# Acknowledgement

During the progress of the project, the developer faced a lot of challenges which was required to be solved. This is because the Microsoft Azure platform was an interesting platform to learn and there are too many features to explore within it. The developer owns thanks to many people whom without their support it would not have been possible to complete the project. Firstly, I would like to thank the module lecturer Dr. Kalai Anand Ratnam for his guidance and supervision throughout the construction and publishing process of this project on the cloud as the project could not be accomplished without him. Next, I would like to offer my gratitude to my course mates whom helped me during the difficulties encountered in coding part and shared valuable information about the project. Not forgetting, parents and my family members whom were giving me continuous morale support regardless of any situation. Moreover, the developer also would like to thank the University for providing the facilities required to conduct the project. Lastly, the developer would like to thank all the peers whom guided throughout the project.

Table of Contents

[Acknowledgement 1](#_Toc511260086)

[1.0 Introduction 3](#_Toc511260087)

[1.1 Project Background 3](#_Toc511260088)

[1.2 Objectives 4](#_Toc511260089)

[1.3 Scopes 4](#_Toc511260090)

[1.4 Requirement Specification 4](#_Toc511260091)

[2.0 Project Plan 5](#_Toc511260092)

[3.0 Design 6](#_Toc511260093)

[3.1 Design Considerations 6](#_Toc511260094)

[3.2 Architectural Design 8](#_Toc511260095)

[3.3 Modelling 9](#_Toc511260096)

[3.3.1 Use Case Diagram 9](#_Toc511260097)

[3.3.2 Use Case Description 10](#_Toc511260098)

[3.3.3 Sequence Diagram 13](#_Toc511260099)

[4.0 Implementation 17](#_Toc511260100)

[4.1 Application Development 17](#_Toc511260101)

[4.1.1 Developed Container Management System 21](#_Toc511260102)

[4.2 Azure Publishing 25](#_Toc511260103)

[4.3 Web App Plan 32](#_Toc511260104)

[4.3.1 Standard Service Plan 32](#_Toc511260105)

[4.4 Application Scaling 32](#_Toc511260106)

[4.4.1 Scale Up 32](#_Toc511260107)

[4.4.2 Scale Out 34](#_Toc511260108)

[4.5 Azure SQL Database 35](#_Toc511260109)

[4.6 Traffic Management (Optional) 36](#_Toc511260110)

[4.7 Azure Active Directory 38](#_Toc511260111)

[5.0 Test Plan and Test Discussion 41](#_Toc511260112)

[5.1 Functional Testing 41](#_Toc511260113)

[5.1.1 User Interface 41](#_Toc511260114)

[5.2 Unit Testing 43](#_Toc511260115)

[5.2.1 Test Plan 43](#_Toc511260116)

[5.3 Performance Testing 49](#_Toc511260117)

[6.0 Conclusion 53](#_Toc511260118)

[7.0 References 54](#_Toc511260119)

[8.0 Appendix 54](#_Toc511260120)

# 1.0 Introduction

## 1.1 Project Background

The Maersk Line company is considering at designing and developing a Container Management System (CMS) to implement a solution to maintain the containers which would lower down the overall supply chain cost and lead a prominent manner to manage the logistics in different geological location.

This company is the global container division and the biggest operating unit of the A.P. Moller-Maersk Group, a Danish business conglomerate which was founded in 1928. Maersk Line company is also the largest container shipping group in which holding out customers over 374 offices at 116 countries. Moreover, it also employs approximately 7,000 sea farers and 25,000 land-based people. This company also conduct over 600 vessels and has a capacity of 2.6 million TEU.

At first glance it would appear Danish shipping company, Maersk Line is already handling all the cargo it can manage as it operates in 100 countries and transporting goods around the globe. Hence, when Maersk identify that volume of most goods it was shipping had grown to full capacity, the company decided that empowering cloud-based solution would be a important factor of rectifying the current situation.

In order to support further business growth and the increase organizational flexibility, the company has decided to centralize all the data centers and servers room operating worldwide into a virtualized platform.

## 1.2 Objectives

* Design and Implement a single tenant web application hosted on Microsoft Azure as an App Service (Web App).
* Implement the web app and link with relational database.
* Determine web application performance with monitoring tools.
* Able to scope the solution to meet the require of demands during peak seasons.
* Source code to be situated in source control management services.
* To develop a cloud infrastructure that will be effective and cost efficient.

## 1.3 Scopes

* Indicate the understanding of cloud computing and its different forms and to what degree the Microsoft Azure fits within the cloud computing space.
* Analyze the Azure implementation environment.
* Design, Develop and Deploy the Container Management System in Azure.
* Architecturally design the Container Management System, utilizing Azure as the public cloud platform.

## 1.4 Requirement Specification

* From import, export and transshipment processing to gate operations.
* To be able to scale the solution to meet the requirement of demands on peak seasons.
* Improves profitability, decrease costs, increases productivity, exterminate errors and enhance resources to future-proof cargo handling business for better performance.
* Assurance & Reliability over Failover Management.
* Accurately allocates inbound containers to yard locations and plan outbound containers to individual hauler vehicles.
* Manage the entire booking process from schedule search to booking confirmation.

# 2.0 Project Plan

Project plan for the Maersk Line Application is to be able to log into the application without registration but using azure active directory. The functions of the system that required to be implemented are such as make bookings, maintain containers, warehouse, shipping, ship yards and workers thus, the users can be able to register bookings, shipments, containers, warehouses and ship yards to manage import and export. The needed requirement such as duration and budget has been planned earlier from the higher level to ensure the project can be implemented easily.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Task No. | Activity Name | Days | Start Date | End Date | Status |
| 1 | 1 | **Implementing Database** | 1 | 1 Dec 2017 | 2 Dec 2017 |  |
| 2 | 2 | **Web Design (Maersk Line)** |  | 10 Dec 2017 | 29 Dec 2017 |  |
| 3 |  | * **Container Management** | 3 |  |  |  |
| 4 |  | * **Warehouse Management** | 4 |  |  |  |
| 5 |  | * **Manage Bookings** | 4 |  |  |  |
| 6 |  | * **Shipment Management** | 3 |  |  |  |
| 7 |  | * **ShipYard Registration** | 3 |  |  |  |
| 8 |  | * **Partner Registration** | 3 |  |  |  |
| 9 | 3 | **Azure deployment** |  | 25 Feb 2018 | 28 Feb 2018 |  |
| 10 |  | * **Azure Configuration** | 4 |  |  |  |
| 11 | 4 | **Testing** |  | 2 Mar 2018 | 6 Mar 2018 |  |
|  |  | * **Functional Testing** | 1 |  |  |  |
| 12 |  | * **Unit testing** | 3 |  |  |  |
| 13 |  | * **Performance Testing** | 1 |  |  |  |
| 14 | 5 | **Documentation** |  | 20 Mar 2018 | 7 Apr 2018 |  |
| 15 |  | * **Technical documentation** | 12 |  |  |  |
| 16 |  | * **Final report** | 7 |  |  |  |

# 3.0 Design

## 3.1 Design Considerations

Before an implementation is to take place, there are many important steps that has to be considered. The main considerations would be determining the design, web components and the portal which must be used for the web portal development. “Design considerations contribute activities which has to be carefully planned that will aid to decrease time and effort required for implementation.”, (Oracle.com, 2016). The outermost component of the design consideration for the web portal development is the website structure. Website structure indicates to the architecture or appearance of the website which is quickly recognizable by consumers or clients. While developing web applications for business organization such as Maersk Line, the web structure must be given high prioritize. It is because the web portal is the tool which dominate the managements brand to its consumers. Website structure that includes efficiency of the web architecture to accomplish all the requirements. The web structure must be flexible and responsive to every different environment which the web portal will be performing on. To be responsive and flexible, the web structure must be identically updated along with latest technological aspects. The technological access for web implementation that includes the programming language, design frame and deployment portal. For the development and deployment of the container management portal, tools such as ASP.Net, HTML and Microsoft Azure portals were used.

Moreover, the next design consideration which has considered by developer is the UI & Technical Consideration. User Interface consideration displays the detailed graphical representation of the website structure. “User Interface (UI) focuses on anticipating interface components which users must need to access, understand and ease the website actions.”, (Usability.gov, 2016). For example, user interface elements which include dropdown list, navigation components and text fields. The user interface must be easy, persistent and purposeful by developer. For the development, of the Maersk Line Container web portal, the user interface is designed in a very plain way where it is understandable by all different users and user friendly. (Ibm.com, 2016)

Next is the technical consideration which focuses on the technical aspects of the system that will reflect of corporation business level, brand reputation and customer satisfaction. Technical factors cover page loading, site availability and coding. The developer has conducted only the coding which is relevant for the requirements of the client. The codes implemented must be easy to be understood and deploy by authorities. As the Container Management System is deployed in Azure Portal, the developer has to give importance to page loading and availability. The developer considered to design the website in such way it is continuously available on the cloud infrastructure to solve handling of high loads during peak hours. There was minimal amount of information located on the homepage of the application and users can direct to other pages from there. This will to balance the load time and web availability.

The most essential consideration for the web portal development is the platform that is going to be used for the deployment. Unlike other applications, the container management web application cannot be deployed in normal browsers. Maersk Line has decided to approach the cloud deployment. The developer used the Microsoft Azure platform to deploy the implemented application. “Microsoft Azure is a growing platform with collection of integrated cloud services, databases, storage and web.”, (Azure.microsoft.com, 2018). The usage of Microsoft Azure will strengthen corporations to experience cloud computing in their management application. Maersk Line will have an advantage of using open and effective platform, big data protection to make faster decisions using the cloud platform.

