The Design and Development of PEO STRI Logistics Management System
Capstone 2 - Final Project Report
Florida Polytechnic University
Spring 2022

submitted by:
U.S. Army PEO STRI Team
Braden Cariaga – Computer Science
Henry Thiel – Computer Science
Ben Benyehuda – Business Analytics
Timothy Glisson – Business Analytics

submitted to:
Dr. Jing Hou
Dr. James Mennie
Dr. Sutanu Bhattacharya
2 May 2022

Table of Contents

1. Introduction

The Program Executive Officer for Simulation, Training, and Instrumentation (PEO STRI) is a leader in delivering unmatched testing, training, and information operations capabilities to enhance operational readiness in support of our national defense. The purpose of the project that we have received from PEO STRI is to create a software package that can manage and monitor their project timelines. The main issue that this software will resolve is the current lack of organization in how projects are managed. A few secondary issues that we hope to solve with this software are the timeliness of project completion and the monitoring of improvements from project to project.

Over the course of these two semesters, we are tasked with creating an enterprise resource planning software for PEO STRI. PEO STRI is a division of the U.S. Army that is tasked with managing simulations, training, and instrumentation. The software that we will be creating will manage the projects that are currently in PEO STRI's portfolio and create an easy-to-use dashboard that will enable the project managers and director of logistics to have easier knowledge of how the projects are doing and if they have followed all the procedures that must be followed.

2. Customer Needs, Objectives, and Team Interpretation

2.1 Customer Needs

Currently, PEO STRI does not have a comprehensive tool to demonstrate compliance with logistics policies. The acquisition logistics requirements span three main phases across the army, and all have specific regulatory steps that must be completed before any type of equipment is sent out to the Army for training. Those three main regulatory processes include integrated product support, material release, and material fielding. Under each of these main process phases, there are multiple sub requirements that need to be tracked to ensure successful and sustainable material fielding to the soldiers. Currently, there is no mechanism for the project managers and higher-level leadership to easily attain information on where the project is currently standing within the regulatory process. Currently those processes are just followed up via email and excel spread sheets. The main issue of not having a unified platform to track the regulatory steps is that there is no way for the project managers to know the status of a regulatory phase without asking around the office. Another issue that can arise due to the current state of the progress tracking system is that it is extremely easy to miss a sub-step, because of how many steps need to be completed. In the case that a step was missed due to human error, the Army will reject the project, until all steps are completed, even if the equipment was already sent to the Army facility. PEO STRI separates their main processes into three categories which are described below.

Process for Fielding Approval

For PEO STRI to deliver the project deliverables, in the forms of equipment and materials, to the Army for training, each project must go through strict regulatory steps before fielding. These steps help to make sure that the material given is safe, maintainable, and safely stored under the conditions necessary for its care. The process of fielding approval includes three phases: integrated product support, material release, and material fielding.

Integrated Product Support

The integrated product support level of regulatory approval includes any of the steps related to ensuring that the project will have support throughout its lifetime and that there is a plan of action across the project lifetime for maintenance, storage, and transportation. This step also includes ensuring that there is a plan for what to do at the project's end of life stage. Planning for the lifetime of the project includes creating technical manuals, training for new equipment, determining what facilities are needed for the project, insurance of funding, and allocation of any equipment needed to support the project. Other considerations that are made at this step include how the physical portions of the project need to be packaged and shipped to their destination as well as what personnel need to be allocated to the maintenance of the project. Considerations that need to be made for handling the product's end of life stage would have to do with the proper disposal of the material including following any rules for its classification level and any for safe disposal of materials in terms of the environment.

Material Release

The material release phase for fielding approval includes mostly safety regulations, suitability (the proper training needed before using the equipment), and supportability. This phase is shorter compared to the other phases of the process, but this phase is the most crucial to be completed properly. The regulations that apply to this phase help make sure that the material is safe for soldiers to use when operated within its stated parameters, which are set by PEO STRI. In addition, this phase makes sure that the material is suitable and fully tested to be used under the training conditions. This phase also ensures that the material can be supported logistically within the environment that it will be used in.

Material Fielding

The material fielding process is the main component that will determine the critical path of each project. One of the army regulations that must be followed during this section is that a memorandum of notification must be sent to whoever will be impacted by the new material a minimum of 170 days before the material begins arriving. There also needs to be an agreement between the installation receiving the material and PEO STRI on file for material to be sent. In addition, most projects will need to have a plan for the fielding. This plan is most used in projects that have multiple stages of fielding whether it is being sent to many locations or needs to be spaced out for when the installation receives it. A National Stock Number (NSN) is required for any new material before it can be fielded as well as a Material Component List (MCL). The NSN allows for location and condition tracking of the material which can be used alongside the MCL to ensure that everything that is needed for successful operation is together. Material fielding is the most logistically complex part due to requiring interaction from external departments.

2.2 Team Interpretation

Currently, PEO STRI does not have a comprehensive tool to demonstrate compliance with logistics policies. The acquisition logistics requirements span three main phases across the army, and all have specific regulatory steps that must be completed before any type of equipment is sent out to the Army for training. Those three main regulatory processes include integrated product support, material release, and material fielding. Under each of these main process phases, there are multiple sub requirements that need to be tracked to ensure successful and sustainable material fielding to the soldiers. Currently, there is no mechanism for the project managers and higher-level leadership to easily attain information on where the project is currently standing within the regulatory process. Currently those processes are just followed up via email and excel spread sheets. The main issue of not having a unified platform to track the regulatory steps is that there is no way for the project managers to know the status of a regulatory phase without asking around the office. Another issue that can arise due to the current state of the progress tracking system is that it is extremely easy to miss a sub-step, because of how many steps need to be completed. In the case that a step was missed due to human error, the Army will reject the project, until all steps are completed, even if the equipment was already sent to the Army facility.

For this project we decide that the best solution to PEO STRI's problem was to create a software package that can track and display the current progress of all the ongoing projects at PEO STRI and enable them to have easier access to accurate information in an easy-to-understand dashboard that quickly shows the progress of project and allows for approvals to move to the next step of the project. This software will also allow a more detailed look at specific projects compared to the normal view that shows a high-level overview with averages across all projects.

2.3 Functional and Non-functional Requirements

Functional Requirements

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.

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.

Templates should exist
Templates can 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.

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.

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 them.

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.

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.

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.

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.

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.

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.

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.

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.

Logisticians must be able to compile all information into a single location
All information about a process can be displayed to the user from 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.

