

Liberty Dash

Requirements Specification

Team Iceberg

Project Manager:
Maya Bergandy

Team Members:

Bryce Bodley-Gomes

Tony Gao

Kevin Silva

Matthew Hinsley

Bhavik Jain

Adrian Poveda McKay

Chenhao Huang

Zachary Tousignant

Michael Schiller

Table of Contents

[Table of Contents](#)

[1 - Introduction](#)

[1.1 - Purpose](#)

[1.2 - Objective](#)

[1.3 - Stakeholders](#)

[2 - Overall System Description](#)

[2.1 - Functionality](#)

[2.2 - External Users](#)

[3 - Functional Requirements](#)

[3.1 - Web Application Functionalities](#)

[3.1.1 - View Home Page](#)

[3.1.2 - Invoke Macro](#)

[3.1.3 - View Changelog](#)

[3.1.4 - View Submitted Peer Reviews](#)

[3.1.5 - Accept or Reject Pending Peer Reviews](#)

[3.1.6 - View Pending Processes](#)

[3.1.7 - View Processes on Hold](#)

[3.1.8 - View Running Processes](#)

[3.1.9 - View Successful Processes](#)

[3.1.10 - View Failed Processes](#)

[3.1.11 - Restart Processes](#)

[3.1.12 - Search Processes](#)

[3.1.13 - Cancel Processes](#)

[3.1.14 - Login to Account](#)

[3.1.15 - Logout of Account](#)

[3.1.16 - View Notifications](#)

[3.1.17 - View or Edit Settings](#)

[4 - Non-Functional Requirements](#)

[4.1 Exception Handling](#)

[4.2 - Security](#)

[4.2.1 - Login](#)

[4.2.2 - Encryption](#)

[4.2.3 - Access Policy](#)

[4.3 - Database](#)

[4.4 - Usability](#)

[4.4.1 Platform](#)

[4.4.2 - Ease of Use](#)

[4.5 - Hardware](#)

[4.5.1 Host Server](#)

[5 - Interface Design](#)

[6 - Future Functionality](#)

[6.1 - Performance Trends](#)

[6.2 - Service Level Agreements](#)

[6.3 - Create New Macro](#)

[7 - Glossary](#)

1 - Introduction

This is the introduction for the requirements specifications of the Liberty Dash system. This section will discuss the system's purpose, stakeholders, usability and benefits.

1.1 - Purpose

This system is going to simplify the process of running of SQL statements and pre-coded macros from Liberty Mutual. The main goal of this system is to streamline the execution of Liberty Mutual macros by implementing a user-friendly GUI. This GUI will simplify the input and manipulation of Liberty Mutual metadata by providing a user-friendly interface which will verify correct input of parameters by a combination of error checking, and Peer Review (see glossary for explanation). This will minimize the risk of human error. By reducing the portion of work that is done manually, there will be a much lower risk for data corruption.

1.2 - Objective

The Liberty Dash system is being implemented for Liberty Mutual, and is intended to be used by their employees. The objective of our system is to create a user-friendly interface that will automate the process of Liberty Mutual employees interacting with the External Liberty Mutual Database by launching macros. This will save employees a significant amount of time and effort.

1.3 - Stakeholders

This system is being developed for Liberty Mutual by students at the University of Massachusetts Amherst, and is sponsored by Israel Abraham and Joe Ellsey. The stakeholders of the system consist of project liaison Professor Anderson, as well as project managers and software development teams comprised of students from the University of Massachusetts Amherst. The final product is intended to be used by Liberty Mutual employees, specifically administrators and developers.

2 - Overall System Description

This section will provide a high-level description of the system and its functionality. This section will also identify and describe the user categories involved.

2.1 - Functionality

The Liberty Dash system is a stand-alone system that will have a GUI solution implemented for the Audit and Control Manipulation Macros. This interface will enable the user to run macros that manipulate data on the External Liberty Mutual Database. This “data” consists of metadata extracted from a data warehouse and stored into a database (Liberty Mutual External Database) for our application to manipulate. This metadata is currently manipulated manually via macros by Liberty Mutual employees. Our system will provide this same functionality for manipulation of the data through a simple user-friendly GUI. A peer review approval by another General User may (depending on settings in the system allowing the bypass of Peer Reviews in emergency situations) be needed before any changes to the External Liberty Mutual Database are executed. These changes will then be documented in the Changelog for future reference.

2.2 - External Users

The users of our system are categorized as General Users and Admin Users.

User Category	Description
General User	A user with access to all basic functionalities of the system.
Admin User	A user with access to the same functionality as a General User, with additional permissions.

Table 2.20: User Categories

3 - Functional Requirements

This section describes the behavior of the functions in our system. Each element describes the functionality of a specific use case.

3.1 - Web Application Functionalities

This section features all the functionality of the web application. Figure 3.1 below is a navigation flow diagram to visualize the paths a user can follow through our web application.

