# **Acknowledgement**

I would like to express my gratitude to my lecturer, Dr Kalai Anand in assisting on the assignment completion providing the guidance and material related to the resources of Azure along with the hands-on lab exercises that applying those resources throughout the course in this semester.

Secondly, during the process of completing this assignment, I would also like to thank my classmate and friend for guiding me on the development of web application with ASP.Net and some steps in using the resources in Azure.

**Table of Contents**

[**Acknowledgement** 1](#_Toc495004612)

[**1.** **Introduction** 3](#_Toc495004613)

[**1.1** **Background** 3](#_Toc495004614)

[**1.2** **Objectives** 3](#_Toc495004615)

[**1.3** **Scope** 4](#_Toc495004616)

[**1.4** **Requirement Specifications** 4](#_Toc495004617)

[**2.** **Project Plan** 4](#_Toc495004618)

[**3.** **System Design** 5](#_Toc495004619)

[**3.1** **Cloud Design Pattern** 5](#_Toc495004620)

[**3.2** **Cloud Architecture Diagram** 7](#_Toc495004621)

[**3.3** **Use Case Diagram** 9](#_Toc495004622)

[**3.4** **Class Diagram** 13](#_Toc495004623)

[**3.5** **Sequence Diagram** 14](#_Toc495004624)

[**4.** **Implementation** 16](#_Toc495004625)

[**4.1** **System** **Demonstration** 16](#_Toc495004626)

[**4.2** **Application Demonstration and Publishing to Azure** 21](#_Toc495004627)

[**4.3** **Azure Components Applied in the Project** 22](#_Toc495004628)

[**4.4** **Application Scaling** 23](#_Toc495004629)

[**4.4.1** **Comparison in Service Plan (S1 vs S2)** 28](#_Toc495004630)

[**4.4.2** **Why Perform Scaling?** 29](#_Toc495004631)

[**4.5** **Managed Database** 30](#_Toc495004632)

[**4.5.1** **Database Chosen** 32](#_Toc495004633)

[**4.5.2** **Platform as a Service (PaaS)** 34](#_Toc495004634)

[**4.5.3** **Database Deployment** 34](#_Toc495004635)

[**5.** **Testing** 37](#_Toc495004636)

[**5.1** **Testing Cloud Application** 37](#_Toc495004637)

[**5.1.1** **Performance Testing** 37](#_Toc495004638)

[**5.1.2** **Unit Testing** 58](#_Toc495004639)

[**6.** **Conclusion** 63](#_Toc495004640)

[**7.** **Reference** 64](#_Toc495004641)

[**8.** **Appendices** 65](#_Toc495004642)

# **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.

To handle these challenges, Dmitriy Prudnikov, Chief Information Officer at Ukraine International Airlines, realized that migrating the website out of UIA datacenters into a public cloud will be an optimal solution to ensure the reliability and security of the services provided. UIA has look into different Cloud services and platform to deploy their web application, with Microsoft Azure and Amazon Web Services become the top options. However, Dmitriy has chosen Microsoft Azure to deploy this service.

## **Objectives**

The following are the objectives of the project:

* To create a web application that able to help client in managing the fight booking process
* To deploy the web application to Microsoft Azure environment
* To deploy the application to different regions
* To maximize the efficiency and performance of the web application by providing several accessing endpoint in different regions
* To test on the efficiency of the web application by performance testing and service tier scaling to determine the best and suitable performance for the web application

## **Scope**

The following are the scope in the system development, deployment and testing:

* The system deployment will cover 2 regions, which are West Europe and South East Asia
* The current system development will only cover 5 airports from different countries, which are Germany, Italy, Ukraine, Spain and Netherlands
* During the testing, only the comparison within Standard tier will be done instead of upgrading to tiers higher than Standard
* The performance testing will be up to 750 users on each deployment region, and will be done before scaling and after scaling

## **Requirement Specifications**

The following are the requirements of the web application that should be concerned on in term of functionalities and deployment:

* The system must be able to handle the flight booking processes
* Customer must be able to check their flight booking history
* Customer must be able to choose the trip type for their flight, which is either Round Trip or One Way Trip
* The flight information of the trip must always up to date by showing the trips from current date and time, and onward.
* The system must be able to show the latest total number of seat availability based on the bookings that being done

# **Project Plan**

# **System Design**

## **Cloud Design Pattern**

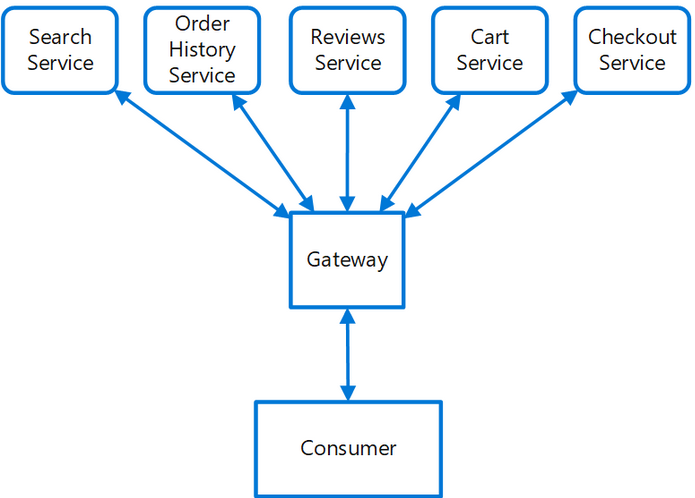
The Cloud Design Pattern that being applied in this project is Gateway Routing. This design pattern allows the user to access the different services hosted in Azure (or other Cloud environment) through a traffic routing (docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing, 2017). For instance, the application may include different functions like search and booking. Those functions can be connected to a single traffic and provides the user with the access to those functions with that particular endpoint that representing each of the function. Besides that, this concept can also be applied if the same application is being deployed into different regions. When user accessing the application, the traffic will route them to the region that nearest to their location.

In this application, the routing is done by using Traffic Manager. Firstly, the web application is deployed to 2 App Services from South East Asia and West Europe. Then, the Traffic Manager will be used to add the 2 App Services as the routing point so that when user intended to access the web application, they will only need to use the DNS name as the URL instead of the full URL of each App Service.

As this application is using Performance as Routing Method in Traffic Manager, the user will be directed to the App Service based on their current location that nearest to the deployment region. The distance between the deployment region and user’s current location will be determined based on the network latency in accessing the web application. However, if one of the routing point to the particular web application is down, user will be redirected to the other App Service in the nearest region.

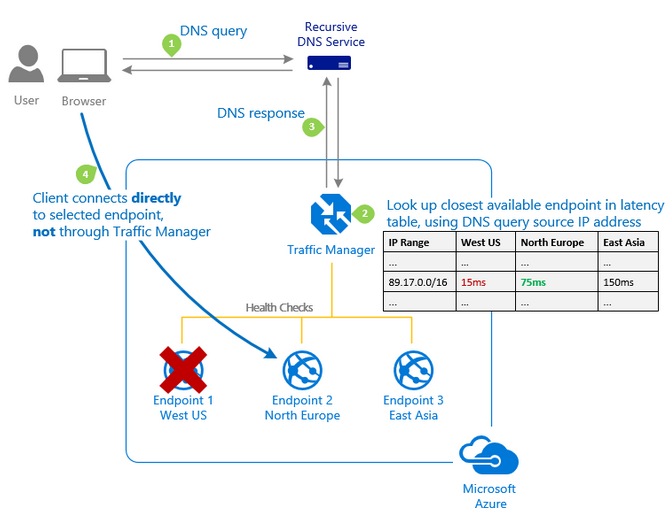
This design pattern is providing the advantage of allowing the use of the same web application to be deployed to different regions and routed with the Traffic Manager so that a single URL can be used to access the different regions instead of required to have different URLs for each different region. Having different URLs will be troublesome for the users to access with that particular URL if the application is deployed to many regions (rather than 2 like in the assignment).

The following diagram shows the structure of this design pattern that route the consumer to different services in the same application:

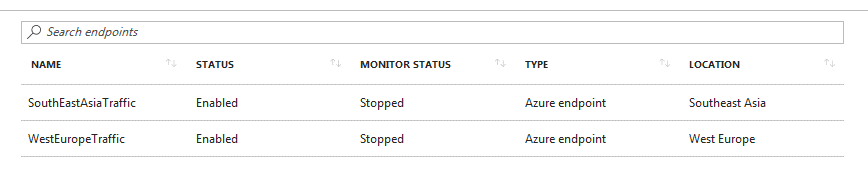


(docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing, 2017)

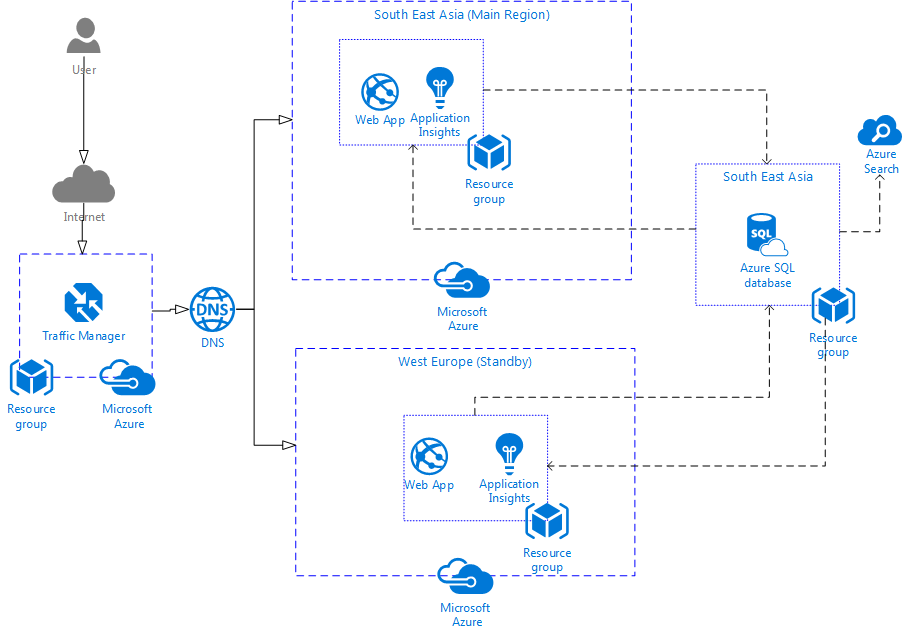
The following diagram shows the diagram of the pattern that applicable to the project that route users to App Service in different regions:



(docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-routing-methods, 2017)

Routing to South East Asia and West Europe with Traffic Manager.

