Module Code	**:**	CT071-3.5-3-DDAC
Intake Code	**:**	UC3F1711SE
Lecturer Name	**:**	MOHAMMAD FIRDAUS BIN CHE ABDUL RANI
Hand in Date	**:**	25 JULY 2018
Tutorial No.	**:**	T2
Student Name	**:**	TER WEI LIANG
Student ID	**:**	TP039994

ACKNOWLEDGEMENT

First, I would like to thank my DDAC lecturer, Mr. Mohammad Firdaus bin Che Abdul Rani for the lessons throughout the semester as he makes the class easy to understand and very willing to help his students upon any confusions. Also, it is a must to thank Asia Pacific University for providing this module as an opportunity for students to get exposure onto cloud computing.

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Project Background

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. The company 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.

According to Soeran Lorenzen, an account general manager with HewlettPackard company who is involved first-hand with Maersk’s ITO efforts, a question for the company to support the overall business strategy from an IT perspective is made. Then, a solution to support further business growth and increase organizational flexibility is created by consolidating all of Maersk’s data centers and server rooms that are operating worldwide onto a virtualized platform. While some of Maersk’s IT environment was already hosted on Microsoft Azure, the company decides to change over its IT setup to be based on Microsoft Azure, starting with the desktop environment up to container management.

1.2 Objectives

- Design, develop the Container Management System (CMS).
- Create database on Azure and link with the Container Management System (CMS).
- Deploy the Container Management System (CMS) onto Azure.
- Demonstrate the use of Azure applications for cloud computing.
- Test the system with various testing available on Azure.

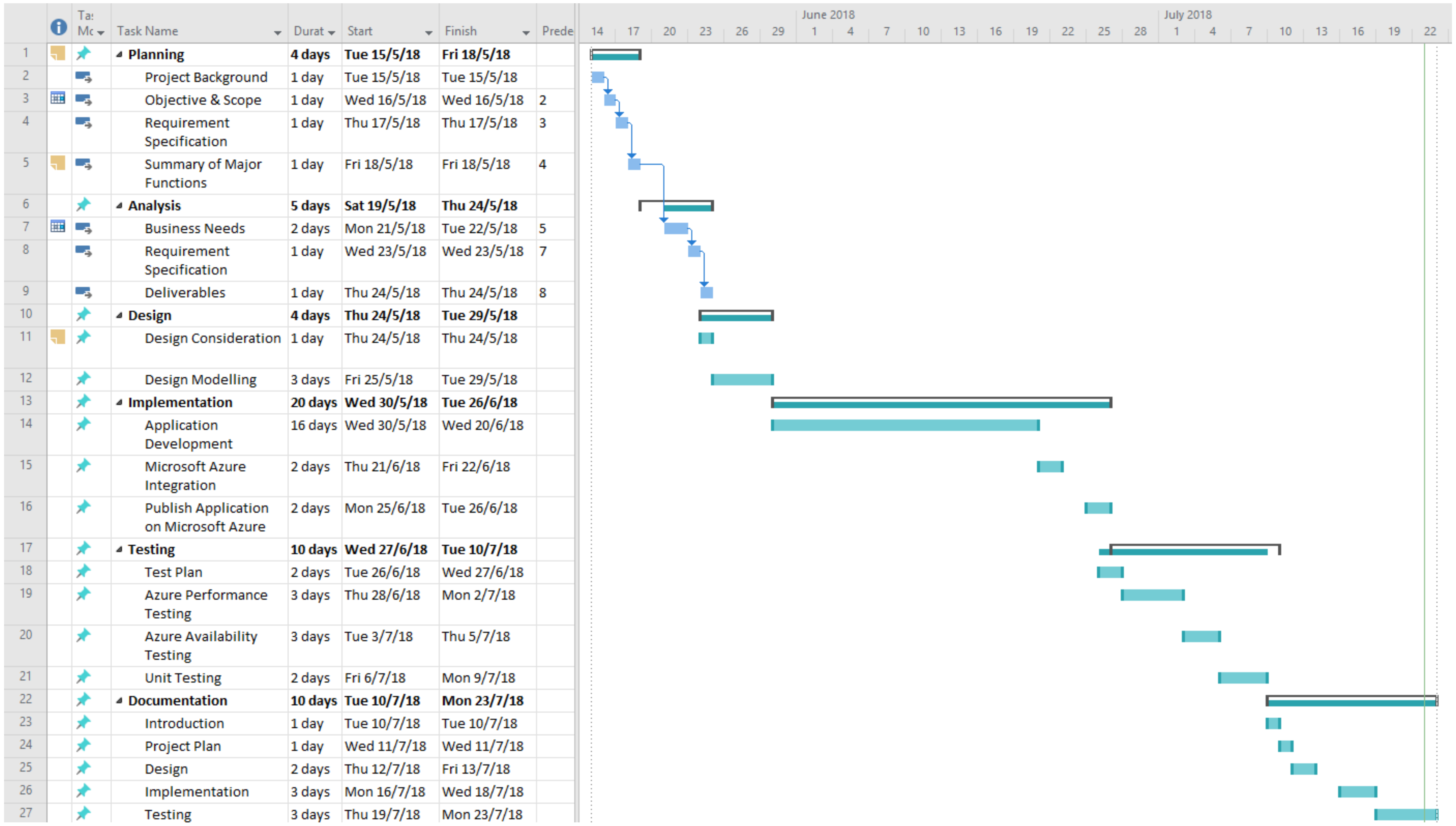
1.3 Requirement Specification

- From import, export and transhipment processing to gate operations.
- To be able to scale the solution to meet the needs of demands during peak seasons.
- Improves profitability, cut costs, increases productivity, eradicates error and optimizes resources to future-proof the cargo handling business for high performance.
- Assurance and reliability through Failover Management.
- Manage the entire booking process from schedule search to booking confirmation.

1.4 Summary of Major Functions

- Design & Develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App)
- Consume SQL Database
- Consist of 5 to 10 interlinked pages
- Provide quality content and design
- Analyze web application performance with monitoring tools
- Suggest solution to scale-up or scale-down to meet the needs of demands during peak season
- Source code to be placed in source control management services

2.0 PROJECT PLAN



	Task Name	Durat	Start	Finish	Prede
1	**Planning**	**4 days**	**Tue 15/5/18**	**Fri 18/5/18**	
2	Project Background	1 day	Tue 15/5/18	Tue 15/5/18	
3	Objective & Scope	1 day	Wed 16/5/18	Wed 16/5/18	2
4	Requirement Specification	1 day	Thu 17/5/18	Thu 17/5/18	3
5	Summary of Major Functions	1 day	Fri 18/5/18	Fri 18/5/18	4
6	**Analysis**	**5 days**	**Sat 19/5/18**	**Thu 24/5/18**	
7	Business Needs	2 days	Mon 21/5/18	Tue 22/5/18	5
8	Requirement Specification	1 day	Wed 23/5/18	Wed 23/5/18	7
9	Deliverables	1 day	Thu 24/5/18	Thu 24/5/18	8
10	**Design**	**4 days**	**Thu 24/5/18**	**Tue 29/5/18**	
11	Design Consideration	1 day	Thu 24/5/18	Thu 24/5/18	
12	Design Modelling	3 days	Fri 25/5/18	Tue 29/5/18	
13	**Implementation**	**20 days**	**Wed 30/5/18**	**Tue 26/6/18**	
14	Application Development	16 days	Wed 30/5/18	Wed 20/6/18	
15	Microsoft Azure Integration	2 days	Thu 21/6/18	Fri 22/6/18	
16	Publish Application on Microsoft Azure	2 days	Mon 25/6/18	Tue 26/6/18	
17	**Testing**	**10 days**	**Wed 27/6/18**	**Tue 10/7/18**	
18	Test Plan	2 days	Tue 26/6/18	Wed 27/6/18	
19	Azure Performance Testing	3 days	Thu 28/6/18	Mon 2/7/18	
20	Azure Availability Testing	3 days	Tue 3/7/18	Thu 5/7/18	
21	Unit Testing	2 days	Fri 6/7/18	Mon 9/7/18	
22	**Documentation**	**10 days**	**Tue 10/7/18**	**Mon 23/7/18**	
23	Introduction	1 day	Tue 10/7/18	Tue 10/7/18	
24	Project Plan	1 day	Wed 11/7/18	Wed 11/7/18	
25	Design	2 days	Thu 12/7/18	Fri 13/7/18	
26	Implementation	3 days	Mon 16/7/18	Wed 18/7/18	
27	Testing	3 days	Thu 19/7/18	Mon 23/7/18	

