Software Requirements Specification

for

Logistics Management System

Version 1.0 – Not Approved

Prepared by Henry Thiel

&

Braden Cariaga

Florida Poly Capstone

12/03/2021

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Braden Cariaga Henry Thiel	12/02/21	Initial Document Draft	1.0

1. Introduction

1.1 Purpose

This document is used to describe the fundamental features and major functionality of the Logistics Management System version 1.0 to be delivered to the U.S. Army Program Executive Office for Simulation, Training, and Instrumentation *(PEOSTRI)*. This software version will only cover the initial requirements to satisfy the objective of providing a comprehensive tool to enable users to practice compliance of specific logistical policies while tracking the overall progress and status of each requirement and sub-requirement. This software will be a standalone software with no immediate integrations to any existing software, although the software will be built with a potential for future integration in mind.

1.2 Document Conventions

Terms which are defined in the glossary will be indicated by a bold font. Acronyms which are defined in the glossary will be indicated by an italicized font. Colored fonts are used on the requirements to indicate the order of priority. From highest to lowest priority the colors are as follows: red, orange, green. All the functional requirements will be italicized. References to requirements will be in parenthesis and italics, with numbers referring to their section (*1.2*). Figures will be named by the section number followed by an alphabet letter in the alphabetic order in which they appear. Figures will also be indicated by an italicized font.

1.3 Intended Audience and Reading Suggestions

This document is intended for PEO STRI management and future maintainers of the system. The document will include an **Appendix** with definitions for relevant keywords and explanations of acronyms. This document is intended as a high-level overview of the system and will not feature code samples or specific implementation details or plans, and instead will discuss project features and their constraints.

1.4 Product Scope

PEO STRI has requested a system to help in enabling logisticians and project groups to track their individual and overall progress of ongoing projects while advancing their current process workflow for technological implementation. The software being developed is a project management system to be used in capturing logistical data and displaying such data to the respective users. The scope of the application is to create an accessible, easy to use application for the PEO STRI staff to use in tracking projects and their respective processes and regulations. This proposed system is designed to be scalable and generalized; therefore, the system will not initially tackle the specific workflow processes that have been described to us. The specific workflows will be configurable on a project-by-project basis with an optional templating system for duplication once the system has been initialized. These templates can implement the specific workflows provided.

1.5 References

ReactAdmin Demos – https://marmelab.com/react-admin/Demos.html
ReactAdmin E-commerce Demo – https://marmelab.com/react-admin-demo/

2. Overall Description

2.1 Product Perspective

This product will be a standalone software application with no integration to other systems. The system will utilize a **client-server model** to provide two separate web services available. The client service will be the main entry point for users to interface with the system. The server service will be responsible for the processing of information and interfacing with the database. The client service will be accessible by any web browser, but content will be restricted by authentication. The server service will have a protected application programming interface (*API*), meaning the service will be open for all connections, but content will only be accessible by those with the appropriate credentials. The server service will be built with scalability and security in mind; therefore, the system will have the option for a **Cross-Origin Resource Sharing** (*CORS*) implementation in later versions of the software. For the system, a user will utilize a web browser to access the user interface. Through the user interface, the client will submit Hyper-Text Transfer Protocol (*HTTP*) requests to the server service's protected API. The server service will then process the request and manipulate the database, as necessary.