## **Cloud Architecture Diagram**



The design above shows the deployment of web application to different regions and provide the accessibility to either region with the routing of Traffic Manager. In order to distinguish each of the resources for easier configuration, multiple Resource Group are being used. The following are the uses of Resource Group for these resources:

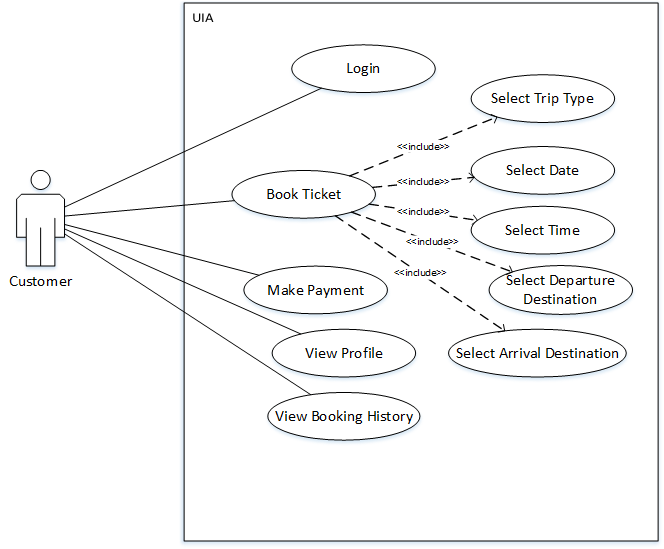
* Traffic Manager
* SQL Database
* App Service and Application Insights

Based on the design above, the user will be accessing the web application through the routing of Traffic Manager by accessing its DNS of the Traffic Manager. As mentioned in previous section, the Traffic Manager is routed by performance, thus the user will be directed to the App Service of nearest region based on the user’s current location. In this case, user will be directed to either South East Asia or West Europe.

In both of the deployment regions, they will be directly connecting to the same database that hosted in South East Asia. Thus, whichever regions that accessed by the users, their activities like retrieve flight record and book flight ticket will be modifying the same database directly instead of applying the database replication process (Geo-Replication) as the current Azure subscription might not be sufficient to support the cost required for Geo-Replication in long term.

This design is able to support the issue of service failing by using another region as backup. The DNS will route the user to the nearest deployment region for best performance based on network latency. However, the secondary concern comes when one of the deployment region is down and becomes unavailable. In this case, the other regions will be used as backup with the Traffic Manager routing users to another region with lowest latency required for accessing. In this project, as there are only 2 regions, the Traffic Manager will directly route user to another region if the main region becomes unavailable.

## **Use Case Diagram**



|  |  |
| --- | --- |
| **Use Case ID** | UC01 |
| **Use Case Name** | Login |
| **Actor(s)** | Customer |
| **Description** | Login into the system with correct username and password |
| **Precondition** | - |
| **Dependency** | - |
| **Main Flow** | 1. Access to the website 2. Enter ID and password and click Login button. 3. The system will authenticate the input by checking on its user database. 4. If the correct detail found, the customer will be authenticated and thus, login successful. |
| **Alternative Flow** | 4.1 If authentication failed, the customer will need to re-enter the detail. |
| **Post Condition** | Customer will be authenticated successfully and able to access the system |

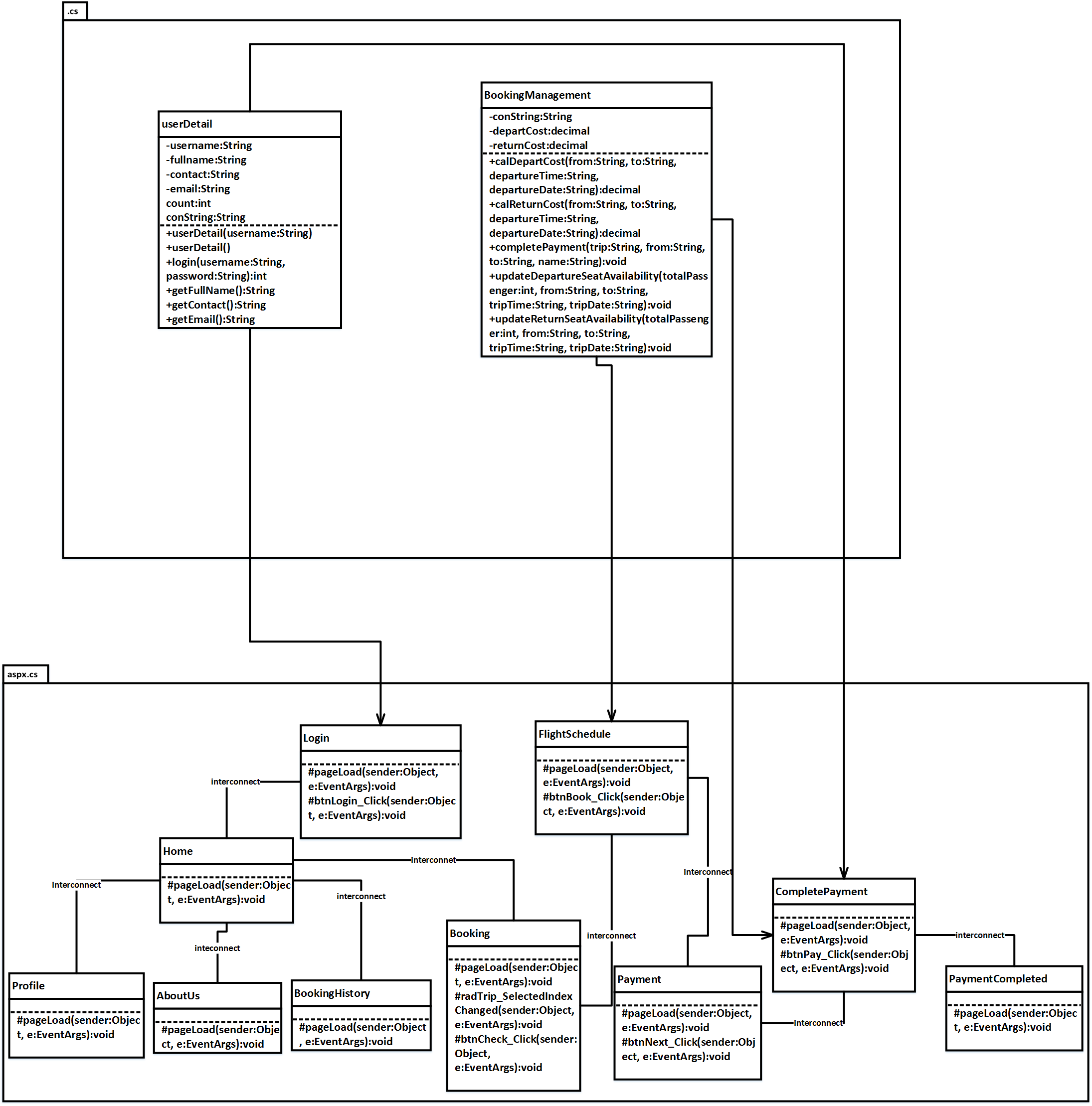
|  |  |
| --- | --- |
| **Use Case ID** | UC02 |
| **Use Case Name** | Book Ticket |
| **Actor(s)** | Customer |
| **Description** | Book Flight Ticket for selected trip |
| **Precondition** | Login |
| **Dependency** | - |
| **Main Flow** | 1. Access to the website 2. Navigate to Book Ticket Section 3. Select Trip Type 4. Select Departure Destination 5. Select Arrival Destination 6. Select trip time |
| **Alternative Flow** | 6.1 If the trip type is Return Trip, select both departure and return time |
| **Post Condition** | User will be directed to payment page |

|  |  |
| --- | --- |
| **Use Case ID** | UC03 |
| **Use Case Name** | Make Payment |
| **Actor(s)** | Customer |
| **Description** | Make payment for selected flight |
| **Precondition** | Login  Select trip |
| **Dependency** | - |
| **Main Flow** | 1. Select payment method 2. Confirm trip detail 3. Confirm payment |
| **Alternative Flow** | - |
| **Post Condition** | Payment will be completed, and flight ticket is booked. |

|  |  |
| --- | --- |
| **Use Case ID** | UC05 |
| **Use Case Name** | View Profile |
| **Actor(s)** | Customer |
| **Description** | Allow customer to view their own information |
| **Precondition** | Login |
| **Dependency** | - |
| **Main Flow** | 1. Access to the website 2. Login with username and password 3. Navigate to Profile page |
| **Alternative Flow** |  |
| **Post Condition** | Customer will be able to view their own profile with information like name and email address |

|  |  |
| --- | --- |
| **Use Case ID** | UC06 |
| **Use Case Name** | View Booking History |
| **Actor(s)** | Customer |
| **Description** | Allow customer to view their flight booking history |
| **Precondition** | Login |
| **Dependency** | - |
| **Main Flow** | 1. Login to the system 2. Navigate to Booking History page |
| **Alternative Flow** |  |
| **Post Condition** | Customer will be able to view their current and previous flight booking record. |

