

Cameron Stark

2337995

starkc1@my.erau.edu

Software Engineering

Summer 2019

American Airlines

Reliability Engineering Department

July 14, 2019

**Table of Contents**

1. American Airlines
   1. Engineering Departments
2. Projects
   1. ETOPS (Extended Operations) Data Access Application
      1. Background & Customer
      2. Tech Stack
      3. What I did
   2. 787 WAP (Wireless Access Point) Connection Dashboard
      1. Background & Customer
      2. Tech Stack
      3. What I did
   3. Aircraft Event Time Series Dashboard
      1. Background & Customer
      2. Tech Stack
      3. What I did
   4. Interiors Dashboard
      1. Background & Customer
      2. Tech Stack
      3. What I did
   5. Data Access Applications (Shiny Apps)
      1. Background & Customer
      2. Tech Stack
      3. What I did
   6. ACMS Database Data Access Application
      1. Background & Customer
      2. Tech Stack
      3. What I did
   7. MEL-MMEL Reader
      1. Background & Customer
      2. Tech Stack
      3. What I did
3. Conclusion
   1. Company Experience
   2. Future with Company

**Section 1: American Airlines**

*Section 1.a: Engineering Departments*

The structure of the American Airlines Engineering Department, is split up into three major sections being Technical Operations (TechOps), MOC, and Base Support. TechOps consists of over 20 individual departments all working towards keeping the planes flying and arriving/departing on schedule and getting passengers from one airport to the next airport.

Within TechOps there is Fleet Engineering, which contains Airbus Fleets, Boeing Fleets, Embraer/MD80 Fleets, and Reliability. The specific Fleet Engineering teams work on fixing issues on plans, converting FAA/OEM work orders, by writing EO/EAs to tell the AMTs how/what to fix or replace on the plane per the orders of Boeing/Airbus or the FAA. The Reliability Department has two segments the data analysis and the data visualization side. The data analysis team takes all the Delay and Cancel Data, Log page Data, Deferral Data, Flight Load Data, AOS (Aircraft Out of Service) Data and determines the root cause of the issues, then passes the information on to the respected Fleet Engineering teams so that they can work to mitigate the issue. The data visualization team builds dashboards and reports for the data analysis team and all of TechOps, the tools very in stack from Tableau, R Shiny, Angular, Alteryx workflows, and python report scripts.

**Section 2: Projects**

*Section 2.a: ETOPS (Extended Operations) Data Access Applications*

Section 2.a.i: Background & Customer

ETOPS which means Extended Operations is the designation or rating a commercial aircraft can receive after going through several certification flights for the FAA. Once an aircraft attains ETOPS, the aircraft gets a value associated with it usually in the form of minutes which means the aircraft can fly out over the ocean but must always be within the minutes they have from an airport. For example, the American Airlines 777-200 have an ETOPS 330 Mins, which means when out over the ocean they must always be within 330 mins of an airport, in the event of an engine failure or another emergency.

The ETOPS Department requested 6 automated Alteryx Reports and 6 Data Access Applications, so that they could best monitor and track any discrepancy events in the ETOPS fleet which is most of the Flagship Fleet which deal with international flights and try to reduce the amounts of Delays/Cancellations or AOS events.

The Department requested that the 6 reports be emailed to multiple team members so that every morning they can analyze the prior days data, and the 6 Data Access Apps deliver the same report but allow each member to filter and get a specific data set for what that individual is working on.

Section 2.a.ii: Tech Stack

The Tech Stack for this project is broken down into two sections reports and applications. With them both sharing a common Teradata SQL database to store the data for the departments use.

The Reports are built using a program called Alteryx which is a drag and drop visualized SQL querying tool, that has built in capabilities of auto running on a schedule and sending out emailed reports.

The Data Access Applications, known internally as Shiny Apps are websites built using R and the web development package for R called Shiny, which allows the R code to be hosted and implemented as a website. The Shiny Apps allow for quick access to the data and the ability to filter the data. The Shiny Apps, act as a cleaner front end for querying the database, for those who don’t know SQL, because the applications build the query based on the filters selecting which are just appending the “AND” statements to the end of “SELECT” query.

Section 2.a.iii: What I Did

My part of this project was creating the 6 Data Access Apps and working with the customers to get the filters they want and the format of the associated download file from the application. I started the project by understanding the datasets I was working with and then I formatted all of the base SQL queries for each of the apps so that the columns are in the requested order and have the correct more readable names. Then I implemented the different filters they wanted, then get the filters building the query and successfully returning the formatted and filtered data. Once all that is working, I delivered the apps to the ETOPS department and waited for the new set of changes, the implement them and return to customer, and repeat process until it is how they want it.

*Section 2.b: 787 WAP (Wireless Access Point) Connection Dashboard*

Section 2.b.i: Background & Customer

The Boeing 787-8 and 787-9 that American Airlines has, are different than other aircraft, except for the 737 Max series, in that the flight data and updates to the aircraft can be done over the via WAP (Wireless Access Points) that are positioned at every gate the aircraft parks at DFW (Dallas-Fort Worth) Airport, LAX (Los Angeles) Airport, and ORD (Chicago O’Hare) Airport. The Connection downloads all the flight data from the previous flight and uploads to new information on the upcoming flight.

The 787 Fleet Engineering Team and the Avionics/Aircraft Software Department have requested that the Reliability department make a dashboard so that the connection quality and completion status for each aircraft and WAP. The Dashboard would have 4 datasets being, Last Aircraft Connection, Historical Aircraft Connections, Last WAP Connection and Historical WAP connection.

The two “Last” connection datasets will have two views each on the dashboard a Snapshot view which shows all Aircrafts/WAPs at a glance their with last connection time, average RSSI (Received Signal Strength Indicator) for the connection which shows the average strength of the signal during the connection, average SNR (Signal Noise Ratio) which is the ratio of interference and signal during the connection, the bytes transferred, and the aircraft/gate info for each connection. The other view is an Analysis view which has a plot of the connections and a quick access color coordination to determine problem Aircraft/WAPs

The Historical Connection views will contain all 787 aircraft and WAP connections with the option to filter the amount of days back from the current day, and show a list of the previous connections for the respected aircraft or WAP along with the RSSI and SNR values for the connection to allow for analysis of if an aircraft or WAP is having connection issues or if past issues have been resolved.

The final view on the dashboard is a map view, which shows each WAP location on the map and upon hover/click of an icon displays the last connection and the values associated with it.

Section 2.b.ii: Tech Stack

This dashboard is built using Angular 7 as the framework and many Angular Material components for the design, with making use of libraries such as Lodash for array/object manipulation, moment js for date control, and map box for the map functionality. The application is built extensively using a MVC design with the views working off a routing system, so that there is no page reload just view load, and the views being controlled individually with all data being sent and received via a service that is only instantiated when called