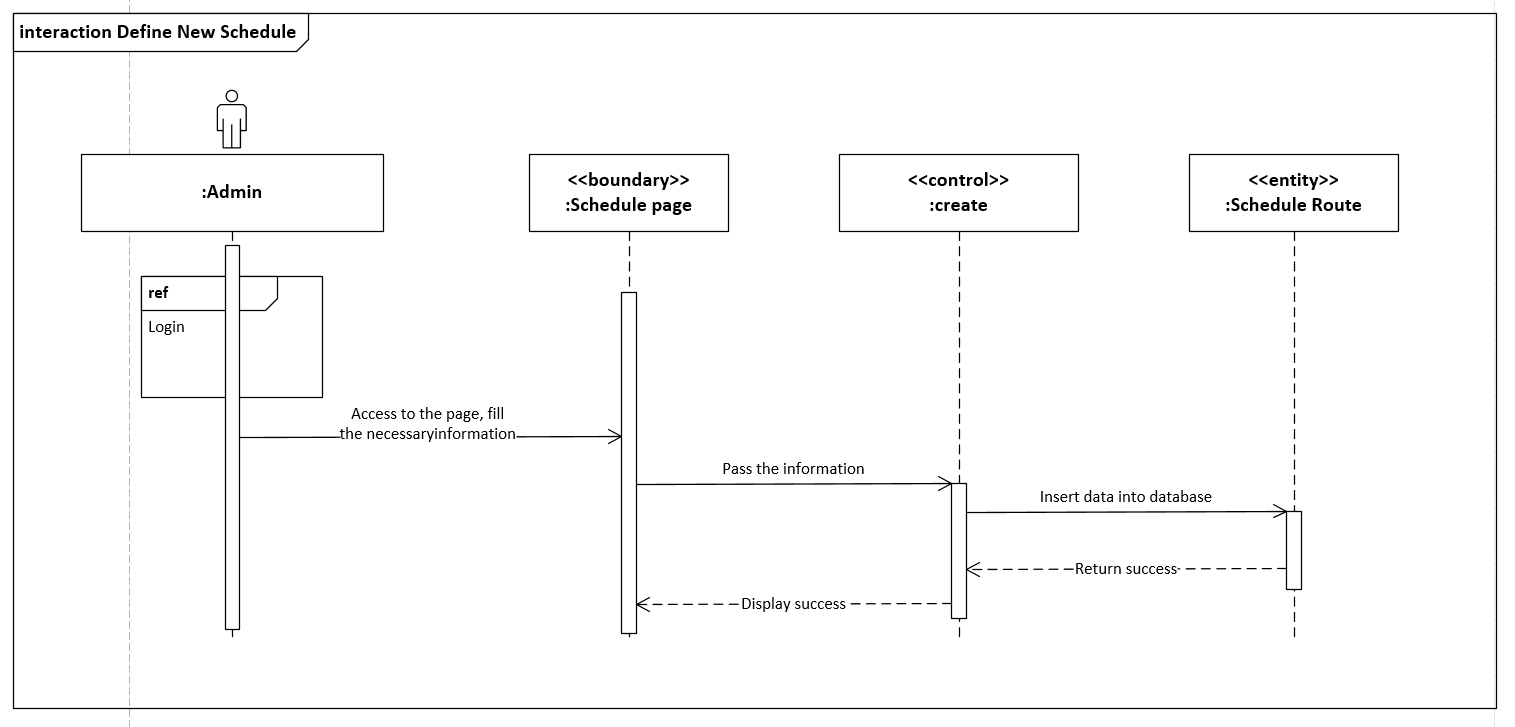


Figure 8 Admin Define new schedule sequence diagram

**Admin accept agent booking sequence diagram**

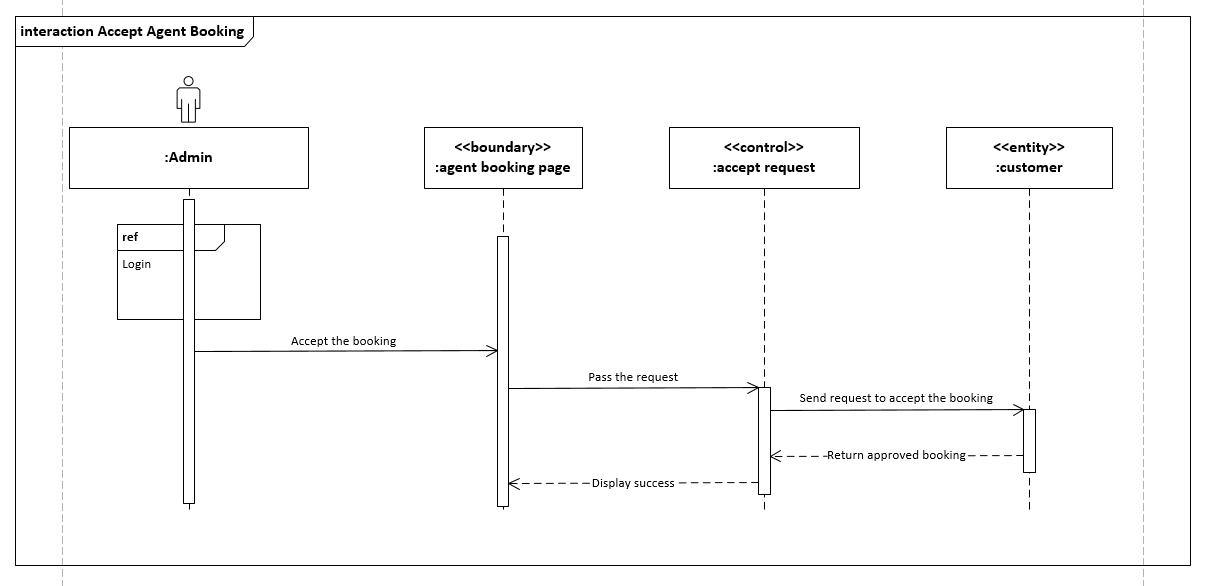


Figure 9 Admin accept agent booking sequence diagram

**Agent book schedule sequence diagram**

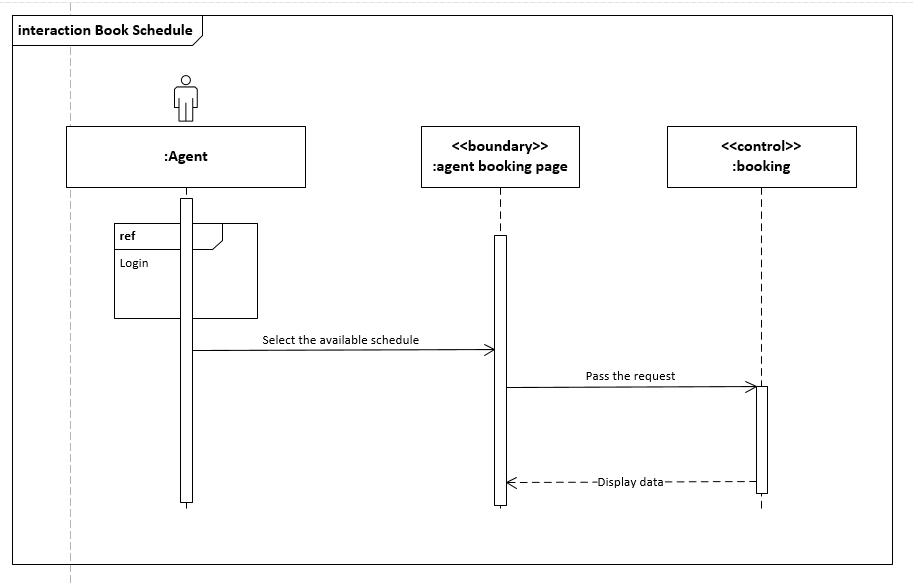


Figure 10 Agent book schedule sequence diagram

**Agent register customer sequence diagram**

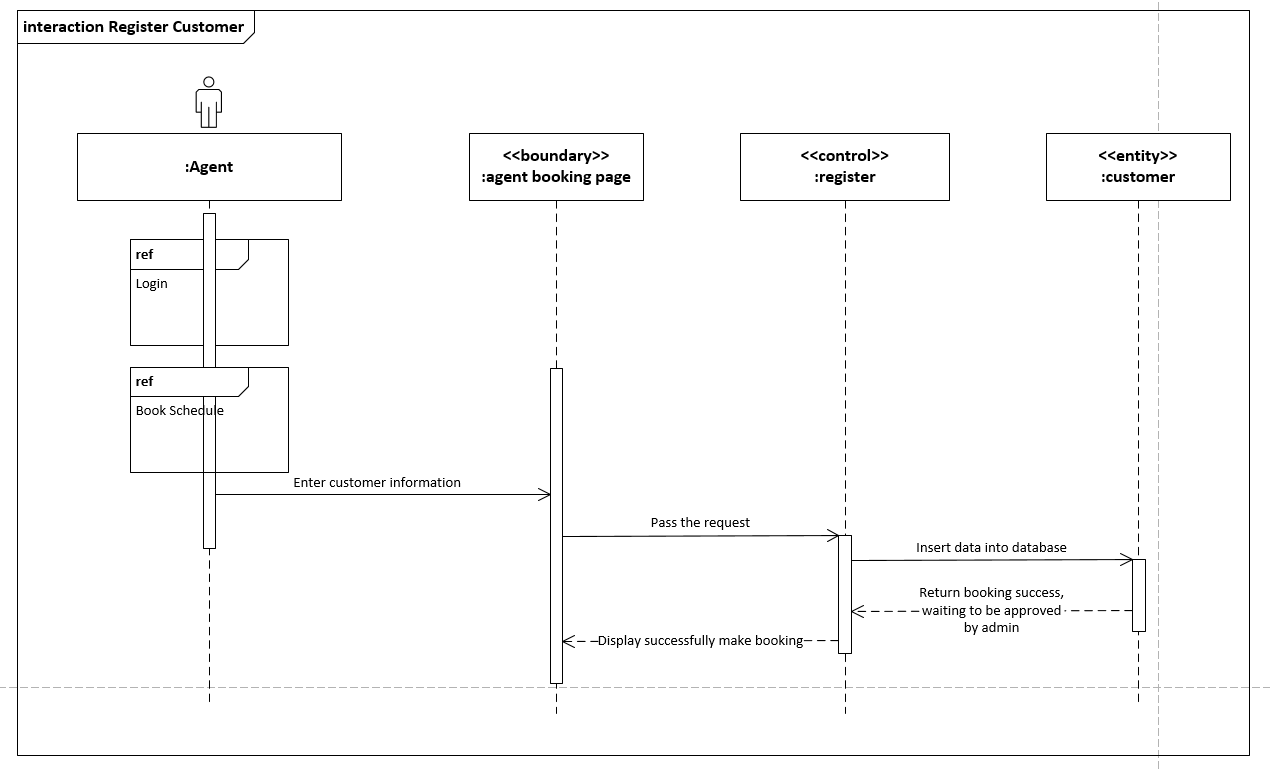


Figure 11 Agent register customer sequence diagram

### 3.1.5 Cloud Architecture

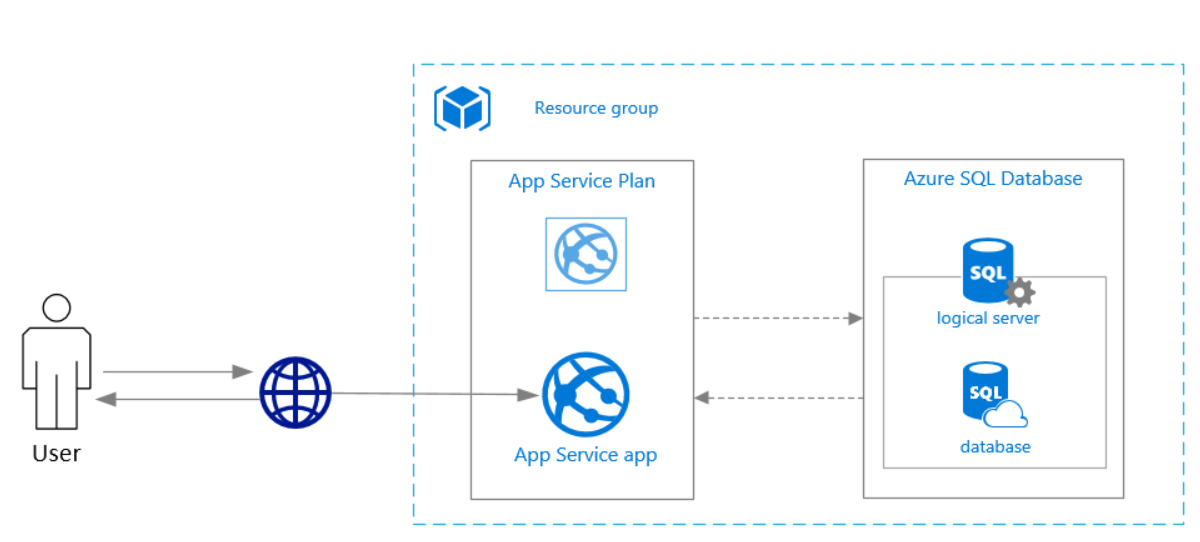


Figure 12 Cloud architecture diagram

The total cost estimation for implementing the above architecture shown as the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Service type** | **Region** | **Description** | **Estimated Cost** |
| App Service | Southeast Asia | 1 instance(s) x 1 Months, Size: S1, Standard tier, 0 SNI connection(s), 0 IP connection(s) | RM306.60 |
| Azure Database for MySQL | Southeast Asia | General Purpose Tier, 1 Gen 4 (2 vCore) x 1 Months, 105 GB Storage, 100 GB Additional Backup storage - LRS redundancy | RM905.97 |
| Support | | | RM0.00 |
| **Monthly Total** | | | **RM1,212.57** |
| **Annual Total** | | | **RM14,550.78** |

# 4.0 Implementation

## 4.1 Application development

The web-application Maersk Line Container Management System is developed with PHP in IDE NetBeans 8.2 version. The first page of the system is index.php, the header as in the navigation is code in separate PHP source file so it does not has to copy and paste in every single page, only require the statement “require (“header.php” )” to make sure every pages has the navigation bar.

