INDIVIDUAL ASSIGNMENT

CT071-3-3-DDAC

DESIGNING DEVELOPING APPLICATIONS ON THE CLOUD

NP3F1801IT

HAND OUT DATE: 26 July 2018

HAND IN DATE: 30 October 2018

WEIGHTAGE: 100%

Arbindra Sutihar (NP000021)

INSTRUCTIONS TO CANDIDATES

1. **Submit your assignment to the administration counter.**
2. **Students are advised to underpin their answers with the use of references (sites using the Harvard Name System of Referencing)**
3. **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld**
4. **Cases of plagiarism will be penalized**
5. **The assignment should be bound in an appropriate style (Comb Bound or Stapled)**
6. **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope/ CD cover and attached to the hardcopy.**
7. **You must obtain 50% overall to pass this module.**

ACKNOWLEDGEMENT

The Basic reason to do this Designing Developing Applications on the Cloud is to satisfy the central necessity of the given task of BSc.IT to achieve the end goal to me along with the rules and guideline that I have been provided from the Asia pacific University (APU). I am very grateful to my lecturer **Mr. Sukanth Kumar Sahu** for helping me in various area and enhancing me with the necessary tools and guide line to complete my project in time and achieve the end goal. He has also provided me the necessary information and contacted people for me to achieve my goal and lead me to better path.

Besides, this I would like to thank my friends and the Administrative people for providing necessary information on this Designing Developing Applications on the Cloud.

Most sincerely I am grateful to the LBEF who has provided me the data and information that is very crucial to complete the investigation report as per the demand of Asia Pacific University (APU).

Sincerely,

Arbindra Sutihar

NP000021

Table of Contents

List of Table

List of Figure

1. INTRODUCTION

As indicated by the given task, Maersk Line is the worldwide holder division and the biggest working 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 might we bolster the general business procedure, and furthermore from an IT point of view," says Soeren Lorenzen, a record general supervisor with Hewlett Packard organization who is included directly with Maersk's ITO endeavors. "There was another CIO who needed to re-appropriate each part of IT, but without [negatively] impacting shipping.”

With an end goal to help further business development and increment authoritative adaptability, Maersk chose to solidify the majority of its server farms and server rooms working overall 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.

Maersk Line, is taking a gander at planning and building up a Container Management System (CMS) to take into account deal with the compartments, an answer that decreases generally production network costs and a proficient method to oversee coordination.

1.1 Objectives

- Exhibit the comprehension of distributed computing in its different structures and how Microsoft Azure fits inside the distributed computing space.
- Investigate the Microsoft Azure improvement condition.
- Configuration, Implement and Deploy Web application on Azure

- To structurally plan productive applications using Microsoft Azure as general society cloud stage.

1.2 Scope

- From import, fare and transhipment preparing to entryway tasks.
- To have the capacity to scale the answer for address the issues of requests amid pinnacle seasons.
- Improves benefit, cuts cost, builds efficiency, kills blunders and improves assets to future-confirmation your payload taking care of business for elite.
- Assurance and unwavering quality through Failover Management.
- Manage your whole reserving procedure from timetable hunt to booking affirmation.

1.3 Requirement Specification

- Design & Develop a single tenant web application hosted on Microsoft Azure as an App Service (Web App).
- Consume Azure Storage or SQL Database.
- Consist of 5 - 10 interlinked pages.
- Provide quality content and design.
- Analyse web application performance with monitoring tools.
- To suggest the solution to scale-up or scale-down to meet the needs of demands during peak seasons.
- Source code to place in source control management services.

1.4 Major Function

Some of the Major functions of the Application are as follows:-

- **Login Function:** One a user has their own username and password they van use those detail to login into the system. This is what Login Function does.
- **Search Function:** This function allows a user to search schedules of the container.
- **Register Function:** This function allows a new user to register so that they can access the system.
- **Book Function:** After accessing the system, user is allowed to book their containers.
- **Register Function**: The system allows normal client to register to the system for performing the function of booking the container.

- **Add Container:** Users who works as an admin can add the containers number with its details.
- **Book Container:** The system allows client check the availability to book the container in a time schedule.
- **Search Container:** The system allows client to search the availability of the container from the specific zone and its countries.
- **Shipment Container:** The system allows end-users to view the shipment details but client can see those details which they have booked from the system.
- **Add Country:** User who works as an admin can directly add country along with the details of the container.
- **View Report:** End-users can view their recorded report of the bookings the container.
- **Schedule Search:** End-users can perform schedule search to search the reports of the booking in a range of time.
- **Edit/Update Booking:** Users who works as an admin can edit/update their bookings.
- **Approve Bookings:** Admin approve the bookings of the users and set it to the depart and arrival date and time.

2. PROJECT PLAN

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Introduction**	**11 days**	**Sat 7/28/18**	**Thu 8/9/18**		**Arbindra Sutih**
2		Background	2 days	Sun 7/29/18	Mon 7/30/18		Arbindra Sutihar
3		Objectives	2 days	Tue 7/31/18	Wed 8/1/18		Arbindra Sutihar
4		Scope	2 days	Thu 8/2/18	Fri 8/3/18		Arbindra Sutihar
5		Requirement Specification	2 days	Mon 8/6/18	Tue 8/7/18		Arbindra Sutihar
6		Summary of the major Function	2 days	Wed 8/8/18	Thu 8/9/18		Arbindra Sutihar
7		Project Plan	1 day	Fri 8/10/18	Fri 8/10/18		Arbindra Sutihar
8		**Design**	**27 days**	**Sun 8/5/18**	**Mon 9/10/18**		**Arbindra Sutih**
9		Cloud Design Patterns	4 days	Mon 8/6/18	Thu 8/9/18		Arbindra Sutihar
10		Architectrial Diagram	3 days	Mon 8/13/18	Wed 8/15/18		Arbindra Sutihar
11		Design Consideration	3 days	Fri 8/17/18	Tue 8/21/18		Arbindra Sutihar
12		Modelling	6 days	Wed 8/22/18	Wed 8/29/18		Arbindra Sutihar
13		**Implementation**	60 days	Mon 9/3/18	**Fri 11/23/18**		**Arbindra Sutih**
14		**Test Plan and Testing Discussion**	21 days	Wed 11/28/18	**Wed 12/26/18**		**Arbindra Sutihar**
15		**Conclusion**	2 days	Fri 12/28/18	**Mon 12/31/18**		**Arbindra Sutih**

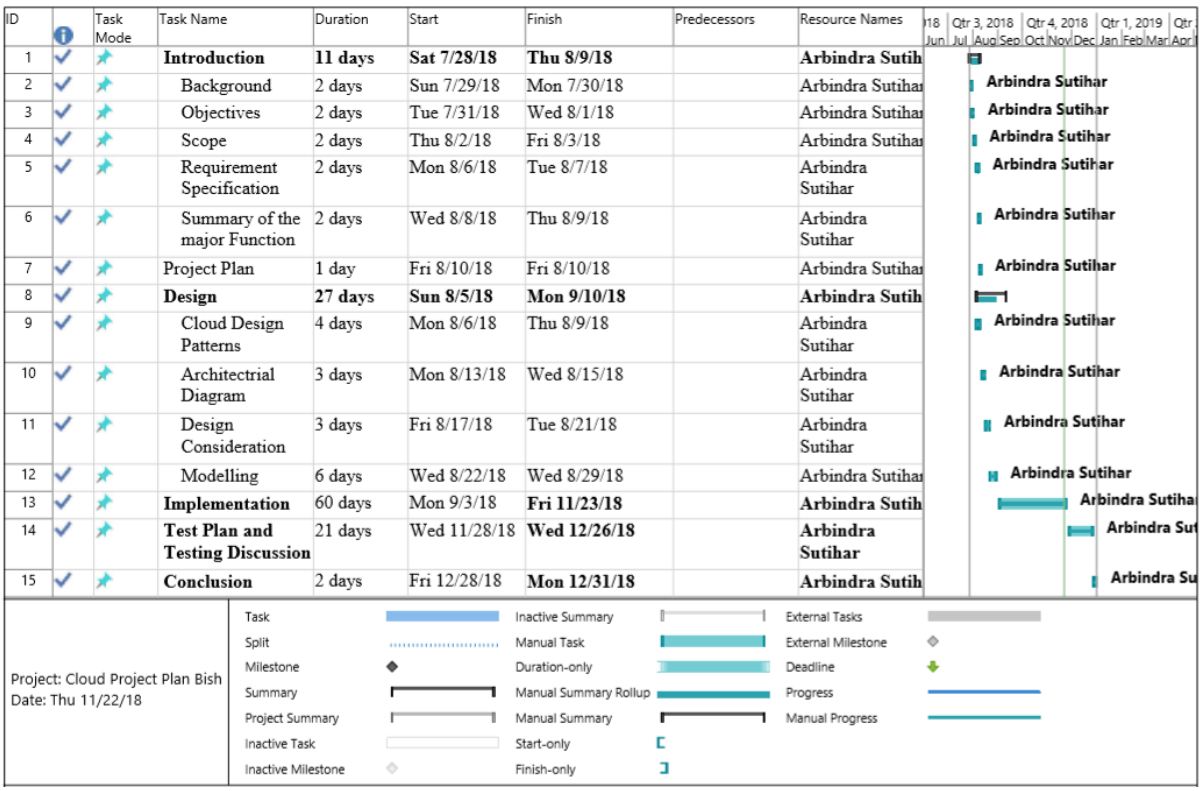

Figure 1Project Plan

3. DESIGN

3.1 Cloud Design Patterns

These structure designs are helpful for building solid, versatile, secure applications in the cloud.

Each example portrays the issue that the example addresses, contemplations for applying the example, and a precedent dependent on Microsoft Azure. The greater part of the examples incorporates code tests or scraps that demonstrate to execute the example on Azure. Notwithstanding, the greater part of the examples are important to any dispersed framework, regardless of whether facilitated on Azure or on other cloud stages. , most of the patterns are relevant to any distributed system, whether hosted on Azure or on other cloud platforms (Christopher, 2017) (Azure, 2018)

Some of the tools are: -

Ambassador: Create helper services that send network requests on behalf of a consumer service or application.

Anti-Corruption Layer: Implement a façade or adapter layer between a modern application and a legacy system.

Backends for Frontends: Create separate backend services to be consumed by specific frontend applications or interfaces.

Bulkhead: Isolate elements of an application into pools so that if one fails, the others will continue to function.

Cache-Aside: Load data on demand into a cache from a data store.

Retry: Enable an application to handle anticipated, temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that's previously failed.

