

**DESIGN AND DEVELOP APPLICATION ON THE CLOUD**

**INDIVIDUAL ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| Module Code | : | CT071-3-5-3-DDAC |
| Student ID | : | **TP039020** |
| Student Name | : | **Zain Fida** |
| Intake Code | : | UC3F1706SE |
| Project Title | : | **Container Booking System** – (Web Application) |
| Lecturer Name | : | **Dr. Kalai Anand A/L Ratnam** |
| Hand in Date | : | 30th March 2018 |

Table of Contents

[1.0 Introduction 4](#_Toc509419711)

[1.1 Background 4](#_Toc509419712)

[1.2 Objective 4](#_Toc509419713)

[1.3 Scope 4](#_Toc509419714)

[1.4 Major functions 5](#_Toc509419715)

[2.0 Project Plan 6](#_Toc509419716)

[2.1 Tasks to be completed 6](#_Toc509419717)

[2.2 Gantt Chart for CBS 7](#_Toc509419718)

[3.0 Design 8](#_Toc509419719)

[3.1 Use case Diagram 8](#_Toc509419720)

[3.2 User Interface Story Board 9](#_Toc509419721)

[Main Screen (Welcome screen) 9](#_Toc509419722)

[Main Screen (Our Mission) 9](#_Toc509419723)

[Main Screen (Our Team) 10](#_Toc509419724)

[Main Screen (Feedback) 10](#_Toc509419725)

[Sign in Page 11](#_Toc509419726)

[View Bookings 11](#_Toc509419727)

[Register Customer or Company 12](#_Toc509419728)

[Book Vessel 1 12](#_Toc509419729)

[Book Vessel 2 13](#_Toc509419730)

[Register Agents (Admin) 13](#_Toc509419731)

[View Route (Admin) 14](#_Toc509419732)

[View or Edit System Users (Admin) 14](#_Toc509419733)

[3.3 Entity Relationship diagram 15](#_Toc509419734)

[3.4 Sequence Diagram 16](#_Toc509419735)

[User: Login/ Register Agent 16](#_Toc509419736)

[Agent: View Routes 16](#_Toc509419737)

[Agent: Book Vessel 17](#_Toc509419738)

[Administrator/Agent: View Booked Vessel 17](#_Toc509419739)

[3.5 Architectural Diagram 18](#_Toc509419740)

[4.0 Implementation 18](#_Toc509419741)

[4.1 Publishing CBS to Azure 18](#_Toc509419742)

[4.2 Application Scaling 21](#_Toc509419743)

[5.0 Test Plan 24](#_Toc509419744)

[5.1 Functional 24](#_Toc509419745)

[View Routes 24](#_Toc509419746)

[Book Vessel 25](#_Toc509419747)

[Book Bay 26](#_Toc509419748)

[View all Bookings 27](#_Toc509419749)

[5.2 Performance 28](#_Toc509419750)

[6.0 Conclusion 30](#_Toc509419751)

[7.0 References 31](#_Toc509419752)

# Introduction

## 1.1 Background

Maersk Line, the global container division and the largest operating unit of the A.P. Moller, 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.

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

## 1.2 Objective

The main objective of the current project is to design and develop a web application which accessible in different regions of the world. Besides, the web application should be able to operate and manage container booking for the Maersk Line customers and provide access to its features from different parts of the world. In addition, the system must be hosted on the Azure Cloud platform, which will help to modify and improve the container booking management and enhance the website in terms of performance, reliability and scalability.

## 1.3 Scope

The main goal of current project is to develop and deploy the web solution for the Maersk Line Company. The solution is a web application which allows to manage customer's container bookings in the system. The current solution should be hosted on the Azure Cloud platform, means the system will be remotely in the cloud so it will be accessible anywhere in the world.

## 1.4 Major functions

The proposed online container booking system (CBS) will have the following set of core or major functions:

|  |  |
| --- | --- |
| Category of functions | Functions |
| Vessel Search: | Current category of the functions will allow the admin and agent to search the available vessel from the database of the system. Moreover, this function allows choose the origin place, destination and date of applicable vessel. |
| Vessel Booking: | Current category allows the agent to book the appropriate vessel. |
| Bay Booking | Bay Booking is the set of functions which allows customer to choose the bay from the system and allows the agent to reserve the number of bays from a selected vessel by keying in the number of bays and the select the registered customer or company and click the book button to confirm booking. |
| Booking View: | Booking View is the set of functions which allows the agent and admin of the system to view the reserved vessel by clicking on the view booking button on their index menu page. |

