

|  |  |  |
| --- | --- | --- |
| **Module Code** | **:** | CT071-3-3-DDAC |
| **Module Title** | **:** | Designing and Developing Applications on the Cloud |
| **Intake Code** | **:** | UC3F1701SE |
| **Lecturer Name** | **:** | DR. KALAI ANAND A/L RATNAM |
| **Hand in Date** | **:** | 2nd October 2017 |
| **Student Name** | **:** | Thum Choon Tat |
| **Student ID** | **:** | TP030470 |

Table of Contents

[Introduction 3](#_Toc494385455)

[Project plan 4](#_Toc494385456)

[Design 5](#_Toc494385457)

[Cloud pattern design 5](#_Toc494385458)

[Architectural diagrams 6](#_Toc494385459)

[Estimated pricing 7](#_Toc494385460)

[Design considerations 7](#_Toc494385461)

[Modelling 8](#_Toc494385462)

[Use case 8](#_Toc494385463)

[Sequence 8](#_Toc494385464)

[Page flow 11](#_Toc494385465)

[ERD 11](#_Toc494385466)

[Implementation 12](#_Toc494385467)

[File system architecture 12](#_Toc494385468)

[System hosting on cloud 13](#_Toc494385469)

[Managed database 14](#_Toc494385470)

[Test plan 16](#_Toc494385471)

[Unit test 16](#_Toc494385472)

[Performance test 26](#_Toc494385473)

[Summary 26](#_Toc494385474)

[Charts 26](#_Toc494385475)

[Analysis 27](#_Toc494385476)

[Conclusion 28](#_Toc494385477)

[References 29](#_Toc494385478)

[Appendix 31](#_Toc494385479)

# Introduction

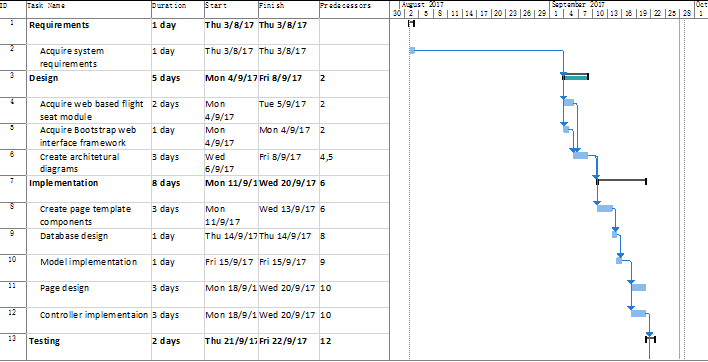
Online shoppers are notoriously fickle. If a website lags for even a few seconds, shoppers are just a couple of clicks away from many more options. Ukraine International Airlines (UIA) is the flagship carrier and largest airline in Ukraine. It operates domestic and international passenger flights and 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 analysing 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 datacentres into a public cloud could solve all these problems. Therefore, an Online Flight Booking System will be developed and hosted into Microsoft Azure cloud service, which allow users to create user accounts and book tickets.

# Project plan



# Design

## Cloud pattern design

Cloud design patterns are useful in developing a reliable, scalable and secure applications in the cloud (Christopher Bennage, 2017). Index Table pattern has been chosen to be implemented into this system. Index Table pattern is implemented by creating indexes on columns which are frequently being queried to improve the performance on getting indexed data (Christopher Bennage, 2017).

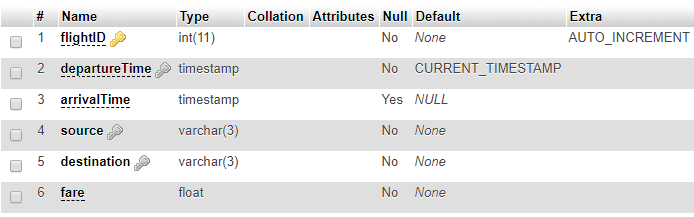


Figure 1: Flight table

The image above shows the table structure for flight table, which has departure time column indexed. The reason of indexing departure time column is to improve the performance of user searching flight based on departure time. Although it is possible to search a flight by flight ID, but the user prefers to search flight by departure time because flight ID in this table are auto incremented. Plus, auto incremented flight ID is meaningless and only used to identify unique flights. Without creating an index on departure time, the query performance will drop because the user can only search flights by departure time. Therefore, the decision of indexing departure time column is made.

