Product TRACkER

Design Document

Team:

SdMay20-02

Client:

Collins Aerospace

Advisor:

Mai Zheng

Team Members/Roles:

Ryan Connolly –– Subject Matter Expert
Tyler Krueger –– Meeting Scribe
Clayton Nida –– Frontend Engineer
Nathan Shull –– Test Engineer
Nathan Yasosky –– Report Manager

Team Email:

sdmay20-02@iastate.edu

Team Website:

sdmay20-02.sd.ece.iastate.edu

Revised:

12/8/2019

Executive Summary

Engineering Standards & Design Practices

- SCRUM Development Style
- Continuous Integration
- Agile Software Programming
- Clean Coding Practice
- Model View Controller Setup
- Controller, Service, Repository Backend Design

Summary of Requirements

- Maintainable electronic database
    - Edit data manually
    - Upload new/updated data via Excel file
    - Excel file can be converted to .csv or .txt
- Search by field values (incl partial) or upload CPN list via Excel
    - Display results on screen
    - Download results to Excel file
- Integration of product and contracts data
    - One product can be on multiple contracts
    - One contract can include multiple products
- Interactive
    - Linking allows moving around between products/contracts

Applicable Courses from Iowa State University Curriculum

- ComS 227
- ComS 228
- ComS 309
- ComS 319
- SE 329
- Engl 314
- SE 409

New Skills/Knowledge acquired that was not taught in courses

- Front end framework: ReactJS
- Connecting Spring Project to MySQL database
- Controller, Service, Repository Backend layout
- SCRUM development standards
- Agile Software Programming Style
- Project Management
- Designing a Gantt Chart

Table of Contents

Executive Summary **1**

Engineering Standards & Design Practices 1

Summary of Requirements 1

Applicable Courses from Iowa State University Curriculum 1

New Skills/Knowledge acquired that was not taught in courses 2

Table of Contents 3

List of figures/tables/symbols/definitions 4

1. Introduction **5**

1.1 Acknowledgement 5

1.2 Problem and Project Statement 6

1.2.1 General Problem Statement 6

1.2.2 General Solution Approach 6

1.3 Operational Environment 6

1.4 Requirements 6

1.5 Intended Users and Uses 11

1.6 Assumptions and Limitations 11

1.7 Expected End Product and Deliverables 11

2. Specifications and Analysis **12**

2.1 Proposed Design 12

2.2 Design Analysis 12

2.3 Development Process 13

2.4 Design Plan 13

2.4.1 Screen Sketches 19

3. Statement of Work **22**

3.1 Previous Work And Literature 22

3.2 Placing Product In Related Work And Literature 22

3.2 Technology Considerations 23

3.3 Task Decomposition 23

3.4 Possible Risks And Risk Management 24
3.5 Project Proposed Milestones and Evaluation Criteria 25
3.6 Project Tracking Procedures 26
3.7 Expected Results and Validation 26
4. Project Timeline, Estimated Resources, and Challenges 27
4.1 Project Timeline 27
4.2 Feasibility Assessment 32
4.3 Personnel Effort Requirements 33
4.4 Other Resource Requirements 39
4.5 Financial Requirements 39
5. Testing and Implementation 40
5.1 Interface Specifications 40
5.2 Hardware and software 40
5.3 Functional Testing 40
5.4 Non-Functional Testing 41
5.5 Process 41
5.6 Results 42
6. Closing Material 42
6.1 Conclusion 42
6.2 References 43
6.3 Appendices 43
Appendix I - Operation Manual 44
Appendix II - Alternative or Previous Versions 45
Appendix III - Other Considerations 45

List of figures/tables/symbols/definitions

Figure 1. Use Case Diagram
Figure 2. Backend Layout Diagram
Figure 3. Front End Layout Diagram

Figure 4. Database Layout Diagram

Figure 5. Controller/Service Layout Diagram

Figure 6. Products Page Sketch

Figure 7. Contracts Page Sketch

Figure 8. Product Details Page Sketch

Figure 9. Notes Page Sketch

Figure 10. Add Product Page Sketch

Figure 11. Edit Product Page Sketch

Figure 12: Milestone 1 Gantt/Timeline/Schedule

Figure 13: Milestone 2 Gantt/Timeline/Schedule

Figure 14: Milestone 3 Gantt/Timeline/Schedule

Figure 15: Milestone 4 Gantt/Timeline/Schedule

Figure 16: Milestone 5 Gantt/Timeline/Schedule

Figure 17: Milestone 6 Gantt/Timeline/Schedule

Figure 18: Milestone 7 Gantt/Timeline/Schedule

Figure 19: Milestone 8 Gantt/Timeline/Schedule

1. Introduction

1.1 Acknowledgement

Throughout the course of the project, we will be assisted by our faculty advisor Mai Zheng. His guidance will help us stay on track, and develop the best plan of attack. Not only will he help keep our team running smoothly, but has already helped tremendously to keep us on track and make sure our project is following and meeting the expectations of the senior design class. Additionally we have and will continue to be in close communication with Jenny Smith. She has already helped out tremendously to relay the problem that Collins has enlisted our help to solve. Consistent communication with her has provided us with guidance on design decisions along with the ability to understand the aspects more closely relating to Collins issue we are trying to solve.

1.2 Problem and Project Statement

1.2.1 General Problem Statement

The original reason for this project and why Collins Aerospace enlisted our help was they did not feel that they had a great system to manage not only their product roadmaps, but also their contracts. The way that they had a lot of important information for both contracts and products stored was through a mess of a wide variety of different excel documents, text files, and other forms of media. This made it difficult to find a lot of the important information they needed on products when drafting contracts with customers, and when they did find the information it was often disorganized. This provided conflicting information, and could sometimes put Collins Aerospace into tough positions where product roadmaps were misaligned with anticipated data in other contracts and employees would have to figure out a way to still fulfill the contract.

1.2.2 General Solution Approach

In order to help solve this problem for Collins Aerospace our goal is to develop a web application of their data in one place, but will allow them to have a way to search their data and make sure they that will manage their contracts and products information. This will allow them to not only have all are coming up with contracts that they know without a doubt will be able to be met for the customer. Therefore our application will need to be able to list all contacts and products Collins currently has. Then it needs to be able to navigate this application easily and search for the exact product or contract that the user is looking for. Additionally the admins of the system will need a way to add more data into the system, that way the application can be long lasting and stay up to date as the company evolves. The last important key about this application is it will need to be able to tell the user the connections that a product has to however many contracts or on the other side a contract and how many products it has associated with it. That way Collins can easily assure customers that their contracts can still be met even as time progresses.

1.3 Operational Environment

The operational environment for this project is a computer that is able to access the Collins internal network in order to hit the website that our application will be deployed on. Since our application is strictly software, the environment on which it is deployed is not going to affect the application. Therefore, there are no physical hazards that we will need to take into consideration while developing this application. However, it is important that we design an application and deploy it an environment that allows the application to have the necessary security regulations on it along with a low-no down time environment.

1.4 Requirements

- Administrative users shall be able to add entries to the database by any of the following:
    - Manually entering individual rows via Add Product or Contract page
    - Uploading a .csv file containing the data to be added
- Administrative users shall be able to modify existing data in the database by any of the following:
    - Manually editing the Product or Contract information on the Details page

    - Uploading a .csv file containing the modified data
- Users shall be able to search for entries in the database by any of the following:
    - Inputting individual field values to the search bar
    - Uploading a .csv file containing a list of Collins Part Numbers
    - Search results shall include entries that match the search parameter for CPN, Contract number, or Customer
    - Search Results shall include partial value matches
- Users shall be able to filter search results
- Product TRACkER shall generate a .csv file containing the search results if "Generate .csv file" is selected in the search options
- Collins Part Numbers displayed on the site shall be hyperlinked to their respective Product Details page
- Contract Numbers displayed on the site shall be hyperlinked to their respective Contract Details page
- Details pages shall open in a new tab
- Product TRACkER shall generate an error message for the following conditions:
    - When one or more of the required data fields are not filled and the user selects "Add Data"
    - When one or more of the data values to be added is invalid for the data type of that field
    - When a Collins Part Number is in the Contract Data to be added, but is not found in the Products Data table
- Product TRACkER shall generate a warning message for the following conditions:
    - When an Administrative user is about to overwrite existing contract or product data with different values
    - When an administrative user is about to remove another user's administrative permissions
- Any action that generates a warning message shall not be executed unless confirmed in the warning prompt
- Product TRACkER shall be accessible from within the Collins Aerospace network
- Product TRACkER shall be available without scheduled downtime from 6A.M. to 10P.M.
- Product TRACkER shall consist of the following components:
    - MySQL Database
    - Spring Boot Framework
    - React.js Framework

- Administrative rights shall be required to view Add Contract and Add Product Page
- Administrative rights shall be required to add or modify entries in the database
- Administrative rights shall be required to remove data from the database
- A user shall have administrative rights if and only if their username is contained in the administrator's table
- Administrative users shall be able to add other users to the administrators table
- Administrative users shall be able to remove other users from the administrators table
- Product TRACkER User Interface shall consist of the following pages:
    - Main Page
    - Product Details
    - Contract Details
    - Add Product
    - Add Contract
    - Administrative Users
    - Help/Hints
    - Person of Contact Details
- The Main Page shall include:
    - A search bar
    - A file upload selection to import a .csv file containing a list of values to search for
    - Sort By dropdown selection
    - Filter Results menu
    - Generate .csv file option
    - Search results pane
    - Help/Hints menu button
- The search Results pane shall be displayed below the search bar
- The Product Detail page shall include:
    - CPN
    - Eqpt
    - Plant
    - Make_or_Buy
    - BU
    - Portfolio

    - POC
    - New_Programs
    - In_Production
    - EOP
    - EOS
    - Replacement
    - Notes
    - A list of contracts containing this CPN
- The Contract Detail page shall include:
    - CM
    - Customer
    - Contract
    - A list of the CPNs included in this contract
    - Help/Hints menu button
- The Add Product page shall include:
    - A File upload selection to import a .csv file containing the new product data
    - A field for every data label in the Product Data table
    - Required fields of data shall be indicated by an '*' by the Data Label
    - Fields with ENUM data type shall be displayed as a drop down menu with the possible choices.
    - Add Product Button
- The Add Contract page shall include:
    - A File upload selection to import a .csv file containing the new product data
    - A field for every data label in the Contract Data table
    - A field to accept a comma separated list of CPNs included in the contract being added.
    - Add Contract Button
- The Administrative Users page shall include:
    - A list of users in the system with administrative rights
    - A required field for entering a users username/email
    - A grant administrative rights button
    - A remove administrative rights button

  - Help/Hints menu button
- The Person of Contact Details page shall include:
  - Exact information fields are still TBD

The product data table shall include the following data columns:

Data Label:	Data Type:	Possible Values:
CPN	VARCHAR	20 characters, alphanumeric
Eqpt	VARCHAR	20 characters, alphanumeric, NO carriage return, NULL
Plant	INT	4 characters, numeric
Make_or_Buy	ENUM	Make, Buy
BU	ENUM	Avionics, MsnSys
Portfolio	ENUM	Based on BU: Avionics: ComAv, B&RA, MA&H, Subc MsnSys: CNG, Subc
POC	VARCHAR	25 characters, alphanumeric
New_Programs	ENUM	Yes, No, Ask
In_Production	ENUM	Yes, No
EOP	DATE	yyyy-mm-dd, default 2099-12-31
EOS	VARCHAR	10 characters, alphanumeric, date (yyyy-mm-dd), TBD, OOS, NULL
Replacement	VARCHAR	120 characters, alphanumeric, NULL
Notes	VARCHAR	255 characters, alphanumeric, NULL

The contract data table shall include the following data columns:

Data Label:	Data Type:	Possible Values:
CM	VARCHAR	25 characters, alphanumeric
Customer	VARCHAR	40 characters, alphanumeric
Contract	VARCHAR	30 characters, alphanumeric

The contract details table shall include the following data columns:

Data Label:	Data Type:	Possible Values:
Contract	VARCHAR	30 characters, alphanumeric

CPN	VARCHAR	20 characters, alphanumeric

1.5 Intended Users and Uses

The end users will be employees at Collins Aerospace. The end uses of this project will be adding products and contracts to Collins' database, along with modifying and retrieving details about products and contracts.This will be used by the Collins Aerospace employees to be able to know all the necessary information about a contract or product before going into discussion and negotiations for possible customers, that way Collins Aerospace can ensure that the contracts and/or products that they promise to customer can in fact be met. This will account for contracts or products being expired or out of production, so that way they do not make false promises.

1.6 Assumptions and Limitations

One assumption is that authentication will be done on Collins' end. This is important to the application because it will allow employees to use the service, and it is also necessary for some tasks that require a certain level of privilege. Another assumption is that this product will not be used outside of Collins Aerospace because this product is being developed specifically for their needs.

One limitation for the project is that we are limited to a certain set of sanitized contract and product data. This is so we don't expose any confidential information from the unsanitized data about Collins Aerospace's contracts. This limits our ability to test our product to this smaller set of sanitized data rather than the full scale dataset that it our product will eventually handle.

1.7 Expected End Product and Deliverables

The final product will have two main parts and those are the products page and the contracts page. The first deliverable will be the products page and this is where the users can search for specific products and filter through them based on various data fields. The user will also be able to add a new product to the database and they will be required to fill out all of the necessary data fields to do so. The users will also be able to upload a csv file to create a new product, or modify an existing product.

