Enhanced Library Management System
Analysis and Design Document

Student: Boros Hanniel
Group: 30432

Revision History

Date	Version	Description	Author
04/04/2018	1.1	Writing Elaboration – Iteration 1	Boros Hanniel
25/04/2018	1.2	Writing – Iteration 1 all	Boros Hanniel

Table of Contents

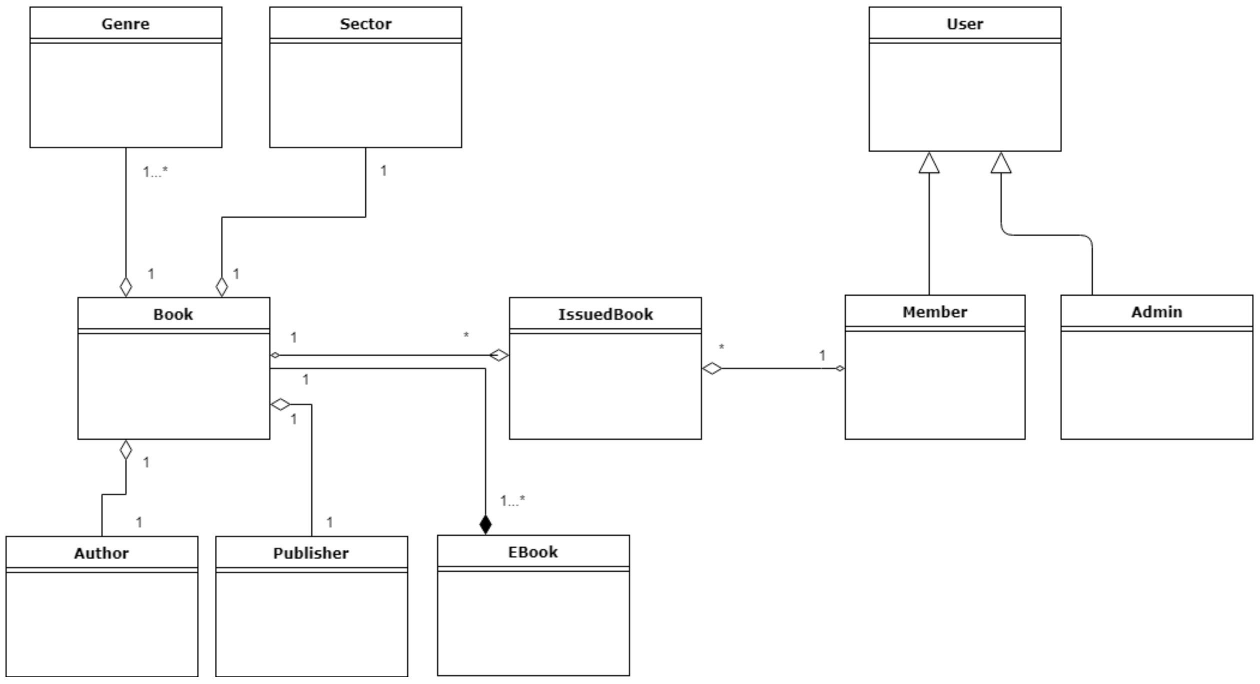
I. Project Specification

This software project is a library management software system with all the basic as well as some innovative features for managing a library). It consists of a large database of various books available in the library. It also lists various books issued to respective readers. The system keeps track of all the books readily available and also the books that have been issued to various readers the time period for which the books have been issued. The system also handles ebooks/pdf database. If the reader needs an ebook/pdf, the ebook can be directly mailed to the client email through the system through a single click. Readers usually tend to forget the date to resubmit their library books so for these books that have been issued to the reader and have reached their expiry date, an email notification is sent to the reader automatically by the system to remind him about the book return date. Thus this innovative library management system provides an enhanced library functionality for this modern day world.

II. Elaboration – Iteration 1.1

1. Domain Model

The Domain Model showing the main classes and the relationships between them.