## 3.2 Architectural Design



*Figure 1 shows Architectural Design of Maersk Line*

## 3.3 Modelling

### 3.3.1 Use Case Diagram



*Figure 2 shows Use Case Diagram for Maersk Line Application*

### 3.3.2 Use Case Description

|  |  |
| --- | --- |
| **Use Case ID** | UC1 |
| **Name** | Register Partner |
| **Summary** | Able to add partner details. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete adding partner details successfully. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Fill in partner details | | 1. Edit the partner details. | | 1. Delete partner details. | |
| **Alternative Sequence** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC2 |
| **Name** | Manage Schedule |
| **Summary** | Able to add schedules. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete schedule management. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Add schedule. | | 1. Edit schedule. | | 1. Delete scheduling time. | |
| **Alternative Sequence** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC3 |
| **Name** | Manage Shipment |
| **Summary** | Able to manage shipment status. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete shipment management. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Add shipment status | | 1. Edit shipment status. | | 1. Delete shipment status. 2. Track Shipment. 3. Manage Shipment Cost. | |
| **Alternative Sequence** | - |

|  |  |
| --- | --- |
| **Use Case ID** | UC4 |
| **Name** | Manage Fleet |
| **Summary** | Able to manage fleet. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete fleet management. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Add fleet details. | | 1. Edit fleet details. | | 1. Delete fleet details. | |
| **Alternative Sequence** | - |

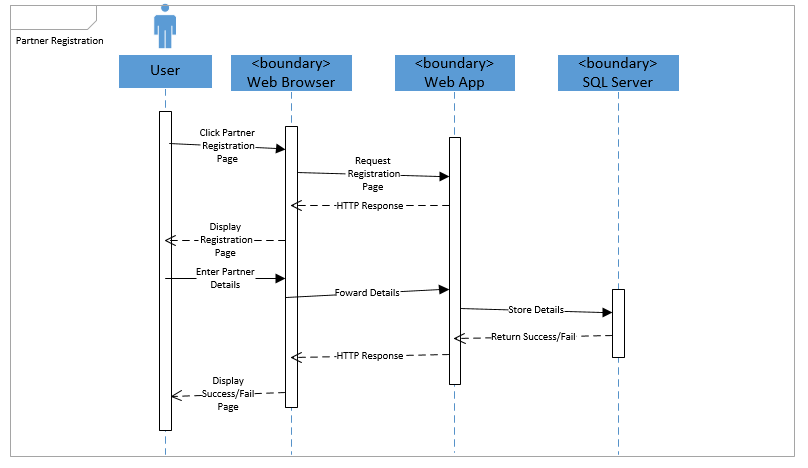
|  |  |
| --- | --- |
| **Use Case ID** | UC5 |
| **Name** | Manage Booking |
| **Summary** | Able to make shipment booking. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete bookings successfully. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Add booking details. | | 1. Edit booking details. | | 1. Delete booking details. | |
| **Alternative Sequence** | - |

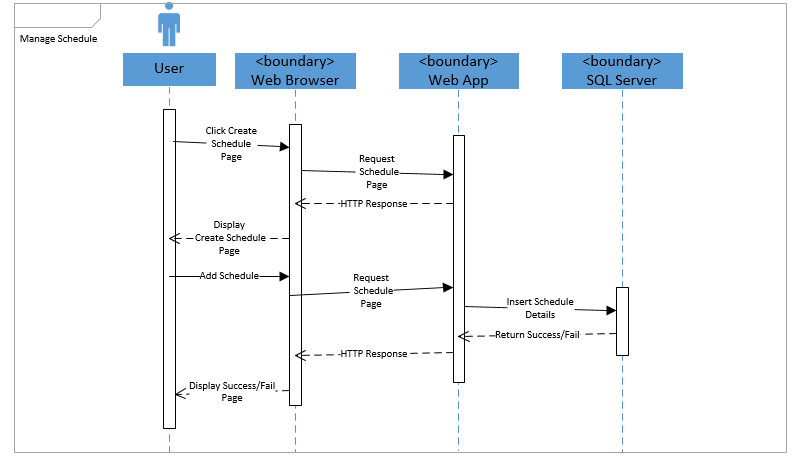
|  |  |
| --- | --- |
| **Use Case ID** | UC6 |
| **Name** | Manage Warehouse |
| **Summary** | Able to manage warehouse activity. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete warehouse management activity. |
| **Actors** | Admin |
| **Main Sequence** | |  | | --- | | Method | | 1. Add warehouse items. | | 1. Edit warehouse items. | | 1. Delete warehouse items. | |
| **Alternative Sequence** | - |

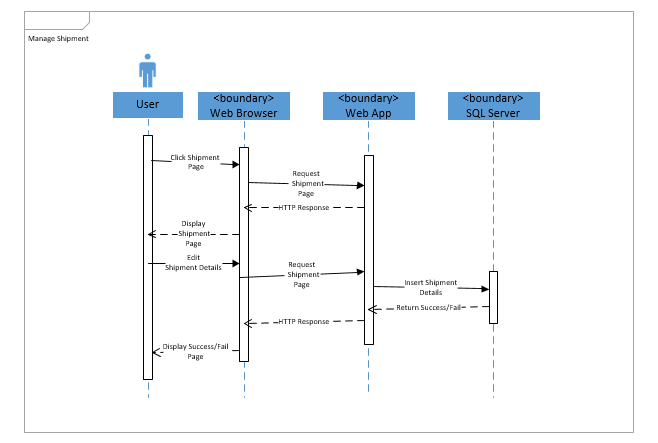
|  |  |
| --- | --- |
| **Use Case ID** | UC7 |
| **Name** | Generate Payment |
| **Summary** | Able to make payment. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete payment. |
| **Actors** | Customer |
| **Main Sequence** | |  | | --- | | Method | | 1. Add payment. | | 1. Edit payment details. | | 1. Delete payment details. | |
| **Alternative Sequence** | - |

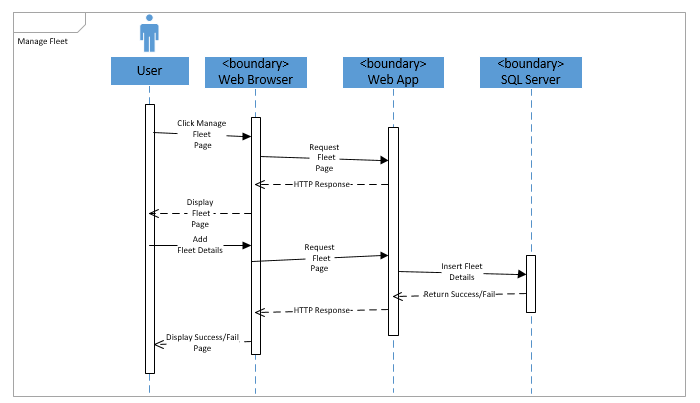
|  |  |
| --- | --- |
| **Use Case ID** | UC8 |
| **Name** | Manage Customer |
| **Summary** | Able to add customer. |
| **Priority** | High |
| **Pre-condition** | Must access to Home Page |
| **Post-condition** | Complete adding customer |
| **Actors** | Partner |
| **Main Sequence** | |  | | --- | | Method | | 1. Add customer details. | | 1. Edit customer details. | | 1. Delete customer details. | |
| **Alternative Sequence** | - |

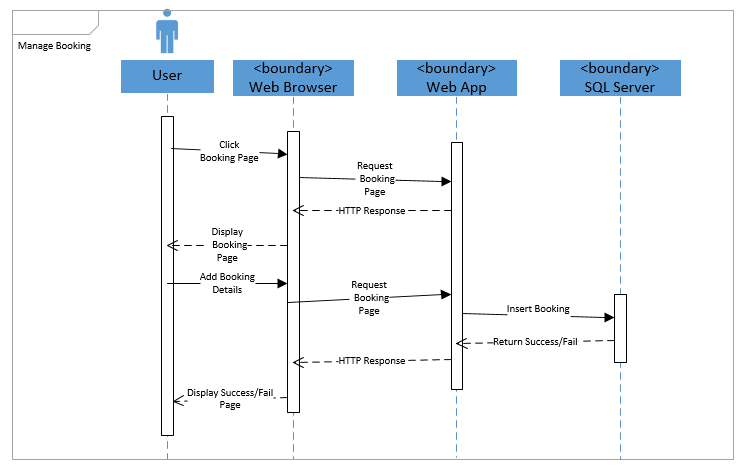
### 3.3.3 Sequence Diagram

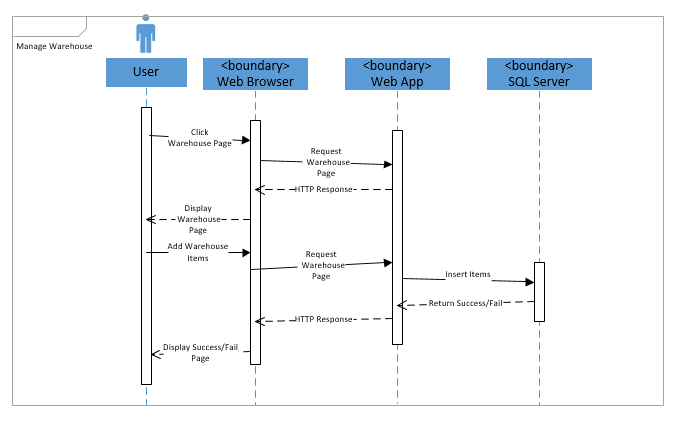


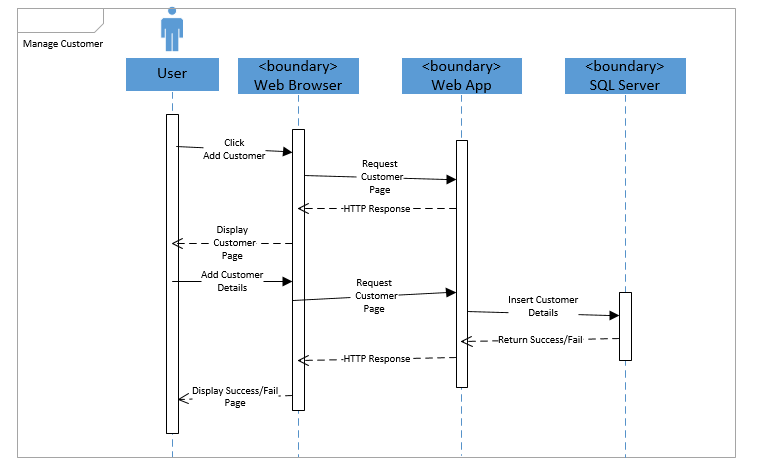








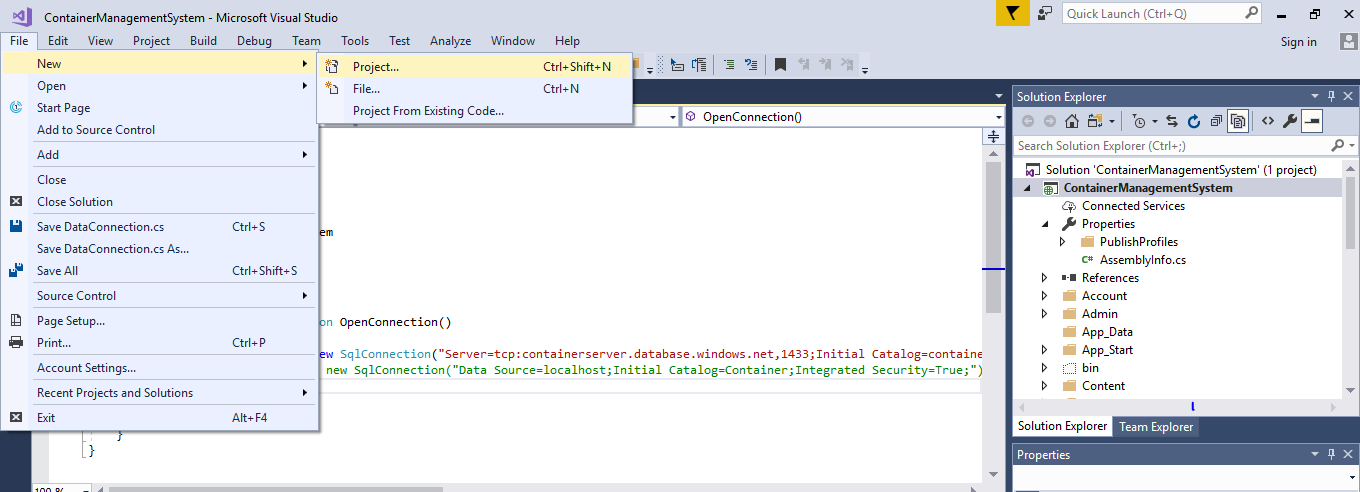




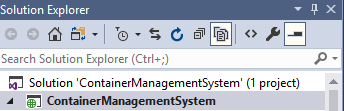
# 4.0 Implementation

## 4.1 Application Development

The development of the Container Management System (CMS) web application was done in IDE Visual Studio 2012. MVC structure was used to develop the user interface which is divided into Model, Controller and View. The application developed was configured based on Microsoft Azure deployment platform. Firstly, the developer creates the project and named it as Container Management System. The other layers were built according to its requirement.

