**London Metropolitan University**

**Faculty of Computing**

**BSc (Hons) Computing**

**PROJECT REPORT HEADER SHEET**

**ID No: 11069940/1**

**Surname: Pokhrel**

**Other Name: Saurav**

**Project Title: Flight Schedule Viewer for Microsoft Flight Simulator X**

**Academic Year: 2012**

**PLAGIARISM**

You are reminded that there exist regulations concerning plagiarism. Please sign below to say that you have understood these regulations:

Signed: …………………… Date: 05/09/2012

**Acknowledgement**

This project is developed as a part of Final Year Project and is made solely for academic and learning process. This project may help others who are interested in learning C# with Visual Studio and .NET framework and developing applications using the mentioned tools and language. The project may also be used as a sample for developing applications for Flight Simulator X using the Simconnect API.

I am grateful toward my friends and classmates who have helped me through the ups and downs related to software development. Furthermore, they have been immensely helpful in cementing out the issues and problems that arose during the development process. Thus I appreciate the help of my friends Anil Kumal and Saajan Man Shrestha along with other classmates.

Moreover, this project would not have been possible to complete without help from my supervisors Mr Rohit Panday and Mr Prabeen Jha who have guided me throughout the software development process. Therefore, I would like to thank them for their unconditional support and guidance.

I would also like to thanks the college, the university, lecturers and tutors for providing knowledge and skill that has helped a lot in developing this project.

Thank You.

**Abstract:**

Flight Simulation Industry has been growing rapidly in past few years paving way for many flight simulation games to be produced like Microsoft’s Flight Simulator, Laminar Research’s X-Plane and Open Source Simulator FlightGear. Also the demand of add-ons for these simulators has been growing in past few years. This project is about developing add-on for one of the above mentioned simulators, Microsoft’s Flight Simulator X (MS FSX), which is able to list flight schedule of airlines at a given airport and also search the airlines and their flight schedules. This is done so as to simulate the use of flight schedule viewer that is used in airports all around the world.

Problem tackled by the project was to integrate to flight schedule viewer with the search module thus increasing the effectiveness and usability of the application. Through the use of incremental prototype model, the system was developed easily and outcome was as expected i.e. aim of developing a flight schedule viewer for Flight Simulator X was achieved.

Keywords: Flight Simulator X, Simconnect API, Flight Schedule Viewer, Flight Simulator X SDK

Table of Contents

[1. Introduction 1](#_Toc324338590)

[1.1. Historical Background 1](#_Toc324338591)

[1.2. Problem Domain 2](#_Toc324338592)

[1.3. Terms and Terminology 2](#_Toc324338593)

[1.4. Structure of the Report: 2](#_Toc324338594)

[2. Background 4](#_Toc324338595)

[2.1. Project Elaboration 4](#_Toc324338596)

[2.2. Comparison with Similar Applications 4](#_Toc324338597)

[2.2.1. Super Traffic Board 4](#_Toc324338598)

[2.2.2. Traffic Toolbox 5](#_Toc324338599)

[2.2.3. Feature Comparison 6](#_Toc324338600)

[2.3. Analysis and Conclusion 6](#_Toc324338601)

[3. Development 7](#_Toc324338602)

[3.1. Language Consideration 7](#_Toc324338603)

[3.2. Methodology Consideration 7](#_Toc324338604)

[4. Requirement Analysis: 10](#_Toc324338605)

[4.1. Objective of this chapter: 10](#_Toc324338606)

[4.2. Introduction: 10](#_Toc324338607)

[4.3. Fact Finding Technique: 10](#_Toc324338608)

[4.4. Introduction of SRS: 10](#_Toc324338609)

[4.4.1. Purpose: 10](#_Toc324338610)

[4.4.2. Intended audience and reading suggestions: 11](#_Toc324338611)

[4.4.3. Project Scope: 11](#_Toc324338612)

[4.5. System Description: 11](#_Toc324338613)

[4.5.1. Functional Requirements: 11](#_Toc324338614)

[Requirement 1: Flight Schedule Viewer 11](#_Toc324338615)

[Requirement 2: Flight Schedule Search Module 11](#_Toc324338616)

[4.5.2. User classes and characteristics 12](#_Toc324338617)

[4.5.2.1. Flight Simulator Users: 12](#_Toc324338618)

[4.5.3. Design and implementation constraints 12](#_Toc324338619)

[4.5.4. Assumptions and Dependencies 12](#_Toc324338620)

[4.6. Features: 12](#_Toc324338621)

[Feature 1: Flight Schedule Viewer: 12](#_Toc324338622)

[Feature 2: Flight Schedule Search Module: 13](#_Toc324338623)

[4.7. External Interface Requirements: 13](#_Toc324338624)

[4.7.1. Software Interface: 13](#_Toc324338625)

[4.7.2. Communication Interface: 13](#_Toc324338626)

[4.8. Non Functional Requirements: 13](#_Toc324338627)

[4.8.1. Performance: 13](#_Toc324338628)

[4.8.2. Security/Privacy: 13](#_Toc324338629)

[4.8.3. Other Software Quality Attribute (SQA) metrics to be considered: 14](#_Toc324338630)

[4.9. Summary of the chapter: 14](#_Toc324338631)

[5. System Design Document: 15](#_Toc324338632)

[5.1. Introduction 15](#_Toc324338633)

[5.1.1. Objective: 15](#_Toc324338634)

[5.1.2. Intended Audiences: 15](#_Toc324338635)

[5.2. System Models: 15](#_Toc324338636)

[5.2.1. User Interface Design: 16](#_Toc324338637)

[5.2.2. Data Flow Diagram: 18](#_Toc324338638)

[5.2.3. Entity Relationship (E-R) Diagram: 19](#_Toc324338639)

[5.2.4. Data Dictionary: 20](#_Toc324338640)

[5.2.5. Use Case Diagram: 22](#_Toc324338641)

[5.2.6. Use Case Description: 23](#_Toc324338642)

[5.2.7. Activity Diagrams: 26](#_Toc324338643)

[5.2.8. Sequence Diagram: 30](#_Toc324338644)

[5.2.9. Summary of the Chapter: 33](#_Toc324338645)

[6. System Implementation: 34](#_Toc324338646)

[6.1. Implementation Issues: 34](#_Toc324338647)

[6.2. Screenshots of Interest: 34](#_Toc324338648)

[6.3. Code Snippet of Interest: 35](#_Toc324338649)

[7. Testing 37](#_Toc324338650)

[7.1. Objective of the chapter: 37](#_Toc324338651)

[7.2. Testing Strategy: 37](#_Toc324338652)

[7.3. Test Plan for the Flight Schedule Viewer: 37](#_Toc324338653)

[7.4. Test Cases: 39](#_Toc324338654)

[7.5. Test Result: 48](#_Toc324338655)

[7.6. Summary of the chapter: 49](#_Toc324338656)

[8. Social and Ethical Issues: 50](#_Toc324338657)

[8.1. Objective of the chapter: 50](#_Toc324338658)

[8.2. Overview: 50](#_Toc324338659)

[8.3. Social and Ethical Issues in Software Development: 50](#_Toc324338660)

[8.4. Summary of the Chapter: 51](#_Toc324338661)

[9. Critical Evaluation and Conclusion: 52](#_Toc324338662)

[9.1. Objective of the Chapter: 52](#_Toc324338663)

[9.2. Project Evaluation: 52](#_Toc324338664)

[9.3. Further Works and Future Development: 52](#_Toc324338665)

[9.4. Summary of the Chapter: 53](#_Toc324338666)

[10. Personal Development Plan: 54](#_Toc324338667)

[10.1. Objective of the chapter: 54](#_Toc324338668)

[10.2. Personal Evaluation: 54](#_Toc324338669)

[10.3. Summary of the chapter: 54](#_Toc324338670)

[11. References 55](#_Toc324338671)

[Appendix 56](#_Toc324338672)

[A. User Manual: 56](#_Toc324338673)

[B. Observation Document for Requirement Analysis: 65](#_Toc324338674)

[C. Program Code: 67](#_Toc324338675)

[D. Gantt Chart: 81](#_Toc324338676)

**Table of Figures**

[Figure 1: Super Traffic Board [Source: Aerosoft.com] 5](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339238)

[Figure 2: Traffic Viewer [Source: Microsoft.com] 5](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339239)

[Figure 3: User Interface Design for Flight Schedule Viewer 16](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339240)

[Figure 4: User Interface design for Flight Search Module 17](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339241)

[Figure 5: Level 0 DFD for Flight Schedule Viewer 18](#_Toc324339242)

[Figure 6: E-R Diagram for Flight Schedule Viewer 19](#_Toc324339243)

[Figure 7: Auto-Generated Database Design 21](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339244)

[Figure 8: Use Case Diagram for Flight Schedule Viewer 22](#_Toc324339245)

[Figure 9: Activity Diagram for Flight Simulator Connection 26](#_Toc324339246)

[Figure 10: Activity Diagram for Viewing Flight Schedule 27](#_Toc324339247)

[Figure 11: Activity Diagram for Searching Flight Schedules 28](#_Toc324339248)

[Figure 12: Activity Diagram for Searching Airline Operation 29](#_Toc324339249)

[Figure 13: Sequence Diagram for Connection with Flight Simulator 30](#_Toc324339250)

[Figure 14: Sequence Diagram for Viewing Flight Schedules 31](#_Toc324339251)

[Figure 15: Sequence Diagram for Searching Flight Schedules 32](#_Toc324339252)

[Figure 16: Sequence Diagram for Searching Airline Operation 33](#_Toc324339253)

[Figure 17: Screenshot for flight schedule search 34](#_Toc324339254)

[Figure 18: Screenshot for main window 35](#_Toc324339255)

[Figure 19: Code snippet for adding data definition for simobjects 35](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339256)

[Figure 20: Code Snippet for requesting flight schedules 36](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339257)

[Figure 21: Test Case 1 - Initial Connection Status 39](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339258)

[Figure 22: Test Case 1 - Connection Status after connection is made 39](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339259)

[Figure 23: Test Case 1 - Message displayed for Connection Established 39](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339260)

[Figure 24: Test Case 2 - Connection Status after disconnecting 40](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339261)

[Figure 25: Test Case 2 - Message for disconnection with Flight Simulator 40](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339262)

[Figure 26: Test Case 3 - Message for no connection with Flight Simulator 41](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339263)

[Figure 27: Test Case 4 - Date and Time for the application 41](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339264)

[Figure 28: Test Case 5 - Message displayed when flight simulator is closed 42](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339265)

[Figure 29: Test Case 6 - Displaying flight schedules in Data Grid View 42](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339266)

[Figure 30: Test Case 7 - Error Message displayed when not connected to FSX 43](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339267)

[Figure 31: Test Case 10 - Message displayed when searching without keyword 44](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339268)

[Figure 32: Error message displayed when no match is found 45](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339269)

[Figure 33: Test Case 12 - Displaying airline operation in Kathmandu 46](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339270)

[Figure 34: Test Case 13 - Displaying flight schedule for Buddha Air 46](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339271)

[Figure 35: Test Case 14 - Displaying search keyword 'VNKT' and 'Buddha Air' 47](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339272)

[Figure 36: Initial Screen for XAMPP Installation 58](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339273)

[Figure 37: Install Location Browser Window 59](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339274)

[Figure 38: Setting Options in the XAMPP installation window 59](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339275)

[Figure 39: Flight Simulator X Traffic Setting Screen 60](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339276)

[Figure 40: Main Window of Flight Schedule Viewer 61](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339277)

[Figure 41: Connection Status after Connecting to FSX 61](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339278)

[Figure 42: Request Flight Schedule Button in the Application 62](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339279)

[Figure 43: Displaying Flight Schedules 62](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339280)

[Figure 44: Search Result for Flight Schedules 63](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339281)

[Figure 45: Search Result for Airline Operation in an Airport 63](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339282)

[Figure 46: Flight Schedule displayed after clicking on Airline Name 63](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339283)

[Figure 47: Observation Document 1 65](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339284)

[Figure 48: Observation Document 2 66](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339285)

[Figure 49: Observation Document 3 66](file:///C:\Users\Acer\Documents\Work%20Folder%20For%20FYP\FYP%20Final%20Report%20-%20Saurav%20Pokhrel%20v2.docx#_Toc324339286)

**Glossary and Abbreviation:**

AI: Artificial Intelligence

API: Application Program Interface

DSDM: Dynamic System Development Method

E-R: Entity Relationship

FSX: Flight Simulator X

MS: Microsoft

PC: Personal Computer

PHP: Hypertext Pre-Processor

SDK: Software Development Kit

STB: Super Traffic Board

UI: User Interface

VB: Visual Basic

# Introduction

This report provides an overall description on the development of flight simulation applications, its historical background and problems related to it in current scenarios.

Flight simulation games are a part of highly popular and lucrative gaming industry (Entertainment Software Association, 2011). Of all the flight simulation software, Microsoft’s Flight Simulator series is one of the most popular flight simulator series till date. Flight Simulator X (FSX) is the latest available simulation game in this series. The popularity of FSX and simulators before it is mostly due to the ability to add user generated contents or add-ons to the game. Provision for open Software Development Kit (SDK) helped a lot in creating interest among users in development of add-on for FS series or more precisely in this context, for FSX. Many new add-ons are now created to make simulation more realistic and enjoyable. There are add-ons for newer aircrafts, airport and other sceneries, weather engines, AI traffics etc. Of these add-on Traffic Viewer (Flight Schedule Viewer) is also one of the add-on that is gaining popularity in recent times as more people are attracted toward AI Aircraft Spotting i.e. hobby of spotting aircrafts in an airport that may range from heavy airliners like Boeing 747 to Business Jets like Cessna Citation X.

