

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT071-3-5-3-DDAC**

**Developing and Designing Cloud Applications**

Name: Prmeshpal Singh a/l K. Jas-pal Singh

TP Number: TP037198

Intake Code: UC3F1710SE

Module Name: Developing and Designing Cloud Applications

Project Title: Ukranian Online Flight Booking System

Lecturer: Dr. Kalai Anand A/L Ratnam

Date Completed: 9th October 2017

# Acknowledgement

It can be said that this project has been an experience as it brought some challenges that needs to be worked upon in this assignment. Cloud is the latest technology that is very much in talk and being use abundantly almost everywhere. This is all due to the Microsoft Azure platform which is although new but very much useful to be experimented with not forgetting the abundance of features in it. The assignment could not be accomplished without the help of my lecturer Dr Kalai Anand Ratnam for guiding us through the process of publishing and maintaining our web application on the cloud not forgetting to thank my course mates for sharing valuable information about the subject.

Table of Contents

[Acknowledgement 2](#_Toc494969908)

[Introduction 4](#_Toc494969909)

[Background Information 4](#_Toc494969910)

[Objectives 5](#_Toc494969911)

[Scopes 5](#_Toc494969912)

[Requirement Specification 5](#_Toc494969913)

[Project Plan 7](#_Toc494969914)

[Design Pattern 8](#_Toc494969915)

[Justification 8](#_Toc494969916)

[Design 10](#_Toc494969917)

[Architectural Diagram 10](#_Toc494969918)

[Design Considerations 13](#_Toc494969919)

[Secure Access 13](#_Toc494969920)

[Modelling 13](#_Toc494969921)

[Implementation 19](#_Toc494969922)

[Publishing to Azure. 19](#_Toc494969923)

[Business Continuity for Optimization and performance Enhancement. 24](#_Toc494969924)

[SQL Server Database 26](#_Toc494969925)

[Testing Cloud Application (Performance Testing) 28](#_Toc494969926)

[Test Plan 29](#_Toc494969927)

[Conclusion 31](#_Toc494969928)

[References 32](#_Toc494969929)

[Appendices. 33](#_Toc494969930)

[Project Screenshots. 33](#_Toc494969931)

[Credentials and Links 39](#_Toc494969932)

# Introduction

## Background Information

With the help of information technologies, the tourism industry has achieved great benefits such as improving business operations, better understanding of customer needs, better service delivery, reaching a greater number of customers and new markets and optimize its resources to increase its efficiency.

Nowadays, online shoppers are very much indecisive. If a website lags for even a few seconds, shoppers are just a couple of clicks away from many more options. First of all, 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.

Due to all of these reasons, the airline is very much keen on expanding into new markets. However, some problems with the current website has often restricted it from sufficiently handling customers not only in Ukraine but other countries as well. Some of the issues that usually happens is that the site experienced severe denial-of-service (DOS) attacks, which hurt site performance and reliability, and it did not have the performance needed to entertain visitors from many parts of the world.

Besides that, UIA has over the years used technology to reduce costs, innovate, and improve customer service. The transformations such as it has gone to a paperless cockpit and uses sophisticated software for analyzing fuel economy. The airline decided that it once again needed to innovate its way out of its web challenges. Dmitriy Prudnikov, Chief Information Officer at Ukraine International Airlines, realized that migrating the website out of UIA datacenters into a public cloud could solve all these problems. A solution of developing an Online Flight Booking System by the Ukraine International Airlines (UIA) using Azure platform which is very much compatible with open source software and operates better in a virtualized platform.

## Objectives

1. To develop a single tenant Online Flight Booking System web application hosted on Microsoft Azure as App service (Web App).
2. Design, develop and link the SQL Database to the web app.
3. Provide better performance through cloud design patterns.
4. Implement Azure Traffic Manager to control request from web clients, which will reroute to the nearest endpoints.
5. Analyse performance of web app with application insights using test plans and testing discussions.

## Scopes

1. To show the understanding of cloud computing terminology and how Microsoft Azure functions in this kind of web applications.
2. Learn, analyse and work with the Azure development environment.
3. Design, Implement and Deploy web application on Azure.
4. Architecturally design the application, utilizing Azure as the public cloud platform for hosting purposes.

## Requirement Specification

The following requirements are stated for the online flight booking system:

1. The searching and choosing of flights using the website application using all the operations provided.
2. To perform seat selection using SVG which is much effective and efficient to use as well as provides better performance.
3. To provide user details and user profile generation in almost no time with a single click.
4. To provide payment options which is very much easy to be done and use.
5. To be able to scale the solution to meet the needs of demands, providing high performance using all means together with managing the booking process from schedule search to booking confirmation.

**Major Functions**

The feature has the basic feature of providing a platform where an user can follow a simple process to do the booking. In terms of simplify the process of whole system below are the features available in the system,

1. Search Flight
2. Choose Flight
3. Provide User Details
4. SVG image seat selection
5. Make Payment
6. User Profile Generation

# Project Plan

The project plan of the Online Flight Booking System is where the users are able to use the system by logging into the system. The features that needs to be developed in the system are searching for flights, choosing of flights, provide user details, SVG image seat selection, make payment and user profile generation. All the specific requirements such as budget and duration has already been pre planned from the higher level to make sure that the project can be developed effortlessly and on time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Task ID | Task Name | Days | Start Date | End Date | Status |
| 1 | 1 | **Creating Database** | 2 | 20 July | 22 July | Done |
| 2 | 2 | **Web Design (Online Fight Booking System)** |  | 27 July | 11 August | Done |
| 3 |  | * Searching for flights | 2 |  |  |  |
| 4 |  | * Choosing of flights | 2 |  |  |  |
| 5 |  | * Provide user details | 2 |  |  |  |
| 6 |  | * SVG image seat selection | 4 |  |  |  |
| 7 |  | * Make Payment | 3 |  |  |  |
| 8 |  | * User Profile Generation | 3 |  |  |  |
| 9 | 3 | **Azure deployment** |  | 25 August | 28 August | Done |
| 10 |  | * Azure configuration | 4 |  |  |  |
| 11 | 4 | **Testing** |  | 15 August | 20 August | Done |
| 12 |  | * Performance Testing | 3 |  |  |  |
| 13 |  | * Testing for cloud storage using discussion | 3 |  |  |  |
| 14 | 5 | **Documentation** |  | 28 September | 2 October | Done |
| 15 |  | * Technical documentation | 3 |  |  |  |
| 16 |  | * Final report | 2 |  |  |  |

# Design Pattern

Imagine that you are developing the next addictive game, you upload it to the Marketplace and border it. Thousands and thousands of users will download it and your backend will fall with so much demand. How can we avoid it? With the design patterns that prepare your application for a cloud architecture, and specifically Microsoft Azure. In this session we will discuss the benefits of applying patterns in the development of cloud applications and the problems they try to solve: availability, scalability, resilience, security, etc.



## Justification

Lessening arrangement time, expanding consistency and empowering dexterity are the advantages you're probably going to expect when you investigate cloud-based ways to deal with your middleware application situations. By and large, conventional ways to deal with setting these sorts of situations have taken up a great deal of time and assets, requiring half a month and many individuals to fabricate.

The accompanying focuses must be considered when choosing how to apply this example:

• Cache information life expectancy: Many stores actualize a termination arrangement that makes information be negated and expelled from the reserve on the off chance that it isn't gotten to for a predefined period. For this example to be successful, ensure that the termination arrangement coordinates the entrance design for the applications that utilization the information. Keep in mind that reserving is more powerful for generally static information or information that is perused as often as possible.

