Production Run Reports

Design Document

sdmay23-06

JEDA Polymers

Dr. Stoytchev

Colton Carlson - Scrum Master / PDF Generation / Email Service

Hayden Havelka - Backend

Jay Arnold - Backend

Connor Linn - Frontend

Noah Meyer - Frontend

sdmay23-06@iastate.edu

https://sdmay23-06.sd.ece.iastate.edu/

Table of Contents

Table of Contents 2

Executive Summary 7

Development Standards & Practices Used 7

IEEE Standards 7

Summary of Requirements 7

Functional Reqs 7

Non Functional Reqs 7

Applicable Courses from Iowa State University Curriculum 7

New Skills/Knowledge acquired that was not taught in courses 8

1 Team 9

1.1 Team Members 9

1.2 Required Skill Sets for Your Project 9

1.3 Skill Sets covered by the Team 9

1.4 Project Management Style Adopted by the team 10

1.5 Initial Project Management Roles 10

2 Introduction 11

2.1 Problem Statement 11

2.2 Intended Users and Uses 11

Persona 1: John - Product Manager 11

Persona 2: Jeff - Owner of JEDA Polymers 11

Persona 3: JEDA Polymer Customers 12

2.3 Requirements & Constraints 12

Types of Requirements 12

Functional Reqs 13

Non Functional Reqs 14

2.4 Engineering Standards 14

3 Project Plan 15

3.1 Project Management/Tracking Procedures 15

3.2 Task Decomposition 15

Front End 15

Backend 15

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 16

Milestones and Metrics 16

3.4 Project Timeline/Schedule 16

Project Timeline 16

Backend 16

Frontend And Integration 17

Testing Beta and Updates 17

3.5 Risks And Risk Management/Mitigation 18

Risks 18

Risk Management 18

3.6 Personnel Effort Requirements 20

Personal Effort 20

4 Design 22

4.1 Design Context 22

4.1.1 Broader Context 22

Public health, safety, and welfare: 22

Global, cultural, and social: 22

Environmental: 22

Economic: 22
4.1.2 Prior Work/Solutions 22
Colton 22
Previous work experience 23
Experience related to project 23
Jay 23
Previous work experience 23
Experience related to project 23
Connor 23
Previous work experience 24
Experience related to project 24
Noah 24
Previous work experience 24
Experience related to project 24
Hayden 24
Previous work experience 24
Experience related to project 24
4.1.3 Technical Complexity 25
4.2 Design Exploration 25
4.2.1 Design Decisions 25
4.2.2 Ideation 25
4.2.3 Decision-Making and Trade-Off 25
4.3 Proposed Design 27
4.3.1 Overview 27
4.3.2 Detailed Design and Visual(s) 28
Frontend 28

Backend 28

PDF 28

4.3.3 Functionality 31

4.3.4 Areas of Concern and Development 32

4.4 Technology Considerations 32

4.5 Design Analysis 33

5 Testing 34

Mindset 34

5.1 Unit Testing 34

Front-End 34

Back-End 35

5.2 Interface Testing 35

5.3 Integration Testing 35

5.4 System Testing 36

5.5 Regression Testing 36

5.6 Acceptance Testing 36

5.7 Security Testing (if applicable) 37

5.8 Results 37

6 Implementation 39

7 Professional Responsibility 40

7.1 Areas of Responsibility 40

7.2 Project Specific Professional Responsibility Areas 43

7.3 Most Applicable Professional Responsibility Area 43

8 Closing Material 44

8.1 Discussion 44

8.2 Conclusion 44

8.3 References 45

8.4 Appendices 45

8.4.1 Team Contract 45

Executive Summary

Development Standards & Practices Used

IEEE Standards

- IEEE 26515-2018 - Agile Development Cycle
    - Developmental cycle we will use. Allows us to be flexible with the customer and have clear constant deliverables.
- IEEE 829 - Software Test Documentation
    - Allows us to easily document our tests, easily shows what test does what and explains expected results.
- IEEE P1363 - Public Key Cryptography
    - Allows us to ensure that only authenticated users will request the generated reports.

Summary of Requirements

Functional Reqs

- Create a report when requested
- Email Reports at end of night
- Log Errors
- Display repeated error messages on on front end

Non Functional Reqs

- Generate reports in 15 seconds.
- Follow approved pdf template.
- Be accessible by Chrome, Safari, Firefox, Opera, and Edge.
- Use JEDA's internal mail server.
- Server runs on Windows 2019 Server.
- Use Canary Database

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- ComS 309
- SE 329
- ComS 311
- ComS 409
- SE 339

- ComS 327
- ComS 319

New Skills/Knowledge acquired that was not taught in courses

- REST API’s
- PDF Generation
- Reading existing documentation
- Standalone Server Development
- C#

1 Team

1.1 Team Members

- Colton Carlson
- Jay Arnold
- Connor Linn
- Hayden Havelka
- Noah Meyer

1.2 Required Skill Sets for Your Project

- Knowledge on how to use GIT
- ASP.Net Development
- React Development
- Email API integration
- Server Side Deployment
- React Framework
- Robot Framework
- HTTP familiarity

1.3 Skill Sets covered by the Team

- Knowledge on how to use GIT
  - Connor Linn
  - Hayden Havelka
  - Colton Carlson
  - Noah Meyer
  - Jay Arnold
- ASP.Net Development
  - Colton Carlson
  - Hayden Havelka
  - Jay Arnold
- React Development
  - Connor Linn
  - Noah Meyer
- Email API integration
  - Colton Carlson
- Server Side Deployment
  - Colton Carlson
  - Jay Arnold
- Robot Framework

    - Colton Carlson
- HTTP requests familiarity
    - Connor Linn
    - Hayden Havelka
    - Colton Carlson
    - Noah Meyer
    - Jay Arnold

1.4 Project Management Style Adopted by the team

Agile - 2 Week Sprints

1.5 Initial Project Management Roles

- Scrum Master: Colton Carlson
- Product Owner John Deeken
- Developers: Colton Carlson, Hayden Havelka, Jay Arnold, Noah Meyer, Connor Linn

2 Introduction

2.1 Problem Statement

Our project is to allow employees at JEDA Polymers to look up work orders, create a production report for the work order, and then view or download in PDF form. An employee will also be able to send the reports through email using an automated email system. They currently look up the data by hand anytime they want specific data. This is wasting a lot of time, over 30 minutes for a subpar potentially inaccurate report, and is tedious. The problem is occurring at the work office where time is precious. It is important to make it easier to get automated production reports because it will save time and less errors in getting specific data. We will solve this problem by having a frontend, backend, and email system.

2.2 Intended Users and Uses

Persona 1: John - Product Manager

Characteristics:

- Likes consistency and precision in a process above other things
- To the point and does not like to waste time on menial tasks
  - Would like to have processes optimized for a speedier resolution

Needs:

- John needs a way to retrieve an accurate report of their production runs in a timely manner because there is currently no way to confirm production output in a timely manner.

Benefits:

- John can use these reports as evidence when customers claim they received bad material.
- John can also use these reports to help give potential customers accurate data to form proper expectations.
  - Show better product quality through transparency of the product making process.

Persona 2: Jeff - Owner of JEDA Polymers

Characteristics:

- Customer satisfaction is crucial for business
- All customer facing aspects should be well designed and look “professional”

Needs:

- Jeff needs to be able to check the runtime of machines by making a report and then checking the runtime of machines.
- Jeff needs to be able to review a report because the current way is looking up data on his own.
- Jeff needs to be able to send a report through email because his quality engineers do not know how to consistently find accurate data.

Benefits:

- Jeff can use these reports to see if a product takes longer than expected.
- Jeff can benefit from these reports by looking at power usage to know accurately how much each run costs the company in electricity.
- Jeff can know if a batch of plastics did not meet production requirements and not send out bad products.

Persona 3: JEDA Polymer Customers

Characteristics:

- Compare plastic manufacturing companies
    - Can use production reports to compare to other companies
- Time oriented
    - The customer doesn't like wasting time.

Needs:

- The customer will need the production reports to have an effective analysis of the data.
- The customer will need verification that issues are not with plastics provided by JEDA Polymer.

Benefits:

- The customer won't have to wait a long amount of time for a report.
- The customer can easily share reports internally.
- The customer can use a report to see that the batch of plastics is up to standard.

2.3 Requirements & Constraints

Types of Requirements

We put all of our requirements into one of two categories: Functional and Non-Functional, which is taught in COM S 409/509 by Dr. Robyn Lutz.

