Contents

[1.0 Introduction 2](#_Toc520226000)

[1.1 Project Background 2](#_Toc520226001)

[1.2 Project Objective 2](#_Toc520226002)

[1.3 Project Scope 2](#_Toc520226003)

[1.4 Project Deliverables 2](#_Toc520226004)

[2.0 Project Plan 4](#_Toc520226005)

[3.0 Design & Solution Architecture 5](#_Toc520226006)

[3.1 Use Case Diagram 5](#_Toc520226007)

[3.2 Sequence Diagram 6](#_Toc520226008)

[3.3 ERD Diagram 10](#_Toc520226009)

[3.4 Site Map 11](#_Toc520226010)

[3.5 Cloud Architecture 13](#_Toc520226011)

[3.6 Cloud Design Pattern 14](#_Toc520226012)

[4.0 Implementation 16](#_Toc520226013)

[4.1 Application Development 16](#_Toc520226014)

[4.2 Deployment 18](#_Toc520226015)

[4.3 Database 19](#_Toc520226016)

[4.4 Application Scaling 20](#_Toc520226017)

[4.5 Traffic Manager 21](#_Toc520226018)

[4.6 Auto Scaling 22](#_Toc520226019)

[4.7 Testing 23](#_Toc520226020)

[4.7.1 Unit Testing 23](#_Toc520226021)

[4.7.2 Performance Testing 29](#_Toc520226022)

[4.8 Managed Database 32](#_Toc520226023)

[4.8.1 Common PaaS Scenario 33](#_Toc520226024)

[4.8.2 Advantages 34](#_Toc520226025)

[5.0 Conclusion 35](#_Toc520226026)

[6.0 Reference 36](#_Toc520226027)

# 1.0 Introduction

## 1.1 Project Background

Maersk Line is the largest container shipping company having customers through 374 offices in 116 countries. Maersk Line is looking at designing and developing a Container Management System (CMS) to cater to manage the container, a solution that reduces overall supply chain costs and an efficient way to manage logistics. The system is developed in web application hosted on Microsoft Azure as an App Service (Web App) which consume SQL Database.

## 1.2 Project Objective

* To develop a Container Management System and deploy on Microsoft Azure at the highest performance

## 1.3 Project Scope

The Container Management System is designed with modelling, that will undergo application development. Once the development is done, it will be tested and deployed in Microsoft Azure. Configuration in Azure such as traffic manager is implemented and performance test is also conducted.

## 1.4 Project Deliverables

Customer

* Allow users to register
* Allow users to login
* Allow users to request shipping
* Allow users to view shipping

