

**Individual Assignment**

**CT071-3-3-DDAC**

**Designing and Developing Applications on the Cloud**

**UC3F1702SE**

**Dr. KALAI ANAND A/L RATNAM**

**Name: Peter Lee Bor Yeong**

**TP No.: TP040784**

**Hand Out Date: 12 July 2017**

**Due Date: 20 November 2017**

# Acknowledgement

I would like to express my utmost gratitude to everyone who supported me and assisted me in doing this assignment. I am really thankful to my friends who provide me useful information and advice throughout the process of the assignment itself. I would also like to express my very great appreciation to my lecturer, Dr. Kalai Anand Ratnam for providing all the necessary information and recommendations related to this investigation report. Finally, I wish to thank my parents for their encouragement and support throughout my study.

Table of Contents

[1. Introduction 1](#_Toc498690661)

[1.1. Background 1](#_Toc498690662)

[1.2. Objectives 1](#_Toc498690663)

[1.3. Scopes 1](#_Toc498690664)

[1.4. Requirement Specifications 2](#_Toc498690665)

[1.5. Deliverables 2](#_Toc498690666)

[2. Project Plan 3](#_Toc498690667)

[3. Design 4](#_Toc498690668)

[3.1. Architectural Diagram 4](#_Toc498690669)

[3.2. Design Considerations 4](#_Toc498690670)

[3.3. Modelling 5](#_Toc498690671)

[Use Case Diagram 5](#_Toc498690672)

[Class Diagram 5](#_Toc498690673)

[Sequence Diagram 6](#_Toc498690674)

[4. Implementation 7](#_Toc498690675)

[4.1. Publishing an Application to Azure 7](#_Toc498690676)

[4.2. Application Scaling with Justification 14](#_Toc498690677)

[SQL Database Pricing Tier 14](#_Toc498690678)

[App Service Pricing Tier 14](#_Toc498690679)

[Scaling 14](#_Toc498690680)

[4.3. Investigate & Analyze Application 15](#_Toc498690681)

[4.4. Implementation & Discussion of Managed Databases 17](#_Toc498690682)

[5. Test Plan & Testing Discussion 18](#_Toc498690683)

[5.1. Unit Testing 18](#_Toc498690684)

[Login 18](#_Toc498690685)

[Register 18](#_Toc498690686)

[Change Password 19](#_Toc498690687)

[Search Flight 19](#_Toc498690688)

[Booking 20](#_Toc498690689)

[5.2. Performance Testing 21](#_Toc498690690)

[6. Conclusion 24](#_Toc498690691)

[7. References 25](#_Toc498690692)

[8. Appendix 27](#_Toc498690693)

# Introduction

## Background

Ukraine International Airlines (UIA) is the flagship carrier and largest airline in Ukraine. It operates domestic and international passenger flights and cargo services to Europe, the Middle East, the United States, and Asia. The airline is eager to expand into new markets, but problems with its website prevented it from adequately serving customers beyond Ukraine. The site experienced severe denial-of-service (DOS) attacks, which hurt site performance and reliability, and it did not have the performance needed to host visitors from many parts of the world.

UIA has long used technology to reduce costs, innovate, and improve customer service. It has gone to a paperless cockpit and uses sophisticated software for analyzing fuel economy. The airline decided that it once again needed to innovate its way out of its web challenges. Dmitriy Prudnikov, Chief Information Officer at Ukraine International Airlines, realized that migrating the website out of UIA datacenters into a public cloud could solve all these problems.

Ukraine International Airlines (UIA), is looking at designing and developing an Online Flight Booking System. UIA looked at both Microsoft Azure and Amazon Web Services and chose Azure. Azure was also very compatible with open source software, which didn’t surprise Prudnikov

## Objectives

* To expand into new markets.
* To prevent denial-of-service (DOS) attacks.
* To migrate the system into Azure cloud.
* To develop an online flight booking system

## Scopes

The scope of this project is to improve performance and reliability in flight booking for UIA. Migrating the website into Azure cloud to improve its performance and also improve the flight booking experiences of customers by routing them to the nearest server.

## Requirement Specifications

1. Manage entire booking process.
2. Creation of customer profile.
3. Configure the online booking engine so that it returns customized information, based on where your customers are located and the type of trip they are booking.

## Deliverables

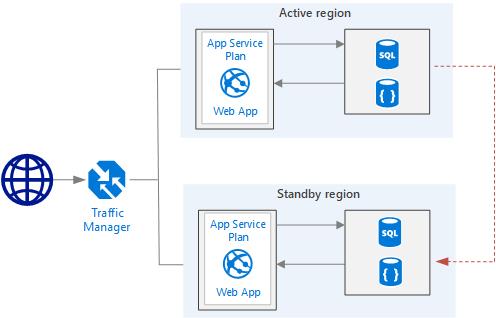
* Allow user to register account
* Allow user to login and logout
* Allow user to change their account password
* Allow user to search flight
* Allow user to make flight booking
* Allow user to check their booking history
* Allow user to see their booking details

# Project Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Finish** |
| **UIA Flight Booking System** | **60 days** | **Mon 28/08/17** | **Fri 17/11/17** |
| **Introduction** | **8 days** | **Mon 28/08/17** | **Wed 06/09/17** |
| Background | 2 days | Mon 28/08/17 | Tue 29/08/17 |
| Objectives | 2 days | Wed 30/08/17 | Thu 31/08/17 |
| Scopes | 2 days | Fri 01/09/17 | Mon 04/09/17 |
| Requirement Specifications | 2 days | Tue 05/09/17 | Wed 06/09/17 |
| Project Plan | 3 days | Thu 07/09/17 | Mon 11/09/17 |
| **Design** | **10 days** | **Tue 12/09/17** | **Mon 25/09/17** |
| Architectural Diagram | 4 days | Tue 12/09/17 | Fri 15/09/17 |
| Design Considerations | 6 days | Mon 18/09/17 | Mon 25/09/17 |
| Modelling | 9 days | Tue 26/09/17 | Fri 06/10/17 |
| **Implementation** | **29 days** | **Tue 26/09/17** | **Fri 03/11/17** |
| Publishing An Application To Azure | 8 days | Tue 26/09/17 | Thu 05/10/17 |
| Application Scaling and Justification | 6 days | Fri 06/10/17 | Fri 13/10/17 |
| Investigate & Analyze Application | 8 days | Mon 16/10/17 | Wed 25/10/17 |
| Implementation & Discussion of Managed Databases | 7 days | Thu 26/10/17 | Fri 03/11/17 |
| **Test Plan & Testing Discussion** | **7 days** | **Mon 06/11/17** | **Tue 14/11/17** |
| Testing Cloud Application | 7 days | Mon 06/11/17 | Tue 14/11/17 |
| Conclusion | 3 days | Wed 15/11/17 | Fri 17/11/17 |

