 Designing and Developing Cloud Applications

Name : Teoh Ghee Yuen

ID number : TP033750

Intake Code : UC3F1702SE

Course Code : CT071-3-3-DDAC

Lecturer Name : Dr. Kalai Anand Ratnam

Due Date : 20th November 2017

# Acknowledgement

First of all, I would like to thank my DDAC Lecturer Dr.Kalai for guiding me throughout this semester for the DDAC module and for this project. The information and materials provided during lectures and tutorials has been proven to be quite impactful towards the outcome of this project and I am immensely grateful for that. Other than that, I would like to thank my friends for supporting me throughout the development of this project, without their support this project would not have been as successful as it is right now.

# Table of Contents

[Acknowledgement 1](#_Toc498932736)

[Table of Contents 2](#_Toc498932737)

[1.0 Introduction 4](#_Toc498932738)

[1.1 Project Background 4](#_Toc498932739)

[1.2 Aims 4](#_Toc498932740)

[1.3 Objectives 4](#_Toc498932741)

[1.4 System Functionality 5](#_Toc498932742)

[2.0 Project Plan 6](#_Toc498932743)

[3.0 Design 7](#_Toc498932744)

[3.1 Architectural Diagram 7](#_Toc498932745)

[3.2 Design Considerations 8](#_Toc498932746)

[3.3 Application Design and Modelling 8](#_Toc498932747)

[3.3.1 Use Case Diagram 8](#_Toc498932748)

[3.3.2 Data Modelling 9](#_Toc498932749)

[3.3.3 Sequence Diagram 9](#_Toc498932750)

[3.3.4 User Interface 12](#_Toc498932751)

[3.3.5 Sitemap 15](#_Toc498932752)

[4.0 Implementation 16](#_Toc498932753)

[4.1 Application Development 16](#_Toc498932754)

[4.2 Cloud Deployment 18](#_Toc498932755)

[4.2.1 Publishing Application to Azure 18](#_Toc498932756)

[4.2.2 SQL Server 21](#_Toc498932757)

[4.2.3 Azure Traffic manager 24](#_Toc498932758)

[4.2.3 Application Scaling 26](#_Toc498932759)

[4.2.4 Investigate & Analyse Application 28](#_Toc498932760)

[4.3 Discussion of Managed Database 32](#_Toc498932761)

[5.0 Testing 34](#_Toc498932762)

[5.1 Unit Testing 34](#_Toc498932763)

[5.2 Performance Testing 36](#_Toc498932764)

[6.0 Conclusion 38](#_Toc498932765)

[7.0 References 39](#_Toc498932766)

[8.0 Appendix 41](#_Toc498932767)

# 1.0 Introduction

## 1.1 Project Background

Ukraine International Airlines (UIA) is a flagship carrier and the largest airline in Ukraine. The company operates domestic and international passenger flights and cargo services to Europe, the Middle East, the United States and Asia. Recently they have been looking at designing and developing an Online Flight Booking System in order to expand into newer markets as its current website prevented it from adequately serving customers beyond its home country Ukraine. It has been decided by UIA that innovation is required to overcome its web challenges and they have arrived at a conclusion to migrate the website out of UIA data-centres into a public cloud that could solve most of their problems.

The two main cloud service providers looked at by UIA are Microsoft Azure and Amazon Web Services, ultimately Azure was chosen as it was very compatible with open source software. The Online Flight Booking System is to be able to handle standard booking processes for the airline such as searching for flights, booking seats for a flight, creating user profiles for booking and managing the profile. The application is to be hosted in Azure along with utilising its cloud database services for the application. Performance and availability concerns must also be address through the application and cloud solutions as it will be crucial to the applications success.

## 1.2 Aims

To design and create an online flight booking application for Ukraine Airlines that is hosted on the Azure cloud service and consumes a cloud database service for its processes.

## 1.3 Objectives

- To create a flight booking application

- To host the application on cloud

- To use a cloud database for the application

- To have geographical routing to cater to customers outside of Ukraine

## 1.4 System Functionality

Below list the functionalities planned to be developed for the online flight booking application:

- Register user account

- Login with user account

- Change password

- Change profile information

- Search flights

- Book seats for flights

- View booked flights for user

# 2.0 Project Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WBS | Task Name | Duration | Start | Finish |
| **1** | **System Planning** | **17 days** | **Mon 17/7/17** | **Tue 8/8/17** |
| 1.1 | Preparation for requirements gathering | 1 day | Mon 17/7/17 | Mon 17/7/17 |
| 1.2 | Conduct interview with UIA staff members | 2 days | Tue 18/7/17 | Wed 19/7/17 |
| 1.3 | Conduct survey wit UIA customers | 5 days | Tue 18/7/17 | Mon 24/7/17 |
| 1.4 | Finalize gathered results | 2 days | Tue 25/7/17 | Wed 26/7/17 |
| 1.5 | Analysis of requirement gathering | 3 days | Thu 27/7/17 | Mon 31/7/17 |
| 1.6 | Produce system requirements specification | 3 days | Tue 1/8/17 | Thu 3/8/17 |
| 1.7 | Finalize system requirements specification | 1 day | Fri 4/8/17 | Fri 4/8/17 |
| 1.8 | Develop project schedule | 2 days | Mon 7/8/17 | Tue 8/8/17 |
| **2** | **System Design** | **9 days** | **Wed 9/8/17** | **Mon 21/8/17** |
| 2.1 | System Logical Design | 2 days | Wed 9/8/17 | Thu 10/8/17 |
| 2.2 | System Physical Design | 2 days | Fri 11/8/17 | Mon 14/8/17 |
| 2.3 | System Database Design | 2 days | Tue 15/8/17 | Wed 16/8/17 |
| 2.4 | Finalize System design and documentation | 3 days | Thu 17/8/17 | Mon 21/8/17 |
| **3** | **System Development** | **21 days** | **Tue 22/8/17** | **Tue 19/9/17** |
| 3.1 | Develop Prototype | 5 days | Tue 22/8/17 | Mon 28/8/17 |
| 3.2 | System Review and Enhancement 1 | 2 days | Tue 29/8/17 | Wed 30/8/17 |
| 3.3 | Develop System Modules | 10 days | Thu 31/8/17 | Wed 13/9/17 |
| 3.4 | System Review and Enhancement 2 | 2 days | Thu 14/9/17 | Fri 15/9/17 |
| 3.5 | Finalize System Development | 2 days | Mon 18/9/17 | Tue 19/9/17 |
| **4** | **System Testing** | **14 days** | **Wed 20/9/17** | **Mon 9/10/17** |
| 4.1 | Conduct Integration Testing | 5 days | Wed 20/9/17 | Tue 26/9/17 |
| 4.2 | Fix Defects and Bugs 1 | 3 days | Wed 27/9/17 | Fri 29/9/17 |
| 4.3 | Conduct User Acceptance Testing | 3 days | Mon 2/10/17 | Wed 4/10/17 |
| 4.4 | Fix Defects and Bugs 2 | 2 days | Thu 5/10/17 | Fri 6/10/17 |
| 4.5 | Finalize System Testing | 1 day | Mon 9/10/17 | Mon 9/10/17 |
| **5** | **System Deployment** | **8 days** | **Tue 10/10/17** | **Thu 19/10/17** |
| 5.1 | Deploy System to cloud | 3 days | Tue 10/10/17 | Thu 12/10/17 |
| 5.2 | Monitor and maintain system | 5 days | Fri 13/10/17 | Thu 19/10/17 |

# 3.0 Design

## C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Architecture.png3.1 Architectural Diagram

Above shows the cloud architectural diagram planned for the online flight booking application. Two web app services are planned to be used both in different geographical locations, whereby the traffic manager will redirect users to their region-specific web applications. Both application however will consume data from the same SQL Database located in the SEA region, whereby it is backed up by a secondary database located in the US region through geo-replication.

