Wireless Engine Downloader - Bluetooth Prototype

Client: Harlan Mitchell and Gary Matsch

Mentor: Austin Sanders

Brandon Samz, Joe Griffith, Robert McIntosh, Corban Stevens

What nobody wants to see

Why care?

- In 2016 - there were 8,185,533 flights world wide
- 65 had accidents 10 of which had fatalities because of them

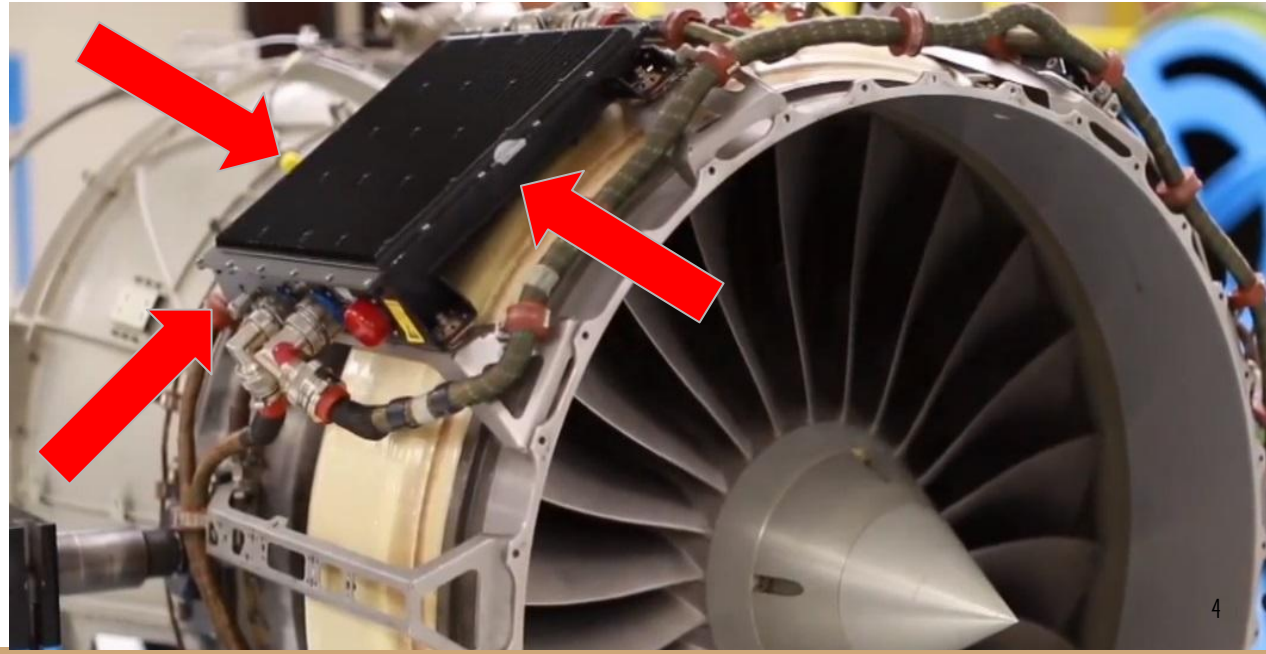
2016 Safety Performance

	2016	2015
Fatalities*	268	136
Total Accidents	65	68
Fatal Accidents	10	4

Data retrieved from: http://www.iata.org/pressroom/pr/Pages/2017-03-10-01.aspx

Preventing Engine Failure

- Gathering data after every flight
- Collecting and analyzing data from many different flights
- Fix problems before happen
- Data is stored on an onboard computer called the engine control unit or ECU

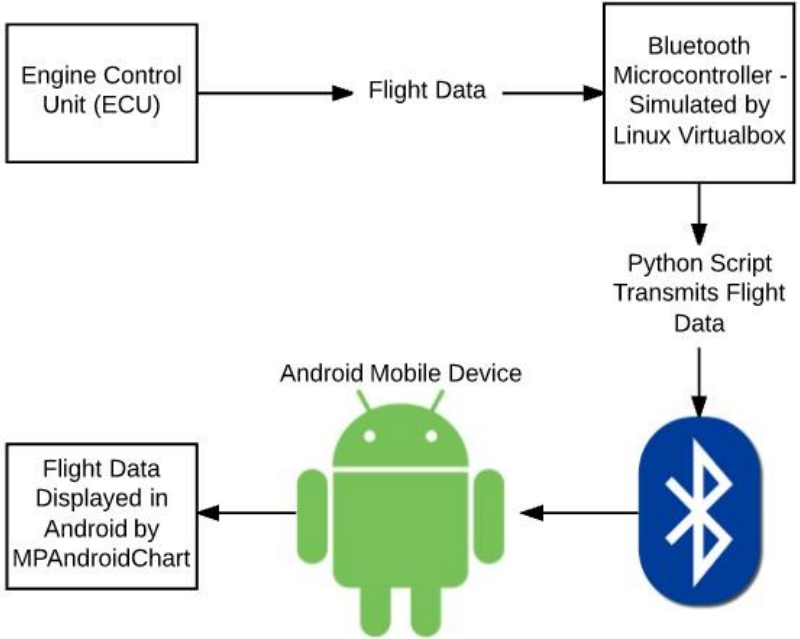


Current Problem

- Data must be downloaded manually through bulky and slow cables
- Download speed is very slow and currently this whole process takes around 30 minutes to get the data off the plane
- Electronic engine interface (EEI) is old and only runs on Windows XP
- All this makes for data that is collected rarely
- Our client wants to upgrade this process so that their solution stands out in the current market