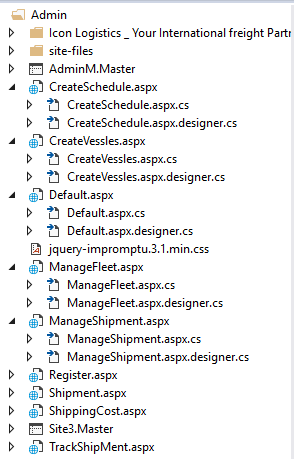


*Figure 3 shows Creating Container Management System on Visual Studio.*



*Figure 4 shows Project Name.*

After the project is created, the developer will create several data access layers. As representation of data access layers, various data models will be developed based on the Container Management System requirements. Data models represents the classes used in the application development. Besides that, the developer also used Entity Framework which can fetch, store, update and delete tables data.



*Figure 5 shows the MVC Framework.*

UI and navigation are the next process conducted by the developer once all the data models are created. UI and navigation refer to the user interface designs and the navigation controls in the Container Management application. The developer has selected ASP.NET as the major programming language for this project development. HTML mark up also been provided once the project is completed.



*Figure 6 shows the Contact Page Code.*



*Figure 7 shows the Login Page Code.*



*Figure 8 shows Manage Shipment Page Code*

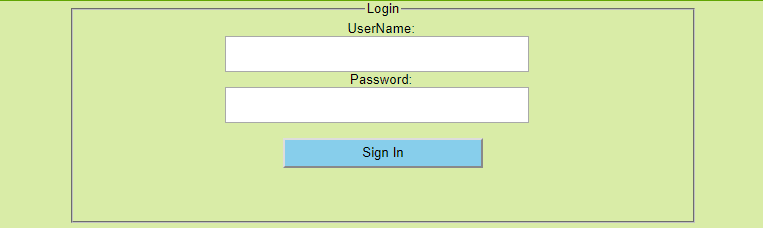


*Figure 9 shows Make Payment Page.*

### 4.1.1 Developed Container Management System



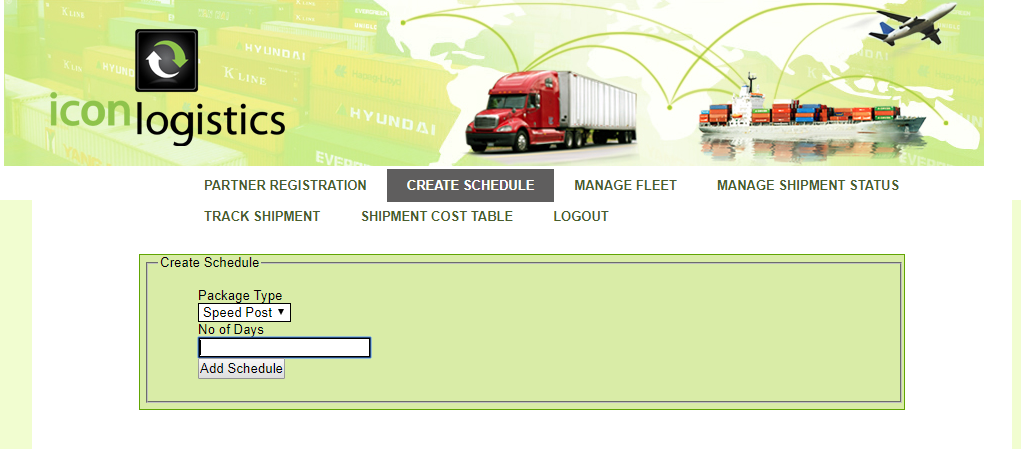
*Figure 10 shows the Home Page of Container Management System.*



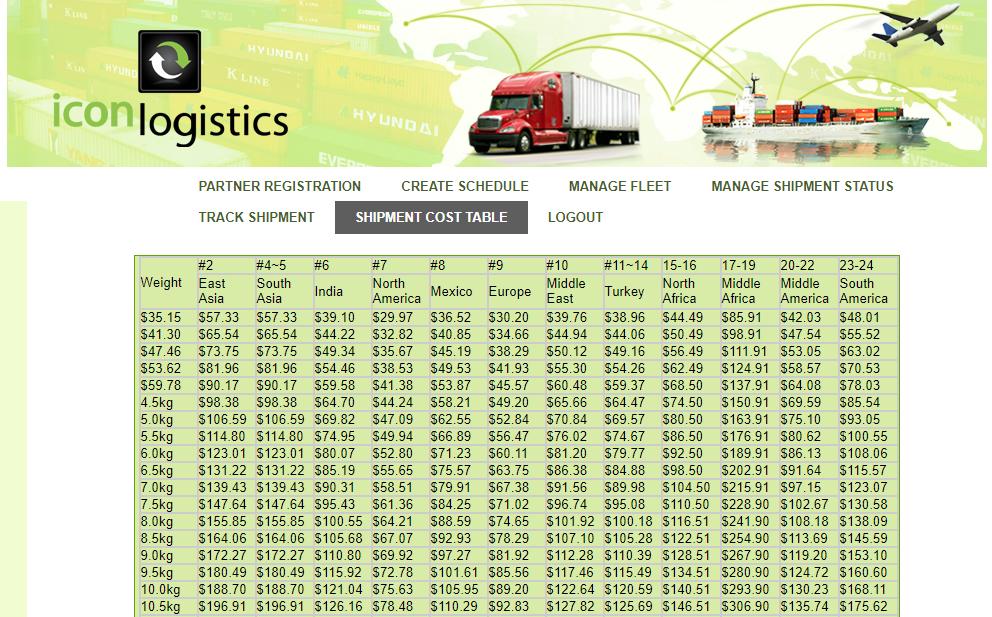
*Figure 11 shows the Login Page of Container Management System.*



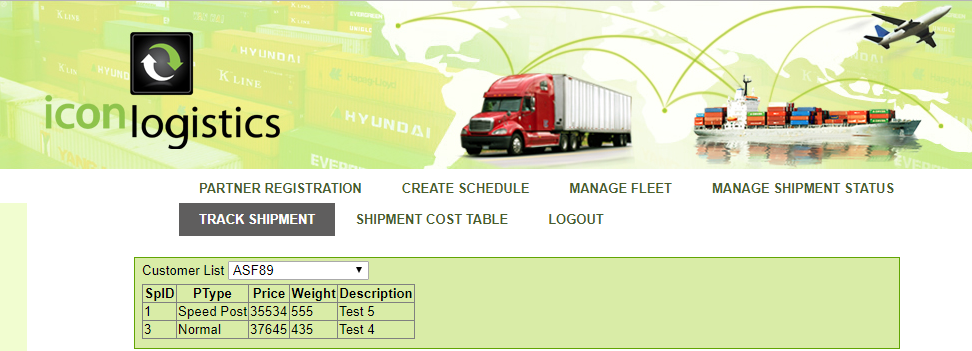
*Figure 12 shows Manage Shipment Page.*



*Figure 13 shows Create Schedule Page.*



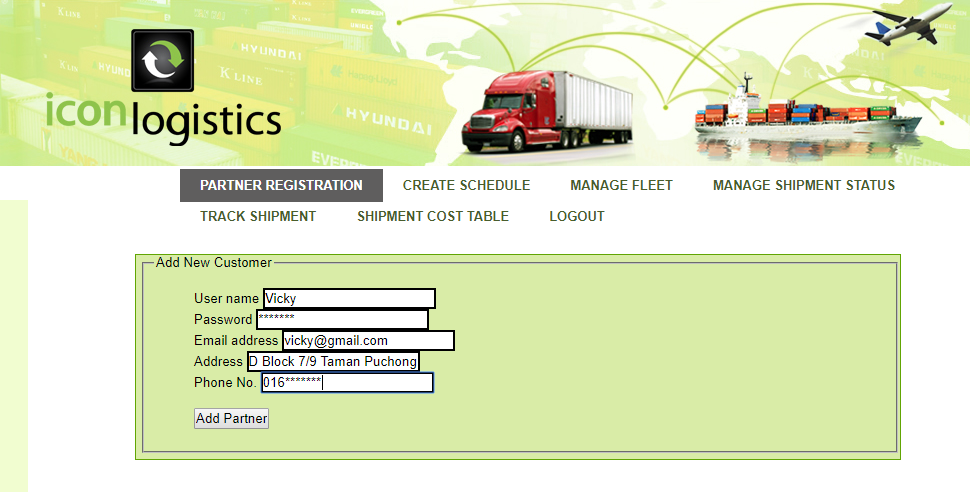
*Figure 14 shows Shipment Cost Table.*



*Figure 15 shows Shipment Tracking Page.*



*Figure 16 shows Manage Fleet Page.*



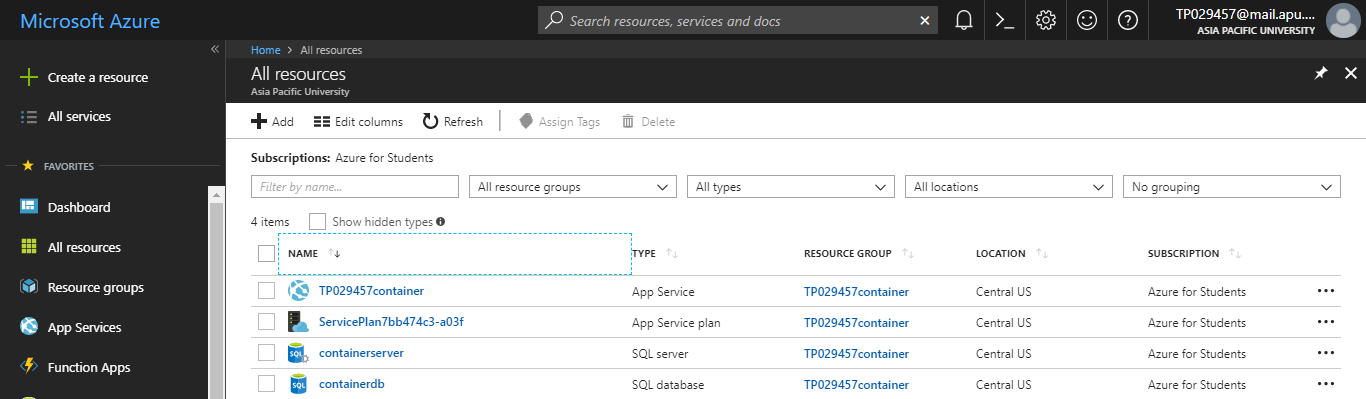
*Figure 17 shows Partner Registration Page.*