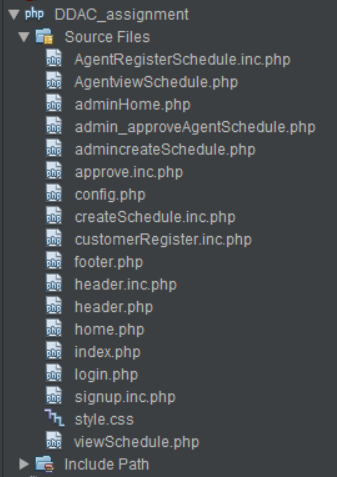


Figure 13 NetBeans screenshot

The website is developed with MySQL database, the MySQL database is included when the developer downloads the XAMPP, it is a free and open source cross-platform web server solution which include Apache HTTP Server and MySQL, when start testing the website simply start the Apache and MySQL from XAMPP control panel, then go to any browser and type “https //localhost/phpmyadmin” for the MySQL database, and “https //localhost/ddac\_website” for the website.

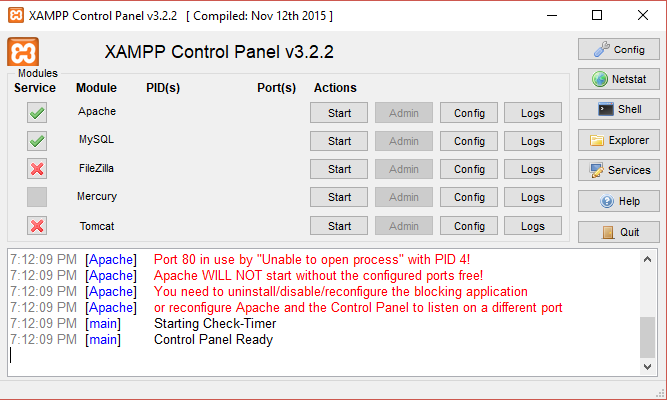


Figure 14 XAMPP control panels

When start developing the website in NetBeans, the first thing is to setup the connection between the following code with database to ensure it can fetch, read and insert data from the database.

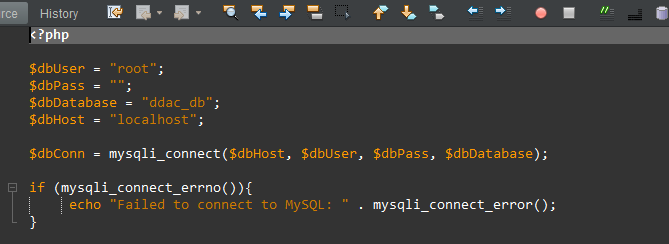


Figure 15 Code Snippet

As the code demonstrate how to connect to database with PHP, in PHP 7.0 and above, the statement “mysql\_query” is no longer support, all statement starts with “mysql” has to change to “mysqli” and the parameter is slightly different from the original. In the “header.php” has use the code “require (“config.php”)” which make all the webpage which has the header is able to directly call the “$dbConn” and linked with database.

A full demo of the functional system is included in the attached CD or online at <https://web.microsoftstream.com/video/38b89f8f-701f-4e71-b1b1-0bb993634694?search=TP038173_YeoKaiWen_DDAC> . The source code of the web application is available at <https://github.com/kaiwenddac96/kaiwenddac96.github.io> .

## 4.2 Azure Publishing

The system was published using the Azure automatic deployment option through GitHub. The benefits of it is when the developer update the code at GitHub, since the Microsoft Azure and GitHub is linked with this deployment, Azure will directly get the newest update from GitHub so the developer do not have to re-upload the code through FTP in Microsoft Azure.

Step 1 - Upload the code to GitHub

The very first step is to register an account on GitHub, when user login to the GitHub for the first time is able to select plan for GitHub (subscription), after selecting the free plan, user can create a new repository, after naming it, it requires the user to either use “msysgit” or GitHub desktop version, the developer use the GitHub desktop version and install it to personal computer. Next, after installing the application, the developer can relocate the local repository file on any place on local computer, then copy the website which developed on NetBeans and paste into the local repository file, the GitHub will detect the changes which made and ask whether want to commit the changes. Commit the changes and publish it to the branch of the GitHub, and the code is now successfully uploaded to the GitHub. The following figure is the code which uploaded to the GitHub.