# Design

## Architectural Diagram

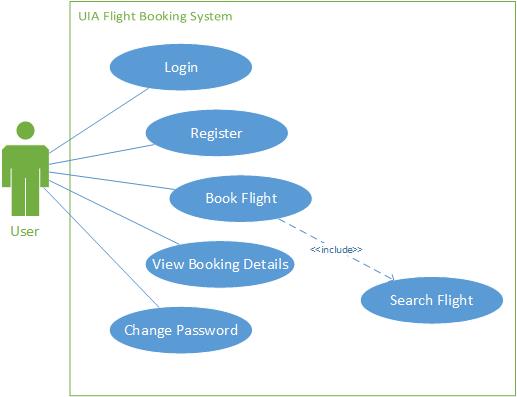


## Design Considerations

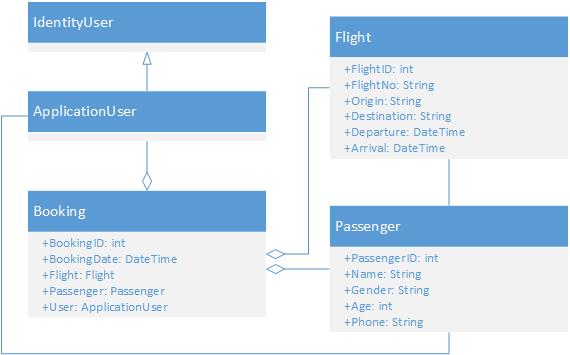
In developing the architecture, these are the considerations. The current subscription only provides RM150 credits per months. In fact, this is very limited. Therefore, every single actions in choosing services and pricing tiers should be consider carefully. In consider of this, SQL Database will be using the Basic Pricing Tier to save the cost. The cost cannot be saved from App Service pricing tier because traffic manager has to be implemented to achieve the requirement of the system which is provide customized information based on where customers are located, and in order to implement traffic manager, the pricing tier must be at least Standard. Besides, geo-replication will be implemented as well to prevent one of the server is down, there will be another backup server handling the request.

## Modelling

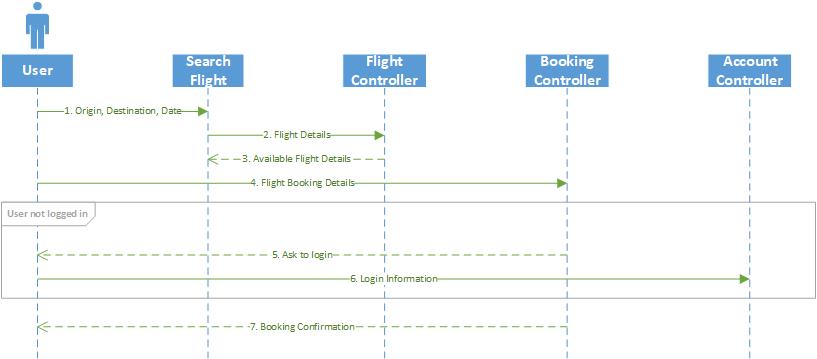
### Use Case Diagram



### Class Diagram



### Sequence Diagram

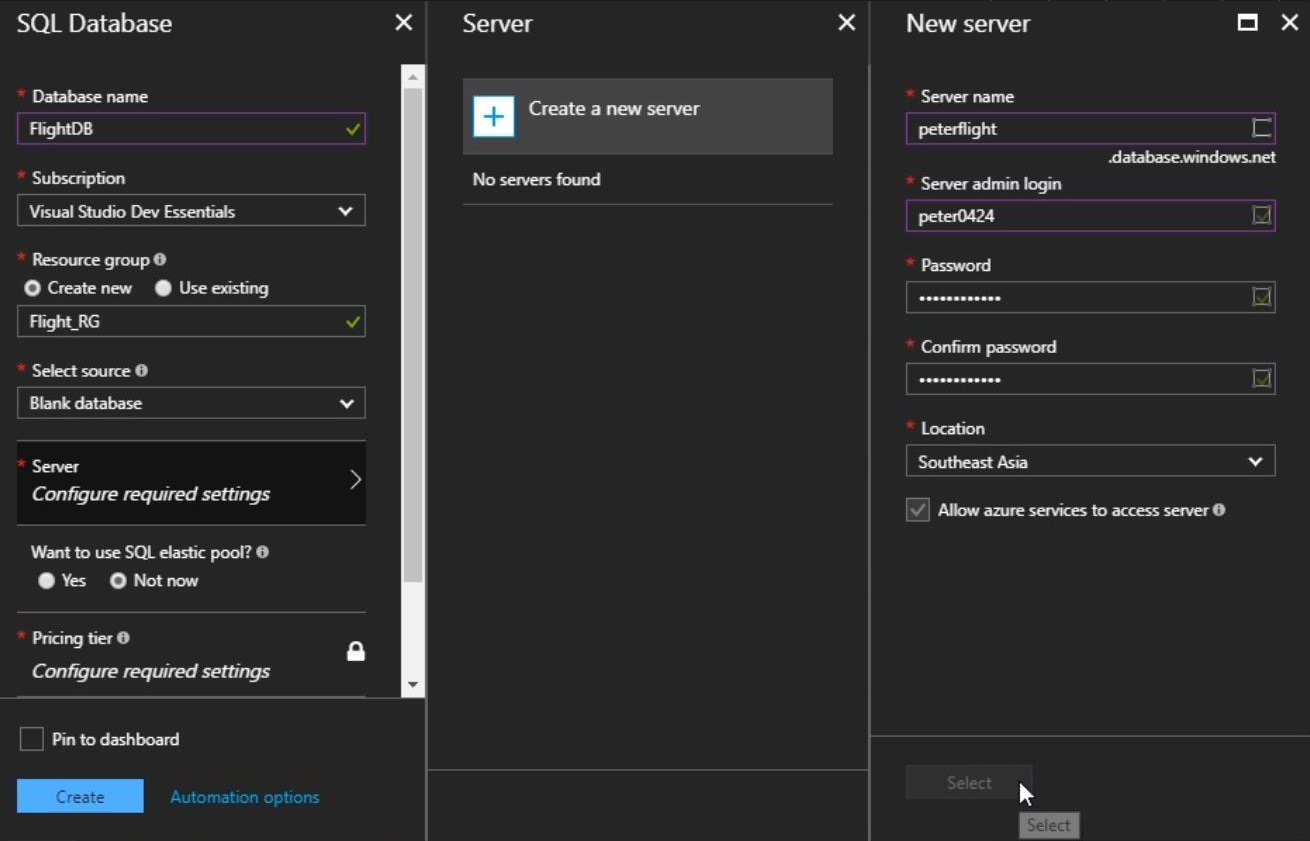


# Implementation

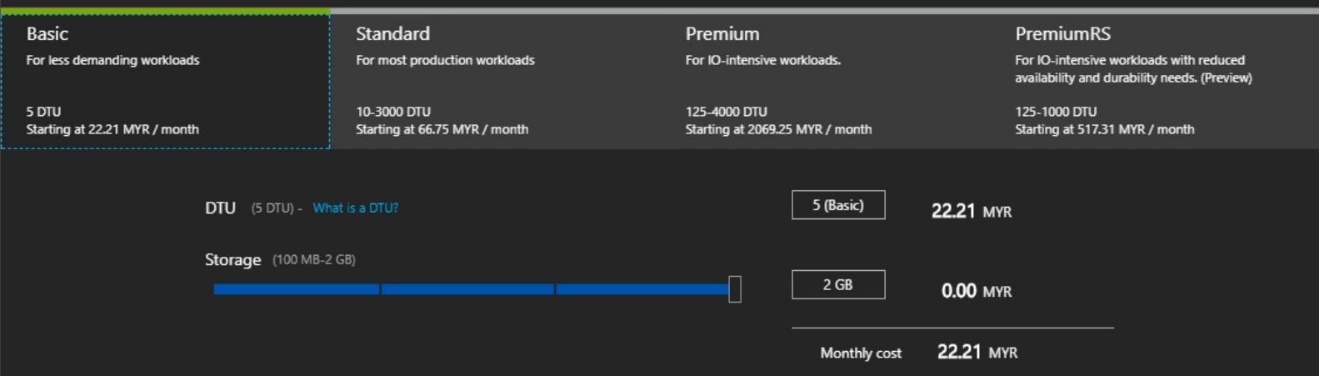
## Publishing an Application to Azure

The language chosen for this web application is ASP.NET Core. It is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications. With ASP.NET Core, developer can deploy the application to the cloud and run on .NET Core or .NET Framework. ASP.NET Core is a redesign of ASP.NET, with architectural changes that result in a leaner and modular framework (Roth, et al., 2017). Besides, the IDE chosen for developing the application is Visual Studio 2017 and the main reason behind this is that Visual Studio 2017 is integrated with the Azure so that it takes less effort to deploy and publish the web app to azure app service.