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.

The system should feature its own login system
A login system is important for ensuring the integrity of the system. Authenticating users will provide this login system.

The system should have an administrative role
A user group can be created with administrative permissions and granted to system administrators.

Nonfunctional Requirements

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.

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.

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.

The software should mitigate risk
A formatting issue or incorrectly signed document can cause significant 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.

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.

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.

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.

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.

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.

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.

Steps in a task can have multiple levels
A task step could potentially require multiple sub-steps, such as 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.

2.4 Limitations

The system needs to be web-based with a client-server-database structure and needs to run on the Army's existing cloud system. This applies to a few design constraints 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.

3. Concept Generation and Analysis of Alternatives

3.1 Literature Search

To generate the concept of how we will provide a solution to the problem statement of PEO STRI, the general approach is to create a system that will act like an ERP system, but instead of tracking inventory, we would make the system track the status of each of the approval phases processes needed to be completed for field approval. Currently, the way that PEO STRI is managing and controlling the fielding process is via email and office 365 tools. Taking that into consideration, we knew that the system that we will create for PEO STRI must be able to control and monitor the entire process. As a result of that, we will need to create a tool that besides having characteristics of an ERP system, will also need to have some of the characteristics that a project management software provides (such as Microsoft Project). Combining the benefits of both software's could help significantly reduce the overall time that the entire process takes, just by the fact that all parties that are involved in the approval process could have live access to check on the status of a project and be notified when there are any issues.

Based on our initial research, we wanted to check if there is any type of software that could provide a solution to PEO STRI. As we thought, there are some general ERP systems such as Oracle, and live project management software such as Asana, but we could not find a system that provides the benefits of these to software at the same time. In addition, PEO STRI has specific needs that none of the current software can provide.

3.2 Functional Decomposition - (Appendix Diagram 1a)

The system is comprised of 5 main functional features: Project/Project Template Administration, Project Proceeding, Logistical Views, Automated Notifications, and Permissions System. For the project/template administration, the system must allow a user to create, modify or remove projects and templates. For a project/template addition, the system will follow a process of:

1. Select a Template - The user will have the option to start a project from a template or create a new project from scratch.

2. Capture General Project Information - The system will capture the general project information from the user.

3. Alter/Create Tasks - The user will be able to create or modify tasks of the project.

    a. For each task, there can be multiple levels of subtasks.

    b. For each of these tasks, the system must:

        1. Capture General Task information – Task name, due date, etc. (more will be determined in the future)

        2. Add User to Task – Set a user or user group who is responsible for the task's-completion.

        3. Add Task Dependencies – Link this Task to another task either by blocking it from starting or being blocked by the other task.

4. Save As Template - The user will have the option to save the current settings as a template.

5. Create Project – Finally, the system will create the project with the settings specified by the user.

As per the modification and removal of the project administration, the system should prompt the user to select a template or product. Then allow the user to modify the item or delete the item. For the second function, Project Proceeding, the user will have a dashboard to process through any awaiting tasks that they have on any project they may be assigned to. The system must notify the user when a task they are assigned to becomes available. Upon availability, they will be able to process through the requirements of the task. Those requirements range from authorizing a document, uploading a document, applying a comment to the task, or checking the task as completed. The system should repeat this process until all tasks are complete. Once all tasks are complete the system will mark the project as complete and archive it.

For the third function of the system, the system should provide a dashboard for logisticians and project managers to view all ongoing projects and filter through all projects. The specific views and information needed to be populated is still being determined by our team.

To move onto our fourth function, the system needs to provide automated notifications to users of the system based on certain events. The specific notifications we need to handle are:

- When a user gets a new task available to them.
- When a task is considered overdue or off track, it should notify the user and project managers/logisticians.
- When a task is completed, it should notify the project manager, and those assigned to the task.

Finally, for our fifth function, the system must handle user group permissions. The system must have a set of user groups that each have their own specific permissions for using the system. The system must allow a user to create, modify, and delete different user groups. In the creation of a user group, the system must collect the general group information (name, color, etc.). Next, the system must allow the user to apply permissions to the user group from a predefined list. Once they have finished, the system will create the user group. The modification and deletion will perform the same functionality, but first allow the user to select a group to modify or delete. All of this can be seen in the diagram located in Appendix 1: Figure 1a.

3.3 Data Flow Diagrams

3.3.1 Context Diagram – (Appendix 1: Figure 1b)

The system has three types of users, which each have special operations they can apply to the system. Approvers can mark tasks as complete and can view the status of more tasks than logisticians can. Logisticians can add or update task steps and view the status of their assigned tasks. System administrators can add or update task templates and are able to view and respond to system faults.

The system has two internal databases: a distributed file system (DFS) for document storage and a metadata database for quickly referencing the files in the DFS. The system maintains both file store locations through additions and modifications and can retrieve relevant data. An external login system is used for authenticating users, and an email system is used to send notifications to the user if a task is delayed.

3.3.2 Level 0 Diagram – (Appendix 1: Figure 1c)

This diagram shows a breakdown of the system into its component systems. Users interact with the web client, which verifies their login using the login database. The client then passes the user's request to the Task Manager, which maintains a Task Database for the system's tasks. The Task Manager abstracts file management and interaction to the File Storage Manager, which maintains a Distributed File System that stores the files and a metadata database for indexing the filesystem and retrieving common data.

The Task Manager also responds to Web Client requests by processing them and sending relevant data back to the Client. The Task Manager also sends notifications through an Email System for tasks that are off-track which are routed back to the users. These notifications are also sent back to the web client.

4. Prototyping Process

The prototyping process has consisted of the development of user stories for agile development and basic wireframe mockups of the user interface. The wireframes for the interface have been described in the Software Requirements Specification Document that has been previously developed and submitted to PEO STRI for review. In the future of this project, we plan to develop a paper prototype of the user interface prior to development of the system user interface.

4.1 Agile User Stories

This proposed system includes eight epics as follows:

Epic 01: As a user, I want to manage projects so I can deliver, maintain, and publish projects.

