Computer Science Capstone
CS 476 - Requirements Engineering

SmartState

Project Title: Event-driven MachinE Learning, Intelligent Assessor (EMELIA)

Requirements Specification

Overview:

The purpose of this document is to present the requirements specified by the client for our Event-driven MachinE Learning, Intelligent Assessor (EMELIA). This document will detail technical requirements, associated risks, and projected development schedule for EMELIA.

Team Members:
Andrew Hurst
David Rodriguez
Reed Hayashikawa
Jianxuan Yao
Jesse Rodriguez

Clients:
Rick Duarte
Jon Lewis

Capstone Mentor:
Fabio Marcos De Abreu Santos

Northern Arizona University
School of Informatics, Computing, and Cyber Systems

General Dynamics
Mission Systems

Version 2.0

Accepted as baseline requirements for the project:

Client:____________________ **Team:**____________________

Date:____________________ **Date:**____________________

Table of Contents

Table of Contents **2**

1. Introduction **3**

2. Problem Statement **5**

3.Solution Vision - EMELIA **6**

3.1 EMELIA System Architecture 6

4. Project Requirements **7**

4.1 Functional Requirements 7

4.2 Non-Functional Requirements 11

4.3 Performance Requirements 13

4.4 Environmental Requirements 13

5. Potential Risks **12**

5.1 Misclassification 15

5.2 Compatibility 16

5.3 User Training 16

5.4 Maintenance 17

6. Project Plan **18**

7. Conclusion **19**

1. Introduction

General Dynamics Missions Systems is a global aerospace and defense company that develops C4ISR solutions in all areas such as land, air, and cyber domains. C4ISR stands for Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance systems. General Dynamics develops and maintains and advanced command, control and direction-finding communications system to execute search and rescue missions. These search and rescue missions are conducted by the U.S. Coast Guard with the use of this advanced communications system. The name for the partnership project between General Dynamics and the U.S. Coast Guard is Rescue 21.

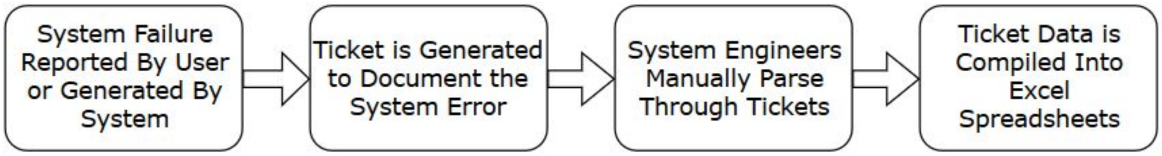
Rescue 21 is the United States Coast Guard's advanced command, control, and direction-finding communications system. This communication system encompasses all major waterways of the United States, including the West Coast, East Coast, and the Great lakes. This system was created to better locate distressed mariners and execute search and rescue (SAR) missions.

The current communications system used by Rescue 21 relies on a ticket system to generate reports regarding system or hardware failures. These tickets contain data regarding:

- Code identifier for the error
- Length of time for which the error was present
- Location for where the error occurred
- How the error was received
- Description of the error
- Description of what was impacted by the system error
- Information regarding the resolution for reported error
- The component impacted by the system failure

Each error must be classified according to the error type to assist system engineers in maintaining the Rescue 21 communications system. System engineers manually parse through the tickets in order to classify the type of error reported. This process is inefficient and reduces our client's ability to respond to system failures.

Figure 1



Our sponsor, General Dynamic Missions Systems has asked our team to create a solution to improve the Rescue 21 communications system. Rick Duarte and Jon Lewis are two systems engineers that will oversee our project. Rick and Jon are part of a larger team of engineers that oversee all of the complex communications systems for our client. Each of these systems is highly specialized and requires time and resources for maintenance and enhancement. By improving system operations and reducing outages, our client is able to better serve the U.S. Coast Guard and other systems that support the U.S. Department of Defense.