2. Architectural Design

2.1 Conceptual Architecture

The architectural patterns used for this project is the Layers architecture pattern (otherwise known as the n-tier architecture pattern), together with the MVC and Client-Server patterns. These architectural patterns helps us to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.
For this project I consider that the combination of these patterns leads to the most convenient architectural design. In the followings I will present shortly each of them applied to the project.
Layers Architectural Pattern
Controller layer
This layer contains the user oriented functionality responsible for managing user interaction with the

system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.

Business layer

This layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use.

Persistence layer

This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

MVC Architectural Pattern

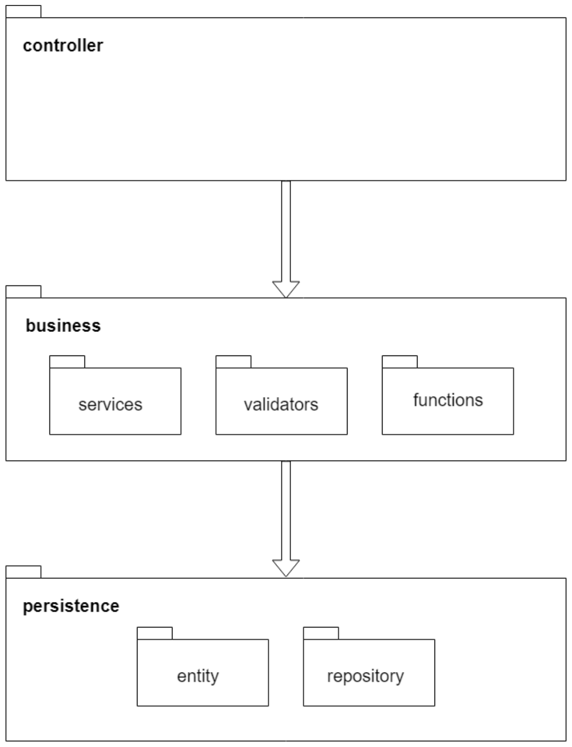
The major (model, controller, view) are decoupled, allowing for efficient code reuse and parallel development. Model-View-Controller patterns have a wide usage and forms. In this project I used Spring MVC which provides a good front-to-back-end data model mapping to UI and back to services, invoking actions.

Client-Server Architectural Pattern

The client/server architectural style describes distributed systems that involve a separate client and server system, and a connecting network. The simplest form of client/server system involves a server application that is accessed directly by multiple clients, referred to as a 2-Tier architectural style. The current assignment was developed in Spring Framework which has a Client-Server design structure handling requests from clients (represented by the html files) in the controller, which then calls the services to perform the required actions.

2.2 Package Design

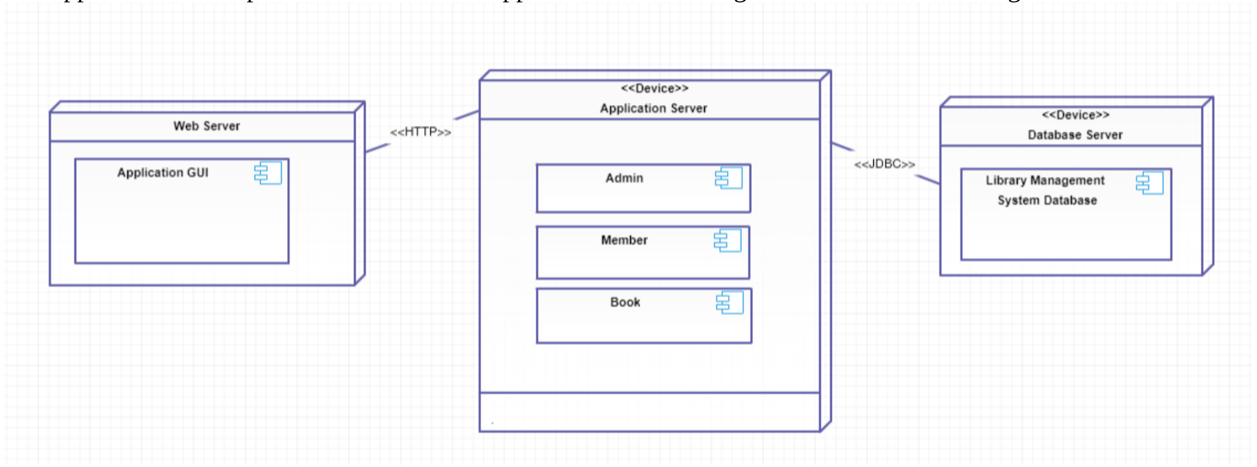
Below I present the package diagram. As it can be seen on the package diagram as well, the layered design is respected. This is just a temporary version of the package diagram, as with the implementation some changes may be arise.



