

CONTENTS

1	Introduction.....	4
1.1	Background	4
1.2	Objectives.....	4
1.3	Scope	4
2	Project plan	5
3	Design	7
3.1	Implementation architecture.....	7
3.2	Design considerations	8
3.3	Modelling	9
3.3.1	Use case diagram	9
3.3.2	Use case description.....	10
3.3.3	Class diagram.....	17
3.3.4	Sequence diagrams.....	18
4	Implementation	29
4.1	Developing web app.....	29
4.1.1	ASP.NET Core and Entity Framework.....	29
4.1.2	Identity management.....	31
4.1.3	Screenshots	31
4.2	Publishing web app to cloud	38
4.2.1	App Service.....	38
4.2.2	Source control service.....	39
4.2.3	Deployment option.....	40
4.2.4	Traffic manager.....	42
4.2.5	Monitoring performance	45
4.3	Application scaling.....	46
4.3.1	Scaling up.....	46

4.3.2	Scaling out	47
4.4	Managed database	48
4.4.1	SQL database	48
4.4.2	Database scaling.....	50
4.4.3	Geo replication.....	51
5	Testing.....	53
5.1	Unit testing	53
5.2	Performance testing.....	58
6	Conclusion	61
	References.....	62
Appendix A	Source code and video.....	63

1 INTRODUCTION

1.1 BACKGROUND

Maersk Line is a global container vision and the largest operating unit of the A.P. Moller – Maersk Group. It is the largest container shipping company in the world, spanning across 116 countries. In an effort to further support business growth and increase organizational flexibility, Maersk decided to move all data centres and servers to cloud platform. Maersk Line wants to deploy a Container Management System on the cloud platform to manage the container business, reducing overall supply chain costs and manage logistics efficiently. The system should be able to manage the entire booking process from schedule search to booking confirmation. On top of that, the system should be able to scale to meet the needs of demands during peak seasons.

1.2 OBJECTIVES

The objectives of this project are:

- To design and develop a Container Management System web application on the cloud to manage containers
- To consume a relational database to store persistent data of the system
- To analyse web application performance with monitoring tools
- To scale the web application to accommodate high traffic during peak seasons
- To place source code on source control management services

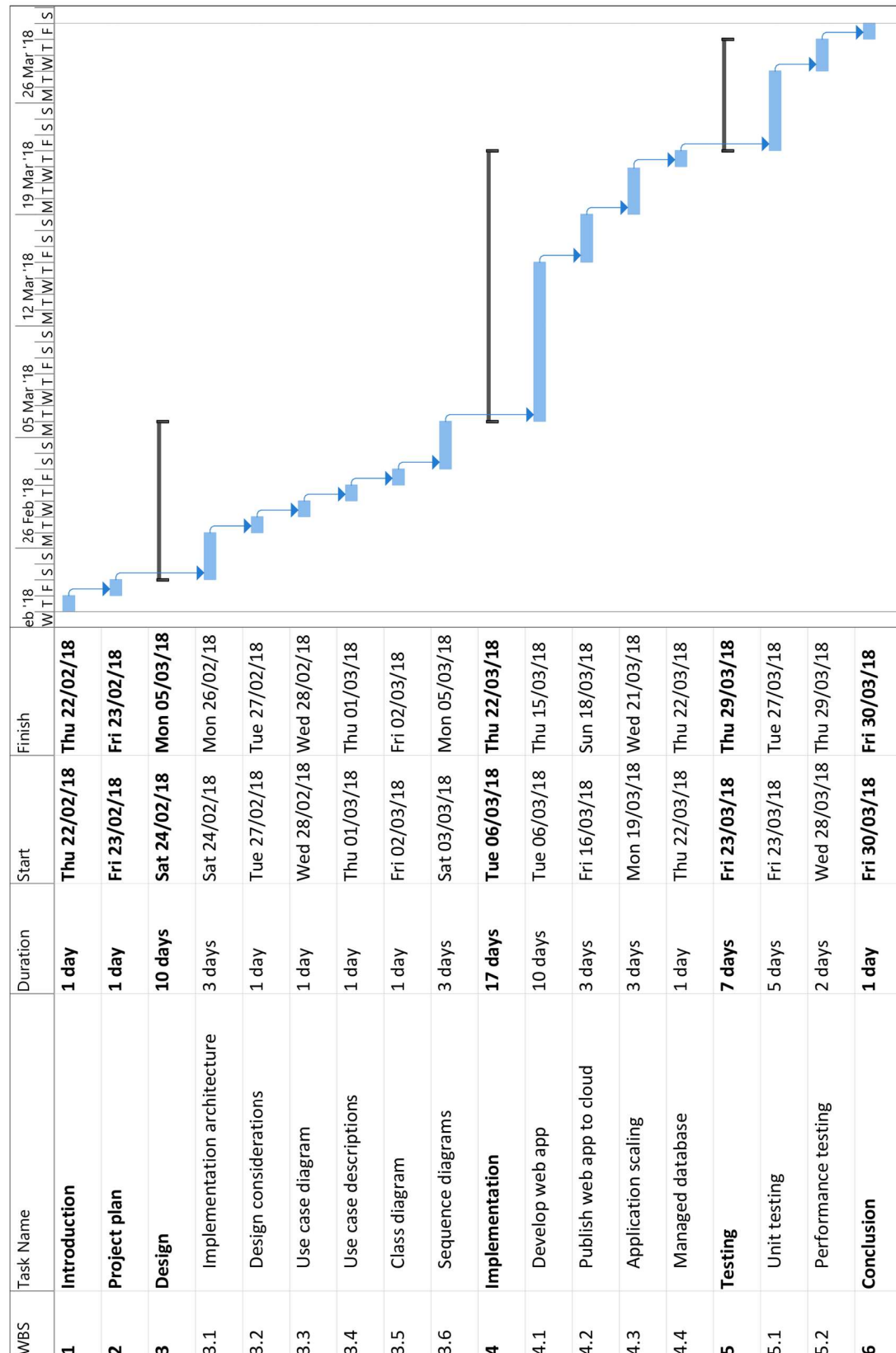
1.3 SCOPE

The Container Management System should be able to:

- Allow administrators to manage vessels
- Allow administrators to manage shipping schedules
- Allow agents to register and manage customers
- Allow agents to create bookings for customers
- Allow administrators to view bookings created by agents

2 PROJECT PLAN

WBS	Task Name	Duration	Start	Finish
1	Introduction	1 day	Thu 22/02/18	Thu 22/02/18
2	Project plan	1 day	Fri 23/02/18	Fri 23/02/18
3	Design	10 days	Sat 24/02/18	Mon 05/03/18
3.1	Implementation architecture	3 days	Sat 24/02/18	Mon 26/02/18
3.2	Design considerations	1 day	Tue 27/02/18	Tue 27/02/18
3.3	Use case diagram	1 day	Wed 28/02/18	Wed 28/02/18
3.4	Use case descriptions	1 day	Thu 01/03/18	Thu 01/03/18
3.5	Class diagram	1 day	Fri 02/03/18	Fri 02/03/18
3.6	Sequence diagrams	3 days	Sat 03/03/18	Mon 05/03/18
4	Implementation	17 days	Tue 06/03/18	Thu 22/03/18
4.1	Develop web app	10 days	Tue 06/03/18	Thu 15/03/18
4.2	Publish web app to cloud	3 days	Fri 16/03/18	Sun 18/03/18
4.3	Application scaling	3 days	Mon 19/03/18	Wed 21/03/18
4.4	Managed database	1 day	Thu 22/03/18	Thu 22/03/18
5	Testing	7 days	Fri 23/03/18	Thu 29/03/18
5.1	Unit testing	5 days	Fri 23/03/18	Tue 27/03/18
5.2	Performance testing	2 days	Wed 28/03/18	Thu 29/03/18
6	Conclusion	1 day	Fri 30/03/18	Fri 30/03/18



3 DESIGN

3.1 IMPLEMENTATION ARCHITECTURE

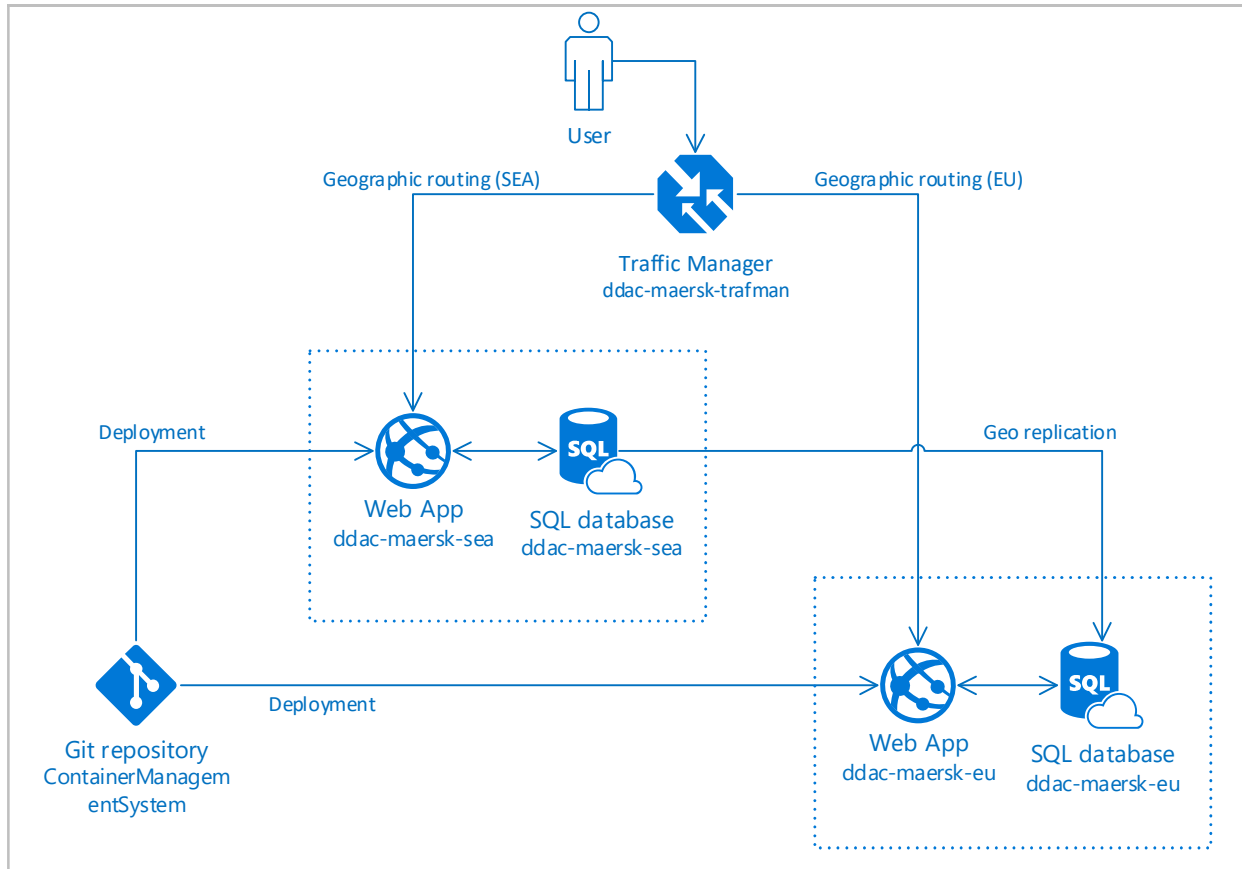


Figure 3-1 Cloud architecture

The implementation architecture involves deploying the Maersk Line Container Management System to support international business at different geographic location. Developers will push the source code to a Git repository, and the App Service will build from the source code, and publish to production. Users will access the web application through a traffic manager, routing the user to the appropriate app service based on their location. There will be one primary database, and secondary databases are replicated from that one.

3.2 DESIGN CONSIDERATIONS

Several assumptions are taken into consideration when designing the Maersk Line Container Management System.

1. The system will mainly be used for two geographic regions – Asia and Europe. The system is to be designed such that it could be expanded to reach other regions in the future.
2. Each cargo/item takes one booking entry. Multiple cargos/items per booking will not be supported.
3. Bookings are confirmed at the moment of creation. Created bookings cannot be edited after creation.
4. Administrators can see all bookings and customers created by any agent.
5. Only administrators can create new user accounts for administrators and agents.
6. It is assumed that there are no legacy systems to integrate with.

3.3 MODELLING

3.3.1 Use case diagram



3.3.2 Use case description

Use case	#1 Log in
Description	User logs in to the web app
Actors	Administrator, Agent
Pre-condition(s)	User must not be logged in yet
Post-condition(s)	User will be logged in User gains access to the other features of the system
Basic flow	<ol style="list-style-type: none"> 1. User clicks log in 2. Displays log in form 3. User submits username and password 4. Checks username and password 5. Redirect to home page
Alternative flow(s)	5a. If username or password is invalid, displays error in the form.

Use case	#2 Register agent
Description	Administrator creates a new account for agents to access the web app
Actors	Administrator
Pre-condition(s)	Administrator is logged in
Post-condition(s)	New agent account is created
Basic flow	<ol style="list-style-type: none"> 1. User clicks create new account 2. Displays registration form 3. User submits account information 4. Checks for form errors 5. Create a new account and save it in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#3 Register customer
Description	Agent creates a new customer profile
Actors	Agent, Administrator
Pre-condition(s)	User is logged in
Post-condition(s)	Customer profile is created
Basic flow	<ol style="list-style-type: none"> 1. User clicks create new customer 2. Displays form 3. User submits the form with customer details 4. Checks for form errors 5. Save new customer profile in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#4 Edit customer
Description	Agent edits existing customer profile
Actors	Agent, Administrator
Pre-condition(s)	User is logged in There are existing customer profiles
Post-condition(s)	The customer profile is updated
Basic flow	Optionally extends to Use Case #3 <ol style="list-style-type: none"> 1. User selects an existing customer profile and clicks edit 2. Displays the form with existing customer details 3. User submits the modified customer profile 4. Checks for form errors 5. Save customer profile in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#5 View customers
Description	User retrieves a list of customer profiles in the system
Actors	Agent, Administrator
Pre-condition(s)	User must be logged in There are existing customer profiles
Post-condition(s)	None
Basic flow	1. User selects “Customers” from the navigation bar 2. Retrieves all customer profiles from the database 3. Display the customer profiles in a list
Alternative flow(s)	None

Use case	#6 Add vessel
Description	Administrator adds new vessels to the system
Actors	Administrator
Pre-condition(s)	Administrator is logged in
Post-condition(s)	New vessel is added to the system
Basic flow	1. User clicks create new vessel 2. Displays empty form 3. User submits form with vessel details 4. Checks form for errors 5. Save new vessel in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#7 Edit vessel
Description	Administrator updates existing vessels in the system
Actors	Administrator
Pre-condition(s)	Administrator is logged in There are existing vessels registered
Post-condition(s)	The vessel details are updated
Basic flow	Optionally extends to Use Case #6 1. User selects an existing vessel and clicks edit 2. 2. Displays the form with existing vessel details 3. User submits the modified vessel details 4. Check for form errors 5. Save the updated vessel details in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#8 View vessels
Description	User retrieves a list of vessels in the system
Actors	Administrator, Agent
Pre-condition(s)	User must be logged in There are existing vessels registered
Post-condition(s)	None
Basic flow	1. User selects “Vessels” from the navigation bar 2. Retrieves all vessel details from the database 3. Display the vessel details in a list
Alternative flow(s)	None

Use case	#9 Add shipping schedule
Description	Administrator schedule a new shipping in the system
Actors	Administrator
Pre-condition(s)	Administrator is logged in There are existing vessels in the system
Post-condition(s)	Shipping schedule is added to the system
Basic flow	<ol style="list-style-type: none"> 1. User clicks create new shipping schedule 2. Displays empty form 3. User submits form with schedule details 4. Checks form for errors 5. Save new schedule in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

Use case	#10 Edit shipping schedule
Description	Administrator modifies an existing shipping schedule
Actors	Administrator
Pre-condition(s)	Administrator is logged in There are existing shipping schedule in the system
Post-condition(s)	Shipping schedule is updated with new details
Basic flow	<p>Optionally extends to Use Case #9</p> <ol style="list-style-type: none"> 1. User selects an existing shipping schedule and clicks edit 2. Displays form with existing schedule details 3. User submits updated schedule details 4. Checks form for errors 5. Update schedule in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