- User Story 01: As a logistician I want to create projects so I can deliver projects.
  - Task 1: User interface – Medium (6/10)
    - A. On the project administration page, add a button titled "Create a Project"
    - B. On the click action of "Create a Project", the user should be redirected to a "Create a Project" Page.
    - C. The Create a Project page needs to be created with a form featuring project information fields.
  - Task 2: User Interface and Application Server Interaction – Medium (6/10)
    - A. On the submit button action of the user interface, the client needs to submit a POST HTTP request to the application server containing the form fields as the body.
  - Task 3: Application Server – High (9/10)
    - A. On the application server, a route needs to be created to receive the HTTP POST request from the client interface.
    - B. On the route, the server needs to:
      - Verify authenticity and permissions of the requestor.
        - On failure a response code of 401 is returned to the requestor.
      - Validate body of request.
        - On failure a response code of 400 is returned to the requestor.
      - Connect to the database and store project information in the projects document.
      - Issue a 201-response code.
  - Task 4: Database – High (8/10)

        A. On the database, a Projects document needs to be created so that the projects can be stored upon request.

- User Story 02: As a system administrator I want to edit projects so I can fix mistakes that were made when creating.
    - Task 1: User interface – Medium (6/10)
        A. On the project administration page, a product list should be shown. From that list, there should be a button titled “Edit Project”.
        B. On the click action of the “Edit Project” button, the user should be redirected to a “Edit a Project” page, which is styled like the “Create a Project” page.
        C. The page should pre-fill the form fields of the create a project.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. On the submit button action of the user interface, the client needs to submit a PUT HTTP request to the application server containing the form fields as the body.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route needs to be created to receive the HTTP PUT request from the client interface.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate body of request.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and update the project information in the projects document.
            - Issue a 200-response code.
- User Story 03: As a project manager I want to archive projects after completion so I can mark them as complete.
    - Task 1: User interface – Medium (6/10)
        A. On the project administration page, a product list should be shown. From that list, there should be a button titled “Archive Project”.
        B. On the click action of the “Archive Project” button, the user should have a modal appear asking to confirm their action.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. Once the user confirms their action on the user interface, the client needs to submit a PUT HTTP request to the application server with the _key of the project as path of the request URL.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route with a dynamic path for the _key of the project needs to be created to receive the HTTP PUT request from the client interface.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate the supplied _key from the route path.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and update the project status to archived.
            - Issue a 200-response code.

Epic 02: As a user, I want a mechanism to duplicate process workflow for projects so I can save time creating new projects.

- User Story 01: As a project manager, I want to be able to create template processes, so I can streamline project creation for projects with the same workflow.

    - Task 1: User interface – Medium (6/10)
        A. On the project creation page after the form and before the submit button, there should be a button titled “Save as Template”.
        B. On the click action of the “Save as Template” button, the user should have a modal popup. On this modal there should be a text entry to name the template followed by a submit.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. On the submit button action of the user interface, the client needs to submit a POST HTTP request to the application server containing the form fields as the body.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route needs to be created to receive the HTTP POST request from the client interface.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate body of request.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and store the template information in the templates document.
            - Issue a 200-response code.
    - Task 4: Database – High (8/10)
        A. On the database, a Templates document needs to be created to store the templates.
- User Story 02: As a Project Manager, I want to create a project using a template, so I do not have to create all tasks and subtasks individually.
    - Task 1: User interface – Medium (6/10)
        A. On the project administration page, there should be a button titled “Create a Project from Template”.
        B. On the click action of the “Create a Project from Template” button, the user should be presented with a modal popup containing a dropdown list of all project templates.
        C. Upon selection of a template, the user will be redirected to a project creation page with a pre-filled form of the template contents.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. On the project administration page on the modal popup, the client needs to fetch template names from the server via a GET HTTP request.
        B. During the redirect from the project administration page to the project creation page, the template information will need to be fetched from the server via a GET HTTP request while supplying the selected templates _key in the route path.
    - Task 3: Application Server – High (9/10)
        A. The server must have a GET HTTP route created to fetch all template names.
        B. The server must have a GET HTTP route with a dynamic path for the _key of a template to fetch the specific template information.
        C. On each of the routes, the server must:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate the request contents.
                - On failure a response code of 400 is returned to the requestor.
            - Select the specified information from the database.

        - Return the requested information to the user with a 200-response code.
- User Story 03: As a Project Manager, I want to edit a premade template so I can alter the process workflow.
    - Task 1: User interface – Medium (6/10)
        A. On the project creation page, there should be a tab which will enable the user to view a list of templates.
        B. On the list of templates, there should be a button titled "Edit" to edit the template.
        C. On the click action of the "Edit" button, the user will be redirected to the "Edit a Template" page which should be styled like the template creation page.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. Once completed with the edits on the editing page, the user will submit which will cause the client to submit a PUT HTTP request to the server with the form fields contained in the request body. The _key of the template being edited will be supplied into the route path.
    - Task 3: Application Server – High (9/10)
        A. The application server must have a PUT HTTP route with a dynamic path for the _key of the template created to handle the template updates.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate the request contents.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and update the template supplied.

Epic 03: As a user, I want to be able to complete tasks so I can progress through a project.

- User Story 01: As a logistician I want to upload files so I can prepare documents for approval.
    - Task 1: User interface – Medium (6/10)
        A. When viewing a project, there will be a list of tasks associated with the project.
        B. Upon selecting a task, a modal will appear where the user can view more information.
        C. If the user is assigned to this task or has a specific permission, they will have the ability to complete this task file upload action.
        D. On the file upload action item, the user will have a button which will open a desktop file explorer to upload a document.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. After selecting their file, the file will be automatically uploaded to the application server via a POST HTTP request while supplying the path with the _key of the task and project. As well as a multipart/formdata for the request body.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route with a dynamic path for the _key of the task and project needs to be created to receive the HTTP POST request from the client.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate body of request.
                - On failure a response code of 400 is returned to the requestor.
            - Write the file from the request body to the file system.