Figure : Major Functions of CBS

# 2.0 Project Plan

## 2.1 Tasks to be completed



Figure : Tasks To be Completed for CBS

## 2.2 Gantt Chart for CBS

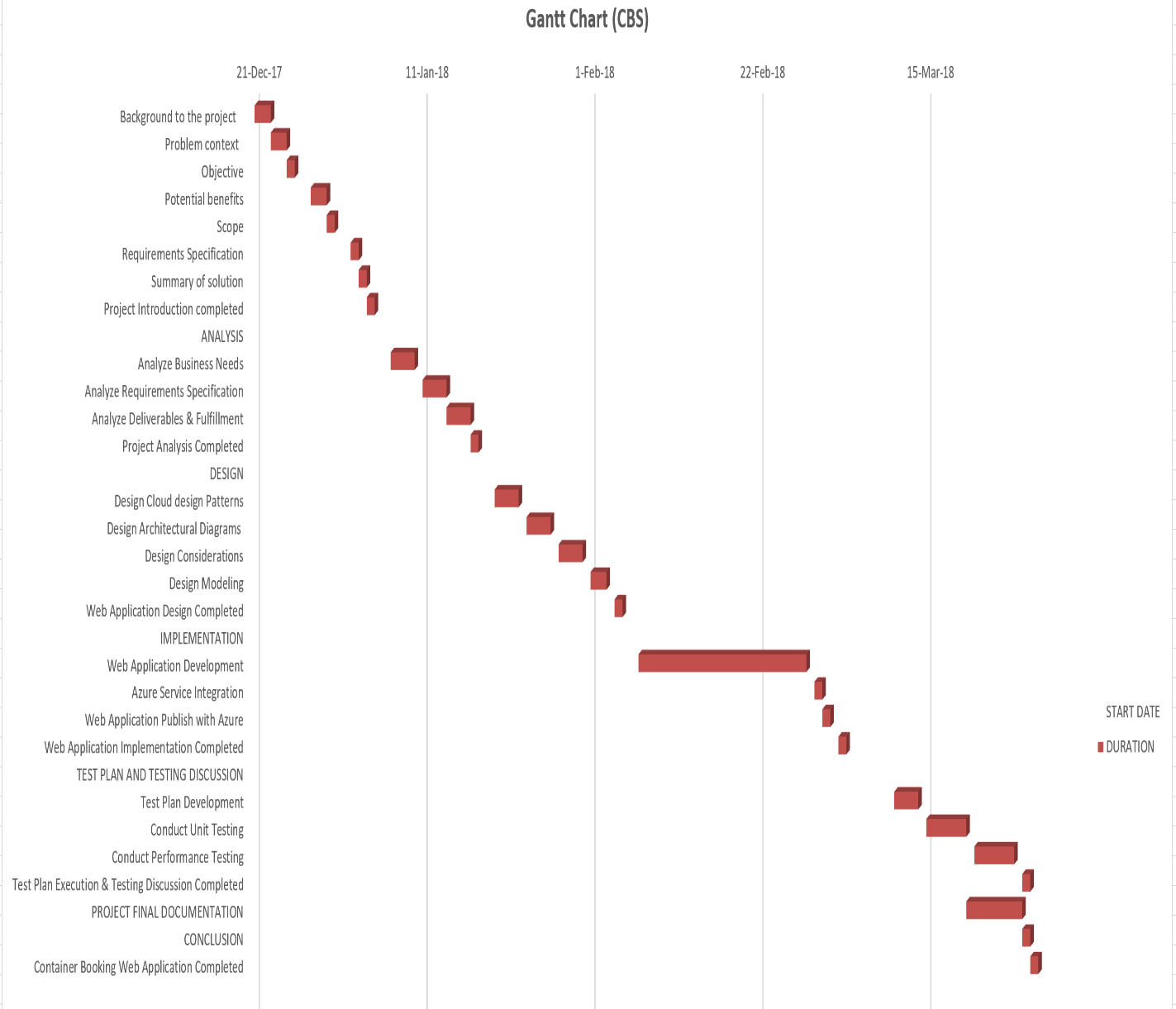


Figure : Gantt Chart for CBS

# 3.0 Design

## 3.1 Use case Diagram

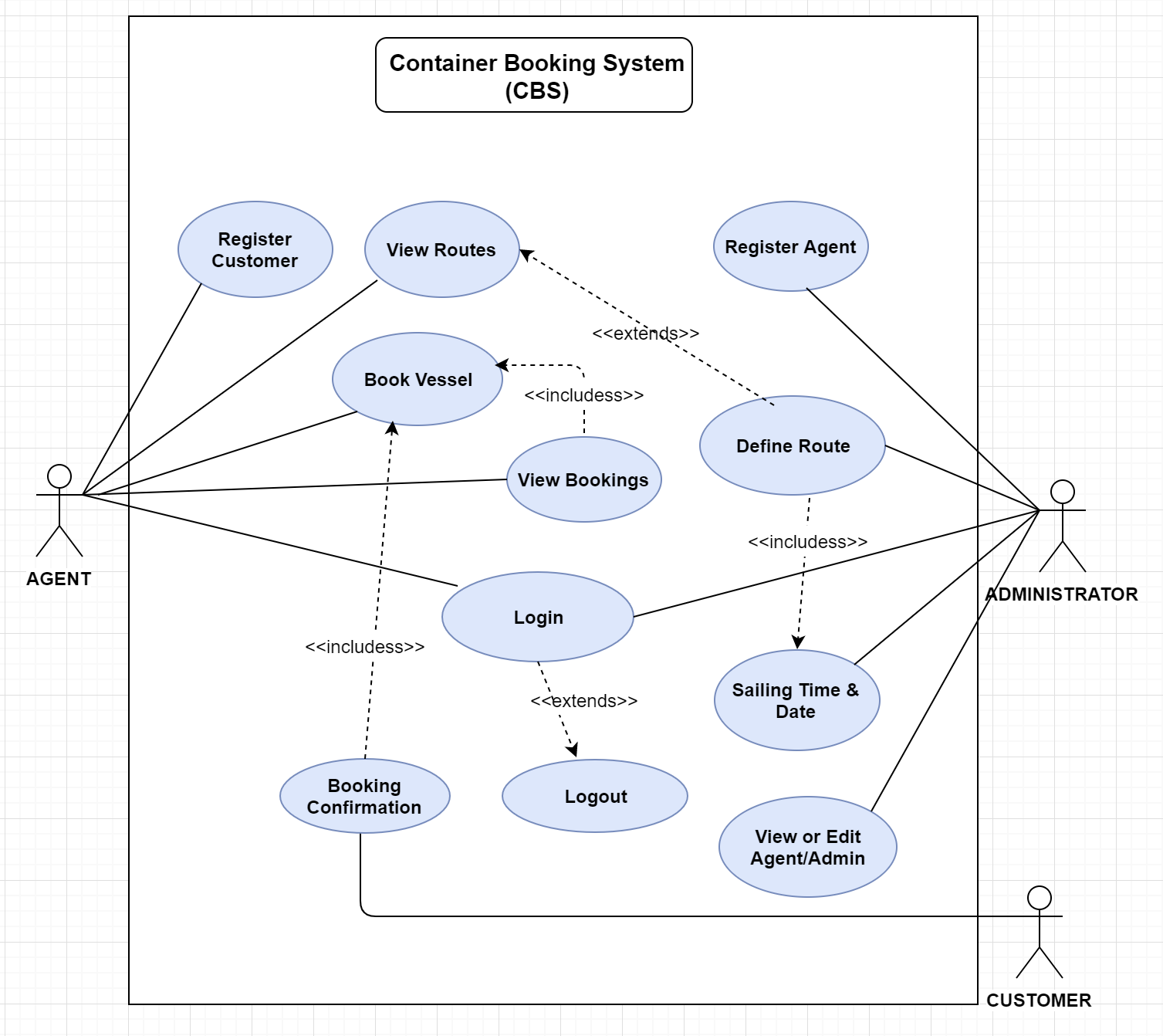


Figure : Use Case Diagram

## 3.2 User Interface Story Board

### Main Screen (Welcome screen)

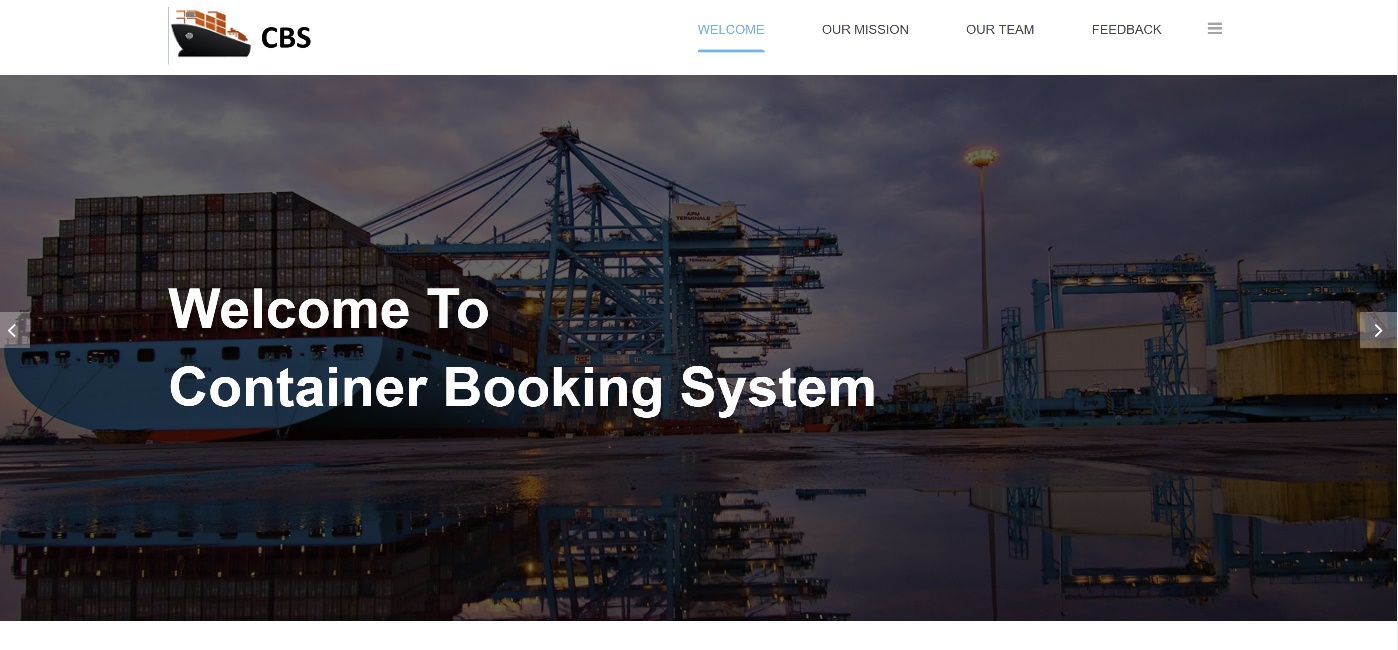


Figure : Main Screen (CBS)- Welcome Message

### Main Screen (Our Mission)



Figure : Main Screen (CBS) – Our Mission

### Main Screen (Our Team)

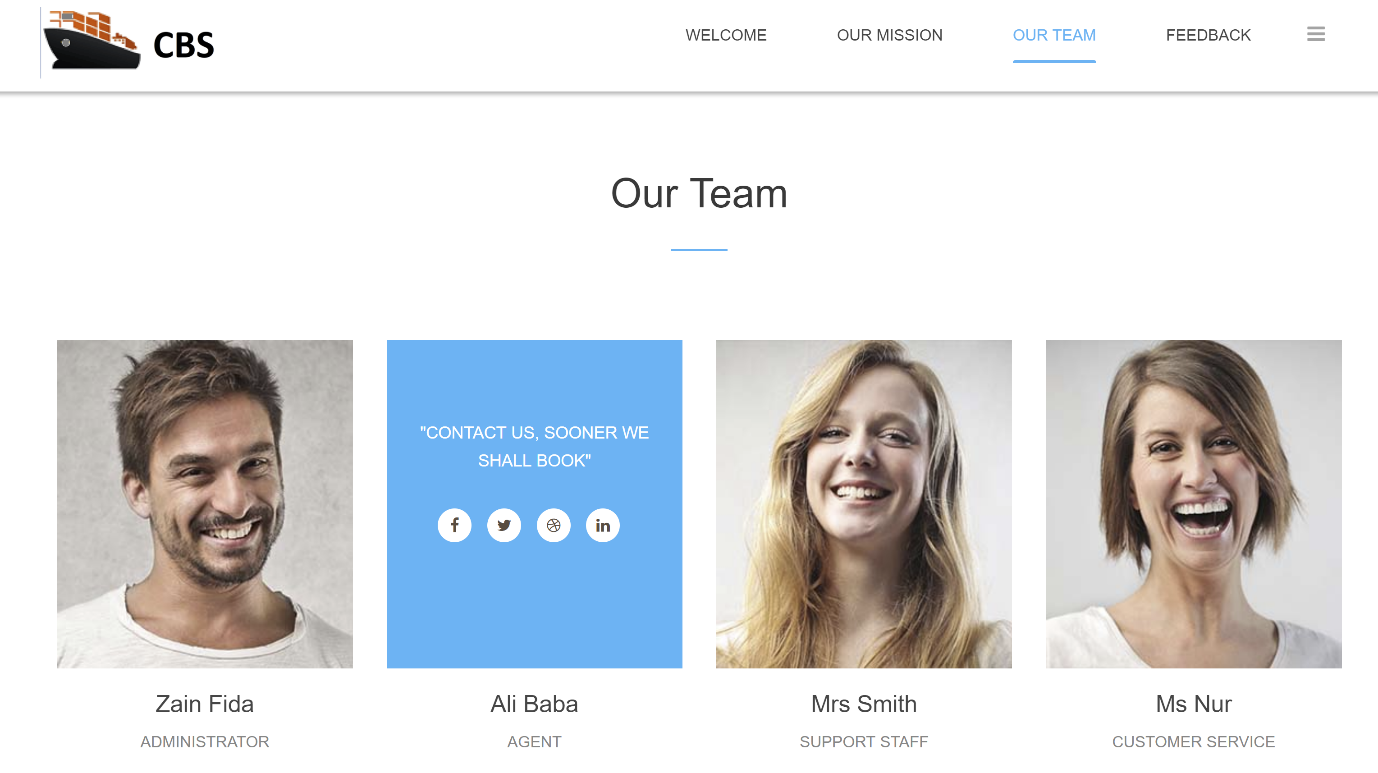


Figure : Main Screen (CBS) - Our Team

### Main Screen (Feedback)

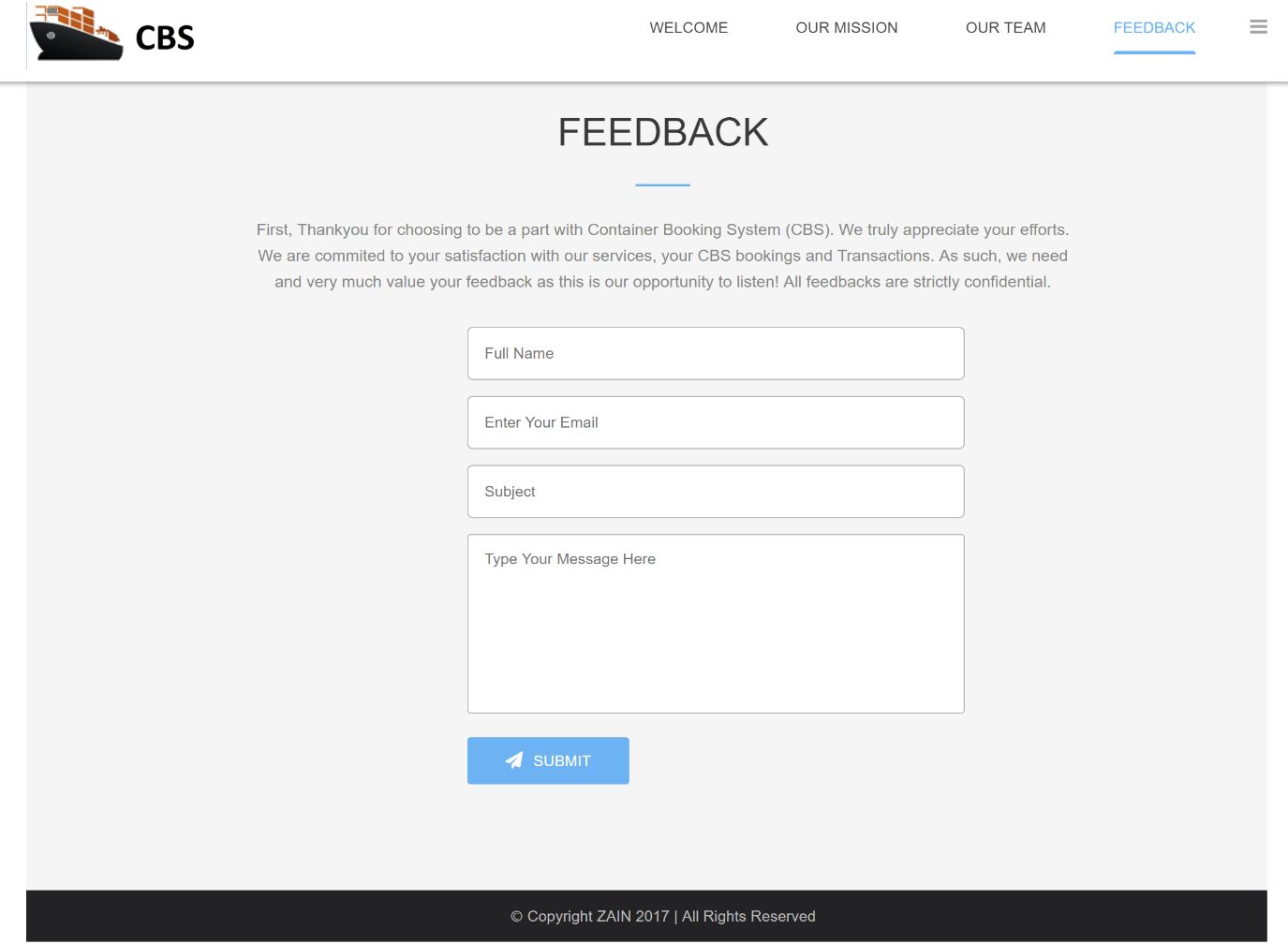


Figure : Main Screen (CBS) - Feedback

### Sign in Page

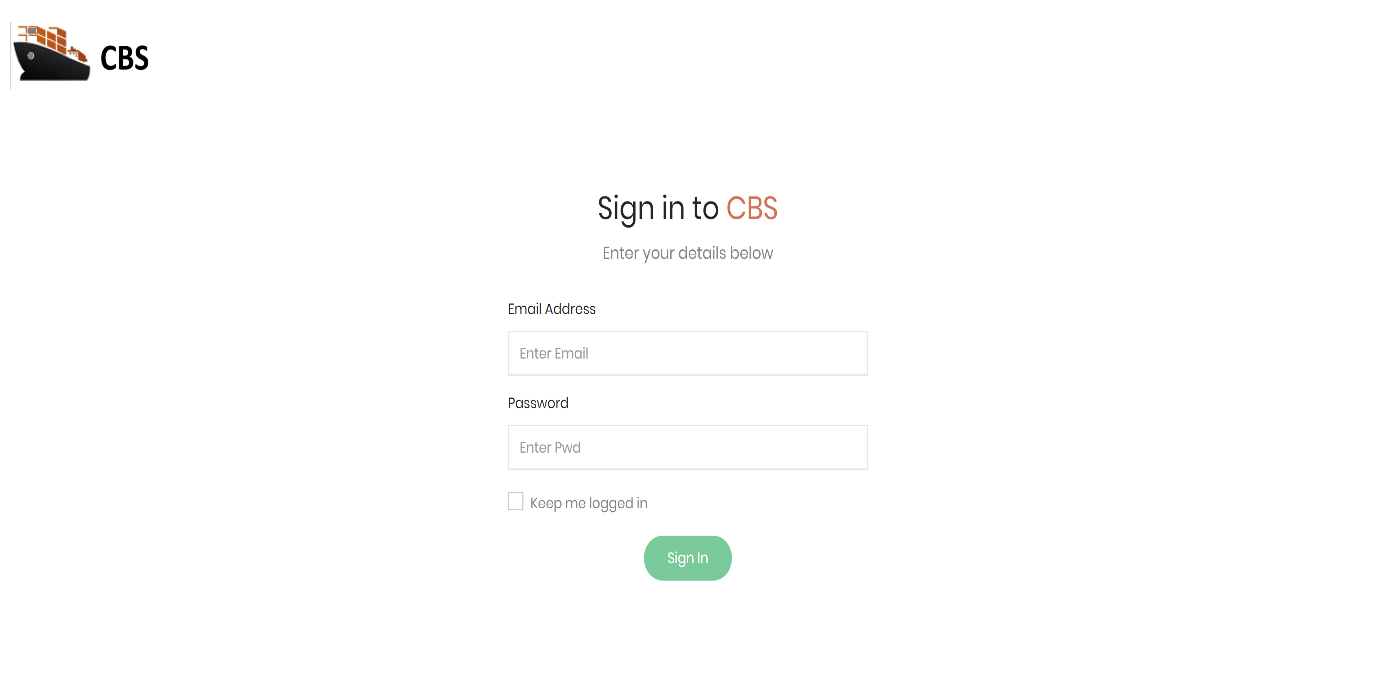


Figure : Sign In Page (CBS)