Throttling: Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service.

Valet Key: Use a token or key that provides clients with restricted direct access to a specific resource or service.

3.2 Architectural Diagrams

The architecture diagram below shows how the Maersk Line web service works on the Azure cloud. First, it will deploy on native host server so it will join to the Azure cloud local resources web services to deploy to the cloud. And then, there will be a SQL database backend to attach to the database information server and additionally a traffic manager to manage the outgoing control.

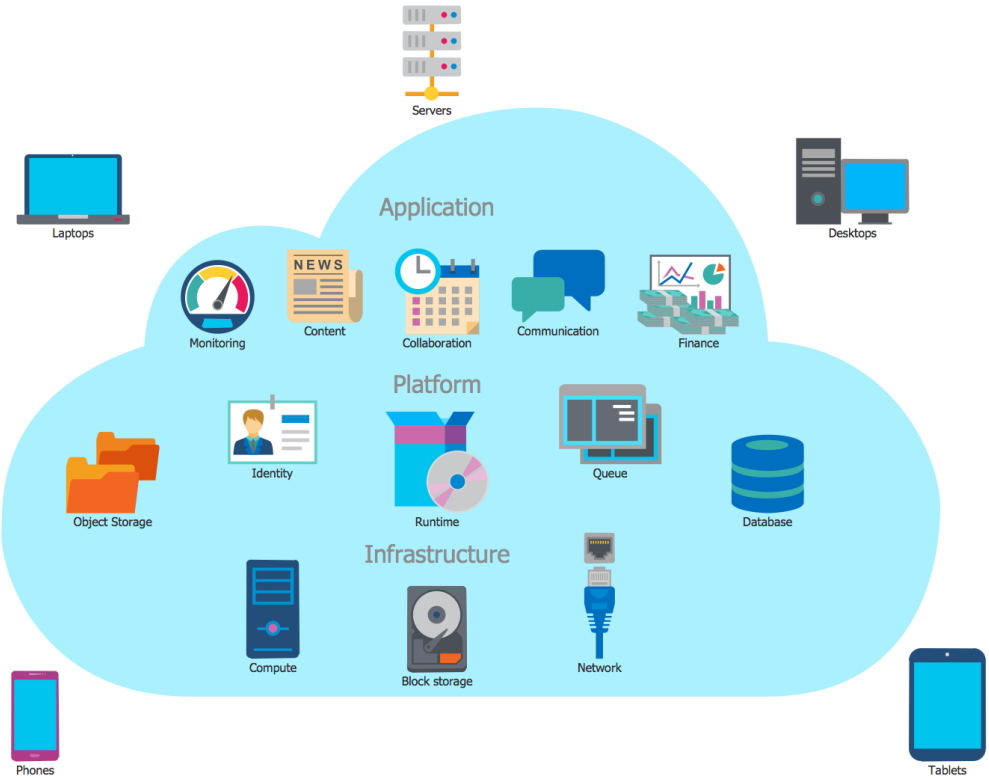

Figure 2 Cloud Computing Architecture

The Cloud Computing setup is that the course of action of the structure that incorporates neighborhood and cloud resources, organizations, middleware and programming pack, cloud clients and cloud reserves, frameworks, geo-territory. Note that the Cloud Computing arrangement relies upon the necessities of the end-customer - the cloud client and depicts at any rate of these parts are made and related. For account the Cloud Computing Architecture with a target to energize the correspondence between accomplices are adequately used the Cloud Computing Architecture diagrams. Each Cloud Computing Architecture plot apparently depict the cloud parts and associations between them. (Segal, 2018).

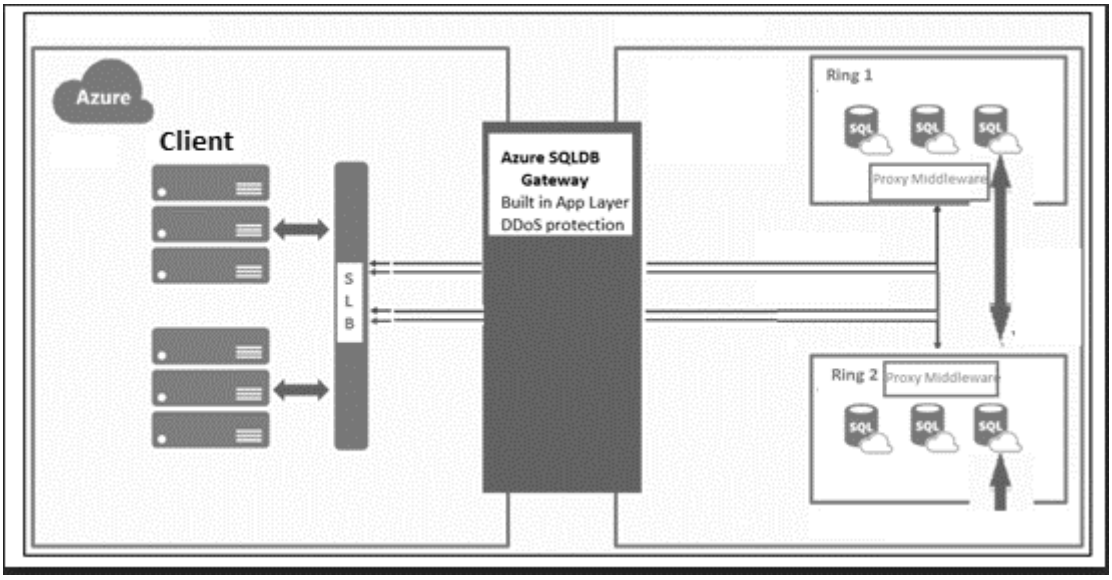

Figure 3Azure SQL Database Connectivity

Source: https://docs.microsoft.com/en-us/azure/sql-database/media/sql-database-connectivity-architecture/architecture-overview.png

Purposed application is based on N-Tire Architecture which divide application into logical and physical tires which are the ways of separating responsibilities and manage dependencies with specific responsibility with each different Tire. Whereas a higher layer can use services in a lower, but not the other way around.

Tiers are physically separated, running on separate machines. A tier can call to another tier directly, or use asynchronous messaging (message queue). Although each layer might be hosted in its own tier, that's not required. Several layers might be hosted on the same tier. Physically separating the tiers improves scalability and resiliency, but also adds latency from the additional network communication. A traditional three-tier application has a presentation tier, a middle tier, and a database tier. The middle tier is optional. More complex applications can have more than three tiers. The diagram above shows an application with two middle tiers, encapsulating different areas of functionality.

An N-tier application can have a closed layer architecture or an open layer architecture:

- In a closed layer architecture, a layer can only call the next layer immediately down.
- In an open layer architecture, a layer can call any of the layers below it.

A closed layer architecture limits the dependencies between layers. However, it might create unnecessary network traffic, if one layer simply passes requests along to the next layer (Wasson, 208).

3.3 Design Considerations

Methodology, client, markets, innovation, laws, gauges and rivalry, because of these factor plans are constantly unique. A portion of the structure thought which has been pursued are portrayed beneath:

Scalability

Scalability should center around any necessity in order to add extra ability to the application and related administration to deal with increments in load and request. Especially it is critical to consider every application level when planning for versatility. (Brown, 2018)

Capability

- Singular application layer should be done or not should be thought about additionally by what means may we achieve this without impacting openness is contemplated.
- How quick do we have to scale solitary organization ought to be figure?
- How would we add additional capacity to the application or any bit of it?
- Will the application need to continue running at scale 24x7, or would we have the capacity to cut back outside business hours or at closures of the week for example?

- **Platform**
- Work can be done within the constraints of our chosen persistence service while working at scale or not.
- How can we partition our data to aid scalability within persistence platform constraints (e.g. maximum database sizes, concurrent request limits, etc.)?
- How can we ensure we are making efficient and effective use of platform resources? As a rule of thumb, I generally tend towards a design based on many small instances, rather than fewer large ones.
- Can we collapse tiers to minimize internal network traffic and use of resources, whilst maintaining efficient scalability and future code maintainability?

Load

- How designing can be improving in order to avoid contention issues and bottlenecks? For instance, can queues be used or service bus between services in a co-operating producer, competing consumer pattern?
- For handling asynchronously to help balance load at peach times which operation should be used.
- How can be platform features for used for rate-levelling.