The document will be organized into six main categories. The first section will highlight the will be the introduction. The second section focuses on problem we are working to solve for our client. The following section is solution vision we have regarding EMELIA. This section will be a collection of information that details the technological tools and how each tool contributes to EMELIA's implementation. The fourth section will detail the list of environmental requirements, performance requirements, non-functional requirements, and functional requirements for EMELIA. Section five identifies the risk involved in the development of our project. The final section will detail the development plan for the project. This timeline contains all major deadlines anticipated for the development team. Our conclusion will summarize all sections to provide all high level requirements regarding the project.

2. Problem Statement

Rescue 21 utilizes hundreds of communication towers all across the United States. In order to maintain such a large infrastructure like Rescue 21 an entire team of System Engineers is necessary for the sole purpose of tracking and correcting failures within the system and creating solutions to resolve, predict, and prevent future issues. This team is the FRACAS team (Failure Reporting, Analysis, and Corrective Action Systems). The system reports failures and other errors in the form of tickets. These tickets are the primary way engineers interact with failures and document outages to the system and report system availability to the customer. Tickets are rarely unique and trend in alignment with common overarching issues. The FRACAS team categorizes these commonalities to determine outage within specific subsystems and track trends. Currently, tickets are analyzed by people, where human interaction is required for the often repetitive categorization aspect of ticket processing. An engineer will read a report, look for specific in-text references, and then categorize the ticket via several classification fields.

The current problem for our client is the time it takes for the ticket classification. Currently, the system is not automated, which requires human intervention in the ticket classification process. Manual classification of ticket data presents several issues pertaining to time and resources our client. The following bullet points detail issues regarding the ticketing process. The primary concerns with the ticket system are as follows:

- Manual classification leads to ticket classification errors
- Classification for tickets is inefficient
    - Time needed to classify ticket correctly is approximately 20 minutes
        - Time needed to correct a misclassified ticket is an additional 10 minutes
- Response time for system outages and system maintenance is extended
- Technical personnel are diverted from tasks to complete the ticket classification process
- The client has no method to observe data trends for the ticketing system

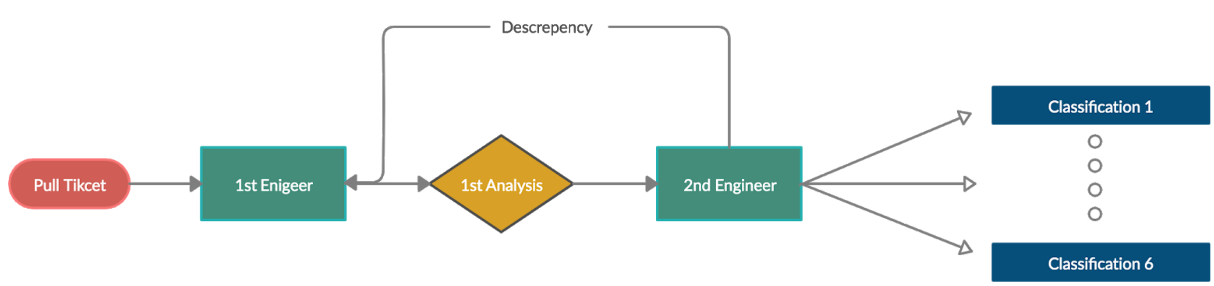


Figure 2

The figure above describes the process for engineers during the ticketing classification process. Using engineers in this process is inefficient and costly (time, Engineer focus). By automating a task like this with a simple Machine Learning model not only resources can be saved but potentially lives by keeping the Rescue 21 system functional and as efficient as possible.

3.Solution Vision - EMELIA

In order to properly address the problem at General Dynamics, we will create an Event Driven MachinE Learning Intelligent Assessor(EMELIA) that will effectively classify failure reports. With the current issue of engineers at General Dynamics manually categorizing these “tickets”, we are working closely with our clients Rick Duarte and Jon Lewis to build a system that will automate this process to be more efficient. We will begin by understanding the data that is given by the organization and try to locate specific similarities and patterns that our program can identify.

Our solution is to develop an Event-driven MachinE Learning, Intelligent Assessor (EMELIA) to classify the system failures for our client. EMELIA will be able to :

- Take in data created from the ticket system as input
- Pass the input to a neural network to begin the classification process
- Classify the input data with a 90% success rate
- Utilize a command-line interface for use by engineers
- Accept commands that will assess EMELIA’s data classification performance

3.1 EMELIA System Architecture

The system architecture of EMELIA is critical when it comes to producing an efficient and accurate classifier. We will implement features within our system based on the technologies we have extensively researched along with the technologies recommended by our clients. We will define which tools from the analysis will integrate cohesively to build EMELIA.

