LIBRA WEB SYSTEM

Name:	RT Fitchat
Student number:	13156128
Document Description:	ITRW221 System Proposal
Date:	12 August 2005
Project name:	Libra Web System (formerly entitled Driehoek Library System)
Development team:	3key Systems

Table of Contents

1. Description

Libra Web System is a web-based library content management system that provides user-friendly access to library features for personnel and patrons. It is written in C++ and uses the open-source Apache HTTP web server software together with MySQL as the underlying DBMS.

Libra Web System (henceforth also referred to as Libra) includes administrative facilities for handling:

- attainment and removal of books from library stock;
- adding, removing and updating member accounts;
- checkouts, check-ins and the reservation of books; and
- blacklisted members, customarily prohibiting them from using library facilities.

Aside from these standard features, Libra provides a uniform interface in both Afrikaans and English, with the localization modularity to allow extension to additional languages.

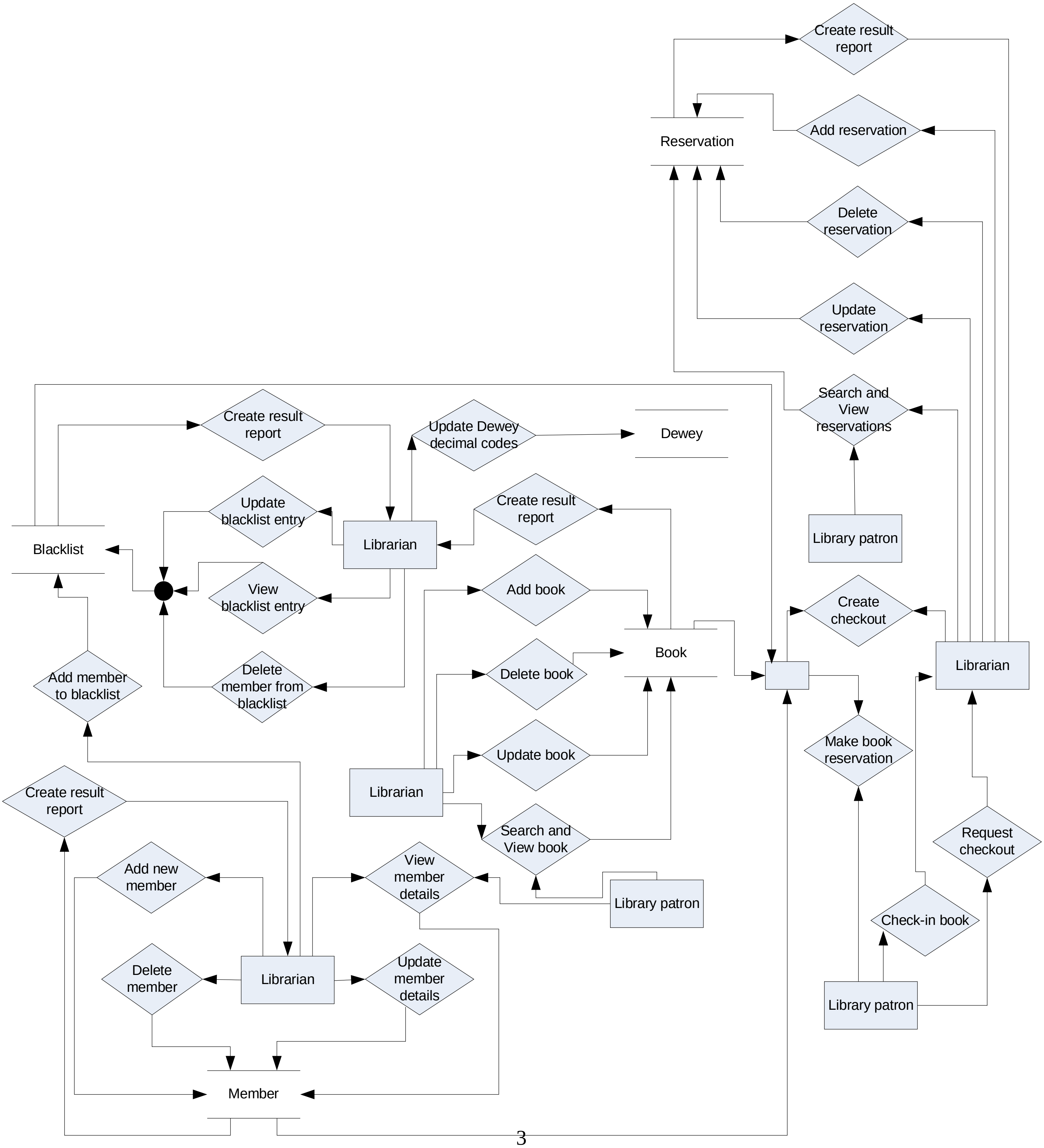
The features supplied for library members stand apart from the administrative section of Libra. Members have their own section where they are allowed to:

- search for library books
- make reservations
- view membership information

Apart from the web interface, plans in the pipelines include hardware support for members to make reservations via WAP-enabled devices, and for a barcode scanning device to ease the identification process of library books. Data integrity is of prime concern, and as such, an automated database backup feature will also be on the agenda.

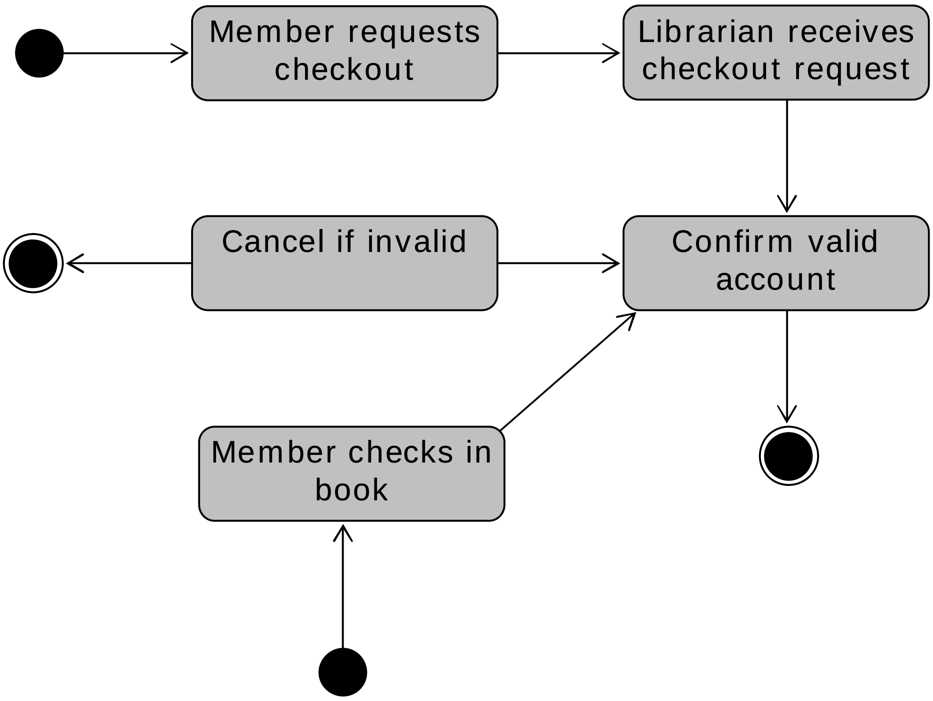
We have moved into an era where faster is better, and less is often more. In terms of systems analysis, this translates to developing systems that make foundational use of Internet connectivity.

