INDIVIDUAL ASSIGNMENT

Project Title	**:**	Maersk Line Container Management System
Module Code	**:**	CT071-3-3-DDAC
Module Title	**:**	Designing and Developing Applications on The Cloud
Intake Code	**:**	UC3F1706SE
Student Name	**:**	Chew Rui Zhe
Student ID	**:**	TP038150
Lecturer's Name	**:**	Dr. Kalai Anand Ratnam
Hand in Date	**:**	13th April 2018

Acknowledgement

There is a group of people who provided their advice and guidance throughout this project until its completion. in this section, I would like to take this opportunity to express my gratitude to those who provide their supports in terms of knowledge and experience in this project.

Firstly, I would like to thank Asia Pacific University which provides the module of Designing and Developing Applications on the Cloud (CT071-3-3-DDAC) for the students to design, develop and deploy the cloud applications. Furthermore, I would like to thank Dr. Kalai Anand Ratnam for his guidance since the starting point of the project. His professional advice has given me a clear understanding upon my project requirement. this is very important as it would allow me to carry out my work more efficiently to produce a better report. His passion in cloud computing has helped me in improving my knowledge on deploying the web application to Microsoft Azure as a hosting server.

Next, I would like to thank my parents who supported me in the period of this project. They tried to motivate me mentally whenever I faced any issues that need more time to fix and cause sleepless nights. The financial support they provided is much appreciated especially on the basic daily needs. A good working environment they provided has bring me success to this project.

Table of Contents

1.0 Introduction

1.1 Project Background

Maersk Line was founded by A.P. Moller Maersk Group and it begins its operations in 1928. Maersk Line considered as the largest container shipping company globally with the largest operating unit of the Danish business conglomerate. Generally, Maersk Line consists of 374 offices in total, accommodating 116 countries worldwide with around 32,000 employees including sea farers and land-based people. Furthermore, the company operates more than 600 vessels with a capacity of 2.6 million TEU.

As the Maersk Line has grown its business to a maximum potential, an online cloud solution is needed for the company to support the large volume of business appears in Maersk Line. Here, an online cloud application is proposed to be implemented in Maersk Line to assist the company in managing their IT support to meet the demand of the company business. Microsoft Azure has been chosen as the hosting of the cloud computing application that provides a virtualized platform over the network globally to increase the capability of the Maersk Line system, from a physical local machines setup to a virtualized online system.

1.2 Objective

The objectives of the project are:

- To improve the business process of Maersk Line through the Microsoft Azure, a cloud computing service platform provided by Microsoft.
- To provide a reliable online system for the customer to manage the container bookings and routes.

1.3 Scope

The scope of this project is to develop a web application that manage the container booking process and deploy the web application to Microsoft Azure to allow integration and the company users to access the system, including the Admin and the Agent as the primary users.

1.4 Requirement Specification

The requirement specifications including the **maintainability** of the system where the web application should have the capability to perform upgrades and carry out maintenance tasks. Secondly, the **availability** of the system is one of the requirement specification where the web application is available to be used from time to time. Since the availability of the web application should be maintained at all time, **monitoring** is considered as one of the requirement specification to monitor the traffic of the system that allows problem identification and solutions to the problems. In order to meet the demand of the web application, **scalability** of the system is mandatory to be met.

1.5 Summary of Functions Specification

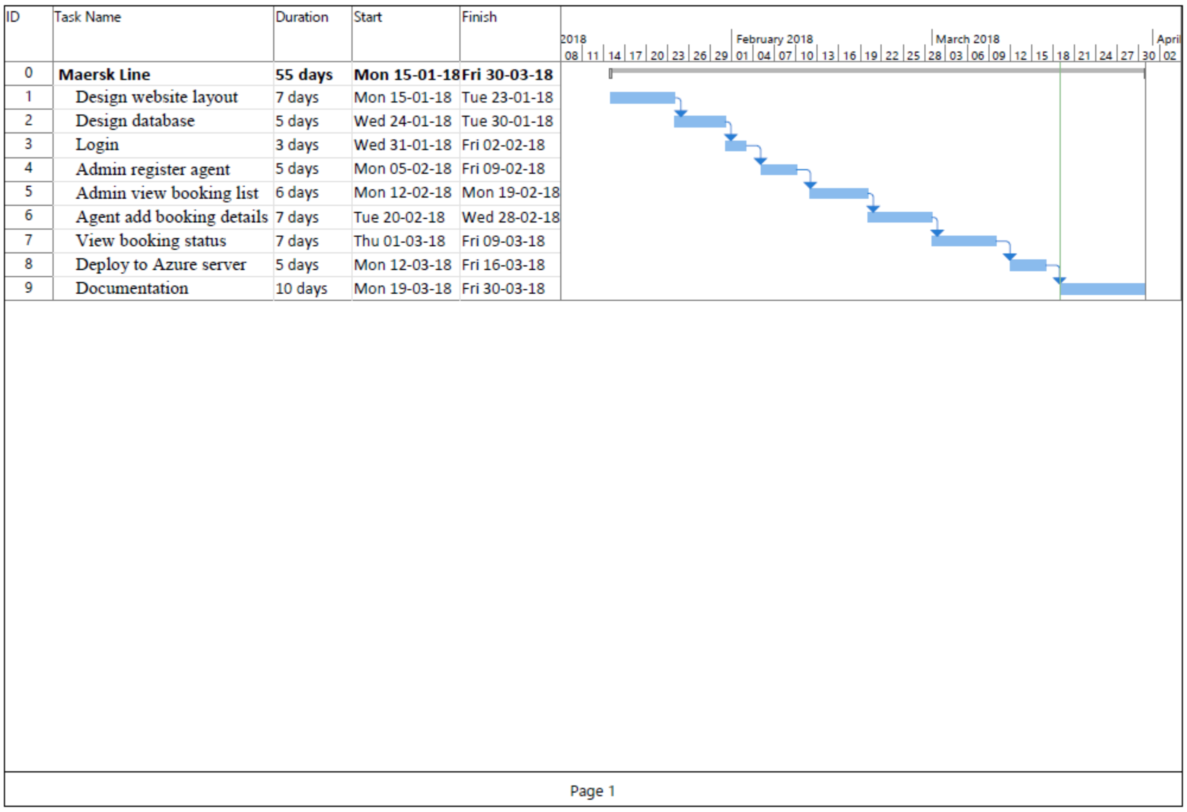
There are mainly two types of user who will be interacting with this web application: **Admin** and **Agent**. The Admin is needed to create profiles of agents, view all booking schedules including container management and update agent details. Next, the agents are required to manage bookings by creating bookings including item details, customer details, vessel details, view container profile and view booking schedule.

2.0 Project Plan

2.1 Tasks to be completed

Task Id	Task Names	Days	Start Date	End Date	Status
1	Design website layout	7	15/01/2018	23/01/2018	Completed
2	Design database	5	24/01/2018	30/01/2018	Completed
3	Login	3	31/01/2018	02/02/2018	Completed
4	Admin register agent	5	05/02/2018	09/02/2018	Completed
5	Admin view booking list	6	12/02/2018	19/02/2018	Completed
6	Agent add booking details	7	20/02/2018	28/02/2018	Completed
7	View booking status	7	1/03/2018	09/03/2018	Completed
8	Deploy to Azure server	5	12/03/2018	16/03/2018	Completed
9	Documentation	10	19/03/2018	30/03/2018	Completed

2.2 Gantt Chart



ID	Task Name	Duration	Start	Finish
0	**Maersk Line**	**55 days**	**Mon 15-01-18**	**Fri 30-03-18**
1	Design website layout	7 days	Mon 15-01-18	Tue 23-01-18
2	Design database	5 days	Wed 24-01-18	Tue 30-01-18
3	Login	3 days	Wed 31-01-18	Fri 02-02-18
4	Admin register agent	5 days	Mon 05-02-18	Fri 09-02-18
5	Admin view booking list	6 days	Mon 12-02-18	Mon 19-02-18
6	Agent add booking details	7 days	Tue 20-02-18	Wed 28-02-18
7	View booking status	7 days	Thu 01-03-18	Fri 09-03-18
8	Deploy to Azure server	5 days	Mon 12-03-18	Fri 16-03-18
9	Documentation	10 days	Mon 19-03-18	Fri 30-03-18

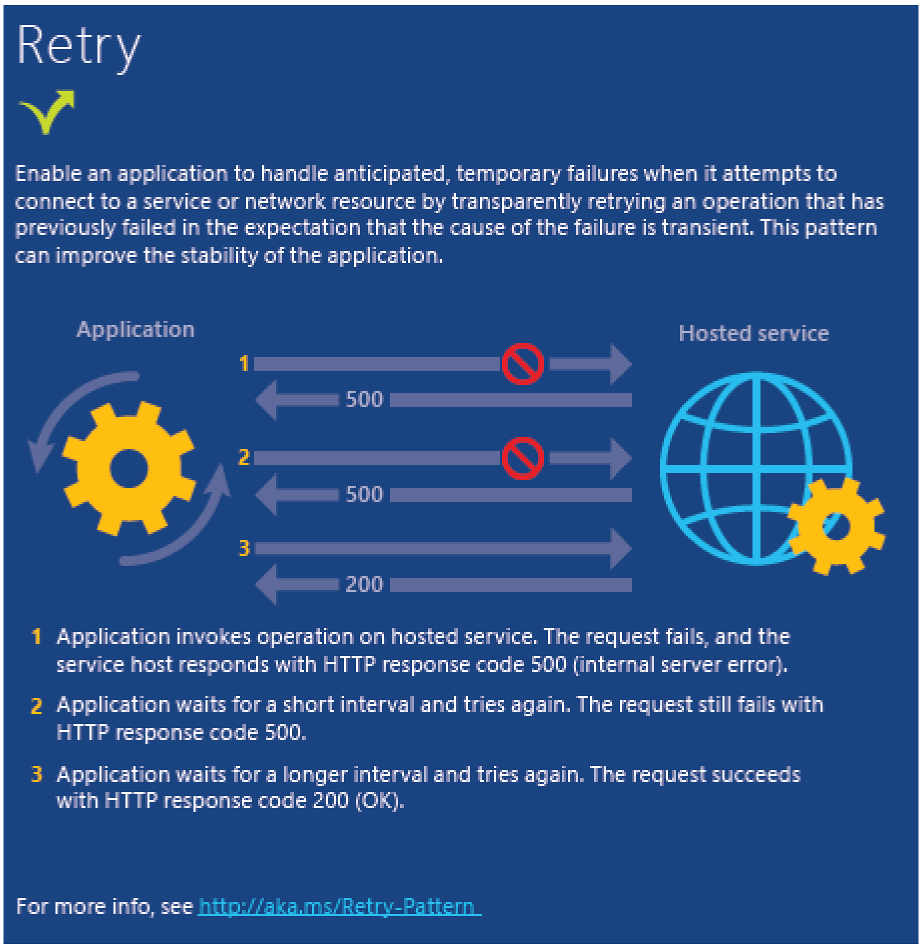
3.0 Design

3.1 Design Consideration

There are several **considerations** and **assumptions** made throughout the project planning phase for the Maersk Line Container Management System:

1. There will be only one admin account that handles the management of booking details and agent accounts, which includes registration of agents.
2. The system is capable of adding more admins, therefore the main controller will prompt an error right after the agent login into their accounts as the main controller is set to read a list of admin accounts.
3. The default booking status will set as "**Paid**" since there is no integration of payment system works with this project. As such, the admin will only be able to update the booking status once the booking has reached the destination. The booking will not be allowed to update for now.
4. RM850 Azure credit is provided for the developer to perform development activities and to show the validity system concept.

3.2 Cloud Design Patterns



Retry

Enable an application to handle anticipated, temporary failures when it attempts to connect to a service or network resource by transparently retrying an operation that has previously failed in the expectation that the cause of the failure is transient. This pattern can improve the stability of the application.

1 Application invokes operation on hosted service. The request fails, and the service host responds with HTTP response code 500 (internal server error).

2 Application waits for a short interval and tries again. The request still fails with HTTP response code 500.

3 Application waits for a longer interval and tries again. The request succeeds with HTTP response code 200 (OK).

For more info, see http://aka.ms/Retry-Pattern

Figure 1: Retry Pattern

The **Retry Pattern** will be implemented in the Maersk Line project to improve the online system's stability. Typically, this pattern is responsible in managing transient failures whenever it tries to establish connection to a network service or network resource (Narumoto, 2017). It is considered a self-correcting mechanism when the connectivity is loss between the system and the services. When the connection is failed, it will try again after some delays and ensure a higher success rate of connection. The Retry Pattern is suitable in the condition of interacting with a remote service or accessing a remote resource. There are three strategies exist in the pattern: **Cancel**, **Retry** and **Retry after delay**. An exception is caught when the fault shows that the failure is not transient and the operation should be cancel. Retry action is the specific fault that is unusual or rare which will be caused by corrupted network packet. Retry after delay considered connectivity or busy failures where the connectivity issues are repaired the application is put under a suitable time before sending the retry request again.

3.3 Cloud Architectural Diagram

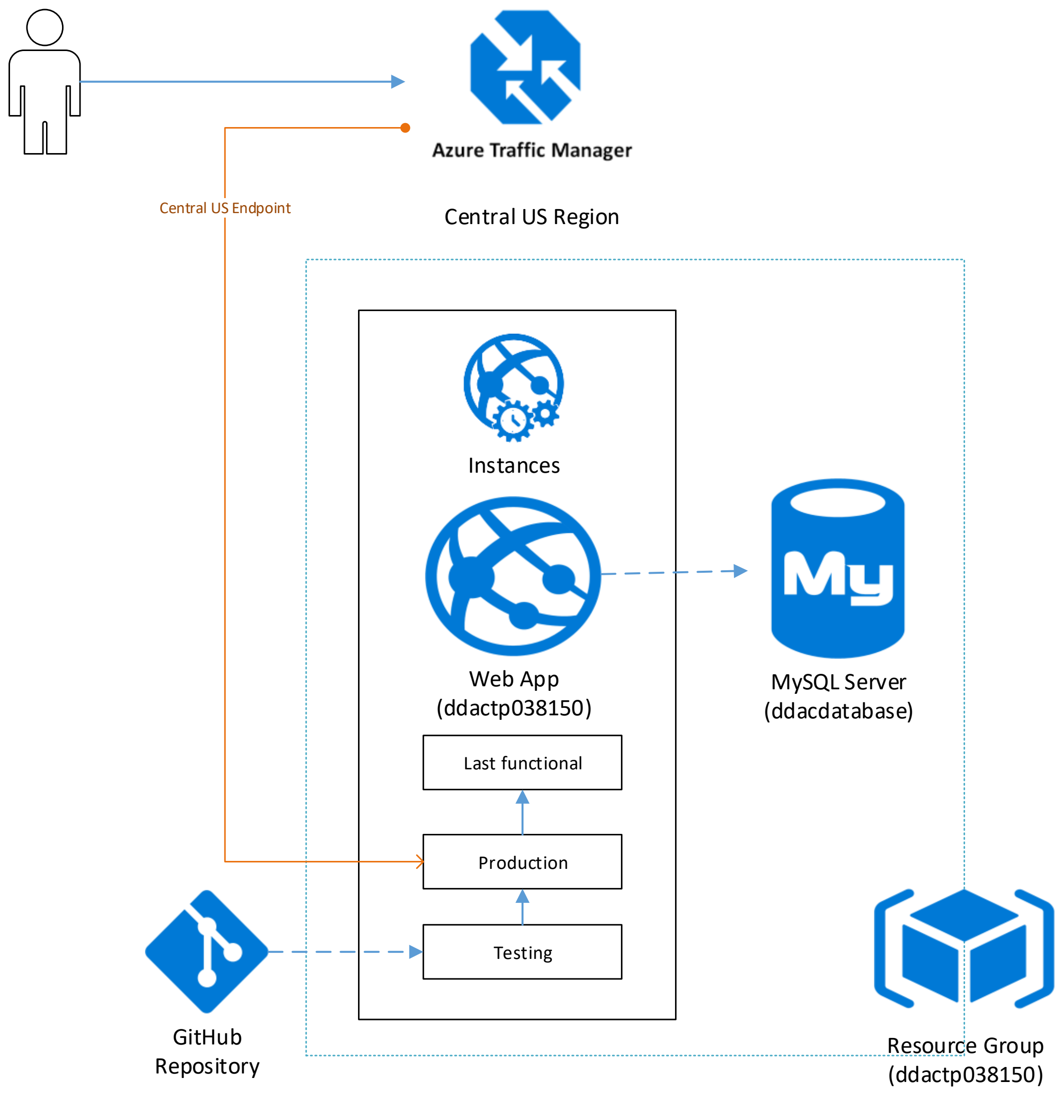


Figure 2: Architectural Diagram of the project

Figure 2 indicates the ideal architecture on how the Maersk Line CMS (ddactp038150) being deployed on the Azure web services. The first step is to deploy the completed web application on a local host server. Some configuration settings have to been done to connect the application to the Azure cloud resources and deploy it to the cloud. In addition, a MySQL database will be connected to the database server together with a setting up of a traffic manager that controls the user traffic distribution. The web application is being placed at the location of Central US. Although the current project has only one region, the **traffic manager is created for future use** when the business requirements expanded to other regions and the system might need more service plan to control the traffic usage.

Cost Estimation

Service	**Description**	**Cost Estimation (RM)**
Azure Database for MySQL Server	Gen 5 vCores x 2 + 5GB General Purpose Storage	302.05
App Service	S1 Standard Plan	312.48
Traffic Manager	Central US Region 2 Azure Endpoints	15.22
	Monthly Cost:	**629.75**
	Annual Cost:	**7557.00**

Table 1: Cost Estimation according to Figure 2 Architectural Diagram

3.4 Modelling

3.4.1 Use Case Diagram

Admin Module

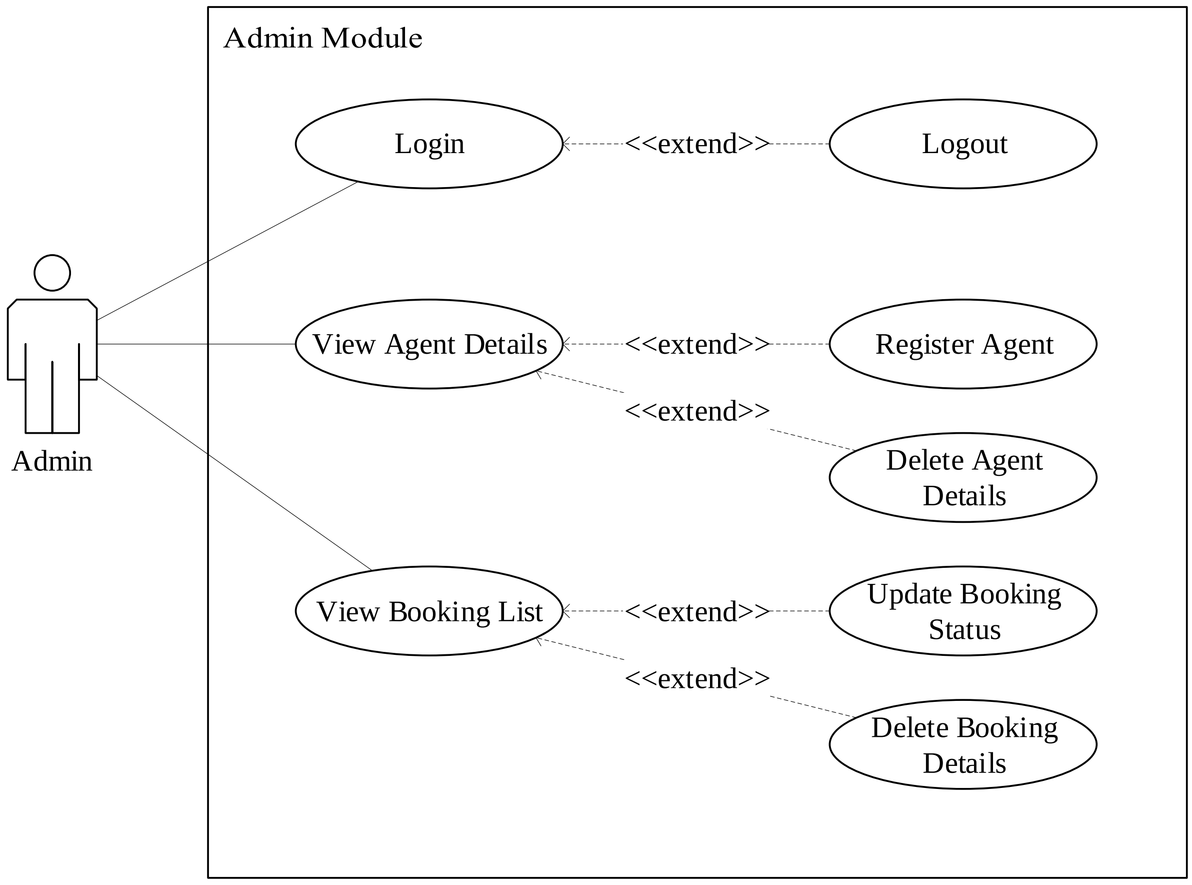


Figure 3: Maersk Line Admin Module Use Case Diagram

Agent Module

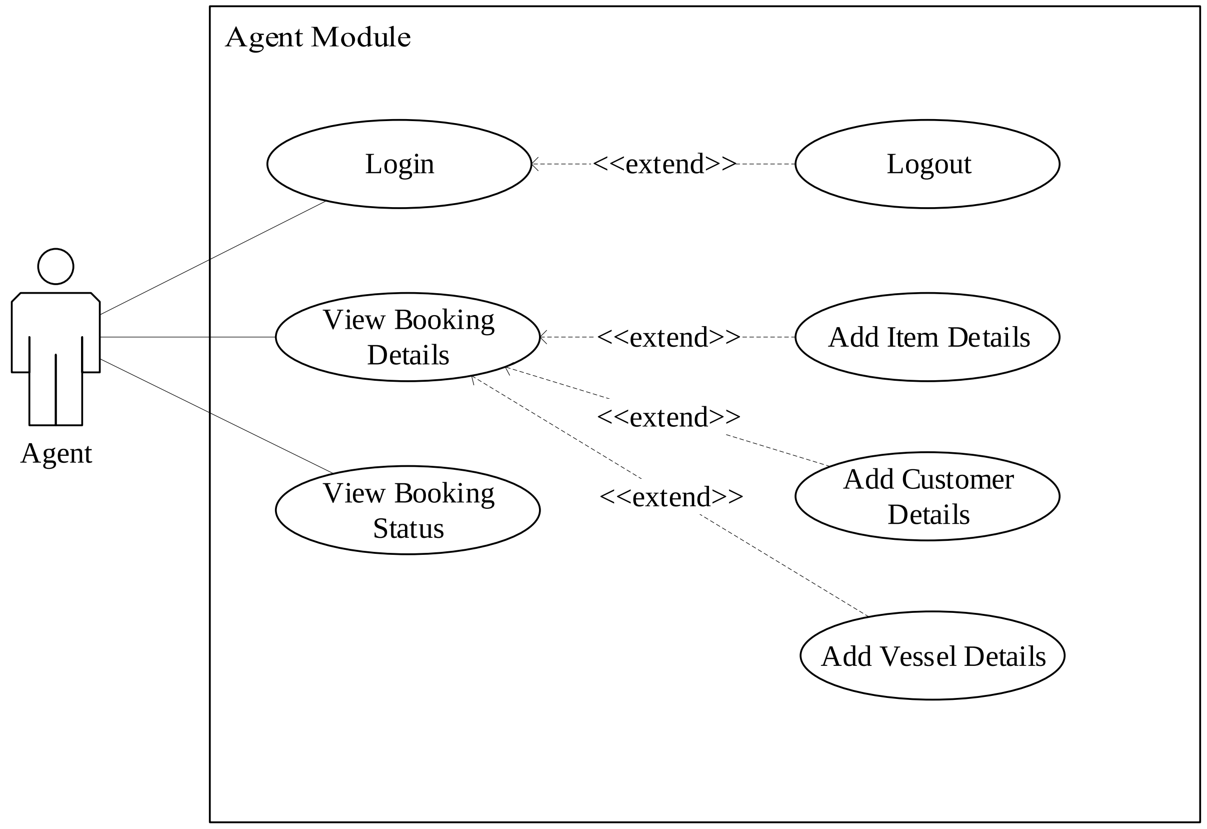


Figure 4: Maersk Line Agent Module Use Case Diagram

3.4.1.1 Admin Module Use Case Specification

Use Case Number	1
Use Case Name	Login
Description	Allows admin to login to the system.
Pre-Conditions	The admin account is already created.
Post-Conditions	System shows admin page.
Primary Actor	Admin
Trigger	Enter the Maersk Line login URL.
Main Scenario	1. Admin enters the website.
	2. Admin keys in admin username and password.

