Table of Contents

[Table of Figure iii](#_Toc498953336)

[Introduction 1](#_Toc498953337)

[Aims 1](#_Toc498953338)

[Objectives 1](#_Toc498953339)

[Use Case Diagram 3](#_Toc498953340)

[Class Diagram 4](#_Toc498953341)

[Activity Diagram 5](#_Toc498953342)

[Entity Relationship Diagram 8](#_Toc498953343)

[Cloud Design Patterns 9](#_Toc498953344)

[Federated Identity Pattern 10](#_Toc498953345)

[Retry Design Pattern 11](#_Toc498953346)

[Implementation 12](#_Toc498953347)

[Creation of Active Directory 13](#_Toc498953348)

[Creating a Database Server 15](#_Toc498953349)

[Creating a Database 16](#_Toc498953350)

[Creation of a Web Application 17](#_Toc498953351)

[Deploying the Actual App on Microsoft Azure 18](#_Toc498953352)

[Screenshots 19](#_Toc498953353)

[Home Page 19](#_Toc498953354)

[Special Offers 20](#_Toc498953355)

[Login Panel 21](#_Toc498953356)

[User Panel 22](#_Toc498953357)

[Profile Modification 23](#_Toc498953358)

[Flight Schedule 24](#_Toc498953359)

[Booking 25](#_Toc498953360)

[Payment 26](#_Toc498953361)

[Online Check-in 27](#_Toc498953362)

[Testing 28](#_Toc498953363)

[Unit Testing 29](#_Toc498953364)

[Performance Testing 31](#_Toc498953365)

[Conclusion 32](#_Toc498953366)

[References 33](#_Toc498953367)

# Table of Figure

[Figure 1: Use case diagram, 2](#_Toc498588696)

[Figure 2: Class diagram 3](#_Toc498588697)

[Figure 3: Add activity 4](#_Toc498588698)

[Figure 4: Update activity 5](#_Toc498588699)

[Figure 5: Delete activity 5](#_Toc498588700)

[Figure 6: View activity 6](#_Toc498588701)

[Figure 7: ERD 7](#_Toc498588702)

[Figure 8: Federated identity pattern 9](#_Toc498588703)

[Figure 9: Retry pattern 10](#_Toc498588704)

[Figure 10: Creating active directory 12](#_Toc498588705)

[Figure 11: Providing active directory settings 13](#_Toc498588706)

[Figure 12: Creating database server 14](#_Toc498588707)

[Figure 13: Creating database 15](#_Toc498588708)

[Figure 14: Creating database 16](#_Toc498588709)

[Figure 15: Deploying the system 17](#_Toc498588710)

[Figure 16: Home page 18](#_Toc498588711)

[Figure 17:Special offers 19](#_Toc498588712)

[Figure 18: Login panel 20](#_Toc498588713)

[Figure 19: User panel 21](#_Toc498588714)

[Figure 20: Profile modification 22](#_Toc498588715)

[Figure 21: Flight schedule 23](#_Toc498588716)

[Figure 22: Booking 24](#_Toc498588717)

[Figure 23: Payment 25](#_Toc498588718)

[Figure 24: Check-in 26](#_Toc498588719)

# Introduction

This paper is concerned with describing and demonstrating principles and procedures involved in engineering and deploying cloud system. Throughout the course of this paper, an airline management system specialize for the cloud platform will be engineered and developed as an example for demonstrating cloud application development. The proposed system can be described as an application that facilitates management of daily tasks and routines in airline companies together with allowing customers to search flight, purchase ticker, and view their purchases in the system. The proposed cloud system will be targeted for Microsoft Azure platform and for that reason it will be developed using Microsoft ASP.Net as the server side development framework, and Microsoft SQL Server as the relational database management system to preserve and maintain system data. The next chapters of this paper will provide much greater information in this regard.

## Aims

Provides a Web Application for Ukraine International Airlines (UIA) into public cloud platform to serve their customers that beyond Ukraine and to prevent the denial-of-service (DOS) attacks, which hurt site performance and reality, and it did not have the performance needed to host visitors from many parts of the world.

## Objectives

To achieve the aims of this project, there are several things need to be done by the developer’s teams that will be listed in this objective session:

* To provide a system that could manage the airlines schedule.
* To provide a system that could show the flight schedule or list that can be seen by the users.
* To provide a system that only can be accessed by those who have the account to access the system only.

|  |  |  |  |
| --- | --- | --- | --- |
| **TASK NAME** | **DURATION** | **START** | **FINISH** |
| **Ukraine international Airlines (UIA)**  **Online Ticket Booking System** | **79 Days** | **Fri 09/01/17** | **Sat 11/18/17** |
| **Introduction** | **6 Days** | **Fri 09/01/17** | **Wed 09/06/17** |
| Company Background | 1 Day | Fri 09/01/17 | Fri 09/01/17 |
| Aims | 1 Day | Sat 09/02/17 | Sat 09/02/17 |
| Objectives | 1 Day | Sun 09/03/17 | Sun 09/03/17 |
| Scopes | 2 Days | Mon 09/04/17 | Tue 09/05/17 |
| Goals and Requirements | 1 Day | Wed 09/06/17 | Wed 09/06/17 |
| **Project Plan** | **2 Days** | **Thu 09/07/17** | **Fri 09/08/17** |
| Gantt Chart | 2 Days | Thu 09/07/17 | Fri 09/08/17 |
| **Design** | **24 Days** | **Sat 09/09/17** | **Mon 10/02/17** |
| Architectural Diagram | 4 Days | Sat 09/09/17 | Tue 09/12/17 |
| Use Case Diagram | 5 Days | Wed 09/13/17 | Sun 09/17/17 |
| Use Case Specification | 5 Days | Mon 09/18/17 | Fri 09/22/17 |
| Activity Diagram | 5 Days | Sat 09/23/17 | Wed 09/27/17 |
| Design Considerations | 3 Days | Thu 09/28/17 | Sat 09/30/17 |
| Modelling | 2 Days | Sun 10/01/17 | Mon 10/02/17 |
| **Implementation** | **40 Days** | **Tue 10/03/17** | **Sat 11/17/17** |
| Graphical User Interface | 20 Days | Tue 10/03/17 | Sun 10/22/17 |
| Deploy from Local to Azure | 20 Days | Mon 10/23/17 | Sat 11/11/17 |
| **Testing** | **6 Days** | **Sun 11/12/17** | **Fri 11/17/17** |
| Functional | 3 Days | Sun 11/12/17 | Tue 11/14/17 |
| Performance | 3 Days | Wed 11/15/17 | Fri 11/17/17 |
| **Conclusion** | **1 Day** | **Sat 11/18/2017** | **Sat 11/18/17** |

# Use Case Diagram

Analysis of the use case scenarios can be regarded as one the underlying approaches prior to proceeding with designing a software solution. The use case diagram can be regarded as a graphical illustration in which all required functionalities in a software system as well as their relationships with each involved entity are represented. In other words, the use case diagram can be regarded as an abstract blueprint which clarifies the overall purpose and image of a software system to developers. The following use case diagram has been developed for the proposed cloud system.



Figure 1: Use case diagram,

# Class Diagram

Classes can be regarded as the building blocks of the object orientation paradigm and for that reason, their modeling and the documentation of their hierarchical structure is of significant importance. The use case diagram is accordingly a graphical illustration which represents the hierarchical relationship between classes involved in a software solution together with showing the data and behavior which each class carries. The following class diagram has been developed for the proposed system.