2. Data Flow Diagram

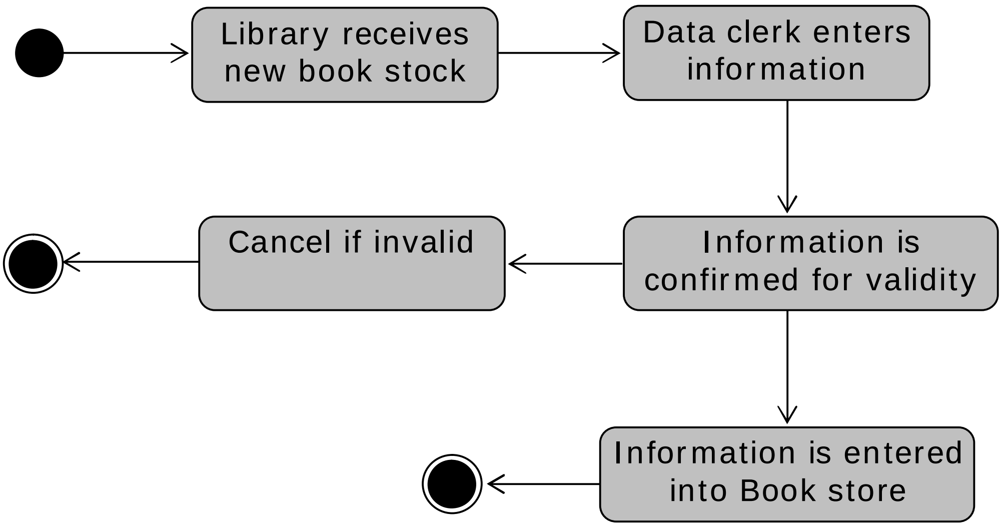


3. Activity Diagrams

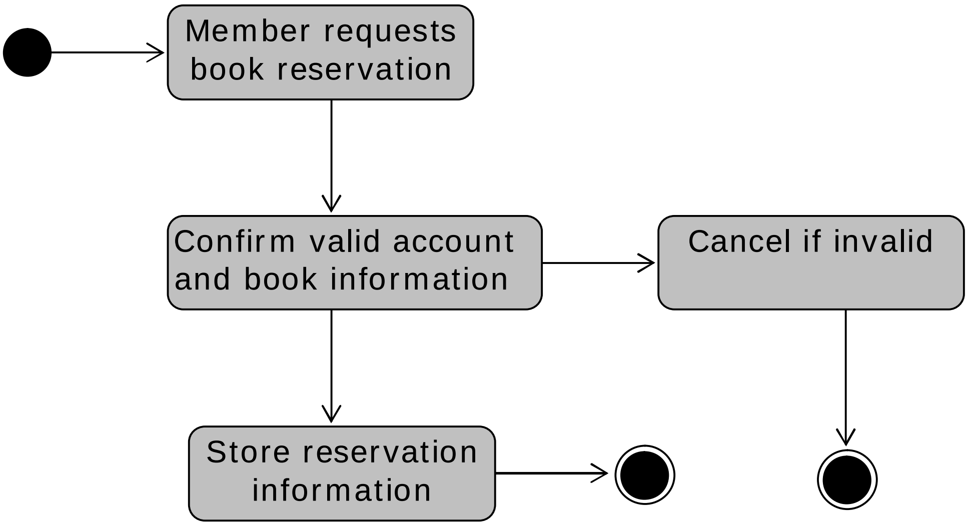
Activity 1: Book checkout/check-in



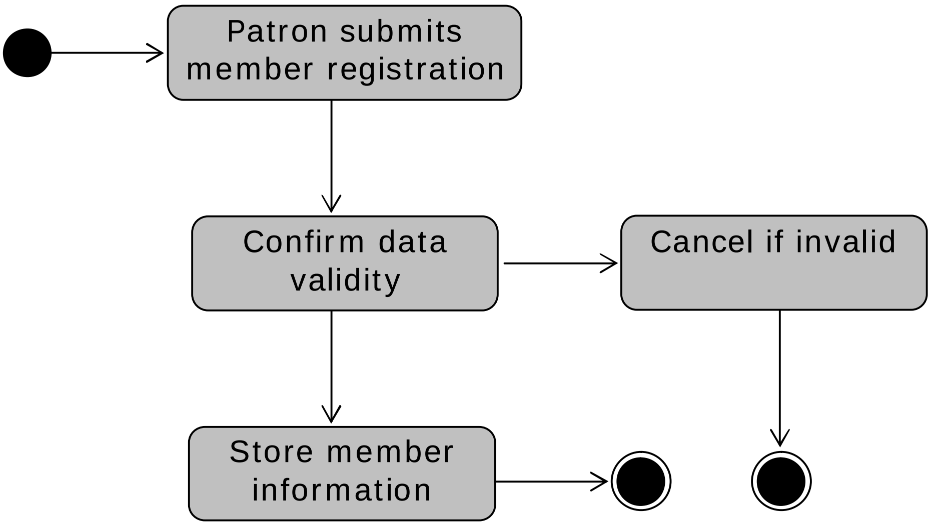
Activity 2: Add Book to Library


Library receives new book stock
Data clerk enters information
Cancel if invalid
Information is confirmed for validity
Information is entered into Book store

