Case Study : Design
- Library Management System

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr
Stage 2000. Build
6 Phases of ‘Build’ Stage

1000 Plan and Elaboration

2000 Build

3000 Deployment

2100 Cycle 1

2200 Cycle 2

2n00 Cycle n

2110 Revise Plan

2120 Sync. Artifacts

2130 Analyze

2140 Design

2150 Construct

2160 Test

Konkuk University
Phase 2040.
Design
Phase 2040. Design

- Phase 2040 Activities

2140 Design

- 2141 Design Real Use Cases
- 2142 Define Reports, UI, and Storyboards
- 2143 Refine System Architecture
- 2144 Define Interaction Diagrams
- 2145 Define Design Class Diagrams
- 2146 Define Database Schema

a. In parallel with interaction diagrams
b. Varied order
Activity 2041. Design Real Use Cases

- Define Interaction Diagrams
- Define Reports, UI, and Storyboards
- Design Real Use Cases
- Define Design Class Diagrams
- Refine System Architecture
- Define Database Schema

- In parallel with interaction diagrams
- Varied order
# Activity 2041. Design Real Use Cases

## 1. Make Reservation

<table>
<thead>
<tr>
<th>Use Case</th>
<th>1. Make Reservation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Create a new reservation</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R1.1, R3.1</td>
</tr>
<tr>
<td></td>
<td>Use Case: “Add Borrower”</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>A borrower should be registered.</td>
</tr>
</tbody>
</table>

### Typical Courses of Events

1. (A) A librarian inputs an isbn and ssn of the title
2. (S) Find a corresponding title
3. (S) Find a corresponding borrower
4. (S) Create a new reservation
5. (S) Store the new reservation
6. (S) Increase reservationCount in the borrower
7. (S) Increase reservationCount in the title

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

- Line 2: If the title does not exist, display an error message.
- Line 3: If the borrower does not exist, display an error message.
## Activity 2041. Design Real Use Cases

### 2. Remove Reservation

<table>
<thead>
<tr>
<th>Use Case</th>
<th>2. Remove Reservation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Remove a reservation information</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference   | System Functions: R1.2, R1.3  
                  | Use Case: “Lend Item”    |
| Pre-Requisites    | A borrower should be registered.  
                  | A title should have been reserved. |
| Typical Courses of Events | (A) : Actor, (S) : System  
                          | 1. (A) A librarian inputs an isbn of the title  
                          | 2. (S) Find a corresponding reservation  
                          | 3. (S) Remove the reservation  
                          | 4. (S) Decrease reservationCount of the borrower  
                          | 5. (S) Decrease reservationCount of the title |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | Line 2: If the reservation does not exist, display an error message. |
Activity 2041. Design Real Use Cases

3. Lend Item

<table>
<thead>
<tr>
<th>Use Case</th>
<th>3. Lent Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lend items to a borrower</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R1.3, R1.2, R3.1</td>
</tr>
<tr>
<td></td>
<td>Use Cases: “Remove Reservation”, “Add Borrower”</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>An item should exist.</td>
</tr>
</tbody>
</table>

**Typical Courses of Events**

1. (A) A librarian inputs an item’s ID and ssn of the borrower
2. (S) Find a corresponding borrower
3. (S) Find a corresponding item
4. (S) Create a new loan
5. (S) Store the new loan
6. (S) Set validLoan to true
7. (S) Increase loanCount of borrower
8. (S) Set available to false
9. (S) Decrease availableCount of the title

**Alternative Courses of Events**

N/A

**Exceptional Courses of Events**

Line 2: If the borrower does not exist, invoke “Add Borrower” use case.
## Activity 2041. Design Real Use Cases

### 4. Return Item

<table>
<thead>
<tr>
<th>Use Case</th>
<th>4. Return Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Return items loaned</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R1.4.1, R1.4.2, R1.6</td>
</tr>
<tr>
<td></td>
<td>Use Cases: “Calculate Late-Return-Fee”, “Notify Availability”</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>An item should have been loaned.</td>
</tr>
</tbody>
</table>