2.3 Component and Deployment Diagrams

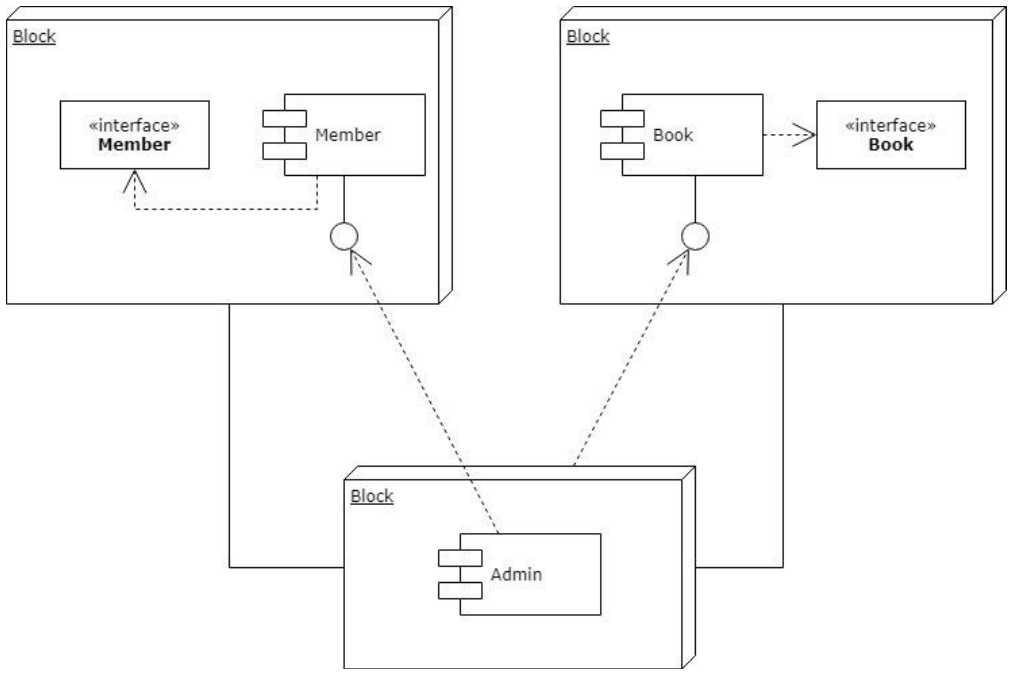
Deployment diagram

The application is composed of two tiers: the application business logic and the database storage.



Component Diagram

There are three main components of the application: the administrator, the member and the book. The administrator can operate on library members and books as well.