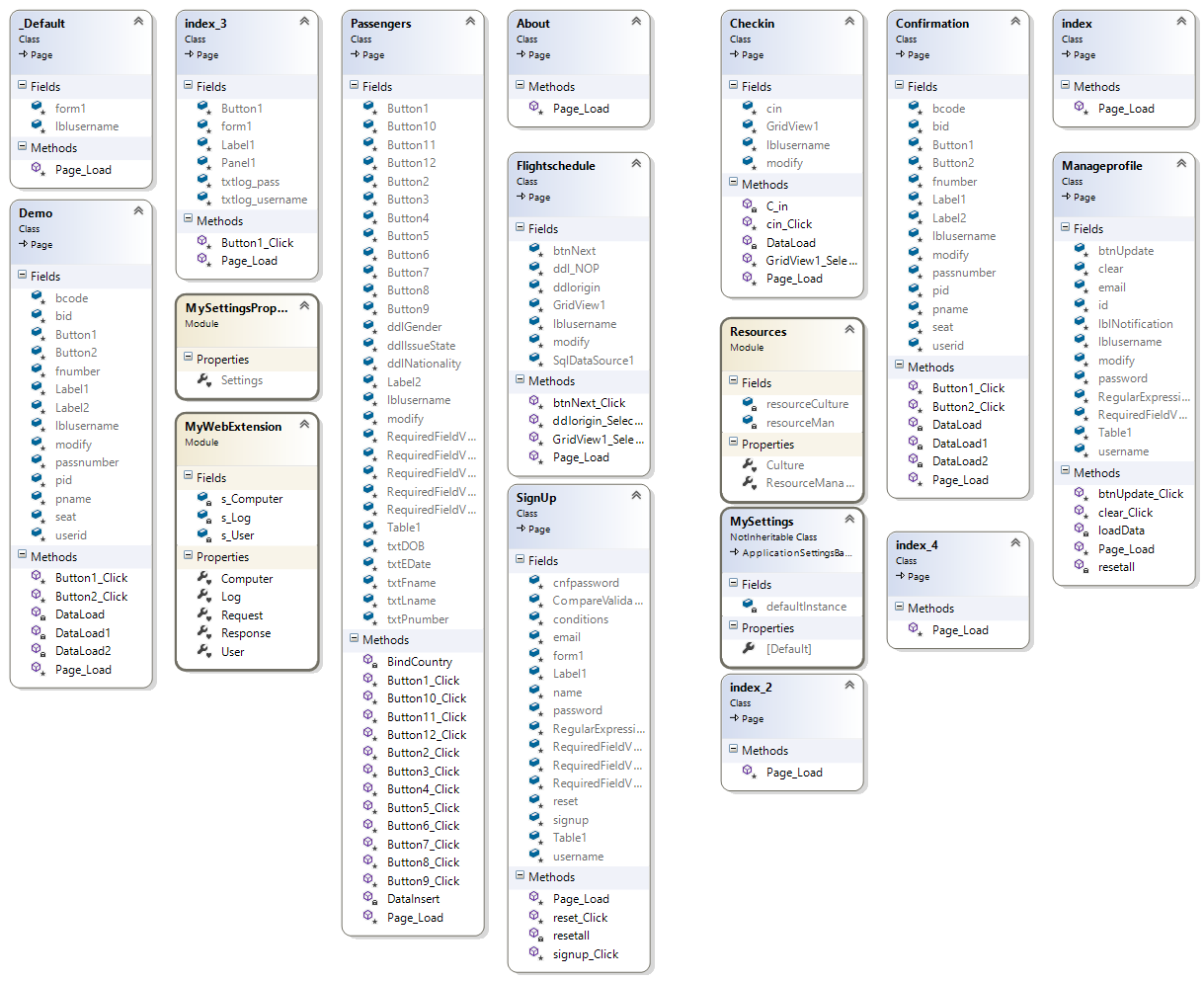


Figure 2: Class diagram

# Activity Diagram

All software systems are made up of a series of stepwise procedures whose completion in a certain order will yield desirable result. Such stepwise procedures are technically addressed as activities in computer science and the activity diagram is concerned with representing the flow between such activities in a certain component of a software system. The following activity diagrams have been developed for the proposed cloud system.

* Add



Figure 3: Add activity

* Update



Figure 4: Update activity

* Delete



Figure 5: Delete activity

* View



Figure 6: View activity

# Entity Relationship Diagram

The term Entity Relationship Diagram or simply ERD can be technically described as a graphical illustration which represents the relationship between various entities involved in a relational schema. The following ERD has been developed for the proposed system.



Figure 7: ERD

# Cloud Design Patterns

The term cloud deign pattern can be defined as a proven solution for a prevalent cloud problem. To be particular, software developers often encounter common problem when developing software systems and design patterns can be regarded as approved solutions that can address such problem in an effective and standard way. Such phenomenon also exists in the cloud computing paradigm and for that reason there are design patterns that have been recommended particularly for solving common cloud computing issues. The next sections of this chapter will justify and demonstrate the cloud design patterns applicable to the proposed project.

## Federated Identity Pattern

The Federated Identity design pattern can be described as a solution which delegates authentication to external identity providers. This design pattern can help simply the development and more importantly the maintenance of cloud application with respect to the fact the authentication task is rerouted to a robust and renowned identity provider such as Microsoft, Google, or Facebook. In other words, this design pattern allows users to login to a system using the credentials they use for logging into another account provided by a well-known company such as Microsoft or such as a Hotmail account. The following diagram better represents the mechanism of the federated identity design pattern.

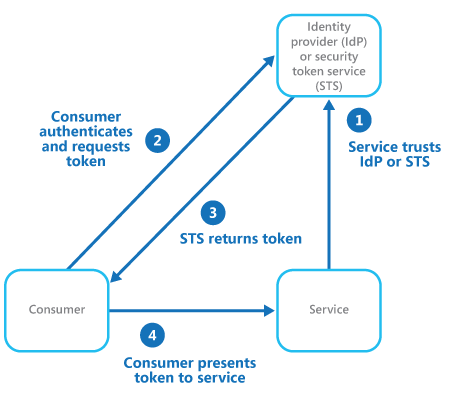


Figure 8: Federated identity pattern

## Retry Design Pattern

The Retry design pattern allows for handling transient faults in an application when it tries to connect to a network resource or a service through creating transparency between retrying and a failed operation. This can lead to the improvement of the stability and consistency in an application. The key reason behind the usefulness of this pattern and its requirement in the proposed system is that the majority of application which communicate with elements running on the cloud require to be sensitive to transient failures which take place in that particular environment. Accordingly, this pattern can help increasing the stability of the application through handling faults elegantly and transparently. The following figure better describes the overall mechanism of this pattern.

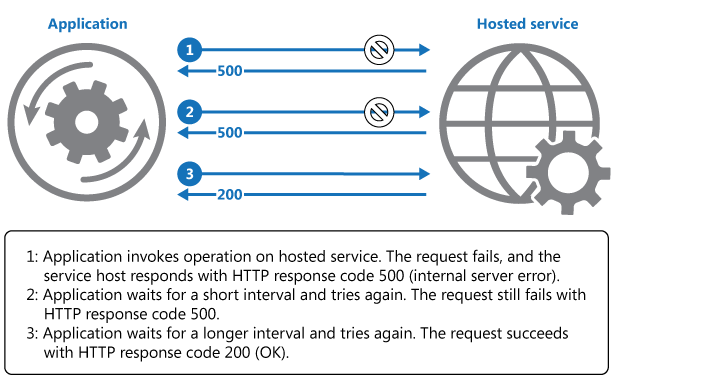


Figure 9: Retry pattern

# Implementation

This chapter of the document is concerned with describing and demonstrating the implementation of the proposed system. The next sections of this chapter will provide greater information in this respect.