## Architectural diagrams

|  |
| --- |
|  |

Figure 2: Cloud architecture

### Estimated pricing

|  |  |  |  |
| --- | --- | --- | --- |
| **Microsoft Azure Estimate** | | |  |
| **Service type** | **Region** | **Description** | **Estimated Cost** |
| App Service | West Europe | 1 instance(s) x 1 Months,  Size: S1, Standard tier | RM331.08 |
| App Service | UK South | 1 instance(s) x 1 Months,  Size: S1, Standard tier | RM413.85 |
| Traffic Manager | West Europe | 1 million DNS queries/month,  1 Azure endpoint(s) | RM4.005 |
| Traffic Manager | West Europe | 1 million DNS queries/month,  2 Azure endpoint(s) | RM5.607 |
| ClearDB MySQL database | West Europe | Pricing Tier: Venus  Backup ready | RM42.00 |
| Support |  | **Support** | RM0.00 |
|  |  | **Monthly Total** | **RM796.542** |
|  |  | **Annual Total** | **RM9,558.504** |

The estimated pricing for each service type listed above, except ClearDB MySQL database, are retrieved from Microsoft Azure Pricing Calculator.

## Design considerations

As the main users of Online Flight Booking System are mostly from Ukraine, the web application service as well as database will require to be hosted in Azure datacentre near Ukraine to provide high availability system for them. A secondary web application service will be hosted in the second nearest datacentre for other users to access the web application and will uses the database service hosted in nearest datacentre. With limited credit of RM150 provided, the number of web application hosting days will be reduced and all performance tests are required to be completed in limited time as the monthly cost for hosting the web applications are around RM796 as described above.

## Modelling

### Use case



Figure 3: Use case diagram

### Sequence

|  |
| --- |
|  |

Figure 4: Login

|  |
| --- |
|  |

Figure 5: Register

|  |
| --- |
|  |

Figure 6: View booking

|  |
| --- |
|  |

Figure 7: Make booking

### Page flow

|  |
| --- |
|  |

Figure 8: Page flow

### ERD

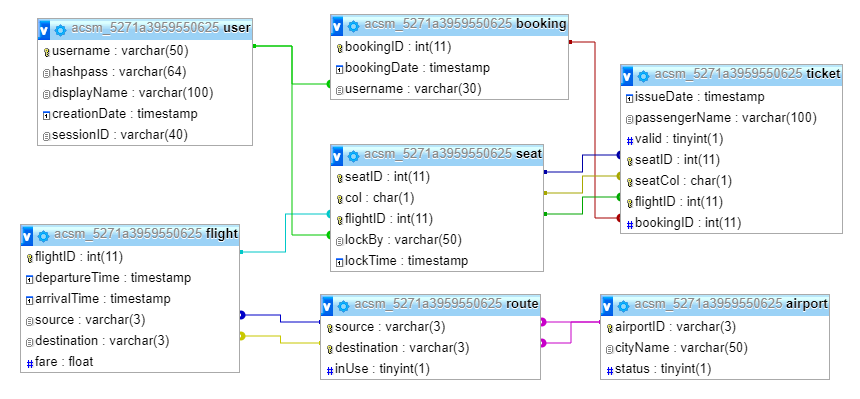


Figure 9: ERD

# Implementation

## File system architecture

The UIA flight booking system is developed with PHP with MVC architecture implemented. The directories for the system are as follows:



|  |  |
| --- | --- |
| **Directory** | **Description** |
| Root | System root directory |
| Model | Contains model components |
| Class | Store database connector, message and user classes |
| Function | Stores flight, seat and ticket functions |
| Pages | Contains view and controller components |
| Resource | Contains web page resources, such as CSS, JS and images |
| Template | Stores common view for pages across the system |

Every request received by this system will be redirected to index page and load page contents from pages directory based on user privilege and requested page. The request redirect rule is written in configuration file named “.htaccess”. Header and footer will be included in every page fetched from Template directory. The purpose of redirecting all requests to index page is to restrict user from accessing and modifying system files such as configuration files.