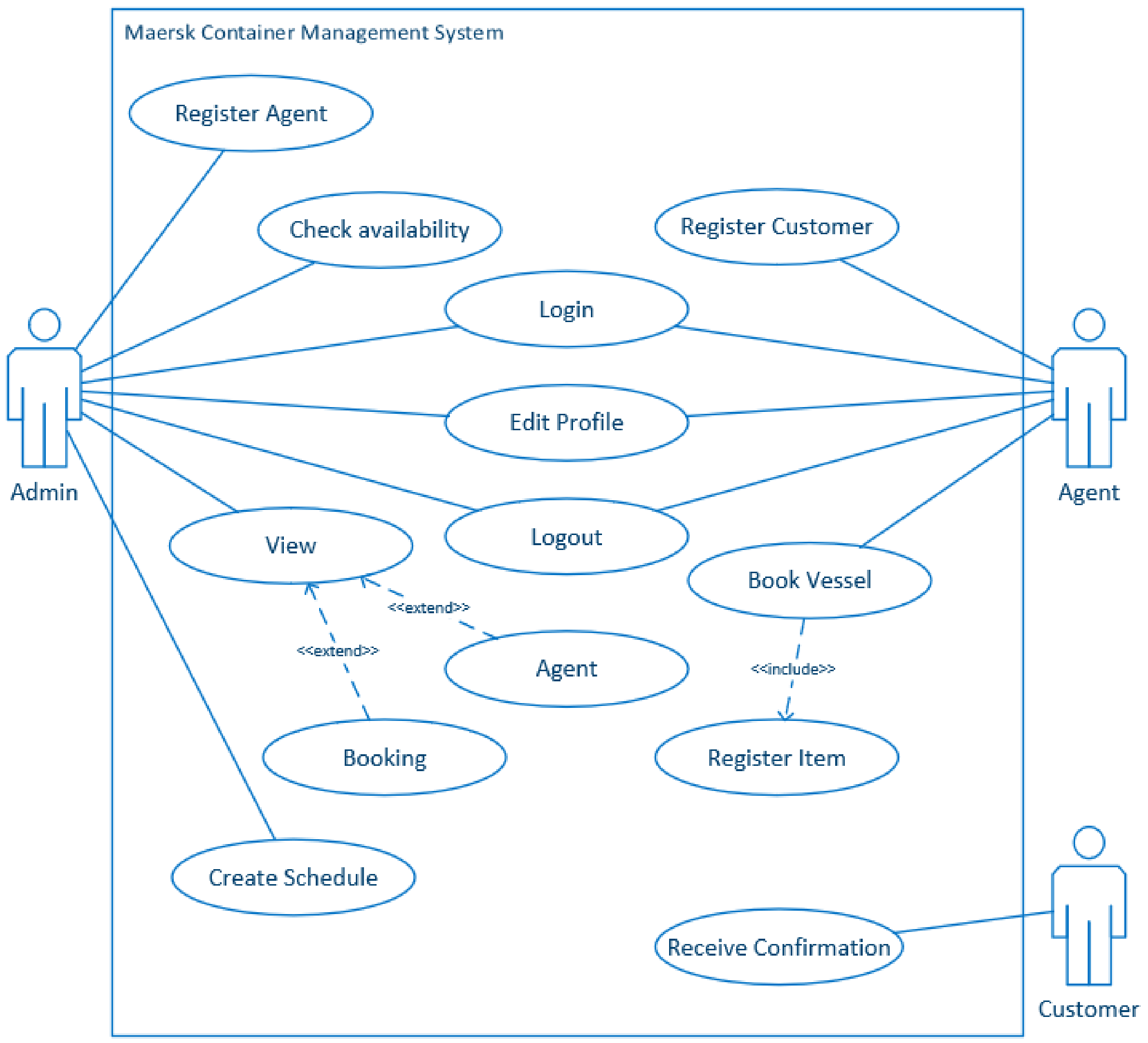
3.0 DESIGN

3.1 Design Considerations

- This web application is assumed to be developed focused on the internal users such as the agent and admin only.
- With the free first 100USD on Azure platform, the credit will be used for the deployment of the system, different types of testing and relational database.
- Create cloud database and put the database into use with system.
- System testing done to fulfil requirements.

3.2 Modelling

3.2.1 Use Case Diagram



3.2.2 Use Case Descriptions

Use Case	Login
Summary	User access the system
Dependency	-
Actor	Admin, Agent
Precondition	-
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Redirects to home page
Description of alternative sequence	2.1 Error message on false credentials 2.2 Error message on empty fields
Post Condition	User can now use the features

Use Case	Logout
Summary	User logs out of system
Dependency	-
Actor	Admin, Agent
Precondition	User logged in
Description of main sequence	1. User clicks logout 2. System logout 3. Redirects to welcome
Description of alternative sequence	-
Post Condition	User logged out of system

Use Case	Register Agent
Summary	Register new agent account
Dependency	-
Actor	Admin
Precondition	Admin login
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Notification upon success
Description of alternative sequence	2.1 Error message on existed credentials 2.2 Error message on empty fields
Post Condition	Registration of agent successful

Use Case	View Booking
Summary	User views booking
Dependency	View
Actor	Admin
Precondition	Admin login
Description of main sequence	1. User clicks view booking button 2. System redirects user to page 3. System shows existing bookings
Description of alternative sequence	-
Post Condition	Shows booking

Use Case	View Agent
Summary	User views agent information
Dependency	View
Actor	Admin
Precondition	Admin login
Description of main sequence	1. User clicks view agent button 2. System redirects user to page 3. System shows existing agents
Description of alternative sequence	-
Post Condition	Shows agents

Use Case	Create Schedule
Summary	Make new schedule for vessels
Dependency	-
Actor	Admin
Precondition	Admin login
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Notification upon success
Description of alternative sequence	2.1 Error message on existed credentials 2.2 Error message on empty fields
Post Condition	Schedule created

Use Case	Register Customer
Summary	Register new customer in system
Dependency	-
Actor	Agent
Precondition	Agent Login
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Notification upon success
Description of alternative sequence	2.1 Error message on existed credentials 2.2 Error message on empty fields
Post Condition	Registration of customer successful

Use Case	Edit Profile
Summary	Update account information
Dependency	-
Actor	Admin, Agent
Precondition	User login
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Notification upon success
Description of alternative sequence	2.1 Error message on existed credentials 2.2 Error message on empty fields
Post Condition	Update successful

Use Case	Book Vessel
Summary	Make booking
Dependency	Register Item
Actor	Agent
Precondition	Agent login
Description of main sequence	1. User uses port range to find vessel 2. System shows available vessel 3. User clicks make booking button
Description of alternative sequence	-
Post Condition	Register Item page

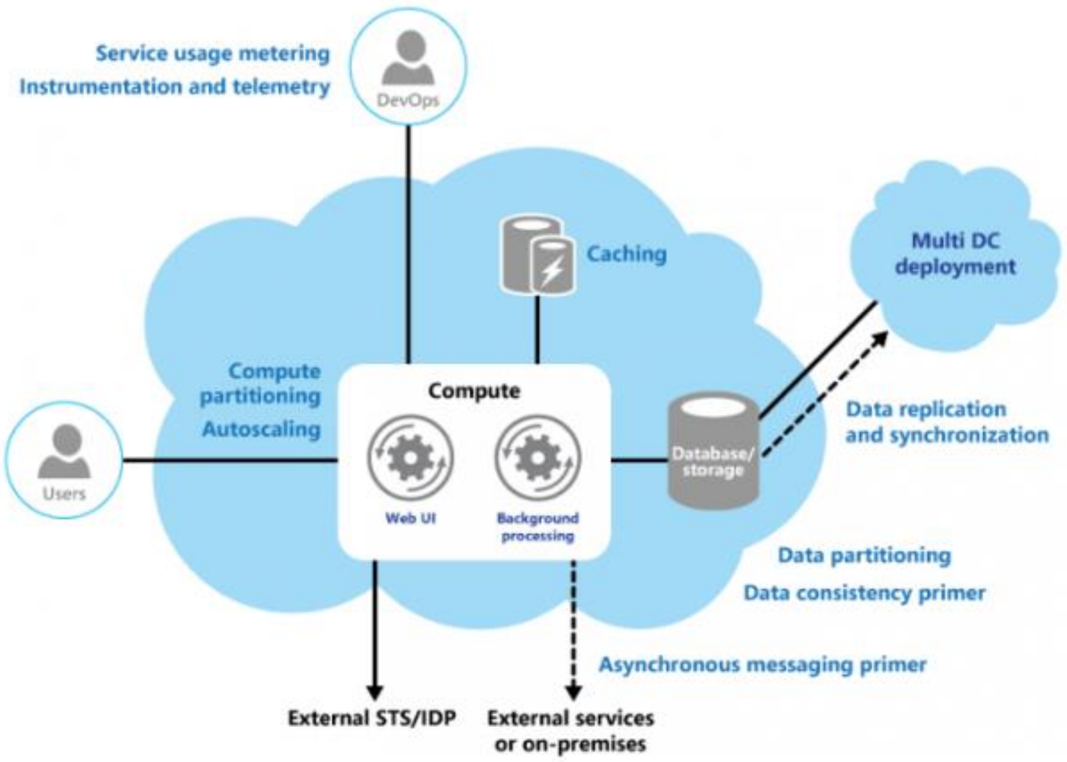
Use Case	Register Item
Summary	Register a booking for customer
Dependency	Email Confirmation
Actor	Agent
Precondition	Book Vessel
Description of main sequence	1. User enter credentials 2. System validate credentials 3. Notification upon success
Description of alternative sequence	2.1 Error message on existed credentials 2.2 Error message on empty fields
Post Condition	Registration successful

Use Case	Email Confirmation
Summary	Email customer on registration success
Dependency	-
Actor	Agent
Precondition	Register Item
Description of main sequence	1. System sends email to customer
Description of alternative sequence	-
Post Condition	Email sent

Use Case	Receive Confirmation
Summary	View confirmation email
Dependency	-
Actor	Customer
Precondition	Registered as customer
Description of main sequence	1. Customer logs into registered email inbox 2. Email states successful
Description of alternative sequence	-
Post Condition	Email seen

3.3 Azure Cloud Design Patterns

Cloud design patterns are made to help provide solutions by identifying and describing the problem context. Each design pattern is differentiated as it is made to solve different problems which the other does not have the capability to. There are currently 24 of these design patterns which are then divided into 8 different problem areas which is the availability, data management, design and implementation, messaging, management and monitoring, performance and scalability, resiliency and security. The figure below represents the basic orientation for developing applications in the cloud. (Cecaro, 2015)



(Cecaro, 2015)

3.3.1 Index Table Pattern

This pattern is used to create indexes over the fields in data stores which are frequently referred by queries. The main reason of having this pattern is to improve query performance by allowing applications to locate the data to be retrieved from the data store quicker. (Microsoft Docs, 2015)

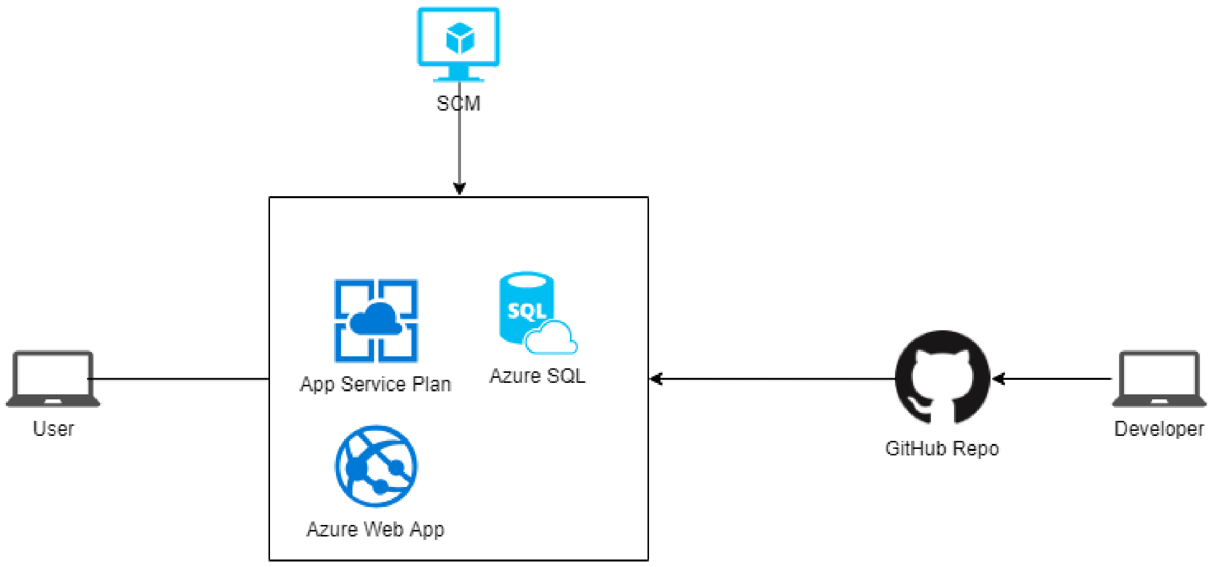
Content and Problem

Most of the data stores are organized through a collection of entities by using the primary key. The problem starts when an application is not able to use the primary key to retrieve data based on other fields. Normally to solve this, developer will create secondary indexes to support different queries but in cloud applications that are using the NoSQL data stores, it does not provide such features. (Microsoft Docs, 2015)