The second deliverable will be the contracts page and this page is very similar to the products page but it deals with contracts. This means that users will go to this page to search for specific contracts and filter through them based on certain data fields. Users will also be able to add a new contract to the database manually or with a csv file just like the products page. When they add a new contract it will be required that all of the necessary information will be included and filled out.

For both of these deliverables the goal is to have a working prototype by the end of our first semester, and then a final version of the project by April of 2020.

2. Specifications and Analysis

2.1 Proposed Design

The proposed design decided upon for the front end came after some careful consideration. The product that the client needed called for a web application that could be deployed on their intranet for use to any of the required employees. Because this was going to be a web application, we decided to use ReactJS as well as ReactRouter to create this web application and be able to seamlessly link between the pages and tabs, as requested by the client. We believed that the use of React and its ability to create large and scalable web application with fast access and reloading of data was a good decision. The incorporation of ReactRouter to connect all of the pages we would need and link to separate pages was the perfect addition. Our plan is to get the basic pages laid out and add the connecting ability, and then work on diving deeper into the functionality of each of the pages and have them do what they are desired to do. Then be able to connect the front end of our project to the database of products that the back end has created for us.

The proposed design we decided to go with on the backend was using a layered approach. This is a style that is used by several of the developers on our team, and is something that our team is very familiar with, which will hopefully allow the team to work faster. The layers that we choose to go with was a simple, controller, service, and repository layout on our backend, using view models and models for objects during the transition. We went with this layout because we know that this a layout that is still being used in industry today, and is quite popular. This allows to hopefully make for an easy transition when we have to hand our project off to the client. As far as our progress on this proposed design, since this is a design that is quite popular with industry we have been able to really take that proposed design plan and build upon it. We already created all the different packages for each layer there is, and we have added several files to each package. Therefore, we are already underway on trying to implementing this design plan, meeting with the client, and confirming that this design plan with not only meet their requirements, but will best suited to match their expectations and needs.

2.2 Design Analysis

On the front end side of the design plan, so far we have made bits of progress getting actual code written to start developing the front end and the UI for the application. We have experience creating a web application, but we don't have as much knowledge into the framework and language that we chose to use. We have had to use some of our time to become more familiar with ReactJS, which is one of the things that could be considered a weakness of our proposed solution. We have the basic web pages set up, with linking capability between them as of now. We will need to continue our research of the framework as well as our continuous development of the project to ensure the best project for our client.

As I briefly touched on towards the end of the last section with the backend design. So far the initial design analysis on the backend has been a huge success. The design has worked with the issues that we are trying to solve and has allowed us to meet the requirements that the client has proposed. We have also have displayed our plan to the client, and they have like what we have so far. So we have designed our spring boot project, added in the packages for each layer in the layout design, and began getting files added to these packages. Since the design is new to several people on the team, we need to make sure to work with the team and the client to make sure everyone understands the design. That way we can make sure everyone is staying on the same design structure, and this will allow us to have a smooth handoff with the client since they will already know about the design we had proposed earlier to them.

2.3 Development Process

We are following an Agile process with elements of a SCRUM framework. We will be breaking up our tasks into sprints that are three weeks long, that way they coincide with our milestone layout. Each task that we have to get completed will be assigned a ticket, this will allow us to easily be able to keep track of the progress and work that each team member has done. The tickets will then be picked up with each sprint, and assigned to a developer. The developer then take the ticket, and work to get it completed. Once the assigned ticket is completed, they will open up a pull request to get reviewed by the team, and put their ticket into in review. After the code has been reviewed by a few members the developer may then add the ticket into master. Once the ticket has been added the developer will need to ensure that the ticket was completed properly after their code has been deployed to the server. Once the code has been deployed to the server, and everything looks good it will be added to our items for demo. We will have a demo with our client after each milestone completion to ensure that everything is completed and satisfactory to what the client is wanting.

2.4 Design Plan

Because of the limited user base for our finished project, we were able to focus our design plan for Product TRACkER very specifically to Collins Aerospace's needs. Our primary needs are a user-friendly interface and a stable connection from that interface to a database that stores all of the necessary data. Usually security would also be a main concern, but that will be handled on our client's intranet. As depicted in the following diagrams that detail our proposed project structure, we chose to use a React framework to build the user interface, and a Spring Boot Application to handle the backend connection to our MySQL database.

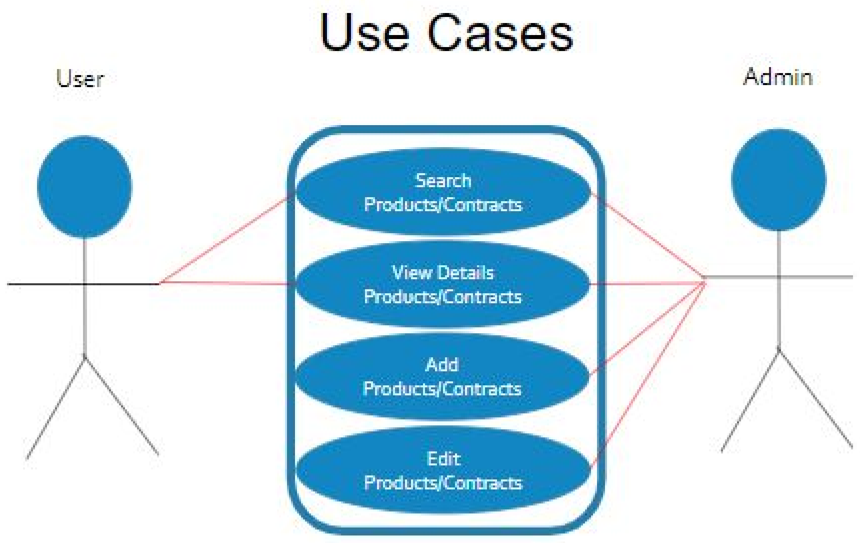


Figure 1. Use Case Diagram

In figure 1 one of the biggest ideas that we were trying to depict when coming up with this diagram was the authorization that we would have in our application. One of the items that was requested by Collins Aerospace was that we have both an admin user and a basic user. The reason they wanted this in the application was because they did not just anybody to be going in and adding, editing, or removing any of their contracts or products. So in order to ensure we knew what was going on correctly we presented Collins with this basic diagram, that not only showed that we understood how the authorization would work, but also provided us with documentation that we could fall back on in case we forgot what we were going for. Additionally it allowed us to confirm one last time with the client before we began development. The big idea we ended up settling on which can be seen in the diagram is the basic user is able to search and view products and contracts, whereas the admin can do that and manage the admins, add products or contracts, and edit products or contracts.

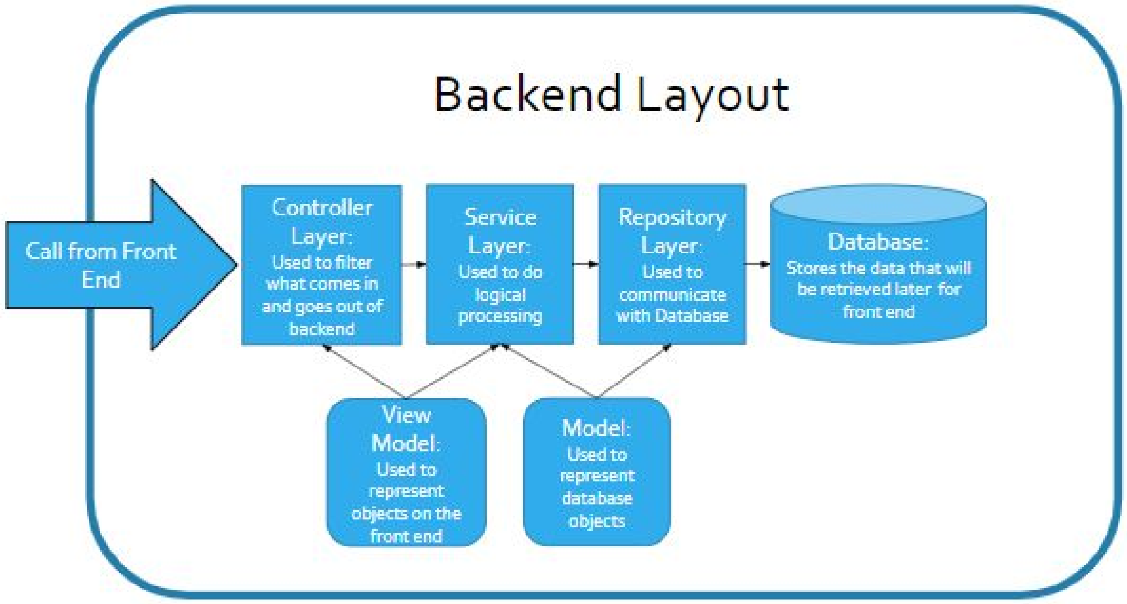

Backend Layout
Call from Front End
Controller Layer: Used to filter what comes in and goes out of backend
Service Layer: Used to do logical processing
Repository Layer: Used to communicate with Database
Database: Stores the data that will be retrieved later for front end
View Model: Used to represent objects on the front end
Model: Used to represent database objects

Figure 2. Backend Layout Diagram

In figure 2 we laid out what our backend layout will look like. One of the biggest goals that was wanted from the client and we were trying to meet with this backend layout was the maintainability of our code. Since we know that we will not be around to help if there are any issues with the code after we hand it off to Collins Aerospace we tried to develop a layout that is easy to follow, and hopefully easy to debug should it run into any issues when we are not around. In order to ensure that we could provide Collins with a very maintainable backend we went with a backend layout that is very popular in industry. That way even though we are gone they will still recognize the different layers to our backend and know where to find the issue if one may arise. The big idea to our design is that when a request comes into our backend it will be sent into the controller layer in the form of view model. In the controller layer it will process to make sure it is receiving the correct input, after which case it will pass the viewmodel over to the service layer which will do all the logic processing and convert it into database objects or models. These models then will get passed to the repository layer. In the repository layer it will prep the data to be loaded into the database, and then load it into the correct table. Now when sending a request back it will do this whole path except in reverse order.

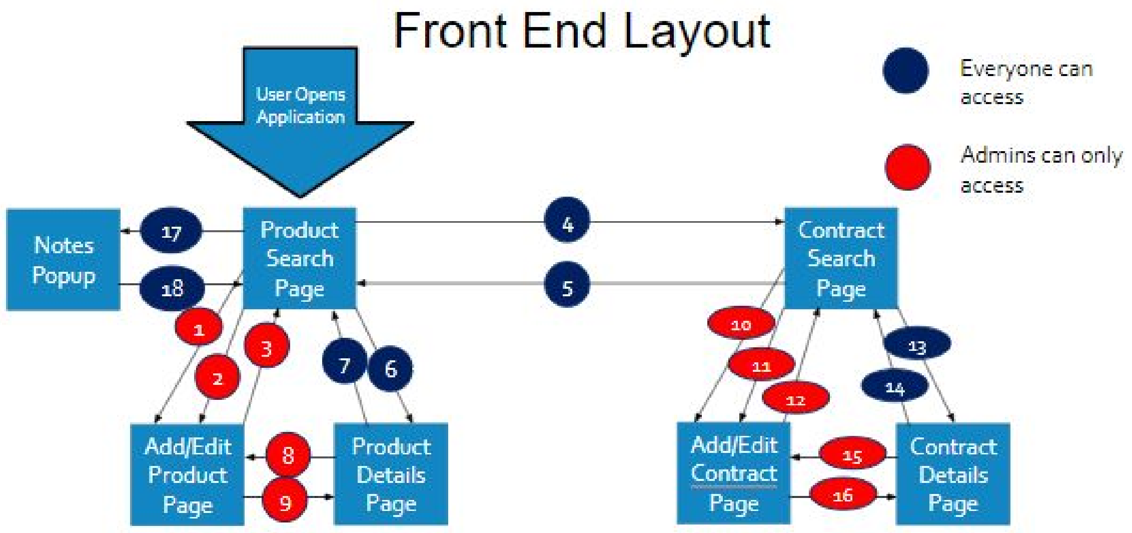
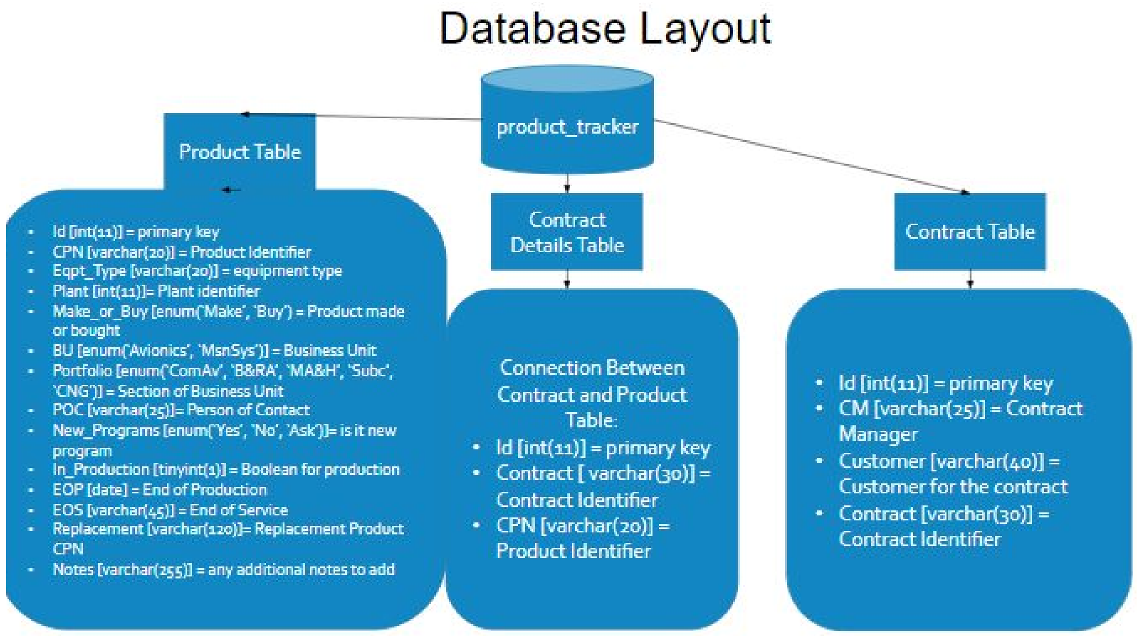