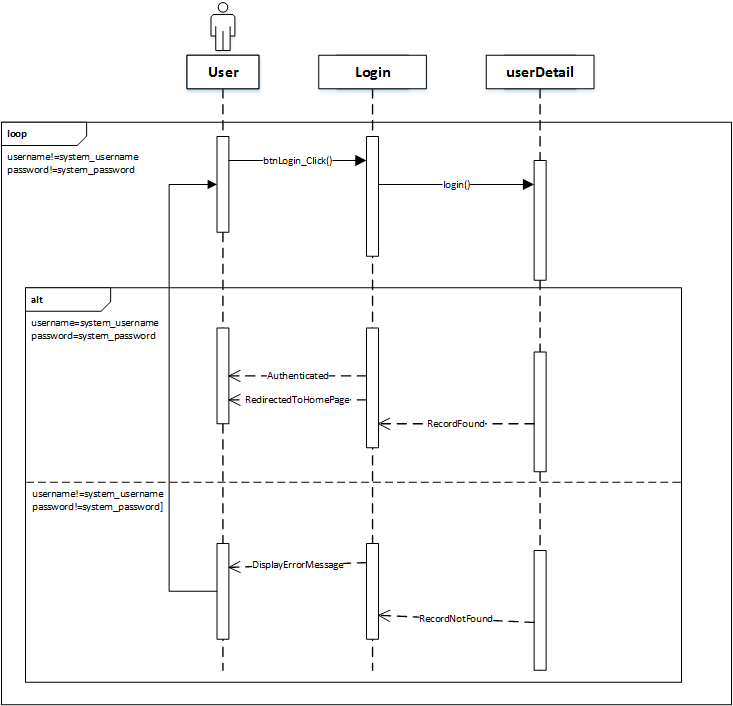
## **Class Diagram**

****

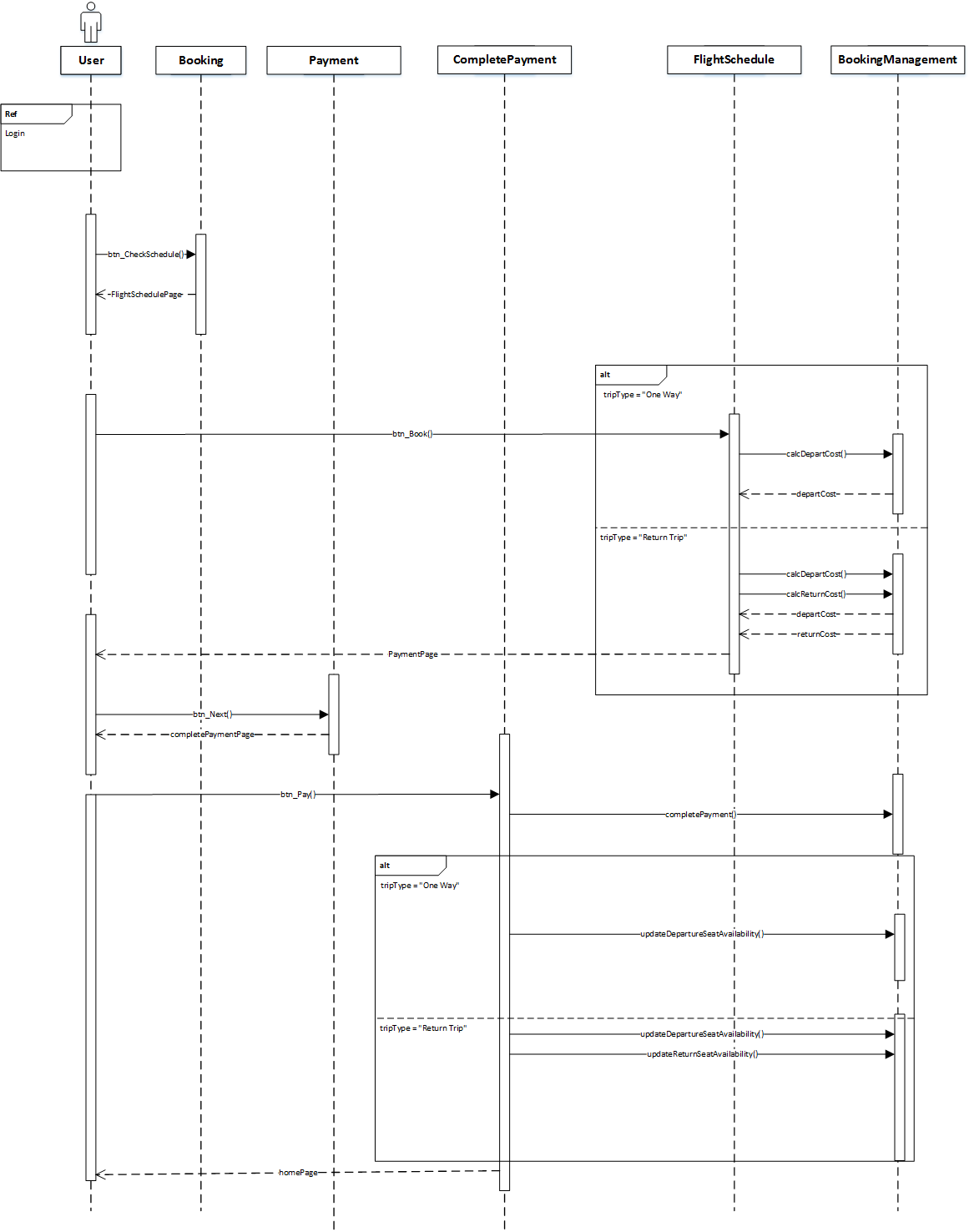
The class diagram shows the class diagram that representing the whole functionalities of the system. The classes in aspx.cs package are storing the methods for the control events like button\_click and page\_load. Whereas, the .cs package is storing the normal classes that can be called from aspx.cs package.