Block
«interface» Member
Member
Block
Book
«interface» Book
Block
Admin

III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

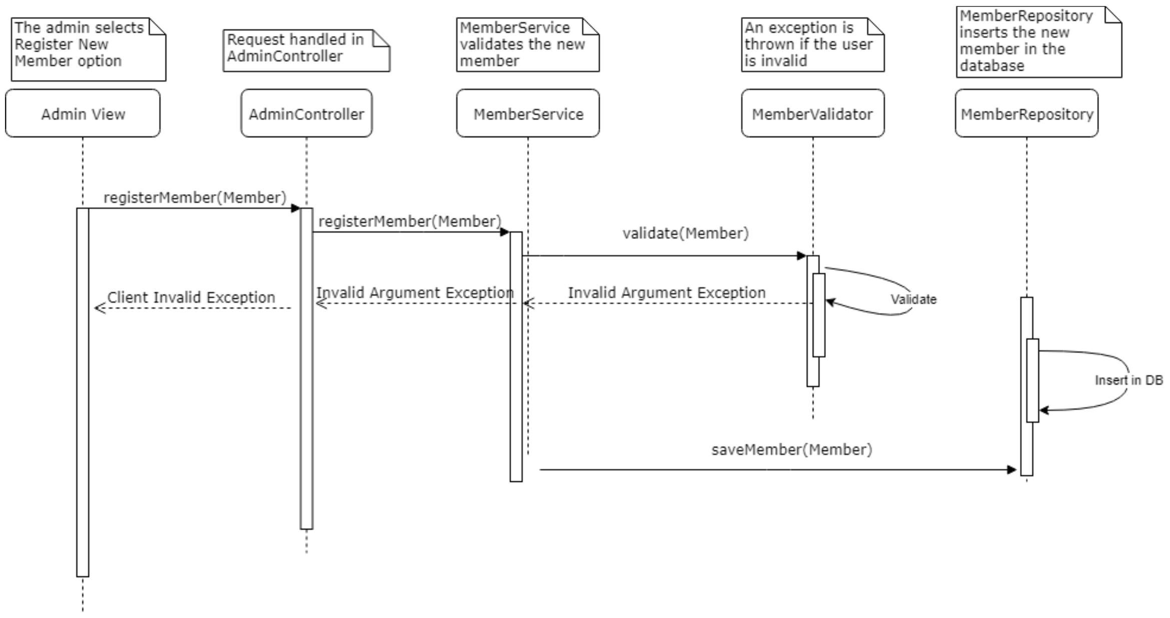
Interaction Diagrams

The interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration (communication) diagram.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

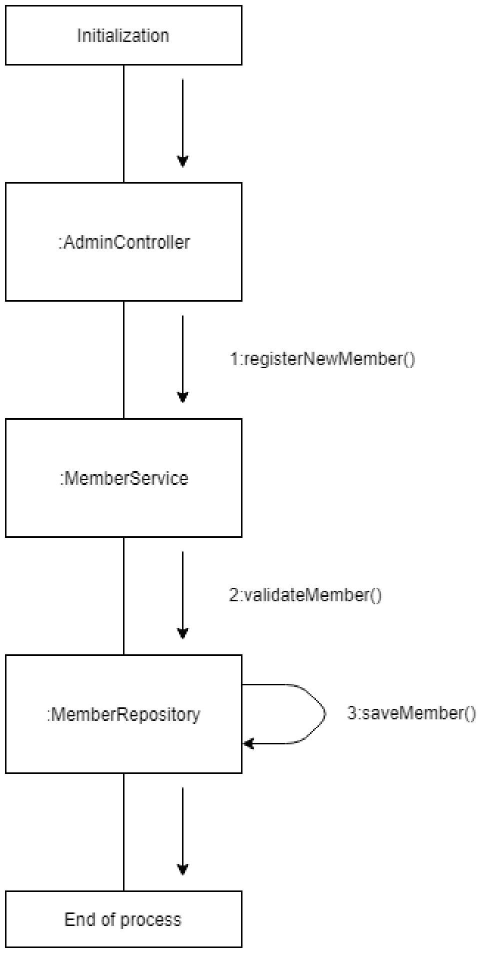
Sequence diagram

Scenario: Registering a new member to the library member


The admin selects Register New Member option
Request handled in AdminController
MemberService validates the new member
An exception is thrown if the user is invalid
MemberRepository inserts the new member in the database
Admin View
AdminController
MemberService
MemberValidator
MemberRepository
registerMember(Member)
registerMember(Member)
validate(Member)
Client Invalid Exception
Invalid Argument Exception
Invalid Argument Exception
Validate
Insert in DB
saveMember(Member)