3.4 Modelling

3.4.1 Use-Case Diagram

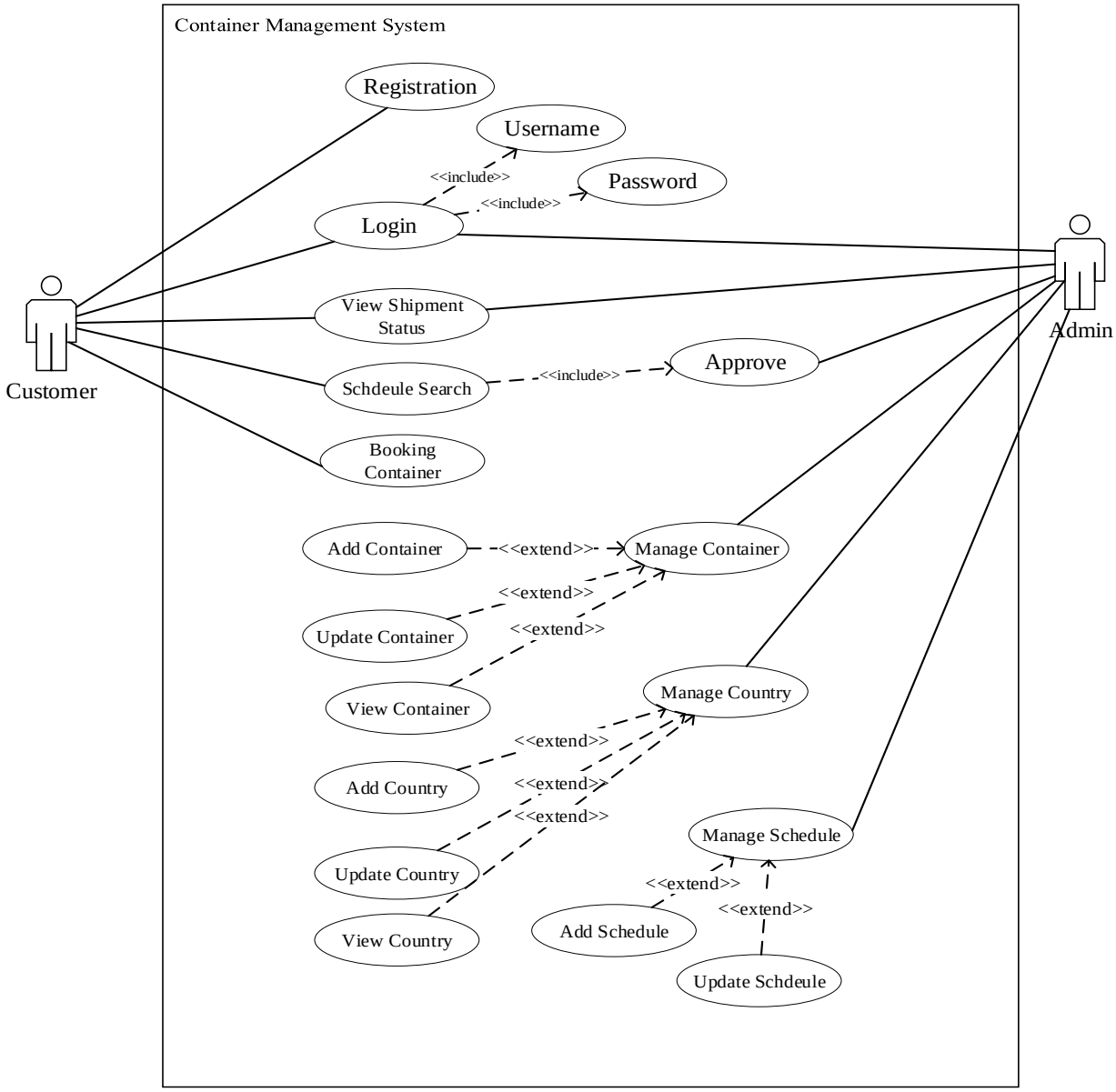

Figure 4 Use Case Diagram

3.4.2. Use Description

Use case	Register Users
Brief description	This use case enables guest users to registered
Actors	Customer
Precondition	Nil

Main flow	1. Fill the fields with valid user general information 2. Fill the valid bank details 3. Click on the Register button. 4. User is created and credentials are passed to the respective user.
Alternative flow	• If a user email is already taken then error message is displayed. • Any invalid fields will display the error
Post condition	Registration successful.

Table 1 Use Case Specification Register Users

Use case	Login
Brief description	This use case enables actors to login to the system.
Actors	Customer
Precondition	Have a valid email and password
Main flow	1. Enter the email and a password. 2. Click on the Login button. 3. Actor is logged in to the system and redirected to the Dashboard according to their roles.
Alternative flow	If email or password is not valid then error message is displayed
Post condition	Login successful.

Table 2 Use Case Specification Login

Use case	Search container schedule
Brief description	This use case enables actors to search for the container schedule on the system.
Actors	Customer
Precondition	Customer should be logged in
Main flow	1. Select country name and Timestamp to search available containers in given date. 2. Click on the search button. 3. Results are displayed.
Alternative flow	No containers are found
Post condition	Available containers in given date are shown

Table 3 Use Case Specification Search container schedule

Use case	Book Container
Brief description	This use case enables actors to book containers
Actors	Customer

Precondition	Customer should be logged in and available container should be selected
Main flow	1. Enter weight required 2. Click on book button
Alternative flow	Weight entered is not more than available
Post condition	Container booking request has send for approval

Table 4 Use Case Specification Booking Container

Use case	Approve Booking Request
Brief description	This use case enables actors to approve book containers
Actors	Admin
Precondition	Admin should be logged in
Main flow	1. List of booking request are shown 2. Click approve button to approve the booking
Alternative flow	Click reject for not approving booking
Post condition	Booking is approved

Table 5 Use Case Specification Booking Container Approve by Admin

Use case	Approve Shipping received
Brief description	This use case enables actors to confirm shipping received
Actors	Admin
Precondition	Admin should be logged in
Main flow	1. List of shipping are shown 2. Click received button to confirm shipping is received
Alternative flow	Leave list as it is for no change
Post condition	Shipping is received