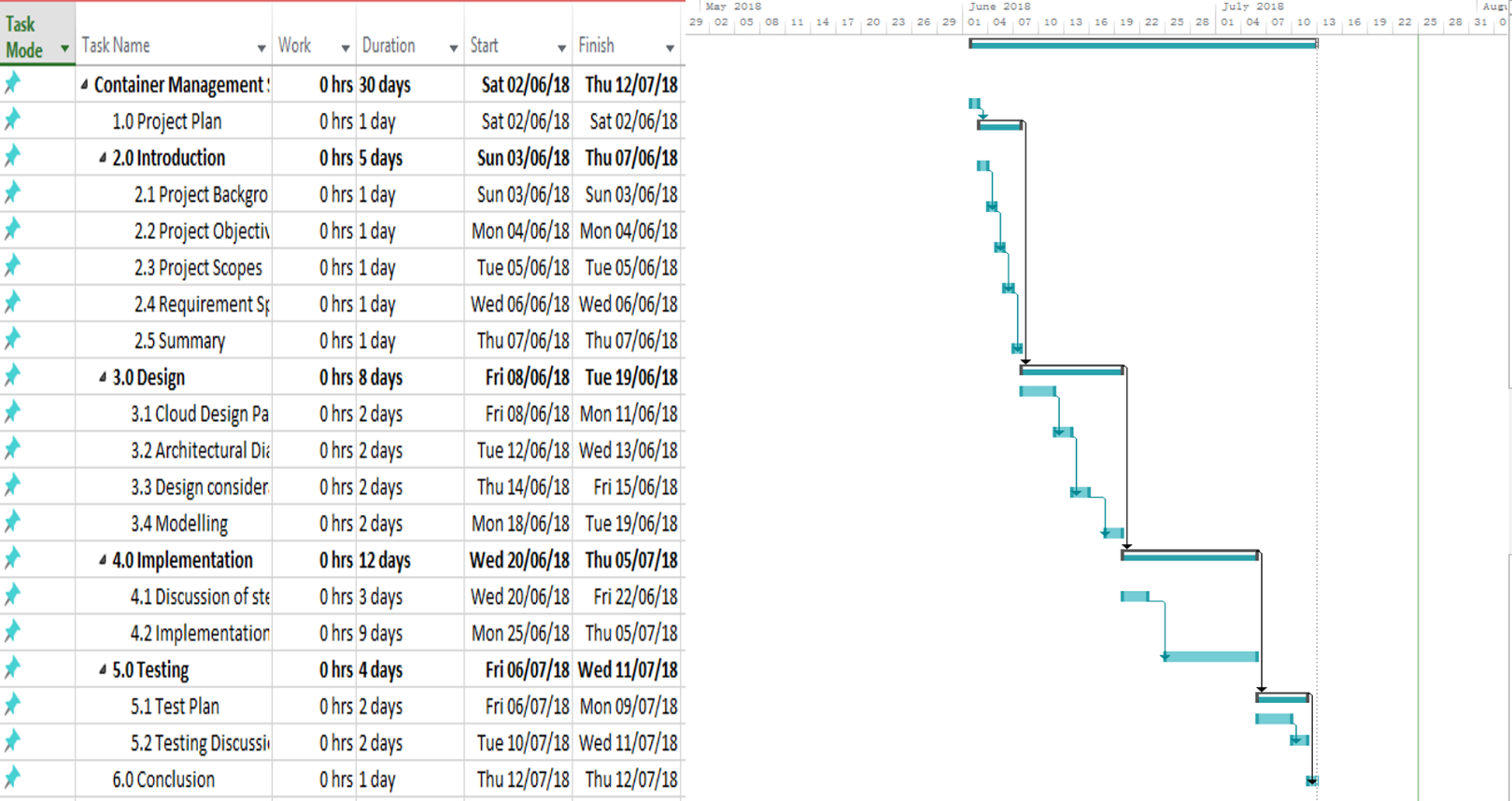
Departure Port User

* Allow users to view customers’ shipment request
* Allow users to approve shipment request
* Allow users to update shipment status to shipping

Arrival Port User

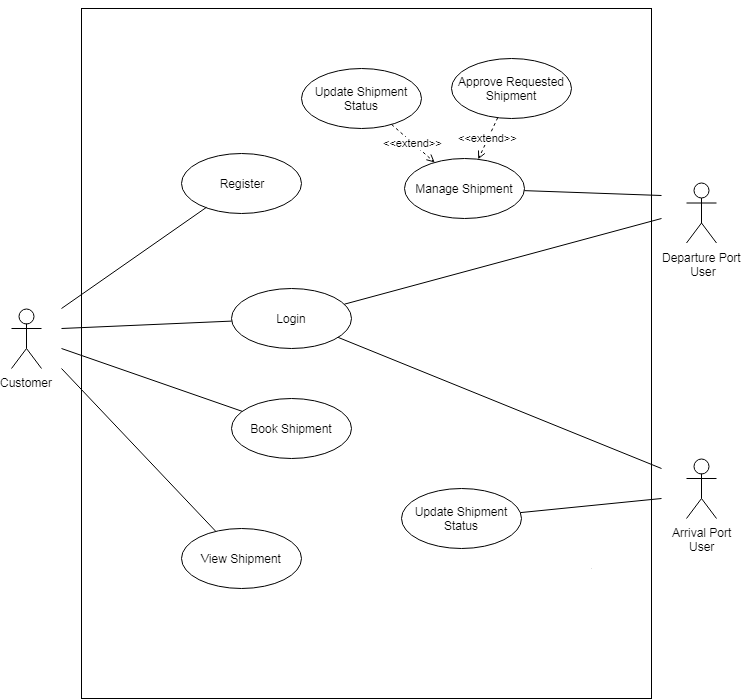
* Allow users to view notification
* Allow users to update shipment status to arrived

