INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

DESIGNING AND DEVELOPING APPLICATION ON THE CLOUD

CT071-3-3-DDAC

Name	:	NG CHUN YIEN
TP Number	:	TP034900
Intake Code	:	UC3F1706SE(T3)
Lecturer's Name	:	DR KALAI ANAND A/L RATMAN
Hand In Date	:	13 APRIL 2018

Acknowledgement

First of all, I would like to thank the lecturer, Dr Kalai who have provided valuable guidance and knowledge in developing application on the cloud. Besides, I would like to thank my parents on continuous support by providing funding and moral support. Without their support and funding, the project will not be able to be completed on time. Lastly, I would like to express my appreciation to my course mates who have share various knowledge when I faced any difficulties in completing this project. Their feedback and guidance are highly appreciated.

Table of Contents

1 Introduction

1.1 Project Background

Maersk Line is the global container division and the largest operating unit of A.P. Moller – Maersk Group, a Danish business conglomerate and it is the world's largest container shipping company having customers through 374 offices in 116 countries. As to operate more than 600 vessels and the capacity of 2.6 million TEU, it employs approximately 7,000 sea farers and 25,000 land-based people. It was founded in 1928.

Although Maersk Line is already handling all the cargo it can manage, the company found out that the volume of most of the goods it was shipping had grown to full capacity. As to rectifying the situation, the company decided to implement cloud powered solutions.

Soeren Lorenzen, an account general manager with Hewlett-Packard company said, 'How can we support the overall business strategy, and also from an IT perspective," while involving in first-hand with Maersk's ITO efforts. There was a new CIO who wanted to outsource every part of IT, but without [negatively] impacting shipping."

As to support further business growth and increase organizational flexibility, Maersk decided to consolidate all of its data centers and server rooms operating worldwide on to 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 Objective

The objective of this project is to design and develop a Container Management System (CMS) to cater to manage the containers, reduces overall supply chain costs and an efficient way to manage logistics.

1.3 Scope

Table 1: Scope of the project

Provisioning	Provide a web application that suitable for Microsoft Azure Platform
Monitoring	Identify errors and troubleshoot them by monitoring the web application.
Availability	Ensure the web application is available most of the time without affected by any bugs or errors
Scalability	Provide a scalable web application that satisfy the need of application

1.4 Requirement Specification

1. From import, export and transshipment processing to gate operations.
2. To be able to scale the solution to meet the needs of demands during peak seasons.
3. Improves profitability, reduce costs, increases productivity, eradicates errors and optimizes resources to future-proof your cargo handling business for high performance.
4. Assurance & reliability through Failover Management.
5. 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.
6. Manage your entire booking process from schedule search to booking confirmation.

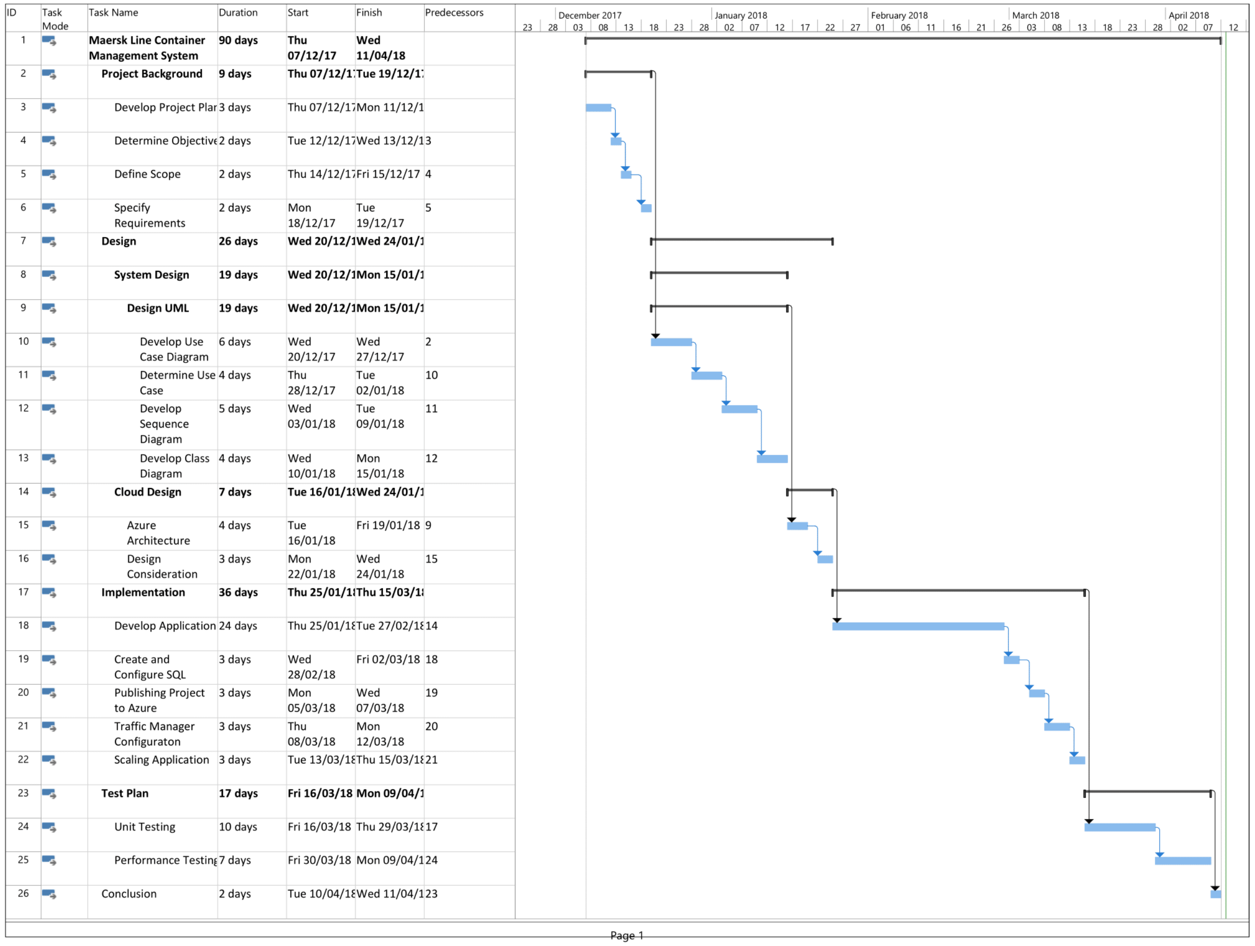
1.5 Summary of Major functions

There major function of Maersk Line Container Management System (CMS) are as follows:

Table 2: Summary of major functions

Admin	Agent
- View Booking - Add Agent - View Agent - Add Ship - Add Route - Create Schedule - Top up Credit - Edit Agent Profile	- Register Customer - Register Item - Book Vessel - View Booking

2 Project Plan



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**Maersk Line Container Management System**	**90 days**	**Thu 07/12/17**	**Wed 11/04/18**	
2		**Project Background**	**9 days**	**Thu 07/12/1**	**Tue 19/12/1**	
3		Develop Project Plar	3 days	Thu 07/12/17	Mon 11/12/1	
4		Determine Objective	2 days	Tue 12/12/17	Wed 13/12/1	3
5		Define Scope	2 days	Thu 14/12/17	Fri 15/12/17	4
6		Specify Requirements	2 days	Mon 18/12/17	Tue 19/12/17	5
7		**Design**	**26 days**	**Wed 20/12/1**	**Wed 24/01/1**	
8		**System Design**	**19 days**	**Wed 20/12/1**	**Mon 15/01/1**	
9		**Design UML**	**19 days**	**Wed 20/12/1**	**Mon 15/01/1**	
10		Develop Use Case Diagram	6 days	Wed 20/12/17	Wed 27/12/17	2
11		Determine Use Case	4 days	Thu 28/12/17	Tue 02/01/18	10
12		Develop Sequence Diagram	5 days	Wed 03/01/18	Tue 09/01/18	11
13		Develop Class Diagram	4 days	Wed 10/01/18	Mon 15/01/18	12
14		**Cloud Design**	**7 days**	**Tue 16/01/1**	**Wed 24/01/1**	
15		Azure Architecture	4 days	Tue 16/01/18	Fri 19/01/18	9
16		Design Consideration	3 days	Mon 22/01/18	Wed 24/01/18	15
17		**Implementation**	**36 days**	**Thu 25/01/1**	**Thu 15/03/1**	
18		Develop Application	24 days	Thu 25/01/1	Tue 27/02/1	14
19		Create and Configure SQL	3 days	Wed 28/02/18	Fri 02/03/18	18
20		Publishing Project to Azure	3 days	Mon 05/03/18	Wed 07/03/18	19
21		Traffic Manager Configuraton	3 days	Thu 08/03/18	Mon 12/03/18	20
22		Scaling Application	3 days	Tue 13/03/1	Thu 15/03/1	21
23		**Test Plan**	**17 days**	**Fri 16/03/18**	**Mon 09/04/1**	
24		Unit Testing	10 days	Fri 16/03/18	Thu 29/03/1	17
25		Performance Testing	7 days	Fri 30/03/18	Mon 09/04/1	24
26		Conclusion	2 days	Tue 10/04/1	Wed 11/04/1	23

Figure 1: Gantt Chart

3 Design

3.1 Design Consideration

There are several considerations and assumptions before designing the application as stated below:

3.1.1 Availability Consideration

Availability is the ability of a system to survive when one of its components went failure. It may get affected by system errors, malicious attacks or others. The redundancy of key components is one of the criteria that shall be considered as to ensure quick recovery of system failures and interruptions when ensuring constant availability. (Serhiy, 2016) According to Azure, their service level agreement (SLA) stated that the app service provided is up to 99.95% of success rate and 99.99% of success rate for its SQL Database. Therefore, by using azure as the cloud platform, it can ensure that the application is available most of the time.

3.1.2 Scalability Consideration

According to Microsoft (2017), scalability is the ability of a system to expand to meet the business needs. Azure app service which provide scale up and scale out for the system administrator to handle its web application. During peak period, the system administration can scale up the application by selecting a bigger plan in which large memory size, more of number of cores and others. It can also be scale out during non-peak period as to minimise the operating cost. Hence, it is flexible for the system administrator whether to scale up or scale out the application to meet the business needs.

3.1.3 Manageability Consideration

Manageability is defined as the easiness of the operating, maintaining and updating application by the system administrator. A manageable application shall allow easy diagnosing of the problems that occurred. Hence, Azure which provide monitoring and diagnostic tool allow the system administrator to identify the problem arose from the application.

3.2 Modelling

3.2.1 Use Case Diagram

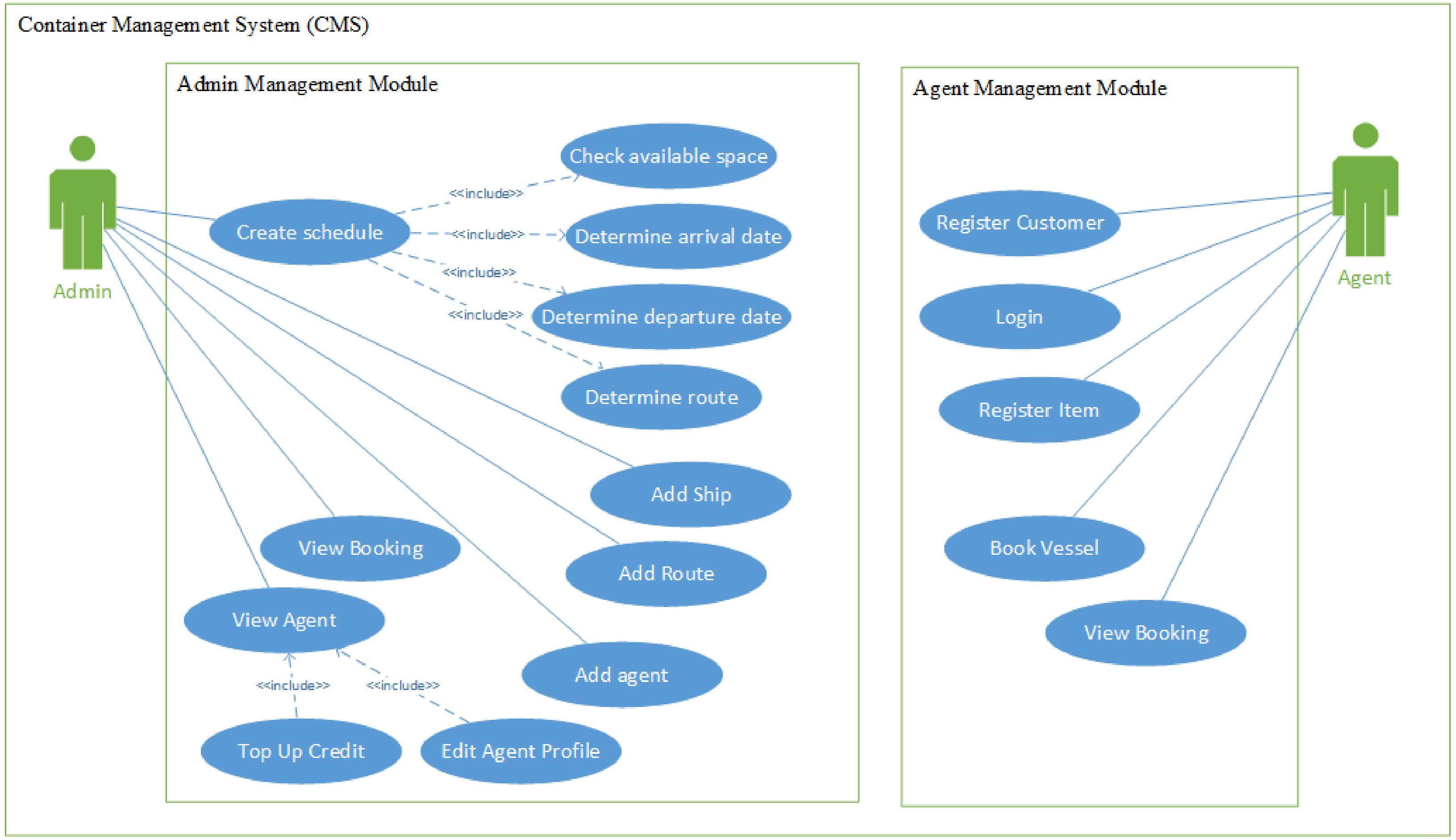


Figure 2: Use Case Diagram

3.2.2 Use Case Description

3.2.2.1 Add Agent

Use Case: Add Agent	
Summary:	Add an agent account which allow accessing to the website
Dependency:	-
Actors:	Admin
Pre-condition:	Admin wished to register a new agent onto the website.
Main flow:	1. Admin logged in to the application. 2. Admin selects ‘Add Agent’ through navigation bar. 3. The system prompts the user to Add Agent page. 4. Admin inputs all the required fields and click on ‘Add’ button. 5. The system shows “Agent added successfully” message.
Alternative flow:	4(a) If the username has been registered, error message will be showed. 4(b) If password and confirm password are not same, error message will be showed. 4(c) If the username or password contains “admin”, error message will be showed.
Post-condition:	Agent is registered successfully and saved into database. The agent can now log in with the username and password.

3.2.2.2 Add Route

Use Case: Add Route	
Summary:	Admin can add the route by inputting departure and arrival destination for the vessel.
Dependency:	-
Actors:	Admin
Pre-condition:	Admin want to add new route for the ship while creating schedule.
Main flow:	1. Admin selects ‘Add Route’ through navigation bar. 2. The system prompts the user to Add Route page. 3. Admin inputs all the required fields and click on ‘Add’ button. 4. The system shows “Route added successfully” message.
Alternative flow:	3(a) If the departure and arrival destination has been registered in the database, error message will be showed.

Post-condition:	Route is now added successfully and schedule can be created on the new route.

3.2.2.3 Add Ship

Use Case: Add Ship	
Summary:	Admin can add new ship for more selection of ship.
Dependency:	-
Actors:	Admin
Pre-condition:	There is new ship that has not been added into the system.
Main flow:	1. Admin selects ‘Add Ship’ through navigation bar. 2. The system prompts the user to Add Ship page. 3. Admin inputs all the required fields and click on ‘Add’ button. 4. The system shows “Ship added successfully” message.
Alternative flow:	-
Post-condition:	Ship is added successfully and saved into database in which the ship is now available to be selected when creating schedule.

3.2.2.4 View Booking

Use Case: View Booking	
Summary:	User can view the booking made on that day.
Dependency:	-
Actors:	Admin, Agent
Pre-condition:	User wishes to view today’s booking.
Main flow:	1. User logged in to the system. 2. The system retrieves booking details from database. 3. The system navigates to Portal page and all the item and booking regarding that day will be showed. 4. Admin select ‘View Booking’ through navigation bar. 5. The system retrieves all the past booking and display in ‘View Booking’ page. 6. Admin can filter the booking by month by choosing the month desired. 7. Bookings that made on that specific month will be showed.
Alternative flow:	3(a) If no booking made on that day, no data will be showed.

Post-condition:	User can identify the number of booking made throughout that period.

3.2.2.5 View Agent

Use Case: View Agent	
Summary:	Admin can view all the agents that registered onto the web application
Dependency:	-
Actors:	Admin
Pre-condition:	Admin wishes to know the details of agents that uses the web application.
Main flow:	1. Admin selects 'View Agent' through navigation bar. 2. The system retrieves all the agent's details from the database. 3. The system prompts the user to View Agent page and all the agent details are showed in table.
Alternative flow:	3(a) If there is no agent in the database, no data will be showed.
Post-condition:	Admin can search the agent profile by inputting the name of the agent.

3.2.2.6 Top Up Credit

Use Case: Top Up Credit	
Summary:	Admin can top up credit onto agent's account for booking the vessel.
Dependency:	<<include>> View Agent
Actors:	Admin
Pre-condition:	Agent wishes to top up credit by informing the agent.
Main flow:	1. Admin inputting the agent name in "View Agent" page. 2. The system retrieves the respective agent from the database and prompts the admin to "Agent Profile" page. 3. Admin can fill in the Top Up Credit text field and click on "Top Up" button. 4. The system shows "Credit top up successfully" message.

Alternative flow:	1(a) If the agent name inputted is not available, error message will be showed.
Post-condition:	Credit was added into agent’s account and allow for vessel booking.

3.2.2.7 Edit Agent Profile

Use Case: Edit Agent Profile	
Summary:	Admin can update the agent profile by inputting the information
Dependency:	<<include>> View Agent
Actors:	Admin
Pre-condition:	Admin wishes to make some changes onto the agent’s profile
Main flow:	1. Admin inputting the agent name in “View Agent” page. 2. The system retrieves the respective agent from the database and prompts the admin to “Agent Profile” page. 3. Admin can edit the desired text field and click on “Update Profile” button. 4. The system shows “Profile updated successfully” message.
Alternative flow:	1(a) If the agent name inputted is not available, error message will be showed. 3(a) If the password and confirm password are not same, error message will be showed.
Post-condition:	Agent’s profile is being updated in the database.

3.2.2.8 Create Schedule

Use Case: Create Schedule	
Summary:	Admin adds a new schedule by choosing ship, route, departure and arrival date.
Dependency:	<<include>> Determine Route, Check Available Space, Determine Arrival Date, Determine Departure Date
Actors:	Admin
Pre-condition:	Admin wants to add new schedule which allowed to be chosen for vessel booking.
Main flow:	1. Admin selects ‘Create Schedule through navigation bar. 2. The system retrieves all the available ship and route from the database.

	3. The system prompts the user to Create Schedule page. 4. Admin select the desired ship, route and determine the departure and arrival date. 5. The system shows “Created schedule successfully” message.
Alternative flow:	4(a) If no date was chosen, error message will be showed. 4(b) If the departure date chosen is before the current date, error message will be showed. 4(c) If the departure and arrival date chosen for that ship is occupied, error message will be showed.
Post-condition:	The schedule is now ready to be booked by the agent for vessel booking.

3.2.2.9 Register Customer

Use Case: Register Customer	
Summary:	Agent can register the details of new customers for shipping process.
Dependency:	-
Actors:	Agent
Pre-condition:	There is new customer that wishes to ship his item.
Main flow:	1. Agent selects ‘Register Customer’ through navigation bar. 2. The system prompts the user to Register Customer page. 3. Admin inputs all the required fields and click on ‘Add’ button. 4. The system shows “Customer registered successfully” message.
Alternative flow:	3(a) If the name of customer has been registered, error message will be showed.
Post-condition:	The customer is now registered in database and allow for adding item.

3.2.2.10 Login

Use Case: Login	
Summary:	Agent logged in to the system for business operation.
Dependency:	-
Actors:	Agent

Pre-condition:	Agent wishes to perform business operation.
Main flow:	1. Agent select “Login” through navigation bar. 2. Agent inputs the username and password. 3. The system validates the username and password. 4. The system prompts agent to portal page.
Alternative flow:	3(a) If the username or password is wrong, error message will be showed.
Post-condition:	Agent is now able to perform business operation such as booking vessel, register customer and register item for customer.

3.2.2.11 Register Item

Use Case: Register Item	
Summary:	Agent register the item onto the customer.
Dependency:	-
Actors:	Agent
Pre-condition:	Customer wishes to ship his item.
Main flow:	1. Agent selects ‘Register Item’ through navigation bar. 2. The system retrieves all the customers available from the database. 3. The system prompts the user to Add an Item page. 4. Admin chooses the customer from the dropdown list and inputs all the required fields and click on ‘Add’ button. 5. The system shows “Item registered successfully” message.
Alternative flow:	2(a) If there is no customer in the database, no data will be showed in dropdown list.
Post-condition:	Item was registered successfully for the customer and vessel can be booked now.