Table 6 Use Case Specification Shipping Received

3.4.2 Entity Relationship Diagram (ER-Diagram)

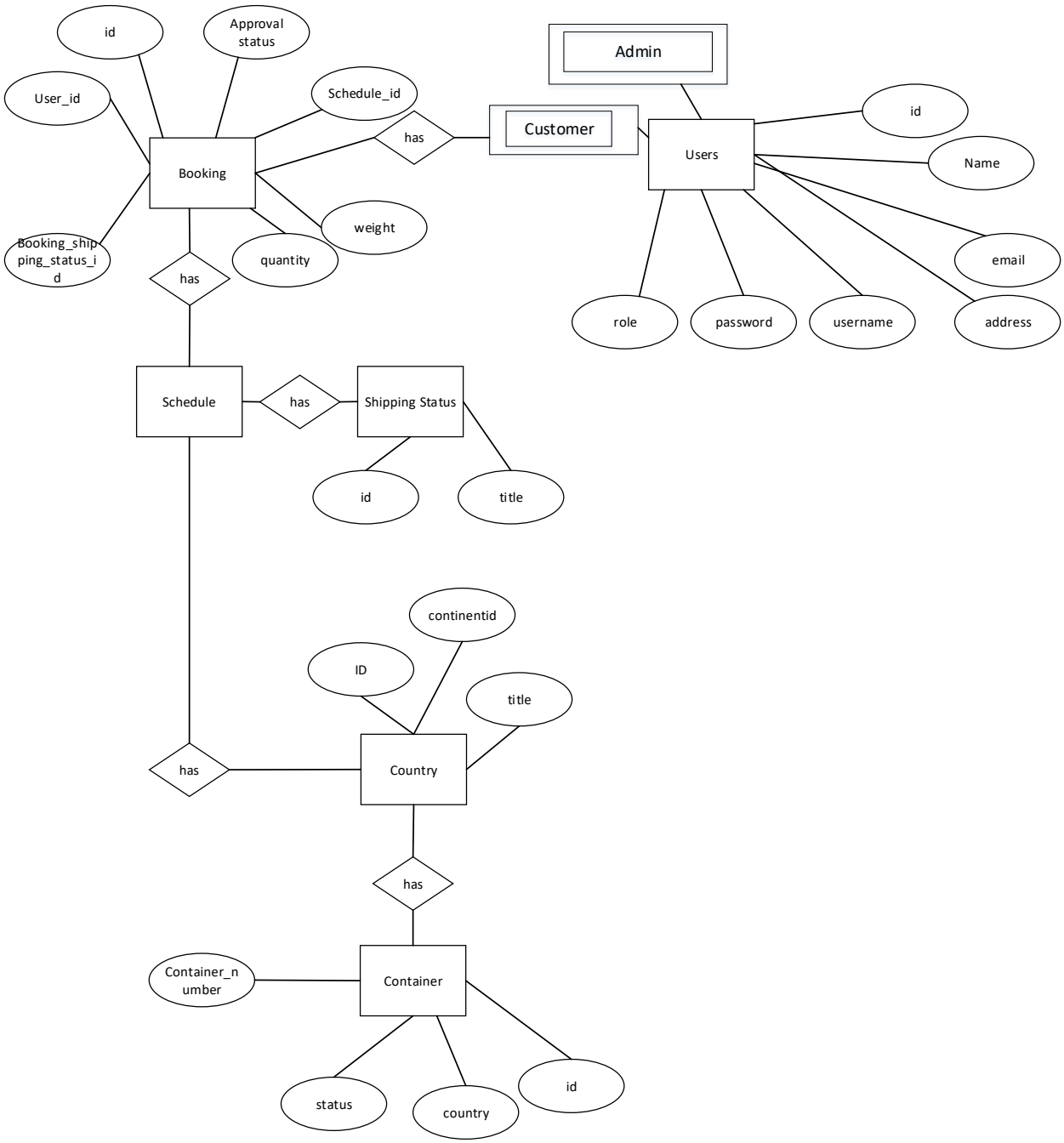

Figure 5 ER Diagram

3.4.3 Sequence Diagram

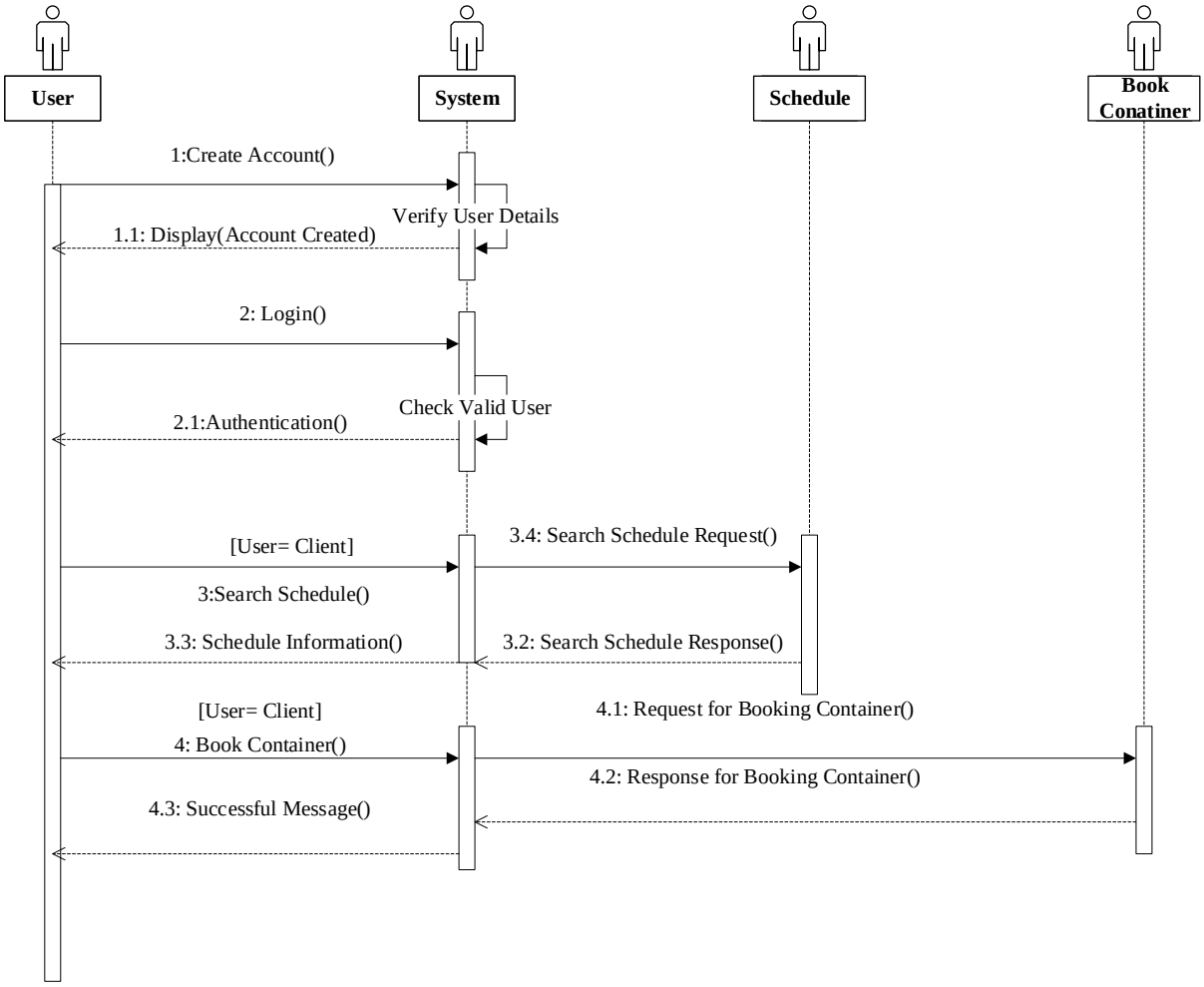

Figure 6Sequence Diagram- Admin

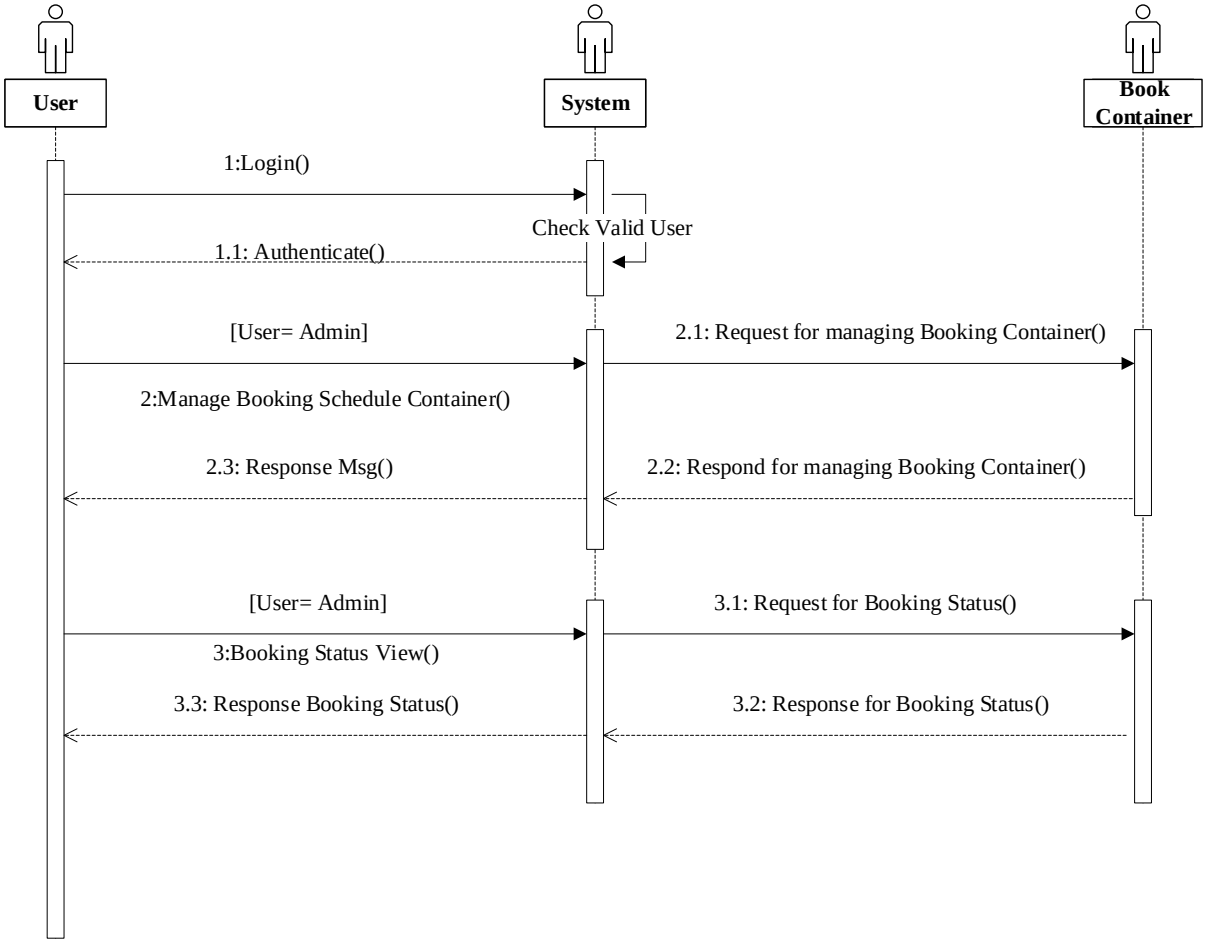

Figure 7Sequence Diagram-Client

4. IMPLEMENTATION

The proposed system aims to solve the problem of Maersk Line, by designing and developing a Container Management System. The system is developed using Asp.net with c# and back end with MsSql Language and for the Data Storage, Azure Store is chosen.

4.1 Screenshot

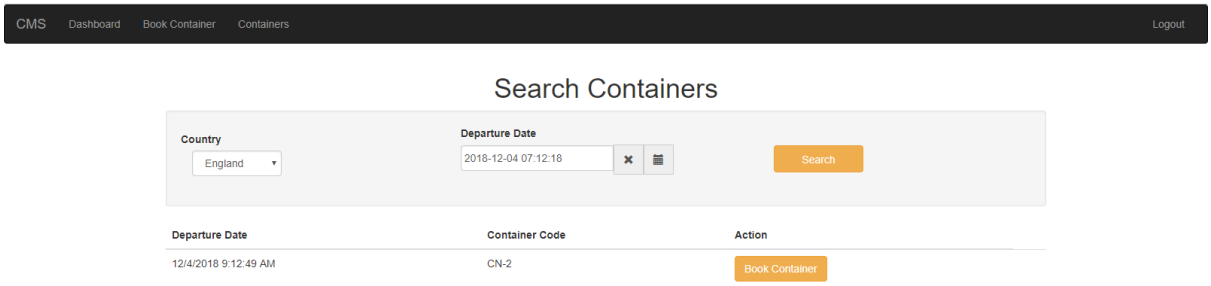

Figure 8 Search Container

Description: -Search container page helps us to search container via unique number and country from which we can see other status of the container once container is found.

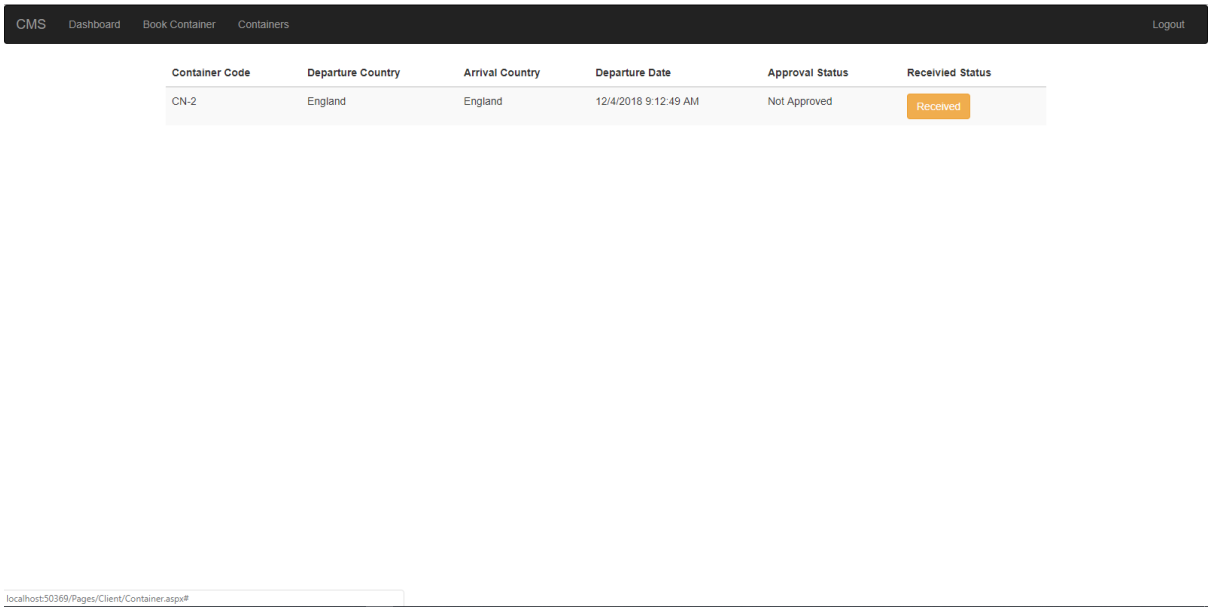

Figure 9 Booking Conformation

Description: - Booking container page let the user to view the container code, departure country, arrival country, approval date, approval status and finally receive status from which we can see whether container has reached its destination or not.

4.2 Deployment

Azure Web Apps provides a highly scalable, self-patching web hosting service. The step shows how to deploy ASP.NET Core web app to Azure Web Apps. After finishing, we will have a resource group that consists of an App Service plan and an Azure web app with a deployed web application. An account is necessary before beginning the deployment. First of all, an application should be made in ASP.NET

At the point when the application is finished the application ought to be distributed by tapping the distribute enter in the arrangement. Subsequent to distributing marking into Azure is required in the wake of marking in the primary deployment activity starts the steps are mentioned as follows: -

Create an App Service plan

An App Service plan specifies the location, size, and features of the web server farm that hosts your app. When hosting multiple apps by configuring the web apps to share a single App Service plan money can be saved.

App Service plans define:

Region (for example: North Europe, East US, or Southeast Asia)

Instance size (small, medium, or large)

Scale count (1 to 20 instances)

SKU (Free, Shared, Basic, Standard, or Premium)

Next to **Hosting Plan**, select **New**.

Configure Hosting Plan dialog screenshot is displayed below where the settings were allocated are described in the table.