# 2.0 Project Plan

* 

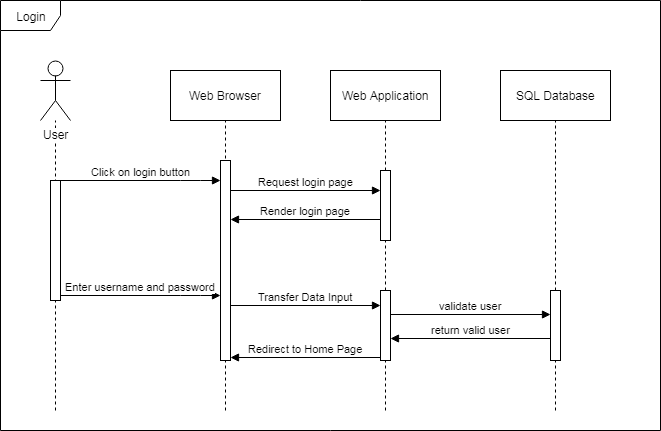
# 3.0 Design & Solution Architecture

## 3.1 Use Case Diagram

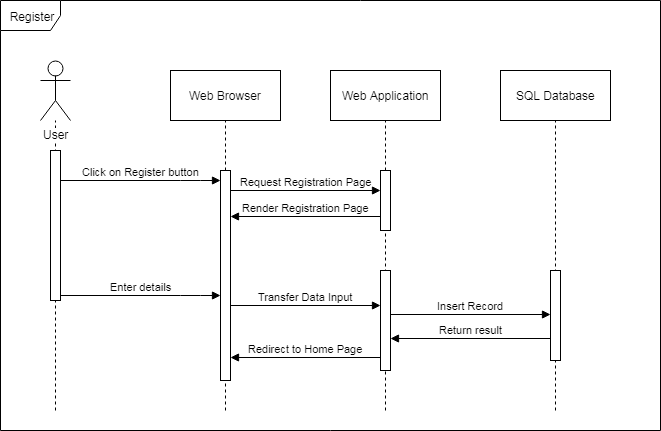


## 3.2 Sequence Diagram

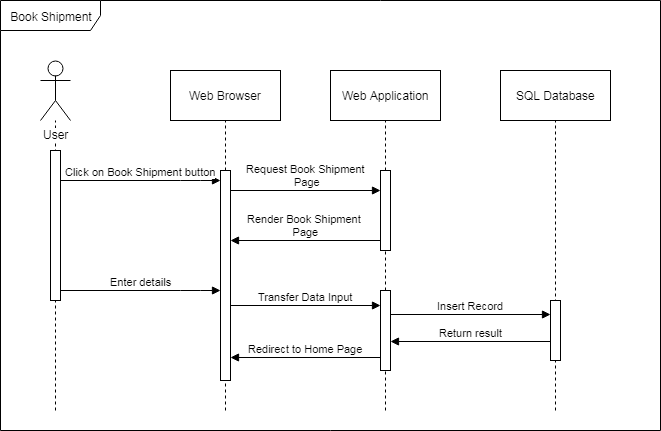
**Login**



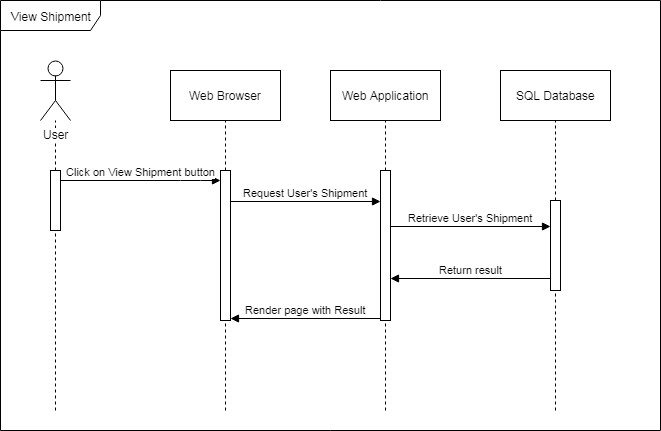
**Register**



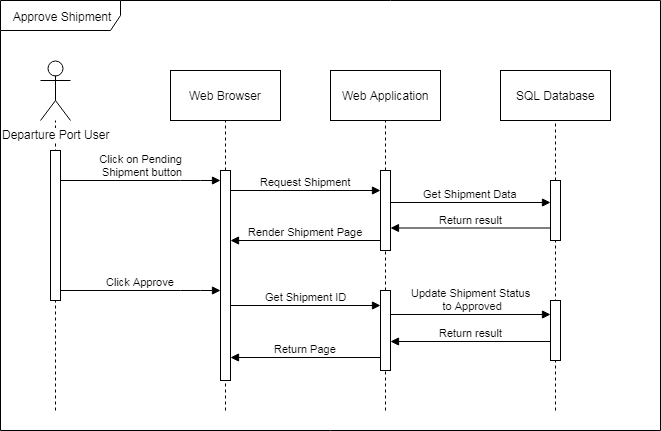
**Book Shipment/Request Shipment (Customer)**



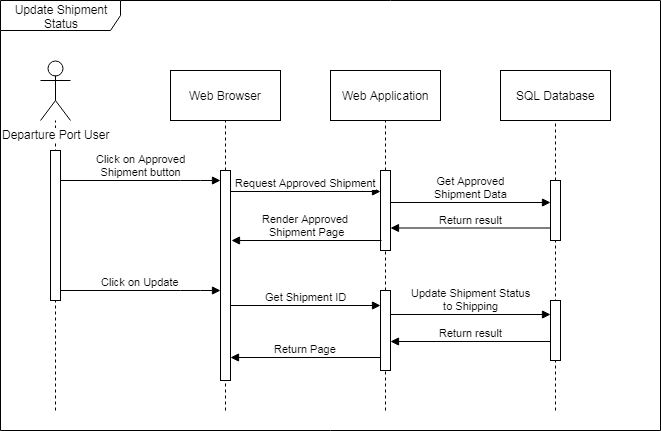
**View Shipment (Customer)**



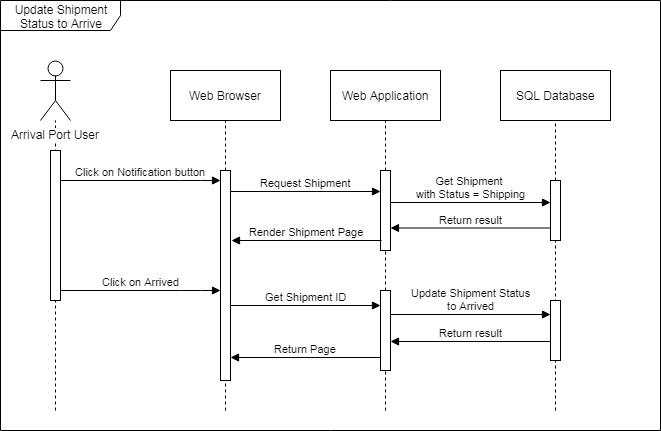
**Approve Shipment (Departure Port User)**



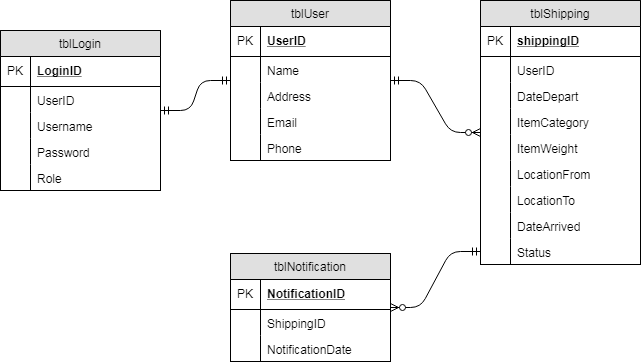
**Update Shipment Status to shipping (Departure Port User)**