3.2.2.12 Book Vessel

Use Case: Book Vessel	
Summary:	Agent book the vessel by specifying the route, departure date, customer and items that wished to be shipped.
Dependency:	-
Actors:	Agent

Pre-condition:	Item was registered for the customer.
Main flow:	1. Agent selects 'Book Vessel' through navigation bar. 2. The system retrieves all the schedule from the database. 3. The system prompts the user to Book Vessel page. 4. Admin chooses the route, desired departure date, customer from the dropdown list and select the items that wanted to be shipped and click on 'Add' button. 5. The system shows "Booked vessel successfully" message.
Alternative flow:	4(a) If no schedule available for that route and item is not available for the customer, Add button will be disabled. 4(b) If no item is selected, error message will be showed. 4(c) If the price exceeds the agent's credit, error message will be showed. 4(d) If the bay of ship chosen is full, error message will be showed.
Post-condition:	Vessel is now booked successfully and items will be shipped to the desired destination.

3.2.3 Sequence Diagram

3.2.3.1 Add Agent

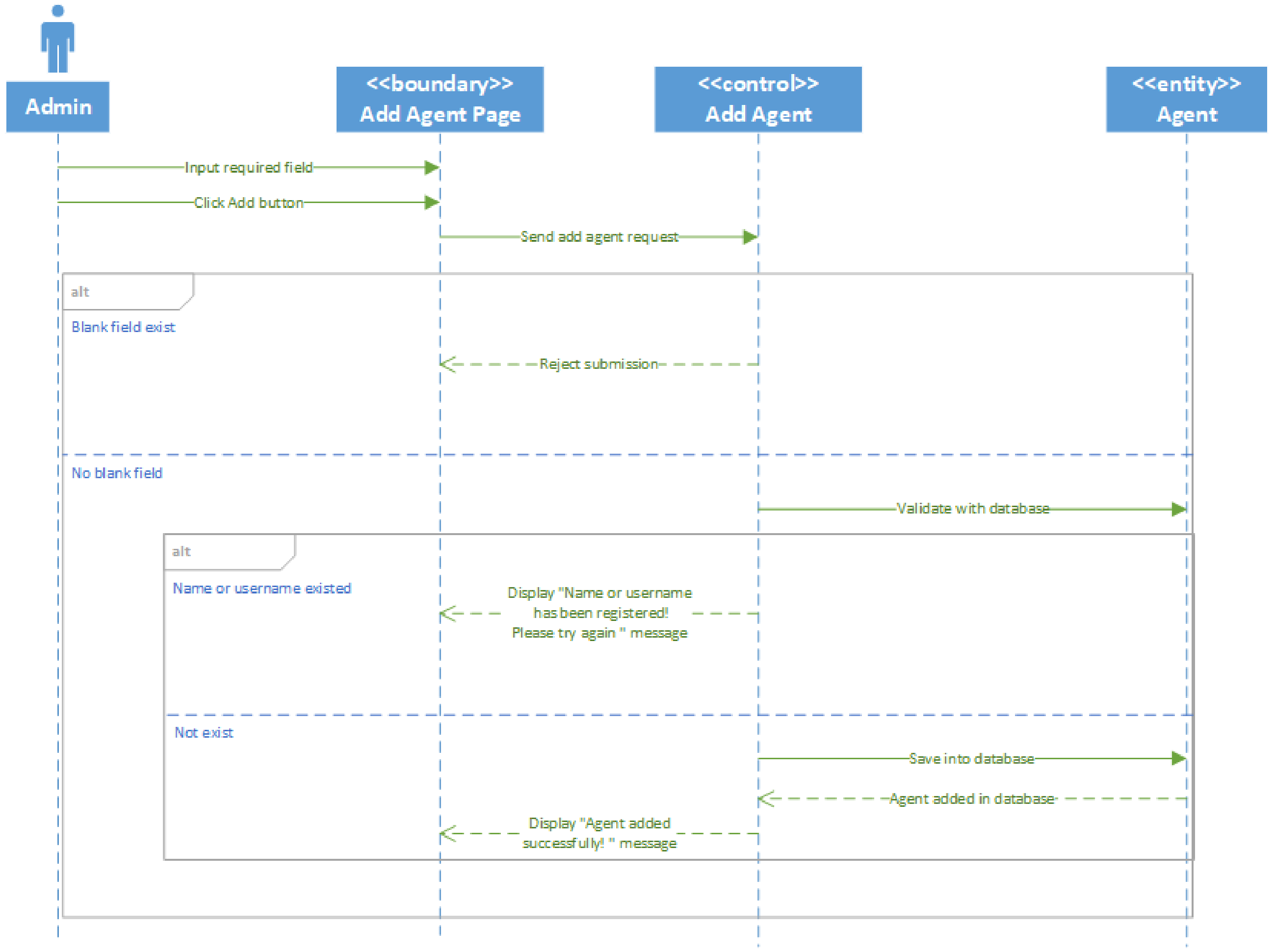


Figure 3: Sequence Diagram for Add Agent

3.2.3.2 Add Route

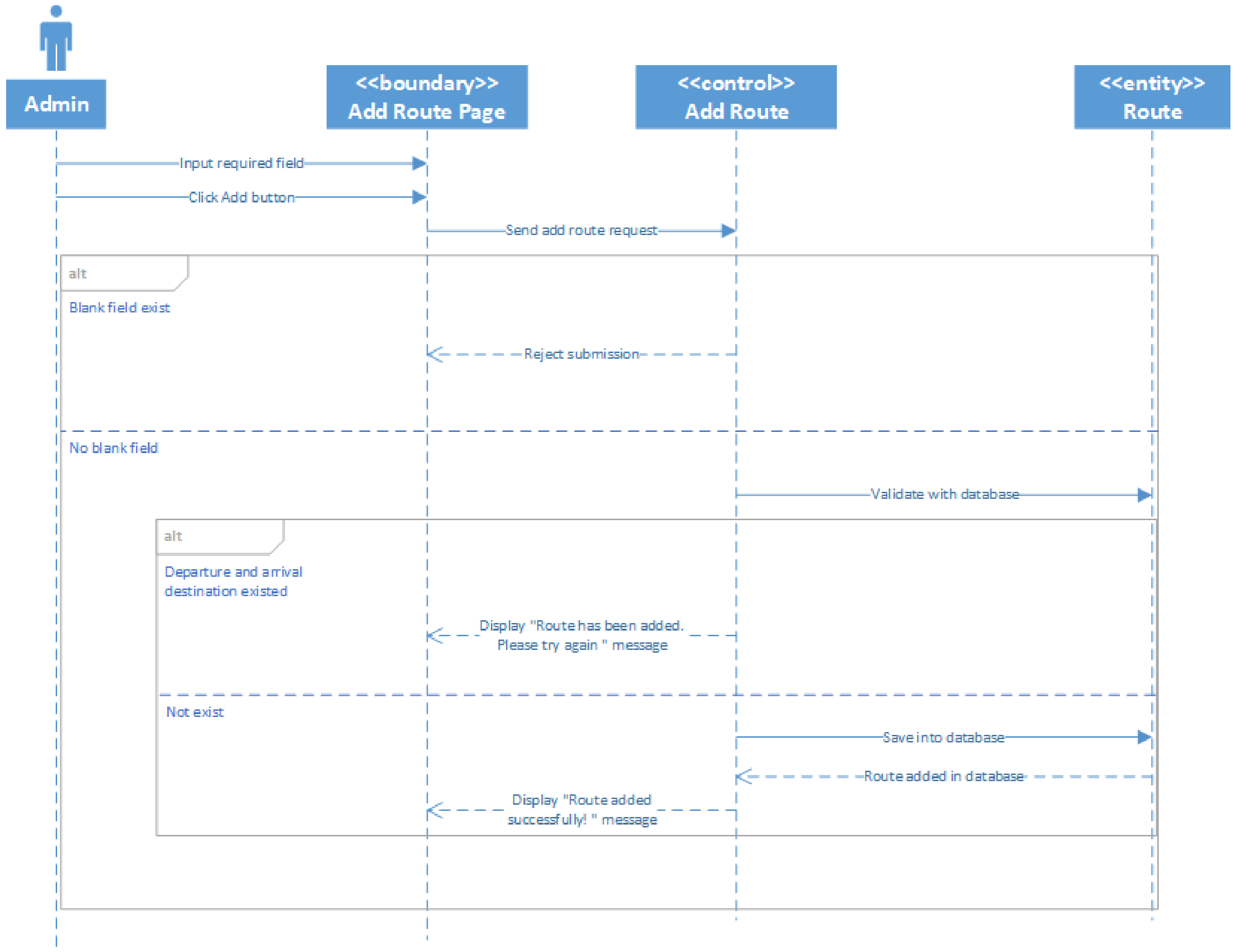


Figure 4: Sequence Diagram for Add Route

3.2.3.3 Add Ship

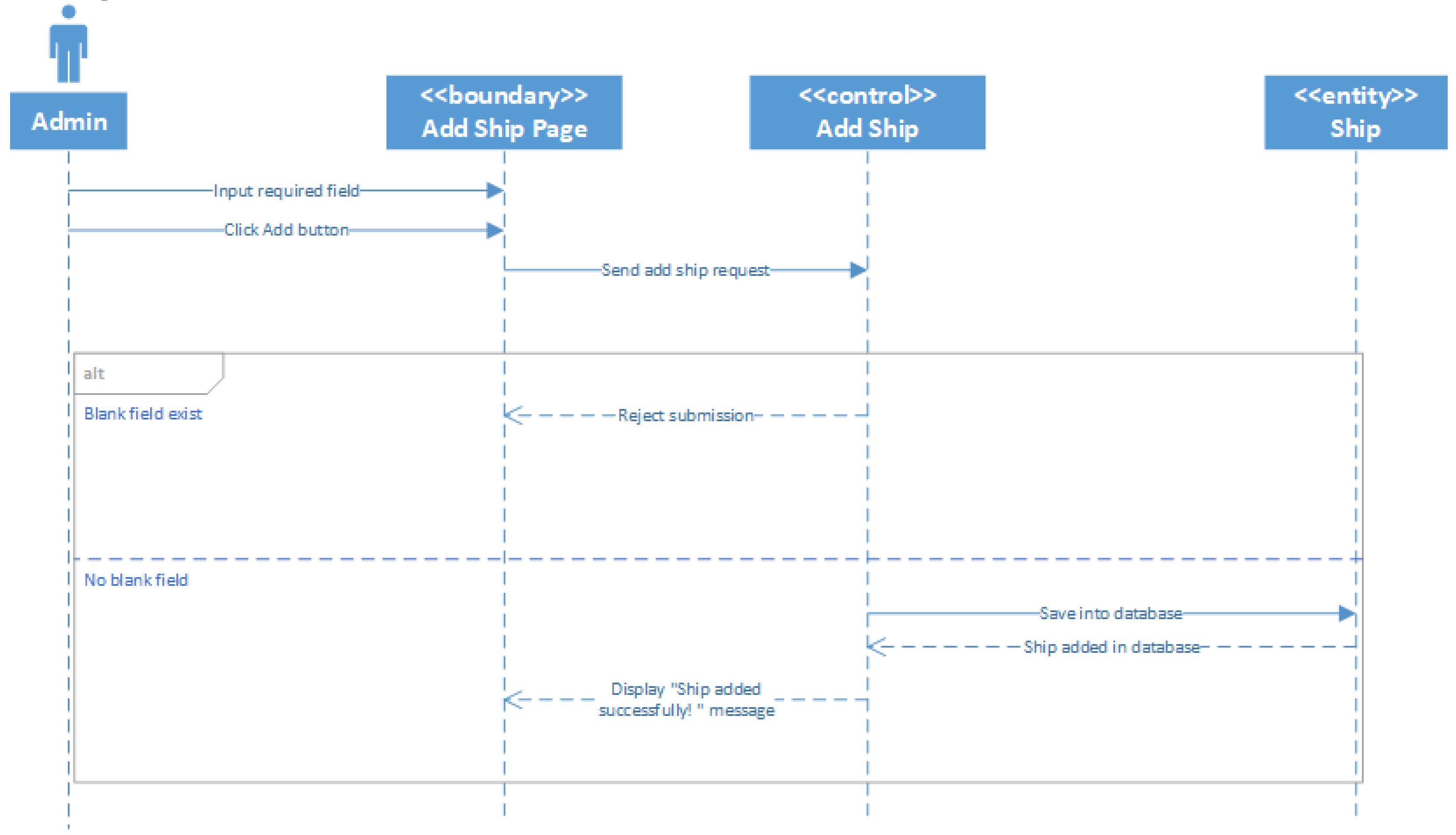


Figure 5: Sequence Diagram for Add Ship

3.2.3.4 View Booking

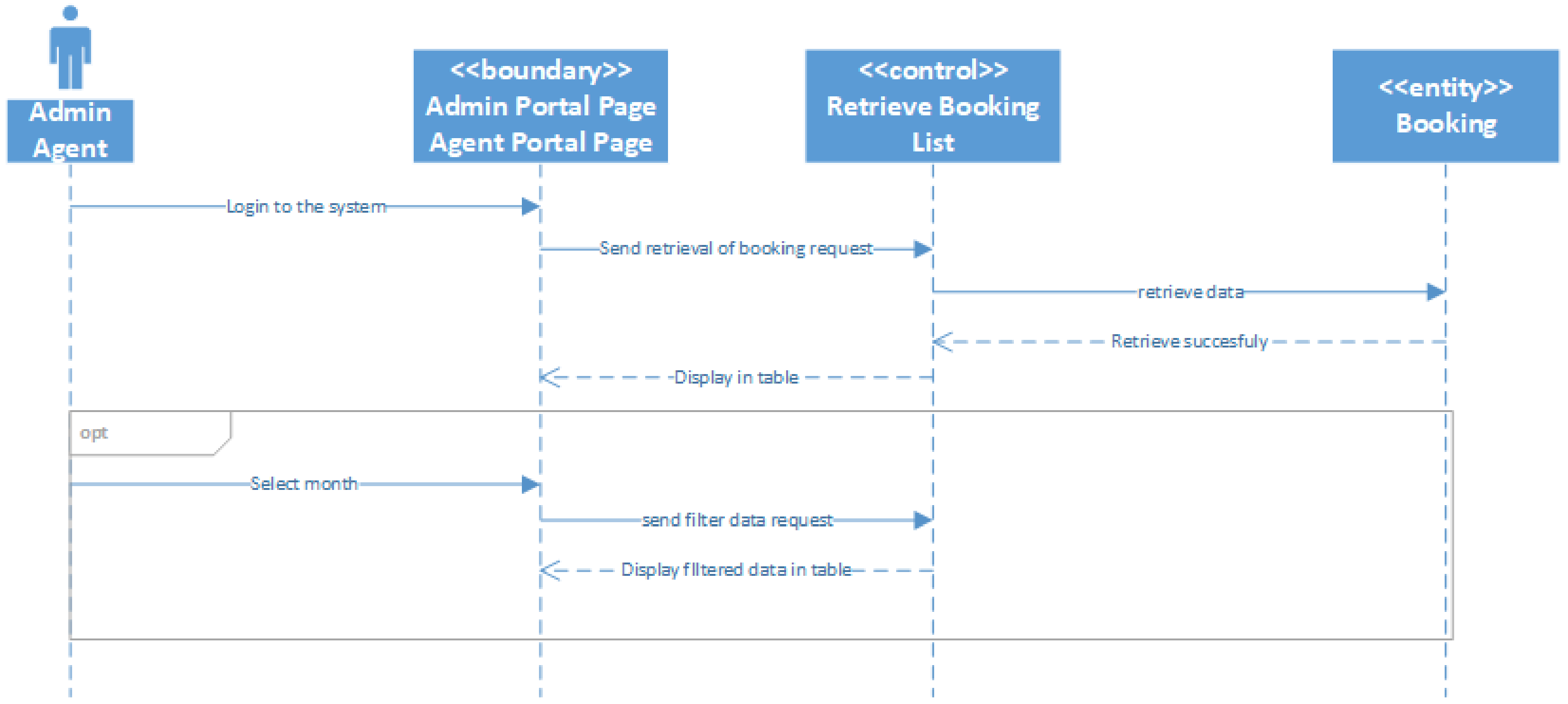


Figure 6: Sequence Diagram for View Booking

3.2.3.5 View Agent

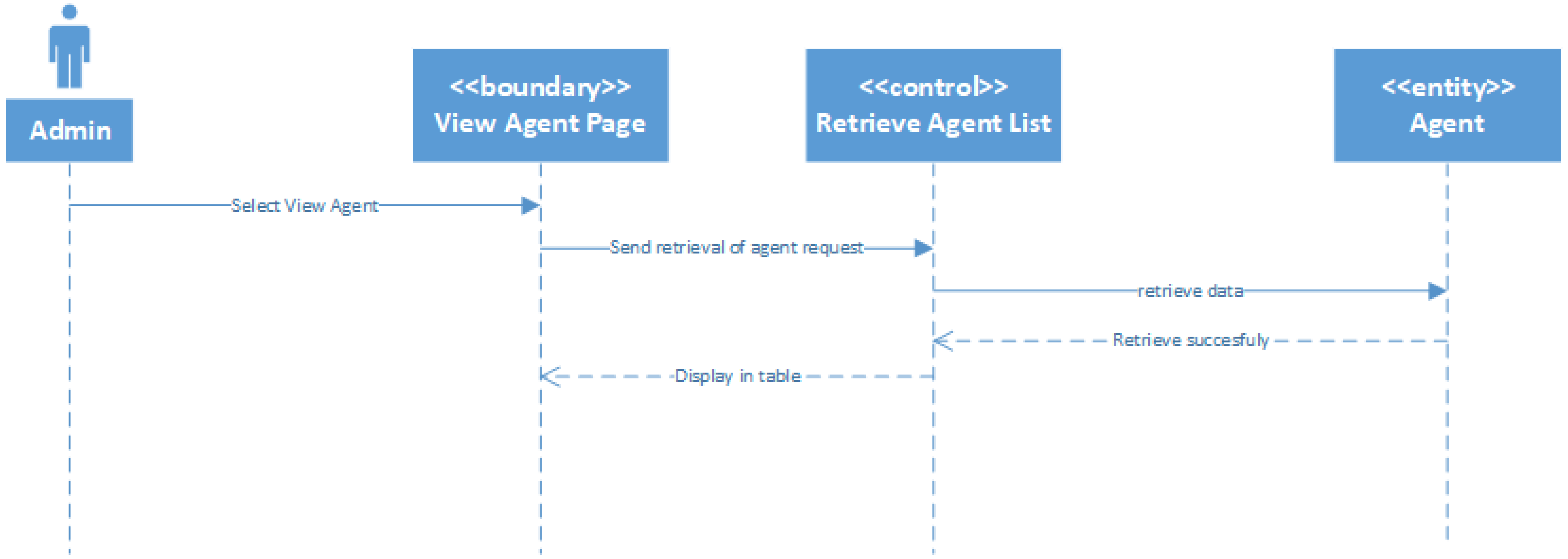

Admin
<<boundary>> View Agent Page
<<control>> Retrieve Agent List
<<entity>> Agent
Select View Agent
Send retrieval of agent request
retrieve data
Retrieve succesfuly
Display in table