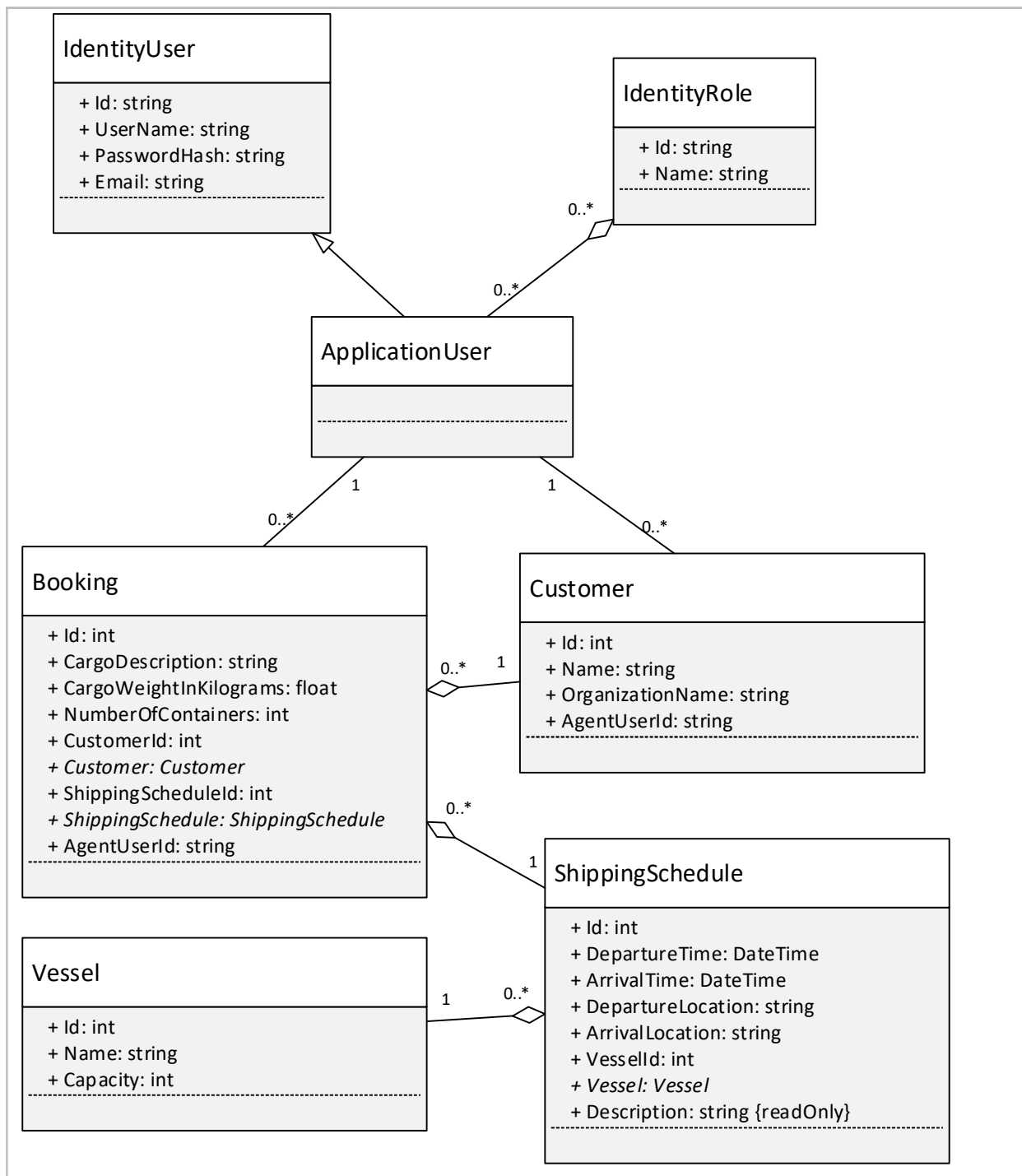
Use case	#11 View shipping schedules
Description	User retrieves a list of existing shipping schedules
Actors	Administrator, Agent
Pre-condition(s)	User is logged in There are existing shipping schedules
Post-condition(s)	None
Basic flow	<ol style="list-style-type: none"> 1. User selects “Schedules” from the navigation bar 2. Retrieves all shipping schedules from the database 3. Display the shipping schedules in a list
Alternative flow(s)	None

Use case	#12 Search shipping schedules
Description	User searches for shipping schedules by location
Actors	Administrator, Agent
Pre-condition(s)	User is logged in There are existing shipping schedules
Post-condition(s)	None
Basic flow	Optionally extends to Use Case #11 <ol style="list-style-type: none"> 1. User enters substring of a location in the search bar 2. Retrieves all shipping schedules that matches the search query from the database 3. Display the filtered shipping schedules in a list
Alternative flow(s)	None

Use case	#13 Add booking
Description	User books a shipping schedule for his or her customer
Actors	Agent, Administrator
Pre-condition(s)	User is logged in There are existing shipping schedules There are existing customers
Post-condition(s)	New booking is added to the system
Basic flow	<ol style="list-style-type: none"> 1. User clicks create new booking 2. Displays empty form 3. User submits booking details 4. Checks for errors in the form 5. Save the new booking in the database 6. Display success message
Alternative flow(s)	5a. If user input has errors, display errors in the form.

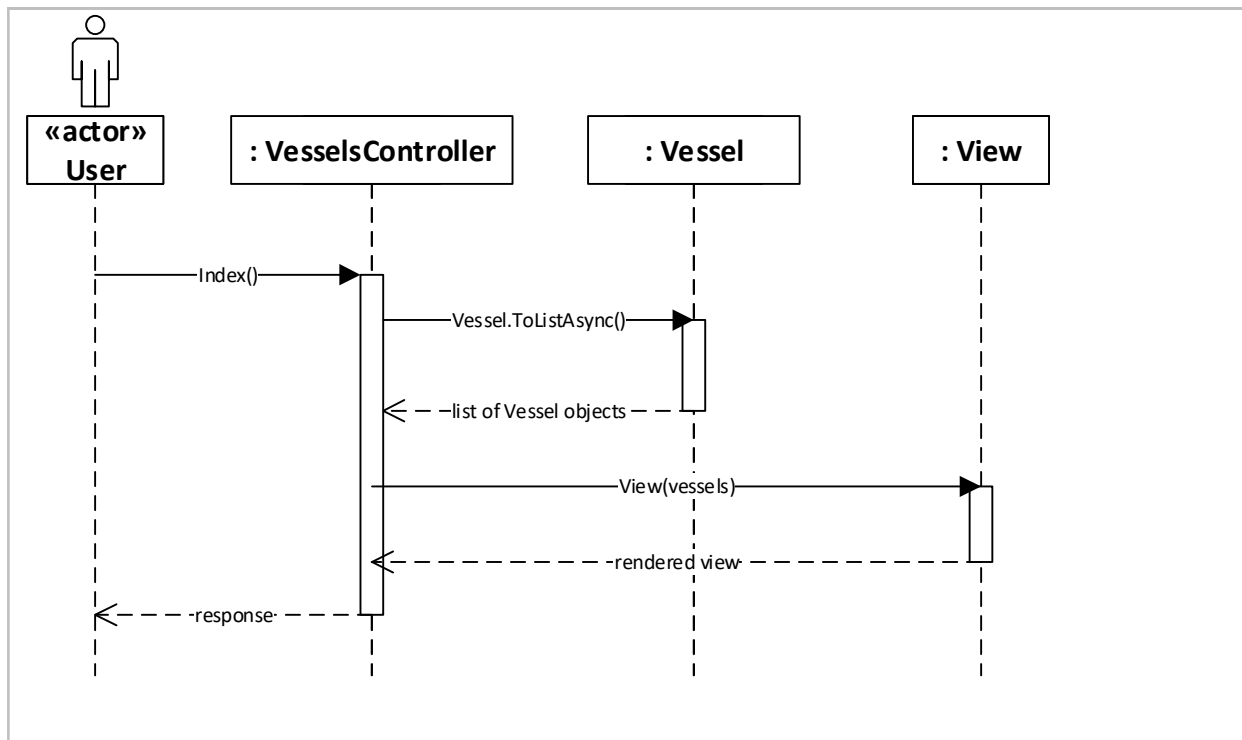
Use case	#14 View bookings
Description	User retrieves a list of shipping schedules
Actors	Agent, Administrator
Pre-condition(s)	User is logged in There are existing bookings
Post-condition(s)	None
Basic flow	<ol style="list-style-type: none"> 1. User selects “Bookings” from the navigation bar 2. Retrieves all bookings from the database 3. Display the bookings in a list
Alternative flow(s)	None