#### Typical Courses of Events

<table>
<thead>
<tr>
<th></th>
<th>(A) : Actor, (S) : System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(A) A librarian inputs an item’s ID</td>
</tr>
<tr>
<td>2</td>
<td>(S) Find a corresponding loan</td>
</tr>
<tr>
<td>3</td>
<td>(S) Get item information from the loan</td>
</tr>
<tr>
<td>4</td>
<td>(S) Get title information from the item</td>
</tr>
<tr>
<td>5</td>
<td>(S) Get loanPeriod from the title</td>
</tr>
<tr>
<td>6</td>
<td>(S) Compute calculateLateReturnFee</td>
</tr>
<tr>
<td>7</td>
<td>(S) Check reservationCount of the title.</td>
</tr>
<tr>
<td>8</td>
<td>(S) If the title is reserved, find the corresponding reservation</td>
</tr>
<tr>
<td>9</td>
<td>(S) Decrease loanCount of the loan.</td>
</tr>
<tr>
<td>10</td>
<td>(S) Decrease loanCount of the Borrower.</td>
</tr>
<tr>
<td>11</td>
<td>(S) Set validLoan of the borrower to false.</td>
</tr>
<tr>
<td>12</td>
<td>(S) Set available of the item to true.</td>
</tr>
<tr>
<td>13</td>
<td>(S) Increase AvailbaleCount of the title.</td>
</tr>
</tbody>
</table>

#### Alternative Courses of Events

N/A

#### Exceptional Courses of Events

Line 2: If the loan does not exist, display an error message.
Activity 2041. Design Real Use Cases

5. Calculate Late-Return-Fee

<table>
<thead>
<tr>
<th>Use Case</th>
<th>5. Calculate Late-Return-Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>None</td>
</tr>
<tr>
<td>Purpose</td>
<td>Compute late-return fee for an item returned late</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference          | System Functions: R1.4.1, R1.4.2  
                          | Use Cases: “Return Item”    |
| Pre-Requisites           | Lending time of an item should have expired |
| Typical Courses of Events| (A) : Actor, (S) : System  
                          | 1. (S) Calculate Late-Return-Fee of the item  
                          | 2. (S) Display the Late-Return-Fee |
| Alternative Courses of Events | N/A                       |
| Exceptional Courses of Events | N/A                       |
## Activity 2041. Design Real Use Cases

### 6. Get Replacement-Fee

<table>
<thead>
<tr>
<th>Use Case</th>
<th>6. Get Replacement-Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Compute replacement-fee for a lost title</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R1.5</td>
</tr>
<tr>
<td></td>
<td>Use Cases: -</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>A title should be lost.</td>
</tr>
</tbody>
</table>

**Typical Courses of Events**

- (A) : Actor, (S) : System
- 1. (A) A librarian inputs an item’s ID
- 1. (S) Find a corresponding loan
- 2. (S) Get an item from the loan
- 3. (S) Get a title from the item
- 4. (S) Get price of the title
- 5. (S) Compute replacementFee
- 6. (S) Set validLoan to false
- 7. (S) Update the loan
- 8. (S) Decrease loanCount of the borrower.
- 9. (S) Set the lost of the item to true. (?)
- 10. (S) Decrease numOfItem of the title.

**Alternative Courses of Events**

N/A

**Exceptional Courses of Events**

Line 2: If the loan does not exist, display an error message.
# Activity 2041. Design Real Use Cases

## 7. Notify Availability

<table>
<thead>
<tr>
<th>Use Case</th>
<th>7. Notify Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Notify availability of a reserved item</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Primary and Real</td>
</tr>
<tr>
<td><strong>Cross Reference</strong></td>
<td>System Functions: R1.4.1, R1.6, R2.4 Use Cases: “Return Item”, “Add Item”</td>
</tr>
<tr>
<td><strong>Pre-Requisites</strong></td>
<td>An item should have been returned or a new item should have been added.</td>
</tr>
<tr>
<td><strong>Typical Courses of Events</strong></td>
<td>(A) : Actor, (S) : System 1. (S) Print a post-card</td>
</tr>
<tr>
<td><strong>Alternative Courses of Events</strong></td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Exceptional Courses of Events</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>
## Activity 2041. Design Real Use Cases

### 8. Add Title

<table>
<thead>
<tr>
<th>Use Case</th>
<th>8. Add Title</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Librarian</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Register a new title</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| **Cross Reference** | System Functions: R2.1, R2.4  
Use Case: “Add Item” |
| **Pre-Requisites** | N/A |

