A · P · U

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Module Code	:	CT071-5-3-DDAC
Module Name	:	Designing & Developing Cloud Applications
Project Title	:	Maersk Line Container Management System
Student ID	:	TP034620
Student Name	:	Tan Sin Wei
Intake Code	:	UC3F1706SE
Lecturer Name	:	Dr. Kalai Anand A/L Ratnam
Hand in Date	:	13 April 2018

Acknowledgement

Firstly, I would like to express my gratitude to my DDAC lecturer, Dr. Kalai for sharing his knowledge and guiding me throughout this semester. Thanks to him, I have learned the basics of cloud computing, as well as setting up and hosting my application in Azure hosting server. Besides, I would like to thank my fellow friends who provided guidance and support that keeps me in smooth progress. We helped each other by discussing our progress and checking each other's work. Other than that, we have each other to perform testing and ensure that the system is working fine before hosting it using Azure. Thanks to the help of Dr. Kalai and my fellow friends, I am able to complete this assignment fluently without much mistakes.

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. 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.

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

1.2 Objectives

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

1.3 Scopes

- Able to manage import, export, transshipment processing, and gate operations.
- Able to scale the system based on the needs of demands during peak seasons.
- Improves profitability, reduce costs, increases productivity, eradicates errors and optimizes resources to future-proof your cargo handling business for high performance.
- Provide assurance & reliability through Failover Management.

- Accurately allocates inbound containers to yard locations and plan outbound containers to individual haulier vehicles, delivering an exceptional level of automation and removing human error.
- Manage entire booking process from schedule search to booking confirmation.

1.4 Requirement Specification

- Design and develop a single tenant web application hosted on Microsoft Azure as an App Service.
- Consume Relational Database
- Consist of 5 - 10 interlinked pages.
- Provide quality content and design.
- Analyze web application performance with monitoring tools.
- Able to scale the solution to meet the needs of demands during peak seasons.
- Source code to place in source control management services.

1.5 Summary of Major Functions / Solution Content

- **Administrator**
    - Manage agent accounts.
    - Manage ship details.
    - Manage shipment schedules.
    - Manage shipment schedule booking requests.
- **Agent**
    - Manage customer records.
    - Manage customer's item details.
    - View booking done by the agent.
    - View available shipment schedule for booking.
    - Book shipment schedule for customer's item.

2.0 Project Plan

WBS	Task Name	Duratic	Start	Finish	Predecessors
1	**Maersk Line Container Management System**	**44 days**	**Tue 2/13/18**	**Fri 4/13/18**	
1.1	**Requirement Gathering and Analysis**	**5 days**	**Tue 2/13/18**	**Mon 2/19/18**	
1.1.1	Gather requirements from Maersk Line representative	2 days	Tue 2/13/18	Wed 2/14/18	
1.1.2	Analyze gathered requirements	1 day	Thu 2/15/18	Thu 2/15/18	3
1.1.3	Generate System Requirement Specification	2 days	Fri 2/16/18	Mon 2/19/18	4
1.1.4	Project requirements finalized	0 days	Mon 2/19/18	Mon 2/19/18	
1.2	**System Design**	**4 days**	**Tue 2/20/18**	**Fri 2/23/18**	
1.2.1	Identify design considerations	1 day	Tue 2/20/18	Tue 2/20/18	5
1.2.2	**Design system information flow**	**1 day**	**Wed 2/21/18**	**Wed 2/21/18**	
1.2.2.1	Generate Use Case Diagram	1 day	Wed 2/21/18	Wed 2/21/18	8
1.2.2.2	Generate Sequence Diagram	1 day	Wed 2/21/18	Wed 2/21/18	8
1.2.3	Design system architecture	1 day	Thu 2/22/18	Thu 2/22/18	11
1.2.4	Design cloud architecture	1 day	Fri 2/23/18	Fri 2/23/18	12
1.2.5	Finalized system design	0 days	Fri 2/23/18	Fri 2/23/18	
1.3	**System Development**	**24 days**	**Mon 2/26/18**	**Thu 3/29/18**	
1.3.1	Develop modules for Administrator	8 days	Mon 2/26/18	Wed 3/7/18	13
1.3.2	Develop modules for Agent	8 days	Thu 3/8/18	Mon 3/19/18	16
1.3.3	Develop system user interface	3 days	Tue 3/20/18	Thu 3/22/18	17
1.3.4	Integrate system moduels	1 day	Fri 3/23/18	Fri 3/23/18	18
1.3.5	Perform Unit Testing and fixing	4 days	Mon 3/26/18	Thu 3/29/18	19
1.3.6	System is fully developed	0 days	Thu 3/29/18	Thu 3/29/18	
1.4	**System Deployment**	**6 days**	**Fri 3/30/18**	**Fri 4/6/18**	
1.4.1	Configure and host system to Azure	2 days	Fri 3/30/18	Mon 4/2/18	20
1.4.2	Configure and host database to Azure	2 days	Tue 4/3/18	Wed 4/4/18	23
1.4.3	Perform Performance Testing	2 days	Thu 4/5/18	Fri 4/6/18	24
1.4.4	System and database deployed on Azure	0 days	Fri 4/6/18	Fri 4/6/18	
1.5	**Project Documentation**	4 days	Mon 4/9/18	Thu 4/12/18	25
1.6	**Project Submission**	1 day	Fri 4/13/18	Fri 4/13/18	27
1.7	Maersk Line Container Management System Project Completed	0 days	Fri 4/13/18	Fri 4/13/18	

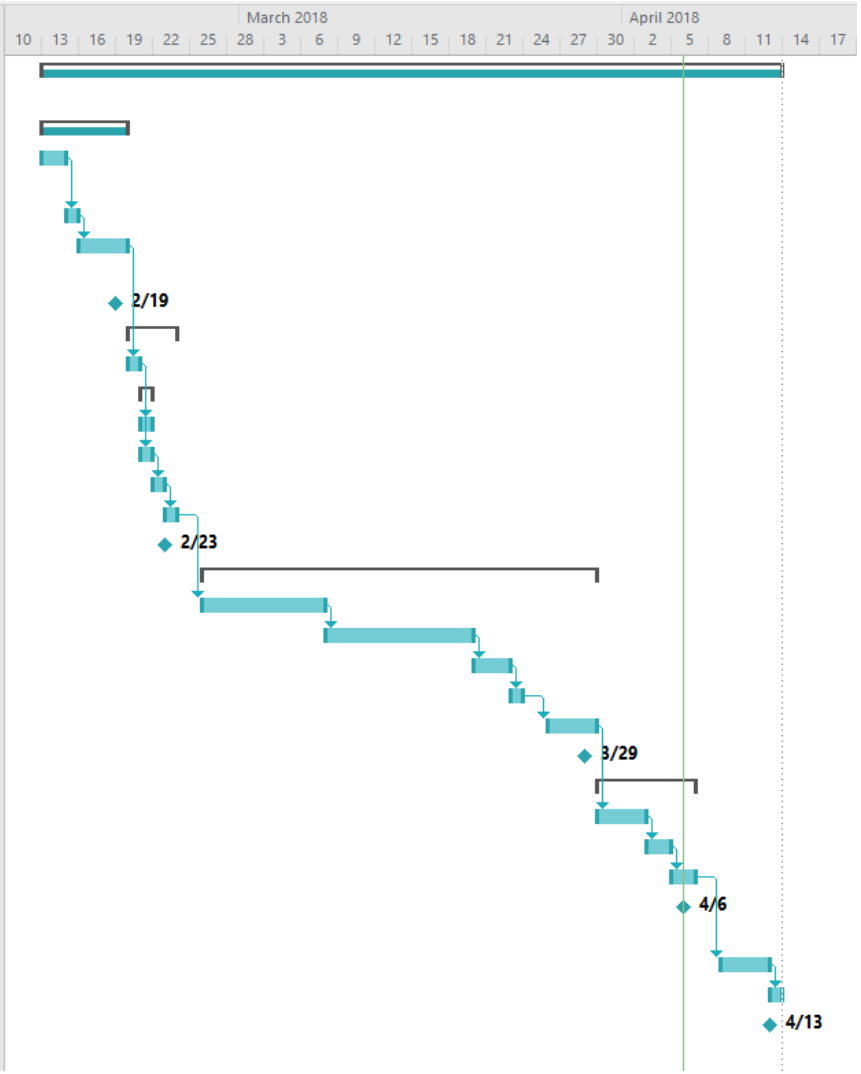

Figure 1: Project Plan Gantt Chart

3.0 Design

3.1 Design Considerations

Considerations and Assumptions are made before the system design process. One of them is that Maersk Line has provided Azure credit for the development team to host the Container Management System and Database on Azure. Besides, the credits are used to perform performance testing and configuration in order to deploy the system using the most suitable and efficient method. Moreover, it is assumed that Maersk Line is used to MySQL database and programming language, such as Java. Therefore, MySQL database and Java JSF are used to develop the Container Management System which will allow Maersk Line IT personnel to have a better understanding and maintainability. Furthermore, considering that the system users will be using the system via different devices with different screen size, the entire Container Management System is designed to be responsive to all the screen size. Other than that, it is assumed that this project will test run on SEA region for 6 months before implementing in other regions. Which means more application instance and database geo-replication will be needed in the future.

3.2 Modelling

3.2.1 Use Case Diagram

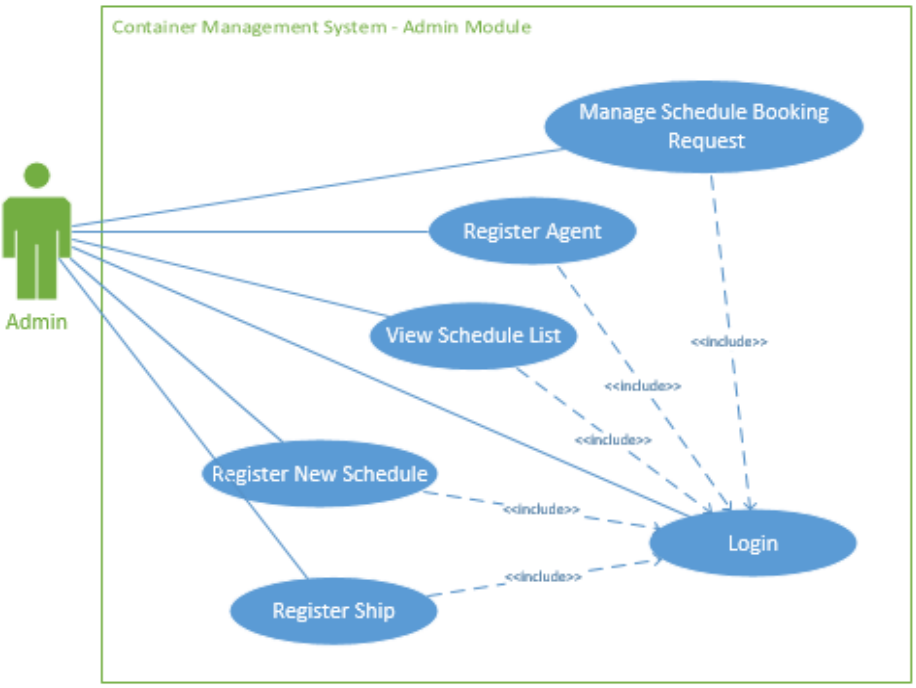

Figure 2: Use Case Diagram for Admin Module

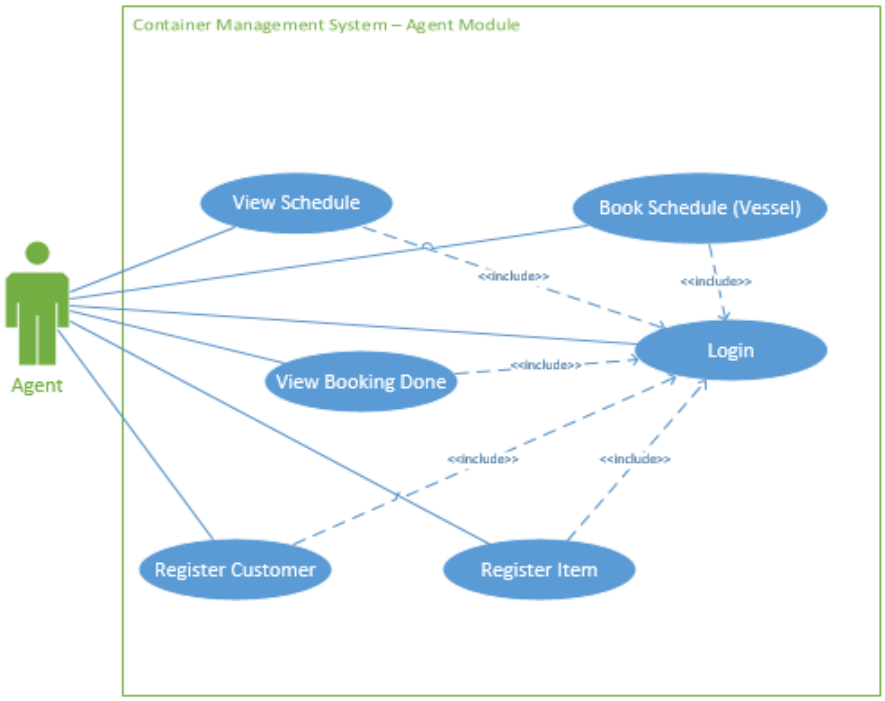

Figure 3: Use Case Diagram for Agent Module

3.2.2 Use Case Description

Use Case ID	UC1
Name	Login
Description	Allow admin and agent to login to the system.
Dependency	-
Actor	Admin and Agent
Pre-conditions	System is not logged in with any other user.
Event flow	1. Admin enter valid username and password. 2. System perform login validation. 3. Admin is redirected to the Admin Home Page.
Alternative flow	1. Agent enter valid username or password. 2. System perform login validation. 3. Agent is redirected to the Agent Home Page.

Post-condition	Admin or Agent is redirected to their Home Page with correct name displayed on the top right corner.

Table 1: Use Case Description for Login

Use Case ID	UC2
Name	Register Agent
Description	Allow admin to register new agent account.
Dependency	<<includes>>Login
Actor	Admin
Pre-conditions	System is logged in with an Admin account.
Event flow	1. Admin enter valid new agent account details. 2. System perform agent registration validation. 3. New agent account is registered into the database.
Alternative flow	1. Admin enter invalid new agent account details. 2. System perform agent registration validation. 3. Error message is displayed.
Post-condition	The newly registered Agent account can be used to login the system.

Table 2: Use Case Description for Register Agent

Use Case ID	UC3
Name	Register Ship
Description	Allow admin to register new ship details.
Dependency	<<includes>>Login
Actor	Admin
Pre-conditions	System is logged in with an Admin account.
Event flow	1. Admin enter valid new ship details. 2. System perform ship registration validation.

	3. New ship details are registered into the database.
Alternative flow	1. Admin enter invalid new ship details. 2. System perform ship registration validation. 3. Error message is displayed.
Post-condition	The newly registered ship details can be used to register new schedule.

Table 3: Use Case Description for Register Ship

Use Case ID	UC4
Name	View Schedule List
Description	Allow admin to view registered schedule list.
Dependency	<<includes>>Login
Actor	Admin
Pre-conditions	System is logged in with an Admin account and at least one schedule is registered.
Event flow	1. Admin click on Manage Schedule and Booking. 2. Admin click on Schedule List tab.
Alternative flow	-
Post-condition	The list of registered schedules will be displayed with all the details.

