

Cameron Stark

2337995

starkc1@my.erau.edu

B.S. 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. 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

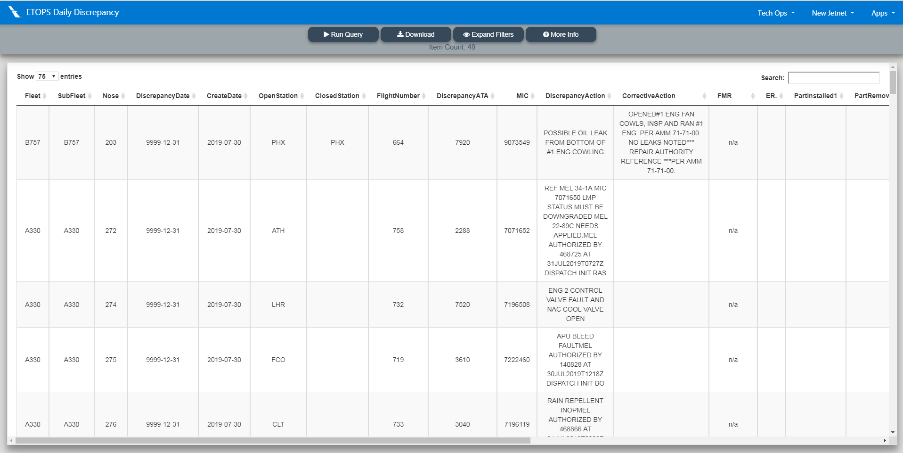
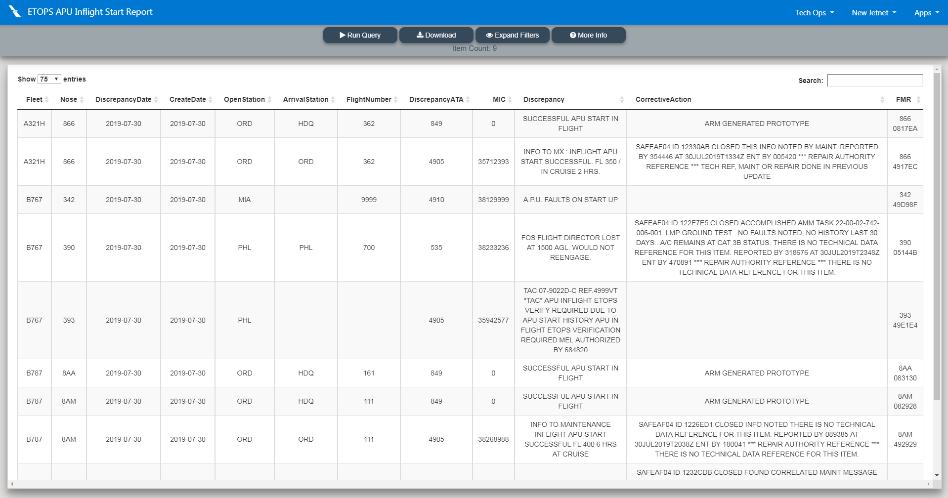
ETOPS which means Extended Operations is the designation or rating a commercial aircraft can receive after going through several certification flights for the FAA. Once an aircraft attains ETOPS, the aircraft gets a value associated with it usually in the form of minutes which means the aircraft can 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 of 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.

Figure ETOPS Daily Discrepancy Report

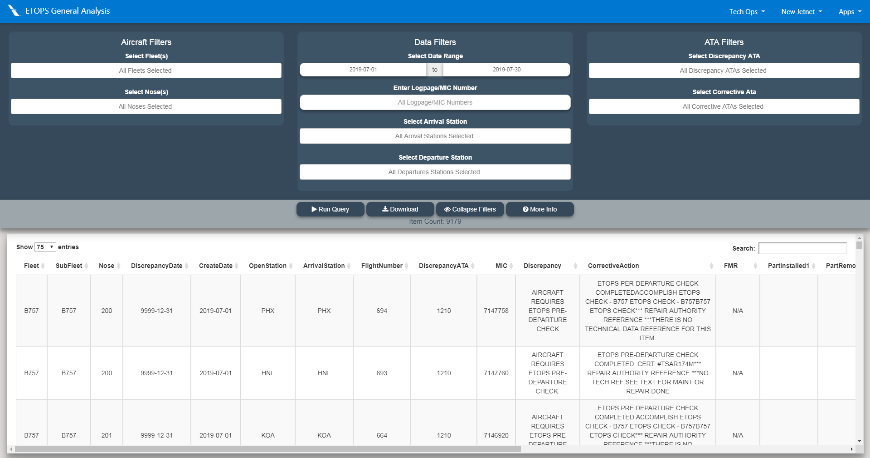
The ETOPS Department requested 6 automated Alteryx Reports and 6 Data Access Applications, so that they could best monitor and track any discrepancy events in the ETOPS fleet which is most of the Flagship Fleet which deal with international flights and try to reduce the amounts of Delays/Cancellations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

Figure ETOPS APU Inflight Start Report

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.