### View Bookings

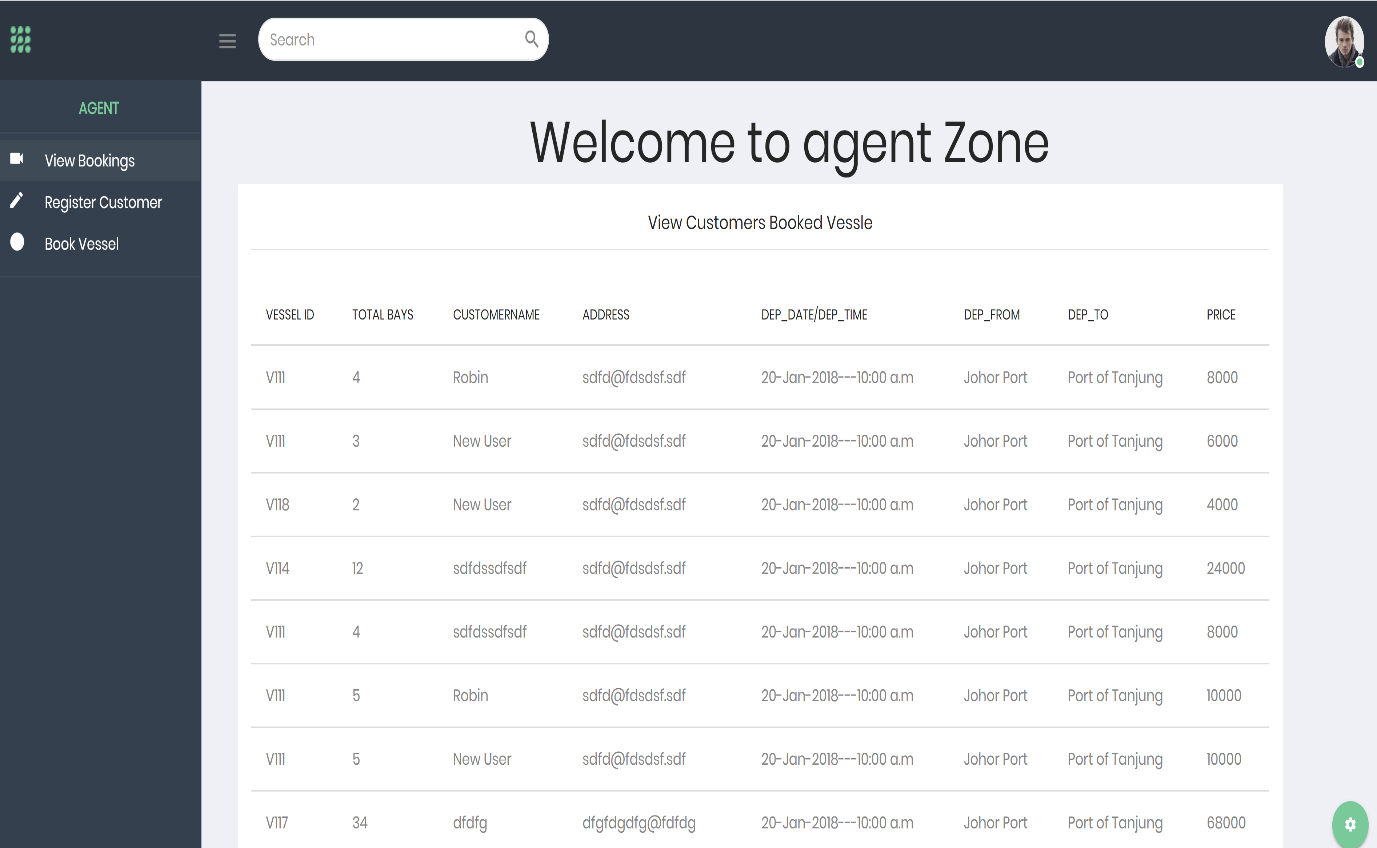


Figure : View Bookings Page (CBS)

### Register Customer or Company

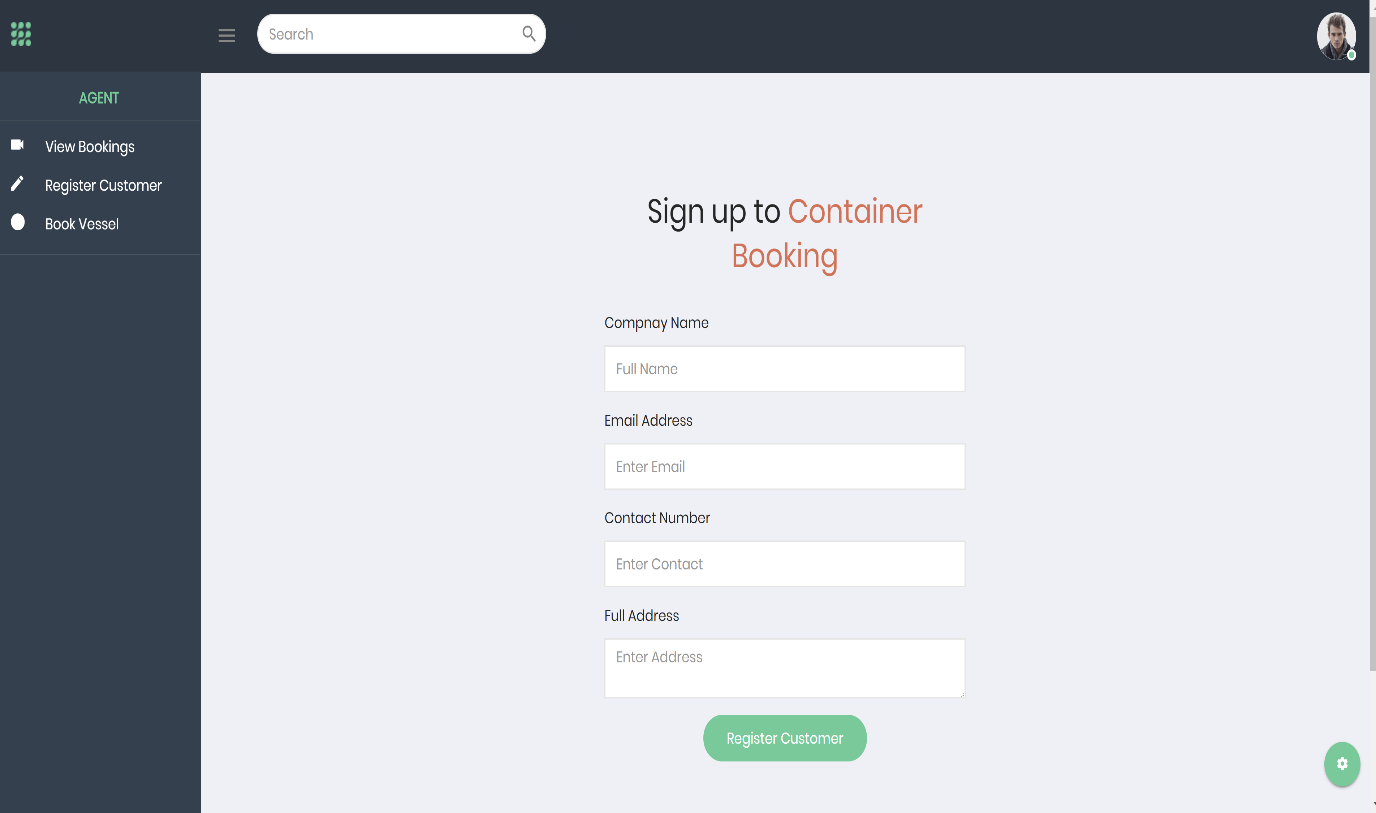


Figure : Register Customer page (CBS)

### Book Vessel 1

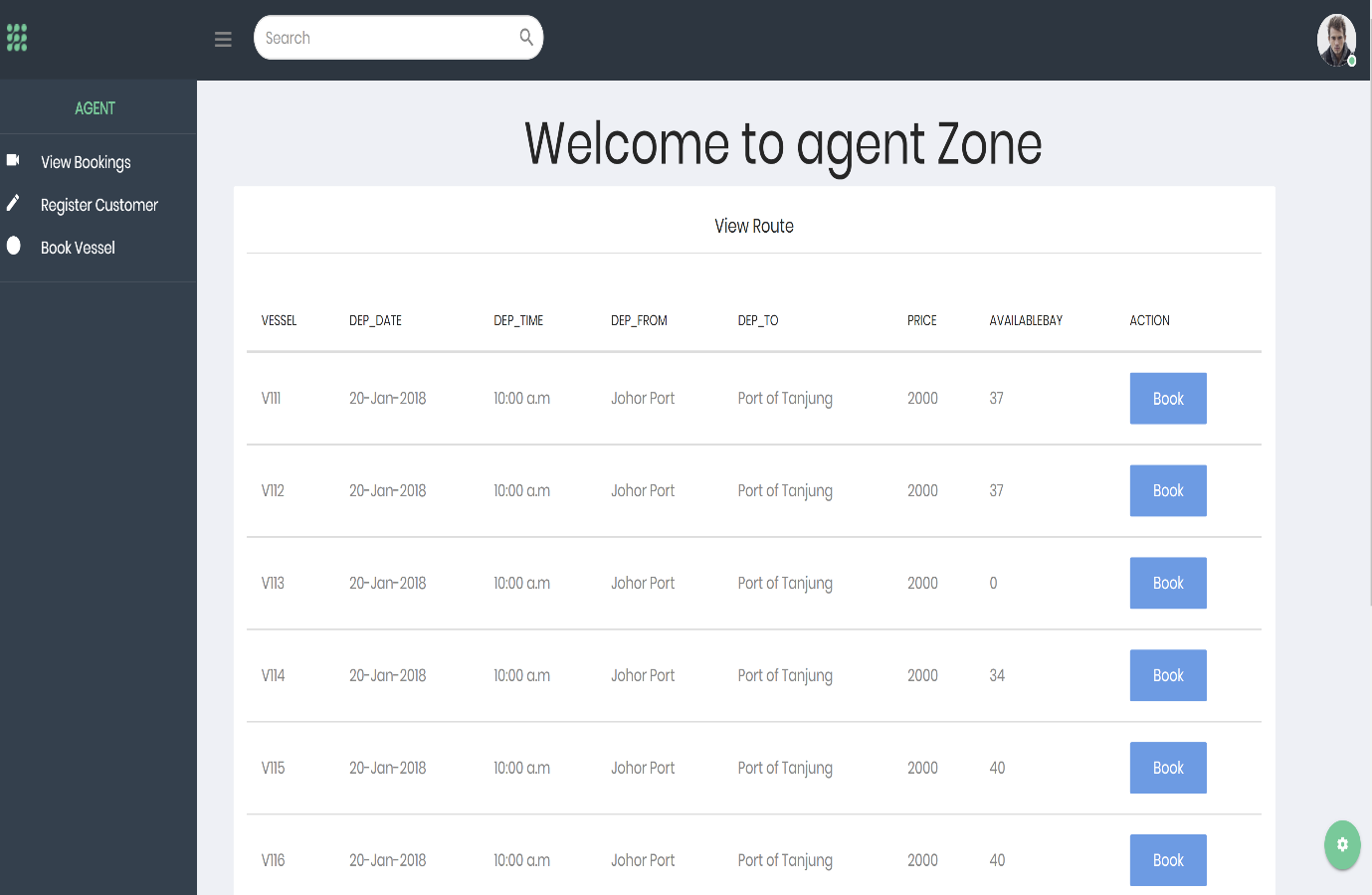


Figure : Book Vessel page (CBS)

### Book Vessel 2

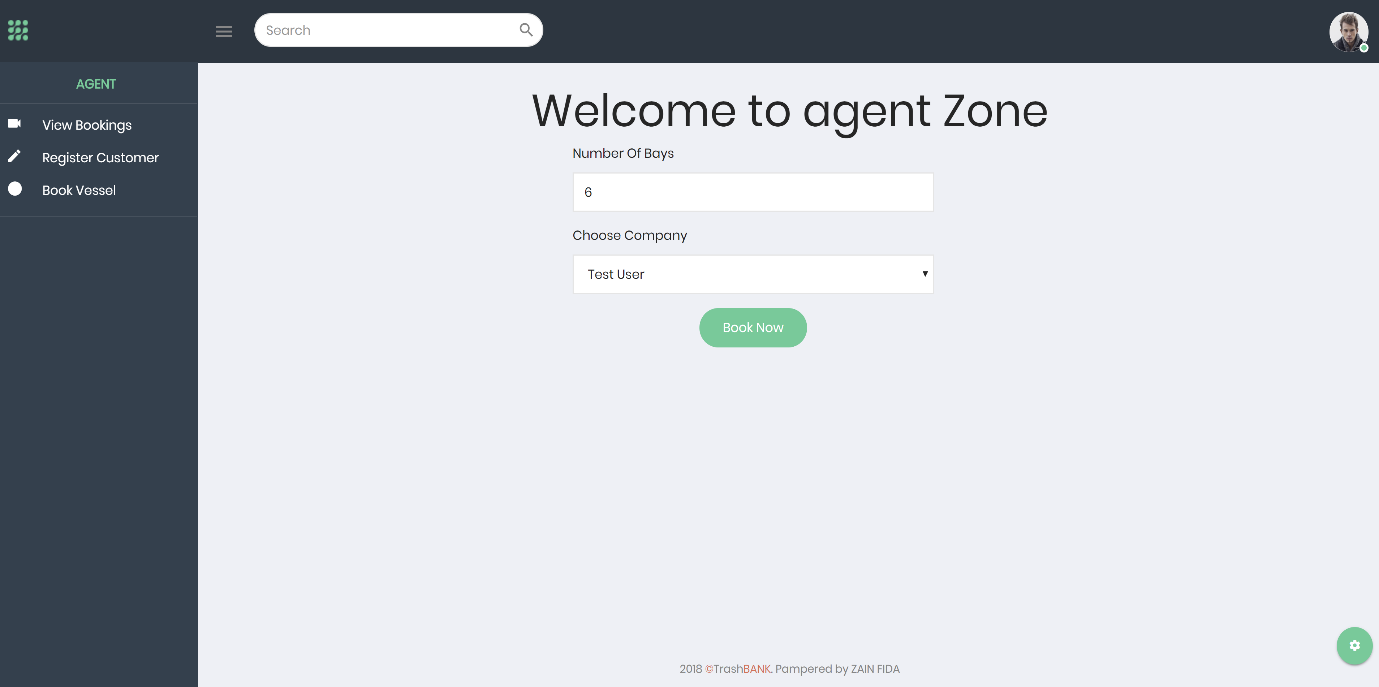


Figure : Book Vessel 2 (CBS)

After clicking on the book button for a desired vessel, the above page shall be displayed, where agent will input the number of bays and select the registered company or customer from the drop-down list before clicking the book button. Upon clicking the book button, the booking shall be confirmed and stored in the database and a confirmation message will be shown to the agent and forwarded to the company or the particular person.