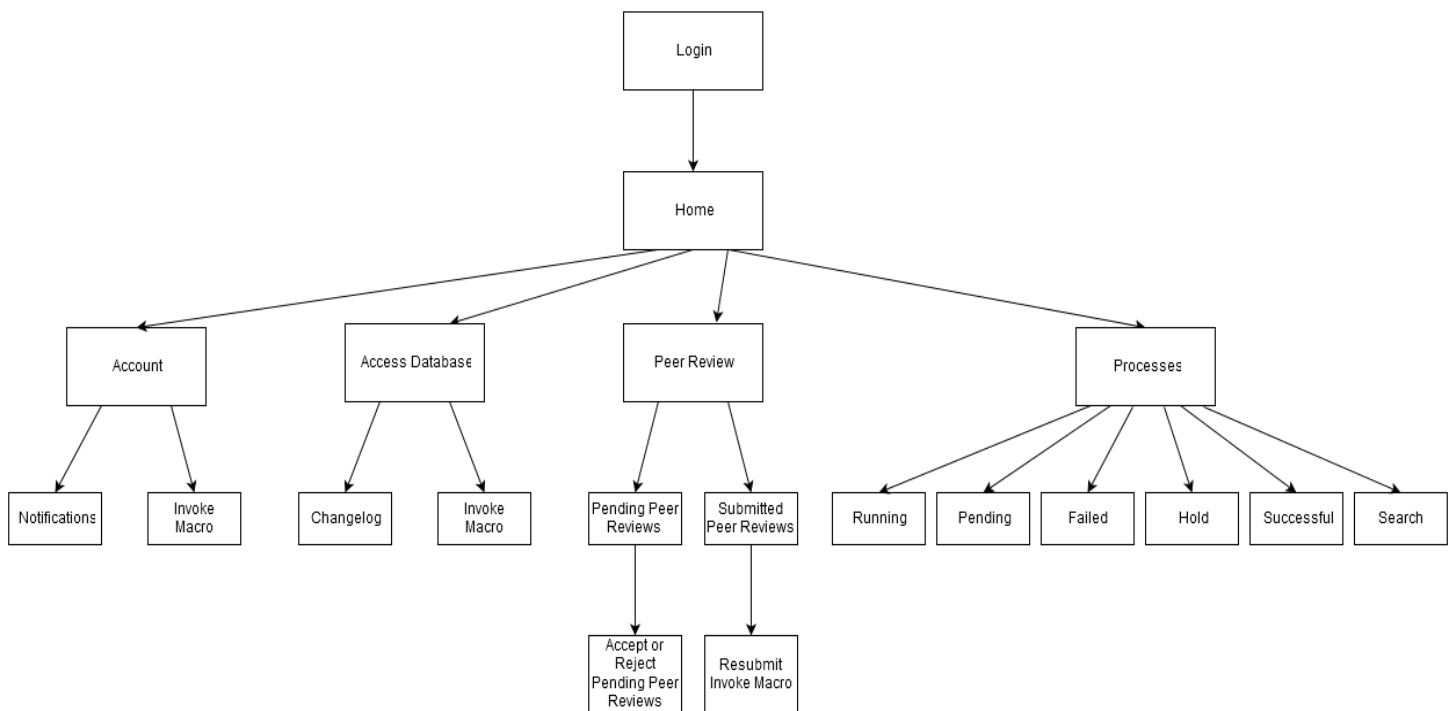


Figure 3.1: Navigation Flow Diagram

3.1.1 - View Home Page

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User has navigated to the “Home” page.

Steps:

1. The User clicks the “Home” button and is navigated to the “Home” page (see Figure 5.2).

3.1.2 - Invoke Macro

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User has submitted a request to edit the External Liberty Mutual Database.

Steps:

1. The User clicks on the “Access Database” dropdown menu.
2. The User selects “Invoke Macro” from the “Access Database” dropdown menu and is navigated to the “Invoke Macro” page (see Figure 5.4).
3. The User selects which macro they wish to execute.
4. The User inputs the necessary parameters for that macro.
5. The User adds any Tags they wish to associate with the process.
6. The User clicks the “Enter” button.
7. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18). When a user accepts the peer review of this macro the associated information (creator, peer reviewer, and macro parameters) are added to the Changelog.

Alternate Flow: Bypass Peer Review

6. The User deselects the Peer Review checkbox, and then clicks the “Enter” Button.
7. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18). The macro parameters, and the user who submitted it are added to the changelog at this point.

Exceptions:

1. If any parameters were input incorrectly in Step 5, the User will receive a popup notification detailing which fields are incorrect and a reasoning why. The User will then return to Step 5.

3.1.3 - View Changelog

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to view the Changelog.

Steps:

1. The User clicks on the “Access Database” dropdown menu.
2. The User selects “Changelog” from the “Access Database” dropdown menu and is navigated to the “Changelog” page (see Figure 5.5).

3.1.4 - View Submitted Peer Reviews

Actors: General User, Admin User.

Preconditions: User is logged in. User has submitted at least one Peer Review.

Postconditions: User is able to view any given Peer Review they have submitted that has not yet been approved.

Steps:

1. The User clicks the “Peer Review” dropdown menu.
2. The User selects “Submitted Peer Reviews” from the “Peer Review” dropdown menu and is navigated to the “Submitted Peer Reviews” page (see Figure 5.7).

3.1.5 - Accept or Reject Pending Peer Reviews

Actors: General User, Admin User.

Preconditions: User is logged in. User has Peer Reviews pending their approval.

Postconditions: The selected Peer Review is given a state of “Accepted” or “Rejected”.

Steps:

1. The User clicks the “Peer Review” dropdown menu.
2. The User selects “Pending Peer Reviews” from the “Peer Review” dropdown menu and is navigated to the “Pending Peer Reviews” page (see Figure 5.6).
3. The User clicks the particular Peer Review being examined and is navigated to the “Accept or Reject Peer Reviews” page (see Figure 5.17).
4. The User selects the checkbox “Accept”.
5. The User clicks the “Submit” button.
6. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18).

Alternate Flow: Reject Peer Review

4. The User selects the checkbox “Reject.”
5. The User enters the reasoning for rejecting the request into the textbox.
6. The User clicks the “Submit” button.
7. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18).

3.1.6 - View Pending Processes

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to look at the Pending Processes.

Steps:

1. The User clicks the “Processes” dropdown menu.
2. The User selects “Pending” from the “Processes” menu and is navigated to the “Pending Processes” page (see Figure 5.8).

3.1.7 - View Processes on Hold

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to look at the Processes on Hold.

Steps:

1. The User clicks on the “Processes” dropdown menu.
2. The User selects “Hold” from the “Processes” menu and is navigated to the “Processes on Hold” page (see Figure 5.9).

3.1.8 - View Running Processes

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to look at the Running Processes.

Steps:

1. The User clicks on the “Processes” dropdown menu.
2. The User selects “Running” from the “Processes” menu and is navigated to the “Running Processes” page (see Figure 5.10).

3.1.9 - View Successful Processes

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to look at the Successful Processes.

Steps:

1. The User clicks on the “Processes” dropdown menu.
2. The User selects “Successful” from the “Processes” menu and is navigated to the “Successful Processes” page (see Figure 5.11).

3.1.10 - View Failed Processes

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is able to look at the Failed Processes.

Steps:

1. The User clicks on the “Processes” dropdown menu.
2. The User selects “Resubmit” from the Processes Failure Error Message (see Figure 5.13).
3. The User selects “Failed” from the “Processes” menu and is navigated to the “Failed Processes” page (see Figure 5.12).

3.1.11 - Restart Processes

Actors: General User, Admin User.

Preconditions: User is logged in. User is the “Failed Processes” page. There is at least one failed process to restart.

Postconditions: User has submit a request to restart a Failed Process.

Steps:

1. The User clicks on a failed process.
2. In the popup displayed, the User selects “Resubmit”.
3. The User is brought to the “Edit Database” page with all of the necessary parameters already filled out for the appropriate macro with the information from the prior execution (see Figure 5.15).
4. The User fixes any errors in the input parameters.
5. The User selects if they want this request to be peer reviewed.
6. The User adds any Tags they wish to associate with the process.
7. The User clicks the “Enter” button.
8. The User selects “Yes” from the “Are you sure?” Prompt (see Figure 5.18).

3.1.12 - Search Processes

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: Information about the specific process is output to User.