#### Typical Courses of Events

1. (A) A librarian inputs title information such as name, isbn, price, publisher, loanPeriod (Book: author, Magazine:month, publishCycle)
2. (S) Find a corresponding title
3. (S) Create a new title
4. (S) Store the new title
5. (S) Invoke “Add Item”

#### Alternative Courses of Events

N/A

#### Exceptional Courses of Events

Line 1: If the title already exists, display an error message.
Activity 2033. Define Domain Model

- **Item**
  - ID: Integer
  - available: Boolean

- **Title**
  - name: String
  - isbn: String
  - count: Integer
  - price: Float
  - publisher: String
  - lending time: Integer

- **Reservation**
  - date: Date

- **Book**
  - author: String

- **Magazine**
  - month: Integer

- **Borrower**
  - name: String
  - age: Integer
  - SSN: String
  - address: String
  - phone: String
  - zip: String

- **Librarian**
  - name: String
  - user ID: String
  - password: String

- **Has/Have** relationships:
  - 1..* Item
  - 1..* Loan
  - 1..* Book
  - 1..* Magazine
  - 1..* Borrower

- **Copy of** relationship:
  - 1..* Item

- **Refer to** relationships:
  - 1..* Loan
  - 0..1 Date

- **Has/Have** relationships:
  - 1..* Librarian
  - 1..* Borrower

Konkuk University
# Activity 2041. Design Real Use Cases

## 9. Remove Title

<table>
<thead>
<tr>
<th>Use Case</th>
<th>9. Remove Title</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Librarian</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Delete information of a title</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Primary and Real</td>
</tr>
<tr>
<td><strong>Cross Reference</strong></td>
<td>System Functions: R2.2</td>
</tr>
<tr>
<td></td>
<td>Use Case: -</td>
</tr>
<tr>
<td><strong>Pre-Requisites</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Typical Courses of Events

1. (A) A librarian inputs a title's isbn to remove
2. (S) Find a corresponding title
3. (S) Check if the corresponding title is reserved.
4. (S) If the title is reserved, Remove the reservation
5. (S) Check the item of the title is loaned. (and then what about items?)
6. (S) Remove the title

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

Line 2: If the title does not exist, display an error message.
Activity 2041. Design Real Use Cases

10. Update Title

<table>
<thead>
<tr>
<th>Use Case</th>
<th>10. Update Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Update information of a title</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R2.3 Use Case: -</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>N/A</td>
</tr>
<tr>
<td>Typical Courses of Events</td>
<td>(A) : Actor, (S) : System</td>
</tr>
<tr>
<td></td>
<td>1. (A) A librarian inputs a title's isbn and information of the title to change</td>
</tr>
<tr>
<td></td>
<td>2. (S) Find a corresponding title</td>
</tr>
<tr>
<td></td>
<td>3. (S) Update the title (How?)</td>
</tr>
<tr>
<td>Alternative Courses of Events</td>
<td>N/A</td>
</tr>
<tr>
<td>Exceptional Courses of Events</td>
<td>Line 2: If the item does not exist, display &quot;Not Existing Title&quot;. Error message. Line 3: If the isbn is changed, then update items too, (How?)</td>
</tr>
</tbody>
</table>
Activity 2041. Design Real Use Cases

11. Add Item

<table>
<thead>
<tr>
<th>Use Case</th>
<th>11. Add Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Add a new item</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference   | System Functions: R2.4  
                  | Use Cases: “Add Title” |
| Pre-Requisites    | N/A               |

**Typical Courses of Events**
- (A) : Actor,  (S) : System
  1. A librarian inputs an item’s id
  2. Find a corresponding title
  3. Get an item’s ID from the title
  4. Create a new item
  5. Store the new item
  6. Increase `numOfItem` of the title (?)
  7. Increase `availablecount` of the item

**Alternative Courses of Events**
- N/A

**Exceptional Courses of Events**
- Line 2: If the title does not exist, display an error message.
## 12. Remove Item

<table>
<thead>
<tr>
<th>Use Case</th>
<th>12. Remove Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Remove information of an item</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference | System Functions: R2.1, R2.5  
Use Case: “Remove Title” |
| Pre-Requisites | N/A |