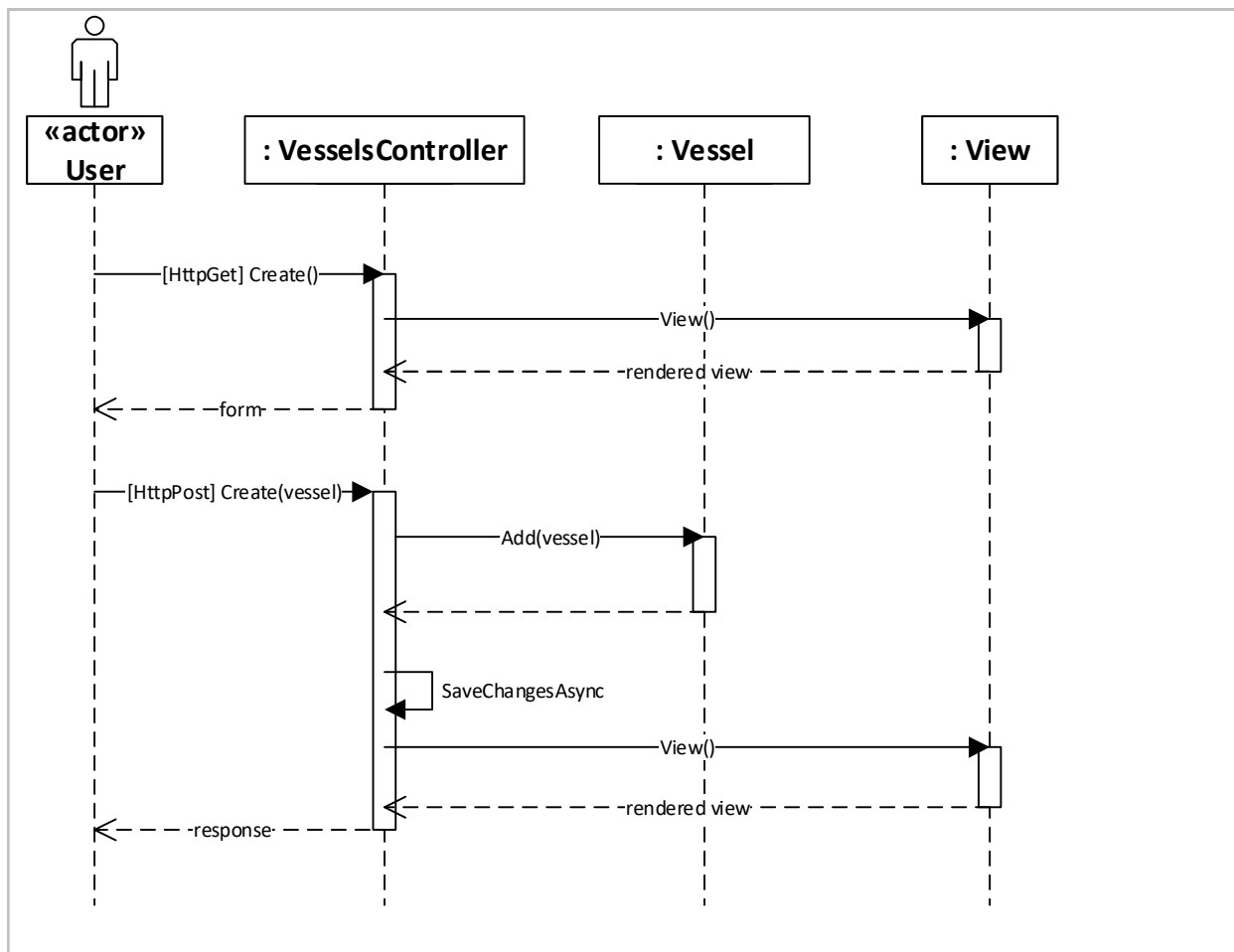
3.3.3 Class diagram

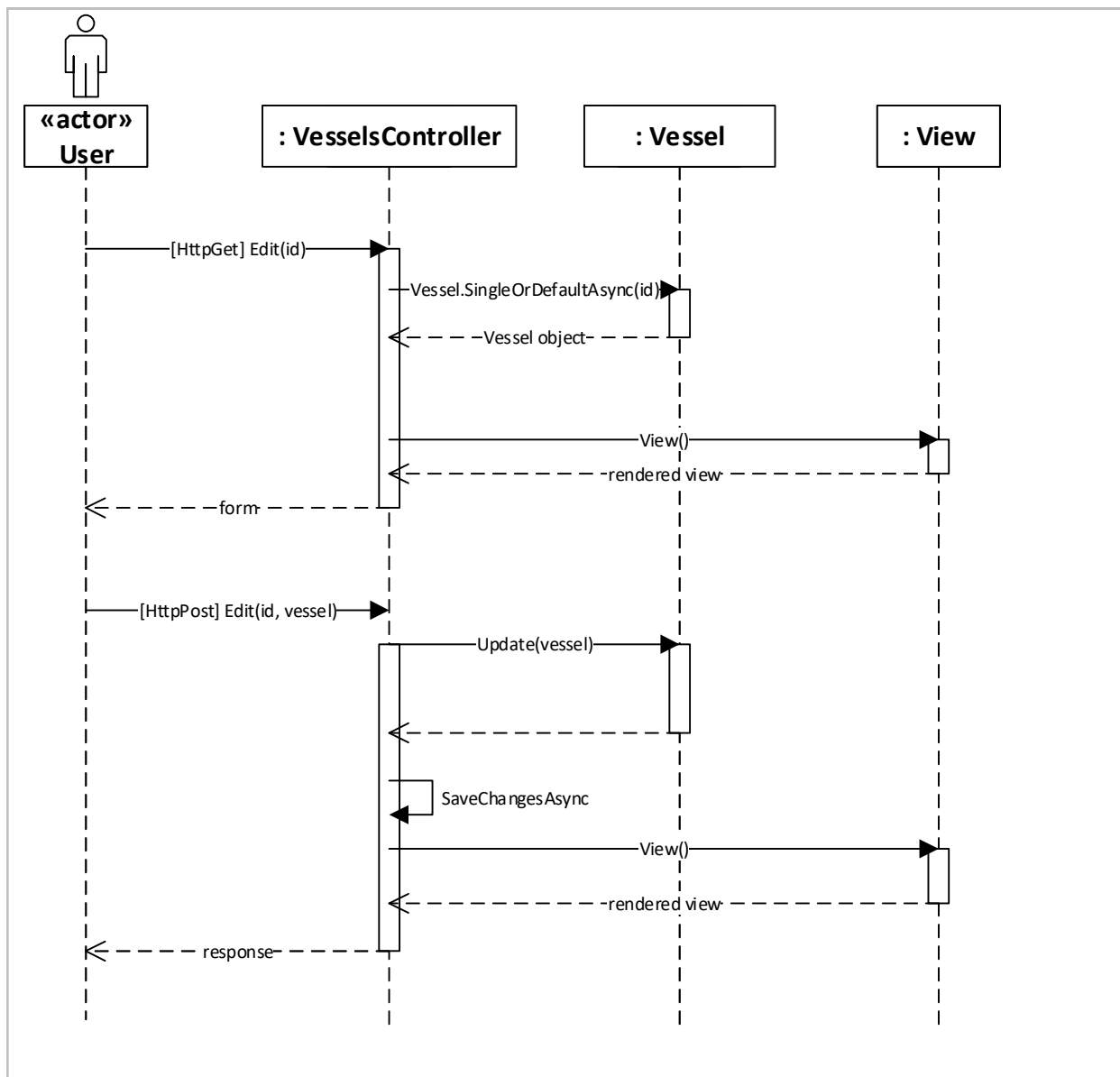


3.3.4 Sequence diagrams

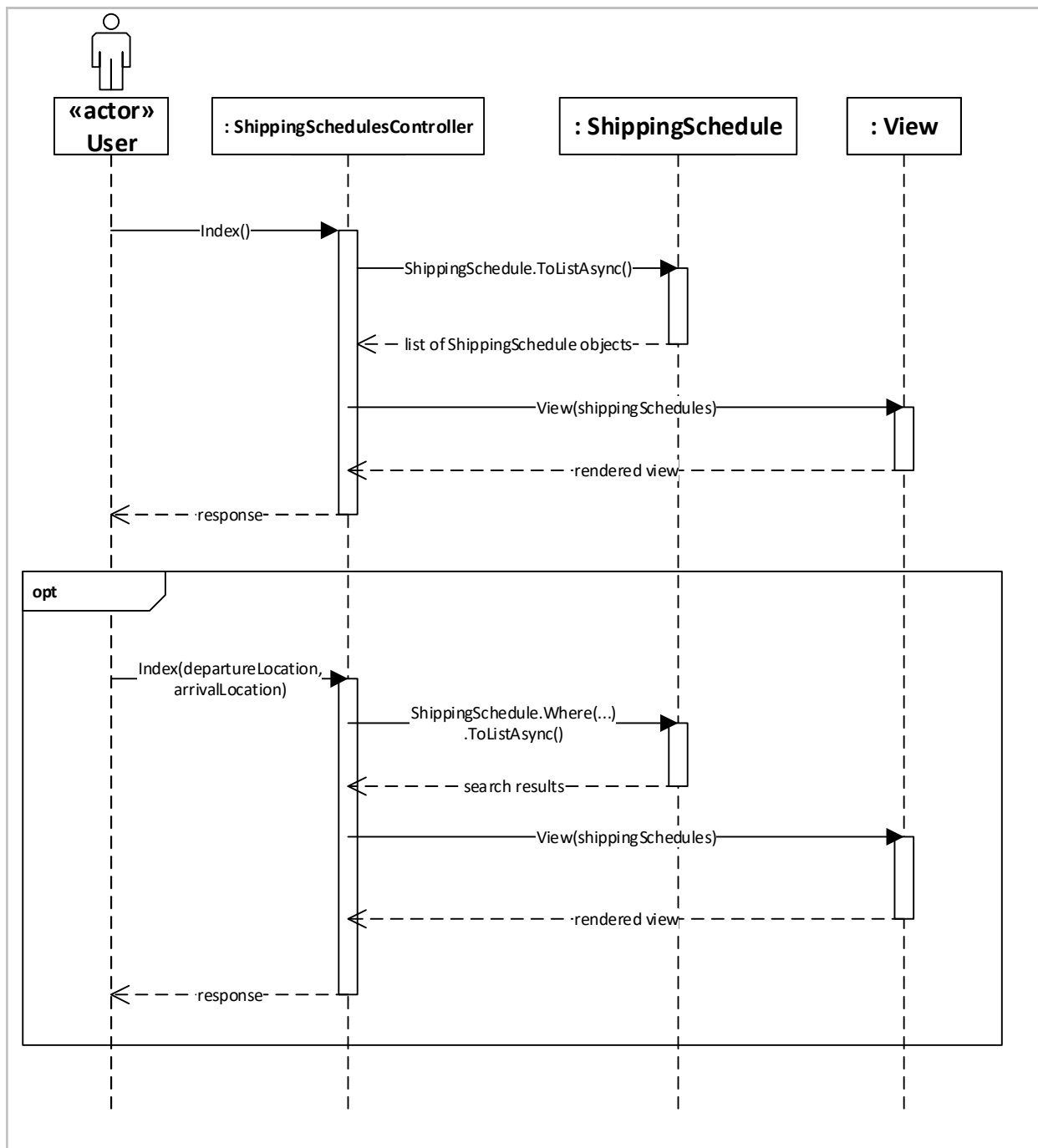
View vessels



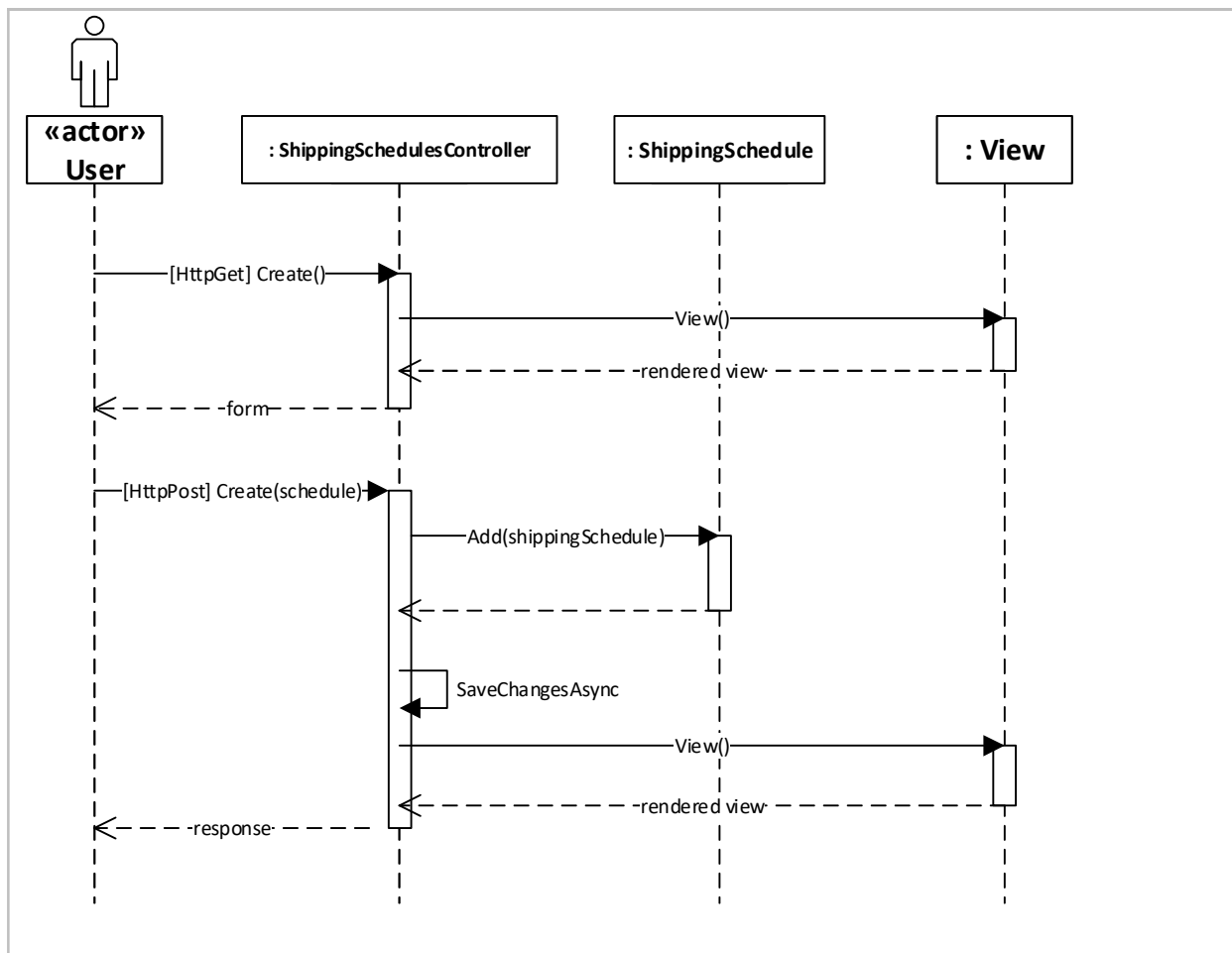
Add vessel

Edit vessel

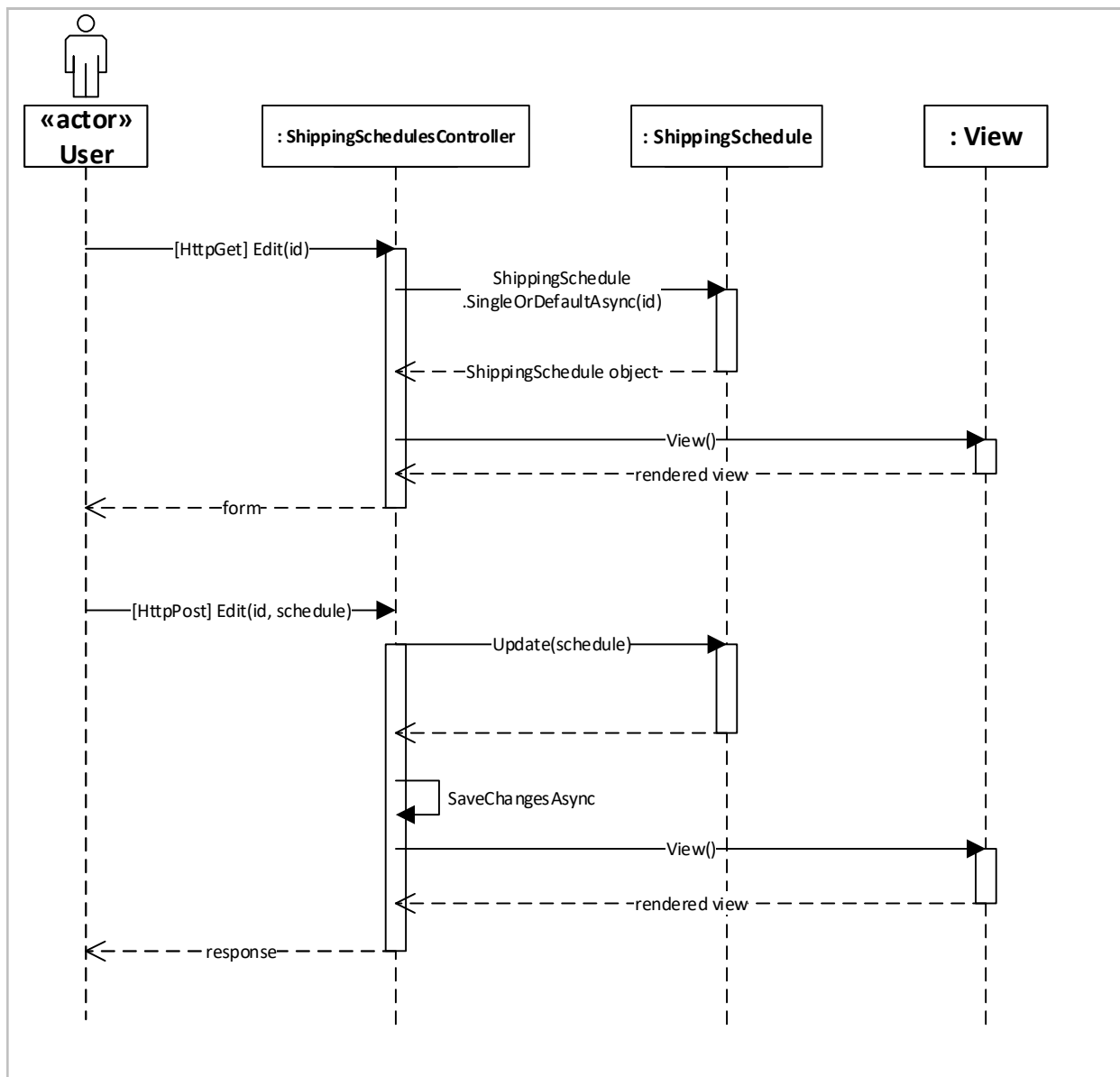
View shipping schedules

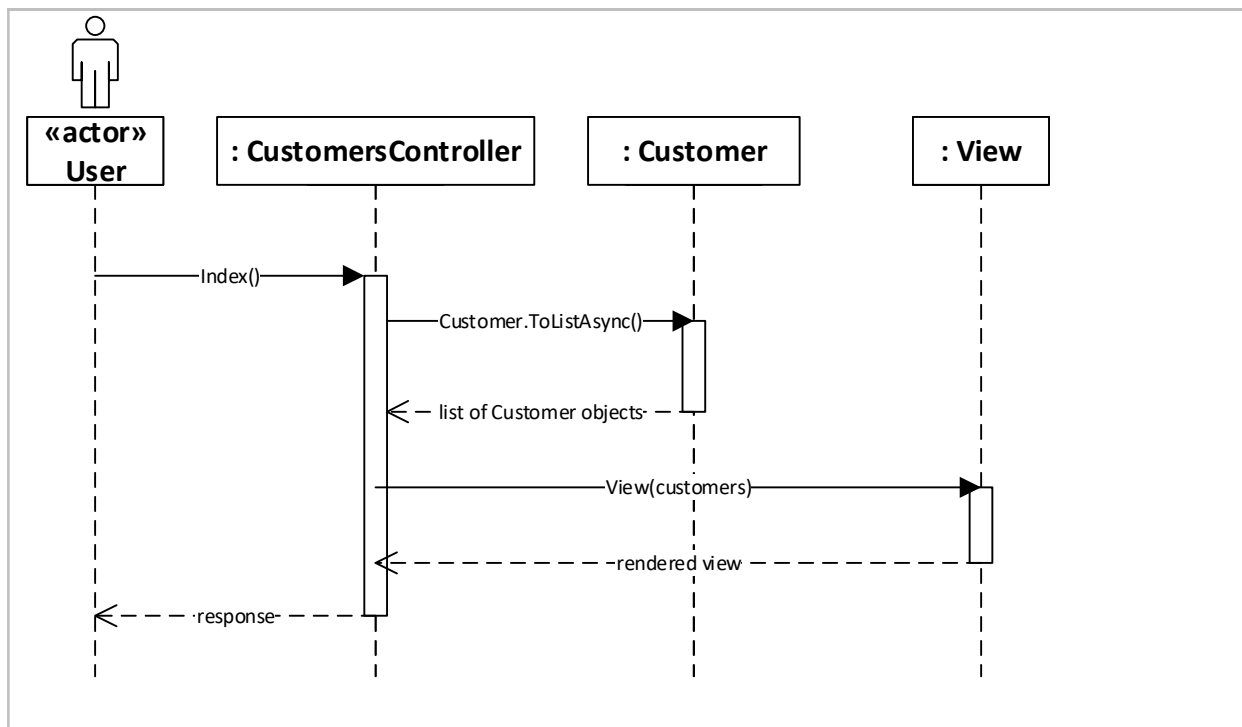


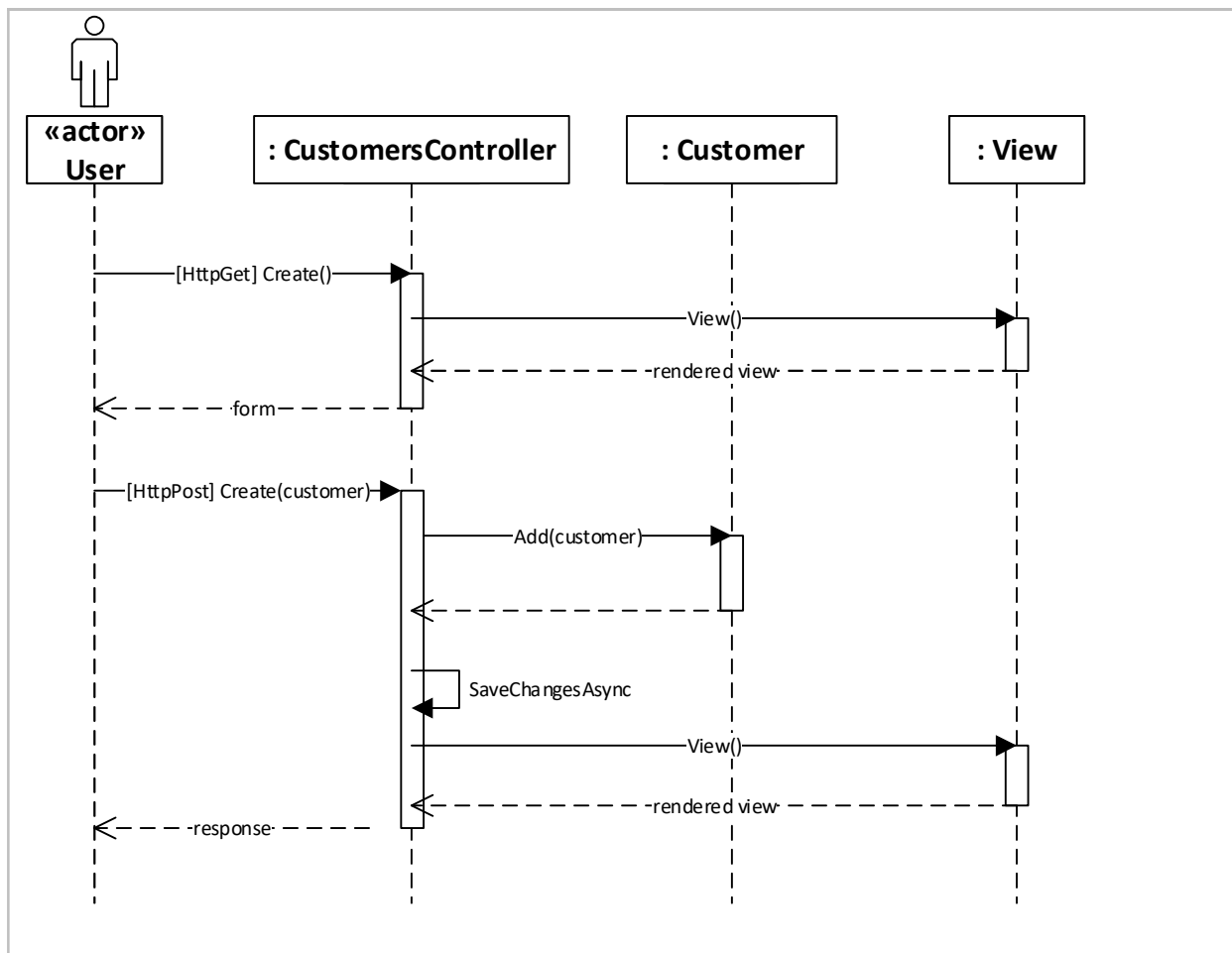
Add shipping schedule

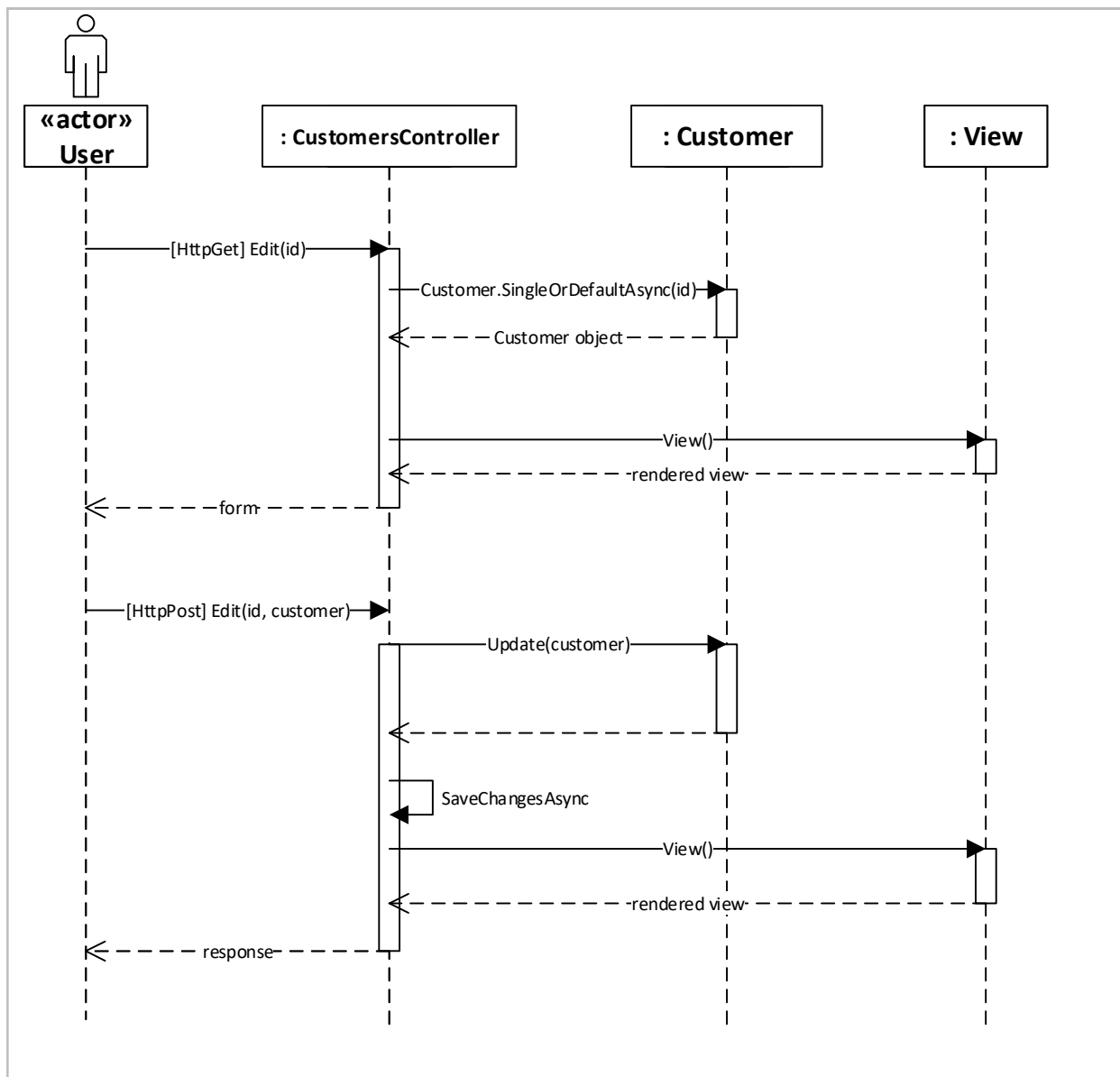


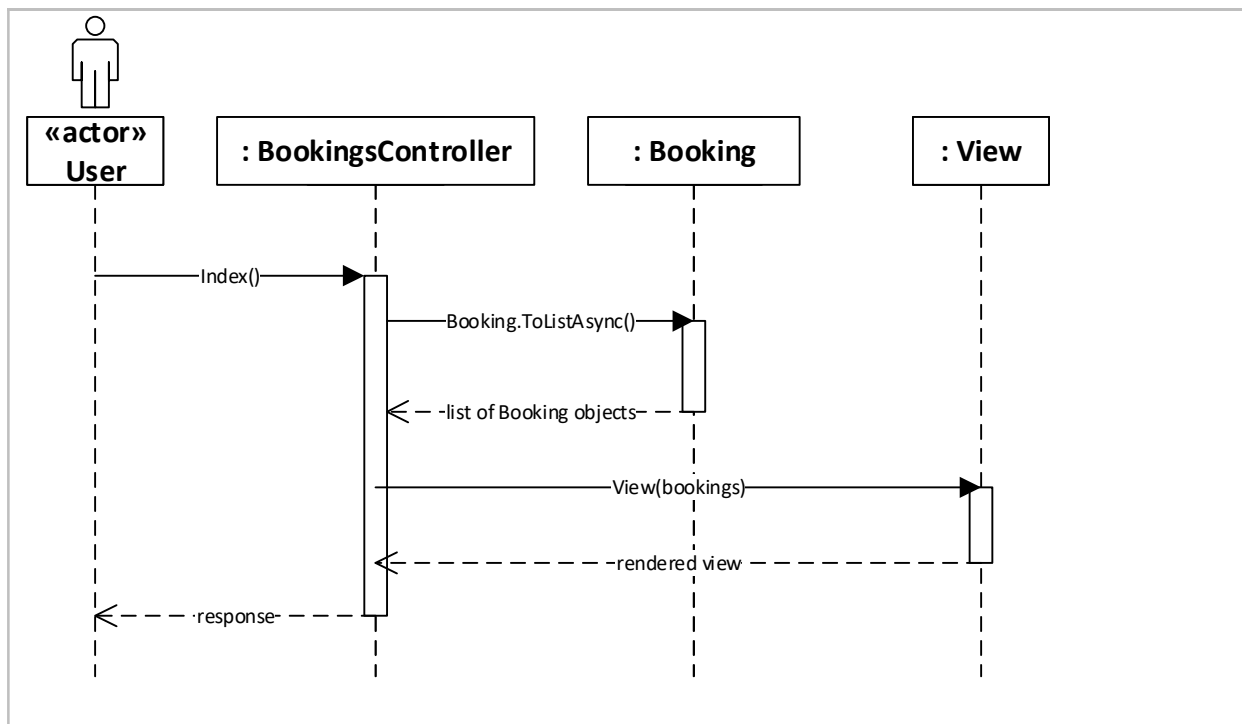
Edit shipping schedule

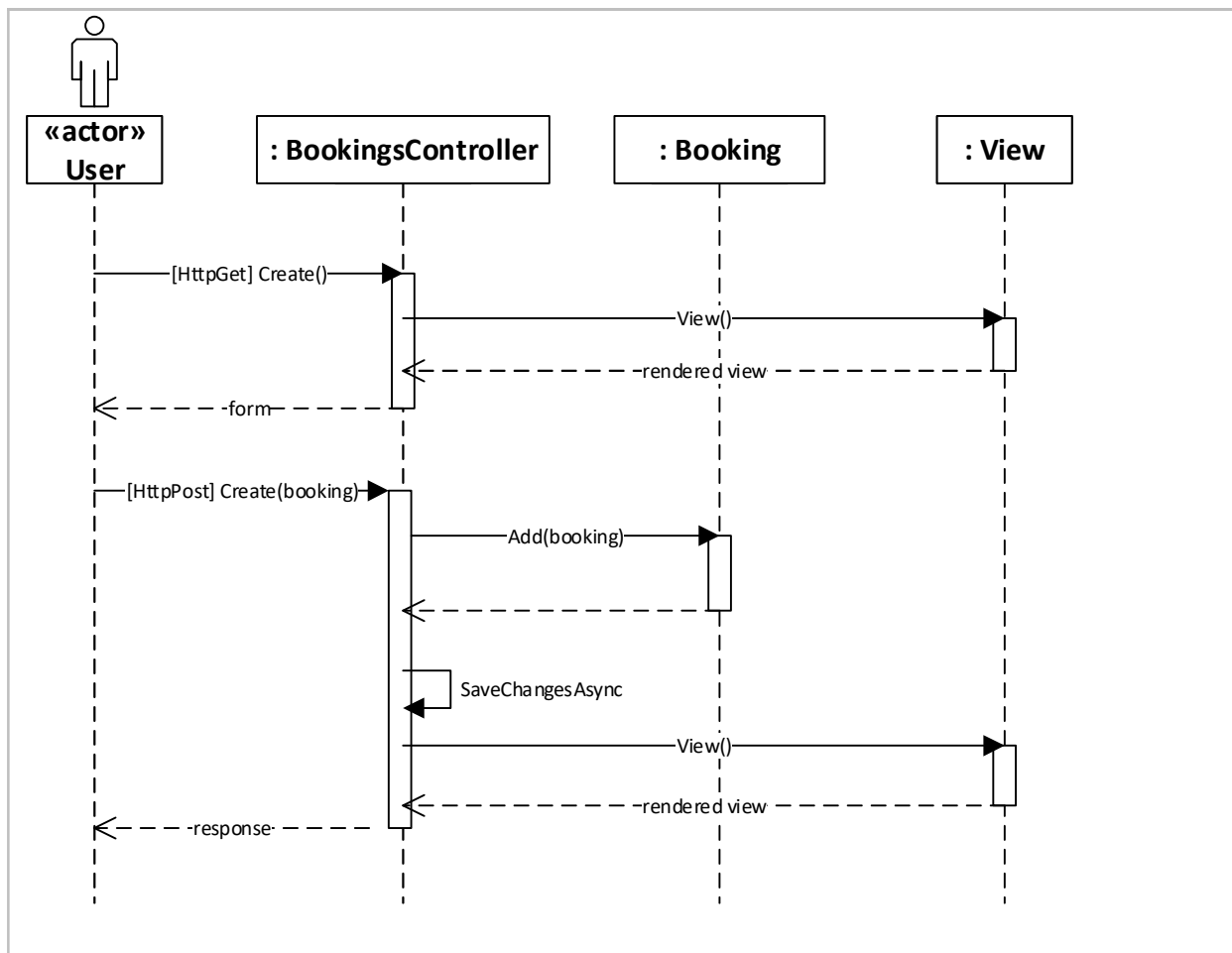


View customers

Add customer

Edit customer

View bookings

Add booking

4 IMPLEMENTATION

4.1 DEVELOPING WEB APP

4.1.1 ASP.NET Core and Entity Framework