Solution Overview

- Bluetooth connection to the microcontroller is paramount
- The functions of the microcontroller will be simulated with *Linux Virtualbox* for testing purposes
- *Android* will be our mobile platform of choice
- Flight data will be displayed using *MPAndroid Chart*

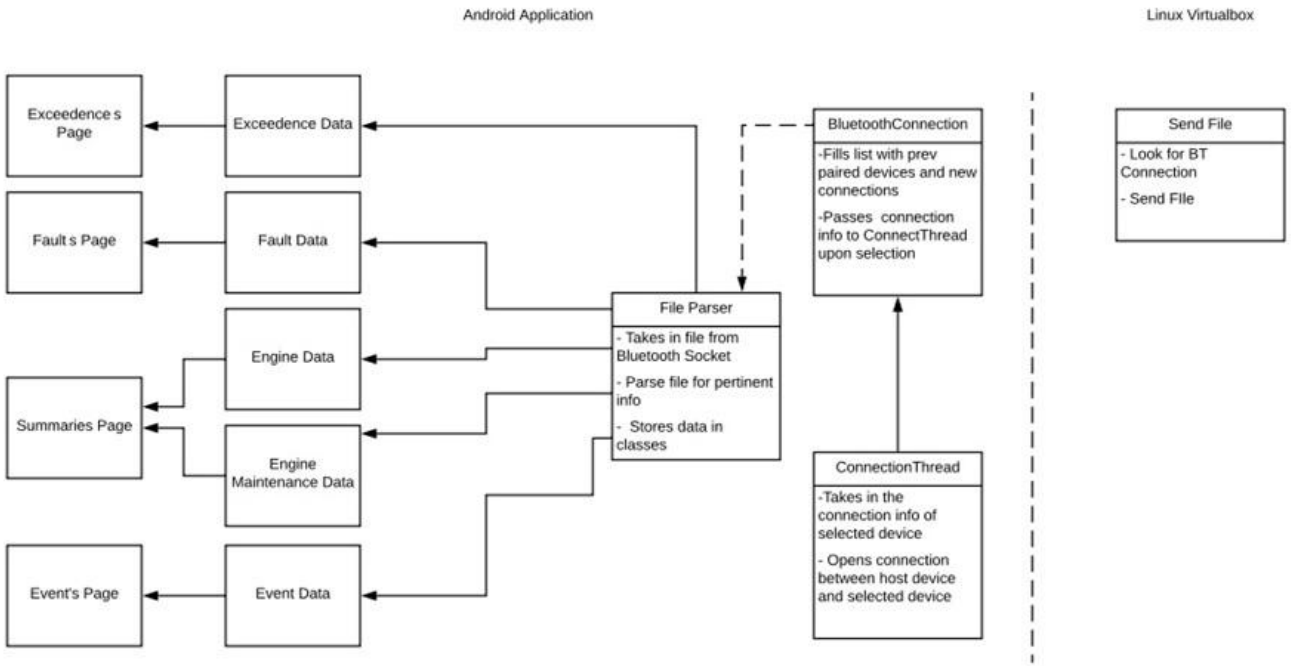


Key Requirements

- Engine download application connects to the microcontroller via Bluetooth and downloads engine data
  - Obtains Bluetooth socket
    - Connects to Bluetooth socket
  - Receives input stream
    - Reads from input stream
    - Data stored on device
  - Closes input stream and Bluetooth socket
- Download time will be under 30 seconds
- Engine data can be downloaded anytime or place the plane has landed, with only a smartphone running the engine download application
- Application should allow for review of engine data, with functionality similar to EEI