## 4.2 Azure Publishing

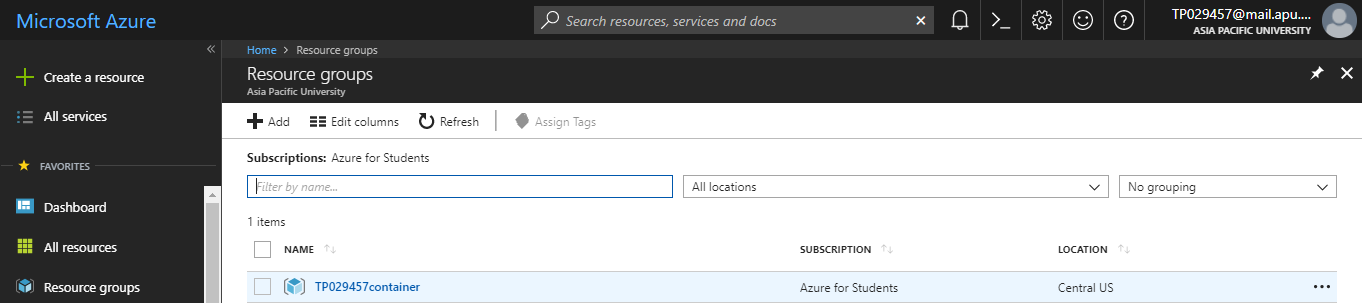
The developed Container Management application will be deployed on the cloud using the Microsoft Azure Platform. The figures below will periodically show the steps required to deploy the developed system on the cloud. Azure provides the same service and technologies to a global amount of developers and IT professionals. Moreover, Microsoft Azure also supports the widest selection of systems, frameworks, tools, databases, devices and programming languages. Besides that, Azure’s “pay as you go” services can also scale up or out to match demand. Hence, users only pay for what they use. For Container Management System application, it has 3 users which is the admin, partner and customer. The developer will publish the web application to the cloud to make the system publicly available.

**Step 1: Create Resource Group**

Before deploying the web application, resource group should be created to display which particular resource group the application services must be deployed into. App services which will be created later will be deployed into the resource group too. Cloud provider can click on the resource group icon and will be directed to the resource group page. After that, provider can press on add and add the resource group. Resource groups will enable Container Management System to manage all its resources within the application together which is enabled by Azure Resource Manager. Resource manager which will allow Container Management System to group multiple resources as a logical group that serves as a lifecycle frontier for all the resource contained within it. Hence, a group will consist of resources related to a specific web application. The figure below displays all the resources which contains a website resource which host Container Management System public website, SQL database which stores relational data used by the website, a storage account which stores non-relational assets.



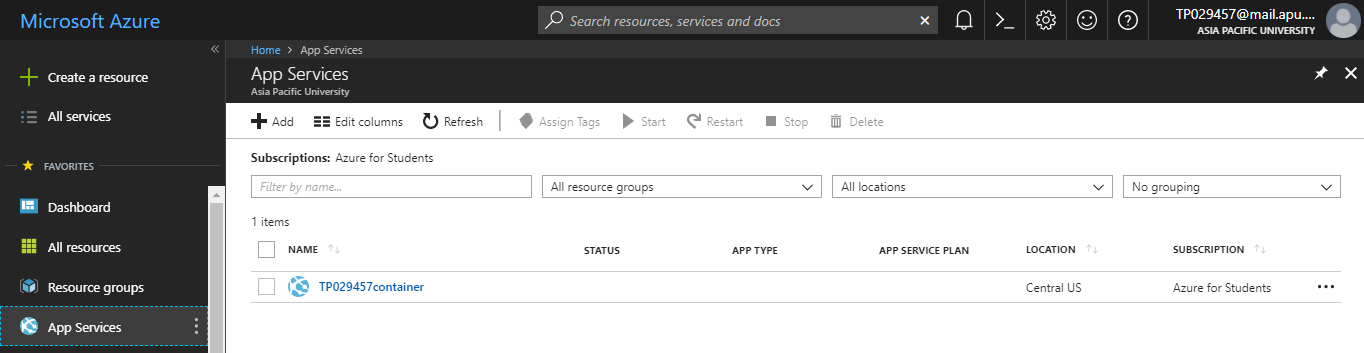
*Figure 18 shows Resource Group of Container Management System.*



*Figure 19 shows Resource Group that contains Storage Account.*

**Step 2: Create Web App Services**

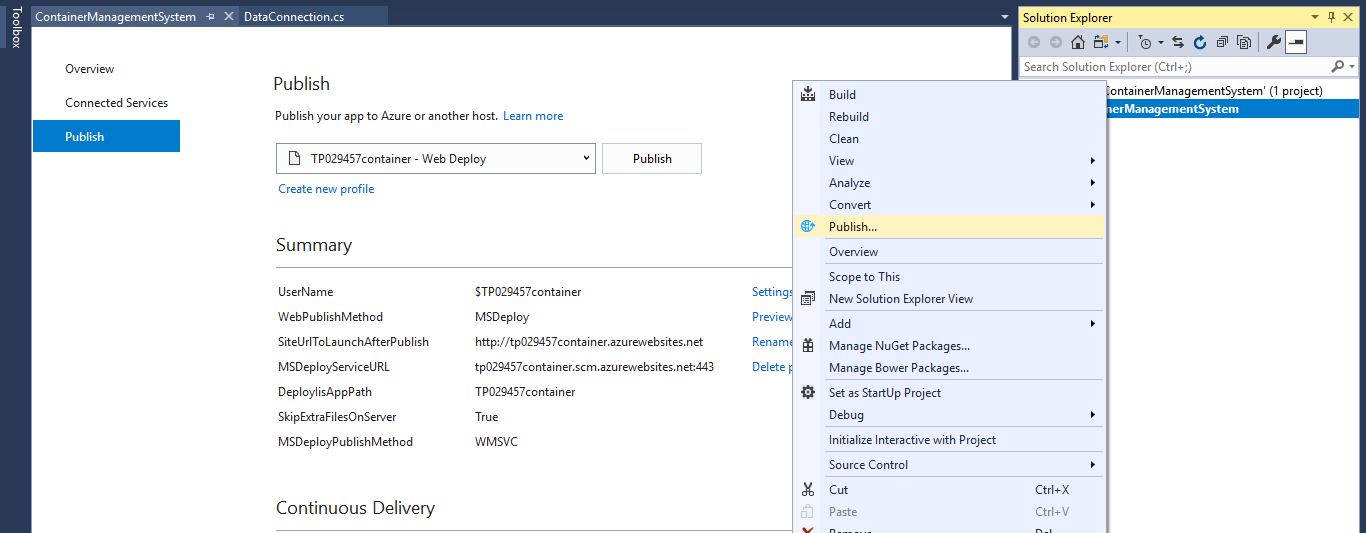
Once resource group has been created, cloud provider should create the app services to deploy the particular project into the resource group. Cloud provider can choose the app services icon and will be directed to main app service page. Then, cloud provider can click ass to add new app services. Microsoft Azure web sites delivers flexible and secure development, deployment and scaling options for various sized web application. There are many advantages of web app service such as provision as well as fast deployment, great benefit for visual studio developers, rapid method for cloud development and flexible as well as open for anyone.



*Figure 20 shows app service page.*

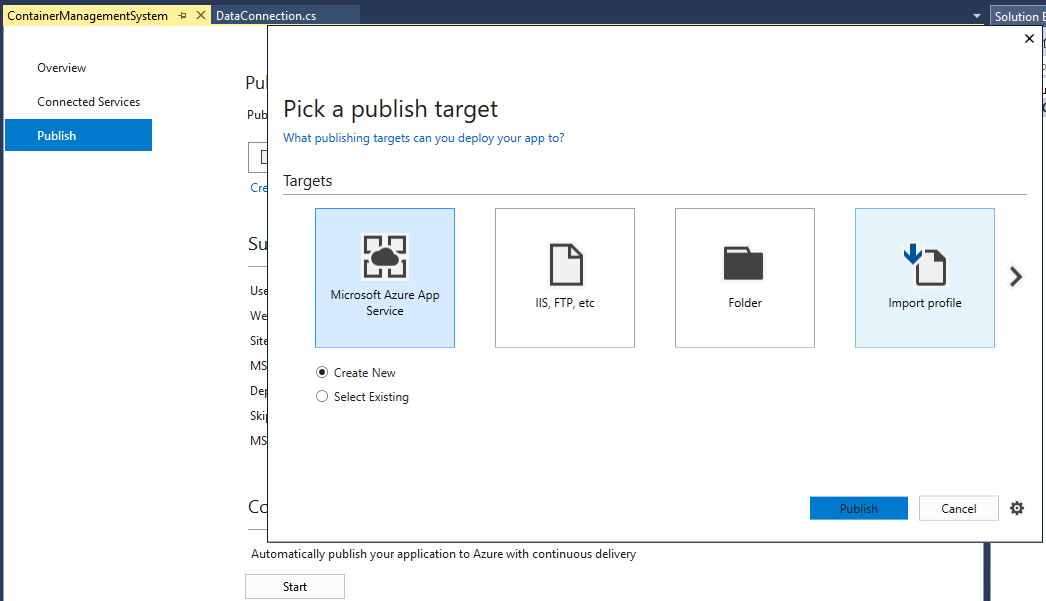
**Step 3: Deploy from Visual Studio to Azure**

Resource group and app services are created to directly deploy application from Visual Studio to Azure platform. The complete developed Container Management web application will be prepared for deployment. After development is completed fully then application is ready for deployment which means can be published to Azure portal. Cloud provider must be first sync Visual Studio to an active Azure account. Then, cloud provider can right click on the solution file and select **publish**.