Solution

To solve the issue that the data store does not support features of secondary indexes, the developer can use this pattern to manually create index tables. In index table pattern, there are three common strategies used depending on the number of secondary indexes required and the nature of the queries that an application performs. The first method is by duplicating the data in each index table but also organizing them with different keys. The second method is to create multiple normalized index tables which are organized by different keys. Also using this method will require referencing to the original data by using the primary key rather than duplicating it. The third method is to create partially normalized index tables that are organized by different keys that duplicates frequently duplicates retrieved fields. (Cecaro, 2015)

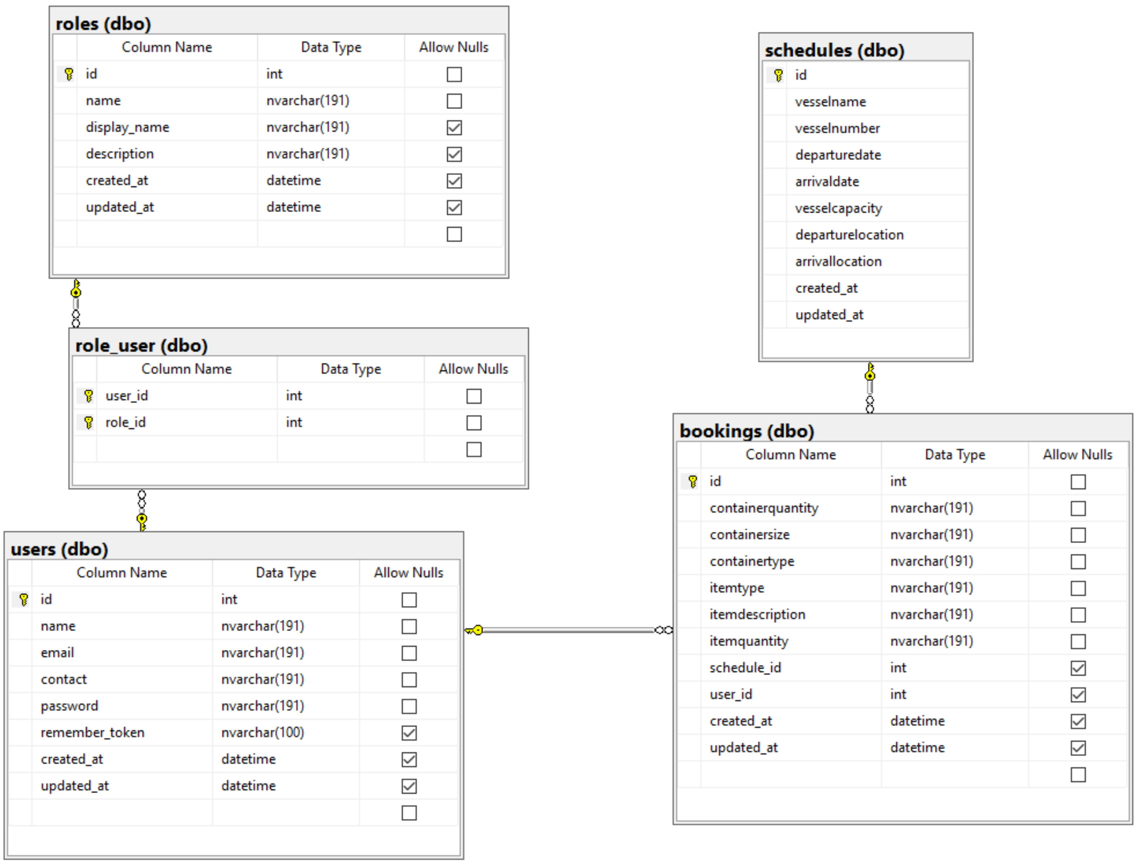
3.4 Architecture Diagram



This diagram shows the basic overview of the system. The developer first uploads the code on GitHub and from there deployment is made. Azure provides the SQL database, web app and the service plan. Also, to configure PHP on Azure, there must be a SCM else the web app will not work. Lastly, user will be able to browse the website.

3.5 Database Design

3.5.1 Entity Relationship Diagram



roles (dbo)

	Column Name	Data Type	Allow Nulls
🔑	id	int	☐
	name	nvarchar(191)	☐
	display_name	nvarchar(191)	☑
	description	nvarchar(191)	☑
	created_at	datetime	☑
	updated_at	datetime	☑
			☐

schedules (dbo)

🔑	id
	vesselname
	vesselnumber
	departuredate
	arrivaldate
	vesselcapacity
	departurelocation
	arrivallocation
	created_at
	updated_at

role_user (dbo)

	Column Name	Data Type	Allow Nulls
🔑	user_id	int	☐
🔑	role_id	int	☐
			☐

bookings (dbo)

	Column Name	Data Type	Allow Nulls
🔑	id	int	☐
	containerquantity	nvarchar(191)	☐
	containersize	nvarchar(191)	☐
	containertype	nvarchar(191)	☐
	itemtype	nvarchar(191)	☐
	itemdescription	nvarchar(191)	☐
	itemquantity	nvarchar(191)	☐
	schedule_id	int	☑
	user_id	int	☑
	created_at	datetime	☑
	updated_at	datetime	☑
			☐

users (dbo)

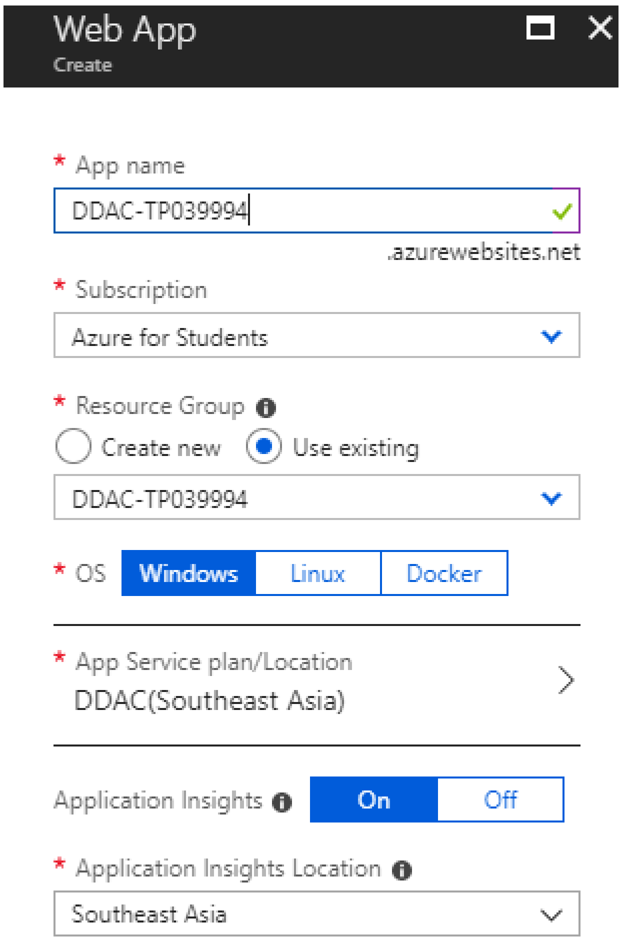
	Column Name	Data Type	Allow Nulls
🔑	id	int	☐
	name	nvarchar(191)	☐
	email	nvarchar(191)	☐
	contact	nvarchar(191)	☐
	password	nvarchar(191)	☐
	remember_token	nvarchar(100)	☑
	created_at	datetime	☑
	updated_at	datetime	☑
			☐

4.0 IMPLEMENTATION

4.1 Publishing the Container Management System onto Azure

The process of deploying this web application onto Azure is rather difficult compared to using Visual Studio. This is because the program was created using **PHPSTORM** IDE while the language is **PHP** and running on **LARAVEL** framework. First, to publish the web application, the developer will have to set up the application as a Git repo. To do this, **GitHub** has been selected as the provider.

Next, the developer will have to create an **Azure Web App**, doing the configurations and for this assignment, the standard service plan has been selected with Southeast Asia being its location.



* App name: DDAC-TP039994.azurewebsites.net

* Subscription: Azure for Students

* Resource Group: Create new [✓ Use existing]

DDAC-TP039994

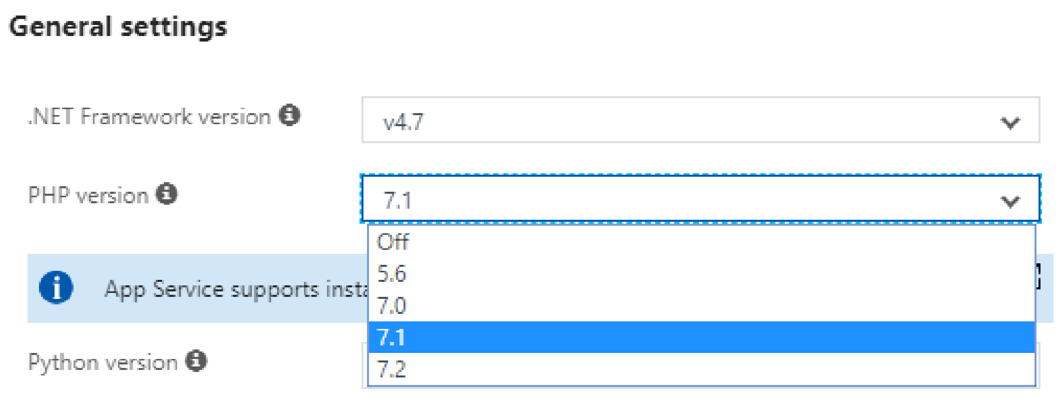
* OS: [✓ Windows] Linux Docker

* App Service plan/Location: DDAC(Southeast Asia)

Application Insights: [✓ On] Off

* Application Insights Location: Southeast Asia

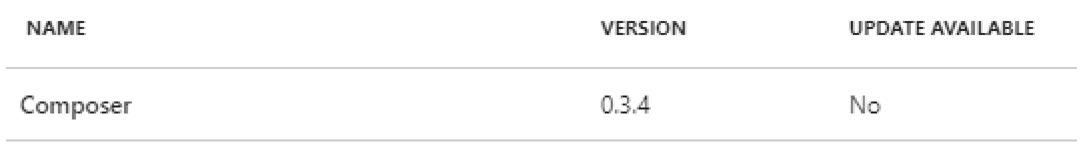
Once created, head over to the **APPLICATION SETTINGS** and check for the PHP version to see if it matches with the one on our framework.