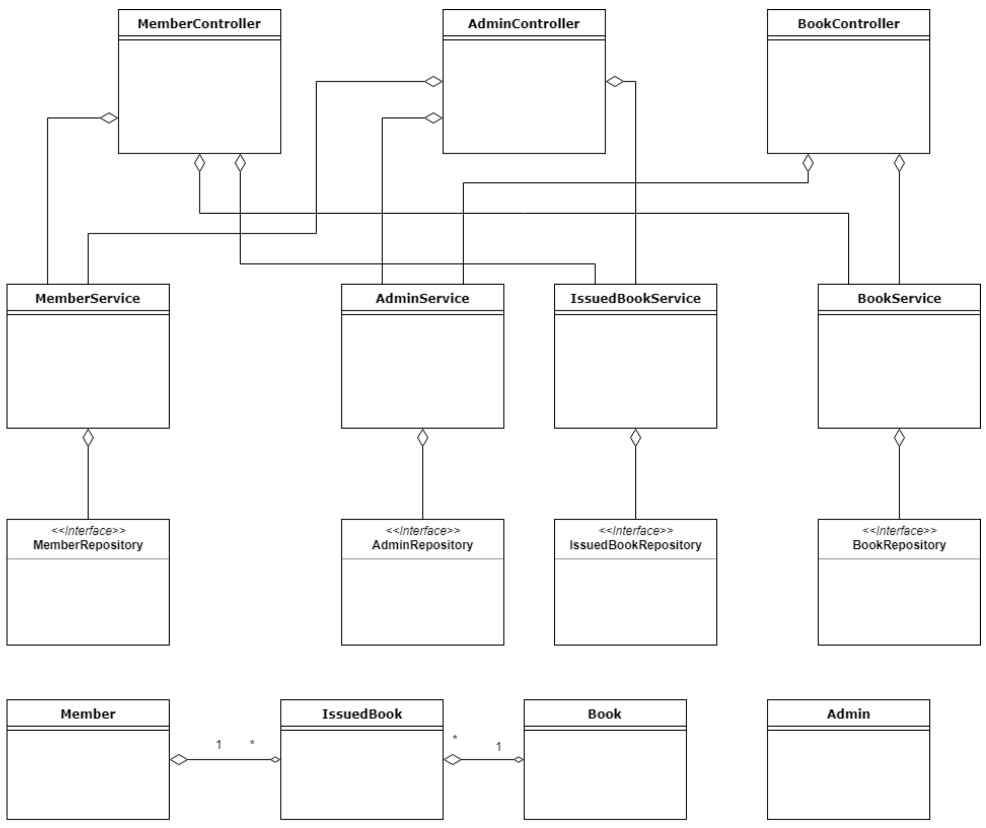
Collaboration diagram

Scenario: Registering a new member to the library

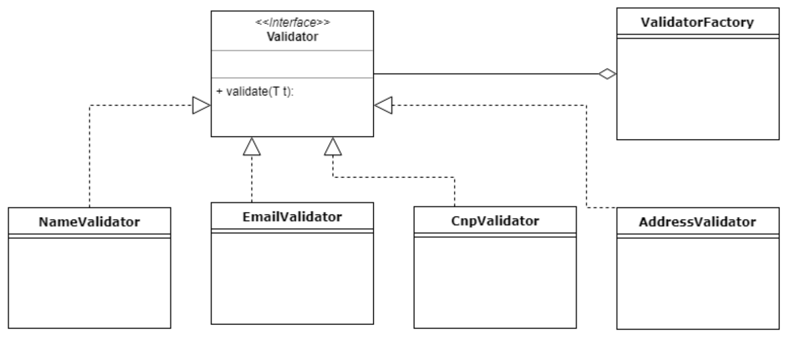


1.2 Class Design

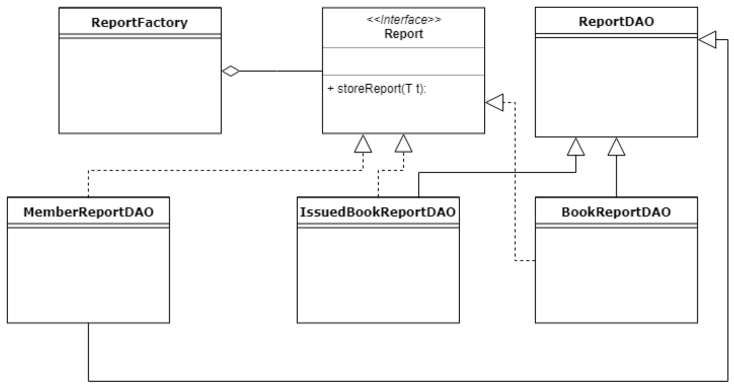
Below I present the UML class diagram. It is not the final, as changes may arise. Important to mention that here are not all the entities represented, just the ones I considered that have a more important meaning for the understanding of the project. Other entities (mostly those which model the characteristics of a book such as Author, Publisher, Sector, etc) are not represented here, as the diagram