Figure 3. Front End Layout Diagram

In figure 3 we sketched up the diagram that shows how the front end will be laid out. For the front end one of the biggest requests that we received was that they wanted a very user friendly website for their team to work with. They wanted to make sure that anybody could look at the website and know how to navigate. We added in the screen sketches that our team created in the next section. However one other important part to a user friendly UI is that the transitions need to be smooth, and make sense to any basic user. So we worked on this for quite a while to make sure that we developed a design that our whole team was happy with, and then described this design to the client. When showing it to our client they were impressed and liked the design that we set forth. The design we came up with is laid out above. The big idea of how this design works is as follows, when the user first logs in they will be taken to our product search page. Once on the product search page, they are able to either navigate to the contract search page or select to view the details of a product, look at its notes, or add or edit a product. If the user decides to navigate to the contract search page they are able to view the details of a contract, or add or edit a contract. Two other features that we added into this diagram is the numbers in the circles on the lines, and the colors of those circles. Well the circle colors denote which user role has permission to perform that action. If the circle is red this means you need to be an admin, if the circle is blue, this means that either the admin or basic user can do this. Now as far as the numbers in the circle, each circle represents some form of navigation on the website. The numbers correspond to way to use the transition. So in order to understand how to transition you will have to compare that number with the page transitions that we have listed here:

1. Click the add button to insert a new product
2. Click on a product and clicking the edit button to edit the product
3. Click the x on the add/edit popup to go back to search page
4. Clicking on the Contract Tab to bring you to the contract search page
5. Clicking on the Product Tab to bring you to the product search page

6. Clicking on the CPN of the Product to bring up the products detail page
7. Clicking on the x to take you back to the product search page
8. Clicking on the Edit Button on the Product Details Page
9. Clicking the x on the edit popup to take you back to the product details page
10. Click the add button to insert a new contract
11. Click on a contract and clicking the edit button to edit the contract
12. Click the x on the add/edit popup to go back to search page
13. Clicking on the Contract Number of the Contract to bring up the contract detail page
14. Clicking on the x to take you back to the contract search page
15. Clicking on the Edit Button on the Contract Details Page
16. Clicking the x on the edit popup to take you back to the contract details page
17. Open the notes on a product by clicking notepad on the side of the listing
18. Clicking the x to go back the Product Search Page



Database Layout
product_tracker
Product Table
Id [int(11)] = primary key
CPN [varchar(20)] = Product Identifier
Eqpt_Type [varchar(20)] = equipment type
Plant [int(11)]= Plant identifier
Make_or_Buy [enum('Make', 'Buy') = Product made or bought
BU [enum('Avionics', 'MsnSys')] = Business Unit
Portfolio [enum('ComAv', 'B&RA', 'MA&H', 'Subc', 'CNG')] = Section of Business Unit
POC [varchar(25)]= Person of Contact
New_Programs [enum('Yes', 'No', 'Ask')]= is it new program
In_Production [tinyint(1)] = Boolean for production
EOP [date] = End of Production
EOS [varchar(45)] = End of Service
Replacement [varchar(120)]= Replacement Product CPN
Notes [varchar(255)] = any additional notes to add
Contract Details Table
Connection Between Contract and Product Table:
Id [int(11)] = primary key
Contract [varchar(30)] = Contract Identifier
CPN [varchar(20)] = Product Identifier
Contract Table
Id [int(11)] = primary key
CM [varchar(25)] = Contract Manager
Customer [varchar(40)] = Customer for the contract
Contract [varchar(30)] = Contract Identifier

Figure 4. Database Layout Diagram

In figure 4 we are demonstrating what our database layout turned out to be. The big item that we were trying to ensure here was that we were meeting all the clients data requirements. When we first started this project one of the items that the client requested of us was to have a certain information stored about both the contract and the products that we have in our database. So, we wanted to ensure that all the data we had was present. Therefore a lot of the information that you see in the product table was the information that the client requested to be stored on the product, and the same goes for the contract table, all this information were items that the client requested to be stored on the client. As far as the third table that was created this was to ensure we could meet one of the requests asked by the client. In this request the client asked if we could also draw a connection between what contracts had what products and what products had what contracts. In order to do this we came up with the best way we could think of which was to have a seperate table which we called contract details. In this table we will store an identifier of both a contract and product. This in essence will allow us to draw the link between the contracts and the products table, which is one of the use cases that the client asked us to find a way to solve, creating the contract details table.

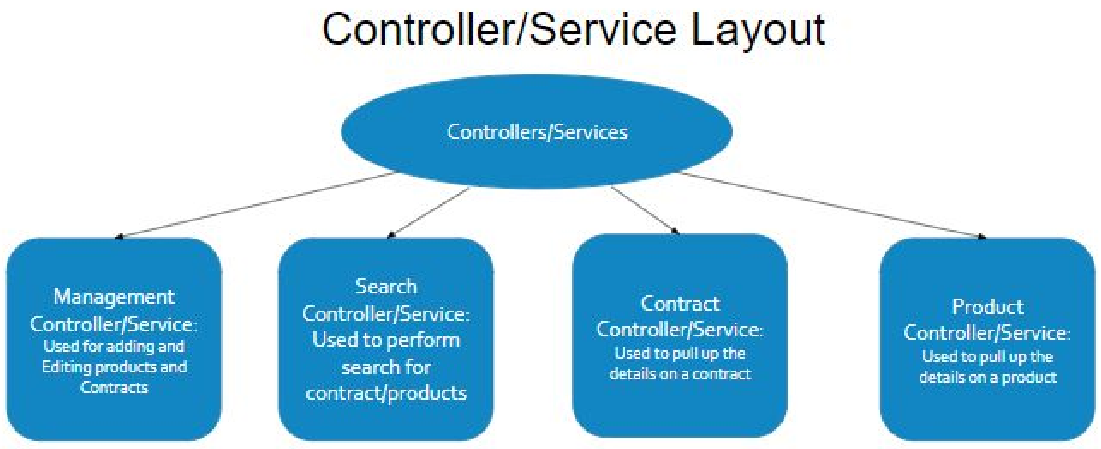

Controller/Service Layout
Controllers/Services
Management Controller/Service: Used for adding and Editing products and Contracts
Search Controller/Service: Used to perform search for contract/products
Contract Controller/Service: Used to pull up the details on a contract
Product Controller/Service: Used to pull up the details on a product

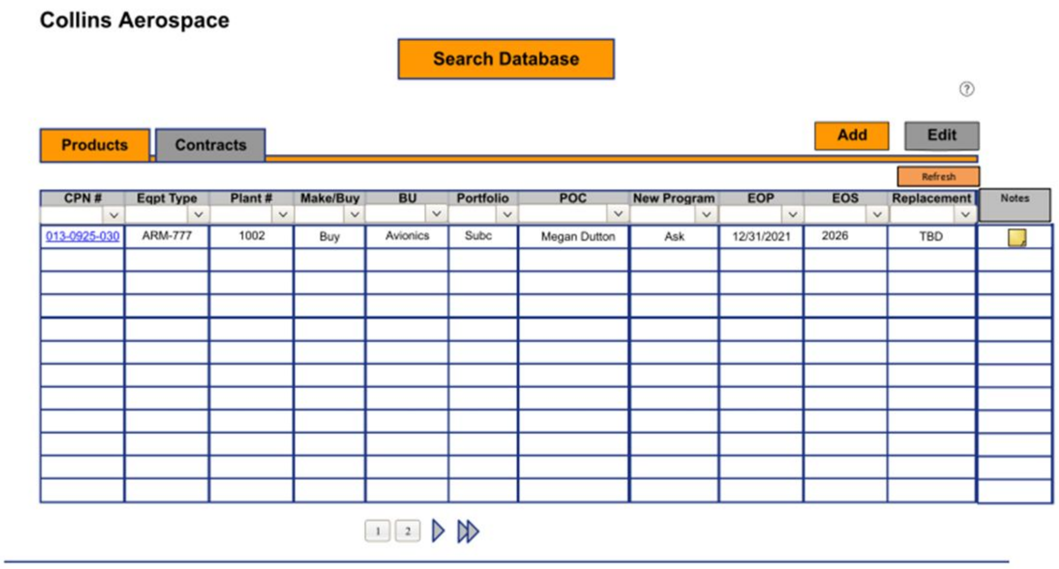
Figure 5. Controller/Service Layout Diagram

The last diagram that was designed for our design plan, shown in figure 5, was the controller and service layout diagram. In this diagram the goal we were trying to meet here was furthering the idea of writing easily maintainable code. Again we wanted to ensure that even when we were no longer working on this project, that the developers at Collins Aerospace would know how to not only fix bugs should they arise, but also be able to add new features to the tool when they are needed. In order to make sure they knew how to find the code the developers may be looking for, we presented the clients with the above diagram. The idea behind this diagram was it would show the client where to find each feature in relation to the controller and service layers. So, for example the management controller/service has our add and editing of data features, the search controller/service has the filtering or search features, the contract and product controllers/services have the details for the contracts and products respectively. This would not only help the client know where to locate a feature in our controller and service layer, but it will also allow us as developers to stay on the same page and make sure that we aren't all creating our own service or controller. This will help make sure that our code stays clean, concise, on the same page, and make sense with the client. Again when we presented this design to the client they were impressed, and like the direction that we were heading.

As I mentioned above all of these designs were presented to the client, and they were very impressed with the work and the designs that we came up. Therefore, we will confidently be moving forward with these designs since we know that the client approved, and a lot of the use cases that the client presented us with were able to be solved with these designs that we came up with.

2.4.1 Screen Sketches

One item that was important to our use cases for our project was the idea of having a very user friendly UI. In order to do this we ran through several different scenarios as a team, and developed the final screen sketches below. These sketches we ran through with the client, and we were given to move forward with the screen sketches that are presented here:



CPN #	Eqpt Type	Plant #	Make/Buy	BU	Portfolio	POC	New Program	EOP	EOS	Replacement	Notes
013-0925-030	ARM-777	1002	Buy	Avionics	Subc	Megan Dutton	Ask	12/31/2021	2026	TBD	

Figure 6. Products Page Sketch

Figure 6 shows the page that will be used to search and filter through all of the products within the database. The page will have buttons at the top to switch back and forth between the product search page and the contract search page. There will also be buttons to add a new product or edit and existing one. The page will be structured with a grid-like layout and when a product matches the search it will fill in a row of the grid. Users will be able to type into the data fields at the top of each column and the page will start a search and return products that match completely or partially.

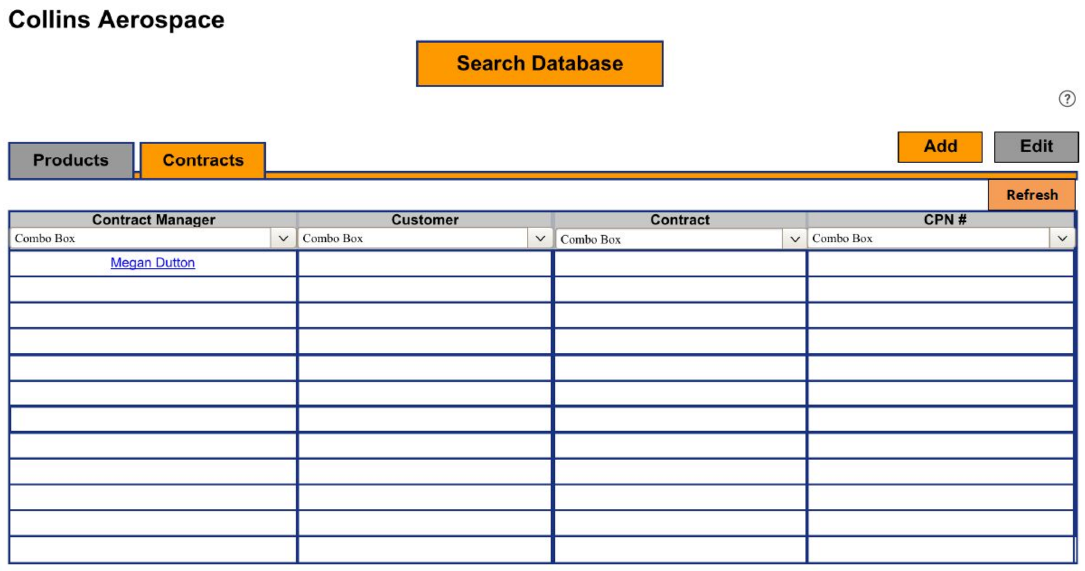


Figure 7. Contracts Page Sketch

Figure 7 shows the page that will be used to search and filter through contracts within the database. This page will be one of the main pages that users will see when they start the product. It will have buttons to switch back and forth between the product search page and the contract search page. It will have buttons to add a new contract or edit an existing one. The page will structured in a grid-like layout where the contract that match the search will fill in a row. Users will be able to type into a data field at the top of each column and the page will start searching the database and return any partial or complete matches.