Our functional requirements can simplified down to one of 5 types:

- Ubiquitous
    - The <system> shall <response>

- Event-Driven
    - When <trigger> the <system name> shall <response>
- State-Driven
    - While <in specific state>, the <system name> shall <response>
- Option
    - Where <feature is included>, the <system name> shall <response>
- Unwanted Behavior
    - If <unwanted trigger>, then the <system name> shall <response>

Our Non-Functional Requirements can be watered down to one of, or a combination of, the following 8 types.:

- Look and Feel
    - Appearance
- Usability
    - Ease of use/understanding
- Performance
    - Timing/Reliability/Accuracy
- Operational
    - Context/Environment
- Maintainability
    - Anticipate changes/updates
- Security
    - Confidentiality
    - Privacy
    - Integrity
    - Auditability
    - Availability
- Cultural and Political
- Legal

Functional Reqs

- When a request is made the system shall produce a new report
- When an email request is made, the system shall email the most recently made report to the specified email address
- Where the line number is included, the system shall only look in the line specified for the work order.
- When the time turns 11:59 PM CST the system shall email a report for every work order that finished during the previous 24 hours.
- If a work order does not exist, then the system shall notify the user.
- When a report is produced, then any locally obtained data shall be deleted.

- When a report is requested, add date, time, and work order number to the log file
- When a report is finalized, add date, time, and work order number to the log file
- When a report is emailed, add date, time, and work order number to the log file
- If a report fails to be generated, then add date, time, work order number, and error to the log file, then retry.
- If a report fails to be generated 5 times in a row, display the error message on the front end for the user to see.

Non Functional Reqs

- The system shall produce a report in 15 seconds. ***Constraint***
- The system shall produce a PDF report based on the template decided on. ***Constraint***
- The system shall produce a PDF that contains the machine data from the requested work order.
- The system shall use company colors. ***Constraint***
- The system shall be able to support new lines/components.
- The systems frontend shall use React/Node.js libraries ***Constraint***
- The systems frontend shall be accessible from Mac and Windows. ***Constraint***
- The system shall run on Safari, Chrome, Firefox, Opera, Edge. ***Constraint***
- The system’s backend shall run on a Windows 2019 Server. ***Constraint***
- The system shall not include null/zero values in the report.
- The system shall use FTP to send the report to the front end. ***Constraint***
- The system shall use the internal email system to send emails.
- The system shall use IMAP as the protocol for sending emails. ***Constraint***
- The system shall use OPC UA as the protocol for communicating with Canary ***Constraint***
    - This allows canary and backend communication to be OS independent.
- The system shall have access to the work machine’s data via the Canary Database.

2.4 Engineering Standards

IEEE Standards

- IEEE 26515-2018 - Agile Development Cycle
    - Developmental cycle we will use. Allows us to be flexible with the customer and have clear constant deliverables.
- IEEE 829 - Software Test Documentation
    - Allows us to easily document our tests, easily shows what test does what and explains expected results.
- IEEE P1363 - Public Key Cryptography
    - Allows us to ensure that only authenticated users will request the generated reports.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our group decided to go with the Agile development methodology. One of the main reasons we picked this is because we have a client with needs that could change. With Agile, we are able to adapt to changing needs and implement/change the code. Another reason is that agile is used by most software/tech companies. Using agile for this project will be beneficial because we are learning popular methods that are applicable to the real world. We are using GitLab for the repo and to track progress.

As a team we also are using GitLab to help with version control and code integration. Git is pretty standard across most coding projects. Since most of the group has already used this before, Git was a no brainer for our version control in this project. We added security protocols such as not being able to push directly to main, and requiring merges to have at least one approval.

3.2 Task Decomposition

Front End

- Create 5 pages that the user will interact with
  - (Log in, Request Data, Results, Make New Account, Reset Password)
  - Make all pages follow a similar format for ease of use and a uniform look
- Integrate REST API to make HTTP calls to backend
- Testing

Backend

- Report generation
  - Creates report from template with data from database
- Database
  - Use Canary API to get data from database
- Email API
  - Use Email API to send report PDFs through email
- Log
  - Configure start/end times for report generation
- Config File
- Windows Server 2019
  - Configure the application to be hosted on Windows Server 2019

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones and Metrics

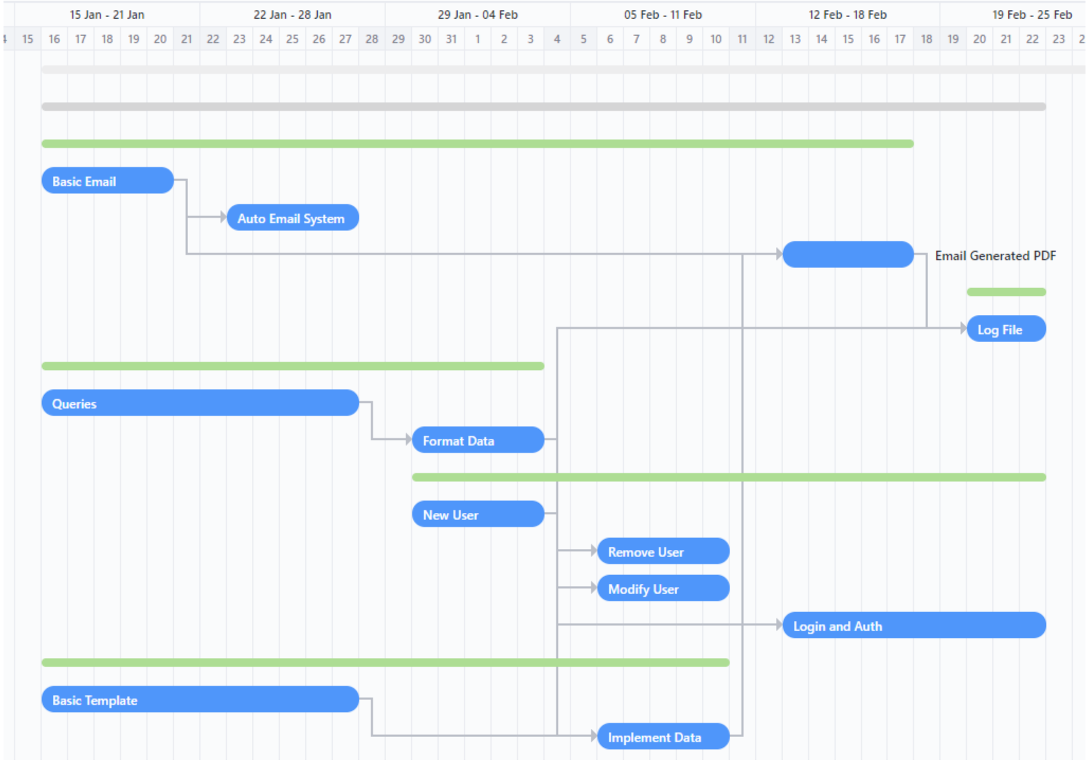
Go over milestones for each task, testable and easy to say yes or no it works.

1. An accurate report is generated with 90% accuracy.
2. A report is generated within 15 seconds.
3. A report is emailed to specific emails using the frontend.
4. The log file is correctly displaying start/end times for report generation.
5. The frontend correctly displays a report 90% of the time.
6. The frontend allows new accounts with usernames and passwords 90% of the time.
7. The reset/forgot password page works 90% of the time.
8. The application works on 3 out of the 5 total browsers.