## Historical Background

Flight Simulation has a long history dating back to 1928 when a simple trainer where used for training pilots of that time (Moore, 2008). Instrument flight training was started in late 1920’s and by late 1930’s flight simulation training had moved on to electrical and electronic methods. Slowly flight simulation as part of training was accepted by the military of many countries. By 1960s flight simulators had become an important part of commercial airliners. But it was only in 1980 when flight simulation software was developed by Bruce Artwick for Apple II, that concept of flight simulation as of game was perceived. SubLOGIC, a company established by Artwick himself, managed and distributed the software in initial phase. Later Microsoft bought the license for the software and first flight simulator for PC, FS 1.0 was born (SimFlight, 2005). Starting from FS2000, it allowed people to add their own aircraft, scenery or other content to the software (SimFlight, 2005). This opened the floodgate for user generated content in the flight simulation software. Today it is one of the most popular software among aviation enthusiast and others alike. It celebrated its 25 years of development in 2005 (SimFlight, 2005) which strengthens its popularity in flight simulation sector.

## Problem Domain

Though there are add-ons for AI Flight Schedule viewing, they are not without problems. Problems that are not addressed by current available applications (add-on) in the field of Traffic Viewer (Flight Schedule Viewer) are listed below:

1. No single application has both flight schedule monitor and schedule search function integration.
2. These applications assume users to be tech savvy and do lot of configuration themselves before using the application. This makes application viable to only those users that are technically sound and can tinker with configurations and setting.

## Terms and Terminology

1. User generated contents: These are the content that is developed by a user to add new features in a program. This content can range from plug-ins, add-ons, mods or widgets.
2. Software Development Kit (SDK): It is a collection of tools that helps a person to develop a related application or program based on the SDK.
3. Weather Engine: Weather engine in this case refers to application that can generate real-time or custom weather/climate in the game.
4. AI (Artificial Intelligence) Traffic/Aircraft: AI Aircraft is a computer operated aircraft and AI traffic refers to collection of these aircraft flying anywhere in the game.

## Structure of the Report:

This report is divided into further 8 chapters excluding the current chapter i.e. the Introduction chapter. Brief description of each chapter is given below:

Chapter 1 provides a brief background to development of flight simulation software/games and their existing problems. Chapter 2 deals with the introduction to the project and how the existing problems will be tackled by it. It also provides a brief overview on similar software used/found in market currently.

In chapter 3, a detailed view of methodology and language considered and chosen for the development of project is discussed. Chapter 4 deals with requirement analysis process used for the project and also provides System Requirement Specification along with it. Chapter 5 describes the designs and diagrams that are used for the development of the system. This includes system diagrams like use case and context diagram along with database designs.

In chapter 6, test plans for testing system are prepared and test cases with test outcome are produced. Chapter 7 deals with any social or ethical issues that may occur by the use of the system and measures to tackle those issues. Chapter 8 is about critical evaluation of the system and discusses further development process for the system. In chapter 9, personal reflection on the development of system is done along with discussion on effectiveness of tools and languages used, new skills

# Background

## Project Elaboration

The project “Flight Schedule Viewer for Microsoft Flight Simulator X” is an application that will be used to view flight schedules of airlines in a given airport. The application will be connected to Microsoft Flight Simulator X with the help of Simconnect API. Simconnect API is an Application Program Interface that can extract live data like aircraft position, gate position, flight status etc. from the Flight Simulator (Microsoft, 2008a). This data will then be used in the application to provide the flight schedule list in an airport. The application will also have a search module so that client can either search for flight schedules of a particular airlines or airlines operating flights in a particular airport. For this purpose, all the required data like Airport lists, Airline lists, Aircraft lists and Flight schedules are stored in the database. When user searches for any data, the system first checks if the database is open or not and the searched for the matching data. If the data is found it is then displayed to the user.

## Comparison with Similar Applications

One comparable application for Flight Schedule Viewing for FSX is in the market. Also another similar application is provided along with the FSX’s Software Development Kit.

These applications are listed below:

### Super Traffic Board

Super Traffic Board (STB) is a Flight Viewer application for Microsoft Flight Simulator developed by Flying-W Simulations. It provides interactive flight information board that list both User Aircraft and AI aircrafts listed for flights (Flying-W Simulations, n.d). Some features of Super Traffic Board are interactive commands like “View AI” and “Follow AI”, traffic optimisation for departing and approaching aircrafts and displaying of additional aircraft data like speed, distance from airport etc. STB is one of the applications for FSX that uses Simconnect to extract live aircraft data to the application. STB can be used in-game through menu list or in a separate window.

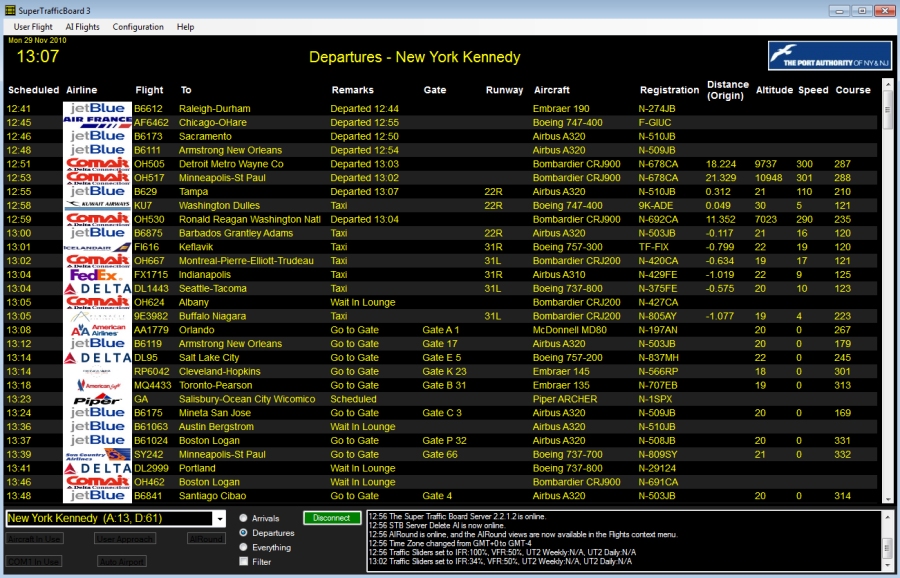
Screenshot of STB is presented below.

Figure 1: Super Traffic Board [Source: Aerosoft.com]

### Traffic Toolbox

Traffic Toolbox is a set of utilities consisting of 4 tools for customization and configuration of AI traffic (Microsoft, 2008b). It is a product of Microsoft and is packaged with the FSX’s Software Development Kit. Traffic Toolbox’s Traffic Explorer Tool is used for viewing AI traffic in the game. Screenshot of Traffic Explorer from Traffic Toolbox is shown below:

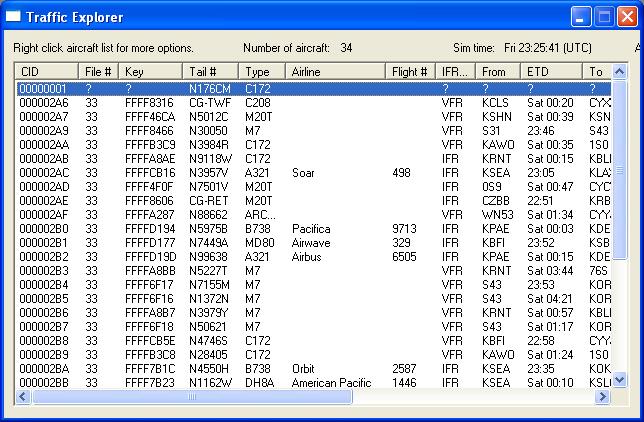


Figure 2: Traffic Viewer [Source: Microsoft.com]

Traffic Explorer does not use Simconnect for extracting data but some other proprietary method is used for doing so. Traffic Explorer can only be used in-game unlike STB.

### Feature Comparison

|  |  |  |
| --- | --- | --- |
| Features | Super Traffic Board | Traffic Explorer |
| View Flight Schedule | Available | Available |
| View Arrival and Departure Separately | Available | Not Available |
| Display Flight Schedules by Airport | Available | Not Available |
| Search Flight Schedules | Not Available | Not Available |
| Follow AI Aircraft | Available | Not Available |

## Analysis and Conclusion

Super Traffic Board is a great application with good interface and some unique features like following AI aircrafts and Follow AI Aircraft. Traffic toolbox is also a good application for viewing AI aircrafts plus it comes with the Flight Simulator X package. But in the both applications mentioned above, they lack search function to search flight schedule of a desired airlines or that operating in a particular airport. This lack of feature in aforementioned software was the main reason for starting this project which will provide both flight schedule viewer along with a search module for searching flight plans.

# Development

## Language Consideration

For the development of this project, languages that were considered at the beginning were C#.net, Vb.net and C++.net all of which uses Microsoft Visual Studio as the base and .NET as the framework. This was done in accordance to the SDK to be used for the development of the project which works with either of the above mentioned languages. Thus other popular languages like PHP and Java were not considered.

Among the languages considered, C#.net was chosen for the following reasons:

1. C# provides powerful yet efficient development environment when combined with Visual studio and .NET for developing desktop applications.
2. When familiarity with the above mentioned languages is considered, due to prior experiences with coding I am more comfortable in C#.NET than in using VB.NET or C++.NET.

No database was considered at the initial point as the proposed project could be developed using file based records. But as file based systems are near obsolete nowadays and as the use of database would make the development easier, faster and efficient, need for a database system arose. For the database to be used MySQL and Microsoft’s SQL Express were considered. Oracle database was not considered for the project because though powerful and packed with security features, it would not be viable for the project that just needs database to store Flight Schedules and Airline’s details. Eventually MySQL was chosen as the backend for the project. Reasons for choosing MySQL over SQL Express are listed below:

1. MySQL is one of the simplest and easiest database systems to install and use.
2. When used with PHPMyAdmin, though primarily used with PHP, provides simple yet intuitive control panel for creating and manipulating the database.
3. As large scale database is not needed for the application, MySQL would better suit the purpose than the SQL Server.

## Methodology Consideration

Software Development Methodology to be used for the project will be Incremental Prototype Model where a prototype of the application will be developed continuously until required specifications are met.

As the application is based on an existing platform i.e. FSX, use of agile methods like DSDM or SCRUM would not be a viable option and may make the development process more cumbersome whereas waterfall model is not really applicable in real time development processes. Thus Incremental method with prototyping looks more suitable for this project.

The reasons for choosing Incremental method with prototyping over other software development methods are listed below:

1. As all requirements may not be found at the initial phase during the project, prototyping will help in finding the needed features for the software and iron out unnecessary features that may bog down the application.
2. Prototype model will deliver application as increments thus it will be easy to divide software into modules and develop one module at a time.
3. It will make testing the application easier as unit testing and component testing can be done side-by-side with the development of the project.

Prototype model generally consists of 4 phase. These phases are listed below (Pressman, 2005):

1. Requirement Gathering: Requirement gathering phase is related with analysis and collecting requirements for the new system. For this purpose, various fact finding techniques can be used and the gathered requirements are filtered. These filtered requirements are then presented as the Requirement Specification Document.

In case of this project, requirement gathering phase will be dedicated to study of the current available applications and working of Simconnect API. Also various posts in the forums will be observed in order to obtain requirements for the system.

1. Design Phase: In the design phase, all the requirements collected from the previous stage will be used to prepare system designs and diagrams like Context diagram, Use case diagrams and User Interface designs.

In design phase the UI designs for the system will be developed. Also various diagrams like UML diagrams including Use case, Sequence and Activity diagram will be prepared for the application. Also database design will be prepared.

1. Development Phase: Development phase is that part where the actual coding of the program is done. Also all testing will be done in the development stage.

In case of this project, programming of application in C#.net will take place in this stage. Furthermore, database will be created and required data will be inserted into the tables. Likewise, testing of the application will also take place in the stage.

1. Evaluation Phase: Evaluation phase is focused on the critical assessment of the developed prototype, module or the complete system.

For the purpose of the project, evaluation phase will deal with assessment of the developed prototype and necessary changes will be made if any errors are found or if any requirements are missing.

# Requirement Analysis:

## Objective of this chapter:

The objective of this chapter is to gather and capture requirements for the proposed system and present in a clear and unambiguous manner. This chapter tries to gather requirement that is needed in order to develop Flight Schedule Viewer with the help of various tools and approaches.

## Introduction:

Requirement Analysis is an important and critical part of software development through which the requirement for the new system is gathered. Requirement Analysis is done after the initial study and must be accurate, concise and clear. The requirements gathered then can be used for developing the designs and further develop into the complete system. As this stage paves way not only for design and development but also for the testing phase, software engineers must be careful while collecting the information from the client and presenting it in the concise and coherent order.

## Fact Finding Technique:

In order to collect requirement for the system, fact finding technique like background study was done. Background study is the process of studying past articles, documents and/or books in order to analyse and find problems related to the current available systems. In this case, it was performed through study of various discussion topics on forums like **FlightSim.com**, **Avsim.com** and **FSDeveloper.com**. Current state of similar application was also studied and features that are lacking from these applications were noted. These requirements were divided into functional and non- functional requirements and are presented below as System Requirement Specification (SRS). Observed sites and posts are shown in the **Appendix** section of this document.

## Introduction of SRS:

### Purpose:

This document provided a detailed overview to the system. Also all the functional and non-functional requirements of Flight Schedule Viewer are defined in the document.

### Intended audience and reading suggestions:

This document is produced for the development team of the system and they will be benefited as the document describes each and every requirements of the system and provides designs and diagrams to support it.

The document is divided into various parts that are listed below:

1. Description of the project including product requirements, user classes and characteristics, design and implementation constraints, assumptions and dependencies.
2. System designs and diagrams.
3. Non-functional Requirements like performance, security/privacy and usability of the system.

### Project Scope:

The project is to develop an application for Microsoft Flight Simulator X that can display flight schedules for AI aircrafts through the use of Simconnect API and also search the flight schedules for these airlines.

## System Description:

## Functional Requirements:

### Requirement 1: Flight Schedule Viewer

1. Displays AI Aircraft data as per Flight plans.
2. Arrange these data in a particular form so as to represent AI Aircraft’s flight schedules.

### Requirement 2: Flight Schedule Search Module

1. Search for Flight Schedules of a particular airline
2. Search for airlines operated in a particular airport.

## User classes and characteristics

### Flight Simulator Users:

These are those people who use Flight Simulator X. They shall perform following functions on the application:

1. They shall view flight schedules of various airlines.
2. They shall search flight schedules of any airline.
3. They shall search for airlines operated from a given airport.