Each directory in Pages directory are treated as a page with 3 files, which are core file, resources file and content file. Core file act as the controller for the page, resources file act as the container of additional web file resources such as cascading stylesheets and JavaScript, and content file act as the view component of the page. The index page will load these files based on the URL entered by the user. For example, if the user enters “/login” at the end of the domain URL, the index script will fetch the 3 files from “/pages/login” directory. Any other files included in pages directory will not affect the system behaviour and will be ignored by the system. If the content file does not exist in the page, a temporary page content will be fetched from Template directory stating that the page is empty. The purpose of separating controller file from content file is to enable the controller to perform page redirect action as page redirect action is not available after writing response contents.

## System hosting on cloud

After the system is developed and tested on local server in local machine, the system is then deployed into cloud in West Europe region and UK South region. The reason of deploying the application in these regions is to conduct user load test by simulating traffics coming from regions near Ukraine. Although Germany region is closer to Ukraine compared with West Europe, the option to deploy the web application on Germany data centre is not available.

Web application hosted on both regions are deployed using single repository in GitHub source control management service, therefore changes of web application will updates both hosted web application at once. Both web application uses app service with standard pricing tier (S1 Standard) applied. The reason of choosing standard pricing tier is to allow the web application to be configured in traffic manager.



Figure 10: Traffic manager

The traffic manager is configured to use geographical routing method and added endpoints to both web application deployed in West Europe and UK South region. Additional traffic manager profile with performance routing method is created to be included as a nested profile for the geographical routed traffic manager as shown above. As the main users of this system are from Ukraine, the endpoint in West Europe is only exposed to users from Ukraine and will have experience on high performance web application.

## Managed database

Platform as a service (PaaS) is a category in cloud computing which provides platform and environment for developers to build applications and services (Interoute Communications, 2017). Therefore, the developers will not require to install additional hardware and software to develop and run applications (Rouse, 2015) as they are provided by cloud service provider or partners of cloud service provider.

In the case of this application, MySQL database provided by ClearDB is selected to store flights and bookings data hosted in West Europe. The reason of hosting database in West Europe is to reduce latency of users from Ukraine accessing booking and flights data as it is located near Ukraine. Plus, the pricing for the selected database is lower than MySQL database provided by Microsoft Azure.

MySQL database provided by ClearDB guarantees service availability of 99.5% Service Level Agreement for single instance database (ClearDB, 2017), which is current being used in this system. ClearDB also provides scheduled backups in multiple regions in multiple time zones and have their teams constantly monitoring databases to fulfil their SLA (ClearDB, 2017).

By hosting large databases from the cloud, the cost of managing it will be reduced as scaling databases is cheaper on the cloud than purchasing and configure additional servers manually (Howell, 2013). Also, backup services are automated and available for disaster recovery and remote access to the database (Network Specialists, 2014). Although hosting databases on cloud provide significant advantages, but the user will not have full control on the servers hosting the databases (Network Specialists, 2014). Plus, there will be hesitations on storing sensitive data on the cloud such as financial data or health reports (Sears, 2014). Taking advantage of storing sensitive data at on site servers and other data on the cloud could be a solution to the security issue on sensitive data, but the user will be responsible for disaster recovery for data stored in on site servers.