## Creation of Active Directory

As previously discussed in the earlier chapters of this paper, it has been determined that the federated identity pattern is one of the design pattern from which the proposed system can highly benefit. For that reason, implementation of this design pattern has been considered in the system. In order to implement the federated identity pattern, the first step that has to be carried out is to create an Active Directory in Microsoft Azure control panel followed by adding authorized users to the directory. In order to carry out this process, the Active Directory node in the Azure navigation bar can be reached followed by clicking on the create button and configuring the directory settings. The following figure demonstrates the completion of this task during the deployment of the proposed system.

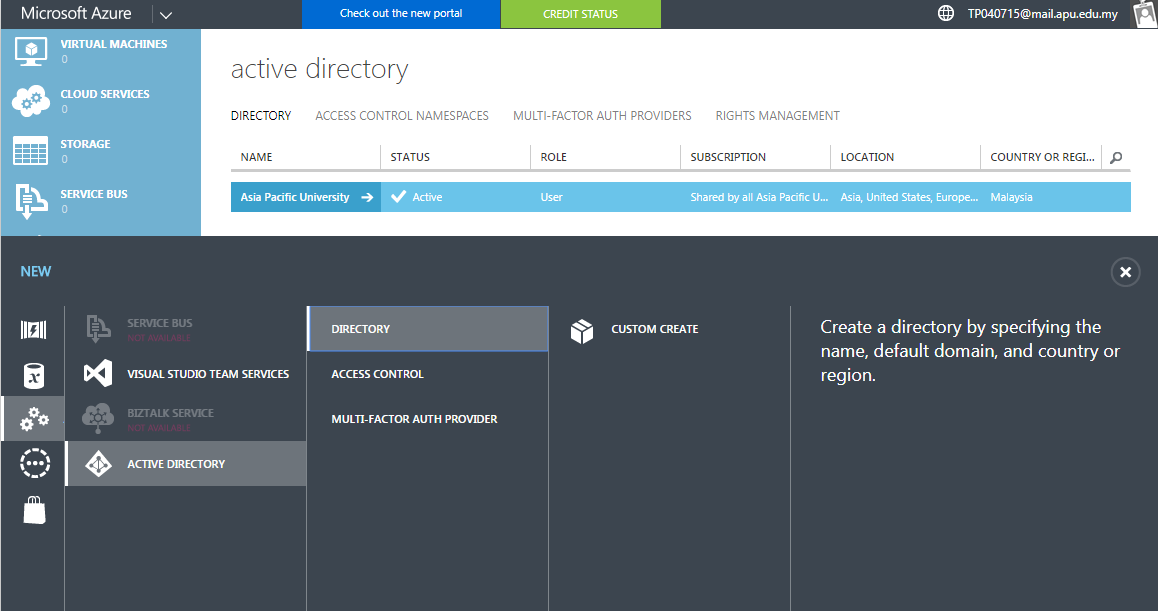


Figure 10: Creating active directory

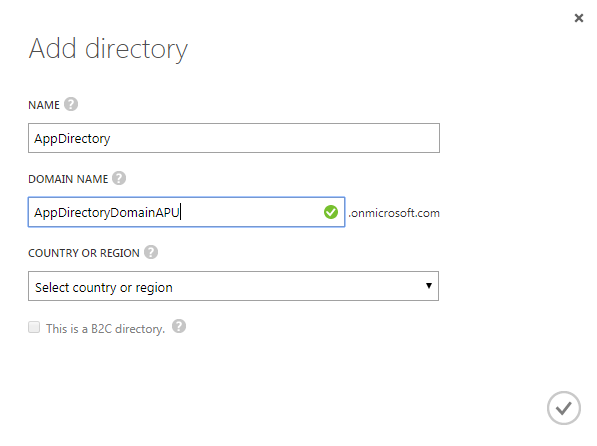


Figure 11: Providing active directory settings

## Creating a Database Server

Before proceeding with creating a database in the Microsoft Azure portal, it is necessary to create a database server first. This can be carried out by navigating to the database section in the Microsoft Azure portal followed by moving onto the server creation area. Upon providing the overall server settings such as the server name, password, and location Microsoft Azure begins creating a database server. The following figure demonstrates the completion of this process during the deployment of the system.

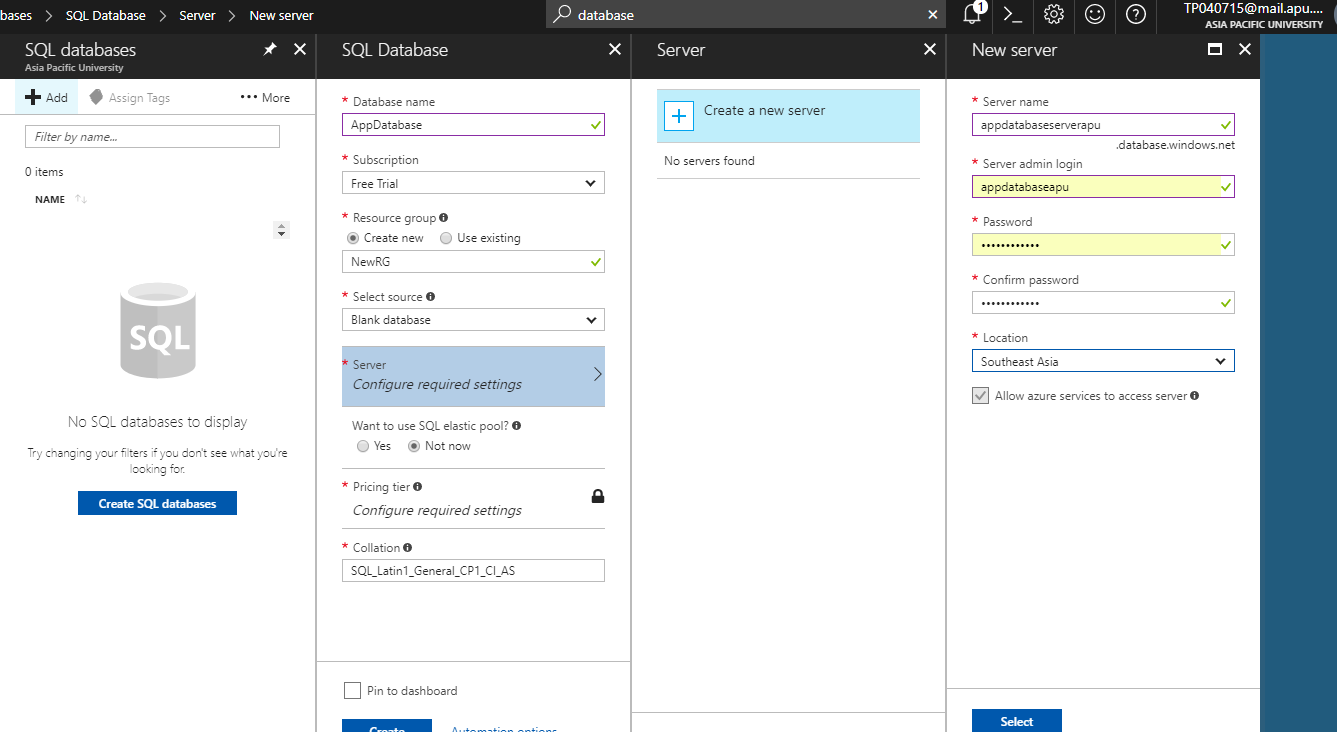


Figure 12: Creating database server

## Creating a Database

Once a database server has been created on Microsoft Azure portal, a database can be created on that particular server through navigating to the database section in the Azure menu. This is followed by providing the overall setting for the required database such as on which server it must be created, name, and size. Once the required information has been provided, Microsoft Azure begins creating the required database on the selected server. The following figure demonstrates this process.

