

**DESIGNING AND DEVELOPING CLOUD APPLICATIONS**

**CT071-3-5-3-DDAC**

**INDIVIDUAL ASSIGNMENT**

|  |  |
| --- | --- |
| **DDAC Assignment** | |
| **Module Name:** | DESIGNING AND DEVELOPING CLOUD APPLICATIONS |
| **Module Code:** | CT071-3-5-3-DDAC |
| **Student Name:** | Chan Kuok Hong |
| **TP Number:** | TP034717 |
| **Intake Code:** | UC3F1706SE |
| **Lecturer Name:** | DR. KALAI ANAND RATNAM |
| **Date Submission:** | 13rd April 2018 |

# Acknowledgement

Firstly, I am very grateful of having this opportunity to complete this designing & Developing Cloud Applications assignment within the time given by our lecturer Dr. Kalai. I am sincerely thankful for our lecturer Dr. Kalai, who also an expertise in cloud computing for the guidance and encouragement in finishing this assignment, also for teaching us regarding to cloud computing knowledge. Next, my gratitude is extended to my friends and respondents for the support and willingness to provide his/her precious opinions on the topic.

# Table of Contents

[Acknowledgement 2](#_Toc511372306)

[Table of Contents 3](#_Toc511372307)

[Table of Figures 4](#_Toc511372308)

[1.0 Introduction 5](#_Toc511372309)

[1.1 Project Background 5](#_Toc511372310)

[1.2 Objective and Scope 5](#_Toc511372311)

[1.3 Requirement Specifications 5](#_Toc511372312)

[1.4 Summary of Major Functions and Solution Contents 6](#_Toc511372313)

[2.0 Project Plan 7](#_Toc511372314)

[3.0 Design 9](#_Toc511372315)

[3.1 Cloud Design Patterns 9](#_Toc511372316)

[3.2 Implementation Architectural Diagrams 11](#_Toc511372317)

[3.2.1 The Surveys Application Architecture 11](#_Toc511372318)

[3.2.2 Geo-replicating relational databases 12](#_Toc511372319)

[3.3 Design considerations 12](#_Toc511372320)

[3.4 Modelling 13](#_Toc511372321)

[3.4.1 Use case Diagram 13](#_Toc511372322)

[3.4.2 Sequence Diagram 14](#_Toc511372323)

[3.4.3 Data Modelling 20](#_Toc511372324)

[4.0 Implementation 21](#_Toc511372325)

[4.1 Application Development 21](#_Toc511372326)

[4.2 Azure Publishing 29](#_Toc511372327)

[4.3 Application Scaling 33](#_Toc511372328)

[4.3.1 Reliability & Performance 34](#_Toc511372329)

[4.4 Testing 35](#_Toc511372330)

[4.4.1 Unit Testing 35](#_Toc511372331)

[4.4.2 Performance Testing 40](#_Toc511372332)

[4.4.3 Analysis 41](#_Toc511372333)

[4.5 Managed Databases 42](#_Toc511372334)

[5.0 Conclusion 44](#_Toc511372335)

[6.0 References 45](#_Toc511372336)

# Table of Figures

[Figure 1 - Project plan - Gantt Chart 7](#_Toc511371434)

[Figure 2 Project Plan – Gantt Chart 8](#_Toc511371435)

[Figure 3 - Event Sourcing Pattern Source from (Microsoft Azure, 2017) 9](#_Toc511371436)

[Figure 4 The Surveys Application Architecture Source from (Microsoft, 2018) 11](#_Toc511371437)

[Figure 5 Geo-replicating from one region to another region Source from (Waight, 2014) 12](#_Toc511371438)

[Figure 6 – Use case diagram 13](#_Toc511371439)

[Figure 7 Login - Sequence diagram 14](#_Toc511371440)

[Figure 8 Register as a new user - Sequence diagram 15](#_Toc511371441)

[Figure 9 Forgot Password – Sequence diagram 16](#_Toc511371442)

[Figure 10 View Schedule - Sequence diagram 17](#_Toc511371443)

[Figure 11 Place booking - Sequence diagram 18](#_Toc511371444)

[Figure 12 View bookings – Sequence diagram 19](#_Toc511371445)

[Figure 13 Data Modelling 20](#_Toc511371446)

[Figure 14 Cookies in Web.config 21](#_Toc511371447)

[Figure 15 Login Screen 21](#_Toc511371448)

[Figure 16 Solution Explorer – Part 1 22](#_Toc511371449)

[Figure 17 – User Model – UserMetatdata 23](#_Toc511371450)

[Figure 18 – Solution Explorer Part 2 - Views 23](file:///C:\Users\quiet\Desktop\Final%20Sem\DDAC\Final%20Document%20DDAC-TP034717%20Chan%20Kuok%20Hong\DDAC-Assignment-TP034717.docx#_Toc511371451)

[Figure 19 Login cshtml 24](#_Toc511371452)

[Figure 20 - Design stylesheet in cshtml 24](#_Toc511371453)

[Figure 21 Screen Design home page 26](#_Toc511371454)

[Figure 22 Home screen 27](#_Toc511371455)

[Figure 23 - No matching records found 27](#_Toc511371456)

[Figure 24 - Schedule booking - Cargo, Container Type 28](#_Toc511371457)

[Figure 25 – SQL Server Object Explorer - Database 28](#_Toc511371458)

[Figure 26 Step 1: Publish through web application: ASP.NET 29](file:///C:\Users\quiet\Desktop\Final%20Sem\DDAC\Final%20Document%20DDAC-TP034717%20Chan%20Kuok%20Hong\DDAC-Assignment-TP034717.docx#_Toc511371459)

[Figure 27 - Application which have already instance 30](#_Toc511371460)

[Figure 28 Publish web app by clicking “creating new profile” 30](#_Toc511371461)

[Figure 29 App Service for resource group, subscription, name for the app 31](#_Toc511371462)

[Figure 30 Updating DB Entities for the web app 32](#_Toc511371463)

[Figure 31 – Standard plan 33](#_Toc511371464)

[Figure 32 - Auto scaling 34](#_Toc511371465)

[Figure 33 New Performance Testing 40](file:///C:\Users\quiet\Desktop\Final%20Sem\DDAC\Final%20Document%20DDAC-TP034717%20Chan%20Kuok%20Hong\DDAC-Assignment-TP034717.docx#_Toc511371466)

[Figure 34 Performance test – Part 1 40](#_Toc511371467)

[Figure 35 - Performance test - Part 2 41](#_Toc511371468)

[Figure 36 Saas, PaaS and laaS Source from (Microsoft Azure, 2018) 42](#_Toc511371469)

# 1.0 Introduction

## 1.1 Project Background

Maersk Line is one the global container division and the largest operating unit of the A.P Moller – Maersk Group, a Danish business conglomerate. Maersk Line also consider as the one of the largest container shipping company that ships containers or cargo over the globe. However, as business grows, it is difficult to handle all the cargo at a time. Without appropriate data centers or data servers that operate over the organizational business solution, the business will face denial-of-service (DOS) attacks, which damage the server internally, and server failure due to not manageable business resources and so on. Therefore, Maersk Line has recruited us to deploy the Maersk Line web application on Azure solution to increate organizational flexibility and business performance. The system will be uploaded into GitHub once development has been completed, also hosted on Azure to have continuous maintainability over the web application.

## 1.2 Objective and Scope

* System deployed is scalable to meet the needs of demands during peak solutions therefore reducing system downtime and system data failures.
* To provide a reliable and faster interactive system for the users in terms of usability and functionality of the system.
* To provide accurate business solutions that architecturally demonstrate entire booking process by utilizing Microsoft Azure as the public cloud platform for the system.
* To reduce data server latency issues therefore delivering an exceptional level of automation system and removing human error.

## 1.3 Requirement Specifications

**Maintainability** – The system ensures the software products can be analyzed, changed and tested by the developers. The system will need to have constant updates to avoid unexpected effects from modifications to the software. The sub characteristics of maintainability are modularity, reusability, analyzability, modifiability, and testability of the system.

**Benefits:**

• Maintainable software allow developer to make a fix that prevents a bug from occurring in future.

• Software maintainability ensures system availability by reducing system failures.

• Any faults existed in the software application can be easily diagnosed and corrected, thus reducing system downtime and meeting delivery schedules.

• Enhance fielded software much efficient and at lower cost.

**Usability** – System provides the ease of a function demonstrates by the system that stand on the effort needed for use in terms of user-friendliness to the user interfaces.

**Benefits**:

• Eliminate user frustration and increase user satisfaction to the mobile application

• Ensure compatibility with user interface standard

• Reduced development time and costs

• Enhance performance of the system

The developer is required to design and develop a business web solution that meets the business criteria as follows:

* To build a booking system in terms of user interactive and user-friendliness of the system for the entire booking process from schedule search to booking confirmation.

## 1.4 Summary of Major Functions and Solution Contents

1. Design and Develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App).

2. The Web app will consume a SQL Database, which is needed to deploy onto the Microsoft Azure.

3. The web application will allow guest to login, register and retrieve own password through receiving verification email (Forgot Password).

4. The web application will also allow registered user to search shipment schedule based on (from location and to location). User will be needed to select a schedule once schedule has been searched.

