United Airlines Management

Angel Petrov (266489)
Chunhui Liu (273452)
Ziad Akram Bathish (273442)

SEP6 Autumn/2020

Supervisors:

Richard Brooks
Jakob Knop Rasmussen
Laurits Ivar Anesen

Software Technology Engineering
6th semester
18.12.2020

Table of contents:

Introduction

The project has been made to solve the United Airplanes Association problem, that few managers need to get airport-related data at any time and anywhere with an internet connection from their devices' web browser, to stay fully informed about all airports, airlines, planes and weather data.
The project must give the few selected managers a complete and detailed overview of these data.
The company did not want to host the application on their own servers, therefore there is a need for a cloud provider that can help to deploy our system so we used both Microsoft Azure and Google Cloud.

Requirements

a. Functional Requirements:

1. Total number of flights per month
2. The total number of flights per month from the three origins in one plot. We want to be able to see:
   - Frequency
   - Frequency stacked
   - Stacked percentage
3. The top-10 destinations and how many flights were made to these
   - For these 10 destinations, make a visualization of the number of flights from the three origins to the top-10 destination.
4. The mean airtime of each of the origins in a table
5. How many weather observations there are for the origins in a table
6. For each of the three origins, all-temperature attributes (i.e. attributes involving a temperature unit) in degree Celsius (i.e. you need to convert from Fahrenheit to Celsius).
7. The temperature (in Celsius) at JFK.
8. The daily mean temperature (in Celsius) at JFK.
9. The daily mean temperature (in Celsius) for each origin in the same plot.
10. Mean departure and arrival delay for each origin in a table.
11. The manufacturers that have more than 200 planes.
12. The number of flights each manufacturer with more than 200 planes is responsible for.
13. The number of planes in each Airbus Model.

b. Non-Functional Requirements:

1. The application has a responsive view so it can be readable on tablet's or other devices' web browsers.
2. The application has a cloud provider.
3. The application is a high uptime response.
4. The application is available from anywhere with an internet connection and a web browser.

Analysis

The system implementation makes use of a four-layered architecture consisting of the presentation layer (PL), business logic layer (BLL), data access layer (DAL) and finally, the database which holds relevant data.
Each layer provides modularity of the system. When a layer is to be modified by the responsible development team, maintenance of the application is easier because of the low coupling between layers(***Answers. 2020***). The architecture also provides a firm ground for testing individual parts as functionality is separated into the individual layers. Layering also provides a good basis for testing individual packages that are part of a layer, whereby it is not necessary to call packages that are part of other layers in order for a test to pass.
The flow of data in the system will involve the user making a request for the required page from the presentation layer and receiving a response from the upper layer. Sending data from the presentation layer and saving it up into a higher layer is not a feature that is required.

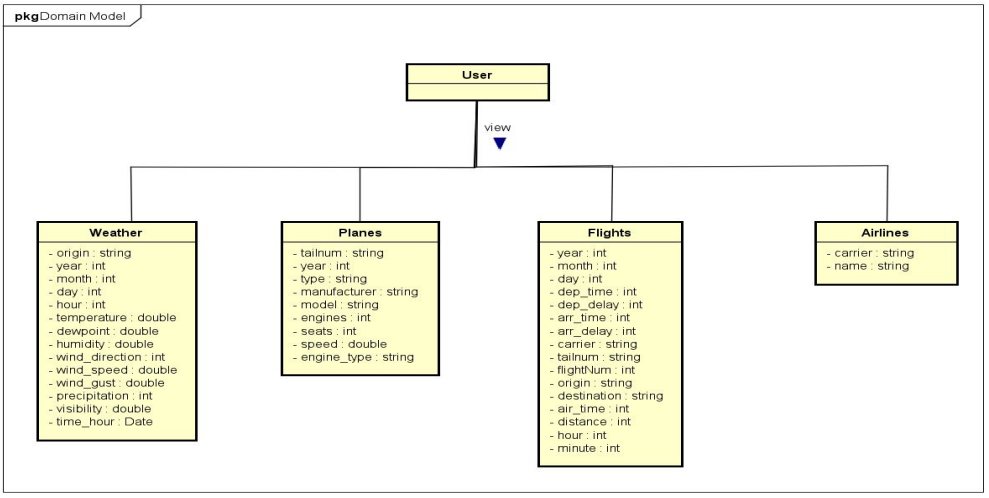

Figure 1: Domain Model

The data that is provided to the development team at the start of the project is in the form of CSV files which are to be cleansed of any invalid entries that could cause a bad import of the data into other layers of the n-tier architecture layer.

Design

The design was important to have a better idea about the system and how all layers and connection between layers will be made.
MVC pattern was the best choice to implement this kind of projects because it will help with having high cohesion, low coupling and ease of modification while it divides the application into three kinds of components: models, views and controllers.
Here it was important to discuss the class diagram and how the process can be done in the best way to make it easier in the implementation stage.
A simple class diagram was made to have a better look at the system before starting to implement it. The data will be loaded in the model from the Data Access Layer. All business logic will be done in the controller, which will send the processed data to the views (Presentation Layer) to be shown to the end-user.

