**Library Management System**

**Iteration 3 Report**

CSCE 3513 – Software Engineering

Group 1:

Alexander Anderson

Cadence Gatewood

Ryan Morgan

Brandon Roden

Christopher Rogers

**Introduction**

This report accompanies the third release of the LMS (Library Management System). The goal of the LMS is to aid the employees and patrons of a library by easing the amount of effort which is required to accomplish their specific tasks. In this release we focused on implementing the admin user stories. We also implemented the book reservation system for patrons. The workstation GUI was updated to include numerous visual improvements.

In this report, the user requirements and stories that were targeted in this release are discussed. This discussion is then followed by an explanation of the software engineering practices which were used during the development of this third release of the LMS.

**User Stories**

The admin user stories from the LMS System Requirements Document were scheduled to be implemented in this release. These user stories are listed below, as well as the user story for book reservations.

**User Story 1.1:** Book reservations implemented.

**Discussion:** Functionality was added to allow patrons to see which books they have currently placed on reserve.

**User Story 3.1:** The administrator would like to add a new librarian account for a newly hired librarian. The administrator should supply the name of the new librarian and the LMS should create a librarian account under that name.

**Discussion:**  In this iteration of the LMS we have added functionality so that an admin can add a librarian account by supplying the LMS with the new librarian’s username, password, email, first name, and last name. The database will not allow the user to be added in the event that the name they provided is already in the database. If the passwords entered are blank, a message dialog will appear alerting the admin to the error, but the LMS will still run having queried the database with just a blank string. Also if the password and re-entered password do not match another message dialog will appear that alerts the admin to the error.

**User Story 3.1:** The administrator would like to remove the account of a librarian who no longer works at the library. The administrator should enter the name of the librarian and request the account be removed. After confirming that they wish to complete this action, the librarian’s account and data should be permanently removed from the LMS.

**Discussion:** In this iteration of the LMS we have added the functionality so that an admin can remove a librarian account by entering the username of a librarian. The database will find account with the entered username and remove the account from the database. If the user is not found then a message will be returned alerting the admin that the query failed. A message dialog will appear alerting the admin if either the username is blank and if the query was successful or not.

**User Story 3.3:** The administrator would like to configure the system to charge a certain amount of money per day for late fees/fines. Once this rate is set, the system should keep track of the amount of money owed for each late book for each patron.

**Discussion:** In this iteration of the LMS we have added the functionality so that an admin can change the rate at which patrons will be fined per day for each day that passes after the due date of each book that has been checked out. The admin is alerted as to the result of this action and if the entered information is invalid.

**User Story 3.5:** An administrator would like to change the password they use when logging-in to their workstation. The administrator will provide their username, old password, and new password. The LMS will check that the old password and username are a correct pair and proceed to update the administrator’s password if they are a correct pair. Otherwise the administrator will be notified that the password provided is not the correct password for the username.

**Discussion:** In this iteration of the project the administrator can change the password that they use. The administrator will enter their current password, the new password that they want to change to, and they will re-enter the new password. If the action is successful then a success message dialog will be returned. If it fails then a failure message dialog will be returned and displayed.

**Software Engineering Practices**

During the development of the LMS’ third release, the development team used software engineering practices which belong to the XP (Extreme Programming) model. There are six practices which comprise developing software with the XP model: Incremental Planning, Pair Programming, Small Releases, Simple Design, Refactoring, Continuous Integration.

**Incremental Planning**

This project utilized incremental planning in that the team created a list of user stories and the grader choose which stories to be implemented in each release.

**Small Releases**

This project utilized small releases in that the first release was developed to have the essential needs of the LMS. The subsequent releases were scheduled at two to three week increments and these small releases added functionality to the initial release.

**Pair Programming**

Several parts of the LMS were further developed using pair programming, including but not limited to: the Workstation, the Database Manager, and the Website. Between two or more programmers, depending on any particular meeting, advances in the code were made through direct interaction. The means to which this was accomplished was through the individuals involved with any instance of pair programming looking at each others code, making suggestions, and giving advice as to how to overcome identified problems. For all of the systems mentioned, two or more programmers came to a comprehensive knowledge of how the system that was being worked on functioned to a point where full dual responsibility was developed.

**Simple Design:**

The Simple Design principle specifies that the designs that are created should be sufficient in that it should meet the target requirements and nothing more. The development of the LMS used this guideline when giving priority of work assessments to individual sections that needed to be either developed or further touched up in the LMS. Parts of the LMS that were deemed more important were worked on first and for longer periods of time than parts that were deemed of a lower priority. In doing so, features that were more important, in the case of this iteration, administrative actions, were completed before aesthetic effects such as icons for the website and GUI were added. This allowed for programming resources to be focused on more important things like functionality over lesser things like creating a more pleasing appearance.

**Refactoring:**

When a bug or functionality flaw was found, the system where it was found was tested until the problem was isolated and then resolved. If it was not resolved then other programmers were consulted in order to come up with a pair-programming type solution. Response-queries, encryption, networking, and other miscellaneous files were combined into a new file named LMSCommon which was designed for the purpose of a very streamline integration between the files inside and other files that needed to use the contained files and projects. These contained files and projects were combined after a realization that they were accessed often in parallel and accessed frequently. Now with all of in the information in a central file, accessing the shared files has become much more streamlined, and easier to view in the form as a whole project as opposed to scattered files that are loosely connected

**Continuous Integration:**

In keeping with the principle of Continuous Integration, the development team successfully added a new component to the application. This component is a combination of several basic routines which are used throughout the application. This component was integrated into the existing application after testing that the operation of the component’s functionalities were maintained throughout the process of its addition.

**Time Spent:**

The time spent on the GUI was about 35 hours. The time spent developing the website was 35. The time spent adding admin queries/responses and debugging the workstation function 14 hours split between them.

**Test cases include using multilingual books**

**Test Cases:**

Several testing scenarios were used for validating the functionality of the Workstation GUI. In order to test the proper handling of uncommon characters, we used several books written in languages other than english. To test if the GUI was able to handle books without proper image urls, we tested with several ISBNs that google didn’t not have images. To test if the system was able to handle the creation of multiple users with the same name, we attempted to add multiple users with the same name. The system was able to successfully handle all of these tests.