

**Designing and Developing Applications on the Cloud**

**CT071-3-3-DDAC**

**INDIVIDUAL ASSIGNMENT**

**Hand in Date:** 9 October 2017

**Assignment Title:** Ukraine International Airlines (UIA)

**NAME**: Neow Chu Jing

**STUDENT** **ID**: TP037217

**INTAKE**: UC3F1701SE

**LECTURER**: Dr. Kalai Anand Ratnam

Acknowledgment

I would like to express my sincere gratitude to Dr. Kalai Anand Ratnam for guiding me in pursuing knowledge of designing and developing applications on the cloud. He has shown great expertise, hard work and patience when guiding me. In addition, he has extensive knowledge in Cloud Technology that helped me to explore better when learning. In his guidance, I was fortunate to learn about Cloud Technology and get the opportunity to practice and apply it. As a result, I have successfully designed and developed Ukraine International Airlines (UIA) through Microsoft Azure.

**CONTENTS**

[1.0 Introduction 4](#_Toc495346162)

[1.1 Project Background 4](#_Toc495346163)

[1.2 Project Objectives 4](#_Toc495346164)

[1.3 Project Scope 4](#_Toc495346165)

[1.4 Requirement Specifications 5](#_Toc495346166)

[2.0 Project Plan 6](#_Toc495346167)

[3.0 Design 7](#_Toc495346168)

[3.1 Modelling 7](#_Toc495346169)

[3.1.1 Use Case Diagram 7](#_Toc495346170)

[3.1.2 Sequence Diagram 8](#_Toc495346171)

[3.1.2.1 Register 8](#_Toc495346172)

[3.1.2.2 Login 8](#_Toc495346173)

[3.1.2.3 Book Flight 9](#_Toc495346174)

[3.1.2.4 View Booking 9](#_Toc495346175)

[3.1.3 Data Modelling 10](#_Toc495346176)

[3.1.4 Sitemap 10](#_Toc495346177)

[3.2 Cloud Architecture 11](#_Toc495346178)

[4.0 Implementation 12](#_Toc495346179)

[4.1 Application Development 12](#_Toc495346180)

[4.2 Azure Publishing 13](#_Toc495346181)

[4.3 Application Scaling 14](#_Toc495346182)

[4.3.1 Traffic Manager 14](#_Toc495346183)

[4.3.2 Auto Scale 15](#_Toc495346184)

[4.4 Testing 16](#_Toc495346185)

[4.4.1 Unit Testing 16](#_Toc495346186)

[4.5 Screenshot and Relevant Codes 21](#_Toc495346187)

[4.6 Performance Testing 26](#_Toc495346188)

[4.7 Managed Database 29](#_Toc495346189)

[5.0 Conclusion 33](#_Toc495346190)

[6.0 References 34](#_Toc495346191)

# Introduction

## Project Background

Ukraine International Airlines (UIA) is one of the largest airline in Ukraine which offers domestic and international passenger flights and cargo services to Europe, Middle East, United States and Asia. The company is planning to expand their business to a new market. As we all know, airlines make profits when selling flight tickets or when customers purchase their cargo services. As a result, they rely on their websites to allow customers to purchase tickets and services. Therefore, website becomes very important for UIA as it will affect not only the profits but customer’s impression, system’s efficiency, business operation and etc. Currently, UIA are able to serve customers in Ukraine without a problem but are facing problems when serving customers out of Ukraine. The main problem is they have experienced denial-of-service (DOS) attacks which harm site performance and reliability and also did not have adequate performance to serve their visitors from many parts of the world. In order to solve these problems and improve their services, CEO of UIA decided to migrate their website out of their current datacentres into a public cloud. In summary, UIA decided to design and deploy an Online Flight Booking System with using Microsoft Azure.

## Project Objectives

* Develop an Online Flight Booking System that compatible with Cloud Concept
* Deploy an Online Flight Booking System that able to defend denial-of-service (DOS) attacks
* Deploy an Online Flight Booking System which supports multiple regions with performance guaranteed
* Deploy an Online Flight Booking System which supports load balancing

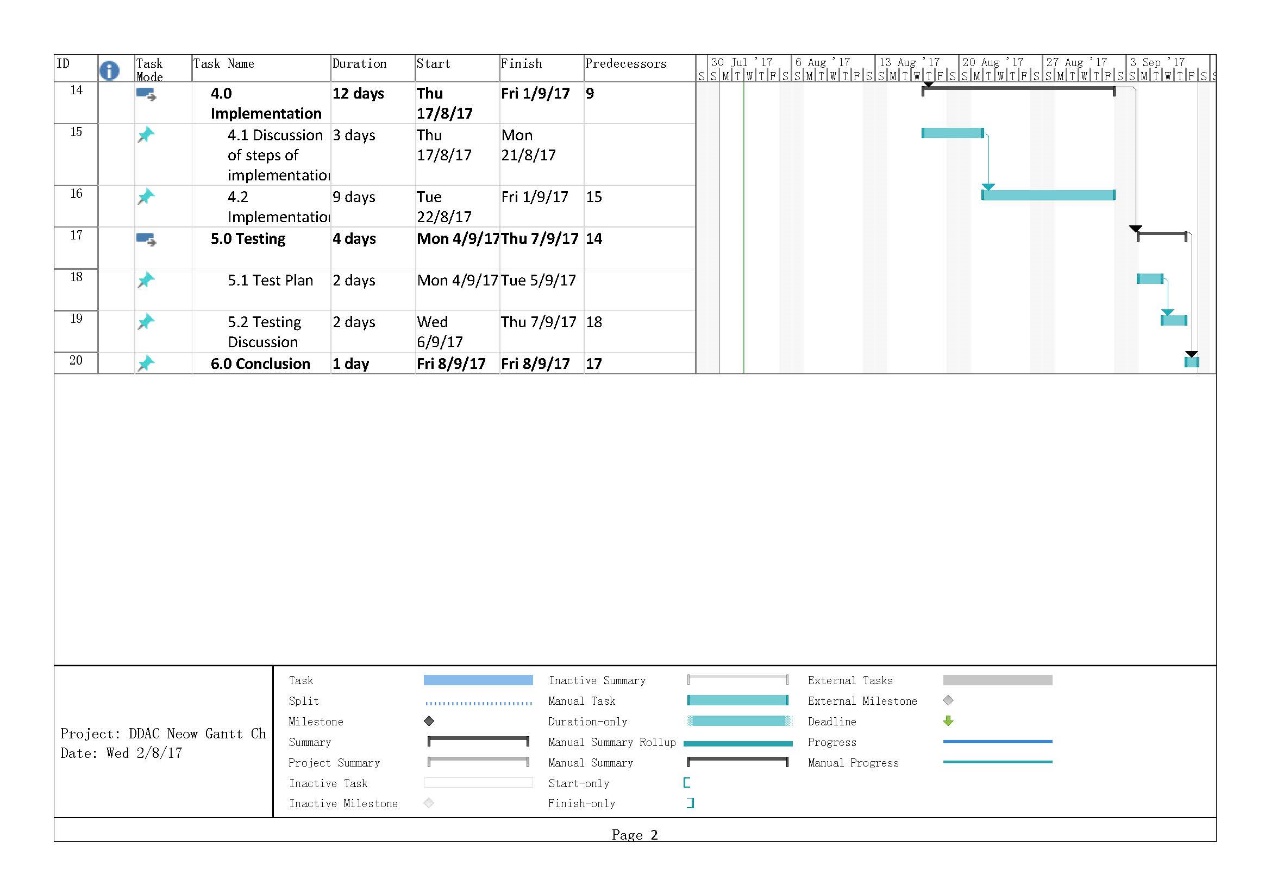
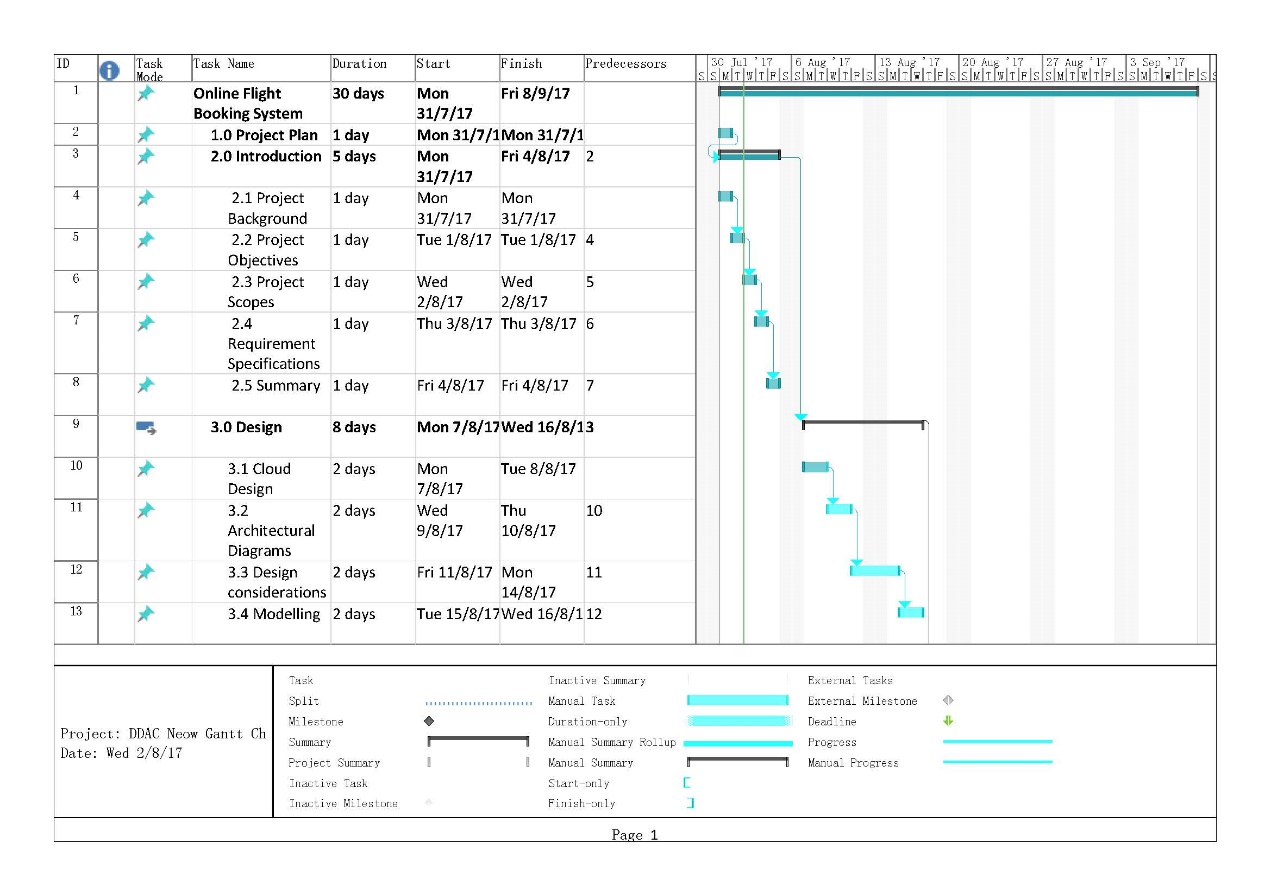
## Project Scope

