

**Asia Pacific University of Technology and Innovation**

[**D**](http://lms.apiit.edu.my/course/view.php?id=305)**esign and Developing Application on the Cloud**

**CT071-3-3-DDAC**

|  |  |
| --- | --- |
| **Project Title:** | **Maersk Line Container Management System** |
| **Intake Code:** | **UC3F1706SE** |
| **TP Number:** | **TP037694** |
| **Name:** | **Tan Li Feng** |
| **Lecturer Name:** | **DR. KALAI ANAND A/L RATNAM** |
| **Submission Date:** | **13/04/2018** |

# Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our DDAC Lecturer, DR. KALAI ANAND A/L RATNAM, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report and every comment and advice. He had invested her full effort in guiding me in achieving the goal.

Table of Contents

[Acknowledgement 2](#_Toc511353766)

[1.0 Introduction 5](#_Toc511353767)

[1.1 Project Background 5](#_Toc511353768)

[1.2 Objective & Scope 6](#_Toc511353769)

[1.3 Requirements 6](#_Toc511353770)

[1.4 Deliverables & Fulfillment 7](#_Toc511353771)

[2.0 Project Plan 8](#_Toc511353772)

[3.0 Design 9](#_Toc511353773)

[3.0 Use Case Diagram 9](#_Toc511353774)

[3.1 Use Case Specifications 10](#_Toc511353775)

[3.2 Activity Diagram 14](#_Toc511353776)

[3.2.1 Login 14](#_Toc511353777)

[3.2.2 Change Password (Agents) 15](#_Toc511353778)

[3.2.3 View Booking (Agents) 16](#_Toc511353779)

[3.2.4 View Customer (Agents) 17](#_Toc511353780)

[3.2.5 View Schedules (Agents) 18](#_Toc511353781)

[3.2.6 View Agents (Admins) 19](#_Toc511353782)

[3.2.7 View Bookings (Admins) 20](#_Toc511353783)

[3.2.8 View Schedules (Admins) 21](#_Toc511353784)

[3.3 Sequence Diagram 22](#_Toc511353785)

[3.3.1 Logins 22](#_Toc511353786)

[3.3.2 View Schedule(Agents) 23](#_Toc511353787)

[3.3.3 Change Password (Agents) 24](#_Toc511353788)

[3.3.4 View Customers (Agents) 25](#_Toc511353789)

[3.3.5 View Bookings (Agents) 26](#_Toc511353790)

[3.3.6 View Schedule(Admins) 27](#_Toc511353791)

[3.3.7 View Agents(Admins) 28](#_Toc511353792)

[3.3.8 View Bookings (Admins) 29](#_Toc511353793)

[3.4 Class Diagram 30](#_Toc511353794)

[3.5 Data Dictionary 31](#_Toc511353795)

[3.6 Cloud Architecture 33](#_Toc511353796)

[3.7 Design Consideration 35](#_Toc511353797)

[3.7.1 Data information could be achieved the accurate 35](#_Toc511353798)

[3.7.2 HTTP Sessions Not Persisted or Replicated 35](#_Toc511353799)

[4.0 Implementation 36](#_Toc511353800)

[4.1 Application Development 36](#_Toc511353801)

[4.1.1 User Interface Example (Bootstrap) 38](#_Toc511353802)

[4.1.2 Example of Password Validation Function (JavaScript) 40](#_Toc511353803)

[4.1.3 Example Code of Register Agent Account to Database (Servlet) 41](#_Toc511353804)

[4.2 Azure Publishing 42](#_Toc511353805)

[4.2.1 Create New Web Service On Azure 42](#_Toc511353806)

[4.2.2 Azure SQL Server and Database Setup 43](#_Toc511353807)

[4.2.3 Web Application Publishing 45](#_Toc511353808)

[4.3 Application Scaling 47](#_Toc511353809)

[4.3.1 Web App Scale 47](#_Toc511353810)

[4.3.2 SQL Database Scale 48](#_Toc511353811)

[4.4 Managed Database (Paas) 49](#_Toc511353812)

[5.0 Testing 51](#_Toc511353813)

[5.1 Unit Testing 51](#_Toc511353814)

[5.2 Load Performance Testing 56](#_Toc511353815)

[5.2.1 Result of 20 User Load 57](#_Toc511353816)

[5.2.2 Result of 30 User Load 58](#_Toc511353817)

[5.2.3 Result of 40 User Load 59](#_Toc511353818)

[5.2.4 Analysis 60](#_Toc511353819)

[6.0 Conclusion 61](#_Toc511353820)

[References 62](#_Toc511353821)

[Appendix 63](#_Toc511353822)

[Project Links 63](#_Toc511353823)

[Test Account for demostration 63](#_Toc511353824)

# Introduction

## Project Background

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

Operating in 100 countries and transporting goods around the globe, at first glance it would appear Danish shipping company Maersk Line is already handling all the cargo it can manage. But when Maersk determined that the volume of most of the goods it was shipping had grown to full capacity, the company decided that cloud powered solutions would be a crucial part of rectifying the situation.

“There was a ‘mind-opener’ where Maersk said, ‘How can we support the overall business strategy, and also from an IT perspective,” says Soeren Lorenzen, an account general manager with Hewlett-Packard company who is involved 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.”

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

Therefore, in this assignment, the resulting Maersk Line Online Container Management System will be developed and deployed onto the Azure cloud platform. With the advent of cloud services, it is no surprise that such a decision was made. The application provides Maersk Line’s staffs the ability to manage shipping schedule and book vessel online.

## Objective & Scope

Designing and developing a Container Management System (CMS) to cater to manage the containers, a solution that reduces overall supply chain costs and an efficient way to manage logistics.

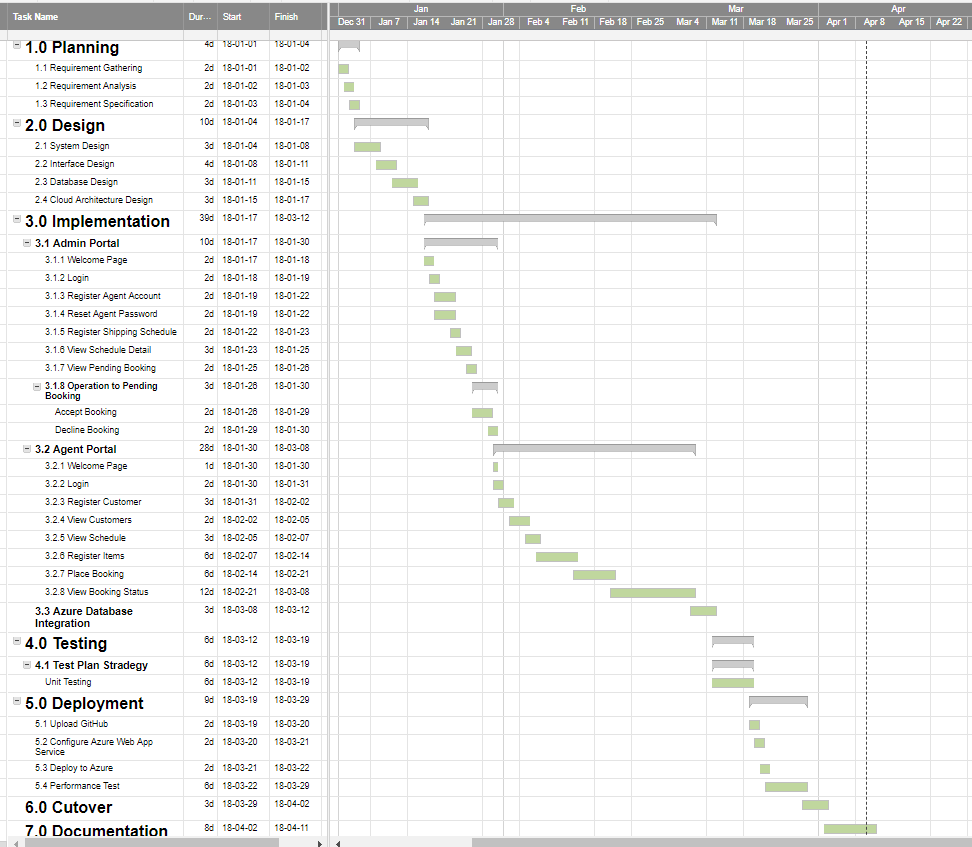
## Requirements

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 entire booking process from schedule search to booking confirmation.

## Deliverables & Fulfillment

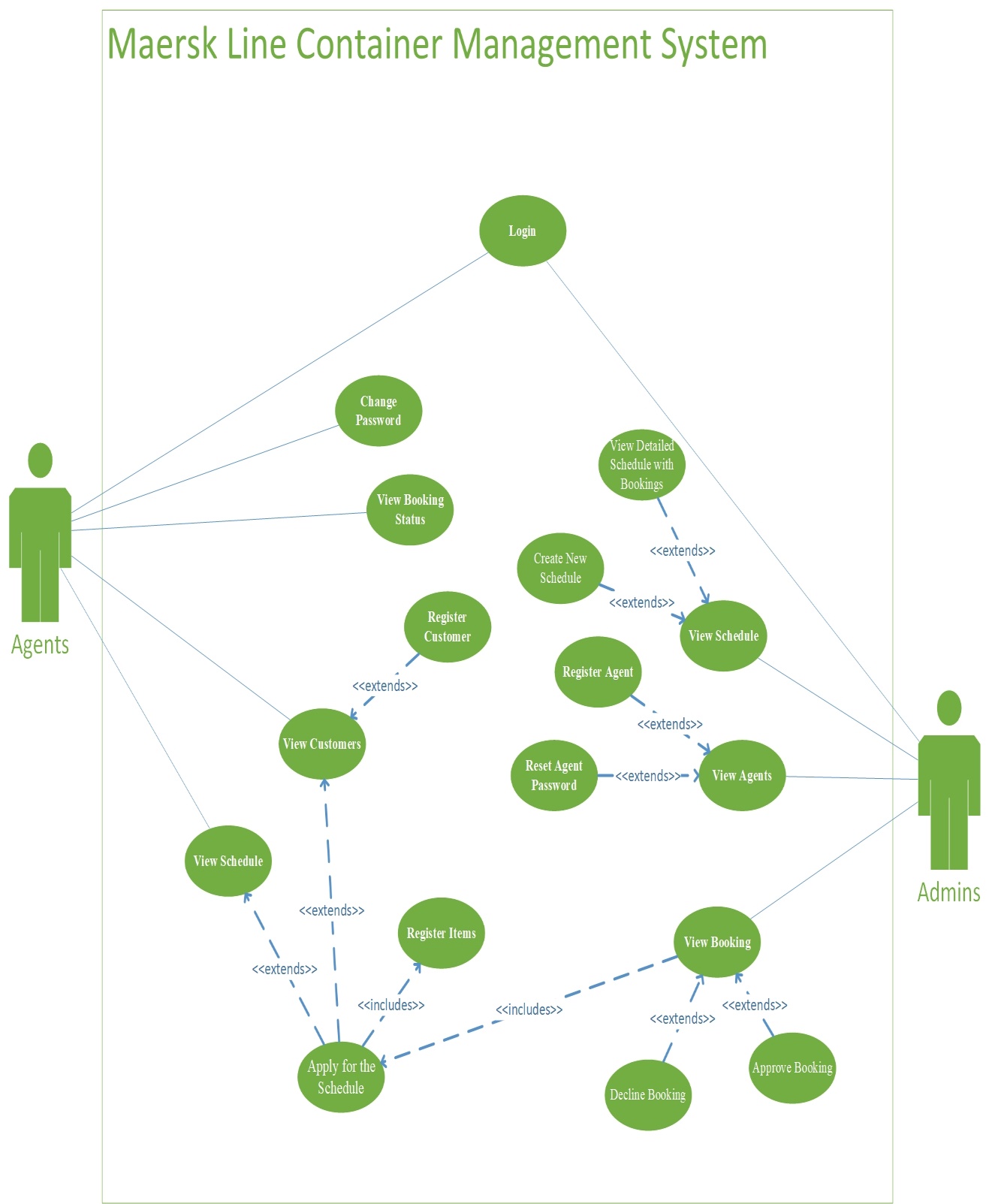
1. Design & Develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App) or on AWS Elastic Beanstalk.
2. Consume Relational Database
3. Consist of 5 - 10 interlinked pages.
4. Provide quality content and design.
5. Analyze web application performance with monitoring tools.
6. To be able to scale the solution to meet the needs of demands during peak seasons.
7. Source code to place in source control management services.

