

M32COM

INTERNET SYSTEMS DEVELOPMENT

Coursework Documentation

|  |  |
| --- | --- |
| **Student:** | **ID:** |
| Eduard Rumen Georgiev | 8240390 |
| Saif Mohammad Alzubi | 6674234 |
| Oumaru Beyai | 8042749 |
|  | |
| **Lecturer:** | Dr Leon Smalov |
| **Image result for github icon GitHub link:** | <https://github.com/ERGeorgiev/MidlandsFly> |

Table of Contents

[1 Introduction 2](#_Toc499859420)

[1.1 Objective 2](#_Toc499859421)

[1.2 Scenario 2](#_Toc499859422)

[1.3 Description and Features 2](#_Toc499859423)

[1.4 Roles 3](#_Toc499859424)

[1.5 Software and Services 3](#_Toc499859425)

[2 Project Management 4](#_Toc499859426)

[2.1 Development Process 4](#_Toc499859427)

[2.1.1 Design 5](#_Toc499859428)

[2.1.2 Programming 5](#_Toc499859429)

[2.1.3 Quality Assurance 5](#_Toc499859430)

[2.2 Contribution 6](#_Toc499859431)

[2.3 Communication 7](#_Toc499859432)

[3 Methodology 7](#_Toc499859433)

[3.1 Simulation 8](#_Toc499859434)

[3.1.1 Aircraft Timeline 8](#_Toc499859435)

[3.2 XML Data 9](#_Toc499859436)

[3.3 Error-Handling 9](#_Toc499859437)

[3.4 Table Creation 9](#_Toc499859438)

[3.5 Encryption Size Calculation 9](#_Toc499859439)

[4 Software development life cycle 9](#_Toc499859440)

[5 Testing 9](#_Toc499859441)

[6 Evaluation and Future Technologies 10](#_Toc499859442)

[7 Conclusion 10](#_Toc499859443)

[8 References 10](#_Toc499859444)

# Introduction

This is the final documentation of the group coursework for Coventry University, module M32COM. The aim of this project is to fully develop a database management website for an airline company called ‘Midlands Fly’. Users can access the database data by using the registration page to make an account and then logging in. The website offers a simulation mode from the menu that can be used to insert fake data into the database and simulate how it is being processed as time passes. Different ASP.NET software patterns were implemented using Visual Studio, such as Identity and Web Forms. All sensitive user information is encrypted and stored in the database. The graphical user interface (GUI) is simple and functional, yet very secure, stable and error-free.

## Objective

The primary objective of this documentation is to provide a concise overview of the project’s planning, development, implementation, testing phases and deployment. It will explain the basic methodologies behind database creation, database management, back and front-end programming, error-handling and overall production process.

## Scenario

Due to the ever growing and rapid development of information technology, many organisations are turning to online services for a more precise and easy to use database system. Companies such as Midlands Fly need websites to allow their employees to securely access company information online in order to have real-time aircraft data.

## Description and Features

The website allows registered employees to explore tables filled with data about different types of aircraft, maintenance, history and assignment. Webpages that contain database information are strictly forbidden to users that are not registered, trying to access any of those pages will lead the user directly to the login page.

Sensitive data such as usernames are encrypted using Microsoft Identity encryption and stored in the database along with the password. Employee names are encrypted by a passphrase, a method that is provided by Microsoft Azure. Registered employees that have access to the database pages can view unencrypted employee names.

Using a smart update system, the web server has the ability to record and synchronise total number of flying hours for each aircraft and its assigned employees that are part of the crew. Each aircraft is serviced on landing after flying for a certain number of hours.

After going through maintenance, repairs done by each employee are recorded in the maintenance history table and associated with the aircraft.

Tables that contain too much data can easily be filtered by table registration number. The registration number entered in the provided text box is verified both client and server-side. The user is notified when a wrong format has been detected.

For simulation purposes, registered users are also provided with the option to generate new data, reset tables or add a certain number of hours to the simulation. There is also a feature on every page that allows employees to put the server in “Seconds” mode, where it treats seconds like hours. Since updating all records every second would put too much strain on the server, the “Fast” mode option comes with added flood protection, where the server will only update the database once every 10 seconds.