## 3.2 Design Considerations

The design of the application is done with several constraints and assumptions done. One of the main considerations of the application design is the RM150 per month of Azure credit constraint given by UIA to develop a fully functioning online flight booking system hosted in Azure. All functionalities stated in **Section 1.4** will be considered during application design. Cloud functionalities provided by azure to be used as shown in the previous section will also be considered during application design such as App Service Web App, SQL Database and Traffic manager.

## 3.3 Application Design and Modelling

### C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\UseCase.png3.3.1 Use Case Diagram

### C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\ERD.png3.3.2 Data Modelling

### C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Sequence\Login.png3.3.3 Sequence Diagram

C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Sequence\SearchFlight.pngC:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Sequence\Register.png

C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Sequence\View Booking.pngC:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\Sequence\BookFlight.png

### 3.3.4 User Interface

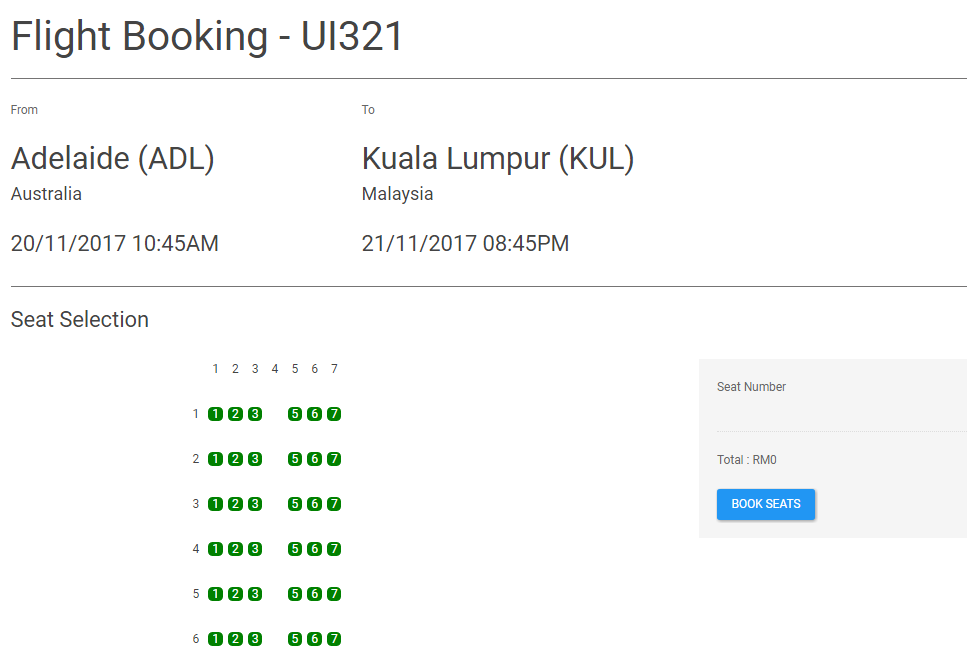
#### 3.3.4.1 Home

#### 3.3.4.2 Register

#### 3.3.4.3 Login

#### 3.3.4.4 Search Flight

#### 3.3.4.5 Book Flight



#### 3.3.4.6 Profile

#### 3.3.4.7 Change Password

#### 3.3.4.8 Edit Information

#### 3.3.4.9 View Bookings

### 3.3.5 Sitemap

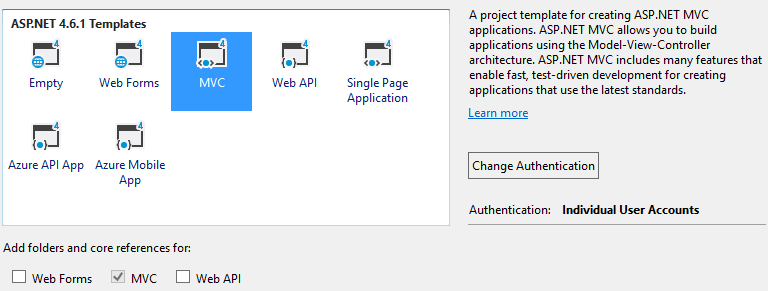
C:\Users\Teoh\Desktop\GY\Degree LVL3\LVL2\DDAC\Assignment\SiteMap.png

Above shows the sitemap for the web application whereby the default page would be the home page linking to login, register and search flights. Change password, view bookings and manage profile information page will be available after login. Users will also be able to book flights only after searching flights.

# 4.0 Implementation

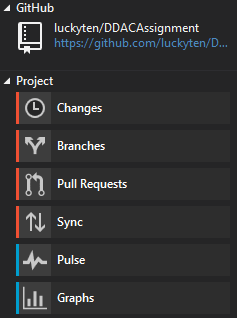
## 4.1 Application Development

The online flight booking system is developed using C# and the visual studio 2017 IDE. The application is created as an Asp.Net Web Application using the .NET Framework.

The application is developed upon a base MVC template given by visual studio along with a built in basic authentication functionality for user accounts.

The reason why the web application is developed using the MVC (Model-View-Controller) architectural pattern is because that MVC pattern provides a better framework for faster development as it separates the application into three distinct layers called Model, View and Controller whereby each layer can be developed separately and managed for efficiently (BrainVire, 2016). The ASP.NET MVC framework provided by visual studio is a lightweight and highly testable presentation framework that is widely used in the market that is easy to use for development and testing which is suitable for this project (MSDN, n.d).

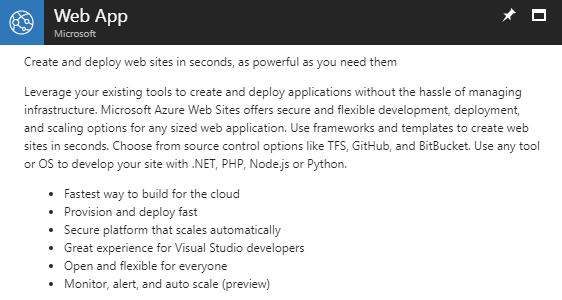
As for data modelling and data connection the application will be using entity framework code-first whereby it allows the developer to design the domain classes as models in C# code and then the database tables and relationships are auto generated based on the classes created (EntityFrameworkTutorial, n.d).

The application will also be using GitHub as its source control repository. Whereby GitHub is an online source control repository that helps with the version control of the application as well as storing revisions of the projects (Finley, 2012). A GitHub extension for visual studio is used to further simply the process of pulling and pushing changes to the repository.