        - Connect to the database and store the file object on the task located in the Projects document.
- User Story 02: As an approver, I want to approve documents so I can allow projects to progress.
    - Task 1: User interface – Medium (6/10)
        A. When viewing a project, there will be a list of tasks associated with the project.
        B. Upon selecting a task, a modal will appear where the user can view more information.
        C. If the user is assigned to this task or has a specific permission, they will have the ability to select a button to complete the task.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. Once the user selects complete on a task, the client will submit a PUT HTTP request with a supplied path for the _key of the task and project to the server.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route with a dynamic path for the _key of the task and project needs to be created to receive the HTTP PUT request from the client.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate body of request.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and update the status of the task to complete on the task located in the Projects document.
- User Story 03: As a logistician I want to add comments so I can resolve issues with approvals.
    - Task 1: User interface – Medium (6/10)
        A. When viewing a project, there will be a list of tasks associated with the project.
        B. Upon selecting a task, a modal will appear where the user can view more information.
        C. If the user is assigned to this task or has a specific permission, they will have the ability to attach a comment through a textbox entry.
    - Task 2: User Interface and Application Server Interaction – Medium (6/10)
        A. Once the user submits their comment on a task, the client will submit a PUT HTTP request with a supplied path for the _key of the task and project to the server.
    - Task 3: Application Server – High (9/10)
        A. On the application server, a route with a dynamic path for the _key of the task and project needs to be created to receive the HTTP PUT request from the client.
        B. On the route, the server needs to:
            - Verify authenticity and permissions of the requestor.
                - On failure a response code of 401 is returned to the requestor.
            - Validate body of request.
                - On failure a response code of 400 is returned to the requestor.
            - Connect to the database and attach the comment in the comments object of the task located in the projects document.

Epic 04: As an administrator, I want a mechanism for users to have distinct roles and permissions so I can improve security and focus project visibility.

- User Story 01: As a system administrator I want to create roles for users so I can manage their access and administrative functionality.
    - Task 1: User interface – Medium (6/10)

A. Create a “Role Management” button on the user’s main page
B. Link the new button to a “Role Management” page
C. Create a “New Role” button on the “Role Management” page
D. The “New Role” button shows a form to the user to add permissions and a name
E. Create a “Apply Role” button on the “Role Management” page
F. Link the “Apply Role” button to a “Apply Role” form
G. The “Apply Role” form should present a list of users with an add button

- Task 2: User Interface and Application Server Interaction – Medium (6/10)
  A. On the submit button of the user interface for the “Create Role” form, the client needs to submit a POST HTTP request to the application server containing the form fields as the body.
  B. On the submit button of the user interface for the “Apply Role” form, the client needs to submit a PUT HTTP request to the application server containing the form fields as the body.
- Task 3: Application Server – High (9/10)
  A. On the application server, a route needs to be created to receive the PUT and POST requests from the client interface.
  B. On the route, the server needs:
     - Verify authenticity and permissions of the requestor.
       - On failure a response code of 401 is returned to the requestor.
     - Validate body of request.
       - On failure a response code of 400 is returned to the requestor.
     - Connect to the database and create or update the role information in the roles document.
- Task 4: Database – High (8/10)
  A. On the database, a Roles document needs to be created to store the system’s roles.
  B. The Users document needs to be extended to store a user’s roles.

- User Story 02: As a system administrator I want to restrict system access so I can ensure security and integrity.
  - Task 1: User Interface and Application Server Interaction – Medium (6/10)
    A. All user interface interactions must be cross-referenced with the authenticated user’s permissions
    B. If permissions are not met, the menu button or page should not render or present a HTTP 403 Forbidden status page.
  - Task 2: Application Server – High (9/10)
    A. On the application server, routes need to be updated to verify the user has permission to complete the request.
    B. These routes will connect to the database and request the user’s permissions based on their session

5. Results

Overall, the project was completed in almost its entirety. As per the implementation status, we have designed and developed all listed functional and non-functional requirements which were stated during the requirements process. Unfortunately, a few added requirements were unable to be completed. Those being the user permission restrictions and email notifications. Although those weren't completed, the system was developed with hooks to easily adapt those features. The system was successfully handed off to PEO STRI on (03/29/22) and we are in continuous communication with MITRE, their contracted developers, to ensure a smooth transition.

Our final product is capable of handling all of their logistical paperwork needs using a stepped task system. This system will allow for quick and easy navigation through the bureaucratic paperwork that is required by army regulation for the projects that are handled by PEO STRI as well as giving them the ability to immediately see the current progress on any project as well as to note who the system is waiting on before moving to the next step.

All of these features lead to a very complicated system with many lines of code that have to all work together perfectly in order to perform their functions consistently. This contributed to us having to take a substantial amount of time on final product testing and debugging to ensure that the system was operational for our final deliveries of the product.

5.1 Final Product Testing

During development, we encountered many bugs. Most of these were discovered during implementation of front-end user interface processes. Many project creation and viewing bugs were discovered during implementation of these features. These bugs were mostly identified thanks to input validation on the backend or incorrect return status from the frontend requests. Locating the root cause of these bugs was done by examining the request body and error stack trace. The backend displays erroneous request bodies, which makes investigating what caused the error much easier.

Some bugs, such as incorrect suspense dates or user interface issues were much harder to track down. These bugs were mostly discovered through manually comparing expected results against the system's output. Since these bugs are more fundamental to system design (rather than a malformed HTTP request), identifying the root cause and resolving its issue(s) was much more difficult. Several frontend bugs caught late in development were very difficult to investigate.

Before and during the Capstone showcase we discovered other minor bugs, which we were able to resolve that day, before handing the system over to PEO STRI. Thankfully, none of these issues affected our ability to present the system's features to the public. The system that we handed over to MITRE contains an automated black-box backend request testing suite and in-line documentation to assist with later development and bug fixing.

6. Conclusions

In concluding the project and the results, we delivered to PEO STRI a solution to their initial problem statement. The system can streamline and improve all the processes by 50% needed to be completed before fielding approval. The system will allow a more efficient workflow and allow soldiers to get their training equipment much faster.

6.1 Summary

The initial problem statement that PEO STRI had was that it was difficult for them to manage and track projects that have many processes that must be completed before fielding approval. All the documentation has been sent for approval via email, which made the process long, inefficient, and in some cases, would cause a loss of track of the progress.

To deliver a solution to PEO STRI's problem statement, there were two main actions that we needed to take. First of all, our computer scientists were tasked to develop a software package capable of creating a single work environment for all the processes. The software needed to be capable of allowing the users to upload documents to the system (instead of emailing), reviewing/approving the documents by multiple parties, and most importantly, would be able to track the stages of the whole project making sure it does not pass the defined suspense date. Secondly, with the capabilities of the new software, we could improve PEO STRI's current workflow for each process. This is mainly a result of the capability of allowing all defined users to work on the document simultaneously. With that capability, there is no need for the document to be passed by one person at a time, which was the main reason for the initial inefficiency. To make those adjustments to how all the processes will flow in the new system, the business analysts have received from PEO STRI flowcharts of all current processes. We have created an improved version for each flowchart tailored to the new system's capabilities and works along the military hierarchy when needed.