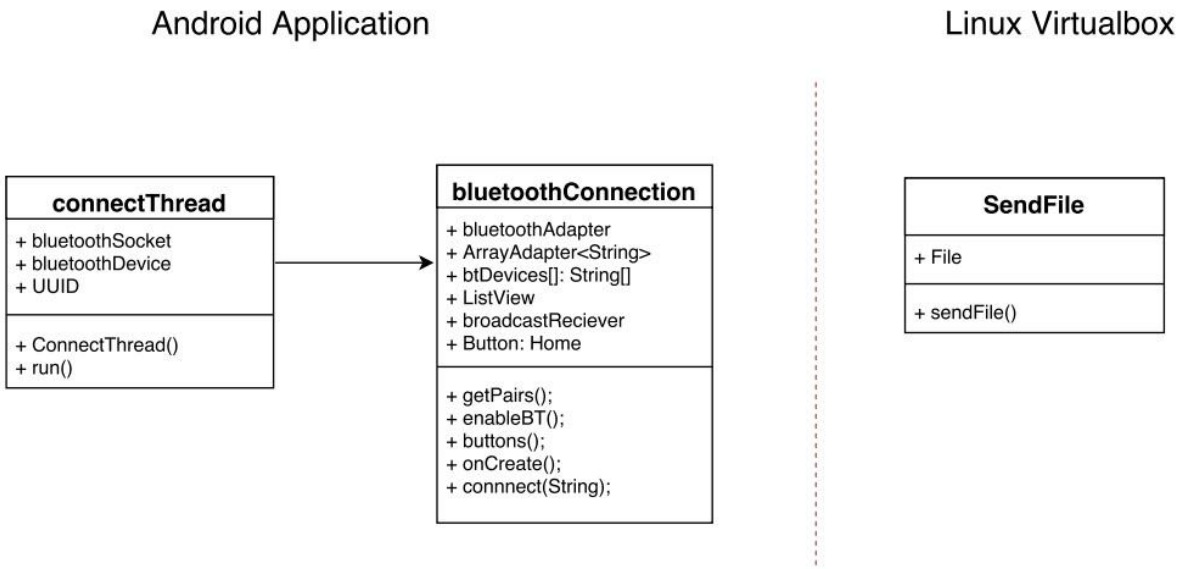
Architecture Overview

- Model-View-Presenter:
  - Model: Download file
  - View: GUI - Charts and Tables
  - Presenter: File parser



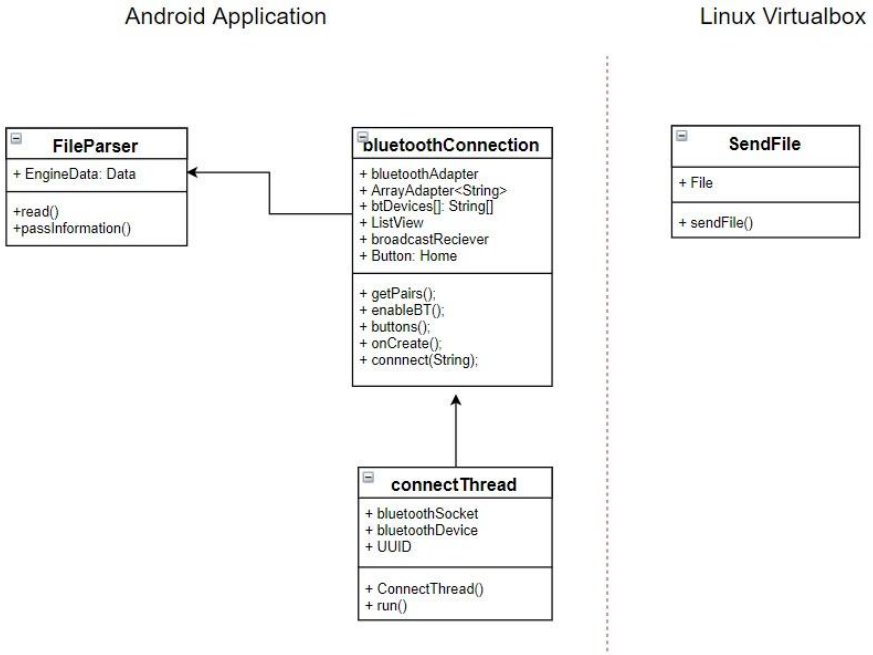
Implementation Overview - Bluetooth Handler

- Bluetooth Handler
  - Opens Bluetooth sockets
  - Writes data to file
  - Closes socket