would be too complex. However they are modeled in a way which is easy to understand and follows common sense. Being Book characteristics, they are composed by the Book entity (composition).



MemberController
AdminController
BookController
MemberService
AdminService
IssuedBookService
BookService
<<Interface>>
MemberRepository
<<Interface>>
AdminRepository
<<Interface>>
IssuedBookRepository
<<Interface>>
BookRepository
Member
1
*
IssuedBook
*
1
Book
Admin



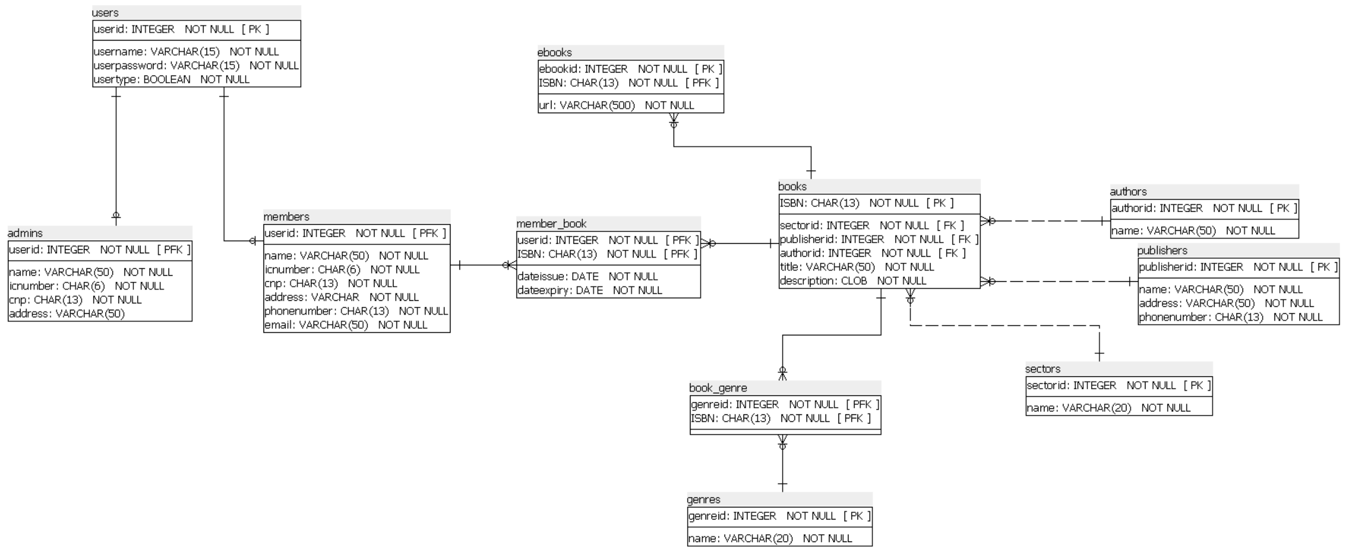
<<Interface>>
Validator
+ validate(T t):
ValidatorFactory
NameValidator
EmailValidator
CnpValidator
AddressValidator