5. Booking information will be included the shipment schedule and text input information (cargo, container type).

6. User is expected to view or add new booking information.

# 2.0 Project Plan

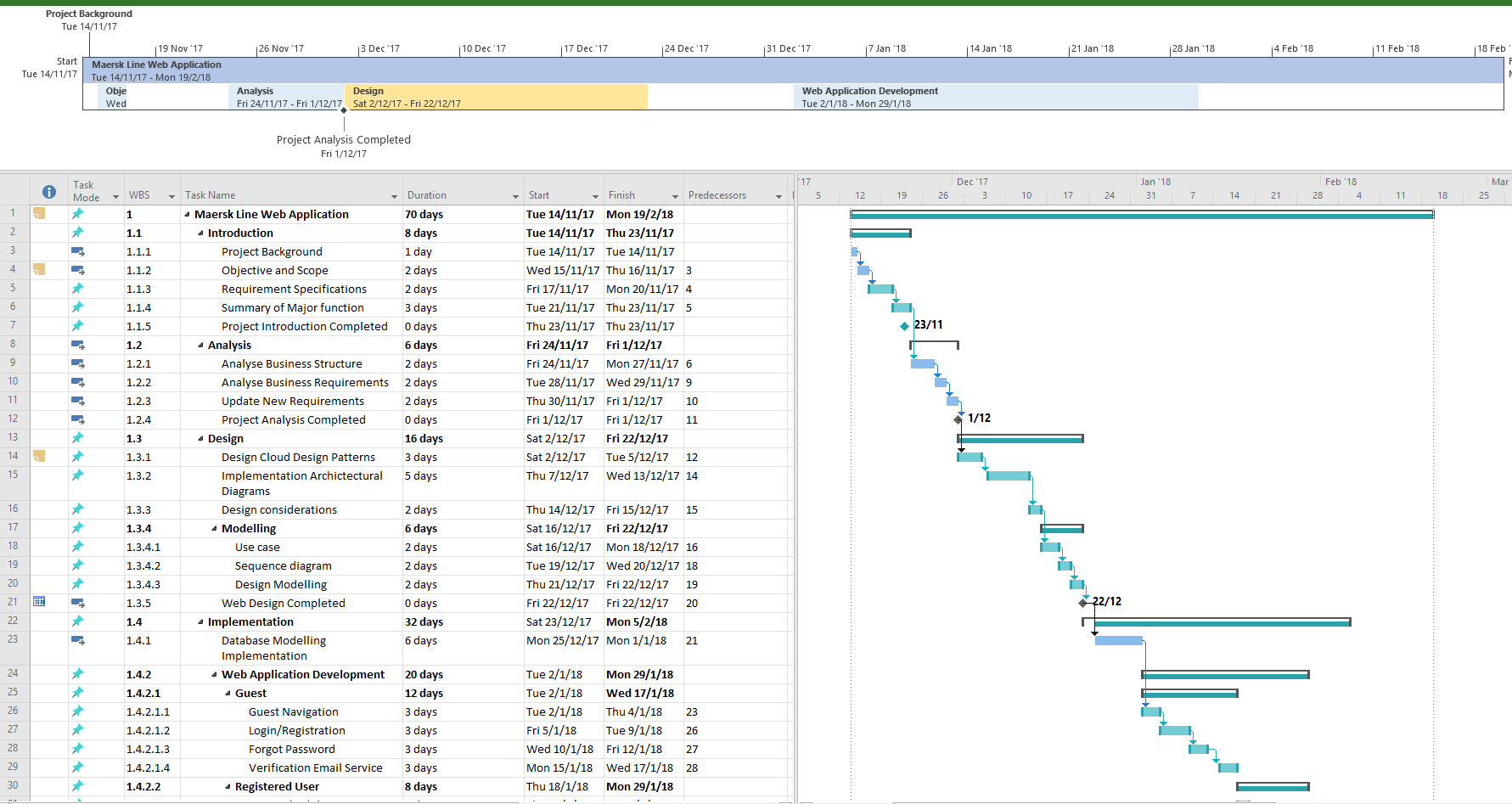


Figure - Project plan - Gantt Chart

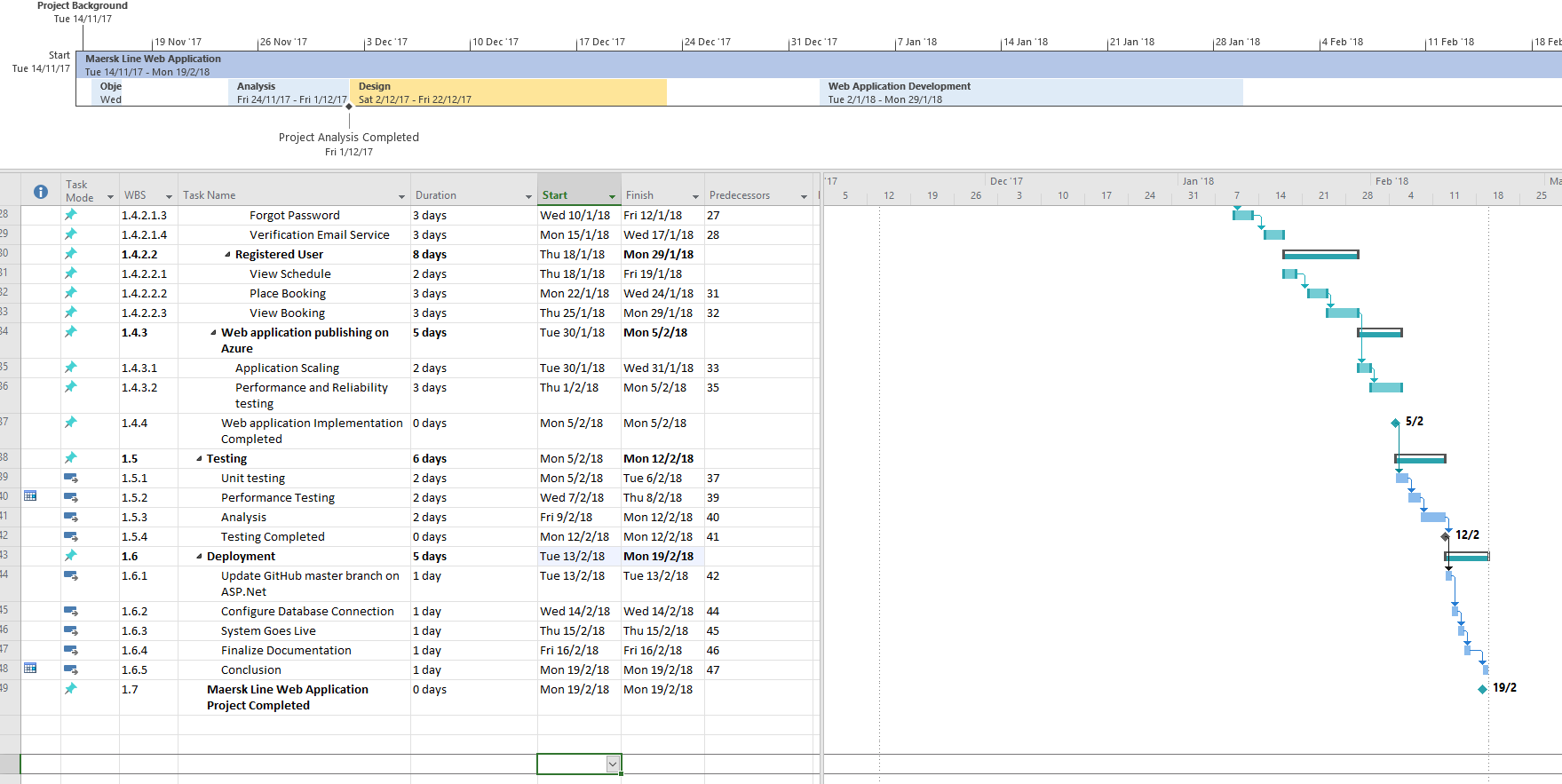


Figure Project Plan – Gantt Chart

# 3.0 Design

## 3.1 Cloud Design Patterns

One of the core cloud design patterns that were applied into the system is the **Event Sourcing** Pattern. This pattern uses an append-only store to record the full series of events managed that describe actions taken on data in a domain. For example, CRUD action in the schedule booking were considered in the system. CRUD works well for most of the web application, however, when it comes to update operations in the CRUD system directly against a data store, this can gradually be slowing down the performance and responsiveness and limit scalability due to the processing overhead it requires. Also, unless there is an additional auditing mechanism to specifically record all the transaction, the history will still not be stored. Therefore, an architecture cloud design pattern, the Event Sourcing pattern helps the Maersk Line web application to handle operations on data that is driven on data by a sequence of events, each of which is recorded in an append-only store. An architecture is demonstrated as below:

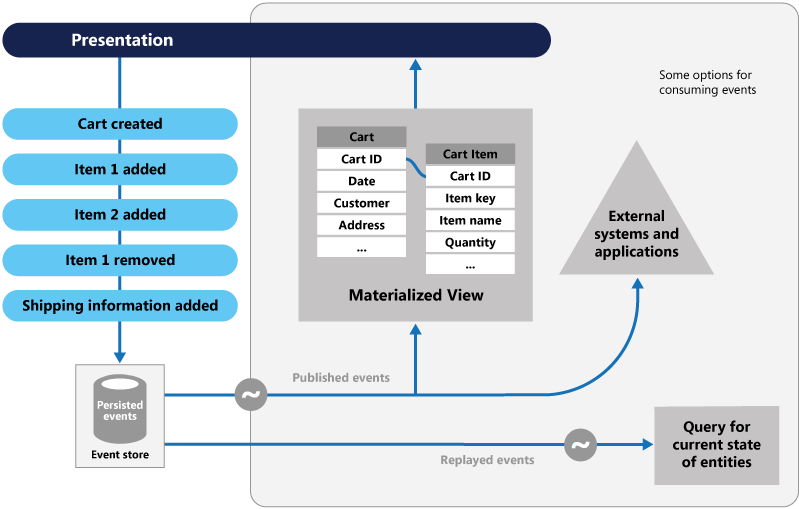


Figure - Event Sourcing Pattern Source from (Microsoft Azure, 2017)

Event Sourcing Pattern is use when the following scenarios occur:

* When the system is intended to capture data, purpose or reason in the data. For example, changes to the customer entity can be captured as a series of specific event types such as Moved home, Delete account and so on.
* When using events is a natural feature of the application, and it requires implementation effort. For example, the schedule booking event.
* When it is important to completely avoid the occurrence of conflicting updates to data. When there are one or more users updating the data at a time.

**Example**:

A shipment booking system on the Maersk Line needs to track the number of completed bookings for each shipment so that it can check whether there are available resources when a potential user tries to make a new booking. The system could store the total number of bookings in at least two ways:

1. The system could store the information about the total number of bookings in a way of separate entity in the SQL database that holds booking details. Therefore, in regard to each booking made or cancelled, the system can track by total number of booking by incrementing or decrementing it. This approach is simple and logically to implement into the system.
2. The second approach is the system could store booking information and booking cancellations as events held in the data store. It could easily calculate the number of seats available by replaying these events. This approach will be more scalable in a way due to immutability of booking schedule event. This is due to; the system is necessarily in reading data instead of rewriting it. This saves times and increases performance due to unnecessarily of rewriting a database in the system.

## 3.2 Implementation Architectural Diagrams

### 3.2.1 The Surveys Application Architecture

The development of Maersk Line web application is based on MVC asp.net core application. The application architecture is referring as below diagram. The core of the application uses Windows Azure web roles, worker roles and the storage. As from the figure shows, there are three group of users who access the application, the application owner, the public who is the guest in our system and the subscribes to the Survey service, (in this case it will be Adatum and Fabrikam) who constantly making relationship with the system and sending data to the MVC web application. Moreover, the application also uses Windows Azure SQL Database to provide a mechanism for the subscribers to replicate all the data or by sending data result into the relational database. Therefore, analyzing it or reading it from the MVC web application for business purposes.

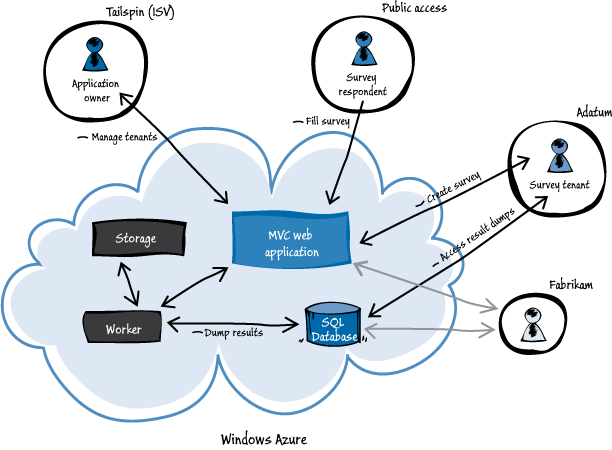


Figure The Surveys Application Architecture Source from (Microsoft, 2018)

The architecture is very useful when it comes to multi-tenant application, where a lot of users accessed into the system without crashing the system occasionally when it comes to 1000 users accessing the system at a time. Therefore, achieving performance and scalability of the system.

### 3.2.2 Geo-replicating relational databases

Initially, the database server which was deployed into the SEA region can ultimately deploy another database server on other region like Australian region. As in figure below, by having two Azure SQL Database Servers in two Regions (locally and geographically redundant) will help utilize the Active Geo-replication to provide a read-only replica in a secondary Azure Region which will take advantage of Fault and Upgrade Domains to ensure that at least one instance remains operational at all times. This means, when one crashes, another still works concurrently to ensure failures are less likely to impact the business. This will also increase business scalability, as database server are not expected to crash or ruin your business operational. Also, it adds performance to the developer, as the maximum speed of response from the server can speed up development with minimal consumption of computational power (Ganguli, 2018).

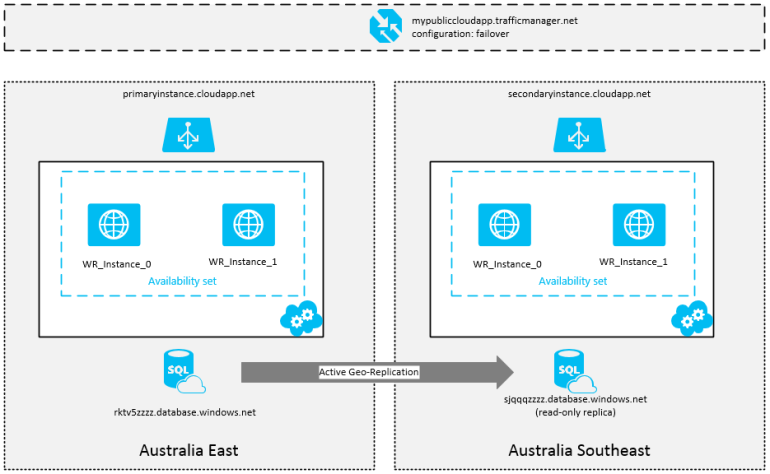


Figure Geo-replicating from one region to another region Source from (Waight, 2014)

## 3.3 Design considerations

Before design process begins, assumptions are made for example, the organization, Maersk Line has the intention to expand its services through SEA region in order to improve its profitability and eradicate errors in cargo handling for high business. Azure credit of RM 900 was given to the developer to design and develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App). However, these setup for architectural design does not cheap (example: Azure SQL Database Premium P3; AUD4, 267.49/mon in AU East) (Waight, 2014). This should also expect the service availability of the servers provide.