# Test plan

## Unit test

| **ID** | **Title** | **Description** | **Result** | |
| --- | --- | --- | --- | --- |
| **Expected** | **Actual** |
| **General site test suite** | | | | |
| GS1 | 403 forbidden | Notify user when accessing valid pages without enough privileges  Procedure:   1. Enter “/booking” at the end of domain URL while not logged in | Redirect to error 403 page | 403 error page is shown |
| GS2 | 404 not found | Notify user when accessing invalid pages  Procedure:   1. Enter “/shop” at the end of domain URL | Redirect to error 404 page | 404 error page is shown |
| GS3 | Error page direct access | Redirect user to home page when trying to access error page manually  Procedure:   1. Enter “/error” at the end of domain URL | Redirect to home page | Home page is shown |
| GS4 | New session | Stores session ID into database when logged in  Procedure:   1. Login with valid user credential | Store session ID into database | Session ID is stored |
| GS5.1 | Destroy session | User session destroys when user logs out  Procedure:   1. While logged in, click on “Logout” button at the right side of navigation bar | Redirect user to home page with sign in and register option enabled | User is redirected to home page with login and register option enabled |
| GS5.2 | User session destroys when user closes web browser  Procedure:   1. Closes web browser while logged in 2. Open web browser with domain URL entered | Show home page with sign in and register option | Home page with sign in and register option is shown |
| GS5.3 | User session destroys when same account is being used in another web browser  Procedure:   1. Sign in with valid user credential in web browser A 2. Sign in with same user credential in web browser B 3. Refresh ticket list page in web browser A | Web browser A shows home page with sign in and register option enabled | Web browser A redirected to public home page |
| **User privilege test suite** | | | | |
| UP1 | User navigation links | Loads user navigation links when logged in  Procedure:   1. Sign in with valid user credential | Show booking navigation link | Booking navigation link is shown |
| **Navigation bar test suite** | | | | |
| NB1.1 | Website title | Redirect user to home page when clicked on “UIA” while not signed in | Redirect user to public home page | Redirect user to public home page |
| NB1.2 | Redirect user to home page when clicked on “UIA” while signed in | Redirect user to ticket list page | Redirect user to ticket list page |
| NB2.1 | Home page | Redirect user to home page when clicked on “Home” while not signed in | Redirect user to public home page | Redirect user to public home page |
| NB2.2 | Redirect user to home page when clicked on “Home” while signed in | Redirect user to ticket list page | Redirect user to ticket list page |
| NB3 | Booking page | Redirect user to boking page when clicked on “Booking” while signed in | Redirect user to booking page | Redirect user to booking page |
| **Home page test suite** | | | | |
| HP1.1 | Page content | Shows home page while not logged in | Show home page with website title | Home page with website title is shown |
| HP1.2 | Shows home page while logged in | Show ticket list page | Ticket list page is shown |
| HP2.1 | Search tickets | Shows ticket list with valid booking date range  Procedure:   1. Enter valid booking date range 2. Click on “Search Tickets” button | Show tickets associate with booking date range | Tickets are shown |
| HP2.2 | Shows ticket list with invalid booking date range  Procedure:   1. Enter valid booking date range 2. Click on “Search Tickets” button | Does not show ticket list | Tickets are not shown |
| **Login page test suite** | | | | |
| LP1.1 | Login | Shows alert on invalid user credential used to sign in  Procedure:   1. Enter invalid username and password 2. Click on “Login” | Show alert with “Invalid username / password” message | Alert shown |
| LP1.2 | Redirect user to home page upon successful login  Procedure:   1. Enter valid username and password 2. Click on “Login” | Redirect user to ticket list page | Ticket list page shown |
| LP2 | Register link | Redirect user to registration page  Procedure:   1. Click on “here” link at the bottom of “Login” button | Redirect user to register page | Registration page is shown |
| **Registration page test suite** | | | | |