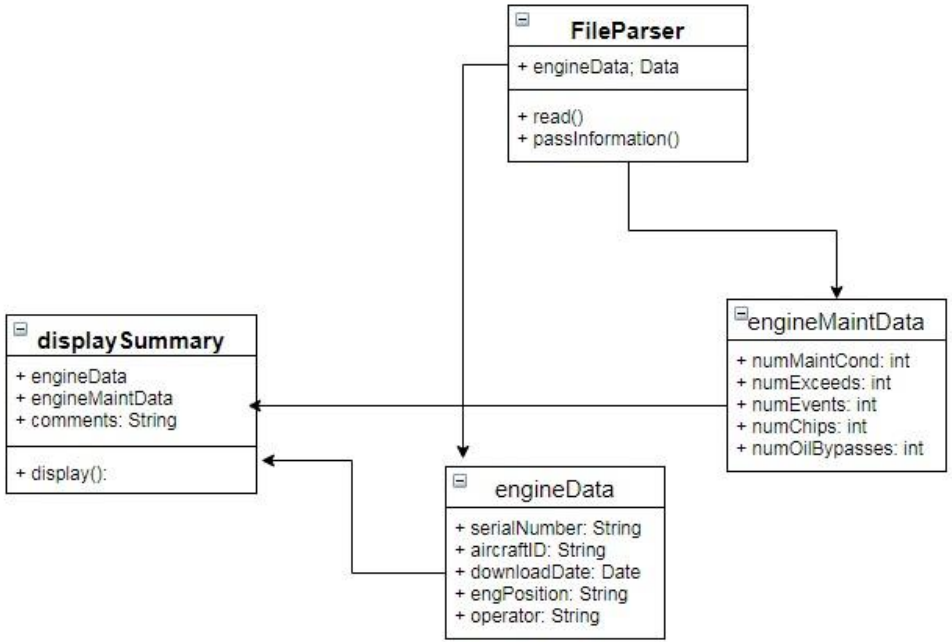
Implementation Overview - File Parser

- File Parser
  - Opens YAML file stored by the Bluetooth module
  - SnakeYAML is used to parse the data
  - Parsed data directly populates an object of DownloadData class



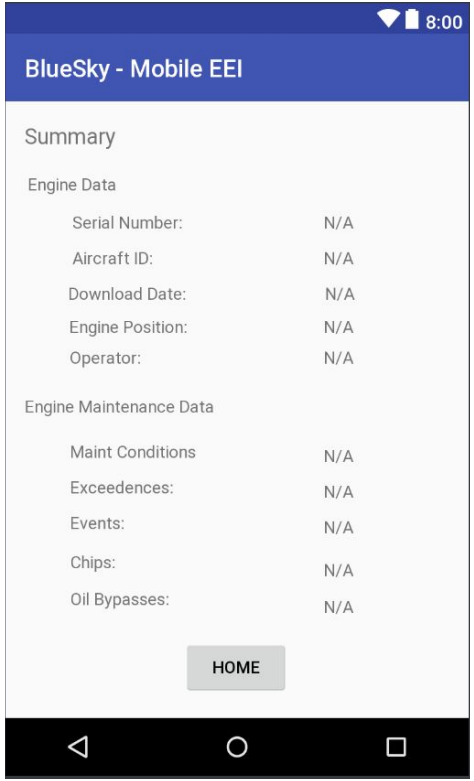
Implementation Overview - Individual Pages

- Individual Pages
  - Retrieves necessary data from DownloadData class
  - Displays this data
  - Some pages use MPAndroidChart to display data



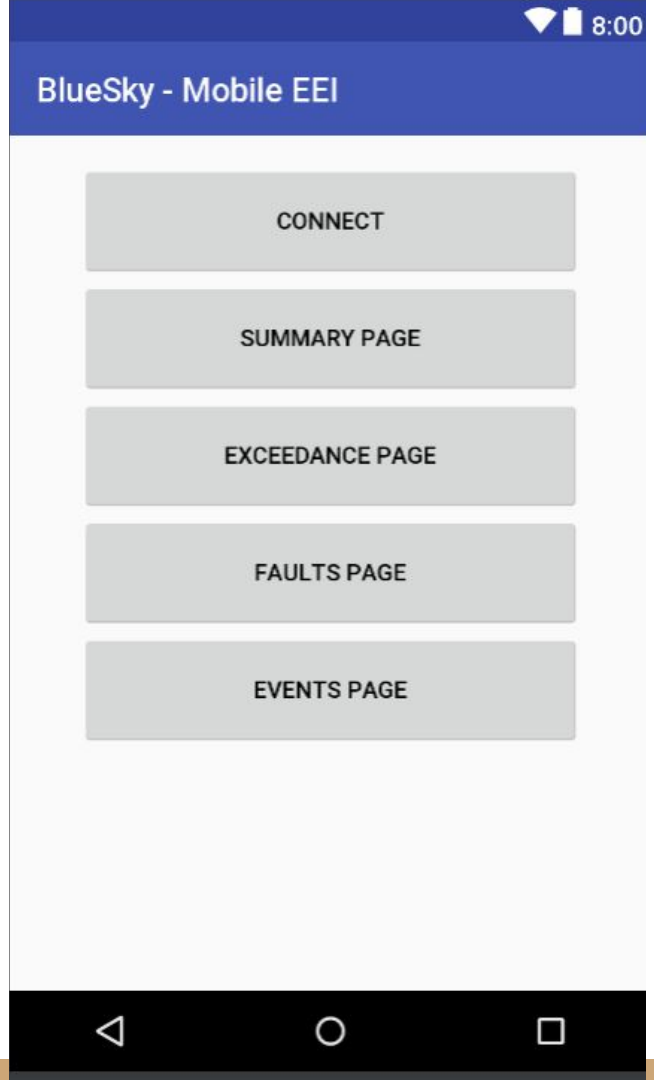
Implementation Overview - GUI

- GUI
  - Data is displayed in a format easy for user to read
  - User is able to navigate through easily and find necessary information