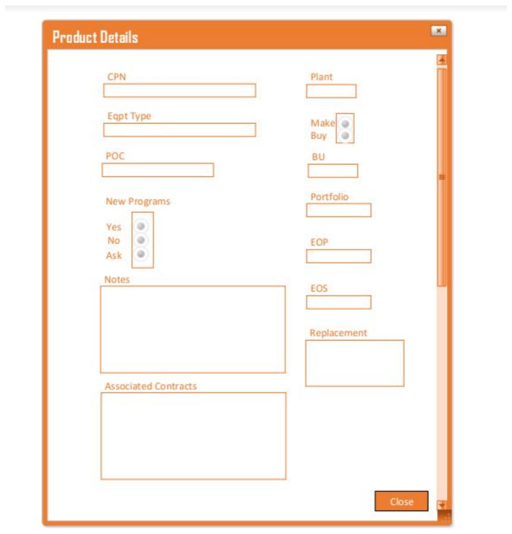


Figure 8. Product Details Page Sketch

In Figure 8 the Product details page is shown and it will be a pop-up window page that will quickly and easily display the information that is associated with the product. The page will consist of data fields and boxes that the information will be displayed in, and there will be a button on the bottom of the screen to exit the page.

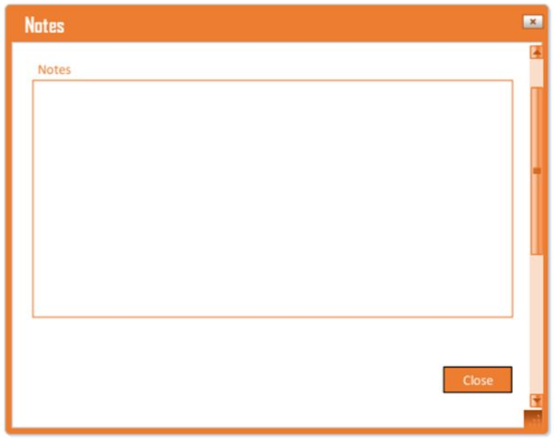


Figure 9. Notes Page Sketch

Figure 9 shows the notes page, and this page is used to have any special comments about a product or contract. This page will be accessed from the product and contract search pages and it will be a pop-up window. The page will consist of a text box and a button at the bottom to exit the page.

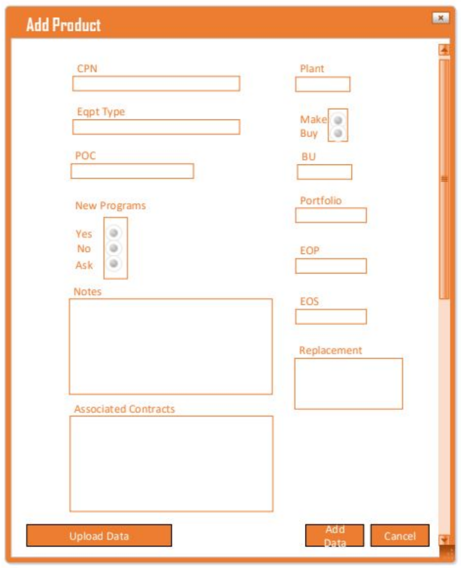


Figure 10. Add Product Page Sketch

Figure 10 shows the page that will be used to add a product to the database. This page will be a pop-up window and it will consist of data fields that the user can fill in. The page will also have three buttons at the bottom. One button will be used to exit the page, one button will be used to add the product to the database, and the last button will be used to upload and fill in the data fields via a csv file.

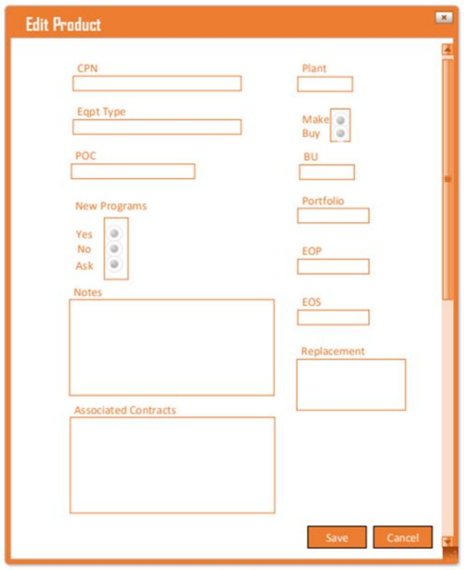


Figure 11. Edit Product Page Sketch

Figure 11 shows the page that will be used to edit a product that already exists within the database. This page will be a pop-up window that displays the information that is associated with the product already. The page will look very similar to the "Product Add Page" having the same data fields, but the button at the bottom will update the product in the database rather than create a brand new entry.

3. Statement of Work

3.1 Previous Work And Literature

This project is about tracking details for contracts and products within Collins Aerospace inventory and system. This project is unique to Collins and it is not meant to be applied to any other businesses or companies. This makes it difficult to compare to any other similar projects on the market right now because as previously stated this project is unique to only Collins Aerospace. This project is brand new so there will be no following of previous work.

In order to thoroughly understand the end use of our product, we held several meetings with our client at Collins Aerospace. We worked through a sanitized dataset to learn the relationships between different components in the data set, and developed several user stories in order to understand how the data will be used in the end product by typical users.

3.2 Placing Product In Related Work And Literature

Although unique in several ways, there are many products that use similar ideas as this one. For example there are many products that compile tables of data to be easily displayed, filtered, sorted and customized in such a way that it's easily accessible and readable. That's exactly what Product Tracker does. Like others, Product Tracker takes data that's not easily accessible or readable and organizes it for the User and displays it; Offering many features to allow the data to be customizable. In many ways products like Ebay, Amazon and other shopping web apps are very similar to Product Tracker. In the sense that you are searching for data, data is returned from a database, given attributes and displayed for the user. Although the formatting is different, the ideas are still the same.

3.2 Technology Considerations

For the front-end development we had to decide which framework to use and we narrowed down our options to React and Angular.

React Pros: Better user experience, faster development, Time-saving, and code stability

Angular Pros: Cleaner code, High performance and low app size, and better error handling

For us we went with React because it seemed to make more sense for our project. It also has a much easier learning curve which is important given that we only have two semesters to work on this project.

For the back end development we decided we wanted to go with something that would also us to use Java on the backend. The reason we wanted to use Java on the backend is because this is the language that most of us have worked with on a day to day, and is something that we all are familiar with. Several members of the team suggested that we take a look at SpringBoot since this was a framework that they had used before. Therefore we decided to move forward with SpringBoot since it had a smaller learning curve being we have worked with Java before. It also would allow us to have experts on the team by choosing SpringBoot.

For the Database we wanted to go with one that was easy and cost effective for us to work with. Since a lot only worked with SQL queries we wanted to use a database that used SQL. Several of our group members had used MySQL before. Therefore we went with MySQL for much the same reason we choose SpringBoot, because it provided us with a query language we were familiar with and a system we were familiar with.

3.3 Task Decomposition

We began by splitting the work in half. This was separated into frontend and backend. This way we could easier focus each person's strengths and into a more focused area. We then decomposed each of these halves into more specific tasks.

For the front end, we began coming up with an overview design for the application. After a few failed designs, we came up with a final design idea. This was done after first creating the flow of the application, then by deciding the number of forms/pages needed to be made and finally by determining an optimal layout for each page. From there we needed to put this design into visual form. We broke down the tasks of design into unique window screen sketches. Once we had all the sketches created it was time to develop. The development process was decomposed into tasks that would support a layered, modular design. This way each person that takes part in the development process will be able to do so without worry of affecting others work. The tasks composed for the front end are:

1. Create Product Add Page
2. Create Contract Add Page
3. Create Product Search Page
4. Create Contract Search Page
5. Create Product Details Page
6. Create Contract Details Page
7. Create Product Edit Page
8. Create Contract Edit Page
9. Create Admin Login Prompt
10. Create Admin User Management Page

For the backend, we took a bit of time to analyze the requirements and make sure that we developed something that would cover every requirement. After we developed the tasks for the backend we had to lay them out in such a way that it would not affect the tasks that were before it. Ultimately after working with the backend team we decided that the following tasks needed to be done, and relatively following this ordering:

1. Create an excel file (.csv) multi product data upload endpoint
2. Create an excel file (.csv) multi contract data upload endpoint
3. Create a form search endpoint for filtering for product search
4. Create a form search endpoint for filtering for contract search
5. Create an endpoint to retrieve product details
6. Create an endpoint to retrieve contract details
7. Create an endpoint to edit product details
8. Create an endpoint to edit contract details
9. Add in admin users table
10. Create endpoint to add/edit admin users
11. Create an endpoint to delete admin users
12. Restrict access on the admin endpoints

13. Implement tests on the application

This is the foundation of our tasks. However, when we delve into each tasks subtasks may break out, and we change up the priority of the tasks, if the priority of the requirements change.

3.4 Possible Risks And Risk Management

Our team developed the following risks that we foresee running into during the development of product tracker. First, our team is relatively new, there is a mix of people who know each other, and there are also new members on our team. Therefore we will have to work together to learn each others communication styles, and make sure that we work to prevent any and all issues among our team. We want to make sure everyone is onboard with our development plan, and everybody is able to have their voices heard. The next risk big risk that our team came up with as being a possible issue is the new technologies and development structures that we are using. Several members on the team have not gotten much experience with items such as the React Js Framework, this is new to a lot of people on the team. So our team has had to set aside time in the development process for members of the front end team to be caught up on this process. Another item that is new to our team is structure of our back end and how all the components work together. This has needed a bit of explaining and will continue to need a bit of explaining to make sure that all team members understand the layered layout approach, that way we can ensure that every member of our team knows what is going on. Additionally to ensure that our code is clean, and easy to follow with every team member following the same design layout.

3.5 Project Proposed Milestones and Evaluation Criteria

After working with the team we have come up with a few different milestones. The milestones we have selected were easy benchmarks to show our progress and an easy way for the client and ourselves to ensure that we stay on track. The way we picked them was trying to find the core parts for our project, and we decided that the completion showed a core part of requirements were complete, and kept both our front end and back end teams on track and moving at the same speed. Here are the milestones we came up with:

1. Initial Framework Setup and Deployed
2. Product and Contract Data Add Pages and Endpoints Completed
3. Product and Contract Search Pages and Endpoints Completed
4. Product and Contract Details Pages and Endpoints Completed
5. Product and Contract Edit Pages and Endpoints Completed
6. Admin Authentication and Page Restriction Endpoints and Pages Completed
7. Product TRACkER test cases front end back end written and passed
8. UI and Feature Enhancements/Final Turnins

In order to make sure that we will stay on track, and make sure that these milestones are met we came up with the following ways to test and ensure our milestones are met:

1. Confirm that the database, Spring boot, is able to boot up, and be deployed onto our server
2. Confirm that our add pages meets all our clients needs during the demonstration
3. Confirm that our search pages meets all our clients needs during the demonstration
4. Confirm that our detail pages meets all our clients needs during the demonstration
5. Confirm that our edit pages meets all our clients needs during the demonstration
6. Confirm that a user who was given admin permissions is able to add and edit data and manage admin users and that the user
7. Confirm that we have 100 percent or close to 100 percent code coverage test cases and our application pasts all tests
8. Make sure that any features that the client wanted added or changed get done, and that all items are turned in and able to be handed off to the client smoothly

3.6 Project Tracking Procedures

Our team is using the GitLab Issues board to manage tasks and track our progress. The process that each ticket will undergo is as follows. First the ticket will be created and added into our backlog, based on when the team decides it will be needed to be completed it will be assigned to one of our three week sprints. The item will be assigned to a sprint based on what milestone it is under. Then we will go ahead and assign that ticket to a team. From here we will be able to check on each team members progress as we are able to see they ticket move from in progress, in review, and closed. Once the ticket is completed it will undergo the demo with corresponding milestone to ensure that the client approves of the work we have done. We will also have our Gantt that we will use for recording and updating our progress so the client and teammates can know the latest on our progress and how we are staying on track.