## Design and implementation constraints

1. The application works only with Flight Simulator X and no flight schedules are displayed if FSX is not installed in one’s system.
2. All the flight schedules are stored in the MySQL database, thus it must be installed in the users’ system.
3. The application is developed for the x86 systems and may not work on x64 systems.

## Assumptions and Dependencies

1. User has a copy of FSX installed in his/her system.
2. User has adequate knowledge on installing add-ons for FSX.
3. Correct version of SDK for FSX is installed in the system.
4. MySQL database is installed and flight schedules are stored in this database.
5. User has adequate knowledge on adding data to a MySQL database.

## Features:

### Feature 1: Flight Schedule Viewer:

1. The system shall allow user to view flight schedules of the airlines.
2. The flight schedules shall be displayed only if application is connected to the Flight Simulator.
3. The system shall display additional flight schedules of aircrafts within the user’s range.
4. The flight schedules shall be updated with each change in state of objects in FSX changes.

### Feature 2: Flight Schedule Search Module:

1. The system shall enable users to search flight schedules of airlines.
2. The system shall allow users to search for airlines operating from an airport.
3. The flight schedules shall not be editable from the application.

## External Interface Requirements:

### Software Interface:

1. The system shall require Microsoft Windows operating system, preferably Windows 7 OS, to run the application.
2. Simconnect API shall be required for the connecting the application with the Microsoft Flight Simulator X.
3. All messages like logs and error messages shall be displayed in a particular area within the application.

### Communication Interface:

1. The system shall operate in a single user environment.
2. Multi-Networked environment shall not be supported by this system.

## Non Functional Requirements:

### Performance:

As performance of the system plays an important role in measuring its successful implementation, the application must perform fairly well. Some performance related conditions are listed below:

1. **UI response time: 2 sec**
2. **Connection to Flight Simulator X: 2-4 sec**
3. **Displaying Flight Schedule: 2-4 sec**
4. **Searching for flight schedules: 3 sec**

### Security/Privacy:

In this application security and privacy do not have huge significance as this is a part of a simulation game and do not have any critical user information stored in it.

### Other Software Quality Attribute (SQA) metrics to be considered:

#### Usability:

The application must be easy to use and simple to understand. User interface must be simple yet intuitive.

#### Reliability/Availability:

Reliability and Availability are not one of the critical attributes for this system, yet emphasis must be given for providing relatively high reliability and availability.

#### Maintenance:

Maintenance in this means updates and upgrades to the system which must be provided when and if deemed necessary.

## Summary of the chapter:

This chapter describes fact finding approach used for the requirement analysis of the system. Also a detailed SRS document provided and explained in the chapter that comprises both functional and non-functional requirements of the system.

# System Design Document:

## Introduction

### Objective:

The Design Specification provides an overall description to the designs and diagrams that are required for preparing the application or system. This also provides the working flow of the system and processes involved for it.

### Intended Audiences:

The intended audience for this document are as follows:

1. Programmers: It helps programmers to develop the actual software with the work flow described in the diagrams.
2. Testers: The document helps testers to prepare test plans, test cases and perform testing on the system or application.

## System Models:

System Models are the most vital designs required for the development of any system. Requirements analysed and gathered in the requirement analysis phase is used for preparing these diagrams. Some of the important diagram needed for the system is described below:

1. User Interface Design:

User Interface Design represents the layout of interface to be developed into the real system.

1. Entity Relationship Design:

Entity Relationship Design is the overall layout of physical database represented as entities and its attributes.

1. Context Diagram:

Context diagram describes the interaction between external entities and the system. It also shows the flow of process from system to entities or vice-versa.

1. Use Case Diagram:

Use Case Diagram is a part of UML diagrams and describes the interaction between users and the system.

1. Sequence Diagram:

Sequence Diagram describes the sequence of flow of messages between two or more objects in the system.

1. Activity Diagram:

Activity Diagram represents the flow of processes between user and the system.

### User Interface Design:

#### Flight Schedule Viewer:

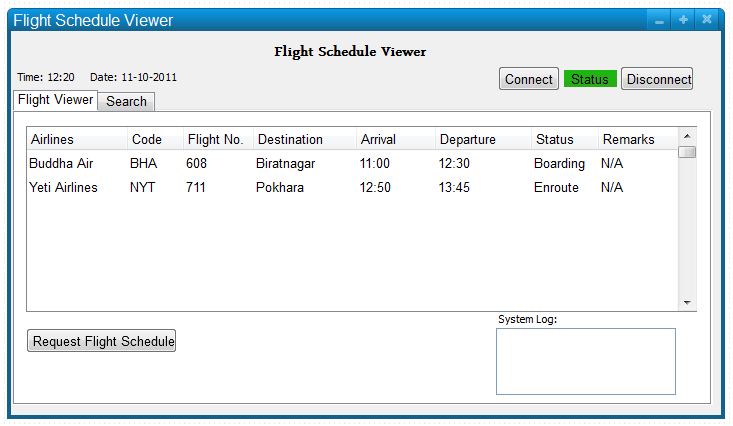
The figure 3 shows the flight schedule viewer screen which is used for viewing flight schedules when requested by user. The “Connect” button is used to connect with the FSX while “Disconnect” button is used for disconnecting with flight simulator. User can request flight schedule by clicking “Request Flight Schedule Button”. Flight Schedules are listed in the grid view i.e. the middle section of the application interface. All the messages are displayed in the System Log section at the bottom corner of screen.

Figure 3: User Interface Design for Flight Schedule Viewer

#### Flight Search Module:

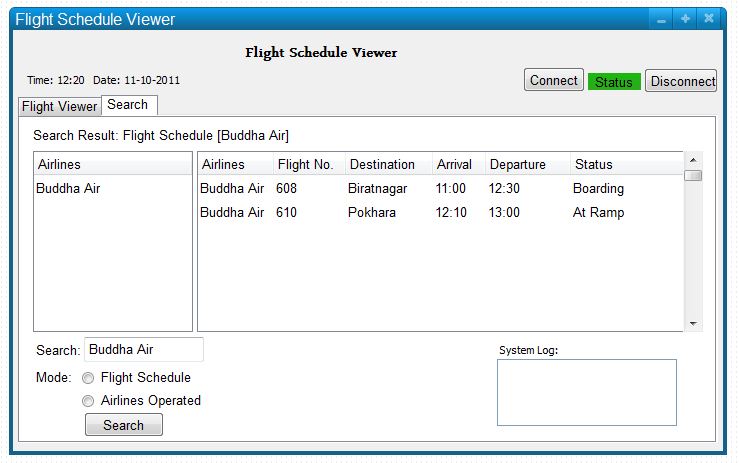


Figure 4: User Interface design for Flight Search Module

The figure 4 shows the flight schedule search module screen which is used for searching flight schedules when searched by the user. The “Search” button is used to search result when user enters search keyword in the textbox and selects from any one of the two radio buttons i.e. Flight schedules or airline operated. If the user has search for flight schedules it is displayed in the right grid view whereas if airline operated is selected the result is displayed in the left grid view. All the messages are displayed in the System Log section at the bottom corner of screen.

### Data Flow Diagram:

Data Flow Diagram is a representation of flow of data between the system and the external entities. The figure below shows the Level 0 DFD (Context Diagram) for the Flight Schedule Viewer Project.



Figure 5: Level 0 DFD for Flight Schedule Viewer

Description of the diagram:

The above figure, describes the context diagram for the Flight Schedule Viewer. Here the user at first sends request for connection with the Flight Simulator through Simconnect. When the connection with FSX is established, the user then requests for Flight Schedule that goes through the Simconnect API. The API then sends requests for simulation objects and simulation variables simultaneously. The requested data when found is sent via Simconnect API and displayed to the user. To search for any flight schedules or airline operation in an airport, the user enters any keyword and searches it. If the keyword matches with the data in the database, the result is then displayed to the user.

### Entity Relationship (E-R) Diagram:

The E-R Diagram for the system is shown below:



Figure 6: E-R Diagram for Flight Schedule Viewer

Description:

It consists of four entity namely Airport, Aircraft, Airport and FlightPlan. Airport entity consists of three attributes AirportID which also acts as the primary key, AirportName and AirportCountry. Aircraft entity is made up of two attributes AircraftName and AircraftID which is the primary key in the entity. Airline entity also has two attributes namely AirlineID and AirlineName where AirlineID is the primary key. FlightPlan entity has 7 attributes as FlightNo, also the primary key in the table, FromAirport, ToAirport, ArrivalTime, DepartureTime, AircraftID and AirlineID. Here FromAirport and ToAirport has the foreign key mapped to AirportID in the Airport entity. AircraftID is the foreign key for Aircraft entity’s AircraftID and similarly AirlineID is the foreign key mapped to the Airline entity’s AirlineID. Aircraft and FlightPlan entity and Airport and FlightPlan entity both has one to many relationship whereas FlightPlan and Airline entity has one is to one relationship.

### Data Dictionary:

Data Dictionary: Airport

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributes | Data Type | Length | Attributes Property | Null |
| AirportID | Varchar | 4 | Primary Key | NO |
| AirportName | Varchar | 256 |  | NO |
| AirportCountry | Varchar | 100 |  | NO |

Data Dictionary: Aircraft

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributes | Data Type | Length | Attributes Property | Null |
| AircraftID | Varchar | 5 | Primary Key | NO |
| AircraftName | Varchar | 256 |  | NO |

Data Dictionary: Airline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributes | Data Type | Length | Attributes Property | Null |
| AirlineID | Varchar | 4 | Primary Key | NO |
| AirlineName | Varchar | 256 |  | NO |

Data Dictionary: Flightplan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attributes | Data Type | Length | Attributes Property | Null |
| FlightNo | Int | 4 | Primary Key | NO |
| FromAirport | Varchar | 4 | ForeignKey (Airport.AirportID) | NO |
| ToAirport | Varchar | 4 | ForeignKey (Airport.AirportID) | NO |
| ArrivalTime | Time |  |  | NO |
| DepartureTime | Time |  |  | NO |
| AirlineID | Varchar | 4 | ForeignKey (Airline.AirlineID) | NO |
| AircraftID | Varchar | 5 | ForeignKey (Aircraft.AircraftID) | NO |

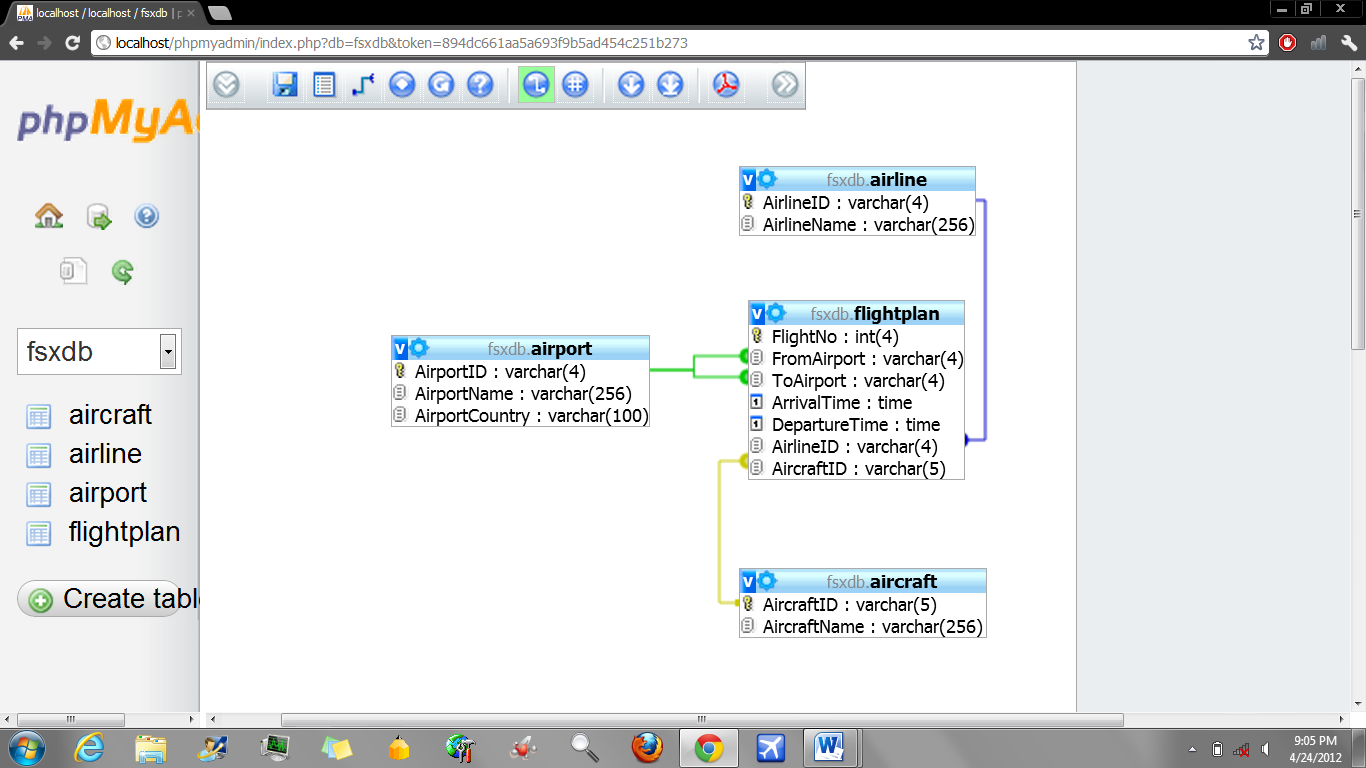
Shown below is the database design generated by the use of PHPMyAdmin describing attributes, its data type, data length and the referential entities between the tables.

Figure 7: Auto-Generated Database Design

### Use Case Diagram:

Use Case Diagram pictorial representation of related interactions or use case between user and the system.