Throughout the two semesters of Capstone, the team reached most of the defined goals and stayed within the defined scope. PEO STRI was happy with the final project results and believes that they will be able to improve their workflows by 50% with the new system.

There are minor parts of the system that the team could not complete due to time restrictions; however, the system can deliver a solution to PEO STRI, and the system was handed over to MITRE for further development.

6.2 Individual Contribution

Team Member	Design	Development	Deliverables
Braden	30%	45%	25%
Henry	30%	45%	25%
Timothy	20%	5%	25%
Ben	20%	5%	25%

6.3 Reflection Report

First of all, we all have an excellent group dynamic, and everyone did their best to contribute to the project. Also, our project sponsors were constant updates on our work progress as we had biweekly follow-up meetings. Besides the biweekly meetings with the sponsor, they were very responsive to any questions we had between the follow-up meetings.

As a team, we met most of the goals that we had set; however, some of the system capabilities that the sponsor had asked for were not implemented due to time constraints. That being said, we all believe that there was an uneven workload distribution between the team members. To achieve a solution to the problem statement of PEO STRI, the majority of the project was to build software that would execute the improved workflows. As a result, Braden and Henry (the CS majors of the team) had to take a heavy load of developing software. At the same time, the task of streamlining the workflow of each of the processes was not as complex and was distributed between Ben and Timothy (the Business Analyst of the project). That being said, we all agree that we would have been able to complete the project in a better way if the team had three computer scientists and one business analyst. Having three computer scientists would allow them to develop the software more efficiently and accomplish much more.

On the other hand, one business analyst could have still completed all the work of improving the workflow alone. Despite the uneven workload, we all agree that it was necessary to have business analysts on this team, as they were the ones that found a way to improve the current workflows. They were the ones that guided the software development actually to provide the solution to PEO STRI.

Appendix 1 - Prototyping Process

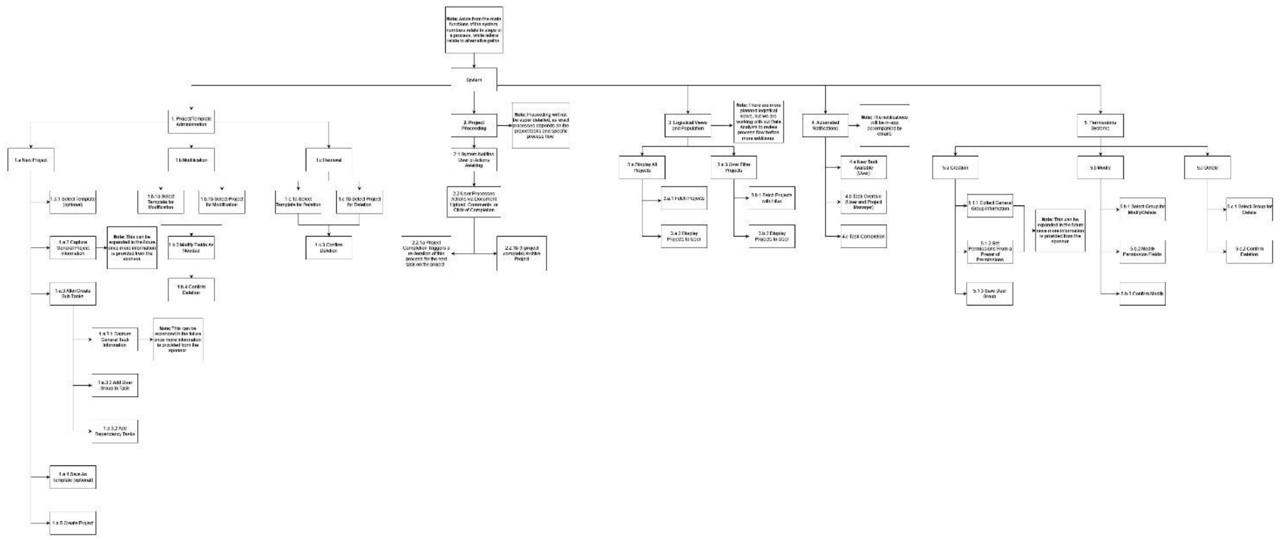

Figure 1a. Functional Decomposition diagram (https://imgur.com/v3CCATc)

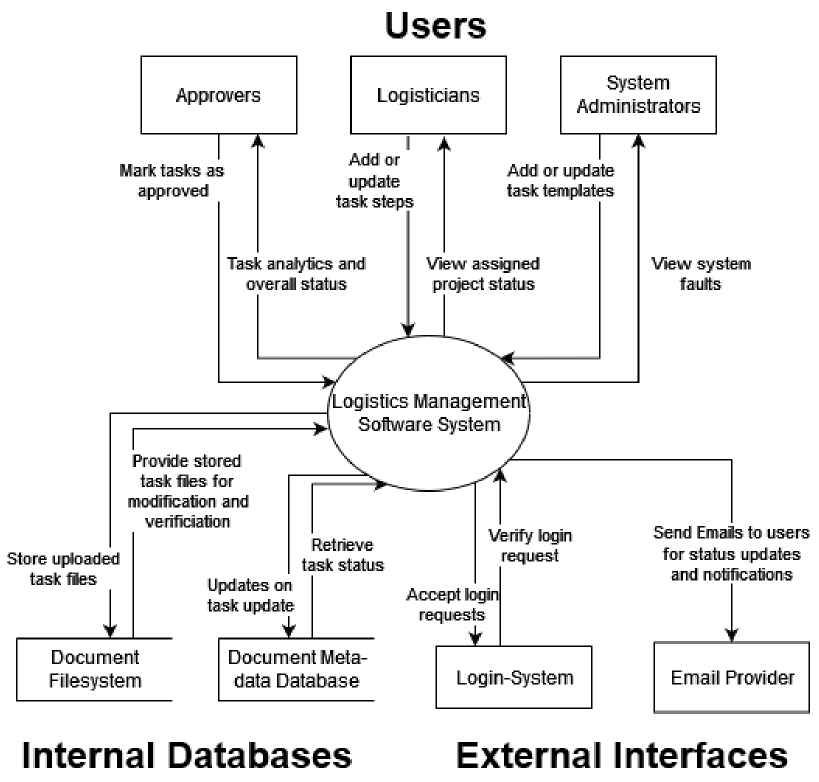
Users
Approvers
Logisticians
System Administrators
Mark tasks as approved
Add or update task steps
Add or update task templates
Task analytics and overall status
View assigned project status
View system faults
Logistics Management Software System
Provide stored task files for modification and verificiation
Store uploaded task files
Retrieve task status
Updates on task update
Verify login request
Accept login requests
Send Emails to users for status updates and notifications
Document Filesystem
Document Meta-data Database
Login-System
Email Provider
Internal Databases
External Interfaces