• Data throwing: Most reserves are restricted in measure contrasted with the information store from which the information starts, and will exhaust the information if essential. Most reserves receive a later use arrangement to choose things to be exhausted, yet this can be redone. Design the worldwide lapse property and different properties of the reserve, and the termination property of each stored thing, to help guarantee that the reserve is beneficial. It may not generally be fitting to apply a worldwide purging approach for all components of the store. For instance, if a reserve component is extremely costly to recover from the information store, it might be advantageous to keep it in the reserve to the detriment of different components that are gotten to all the more much of the time,

• Preload the Cache: Numerous arrangements pre-populate the store with the information that you may require as a component of the startup procedure of an application. This example can be helpful if some of this information terminates or is erased.

• Consistency: Executing this example does not ensure consistency between the information store and the reserve. A thing in the information store can be changed whenever by an outside procedure, and this change isn't reflected in the reserve until whenever the thing is stacked into the reserve. In a framework that recreates information through information stores, this issue can turn out to be particularly genuine if synchronization happens as often as possible.

• Local Caching (In-Memory): A reserve could be neighborhood to an occurrence of the application and be put away in memory. The example can be valuable in this condition if an application more than once gets to similar information. Be that as it may, if a nearby reserve is private and diverse examples of the application have a duplicate of similar information in the store, the information could rapidly end up plainly conflicting between reserves, so it might be important to survey the arrangements when the information terminates put away in a private reserve and refresh them all the more regularly. In these situations the utilization of a disseminated reserving system might be proper.

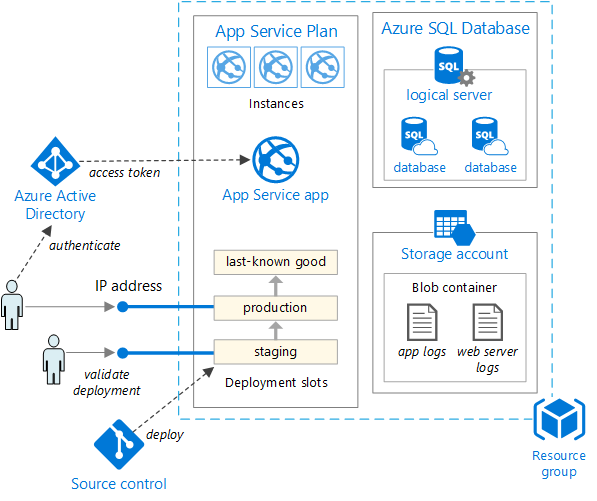
Furthermore, it is frequently hard to accomplish predictable obtainment comes about on account of the quantity of steps and the many-sided quality included. Notwithstanding adding to its increasing expenses in the course of the most recent 10 years, these difficulties keep organizations from accomplishing the sort of readiness they require in the present changing shopper markets.

So for Flight Booking application the design pattern to move on the azure is very much needed to provide higher availability.

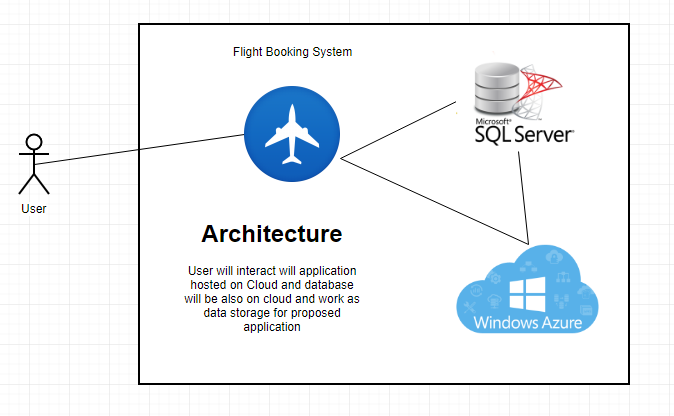
# Design

## Architectural Diagram

The following is a system with the SQL Server as Database which will also go on azure and work as storage of data for the application. Azure will provide the online platform for the application and will be hosted on two different locations in order to give better access. Below is the architecture:



*Figure 1: Online Flight Booking System cloud application design architecture*



*Figure 2: Online Flight Booking System cloud application design architecture data model*

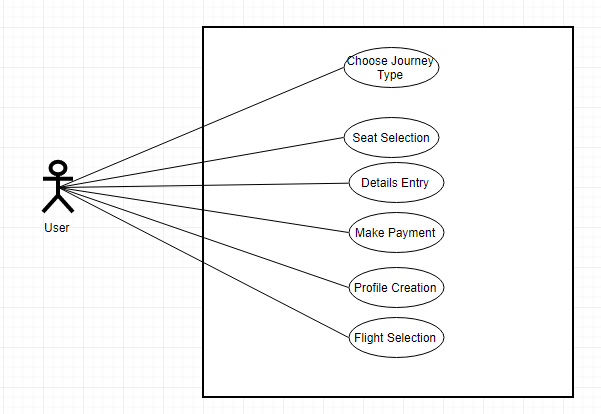
## Design Considerations

### Secure Access

The ASP.NET with the blend of C# programming language is most commonly used to develop and publish the particular Online Flight Booking System website. This is due to the prime reason that is the most convenient, simplest and most easy to understand platform to develop web applications as it very much uses the Drag and Drop tools as well as interactive design of web pages that is given to use by the Visual Studio IDE makes the process of designing excellent websites much more straightforward. The features of ASP.NET such as WEB API’S, real time technologies and single site application makes the programming language to stand out from the others. This kind of features will very much be helpful in searching for information and in the end, providing secure access using the tools provided such as SSL and also a design surface, hundreds of controls and components let you rapidly build complex, powerful UI-driven sites with data access. (Microsoft, 2017).

## Modelling

**Use Case Diagram**



*Figure 3: Use Case diagram for Online Flight Booking System*

**Use Case Specification**

|  |  |
| --- | --- |
| **Use Case ID** | **UC001** |
| Use Case name | Choose Journey Type |
| Actors | Traveller |
| Description | Choose the journey type,dates and destinations fr fly |
| PreCondition | User should move to default.aspx page |
| PostCondition | Traveller will get list of flights to choose |
| Priority | 1 |
| Normal Flow | 1. Run the project 2. Provide all inputs 3. Click Button |
| Alternative Flow | None |
| Includes | Traveller |

|  |  |
| --- | --- |
| **Use Case ID** | **UC002** |
| Use Case name | Seat Selection |
| Actors | Traveller |
| Description | Choose seats |
| PreCondition | User should have choosen the flight already |
| PostCondition | Traveller will get list of seats to choose |
| Priority | 3 |
| Normal Flow | 1. Run the project 2. Provide details regarding flight needed 3. Choose Flight 4. Give all details 5. Choose seats inside flight 6. Click Button |
| Alternative Flow | None |
| Includes | Traveller |

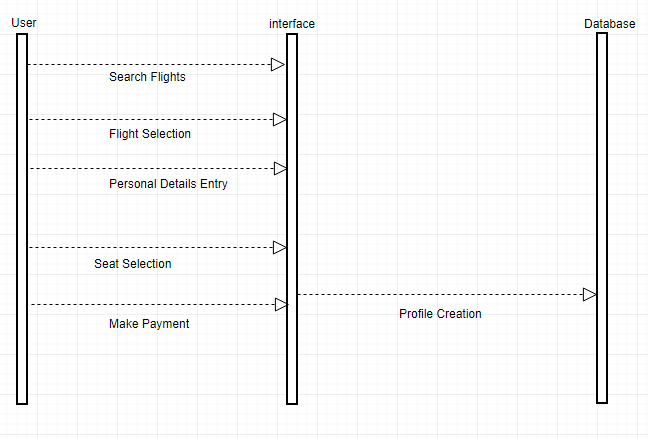
|  |  |
| --- | --- |
| **Use Case ID** | **UC003** |
| Use Case name | Details Entry |
| Actors | Traveller |
| Description | Enter all the personal details of traveller |
| PreCondition | User should have choosen the flight already |
| PostCondition | Traveller will input all his details |
| Priority | 2 |
| Normal Flow | 1. Run the project 2. Provide details regarding flight needed 3. Choose Flight 4. Give all details |
| Alternative Flow | None |
| Includes | Traveller |

|  |  |
| --- | --- |
| **Use Case ID** | **UC004** |
| Use Case name | Make Payment |
| Actors | Traveller |
| Description | Make payment for booking completion and profile creation |
| PreCondition | User should have choosen the flight already |
| PostCondition | Traveller will complete booking |
| Priority | 1 |
| Normal Flow | 1. Run the project 2. Provide details regarding flight needed 3. Choose Flight 4. Give all details 5. Choose seats inside flight 6. Make profile |
| Alternative Flow | None |
| Includes | Traveller |