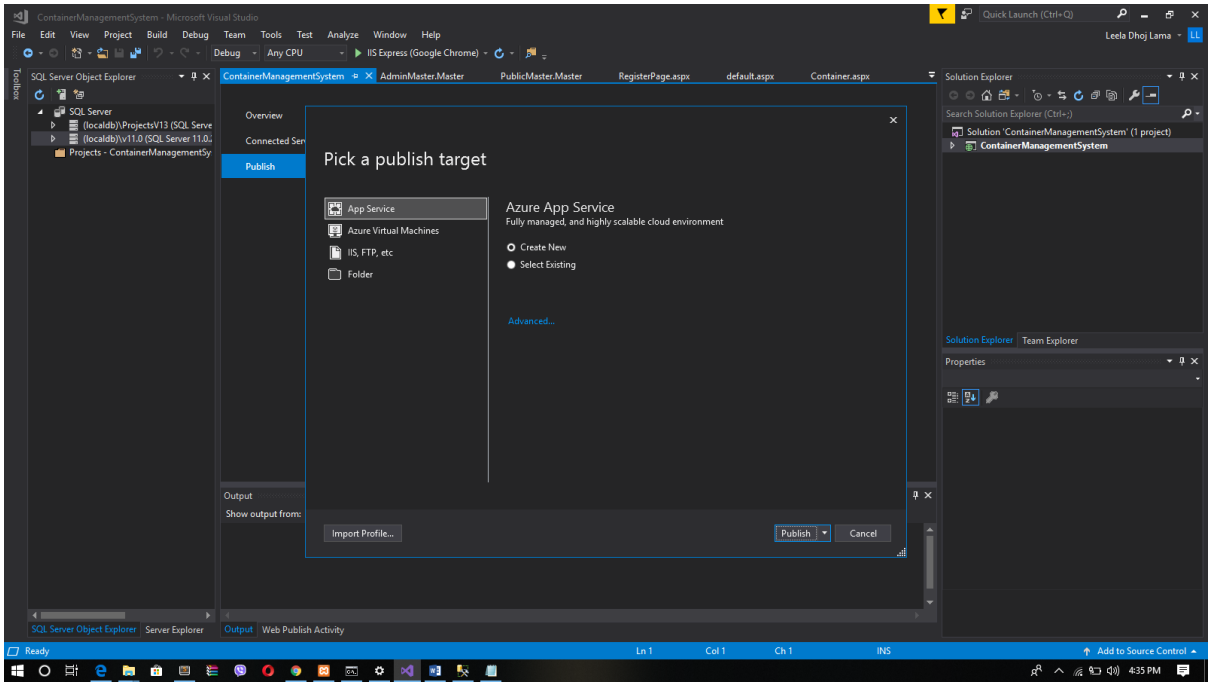

Figure 10 Configure Hosting Plan

Create and publish the web app

Subsequent to arranging facilitating plan azure resources must be characterized and the web application had been made. When the wizard finishes, it distributes the ASP.NET Core web application to Azure, and afterward dispatches the application in the default program. After the application name is determined in make and distribute. At that point ASP.NET Core web application is running live in Azure App Service subsequent to refreshing and redeploying the application tapping the distribute catch on the distribute synopsis page. When distributing finishes, Visual Studio dispatches a program to the URL of the web application.