### Register Agents (Admin)

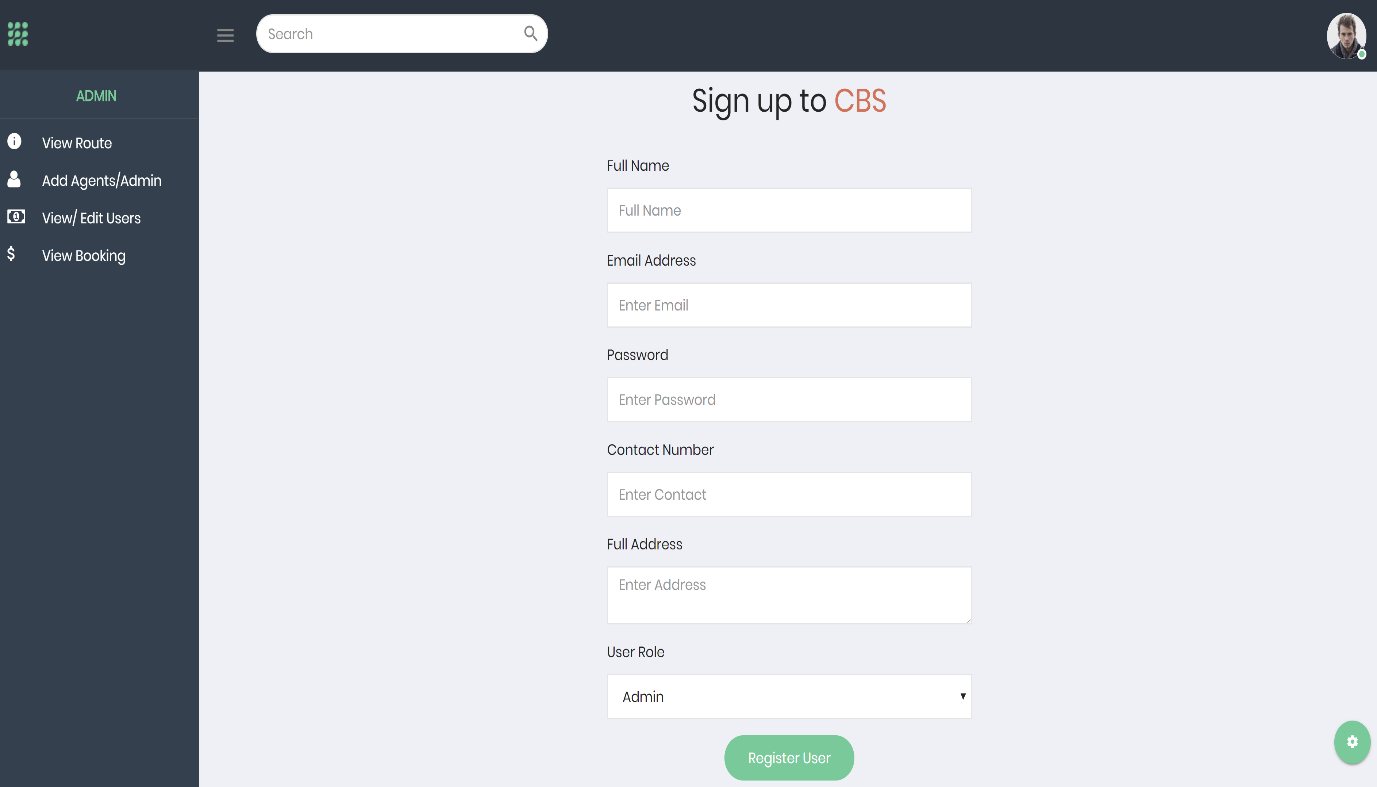


Figure : Register Agents page (CBS)

### View Route (Admin)

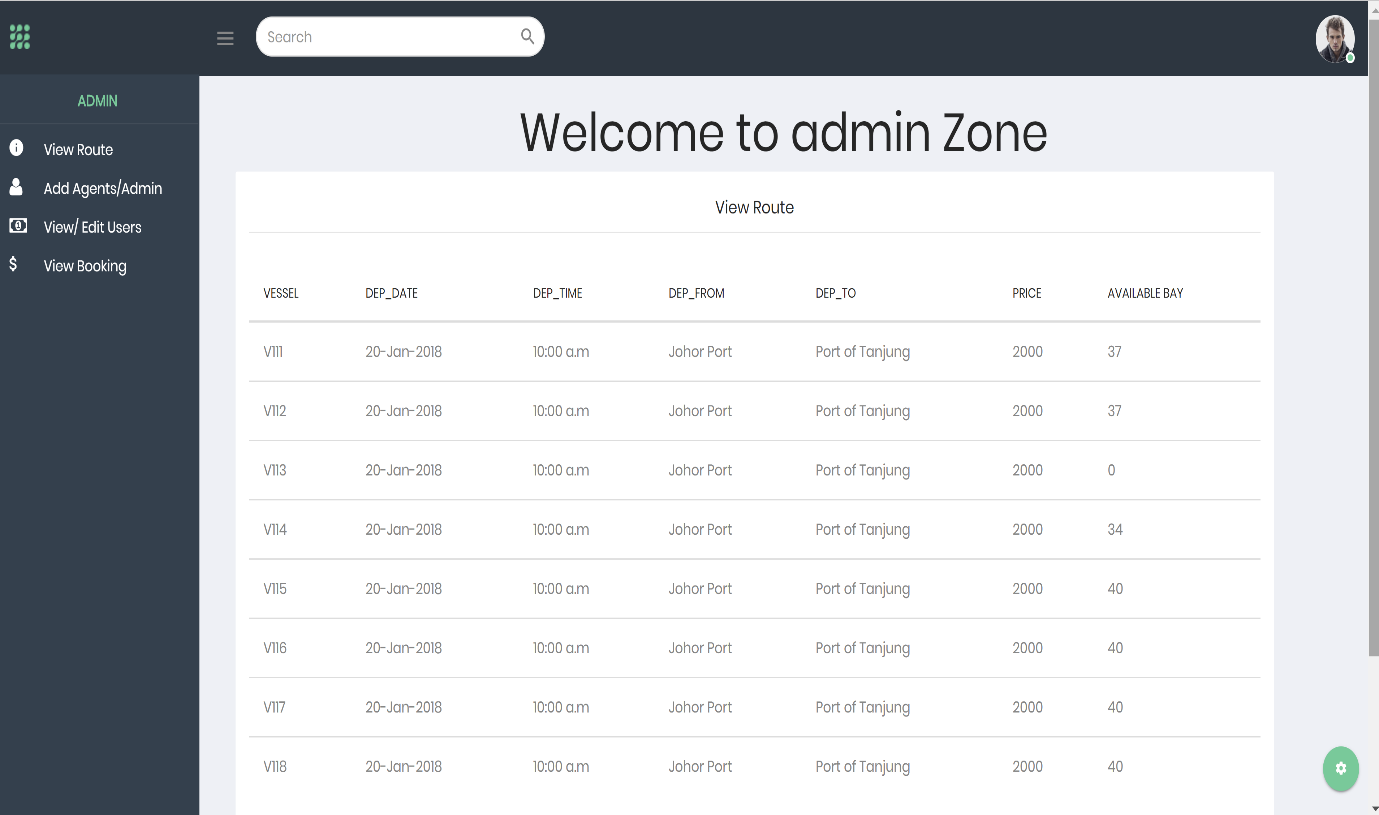


Figure : View Route page (CBS)

### View or Edit System Users (Admin)

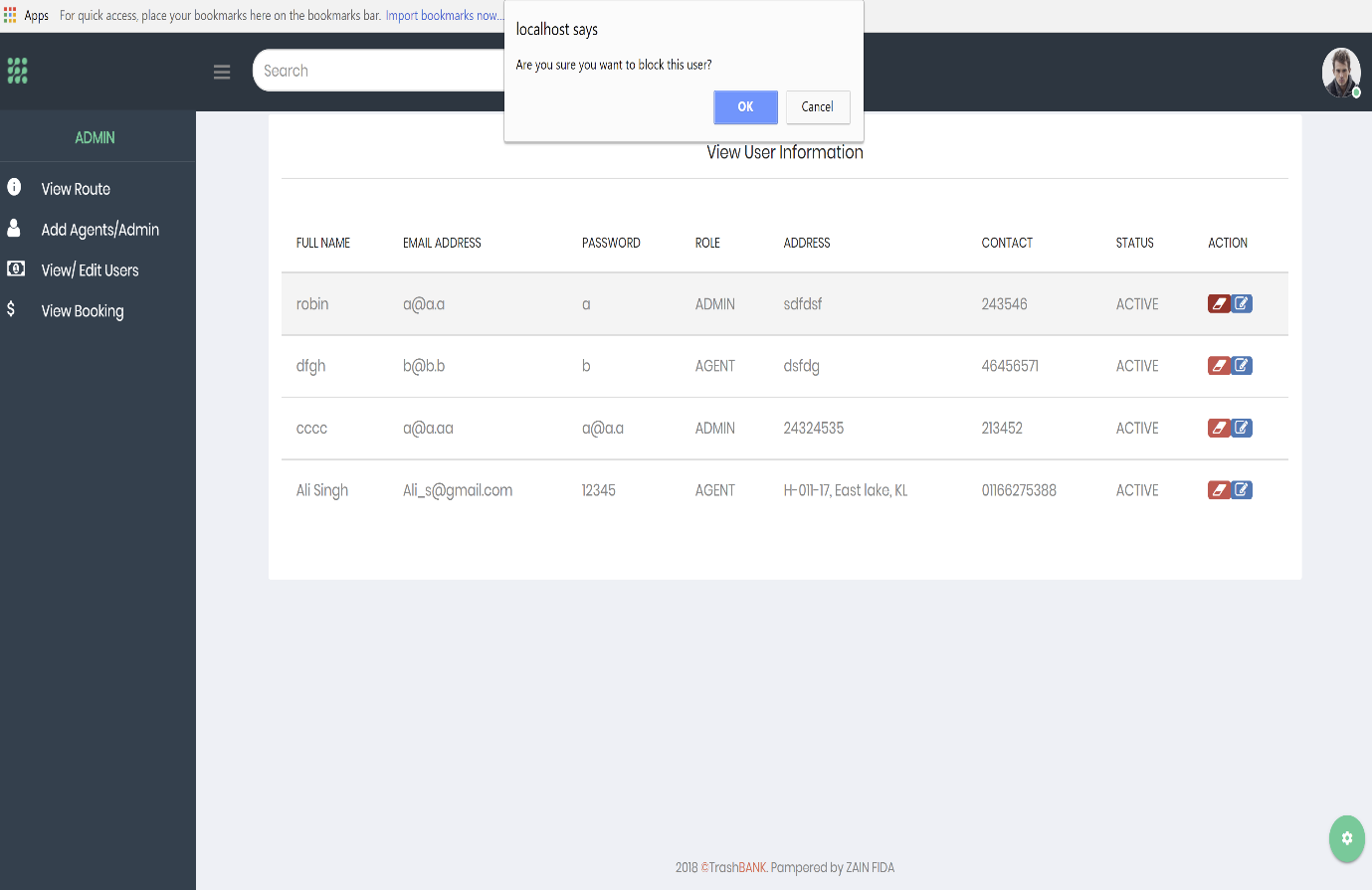


Figure : View or Edit System users (CBS)

## 3.3 Entity Relationship diagram

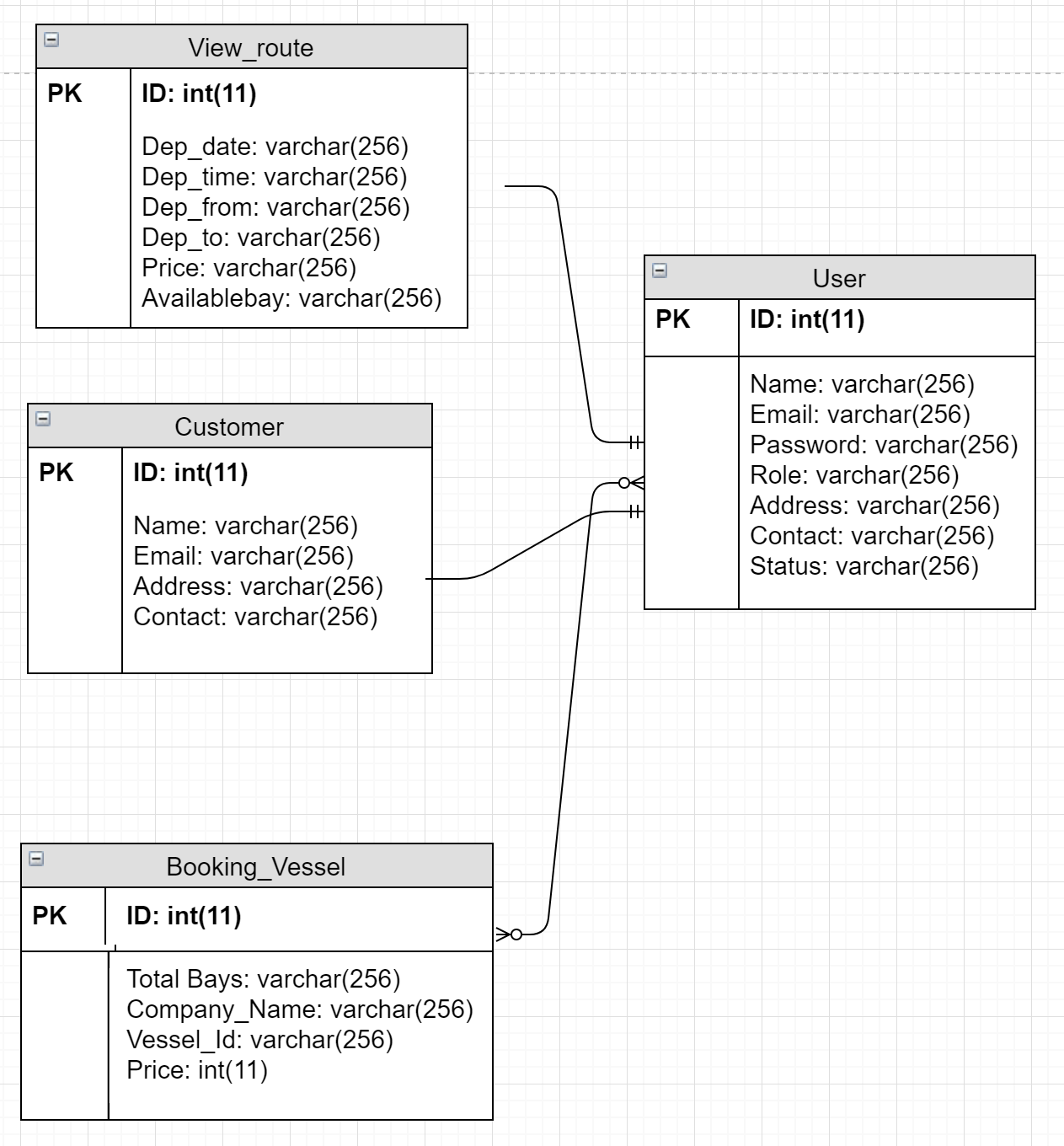


Figure : ERD Diagram (database design)

The database design above shows the entity relationship diagram of Container Booking System (CBS). There are only 4 tables which are user, booking vessel, customer and view route. These 4 tables are the only data that the system and the user will need to interact and make the web application fully functional. Hence the above tables are what the system requires to store all but not a lot of data that are needed to be stored.