## 3.4 Modelling

### 3.4.1 Use case Diagram

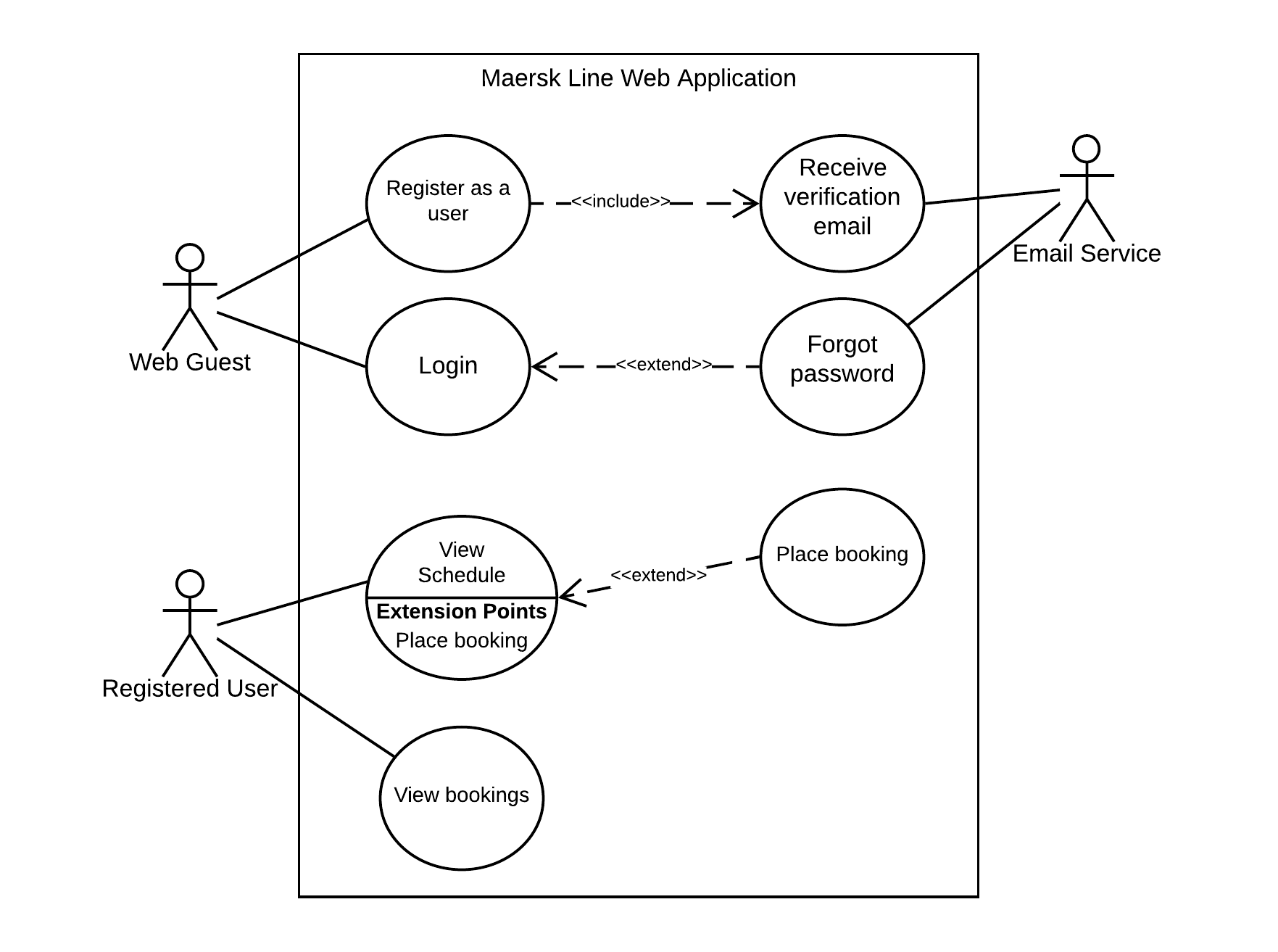


Figure – Use case diagram

### 3.4.2 Sequence Diagram

#### 3.4.2.1 Login

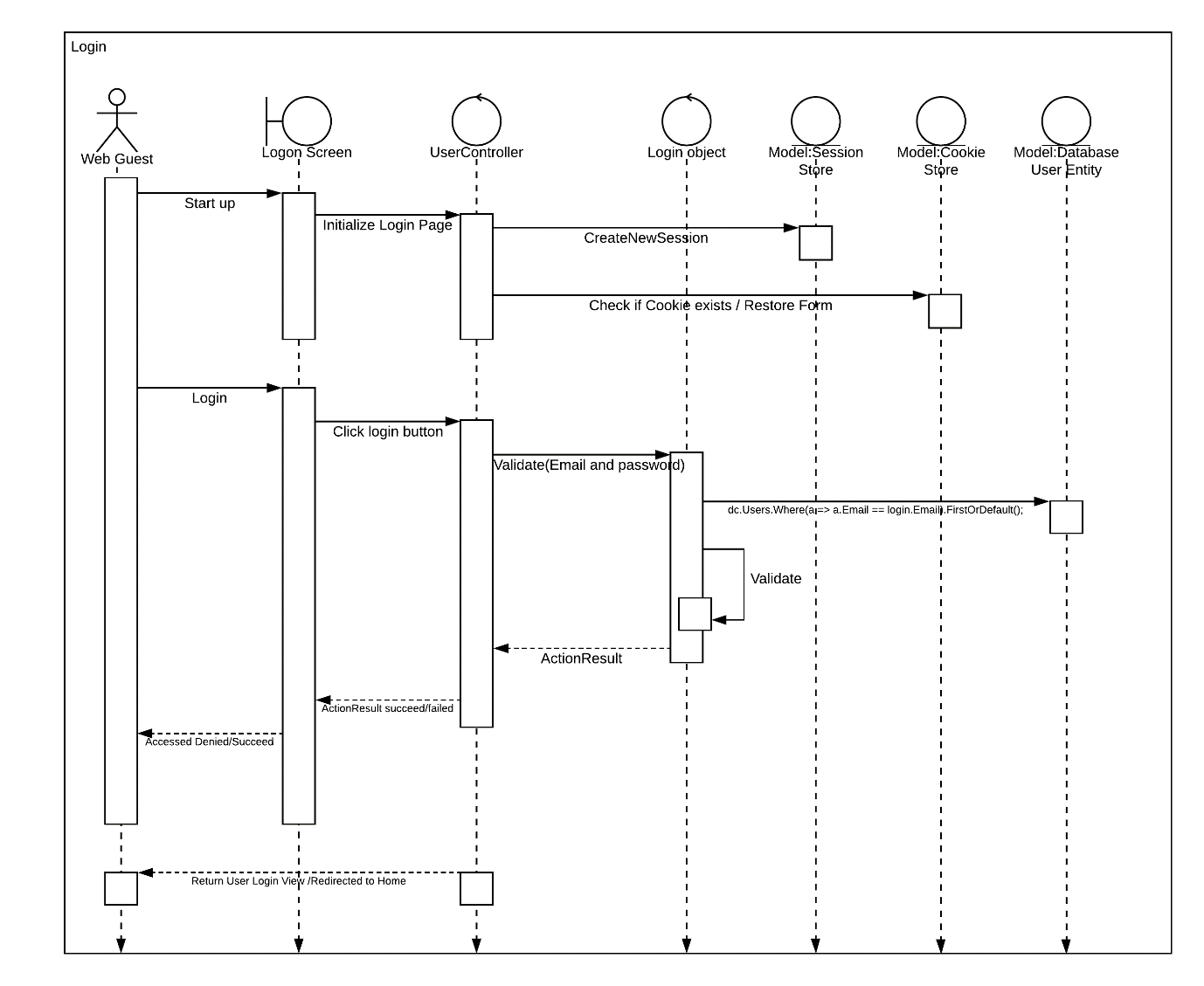


Figure Login - Sequence diagram

#### 3.4.2.2 Register as a new user

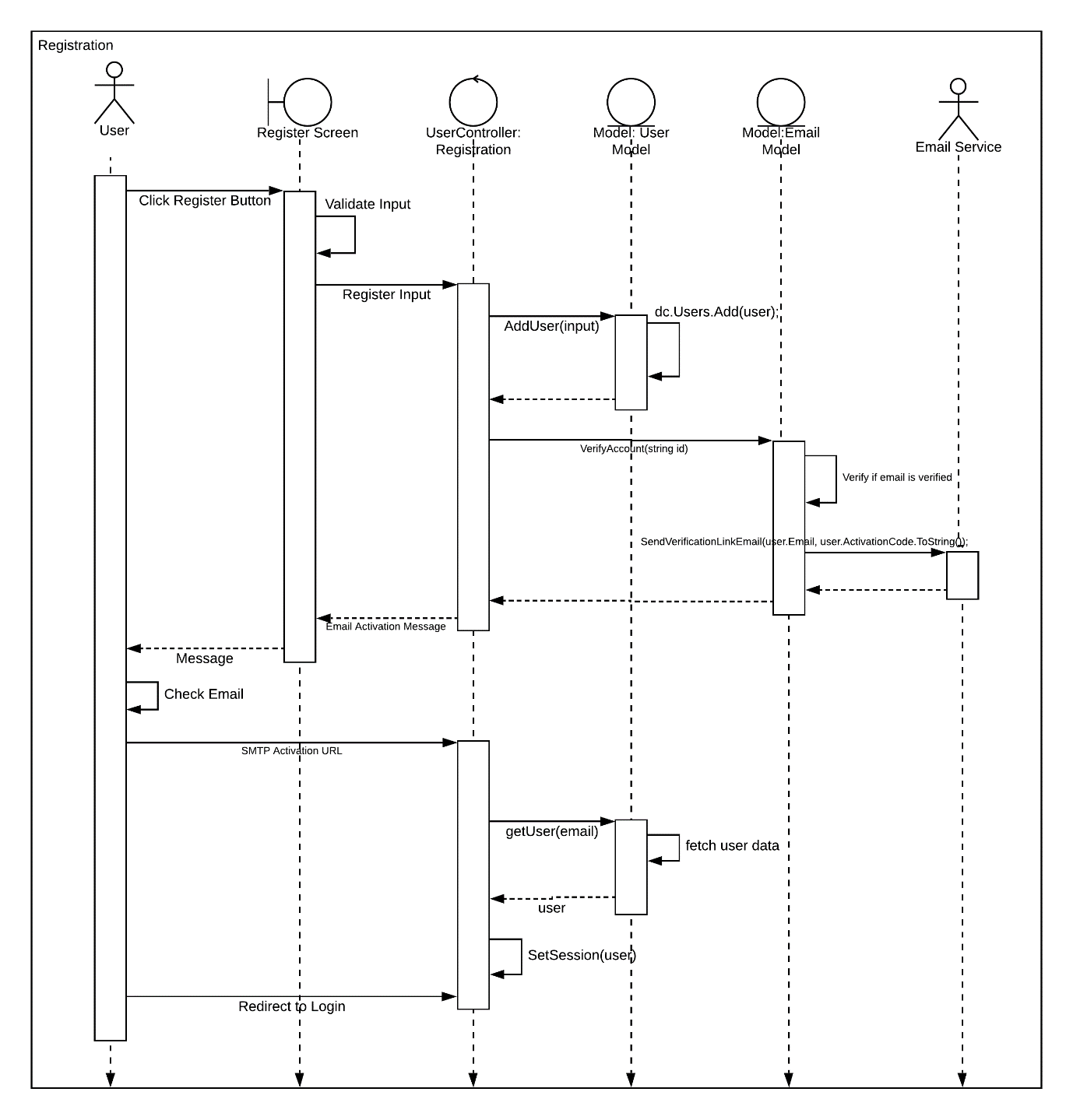


Figure Register as a new user - Sequence diagram

#### 3.4.2.3 Forgot Password

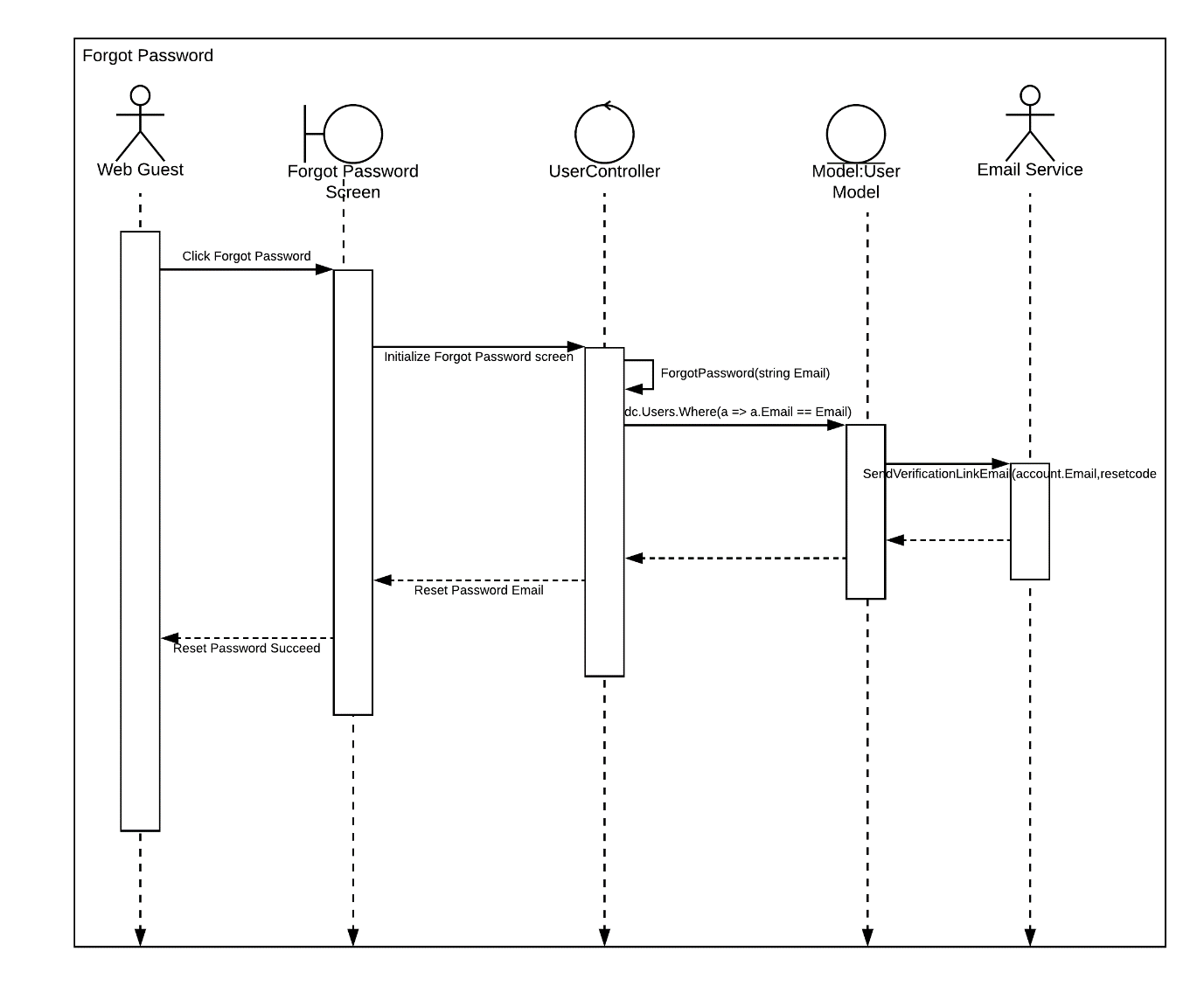


Figure Forgot Password – Sequence diagram

#### 3.4.2.4 View Schedule

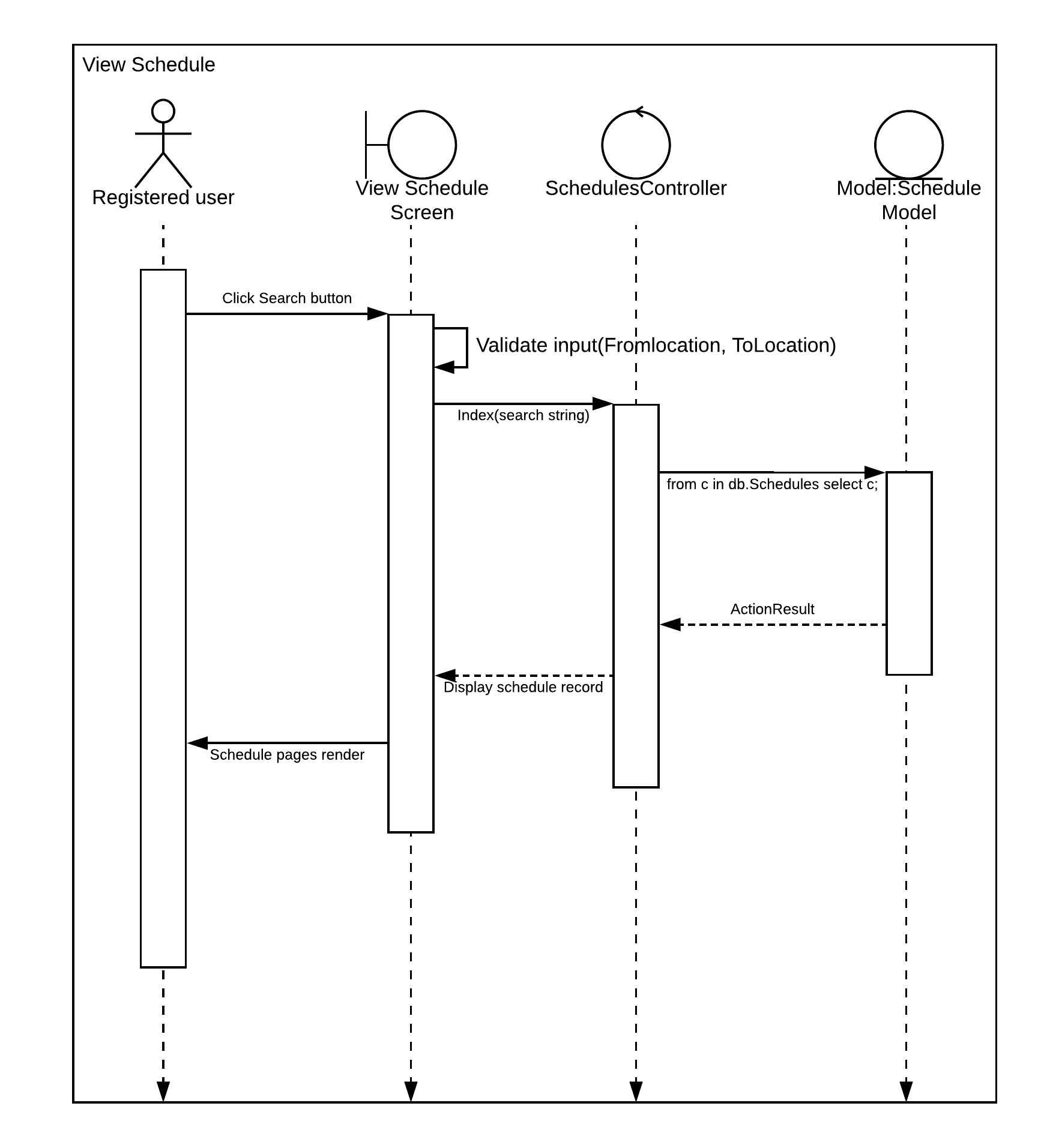


Figure View Schedule - Sequence diagram

#### 3.4.2.5 Place booking

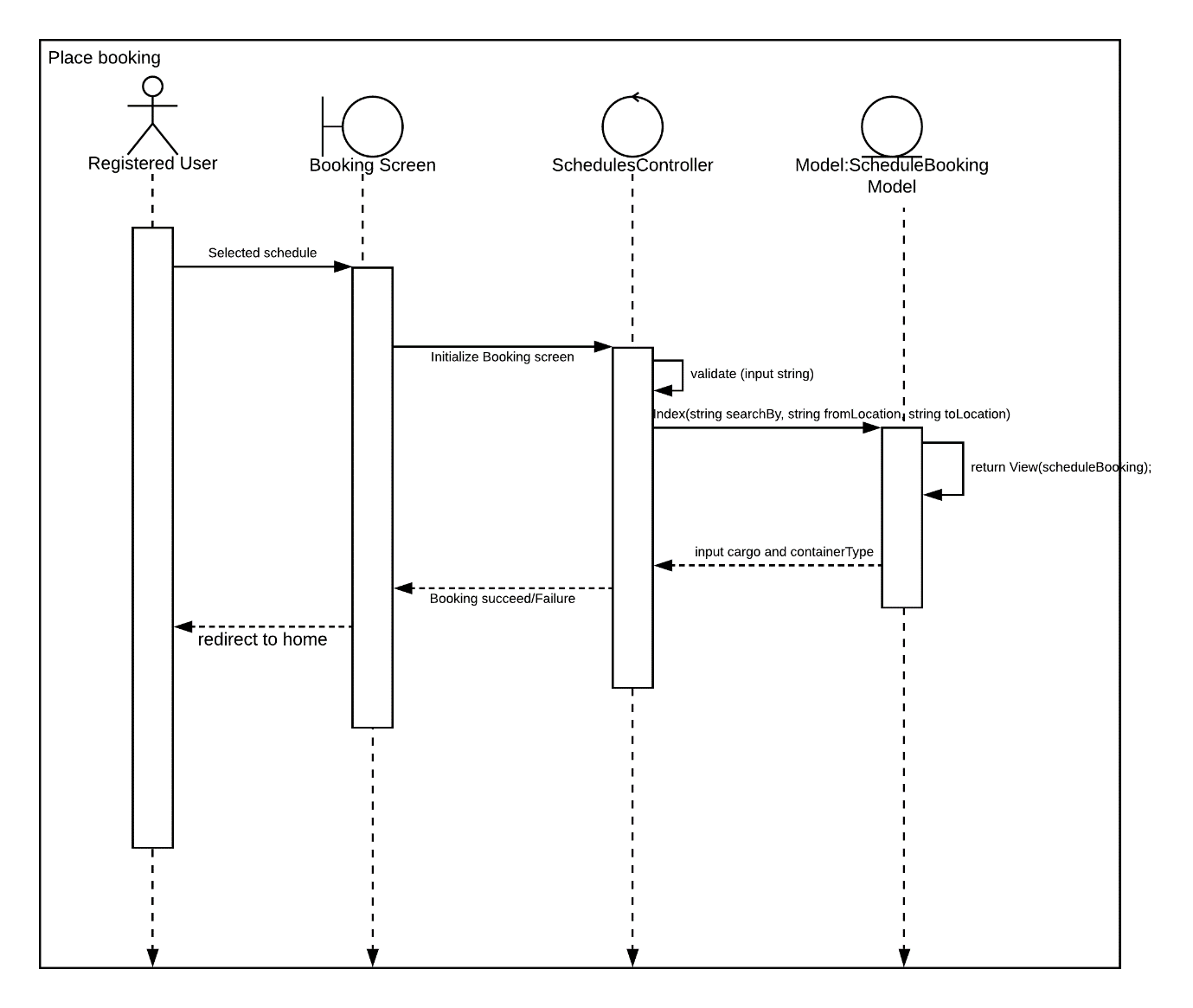


Figure Place booking - Sequence diagram

#### 3.4.2.6 View bookings

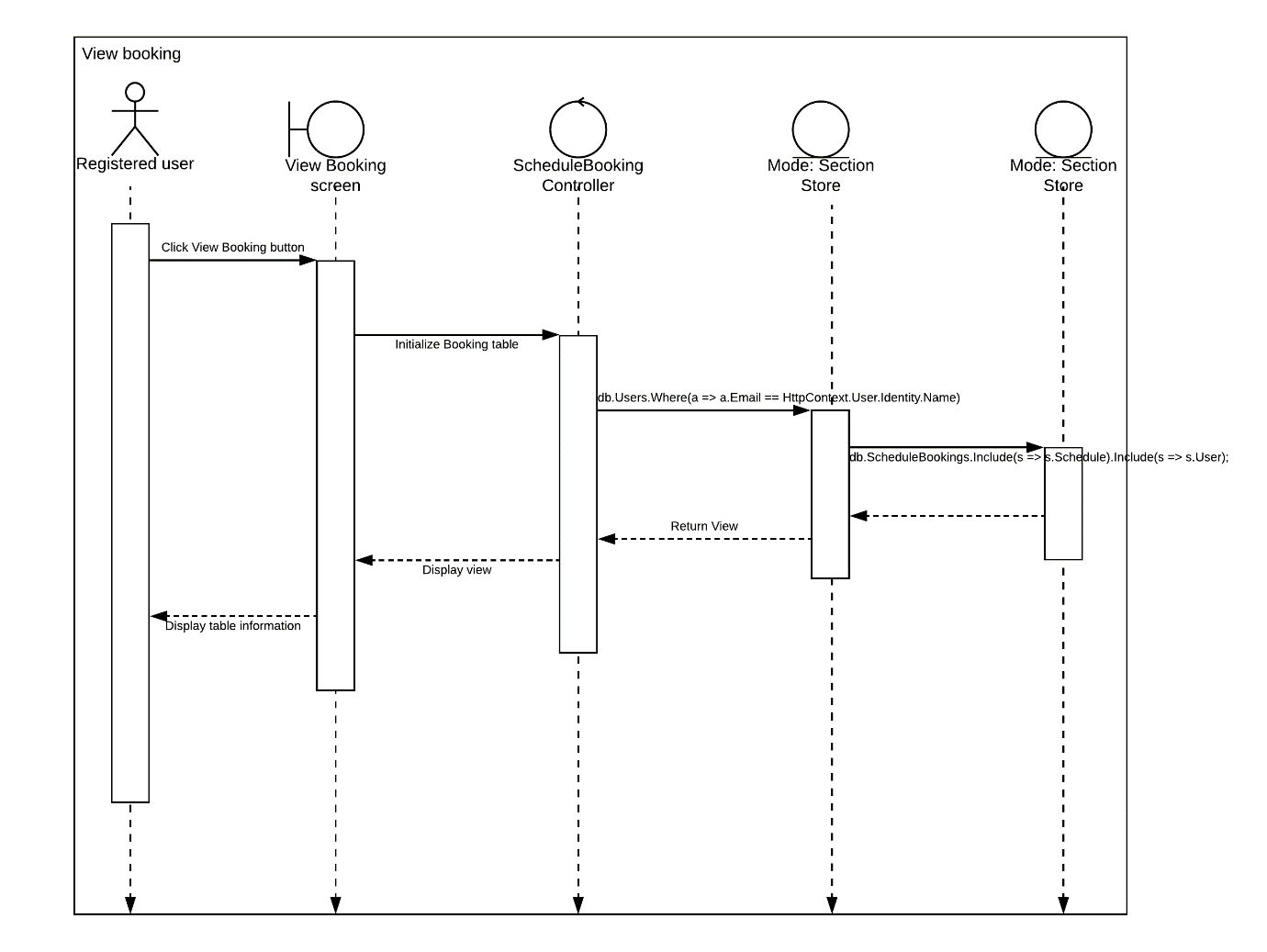


Figure View bookings – Sequence diagram

### 3.4.3 Data Modelling

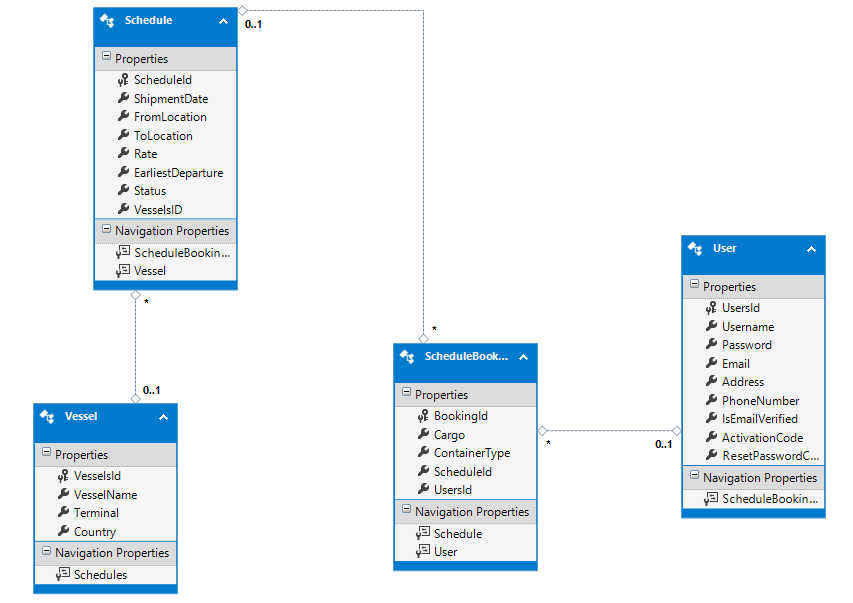


Figure Data Modelling