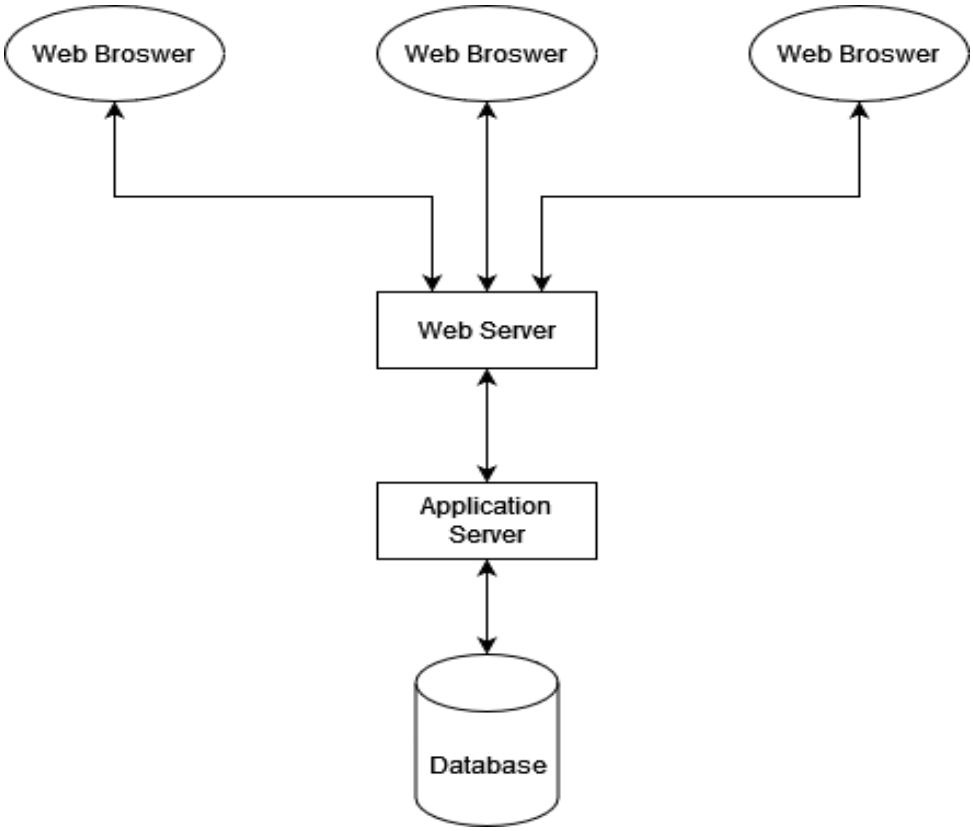

Figure 2.1.a – Diagram of relationships between all major components.

2.2 Product Functions

This software must perform the following functions:

- Project Administration – A user must be able to create, read, update, and delete projects. The projects should be able to be created from scratch using a form field, or through auto population of selecting a template.
- Routing and Dependencies of Projects – All projects will have tasks related to their completion. Those tasks should be able to be created with other tasks as either being blocked or blocking this task; thus, creating a project flow.
- Template Administration – A user must be able to create, read, update, and delete templates for projects that follow a process that is often duplicated.
- Project Proceeding – A user must be able to progress through the tasks of a project by completing the requirements of each task. They can do so through the following actions:
    - Files – A user must be able to progress a task through uploading a file to the task, or by downloading a file for review.
    - Submit Approval to Project – A user must be able to submit approval of a document/task. The approval should mark the task as complete and progress the project if it does not need more approvals. As an optional feature/extension of this functionality, we are exploring integrated document viewing and signing (see section *C.1*).
    - Attach Comments to Project – A user must be able to attach comments to a project or task of a project and review other comments.
- Notifications – The system should feature notifications for each user of the system for their awaiting tasks/actions needed. The system will deliver notifications through the user interface and through emails.
- Logistical Views – The system should feature several logistical views for population of the logistical data being collected. The system should display it in an intuitive, easy to understand manner.
- User Authentication – The system should feature its own authentication service.
- User Permissions – The system should feature its own user permissions system enabling a system administrator to create a set of roles with pre-defined settings for allowance of action on the system.

2.3 User Classes and Characteristics

This system will feature several different types of users with all different perspectives on how the system will be used. The user permission system (see section *4.10*) will be the main decider as to what access and functionality each user has based on their respective roles. In general, the users of the system can be classified into four main categories:

- Project Members – The project members are the lowest users of the system having the least amount of functionality. The main action of these users is proceeding/accomplishing the tasks of a project. The proceeding that may be done is document upload, document review, and comment attachments. This user will receive notifications on any task in which they are assigned.
- Project Approvers – The project approvers are the second level of user in the system. This user is responsible for reviewing and making decisions on a task (see section *4.5*).
- Logisticians – The logisticians are the main overseers of the projects. This user is responsible for creating projects and templates, as well as viewing logistical statistics about all ongoing projects.
- System Administrators – The system administrators are the highest level of user rank. For most of the systems use, these users do not play much of a role. These users have the

permissions to alter all data records of the system. In addition, these users are responsible for defining the specific user roles and permissions in the user permission system (see section *4.10*).

2.4 Operating Environment

This system will be flexibly designed to operate on any average computer server. The software will utilize the NodeJS runtime environment, so it will be agnostic for the operating systems supporting NodeJS (Windows, MacOS, Linux). The operating environment must have NodeJS installed and be running for our software to work. There are no specific hardware requirements for the system's environment.

2.5 Design and Implementation Constraints

The system needs to be web-based with a client-server-database structure (see section *5.2.1*) and needs to run on the Army's existing cloud system. This applies a few design constraints (see section *5.1.1, 5.1.2*) due to running the system on Army hardware. The system must be approved by the Army Approval Process for Applications and feature open-source **libraries** which limits what libraries our system can implement.

The system also needs to share their existing cloud platform, so the system should be designed to operate in that kind of environment. Our current plan is to utilize platform-agnostic solutions that will run on any hardware; however, we are keeping the army's hardware in mind for solutions that only operate on specific hardware environments.