The Online Flight Booking System will allow users to register, login, search latest flight, book flights and view booking information. This system will be deployed on Microsoft Azure and will support most browsers such as Internet Explorer, Google Chrome, Mozilla Firefox and etc.

## Requirement Specifications

* System allows users to register
* System allows users to login
* System allows users to search flight
* System allows users to book flight
* System allows users to view booking information
* System has good scalability. It automatically scale up and scale down to ensure good performance
* System has high availability. It always be available even when users come from different regions
* System has high security. It stops denial-of-service (DOS) attacks and able to work as usual.

# Project Plan

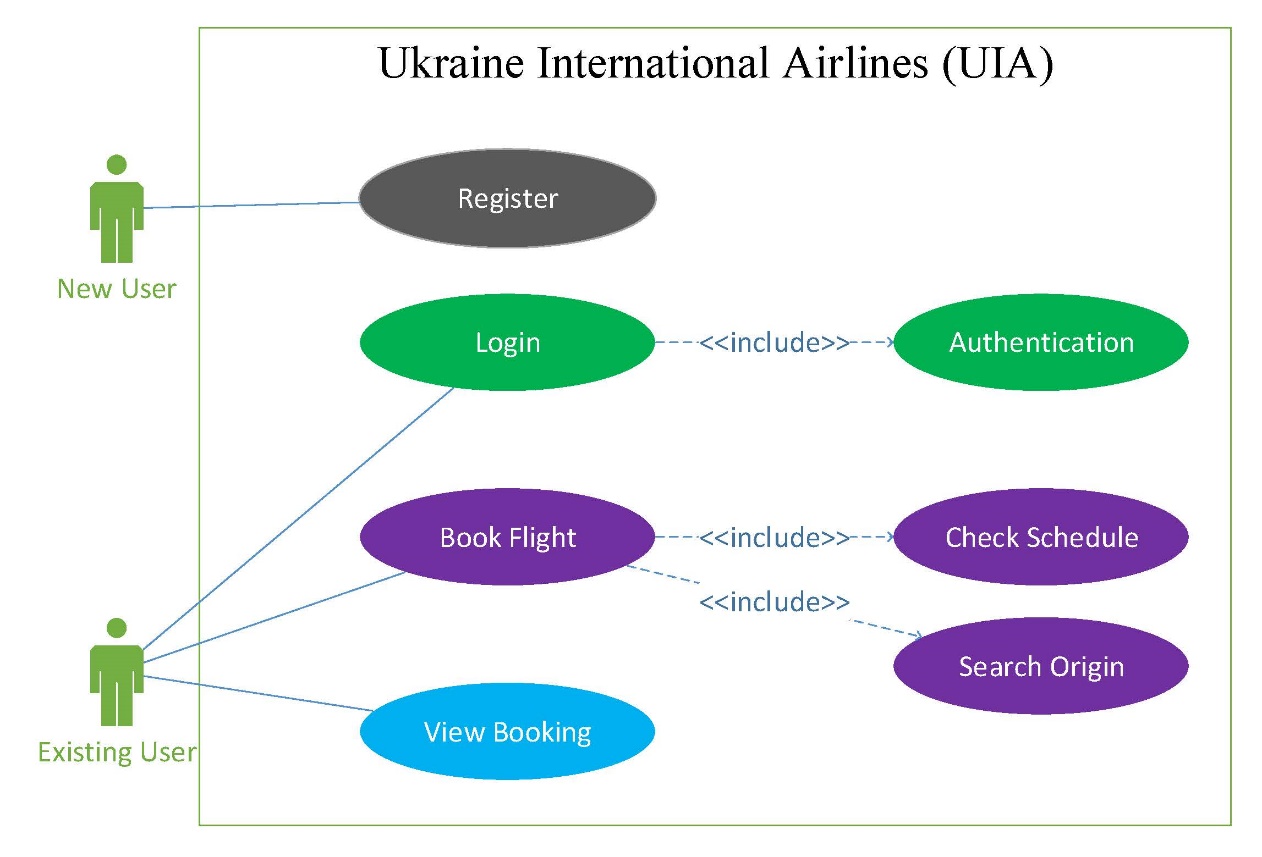


# Design

This section discusses about the design of the system. It contains modelling of the system which involves Use Case Diagram, Sequence Diagram, Data Modelling, Site Map, Cloud Architecture and etc. In detail, there are several considerations will be discussed so that the priorities will be well arranged. First of all, UIA’s is expanding its market to different corners around the world. However, they will mainly target on the US Market. In the meantime, they also would like to test their system on SEA region. As a result, project team was instructed to deploy the system on US and SEA with a given budget.