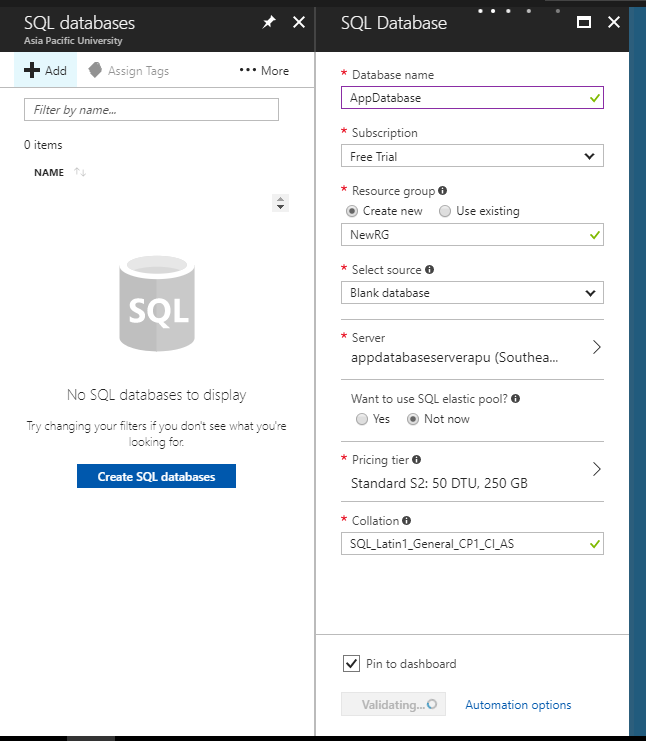


Figure 13: Creating database

## Creation of a Web Application

A web application can be created on the Azure platform through navigating to the App Service section of the Azure menu following by clicking on the create button. At this step, the basic settings for the required web application must be provided which involves resource group, name, and location. Once these settings have been set, the Azure platform proceeds with creating the required web application.

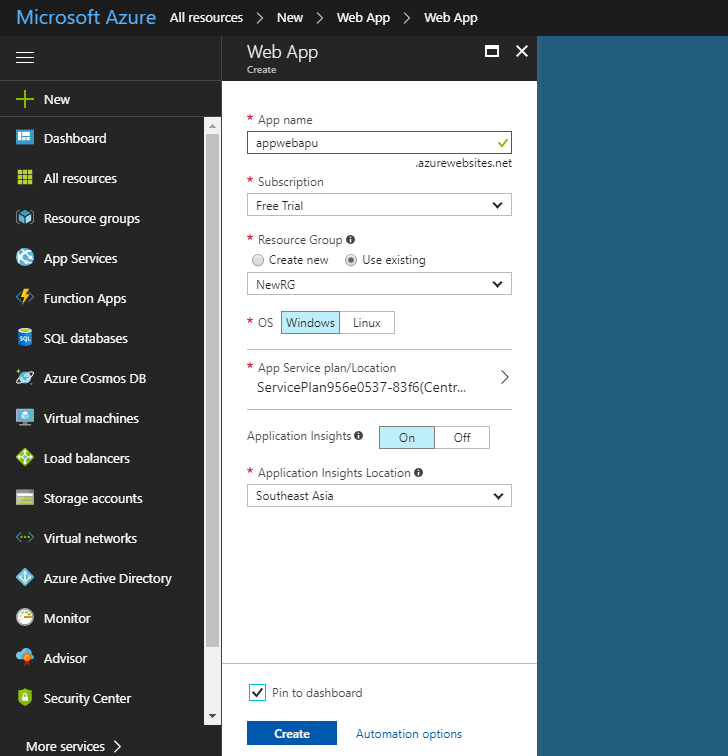


Figure 14: Creating database

## Deploying the Actual App on Microsoft Azure

In order to deploy the actual developed system in Microsoft Azure, the developed solution must be packed by the Visual Studio. In order to achieve this, the build dropdown menu in the top navigation bar must be reached followed by clicking on the publish button. The publish profile for the created web application in Microsoft Azure must be downloaded at this time and it must be provided to Visual Studio in order to deploy the solution on the cloud platform. Visual Studio will automatically read the settings in the publish profile and will deploy the application on Microsoft Azure.