Use Case Number	2
Use Case Name	Logout
Description	Allows admin to log out from the system.
Pre-Conditions	Admin has logged in to the system.
Post-Conditions	Admin successfully logout from the system.
Primary Actor	Admin
Trigger	Admin clicks Logout.
Main Scenario	1. Admin has done the actions that he or she wanted to do.
	2. Admin clicks Logout button.

Use Case Number	3
Use Case Name	View Agent Details
Description	Allows admin to view a list of agent accounts.
Pre-Conditions	Admin already successfully logged in to the system.
Post-Conditions	System shows a data table that consists of agent accounts.
Primary Actor	Admin
Trigger	Admin clicks Agent Management.
Main Scenario	1. Admin clicks Agent Management button.
	2. The agent list becomes visible.

Use Case Number	4
Use Case Name	Register Agent
Description	Allows admin to register agent account.
Pre-Conditions	Admin already successfully logged in to the system.
Post-Conditions	System shows agent account registered successful.
Primary Actor	Admin
Trigger	Admin clicks Agent Management.
Main Scenario	1. Admin clicks Add button.
	2. Admin keys in agent username, agent password, agent name, agent contact, and agent address.
	3. Admin clicks Insert button after all the fields are filled up.

Use Case Number	5
Use Case Name	Delete Agent
Description	Allows admin to remove agent account.
Pre-Conditions	1. Admin already successfully logged in to the system. 2. Agent account is existed in the database.
Post-Conditions	System shows agent account removed successful.
Primary Actor	Admin
Trigger	Admin clicks Agent Management.
Main Scenario	1. Admin clicks Delete button on the desired agent account.
	2. Admin clicks confirmation button.

Use Case Number	6
Use Case Name	View Booking List
Description	Allows admin to view a list of bookings registered by agent.
Pre-Conditions	Booking lists has added by the agents.
Post-Conditions	System shows a list of booking details.
Primary Actor	Admin
Trigger	Admin clicks Booking Management.
Main Scenario	1. Admin select Booking Management.
	2. The booking list becomes visible.

Use Case Number	7
Use Case Name	Update Booking Status
Description	Allows admin to update the status of the bookings.
Pre-Conditions	Booking lists has added by the agents.
Post-Conditions	System shows either the booking is paid or shipped.
Primary Actor	Admin
Trigger	Admin clicks Booking Management.
Main Scenario	1. Admin clicks Update button on the desired booking ID.
	2. Admin clicks either paid or shipped as the status of the booking.

Use Case Number	8
Use Case Name	Delete Booking Details
Description	Allows admin to remove the bookings.
Pre-Conditions	Booking lists has added by the agents.
Post-Conditions	System removes the booking details.
Primary Actor	Admin
Trigger	Admin clicks Booking Management.
Main Scenario	1. Admin clicks Delete button on the desired booking ID.
	2. Admin clicks confirmation button.

3.4.1.2 Agent Module Use Case Specification

Use Case Number	1
Use Case Name	Login
Description	Allows agent to login to the system.
Pre-Conditions	The agent account is already created.
Post-Conditions	System shows agent page.
Primary Actor	Agent
Trigger	Enter the Maersk Line login URL.
Main Scenario	1. Agent enters the website.
	2. Agent keys in admin username and password.

Use Case Number	2
Use Case Name	Logout
Description	Allows agent to log out from the system.
Pre-Conditions	Agent has logged in to the system.
Post-Conditions	Agent successfully logout from the system.
Primary Actor	Agent
Trigger	Agent clicks Logout.
Main Scenario	1. Agent has done the actions that he or she wanted to do.
	2. Agent clicks Logout button.

Use Case Number	3
Use Case Name	View Booking Details
Description	Allows agent to view bookings using the system.
Pre-Conditions	Agent has logged in to the system.
Post-Conditions	System shows a list of booking details.
Primary Actor	Agent
Trigger	Agent views booking details.
Main Scenario	1. Agent clicks Add / View Booking button.
	2. Booking list table is visible.

Use Case Number	4
Use Case Name	Add Item Details
Description	Allows agent to add booking details into the system.
Pre-Conditions	Agent clicks Add / View Booking button.
Post-Conditions	System shows that the item details has successfully added.
Primary Actor	Agent
Trigger	Agent clicks Add button.
Main Scenario	1. Agent fill in the item name, item weight, item quantity and item description.
	2. Admin clicks Insert button after all the fields are filled up.

Use Case Number	5
Use Case Name	Add Customer Details
Description	Allows agent to add customer details into the system.
Pre-Conditions	Agent clicks Add / View Booking button.
Post-Conditions	System shows that the customer details has successfully added.
Primary Actor	Agent
Trigger	Agent clicks Add button.
Main Scenario	1. Agent fill in the customer name, customer contact, customer email and customer address.
	2. Admin clicks Insert button after all the fields are filled up.

Use Case Number	6
Use Case Name	Add Vessel Details
Description	Allows agent to add vessel details into the system.
Pre-Conditions	Agent clicks Add / View Booking button.
Post-Conditions	System shows that the vessel details has successfully added.
Primary Actor	Agent
Trigger	Agent clicks Add button.
Main Scenario	1. Agent choose the drop-down menu of vessel, harbor, terminal and schedule.
	2. Admin clicks Insert button after all the fields are filled up.

Use Case Number	7
Use Case Name	View Booking Status
Description	Allows agent to view booking status.
Pre-Conditions	Agent has logged in to the system.
Post-Conditions	System shows a list of booking status according to the booking ID.
Primary Actor	Agent
Trigger	Agent clicks View Booking Status button.
Main Scenario	1. Agent selects the view booking status.
	2. The list of booking status becomes visible.

3.4.2 Sequence Diagram

3.4.2.1 Admin Login Sequence Diagram

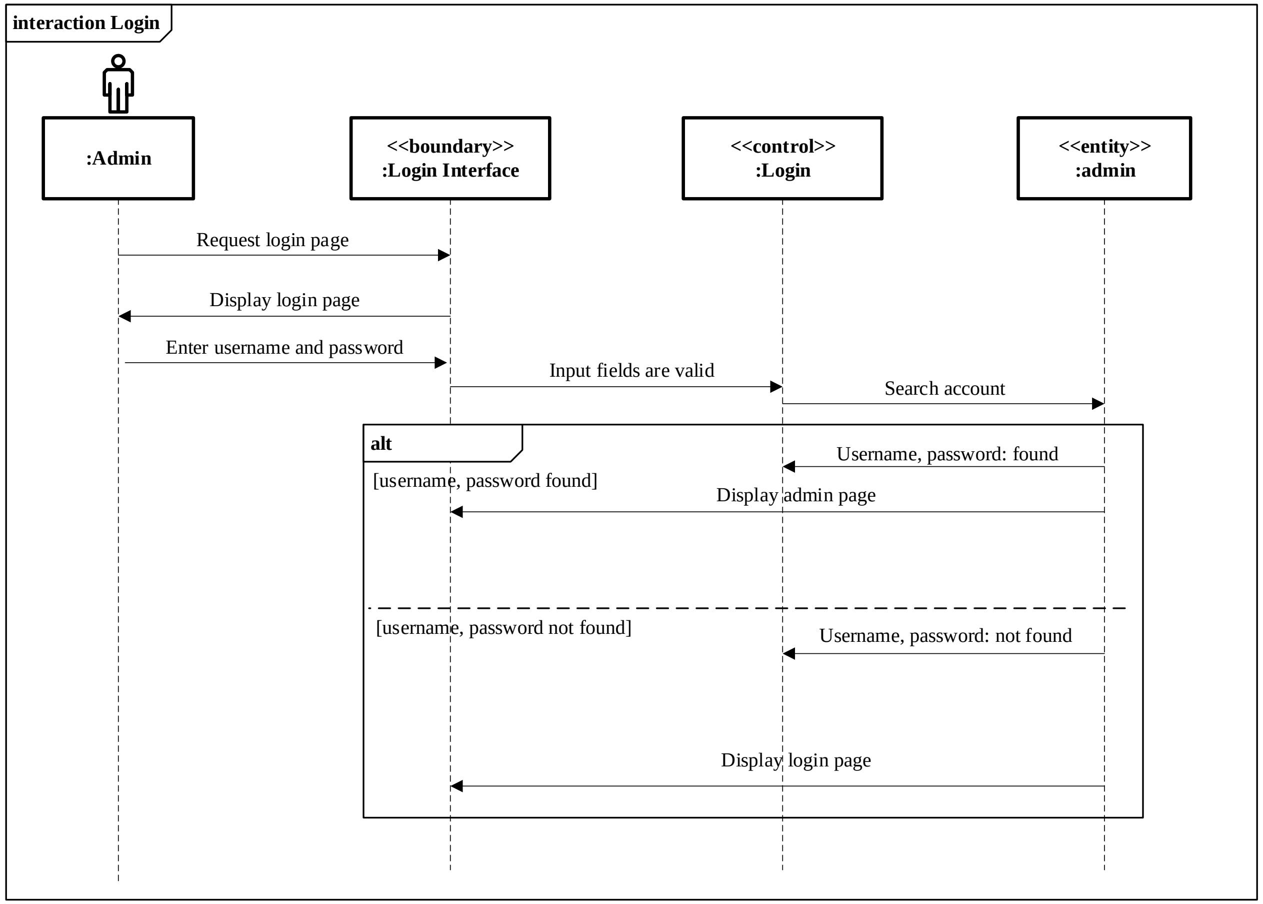


Figure 5: Admin Login Sequence Diagram

3.4.2.2 View Agent Details Sequence Diagram

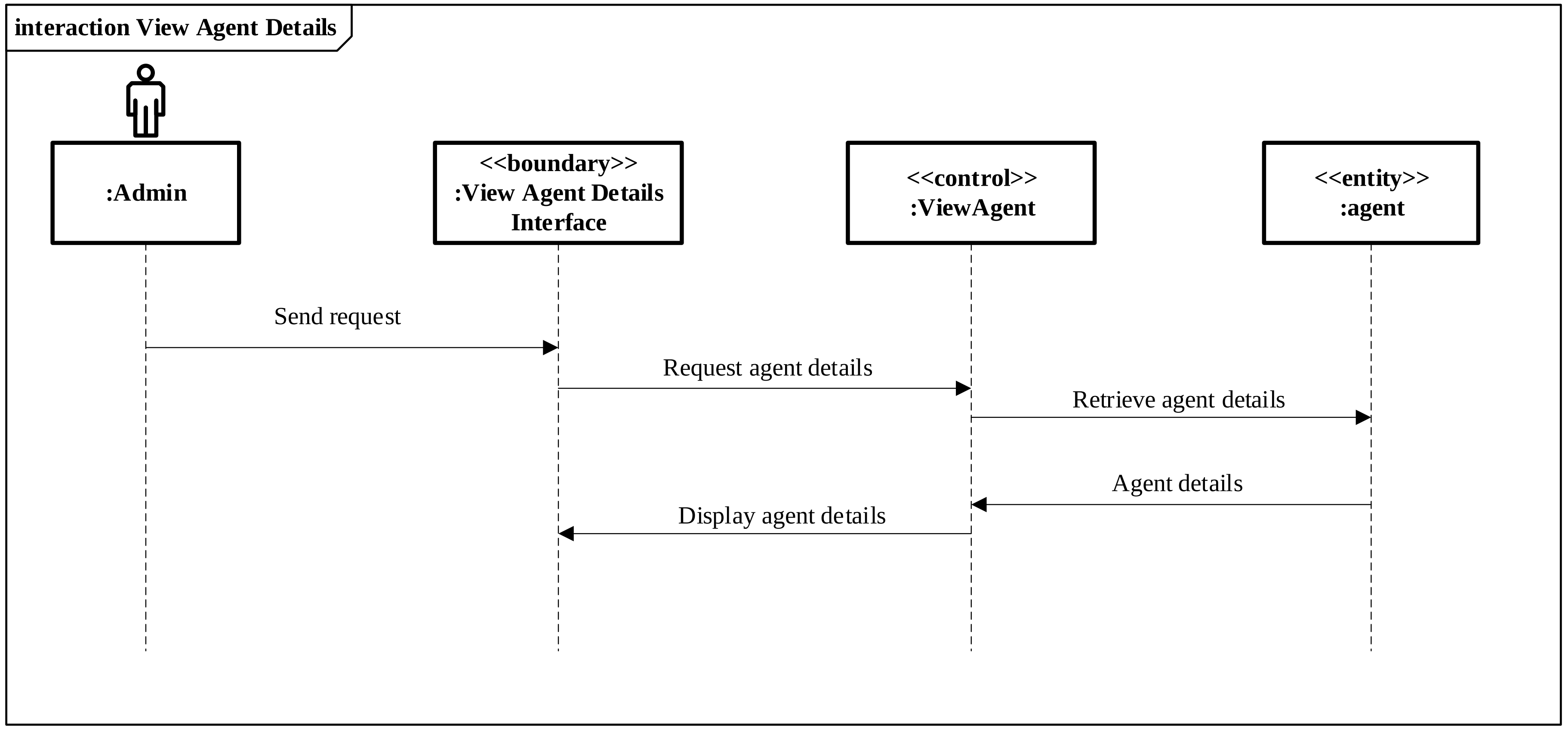


Figure 6: View Agent Details Sequence Diagram

3.4.2.3 Register Agent Sequence Diagram

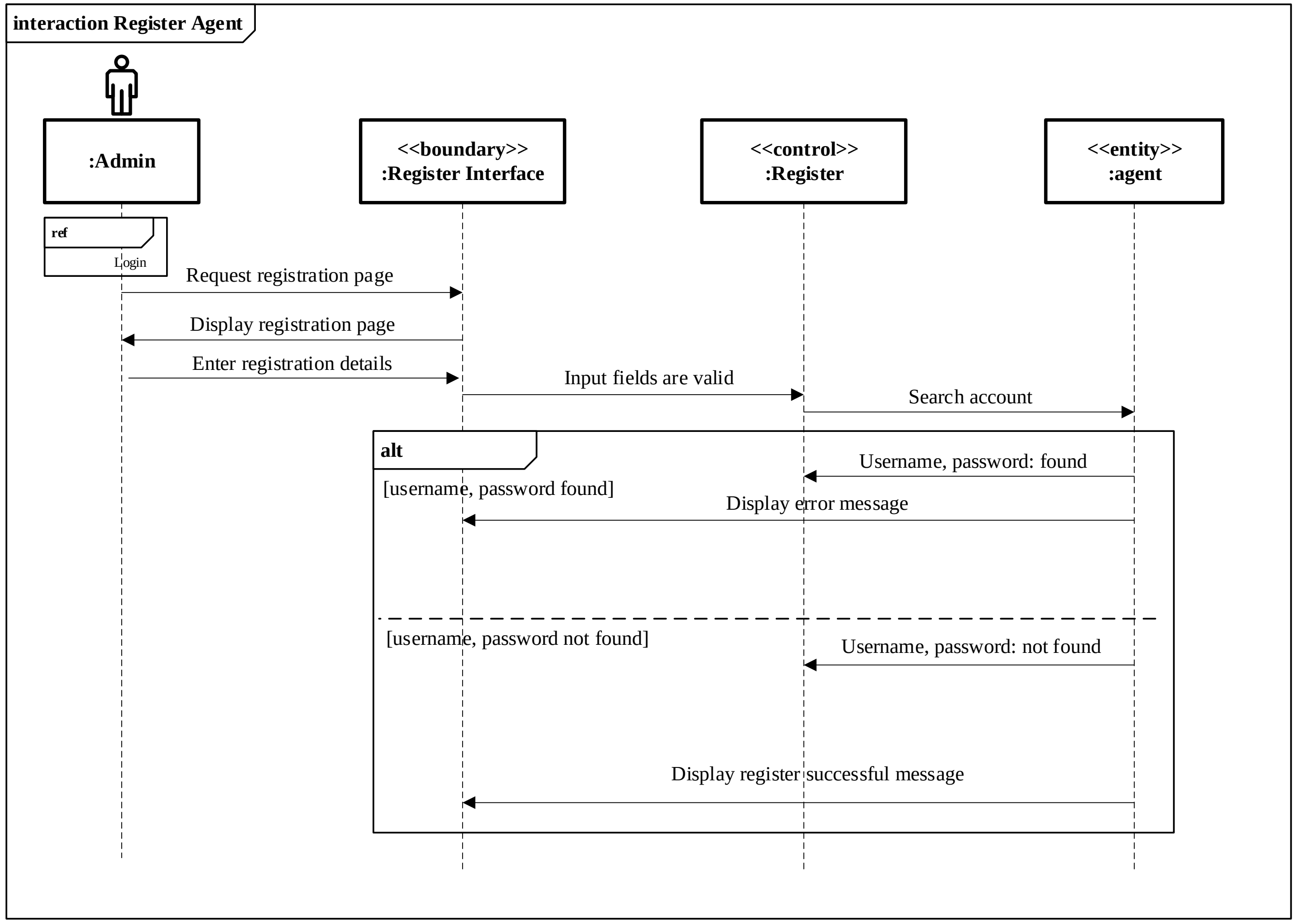

interaction Register Agent
:Admin
<<boundary>> :Register Interface
<<control>> :Register
<<entity>> :agent
ref
Login
Request registration page
Display registration page
Enter registration details
Input fields are valid
Search account
alt
Username, password: found
[username, password found]
Display error message
[username, password not found]
Username, password: not found
Display register successful message