Figure 7: Sequence Diagram for View Agent

3.2.3.6 Top Up Credit

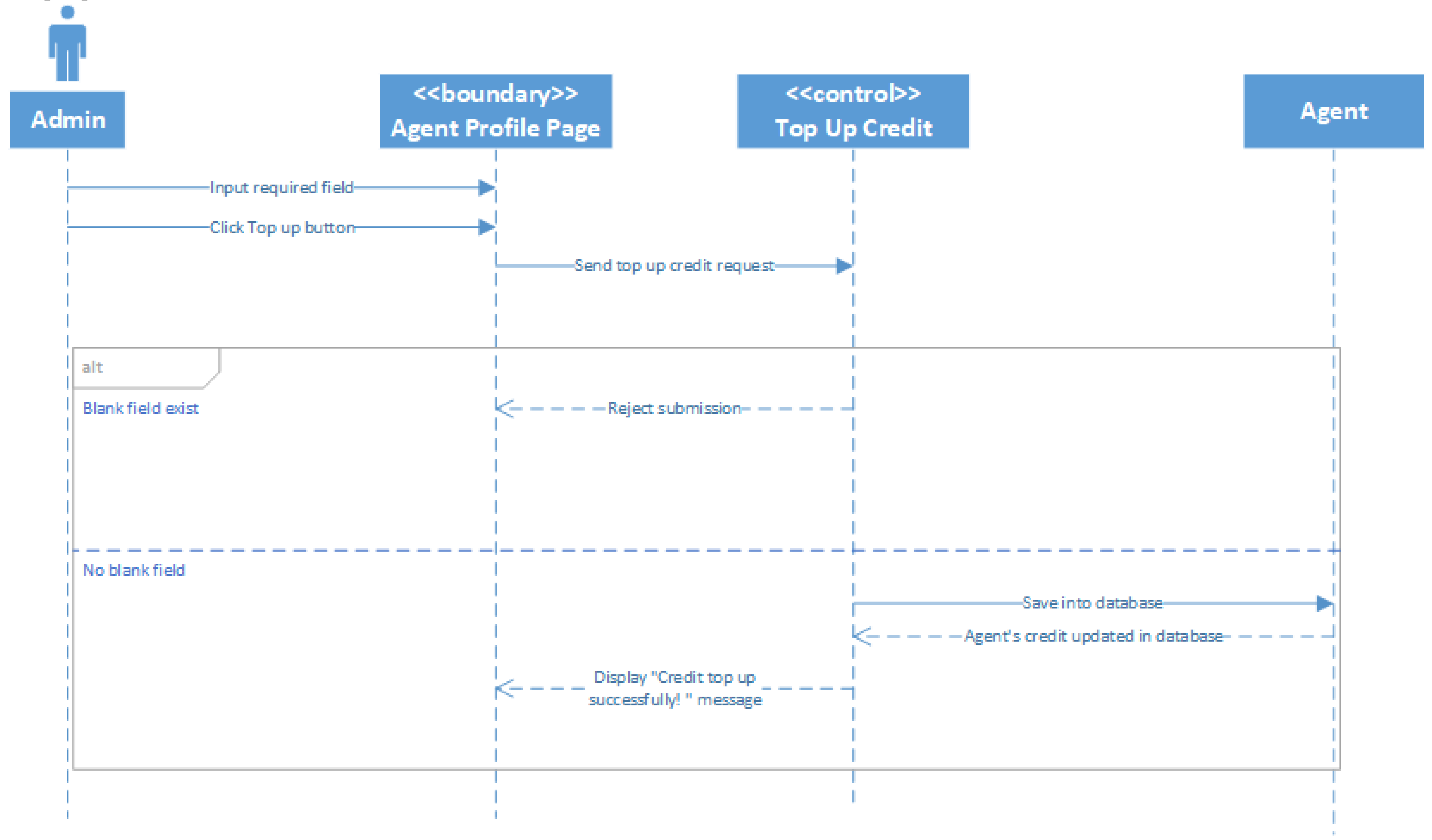


Figure 8: Sequence Diagram for Top Up Credit

3.2.3.7 Edit Agent Profile

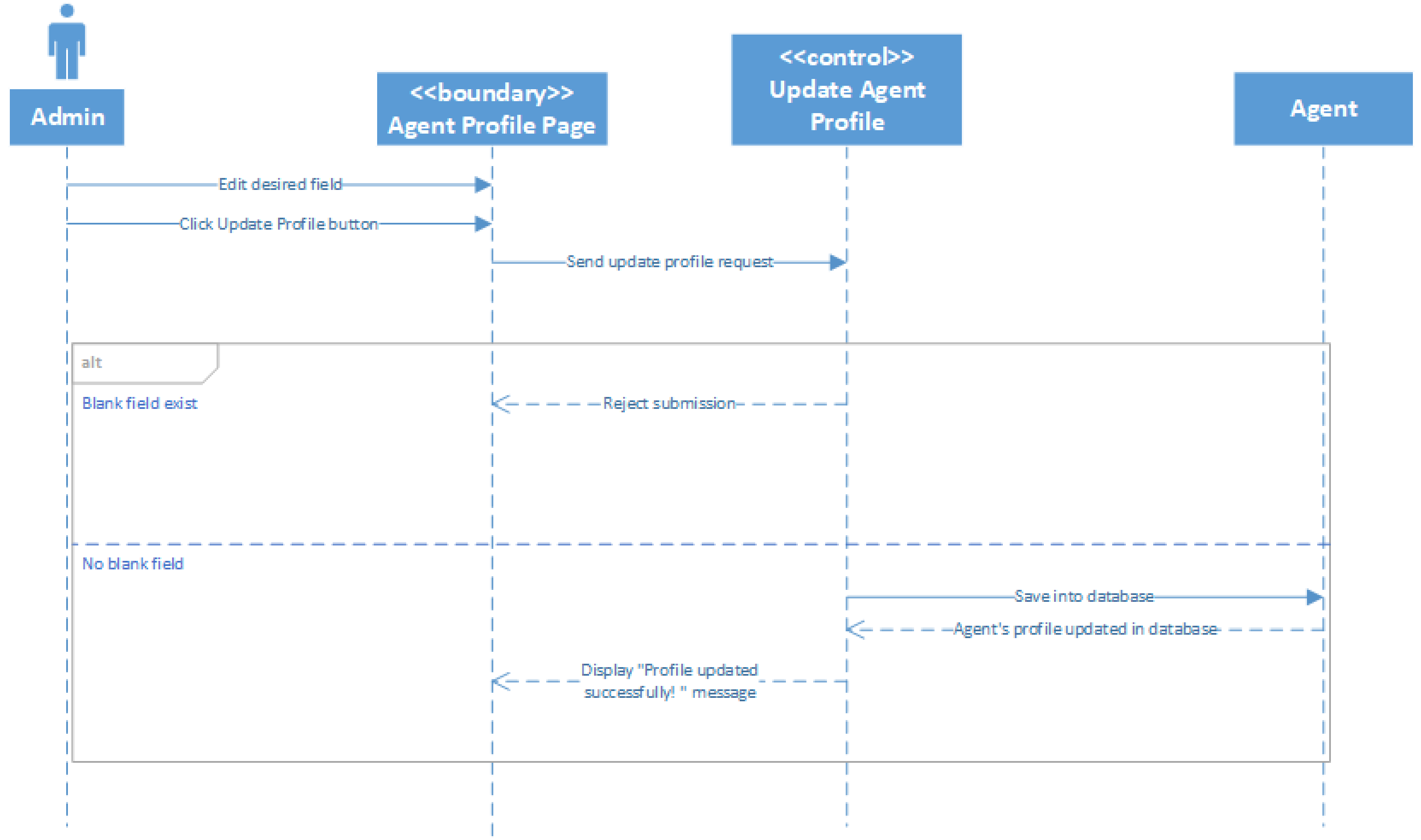

Admin
<<boundary>> Agent Profile Page
<<control>> Update Agent Profile
Agent
Edit desired field
Click Update Profile button
Send update profile request
alt
Blank field exist
Reject submission
No blank field
Save into database
Agent's profile updated in database
Display "Profile updated successfully! " message

Figure 9: Sequence Diagram for Edit Agent Profile

3.2.3.8 Create Schedule

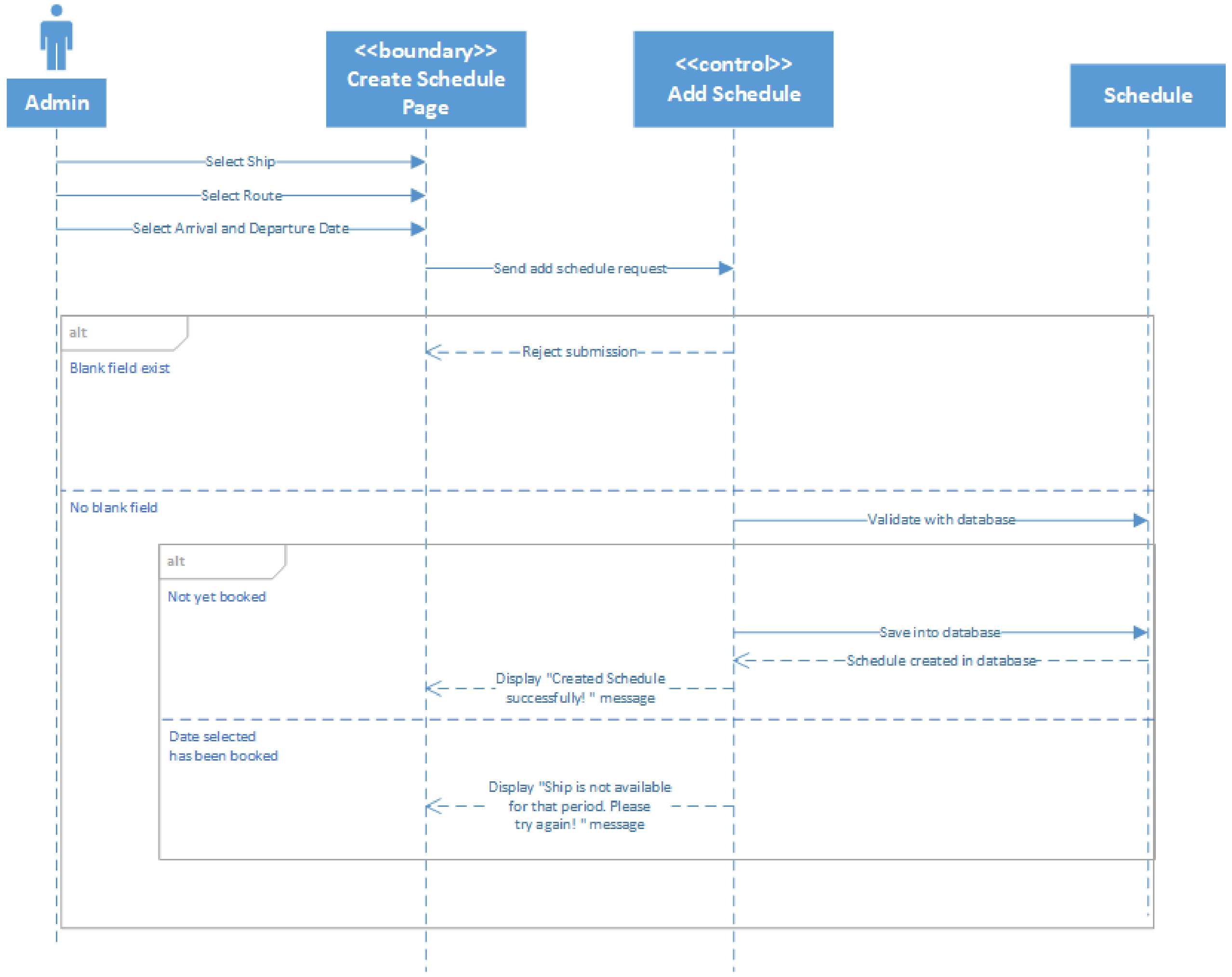

Admin
<<boundary>> Create Schedule Page
<<control>> Add Schedule
Schedule
Select Ship
Select Route
Select Arrival and Departure Date
Send add schedule request
alt
Blank field exist
Reject submission
No blank field
Validate with database
alt
Not yet booked
Save into database
Schedule created in database
Display "Created Schedule successfully! " message
Date selected has been booked
Display "Ship is not available for that period. Please try again! " message

Figure 10: Sequence Diagram for Create Schedule

3.2.3.9 Register Customer

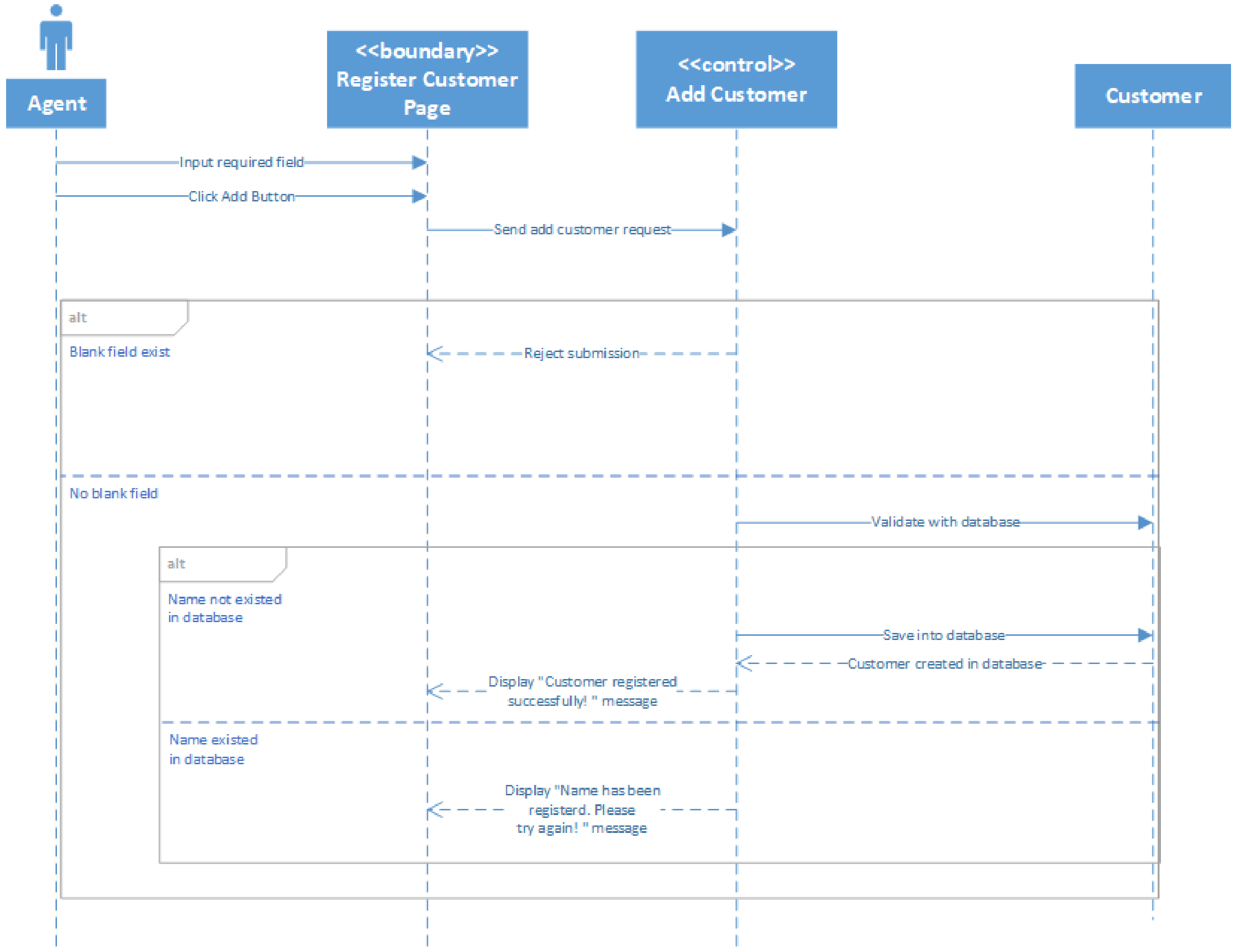

Agent
<<boundary>> Register Customer Page
<<control>> Add Customer
Customer
Input required field
Click Add Button
Send add customer request
alt
Blank field exist
Reject submission
No blank field
Validate with database
alt
Name not existed in database
Save into database
Customer created in database
Display "Customer registered successfully! " message
Name existed in database
Display "Name has been registerd. Please try again! " message

Figure 11: Sequence Diagram for Add Customer

3.2.3.10 Login

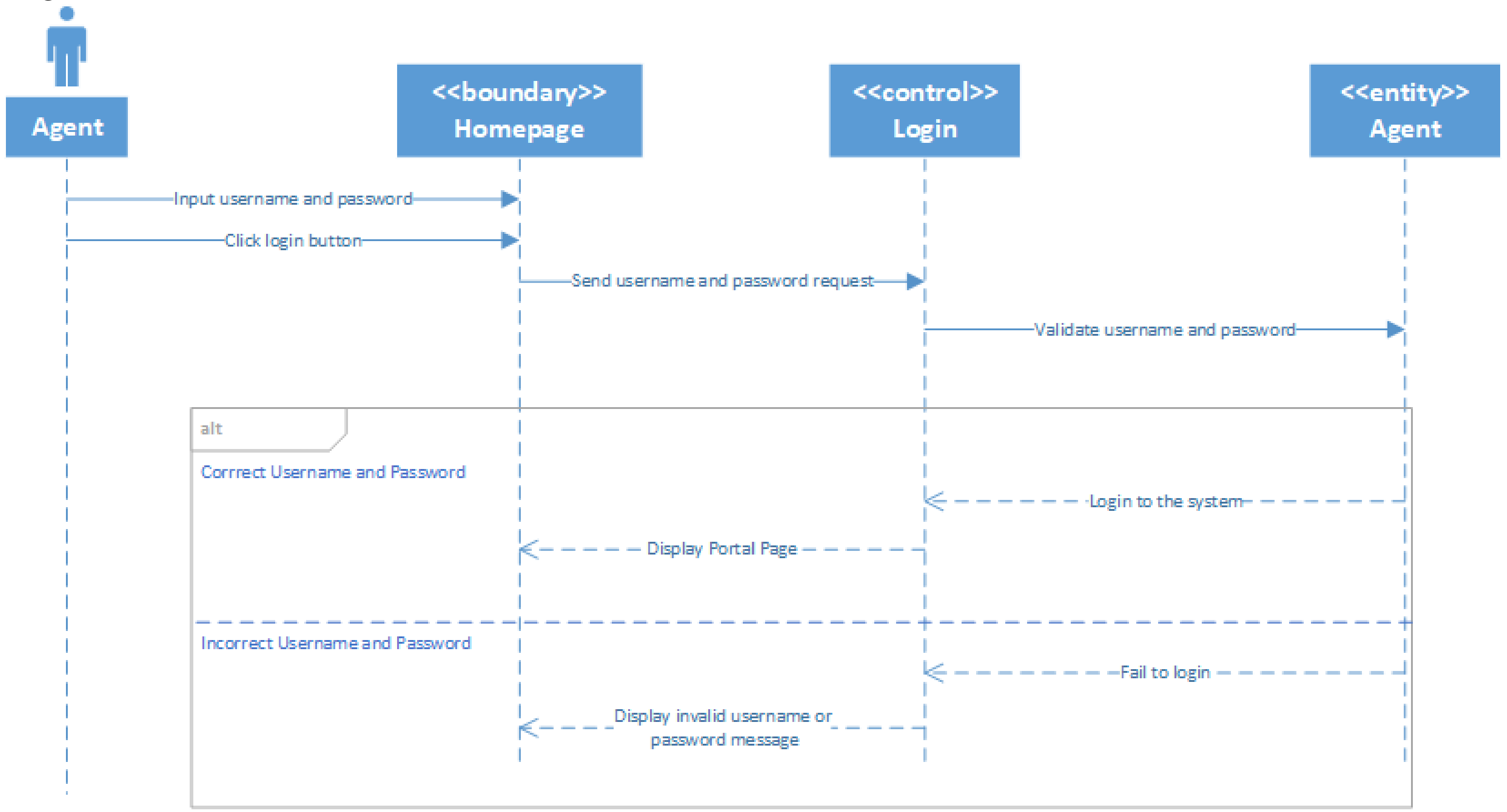

Agent
<<boundary>> Homepage
<<control>> Login
<<entity>> Agent
Input username and password
Click login button
Send username and password request
Validate username and password
alt
Corrrect Username and Password
Login to the system
Display Portal Page
Incorrect Username and Password
Fail to login
Display invalid username or password message