Figure 8: Use Case Diagram for Flight Schedule Viewer

The use case diagram consists of one or more user, system, its boundary and the use cases. Here in the figure, in above figure, user or precisely, flight simulator user clicks on “Connect” button to connect to the Flight Simulator X. After connecting with Flight Simulator, the user can then view flight schedules search the flight schedules or the airline operation in the airport. The user can also disconnect to flight simulator when needed.

### Use Case Description:

|  |  |  |
| --- | --- | --- |
| **S.N.** | **1** | |
| **Use Case** | **Connect to Flight Simulator** | |
| **Actor(s)** | User | |
| **Purpose** | To connect with the Flight Simulator. | |
| **Description** | It is used for the connection with the flight simulator. | |
| **Typical Course of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. It starts when “Connect” Button is pressed or clicked. | | 1. If connection is established, connection status turns green. |
| **Alternative Flow** | | |
| If no connection is made, an error message is displayed. | | |
| **Precondition** | Flight Simulator must be open and running. | |
| **Post Condition** | Connection Status must turn green and connection acknowledgement must be displayed. | |

|  |  |  |
| --- | --- | --- |
| **S.N.** | **2** | |
| **Use Case** | **View Flight Schedule** | |
| **Actor(s)** | User | |
| **Purpose** | To view airline’s flight schedules. | |
| **Description** | It is used for viewing the flight schedules of AI Aircrafts in Flight Simulator. | |
| **Typical Course of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. It starts when “Request Flight Schedules” Button is pressed or clicked. | | 1. If connection is established with the simulator, flight schedules are displayed to the user. |
| **Alternative Flow** | | |
| If no connection is made or if flight simulator is not open, an error message is displayed. | | |
| **Precondition** | Flight Simulator must be open and running, Connection must be established prior to request for flight schedule. | |
| **Post Condition** | Flight Schedules must be displayed in the application. | |

|  |  |  |
| --- | --- | --- |
| **S.N.** | **3** | |
| **Use Case** | **Search Flight Schedules** | |
| **Actor(s)** | User | |
| **Purpose** | To search airline’s flight schedules. | |
| **Description** | It is used for searching flight schedules of a required airline. | |
| **Typical Course of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. This begins if the user enters any keyword and clicks the “Search” Button. | | 1. If keyword matches with the data in the database, required result is displayed. |
| **Alternative Flow** | | |
| If no matches are found for the keyword, an error message is displayed. | | |
| **Precondition** | Connection with the database must be established beforehand. | |
| **Post Condition** | The search result must be displayed to the user. | |

|  |  |  |
| --- | --- | --- |
| **S.N.** | **4** | |
| **Use Case** | **Search Airlines Service in an Airport** | |
| **Actor(s)** | User | |
| **Purpose** | To search airline service in an airport. | |
| **Description** | This is used for searching airlines operation in a given airport. | |
| **Typical Course of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. This begins if the user enters any keyword and clicks the “Search” Button. | | 1. If keyword matches with the data in the database, required result is displayed. |
| **Alternative Flow** | | |
| If no matches are found for the keyword, an error message is displayed. | | |
| **Precondition** | Connection with the database must be established beforehand. | |
| **Post Condition** | The search result must be displayed to the user. | |

|  |  |  |
| --- | --- | --- |
| **S.N.** | **5** | |
| **Use Case** | **Disconnect with Flight Simulator** | |
| **Actor(s)** | User | |
| **Purpose** | To disconnect with the Flight Simulator. | |
| **Description** | It allows user to disconnect from the flight simulator. | |
| **Typical Course of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. User clicks “Disconnect” Button. | | 1. Connection status changes from green to red. 2. System will display a message stating Connection status. |
| **Alternative Flow** | | |
|  | | |
| **Precondition** | Connection with FSX is established | |
| **Post Condition** | A message stating connection state is displayed to the user. | |

### Activity Diagrams:

#### Connection with Flight Simulator:



Figure 9: Activity Diagram for Flight Simulator Connection

Figure above represents the activity diagram for the connection with flight simulator. Here the user at first starts the application and initiates connection request by pressing “Connect” Button. The system or the application then check for the flight simulator state i.e. open or closed with the help of Simconnect API. If the connection is established between application and flight simulator, connection acknowledgement is sent in the form of message. If connection process fails then an error message is displayed to the user.

#### View Flight Schedules:



Figure 10: Activity Diagram for Viewing Flight Schedule

In the figure represented above, activity diagram for viewing flight schedules is shown. Here, connection with flight simulator is established through the same process as in previous activity diagram. After the connection is made, the user then requests for flight schedules. This invokes Simconnect to request SimObjects and Simulation variables from the flight simulator. When the data requested are received, combination of these two data is displayed as flight schedules to the user.

#### Search Flight Schedules:



Figure 11: Activity Diagram for Searching Flight Schedules

In the figure represented above, activity diagram for searching flight schedules is shown. First, the user starts the application and enters search keyword and presses “Search” button. The system then checks if the database and its related tables exists or not. If they exist, it searches for the match with the keyword entered. If no database or its table are found, an error message if displayed. If match is found, the result is displayed to the user or else an error message is displayed.

#### Search Airline Operation in an Airport:



Figure 12: Activity Diagram for Searching Airline Operation

In the figure represented above, activity diagram for searching airline operation in an airport is shown. First, the user starts the application and enters search keyword and presses “Search” button. The system then checks if the database and its related tables exists or not. If they exist, it searches for the match with the keyword entered. If no database or its table are found, an error message if displayed. If match is found, the searched results i.e. airlines are displayed to the user or else an error message is displayed.

### Sequence Diagram:

#### Connection to Flight Simulator:



Figure 13: Sequence Diagram for Connection with Flight Simulator

The figure above represents the sequence diagram for connection with flight simulator. Here, first of all the user presses “connect” button to invoke connection request. This then requests for Simconnect event state i.e. null or not. If the Simconnect state is not null then the connection status is acknowledged to the user.

#### View Flight Schedule:



Figure 14: Sequence Diagram for Viewing Flight Schedules

The figure above represents the sequence diagram for viewing flight schedules. First of all, the user presses “connect” button to invoke connection request. This then requests for Simconnect event state i.e. null or not. If the Simconnect state is not null then the connection status is acknowledged to the user. Then the users requests for flight schedules by pressing “Request Flight Schedule” button. This invokes to methods namely Request SimObjectsByType() and Request Simvariable through AddDataDefination() method. This searches for SimObject, in this case Aircraft and its related flight schedules. This flight schedule is then displayed to the user.

#### Search Flight Schedule:



Figure 15: Sequence Diagram for Searching Flight Schedules

The figure above represents the sequence diagram for searching flight schedules. Here, the user enters search keyword and then presses “search” button to search the database. Search function is then started which searches the database for the correct match. If match is found, the result is then displayed to the user.

#### Search Airline Operation:



Figure 16: Sequence Diagram for Searching Airline Operation

The figure above represents the sequence diagram for searching airline operation in an airport. Initial state is invoked when the user enters search keyword and then presses “search” button to search the database. Search function is then started which searches the database for the correct match. If match is found, the result is then displayed to the user.

### Summary of the Chapter:

This chapter presents detailed view of diagrams and designs to be used for the development of the system. It also provides brief description of each diagram present in the chapter.

# System Implementation:

After the completion of the project, the system will reach the implementation stage where final testing of the system will be done. Any errors found will be corrected and project will be implemented in the client’s work environment.

In this case as the project is about add-on for Flight Simulator, the project when completed will be uploaded to forum like FlightSim.com and FSDeveloper.com for initial response from the forum users. If response is good then it will be uploaded for downloading in same sites as above.

## Implementation Issues:

As the application is developed by using MS Visual Studio and FSX’ SDK, the application cannot be used for commercial purpose. Thus the application will be distributed as freeware which is within the terms and condition of FSX and Visual Studio.

## Screenshots of Interest:

Some screenshots of the application are given below:

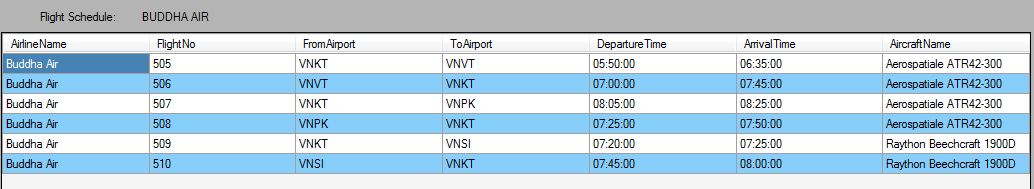


Figure 17: Screenshot for flight schedule search

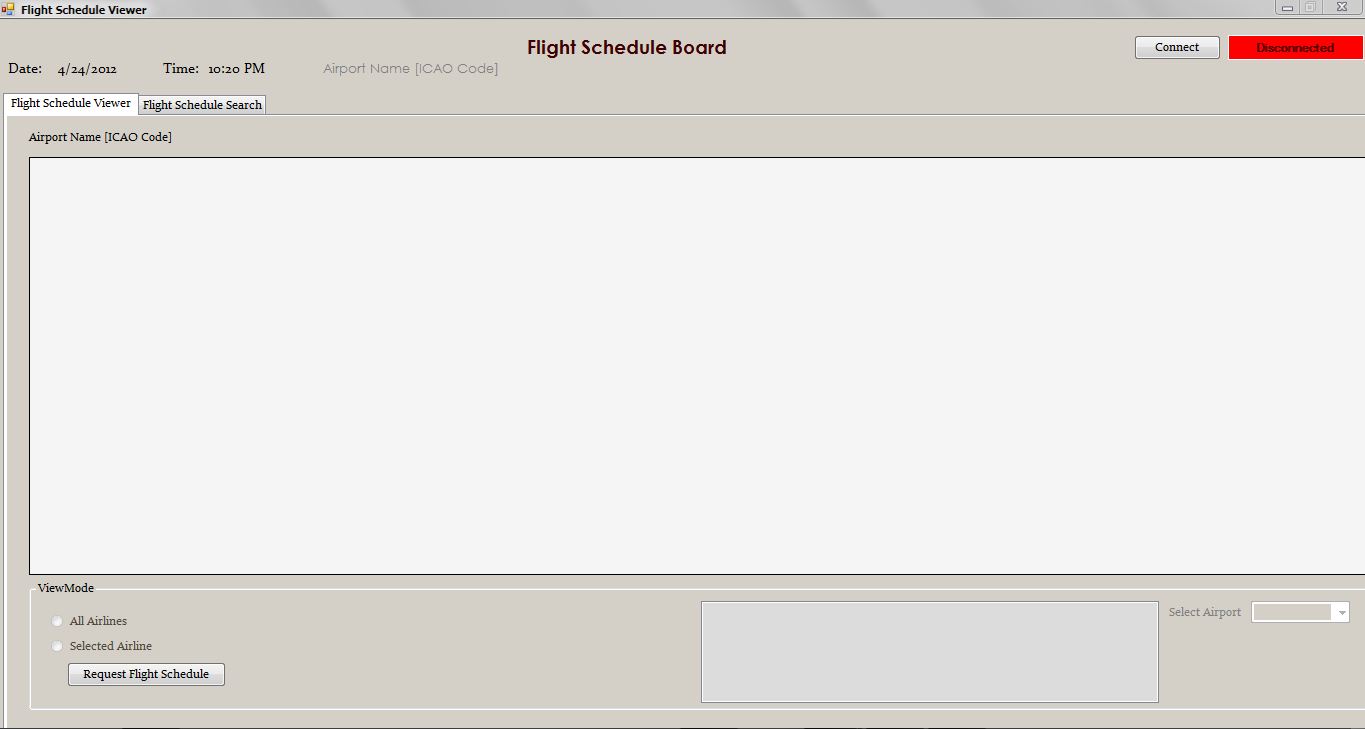


Figure 18: Screenshot for main window

## Code Snippet of Interest:

Some codes that have been used for the project are shown below:

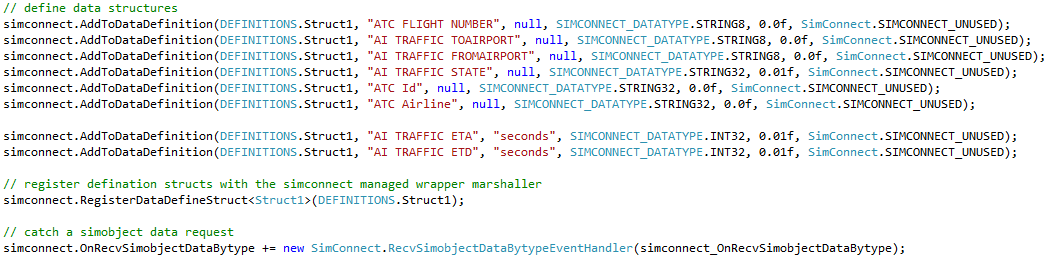
Adding data definition for sim objects:

Figure 19: Code snippet for adding data definition for simobjects

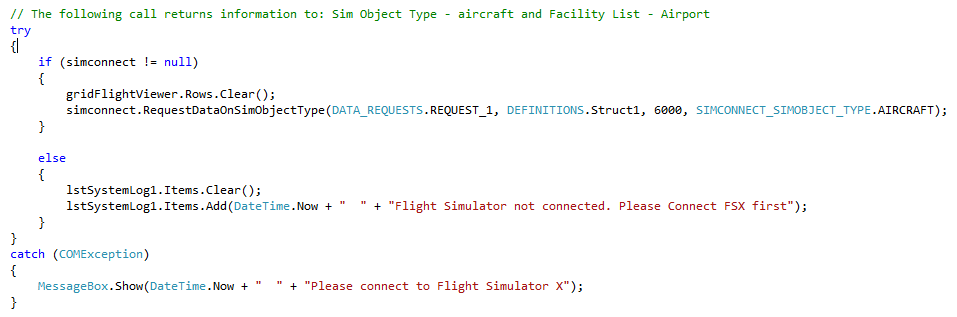
Code for requesting flight schedules:

Figure 20: Code Snippet for requesting flight schedules

# Testing

## Objective of the chapter:

The main objective of this chapter is to provide test plan and test cases for testing the application for errors and bugs. Different testing approaches will be used to ensure that the new system is free of errors and ready to be implemented in real time environment.