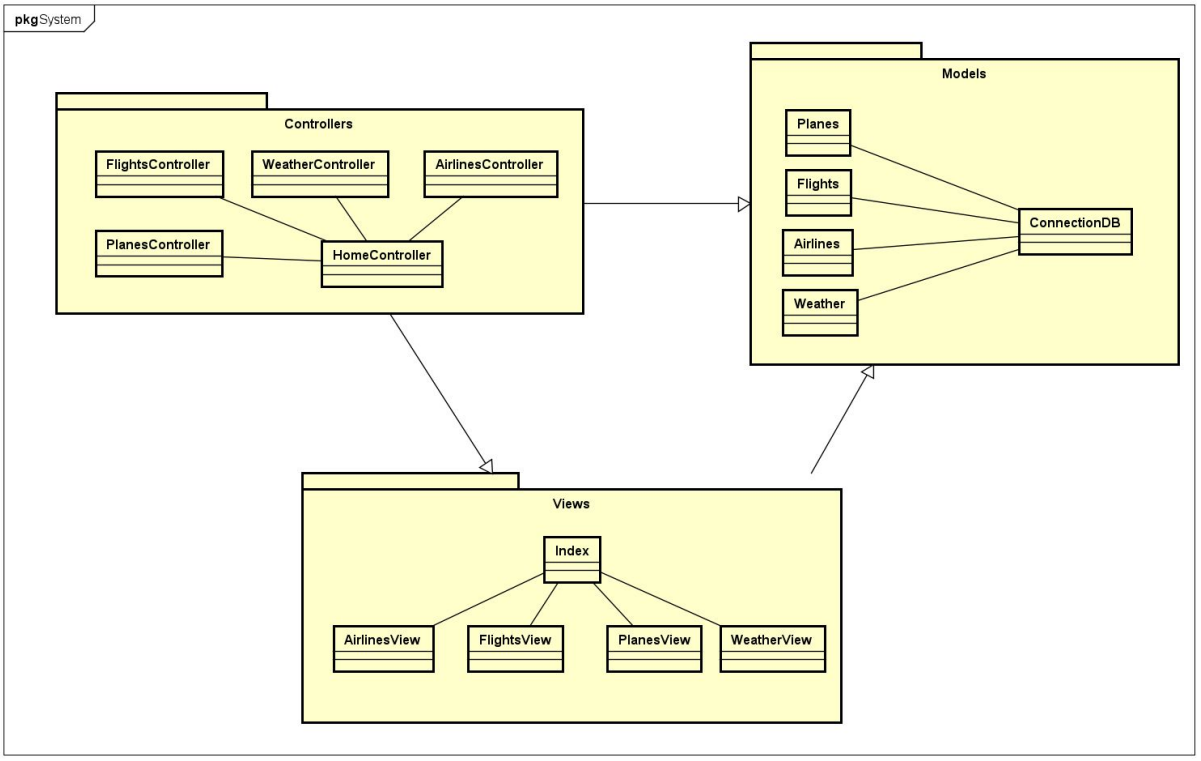

Figure 2: Simplified Class Diagram

- For a better understanding of the provided data, an ER diagram was made to give an overview of how tables are related and where to get the right data while working on implementation.