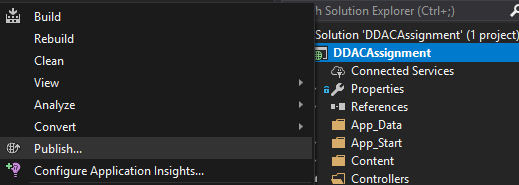
The full application and its change history will be available on the GitHub repository at the hyperlink: <https://github.com/luckyten/DDACAssignment> .

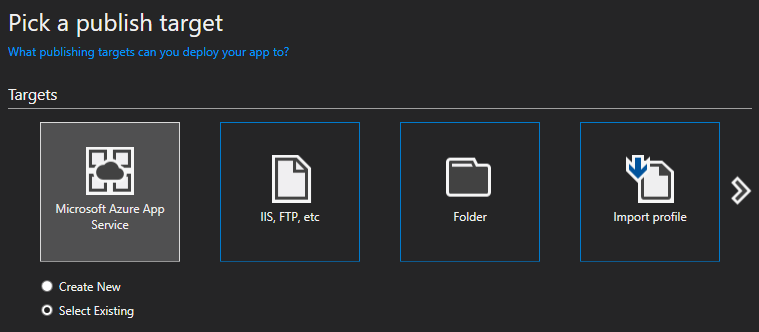
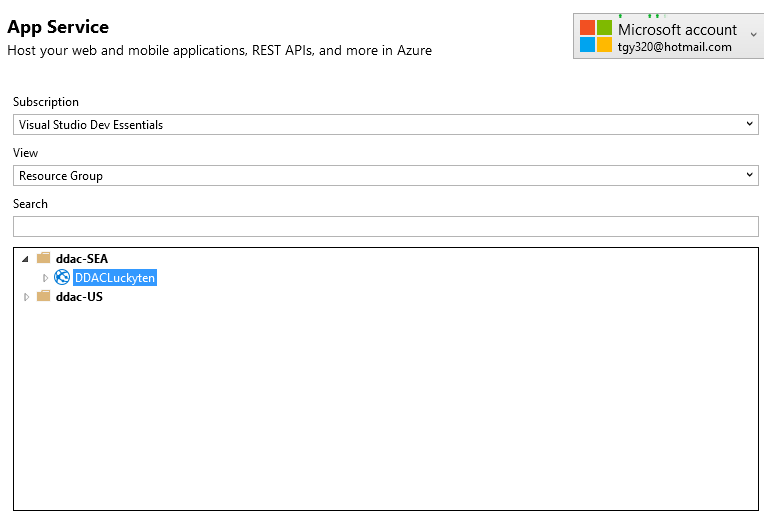
## 4.2 Cloud Deployment

### 4.2.1 Publishing Application to Azure

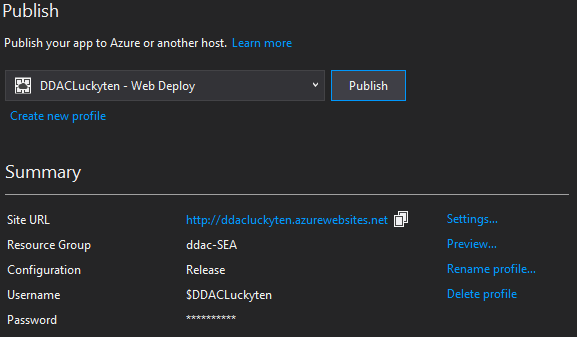
The application is published to Azure as a web application whereby a web app service is created and used in Azure. The Azure Web App is a service that hosts web applications that is developed in many different languages such as .NET, Java, Ruby, Node.js, PHP or Python and providing many other different services to the published application such as load balancing and auto-scaling (Lin, 2017).

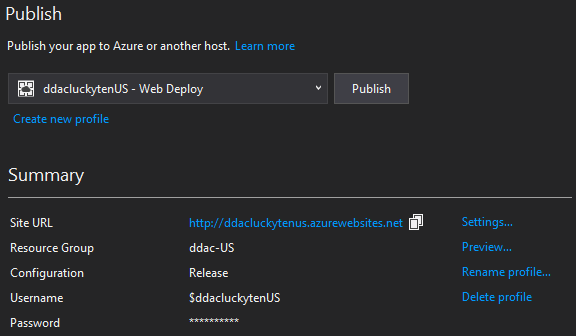
Two web app services will be created and published to whereby both web app services will be for different regions as depicted in the cloud architectural design in **Section 3.1.**

Once the web application service is created on azure the application is published through the visual studio IDE.

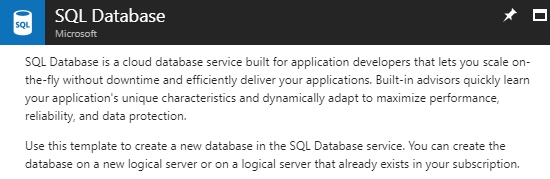
When publishing the application the Microsoft azure App Service is selected and an existing app service will be selected as seen below.

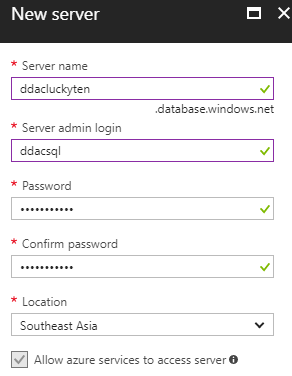
For the above screenshot the web application service to be published to will be the web application for the SEA region.

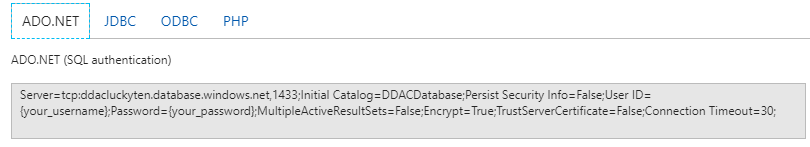
Once the web app service is selected the application is now ready to be published to the Azure web app service and to be consumed by public users.

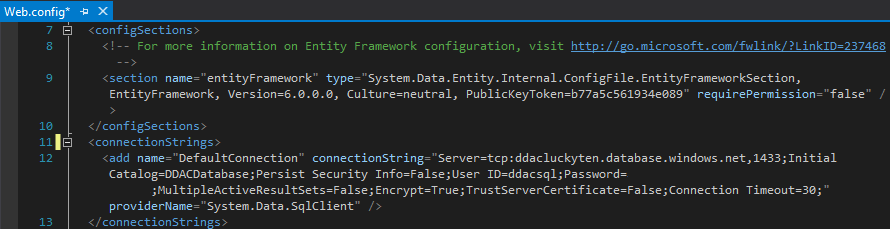
After successful deployment of the web application for the SEA region the similar process is repeated to publish another application for the US region by publishing it to the second web application service created for the US.