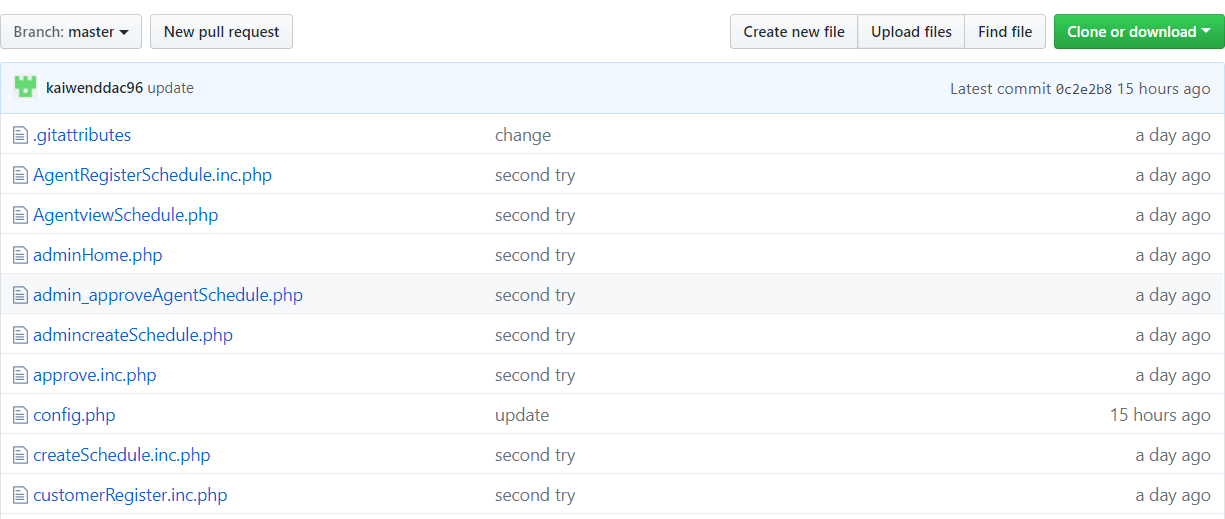


Figure 16 GitHub screenshot

Step 2 – Create a new Web Service on Microsoft Azure

After done publishing the code the GitHub, next step is to create the web service on the Microsoft Azure. After login to Azure, select “Create a resource” and create “Web App + MySQL”. Enter the application name, subscription, resource group, database provider, app service plan and location for the web application. All the names have to be unique to identify in resource group. Since is select the “Web App + MySQL”, it also required to setup database server in this section, the user can manually change the server name as in host name and server admin login name, the password also has to be created to establish the connection with the MySQL workbench in next few steps. The App Service Plan and Database Pricing Tier both have to choose wisely to make sure enough for current system. Not necessary reserve for future expand as the setting in the resource group can always “scale up” the plan and pricing tier. After all details have been filled, simply click “Create” and wait for few minutes for creating it.

Step 3 – Link with GitHub

After creating the resource group, next time is to link the resource group with the code that just upload in the GitHub. In the resource group, select “Deployment options” in “Deployment”, then select GitHub and link to the repository which created just created.

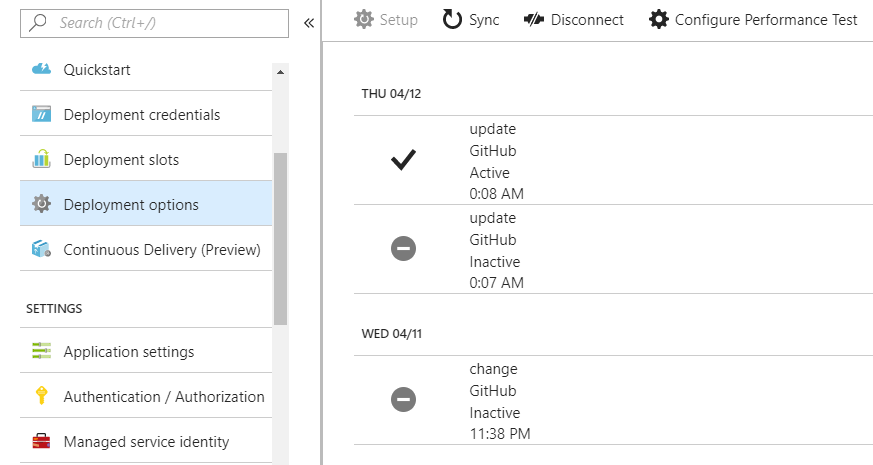


Figure 17 GitHub Desktop version

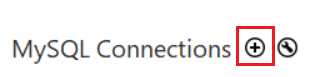
After it linked, the website can be access through the URL which located the resource group.



Since only the coding part has been linked, the website can be accessed but could not login as the database is not linked.

Step 4 – Connect with Database

The final step of publishing is to link the database, for MySQL, it required to use MySQL workbench. After installing the workbench, use the workbench to link the Azure database.



Select the icon which highlighted above, then setup the connection as:

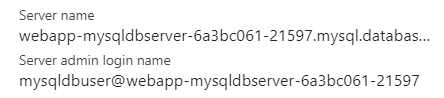


Figure 18 Azure screenshot

The host name will be the server name, the admin login is the username and the password has been created while creating “Web App + MySQL”.

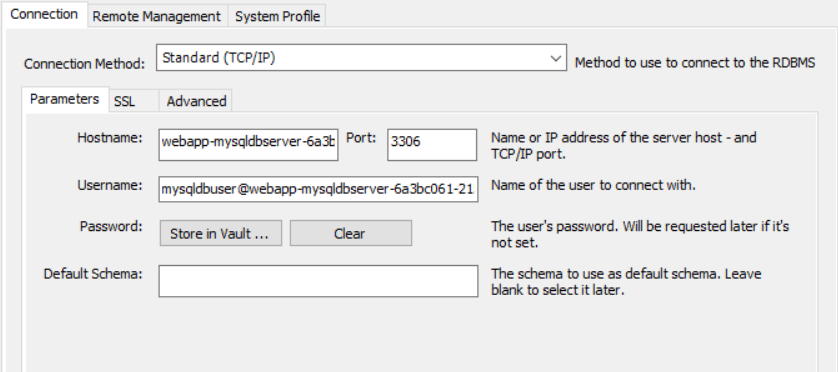


Figure 19 MySQL workbench screenshot

After all the information filled, create the connection and then export all the table from “PHPmyadmin” which is MySQL database and import into MySQL workbench. Then the website publishing is done.

## 4.3 Application Scaling

The web application deployed will be using S1 tier App Service Plan for the Maersk Line Container Management System in Southeast Asia region. As currently the web application still small, it does not consume too much RAM, the traffic manager is also available in the S1 tier, if future there is more region, the traffic manager can help at improve availability and responsiveness for high-performance applications. The Microsoft Azure allow the user to scale up or out at anytime, so when the system become larger, user can manually scale up the plan for higher RAM and more core to improve performance.

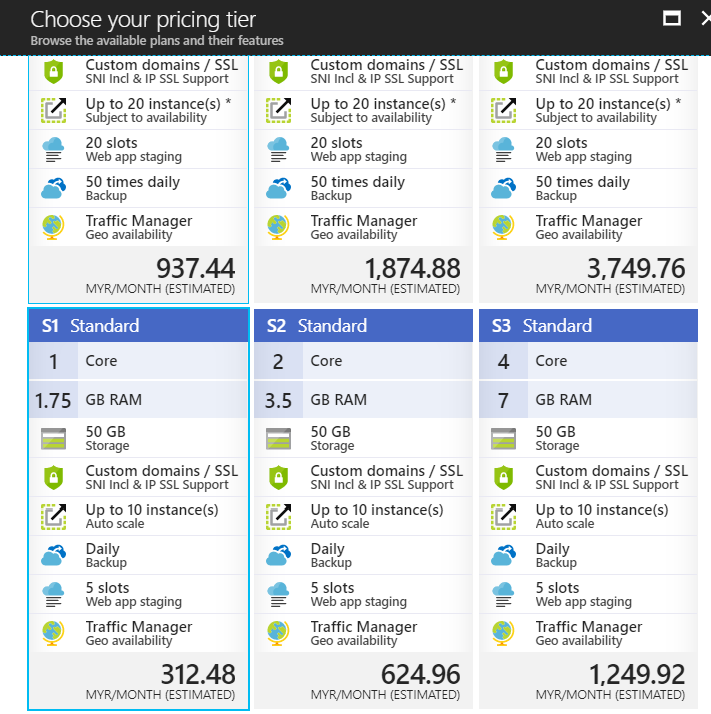


Figure 20 Azure App Service Plan

S1 Standard has the features such as auto scaling for in and out which can set rules when reaching a certain percentage, by increasing the instance count to distribute the workload, and daily backup to decrease the risk on losing data, it contains the minimum requirement for deploying a web application. With the custom domains, the company able to custom their own domains for their website by using add hostname.