**Update Shipment Status to Arrive (Arrival Port User)**

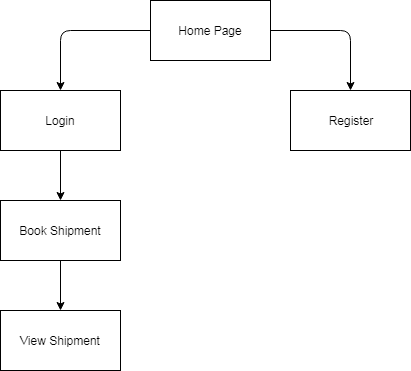


## 3.3 ERD Diagram

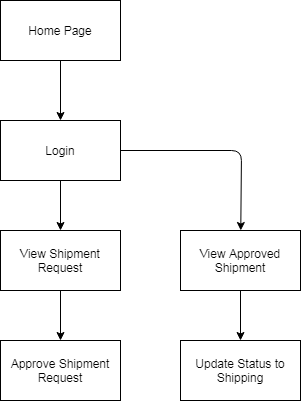


## 3.4 Site Map

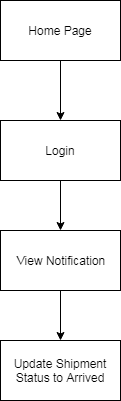
**Site Map (Customer)**



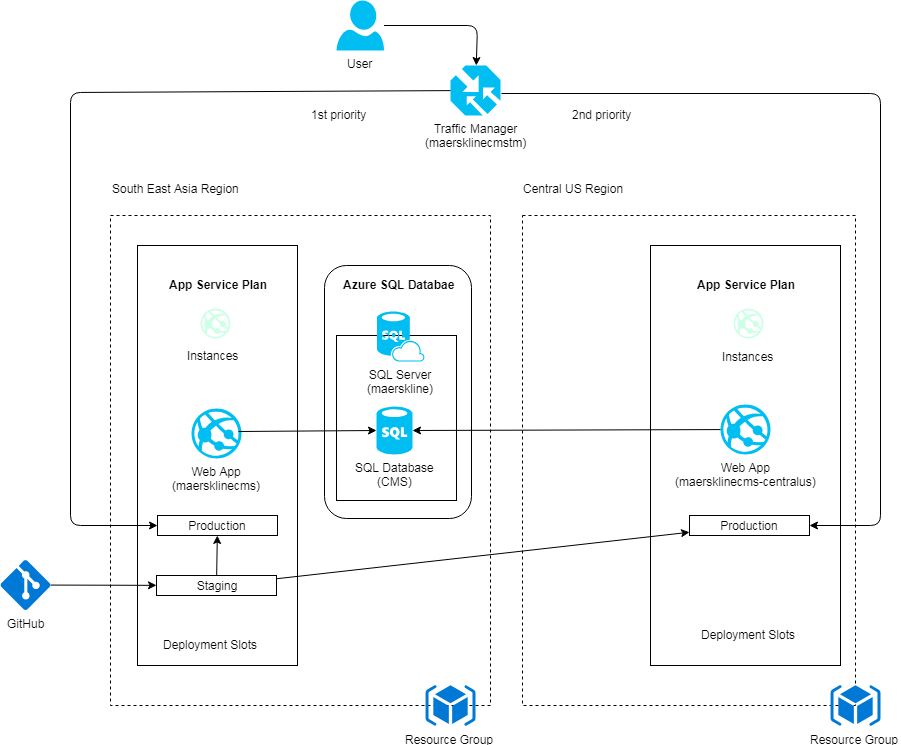
**Site Map (Departure Port User)**



**Site Map (Arrival Port User)**



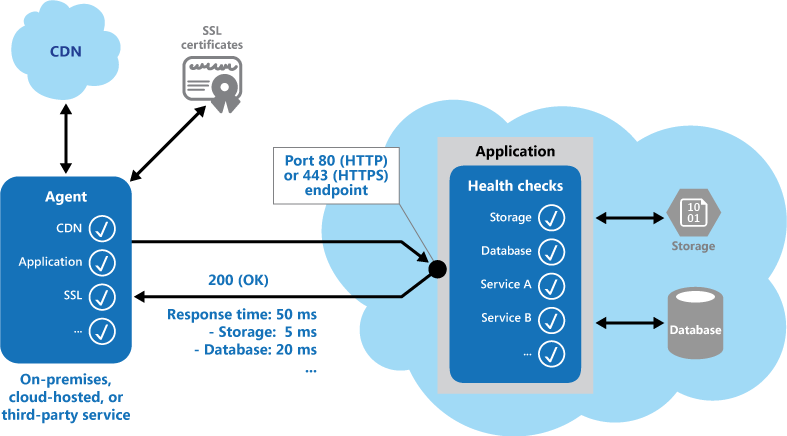
## 3.5 Cloud Architecture



The diagram above shows the cloud architecture used for deploying Container Management System web application on Microsoft Azure cloud platform. Maersk Line prioritize on expanding into the South-East Asia region and Central US as their secondary target region. Two web apps is hence placed in the South-East Asia region and the secondary web app is placed in Central US region. SQL Server used to store the data transaction which is placed in primary region. South-East Asia. The same database is shared across the secondary web application which is deployed in Central US.

## 3.6 Cloud Design Pattern

One of the cloud design pattern is Health Endpoint Monitoring Pattern. It ensures the application and services are performing correctly by sending requests to an endpoint of the application to perform necessary checks which will return an indication of the status. The diagram below shows an overview of the design pattern.



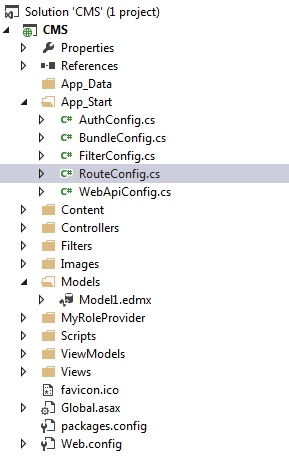
The total estimated cost for implementing cloud architecture drawn above is as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Service Type** | **Region** | **Description** | **Estimated Cost** |
| App Service Plan | South East Asia | S1 standard, 1x cores, 1.75GB memory, support features: -  Traffic manager, auto scales | RM 312.48 |
| App Service Plan | Central US | S1 standard, 1x cores, 1.75GB memory, support features: -  Traffic manager, auto scales | RM 312.48 |
| SQL Server Azure Database | South East Asia | Standard S2, 50 DTU, 250 GB | RM 315.08 |
| Monthly Total | | | RM 940.04 |
| Annual Total | | | RM 11280.48 |

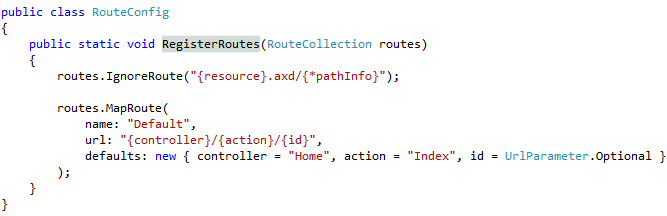
# 4.0 Implementation

## 4.1 Application Development

The Web Application is developed by using C# ASP .NET MVC Web Application framework.