Figure 12: Sequence Diagram for Login

3.2.3.11 Register Item

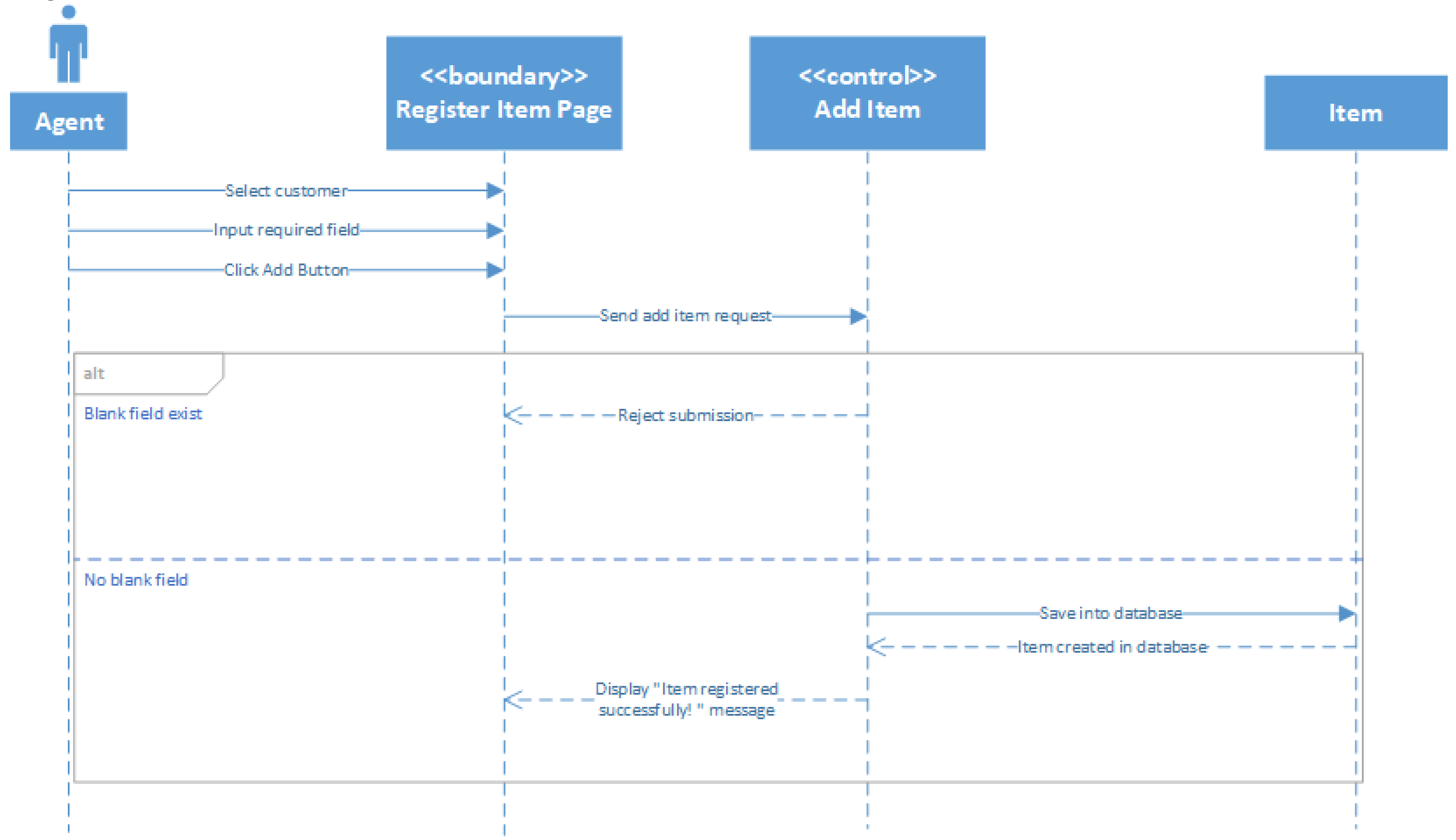


Figure 13: Sequence Diagram for Add Item

3.2.3.12 Book Vessel

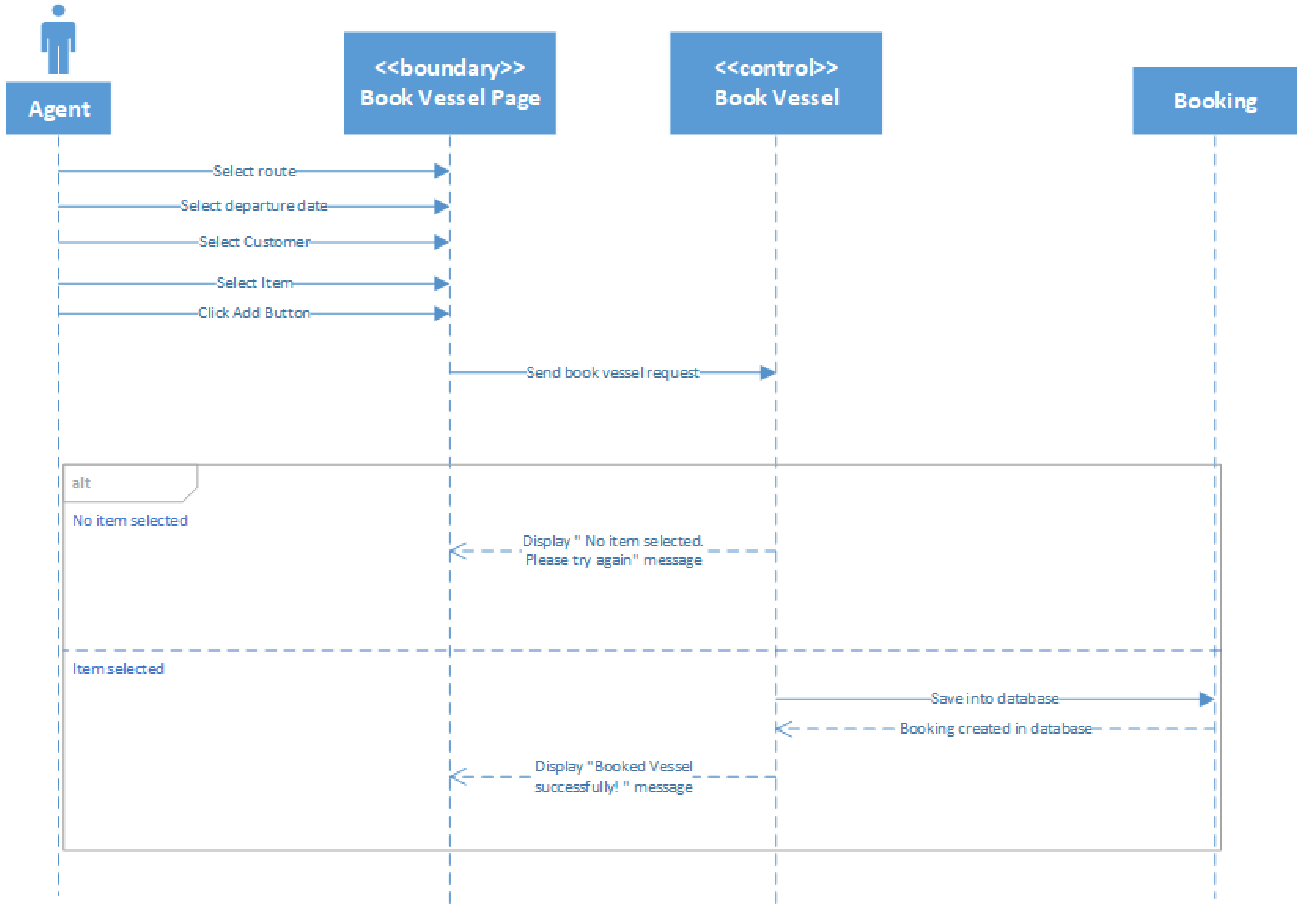


Figure 14: Sequence Diagram for Book Vessel

3.2.4 Class Diagram

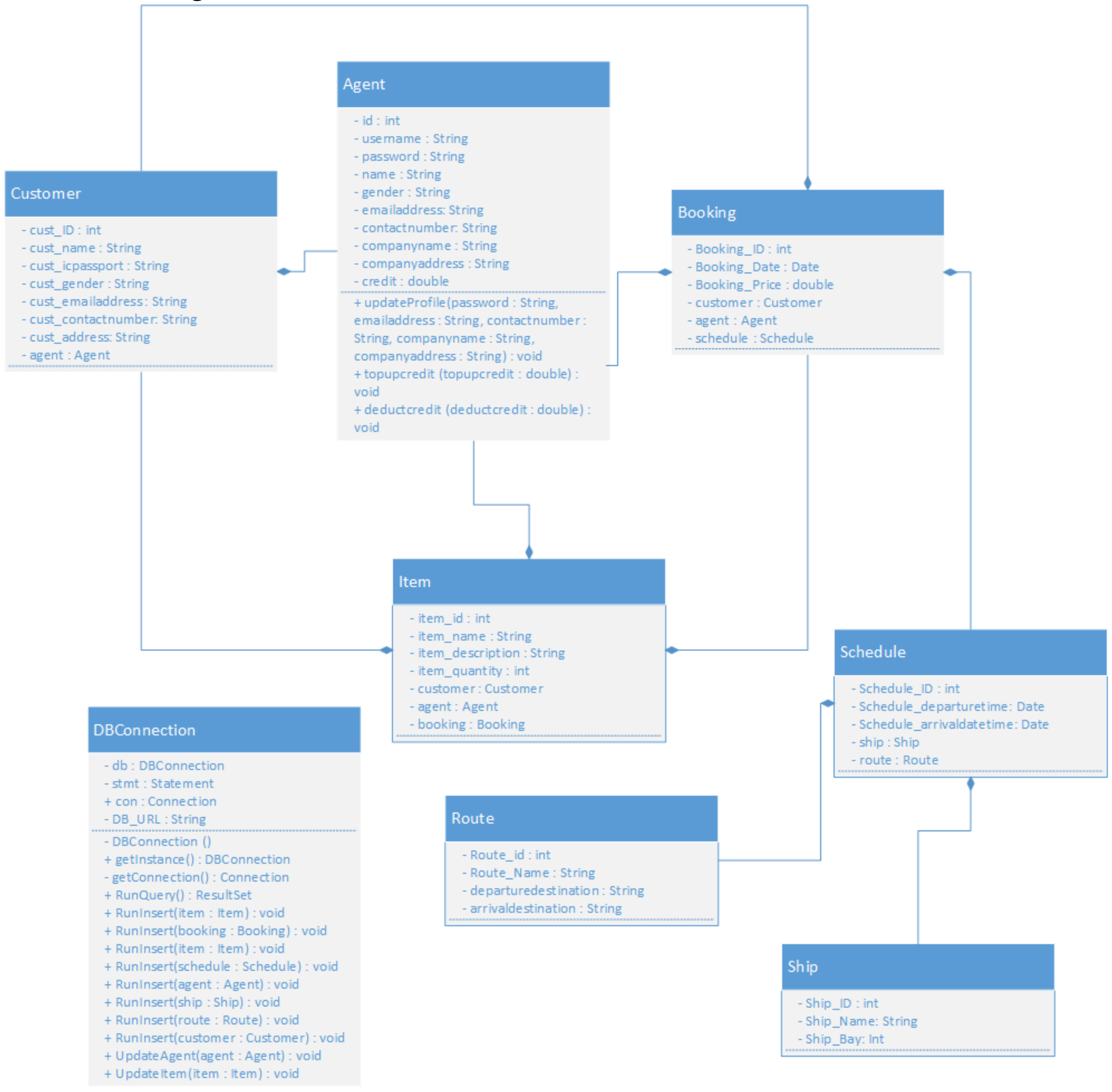

Agent
- id : int
- username : String
- password : String
- name : String
- gender : String
- emailaddress: String
- contactnumber: String
- companyname : String
- companyaddress : String
- credit : double
+ updateProfile(password : String, emailaddress : String, contactnumber : String, companyname : String, companyaddress : String) : void
+ topupcredit (topupcredit : double) : void
+ deductcredit (deductcredit : double) : void
Customer
- cust_ID : int
- cust_name : String
- cust_icpassport : String
- cust_gender : String
- cust_emailaddress: String
- cust_contactnumber: String
- cust_address: String
- agent : Agent
Booking
- Booking_ID : int
- Booking_Date : Date
- Booking_Price : double
- customer : Customer
- agent : Agent
- schedule : Schedule
Item
- item_id : int
- item_name : String
- item_description : String
- item_quantity : int
- customer : Customer
- agent : Agent
- booking : Booking
Schedule
- Schedule_ID : int
- Schedule_departuretime: Date
- Schedule_arrivaldatetime: Date
- ship : Ship
- route : Route
DBConnection
- db : DBConnection
- stmt : Statement
+ con : Connection
- DB_URL : String
- DBConnection ()
+ getInstance() : DBConnection
- getConnection() : Connection
+ RunQuery() : ResultSet
+ RunInsert(item : Item) : void
+ RunInsert(booking : Booking) : void
+ RunInsert(item : Item) : void
+ RunInsert(schedule : Schedule) : void
+ RunInsert(agent : Agent) : void
+ RunInsert(ship : Ship) : void
+ RunInsert(route : Route) : void
+ RunInsert(customer : Customer) : void
+ UpdateAgent(agent : Agent) : void
+ UpdateItem(item : Item) : void
Route
- Route_id : int
- Route_Name : String
- departuredestination : String
- arrivaldestination : String
Ship
- Ship_ID : int
- Ship_Name: String
- Ship_Bay: Int

Figure 15: Class Diagram

3.2.5 Cloud Architecture Diagram

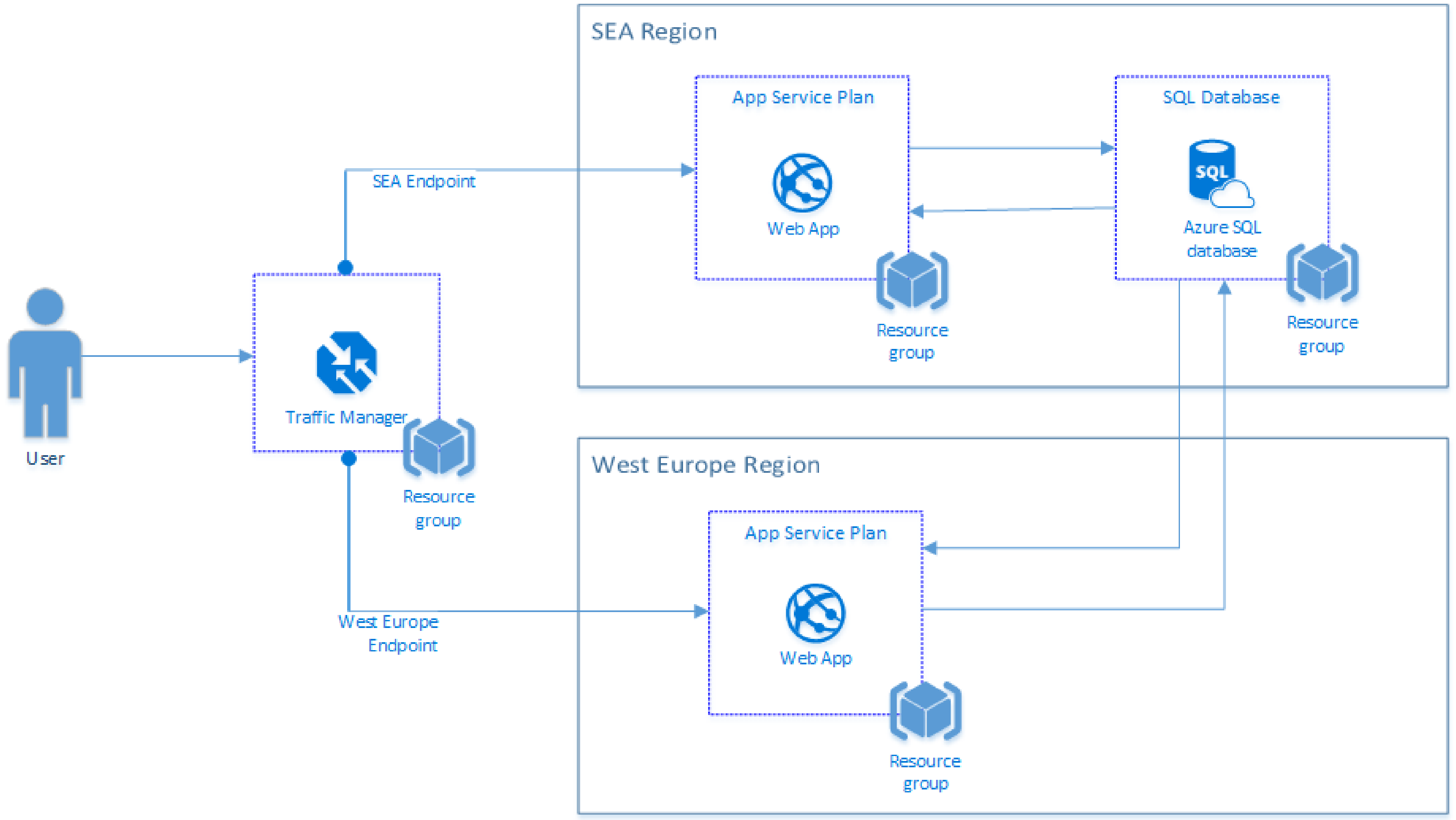
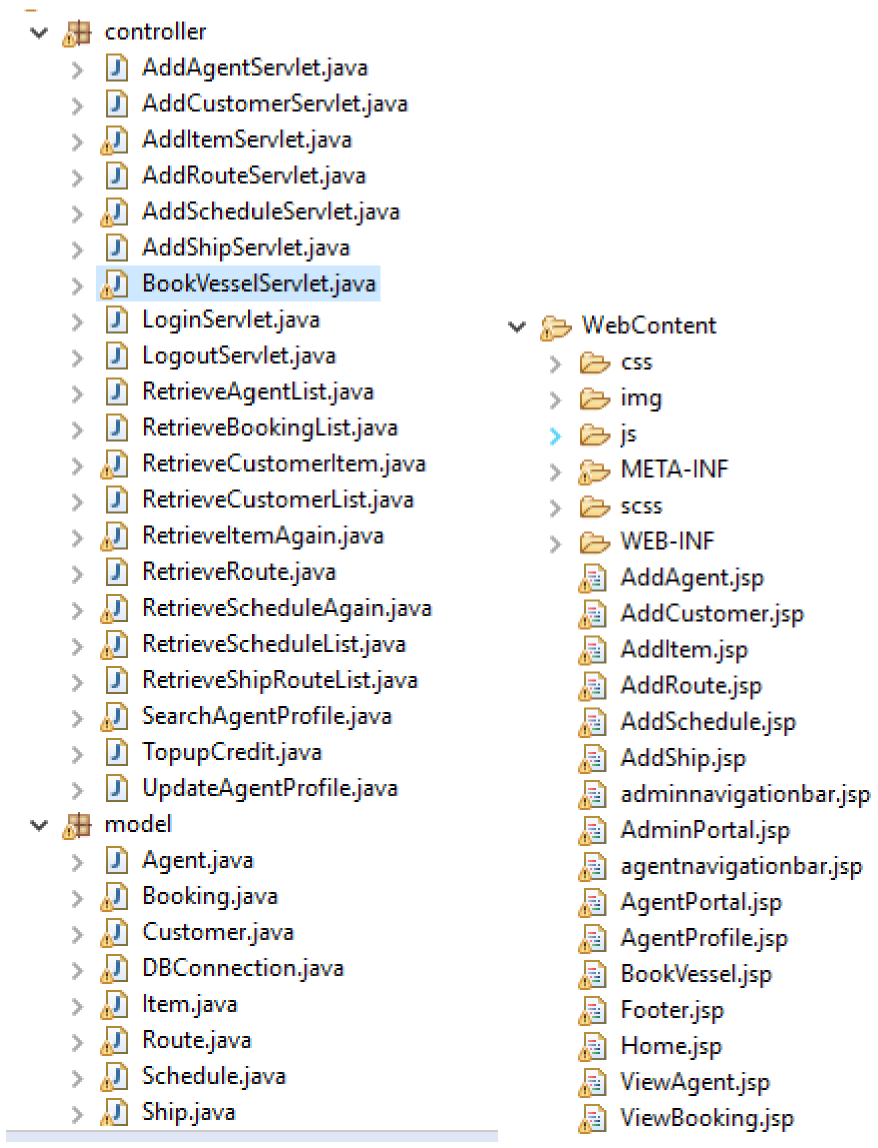


Figure 16: Cloud Architecture Diagram

4 Implementation

4.1 Application Development

The application was developed by using the IDE, Eclipse in which uses Java. The application was implemented by MVC structure in which the classes are the model, JSP are the view and Servlet acts as controller. Furthermore, Microsoft SQL database was used to store relevant data of the application. The file structure is showed as below:



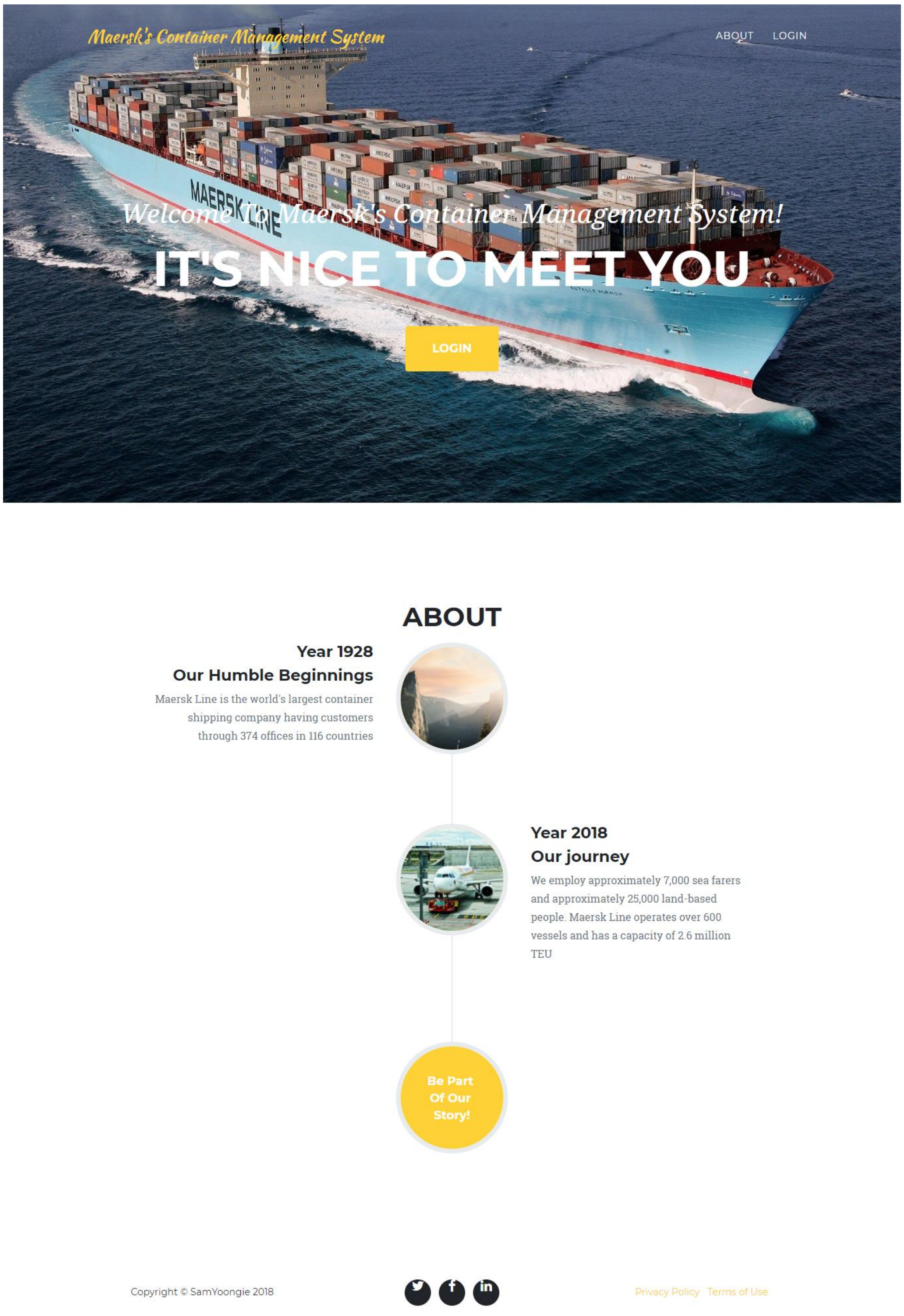
```
controller
    AddAgentServlet.java
    AddCustomerServlet.java
    AddItemServlet.java
    AddRouteServlet.java
    AddScheduleServlet.java
    AddShipServlet.java
    BookVesselServlet.java
    LoginServlet.java
    LogoutServlet.java
    RetrieveAgentList.java
    RetrieveBookingList.java
    RetrieveCustomerItem.java
    RetrieveCustomerList.java
    RetrieveItemAgain.java
    RetrieveRoute.java
    RetrieveScheduleAgain.java
    RetrieveScheduleList.java
    RetrieveShipRouteList.java
    SearchAgentProfile.java
    TopupCredit.java
    UpdateAgentProfile.java
model
    Agent.java
    Booking.java
    Customer.java
    DBConnection.java
    Item.java
    Route.java
    Schedule.java
    Ship.java

WebContent
    css
    img
    js
    META-INF
    scss
    WEB-INF
    AddAgent.jsp
    AddCustomer.jsp
    AddItem.jsp
    AddRoute.jsp
    AddSchedule.jsp
    AddShip.jsp
    adminnavigationbar.jsp
    AdminPortal.jsp
    agentnavigationbar.jsp
    AgentPortal.jsp
    AgentProfile.jsp
    BookVessel.jsp
    Footer.jsp
    Home.jsp
    ViewAgent.jsp
    ViewBooking.jsp
```

Figure 17: File Structure

The first page of the application will be Home.jsp in which this is the home page of the entire application. Once the application starts up, it will redirect the user to Home.jsp. Users are required to login through this page before accessing to another page. After users has logged in to the application, the application will either redirect the user to Agent Portal or Admin Portal based on the username and password inputted.