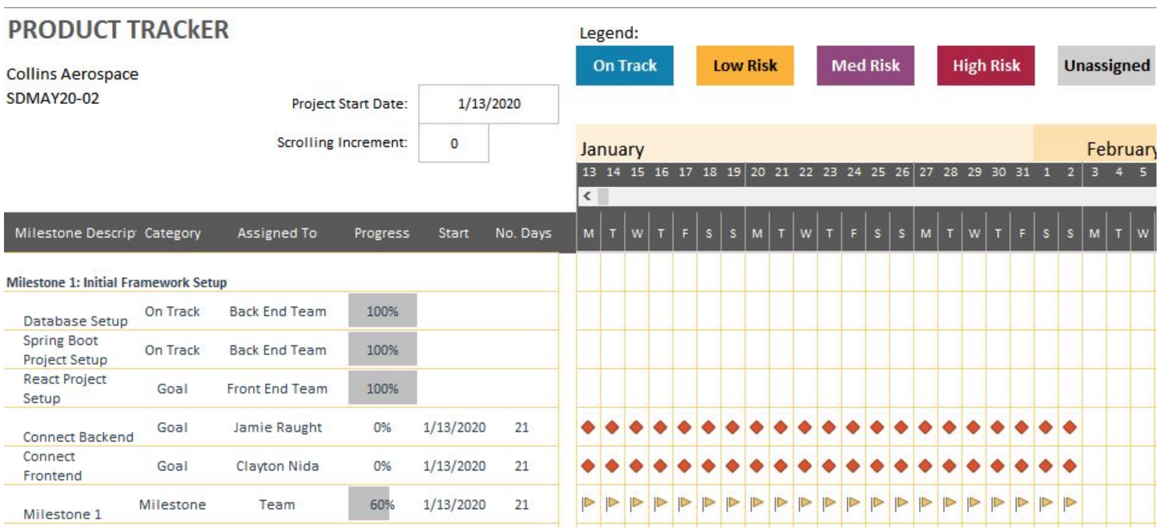
3.7 Expected Results and Validation

Our desired outcome is a web application that completely meets customer requirements and needs. Before product delivery, our software shall pass one-hundred percent of our test cases. At a higher level, we plan to have regular customer demonstrations throughout the development process, and this will allow us to integrate frequent feedback from our client on our solutions. This tool will allow will allow the client to be able to lookup their contracts and products, and to also be able to see the connections between the contracts and products. This will make it easier for Collins employees to ensure that their contracts and business pursuits will be the best fit for both their teams and their customers.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

The Gantt chart and schedule we came up with was as follows:



PRODUCT TRACKER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 0

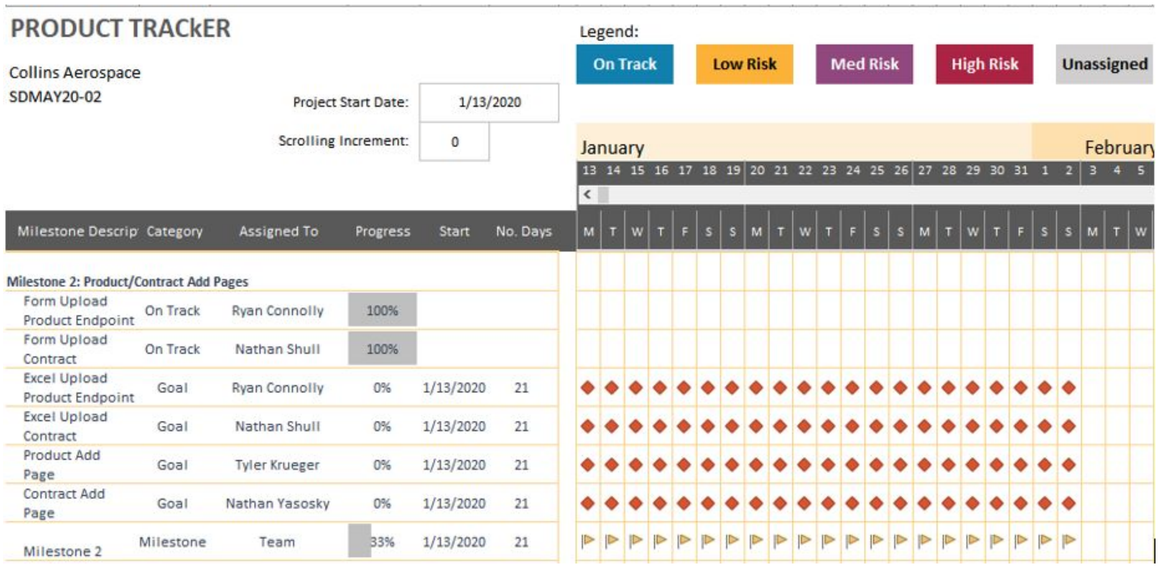
Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

January / February

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	13 M	14 T	15 W	16 T	17 F	18 S	19 S	20 M	21 T	22 W	23 T	24 F	25 S	26 S	27 M	28 T	29 W	30 T	31 F	1 S	2 S	3 M	4 T	5 W
Milestone 1: Initial Framework Setup																													
Database Setup	On Track	Back End Team	100%																										
Spring Boot Project Setup	On Track	Back End Team	100%																										
React Project Setup	Goal	Front End Team	100%																										
Connect Backend	Goal	Jamie Raught	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Connect Frontend	Goal	Clayton Nida	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 1	Milestone	Team	60%	1/13/2020	21	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑			

Figure 12: Milestone 1 Gantt/Timeline/Schedule

Figure 12 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 1: Initial Framework Setup. The description for how it all works and was decided upon is below.



PRODUCT TRACkER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 0

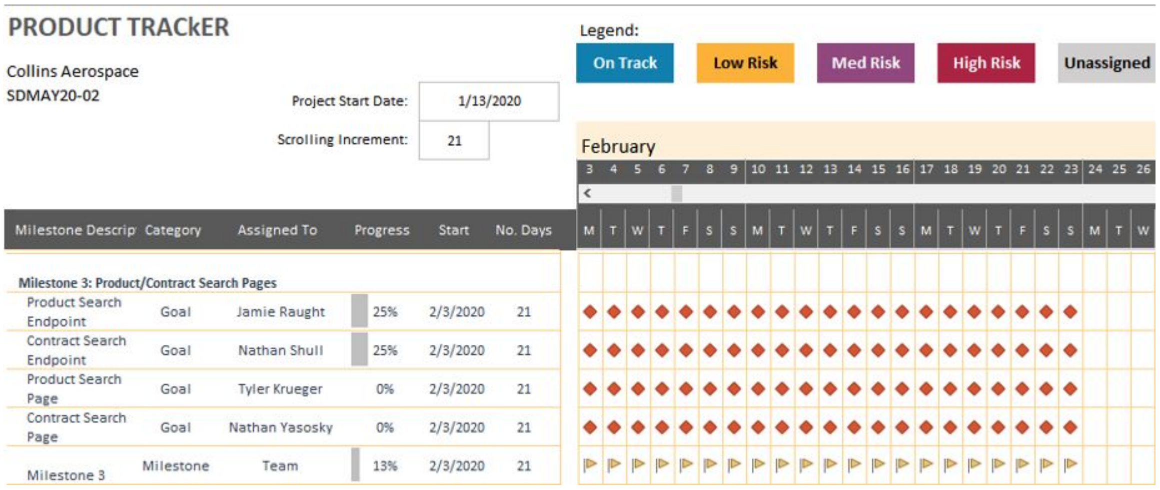
Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

January February

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	13 M	14 T	15 W	16 T	17 F	18 S	19 S	20 M	21 T	22 W	23 T	24 F	25 S	26 S	27 M	28 T	29 W	30 T	31 F	1 S	2 S	3 M	4 T	5 W
Milestone 2: Product/Contract Add Pages																													
Form Upload Product Endpoint	On Track	Ryan Connolly	100%																										
Form Upload Contract	On Track	Nathan Shull	100%																										
Excel Upload Product Endpoint	Goal	Ryan Connolly	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Excel Upload Contract	Goal	Nathan Shull	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Product Add Page	Goal	Tyler Krueger	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Contract Add Page	Goal	Nathan Yasosky	0%	1/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 2	Milestone	Team	33%	1/13/2020	21	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷			

Figure 13: Milestone 2 Gantt/Timeline/Schedule

Figure 13 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 2: Product/Contract Add Pages. The description for how it all works and was decided upon is below.



PRODUCT TRACkER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 21

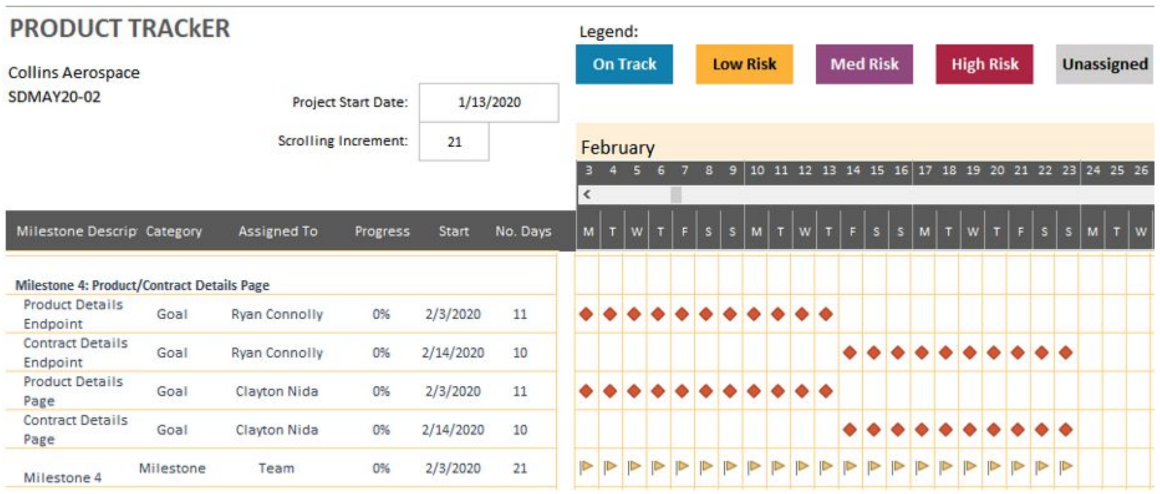
Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

February

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	3 M	4 T	5 W	6 T	7 F	8 S	9 S	10 M	11 T	12 W	13 T	14 F	15 S	16 S	17 M	18 T	19 W	20 T	21 F	22 S	23 S	24 M	25 T	26 W
Milestone 3: Product/Contract Search Pages																													
Product Search Endpoint	Goal	Jamie Raught	25%	2/3/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Contract Search Endpoint	Goal	Nathan Shull	25%	2/3/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Product Search Page	Goal	Tyler Krueger	0%	2/3/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Contract Search Page	Goal	Nathan Yasosky	0%	2/3/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 3	Milestone	Team	13%	2/3/2020	21	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷	▷			

Figure 14: Milestone 3 Gantt/Timeline/Schedule

Figure 14 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 3: Product/Contract Search Pages. The description for how it all works and was decided upon is below.


PRODUCT TRACkER

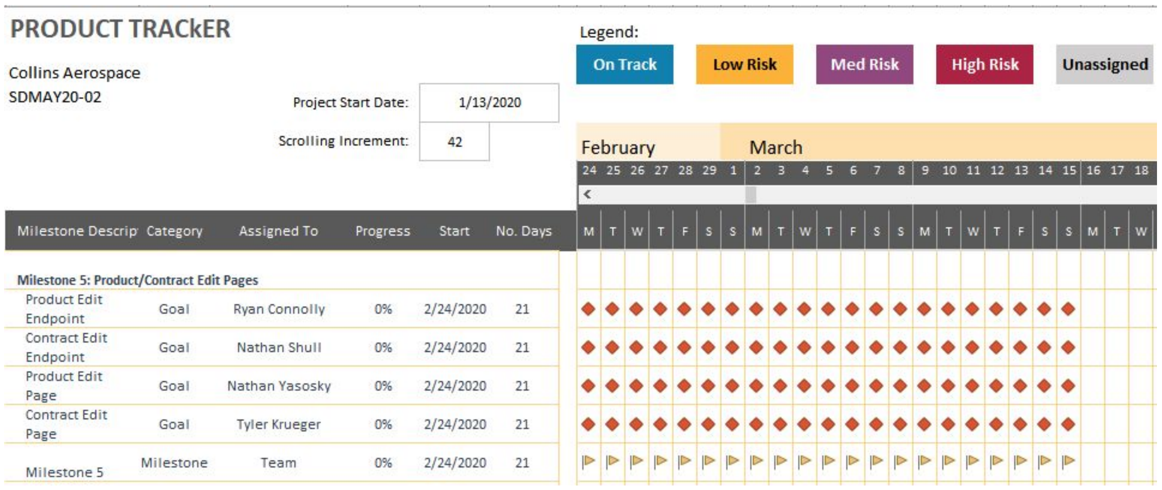
Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 21

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days
Milestone 4: Product/Contract Details Page					
Product Details Endpoint	Goal	Ryan Connolly	0%	2/3/2020	11
Contract Details Endpoint	Goal	Ryan Connolly	0%	2/14/2020	10
Product Details Page	Goal	Clayton Nida	0%	2/3/2020	11
Contract Details Page	Goal	Clayton Nida	0%	2/14/2020	10
Milestone 4	Milestone	Team	0%	2/3/2020	21

Figure 15: Milestone 4 Gantt/Timeline/Schedule

Figure 15 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 4: Product/Contract Details Pages. The description for how it all works and was decided upon is below.


PRODUCT TRACkER

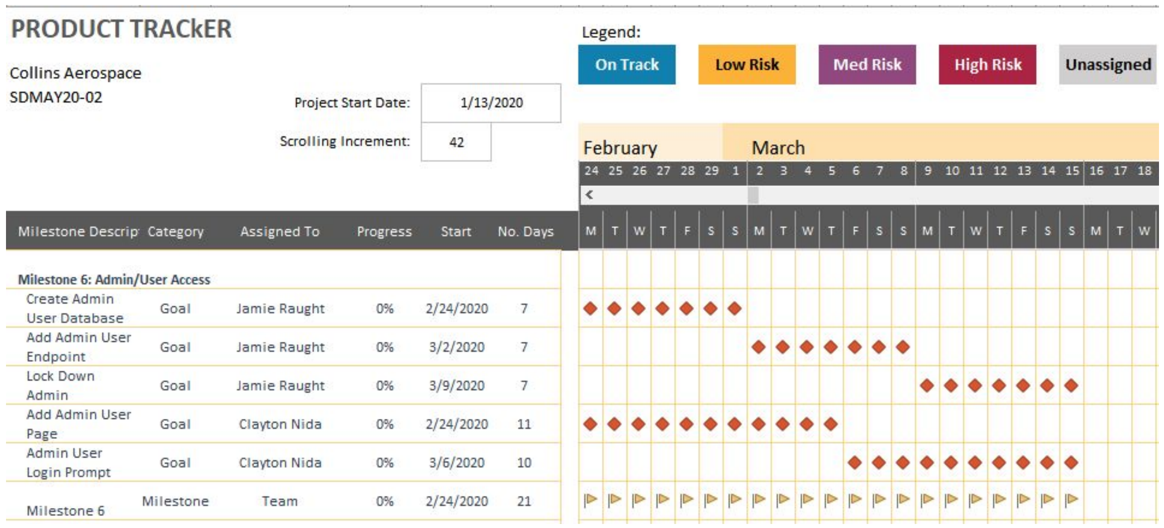
Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 42

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days
Milestone 5: Product/Contract Edit Pages					
Product Edit Endpoint	Goal	Ryan Connolly	0%	2/24/2020	21
Contract Edit Endpoint	Goal	Nathan Shull	0%	2/24/2020	21
Product Edit Page	Goal	Nathan Yasosky	0%	2/24/2020	21
Contract Edit Page	Goal	Tyler Krueger	0%	2/24/2020	21
Milestone 5	Milestone	Team	0%	2/24/2020	21

Figure 16: Milestone 5 Gantt/Timeline/Schedule

Figure 16 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 5: Product/Contract Edit Pages. The description for how it all works and was decided upon is below.