Table 4: Use Case Description for View Schedule List

Use Case ID	UC5
Name	Register New Schedule
Description	Allow admin to register new schedule list.
Dependency	<<includes>>Login
Actor	Admin

Pre-conditions	System is logged in with an Admin account and at least one ship is registered.
Event flow	1. Admin select a registered ship. 2. Admin fill in the other schedule details with valid details. 3. Admin click on Register button.
Alternative flow	1. Admin select a registered ship. 2. Admin fill in the other schedule details with invalid details. 3. Admin click on Register button. 4. Error message is displayed.
Post-condition	The new schedule is added into the schedule database.

Table 5: Use Case Description for Register New Schedule

Use Case ID	UC6
Name	Manage Schedule Booking Request
Description	Allow admin to view schedule booking request list and approve or reject the requests.
Dependency	<<includes>>Login
Actor	Admin
Pre-conditions	System is logged in with an Admin account and at least one schedule booking is requested.
Event flow	1. Admin select a schedule booking request and right click on it to view the context menu. 2. Admin select Approve from the context menu. 3. Admin select Yes for the confirmation dialog. 4. The approved schedule booking request is removed from the list.
Alternative flow	1. Admin select a schedule booking request and right click on it to view the context menu. 2. Admin select Reject from the context menu. 3. Admin type in the reason for rejecting this request.

	4. Admin select Confirm for the context menu. 5. The rejected schedule booking request is removed from the list.
Post-condition	The status of the schedule booking request is updated in the database and the agent will be notified. If the request is approved, the available weight for that specific schedule will be updated as well.

Table 6: Use Case Description for View Schedule Booking Request List

Use Case ID	UC7
Name	View Schedule
Description	Allow agent to view schedules that are available for booking.
Dependency	<<includes>>Login
Actor	Agent
Pre-conditions	System is logged in with an Agent account and at least one schedule is available.
Event flow	1. Admin click on Book Schedule in the navigation bar. 2. Admin select Available Schedule List Tab.
Alternative flow	-
Post-condition	All of the available schedules are listed with full details.

Table 7: Use Case Description for View Schedule

Use Case ID	UC8
Name	View Booking Done
Description	Allow agent to view his or her booking that have been done.
Dependency	<<includes>>Login
Actor	Agent
Pre-conditions	System is logged in with an Agent account and at least one booking is done.

Event flow	1. Admin click on Book Schedule in the navigation bar. 2. Admin select Booking List Tab.
Alternative flow	-
Post-condition	All of the bookings done by the logged in Agent are listed with full details.

Table 8: Use Case Description for View Booking Done

Use Case ID	UC9
Name	Register Customer
Description	Allow agent to register new customer record.
Dependency	<<includes>>Login
Actor	Agent
Pre-conditions	System is logged in with an Agent account.
Event flow	1. Admin enter valid new customer details. 2. Admin click on Register button. 3. System perform new customer registration validation.
Alternative flow	1. Admin enter invalid new customer details. 2. Admin click on Register button. 3. System perform new customer registration validation. 4. Error message is displayed
Post-condition	The new customer is registered into the database.

Table 9: Use Case Description for Register Customer

Use Case ID	UC10
Name	Register Item
Description	Allow agent to register new item record.
Dependency	<<includes>>Login

Actor	Agent
Pre-conditions	System is logged in with an Agent account and at least one customer record is registered.
Event flow	1. Admin select a customer record. 2. Admin enter the valid item details that belongs to the customer. 3. Admin click on Register button.
Alternative flow	1. Admin select a customer record. 2. Admin enter the invalid item details that belongs to the customer. 3. Admin click on Register button. 4. Error message is displayed
Post-condition	The new item is registered into the database.

Table 10: Use Case Description for Register Item

Use Case ID	UC11
Name	Book Schedule
Description	Allow agent to book schedule for registered item.
Dependency	<<includes>>Login
Actor	Agent
Pre-conditions	System is logged in with an Agent account and at least one item record is registered, and one schedule is available.
Event flow	1. Admin select an available schedule. 2. Admin select a registered item that weights lesser or equal to the available weight for the selected schedule. 3. Admin click on Book Schedule button.
Alternative flow	1. Admin select an available schedule. 2. Admin select a registered item that weights greater than the available weight for the selected schedule. 3. Admin click on Book Schedule button.

	4. Error message is displayed
Post-condition	The schedule booking request will be recorded into the database and sent to the admin.

