Library Management System

Iteration 2 Report

CSCE 3513 – Software Engineering

Group 1:

Alexander Anderson

Cadence Gatewood

Ryan Morgan

Brandon Roden

Christopher Rogers

Introduction

This report accompanies the second 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 rest of the librarian user stories as well as refactoring portions of our existing code to facilitate future changes.

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 first release of the LMS.

User Stories

Four user stories from the LMS System Requirements Document were scheduled to be implemented in this release. These user stories are listed below.

User Story 1.1: The user shall now be able to search for a book based off of different fields.

User Story 1.2: A patron shall be able to navigate to the library website using a web browser. The patron then can log into the LMS website using their previously set username and password. the LMS then authenticates this login attempt by comparing it with the password entered with the password for the specified username. Under the “Patron Information” section, the patron can view any book currently check-out to them, the due dates for the books that are currently checked out under their account, and any late fines and fees changed to their account. At this point the patron cannot view the books that they currently have on reserve. If the user wants to change the settings to their accounts they can do so by going to the “Patron Settings” section. Once in this section they may change their password that is associated with their username, the email address that they wish to receive notifications to, and their preference on whether or not they want to receive email notifications. If the patron submits an invalid username and password combination, an error message will be sent back saying that the given information is incorrect.

User Story 1.3: A patron may get onto the library website using a web browser and from there the patron can navigate to the “Create an account” section. The patron can enter a username, password, and email address into the LMS. The LMS then will check the given information with the database to make sure that the username is not taken. The LMS does not yet send a verification email to the provided email address. If the username is already taken, then an error will be given and the patron will be asked to give a valid username.

User Story 2.4: When a patron would like to pay their outstanding fees they will pay a librarian who will input the amount paid to the LMS the LMS will subtract this number from their total balance.

Discussion: We have fully implemented this functionality; however the LMS assumes that payments are made in full and does not return the patron’s balance at this time. In later releases we plan to allow for partial payments.

User Story 2.5: The library has obtained more copies of a book and a librarian would like to add these copies to the database. The librarian will do this by entering an ISBN as well as the number of copies to add. The LMS will then add the specified book as well as its other information (author, title, etc..) to the database. At this time the LMS will not verify with the Librarian that the information found is valid before adding it to the database.

Discussion: The LMS currently will add a number of books specified by the author. The LMS will not allow the librarian to manually fill in the book information in the event that it was not automatically gathered at this time.

User Story 2.6: A librarian will be able to change their password if they so desire. They will have to provide the correct current password as well as a new password they would like to use. After completing this process the old password should no longer be valid and the LMS should only accept their new password.

Discussion: Our current system forces the user to enter their userID, their old password, and the new password all together. These three values are passed to the database together where the userID and old password will be checked to see that the librarian entered the proper information. If the userID and old password are a correct pair the LMS will then change the password associated with the userID to be the new password. If the userID and old password are not a correct pair the LMS will not change the password associated with the userID.

Software Engineering Practices

During the development of the LMS’ second 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

The user stories that were implemented were picked ahead of time. Several user stories were dropped once the requirements for the 2nd release were changed. For this release all stories involving Librarian actions were to be implemented with only a few exceptions. As exceptions were either reached or found, they were kept excluded or added in incrementally. For the Patron user stories all aspects of the stories were to be added except for sending out emails and having a review system for books. These were intentionally planned to be released with release 3.

Pair Programming

The Workstation, the Database Manager, and the Website all were worked on in pair programming. Progress was made both simultaneously working on code with intense interaction between two programmers, and also with one programmer working on a piece of code with a 2nd programmer looking at the same code making suggestions and advising the work. All 3 of these systems were designed so that at the very least 2 programmers were accountable for their functionality. The Website and the workstation in particular were developed with 3 programmers accountable for their code and functionality. A good majority of the code was developed with intensive pair programming.

Small Releases:

As parts of code were finished they were pushed onto GitHub and merged. Each working iteration of code was pulled when either the part that was changed or the entire system was executable and operational. These small releases were able to allow for progress assessment and for test cases to be created to make sure that user stories were correctly implemented as the code base grew.

Simple Design:

The Simple Design principle specifies that only the designs created should be enough to meet the target requirements and no more than this. The development used this principle when deciding which components should receive the most attention during development. This allowed the team to focus their effort into the components which are critical to the successful operation of the LMS.

Refactoring:

Whenever an error was found or detected, a test case was made for it. Once the error was located then the programmer tried to the best of their ability to find the source of the problem and resolve it. The GUI was refactored in how it looked and what it could search. It was made much more robust and its functionality was greatly enhanced. A response object was added and some of the functionality of the Query object was moved onto the response object. Authentication was fully made and integrated with the system as there had been no authentication before. The response object now has all of the information tied to Librarian accounts and displays what the GUI needs to know. And now all login requests, patron requests, librarian requests, and admin requests are now run through authentication to check to make sure that they exist in the database and rejects them if not, and checks if the user has the authority level required to pass the type of request they are passing.

Continuous Integration:

Once the Response Object was made then it was immediately implemented and tested to make sure that the information that it was supposed to pass and display was correctly carried to its destination. There were several specific functionalities that the Query object had that were carried over to the Response Object and they were immediately deleted from the Query object and tested inside of the Response Object to insure that they still worked as designed after being moved. Once the integrity of the entire program was ensured, then the Response Object was considered integrated into the system. When authentication was finished, it was combined with the network and query and response query objects to make sure that login, patron, librarian, and admin queries were all correctly verified. Authentication also generated a session ID with a session timeout value that once tested was implemented throughout the LMS as needed.

Time Spent:

The time spent on the GUI was about 24 hours. The time spent developing the website was 30. The time spent refactoring the response and the query objects was around 12 hours. The time spent developing and integrating authentication was about 8 hours.

Test cases:

Test cases were developed for authentication. However the junit that was written failed whenever a user was stored in the database, and then the database was immediately read from after the write. If the junit was run twice then it would succeed because the SQL exception was thrown and then only the read of the database was executed. This problem was attempted to be fixed by using the sleep() method but even at 1000 milliseconds the same errors occurred. Unfortunately the result was that the tests for authentication were commented out and rendered void. The AESCipher in Encryption has 3 unique tests. The first is just a short string with a capital, lower case letters, and a punctation. The next test case is much more complex in that it has a mix of mathematical symbols, arabic, box symbols, and an entire paragraph from “A Christmas Story” (which is not copyrighted and needs no citation) that are all combined into a single string. The last test case is a blank string, which had previously given many errors while in use in the LMS. All 3 test cases have no errors. Then for RSACipher these same 3 test cases were made. For the short-simple string, and the long-complex string, the RSACipher did not have errors when run as junit. However on the 3rd case when a blank string was given, an error was detected. This will be fixed at a later time as a blank string should not and will not ever be passed to RSACipher.