### Typical Courses of Events

- (A) Actor, (S) System
  1. (A) A librarian inputs an item’s ID
  2. (S) Find a corresponding item
  3. (S) Check if the item is borrowed
  4. (S) If the item is borrowed, decrease numOfItem of the title
  5. (S) Decrease availableCount of the title
  6. (S) Remove the item

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

- Line 2: If the item does not exist, display an error message.
- Line 3: If the item was already borrowed, display an error message.
## Activity 2041. Design Real Use Cases

### 13. Update Item

<table>
<thead>
<tr>
<th>Use Case</th>
<th>13. Update Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Update information of an item</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference  | System Functions: R2.6  
Use Case: - |
| Pre-Requisites   | N/A             |

### Typical Courses of Events

- (A) : Actor, (S) : System
  1. (A) A librarian inputs the item's id and information to change
  2. (S) Find A corresponding item
  3. (S) Update the item
  4. (S) If a lost of the item is true, decrease numOfItem of the title.
  5. (S) Decrease the availableCount of the title.
  6. (S) If a lost of the item is false, increase numOfItem of the title.
     (What? Only for these cases “Update Item” are used?)
  7. (S) Increase availableCount of the title

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

Line 2: If the item does not exist, display an error message.
Activity 2041. Design Real Use Cases

14. Add Borrower

<table>
<thead>
<tr>
<th>Use Case</th>
<th>14. Add Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Register a new borrower</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R1.1, R1.3, R3.1 Use Cases: “Make Reservation”, “Lend Item”</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>N/A</td>
</tr>
<tr>
<td>Typical Courses of Events</td>
<td>(A) : Actor, (S) : System</td>
</tr>
<tr>
<td>1. (A) A librarian inputs a borrower's name, ssn, and address.</td>
<td></td>
</tr>
<tr>
<td>2. (S) Find a corresponding borrower</td>
<td></td>
</tr>
<tr>
<td>3. (S) Create a new borrower</td>
<td></td>
</tr>
<tr>
<td>4. (S) Store the new borrower</td>
<td></td>
</tr>
<tr>
<td>Alternative Courses of Events</td>
<td>N/A</td>
</tr>
<tr>
<td>Exceptional Courses of Events</td>
<td>Line 2: If the borrower exists already, display an error message.</td>
</tr>
</tbody>
</table>
## 15. Remove Borrower

<table>
<thead>
<tr>
<th>Use Case</th>
<th>15. Remove Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Librarian</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Remove information of a borrower</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Primary and Real</td>
</tr>
<tr>
<td><strong>Cross Reference</strong></td>
<td>System Functions: R3.2 Use Case: -</td>
</tr>
<tr>
<td><strong>Pre-Requisites</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Typical Courses of Events

- (A) : Actor, (S) : System
- 1. (A) A librarian inputs the borrower’s ssn
- 2. (S) Find a corresponding borrower
- 3. (S) Find a loan of the borrower
- 4. (S) If the loan is invalid, find a reservation
- 5. (S) Get the title of the reservation
- 6. (S) Decrease reservationCount of the title
- 7. (S) Remove borrower

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

Line 2: If the borrower does not exist, display an error message.
Line 3: If the loan is still valid, display an error message.

Konkuk University
## 16. Update Borrower

<table>
<thead>
<tr>
<th>Use Case</th>
<th>16. Update Borrower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Update information of a borrower</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Primary and Real</td>
</tr>
</tbody>
</table>
| Cross Reference     | System Functions: R3.3  
Use Case: -          |
| Pre-Requisites      | N/A                 |
| Typical Courses of Events | (A) : Actor, (S) : System  
1. (A) A librarian inputs a borrower’s ssn and information to change  
2. (S) Find a corresponding borrower  
3. (S) Update the borrower |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | Line 2: If the borrower does not exist, display an error message. |
Activity 2041. Design Real Use Cases