The Maesrk Container Management System is developed with ASP.NET Core 2.0 MVC with Entity Framework Core 2.0. ASP.NET Core 2.0 is an open source cross-platform web framework developed by Microsoft. The MVC (Model-View-Controller) architecture is used to accelerate the development involving create, read, update, and delete operations in the web application. Entity Framework Core 2.0 is an object-relational mapping framework, integrated in ASP.NET Core to reduce the effort interacting with relational databases. The project is developed in Visual Studio 2017.

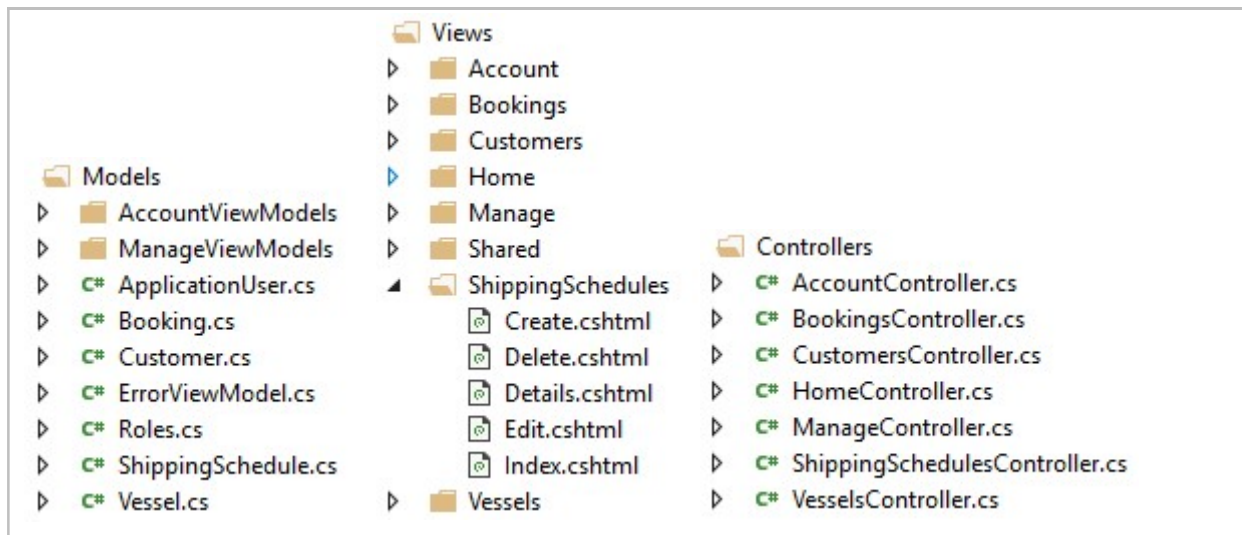


Figure 4-1 Models, views, and controllers

The models are the definitions of the domain objects. The views are templates to render the HTML to the user. The controllers connect the models and the views. Most of the controllers and views can be generated using Visual Studio after defining the model.