Prototype Review

- Main Menu
  - Menu page where users can navigate through the functionality of the app

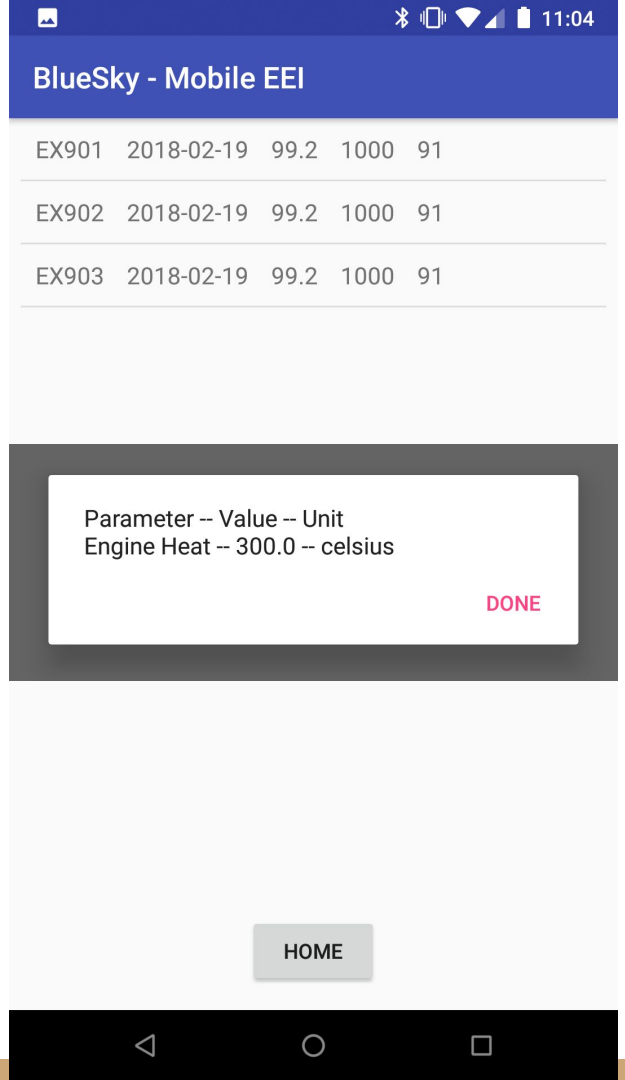


Prototype Review

- Connection Page
  - In the connections page users can select from previously paired devices in order to choose a connection
- Once the connection is made the download takes place in under 10 seconds

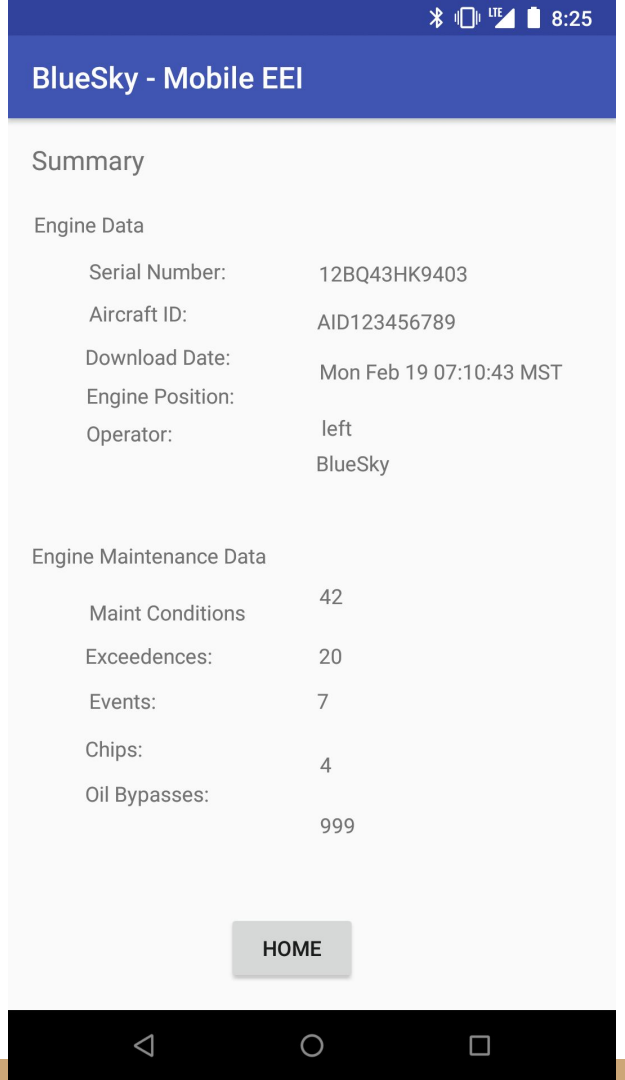
Prototype Review

- Exceedances Page
  - After the data has been downloaded pages displaying information about the engine data can be accessed
  - In the exceedances page the user can view the different exceedances in the data and tap on each individual exceedance in order to display more information about it