When comes to scaling the application, the user can manually set the rule for scaling in and out when the application reaches certain percentage.

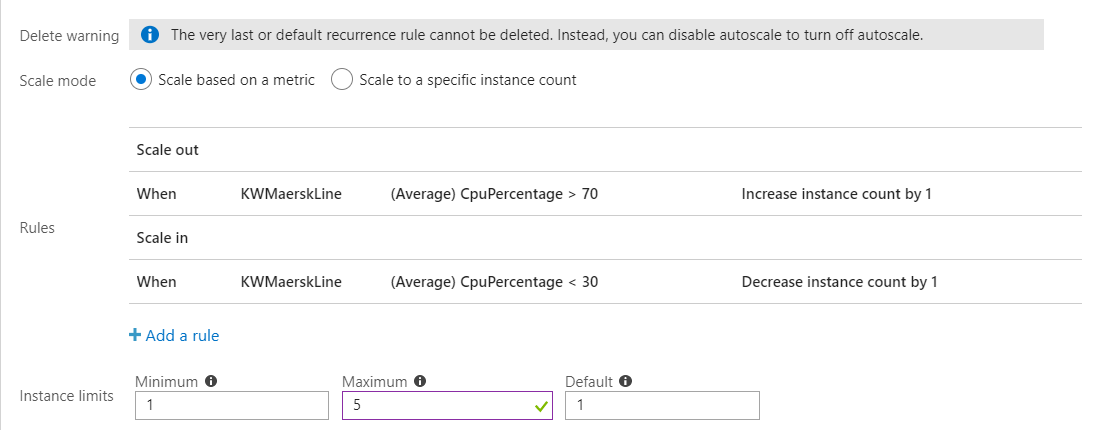


Figure 21 Scale out screenshot

The rule which has been added is increase instance count when CPU percentage over 70 percent and decrease when lower than 30 percent while the application is running.

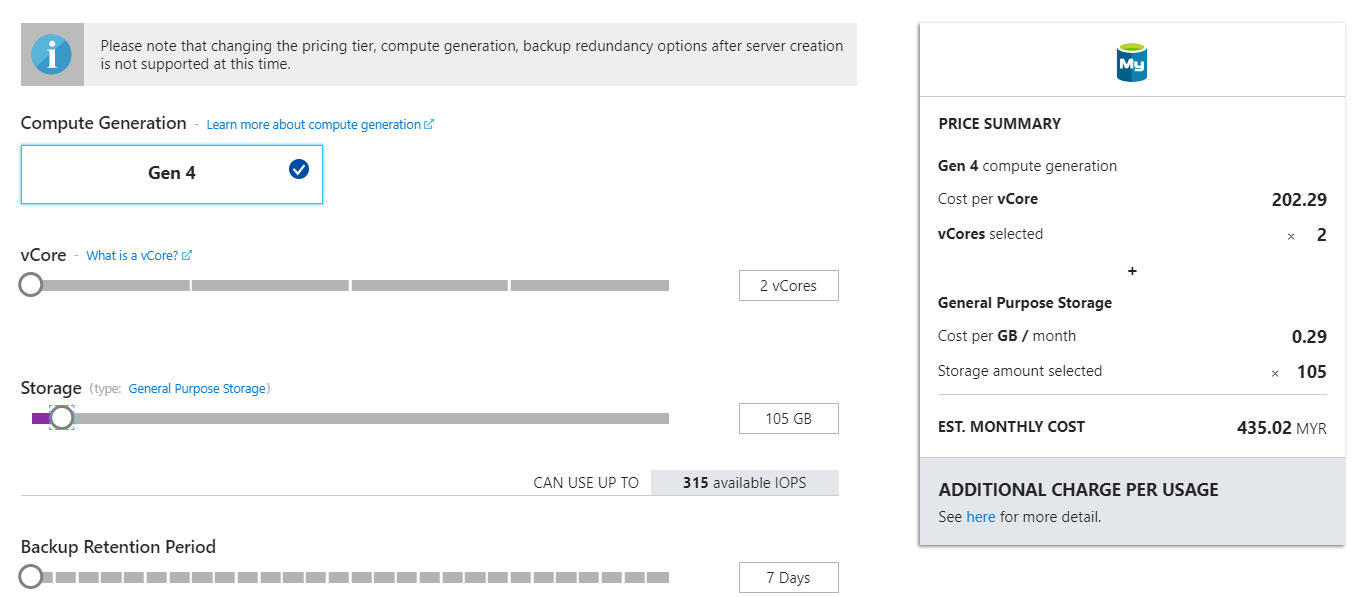


Figure 22 Database configuration screenshot

The Azure MySQL would be using the most standard plan, using the most basic 2 vCores and 105 GB storage. The database can be scaling up in future when it requires more storage and vCores. SQL Database automatically allocates 30% of additional storage for the log files and 32GB per vCore for TempDB, but not to exceed 384GB. TempDB is located on an attached SSD in all service tiers (Azure, 2018).

### 4.3.1 Reliability & Performance

As the S1 Standard pricing tier plan enable auto scaling features, it makes the app service has better performance and save the cost while the CPU usage is low. Autoscaling is the process of dynamically allocating resources to match performance requirements. As the volume of work grows, an application may need additional resources to maintain the desired performance levels and satisfy service-level agreements (SLAs) (Azure, 2017). In the App Service, there is an option in “Setting” which is “Scale out (App Service Plan)”, after enable the autoscale option, the web page will be similar with the following figure.