## Testing Strategy:

For the testing the new system, various testing methods can be used like unit testing, system testing, component testing, error testing, black box testing and white box testing. In this case, test plan for white box testing and error testing has been prepared. This is to ensure that new system is usable and acceptable even though it is not possible to achieve 100% error free system practically.

## Test Plan for the Flight Schedule Viewer:

Below is the list of test plan to be used for testing the system.

|  |  |  |
| --- | --- | --- |
| S. No | Test Type | Test Case |
| 1 | Use Case Testing (Connect to Flight Simulator) | To check if Flight Schedule Viewer is connected to the Flight Simulator when connect button is pressed |
| 2 | Use Case Testing (Disconnect with Flight Simulator) | To check if Flight Schedule Viewer is disconnected with the Flight Simulator when disconnect button is pressed |
| 3 | Error Testing | To check if Error Message is displayed when connecting Flight Schedule Viewer without opening Flight Simulator |
| 4 | System Testing | To check if Date and time are displayed correctly and properly |
| 5 | Error Testing | To check if Message is displayed if Flight Simulator is closed |
| 6 | Use Case Testing (View Flight Schedule) | To check if Data are displayed in data grid when request flight schedule button is pressed |
| 7 | Error Testing | To check if Error Message is displayed if Flight Simulator is not open and request flight schedule button is pressed |
| 8 | Use Case Testing | To check if Flight Schedule Viewer is disconnected when the Flight Simulator is closed |
| 9 | Error Testing | To check if Flight Schedules are updated with every change in state of SimObjects. |
| 10 | Error Testing | To check if Error Message is displayed when pressing search button without any search keyword and radio button checked |
| 11 | Error Testing | To check if Error Message is displayed if no result is found when searching |
| 12 | Use Case Testing **(**Search Flight Schedules) | To check if Search result for airlines operated is displayed in left data grid when keyword matches and airline operated button is checked |
| 13 | Use Case Testing **(**Search Airline Operation) | To check if Airline's Flight Schedule is displayed in right data grid if keyword matches and flight schedule button is checked |
| 14 | System Testing | To check if Searched keyword is displayed at the top of data grid if search result is found |
| 15 | System Testing | To check if all displayed data are in correct format |
| 16 | Validation Testing | To Check whether the database exists or not. |
| 17 | Validation Testing | To Check whether the tables in the database exist or not. |
| 18 | Validation Testing | To Check whether the Application connects to the database or not. |

## Test Cases:

|  |  |
| --- | --- |
| Case No. | 1 |
| Test Case | To check if flight Schedule Viewer is connected to the Flight Simulator when connect button is pressed |
| Test Prerequisite | Flight Simulator is Open |
| Test Step | Connect button is pressed |
| Expected Result | Flight Schedule Viewer must be connected to Flight Simulator and Connection status button must be green |
| Actual Result | Flight Schedule Viewer was connected to Flight Simulator and Connection status button was green |



Figure 21: Test Case 1 - Initial Connection Status



Figure 22: Test Case 1 - Connection Status after connection is made

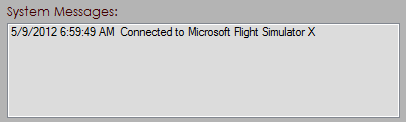


Figure 23: Test Case 1 - Message displayed for Connection Established

|  |  |
| --- | --- |
| Case No. | 2 |
| Test Case | To check if flight Schedule Viewer is disconnected with the Flight Simulator when disconnect button is pressed |
| Test Prerequisite | Flight Simulator is Open |
| Test Step | Disconnect button is pressed |
| Expected Result | Flight Schedule Viewer must disconnect with Flight Simulator and connection status button must be red. |
| Actual Result | Flight Schedule Viewer was disconnected with Flight Simulator and connection status button was red. |



Figure 24: Test Case 2 - Connection Status after disconnecting

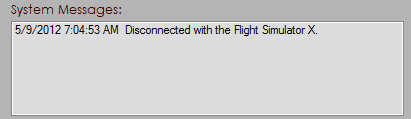


Figure 25: Test Case 2 - Message for disconnection with Flight Simulator

|  |  |
| --- | --- |
| Case No. | 3 |
| Test Case | To check if error Message is displayed when connecting Flight Schedule Viewer without opening Flight Simulator |
| Test Prerequisite | Flight Simulator is not open |
| Test Step | Connect button is pressed |
| Expected Result | Error Message must be displayed in the status box at the bottom of the application |
| Actual Result | Error Message was displayed in the status box at the bottom of the application |

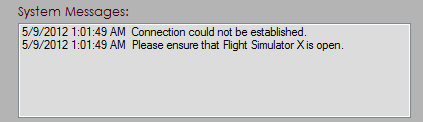


Figure 26: Test Case 3 - Message for no connection with Flight Simulator

|  |  |
| --- | --- |
| Case No. | 4 |
| Test Case | To check if date and time are displayed correctly and properly |
| Test Prerequisite | Not applicable |
| Test Step | Flight Schedule Viewer is opened |
| Expected Result | Current Data and time is displayed in the application |
| Actual Result | Current Data and time is displayed in the application |

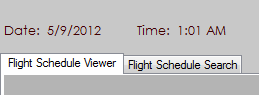


Figure 27: Test Case 4 - Date and Time for the application

|  |  |
| --- | --- |
| Case No. | 5 |
| Test Case | To check if message is displayed if Flight Simulator is closed. |
| Test Prerequisite | Flight Simulator is open |
| Test Step | Flight Simulator is closed |
| Expected Result | Message is displayed in status box. |
| Actual Result | Message is displayed in status box. |

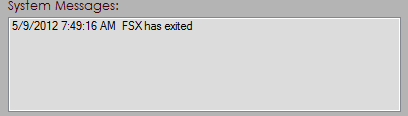


Figure 28: Test Case 5 - Message displayed when flight simulator is closed

|  |  |
| --- | --- |
| Case No. | 6 |
| Test Case | To check if data are displayed in Data grid when request flight schedule button is pressed |
| Test Prerequisite | Flight Simulator is open |
| Test Step | Request Flight Schedule Button is clicked |
| Expected Result | Data must be displayed in the data grid view. |
| Actual Result | Data was displayed in the data grid view. |

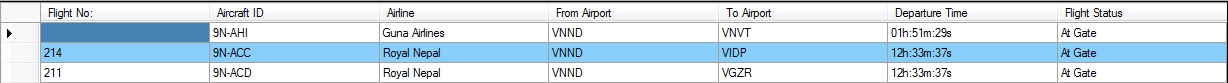


Figure 29: Test Case 6 - Displaying flight schedules in Data Grid View

|  |  |
| --- | --- |
| Case No. | 7 |
| Test Case | To check if error Message is displayed if Flight Simulator is not open and request flight schedule button is pressed |
| Test Prerequisite | Flight Simulator is not open |
| Test Step | Request Flight Schedule Button is clicked |
| Expected Result | Error message must be displayed in the status box |
| Actual Result | Error message was displayed in the status box |

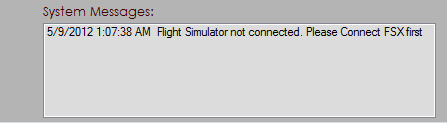


Figure 30: Test Case 7 - Error Message displayed when not connected to FSX

|  |  |
| --- | --- |
| Case No. | 8 |
| Test Case | To check if flight Schedule Viewer is disconnected when the Flight Simulator is closed |
| Test Prerequisite | Flight Simulator is open |
| Test Step | Flight Simulator is closed |
| Expected Result | Connected Status must change to red. |
| Actual Result | Connected Status was change to red. |

|  |  |
| --- | --- |
| Case No. | 9 |
| Test Case | To check if flight Schedules are updated in every 30 seconds automatically |
| Test Prerequisite | Flight Simulator is Open, Data request is sent to Flight Simulator. |
| Test Step | None (Automated) |
| Expected Result | Flight Schedules must be updated with every change in state of SimObjects. |
| Actual Result | Flight Schedules was not updated with every change in state of SimObjects. (Failed) |

|  |  |
| --- | --- |
| Case No. | 10 |
| Test Case | To check if error Message is displayed when pressing search button without any search keyword and radio button is not checked |
| Test Prerequisite | Database is connected, database has needed tables |
| Test Step | Search button is clicked |
| Expected Result | Message must be displayed in the status box |
| Actual Result | Message is displayed in the status box |

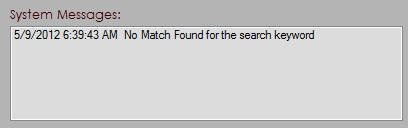


Figure 31: Test Case 10 - Message displayed when searching without keyword

|  |  |
| --- | --- |
| Case No. | 11 |
| Test Case | To check if error Message is displayed if no result is found when searching |
| Test Prerequisite | Database is connected, database has needed tables |
| Test Step | Search text is added and search button is clicked |
| Expected Result | Message must be displayed in the status box. |
| Actual Result | Message is displayed in the status box. |

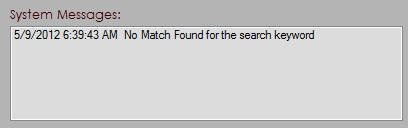


Figure 32: Error message displayed when no match is found

|  |  |
| --- | --- |
| Case No. | 12 |
| Test Case | To check if search result for airlines operated is displayed in left data grid when keyword matches and airline operated button is checked |
| Test Prerequisite | Database is connected, database has needed tables |
| Test Step | Search text is added and search button is clicked |
| Expected Result | Search result for airlines operated must be displayed in left data grid |
| Actual Result | Search result for airlines operated was displayed in left data grid |

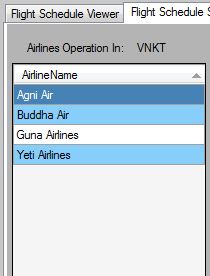


Figure 33: Test Case 12 - Displaying airline operation in Kathmandu

|  |  |
| --- | --- |
| Case No. | 13 |
| Test Case | To check if airline's Flight Schedule is displayed in right data grid if keyword matches and flight schedule button is checked |
| Test Prerequisite | Database is connected, database has needed tables |
| Test Step | Search text is added and search button is clicked |
| Expected Result | Search result for airlines’ flight schedule must be displayed in right data grid |
| Actual Result | Search result for airlines’ flight schedule was displayed in right data grid |

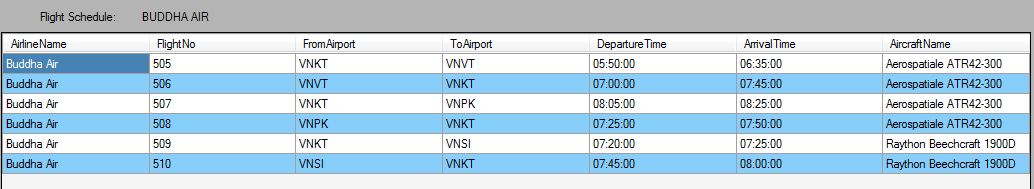


Figure 34: Test Case 13 - Displaying flight schedule for Buddha Air

|  |  |
| --- | --- |
| Case No. | 14 |
| Test Case | To check if searched keyword is displayed at the top of data grid if search result is found |
| Test Prerequisite | Search keyword is found |
| Test Step | N/A |
| Expected Result | Searched keyword must be displayed at the top of data grid |
| Actual Result | Searched keyword is displayed at the top of data grid |

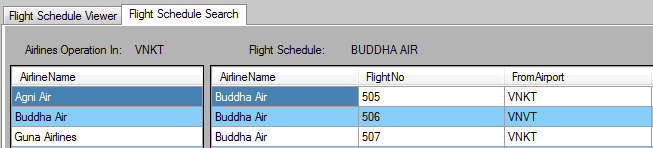


Figure 35: Test Case 14 - Displaying search keyword 'VNKT' and 'Buddha Air'

|  |  |
| --- | --- |
| Case No. | 15 |
| Test Case | To Check if all displayed data are in correct format |
| Test Prerequisite | N/A |
| Test Step | Display data in the data grid view. Check each and every data format. |
| Expected Result | Displayed data must be in correct format. |
| Actual Result | Displayed data was in correct format. |

|  |  |
| --- | --- |
| Case No. | 16 |
| Test Case | To Check whether the database exists or not. |
| Test Prerequisite | Database is created and exists |
| Test Step | Enter any keyword and press search button. |
| Expected Result | No errors or exception must be thrown |
| Actual Result | No errors or exception was thrown |

|  |  |
| --- | --- |
| Case No. | 17 |
| Test Case | To Check whether the tables in the database exists or not. |
| Test Prerequisite | Tables are created and exist. |
| Test Step | Enter any keyword and press search button. |
| Expected Result | No errors or exception must be thrown |
| Actual Result | No errors or exception was thrown |

|  |  |
| --- | --- |
| Case No. | 18 |
| Test Case | To Check whether the database exists or not. |
| Test Prerequisite | Database is created and exists |
| Test Step | Enter any keyword and press search button. |
| Expected Result | No errors or exception must be thrown |
| Actual Result | No errors or exception was thrown |

## Test Result:

|  |  |  |
| --- | --- | --- |
| S. No | Test Case | Test Result |
| 1 | To check if Flight Schedule Viewer is connected to the Flight Simulator when connect button is pressed | Successful |
| 2 | To check if Flight Schedule Viewer is disconnected with the Flight Simulator when disconnect button is pressed | Successful |
| 3 | To check if Error Message is displayed when connecting Flight Schedule Viewer without opening Flight Simulator | Successful |
| 4 | To check if Date and time are displayed correctly and properly | Successful |
| 5 | To check if Message is displayed if Flight Simulator is closed | Successful |
| 6 | To check if Data are displayed in data grid when request flight schedule button is pressed | Successful |
| 7 | To check if Error Message is displayed if Flight Simulator is not open and request flight schedule button is pressed | Successful |
| 8 | To check if Flight Schedule Viewer is disconnected when the Flight Simulator is closed | Successful |
| 9 | To check if Flight Schedules are updated with every change in state of SimObjects. | Failed |
| 10 | To check if Error Message is displayed when pressing search button without any search keyword and radio button checked | Successful |
| 11 | To check if Error Message is displayed if no result is found when searching | Successful |
| 12 | To check if Search result for airlines operated is displayed in left data grid when keyword matches and airline operated button is checked | Successful |
| 13 | To check if Airline's Flight Schedule is displayed in right data grid if keyword matches and flight schedule button is checked | Successful |
| 14 | To check if Searched keyword is displayed at the top of data grid if search result is found | Successful |
| 15 | To check if all displayed data are in correct format | Successful |
| 16 | To Check whether the database exists or not. | Successful |
| 17 | To Check whether the tables in the database exist or not. | Successful |
| 18 | To Check whether the Application connects to the database or not. | Successful |

## Summary of the chapter:

Careful approach to testing had been used for this project. Test plans were used to test the system and produce test cases based on it. Any errors found during testing phase was corrected and removed. Here one test case was unsuccessful and could not be solved. The reason for this issue has been given in Chapter 9 ‘Critical Evaluation and Conclusion’.