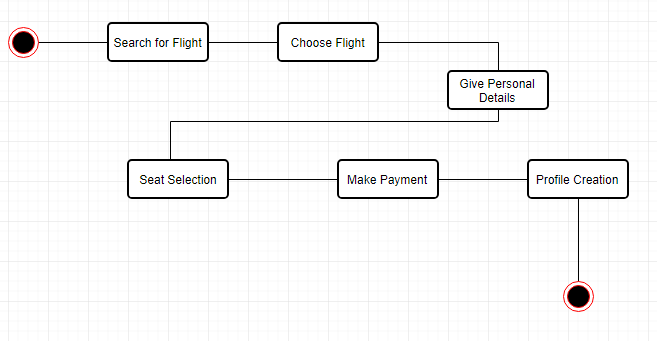
|  |  |
| --- | --- |
| **Use Case ID** | **UC005** |
| Use Case name | Profile Completion |
| Actors | Traveller |
| Description | Make payment for booking completion and profile creation |
| PreCondition | User should have choosen the flight already |
| PostCondition | Traveller will complete booking |
| Priority | 1 |
| Normal Flow | 1. Run the project 2. Provide details regarding flight needed 3. Choose Flight 4. Give all details 5. Choose seats inside flight 6. Booking complete |
| Alternative Flow | None |
| Includes | Traveller |

|  |  |
| --- | --- |
| **Use Case ID** | **UC006** |
| Use Case name | Flight Selection |
| Actors | Traveller |
| Description | Choose seats in flight |
| PreCondition | User should have choosen the flight already |
| PostCondition | Traveller will choose seats |
| Priority | 1 |
| Normal Flow | 1. Run the project 2. Provide details regarding flight needed 3. Choose Flight 4. Give all details 5. Choose seats inside flight |
| Alternative Flow | None |
| Includes | Traveller |

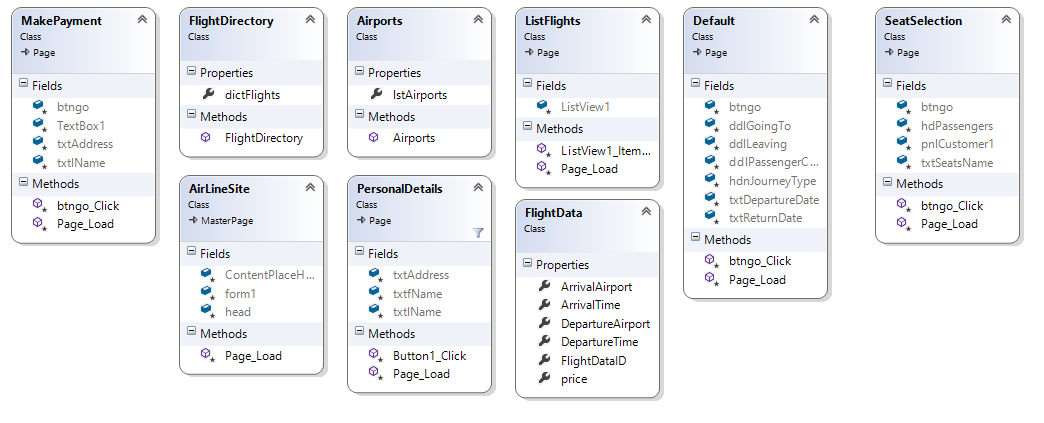
**Sequence Diagram**

****

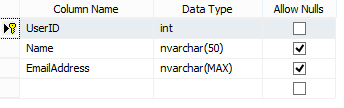
**Activity Diagram**

****

**Class Diagram**



**Database Table**

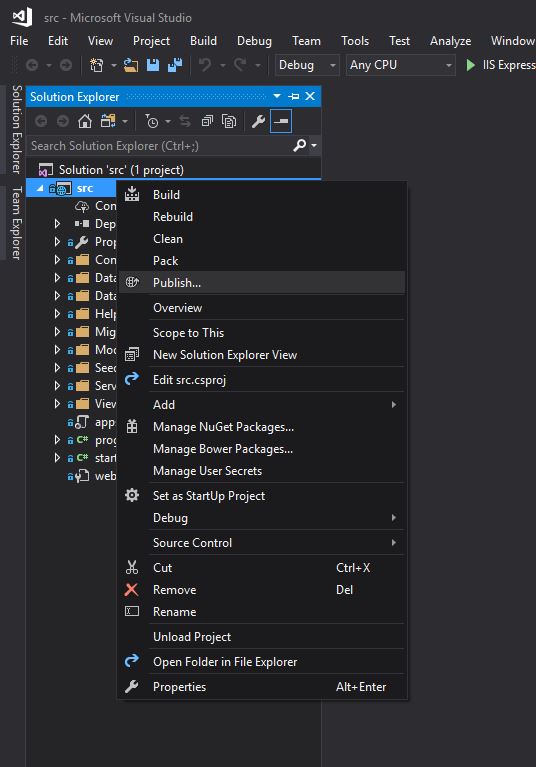


# Implementation

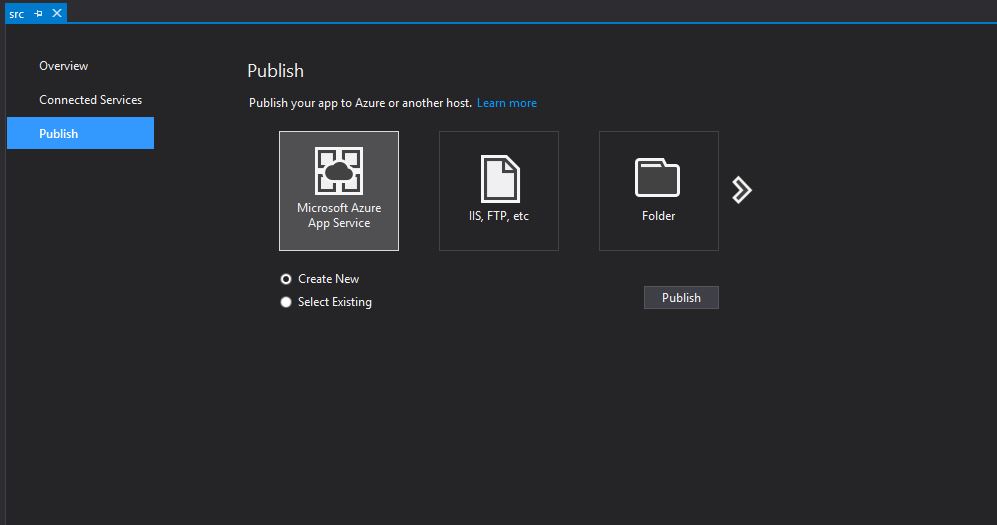
## Publishing to Azure.

Microsoft Azure is the cloud technology that provide the same features, services and tools to be used by the developers and IT professionals almost all around the world. This is due to the capability of supporting various systems, frameworks and many others. One of the benefit of using the Azure cloud platform to for publishing of websites is the pay as u go policy services that enables the developer to use the resources accordingly to the requirements and only pay for what they had used for it. For this project, the Online Flight Booking System only has one user which is the admin whereas the developer will have the responsibility to publish the web application to the cloud platform to make the system available to all for use. (Perkins, 2017).