PRODUCT TRACkER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020

Scrolling Increment: 42

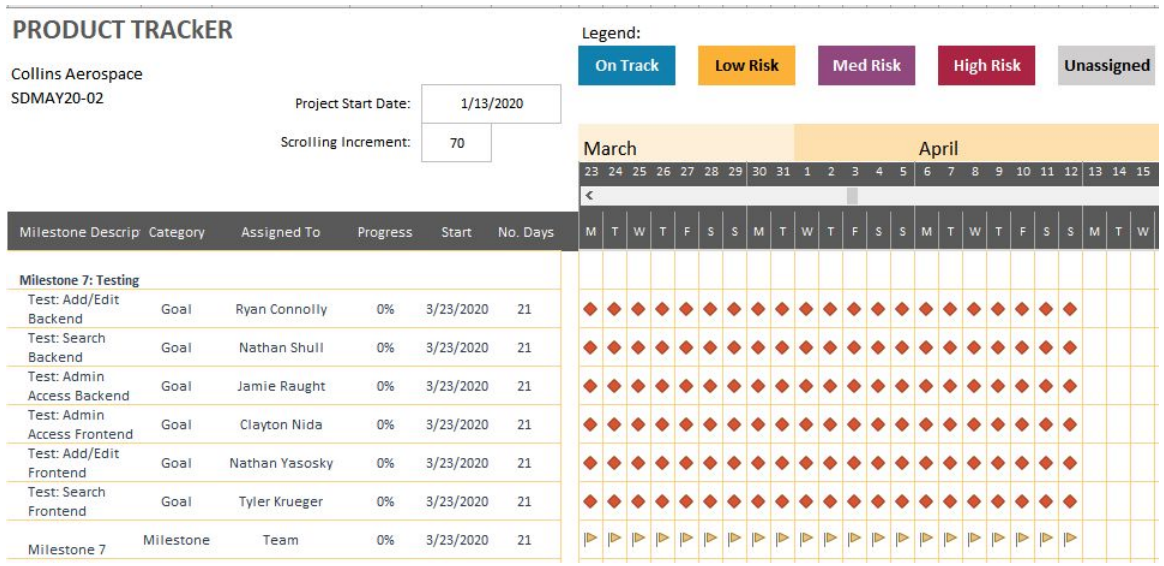
Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

February | March

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	24	25	26	27	28	29	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
						M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W
Milestone 6: Admin/User Access																													
Create Admin User Database	Goal	Jamie Raught	0%	2/24/2020	7	◆	◆	◆	◆	◆	◆	◆																	
Add Admin User Endpoint	Goal	Jamie Raught	0%	3/2/2020	7								◆	◆	◆	◆	◆	◆	◆										
Lock Down Admin	Goal	Jamie Raught	0%	3/9/2020	7															◆	◆	◆	◆	◆	◆	◆			
Add Admin User Page	Goal	Clayton Nida	0%	2/24/2020	11	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆													
Admin User Login Prompt	Goal	Clayton Nida	0%	3/6/2020	10												◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 6	Milestone	Team	0%	2/24/2020	21	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑			

Figure 17: Milestone 6 Gantt/Timeline/Schedule

Figure 17 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 6: Admin/User Access. The description for how it all works and was decided upon is below.


PRODUCT TRACkER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020

Scrolling Increment: 70

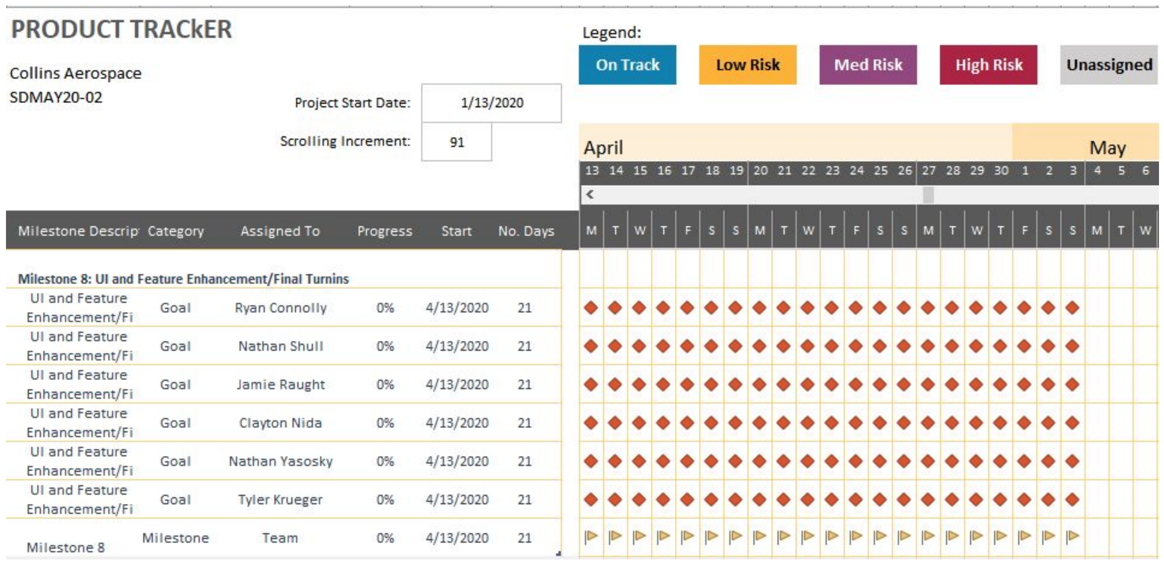
Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

March | April

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
						M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W
Milestone 7: Testing																													
Test: Add/Edit Backend	Goal	Ryan Connolly	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Test: Search Backend	Goal	Nathan Shull	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Test: Admin Access Backend	Goal	Jamie Raught	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Test: Admin Access Frontend	Goal	Clayton Nida	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Test: Add/Edit Frontend	Goal	Nathan Yasosky	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Test: Search Frontend	Goal	Tyler Krueger	0%	3/23/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 7	Milestone	Team	0%	3/23/2020	21	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑			

Figure 18: Milestone 7 Gantt/Timeline/Schedule

Figure 18 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 7: Testing. The description for how it all works and was decided upon is below.


PRODUCT TRACkER

Collins Aerospace
SDMAY20-02

Project Start Date: 1/13/2020
Scrolling Increment: 91

Legend: On Track | Low Risk | Med Risk | High Risk | Unassigned

April | May

Milestone Descrip	Category	Assigned To	Progress	Start	No. Days	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6
						M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W
Milestone 8: UI and Feature Enhancement/Final Turnins																													
UI and Feature Enhancement/Fi	Goal	Ryan Connolly	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
UI and Feature Enhancement/Fi	Goal	Nathan Shull	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
UI and Feature Enhancement/Fi	Goal	Jamie Raught	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
UI and Feature Enhancement/Fi	Goal	Clayton Nida	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
UI and Feature Enhancement/Fi	Goal	Nathan Yasosky	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
UI and Feature Enhancement/Fi	Goal	Tyler Krueger	0%	4/13/2020	21	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆	◆			
Milestone 8	Milestone	Team	0%	4/13/2020	21	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑	⚑			

Figure 19: Milestone 8 Gantt/Timeline/Schedule

Figure 19 is the timeline and gantt that we came up with for tasks that we felt fit under the milestone 8: UI and Feature Enhancement/Final Turnins. The description for how it all works and was decided upon is below.

For the timeline, we took the divided tasks mentioned in 3.3 and categorized them into milestones, this allowed us to create goals for different phases of the project. With these milestones, we were able to supply a time for each of the tasks associated with it's parent milestone. This will help the team to get a better understanding for what needs to be done, how long it should take and see what tasks should already be completed. Not only will this timeline give better foresight to the team, but it will help the team remain on target, as they now know the specific tasks that they need to accomplish. Lastly, the creation of this chart will help organize what tasks should be included in each of our sprints, which only furthers our efficiency.

Each of our milestones were laid out in such a way, so that the previous milestone should be nearly resolved before its latter. As you can see in the chart, the milestones chosen are as follows:

- Initial Framework Setup
  - What needs to be done to set the foundation for the product, including: The database, UI and back-to-front connections
- Product/Contract Add Pages
  - These are the pages required to upload or create data that will be used within the product
- Product/Contract Details Pages
  - These are the pages that will display details about a selected product or contract
- Product/Contract Edit Pages
  - These are the pages that will allow an admin to edit existing data within the product
- Admin/User Access
  - Where the differences of usability between an Admin and a User is distinguished

- Testing
    - Where the products features are tested
- UI and Feature Enhancement/Final Turnins
    - Where the product is polished and final (minor) improvements are made before it is delivered

The chart includes a column that tracks the current progress of each of these milestones and its goals. Those that are at 0% are marked as a goal and as such will be for later stages of the project, others will take time and dependencies into account and a risk will be assigned. Each and every milestone contains a task that is for later stages of the project, and save Testing, only the first three milestones contain tasks for the current version of the project.

4.2 Feasibility Assessment

Product TRACkER will be a site used by the client that will show information about contracts and products within the client's database and system. Administrative users will be able to add contracts and products via specific add pages. They will also be able to edit information that is associated with products and contracts via specific edit pages. All users will be able to look up any product or contract within the system via a filtering page and data fields associated with contracts and products. The users will be able to type into the data fields and contracts and products that match the typed data will be displayed.

One of the foreseen challenges of this project could be designing a UI that will be easy for everyone to use and understand right away. We have had multiple iterations of our screen sketches because both our team and our client thought that they could be improved and that some people might get confused at times when looking at it. Creating an easy to use and understand UI can be a challenge because as the developers of the UI we will know exactly how to use it right away, and what might be obvious to us will not always be obvious to other people. This is one reason why strong communication between our team and the client will help us develop the most understandable and clear UI for Product TRACkER.

A challenge for the backend team will be polishing the filtering algorithm for products and contracts. This is an important feature for Product TRACkER and it will be one of the more challenging parts because there are many different data fields that the user can choose to filter the results by. Our goal is to also have the results display in real-time as the user types something into one of the data fields. This will add an extra layer of difficulty as opposed to having the user type something into a data field to filter by, and then clicking a button to display the new results.

Another potential challenge will be adding the functionality of adding, modifying, and searching products and contracts via uploading a csv file. This is a requirement for Product TRACkER and it is an alternate method of using the tool for just about all of the functionality cases. None of us have ever used React, JavaScript, and csv files all together at the same time to read information so this will take some time to implement as it is new to us.

4.3 Personnel Effort Requirements

For this section we went ahead and broke down each task. For each task we got the person that was assigned to the ticket to estimate with their best guest how long they felt it would take them to do. As far as who did each Task, well if you look above in our Gantt chart each task was assigned to a person, the person that was assigned to that task put their estimation down below along with a description as to why:

Task	Estimation	Details
Frontend Design	2-3 week(s)	This is where the Frontend (UI) of the project is designed. This included the creation of conceptual screen sketches and flow diagrams. This area of the project has already been completed and took around 2-3 weeks to be done correctly. This task was labeled as Medium effort for the amount of effort needed to create the diagrams, and for the need of trial and error.
Backend Design	2-3 week(s)	This is where the Frontend (UI) of the project is designed. This included the creation of backend layout, controller service layout, and database layout. This area of the project has already been completed and took around 2-3 weeks to be done correctly. This task was labeled as Medium effort for the amount of effort needed to create the project layout, spring boot project, database layout, and for the need of trial and error.
Database Setup	1 week(s)	This is where the Database was created. The design for the Database was relatively easy to create, since the data and its data types were given to us by the client. This area of the project has already been completed and took around 1 week to be done correctly. This task was labeled as Low effort because the data and data types were supplied for us.
React Project Setup	1 week(s)	This is where the React project is created and hosted on our server. The process is well documented and shouldn't take more than a week to be done correctly. This was labeled as Low effort since the procedure to create a React project is well known/documented.