The following code segment is the shipping schedule model. It defines the fields in the model, as well as attributes of the fields such as data type, display name, and whether the field is required. The ID field value is automatically generated, with the help of an annotation.

```

public class ShippingSchedule
{
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    7 references | cx, 22 days ago | 1 author, 1 change
    public int Id { get; set; }

    [Display(Name = "Departure time")]
    [DataType(DataType.DateTime)]
    [Required]
    1 reference | cx, 22 days ago | 1 author, 1 change
    public DateTime DepartureTime { get; set; }

    [Display(Name = "Arrival time")]
    [Required]
    [DataType(DataType.DateTime)]
    0 references | cx, 22 days ago | 1 author, 1 change
    public DateTime ArrivalTime { get; set; }

    [Display(Name = "Departure location")]
    [Required]
    2 references | cx, 22 days ago | 1 author, 1 change
    public string DepartureLocation { get; set; }

    [Display(Name = "Arrival location")]
    [Required]
    2 references | cx, 22 days ago | 1 author, 1 change
    public string ArrivalLocation { get; set; }

    [Display(Name = "Vessel")]
    [Required]
    3 references | cx, 22 days ago | 1 author, 1 change
    public int VesselId { get; set; }
    3 references | cx, 22 days ago | 1 author, 1 change
    public virtual Vessel Vessel { get; set; }

    0 references | cx, 15 days ago | 1 author, 2 changes
    public string Description
    {
        get => $"{{String.Format("{0:yyyy-MM-dd HH:mm}", DepartureTime)}} ({{DepartureLocation}} → {{ArrivalLocation}})";
    }
}

```

Figure 4-2 Shipping schedule model

The following code segment ensures that the database tables are created and ensures that the Entity Framework migrations have been executed each time the web application starts:

```

public static void Initialize(IServiceProvider serviceProvider)
{
    var context = serviceProvider.GetRequiredService<ApplicationDbContext>();
    context.Database.EnsureCreated();
    context.Database.Migrate();

    SeedRoles(serviceProvider).Wait();
    SeedUsers(serviceProvider).Wait();

    context.SaveChanges();
}

```

Figure 4-3 Database initialization

4.1.2 Identity management

User identity is managed by identity framework in ASP.NET Core. Log in and account creation is managed by the framework. Identity roles are used to define the pages each user can access. Every user account has a set of identity roles. For example, the following controller action, marked with an annotation, can only be accessed by an account with administrator role:

```
// GET: ShippingSchedules/Create
[Authorize(Roles = Roles.Administrator)]
0 references | cx, 18 days ago | 1 author, 2 changes
public IActionResult Create()
{
    ViewData["VesselId"] = new SelectList(_context.Vessel, "Id", "Name");
    return View();
}
```

Figure 4-4 Role restricted controller action method

4.1.3 Screenshots

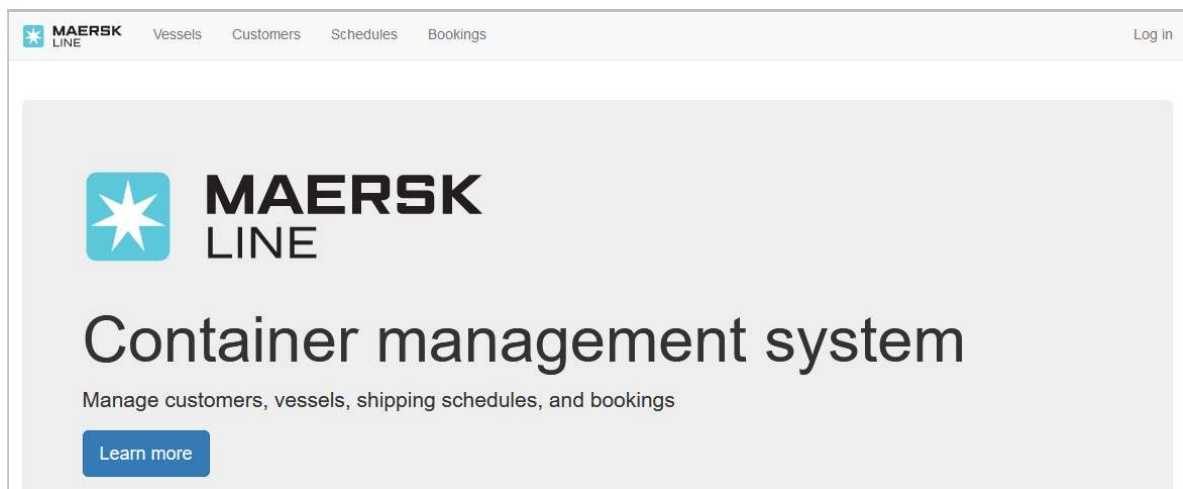
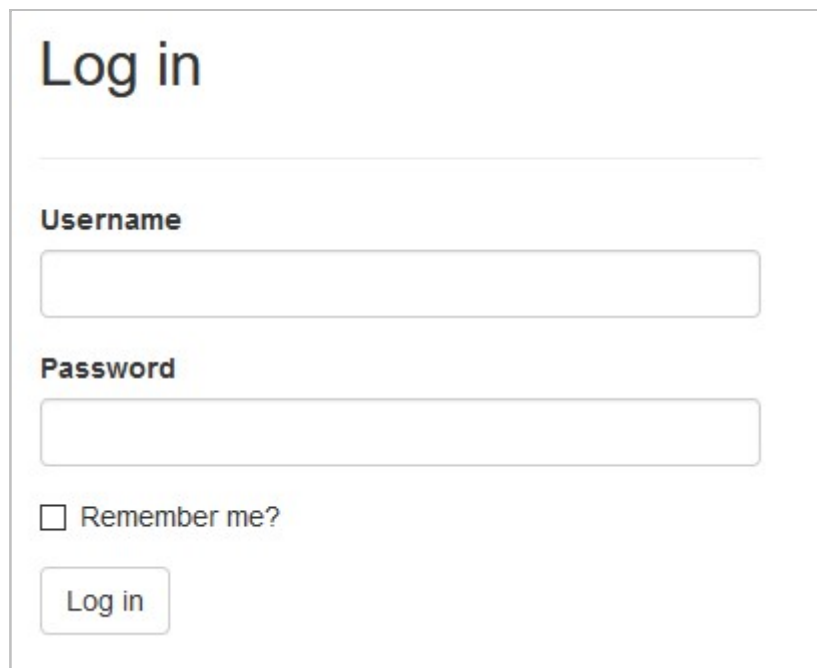


Figure 4-5 Home page

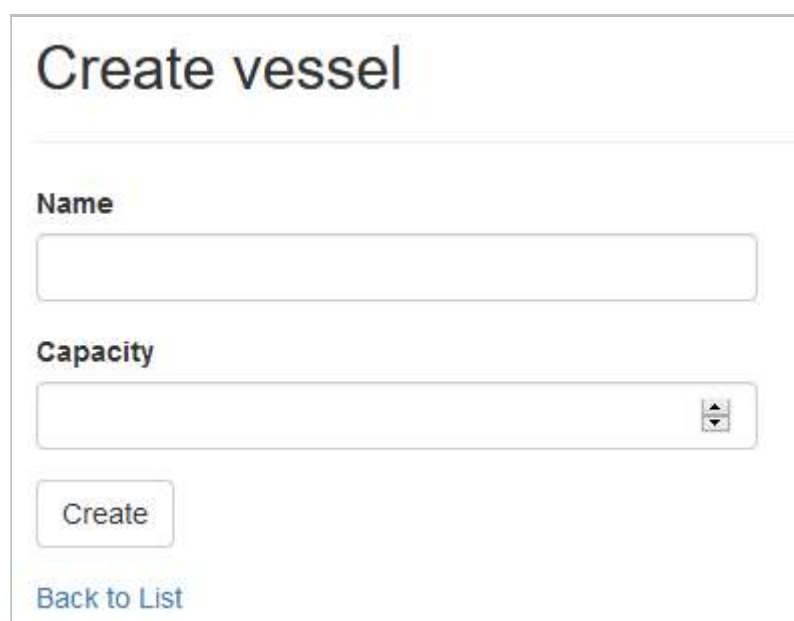
Every page will have a navigation bar on top of the page for users to navigate between different parts of the web application.



A login form titled "Log in" with a horizontal separator line. It contains two text input fields: "Username" and "Password". Below the password field is a checkbox labeled "Remember me?". At the bottom is a "Log in" button.

Figure 4-6 Log in page

Standard log in page to allow the user to access their registered user account. Registration can only be done by an administrator to create more administrator or agent accounts.



A form titled "Create vessel" with a horizontal separator line. It contains two text input fields: "Name" and "Capacity". The "Capacity" field has a small up/down arrow icon on its right side. Below the "Capacity" field is a "Create" button. At the bottom is a "Back to List" link.

Figure 4-7 Vessel create form

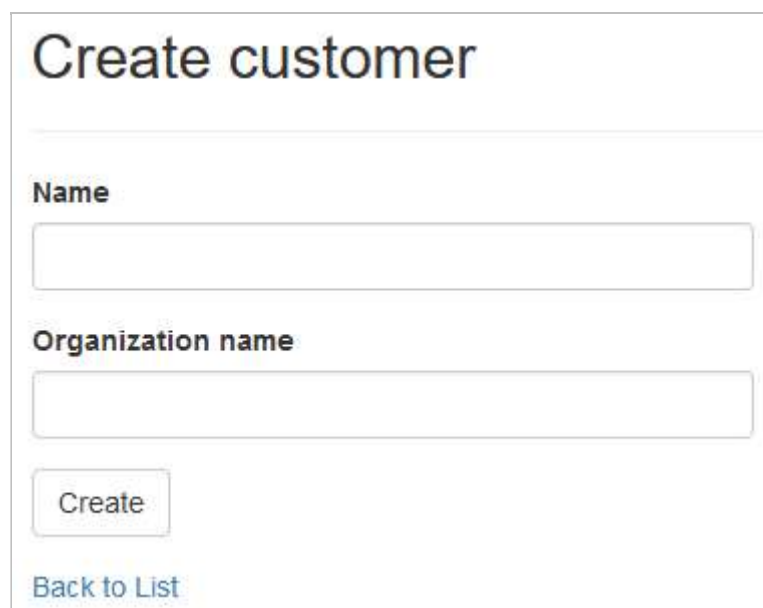
An administrator can add new vessels to the system, specifying the name of the vessel and its capacity.



Vessels		
Create New		
Name	Capacity	
SS Sample	200	Edit Details Delete
SS Some Ship	150	Edit Details Delete

Figure 4-8 Vessel list

Vessels registered in the system will show up in a list. Administrators can also see options to add, edit or delete the vessel entries.



Create customer

Name

Organization name

[Back to List](#)

Figure 4-9 Customer create form

Agents can add their customers to the system, specifying the name of the customer and the organization associated to the customer.

Customers		
Create New		
Name	Organization name	
Alice	Wonderland Inc	Edit Details Delete
Bob	The Builder Pte Ltd	Edit Details Delete

Figure 4-10 Customer list

Registered customers will show up in a list, along with options to edit or delete the customer entries. Agents can only see customers registered under their account.

Create shipping schedule

Departure time

Arrival time

Departure location

Arrival location

Vessel

SS Sample

▼

Create

[Back to List](#)

Figure 4-11 Shipping schedule create form

Administrators can create shipping schedules for the agents to book for their customers. A shipping schedule has departure and arrival date and time, departure and arrival location, as well as the vessel involved in the shipment. The vessel must be selected from one of the registered vessels.

Shipping schedules

Departure location

Arrival location

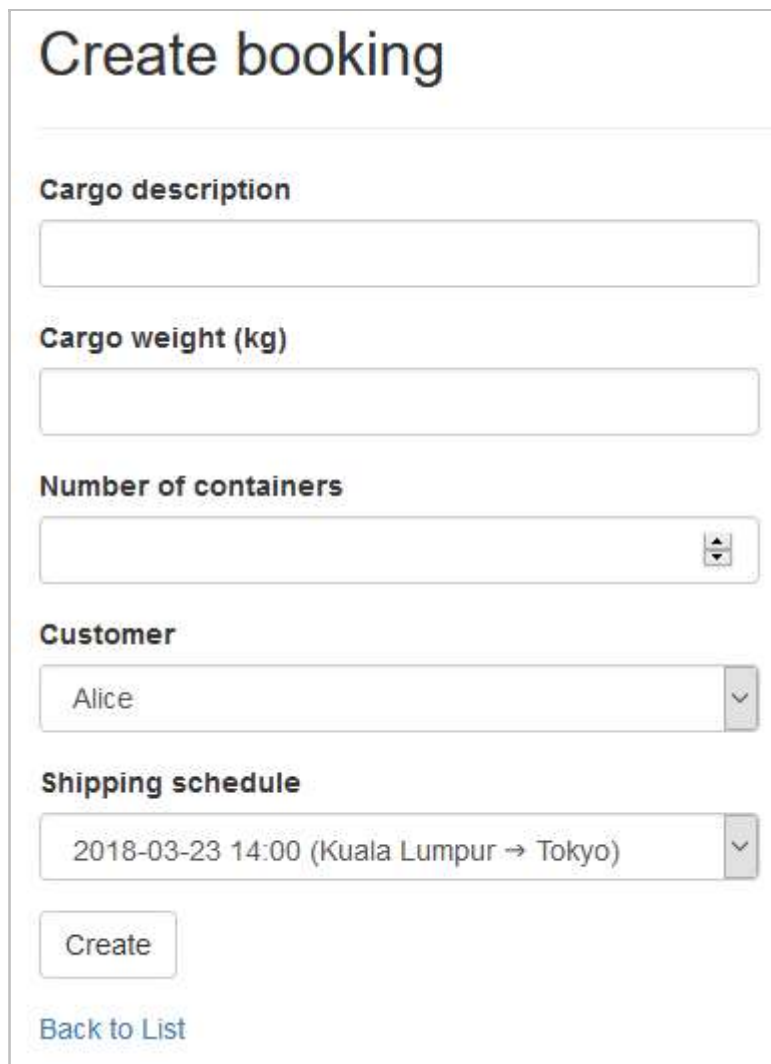
Search

Create New

Departure time	Arrival time	Departure location	Arrival location	Vessel	
3/23/2018 2:00:00 PM	3/23/2018 9:00:00 PM	Kuala Lumpur	Tokyo	SS Sample	Edit Details Delete
3/31/2018 6:00:00 AM	4/1/2018 12:00:00 AM	Hong Kong	Singapore	SS Some Ship	Edit Details Delete

Figure 4-12 Shipping schedule list and search

Created shipping schedules will show up in a list. On top of that, the shipping schedules can be searched and filtered with either the departure location or arrival location, or both.



The form is titled "Create booking" and contains several input fields and a button. The fields are: "Cargo description" (text input), "Cargo weight (kg)" (text input), "Number of containers" (text input with a spinner), "Customer" (dropdown menu showing "Alice"), and "Shipping schedule" (dropdown menu showing "2018-03-23 14:00 (Kuala Lumpur → Tokyo)"). Below the fields is a "Create" button and a "Back to List" link.

Create booking

Cargo description

Cargo weight (kg)

Number of containers

Customer

Alice

Shipping schedule

2018-03-23 14:00 (Kuala Lumpur → Tokyo)

Create

[Back to List](#)

Figure 4-13 Booking create form

Agents can create bookings for their customer. Each booking should contain cargo description, cargo weight, number of containers, customer, as well as the shipping schedule. Customer and shipping schedule must be selected from registered customers and shipping schedules, respectively.

Booking details

Booking confirmed. The cargo is scheduled to be shipped.

Cargo description	Test
Cargo weight (kg)	1
Number of containers	1
Customer	Alice
Shipping schedule	2018-03-23 14:00 (Kuala Lumpur → Tokyo)

[Back to List](#)

Figure 4-14 Booking confirmation

Once a booking has been scheduled, the user will be shown with a confirmation to notify the user that the booking has been confirmed.

Bookings					
Create New					
Cargo description	Customer	Shipping schedule	Cargo weight (kg)	Number of containers	
Wonderland goods	Alice	2018-03-23 14:00 (Kuala Lumpur → Tokyo)	500	5	Details Delete
More wonderland goodies	Alice	2018-03-31 06:00 (Hong Kong → Singapore)	1000	10	Details Delete

Figure 4-15 Booking list

Scheduled bookings will show up in a list, along with the option to remove existing bookings. Agents can only see bookings booked under their account.

4.2 PUBLISHING WEB APP TO CLOUD

4.2.1 App Service

The web application will be published to Microsoft Azure. To prepare for that, an App Service instance is created on Microsoft Azure using the Azure Portal, specifying the app name, subscription, resource group, and the app service plan. Once the App Service instance is ready, the web application can be published to the App Service instance, and the web application will be served over the internet.

The screenshot shows the 'Web App' creation wizard in the Azure Portal. The form is titled 'Web App' with a 'Create' button in the top left corner. The fields are as follows:

- App name:** A text input field containing 'ddac-maersk-sea' with a green checkmark and '.azurewebsites.net' as a suffix.
- Subscription:** A dropdown menu showing 'Azure for Students'.
- Resource Group:** Radio buttons for 'Create new' and 'Use existing' (selected). Below is a dropdown menu showing 'ddac-maersk-afs-sea'.
- OS:** Two buttons, 'Windows' (selected) and 'Linux'.
- App Service plan/Location:** A dropdown menu showing 'freeplan-afs-sea(Southeast Asia)' with a right arrow.
- Application Insights:** A toggle switch with 'On' (selected) and 'Off' buttons.
- Pin to dashboard:** An unchecked checkbox.
- Create:** A large blue button at the bottom left.
- Automation options:** A link at the bottom right.