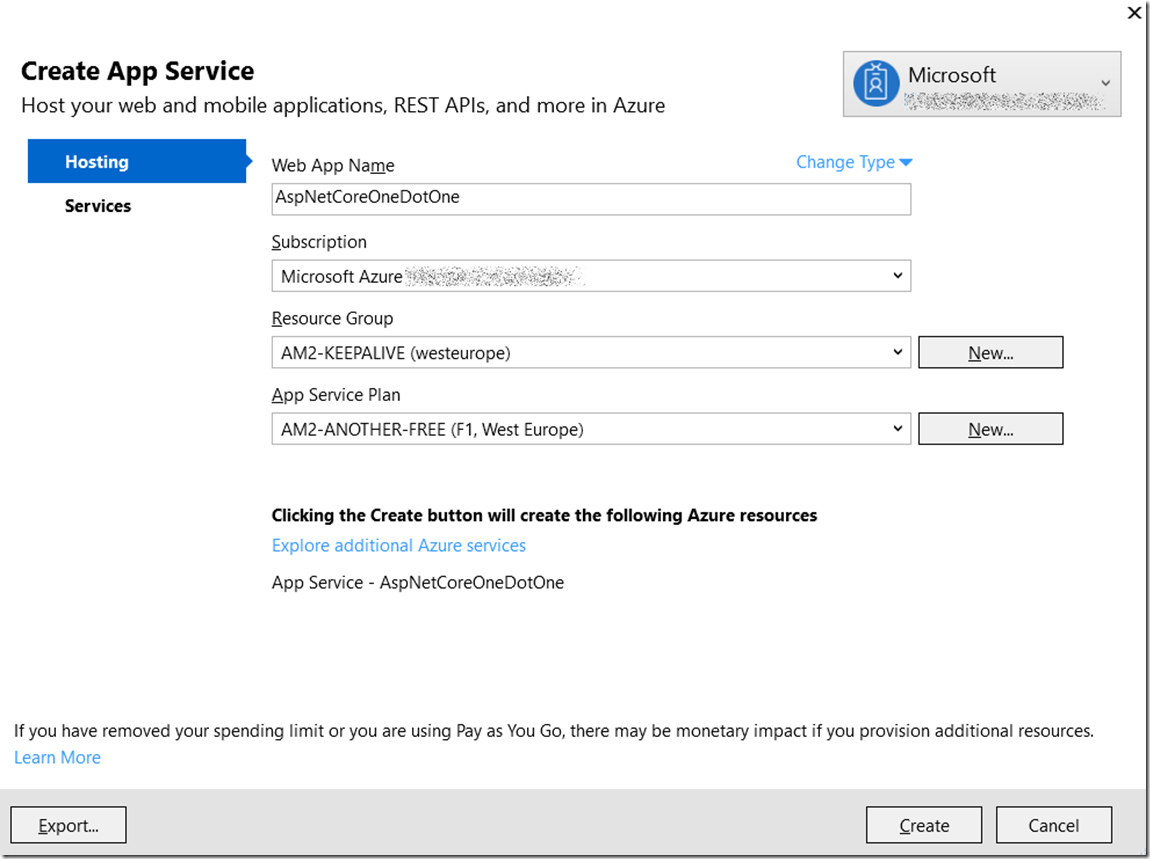
First of all, click on the solution file and choose publish. After that, by clicking on the publish button, the user will be displayed with the forms where Microsoft Azure App Service is selected then Create New radio button is chosen and new the Publish button is pressed for the creation of a new publish profile. Next, after signing up to Azure, the user will be taken to the Create App service wizard where a form will need to be filled accordingly. In there, the user will have to choose a name, resource group and the App Service Plan for the purpose of deploying the website application by clicking on the new button where the Application Service Plan, location and size will be displayed. Lastly, the resource group as well as the website application is deployed to the cloud platform and is opened in the default browser for viewing not forgetting running of application purposes. (Taylor, 2017).