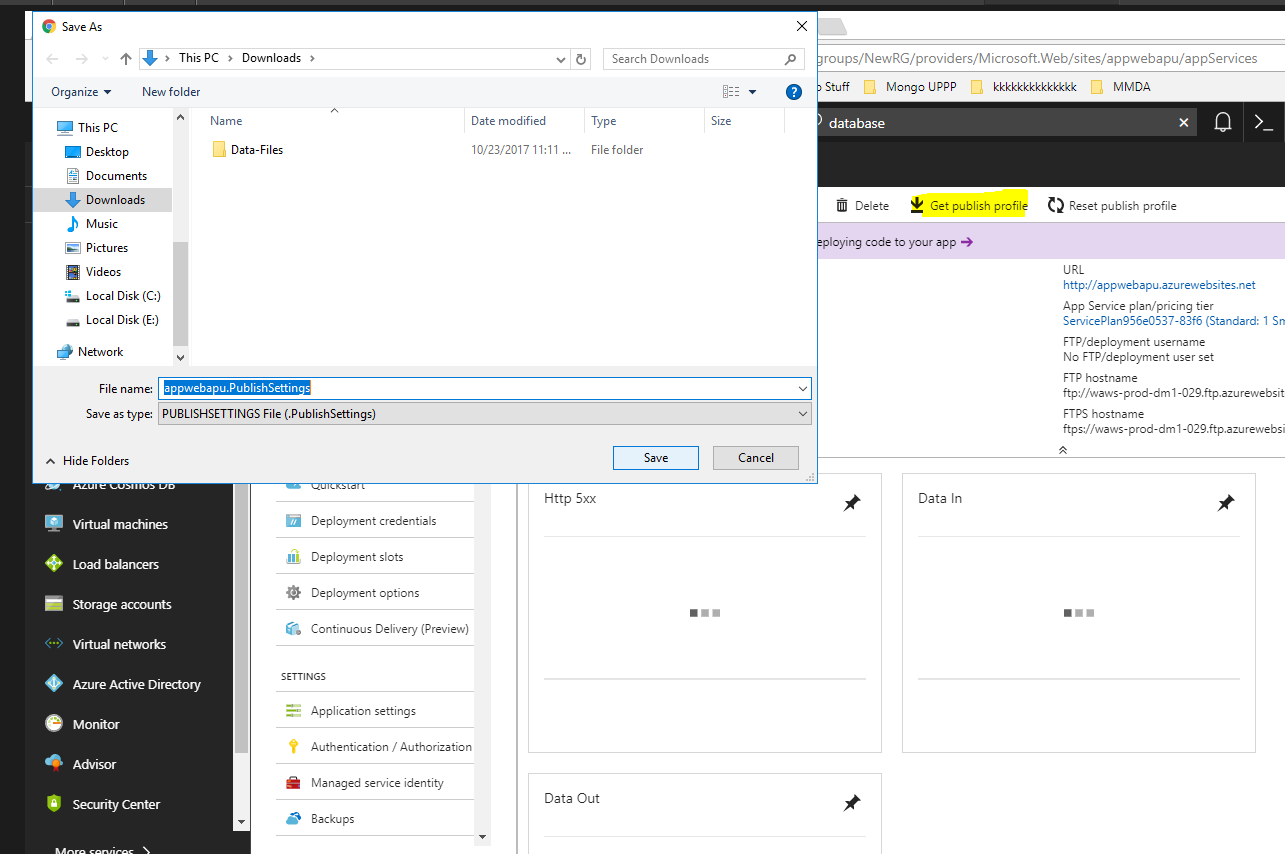


Figure 15: Deploying the system

# Screenshots

This chapter of the document is concerned with demonstrating the actual implementation of the system through providing screenshots for each implemented section. The next sections of this chapter will provide greater information in this respect.

## Home Page

This area of the application provides users with overall information regarding the website and it allows them to navigate across the website and use different functions.

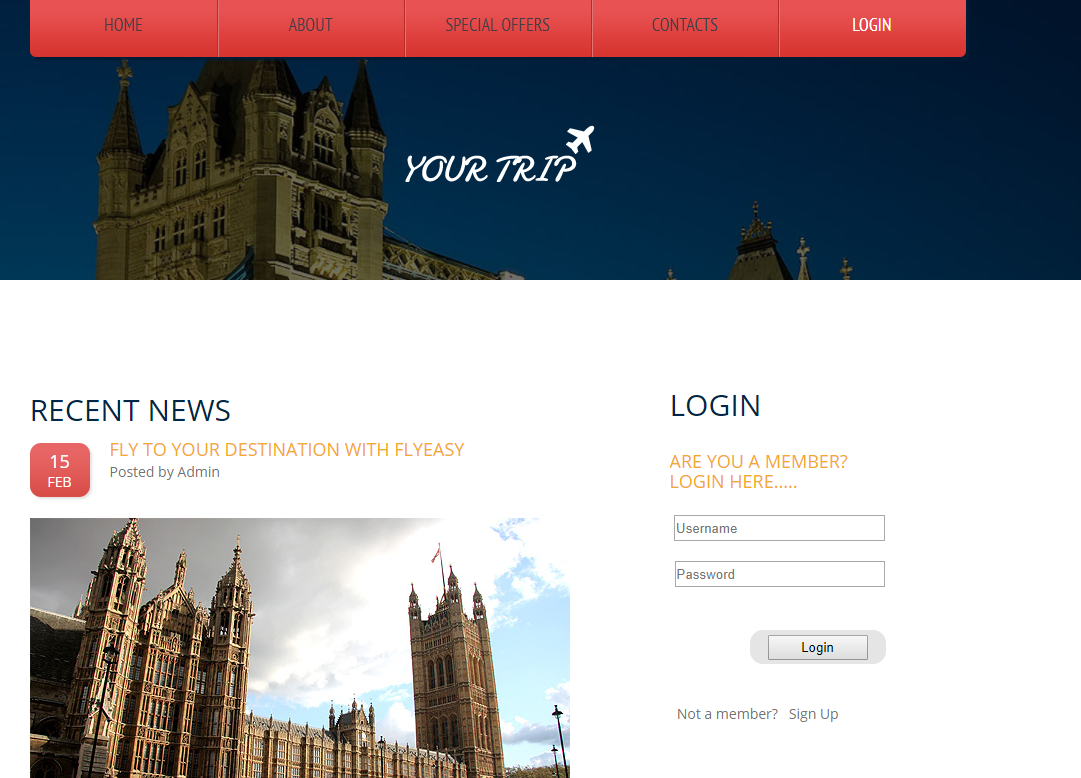


Figure 16: Home page

## Special Offers

This area of the system has been designed to display the latest special offers as decided by the company.

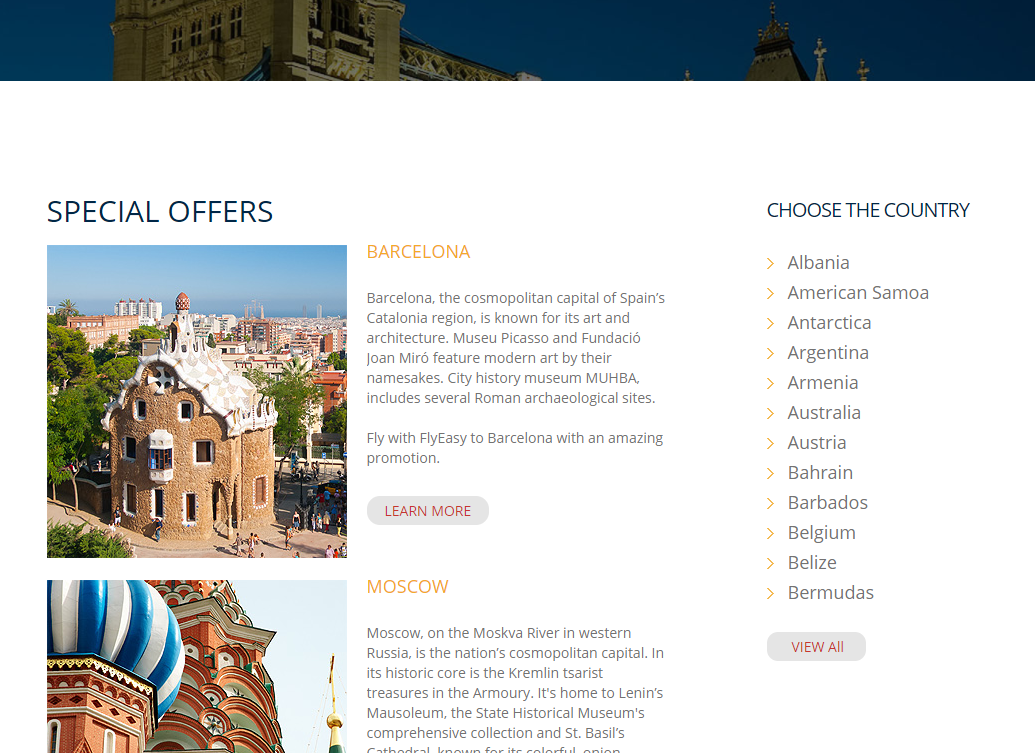


Figure 17:Special offers

## Login Panel

This area of the system allows user to login to their own control panels and the main purpose of this section is to protect the system against unauthorized access.