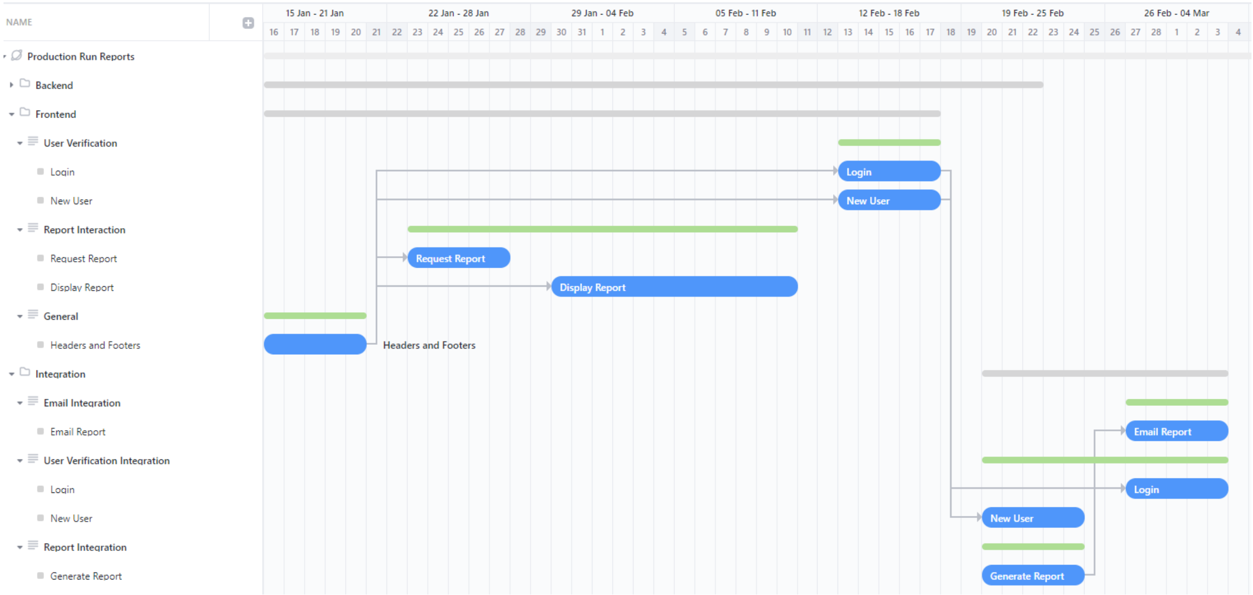
3.4 Project Timeline/Schedule

Project Timeline

Backend



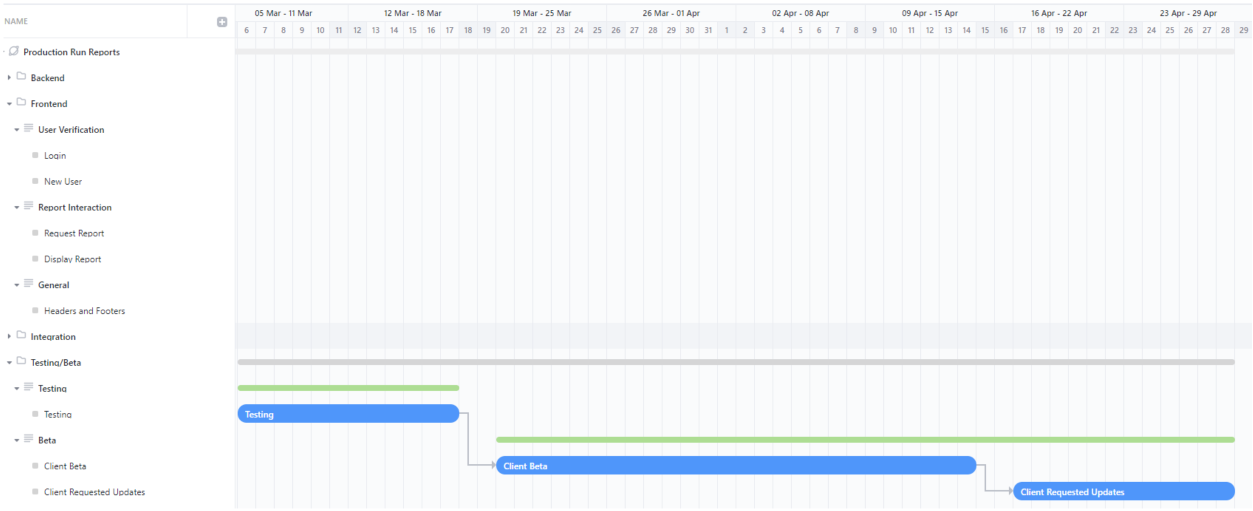
Frontend And Integration



NAME

- Production Run Reports
  - Backend
  - Frontend
    - User Verification
      - Login
      - New User
    - Report Interaction
      - Request Report
      - Display Report
    - General
      - Headers and Footers
  - Integration
    - Email Integration
      - Email Report
    - User Verification Integration
      - Login
      - New User
    - Report Integration
      - Generate Report

Testing Beta and Updates



NAME

- Production Run Reports
  - Backend
  - Frontend
    - User Verification
      - Login
      - New User
    - Report Interaction
      - Request Report
      - Display Report
    - General
      - Headers and Footers
  - Integration
  - Testing/Beta
    - Testing
      - Testing
    - Beta
      - Client Beta
      - Client Requested Updates

3.5 Risks And Risk Management/Mitigation

Agile project can associate risks and risk mitigation with each sprint.

Risks

			Impact		
Probability	Very Low	Low	Medium	High	Very High
Highly Probable	5 - Moderate	10 - Major	15 - Major	20 - Severe	25 - Severe
Probable	4 - Moderate	8 - Moderate	12 - Major	16 - Major	20 - Severe
Possible	3 - Minor	6 - Moderate	9 - Moderate	12 - Major	15 - Major
Unlikely	2 - Minor	3 - Moderate	6 - Moderate	8 - Moderate	10 - Major
Rare	1 - Minor	2 - Minor	3 - Minor	4 - Moderate	5 - Moderate

** Chart values are referenced in the Risk Mitigation section when discussing risks and their solutions.

Risk Management

Risk	Solution
Client backing out - 10	As a team we are developing an important tool for their business. Additionally we are keeping in close contact with them so that they have an

	understanding of the development process.
Losing access to the database that holds the machine data - 9	Ensuring that we constantly have the most up to date credentials to the machine's database.
Bad data injection from malicious users - 10	Authenticating users to ensure only users with the correct credentials are accessing the system.
The web application does not load properly on different browsers - 3	Check that the frontend is able to run and display correctly on multiple browsers.
Credentials server fails to authenticate users	Have thorough testing on the login system, as well as a flushed-out system to reset a users page.
PDF generation fails - 9	Check connection to database, check if the queries are correct.
Queries return wrong results or NULL even when the correct data is there - 6	Check the connection to the database, and make sure the queries are correct.
Report generates with incorrect data - 12	Test to make sure that we are getting correct data from the database. Once the queries are working correctly, we can then inject data into the report.
Report is not emailed to the correct recipients - 6	Test to make sure that emails are sending reports to the correct recipients.
Log file does not show correct PDF generation data - 6	Check to make sure reports are being generated correctly by testing.

New user’s account is not registered into the system - 12	Check to make sure the APIs are being used correctly by testing.

3.6 Personnel Effort Requirements

Personal Effort

The following chart is a basis for the expected number of hours per task that is expected to be spent. These numbers were chosen based on the factors of: number of individuals working on a task, time to learn necessary skills per task, average time spent on similar tasks.

Task	**Hours**
Report Generation	10
Query Database	16
Log File	6
Formatting Data	40
Email API	20
Login Page	10
Request Data Page (WO# and L#)	16
Results Page	25

Make New Account Page	10
Reset/Forgot Password Page	8
Integration	20
Testing	45
Optimization and Bug Fixing	45

4 Design

4.1 Design Context

4.1.1 Broader Context

Public health, safety, and welfare:

The general well being of various stakeholder groups will be affected by the project. A reason for the direct users is because it will most likely decrease stress on the job. The purpose of the project is to automate the process of making a report. The client is having to manually go in and get data. This is taking time and can be a nuisance. The indirect users, clients of JEDA Polymers, will be benefited because the reports will be professional and effective. The project will not harm anyone or anything.

Global, cultural, and social:

The project only aims at one cultural group. The cultural group is JEDA Polymers workplace. The operation of this project will not violate JEDA Polymers code of ethics. It won't make an undesired impact in the workplace practices. The project will only help workplace practices by increasing productivity.

Environmental:

The project will have a couple environmental impacts. First, the project will require electricity to produce and run the software application. There are environmental impacts based on how the electricity is produced, but that is out of our control. The second impact is that the project will be incorporated into a company that manufactures plastic. The project itself is not manufacturing plastics, but the project can be associated with plastics.

Economic:

This project will have some economic impact. The application will not be very expensive to make and run. It is a feasible cost for the client. Also, the users benefiting from this project will spend less time making production reports. Furthermore, they will have more time to work on other work.

4.1.2 Prior Work/Solutions

Colton

Previous work experience

2 years at Collins Aerospace, 2 Summers and currently employed at Citrix Systems (Cloud Software Group).

Experience related to project

At Collins I worked with C# and .NET programs that had to run on windows natively. This relates to the project as our backend needs to run on a Windows 2019 Server and having it run natively reduces any friction if wanting to put the program on a fresh install. I also worked on an automated report to be emailed, while not writing the actual email calls it did get me exposure to the thought process, which I will use when writing the API calls to send the reports.

At Citrix I worked on remote servers sending and receiving data and sending operations to be completed from client to host with results being retrieved. This relates to the project as having a smooth interaction between web clients and the backend host is crucial, and reducing lag and or bloated run times. Finally at Citrix I am working on pioneering the introduction of a new frameworks that our team wants to introduce, and as there is no one else on the team that knows how to use it I am reading a lot of documentation, which will be a useful skill when reading documentation on PDF generation / .NET Rest API's.

Jay

Previous work experience

Two internships with OnPoint Solutions as a DevOps engineer.

Experience related to project

I have learned about software development during my internships. I worked on some CDK stacks that were in Python. I also worked on another project which was written in TypeScript. Other than that I didn't get any experience with frameworks or databases in my internships.

I took a project class, ComS 309, that gave me some experience related to this project. During the class, I used Springboot and Java to develop the backend for an Android application. I also used SQL to query a database. This is related to the project because it will require backend knowledge with a database.

Connor

Previous work experience

An internship for Empirical Foods as a software engineer.

Experience related to project

At this internship I got both experience with .NET and frontend development using React, as well as ISS and accessing servers.

Noah

Previous work experience

3 summer internships at John Deere.

Experience related to project

At John Deere I worked in embedded systems software and developed code in C, C++, Python, and Java. I did some front end development when I programmed a virtual simulator that administered automated software tests.

Hayden

Previous work experience

Entrepreneurial endeavors and class experience: 309, 319

Experience related to project

I started a business that required a website. Because of this I was able to learn more about the web development process, as well as, web design. This was in addition to the web development related classes that I have taken during my time at Iowa State such as 309 and 319. For 309 I learned about HTML, Javascript and React. For 319 I learned about backend server development. All of this knowledge will help me to understand the development process for the web based services we are developing.