The UI of the application was developed using JSP technology and the Homepage for Container Management System is as below:



Maersk Line is the world's largest container shipping company having customers through 374 offices in 116 countries

We employ approximately 7,000 sea farers and approximately 25,000 land-based people. Maersk Line operates over 600 vessels and has a capacity of 2.6 million TEU

Figure 18: Home page

4.2 Azure Publishing

4.2.1 Creating and Configuring a SQL Database

1. In Azure Portal, select Databases > SQL Database from the menu.

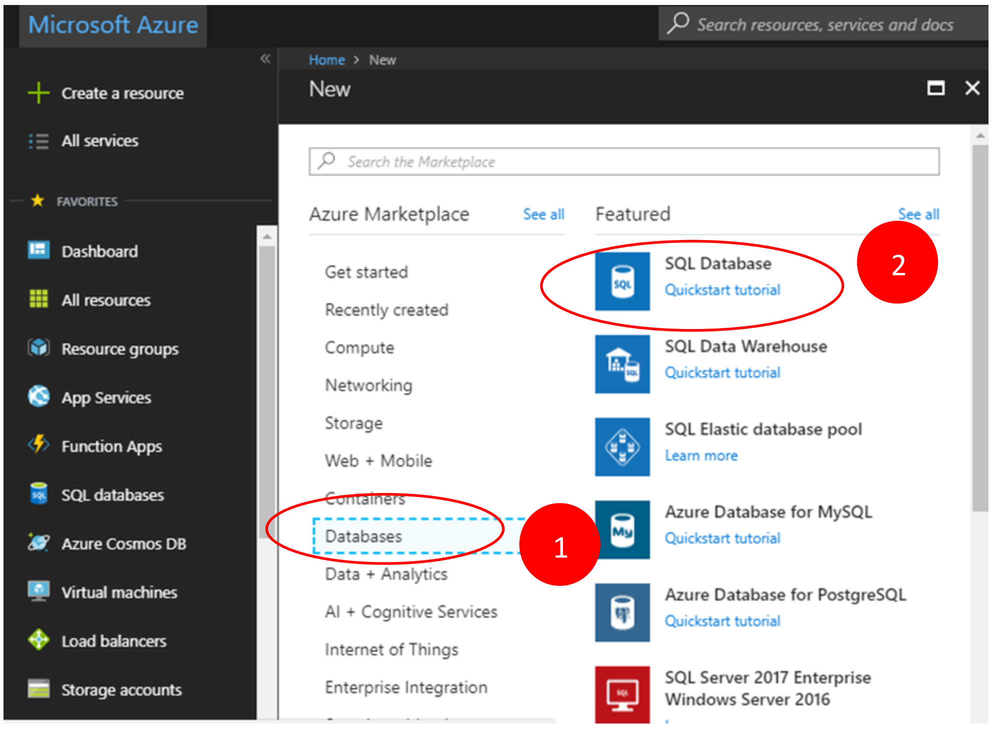
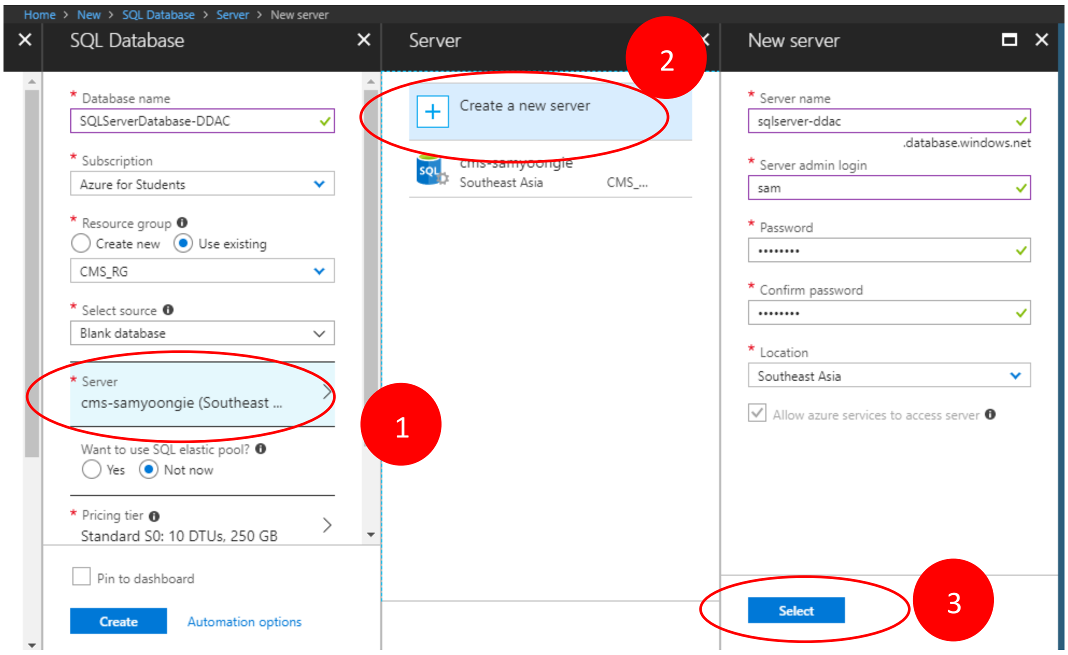


Figure 19: SQL Database

2. Fill in the required details. Click on Server option and choose Create a new server. Fill in the required field and click on Select button.



* Database name: SQLServerDatabase-DDAC
* Subscription: Azure for Students
* Resource group: Create new [✓ Use existing] CMS_RG
* Select source: Blank database
* Server: cms-samyoongie (Southeast ...
Want to use SQL elastic pool? Yes [✓ Not now]
* Pricing tier: Standard S0: 10 DTUs, 250 GB

* Server name: sqlserver-ddac .database.windows.net
* Server admin login: sam
* Password: ••••••••
* Confirm password: ••••••••
* Location: Southeast Asia
[✓] Allow azure services to access server

Figure 20: SQL Database

3. At the pricing tier, choose Standard S0, and move the DTUs to 0. Then, click on Apply button and Create button.

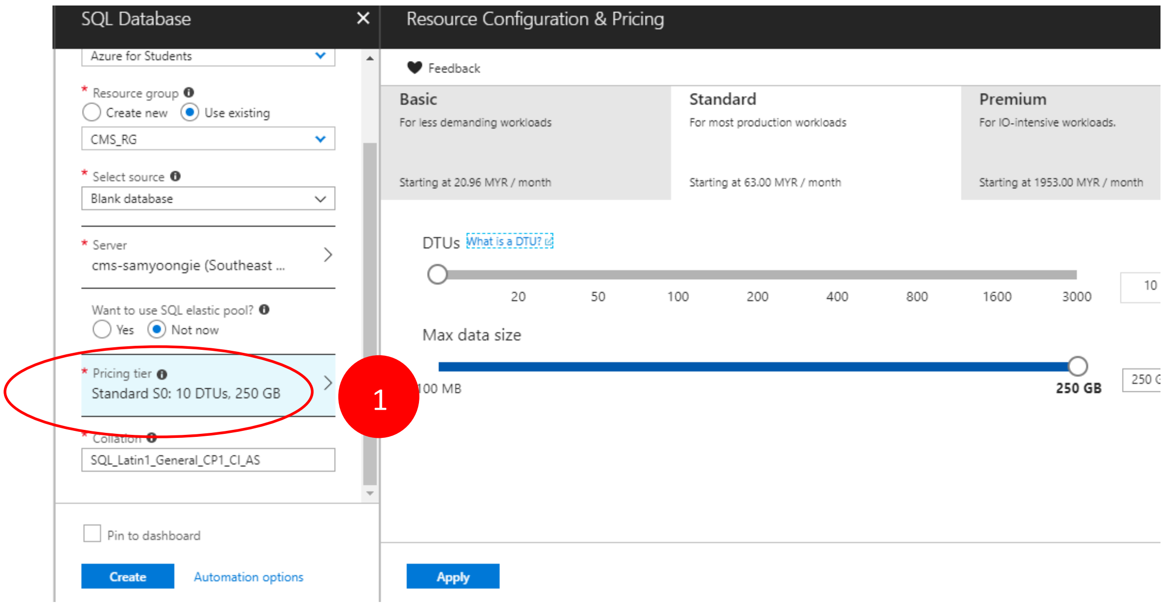

SQL Database
Azure for Students
Resource group
Create new
Use existing
CMS_RG
Select source
Blank database
Server
cms-samyoongie (Southeast ...
Want to use SQL elastic pool?
Yes
Not now
Pricing tier
Standard S0: 10 DTUs, 250 GB
Collation
SQL_Latin1_General_CP1_CI_AS
Pin to dashboard
Create
Automation options
1
Resource Configuration & Pricing
Feedback
Basic
For less demanding workloads
Starting at 20.96 MYR / month
Standard
For most production workloads
Starting at 63.00 MYR / month
Premium
For IO-intensive workloads.
Starting at 1953.00 MYR / month
DTUs What is a DTU?
20 50 100 200 400 800 1600 3000
10
Max data size
100 MB
250 GB
Apply

Figure 21: SQL Database

4. Now, Go to Home and click on the SQL Server that has been deployed earlier.

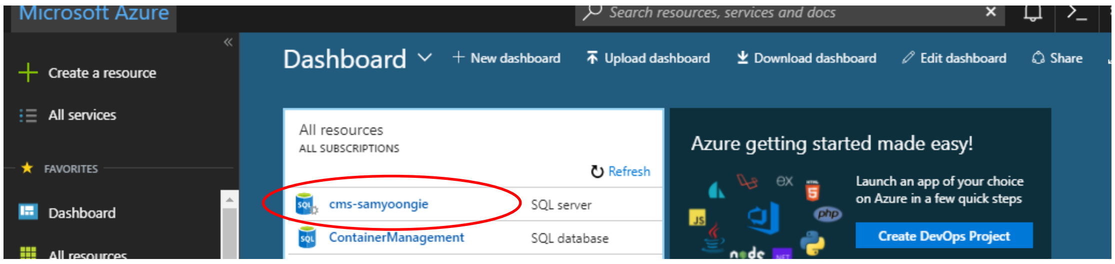
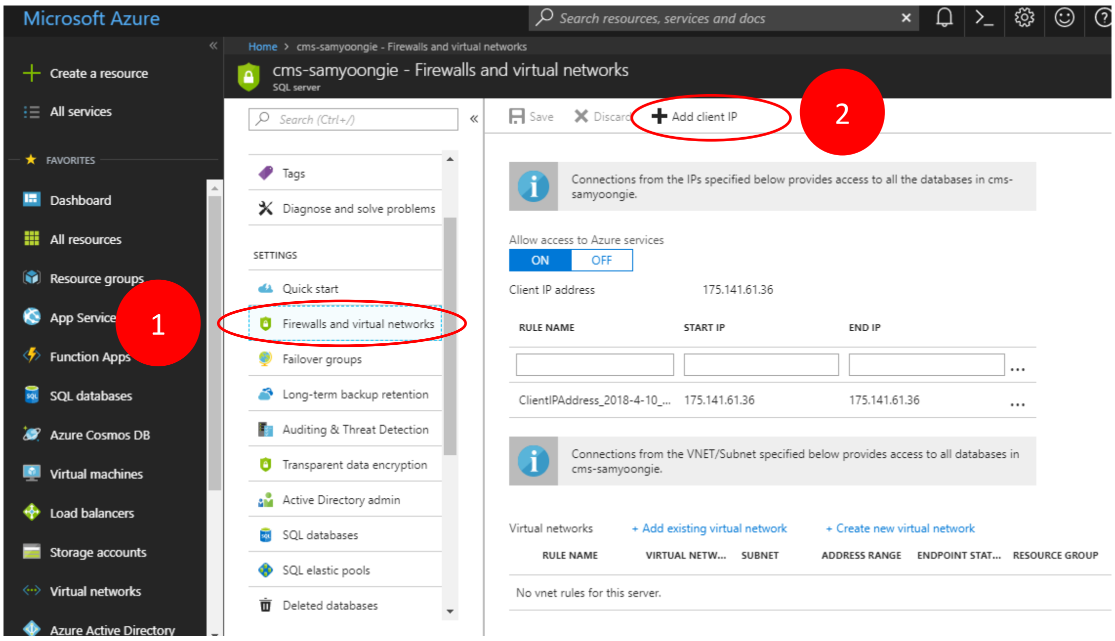


Figure 22: SQL Database

5. Select on Firewalls and virtual networks and clicks on Add client IP to allow the local machine to access the SQL Server created in Azure. Lastly, click on Save button.



Connections from the IPs specified below provides access to all the databases in cms-samyoongie.

Allow access to Azure services: [✓ ON] OFF

Client IP address 175.141.61.36

RULE NAME	START IP	END IP
ClientIPAddress_2018-4-10_...	175.141.61.36	175.141.61.36

Connections from the VNET/Subnet specified below provides access to all databases in cms-samyoongie.

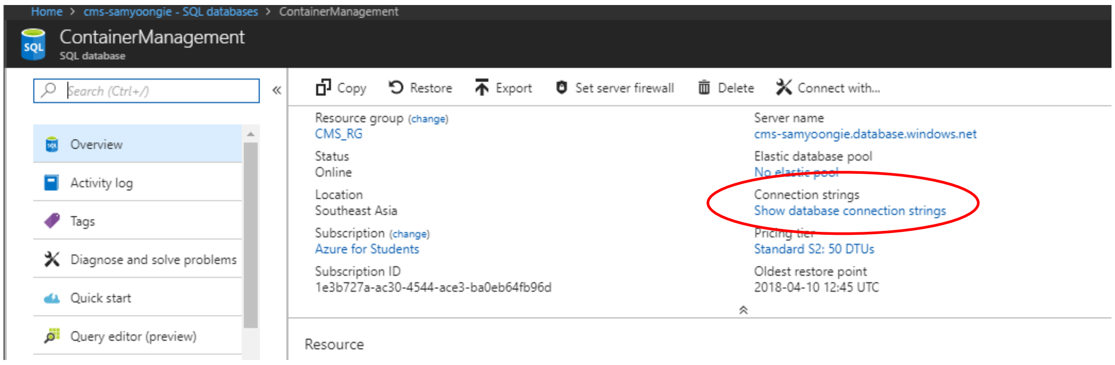
Virtual networks

RULE NAME	VIRTUAL NETW...	SUBNET	ADDRESS RANGE	ENDPOINT STAT...	RESOURCE GROUP

No vnet rules for this server.

Figure 23: SQL Database

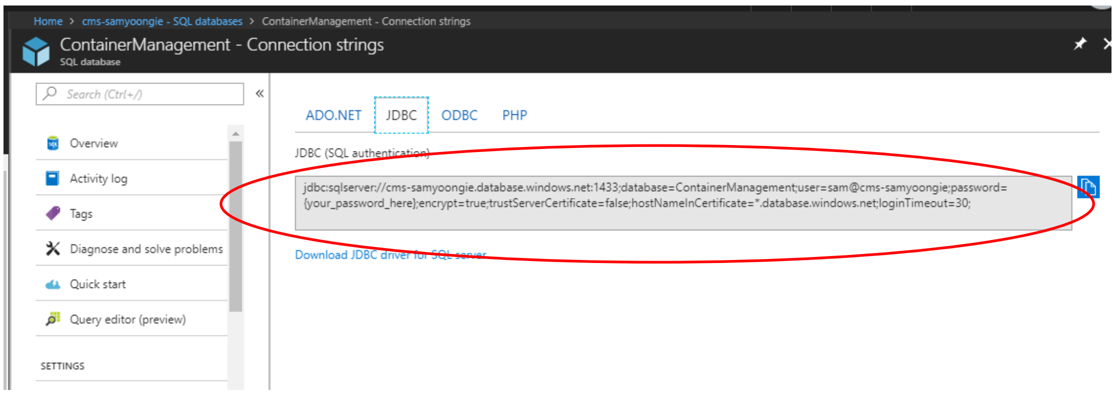
6. Select on the database that selected earlier through Home > SQL Server created > SQL Database and click on Show database connection strings.



Resource group (change)
CMS_RG
Status
Online
Location
Southeast Asia
Subscription (change)
Azure for Students
Subscription ID
1e3b727a-ac30-4544-ace3-ba0eb64fb96d
Server name
cms-samyoongie.database.windows.net
Elastic database pool
No elastic pool
Connection strings
Show database connection strings
Pricing tier
Standard S2: 50 DTUs
Oldest restore point
2018-04-10 12:45 UTC

Figure 24: SQL Database

7. Copy the JDBC connection string onto the application and now the application can access the SQL database created through Azure.


Home > cms-samyoongie - SQL databases > ContainerManagement - Connection strings
ContainerManagement - Connection strings
SQL database
Search (Ctrl+/)
Overview
Activity log
Tags
Diagnose and solve problems
Quick start
Query editor (preview)
SETTINGS
ADO.NET
JDBC
ODBC
PHP
Download JDBC driver for SQL server

JDBC (SQL authentication)

```
jdbc:sqlserver://cms-samyoongie.database.windows.net:1433;database=ContainerManagement;user=sam@cms-samyoongie;password=
{your_password_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;
```

Figure 25: SQL Database

4.2.2 Creating App Service Plan

1. Go to Home > New > Marketplace > Everything. Search app service in the search bar and click on App Service Plan.

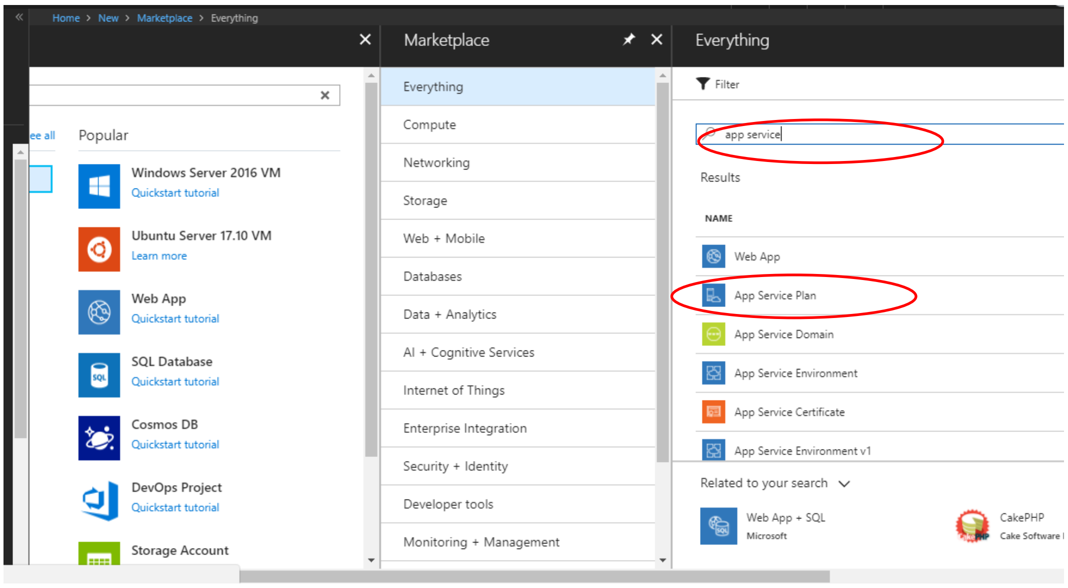
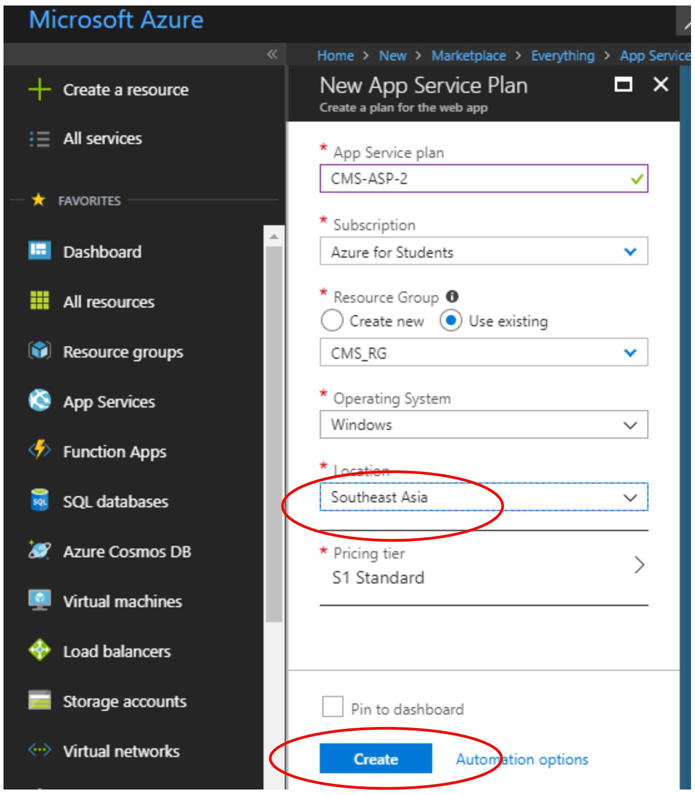


Figure 26: App Service Plan

2. Enter the required field. Select the nearest location and choose the appropriate pricing tier and click Create button.