# 4.0 Implementation

## 4.1 Application Development

The development of web application was done using ASP.NET with MVC framework, which comprises of Model, Controller and the View. Model is used to store the entities, Controller is the relationship between the View and the model, whereas, the view is the layout or where the html takes place, view is also the design of the system. All, application design is based on CSS style sheet, JavaScript, and jQuery files. Also, a working SQL database will be hosted onto the Microsoft Azure to allow offline accessing while Azure resources are running on Microsoft Azure. File structure is shown as follows diagrams:

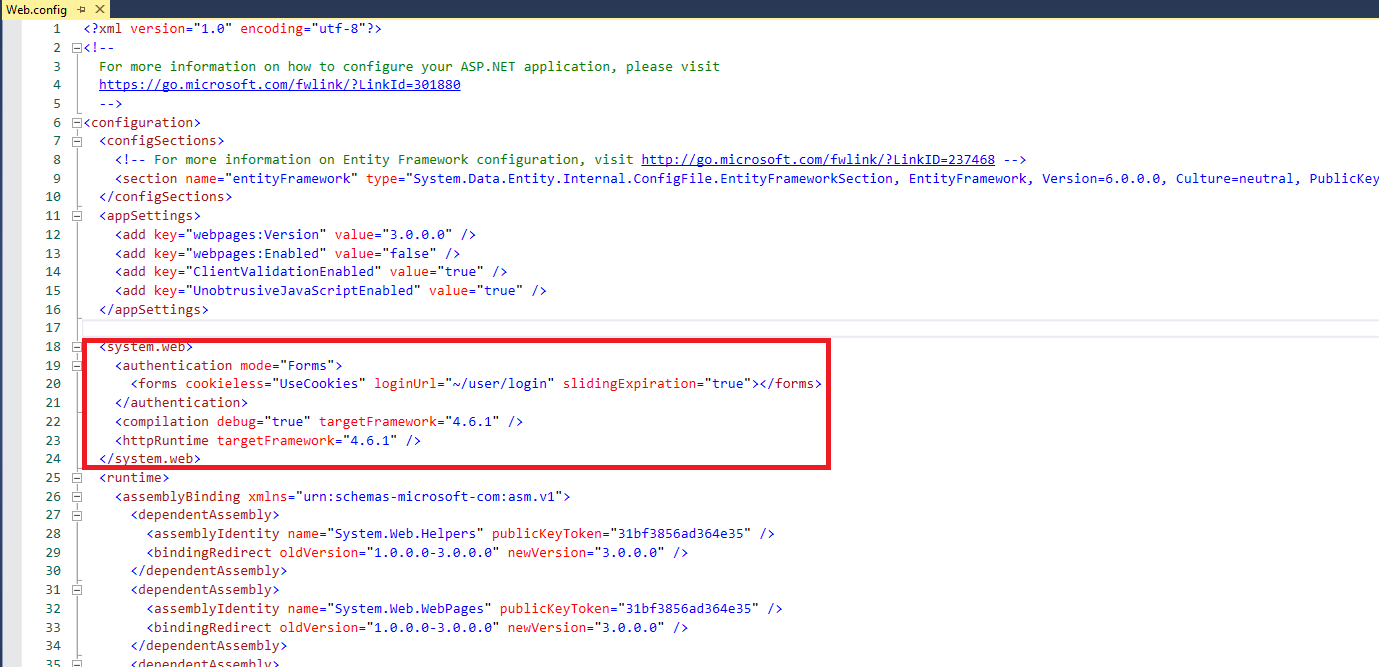


Figure Cookies in Web.config

From the above figure, this is where the connection string of database and system configuration takes place. Where in the image above, the cookies configuration is used to help the user to remember his/her current login, these cookies stored will allow user convenient and therefore no need retype his/her login credentials. This interface is shown in **Figure 12** below.