Figure 7: Register Agent Sequence Diagram

3.4.2.4 Delete Agent Details Sequence Diagram

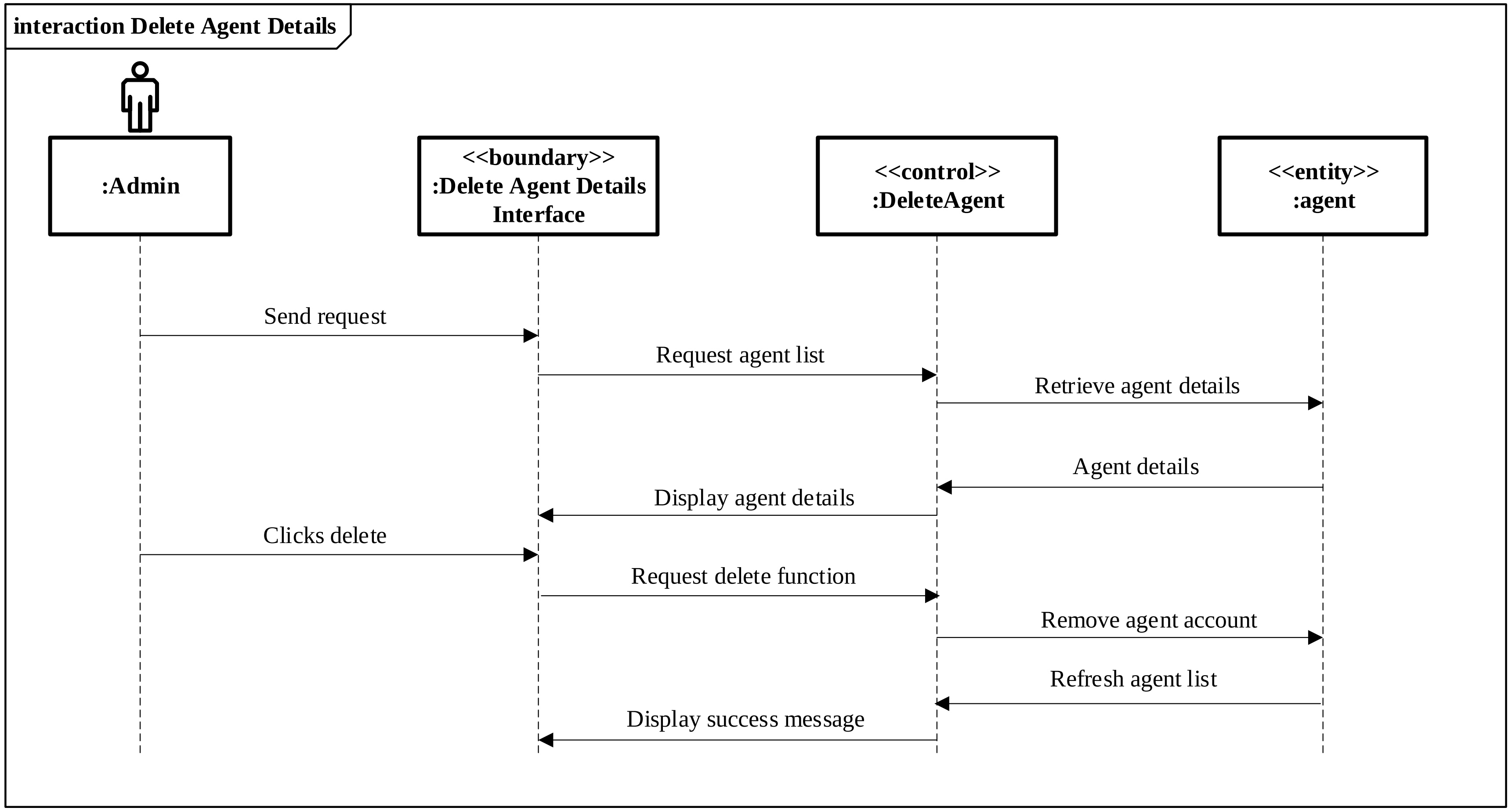


Figure 8: Delete Agent Details Sequence Diagram

3.4.2.5 View Booking List Sequence Diagram

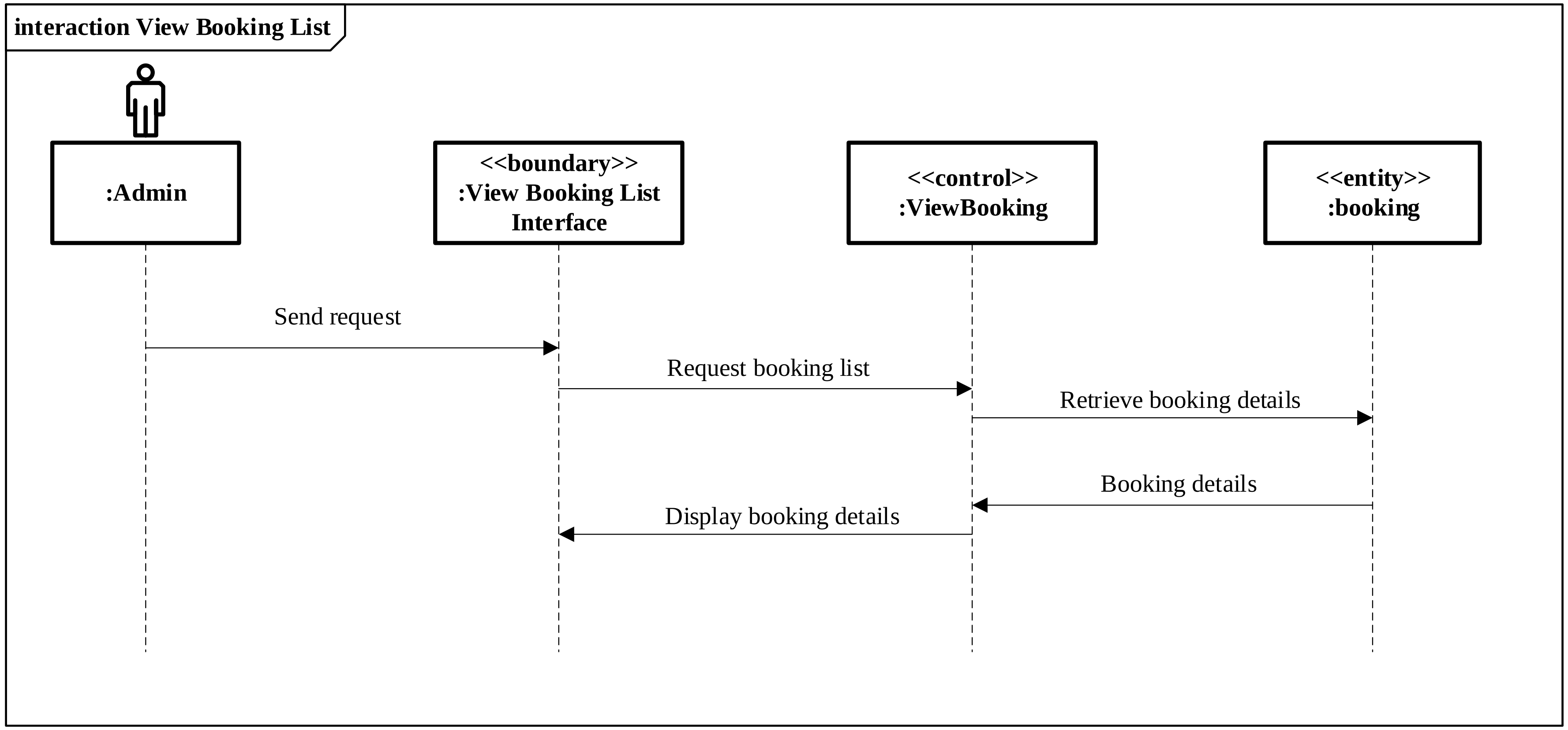


Figure 9: View Booking List Sequence Diagram

3.4.2.6 Update Booking Status Sequence Diagram

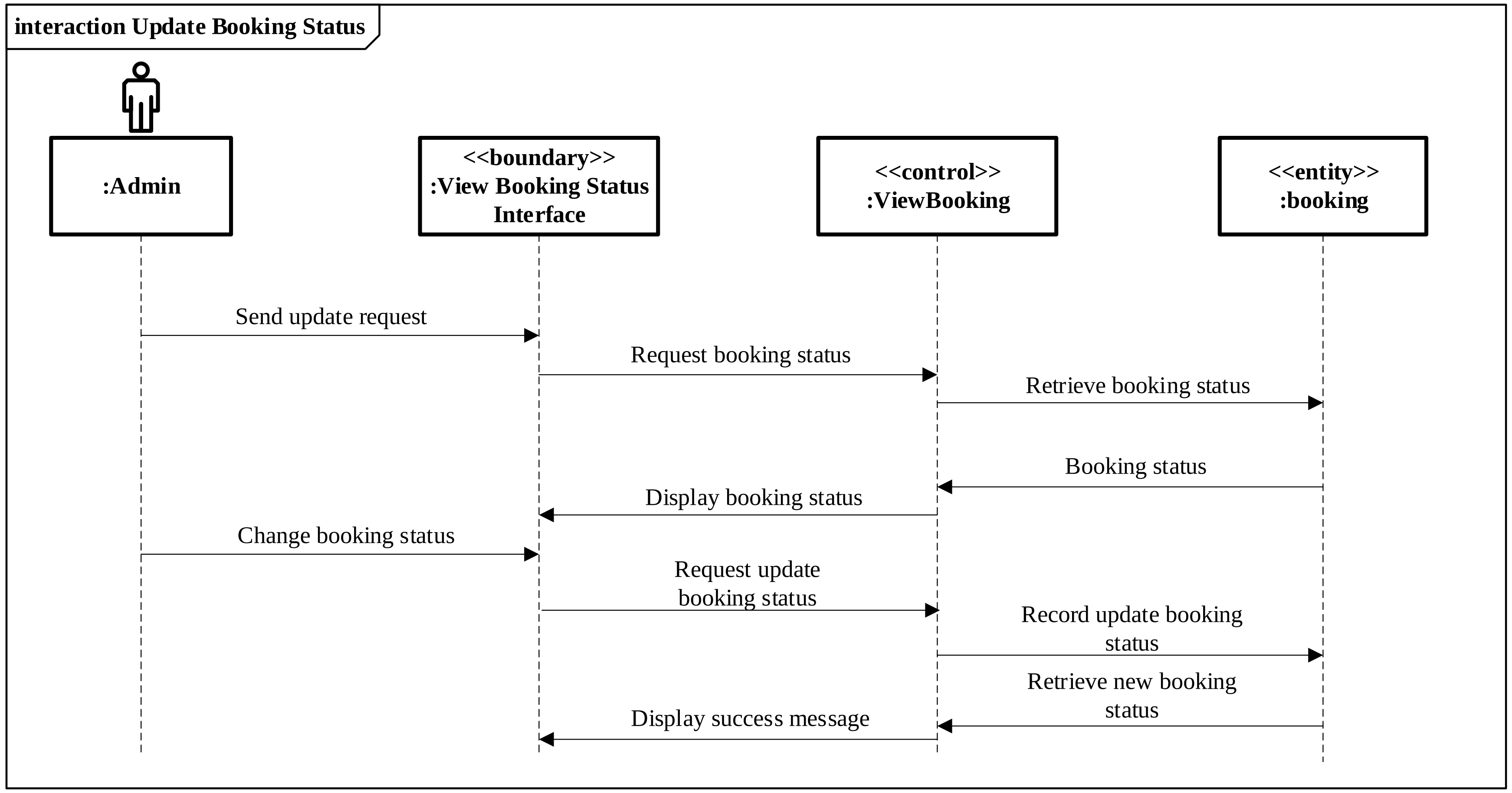


Figure 10: Update Booking Status Sequence Diagram

3.4.2.7 Delete Booking Details Sequence Diagram

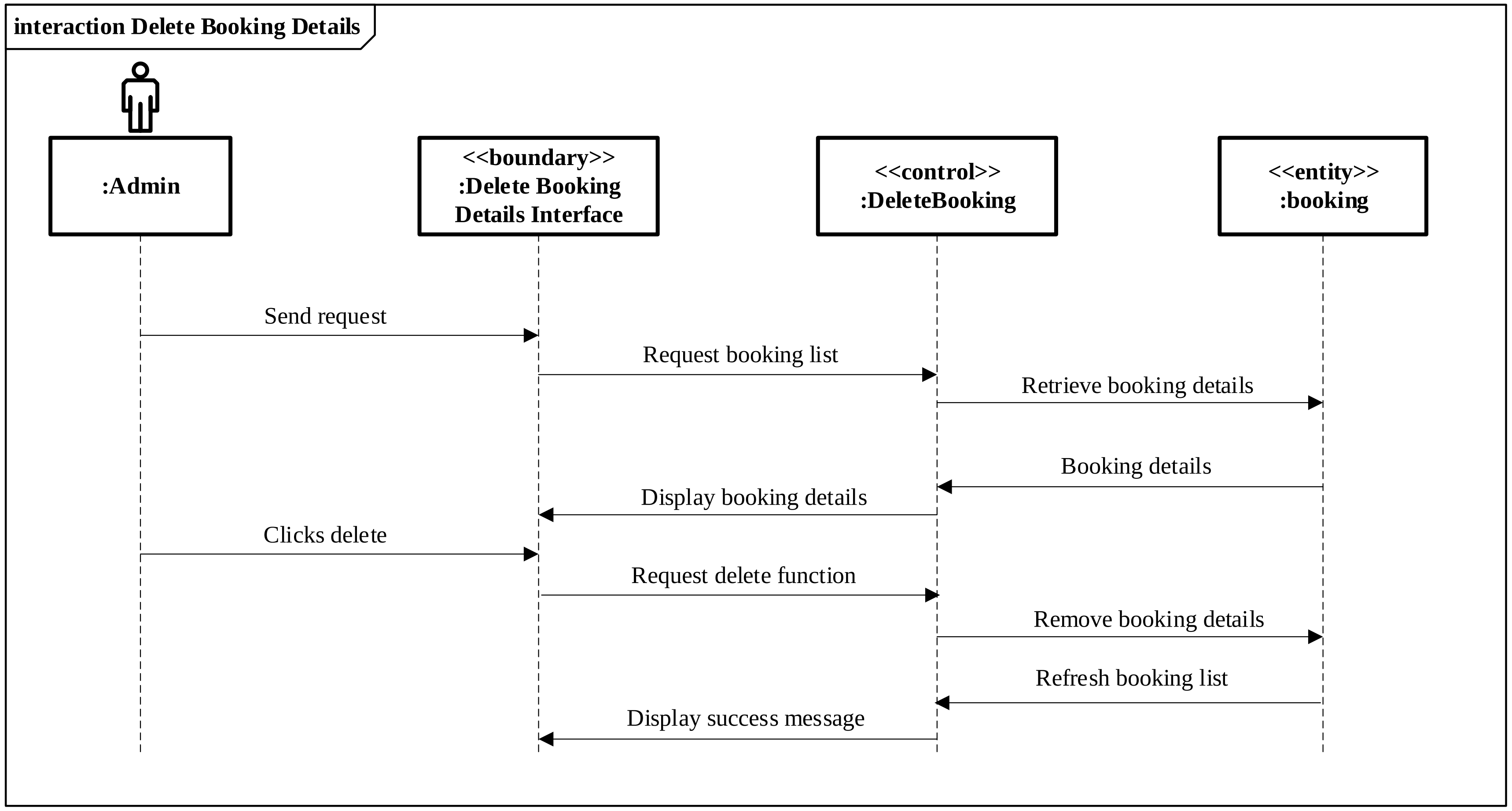


Figure 11: Delete Booking Details Sequence Diagram

3.4.2.8 Agent Login Sequence Diagram

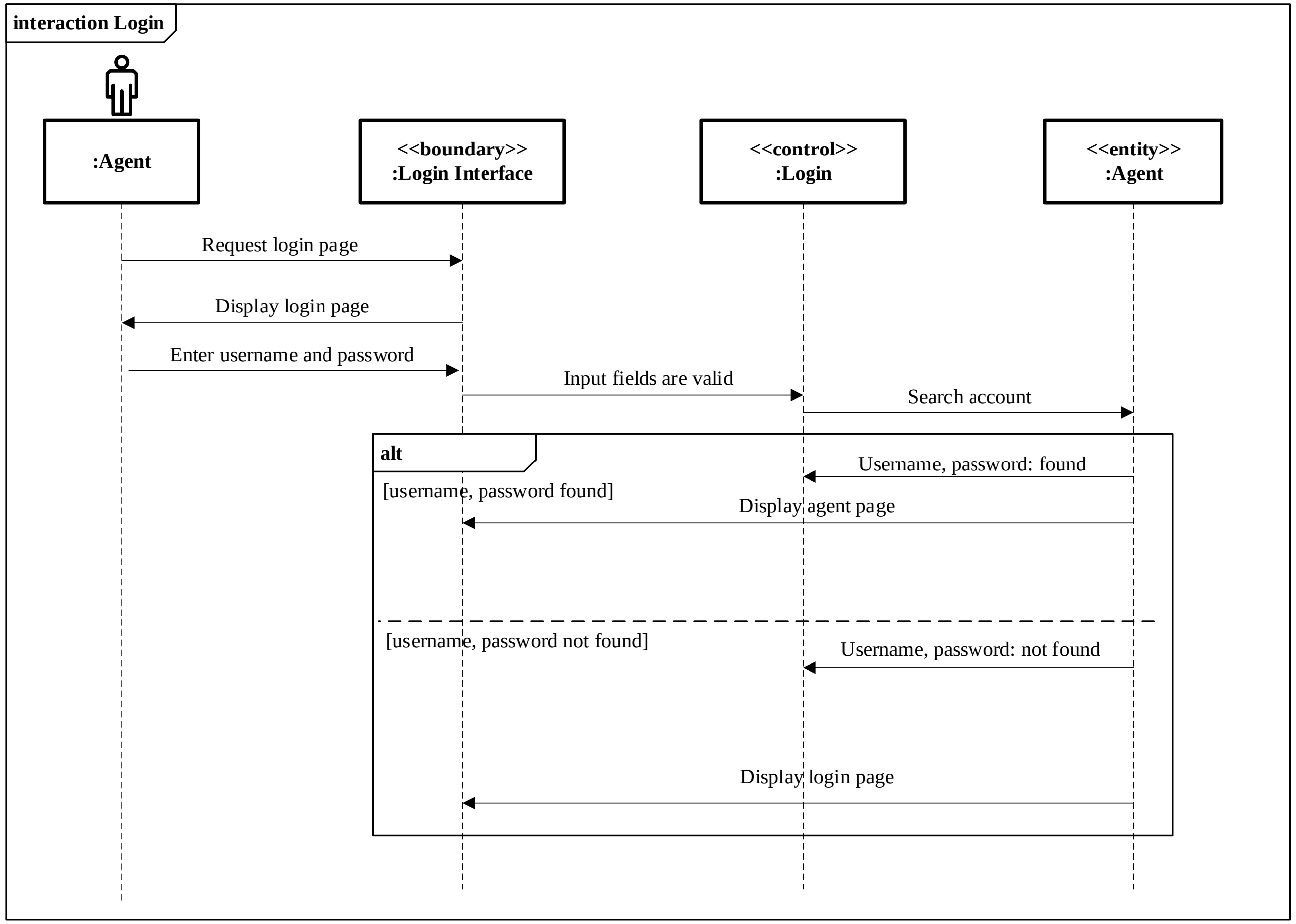


Figure 12: Agent Login Sequence Diagram

3.4.2.9 View Booking Details Sequence Diagram

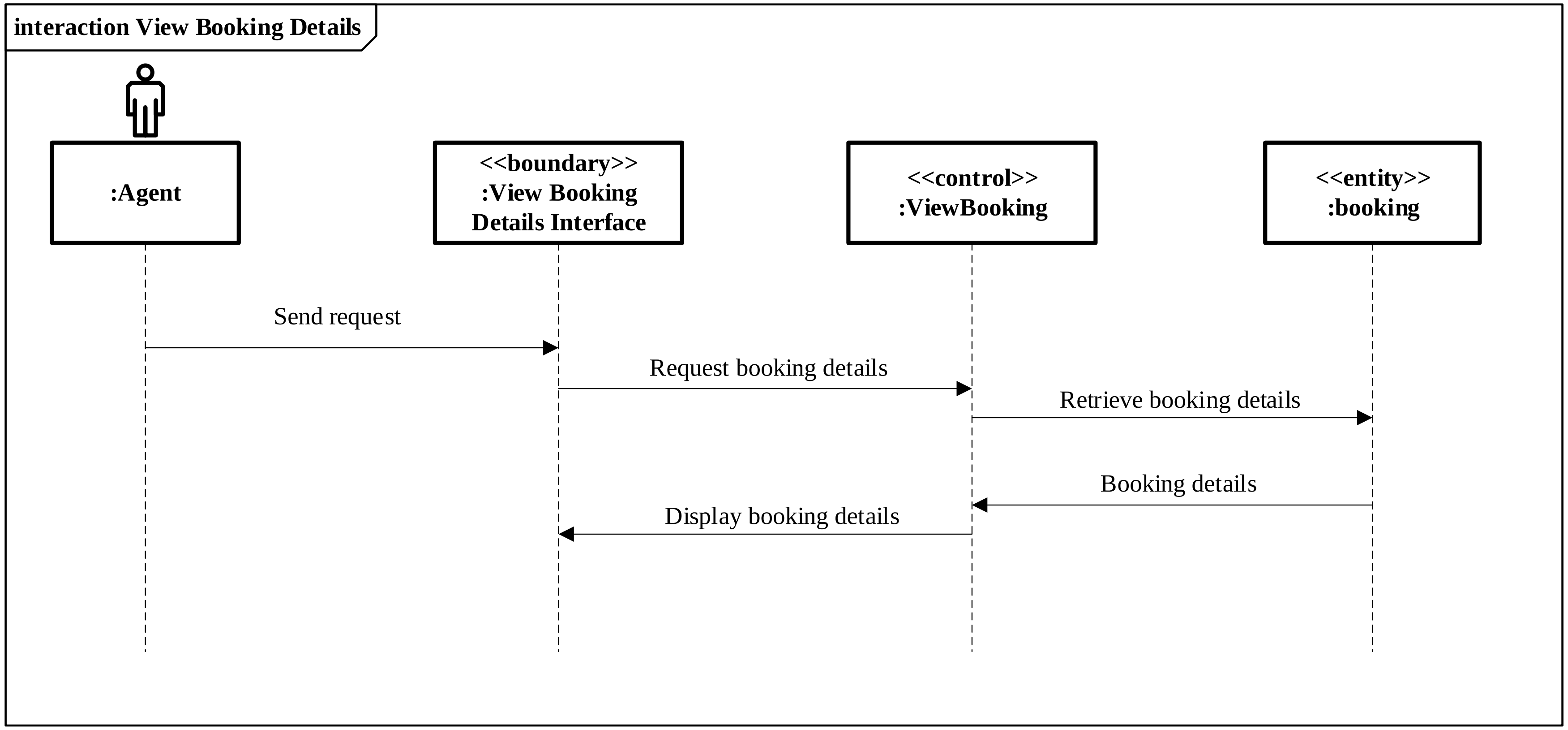


Figure 13: View Booking Details Sequence Diagram

3.4.2.10 Add Item Details Sequence Diagram

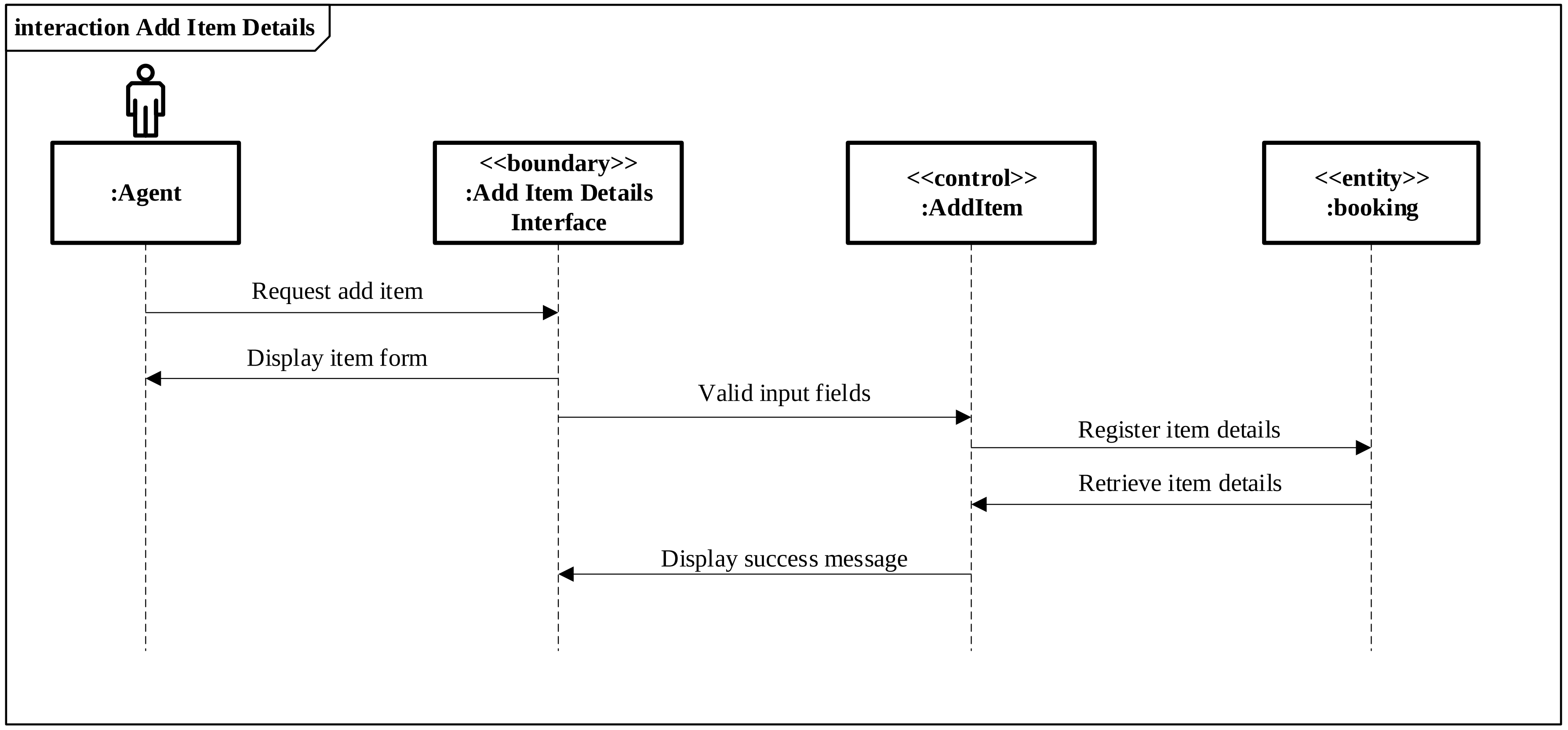


Figure 14: Add Item Details Sequence Diagram

3.4.2.11 Add Customer Details Sequence Diagram

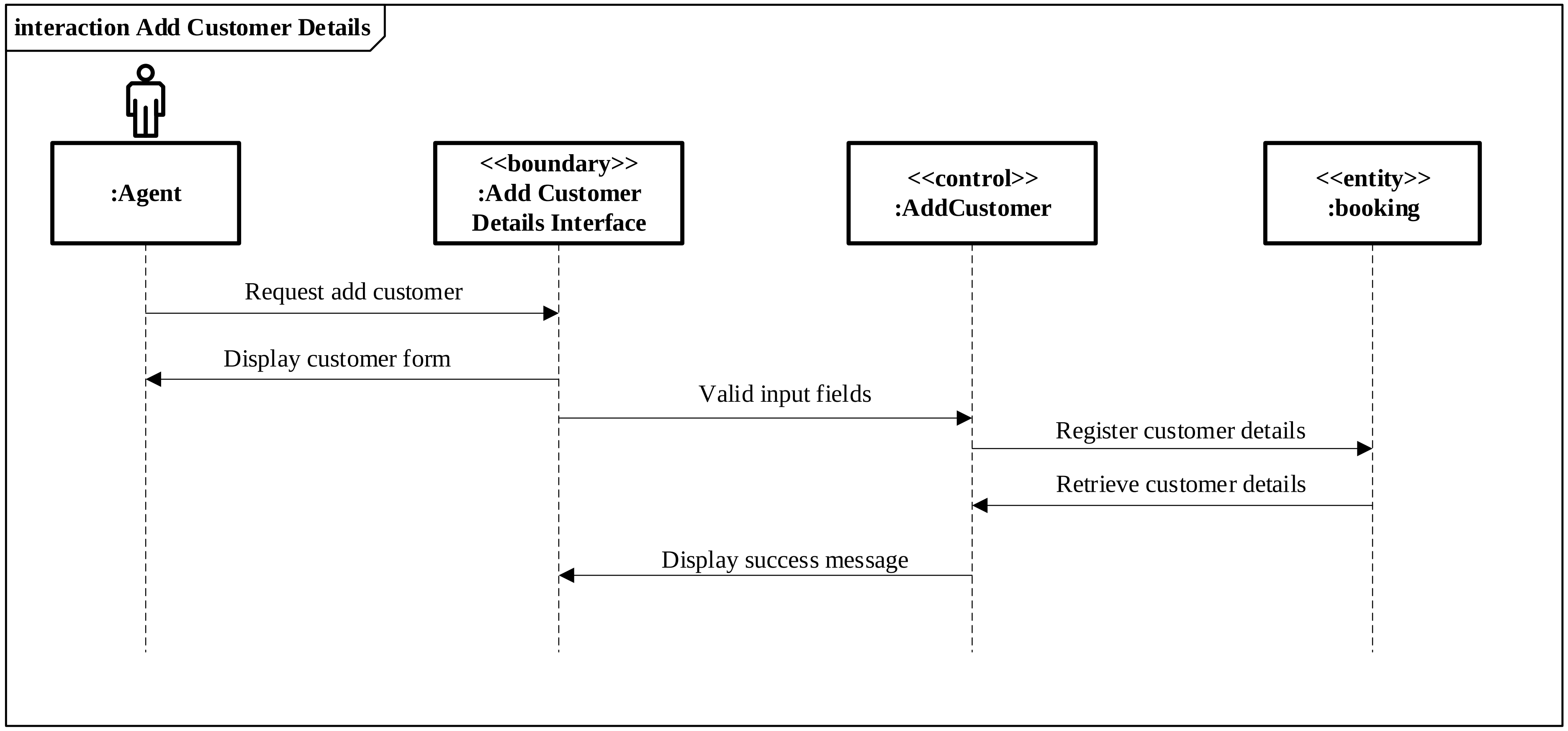


Figure 15: Add Customer Details Sequence Diagram

3.4.2.12 Add Vessel Details Sequence Diagram

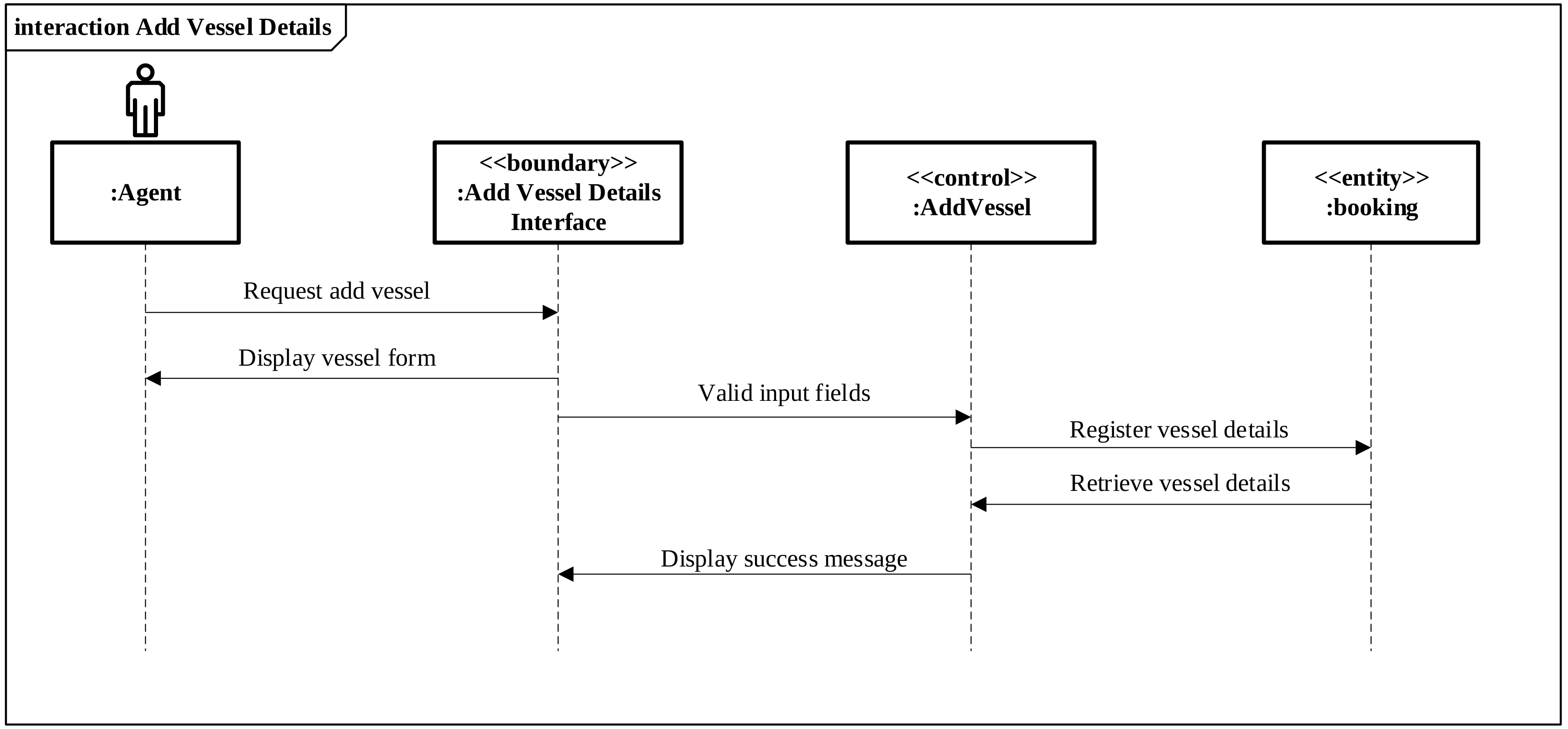


Figure 16: Add Vessel Details Sequence Diagram

3.4.2.13 View Booking Status Sequence Diagram

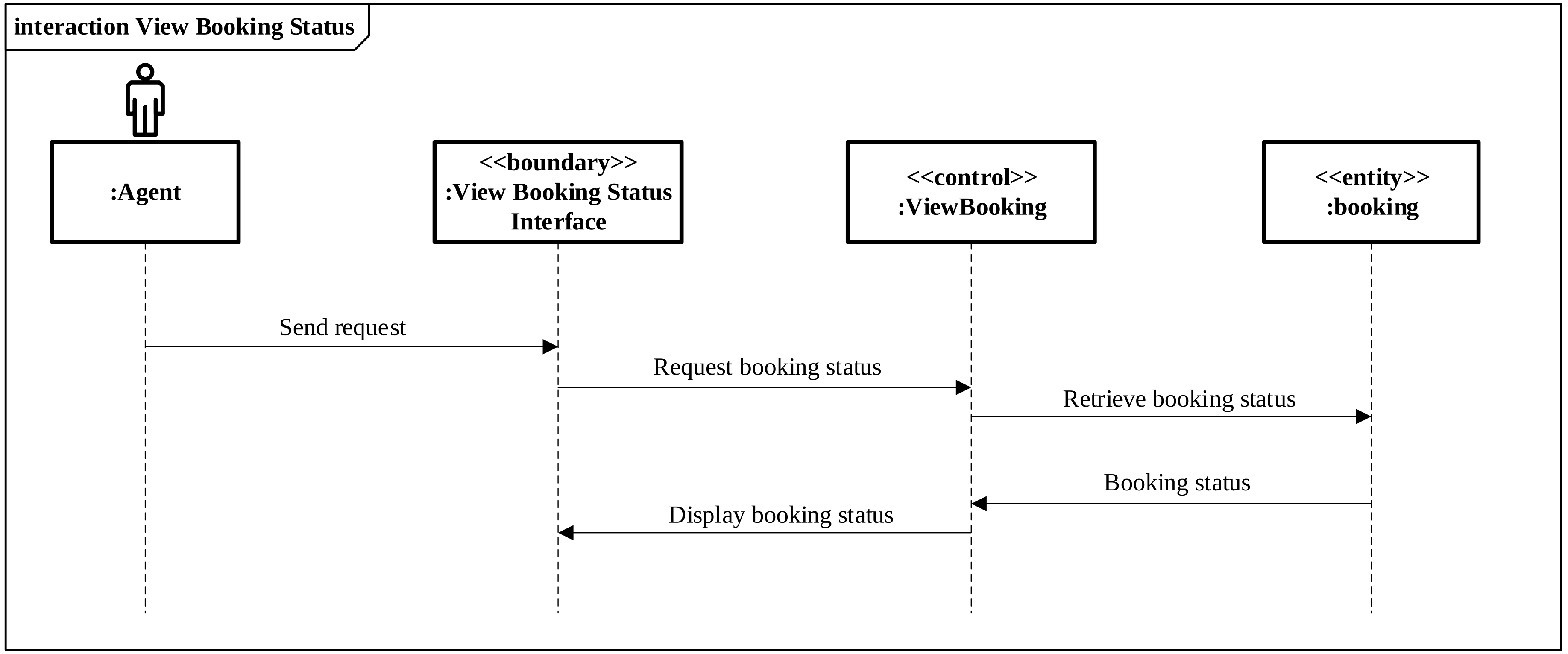


Figure 17: View Booking Status Sequence Diagram

3.4.3 Entity Relationship Diagram

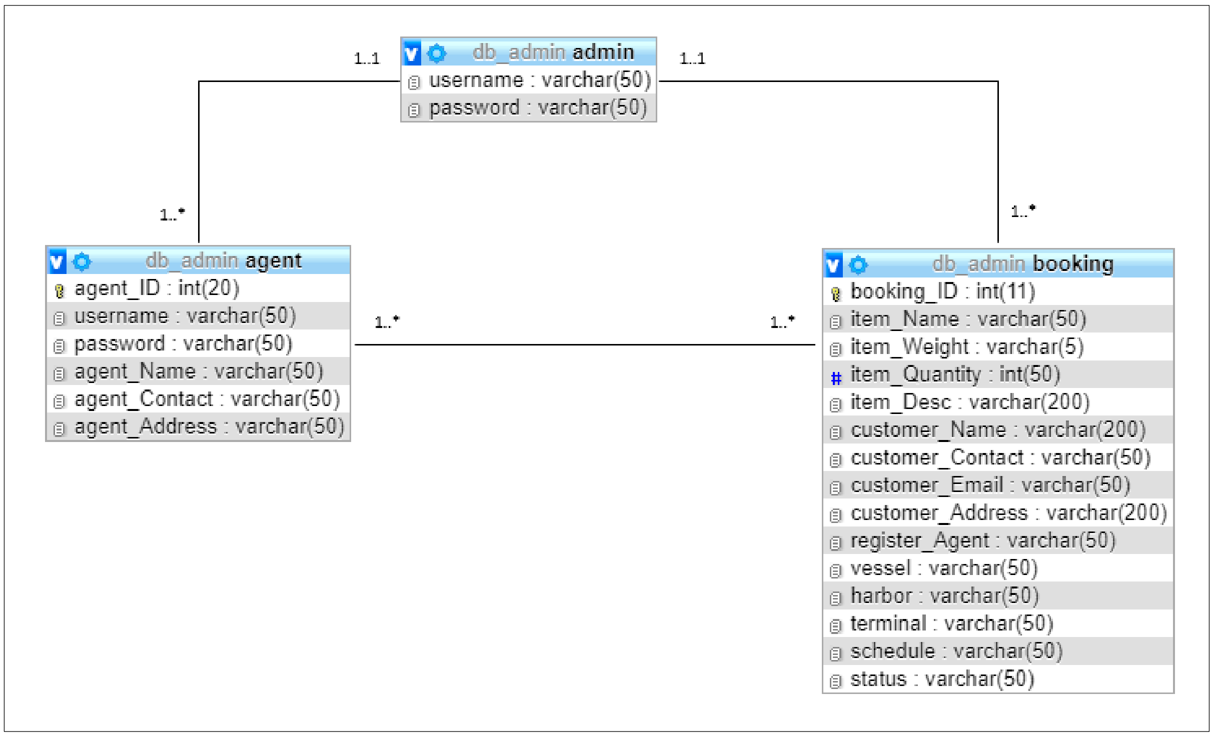


Figure 18: Entity Relationship Diagram

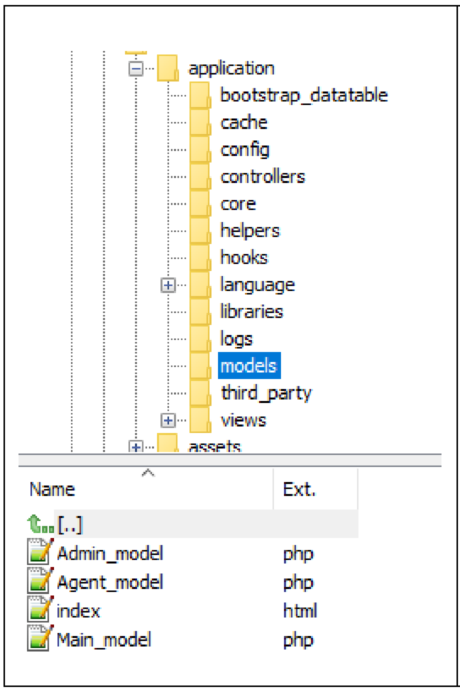
4.0 Implementation

4.1 Application Development

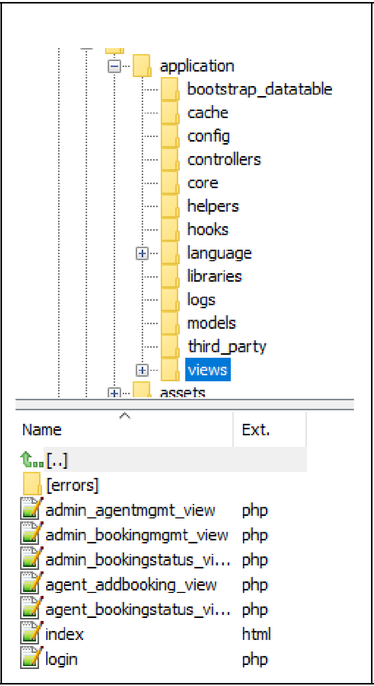
This section explains the development concept using frameworks and databases to develop the Maersk Line CMS.

<u>Selection of Programming Language and Database Language</u>

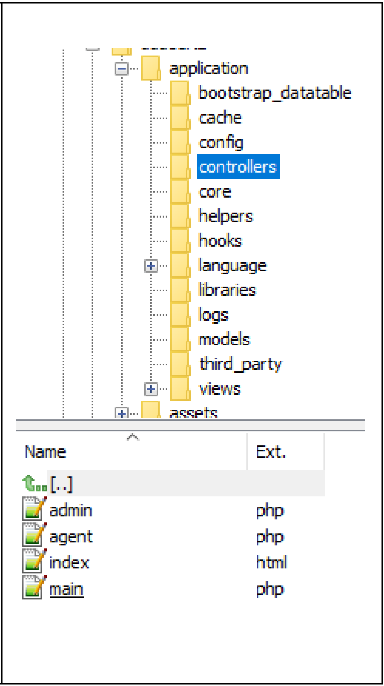