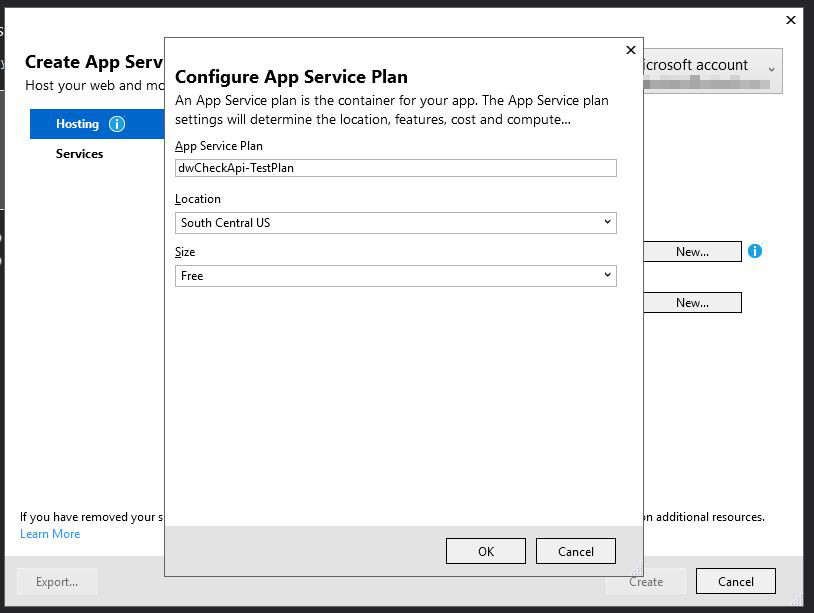
*Figure 4: Start Web Publishing*



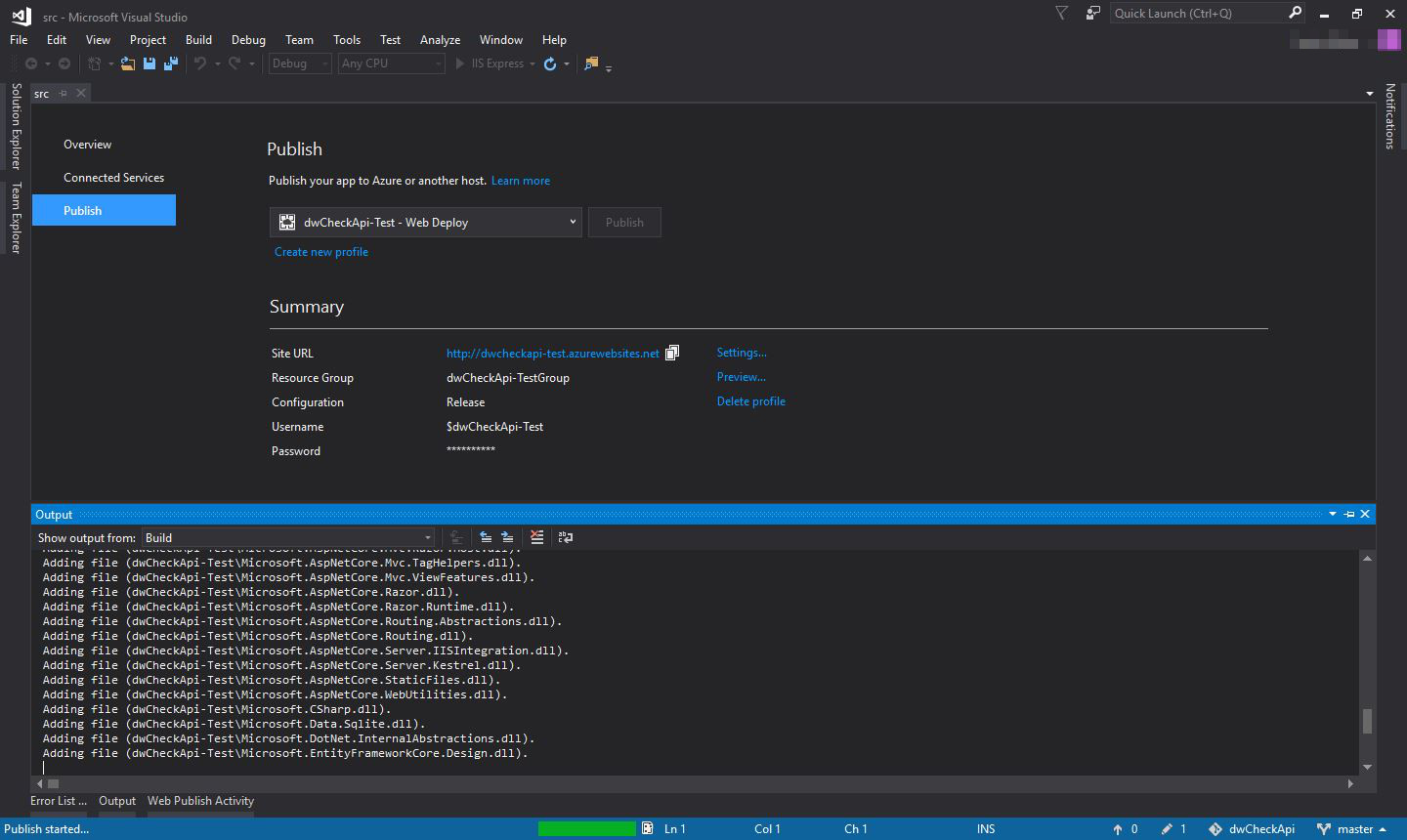
*Figure 5: Configure Resource Publish*



*Figure 6: Create App Service Plan*



*Figure 7: Configure App Service Plan*



*Figure 8: Publish Project*

## Business Continuity for Optimization and performance Enhancement.

**Traffic Manager Profile Creation**

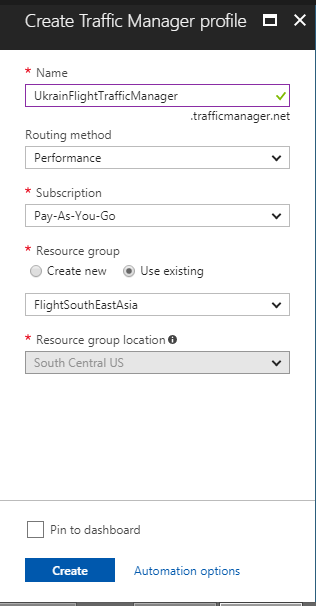
The azure traffic manager, takes into consideration the dispersion of client movement for benefit endpoints in various data centers. Administration endpoints upheld by activity director incorporate Azure VM's, Web Apps, and cloud administrations. Activity Manager additionally can be utilized with outside non Azure endpoints.

Traffic Manager likewise utilizes the Domain Name System (DNS) to guide customer solicitations to the most suitable endpoint in light of an activity – steering strategy and the strength of the endpoints. Activity Manager gives a scope of movement – directing choices to suit diverse application needs and programmed failover models. Activity Manager is versatile to disappointment, including the disappointment of a whole Azure area (Microsoft Azure, 2017).

The Microsoft Windows Azure Traffic Manager permits the control of distribution of network traffic from service endpoints to your Cloud Services and VMs hosted within Windows Azure or different datacentres. Besides that, it also is said that the service endpoints that are backed by Traffic Manager also comprises of include Azure VMs, Web Apps, and cloud services. Other than that, Traffic Manager can also be implemented with the external, non-Azure endpoints. The Windows Azure Traffic Manager also comes with a number of different methods of distributing internet traffic among two or more cloud services or VMs, all accessible with the same URL, in one or more Windows Azure datacentres. In general, it is seen as an allocation of DNS service that knows which Windows Azure services are sitting behind the traffic manager URL and administers requests based on three possible profiles which are failover, performance and round robin. (Balliauw, 2013).

Next, Traffic Manager adopts the Domain Name System (DNS) in order to channel the client requests to the most appropriate endpoint in accordance to a traffic-routing method and the health of the endpoints. Traffic Manager arranges a range of traffic-routing methods and endpoint monitoring options to conform with different application needs and automatic failover models due to the Traffic Manager resiliency to failure, including the failure of an entire Azure region.

For the creation of the Traffic Manager, the traffic manager profile is examined in the add button which is then created. There, the information required to be filled in the create traffic manager profile is filled and then distributed to the appropriate locations which is Southeast Asia and Western Europe accordingly. For that, both the endpoints were connected where the traffic manager’s overview was assessed for the purpose of confirming that both the endpoints were online and running well and then the DNS name was saved for the testing purpose of the traffic manager routing as well as the getting into the system using the online method which is the internet. (Microsoft Azure, 2017).



*Figure 9: Create Traffic Manager Profile*

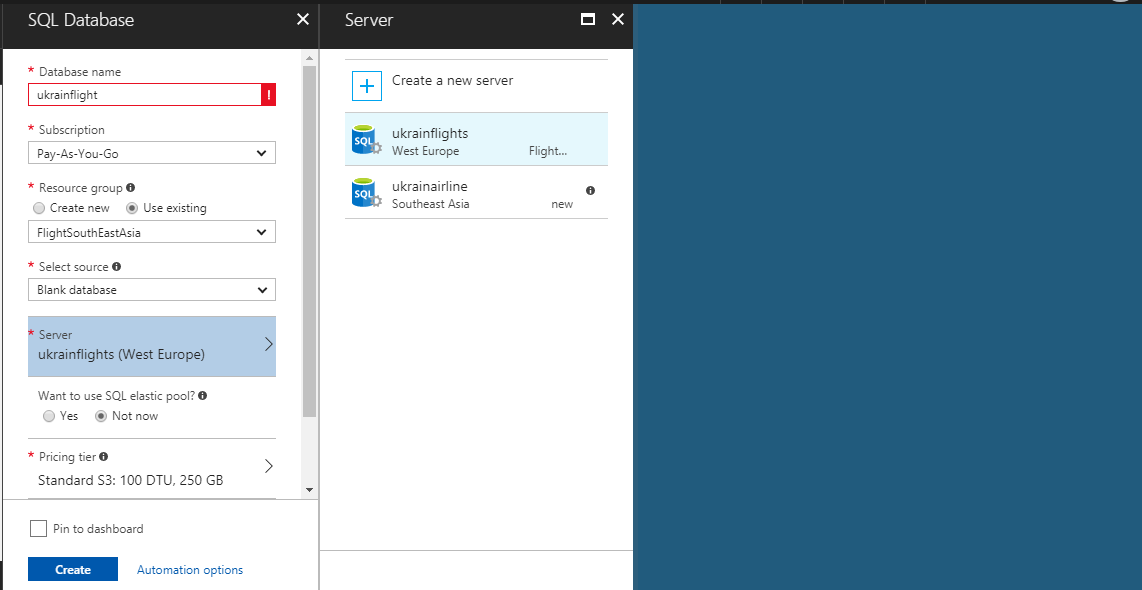
## SQL Server Database

Microsoft SQL Azure database is a cloud based relational database service that is develop on SQL Server technologies and operates in Microsoft data centres on hardware that is owned, hosted and maintained by Microsoft. It is also seen as a relational database-as-a-service (DBaaS) introduced in the Azure cloud that falls into the industry categories of Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS). In general, Azure SQL Database is able to increase the number of databases that are handled by a single IT or development resources adequately. Other than that, built-in-services and functionality helps the SQL database by developing directly on the service where using the SQL Database, you pay-as-you-go with options to scale up or out for greater power with no interruption. (Microsoft Azure, 2017).

For the creation of a SQL database on Azure, the add button is clicked and the SQL database is searched for where the create button is then pressed. Next, the following data were recorded into the form for a new SQL Server creation for hosting of database purpose. After that, the particular web application needs to be connected to the database where it can be done by obtaining the connection string and then to copy paste it to the webconfig file for it to function as well as run well.