# Project Plan

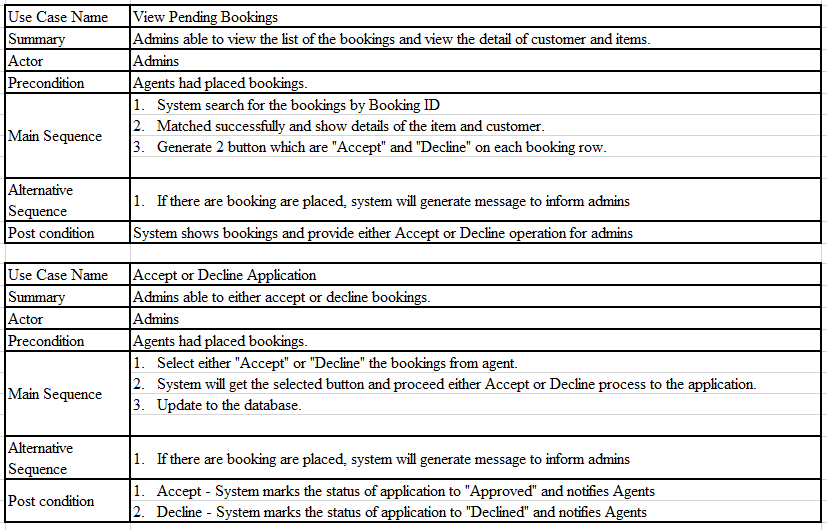
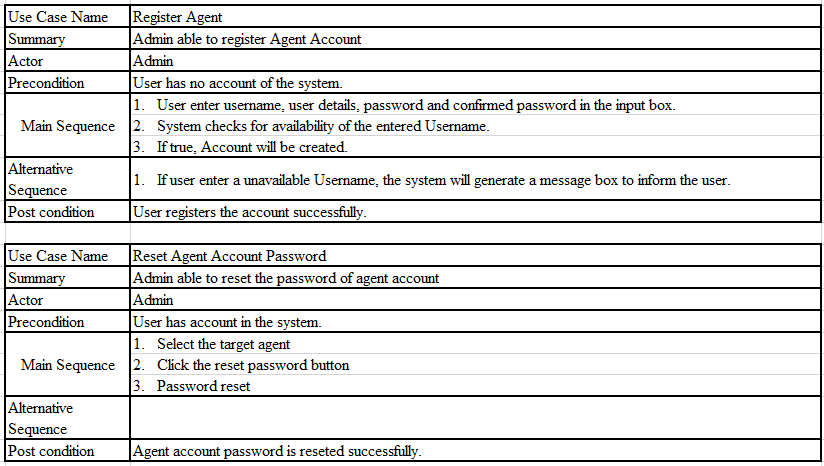
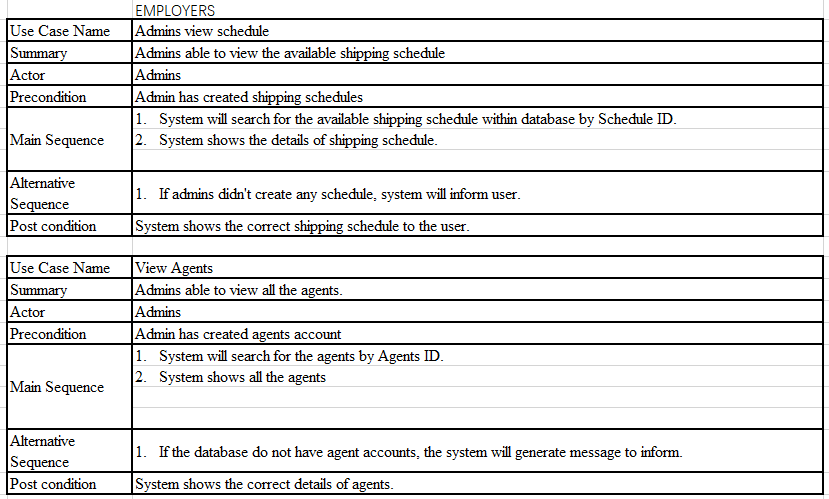
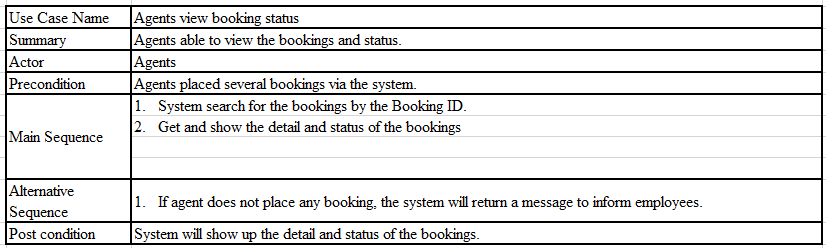
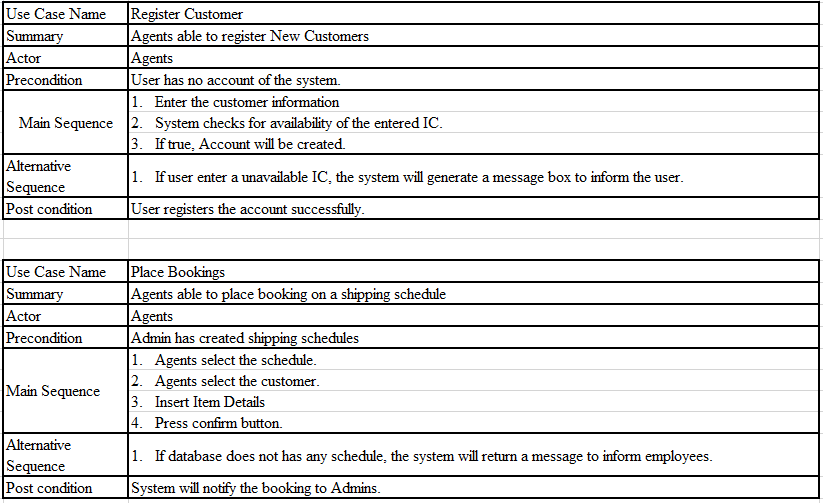
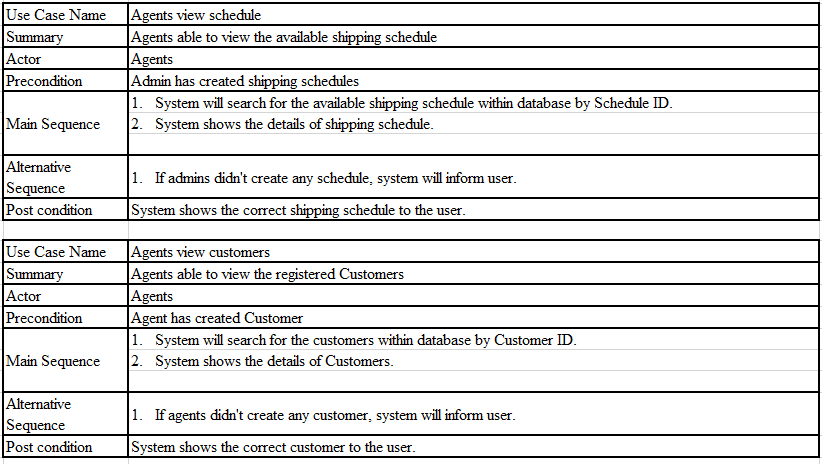
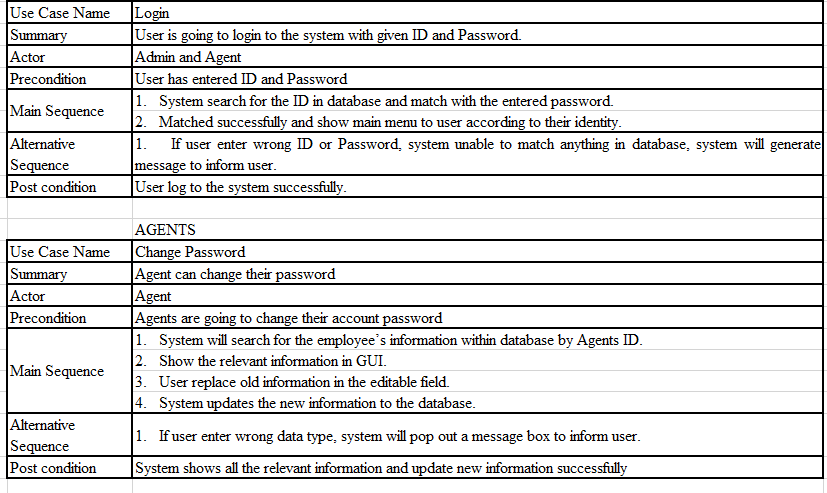