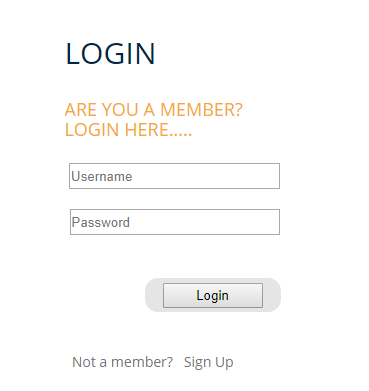


Figure 18: Login panel

## User Panel

This area allows user to utilize various functions implemented in the system such as searching for flights, make bookings, or check-in online.

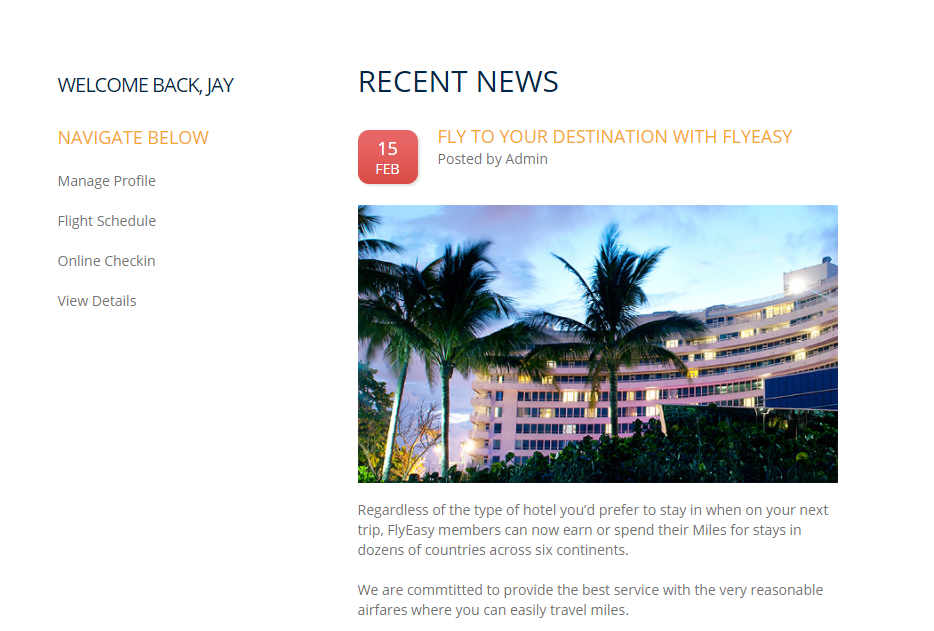


Figure 19: User panel

## Profile Modification

This area of the system allows uses to edit their personal information when needed in the system.

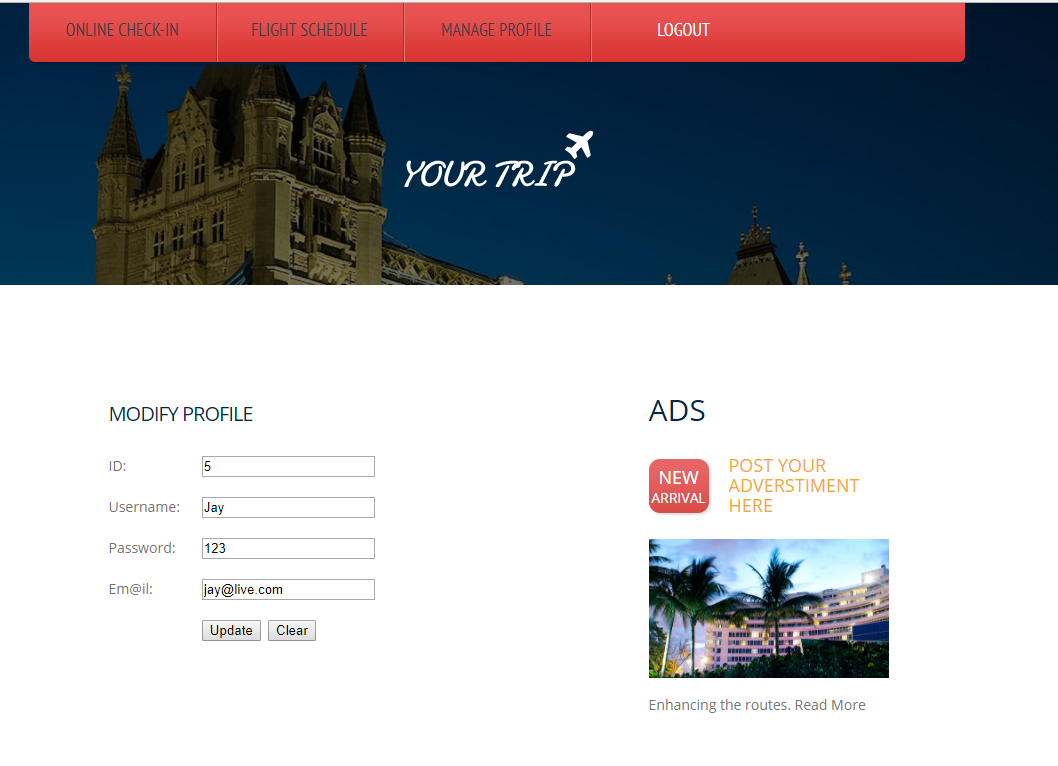


Figure 20: Profile modification

## Flight Schedule

This area of the system enables users to search flights in the system.

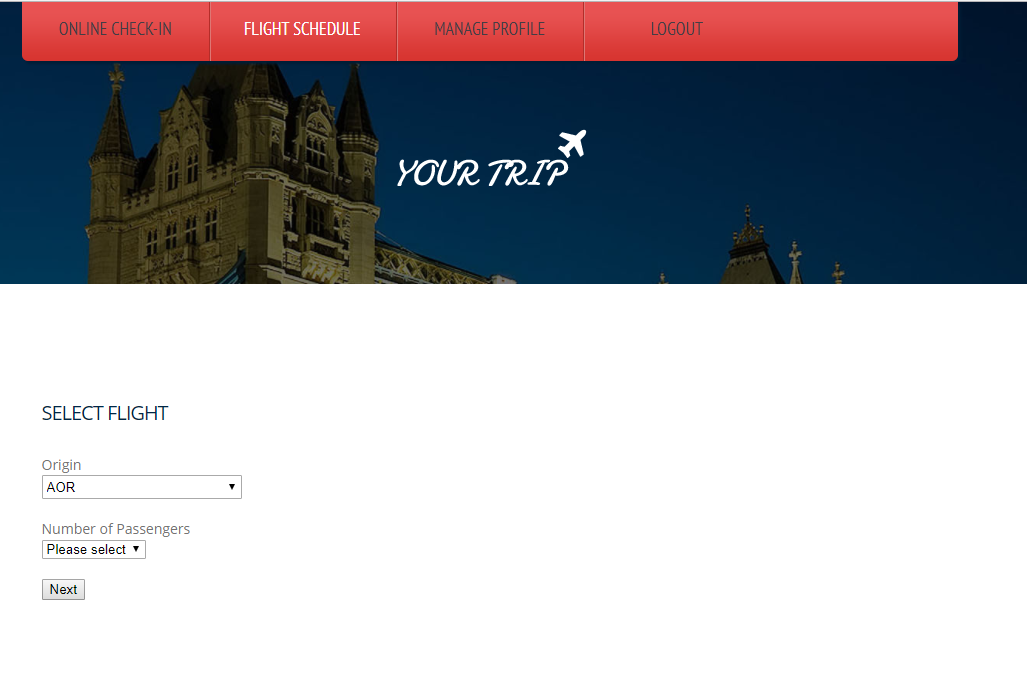


Figure 21: Flight schedule

## Booking

This section of the system allows for making reservation in the system.