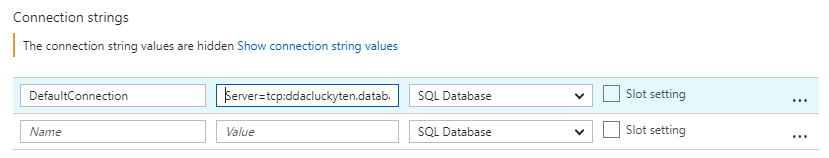
### 4.2.2 SQL Server

Once the application is published and ready for viewing to the public users it is still unable to perform its core functionalities as a database is yet to be linked to the application. The application will be using an SQL database service which is a cloud database service provided by Azure.

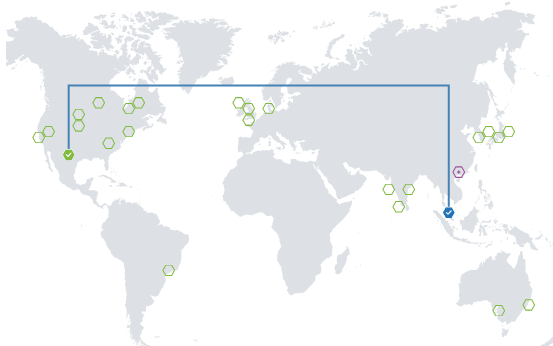
In order to create an SQL database a SQL server must be created alongside to host the database, whereby a username and password is provided that will be used by the application later on to connect to the database

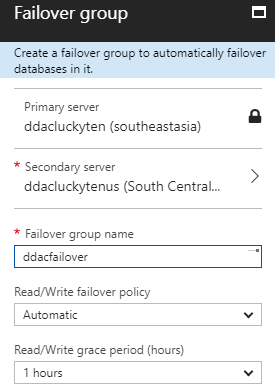
Once the SQL database and SQL server are created the next step is to obtain the connection string for the database.

The connection string with replaced username and password is then copied and pasted into the Web.config file of the application that stores the connection string for the application.

The connection string is also updated in the web application settings in azure.

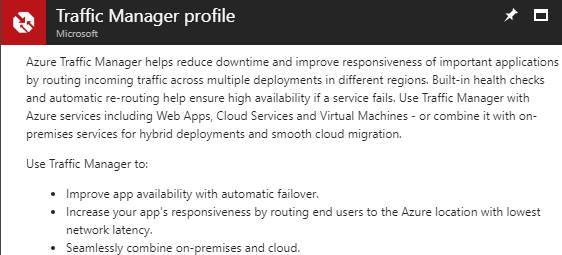
Once all the connection strings are updated the solution is published again to ensure the new changes are published to the live version of the application. Now the online flight booking system is able to perform functions linking to the database such as register, login, search and book flights.

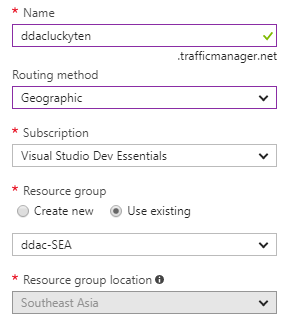
To ensure that the website is functional even in the event of a database failure a secondary database is configured through geo-replication on azure.

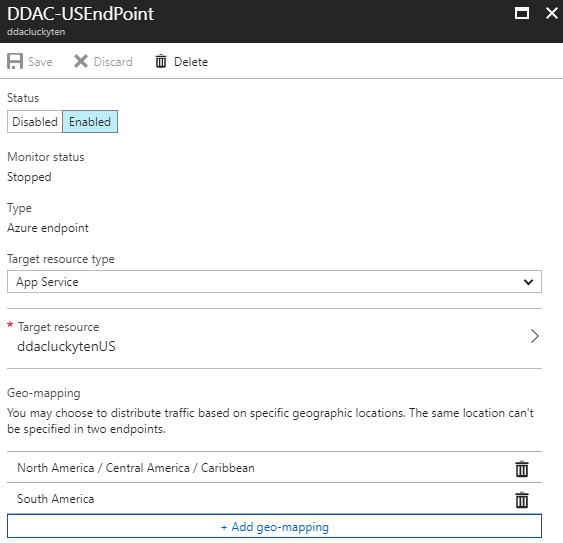
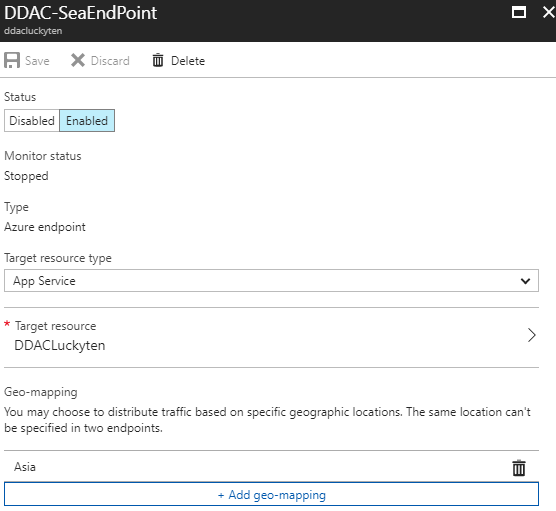
Geo replication is done on the primary SQL server located in the SEA region and replicated to a secondary SQL server located in the US region as shown above. Once the secondary database is deployed a failover rule is setup to ensure that when the primary database fails the secondary database comes online to replace it.

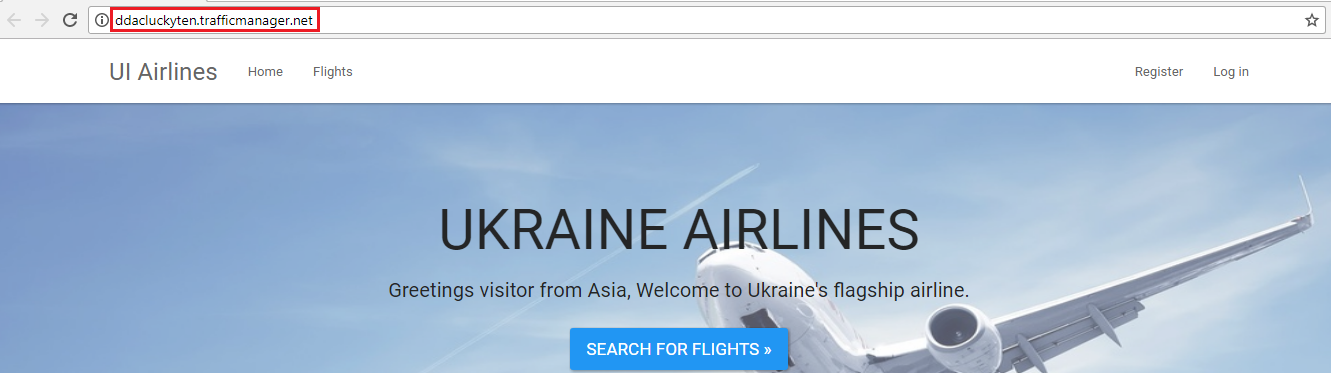
Now the database used by the system is reliable with a secondary failover option in the event of a primary database failure.

### 4.2.3 Azure Traffic manager

Once both the web applications in Section 4.2.1 is set up and fully functional a mechanism to redirect users from different region to their regional specific web application is required. This is where Azure’s traffic manager service come in whereby it allows us to control the distribution of user traffic for service endpoints by using the DNS (Domain Name System) to direct client requests to the most appropriate endpoint based on a set traffic routing method (Microsoft1, 2017).