* App Service plan: CMS-ASP-2
* Subscription: Azure for Students
* Resource Group: Create new [✓ Use existing]
CMS_RG
* Operating System: Windows
* Location: Southeast Asia
* Pricing tier: S1 Standard

Figure 27: App Service Plan

4.2.3 Publish Project through Eclipse

1. In Eclipse, right click the project and choose Azure > Publish as Azure Web App.

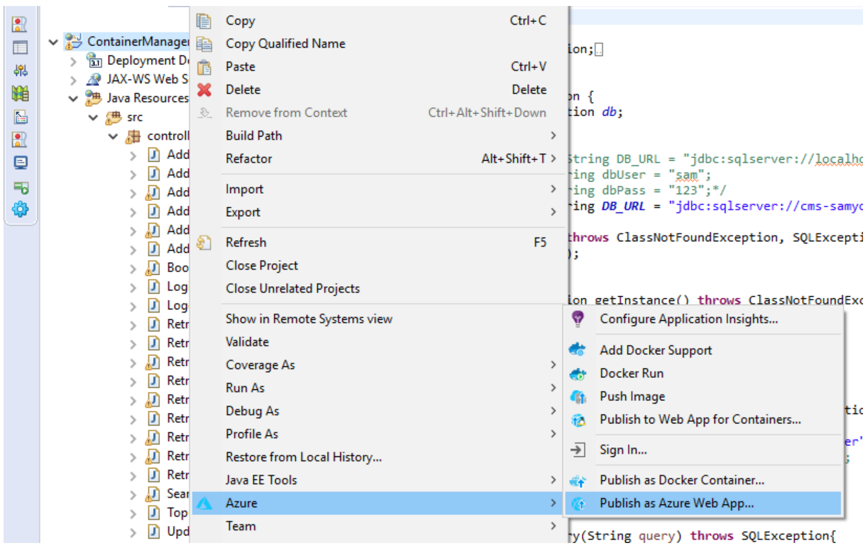
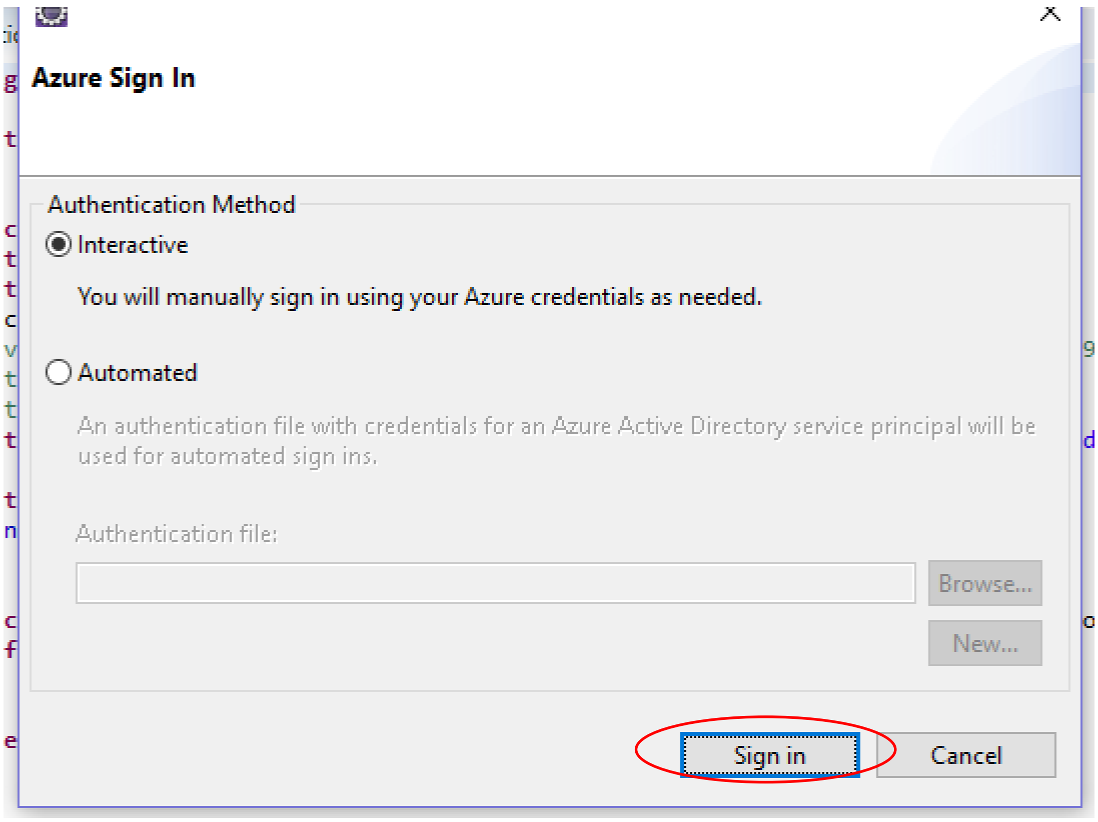


Figure 28: Publish Project

2. Sign in with Azure account.


Azure Sign In

Authentication Method

(●) Interactive

You will manually sign in using your Azure credentials as needed.

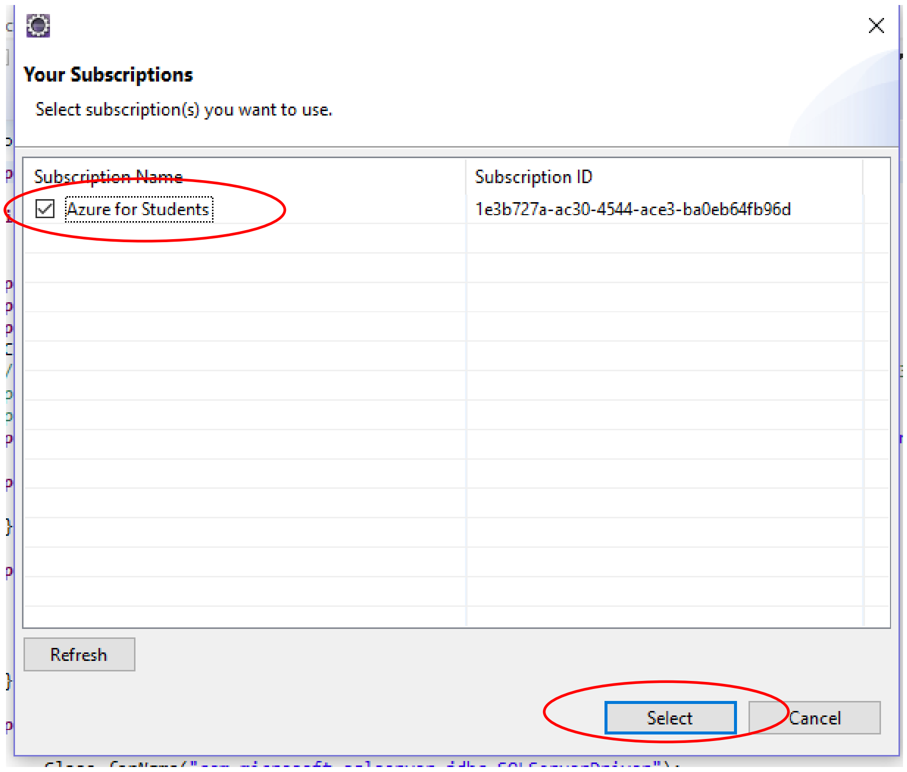
( ) Automated

An authentication file with credentials for an Azure Active Directory service principal will be used for automated sign ins.

Authentication file:

Figure 29: Publish Project

3. Choose the subscription and click Select button.



Subscription Name	Subscription ID
☑ Azure for Students	1e3b727a-ac30-4544-ace3-ba0eb64fb96d

Figure 30: Publish Project

4. Click Create Button.

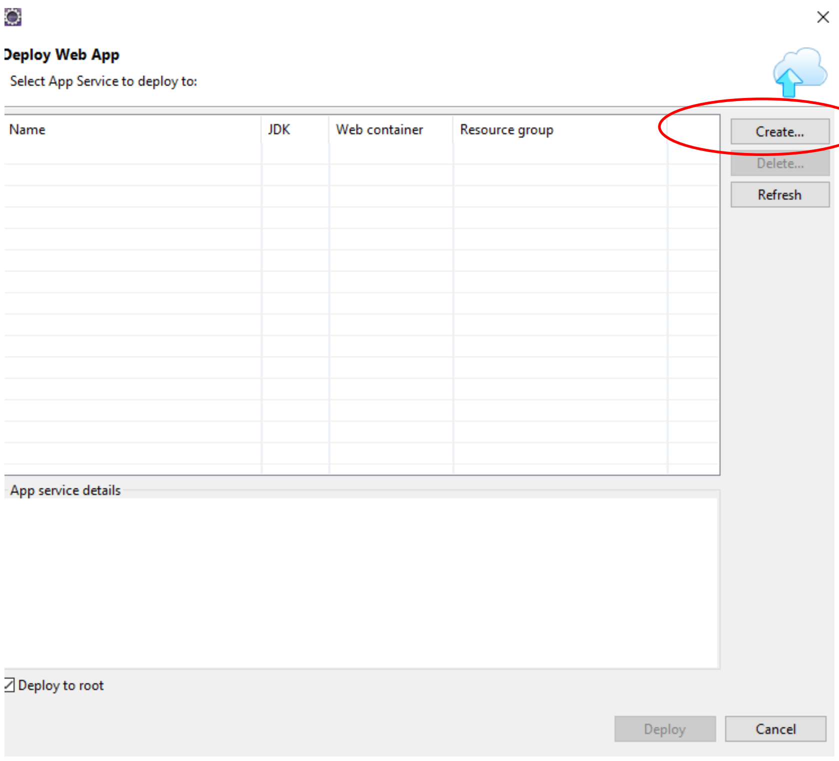
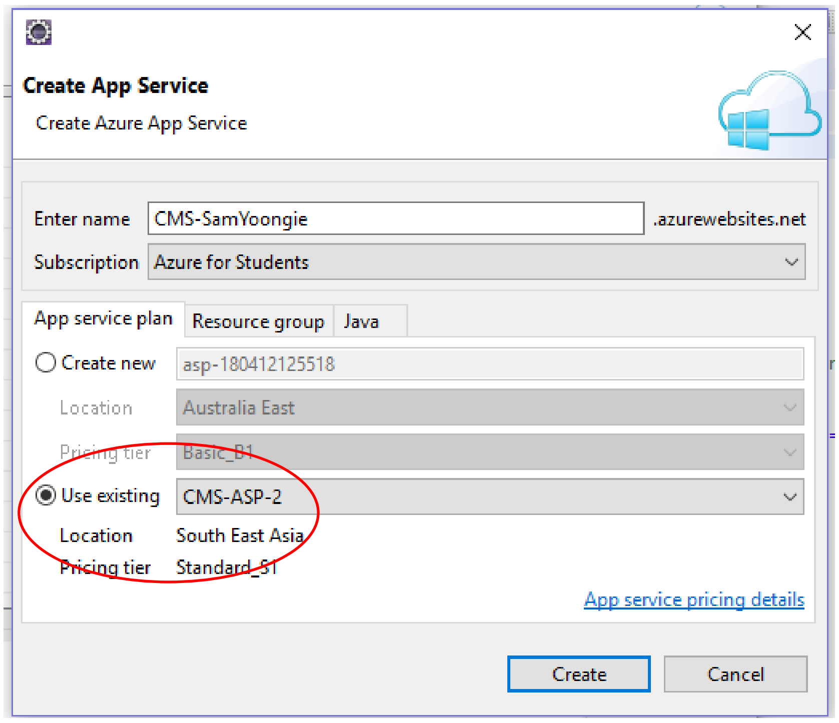


Figure 31: Publish Project

5. Fill in the required details and choose the app service plan and resource group that created earlier. Choose the Java version and web container for the application. Lastly, click on create button.



Create App Service
Create Azure App Service

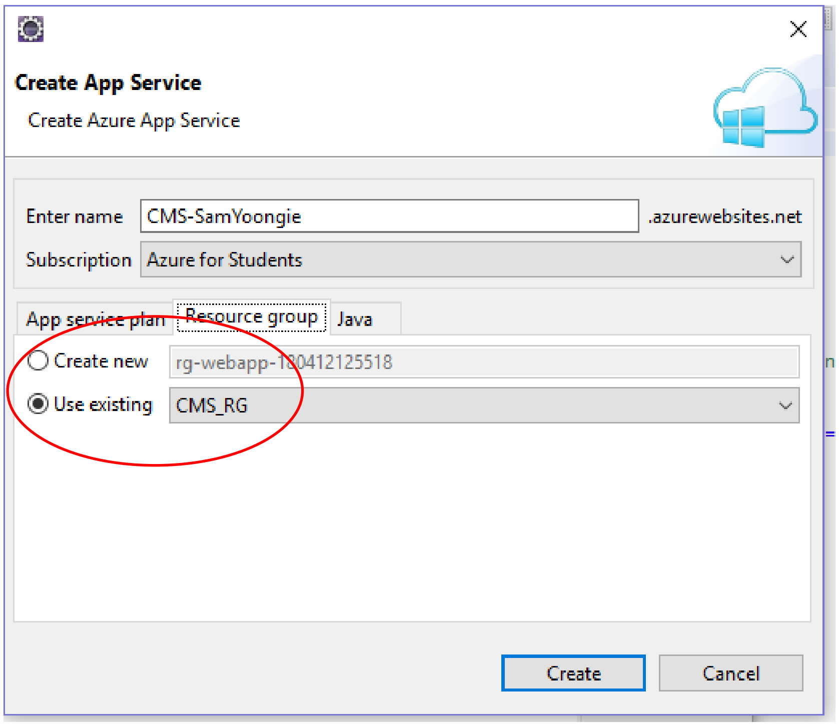
Enter name: CMS-SamYoongie .azurewebsites.net
Subscription: Azure for Students

App service plan | Resource group | Java

( ) Create new: asp-180412125518
Location: Australia East
Pricing tier: Basic_B1
(●) Use existing: CMS-ASP-2
Location: South East Asia
Pricing tier: Standard_S1

App service pricing details

Figure 32: Publish Project



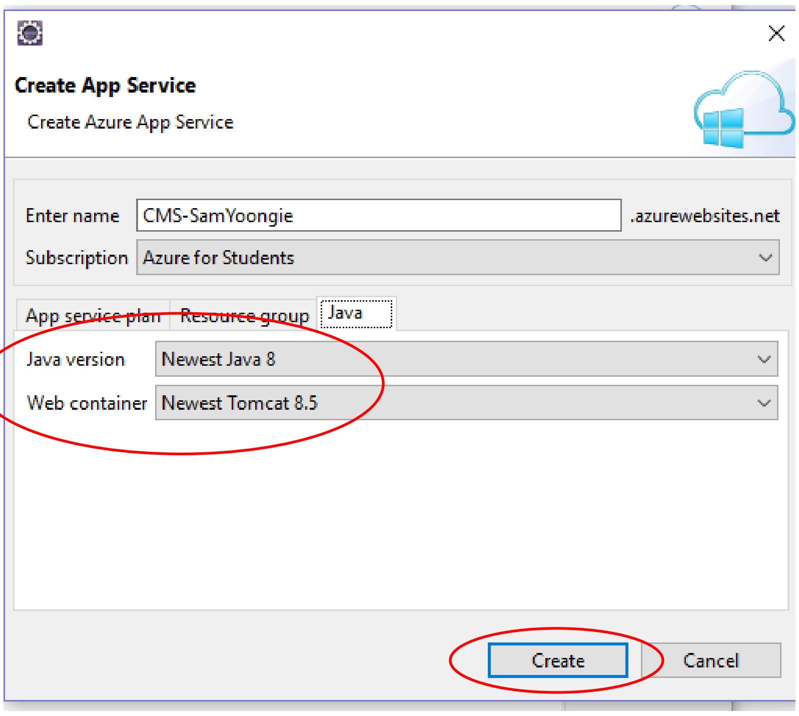
Create App Service
Create Azure App Service

Enter name: CMS-SamYoongie .azurewebsites.net
Subscription: Azure for Students

App service plan | Resource group | Java

( ) Create new: rg-webapp-180412125518
(●) Use existing: CMS_RG

Figure 33: Publish Project



Create App Service

Create Azure App Service

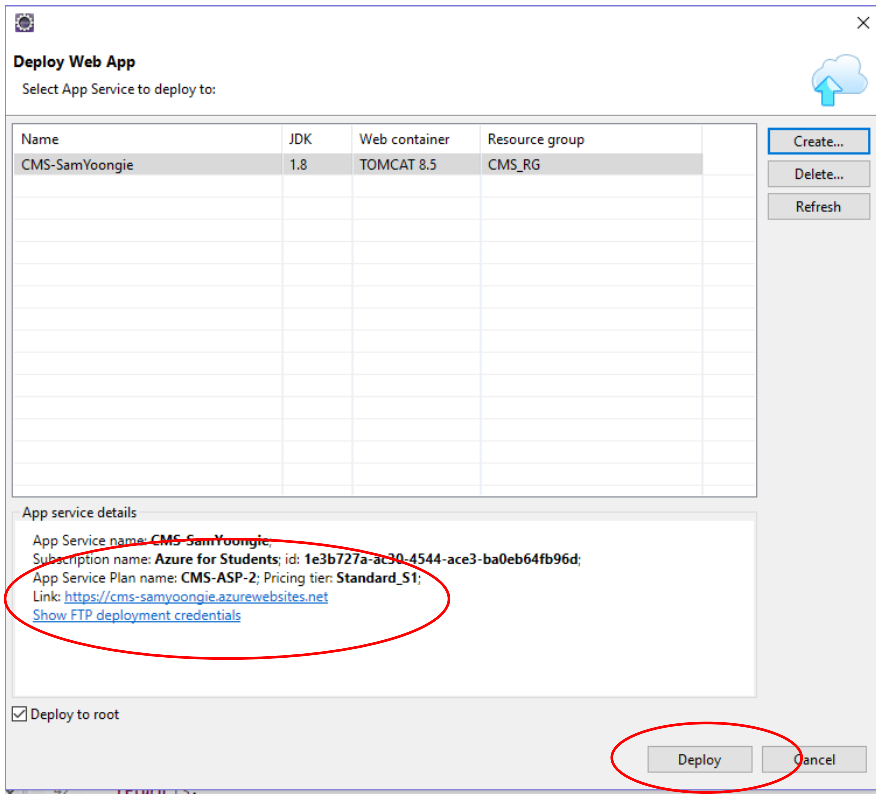
Enter name	CMS-SamYoongie	.azurewebsites.net
Subscription	Azure for Students	

App service plan | Resource group | Java

Java version	Newest Java 8
Web container	Newest Tomcat 8.5

Figure 34: Publish Project

6. Once the web app is ready, click on Deploy button. Now, the web application is now accessible through the link provided.