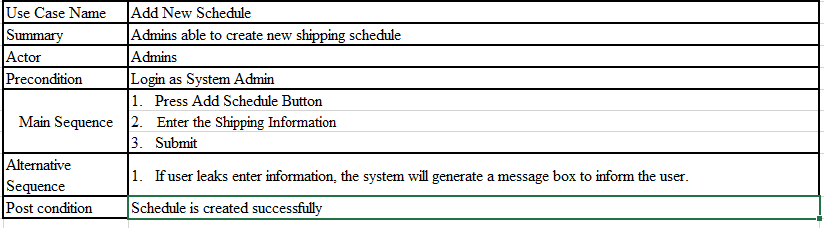
# Design

## Use Case Diagram



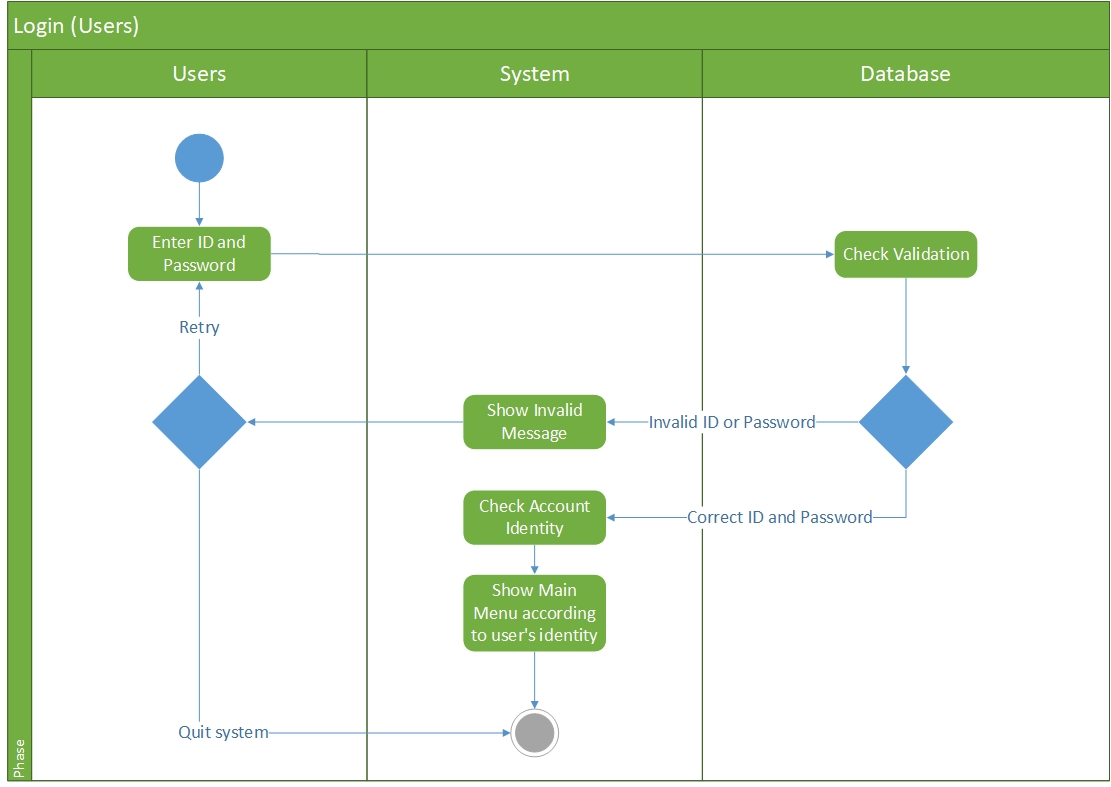
## Use Case Specifications



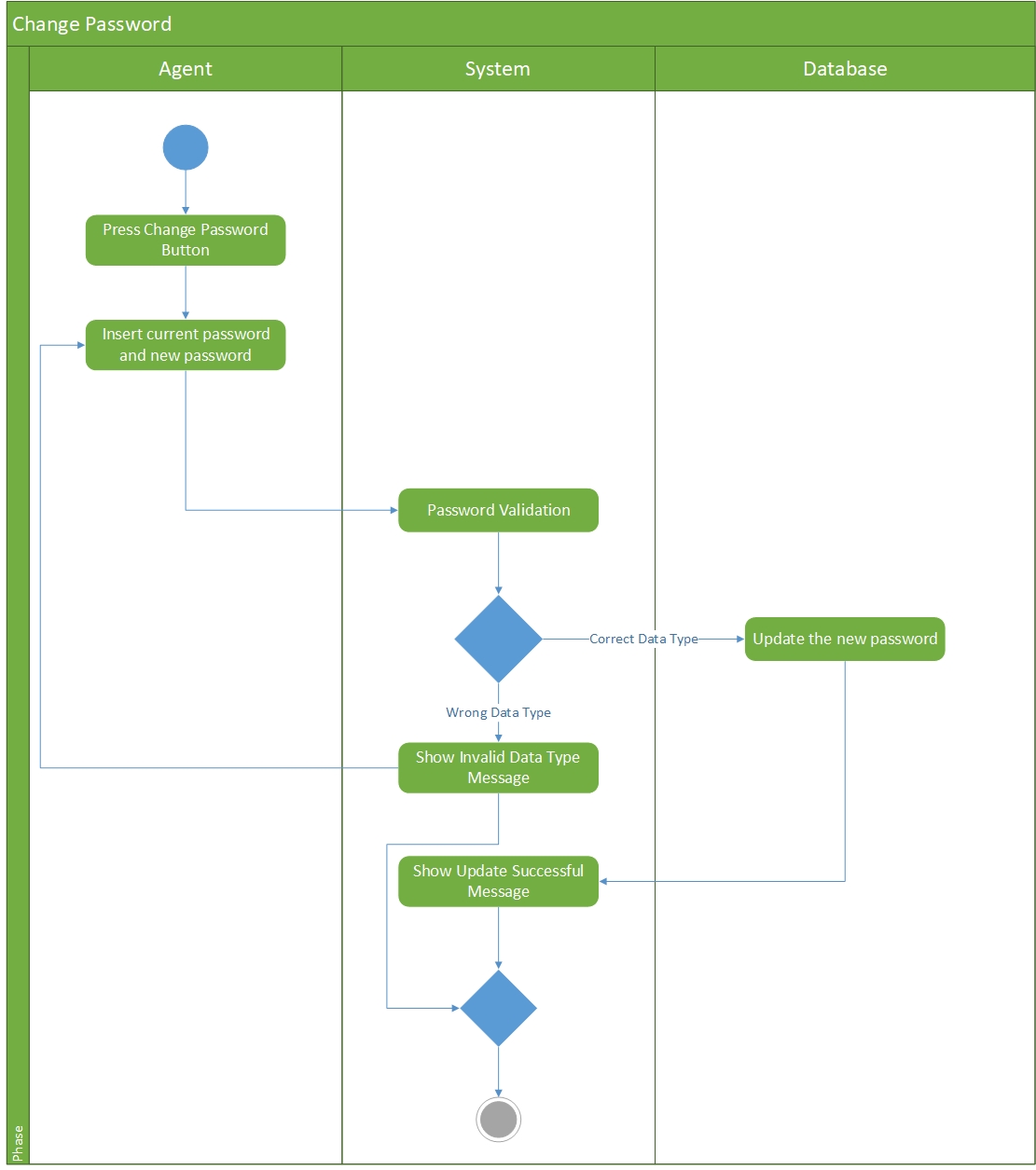


## Activity Diagram

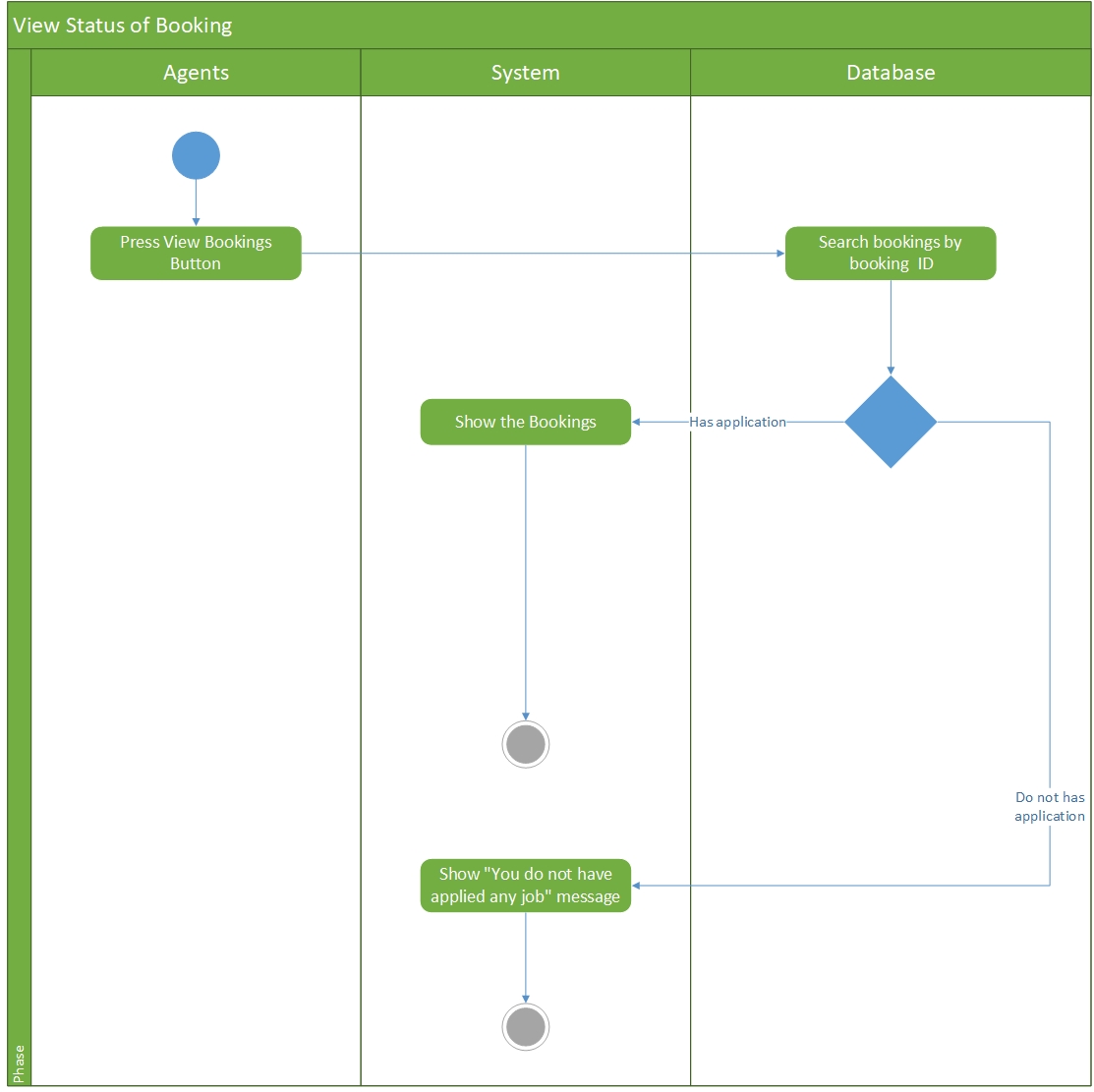
### Login



### Change Password (Agents)

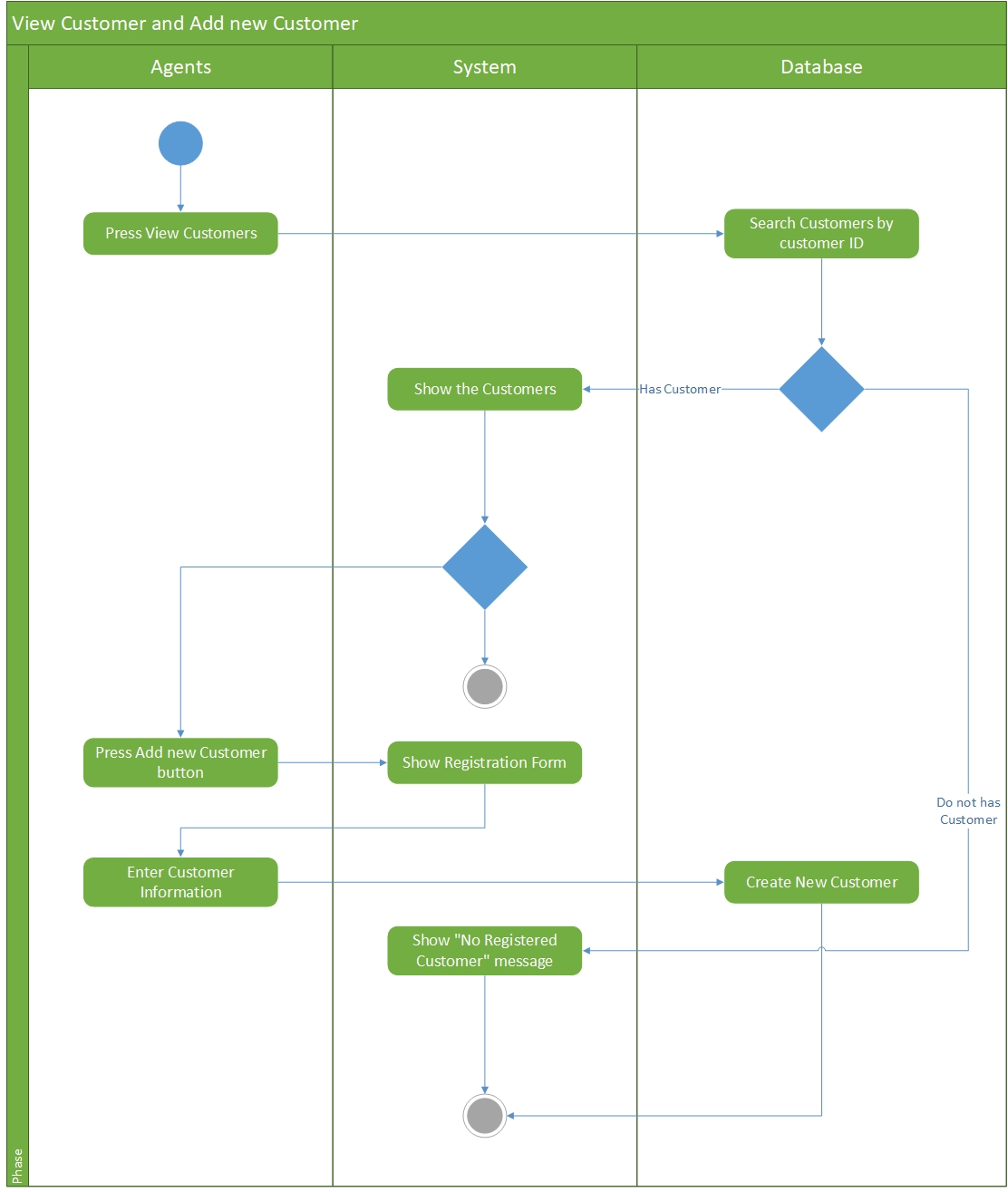


## View Booking (Agents)

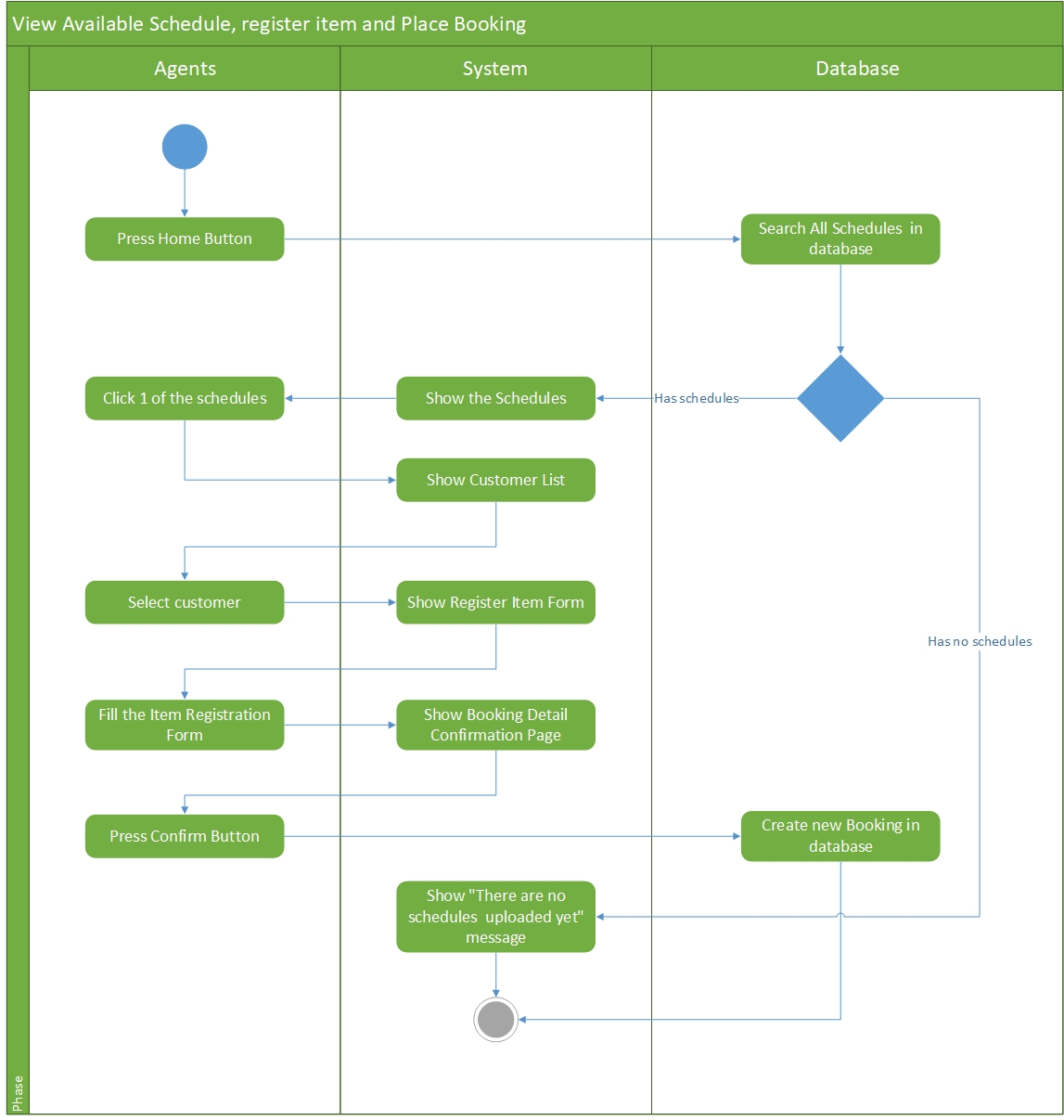


Show “No Bookings are placed” message

### View Customer (Agents)



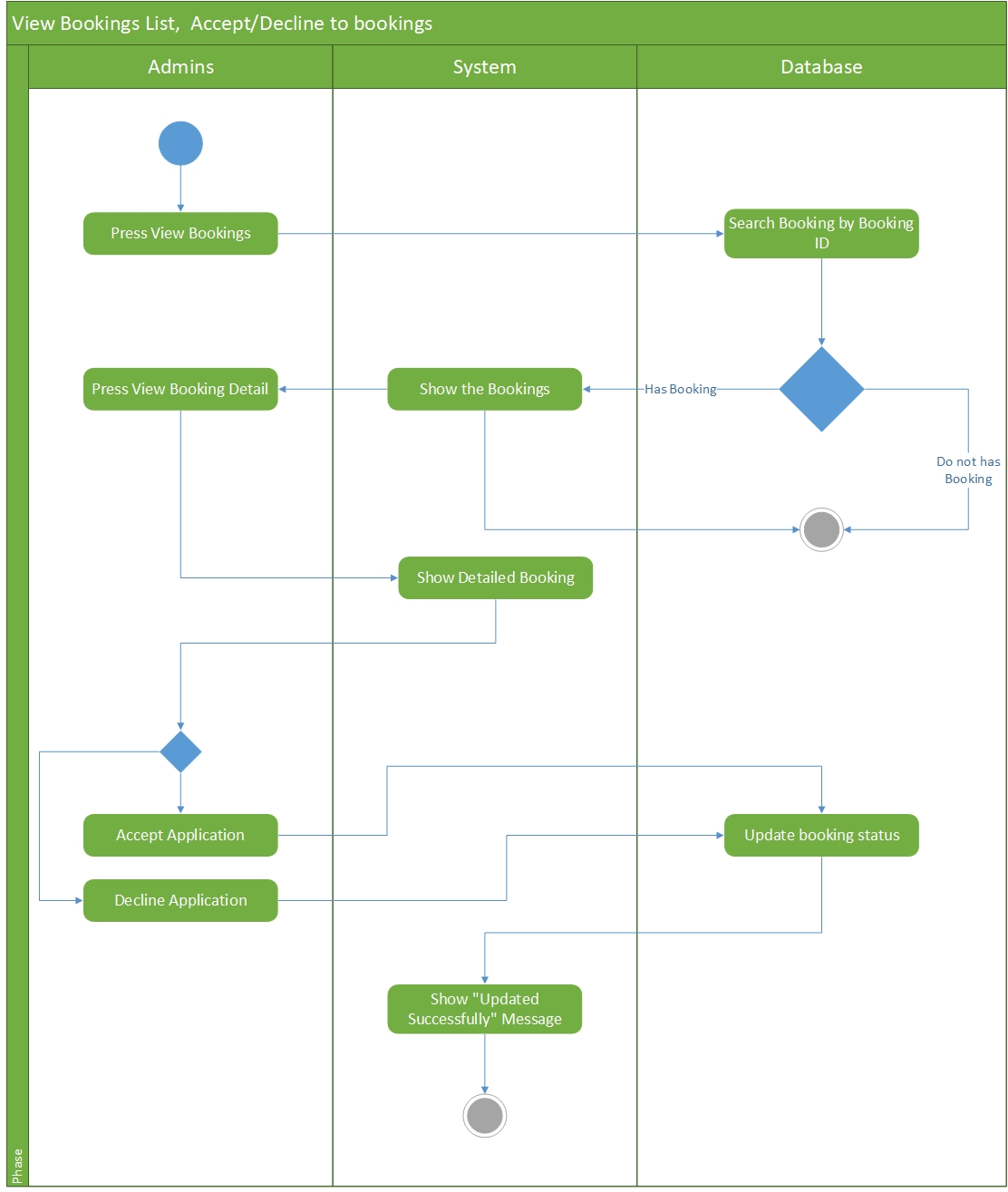
### View Schedules (Agents)



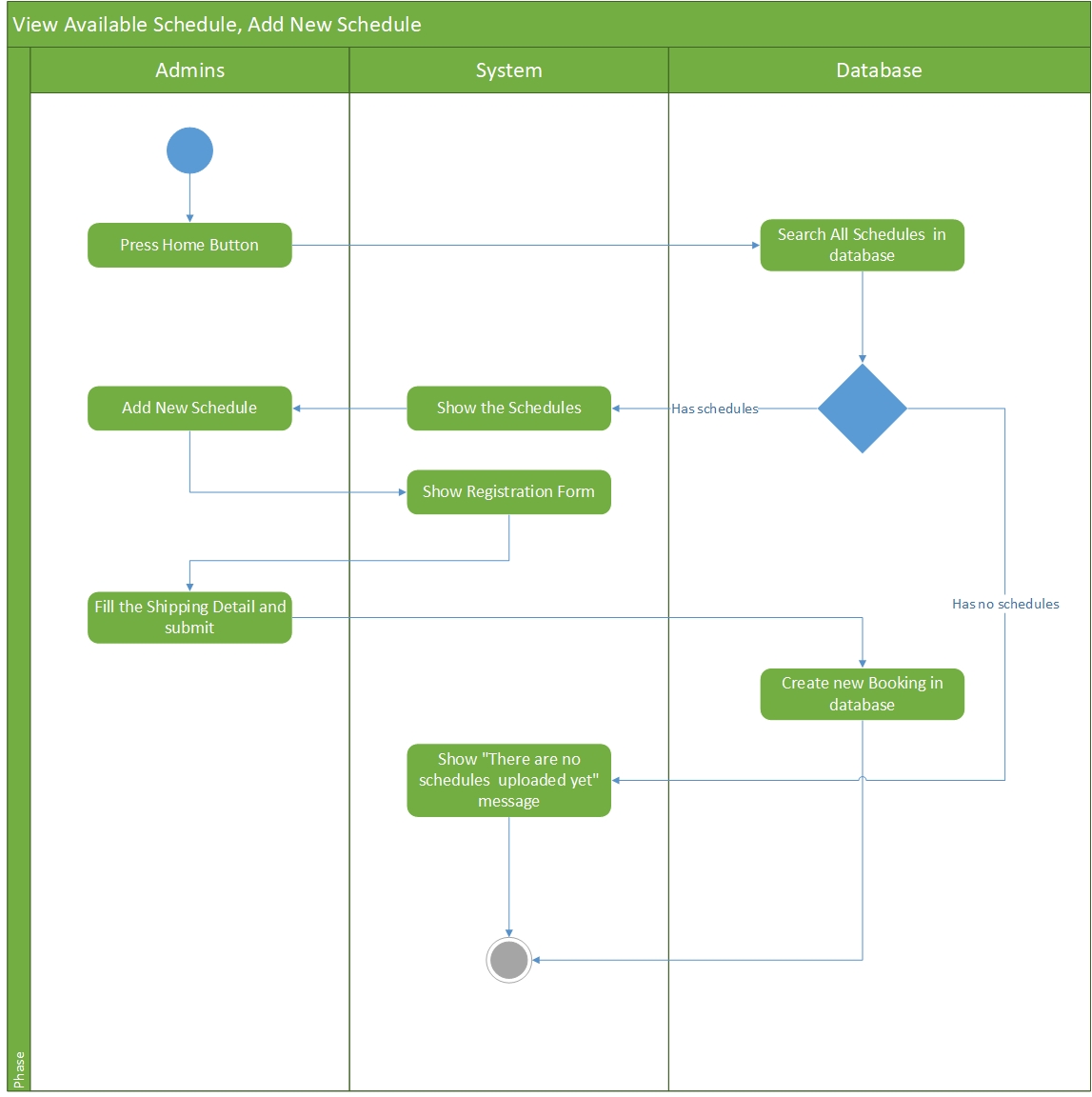
### View Agents (Admins)