Table 11: Use Case Description for Book Schedule

3.2.3 Sequence Diagram

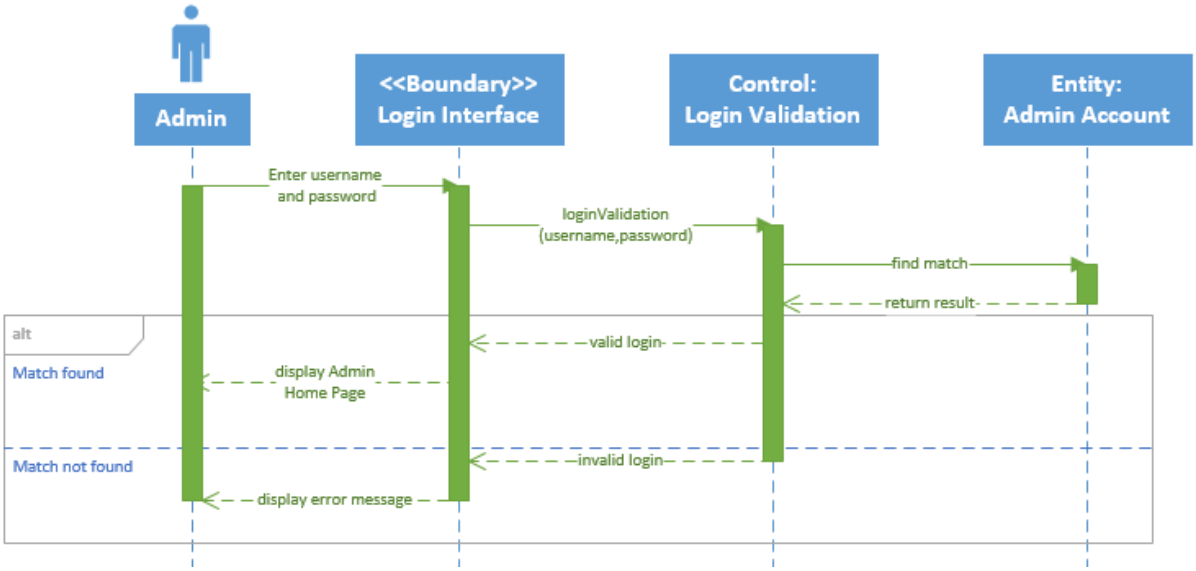

Figure 4: Sequence Diagram for Login

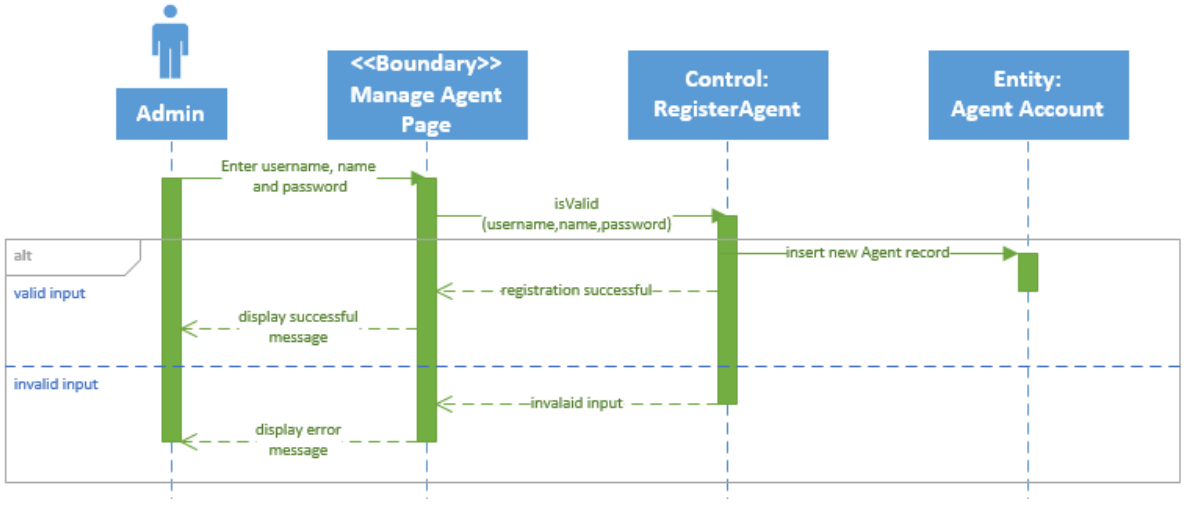

Figure 5: Sequence Diagram for Register Agent

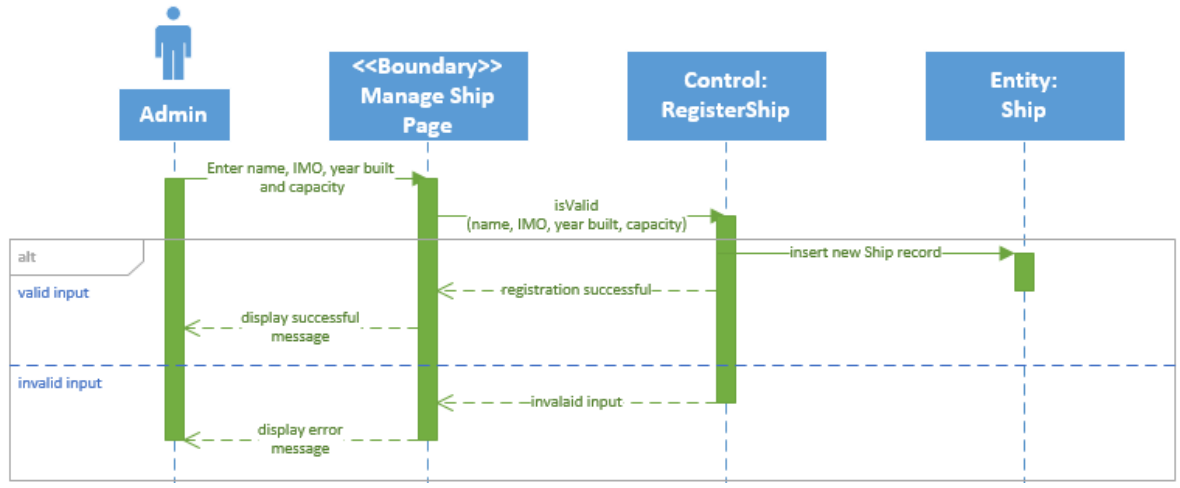

Figure 6: Sequence Diagram for Register Ship

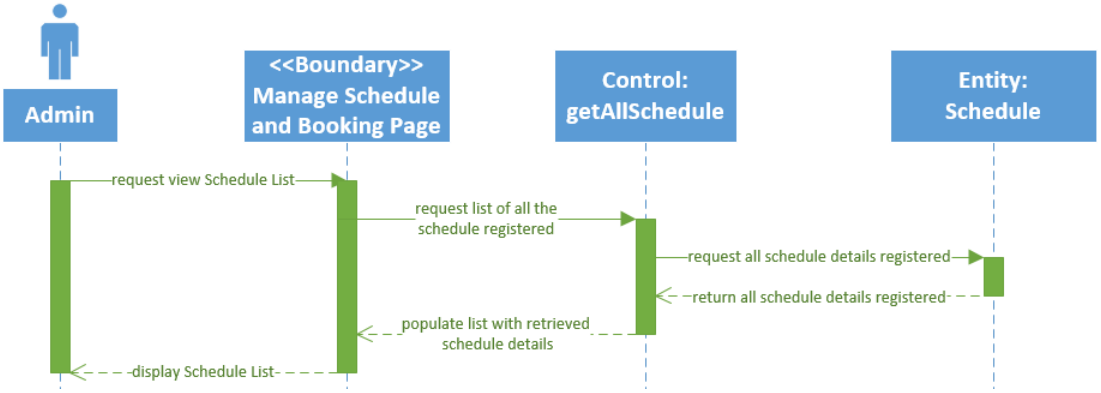

Figure 7: Sequence Diagram for View Schedule List

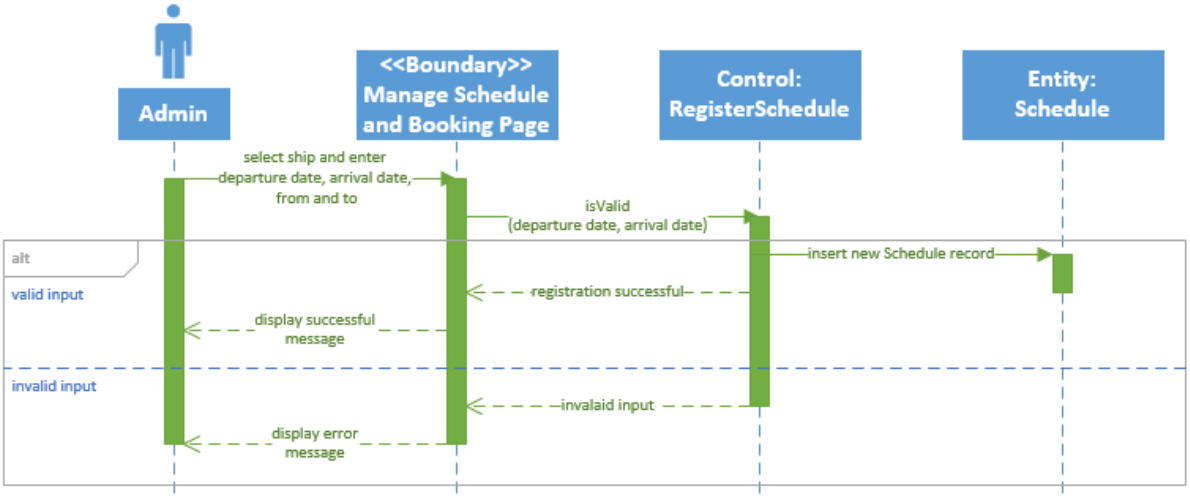

Figure 8: Sequence Diagram for Register New Schedule

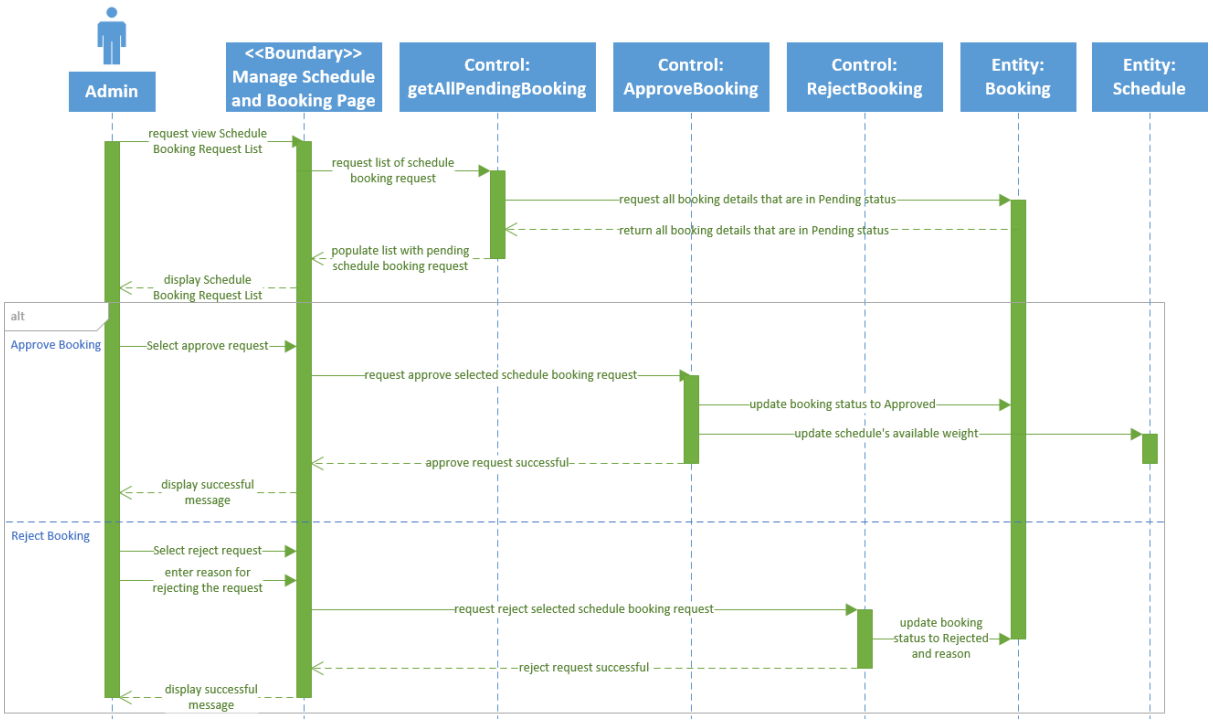

Figure 9: Sequence Diagram for Manage Schedule Booking Request

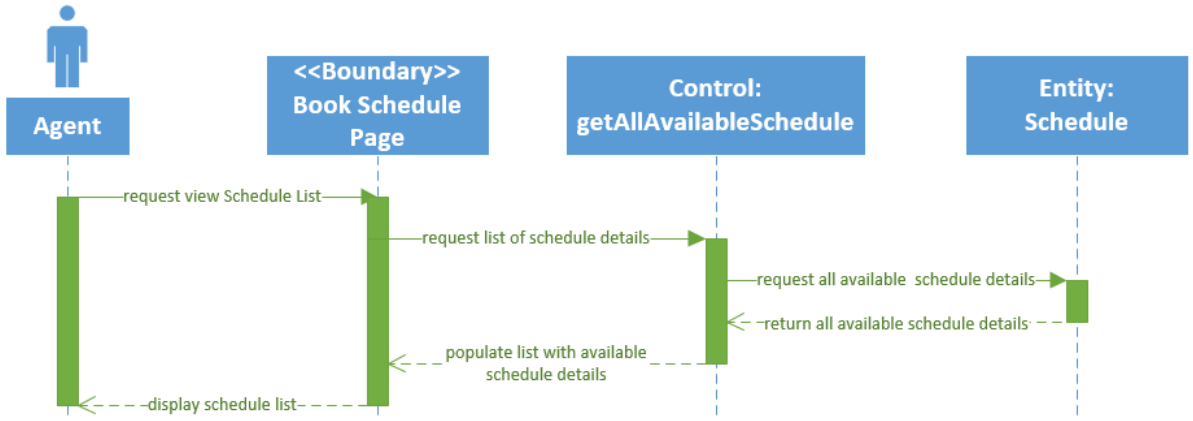

Figure 10: Sequence Diagram for View Schedule

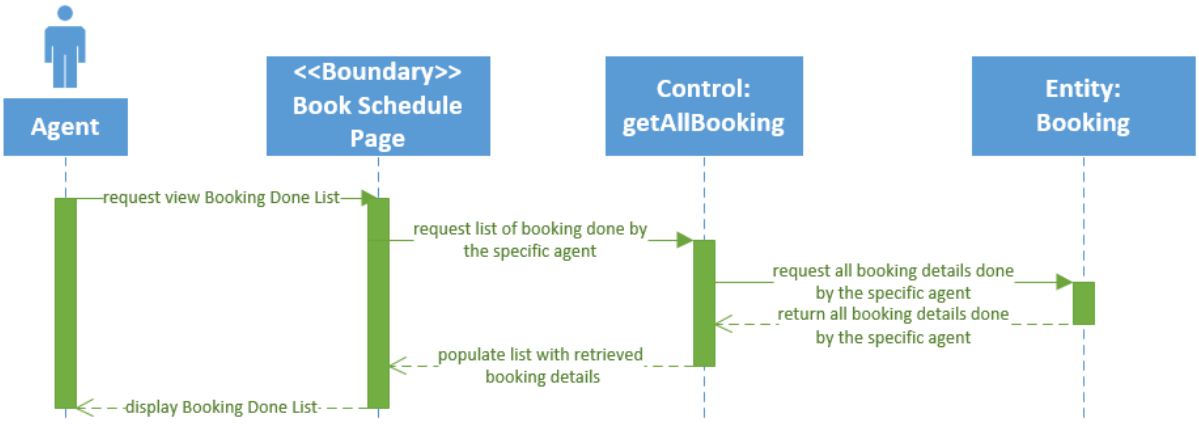

Figure 11: Sequence Diagram for View Booking Done

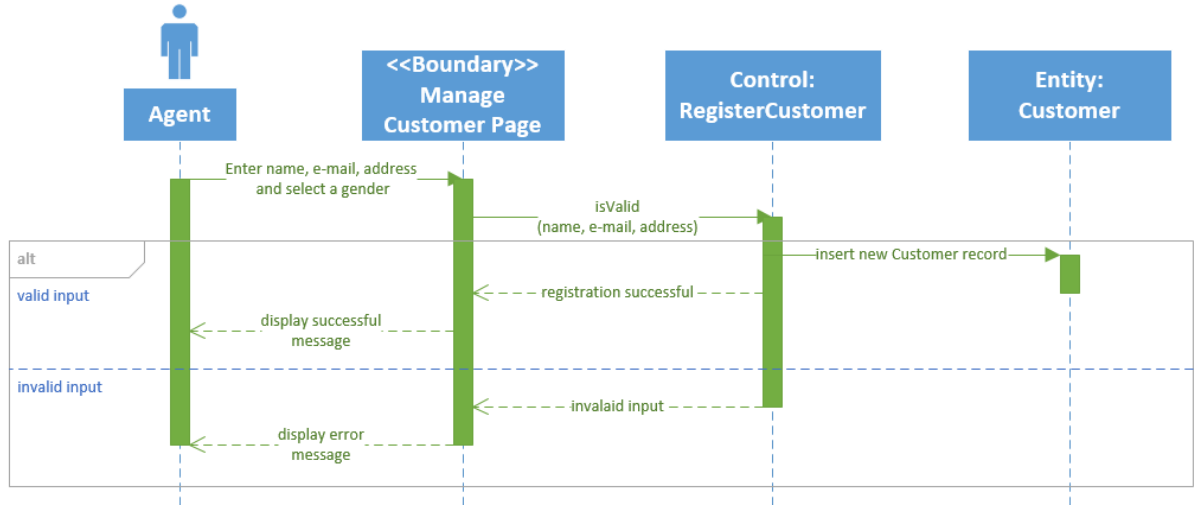

Figure 12: Sequence Diagram for Register Customer

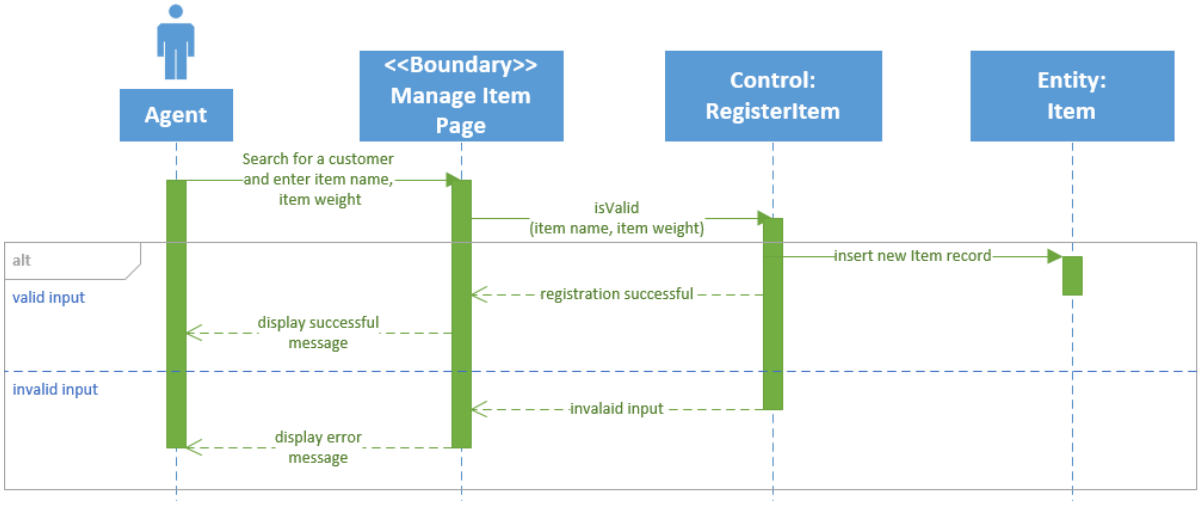

Figure 13: Sequence Diagram for Register Item

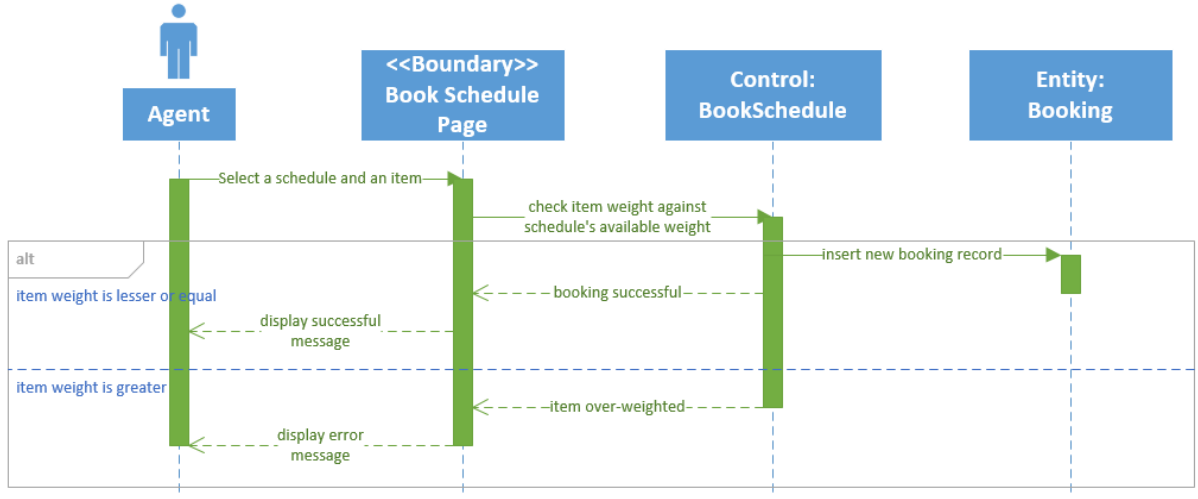

Figure 14: Sequence Diagram for Book Schedule

3.3 Class Diagram

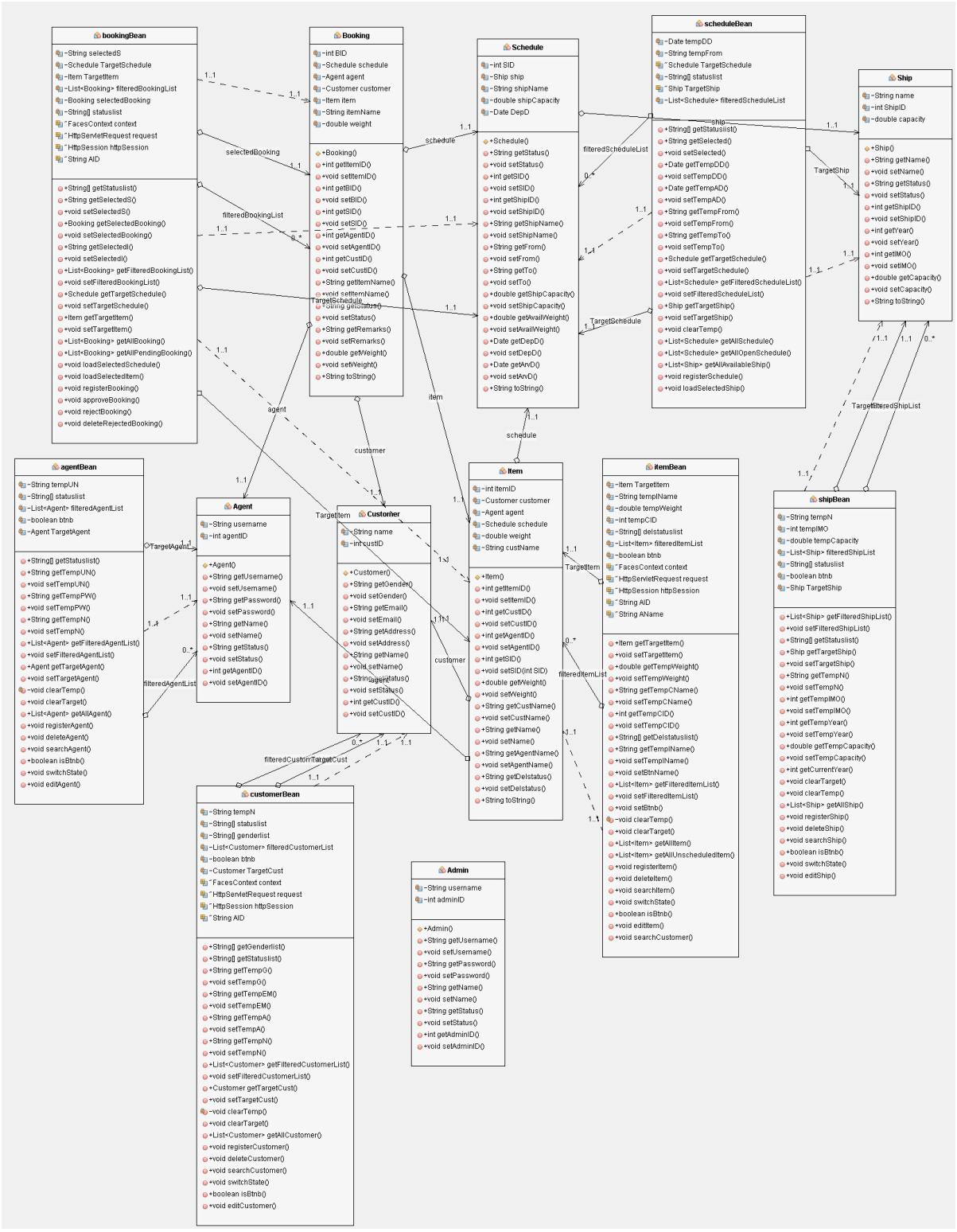

Figure 15: Class Diagram for Maersk Line CMS

3.4 Azure Cloud Architecture (Development purposes)

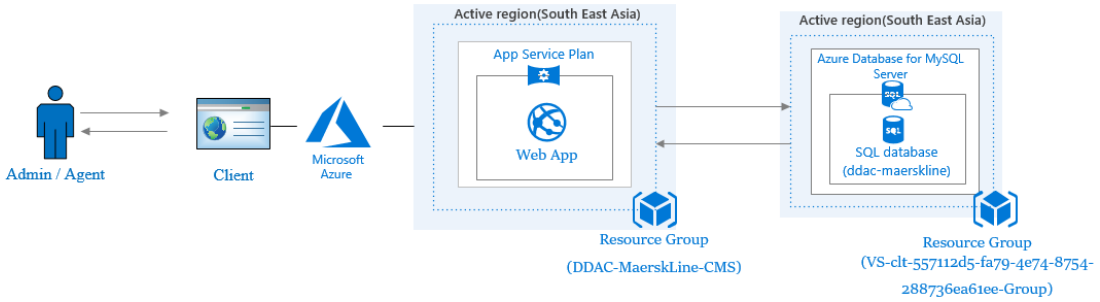

Figure 16: Azure Cloud Architecture for development purpose

The figure above shows the Azure Cloud Architecture diagram when the system is in development stage. While developing, the developer is given Azure credits enough for hosting the app service and database server on Azure. The app service is hosted using the F1 Free Tier while the database server is hosted using Basic Pricing Tier will minimum amount of vCore, storage, and backup retention days. The above pricing tier is chosen as the main objective in development phase is to ensure that the system and database is working after hosted on Azure.