## Roles

1. Unregistered Users
   * Home page access
   * Registration (if enabled)
   * Login page access
2. Registered Users
   * Full database tables access
   * Account management
   * Server “Seconds” mode
3. Website
   * User GUI
   * Active and PostBack data validation
   * Provides users with usage help
   * Displays server message
4. Server
   * Server-side data validation
   * Error-handling
   * Informing users about certain events
   * Security and encryption
   * Interfacing with the database
5. Database
   * Provide sufficient storage procedures for the server to use
   * Security, firewall and encryption
   * Execute database queries sent from the server

## Software and Services

1. Visual Studio Community 2017
2. Microsoft Azure
3. Notepad++
4. Microsoft SQL Server
5. GitHub
6. Microsoft Visio 2016

# Project Management

Each group member was assigned a task that complements their skills and gives them the opportunity to further develop themselves.

## Development Process

The project was split in many parts, so each group member can contribute.

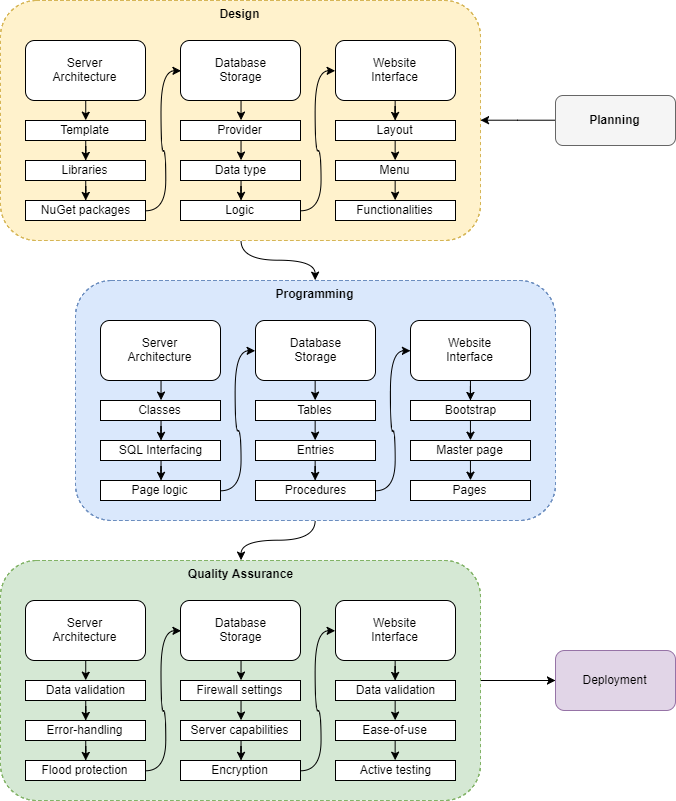


Figure 2.1 – Development Process

### Design

|  |  |  |
| --- | --- | --- |
| Server Architecture | Database Storage | Website Interface |
| * Template | * **Provider** | * **Layout** |
| A project or a website template is chosen. The template used in this project is Web Forms. | The group decided to use Microsoft Azure as the database server provider. | The template used in the project provided a sufficient GUI layout for the website. |
| * Libraries | * **Data type** | * **Menu** |
| Deciding which libraries and functions to use. | Variable type and length are agreed on. | Minimalist page buttons were decided upon, along with a simulation drop-down list. |
| * NuGet packages | * **Logic** | * **Functionalities** |
| Installing services that the server will need to function correctly. | Developing equations to add hours and calculate aircraft status. | The menu provides simulation tools, easy page access, references and contacts. |

Table 2.1 – Design process

### Programming

|  |  |  |
| --- | --- | --- |
| Server Architecture | Database Storage | Website Interface |
| * Classes | * **Tables** | * **Bootstrap** |
| Designing classes that will represent database entries and their variables. | Creating tables. | Modifying the already provided bootstrap to fit the project’s needs. |
| * SQL Interfacing | * **Entries** | * **Master page** |
| Creating classes and methods that connect and manipulate the database. | Adding default entries. | Making changes to the master page to reflect the developer’s vision. |
| * Page logic | * **Procedures** | * **Pages** |
| Developing page code-behind, enabling simulation controls. | Creating procedures for server-database interfacing. | Adding page-specific methods and functions. |

Table 2.2 – Programming Process

### Quality Assurance

|  |  |  |
| --- | --- | --- |
| Server Architecture | Database Storage | Website Interface |
| * Data validation | * **Firewall settings** | * **Data validation** |
| Adding server-side data validation so users cannot invoke a crash. | Setting up the server firewall to deter all unauthorized access. | Implementing code for active user input data validation. |
| * Error-handling | * **Server capabilities** | * **Ease-of-use** |
| Encapsulating all methods that could cause an error. | Deciding on what the server specifics and limits are and need to be. | Making sure that the website is easy to understand and accessible. |
| * Flood protection | * **Encryption** | * **Active testing** |
| Restricting updates that happen too often on PostBack. | Performing encryption by a Passphrase. | Testing the whole system using the graphical user interface. |