## Modelling

### Use Case Diagram

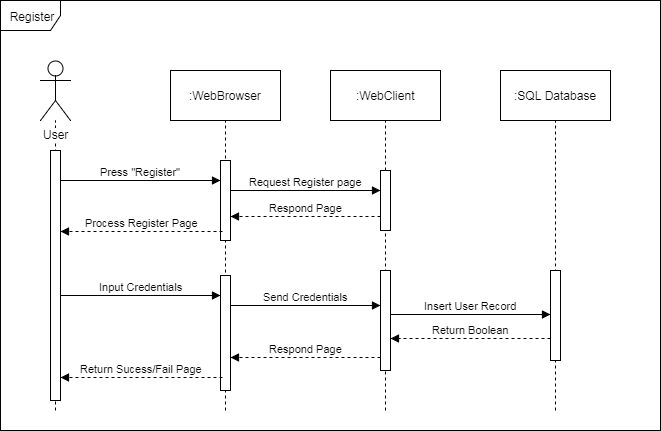


Above diagram shows that there are two types of users in this system. New User is allowed to register themselves. Existing User may be able to login to the system which involves authentication. Besides, Existing User may be able to book flight which he/she may check the schedule and origin to check the latest flight information. After booking a flight, he/she may navigate to view booking for viewing booking information.

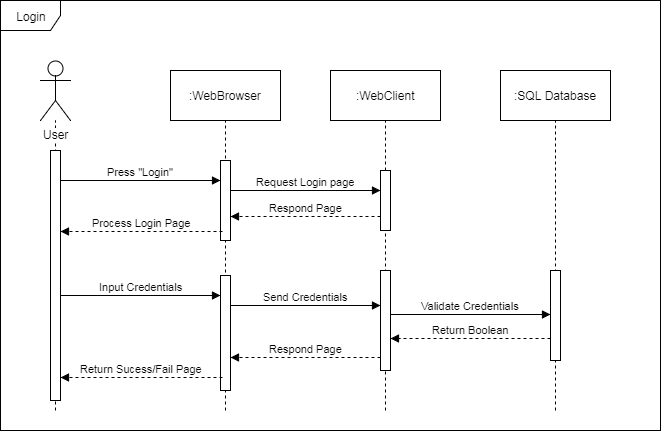
### Sequence Diagram

Sequence Diagram is an interaction diagram that shows the overview and sequences of the system. It illustrates the flow of the system which helps readers to understand easily how the system works.

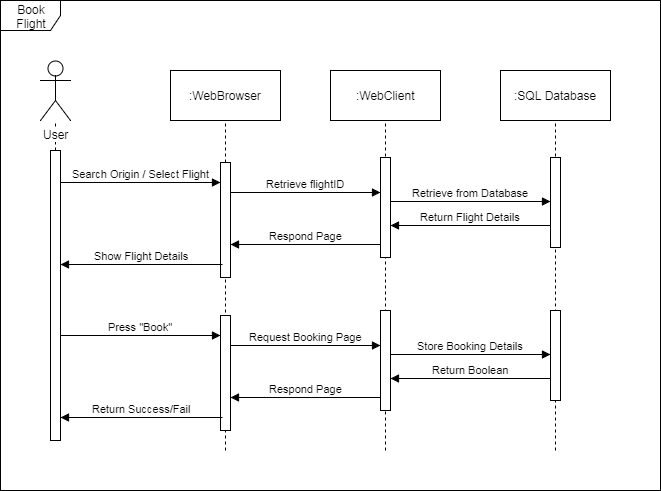
#### Register



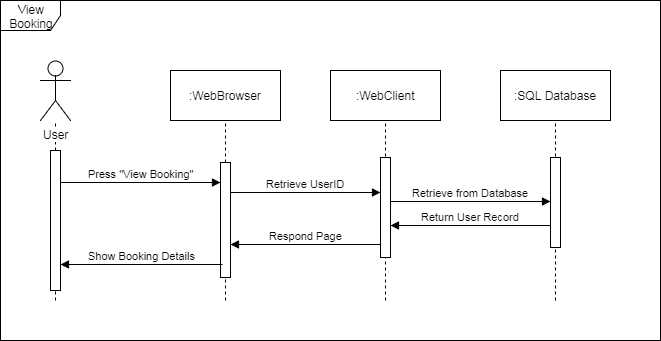
#### Login



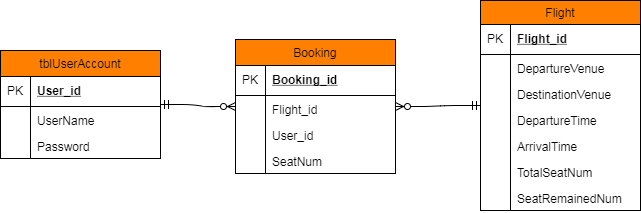
#### Book Flight



#### View Booking

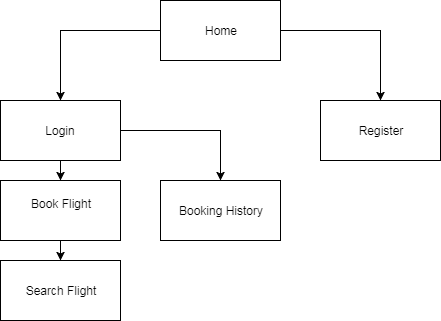


### Data Modelling



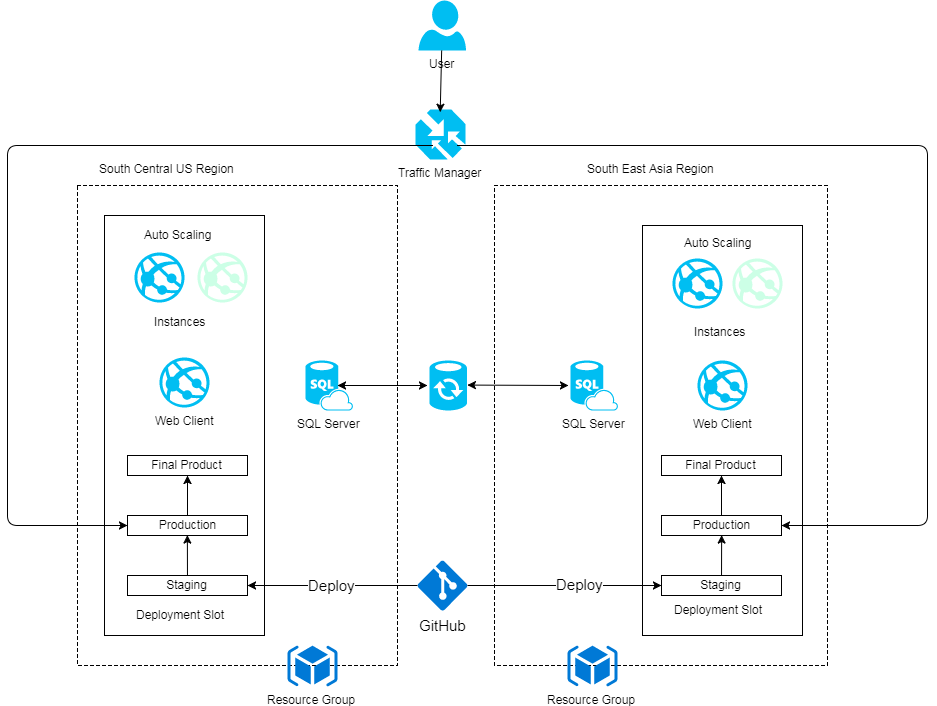
Above diagram is an entity relationship diagram (ERD) that shows relationships between different tables in the database. In tblUserAccount, it saves user information. In Booking, it saves booking information and user information after users made a booking. In Flight, it stores flight information which contains availability of flights, destinations arrival time and etc.