2.6 User Documentation

This system will have an extensive user documentation to support all potential users of the system. There will be two documentation files created for each interface:

- Graphical User Interface (GUI)
- Application Programming Interface (API)

For the GUI, a user manual will be developed. The user manual will detail all major functions of the system, independent of user rank. The functions which will be specifically detailed in a step-by-step manner are user login/signup, project proceeding, project administration, template administration, notification handling, logistical population and customization, and permissions administration. The user documentation will be provided in the form of a Portable Document Format (*PDF*) document, as well as a specific documentation page located on the graphical user interface. This documentation is made to be used by the users of the system.

For the API, a detailed PDF document will be developed, cataloging all the HTTP requests supported by our system. This document should provide a description of the request, permission level of the request, parameters, response codes, and an example request. This documentation is made to be used by developers of the system, or those who are interfacing with our system.

This document, the Software Requirements Specification and the Software Design Specification will act as the developer documentation which details this system.

2.7 Assumptions and Dependencies

Our main assumption is that this system will be expanded on in the future. This system will eventually integrate with other systems PEO STRI is developing, which means our system must be well-documented and open to extension and modification.
We are planning to use the open-source **libraries** KoaJS, Arango DB, ReactAdmin, and NodeJS to develop the system. These are our software dependencies for the system and will provide web utilities that will make the system possible.

3. External Interface Requirements

3.1 User Interfaces

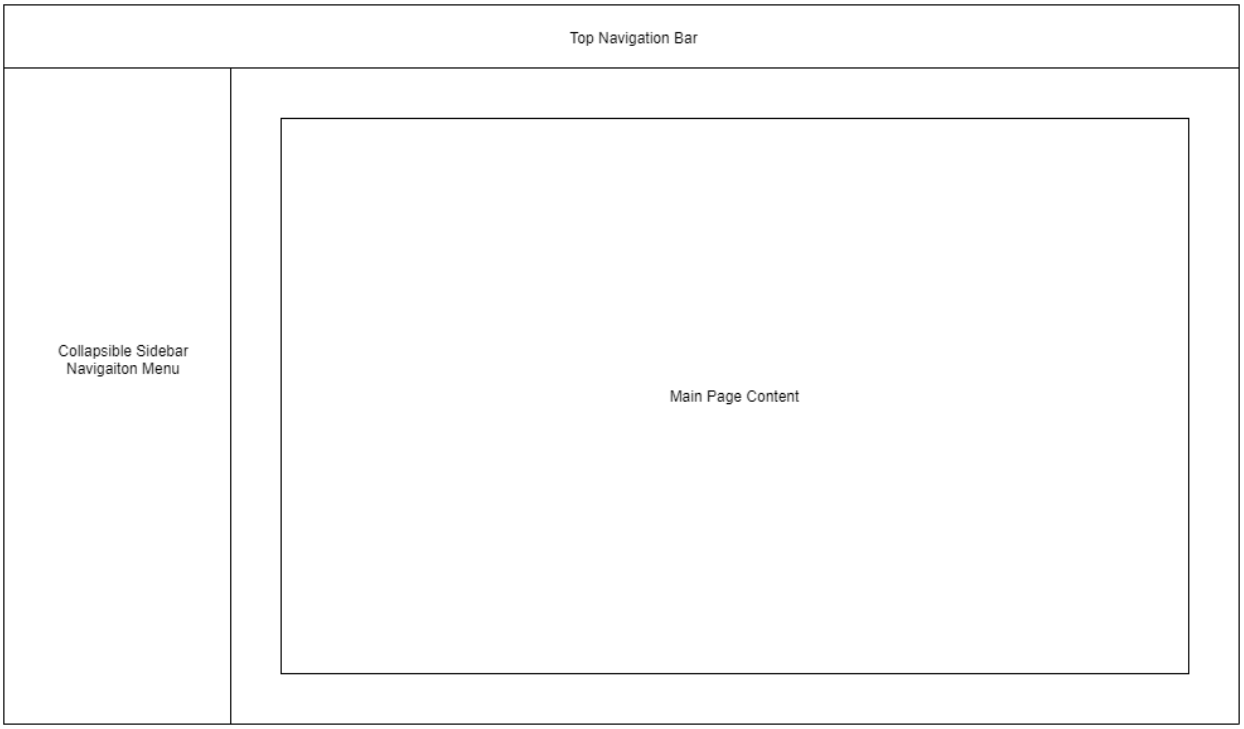

Figure 3.1.a – Wireframe design of the user interface.

The designed wireframe displayed in *Figure 3.1.a* shows the rough outline of our user interface. The interface will feature two main navigation bars making it a two-dimensional interface. The top navigation bar will be used to show the current page status and provide other helpful information/actions relating to the specific page. The left side navigation menu will be the main navigation used by the user. This navigation will feature all the pages that the user has access based on its role permissions. The main page content will be the location of the information for each page that the user is interacting with. The main page content will likely be in the form of a list, information boxes in a tiled format, or a form for user inputs. For the user interfaces, we will be utilizing ReactAdmin (see section *3.3)*, a client-side JavaScript library built from React JS and Material UI. The library is predefined with several components, allowing for modification and repurpose. In addition to the wireframe, ReactAdmin has developed several demos showcasing the

ability of the software. Based on those demos, we plan to focus our design closely resembling the e-commerce demo layout and design (see references).

