**Library Management System**

**Iteration 1 Report**

CSCE 3513 – Software Engineering

Group 1:

Alexander Anderson

Cadence Gatewood

Ryan Morgan

Brandon Roden

Christopher Rogers

**Introduction**

This report accompanies the first 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 initial release, three of the most critical user requirements have been implemented to ensure that a solid foundation is created for the future needs of the LMS and its users.

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:** A patron shall be able to see if the library is in possession of a particular book. If the book is in the library’s collection the LMS should direct the patron to a webpage with information about the book. The information returned to the patron should consist of the title, author, ISBN, as well as “likes” and “dislikes”. If the book is checked out the patron should be able to place themselves on a waiting list to reserve the book.

**Discussion:** Our first iteration allows patrons to retrieve all of the specified information except for “likes” and “dislikes”. We have not yet implemented a user account system so there are not any “likes” or “dislikes” attached to books. Patrons cannot yet place themselves on a waiting list to check out the book for the same reason.

**User Story 2.1:** A librarian shall be able to look up information regarding a specific book using its ISBN. The librarian should be shown the book’s title, author, status and shelf-location. If the book is not in the library’s collection the LMS will notify the librarian.

**Discussion:** The ordering of the books on the library’s shelves is currently unknown. As a result, librarians are given all of the required information except for the books shelf location.

**User Story 2.2:** A librarian shall be able to check out a book to a patron using the LMS workstation. The LMS should let the librarian know if the book is available for checkout, and if it is the librarian should be able to update the LMS’s database so that the book is marked as checked out.

**Discussion:** Librarians can request that a book be marked as “checkout”; however, a book cannot be checked out to a specific patron in this release of the LMS. This is due to the fact that patron accounts will not be implemented until the next release of the LMS.

**User Story 2.3:** A librarian shall be able to check in a book being returned by a patron, using the LMS workstation. The LMS should properly update the book’s availability status in the LMS’s database.

**Discussion:** This user story is fully implemented in this release of the LMS.

**Software Engineering Practices**

During the development of the LMS’ first 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 Incremental Planning principle specifies that requirements are recorded as stories and the stories included in a release are determined by the customer based on time available and their relative priority. This principle is an inherent part of our development process due to the assignment specifications. We first typed a user requirements document that listed all of our requirements in the form of user stories. Then we met with Jean Pierre Habimana to determine what user stories to implement first. The selection was based on which user stories he determined to be the most important to our project and what was feasible in our time frame. In this way Jean Pierre acted as our customer.

**Pair Programming**

After all of the individual components were developed to a point of minimum functionality, pair programming was used specifically in the process of putting a cohesive unit together that had a proper and fluid communication between each separate component. The members of the team worked together looking at each others’ code in order to understand how to integrate the components. This allowed for a more thorough debugging of both the individual components and the entire system as the LMS was pieced together. This simultaneously gave the development team collective ownership of both the functionality and reliability of each specific component as well as ownership of the collaborative final product.

**Small Releases:**

For the initial release of the LMS, the development team focused their efforts in delivering a core set of key functionalities which allow the LMS to operate at a basic level. The basic functions needed to operate a library consist of determining the availability of a book and indicating if a given book is currently on the library’s shelves. Therefore, these three functionalities were chosen to be implemented in the first release of the LMS.

**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:**

The Refactoring principle specifies that all developers are expected to refactor their code as soon as improvements are found. Refactoring was not utilized in this release as this is the initial release of the LMS. Since refactoring is primarily used to improve upon an existing code base, our first release focused on creating the code base that may be refactored in future releases.

**Continuous Integration:**

The Continuous Integration principle specifies that as soon as a task is complete it is integrated into the whole system. To utilize this principle, the development team used a version control tool (Git) to work on components and submit them to the central repository as they were completed.

**Testing:**

The development team created a system test case to verify that the LMS’ server can correctly handle a request sent by the workstations. In addition to this system test case, several informal component tests were written to ensure that specific components of the LMS’ server and workstation operate as expected.