### Sitemap



Above diagram is the sitemap of the system. It illustrates the flow of the system. All users will start using the system from “Home”.

## Cloud Architecture



Above diagram is the cloud architecture of deploying the system using Microsoft Azure. As we have talked about that UIA is entering US Region and SEA Region. In addition, the primary region that the system will focus on is on US Region and another instance will be on SEA Region. Two of the instances will be sharing the same database, there will be no effect on users even they access the system at different locations. Besides, when there is a new version or release, project team will deploy it over GitHub and it will automatically synchronize with Microsoft Azure which saving much time by applying Cloud concept.

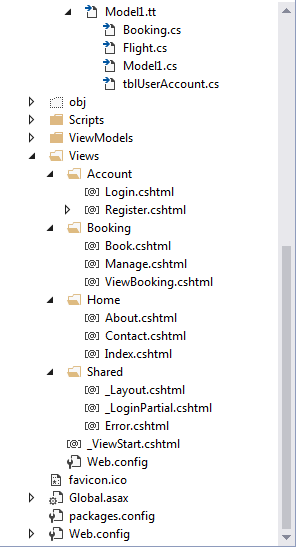
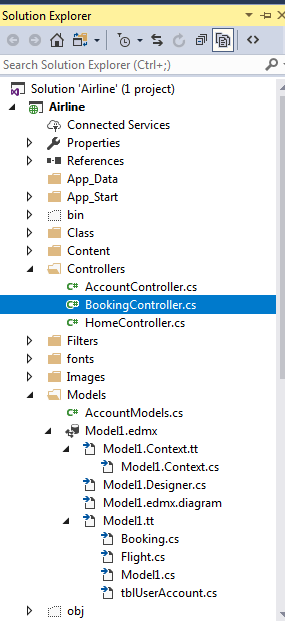
# Implementation

The source code of this system is available at https://github.com/adrianneow/uia

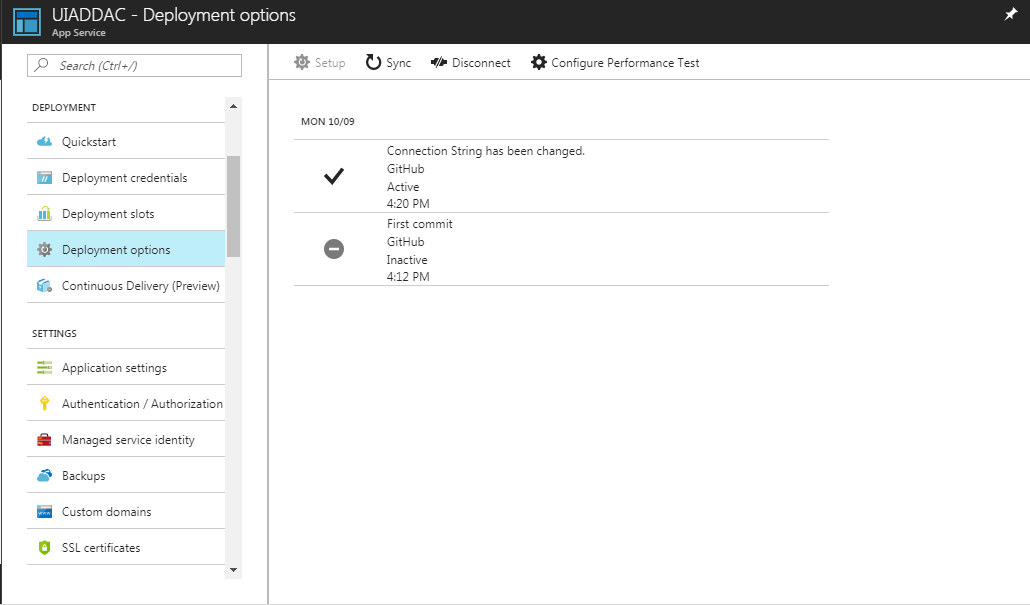
## Application Development



The development of the system was developed using C# MVC and Microsoft SQL Server. Basically, this system is made up of several components which are model, view and controller. Each of the components has their own roles. First of all, model is taken care of the data storage, integrity, consistency, queries and mutations. Model mainly used to serve database retrieves and stores data. Controller is responsible to receive, interpret & validation input and interacts with model to create data for the view. View is the presentation of assets & code and design. View is used to render the user interface and output. Below are some pictures of the structures of the system.



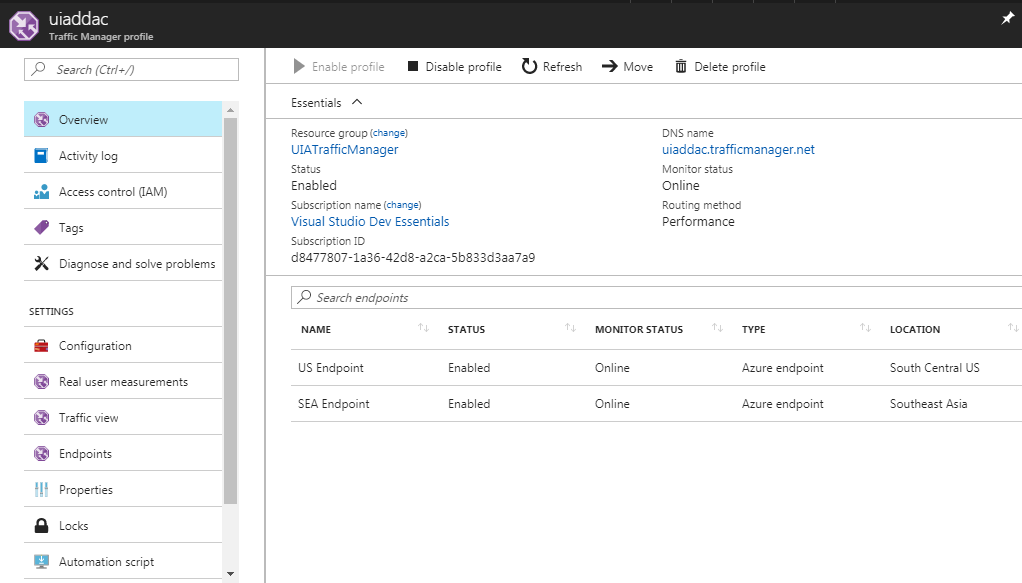
## Azure Publishing



As we have mentioned, this system will be uploaded to Microsoft Azure. At the same time, it was published through development option with using GitHub. This can allow all changes or updates made on GitHub will automatically synchronize with Microsoft Azure.

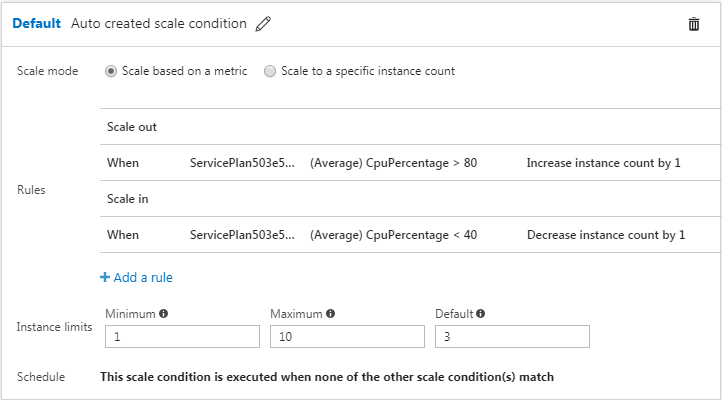
## Application Scaling

### Traffic Manager



As UIA will also be covering users from US and SEA Region, traffic manager will need to be configured to guarantee performance. In detail, traffic manager will route users to the nearest server to specific endpoint which has lower network latency to the user. By using Traffic Manager, if one of the server is having problem, it will route the users to other server which all users will still be able to access to the application. Besides, it also has load balancing feature which helps to distribute the load across different servers.