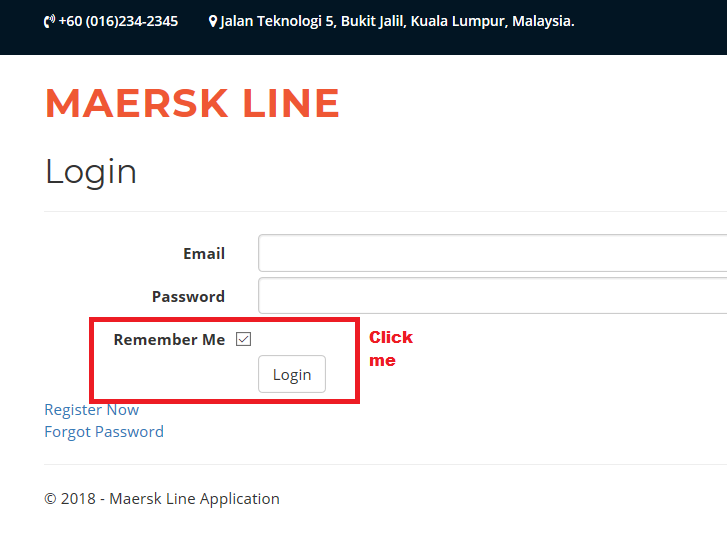


Figure Login Screen

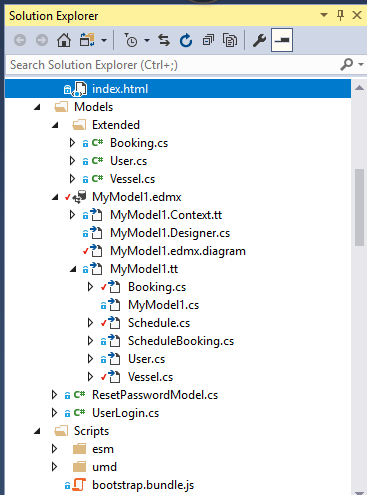
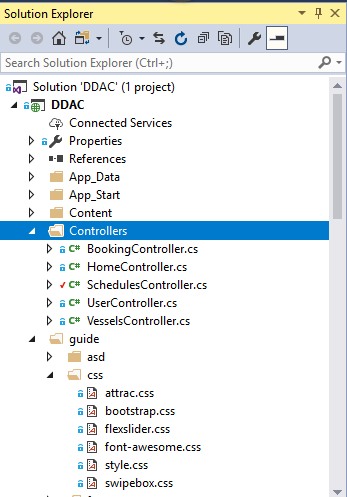


Figure Solution Explorer – Part 1

From the above figures, is where the files are located (Solution Explorer). As for controller file, is where the controllers have been placed. Controllers are necessarily to manage models from the data modelling as shown in **Section 3.4.3.** Controllers also help to retrieve Views, which is the interface when successful action has been performed. Booking Controller and schedules controller is where the booking and shipment schedule took place. Whereas, the user controller is where the management of user’s information (login, registration, forgot password and verification of users take place). Lastly, the vessels controller is where the vessels for current shipment takes place.

As for the second figure, is where the model files applied. These model files are responsible in managing the model which is Booking, User, Vessels, scheduleBooking and the relational database took place. Model is responsible in controlling the datatype or objects through get and set (**Figure 14** below). The model can also be implemented with the datatype or validation for the particular text boxes.



Figure – User Model – UserMetatdata

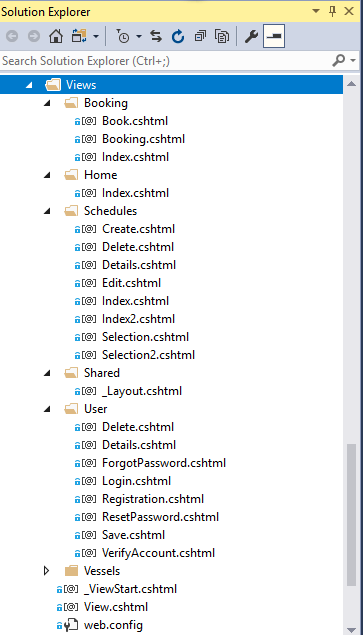
From the **Figure 15** besides, this is where the view takes place. The view will be responsible in storing the cshtml files. These cshtml file can be used to create text boxes and by validating the text boxes from each model used. On each cshtml, it can be included with the CSS style sheet, JavaScript or jQuery designs by **referencing** design files.

Figure 18 – Solution Explorer Part 2 - Views



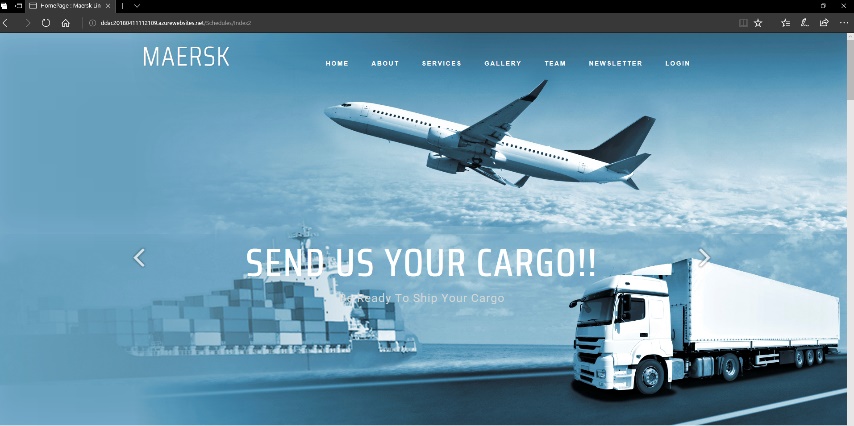
Figure Login cshtml

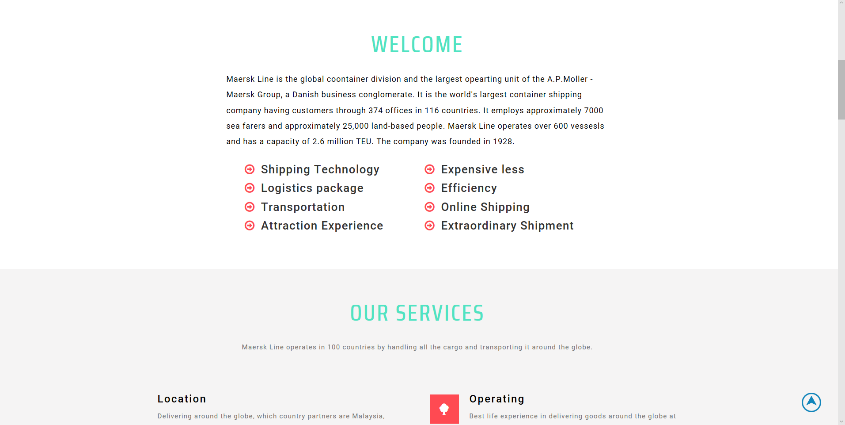
From the **Figure 16** above, this is where the login cshtml looks like, it will create the email, password text boxes and remember me check box and the login button, it can be seen in Figure 12 Login screen.

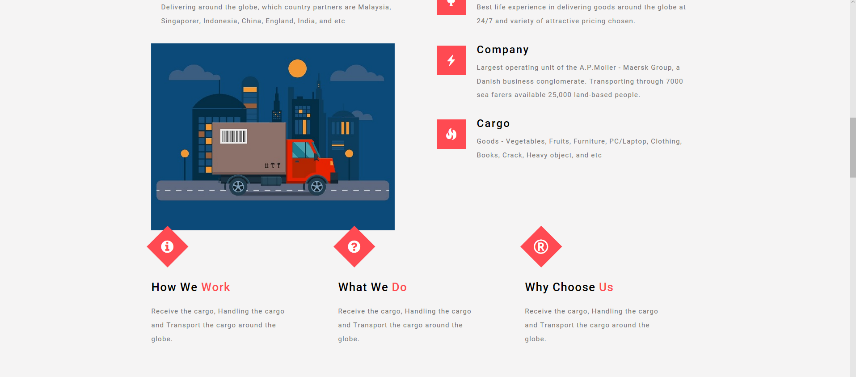


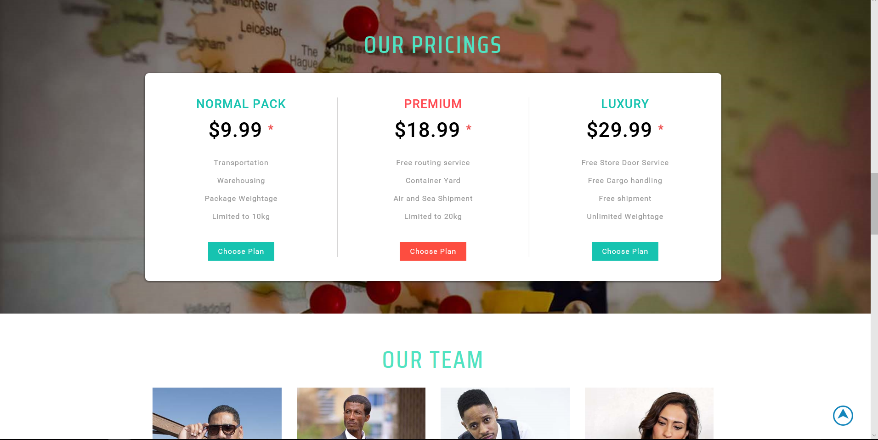
Figure - Design stylesheet in cshtml

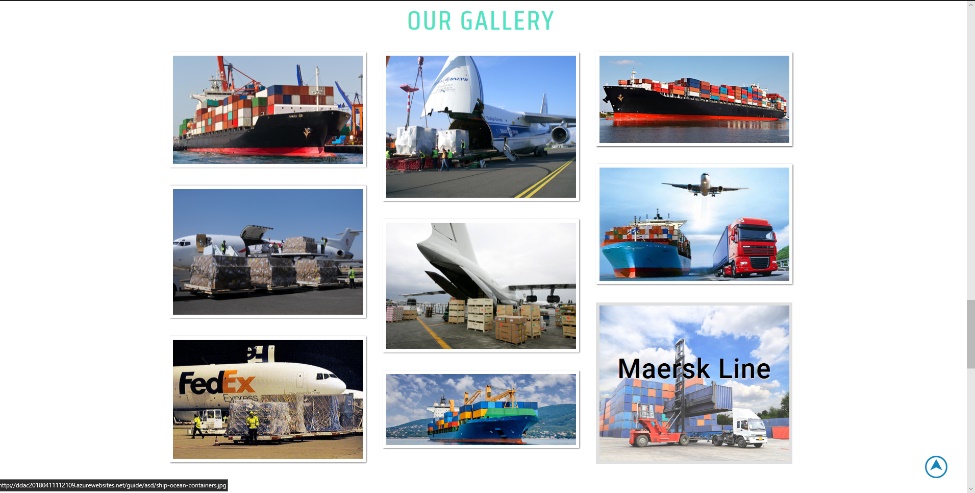
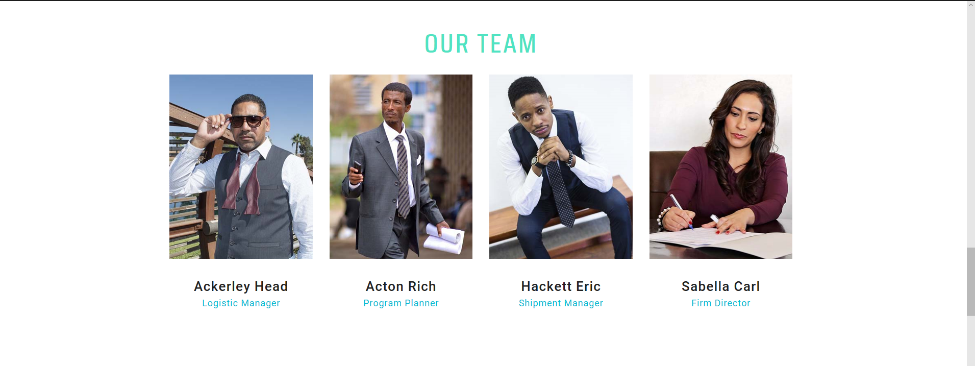
CSS style sheet referenced can be added as the design as if necessarily.