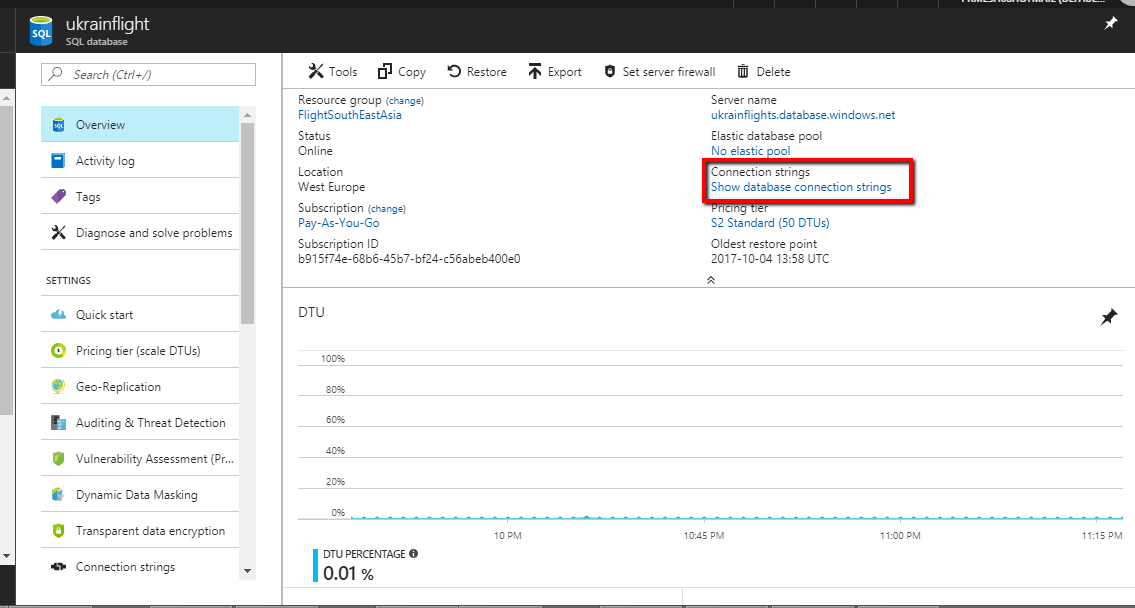
**Database creation on Azure**

First create the database by filling all the details from below screen. Once all input will be given then a database user will also get added with database.



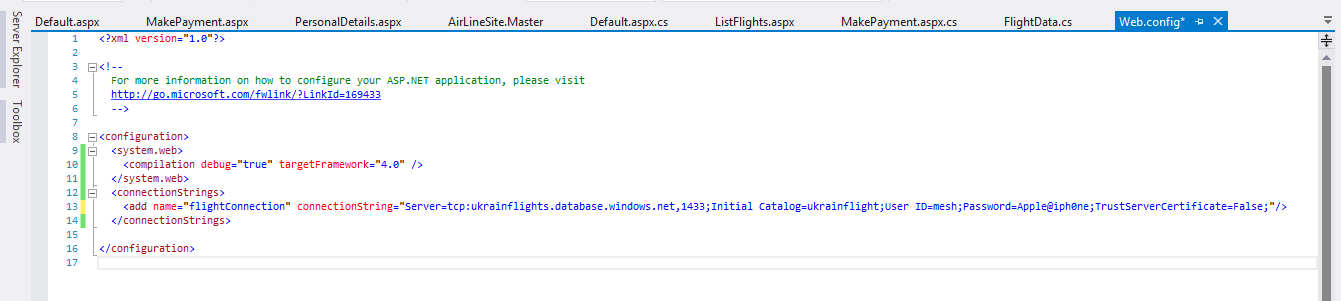
*Figure 10: Creating and Configuring SQL Database.*

Once database is added successfully get the connection string add connection string into the web.config file:



*Figure 11: Post configure SQL Database setup.*

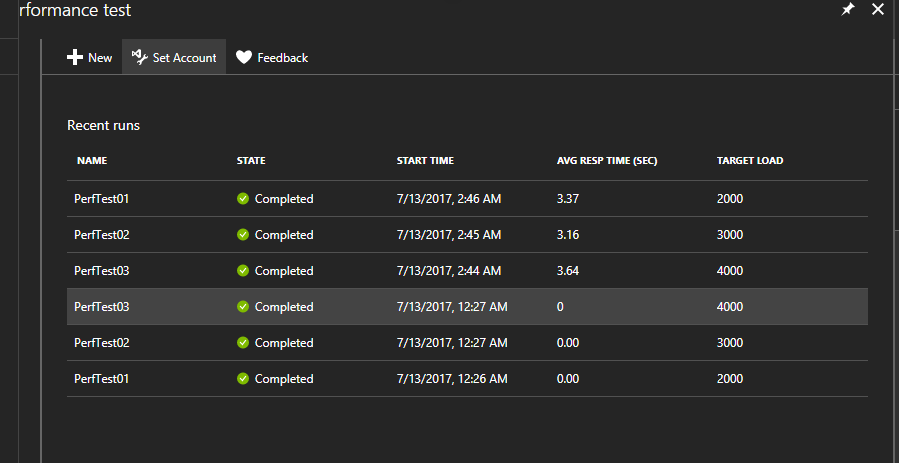
**Web.Config Entry for database**



*Figure 12: Linking SQL Database string to web application.*

# Testing Cloud Application (Performance Testing)

The Azure performance test tools, are used by the developer to perform testing of the system throughput which is the average response time per second and the CPU processing time and working memory required.



*Figure 13: Performance tests conducted.*

## Test Plan

The objective of the tests is not to prove that the system works, but to find mistakes. Empirically we have seen that most of the errors in the functions are found in the input of nulls, empty strings and limit values. By limit values we understand that if a function supports a range of values, the limits are the boundaries of those values. Normally in all tests the first step has been to prove that it does what it has what to do with the expected values. Then they have normally been tested with null values, empty strings and limit values.

We have not tried all the classes, only those that had some complexity. Usually a test by method has been done although in some cases they have been had to do more.

Below are the test cases performed in the system along with their results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Name | Steps | Expected Result | Result |
| 1 | Search Flight | 1. Choose Round  2. Select Departure Airport 3. Select Arrival Airport 4. Chose the Departure Date 5. Choose the Return Date 6. Choose the number of Passenger 7. Click on Go Button | Page should move to display available flights. | Working as Expected |
| 2 | Choose Flight | 1. Choose flight from the given list | Page should move to ask user to enter traveller’s details. | Working as Expected |
| 3 | Enter User Details | 1. Enter the first and last name 2. Enter Address 3. Click on Seat Selection Button | Page should move to seat selection UI. | Working as Expected |
| 4 | Seat Selection | Choose seats from SVG Map Click on Make Payment Button | Seats will get added into the textbox and page will move to payment page | Working as Expected |
| 5 | Make Payment | Choose Credit or Debit Card Enter Card No Enter CVV Code Click on Make Payment Button | User Profile should be created and Payment done confirmation | Working as Expected |

# Conclusion

In conclusion, it can be said that this assignment of Online Flight Booking System was very much a challenge because it was the very first time of using the cloud services provided by Azure which was also a relative new technology. However, the developer managed to complete the assignment where the end product of a working website was achieved with all the resources and others that have been learned. Furthermore, the developer will still continue to learn and work on mastering about the other Azure features as well as tools for future cloud projects implementation. A huge thanks to Dr. Kalai Anand Ratnam for enriching us with this cloud technology subject on how to work with Azure and working with them on the cloud platform. Last but not the least, this knowledge would very much be useful and helpful for the forthcoming project developments as well as deployments on cloud for software engineers in future.

# References

Balliauw, M., 2013. *Maarten Balliauw.* [Online]   
Available at: https://dzone.com/articles/windows-azure-traffic-manager  
[Accessed 28 September 2017].

Microsoft Azure, 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 3 October 2017].