The URL request will first goes to RegisterRoutes() function in RouteConfig.cs which is called during application starts.

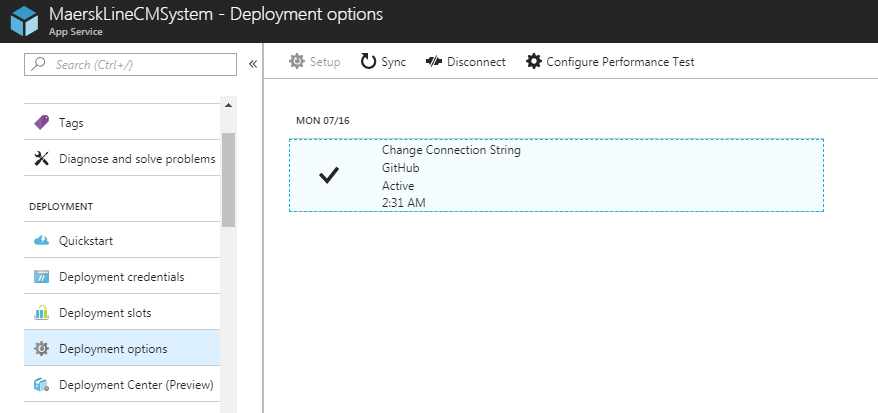
 The request is delivered to the appropriate controller and at this phase, the controller will determine the proper action in it through request passed in URL. The GET method of proper action will be called and its View (User Interface) will be rendered to the user. Any submitted inputs in the form groups by users will be transferred to the View’s Controller and its POST method will be invoked. Activities such as data validation, and querying database takes place here. The Model is where the validation and communication with database takes place. The Model transfers the inputs, which were passed from the View to Controller, to database and the result of the database queries invoked is return by Model to the Controller and Controller renders the result to View for user.

## 4.2 Deployment

App Services

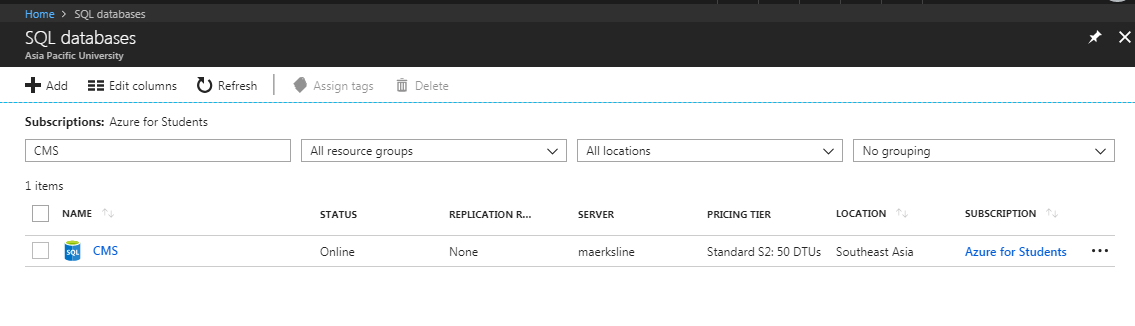
The web application is developed and tested on local server by developer and is deployed into Microsoft Azure cloud service in South-East Asia region and Central US region. The reason of deploying the web application in these two regions is to conduct user load test by simulating traffics coming from these two targeted regions. The deployment tool to publish the web application hosted on the two targeted regions is by using single repository in GitHub source control management service, thus changes made only need to be updated once to both of the hosted web application. Both web applications apply S1 Standard app service to allow the web applications to be configured in traffic manager. The traffic manager is set to use performance routing method and two endpoints are added to both web applications deployed in South-East Asia region and Central US region. Hence, the web application surf by user will be redirected according to the user’s geographical area where it achieves at higher performance.

The web application is published using GitHub through the Azure automatic deployment options. The option enables all changes to be pushed to the master branch in GitHub repository and deploys directly to the Azure servers as shown below along with the branch history. This method was used in conjunction with the deployment slots to provide a continuous delivery environment.