## **Sequence Diagram**

#### **Login**



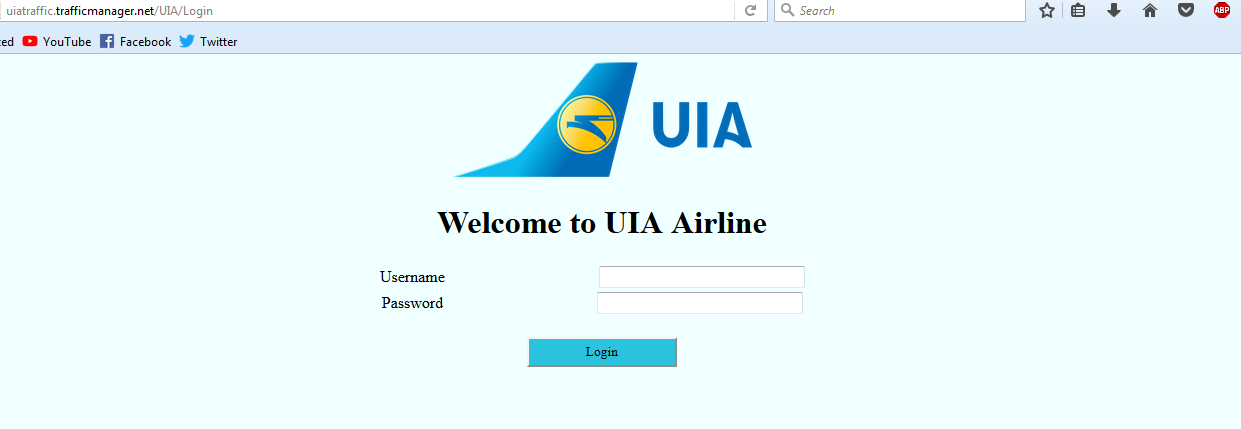
#### **Booking Process**



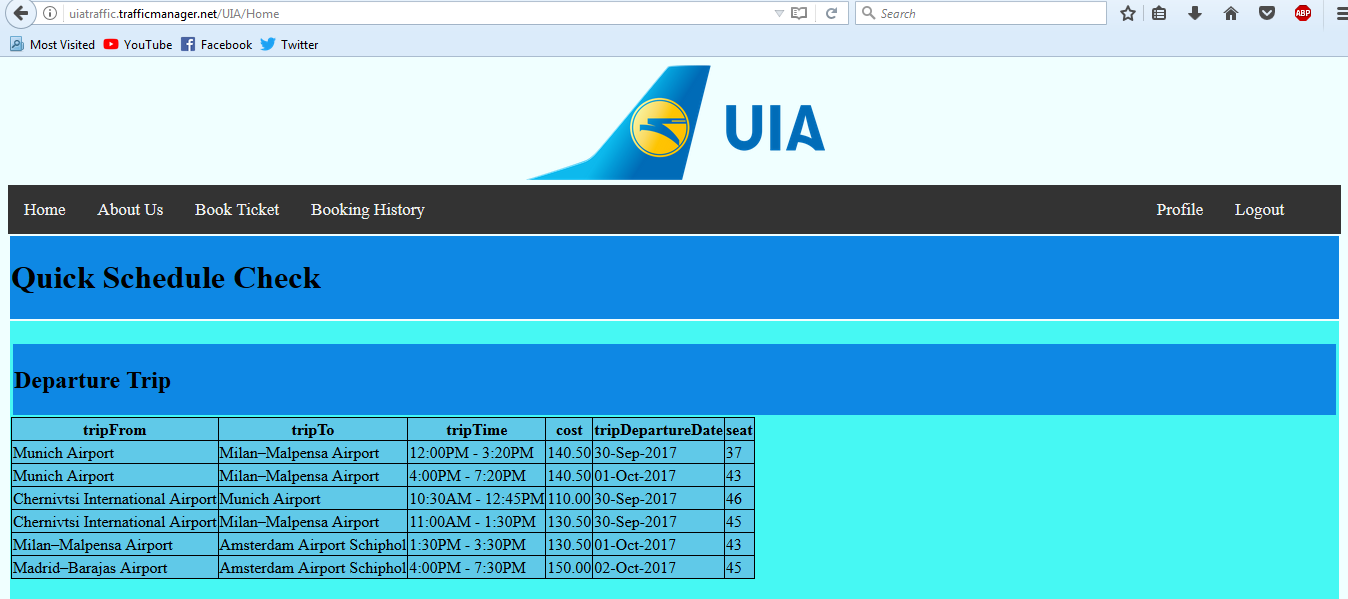
# **Implementation**

## **System** **Demonstration**

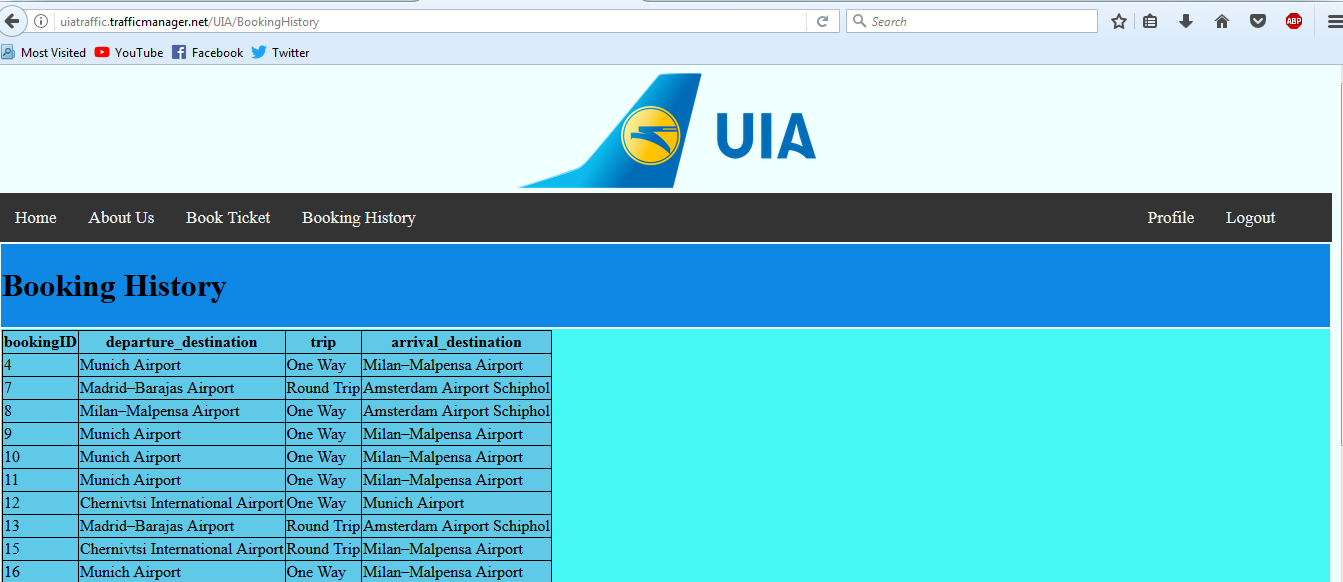
The system implemented is a flight booking system that being hosted with Azure. The following shows the screenshot of the hosted system with URL “uiatraffic.trafficmanager.net/UIA/[webpage name].aspx”, as shown in the address bar at the screenshot below :