The developer has completed the development of Maersk Line CMS using **PHP CodeIgniter Framework** and **MySQL** as the database for the system. Typically, this application implements the concept of Model-View-Controller structure with the integration of **AdminLTE** themes that allows the system to be more organized using datatables. The Models contain all methods that operate on the data that it writes the data into the database. The Views contain the codes for the user interface where it interprets the user actions. On the other hand, Controller works between Model and View where it calls the method from view to controller as shown in Table 1 (kth, 2018).



```
application
    bootstrap_datatable
    cache
    config
    controllers
    core
    helpers
    hooks
    language
    libraries
    logs
    models
    third_party
    views
assets

Name            Ext.
[..]
Admin_model     php
Agent_model     php
index           html
Main_model      php
```



```
application
    bootstrap_datatable
    cache
    config
    controllers
    core
    helpers
    hooks
    language
    libraries
    logs
    models
    third_party
    views
assets

Name                       Ext.
[..]
[errors]
admin_agentmgmt_view       php
admin_bookingmgmt_view     php
admin_bookingstatus_vi...  php
agent_addbooking_view      php
agent_bookingstatus_vi...  php
index                      html
login                      php
```

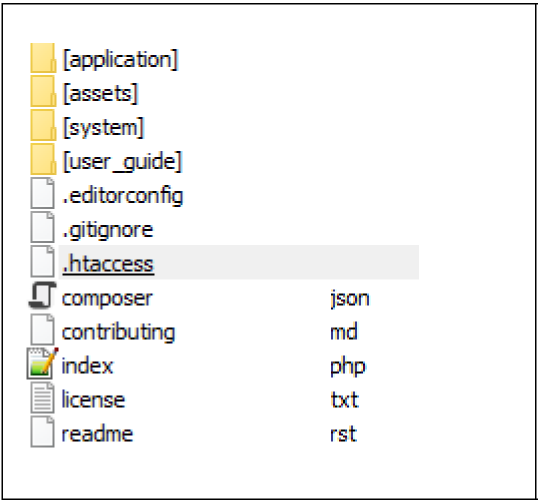


```
application
    bootstrap_datatable
    cache
    config
    controllers
    core
    helpers
    hooks
    language
    libraries
    logs
    models
    third_party
    views
assets

Name     Ext.
[..]
admin    php
agent    php
index    html
main     php
```

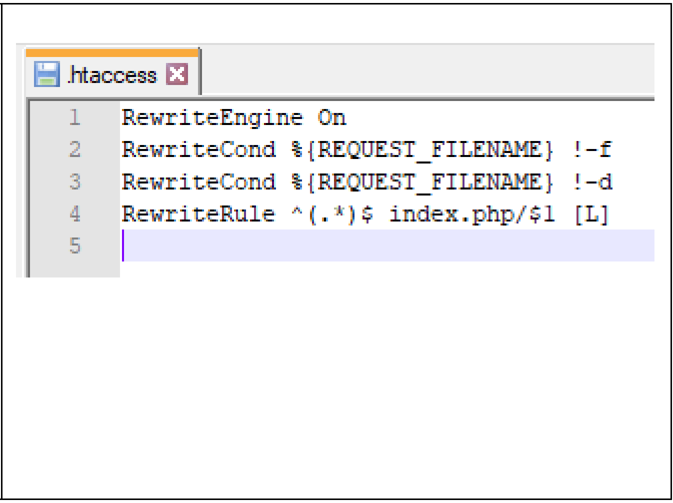
Table 2: MVC Framework Concept

Removing *index.php*

To enable easy access to the Uniform Resource Locator (URL) on every view, the developer has removed the "index.php" from the URL from http://localhost/ddacCRZ/index.php/main/login to http://localhost/ddacCRZ/main/login. Since the developer was running Apache as a local host server, removing index.php can be achieved without great effort. However, it is important to ensure the server is set up using 404 requests behind the scenes (CraftCMS, 2018). It is considered a basic rule to hide index.php from the URLs by enabling mod_rewrite in the .htaccess file using the codes shown below:

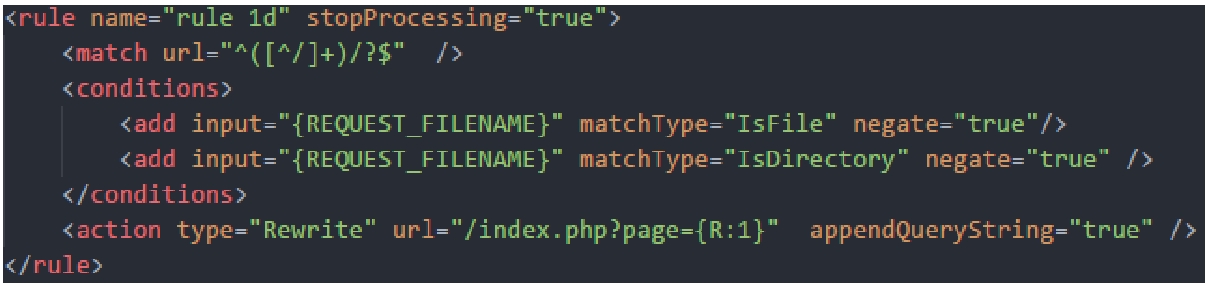


[application]	
[assets]	
[system]	
[user_guide]	
.editorconfig	
.gitignore	
.htaccess	
composer	json
contributing	md
index	php
license	txt
readme	rst



```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]
```

The above method will only work in Apache server. As such, the .htaccess rule will need to be written again in the web.config when the developer deploys the web application to the Azure portal as shown in Figure 19.



```
<rule name="rule 1d" stopProcessing="true">
    <match url="^([^/]+)/?$"  />
    <conditions>
        <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true"/>
        <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
    </conditions>
    <action type="Rewrite" url="/index.php?page={R:1}"  appendQueryString="true" />
</rule>
```

Figure 19: Remove index.php in every URL

Explanation of CodeIgniter MVC on Maersk Line

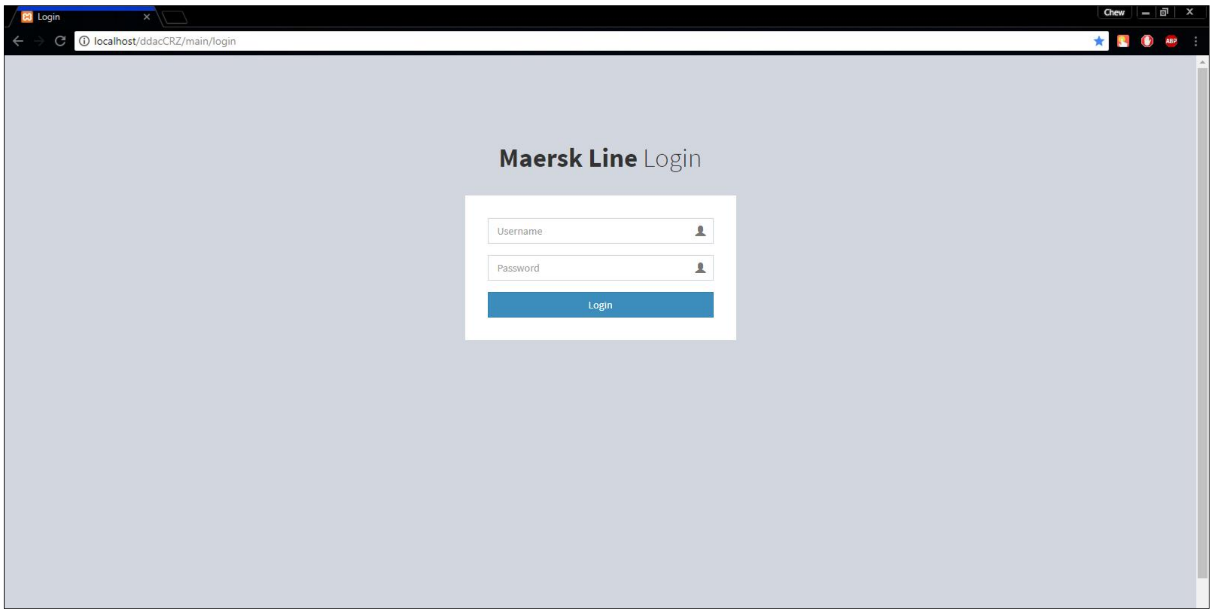
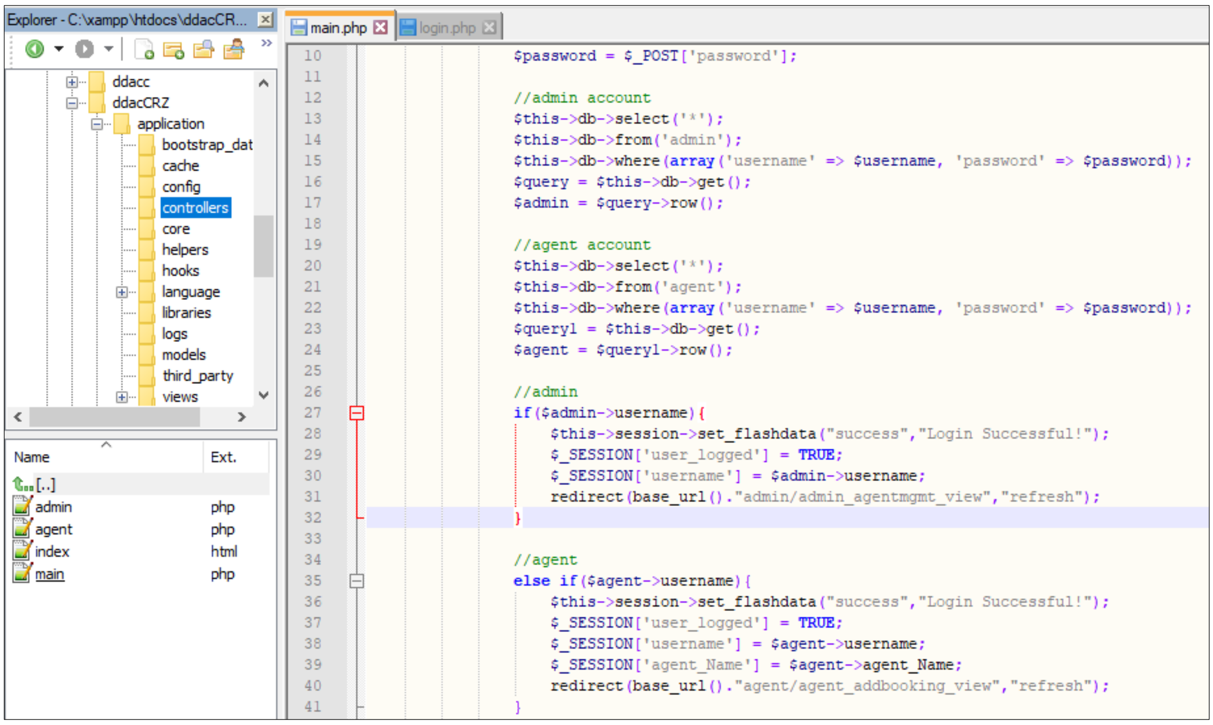


Figure 20: Maersk Line Login Page

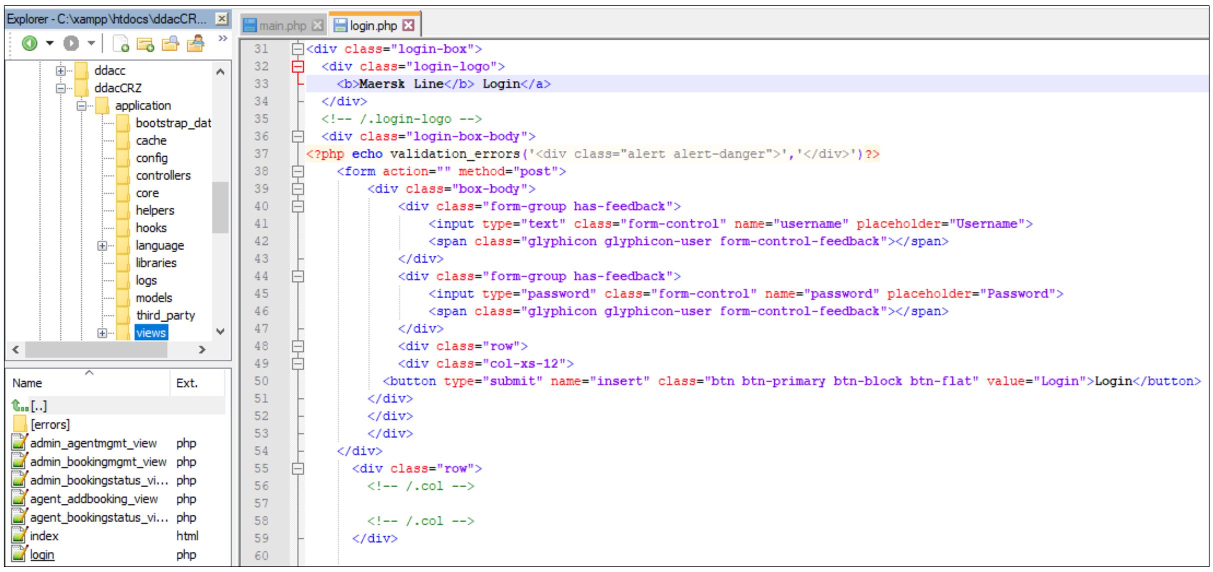
Figure 20 shows the login URL (http://localhost/ddacCRZ/main/login) for Maersk Line. This login page is load from the *main controller* with login() function as shown in Figure 21 to prompt the login page with *login.php* as shown in Figure 22.



Name	Ext.
[..]	
admin	php
agent	php
index	html
main	php

```php
                $password = $_POST['password'];

                //admin account
                $this->db->select('*');
                $this->db->from('admin');
                $this->db->where(array('username' => $username, 'password' => $password));
                $query = $this->db->get();
                $admin = $query->row();

                //agent account
                $this->db->select('*');
                $this->db->from('agent');
                $this->db->where(array('username' => $username, 'password' => $password));
                $query1 = $this->db->get();
                $agent = $query1->row();

                //admin
                if($admin->username){
                    $this->session->set_flashdata("success","Login Successful!");
                    $_SESSION['user_logged'] = TRUE;
                    $_SESSION['username'] = $admin->username;
                    redirect(base_url()."admin/admin_agentmgmt_view","refresh");
                }

                //agent
                else if($agent->username){
                    $this->session->set_flashdata("success","Login Successful!");
                    $_SESSION['user_logged'] = TRUE;
                    $_SESSION['username'] = $agent->username;
                    $_SESSION['agent_Name'] = $agent->agent_Name;
                    redirect(base_url()."agent/agent_addbooking_view","refresh");
                }
```

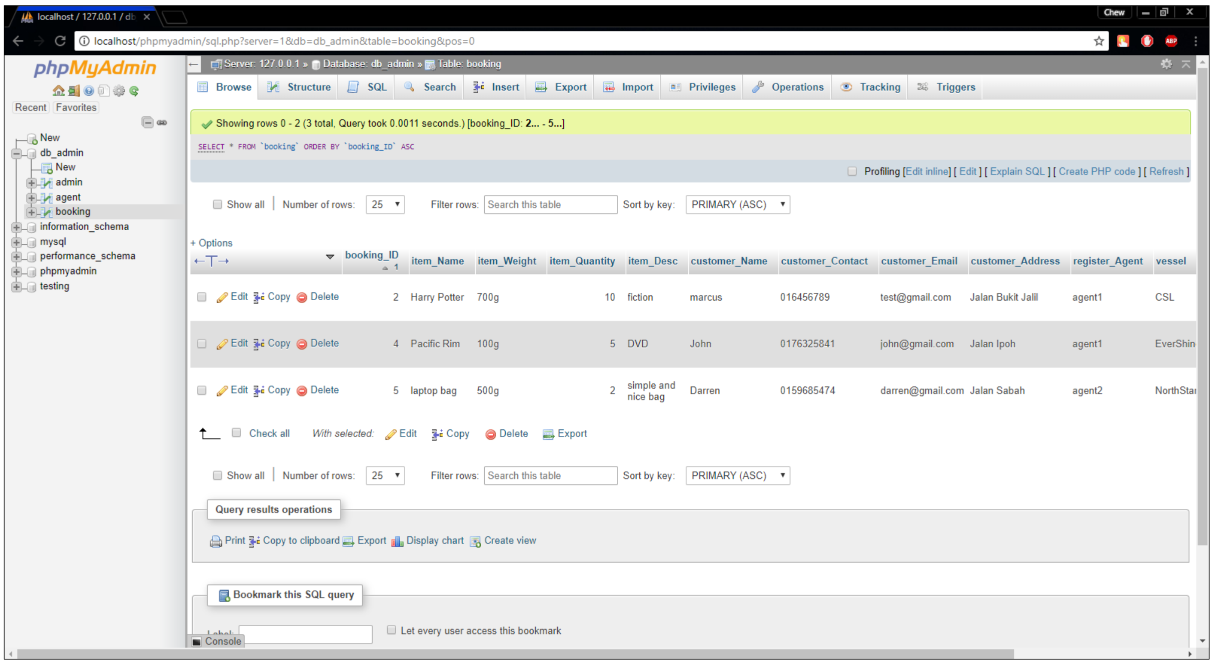
Figure 21: Main Controller



```
<div class="login-box">
  <div class="login-logo">
    <b>Maersk Line</b> Login</a>
  </div>
  <!-- /.login-logo -->
  <div class="login-box-body">
<?php echo validation_errors('<div class="alert alert-danger">','</div>')?>
    <form action="" method="post">
        <div class="box-body">
            <div class="form-group has-feedback">
                <input type="text" class="form-control" name="username" placeholder="Username">
                <span class="glyphicon glyphicon-user form-control-feedback"></span>
            </div>
            <div class="form-group has-feedback">
                <input type="password" class="form-control" name="password" placeholder="Password">
                <span class="glyphicon glyphicon-user form-control-feedback"></span>
            </div>
            <div class="row">
            <div class="col-xs-12">
           <button type="submit" name="insert" class="btn btn-primary btn-block btn-flat" value="Login">Login</button>
        </div>
        </div>
        </div>
    </div>
      <div class="row">
        <!-- /.col -->

        <!-- /.col -->
      </div>

```

Figure 22: Login View

<u>Maersk Line using phpMyAdmin as database</u>



localhost/phpmyadmin/sql.php?server=1&db=db_admin&table=booking&pos=0

Server: 127.0.0.1 » Database: db_admin » Table: booking

Showing rows 0 - 2 (3 total, Query took 0.0011 seconds.) [booking_ID: 2... - 5...]

```
SELECT * FROM `booking` ORDER BY `booking_ID` ASC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: PRIMARY (ASC)

	booking_ID	item_Name	item_Weight	item_Quantity	item_Desc	customer_Name	customer_Contact	customer_Email	customer_Address	register_Agent	vessel
Edit Copy Delete	2	Harry Potter	700g	10	fiction	marcus	016456789	test@gmail.com	Jalan Bukit Jalil	agent1	CSL
Edit Copy Delete	4	Pacific Rim	100g	5	DVD	John	0176325841	john@gmail.com	Jalan Ipoh	agent1	EverShin
Edit Copy Delete	5	laptop bag	500g	2	simple and nice bag	Darren	0159685474	darren@gmail.com	Jalan Sabah	agent2	NorthSta

Check all With selected: Edit Copy Delete Export

Figure 23: phpMyAdmin Database

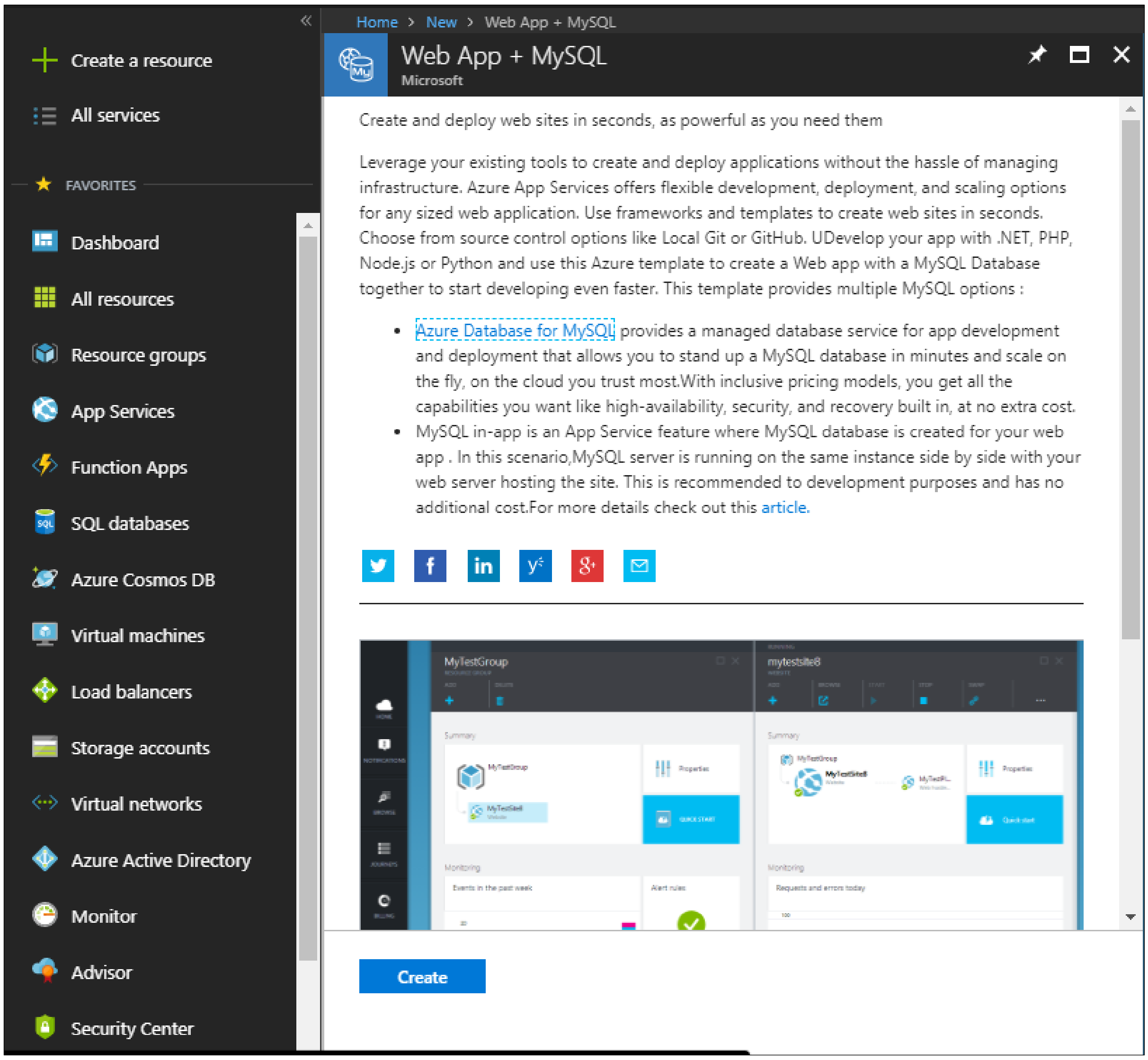
Before the developer deploys the system to Azure, the phpMyAdmin is used as a local database for setting up the storage of data with the use of MySQL database language. MySQL Workbench 6.3 CE was then used by the developer to deploy the database to Azure and the explanation is shown in Section 4.2.

Source Code	:	https://github.com/fernandochewrz/ddacCRZ
Maersk Line CMS URL	:	https://ddactp038150.azurewebsites.net/main/login
Presentation Video	:	https://web.microsoftstream.com/video/b1828482-d770-4459-bba4-38fadf048a05

4.2 Azure Publishing

This section explains several steps taken to deploy the Maersk Line CMS on Microsoft Azure.

<u>Step 1: Choose a web service for Maersk Line on Microsoft Azure</u>



Create and deploy web sites in seconds, as powerful as you need them

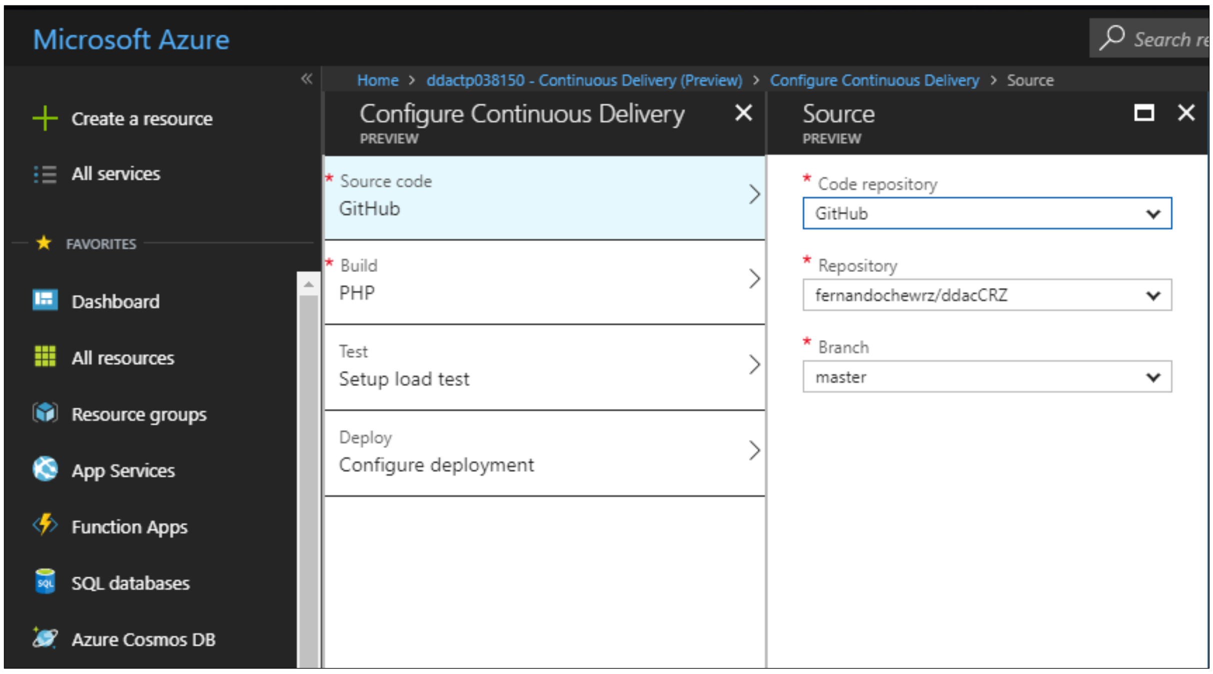
Leverage your existing tools to create and deploy applications without the hassle of managing infrastructure. Azure App Services offers flexible development, deployment, and scaling options for any sized web application. Use frameworks and templates to create web sites in seconds. Choose from source control options like Local Git or GitHub. UDevelop your app with .NET, PHP, Node.js or Python and use this Azure template to create a Web app with a MySQL Database together to start developing even faster. This template provides multiple MySQL options :

- Azure Database for MySQL provides a managed database service for app development and deployment that allows you to stand up a MySQL database in minutes and scale on the fly, on the cloud you trust most.With inclusive pricing models, you get all the capabilities you want like high-availability, security, and recovery built in, at no extra cost.
- MySQL in-app is an App Service feature where MySQL database is created for your web app . In this scenario,MySQL server is running on the same instance side by side with your web server hosting the site. This is recommended to development purposes and has no additional cost.For more details check out this article.

Figure 24: Selection of Web Services

Firstly, the developer chosen “Web App + MySQL” as the web service to host Maersk Line CMS as shown in Figure 24. This service has become the first choice since the developer has built the Maersk Line system using PHP and MySQL as the web application platform. The integration between the system and database would be easier throughout the deployment process.

Step 2: Configure the Maersk Line System with Continuous Delivery

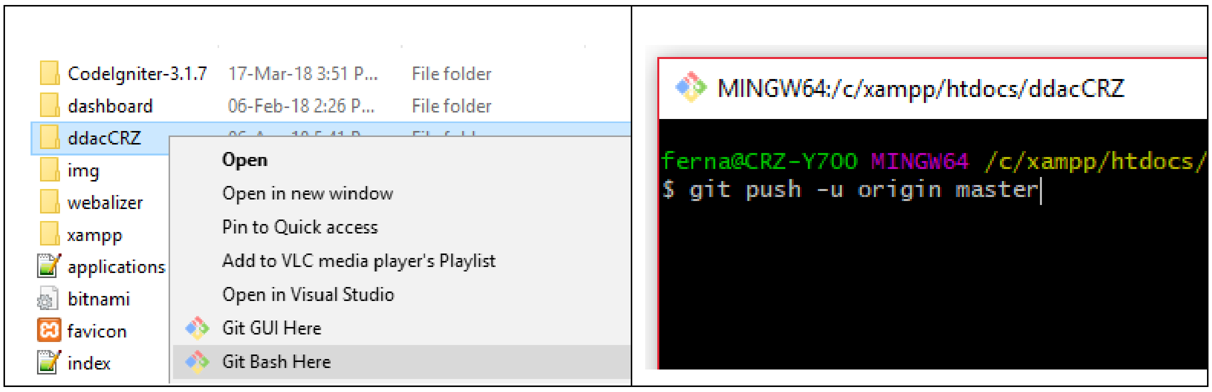


Source code: GitHub
Build: PHP
Test: Setup load test
Deploy: Configure deployment

Code repository: GitHub
Repository: fernandochewrz/ddacCRZ
Branch: master

Figure 25: Project Continuous Delivery from GitHub

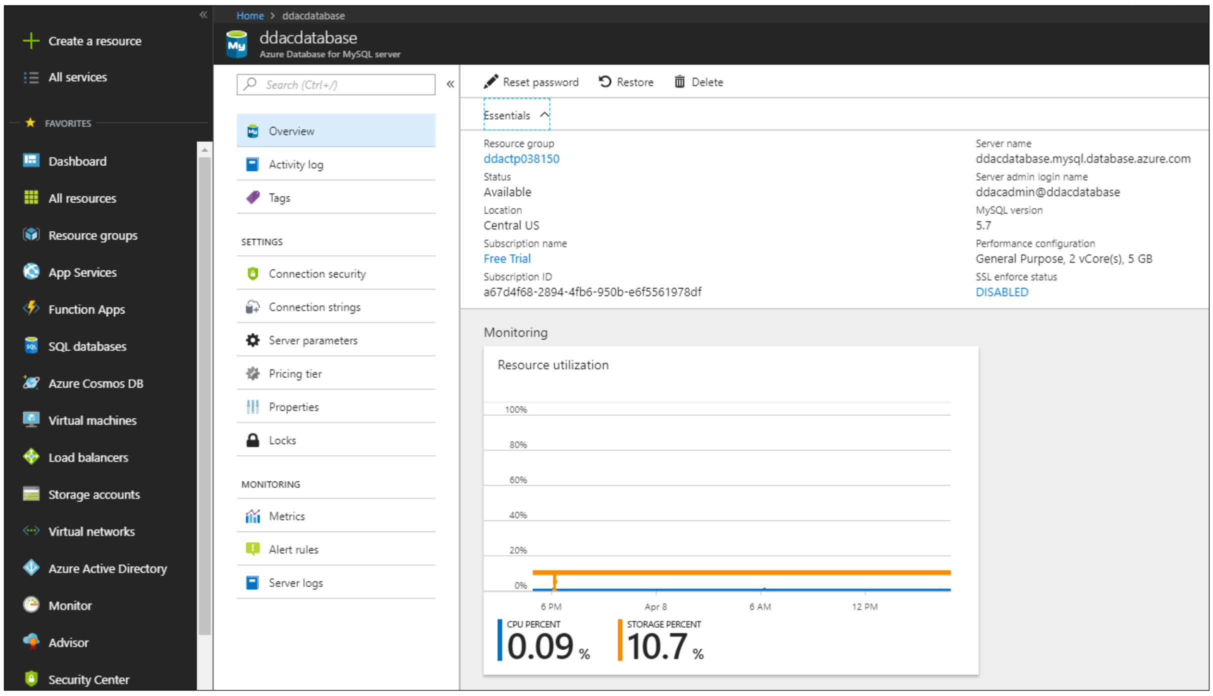
Figure 25 shows the Maersk Line CMS was configured using Continuous Delivery through GitHub for continuous deployment purposes. The developer considered the system should be easy for **maintainable**, therefore this selection is carried out to enable the project to be synced from GitHub whenever the system code is being committed through Git Bash from the developer's computer as shown in Figure 26.


CodeIgniter-3.1.7 17-Mar-18 3:51 P... File folder
dashboard 06-Feb-18 2:26 P... File folder
ddacCRZ
img
webalizer
xampp
applications
bitnami
favicon
index
Open
Open in new window
Pin to Quick access
Add to VLC media player's Playlist
Open in Visual Studio
Git GUI Here
Git Bash Here
MINGW64:/c/xampp/htdocs/ddacCRZ

