TLDWorkerBee

Team: Austen Christensen, Morgan Lovato, Wei Song
Sponsor: Harlan Mitchell - Honeywell
Mentor: Austin Sanders

Requirements Document - Version 2
December 5, 2018

Accepted as baseline requirements for the project:

_____	_____
Client Signature	Date
_____	_____
Team Lead Signature	Date

Table of Contents

1. Introduction

Planes are one of the most popular ways to travel quickly. For instance, a traveler can get from Phoenix to Los Angeles in a little over an hour by plane when it takes 8+ hours to drive. In order to travel such long distances, these machines generally have a cruising altitude of 33,000 to 42,000 feet. Every day, there are over one hundred-thousand flights scheduled across the globe with up to one million people in the air at any given point in time. This makes it vital for every plane's engines to constantly be working properly since there is nothing stopping a plane from falling out of the sky other than it's own momentum.

Our team is TLD Worker Bee, and we are working on the project Prototype Time Limited Dispatch (TLD) Application for our sponsor, Harlan Mitchell from Honeywell Aerospace, a leading manufacturer of aircraft engines.Honeywell Aerospace develops turbine engines and engine control systems for a variety of private and commercial jets. These engines and their connected systems generate data every flight that is important for the functionality of their product. While in flight, an engine is constantly reading sensor data and storing it on the onboard computer. The computer will read the data and create a time-limited dispatch. Time-limited dispatch allows the degraded redundancy dispatch of aircraft. Aircraft can be dispatched with certain control system faults and fault combinations for specified periods of time if the failure rates from those configurations meet certification requirements. The various system faults and fault combinations are assigned to dispatch categories according to these failure rates. This gives the dispatch criteria for the system.

Currently, to gather this data, a technician will downloaded physically download the data from the Engine Control Unit (ECU) through a wire connection. They do not check it after every flight but will connect periodically to the ECU to retrieve the data. The cumbersome process of physically connecting to a computer and downloading this data on location greatly limits the amount of flight data to collect.

Our prototype will be a webapp that uses an internet connection to connect to the data stored in the cloud for a completely wireless experience.

2. Problem Statement

The current way of gathering TLD data is a long and inefficient task. In order to gather the data, a technician must physically connect a computer to the Engine Control Unit (ECU) via an RS-232 cable and download the data manually using EEI software. This software is so old that it requires Windows XP to run and can take up to thirty minutes to download. This can be very cost inefficient because the technician has to be on site to download the data, determine what work needs to be done on the engine, and schedule the maintenance accordingly. Since the technicians do not always know what to expect, they might not have the right parts or tools they need to perform the maintenance, making the aircraft stay grounded for a longer period of time. If that plane is a commercial aircraft, then being grounded means your customers are losing money and that does not end well for anyone.

The problem that TLD Worker Bee seeks to resolve is to remove the entire physical aspect of diagnosing data from the ECU. This is a problem we seek to resolve because the way that data is currently collected is cumbersome and slow which is not acceptable for clients. A client who pays hundreds of thousands of dollars for an aircraft expects it to be working at all times.

3. Solution Vision

The solution we have in mind is to remove the physical aspects of our client's existing workflow. Honeywell has already accomplished a method of autonomous engine data downloads via wireless uploads to a cloud. Honeywell hopes to make the rest of their workflow for this autonomous as well. The data, once in the cloud, needs an efficient way of being accessed by its users. Our team will be working on a solution to allow users to see the TLD engine data in a cloud connected GUI.

Our team will have to recreate what Honeywell has already accomplished with the cloud, but by simply being able to host data in a cloud. We also have to have a way to pull that data from the cloud into a database of our choice and display it in our chosen GUI. We have chosen the tools we will be using for our cloud, database, and GUI. Two diagrams of our solution are below:

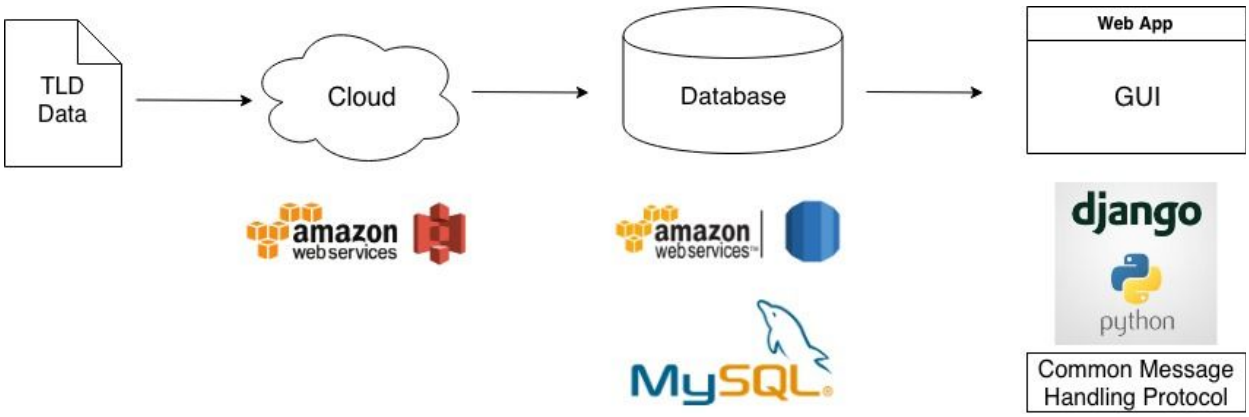

Fig. 1. – System Diagram

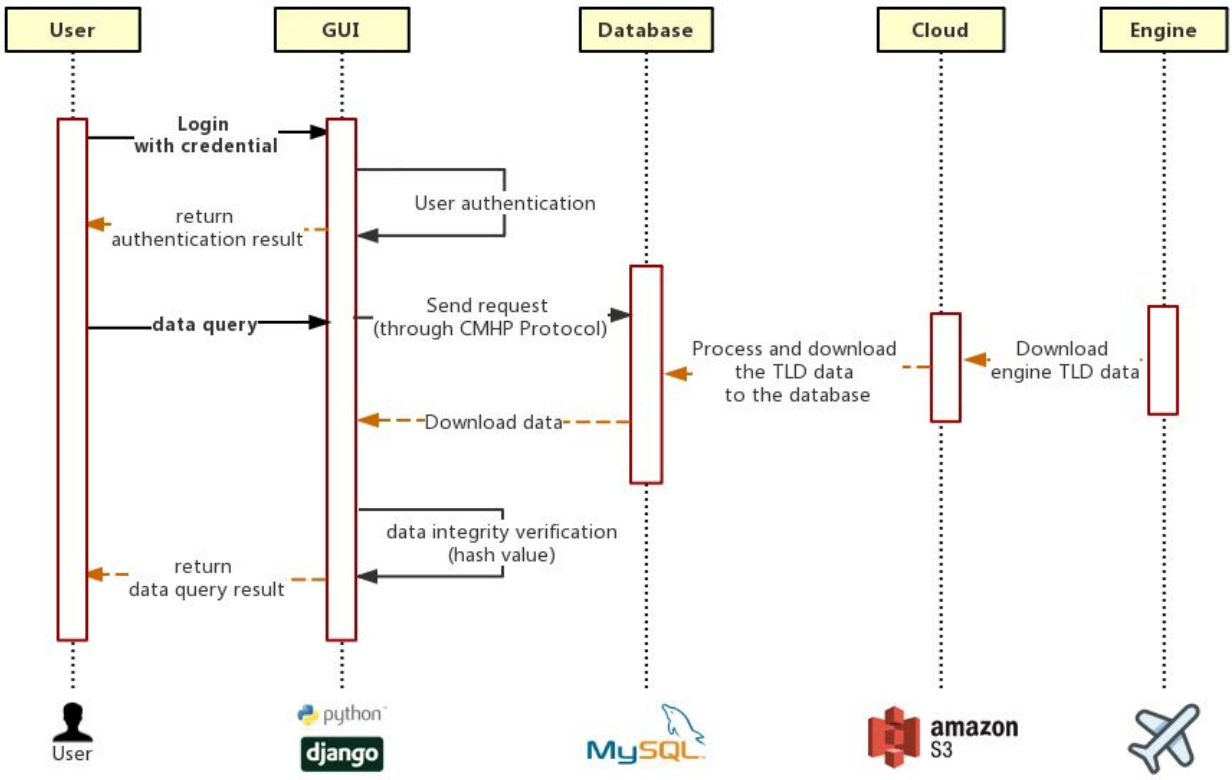

Fig. 2. – UML System Architecture

We will be using:

- Amazon Web Services Simple Storage Service (AWS S3) for our cloud storage
- AWS Relational Database Service (RDS) and MySQL for our database
- Django to implement our GUI

We have two tools for our database because one is local and one is cloud based; we are using both because our client is unsure of which method they will like to use in the future and would like us to experiment with both. Because we are displaying the data into a web app for our GUI, we need an internet protocol to ensure data integrity so we will be using Common Message Handling Protocol (CMHP).

We will be given sample TLD data from our client that we will upload to our cloud, pull into our database, and display in our GUI. In Honeywell's use, their data will come from their previously implemented autonomous upload of data to the cloud; they will use the backend of our database and GUI for further development of the process they are hoping to accomplish.

Our solution will help Honeywell and our client finish what they have previously started. Their current workflow is almost completely autonomous and our project will help them accomplish this goal.

4. Project Requirements

In this section, we will outline the requirements that our web app must meet in order for our product to effectively meet our client's needs. All of the functional, non-functional, and environmental requirements will be expanded in sub sections labeled as such. Each requirement has a tag associated with it to help with identification of requirements; tag prefixes are: SYS (System), DT (Data), and GUI (Graphical User Interface). Each requirement is then numbered by tag prefixes to further help identification. Lower level requirements are numbered according to the higher level requirement they fall under.

To come up with our project requirements, our team began evaluating use cases of our system. We were able to come up with a typical use case for a technician; it is as follows:

- The user creates an account with given access code.
- The user logs in to the system with their username and password.
- The user searches for their desired data by plane identification number.
- The user is able to read and analyze this data from within their web browser.

We also came up with a typical use case for an administrator; it is as follows:

- The administrator groups planes together by owner.
- The administrator gives owners access code.
- The administrator updates files in cloud.
- The administrator updates the list of owners.

The use cases helped us identify what functions our system will need to have and how it will be expected to perform.

4.1 Functional Requirements

In order for our product to be considered a success, it must meet the following functional requirements:

[F-SYS] The system shall gain access to the data in the cloud.

> [F-SYS1] The system shall download the EEI file from the cloud to the user's computer upon user's request. This will make it so the user can optionally download the file to their computer for offline use.
>
> > [F-SYS1.1] The administrator shall be the only one to update files in the cloud. This feature will help with data security.
>
> [F-SYS2] The system shall gain access to the data stored in the database to view. This will be the online alternative for viewing the data.
>
> > [F-SYS2.1] The administrator shall be the only one to update/add entries in the database. This will help with data security.
>
> [F-SYS3] The system shall reflect a new and/or changed data entry within ten minutes of the database being updated. This will reduce the number of incidents of incorrectly displayed information by 75%.

[F-DT] The data shall be accessible to anyone with an internet connection and permission.

> [F-DT1] The system shall verify user has permission to view data. The data shall only be viewed by the owner and/or mechanic.
>
> > [F-DT1.1] The system shall authenticate a user login. This will ensure the data is secured and cannot be viewed by outside parties.
> >
> > [F-DT1.2] The system shall prompt user to login if not currently logged in before displaying data. This will ensure the user who is viewing the data, has permission.

[F-DT1.3] The system shall prompt user if they are not authorized to view data. This will help users know they made an error or need to contact support to gain access.

[F-DT1.4] The system shall display information that is customized based on the user. This feature is so the user will only see the data for planes they own or operate on. This makes it easier and faster for users to navigate to their desired data.

[F-DT1.4.1] The administrator shall store the owner/business's ID along with the data for easy filtering.

[F-DT1.5] The administrator shall update the list of owners and their planes. This will make it so there can't be any mistakes about who owns what plane and there is more control on the distribution of access codes.

[F-DT1.5.1] The system shall generate an access code for administrators to give out to owners and mechanics so they may create an account. The system will only do this when an administrator enters in a new owner into the database.

[F-GUI] The Graphical User Interface shall have the following functionalities:

[F-GUI1] The user shall search for data by plane tail number. This makes desired data easily reachable.