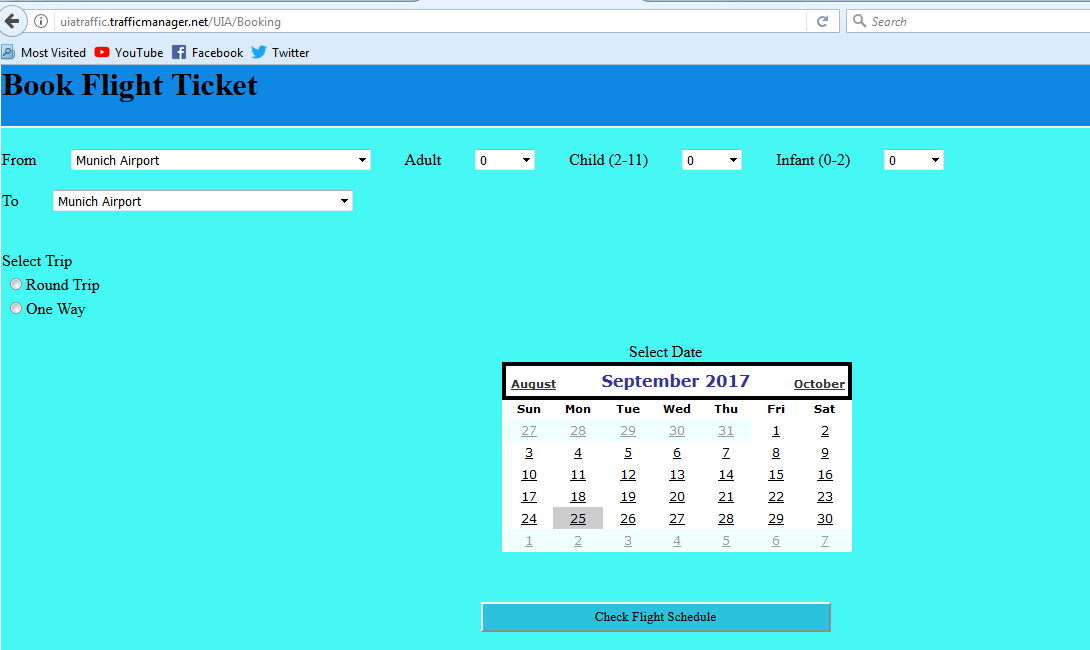
**Login Page**



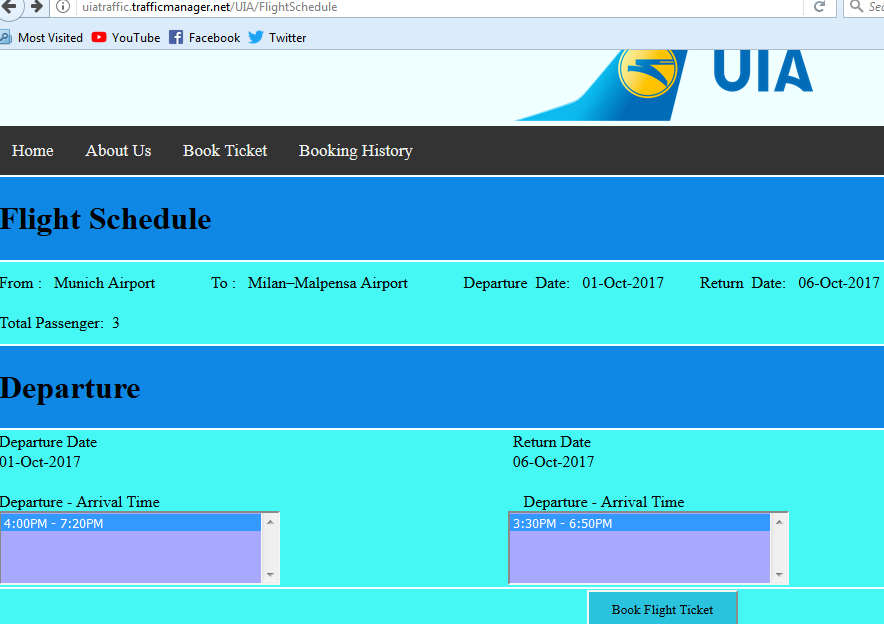
**Home Page**



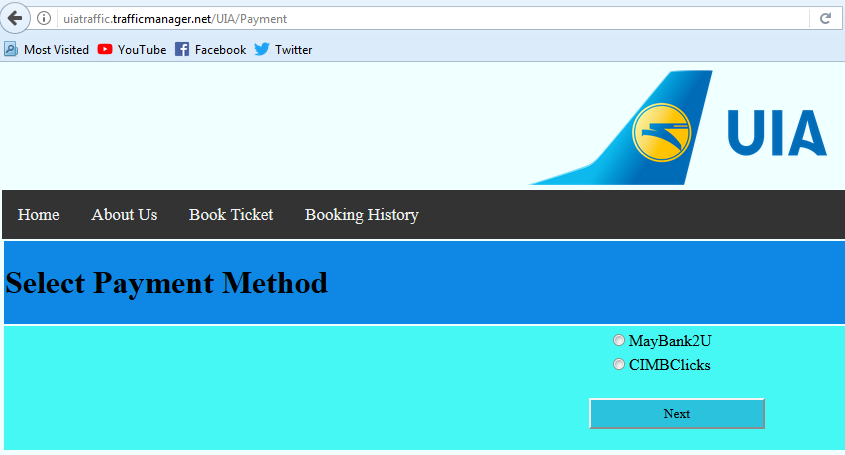
**Booking History**



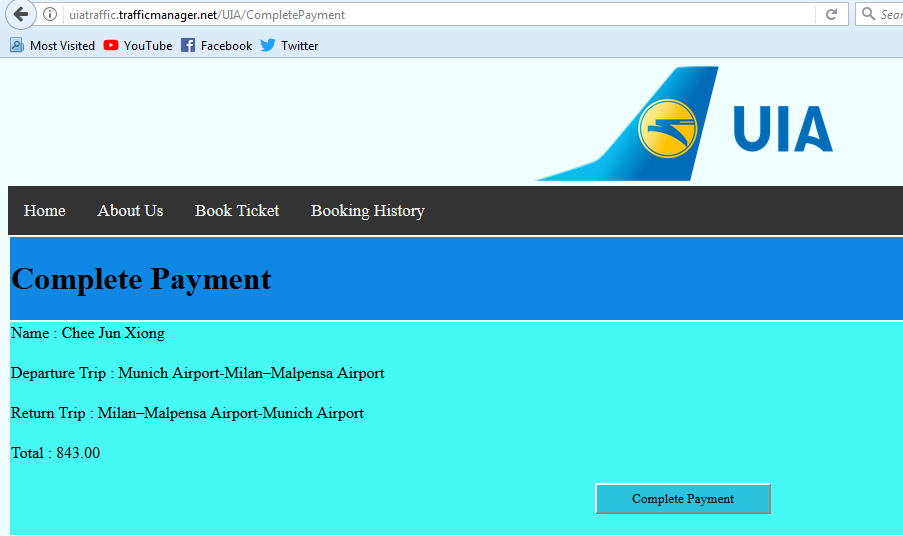
**Booking Page**



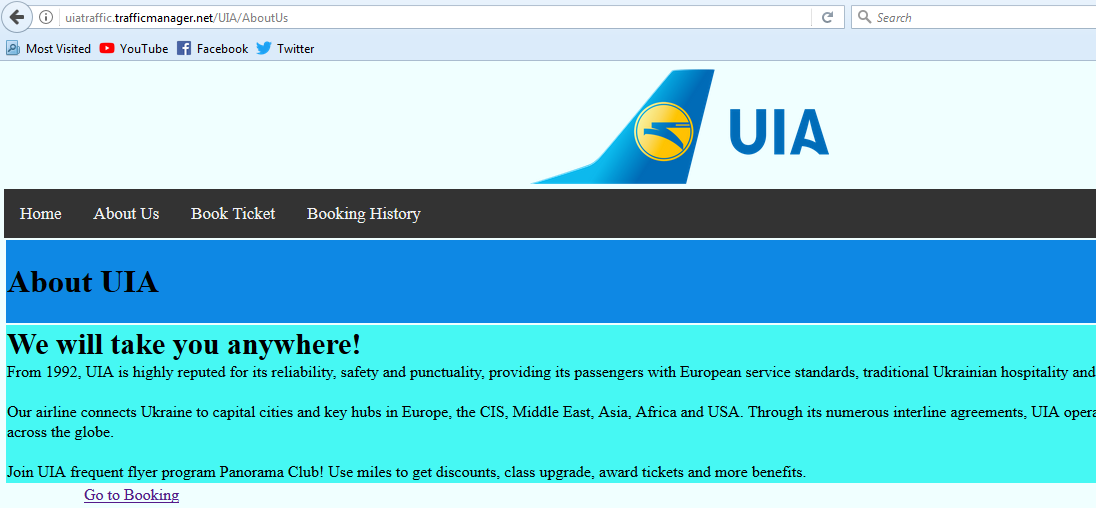
**Booking Page (con’t)**



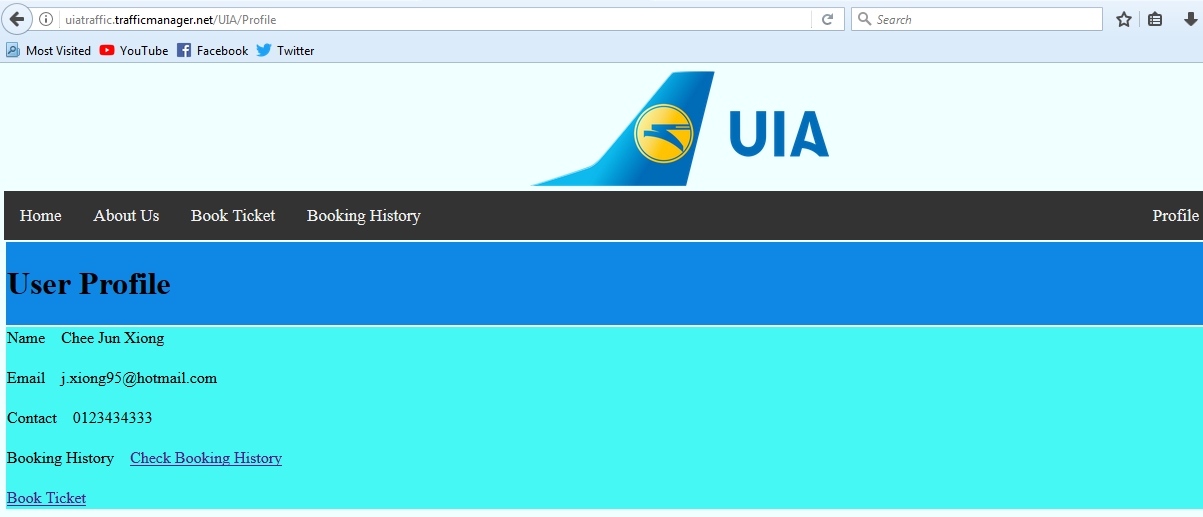
**Confirm Payment Page**



**Payment Completion**



**About Us page**



**Profile Page**

## **Application Demonstration and Publishing to Azure**

The system is created by using ASP.Net and Visual Studio. From Visual Studio, the coding of the web application is done. Visual Studio also provides the capability to connect to Microsoft Azure account, thus, the publishing can be done directly with Visual Studio.

Prior to the publishing, the App Service with suitable service plan must be created in Azure so that the web application created can be deployed into the App Service. The App Service can be created in Azure portal or with Visual Studio after connecting to the Azure account. In this case, I created 2 App Services for different regions in Azure portal and deploy the web application to the App Service from Visual Studio.

After the deployment and publishing to Azure is done, the web application will be accessible through a valid URL, which will be shown in the screenshot at the subsequent parts of the document.

Other than the App Service, the web application is also working with database. The database is first created in Microsoft SQL Server Management Studio with the tables necessary. Then, in Azure, the SQL database and database server are created by selecting the suitable pricing tier. In this case, I choose S1 with 20DTU and 250GB space. After creation is completed in Azure, the database created in SQL Server will be migrated into Azure by connecting to the database server created in Azure.

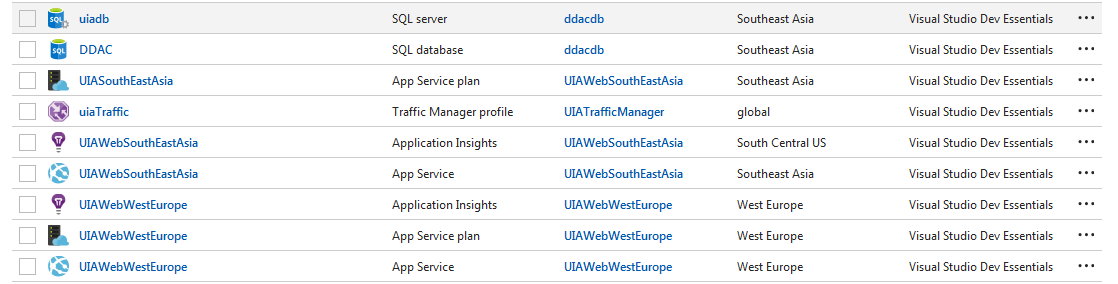
After the migration is done, the modification of connection string will need to be done in the web.config file of the web application created in Visual Studio by changing the connection string from local database’s to the Azure database’s connection string, as shown below:

 Then, publish the web application to Azure with Visual Studio.

## **Azure Components Applied in the Project**

With the system being hosted in Azure, the user will be able to access the web application with the correct URL, which is shown in the address bar in the screenshots above.

In order to do the hosting, there are several resources in Azure are being applied, as shown below:



The resources that being used are

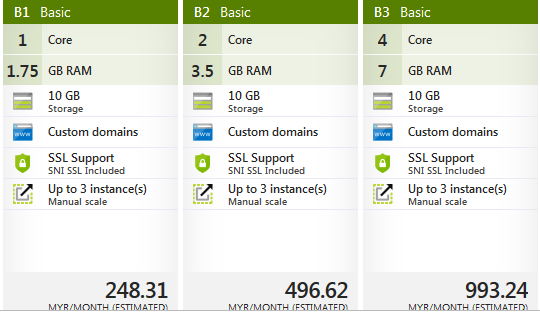
* SQL database that store the database for the web application. The database server will be created in Azure, whereas, the database will be migrated from SQL Server in local environment.
* 2 App Services that being deployed in South East Asia and West Europe. The App Service will contain the web application that deployed through Visual Studio.
* Traffic Manager that acts as the route for users to access the web application that closest to the deployment region with the traffic DNS. It will route user to either South East Asia or West Europe.
* Application Insights is used to help in identifying any issue that happen to the App Service. Each App Service will be containing an Application Insights.

## **Application Scaling**

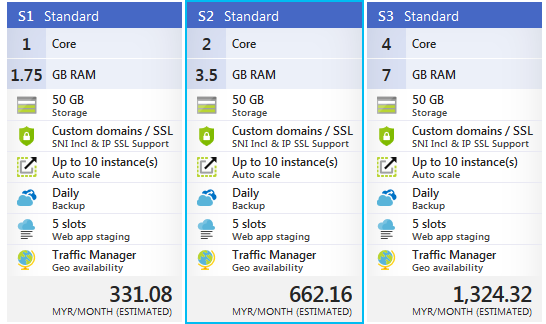
Usually, the applications that hosted in Cloud environment like Azure will provide the users with the options to set the power of the App Service that host the web application. In Azure, the users will be able to host their web application that created with languages like C# (ASP.Net) and PHP from local environment to the Cloud environment with the use of App Service. The App service is providing user with different level of features and performance with charges on the service being required.

The following shows the examples of different tier of the App Service:

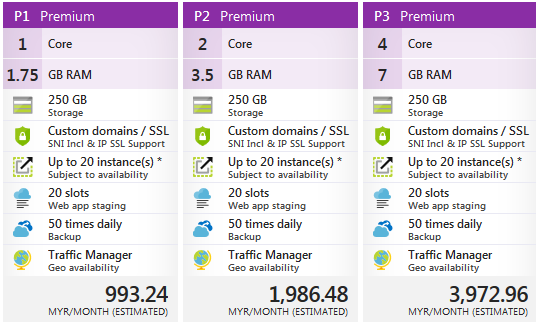
1. Basic Level



1. Standard Level



1. Premium Level



From the screenshots shown above, it shows that there are different level of tiers and services that being offered to the users. Even though each of the App Service can be scaled to different features, the price to use that particular level will be insufficient.

In this assignment, the Standard S1 App Service tier is chosen as the performance and features provided by the S1 tier is enough to make the system functioning. However, I am also found that the use of Standard S2 app will be able to improve the performance of the current system. However, the necessity to rescale the tier can be based on several factors like the services and functionalities of the resource.

The following diagrams will show the statistic of the data in, data out, number of requests load and the response time of the web application with S1 and S2.