### View Bookings (Admins)

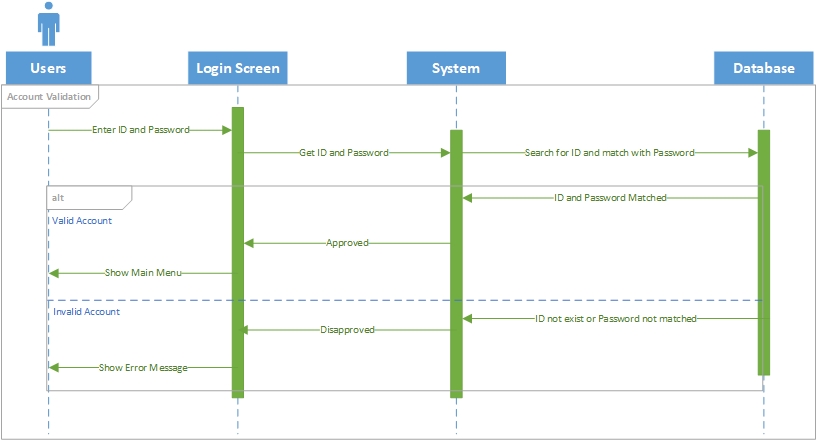


### View Schedules (Admins)

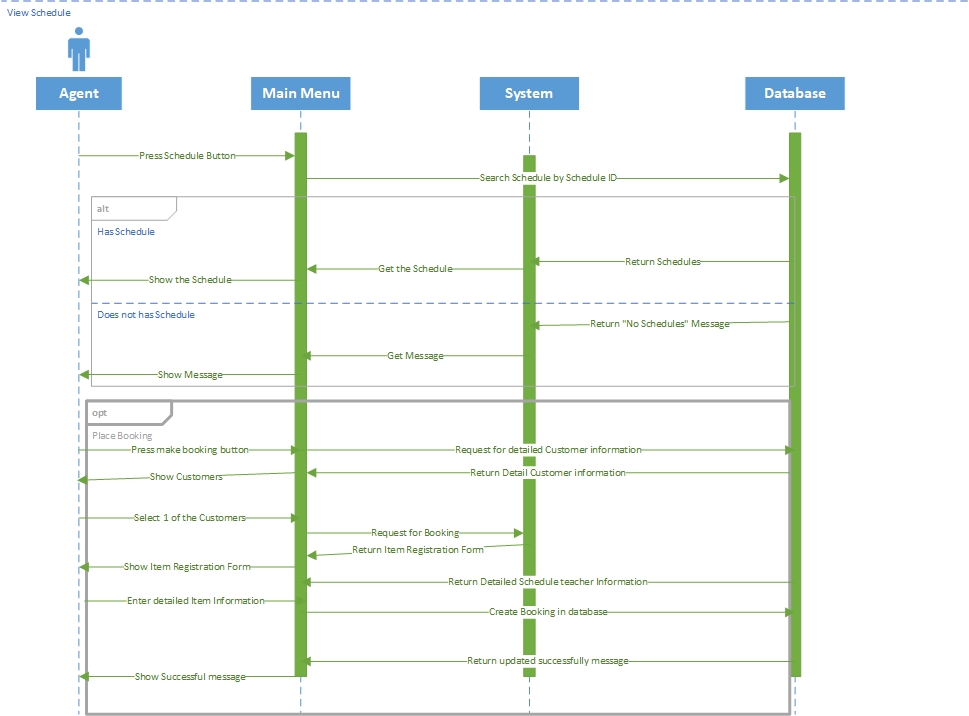


## Sequence Diagram

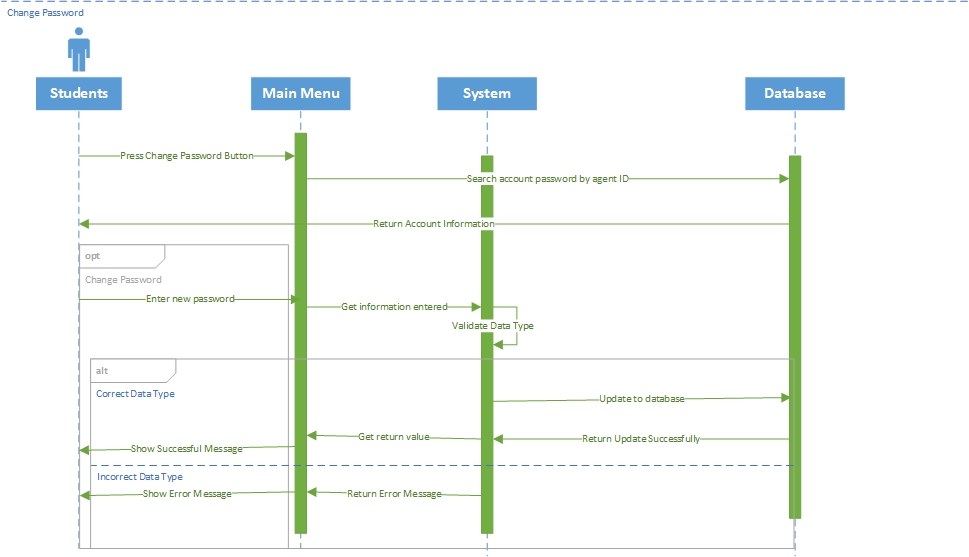
### Logins



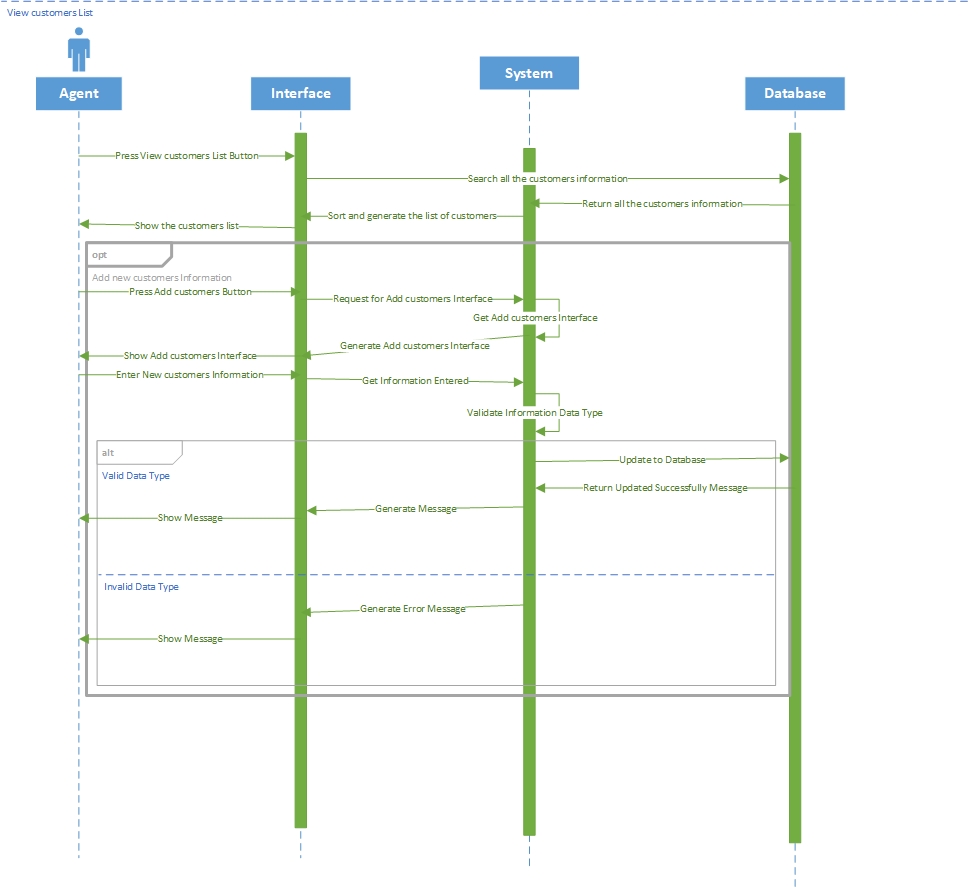
### View Schedule(Agents)



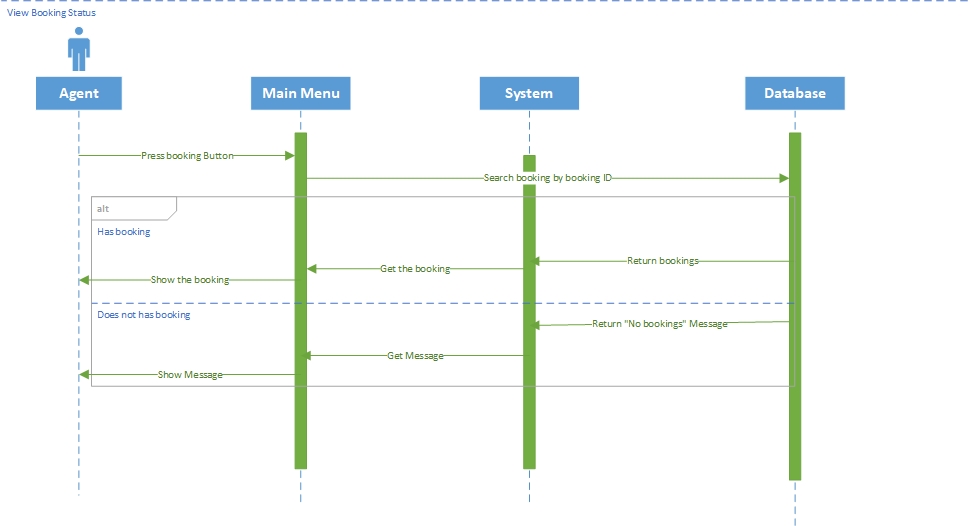
### Change Password (Agents)



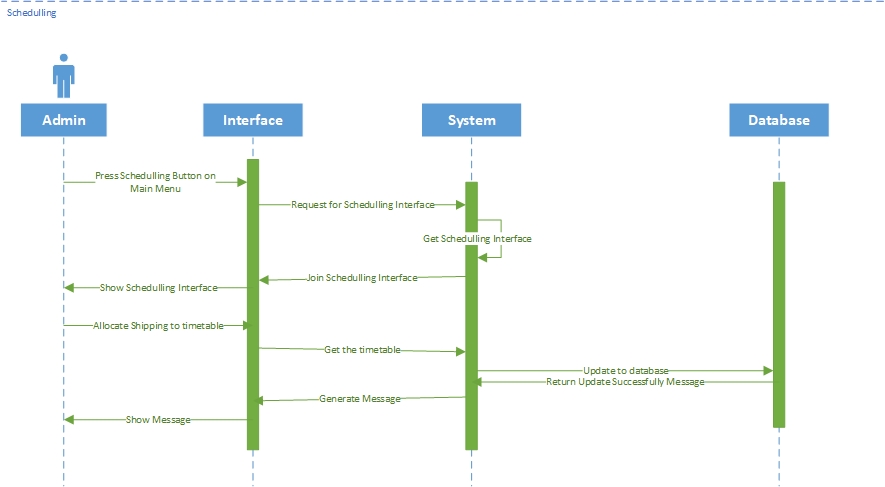
### View Customers (Agents)



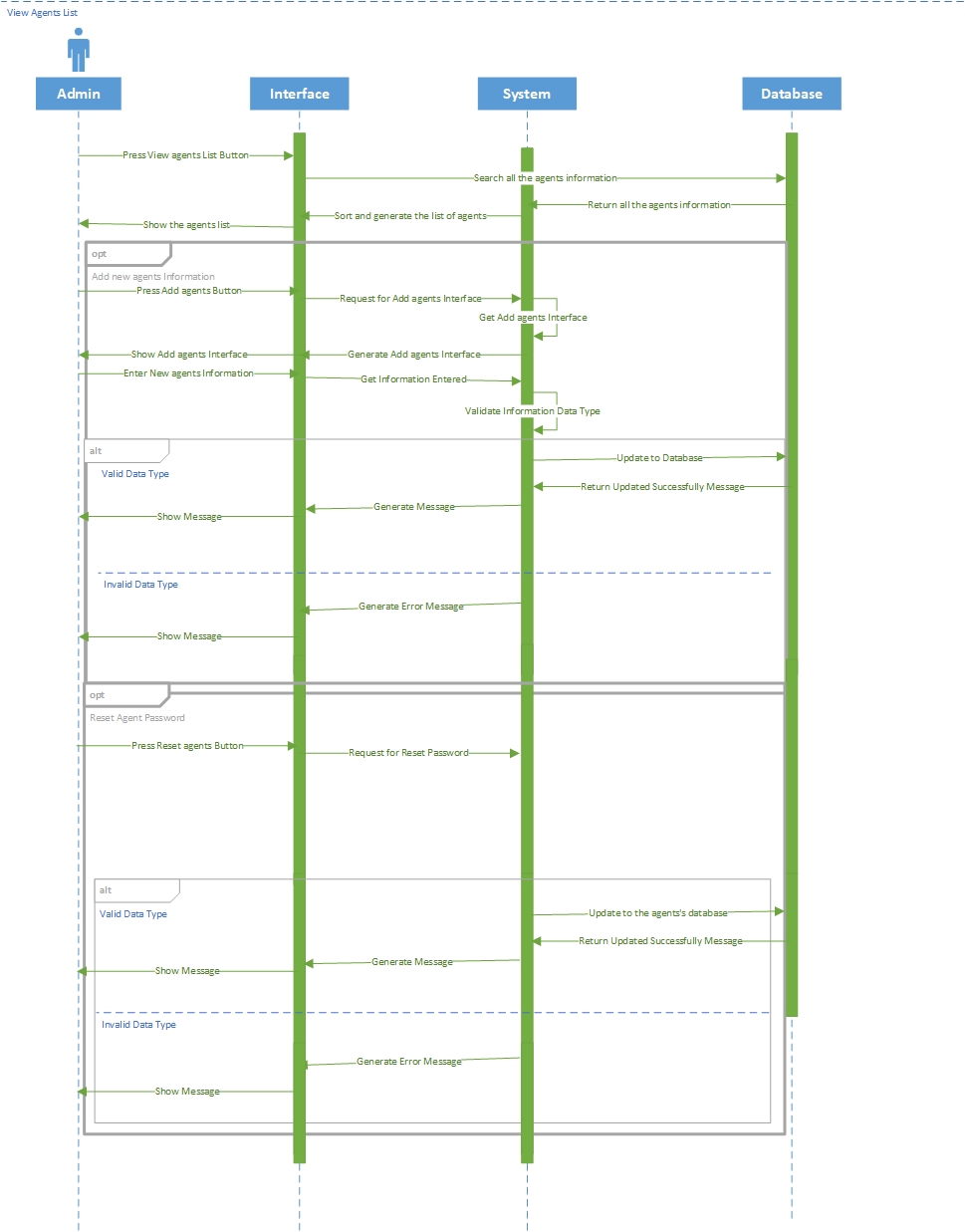
### View Bookings (Agents)