3.2 Hardware Interfaces

The only hardware interface for this system is the cloud server in which the system is hosted. There are no external hardware interfaces in which the system interacts with.

3.3 Software Interfaces

This system will utilize the following **software libraries** and their related dependencies:

- NodeJS (v16.13.1)
- KoaJS (v2.13.4)
- ReactAdmin (v3.19.2)
- Arango DB (v3.8.3)
- ArangoJS (v7.6.1)

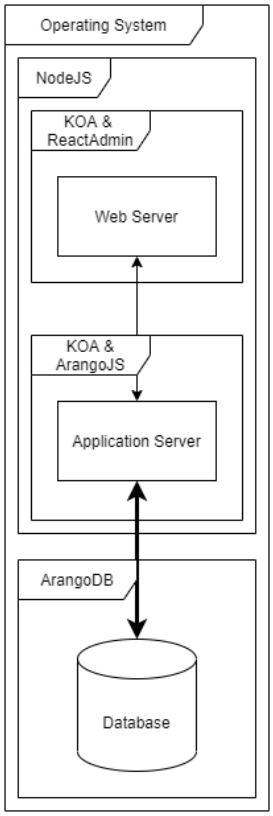

Figure 3.3.a – Diagram depicting the software libraries as layers upon one another and the relationship to our system architecture.

All the **software libraries** will be built on any operating system with a runtime environment of NodeJS. Running on our environment are two separate server instances which communicate with one another. The web server is a combination of ReactAdmin and KoaJS. ReactAdmin is a client **library** used to build the user interface of the system. Alone ReactAdmin is only the bare bones of the interface; therefore, it is paired with KoaJS to route the interface to the client browsers. On the

application server, KoaJS is used to provide HTTP routes to handle the requests sent from the user interface. The application server also utilizes ArangoJS to enable the communication between the application server and the database on request processing. Back on the operating system an instance of Arango DB is employed to provide a database server for the information to be stored.

3.4 Communications Interfaces

The system will mainly utilize HTTP requests to transfer data between the different components. The client browsers will use HTTP to request the web interfaces and communicate actions to the application server. The application server will directly connect to the database through HTTP requests, as well. In addition to the main components of our architecture, our software will also utilize an email server and SMTP protocol to transfer the email notifications to the users (see section *C.3*). Another interface used is the client web browser. Our system will be accessible on any JavaScript enabled web browser (including mobile devices). In addition, the system will have the possibility of accessibility with the application server through strict HTTP request, without the need of a browser or user interface. Although that functionality will be dependent on the specific configuration once initialized.

4. System Features

4.1 Project Administration

4.1.1 Description and Priority

This feature consists of the three main operations that users can complete on projects: creation of new projects, modification of existing projects, and removal of projects. This is of high (9/10) priority. Projects cannot be used if they are not able to be created. There are a few sub-features of this system that require other features to be completed before this system can implement them. Until then, this feature will lack those sub-features.

4.1.2 Stimulus/Response Sequences

Users will start by opening a project's administration panel. The server will present them with the operations they have permissions for (see section *4.10*). After selecting the operation, they want to perform, the system will provide a confirmation message before performing the operation.

To create a new project, the user starts by selecting a template (see section *4.2*) or creating from an empty template. General project data is collected from the user (project name, expected completion date, etc.) and stored into the task manager. Sub-tasks are added next, by adding task data, applying user-groups (see section *4.10*), and creating the dependency tree. New projects can be saved as templates for reuse.

Modification of a project presents the user with a form containing all the project’s fields and their current values. These fields can be modified, and requests are sent back to the server to update the project. All modifications are confirmed by the user before submission.

4.1.3 Functional Requirements

4.1.3.1 Projects should have management tools

Projects are the main deliverable of the system and they need to feature tools for managing them. This feature provides the operations required to do so.

4.2 Project Proceeding – Routing and Dependencies

4.2.1 Description and Priority

The tasks in projects can have dependencies, which need to be notified of progress when their subtask is completed. This feature updates what tasks in the project are currently awaiting interaction, and the system will do this automatically based on the project's dependencies. This is a medium priority (6/10) feature. Dependencies and routing are very good productivity features and greatly improve efficiency.

4.2.2 Stimulus/Response Sequences

After a user approves and completes a task, the next tasks in the dependency tree will be opened and users can interact with it. This will also send a notification using *4.7 Notifications* when a task is ready to be routed to the next task in the dependency tree.