**SouthEast Asia (S1 vs S2)**

|  |  |  |
| --- | --- | --- |
|  | **S1** | **S2** |
| **Data In** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 3 Days (21-23 September)\Data_In.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 2 days (23-25)\Data_In.png |
| **Data Out** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 3 Days (21-23 September)\Data_Out.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 2 days (23-25)\Data_Out.png |
| **Request** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 3 Days (21-23 September)\Request.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 2 days (23-25)\Request.png |
| **Average Response Time** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 3 Days (21-23 September)\Response_Time.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 2 days (23-25)\Response.png |
| **Http Error** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 3 Days (21-23 September)\Http_Error.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Statistic After 2 days (23-25)\Error.png |

**West Europe (S1 vs S2)**

In this case, the data in and data out of S2 is less than S1 because the time where the App Service is applying S2 is less.

|  |  |  |
| --- | --- | --- |
|  | **S1** | **S2** |
| **Data In** | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 3 days (21-23 September)\Data_In.png** | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 2 days (23-25 September)\Data_In.png** |
| **Data Out** | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 3 days (21-23 September)\Data_Out.png** | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 2 days (23-25 September)\Data_Ouy.png** |
| **Request** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 3 days (21-23 September)\Request.png | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 2 days (23-25 September)\Request.png** |
| **Average Response Time** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 3 days (21-23 September)\Response_Time.png | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 2 days (23-25 September)\Response.png** |
| **Http Error** | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 3 days (21-23 September)\Http_Error.png | **C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Statistic After 2 days (23-25 September)\Error.png** |

### **Comparison in Service Plan (S1 vs S2)**

This section will focus on the comparison on the resources and functionalities provided in Standard tier, as this project is applying the Standard tier service plan. Within this tier, the resources provided are different in core and RAM.

Core is acts like a normal computer CPU that perform the instruction like opening an application and process the data to perform certain function in the web application and navigation. In this case, higher core number will have better performance in handle the data processing. Whereas, RAM is closely working with the processor core by keeping the data that being processed and accessed when accessing the web application so that the processor can retrieve those data from RAM. In this case, the higher RAM is also able to provide better performance.

From the comparison on Southeast Asia shown above, it is shown that the App Service with higher pricing tier (S2) is able to handle the process more efficiently compare to lower tier (S1). In this case, despite being applied for less period, the higher tier service plan is able to handle more data load, which shown in the Data In and Data Out statistic. Other than that, the requests that can be handled in S2 is also higher than S1, and the response time is also slightly better in S2. However, both service tiers are still possible in having HTTP errors.

For West Europe, even though the period of using S2 and the values in the statistic shown are lower compare to S1, its App Service is having the value that close to S1 in term of Data in, Data Out and Request. This can also prove that the S2 is capable to handle larger data and request load compare to S1, which can be further proved if the period that being applied on App Service with S2 is longer.

From this number in the statistic, it is shown that the S2 with higher specification like more RAM and higher core is able to provide better performance for the web application that hosted in Azure. In this situation, the higher RAM in S2 allows a more efficient response time, which is lower in S2 compare to S1, whereas, the higher core number in S2 allows the data to be processed more efficiently by being able to handle more requests in S2.

### **Why Perform Scaling?**

The pricing tier of App Service is selected based on the robustness of the web application that being deployed into Azure. For instance, if the web application is ready to accept high user loads from different places and time, it will be suitable for the App Service to have higher pricing tier, for instance, using Premium service tier will allows the web application to have more storage compare to Standard tier, and S2 tier is having higher RAM compare to S1 tier.

In the current design and development of the web application, the Standard tier (S1) is providing acceptable performance to handle the user loads as the S1 service plan provided is giving enough resources like number of core and RAM size to process the user load and requests. This service tier allows the application to have acceptable range of data processing, request processing and response time with 750 user loads during the performance testing.

However, when the user load is more than 750 with the users are coming from more than 2 regions, the response time will be increased and the average requests that need be handled will also be increased. In this case, S1 will become inefficient to provide the necessary service and performance. Thus, in order to handle the larger user loads that accessing the web application, the developer will need to consider to upgrade the service plan to higher tier, for instance S2, S3 and Premium service plan. It is because those service plans are providing higher technical specs like higher RAM, more storage and more core numbers in the handling of the data input and output. With higher tier, the response time will also be improved by decreasing the time required while having big user loads.

The high RAM number will help in improving the response time, whereas the number of core will allow the system to handle the simultaneous requests sent from the user loads that access the system.

Besides, this system is required to be deployed to different regions, which are South East Asia and West Europe. Thus, when selecting the service plan, the developer will need to consider on choosing the plan that able to support the use of Traffic Manager and geo-availability. In this case, the App Service should not apply the Basic tier or lower level tier due to their incapability to support the use of Traffic Manager.

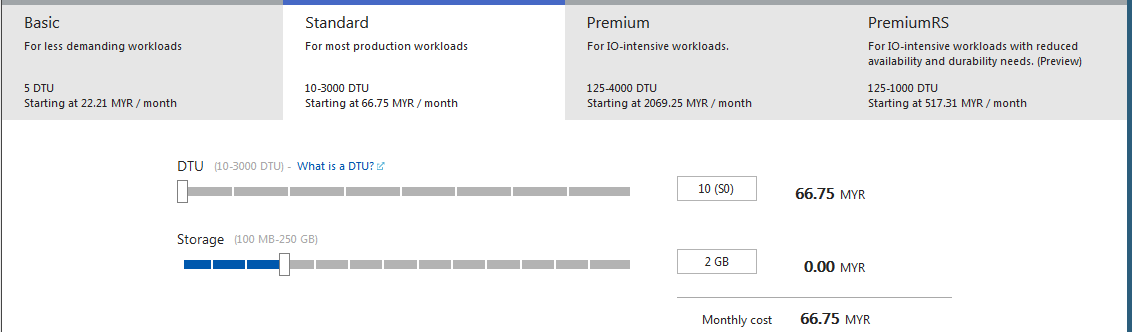
Other than that, the different service tiers will also provide the capabilities to support different number of application instances. The number of instances count when multiple App Services are using the same service plan that currently used by the current App Service. The sharing of the service plan can be applied when it is necessary to have the same instance of the deployed web application. In this application, the instances are not shared as the App Services that being deployed into different regions with their own independent service plan. Thus, Standard (S1) tier is chosen for its capability to support the number of available instances and Traffic Manager. However, in order to have more instances running for the same App Service with same pricing tier, the service tier must be analyzed properly to check on the number of instances available as each tier is providing different number of instances.

However, as the pricing for each tier, from S1 to S2, or from Standard to Premium are imposing different charges, the developer must consider on the performance of the web application that they want to achieve, the robustness of the web application, the resources required like Traffic Manager and the amount of resources like number of core and size of RAM in the service plan before choosing the plan so that the spending is done appropriately based on the necessity of the resources required. In this case, the Standard tier (S2) will be sufficient and providing a more optimal performance based on the current workload and future workload in further system upgrade that will be performed and processed by the web application.

## **Managed Database**

In this project, the database that being used in Azure is SQL database. In order to connect with the web application, the SQL database is first created in Azure. With the completion of creation, the database can be connected directly to the SQL Server Management Studio by providing the correct database server detail. After connected, the user can either working directly with the database created in Azure or import the database in local environment into Azure database server. In this case, I am creating the database in local environment with SQL Server Management Studio and migrate the database to the server in Azure.

When creating the database, user is opened to choose the service tier that will be providing different storage space and Database Transaction Unit (DTU). The provided resources will be segregated based on the different pricing tier. The pricing tier are consist of Basic, Standard, Premium and PremiumRS, with the higher tier providing more range of resources with higher charges in each tier, as shown below:



The performance of database is determined by the unit DTU. DTU is used for determining the throughput for the processes that take place in the database, for instance, access a database table, update record and retrieve record. This unit will guarantee the database with a fixed performance in throughput so that the throughput of that particular database can process the specific amount of workload without being affected by issues like timeout or slower response (Anon., 2017). The response will be affected when the DTU quota (counted in %) is reaching the initial value.

Whereas, the storage of the database size will be determining the amount of data that will be stored in the database. The storage might store the database resources database table, stored procedure, and index. The data that can be stored in those database resources will be determined by the storage size of the database.

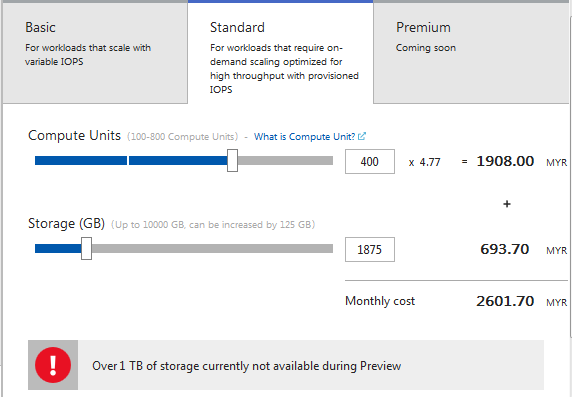
In Azure, the higher DTU and storage size will guarantee a better and more stable performance, but those are bound to a certain price. Therefore, the pricing tier of the database must be chosen wisely based on the amount of data required to be stored and the robustness of the system in term of performing the processes that involve data retrieval, data storing and update. This is to ensure that the resources that being applied are not too redundant and result in higher cost even though lower service tier is already sufficient for the application.

### **Database Chosen**

As mentioned in previous section, the database chosen in Azure environment for this application is SQL database. Besides SQL Database, Azure is also providing different database management system for other options, like MySQL Database, Azure Database for MySQL and Azure Database for PostgreSQL.