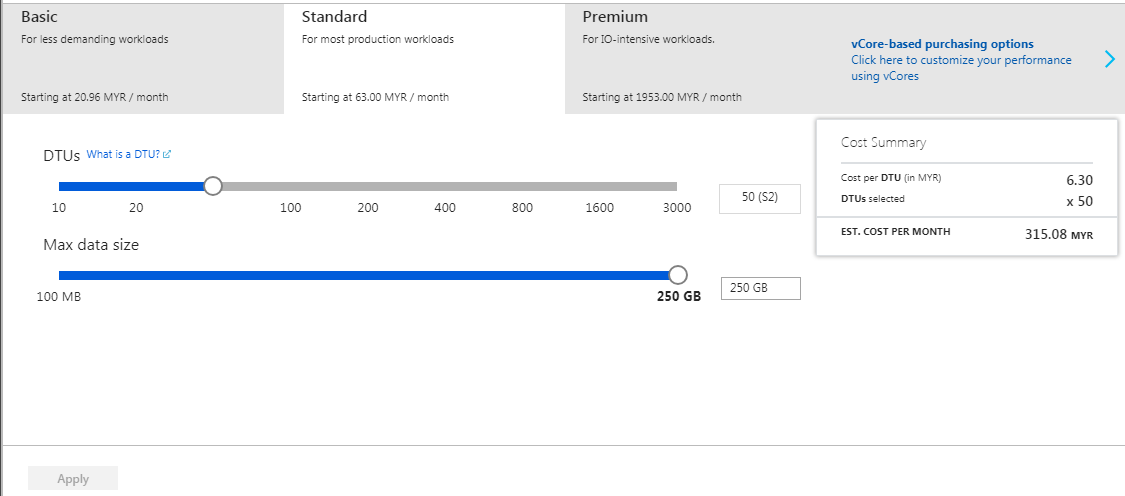


## 4.3 Database

SQL server on Azure is used as the database for the development. It offers features such as scaling and monitoring. Since the database operates independently of the application and cannot be stopped, it was used directly during the development to avoid the need to change configuration files upon pushing changes to the production stage.



The pricing tier for Azure SQL Database is Standard S2: 50 DTUs. It provides 50 DTUs which will be sufficient to serve for two regions. It will also have a total of 250GB storage size which is enough to support the functions. The pricing tier will be upgraded if any changes in business requirement or the users load increase.

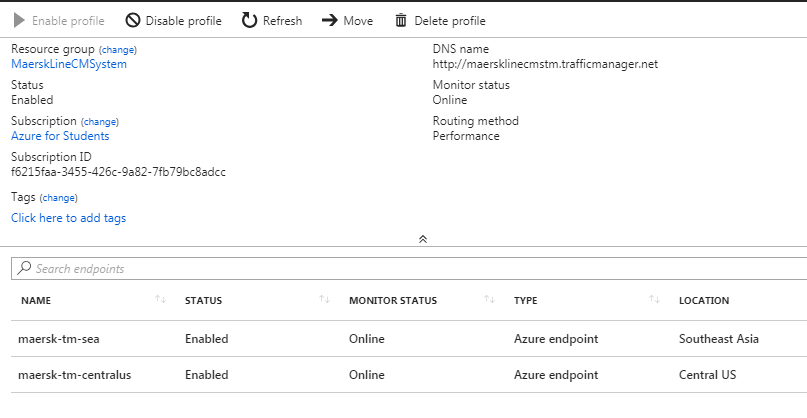


## 4.4 Application Scaling

The web application is deployed to South-East Asia Region and Central US Region which use the S1 tier App Service as the allocated budget is limited. The App Service plan will be upgraded to S3 plan or higher once Maersk Line approved the project and is ready to be published for publics use.

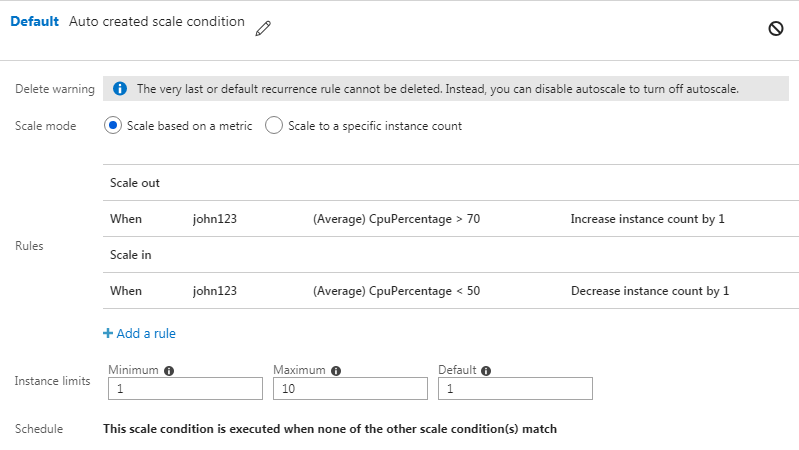
Both web applications use Standard S1 plan as it has essential features such as custom domains for Maersk Line to use. It provides up to 10 instances to be created as well as auto scaled (out or in) based on metrics such as server load and memory usage. Besides that, data in server and configuration are backup daily to ensure no data lost due to any uncertain circumstances. For the CI/CD workflow employed, this is needed. Developer can continue to develop the application and know what changes has been made each time they commit the changes, making source control easier.

## 4.5 Traffic Manager



The select pricing tier is Standard S1 plan which supports Traffic Manager and it is implemented. This allows to control the distribution of user traffic for service endpoints in different datacenters. It enables the website to improve the responsiveness of the system by redirecting the traffic to the endpoint with lowest network latency for user. For example, users in South-East Asia will redirect to SEA endpoint, reducing the time of response of the system thus improving the performance of the system.

## 4.6 Auto Scaling



Application service will run 1 instance which could add up to maximum 10 instances for higher reliability. Rules are set that an additional instance will be added every 10 minutes whenever the server CPU usage is more than 70% and an instance will be removed by one every 10 minutes when it is below 50%. When the business need expands in future, the application service can be scaled up to use a better premium plan which provides even more features such as higher backup frequencies and better hardware such as SSDs.