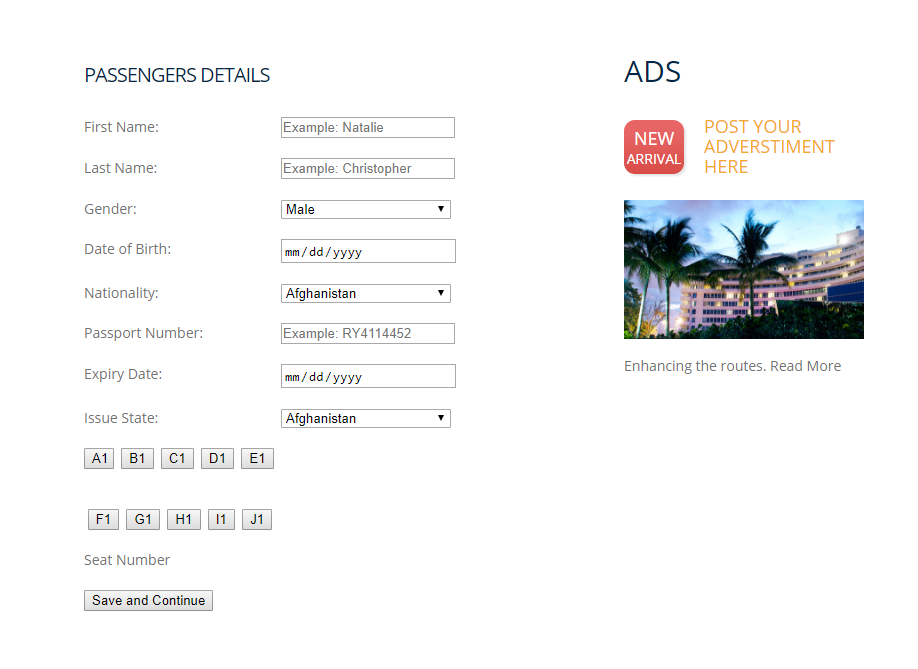


Figure 22: Booking

## Payment

Using this section of the system, users can finalize their bookings and make the payment.

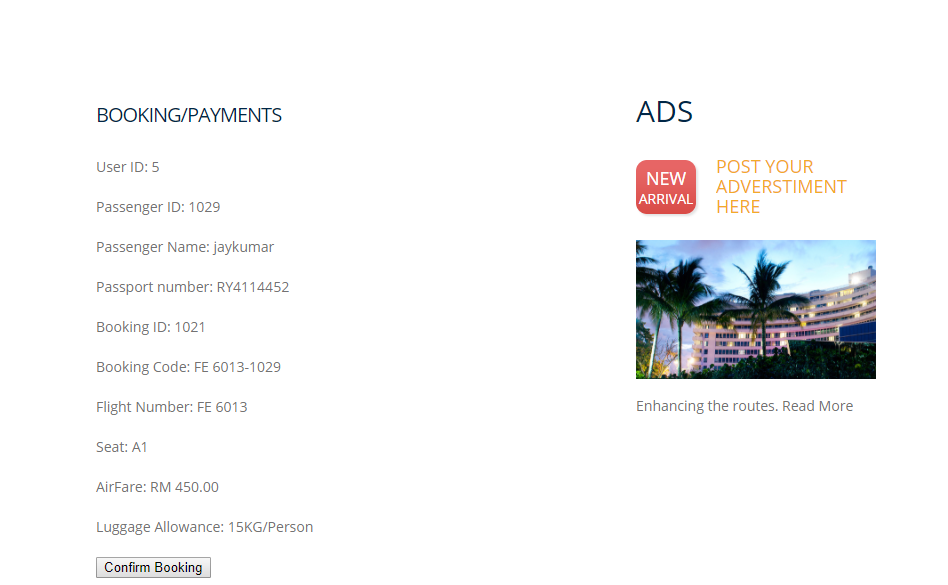


Figure 23: Payment

## Online Check-in

Users can utilize this area of the system to check-in online.

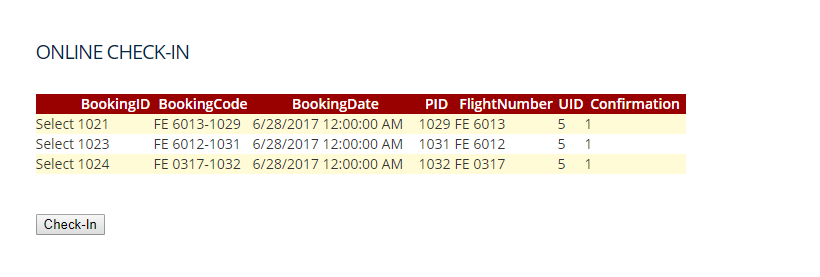


Figure 24: Check-in

# Testing

In order to fully test the developed solution, two different testing methods have been used which are respectively unit testing and performance testing. The unit testing phase has been carried out before deploying the developed system in order to make sure all implemented components are fully functional and they meet satisfactory criteria. On the other hand, the performance testing was used once the system had been deployed on Azure platform in order to test the actual performance of the system in real-world environment. The following sections of this chapter will provide greater information in this respect.