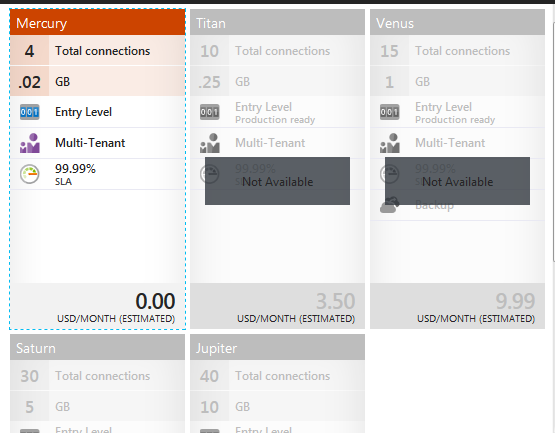
In this project, SQL Database is chosen compare to the other databases mentioned due to:

* Azure Database for MySQL and Azure Database for PostgreSQL are still under preview stage. Thus, despite being able to be used, there might be some limitation in term of functionalities, for instance, in preview stage, the database storage expansion is limited and the service tier is also not fully available yet to support the bigger needs, as shown below:

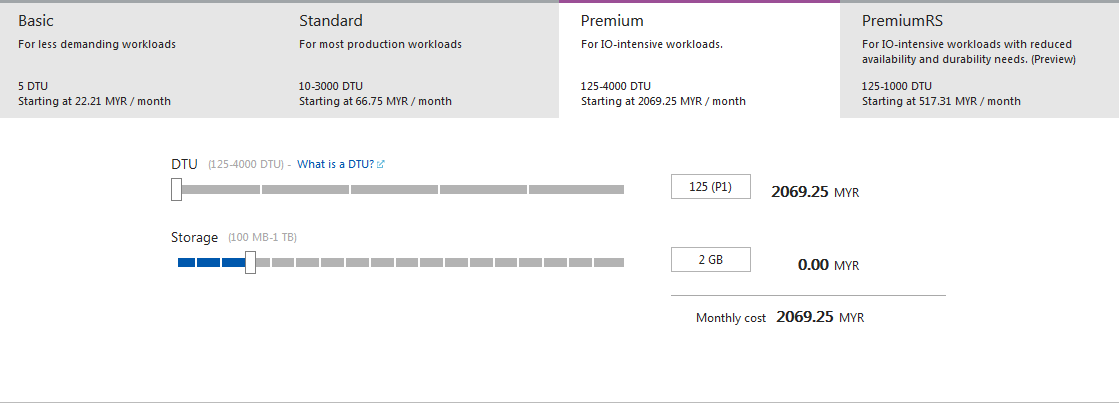


**Storage expansion cannot exceed 1TB and Premium Tier is not available yet**

* MySQL Database is not able to provide more choices in term of pricing tier with the current subscription, thus the development will be limited to only one choice and it might limit the future development and update of the system as well. The limitation is as shown below:



With SQL Database, the service provided is more stable due to it being able to scale to higher storage size and with more tier options, as shown below:



Other than that, SQL database is a stable version instead of preview, so it will be more stable in term of aspects like performance and the services provided, for instance, able to scale the database size without being restricted by MySQL database.

Besides, with the SQL Database, the database migration from SQL Server Management Studio is also easier compare to some other DMBS as SQL Server Management Studio has already provide the mechanism to migrate database to Azure environment with the condition that remote connection options like TCP/IP, instead of downloading additional add-on for the DBMS. For instance, to migrate MySQL database to Azure, Microsoft SQL Server Migration Assistant (SSMA) or SSIS will be needed. Using those approaches might require more configuration steps and it is also possibly hard to use compare to the migration with SQL Server Management Studio.

### **Platform as a Service (PaaS)**

There are 2 approaches on creating database in Azure are by the concepts of Platform as a Service (PaaS) or with Infrastructure as a Service (IaaS), which creating a Virtual Machine associated with Azure, and from there, database creation and migration will be done (docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas, 2017). In this project, the approach of PaaS is applied.

PaaS allows the full utilization of resources within the Azure environment by paying on the service that being purchased in Azure. In this case, the database is migrated into Azure environment from local environment, and a pricing tier is selected. This pricing tier is different based on the performance required (DTU), and the total charge is calculated on monthly basis, which means that the concept of “Pay as you Go” is applied by paying only on the resources that being used (azure.microsoft.com/en-us/overview/what-is-paas/, n.d.). After the setting is done, the database can be accessed and configured in Azure environment with the presence of internet connection. With this approach, the database can also be rescaled to higher performance or higher tier with the condition that the user must have enough credit to afford for the charges in the service.

### **Database Deployment**

In this project, the service tier chosen is Standard tier (S1) with 20 DTU and 250GB storage. The database will then be deployed in South East Asia as the project is aiming to start the deployment from Asia region as the deployment region. As the current system is only aiming to be deployed in South East Asia and West Europe before any further plan is being implemented, the use of the service plan mentioned is tend to be feasible because the storage size in this tier is still able to handle the data store that needs to be done from 2 regions and the DTU is able to provide the acceptable performance in term of throughput, as shown below:

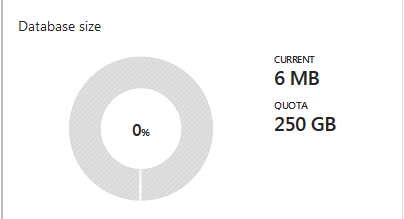


Figure Database Size

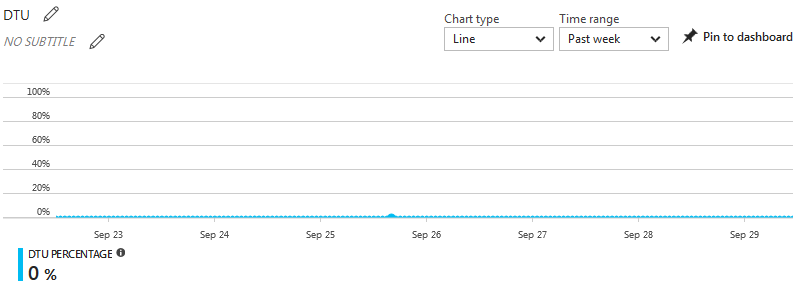
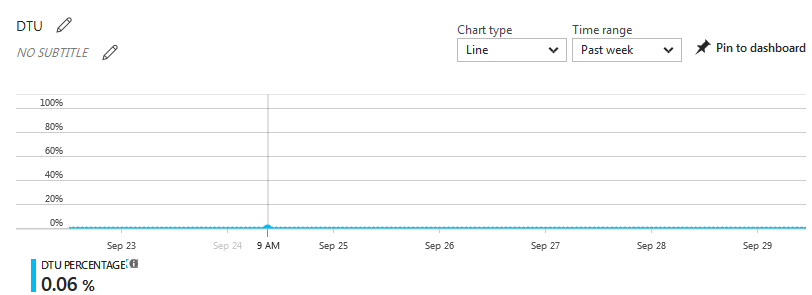


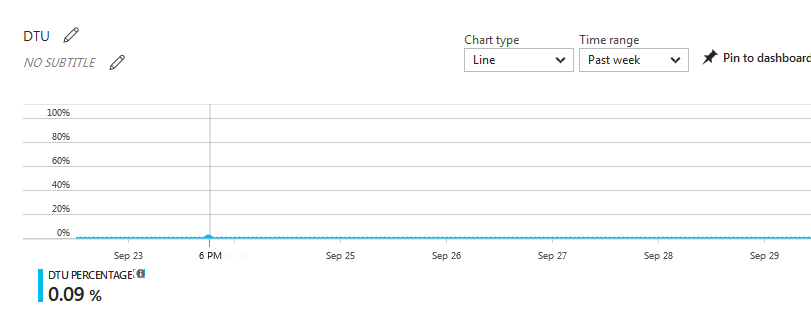
Figure DTU

From the figure shown on the database storage, it is shown that the quota of 250GB is not fully utilized yet as only a portion (6MB) of the given storage is used. Thus, the developer can choose to rescale the storage size based on the current needs as the charges imposed are only applied on DTU. However, if the development plan is going to be expanded by having the deployment of the web application done in more regions and the application itself will also be updated in term of functionality, the larger database storage will be mandatory to cope with the updates. The following discussion will be talking about the sufficiency of DTU to handle the performance with user loads of 750 users from performance test and the loads consumed during unit testing and storage size to handle the database resources.

For the DTU, the figure above shows that the workloads from the database are only using a small percentage of the chosen DTU, which is 20 DTU. As the percentage of average DTU covered is less than 1%, thus, the overall DTU covered in the past weeks are shown as 0%. The following screenshot shows the coverage of the minority of DTU:



0.06% is covered in 24th September.



0.09% is covered in 23rd September.

Thus, from the statistics above, it is shown that the DTU chosen for the current database is sufficient but might be redundant based on the amount of workloads that being performed within the database when using the web application. In this case, the option to scale down the DTU can be considered, for instance from S1 to S0, so that the charges on the DTU will not be redundant. However, the consideration must also be made on the future plan to update the web application, for instance, there will be possibilities that the application will be deployed into more regions that result in more users and request loads and the performance in retrieving the database resources like retrieve record from tables or indexes, at which there are many records that need to be loaded and retrieved.

# **Testing**

## **Testing Cloud Application**

### **Performance Testing**

In this section, the performance testing on the web application from both regions will be done by accessing each of the region. Thus, it means that another region will be temporary closed for the performance testing so that the user load is only entering one region in order to obtain accurate result. The testing is done by having total of 750 user loads (start from 250) on each region within the time interval of 5 minutes. Then, the testing is repeated with the same user load and time interval after the scaling of each of the App Services are done, from S1 to S2.