17. Log-In

<table>
<thead>
<tr>
<th>Use Case</th>
<th>17. Log-In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Librarian</td>
</tr>
<tr>
<td>Purpose</td>
<td>Check access authority of a librarian</td>
</tr>
<tr>
<td>Overview</td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td>Type</td>
<td>Secondary and Real</td>
</tr>
<tr>
<td>Cross Reference</td>
<td>System Functions: R4.1 Use Case: -</td>
</tr>
<tr>
<td>Pre-Requisites</td>
<td>A librarian should have user name and password.</td>
</tr>
</tbody>
</table>
| Typical Courses of Events | (A) : Actor, (S) : System  
1. (A) A librarian inputs an userID and password  
2. (S) Check if the userID and password are correct |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | Line 2: If the userID and password are not correct, display an error message. |
# Activity 2041. Design Real Use Cases

## 18. Log-Out

<table>
<thead>
<tr>
<th>Use Case</th>
<th>18. Log-Out</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Librarian</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Exit the library management system</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Secondary and Essential</td>
</tr>
</tbody>
</table>
| **Cross Reference** | System Functions: R4.1  
Use Case: - |
| **Pre-Requisites** | A librarian should have user name and password. |

### Typical Courses of Events

(A) : Actor,  (S) : System
1. (A) A librarian selects “LogOut”
2. (S) Check if the *userID* is correct and then exit the system

### Alternative Courses of Events

N/A

### Exceptional Courses of Events

Line 2: If the *userID* is incorrect, display an error message.
## Activity 2041. Design Real Use Cases

### 19. Count Loans

<table>
<thead>
<tr>
<th>Use Case</th>
<th>19. Count Loans</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Librarian</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Compute total count of the titles checked out</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>(As in the business use case)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Secondary and Essential</td>
</tr>
<tr>
<td><strong>Cross Reference</strong></td>
<td>System Functions: R5.1 &lt;br&gt; Use Case: -</td>
</tr>
<tr>
<td><strong>Pre-Requisites</strong></td>
<td>A librarian should have user name and password.</td>
</tr>
<tr>
<td><strong>Typical Courses of Events</strong></td>
<td>(A): Actor, (S): System  &lt;br&gt; 1. (A) A librarian requests loan count  &lt;br&gt; 2. (S) Get numOfLoan of the loan</td>
</tr>
<tr>
<td><strong>Alternative Courses of Events</strong></td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Exceptional Courses of Events</strong></td>
<td>N/A (Really?)</td>
</tr>
</tbody>
</table>
Activity 2042.
Define Reports, UI, and Storyboards

- Design Real Use Cases (2141)
- Define Reports, UI, and Storyboards (2142)
- Refine System Architecture (2143)
- Define Interaction Diagrams (2144)
- Define Design Class Diagrams (2145)
- Define Database Schema (2146)

Options:
- In parallel with interaction diagrams
- Varied order

Konkuk University
Activity 2042. Define Reports, UI, and Storyboards

- Make Reservation
Activity 2042. Define Reports, UI, and Storyboards

- Lent Item
Activity 2042.
Define Reports, UI, and Storyboards

• Count Loans
Activity 2043. Refine System Architecture

- 2141 Design Real Use Cases
- 2142 Define Reports, UI, and Storyboards
- 2143 Refine System Architecture
  - a. In parallel with interaction diagrams
  - b. Varied order
- 2144 Define Interaction Diagrams
- 2145 Define Design Class Diagrams
- 2146 Define Database Schema
Activity 2043. Refine System Architecture

• Steps 1~3 : Drawing Deployment Diagram
Activity 2043. Refine System Architecture

- Steps 4~7: Drawing Package Diagram

**Application Logic Layer**
- Business Object Package
  - Loan
  - Borrower
  - Book
  - Item
  - Title
  - Reservation
  - Magazine
  - Librarian

**Storage Layer**
- Database Package
  - Database
Activity 2044. Define Interaction Diagrams

- Define Reports, UI, and Storyboards
- Refine System Architecture
- Define Design Class Diagrams
- Define Database Schema

- In parallel with interaction diagrams
- Varied order
1. Make Reservation

makeReservation()

- titleRef:= searchTitle(isbn:String)
  - [titleRef is invalid]
    - displayMessage("Error")
  - [titleRef is valid]
    - borrowerRef:= searchBorrower(ssn:String)
      - [borrowerRef is invalid]
        - displayMessage("Error")
      - [borrowerRef is valid]
        - reservationRef:= new Reservation(titleRef:Title, borrowerRef:Borrower)
          - addReservation(reservationRef:Reservation)
            - addReservationDB(reservationRef:Reservation)
              - increaseReservationCount()
2. Remove Reservation