4.2.3 Functional Requirements

4.2.3.1 Project tasks should have dependencies

Projects are not composed of a single task, and usually have dependencies. These dependencies should be managed by the system and completed tasks should automatically notify users.

4.3 Template Administration

4.3.1 Description and Priority

This feature is very similar to *4.1 Project Administration*, except that it performs these operations on templates rather than operating projects. Templates are used to speed up creation of commonly used projects but must be created and modified before their use. The same creation, modification, and deletion operations are available. This is a low (3/10) priority feature since templates speed up workflow but are not required for other major requirements.

4.3.2 Stimulus/Response Sequences

User sequences are very similar to *4.1 Project Administration* however the user is informed they are operating on templates, rather than projects. Additionally, template administration is accessed from a shared location, rather than the project's administration form.

4.3.3 Functional Requirements

4.3.3.1 Templates should exist

Templates can greatly speed up a project's workflow by eliminating the amount of boilerplate required to produce a functional project. This enables logisticians to focus on actual work rather than spending time on management tools.

4.3.3.2 Templates should have management tools

Creation of templates can be done during project administration; however, modification and deletion of a template will require its own interface.

4.4 Project Proceeding – Files

4.4.1 Description and Priority

The most fundamental operation of the system is uploading documents. These documents are filled out locally by the user and are uploaded to the system. Files are associated with project tasks, users, and metadata gathered during upload. Files can be viewed and downloaded for signing. This feature is of high (10/10) priority; while some sub-features are of lower priority, the system cannot operate at a base level without file uploading.

4.4.2 Stimulus/Response Sequences

To upload a file, users will start by selecting the subtask or project the file is associated with. The website will present the user with an upload box, which will accept a local file from the user. The system will then pull relevant metadata about the file and store both the metadata and the file on the server. The user will also be able to modify the file's metadata during upload.
Files associated with a project are viewable on the project's dashboard and can be downloaded and viewed from the website. The metadata will also be viewable, and some fields will be editable.

4.4.3 Functional Requirements

4.4.3.1 The system should track data and forms

Uploaded files and comments are the granular data the system will store and track. These files need to be managed and easily accessible by those who need it.

4.4.3.2 The system should allow files to be downloaded locally

This allows for manual movement of files and use of the files beyond the scope of the system. Viewing a file will allow the user to download their own copy.

4.5 Project Proceeding – Approval

4.5.1 Description and Priority

After a document is uploaded and associated with a project or task, it needs to be approved. Documents may need to be approved my multiple parties, with occasional dependencies between approvals. When a document needs to be approved, an approver will be sent a notification and sent access to the task. Initially, the user will need to download a local copy, sign their approval, and then re-upload the signed document. A lower priority feature will be to allow users to sign

documents directly from the system. This is a high (9/10) priority feature. Approving tasks is the second most important requirement of the system, after managing files.

4.5.2 Stimulus/Response Sequences

Files are uploaded using *4.3 Project Proceeding – Files* and associated with their specific task upon upload. Associated approvers will be automatically sent a notification when a file requires their approval and will be given a link to their task. The links will allow the user digitally to approve the file, initially by downloading and eventually in-app. After approval, the file will be automatically routed to the next approver, if required, or back to the initial logistician.

4.5.3 Functional Requirements

4.5.3.1 The system should allow for projects to be reviewed/approved

This system will allow this requirement to be fulfilled. Documents will be routed automatically without the need for an employee to micromanage the document's path.

4.5.3.2 Documents are approvable in parallel

If a document has multiple approvers, they will all be sent a notification at once, to parallelize the process and eliminate micromanagement. If an approver decides to deny the document, it will be kicked back for review by the logistician, with multiple denied approvers being able to list their grievances at the same time.

4.5.3.3 All steps must be reversable

If a document is denied for any reason, due to a misfile, an error in the document, or a formatting issue, there should exist an option to revert to a previous state. A denied document will cause a reverse in system state.

4.6 Project Proceeding – Comments

4.6.1 Description and Priority

Uploaded documents sometimes require additional context and documentation that cannot be put directly into the document. Comments can be posted on projects or documents so employees can communicate and share feedback on the project. This is a medium (7/10) priority feature. Comments are vital for explaining why a task was rejected by an approver and can greatly improve the project's efficiency by allowing logisticians to communicate about the project.

4.6.2 Stimulus/Response Sequences

A comments panel will exist on each document, task, and project window. A user can post and view comments using these panels. After selecting a system view, comments can be posted in text form. The system will store these comments in a database. There will also be management tools including deletion and editing.

4.6.3 Functional Requirements

4.6.3.1 Approvers should be able to provide feedback

Comments on projects and tasks will be the feedback the system requires. Approvers can inform logisticians why they denied a project request and provide information on what needs to be resolved for the project to continue.

4.6.3.2 Logisticians should be able to communicate

Comments can allow logisticians working on the same project to coordinate their efforts. Doing this through an external system works, however comments in the system allow for richer comments.

4.7 Notifications

4.7.1 Description and Priority

Various operations will send notifications out to related users. This includes when a user needs to approve a document, when a document is approved, when a project is completed, when comments are sent, and other functions. This is a medium (6/10) feature and can greatly improve productivity. However, it is not a core functionality of the program. This will also include email notifications, which will be sent out to a user's listed email address.

4.7.2 Stimulus/Response Sequences

Certain operations will cause notifications to be sent out. These can be viewed on the user's dashboard, and can be filtered based on project, sender, date, and other options. Users can mark notifications as "read" which will remove them from the notification list.

4.7.3 Functional Requirements

4.7.3.1 The system should provide alerts

Sending alerts for delayed tasks is extremely important for ensuring that projects operate smoothly. By sending notifications to users, this will allow users to receive information about project statuses quickly, without having to manually check the status of each project individually.

4.8 Logistical Views

4.8.1 Description and Priority

Logistical Views includes viewing the status of all projects, what task they are on, and filtering project searches based on a query. This allows logisticians to quickly view the status of projects and determine what needs to be done for a project and if the project is off-track. This is a high priority feature (8/10) and is a major improvement over the existing email system. The system will also track various metrics about the project, as required.

4.8.2 Stimulus/Response Sequences

The main menu will allow users to open a logistical view, which allows them to view project status. Users can also enter a query to filter results. Queries can include project names, users working on a project, status, and other subjects.

4.8.3 Functional Requirements

4.8.3.1 The system should track data ...

Data includes Technical Publications, Logistical Product Data, Drawings, Spares Packages, Material Identified (using IUID), NET materials, National Stock Number, and Additional Information. Tracked data is vital to ensuring logisticians have the information they need to manage projects.

4.8.3.2 Logisticians must be able to compile all information into a single location

All information about a process can be displayed to the user by a logistical view and will automatically be compiled by the system into one location. A single compiled location removes the need for logisticians to hunt down project status and simplifies the entire management workflow.

4.8.3.3 Project status should be visible

Steps awaiting processing can be filtered by the user and information about future tasks are visible. Logistical views can inform users of project status.

4.9 User Authentication

4.9.1 Description and Priority

Users will need to authenticate their session to properly conduct business. Otherwise, projects could be approved by incorrect users, information could get leaked, or the entire system could be deleted. To resolve this, an authentication system should be implemented. This will use our own login database but should be built to potentially interface with existing login systems. This is a low (4/10) priority feature. While ensuring users can verify their identity is important, a simple authentication feature can be implemented quickly and will suffice.

4.9.2 Stimulus/Response Sequences

When a user first interacts with the system, it will prompt them for a username (or email) and a password. This will be sent to the server, verified, and an authenticated user state will be created.

4.9.3 Functional Requirements

4.9.3.1 The system should feature its own login system

A login-system is important for ensuring integrity of the system. Authenticating users will provide this login system.

4.10 User Permissions

4.10.1 Description and Priority

Not all users should be able to perform all tasks the system requires. These are noted using user permissions, which allow for different users to perform different tasks, such as only giving project creation access to high-level users. This is a low (2/10) priority feature since the system will feature a generic authentication system to prevent outside actors from interacting with the system and management tools can be given to all users to start.

4.10.2 Stimulus/Response Sequences

Permissions can be granted in the interface, but only users with that permission can do so. After selecting a user, an administrator can add or remove permissions. There will also be user groups, which allow preset user permissions.

4.10.3 Functional Requirements

4.10.3.1 The system should have an administrative role

A user group can be created with administrative permissions and granted to system administrators.

5. Other Nonfunctional Requirements

5.1 Security and Safety Requirements

5.1.1 Must pass Army Approval Process for Applications

This process is required to be passed for any application that runs on Army hardware or interfaces with Army systems. Our software must pass this, which means being careful about the data we store and what we do with it.

5.1.2 Should utilize only open-source libraries

Using only open-source **software libraries** gives the software more freedom and control over itself and is much cheaper to develop with. This is a requirement imposed by PEO STRI directly, but only slightly limits our development options.

5.1.3 Should not allow data leaks or vulnerabilities

The information tracked will be protected and should not be accessible without having to authorize their web session. Security for the system should be kept in mind during construction of the system.

5.1.4 The software should mitigate risk

A formatting issue or incorrectly signed document can cause major issues for the project and can lead to massive issues down the line. By ensuring logisticians always know project status and through implementation of safeguards, these risks can be mitigated.

5.1.5 Specifics of a project should be “controlled” information

While the general information of the project is unclassified, any information about project proceedings and their status should be handled as “controlled” information and held to higher security standards.

5.2 Software Quality Attributes

5.2.1 The system should be web-based and accessible in a web browser

A web-based system ensures that the system is usable on as many devices as possible and makes accessing the system on user devices extremely easy. This does apply constraints on the system, mostly for security and usability.

5.2.2 The system should be scalable to support step/task changes

Army regulations can change quickly, and hard-coding current regulations can make the system obsolete. Our system should be modular and flexible to allow for steps and tasks to be added or edited. This ensures the system will remain useful for longer.

5.2.3 The user interface should be easy-to-use

A user-interface that works poorly can be extremely detrimental for end users. To resolve this, our interface will be designed using UX standards and well-supported **software libraries**. We will also test our interface to ensure that it is usable and intuitive.

6. Other Requirements

6.1 Task Requirements

6.1.1 Tasks have several regulatory steps

These steps include “mark as completed”, “document upload”, “comments”, and/or “request approval.” Different tasks will have a different number of these regulatory steps. To manage these, the project will allow for project tasks to have these different steps.

6.1.2 Some steps of a task can be completed in unison

The system will send out notifications for multiple steps or tasks whenever possible so multiple users can operate on the project at once. A failed step or task will notify other users of the issue and potentially revert the project back to a previous state.

6.1.3 Steps in a task can have multiple levels

A task step could potentially require multiple sub-steps, such as by uploading multiple documents that need approval and comments. To manage this, the dependency system should be functional enough to allow for this kind of hierarchy.

Appendix A: Glossary

Definitions

Term	Definition
Client-Server Model	Distributed application structure: “clients” run by users connect to a “server” run by administrators

	over the internet allowing a client to access the system from almost anywhere on the internet
Cross-Origin Resource Sharing	Allows websites to load resources (images, text, layouts, etc.) from external sources
Software Library	A collection of computer software designed to be implemented into a bigger system, building blocks for systems

Project Hierarchy Definitions

Term	Definition
Project	Highest level; a deliverable from PEO STRI
Task	Medium level; projects have multiple tasks; tasks include Materiel Release, Materiel Fielding, Product Support Planning
Step	Lowest level; tasks have multiple steps; steps include approval, document upload, comments, requests

Acronyms

Acronym	Meaning
API	Application Programming Interface
CORS	**Cross-Origin Resource Sharing**
GUI	Graphical User Interface
PDF	Portable Document Format
PEOSTRI	U.S. Army Office of Program Executive Office for Simulation, Training, and Instrumentation

Appendix B: Analysis Models

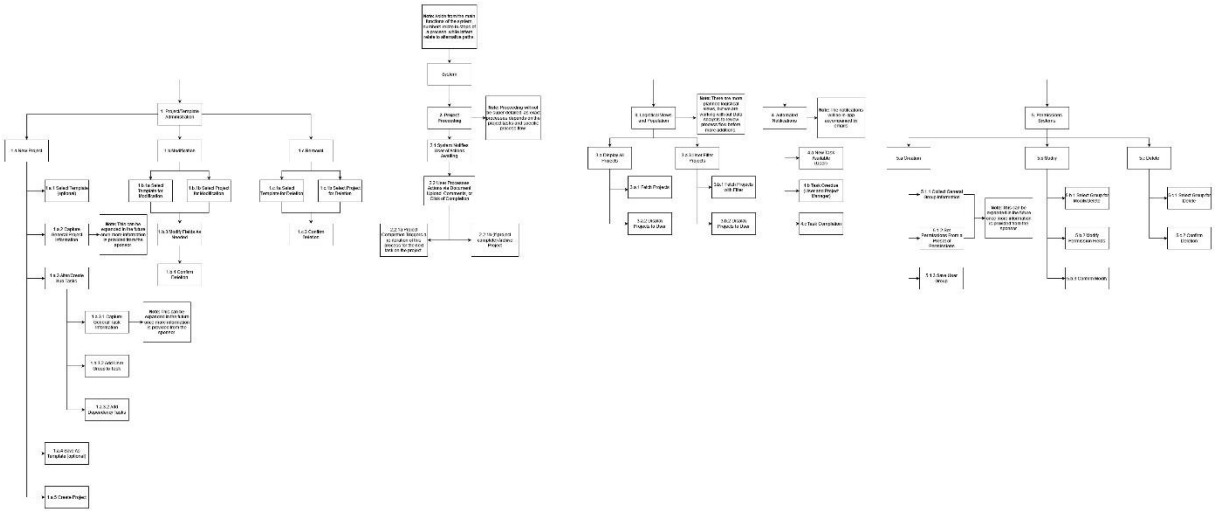

Figure B.a - Functional Decomposition Diagram depicting a high-level overview of the system's main functions and their processes. https://imgur.com/v3CCATc

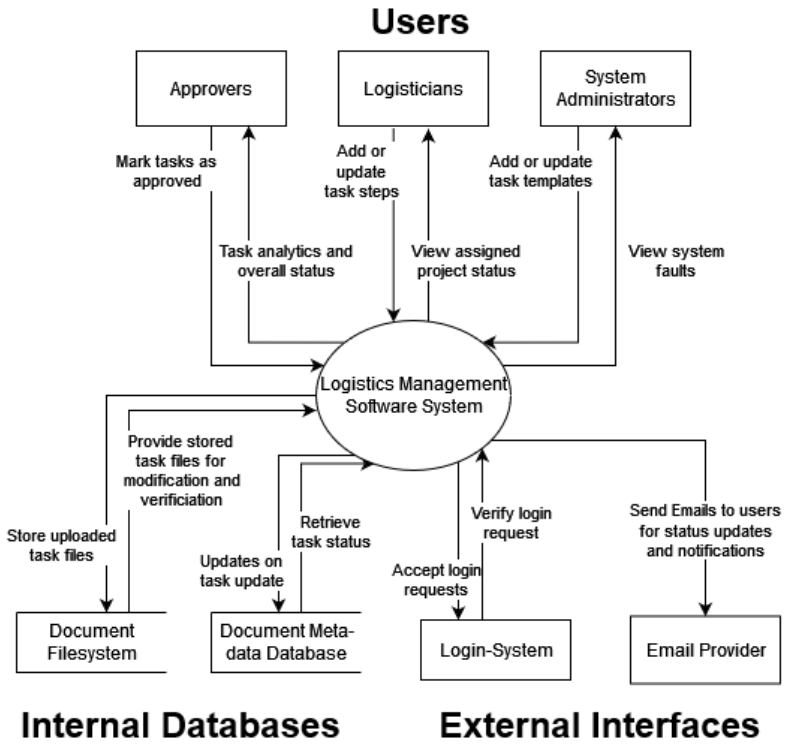

Figure B.b - Context Diagram showing external interactions between the system and its interfaces.

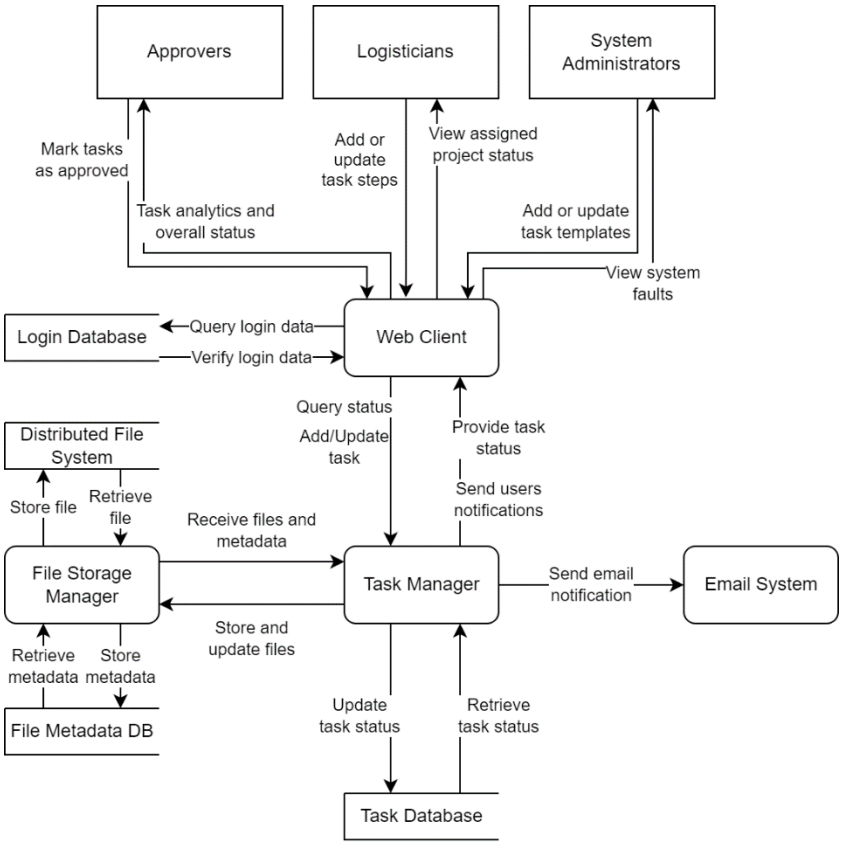

Figure B.c - Level 0 Data Flow Diagram showing how different subsystems interact with each other and external systems.

Appendix C: To Be Determined List

1. Integrated Document Viewing and Signing – This is a nice to have feature but is not a guaranteed feature. Currently, we are unable to determine if this has the potential to be developed.
2. Interfacing with Existing Army Login System – Another convenience feature, however this requires extensive review and regulations.
3. Email Service – Sending notifications over email would increase productivity, however we are unsure of the specific requirements of implementing an email service that will operate within the Army's regulations and existing system.