The following is the test plan of performance testing for West Europe and South East Asia with tier S1 and S2:

|  |  |
| --- | --- |
| **User load** | **Time Interval (minutes)** |
| 250 | 5 |
| 500 | 5 |
| 750 | 5 |

#### **Southeast Asia**

The following shows the result of the performance test in South East Asia with S1:

|  |  |
| --- | --- |
| 250 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(250_users).png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(250_users)-1.png |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(250_users)-2.png |
| 500 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(500_users).png |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(500_users)-1.png |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(500_users)-2.png |
| 750 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(750_users).png |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(750_users)-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(750_users)-2.png |

The following shows the result of the performance test in South East Asia with S2:

|  |  |
| --- | --- |
| 250 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_250_User.png |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_250_User-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_250_User-2.png |
| 500 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_500_User.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_500_User-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_500_User-2.png |
| 750 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_750_User.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_750_User-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_750_User-2.png |

#### **West Europe**

The following shows the result of the performance test in West Europe with S1:

|  |  |
| --- | --- |
| 250 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_250_Shutdown_SEA.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_250_Shutdown_SEA-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_250_Shutdown_SEA-2.png |
| 500 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_500_Shutdown_SEA.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_500_Shutdown_SEA-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_500_Shutdown_SEA-2.png |
| 750 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_750_Shutdown_SEA.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_750_Shutdown_SEA-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_750_Shutdown_SEA-2.png |

The following shows the result of the performance test in West Europe with S2:

|  |  |
| --- | --- |
| 250 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_250.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_250_-_1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_250_-_2.png |
| 500 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_500.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_500-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_500-2.png |
| 750 users | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_750.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_750-1.png |
|  | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_750-2.png |

#### **Result Analysis**

Based on the comparison, it shown that the performance of the web application like the capability to handle the number of request per second and response time are directly affected by the service plan of the web application. As shown in the comparison above, the handling of 250 users load in S2 is having better performance compare to S1, which can be shown on the response time as follow (South East Asia):

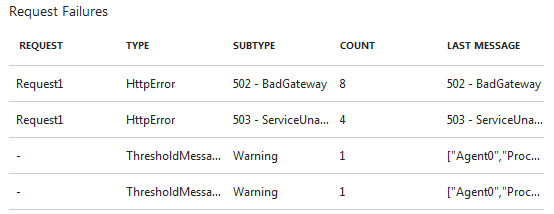
|  |  |
| --- | --- |
| **S1** | **S2** |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test 250 users (shut down west europe)\Performance_Test_(250_users)-1.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp SouthEastAsia Performance\Performance Test  (shut down west europe) Tier 2\Performance_250_User-1.png |

The result above shows that with the same user load of 250, the requests that can be handled per second in S2 is slightly more than S1, and the average response time is also lower compare to S1. The same outcome is also happen to the test cases with 500 and 750 user loads in S1 and S2.

However, despite the App Service being scaled up from S1 to S2, the possibility of getting HTTP error is still exist as under certain situation. The following shows the errors that still occur after scaling up (West Europe) with 250 users load:

|  |  |
| --- | --- |
| S1 | S2 |
| C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Performance_Testing_250_Shutdown_SEA.png | C:\Users\Admin\Desktop\APIIT\DDAC\WebApp WestEurope Performance\Performance Testing 750 User (Shutdown SEA)\Tier 2\Performance_Test_250.png |

From the comparison above, it is shown that the higher tier service plan is not necessary to be able to prevent the request failure despite S2 is having more core and RAM than S1. The following image shows the errors that received during the performance test:



The HttpError 502 – BadGateway is an error that occurred when the website is overloaded by user access and request (Schofield, 2013). This error might also occur due to the programming logic of the web application, for instance, a proper design pattern is not applied and result in the workload in that particular C# (or any related programming language) class becomes higher, thus, slow down the process of accessing the web application. For HttpError 503, it signifies that the website is currently unavailable for accessing due to the issues like server busy and overload (Fisher, 2017). From this situation, it is shown that the errors occurred not because of the RAM and core that applied to the App Service, but due to the overload of request and overload during that specific timing, at which there are simultaneous requests that being sent in that result in the server not being able to handle the requests. Therefore, this is the reason why the S2 service plan met the error and S1 is not getting the similar errors.

From the result above, it is shown that the performance is directly affected by the user load and also the service plan applied to the App Service. With the higher core and RAM, the application will be able to accept more data and request loads and having better response time. However, the errors that occurred are not only affected by the service plan, but also the accepted requests loads during a specific time, which means that during that specific time, there might be overloading of request that render the website becomes unavailable for accessing, as shown in the screenshot above.

### **Unit Testing**

#### **Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | Login with correct username and password  Use the following detail:  Username: Junz95  Password: Password123 | Navigate to home page | Navigate to home page | Pass |
| TC02 | Login with incorrect username or password  Use the following detail:  Username: Junz95  Password: Password12 | Message “Login Failed. Please check username and password” will be displayed | Message “Login Failed. Please check username and password” is displayed | Pass |
| TC03 | Login without entering username and password. | Message “Login Failed. Please check username and password” will be displayed | Message “Login Failed. Please check username and password” is displayed | Pass |

#### **Booking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | Enter the departure and arrival destination, trip type, trip date, and number of passengers.  Then, click Check Flight Schedule button  Use the following detail:  From:  Munich Airport  To:  Milan Airport  Trip Type:  One Way  Number of Passengers:  1 Adult  Departure Date:  30th September 2017 | Navigate to Flight Schedule page | Navigate to Flight Schedule page | Pass |
| TC02 | Click Check Flight Schedule button without enter any detail | Message “Please check booking detail” will be displayed | Message “Please check booking detail” is displayed | Pass |
| TC03 | Select the departure and arrival destination, select number of passenger and trip date without setting the Trip Type.  Use the following detail:  From:  Munich Airport  To:  Milan Airport  Number of Passengers:  1 Adult  Departure Date:  30th September 2017 | Message “Please check booking detail” will be displayed | Message “Please check booking detail” is displayed | Pass |
| TC04 | Select the departure and arrival destination, Trip Type, and select number of passenger without Trip date  Use the following detail:  From:  Munich Airport  To:  Milan Airport  Trip Type:  One Way  Number of Passengers:  1 Adult | Message “Please check booking detail” will be displayed | Message “Please check booking detail” is displayed | Pass |
| TC05 | Select the departure and arrival destination, Trip Type, and select Trip date without the number of passenger  Use the following detail:  From:  Munich Airport  To:  Milan Airport  Trip Type:  One Way  Departure Date:  30th September 2017 | Message “Please check booking detail” will be displayed | Message “Please check booking detail” is displayed | Pass |
| TC06 | Select Trip Type, and select Trip date and the number of passenger without select the destination  Use the following detail:  Trip Type:  One Way  Number of Passengers:  1 Adult  Departure Date:  30th September 2017 | Message “Please check booking detail” will be displayed | Message “Please check booking detail” is displayed | Pass |

#### **Flight Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | Trip Type is One Way and the departure time is selected | Navigate to Payment page | Navigate to Payment page | Pass |
| TC02 | Trip Type is Round Trip, the departure time and return time are selected | Navigate to Payment page | Navigate to Payment page | Pass |
| TC03 | Trip Type is One Way and departure time is not selected | Message “Please select departure time” will be displayed | Message “Please select departure time” is displayed | Pass |
| TC04 | Trip Type is Round Trip. The departure time and return time are not selected | Message “Please select departure time” and “Please select return time” will be displayed | Message “Please select departure time” and “Please select return time” are displayed | Pass |
| TC05 | Trip Type is Round Trip. Return time is not selected. | Message “Please select return time” will be displayed | Message “Please select return time” is displayed | Pass |
| TC06 | Trip Type is Round Trip. Departure time is not selected. | Message “Please select departure time” will be displayed | Message “Please select departure time” is displayed | Pass |

#### **Payment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | Select MayBank2U as payment method. Click Next button. | Navigate to Complete Payment page | Navigate to Complete Payment page | Pass |
| TC02 | Select CIMBClicks as payment method. Click Next button. | Navigate to Complete Payment page | Navigate to Complete Payment page | Pass |
| TC03 | Click Next button without selecting the payment method. | Message “Please Select Payment Method” will be displayed | Message “Please Select Payment Method” is displayed | Pass |

#### **Complete Payment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | The booking detail is displayed. Click Complete Payment button. | Navigate to Payment Completed page | Navigate to Payment Completed page | Pass |

#### **Payment Completed**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Action** | **Expected Output** | **Current Output** | **Status/Remark** |
| TC01 | Payment is completed. Click on the hyperlink to navigate to home page. | Navigate to Home Page. The seat availability prior to the booking that shown at Home Page will be deducted based on the number of passenger selected.  In this case, the total seat is deducted by 1. | Navigate to Home Page. The seat availability shown at Home Page is deducted based on the number of passenger selected.  Total seat availability is deducted by 1. | Pass |

# **Conclusion**

In conclusion, the system development is completed successfully and is able to be deployed into Azure environment, thus make it accessible from every browser. From the deployment in Azure, it is shown that the performance of the web application is playing an important role in allowing the capability of accepting more user loads and requests so that more data and requests can be processed, and the optimal response time can be provided for user who access the application. As the application is deployed into 2 different regions, the capability to allow user in accessing the different regions must be provided with suitable Cloud Design Pattern so that it will be easier in term of accessibility and easier to be maintained and configured by the developers. Thus, the Gateway Routing pattern is applied to suit this needs by using Traffic Manager so that user will only require to access either of the region through the DNS. Even though the system is developed and deployed successfully, there are still several improvements that can be done in the future. For instance, improve the booing process by allowing the system to show clear seat layout instead of showing only the number of available seats, and improve the deployment in Azure like deploy the application to more regions instead of 2 and apply Geo-Replication to the database by adding more secondary databases in the same or different region. Those improvement will be important to improve the system reliability if there is any issue that result in one of the regions or databases is down.

# **Reference**

Anon., 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 29 September 2017].

azure.microsoft.com/en-us/overview/what-is-paas/, n.d. *What is PaaS? Platform as a service.* [Online]   
Available at: https://azure.microsoft.com/en-us/overview/what-is-paas/  
[Accessed 30 September 2017].

docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing, 2017. *Gateway Routing pattern.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing  
[Accessed 30 September 2017].

docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas, 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 30 September 2017].

docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-routing-methods, 2017. *Traffic Manager routing methods.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-routing-methods  
[Accessed 30 September 2017].

Fisher, T., 2017. *503 Service Unavailable.* [Online]   
Available at: https://www.lifewire.com/503-service-unavailable-explained-2622940  
[Accessed 28 September 2017].

Schofield, J., 2013. *502 Bad Gateway error: what to do when you can't get through to a website.* [Online]   
Available at: https://www.theguardian.com/technology/askjack/2013/aug/01/502-bad-gateway-error  
[Accessed 28 September 2017].

# **Appendices**

URL of presentation videos:

<https://web.microsoftstream.com/video/91294379-4802-45b2-9c45-1ed53d49172d> (introduction and system design)

<https://web.microsoftstream.com/video/42a5bf6c-4952-4409-af6a-49571524d644>

(all subsequent contents)

Sources in GitHub:

<https://github.com/gsuan2001/DDACAssignment.git>