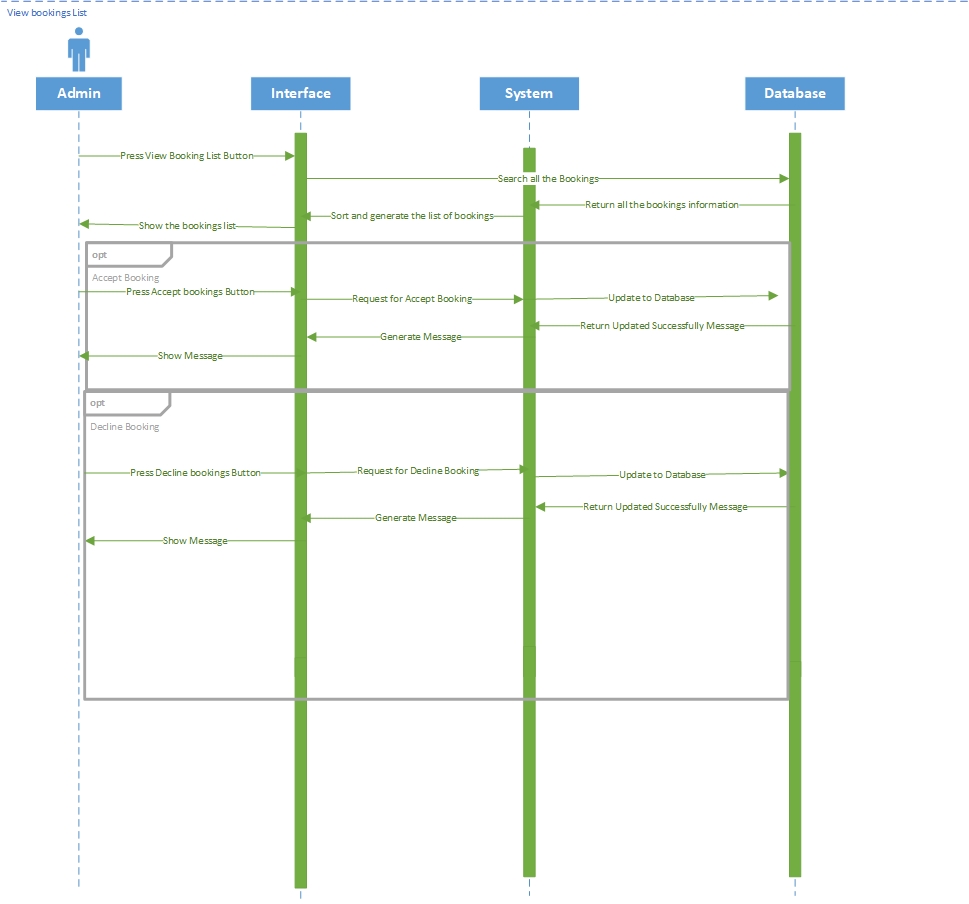
### View Schedule(Admins)



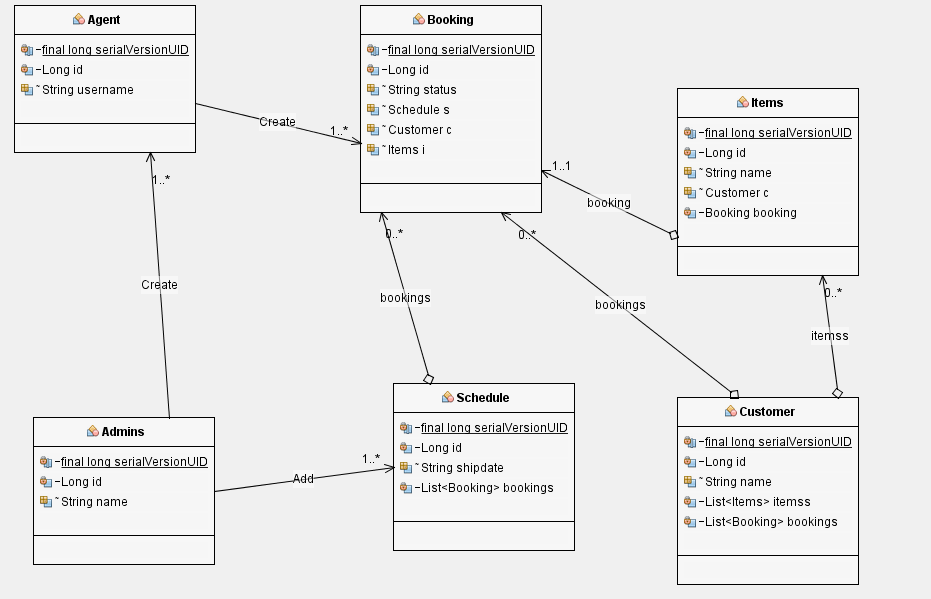
### View Agents(Admins)



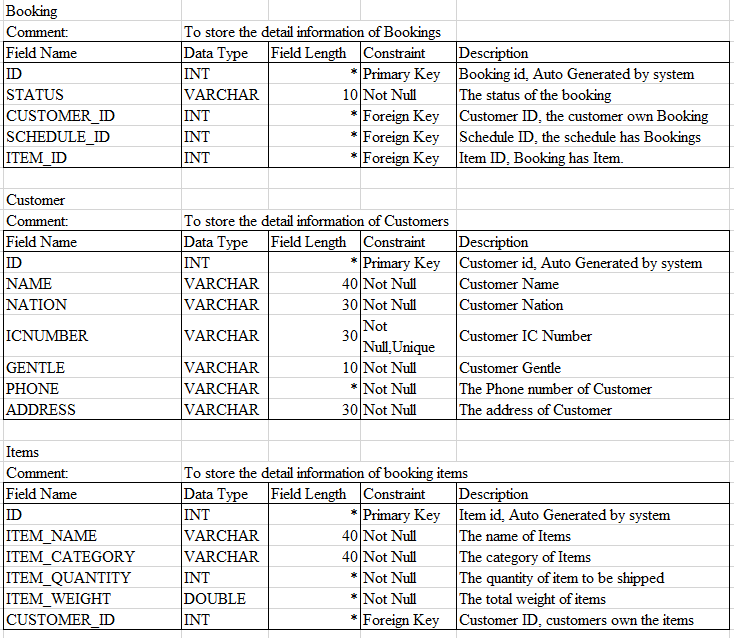
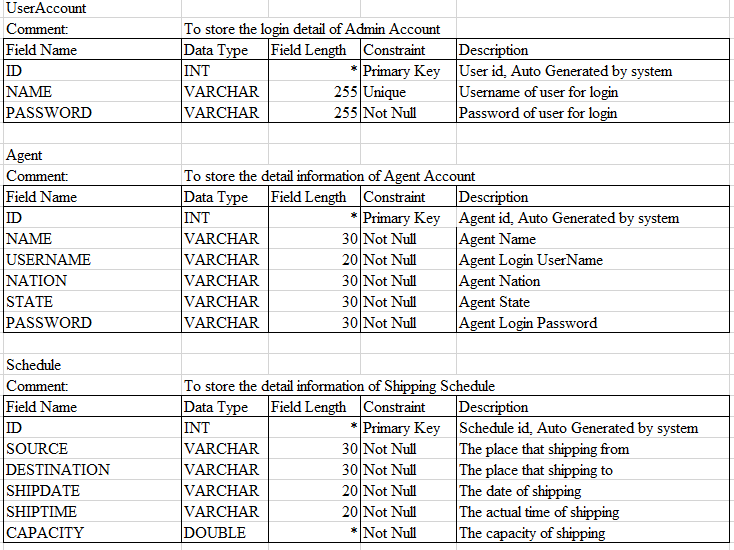
### View Bookings (Admins)



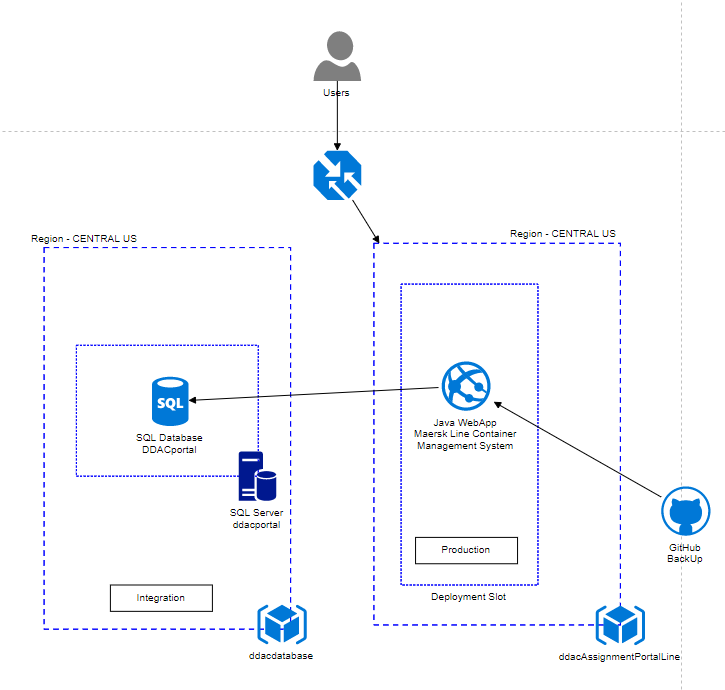
## Class Diagram



## Data Dictionary



## Cloud Architecture



The initial implementation of the Maersk Line Container Management System in the cloud environment focused primarily in the Central US region, and developer placed the main web application in one of the resource groups. The GITHUB will be holding the entire system as a backup for unexpected system crash. Among the services registered in the web application, there is a web service that provides a container for holding the Maersk Line Container Management System, and services can be scaled out according to the server's performance needs and future needs. Besides, the Azure database in a SQL Server database is another important function necessary to keep the data necessary for daily operation. Because of the first method of the database is applied during development, SQL Server is set to a different resource group, and the web application can only connect to the SQL database after setting up the firewall rule in the SQL Server to achieve data security.

## Design Consideration

### Data information could be achieved the accurate

In the design of the website, the developer sets all information fields and performs text validation to prevent empty data types or incorrect data types from storing information. The Empty Textbox field will cause all users to have a common error in all applications. Therefore, the developer designs all web applications in the input text field using the verification method as a diagram design to avoid blank mistakes in the input text field.

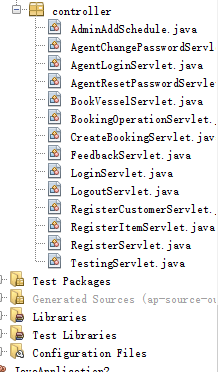
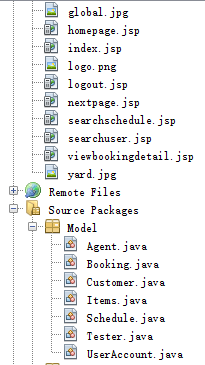
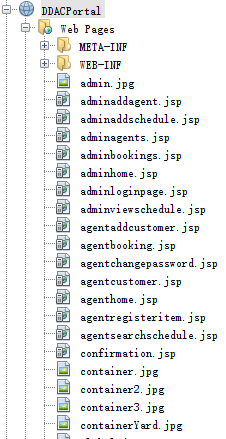
### HTTP Sessions Not Persisted or Replicated

The Maersk Line Container Management System must support session affinity or sticky sessions for incoming HTTP requests to applications when session cookies are used. If multiple instances of an application are running on Maersk Line, all requests from a particular client are routed to the same application instance. This allows the web application container and framework to store session data specific to each user session. Maersk Line does not persist or replicate HTTP session data. When a user session persistent of an instance termination or crash makes another HTTP request, the request is routed to another instance of the application.

# Implementation

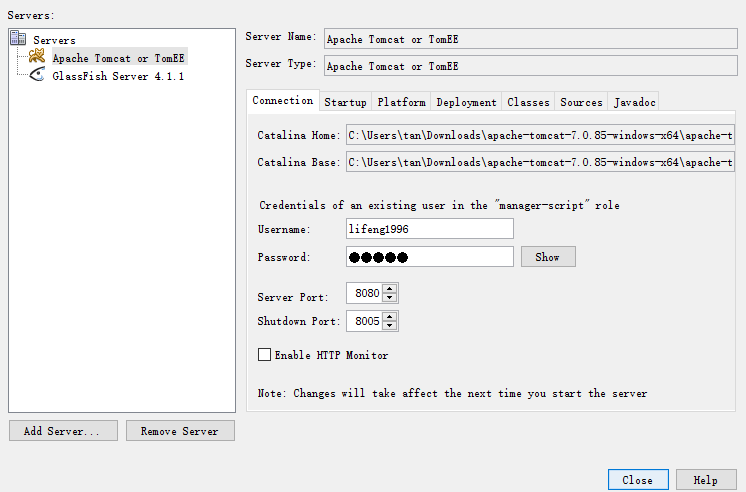
## Application Development

Development of the Maersk Line Container Management System is implemented in Java Programming Language and using the HTML to implement the user interfaces. In Java Web app development environment, it mainly breaks down the application into three files which are Model, Controller and WebSites. Besides, this web application needs to integrate with SQL database to store and retrieve the data needed.



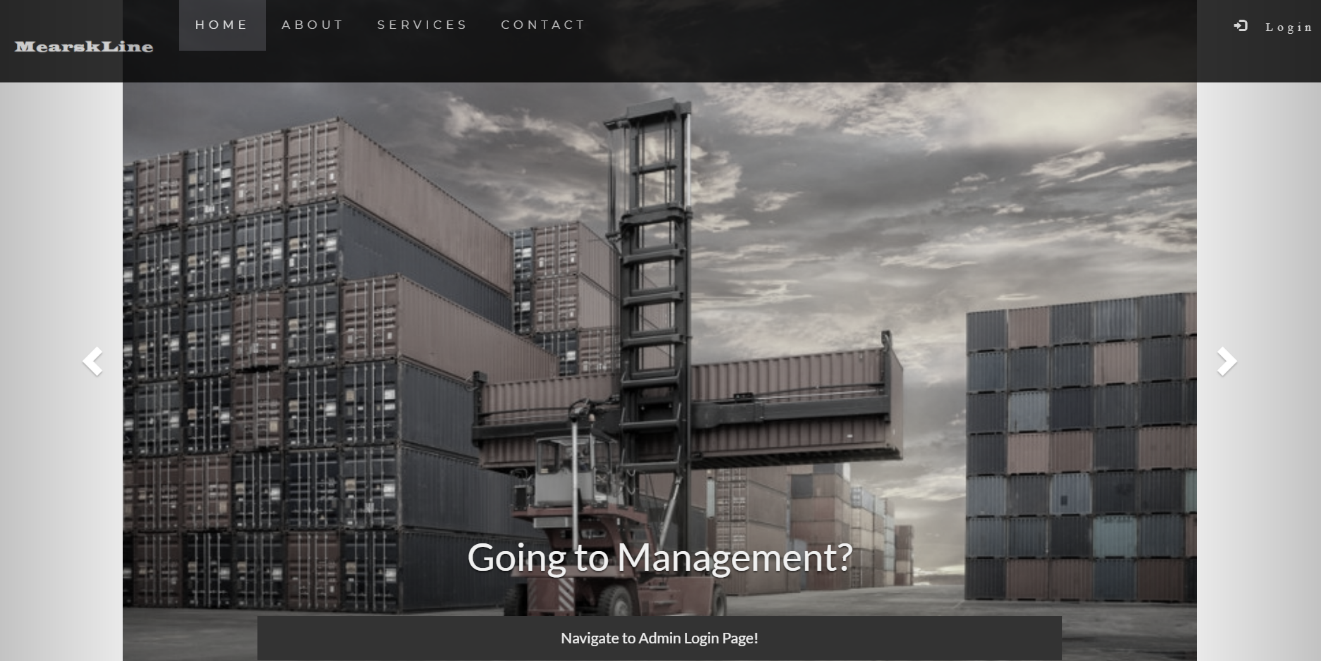
The Model file is used to store the Entity Classes that created for the system; The controller file is storing the Java Servlet which is used to process some of the backend progress in the system; While the WebSites file is holding all the user interfaces file (jsp files).

Besides, most of the design and function of the websites are using the Bootstrap and Javascript. As the prerequisite of the web application development, the server container is playing an important rule which enable a web application can be run well. The web server container is decided to user Apache Tomcat 7.0 since this version is relatively more stable than the other latest version.