4.1.3 Technical Complexity

This project is giving us numerous challenges. For the frontend portion of the project, none of us have had to build a fully flushed out website using React before. Let alone making a site that can function on any given browser with full responsiveness. We will also have to explore the complexities of making the site work on both Mac and Windows.

As we move to the backend portion of the project we are faced with setting up a server that will be running 24/7. Not only have we never set up a server before but making one that can run seamlessly for 24 hours a day will be a massive challenge. This will also have to connect to the Canary database which is a foreign architecture to everyone in the group. And on top of all this we need to cover our bases with security protocols and we do not have a cyber security specialist in our group.

Then we have to make sure that our system works within the company's given suite, run quickly enough that the client can access the data within a reasonable amount of time, and have an interface that is user-friendly enough for the client to continue using it after we leave. Previously all of our classes projects have been built by engineers, for engineers. Making a product that a general user can run efficiently is imperative for this project's success.

4.2 Design Exploration

4.2.1 Design Decisions

1. React/Node JS was chosen to develop for the front end.
2. Canary Labs was chosen to query machine data.
3. The .Net REST API was chosen for the email service.

4.2.2 Ideation

In order to come up with the best development language for the front end we choose from five different front end development languages as listed below:

- PHP - Scripting language used in web development.
- Javascript - programming/scripting language that expands the functionality of HTML/PHP
- React - Library/framework that expands the functionality of Javascript
- Flask - Flask is a frontend framework built on Python for web development
- Angular - Library/framework that expands the functionality of Javascript

4.2.3 Decision-Making and Trade-Off

After starting our comparison between the different languages React started with a significant leg up on the rest of the options. To start with, several of our members have experience using React already. Additionally, REST Api's make the connection between our front and back end seamless. Moving forward it was more of comparing each of our other options to how they stood up against React rather than against each other. However, this was still an easy choice because the comfortability of our developers working in an environment they are already familiar with seemed to outweigh any other major issues. This does not mean we chose an easy language because of familiarity though. React still stood higher on our scale than the other languages in terms of speed and usability. The fact that we were already comfortable in the language was simply a fortunate consequence.

Criteria	**PHP**	**React**	**JavaScript**	**Flask**	**Angular**
Functionality	3	5	4	1	3
Browser Compatibility	5	5	3	2	1
Code Simplicity	2	4	2	5	2
Speed	3	5	5	2	3
Learnability	1	3	2	5	1
Total (unweighted):	14	22	16	15	10

Higher is better

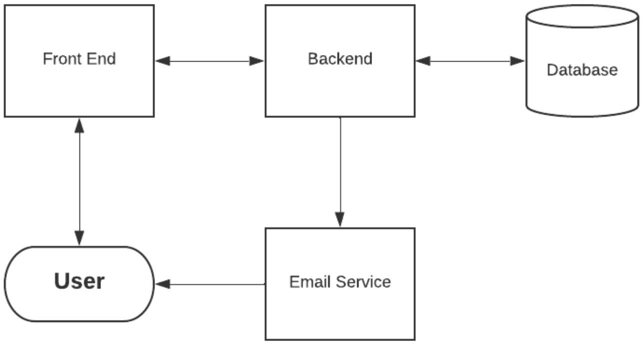
Criteria	**Functionality**	**Browser Compatibility**	**Code Simplicity**	**Speed**	**Learnability**
Weight	1.0	1.5	1.0	1.25	.75

	PHP	**React**	**JavaScript**	**Flask**	**Angular**
Total (Weighted)	17	25	18.25	15.25	11

After looking through all of our options, we chose React/Node JS because of its compatibility with our backend services as well as the variety of design options that it offers.

4.3 Proposed Design

4.3.1 Overview



User *= John/Any other user who wants a work order generated.*

Email Service *= REST API calls to send emails of generated PDFs to the user.*

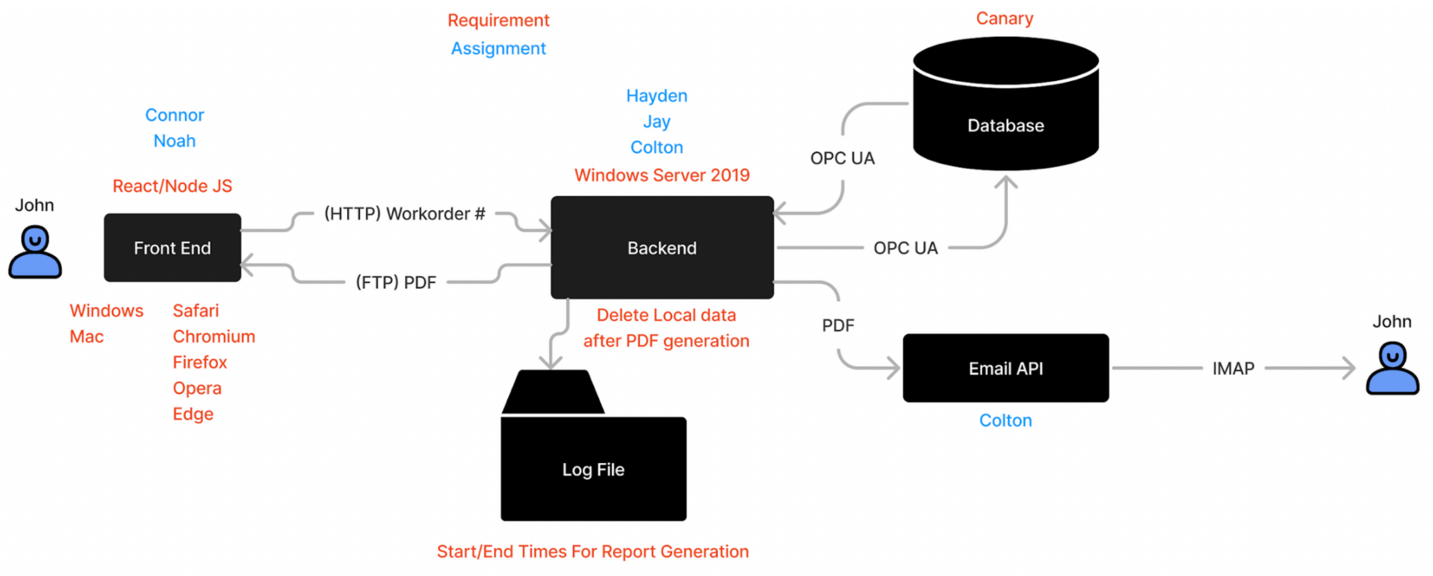
Front End *= React website where users will be able to input work orders and be displayed results.*

Backend *= ASP.Net Core web API hosted on a Windows Server 2019.*

Database *= Canary Database storing all work order information.*

Above is a very simplified version of our software architecture. The user will interact with our frontend which will make calls to our backend using our chosen APIs. In the backend a report will be generated and then formatted into a simple, easy-to-read PDF. The PDF will then be sent back to the frontend for the user to view, as well as emailed to the user if needed.