```
ferna@CRZ-Y700 MINGW64 /c/xampp/htdocs/
$ git push -u origin master
```

Figure 26: Steps for committing source code to GitHub

Step 3: Setup Azure Database



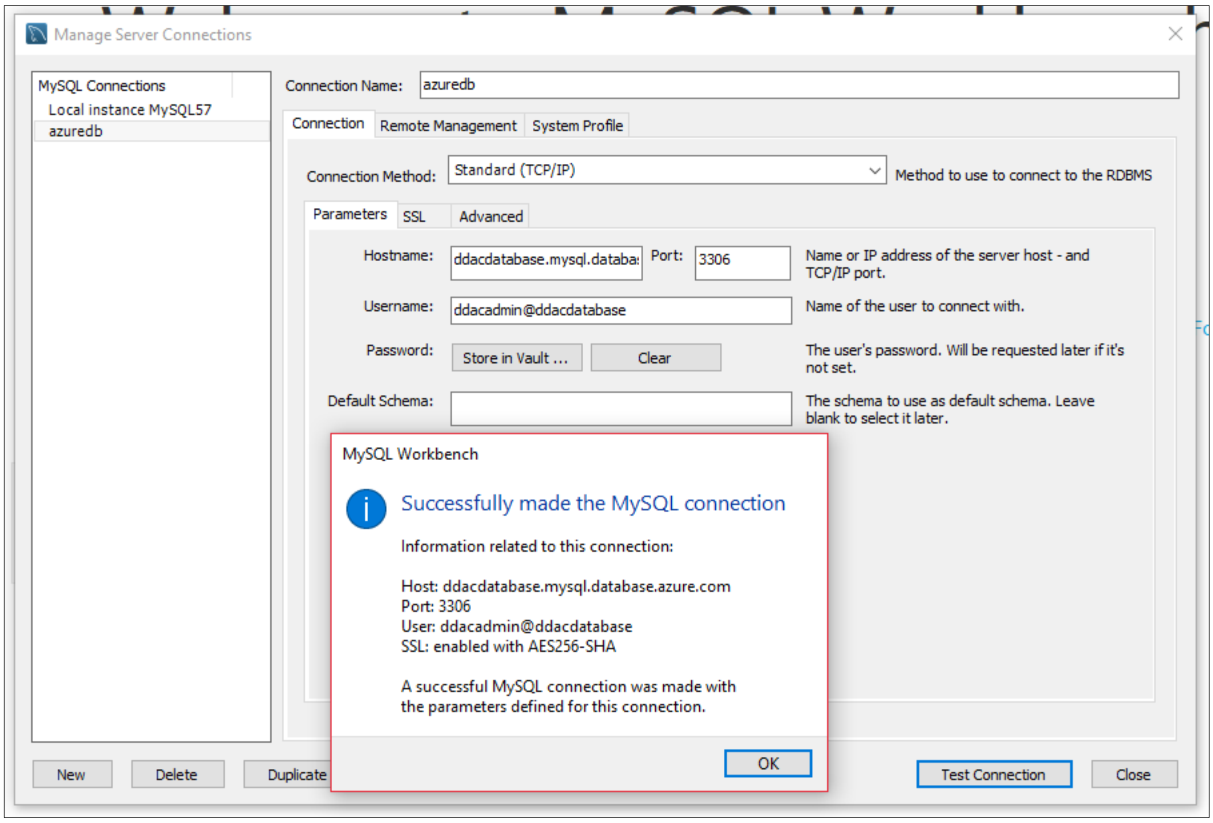
Resource group: ddactp038150
Status: Available
Location: Central US
Subscription name: Free Trial
Subscription ID: a67d4f68-2894-4fb6-950b-e6f5561978df
Server name: ddacdatabase.mysql.database.azure.com
Server admin login name: ddacadmin@ddacdatabase
MySQL version: 5.7
Performance configuration: General Purpose, 2 vCore(s), 5 GB
SSL enforce status: DISABLED

CPU PERCENT 0.09 %
STORAGE PERCENT 10.7 %

A MySQL database is required to store the admin, agent and customer booking data in Maersk Line CMS. Furthermore, this database on Azure offers monitoring and scaling capability by connecting to the service plan that hosts the Maersk Line CMS where it requires the subscription plan and pricing tier. The name of the database can be customized, including the server name, server admin login name and login password.

Step 4: Connects Azure Database

MySQL Workbench 6.3 CE is used to deploy the local database to the web application on Microsoft Azure specifically for Azure MySQL database. It is accomplished by connecting the workbench and Azure database. Figure 27 shows the connection has successfully established between the local workbench and Azure database.



Connection Name: azuredb

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Hostname: ddacdatabase.mysql.databa: Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: ddacadmin@ddacdatabase Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

MySQL Workbench

Successfully made the MySQL connection

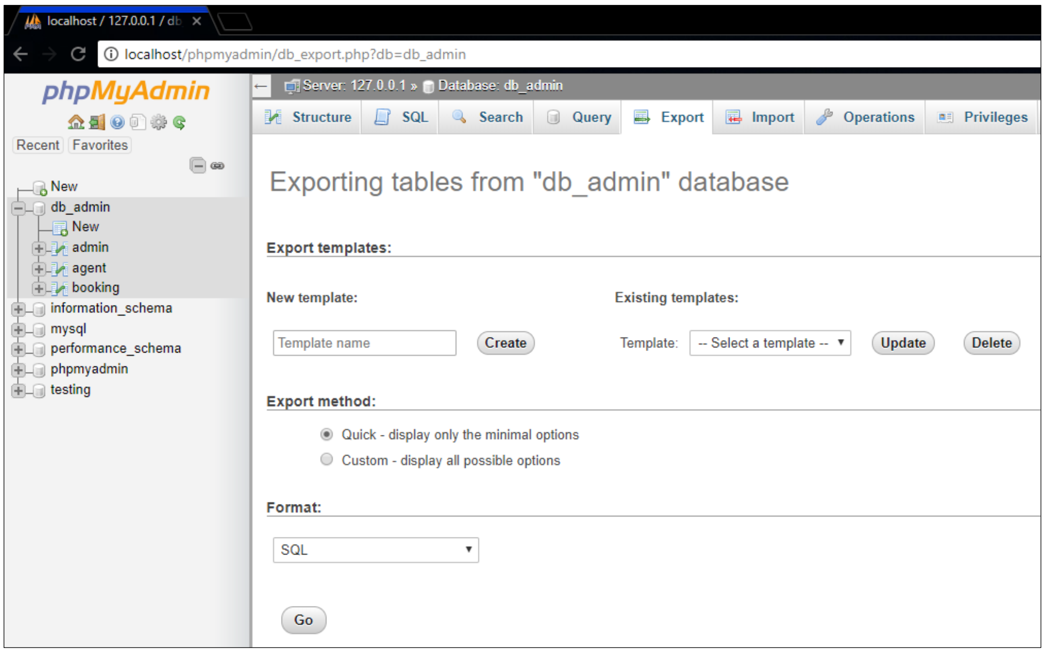
Information related to this connection:

Host: ddacdatabase.mysql.database.azure.com
Port: 3306
User: ddacadmin@ddacdatabase
SSL: enabled with AES256-SHA

A successful MySQL connection was made with the parameters defined for this connection.

Figure 27: Manage server connections

The process is followed by exporting the database from phpMyAdmin as shown in Figure 28 and import the database into MySQL Workbench as shown in Figure 29.


localhost/phpmyadmin/db_export.php?db=db_admin

Server: 127.0.0.1 » Database: db_admin

Exporting tables from "db_admin" database

Export templates:

New template: Template name Create

Existing templates: Template: -- Select a template -- Update Delete

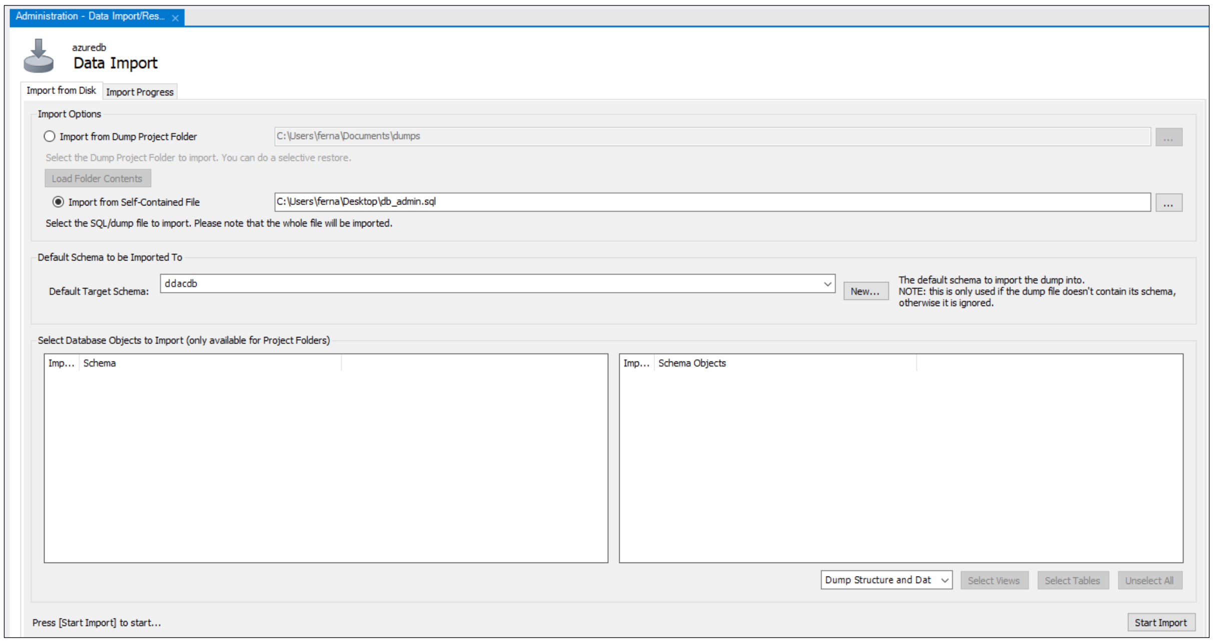
Export method:

(●) Quick - display only the minimal options
( ) Custom - display all possible options

Format: SQL

Go

Figure 28: Export db_admin database from phpMyAdmin


azuredb
Data Import

Import Options

( ) Import from Dump Project Folder: C:\Users\ferna\Documents\dumps
Select the Dump Project Folder to import. You can do a selective restore.

(●) Import from Self-Contained File: C:\Users\ferna\Desktop\db_admin.sql
Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Schema to be Imported To

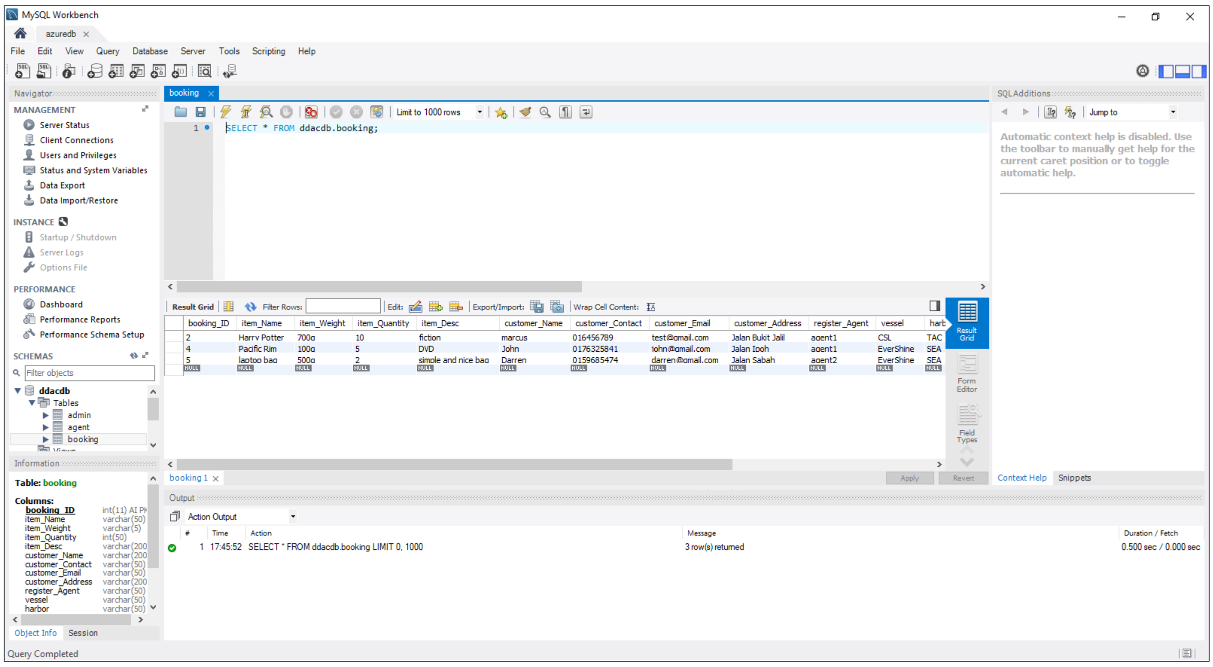
Default Target Schema: ddacdb
The default schema to import the dump into.
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

Select Database Objects to Import (only available for Project Folders)

Dump Structure and Dat

Press [Start Import] to start...

Figure 29: Import db_admin database into MySQL Workbench