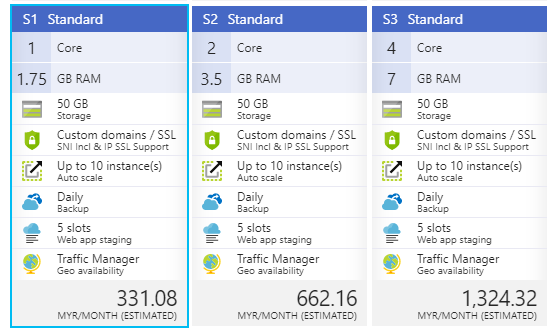
First a traffic manager profile is created on azure with the routing method for the traffic manager set to geographic instead of performance to redirect the users by their visiting geographic location.

Once the traffic manager profile is created two endpoints for both the app services is created along with their corresponding geographical location to be redirection from.

Once both endpoints are created, and set-up users will be able to visit the website through the traffic manager DNS to be routed to the regional websites instead of referring the web app services direct URL. For example in this case a user will be able to access the application through <http://ddacluckyten.trafficmanager.net/> whereby the traffic manager will redirect users to different endpoints of the application such as the SEA end point <http://ddacluckyten.azurewebsites.net> if the user is from the Asia region or the US endpoint <http://ddacluckytenus.azurewebsites.net/> for users from the US Region.

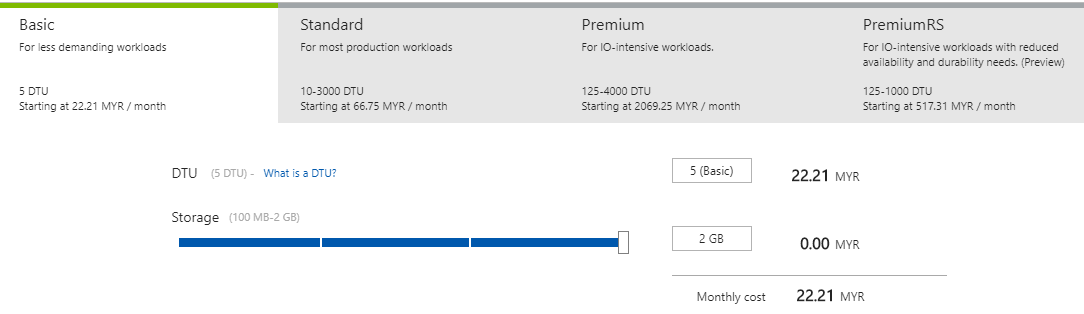
### 4.2.3 Application Scaling

#### 4.2.3.1 Application Pricing Tier

For Azure web applications the pricing of the web application is based off the app service plan of the web application. For both the web application for SEA and the US both will be using an app service plan using the pricing tier S1 – Standard as shown below.

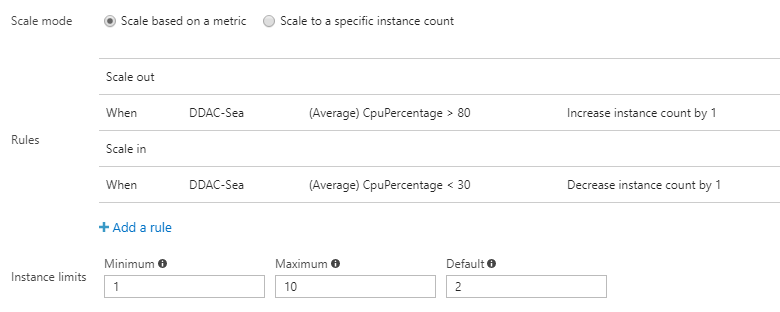
The reason why S1 Standard is chosen for both the web application’s is mainly due to the RM150.00 Azure credit budget limitation. The S1 Standard is the cheapest tier that provides the traffic manager functionality which is crucial to UIA’s success of improving its application performance and availability outside of its main country Ukraine. S1 Standard is also the cheapest tier that provides the auto-scaling functionality for instances which will be discussed further on in **Section 4.2.3.2**. The current plan also provides daily backup service which improves the reliability of the system in the case of a failure.

But the pricing tier used currently is the minimum pricing tier to be used by UIA as it is the cheapest tier that can satisfy the requirements of UIA as stated before. However if UIA’s market expansion in either the SEA or US region is successful and traffic increases up to a certain degree that the current pricing tier is unable to perform optimally it can always **Scale-Up** to either S2 Standard or S3 Standard depending on the performance requirement in the future. For now, S1 Standard is sufficient to provide users with optimal performance in the SEA and US region.

As for SQL Databases the pricing tier chosen for both the primary and secondary databases will be the Basic pricing tier with 5 DTU. In which DTU is called a database transaction unit that is a blended measure of CPU, memory and data/transaction log I/O whereby this benchmark measurement is provided by Microsoft (Microsoft2, 2017). DTU is the relative amount of resources for Azure SQL databases at different pricing tiers, whereby doubling the DTU equates to doubling the set of resource and performance level of a database (Microsoft2, 2017).

As for UIA the basic tier of database is sufficient as there is currently not much users from both the SEA and US region combined and the data storage of 2 GB is also sufficient to store all of UIA’s operational data as of now.

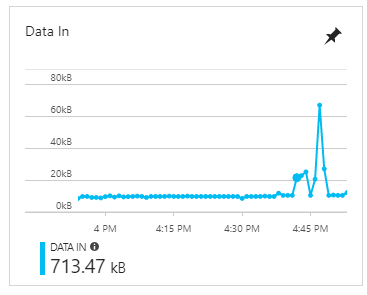
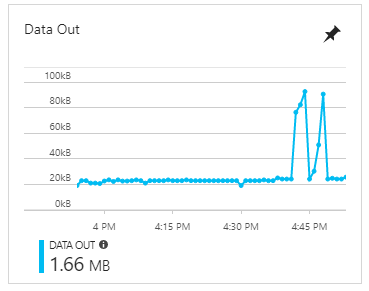
#### 4.2.3.2 Application Auto-scaling

As stated before in **Section 4.2.3.1** the pricing tier of S1 Standard it allows the use of the auto-scaling functionality for the web applications. Whereby it allows the automatic scaling in and scaling out of instances used by the web applications (Microsoft3, 2017). Automatic scaling is done for the online flight booking application web applications using the rules as shown below.

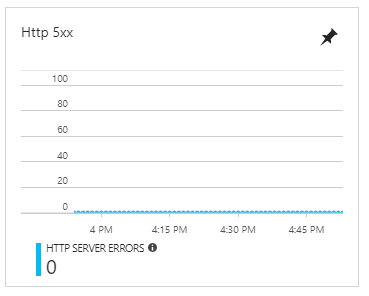
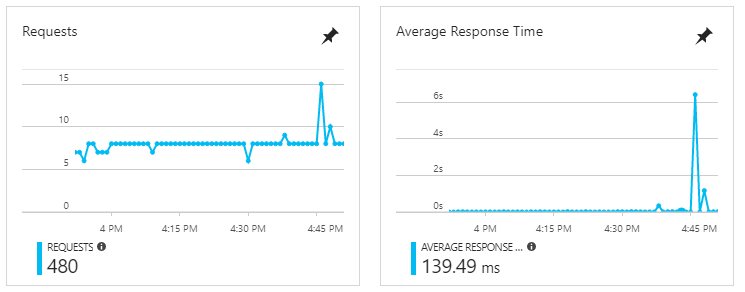
By default, 2 instances will be used for the web applications. If the average CPU usage exceeds by 80% an extra instance count will be added and if the average CPU usage is below 30% the amount of instance being used will be reduced. This allows the application to scale the number of instances being used by the web application according to its current CPU usage.