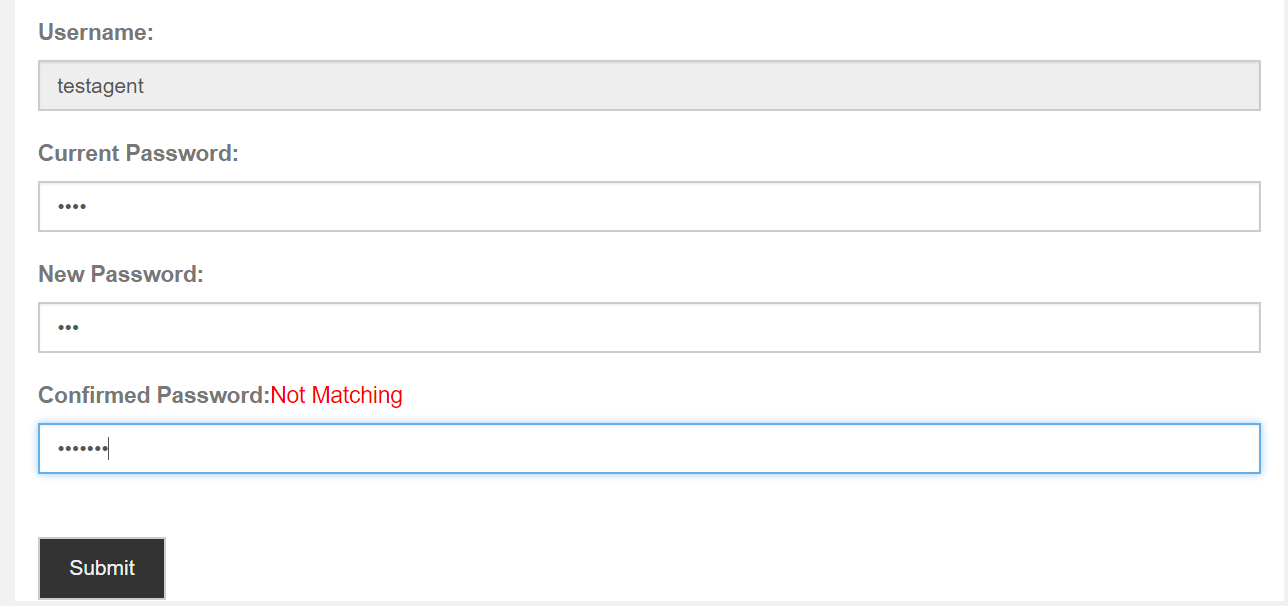
### User Interface Example (Bootstrap)

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. It allows developers can quickly prototype ideas and build the whole application quickly using Sass variables and mixins, highly responsive grid systems, rich embedded components, and powerful plugins built with jQuery (Bootstrap, 2018). In the web application development, Bootstrap provides a convenience for developers to structure the web site layout.



### Example of Password Validation Function (JavaScript)

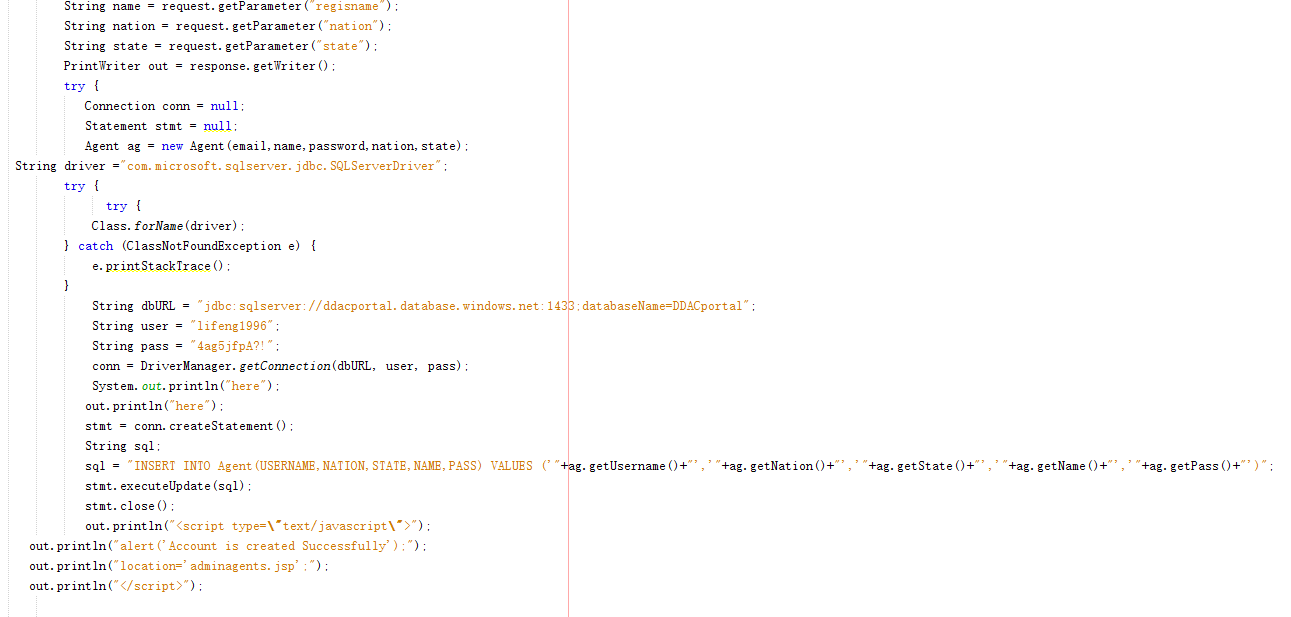
JavaScript (JS) is a lightweight, interpreted programming language with first class features (Mozilla, 2018). In the browser environment best known as the scripting language of the web page, it uses this environment other than many browsers, such as node.js and Apache CouchDB. JS is a prototype-based multi-paradigm dynamic scripting language that supports object-oriented, imperative, declarative style (such as functional programming) style (Mozilla, 2018). In this web application development, JavaScript is being used frequently to achieve several real-time and dynamic functions without communicate to the Java Servlet such as password checking, table filtering and Username Availability.





### Example Code of Register Agent Account to Database (Servlet)

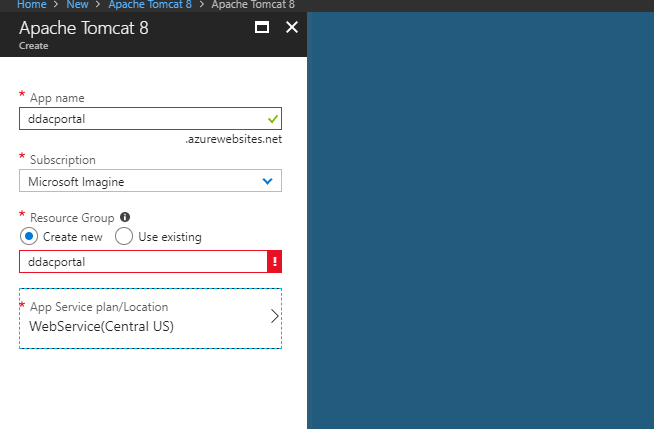
The servlet code is designed to get all the information inserted by the user from the website and process the registration. Firstly, the system will get the username and check the availability of the username in database. If the username is available, the system will create the account and update to the database; Else the system will show “Username is not available” message to the user.



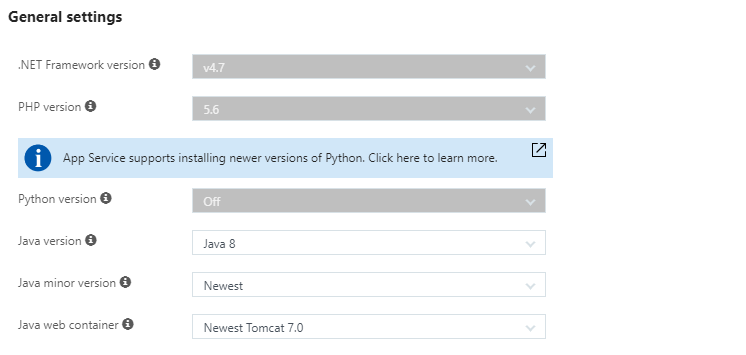
## Azure Publishing

### Create New Web Service On Azure

First, create a new Tomcat Web Server Container, register the application container, obtain the application name, subscription, resource group, application service plan and region, operating system needed. The important point is that because the application name is unique and it is necessary to analyze the visitor flows in different areas appropriately, in order to maintain the reliability of the application, the pricing layer most suitable for the Web application is registered.



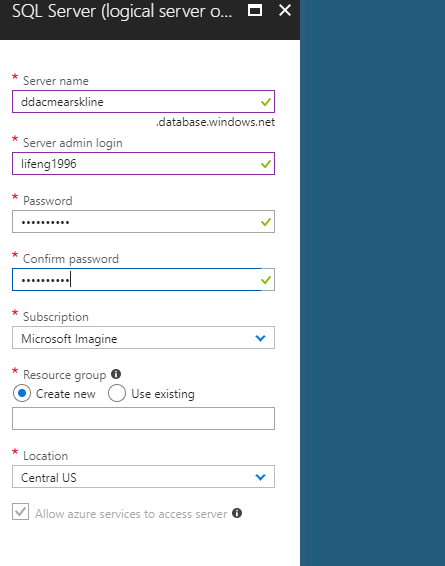
Besides, since this project system is developed in Java Language hence a proper setup of Java Language Environment is required and important in the web service configuration.



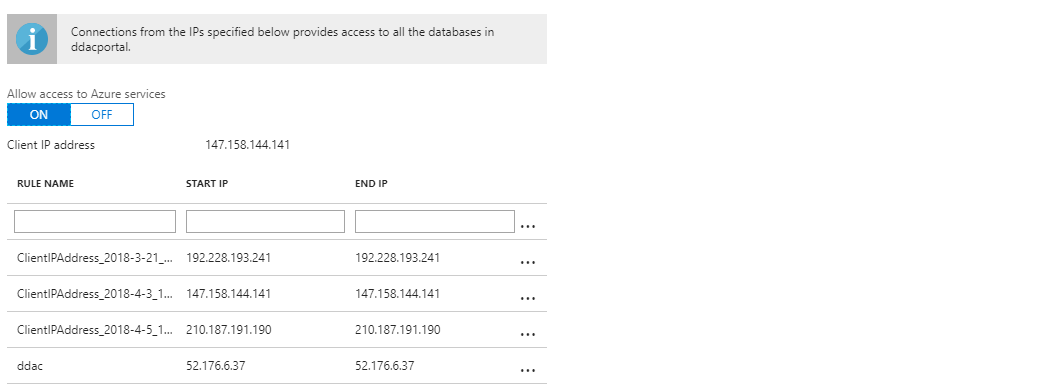
### Azure SQL Server and Database Setup

Next, set up an SQL database to store and retrieve data for web applications. The server is also created here and hosts databases of a specific region with the user name and password assigned to access the server.

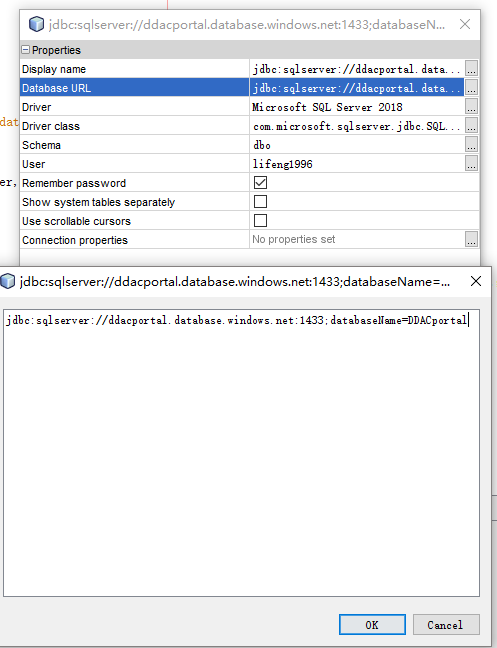
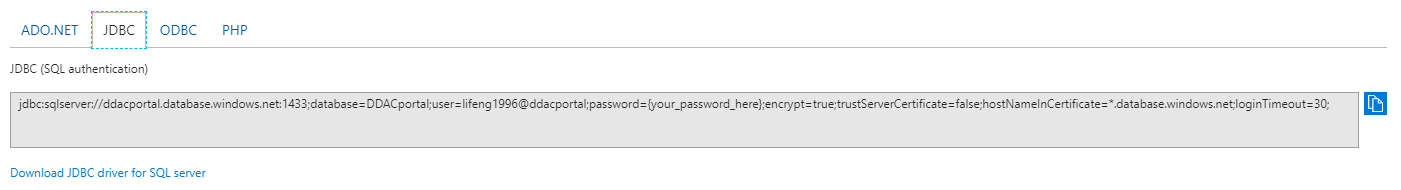
This requires information on database name, subscription, resource group, source, server, and price tier. The database name must be unique among the individual Azure databases.



Once setup the database server, to allow the web application to connect the server, firewall setting to allow the web application server connects the database server is required.

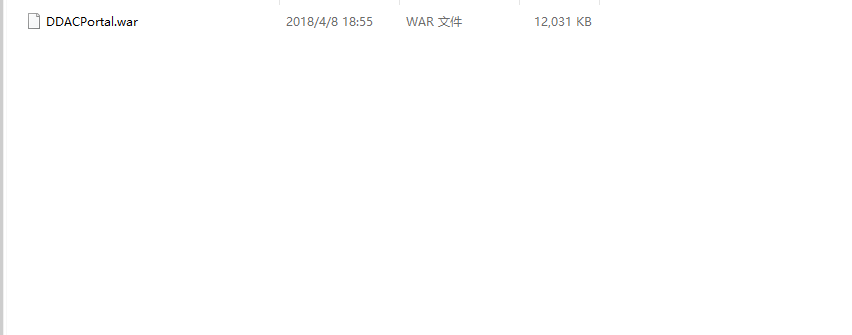


In the management page of SQL database, copy the connect string which allows web application to connect the target database correctly.

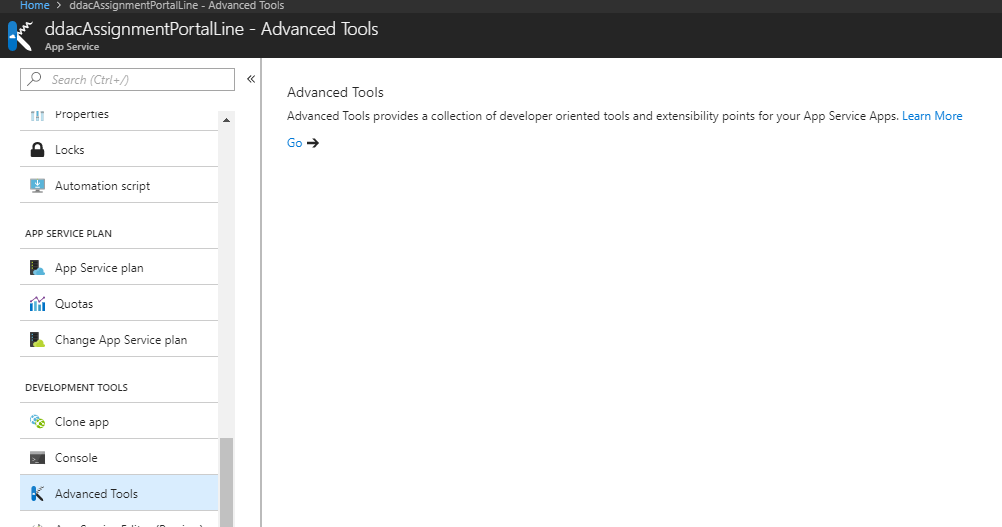


### Web Application Publishing

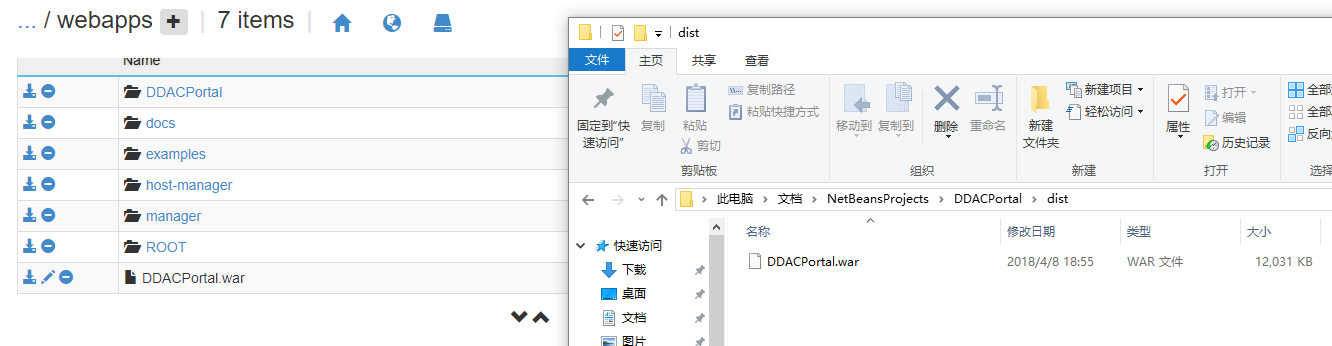
The last step is to publish the completed web application to the prepared container in the Azure. In Java Web Application, there is a WAR file will be built and stored in “dist” file. This WAR file is a compressed web application file.