[F-GUI2] The user shall create a login and password with an access code. The access code will link their account to a group of planes. This feature makes it easier for users to gain access to their group of planes.

[F-GUI3] The GUI shall be adaptive to any kind of platforms. E.g. Cell phone, desktop, tablets, etc. This will increase the ease of use and make the data more accessible.

Requirement F-SYS addresses the basic goal for this project from our client: to develop a product to allow engine downloads the TLD data, upload to our cloud prototype wirelessly and then be able to access remotely.

Requirement F-DT addresses the importance of the system security. We need to make sure that only the persons with correct credential can get access to the TLD data, so an authenticate process is necessary and provided in the requirement F-DT1.

Requirement F-GUI addresses some main features of the user interface, which is specified in the project description that our team needs to prototype a modern cloud-connected GUI for the purpose of evaluating TLD data.

With the functional requirements listed and explained, table 1 shows how they relate to the use cases described above:

Table 1 - Functional Requirements and Use Cases

Use Case	Requirement(s)
The user creates an account with given access code.	F-DT1.5.1, F-GUI2
The user logs in to the system with their username and password.	F-DT1.1, F-DT1.2, F-DT1.3
The user searches for their desired data by plane identification number.	F-GUI1
The user is able to read and analyze this data from within their web browser.	F-GUI3
The administrator groups planes together by owner.	F-DT1.4.1
The administrator gives owners access code.	F-DT1.5.1
The administrator updates files in cloud.	F-SYS1.1, F-SYS2.1
The administrator updates the list of owners.	F-DT1.5

4.2 Performance Requirements

In this section we will be evaluating the performance requirements for our system. These non-functional requirements serve as benchmark goals for navigation times and data accuracy. The performance requirements for our system are as follows:

[GUI4] The user shall be able to quickly navigate to their plane of interest.

> [GUI4.1] The user shall be able to navigate to the data they are looking for in under two minutes of logging in.
>
> [GUI4.2] 90% of certified technicians shall read the data and understand the general layout within their first use.

[DT2] The database shall be protected.

> [DT2.1] The database shall reject SQL injection 100% of the time.

[SYS4] The system shall verify data for correctness.

> [SYS4.1] The system shall display accurate data 100% of the time.
>
> [SYS4.2] The system shall explicitly validate the data after receiving it from the cloud before displaying it on the web browser.

[SYS5] Access codes shall be given to administrators first.

> [SYS5.1] The system shall only allow five accounts per access code. (This amount can easily be changed to allow for more or less accounts in the future for easy scalability.)

[SYS6] Users can quickly create an account.

> [SYS6.1] Users can create an account in under 2 minutes if they have already received an access code.

Requirement GUI4 addresses the speed it should take for typical user to find the data they are interested in. We have a requirement that this should be quick, but we further specified that in requirement GUI4.1 putting a specific time limit on how long this action should take. GUI4.2 states 9/10 users should understand the interface after their first use.

Requirement DT2 addresses database protection. DT2.1 states our database shall reject SQL injection 100% of the time to ensure protection.

Requirement SYS4 address the data accuracy importance our client has stressed to our team. The data should be verified for correctness, but as requirement SYS4.1 states, this data should be displayed correctly 100% of the time and SYS4.2 states when the system will be explicitly validating the data.

Requirement SYS5 addresses the use of access codes; SYS5.1 states that only five accounts can be created for every access code.

Requirement SYS6 address the speed at which a user should be able to create an account and SYS6.1 specifies a time limit for that action.

With the performance requirements listed and explained, table 2 shows how they relate to the use cases described above:

Table 2 - Performance Requirements and Use Cases

Use Case	Requirement(s)
The user creates an account with given access code.	SYS6, SYS6.1, SYS5.1
The user logs in to the system with their username and password.	GUI4.1
The user searches for their desired data by plane identification number.	GUI4
The user is able to read and analyze this data from within their web browser.	GUI4.2
The administrator gives owners access code.	SYS5, SYS5.1

4.3 Environmental Requirements

In this section we will be evaluating the environmental requirements for our system. These can be considered constraints imposed on our project either directly from the client themselves or from the need to integrate the parts of our solution. The environmental requirements for our system are as follows:

[SYS7] The system shall be qualified per the FAA software development process.