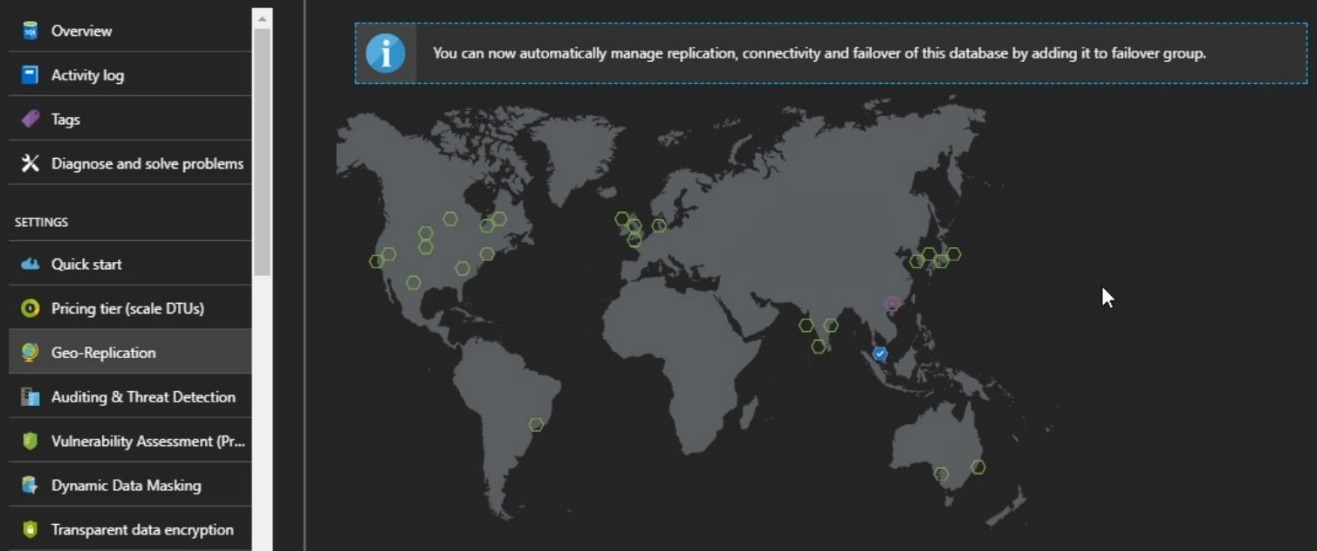
After done coding part at Visual Studio, there is needed some setup in Azure. Below are the steps taken in publishing the application to Azure.



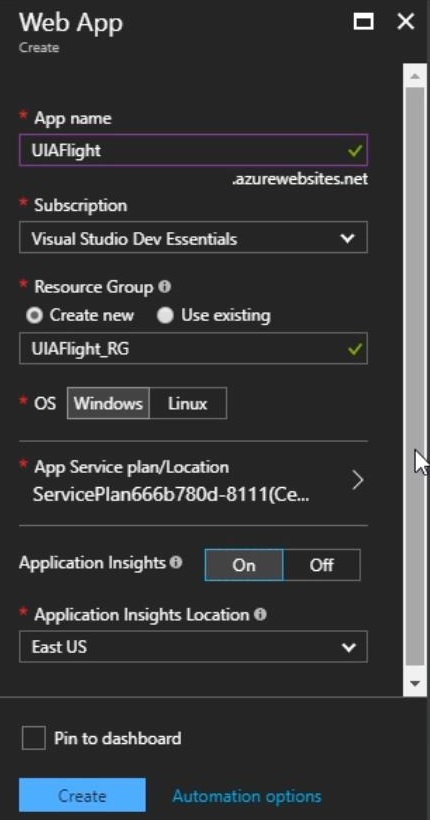
Create a SQL Database, create a new resource group for it, and also create a new server and set to the nearest location, in this case, is Southeast Asia.



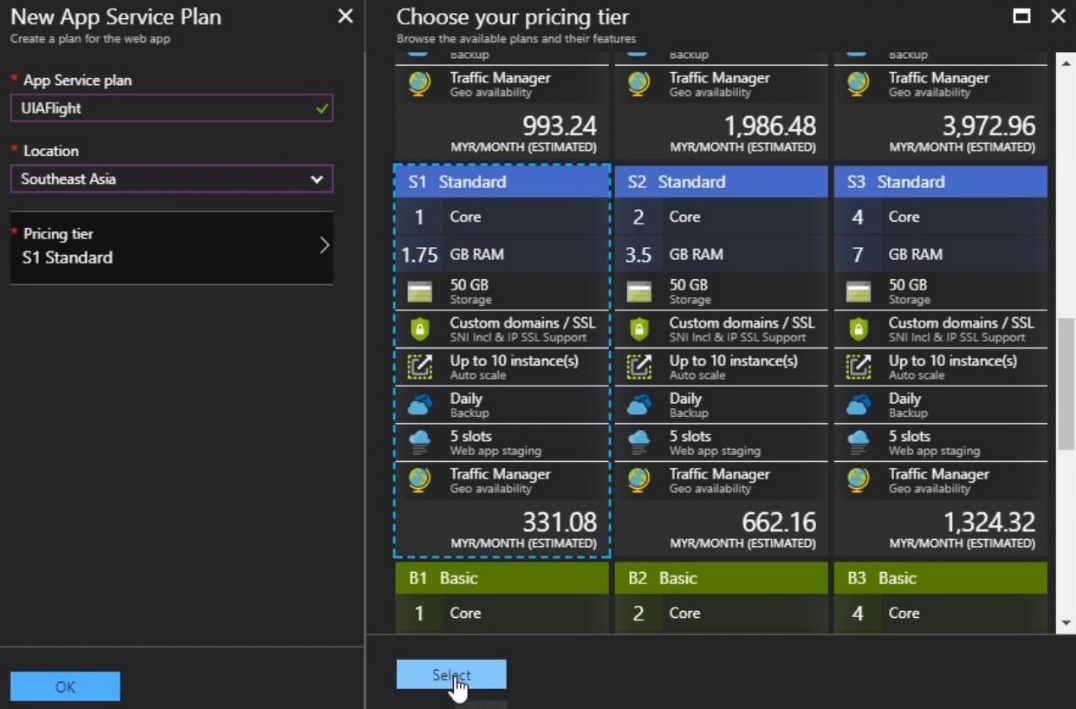
Choose the pricing tier for the SQL Database, in this case, the chosen pricing tier is Basic, which come with 5 DTUs and 2GB of storage space per months. This tier is chosen is because the workload of this web app is not heavy and the price is very affordable.