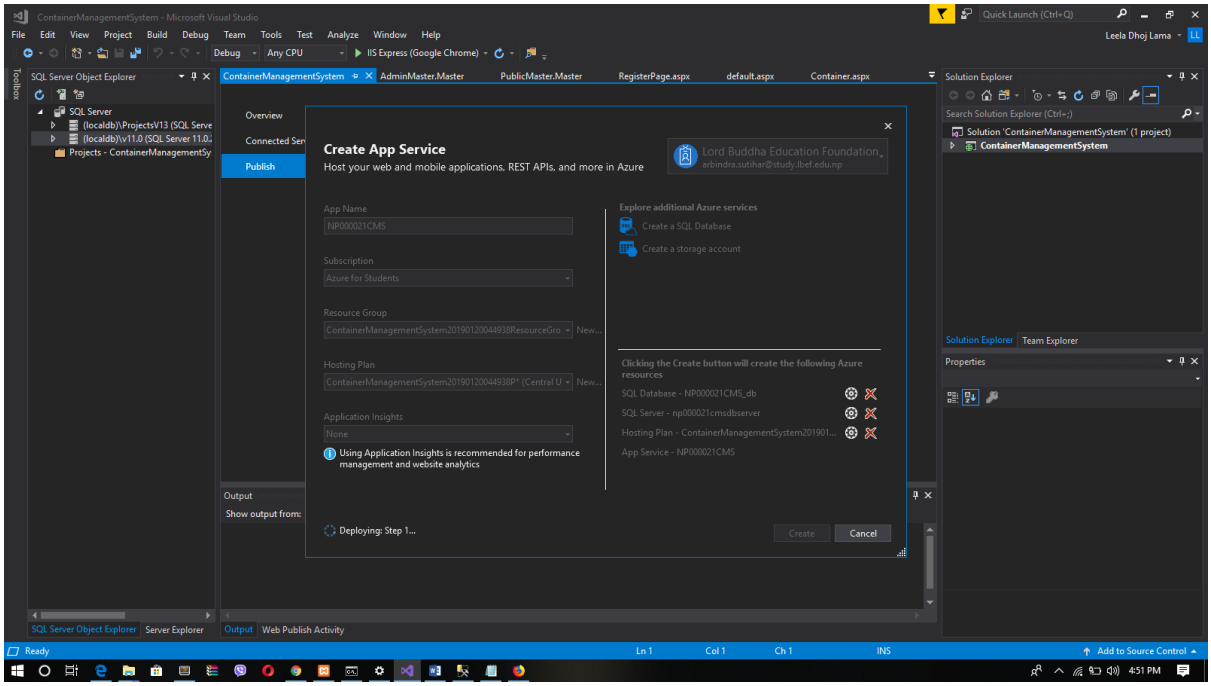

Figure 11 Create App Service

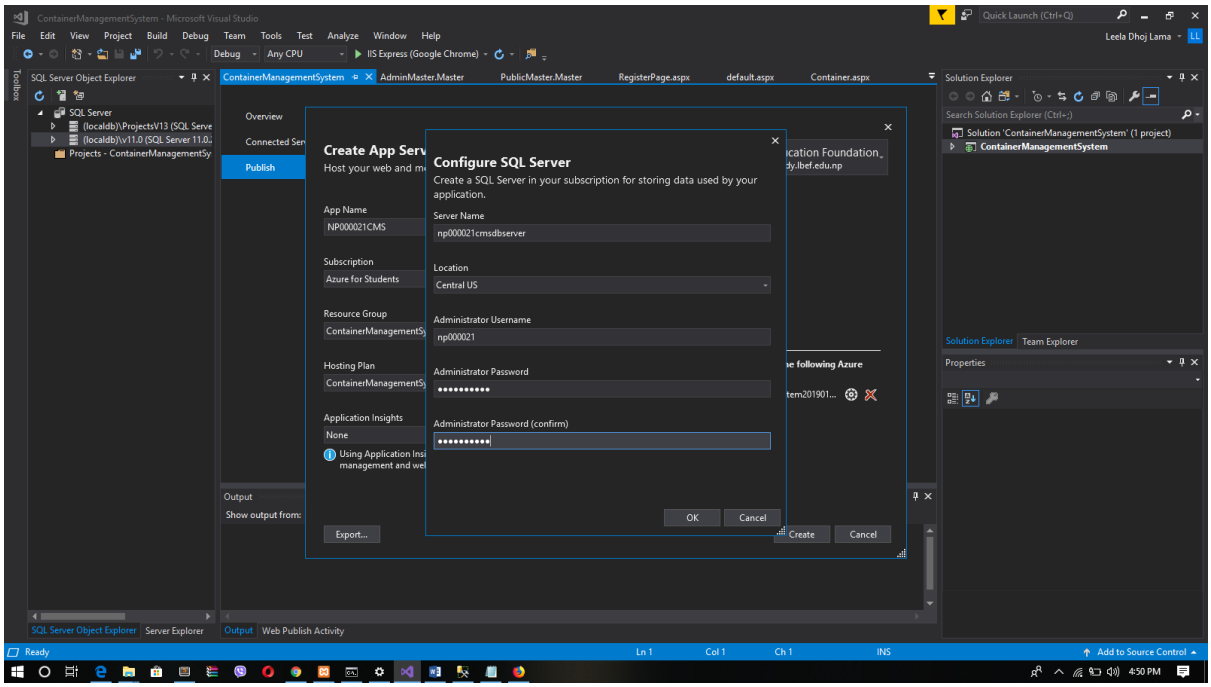

Figure 12 Configure Application Server

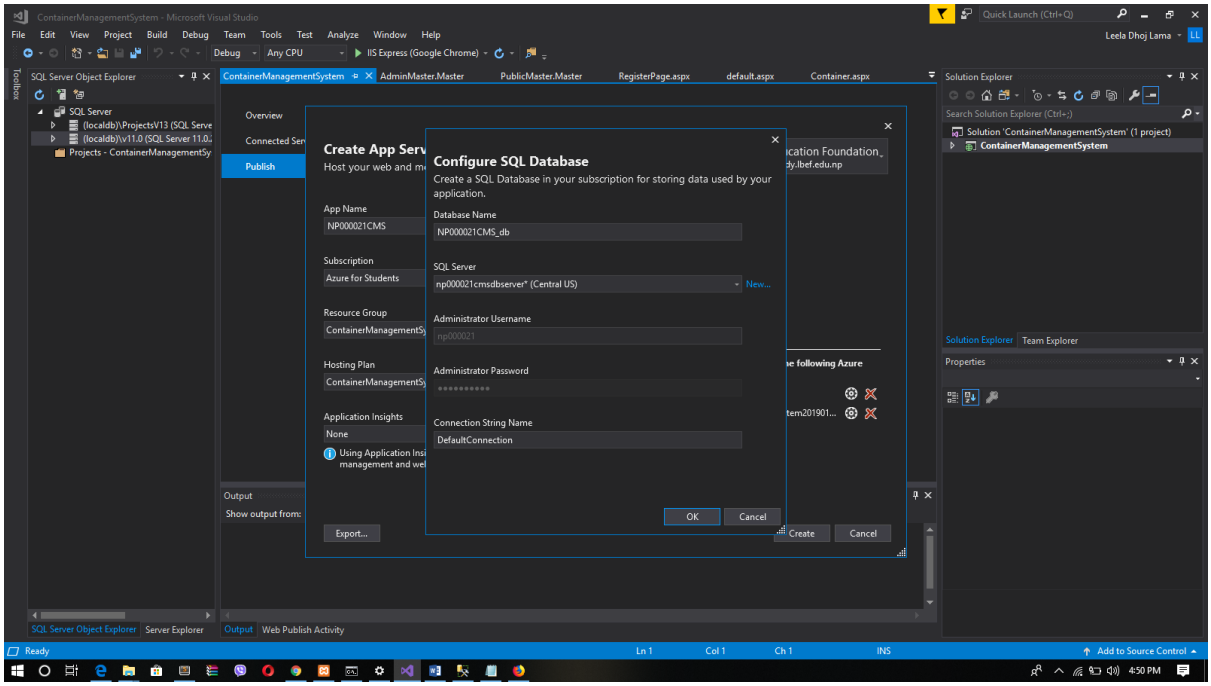

Figure 13 Configuring SQL Database

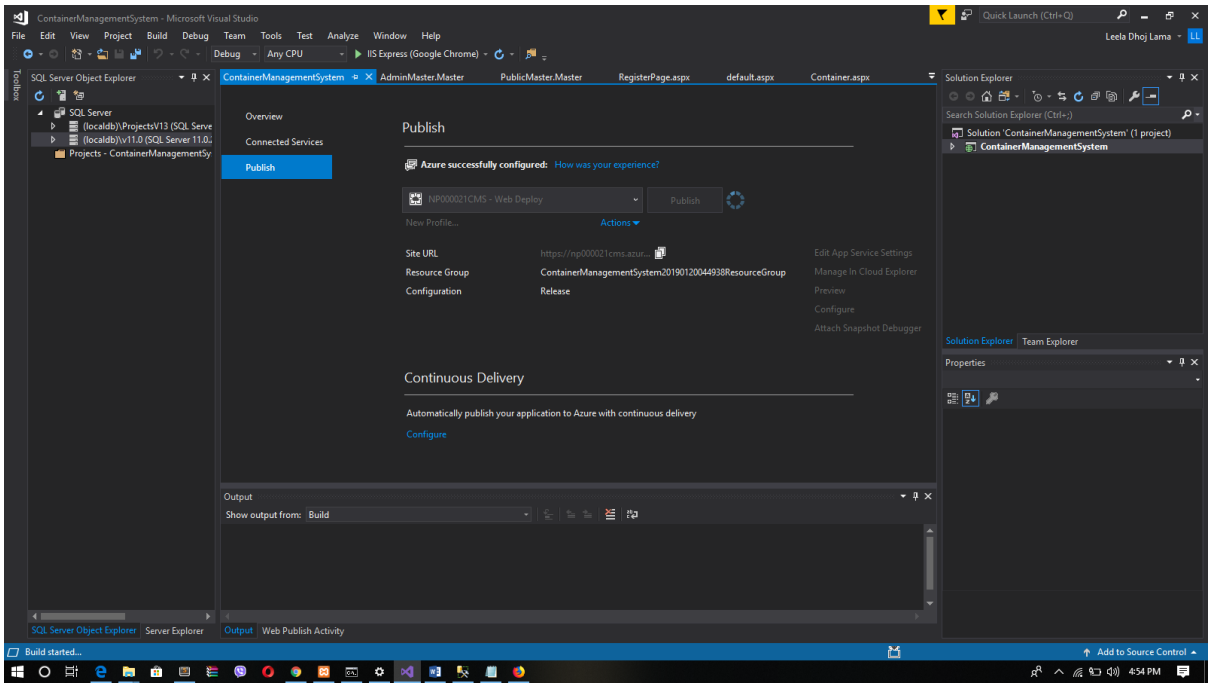

Figure 14 Publish Succeeded

Create and publish the web app

After configuring hosting plan azure resources has to be defined and the web app had been created. Once the wizard completes, it publishes the ASP.NET Core web app to Azure, and then launches the app in the default browser. After the app name is specified in the create and publish. Another step is used as the URL prefix in the format http://<app_name>.azurewebsites.net. Then ASP.NET Core web app is running live in Azure App Service after updating and redeploying the app clicking the publish button on the publish summary page. When publishing completes, Visual Studio launches a browser to the URL of the web app.

Manage the Azure web app

After effectively finishing the previously mentioned ventures through the Azure entrance the web can be overseen as appeared in the screen capture beneath: -

Figure 15Managing Azure Web App (i)

This is the overview page where we can perform basic management tasks like browse, stop, start, restart, and delete.

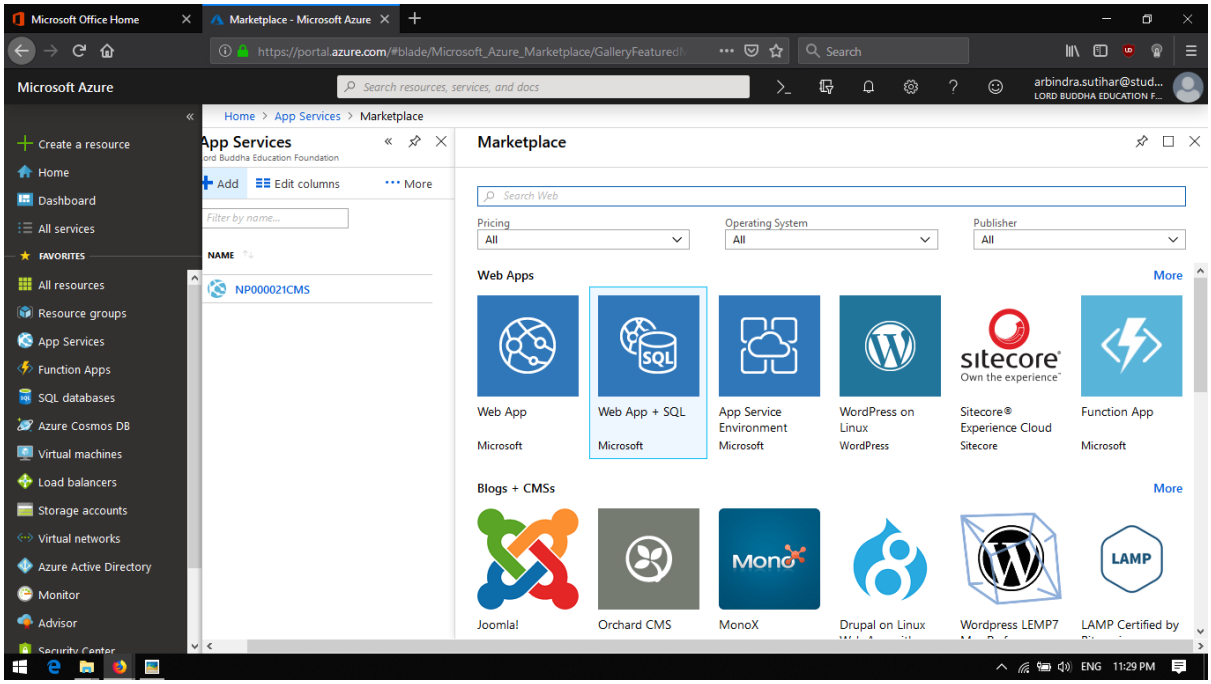

Figure 16Managing Azure Web App (ii)

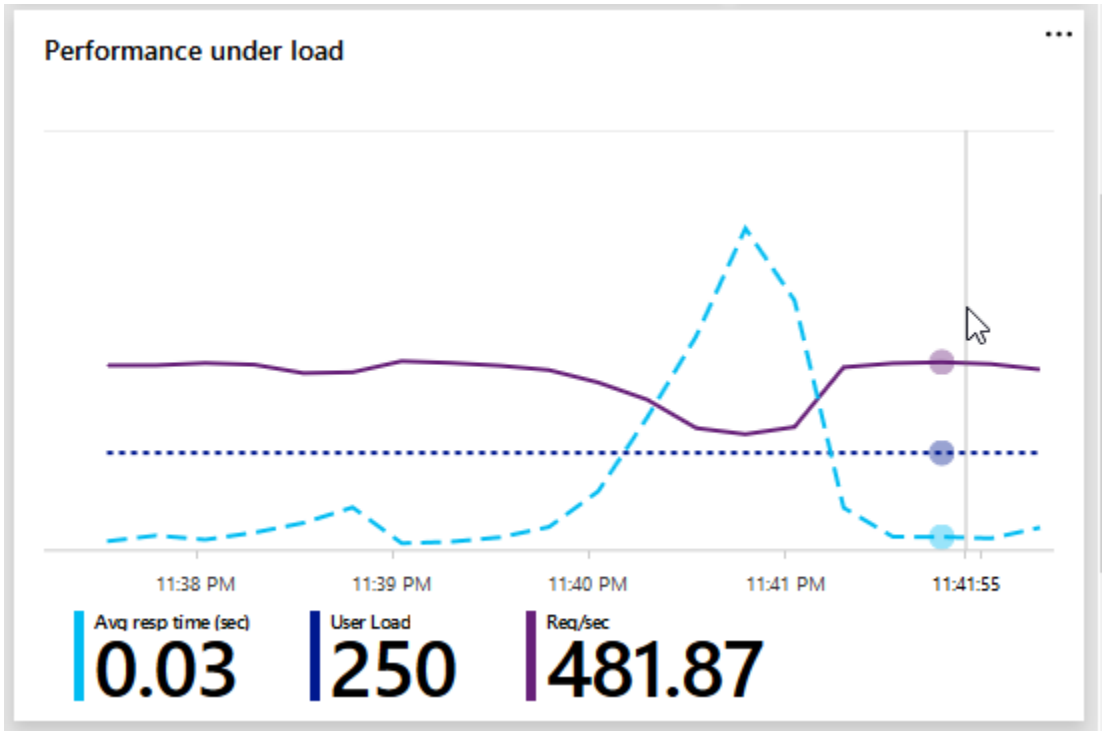

Figure 17Managing Azure Web App (ii)

The left menu provides different pages for configuring the app. (Microsoft Azure, 2018)

4.3 Application Scaling

For the most part extends are begin little with fundamental highlights. Distinctive sites or web application are beginning little toward the start however it won't remain in same frame as each venture are upgraded by the interest. As per the improvement of the site or web application assets additionally ought to be scale to run framework easily. While conveying the site determination of the correct valuing tire for the site is essential. When the site is finished and facilitated it ought to make certain that site is accessible and responsive according to the business needs staying serenely inside a financial plan. (Tardif, 2013)