Deploy Web App

Select App Service to deploy to:

Name	JDK	Web container	Resource group
CMS-SamYoongie	1.8	TOMCAT 8.5	CMS_RG

App service details

App Service name: **CMS-SamYoongie**;
Subscription name: **Azure for Students**; id: **1e3b727a-ac30-4544-ace3-ba0eb64fb96d**;
App Service Plan name: **CMS-ASP-2**; Pricing tier: **Standard_S1**;
Link: https://cms-samyoongie.azurewebsites.net
Show FTP deployment credentials

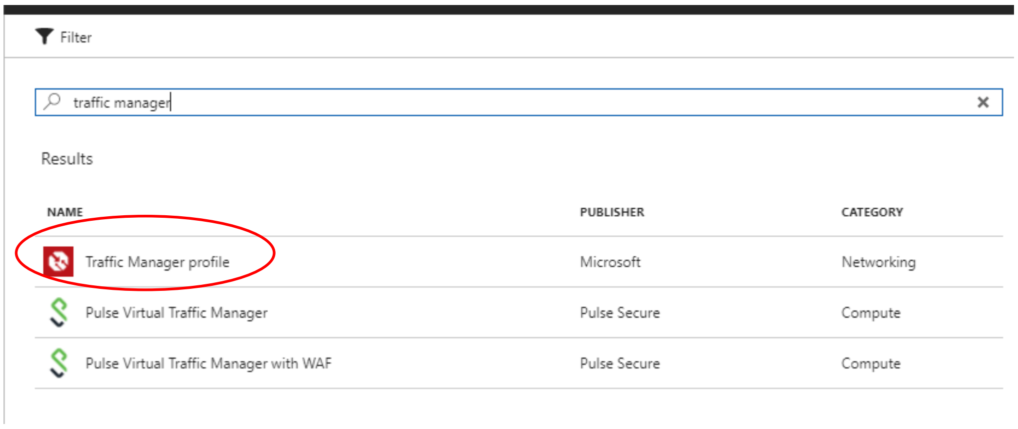
☑ Deploy to root

Figure 35: Publish Project

4.2.4 Configuring Traffic Manager

Traffic Manager was used to navigate the users to the nearest endpoint based on traffic routing method. Whenever the endpoint fails, the traffic manager will direct the user to another endpoint. This is very important in making sure the application has high reliability and availability. It is all handled by traffic manager in which providing automatic failover when an endpoint went down. Besides, traffic manager also navigates the user to the nearest endpoint to reduce the network latency and improve the overall performance of the web application. Hence, traffic manager was implemented for this project. The steps are as followed:

1. Select Traffic Manager profile from the Menu.



Filter

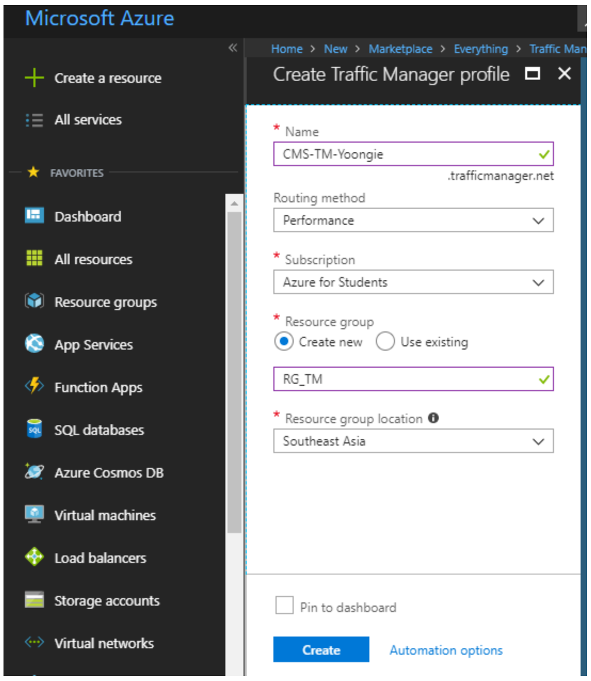
traffic manager

Results

NAME	PUBLISHER	CATEGORY
Traffic Manager profile	Microsoft	Networking
Pulse Virtual Traffic Manager	Pulse Secure	Compute
Pulse Virtual Traffic Manager with WAF	Pulse Secure	Compute

Figure 36: Traffic Manager

2. Enter the required field and click on Create button.



Create Traffic Manager profile

* Name: CMS-TM-Yoongie .trafficmanager.net

Routing method: Performance

* Subscription: Azure for Students

* Resource group: [✓ Create new] Use existing

RG_TM

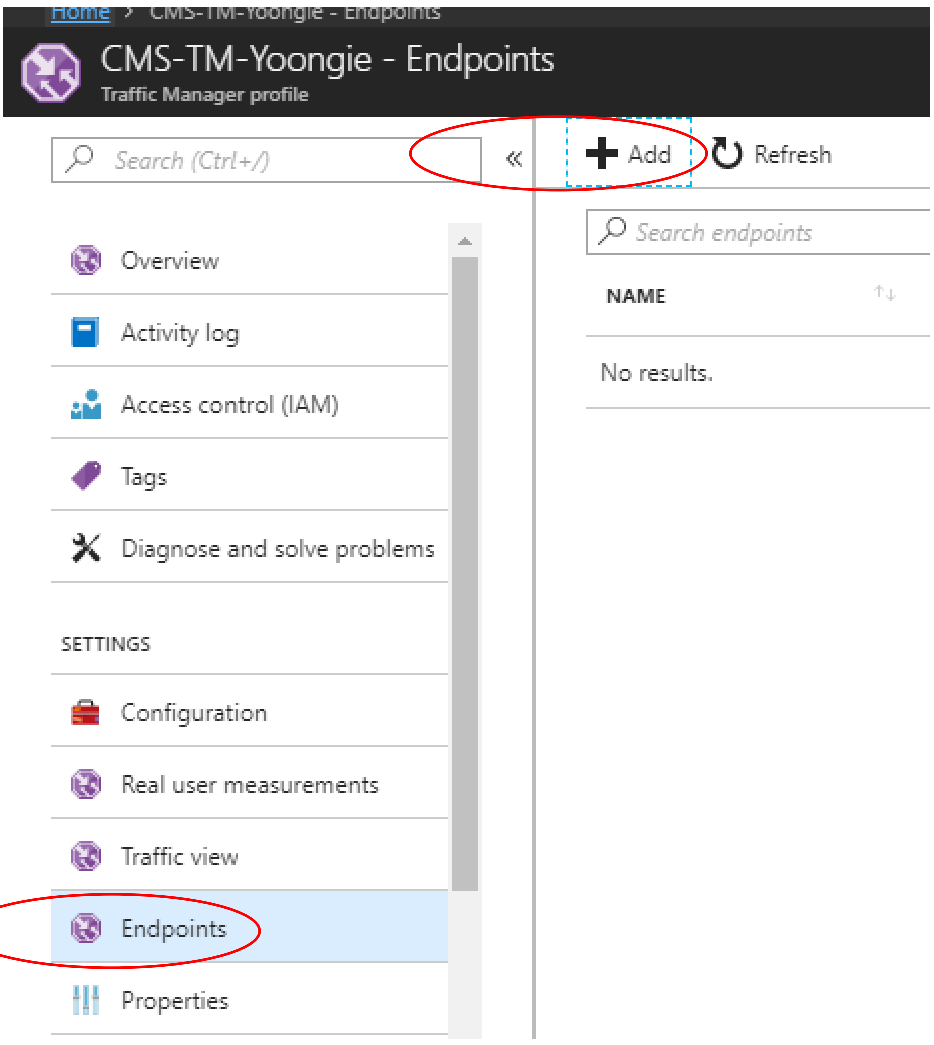
* Resource group location: Southeast Asia

Pin to dashboard

Create | Automation options

Figure 37: Traffic Manager

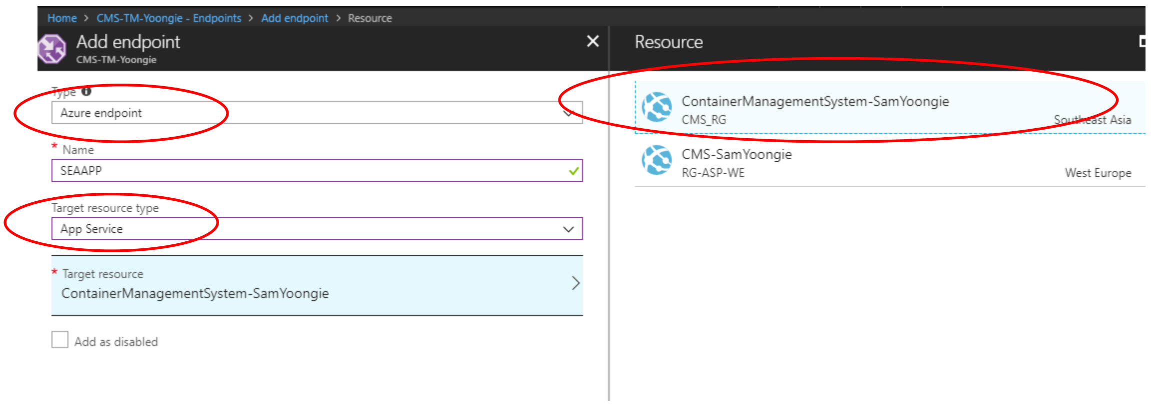
3. Add 2 endpoints in the traffic manager that created earlier. Select endpoint from the menu and click Add button.



No results.

Figure 38: Traffic Manager

4. In this page, make sure “Azure endpoint” is selected and select “App Service” in target resource type. Then, select the application and click on OK button.



Type: Azure endpoint

* Name: SEAAPP

Target resource type: App Service

* Target resource: ContainerManagementSystem-SamYoongie

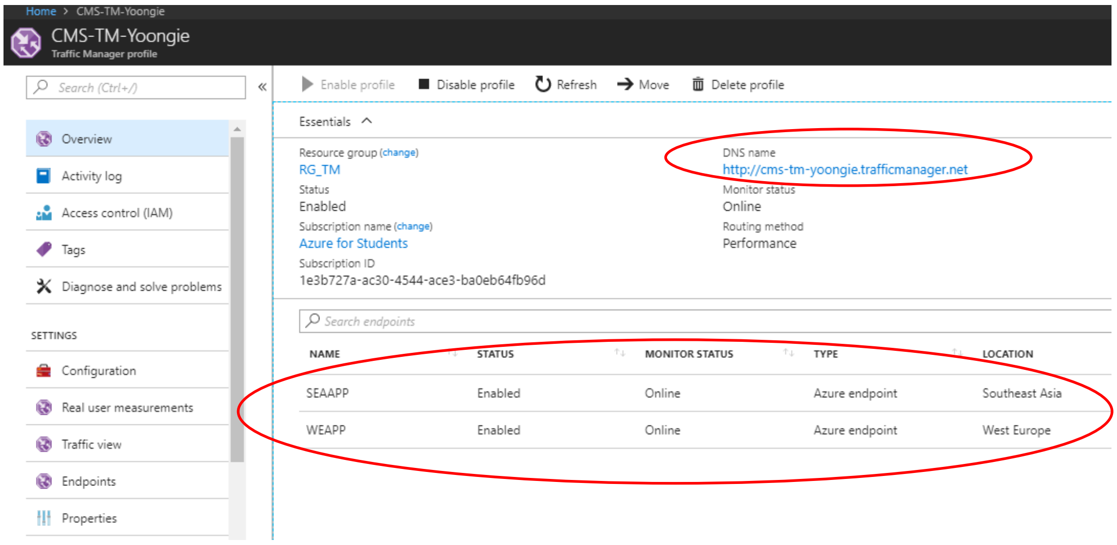
Add as disabled

ContainerManagementSystem-SamYoongie	CMS_RG	Southeast Asia
CMS-SamYoongie	RG-ASP-WE	West Europe

Figure 39: Traffic Manager

5. Do the same steps for the other region.

6. Select configuration for the menu and ensure that “Performance” was chosen as the routing method and click on “Save” button. Performance was chosen as the routing method as to direct the user to other available endpoint when the nearest endpoint is unavailable. After adding the endpoints, the user can browse the web application through the DNS name of the traffic manager.



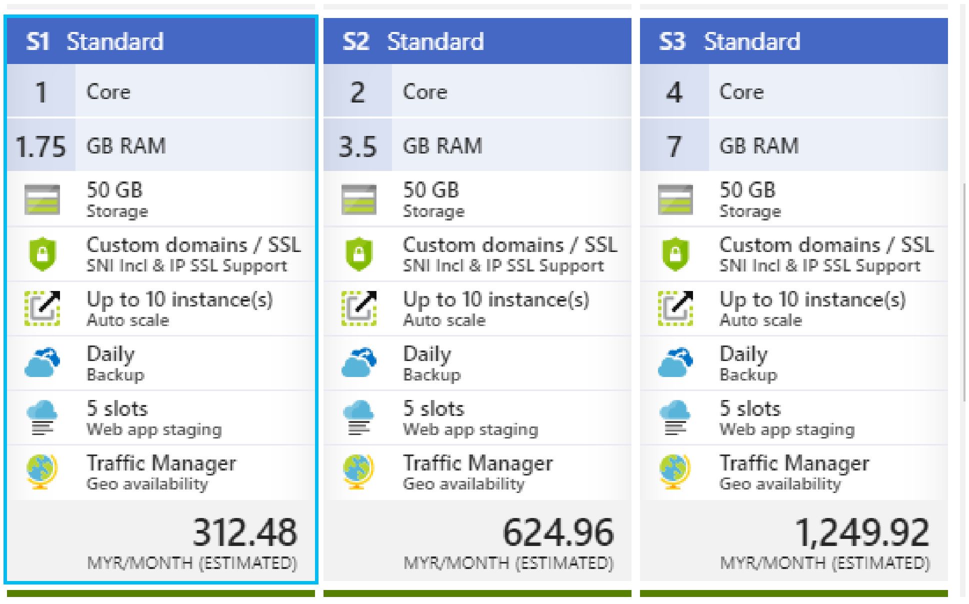
Resource group (change): RG_TM
Status: Enabled
Subscription name (change): Azure for Students
Subscription ID: 1e3b727a-ac30-4544-ace3-ba0eb64fb96d
DNS name: http://cms-tm-yoongie.trafficmanager.net
Monitor status: Online
Routing method: Performance

NAME	STATUS	MONITOR STATUS	TYPE	LOCATION
SEAAPP	Enabled	Online	Azure endpoint	Southeast Asia
WEAPP	Enabled	Online	Azure endpoint	West Europe

Figure 40: Traffic Manager

4.2.5 Application Scaling

Azure cloud environment offers the auto-scaling feature where the resources can be allocated based on need to meet the performance requirement. This is very important for this application as there are high traffic during peak period. Hence, it requires to allocate more resources to handle the traffic as to maintain the availability of the application. By applying auto-scaling feature, no resources will not be wasted because Azure will help to deallocate the resources when the resources are not in use. Therefore, operational cost can be reduced. The system administrator does not require to monitor the application and perform scale in or scale out manually as all this will be automatically handled by Azure.

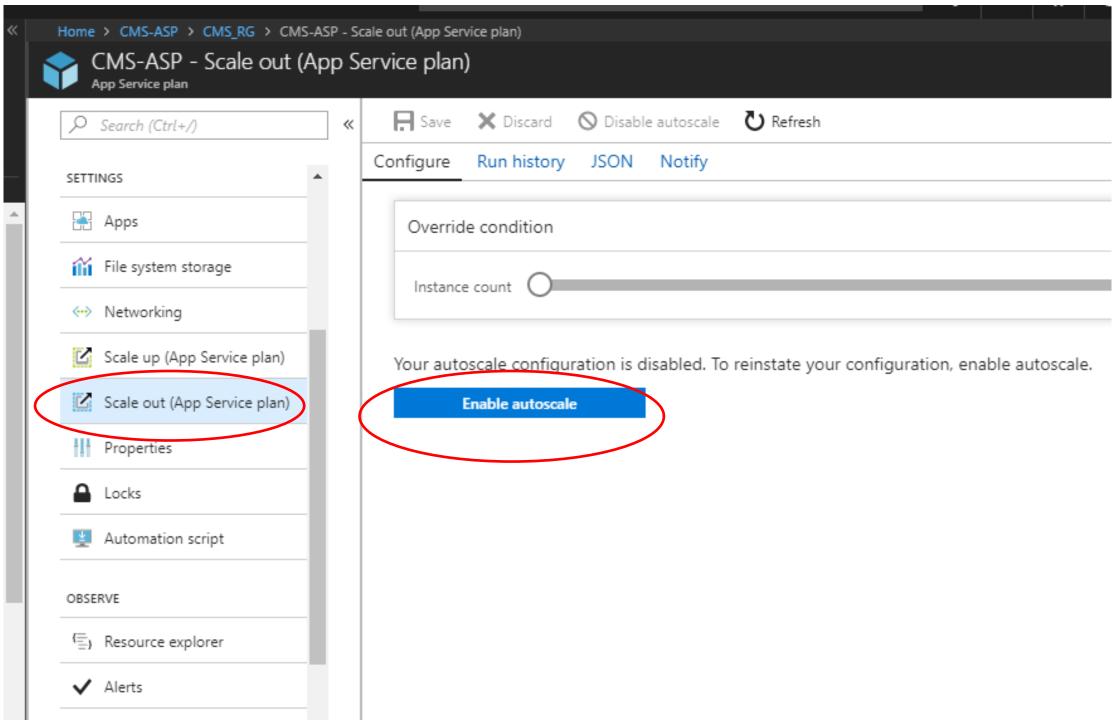


S1 Standard	S2 Standard	S3 Standard
1 Core	2 Core	4 Core
1.75 GB RAM	3.5 GB RAM	7 GB RAM
50 GB Storage	50 GB Storage	50 GB Storage
Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support
Up to 10 instance(s) Auto scale	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 41: App Service Plan

This application uses Standard S1 app service plan because it includes custom domains for the company. It is very important feature because the company will usually use their own domains as to access to the web application. Besides, it offers up to 10 instances and it allows auto-scaling, either scale out or scale in based on the traffic, server load and others. There is a daily backup on the standard plan in which data will not be easily lost during failure and able to recover to the latest backup. In addition, traffic manager which included in Standard Plan is important for web application as this application will be accessed worldwide. Hence, traffic manager is required to redirect the user to the nearest instances as to maintain the performance. As to configure auto-scaling, the steps are below:

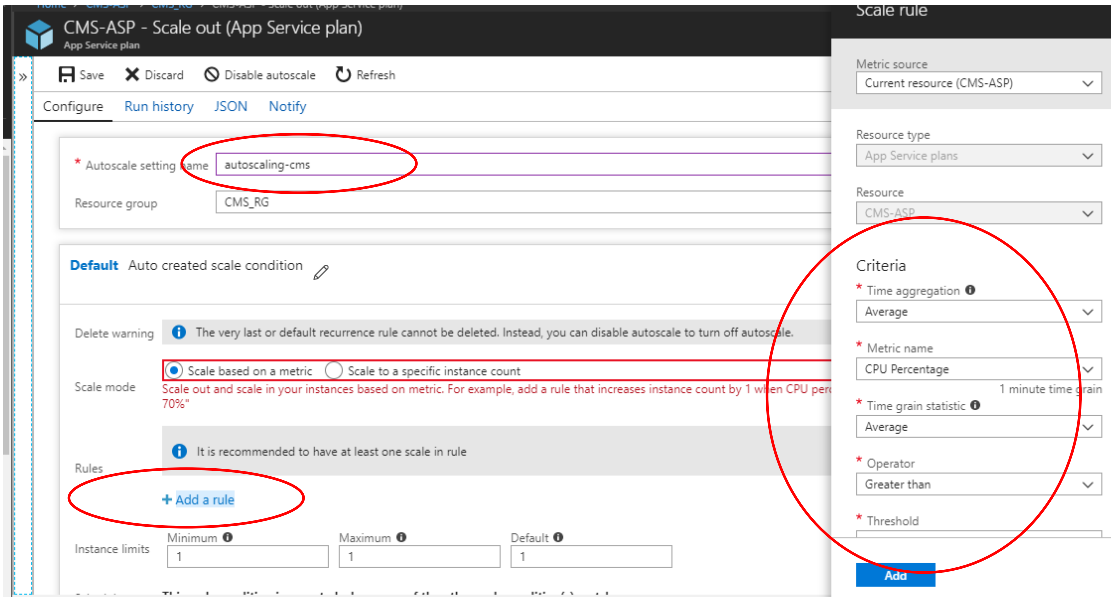
1. In the app service plan, select Scale Out and click on Enable autoscale.



Your autoscale configuration is disabled. To reinstate your configuration, enable autoscale.

Figure 42: Application Scaling

2. Now, add a new rule that state the condition whether to add or remove the instance.



Autoscale setting name: autoscaling-cms

Resource group: CMS_RG

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

Scale out and scale in your instances based on metric. For example, add a rule that increases instance count by 1 when CPU per 70%"

It is recommended to have at least one scale in rule

Instance limits: Minimum 1 Maximum 1 Default 1

Metric source: Current resource (CMS-ASP)

Resource type: App Service plans

Resource: CMS-ASP

Criteria

Time aggregation: Average

Metric name: CPU Percentage

1 minute time grain

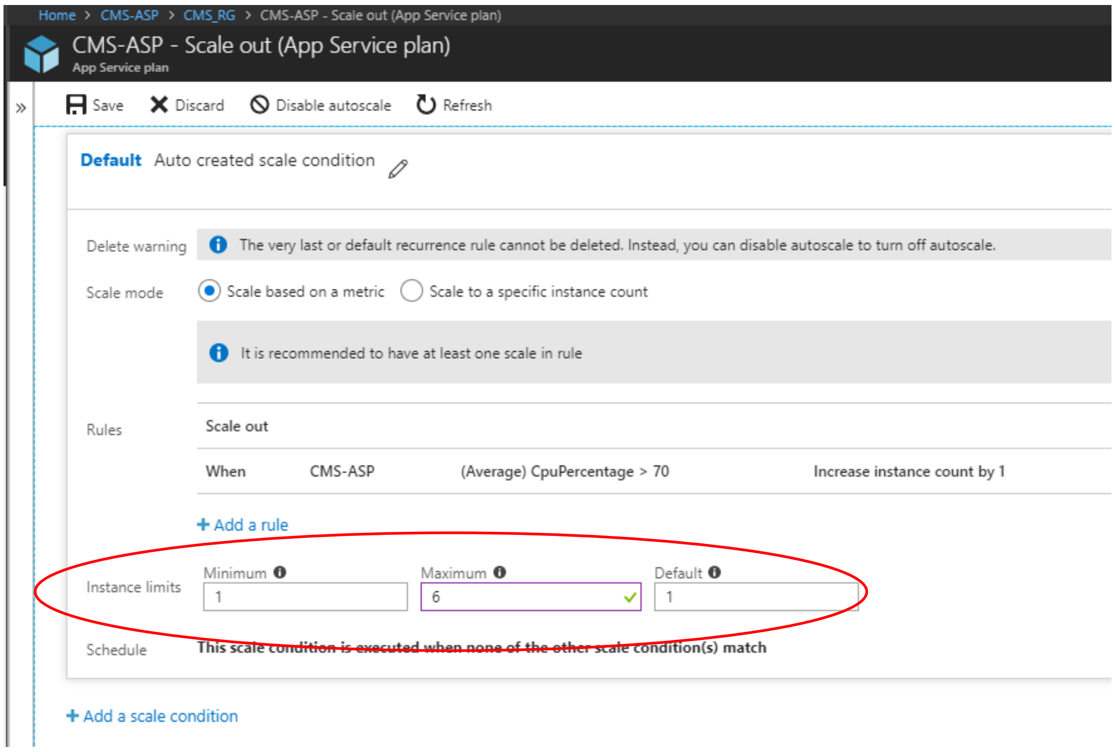
Time grain statistic: Average

Operator: Greater than

Threshold

Figure 43: Application Scaling

3. Once the rule gets created, specify the minimum, maximum and default number of instances that will be used for auto scaling and click on Save button.