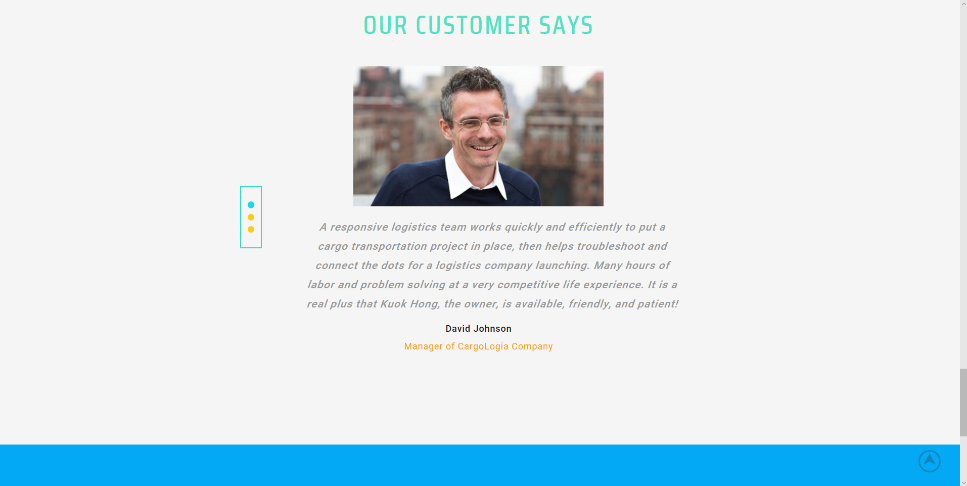












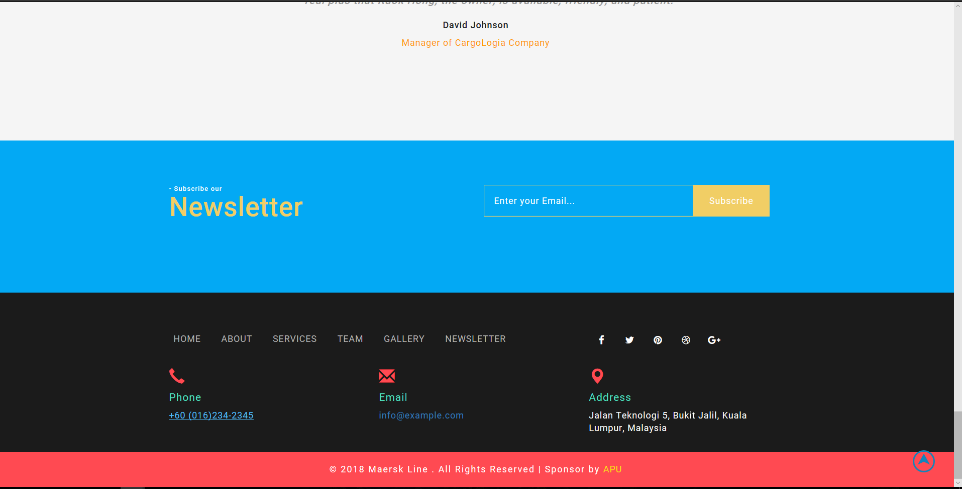
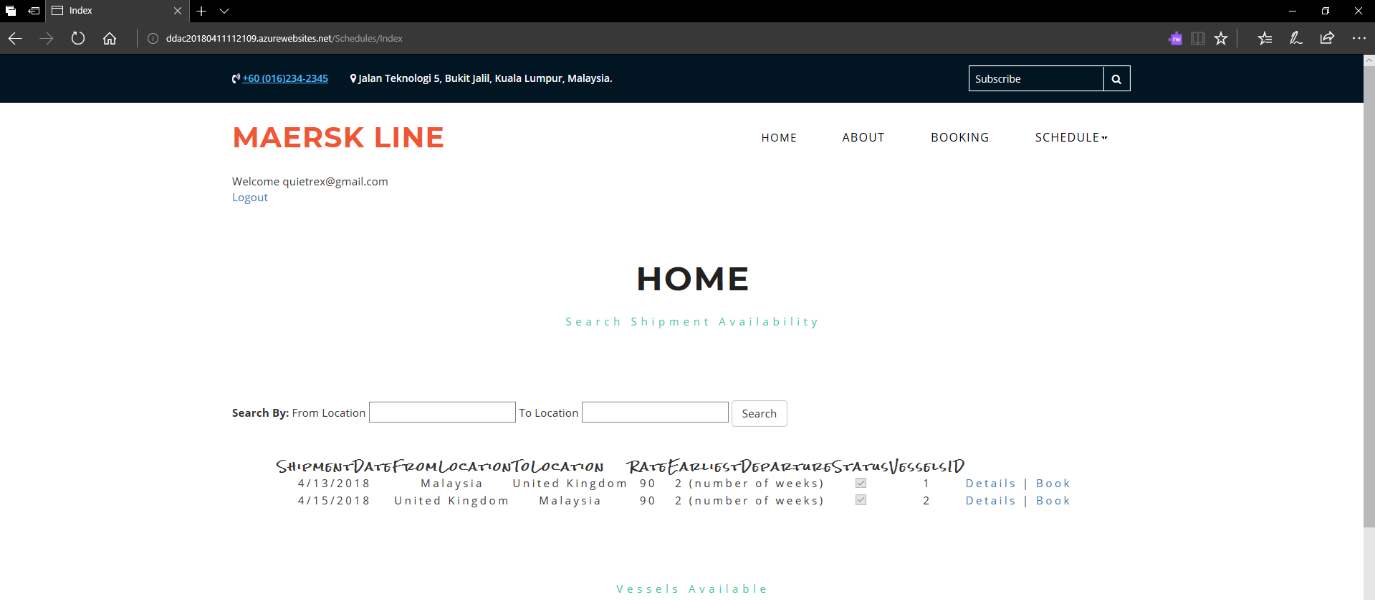


Figure Screen Design home page

This home page is based created to allow attract user when it comes to appropriate and interactive web design.



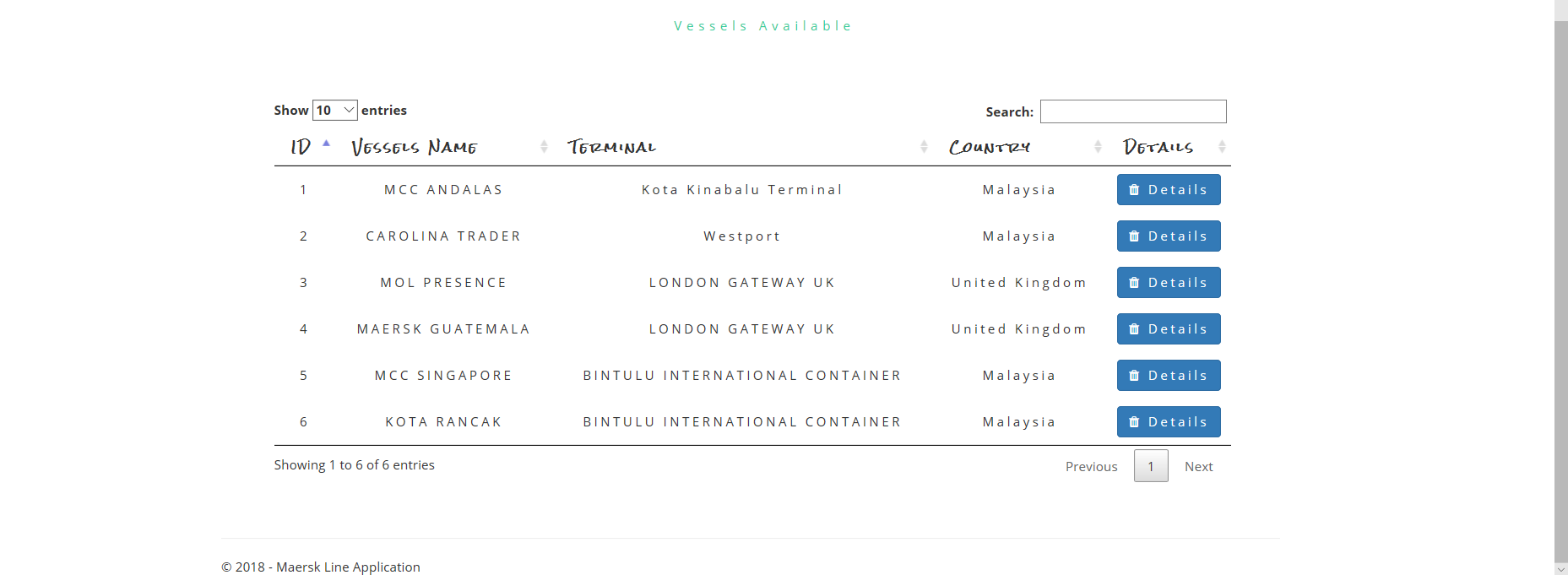


Figure Home screen

On the above figure, this is where the home screen looks like, user can be finding his/her desired shipment schedule by typing from location and to location. Shipment schedule will be shown, if no schedule available it will show no record (**Figure 21**).

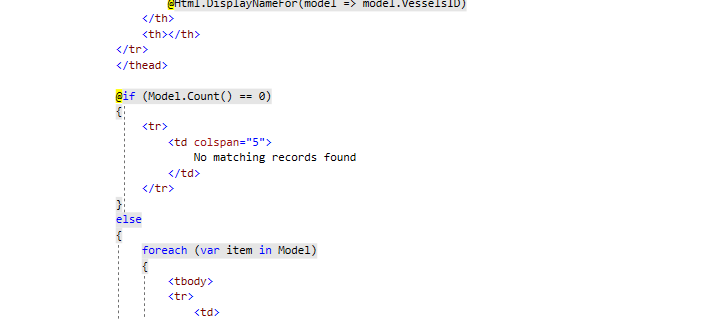


Figure - No matching records found

User can proceed booking particular schedule which has been found. User can be typing his/her desired cargo and container type. When save is clicked, it will prompt successful booking.

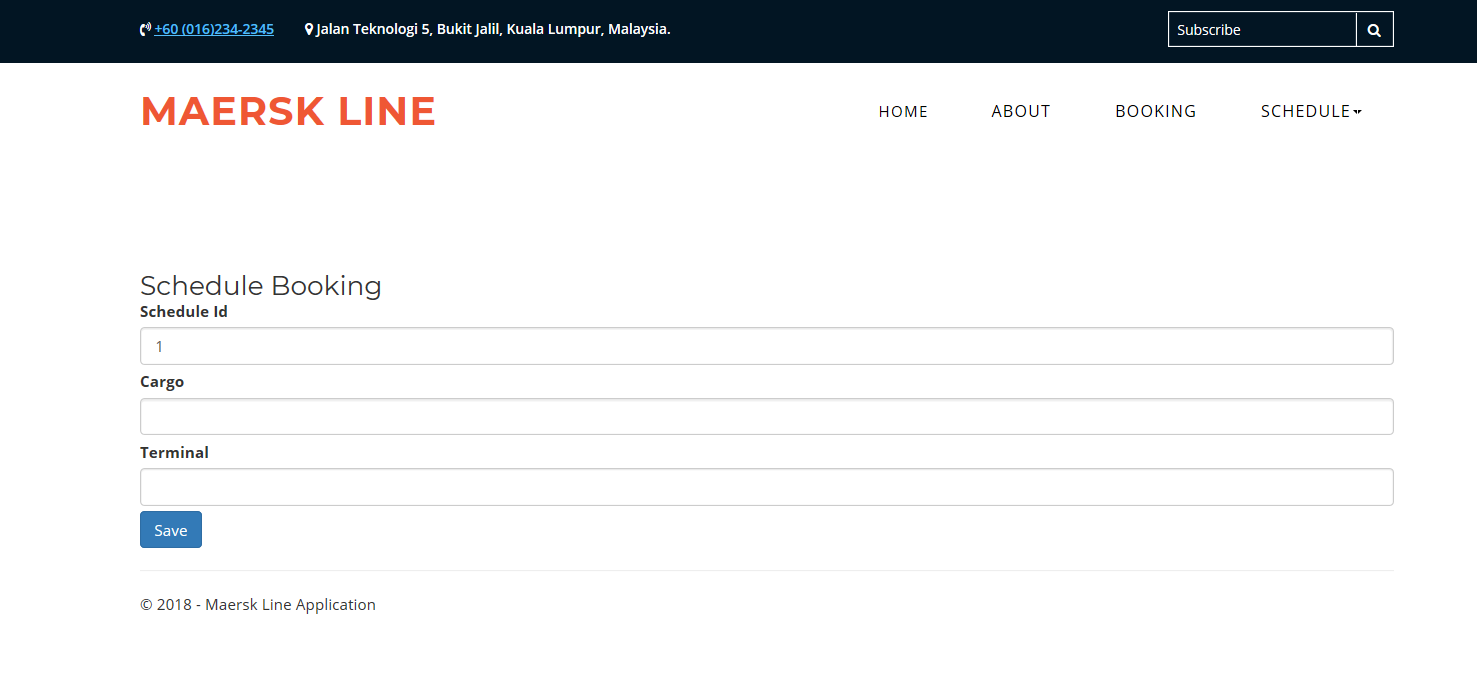


Figure - Schedule booking - Cargo, Container Type

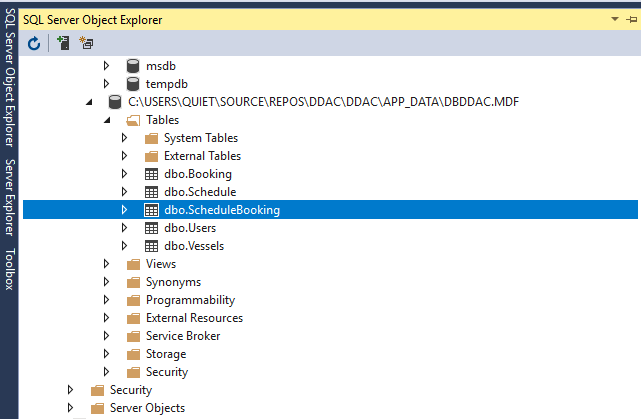


Figure – SQL Server Object Explorer - Database

This booking information will be once saved into the database which will later deploy into the Microsoft Azure. Also, this Maersk Line web application will be managed by having GitHub version control. The **source code** for the application is available at <https://github.com/quietrex117/maersklinewebapp.git>

A **full demo of the functional system** is included in the attached CD or online at <http://ddac20180411112109.azurewebsites.net/> and **video** at

<https://web.microsoftstream.com/video/5f38d13d-a628-4bc4-a8f0-649afe7c354d>

## 4.2 Azure Publishing

There are two options in getting web application to be published, one most convenient is by publishing inside ASP.NET which is shown in figure below. By publishing it through ASP.NET, it saves more time as web application is only needs to be published when the application has been succeed developing.

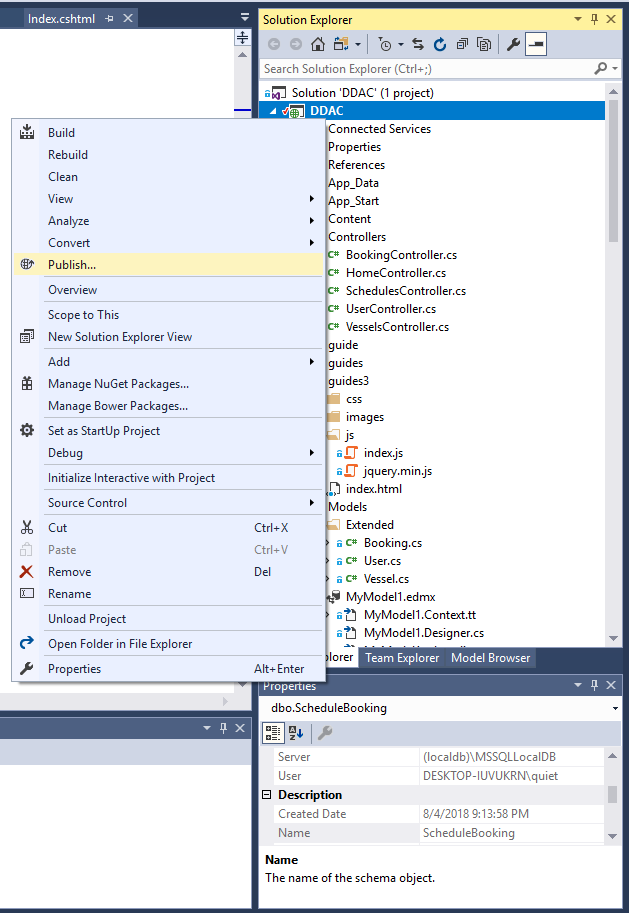


Figure Step 1: Publish through web application: ASP.NET

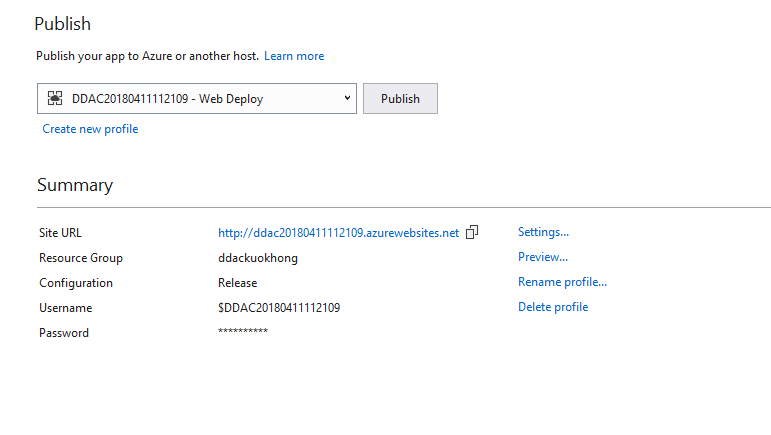


Figure - Application which have already instance

As can be seen, once publish button has been clicked, it has prompted with as above, as the web application has been published, it can be seen as above. As if new application the developer needs to publish, it can be done by clicking create new profile. It will prompt by picking new publish target.