Figure 4-16 Creating new App Service

The Maersk Line Container Management System is deployed as an App Service instance in the Southeast Asia region. The following are the details of the App Service instance. The web application can be accessed at <https://ddac-maersk-sea.azurewebsites.net>.

Resource group (change) ddac-maersk-afs-sea	URL https://ddac-maersk-sea.azurewebsites.net
Status Running	App Service plan/pricing tier freeplan-afs-sea (Free: 0 Small)
Location Southeast Asia	GitHub Project https://github.com/tp033739/ContainerManagementSystem
Subscription (change) Azure for Students	
Subscription ID 8b480d27-4bbc-4881-ae4f-99f0cbbafa98	

Figure 4-17 App Service details overview

4.2.2 Source control service

The source code of the web application is published on a source control service, GitHub. GitHub is a Git source control service, providing free public source code repositories. The source code of Maersk Line Container Management System is hosted on <https://github.com/tp033739/ContainerManagementSystem>. This repository will be used for deployment of the web application on the App Service.

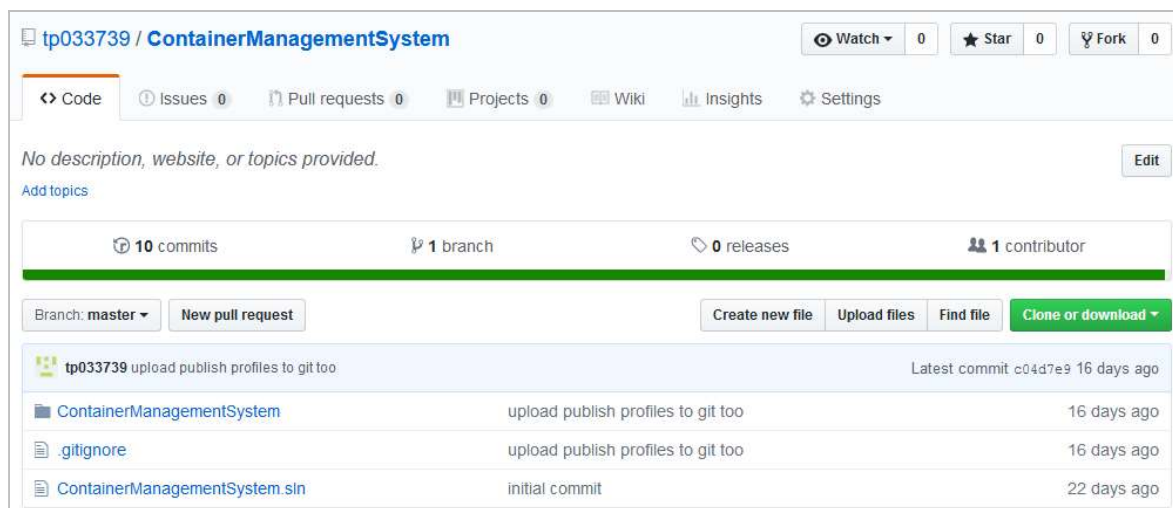


Figure 4-18 GitHub repository

4.2.3 Deployment option

To deploy the web application from the GitHub repository to the App Service, deployment option is configured in the App Service. Under the App Service, GitHub along with other deployment options can be selected as a deployment option. When configuring deployment option, the source (GitHub), project, and branch (the only branch is “master” for this project) must be specified.

Home > ddac-maersk-sea > Deployment option > Choose project

Deployment option ×

Set up deployment option

- * Choose Source > GitHub
- * Authorization > tp033739
- * Choose project > ContainerManagementSystem
- * Choose branch > master
- Performance Test > Not Configured

Click here if you are having trouble seeing your repositories. ↗

OK

Choose project □ ×

ContainerManagementSystem

Figure 4-19 Configuring deployment option

Once authorization is given from GitHub to Azure, the App Service will automatically pull latest source code and build the project, and subsequently publish it. The latest commit will be built and published to the App Service. The deployment options screen will show a list of commits that have been built and published to the App Service.

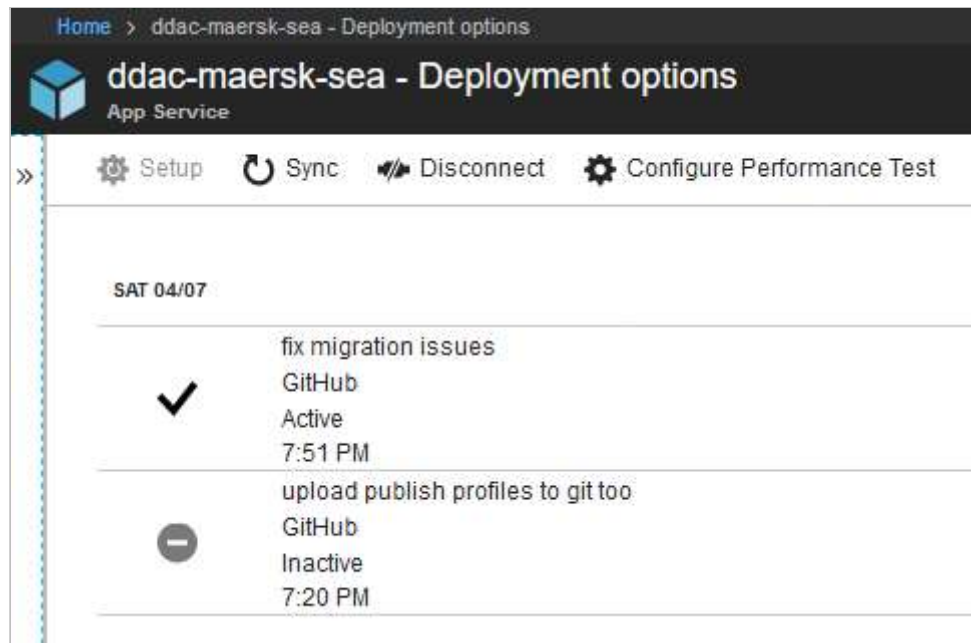
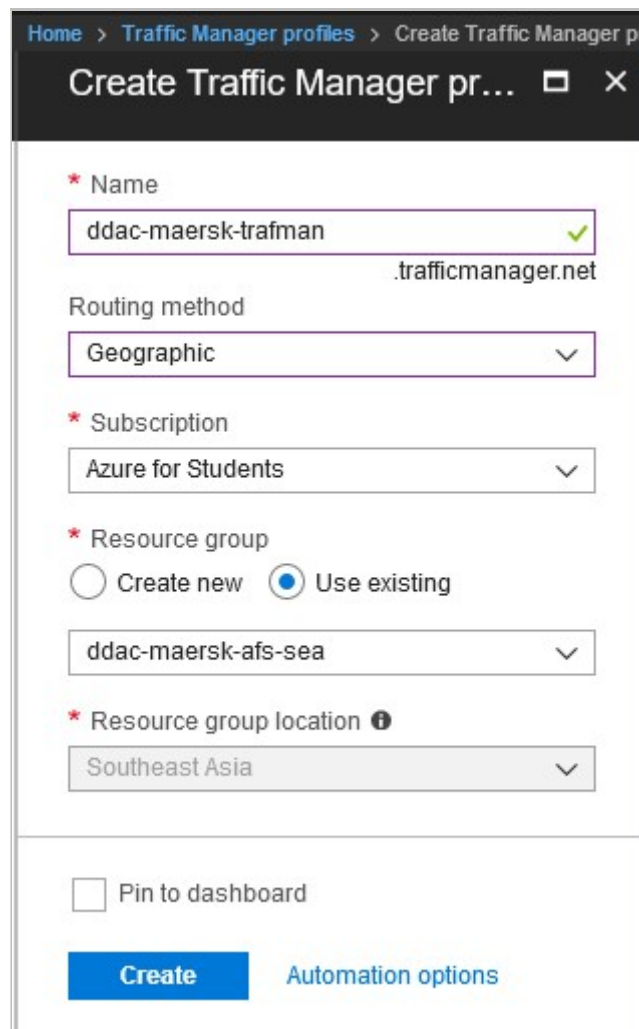


Figure 4-20 Deployment history

4.2.4 Traffic manager

To provide services to multiple geographic regions, multiple instances of App Service can be set up in multiple locations around the globe (Dwivedi, et al., 2017). To correctly direct users to the appropriate instance of App Service, a Traffic Manager profile can be used. Traffic Manager profiles uses DNS to direct the users to the App Service instance assigned for the geographic location. For example, if a user in Southeast Asia visits <https://ddac-maersk-trafman.trafficmanager.net>, the user will be directed to the Southeast Asia instance of the App Service.

To create a Traffic Manager profile, the name, routing method (Geographic, in this case), subscription, and resource group must be specified.



The screenshot shows the 'Create Traffic Manager profile' form in the Azure portal. The breadcrumb navigation at the top reads 'Home > Traffic Manager profiles > Create Traffic Manager pr...'. The form title is 'Create Traffic Manager pr...' with a close button. The form contains the following fields and options:

- Name:** A text input field containing 'ddac-maersk-trafman' with a green checkmark icon. Below the input, the domain '.trafficmanager.net' is displayed.
- Routing method:** A dropdown menu with 'Geographic' selected.
- Subscription:** A dropdown menu with 'Azure for Students' selected.
- Resource group:** Radio buttons for 'Create new' and 'Use existing' (selected). Below, a dropdown menu shows 'ddac-maersk-afs-sea'.
- Resource group location:** A dropdown menu with 'Southeast Asia' selected.
- Pin to dashboard:** An unchecked checkbox.
- Buttons:** A blue 'Create' button and a blue link 'Automation options'.

Figure 4-21 Creating Traffic Manager profile

Once the profile is created, endpoints can be added for each geographic region. When adding an endpoint to a geographic routing Traffic Manager profile, the name, target resource, and regional grouping must be specified. In this case, the “ddac-maersk-sea” instance will be served to users from Asia. Only App Service instances with Standard S1 pricing tier or above can be used in Traffic Manager profiles.

The screenshot shows the 'Add endpoint' dialog box in the Azure portal. The breadcrumb navigation at the top reads 'Home > ddac-maersk-trafman - Endpoints > Add endpoint'. The dialog title is 'Add endpoint' with a sub-label 'ddac-maersk-trafman'. The form contains the following fields and options:

- Type:** A dropdown menu set to 'Azure endpoint'.
- Name:** A required field (marked with a red asterisk) containing 'ddac-maersk-sea', which is validated with a green checkmark.
- Target resource type:** A dropdown menu set to 'App Service'.
- Target resource:** A required field (marked with a red asterisk) containing 'ddac-maersk-sea', with a right-pointing chevron icon.
- Geo-mapping:** A section with explanatory text: 'You may choose to distribute traffic based on specific geographic locations. The same location can't be specified in two endpoints.'
- Regional grouping:** A required field (marked with a red asterisk) containing 'Asia', with a trash icon and a dropdown arrow.
- Country/Region:** An optional dropdown menu with the text 'Choose a Country/Region (optional)'.
- + Add geo-mapping:** A button to add additional geographic mappings.
- Add as disabled:** A checkbox that is currently unchecked.
- OK:** A blue button at the bottom left to confirm the addition.

Figure 4-22 Adding endpoints to Traffic Manager profile

Added endpoints will show up in a list in the overview section of the Traffic Manager profile, enumerating the details of all endpoints in the profile. The Traffic Manager profile must then be activated to take effect.

The screenshot displays the Azure Traffic Manager profile overview for 'ddac-maersk-trafman'. The interface includes a header with the profile name and a star icon. Below the header, there are action buttons: 'Enable profile', 'Disable profile', 'Refresh', 'Move', and 'Delete profile'. The 'Essentials' section provides key details about the profile, including the resource group, status, subscription, DNS name, monitor status, and routing method. A search bar for endpoints is located below the essentials. The endpoints table lists two endpoints: 'ddac-maersk-sea' and 'ddac-maersk-eu', both with an 'Enabled' status and 'Stopped' monitor status, identified as 'Azure endpoint' type.

NAME	STATUS	MONITOR STATUS	TYPE
ddac-maersk-sea	Enabled	Stopped	Azure endpoint
ddac-maersk-eu	Enabled	Stopped	Azure endpoint

Figure 4-23 Traffic Manager profile overview and list of endpoints

4.2.5 Monitoring performance

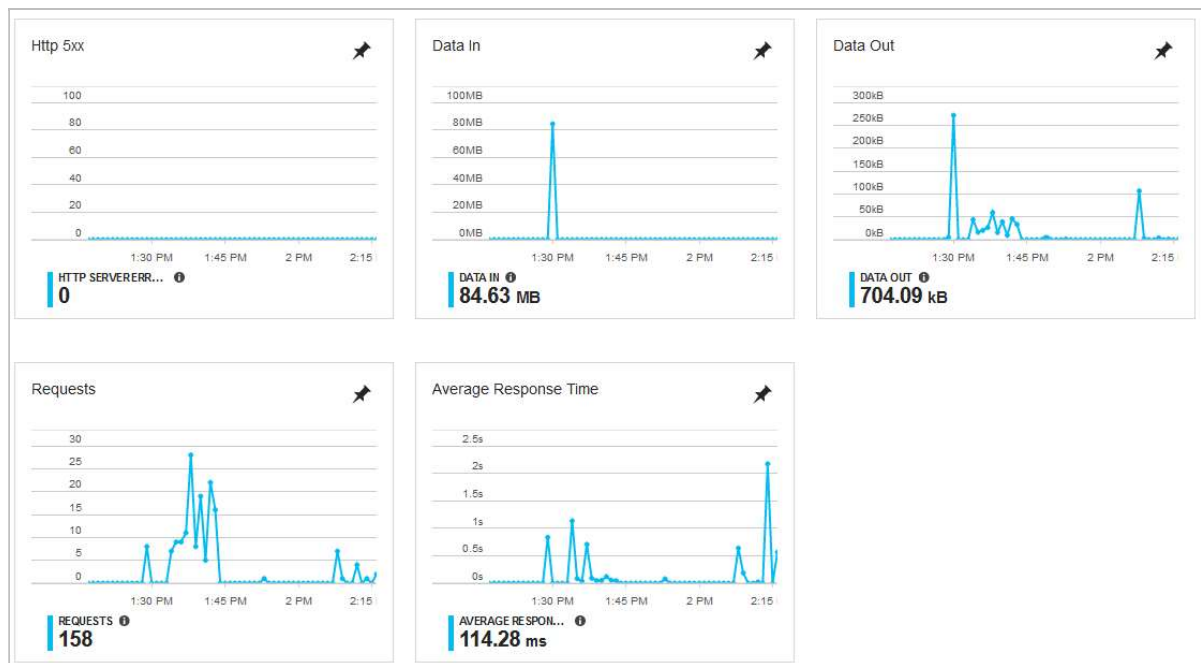


Figure 4-24 Performance statistic graphs

The App Service overview page displays the various statistics of the App Service performance in real time. Statistics include number of server errors (HTTP 5xx), incoming traffic, outgoing traffic, number of requests, as well as average response time. These statistics can help the developer to identify performance issues and scale App Service accordingly. Spikes in average response time and number of requests may indicate the server is under heavy load, and may require scaling to optimize the performance of the App Service.

4.3 APPLICATION SCALING

4.3.1 Scaling up

During peak seasons, the amount of traffic will increase significantly due to the increase of customers and bookings. To ensure the web application can accommodate the increase in traffic, the App Service must be scaled up. Scaling up, or vertical scaling, is a type of scaling that increases the hardware resources allocated for a single node – for example, increase in CPU cores, increase in RAM memory, and increase in storage space. (Beaumont, 2014)

Microsoft Azure offers many pricing tiers for developers to pick from. Each pricing tier has different number of CPU cores, RAM memory, storage space, as well as other miscellaneous features such as SSL support, horizontal scalability, backup, deployment slots, as well as availability of traffic manager. The Standard pricing tier comes with three variants – S1, S2, and S3, offers more resources one after another in that sequence.

During peak seasons, the web application can be scaled to S2 Standard or S3 Standard depending on the number of concurrent users. More CPU cores and RAM memory will improve the performance of the web application and handle more concurrent users.