3.5 Azure Cloud Architecture (After scaling)

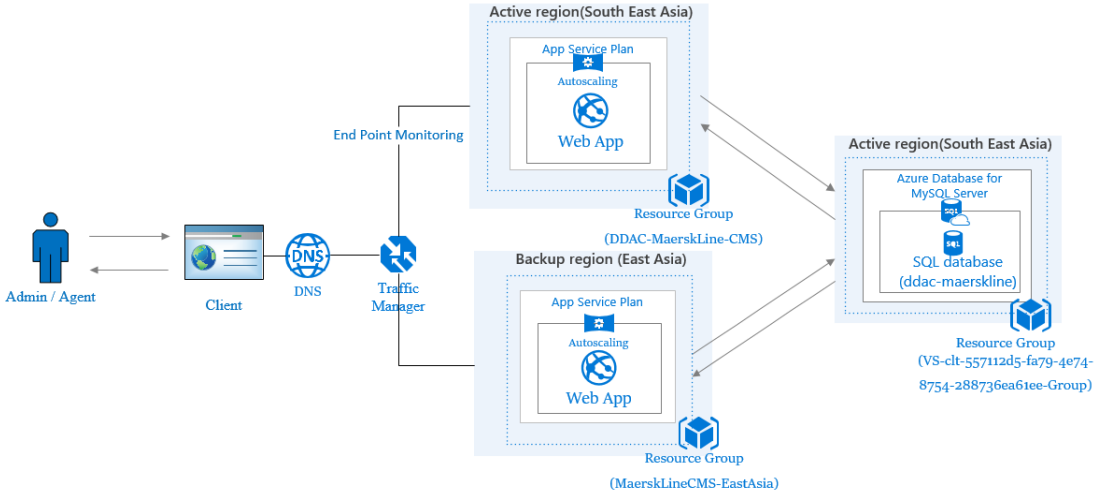

Figure 17: Azure Cloud Architecture after scaling

The figure above shows the Azure Cloud Architecture after scaling up to support actual operation and the app service in another region. In this stage, app service will be using S2 Standard pricing tier while the database server will be using General Purpose pricing tier with 4 vCores and 250 GB storage. As shown above, traffic manager is implemented to route the incoming requests to the prioritized region, which is the South East Asia (SEA) in this case. If failover occurs on SEA region, traffic manager will route the users to the next prioritized region. Further details of the scaling will be discussed in section 4.3.

4.0 Implementation

4.1 Application Development

The development of Maerks Line Container Management System was completed using Netbeans IDE and MySQL Workbench as the database management system. MySQL Workbench is chosen as MySQL is the world's most famous open source database developed by Oracle, which enables the delivery of reliable, high-performance and scalable web-based and embedded database application (Oracle, 2018). Netbeans IDE is chosen as it is bundled with built-in SQL support which facilitates the development process (Azharuddin, 2012). Besides, built-in support for the programming language that is going to be used to develop the system, Java JSF is included in Netbeans IDE as well (Netbeans, 2018).

Java JSF is chosen as the programming language and development framework to develop the system because it provides web application lifecycle management by the help of controller servlet, and a rich collection of component model with component rendering and event handling (Qusay, 2004). Moreover, the Model View Controller (MVC) model is implemented in the development of this system as Java JSF is a technology that based on MVC for separating logical layer and presentation layer (Tutorials Point, 2013). Below shows the system file structure.

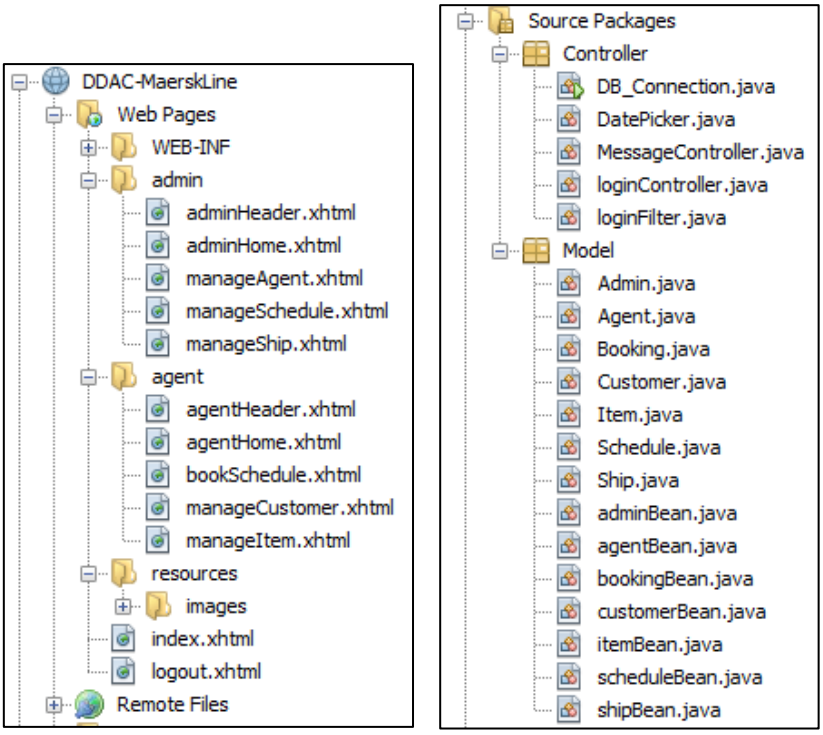

Figure 18: System file structure

As shown in the figure above, the system file structure is organized as MVC model. The Model folder consist of all the database models and their managed beans, Controller folder consist of the other controller java classes, and all the view, the xhtml files are located in the Web Pages folder. However, those java classes in the Controller folder are not the only controller used in this system. The JSF default controller, Faces Servlet is used as well. Besides, the view, xhtml files are separated based on user group. This is to facilitate the process of determining which user group the login user belongs to and grant different access to the web pages based on it. The web pages that are not in admin folder or agent folder are free to access by all users.

Upon launching the system, index.xhtml displayed as the login interface which allows the users to login to the system as an Admin or Agent. After a successful login as an Admin, the user will be redirected to the Admin Home Page with several functions categorized under Manage Agent, Manage Schedule and Booking, and Manage Ship. After a successful login as an Agent, the user will be redirected to the Agent Home Page with several functions categorized under Manage Customer, Manage Item, and Book Schedule. Below show the mentioned user interfaces.

Figure 19: User Interface for Login Interface (index.xhtml)

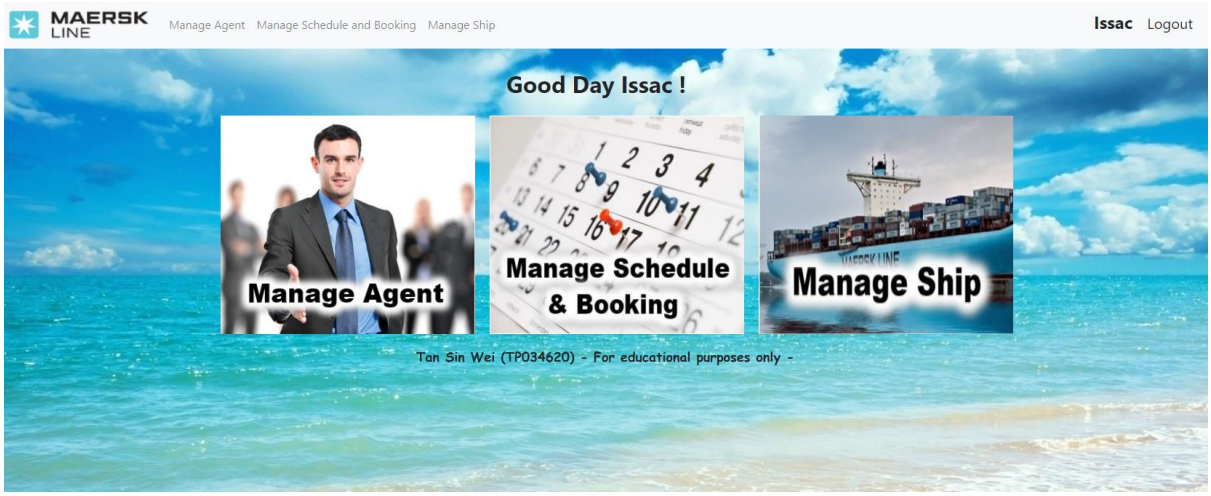

Figure 20: User Interface for Admin Home Page (adminHome.xhtml)

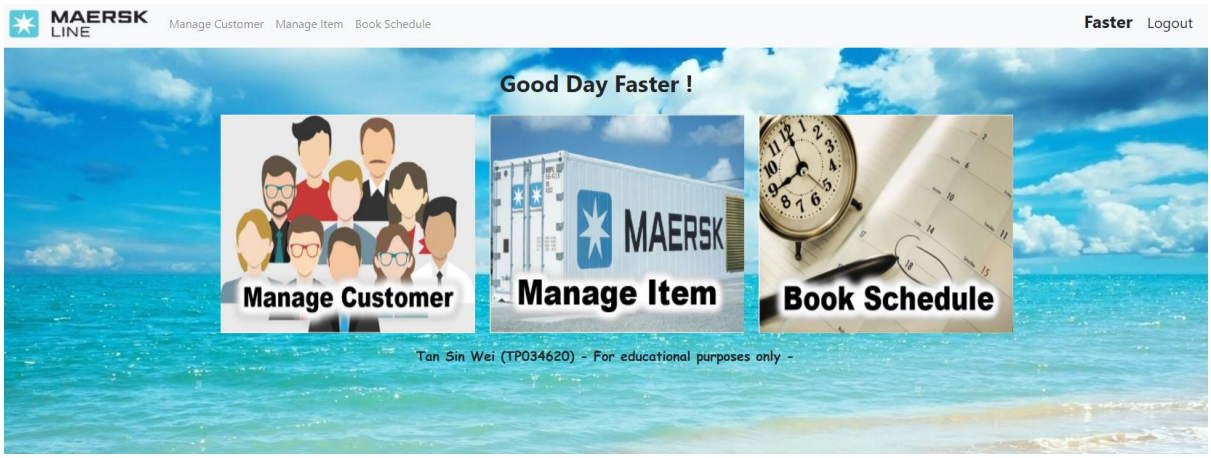

Figure 21: User Interface for Agent Home Page (agentHome.xhtml)

Validations are done to prevent unauthorized access to the system as well. Other than using the login validation mechanism, Filter is used to determine the user's access permission by checking on the session. If the logged in user is an Admin, the user will not be able to access Agent's web pages and vice versa. Besides, by using the Filter, the users are unable to access the protected pages by directly entering the URL of a protected page or pressing back button on the browser after logout. This is done by checking on the value stored in the session and checking on the URL the user was trying to access before rendering the web page. If a user is trying to access protected pages from login interface, the user will be redirected back to the login interface. While an Admin is trying to access Agent's web pages, the system will redirect the Admin back to Admin Home Page and vice versa. Below shows the implementation of Filter.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse resp = (HttpServletResponse) response;
    loginController session = (loginController) req.getSession().getAttribute("loginController");
    String url = req.getRequestURI();

    resp.setHeader("Cache-Control", "no-cache, no-store, must-revalidate"); // HTTP 1.1.  NO CACHE
    resp.setHeader("Pragma", "no-cache"); // HTTP 1.0.   NO CACHE
    resp.setDateHeader("Expires", 0); // Proxies.   NO CACHE