Microsoft Azure, 2017. *Overview of Traffic Manager.* [Online]   
Available at: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview  
[Accessed 29 September 2017].

Microsoft, 2017. *ASP.NET overview.* [Online]   
Available at: https://docs.microsoft.com/en-us/aspnet/overview  
[Accessed 24 August 2017].

Perkins, B., 2017. *How to deploy an ASP.NET Core 1.1 application to an Azure App Services Web App using Visual Studio 2017.* [Online]   
Available at: https://blogs.msdn.microsoft.com/benjaminperkins/2017/03/21/how-to-deploy-an-asp-net-core-1-1-application-to-an-azure-app-services-web-app-using-visual-studio-2017/  
[Accessed 15 September 2017].

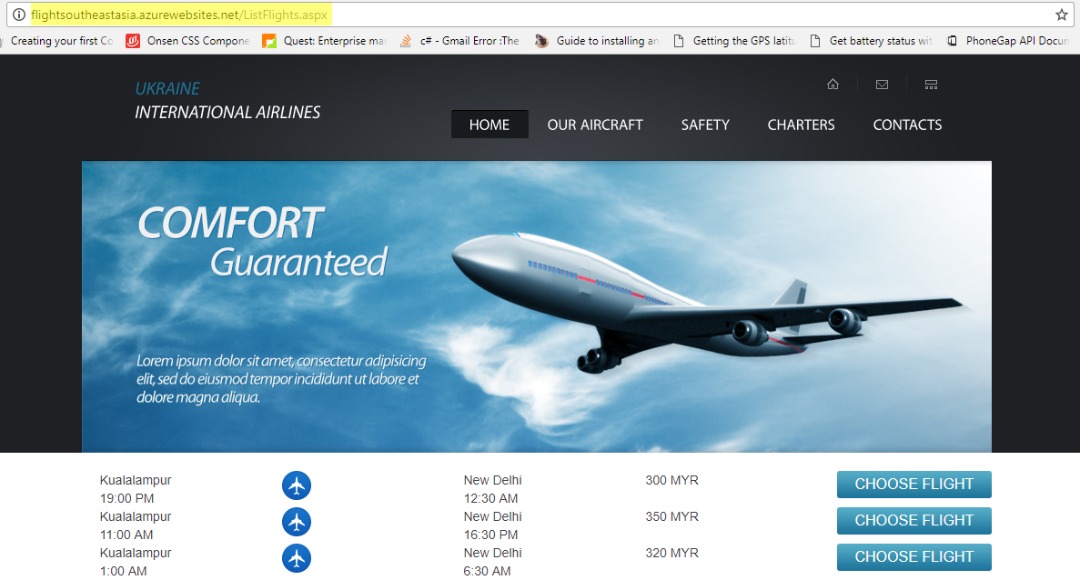
Taylor, J., 2017. *Publishing Your First .NET Core Application to Azure.* [Online]   
Available at: https://dotnetcore.gaprogman.com/2017/03/30/publishing-your-first-net-core-application-to-azure/  
[Accessed 22 September 2017].

Taylor & Jamie, 2017. *Publishing Your First .NET Core Application to Azure.* [Online]   
Available at: https://dotnetcore.gaprogman.com/2017/03/30/publishing-your-first-net-core-application-to-azure/  
[Accessed 21 September 2017].

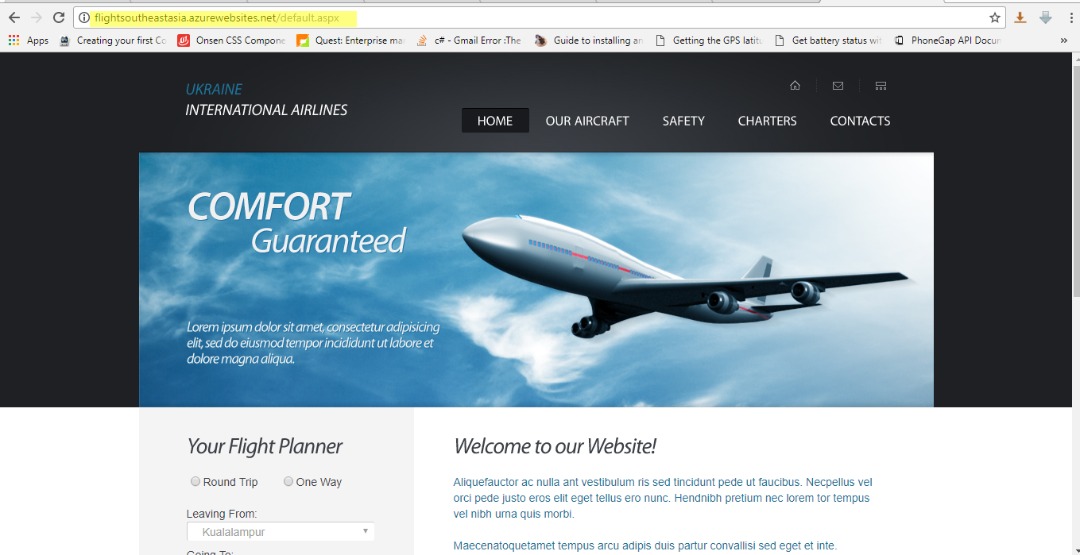
# Appendices.