ReportFactory
<<Interface>>
Report
+ storeReport(T t):
ReportDAO
MemberReportDAO
IssuedBookReportDAO
BookReportDAO

2. Data Model

Below I present the Data Model IR diagram. This was used to represent and model the data of the application. There are eleven (11) tables: nine (9) of them for direct data storage (users, admins, members, books, authors, publishers, sectors, genres, ebooks) and the other two (2) for relationships (they model a many-to-many relationship by a one-to-many ⇔ many-to-one relationship: member_book and book_genre).


users
userid: INTEGER NOT NULL [PK]
username: VARCHAR(15) NOT NULL
userpassword: VARCHAR(15) NOT NULL
usertype: BOOLEAN NOT NULL
ebooks
ebookid: INTEGER NOT NULL [PK]
ISBN: CHAR(13) NOT NULL [PFK]
url: VARCHAR(500) NOT NULL
books
ISBN: CHAR(13) NOT NULL [PK]
sectorid: INTEGER NOT NULL [FK]
publisherid: INTEGER NOT NULL [FK]
authorid: INTEGER NOT NULL [FK]
title: VARCHAR(50) NOT NULL
description: CLOB NOT NULL
authors
authorid: INTEGER NOT NULL [PK]
name: VARCHAR(50) NOT NULL
publishers
publisherid: INTEGER NOT NULL [PK]
name: VARCHAR(50) NOT NULL
address: VARCHAR(50) NOT NULL
phonenumber: CHAR(13) NOT NULL
admins
userid: INTEGER NOT NULL [PFK]
name: VARCHAR(50) NOT NULL
icnumber: CHAR(6) NOT NULL
cnp: CHAR(13) NOT NULL
address: VARCHAR(50)
members
userid: INTEGER NOT NULL [PFK]
name: VARCHAR(50) NOT NULL
icnumber: CHAR(6) NOT NULL
cnp: CHAR(13) NOT NULL
address: VARCHAR NOT NULL
phonenumber: CHAR(13) NOT NULL
email: VARCHAR(50) NOT NULL
member_book
userid: INTEGER NOT NULL [PFK]
ISBN: CHAR(13) NOT NULL [PFK]
dateissue: DATE NOT NULL
dateexpiry: DATE NOT NULL
sectors
sectorid: INTEGER NOT NULL [PK]
name: VARCHAR(20) NOT NULL
book_genre
genreid: INTEGER NOT NULL [PFK]
ISBN: CHAR(13) NOT NULL [PFK]
genres
genreid: INTEGER NOT NULL [PK]
name: VARCHAR(20) NOT NULL

3. Unit Testing

Unit testing was used in order to test the application.
Implementation is made to be favorable for and sustain testing.
Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behavior of the object one wants to test, it replaces the other objects by mocks that simulate the behavior of the real objects. This is useful if the real objects are impractical to incorporate into the unit test.
In short, mocking is creating objects that simulate the behavior of real objects.
Example of mocking test if a member was correctly inserted in the database:

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]

2. Design Model Refinement

[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]

V. Construction and Transition

1. System Testing

[Describe how you applied integration testing and present the associated test case scenarios.]
Implementation needed.

2. Future improvements

There are several possible future improvements. Considering that nowadays everyone is attached to mobile phones, the notification system could be done via SMS-sending, instead (or with) email notification.
As well, there could be implemented a messaging service, in order to make communication more easier between the library members and the library administration. These are the improvements which I consider to be the most relevant ones, but there could be others as well.

VI. Bibliography