    if (session == null || !session.islogged) {
        System.out.println("1");
        if (url.indexOf("/faces/admin") >= 0 || url.indexOf("/faces/agent") >= 0) {
            System.out.println("2");
            resp.sendRedirect(req.getServletContext().getContextPath() + "/faces/index.xhtml");
        } else {
            System.out.println("3");
            chain.doFilter(request, response);
        }
    } else {
        System.out.println("4");
        if (url.indexOf("logout.xhtml") >= 0) {
            System.out.println("5");
            req.getSession().removeAttribute("loginController");
            resp.sendRedirect(req.getServletContext().getContextPath() + "/faces/index.xhtml");
        } else if (session.getUserType().equals("Admin")) {
            if (url.indexOf("/faces/agent") >= 0) {
                resp.sendRedirect(req.getServletContext().getContextPath() + "/faces/admin/adminHome.xhtml");
            } else {
                chain.doFilter(request, response);
            }
        } else if (session.getUserType().equals("Agent")) {
            if (url.indexOf("/faces/admin") >= 0) {
                resp.sendRedirect(req.getServletContext().getContextPath() + "/faces/agent/agentHome.xhtml");
            }else{
                chain.doFilter(request, response);
            }
        } else {
            System.out.println("6");
            chain.doFilter(request, response);
        }
    }
```

Figure 22: Code Snipper for loginFilter.java

Finally, all the required functionalities are developed for both Admin and Agent. The source code of the system will be available at https://github.com/tsweiform4e1/DDAC-Netbeans and a full demo of the system functionality in video format will be included in the attached CD and https://web.microsoftstream.com/video/e94abe43-d30c-489a-8508-4ea178639cf4?list=studio .

4.2 Azure Publishing

As mentioned earlier, the entire Maersk Line Container Management System and the associated database will be published and hosted on Microsoft Azure. However, the publishing and hosting method are different for the system and database. There will be only one portal for both Admin and Agent user as proper validation and security measures are implemented to prevent users from accessing unauthorized web pages.

4.2.1 Deploy Azure Database for MySQL Server

During the development phase, the system is developed on Netbeans along with MySQL Workbench as the database management system. Therefore, to deploy the MySQL database to Azure, an Azure Database for MySQL will be created first as shown in the figure below.

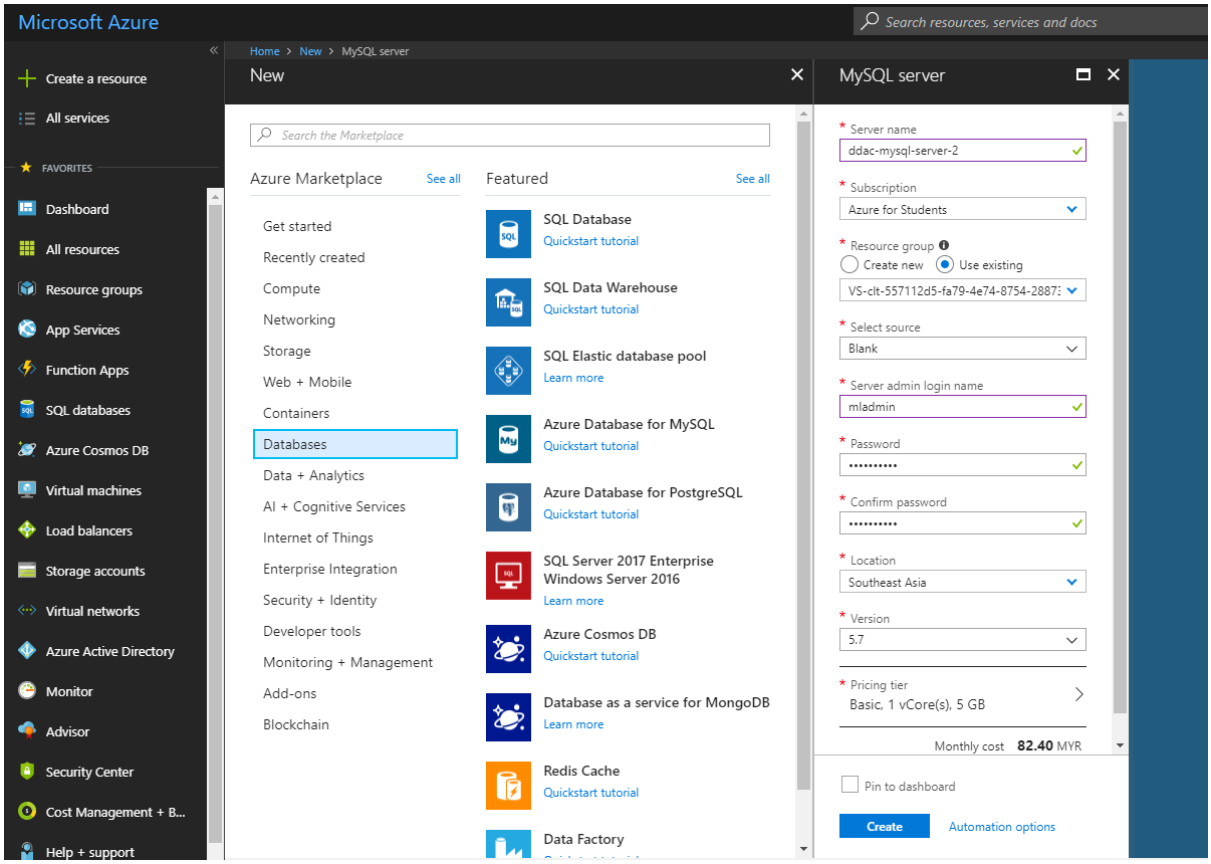

Figure 23: Create Azure Database for MySQL

After the server is successfully created and deployed, the developer will need to take note of the server properties as shown below in order to create a connection with the database in MySQL Workbench. The server properties needed can be found by navigating to the created server and access Properties.

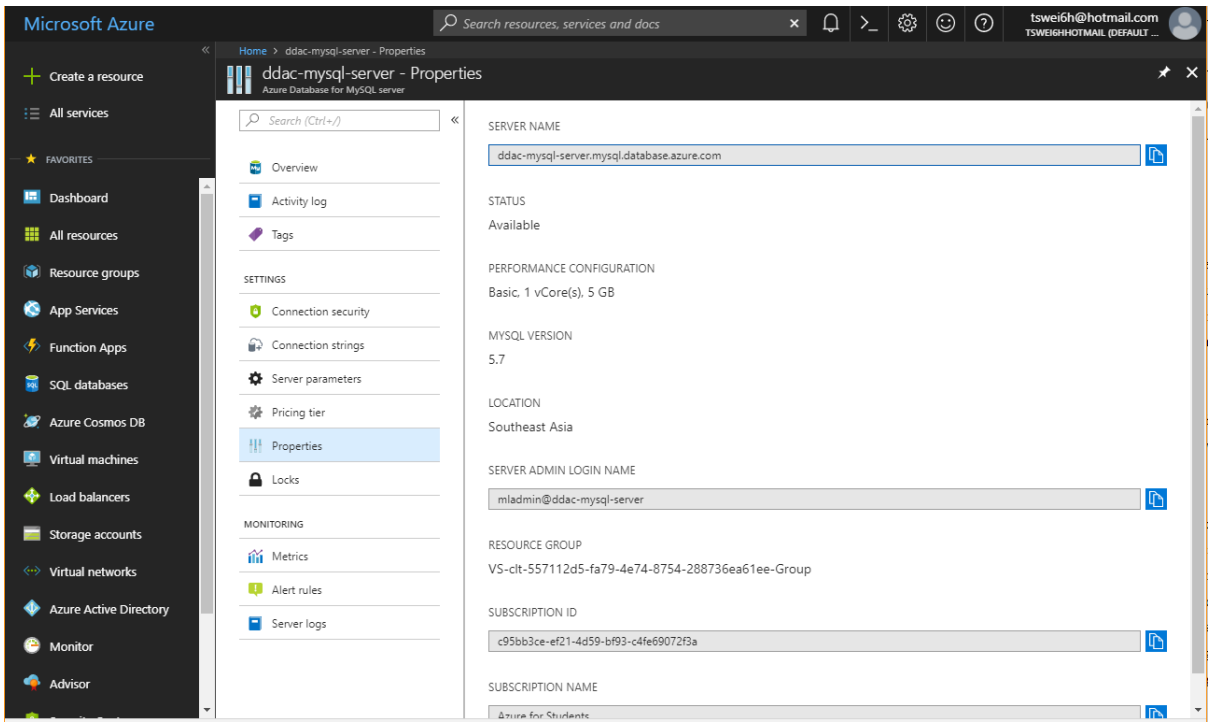

Figure 24: Properties of the Azure Database for MySQL Server

To configure and establish a connection between the Azure Database for MySQL Server and the database in MySQL Workbench, click the plus (+) icon upon launching the MySQL Workbench. Then, fill in the details as per the Azure Database for MySQL Server properties as shown in the figure below. After testing the connection and get a successful message, the developer will need to migrate or duplicate the entire database from local connection database to the Azure connection database.

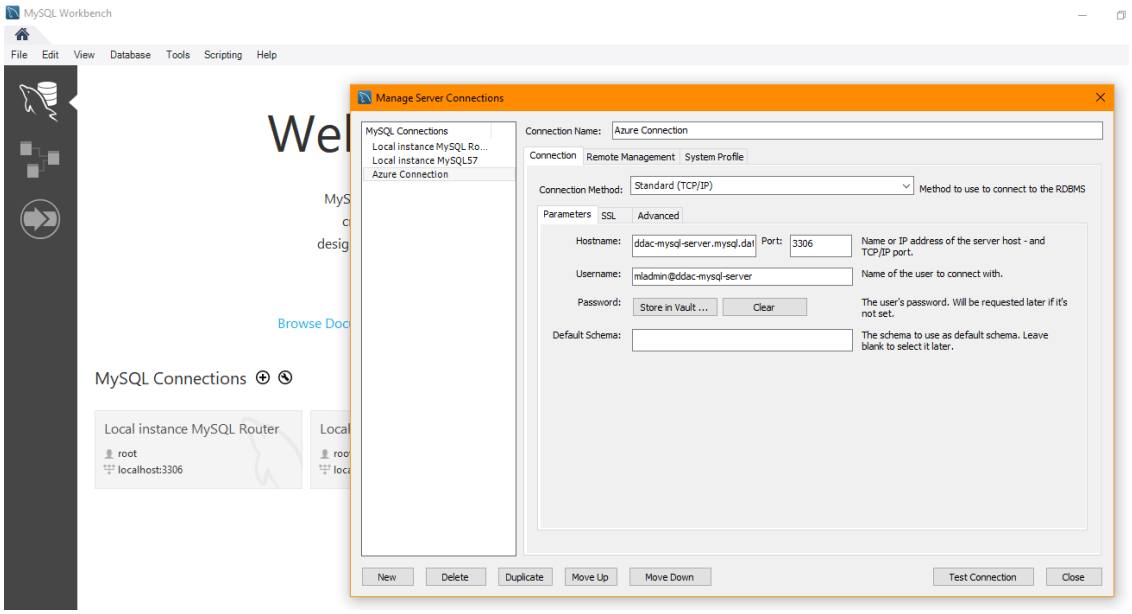

Figure 25: Setup connection between Azure Database for MySQL Server and MySQL Workbench

By this point, the database is successfully hosted on Azure through the connection with MySQL Workbench. As the system is developed using local environment first, the database connection string in the system will be pointing to the local connection. Therefore, the database connection string in the system is required to be replaced by the connection string provided by the Azure Database for MySQL Server as shown in the figure below.

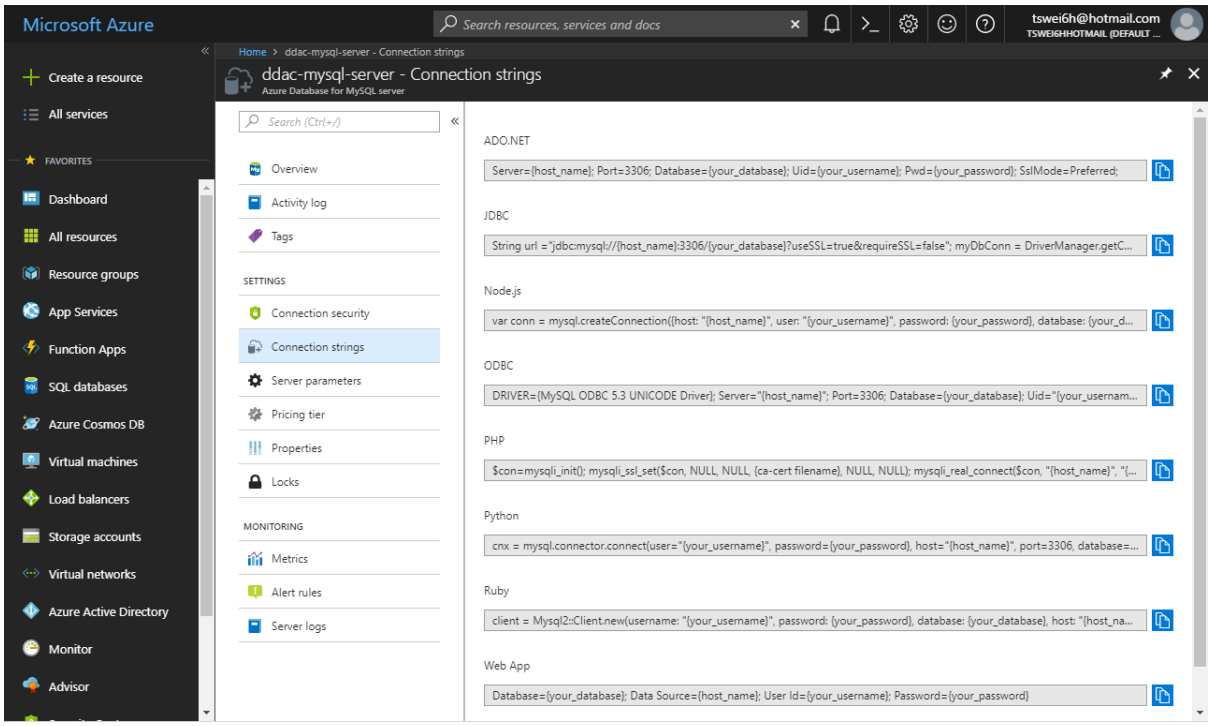

Figure 26: Connection strings for Azure Database for MySQL Server

The connection string used for this system is the one under JDBC as the system is developed using Java JSF. The JDBC connection string is then used to replace the local connection string in the system. However, the local connection string will be commented instead of removed for future testing purposes. Below shows the connection string used in the system. After this, the deployment for Azure Database for MySQL Server is completed. Any changes made using the MySQL Workbench or through the system will be synced.

```java
public class DB_Connection {

    public static void main(String[] args) {
        DB_Connection dbcon = new DB_Connection();
        System.out.println(dbcon.getConnection());
    }

    public Connection getConnection() {

        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            //con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ddac-maerskline?useSSL=false", "root", "12345"); // Local Connection
            con =DriverManager.getConnection("jdbc:mysql://ddac-mysql-server.mysql.database.azure.com:3306/"
                    + "ddac-maerskline?useSSL=true&requireSSL=false","mladmin@ddac-mysql-server", "Tswei12345");  //Azure Connection
            System.out.println("Connection to DB is successful !");