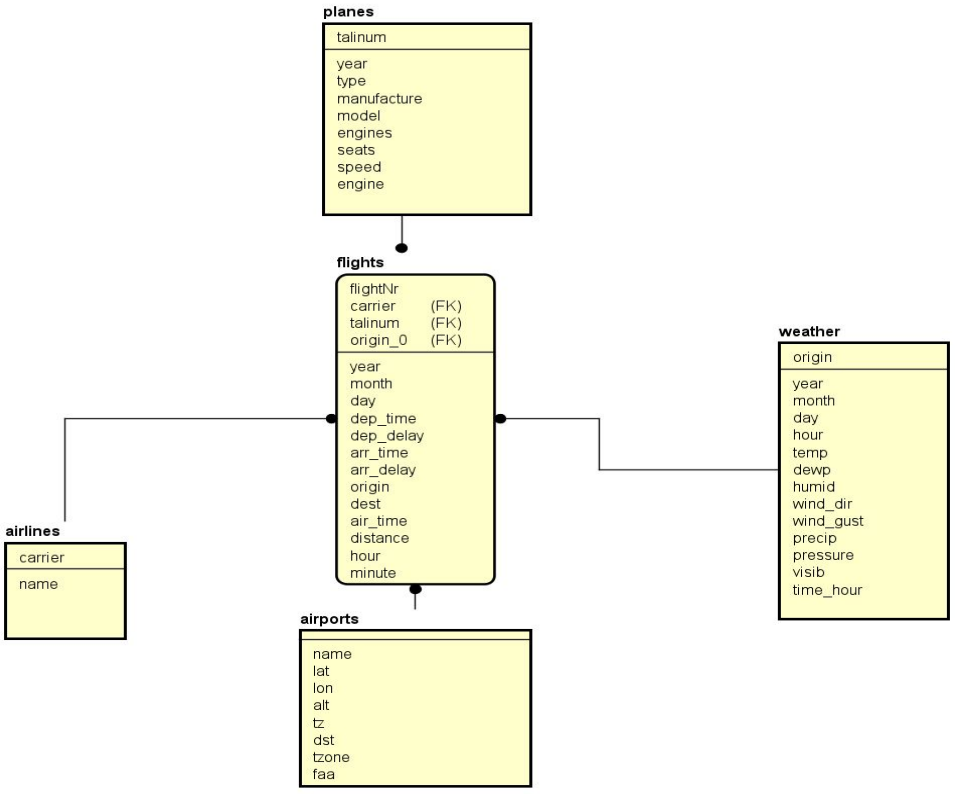

Figure 3: ER Diagram

Cloud Computing

More and more applications are migrating to the cloud, and the trend is increasing. Traditional applications are becoming more and more complex, and enterprises have to buy all kinds of hardware and software to cope with the increasing demand, which is often huge and unaffordable. The advent of cloud computing solves this problem. Once applications are deployed in the cloud, they can use common hardware and enjoy services for a fee.

Cloud computing features include virtualization, distribution, and high availability, as well as on-demand services and billing based on usage. Cloud computing uses virtualization technology so that users do not have to focus on the hardware entity. In addition, using cloud services can effectively reduce security risks with the help of more professional security teams.

In terms of performance, cloud services typically run on a global network of secure data centres, reducing application network latency and increasing the economy of scale compared to a single enterprise data centre.

On the reliability side, cloud computing can simplify data backup, disaster recovery, and business continuity at a lower cost.

In this project, cloud computing technology is used. The development team uses Google Cloud for facilitating the data into a database and we use Azure Web Deploy to deploy our applications which in term, accesses the data from the database that is hosted on Google Cloud. Both services are serverless.

Comparing Google Cloud with Microsoft Azure which was used in our system, Google Cloud has industry-leading deep learning and artificial intelligence, machine learning and data analysis tools. It is known for offering discounts and is in a position to grow aggressively. Azure Cloud strives to interact with data centres, and the hybrid cloud is a huge advantage, as well as a strong advantage in Saas.

DevOps

DevOps was applied in this project to have one team that are all responsible for the development, testing and deployment.
It usually helps with having continuous delivery of value, the team didn't have to wait until finishing the whole project before releasing.

The process involves:

1. Planning: the project was started by planning what tasks will be there, using the GitHub Project tool and making an automated kanban board template.
2. Development: implementing tasks on the board and applying unit testing for each.
3. Release: using Github actions and circle CI to test the pushed code before merging with the master branch to make sure that there are no conflicts with the current version. In this stage usually, the project is ready to be released to the end-user.

Implementation

The database is facilitated by Google Cloud which acts as a Cloud provider. In Google Cloud, a new project has been created with billing enabled from the given free trial period with a debit card as the development team was unable to utilize their 50 dollars of given credit which was marked as expired in the Overview section for Billing for an unknown reason.

First, we need to create the database instance by expanding the list under the hamburger button on the left side of the main overview page and selecting **SQL** from the dropdown list. Then, we click on **Create Instance** and select MySQL. From there, we provide all necessary details and select the closest region to where we are located - in our case Europe-west1(Belgium)

Moving forward with setting up the project, we need to navigate to the **API Library** section and enable the **Compute Engine API** for our project. This API would allow us to run a virtual machine in the cloud.

Next thing that we needed to do is set up a static IP address. A static IP address is an external IP address that is reserved for the project. To reserve a static IP address, we must find the section **VPC network** and select the sub-option **External IP addresses**. From there, we would provide our specified name, IP protocol to use, and region, which, in our case, are IPv4 and Europe-west1(Belgium) for closer proximity to the remote server. If we were to lookup this address using whatismyipaddress.com, we can see that our settings are in fact correctly applied.

To manage data, we used HeidiSQL Server, a free and open-source database management tool for MySQL and its branches that allows us to easily import, edit, view, and modify data. We created the Cloud SQL service by setting the database ID and password and then getting the remote database address so that we could process the data using HeidiSQL.
In order to successfully import data in CSV format, we changed the original data file to CSV - UTF8 format.

At the same time, in order to process Na value data, we first check the data. If this column data contains Na value, we first set the data type to VARCHAR in the database, and then replace Na value with NULL with SQL statement, then we can set this column data to the required data type.