## 3.4 Sequence Diagram

The sequence diagrams below correlate with each use case to detail how the application will execute the use cases.

### User: Login/ Register Agent

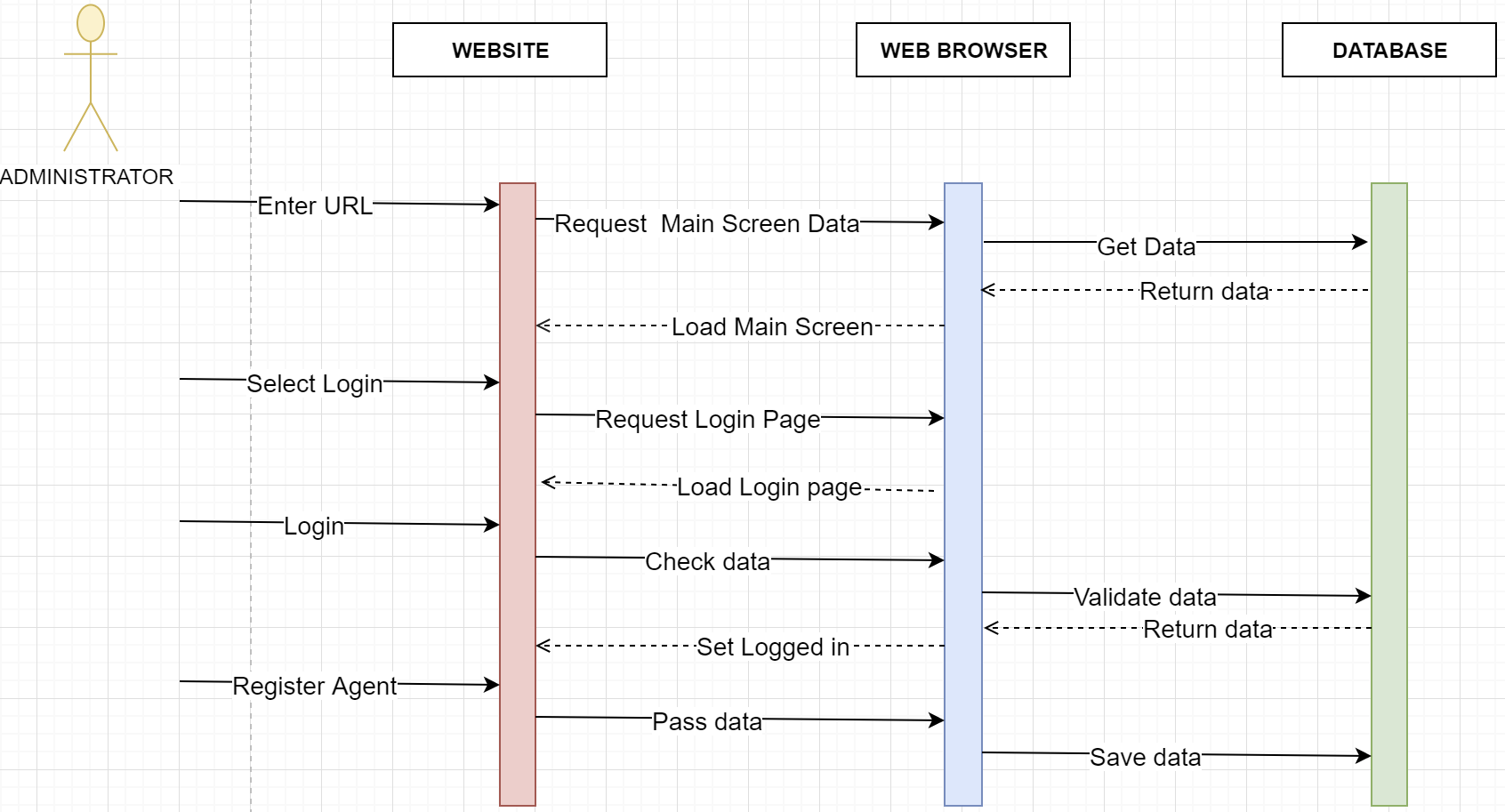


Figure : Sequence diagram (Login / Register Agent)

### Agent: View Routes

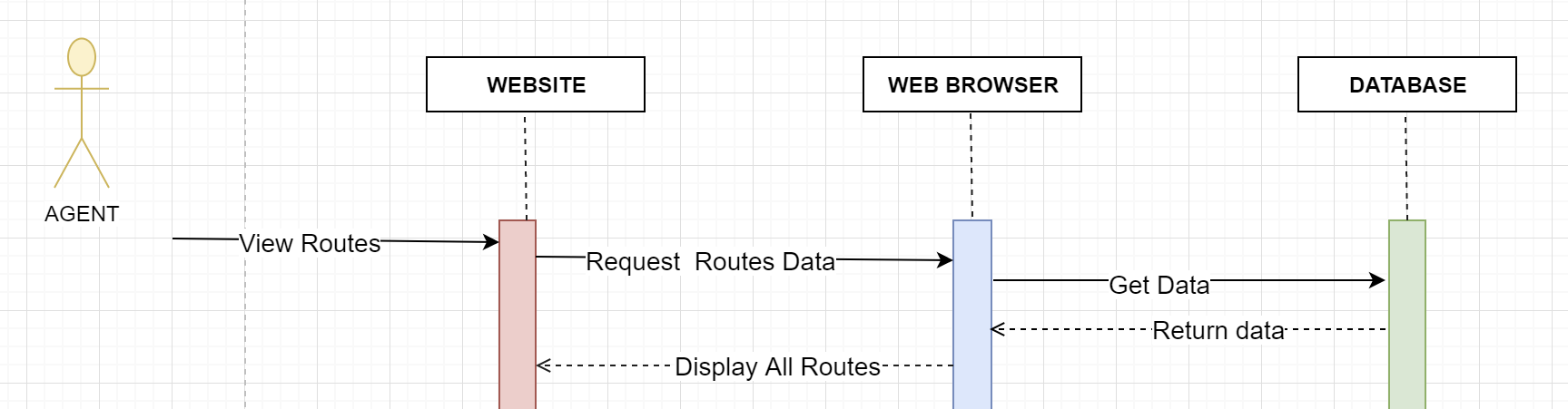


Figure : Sequence diagram (View routes)

### Agent: Book Vessel

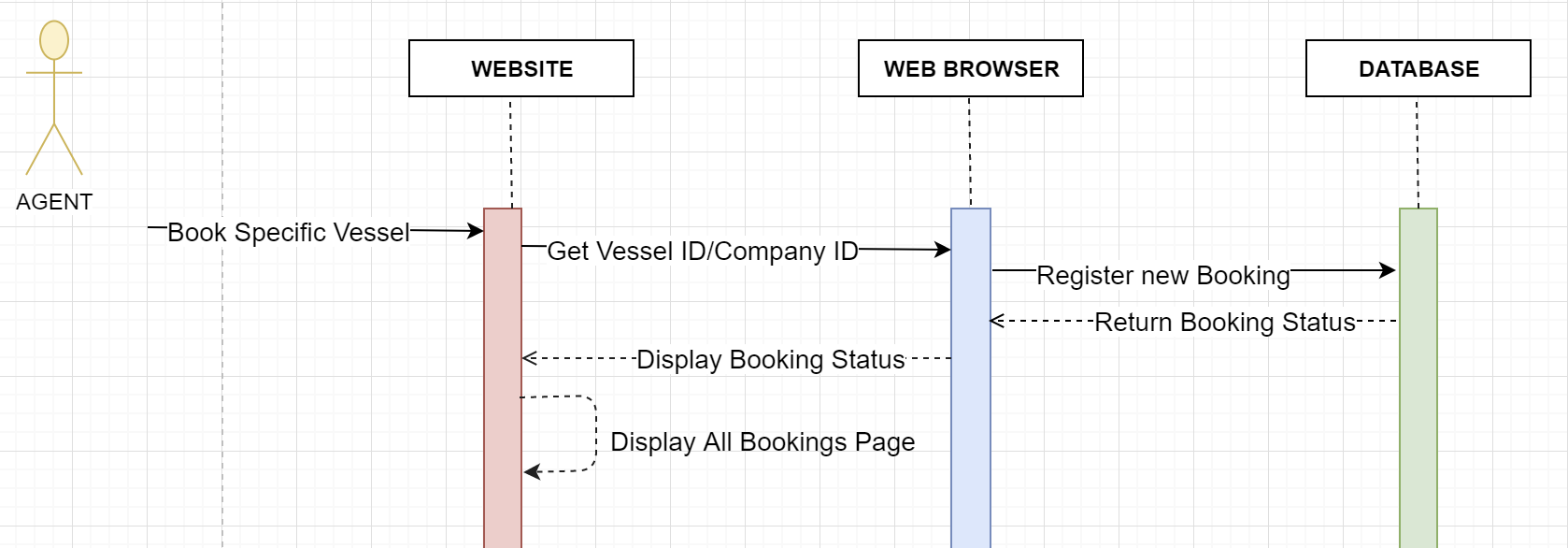


Figure : Sequence diagram (Book Vessel)

### Administrator/Agent: View Booked Vessel

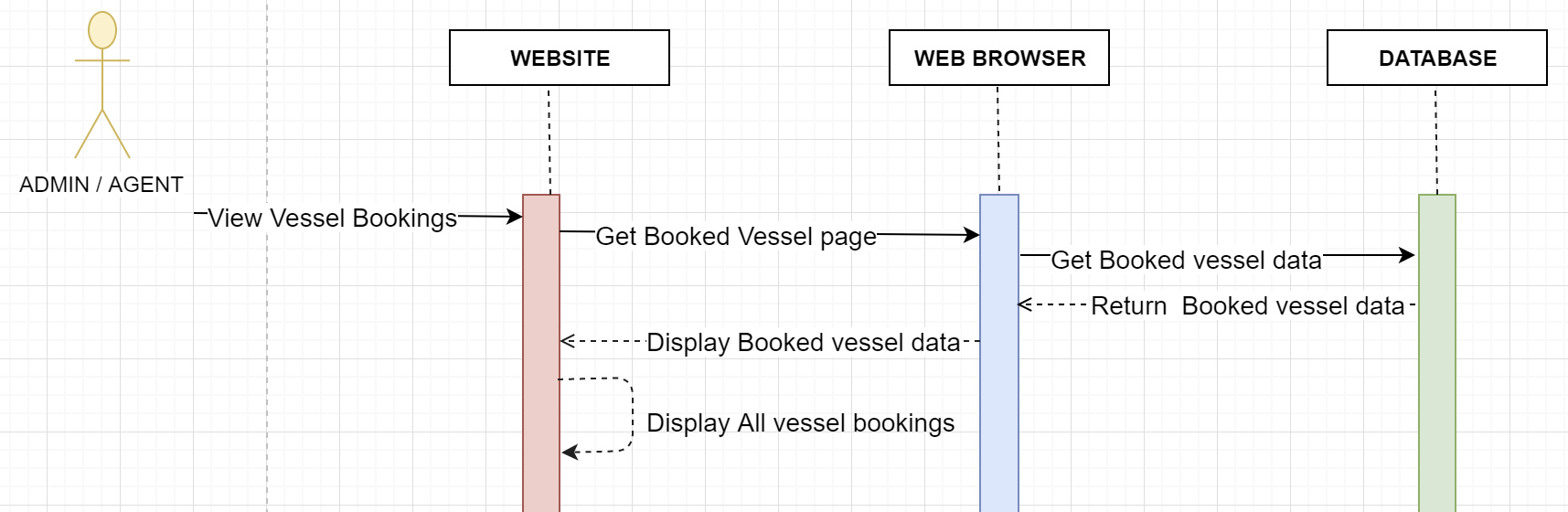


Figure : Sequence Diagram (View booked Vessel)