# Social and Ethical Issues:

## Objective of the chapter:

This chapter deals with any ethical and social issues that may occur during the development of project or during the implementation of it in the future.

## Overview:

Social and/or Ethical issue in software development is one of those topics that may look trivial and is often neglected but has an important role in defining the success of any projects. Thus consideration must be given while developing any application or system that the developed system does not cause any social problem and is within the social norms and conditions. The system also must remain within the ethical boundary of software development.

## Social and Ethical Issues in Software Development:

As this project is developed using Microsoft’s Flight Simulator X and its Software Development Kit (SDK), it has to comply with the terms and condition or End User License Agreement of Flight Simulator X and the Flight Simulator X’s SDK. FSX’s EULA grants right for enhancing the software through use of add-ons or additional software. Thus it can be ensured that no rules in the license agreement are broken or violated while developing the application. Furthermore as the application does not use or share resources with any of the FSX’s default objects like aircrafts, sceneries or even the textures and models, hence the application is well within the boundary of the EULA.

Another issue that may concern people is security and privacy issues. This is a critical part of life in today’s world where hackers are lurking to get user’s vital information. Thus every measure has been undertaken in order to make sure that no information can be gained by the use of this application.

The software also complies with the terms and conditions of software used during the development like Microsoft Visual Studio and MySQL Database.

**The developed project is an extension to the game and thus no other social issues or complications will incur while or during the use of this application.**

## Summary of the Chapter:

Each and every possible method has been performed to ensure that no rules and regulation of any development software has been broken nor will any social or ethical impact occur through or during the use of the application. Thus the system developed is within the ethical and social boundary of software development.

# Critical Evaluation and Conclusion:

## Objective of the Chapter:

The main objective of this chapter is to evaluate the system and review its initial aims and objectives. This chapter also tries to determine if the objectives of the projects has been fulfilled or not and if methods used were effective enough or not. Detailed discussion on project, its methodology, aims and objectives and outcome of the projects is done in this chapter.

## Project Evaluation:

The main objective of this project was to develop a desktop application for viewing flight schedules in Flight Simulator X and search for needed flight schedules. After the completion of the project, it can be confirmed that this objective has been met but no other additional features could be added in the application. Furthermore, the methodology used for the development of the application has been quite effective but due to its nature, changes to requirements occurred from time to time. Due to this fact, database was incorporated at the latter stage for storing flight schedules. Also various modifications to the User Interfaces had been made.

Moreover, some limitations of Flight Simulator X made the development process harder and time consuming. For e.g., auto updating of flight schedules could not be done as auto update of data only works for objects like vehicles and airport facilities but not for AI Aircrafts. Also data for Airline Name, Airline Registration Number and Flight Number is taken from the ‘Aircraft.cfg’ file that contains all the information of an aircraft. But if no information on the above mentioned data is present in this file, these data will return blank value. Similarly, estimated departure time for aircraft is not precise as FSX do not consider small changes to the time. Thus, it was found that FSX can limit the functionality of the application.

## Further Works and Future Development:

If given chance to start this project again, I would be using the same methodology i.e. incremental prototype but would have focused more on effective requirement gathering process as this would have helped in filtering out the unwanted requirements. Also, I would have used development tools more effectively in order to make application more efficient and easy to use. Some minor changes to UI could have made the application more attractive to the users. This can be done by using images and icons to represent buttons, clearly visible error messages and status messages.

In future version, features that can be added to the application are:

* + 1. Allowing Users to add flight schedules to the database: For this flight schedules that are stored in text format (i.e. in .bgl format) by default can be converted into database readable format and added to the database.
    2. Saving system logs: All the messages shown in the application can be saved in a text file as a system log. This can help in filtering out any bugs if occurred during the use of the application.
    3. Porting the application over mobile devices: As many the applications developed today are made as mobile applications, this project can also be ported to mobile platforms like Windows Phone OS and Android OS.

## Summary of the Chapter:

Initial aims and objectives of the projects have been met but many other features could have been added but were halted due to time limitations. These features can be added in future versions making the application more feature rich and effective. Furthermore, use of different approach to development can help in having a better system with better performance and accuracy.

# Personal Development Plan:

## Objective of the chapter:

The main objective of this chapter is self-evaluation on the skills gained and understanding of the concepts. This chapter tries to reflect the personal effort put and evaluating outcome of that effort.

## Personal Evaluation:

During the process of development of this system, I was able to gain various skills like analytical skills and designing skills. Along with that, my programming skill has been enhanced and I now have better understanding of the Simconnect API and overall interaction of Simconnect API with the Flight Simulator X. This has helped me understand the limitations of FSX and what can be done with it and what cannot be. Moreover, I have learned the pros and cons of using incremental prototype model.

In overall, this project has helped me a lot in learning new skills, concepts and approaches to software development, understanding the issues related to development process and how to tackle it. Furthermore, my own weaknesses with coding and analysis have been revealed which has helped me resolve these issues.

## Summary of the chapter:

Many new skills and techniques have been learned in the process of developing the system and many weaknesses have also been found. This all has helped me gain important lesson that can be implemented in future development activities.

# References

Entertainment Software Association, (2011), *Essential Facts about Computer and Video Game Industry*, Entertainment Software Association [pdf] Available at: <http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf> [Accessed at: 1-6-2012]

Flying-W Simulations (n.d), *Super Traffic Board – Product Features*, Flying-W Simulations [online] Available at: <http://www.supertrafficboard.com/page2.html> [Accessed at: 23-12-2011]

Moore, K (2008), *A Brief History of Aircraft Flight Simulation*, NTLWorld.com [online] Available at: <http://homepage.ntlworld.com/bleep/index.html> [Accessed at: 22-12-2011]

SimFlight.com (2005), *Flight Simulator Timeline*, SimFlight.com [online] Available at: <http://fshistory.simflight.com/fsh/timeline.htm> [Accessed at: 23-12-2011]

Microsoft Corp. (2008a), *Microsoft ESP SDK – Simconnect SDK Reference*, Microsoft Corporation [online] Available at: <http://msdn.microsoft.com/en-us/library/cc526983.aspx> [Accessed at: 23-12-2011]

Microsoft Corp. (2008b), *Microsoft ESP SDK – Traffic Toolbox*, Microsoft Corporation [online] Available at: <http://msdn.microsoft.com/en-us/library/cc526965.aspx> [Accessed at: 23-12-2011]

Pressman, R.S. (2005), *Software Engineering – A Practitioner’s Approach*, Sixth Edition, McGraw Hill Publications

Appendix:

## User Manual:

1. **What is Flight Schedule Viewer?**

Flight Schedule Viewer is an interactive application for displaying Flight Simulator X’s AI Traffic flight schedules. Flight Schedule viewer has two modules for two different purposes. These functions are listed below:

1. **Airport Flight Schedule Board:**

Flight Schedule Viewer acts as an Airport Interactive Flight Schedule Board. In this module the user can request flight schedule details from the flight simulator X and this is displayed in the application window.

1. **Flight Search Module:**

Another function of the Flight Schedule Viewer is to search the individual flight schedules of an airline. The user can also search for airlines that are operated in the particular airport.

1. **Supported AI Packages:**

This section describes the supported AI packages that can be used with Flight Schedule Viewer and also recommend the most suitable AI Traffic package to be used with this application.

Flight Schedule Viewer has been tested with AI packages like World of AI and Custom packages from Alpha India Group, Globusworld.net and United Traffic Team AI Packages. Thus it works properly with these above mentioned AI Packages.

Though not tested with pay-ware AI packages like My Traffic X, Ultimate Traffic X and Traffic X, it must work with these packages without any glitches.

Also, Flight Schedule Viewer can handle any customer made AI flight plans and packages and should work smoothly with these. However, use of World - of - AI packages is mostly recommended with the use of Flight Schedule Viewer.

1. **Requirements**

This chapter provides an overall system requirements needed to run the application. Note that this does not represent the requirements for operating the Flight Simulator X.

1. **Hardware Requirements**

* Recommended Requirements:

1. Intel i3 processors or higher/ AMD equivalent
2. 4 GB DDR3 RAM
3. NVidia GeForce 210GT / AMD Equivalent
4. 20 MB of available Hard Disk.

* Minimum Requirements:

1. Intel Core 2 Duo Dual Core Processors/ AMD equivalent
2. 2 GB DDR3 RAM
3. NVidia GeForce 9600GT / AMD Equivalent
4. 20 MB of available Hard Disk.
5. **Software Requirements**

* Recommended Requirements:

1. Windows 7 SP1
2. Microsoft .NET Framework v2 and v4
3. Flight Simulator X Acceleration
4. MySQL Database.

* Minimum Requirements:

1. Windows XP SP3
2. Microsoft .NET Framework v2 and v4
3. Flight Simulator X SP1
4. MYSQL 5.1 or higher database.
5. **Installation of Software**

Installation of the software is easy and fast. Step by step guide to install the software is given below:

1. Just click on the provided setup.exe file in order to install the software.
2. **Installation of Database**
3. **Using XAMPP server:**

This method also installs the PHP and Apache Server in the system. These services can be stopped latter through the XAMPP control panel.

1. Double click on the XAMPP executable file.

Figure 36: Initial Screen for XAMPP Installation

1. Click next to continue installation process.

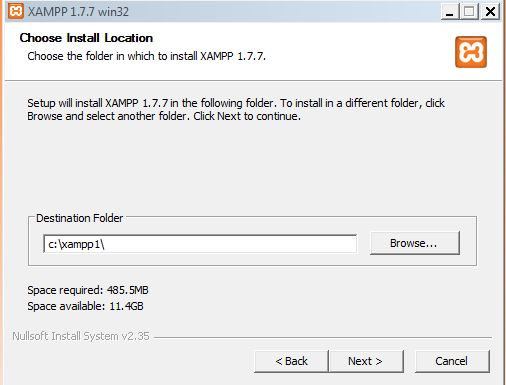


Figure 37: Install Location Browser Window

1. Choose the installation folder and click next.

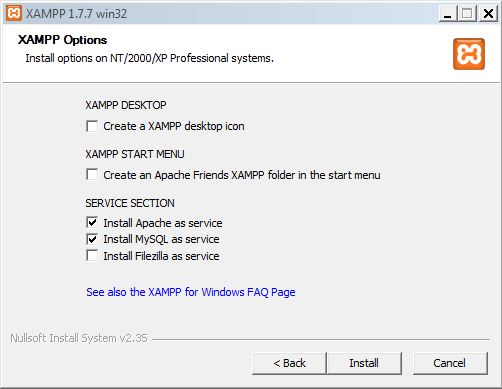


Figure 38: Setting Options in the XAMPP installation window

1. Choose the required services. Make sure to check “Install MySQL as Service”.
2. Click install to begin installation process of MySQL database.
3. **Recommended Settings:**

This chapter presents the recommended setting that must be maintained in Flight Simulator X in order to make optimum use of the application.

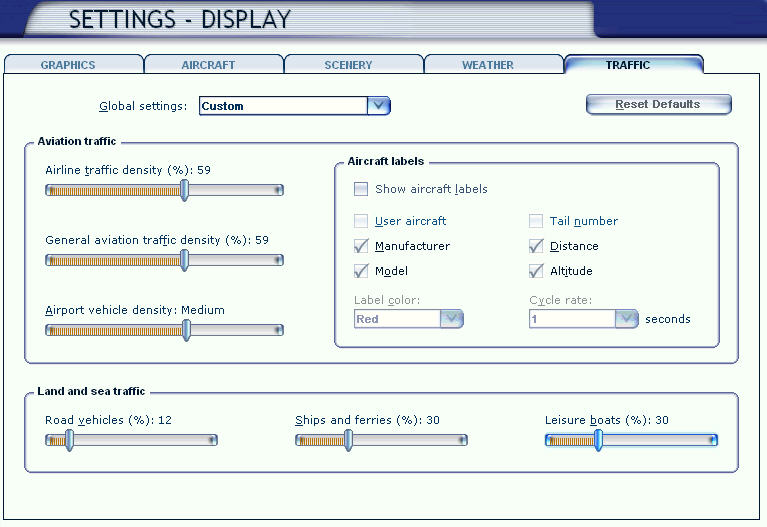
The settings that need change in Flight Simulator are listed below:

Figure 39: Flight Simulator X Traffic Setting Screen

1. When the Flight Simulator X is open, go to the settings menu.
2. In the settings menu, click on the Traffic tab as shown in the figure above.
3. In this window only two of the settings need to be adjusted.
4. Increase the Airline Traffic Density to at least 50% or higher. Higher the value more airliners is displayed. But this may affect the overall performance of FSX.
5. Increase General Aviation Traffic Density to at least 30% or higher.
6. Other settings can be customised according to your need but is not required for the use of Flight Schedule Viewer.
7. **Using Flight Schedule Viewer**

This chapter provided an overall description on the use of flight schedule viewer and how to use it.