S1 Standard		S2 Standard		S3 Standard	
1	Core	2	Core	4	Core
1.75	GB RAM	3.5	GB RAM	7	GB RAM
	50 GB Storage		50 GB Storage		50 GB Storage
	Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support
	Up to 10 instance(s) Auto scale		Up to 10 instance(s) Auto scale		Up to 10 instance(s) Auto scale
	Daily Backup		Daily Backup		Daily Backup
	5 slots Web app staging		5 slots Web app staging		5 slots Web app staging
	Traffic Manager Geo availability		Traffic Manager Geo availability		Traffic Manager Geo availability
312.48 MYR/MONTH (ESTIMATED)		624.96 MYR/MONTH (ESTIMATED)		1,249.92 MYR/MONTH (ESTIMATED)	

Figure 4-25 Standard pricing tiers for app service plan

4.3.2 Scaling out

Other than scaling up, scaling out can also be performed to prepare for the peak season. Scaling out, or horizontal scaling, is a type of scaling that increases the number of nodes, distributing the load across the nodes (Beaumont, 2014). Increase in number of nodes will help to handle more concurrent users, as they are technically separate instances of the web application.

Microsoft Azure offers to automatically horizontal scale based when certain conditions are met. Metrics available for defining condition includes CPU usage, RAM usage, and so on. For example, by selecting “Scale based on a metric” and “CpuPercentage > 70”, when the CPU usage exceeds 70% in one instance, the App Service will automatically scale out to create new instance to distribute the load. Maximum number of instances can be defined to prevent incurring charges for unexpected events such as flooding.

Save Discard Disable autoscale Refresh

Configure Run history JSON Notify

* Autoscale setting name ✓

Resource group ▼

Default Auto created scale condition ✎

Delete warning ⓘ The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode ☒ Scale based on a metric ☐ Scale to a specific instance count

ⓘ It is recommended to have at least one scale in rule

Rules

Scale out			
When	freeplan-afs-sea	(Average) CpuPercentage > 70	Increase instance count by 1

+ Add a rule

Instance limits

Minimum ⓘ	Maximum ⓘ	Default ⓘ
<input type="text" value="1"/> ✓	<input type="text" value="5"/> ✓	<input type="text" value="1"/> ✓

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

+ Add a scale condition

Figure 4-26 Auto scale configuration

4.4 MANAGED DATABASE

4.4.1 SQL database

Microsoft Azure offers managed databases such as SQL Server (Microsoft, 2018) and Cosmos DB (Microsoft, 2018). Maersk Line Container Management System consumes a relational database, that is SQL server on Microsoft Azure. When creating an SQL database, the database name, subscription, resource group, server name, admin credentials, and server location must be specified.

The screenshot displays the Azure portal interface for creating a new SQL server and database. The interface is split into three main sections:

- SQL Database:** Contains configuration options for the database itself. Fields include:
 - Database name:** ddac-maersk-sea (with a green checkmark).
 - Subscription:** Azure for Students (dropdown).
 - Resource group:** ddac-maersk-afs-sea (dropdown).
 - Select source:** Blank database (dropdown).
 - Server:** A link to 'Configure required settings'.
 - Want to use SQL elastic pool?:** Not now (selected).
 - Pricing tier:** A link to 'Configure required settings'.
 - Collation:** SQL_Latin1_General_CP1_CI_AS.
- Server:** Shows a 'Create a new server' button and a message 'No servers found'.
- New server:** Contains configuration options for the server instance. Fields include:
 - Server name:** ddac-maersk-sea (with a green checkmark).
 - Server admin login:** maerskdbadmin (with a green checkmark).
 - Password:** Masked with dots (with a green checkmark).
 - Confirm password:** Masked with dots (with a green checkmark).
 - Location:** Southeast Asia (dropdown).
 - Allow azure services to access server:** Checked checkbox.

At the bottom of the 'New server' pane is a 'Select' button. The 'SQL Database' pane also has a 'Create' button and a link to 'Automation options'.

Figure 4-27 Creating SQL server and database

Once the database is ready, the connection string can be retrieved and applied to the App Service so that the web application can consume the cloud database.

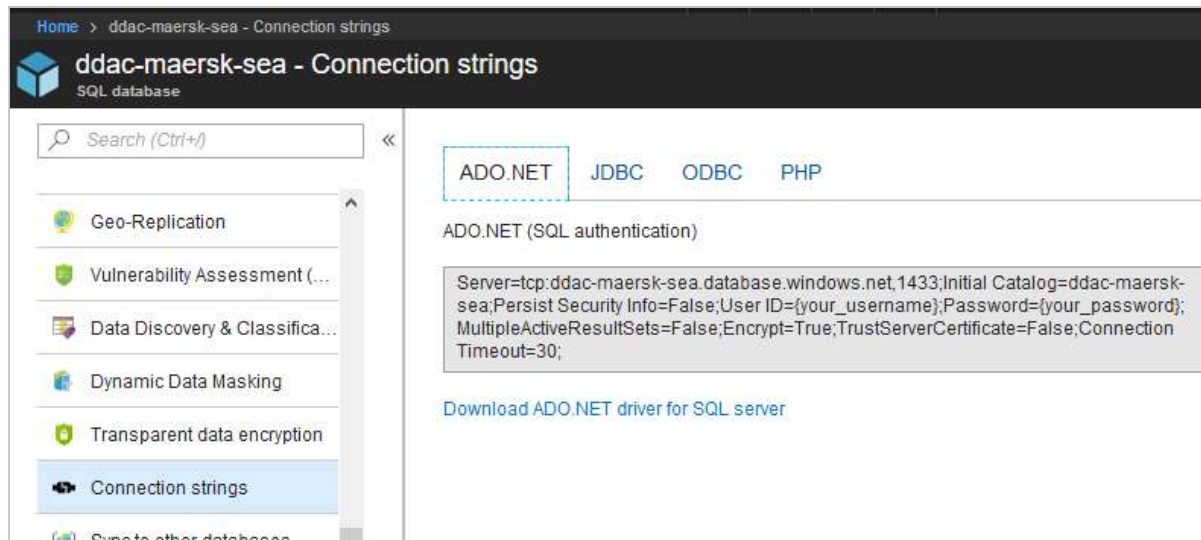


Figure 4-28 Retrieving connection string

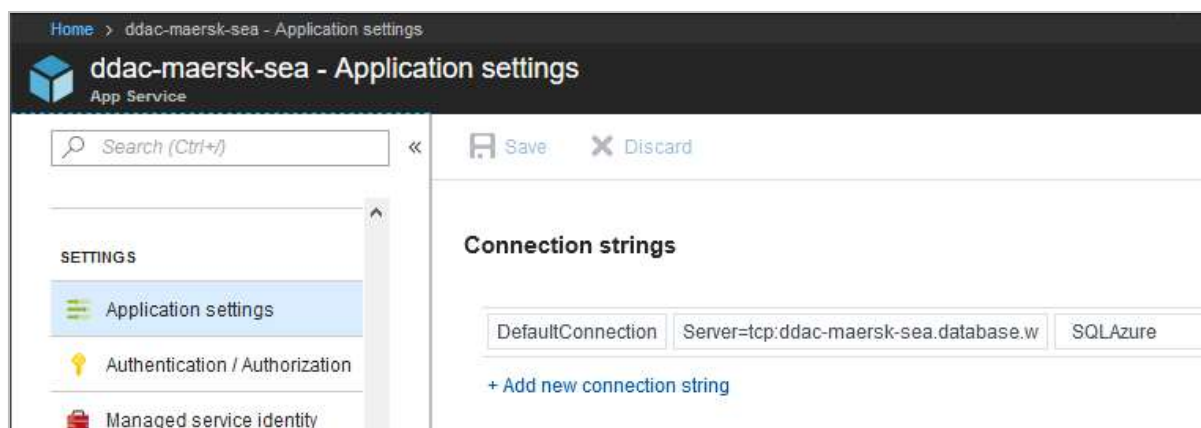


Figure 4-29 Applying the connection string on App Service

4.4.2 Database scaling

Similar to App Service, an SQL database can also be scaled if needed. Microsoft Azure offers three tiers of scaling for SQL database – Basic, Standard, and Premium, each with different range of DTUs (data transaction units), which is a bundled measure of computing power, storage, and input-output resources. The pricing of each tier is also determined by the number of DTUs selected. The more the number of DTUs, the better the performance of the database transactions, the higher the price. The number of DTUs can be increased or decreased as necessary. (Nolting, 2013)

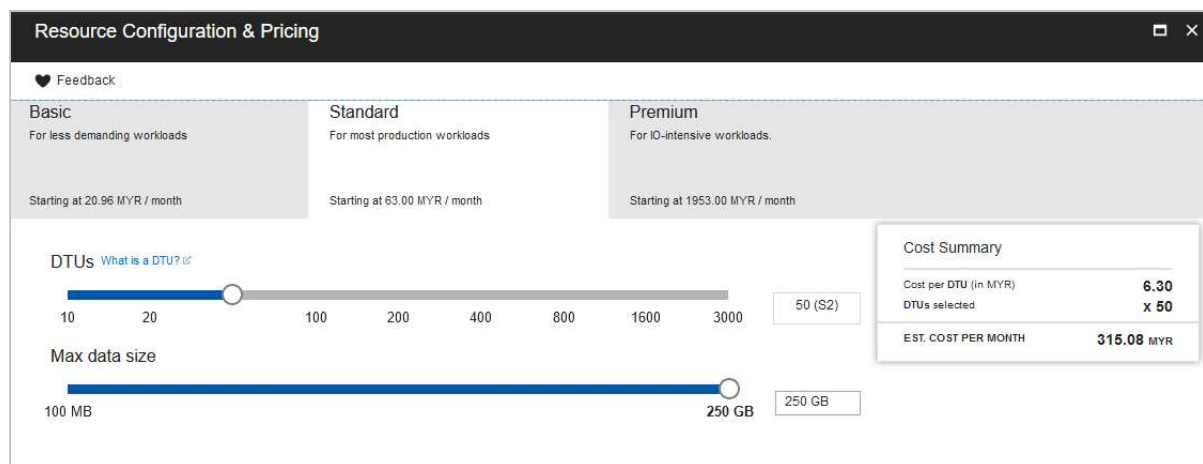


Figure 4-30 SQL database scaling

4.4.3 Geo replication

To improve the database performance when deploying across multiple geographic location, the database could be replicated to multiple secondary databases (Microsoft, 2018). These secondary databases have lower latency for their respective geographic region, compared to directing all database transactions all over the world towards a single geographic region. To geo replicate a database, the target region, new server name, admin credentials, and pricing tier must be specified.

Home > ddac-maersk-sea - Geo-Replication > Create secondary > Server > New server

Create secondary

Create geo-replicated secondaries to protect against prolonged datacenter outages. Secondaries have price implications. [Learn more](#)

Region
West Europe

Database name
ddac-maersk-sea

* Secondary type
Readable

* Target server
Configure required settings

Elastic database pool
None

* Pricing tier
Configure required settings

☐ Pin to dashboard

OK

Server

+ Create a new server

No servers found

New server

* Server name
ddac-maersk-eu

* Server admin login
ddacmaerskdbuser

* Password
.....

* Confirm password
.....

* Location
West Europe

☒ Allow azure services to access server

Select

Figure 4-31 Setting up geo replication

Once the replication is successful, the primary and secondary databases will constantly sync to provide optimum database performance across different geographic locations. Secondary databases are readable only, and write-transactions are directed back to the primary database. The following shows the primary database in Southeast Asia, is replicated to a secondary database in West Europe.

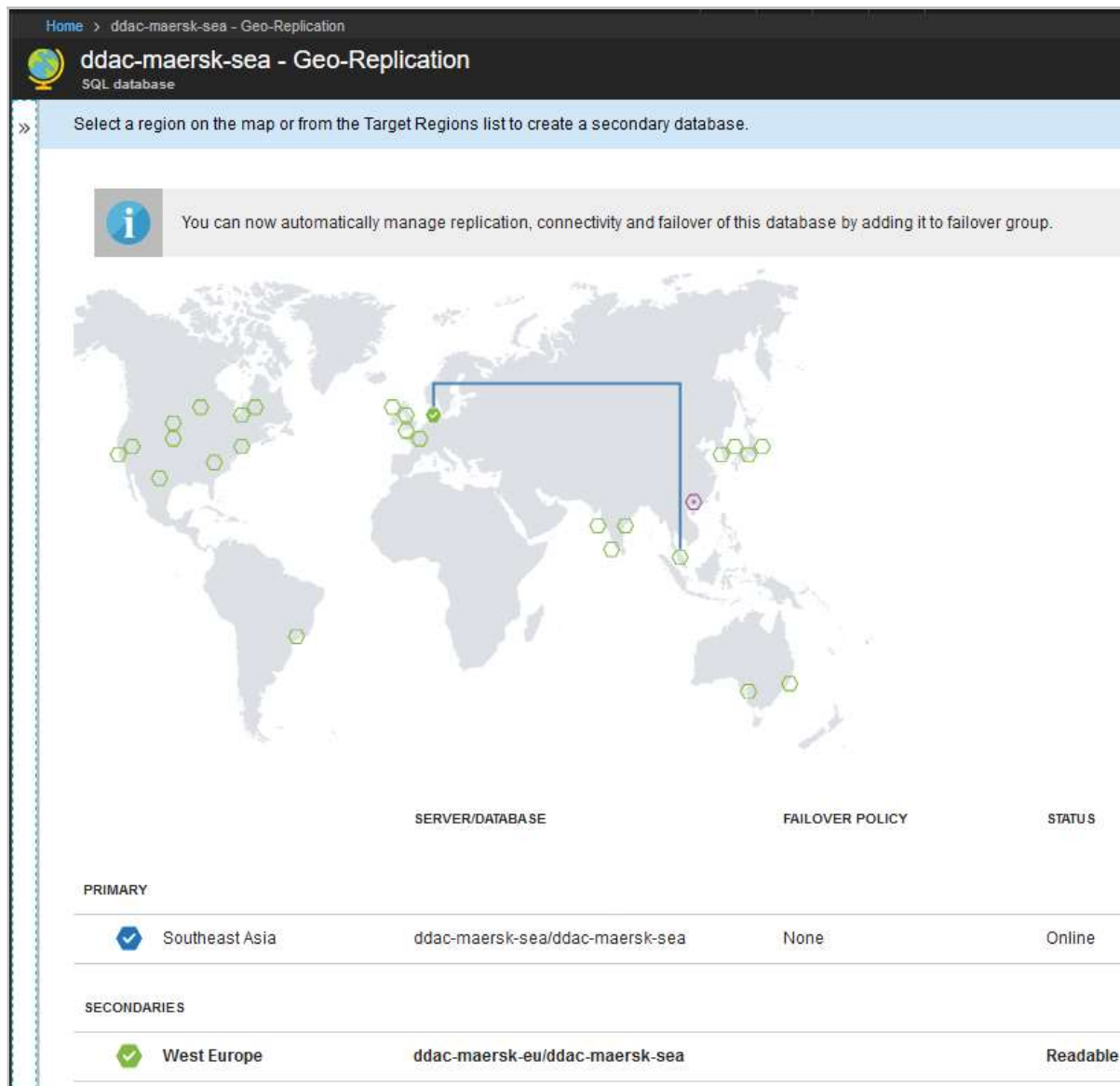


Figure 4-32 Geo-replicated database

5 TESTING

5.1 UNIT TESTING

Test case	Test procedure	Expected result	Actual result
1.1 Log in	<ol style="list-style-type: none"> 1. Visit the homepage 2. Click “Log in” on navigation bar 3. Enter valid username and password 4. Click “Log in” button or press Enter 	User is logged in successfully and redirected to home page	User is logged in successfully and redirected to home page
1.2 Invalid username	<ol style="list-style-type: none"> 1. Visit the home page 2. Click “Log in” on navigation bar 3. Enter invalid username 4. Click “Log in” button or press Enter 	User is not logged in. Error displaying “invalid login”	User is not logged in. Error displaying “invalid login”
1.3 Invalid password	<ol style="list-style-type: none"> 1. Visit the home page 2. Click “Log in” on navigation bar 3. Enter invalid password 4. Click “Log in” button or press Enter 	User is not logged in. Error displaying “invalid login”	User is not logged in. Error displaying “invalid login”
1.4 Blank credentials	<ol style="list-style-type: none"> 1. Visit the home page 2. Click “Log in” on navigation bar 3. Click “Log in” button without filling the form 	User is not logged in. Error displaying “username field and password field is required”	User is not logged in. Error displaying “username field and password field is required”
1.5 Log out	<ol style="list-style-type: none"> 1. Log in 2. Click “Log out” on navigation bar 	User is no longer logged in	User is no longer logged in