To deploy the web application by WAR, it can be done easily by entering the “Kudu Service” which can be accessed from the “Deployment Tool” of the App Service



Then, go to the “Debug Console” and choose either CMD or PowerShell. Enter the command “cd home\site\wwwroot\bin\apache-tomcat-8.5.24\webapps”. Then just “Pull and Drop” the WAR file in the file.

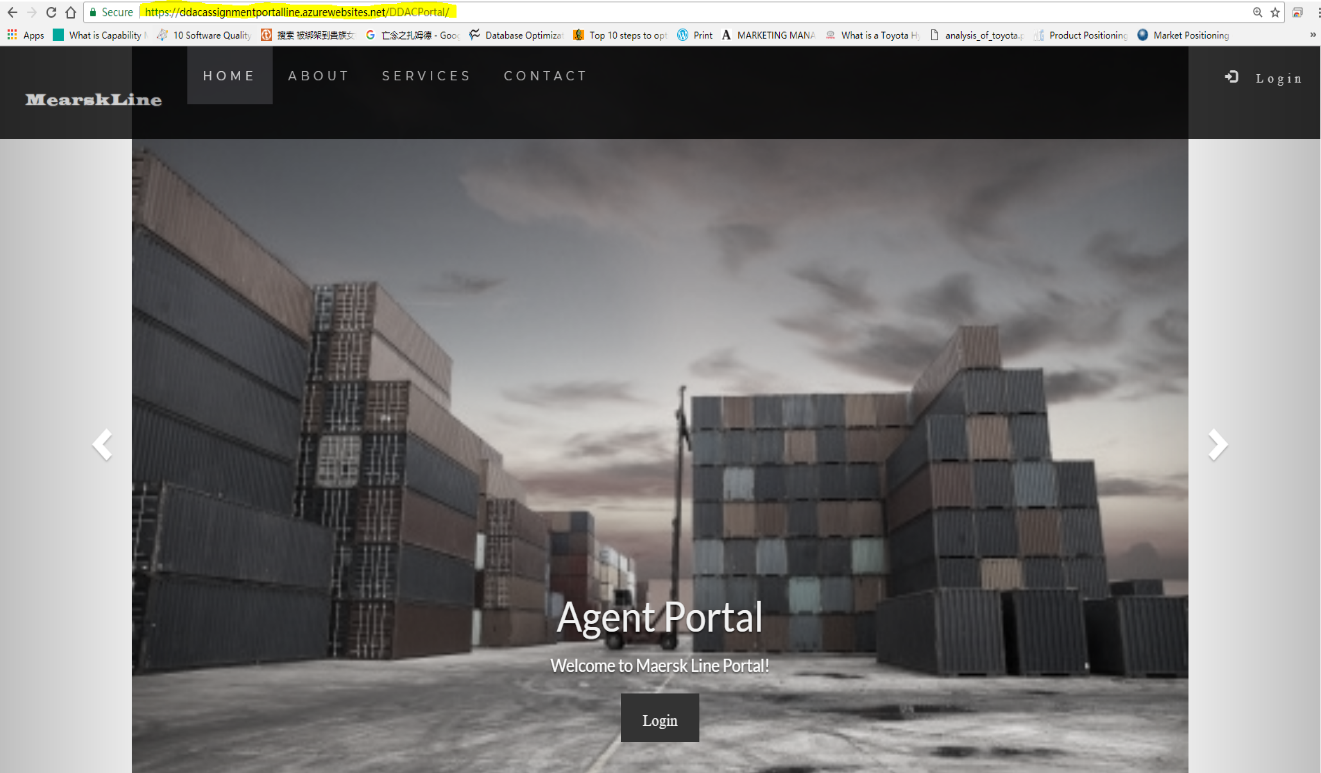


After uploaded successfully, it will be unzipped automatically and now is able to browse the web application.

GitHub Link: <https://github.com/lifeng1996/DDACportal>

Azure Link: <https://ddacassignmentportalline.azurewebsites.net/DDACPortal/>

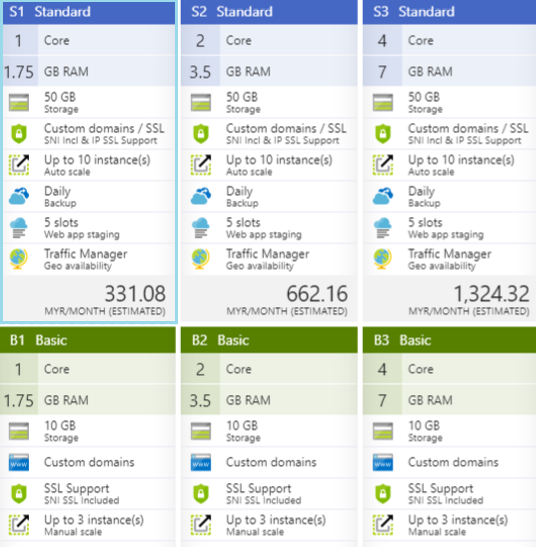
Microsoft Stream Link: <https://web.microsoftstream.com/video/216c6811-0a1e-4d29-8b28-4fe88970af74>



## Application Scaling

Application scaling allows the application to respond to changes in traffic and automatically allocate the resources needed to process the current request (Cavale, 2017). Azure Web App can monitor incoming web traffic and automatically add or delete application gears to accommodate changes in demand.

### Web App Scale



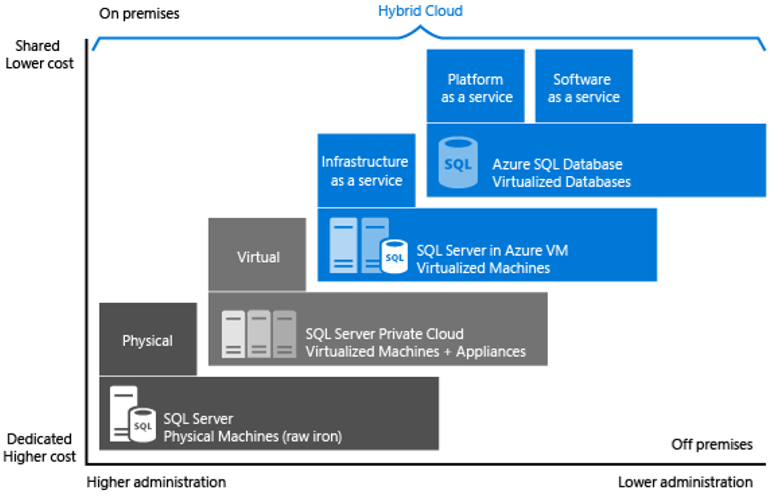
Among the pricing plans listed above, there are basic categories and standard ones that provide different offerings. The pricing plan of the web app service is decided to apply for the Standard S1 with the following reasons. First, compare with the other standard plans, there are nothing different between three of them but only the CPU and RAM. Moreover, since the system will be tested for a given period hence it is not a large scale of system in early stage. The standard plan provides a medium storage which has capability than a 50 GB and able to handle large volumes of transactions within a certain period of time. Besides, by providing custom domains and SSL in both plans, the standard plan provides special advantages and SSL support. In addition, autoscale is also a general offer, but up to 10 instances can be extended with the standard plan. The other additional feature provided only in the standard plan that backs up application settings and data daily to ensure data consistency. Furthermore, in preparation for unexpected situations, the extra five deployment slots can be used to hold the recent deployment restore points. In addition, traffic managers can handle stable connections between clients and servers by connecting to the closest server in the region.

### SQL Database Scale

In this deployment, Azure SQL Server also uses standard plans as well. Since the system will be tested for a given period and it is not requiring high computation and storage for the testing data hence a standard plan with 100GB of storage and 10 compute units are enough for the current system. Besides, the storage and compute units of the database can be scaled up in future when the system is getting mature and requiring a higher performance of database.

## Managed Database (Paas)

Due to the widespread use of online cloud and management services, services previously available only on locally provisioned servers became available by clicking buttons. The concept of a platform as a service (PaaS) is a valuable tool for markets that are trying to thoroughly cut development and deployment time.



According to Microsoft (Microsoft, 2018), Azure SQL database enables development of platform (DBaaS), SaaS (as-a-service) application as service (PaaS) database or database (DBaaS). It provides compatibility with most SQL Server features. There is nothing different from the general SQL Server available on physical premises machine, private cloud environment, third party private cloud environment, and public cloud (Microsoft, 2018). In these environments, the same server products, development tools, and sets of expertise are provided.

There are several advantages to using PaaS instead of local server. First, the speed of provisioning increases the agility of innovation (Yard, 2018). With PaaS, developers can create 10 servers using one script and delete servers after 5 minutes testing (Cavale, 2017). Also, developers can reduce operating costs. Developers can reduce costs by using features such as autoscaling (Cavale, 2017) with the Microsoft Azure service and provisioning only the resources organization needs.

An Azure managed database is suitable for new cloud-designed applications that have time constraints in development and marketing (Microsoft, 2018). Besides, teams that need built-in high availability, disaster recovery, and upgrade for the database while also do not want to manage the underlying operating system and configuration settings are recommended to utilize the Azure Managed Database (Microsoft, 2018).