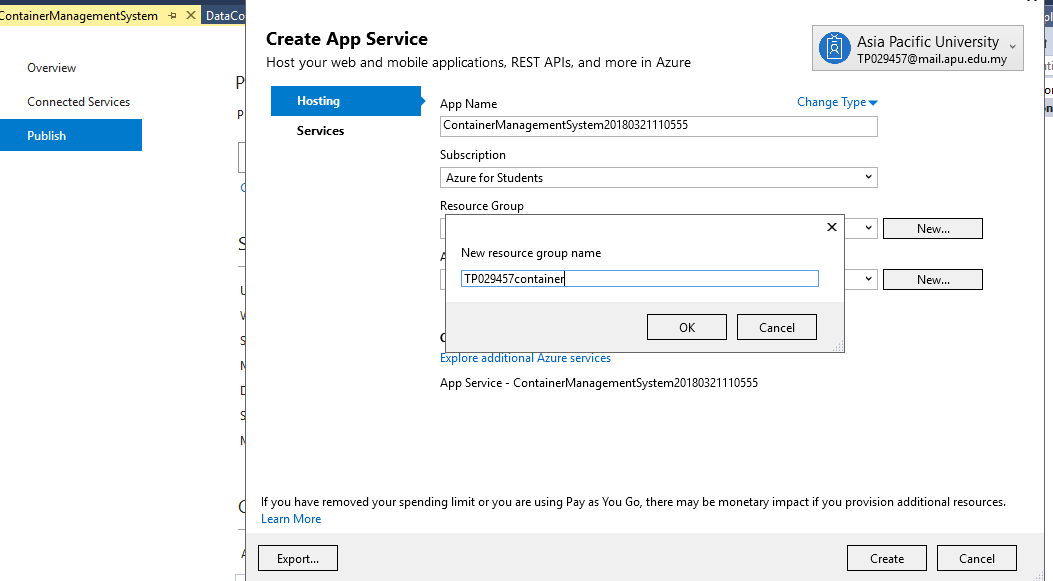


*Figure 21 shows select* ***Publish*** *to deploy application.*

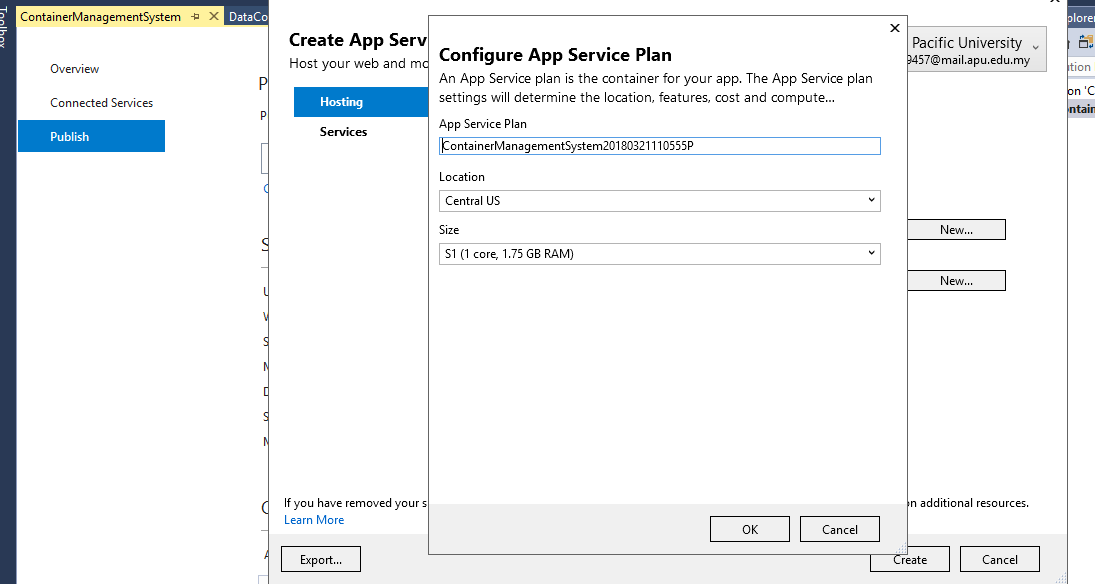
The Container Management System web application was published by using the Azure automatic deployment options. The option will allow for every change pushed to a branch in GitHub to be deployed directly to the Azure servers. After cloud provider clicks on publish, the page will be directed to the publish web page in which all the required information that needed to deploy the application has to be selected. Cloud provider can choose to deploy on Microsoft Azure app services and select next, where it will be directed to connection page to select the app services and resource group connection. Finally, after all the settings are completed, cloud provider must be able to preview files to be deployed and choose to publish to publish the application on the azure platform.



*Figure 22 shows Configure Resource Publish.*



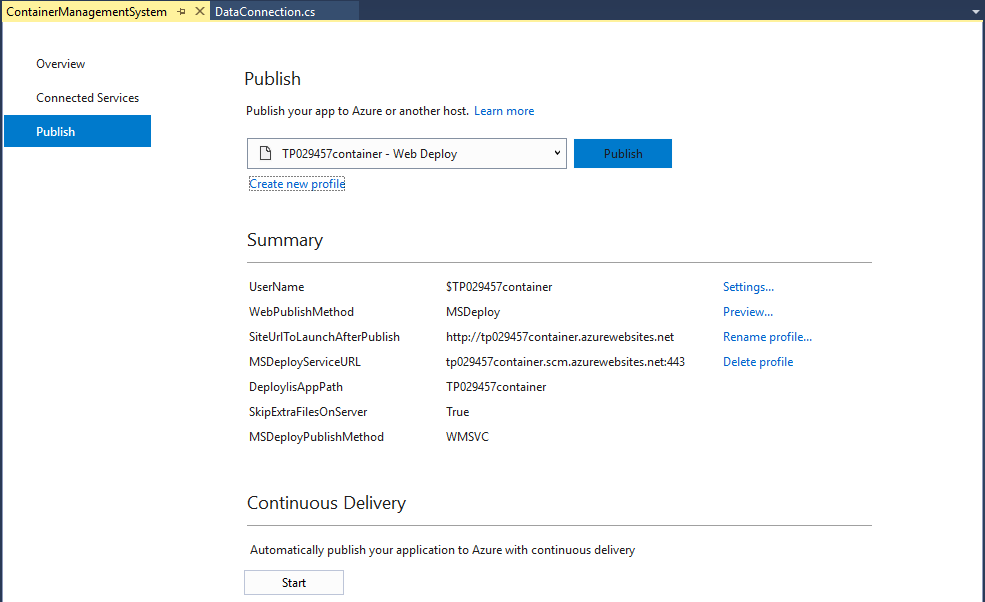
*Figure 23 shows Create App Service page.*



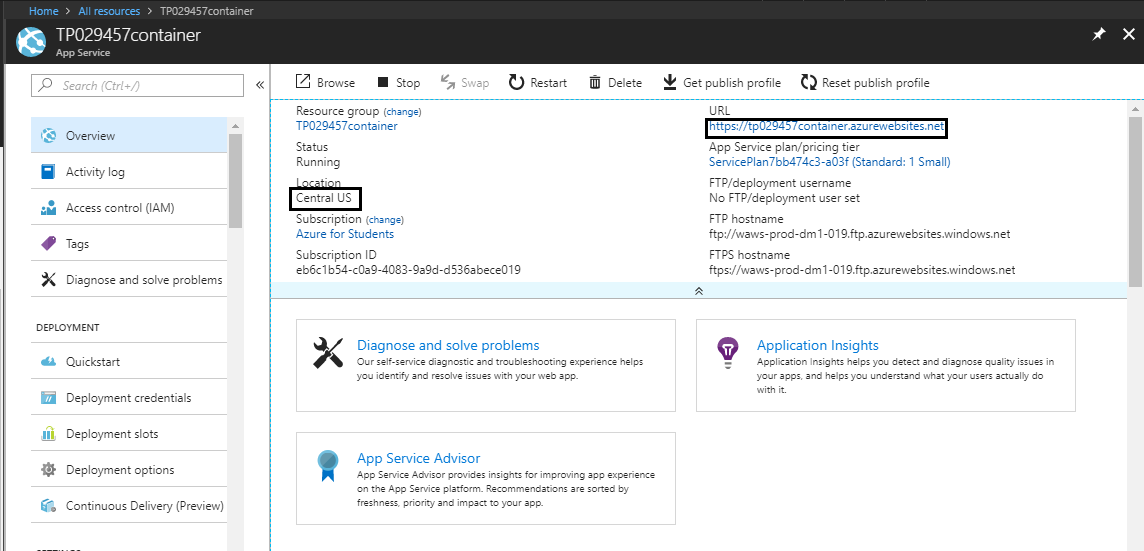
*Figure 24 shows Configure App Service plan page.*



*Figure 25 shows Create App Service page.*



*Figure 26 shows Publishing Container Management System App on Azure.*

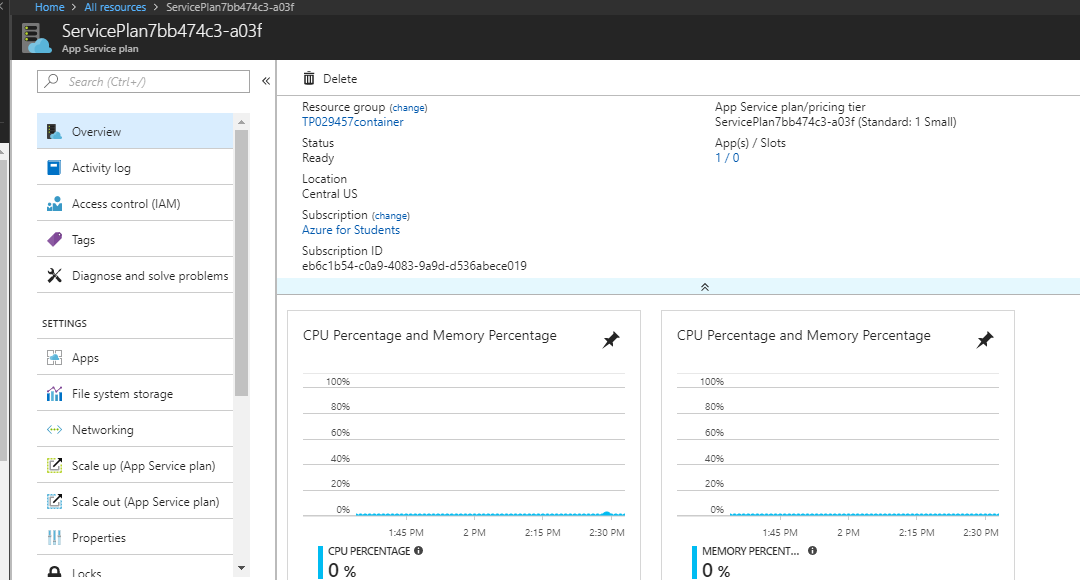


*Figure 27 shows Resource Group holds web app service on Central US Region.*

## 4.3 Web App Plan

### 4.3.1 Standard Service Plan

Web App Plan is constructed for API, Web apps and Mobile productions. There is no restriction on the amount of apps or domains which can be hosted by using the Standard service plan. The price will be according to the size and number of occurrence run. It is created in network load balancing which supports automatically circulate traffic beyond the instances. Besides that, the standard plan service also comprises built in auto scale support which can automatically modify the amount of instances operating to match the traffic requirement. This also comprise of build in backup support. (Azure.microsoft.com, 2018)

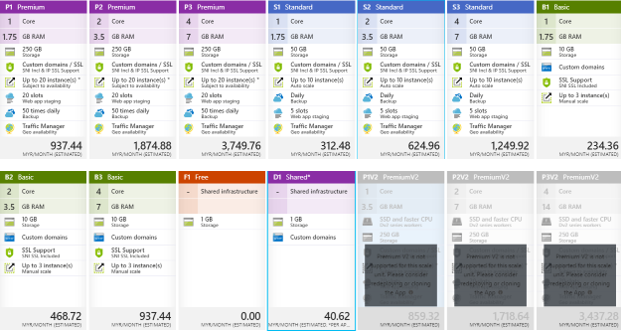


*Figure 28 shows App Service Plan of Container Management Application.*

## 4.4 Application Scaling

### 4.4.1 Scale Up

Azure cloud platform serves the web services for auto scaling that the resource can be actively allocated according to the requirements to meet the performance needs. This operation defines the azure web sites cloud corresponds of moving a non-cloud web site to a larger environmental server. Moreover, application scaling can be conducted practically all the sites without uncertainty on the indications of more instances of data consistency (Tardif, 2013). This process is very important as when the application accelerate the tasks, hence further resource is required to manage the desired performance level. The advantage of this is it does not require a dedicated personal to always check the server performance.