## 3.5 Architectural Diagram

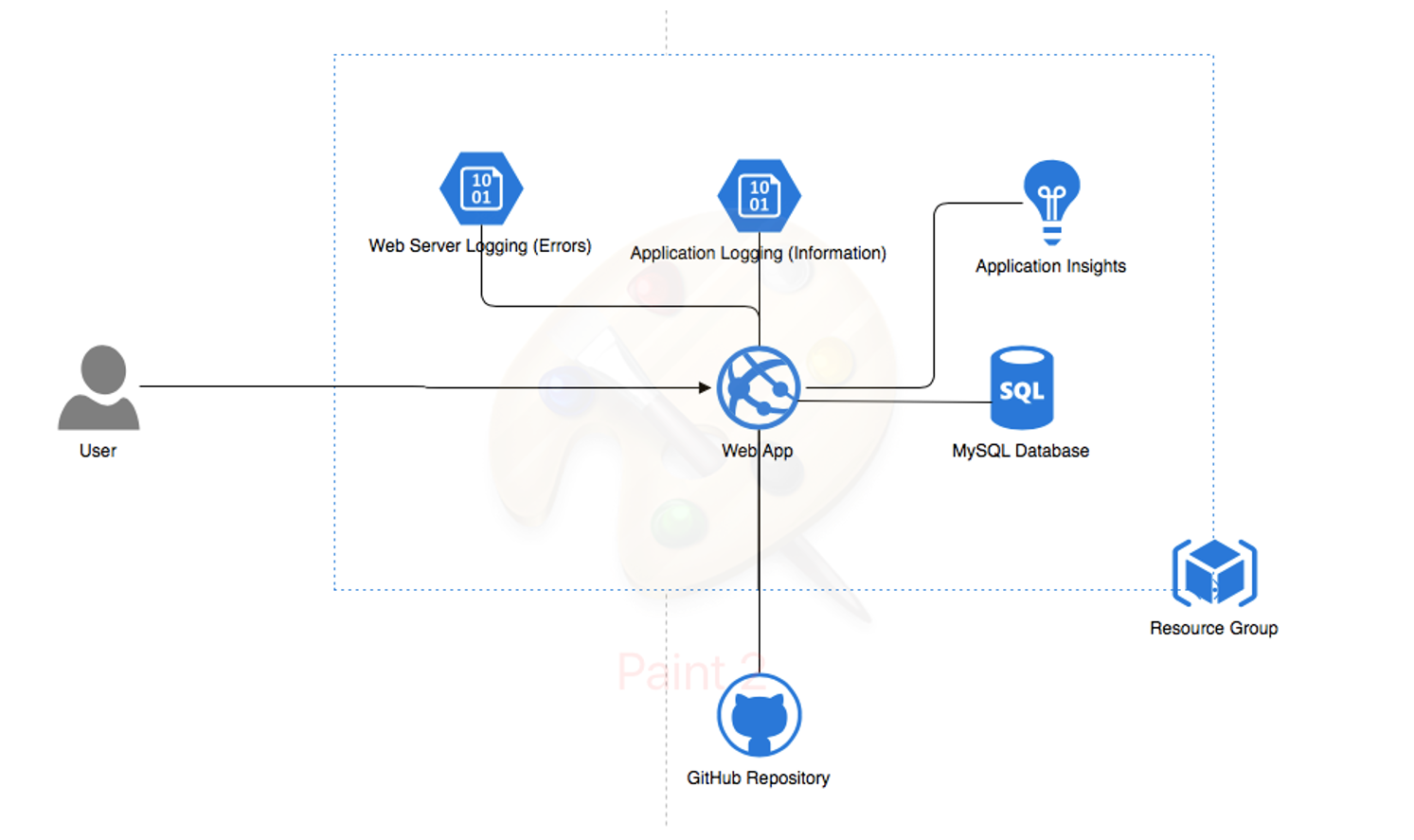


Figure : Architectural Diagram (CBS)

# 4.0 Implementation

## 4.1 Publishing CBS to Azure

In the deployment of the local web site to the cloud (Azure), there are three steps such as Creating Web App resource in Azure, Publishing the source code filed to GitHub repository and lastly Creating an Azure database for MySQL server in Azure. The first and second step are to provide the Web Application with all necessary resources as the proposed application is a web application which stores and retrieves all data from the database. The creation of an Azure database for MySQL server in Azure Cloud involves simple steps where the user should enter the database name, the name for the resource group, select the pricing tier and create the database. After the creation of the database, Azure provides the server name, status, the resource group which is the same for the web app and Server login name. In the screenshot below, the information given by Azure will help to connect the MySQL database and Web application.

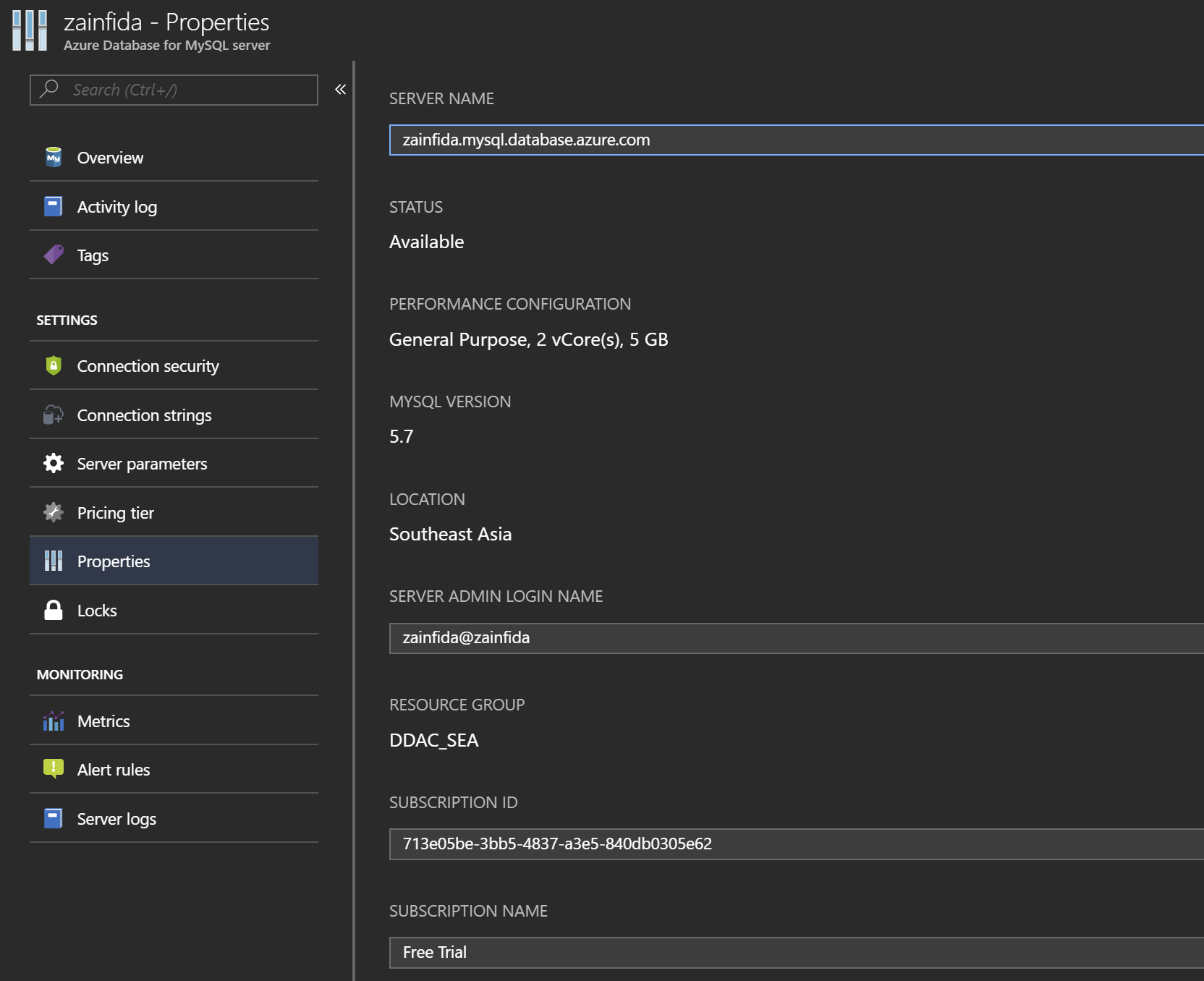
****

Figure : MySQL Information

After creation of the Azure database for MySQL server in Azure, the next phase is to create the Web App resource in Azure. To do so, the user shall fill up the form, which has values such as application name, subscription type, resource group and location of app service plan. The location of the app service plan for the proposed system is Southeast Asia region, from where the application will be available to customers for use. Moreover, the subscription can be changed, for this project the developer has chosen 30 days Free Trial so Southeast Asia was the best option, as the closest the region you chose the lesser you will be charged for the service.

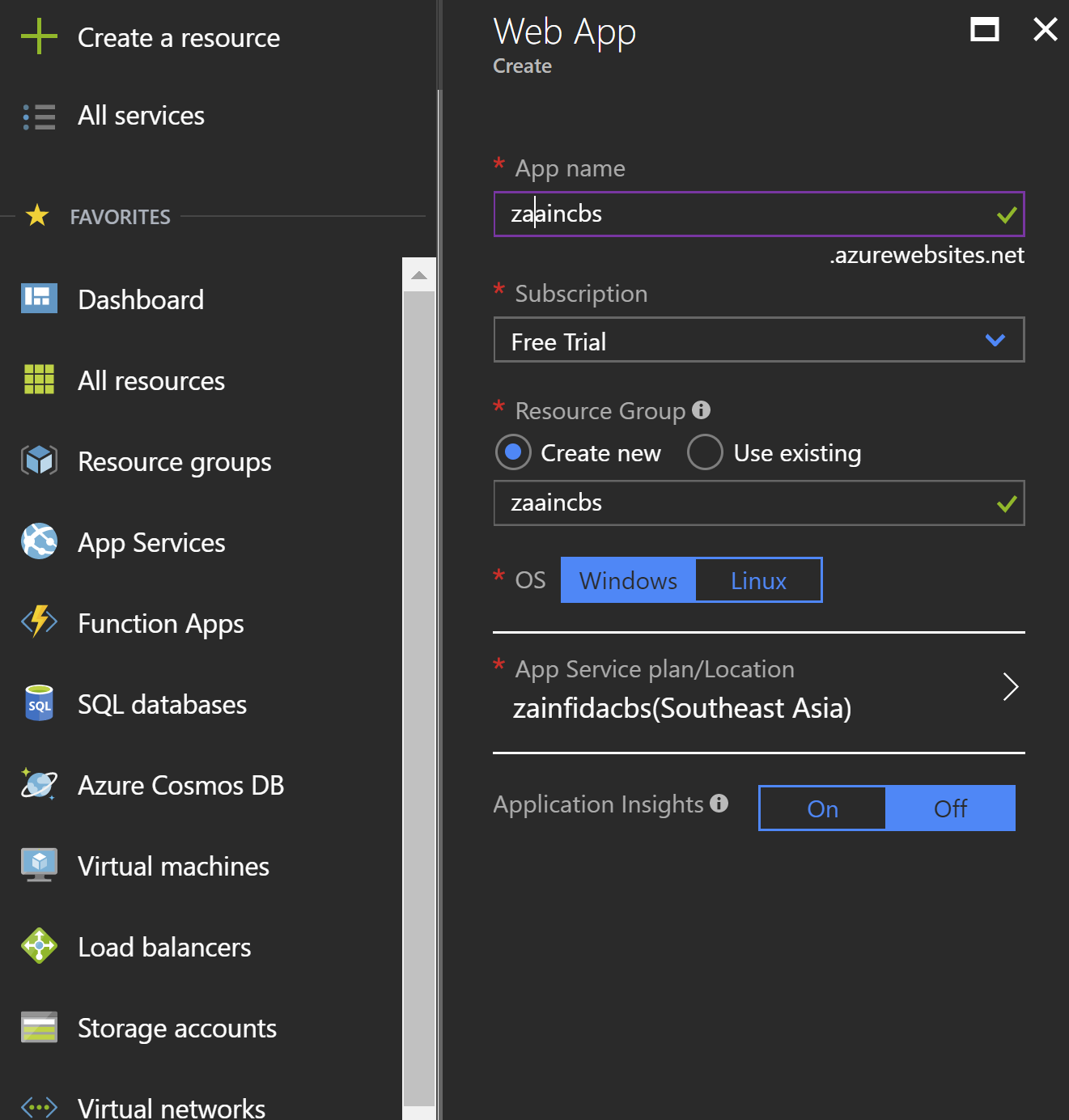


Figure : Create Web App service in Azure

After creating the Web App in Azure, the next step is to configure it by adding the server name and by selecting the deployment option, which is GitHub repository. So, after deploying the local Web Application to the Azure, user shall import the MySQL database from the local host to the Azure cloud. For this developer downloaded **MySQL Workbench 6.3** and imported the database on Workbench from the local pc. Workbench helps migrate the MySQL database to the azure cloud. Once the MySQL database is migrated and successfully connected to the Web app on cloud. The Web app is ready to be accessed on Azure platform with the database connection from Workbench. So, after deploying the local Web Application to the Azure and migrating the MySQL database to Azure, the user should just click on the URL which is given in the right corner of the Overview of the Web App.