General settings

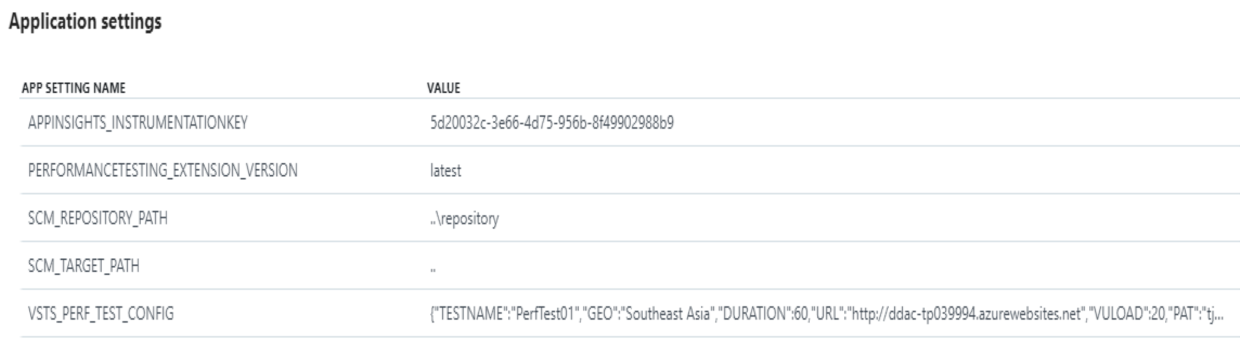
.NET Framework version	v4.7
PHP version	7.1
Python version	

Then, head over to **DEVELOPMENT TOOLS** and on the **EXTENSIONS**, add **COMPOSER**.



NAME	VERSION	UPDATE AVAILABLE
Composer	0.3.4	No

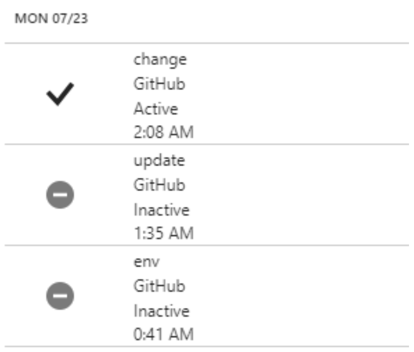
After that, the developer will have to configure the environment variables by going over to **APPLICATION SETTINGS**, scroll down to **APP SETTINGS** then enter **<SCM_REPOSITORY_PATH>** with the value **<..\repository>** and also **<SCM_TARGET_PATH>** with value **<..>**. Then on the same page, scroll down to **VIRTUAL APPLICATIONS AND DIRECTORIES** and change the public directory value from **<site\wwwroot>** to **<site\public>**.



Application settings

APP SETTING NAME	VALUE
APPINSIGHTS_INSTRUMENTATIONKEY	5d20032c-3e66-4d75-956b-8f49902988b9
PERFORMANCETESTING_EXTENSION_VERSION	latest
SCM_REPOSITORY_PATH	..\repository
SCM_TARGET_PATH	..
VSTS_PERF_TEST_CONFIG	{"TESTNAME":"PerfTest01","GEO":"Southeast Asia","DURATION":60,"URL":"http://ddac-tp039994.azurewebsites.net","VULOAD":20,"PAT":"tj...

Next, head over to **DEPLOYMENT** section and click on **DEPLOYMENT OPTIONS**. Here, the developer will select the Remote Git Provider which in this assignment is **GitLab**. Then after confirming the credentials, the developer will see deployments that has been committed.



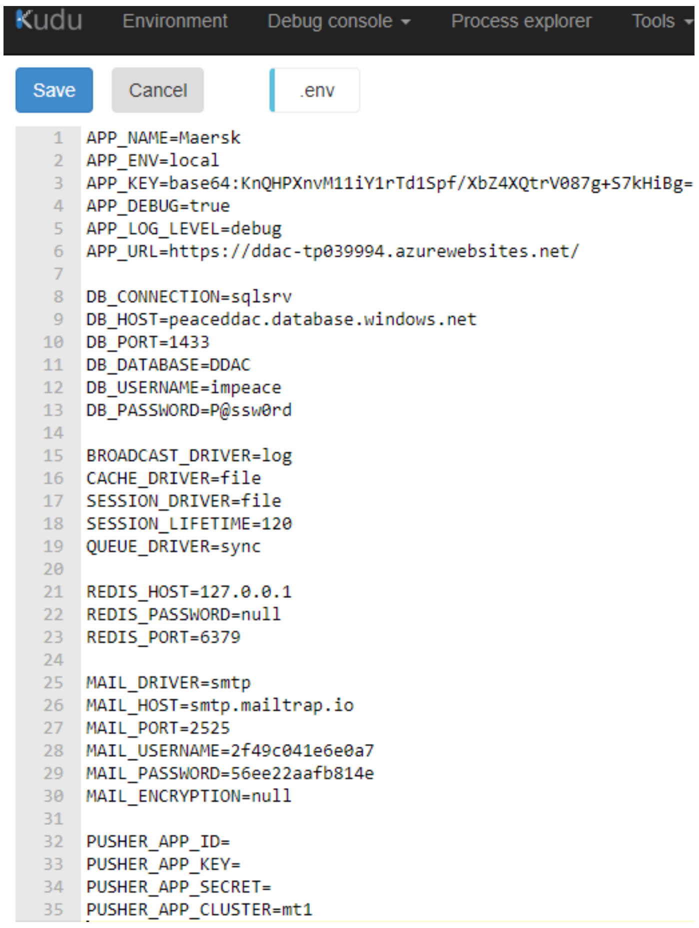
MON 07/23

✓ change
GitHub
Active
2:08 AM

update
GitHub
Inactive
1:35 AM

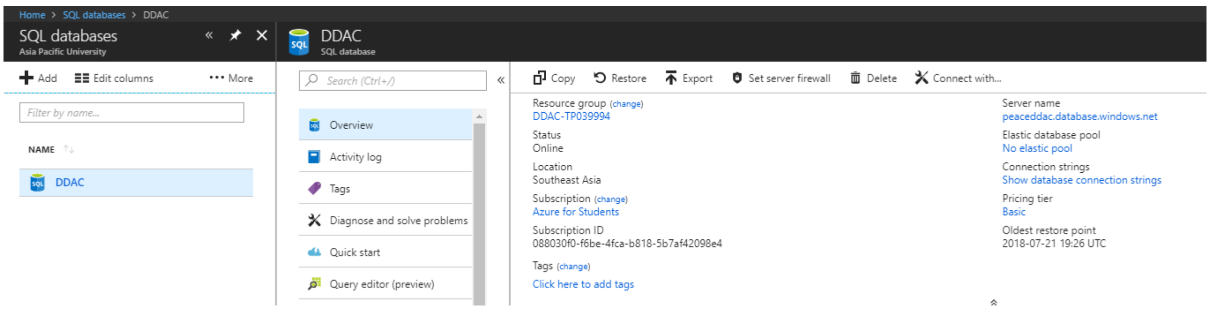
env
GitHub
Inactive
0:41 AM

After its done, the developer will next have to add a 'web.config' file in the SCM of the web application. Lastly, as this is running on Laravel framework, the .env file will not be pushed onto the Git Providers therefore at the SCM Console, the developer will have to clone the .env.example to .env and fill in the credentials. (Adepoju, 2017)



```
APP_NAME=Maersk
APP_ENV=local
APP_KEY=base64:KnQHPXnvM11iY1rTd1Spf/XbZ4XQtrV087g+S7kHiBg=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=https://ddac-tp039994.azurewebsites.net/

DB_CONNECTION=sqlsrv
DB_HOST=peaceddac.database.windows.net
DB_PORT=1433
DB_DATABASE=DDAC
DB_USERNAME=impeace
DB_PASSWORD=P@ssw0rd

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=2f49c041e6e0a7
MAIL_PASSWORD=56ee22aafb814e
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1
```

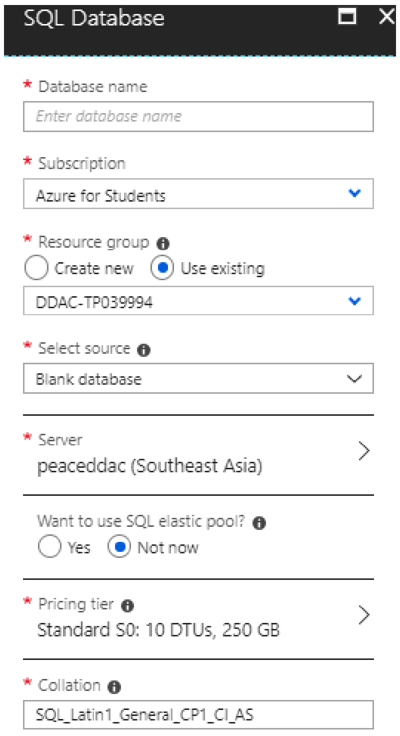
For the database, the Azure SQL Database is used in this assignment. However, having use a cloud database will reduce to flexibility of Laravel but it became manageable with the SCM console. Laravel has special commands such as “php artisan migrate –seed” which will migrate all the database that is supposed to be available for the system and seeded data will be pumped into the system.



Resource group (change)
DDAC-TP039994
Status
Online
Location
Southeast Asia
Subscription (change)
Azure for Students
Subscription ID
088030f0-f6be-4fca-b818-5b7af42098e4
Tags (change)
Click here to add tags

Server name
peaceddac.database.windows.net
Elastic database pool
No elastic pool
Connection strings
Show database connection strings
Pricing tier
Basic
Oldest restore point
2018-07-21 19:26 UTC

To create the database, developer will have to fill in the credentials in the figure below.