- removeReservation() [reservationRef is valid]
- displayMessage("Error") [reservationRef is invalid]
- removeReservation(reservationRef: Reservation)
- removeReservationDB(reservationRef: Reservation)
- decreaseReservationCount()
- decreaseReservationCount()
- displayMessage("OK")

- reservationRef := searchReservation(isbn: ISBNType)
- searchReservationDB(isbn: ISBNType)
3. Lend Item

lendItem()

borrowerRef:= searchBorrower(ssn: String)

itemRef:= searchItem(itemID: String)

loanRef:= new Loan(itemRef: Item, borrowerRef: Borrower)

addLoan(loanRef: Loan)

setValidLoan(true)

increaseLoanCount()

setAvailable(false)

decreaseAvailableCount()

displayMessage("OK")

[borrowerRef is invalid]
displayMessage("Error")

searchBorrowerDB(ssn: String)

searchItemDB(itemID: String)

addLoanDB(loanRef: Loan)
4. Return Item

- **Controller**
  - `returnItem()`
  - `loanRef := searchLoan(itemID: String)`
  - `searchLoanDB(itemID: String)`

- **DataBase**
  - `reservationRef := searchReservationDB(titleRef: Title)`
  - `printNotifyCard(titleRef: Title)`
  - `printCard(reservationRef: Reservation)`

- **Borrower**
  - `displayMessage(lateReturnFee)`
  - `displayMessage("OK")`

- **Item**
  - `itemRef := getItem()`
  - `titleRef := getTitle()`
  - `loanPeriod := getLoanPeriod()`

- **Loan**
  - `decreaseLoanCount()`
  - `setAvailable(true)`
  - `setValidLoan(false)`

- **Reservation**
  - `valid := isReserved()`
  - `lateReturnFee := calculateLateReturnFee(loanPeriod)`
  - `reservationRef := searchReservation(titleRef: Title)`
  - `increaseAvailable`

- **Librarian**
  - `displayMessage("Error")`
  - `displayMessage("Not Reserved")`
6. Get Replacement-Fee

- Librarian
  - getReplacementFee()
  - loanRef:=searchLoan(itemID: String)
  - decreaseLoanCount()
  - setLost(true)
  - searchLoanDB(itemID: String)
  - displayMessage(replacementFee)
  - displayMessage(“OK”)

- Controller
  - :Loan
  - :Title
  - :Item
  - Borrower
  - DataBase

- itemRef:=getItem()

- [loanRef is valid]
  - itemRef:=getIem()
  - titleRef:=getTitle()
  - price=getPrice()
  - replacementFee:=calculateReplacementFee(price)
  - setValidLoan(false)
  - updateLoan(loanRef: Loan)
  - updateLoanDB(loanRef: Loan)
  - decreaseLoanCount()
  - setLost(true)
  - decreaseNumOfItem()
8. Add Title

![UML diagram showing the process of adding a title to the database. The diagram includes the following steps:

1. Librarian initiates the process by calling `addTitle()`.
2. If the reference is valid, `searchTitle(isbn: String)` is called.
3. If the reference is invalid and the type is "Book," a new `Book` object is created with the provided parameters.
4. If the reference is invalid and the type is "Magazine," a new `Magazine` object is created with the provided parameters.
5. The objects are then added to the database with `addTitleDB(titleRef: Title)` and `addItem(isbn: ISBNType)`.
9. Remove Title

```
removeTitle()
[titleRef is invalid]
displayMessage("Error")

[titleRef is valid]
titleRef:=searchTitle(isbn: String)
searchTitleDB(isbn: String)
[reservationRef is valid]
reservationRef:=searchReservation(titleRef:Title)
searchReservationDB(titleRef:Title)
decreaseReservationCount()
removeTitle(titleRef:Title)
removeItem(isbn: ISBNType)
removeTitleDB(titleRef:Title)
declineReserervationCount()
declineReserervationCount()
removeItem(isbn: ISBNType)
removeTitleDB(titleRef:Title)
```