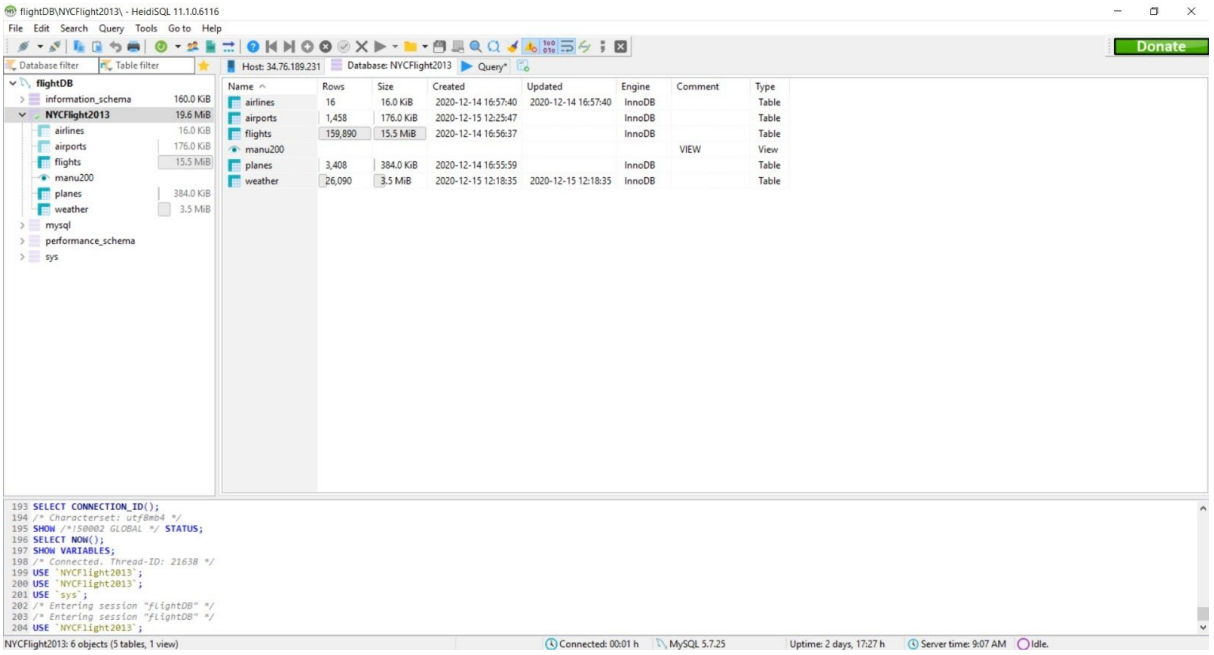

Figure 4: Overview of NYCFlight2013

For visualizing the data, we used JavaScript and C#. We used Echarts and CanvasJS as tools. When we first tried to use Echarts for charting, we found that it was very simple to make tables, but more complicated to drawbar charts and other diagrams.
Later we looked at other ways and found that CanvasJS was a convenient tool and did not require the installation of additional plug-ins.
In the web page design, we analyzed the data required by each requirement and found that most of the data required by the requirement only needed to be operated in one table, so we made corresponding webpage pages for each table.
To get the data and display it in the diagram, we use the following method in the Controller and view.Here is how the code is used and what is implemented in the web page:

```
public IActionResult Index()
{
    var airlines = GetAllAirlines();
    ViewData["airlines"] = airlines;
    return View("~/Views/Home/Airline.cshtml");
}
```

```
@{
    ViewData["Title"] = "Home Page";
    @model NYCFlights2013.Models.Airlines
    var airlines = (List<Airlines>)ViewData["airlines"];
}
```