        } catch (Exception e) {
            System.out.println(e);
        }
        return con;
    }
}
```

Figure 27: Database connection string for the system

4.2.2 Deploy Maersk Line CMS Web Application

After the system is fully developed and tested, it will be prepared for deployment on Azure. Firstly, an Apache Tomcat 8 App Service will be created and deployed due to the system is developed based on Tomcat server as shown in the figure below. Besides, the app service is not deployed in the same resource group as the Azure Database for MySQL Server, so that the system security can be enhanced. This is because in the case when the resource group for app service is breached by the attacker, the attacker will not gain access to the Database. Other than that, deploying them in different resource groups facilitates the process of resource management and monitoring.

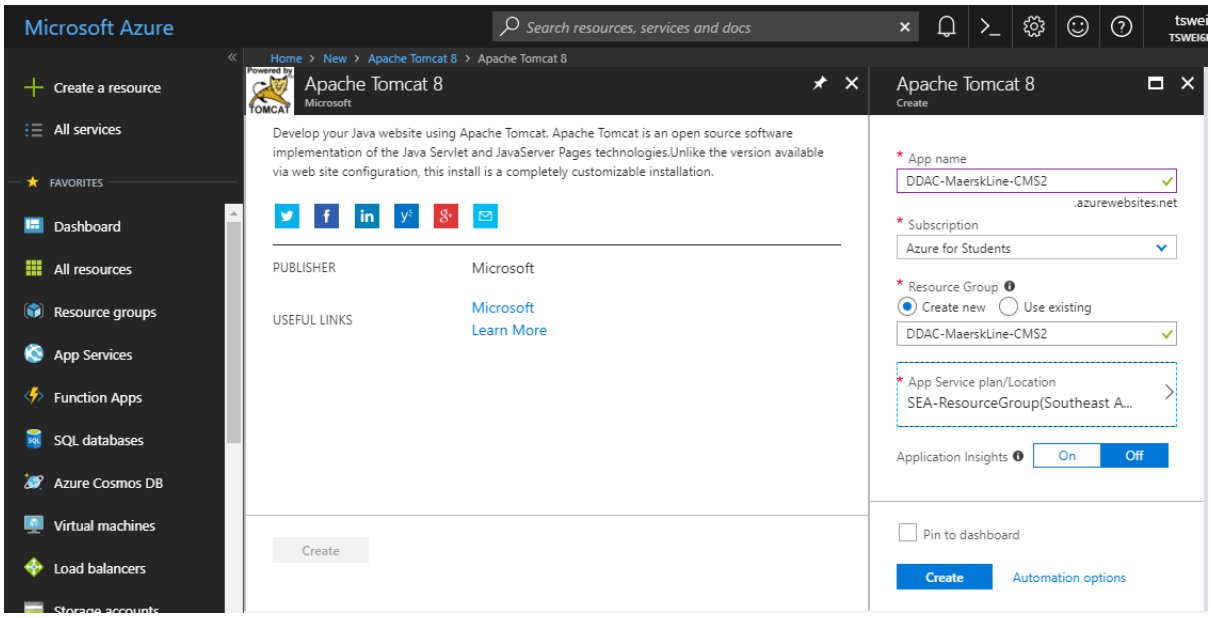

Figure 28: Create Tomcat App Service

In order to host the system using Tomcat App Service, a war file will need to be generated from the system. In Netbeans, it can be done by right clicking the **build.xml** in the system files, select **Run Target**, then select **dist**. Once the war file is generated, the system is ready to be deployed.

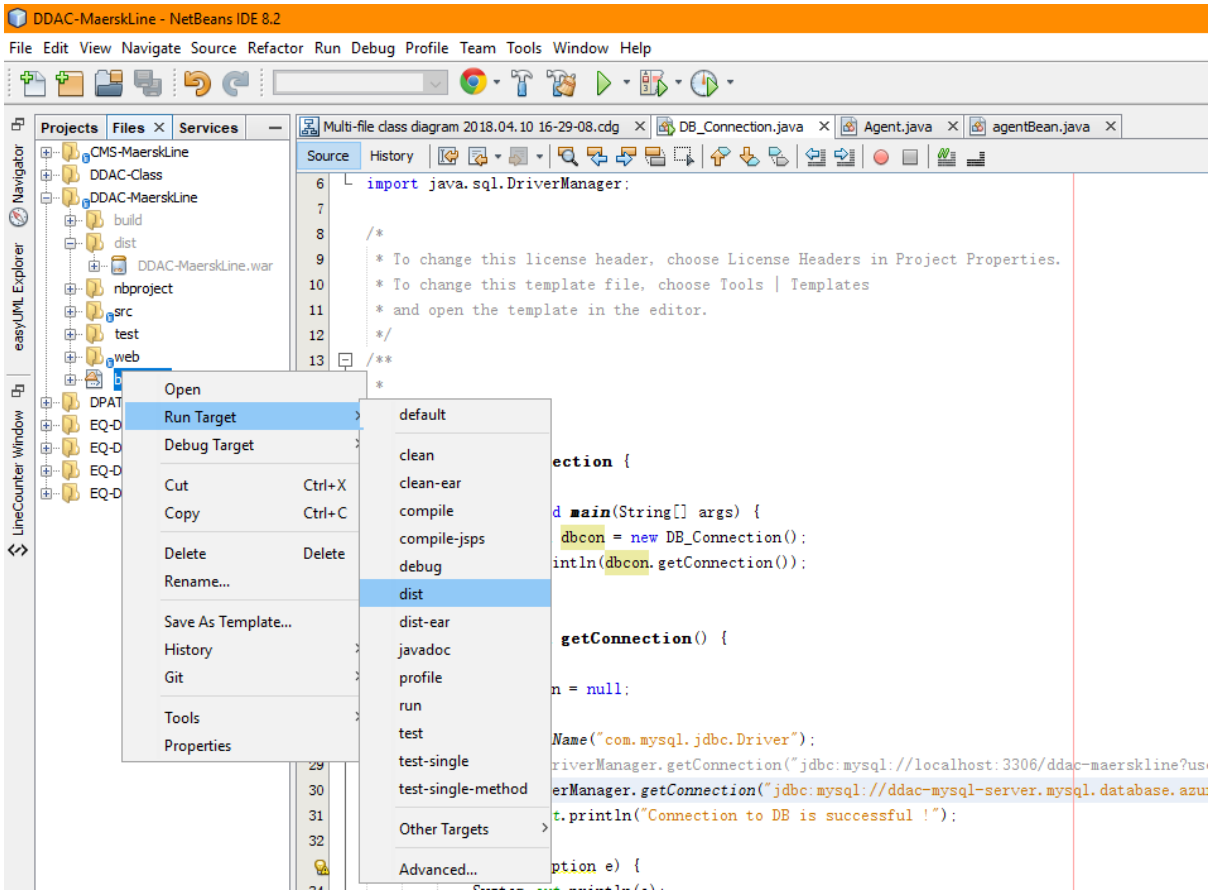

Figure 29: Generate system's war file

Once the Tomcat App Service is successfully deployed, navigate to the Tomcat App Service and select **Advanced Tools** from the side menu to navigate to the **Kudu Services**. Kudu is the engine behind git deployments in Azure web applications. Then, select **PowerShell** under the **Debug console** on the navigation bar. After that, navigate to the following location: **site/wwwroot/bin/apache-tomcat-8.5.24/webapps**. The location is where the system will be deployed. The developer will need to drag and drop the war file generated earlier into the location above as shown in the figure below.

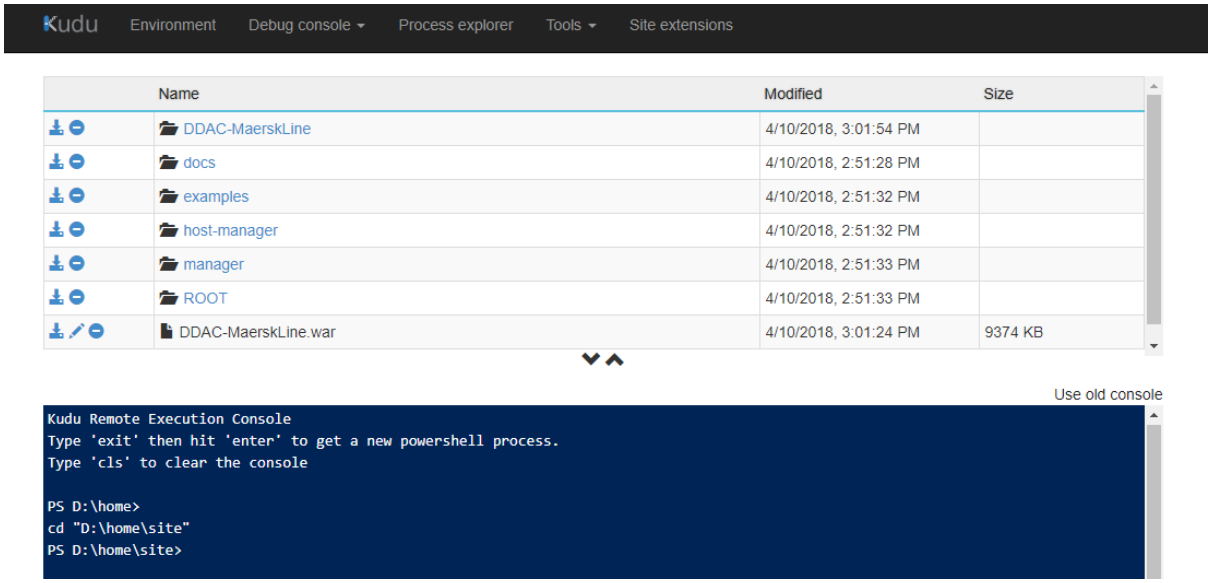

Figure 30: Deploy system war file on Kudu Service

Once the war file is deployed in this path, the deployment for Maersk Line CMS web application is basically done. The last step is to restart the Tomcat App Service and the system is ready to go. At this point, the Maersk Line CMS web application and its associated MySQL database is hosted on Azure platform and ready for operations.

4.3 Application Scaling

The Tomcat App Service is running on F1 Free pricing tier and the Azure Database for MySQL Server is running on Basic pricing tier with minimum vCore, storage, and backup retention period. Both of the services are running on the minimum pricing tier due to the credit limitation provided for development purposes. Once the system is ready to be published, both of the services will be scaled up in order to perform better and acquire more abilities that will be discussed right after.

4.3.1 Azure Database for MySQL Server Scaling

In consideration that this project is a test run on SEA region and will be implemented in more region in 6-month time, the server will be scaled up to General Purpose pricing tier with Gen 4 compute generation, 4 vCores, 250 GB storage, 7 days of backup retention period, and geo-redundant backup redundancy option as shown below.

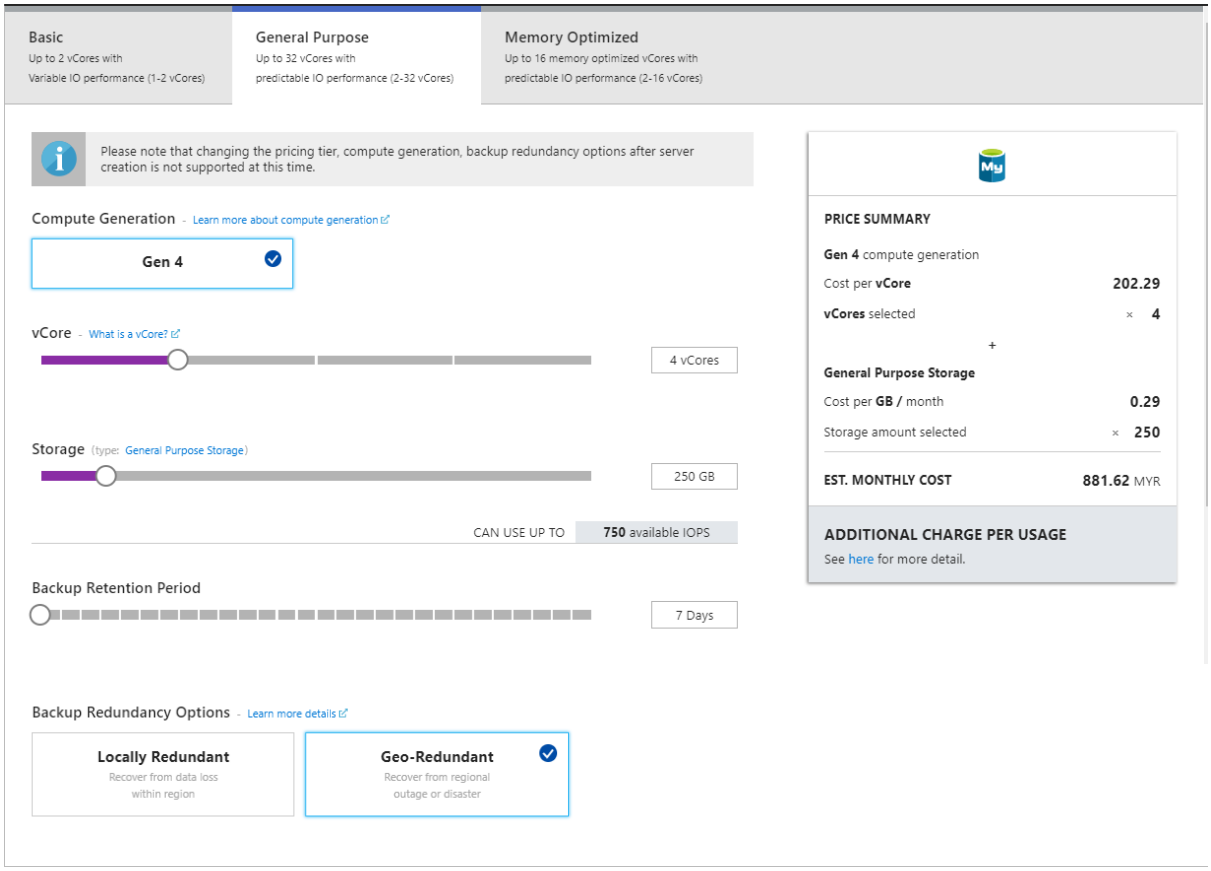

Figure 31: Azure Database for MySQL Server Scaling

There are several reasons that supports the selection of pricing tier shown in the figure above. First of all, to prepare the system for large number of concurrent users performing complicated operations, the vCore is scaled up to 4 units. The vCore represents the logical CPU of the underlying server hardware which provides computing resources (Microsoft Azure, 2018). Obviously, the more the vCore, the higher the computing capability and memory provisioned. Scaling up from Basic pricing tier to General Purpose pricing tier provides double the amount of memory per vCore compared to the Basic tier as well (Microsoft Azure, 2018).

Next, the storage will be scaled up to 250 GB. Although the data will only occupy average amount of storage, the storage is scaled up to prepare for uncertainties and future deployment of web application services from other regions. As this database stores important and sensitive transaction data, the backup retention period is scaled to the minimum, 7 days. This is to ensure that the database is frequently backup to minimize the risk of data lost. Lastly, the Geo-Redundant is chosen as the backup redundancy option, because the backups will be stored in the geo-paired region of the region where the server is created (Microsoft Azure, 2018). The other reason for choosing Geo-Redundant instead of Locally Redundant is that Geo-Redundant provide physical isolation of the datacenters which reduces the risk of data lost due

to natural disasters, power outages, or network outages. The storage is auto replicated to the paired region as well to facilitate the operation of the system that is going to be deployed in another region. Besides, in case of broad outage which requires recovery, one region will be prioritized out of every geo-paired region. Which means the system that are deployed across paired regions are guaranteed to have one of the regions recovered with priority (Microsoft Azure, 2018). This helps to maintain a high availability of the system.

4.3.2 Maersk Line CMS Web Application Service Scaling

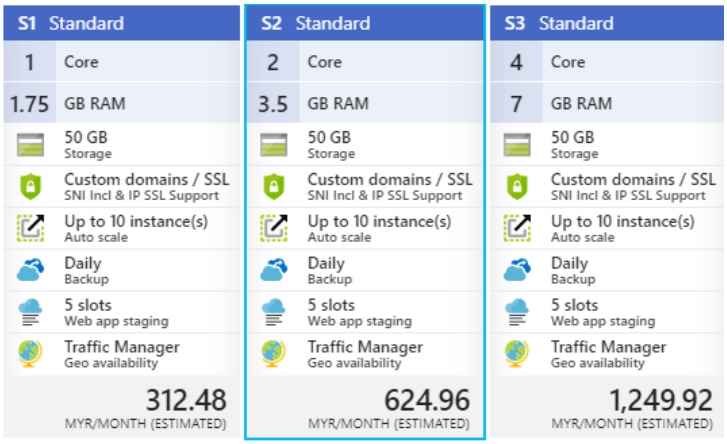

Figure 32: App Service pricing tier

As for the application service, it will be scaled up from F1 Free pricing tier to S2 Standard pricing tier. Basic pricing tier is not considered as it doesn't include geo availability and traffic manager. This system will be deployed in multiple region in 6-month time. Which means, traffic management and database geo replication are the essential features to be obtained. Traffic Management helps to direct user traffic to the best performed internet location based on the endpoints available. This reduces the chances of users from experiencing latency.

Other major difference is that Standard pricing tier provides up to 50 GB storage and 10 instances with auto scaling feature. The significant increase of instances (Virtual Machine) provides more cores, memory, and storage which increase the system performance (Microsoft Azure, 2017). Auto scaling is one of the important features to be obtained as it can automatically scale up or down the number of VM instances running based on the traffic needs. This helps to maintain the system performance during high traffic and save cost during low traffic.

In comparison with the F1 Free pricing tier, Standard pricing tier provides custom domains and SSL support for the system. This allows Maersk Line to use their own domain name instead of Azure's domain name and create an encrypted channel between the server and

client to prevent eavesdropping. Another feature that Standards provides and any pricing tier before it doesn't provide are Daily Backup and Web App Staging with 5 slots. The 5 slots are associated to every web application deployed under the pricing tier. One of the benefits for deploying the system in slot is to validate the system changes in a staging deployment slot before swapping it with the production slot (Microsoft Azure, 2016).

Above discussed the significant difference between Standard pricing tier with the Basic and Free pricing tier. In the case within Standard pricing tier, S2 is chosen instead of S1 and S3 is due to the consideration of difference in pricing and performance. As shown in the figure above, the estimated cost per month for S2 is double of S1, and S3 is double of S2. Due to the huge difference between pricing, there is the need to analyze which app service plan most suited for the system. The decision of choosing S2 app service plan is based on the performance testing done that will be discussed in section 5.2.

4.4 Reliability and Performances

As mentioned earlier, in actual operation, there will be two app services deployed in different region sharing a same Azure Database for MySQL Server while managed by Traffic Manager. With the help of Traffic Manager, the system reliability and performance can be greatly enhanced. This is because the Traffic Manager will use the Domain Name System to direct client request to the best performed endpoint based on the determined traffic routing method and the health of the endpoints. Besides, in the case of an app service failover, the Traffic Manager will automatically reroute the traffic to the next endpoint with lowest latency for the client which boosted the system reliability and performance. Below figure shows the overview of the Traffic Manager profile implemented.

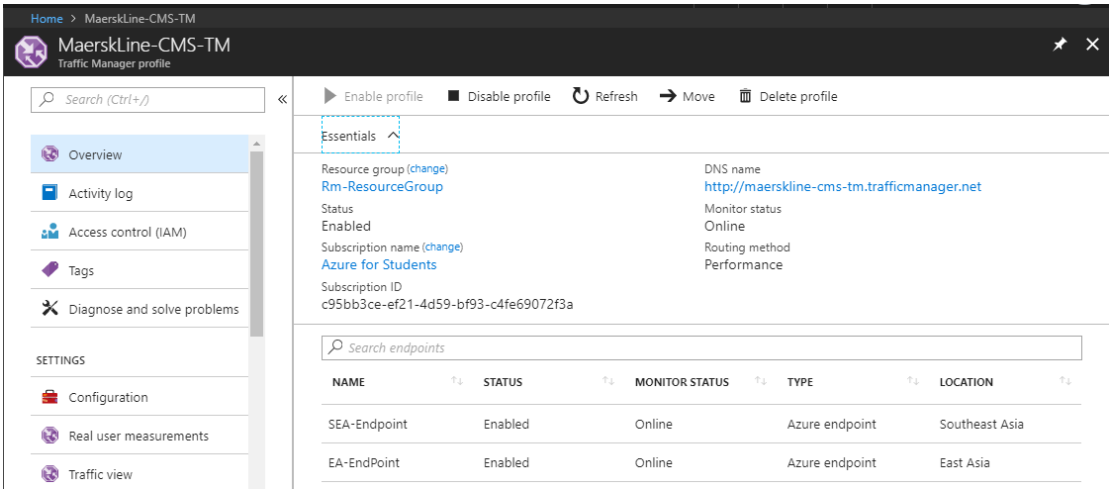

Figure 33: Traffic Manager profile overview

Furthermore, since the pricing tier selected includes auto scaling feature, it is implemented for the app services as well for better performance and cost effective. Auto scaling is the process of dynamically adjusting the resources needed for the app service to maintain the desired performance level and satisfy the service-level agreements (SLAs) (Microsoft Azure, 2017). The auto scaling method implemented in this system is horizontal scaling, also known as scaling in and out. This scaling method focus on adding or removing resource instances based on the determined rules. In this case, the auto scaling configuration is shown in the figure below. When the average CPU usage percentage is greater than 70%, a resource instance will be added, while the average CPU usage percentage is lesser than 20%, a resource instance will be removed. The amount of instance used will be 1 with the limit of 10 instance max. With the mentioned configuration, the system is ensured to maintain the desired performance level while saving the cost when unnecessary resource instances are deallocated.