| RP1.1 | Input field hints | Show hint on focus of “Display Name” text field  Procedure:   1. Click on “Display Name” text field | Show hint | Hint shown |
| RP1.2 | Show hint on focus of “Username” text field  Procedure:   1. Click on “Username” text field | Show hint | Hint shown |
| RP1.3 | Show hint on focus of “Password” text field  Procedure:   1. Click on “Password” text field | Show hint | Hint shown |
| RP1.4 | Show hint on focus of “Reenter Password” text field  Procedure:   1. Click on “Reenter Password” text field | Show hint | Hint shown |
| RP2 | Clear form | Clear registration form when clicked on “Clear” button  Procedure:   1. Enter user registration info 2. Click on “Clear” button | Clear registration form | Form is cleared |
| RP3.1 | Registration | Register user account with invalid input field  Procedure:   1. Enter invalid user account info 2. Click on “Register” button | Error message shown on top of every text field with invalid information entered | Error messages shown |
| RP3.2 | Register user account with valid input field  Procedure:   1. Enter valid user account info 2. Click on “Register” button | Show alert with “Registration successful” message | Alert shown |
| **Booking page test suite** | | | | |
| BP1 | Search flight | Search flight based on entered departure date  Procedure:   1. Enter departure date 2. Click on “Search” button | Show flight that matches the given departure date | Flights shown |
| BP2 | Select flight | Redirect user to seat selection page upon selecting a flight  Procedure:   1. Select a flight from flight list | Redirect user to seat selection page | Redirected to seat selection page |
| **Select Seat page test suite** | | | | |
| SSP1 | Flight info | Display selected flight info | Show flight info | Flight info shown |
| SSP2.1.1 | Seat selection | Select available seats  Procedure:   1. Click on available seats | Highlight selected seats | Seats are highlighted |
| SSP2.1.2 | Enable “Book Selected Seat” button | Button enabled |
| SSP2.2.1 | Deselect selected seats  Procedure:   1. Click on selected seats | Remove highlight on selected seats | Highlights removed |
| SSP2.2.2 | Disable “Book Selected Seat” button if no seats are selected | Button disabled |
| SSP2.3 | Select booked seats  Procedure:   1. Click on booked seats | Seat is not selected | Seat not selected |
| SSP2.4 | Select locked seats  Procedure:   1. Click on locked seats | Seat is not selected | Seat not selected |
| SSP3 | Reselect flight | Redirect user to flight selection page  Procedure:   1. Click on “Back to flight selection” button | Redirect user to flight selection page | Flight selection page shown |
| SSP4.1 | Confirm seats | Book seats without selecting any seats  Procedure:   1. Click on “Book Selected Seat” button without selecting any seat | User is not redirected to passenger info page | Nothing happens |
| SSP4.2 | Book seats with at seats selected  Procedure:   1. Click on “Book Selected Seat” button with at least 1 seat selected | User is redirected to passenger info page | Passenger info page shown |
| **Passenger Details page test suite** | | | | |
| PDP1 | Flight info | Display selected flight info | Show flight info | Flight info shown |
| PDP2.1 | Name fields | Display name fields for each selected seat | Show name fields | Name fields shown |
| PDP2.2 | Show hints for each text fields  Procedure:   1. Click on each text field | Show hints | Hints shown |
| PDP3.1 | Book ticket | Book ticket without name field entered  Procedure:   1. Click on “Book” button with at least 1 empty field | No action is taken | Nothing happens |
| PDP3.2 | Book ticket with invalid name entered  Procedure:   1. Click on “Book” button with at least 1 field with invalid name entered | Show alert on invalid passenger name entered | Alert shown |
| PDP3.3 | Book ticket with valid names entered  Procedure:   1. Click on “Book” button with all valid names entered | Redirect user to ticket list page with ticket list updated | Booked ticket list shown |
| PDP4 | Reselect seat | Redirect user to reselect seat page on click on “Reselect Seats” button | Redirect user to seat selection page | Seat selection page shown |
| PDP5 | Cancel booking | Cancels booking and redirect user to ticket list page | Redirect user to ticket list page | Ticket list page is shown |