### Auto Scale



Autoscale is a built-in feature of Cloud Services, Mobile Services, Virtual Machines, and Websites that helps applications perform their best when demand changes. Of course, performance means different things for different applications (Microsoft Azure 2017). The main usage of Autoscale is to allow system to run smoothly even when there are high loads in the system. In UIA, we have set that Microsoft Azure will increase instance by 1 when the average CPU Percentage has reached 80 and will decrease instance by 1 if the CPU Percentage is below than 40. In fact, this is very helpful as UIA may handle loads during promotion period or peak seasons.

## Testing

### Unit Testing

Register

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Element** | **Test Case ID** | **Test Condition** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| Register | A01 | Username: test  Password: test | User fills up all field and valid details | Register is successfully | Register is successfully | Pass |
| Register | A02 | Username: test  Password: | User does not fill up all field | Register is unsuccessful | Register is unsuccessful | Pass |
| Register | A03 | Username: existed  Password: existed | User attempts to register | Register is unsuccessful | Register is unsuccessful | Pass |

Login

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Element** | **Test Case ID** | **Test Condition** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| Login | B01 | Username: user  Password: 123456 | User fills up all field and valid details | Login is successfully | Login is successfully | Pass |
| Login | B02 | Username: user  Password: | User does not fill up all field | Navigate to login page | Navigate to login page | Pass |
| Login | B03 | Username: user  Password: wrong | User fills up invalid details | Navigate to login page | Navigate to login page | Pass |

Search Flight

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Element** | **Test Case ID** | **Test Condition** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| Search Flight | C01 | DepartureVenue was predefined into database. User searched valid details | User searched valid details | System shows user result successfully | System shows user result successfully | Pass |
| Search Flight | C02 | DepartureVenue was predefined into database. User searched valid details | User searched valid details with lowercase letter | System shows user result successfully | System shows user result successfully | Pass |
| Search Flight | C03 | DepartureVenue was not defined into database. User searched invalid details | User fills up invalid details | System returns “No Result” message | System returns “No Result” message | Pass |

Book Flight

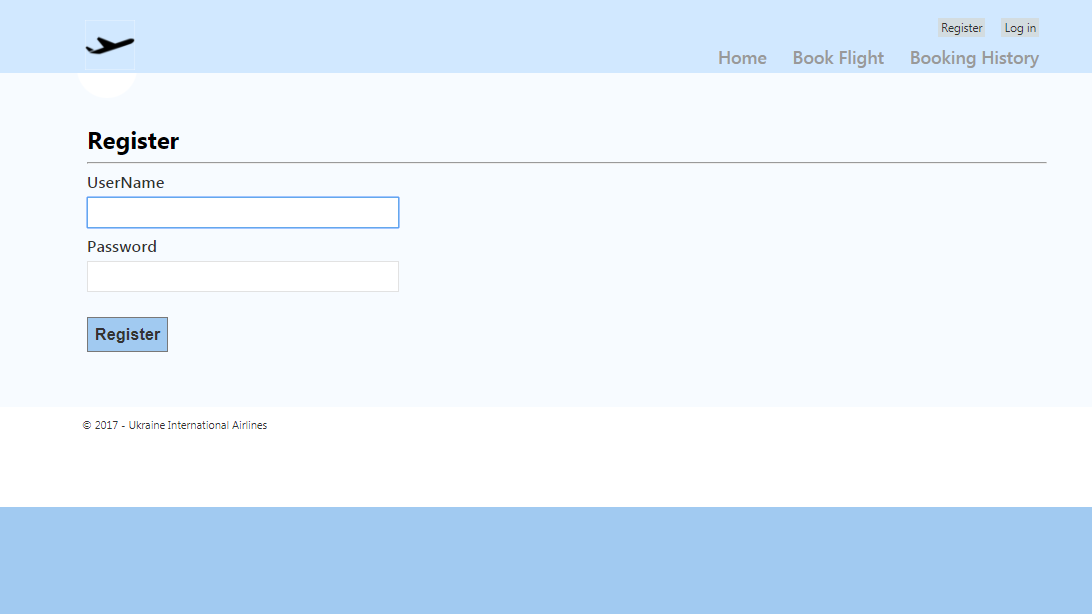
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Element** | **Test Case ID** | **Test Condition** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| Book Flight | D01 | User clicked on “Book Now” on specific flight with logged in | User selected a flight to book | User has successfully booked the flight | User has successfully booked the flight | Pass |
| Book Flight | D02 | User clicked on “Book Now” on specific flight without login | User selected a flight to book without login | System navigate users to login page | System navigate users to login page | Pass |

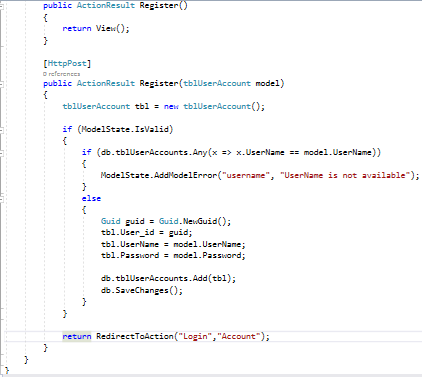
Booking History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Element** | **Test Case ID** | **Test Condition** | **Description** | **Expected Result** | **Actual Result** | **Status** |
| Booking History | E01 | User clicked on “Booking History” with login. User has booked a flight before. | User selected on “Booking History” with login | System shown user booking history. | System shown user booking history. | Pass |
| Booking History | E02 | User clicked on “Booking History” with login. User has not booked any flight before. | User selected on Booking History with login but never booked a flight. | System shows user “You have not booked any flight” | System shows user “You have not booked any flight” | Pass |
| Booking History | E03 | User clicked on “Booking History” without login | User selected on “Booking History” without login | System navigate users to login page | System navigate users to login page | Pass |