Home > CMS-ASP > CMS_RG > CMS-ASP - Scale out (App Service plan)

CMS-ASP - Scale out (App Service plan)
App Service plan

Save | Discard | Disable autoscale | Refresh

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

It is recommended to have at least one scale in rule

Rules:

Scale out			
When	CMS-ASP	(Average) CpuPercentage > 70	Increase instance count by 1

+ Add a rule

Instance limits: Minimum 1, Maximum 6, Default 1

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

+ Add a scale condition

Figure 44: Application Scaling

5 Test Plan & Testing Discussion

5.1 Unit Testing

5.1.1 Navigation Bar

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
1	Navigation to View Agent	Click on View Agent through navigation bar	• Navigate to view agent page and display table	• Navigated to view agent page and table displayed	Pass
2	Navigation to View Booking	Click on View Booking through navigation bar	• Navigate to view booking page and display table	• Navigated to view booking page and table displayed	Pass
3	Navigation to Add Agent	Click on Add Agent through navigation bar	• Navigate to add agent page	• Navigated to add agent page	Pass
4	Navigation to Add Ship	Click on Add Ship through navigation bar	• Navigate to add ship page	• Navigated to add ship page	Pass
5	Navigation to Add Route	Click on Add Route through navigation bar	• Navigate to add route page	• Navigated to add route page	Pass
6	Navigation to Create Schedule	Click on Create Schedule through navigation bar	• Navigate to create schedule page	• Navigated to create schedule page	Pass
7	Navigation to Register Customer	Click on Register Customer through navigation bar	• Navigate to register customer page	• Navigated to register customer page	Pass
8	Navigation to Register Item	Click on Register Item through navigation bar	• Navigate to register item page	• Navigated to register item page	Pass

9	Navigation to Book Vessel	Click on Book Vessel through navigation bar	• Navigate to book vessel page	• Navigated to book vessel page	Pass
10	Navigation to Home	Click on Logout through navigation bar	• Navigate to Home	• Navigated to Home	Pass

5.1.2 Home

Test Case #	Description	Test Step	• Expected Output	Actual Output	Remarks
11	Display Login pop up box	Click on Login through navigation bar	• Login pop up box being displayed	• Login pop up box displayed	Pass

5.1.3 Add Agent

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
12	Add Agent	1. Fill in all the required field ➢ Username and name must be unique ➢ Password and confirm password must be the same 2. Click on Add button 3. Check database	• Database get updated • Display “Agent added successfully” message	• Database get updated • “Agent added successfully” message was displayed	Pass

13	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Add button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass
14	Input registered name	1. Fill in the required text field 2. Input registered name 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Name or username has been registered! Please try again." message • No changes on database	• Form unable to get submitted • "Name or username has been registered! Please try again." message was displayed • No changes on database	Pass
15	Input registered username	1. Fill in the required text field 2. Input registered username 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Name or username has been registered! Please try again." message • No changes on database	• Form unable to get submitted • "Name or username has been registered! Please try again." message was displayed • No changes on database	Pass
16	Input different password and confirm password	1. Fill in the required text field 2. Input different password and confirm password 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Password are not same!" message • No changes on database	• Form unable to get submitted • "Password are not same!" message was displayed • No changes on database	Pass

5.1.4 Add Route

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
17	Add Route	1. Fill in all the required field 2. Click on Add button 3. Check database	• Database get updated • Display “Route added successfully!” message	• Database get updated • “Route added successfully!” message was displayed	Pass
18	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Add button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass
19	Input added departure destination and arrival destination	1. Fill in the required text field 2. Input added departure destination and arrival destination 3. Click on Add button 4. Check database	• Form unable to get submitted • Display “Route has been added. Please try again” message • No changes on database	• Form unable to get submitted • “Route has been added. Please try again” message was displayed • No changes on database	Pass

5.1.5 Add Ship

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
20	Add Ship	1. Fill in all the required field 2. Click on Add button 3. Check database	• Database get updated • Display "Ship added successfully!" message	• Database get updated • "Ship added successfully!" message was displayed	Pass
21	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Add button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass

5.1.6 View Booking

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
22	View Booking	1. Select View Booking through navigation bar	• Booking that made in the past are being retrieved from database • Display in a table	• Booking that made in the past are being retrieved from database • Table was displayed	Pass
23	Filter Booking	1. Select desired month	• Only booking that made on that month will be displayed	• Only booking that made on that day was displayed	Pass

5.1.7 View Agent

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
24	View Agent	1. Select View Agent through navigation bar	• All the agents details being retrieved from database • Display in a table	• All the agents details are being retrieved from database • Table was displayed	Pass

5.1.8 Top Up Credit

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
25	Top up Credit	1. Fill in all the required field 2. Click on Top Up button 3. Check database	• Database get updated • Display "Credit top up successfully!" message	• Database get updated • "Credit top up successfully!" message was displayed	Pass
26	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Top Up button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass

5.1.9 Edit Agent Profile

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
27	Edit Agent Profile	1. Fill in all the required field 2. Make changes on text field 3. Click on Update Profile button 4. Check database	• Database get updated • Display "Profile update successfully!" message	• Database get updated • "Profile update successfully!" message was displayed	Pass
28	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Update Profile button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass
29	Input different password and confirm password	1. Fill in the required text field 2. Input different password and confirm password 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Confirm password and password is not same. Please try again!" message • No changes on database	• Form unable to get submitted • "Confirm password and password is not same. Please try again!" message was displayed • No changes on database	Pass

5.1.10 Create Schedule

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
30	Create Schedule	1. Select ship, route, departure date and arrival date 2. Click on Add button 3. Check database	• Database get updated • Display "Created Schedule successfully!" message	• Database get updated • "Created Schedule successfully!" message was displayed	Pass
31	Leave blank field in the form	1. Leave one or more field blank 2. Click on Add button 3. Check database	• Display "Please select the date!" message • Form unable to get submitted • No changes on database	• Display "Please select the date!" message was displayed • Form unable to get submitted • No changes on database	Pass
32	Select departure date earlier than today	1. Fill in the required text field 2. Select departure date earlier than today 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Invalid date. Please try again!" message • No changes on database	• Form unable to get submitted • "Invalid date. Please try again!" message was displayed • No changes on database	Pass
33	Choose departure or arrival date that added before	1. Select departure date or arrival date that occupied by other schedule 2. Click on Add button 3. Check database	• Form unable to get submitted • Display "Ship is not available for that period. Please try again!" message • No changes on database	• Form unable to get submitted • "Ship is not available for that period. Please try again!" message was displayed. • No changes on database	Pass

5.1.11 Register Customer

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
34	Register Customer	1. Fill in all the required field ➢ Name must be unique 2. Click on Add button 3. Check database	• Database get updated • Display "Customer Registered successfully" message	• Database get updated • "Customer Registered successfully" message was displayed	Pass
35	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Add button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass
36	Input registered name	1. Fill in the required text field 2. Input registered name 3. Click on Add button 4. Check database	• Form unable to get submitted • Display "Name has been registered! Please try again!" message • No changes on database	• Form unable to get submitted • "Name has been registered! Please try again!" message was displayed • No changes on database	Pass

5.1.12 Login

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
37	Login with valid username and password	1. Fill in valid username and password 2. Click on Login button	• User login successfully • Navigate to Portal page	• User login successfully • Navigated to Portal page	Pass
38	Login with invalid username and password	1. Fill in invalid username and password 2. Click on Login button	• User login fail • Display "Invalid username or password" message	• User login fail • Display "Invalid username or password" message was displayed	Pass
39	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Login button	• Form unable to get submitted • User Login fail	• Form unable to get submitted • User login fail	Pass

5.1.13 Register Item

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
40	Register Item	1. Fill in all the required field 2. Click on Add button 3. Check database	• Database get updated • Display "Item registered successfully!" message	• Database get updated • "Item registered successfully!" message was displayed	Pass

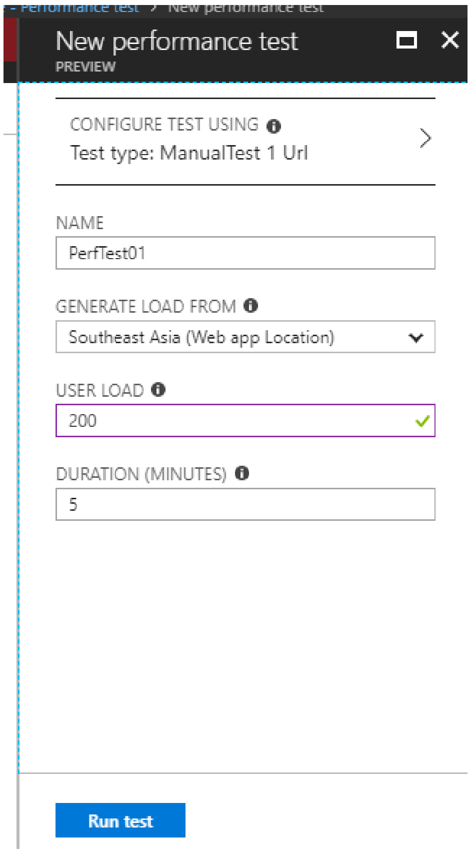
41	Leave blank field in the form	1. Fill in the text field 2. Leave one or more field blank 3. Click on Add button 4. Check database	• Form unable to get submitted • No changes on database	• Form unable to get submitted • No changes on database	Pass

5.1.14 Book Vessel

Test Case #	Description	Test Step	Expected Output	Actual Output	Remarks
42	Book Vessel	1. Select required field 2. Click on Add button 3. Check database	• Database get updated • Display "Booked vessel successfully!" message	• Database get updated • "Booked vessel successfully!" message was displayed	Pass
43	Not selecting any item	1. Do not select any item 2. Click on Add button 3. Check database	• Display "No item selected! Please try again." message • Form unable to get submitted • No changes on database	• Display "No item selected! Please try again." message was displayed • Form unable to get submitted • No changes on database	Pass
44	Choose fully occupied ship	1. Choose fully occupied ship 2. Click on Add button 3. Check database	• Form unable to get submitted • Display "The chosen ship is full. Please try another ship!" message • No changes on database	• Form unable to get submitted • "The chosen ship is full. Please try another ship!" message was displayed • No changes on database	Pass

5.2 Performance Testing

Azure offers performance tools which simulating certain number of users visiting the web application. It is an important tool to check whether the current service plan supports large number of users. Hence, performance testing of CMS was conducted through the performance tools. The user load will be tested are 200, 400, 600 and 800. Below are the details of the testing which include the number of user load, testing duration and the location where the load generate from:



CONFIGURE TEST USING
Test type: ManualTest 1 Url

NAME
PerfTest01

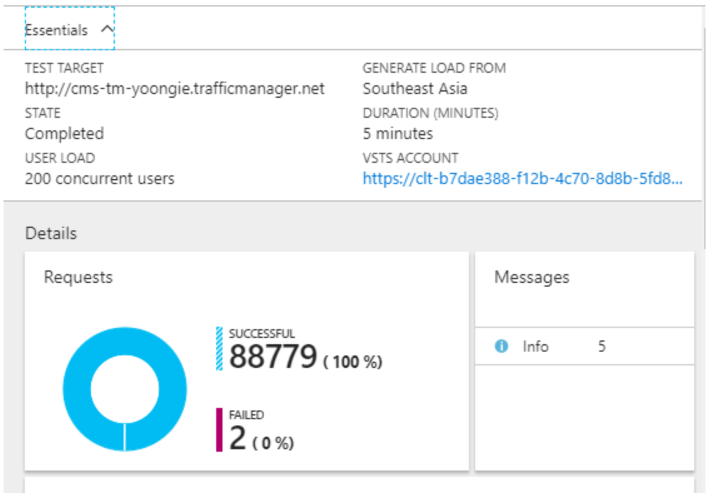
GENERATE LOAD FROM
Southeast Asia (Web app Location)

USER LOAD
200

DURATION (MINUTES)
5

Figure 45: Performance test

By having 200 user loads for a duration of 5 minutes and uses SEA server, the results are as below:



Essentials

TEST TARGET	GENERATE LOAD FROM
http://cms-tm-yoongie.trafficmanager.net	Southeast Asia
STATE	DURATION (MINUTES)
Completed	5 minutes
USER LOAD	VSTS ACCOUNT
200 concurrent users	https://clt-b7dae388-f12b-4c70-8d8b-5fd8...

Details

Requests

SUCCESSFUL 88779 (100 %)

FAILED 2 (0 %)

Messages

Info 5

Figure 46: Result for Performance Test 1

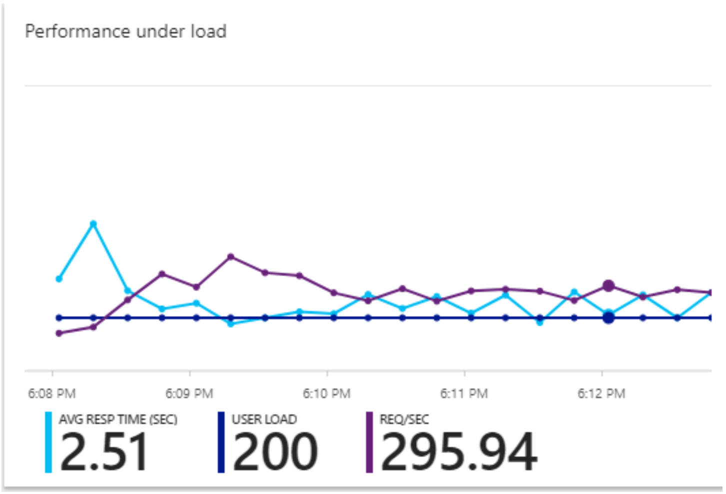


Figure 47: Result for Performance Test 1

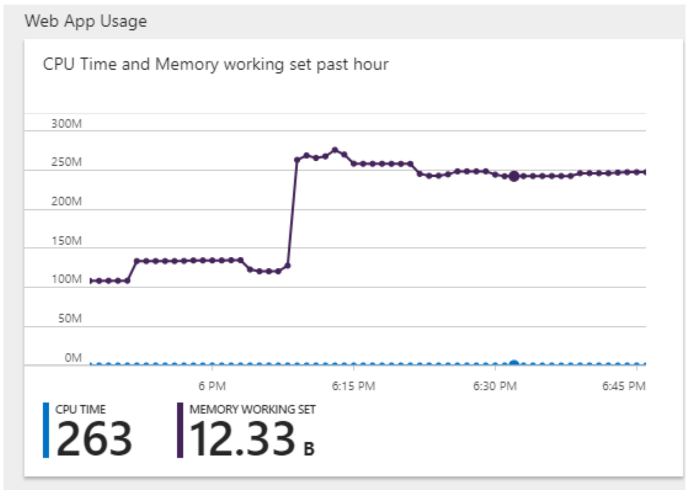
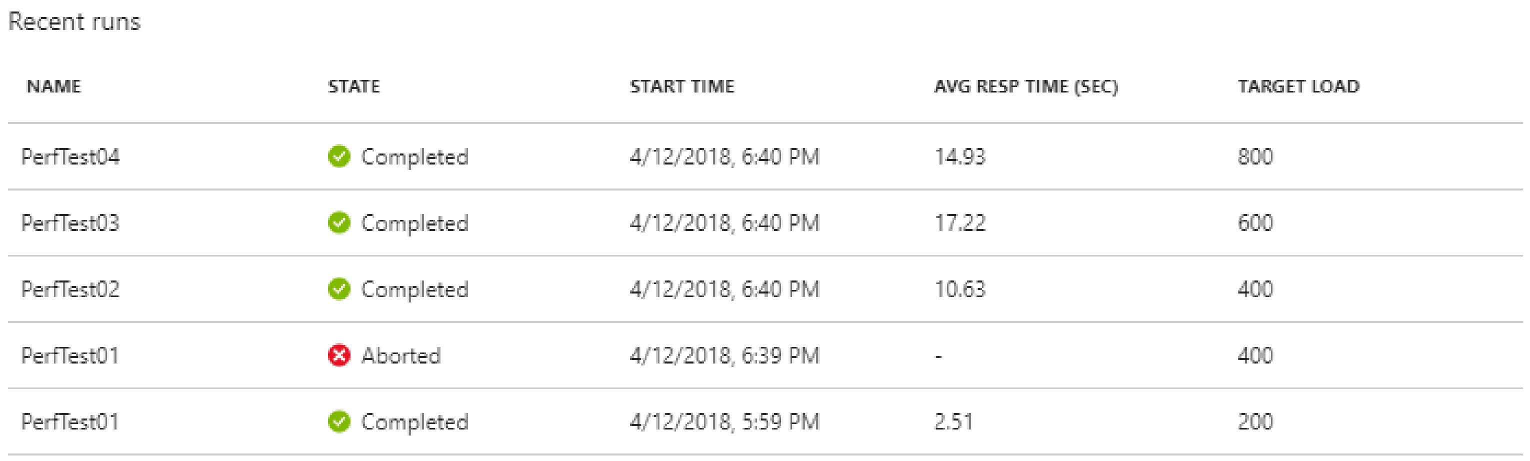


Figure 48: Result for Performance Test 1


Recent runs

NAME	STATE	START TIME	AVG RESP TIME (SEC)	TARGET LOAD
PerfTest04	✓ Completed	4/12/2018, 6:40 PM	14.93	800
PerfTest03	✓ Completed	4/12/2018, 6:40 PM	17.22	600
PerfTest02	✓ Completed	4/12/2018, 6:40 PM	10.63	400
PerfTest01	✗ Aborted	4/12/2018, 6:39 PM	-	400
PerfTest01	✓ Completed	4/12/2018, 5:59 PM	2.51	200

Figure 49: Results of four performance test

5.2.1 Summary of Performance Testing

Test Case	User Load	Success Rate	Fail Rate	Average Response Time (second)	Request per second	Pricing Tier
PerfTest01	200	100%	2%	2.5	295.94	Standard 1
PerfTest02	400	65.97%	34.03%	10.63	98.35	
PerfTest03	600	78.68%	21.32%	17.22	109.26	
PerfTest04	800	74.59%	25.41%	14.93	159.6	

As seen from the table above, it can clearly have showed that the success rate reduced and fail rate increased when increasing the user load. In the first test, the application was visited by 200 concurrent users in which there 100% success rate with 2.5 second of response time. When the number of concurrent users increased by 200 in second test, the success rate has dropped to 65.97% with 10.63 seconds. Then, 200 concurrent users were added for the third test which results in 78.68% of success rate and 17.22 second to response the request. Lastly, the forth test was conducted by 800 concurrent users and has 74.59% of success rate with 14.93 second of response time. As to conclude, the average response time was increased when the user load increases. The web application requires more time to handle the response as the number of response grows. Hence, a higher tier of service plan shall be used to ensure the performance of the web application.

5.3 Managed Database

As the popularity of the cloud computing increases, there are more and more managed services get introduced. These services help the developer to maintain high reliability and availability of the system as the developer does not required to manage the database or other components by themselves. It is a platform as a service (Paas) in which the service is managed by the service provider and the user use the service remotely. Cloud service provider such as Microsoft, Google and Amazon have introduced various services as to provide convenient to the user.

For instance, one of the popular managed services provided by Microsoft is the managed database which is the Azure SQL database. The Azure SQL database that offered by Microsoft has no much different with the normal SQL database in which it has all the SQL server features. The only difference between them is the Azure SQL database can be accessed by various machine as it was hosted on the cloud while the normal SQL database are required to be hosted in physical servers which may bring problem.

Furthermore, the managed database is cost-effective with the least monitoring required will can help to reduce the operational cost. Hence, it is highly recommended to use managed database provided by cloud service provider as these databases are handled by expertise and database fault can be reduced to the lowest. In addition, ongoing administration activities such as managing virtual machines, operating system can be reduced because Azure done all these by themselves.

Azure SQL database owns various characteristics such as it is suitable for development of a cloud-based application which has time constraints. Besides, it is suitable for project that should have high availability of database and upgrade for database. By applying Azure SQL database, the hardware and administrative costs can be reduced to the max.

There are various reasons supported that Azure SQL database shall be used for the database of the project which includes cost, administration and service level agreement.

As for cost, there are different service price such as basic, standard, premium and others. All these different service price serves different performance. If the database is huge and shall support more concurrent users, a higher price service plan shall be chosen. This is because higher price service plan offers more database instances and bigger volume of storage.

Furthermore, administration activities such as configuring database engine or server is no longer needed as it will be handled by Azure. The user can focus more on daily business operation rather than administering the databases.

As guaranteed by Azure, the Azure SQL database has a high availability of 99.99% in which there is only 0.01% that the database will be down. Hence, it is suitable for project which emphasize on the availability of databases.

6 Conclusion

As to conclude, the project was developed successfully onto the cloud which is the Azure platform. There are two app service which one is being deployed in West Europe while another one is being deployed in South East Asia. This is to ensure high reliability and availability whenever any of the region went down. Throughout the project, the developer has gained a lot of knowledge on cloud computing as in its benefits, easiness and others. The developer gets a better understanding on how to deploy the web application through Azure. Other than just the knowledge of deployment of application, the developer also understood the different configuration such as traffic manager, auto-scale features and others. The developer believed that all these knowledges will bring benefits in the future.

7 References

Cliff, 2018. *10 Microsoft Azure Business Benefits in One Convenient List.* [Online]
Available at: https://www.redpixie.com/blog/top-ten-microsoft-azure-business-benefits
[Accessed 20 March 2018].

Haldane, H., 2015. *What are the BENEFITS of Microsoft Azure?.* [Online]
Available at: http://www.saasplaza.com/blog/what-are-benefits-microsoft-azure
[Accessed 20 march 2018].

Kumar, R., 2017. *Microsoft extends Azure managed database services with introduction of MySQL and PostgreSQL.* [Online]
Available at: https://azure.microsoft.com/en-us/blog/microsoft-extends-azure-managed-database-services-with-introduction-of-mysql-and-postgresql/
[Accessed 20 March 2018].

Microsoft, 2017. *What Is Scalability?.* [Online]
Available at: https://docs.microsoft.com/en-us/biztalk/core/what-is-scalability
[Accessed 20 March 2018].

Microsoft, 2018. *Create your first Java web app in Azure.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-java
[Accessed 20 March 2018].

Serhiy, 2016. *How to Build a Scalable Web Application?.* [Online]
Available at: https://www.romexsoft.com/blog/scalable-website/
[Accessed 20 March 2018].