Figure ETOPS General Analysis

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

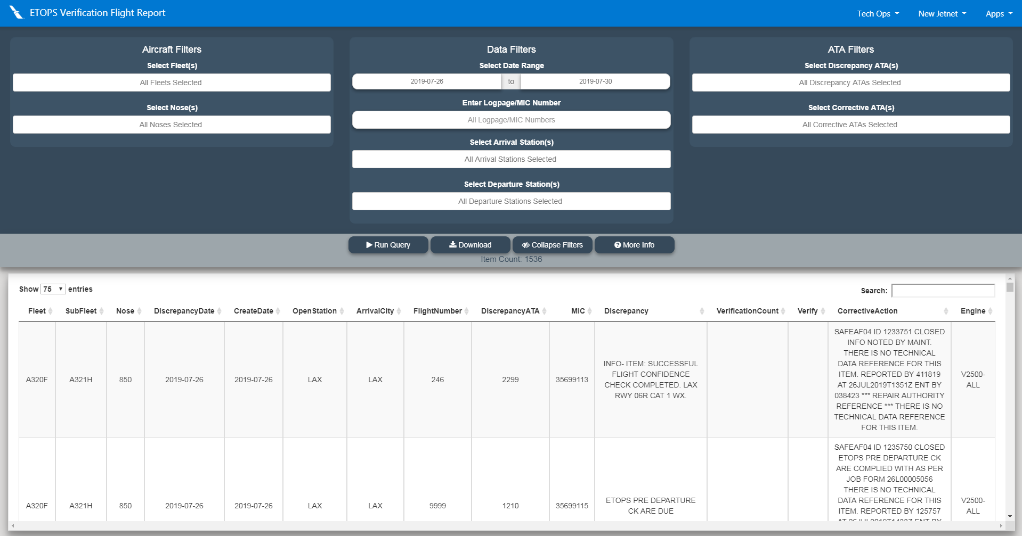
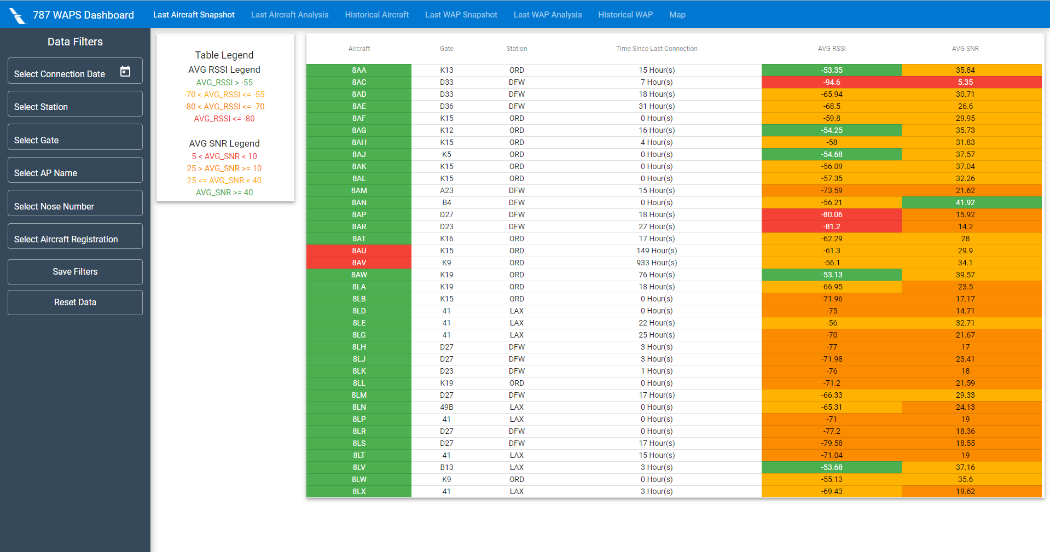
My part of this project was creating the 6 Data Access Apps and working with the customers to get the filters they want and the format of the associated download file from the application. I started the project by understanding the datasets I was working with and then I formatted all of the base SQL 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. Through this process I worked with another Reliability engineer who worked on the backend and built the ETOPS table necessary for the applications.

Figure ETOPS Verification Flight

*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.