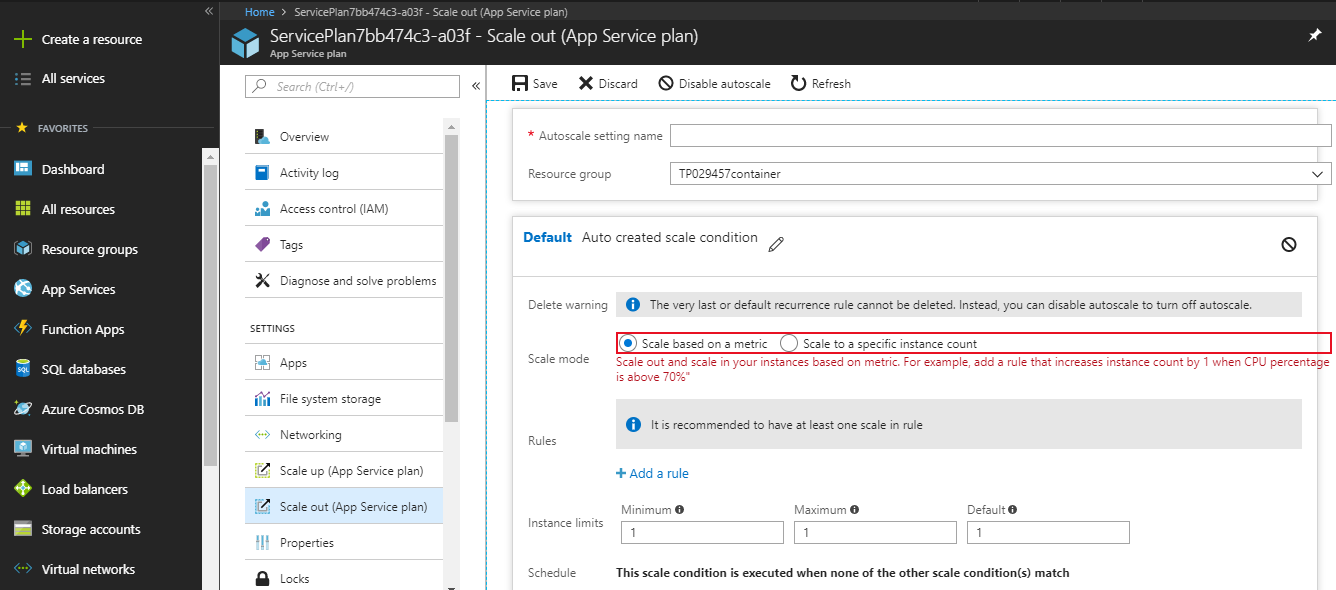
*Figure 29 shows Available Web App Plan for CMS.*

Every application which is deployed will use minimum a standard plan to enable the fundamental functions to the company such as custom domains. The Container Management System web application will be using the S1 standard app service plan as its price is comfortably affordable. The comparison between standard plan and basic plan will indicate the reason to choose the S1 tier app service for this application deployment. The standard plan provides with 1 Core 1.75 GB Ram server which will be more than enough for the expected traffic.

The major point to select the standard plan is because it provides 50 GB storage where by it is five times more than the storage from basic plan. Hence, it could manage a bigger number of transactions process beyond a provided period. Moreover, the custom domain that has been provided in both the basic and standard plan but in standard plan it offers additional IP SSL Support where this is an advantage. Furthermore, the auto scale is also a major point for selecting standard plan as it offers up to 10 instances compare to basic plan which offers 3 instance plans. This is very useful when the Container Management System application is expected to explore by different customers from all around the world. Besides that, the other extra features offered by only standard plan such as daily backup of application settings and data to make sure data consistency. Standard plan also provides 5 deployment slots in which it grips the contemporary deployment restore point if any unprevent situation happens. Finally, it allows the traffic manager to manage the connection between the client and server to make sure the stability of interaction by allowing them to be connected to the server that is nearby from the region.

### 4.4.2 Scale Out

Scale out operation is the corresponding of creating multiple type of Container Management System app and adding a load balancer to present the demand between them (Tardif, 2013). Meanwhile, when the Container Management System is scale out in Azure web sites hence there us no need if configuration of load balancing severally since its already been given by the platform itself.

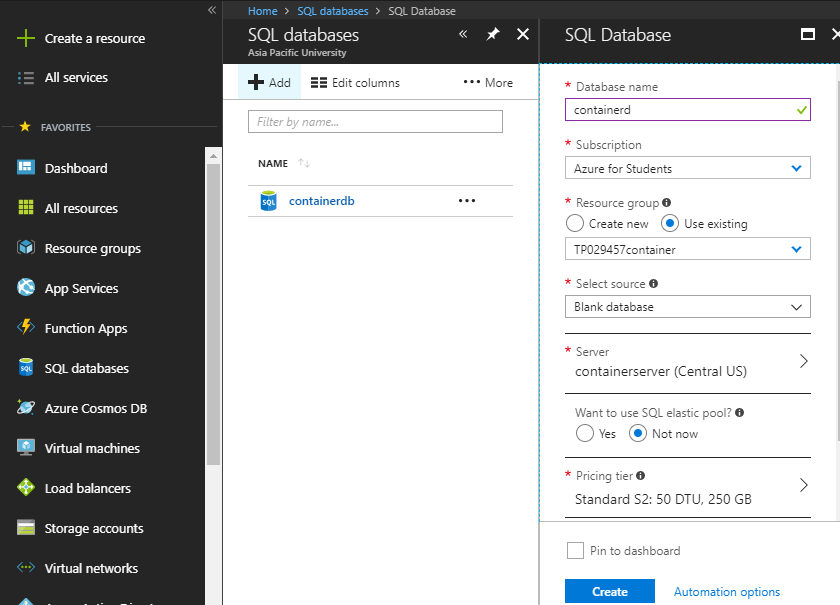


*Figure 30 shows Creation of Scale condition.*

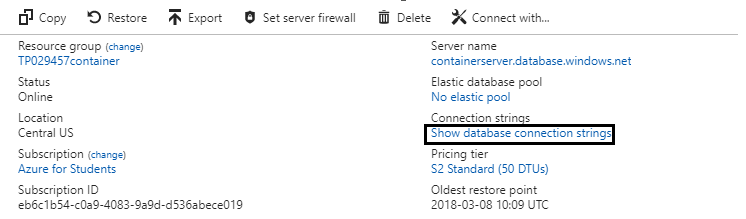
The Auto Scale created which holds the condition that will increase the count of instance by 1 when CPU percentage or Memory Percentage is above **70%**. Azure Web Sites will allow using the instance count slider to modify the instance count that is between 1 and 6 in shared mode and 1 and 10 in reserved mode. This will generate multiple running copies of Container Management System web application as well as manage the load balancing configurations required to circulate the incoming requests across every instance.

## 4.5 Azure SQL Database

SQL Azure database is a cloud based relational database service which was built on SQL server technologies and runs in Microsoft data centers on hardware which hosted as well as managed by Microsoft for application developers that lets them scale on the fly without downtime and efficiently deliver the Container Management System application. The built-in advisors will adapt by learning Container Management System application’s exclusive characteristics and will maximize reliability, performance and data protection. This requires information such as the database name, subscription, resource group, source, pricing tier and server. The database name must be unique than the databases in the Azure. The server will be created too to host the database in a region along with username and password to explore the server. The database performance configuration will be evaluated in detail to obtain the maximum database performance with DTU. Finally, creating of geo-replication database in various region hence there will be a back up if it fails to function. (Azure.microsoft.com, 2018)



*Figure 31 shows Configure SQL Database Server.*



*Figure 32 shows Post Configure SQL Database Setup.*

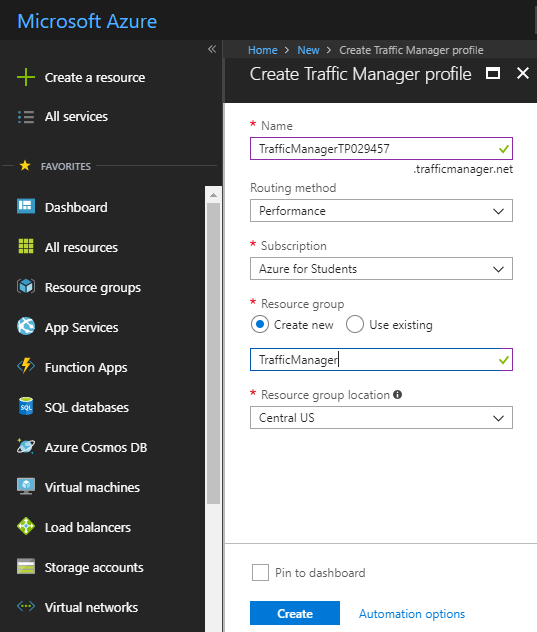
Once the creation of the database is done, the web application developed needs to be linked to the database. Hence, this is done by getting the connection string as seen above, copy and pasting the string to the webconfig file.

Connection String:

Server=tcp:containerserver.database.windows.net,1433;InitialCatalog=containerdb;Persist SecurityInfo=False;UserID={your\_username};Password={your\_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

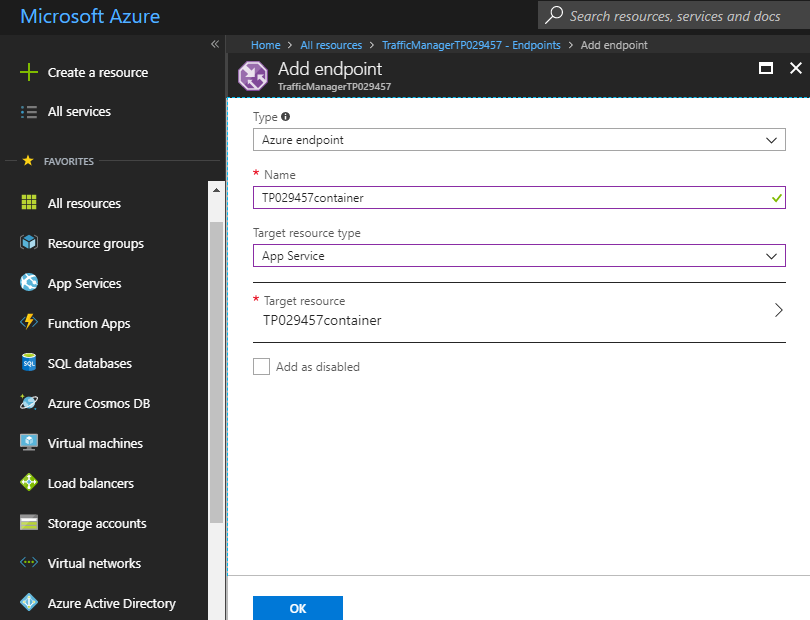
## 4.6 Traffic Management (Optional)