[SYS7.1] The system shall comply to the DO-178C development process.

[GUI5] The data shall be displayed in a web browser.

[GUI5.1] The system shall support browsers: Google Chrome, Apple Safari, Mozilla Firefox, and Internet Edge.

Requirement SYS7 addresses a constraint placed on our software development process. Because our client works for Honeywell, our software must meet FAA software development process guidelines. The specific guidelines we have to follow are named in requirement SYS7.1. The DO-178C is a document used by the FAA to approve commercial software-based aerospace systems. Our team must follow this document when developing our software.

Requirement GUI5 addresses the method of data display our client would like us to use. We have been asked to display this data in a web browser and GUI5.1 address which browsers we will have support for.

With the environmental requirements listed and explained, table 3 shows how they relate to the use cases described above:

Table 3 - Environmental Requirements and Use Cases

Use Case	Requirement(s)
The user is able to read and analyze this data from within their web browser.	GUI5, GUI5.1

5. Potential Risks

In the feasibility analysis, we have analyzed four technical challenges and outlined the whole software architecture for this project. Currently, our team has identified three potential risks based on those challenges and the whole software architecture. These three risks are cloud and database error, network connectivity and network security.

5.1 Cloud and Database Error

The first potential risk is cloud and database error. This is one critical risk we need to consider. In some cases, by either some factor like the power failure on the machine or human factor like incorrect command send to the cloud and the database, the whole system may break down and the whole database may not be able to recover. This is a

very serious problem because if we lose all the data, then the engine cannot be diagnosed properly, which will result in engine failure or even a plane crash.

However, we can have some mechanisms to solve this problem conveniently. One possible solution is to introduce a backup mechanism for the whole system. For example, we can have another local database backup storage. If the database failed, we can use our backup data to recover the whole database. If the cost is acceptable, we can even introduce one or more backup cloud and database systems. These two systems will work separately, both getting data from the engine and process them. If one system failed, then we could switch to another system immediately and seamlessly.

We identified this issue as high severity since it is crucial to keep all the TLD data safely; however, the chance this case happen is very low, so we identify the likelihood as 3/10.

5.2 Network Connectivity

Another potential risk is the network connectivity. Since our whole project is based on the network and all of our data are transferred through the network, it is important for us to make sure the connectivity of all the components in our software system. If the network breakdown, the whole system will not be able to work either. So, besides the ways to transfer all the data through the network, we also need to have a backup plan to transfer the data between the components in a different way.

Some possible solution is to keep using the current EEI solution that use the serial connection or USB connection to transfer the data as a backup plan.

Since the network breakdown happens quite often, we identified this issue as a high-level severity and the likelihood is 7/10.

5.2 Network Security

The third potential risk is network security. All the data is processing through the network, which means that there is a chance the whole system will be attacked from outside of the network. In the worst cases, if someone finds out the most vulnerable positions in the system, he may be able to operate and destroy the cloud and the database through SQL injection or other methods. The weakest component in the whole system is the frontend (GUI), which is exposed on the network and directly communicate to the cloud and the database.

For this issue, we can add a credential and verification process to someone who wants to get access to the data. Some techniques to avoid SQL injections may be applied to the software like Prepared Statements (with Parameterized Queries), Stored Procedures, White List Input Validation and Escaping All User-Supplied Input.

We identified this issue as medium severity, since once we can make sure only the people with correct credential can get access to the system, then the security of the data transfer process can be guaranteed by the CMHP protocol. The chance this case happens is also very high, so we identify the likelihood as 7/10.

Table 4 shows a summary of the potential risks, their severity, and their likelihood.

Table 4 - Potential Risks

Risk	Severity	Likelihood
Cloud and Database Error	High	3 / 10
Network Connectivity	High	7 / 10
Network Security	Medium	7 / 10

6. Project Plan

For our team schedule, we made a Gantt chart for our project plan.