```
SELECT * FROM ddacdb.booking;
```

booking_ID	item_Name	item_Weight	item_Quantity	item_Desc	customer_Name	customer_Contact	customer_Email	customer_Address	register_Agent	vessel	harb
2	Harry Potter	700g	10	fiction	marcus	016456789	test@gmail.com	Jalan Bukit Jalil	agent1	CSL	TAC
4	Pacific Rim	100g	5	DVD	John	0176325841	john@gmail.com	Jalan Ipoh	agent1	EverShine	SEA
5	laptop bag	500g	2	simple and nice bag	Darren	0159685474	darren@gmail.com	Jalan Sabah	agent2	EverShine	SEA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: booking

Columns:

booking_ID	int(11) AI PK
item_Name	varchar(50)
item_Weight	varchar(5)
item_Quantity	int(50)
item_Desc	varchar(200
customer_Name	varchar(200
customer_Contact	varchar(50)
customer_Email	varchar(50)
customer_Address	varchar(200
register_Agent	varchar(50)
vessel	varchar(50)
harbor	varchar(50)

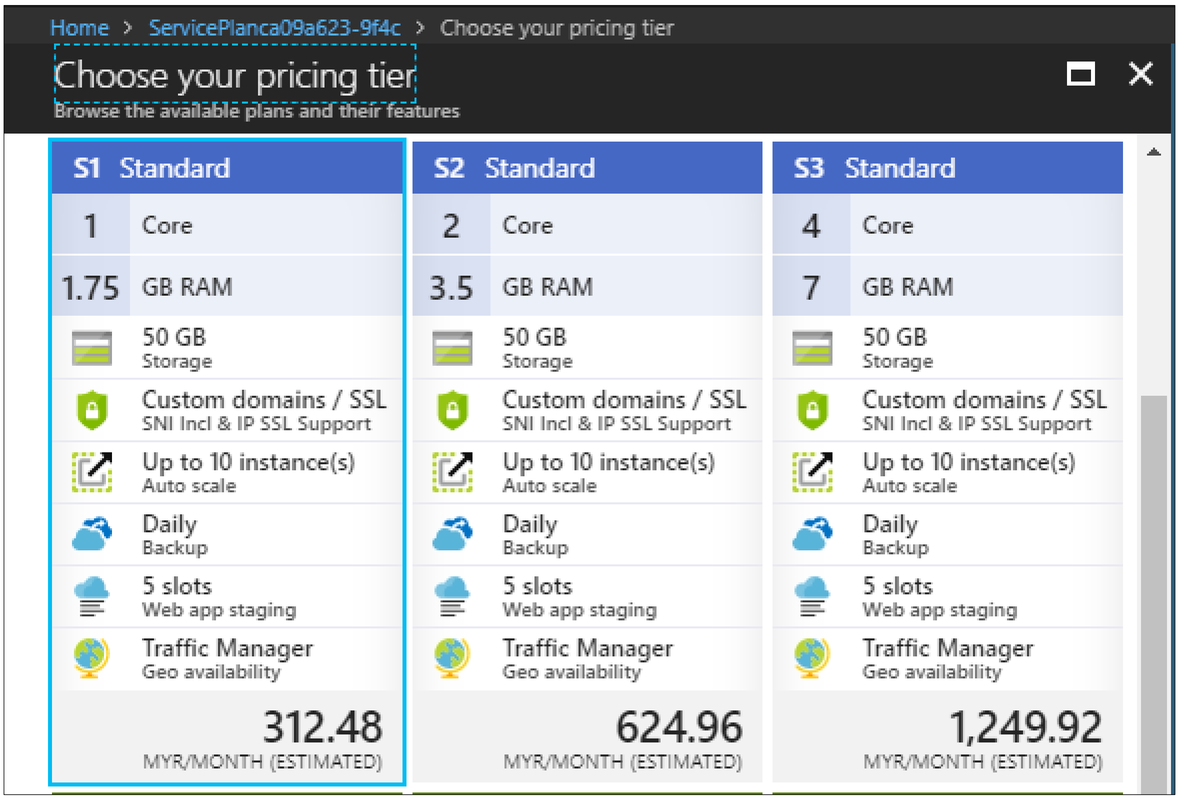
```
#  Time      Action                                         Message             Duration / Fetch
1  17:45:52  SELECT * FROM ddacdb.booking LIMIT 0, 1000     3 row(s) returned   0.500 sec / 0.000 sec
```

Query Completed

Figure 30: Database imported to MySQL Workbench

Figure 30 shows the db_admin database is successfully imported to Azure MySQL database, together with the data stored previous on phpMyAdmin also being updated on the Azure database.

4.3 Application Scaling


Home > ServicePlanca09a623-9f4c > Choose your pricing tier

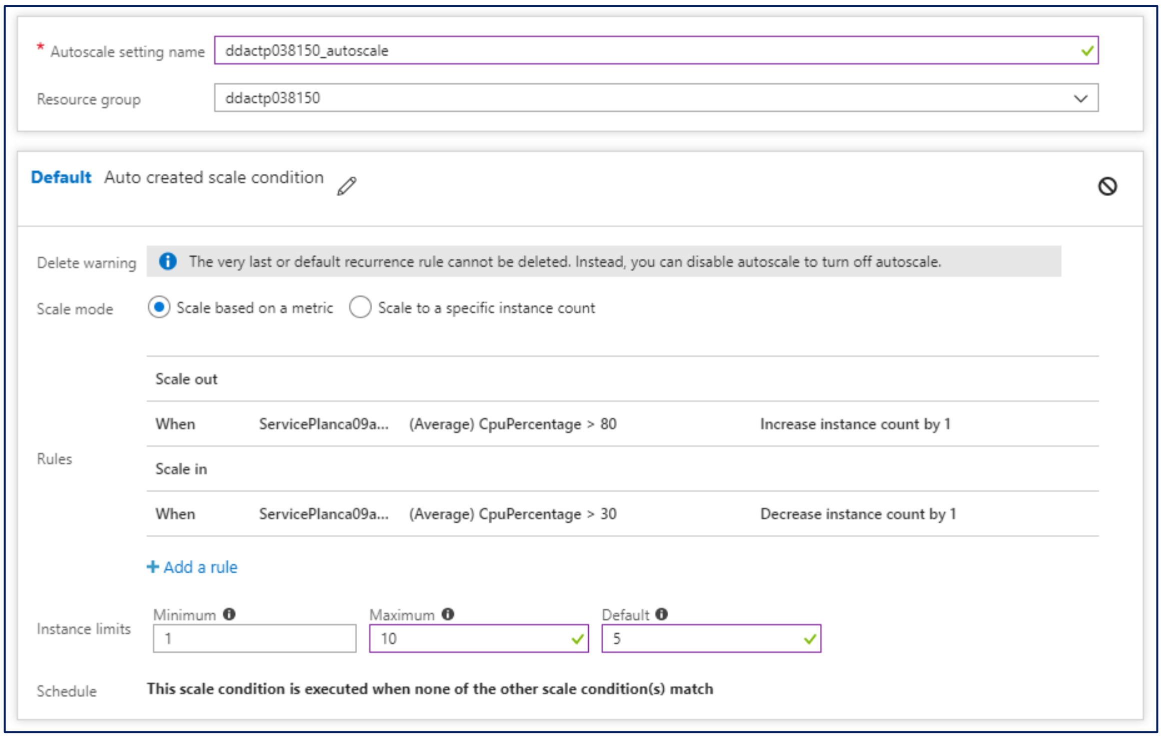
Choose your pricing tier

Browse the available plans and their features

S1 Standard	S2 Standard	S3 Standard
1 Core	2 Core	4 Core
1.75 GB RAM	3.5 GB RAM	7 GB RAM
50 GB Storage	50 GB Storage	50 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
Up to 10 instance(s) Auto scale	Up to 10 instance(s) Auto scale	Up to 10 instance(s) Auto scale
Daily Backup	Daily Backup	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
312.48 MYR/MONTH (ESTIMATED)	624.96 MYR/MONTH (ESTIMATED)	1,249.92 MYR/MONTH (ESTIMATED)

Figure 31: Maersk Line CMS Pricing Tier

The S1 Standard is being chosen as the web application plan for Maersk Line CMS in the region of Central US due to budget limit. However, it should be upgraded to at least S3 Standard plan where four cores are needed for concurrency multiple of users. The upgrade plan will be executed when the company hires more agents to utilize the system where the booking tasks will be increased as well. 50 GB of storage should be sufficient for the company since the web application stores only characters datatypes without image and videos. Custom domain is needed for the company to put in their desired URL. In addition, the system should have at least 10 instances under auto scaled function to enable balance load without delays when there are more agents who access the web application. Daily backup is essential to enable the data recovery once the data is lost.



* Autoscale setting name: ddactp038150_autoscale

Resource group: ddactp038150

Default Auto created scale condition

Delete warning: The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode: [✓ Scale based on a metric] Scale to a specific instance count

Rules:

Scale out			
When	ServicePlanca09a...	(Average) CpuPercentage > 80	Increase instance count by 1
Scale in			
When	ServicePlanca09a...	(Average) CpuPercentage > 30	Decrease instance count by 1

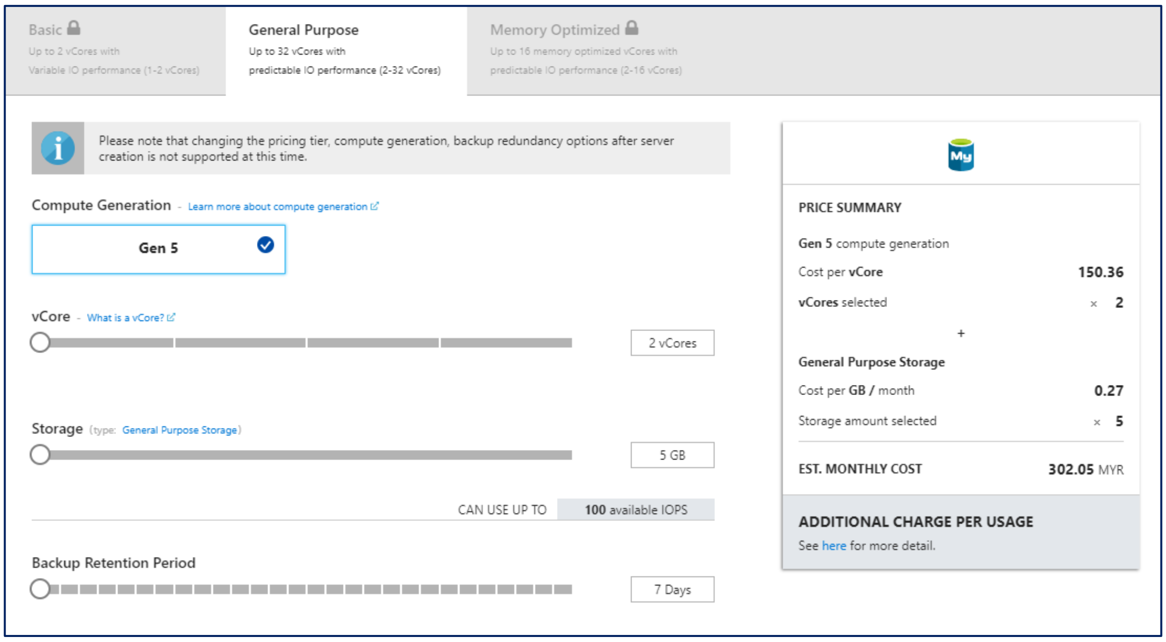
+ Add a rule

Instance limits: Minimum 1, Maximum 10, Default 5

Schedule: This scale condition is executed when none of the other scale condition(s) match

Figure 32: Maersk Line Web Application Scaling Plan

The scaling plan shown in Figure 32 shows that when the CPU usage is greater than 80%, one instance will be added to manage the heavier workload and when the CPU usage is lower than 30%, the instance will be decreased by one. There will be more options for scaling the service plan according to the business requirements such as upgrading the instances and storage capacity in the future as shown in Figure 33.



Basic — Up to 2 vCores with Variable IO performance (1-2 vCores)

General Purpose — Up to 32 vCores with predictable IO performance (2-32 vCores)

Memory Optimized — Up to 16 memory optimized vCores with predictable IO performance (2-16 vCores)

Please note that changing the pricing tier, compute generation, backup redundancy options after server creation is not supported at this time.

Compute Generation - Learn more about compute generation: Gen 5 ✓

vCore - What is a vCore?: 2 vCores

Storage (type: General Purpose Storage): 5 GB

CAN USE UP TO 100 available IOPS

Backup Retention Period: 7 Days

PRICE SUMMARY

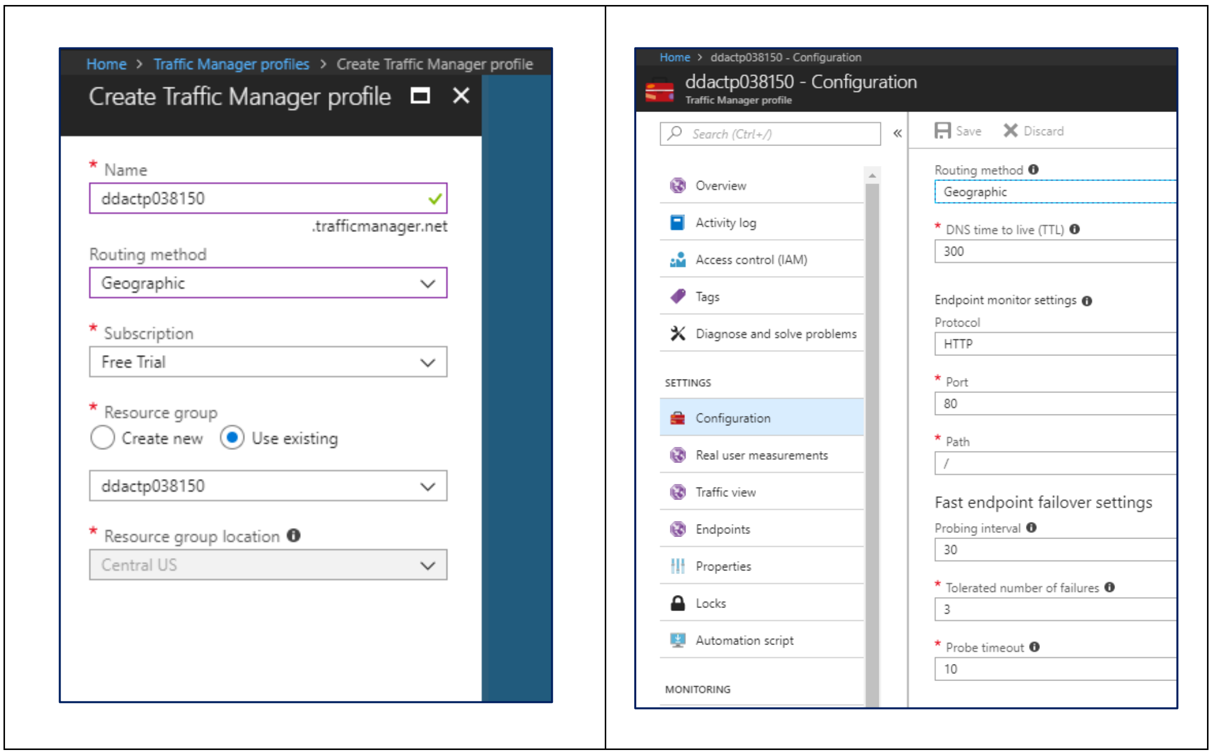
Gen 5 compute generation	
Cost per vCore	150.36
vCores selected	× 2
+	
General Purpose Storage	
Cost per GB / month	0.27
Storage amount selected	× 5
EST. MONTHLY COST	302.05 MYR

ADDITIONAL CHARGE PER USAGE

See here for more detail.

Figure 33: Azure MySQL Database pricing tier

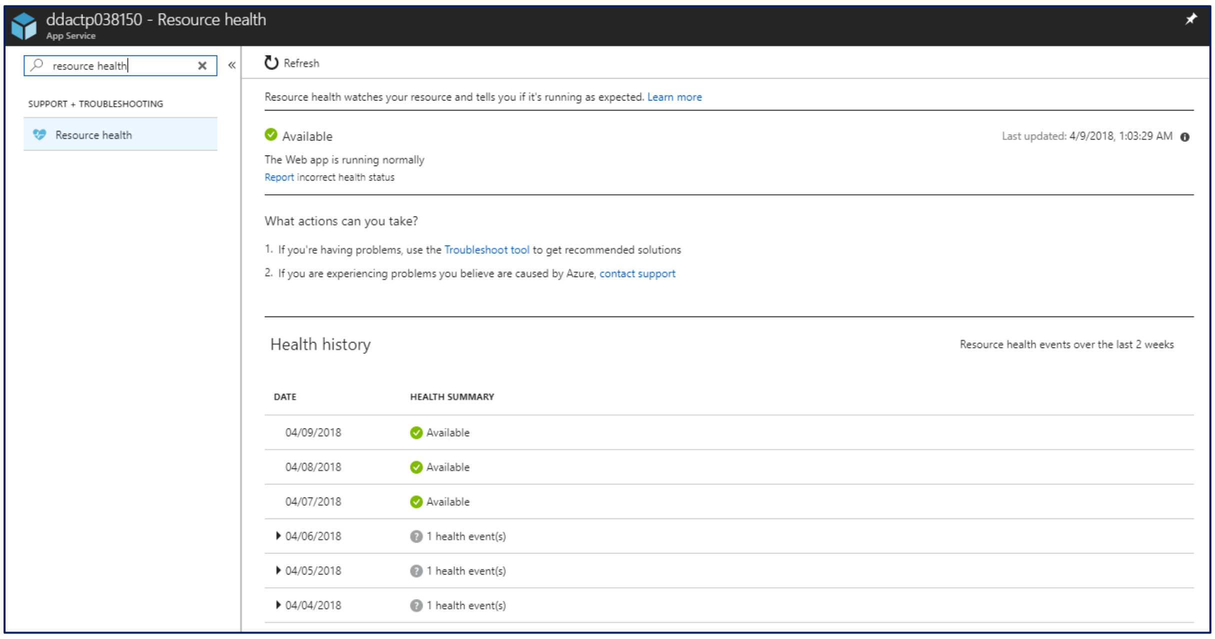
4.4 Reliability and Performance


Home > Traffic Manager profiles > Create Traffic Manager profile
Create Traffic Manager profile

* Name: ddactp038150 .trafficmanager.net
Routing method: Geographic
* Subscription: Free Trial
* Resource group: Create new [✓ Use existing]
ddactp038150
* Resource group location: Central US

Routing method: Geographic
* DNS time to live (TTL): 300
Endpoint monitor settings
Protocol: HTTP
* Port: 80
* Path: /
Fast endpoint failover settings
Probing interval: 30
* Tolerated number of failures: 3
* Probe timeout: 10

Figure 34: measuring the system health



Resource health watches your resource and tells you if it's running as expected. Learn more

Available

Last updated: 4/9/2018, 1:03:29 AM

The Web app is running normally

Report incorrect health status

What actions can you take?

1. If you're having problems, use the Troubleshoot tool to get recommended solutions
2. If you are experiencing problems you believe are caused by Azure, contact support

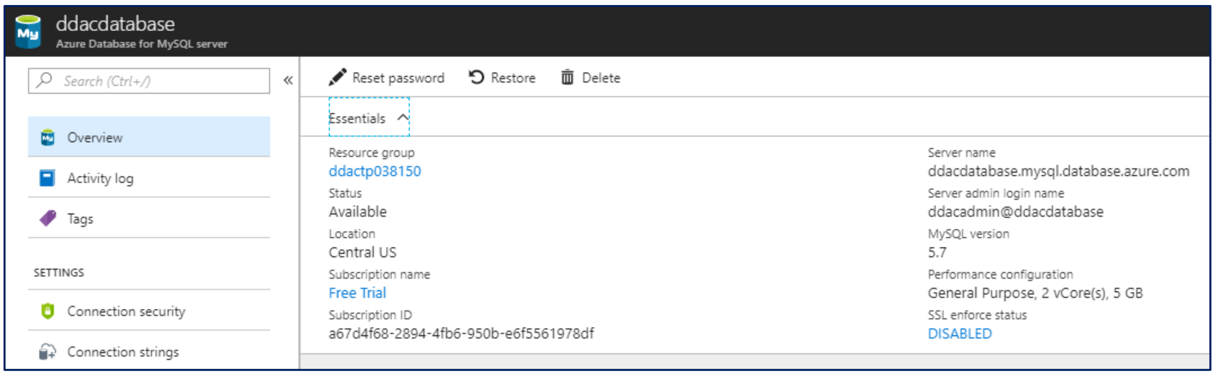
Health history

Resource health events over the last 2 weeks

DATE	HEALTH SUMMARY
04/09/2018	Available
04/08/2018	Available
04/07/2018	Available
04/06/2018	1 health event(s)
04/05/2018	1 health event(s)
04/04/2018	1 health event(s)

Figure 35: Monitoring system health

Figure 34 shows that the traffic manager is created to measure the reliability of Maersk Line CMS web application. It is measured through the health endpoint which the endpoint is responsible to check the instances of the web application. The traffic manager will redirect the traffic to another instance when the instance does not respond with a HTTP 200 status. A health summary is provided as well as shown in Figure 35.



Resource group: ddactp038150
Status: Available
Location: Central US
Subscription name: Free Trial
Subscription ID: a67d4f68-2894-4fb6-950b-e6f5561978df
Server name: ddacdatabase.mysql.database.azure.com
Server admin login name: ddacadmin@ddacdatabase
MySQL version: 5.7
Performance configuration: General Purpose, 2 vCore(s), 5 GB
SSL enforce status: DISABLED

Figure 36: MySQL server optimization

In Figure 36, the Azure MySQL server instance is being optimized internally and the instance is being kept in Central US. In addition, the database is capable of adding geo-replication in the future when it is needed for improve data integrity and enable data recovery from different location when the operational database is affected by natural disaster such as earthquake.

5.0 Test Plan & Testing Discussion

5.1 Unit Testing

1. Admin Login

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-AL-01	1. Enter valid username and password 2. Click Login button	System is logged in into admin page.	System is logged in into admin page.	Pass
TC-AL-02	1. Enter incorrect username invalid password 2. Click Login button	Display error message “Incorrect Username!” and “Incorrect Password!”.	Display error message “Incorrect Username!” and “Incorrect Password!”.	Pass
TC-AL-03	1. Enter correct username invalid password 2. Click Login button	Display error message “Incorrect Password!”.	Display error message “Incorrect Password!”.	Pass
TC-AL-04	1. Do not enter any value 2. Click Login button	Display error message “The Username field is required.” and “The Username field is required.”	Display error message “The Username field is required.” and “The Username field is required.”	Pass
TC-AL-05	1. Do not enter username 2. Enter password 3. Click Login button	Display error message “The Username field is required.”	Display error message “The Username field is required.”	Pass
TC-AL-06	1. Do not enter password 2. Enter username 3. Click Login button	Display error message “The Password field is required.”	Display error message “The Password field is required.”	Pass

2. Agent Login

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-AL-07	1. Enter valid username and password 2. Click Login button	System is logged in into agent page.	System is logged in into agent page.	Pass
TC-AL-08	1. Enter incorrect username invalid password 2. Click Login button	Display error message “Incorrect Username!” and “Incorrect Password!”.	Display error message “Incorrect Username!” and “Incorrect Password!”.	Pass
TC-AL-09	1. Enter correct username invalid password 2. Click Login button	Display error message “Incorrect Password!”.	Display error message “Incorrect Password!”.	Pass
TC-AL-10	1. Do not enter any value 2. Click Login button	Display error message “The Username field is required.” and “The Username field is required.”	Display error message “The Username field is required.” and “The Username field is required.”	Pass
TC-AL-11	1. Do not enter username 2. Enter password 3. Click Login button	Display error message “The Username field is required.”	Display error message “The Username field is required.”	Pass
TC-AL-12	1. Do not enter password 2. Enter username 3. Click Login button	Display error message “The Password field is required.”	Display error message “The Password field is required.”	Pass

3. View Agent Details

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-VAD-01	1. Admin login to the system	System display a list of agents.	System display a list of agents.	Pass

4. Register Agent

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-RA-01	1. Admin login to the system 2. Click “Add” 3. Enter agent username 4. Enter agent password 5. Enter agent name 6. Enter agent contact 7. Enter agent address 8. Click Insert button	System display “Successfully added Agent”. Agent list shows the added agent details.	System display “Successfully added Agent”. Agent list shows the added agent details.	Pass
TC-RA-02	1. Admin login to the system 2. Click “Add” 3. Do not agent username 4. Enter agent password 5. Enter agent name 6. Enter agent contact 7. Enter agent address	System display “Fail to add Agent”.	System display “Fail to add Agent”.	Pass

	8. Click Insert button			
TC-RA-03	1. Admin login to the system 2. Click “Add” 3. Enter agent username 4. Do not enter agent password 5. Enter agent name 6. Enter agent contact 7. Enter agent address 8. Click Insert button	System display “Fail to add Agent”.	System display “Fail to add Agent”.	Pass
TC-RA-04	1. Admin login to the system 2. Click “Add” 3. Enter agent username 4. Enter agent password 5. Do not enter agent name 6. Enter agent contact 7. Enter agent address 8. Click Insert button	System display “Fail to add Agent”.	System display “Fail to add Agent”.	Pass
TC-RA-05	1. Admin login to the system 2. Click “Add” 3. Enter agent username 4. Enter agent password 5. Enter agent name 6. Do not enter agent contact 7. Enter agent address 8. Click Insert button	System display “Fail to add Agent”.	System display “Fail to add Agent”.	Pass

TC-RA-06	1. Admin login to the system 2. Click “Add” 3. Enter agent username 4. Enter agent password 5. Enter agent name 6. Enter agent contact 7. Do not enter agent address 8. Click Insert button	System display “Fail to add Agent”.	System display “Fail to add Agent”.	Pass

5. Delete Agent Details

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-DAD-01	1. Admin login to the system 2. Click Delete	System prompt confirmation dialog. System delete the selected record if user clicks OK.	System prompt confirmation dialog. System delete the selected record if user clicks OK.	Pass
TC-DAD-02	1. Admin login to the system 2. Click Cancel	System prompt confirmation dialog. System exit the dialog do not perform deletion if user clicks Cancel.	System prompt confirmation dialog. System exit the dialog do not perform deletion if user clicks Cancel.	Pass

6. View Booking List

Test Case ID	Description	Expected Result	Actual Result	Status
TC-VBL-01	1. Admin login to the system 2. Click Booking Management	System displays booking list.	System displays booking list.	Pass

7. Update Booking Status

Test Case ID	Description	Expected Result	Actual Result	Status
TC-UBS-01	1. Admin login to the system 2. Click Booking Status 3. Select Shipped from drop down menu 4. Click Update button	System displays booking list and the booking status is being changed from Paid to Shipped.	System displays booking list and the booking status is being changed from Paid to Shipped.	Pass

8. Delete Booking Details

Test Case ID	Description	Expected Result	Actual Result	Status
TC-DBD-01	1. Admin login to the system 2. Click Booking Management 3. Click Delete on the row of ideal booking ID 4. Click OK	System prompt confirmation dialog. System delete the selected record if user clicks OK.	System prompt confirmation dialog. System delete the selected record if user clicks OK.	Pass
TC-DBD-02	1. Admin login to the system 2. Click Booking Management 3. Click Delete on the row of ideal booking ID 4. Click Cancel	System prompt confirmation dialog. System exit the dialog do not perform deletion if user clicks Cancel.	System prompt confirmation dialog. System exit the dialog do not perform deletion if user clicks Cancel.	Pass

9. View Booking Details

Test Case ID	Description	Expected Result	Actual Result	Status
TC-VBD-01	1. Agent login to the system	System displays booking list.	System displays booking list.	Pass

10. Add Item Details