SQL Database

* Database name
Enter database name

* Subscription
Azure for Students

* Resource group
Create new [✓ Use existing]
DDAC-TP039994

* Select source
Blank database

* Server
peaceddac (Southeast Asia)

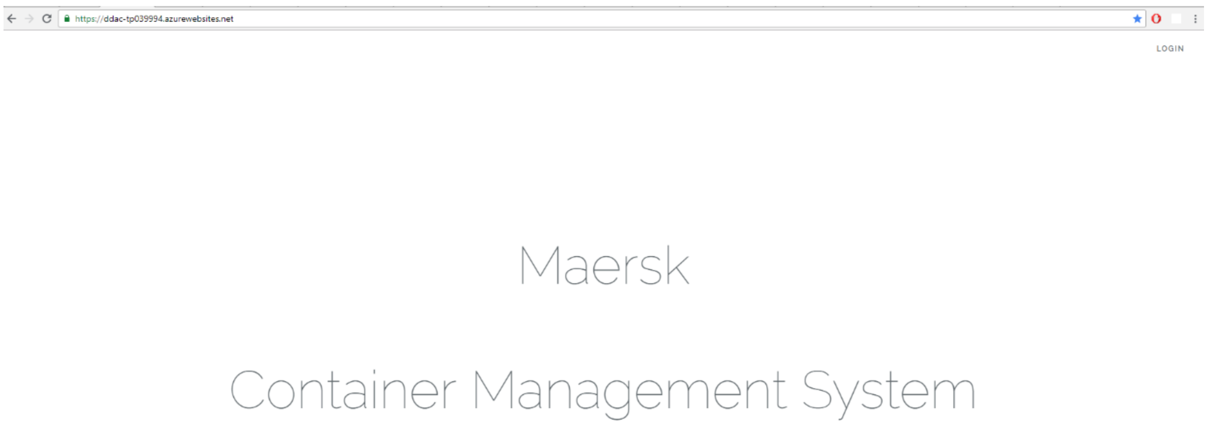
Want to use SQL elastic pool?
Yes [✓ Not now]

* Pricing tier
Standard S0: 10 DTUs, 250 GB

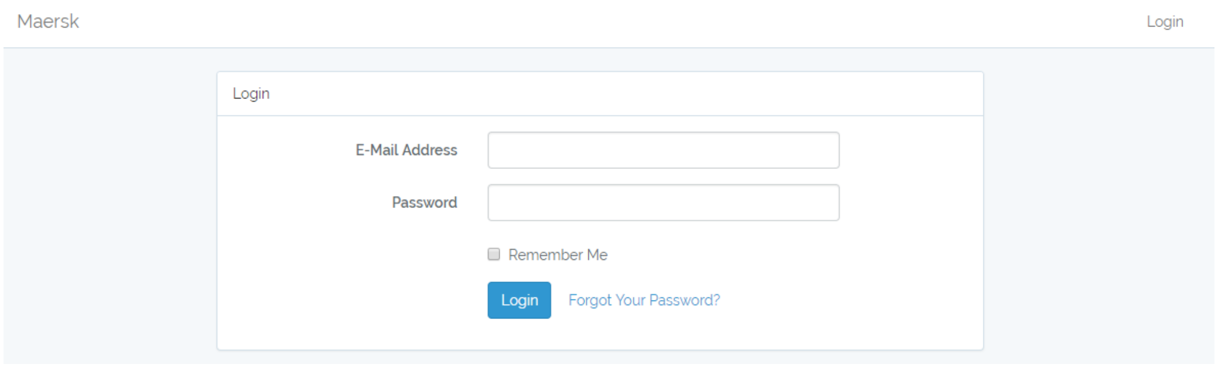
* Collation
SQL_Latin1_General_CP1_CI_AS

4.2 Screenshots

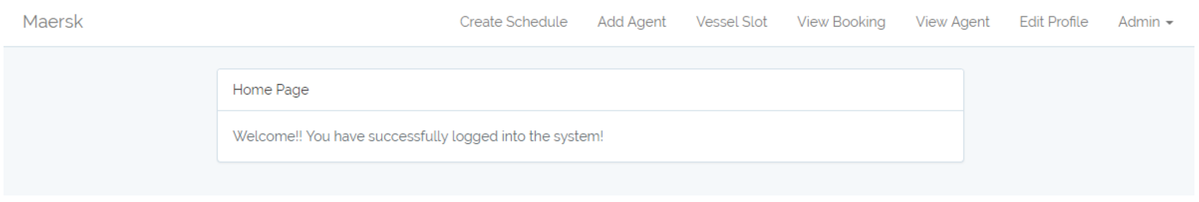
4.2.1 Home Page


https://ddac-tp039994.azurewebsites.net
LOGIN
Maersk
Container Management System

4.2.2 Login

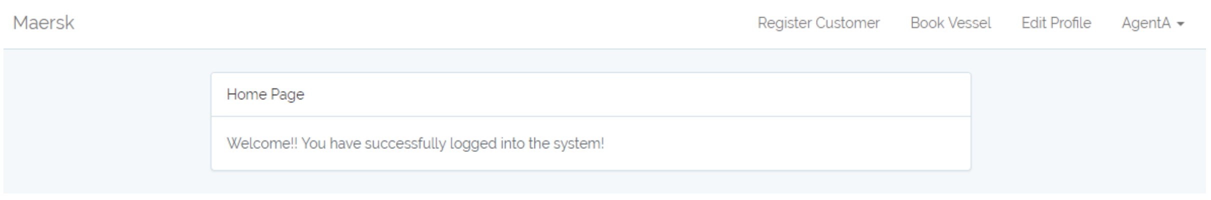


4.2.3 Admin Menu



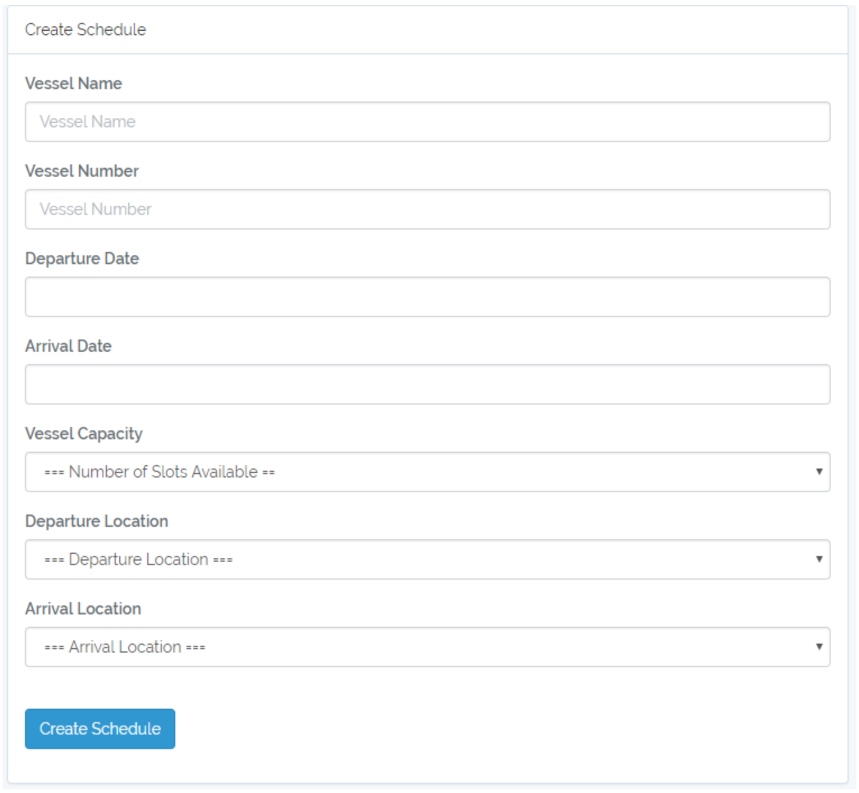
Welcome!! You have successfully logged into the system!

4.2.4 Agent Menu



Welcome!! You have successfully logged into the system!