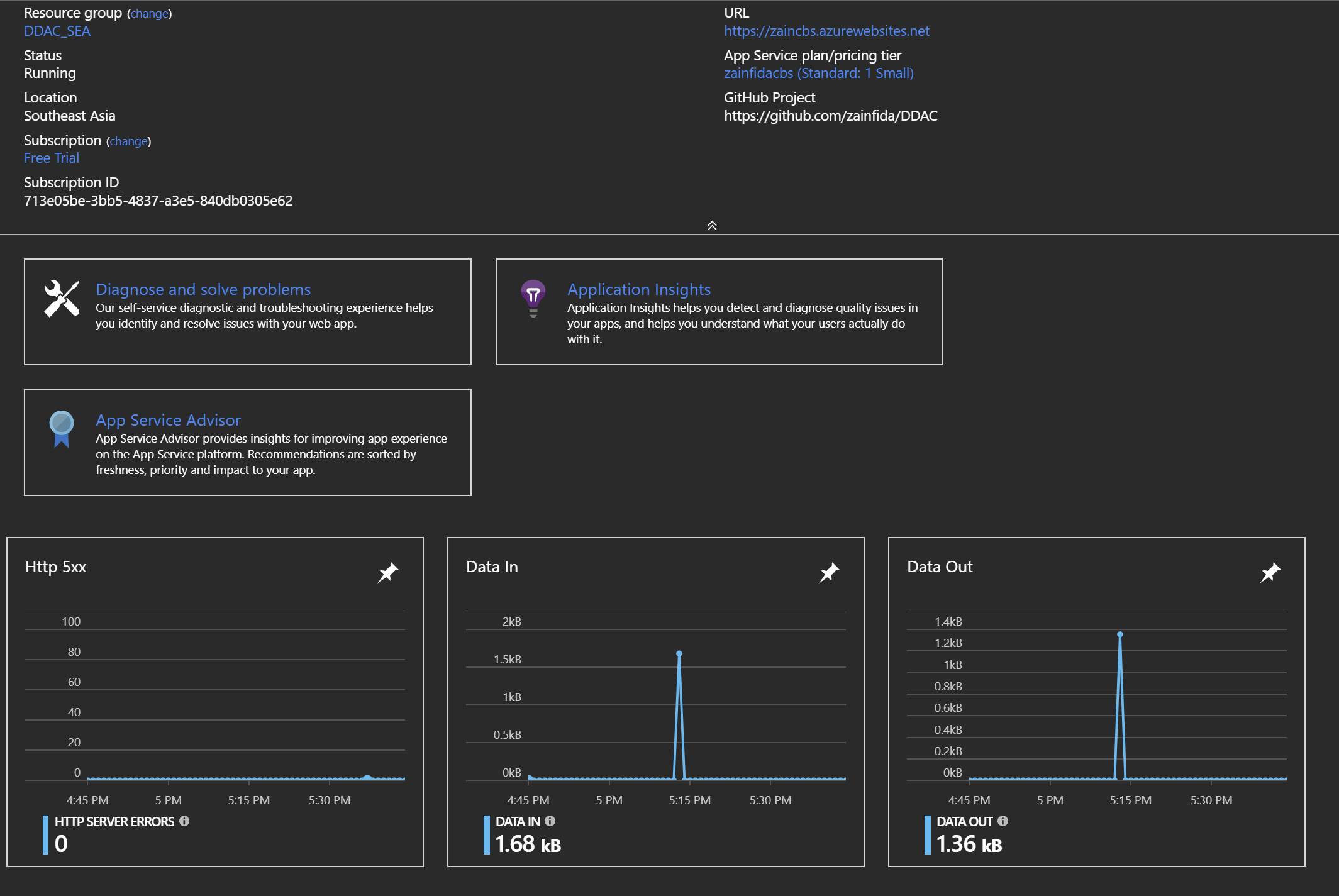


Figure : Overview of the Web App

## 4.2 Application Scaling

Conferring to (Micorost Azure, 2017), Azure cloud environment provides the web service for auto scaling which the resource can be dynamically allocated based on the needs to match with performance requirement. This is particularly important when CBS Container Booking Web Application increase the workload, more resource is needed to maintain the desired performance levels and satisfy service-level agreements (SLAs). And it is no worry for the extra resource allocated during high peak, it will be a waste as they will be automatically deallocated when no longer needed to minimize the operation cost. It has the advantage of not having a dedicated personal to always monitor the server performance and deciding on whether to proceed with allocating or deallocating more resource.



Figure : Pricing Tier for CBS

There are some of the pricing plan shown in the figure above, which covers category of basic and standard that are providing different offers. The reasons considered when selecting standard plan rather than basic plan, is it provide the extra essential features to the company which are discussed below.

First, it has greater storage than any basic plan which is 50 GB that could handle a larger amount of transaction within a given time. Next, with the custom domain and SSL SNI Incl, provided in both plan, standard plan offers extra benefit, and IP SSL Support. Also that, the auto scale is another common offer, but standard plan allows the scaling up to 10 instances which is extremely useful when the application is expected to be visited by customers from all over the world. The rest would be the extra features only provided in standard plan which the first would be daily backup of application setting and data to ensure data consistency. Secondly, it provided 5 deployment slot which holding the recent deployment restore point in case the unexpected situation happened. Finally, it allows the traffic manager to handle the connection between client and server to ensure the stable communication by having them connected to nearest server from their region.

While creating an Azure database for MySQL in Azure, we chose 2 vCores, 5 GB memory and Geo-redundant with a backup retention period of 7 days for CBS. This cost us around RM 406 per month. Since we are using free trial with a credit balance of 900 per month, we have enough balance for the chosen specifications. Below is the screenshot showing the chosen specifications.

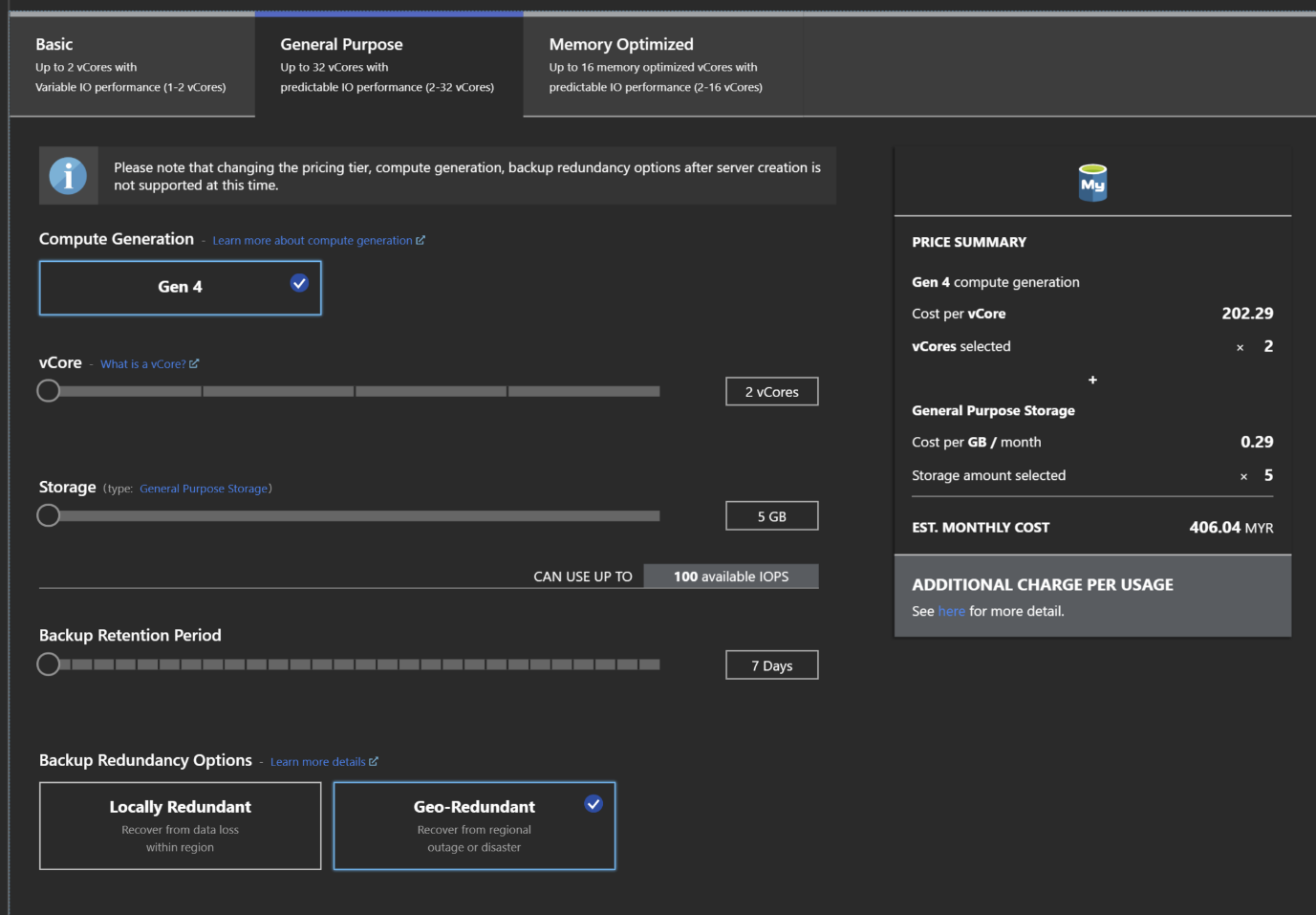


Figure : Chosen CBS MySQL specifications

# 5.0 Test Plan

## 5.1 Functional

The purpose of unit testing was to make sure that all the functions of the web application are working without errors. The table below demonstrates the unit testing of the web application. There are several functions to be tested such as: View Routes, Book Vessels, Book Bays, and View all Bookings. First test case will test the View Routes feature, second test case will test the Book Vessel function, third test case will test the Book Bay Function and last but not the least case will test the View all Bookings function.

### View Routes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name:** | | CBS- Container Booking System | | | |
| **Test No** | | 1 | | | |
| **Testing Module** | | View Routes | | | |
| **Testing Method** | | Unit testing | | | |
| **Date** | | 18/03/2018 | | | |
| **Name of Tester** | | **Zain Fida** | | | |
| **Description of module** | | The registered agent or administrator of the system clicks on the view route button on their respective index menu page, and the system should redirect the user to a new page displaying all the available vessels, their price, available bays and their schedule. | | | |
| No | Input | Expected Result | Result | Status | Error |
| 1 | Click on the view routes button | The system should display all the vessels and their prices, availability and schedule. | System successfully displayed all the vessels based on the upcoming schedule | Successful | No |

### Book Vessel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name:** | | CBS- Container Booking System | | | |
| **Test No** | | 2 | | | |
| **Testing Module** | | Book Vessel | | | |
| **Testing Method** | | Unit testing | | | |
| **Date** | | 18/03/2018 | | | |
| **Name of Tester** | | **Zain Fida** | | | |
| **Description of module** | | The registered agent of the system clicks on the book vessel button on their respective index menu page, the system should redirect the agent to a new page displaying all the upcoming scheduled vessels, along with their price, available bays and their schedule. | | | |
| No | Input | Expected Result | Result | Status | Error |
| 1 | * Select a suitable upcoming vessel. * Click on the book button for desired vessel | The system should display all the vessels and their prices, availability and schedule. Upon clicking the book vessel button. | System successfully displayed all the vessels based on the upcoming schedule. And successfully redirect the user to the booking confirmation page and showed a successful booking confirmation message | Successful | No |

### Book Bay

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name:** | | CBS- Container Booking System | | | |
| **Test No** | | 3 | | | |
| **Testing Module** | | Book Bay | | | |
| **Testing Method** | | Unit testing | | | |
| **Date** | | 18/03/2018 | | | |
| **Name of Tester** | | **Zain Fida** | | | |
| **Description of module** | | The registered agent of the system clicks on the book button for the desired vessel and the system should redirect the user to a new page displaying number of bays to be booked and for who (selecting the register company or customer name from the drop down list | | | |
| No | Input | Expected Result | Result | Status | Error |
| 1 | Select the registered customer from drop-down list and enter number of bays for the desired Vessel | After the book button is clicked for the desired vessel user will see a new page where agent will enter number of bays and select the customer company from the drop-down list and click the book button. System shall show a message and he transaction is recorded and update the data in the database | System successfully displayed the book bay page and after entering the number of bays and selecting the customer from the drop-down list and clicking the book button, showed the booking confirmation message. | Successful | No |

### View all Bookings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name:** | | CBS- Container Booking System | | | |
| **Test No** | | 4 | | | |
| **Testing Module** | | View all Bookings | | | |
| **Testing Method** | | Unit testing | | | |
| **Date** | | 18/03/2018 | | | |
| **Name of Tester** | | **Zain Fida** | | | |
| **Description of module** | | The registered agent and administrator of the system clicks on the view bookings button on their respective index menu page of the user, to view all the bookings made by respective agents of the system. | | | |
| No | Input | Expected Result | Result | Status | Error |
| 1 | Click on the view Booking button on the index menu page. | After the view bookings button is clicked, the system shall redirect the user to the bookings page, where all the bookings made by agents shall be displayed with their respective details of each booking. | System successfully displayed the view all bookings page and correctly displayed each booking with their desired details | Successful | No |

## 5.2 Performance

The performance test has been conducted for the online container booking system to check the performance of the web application hosted in Azure Cloud. The application had 1000 users which created by the Azure portal to perform the test. The test duration is 5 minutes. Within the given range the system should handle the 1000 user’s requests. Azure allows to create the test cases for the performance test.