1. **Using flight schedule viewer:**

In order to use Flight Schedule Viewer, following steps must be followed:

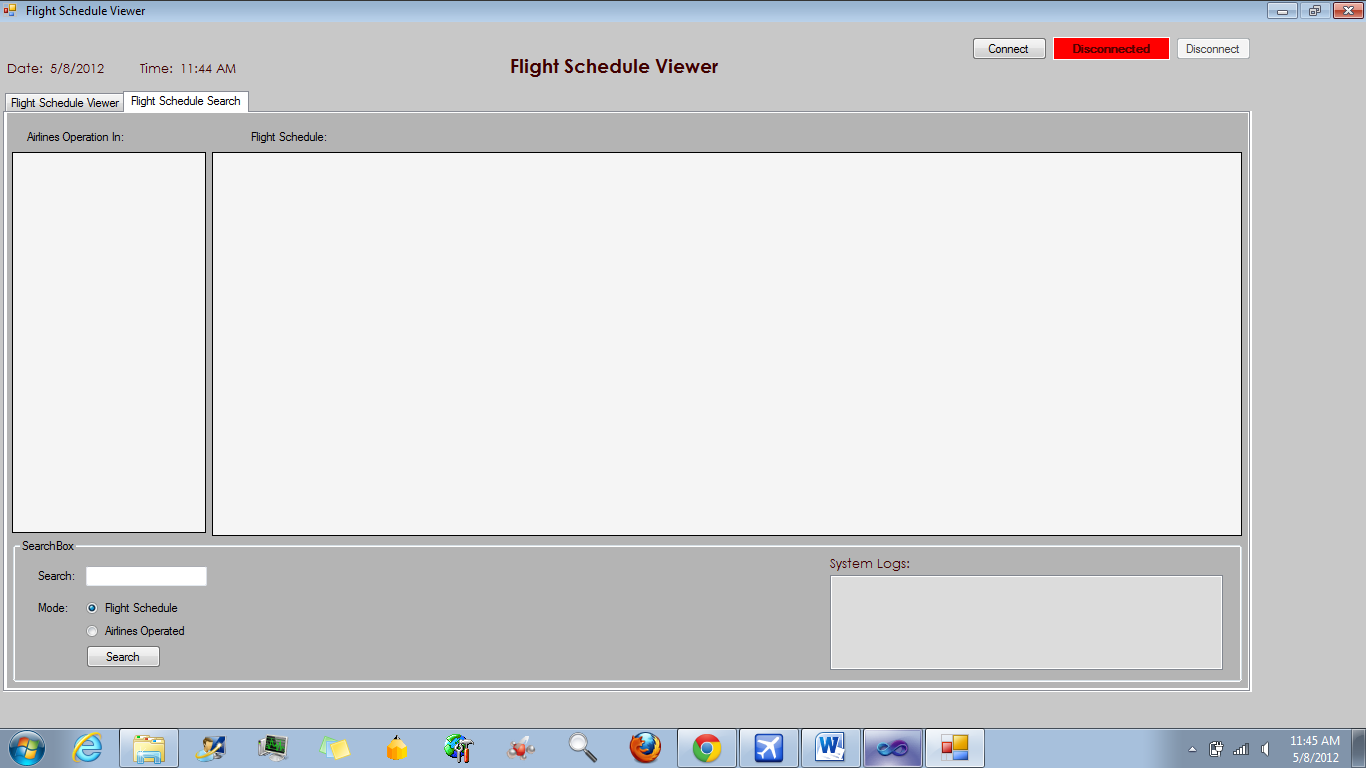
1. Open the application by double clicking the application executable.

Figure 40: Main Window of Flight Schedule Viewer

1. When the application opens, open flight simulator x and start a flight.
2. In the application, press “Connect” button on the top right corner of the screen.



Figure 41: Connection Status after Connecting to FSX

1. When the application is connected with FSX, press “Request Flight Schedule” Button.

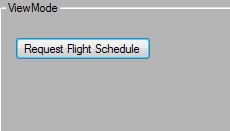


Figure 42: Request Flight Schedule Button in the Application

1. All available lists of airlines and their flight schedules is the shown in the middle space of the application.

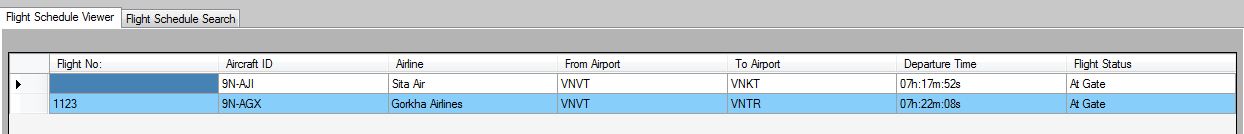


Figure 43: Displaying Flight Schedules

1. **Using flight search module:**

Step by Step approach for using Flight Search Module is given below:

1. Open the application by double clicking the executable file.
2. Toggle to the Flight Search tab to view Flight Search Module window.
3. To search for flight schedule of any airlines, enter airline’s full name in the textbox and check the Flight Schedule button. Press “Search” button to start the search. All available schedules will be displayed in the right grid box. If no result is found, message is displayed in the system log box at the bottom right corner of the application.

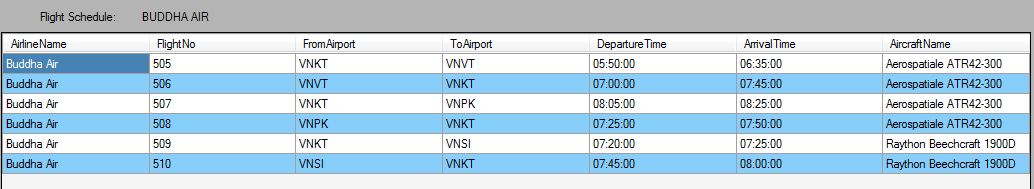


Figure 44: Search Result for Flight Schedules

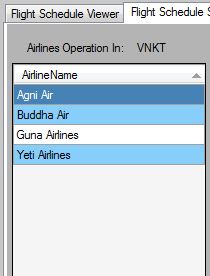
1. To search for airline operation in any airport, enter the airport’s ICAO Code in the textbox and make sure the Airline Operated button is checked. Press “Search” button to start the search. All Airlines operating to and fro the airport will be listed in the left grid box. If no result is found, message is displayed in the system log box at the bottom right corner of the application.

Figure 45: Search Result for Airline Operation in an Airport

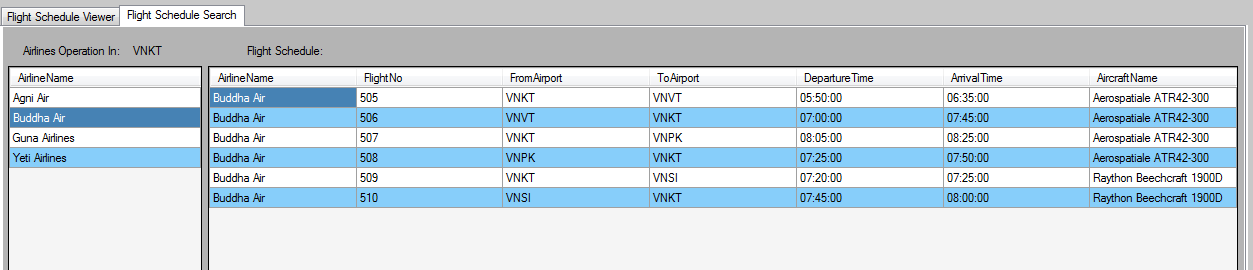
1. You can also click on the Airline Name to view its schedules. Just double click on the Airline’s Name on the left grid to display its flight schedules in the right grid.

Figure 46: Flight Schedule displayed after clicking on Airline Name

[NOTE: All displayed airlines details and airport list vary with the airport, airline and flight plan data available in the database.]

1. **Uninstalling Flight Schedule Viewer**
2. In order to uninstall the Flight Schedule Viewer, just double click on the “uninstall.exe” file on the Flight Schedule Viewer installation folder.
3. This will remove the application from your system. However, this does not remove the database present in your computer.
4. To uninstall the Database, either use XAMPP’s uninstallation facility or else, if using MySQL stand-alone application, just click on uninstall.exe file in the MySQL installation folder.

## Observation Document for Requirement Analysis:

In this section all the documents and posts observed and view for the purpose

Requirement analysis is shown below:

1. Seen at FSDeveloper.com (<http://www.fsdeveloper.com/forum/showthread.php?t=6609>):

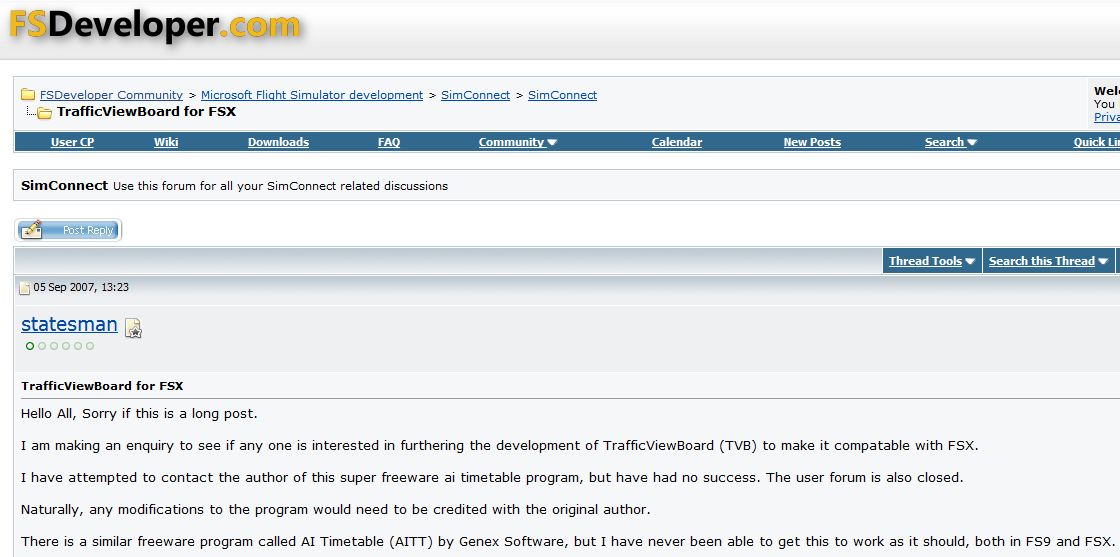


Figure 47: Observation Document 1

Further information on this can be found at the above given link.

1. Another post at FSDeveloper.com ([http://www.fsdeveloper.com/forum/showthread.php?t=15652](http://www.fsdeveloper.com/forum/showthread.php?t=15652&highlight=traffic+board)):

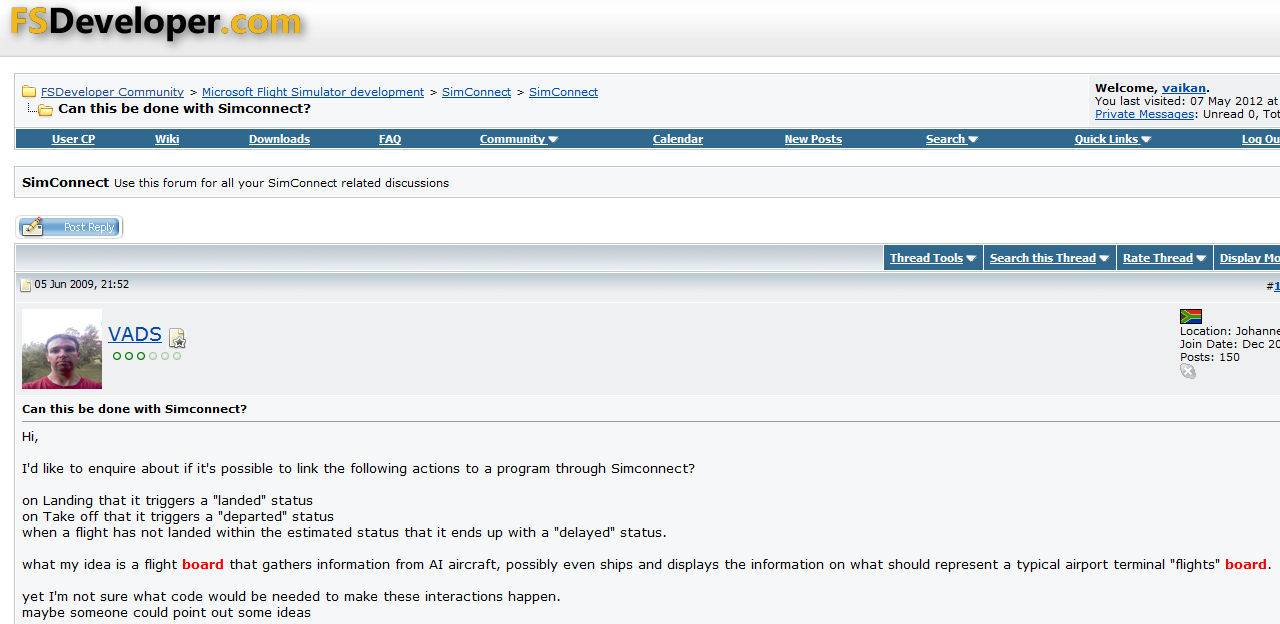


Figure 48: Observation Document 2

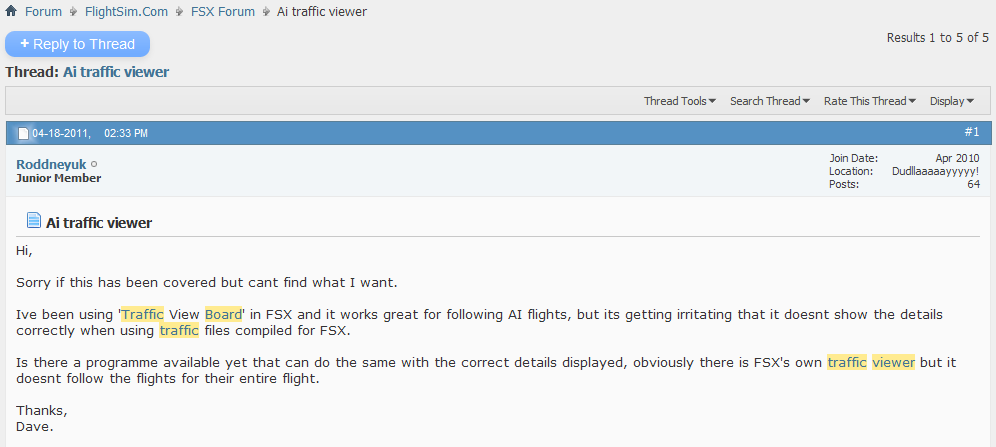
1. Post viewed at FlightSim.com (<http://www.flightsim.com/vbfs/showthread.php?229785-Ai-traffic-viewer>) :

Figure 49: Observation Document 3

## Program Code:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Microsoft.FlightSimulator.SimConnect;

using System.Runtime.InteropServices;

using System.IO;

using System.Collections;

using MySql.Data;

using MySql.Data.MySqlClient;

using System.Web;

namespace Flight\_Schedule\_Viewer

{

public partial class mainForm : Form

{

// initializing a index for datagridview2

int temprowindex;

//connecting to the database

public static string MyConString = "SERVER=localhost;" + "DATABASE=fsxdb;" + "UID=root;" + "PASSWORD=;";

MySqlConnection con = new MySqlConnection(MyConString);

// Declare a SimConnect object

SimConnect simconnect = null;

// User-defined win32 event

const int WM\_USER\_SIMCONNECT = 0x0402;

//add definations structures and data requests

enum DEFINITIONS

{

Struct1,

}

enum DATA\_REQUESTS

{

REQUEST\_1,

};

// declare a data structure so that simconnect knows how to fill it/read it.