Steps:

1. The User clicks on the “Processes” dropdown menu.
2. The User selects “Search” from the “Processes” dropdown menu and is navigated to the “Search Processes” page (see Figure 5.14).
3. The User inputs the ID of the process they wish to search for into the search bar.
4. The User clicks the “Enter” button.

3.1.13 - Cancel Processes

Actors: General User, Admin User.

Preconditions: User is logged in. User is the “Pending Processes” page. There is at least one pending process to cancel.

Postconditions: User has cancelled a Pending Process.

Steps:

1. The User clicks on a desired Pending Process.
2. In the popup displayed, the User selects “Cancel process”.
3. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18).

3.1.14 - Login to Account

Actors: General User, Admin User.

Preconditions: User is not logged in.

Postconditions: The User is logged in to their account.

Steps:

1. The User enters their username and password in the appropriate fields (see Figure 5.1).
2. The User clicks the “Enter” button.

Exceptions:

1. If the username or password entered in Step 1 do not match any in the system, the User receives a popup notification in Step 3 stating that their information was incorrect. The User is then returned to the login screen.

3.1.15 - Logout of Account

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User is logged out and the system is no longer able to be accessed by the current session without logging in again.

Steps:

1. The User hovers the mouse over the User Accounts Icon.
2. The User selects the menu item “Log Out” (see Figure 5.3).
3. The User select “Yes” from the “Are you sure?” Prompt (see Figure 5.18).

3.1.16 - View Notifications

Actors: General User, Admin User.

Preconditions: User is logged in and has Notifications to view.

Postconditions: User is able to look at their current notifications.

Steps:

1. The User hovers the mouse over the User Accounts Icon.
2. The User selects the menu item “Notifications” (see Figure 5.15).

3.1.17 - View or Edit Settings

Actors: General User, Admin User.

Preconditions: User is logged in.

Postconditions: User has accessed and/or updated their individual settings.

Steps:

1. The User hovers the mouse over the User Accounts Icon.
2. The User selects the menu item “Settings” and is navigated to the “Settings” menu (see Figure 5.19).
3. The User checks/unchecks any settings that the User wishes to update.
 - Toggle Peer Review: An Admin User has a unique setting to globally require Peer Review for all General Users. By selecting this option, General Users would no longer be able to toggle off Peer Review when submitting requests to edit the External Liberty Mutual Database.
 - Change Environment: Any User can change the Environment that they are working in by selecting either “Dev” (Development), “Test”, “QA” (Quality Assurance), or “Prod” (Production).
4. The User clicks the “Save” button.

4 - Non-Functional Requirements

The Liberty Dash system must handle exceptions, adhere to the Liberty Mutual security policies, store local data using MySQL, the application must be accessible on common web platforms, and be hosted on capable hardware.

4.1 Exception Handling

The system must be able to gracefully handle exceptions classifying the cause of the failure. This failure message will be communicated via email to the group responsible for the running application. The email message will include the date and cause of the failure.

4.2 - Security

The system must integrate into existing Liberty Mutual authentication policies, which involve Active Directory. The system does not need to support data encryption, as all data is non-personal.

4.2.1 - Login

The system will support Microsoft Active Directory, providing an interface for Windows NT Login and user group/domain restriction. The Microsoft Azure Active Directory Authentication Library (ADAL) will be used to interface with existing Liberty Mutual Active Directory servers. The specific isolated ADAL use-case is “Authenticating a Server Application on Behalf of a User to Access a Remote Resource”. The required credentials will be a valid username and password from an authorized user of the system.

4.2.2 - Encryption

The system will not support encryption of any data located on external or internal data stores.

4.2.3 - Access Policy

The system will support access from the internet only through the Liberty Mutual Firewall. Local area network access is allowed.

4.3 - Database

This application will use MySQL Database as its internal database (Internal Liberty Dash Database) management system. The internal server-side database must be concurrent to handle collisions in cases where multiple users wish to execute the same macros at the same time. The Internal Liberty Dash Database will store a log of executed macros and any associated errors that occurred, this information will be associated by each user's Active Directory ID.

4.4 - Usability

This section will cover the requirements and accessibility of the application including browser support and technical background expertise requirements.

4.4.1 Platform

This application will be written in JavaScript (ReactJS), HTML, and CSS and will minimally support Chrome, Firefox, I.E, and Microsoft Edge.

4.4.2 - Ease of Use

Users must be familiar with Liberty Mutual's technical infrastructure to effectively query the External Liberty Mutual Database and use macros. The application will be intuitive such that the defined use cases will be easily learned and executed easily by new users.

4.5 - Hardware

Liberty Dash is a web application which must be run on capable hardware.

4.5.1 Host Server

This application must be hosted on server hardware that is capable of supporting the runtime load demands of clients. Minimally, a quad core server processor and 8gb RAM will be necessary.

5 - Interface Design

This section shows the wireframes for the Liberty Dash system layout.