Figure Last Aircraft Snapshot

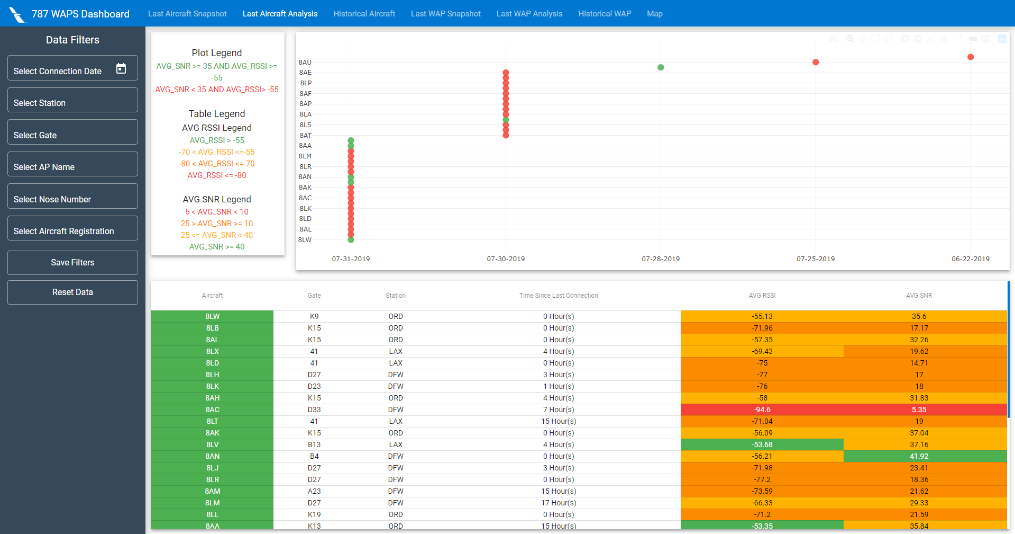
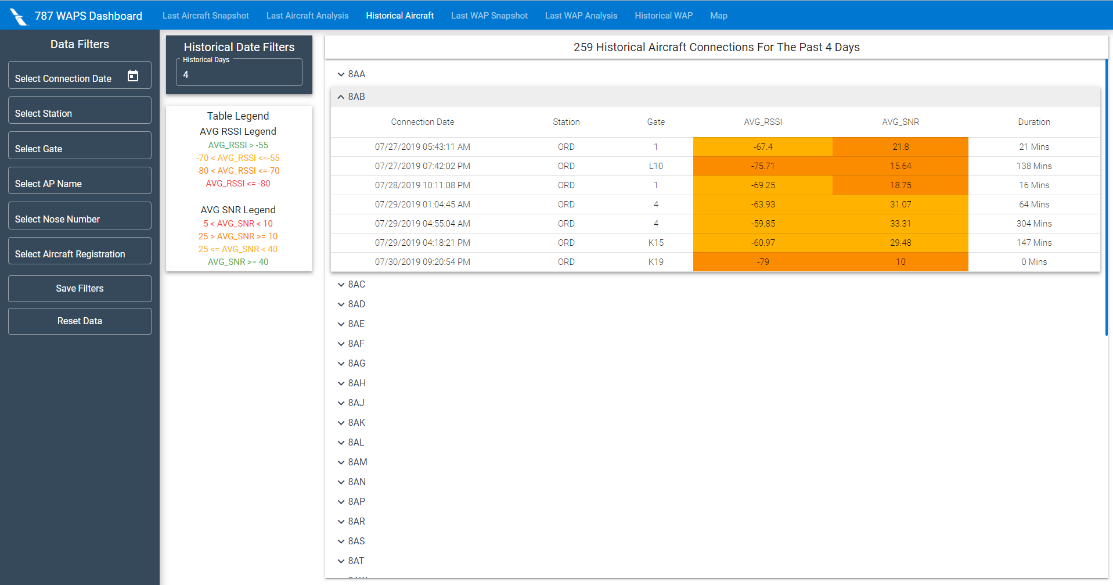
The two “Last” connection datasets will have two views each on the dashboard a Snapshot view which shows all Aircrafts/WAPs at a glance with their last connection time, average RSSI (Received Signal Strength Indicator) for the connection which shows the average strength of the signal during the connection, 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, along with the table of data as in the snapshot view.

Figure Last Aircraft Analysis

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.

Figure Historical Aircraft Connections

Section 2.b.ii: Tech Stack

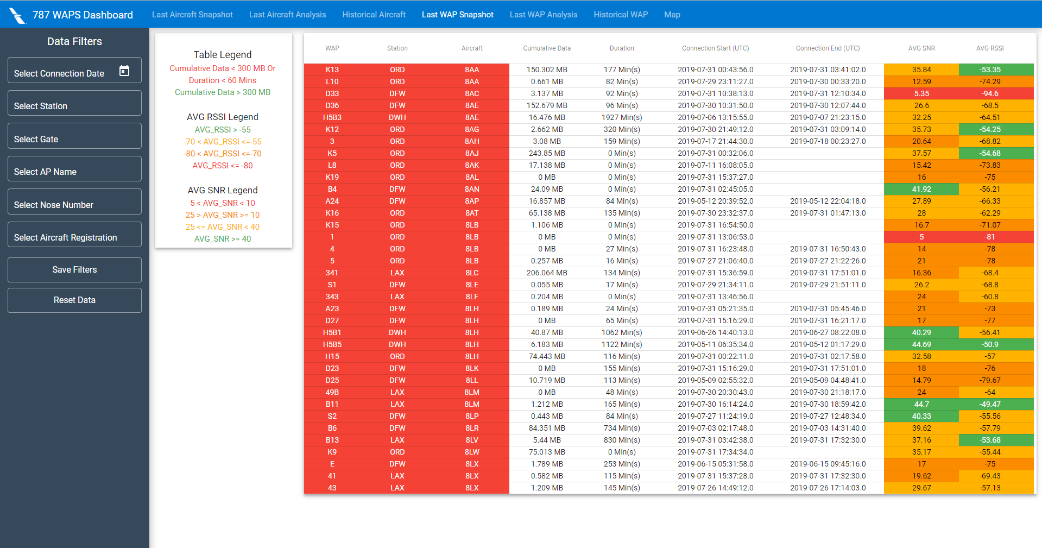
 This dashboard is built using Angular 7 as the framework and many Angular Material components for the design, with making use of libraries such as Lodash for array/object manipulation, moment js for date control, and map box for the map functionality. The application is built extensively using a MVC 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.

Figure Last WAP Connection

To facilitate communication between the angular application and the Teradata database, a NodeJS web api was built, with endpoints for the last aircraft data, last WAP data, historical aircraft data and the historical WAP data. The node api is hosted on a server that access by making web http requests from the angular application.

To host the angular app, it is put on to the internal company SharePoint site to control access to the site to those on the enterprise network or have an American airlines employee login.

Section 2.b.iii: What I Did

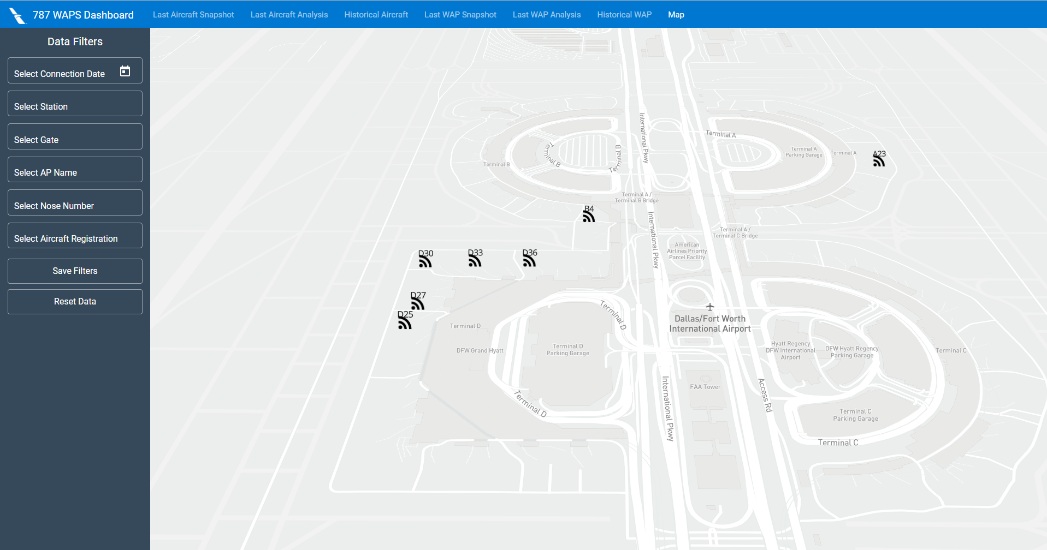
 My part of this project was to meet with and get the requirements/functionality of the application from the customers. From those meetings and emails, I worked on the design of the application to facilitate multiple different view types and the ability to view the snapshot views from a mobile device for those in the field. From there I worked with an existing api built for the four tables, and determined the data coming back from the end points and created the data service for the application to send the request and return the results into object modeled after the response fields for each of the end points. Once I got the back end of the application complete and working properly, I started development of the user interface which had a main view that contained the sidebar with filters that be accessible to all other views/components via a listener that passes the data through the data service. In the sidebar are the filters of fleet, subfleet and aircraft which all have the ability to waterfall into each other, as well as the filters of station, WAP, and gate. I connected each Last connection view, Historical view and map view to the end points created in the NodeJS api. Upon completion and working fully, I delivered to the customers and awaited for the response with things they liked and any additions/changes that should be made.

Figure WAP Map

*Section 2.c: Aircraft Event Time Series Dashboard*

Section 2.c.i: Background & Customer

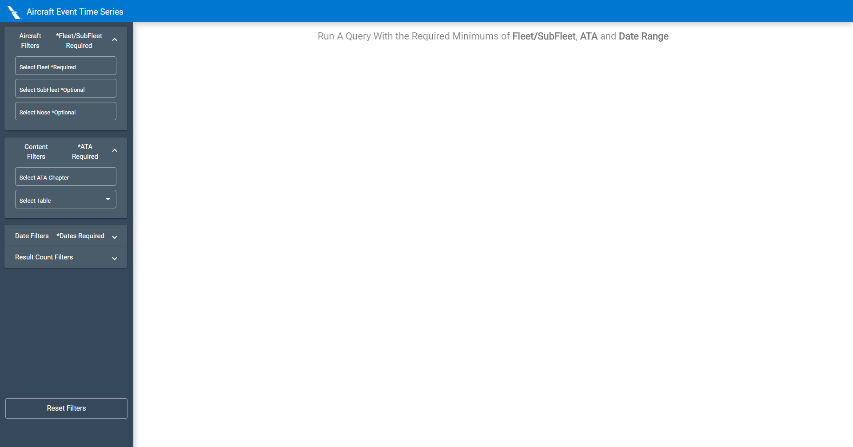
 Currently if an engineer wants to trend across multiple data sets for a certain aircraft or data set, they would have pull data from each of the individual data sets and then analyze each of them to find the trend of the an aircraft’s event, such as a series of delays and then a part removal can show either a drastic decrease in delays or an increase which can be used to show that fix was successful or not. This tool at the request of fleet engineering and MOC (maintenance operations center) would hit all of the necessary tables and provide a detail trending plot for a specific aircraft and the events that have occurred over the selected date range. Each of the events on the plot are clickable to show more detailed information in a table below the plot and if the same event type occurs multiple times on the same date, they will be represented. The customers also requested to have a chronic aircraft detail tab that listed the most chronic aircraft in a table for each of the datasets for easy quick access to problem aircraft in the fleet.

Figure Landing View

Section 2.c.ii: Tech Stack

 The dashboard is built as an Angular 7 application, so that the large amounts of data being pulled could effectively and efficiently requested and processed asynchronously with the use of Typescript and Lodash to sort through and format the data without slowing down or freezing the web page, to ensure the best customer user experience. The dashboard is designed as a single page web application, with different views built off of routing to facilitate each of the different views that will be necessary for the application.

Figure Query Result

The api for the application to connection to the Teradata database is built using NodeJS and express, with endpoints for each of the filters in the sidebar and endpoints with parameters for each of the datasets that will be pulled.

Section 2.c.iii: What I Did

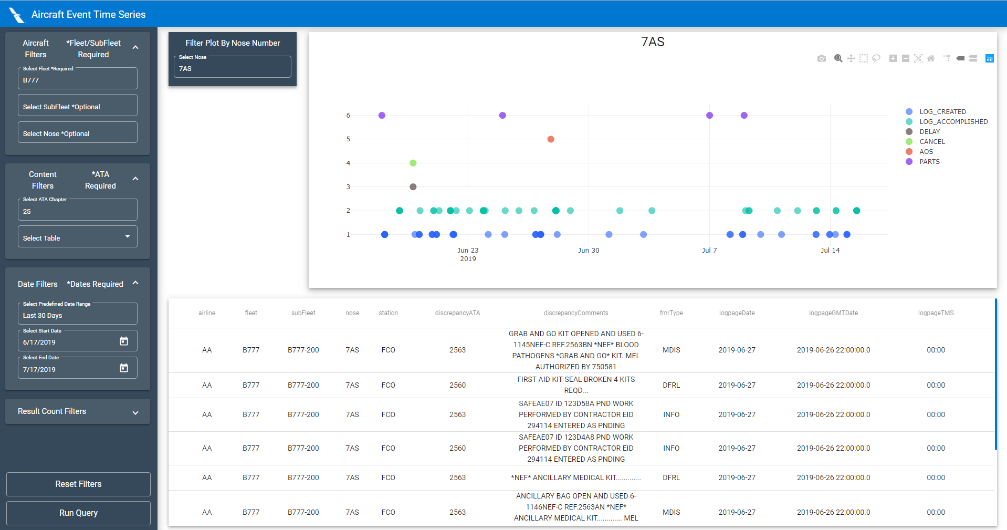
 My part of the project was to build the whole angular application from design stage to the current testing and production stage. With the mock up design being done in a tool called figma that allows me to get the design the way I want before I even touch the code and then I built the framework of the user interface. Once that was complete I built the models for the data and the data service to communicate to the API. From there I knew what endpoints and format of the data, so I started on the NodeJS API, and got that effectively working and returning the requested data. After that I connected the two and tested and fixed any data format issues and rendering issues, then passed it on to the customers and continued the iterative fix/modify and return process. The next process for the application, which will be continued with my remaining time here and after I leave is pitching the idea to IT to make it a full enterprise application.

Figure Plot Clicked View

*Section 2.d: Interiors Dashboard*

Section 2.d.i: Background & Customers

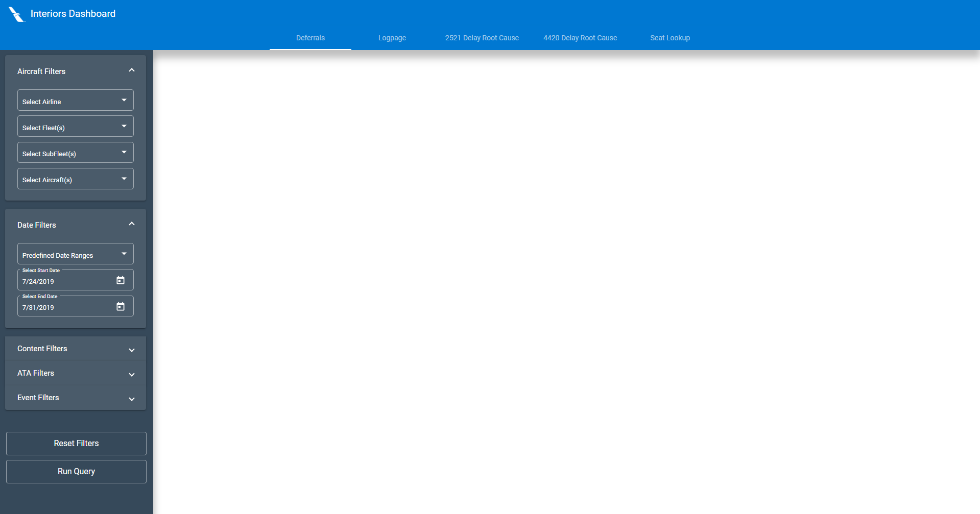
 The Interiors Dashboard was a request by some engineers in the Reliability Department, Fleet Engineering and Interiors Department, who wanted to be able to pull Deferral Data, Logpage Data and Root Cause data, but wanted to only see the ATAs (which are a set of codes that define parts of the system, such as 36 is air conditioners) that pertained to interiors which would air conditioners, and cabin oriented ATAs. The customers requested 5 views, each being the Interiors Deferral View, Interiors Logpage View, 2521 Root Cause, 4420 Root Cause, and Seat Lookup. The First two views are just straight data pulls and filters, with the ability to download as csv for further manipulation. The next three views are data input and modification views that allow the user to add and change the data, based on root cause analysis and an aircraft getting a new seat type/configuration.

Figure Deferrals Landing

Section 2.d.ii: Tech Stack

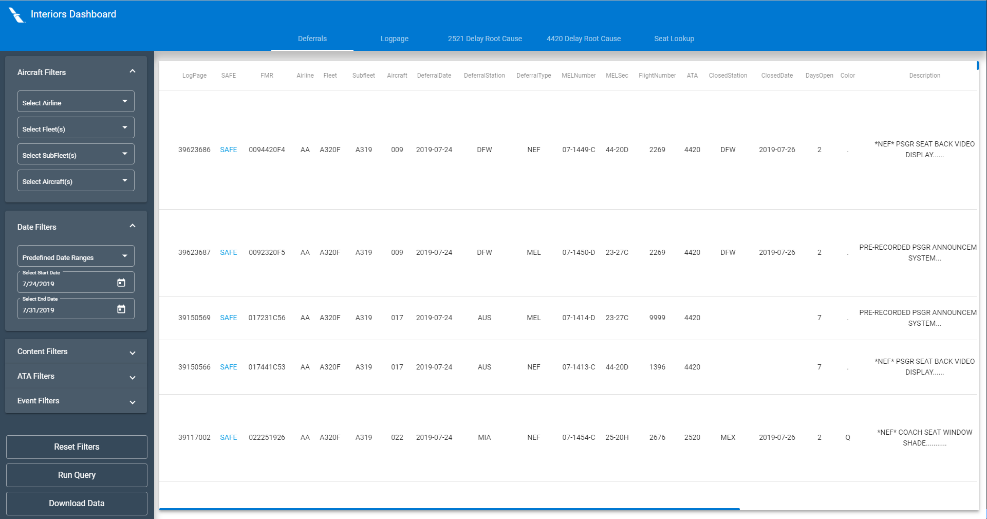
 The dashboard and its five views are built using Angular 7, Lodash, MomentJS, and Angular Material for the front end and data manipulation to display on the front-end or send to the database. The application sends and receives to the Teradata Database, using a NodeJS api with endpoints for each view different set of filters, full data, and the write capabilities of the last three views. The app is built using a tab system so that each view is own page inside the larger application since each set of filters and datasets are unique for each view.

Figure Deferrals Search Results

Section 2.d.iii: What I Did

I built the application starting with a mock up to see how I wanted to deal with the different views, and then determined the filters and layout of the columns in the data with the customers. I built the user interface and the data service for the connection to the NodeJS api. I also built the NodeJS api has endpoints for the filters for each database and the full data with parameters, the last three views will have endpoints for updating and adding new data items.

*Section 2.e: Data Access Applications (Shiny Apps)*

Section 2.e.i: Background & Customer

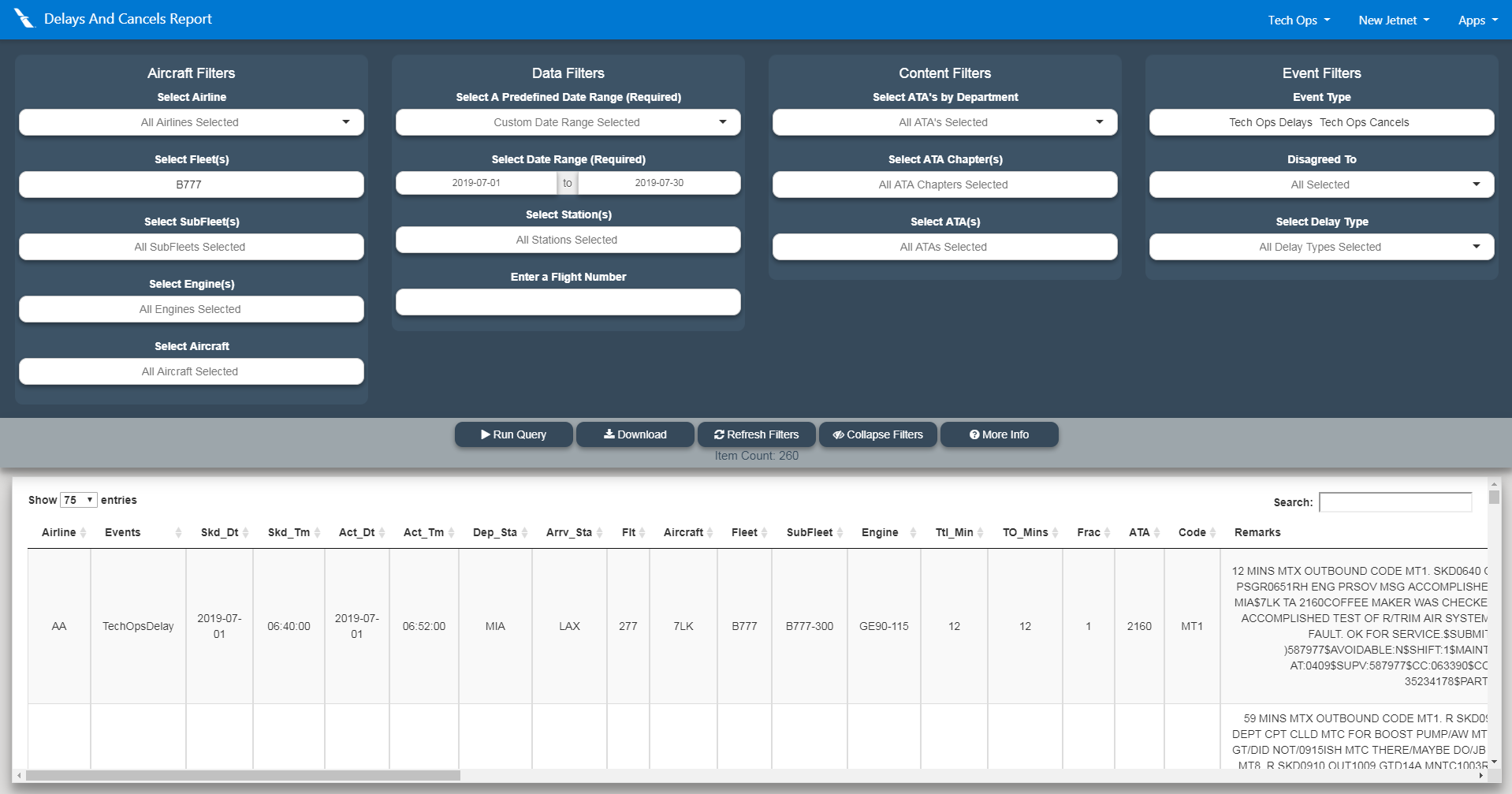
 The Shiny Apps as they are referred to around the company, are web apps built using the language R, and a framework for it called Shiny that allows the R code to be converted and act like a website, so that you get the efficiency and speed of R when dealing with data and the portability of a website. The purpose of the Shiny Apps, are at the request of all the engineers around Tech Ops who have to pull large amounts of data every morning, and the current process is using Cognos reports which are old and slow, for example pulling the last three months of Delay/Cancel data for all fleets is around 20,000 items, which in the cognos report would take upwards of 20 mins for each person doing the request, as for the Shiny Apps that I worked I can pull the last two years of Delay/Cancel data for all fleets which is around 130,000 and takes at most 30 seconds to pull all of the data. There are cognos reports for all of the major datasets are most commonly needed for the engineers, like the Delay/Cancel, Deferrals, Logpages, AOS (Aircraft Out of Service), LUS (Legacy US Airways) Part Removal, LAA (Legacy American Airlines) Part Removal and many others.

Figure Delay And Cancel Report

Section 2.e.ii: Tech Stack

The Shiny Apps are built using the previously explained R library called Shiny, and are hosted on a R Server, running on a linux computer that the department has, with the final website being linked into an iframe on each respected website URL on the company SharePoint site, so that access to the data can be controlled to the those only on the enterprise network. The applications make use of the publicly available Teradata drivers, to build, request and receive data from Teradata by sending SQL transactions to the database similar to the a database explorer.

Section 2.e.iii: What I Did

I took the requested data that the engineers want, found the necessary table or tables in the case of a join being needed, and determined what filters are possible or needed for the dataset, and then started to build the filters for the application and then worked on the portion of building the sql query based on which filters were populated, with the date fields always being required filters as to limit the amount of results being returned, upon getting everything working I sent the application to the customers and they would reply back with filters that should be added or removed, and any change of orders for the returned columns.

*Section 2.f: MEL-MMEL Reader*

Section 2.f.i: Background & Customer

The MEL-MMEL Reader was requested by Reliability Engineering Managers, who want an organized list of the MEL (Minimum Equipment List), the description for it, and its associated MMEL, but the only current list of them for each ATA and each Flight is a bunch of PDF files. The managers requested a tool to read the pdfs, parse the necessary information and export as an excel file. The MEL is important because if an item causing a delay is on the list, the maintenance crew can read the steps associated with it and most of the time can allow a deferral on the aircraft allowing it to continue on the route and not interrupt passengers and the daily operation of the airline.

Section 2.f.ii: Tech Stack

The MEL-MMEL reader is built fully in NodeJS, using multiple libraries such as pdfreader, fs, request-promise, request-promise-native and json2csv. The application has 3 main functions, download pdf which is done through the pdf url, the rdf reader which is through the pdfreader library and then the final step of saving to excel file.

Section 2.f.iii: What I Did

I spent the majority of this project parsing and trying to understand the format the PDF files so that I could find the points where the MEL, Description and MMEL are located and how to collect the three of them so that the are associated with each other and dealing with some cases where the description would have next line characters. Once I figured out how to go about collecting those, moved on to figuring out how to pull the pdfs from the webpage and determining the url structure of the PDFs as to have the program build the URL, make the request, download, parse and then export as csv. Once that was figured out, I moved to exporting as csv with the json2csv library.

**Section 3: Conclusion**

*Section 3.a: Company Experience*

My experience with the company has been an amazing one, at other internships I was viewed as an “Intern” which meant I didn’t have control in the my projects and not much credit for the project I work on, where as at American Airlines, I was the lead on most projects and the POC (Point of contact) for the Shiny Apps, I presented to the VP of Engineering, I was able to demo one of my projects to AMTs at multiple hangars and stations.

With all of my projects I was given the tasks and the customers, then my managers knew I had the knowledge and skillset to figure out what needed to be done, give them updates and ultimately deliver the product. This provided the sense of actually being a part of the team and department rather then a student just there for the work experience. During meetings and presentations on new TODS (TechOps Data Source) meetings, my managers would make a point of saying “The reliability Co-Op built this site” and that if any problems would occur, I would be the source and receiver of that information.

Lastly the great thing about American Airlines is the travel benefits, which are just as good as they sound, and the managers know that the co-ops like to travel so as long as you put in your hours and do good work they will be flexible if a flight fills up and you get stuck somewhere (which will happen).

Section 3.b: Future With Company

I could very easily see my self coming back to the company, and my managers have been trying to convince me to either graduate earlier or don’t leave for school because they like the work I do and don’t want it to stop when I leave for school. Another point in favor of returning to American Airlines as a full-time employee is the Pay and Benefits for a starting engineer is very good, and the pay raises are competitive, especially in Texas where it is relatively cheap to live.