## 4.7 Testing

### 4.7.1 Unit Testing

**Registration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| R01 | Input Validation | 1. Enter registration page. 2. Leave all required information. 3. Click Register. | Messages will be shown for required fields that are not filled and prompt to enter input | As expected |
| R02 | Valid Input and Register | 1. Enter registration page. 2. Register with username “aaa” 3. Fill in other required fields with correct pattern and datatype. 4. Click submit. | Registration success and redirect user to Home Page | As expected |
| R03 | Enter Existing username | 1. Enter registration page. 2. Register with username “aaa”. 3. Fill in other required fields with correct pattern and datatype. 4. Click submit. | “Username is not available” message will be shown. | As expected |

**Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| L01 | Input Validation | 1. Enter login page. 2. Leave username and password blank. 3. Click login. | Prompts user to enter username and password | As expected |
| L02 | Invalid login credentials | 1. Enter login page. 2. Enter invalid username and password. 3. Click login. | “Invalid username or password” message will be shown and prompt user to enter correct details. | As expected |
| L03 | Valid login credentials | 1. Enter login page. 2. Enter correct username and password. 3. Click login. | Redirects user to Home Page | As expected |

**Book** **Shipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| BS01 | Input Data Type input | 1. Enter book shipment page 2. Enter “abc” in weight field 3. Click Book Shipment Button | “The field of Weight must be a number” message will be shown and prompt user to enter valid datatype input. | As expected |
| BS02 | Invalid Date selection | 1. Enter book shipment page 2. Select the date before today e.g. yesterday’s date. 3. Click Book Shipment Button | “Please select one day after today” message will be shown and prompt user to select valid date. | As expected |
| BS03 | Valid Booking Details | 1. Enter book shipment page. 2. Enter valid datatype fields. 3. Select valid date. 4. Click Book Shipment Button | Book Shipment Success and redirects user to Home Page. | As expected |

**View Shipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| VS01 | List all Shipment | 1. Enter view shipment page. | All user’s shipment details will be shown in table. | As expected |

**Approve Shipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| AS01 | List all Pending Shipment | 1. Enter pending shipment page | All users’ Shipment with status pending will be shown in table. | As expected |
| AS02 | Approve Shipment | 1. Enter pending shipment page 2. Click Approve button | The Shipment status will be updated to Approved and save into database. | As expected |

**Update Shipment to Shipping**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| US01 | List all Shipment | 1. Enter approved shipment page | All users’ Shipment with status “Approved” and arrived will be shown in table | As expected |
| US02 | Update Shipment | 1. Enter approved shipment page 2. Click Update Button | The Shipment status will be updated to Shipping and save into database | As expected |
| US03 | Status “Approved” Shipment placed at top row of table | 1. Enter approved shipment page | The shipments details with status “Approved” will be placed at the top row of table. |  |

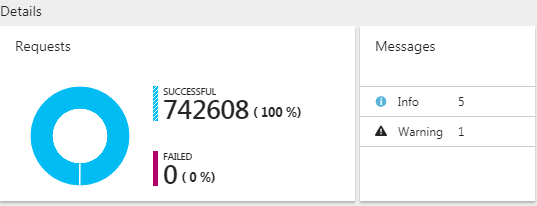
**Update Shipment to Arrived**

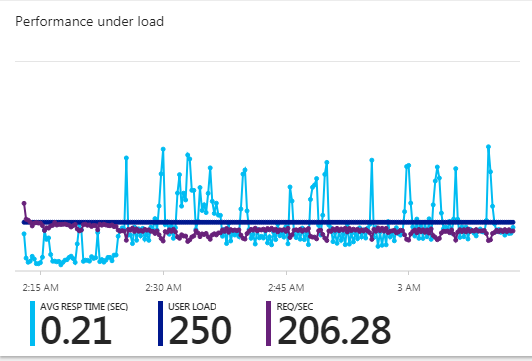
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Title | Description | Result | |
| Expected | Actual |
| UA01 | List all notification regarding Shipment that is under Shipping process | 1. Enter notification page | All users’ Shipment with status “Shipping” will be shown in table | As expected |
| UA02 | Update Shipment | 1. Enter notification page 2. Click “Arrived” Button | The Shipment status will be updated to Arrived and save into database | As expected |

### 4.7.2 Performance Testing

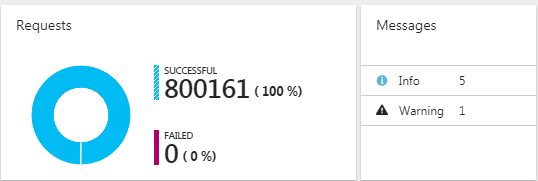
Performance tests were conducted through the functionality provided by Azure web app. The tests will be conducted on the main web app resource and will test a user load of 250 to 350 for 1 hour. The results gathered will include response time and average request per second. The response time recorded for the tests are the time taken for the entire web page to load, and will be recorded in seconds.

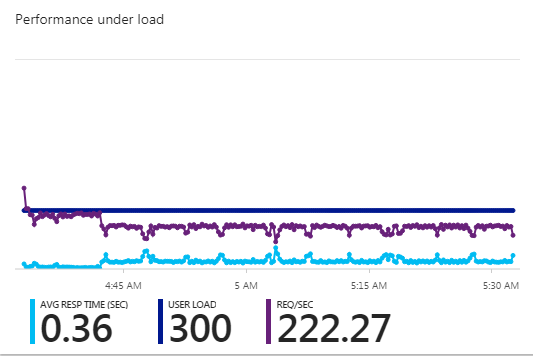
**Performance Test 1 (User Load of 250)**