4.2.5 Create Schedule


Create Schedule

Vessel Name

Vessel Name

Vessel Number

Vessel Number

Departure Date

Arrival Date

Vessel Capacity

=== Number of Slots Available ==

Departure Location

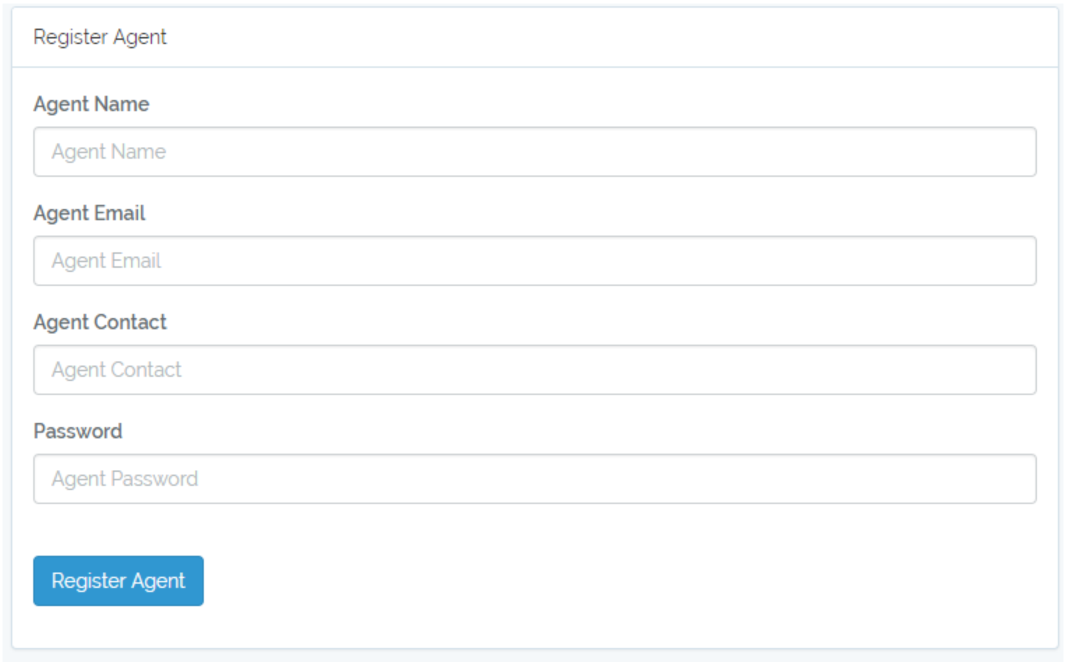
=== Departure Location ===

Arrival Location

=== Arrival Location ===

Create Schedule

4.2.6 Register Agent


Register Agent

Agent Name

Agent Name

Agent Email

Agent Email

Agent Contact

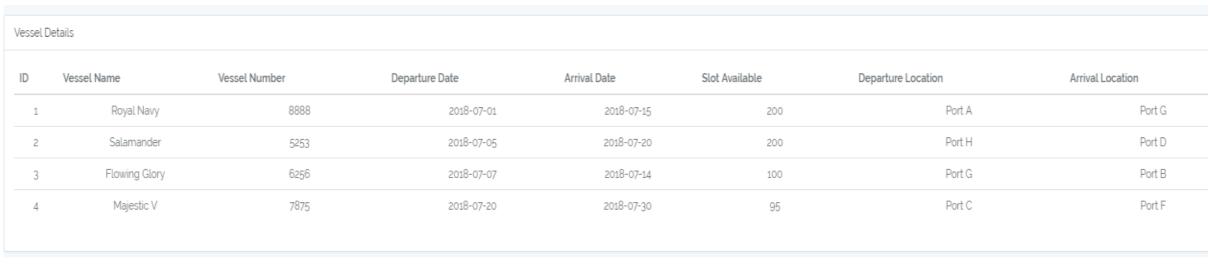
Agent Contact

Password

Agent Password

Register Agent

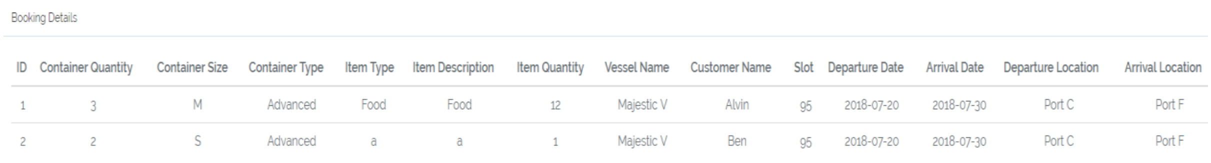
4.2.7 Vessel Details



Vessel Details

ID	Vessel Name	Vessel Number	Departure Date	Arrival Date	Slot Available	Departure Location	Arrival Location
1	Royal Navy	8888	2018-07-01	2018-07-15	200	Port A	Port G
2	Salamander	5253	2018-07-05	2018-07-20	200	Port H	Port D
3	Flowing Glory	6256	2018-07-07	2018-07-14	100	Port G	Port B
4	Majestic V	7875	2018-07-20	2018-07-30	95	Port C	Port F

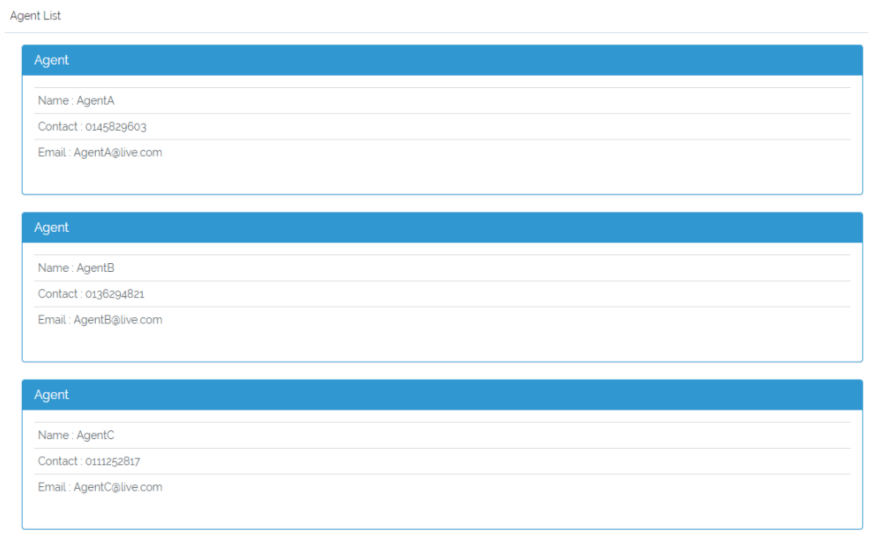
4.2.8 Booking Details



Booking Details

ID	Container Quantity	Container Size	Container Type	Item Type	Item Description	Item Quantity	Vessel Name	Customer Name	Slot	Departure Date	Arrival Date	Departure Location	Arrival Location
1	3	M	Advanced	Food	Food	12	Majestic V	Alvin	95	2018-07-20	2018-07-30	Port C	Port F
2	2	S	Advanced	a	a	1	Majestic V	Ben	95	2018-07-20	2018-07-30	Port C	Port F

4.2.9 Agent List



Agent List

Agent

Name : AgentA

Contact : 0145829603

Email : AgentA@live.com

Agent

Name : AgentB

Contact : 0136294821

Email : AgentB@live.com

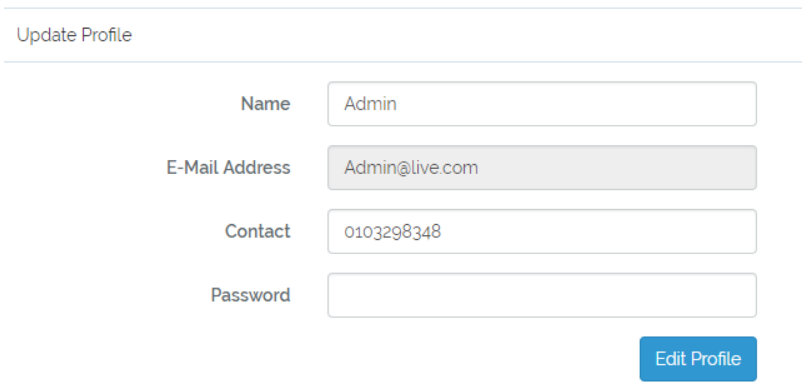
Agent

Name : AgentC

Contact : 0111252817

Email : AgentC@live.com

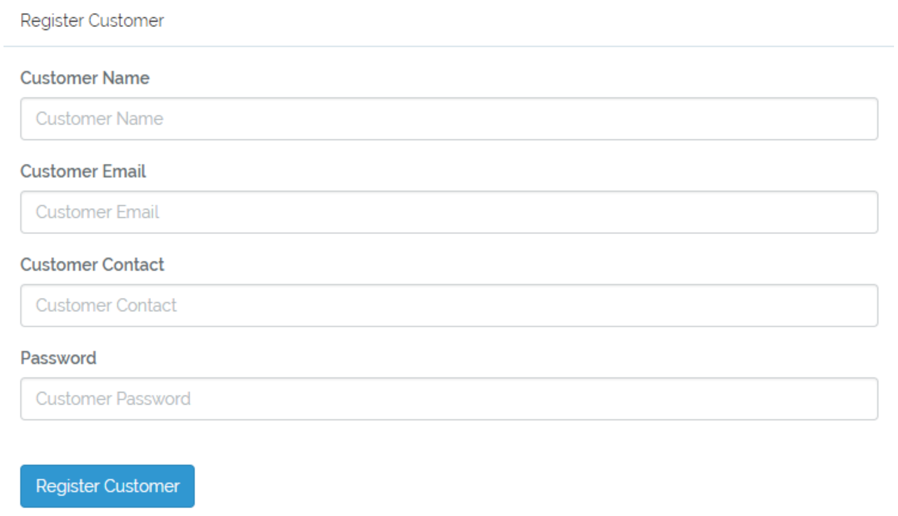
4.2.10 Update Profile



Update Profile

Name	Admin
E-Mail Address	Admin@live.com
Contact	0103298348
Password	

4.2.11 Register Customer



Register Customer

Customer Name

Customer Name

Customer Email

Customer Email

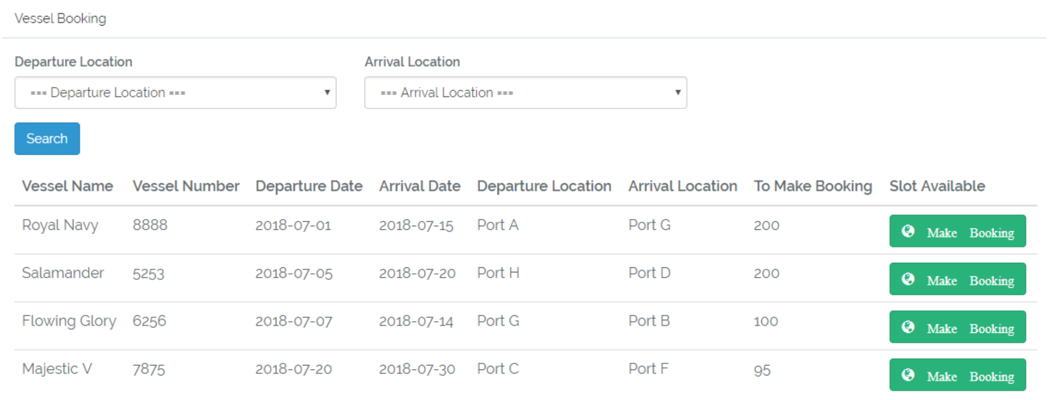
Customer Contact

Customer Contact

Password

Customer Password

4.2.12 Vessel Booking



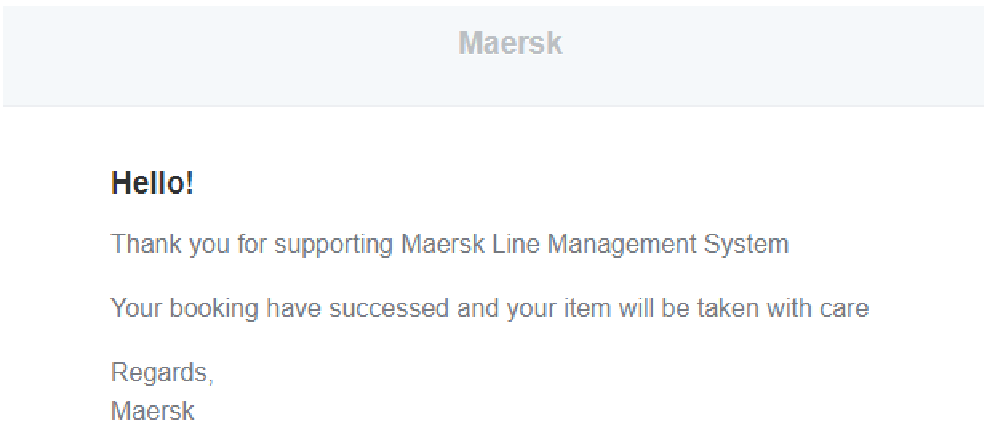
Vessel Booking

Departure Location: === Departure Location ===

Arrival Location: === Arrival Location ===

Vessel Name	Vessel Number	Departure Date	Arrival Date	Departure Location	Arrival Location	To Make Booking	Slot Available
Royal Navy	8888	2018-07-01	2018-07-15	Port A	Port G	200	Make Booking
Salamander	5253	2018-07-05	2018-07-20	Port H	Port D	200	Make Booking
Flowing Glory	6256	2018-07-07	2018-07-14	Port G	Port B	100	Make Booking
Majestic V	7875	2018-07-20	2018-07-30	Port C	Port F	95	Make Booking

4.2.13 Approval Notification



Maersk

Hello!