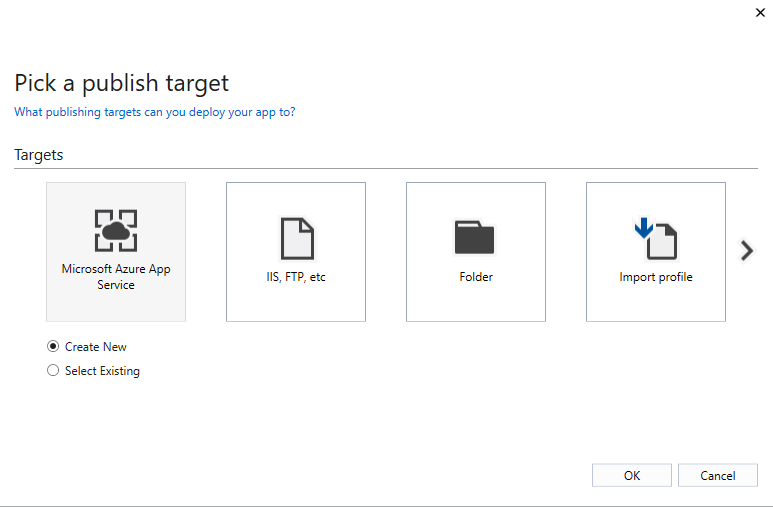


Figure Publish web app by clicking “creating new profile”

There are few options, when it is on azure portal, it can simply publish through azure portal and by importing profile which can be downloaded once web app has been published through azure portal. However, if it is published through the IDE, Visual studio, it can simply remain clicking the “Microsoft Azure App Service”, it will continue to publish the app into the azure portal for the user who already logon into the account.

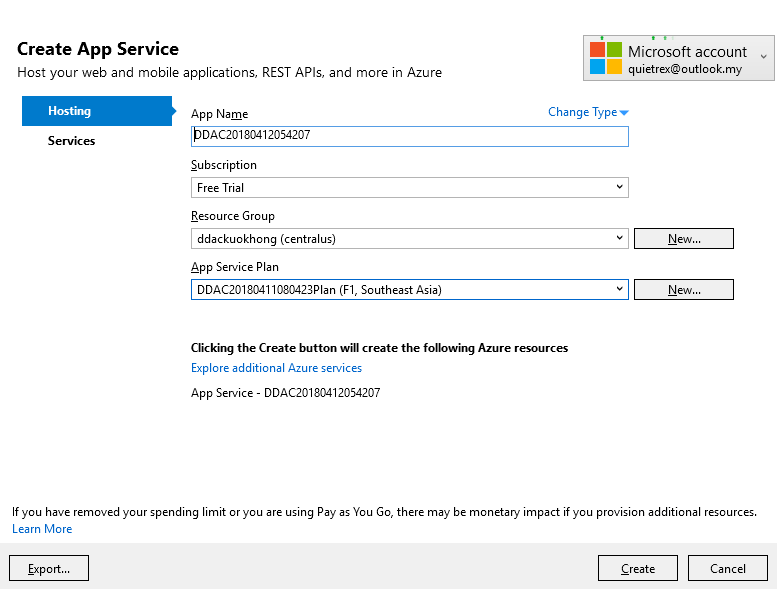


Figure App Service for resource group, subscription, name for the app

From the above, there are a few options to be selected, the app name, which how the azure web host name is, and the subscription, resource group, and the app service plan. These are needed to create in order to allow resource group to manage the following servers or web app server. Once it is done, select **Services** to continue updating the databases for the web application. This can be done from **Figure 25**, by clicking settings or by **Figure 26**, by clicking services tab.

As seen I already published the web application, the step will be follows from the **Figure 28**, by looking into the settings, the user is responsible to find the db. entities which have been created or deployed into the Microsoft Azure. It can be found simply clicking the SQL Object Explorer and using it in Follows below:

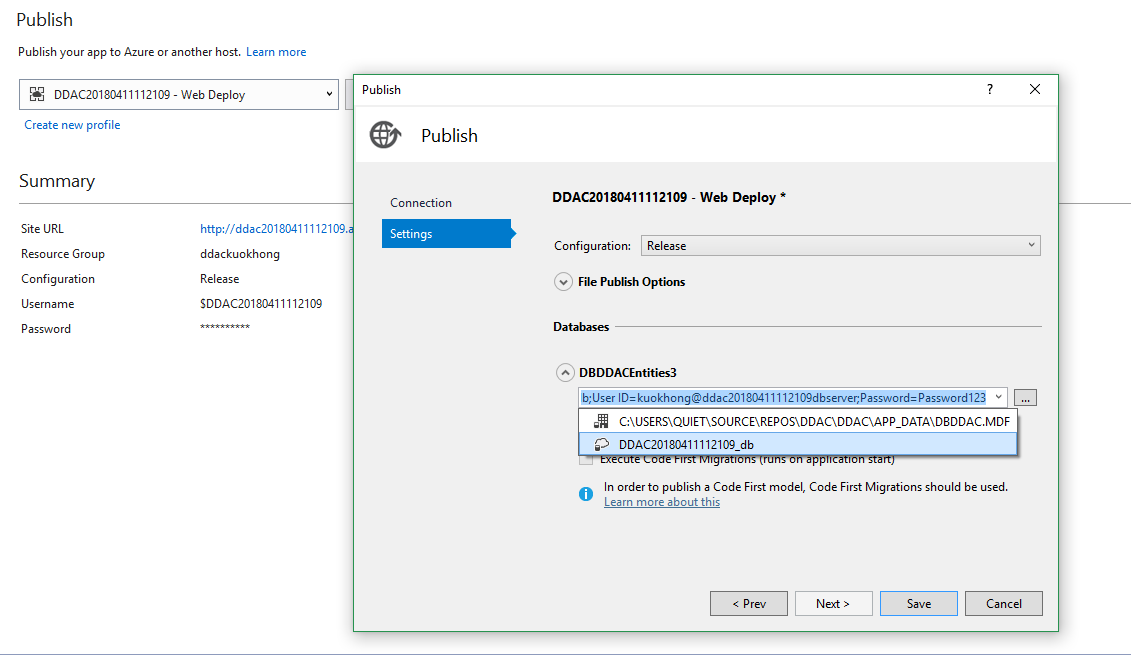


Figure Updating DB Entities for the web app

## 4.3 Application Scaling

According to Microsoft Azure, auto scaling is the process of dynamically allocating resources to match performance requirements. As the volume of works grows, an application may need additional resources to maintain the desired performance levels and satisfy service-level agreements (SLAs). Conversely, if the additional resources are no longer needed, they can be de-allocated to minimize costs (Microsoft Azure, 2017).

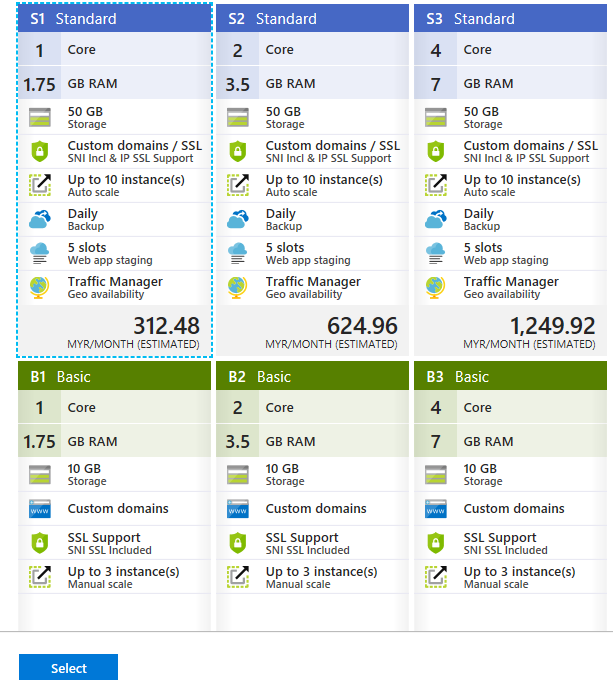


Figure – Standard plan

Scale plan can be seen as in **Figure 29** above. All the Maersk Line web application will be based on S1 Standard plan or above, therefore instances built can be used on auto scaling to remain CPU, and memory usage in order to achieve high server performance. Azure Monitor auto scale provides a common set of auto scaling functionality for Azure App Service on performing scaling on a schedule or based on runtime metrics (Microsoft Azure, 2017). The services can be used to scale out to 10 instances which is on standard plan, on weekdays and scale in to 4 instances like S2 or S3 standard plan on Saturday and Sunday. This is due to more people or how busy the server is. By running or scaling 10 instances will be extremely crucial when you need to run longer the maximum execution time allowed on the Consumption plan provided which will be approximately of 10 minutes period (Microsoft Azure, 2017).

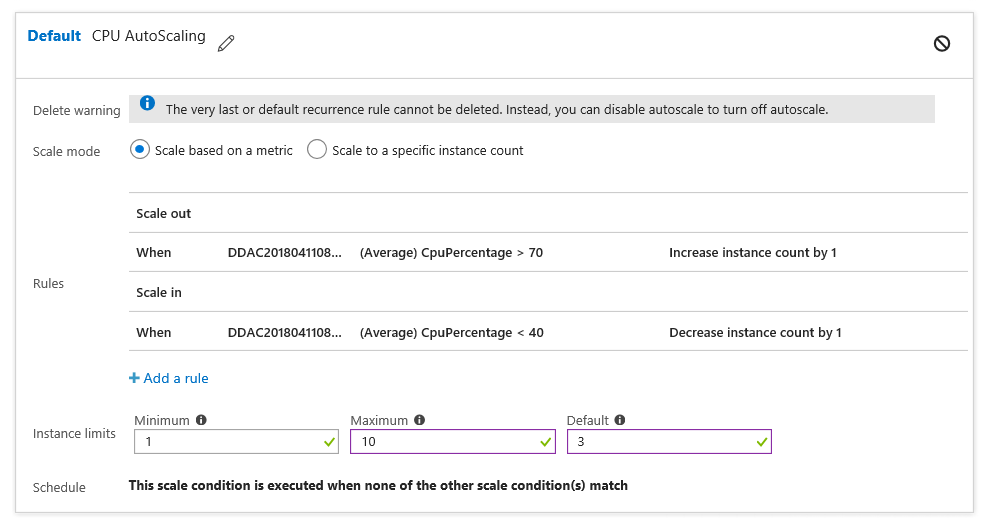


Figure - Auto scaling

From the above figure, it is crucial when designing the CPU scaling plan, for example scale out by one instance if average CPU usage is above 70% and is scale in by one instance if CPU usage falls below 40%. This will help solve the Maersk Line web application system failures and increase high performance for future-proof cargo handling, as the business is expanding.

### 4.3.1 Reliability & Performance

One of the most efficient solution in increasing the reliability and performance is by using traffic manager. As what the documentation has been stated, as two regions are expected to be built in order to solve system downtime failures that causes business operation solution to be stopped. Traffic manager can help solve this, as one of deployed geo-location – Australian and Southeast Asia region, traffic manager allows the system to control the distribution of user traffic for service endpoints in different data centres. If one requests from SEA region, it is expected to directly send services requested to the SEA region instead of Australian region. This development will efficiently improve the responsiveness for high-performance applications, as Azure traffic manager will allow service to run its web application in the data centres around the world with the lowest network latency for the client without system downtime (Microsoft Azure, 2017).