The diagram below in Figure 2 depicts the main components such as the processing of data, the machine learning modeling, and testing that will be implemented in our program. Our system will not have a graphical user interface and will incorporate a command-line interface for users to upload files to be assessed. Our system will be a Linux based application that will incorporate machine learning models sourced from the TensorFlow framework.

We were instructed to create a system that will be able to extract data from file that will be formatted in a CSV and process this data in preparation for the machine learning. We came to the conclusion that the best approach for this is to program our system using the Python programming language because of its simplicity and capabilities in machine learning implementations.

We will then use Python to parse through the file and save the data within our system for future assessment. After certain attributes from the file have been stored, we will forward the data into a machine learning modeling system that the Tensorflow framework provides. The TensorFlow machine learning framework will allow the development team to use our selected learning model. Due to the popularity and community support surrounding this machine learning framework, the development team is confident this framework will benefit the development of EMELIA.

For better assistance we will also be working with other open-source neural network libraries such as Keras. Keras is also written in python and works well with Tensorflow thus will be a great tool to help us engineer the project features and complex methods.

After training the model with the data provided we will then test the model using a series of sample data and compare the results with our control variable. The model will then express our results in a percentage with a minimum goal of 90% accuracy. If the goal is met then the system will report the result back to the user. However, in the event that this goal is not met, the system will reprocess the data and continue to be trained so that it can report the results with its utmost best accuracy. There will be some circumstances where the data that is passed in will be deemed foreign to the system. If the system is unable to classify the data within the ticket, the user will be notified and the ticket will be submitted for manual review.

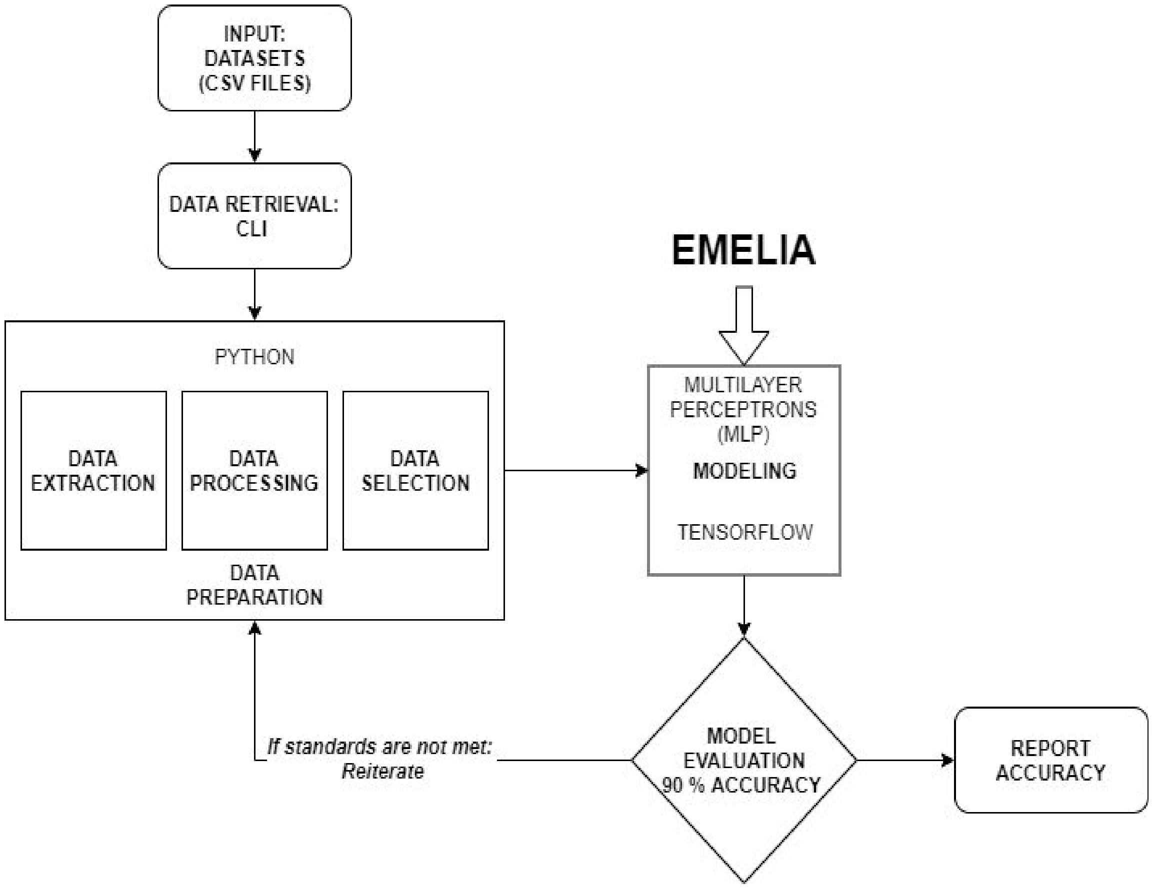


Figure 2

4. Project Requirements

This section will outline the requirements needed to provide a viable product for our client. The requirements section will be organized in the following subsections:

1. Functional Requirements
2. Non-functional Requirements
3. Performance Requirements
4. Environmental Requirements

The requirements have been obtained through weekly meetings with our client. A major challenge of obtaining the requirements obtaining test data due to confidentiality concerns. The rest of this document section will focus on the expectations for the software needed to build EMELIA and the business requirements that will serve our client moving forward.

High level requirements for EMELIA are:

- Train learning model on test ticket data
- Classify the ticket data using the training set
  - Goals is to reach 90% accuracy for classification
- Utilize command line interface for users

4.1 Functional Requirements

The functional requirements will outline features that EMELIA should have in order to properly classify ticket data. Each of the key features of the system will be outlined and followed by a description of each feature. The implementation for EMELIA may vary in comparison to its deployed version, due to the lack of source code and software used to provide input data.

1) EMELIA must train test data and be able to classify tickets
   a) The system should be able to use test data to train the classifier
   b) The training model should classify ticket data using predefined categories
2) EMELIA must be able to receive and process a file containing ticket information
   a) The system must be able to take in large volumes of data at a single time
      i) The data will be taken from a comma separated values (CSV) file
         (1) A database will be used by system engineers once EMELIA is in production
   b) The system will analyze information in the CSV file and format the data as input to be read in by EMELIA
      i) Organizing data is essential for specifying data columns used for classified output data
   c) The system will utilize a supervised learning model
      i) The system will use domain data to create a neural network model for the purpose of machine learning
      ii) Provided data is defined prior to classification, and will be used to produce a predefined output

(1) Ticket data is labeled and the total number of classification types are determined prior to the learning

iii) The system will use the TensorFlow framework as a training model

(1) A neural network has been the proposed machine learning technique for this project.

(2) TensorFlow will allow for the neural network to utilize backpropagation to adjust the input as it is classified.

3) The system will test the model's accuracy and performance

a) The system must test the model's accuracy during training

b) The system must be able to benchmark the system's performance for training of a particular data set

i) Returns the benchmark data to the user through the command-line interface

(1) Time taken to perform data classification

(2) Number of items classified

(3) Filename of the data that was classified

4) EMELIA will utilize a command-line interface for system engineers

a) Since the user's for the system are technical professionals, they will only require a command-line interface to run the system

i) The interface must inform the user of the accuracy for training done on the data set

ii) The interface must inform the user of the data set that is being classified

iii) The interface must inform the user of the time it took to classify data

iv) The interface must allow the user to make changes to the classification process by changing parameters

The following section will detail the use cases for our system according to the listed functional requirements. These use cases will provide rationale regarding the requirements and describe how they should impact the use of EMELIA.

Use Case: Accept ticket data for processing

Actor: Domain Expert

Description: The user will enter command to provide ticket data as input

Precondition: User has access to EMELIA and ticket system files

Postcondition: The user will have successfully provided input ticket data to a trained classifier model

Main Flow:

1. The user navigates to the directory containing ticket data via the command line
2. The user enters command to activates EMELIA and specifies the file containing the ticket data
3. System accepts the file and reports back to the user that input data is loaded successfully

Alternative Flow:

3. System rejects the file and asks the user to verify file name and that the file is in correct format

Use Case: The classifier trains on the data and provides accuracy report

Actor: Domain Expert

Description: The machine learning model begins learning using provided labeled ticket data

Precondition: User has access to EMELIA and ticket system files

Postcondition: The user has a large set of existing labeled data.

Main Flow:

1. The user runs the system to pull labeled ticket data
2. The system trains model with labeled ticket data
3. The system uses a test set of data to pass through the model

4. The system compares the model's output of test data to the test data's actual labels and returns accuracy reading

Alternative Flow:

3. The system produces an error due to inability to classify the provided data

Use Case: The user will run the tests to determine classification accuracy for the system

Actor: Domain Expert / System Engineer

Description: The user will enter command to provide system performance results

Precondition: User has access to EMELIA and ticket system files

Postcondition: The user will see performance results provided by the system

Main Flow:

1. The user runs the command to provide system performance data
2. The system will analyze the total number of tickets assessed and time taken to run the classifier
3. The system will provide accuracy generated by the output layer
4. System displays the performance to the user via the command line

4.2 Non-Functional Requirements

This section will cover non-functional requirements needed for EMELIA. These non-functional requirements have been taken from the functional requirements stated in the previous section and the environmental requirements, which will be detailed in the next section. Each of the non-functional requirements will be detailed according to priority.

1) Accurate
    a) The system should be able to classify data at 90% rate of accuracy
        i) This system needs to be able to classify system failures accurately using the ticket data
            (1) Accuracy and precision needed for resulting data

ii) This non-functional requirement is the most important for this system due to the risk associated with inaccurate results

2) Reliable

a) This system must be accurate and dependable

i) The system should not only produce accurate results, but should produce the same relative result on the same data set

(1) This will be part of the testing process for the system to ensure its viability

3) Retainable

a) System must maintainable and usable long term.

i) This solution should be the start of a larger classifier for our clients communications systems

ii) Retainability will be dependent on the quality of the architecture, accuracy, and reliability of this project

b) The system must be able to integrate successfully with the existing framework

i) The team will be utilizing CSV files to extract and classify data

ii) The team may need to refactor the working version of the assessor to tie into the existing software

(1) Client currently uses software which contains individual ticket data

4) Extensible

a) The system should allow for changes to the classification process by adding and removing parameters.

i) These changes parameters must allow the engineers to increase the classification accuracy for the training model

5) Scalable

a) The engineers that use the system should also be able to change the number of items that are classified within the data set

i) If an engineer would like to classify a subset of the data rather than the entire data set, the system should be flexible enough to allow:

(1) Time efficiency by reducing the total time spent on classification of ticket data

(2) Serve the business needs of our client by accommodating multiple ticket systems.

(a) Usable outside of the current ticket system domain

6) Testable

a) The system should have functionality that allows for the data classification process to be tested

i) System engineers that use the classifier should be able to run a subset of their data through the training model to attain a result

(1) Pass the same subset to the test suite to ensure that they are receiving accurate results

ii) Testing should determine accuracy and precision of data classification

4.3 Performance Requirements

The performance requirements specified for EMELIA pertain to the accuracy and speed of the system. In order for EMELIA to be an effective solution, the system should provide accurate classification of ticket data, which will reduce time and labor that is spent on the process. The system must perform the following:

1) EMELIA needs to classify ticket data at a rate of 90% accuracy

2) EMELIA need to reduce time spent on each ticket

a) A total of 20 minutes is the average time taken to classify ticket correctly

b) A total of 30 minutes is the average time taken to classify and correct a misclassified ticket

These two performance requirements are the most valuable in terms of efficiency for this project. Without the ability to meet these two performance requirements, EMELIA will not significantly impact the problem faced by our client. Based on research, EMELIA should be able to meet these two requirements and solve the process of manually classifying ticket data. The development is confident in the ability of EMELIA to reduce the time and resources needed to respond to failures and maintain our clients ticketing system.

4.4 Environmental Requirements

The client has specified that the development must develop using an anaconda package manager. This environment will keep all dependencies at the same version throughout the development cycle. Package management will keep variability for errors low throughout development and aid the team in troubleshooting. The following are the software version requirements needed for development:

1. Python 3.6
2. TensorFlow 1.13.1
3. Keras 2.2.4
4. Pyodbc 4.0.26
5. Pandas 0.24.2
6. IPython 7.4.0
7. Scikit-Learn 0.20.3

Jupyter notebook and Anaconda Prompt are two more software components that are included as part of the anaconda installation. These technologies will be used to develop a working prototype and aid the team in the later stages of development for this project.

Client source code will not be utilized to develop EMELIA. Due to the security concerns related to the access of source code, the development team will be developing a stand alone software product. EMELIA will need to integrate with the client source code. These environmental constraints will provide an added challenge to the team, but should not prevent the team from delivering a working solution for our client.

5. Potential Risks

There are some risks associated with the development of this product. The risks that are visible at this phase of development are a threat to the performance and quality of our solution. These risks range from low to high in terms of severity and likelihood. We have come up with different solutions in order to fully address these risks to ensure that the problems are mitigated.

Risk assessments are generally undertaken in three clearly-defined stages:

1) Identification of all the hazards
2) Evaluation of the risks
3) Implementation of measures to eliminate or control the risks

Risks Overview:

Risk	Severity	Likelihood
Misclassification of ticket data	High	Low
Compatibility errors when integrating to client’s system	High	Medium
System efficiency is highly dependent on limited training data	Medium	Medium
Program overload operation	Medium	High

Risk Mitigation

1. **Misclassification of ticket data**

   If tickets are classified incorrectly it is assumed that these misclassified tickets are understood as something completely different than its actual content. This leads to a potential time set back greater than what the software would be eliminating.When our program produces the error of classification ticket, it will output the wrong time to make the other party make a wrong judgment.Eliminating this possibility would require addressing the situation of particular tickets in which the prediction model has low confidence in that classification. This means that even with a very high rate of accuracy in prediction tests, the low confidence classification would be part of the complementary percent.

   One common method of fixing this issue is by issuing each classification a real-valued confidence level rating on some scale and setting a threshold value that all confidence levels must be greater than in order to be assigned that classification. If not met, these remaining tickets are placed into an unclassified category for System Engineers to review. Though this doesn't completely eliminate the task of engineers reviewing the tickets by hand, it will minimize the risk drastically saving invaluable amounts of time while maintaining accuracy. As more tickets are classified correctly, they can be placed in a growing set of previously classified tickets which can be used to train the model, improving the accuracy of the prediction model over time.

2. **Compatibility errors when integrating to client's system**

   The risk of this problem is very high. Because when the team develops software, it will not develop on the customer's system, so when the project is completed, it is likely to have incompatibilities with the customer's system.When installing the server of the customer company, their server does not meet the system requirements, resulting in the system cannot be installed or there is a problem. There are various illegal plug-ins and viruses that affect the normal use of the program on the company's client, resulting in the client's failure to work properly.

   When the team is in the research stage of project requirements, it is necessary for the client to know about the server. In the process of our development, we can supplement the functions of the server. In this way, when the system classifies tickets, it can classify tickets into the set classification required by customers through machine learning. If possible, customers can use our original version to test in their company's environment. Practice is the best way to test the results. Feedback the test results to us for improvement.The team needs to ensure the stable operation of the client and put on the latest system patch.

3. **System efficiency is highly dependent on limited training data**

   The problem of user training is not big, because the customer is a computer company, and has a very good programmer.When we finish the project, we need to hand over to each other. But because the other party has no staff to participate in the development process, our team will be responsible for training their staff to complete the project handover. It is convenient for the other party to use the software. But the employees of the other party have other things to do, they may not have enough time to attend the training, or they have questions about how to operate the software. These are the potential risks of training employees.

We can discuss the specific training time with customers. Agree on a place to participate in the training all day. According to the specific use of the process, we will provide necessary consultation and guidance to the other party's employees to ensure that the other party can independently complete various operations.

4. **Program overload operation**

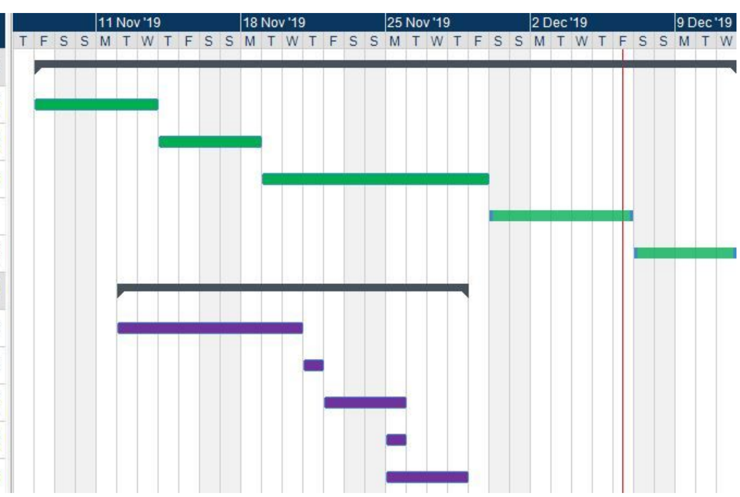
With the increase of the number of imported program tickets, different tickets will be classified and error reported every day, which causes the load of the program to increase. Because of the continuous operation of machine learning algorithm, more and more sets will be generated to classify data, and the memory occupancy rate will increase rapidly, which is a serious burden for the program system.

One of the ways to solve the problem is to use the bisection algorithm to classify the bills correctly and wrongly, to sum up 10% of bills into a special set, and then to determine whether these uncertain bills have problems through human inspection. A large part of the running pressure of the program comes from these uncertain bills. As long as they are manually classified, the load of the program can be greatly reduced. In addition, a new table can be created to store the different collections classified by the program, which will reduce the repeated calls of functions and the burden of running the program.

6. Project Plan

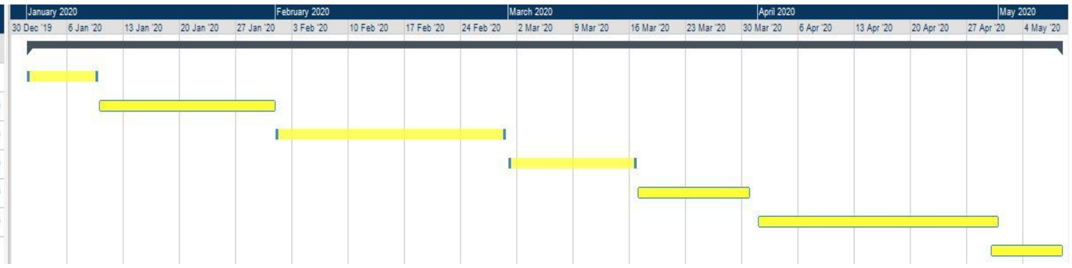
Our plan for the development of our project closely follows the following Gantt charts:
Fall 2019 Semester Plan:

	Task Name	Start	End
1	**Requirement Specification**	**11/8/2019**	**12/11/2019**
2	Non Functional Requirements	11/8/2019	11/13/2019
3	Functional Requirements	11/14/2019	11/18/2019
4	Rough Draft	11/19/2019	11/29/2019
5	Final Draft	11/30/2019	12/6/2019
6	Signed Requirements document	12/7/2019	12/11/2019
7	**Prototype**	**11/12/2019**	**11/28/2019**
8	Research Ticket System for Classification	11/12/2019	11/20/2019
9	Extract/Clean Data from Tickets	11/21/2019	11/21/2019
10	Build Neural Network	11/22/2019	11/25/2019
11	Test Initial Prototype Accuracy	11/25/2019	11/25/2019
12	Prepare for Tech Demo	11/25/2019	11/28/2019



Spring 2020 Semester Plan:

	Task Name	Start	End
18	**Development**	**1/1/2020**	**5/8/2020**
19	Planning	1/1/2020	1/9/2020
20	Prototype Version 2	1/10/2020	1/31/2020
21	Basic Features	2/1/2020	2/29/2020
22	Testing	3/1/2020	3/16/2020
23	Refactor Prototype V2	3/17/2020	3/30/2020
24	Client Review	4/1/2020	4/30/2020
25	Final Product Deploy	4/30/2020	5/8/2020



The milestones for the progression of the development of our project include:

- **Design Review:** A brief review of our project and our plan for development.
- **Requirement Specification:** Describes the requirements that our project will have, our solution vision, and potential risks. This document will serve as a basic outline of the requirements for the development of our prototype. Our client will sign this document at the end of the semester and we will adjust our plan according to the clients' expectations.
- **Prototype:** After choosing a viable machine learning framework and a programming language that can easily integrate into our current environment to accept and train input data provided by our clients, our tech demo will demonstrate the functionality of our process to accept and train test data. We will continue to build on the initial prototype in the spring semester to develop a robust command-line interface that will take input from tickets and classify them with a high degree of accuracy.
- **Development:** In the second semester, we will create a plan for the development of our product, refactor the old prototype to a newer version, add basic features to the command line interface, test the accuracy of the current prototype, refactor the current prototype, submit the prototype to be reviewed by our client, and deploy the final product. We will refactor out prototype to be able to handle high volumes of data, accept multiple input data file formats if necessary, and validate the column headers in the test data file. We will also attempt to improve the accuracy and speed of our machine learning classifier through the use of framework optimizers and using a better, more refined algorithm to correctly classify test data. Will will be conducting numerous tests to try to prove that the accuracy reported by the framework is accurate.

7. Conclusion

The problem that we are addressing is the addition of a machine learning classifier to improve the efficiency of the General Dynamics ticket system. Currently, our client's company has a special team that processes roughly 30 tickets a month that contain failure reports for various equipment across the different departments. The objective of our project is to create an interactive machine learning intelligent assessor for General Dynamics. With the help of our sponsors, Rick Duarte and Jon Lewis, we will develop a system that allows for the retrieval, analysis, and exportation of data that will assist the organization in failure reporting. This intelligent assessor will act as a solution for the current ticketing system.

There are three main requirements for EMELIA as stated by our client. The machine learning classifier must be able to do the following:

- Train learning model on test ticket data
- Classify the ticket data using the training set
    - Goals is to reach 90% accuracy for classification
- Utilize command line interface for users

The neural network machine learning model will be used to train the ticket data. The classification accuracy for this learning model needs to be at least 90% to be used in the ticket system. EMELIA will utilize the test data and tests to determine the accuracy of the model in comparison to manually classified test sets. The user facing challenge will be to use a command line interface for users. Since EMELIA users will be domain experts and system engineers, a command line interface will provide the metrics needed for evaluation. The time needed to classify tickets will need to be determined later in the development cycle. All requirements are subject to change upon agreement by the Team SmartState our client. If necessary, the development team will collaborate with our client to modify requirements. All changes will be reflected in future versions of this document.

The project risk can be divided into technical risk and service risk. Our team is mainly to solve the technical risks, which requires us to be more proficient in using panda and tensorflow and other frameworks to conduct in-depth machine learning, proficient in using methods to screen data machine learning models, and learn other classification data algorithms, so as to ensure the accuracy of classification tickets and maintain the stable operation of the system.

Our team's next phase of development will be focused on the technological demo for our Capstone mentor. This process will begin by processing the test data provided by our client to train our machine learning classifier. Then we will report the initial performance of the multilayer perceptron model and analyze some benchmarking metrics. The demonstration will be successful as long as the program accepts data and reports a percentage in the command line once the data has been processed by the learning model. Moving into the Spring semester, we will continue to make changes to our initial prototype, focusing on improving accuracy and adding in a command line interface. We will also be working with our client, who will approve the refactored version of our prototype, to ensure a timely deployment of the final product.

Overall, the choice of technologies will make our envisioned solution both technologically feasible and less complex to implement. Our technological choices will provide the functionality needed to progress EMELIA into a finished product. Based on the rationale stated throughout the subsections of this document, the development team is confident in our ability to build our Event-driven MachinE Learning, Intelligent Assessor (EMELIA).