4.3.2 Detailed Design and Visual(s)



Frontend

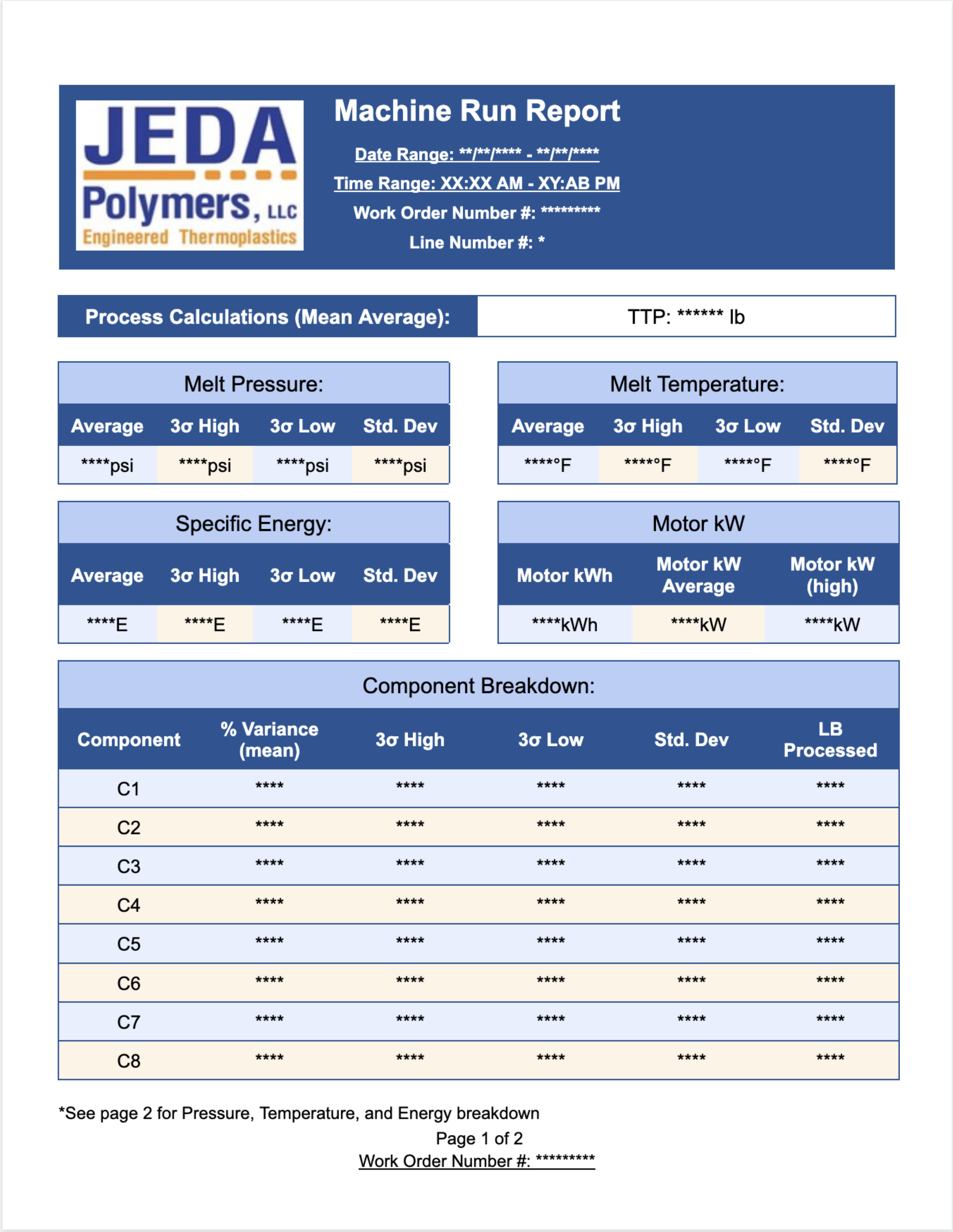
Our frontend will be written using React. It will connect to the backend using our REST API. There will be several pages: Log in, Request Data, Results Page, and New Account/Forgot Password pages. These will have to be a responsive UI that works across all browsers as well as operating systems. The use of React hooks will help make this process clean and concise.

Backend

The backend will be written in C# and ASP.NET Core 6.0. This is where the REST APIs will be programmed along with a few other subcomponents. The first subcomponent is PDF generation. To generate a PDF, QuestPDF is being used. QuestPDF is a modern .NET library for PDF document generation. The next subcomponent is working with a Canary database. The database is configured on our client's server. We will be getting the data from the Canary database to generate a PDF. Once a PDF is generated, it will be sent to the front end to be viewed by the user. Another subcomponent is the log file. The log file will have start and end times for report generation. The last subcomponent is an email API. If a user on the frontend wants a generated PDF sent to emails, then the email API will send the PDF to those emails. All of this will be configured to run on Windows Server 2019 as our client requested.

PDF

Below is our template PDF, while not finalized as the client and us are making adjustments, the general format is how it will remain for the rest of the project.



JEDA Polymers, LLC
Engineered Thermoplastics

Machine Run Report

Date Range: **/**/**** - **/**/****
Time Range: XX:XX AM - XY:AB PM
Work Order Number #: *********
Line Number #: *

Process Calculations (Mean Average):	TTP: ****** lb

Melt Pressure:			
Average	3σ High	3σ Low	Std. Dev
****psi	****psi	****psi	****psi

Melt Temperature:			
Average	3σ High	3σ Low	Std. Dev
*****°F	*****°F	*****°F	*****°F

Specific Energy:			
Average	3σ High	3σ Low	Std. Dev
****E	****E	****E	****E

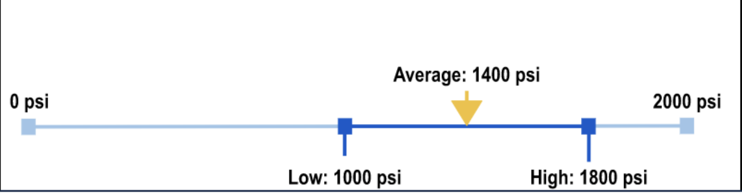
Motor kW		
Motor kWh	Motor kW Average	Motor kW (high)
****kWh	****kW	****kW

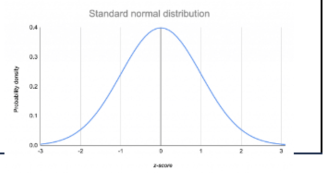
Component Breakdown:					
Component	% Variance (mean)	3σ High	3σ Low	Std. Dev	LB Processed
C1	****	****	****	****	****
C2	****	****	****	****	****
C3	****	****	****	****	****
C4	****	****	****	****	****
C5	****	****	****	****	****
C6	****	****	****	****	****
C7	****	****	****	****	****
C8	****	****	****	****	****

*See page 2 for Pressure, Temperature, and Energy breakdown

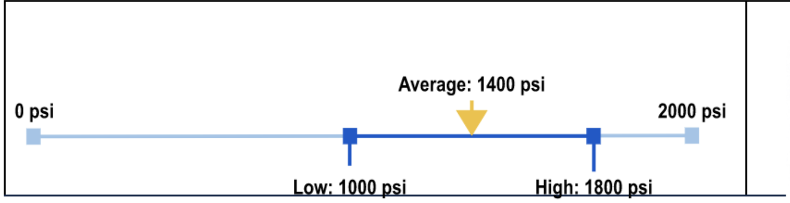
Page 1 of 2
Work Order Number #: *********

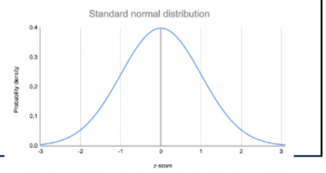
Melt Pressure:



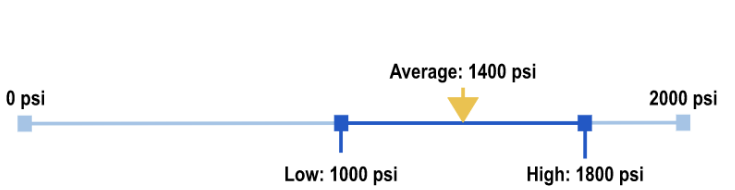


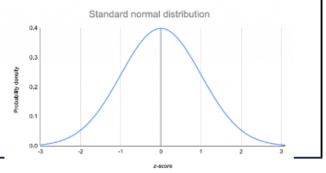
Melt Temperature:





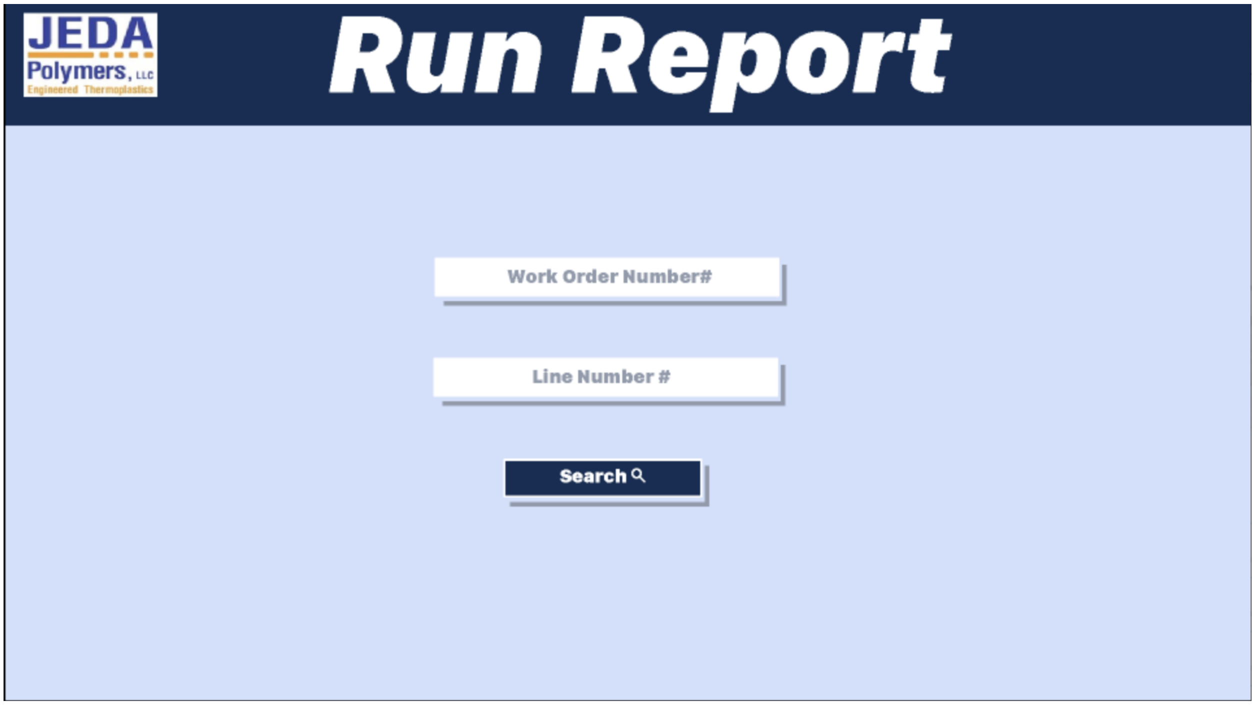
Specific Energy:

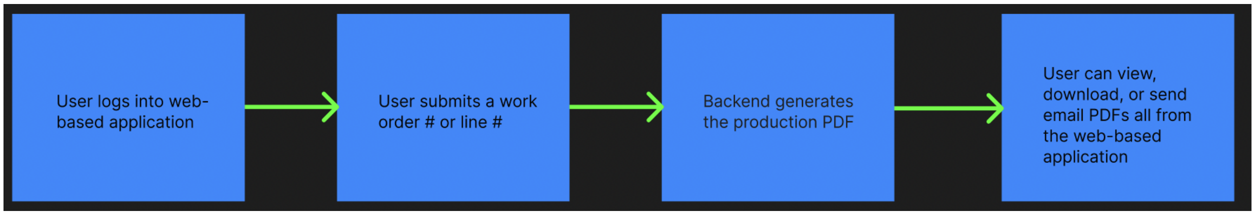




4.3.3 Functionality

This project is designed to work on Chrome, Firefox, Safari, Bravo, and Edge web browsers. Let's go over a scenario with a user and real world context. Our client wants to generate a production report to show to a customer. The user can load up their browser of choice, and type in the web address for the application. Once the user is loaded into the application, they will be greeted with a login page. After logging in, the user can generate a report with a work order number or a line number. A screenshot of a mock screen is shown below. Once one of those is submitted, a PDF production report will be generated. The PDF can be viewed front the website, downloaded or sent to specified emails.




User logs into web-based application
User submits a work order # or line #
Backend generates the production PDF
User can view, download, or send email PDFs all from the web-based application

This timeline above shows the intended real world use of the project.

4.3.4 Areas of Concern and Development

Our current design includes all of the user's needs as well as a few additional features. The only potential concerns for delivering our product are the features that were requested by the client. These features include the email service and user authentication. We agreed that we had the capacity to develop these features in tandem with the main requirements. Currently, we are not concerned that these additional features will not cause the overall quality of the product to decrease. Our project scope and plan will give us ample time to ensure that the quality of the product is up to our team's and the client's standards and satisfaction. As we move forward, we will remain retrospective to ensure that the original requirements are preserved. If we do run into issues regarding the additional requirements, we will meet with the client to reassess the priority of these features.

4.4 Technology Considerations

ASP.net Core 6.0 Web API will be implemented as our backend service. Since it is in C#, it will run natively on windows 2019 servers, which is what our client is wanting to host the server on. ASP.Net is used by several companies in the real world so third party documentation/tutorials are plentiful while maintaining similarities to Spring Boot, a framework we used in SE 309. ASP.Net also has native and easy ways to port our microservices over to docker containers so in unlikely event the client decides to switch from one operating system to another all we would need to do is switch to using docker without losing any of our progress.

QuestPDP is an open source .NET PDF generation library. We like it because of its extensive documentation, including several examples, making learning how to use the framework easier than its competitors. Because the library runs natively in C#, we do not have to worry about installing third party packages or other programming languages when we deploy the final solution creating an easier experience for our client.

Canary Database is the current database JEDA polymers is using to store the data from their production lines. Because this was given to us we did not get a say in choosing a database to use. However, Canary has full documentation online which will make integrating it into our project relatively easy.

React is a user-friendly framework for creating web-applications. This was an easy choice to use on our frontend because several of us already had experience using React. Compared to other languages, it keeps up in speed and makes working on different browsers and operating systems seamless.

GitLab is a DevOps platform that helps automate builds, verification, and integration of code. It will be used as our repo to hold all the code. It will have a build pipeline, issues, and merge requests. Other similar platforms are Azure DevOps, Jenkins, Google Cloud, and more. Gitlab is an industry standard. It is an all in one platform compared to Jenkins which doesn't have release, packaging, monitoring, and configuration services.

4.5 Design Analysis

Our basic implementation of ASP.Net has proven why we chose it. Once setup was completed there were no more additional installations to get started on our project. Everyone can work on their own microservice without having to worry about our code being changed by another developer's microservice.

QuestPDF has proved fruitful. Initially it was very difficult to understand how their controllers/components work in the library but once we figured it out the development has continued without any other roadblocks. This would have been a larger issue if we chose a library with little to no documentation.

As a team we also set up a GitLab to help with version control and code integration. Git is pretty standard across most coding projects. Since most of the group has already used this before, Git was a no brainer for our version control in this project. We added security protocols such as not being able to push directly to main, and requiring merges to have at least one approval.

Future implementations for the frontend will include all of the different pages a user might encounter. The plan is to construct the front end using Postman to test endpoints. Then doing actual user testing by having the backend group go through the site to QA test. After we are confident that there are no bugs we will start testing our calls using the REST APIs.

We believe that all of these different implementations in the project are feasible. We are a group of competent engineers and we all have some experience and knowledge in different aspects of this project to make sure this gets done in our given timeline. If anyone has issues with their code, we have good enough communication as a team to get that problem resolved and continue progressing.

5 Testing

Mindset

Our team is going to build the product in a test-driven development environment. Test-drive development is a style of programming which includes three tightly interwoven activities: coding, testing, and design. The benefits of doing this is that it usually reduces the amount of defect rates.

Test-driven development follows a set of rules to work as efficiently as possible. The first rule is to write a single unit test that describes an aspect of the program. Next, run the test. This will obviously fail because the code base doesn't have the feature yet. After the first test, write the minimal and simplest amount of code to make the test pass. After the second test passes, refactor the code so that every line, method, and class has a distinct purpose, idea, or responsibility. The refactored code should also have a minimum number of components, and it should have no duplication. Once the refactored code is complete, keep repeating until the program is finished with all of the features.

The team believes this is the best way to develop the product. The set of rules will keep development in progress without running into poorly written code that can't pass tests. The rules make sure that every single line of code has a purpose. TDD is a great style of programming for this project.

5.1 Unit Testing

Front-End

Unit testing for the frontend will consist of utilizing Jest which is a viral library that allows developers to conduct unit tests on React. By using Jest's watch mode, we will be able to generate snapshots of our code which is then compared to previous snapshots. If the snapshots differ, we will be able to trace where in our code the tests failed, or produced unintended results. These snapshots will be vital when conducting unit tests because they help us identify areas of our frontend where components changed where they should not have.

The units that we will be testing are the user login page, generate report page, and the generated report page. These three components must all perform correctly in order for our product to work. By ensuring that these all are thoroughly unit tested, we can deliver a product that is up to the standards of the team and client.

An example of a unit test that we will run would be creating a snapshot of the generated report, then generating the same report again. By comparing the snapshots, we can verify that the components display the correct data and are working as intended.

Back-End

Unit testing for the backend will be pretty simple. ASP.NET has the ability to create unit tests for controller actions. Since we will most likely only have one controller, we will only have to create one test controller. The test controller will check to see if the data is correct and properly formatted. To test the log file, we will just feed mock data to the log file. If the data being sent and the data showing up in the log file match, then we know the log file is complete.

An example of a unit test would be creating a test controller for the Work Order controller. This will consist of making a Work Order test object with specific data. Then we can do a check to see if the API call of the Work Order test object is equal to the expectation.

5.2 Interface Testing

For interface testing there are numerous different routes we could take, user testing is going to be our first main form of interface testing. The plan moving forward will be to have both a mix of engineers and people who have never coded before.These groups will comb through the website to test every link and endpoint available. The mixed demographic is important here because engineers are nice for checking edge cases and things that will most likely break the site. Meanwhile, having a non-engineering perspective can be nice for testing the usability of the site and receiving input on how the common user feels about the interface.

Additionally, any formal testing will be completed using a framework called LambdaTest. Lambda test makes automated GUI testing easy and fast. This may not end up being necessary, but additional testing never hurts. With the majority of our code we will test on a regular basis to avoid any foundational errors. However, the majority of the testing will take place after the interface is already made due to the nature of interface testing.

5.3 Integration Testing

For integration testing we plan on using Robot Framework, it is a python library that makes automated test cases easy to run with easy to interpret output. One of our group members is also very familiar with RF, so it will reduce the learning curve.

We found our critical integration pathways by looking at our software architecture as well as our object structure. A key integration test would be linking the front end request button to the back endPDF calls.

List of some of our critical integration parts:

(Front End) Request PDF with (Back End) PDF Generation

Pulling Data and PDF Generation

(Front End) Email PDF with (Back End) Email

PDF Generation with Log File

A sample test case for PDF generation and log file integration (written in robot framework style code):

```
PDF Gen Log File Test
        Generate PDF          ${sWorkOrder}
        @{sLogFile}=          Read Log File
        RUN KEYWORD IF        '${sLogFile}' not in @{sLogFile}        Fail        'Work Order not found in file'
```

5.4 System Testing

Since system testing involves the entire scope of the system we will have to try "end to end testing". This typically means that we will be testing different inputs on various parts of the system to make sure that they yield the desired output. For example, a test input would be given to the front end of the system while a desired output would be tested on the back end of the system. Since multiple tests will have already been done in order to make sure that subsystem outputs are correct, this would allow us to check to make sure that an input can travel through the entire system without throwing any errors or producing an incorrect output. While the majority of our tests will be written to test individual components, after we finish our integration we will run end-to-end tests to make sure that the system works correctly.