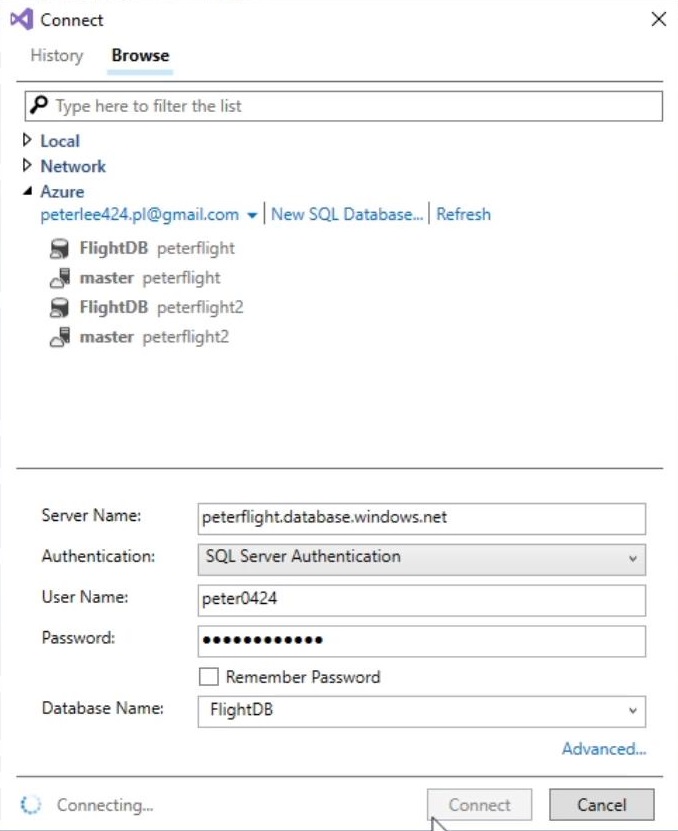
After the deployment succeed, activate the geo-replication for this database. The location of the secondary database will be at West US. The pricing tier of the secondary database and its server will be exactly the same as the primary one.



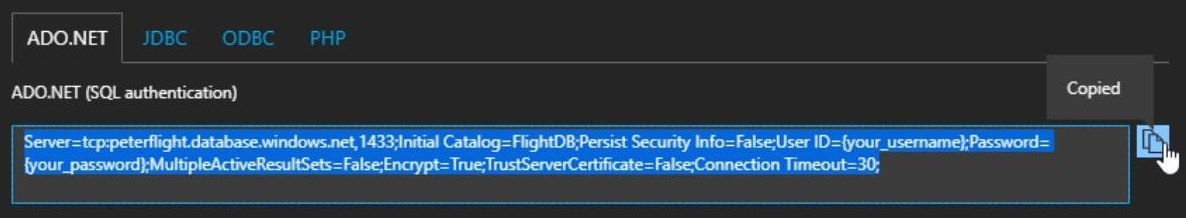
After finish create and setup the databases, now create a Web App. Populate the web app’s details, and turn on the Application Insights, which enables developer to monitor the application in terms of help, status and performance.



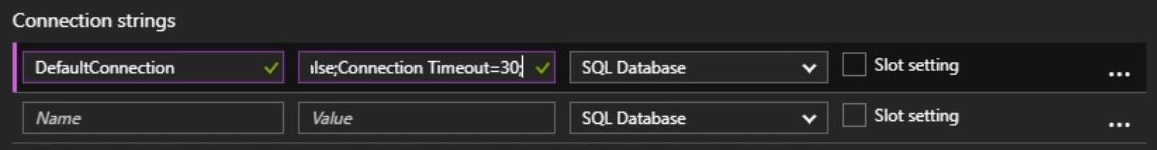
After that, create a new app service plan for the web app. Set the location to the Southeast Asia, and the pricing tier chosen will be Standard 1. The reason behind this is because Basic pricing tier does not support traffic manager, which is needed to be implemented afterward.



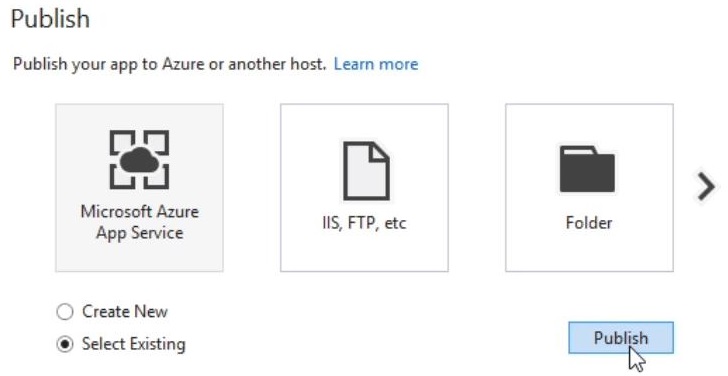
After that, go to the Visual Studio, set a connection to the SQL Database which has been created at the previous steps at Azure. The database at the moment will be empty because haven’t done the migration.