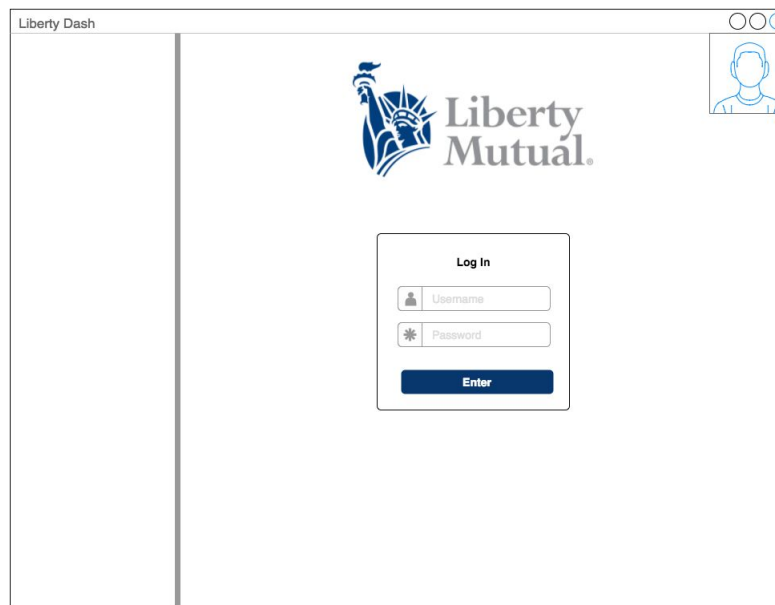


Figure 5.1: “Login” Page

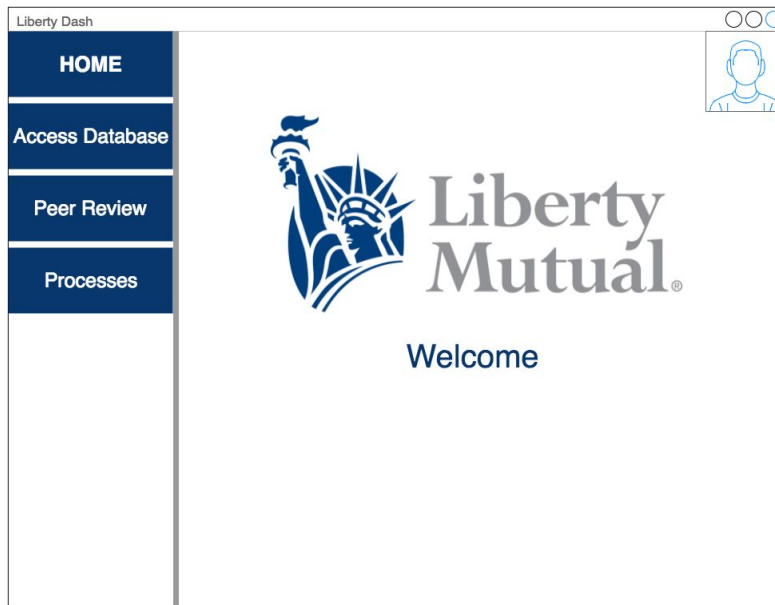


Figure 5.2: "Home" Page

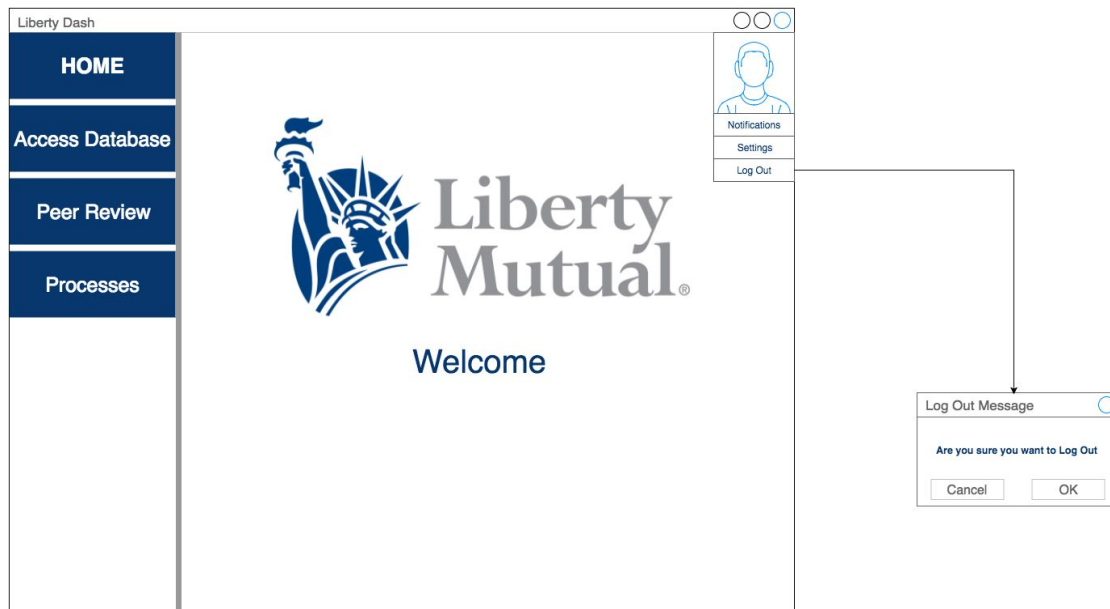


Figure 5.3: "Logout" Page

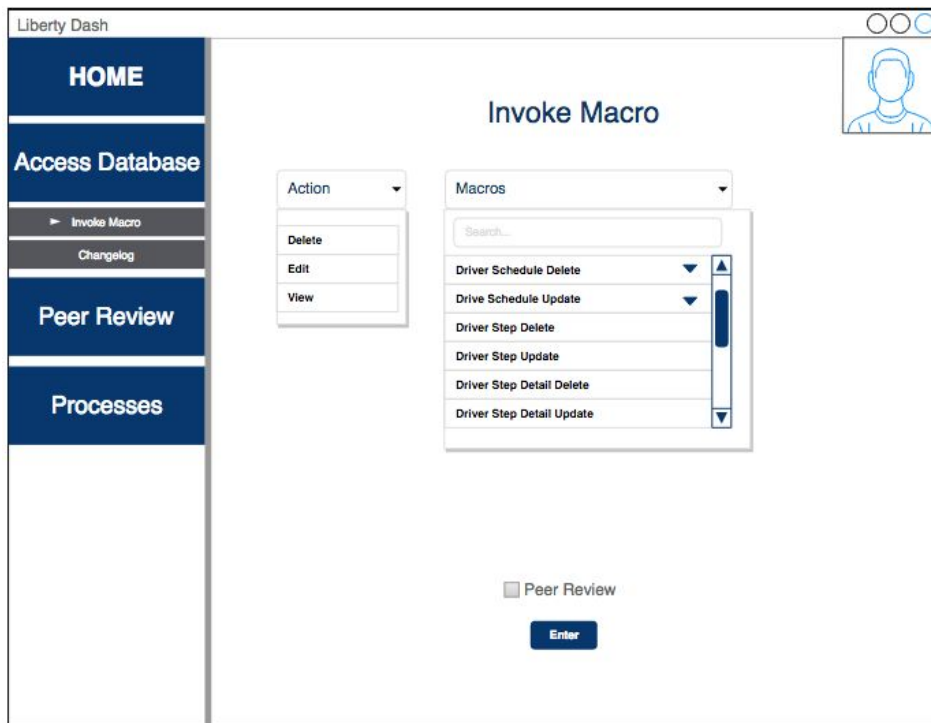


Figure 5.4: “Invoke Macro” Page

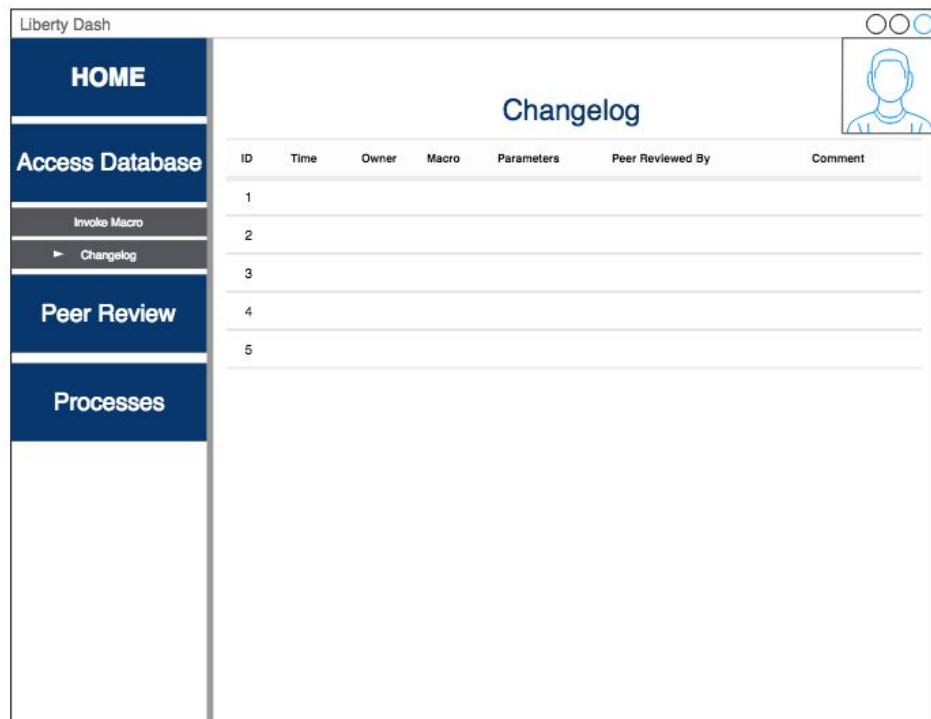


Figure 5.5: “Changelog” Page

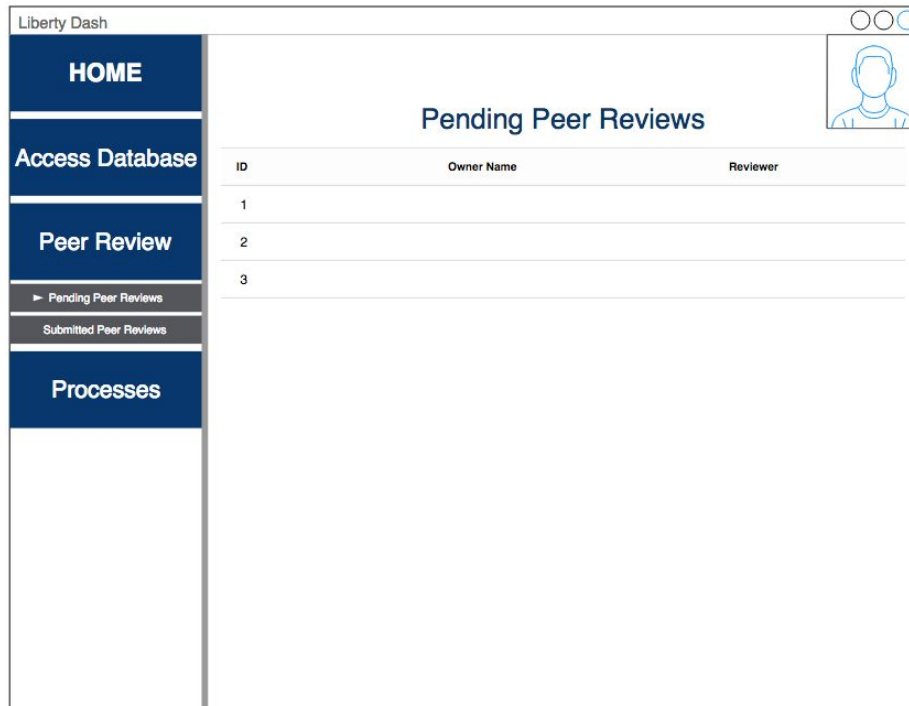


Figure 5.6: “Pending Peer Reviews” Page

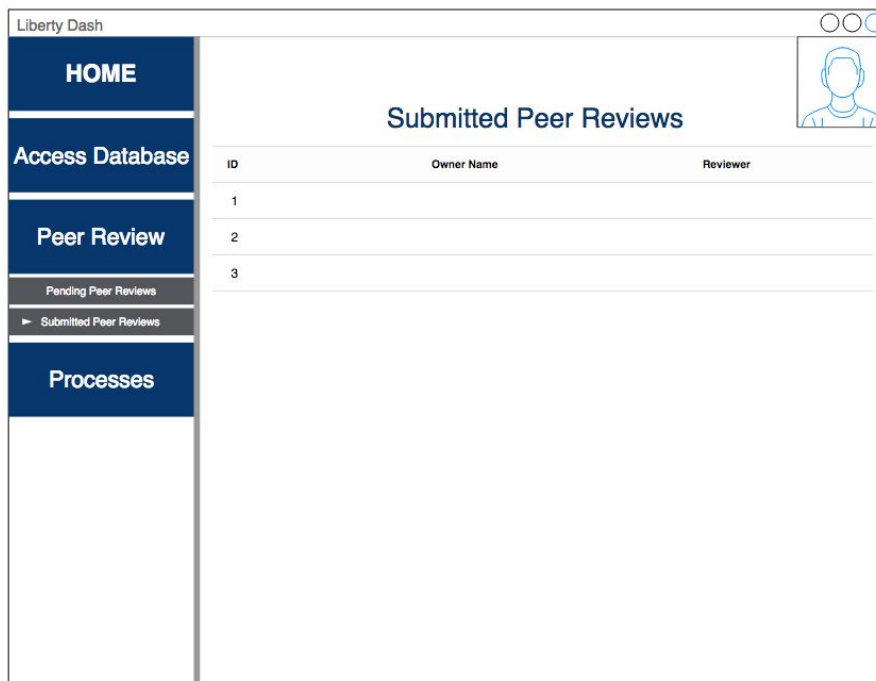