Spring Boot Project Setup	1 week(s)	This is where the Spring Boot project was created and hosted on our server. This part is already complete and only took about a week. This was labeled as Low effort since this already completed in about a week.
Connect Backend	1 week(s)	This is where the Spring Boot project was connected to the database on our server. This part is already complete and took about a week. This was labeled as Low effort since it was already completed within a week.
Connect Frontend	3-5 week(s)	This is where the frontend and backend are joined. This means that the Frontend will request data from the backend and the backend will supply. From there the front end displays the received data. This was labeled as High effort due to the lack of knowledge of connecting two frameworks in such a way. Gaining the knowledge of how to accomplish this and to execute it correctly could take 3-5 weeks depending on discovered difficulty.
Form Upload Product Endpoint	1 week(s)	This is where we are allowing the user to upload product via a form. This will then get stored into the product table. This was labeled as a Low effort due to the fact that we have already completed this and it took approximately 1 week to finish.
Form Upload Contract Endpoint	1 week(s)	This is where we are allowing the user to upload contract via a form. This will then get stored into the contract table. This was labeled as a Low effort due to the fact that we have already completed this and it took approximately 1 week to finish.
Excel Upload Product Endpoint	2-3 week(s)	This is where we create an endpoint that will allow a user to upload a csv that we will then have to parse through each row to enter in the product information into the product table. This was labeled as Medium effort because we have not had to parse through a csv before therefore we will need about 2-3 weeks to learn and get this done.

Excel Upload Contract Endpoint	2-3 week(s)	This is where we create an endpoint that will allow a user to upload a csv that we will then have to parse through each row to enter in the contract information into the contract table. This was labeled as Medium effort because we have not had to parse through a csv before therefore we will need about 2-3 weeks to learn and get this done.
Contract Add Page	2-3 week(s)	This page is used to add contracts to the database. It will be displayed as a pop-up window on the user's screen and it will have data fields that the user will fill in on their own. Once the user is finished they will click a button and the contract will be added to the database. This is Medium effort because this is a large feature and it will have to connect with the backend.
Product Add Page	2-3 week(s)	This page is used to add products to the database. It will be displayed as a pop-up window on the user's screen and it will have data fields that the user will fill in on their own. Once the user is finished they will click a button and the product will be added to the database. This is Medium effort because this is a large feature and it will have to connect with the backend.
Contract Search Page	2-3 week(s)	This page will be used to find and filter through all of the contracts within the database. Users will be able to filter by one or more data fields by partial or complete matches. This is Medium effort because this page is an important page and it will have to update and refresh often for partial matches.
Product Search Page	2-3 week(s)	This page will be used to find and filter through all of the products within the database. Users will be able to filter by one or more data fields by partial or complete matches. This is Medium effort because this page is an important page and it will have to update and refresh often for partial matches.

Product Search Endpoint	2-3 weeks	This endpoint will be used to filter the entries in the product database table, and return only those data values that match the filters. I have assigned this to be a medium effort because there are a lot of components to each entry in the products table, but the built-in functionalities in Spring Boot should allow for a straightforward filtering algorithm.
Product Details Endpoint	2-3 week(s)	This endpoint will be used to pull all the product details for an endpoint that is given by a CPN. This will retrieve the information out of the product table along with the contract details to know associated contracts. This is a Medium effort. The reason I thought that this is because it won't be hard to retrieve, but in the product table it has more data so I am giving it 2-3 weeks
Contract Details Endpoint	1 week(s)	This endpoint will retrieve all the contract details including the associated products. This will pull from the contract and contract details table. The effort for this task is Low. The reason that I think this task will be low is because I will have already worked on the Product Details Endpoint and the contract has less data. So it should only take me about a week
Product Details Page	2-3 week(s)	This is the creation of the Product Details page. The Product Details page is a modal window that appears after a User clicks a link from a row of their search results. This was labeled as Medium effort because it is still an entire feature to be added, but smaller than the other pages that are required. Creating and connecting this page should take 2-3 weeks to be completed correctly.
Contract Details Page	2-3 week(s)	This is the creation of the Contract Details page. The Contract Details page is a modal window that appears after a User clicks a link from a row of their search results. This was labeled as Medium effort because it is still an entire feature to be added, but smaller than the other pages that are required. Creating and connecting this page should take 2-3 weeks to be completed correctly.

Product Edit Endpoint	2-3 week(s)	This is an endpoint that will be able to take an updated form, and then apply that updated form of product data to an existing CPN. This was labeled as Medium effort because it will need to have some filtering logic and checking, but the updating should be simple. Therefore I am giving it 2-3 weeks.
Contract Edit Endpoint	2-3 week(s)	This is an endpoint that will be able to take an updated form, and then apply that updated form of contract data to an existing CPN. This was labeled as Medium effort because it will need to have some filtering logic and checking, but the updating should be simple. Therefore I am giving it 2-3 weeks.
Product Edit Page	2-3 week(s)	This page is used to edit an existing product within the database. This is Medium effort because it should be pretty similar to the product add page but it is still going to take some time to make.
Contract Edit Page	2-3 week(s)	This page is used to edit an existing contract within the database. This is Medium effort because of the similarities with the contract add page, but will still take time.
Admin User Database Table	1-2 week(s)	This is the creation of the table in our database to store information about which users have administrative access. This is extremely straightforward to create, so I have assigned it Low effort.
Add Admin User Endpoint	1-2 week(s)	This endpoint will allow the addition of another username to the list of administrative users stored in the Admin User database table. This is a very simple endpoint and so I have assigned it to be Low effort.
Restrict Access to Admin Endpoints	2-3 week(s)	This will entail restricting access to all of the administrative endpoints to administrative users only. I have assigned this to be medium effort because it involves a number of endpoints and I believe that it can be completed in 2-3 weeks.

Admin/User Login	1-2 week(s)	This is the acquisition of the current user's username from the Collins Authentication. This is something that's process of creation is well known and relatively simple to complete. This should take 1-2 weeks to be completed correctly. Because it is a simple task it was assigned with Low effort.
Admin/User Page	2-3 week(s)	This is the creation of the pages that a User or Admin is directed to after login. These actually aren't pages on their own, but the existing search pages with additional features for admins. Since there is no real page creation but the tasks aren't simple, this was assigned with Medium effort and should be completed correctly within 2-3 weeks.
Admin Access Frontend (Testing)	2-3 week(s)	This is the testing of the admins additional features. Since this is something that will take place from the moment implementation starts to when the product is delivered. Because the creation of the admin features is medium effort and this will take place until completion of the product, then it was assigned with Medium effort.
Test: Add/Edit Frontend	2-3 week(s)	This is the time where the the add and edit functionality of the product will be tested thoroughly. This is Medium effort because we want to take our time when testing this because it is a crucial step in the development process.
Test: Search Frontend	2-3 week(s)	This task is focused on developing tests that will be used to test the front end searching functionality. This will require quite a few unit tests to ensure all facets of the search function are working correctly. This is a Medium effort task because there will be a decent amount of test cases that will need to be covered. This is also one of the vital parts of our web application, so it needs to be tested extensively.

Test: Add/Edit Backend	2-3 week(s)	This task is focused on developing tests that will be used to test our add/edit backend functionality. This will require quite a bit of unit tests, and ensure that everything is working correctly. This is a Medium effort task because there are several different things that will have to be filtered out in the add and edit pages and we want to make sure to catch them. Therefore it will take 2-3 weeks.
Test: Search Backend	2-3 week(s)	This task is focused on developing tests that will be used to test the backend of our search functionality. This will require quite a bit of unit tests to ensure that all aspects of the search are working properly. This is a Medium effort task because there are several different things that will have to be filtered in/out and we want to make sure to catch them. Therefore it will take 2-3 weeks to accomplish.
UI and Feature Enhancement/Final Turnins	3-6 week(s)	This is where additional features are created and implemented. Because these features have not been found/created as of yet, then there is no real known level of effort or time to completion. However, because of the known time constraints of the project and complexity of the project, we can assign this task with an effort level of Medium.

4.4 Other Resource Requirements

This project is purely software based so it does not have many other necessary resources like specific hardware, sensors, or circuit boards. It will only require a computer and a connection to the client's internal network.

4.5 Financial Requirements

This project can be done purely through software so there will be no need to buy any physical resources. The frameworks and software tools we are using to develop this project are all free, so there are not any financial requirements necessary to develop this project.

5. Testing and Implementation

5.1 Interface Specifications

This project is completely software driven, therefore we will have plenty of interfacing that will need testing. The main interfacing needing to be tested will be that between our backend spring boot project, and our front end react project.

The main way we will be testing our interfacing is through a couple of tools called Postman and React Dev Tools. These tools will do an excellent job in ensuring that the two components are properly integrated together, data is properly being transferred between them, and that no data is lost.

Postman is a tool that is used for performing integration testing with API to ensure that they are working properly. It does so by sending requests to the specified web server and returning the response. It allows users to set up all the headers and cookies the API expects, and checks the response.

React Dev Tools is a debugging tool that allows each edeveloper to step through the front end of the application, while constantly allowing them to check the internal state of the application to catch any inconsistencies.

Our test plan includes a combination of automated and manual tests. Testing the front-end of our web application will be done primarily through automated tests run through Selenium. Selenium is an open source tool used to automate UI tests in a browser, and it can easily be integrated with JUnit.

5.2 Hardware and software

We will be using a CI/CD pipeline run primarily by Jenkins, with Postman and Selenium run as plugins. Jenkins will be the driver of the pipeline, with Postman and Selenium test cases being run as part of the build process run by Jenkins. Every time someone makes a commit or merge to a branch, a build will begin within Jenkins, allowing errors caused by the new commit to immediately be found.

As stated in the previous section, Postman has an easy to use REST client used for testing API's. It can be used for both automated and exploratory testing, and will work great for this project because it can easily be integrated with Jenkins. Selenium is an easy to use framework used for testing web applications. It provides a domain-specific language to write tests with popular languages, and those tests can then be run against most web browsers.

5.3 Functional Testing

Unit Testing- We will create individual tests for each requirement needing to be tested. Our goal is to have 100% coverage for all requirements. JUnit tests will be written during development and will be integrated easily with Postman.

Integration Testing- After all of our individual components and functions are properly working to an acceptable standard within our unit testing, we will test larger components and how said components interact with the application as a whole. This can be done with our integration/delivery pipeline that we will build using Jenkins.

System Testing- This step of testing should be performed in an environment that closely mirrors production. For this portion of testing, we will do our best to replicate Collins intranet/production environment, but we will need them to put our application in their environment to truly know if it will work with their systems.

Acceptance Testing- After the unit, integration, and system tests are passing at an acceptable coverage percentage, we will create mock tests when significant components are completed. Sammler sub-portions of acceptance testing will be possible using JUnit, and will be considered part of unit testing since they are smaller.

5.4 Non-Functional Testing

Testing of nonfunctional requirements will end up being one of the more important aspects of our testing process. This is for the main reason that our project, once completed functionally, will be deployed inside of Collins' intranet. We will need to test the portability of our project as we move along. The first priority is getting it to work, but the immediate second priority will be making sure that it works not just in our vacuum of our local database, but that it works inside of Collins' workspace.

5.5 Process

As shown in the diagram, we will start by adding a test that should reflect if a certain portion of code is doing what it is intended to do. Once a commit is made, the tests will begin to run. If the test passes, then we know that the code is functioning properly, and we can continue to develop more test cases and test more features within our software. If the test fails, we will continue to make changes to the test until it passes. Once it passes again, we can go back to writing more test cases.

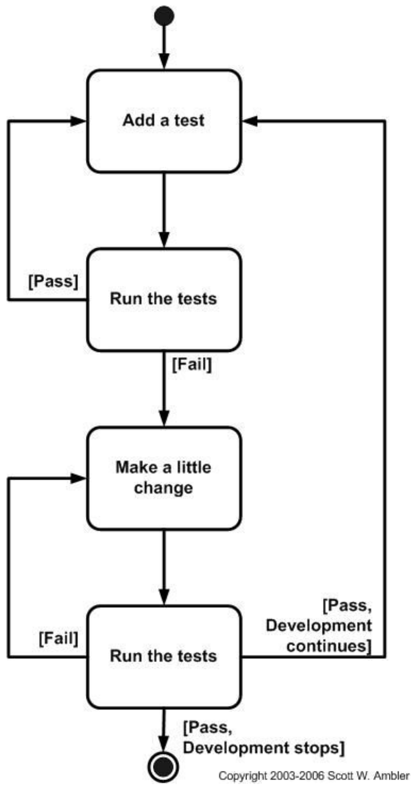

Add a test
Run the tests
[Pass]
[Fail]
Make a little change
Run the tests
[Fail]
[Pass, Development continues]
[Pass, Development stops]
Copyright 2003-2006 Scott W. Ambler

Figure 20: Testing Workflow

Figure 20 shows the workflow for how we will develop tests as it relates to our project. The figure reflects a standard test driven development process, and this is what we will achieve when writing our software.

5.6 Results

Our final test suite is a compilation of jUnit tests that cover each and every one of our back end REST API endpoints. Each endpoint has at least two test cases- one that is verifying the happy path works as intended, and one verifying that the unhappy path will throw an exception. Another tool commonly used for backend testing was Postman. Postman was used throughout the development of new endpoints to determine the endpoint's functionality and validity. In terms of the frontend, all modules have been thoroughly manually tested by our frontend team within our sites application. We did not configure a build pipeline to automatically run our test suite, as we felt this would not directly benefit the client, rather we spent the additional time adding features to the product itself. Our jUnit tests will easily be able to be added to Collin's own test suite with the future ability to further contribute to the tests.

6. Closing Material

6.1 Conclusion

We have been communicating with the client often in order to get the finalized version of the requirements and deciding on a good user interface. The team has been working on this design document this semester and also making system design figures. The team has also worked on coming up with a good user interface design and screen sketches have been made to reflect what we expect the final version of the project will look like. The back-end team has set up some of the necessary endpoints to communicate with the database and by the end of this semester one of the goals for the team is to have a small working prototype that can do a couple of the requirements. Our goals for next semester is to have the final version completed by April 2020 or early May 2020. The way we will do this is by having a meeting together as a team to keep on track, we will have a schedule made so that we can follow along with that week to week, and we will divide the work evenly among everyone on the team. This plan of action will work because it will keep the team organized and this way everyone will know what they have to do for the team, and when they need to have it completed by.

6.2 References

S. W. Ambler, Agile Testing and Quality Strategies: Discipline Over Rhetoric. [Online]. Available: http://www.ambysoft.com/essays/agileTesting.html#TestDrivenDevelopment. [Accessed: 06-Dec-2019].

https://images.app.goo.gl/UkrFhbkfcen7fENz8. (2019). [image].

E. Alberto, “Part 3: Spring Boot Components: Controller, Service, Persistence/DAO,” Medium, 22-Apr-2018. [Online]. Available: https://medium.com/@erwinalberto/part-3-spring-boot-components-controller-service-persistence-dao-7e8c8f1c844b. [Accessed: 08-Dec-2019].

“What is Agile Software Development?,” Agile Alliance, 12-Oct-2019. [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed: 09-Dec-2019].

“Scrum,” Agile Alliance, 26-Sep-2019. [Online]. Available: https://www.agilealliance.org/glossary/scrum/. [Accessed: 09-Dec-2019].

“The Ultimate Guide to Gantt Charts,” ProjectManager.com. [Online]. Available: https://www.projectmanager.com/gantt-chart. [Accessed: 09-Dec-2019].

“SQL Join Types,” SQL Joins Explained. [Online]. Available: http://www.sql-join.com/sql-join-types. [Accessed: 09-Dec-2019].

“React – A JavaScript library for building user interfaces,” – A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/. [Accessed: 09-Dec-2019].

“Spring Projects,” Spring. [Online]. Available: https://spring.io/projects/spring-boot. [Accessed: 09-Dec-2019].

“Standards,” Standards. [Online]. Available: https://standards.ieee.org/standard/index.html. [Accessed: 09-Dec-2019].

6.3 Appendices

CPN - Collins Part Number

EQPT - Equipment type

POC - Point of Contact

EOP - End of Production

EOS - End of Service

CM - Contract Manager

UI - User Interface

Appendix I - Operation Manual

Quick overview

You will need some way to display a web application. We used an apache web server.

You will need to have the front end application build placed in the var/www/html/ directory.

You will need to have the back end application placed on the server and running as a background process.

Front End

Front end is made through the use of React.js in combination with jQuery and accompioning plugins. This means you will need Node.js and NPM(node package manager) installed. *Note: you may want to do the NPM install inside the directory containing the frontend code. If you would like to put any new changes you have made onto the web application you need to do the following:

1. Remove anything in the var/www/html/ directory (if using a linux server: use “sudo rm -rf /var/www/html/*”).
2. Navigate to the directory holding the front end code.
3. To be safe, remove the current Build directory (if using a linux server: use “sudo rm -rf build/”).
4. Run the build command (if using a linux server: use “sudo npm run-script build”). This will create a new Build directory.
5. Move the contents of this build directory into our var/www/html/ directory. (if using a linux server: use “sudo cp -R build/* /var/www/html/”).
6. You should see all your changes!

7. If you would like to create a live-updating test environment, go to the directory containing the front end code and use command “sudo npm start” then go to the url “localhost:3000”

Back End

For the back end it is made using maven and spring boot for compilation. This means you will need to install maven and spring boot. For spring boot I would recommend working an IDE that already has these features built into it, such as IntelliJ or Spring Boot Suite. Now to deploy the application onto the server you will have to follow the following steps:

1. You will have to go to project’s root directory
2. Run mvn clean this will clean out all old builds from the target folder
3. After this has been run, you can run mvn install. This will run all junit tests and compile your build into a single snapshot jar
4. Now that you created the snapshot in the target folder, change into the target directory
5. Once in the target directory run java - jar {snapshot-name}.jar
6. If you would like to have the terminal back in your control run step 5 with an & at the end
7. If you would like the logs to be stored in a file run commands 5 and 6 with nohup at the front this will store the logs in a file, and not fill up your terminal output.
8. Your code is now live on the hostname:8080
9. When you run this code locally, do the same process as above. However when you do the above after completion the code will be live on localhost:8080
10. If the above steps are confusing I would recommend to use the IntelliJ or Spring Boot Suite GUI. This is relatively easy to navigate and for the most part will be able to deploy your application locally with a single push of a button.

Testing

Back End

For the back end we did implement several junit tests in the test directory of the project. The junits are intended to test every endpoint with both a happy and bad path that way we can try and achieve full code coverage without affecting our database. To run the tests please do the following:

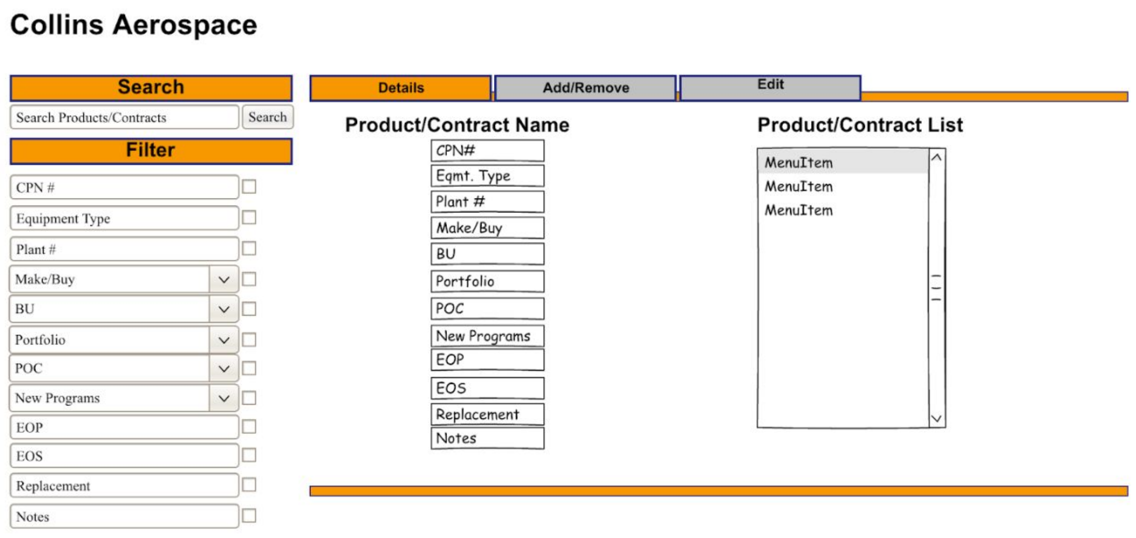
1. To run the junit tests on the backend you can do it three different ways.
2. The first way would be to run the tests testing package in one of the spring boot IDEs such as IntelliJ or Spring Boot Suite by right clicking the testing package and clicking run all tests
3. The next way you can do it is by using maven. You can run mvn test this will run all the tests in the testing package, and will give a little integration testing by seeing maven work with spring boot.
4. The last way you can test the back end code is by following the above build steps. When you run the mvn install this will actually run the junit tests before it will create the snapshot. If the tests fail the snapshot will not build.

Another aspect of testing is to be able to see the whole system work together after creating a new endpoint or such. For doing this type of testing I recommend to use Postman. This has a friendly user interface. All you have to do is put in the endpoint url, and add in the needed parameters. This will then hit the endpoint you need, returning the response code and response in the form of JSON if there is any. Allowing you to validate that the endpoint is taking in the correct data, and is sending back the data you are expecting.

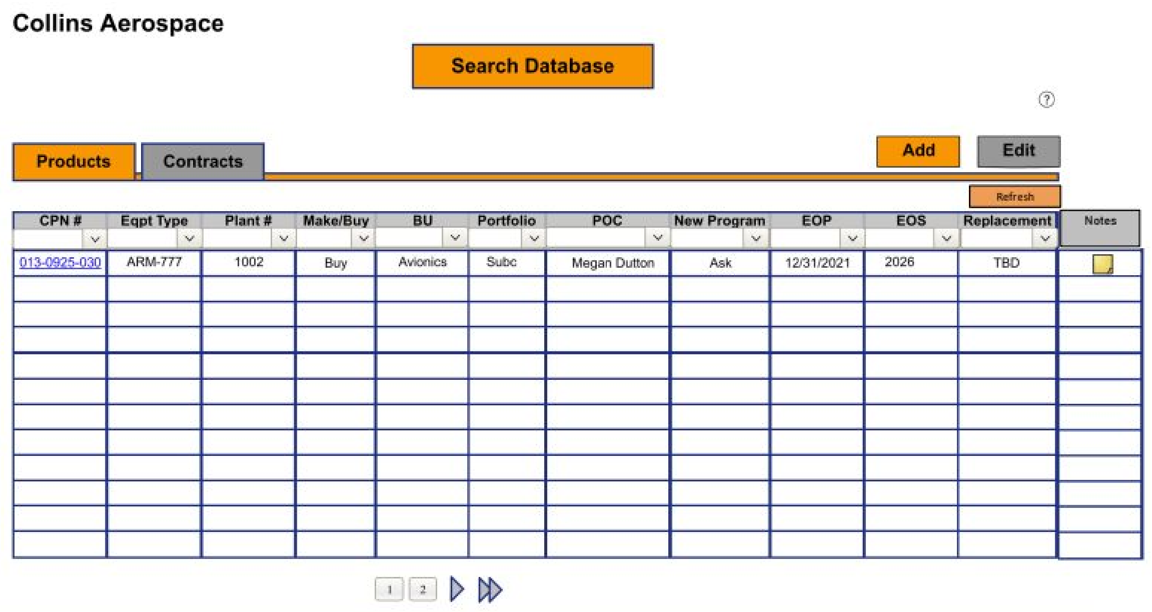
Appendix II - Alternative or Previous Versions

When we were initially introduced to the project and some of the elements that were going to be involved in the project, we all knew we would have to create a web application with a database to hold the large sets of data. One of the initial issues that we ran into was deciding a language that we wanted to use for the front end of the web application. Because the front end team didn't have tremendous experience creating the front end portions of web applications, we did some research as well as consulted with the back end team for their suggestions. We initially decided to go with AngularJS as our framework, as we thought that it would provide the best solutions to our project, with how encapsulating and self-sufficient the framework is. One thing that we didn't anticipate was the learning curve involved with this language though, and we quickly realized we should take the front end in a different direction. Upon a little more research and testing, we decided to move our front end framework towards using ReactJS. The most obvious reason we switched was the simplicity of the language for our web application. We didn't need anything super robust or time consuming to create, we wanted a simple framework that would allow us to develop the UI how we wanted, and that is what ReactJS provided us. This was a great switch and we had relatively few issues the remainder of the semester in regards to the front end framework. We were able to create an easily accessible, traversable, and seamless UI for our client.

Initial Sketch



Updated Sketch



CPN #	Eqpt Type	Plant #	Make/Buy	BU	Portfolio	POC	New Program	EOP	EOS	Replacement	Notes
013-0925-030	ARM-777	1002	Buy	Avionics	Subc	Megan Dutton	Ask	12/31/2021	2026	TBD	

The second issue that we had that we eventually switched from stems from some of the issues we had changing the framework. Later on when we were given some of the MIRs from our client, we started to brainstorm how we wanted the web application to look with the creation of some screen sketches. We had a few ideas on the direction we wanted to go based on the MIRs that we were given and an example of a web application that was already deployed within their intranet. We based our initial screen sketches off of the application they showed us, and then we displayed them to our client in our meeting later that week. This was met with some criticisms and critiques, and we were put back to stage one. After getting an idea from our client, we were able to create a screen sketch and guideline that they were later super happy with and we were able to create into a working version.

Aside from those two small version changes, we really stuck with our initial plan from the start of the year. We planned and scheduled very well to ensure we had covered almost all of our bases from the beginning, and were able to progress smoothly throughout the year.

Appendix III - Other Considerations

It has been a long and busy year for us in group two with Product TRACkER. To say we learned a lot throughout our time together working on this project would be an understatement. There are lots of things that we did correctly to begin with, and even more that we learned to value more along the way. One thing that provided us with a lot of ease of mind later in the year was the amount of scheduling we had done to begin with. Because we were very detailed and in depth with our scheduling, task breakdown, and task delegation we were able to complete things with plenty of time for testing at the end. It also allowed us to know where we were at all times throughout the year, allowing us to stay ahead of deadlines, demos, and meetings.

Stemming from that scheduling, leads to another thing that we are all very happy that we did. Planning out plenty of time, over a month in total for testing at the end of the year. This proved to be very valuable for us, not just to allow us to test our product, but to allow us to get more feedback from our client and possibly work on small features that could enhance the final project. Because we had extra time at the end of the semester, we were able to provide our client with something that more closely resembled the product they were hoping for from us.

Things were not always perfect for us, and we did learn lots of things along the way like I mentioned. Because none of us had tremendous experience working on the front end side of a web application, our front end team learned a lot about the whole approach and execution of creating a workable and functioning UI. We also learned a lot about doing better research to begin with. If we took a little bit more time researching frameworks for the front end, we could've started off on the right foot with the right framework for us. We started a little too quickly, and because of this we had to switch frameworks after a week or two of trying to learn and use it.

We also learned about the tremendous importance of having constant communication and meetings between group members. Because we lacked the proper communication early on in the semester, it led to lots of unnecessary and unwanted issues. We weren't on the same page all the time and we lacked a proper vision to our project for the first few weeks. It also led to issues with some group members feeling left out and not knowing the direction the project was heading. Obviously this was a serious issue, and these political issues can be just as disruptive as the technological issues. Luckily, we were able to correct most of the issues in a timely manner before they became too detrimental and distracting to our group.