Azure Traffic Manager allows for distribution of user traffic for services endpoint for various data centers. Service endpoints supported by traffic manager which comprises of Azure VM’s, web applications and cloud services. Moreover, traffic manager also can be used with external non-Azure endpoints. This helps to decreases downtime and improves responsiveness of applications by routing incoming traffic across multiple deployments in various regions. Traffic manager is also used to improve application availability with automatic failover and enable business continuity as well as recover from disaster. Diagram below shows the creating traffic manager tab where the developer will need to key in the required information to successfully create a traffic manager. Traffic Manager also uses the Domain Name System (DNS) to direct client requests to the most appropriate endpoint based on a traffic – routing method and the health of the endpoints. (Azure.microsoft.com, 2018)

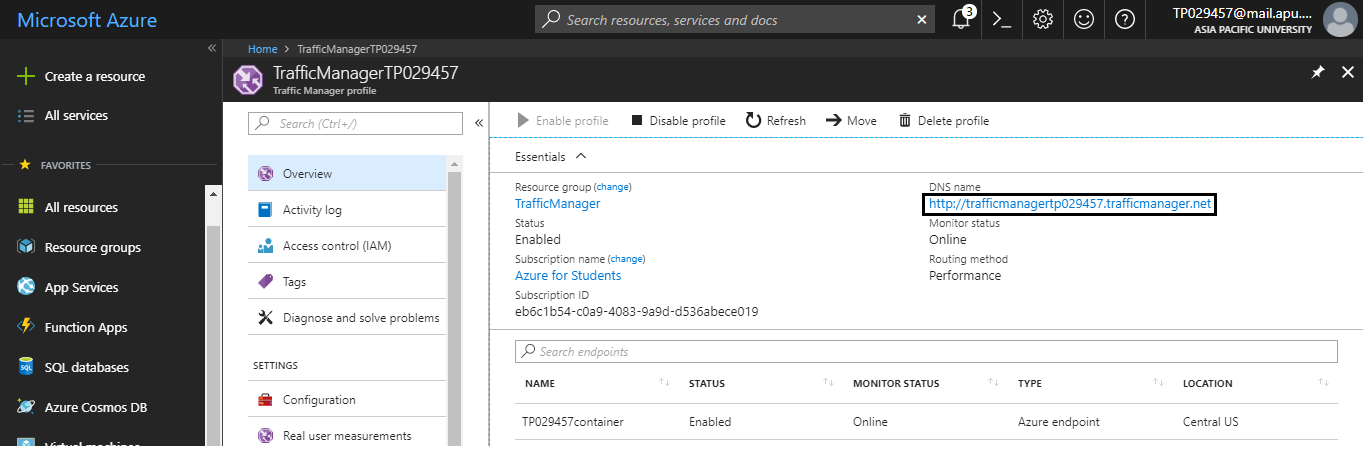


*Figure 33 shows Create Traffic Manager Profile.*

After creating traffic manager successfully, it is required to add end points into the traffic manager which will be shown in the diagram below.



*Figure 34 shows Adding End Points in the web traffic manager.*



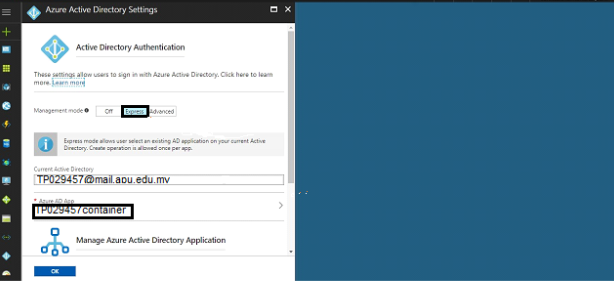
*Figure 35 shows Overview of Traffic Manager.*

Above shows the traffic manager profile after creating the endpoints for Container Management Application. Traffic Manager routing mode will be set to performance. This way the traffic will be controlled according to performance of the web application to the user. The highlighted DNS link was saved as well as use to test the routing of the traffic manager. The DNS link would be used to access the system from the internet.

DNS Link: <http://trafficmanagertp029457.trafficmanager.net>

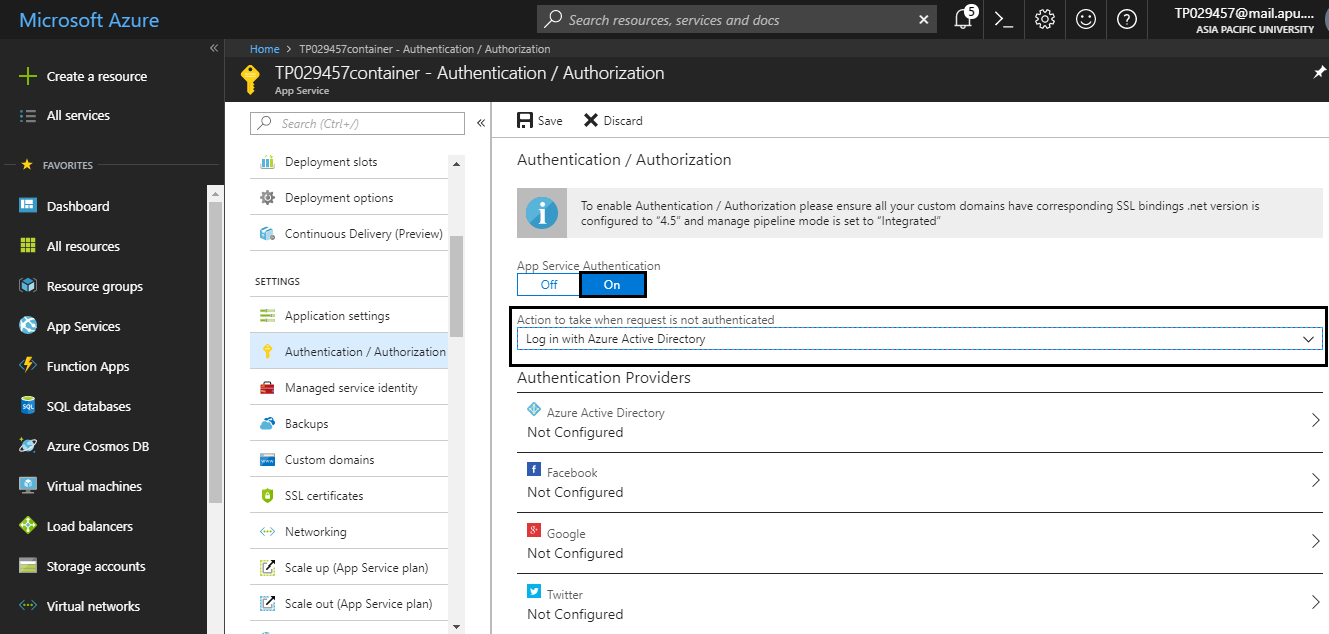
## 4.7 Azure Active Directory

Azure Active Directory service provided by Microsoft facilitate enterprise mobility and covers a broader aspect of domain services, thus gives identity protection over multi authentication (MFA), secure installation media, dynamic MAPI ID support and data mining tools, user location and risk level as well as alerts conditional access policies regard on device health. As of this project, the developer only selects the security authentication service which provided by the Active Directory. The identity management makes it simple for admin to define the scope of the authority and authenticate that the login of a user is from the domain of the company. (Azure.microsoft.com, 2018)



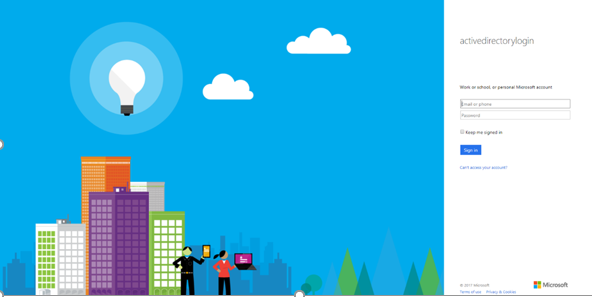
*Figure 36 shows Setting up Authentication/Authorization for web application.*

Developer enters this page by clicking the Azure Active Directory in the authentication/authorization page of the web application to set up authentication for the project. The management mode is set at **Express** and the Azure AD App is set to TP029457container which is the application name.



*Figure 37 shows Setting up Authentication/Authorization for web application.*

The above shows authentication/authorization page after configuring the azure active directory. Click on the authentication and authorization of the selected application to set up authentication for the web application. Before that, make sure that the application service is turned on. The developer has to change the **“Action to take when request if not authenticated”** to **“Log in with Azure Active Directory”.** Then in the application registration page it shows the owner which is [TP029457@mail.apu.edu.my](mailto:TP029457@mail.apu.edu.my) as the default directory that is able to gain access into the web application TP029457container.



*Figure 38 shows Setting up Authentication/Authorization for web application.*

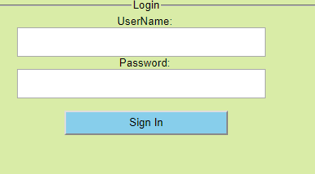
After completing all the steps in the authentication/authorization for the web application then once the application is clicked the following page should be directed to user before it can be accessed.

# 5.0 Test Plan and Test Discussion

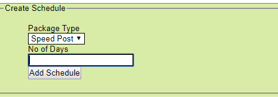
## 5.1 Functional Testing

Functional Testing is a type of software testing in which the system is tested contra to the functional requirements or specifications. The functions or features are tested by provide input and examine the output. This testing mainly ensures whether the requirements are properly satisfied by the application. Functional testing does not concern on how processing occurs, but the results of processing is very essential. Black Box Testing method is used during functional testing whereby the internal logic of the system tested will not known to the tester. Functional testing is usually performed when the levels of system testing and acceptance testing. In this Maersk Line project, all the application functionality is being tested by the developer with proper input and verifying the output. The developer also compares the actual results with the expected results. This includes checking the user interface and database of the application which was done manually. (Software Testing Fundamentals, 2018)

### 5.1.1 User Interface



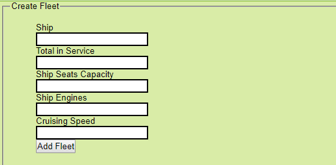
*Figure 39 shows User Login Page*



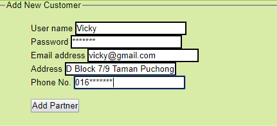
*Figure 40 shows Add Scheduling Page*



*Figure 41 Shipment Tracking Page*



*Figure 42 shows Add Fleet Page*



*Figure 43 shows Add New Customer Page*

## 5.2 Unit Testing

Unit testing defined as a level of software testing whereby elements of a software are tested. The function of the unit testing is to validate all the unit of the software performs as designed. (Software Testing Fundamentals, 2018) The test case results will be noted in the test plan template. The table below shows the test plan implemented and the results for the unit tests. In all the development projects tests will determine the bugs in the system which required to be fixed but since in this project the latest version is already fully fixed so the results of all the tests done are passed. All the tests are done locally except stated that it can be performed on the deployed Azure version by changing the localhost to the hosted domain.