### 4.2.4 Investigate & Analyse Application

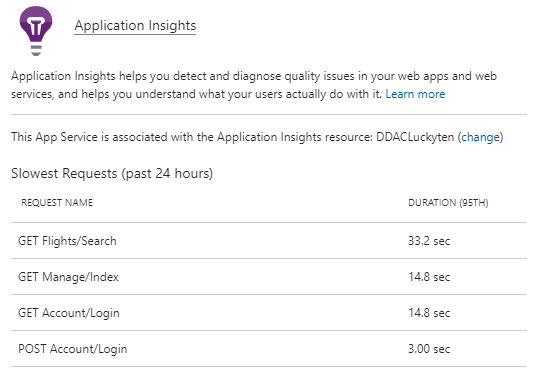
Once the online flight booking application is up and running on Azure there are many statistics and metrics provided for us to analyse the application either through the web application page or through application insights.

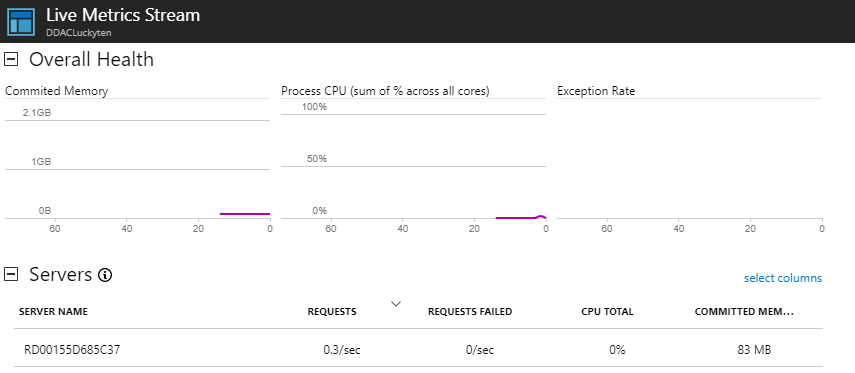
First off from the overview page for the web application there are a few metrics that can help analyse the application.

The data in and data out metric represents the amount of incoming and outgoing bandwidth used across all instances of the web application (Microsoft4, 2016). From this metrics it can be seen how data heavy the application is and infer if tuning is required to reduce the amount of data transferred between users and the application and from which direction. Either through reducing the amount of GET/POST data to reduce data coming into the application or reducing the reducing the size of stylesheets and scripts through minifying them to reduce data going out to the user.

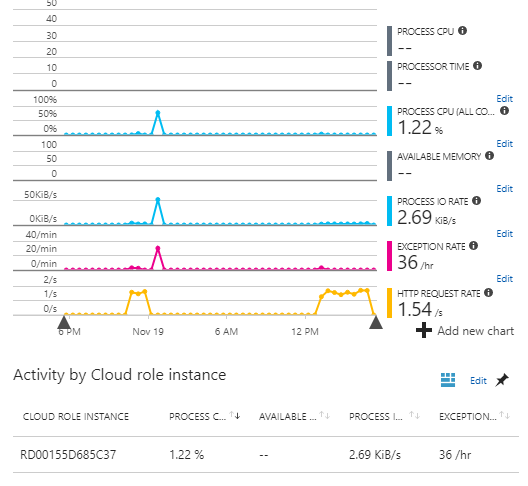
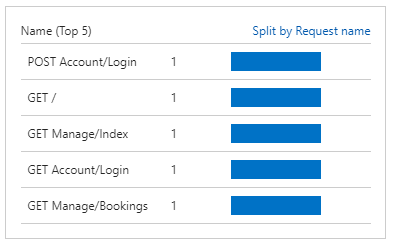
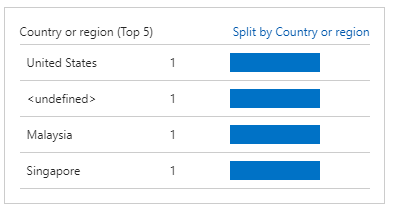
The http server errors metrics show the amount of errors hit by the web application that can be categorized through the http error code classification such as 3xx, 4xx and 5xx to allow the developers to pinpoint the source and type of errors frequently occurring from the web application. For example, a high amount of 4xx type error means there is a large amount of client-side errors being hit and 5xx type error means there is a large amount of server-side errors being hit then the developer can investigate and fix according to these statistics.

The last metric that can be seen from the overview page is the amount of requests and the average response time for the requests over a period of time. From these metrics it is possible to judge if the current service plan is sufficient to serve a high amount of requests yet maintaining a stable and low average response time especially during service peak hours.

There are many more metrics that can be found for the application outside of the overview and in the application insights service provided by Azure which gives insights of the application through application performance management and instant analytics.

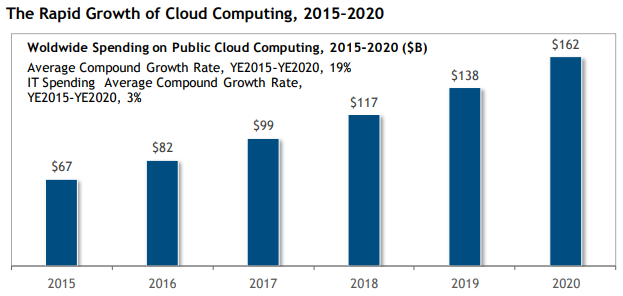
For example, application insight can show the slowest request within a certain period as seen in the picture above which in this case is the flight search’s HttpGet function. In this case which is a normal scenario as flight searching requires communication with the database to fetch data based on the search criteria and then formatting the returned data to be displayed to the user. But in an abnormal case let’s say Login request is the slowest instead of search flight, then developers will be able to look straight into the login functionality to find the source of the problem instead of debugging the entire application.

Furthermore, application insight will also allow us to view the live server health status of the web application. This functionality is especially important during peak hours to determine if a scaling up/out is required to be done immediately to cater to the large amount of users using the application.

Instead of live data application insight also provides historical data of the server’s health status over a period as seen above. Whereby these data will also help determine if a scaling change is required for the application.

There are also other useful data provided by application insights that help analyse the application. Such as the highest number of request calls to the application and geographic distribution of users to the system. From here we can obtain information such as the most visited page of the system and which country is the major consumer for the website. In the case that users from a region is very high a decision to add a new localized application for that region can be made to further cater to users from that region.