Figure 1b. Context Diagram

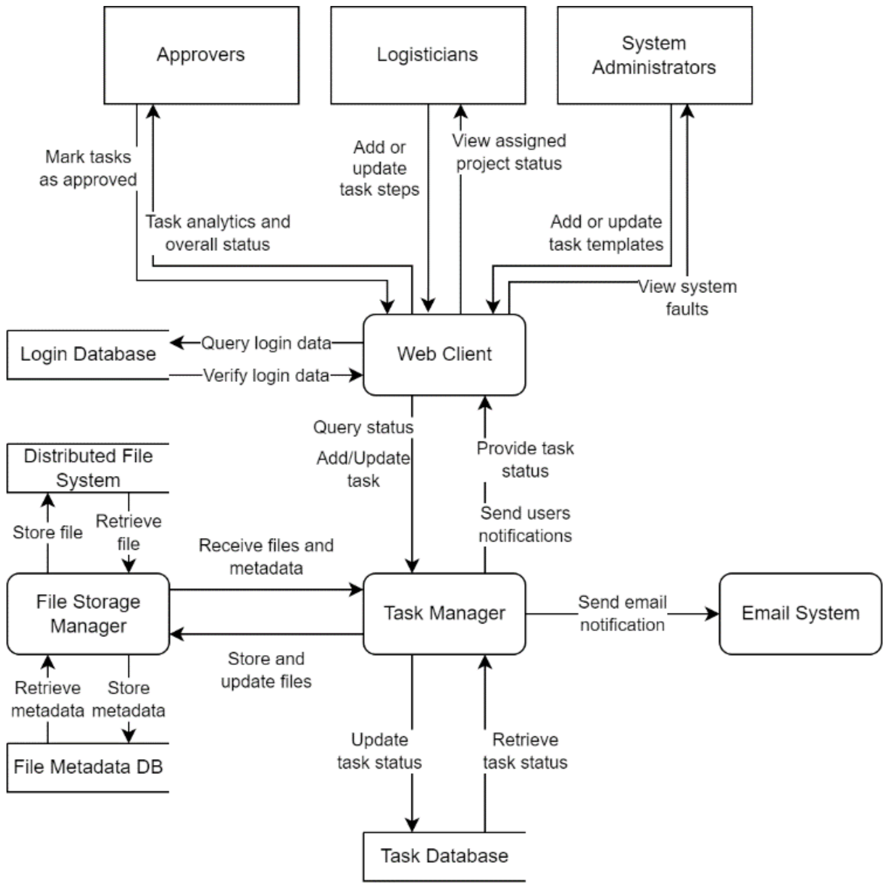

Figure 1c. Level 0 Dataflow Diagram

Appendix 2 – Workflow Improvement

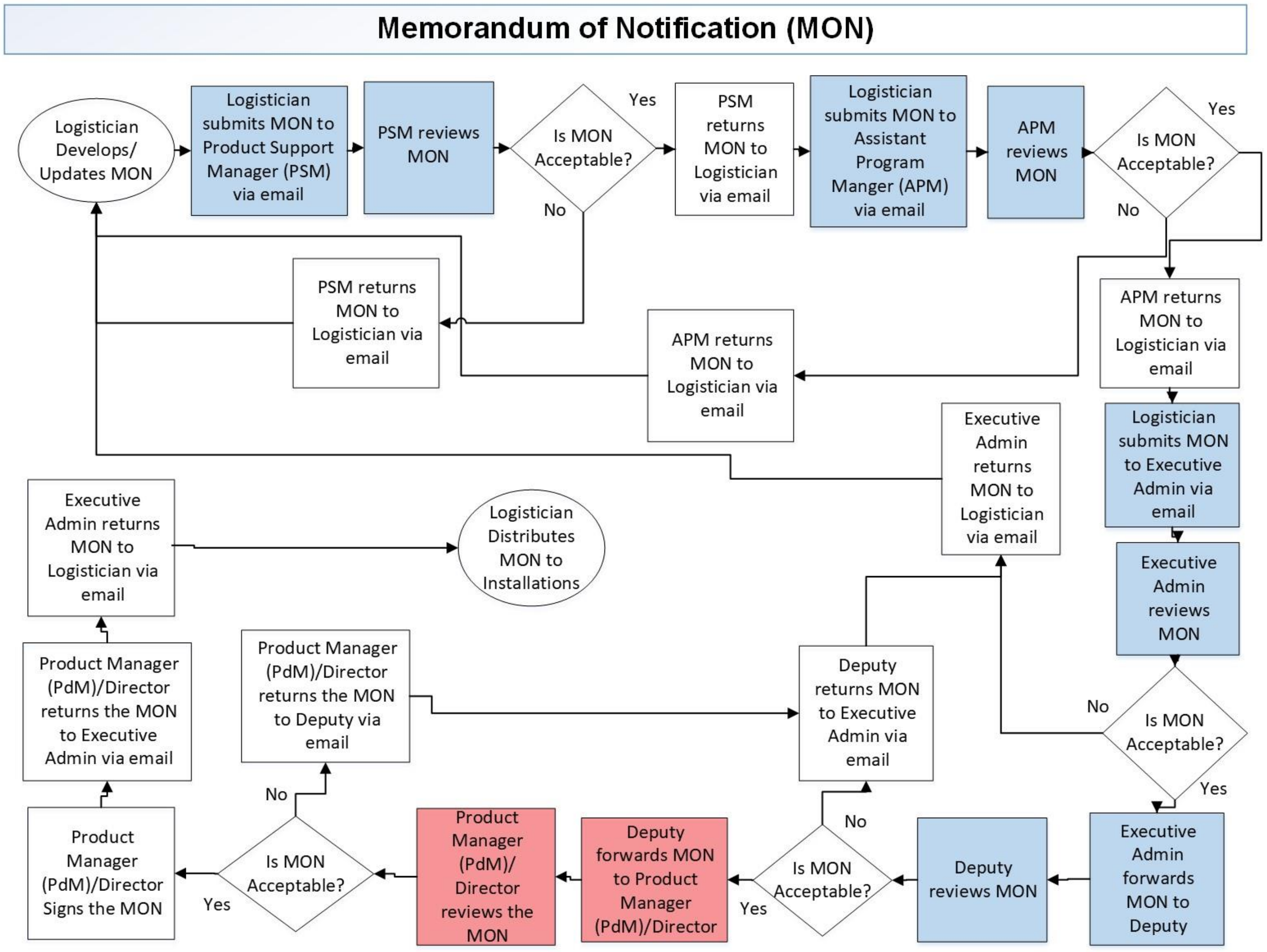
Memorandum of Notification (MON)
Logistician Develops/ Updates MON
Logistician submits MON to Product Support Manager (PSM) via email
PSM reviews MON
Is MON Acceptable?
Yes
No
PSM returns MON to Logistician via email
Logistician submits MON to Assistant Program Manger (APM) via email
APM reviews MON
Is MON Acceptable?
Yes
No
PSM returns MON to Logistician via email
APM returns MON to Logistician via email
APM returns MON to Logistician via email
Logistician submits MON to Executive Admin via email
Executive Admin reviews MON
Is MON Acceptable?
No
Yes
Executive Admin returns MON to Logistician via email
Executive Admin forwards MON to Deputy
Deputy reviews MON
Is MON Acceptable?
No
Yes
Deputy returns MON to Executive Admin via email
Deputy forwards MON to Product Manager (PdM)/Director
Product Manager (PdM)/ Director reviews the MON
Is MON Acceptable?
No
Yes
Product Manager (PdM)/Director returns the MON to Deputy via email
Product Manager (PdM)/Director Signs the MON
Product Manager (PdM)/Director returns the MON to Executive Admin via email
Executive Admin returns MON to Logistician via email
Logistician Distributes MON to Installations