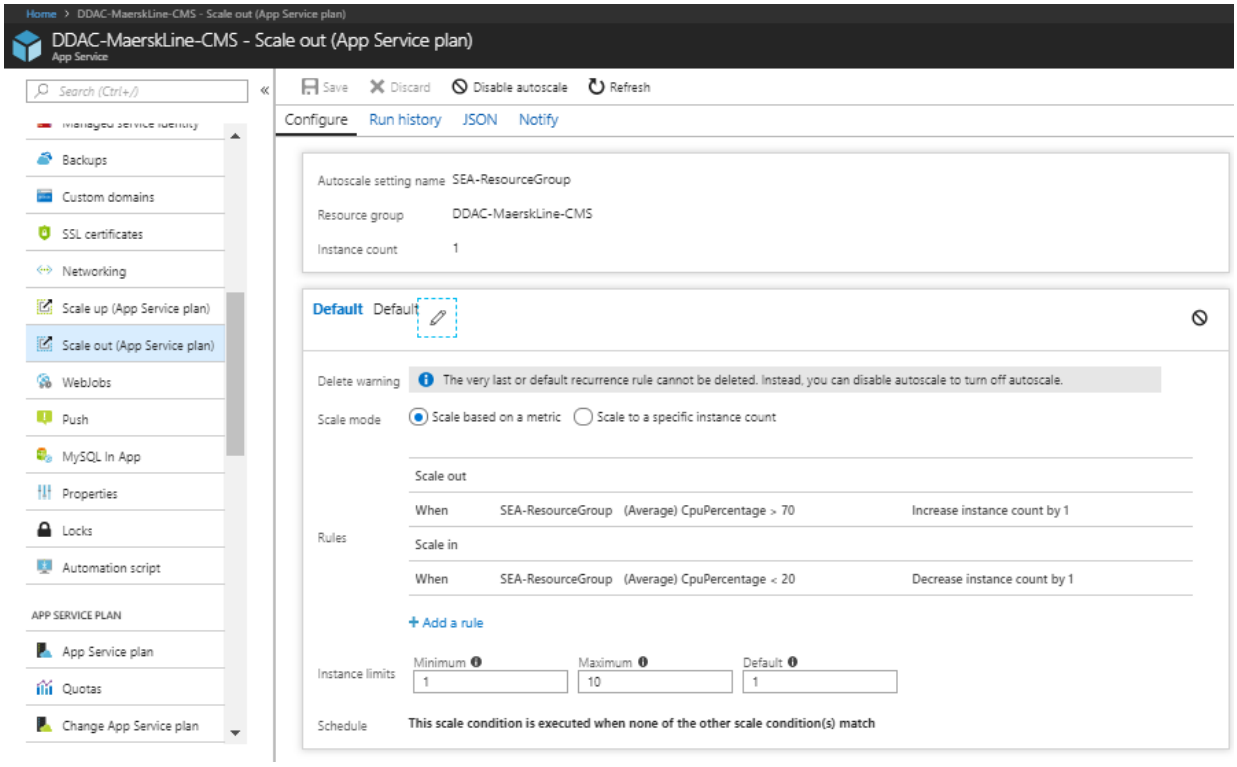

Figure 34: App service auto scaling configuration

Moreover, Azure Database for MySQL Server is used instead of SQL Database to enhance the availability, reliability, and performance. Azure Database for MySQL Server is built by Microsoft Azure to deliver high availability without additional cost, replication, and configuration (Microsoft Azure, 2018). It is also connected with the Microsoft global network of datacenters with unparalleled security and round-to-clock monitoring which results in high availability and reliability (Microsoft Azure, 2018). The reason Azure Database for MySQL Server can provide better performance is that it can flexibly scale the storage or compute within seconds and provisioned within minutes (Microsoft Azure , 2018). The figure below shows the Azure Database for MySQL Server with the databases associated.

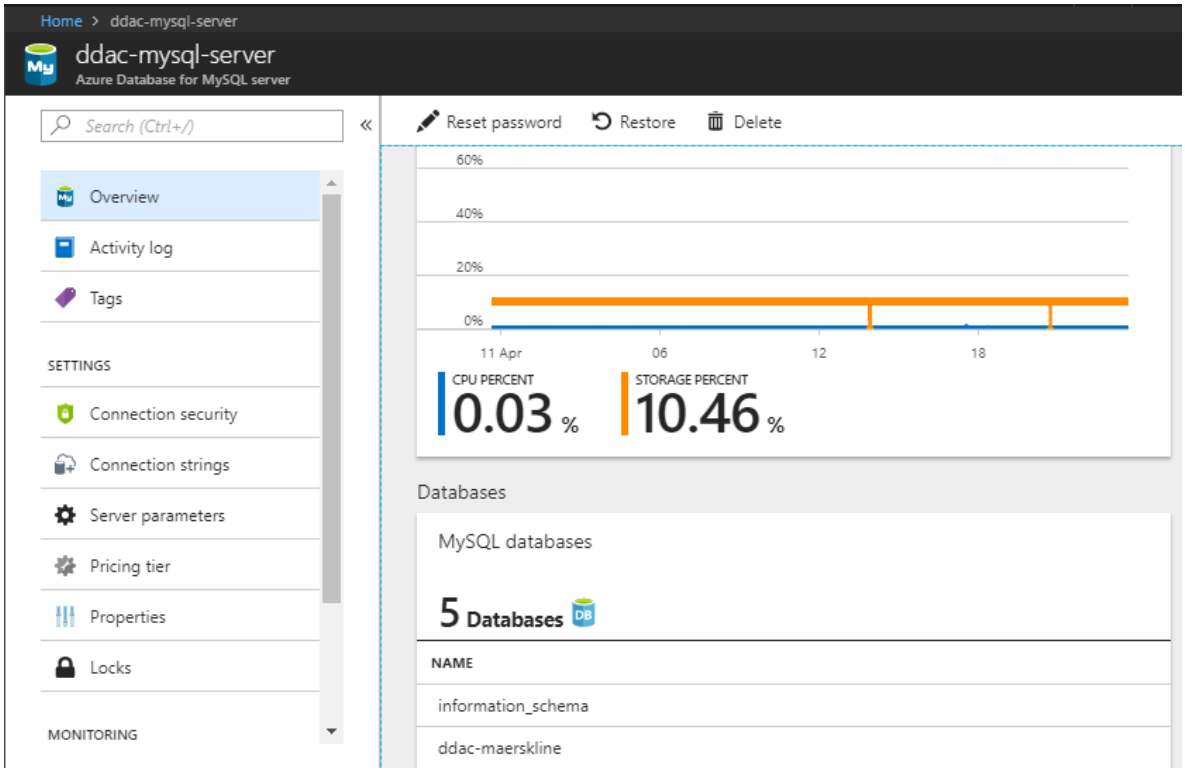

Figure 35: Azure Database for MySQL Server Overview

5.0 Test Plan and Testing Discussion

5.1 Unit Testing

As mentioned above, Unit Testing is performed to ensure that all the functionalities of the system are working as required. For better documentation and traceability, a test plan is constructed to facilitate the unit testing process. The test plan includes all the possibilities of the functions to be tested and it ensures that the testers will not missed out any of the test cases. The steps to perform a task and its expected result will be provided in the test plan and the tester will need to follow it and record the actual result. Below shows the Unit Testing test plan constructed for this system.

5.1.1 Login

No	Test Case	Expected Result	Actual Result
1	Click Login button when username, password and user type are left empty.	Display error message “Please choose a user type and enter your username and password.”	As Expected
2	Login the system using invalid credentials.	Display error message “Invalid Credentials”.	As Expected
3	Login the system using valid credentials	Redirect user to Admin Home Page if the selected user type is Admin and redirect user to Agent Home Page is the selected user type is Agent.	As Expected

Table 12: Login Test Plan

5.1.2 Register Agent

No	Test Case	Expected Result	Actual Result
1	Click Register button when username, password and name are left empty.	Display error message “Please fill in the agent’s username, password and name.”	As Expected
2	Register Agent using invalid values.	Display error message indicating the value that violates the validation rule and what is the validation rule.	As Expected
3	Register Agent using valid values.	Display register successful message and the Agent account is inserted to the database.	As Expected

Table 13: Register Agent Test Plan

5.1.3 Register Ship

No	Test Case	Expected Result	Actual Result
1	Click Register button when name, IMO, year built, and capacity are left empty.	Display error message “Please fill in the name, IMO, year built, and capacity.”	As Expected
2	Register Ship using invalid values.	Display error message indicating the value that violates the validation rule and what is the validation rule.	As Expected
3	Register Ship using valid values.	Display register successful message and the Ship details are inserted to the database.	As Expected

Table 14: Register Ship Test Plan

5.1.4 View Schedule List

No	Test Case	Expected Result	Actual Result
1	Click Manage Schedule and Booking on the navigation bar and select Schedule List tab.	Display a list of schedules with all the details (SID, departure date, arrival date, shipID, ship name, ship capacity, available weight, from, to, and status) correctly.	As Expected
2	Enter values into the filter input texts	The list of schedules is filtered based on the values entered.	As Expected

Table 15: View Schedule List Test Plan

5.1.5 Register New Schedule

No	Test Case	Expected Result	Actual Result
1	Click Register when no ship is selected and the departure date, arrival date, from, and to are left empty.	Display error message “Please select a ship and fill in the departure date, arrival date, from, and to.”	As Expected
2	Register New Schedule using arrival date that is earlier than the departure date.	Display error message “Invalid schedule date selected !”.	As Expected
3	Register New Schedule using valid values.	Display register successful message and the schedule details are inserted to the database.	As Expected

Table 16: Register New Schedule Test Plan

5.1.6 Manage Schedule Booking Request

No	Test Case	Expected Result	Actual Result
1	Select a schedule booking request and select Approve, then select confirm.	The schedule booking request is removed from the list and the booking status is changed to "Approved". The available weight for the schedule and the item status are updated as well. Upon login of the Agent who requested this booking, a notification will be displayed to indicate a booking request is approved.	As Expected
2	Select a schedule booking request, select Reject, enter the reason for rejecting this booking, then select confirm.	The schedule booking request is removed from the list and the booking status is changed to "Rejected". The reason for rejecting this booking is recorded into the database as well. Upon login of the Agent who requested this booking, a notification will be displayed to indicate a booking request is rejected.	As Expected

Table 17: Manage Schedule Booking Request Test Plan

5.1.7 View Schedule

No	Test Case	Expected Result	Actual Result
1	Click Book Schedule on the navigation bar and select Available Schedule List tab.	Display a list of available schedules with all the details (SID, departure date, arrival date, shipID, ship name, ship capacity, available weight, from, and to) correctly.	As Expected
2	Enter values into the filter input texts	The list of available schedules is filtered based on the values entered.	As Expected

Table 18: View Schedule Test Plan

5.1.8 View Booking Done

No	Test Case	Expected Result	Actual Result
1	Click Book Schedule on the navigation bar and select Booking List tab.	Display a list of booking done by the logged in Agent with all the details (BID, CustID, ItemID, item name, weight, SID, and Status) correctly.	As Expected
2	Enter values into the filter input texts	The list of booking is filtered based on the values entered.	As Expected
3	Click on the toggler button of the rejected booking.	The reason for rejecting the booking is displayed along with a "Got It" button.	As Expected
4	Click on the "Got It" button after clicking on the rejected booking toggler button.	The booking record is removed from the list.	As Expected

Table 19: View Booking Done Test Plan

5.1.9 Register Customer

No	Test Case	Expected Result	Actual Result
1	Click Register when name, gender, e-mail, and address are left empty.	Display error message "Please fill in the name, e-mail, address, and select a gender."	As Expected
2	Register Customer using invalid inputs.	Display error message indicating the value that violates the validation rule and what is the validation rule.	As Expected
3	Register Customer using valid values.	Display register successful message and the customer details are inserted to the database.	As Expected

Table 20: Register Customer Test Plan

5.1.10 Register Item

No	Test Case	Expected Result	Actual Result
1	Click Register before searching for a valid customer and the item name and item weight are left empty.	Display error message "Please search for a customer and fill in the item name and item weight."	As Expected
2	Register Item using invalid inputs.	Display error message indicating the value that violates the validation rule and what is the validation rule.	As Expected

3	Register Item using valid values.	Display register successful message and the item details are inserted to the database.	As Expected

Table 21: Register Item Test Plan

5.1.11 Book Schedule

No	Test Case	Expected Result	Actual Result
1	Click Book Schedule before selecting a schedule and item.	Display error message “Please select a customer and Please select an item.”	As Expected
2	Book Schedule using item that weight greater than the schedule’s available weight.	Display error message “Invalid booking request.”.	As Expected
3	Book Schedule with item weight lesser than or equal to the schedule’s available weight.	Display book schedule successful message and the booking details are inserted to the database. Upon Admin login, a notification will be displayed indicating there is a pending schedule booking request to be managed.	As Expected

5.2 Performance Testing

By hosting the system on Azure, performance testing can be done within a few clicks with various configurations that facilitates the analyzing processes. In this performance testing, the system will be using three different plans within the Standard pricing tier for comparison purposes and performance testing will be conducted within 2 minutes with 250, 500, 750, and 1000 user load. Figure on the right shows the configuration for one of the performance testing done. The performance testing will take up to several minutes to run and generate results. The performance testing results includes the number of successful and failed requests, performance under load, and web app usage. Below shows the result from one of the performance testing.

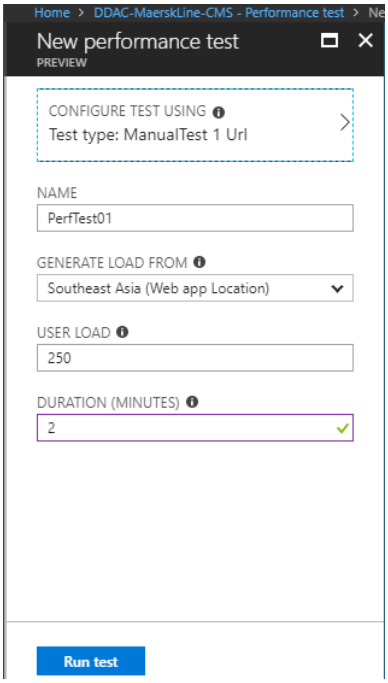

Figure 36: Performance testing configuration