## 4.4 Testing

### 4.4.1 Unit Testing

The table below contains the test plan used on performing unit testing. These tests are comprised on individual components for each functionality for the system. Please note that, all unit test cases are performed locally, that is to faster testing efficiency in finding defects.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Title | Description | Expected Result | Actual Result |
| **Register as a new user** | | | | |
| RU-001 | Input Validation | Test Input Validation   1. Redirect to URL “localhost/registration” 2. Click on “Register” button | Fields under all text boxes display in red, error message displayed. | As Expected |
| RU-002 | Input successful | Test Input Successful   1. Redirect to URL “localhost/registration” 2. Enter correct credentials. 3. User created, activation email sends to your inbox 4. Verify through clicking user. | Successful message displayed, and verification email send to your inbox | As Expected |
| RU-003 | Existing user has been entered | Test Input Validation   1. User enter existing email 2. Click on “Register button” | Error message displayed | As Expected |
| **Login Page** | | | | |
| LP-001 | Input Validation | Test Input Validation   1. Redirect to URL “localhost/login” 2. Click on “Login” button | Error message displayed, prompting invalid account | As Expected |
| LP-002 | Input successful | Test Valid Input   1. Redirect to URL “local/login” 2. Enter valid credentials 3. Tick “remember me” | Redirect user to home page | As Expected |
| LP-003 | Input failed | Test Invalid Input   1. User logout, as from the previous step, user ticked “remember me” 2. Enter invalid credential | Error message displayed, prompting invalid account | As Expected |
| **Forgot Password** | | | | |
| FP-001 | Input valid email | 1. User typed in valid email | Verification email is sent to user account | As Expected |
| FP-002 | Input invalid  Email | User typed invalid email | Verification email does not send to user and error message displayed | As Expected |
| **Schedule Page** | | | | |
| SP-001 | Searching schedule | Test Validation   1. User input valid information (from location and to location) | Shipment schedule is displayed into the table and displaying it to the user | As Expected |
| SP-002 | Typing invalid info | Test Validation   1. User input invalid information (from location and to location) | Shipment schedule is not displayed into the table and table displaying message “no record has been found” | As Expected |
| **Booking Page** | | | | |
| BP-001 | Selected shipment schedule and type valid info | Test Validation   1. User selected a desirable shipment schedule from the schedule page 2. User clicked book 3. User typed in valid information (Cargo and Container Type) | It redirects user to the schedule page once booking has been made | As Expected |
| BP-002 | Selected shipment schedule and type invalid info | Test Validation   1. User selected a desirable shipment schedule page 2. User clicked book 3. User typed in invalid information (Cargo and Container Type) | Error message displayed and no booking has been made | As Expected |
| **Component Test** | | | | |
| CT-001 | Navigation bar on home | User must log in into the system   1. User logged in into the system 2. User request the URL “localhost/home” | Home page loaded even though it is already logged in | As Expected |
| CT-002 | Navigation bar for booking | User must log in into the system   1. User logged in into the system 2. User quest the URL “localhost/booking” | Booking page loads | As Expected |
| CT-003 | Navigation bar for schedule | User must log in into the system   1. User logged in into the system 2. User request the URL “localhost/schedule” | Schedule page loads | As Expected |
| CT-004 | Navigation bar for registration | User must log in into the system   1. User logged in into the system 2. User request the URL “localhost/registration” | Registration page loads | As Expected |
| CT-005 | Navigation bar for guest - Home | User must not log in into the system   1. Do not login 2. User clicks on home 3. User redirect to login page | Login Page loads | As Expected |
| CT-006 | Navigation bar for guest - Booking | User must not log in into the system   1. Do not login 2. User clicks on home 3. User redirect to login page | Login Page loads | As Expected |
| CT-007 | Navigation bar for guest - Schedule | User must not log in into the system   1. Do not login 2. User clicks on home 3. User redirect to login page | Login Page loads | As Expected |
| CT-005 | Navigation bar for guest – Forgot Password | User must not log in into the system   1. Do not login 2. User clicks on home 3. User redirect to login page | Forgot Password loads | As Expected |

### 4.4.2 Performance Testing

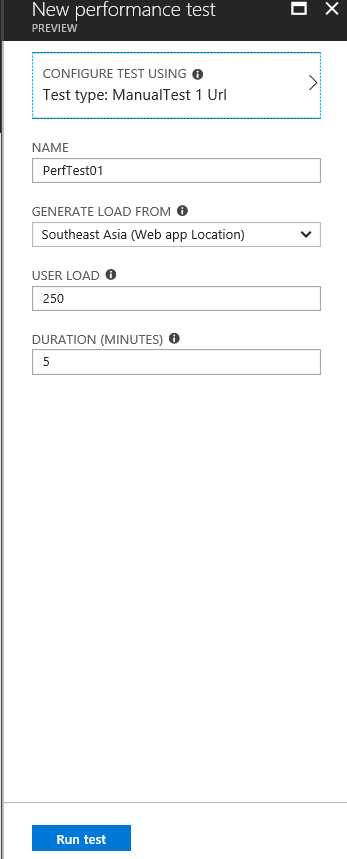
The performance of the Maersk Line Web Application can be conducted with the help of the functionality provided in the Azure portal. The performance test can drastically result of performance for the web app with generated user load of 250 to 850 for five minutes by showing the metrics from the test cases of various service plans. The performance test will be undergone a queue to find out the performance of three running service plans – S1, S2, and S3.

Figure New Performance Testing

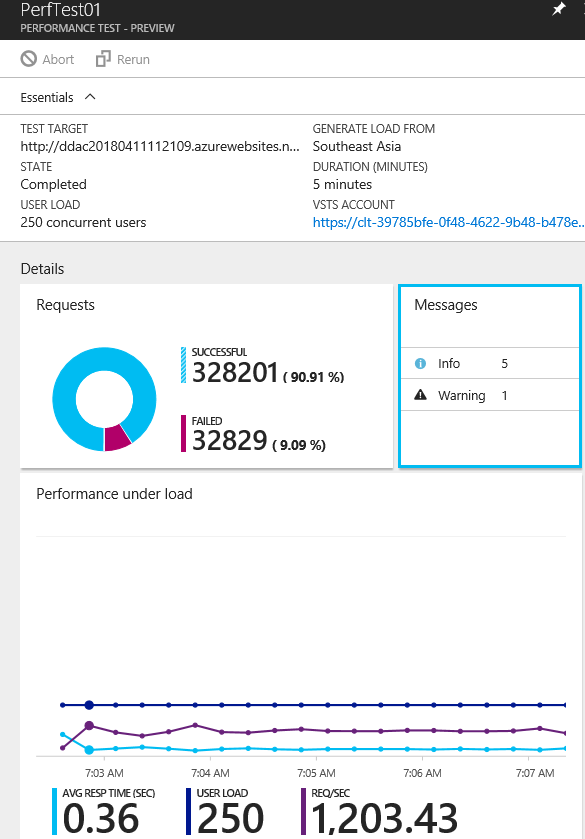


Figure Performance test – Part 1

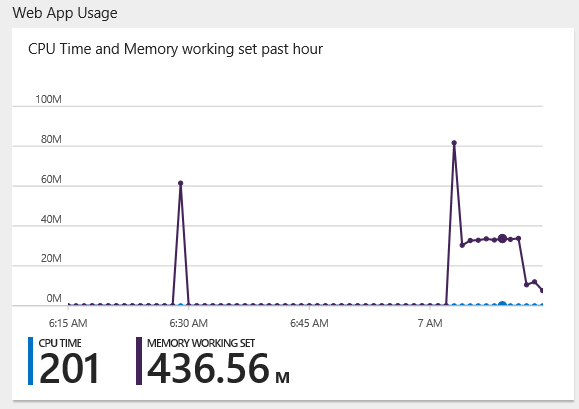


Figure - Performance test - Part 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concurrent Users | 250 | 450 | 650 | 850 |
| App Service Plan |  |  |  |  |
| S1 | 0.30 Sec  0 Failed | 1.36 Sec  0 Failed | 6.53 Sec  30 Failed | 10.56 Sec  532 Failed |
| S2 | 0.34 Sec  0 Failed | 1.36 Sec  0 Failed | 3.62 Sec  40 Failed | 8.02 Sec  302 Failed |
| S3 | 0.335 Sec  0 Failed | 1.36 Sec  0 Failed | 10.41 Sec  100 Failed | 7.41 Sec  221 Failed |

### 4.4.3 Analysis

From the data gathered by performing the performance test above, it can be deducted that, the longer the period, the higher the possibility in getting more accurate result. Also, the more service plan you have for the application, the better handling for performance for more concurrent users. This is because, as can be seen, the more test failures when it comes to S2 and S1, S3 and S2. The higher app service plan can get more accurate result. Which means, the more concurrent users on 850 and 650 can result of different data yet more accurate information. As the data generated from the performance testing as above, S1 can reliable to work on when it has 450 concurrent users, whereas when the App Service for S2 and S3 can work for 650 users. Which S1 will result least failure when comparing to S2 and S3.

## 4.5 Managed Databases

When it comes to cloud computing, it is not common for existing the terms Saas, laaS and PaaS. But what is actually are those terms represent in cloud computing, and will it be effectively causing to your development. Yes, when it comes Azure SQL database that is a SQL database native to the cloud, also known as a platform as a service (PaaS) database or a database or a database as a service DBaaS that is optimized for software-as-a-service (SaaS) app development (Microsoft Azure, 2018). PaaS also known as platform as a service, which is a complete development and deployment environment in the cloud, with resources that enables you to deliver application from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications (Microsoft Azure, 2018). A PaaS provider hosts the hardware and software on its own infrastructure, as a result, PaaS frees users from having to install in-house hardware and software to develop or to run an application (Rouse, 2017). This works fine for the Maersk line web application, PaaS will not typically replace the business’s entire IT infrastructure, this means, it cuts coding time, as PaaS development tools can drastically cut the times required to code new application with pre-coded application components built into the platform, which components are workflow, directory service and etc.

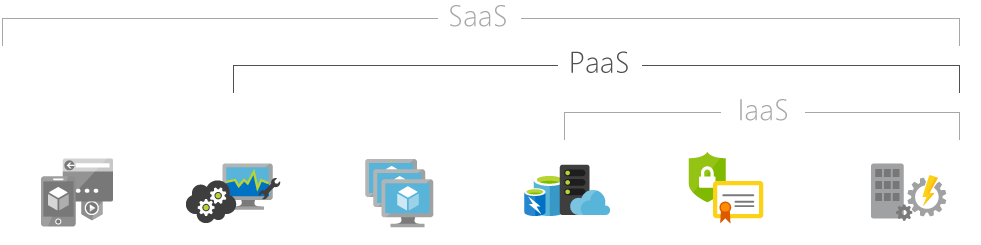


Figure Saas, PaaS and laaS Source from (Microsoft Azure, 2018)

Come back to the three main categories of cloud computing services, the other two are software as a service (SaaS) and infrastructure as a service (laaS). As can be seen in **Figure 34** above, that with laaS, which a provider supplies the basic compute, storage and networking infrastructure along the virtualization layer. Also, by having laaS provider, it supplies a range of services to accompany those infrastructure components such as detailed billing, monitoring, long access and security (Rouse, 2017). Come back to **Figure 34**, PaaS builds on the laaS model, to underlines infrastructure components, providers hsots. Manage and offer operating systems. However, when it comes to deciding laaS and PaaS, laaS users are always concerned about service resilience. As the workload’s availability and performance can be highly dependent on the provider. This means, if an laaS provider experiences internal or external system failures, the users’ workloads will be affected. Because of the multi-tenant architecture laaS has, it will negatively be causing impact on users’ workloads when it comes noisy neighbour on the architecture. Whereas, SaaS is considered a software distribution model in which a third-party provider hosts application and makes services available to the users/clients over the internet (Rouse, 2016). SaaS functions to eliminate the need for organizations to install and run applications on their own machines or in their own datacentres, this increases scalability, which give clients the option to access more or demanded services. However, Integration with software on any systems is either difficult or unsupported as businesses must rely on outside vendors to provide the software, or keep the software running.

When it comes to web app development, it should target for simplicity, the provider supplies in cloud computing services should allow users to access the web application everywhere and anywhere. Also, if the application can be accessed through internet will be one of the best option as if the provider allows internet accessed, it will speed up productivity of development and increases work performance by the employees. As a conclusion, PaaS provider works well for the Maersk line web application, again, PaaS would provide full control over the users accessing the software and the processing of data. As a result, it adds development capabilities as it efficiently manages the application life-cycle (building, testing, deploying, managing and updating) within the same integrated environment). Unlike to SaaS, the PaaS bring your own software run on the platform therefore gaining full control of the software and it effectively increases project profitability, as it can be managed consistently by having PaaS databases.

When it comes to calculate the total application costs, **when using Azure SQL database:**

The formula: **Total costs of application** = Highly minimized administration costs + software development costs + SQL Database service costs

**When using SQL server on Azure VMs**:

**Total cost of application** = Highly minimized software development cost + administration costs + SQL Server and Windows Server licensing costs + Azure Storage costs.

The formula rule for calculate total application costs for Azure SQL database and SQL server on Azure VMs are different, as SQL database pricing and virtual machine pricing are varied due to scalability of software application the developer wants. For example, if the company is big enough to have 1000 of transaction at a couple of minutes, it should have a higher service plan to support the server. Otherwise it will crash the system and resulting system failures. Therefore, it is important to knowing business profit level and scale it as if necessary to avoid system hardware or software failures and to achieve high efficiency for the clients to use the system.

# 5.0 Conclusion

In conclusion, the development of Maersk Line web application has been successfully hosted into the Microsoft Azure and by versioning through Git. The process of development of the assignment is not considered as challenging as how it should be managed through testing. This is because, when conducting performance testing, that is to avoid system downtime and system failures, this is difficult, as it is unpredictable when it comes to more people. Fortunately, there is Microsoft Azure where it can be tested on the performance and scaling through Azure web app when it has exceeded certain value of system data. Moreover, implementation of cloud architectural diagram took me some time as not all architectural is suitable for our web application. It should be effective method yet cheap solution; therefore, it is also one of the most challenging stuff when developing the cloud web application. Last but not least, by having the assignment, it gives us a clear view on how a web application should be hosted into cloud platform like Azure or AWS, and a version control like GitHub or BitBucket can easily versioning the application when it comes to new changing codes or new added requirement. It comes in handy and easy to be used in the system. Hence, learning the development tools, it will help us in building more unique and powerful application in the future or help us faster adapt to the society during employment.

# 6.0 References

Ganguli, R., 2018. *3 Types Of Web Application Architecture.* [Online]   
Available at: https://vinzite.com/3-types-of-web-application-architecture/

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

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

Microsoft Azure, 2017. *Azure Functions scale and hosting.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale

Microsoft Azure, 2017. *Event Sourcing pattern.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/architecture/patterns/event-sourcing

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

Microsoft Azure, 2018. *Choose a cloud SQL Server option: Azure SQL (PaaS) Database or SQL Server on Azure VMs (IaaS).* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas  
[Accessed 1 4 2018].

Microsoft Azure, 2018. *What is PaaS?.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/  
[Accessed 1 4 2018].

Microsoft, 2018. *1 – The Tailspin Scenario.* [Online]   
Available at: https://msdn.microsoft.com/en-us/library/hh534482.aspx

Rouse, M., 2016. *Software as a Service (SaaS).* [Online]   
Available at: https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service  
[Accessed 1 4 2018].

Rouse, M., 2017. *Infrastructure as a Service (IaaS).* [Online]   
Available at: Infrastructure as a Service (IaaS)  
[Accessed 1 4 2018].

Rouse, M., 2017. *Platform as a Service (PaaS).* [Online]   
Available at: https://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS  
[Accessed 1 4 2018].

Waight, S., 2014. *Deploy an Ultra High Availability MVC Web App on Microsoft Azure – Part 1.* [Online]   
Available at: https://blog.kloud.com.au/2014/11/03/deploy-an-ultra-high-availablity-mvc-web-app-on-microsoft-azure-part-1/