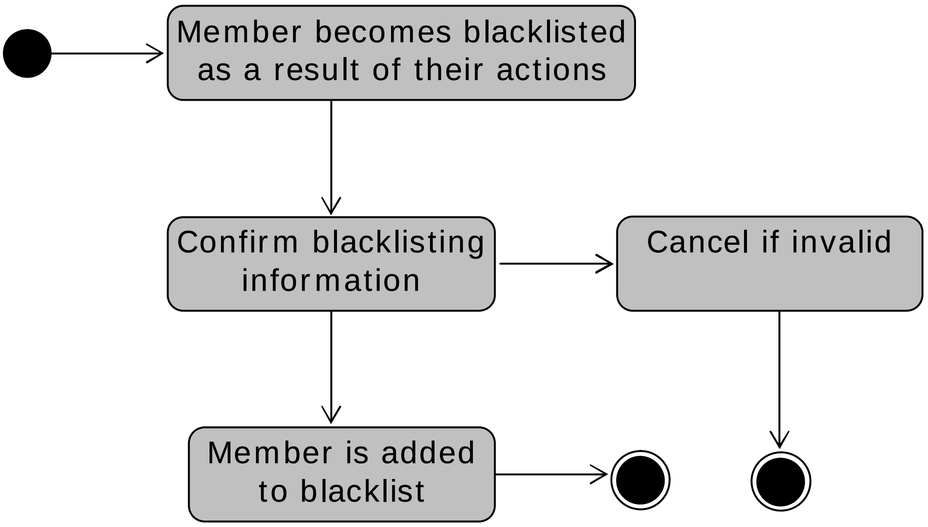
Activity 3: Make Book Reservation


Member requests book reservation
Confirm valid account and book information
Cancel if invalid
Store reservation information

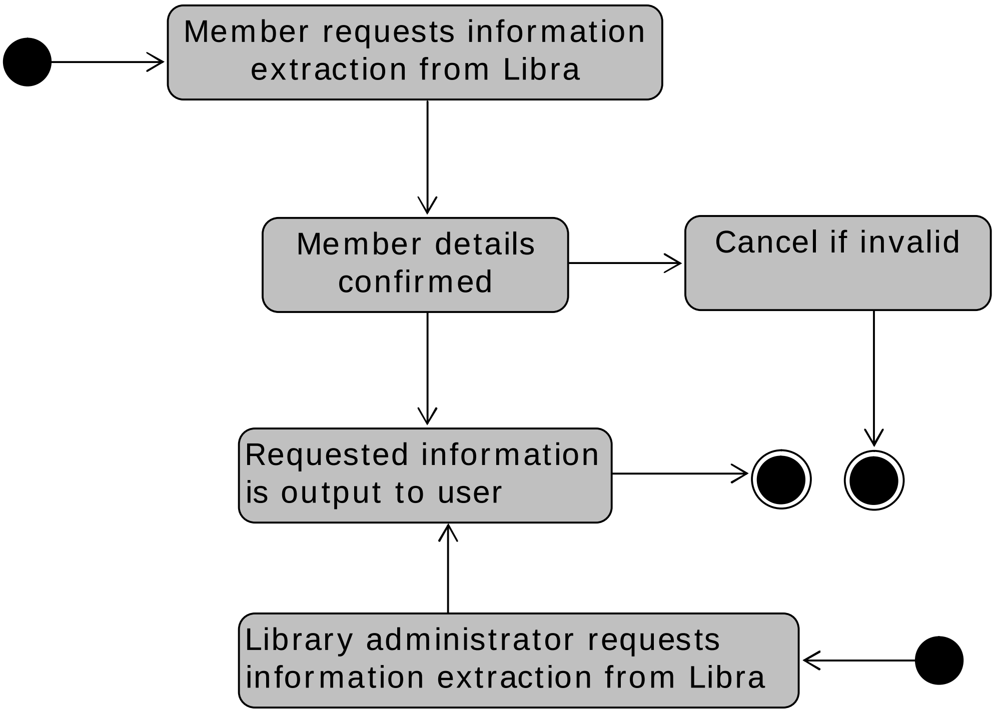
Activity 4: New member registration



Activity 5: Add Member to Blacklist


Member becomes blacklisted as a result of their actions
Confirm blacklisting information
Cancel if invalid
Member is added to blacklist

