**CSIT 425 Software Engineering**

**Final Project**

**Railroad Manifest Project**

**Group 4**

**Taylor Zielinski**

**Chelsie Yan**

**Joshua Oshiro**

**Kenneth Price**

**Table of Contents**

**1. Introduction – 3**

Customer Problem Statement – 3

Stakeholders – 3

Goals – 3, 4

Group Bios – 4

**2. Requirements Engineering - 5**

Plan for requirements elicitation – 5

Functional Requirements – 5, 6

Non-functional Requirements – 6

Use-Cases – 7

**3. Design and Implementation - 8**

Task descriptions/task assignment – 8

Deliverables/milestones(timeline) – 8

System Diagrams – 9

Overview of implementation plan – 10

**4. Validation - 11**

Overview of testing plan – 11

Features to be tested – 11

Test Cases – 11

Testing Schedule – 12

**5. Conclusion - 13**

**1. Introduction**

**Customer Problem Statement**

The problem we were presented with is to create a web form and (potentially) a database. The company is a small regional railroad who is revitalizing an abandoned rail line that extends from Fredonia, south for roughly 35 miles. The company has 8 customers currently and they need to be able to order railcars to be delivered and picked up when empty. Currently the customers employ a fleet of trucks to unload the materials at the rail yard and deliver them to the factory. This is extremely costly and having the rail line deliver cars directly to the facility will save a lot of time and money.

The company needs a system where customers can request railcars deliveries via an online form. This system will also schedule the train for delivery, printing out a manifest for the train conductor every morning. The company charges $750 for delivery of a railcar and $300 to pickup an empty railcar. This information will need to be compiled and processed to the billing department every day that deliveries are being made. The company is only able to run one train per day, with a minimum of 5 cars and a maximum of 14 cars. This information needs to be displayed to customers so that they can schedule a delivery date that will work.

**Stakeholders**

The main stakeholders in this project would be the company we are designing the program for. Other stakeholders include the other companies which interact with the rail car delivery service.

**Goals**

The software we will be creating will allow the railroad’s customers to select the type of rail car they want, insert their payment information, and choose if they want empty car removal. The customer will then receive a generated 4-digit alphanumeric identification. Our software will then send the collected information to the client’s financial department as well as the conductor. In addition to this, we have received additional information in which the program must also allow customers to order “rush” delivery for an additional fee, as the train has been filling up very fast.

**Group Bios**

Joshua Oshiro is a Computer Science major in his senior year. The languages he currently focuses on are C++ and Python. His focus is software development, but he is considering branching out into data science in his spare time. His goal is developing real-time graphics tools for both studio animation and video games. In addition to programming he also likes to draw and play video games with his friends.

Chelsie Yan is a senior year Computer Science major. She is proficient in C++, HTML/CSS, and C#. She plans on gaining more experience with Python and other various scripting languages. She focuses on software development but is extremely interested in AI/machine learning. She also hopes to have a career involving animation or video games. In her free time, she enjoys playing video games, drawing, and cooking.

Kenneth Price is student at SUNY Fredonia who majors in Computer Science and minors in Data Science. He is proficient in Unity. He hopes to have a career in Esports as a team analyst.

Taylor Zielinski is also in his senior year as a Computer Science major at SUNY Fredonia. He is proficient in both Java and Python. In his final semesters at Fredonia, Taylor is attempting to round out his knowledge of Computer Science. This semester he has been furthering his knowledge of C++, C#, and UNIX. He is unsure what he would like to do specifically but is excited to work in the field. In his spare time, he tries to stay active with activities such as skateboarding, playing basketball, and riding his dirt-bike.

**2. Requirements Engineering**

**Plan for Requirements Elicitation**

Through the creation of our product, our team will attend weekly meetings with our Direct Supervisor, Denise Joy, and stakeholder, Jeff Joy. During these meetings, we discuss questions and concerns regarding functional and non-functional requirements. We found the use of email communication greatly effective for requirements elicitation as our Direct Supervisor and stakeholder always replied within twenty-four hours if not sooner. Our team has documented product progress on GitHub. All additional requirements will be added as they arise. We plan to interview the eight customers who will use our product. Some sample questions include:

* Is our form easy to navigate? If not, what did you find confusing?
* In your opinion does using this form provide ease of access for this railroad service?
* Is there a feature you would like to see added?

We hope to pinpoint improvements for the user interface and allow scalability in the event that more customers will use our product.

**Functional Requirements**

Display correct information to the customer:

The customer name, customer ID, railcar ID, priority status, pickup status, and total charge should all accurately display on the form for the customer. If the customer sees they have entered incorrect information, they have the option to change the data before sending it to the train manifest. Success is achieved if the form output matches customer input.

Output correct information in the train manifest:

All information above should be accurately reflected in the train manifest. The information in the manifest is vital for the train conductor and billing. The manifest should be formatted so it is easy to read and understand. To decrease the mistake of duplicating requests, the form will clear text input after sending. Success is achieved when the train manifest outputs the correct customer information.

Ability to prepend in the manifest if priority:

If priority status is checked, the request should prepend to the beginning of the train manifest. The conductor must know what railcar request are set to priority as there is an extra fee associated with it. Success is achieved if all priority requests are at the beginning of the train manifest, regardless of the chronological order submitted.

**Non-Functional Requirements**

User Friendly User Interface

The UI should be simple to navigate for our customers, regardless of their technical prowess. We aim to minimize unnecessary aspects. The visuals on the form should be easy for customers to read. Success will be assessed based on the amount of assistance needed for users to completely understand the UI. This is a non-functional operational requirement.

Maintainability

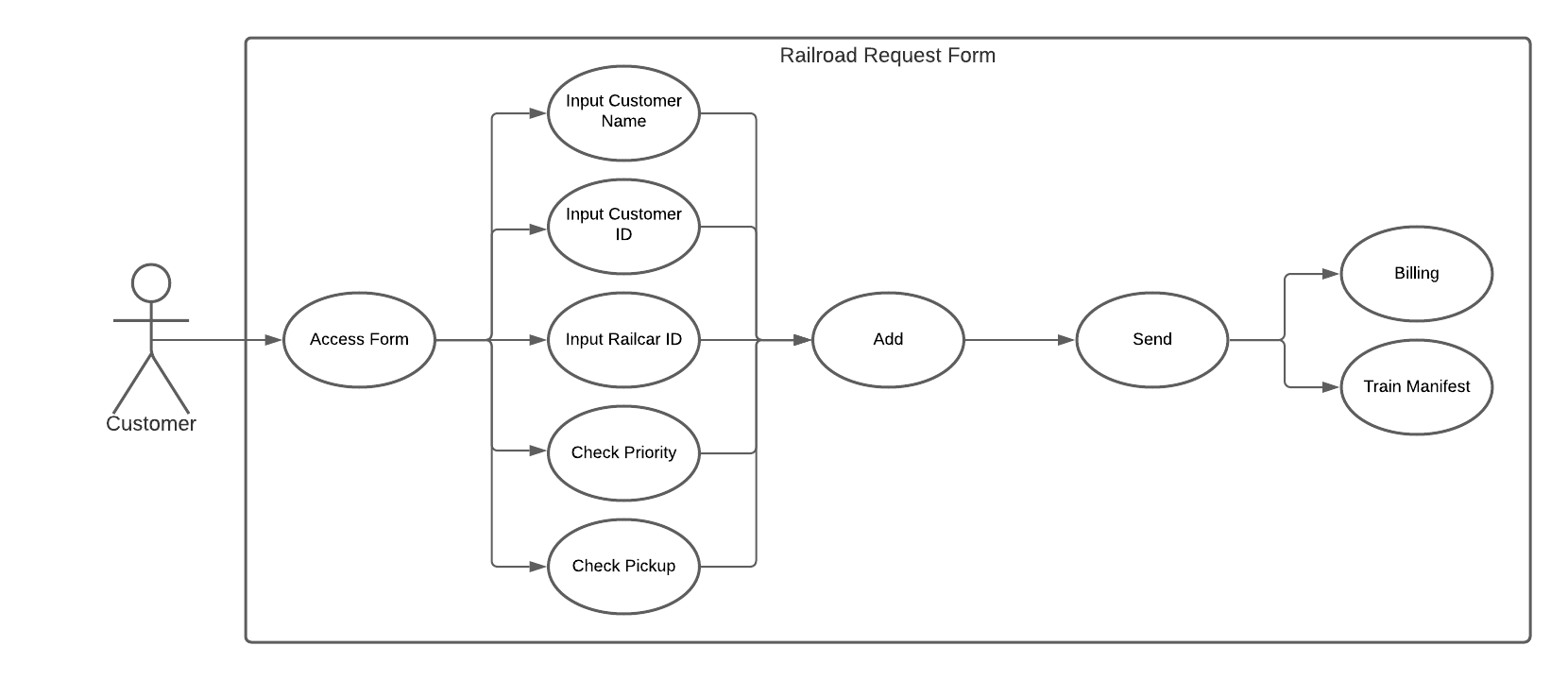
As a team, we plan to annotate and logically organize our code. This will support maintainability if new members are added during or after the product creation. We regularly update our project on GitHub. One can see exactly which lines of code are added and deleted with each patch or version. Success is achieved if the product is able to easily change through enhancements or backtrack to older versions. This is a non-functional product requirement.

Scalability

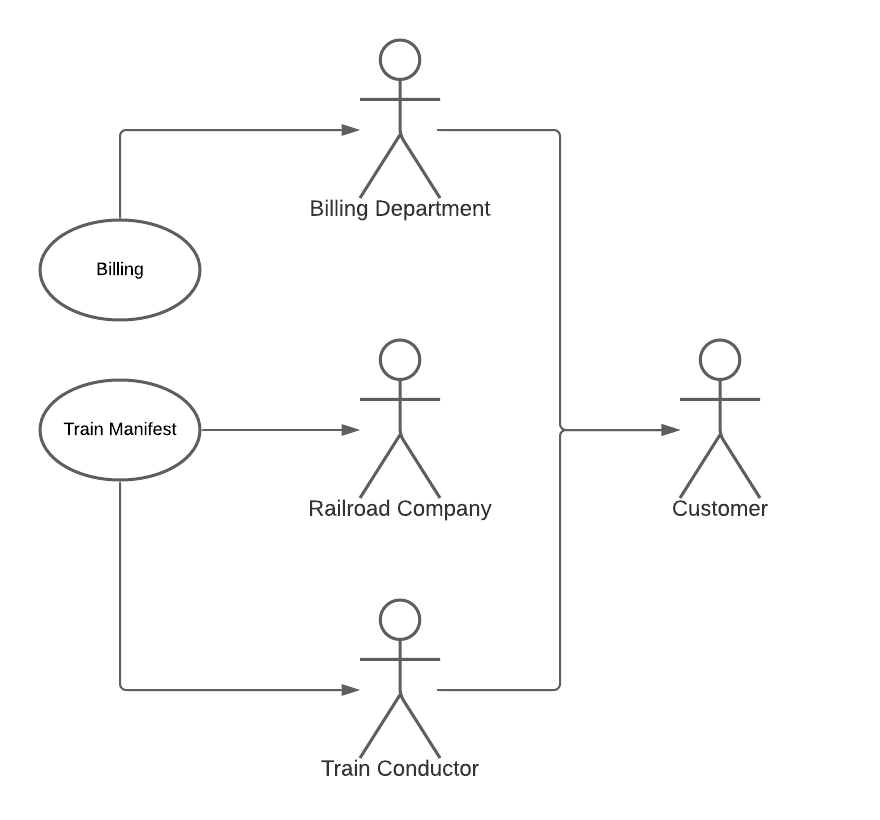
The total number of customers may change at any time. If the railroad service gains or loses customers, we aim for our product to easily adapt. Cross functionality with a database should also be straightforward to implement. Success is achieved if the product can handle a smaller or larger number of total customers. This is a non-functional product requirement.

**Use Cases**

Customer creates and sends a request that is output to the Train Manifest the Billing document:



The Train Manifest is sent to the Railroad Company and Train Conductor and the Billing is sent to the Billing Department. The Train Conductor and Billing Department both interact with the Customer:



**3. Design and Implementation**

**Task descriptions/task assignment**

Project Lead: Taylor Zielinski

The project lead manages the entire team and keeps in close contact with the Scrum Master, Prof Joy. This work consists of splitting up the team working and ensuring the team meets all deadlines and milestones. This member will also code in their spare time.

UI & Error Checking: Chelsie Yan

The UI designer will engineer a suitable user interface that will enhance the customer’s experience. All input checking will be managed by this member to ensure all data is consistent. Additional bug-fixing and feature implementation will be assigned to this member.

System Design: Joshua Oshiro

The system designer will design and implement the main classes and data structures used for the program basic tasks. This member will oversee all data changes and ensure that a correct output is generated. Additional bug-fixing and feature implementation will also be assigned to this member.

Report Editor: Kenneth Price

The report editor will organize and write the design documentation required for the final deadline. This member will edit and ensure all information within the design document remains consistent and free of spelling or grammatical errors.

**Deliverables/milestones (timeline)**

Sunday October 18th, 2020: Part 1 of write-up due

Friday October 30th, 2020: v1.0 Demo

Sunday November 8th, 2020: Testing strategy plan

Sunday November 29th, 2020: v1.5 due

**System Diagrams**

Diagram 1:

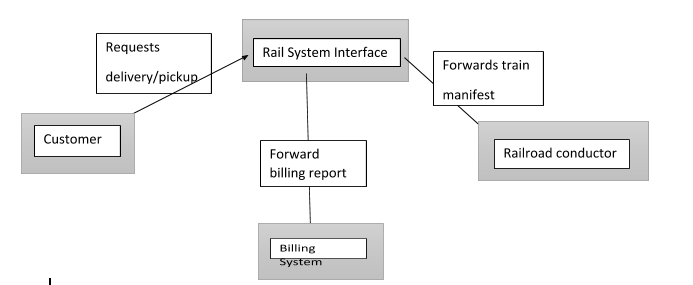
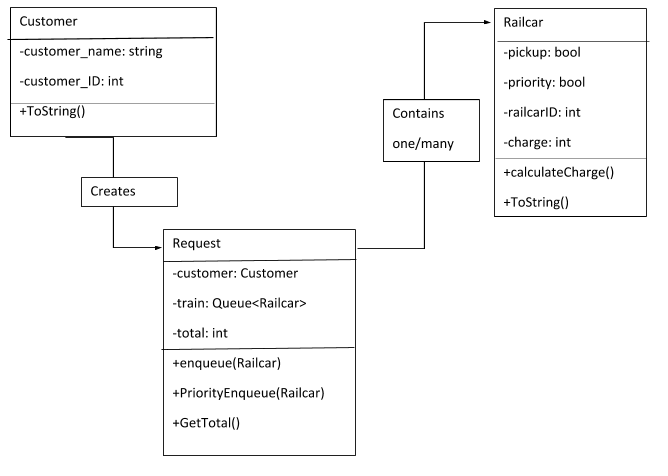


Diagram 2:



**Brief overview of implementation plan**

C# and the .Net framework will be the project’s main programming language and framework. Each of the teams lead programmers have experience with this development stack from a previous class. In addition, the .Net framework allows for easy form application creation with the default “Windows Forms App” project template. This allows for quick prototyping of the user interface and design.

Chelsie Yan is the designer of the user interface, allowing her to take advantage of the benefits the .Net framework offers while remaining consistent with her background. Chelsie also implements the error checking that will occur upon user input.

Joshua Oshiro is the system designer, implementing the data structures and classes required to perform the task. In addition, Joshua is the lead debugger of the project, tackling most bugs that occur within the development process.

Taylor Zielinski is the output designer, implementing the text output functions. The text output will also be correctly designed to allow for an easy-read view for both the train conductor and billing department. This will be Taylor’s largest focus as he is also the Project Lead.

Any additional bugfixes and feature implementation will be distributed amongst these members as the project develops. The final quality of the product will be decided upon by the company contact: Jeff Joy as well as the Scrum master: Denise Joy.

**4. Validation**

**Overview of testing plan**

The testing plan for this project was relatively basic. As we completed our different features we were testing them as well. This allowed us to eliminate a lot of bugs that may have caused further issues if not addressed right away.

**Features to be tested**

The main features we were looking to test were the ones involved with taking customer input and putting it into a proper manifest and billing document. The program needed to take proper input for the customer’s name, their ID number, as well as the rail car’s ID number. We tested this by first setting up our exception handlers and then fed them invalid inputs to see if they would either catch the invalid input, or throw an error in the program. Another function which was important to test was the output to text file functions. These functions were responsible for creating both the train manifest and billing documents. For the train manifest we developed tests by adding numbered rail cars into the text file. This was to ensure each group contains 14 cars as well as properly copying customer orders over without losing any lines. Being the basis of everything happening in the code, it was important to develop and thoroughly test these features.

**Test Cases**

The first and foremost thing we were testing for is functionality. We wanted to make sure the program would work properly without any errors being present. With a program which is constantly changing and adding to text files, a small error could manifest by messing with the data in unpredicted ways. We set up several test cases which would test the programs ability to properly copy data from the user input to the train manifest without losing any orders. This was important as when being utilized, if the program is dropping orders that would cost the company both time and money. During these tests we monitored both text files after each input to make sure they were updating and formatting properly. It was also important to test the billing file as if any data was dropped there, customers could potentially get their cars delivered but never be billed for it.

**Testing Schedule**

October:

We were in early development stages during the month of October. During this stage we were testing the “skeleton” of our code. We used the GUI’s ListBox to confirm that our information was being correctly copied from the input boxes.

Early November (Week 1):

In preparation for our initial demo, we created two separate programs. The first program being the GUI discussed in the first stage, We also created a second program in order to further our knowledge of handling files using c#. This second program is where we created the code to create and edit the train manifest and billing documents. Testing at this stage included checking the ListBox, train manifest, and billing document to ensure our data was being properly processed.

Mid-November (Weeks 2-3):

At this point we had reached full-speed in development. We had our GUI completed but the train manifest and billing documents ran into several formatting issues. We ran many tests each changing the format of the documents until we were able to come up with code which self-corrected the formatting on the train manifest.

Late November (Week 4):

The project entered bug-testing at this stage. We did many tests while developing in order to save time on bug testing at the end. It ended up paying off very well, as we tried many ways to crash the program and were able to fix the bugs we came across.

**5. Conclusion**

**How does the project solve the problem at hand?**

Our project solves the problem by providing the company and their associates with an easy-to-use form that will allow them to place orders. This project provides the conductor with an updated train manifest each morning, which is updated each time an order is added. The project also creates a billing document to keep track of total charges.

**Which process activities were most helpful?**

The most useful process activity to us was the planning stage. We very slowly and methodically broke the problem down into manageable pieces. We were then able to solve the smaller problems and combine them to create a more encompassing solution. We started very slowly as we took several weeks in the planning stage. It turned out to be hugely advantageous as our development process was significantly easier when dealing with smaller tasks as opposed to the problem as a whole.

**What are the stakeholder benefits? How does the organization benefit?**

The stakeholders will benefit from this project by the ease of use it will add to their business. The ability to order cars for both pickup and delivery without having to make a physical phone call will save time. This will allow stakeholders to focus more on what their business does instead of sourcing transportation. The project is also hugely beneficial to the company as it will automate their ordering process. It allows the ability for the train manifest and billing documents to essentially write themselves. This will save time and money in the future, allowing the company to focus on other parts of their business.

**Maintenance**

In order to maintain the software we will leave our contact information within the project as a readme document. This will allow whoever may edit the program to email us directly regarding changes and how to go about doing so.