### 5.2.1 Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Login** | | | |  |
| **Test ID:** | **UT1** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in valid username and password. | | Login successful | As Expected | PASS |
| Key in valid user name and invalid password. | | Display error message, “Invalid Password” | As Expected | PASS |
| Key in invalid user name and valid password. | | Display error message, “Invalid Username” | As Expected | PASS |
| Blank information for both user name and password. | | Display error message, could not login. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Schedule** | | | |  |
| **Test ID:** | **UT2** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in all valid information such as package type and number of days. | | Add Schedule successfully. | As Expected | PASS |
| Key in invalid information for any of the fields. | | Display error message, could not create schedule. | As Expected | PASS |
| Blank information for all the fields. | | Display error message, could not create schedule. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edit Shipment Information** | |  |  |  |
| **Test ID:** | **UT3** |  |  |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Edit new input information. | | New input information overwrites old input. | As Expected | PASS |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Delete Shipment Information** | |  |  | |  |
| **Test ID:** | **UT4** |  |  | |  |
| **Test** | | **Expected Result** |  | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Delete information. | | Information deleted successfully. |  | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Container** | | | |  |
| **Test ID:** | **UT5** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in all valid information | | Added container successfully | As Expected | PASS |
| Key in invalid information for any of the fields. | | Display error message, could not add container. | As Expected | PASS |
| Blank information for all the required fields. | | Display error message, could not add container. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edit Container Information** | |  |  |  |
| **Test ID:** | **UT6** |  |  |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Edit new input container information. | | New input information overwrites old input. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Delete Container Information** | |  | |  |
| **Test ID:** | **UT7** |  | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Delete Container information | | Container information deleted. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Warehouse** | | | |  |
| **Test ID:** | **UT8** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in all the valid warehouse information. | | Warehouse added successfully | As Expected | PASS |
| Key in invalid information for any of the fields. | | Display error message, could not add warehouse. | As Expected | PASS |
| Blank information for all the required fields. | | Display error message, could not add warehouse. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edit Warehouse Information** | |  |  |  |
| **Test ID:** | **UT9** |  |  |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Edit new input warehouse information. | | New input information overwrites old input. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Delete Warehouse Information** | |  | |  |
| **Test ID:** | **UT10** |  | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Delete warehouse | | Warehouse information deleted. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Ship Yard** | | | |  |
| **Test ID:** | **UT11** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in all the valid warehouse information. | | Added Ship Yard successfully. | As Expected | PASS |
| Key in invalid information for any of the fields. | | Display error message, could not add ship yard. | As Expected | PASS |
| Blank information for all the required fields. | | Display error message, could not add ship yard. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edit Ship Yard Information** | |  |  |  |
| **Test ID:** | **UT12** |  |  |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Edit new input ship yard information. | | New input information overwrites old input. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Delete Ship Yard Information** | |  | |  |
| **Test ID:** | **UT13** |  | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Delete ship yard information. | | Ship Yard information deleted. | As Expected | PASS |

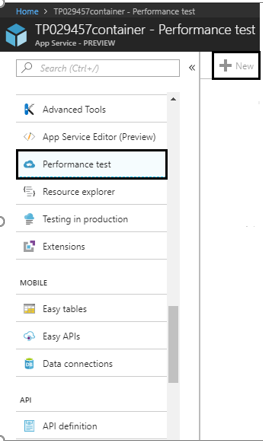
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Customer** | | | |  |
| **Test ID:** | **UT14** | | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Key in all the valid customer details. | | Added Customer Details successfully. | As Expected | PASS |
| Key in invalid details of customer for any of the fields. | | Display error message, could not add customer details. | As Expected | PASS |
| Blank information for all the required fields. | | Display error message, could not add customer details. | As Expected | PASS |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edit Customer Information** | |  |  |  |
| **Test ID:** | **UT15** |  |  |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Edit new input customer details. | | New input details overwrite old input. | As Expected | PASS |

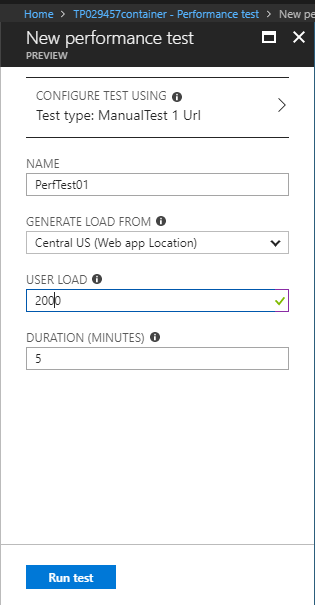
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Delete Customer Information** | |  | |  |
| **Test ID:** | **UT16** |  | |  |
| **Test** | | **Expected Result** | **Actual**  **Result** | **Grade**  **(PASS/FAIL)** |
| Delete customer information. | | Customer information deleted. | As Expected | PASS |

## 5.3 Performance Testing

The developer used Azure performance test tools to perform testing of the web application throughput which is the average response time per second and the CPU processing time as well as working memory needed.



*Figure 44 shows Performance Test Interface.*

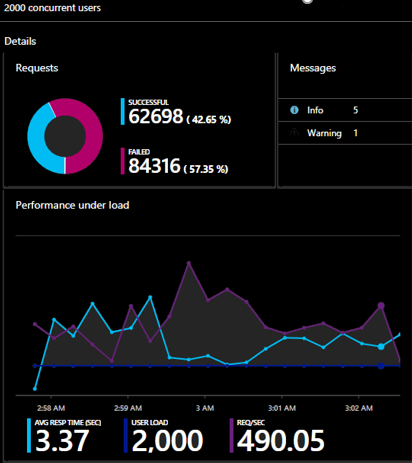


*Figure 45 shows Setting Up Performance Test.*

For the performance test, the name set will be PerfTest01 and the user load is set at 2000, that means there will be 2000 users load on the website concurrently in a period of 5 minutes. The load of the user will be generated from Central US.

**Test 1**

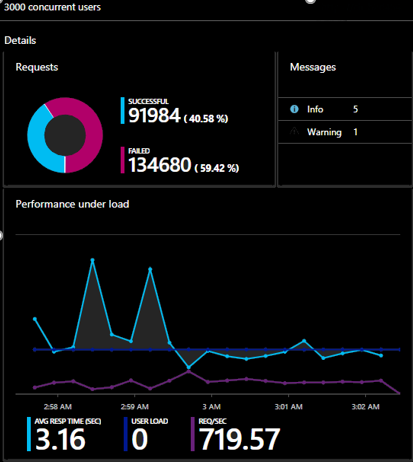
The first test was carried out with 2000 concurrent users accessing the system in 5 minutes time.



*Figure 46 shows Performance Test Conducted with 2000 concurrent users.*

**Test 2**

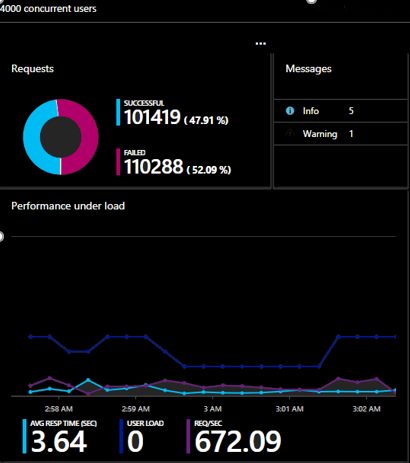
The second test was carried out with 3000 concurrent users accessing the system in 5 minutes time.



*Figure 47 shows Performance Test Conducted with 3000 concurrent users.*

**Test 3**

The third test was carried out with 4000 concurrent users accessing the system in 5 minutes.



*Figure 48 shows Performance Test Conducted with 4000 concurrent users.*

### 5.3.1 Analysis of Test Result

|  |  |
| --- | --- |
| Number of clients over 5 (Min) | Average Response Time (Sec) |
| 2000 | **3.37** |
| 3000 | **3.16** |
| 4000 | **3.64** |

As of the table results above, it can determine that there is a fluctuation of average response as the number of clients over 5 minutes increases. The reason for this fluctuation can be because of the ambiguous internet speed or overwhelming use of resources.

# 6.0 Conclusion

As of the conclusion, the assignment was an eye opener for the developer in demonstrating the power of cloud services provided by Azure platform. The developer has learned many new useful knowledges during the development of the Container Management Web Application especially with the addition of deploying the application on Azure Portal. In this project, the developer has exploit the services of web application and database to control the system up to the standard together successfully linking the database from visual studio into Azure establish the application to a new standard service. Moreover, the developer also has prepared well in advance by setting up a effective cloud architecture. The developer also has selected the Azure Active Directory for user authentication which can increase the security level of the web application by granting access to authorized organization only.

Besides that, the developer also obtain knowledge to utilize the traffic manager to control the traffic flow in the system where it will keep the web application in faster response rate and stable. Furthermore, not only deployment, but the developer has also used the performance test to check the performance of the web application. This performance test was concluded to ensure the system quality as well as the performance running consistently. Developer has also studied to determine the requirement specifically and choose the best service plan in Azure portal to manage with the user load. By doing this, the developer can reduce cost for businesses as the service plans can be purchased according to the requirement.

To conclude this, the system has been built with the functions which was fundamental and within the credit provided in Azure portal. There are much to be learned and the developer will continue to pursue and gain more knowledge about the features and tools provided by Microsoft Azure by implementing in future projects into the cloud platform. This knowledge will be very essential for our future as a software engineer as cloud services is deemed to be the future of information system.

# 7.0 References

Azure.microsoft.com. (2018). Microsoft cloud service | Microsoft Azure. [online]

Available at: <https://azure.microsoft.com/en-us/overview/what-is-azure/>

[Accessed 17 Mar. 2018].

George Reese, 2009, Cloud Application Architectures, O’Reilly Media. Inc. USA.

Jess Chadwick, Todd Snyder, Hrusikesh Panda, 2012, Programming ASP.NET MVC 4:

Developing Real World Web Application with ASP.NET MVC, O’Reilly Media, Inc. USA.

Ibm.com. (2016). IBM Knowledge Center. [online] Available at: <https://www.ibm.com/support/knowledgecenter/SS964W/com.ibm.wbpm.wid.bpel.doc/topics/rdesign.html>

[Accessed 17 Mar. 2018].

Oracle.com. (2016). Design Considerations. [online] Available at: <https://docs.oracle.com/cd/E12890_01/ales/docs32/dvspisec/design.html>

[Accessed 17 Mar. 2018].

Ray Rafaels, 2015, Cloud Computing, Create Space Independent Publishing.

Software Testing Fundamentals. (2018). Functional Testing - Software Testing Fundamentals. [online] Available at: <http://softwaretestingfundamentals.com/functional-testing/>

[Accessed 3 Apr. 2018].

Usability.gov. (2016). User Interface Design Basics | Usability.gov. [online]

Available at: <https://www.usability.gov/what-and-why/user-interface-design.html>

[Accessed 17 Mar. 2018].

Tardif, B. (2013). Scaling Up and Scaling Out in Windows Azure Web Sites. [online] Azure.microsoft.com. Available at: <https://azure.microsoft.com/en-us/blog/scaling-up-and-scaling-out-in-windows-azure-web-sites/>

[Accessed 21 Mar. 2018].

# 8.0 Appendix

GitHub:

Azure: Username – Admin

Password- 123456

Site URL : <http://tp029457container.azurewebsites.net/>

Traffic Manager Name: Traffic ManagerTP029457

URL : <http://trafficmanagertp029457.trafficmanager.net>

Database:

Connection String:

Server=tcp:containerserver.database.windows.net,1433;InitialCatalog=containerdb;Persist SecurityInfo=False;UserID={your\_username};Password={your\_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;