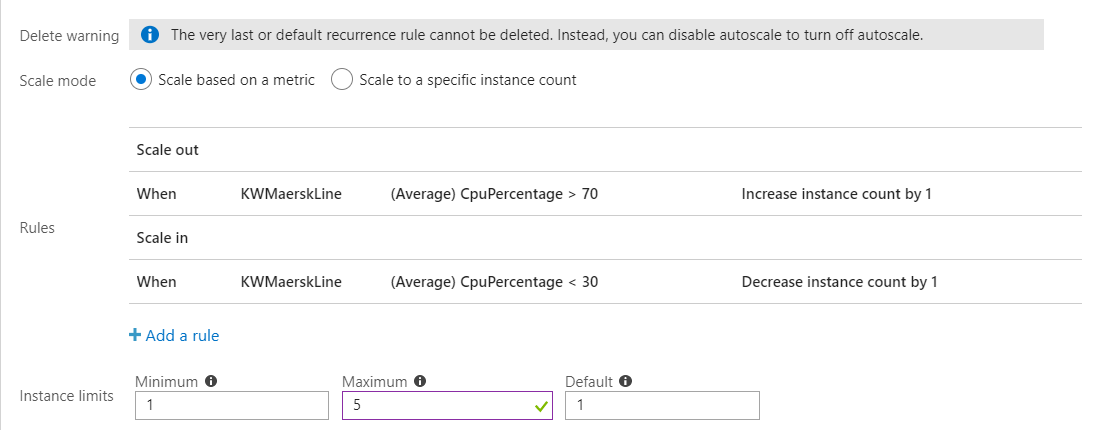


Figure 23 Database configuration screenshot

The rules can be categorized as “Scale out” and “Scale in”, which is adding instance count or remove instance count. The rules which have been set are when CPU usage percentage exceed over 70 percent, it will automatically increase instance to achieve better performance, when the CPU usage percentage is lower than 30 percent, it will decrease the instance count to save the cost, the amount of instance limit is set to 5 as the figure mentioned. This method is used to ensure the performance is consistency, so it achieve the reliability quality attribute. Allow the user to have shorter respond time while accessing the web application.

Next, Azure Database for MySQL is a managed service that you use to run, manage, and scale highly available MySQL Databases in the cloud. With Azure Database for MySQL server will achieve better performance as it is simple and can scale within seconds, it is built to deliver high availability require no extra configuration, replication, etc. So when applying the MySQL database which provided by Azure, the application will be more reliable.

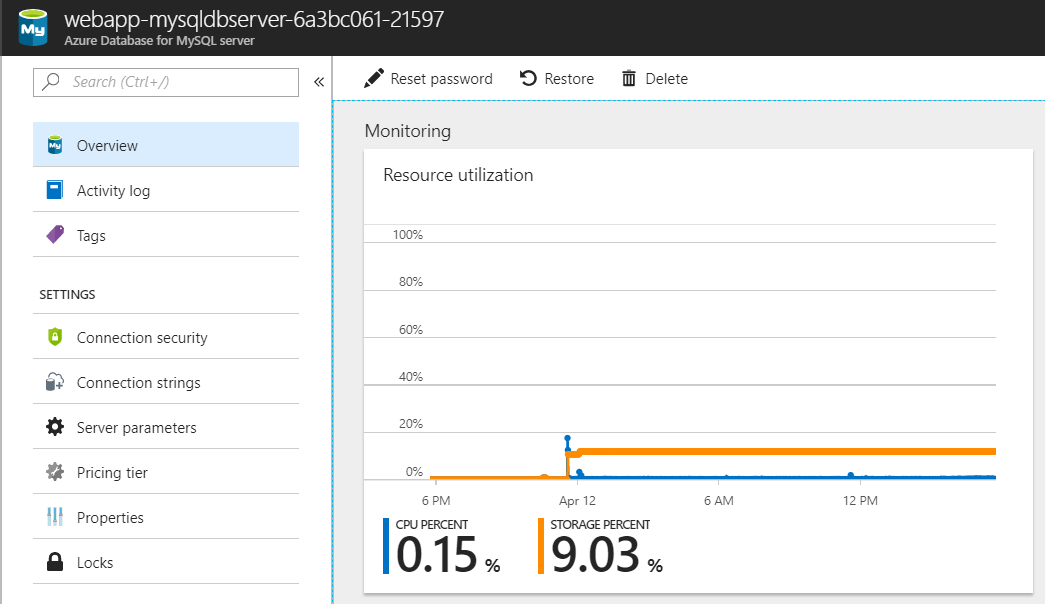


Figure 24 MySQL Database Resource utilization

# 5.0 Test Plan & Testing Discussion

## 5.1 Unit Testing

The following table is the test plan which created for unit testing, the test plan is to test the functionality of the system and check whether they work accordingly. The test is conducted after few functionalities done similar as the project plan.

### 5.1.1 Login Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 1 | Login with correct username and password. (Admin and agent) | Login successfully | Login successfully |
| 2 | Login with correct username but wrong password. (Admin and agent) | Login failed | Login failed |
| 3 | Login with wrong username but correct password. (Admin and agent) | Login failed | Login failed |

### 5.1.2 Logout Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 4 | Click “Logout” button after login to the personal account. | Logout successfully | Logout successfully |

### 5.1.3 View Schedule Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 5 | When click on the button “View Schedule”. | Schedules are showed. | Schedules are showed. |

### 5.1.4 Register New Agent Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 6 | Admin register a new agent with unique username and password. | Register successfully. | Register successfully. |
| 7 | Admin register a new agent with duplicate username but unique password. | Register failed | Register failed |
| 8 | Admin register a new agent with duplicate username but duplicate password. | Register failed | Register failed |

### 5.1.5 Define New Schedule Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 9 | Define a new schedule with every information filled. | Successfully created. | Successfully created. |
| 10 | Define a new schedule without filling all information | Schedule created failed. | Schedule created failed. |

### 5.1.6 Accept Agent Booking Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 11 | Click on the “Accept” button on the agent’s booking page. | Agent’s booking successfully accepted. | Agent’s booking successfully accepted. |
| 12 | Two admin accept one agent’s booking at the same time. | The admin who click first will success, the other will fail. | The admin who click first will success, the other will fail. |

### 5.1.7 Book Schedule Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 13 | Agent select the schedule from “View Schedule” webpage. | Customer registration form showed. | Customer registration form showed. |

### 5.1.8 Register Customer Function Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Case No. | Description | Expected Result | Actual Result |
| 14 | After selecting schedule, enter all customer information with a weight which is lesser than ship’s remaining weight. | Register successfully. | Register successfully. |
| 15 | After selecting schedule, enter all customer information with a weight which is greater than ship’s remaining weight. | Register failed. | Register failed. |
| 16 | After selecting schedule, agent did not enter all require information. | Register failed. | Register failed. |
| 17 | After successfully register a customer, when admin accept the booking, the ship will update the remaining weight. | Update successfully. | Update successfully. |

## 5.2 Performance

Through using Microsoft Azure, performance testing can be used to test the performance of the web application. In the testing, the developer tests the performance three times with 250, 500 and 750 concurrent users within 5 minutes. The figure below shows the configuration for the 500 concurrent users with a duration of 5 minutes. The option can be found in the resource group under the “development tools” option and select the performance test. For the free license type, it only can test the duration in 1 minute and maximum 40 user load, in this project, since it using S1 Standard App Service Plan, it can test up to 60 minutes as max duration and 20,000 user load (Microsoft, 2018).