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-AID-01	1. Agent login to the system 2. Click Add button 3. Enter item name 4. Enter item weight 5. Enter item quantity 6. Enter item description 7. Click Insert button	System prompts “Insert Successful” message.	System prompts “Insert Successful” message.	Pass
TC-AID-02	1. Agent login to the system 2. Click Add button 3. Do not enter item name 4. Enter item weight 5. Enter item quantity 6. Enter item description 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-AID-03	1. Agent login to the system 2. Click Add button 3. Enter item name 4. Do not enter item weight 5. Enter item quantity 6. Enter item description 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-AID-04	1. Agent login to the system	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

	2. Click Add button 3. Enter item name 4. Enter item weight 5. Do not enter item quantity 6. Enter item description 7. Click Insert button			
TC-AID-05	1. Agent login to the system 2. Click Add button 3. Enter item name 4. Enter item weight 5. Enter item quantity 6. Do not enter item description 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

11. Add Customer Details

Test Case ID	Description	Expected Result	Actual Result	Status
TC-ACD-01	1. Agent login to the system 2. Click Add button 3. Enter customer name 4. Enter customer contact 5. Enter customer email 6. Enter customer address 7. Click Insert button	System prompts “Insert Successful” message.	System prompts “Insert Successful” message.	Pass
TC-ACD-02	1. Agent login to the system 2. Click Add button 3. Do not enter customer name 4. Enter customer contact 5. Enter customer email 6. Enter customer address 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-ACD-03	1. Agent login to the system 2. Click Add button 3. Enter customer name 4. Do not enter customer contact 5. Enter customer email 6. Enter customer address 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-ACD-04	1. Agent login to the system	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

	2. Click Add button 3. Enter customer name 4. Enter customer contact 5. Do not enter customer email 6. Enter customer address 7. Click Insert button			
TC-ACD-05	1. Agent login to the system 2. Click Add button 3. Enter customer name 4. Enter customer contact 5. Enter customer email 6. Do not enter customer address 7. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

12. Add Vessel Details

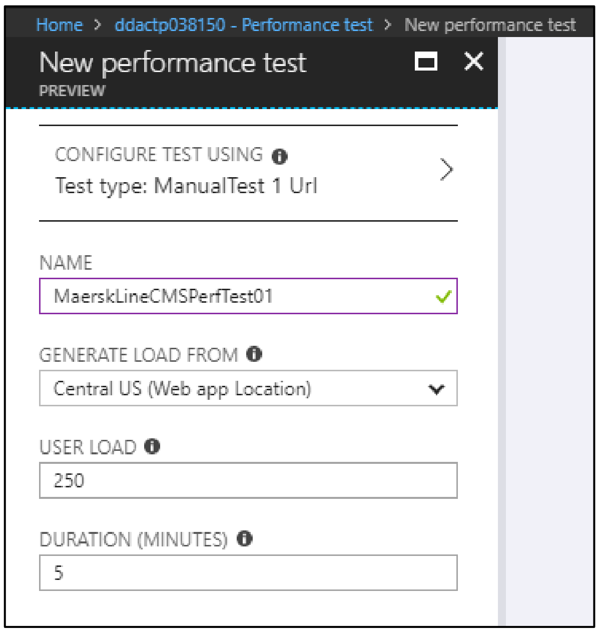
Test Case ID	Description	Expected Result	Actual Result	Status
TC-AVD-01	1. Agent login to the system 2. Click Add button 3. Select Vessel 4. Select Harbor 5. Select Terminal 6. Select Schedule 7. Select Status 8. Click Insert button	System prompts “Insert Successful” message.	System prompts “Insert Successful” message.	Pass
TC-AVD-02	1. Agent login to the system 2. Click Add button 3. Do not select Vessel 4. Select Harbor 5. Select Terminal 6. Select Schedule 7. Select Status 8. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-AVD-03	1. Agent login to the system 2. Click Add button 3. Select vessel 4. Do not select Harbor 5. Select Terminal 6. Select Schedule	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

	7. Select Status 8. Click Insert button			
TC-AVD-04	1. Agent login to the system 2. Click Add button 3. Select vessel 4. Select Harbor 5. Do not select Terminal 6. Select Schedule 7. Select Status 8. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-AVD-05	8. Agent login to the system 9. Click Add button 10. Select vessel 11. Select Harbor 12. Select Terminal 13. Do not select Schedule 14. Select Status 15. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass
TC-AVD-06	1. Agent login to the system 2. Click Add button 3. Select vessel 4. Select Harbor 5. Select Terminal 6. Select Schedule 7. Do not select Status 8. Click Insert button	System prompts “Insert Failed” message.	System prompts “Insert Failed” message.	Pass

10. View Booking Status

Test Case ID	**Description**	**Expected Result**	**Actual Result**	**Status**
TC-VBS-01	1. Agent login to the system 2. Click Booking Status	System displays booking list with the booking status.	System displays booking list with the booking status.	Pass

5.2 Performance Testing


Home > ddactp038150 - Performance test > New performance test

New performance test
PREVIEW

CONFIGURE TEST USING
Test type: ManualTest 1 Url

NAME
MaerskLineCMSPerfTest01

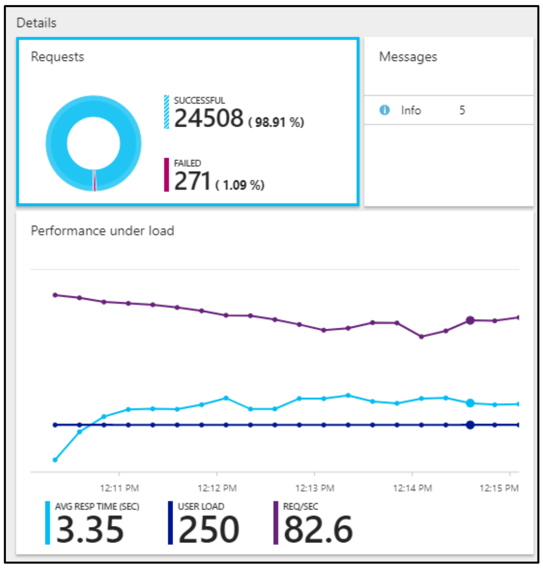
GENERATE LOAD FROM
Central US (Web app Location)

USER LOAD
250

DURATION (MINUTES)
5

Figure 37: Performance Test Setup for Maersk Line CMS

The performance of Maersk Line CMS can be accomplished by conducting performance test provided by Azure. As shown in Figure 37, the test is planned to carry out with 250 user loads in 5 minutes and the final results will be collecting the response time and fail requests as shown in Figure 38. In addition, the test will be repeated for 450 user loads, 650 user loads, and 850 user loads with 5 minutes for each test. The response time is being recorded for further analysis below. The test attributes including the status message, performance under load and CPU performance with memory usage.


Details

Requests
SUCCESSFUL 24508 (98.91 %)
FAILED 271 (1.09 %)

Messages
Info 5

Performance under load

AVG RESP TIME (SEC) 3.35
USER LOAD 250
REQ/SEC 82.6

Figure 38: Performance Test Example

Concurrent User / App Service Plan	250	500	750
S1	3.35s 271 failed	6.15s 375 failed	8.73s 449 failed
S2	3.78s 224 failed	**5.78s** 287 failed	7.21s 382 failed
S3	3.22s 133 failed	5.81s 158 failed	7.67s 187 failed

Table 3: Performance Test Results

5.3.1 Analysis

According to the test results in Table 3, it can be concluded that the higher specification of service plan provides better handling of concurrency control in terms of multiple users accessing the web application in a period of time. For instance: S2 Standard Plan shows better performance compared to S1 Standard Plan with the range of 250 users to 500 users at 5.78s. S2 plan with 500 concurrent users showed a better management of multiple users where its response time has shortened down from 6.15s to 5.78s. However, S3 does not show much improvements compared to S2 in terms of response time. The response time has slightly increased in S3 with 500 user loads compared to S2 with 500 user loads. This means that the S2 plan considered the most cost-efficient plan to be upgraded for this project with estimated RM624.96 per month.

6.0 Managed Databases

Managed database service is considered one of the essential features provided by cloud providers under the form of platform-as-a-service. For instance: Amazon Relational Database Service (Amazon RDS) allows the user to operate and scale a relational database in the cloud services (Amazon, 2018). On the other hand, Microsoft Azure provides Azure SQL Database that serves as a relational database using Microsoft SQL Server Engine (Microsoft, 2018). Furthermore, the Azure SQL Database provided by Microsoft Azure has serves as a platform-as-a-service (PaaS) database or a database-as-a-service (DBaaS) that was being optimized for software-as-a-service (SaaS) application development. Typically, the Azure SQL Database provides a wide compatibility of integration with various modules that uses most of the features of SQL Server.

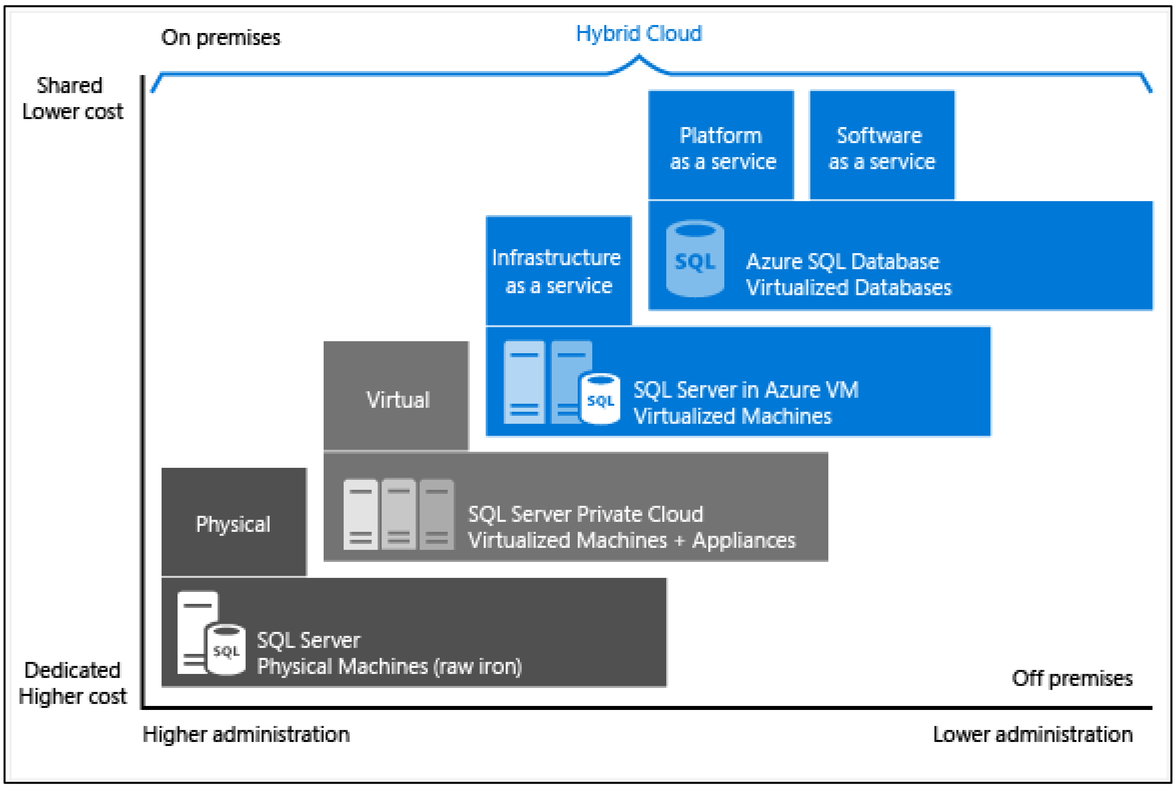


Figure 39: Features for different level of administration over cost in Azure managed database

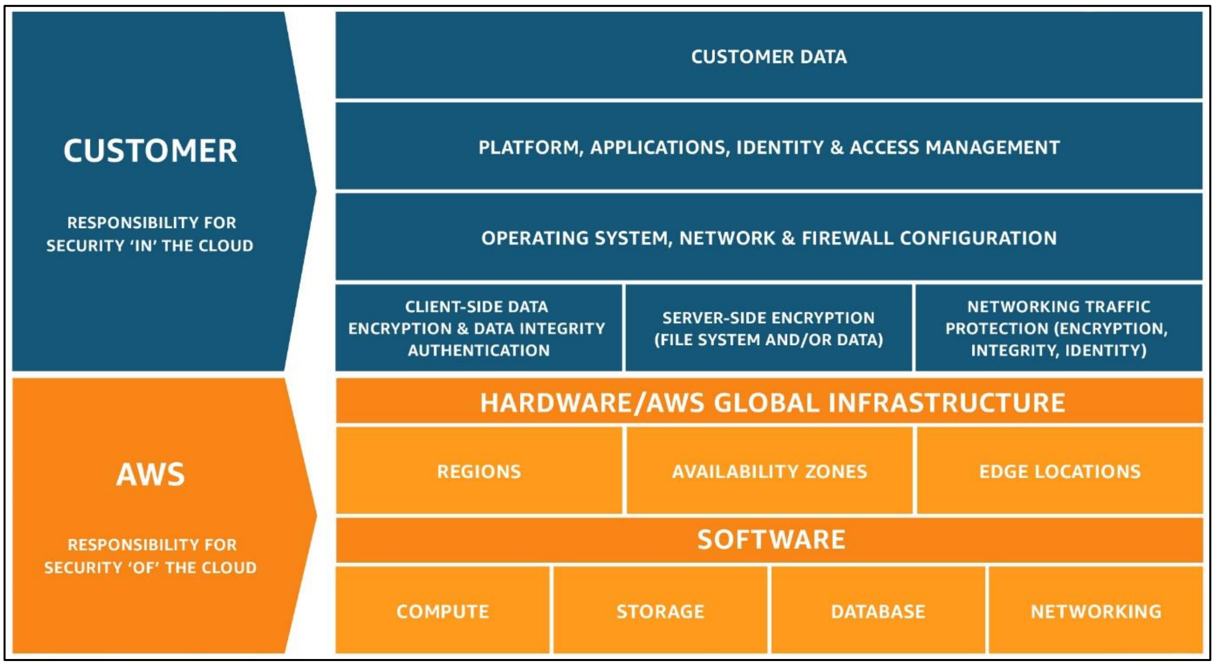

CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING

Figure 40:AWS Shared Responsibility Model (Amazon, 2018)

The most general differences between the Azure SQL Database and Amazon RDS is that Azure SQL targets a smaller database usage of companies compared to Amazon who comparatively a wider segment of users. Amazon RDS is capable of storing 1TB per database instance whereas Azure prefers sharing of data if the storage requirement exceeds 10 GB. However, it is very important for an organization to choose the most suitable and cost-efficient database service to reduce operational cost with high productivity database performance. With the help of Azure managed database, the developers are allowed to build their systems directly on top of the web services and databases that are capable of optimizing the scale of performance. Moreover, the operational tasks of the organization such as handling and maintaining the virtual machines in the servers can be reduced since the tasks are being taken over by Azure services. The remaining tasks are only scaling the performance of the services according to business needs with a few clicks. By doing so, the developers can focus more on other development activities instead of handling and managing the system at all time. This resulted in improving productivity of the development team especially for the management of database.

The Azure SQL managed database can be accomplished through several ways:

Approach	**Description**
Azure portal	Manage the database through the database's resource group. The performance of database can be optimized through Azure portal.
PowerShell	PowerShell cmdlets are fully utilized for Azure SQL managed database. For example: *New-AzureRmSqlDatabase* is used to create database, *Get-AzureRmSqlDatabase* is used to get one or more databases and *Set-AzureRmSqlDatabase* is used to set properties for a database, or moves an existing database into an elastic pool.
Azure CLI	Azure CLI SQL Database commands is used to mange Azure SQL database using Cloud Shell to run the CLI on the browser. For instance: *az sql db list-usages* is a command that used to return database usages and *az sql db list-editions* is used to list out the available service objectives and storage limits of a database.
Transact-SQL	Transact-SQL is also known as T-SQL where it utilizes T-SQL commands on the Azure portal, SQL Server Management Studio or other program which is able to connect to an Azure SQL Database. For instance: *ALTER DATABASE (Azure SQL Database)* is used to modify an Azure SQL database.
REST API	REST API is another approach to create and manage the Azure SQL databases using requests methods. For example: *Servers - Create Or Update* is used to create or update a new server.

Table 4: List of approaches for Azure Managed Database (Microsoft, 2017)

The table below justify the factors of selecting Azure managed database:

Consideration Factor	**Description**
New or existing application/service	Migrating existing applications requires to emulate on-premises application behavior where new applications does not require the emulation.
Application/Service requirements	With Azure SQL Database, the developer is allowed to create a database as part of a managed instance or create a database that is either a single database or a database that is part of an elastic pool.
Database size	Since the developed system does not requires a large capacity of storage, the current space provided by Azure managed database is sufficient with upgradable plan if needed.
Willingness to (Re)architect/partition for economies of scale	Azure managed database is capable to scale-out using technology such as Federations in Windows Azure SQL Database to meet the scale-out requirements. Furthermore, a degree of re-architecture is needed to apply the scale-out function.

Table 5: Factors for choosing Azure managed database (Microsoft, 2017)

7.0 Conclusion

The researcher has gained a lot of knowledge through the investigation of the Maersk Line system since the beginning of this project. Firstly, understanding the concept of Microsoft Azure is mandatory to gain expertise in the area of cloud computing. Furthermore, there are a list of services available in Azure Cloud Services and the developer has to understand which services is applicable for the project. As each of the chapter explains the requirement and guideline of the project, the researcher has to refine every point to fit the documentation and this has resulted in further understanding about cloud computing knowledge of the researcher.

Time management is important to carry out studies about the web services and it is extremely important to keep the project in progress so that the fully functional system can be delivered on time. Planning a project timeline has assisted the researcher to focus in deliverables of system in a given timeframe and accomplish every task in a given timeline. This could also avoid procrastination of the project delivery as well.

The Maersk Line is believed to meet the objectives mentioned in the early stage of this documentation. The idea is very straight forward to improve the business process of Maersk Line through the Microsoft Azure cloud computing service platform by providing a reliable online system for the customer to manage the container bookings and routes. It will be fully utilized by the users of Maersk Line in terms of good usability of the system.

8.0 Reference

Amazon, 2018. *aws.* [Online] Available at: https://aws.amazon.com/compliance/shared-responsibility-model/ [Accessed 9 April 2018].

aws, 2018. *aws.* [Online] Available at: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html [Accessed 10 April 2018].

CraftCMS, 2018. *CraftCMS.* [Online] Available at: https://craftcms.com/support/remove-index.php [Accessed 20 March 2018].

kth, 2018. *kth.* [Online] Available at: https://www.kth.se/social/files/57db8d9ef276542790443813/php-mvc-fw.pdf [Accessed 20 March 2018].

Microsoft, 2017. *Microsoft Azure.* [Online] Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-servers-databases
[Accessed 9 April 2018].

Microsoft, 2017. *Microsoft Azure.* [Online] Available at: https://azure.microsoft.com/pt-br/blog/choosing-between-sql-server-in-windows-azure-vm-windows-azure-sql-database/
[Accessed 9 April 2018].

Microsoft, 2018. *Microsoft Azure.* [Online] Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas
[Accessed 10 April 2018].

Narumoto, M., 2017. *Microsoft Azure.* [Online] Available at: https://docs.microsoft.com/en-us/azure/architecture/patterns/retry [Accessed 12 March 2018].