Activity 6: System User Requests Information Extraction


Member requests information extraction from Libra
Member details confirmed
Cancel if invalid
Requested information is output to user
Library administrator requests information extraction from Libra

4. Use-Case Narratives

Use-Case Name:	Checkout Book	**Use-Case Type:** **Business Requirements:** ☑
Use-Case ID:	LUC03	
Priority:	High	
Source:	Management	
Primary Business Actor:	Member	
Other Participating Actors:	Librarian	
Description:	This use case describes the event of a member checking out a book.	
Precondition:	• The patron is a member. • The member is not blacklisted. • The member's account has not expired. • The book to be checked out is in the Book table of the database. • The book is not already checked out. • The book is not reserved by another member. • The return date is not before the current date.	
Trigger:	This use case is initiated by a librarian whenever a member checks out a book.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: Librarian enters information through the Libra software interface.	**Step 2:** Libra validates supplied information. **Step 3:** Libra inserts checkout information into Checkout table of the database.
Alternate Courses:	**Alt-Step 2:** Libra notifies system user of discrepancy in supplied information, and aborts the process.	
Conclusion:	The use case concludes when the checkout is inserted into the database, or if an error occurred causing the process to abort.	
Postcondition:	The entry is assigned a Checkout ID, and the member and book details are recorded, together with the checkout date and expected return date.	
Business Rules:	The Book ID is not already present in an instance where the *inDate* field equals null.	

Use-Case Name:	Search Book	**Use-Case Type:** **Business Requirements:** ☑
Use-Case ID:	LUC08	
Priority:	High	
Source:	Management	
Primary Business Actor:	Librarian	
Other Participating Actors:	Member	
Description:	This use case describes the event of a system user searching for books by entering the appropriate search criteria.	
Precondition:	The relevant information to allow processing has been passed.	
Trigger:	This use case is initiated by a librarian, patron or member who launches a book search from a Libra interface.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: System user enters information through a Libra software interface.	**Step 2:** Libra validates supplied information. **Step 3:** Libra outputs the desired information by means of a generated report.
Alternate Courses:	**Alt-Step 2:** Libra notifies system user of discrepancy in supplied information, and aborts the process.	
Conclusion:	The use case concludes when the information has been extracted (by means of the *Extract information from Libra* use-case) and presented to the system user.	
Postcondition:	N/A	
Business Rules:	• A member supplies the Book Title, Author, ISBN and Year, of the book they are searching for. • A librarian may supply either some, or no, criteria, in SQL format. If no criteria are supplied, all books are extracted from the Book table.	

Use-Case Name:	Make Book Reservation	**Use-Case Type:** **Business Requirements:** ☑
Use-Case ID:	LUC05	
Priority:	High	
Source:	Management	
Primary Business Actor:	Member	
Other Participating Actors:	Librarian	
Description:	This use case describes the event of a system user reserving a previously checked out book.	
Precondition:	• The book is already checked out. • The book has not already been reserved by this member. • The member is not blacklisted.	
Trigger:	This use case is initiated by a system user to reserve a checked out book.	
Typical Course of Events:	**Actor Action**	**System Response**
	Step 1: System user enters information through the Libra software interface.	**Step 2:** Libra validates supplied information. **Step 3:** Libra inserts member information into Reservation table of the database.
Alternate Courses:	**Alt-Step 2:** Libra notifies system user of discrepancy in supplied information, and aborts the process.	
Conclusion:	The use case concludes when the reservation is inserted into the database, or if an error occurred causing the process to abort.	
Postcondition:	The member has reserved the checked out book, and will potentially be notified when the book is checked in.	