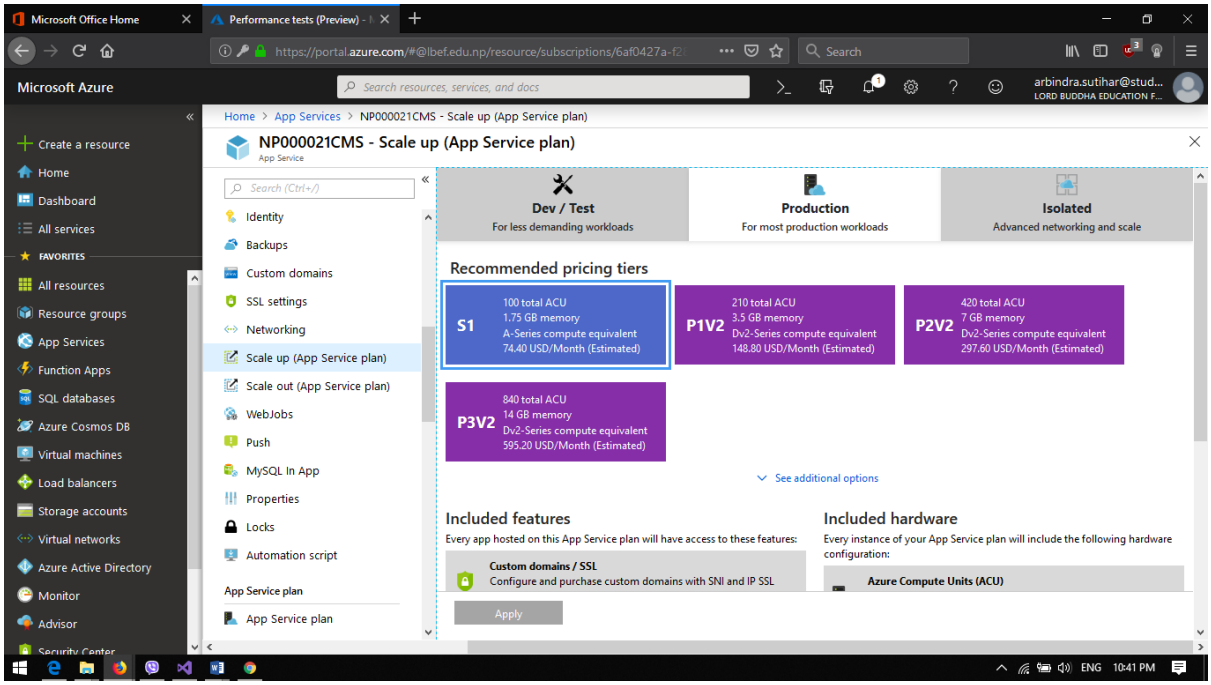

Figure 18 Web App Scaling

Windows Azure Web Sites (WAWS) offers 3 modes i.e. Standard, Free and Shared. Each of these modes has their very own detail. They offer distinctive arrangement of amounts that keep up the quantity of assets that a site can devour. They likewise give distinctive scaling abilities. (Tardif, 2013)

Azure provides multiple ways to scale website using Management Portal. Websites can be scale through the visual studios as well if the site is being maintain in visual studio. Different ways to scale sites are (Luijbregts, 2017)

Vertical Scale

Vertical scaling is procedure of scaling the number and power if assets here and there. Vertical scaling will be scaling here and there. For a precedent greater server with more torque is utilized for scale up and a littler server with lower strength one to downsize. (Luijbregts, 2017)

Horizontal Scale

Horizontal scaling is procedure of scaling number of equipment assets that the applications runs. Even scale is scaling in and out. Scaling out is expanding the quantity of equipment assets and scaling in is diminishing number of equipment assets utilized by the site. (Luijbregts, 2017)

Scale up

Scale up is the procedure in the Azure in which non-cloud site are moved to a greater physical server. It is valuable when the current modes are exceeding or a site is hitting portion. It tends to be done on essentially on any site without stressing over the outcomes of multi-instances data consistency. In Azure scale up operation for the website can be done by manually in the Azure Portal and with automation using Azure Rest API (Luijbregts, 2017)

Scale out

Scale out is process of increasing the number of hardware resources that the application runs. Scale out can be done using azure portal and the Azure Rest API. As scaling is the process of increasing number of hardware resources, number of instance of the app is increased. The simplest way of scaling instance is manually. (Luijbregts, 2017)

In initial phase system can be deployed using limit sources but in future as per the requirement the application be scaled. Hardware resources can be increased and the system can be deployed in server with more horsepower or capacity.

4.4 Performance Testing of Application

Sending of the framework isn't end of the framework improvement. Execution of framework additionally ought to be test to guarantee that is performing great. Purplish blue additionally gives the administration of execution testing. In this way, we can without much of a stretch screen and test the execution of use. Utilizing sky blue entry execution testing of use should be possible.

For the compartment the executive's framework heap of 250 simultaneous clients were provided. Execution testing is done to discover the quantity of clients that the framework can deal with. Execution testing result for the framework is demonstrated as follows

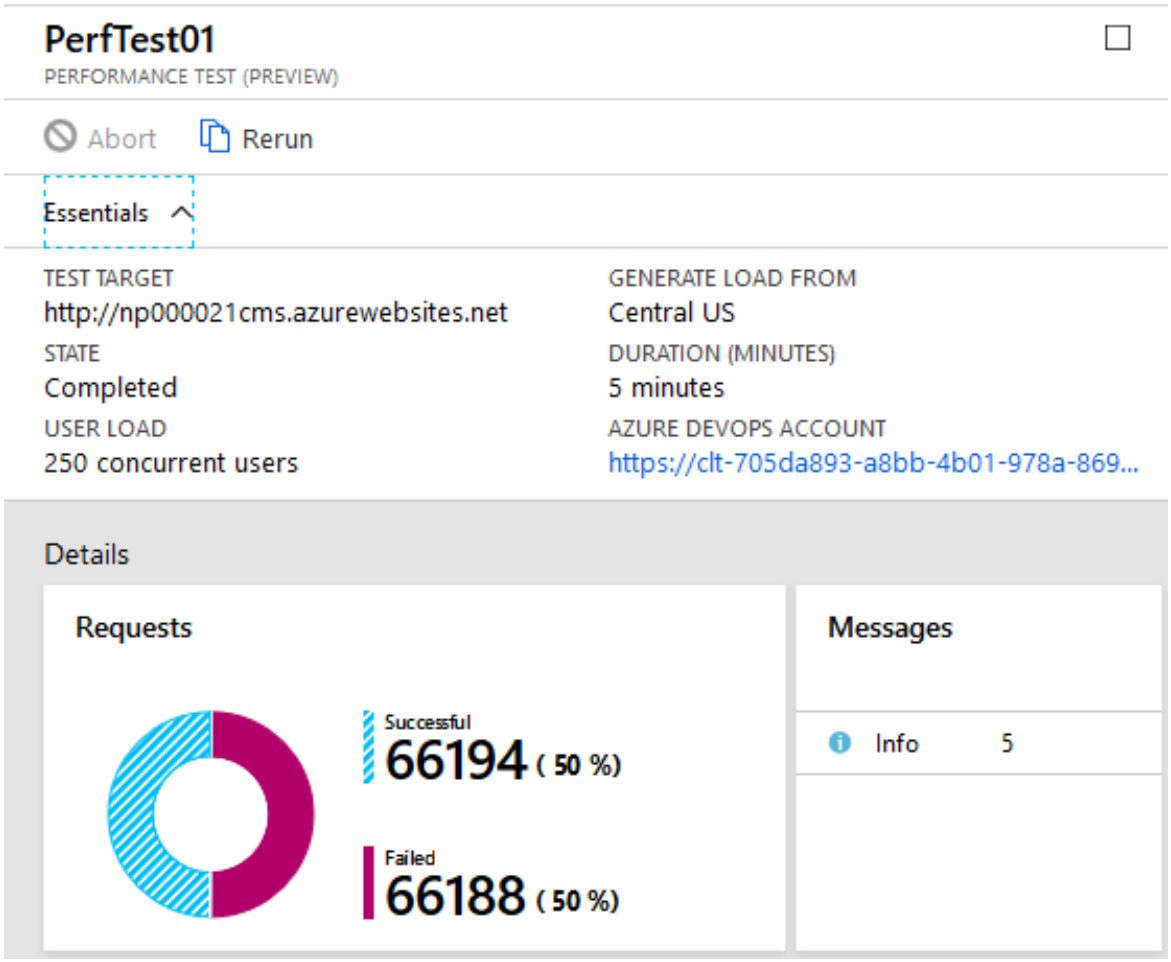

Figure 19 Performance test Under Load(a)

By doing performance testing it can be concluded that the system has successfully response 90.16% for the request and 9.84% were failed to response. Azure also provide testing results more details.

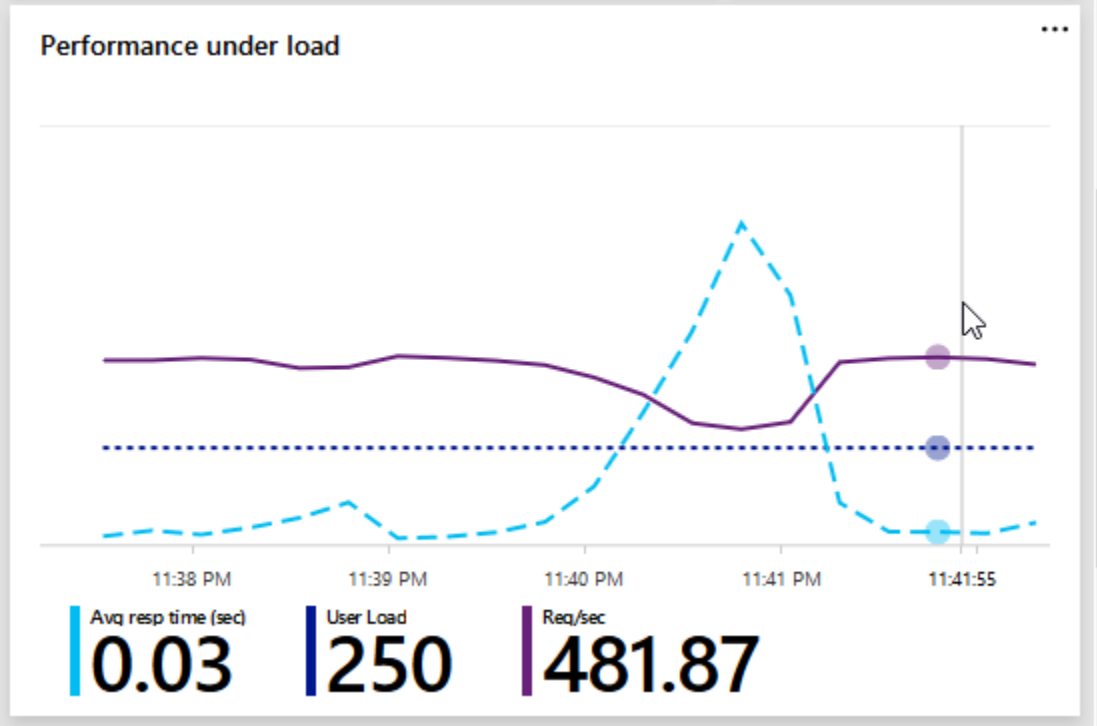

Figure 20Performance test Under Load(b)