## Screenshot and Relevant Codes

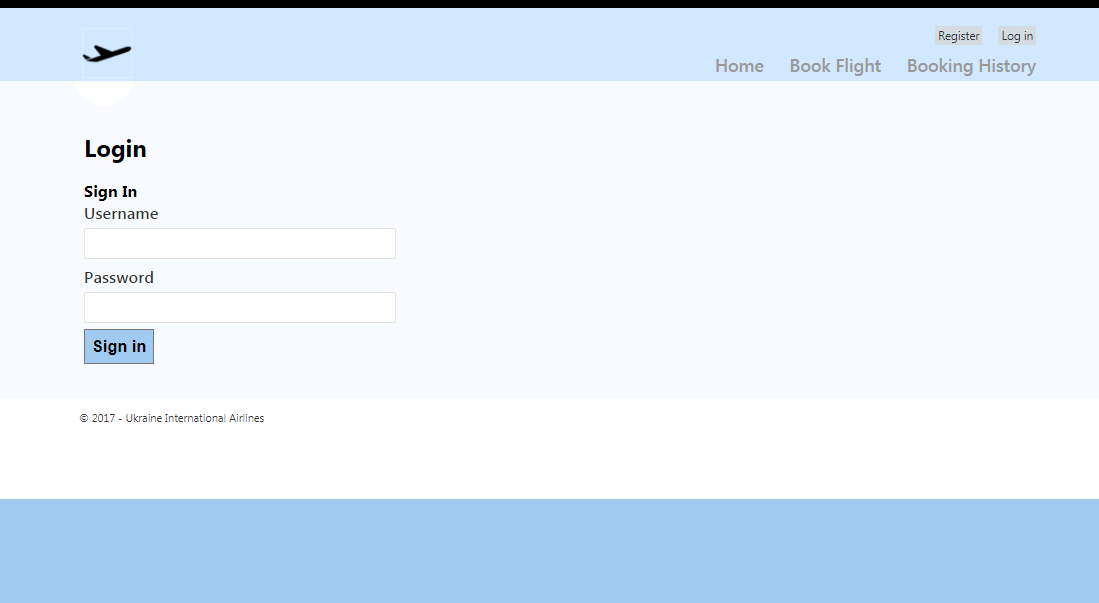
Register

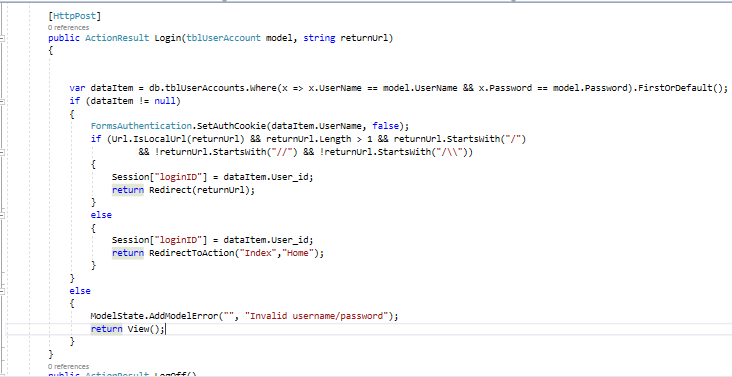




Above codes shows the actual implementation for register of the system. In detail, user will input username and password as the identification to register into the system. After user entered, it will be compared with database to check if the inputted username is already existed. User will be successfully registered to the system if the username is still available.

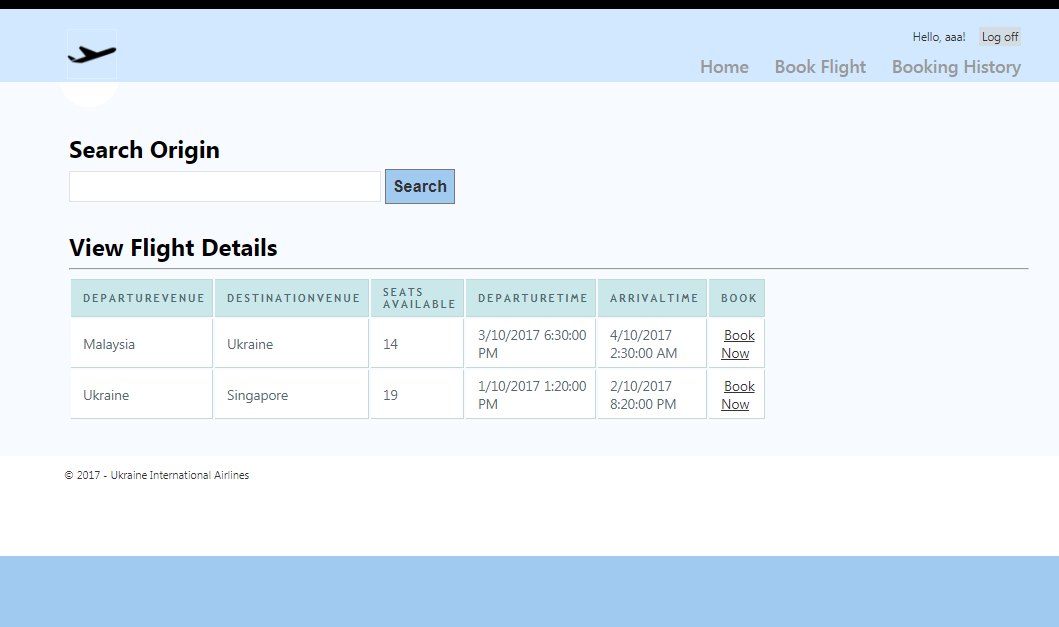
Login

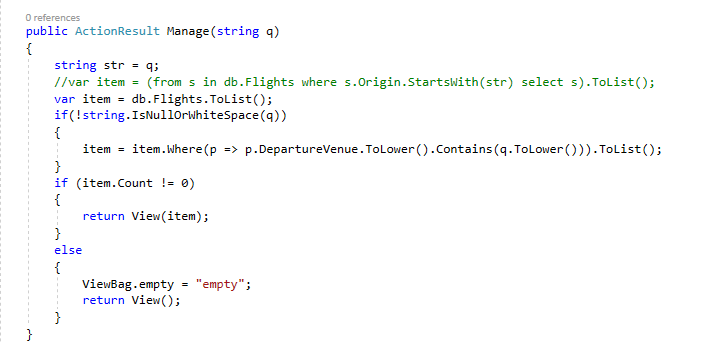




Above codes shows the actual implementation for login of the system. In detail, user will input username and password as the credential to login into the system. After user entered, it will be compared with database to validate the credentials. User will be able to login into the system and these data will store into cookie as well.

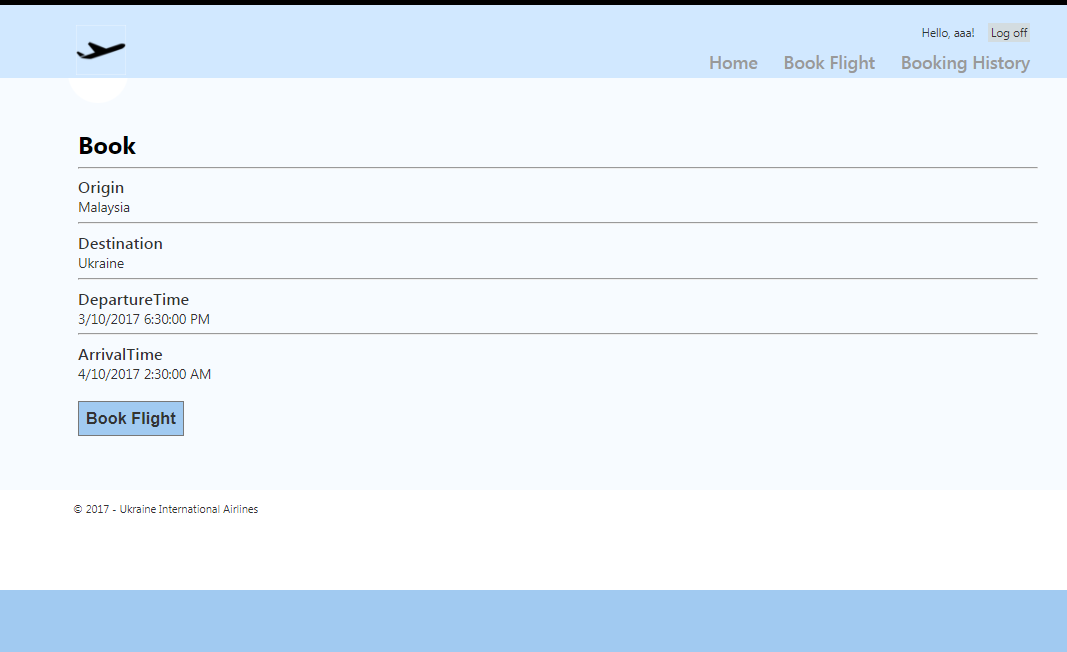
Search Flight

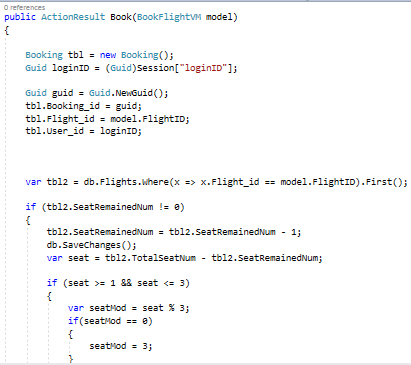




Above codes shows the actual implementation for searching departure place of a flight. In detail, user will input departure place in the search field. After user entered, it will be retrieved and checked with database. System will show users the results after checking with the database.