Figure 5.7: “Submitted Peer Reviews” Page

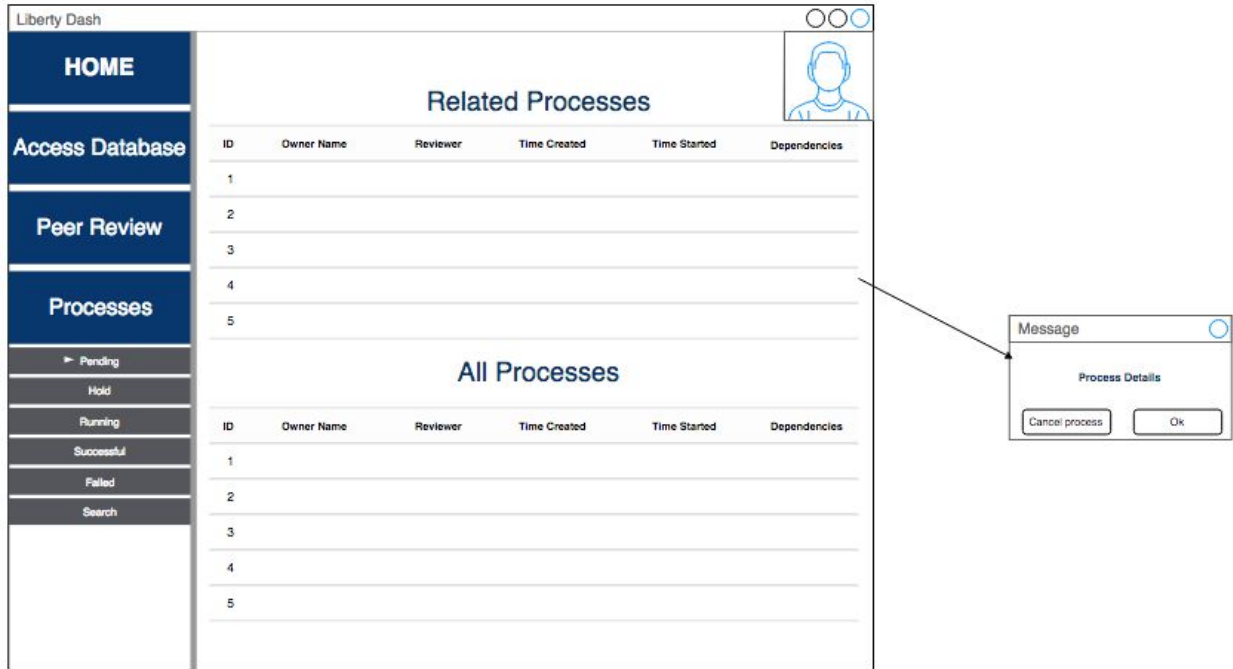


Figure 5.8: “Pending Processes” Page

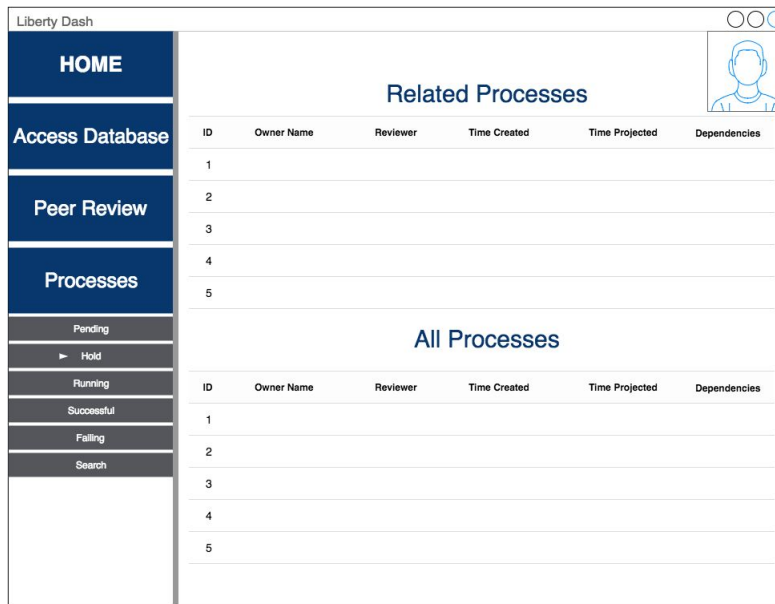


Figure 5.9: “Processes on Hold” Page

Liberty Dash

HOME

Access Database

Peer Review

Processes

Pending

Hold


▶ Running

Successful

Failing

Search

Related Processes



ID	Owner Name	Reviewer	Time Created	Time Started	Dependencies
1					
2					
3					
4					
5					

All Processes

ID	Owner Name	Reviewer	Time Created	Time Started	Dependencies
1					
2					
3					
4					
5					

Figure 5.10: “Running Processes” Page

Liberty Dash

HOME

Access Database

Peer Review

Processes

Pending

Hold


Running

► Successful

Failed

Search

Related Processes



ID	Owner Name	Reviewer	Time Created	Time Finished	Dependencies
1					
2					
3					
4					
5					

All Processes

ID	Owner Name	Reviewer	Time Created	Time Finished	Dependencies
1					
2					
3					
4					
5					

Figure 5.11: “Successful Processes” Page

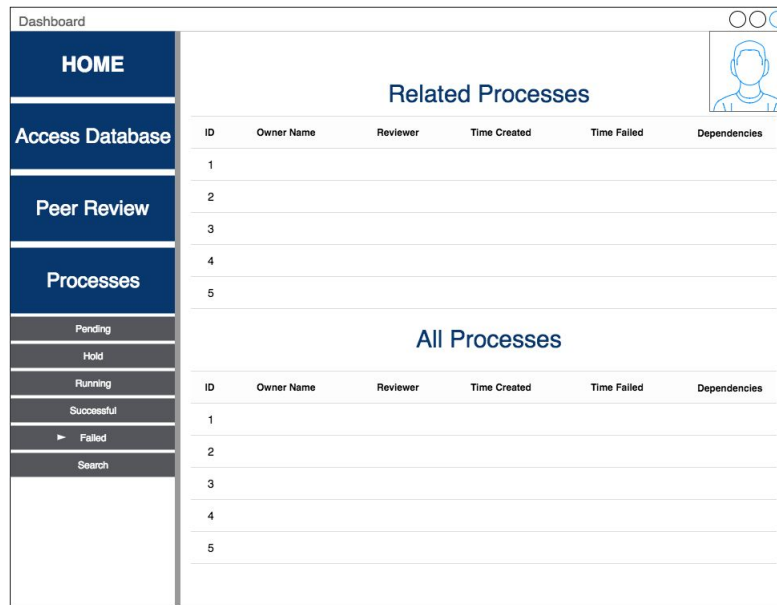


Figure 5.12: “Failed Processes” Page



Figure 5.13 - Processes Failure Error Message

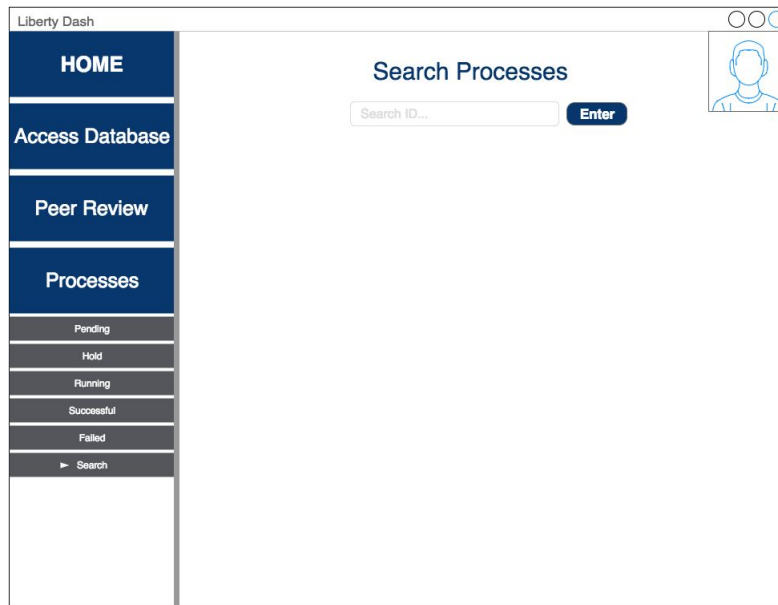


Figure 5.14: “Search Processes” Page

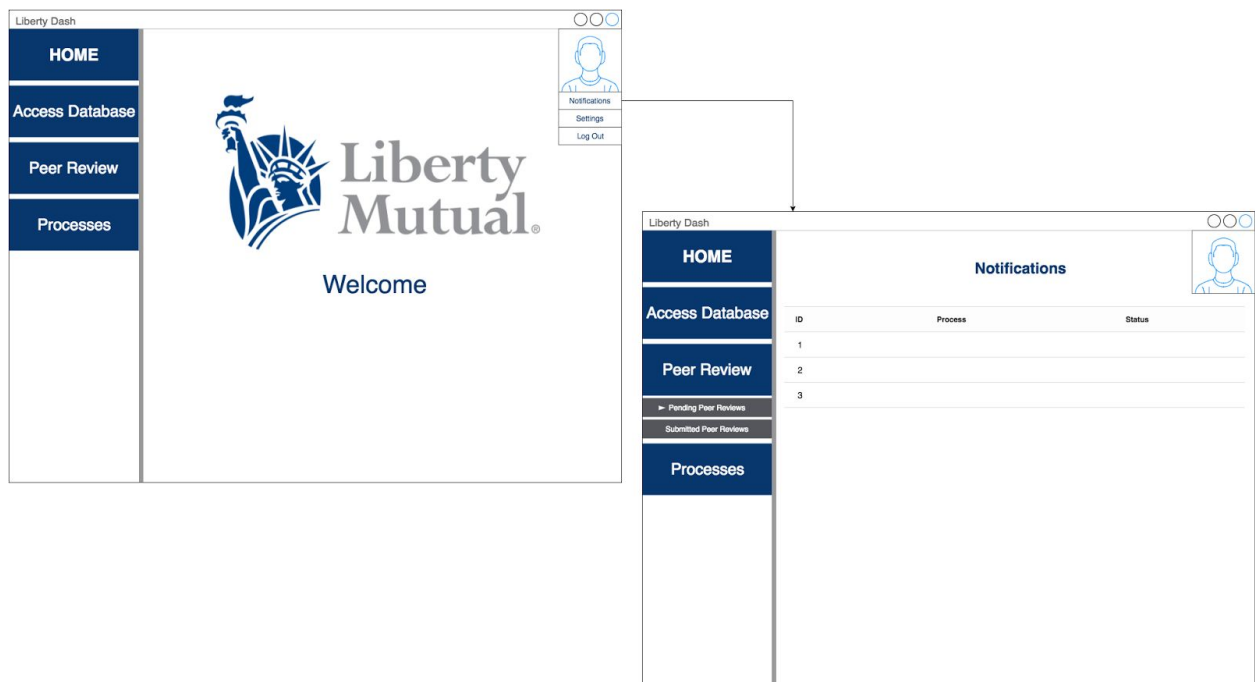


Figure 5.15: “ Notifications” Page

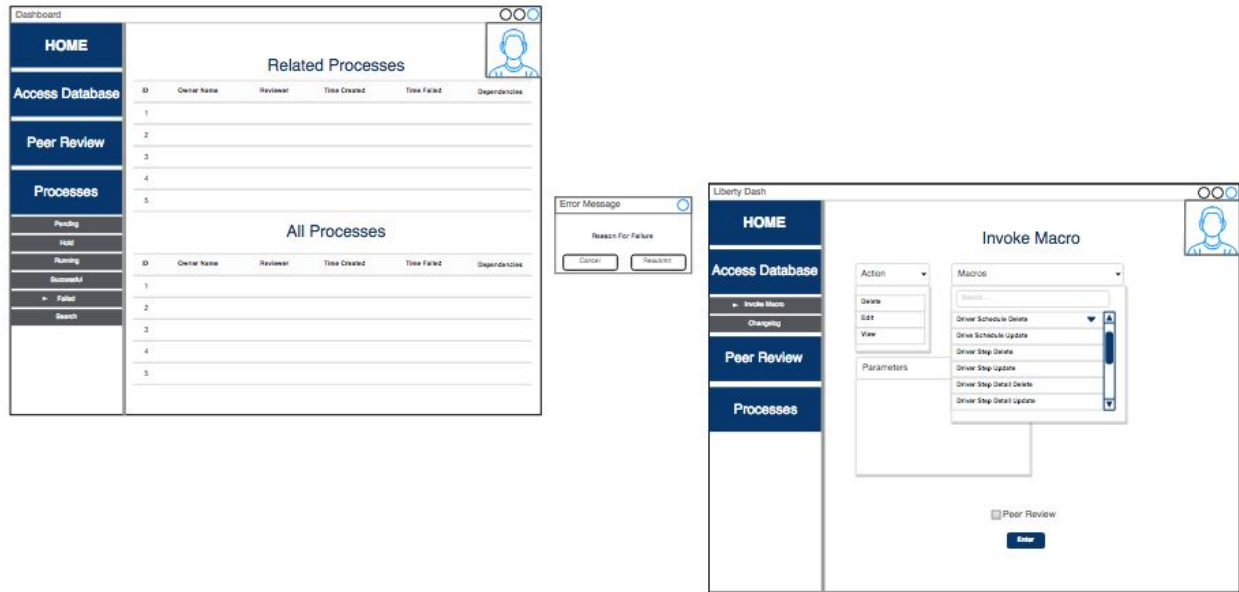


Figure 5.16: “Restart Processes” Page

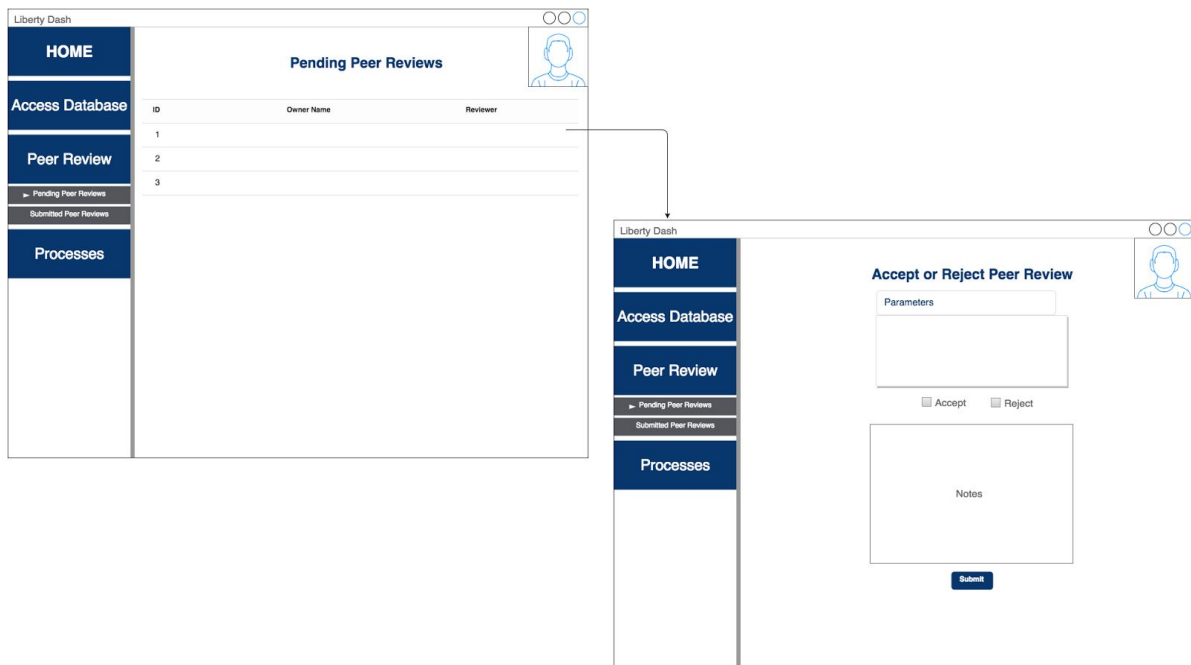


Figure 5.17: “Accept or Reject Pending Peer Reviews” Page



Figure 5.18: “Are you sure?” Prompt

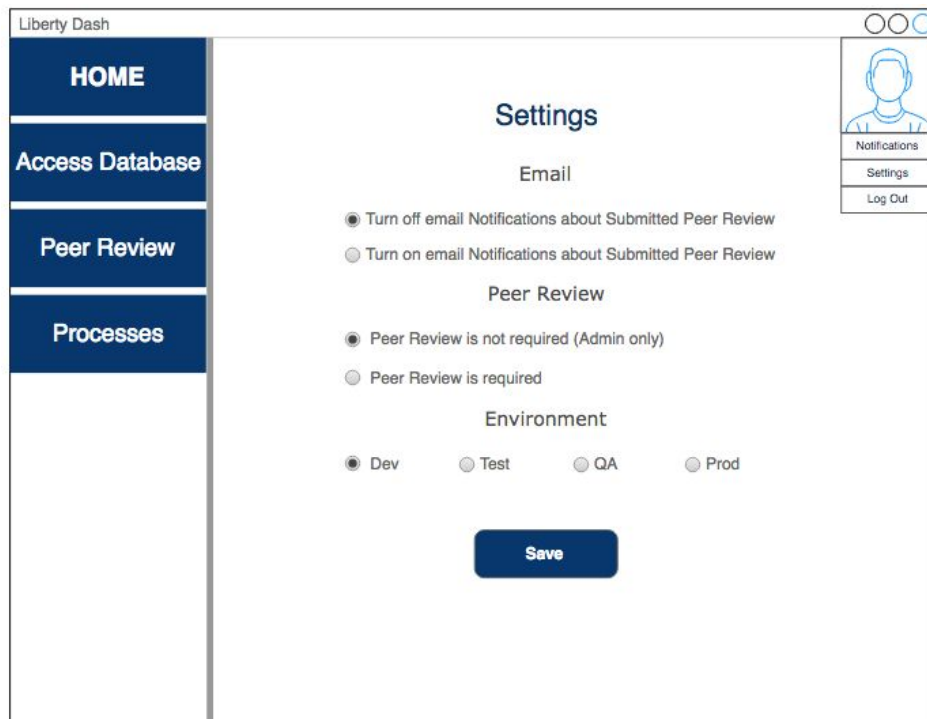


Figure 5.19: “Settings” Page

6 - Future Functionality

This section defines Liberty Dash functionality that will be implemented in the future.

6.1 - Performance Trends

The Internal Liberty Dash Database will store run logs of the system. From these logs the General User or the Admin User can pull historical runtime averages (separated into batches, jobs, and steps) and system failures to analyze the performance of the dashboard over time. This data can then be represented as graphs for the user to view and analyze.

6.2 - Service Level Agreements

A method of defining and enforcing service level agreements that schedule and run jobs at set trigger states will be added to a process editor, which will be added to the process view screen.

6.3 - Create New Macro

A functionality that allows a user to dynamically add new macros to the system that can be used to run processes.

7 - Glossary

Active Directory: A database that stores all of an organization's user login credentials.

Changelog: A file stored on the Internal Liberty Dash Database. This file documents the changes made through the system on the External Liberty Mutual Database. The Changelog includes information on: what was run (macro and parameters), who peer reviewed (if applicable), and a comment regarding why this macro was invoked.

External Liberty Mutual Database: The database from Liberty Mutual that holds Liberty Mutual's metadata. The Liberty Dash system will interact with this database, as it is external to our system.

