Introduction to Programming II
Fall 2018 – Course Project
A Library Management System

This document describes the course project for Introduction to Programming II: a Library Management System. The project is divided into 4 deliveries. You need to present in each delivery the implementation, testing and documentation of the corresponding system. It will be developed in groups of 3 or 4 students. Members of a group must belong to the same lab group (e.g. all members belong to BS-3). This document is an initial specification of the system, your TA is the client of the application. Should you have any question about the requirements, ask your client.

Your group will need to make a presentation (demo) of each delivery. Make sure every member of the group works on the project. Questions will be asked to specific members of the group (e.g. randomly), and the grade will be for the whole group. The project needs to be hosted in an online version control system. You need to commit your changes regularly (this will be taken into account for the final grade). You need to add your TA as a member of the project (it could be with read-only permission).

1 Overview

Your task is to implement a Library Management System (LMS). LMSs are used in libraries to track the different items in the library. The system contains all information about books, magazines, audio/video materials, as well as people allowed to check out the materials or those in charge of the management. Your team will implement a LMS for the Innopolis library, LMS-INNO. Your first task is to define a better name for the application.

LMS-INNO will enable users to search for documents, check out or enter new materials, manage users of the library, among other functionalities.

2 Data (entities)

2.1 Users

There are two types of users (your implementation should be flexible enough to allow the extension to more types of users):

Patron. Library patrons can search for, check out and return documents. Documents can be checked out for certain time periods. There are two sub-types of patrons: Faculty, representing all the teaching body (e.g. professors, instructors, TAs) and Students of the university. A student cannot be a faculty member.

Librarians. Librarians can check overdue documents, manage patrons, and add/delete/modify documents.

2.2 Documents

The main asset of the library are documents. Your application should contain the following type of documents, although, your solution should be flexible enough to allow the extension to more types:

Books: Books are written by one or more authors and published by a publisher. Books have a title and may exist in different editions – each published in a certain year. For example, "Introduction to Algorithms" is a book written by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. It was published by the MIT Press. The third edition was published in 2009.

Journal Articles: Journal articles are written by one or more authors, have a title, and are published in a certain journal. Journals have a title and are published by a publisher in issues. Issues have editors and a publication date. For example, "Communication ACM" is a journal. The article "Go to Statement Considered Harmful" written by Edsger W. Dijkstra was published in the "March 1968" issue of this journal and Edward Nash Yourdon was the editor of the issue.

Audio/Video (AV) materials: AV materials have a title and the list of authors.

Documents also have the price value (in Rubles) – used when a patron gets a fine, and a list of keywords (e.g. “Programming Languages”, “course material”) – used when a patron is searching for documents.

2.3 Copies

A library may have several copies of each document. Copies are stored in a certain place inside the library, e.g., a room, level. For each copy we need to know whether it is currently checked out and by whom. Patrons are only allowed to check out documents for a certain time (e.g., 2 weeks).

3 Requirements

1. Each user of the library (librarians and patrons) has one unique card for as long as they are in the system.

2. The library needs to know at least the name, address, phone number, and library card number for each person of the system.

3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron in the system has checked out, when they are due, and any outstanding overdue fines.

4. A patron can check out copies of documents. The system needs to make sure that it is not possible to check out a copy of a document if all copies of this document are currently checked out by other patrons.

5. Books are checked out for three weeks, unless:

    - they are current best sellers, in which case the limit is two weeks or
    - they are checked out by a faculty member, in which case the limit is 4 weeks (regardless the book is best seller)

6. AV materials and journals may be checked out for two weeks.

7. The overdue fine is a hundred rubles per item per day, but cannot be higher than the value of the overdue item.

8. The library also has `reference` books and magazines, which cannot be checked out.

9. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the item needs to be returned immediately.

10. Users of the library (librarians and patrons) can search for documents. The application should support different search criteria (e.g., by author, by title) and combinations of these search strategies. For example, return all books that are written by Alonzo Church (author) for which there are copies available for checking out. Both complete matches and partial matches should be supported. The results of a search should be shown as a list of documents to the user. Users should be able to access detailed information for each document or refine/modify his search.

11. Librarians can add/delete/modify documents and patrons to the library system. For example, a librarian can add a new copy for a book to the library and record where it is kept within the library.

4 Deliveries

Components to be implemented

- entities: Users (Cards) and Documents
  - and the corresponding interface for features `add`, `delete` and `modify`.
- the Booking system
- the Return system (including Fine and Renew subsystems)

These are the minimum requirements for each delivery:

D1: i) Implementation of Users–Cards and Documents. Only the representation in a programming language. ii) Implementation of the Booking system (interface). – Deadline: Monday February 5th, via Moodle.

D2: i) Implementation of the Return system (interface). ii) Implementation of the storage system for Users-Cards and Documents (see comment below). iii) Implementation of features `add`, `delete` and `modify` for Users–Cards and Documents. – Deadline: Monday March 5^{th}, via Moodle.

D3: Implementation of subsystems Fine and Renew (interface). – Deadline: Monday April 2^{nd}, via Moodle.

D4: i) Implementation of the Search feature. ii) The whole system working together. – Deadline: Monday April 23^{rd}, via Moodle.

You are free to define how to store the information about Users (Cards) and Documents, e.g. a Data Base, a JSON file. You are also free to define the interface of the components, e.g. GUI, Web, Console.

5 Final remarks

Each delivery should contain the corresponding implementation of the system, a test suite and the corresponding code documentation (e.g. in the form of contracts). You need to make a presentation (demo) of the delivery during the lab session on the Wednesday right after each deadline (e.g. presentation for D1 will be on Wednesday February 7^th). During the presentation (demo)

- there will be a Question/Answer session. Questions will be asked to a specific member of the team, however his/her answer will be taken into account for the whole group. Make sure every member of the group works on the project.
- Right after each delivery deadline and before the presentation, you will be asked to implement some additional test cases. During the presentation you will need to show those test cases (test cases should run on the code you submitted for the delivery deadline).