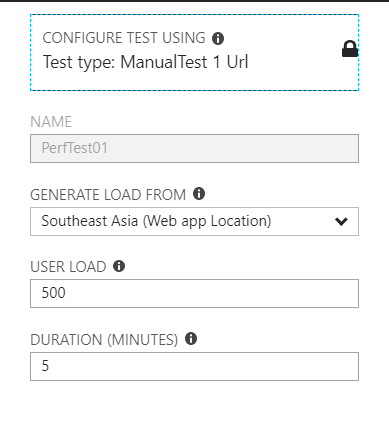


Figure 25 Performance Test Screenshot

After running the test, it takes few minutes to queue and complete the testing, the testing shows details such as total request successful and failed, the performance under load which has average respond time, etc. The following figure show the performance test of the web application with 500 concurrent users within 5 minutes.

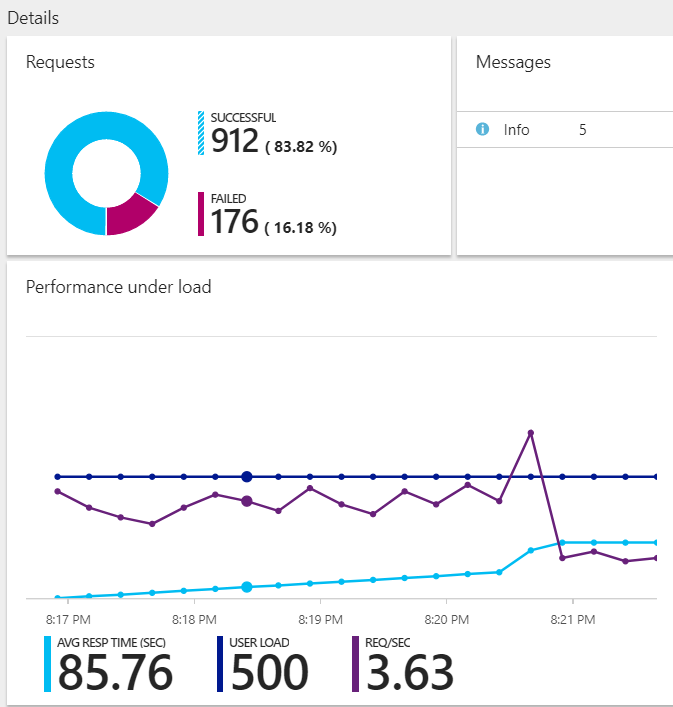


Figure 26 Performance Test Screenshot

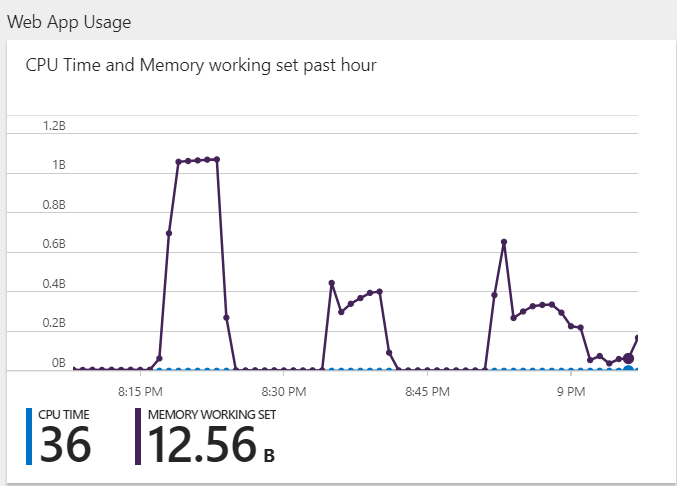


Figure 27 Performance Test Screenshot

As the result shown, the average respond time is 85.76 second and has 3.63 request per second. The table below show the comparison of three different user load which as 250, 500 and 750.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 250 concurrent users | 500 concurrent users | 750 concurrent users |
| Successful request | 926 | 912 | 832 |
| Failed request | 190 | 176 | 215 |
| Average response time | 85.13 | 85.76 | 92.4 |
| Request per sec | 3.72 | 3.63 | 3.49 |

As the table shown, the performance will drop when the concurrent user has become more and more, the average response time ranging from 85.13 seconds to 92.4 seconds. There is still much space to improve the performance of the web application, one of the way is to create the auto scaling rule to enhance the performance of the system.

## 5.3 Implementation & Discussion of Managed Databases

Platform as a Service which also known as PaaS, is a cloud model designed for software developers that simplify the development process by shifting specific aspects of systems management to the service provider, it often used to develop web and mobile applications using components that are pre-configured and maintained by the service provider (Sullivan, 2014). Microsoft Azure provide a complete development and deployment environment in the cloud. Besides Microsoft, there are also Amazon, IBM, Google and many others are proving the cloud service in current market. Microsoft is second largest cloud services provider beside the Amazon (Gacinga, 2014).

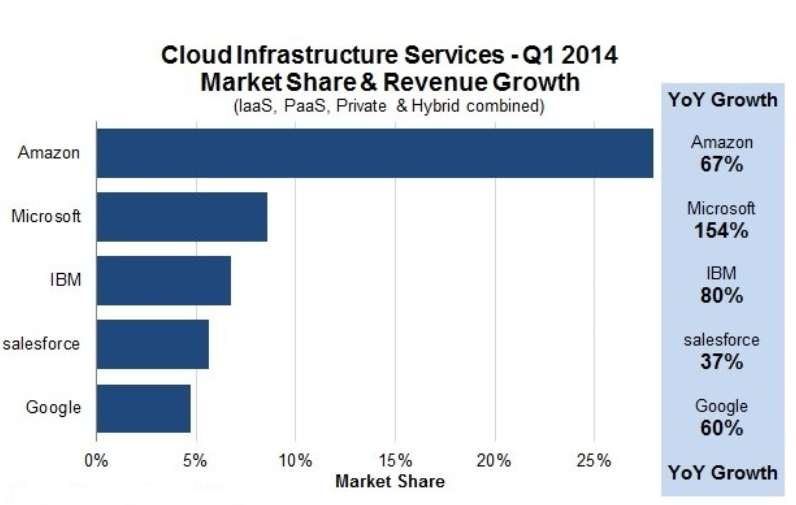


Figure 28 Market Share & Revenue Growth for Cloud Infrastructure Services (Gacinga, 2014)

In infrastructure as a service (IaaS), it offers server/storage, networking firewalls and datacentre infrastructure for the user, these hardware is all provided and managed by the provider, which allow the user to reduce the cost on buying physical servers, etc. PaaS is similar with IaaS, include infrastructure such as servers, storage, etc., it also has middleware, development tools, business intelligence (BI) services, database management systems, etc (Azure, 2018).