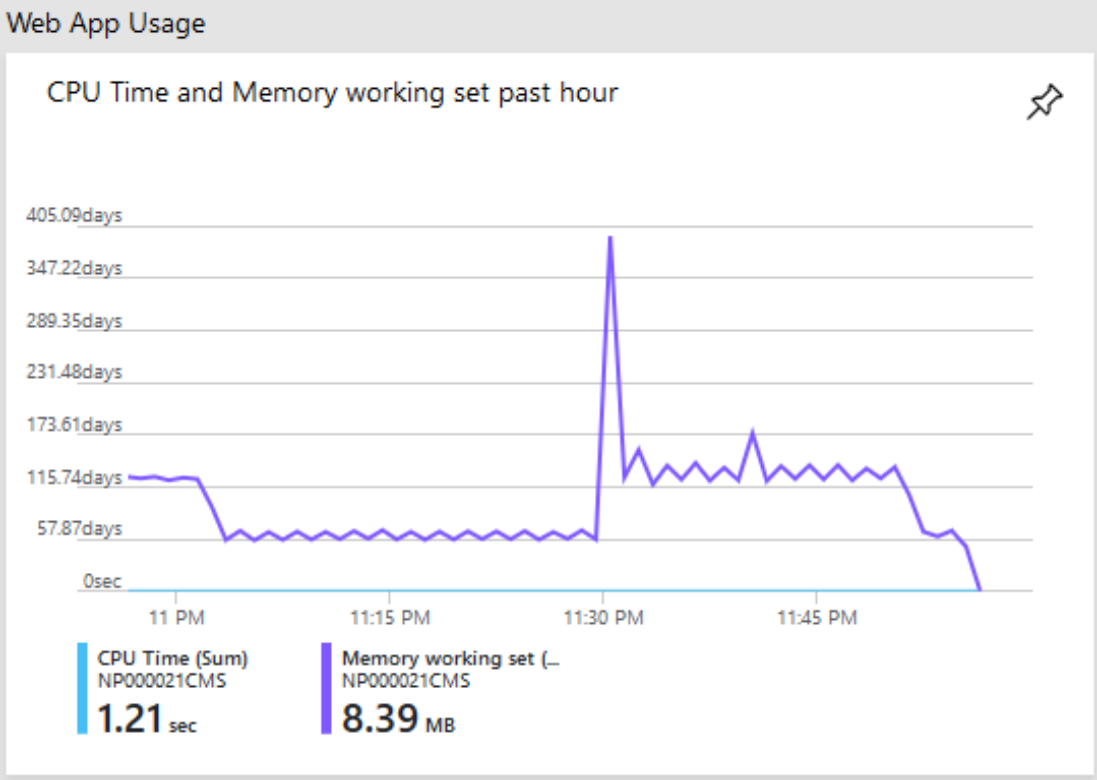

Figure 21Performance test Under Load(c)

Above figure is of execution under load. It demonstrates the execution of the framework under the heap of 250 clients and can be presumed that the framework can keep running in the heap of 250 clients.

As purplish blue give checking instruments to the administrator to screen the framework execution and diverse asset that has used by the framework, administrator can without much of a stretch screen the execution of framework.

4.5 Azure Platform as a Service (PaaS)

Platform as a Service (PaaS) is an administration of cloud given by the purplish-blue which finish advancement and organization condition in the cloud with assets which empower to convey everything from cloud-based applications to modern cloud-empowered endeavor application. PaaS incorporates foundation, for example, servers, stockpiling and organizing and furthermore middleware improvement instruments, business insight (BI) administrations, database the board frameworks and that's only the tip of the iceberg. It bolsters the entire web application lifecycle: building, testing, sending, overseeing and refreshing. (Microsoft Azure, 2018)

For the system Deployment of the system in azure PaaS is used as it provided the environment for the deployment. As it provides the resources such as database management system, it makes easier to deploy because the system contains the database and requires the database management system

5. TEST PLAN & TESTING DISCUSSION

Testing is a procedure of executing a program with the point of discovering mistake. To influence our product to perform well it ought to be sans blunder. In the event that testing is done effectively it will expel every one of the mistakes from the product. Testing ought to be begun as right on time as conceivable to diminish the expense and time to adjust and create programming that is without bug so it very well may be conveyed to the customer. There are diverse sorts of testing which should be possible for the framework, for example

Unit testing

Unit testing is a trying of programming where single units or parts of a product are tried. The reason for performing unit testing is to approve whether every unit of the product executes as required or not. A unit is the littlest testable piece of any product. Units has normally one or a couple of sources of info and as a rule has a solitary yield. In procedural programming, a unit might be an individual program, work, system, and so forth. In question situated programming, the littlest unit is a strategy, which may have a place with a base super class, conceptual class or determined or kid class.

Integration Testing

Integration Testing is a trying of programming where single units are joined together and tried as a gathering. The reason for this testing is to discover blames in the joint effort between coordinated units.

System Testing

System Testing is a trying of programming where a total and coordinated programming is tried. The reason for this test is to check whether the frameworks satisfies the predetermined prerequisites. It is critical to test the framework as entire in light of the fact that the framework may not satisfy the necessities or execute as required.

For the project Unit testing is done. Unit testing is important because every unit in the system should perform as required to fulfill the requirement of the system. If the units of the system does not perform well then the system does not perform well. Below tables show the testing plans of the unit in the system.

This is the arrangement which particularly accentuation on discovering mistakes in the framework. Each client needs a framework which is without mistake so the framework is viable

to work with. This arrangement will permit to make sense of some other minor and significant blunders from the framework. Testing ought to be begun as right on time as conceivable to decrease the expense and time to modify and create programming that is without bug so it very well may be conveyed to the customer. Amid the testing stage following things were utilized.

Every user wants a system which is error free so that the system is effective to work with. This plan will allow to figure out any other minor and major errors from the system. Testing should be started as early as possible to reduce the cost and time to rework and produce software that is bug-free so that it can be delivered to the client. During the testing phase following things were used.

Table 7: Test Plan for Login

Module Name	**Login**		
	Test description	**Excepted Result**	**Action Result**
1	Submitting the form without any detail in any column.	Error message is displayed	
2	Submitting wrong password but correct username	Error message is displayed	
3	Submitting correct password and correct username	Takes into the dashboard	

Table 8: Test Plan for Booking

Module Name	**Booking**		
	Test description	**Excepted Result**	**Action Result**
1	Search for location, Select Available Container and Submit	List of containers are displayed and booking is confirmed	
2	Search for location, does not select container and submit	Error message is displayed	
3	Submitting correct password and correct username	Takes into the dashboard	

6. CONCLUSION

Maersk Line works in excess of 600 vessels and has a limit of 2.6 million TEU. Maersk Line is the overall holder division and the greatest working unit of the A.P. It is the world's greatest compartment shipping association having customers through 374 work environments in 116 countries. Working in 100 countries and transporting stock far and wide, at first look it would appear to be Danish conveyance association Maersk Line is starting at now managing all the cargo it can direct. Progressing, Lorenzen says Maersk is at present changing over its IT setup reliant on Microsoft Azure, start with the work region condition up to compartment organization. Microsoft Azure was by then encouraging a bit of Maersk's IT condition, and in March 2016 Maersk at first pushed toward Microsoft about expanding the degree of the relationship. Moller – Maersk Group, a Danish business total. Regardless, when Maersk affirmed that the volume of most of the items it was shipping had created as far as possible, the association picked that cloud filled courses of action would be a basic bit of adjusting the situation.

REFERENCES

Azure, M., 2018. *Microsoft Azure.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/architecture/patterns/
[Accessed 21 10 2018].

Christopher, 2017. *Cloud Design Patterns.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/architecture/patterns/
[Accessed 4 November 2018].

Luijbregts, B., 2017. *Stackify.* [Online]
Available at: https://stackify.com/autoscale-azure-app-services-cloud-services/
[Accessed october 2018].

Microsoft Azure, 2018. *Create an ASP.NET Core web app in Azure.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-dotnet

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

PLURALSIGHT, 2018. *PLURALSIGHT.* [Online]
Available at: https://www.pluralsight.com/courses/azure-design-patterns-design-implementation?gclid=Cj0KCQiAxNnfBRDwARIsAJlH29DBgQrCJem1FZ1Cu-pJvcrDYCRDPCh8r7w9L1-zYlCndDxtod-kSI8aAvKAEALw_wcB&ef_id=Cj0KCQiAxNnfBRDwARIsAJlH29DBgQrCJem1FZ1Cu-pJvcrDYCRDPCh8r7w9L1-zYlCndD
[Accessed 10 10 2018].

Segal, J., 2018. *conceptdraw.* [Online]
Available at: https://www.conceptdraw.com/How-To-Guide/cloud-computing-architecture-diagrams
[Accessed 1 10 2018].

Tardif, B., 2013. *Microsoft Azure.* [Online]
Available at: https://azure.microsoft.com/es-es/blog/scaling-up-and-scaling-out-in-windows-azure-web-sites/
[Accessed 8 2018].

Wasson, M., 208. *N-tier architecture style.* [Online]
Available at: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

APPENDIX

Appendix I: Marking Scheme

Student Name: Arbindra Sutihar

No	Criteria	Mark Allocated	Score
	Documentation		
1.	Introduction	15	
2.	Project Plan	15	
3.	Design & Solution Architecture	60	
4.	Conclusion	10	
	Total (Documentation)	100	
	Implementation		
5.	Publishing an Application to Azure	10	
6.	Application Scaling with Justification	20	
7.	Testing Cloud Applications	25	
8.	Investigate & Analyze Application (Plan, collect, and interpret diagnostics and instrumentation data)	20	
9.	Implementation & Discussion of Managed Databases (PaaS)	25	
	Total (Implementation)	100	

Grading:

Grade		Range
A+	Distinction	>=80%
A		75-79%
B+	Credit	70-74%
B		65-69%
C+	Pass	60-64%
C		55-59%
C-		50-54%
D	Marginal Fail	40-49%
F+	Fail	30-39%
F		20-29%
F		0-19%