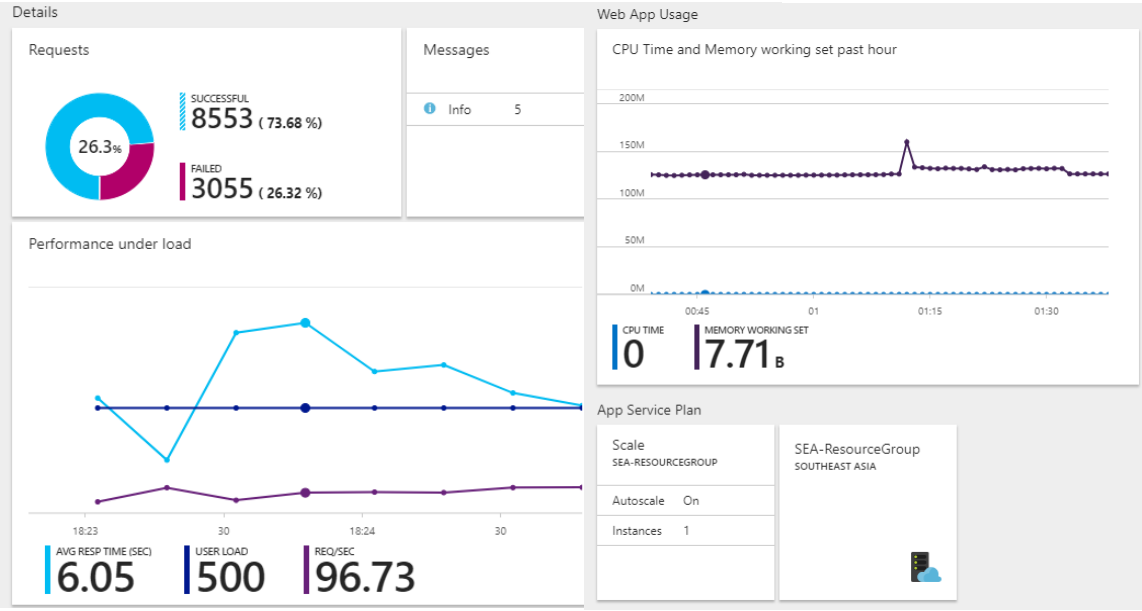

Figure 37: Performance testing result

Obviously, different plan within the Standard pricing tier provides different amount of core and RAM. S1 Standard provides 1 Core and 1.75 GB of RAM, while S2 Standard provides double of the Core and RAM provided by S1 Standard and S3 Standard provides double of the Core and RAM provided by S2 Standard. Next page shows the comparison table of performance testing done while using different plan within the Standard pricing tier with auto scaling configured.

App Service Plan	Concurrent Users / Aspect of Comparison	250	500	750	1000
S1	Percentage of failed request	0.03%	0%	0%	0.14%
	Average response time	11.84 sec	19.56 sec	21.13 sec	15.15 sec
S2	Percentage of failed request	0%	0.02%	0.34%	0.59%
	Average response time	3.49 sec	2.06 sec	3.82 sec	3.80 sec
S3	Percentage of failed request	0%	0%	0.05%	0%
	Average response time	1.81 sec	1.86 sec	1.41 sec	2.16 sec

Table 22: Performance testing result comparison

As shown in the table above, the better the app service plan, the better the system performance and the ability to handle concurrent requests. While S1 app service plan performs with average response time ranging from 11.84 seconds to 21.13 seconds, S2 app service plan greatly outperformed S1 app service plan with average response time ranging from 2.06 seconds to 3.82 seconds. However, there are not much improvements when the system is scaled up to S3 service plan which performs with average time ranging from 1.41 seconds to 2.16 seconds. Considering that the pricing for S3 app service plan is double of S2 and S2 is double of S1, it can be concluded that S2 app service plan will be the ideal app service plan for this system. This is because S2 app service plan has a significant improvement compared to S1 app service plan which made the doubled price worth it, but the same case does not apply on the S3 app service plan.

6.0 Implementation and Discussion on Managed Database (PaaS)

PaaS, also known as Platform as a Service, is a rich environment established in the cloud by cloud service provider that allow users to publish their applications ranging from simple cloud-based apps to cloud-enabled enterprise apps (Microsoft Azure, 2014). Every cloud service provider provides a wide variety of services with different characteristics which allows their subscribers to purchase the desired resources and services they need for their published system over a secure internet connection. PaaS includes infrastructures such as development tools, OS, servers and storage, networking firewall, and data center physical plant which designed to support the entire web application lifecycle. When talking about cloud service, Amazon.com will not be absent in the discussion as it has dominated the public cloud market as well as the global PaaS market according to Global Industry Analysts, Inc and Gartner, a global industry analyst firm (Global Industry Analysts, Inc, 2015; Cameron, 2017).

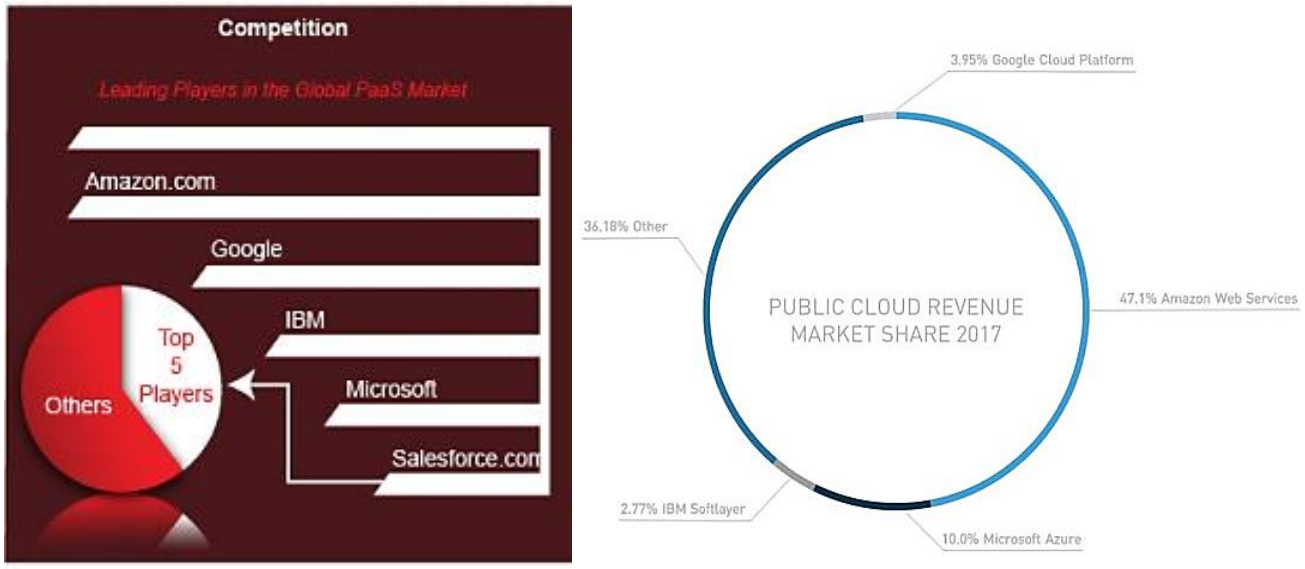

Figure 38: Global cloud service and PaaS market share (Cameron, 2017; Global Industry Analysts, Inc, 2015)

As PaaS typically includes a wide variety of services and resources, it is used by the organizations for different purposes instead of just publishing applications. Publishing application aside, PaaS is also used in development phase. In fact, PaaS can be used for the entire cloud-based application development phase as most of the PaaS provides development framework that allow developers to build upon using built-in software components and reduce the amount of coding needed (Microsoft Azure, 2014). Besides, business intelligence services are included in the PaaS services which allows the organizations to analyze their applications and data for further improvement or decision making (Microsoft Azure, 2014).

As discussed in the previous sections, the Maersk Line Container Management System is developed based on PaaS provided by Microsoft Azure. Therefore, the following will discuss the advantage of using PaaS over self-deployed local server which may not be constructed with necessary expertise. First of all, the tools and service provided by PaaS increased the capability of the development team without extra cost for recruiting new members with specific expertise. For example, the development team for this system that usually only focus on system development can now take over the deployment and hosting task within a few clicks in Microsoft Azure.

Moreover, PaaS made the cross-platform and multiple platform applications easier and faster to develop. This is because most of the cloud service providers provides the development options for multiple or cross-platform applications which includes the deployment process. For example, Microsoft Azure provides resources options and tutorials as shown in the figure below which facilitates the development and deployment process from the very beginning of the development phase.

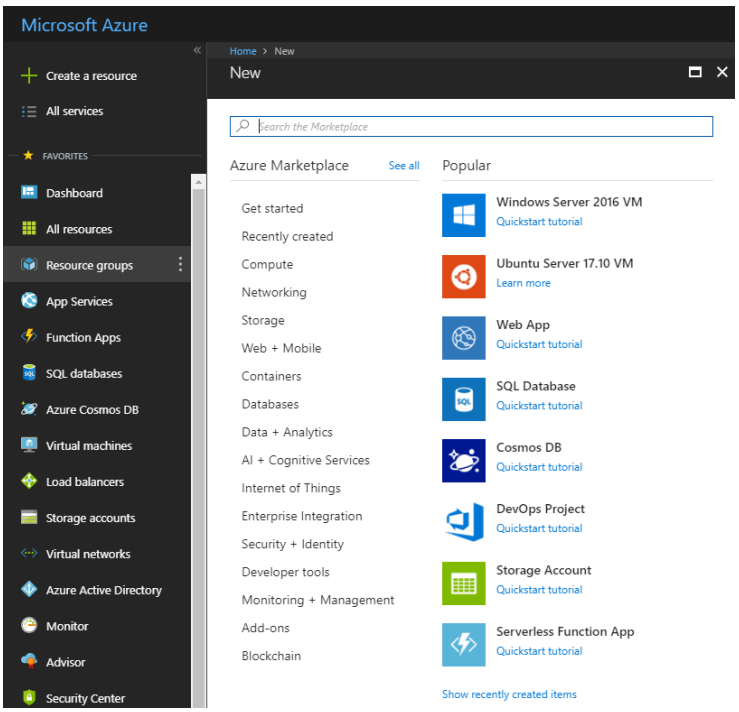

Figure 39: Microsoft Azure resources

Furthermore, PaaS allow the developers to utilize the advanced tools and services provided affordably on a pay-as-you-go basis. The advanced tools and services includes development software, business intelligence and analytic tools which usually bundled in a big and expensive package in the market. It is common that in some cases, the developer only

required one of the tools included in the big and expensive package. Therefore, PaaS is used as the tools and services are not bundled and it allow the developers to only pay when they are using the tools and services. This saved the expenses and complexity of buying and managing different applications and software licenses.

Other than that, PaaS is a development platform accessed over the internet worldwide. Which means the development team can work together on a project even they are separated regionally. All in all, by using the PaaS, the development team only need to concern with the application development and the configuration of the tools and services adopted from the cloud service, while the cloud service provider will manage everything else.

7.0 Conclusion

In a nutshell, the development and deployment of Maersk Line Container Management System are completed successfully with all the requirements met. Throughout this project, the developer has explored the Cloud Computing knowledge region, which is entirely new for the developer. From only concerning with application development, the developer has widened his capability to deploy and host the application on cloud services. The developer has learned the basic concepts of cloud computing and practiced with actual implementation on Microsoft Azure. Other than learning and practicing the theoretical and practical operations of hosting, the developer has utilized the tools and services provided by Microsoft Azure to ease the development, deployment, and testing processes. All in all, this project exposed the developer to the cloud computing knowledge base which prepares the developer for future career as cloud computing has became commonly used in the industry nowadays.

8.0 References

1. Azharuddin, K., 2012. *The Windows Club.* [Online]
   Available at: http://www.thewindowsclub.com/difference-eclipse-netbeans
   [Accessed 8 April 2018].

2. Cameron, C., 2017. *Overview of Cloud Market in 2017 and Beyond.* [Online]
   Available at: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/
   [Accessed 12 April 2018].

3. Global Industry Analysts, Inc, 2015. *Platform as a Service (PaaS) Market Trends.* [Online]
   Available at:
   http://www.strategyr.com/MarketResearch/Platform_as_a_Service_PaaS_Market_Trends.asp
   [Accessed 12 April 2018].

4. Microsoft Azure , 2018. *Azure Database for MySQL.* [Online]
   Available at: https://azure.microsoft.com/en-us/services/mysql/
   [Accessed 11 April 2018].

5. Microsoft Azure, 2014. *What is PaaS?.* [Online]
   Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/
   [Accessed 12 April 2018].

6. Microsoft Azure, 2016. *Set up staging environment for App Service.* [Online]
   Available at: https://docs.microsoft.com/en-us/azure/app-service/web-sites-staged-publishing
   [Accessed 11 April 2018].

7. Microsoft Azure, 2017. *Autoscaling.* [Online]
   Available at: https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling
   [Accessed 11 April 2018].

8. Microsoft Azure, 2017. *Azure App Service Plan Overview.* [Online]
   Available at: https://docs.microsoft.com/en-us/azure/app-service/azure-web-sites-web-hosting-plans-in-depth-overview
   [Accessed 11 April 2018].

9. Microsoft Azure, 2018. *Azure Database for MySQL.* [Online]
   Available at: https://azure.microsoft.com/en-us/services/mysql/
   [Accessed 11 April 2018].

10. Microsoft Azure, 2018. *Azure Database for MySQL pricing tiers.* [Online]
Available at: https://docs.microsoft.com/en-gb/azure/mysql/concepts-pricing-tiers#compute-generations-vcores-and-memory
[Accessed 11 April 2018].

11. Netbeans, 2018. *JSF 2.X Support in Netbeans IDE.* [Online]
Available at: https://netbeans.org/kb/docs/web/jsf20-support.html
[Accessed 8 April 2018].

12. Oracle, 2018. *MySQL.* [Online]
Available at: http://www.oracle.com/technetwork/database/mysql/index.html
[Accessed 8 April 2018].

13. Qusay, H. M., 2004. *Developing Web Application with JavaServer Faces.* [Online]
Available at: http://www.oracle.com/technetwork/articles/java/javaserverfaces-135231.html
[Accessed 8 April 2018].

14. Tutorials Point, 2013. *JSF Architecture.* [Online]
Available at: https://www.tutorialspoint.com/jsf/jsf_architecture.htm
[Accessed 8 April 2018].