Thank you for supporting Maersk Line Management System

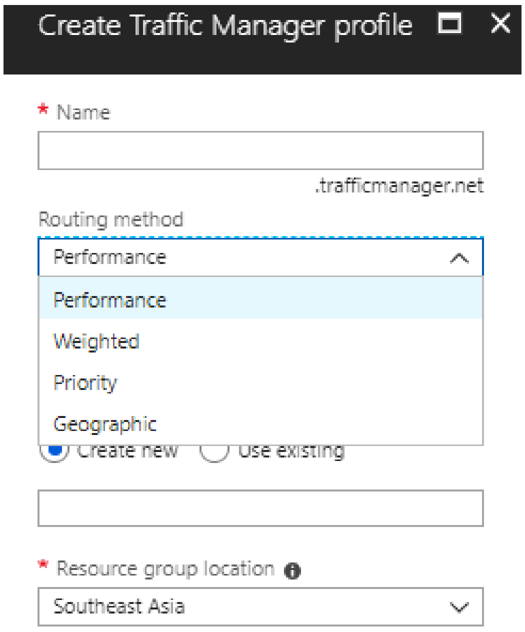
Your booking have successed and your item will be taken with care

Regards,
Maersk

5.0 TRAFFIC MANAGER

The Azure Traffic Manager supports four different types of traffic routing methods which are used to determine how to route the network traffic onto various service endpoints. It is applied to each DNS query that it receives, and the methods will determine which endpoint it should return on the side of response. The traffic routing methods are Priority, Weighted, Performance and Geographic.

5.1 Create Traffic Manager Profile


Create Traffic Manager profile

* Name

.trafficmanager.net

Routing method

Performance

Performance

Weighted

Priority

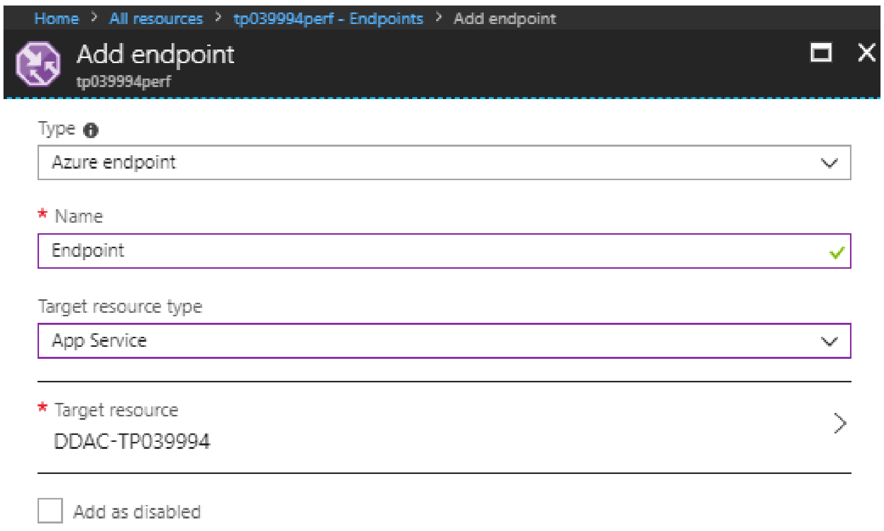
Geographic

Create new Use existing

* Resource group location

Southeast Asia

5.2 Add Traffic Manager Endpoints


Home > All resources > tp039994perf - Endpoints > Add endpoint

Add endpoint

tp039994perf

Type

Azure endpoint

* Name

Endpoint

Target resource type

App Service

* Target resource

DDAC-TP039994

Add as disabled

6.0 WEB APP TESTS

6.1 User Acceptance Test

Category No	Category	Scale									
		1	2	3	4	5	6	7	8	9	10
1	User Friendliness							✓			
2	Security								✓		
3	Performance							✓			
4	Reliability							✓			
5	Meet Objectives							✓			

6.2 Unit Testing

No.		1		
Tested on		Login		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Redirected to home page	As expected	Pass
2	Input wrong credentials on any input field	Credentials does not match records	As expected	Pass
3	Leave input field blank	Prompt to fill out the field	As expected	Pass

No.		2		
Tested on		View Vessel		
Test Case	Description	Expected Result	Actual Result	Remarks
1	View vessels	Display vessel information	As expected	Pass

No.		3		
Tested on		Edit Profile		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Successful Update	As expected	Pass
2	Leave input field blank	Prompt to fill out the field	As expected	Pass

No.		4		
Tested on		Create Schedule		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Notified on success	As expected	Pass
2	Leave input field blank	Prompt to fill out the field	As expected	Pass

No.		5		
Tested on		Add Agent		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Notified on success	As expected	Pass
2	Leave input field blank	Prompt to fill out the field	As expected	Pass

No.		6		
Tested on		View Booking		
Test Case	Description	Expected Result	Actual Result	Remarks
1	View Booking	Display booking information	As expected	Pass

No.		7		
Tested on		Register Customer		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Notified on success	As expected	Pass
2	Leave input field blank	Prompt to fill out the field	As expected	Pass
3	Input existed info	Error stating info existed	As expected	Pass

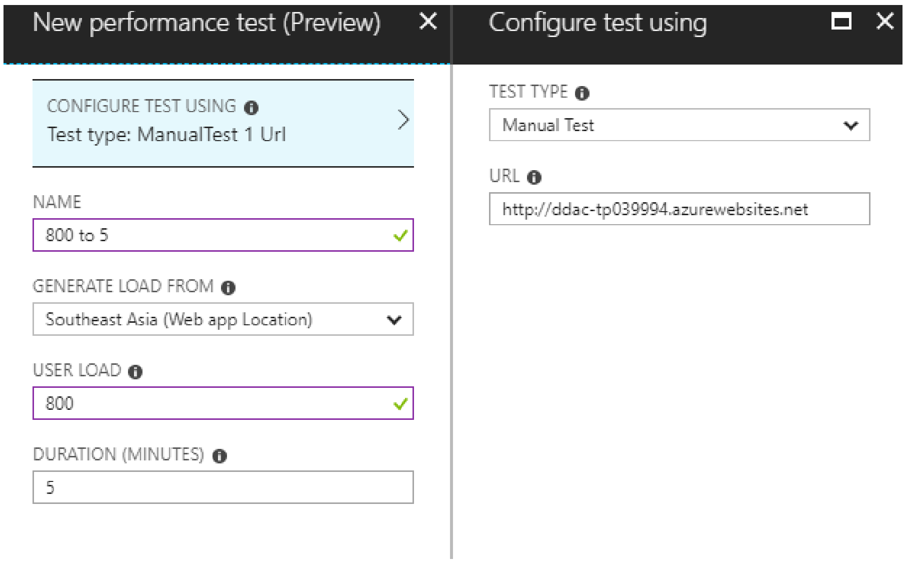
No.		8		
Tested on		View Agent		
Test Case	Description	Expected Result	Actual Result	Remarks
1	View Agent	Display agent information	As expected	Pass

No.		9		
Tested on		Register		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Input correct credentials	Notified on success	As expected	Pass
2	Leave input field blank	Prompt to fill out the field	As expected	Pass

No.		10		
Tested on		Book Vessel		
Test Case	Description	Expected Result	Actual Result	Remarks
1	Click on Make Booking	Move to Register item page	As expected	Pass

6.3 Azure Performance Test

Configuring the performance test



New performance test (Preview)

CONFIGURE TEST USING
Test type: ManualTest 1 Url

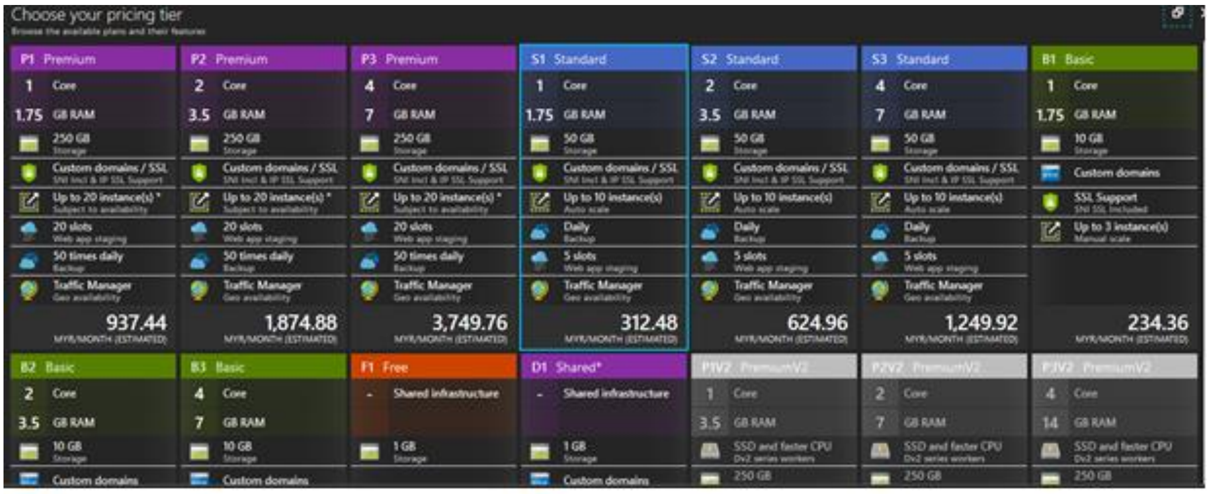
NAME	800 to 5
GENERATE LOAD FROM	Southeast Asia (Web app Location)
USER LOAD	800
DURATION (MINUTES)	5

Configure test using

TEST TYPE	Manual Test
URL	http://ddac-tp039994.azurewebsites.net

The figure above shows the configuration for one of the configuration tests where the user load is at 800 concurrent users and 5 minutes being its duration. The same type of testing will be running for 5 times with the same amount of time but different user load at 200, 400, 600, 800 and 1000 respectively.