We plan to use the versatile framework "Robot Framework" as mentioned in previous sections. For more information on how we plan to integrate Robot Framework see the integration testing portion of the document. For more information on how we plan to receive and use the output of Robot Framework see the results portion of the document.

5.5 Regression Testing

JEDA Polymers doesn't have any current report generation for us to add on/modify.

5.6 Acceptance Testing

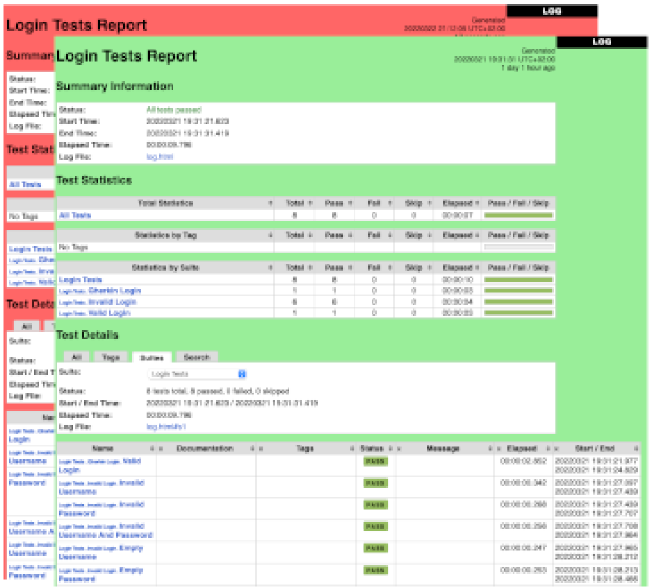
This portion of testing will consist of two things. The first form of demonstration will be going through the requirements list. Each functional and nonfunctional requirement will be reviewed to see if it is being met. The last form of demonstration is to show the client a “beta” version of the product. This will ensure that the product has all of the requirements desired by the client. Although most of the product is already done, we might be able to change some small things that the client desires. For the most part it will be demonstrating to the client that the product meets the requirements list.

5.7 Security Testing (if applicable)

Security testing is not applicable for this project.

5.8 Results

Our results from Robot Framework will be automatically formatted into a report, log, and output file, below is a picture from Robot Framework’s website where they display a test result. We can see that it breaks down test cases into steps, time for each step and case, inputs and outputs, as well as reasons for failures. This makes it very easy to interpret the results, and when we want to show our client our testing results it will not be complicated for them to understand.



Login Tests Report

Generated
20220321 19:31:31 UTC+02:00
1 day 1 hour ago

Summary Information

Status:	All tests passed
Start Time:	20220321 19:31:21.623
End Time:	20220321 19:31:31.419
Elapsed Time:	00:00:09.796
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	8	8	0	0	00:00:07	🟩

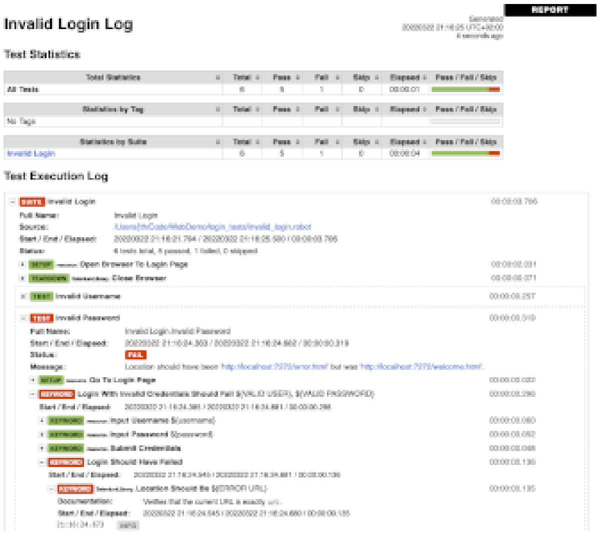
Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Login Tests	8	8	0	0	00:00:10	🟩
Login Tests . Gherkin Login	1	1	0	0	00:00:03	🟩
Login Tests . Invalid Login	6	6	0	0	00:00:04	🟩
Login Tests . Valid Login	1	1	0	0	00:00:03	🟩

Test Details

Suite:	Login Tests
Status:	8 tests total, 8 passed, 0 failed, 0 skipped
Start / End Time:	20220321 19:31:21.623 / 20220321 19:31:31.419
Elapsed Time:	00:00:09.796
Log File:	log.html#s1

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
Login Tests . Gherkin Login . Valid Login			PASS		00:00:02.852	20220321 19:31:21.977 20220321 19:31:24.829
Login Tests . Invalid Login . Invalid Username			PASS		00:00:00.342	20220321 19:31:27.097 20220321 19:31:27.439
Login Tests . Invalid Login . Invalid Password			PASS		00:00:00.268	20220321 19:31:27.439 20220321 19:31:27.707
Login Tests . Invalid Login . Invalid Username And Password			PASS		00:00:00.256	20220321 19:31:27.708 20220321 19:31:27.964
Login Tests . Invalid Login . Empty Username			PASS		00:00:00.247	20220321 19:31:27.965 20220321 19:31:28.212
Login Tests . Invalid Login . Empty Password			PASS		00:00:00.253	20220321 19:31:28.213 20220321 19:31:28.466



Invalid Login Log

Generated
20220322 21:16:25 UTC+02:00
4 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	6	5	1	0	00:00:01	🟩🟥

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Invalid Login	6	5	1	0	00:00:04	🟩🟥

Test Execution Log

```
- SUITE Invalid Login                                                                   00:00:03.796
  Full Name:            Invalid Login
  Source:               /Users/jth/Code/WebDemo/login_tests/invalid_login.robot
  Start / End / Elapsed: 20220322 21:16:21.794 / 20220322 21:16:25.590 / 00:00:03.796
  Status:               6 tests total, 5 passed, 1 failed, 0 skipped
  + SETUP Open Browser To Login Page                                                    00:00:02.031
  + TEARDOWN SeleniumLibrary . Close Browser                                            00:00:00.071
  + TEST Invalid Username                                                               00:00:00.257
  - TEST Invalid Password                                                               00:00:00.319
    Full Name:            Invalid Login.Invalid Password
    Start / End / Elapsed: 20220322 21:16:24.363 / 20220322 21:16:24.682 / 00:00:00.319
    Status:               FAIL
    Message:              Location should have been 'http://localhost:7272/error.html' but was 'http://localhost:7272/welcome.html'.
    + SETUP Go To Login Page                                                            00:00:00.022
    - KEYWORD Login With Invalid Credentials Should Fail ${VALID USER}, ${VALID PASSWORD} 00:00:00.296
      Start / End / Elapsed: 20220322 21:16:24.385 / 20220322 21:16:24.681 / 00:00:00.296
      + KEYWORD Input Username ${username}                                              00:00:00.060
      + KEYWORD Input Password ${password}                                              00:00:00.052
      + KEYWORD Submit Credentials                                                      00:00:00.048
      - KEYWORD Login Should Have Failed                                                00:00:00.136
        Start / End / Elapsed: 20220322 21:16:24.545 / 20220322 21:16:24.681 / 00:00:00.136
        - KEYWORD SeleniumLibrary . Location Should Be ${ERROR URL}                     00:00:00.135
          Documentation:        Verifies that the current URL is exactly url.
          Start / End / Elapsed: 20220322 21:16:24.545 / 20220322 21:16:24.680 / 00:00:00.135
          21:16:24.573  INFO
```

Testing will ensure compliance with our requirements as all of our requirements are testable, and if we add any non testable requirements we can add a FIT criteria which will make it testable.

In theory we should be able to have a test suite/case for every individual requirement, and with RF’s auto formatting the results will be easy to verify we meet with the specified requirements.

6 Implementation

The plan is to follow the Gantt chart from section 3.4. Back-end and front-end can work parallel to each other. Once both ends think the functionality is done, the plan is to test the front-end and the back-end together before implementing the code onto the server. This way we can find out bugs and improve the code before bringing more variables into the setup. Once we know the front-end and back-end communicate flawlessly, we can implement the code onto the server. This way we know that any issues are related to the server and not to the code. After that is working, we will start our testing.

7 Professional Responsibility

This discussion is with respect to the paper titled “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Table 1. The seven areas of professional responsibility in the assessment instrument(Modified)

Area of responsibility	**Definition**	**NSPE Canon**	**IEEE Computer Society / ACM Code of Ethics for Software Engineers**	**Importance**	**Performance**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Principle 1 of the SE code of ethics states that, among other things, workers should accept responsibility for their own work, approve software if it is up to certain standards, and report to the correct authorities it is not. The only similarity that this section mainly shares with the NSPE cannon is that they both mention that deceptive acts should always be avoided.	High: We are expected as a team to perform to the best of our abilities to meet all of our customers needs/wants in a timely and professional manner while upholding ethical pillars like honesty.	High: The team has strived to meet all goals set by the customer in order to meet all expectations, wants, and needs.
Financial Responsibility	Deliver products and services of realizable value and	Act for each employer or client as faithful agents or	Principle 2 of the SE code of ethics states that, among other things, workers should be	Low: Our group did not require any specific funding as the project does not	Low: since the project has not required any funding as of this current moment in

	at reasonable costs.	trustees.	completely honest and forthcoming about any issues on the project such as lack of schooling/experience, funding, issues with intellectual property law, or any other problematic topics. This section differs from the NSPE cannon as it specifically lists ways in which a person should not hurt their employer or client by accepting detrimental work or simultaneously committing illegal practices.	require the use of any physical components and the team is not on any form of salary.	time.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	Principles 1 and 8 mention that a worker should always be truthful to specifically the client, employer, and public. This results in being more specific than the NSPE canon.	High: all issues with the project should be communicated with the client (customer) and employer (advisor). All statements made should be shared in complete honesty and in great detail.	High: all members of the team have strived to be open with both the client (customer) and employer (advisor) about any issues that the project development has had.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Principle 8 of the SE code of ethics states that, among other things, workers should make code that is safe, reliable and does not harm the customer/client. Principle 1 also mentions that any work that might compromise	Medium: while there is little to no chance the software can create physical harm to the user it still can affect the clients business and business relations.	Medium: Work has been done by the team to ensure that the software meets all of the client's current standards but more work must be done in order to finish the project.

			health, safety, and Well-being of the public will be reported. This section differs from the NSPE cannon as it talks more about more than just the well being of the public and also talks in terms of worker/client relations.		
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Principles 1 and 5 both say that the property of the client should be respected and that intellectual property of any other party should not be used in the context of the project. This section differs from the NSPE section greatly as it talks more about the idea of “property” than acting on behalf of others.	Low: We will not have access to specific company strategies or data that the client owns. All prospects of the project will be authored from scratch.	Low: very little work has been done on the project in terms of these areas of the project.
Sustainability	Protect environment and natural resources locally and globally.		Principle 1 briefly mentions that software should not harm the environment. This contrasts the NSPE’s lack of information on the topic.	Low: the only environmental impact that our software can have is use of electricity which will be quite low as the software will not require a large amount of power.	Low: the project is currently not far enough into development in order to obtain enough information to determine the success of this section.
Social	Produce products and services that	Conduct themselves honorably,	Principles 1,5, and 8 all make mention of the	High: The product should be designed in order	High: the product is being developed to be the best

Responsibility	benefit society and communities.	responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	workers responsibility to society/the public, in terms of honesty, responsibility, reputation, possession, usefulness, among others. It also goes a lot more in depth than the NSPE in terms of these topics.	to solely benefit the customer by providing them their wants and needs. No practices should be committed out of any form of malicious intent.	possible version for the clients wants/needs. It is being designed to be a useful software that upholds all ethical related topics.

7.2 Project Specific Professional Responsibility Areas

See the blue column in the table above.

7.3 Most Applicable Professional Responsibility Area

See the green column in the table above.

8 Closing Material

8.1 Discussion

The result of this project will be a website accessible by members of the JEDA Polymers team, which will accurately and quickly generate and display a report for their product (plastic polymers). Our client has already looked at all of our design docs and we have met with him to receive input on any changes he would like to see, and verification after that the modified designs are what he is expecting.

We have laid out our requirements and sent them to our client for approval, currently our solution does not meet all requirements but as it is in the early prototype phase and we have not started working on the finalized product yet it is ok. As we get closer to the final product we will be getting closer and closer to 100% of requirements being met. That is our end goal and anything less would be a let down to the team and our client.

8.2 Conclusion

Overall work done so far for the project was creation of our software architecture, collecting and refining requirements, better general domain knowledge, PDF template design and approval, and research into technologies that will be used in this project. More specific work done this semester are figma designs of the front end, verified by our client. The backend has created a working prototype for PDF generation (with randomly created data).

Some of our major blockers had to do with creating a PDF template that our client liked. This was due to a general lack of domain knowledge. Another big blocker was mac devices not being able to connect to the server JEDA Polymers is using, we solved this by speaking to John Deeken. If we could do anything differently it would be to attempt everything sooner, when we wanted to have a day to work on the server was when we realized we couldn’t connect via Macs, so some better prep would have put us in a better position.

Next semester can be broken down into 3 sections. Firstly we are going to work on developing our product, the frontend and backend will be working in parallel to complete their sections around mid February with integration and testing hopefully being completed by mid March. The second section is giving the beta to John and the team at JEDA Polymers to give a trial run and see any issues/bugs they discover or potential changes they would like to see. Finally, the third phase will be implementing these bug fixes and changes so our customer ends with a product they like.

8.3 References

"IEEE Standard for Software and System Test Documentation," in IEEE Std 829-2008 , vol., no., pp.1-150, 18 July 2008, doi: 10.1109/IEEESTD.2008.4578383.

"IEEE Standard Specifications for Public-Key Cryptography," in IEEE Std 1363-2000 , vol., no., pp.1-228, 29 Aug. 2000, doi: 10.1109/IEEESTD.2000.92292.

"ISO/IEC/IEEE International Standard - Systems and software engineering — Developing information for users in an agile environment," in ISO/IEC/IEEE 26515:2018(E) , vol., no., pp.1-32, 20 Dec. 2018, doi: 10.1109/IEEESTD.2018.8584455.

Lutz, R. (2022, September 8th). Lecture 6, Communicating the Requirements, EARS [PowerPoint slides].

Marvin, A., Wilkinson, P., Harwood, A., & Novak, M. (2009, November 17). *Easy approach to requirements syntax (EARS)*. IEEE Xplore. Retrieved September 20, 2022, from https://ieeexplore.ieee.org/document/5328509

8.4 Appendices

8.4.1 Team Contract

Team Name: sdmay23-06 (Production Run Reports)

Team Members:

1) Connor Linn 2) Hayden Havelka

3) Jay Arnold 4) Noah Meyer

5) Colton Carlson

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Wednesday's 2-3pm Library/Virtual

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e mail, phone, app, face-to-face):

    a. Face-To-Face /Text/Discord

3. Decision-making policy (e.g., consensus, majority vote):

    a. Majority

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived): Noah keeps Meeting Minutes stored on google drive

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. Expected to show up/participate unless circumstances arise.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. Equal, agree on work when it is given out, speak up if you realize you won't be able to meet deadlines.

3. Expected level of communication with other team members:

    a. At least once per week in early stages, more often later.

4. Expected level of commitment to team decisions and tasks:

    a. Average of 5-10 hours a week, depending on deadlines and workload

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

    a. Scrum Master: Colton

    b. Client Interaction: Colton

    c. Component Design: Everybody

    d. Testing: Jay (Lead)

    e. Meeting Notes: Noah

    f. Front-End: Connor (Lead)

    g. Back-End: Hayden (Lead)

    h. Email/Query-Server: Colton (Lead)

2. Strategies for supporting and guiding the work of all team members:

    a. Weekly meetings

    b. Communicating often when hitting roadblocks

3. Strategies for recognizing the contributions of all team members:

    a. Assign user stories to specific people to keep track of programming contribution.

i. 1 pat on back per user story

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

    a. Connor:

        i. Worked with front-end web design at previous internship.

    b. Hayden

        i. Worked with back-end/sql

        ii. Worked on self-made projects from start to finish

    c. Noah

        i. Experience working in professional environments.

    d. Jay

        i. Experience with AWS/Servers

    e. Colton

        i. Worked with production run reports at previous internship.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

    a. We all are friends so no negative pressure

    b. Many a pat on back for good ideas

    c. More you do faster project goes

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

    a. Talk with other group members, privately or publicly, and with TA if needed

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

    a. Design docs for project

b. All requirements gathered

i. Client Approved List

c. Meet Course Requirements

d. Some form of prototype

i. Optional

2. Strategies for planning and assigning individual and teamwork:

a. Agile Development Process

b. Weekly Meetings (Standups)

3. Strategies for keeping on task:

a. Positive Reinforcement

b. Staying ahead of work

c. Don’t want to let group down.

d. Personal Ownership of project

e. Strong communication when stuck

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

a. Meet as team to discuss

b. If continued, talk with advisor

c. If continued, again, one more team session with advisor

2. What will your team do if the infractions continue?

a. Removal From Team

i. Optional Trial by Combat

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.* b) *I understand that I am obligated to abide by these terms and conditions.* c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Colton Carlson DATE 11/30/2022

2) Connor Linn DATE 11/30/2022

3) Hayden Havelka DATE 11/30/2022

4) Noah Meyer DATE 11/30/2022

5) Jay Arnold DATE 11/30/2022