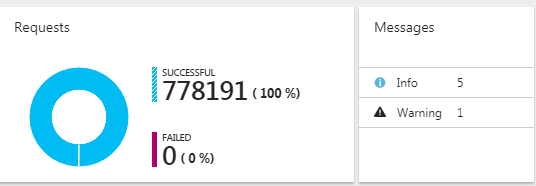


**Performance Test 2 (User load of 300)**





**Performance Test 3 (User load of 350)**



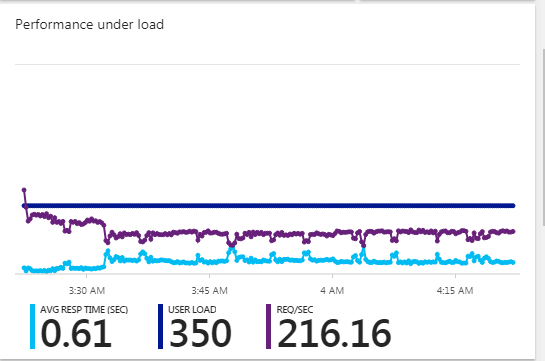


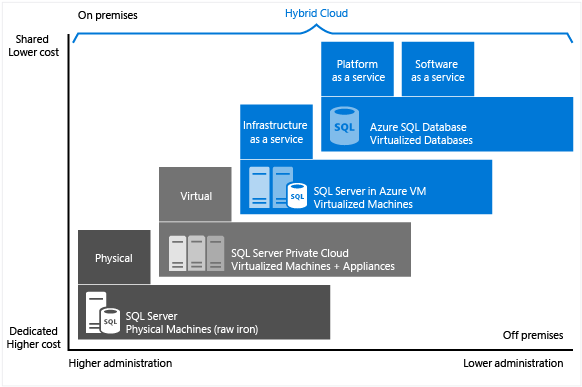
Table of user load and average response time

|  |  |  |
| --- | --- | --- |
| User load | Average Response Time (sec) | Request per Sec |
| 250 | 0.21 | 206.28 |
| 300 | 0.36 | 222.27 |
| 350 | 0.61 | 216.16 |

As a result, all the requests are successful among the 3 performance tests and the average response time in second increases when the user load expand, thus the average response time is getting slower when there are more users surfing the web application. Therefore, the app service plan will have to scale up if the user loads are increasing throughout the future.

## 4.8 Managed Database

Platform as a Service (PaaS), is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet. PaaS services are hosted in the cloud and accessed by users simply via their web browser. (Interoute, 2017). It is regularly used by developers, from conception to the creation of applications, through to testing and deployment. Generally, users only need to pay for what they have used on a subscription basis which resulting in economies of scale. Customers can select the features they want and discarding others, choosing a service that suits their needs.



The diagram above show the cost and the administration gets lower by transferring from physical SQL server to Azure SQL Database (PaaS) and yet it still provides advanced features. It supports development team that needs built-in high availability as well as disaster recovery for their database.

### 4.8.1 Common PaaS Scenario

**Development framework**

PaaS provides a framework that developers can build upon to develop or customize cloud-based applications. Similar to the way you create an Excel macro, PaaS lets developers create applications using built-in software components. Cloud features such as scalability, high-availability, and multi-tenant capability are included, reducing the amount of coding that developers must do.

**Analytics or business intelligence**

PaaS provides Tools as services to allow organizations to analyze and mine their data, finding insights and patterns and predicting outcomes to improve forecasting, product design decisions, investment returns, and other business decisions.

**Additional** **services**

PaaS providers may offer other services that enhance applications, such as workflow, directory, security, and scheduling

### 4.8.2 Advantages

**Cost reduction**

One of the benefits of using PaaS is using virtual infrastructure instead of investing in physical infrastructures, virtual has both cost benefits and practical benefits. It is not required to purchase hardware or employ the expertise to manage it.

**Security**

Besides cost benefits, security is provided including data security as well as backup and recovery.

**Geographical Support**

Furthermore, development team can work together in various locations regardless of geographically barrier. Using the internet, developers spread across several locations can work together on the same application build.

# 5.0 Conclusion

In conclusion, developing web application is quite challenging as well as the deployment onto Azure. It requires deep understanding in utilizing the cloud services and setting up the application must be considerate in order to maintain the performance and reliability of the application. Hence, implementing traffic manager and auto scaling helps to maintain the request from users to improve the response time of the application thus improving the application performance. Furthermore performance test is also conducted to ensure that the application is able to run in heavy or certain numbers of concurrent users without severely affecting the performance of the web application.

# 6.0 Reference

Microsoft. 2017. *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 24 July 2018].

*BENEFITS OF PAAS IN CLOUD COMPUTING*. [ONLINE] Available at: http://www.dummies.com/programming/cloud-computing/hybrid-cloud/the-business-benefits-of-paas-in-cloud-computing/. [Accessed 24 July 2018].

Microsoft. 2017. *Autoscaling*. [ONLINE] Available at: https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling. [Accessed 24 July 2018].

Microsoft. 2017. *Manage an Azure Traffic Manager profile*. [ONLINE] Available at: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-manage-profiles. [Accessed 24 July 2018].

Microsoft. 2017. *How to configure auto scaling for a Cloud Service in the classic portal.* [ONLINE] Available at: https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-how-to-scale. [Accessed 24 July 2018].

Microsoft. 2018. *Load test with the Azure portal*. [ONLINE] Available at: https://docs.microsoft.com/en-us/vsts/test/load-test/app-service-web-app-performance-test?view=vsts. [Accessed 24 July 2018].

GitHub. 2015. *Automating code deployment with GitHub and Azure · GitHub*. [ONLINE] Available at: https://github.com/blog/2056-automating-code-deployment-with-github-and-azure. [Accessed 24 July 2018].

Microsoft Imagine Blog. 2015. *Using continuous integration with Azure & GitHub – Microsoft Imagine Blog*. [ONLINE] Available at: https://blogs.msdn.microsoft.com/microsoftimagine/2015/09/01/using-continuous-integration-with-azure-github/. [Accessed 24 July 2018].