Pricing Tiers



Choose your pricing tier
Browse the available plans and their features

P1 Premium	P2 Premium	P3 Premium	S1 Standard	S2 Standard	S3 Standard	B1 Basic
1 Core	2 Core	4 Core	1 Core	2 Core	4 Core	1 Core
1.75 GB RAM	3.5 GB RAM	7 GB RAM	1.75 GB RAM	3.5 GB RAM	7 GB RAM	1.75 GB RAM
250 GB Storage	250 GB Storage	250 GB Storage	50 GB Storage	50 GB Storage	50 GB Storage	10 GB Storage
Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains
Up to 20 instance(s) * Subject to availability	Up to 20 instance(s) * Subject to availability	Up to 20 instance(s) * Subject to availability	Up to 10 instance(s) Auto scale	Up to 10 instance(s) Auto scale	Up to 10 instance(s) Auto scale	SSL Support SNI SSL Included
20 slots Web app staging	20 slots Web app staging	20 slots Web app staging	Daily Backup	Daily Backup	Daily Backup	Up to 3 instance(s) Manual scale
50 times daily Backup	50 times daily Backup	50 times daily Backup	5 slots Web app staging	5 slots Web app staging	5 slots Web app staging	
Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability	
937.44 MYR/MONTH (ESTIMATED)	1,874.88 MYR/MONTH (ESTIMATED)	3,749.76 MYR/MONTH (ESTIMATED)	312.48 MYR/MONTH (ESTIMATED)	624.96 MYR/MONTH (ESTIMATED)	1,249.92 MYR/MONTH (ESTIMATED)	234.36 MYR/MONTH (ESTIMATED)

B2 Basic	B3 Basic	F1 Free	D1 Shared*	P1V2 PremiumV2	P2V2 PremiumV2	P3V2 PremiumV2
2 Core	4 Core	- Shared infrastructure	- Shared infrastructure	1 Core	2 Core	4 Core
3.5 GB RAM	7 GB RAM			3.5 GB RAM	7 GB RAM	14 GB RAM
10 GB Storage	10 GB Storage	1 GB Storage	1 GB Storage	SSD and faster CPU Dv2 series workers	SSD and faster CPU Dv2 series workers	SSD and faster CPU Dv2 series workers
Custom domains	Custom domains		Custom domains	250 GB	250 GB	250 GB

For this assignment, the tiers of Standard 1 (S1), Standard 2 (S2) and Standard 3 (S3) will be used. Different tier of pricing will impact the system differently and by doing testing of this, there will be an answer to which pricing tier fits the system better.

Data of Standard 1 Pricing Tier.

STANDARD 1	Data				
Concurrent User	200	400	600	800	1000
Minutes	5				
Average Response Time (SEC)	20.06	27.25	25.25	15.71	34.76
Request Successful (%)	100	100	100	100	100
Request Failed (%)	0	0	0	0	0

Data of Standard 2 Pricing Tier.

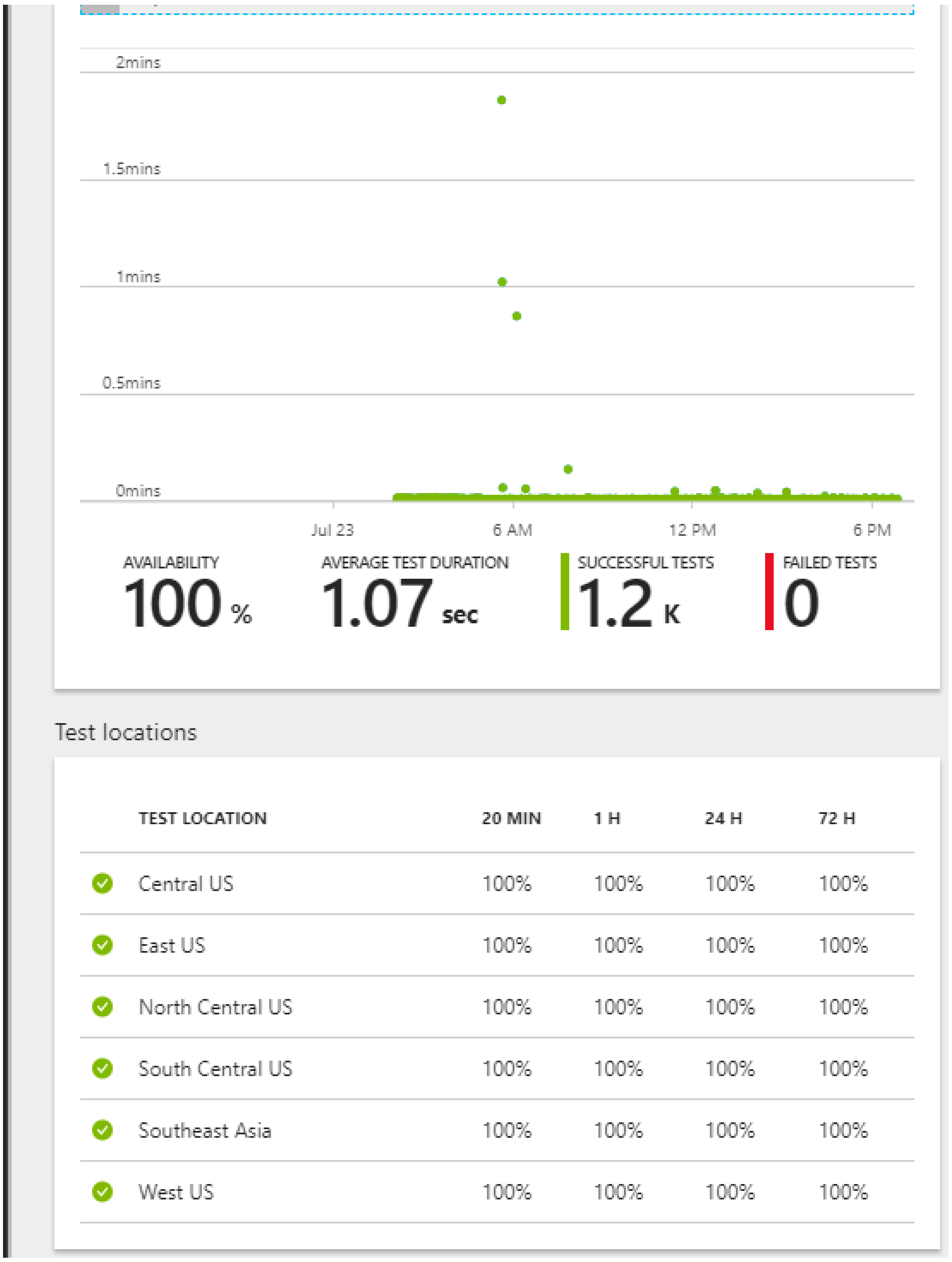
STANDARD 2	Data				
Concurrent User	200	400	600	800	1000
Minutes	5				
Average Response Time (SEC)	21.54	21.61	16.59	29.71	27.19
Request Successful (%)	98.88	99.66	99.72	99.48	99.52
Request Failed (%)	0.12	0.34	0.28	0.52	0.48

Data of Standard 3 Pricing Tier.

STANDARD 3	Data				
Concurrent User	200	400	600	800	1000
Minutes	5				
Average Response Time (SEC)	6.16	8.57	15.78	11.12	13.28
Request Successful (%)	99.44	99.47	99.57	99.41	99.63
Request Failed (%)	0.56	0.53	0.43	0.59	0.37

The above tables capture the data of the performance tests with different pricing tiers. Using pricing tier S1, there is a 100% request successful rate. But in terms of average response time, the pricing tier S3 has the best average time compared to the others. The Maersk Container Line System should not focus on going for the tier 2 pricing as it does not benefit the system compare to their others. If the system is more focused on success rate, therefore pricing tier S1 will be the best fit but if the system is more focused on the average response time, the system should take the S3 pricing tier.

6.4 Azure Availability Test



AVAILABILITY 100 %

AVERAGE TEST DURATION 1.07 sec

SUCCESSFUL TESTS 1.2 K

FAILED TESTS 0

Test locations

	TEST LOCATION	20 MIN	1 H	24 H	72 H
✓	Central US	100%	100%	100%	100%
✓	East US	100%	100%	100%	100%
✓	North Central US	100%	100%	100%	100%
✓	South Central US	100%	100%	100%	100%
✓	Southeast Asia	100%	100%	100%	100%
✓	West US	100%	100%	100%	100%

After 2 days of availability testing, statistic has shown a 100% in all location with different time length. The average test duration is at 1.07 seconds which is considered fast and at the 72nd hour, there has been a total of more than 1000 tests succeeded.

7.0 CONCLUSION

This assignment was a challenging task as it is the first time that an application needs to be deployed in an online real time scenario. Also, because this application was made with PHP, there are some extra steps in comparison with using C# to deploy the website as the IDE is not under Microsoft. It has given a great experience as a developer from coding to deployment and understandings on different cloud testing like the availability and performance tests. With the completion of this assignment, future work will be easier to handle as I will be heading to a company that runs such systems.

8.0 REFERENCES

Adepoju, F., 2017. *Hosting a Laravel Application on Azure Web App.* [Online]
Available at: https://medium.com/@coderonfleek/hosting-a-laravel-application-on-azure-web-app-b55e12514c46
[Accessed 20 7 2018].

Cecaro, F., 2015. *Azure Cloud Design Patterns.* [Online]
Available at: https://cloudacademy.com/blog/azure-cloud-design-patterns/
[Accessed 20 7 2018].

Microsoft Azure, 2017. *Overview of Traffic Manager.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview
[Accessed 20 7 2018].

Microsoft Azure, 2018. *Cloud Services pricing.* [Online]
Available at: https://azure.microsoft.com/en-us/pricing/details/cloud-services/
[Accessed 20 7 2018].

Microsoft Azure, 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 20 7 2018].

Microsoft Azure, 2018. *Monitor availability and responsiveness of any web site.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/application-insights/app-insights-monitor-web-app-availability
[Accessed 20 7 2018].

Microsoft Docs, 2015. *Index Table Pattern.* [Online]
Available at: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/dn589791(v=pandp.10)
[Accessed 20 7 2018].