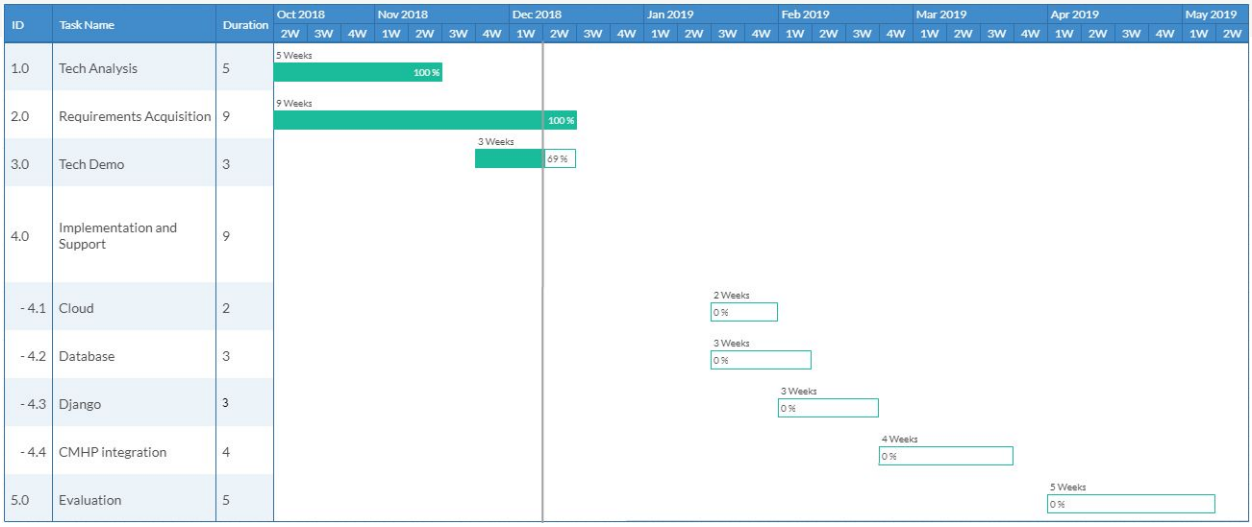

Fig. 3. – Gantt Chart

For now, we have finished team startup in the first 3 weeks, including the team standard, team inventory and team website construction. During October and the first 2 weeks of November, we finished our technical feasibility analysis. Now we are working on the requirements acquisition, which has finished up to 66%, and the tech demo, which will be finished in the last 3 week. In the next semester, we will focus on the implementation and support, which include the cloud (Jan 7 - Jan 20), database (Jan 7 - Jan 27), Django and (Jan 21 - Feb 10) CMHP integration (Feb 11 - March 10). The evaluation will be finished in the last five weeks (April 1 - May 6). For now, our schedule is going pretty well.

7. Conclusion

In conclusion, planes are one of the most popular ways to travel, and it it crucial to have working engines at all times. Our client, Honeywell, wants to make sure that the engine of the airplane does not have any mechanical failures that could result in an accident. To make sure of this, a part of the approach is to download the TLD data for the maintenance and diagnose uses. The current way of getting TLD data is to physically connect a computer to the engine computer via USB and download the data, which is very inefficient. This is our motivation: to provide a fast and secure way to download the TLD data. Our solution is to develop a product to allow engine downloads to be completed autonomously with the data uploaded wirelessly to the cloud where it will then be accessible remotely. Combined with the modern cloud-connected GUI, the engineers in Honeywell can easily read the data and figure out what is wrong with the engine. For the solution overview, we will choose Amazon S3 as our cloud service, Amazon RDS and MySQL as our database system, Django to implement our GUI, and CMHP as our protocol. The next things our team will be focusing on in the next development phase will be the requirements acquisition and tech demo. We believe that our prototype will provide a reliable and easy way to get and display the TLD data for Honeywell.

8. Glossary

- EEI - Electronic Engine Interface - A software used by Honeywell to download and review data.
- AWS - Amazon Web Services - Provides cloud computing, storage services, and more to individuals or companies on a paid basis (a free-tier is available).
- RDS - Relational Database Service - A service offered by AWS that provides a relational database in the cloud.
- S3 - Simple Storage Service - A service offered by AWS that provides cloud object/data storage.
- TLD - Time Limited Dispatch - A kind of fault data concerned with fault that affect thrust control loss.
- CMHP - Common Message Handling Protocol - an application-level protocol developed by the FAA that adds message-delivery assurance to TCP/IP.
- DO-178C - A document used by the FAA to approve commercial software-based systems.