Figure 2a. Example of Previous MON workflow

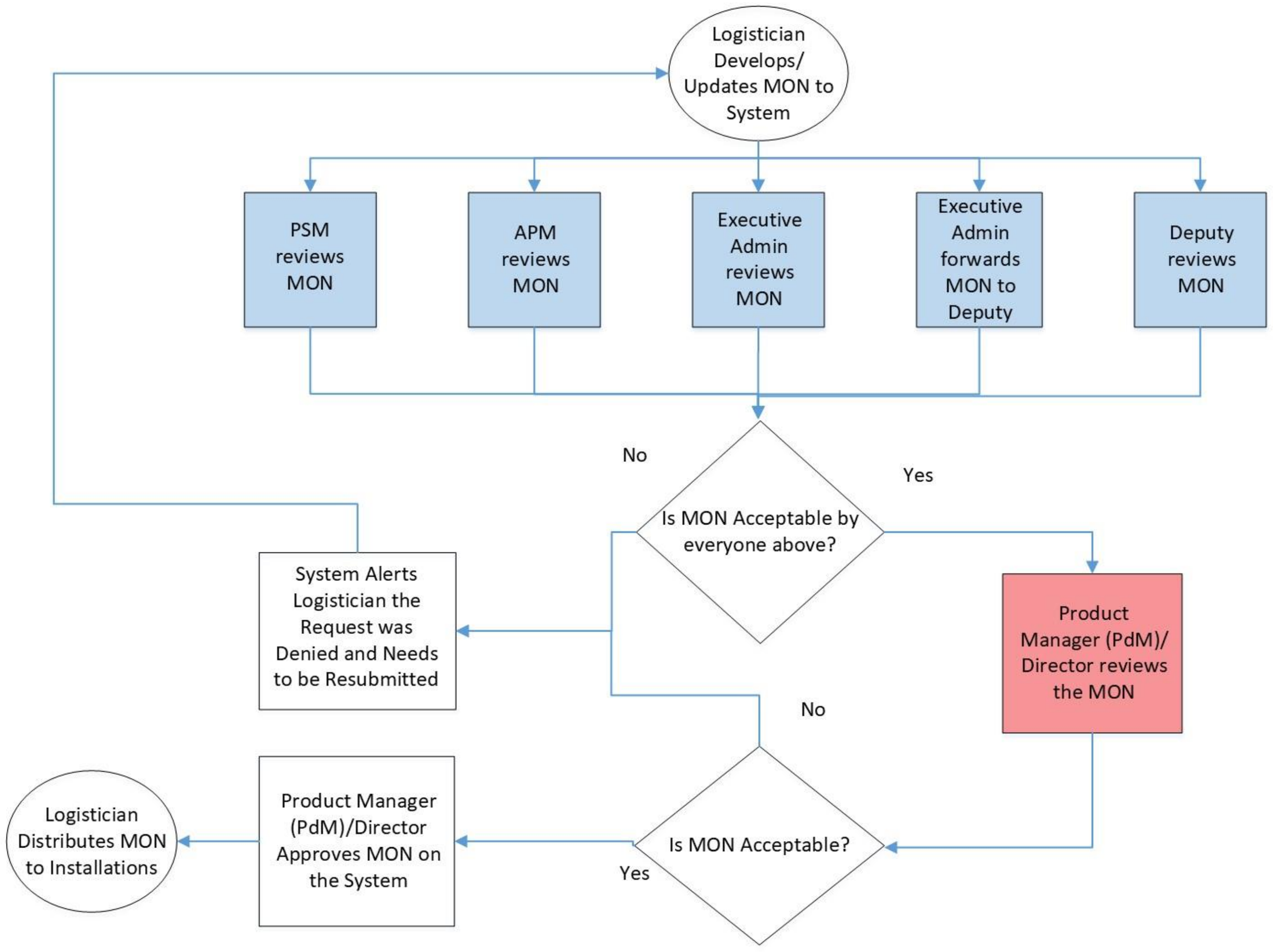
Logistician Develops/ Updates MON to System
PSM reviews MON
APM reviews MON
Executive Admin reviews MON
Executive Admin forwards MON to Deputy
Deputy reviews MON
No
Yes
Is MON Acceptable by everyone above?
System Alerts Logistician the Request was Denied and Needs to be Resubmitted
Product Manager (PdM)/ Director reviews the MON
No
Logistician Distributes MON to Installations
Product Manager (PdM)/Director Approves MON on the System
Is MON Acceptable?
Yes

Figure 2.b Improved MON Workflow

Appendix 3 – Individual Reflections

Braden:

Throughout this project my role was of a Frontend Developer and Tech Lead. My duties in the role of tech lead were to communicate our progress and features of the system to the project sponsor, ensure steady progress on the development, and communicate with MITRE to hand off the system. As the frontend developer, I focused on developing the user facing side of the web application. For this I decided to use ReactAdmin, MaterialUI, and deploy the service with Express.js. Overall, I thoroughly enjoyed the frontend development, except for designing the interface stylings and layout. I prefer to be given a mockup or wireframe, but due to our team setup there was no one who could fill the role of designer. Despite disliking the design phase, I feel as though I came up with a user interface that is intuitive and extremely functional.