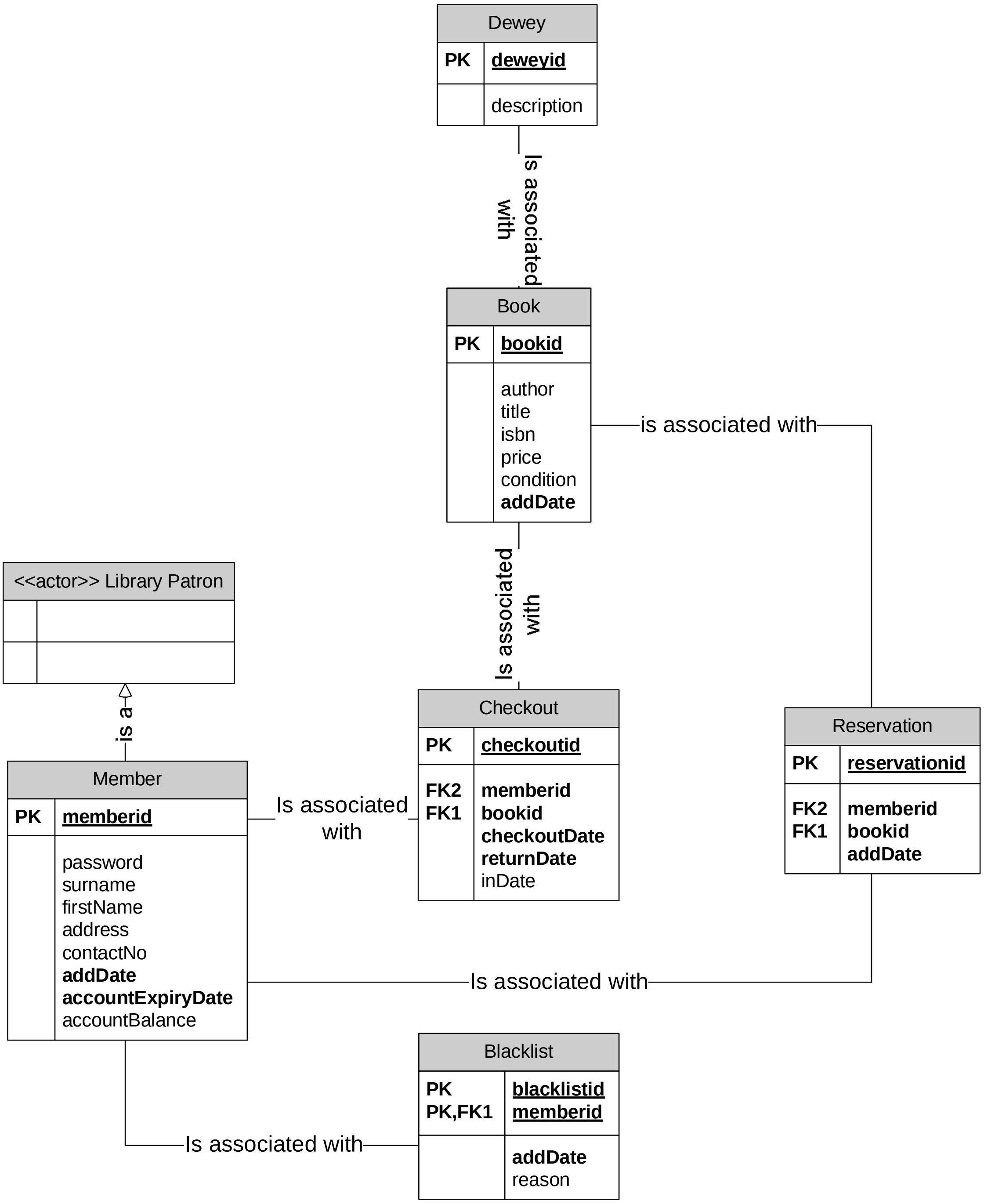
5. Business Objects

The Object List below reflects the objects that the Make Book Reservation use-case (LUC05) interacts with:

Blacklist
Book
Checkout
Dewey
Librarian
Member
Patron
Reservation

In the next section, the Class Diagram illustrates, for the Make Book Reservation use-case, the entities involved, together with their relationships.

6. Class Diagram


Dewey
PK
deweyid
description
Is associated with
Book
PK
bookid
author
title
isbn
price
condition
addDate
is associated with
Is associated with
<<actor>> Library Patron
is a
Checkout
PK
checkoutid
FK2
FK1
memberid
bookid
checkoutDate
returnDate
inDate
Reservation
PK
reservationid
FK2
FK1
memberid
bookid
addDate
Member
PK
memberid
password
surname
firstName
address
contactNo
addDate
accountExpiryDate
accountBalance
Is associated with
Is associated with
Blacklist
PK
PK,FK1
blacklistid
memberid
addDate
reason
Is associated with