LOAFR #1!
Design Document

Authors:
Trent Young
Wil Bishop
Ashlyn Pietrowski
Anakin Nolette

Group: 1

This page intentionally left blank.

Document Revision History

Date	Version	Description	Author
03/17/2022	0.0	Initial draft	Trent Young Wil Bishop Ashlyn Pietrowski Anakin Nolette
03/19/2022	0.1	Added skeleton to Section 5	Wil Bishop
03/19/2022	0.2	Added skeleton to Sections 1, 2	Ashlyn Pietrowski
03/19/2022	0.3	Added skeleton to Section 3	Trent Young
03/19/2022	0.4	Added Class Diagram and started Sections 4 and 6	Anakin Nolette
03/20/2022	0.5	Reviewed skeletons and planned steps to fill them.	Trent Young Wil Bishop Ashlyn Pietrowski Anakin Nolette
03/21/2022	0.6	Fleshed out Section 3	Trent Young
03/22/2022	0.7	Added Class Attributes Section 4, and 6	Anakin Nolette
03/22/2022	0.8	Added sequence diagrams	Wil Bishop
03/23/2022	0.9	Added information in Sections 1 & 2, class attributes	Ashlyn Pietrowski
03/24/2022	0.10	Reviewed and edited entire document	Trent Young Wil Bishop Ashlyn Pietrowski Anakin Nolette
03/26/2022	0.11	Added more methods and modified attributes	Anakin Nolette
03/26/2022	0.12	Detailed touch up on Section 3 and redesigned the diagrams for Section 2.3	Trent Young
03/27/2022	0.13	Added sequence diagrams	Wil Bishop
03/27/2022	0.14	Added method information, modified Sections 1 & 2	Ashlyn Pietrowski
03/27/2022	0.15	Reviewed and edited entire document	Trent Young Wil Bishop Ashlyn Pietrowski Anakin Nolette
03/28/2022	0.16	Final review	Ashlyn Pietrowski
04/11/2022	0.17	Added updates after LOAFR 0.1 built	Ashlyn Pietrowski

This page intentionally left blank.

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a blueprint for the eventual implementation of the LOAFR system. It contains details about the components of the system and how each component relates to one another. More specifically, this document contains information about the system's architecture, interfaces, and overall structure. After the introduction, this document lays out the design overview, interfaces, data stores, structural design, dynamic models, non-functional requirements, and any supplementary documentation.

1.2 System Overview

The product that this document addresses is the Log-file analysis framework (LOAFR) system. It is being created to aid in analyzing and reviewing data logs. The program shall allow the company to adopt a common data logging format and have automated tools to reduce error and time. The LOAFR system shall be able to search, sort, filter, add, save, load, and manipulate the data in the log-files. The LOAFR system shall be easy for engineers to use, with a complete UI and documentation on how to use the system. All of these components shall allow the company to save money on resources devoted to the log-file system and produce log-files more consistently.

1.3 Design Objectives

The software being developed is meant to speed up the creation, analysis and review of the data logs that are being created for testing purposes. The software shall create log-files more accurately and efficiently than manually created log-files. The software shall be able to load and edit log-files, yet, it shall not be able to delete log-files. It shall allow the users to easily and efficiently sort, search, and filter the data logs. This will allow the users to look for patterns or specific data within the log-files. Users can search, sort, and filter through using specific search, sort or filter 'bars' where the user enters in keywords that match the query. The software shall also standardize the data logging format, so when a file is entered into the system, it is put in a standard format. This will occur before the data file is allowed to be edited or searched. The software shall be able to add new data sources and/or data items to allow the user to expand the information in the data. Users will also be able to interact with the data in a standard mode and batch processing mode to test or shuffle through the data. All data types will be accepted in the environment, as long as the data conforms to the standardized format. The client will be able to use the LOAFR system on desktops and laptops within the corporation's private server system. The LOAFR system will be hosted on the client's web system, thus, workers will be interacting with it on a webpage. LOAFR shall have access to the client's server system to save and load files into their private file system.

1.4 References

Requirements Document

LOAFR UML Diagrams

1.5 Definitions, Acronyms, and Abbreviations

Definitions unique to this document are listed below. All other definitions are assumed to be in the Requirements Document.

- TBN-X: *To be named* (which is our placeholder for TBD from the previous document). The X will be a number, representing the location of the 'to be named' item in order of the document from top to bottom. Each TBN refers to a different problem to be addressed.
- TBD-X: *To be determined.* A reference to a 'TBD' from our SRS document. X represents the number that ties the TBD back to the SRS document.
- LOAFR: *Logfile Analysis Framework* - The software system that this document is about.

2 Design Overview

2.1 Introduction

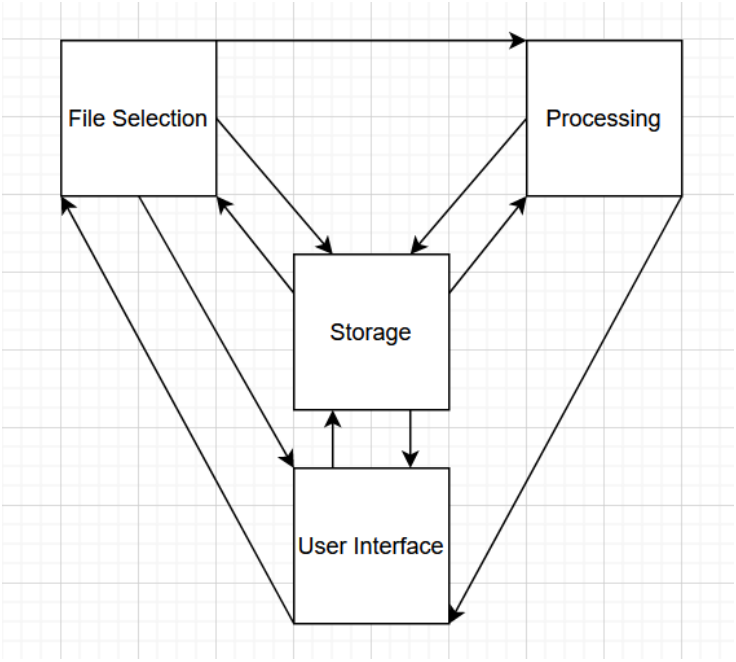
For this program, we will use an object oriented design from the top down to design the system. This will allow us to directly monitor and observe the relationships between entities. Object oriented design will enable us to have a clear hierarchy of classes through inheritance. The top-down design approach will also support this hierarchical structure, but still allow for cycles to occur in order to provide a two-way relationship between sub-classes. For the architecture, we will implement a client server system in order to allow the LOAFR software to be accessed on any employee laptop or desktop. By having clients request and receive access to the LOAFR system through the centralized server, the LOAFR software can be maintained more easily as it is localized to one host machine. The relevant tool for this program is the Pandas database system, as this database allows the developers to load and observe data from CSV files with ease. Pandas will help us read in files, export files, and implement methods/classes such as search, sort and filter.

2.2 Environment Overview

The software shall operate in-house for a safety-critical system manufacturer. The LOAFR software will reside on a centralized server that the client's server engineers are in charge of. It will be executed by opening the application through clicking on an icon button on the user's home page. The hardware shall be primarily average corporate standard workstations (desktops and laptops), however, the software must work with the testing platform used by the client. The operating system shall be Windows 8 or later. The software shall not interact with other applications on the corporate system, however the software shall store and output various data logs and text files in its allocated space.

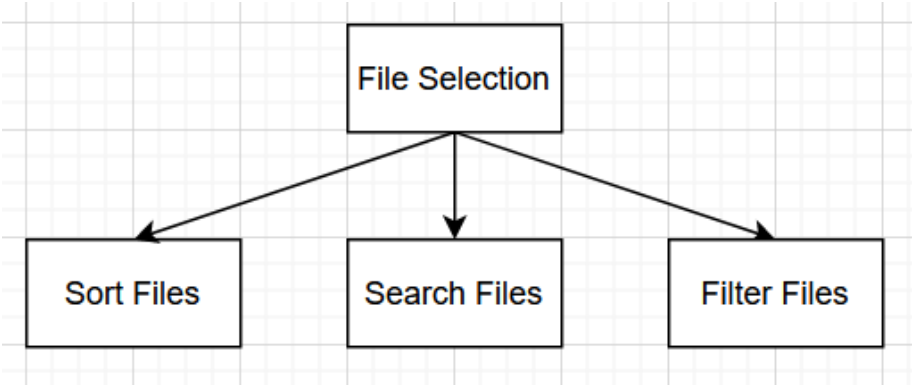
2.3 System Architecture

2.3.1 Top-level system structure of the LOAFR system:



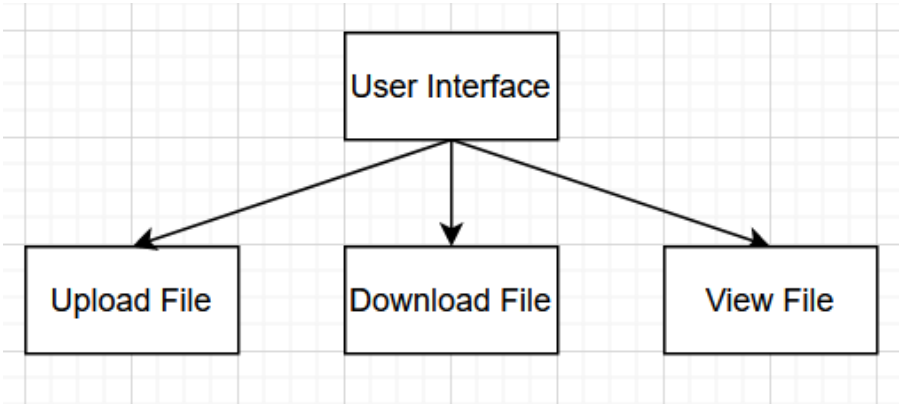
The system consists of four major components: a File Selection subsystem, a Processing subsystem, a Storage subsystem, and a User Interface subsystem. Since the Storage subsystem contains information about standardizing all of the files before the files are manipulated or utilized, it is bidirectional with all other subsystems. File selection and User Interface are bidirectional with each other as one may use their components in conjunction with each other. For example, one would view a file before sorting it. After sorting, searching, or filtering a file in File Selection, one could save, load, or implement batch processing in the Processing section. After all changes have been made, the user can download the file through the User Interface section. However, this is only one example of a path that the user could take through the LOAFR system. A user can start at any of the subsystems, make tests or changes, and end up saving the file onto the client's system. Thus, the LOAFR system is cyclical.

2.3.2 File Selection Subsystem



The File Selection subsystem consists of the options 'sort files', 'search files', and 'filter files'. We can assume that all files are standardized to the proper data format of a standard CSV file before arriving in File Selection. The user can implement a keyword in the search bar, sort bar, or filter bar. Each keyword is associated with a predefined sorting, searching, or filter algorithm / method. If the keyword is invalid or not associated with a predefined algorithm, an error will be raised (TBN 2.3.2-1). If the algorithm / method is complete, a list of items that correspond with the query is returned.

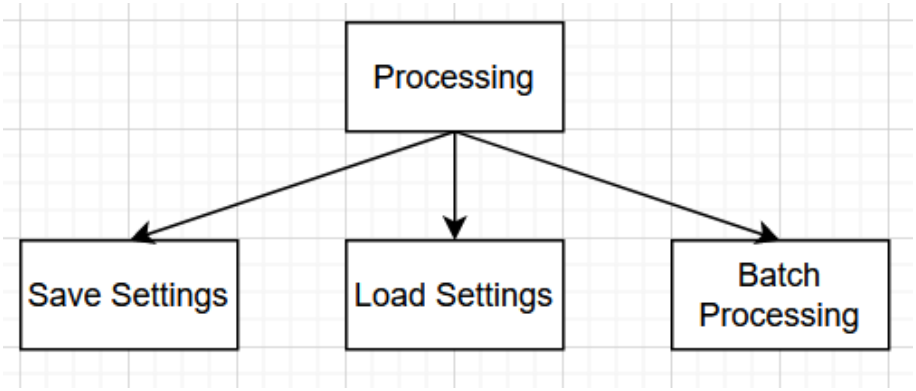
2.3.3 User Interface Subsystem



The User Interface subsystem consists of the options 'upload file', 'download file', and 'view file'. All of these options are displayed as a list of options after the 'User Interface' button is clicked. Users can upload files into the system for testing by selecting the 'Upload File' option.

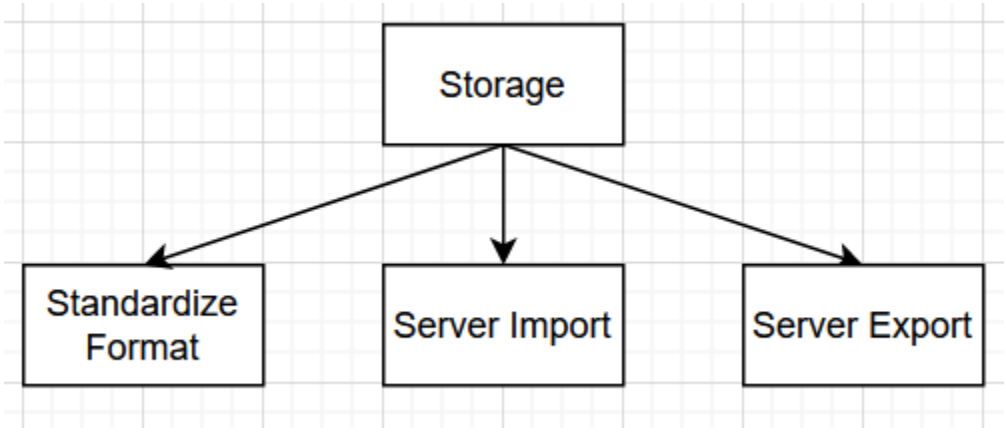
After the file is uploaded, the file will be converted to the standard format of a standard CSV file. Users can download the file from the LOAFR system to the user's home computer by clicking the 'Download File' button. Users can also open or 'view' a file that is already present within the LOAFR system by clicking the 'View File' option.

2.3.4 Processing Subsystem



The Processing Subsystem allows users to process data. Users can select the 'Select' button to select the files that they would like for batch processing, then select the 'Process' button to add specific settings and process the files. The 'Save Settings' button allows the user to save the current search/sort/filter settings. Conversely, the 'Load Settings' button allows users to load previously saved search/sort/filter settings. Only one version of old search/sort/filter settings can be accessed; thus, when the 'Load Settings' button is clicked, the singular version is automatically loaded and implemented into the current batch processing session.

2.3.5 Storage Subsystem



The Storage subsystem allows the user to communicate with the server such that they can move files in and out of the LOAFR system for business purposes. The 'Standardize Format' subsystem is automatically implemented, as it standardizes any and all files brought into the LOAFR system to the standard CSV file format. 'Server Import' allows the user to import the data log file into the main server's file system. 'Server Export' allows the user to export the data log file from the same file system.

2.4 Constraints and Assumptions

1) LOAFR will be programmed in Python, which provides several restrictions. Since Python is a high-level, object oriented language, we must treat all items as objects while programming. Python's easy readability will help with continuing the maintenance of LOAFR after the client fully acquires it. Python will allow us to implement the pandas databases system to more easily manipulate and test files. The implementation of the pandas library will encourage the reuse of LOAFR. Python's libraries are also safe to use on protected networks, like the client's. Since Python is high level, we have less control over memory allocation and file storage, compared to a language like C. However, the file storage will be handled by the users after each log file is created.
2) Since LOAFR must be usable on both desktop and laptop workstations, the software must be able to accommodate these different platforms. For example, LOAFR should be able to work on different versions of the Windows systems. The system must also be capable of adjusting the window size of the program to fit the screen of the workstation.
3) Since the software will be maintained by the client's engineers, LOAFR must be well documented and organized. Comments must be used throughout the codebase, and directions to maintain the software should be included.
4) LOAFR is developed for one company, thus, it must be compliant with that company's standards as well as their industry standards. This is to ensure that the client can use the developed software appropriately and professionally.
5) Since LOAFR has the ability to send and receive files from the company's home server, it must be secure and compatible with the server. It shall not, however, interfere with other applications or restricted documents on the company's home server.
6) LOAFR must comply with the IEEE Software Engineering Standards as it is software developed in a professional environment.
7) Since LOAFR is made from scratch, there are no components within the software that are being re-used. This provides added complexity for the programmers and will increase the amount of time taken to develop the program.

3 Interfaces and Data Stores

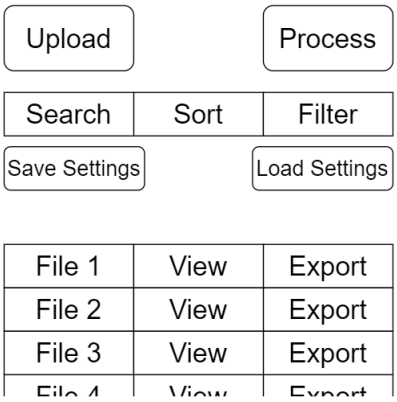
3.1 System Interfaces

3.1.1 Database interface - passing standardized files to and from the database

This interface is used to send and receive files between the LOAFR system and an external database. This interface is responsible for making the interactions between the system and database safe for the system.

3.1.2 Web interface - user to upload files and download converted files, displays application

This interface is used to interface with a user through a web browser. This interface has five interactive buttons, three type bars, two save and load past type bar queries, and one scrollable file list. The buttons the interface has are: an upload button - for uploading new files to the system, a Process file - runs batch processing on the selected files, Export file - exports a copy of the selected file to the machine the user is using, and View - allows the user to open the selected file and see the data stored in the file. The three type bars all interface the same way, taking in keyboard input and adjusting the files displayed in the scrollable file list. The scrollable file list displays a list of file names that have been successfully converted and stored in a standard CSV file format, alongside each file name an individual view and export button will be displayed.



3.2 Data Stores

3.2.1 The system stores a list of all standardized data and relevant flags to allow the sort search and filter bars to operate.
3.2.2 The system will store past inputs to the sort search and filter bars in a CSV file format
3.2.3 The system will store a ‘help’ file with descriptions on how to use the LOAFR system.

4 Structural Design

4.1 Class Diagram

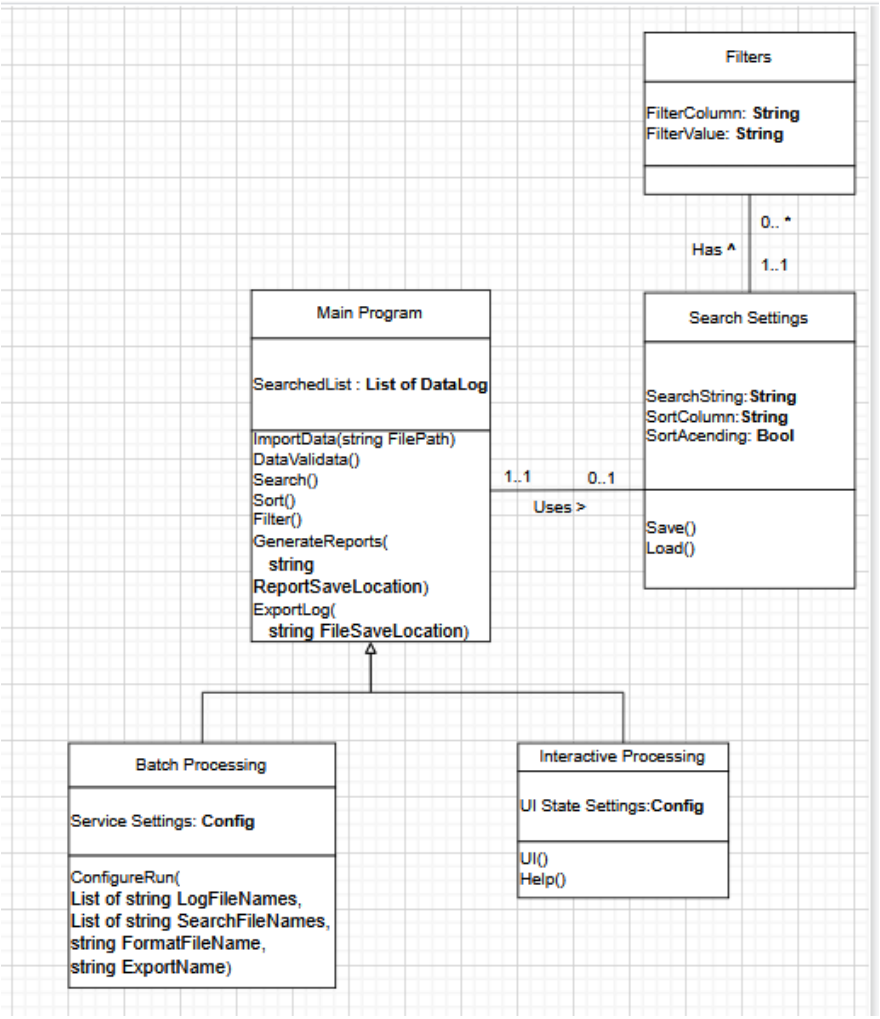


Fig 4.1

Description: There are three classes: Main Program, Filters, and Search Settings. The Main Program is the main part, with the other classes being eventually related to it.

In addition, the Main Program can have a Search Settings Class that stores the attributes. It can have many Filters, which hold the many filters. Finally, the Main Program has two subclasses: Batch Processing and Interactive Processing as one has the UI and the other is more suited to a command line.

4.2 Class descriptions / Classes for LOAFR

4.2.1 Class: Main Program (Loafr) and subclasses

- Purpose: *To model the relevant aspects of the physical tank that stores fuel*
- Constraints: *None*
- Persistent: *No (created at system initialization from other available data)*

4.2.1.1 Attribute Descriptions

- Attribute: *DataLog*
  Type: *List of Data Logs*
  Description: *Holds the formatted Data*
  Constraints: *Uses pandas.read_csv()*

- Attribute: *SearchSettings*
  Type: *Search Settings Class*
  Description: *Holds the Search Settings to filter, search and sort the Logs.*
  Constraints: *None*

- Attribute: *SearchedList*
  Type: *List of DataLog*
  Description: *Holds the list of Filtered DataLog.*
  Constraints: *None*

- Attribute: *Configuration*
  Type: *Object*
  Description: *Holds the configuration for the class type.*
  Constraints: *Not Null*

4.2.1.2 Method Descriptions

- Method: *DataValidate()*
  Return Type: *Bool*
  Parameters: *Data File*
  Return value: *Success or Failure*
  Pre-condition: *UI called*
  Post-condition: *Validates Data*
  Attributes read/used: *DataLog, SearchedList*
  Methods called: *None*

  Processing logic: *Checks if data is valid (fits desired format, has appropriate values) and returns a bool indicating true if the data is valid, or false if it is not. If the file is not found, return false.*

- Method: *Search()*
  Return Type: *SearchedList*
  Parameters: *None*
  Return value: *A list of filtered DataLog*
  Pre-condition: *SearchSetting not null*
  Post-condition: *Returns values that match filter*
  Attributes read/used: *pandas.search()*
  Methods called: *None*

  Processing logic: *Searches through a list to find a specified filter value and returns a list of all of the items that match the filter.*

- Method: *Sort()*
  Return Type: *SearchedList*
  Parameters: *None*
  Return value:*A list of sorted DataLog*
  Pre-condition: *SearchSetting not null*
  Post-condition: *Returns values in correct order of Sorting*
  Attributes read/used: *pandas.sort()*
  Methods called: *None*

  Processing logic: *Sorts SearchedListList based on SortColumn and SortAscending settings.*

- Method: *Filter()*
  Return Type: *SearchedList*
  Parameters: *None*
  Return value: *A list of filtered DataLog*
  Pre-condition: *SearchSetting not null*
  Post-condition: *Returns values that match filters*
  Attributes read/used: *pandas.filter()*
  Methods called: *None*

  Processing logic: *Filters Searched List based on SearchSettings.Filters with a noise allowance of (TBD 423-2)%*

- Method: *ExportLog(string FileSaveLocation)*
  Return Type: *Bool*
  Parameters: *FileSaveLocation - The location where the file will be saved.*
  Return value: *Success or Failure*
  Pre-condition: *Set of Logs being looked at,*
  Post-condition: *A file with the values of the searched List.*
  Attributes read/used: *SearchedList*
  Methods called: *None*

  Processing logic: *The system takes a File Location string and then saves the SearchedList in a log file. If failed, send an error message depending on the type of failure, lack of permissions, or not enough space.*

- Method: *SubsetAnalysis()*
  Return Type: *List of strings*
  Parameters: *None*
  Return value: *List of FileNames of selected files.*
  Pre-condition: *None*
  Post-condition: *None*
  Attributes read/used: *DataLog, SearchSettings*
  Methods called: *None*

  Processing logic: *The user selects the file name using the UI, and calls this function.*

- Method: *ConfigureRun(List of string LogFileNames, List of string SearchFileNames, string FormatFileName, string ExportName)*
  Return Type: *bool*
  Parameters: *LogFileNames - List of files to read data from. SearchFileNames - List of files to lead search settings from. FormatFileName - File to determine the format for the data logs. ExportName - Name of the export file.*

Return value: *Success or Failure*
Pre-condition: *None*
Post-condition: *A report is generated.*
Attributes read/used: *DataLog, SearchSettings, SearchedList, Configuration*
Methods called: *SearchSettings.Load(), GenerateReport()*

Processing logic: *This is a batch processing method. It takes a list of files and then uses them in the methods SearchSetting.Load(). It then generates the report using GenerateReport() and ExportName.*
If failed, send an error message depending on the type of failure, lack of permissions, missing file, wrong format or not enough space.

- Method: *UI()*
  Return Type: *None*
  Parameters: *None*
  Return value: *None*
  Pre-condition: *None*
  Post-condition: *End of Program*
  Attributes read/used: *DataLog, SearchSettings, SearchedList, Configuration*
  Methods called: *ExportLog(), Search(), Sort(), Filter(), DataValidate(), GenerateReport(), Help(), SearchSettings.Save(), SearchSettings.Load(), SubsetAnalysis()*

  Processing logic: *This method will run the UI. It will have direct access to the class attributes and other methods using buttons and other UI inputs.*

- Method: *GenerateReport(string ReportSaveLocation)*
  Return Type: *bool*
  Parameters: *ReportSaveLocation - The location where the report will be generated.*
  Return value: *Success or Failure*
  Pre-condition: *SearchedList is not null or SearchSettings and DataLog*
  Post-condition: *A report is generated in the location specified.*
  Attributes read/used: *DataLog, SearchedList, SearchSettings*
  Methods called: *Search(), Sort(), Filter()*

  Processing logic: *The method will take a file string location and create a report based on the attributes. If no SearchedList is created, then the method will create one using the Search(), Sort(), and Filter() methods. If failed, send an error message depending on the type of failure, lack of permissions, or not enough space.*

- `Method: *groupBy ()*
  Return Type: *List*
  Parameters: *List of strings to search, file to search from*
  Return value: *List of filtered values*
  Pre-condition: *File*
  Post-condition: *A file with the values of the searched column*
  Attributes read/used: *SearchedList*
  Methods called: *None*

  Processing logic: *Group labels by terms.*

- Method: *between()*
  Return Type: *List*
  Parameters: *List*
  Return value: *List*

Pre-condition: *None*
Post-condition:
Attributes read/used: *None*
Methods called: *None*

Processing logic: *finds values between things*

- Method: *Help()*
  Return Type: *None*
  Parameters: *None*
  Return value: *None*
  Pre-condition: *None*
  Post-condition: *Help Screen appears*
  Attributes read/used: *None*
  Methods called: *None*

  Processing logic: *The user activates the Help() method. It will show the documentation on how to use all parts of the system.*

4.2.2 Class: Search Setting

- Purpose: *To model the Search Settings that the user will need.*
- Constraints: *None*
- Persistent: *Yes (Search Settings can be stored in a database or file)*

4.2.2.1 Attribute Descriptions

- Attribute: *SearchString*
  Type: *String*
  Description: *The string that is being matched or searched for.*
  Constraints: *None*

- Attribute: *SortColumn*
  Type: *String*
  Description: *The column that the Search is being used on.*
  Constraints: *None*

- Attribute: *SortAscending*
  Type: *Bool*
  Description: *Determine how to order the search column.*
  Constraints: *None*

- Attribute: *FilterList*
  Type: *List of Filters Class*
  Description: *The list of filters used for searching.*
  Constraints: *None*

4.2.2.2 Method Descriptions

- Method: *Save()*
  Return Type: *None*
  Parameters: *None*
  Return value: *None*
  Pre-condition: *Not Null Attributes*
  Post-condition: *Not Null Attributes*
  Attributes read/used: *SearchString, SortColumn, FilterList*
  Methods called: *None*

Processing logic:*Takes all attributes and saves them to a file, allowing them to be loaded later in a different instance. If failed, send an error message depending on the type of failure, lack of permissions, or not enough space.*

- Method: *Load(string filePath)*
  Return Type: *None*
  Parameters: *filePath - Location of file to load settings from*
  Return value: *None*
  Pre-condition: *None*
  Post-condition: *All Attributes filled by file*
  Attributes read/used: *SearchString, SortColumn, FilterList*
  Methods called: *None*

  Processing logic: *Takes the file path and attempts to read it. If successful, the file will read values that will fill the corresponding attributes of the class. If failed, send an error message depending on the type of failure, lack of permissions, missing file, and wrong format.*

4.2.2.3 Attribute Descriptions

- Attribute: *FilterColumn*
  Type: *String*
  Description: *The column name that is being filtered.*
  Constraints: *None*

- Attribute: *FilterValue*
  Type: *String*
  Description: *The value that is being used as a filter.*
  Constraints: *None*

4.2.3 Class: Filters

- Purpose: *To hold the Filter Values and Columns.*
- Constraints: *None*
- Persistent: *No*

4.2.3.1 Attribute Descriptions

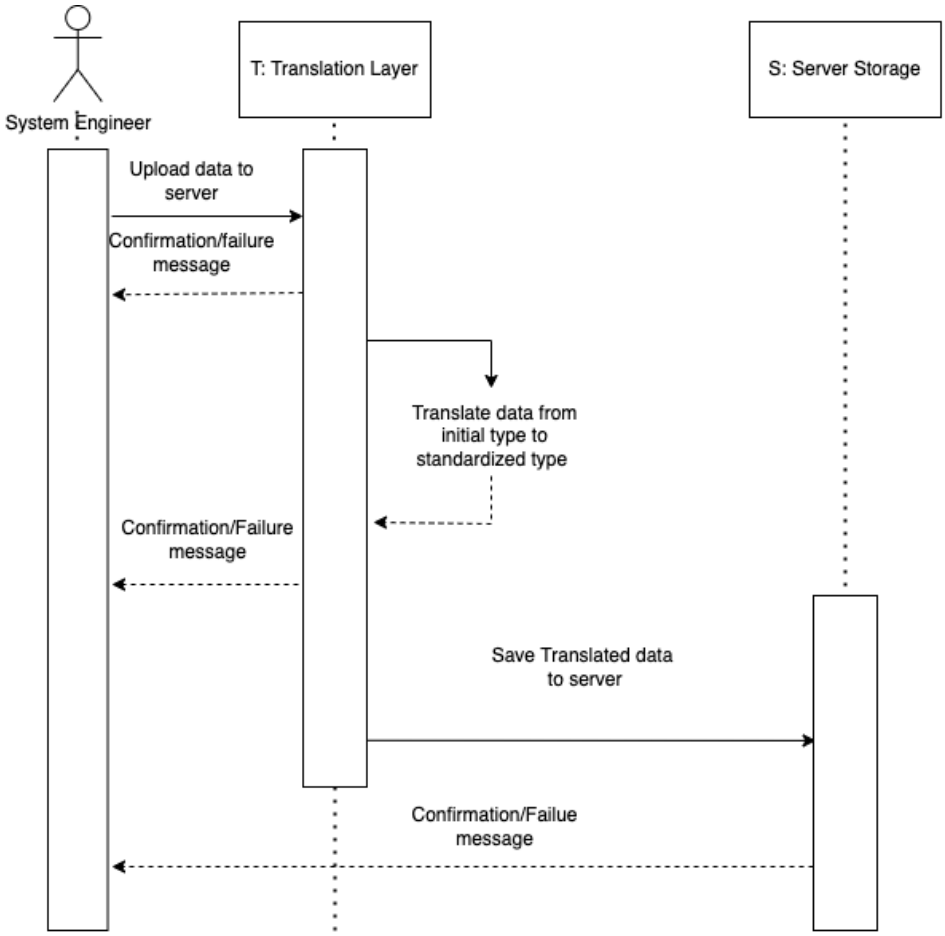
- Attribute: *FilterColumn*
  Type: *String*
  Description: *The column name that is being filtered.*
  Constraints: *None*

- Attribute: *FilterValue*
  Type: *String*
  Description: *The value that is being used as a filter.*
  Constraints: *None*

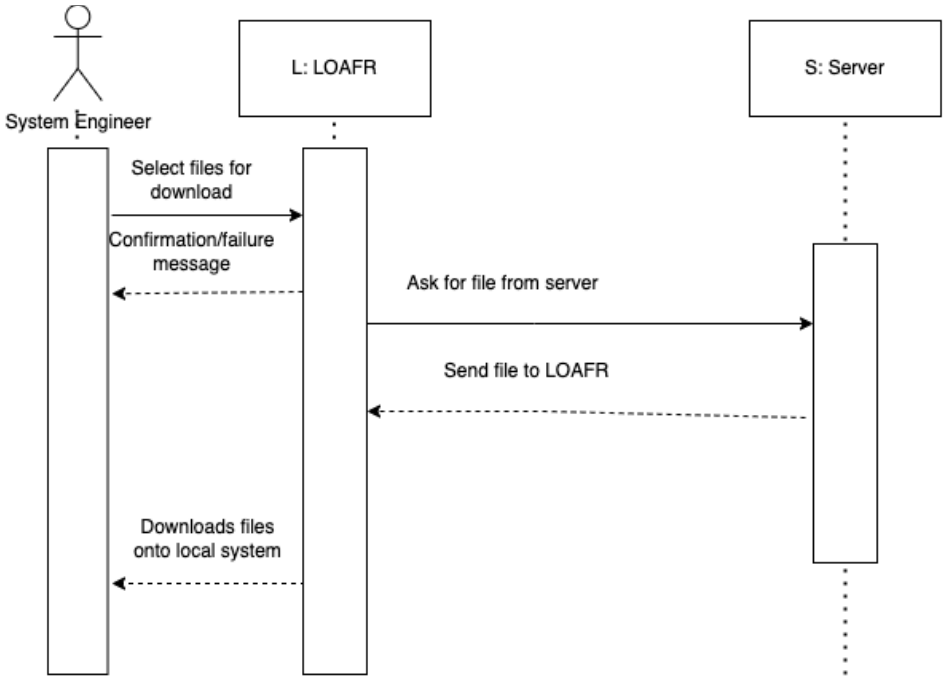
5 Dynamic Model

5.1 Scenarios

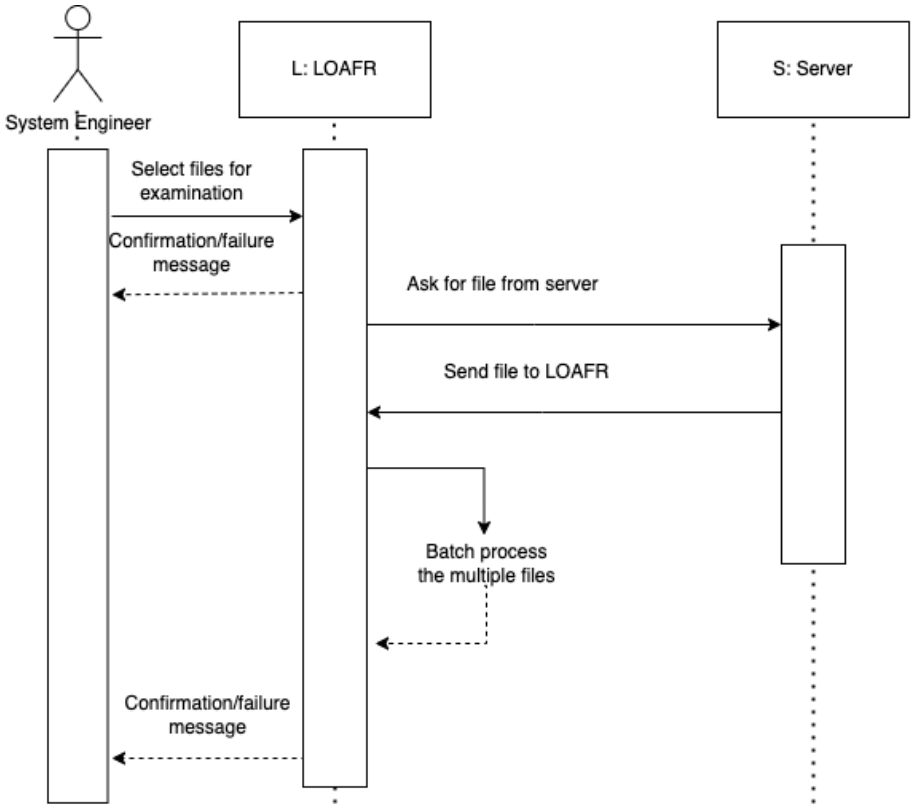
- *Scenario Name: Data Upload*
- *Scenario Description: This scenario has to do with the uploading of data into the LOAFR file system. It uploads data in one standard CSV file format, translates it into another CSV file format, and stores it on the server.*
- *Sequence Diagram:*



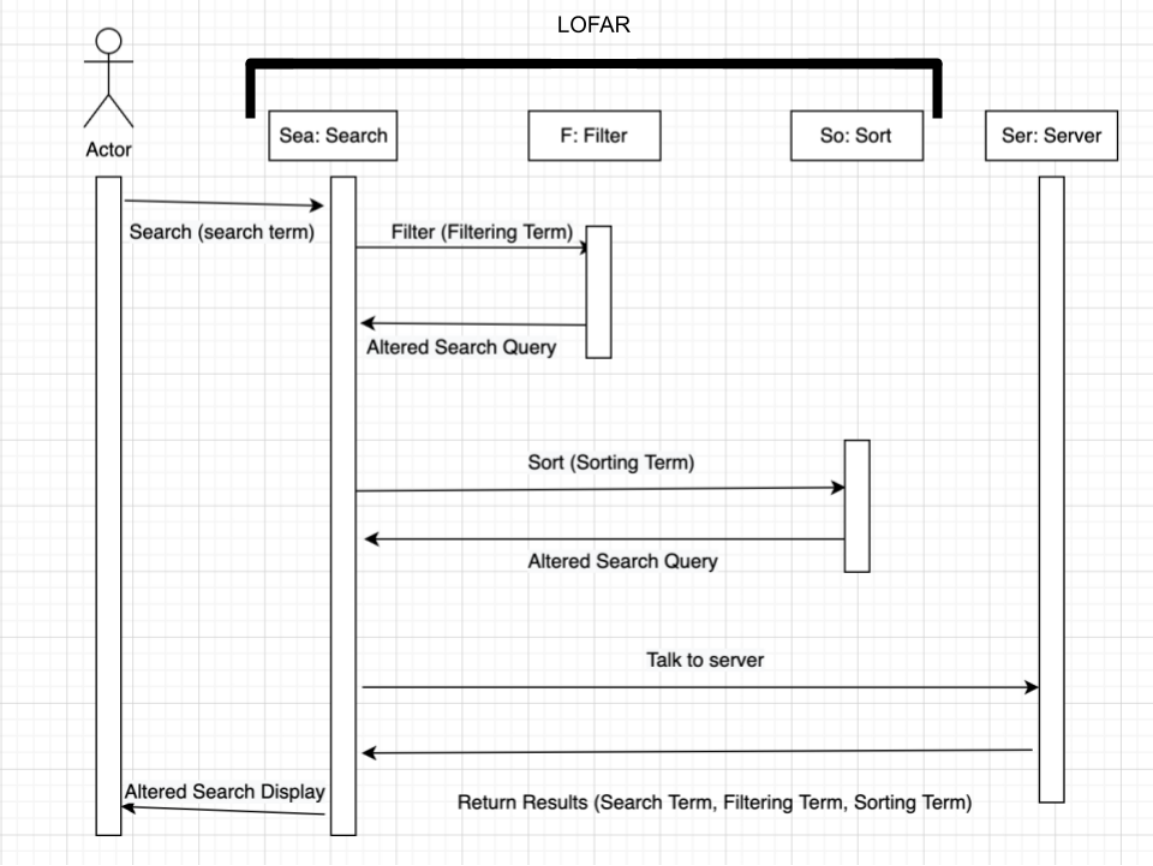
- *Scenario Name: Export Standardized Data File*
- *Scenario Description: This scenario has to do with the exporting of standardized data files of type object. When created, standardized data files are stored only on the server. However, the user can choose to download them to their personal computer.*
- *Sequence Diagram:*



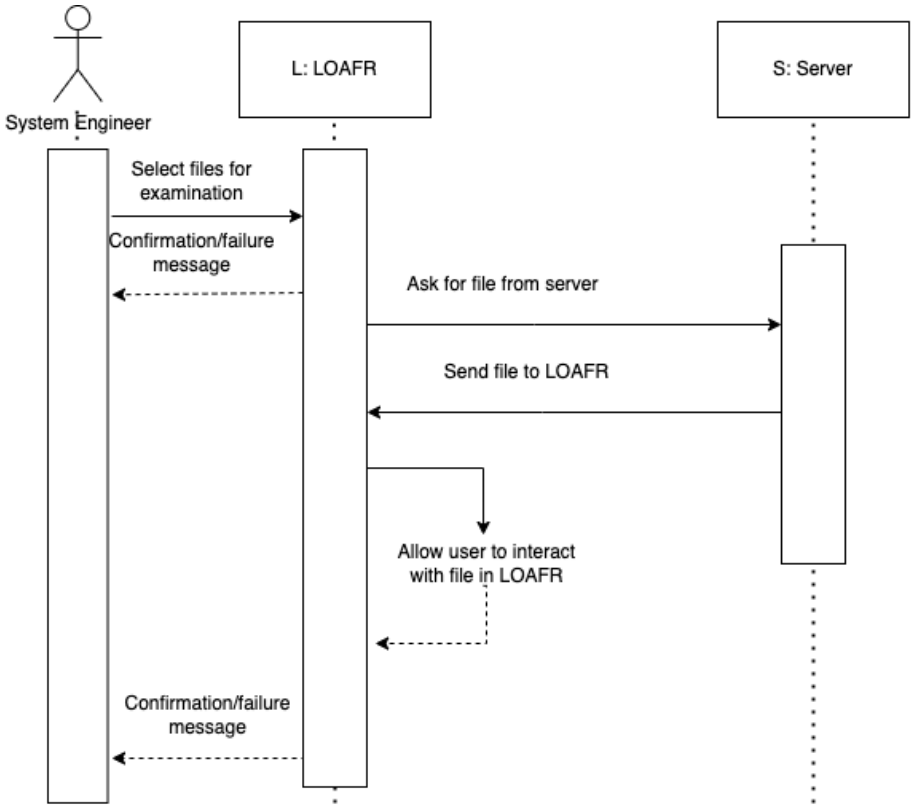
- *Scenario Name: Batch Processing*
- *Scenario Description: This scenario has to do with the treatment of large amounts of data at once instead of one at a time. It allows for the selection of multiple pieces of data of type object, located on the server at once. These files are not downloaded to the local machine, but loaded from server long-term storage into LOAFR online runtime storage.*
- *Sequence Diagram:*



- *Scenario Name: Find Subset of Files*
- *Scenario Description: This scenario has to do with finding files based on search queries, sorting mechanisms, and filtering tags. LOAFR isn't listed as an object, but Search, Filter, and Sort are all processes within LOAFR, and only the Server object is outside of LOAFR.*
- *Sequence Diagram:*



- *Scenario Name: Manual Examination*
- *Scenario Description: This scenario has to do with allowing a person to inspect files within the program. We allow people to visually look at their newly-formatted files in the proprietary format within the program. This is very similar to how it would appear if you opened the file in a text editor.*
- *Sequence Diagram:*



6 Non-functional requirements

1. The methods *Search(),Sort(),Filter(),UI(), SubsetAnalysis(), and GenerateReports()* process inputs under (TBN 51-2) in order to keep the performance requirements.
2. The methods *Save()*, *ExportLog(), ConfigureRun(), and GenerateReport()* will only edit and write files of type object.
3. The methods will use standard file saving libraries in order to fulfill requirements 5.2,and 5.3
4. The *UI()* method only allows user input every 20ms.
5. The data verification subsystem makes the system robust.
6. The use of a Data Log File, allows us to have an adaptable, flexible, and extendable system. It can handle log files from any source when it has a corresponding format.

7 Requirements Traceability Matrix

Section 1:

Design	Requirements
1.1	1.1, 2.1
1.2	1.1, 2.1
1.3	1.4, 2.1, 2.2, 2.3, 2.4, 2.5, 2.7
1.5	1.2

Section 2:

Design	Requirements
2.1	2.4, 3.2, 3.3, 3.4
2.2	2.4, 2.5
2.3.1	4.1 - 4.9
2.3.2	4.2, 4.3, 4.4
2.3.3	4.5, 4.6
2.3.4	4.7, 4.8, 4.9
2.3.5	4.1, 3 (all)
2.4	2.4, 2.5, 2.7, 3 (all)

Section 3:

Design	Requirements

3.1.1	3.2, 3.3, 3.4
3.1.2	3.1
3.2.1	4.2, 4.3, 4.4
3.2.2	4.7, 4.8
3.2.3	4.10

Section 4:

Design	Requirements
4.1	Appendix B
4.2.1	4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.9, 4.10
4.2.2	4.2, 4.3, 4.4, 4.7, 4.8
4.2.3	4.4, 4.7, 4.8
4.2.4	4.1, 4.5, 4.6
4.2.5	4.1, 4.5, 4.6,
4.2.6	4.1, 4.5

Section 5:

Design	Requirements* all of section 5 reference appendix B and C in addition to below
Data Upload (5.1)	3.2, 4.1, 4.5
Export Standardized Data File (5.1)	3.1, 3.2, 4.2, 4.3, 4.4
Batch Processing (5.1)	3.1, 3.2, 4.2, 4.3, 4.4, 4.9
Find Subset of Files (5.1)	3.1, 3.2, 4.2, 4.3, 4.4, 4.7, 4.8
Manual Examination (5.1)	3.1, 3.2, 4.2, 4.3, 4.4, 4.6

Section 6:

Design	Requirements
6.1	5.1
6.2	5.2
6.3	5.2.2, 5.2.3
6.4	5.2.5
6.5	5.4
6.6	5.4