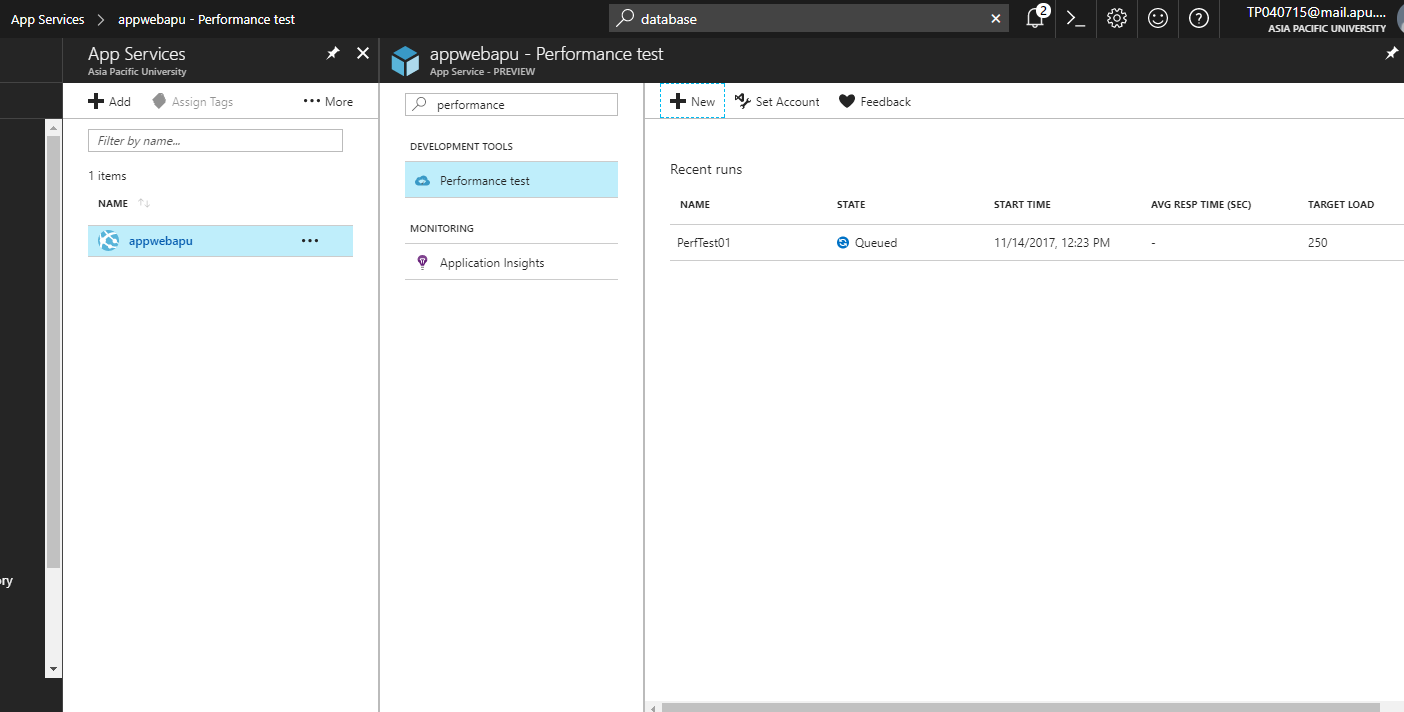
## Unit Testing

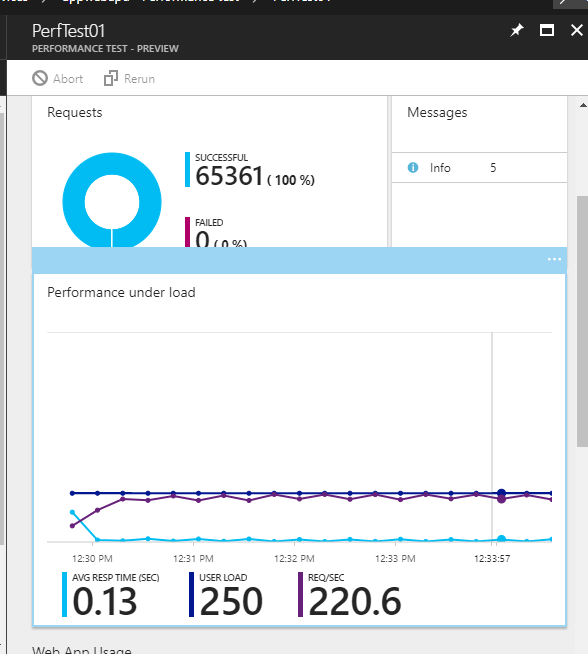
The term unit testing can be described as the practice of evaluating all implemented components in a software system against satisfactory criteria. This is often carried out by comparing the actual result and output of such components against expected results. There are many different frameworks available for this purpose which are in use by software developers. With respect to the fact that the proposed system has been developed using Microsoft visual Studio, the built-in unit testing framework in this IDE has been used for completing this phase of the project. The following table represents the latest result obtained through the unit testing process.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Module | Function | Condition | Expected Result | Actual Result | Satisfactory |
| 1 | Passenger Management | Add | All valid inputs | Add record | Record added | TRUE |
| 2 | Passenger Management | Add | Some invalid inputs | Show validation error | Error shown | TRUE |
| 3 | Passenger Management | Remove | - | Remove record | Record removed | TRUE |
| 4 | Passenger Management | Update | All valid inputs | Update record | Record updated | TRUE |
| 5 | Passenger Management | Update | Some invalid inputs | Show validation error | Error shown | TRUE |
| 6 | Passenger Management | Display | - | Display record | Record display | TRUE |
| 7 | Flight Management | Add | All valid inputs | Add record | Record added | TRUE |
| 8 | Flight Management | Add | Some invalid inputs | Show validation error | Error shown | TRUE |
| 9 | Flight Management | Remove | - | Remove record | Record removed | TRUE |
| 10 | Flight Management | Update | All valid inputs | Update record | Record updated | TRUE |
| 11 | Flight Management | Update | Some invalid inputs | Show validation error | Error shown | TRUE |
| 12 | Flight Management | Display | - | Display record | Record display | TRUE |
| 13 | Route Management | Add | All valid inputs | Add record | Record added | TRUE |
| 14 | Route Management | Add | Some invalid inputs | Show validation error | Error shown | TRUE |
| 15 | Route Management | Remove | - | Remove record | Record removed | TRUE |
| 16 | Route Management | Update | All valid inputs | Update record | Record updated | TRUE |
| 17 | Route Management | Update | Some invalid inputs | Show validation error | Error shown | TRUE |
| 18 | Route Management | Display | - | Display record | Record display | TRUE |
| 19 | Admin Management | Add | All valid inputs | Add record | Record added | TRUE |
| 20 | Admin Management | Add | Some invalid inputs | Show validation error | Error shown | TRUE |
| 21 | Admin Management | Remove | - | Remove record | Record removed | TRUE |
| 22 | Admin Management | Update | All valid inputs | Update record | Record updated | TRUE |
| 23 | Admin Management | Update | Some invalid inputs | Show validation error | Error shown | TRUE |
| 24 | Admin Management | Display | - | Display record | Record display | TRUE |
| 25 | Booking Management | Add | All valid inputs | Add record | Record added | TRUE |
| 26 | Booking Management | Add | Some invalid inputs | Show validation error | Error shown | TRUE |
| 27 | Booking Management | Remove | - | Remove record | Record removed | TRUE |
| 28 | Booking Management | Update | All valid inputs | Update record | Record updated | TRUE |
| 29 | Booking Management | Update | Some invalid inputs | Show validation error | Error shown | TRUE |
| 30 | Booking Management | Display | - | Display record | Record display | TRUE |
| 31 | Login | - | Login credentials | Log user in | User logged in | TRUE |
| 32 | Login | - | Invalid credentials | Show Error | Error shown | TRUE |
| 33 | Logout | - | - | Log user out | User logged out | TRUE |

## Performance Testing

In order to complete this phase of the project, the built-in performance testing tool in Microsoft Azure has been used. The following figures demonstrate the completion of this process in the project.





## Conclusion

In order to implement the proposed system, various aspects of programming and cloud computing have been researched and analyzed of which web development, cloud design pattern, and Microsoft Azure platform can be regarded as the most important. Throughout the course of the project, various problems have been faced which were determined to be solvable through the use of particular cloud design pattern. The implemented solution has been contentiously tested through unit testing and once a done-working version of the system was achieved, the project entered the actual deployment phase in which the implemented solution was physically deployed on the cloud platform. Once the application was deployed, the performance of the deployed system was tested in the cloud platform which was determined to be reasonable and suitable for commercial environment.

# References

Bond, J., 2015. *The Enterprise Cloud.* s.l.:O'Reilly Media.

Garber et al., D., 2013. *Windows Azure Hybrid Cloud.* s.l.:Wrox.

Mather et al., T., 2009. *Cloud Security and Privacy.* s.l.:O'Reilly Media.

Reese, G., 2009. *Cloud Application Architectures.* s.l.:O'Reilly Media.

Spaanjaars, I., 2010. *Beginning ASP.NET 4 in C# and VB.* s.l.:Wrox.

Spaanjaars, I., 2014. *Beginning ASP.NET 4.5.1: in C# and VB.* s.l.:Wrox.

Wilder, B., 2012. *Cloud Architecture Patterns.* s.l.:O'Reilly Media.