Figure 5: Code for connecting controller and view

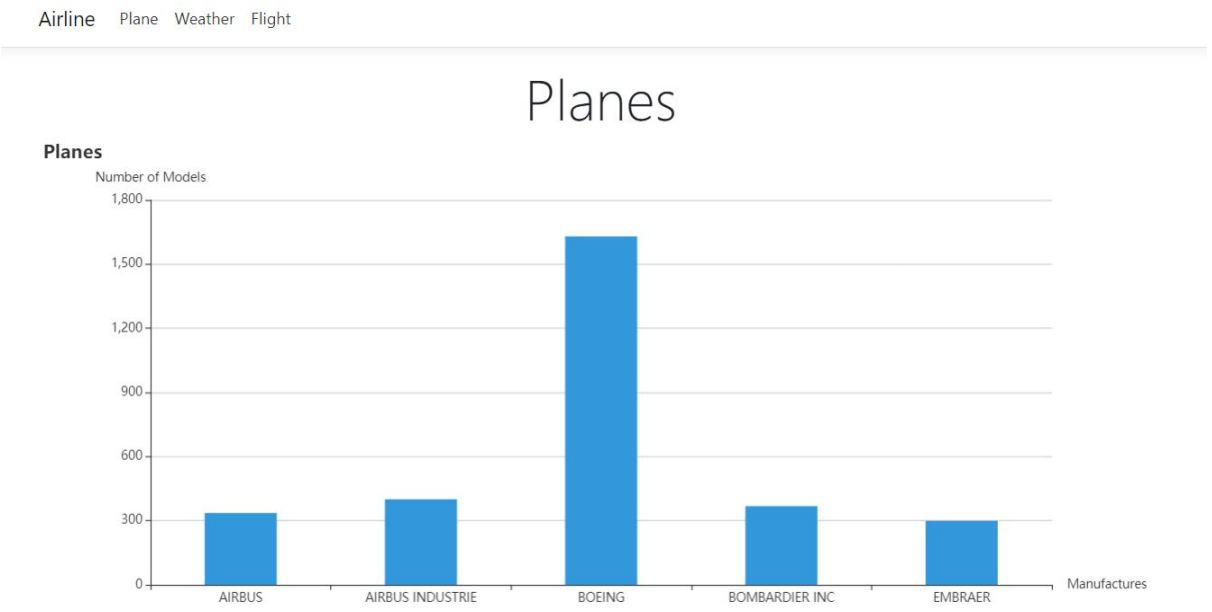

Figure 6: The manufacturers that have more than 200 planes

Manufacture	Number of Models
AIRBUS	336
AIRBUS INDUSTRIE	400
BOEING	1630
BOMBARDIER INC	368
EMBRAER	299

Figure 7: Number of models

Manufacture	Number of Flights
AIRBUS	11676
AIRBUS INDUSTRIE	36244
BOEING	67623
EMBRAER	4856

Figure 8: Number of flights

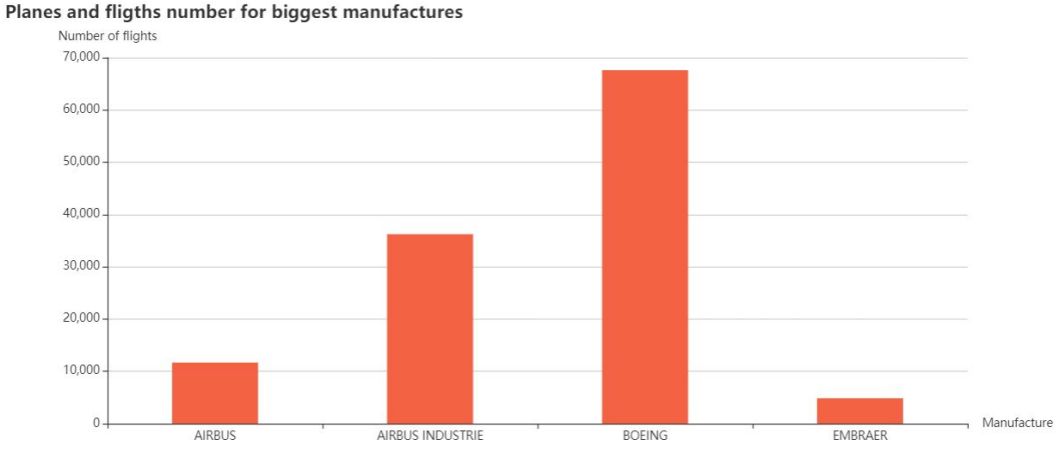

Figure 9: Planes and flights number for biggest manufactures

The number of planes in each Airbus Model:

model	number of planes
AIRBUS	336
AIRBUS INDUSTRIE	400

Figure 10: The number of planes of each Airbus Model

Weather Observations table

Origin	Weather Observations
EWR	8708
JFK	8711
LGA	8711

Figure 11: Weather Observations table

Weather Observation Chart

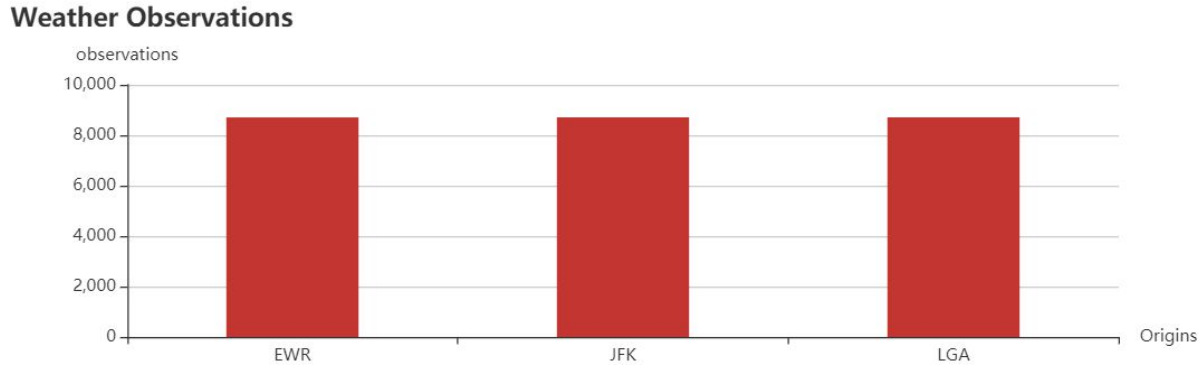

Figure 12:Weather Observation Chart

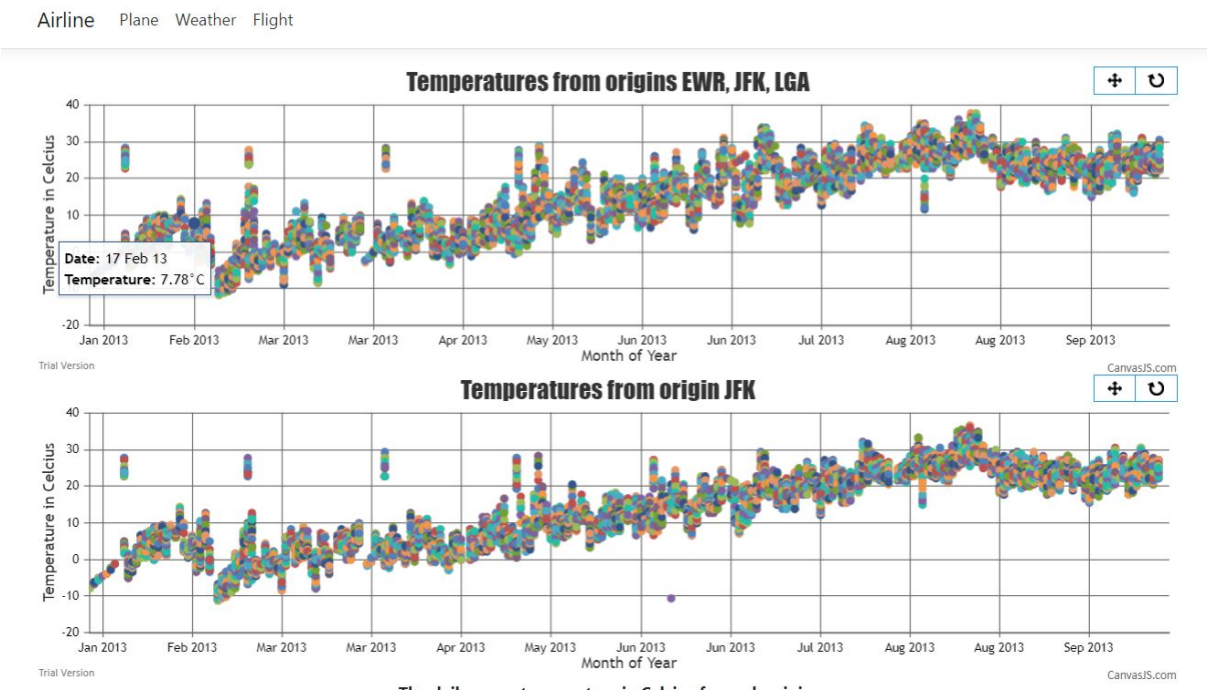

Figure 13: For each of the three origins, all temperature attributes

The daily mean temperature in Celcius for each origin:

Origin	Mean Temp	Day	Month	Year
EWR	3.45	1	1	2013
EWR	-1.76	2	1	2013
EWR	-1.41	3	1	2013
EWR	0.83	4	1	2013
EWR	2.62	5	1	2013
EWR	3.33	6	1	2013
EWR	5.16	7	1	2013
EWR	3.52	8	1	2013
EWR	4.88	9	1	2013
EWR	6.78	10	1	2013
EWR	5.19	11	1	2013
EWR	7.59	12	1	2013
EWR	7.71	13	1	2013

Figure 14: The daily mean temperature (in Celsius) for each origin

The daily mean temperature in Celcius at JFK:

Origin	Mean Temp	Day	Month	Year
JFK	3.66	1	1	2013
JFK	-1.92	2	1	2013
JFK	-1.23	3	1	2013
JFK	1.13	4	1	2013
JFK	2.71	5	1	2013
JFK	3.06	6	1	2013
JFK	5.49	7	1	2013
JFK	3.7	8	1	2013
JFK	4.88	9	1	2013
JFK	7.25	10	1	2013
JFK	4.51	11	1	2013
JFK	7.57	12	1	2013
JFK	6.69	13	1	2013

Figure 15: The daily mean temperature (in Celsius) at JFK

Figure 16: The top-10 destinations and Total number of flights per month

mean airtime of each of the origins:

origin	meantime
EWR	196.6121
JFK	235.3726
LGA	125.6053

Mean departure and arrival delay for each origin:

origin	delayD	delayA
EWR	11.6126	3.3734
JFK	8.7551	-0.0746
LGA	7.8196	2.3666

Figure 17: The mean airtime of each of the origins and Mean departure and arrival delay for each origin

Number of flights from the three origins to the top-10 destination:

origin	dest	top10 flights #
EWR	SFO	4344
EWR	LAX	4129
EWR	IAH	3973
EWR	ORD	3822
EWR	BOS	3342
EWR	ATL	3256