## Project Screenshots.

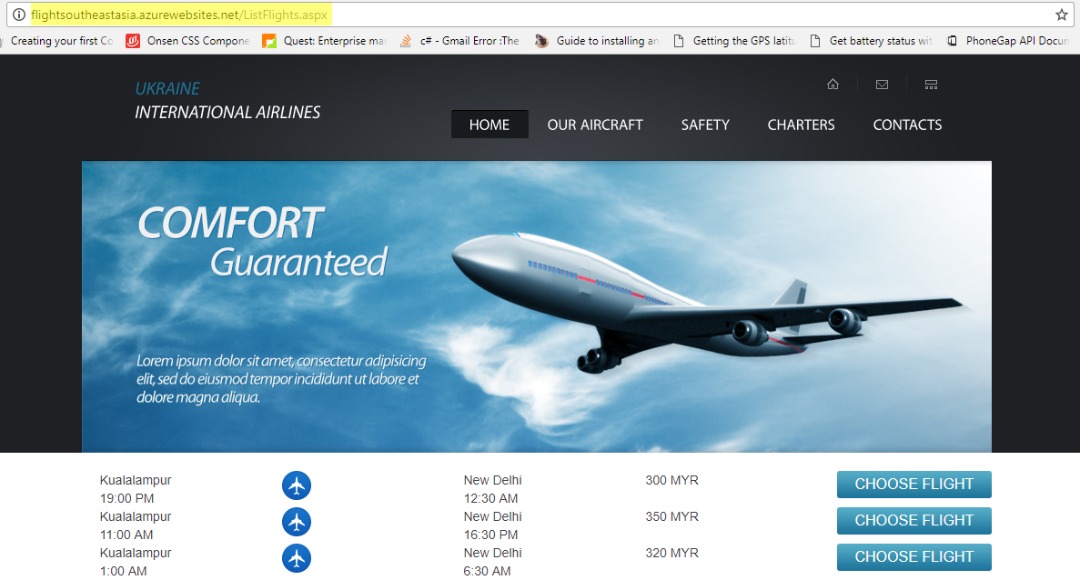
**Homepage**



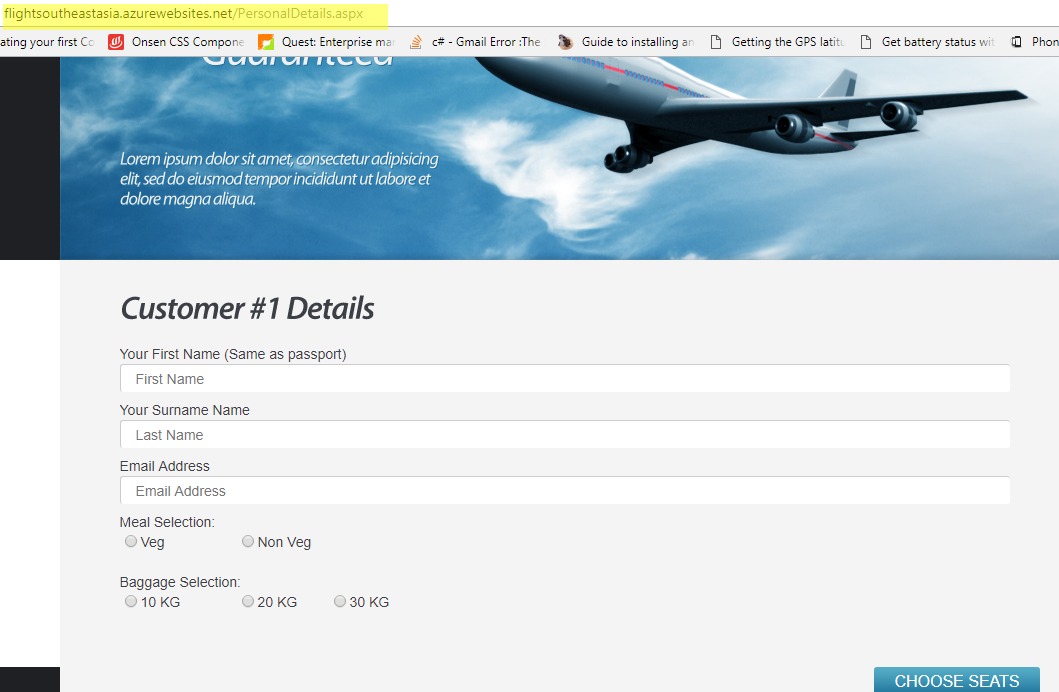
**Flight Planner**



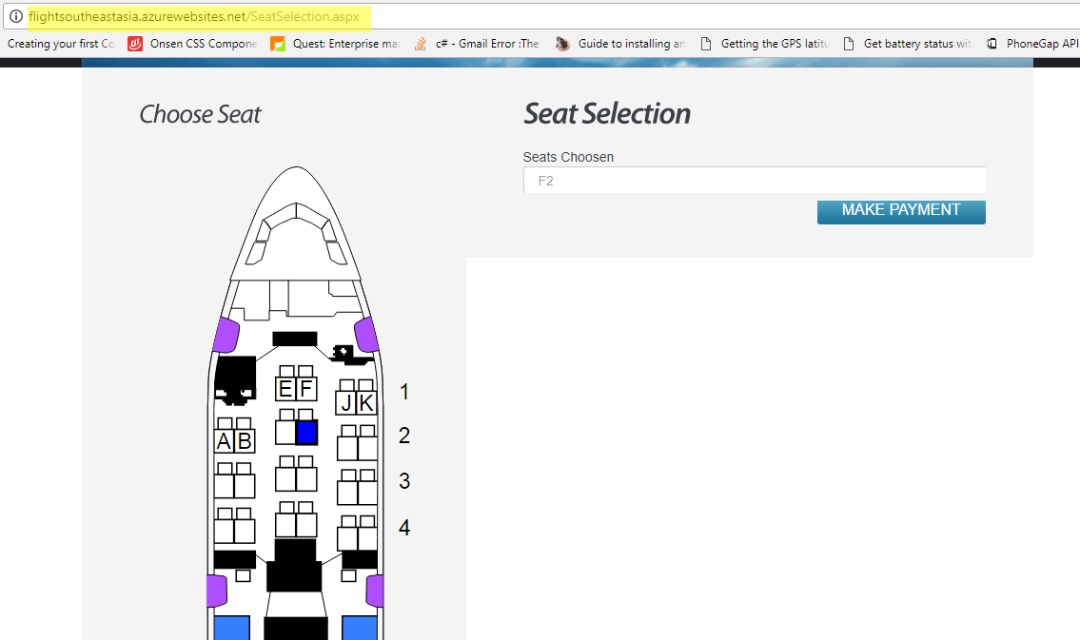
**Choose Flight**



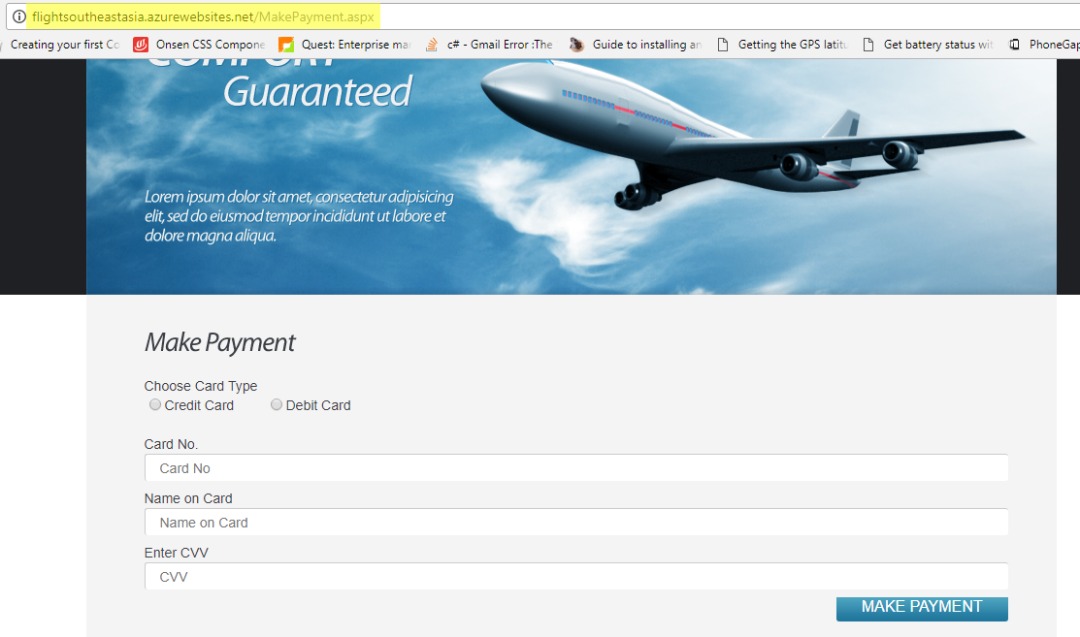
**Customer Detail**



**Seat Selection**



**Make Payment**



## Credentials and Links

Southeast Asia:

[http://flightsoutheastasia.azurewebsites.net/default.aspx](https://l.facebook.com/l.php?u=http%3A%2F%2Fflightsoutheastasia.azurewebsites.net%2Fdefault.aspx&h=ATOtDeMuuU1JtGfWvgPy0BhDXHLJ1-SwoD8GhbtbDtueB0UhVlShpsmkUon5HEMCOXBohu721wftfJ3oziuxrHbgZpH9NHwDxlNYtW1niPhsf3w0RvjpTNLdT242XqGvMKGi2JSJH-hcc97RqBlzu7Etah_3uQ)

Western Europe:

[http://flightwesterneurope.azurewebsites.net/default.aspx](https://l.facebook.com/l.php?u=http%3A%2F%2Fflightwesterneurope.azurewebsites.net%2Fdefault.aspx&h=ATOtDeMuuU1JtGfWvgPy0BhDXHLJ1-SwoD8GhbtbDtueB0UhVlShpsmkUon5HEMCOXBohu721wftfJ3oziuxrHbgZpH9NHwDxlNYtW1niPhsf3w0RvjpTNLdT242XqGvMKGi2JSJH-hcc97RqBlzu7Etah_3uQ)

Username: [prmesh08@hotmail.com](mailto:prmesh08@hotmail.com)

Password: mesh1231

**GitHub**

<https://github.com/Prmeshpal/UkrainFlightBooking>

**Video Link**

<https://www.screencast.com/t/YPfd8UmXoGP>