Failed Process: A running process that was interrupted due to errors.

Internal Liberty Dash Database: The database internal to Liberty Dash System, that stores the Changelog table, Peer Review, and information on processes that have run and indication of whether the process completed or encountered errors.

Liberty Dash: The automated web system specified by this document.

Lock File: File created when a driver process begins to prevent multiple starts of the same process.

Notifications: Emails and/or messages sent to a General User in order to notify about a peer review, or status of a process (e.g., failed process).

MySQL Database: A relational database management system produced by Oracle.

Peer Review: The process by which submitted tasks are reviewed and either approved or rejected by other user in the system. By default for all macros Peer Review will be selected. However, if an administrator allows modifies the settings to allow developers to toggle it, when attempting to invoke a macro the developers will be unable to uncheck the “Peer Review” button to bypass this system in emergencies.

Pending Process: A process that has not yet reached its scheduled start time.

Process on Hold: A process that has been stopped from running at its scheduled time and can be released when ready.

Running Process: A process that is currently being executed.

Stand-Alone System: A system that is able to function independently of other systems.

Successful Process: A completed process that was executed without any errors.

Tag: A String of literals associated with data that allows a user to quickly locate a process or series of related processes.

User Account Icon: Icon on the right-hand top corner of the system’s User Interface. Gives the General User access to a menu of options such as Notifications and settings.

Windows NT Login: A Microsoft Windows personal computer operating system designed for users and businesses needing advanced capability.