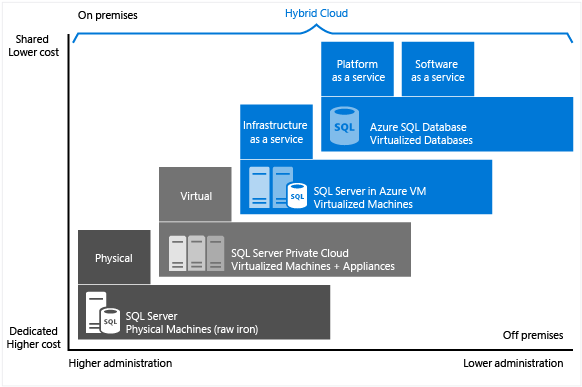
## 4.3 Discussion of Managed Database

As time goes on cloud platforms has been enabling new, more complex business models and building more globally-based integration networks than many analysts has ever predicted (Columbus, 2017). With the rapid amount of growth cloud computing has within the IT market it is expected to have a growth rate of 6 times the rate of overall IT spending growth from 2015 to 2020 as shown in the figure below (Columbus, 2017).

**Figure 4.1 Cloud Computing Market Growth (Taken From: forbes.com)**

Without a doubt cloud computing has slowly make its way into the IT industry with a bang. One of the major fields of cloud computing is managed database services such as the one being used in this project the Azure SQL Database (PaaS).

Azure SQL Database is basically a SQL database hosted in the cloud also known as a platform as a service (PaaS) database or a database as a service (DBaaS) that offers compatibility with most SQL server features across multiple versions (Microsoft5, 2017). The major benefits of using PaaS database such as Azure SQL Database is that the managing of the backend operating system, the server instance and the hosting machine itself is all done by Microsoft, the consumer is only exposed to the database services whilst being able to use other services provided by Microsoft such as monitoring and tuning (Otey, n.d). PaaS ultimately allows one to avoid the expense and complexity of buying and manging software licenses and the underlying application infrastructure. Whereby the consumer will only manage the application and services developed and the cloud service provider will manage everything else (Microsoft Azure, n.d).

**Figure 4.2 Physical vs Cloud SQL Server (Taken from: Microsoft5)**

As seen from the diagram above the difference between a localise storage solution such as physical on-premise databases and a cloud based storage solution such as Azure SQL Databases is the difference in both cost and administration which is the amount of management required.

The reduction in administration whilst using Azure SQL Database instead of a local SQL Database means the team will be able to focus more on the development of the application instead of managing the infrastructure of the application which in this case is the database itself, thus cutting the overall development time of the application (Hurwitz et al., n.d). The pay-as-you-go model for PaaS and the virtualization of infrastructure brings to a big decrease in development cost for the application with the reduction of maintenance cost and hardware cost (Woodford, 2016).

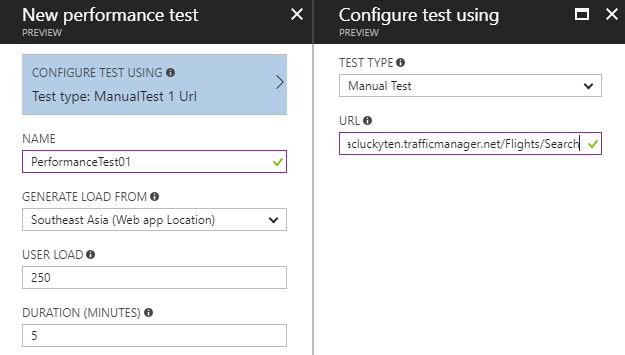
Not only does PaaS exposes the core service to the consumers but also allow them to provide other tools and services to further compliment the core service which helps improve the quality of the overall application being developed with the use of PaaS. Which in this case the use of Azure SQL Database in place of on-premise SQL database also allows the developers to use sophisticated development tools and services provided by Microsoft Azure such as the database monitoring and performance tuning recommendation services (Microsoft6, 2017).

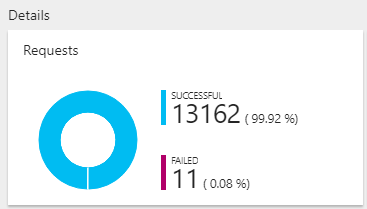
# 5.0 Testing

## 5.1 Unit Testing

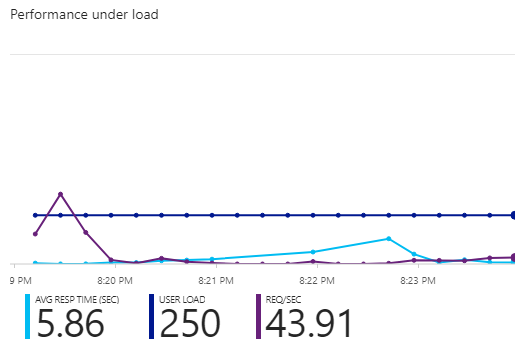
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Description | Test Steps | Expected Result | Actual Results |
| Webpage Navigation | | | | |
| 1 | Default Webpage to Home | 1. Navigate to website | Home index page loads | Home index page loaded |
| 2 | Header Navbar - Home | 1. Navigate to website  2. Click Home on Navbar | Home index page loads | Home index page loaded |
| 3 | Header Navbar - Flights | 1. Navigate to website  2. Click Flights on Navbar | Search Flight page loads without result | Search Flight page loaded |
| 4 | Header Navbar - Register | 1. Navigate to website  2. Click register on Navbar | Register page loads | Register page loaded |
| 5 | Header Navbar - Login | 1. Navigate to website  2. Click login on Navbar | Login page loads | Login page loaded |
| Account Functionality | | | | |
| 6 | Account - Register | 1. Navigate to website  2. Click register on Navbar  3. Fill in information  4. Click Register button | Account successfully registered into the system | Account successfully registered into the system |
| 7 | Account - Login | 1. Navigate to website  2. Click login on Navbar  3. Fill in information  4. Click Log In | Account is authorized and logged into the system as a user | Account is authorized and logged into the system as a user |
| 8 | Header Navbar - Profile | 1. Navigate to website  2. Login to website  3. Click email on Navbar | Profile action page loads | Profile action page loaded |
| 9 | Header Navbar – Log Off | 1. Navigate to website  2. Login to website  3. Click Log Off on Navbar | Logged in account is logged off | Account successfully logged off system. |
| 10 | Account – Change password | 1. Navigate to website  2. Login to website  3. Click email on Navbar  4. Click change password  5. Fill in new password  6. Click change password button | Password successfully changed for logged in user | Password has been changed for logged in user |
| 11 | Account – Change Profile Information | 1. Navigate to website  2. Login to website  3. Click email on Navbar  4. Click change profile information  5. Edit new profile information  6. Click save button | Profile information successfully updated for logged in user | Profile information successfully updated for logged in user |
| Flight Functionality | | | | |
| 12 | Flight - Search | 1. Navigate to website  2. Click Flights on Navbar  3. Fill in search criteria  4. Click search button | Correct flight results for search criteria is displayed | Correct flight results for search criteria is displayed |
| 13 | Flight – Book (Authorized) | 1. Navigate to website  2. Login to website  3. Click Flights on Navbar  4. Search for flights  5. Click book on a flight  6. Select seats to book  7. Click book seats button | Selected seats for selected flight has been booked and recorded into the system under the logged in user. | Selected seats for selected flight has been booked and recorded into the system under the logged in user. |
| 14 | Flight – Book (Unauthorized) | 1. Navigate to website  2. Click Flights on Navbar  3. Search for flights  4. Click book on a flight | User redirected to login page. | User is redirected to login page. |
| 15 | Flight – Book (Unavailable Seats) | 1. Navigate to website  2. Login to website as User1  3. Search for flights  4. Book 5 random seats.  5. Logout of system  6. Login to website as User2  5. Search for the same flight  6. Click book for same flight | Seats that are booked are greyed out and unavailable for selection anymore by other users. | Seats that are booked are greyed out and unavailable for selection anymore by other users. |
| 16 | Account – View Booking | 1. Navigate to website  2. Login to website  3. Search for flights  4. Book 5 random seats.  5. Click email on Navbar  6. Click view booking | Flight and seats booked are correctly shown in view booking page. | Flight and seats booked are correctly shown in view booking page. |