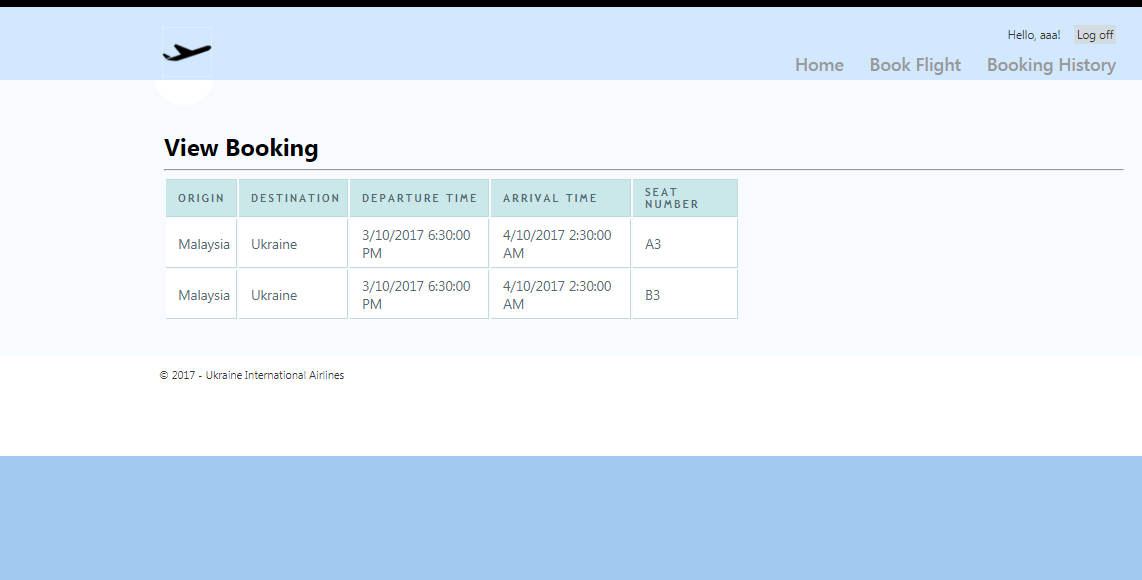
Book Flight

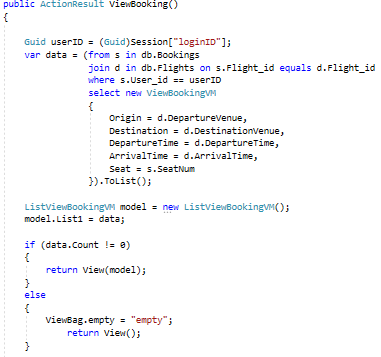




Above codes shows the actual implementation for booking flight. In detail, user will select book now on specific flight in the table. System will show users how many seats are remaining. Once user has booked a flight, user information such as Booking ID, Flight ID, User ID will be stored into the database. At the same time, system will generate the Seat Number which follows A1…B1… and so on and saved it to the database.

Booking History

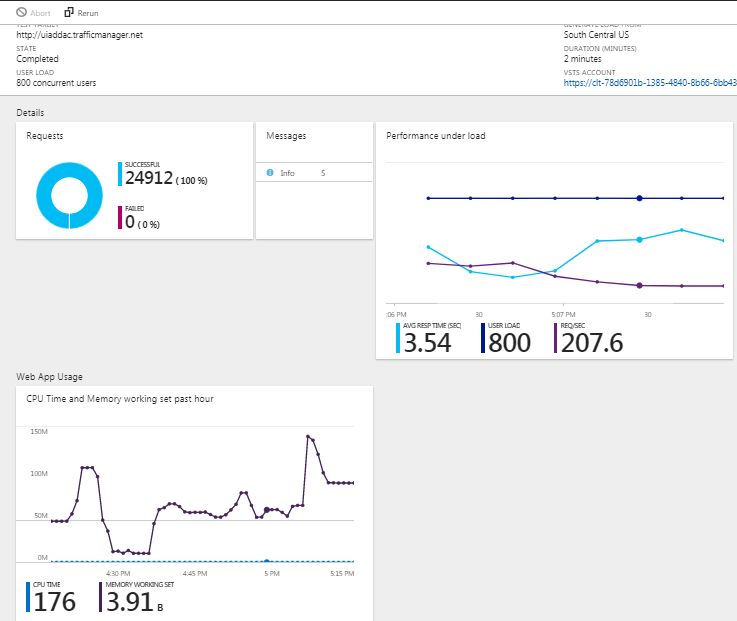


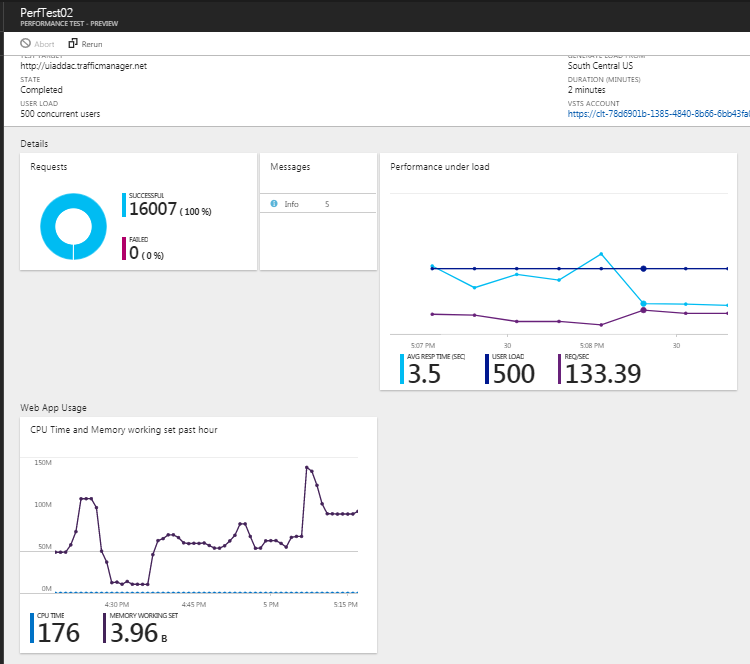


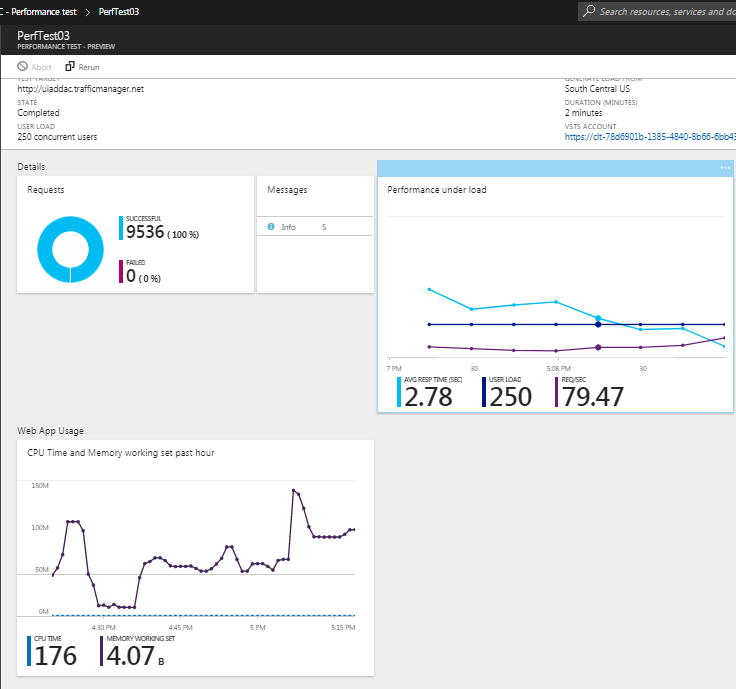
Above codes shows the actual implementation for booking history. In detail, user will select view history button. After that, system will retrieve booking information from the database. If there is no information, user will receive a message saying that “You have not booked any flight”. If user has booked a flight, user will be shown with a table showing the booking information that he/she has booked.