## Performance test

The performance tests conducted in duration of 5 minutes with traffic load source generated from West Europe to root traffic manager on 26th September 2017. The load tests are tested on slack hours and peak hours based on Ukraine time.

### Summary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test | | | | | | Performance (average in sec) | |
| ID | Ukraine time | | Local time | | User load | Response time | Request |
| Start | End | Start | End |
| 16 | 07:49 | 07:54 | 12:49 | 12:54 | 250 | 0.3 | 827.37 |
| 20 | 13:26 | 13:32 | 18:26 | 18:32 | 0.27 | 825.2 |
| 17 | 08:02 | 08:07 | 13:02 | 13:07 | 500 | 0.97 | 868.84 |
| 21 | 13:48 | 13:53 | 18:48 | 18:53 | 0.87 | 852.65 |
| 18 | 08:39 | 08:45 | 13:39 | 13:45 | 750 | 1.32 | 858.56 |
| 22 | 14:07 | 14:12 | 19:07 | 19:12 | 1.42 | 841.56 |

### Charts

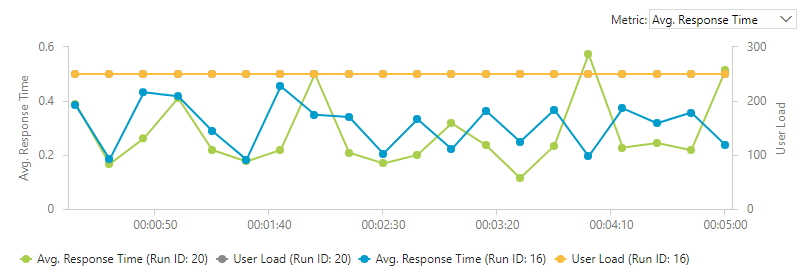


Figure 11: Average response time with user load of 250

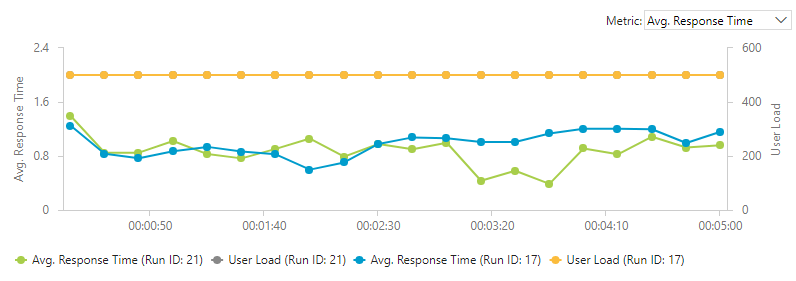


Figure 12: Average response time with user load of 500

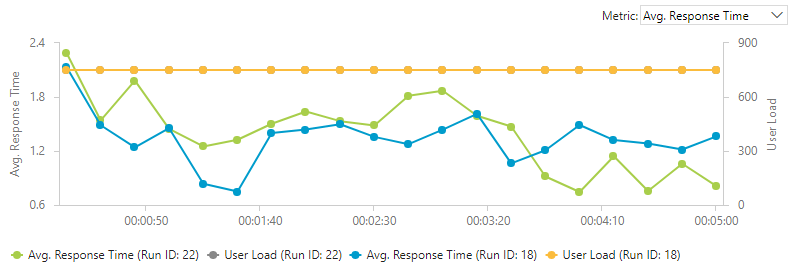


Figure 13: Average response time with user load of 750

## Analysis

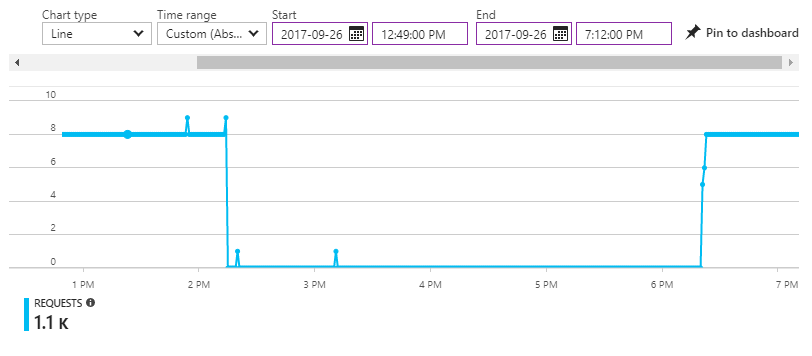


Figure 14: Requests during test (West Europe)

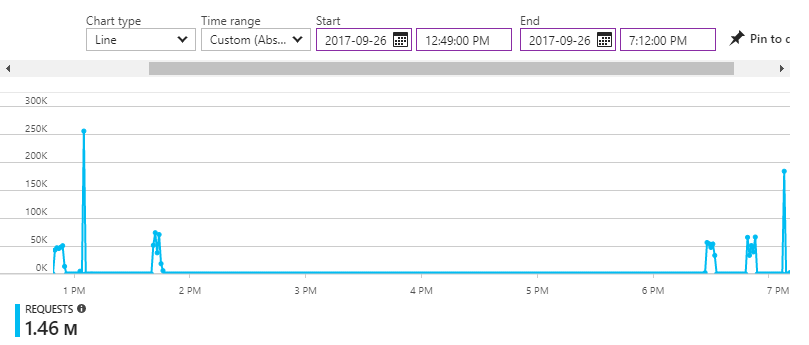


Figure 15: Requests during test (UK South)

Figure 14 and Figure 15 shows the number of requests received during performance tests conducted. Although both test result in Figure 14 and Figure 15 shows that the load test for 750 concurrent users end at 1:07PM on local time based on summary of the load test, the requests are still coming in from Ukraine with the assumptions of:

* Virtual users are generated in multiple loops or cycles
* Virtual users generated begins to keep send requests once it is generated until the process has been stopped
* The data capturing of the load test does not start until all virtual users requested by the user are generated
* Once the testing duration has reached the target, generated virtual users service are stopped in phases, but the requests from uninterrupted virtual users continue to send request

The requests received at around 3:10PM as shown in Figure 14 is not part of the load test, but the requests for updating “readme.md” markdown file from GitHub. The number of request received from UK South is more than West Europe is due to the settings of traffic manager as explained above.