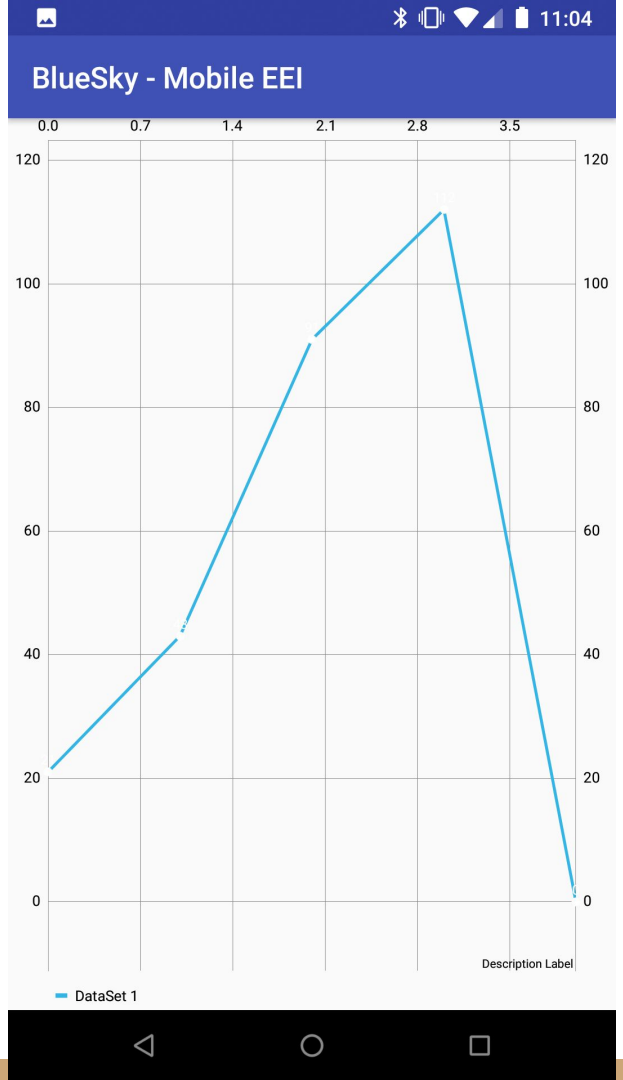
Prototype Review

- Summary Page
  - Basic information is displayed here, which provides the user with basic information about the engine this data came from