## Performance Testing

UIA is a big company which they could not disappoint their respected clients by comprising performance. Therefore, it is very important for them to perform performance testing to ensure customer likes their web services. Below are some diagrams and statistics of performance testing of the web application.







Above diagrams show three different tests with different amount of users in 2 minutes. Each of them range from 250 users, 500 users and 800 users. From above diagrams, we have tabulated the data for easier to analyse.

|  |  |  |  |
| --- | --- | --- | --- |
| Concurrent User | 250 | 500 | 800 |
| Average Response Time (sec) | 2.78 | 3.5 | 3.54 |
| Fail | 0 | 0 | 0 |

## Managed Database

As the trend of developing an application has been changed, most companies today are looking at platform as a service (PaaS) to deploy their application. There are many factors that why they rely on PaaS to manage their application. Mainly, it is because of it allows users to manage, monitor, modify and other operations with just few clicks and all of them can be executed through platform.

What is Platform as a service (PaaS)

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications (Microsoft Azure 2017). Inside a PaaS, it have various tools which are helpful for administrators to use such as development tools, database management, business analytics and etc.

Advantages of PaaS

As we have mentioned, companies are deploying their applications via PaaS. Indeed, it comes with many advantages.

**Coding Time can be shorten**

By using PaaS, it comes with development tools and deployment tools. This greatly help developers to reduce the coding time as developers were required to code from scratch and ensuring the developed application will be compatible with its requirements. With PaaS, as it is rich with resources and tools, many components are compatible such as database, web application and other components making development and deployment much easier.

**Save cost by hiring less staffs**

With PaaS, companies can save cost by hiring less staffs. Typically, companies need to have a lot of talents to monitor and maintain a specific application. With PaaS, it already comes with different analytic tools which making monitoring and maintain process much easier. Besides, companies may work with different vendors that are very familiar with PaaS which directly save much costs in terms of hiring staffs.

**Compatible for different platforms**

Some PaaS come with the capability to allow developers develop for multiple platforms. With this, it greatly save developer’s time as they could easily develop an application in a specific platform that also capable to work in other platforms. As a result, developers may focusing on a specific platform and use available tools to deploy in other platform, resulting less administrative cost and time.

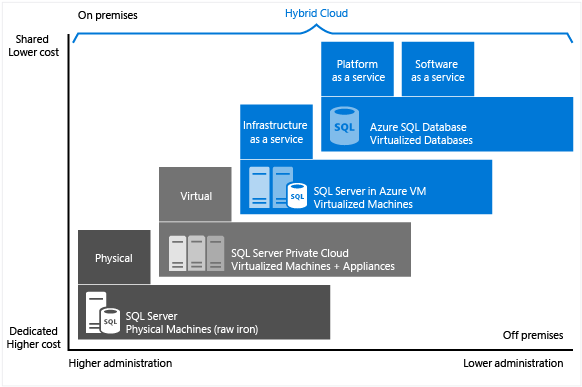
**Use sophisticated tools affordably**

As mentioned, PaaS comes with various sophisticated tools which allow companies to use these tools with a cheaper rate. In detail, PaaS usually has comprehensive tools which allow users to pay on subscription basis. It means that companies will need to only pay for their usages and choose only certain desire services. This means that companies will also able to combine different tools in their subscription.

**Support geographically distributed development teams**

As PaaS applies Cloud Technology, it allows users to work towards a project anytime and anywhere as long as they have internet access. With this, it is very beneficial for big companies such as Microsoft which has extremely high number of employees. In detail, Microsoft Human Resource Department could easily assign members to a project team comes from different locations. As a result, project teams may get talented members which already has experienced and familiar with the specific company.

Implementation & Discussion of Managed Databases (PaaS)



In PaaS, it comes with managed database which is not same as database in a Virtual Machine. In order to bring database to work in cloud environment, there are different techniques to achieve it.

First of all, putting a database inside a local server. This means that companies will need to maintain the local server and the database simultaneously which might ended up extremely high cost.

Apart from that, putting a database into a virtual machine in the cloud. This typically achievable by using Infrastructure as a Service (IaaS). Users subscribe to IaaS and in return will receive an infrastructure which allows users to use the infrastructure just like a local server. Users allowed to choose their desire hardware specifications from configuring the processors, memory, technology of hard drive, operating system and etc.

However, through PaaS, it is different with above scenarios. With PaaS, users could only upload their database to PaaS. With that, PaaS already comes with multiple tools from development, deployment to maintaining the database. With PaaS, users do not need to worry about hardware or infrastructure failure such as when using IaaS. Besides, above diagram shows that PaaS can save much costs when using Managed Database. Other than that, PaaS offers high security. Compare to a local service, users will need to worry about the encryption, firewall, security and other factors when deploying a database. With PaaS, users can check on the standards that these platforms have such as certain ISO standards. Other than that, PaaS also comes with statistics tools which allow users to monitor the system with statistics. At the same time, managed database in PaaS support multiple types of database such as MYSQL, Oracle and etc. This provide convenience to the users and do not need to worry about compatibility issues. Besides, with PaaS, users are allowed to deploy their database in multiple regions and the web applications. This is very helpful in traffic management as developers may route users from different regions to load the web applications at their nearest server to guarantee performance.

# Conclusion

In summary, the development of Ukraine International Airlines (UIA) was not an easy project. However, it is a big exploration and opportunity to design and deploy. By completing this project, it shows the importance and benefits of using Microsoft Azure. I was very glad that I able to learn to apply Cloud Technology as I strongly understand that the trend of IT Industry is now Cloud.

# References

Microsoft Azure 2017, *Autoscaling*, Microsoft Azure, viewed 5 October 2017, <https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling>.

Microsoft Azure 2017, *Scale instance count manually or automatically*, Microsoft Azure, viewed 7 October 2017, <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/insights-how-to-scale>.

Microsoft Azure n.d., *Create a Traffic Manager profile*, Microsoft Azure, viewed 6 October 2017, <https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-create-profile>.

Microsoft Azure n.d., *What is PaaS?*, Microsoft Azure, viewed 4 October 2017, <https://azure.microsoft.com/en-us/overview/what-is-paas/>.

Rogue Wave n.d., *The MVC Design Pattern*, Rogue Wave, viewed 6 October 2017, <http://docs.roguewave.com/stingray/11/html/sflug/8-2.html>.

Shelest, A 2009, *Model View Controller, Model View Presenter, and Model View ViewModel Design Patterns*, Core Project, viewed 6 October 2017, <https://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod>.

TutorialsTeacher.com n.d., *ASP.NET MVC Tutorials*, TutorialsTeacher.com, viewed 5 October 2017, <http://www.tutorialsteacher.com/mvc/asp.net-mvc-tutorials>.