Based on the load test conducted, the difference of average response time is small with 250 concurrent virtual users tested shows that the web application can serve 250 or lesser concurrent users with similar response time. Meanwhile, the huge difference between peak hours and slack hours begin from 500 and 750 concurrent users accessing the web application shows that the application is starting to slow down and might be unable to handle successful requests, although the test result does not suggest it.

# Conclusion

Hosting applications on the cloud has given significant advantages, such as performance monitoring, to many organizations, therefore more organizations are moving their solutions to the cloud as more technologies and features are to be available on the cloud. Although some organization hesitates to move sensitive information to the cloud, most of them still chooses to move their solution to the cloud as it may had solved many issues such as maintenance cost.

# References

Apache Friends. (2017). *About the XAMPP project*. Retrieved from Apache Friends: https://www.apachefriends.org/about.html

Christopher Bennage, M. W. (2017, 9 8). *Cloud Design Patterns | Microsoft Docs*. Retrieved from Microsoft Azure | Documentation: https://docs.microsoft.com/en-us/azure/architecture/patterns/

Čihař, M. (2015). *phpMyAdmin*. Retrieved from phpMyAdmin: https://www.phpmyadmin.net/

ClearDB. (2017). *ClearDB on Microsoft Azure – ClearDB*. Retrieved from ClearDB: http://w2.cleardb.net/azure/

ClearDB. (2017). *ClearDB Service Level Agreement – ClearDB*. Retrieved from ClearDB: http://w2.cleardb.net/sla/

ClearDB. (2017). *Why ClearDB? – ClearDB*. Retrieved from ClearDB: http://w2.cleardb.net/why-cleardb/

GitHub. (2017). *The world's leading software development platform*. Retrieved from GitHub: https://github.com/home

Howell, D. (2013, February 25). *On-site servers or the cloud? | TechRadar*. Retrieved from TechRadar.pro: http://www.techradar.com/news/world-of-tech/roundup/on-site-servers-or-the-cloud-1133159

Interoute Communications. (2017). *What is PaaS? | Interoute*. Retrieved from Interoute: http://www.interoute.com/what-paas

jQuery Foundation. (2006). *jQuery*. Retrieved from jQuery - Write less, do more.: https://jquery.com/

Lazaris, L. (2010). *jQuery Tutorial for Beginners: Nothing But the Goods | Impressive Webs*. Retrieved February 2, 2015, from http://www.impressivewebs.com/jquery-tutorial-for-beginners/

Mark Otto. (2011). *GetBootstrap*. Retrieved January 28, 2015, from http://getbootstrap.com/

Network Specialists. (2014, Jan 13). *Onsite Versus Hosted Servers – Which One Is Right for Your Business? | Network Specialists*. Retrieved from Network Specialists: http://www.networkspecialists.com/onsite-versus-hosted-servers-which-one-is-right-for-your-business/

OpenFlights. (2015). *OpenFlights: Airport and airline data*. Retrieved from OpenFlights.org: https://openflights.org/data.html

Rouse, M. (2015, January). *What is Platform as a Service (PaaS)? - Definition from WhatIs.com*. Retrieved from TechTarget Network: http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS

Sears, A. (2014, February 21). *In-House Server vs. Cloud: Which Option is Better for Your Business? | Business.org*. Retrieved from Business.org: http://www.business.org/it/house-server-vs-cloud-option-better-business/

The PHP Group. (2001). *PHP: Hypertext Preprocessor*. Retrieved Jan 10, 2015, from http://php.net/

# Appendix

Project URL

<https://github.com/thumchoontat/ukraine-air>

Video URL