Test case	Test procedure	Expected result	Actual result
2.1 Add vessel	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Vessels” on navigation bar 3. Click “Create new” 4. Fill in form with name and capacity of vessel 5. Click “Create” 	Vessel is added with the correct information	Vessel is added with the correct information
2.2 Add vessel without name	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Vessels” on navigation bar 3. Click “Create new” 4. Fill in the form except the name field 5. Click “Create” 	Vessel is not added. Error displaying “Name field is required”	Vessel is not added. Error displaying “Name field is required”
2.3 Add vessel without capacity	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Vessels” on navigation bar 3. Click “Create new” 4. Fill in the form except the capacity field 5. Click “Create” 	Vessel is not added. Error displaying “Capacity field is required”	Vessel is not added. Error displaying “Capacity field is required”
2.4 Delete vessel	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Vessels” on navigation bar 3. Click “Delete” for one of the entries 4. Click “Delete” 	Vessel is removed	Vessel is removed
2.5 Edit vessel	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Vessels” on navigation bar 3. Click “Edit” for one of the entries 4. Make changes to the form 5. Click “Save” 	Vessel information is updated	Vessel information is updated
2.6 No create or edit for agent	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Vessels” on navigation bar 	No option for “Create new” or “Edit”	No option for “Create new” or “Edit”

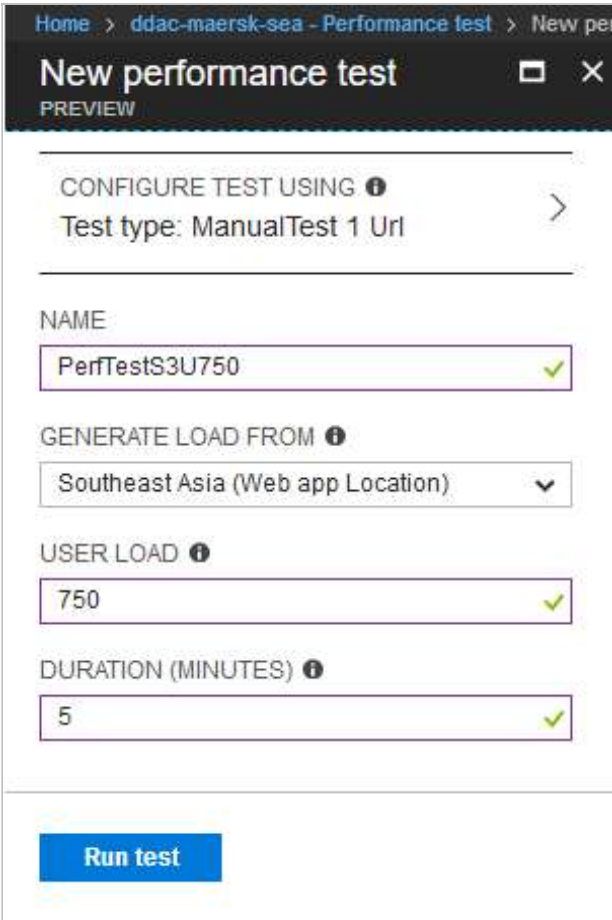
Test case	Test procedure	Expected result	Actual result
3.1 Add customer	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Customers” on navigation bar 3. Click “Create new” 4. Fill in form with customer details 5. Click “Create” 	Customer is created	Customer is created
3.2 Add customer missing details	<ol style="list-style-type: none"> 1. Execute test case 3.1 until step 3 2. Click “Create” without filling the form 	Customer is not created. Display errors next to empty fields	Customer is not created. Display errors next to empty fields
3.3 Delete customer	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Customers” on navigation bar 3. Click “Delete” for one of the entries 4. Click “Delete” 	Customer is removed	Customer is removed
3.4 Edit customer	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Customers” on navigation bar 3. Click “Edit” for one of the entries 4. Make changes to the form 5. Click “Save” 	Customer information is updated	Customer information is updated
3.5 Only show customers created by agent	<ol style="list-style-type: none"> 1. Log in as agent A 2. Create a customer like in test case 3.1 3. Log out and log in as agent B 4. Click “Customers” on navigation bar 	Customer created by Agent A is not visible to Agent B	Customer created by Agent A is not visible to Agent B
3.6 Show all customers to admin	<ol style="list-style-type: none"> 1. Log in as agent 2. Create a customer like in test case 3.1 3. Log out and log in as admin 4. Click “Customers” on navigation bar 	Customer created by agent is visible to admin	Customer created by agent is visible to admin

Test case	Test procedure	Expected result	Actual result
4.1 Add shipping schedule	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Schedules” on navigation bar 3. Click “Create new” 4. Fill in form with schedule information 5. Click “Create” 	Shipping schedule is created	Shipping schedule is created
4.2 Add shipping schedule without all fields	<ol style="list-style-type: none"> 1. Execute test case 4.1 until step 3 2. Click “Create” without filling the form 	Shipping schedule is not created. Display errors next to empty fields	Shipping schedule is not created. Display errors next to empty fields
4.3 Search shipping schedule	<ol style="list-style-type: none"> 1. Create shipping schedule from “Kuala Lumpur” to “Tokyo” (see test case 4.1) 2. Create another shipping schedule from “Hong Kong” to “Singapore” 3. In the search form, fill in “Kuala Lumpur” for departure location 4. Click “Search” 	Only the first shipping schedule (“Kuala Lumpur” to “Singapore”) is displayed	Only the first shipping schedule (“Kuala Lumpur” to “Singapore”) is displayed
4.4 Delete shipping schedule	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Schedules” on navigation bar 3. Click “Delete” for one of the entries 4. Click “Delete” 	Shipping schedule is removed	Shipping schedule is removed
4.5 Edit shipping schedule	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Schedules” on navigation bar 3. Click “Edit” for one of the entries 4. Make changes to the form 5. Click “Save” 	Shipping schedule information is updated	Shipping schedule information is updated
4.6 No create or edit for agent	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Schedules” on navigation bar 	No option for “Create new” or “Edit”	No option for “Create new” or “Edit”

Test case	Test procedure	Expected result	Actual result
5.1 Add booking	<ol style="list-style-type: none"> 1. Log in as agent 2. Click “Bookings” on navigation bar 3. Click “Create new” 4. Fill in form with booking details 5. Click “Create” 	Booking is added and the details are displayed.	Booking is added and the details are displayed
5.2 Add booking without all fields	<ol style="list-style-type: none"> 1. Execute test case 5.1 until step 3 2. Click “Create” without filling the form 	Booking is not added. Display errors next to each empty field	Booking is not added. Display errors next to each empty field
5.3 Delete booking	<ol style="list-style-type: none"> 1. Log in as administrator 2. Click “Bookings” on navigation bar 3. Click “Delete” for one of the entries 4. Click “Delete” 	Booking is removed	Booking is removed
5.4 Only show bookings created by agent	<ol style="list-style-type: none"> 1. Log in as agent A 2. Create a booking like in test case 3.1 3. Log out and log in as agent B 4. Click “Bookings” on navigation bar 	Booking created by Agent A is not visible to Agent B	Booking created by Agent A is not visible to Agent B
3.6 Show all bookings to admin	<ol style="list-style-type: none"> 1. Log in as agent 2. Create a booking like in test case 3.1 3. Log out and log in as admin 4. Click “Bookings” on navigation bar 	Booking created by agent is visible to admin	Booking created by agent is visible to admin

5.2 PERFORMANCE TESTING

Performance testing is required to assess how well the App Service is running on Microsoft Azure by measuring metrics such as failure rate and response time (Microsoft, 2018). To run a performance test on a specific App Service, it must be created and queued for execution. When creating a performance test, name of the performance test, location of the load generation, number of concurrent users, and duration must be specified. Once created, it will be queued for execution. It will take 15 minutes to queue and allocate resources to run the performance test. Microsoft Azure will generate traffic to the App Service with the number of concurrent users specified.



Home > ddac-maersk-sea - Performance test > New per

New performance test

PREVIEW

CONFIGURE TEST USING ⓘ

Test type: ManualTest 1 Url >

NAME

PerfTestS3U750 ✓

GENERATE LOAD FROM ⓘ

Southeast Asia (Web app Location) ▼

USER LOAD ⓘ

750 ✓

DURATION (MINUTES) ⓘ

5 ✓

Run test

Figure 5-1 Creating performance test

While the test is running, the progress of the performance test can be observed. The report displays the number of requests generated, how many of those failed, the average response time of the App Service, average number of requests sent per second, as well as the CPU time and memory usage of the App Service during the performance test.

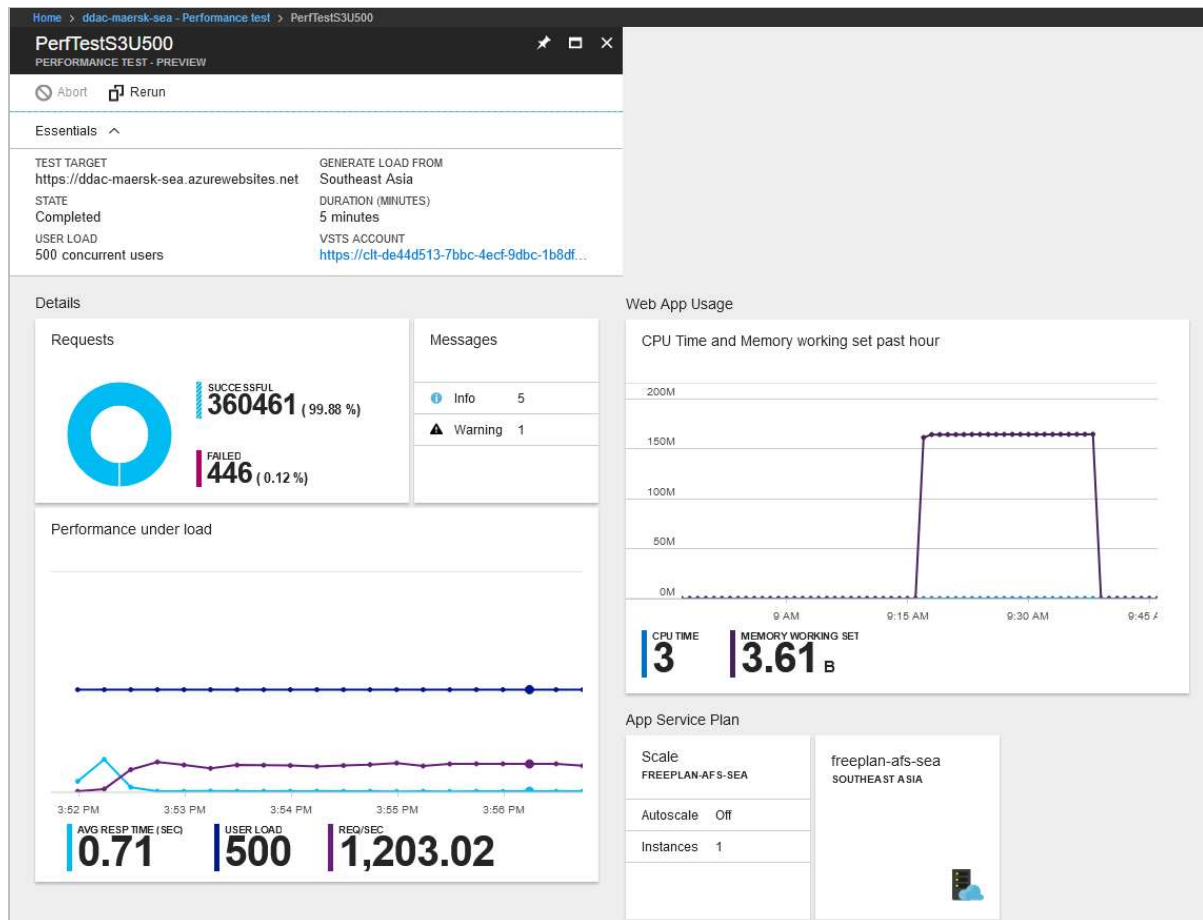


Figure 5-2 Performance test report

A series of performance tests are run to check the performance of the Maersk Line Container Management System when using different pricing tiers with different number of concurrent users. The tested pricing tiers are Standard S1, Standard S2, and Standard S3. For each pricing tier, the App Service is tested with 250, 500, 750, and 1000 concurrent users. The tests will run for 5 minutes each. The average response time and request failure rate is recorded for analysis.

Pricing tier	250 concurrent users	500 concurrent users	750 concurrent users	1000 concurrent users
S1	0.95s 0.00% failed	1.92s 0.00% failed	2.85s 0.04% failed	3.63s 0.00% failed
S2	0.27s 0.00% failed	0.95s 0.00% failed	1.36s 0.00% failed	1.89s 0.00% failed
S3	0.09s 0.00% failed	0.71s 0.12% failed	0.88s 0.00% failed	1.39s 0.04% failed

Table 5-1 Performance against number of concurrent user for each pricing tiers

Based on the test results, Microsoft Azure is considerably reliable even under heavy load, seeing that the failure rates are below 1% for all pricing tiers tested. However, the average response time for each pricing tier has significant differences. The Standard S1 takes more than 3.63 seconds to respond when under load of 1000 concurrent users, while the Standard S3 takes only 1.39 seconds to handle the same load. This proves that the extra CPU cores and memory that comes with Standard S3 is indeed able to handle more load. Scaling up to Standard S3 or higher can help to improve response time during peak seasons when traffic is more heavy.

6 CONCLUSION

The Maersk Line Container Management System is successfully developed and deployed on Microsoft Azure. The system is able to handle the booking process from searching shipping schedule to confirming the bookings. Requirements of the system have been achieved. Database is managed by cloud and has geo-replication to reduce latency for each region. On top of that, the system is able to scale itself during the peak seasons by detecting the CPU usage. Unit tests are performed and has verified that the system is working correctly. Performance tests are performed and has verified that the system is able to handle traffic under heavy load at different pricing tiers.

Developing a web application and deploying it on the cloud was challenging. I have acquired knowledge and deeper understanding of cloud development. I strongly believe that this experience will be useful in my career as a software engineer, as cloud-powered applications will be the norm for future software services.

REFERENCES

Beaumont, D., 2014. *How to explain vertical and horizontal scaling in the cloud*. [Online]

Available at: <https://www.ibm.com/blogs/cloud-computing/2014/04/09/explain-vertical-horizontal-scaling-cloud/>

[Accessed 10 April 2018].

Collier, M. & Shahan, R., 2016. *Microsoft Azure Essentials: Fundamentals of Azure*. 2nd ed. s.l.:Microsoft Press.

Dwivedi, K., Madureira, J. & Tuliani, J., 2017. *Overview of Traffic Manager*. [Online]

Available at: <https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview>

[Accessed 10 April 2018].

Microsoft, 2018. *Azure Cosmos DB - Globally Distributed Database Service*. [Online]

Available at: <https://azure.microsoft.com/en-gb/services/cosmos-db/>

[Accessed 10 April 2018].

Microsoft, 2018. *Configure active geo-replication for Azure SQL Database in the Azure portal and initiate failover*. [Online]

Available at: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-geo-replication-portal>

[Accessed 10 April 2018].

Microsoft, 2018. *Load test with the Azure portal*. [Online]

Available at: <https://docs.microsoft.com/en-us/vsts/load-test/app-service-web-app-performance-test?view=vsts>

[Accessed 10 April 2018].

Microsoft, 2018. *SQL Database - Cloud Database as a Service*. [Online]

Available at: <https://azure.microsoft.com/en-us/services/sql-database/>

[Accessed 10 April 2018].

Nolting, D., 2013. *5 Best Practices for Database Scaling*. [Online]

Available at: <https://www.bluelock.com/blog/5-best-practices-database-scaling/>

[Accessed 10 April 2018].

APPENDIX A SOURCE CODE AND VIDEO

Source code

<https://github.com/tp033739/ContainerManagementSystem>

Presentation video

<https://web.microsoftstream.com/video/6d5bfd1b-63e3-4245-8a02-18684b5c664d>

Web app

<https://ddac-maersk-sea.azurewebsites.net>

<http://ddac-maersk-trafman.trafficmanager.net>