The Maersk Line Container Management System is developed based on PaaS which provided by Microsoft Azure as it has many advantages, as the Microsoft Azure already providing a platform which allow the user to create resource group such as “Web App + MySQL” and easy to manage resource such as CPU usage or others, it also allow the developer to grant privilege to user to access the resource group.

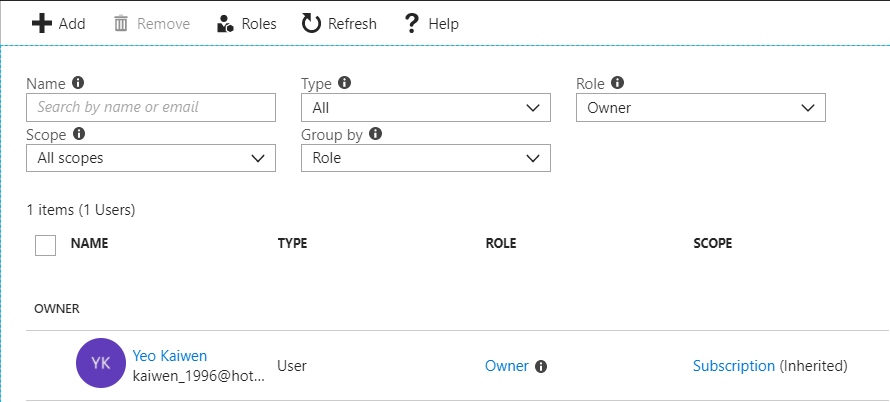


Figure 29 Azure screenshot

For example, in the developer’s resource group, the only user who can access the resource group is the developer himself, which can improve the security as it limits the user who can access it. With PaaS, if also can efficiently manage the application lifecycle which are building, testing performance, deploying, managing and updating within the same integrated environment. As the user accesses the Microsoft Azure through website, it is not an application that has to depends on Linux, Windows or Mac to download different version. Azure can be accessed through the web browser which archive cross-platform.

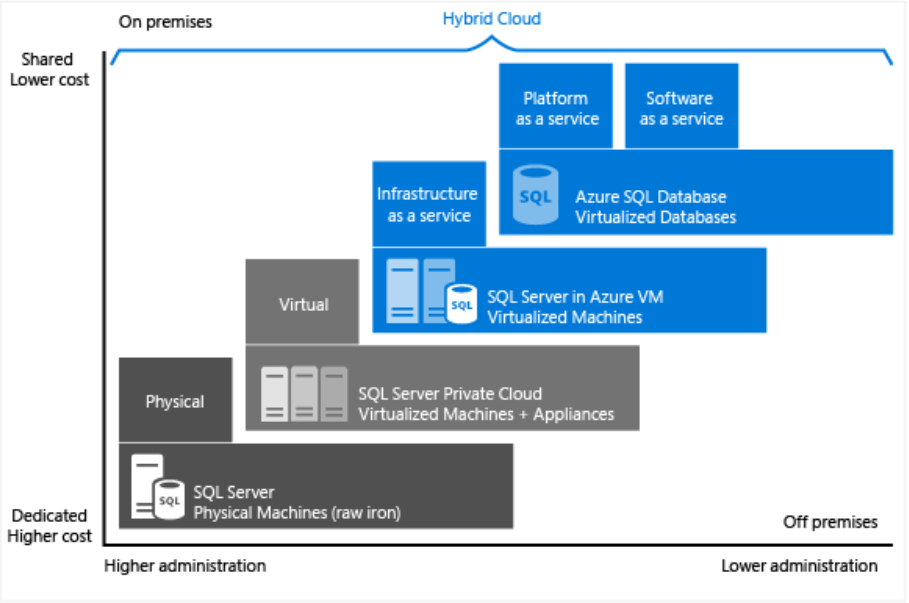


Figure 30 Characteristic of level of administration over degree of cost efficiency (Azure, 2018)

There are four ways to host the database as the figure showed above, Microsoft Azure provide two options to host SQL Server workloads which are Azure SQL Database and SQL Server on Azure Virtual Machines. For Azure SQL Database, it is a SQL database native to the cloud, which is PaaS or can be called database as a service (DBaaS), it is optimized for software as a service (SaaS) app development (Azure, 2018). On the other hands, SQL Server is installed and hosted in the cloud on Windows Server Virtual Machines (VMs) running on Azure, known as an infrastructure as a service (IaaS). It is optimized for migrating SQL Server applications (Azure, 2018). It allows the user to choose which version when choosing it as different version can have different syntax while executing query, user also can host as many databases as needed and do cross-database transactions.

# 6.0 Conclusion

In this documentation, it has discussed the project background at chapter 1, the design of the Maersk Line Container Management System such as use case diagram, description, ERD, etc., have been demonstrated to make sure the development goes smoothly. The implementation such as using PHP and MySQL database for this system and the step to publish the system to Microsoft Azure, and the justification of application scaling also been discussed. Through unit testing and performance testing to ensure the system functionalities work accordingly and reliability. In conclusion, the assignment has demonstrated development and publish the system to the cloud computing with Microsoft Azure. Through the progress of publishing, the developer has learned the skill such as considering the storage of database and performance of the system are important during deployment, developer also learned the concept of cloud architecture for cloud computing for future career.

# References

Azure, M., 2017. *Autoscaling.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling  
[Accessed 7 4 2018].

Azure, M., 2018. *Azure SQL Database vCore-based purchasing model limits (preview).* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-vcore-resource-limits  
[Accessed 8 4 2018].

Azure, M., 2018. *Choose a cloud SQL Server option: Azure SQL (PaaS) Database or SQL Server on Azure VMs (IaaS).* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas  
[Accessed 11 4 2018].

Azure, M., 2018. *What is PaaS?.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/  
[Accessed 10 4 2018].

Gacinga, J., 2014. *Microsoft Corporation Becomes the Second-Largest Cloud Services Provider.* [Online]   
Available at: https://www.fool.com/investing/general/2014/05/11/microsoft-becomes-second-largest-cloud-services-pr.aspx  
[Accessed 10 4 2018].

Microsoft, 2018. *FAQs for load testing.* [Online]   
Available at: https://docs.microsoft.com/en-us/vsts/load-test/reference-qa?view=vsts#qaazure-limits  
[Accessed 9 4 2018].

Sullivan, D., 2014. *PaaS Providers List: Comparison And Guide.* [Online]   
Available at: http://www.tomsitpro.com/articles/paas-providers,1-1517.html  
[Accessed 10 4 2018].