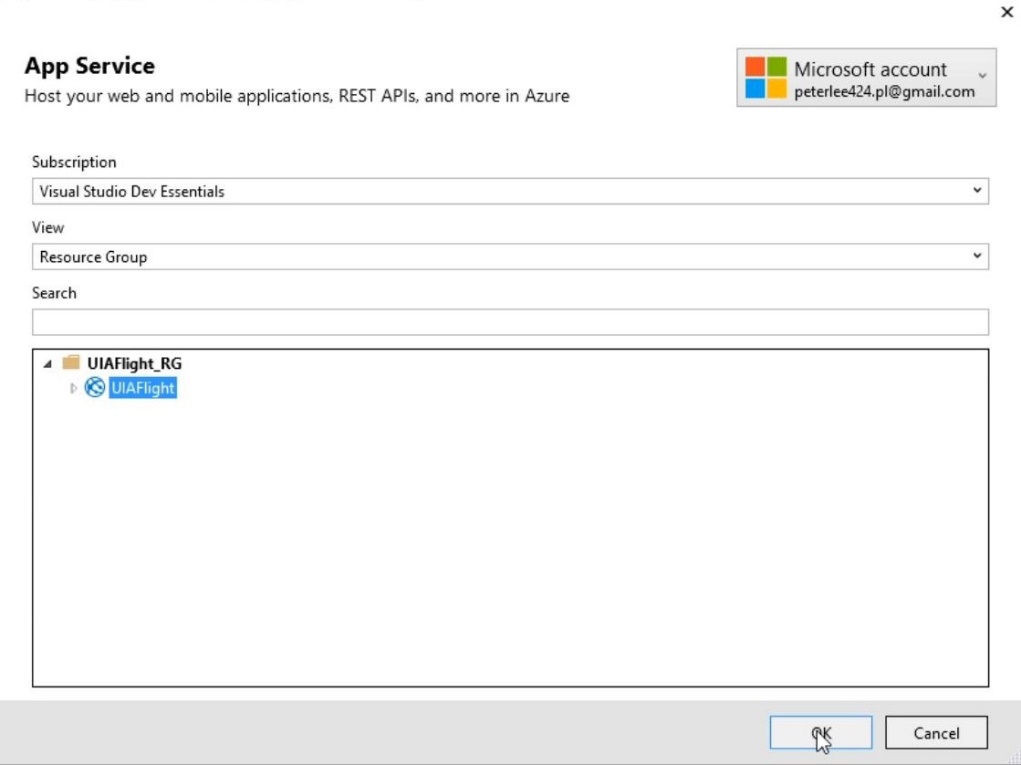






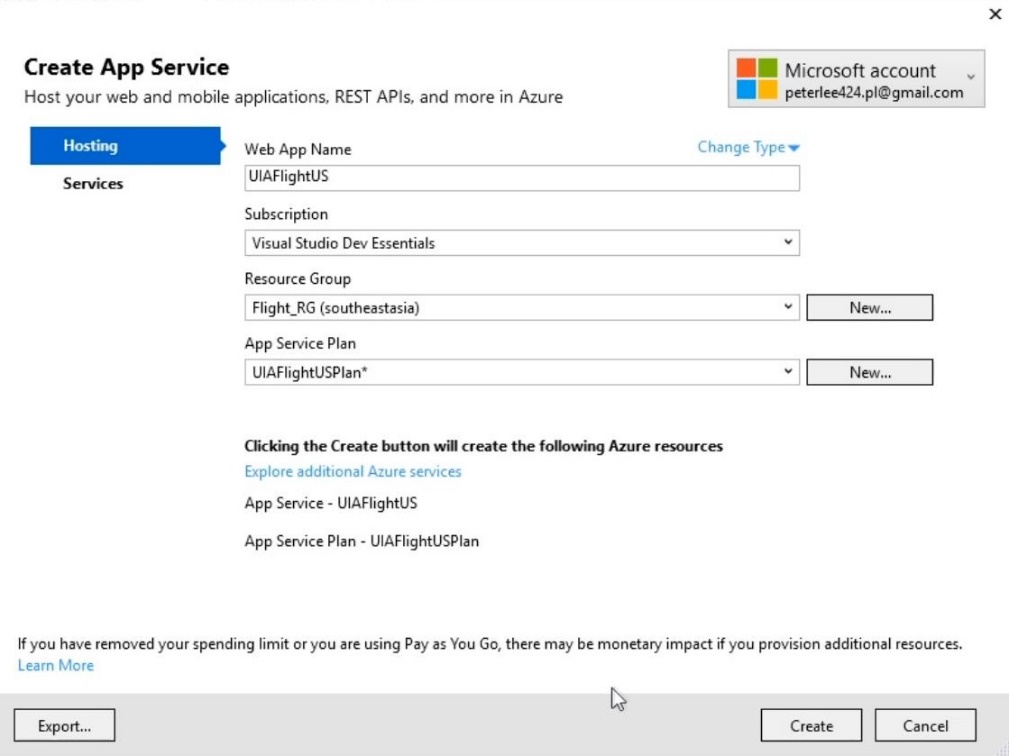
To set the connection string, go back to the Azure, and copy the pre-defined connection string, and paste it to the appsetting.json in the application. Make sure change the username and password to correct one. After that, go to the application setting of the app service in Azure, and add a new variable under Connection String named *DefaultConnection* and paste the copied string as value.



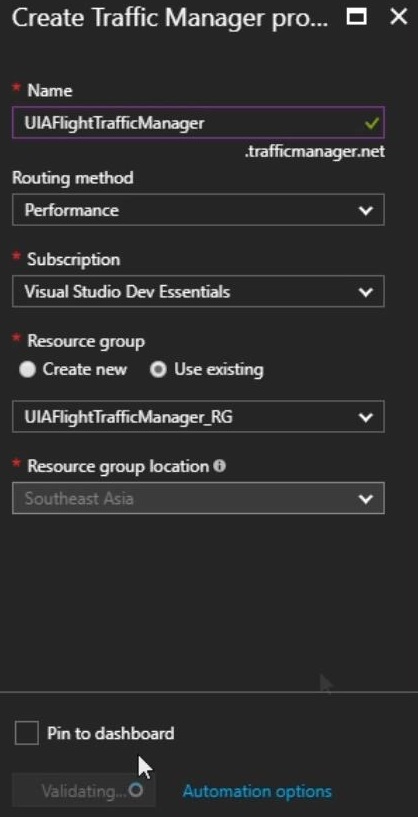


After setting the connection string, it’s time to publish the application to azure. Right click on the solution in Visual Studio, and click on publish. Select the *Select Existing* button, and publish it. Search the app service on the pop-up dialog and select it, click on OK button, then it will start to publish the application to the azure.

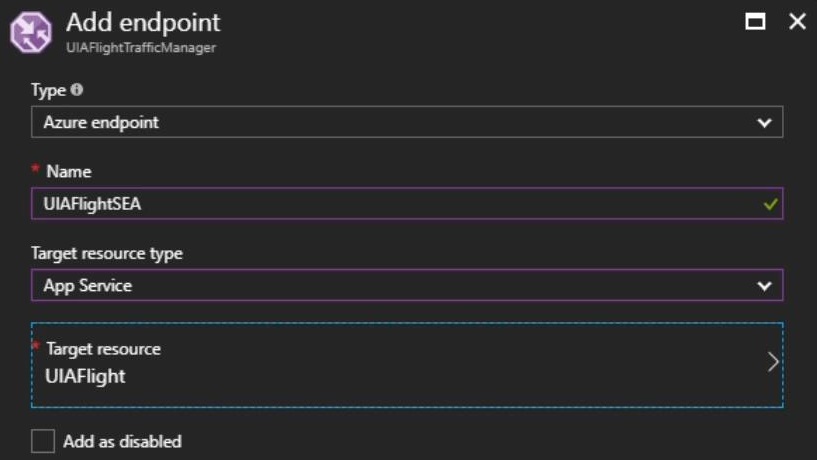
Traffic Manager can be implemented after finish the publishing of the application. Traffic Manager allows you to control the distribution of user traffic for service endpoints in different datacenters. Traffic Manager uses the Domain Name System (DNS) to direct client requests to the most appropriate endpoint based on a traffic-routing method and the health of the endpoints. Traffic Manager provides a range of traffic-routing methods and endpoint monitoring options to suit different application needs and automatic failover models (Microsoft, 2017).

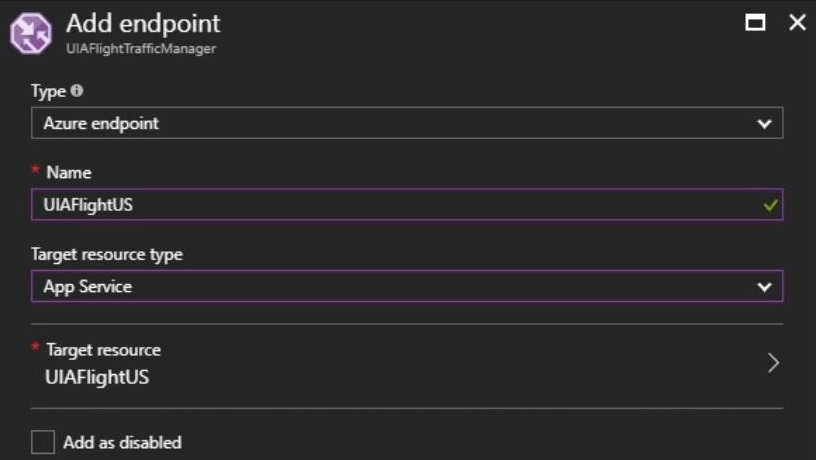


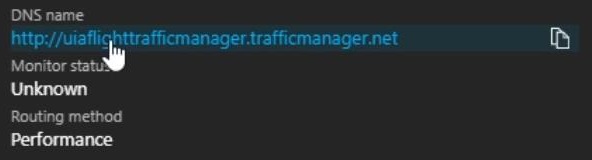
First of all, create a new profile for the web service. The App Service Plan have to be at least Standard Tier in order to support Traffic Manager.



After create the profile, go back to the Azure and create a new Traffic Manager Profile.