Table 2.3 – Programming Process

## Contribution

This section is going to highlight the overview of the group work how it is conducted.

|  |  |
| --- | --- |
| Description | Group member contribution |
| Planning | *Group effort* |
| Website and Server Programming   * C# * HTML * JavaScript * XML | Eduard Georgiev |
| Database Programming   * Stored Procedures * T-SQL | Saif Alzubi  Eduard Georgiev |
| Database Design | Saif Alzubi |
| Database Management | Saif Alzubi |
| Microsoft Azure Implementation | *Group effort* |
| Tutorial for Colleagues | Oumaru Beyai |
| Data Encryption | Eduard Georgiev  Saif Alzubi |
| Data Validation | Eduard Georgiev |
| Error-Handling | Eduard Georgiev |
| Research | Oumaru Beyai |
| Documentation | Eduard Georgiev  Oumaru Beyai |
| Testing | *Group effort* |
| Implementation | *Group effort* |
| NewCycle Equations | Eduard Georgiev |
| Simulation Implementation | Saif Alzubi  Eduard Georgiev |
| GitHub | *Group effort* |

Table 2.4 - Group Contribution

Total time spent: **200+ hours**.

## Communication

Communication planning is necessary in any group project, as it enables the group leader to establish roles and assign tasks. It can be essential towards project success, as agreed by Cleveland and Ireland (2006) when they wrote ‘There is a direct relationship between project success or failure and the effectiveness of the communication…’.

The means of communication and project updates in this group work was mainly carried out using **GitHub** and group messaging on a daily basis.

All individual member contributions can be observed on the project’s **GitHub** page.

**Image result for github icon GitHub link:** <https://github.com/ERGeorgiev/MidlandsFly>

# Methodology

The standard application architecture consists of a presentation layer, business layer and a database layer that accesses the database (Microsoft Azure).

|  |  |
| --- | --- |
| Layers | Architecture |
| Presentation layer | Home page, login, database-access tables, simulation tools. |
| Business layer | Database management, validation, updates, encryption, verification, login logout. |
| Database layer | Database management, Transact-SQL, Microsoft Azure access. |

Table 3.1 - Layers

The first layer of the architecture is mainly presentation layer of the user interface that allows registered users to view all flight and cargo activities. The browser displays information provided by the server.

The business layer processes all activities and lets the user communicate with the database layer. Furthermore, it provides encryption, error-handling and data verification. It is run from a trusted server that accepts user requests.

The database layer is where most of the data management is carried out and data such as aircraft specifications, employee details and account credentials are stored.

For a more complex solution, the layers may further be divided into sections. This can often provide higher security and reusability. The business layer can be replaced with Application, Domain and Business services.

By using a multi-layered application architecture, the developer doesn’t have to rely on third party services, which can often introduce communication difficulties.

Many classes in this project implement a **Singleton** to provide more functionality to the programmer.

## Simulation

Since this project is a coursework and there are no actual users to enter their data into the website, a simulation is necessary to test and prove various functionalities.

### Aircraft Timeline

#### Aircraft Stages

By the coursework’s task, an aircraft needs to enter maintenance every 200 flying hours for 24 hours. It is also logical that it cannot fly forever. In that regard, simply adding hours to each aircraft and checking for maintenance every 200 hours could be considered unimaginative. This is further reinforced by the fact that the coursework explicitly states “At the end of a flight, the flying hours of the aircraft and crew are updated automatically”, so the aircraft should be able to land, prepare for take-off and fly. It is also common knowledge that aircraft flights are not all of equal duration, so aircraft flight duration has to be randomized after each full course. Since adding hours to aircraft can All of these requirements make the action of adding hours to an airplane immensely complex.

The following simple graphical representation explains how aircraft stages work:

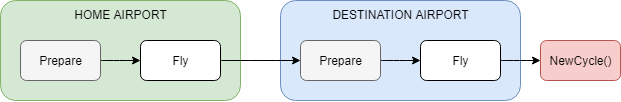


Figure 3.1 – Aircraft Stages

Each arrow represents time that has to pass in order to proceed to the next stage. Preparation time in this case would be 1 hours, while flight time would vary between 1 and 18 hours. The NewCycle procedure at the end generates a new flight routine and calculates the next plane stage.