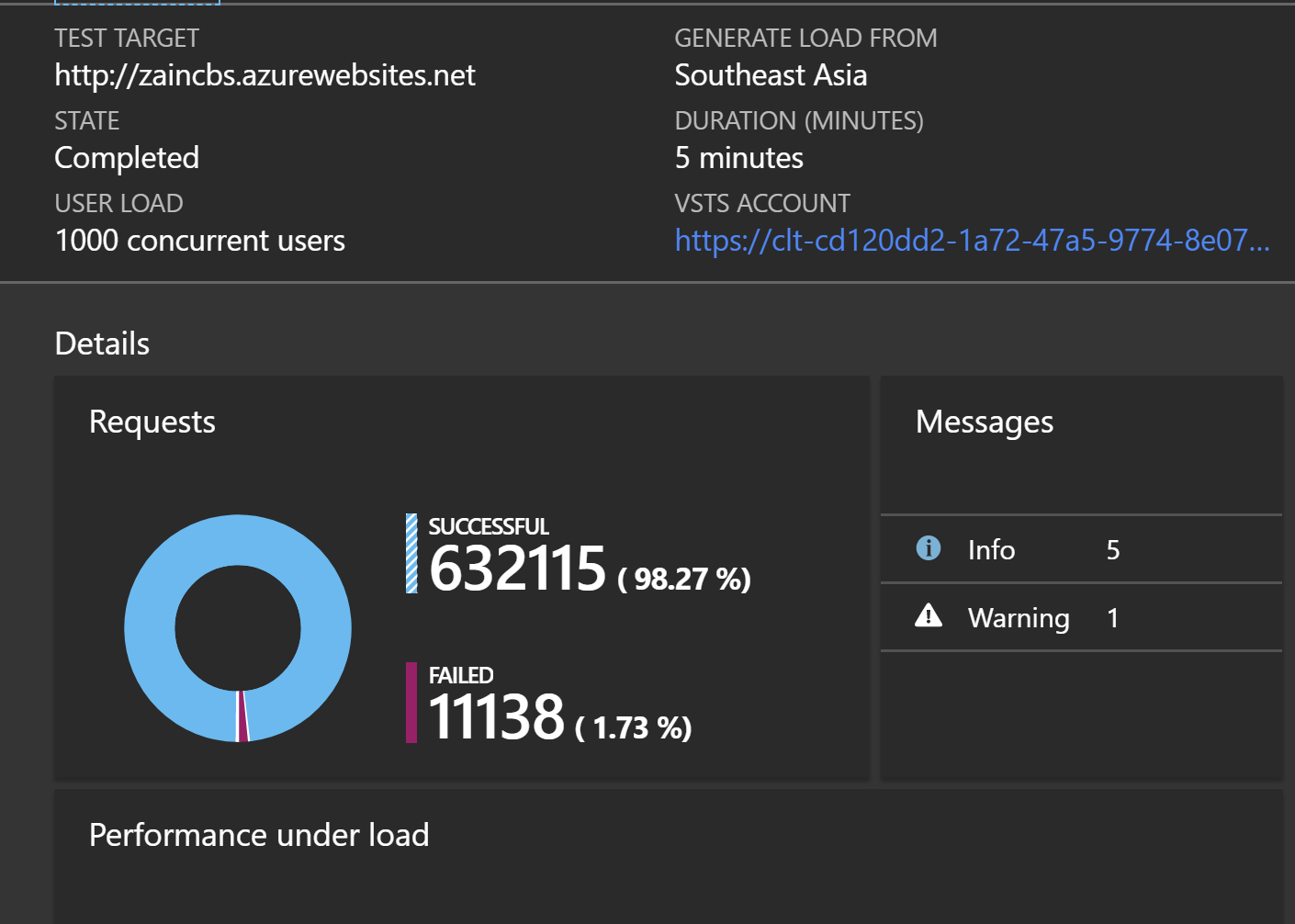
****

Figure : Performance test for the CBS in Azure

The result of the performance testing was successful. The Web App can operate, monitor and manage most of the 1000 concurrent requests. The Web App was able to handle the 99.27% of the request which is about 632115 requests from the simulated users. Unfortunately, the application was not able to handle 11138 request from simulated users, which is about 1.73% out of all requests. Thus, the Web App can operate most of the request and the performance test was completed successfully.

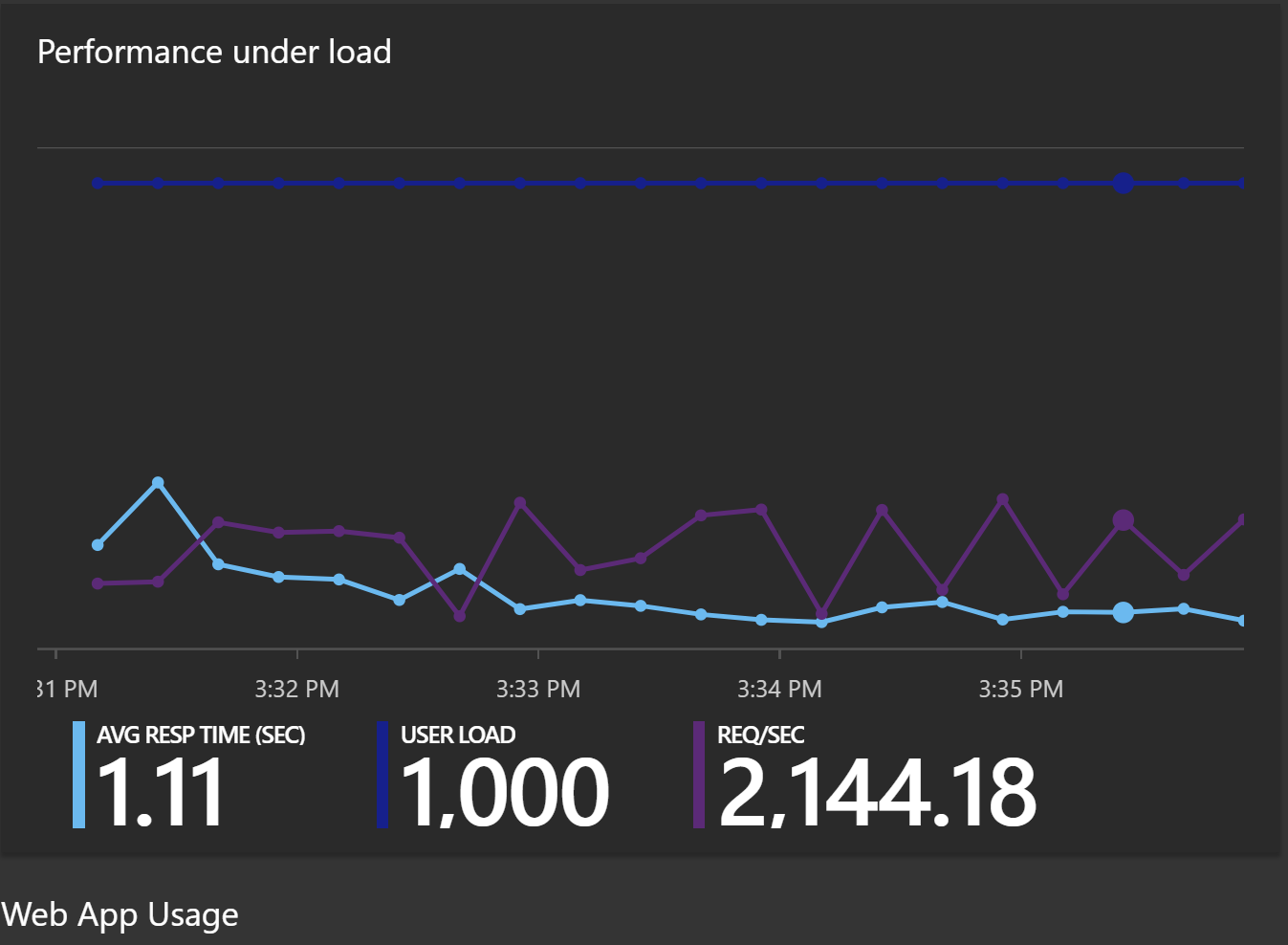
****

Figure : Performance test under load for CBS

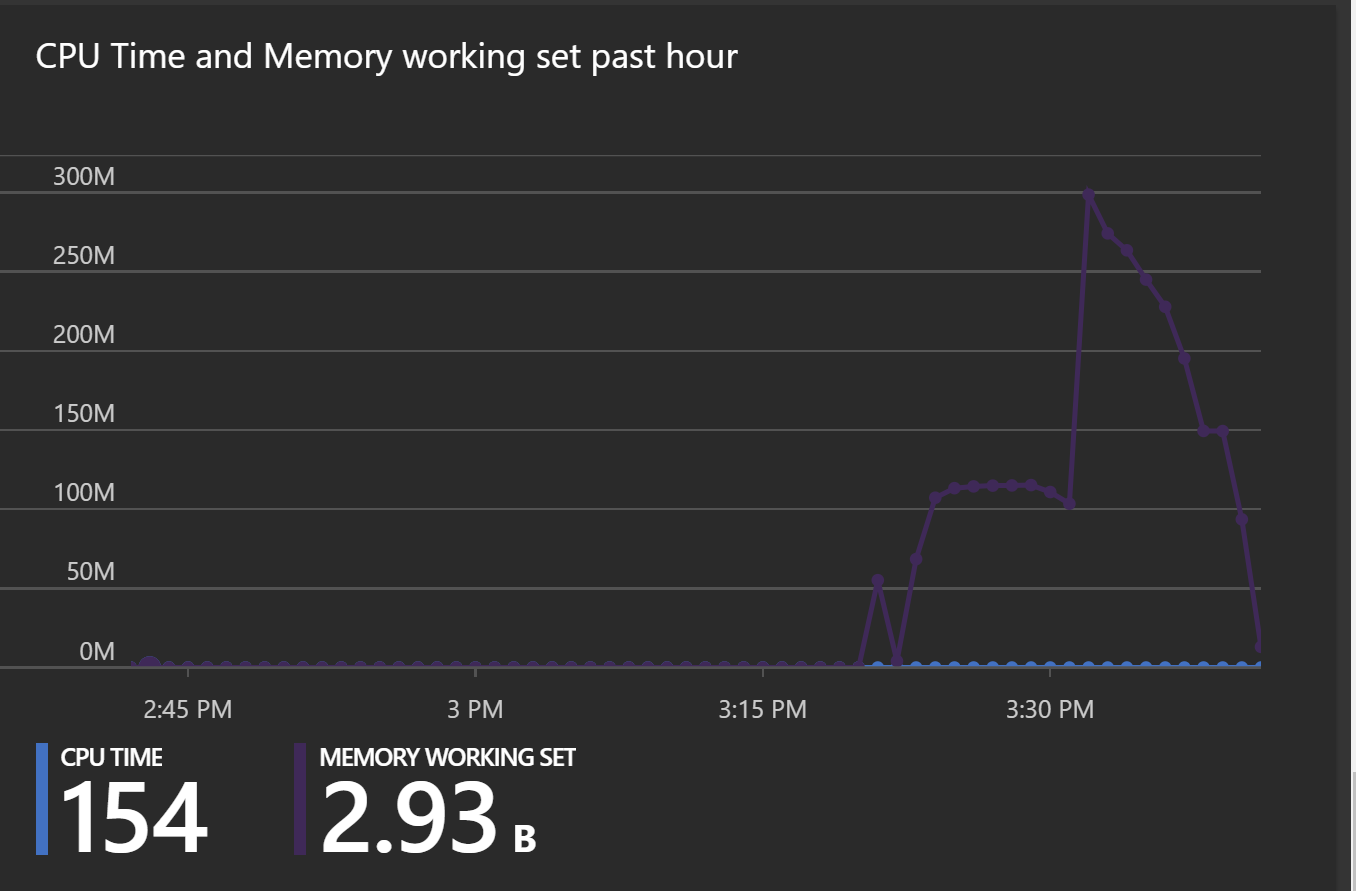
****

Figure :Results for the performance test for CBS in Azure

# 6.0 Conclusion

The project was successfully developed with deployment on cloud environment, Azure. Throughout the project, developer has gained a good understanding on cloud computing in its various forms and how can Microsoft Azure fit with the web application by publishing the CBS Container booking system to Azure. Next, the Microsoft Azure development environment has been thoroughly explored to make use of each provided component, so the web application is more effective and efficient. Besides that, developer also grow in cloud application development through concept realizing and practical work when designing, implementing and deploying the application on Azure. Finally, there is quite a good amount of knowledge gained about designing architecture of efficient application for deploying on Azure. In a nutshell, developer believes that all these could provide the strong foundation when proceeding to career path especially the cloud application development, which is also becoming a trend nowadays.

# 7.0 References

* Amazon Web Services, 2017. Amazon RDS Product Details. [Online] Available at: https://aws.amazon.com/rds/details/ [Accessed 22 March 2018].
* AnyChart.com, 2017. AnyChart. [Online] Available at: https://www.anychart.com/ [Accessed 21 March 2018].
* Apache Friends, 2017. XAMPP Apache + MariaDB + PHP + Perl. [Online] Available at: https://www.apachefriends.org/ [Accessed 22 March 2018].
* Cavale, A., Boucher, R. J. & Kemnetz, 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 22 March 2017].
* Engine Yard, 2017. Top 10 Advantages of Platform as a Service. [Online] Available at: http://www.engineyard.com/whitepapers/top-10-advantages-of-platform-as-aservice [Accessed 22 March 2018].
* generatedata.com, 2017. generatedata.com. [Online] Available at: https://www.generatedata.com/ [Accessed 16 March 2018].
* GitHub Inc., 2017. GitHub. [Online] Available at: https://github.com/ [Accessed 21 March 2018].
* Microsoft Azure, 2017. Autoscaling. Microsoft Azure. [ONLINE] Available at: https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling [Accessed on: 20 March 2018]
* Microsoft, 2017. 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-serveriaas [Accessed on: 21 March 2018]