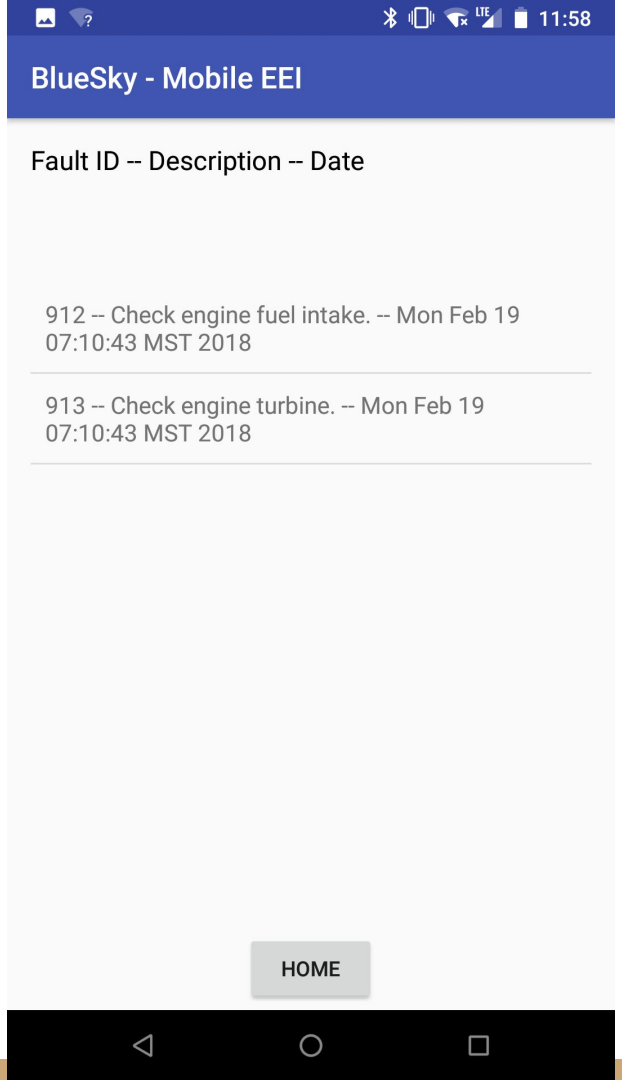
Prototype Review

- Events Page
  - The events page displays a graph over time of a sensor



Prototype Review

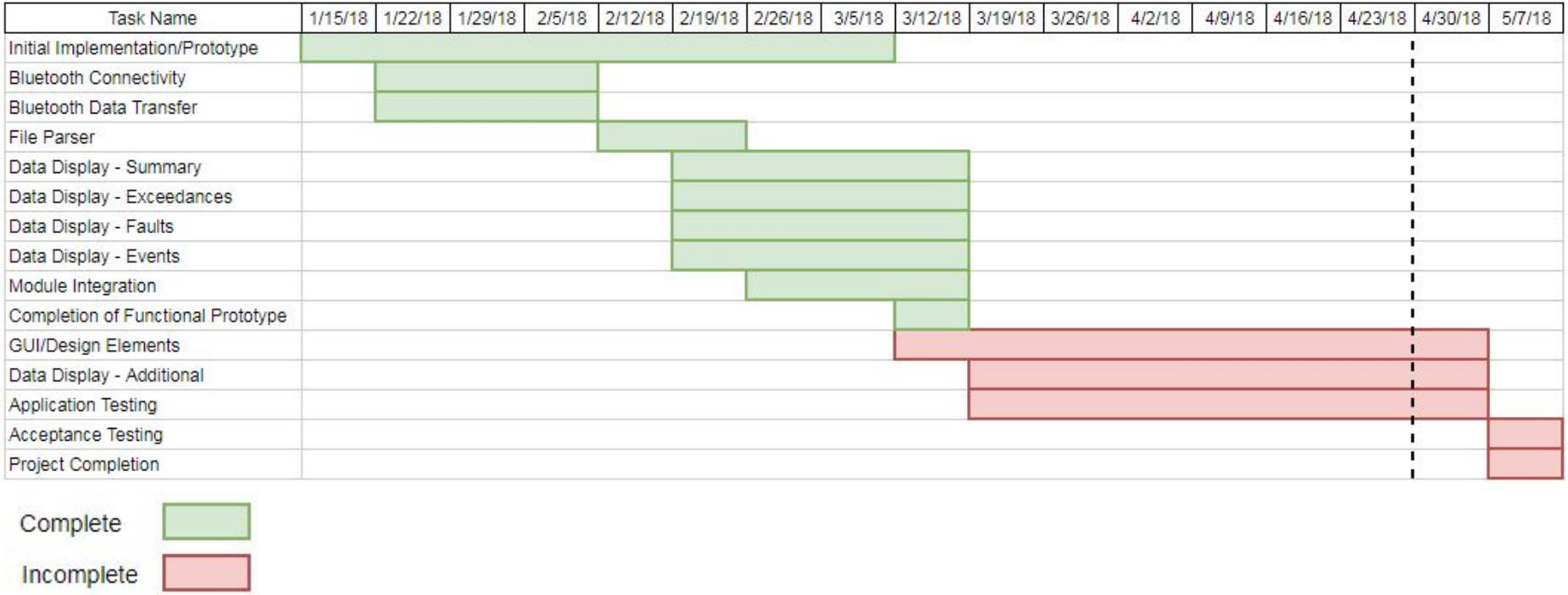
- Faults Page
  - Any faults are populated on this page, which consist of an ID, description, and a date.