## 5.2 Performance Testing

In order to stress test the performance of the web application the azure performance testing tool will be used.

The performance testing tool allows us to test the performance of the system by simulating a fixed amount of concurrent users from a specific region over a set amount of time that requests a specific URL of the web application. Which in the scenario shown above 250 concurrent users from the SEA region requesting the search flights page of the application will be simulated over the course of 5 minutes.

Once the performance test is completed a series of data will be presented such as the one seen above, where it shows the amount of successful and failed requests during the duration of the performance test.

Another analytical data presented is the average response time per second and the average request per second throughout the duration of the performance test. From the given data it is possible to gauge and see if the application is able to steadily handle large amount of responses without giving up too much response time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Concurrent User Load | 250 | 500 | 750 | 1000 |
| No. Successful Request | 99446 (100%) | 104263 (100%) | 99646 (100%) | 96969 (100%) |
| No. Failed Request | 0 (0%) | 0 (0%) | 0 (0%) | 0 (0%) |
| Avg Response Time (Sec) | 0.35 | 1.44 | 1.74 | 2.6 |
| Avg Request (Sec) | 331.49 | 347.54 | 362.6 | 360.8 |

A total of 4 performance tests will be conducted each simulating 250, 500, 750 and 1000 users respectively targeting the same webpage, from the same region and across 5 minutes. The results are recorded to show how the application handle different amount of user loads concurrently.

From the result above the application can optimally handle requests at around 250~500 concurrent users, anything higher will take a toll on the applications ability to handle requests quickly resulting in lesser requests handled over time.

# 6.0 Conclusion

In conclusion an online flight booking application was developed using C# and ASP.NET MVC5 Framework through the Visual Studio 2017 IDE. The application is hosted in Azure through the Web App Service and consuming the SQL Database service provided by Azure. The entire process of developing and hosting the application to cloud is truly an eye-opening experience as it is the first ever application I have personally developed to be hosted and managed through a cloud service. The benefits that are gained by using Cloud services are clearly shown throughout the development of this system, for example the web site can have a 24-hour uptime without needing a physical device hosting the site continuously. Other considerations while building a standard website such as acquiring and paying for a domain name is also done by Azure without the need for us to interfere which simplifies the management of the application.

Furthermore, Azure also provided various tools that are inaccessible or hard to acquire if the application is to be developed and hosted locally instead. Example of such tools are the monitoring and diagnosing tools for the web application such as Application Insight used in **Section 4.2.4** is something that provides beneficial information for us as developers to not only assess the application and detect problems of the application. These tools would not be available for use without hosting the application to Azure, we would need to procure such tools from other third-party sources which may not be compatible with the application. Thus the tools and services provided by Azure in conjunction with the core services they provide is surely a big help to developers consuming their cloud services not only by cost, but also by ease of use and helping in increasing the applications quality overall.

# 7.0 References

1. BrainVire (2016) Six Benefits of Using MVC Model for Effective Web Application Development [Online] Available at: <https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/> [Accessed: 3rd November 2017]
2. Columbus, L. (2017) Roundup of cloud computing forecasts [Online] Available at: <https://www.forbes.com/sites/louiscolumbus/2017/04/29/roundup-of-cloud-computing-forecasts-2017/#70fa5b1131e8> [Accessed: 6th November 2017]
3. EntityFrameworkTutorial (n.d) What is Code First [Online] Available at: <http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx> [Accessed: 3rd November 2017]
4. Finley, K. (2012) What exactly is GitHub Anyway? [Online] Available at: <https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/> [Accessed: 4th November 2017]
5. Gantz, J.F. and Miller P. (2016) IDC Salesforce economy study 2016 [Online] Available at:<http://www.salesforce.com/assets/pdf/misc/IDC-salesforce-economy-study-2016.pdf> [Accessed: 6th November 2017]
6. Hurwitz, J., Kaufman M., Halper F. ad Kirsch D. (n.d) THE BUSINESS BENEFITS OF PAAS IN CLOUD COMPUTING [Online] Available at: <http://www.dummies.com/programming/cloud-computing/hybrid-cloud/the-business-benefits-of-paas-in-cloud-computing/> [Accessed: 8th November 2017]
7. Lin, C. (2017) Web Apps Overview [Online] Available at: <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-overview> [Accessed: 4th November 2017]
8. Microsoft Azure2 (n.d) What is PaaS [Online] available at: <https://azure.microsoft.com/en-us/overview/what-is-paas/> [Accessed: 7th November 2017]
9. Microsoft1 (2017) Overview of Traffic Manager [Online] Available at: <https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview> [Accessed: 4th November 2017]
10. Microsoft2 (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: 5th November 2017]
11. Microsoft3 (2017) Scale Instance count manually or automatically [Online] Available at: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/insights-how-to-scale> [Accessed: 5th November 2017]
12. Microsoft4 (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: 6th November 2017]
13. Microsoft5 (2017) Choose a cloud SQL Server option: Azure SQL (PaaS) Database or SQL Server on Azure VMs (IaaS) [Online] Available at: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas> [Accessed: 7th November 2017]
14. Microsoft6 (2017) Monitoring and performance tuning [Online] Available at: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-troubleshoot-performance> [Accessed: 8th November 2017]
15. MSDN (n.d) ASP.NET MVC OVERVIEW [Online] Available at: <https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx> [Accessed: 4th November 2017]
16. Otey, M. (n.d) SQL Server Azure IaaS vs. SQL Azure Database PaaS [Online] Available at: <http://searchcloudapplications.techtarget.com/tip/SQL-Server-Azure-IaaS-vs-SQL-Azure-Database-PaaS> [Accessed: 7th November 2017]
17. Woodford, G. (2016) 5 Reasons to develop custom applications in the cloud [Online] Available at: <https://www.onixnet.com/insights/5-reasons-the-cloud-is-the-smart-choice-for-custom-app-development> [Accessed: 8th November 2017]

# 8.0 Appendix

GitHub URL : <https://github.com/luckyten/DDACAssignment>

Microsoft Stream URL: <https://web.microsoftstream.com/video/19a68589-d289-4dde-9624-511f4ba93321>