|  |  |  |
| --- | --- | --- |
| No.: | Name: | Description: |
| 0 | **Prepare** *(Home)* | Lasts 1 hour. After the duration is over:  Increments stage by 1. |
| 1 | **Fly**  *(Home)* | Lasts (1 to 18) hour(s). After the duration is over:  Increases the aircraft’s flying hours by flight duration time.  Increases the stage by 1.  Checks if the aircraft has to go through maintenance. |
| 2 | **Prepare** *(Destination)* | Same as Stage No.0 |
| 3 | **Fly**  *(Destination)* | Same as Stage No.1 |
| 4 | **NewCycle** | Calculates plane stage (explained further in the documentation). |

Table 3.2 – Aircraft Stage Description

When adding hours, it is important to first check if the aircraft is in maintenance. If it is, then the hours first need to be added to maintenance (and ground crew needs to be assigned), and only afterwards the leftover hours can be used to advance the current stage.

#### NewCycle Procedure

Instead of using a for loop for every plane, NewCycle provides the database with a calculation tool to understand in what stage the aircraft after adding a certain number of hours to the system.

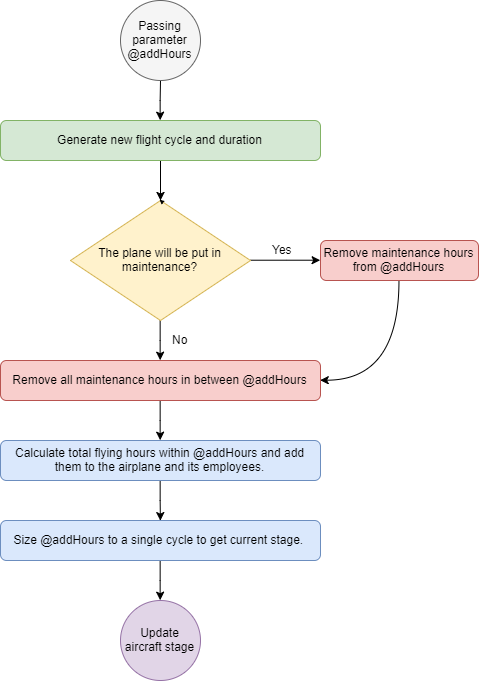


Figure 3.2 – Newcycle flowchart

After passing @addHours, the procedure generates a new flight duration and calculates the new flight cycle. “rand” is equal to a random number between 1 and 18.

The hours at which the upcoming maintenance will start are calculated like so:

If the plane will go through maintenance, the maintenance hours are substracted. In-between maintenance hours are removed from @addHours. Since after adding @addHours to the plane might put it into another maintenance, LastMaintenanceBegin is also calculated and taken into account.

Actual hours spend flying are calculated by only getting the portion of hours spent in the Fly stage.

After adding @addHours the plane might also be in the middle of traveling between destinations, that time must also be added.

To understand what stage the plane is in, @addHours must be sized into a single cycle.

That will set the @addHours within the range [0; modulo(@addHours, FlightDuration)], which is exactly one cycle. Afterwards, the plane’s stage can be easily decided using an IF statement, checking if @addHours is within the range of each stage.

## XML Data

When generating random employee names, a bad approach would be to hard-code every single name into a method the Employee class. A more elegant solution towards importing a list of strings into a List<string> variable can be introduced by using LINQ.

using System.Xml.Linq;

This provides tools for the programmer to import data from an XML file into a suitable list variable. In this project, the Maintenance History and the Employee names are imported using XML files.

## Error-Handling

All methods an functions that are prone to errors are encapsulated using try{} catch{}, and whenever an error occurs it gets translated in catch{} using its error code. A new exception is created using a message that the user can understand and further thrown to the triggering web page, which then takes further action to inform the user and take any necessary action to prevent the error from happening again.

## Database Tables



Figure 3.3 – Database Tables

## Database Load

# Testing

Testing the design system is very important to ensure reliability in a real situation. Various elements will be tested, focusing on the ones that are more likely to be problematic.

|  |  |  |  |
| --- | --- | --- | --- |
| Object | Action | Expected Outcome | Actual Outcome |
| Login button | Click | Display login page | Display login page |
| Registration Number Filter | Enter wrong data | Display error message | Display error message “Wrong format” |
| Registration Number Filter | Enter wrong data, bypassing page validation. | Display error message | Display special error message. |
| Simulation Menu  -> Add Records | Clicking on it multiple times without waiting for server. | Display error message | Display error message “Server is busy” |

# Evaluation and Future Technologies

# Conclusion

# References

Cleland, D. I., & Ireland, L. (2002). ‘*Project management: Strategic design and implementation’* (4th ed., Vol 1). New York: McGraw-Hill.

To credit References from code and website

To credi stackoverflow

To ref coursework

To unprivate github