Figure 18: The number of flights from the three origins to the top-10 destination

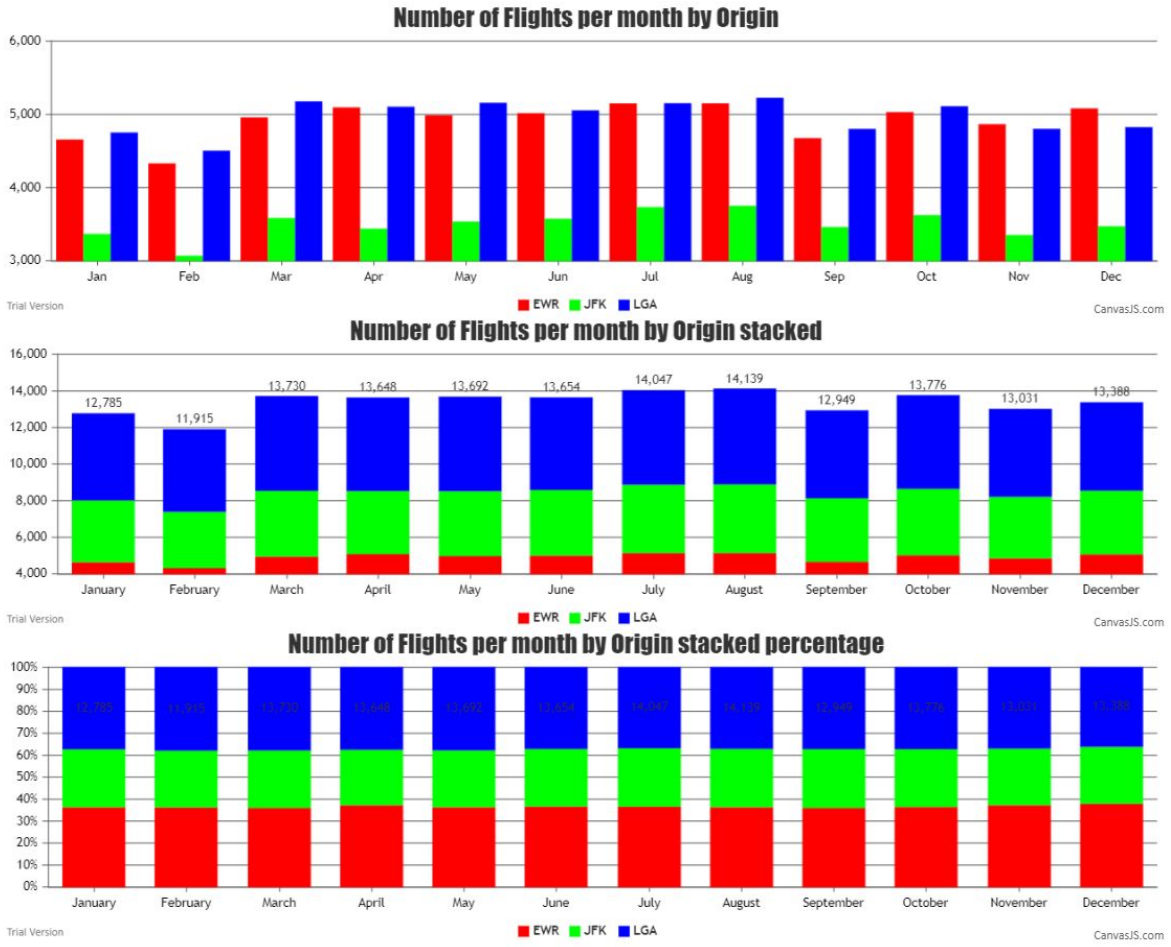

Figure 19: Total number of flights per month from the three origins in one plot

Testing

The testing section involves demonstrating that the developed system conforms to the requirements. As the project is to be tested utilizing .NET Core, the selected child project (named TestNYCFlights2013) that will hold the testing classes is based on NUnit, a unit-testing framework which is compatible with all .NET languages.

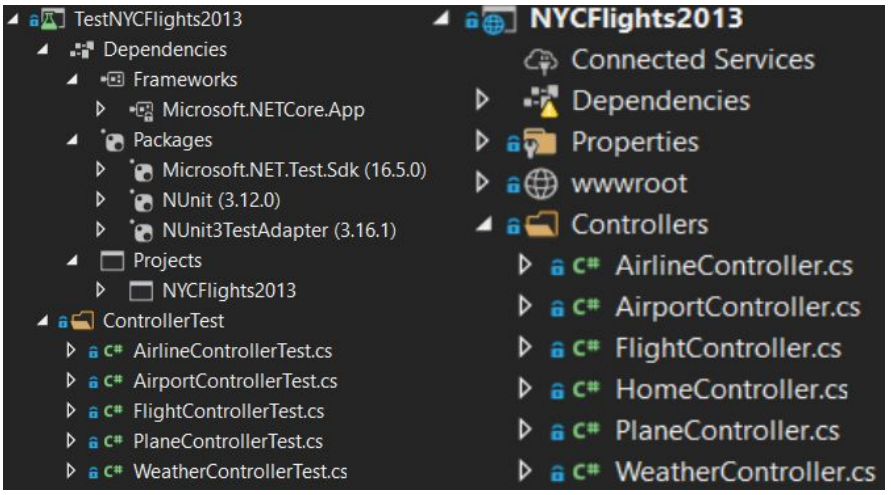

Figure 19: Setup of NUnit project. The classes with "Test" at the end of their name create an instance of their counterpart in the main project.