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]

struct Struct1

{

// declare a fixed size for string variables

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 8)]

public String flightnumber;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 8)]

public String ToAirport;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 8)]

public String FromAirport;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]

public String aicurstatus;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]

public String atcid;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]

public String atcairline;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 8)]

public String curairport;

public int eta;

public int etd;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]

public String parking;

};

public mainForm()

{

InitializeComponent();

}

//ReceiveMessage must be called to trigger the events.

protected override void DefWndProc(ref Message m)

{

try

{

if (m.Msg == WM\_USER\_SIMCONNECT)

{

if (simconnect != null)

{

simconnect.ReceiveMessage();

}

}

else

{

base.DefWndProc(ref m);

}

}

catch (COMException)

{

// clears all system log messages and shows new messages

lstSystemLog.Items.Clear();

lstSystemLog1.Items.Clear();

lstSystemLog.Items.Add(DateTime.Now + " " +"Flight Simulator has been closed");

lstSystemLog1.Items.Add(DateTime.Now + " " +"Flight Simulator has been closed");

}

}

private void mainForm\_Load(object sender, EventArgs e)

{

string currentdate = DateTime.Now.ToShortDateString(); //to add current date

string currenttime = DateTime.Now.ToShortTimeString(); //to add current time

lbDateDisplay.Text = currentdate;

lbTimeDisplay.Text = currenttime;

}

private void btConnect\_Click(object sender, EventArgs e)

{

//connect to flight simulator

try

{

simconnect = new SimConnect("Managed Data Request", this.Handle, WM\_USER\_SIMCONNECT, null, 0);

btConnectStatus.BackColor = Color.Green;

btConnectStatus.Text = "Connected";

btConnectStatus.ForeColor = Color.White;

btConnect.Enabled = false;

btDisconnect.Enabled = true;

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now + " " +"Connected to Microsoft Flight Simulator X");

initDataRequest();

}

catch (COMException)

{

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now + " "+"Connection could not be established.");

lstSystemLog1.Items.Add(DateTime.Now + " " +"Please ensure that Flight Simulator X is open.");

//MessageBox.Show("Connection could not be established. Please ensure that Flight Simulator X is open and running.");

}

}

private void btDisconnect\_Click(object sender, EventArgs e)

{

//for disconnecting from the Flight Simulator

simconnect.Dispose();

simconnect = null;

btConnectStatus.BackColor = Color.Red;

btConnectStatus.Text = "Disconnected";

btConnectStatus.ForeColor = Color.White;

btDisconnect.Enabled = false;

btConnect.Enabled = true;

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now + " " + "Disconnected with the Flight Simulator X.");

}

private void btSearch\_Click(object sender, EventArgs e)

{

//this section is used for searching the airline operation in the airport using a search keyword//

string searchtext = tbSearch.Text;

MySqlDataAdapter DA = new MySqlDataAdapter();

DataTable table = new DataTable();

if (rbAirlinesOperated.Checked && searchtext != null)

{

table.Rows.Clear();

con.Open();

string sql = "select Distinct AirlineName from flightplan, airport, airline where flightplan.AirlineID = airline.AirlineID and flightplan.FromAirport = airport.AirportID and flightplan.FromAirport =" + "'" + searchtext + "'";

DA.SelectCommand = new MySqlCommand(sql, con);

DA.Fill(table);

if (table.Rows.Count>0)

{

BindingSource bSource = new BindingSource();

bSource.DataSource = table;

dataGridView2.DataSource = bSource;

con.Close();

lbAirlineOperatedHidden.Text = searchtext.ToUpper();

}

else

{

lstSystemLog.Items.Clear();

lstSystemLog.Items.Add(DateTime.Now + " " +"No Match Found for the search keyword");

con.Close();

}

}

//this section is used for searching the airline's flight schedules in the airport using a search keyword//

else if (rbFlightSchedule.Checked && searchtext != null)

{

table.Rows.Clear();

con.Open();

string sql1 = "select AirlineName, FlightNo, FromAirport, ToAirport, DepartureTime, ArrivalTime, AircraftName from flightplan, airport, airline, aircraft where flightplan.AirlineID = airline.AirlineID and flightplan.AircraftID = aircraft.AircraftID and flightplan.FromAirport = airport.AirportID and airline.AirlineName =" + "'" + searchtext + "'";

DA.SelectCommand = new MySqlCommand(sql1, con);

DA.Fill(table);

if (table.Rows.Count > 0)

{

BindingSource bSource = new BindingSource();

bSource.DataSource = table;

dataGridView1.DataSource = bSource;

con.Close();

lbFlightScheduleHidden.Text = searchtext.ToUpper();

}

else

{

lstSystemLog.Items.Clear();

lstSystemLog.Items.Add(DateTime.Now + " " + "No Match Found for the search keyword");

con.Close();

}

}

else

{

lstSystemLog.Items.Add(DateTime.Now + " " + "No search keyword entered or no checkbox selected");

}

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

temprowindex = e.RowIndex; //gets a temporary row index for the data grid view

}

private void dataGridView2\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

//code for viewing flight schedule by clicking on the airlines name in the grid view

MySqlDataAdapter DA = new MySqlDataAdapter();

DataTable table = new DataTable();

table.Rows.Clear();

con.Open();

string sql1 = "select AirlineName, FlightNo, FromAirport, ToAirport, DepartureTime, ArrivalTime, AircraftName from flightplan, airport, airline, aircraft where flightplan.AirlineID = airline.AirlineID and flightplan.AircraftID = aircraft.AircraftID and flightplan.FromAirport = airport.AirportID and airline.AirlineName =" + "'" + dataGridView2.Rows[temprowindex].Cells[0].Value.ToString() + "'";

DA.SelectCommand = new MySqlCommand(sql1, con);

DA.Fill(table);

if (table.Rows.Count > 0)

{

BindingSource bSource = new BindingSource();

bSource.DataSource = table;

dataGridView1.DataSource = bSource;

con.Close();

}

}

// Set up all the SimConnect related data definitions and event handlers

private void initDataRequest()

{

try

{

// listen to connect and quit msgs

simconnect.OnRecvQuit += new SimConnect.RecvQuitEventHandler(simconnect\_OnRecvQuit);

// listen to other unexpected exceptions

simconnect.OnRecvException += new SimConnect.RecvExceptionEventHandler(simconnect\_OnRecvException);

// define data structures

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "ATC FLIGHT NUMBER", null, SIMCONNECT\_DATATYPE.STRING8, 0.0f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "AI TRAFFIC TOAIRPORT", null, SIMCONNECT\_DATATYPE.STRING8, 0.0f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "AI TRAFFIC FROMAIRPORT", null, SIMCONNECT\_DATATYPE.STRING8, 0.0f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "AI TRAFFIC STATE", null, SIMCONNECT\_DATATYPE.STRING32, 0.01f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "ATC Id", null, SIMCONNECT\_DATATYPE.STRING32, 0.0f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "ATC Airline", null, SIMCONNECT\_DATATYPE.STRING32, 0.0f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "AI TRAFFIC ETA", "seconds", SIMCONNECT\_DATATYPE.INT32, 0.01f, SimConnect.SIMCONNECT\_UNUSED);

simconnect.AddToDataDefinition(DEFINITIONS.Struct1, "AI TRAFFIC ETD", "seconds", SIMCONNECT\_DATATYPE.INT32, 0.01f, SimConnect.SIMCONNECT\_UNUSED);

// register defination structs with the simconnect managed wrapper marshaller

simconnect.RegisterDataDefineStruct<Struct1>(DEFINITIONS.Struct1);

// catch a simobject data request

simconnect.OnRecvSimobjectDataBytype += new SimConnect.RecvSimobjectDataBytypeEventHandler(simconnect\_OnRecvSimobjectDataBytype);

}

//catch any exceptions and show message.

catch (COMException ex)

{

MessageBox.Show(ex.Message);

}

}

// when the user closes FSX

void simconnect\_OnRecvQuit(SimConnect sender, SIMCONNECT\_RECV data)

{

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now+" "+ "FSX has exited");

simconnect.Dispose();

simconnect = null;

btConnectStatus.BackColor = Color.Red;

btConnectStatus.Text = "Disconnected";

btConnectStatus.ForeColor = Color.White;

btDisconnect.Enabled = false;

btConnect.Enabled = true;

}

void simconnect\_OnRecvException(SimConnect sender, SIMCONNECT\_RECV\_EXCEPTION data)

{

lstSystemLog1.Items.Clear();

lstSystemLog.Items.Clear();

lstSystemLog.Items.Add(DateTime.Now + " " + "Exception received: " + data.dwException);

lstSystemLog1.Items.Add(DateTime.Now + " " + "Exception received: " + data.dwException);

}

void simconnect\_OnRecvSimobjectDataBytype(SimConnect sender, SIMCONNECT\_RECV\_SIMOBJECT\_DATA\_BYTYPE data)

{

switch ((DATA\_REQUESTS)data.dwRequestID)

{

case DATA\_REQUESTS.REQUEST\_1:

Struct1 s1 = (Struct1)data.dwData[0]; //arrange all data from "Struct1" in a array

gridFlightViewer.ColumnCount = 7; //define column number needed in the datagridview

//add name to each of the columns

gridFlightViewer.Columns[0].Name = "Flight No:";

gridFlightViewer.Columns[1].Name = "Aircraft ID";

gridFlightViewer.Columns[2].Name = "Airline";

gridFlightViewer.Columns[3].Name = "From Airport";

gridFlightViewer.Columns[4].Name = "To Airport";

gridFlightViewer.Columns[5].Name = "Departure Time";

//gridFlightViewer.Columns[6].Name = "Arrival Time";

gridFlightViewer.Columns[6].Name = "Flight Status";

//exclude user aircraft and its data

if (s1.etd !=0 && s1.eta !=0)

{

TimeSpan arr = TimeSpan.FromSeconds(s1.eta);

string arrs = string.Format("{0:D2}h:{1:D2}m:{2:D2}s",

arr.Hours,

arr.Minutes,

arr.Seconds);

TimeSpan dept = TimeSpan.FromSeconds(s1.etd);

string depts = string.Format("{0:D2}h:{1:D2}m:{2:D2}s",

dept.Hours,

dept.Minutes,

dept.Seconds);

//change all AI Statuses to make it understandable by the user.

if (s1.aicurstatus == "sleep")

{

s1.aicurstatus = "At Gate";

}

else if (s1.aicurstatus == "preflight support")

{

s1.aicurstatus = "Boarding";

}

else if (s1.aicurstatus == "Push back 1" || s1.aicurstatus == "Push back 2" || s1.aicurstatus == "clearance")

{

s1.aicurstatus = "Final Check";

}

else if (s1.aicurstatus == "pre taxi out" || s1.aicurstatus == "taxi out")

{

s1.aicurstatus = "Taxiing";

}

else if (s1.aicurstatus == "takeoff 1" || s1.aicurstatus == "takeoff 2")

{

s1.aicurstatus = "Departed";

}

else if (s1.aicurstatus == "enroute")

{

s1.aicurstatus = "Enroute";

}

else if (s1.aicurstatus == "landing")

{

s1.aicurstatus = "Landing";

}

else if (s1.aicurstatus == "go around")

{

s1.aicurstatus = "Delayed";

}

else if (s1.aicurstatus == "postflight support")

{

s1.aicurstatus="Post Flight Check";

}

//create new string and add all data in that string in a row

string[] row = new string[] { s1.flightnumber, s1.atcid, s1.atcairline, s1.FromAirport, s1.ToAirport, depts, s1.aicurstatus};

gridFlightViewer.Rows.Add(row); //adds all data to the datagridview

}

break;

default:

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now + " " + "Unknown request ID: " + data.dwRequestID);

break;

}

}

private void btReqData\_Click(object sender, EventArgs e)

{

// The following call returns information to: Sim Object Type - aircraft and Facility List - Airport

try

{

if (simconnect != null)

{

gridFlightViewer.Rows.Clear();

simconnect.RequestDataOnSimObjectType(DATA\_REQUESTS.REQUEST\_1, DEFINITIONS.Struct1, 6000, SIMCONNECT\_SIMOBJECT\_TYPE.AIRCRAFT);

}

else

{

lstSystemLog1.Items.Clear();

lstSystemLog1.Items.Add(DateTime.Now + " " + "Flight Simulator not connected. Please Connect FSX first");

}

}

catch (COMException)

{

MessageBox.Show(DateTime.Now + " " + "Please connect to Flight Simulator X");

}

}

}

}

1. Gantt Chart: