



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
  - a. Engineering Departments
2. Projects
  - a. ETOPS (Extended Operations) Data Access Application
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
  - b. 787 WAP (Wireless Access Point) Connection Dashboard
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
  - c. Aircraft Event Time Series Dashboard
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
  - d. Interiors Dashboard
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
  - e. Data Access Applications (Shiny Apps)
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
  - f. MEL-MMEL Reader
    - i. Background & Customer
    - ii. Tech Stack
    - iii. What I did
3. Conclusion
  - a. Company Experience
  - b. 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

ETOPS Daily Discrepancy

Run QueryDownloadExport FilterMain Menu

Page Count: 12

Show 75 + entries

Flight	SubFlight	Route	DiscrepancyDate	CrashDate	Operation	ClassOfFlight	FlightNumber	DiscrepancyATA	MOC	DiscrepancyAction	CorrectiveAction	FAIR	ER	Participated?	PartNumber
0707	0707	203	9999-12-31	2019-07-30	PIK	PIK	654	7520	SET0549	DISCREPANCY IN THE KAN COULDS, WSP AND RAN BY END, FOR KAN TO DO NO LEAKS NOTED TOWARD AUTHORITY RE: LCHNCL, "MURKIN" 751702		NO			
A330	A330	272	9999-12-31	2019-07-30	ATH		756	2256	7071632	NOT BEL, IN A BNC TESTED LAMP SEVERAL WSP DOWNWASDED WEL TOWARD AUTHORITY APPROVED WEL SEVERAL BULGARYPOTENCE EXTRACTION NOT RAN		NO			
A330	A330	274	9999-12-31	2019-07-30	LHR		752	7520	7190535	END 2 CONTROL WSP PAULT AND NAC COOL WEL OPEN		NO			
A330	A330	275	9999-12-31	2019-07-30	PCO		715	3610	7222480	AND BEL, L PAULTWEL AUTHORISED BY TOWARD AT NAC BEL, TOWARD DISPENSIC NOT DO TOWARD BEL, FIRST TOWARD AUTHORISED BY TOWARD AT		NO			
A330	A330	276	9999-12-31	2019-07-30	SLT		752	3540	7190119			NO			

Figure 1 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.

[illegible]

Figure 2 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.

[illegible]

### Figure 3 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

Fleet	SubFleet	Nose	DiscrepancyDate	CreateDate	Open Station	ArrivalCity	FlightNumber	DiscrepancyATA	MIC	Discrepancy	VerificationCount	Verify	Engine
A100P	A101H	800	2019-07-26	2019-07-26	LAX	LAX	246	229B	30609113	INFO ITEM SUCCESSFUL FLIGHT CONFERENCE CHECK COMPLETED LAX NAVY DER CAT 1 BOX			V200D ALL
A100P	A101H	800	2019-07-26	2019-07-26	LAX	LAX	5999	121D	30609115	ETOPS PNL CLIMATUNG OK AND DUE			V200D ALL

Figure 4 ETOPS Verification Flight

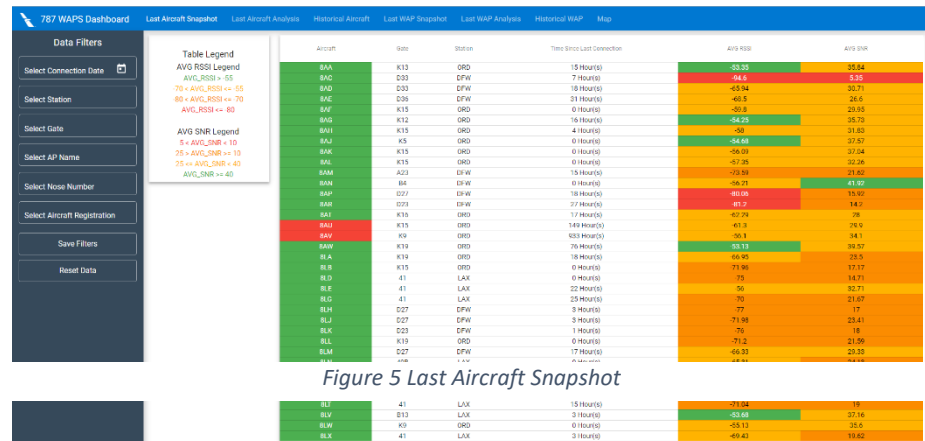
that the columns are in the requested order and have the correct more readable names. Then I implemented the different filters they wanted, then get the filters building the query and successfully returning the formatted and filtered data. Once all that is working, I delivered the apps to the ETOPS department 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.

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

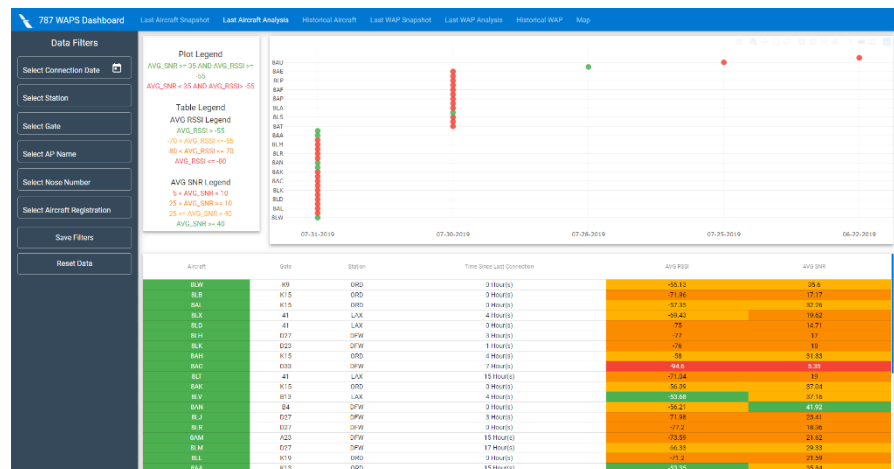
#### Section 2.b.i: Background & Customer

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

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.



(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 plot of the connections. Aircraft/WAPs, along with



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.

**787 WAPS Dashboard** | Last Aircraft Snapshot | Last Aircraft Analysis | Historical Aircraft | Last WAP Snapshot | Last WAP Analysis | Historical WAP | Map

**Data Filters**

- Select Connection Date
- Select Station
- Select Gate
- Select AP Name
- Select Noise Number
- Select Aircraft Registration
- Save Filters
- Reset Data

**Historical Date Filters**

4

**Table Legend**

**AVG RSSI Legend**

- 70 < AVG\_RSSI <= 55
- 80 < AVG\_RSSI <= 70
- AVG\_RSSI <= 80

**AVG SNR Legend**

- 5 < AVG\_SNR <= 10
- 25 < AVG\_SNR <= 10
- 25 < AVG\_SNR <= 40

**259 Historical Aircraft Connections For The Past 4 Days**

Connection Date	Station	Gate	AVG_RSSI	AVG_SNR	Duration
07/27/2019 05:43:11 AM	ORD	1	-47.4	21.8	21 Mins
07/27/2019 07:42:02 PM	ORD	L10	-25.71	15.64	138 Mins
07/28/2019 10:11:06 PM	ORD	1	-49.25	18.75	18 Mins
07/29/2019 01:04:45 AM	ORD	4	-43.93	31.07	64 Mins
07/29/2019 04:55:04 AM	ORD	4	53.83	33.31	304 Mins
07/29/2019 04:18:21 PM	ORD	K15	-43.97	23.48	147 Mins
07/30/2019 09:20:54 PM	ORD	K19	-79	10	0 Mins

Figure 7 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.

**787 WAPS Dashboard** | Last Aircraft Snapshot | Last Aircraft Analysis | Historical Aircraft | Last WAP Snapshot | Last WAP Analysis | Historical WAP | Map

**Data Filters**

- Select Connection Date
- Select Station
- Select Gate
- Select AP Name
- Select Noise Number
- Select Aircraft Registration
- Save Filters
- Reset Data

**Table Legend**

**Cumulative Data <= 300 MB Or**

- Question <= 60 Mins
- Cumulative Data <= 300 MB

**AVG RSSI Legend**

- 70 < AVG\_RSSI <= 55
- 80 < AVG\_RSSI <= 70
- AVG\_RSSI <= 80

**AVG SNR Legend**

- 5 < AVG\_SNR <= 10
- 25 < AVG\_SNR <= 10
- 25 < AVG\_SNR <= 40

**WAP**

WAP	Station	Aircraft	Cumulative Data	Duration	Connection Start (UTC)	Connection End (UTC)	AVG_SNR	AVG_RSSI
K15	ORD	BAA	150,302 MB	177 Min(s)	2019-07-31 00:43:56.0	2019-07-31 00:43:02.0	39.44	33.10
K15	ORD	BAA	3,887 MB	80 Min(s)	2019-07-29 21:17:27.0	2019-07-29 20:39:29.0	32.96	27.75
K15	ORD	BAA	5,137 MB	92 Min(s)	2019-07-31 10:38:13.0	2019-07-31 12:10:34.0	5.19	-44.6
K15	ORD	BAA	152,679 MB	98 Min(s)	2019-07-30 13:31:50.0	2019-07-30 12:07:44.0	26.5	-49.5
K15	ORD	BAA	15,479 MB	1927 Min(s)	2019-07-30 13:35:53.0	2019-07-30 21:23:19.0	22.28	-44.55
K15	ORD	BAA	2,862 MB	320 Min(s)	2019-07-30 21:40:12.0	2019-07-30 20:09:14.0	35.73	-44.29
K15	ORD	BAA	3,308 MB	159 Min(s)	2019-07-31 21:44:20.0	2019-07-31 00:23:27.0	38.04	-40.02
K15	ORD	BAA	240,818 MB	5 Min(s)	2019-07-31 00:23:26.0		27.57	-54.19
K15	ORD	BAA	17,138 MB	5 Min(s)	2019-07-31 16:50:05.0		11.42	-73.82
K15	ORD	BAA	9 MB	5 Min(s)	2019-07-31 15:37:27.0		1.9	-79
K15	ORD	BAA	34,259 MB	5 Min(s)	2019-07-31 02:45:05.0		44.92	-50.21
K15	ORD	BAA	16,957 MB	84 Min(s)	2019-05-12 20:39:52.0	2019-05-12 22:04:19.0	27.09	-46.33
K15	ORD	BAA	65,138 MB	135 Min(s)	2019-07-30 23:52:37.0	2019-07-31 01:47:13.0	39	-42.23
K15	ORD	BAA	1,100 MB	5 Min(s)	2019-07-31 16:50:05.0		16.7	-71.07
K15	ORD	BAA	9 MB	5 Min(s)	2019-07-31 13:50:58.0		5	-61
K15	ORD	BAA	9 MB	27 Min(s)	2019-07-31 16:25:48.0	2019-07-31 16:25:48.0	14	-78
K15	ORD	BAA	5,257 MB	16 Min(s)	2019-07-31 21:26:40.0	2019-07-31 21:22:29.0	21	-78
K15	ORD	BAA	206,064 MB	184 Min(s)	2019-07-31 19:36:59.0	2019-07-31 17:51:03.0	16.96	-49.4
K15	ORD	BAA	3,003 MB	17 Min(s)	2019-07-29 21:34:11.0	2019-07-29 21:31:11.0	56.7	-46.8
K15	ORD	BAA	3,704 MB	5 Min(s)	2019-07-31 16:46:56.0		14	-60.8
K15	ORD	BAA	0,190 MB	24 Min(s)	2019-07-31 09:21:35.0	2019-07-31 09:45:46.0	23	-74
K15	ORD	BAA	8 MB	65 Min(s)	2019-07-31 16:30:29.0	2019-07-31 16:25:17.0	27	-67
K15	ORD	BAA	40,87 MB	5062 Min(s)	2019-06-28 14:40:13.0	2019-06-27 08:22:08.0	40.29	-50.41
K15	ORD	BAA	6,192 MB	1122 Min(s)	2019-06-31 06:25:34.0	2019-06-31 01:17:29.0	44.05	-50.6
K15	ORD	BAA	74,448 MB	116 Min(s)	2019-07-31 02:21:13.0	2019-07-31 02:17:58.0	22.39	-67
K15	ORD	BAA	9 MB	155 Min(s)	2019-07-31 15:16:29.0	2019-07-31 17:51:03.0	16	-78
K15	ORD	BAA	10,719 MB	113 Min(s)	2019-05-01 02:50:32.0	2019-05-01 04:48:41.0	14.79	-79.67
K15	ORD	BAA	9 MB	48 Min(s)	2019-07-30 22:32:42.0	2019-07-30 21:16:17.0	24	-64
K15	ORD	BAA	1,212 MB	165 Min(s)	2019-07-30 16:14:24.0	2019-07-30 16:59:42.0	44.7	-45.47
K15	ORD	BAA	0,440 MB	84 Min(s)	2019-07-27 11:24:19.0	2019-07-27 12:48:34.0	40.33	-55.86
K15	ORD	BAA	64,257 MB	754 Min(s)	2019-07-30 02:17:48.0	2019-07-30 14:01:40.0	31.22	-57.75
K15	ORD	BAA	5,44 MB	830 Min(s)	2019-07-31 03:42:58.0	2019-07-31 17:52:30.0	37.16	-53.66
K15	ORD	BAA	75,313 MB	5 Min(s)	2019-07-31 17:34:54.0		33.17	-55.44
K15	ORD	BAA	1,796 MB	333 Min(s)	2019-06-15 03:51:58.0	2019-06-15 06:45:16.0	1.7	-76
K15	ORD	BAA	0,582 MB	115 Min(s)	2019-07-31 15:37:28.0	2019-07-31 17:52:30.0	15.02	-49.43
K15	ORD	BAA	1,209 MB	145 Min(s)	2019-07-26 16:46:12.0	2019-07-26 17:14:03.0	26.07	-57.13

Figure 8 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

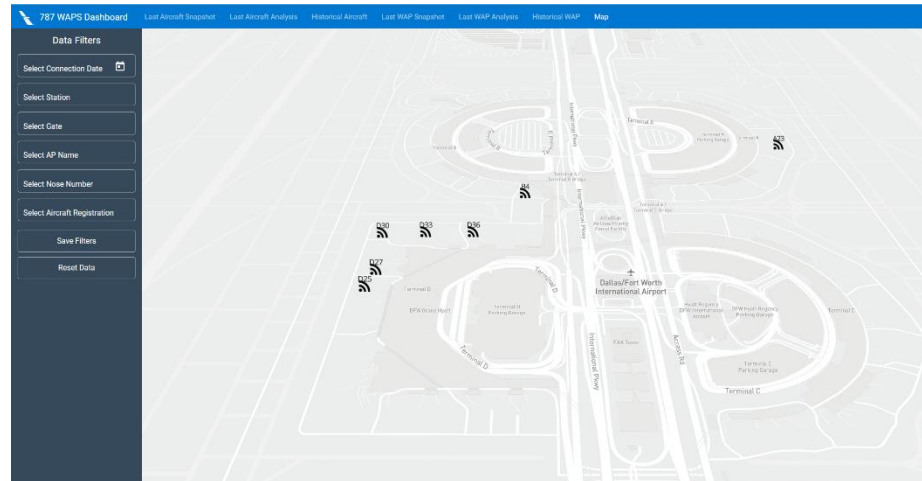


Figure 9 WAP Map

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.



## 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

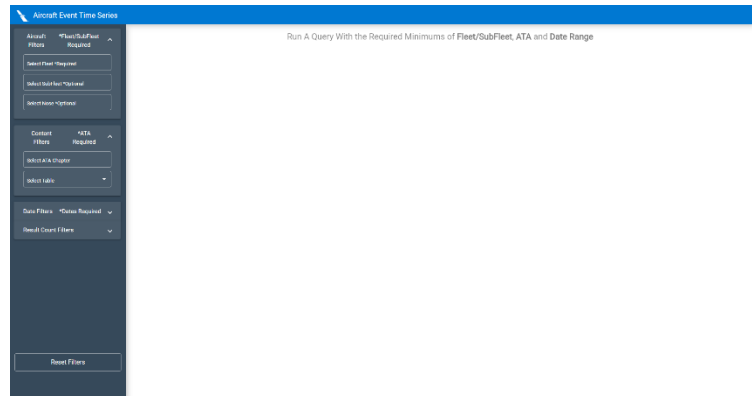


Figure 10 Landing View

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.

## 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 be 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



Figure 11 Query Result

built off of routing to facilitate each of the different views that will be necessary for the application.

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

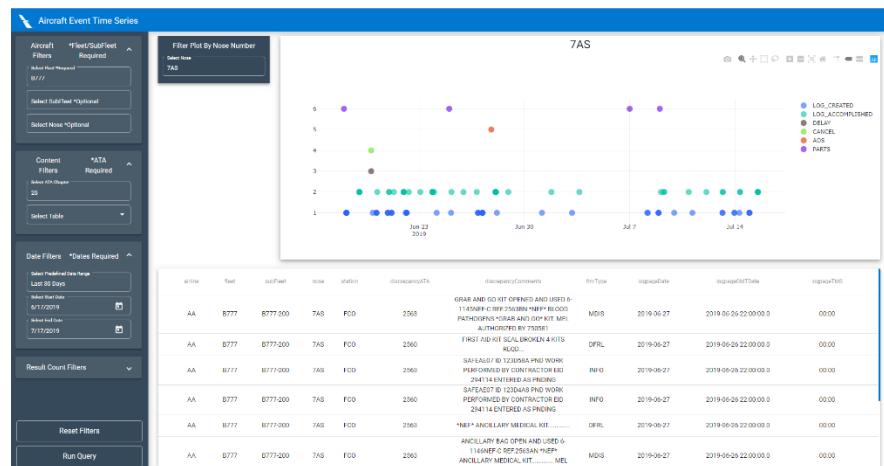


Figure 12 Plot Clicked View

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.

## 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

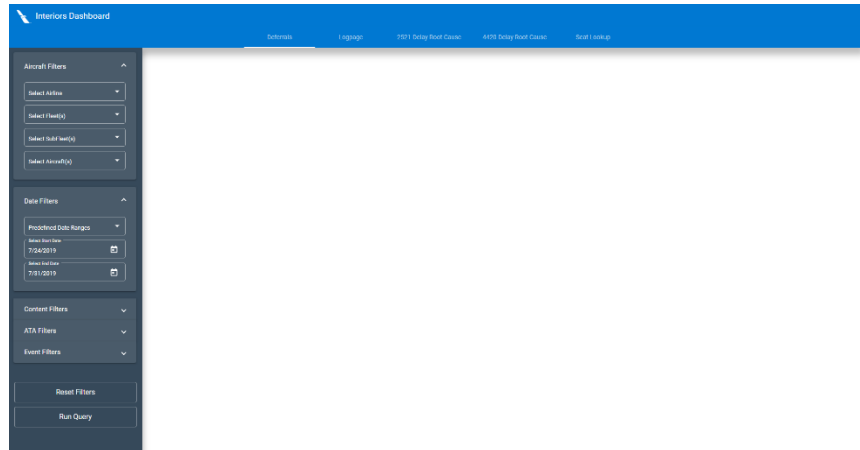


Figure 13 Deferrals Landing

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.

### 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.

LogPage	DATE	FIRM	Airline	Fleet	SubFleet	Aircraft	DeferralDate	DeferralStation	DeferralType	MELNumber	MELDoc	FlightNumber	ATA	Configuration	Countable	Description	Color	Description
39023860	SAFE	009442014	AA	A320F	A319	009	2019-07-24	DFW	NEF	07-1448-C	44-200	2389	4420	DFW	2019-07-25	2		"NEP" PSGR SEAT BACK VIDEO DISPLAY...
39023867	SAFE	009262015	AA	A320F	A319	009	2019-07-24	DFW	MEL	07-1448-C	224-10	2389	4420	DFW	2019-07-25	2		PRE-RECORDED PSGR ANNOUNCEMENT SYSTEM
39105566	SAFE	017231256	AA	A320F	A319	017	2019-07-24	AUS	MEL	07-1414-B	29-270	9999	4420			7		PRE-RECORDED PSGR ANNOUNCEMENT SYSTEM
39105566	SAFE	017441253	AA	A320F	A319	017	2019-07-24	AUS	NEF	07-1413-C	44-200	1366	4420			7		"NEP" PSGR SEAT BACK VIDEO DISPLAY...
39117002	SAFE	022201928	AA	A320F	A319	022	2019-07-24	MIA	MEL	07-1404-C	29-201	2579	2520	MLA	2019-07-25	2	Q	"NEP" COACH SEAT WINDOW SHADE.....

Figure 14 Deferrals Search Results

### Section 2.d.iii: What I Did

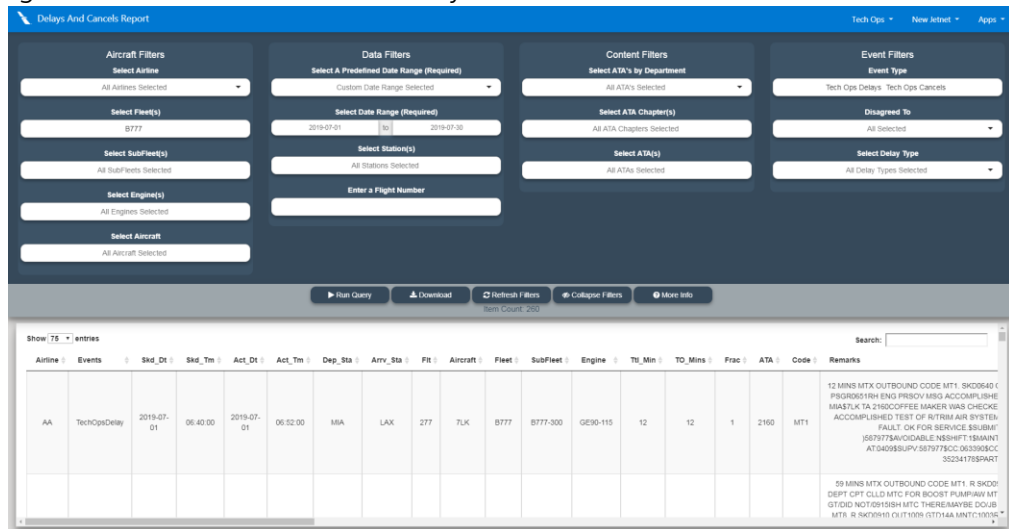
I built the application starting with a mock up to see how I wanted to deal with the different views, and then determined the filters and layout of the columns in the data with the customers. I built the user interface and the data service for the connection to the NodeJS api. I also built the NodeJS 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



Aircraft	Events	Skd_Dt	Skd_Tm	Act_Dt	Act_Tm	Dep_Sta	Arrv_Sta	Flt	Aircraft	Fleet	SubFleet	Engine	Ttl_Min	TO_Mins	Proc	ATA	Code	Remarks
AA	TechOpsDelay	2019-07-01	06:40:00	2019-07-01	06:52:00	MIA	LAX	277	7LK	B777	B777-300	GE90-115	12	12	1	2160	MT1	12 MINS MTK OUTBOUND CODE MT1. SKD0640 C P0000011000 ENG PRESS 1000 ACCOMPLISHED MAINT TA 2100COFFEE MAKER WAS CHECKED ACCOMPLISHED TEST OF INTRIN AIR SYSTEM. FAULT OK FOR SERVICE SUBMIT (567975A/VOIDABLE NSHIFT 15MAINT) AT 540958JUPV 5679778CC 0633000CC 30241789MKT

Figure 15 Delay And Cancel Report

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.

### 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 they 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 than 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.