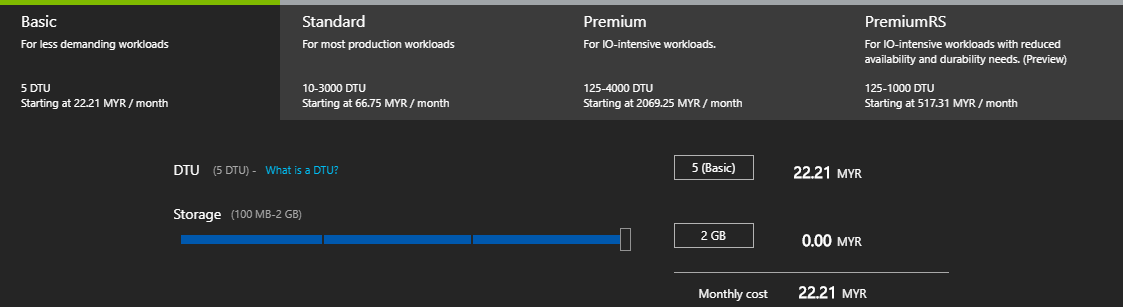




Then, add the endpoint into traffic manager. One will be the Southeast Asia, and another one will be US. Finally, use the newly generated link to access the website. It will redirect user to the nearest server by their location.

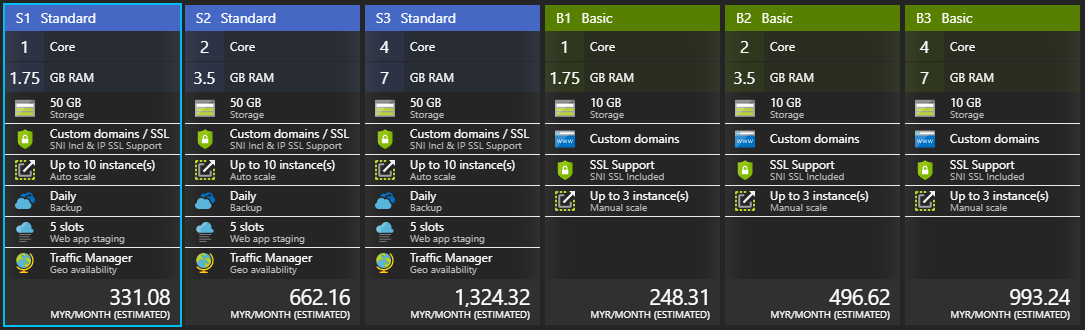
## Application Scaling with Justification

### SQL Database Pricing Tier



Pricing tier chosen for SQL Database was Basic Tier because the application does not require heavy workload at the moment. It provides 2GB of storage spaces per month and 5 DTU. The main reason of choosing this tier was the price, it only cost RM22.21 per month.

### App Service Pricing Tier



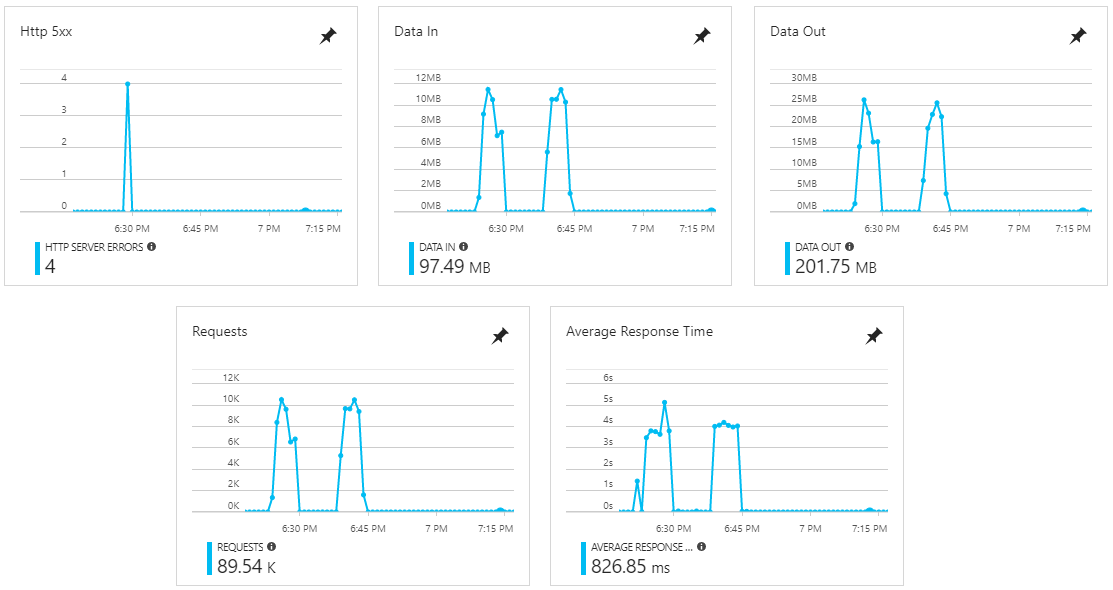
Pricing tier chosen for App Service was Standard 1. The reason behind this is although number of core and RAM is the same, but Basic tier does not support Traffic Manager, which need to be implemented to route incoming traffic for better performance and availability in terms of geographical. Besides, it has up to 10 instances standing by for scaling out. It also provides the daily backup feature to prevent the data loss causing by any possible events.

### Scaling

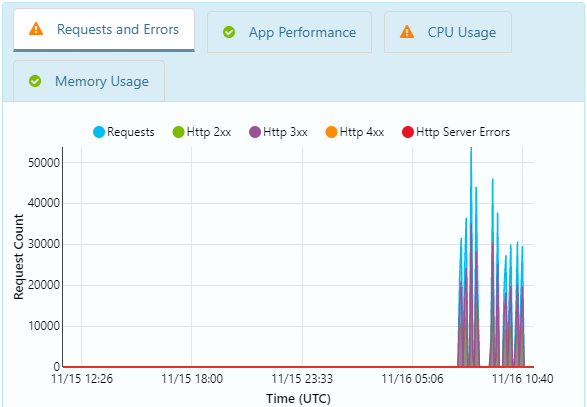
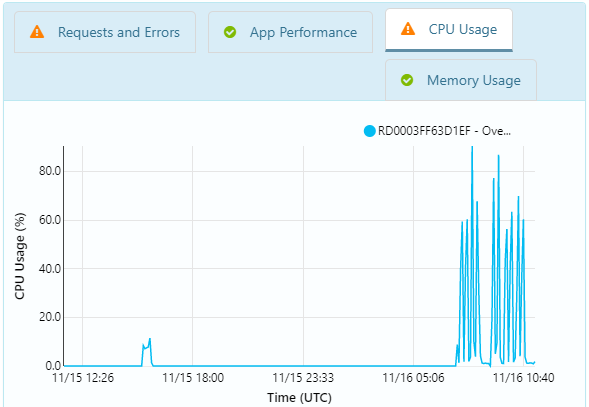
The pricing tier for SQL Database has a high possibility to be scaled up to Standard Tier in future to handle the growth of the business. The pricing tier of SQL Database at the moment is Basic Tier, although the price is affordable, but the services it provides is limited. It provides 2GB storage space per month, which may be insufficient in the future.

Besides, a scale out operation will be performed in app service to create multiple copies of web app and adding a load balancer to distribute the demand between them. This can be perform ideally as the pricing tier chosen was Standard 1 and it supports up to 10 instances for scale out.

## Investigate & Analyze Application

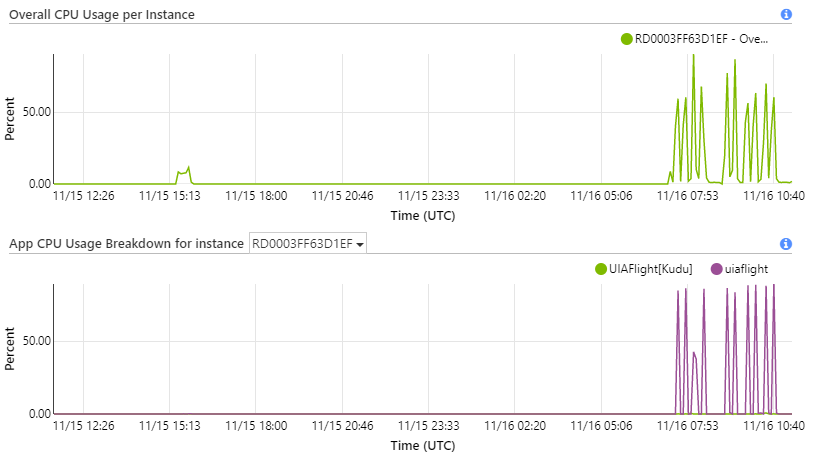


From the first metric, it shows that server has encountered total of 4 server error, which will causing server downtime. Second metric is the average incoming bandwidth used across all instances of the plan. Third is the average outgoing bandwidth used across all instances of the plan. Fourth is the total number of requests regardless of their resulting HTTP status code. The last one is the average time taken for the app to serve requests in millisecond (ms) (Tardif & Lin, 2016). The peaks in the metrics is because during that time the system is performing the performance testing.

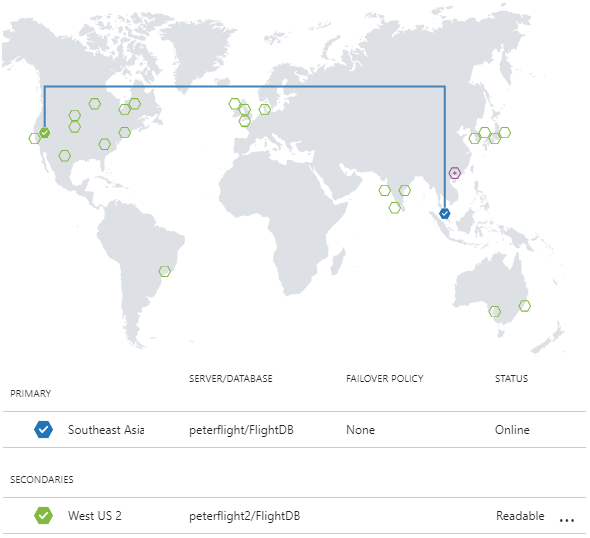
Azure provides diagnostic feature, it generates the web app status with metrics, analyze the data and suggest a solution if there is error. From the metrics above, the first one is showing that the web app is currently running healthy. However, it has detected downtime in the last 24 hours during which your Web App was experiencing errors. The observation is the app may have been impacted due to a site deployment. The solution suggested is review the latest code changes or configuration changes that may have contributed to these failures and take appropriate actions.

Besides, the second metric is showing that the Web App has experienced high CPU usage in the last 24 hours. CPU Usage went as high as 86.9% on that instance during this time.

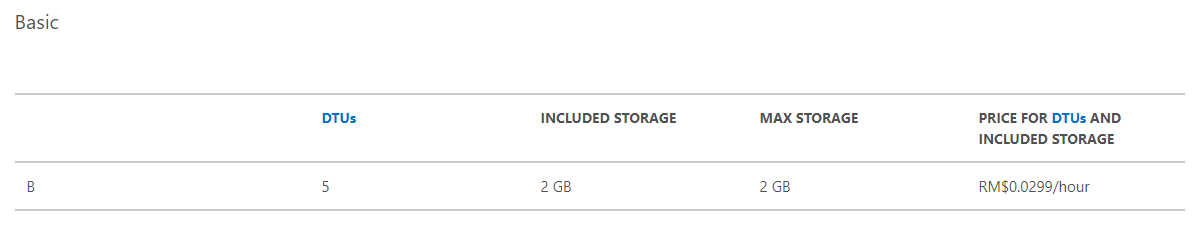


The solution suggested for the high CPU usage is to scale out the app service plan to add more instances. By scaling out, the workload will be distributed into multiple instances, which will resolve the problem of high CPU usage.

## Implementation & Discussion of Managed Databases



The databases used in this application is SQL Database. The region of primary database is in Southeast Asia which is the nearest region to here. Geo-replication is activated and the secondary databases at West US is available for querying and for failover if there is a data center outage or the inability to connect to the primary database. With geo-replication feature, the user can recover multiple related databases automatically in the secondary region after failures or other accidental events that result the loss of the service’s availability of the SQL Database in the primary region (Nosov, et al., 2017).



The pricing tier chosen for the both primary and secondary database is Basic Tier. Basic tier is targeted for light weight development and production. It comes with 2GB of storage spaces per month and 5 DTUs. It has the 7 days of backup retention. The I/O latency is higher than the Premium Tier but in the consideration of the cost, the basic tier is ideal for the web app at the moment. The cost of the basic tier is RM0.0299/hour which is RM22.21/month. (Microsoft, 2017) For the result of performance test, please refer to section 5.1.

# Test Plan & Testing Discussion

If a system not working as expected, user will get frustrated. The objective of this test plan to ensure that every single function of the web application is working well as expected and allow user have a good experience in interacting with the system.

Test Plan (?)

## Unit Testing

### Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Email** | **Password** | **Expected message** | **Actual Output** |
| 1 | Blank | Blank | Email is required  Password is required | Same as expected |
| 2 | <Email> | Blank | Password is required | Same as expected |
| 3 | Blank | <Password> | Email is required | Same as expected |
| 4 | <Email> | <Wrong Password> | Invalid login attempt. | Same as expected |

### Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Email** | **Password** | **Confirm Password** | **Expected message** | **Actual Output** |
| 1 | Blank | Blank | Blank | Email is required  Password is required | Same as expected |
| 2 | <Email> | Blank | Blank | Password is required | Same as expected |
| 3 | Blank | <Password> | Blank | The password and confirmation password do not match | Same as expected |
| 4 | <Email> | <Password> | <Random Characters> | The password and confirmation password do not match | Same as expected |

### Change Password

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Current Password** | **New Password** | **Confirm New Password** | **Expected message** | **Actual Output** |
| 1 | Blank | Blank | Blank | Current Password is required  New Password is required | Same as expected |
| 2 | <Password> | Blank | Blank | New Password is required | Same as expected |
| 3 | Blank | <Password> | Blank | Current Password is required | Same as expected |
| 4 | <Password> | <Password> | <Random Characters> | The new password and confirmation password do not match | Same as expected |

### Search Flight

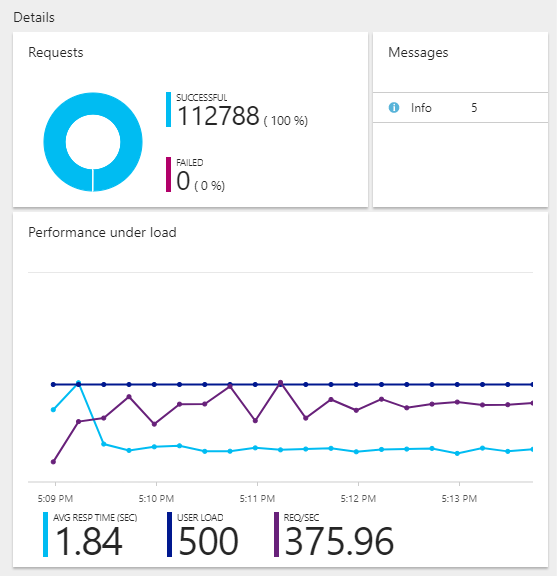
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Origin** | **Destination** | **Date** | **Expected message** | **Actual Output** |
| 1 | Blank | Blank | Blank | Origin is required  Destination is required  Date is required | Same as expected |
| 2 | <Origin> | Blank | Blank | Destination is required  Date is required | Same as expected |
| 3 | <Origin> | <Destination> | Blank | Date is required | Same as expected |

### Booking

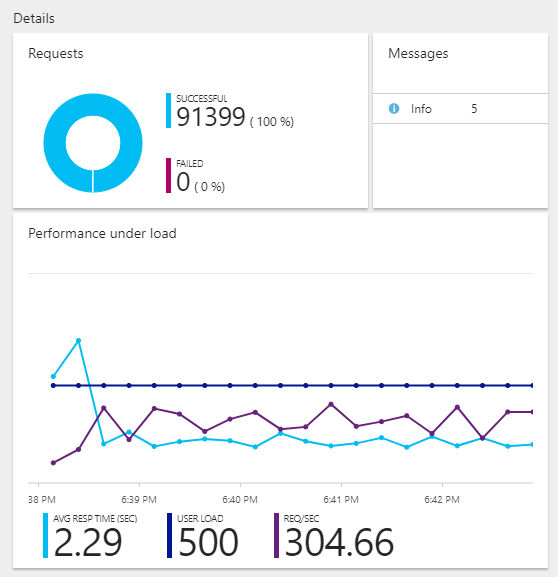
|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Expected message** | **Actual Output** |
| 1 | All Blank | Each field has respective message to fill out text field | Same as expected |
| 2 | Provide invalid phone number | Phone number is invalid | Same as expected |
| 3 | Provide invalid characters in name | Name is invalid | Same as expected |

## Performance Testing

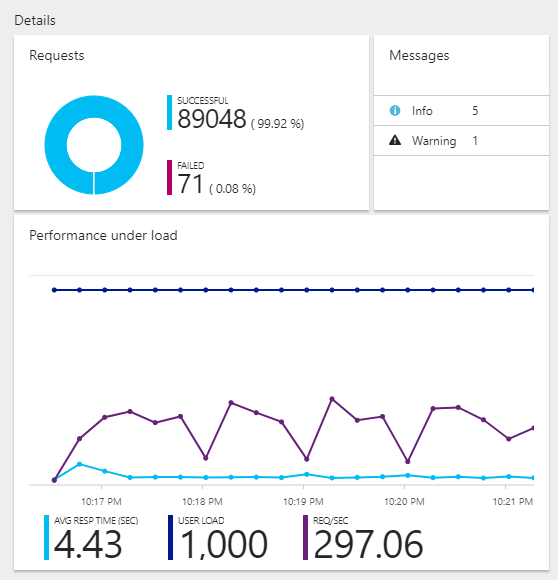
Azure provides performance testing which allow developer to check their web app's performance before you launch it or deploy updates to production. There is 2 sets of test was conducted, first set is test the home page, another is test the search flight page, and each of the tests was conducted with 100, 250, 350, 450, and 500 users attempt to access the web app at the same time within 5 minutes of time period.



The result above is the performance testing of 500 dummy users send the request to access the home page. Total 112,788 of request was received and all of them was handled properly. In this case, the test result is perfect as there is no any server error occur and no single request that was failed to handle. The average respond time of the web app under the pressure given was 1.84 seconds, which is a good result. There are 375.96 requests was being handled per seconds.



The result above is the performance testing of 500 dummy users send the request to access the search flight page with random search request. Total 91,399 of request was received and all of them was handled properly. In this case, the test result is perfect as there is no any server error occur and no single request that was failed to handle. The average respond time of the web app under the pressure given was 2.29 seconds, which still is a good result. There are 304.66 requests was being handled per seconds.



Since the testing results are all good and satisfy, an extra testing was conducted to test the reliability of the web app, which is 1,000 users send the request to access the search flight page with random search request. Image above is the result. Total 89,119 of request was received and 89,048 (99.92%) of them was handled properly, another 71 (0.08%) of requests failed. In this case, the test result is still very good as with such high amount of user’s attempts, the service still able to handle it with 99.92% of success. The average respond time of the web app under the pressure given was 4.43 seconds, which is an acceptable result. There are 297.06 requests was being handled per seconds.

Through the testing, the result is the good evidence of pricing tier chosen is suitable for this web app at the moment. Although the DTU of the SQL Database only 5, but it still able to handle the request perfectly.

# Conclusion

In conclusion, Azure enables organization to get their apps to market faster, it has wide extensive of tools and open source technologies supported. Azure also provides Platform as a service (PaaS), which enable developer to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications (Microsoft, 2017). Besides, active geo-replication enables developer to configure multiple readable secondary databases in the same or different data center locations to stand by if primary database is down. Traffic manager is also a great feature provided by Azure which enables the app service auto reroute the user to the most ideal server based on their location. The web content also can be customized based on customer’s location.

# References

Dwivedi, A., n.d. *7 Reasons Why Azure Is Better Than AWS.* [Online]   
Available at: http://www.saviantconsulting.com/blog/7-reasons-why-azure-is-better-than-AWS.aspx  
[Accessed 17 11 2017].

Microsoft, 2017. *App Service pricing.* [Online]   
Available at: https://azure.microsoft.com/en-us/pricing/details/app-service/  
[Accessed 15 11 2017].

Microsoft, 2017. *Overview of Traffic Manager.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview  
[Accessed 15 11 2017].

Microsoft, 2017. *SQL Database pricing.* [Online]   
Available at: https://azure.microsoft.com/en-us/pricing/details/sql-database/  
[Accessed 15 11 2017].

Microsoft, 2017. *What is Azure?.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-azure/  
[Accessed 16 11 2017].

Microsoft, 2017. *What is PaaS?.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/  
[Accessed 16 11 2017].

Nosov, A., Rabeler, C. & Stein, S., 2017. *Overview: Failover groups and active geo-replication.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-geo-replication-overview  
[Accessed 15 11 2017].

Rabeler, C. & Kerkhove, T., 2017. *Database Transaction Units (DTUs) and elastic Database Transaction Units (eDTUs).* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-what-is-a-dtu  
[Accessed 15 11 2017].

Roth, D., Anderson, R. & Luttin, S., 2017. *Introduction to ASP.NET Core.* [Online]   
Available at: https://docs.microsoft.com/en-us/aspnet/core/  
[Accessed 15 11 2017].

Tardif, B., 2013. *Scaling Up and Scaling Out in Windows Azure Web Sites.* [Online]   
Available at: https://azure.microsoft.com/en-us/blog/scaling-up-and-scaling-out-in-windows-azure-web-sites/  
[Accessed 15 11 2017].

Tardif, B. & Lin, C., 2016. *How to: Monitor Apps in Azure App Service.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/app-service/web-sites-monitor  
[Accessed 15 11 2017].

# Appendix

GitHub Link: https://github.com/peter0424/Azure

Stream Link: https://web.microsoftstream.com/video/af083dac-a7ae-4180-ba91-21144502a891