White-Box testing

With the above project structuring in mind, the development team started making a test class that instantiates each corresponding controller inside. After creating an instance of the controller, the methods to be tested are accessible.

The following table describes the methods and a description of how the operation of testing takes place:

Test #	Responsible class	Invoking Method()	Target class	Invoked method()	Outcome
1	AirlineControllerTest	TestOneGetAllAirlines	AirlineController	GetAllAirlines	Pass
2	FlightControllerTest	TestGetNumOfFlightsM	FlightController	GetNumOfFlightsM	Pass
3	FlightControllerTest	TestGetMeantimeO	FlightController	GetMeantimeO	Pass
4	FlightControllerTest	TestGetNumOfFlightsO	FlightController	GetNumOfFlightsO	Pass
5	FlightControllerTest	TestGetTopdestM	FlightController	GetTopdest	Pass
6	FlightControllerTest	TestGetMeandelay	FlightController	GetMeandelay	Pass
7	PlaneControllerTest	TestGetNumOfManufacture	PlaneController	GetNumberOfManufactures	Pass
8	PlaneControllerTest	TestGetNumOfManuFlights	PlaneController	GetManFlights	Pass
9	PlaneControllerTest	TestGetNumOfAirbus	PlaneController	GetPlanesNumM	Pass
10	WeatherControllerTest	TestGetWeatherTemp	WeatherController	GetWeatherTemp	Pass
11	WeatherControllerTest	TestGetWeatherTempDate	WeatherController	GetWeatherTempDate	Pass
12	WeatherControllerTest	TestGetWeatherTempAtJFK	WeatherController	GetWeatherTempAtJFK	Pass

13	WeatherControllerTest	TestGetDailyMeanTemp	WeatherController	GetDailyMeanTemp	Pass
14	WeatherControllerTest	TestGetDailyMeanTempJFK	WeatherController	GetDailyMeanTempJFK	Pass

Table 1: NUnit White-Box testing results

Black Box Testing

Black box testing was performed on the front-end of the system that is responsible for presenting the View. The view contains a total of four pages: Airline, Plane, Weather and Flight. All four pages are able to give a 200 response code when accessed.

Page Airline is able to display the data about the carrier and the name of the carrier into a table. All data on this page is visualized without any issues.

Page Plane displays the number of produced airplanes for a given manufacturer that are above 200. This data is visualized with an ECharts chart and a table underneath the chart with two data columns for manufacturer and the number of flights. We are able to hover over each chart and see its value. Underneath these two items are two more - another EChart and a table that present data about the number of flights for the biggest manufacturers. Finally, at the very bottom, a table for the number of planes in each Airbus model is shown which contains two columns, one for model and the second being for the number of planes. All data on this page is visualized without any issues.

Page Weather displays a scatter point chart from three origins at the top and a scatter point chart for JFK origin both utilizing CanvasJS evaluation edition. We are able to zoom in on the data by selecting the area that we wish to visualize. Both charts have extract data successfully but have been limited to display data up to 6000 entries where the maximum number of entries from the database is 26130. The reason for this is an issue with displaying data past 6000 entries, whereby, the data after this point may display with a wrong date value and therefore, be placed on a wrong position in the chart. This issue does not manifest from the database nor the used SQL query but rather from how the chart is set up to visualize data. The data is not categorized by origin in the first chart. Despite their limitations, these two charts are still considered functional. Underneath them is a table that displays the daily mean temperature in Celsius for each origin and another table for JFK origin both of which display data correctly.

The last but not least is the page Flight. There are two tables that display their data correctly and five CanvasJS charts underneath one-another - Top 10 destinations, Number of Flights per month, Number of Flights per month by origin, origin stacked and origin stacked percentage. The last three tables have their data for each origin sorted with a different colour in the chart. All data on this page is displayed correctly.

Sources of Information

Explain the benefits of layered architecture? - Answers
https://www.answers.com/Q/Explain_the_benefits_of_layered_architecture
Accessed: 2020-12-16

Appendix

- CloudUserGuide.pdf
- NYCFlights2013Code.zip
- IPADSep6.pdf
- SystemVideoDemo.zip
- CloudDeploymentDashboards.zip
- GitHubLink.pdf

Application Link

https://nycflights20132020121416085 6.azurewebsites.net/