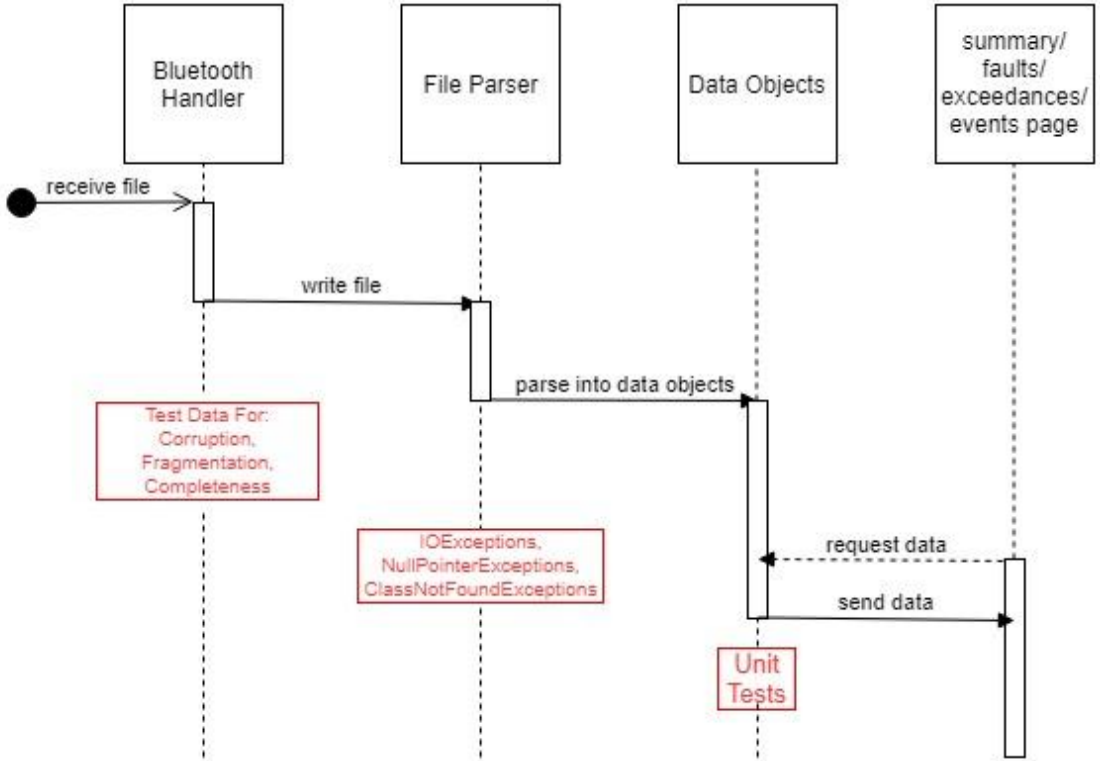
Challenges/Resolutions

Challenges	Resolutions
● Debugging Bluetooth Errors	● Tested the application on a device connected to a computer and displayed messages within the application
● Displaying Pop-Up Messages and Modifying Navigation System	● Certain pages had to be restructured and modified and the data had to be passed differently.

Schedule


Task Name
1/15/18
1/22/18
1/29/18
2/5/18
2/12/18
2/19/18
2/26/18
3/5/18
3/12/18
3/19/18
3/26/18
4/2/18
4/9/18
4/16/18
4/23/18
4/30/18
5/7/18
Initial Implementation/Prototype
Bluetooth Connectivity
Bluetooth Data Transfer
File Parser
Data Display - Summary
Data Display - Exceedances
Data Display - Faults
Data Display - Events
Module Integration
Completion of Functional Prototype
GUI/Design Elements
Data Display - Additional
Application Testing
Acceptance Testing
Project Completion
Complete
Incomplete

Testing Plan - Overview



- Bluetooth Handler:
  - Corruption: Catch any IO or File Not Found Exceptions
  - Fragmentation: Ensure the data file has not been reordered
  - Completeness: Compare downloaded file size to original file size
- File Parser:
  - Catch any IO, Null Pointer, or Class Not Found Exceptions
- Data Objects:
  - Conduct Unit Tests

Testing Plan - Unit Tests

- To test the input to our GUI we will test entering incorrect types to the data file
- In this event our app will display an error message informing the user to check the data file.
- 27 unit tests total

	Test Input	Valid Input	Test Output
Engine Serial	ASCII Character String Null Value	*Type:* Int *Ex:* 84576285	"Error: Invalid input type, check data file
Aircraft ID	Special Characters Malformed String	*Type:* String *Ex:* C7-ABA (Registration Prefix - Designation)	"Error: Invalid input type, check data file
Date - Time	ASCII Characters Strings Int Double Float Malformed Date Obj	*Type:* Date Obj *Ex:* 2018-02-19 14:10:43 (Year, month, Day - Hour, Minute, Second)	"Error: Invalid input type, check data file
Engine Position	ASCII Characters Int Double Float	*Type:* String *Ex:* Left, Right, Center	"Error: Invalid input type, check data file
Operator Name	Int Double Float Special Characters	*Type:* String *Ex:* John Doe	"Error: Invalid input type, check data file

Testing Plan - Usability Testing

- The team will provide scenario to the tester, which will ask for specific pieces of data found within the application.
- Tester will also be provided a questionnaire to provide feedback on user interface and ease-of-use.
- This information will be used to improve the user interface and experience of the application. The goal is to ensure that information is accessible and easy to find and see.

Future Work / Sponsor Impact

- Product will be given back to the sponsor.
- Business case will be presented to higher ups.
- Honeywell teams will finish working on our product and develop the other technologies that will make this solution possible.
- Having this product will allow Honeywell to have an advantage in this market place as there are only a few other companies have technology similar to this.
- Our work has saved Honeywell around 400+ man hours by not having to develop a prototype app for this service.

Conclusion

- Current Problem
  - Problems in aircraft engines can be fatal.
  - Our client builds and maintains aircraft engines.
  - Current method of extracting data off of the engine is cumbersome and slow.
  - Engine data is not collected often enough.
- Solution Overview
  - Build an application that downloads the engine data over Bluetooth.
  - The application should then display the data so that the technician can review it.

BlueSky Group