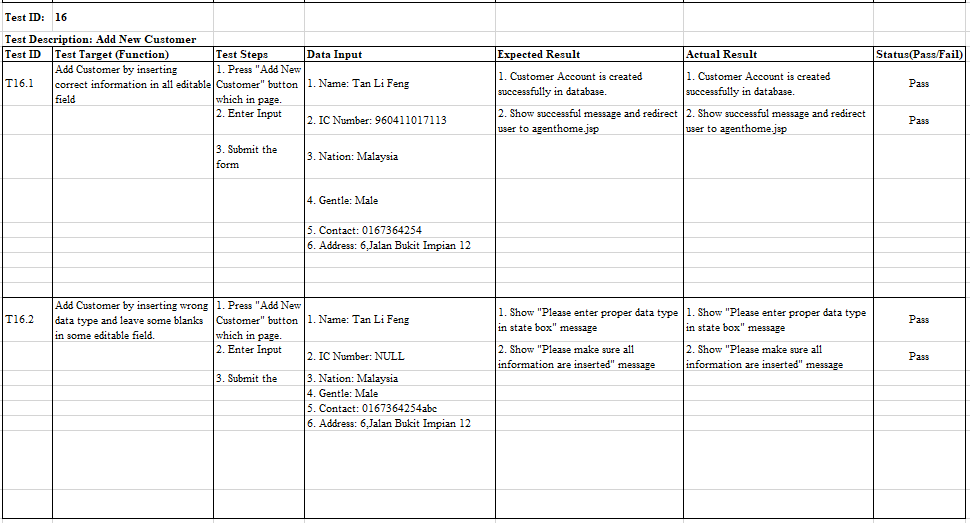
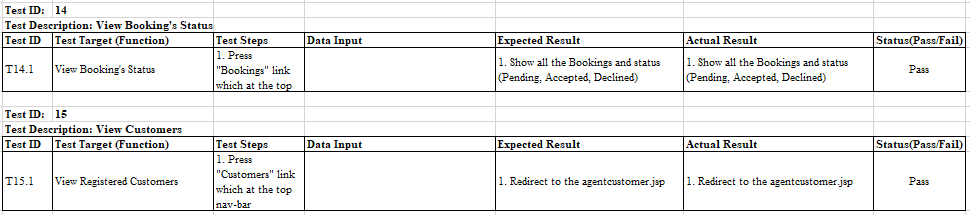
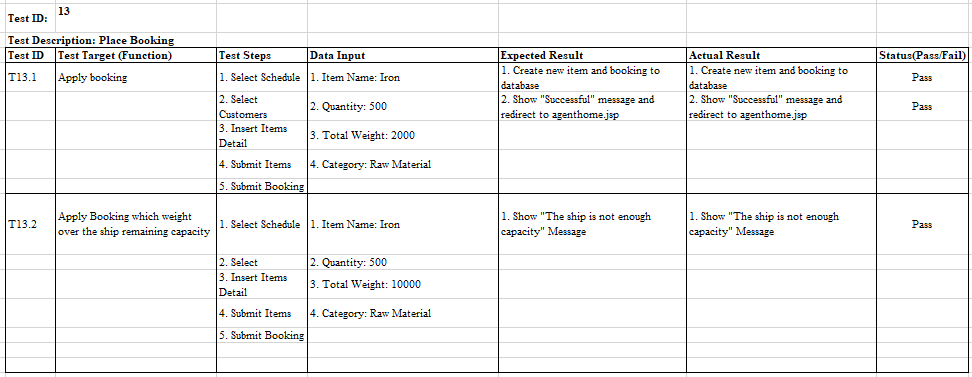
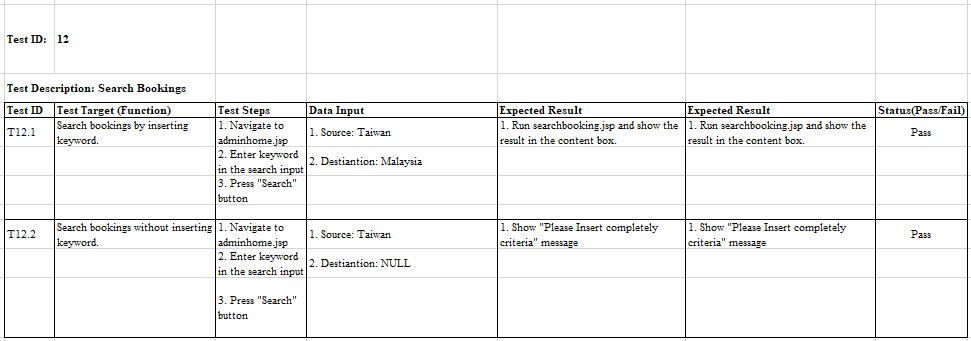
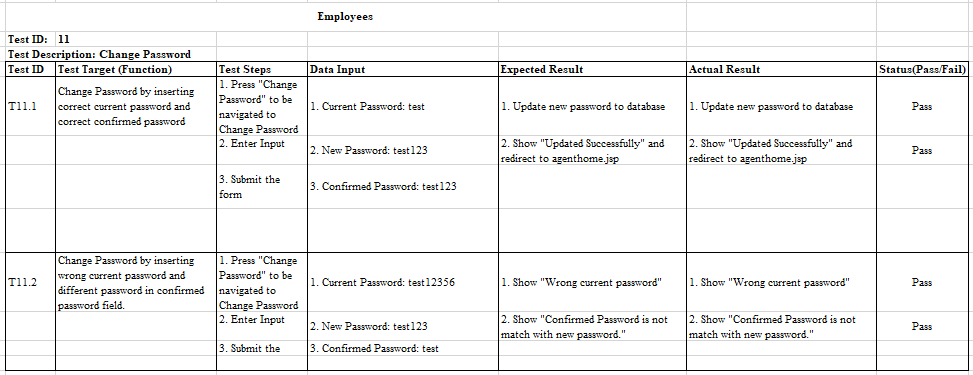
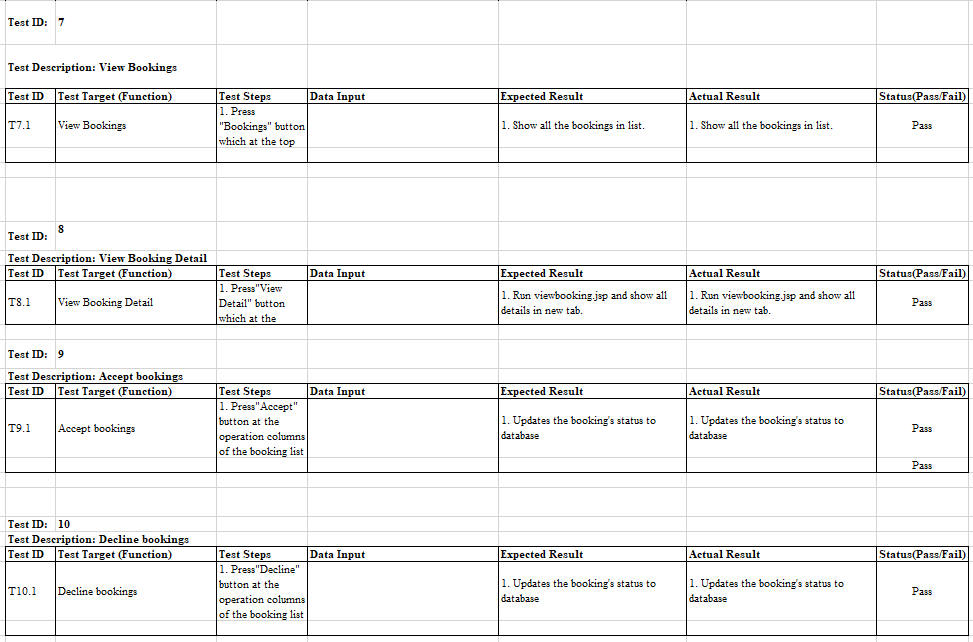
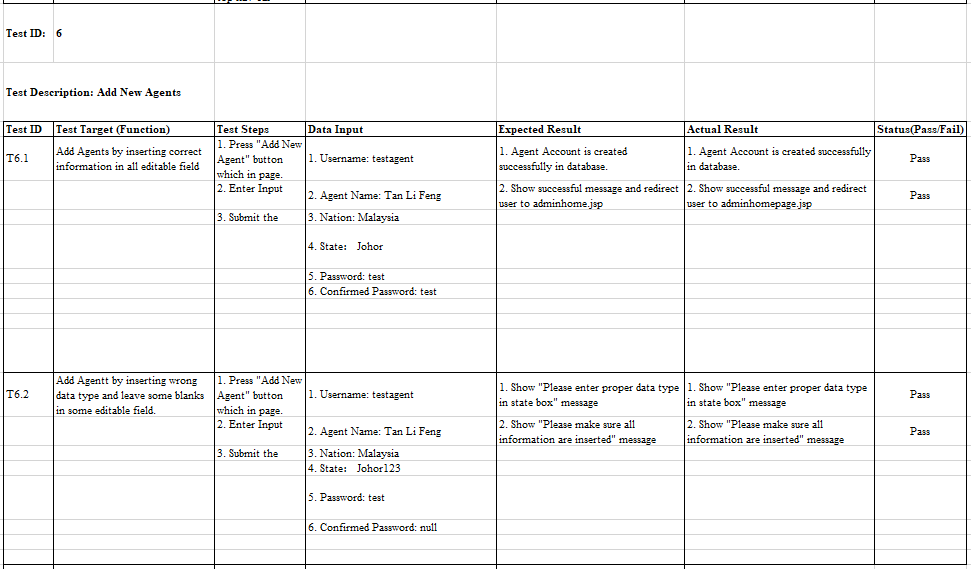
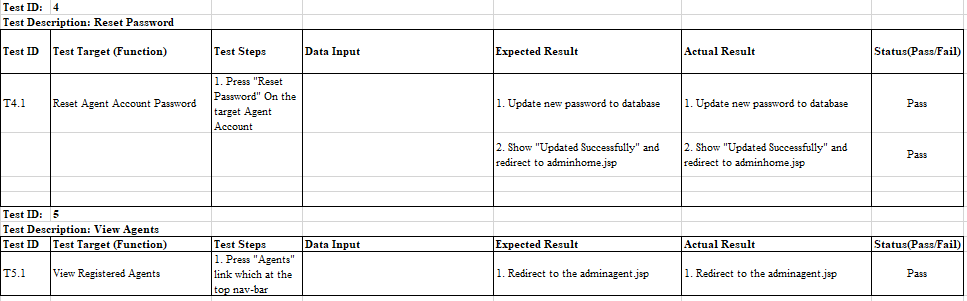
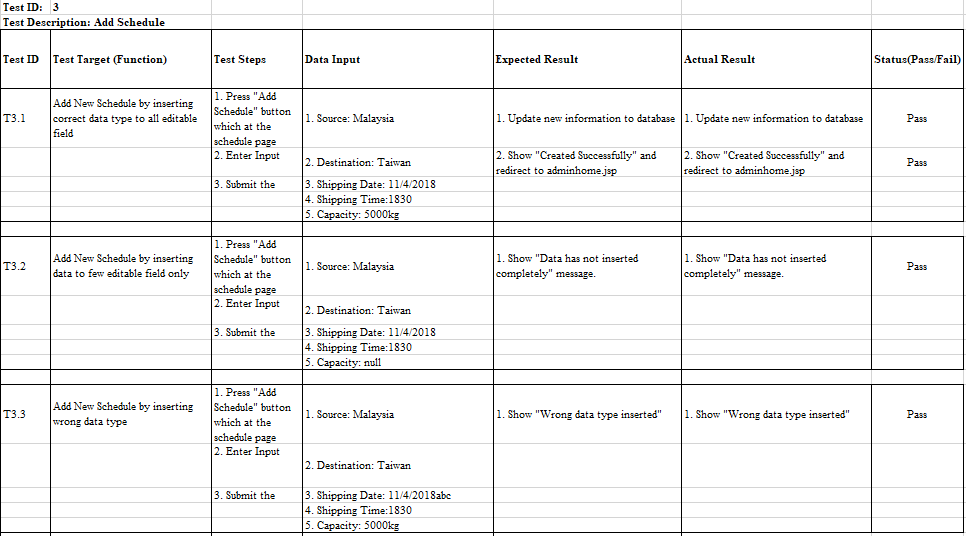
Moreover, since the Azure Managed Database provides a well-structured and highly learnability resources (Microsoft, 2018). It is mainly focus on the application layer while does not want to employ IT resources for configuration and management of the underlying infrastructure (Microsoft, 2018). By eliminating the hardware costs and administrative costs, a cost saving benefit can be achieved by applying Azure Managed Database (Microsoft, 2018).

For business, it is providing the built-in fault tolerance infrastructure capabilities, automated backups, point in time restore, geo restore and active geo replication to improve business performance (Microsoft, 2018). These characteristics enable a business to be more sustainable (Microsoft, 2018). Meanwhile, the high efficiency and convenience of the Azure Managed Database makes On-premises application can access data in Azure SQL Database wherever and whenever (Microsoft, 2018).

# Testing

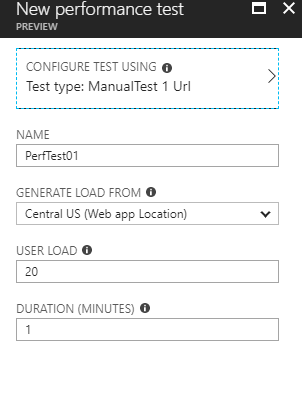
## Unit Testing

Unit Testing will be performed to test each of the functions of the system. It is carried out by entering the input and check for the validation of the output of system. By gathering the result of the testing, developer will able to figure out the shortage of the system. In unit testing, each of the functions of the system will be tested separately to see whether it shows the exactly output as the deliverables.

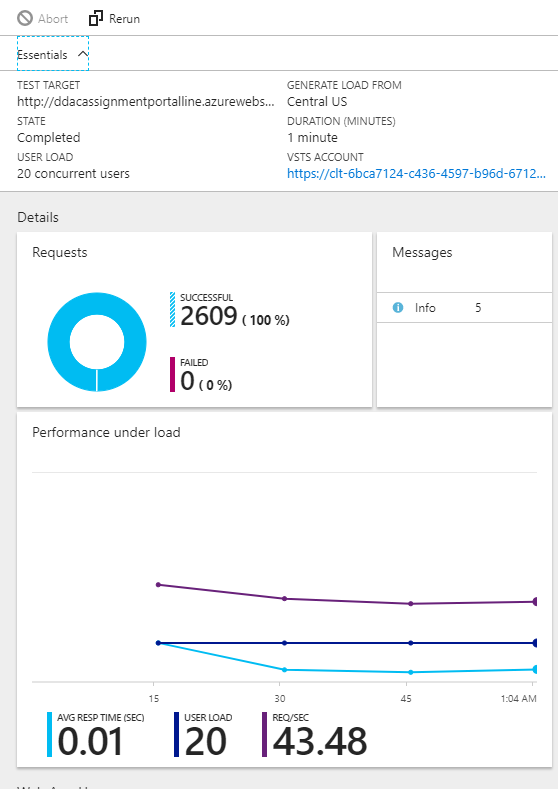


## Load Performance Testing

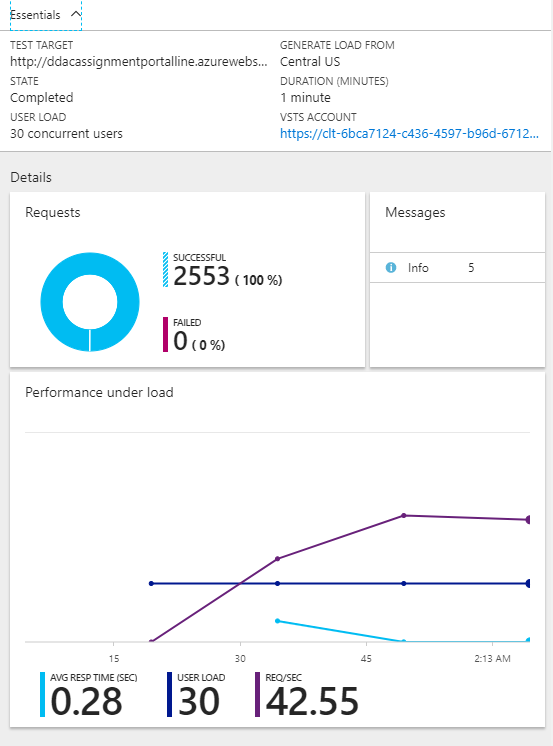
Performance is mostly about response time of the web application. This response time should be in acceptable intervals, and should not increase if transaction count increases (Basaraner, 2013). Performance testing is done with the help of the functionality provided as part of the Azure web application. The test is run on the main web application resource and the getting increase the users incrementally. Each test tests 20 to 40 user load for 1 minute. Results collected include response time and failed request.



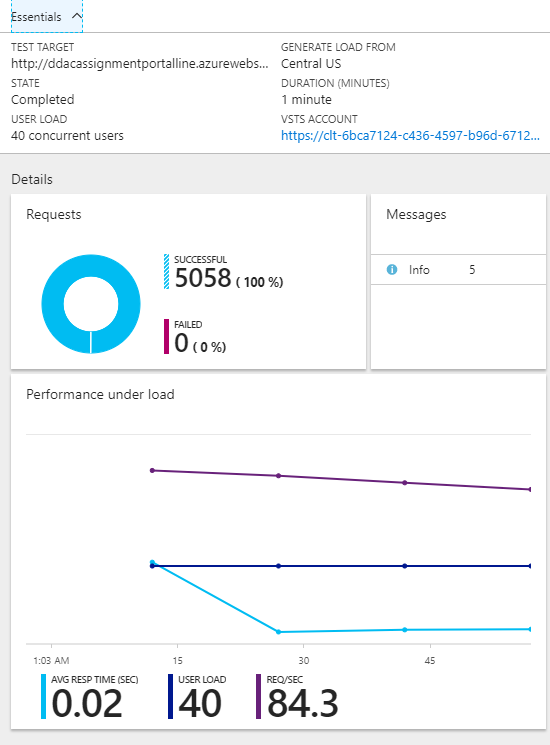
### Result of 20 User Load



### Result of 30 User Load



### Result of 40 User Load



### Analysis

This is based on the above load test chart provided by the developer. The Maersk Line Container Management System is a high performance for handling a large number of users in 60 seconds.

In this load test, the developer provides two scenarios. Two scenarios explaining the diagram which a scenario to handle 20 users in 1 minute, another scenario to handle 40 users in 1 minute.

In the first scenario, in the web application that handles 20 users in 1 minute, the load test shows that the performance of the web application is performed well without suspense, and the web application takes average 0.1 second for the response time per page display to each user.

In the second scenario, the web application was handling 40 users in 1 minute, the load test indicates that the performance of the web application is good and the web application has a response time of 0.2 seconds per page display by the user Indicates that.

After comparing these two scenarios, developers are analyzing that the Maersk Line Container Management System is able to deliver and maintain high performance for a medium scale of concurrent users browsing to the application within 60 seconds.

# Conclusion

In Conclusion, I am appreciated that the proposed system is carried out successfully and which is able to achieve all the deliverables that defined in before but there are still having shortage within the system. In the last few stage of the development, the trial plan of Azure Account has expired which cause the load performance testing can’t be conducted by testing heavier load of users. I am also appreciated to be given an opportunity to finish this project with a proper cloud design pattern and cloud application development, especially this assignment is being brought into case which similar as a real-life project. I had learnt a lot from this project such as proper programming techniques and project management skills which are useful in my future employment. For the challenging facing during this assignment might be doing the brainstorming about the design of GUI and research for the related source code. After finished this project, I am really get learnt about how a program is designed in patterns properly to lead the system become high cohesion and low coupled.

# References

Basaraner, C., 2013. *10 Software Quality Factors That Should Always Be Remembered.* [Online]   
Available at: https://dzone.com/articles/10-groups-software-quality  
[Accessed 11 4 2018].

Bootstrap, 2018. *Bootstrap.* [Online]   
Available at: https://getbootstrap.com/  
[Accessed 11 4 2018].

Cavale, A. B. R. J. &. K. J., 2017. *Scale instance count manually or automatically..* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/insightshow-to-scale   
[Accessed 11 4 2018].

Microsoft, 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 11 4 2018].

Mozilla, 2018. *JavaScript.* [Online]   
Available at: https://developer.mozilla.org/bm/docs/Web/JavaScript  
[Accessed 11 4 2018].

Yard, E., 2018. *Top 10 Advantages of Platform as a Service.* [Online]   
Available at: http://www.engineyard.com/whitepapers/top-10-advantages-of-platform-as-aservic  
[Accessed 11 4 2018].

# Appendix

## Project Links

**More information for the system development and demonstration:**

GitHub Link: <https://github.com/lifeng1996/DDACportal>

Azure Link: <https://ddacassignmentportalline.azurewebsites.net/DDACPortal/>

Microsoft Stream Link: <https://web.microsoftstream.com/video/216c6811-0a1e-4d29-8b28-4fe88970af74>

## Test Account for demostration

Admin Account

Username: test

Password: test

Agent Account

Username: testagent

Password: test