Regarding the software stack, I went into this project with JavaScript knowledge, but no specific expertise in MaterialUI or ReactAdmin. Therefore, the first month of development was mainly spent getting a grip on the software stack and spending countless hours reading through documentation of those libraries. By the end of the development, I had a great understanding of the entire software stack and its place in our system.

After many late nights and over 10,000 lines of code, we developed a great application that I feel perfectly solves the proposed problem, while being extremely scalable in hopes of being a solution to others. This system was extremely large and complex, and if I were to do it over again, I would not have promised so many features in such a short period of time. I feel as though the rushing of completing certain aspects or features of this application jeopardized the quality and limited our ability to test extensively.

Going into this project I had extremely high hopes of what we could accomplish. Sure, enough we nearly completed the project in its entirety, but I feel it was done under extreme pressure and haste. This project shouldn't have been categorized as a DSBA project and included two of those expertise members. This project should have been more focused on and included more computer science members, as roughly 90% this project was focusing on the development of a web application.

Despite our disadvantages placed by the poor planning of our capstone directors, we developed an enterprise level application that I am extremely proud to be a part of. This project acted as a great learning experiment and introductory into the professional world. I am extremely grateful for PEO STRI and MITRE for working with us and allowing us to solve a real-world problem for them.

Henry:

My role was a Backend Developer. My main role was development and design of the backend API routes that accept user requests from the interface and dictate the actions that the system should take. This involves input handling, database manipulation, file storage, and internal system processing.

The backend was written with Typescript in NodeJS, using Koa as a web framework and ArangoJS for database management. The backend was designed with modularity and scalability in mind. Every data structure the system manages uses a shared class, which implements all necessary functions for processing. The system utilizes a specialized schema, which can be easily modified, to control what inputs are valid and how to transform raw database objects into request data.

I went into this project with a lot of general programming language knowledge from personal and school projects, but little experience with web applications or Typescript. The early stages of the backend were rough, built from my inexperience and lack of understanding of the nuances of the language and frameworks. Over time, I learned more about the language and its abilities, and became much more adept at designing and implementing the system in a more correct way. If I could start over, there are structural changes I could make that would make the system much better, using knowledge gained from development.

By the end of development, the final system has accrued a lot of technical debt. While the system does work, some features are implemented in a temporary sort of way and are not as resilient as they could be, especially for deployment on PEO STRI systems. Despite this, I think the product we created is excellent and beyond my expectations.

The sponsor had many requirements, requiring the system to be built extremely rapidly. Having another computer science student would have greatly improved the amount of work we could have performed. Additionally, many fundamental system details were not revealed until the middle of development. The system's internal project hierarchy initially seemed ideal, however PEO STRI's process flowcharts and integration with each other would have benefited from a dependency map rather than a fixed hierarchy.

Despite all the difficulty, I greatly enjoyed this project and the opportunity to work in a professional environment with PEO STRI, MITRE, and my fellow project members. I am glad that we were able to provide most of the sponsor's features, even if some of them still need work or could be improved.

Ben:

Throughout the project, my role in the team was the project manager and coordination between the team and PEO STRI. During these roles, I made sure that the project was within the defined scope, on track, and that there was alignment between all team members on the project's progress. In addition to that, I was the main point of contact between the team and Destiny (the primary contact of PEO STRI). I have coordinated with her all our biweekly meetings and communicated any questions that the team had.

Besides acting as project manager, I have also focused on improving the current workflows of PEO STRI, as this was one of the project's main objectives. The improvement process started with PEO STRI sending us flowcharts of all the functions. Initially, looking into the flowcharts, it was noticeable that the current workflow is inefficient and time-wasting as it was done via email. To find a way to streamline each of the processes, I have brainstormed with the CS members on how to build an environment that would allow all users to work on a project at the same time (instead of emailing) and also to provide a way to track and control a project. Once the CS members had defined all the user needs, I could start working together with Timothy to create new flowcharts of all the processes tailored to the new system capabilities.

After improving all the process workflows, I have assisted our CS member in testing the system for any bugs. In addition, as Braden requested, I have helped him design the dashboard. That included choosing what widgets each user will need to see to get the maximum efficiency and control on the tasks that they need to complete.

I am happy with the software that Braden and Henry have worked very hard on, and so our sponsors were pleased with the software. I believe that the system will significantly increase their work efficiency and reduce the time spent on each project by over 50%. I believe that all team members have out the maximum effort in the project, but I think this project should have been labeled as a CS project and not a DSBA project. Despite that, I believe that it was necessary to have a Business Analyst on the team as part of the problem was to improve workflows. However, Braden and Henry could have used an additional programmer, while only one business analyst could have worked on the improvement aspect alone.

Timothy:

Throughout this project my main role was that of process flow analyzation. I started with analyzing PEO STRI's current flows and adapting them to the system that was mainly developed by Braden and Henry. This was accomplished by working with Ben as well. I also initiated contact with various people at poly whenever we needed to reserve rooms for sponsor meetings or reserve other materials needed for events such as the capstone showcase. These responsibilities happened randomly throughout the year as needed instead of being done in sprints like the other responsibilities that we had. Once the analysis and adaptation of current process workflows was complete, Ben and I both helped the computer science members of our team with debugging and user guide creation. We both spent quite a bit of time finding and reporting bugs to the computer scientists as well as informing them of various user experience design aspects that could improve the system. Ben and I also worked to create the final user help guides with him creating the written guide while I created a video guide to go along with the written one.

Overall, I would consider this project a success, as we were able to develop a high-level enterprise solution for managing the logistics of PEO STRI and providing live time data of their project status to the people in their organization that need it. Our sponsors also seemed very pleased with the product of the project and stated that they think once the application is deployed, it could improve their workflow by up to fifty percent. Throughout the entire project, our sponsors were great and able to help us with whatever we needed, which greatly aided in our success.

The one change, I would make to this project is that I don't think that we needed two Business Analytics students on the project and needed more Computer Science students. I would also suggest using a Data Analytics student instead of any Business Analytics student as well as having a minimum of three Computer Science students.