Konkuk University
10. Update Title

 Librarian

 Controller

 :Title

 DataBase

 updateTitle( )

titleRef:= searchTitle(isbn: String)

 [titleRef is invalid]
 displayMessage("Error")

 [titleRef is valid]
 updateTitle(titleRef: Title)

 searchTitleDB(isbn: String)

 updateTitleDB(titleRef: Title)

displayMessage("OK")

Konkuk University
11. Add Item

```
addItem()  

[titleRef is invalid]  
displayMessage("Error")  

[titleRef is valid]  
titleRef := searchTitle(isbn:String)  

itemID := getNewItemID()  

itemRef := new Item(itemID: String, titleRef:Title)  

addItem(itemRef:Item)  

addItemDB(itemRef:Item)  

displayMessage("OK")  

searchTitleDB(isbn:String)  

increaseAvailableCount()  

increaseNumOfItem()  

Konkuk University
```
12. Remove Item

蓁 Librarian

: Controller

: Title

: Item

: Loan

: DataBase

removeItem( )

itemRef:=searchItem(itemID:String)

searchItemDB(itemID:String)

isBorrowed( )

decreaseAvailableCount( )

decreaseNumOfItem( )

removeItem(itemRef:Item)

removeItemDB(itemRef:Item)

displayMessage("OK")

displayMessage("Error")

[itemRef is invalid]

[itemRef is valid]

displayMessage("Error")

displayMessage("Error")

[itemRef is not valid]

Konkuk University
12. Update Item

```
updateItem(itemRef: Item)
itemRef := searchItem(itemID: String)
searchItemDB(itemID: String)
updateItemDB(itemRef: Item)
decreaseNumOfItem()
  [lost = "True"]
  decreaseAvailableCount()
[lost = "False"]
  increaseNumOfItem()
  increaseAvailableCount()
displayMessage("Ok")
displayMessage("Error")
```
14. Add Borrower

addBorrower() → borrowerRef := searchBorrower(ssn: String)

[borrowerRef is valid] → displayMessage("OK")

[borrowerRef is invalid] → borrowerRef := new Borrower(name: String, ssn: String, address: String)

addBorrower(borrowerRef: Borrower) → addBorrowerDB(borrowerRef: Borrower)

searchBorrowerDB(ssn: String)
15. Remove Borrower

```
removeBorrower()

borrowerRef := searchBorrower(ssn: String)

loanRef := searchLoan(borrowerRef: Borrower)

reservationRef[] := searchReservation(borrowerRef: Borrower)

titleRef := getTitle()

decreaseReservationCount()

removeReservation(borrowerRef: Borrower)

removeBorrower(borrowerRef: Borrower)

displayMessage("OK")
```

![Diagram of the process to remove a borrower](image-url)
16. Update Borrower

updateBorrower()

borrowerRef:=searchBorrower(ssn:String)

[borrowerRef is valid]
updateBorrower(borrowerRef:Borrower)

searchBorrowerDB(ssn:String)

[borrowerRef is invalid]
displayMessage("Error")

displayMessage("OK")
17. Log-In

```
logIn()
valid=validate(userID:String, password:String)
valid=validateDB(userID:String, password:String)
displayMessage("OK")
displayMessage("Error")
```
18. Log-Out

```java
logOut( )
valid=logOut(userID:String)
displayMessage("Error")
[valid="False"]

valid=logOut(userID:String)
[valid="True"]
displayMessage("Ok")
```
19. Count Loans

: Librarian

: Controller

: Loan

countLoans( )

displayMessage(numOfLoan)

numOfLoan:=getNumOfLoan()
Activity 2045. Define Design Class Diagram

- Design Real Use Cases
- Define Reports, UI, and Storyboards
- Refine System Architecture
- Define Interaction Diagrams
- Define Design Class Diagrams
- Define Database Schema

- In parallel with interaction diagrams
- Varied order
Activity 2046. Define Database Schema

- Design Real Use Cases
- Define Reports, UI, and Storyboards
- Refine System Architecture
- Define Interaction Diagrams
- Define Design Class Diagrams
- Define Database Schema

a. In parallel with interaction diagrams
b. Varied order
Phase 2040. Design

• Phase 2040 Activities

  a. In parallel with interaction diagrams
  b. Varied order