|  |  |
| --- | --- |
| **Practicum Case** |  |
| COMP6115 | COMP6115001  Object Oriented Analysis and Design |
| **Computer Science** | **E241-COMP6115-EA01-01** |
| ***Valid on*** *Even Semester Year 2023/2024* | **Revision 00** |

**Learning Outcome**

* LO1 – Conceive the basics concepts of object oriented software engineering

**Topic**

* Session 1 – Use Case Diagram and Activity Diagram

## Sub Topics

* Use Case Component
* Activity Component

## Soal

*Case*

**ElibrAry**

**ElibrAry** is a campus library located in the centre of the city. As **ElibrAry** expands its reach and strives to enhance its services, the need for a comprehensive application to streamline customer interactions becomes paramount. The envisioned application should seamlessly manage patrons' requests, ensuring efficient borrowing and returning processes. Some of the requirements of the application are as follows.

* **User Management**

Every student’s data is automatically synced to the library database, so the **students does not have to register themselves** to the application. The students’ data that are stored are the **student ID**, **name**, **email** **address**, **password**, **address**, and **phone** **number**. The **librarians** will also have their own accounts, that store their data such as their **ID**, **name**, **email** **address**, **password**, and **salary**.

* **Borrowing Management**

Every **borrowing** made in the application will be **recorded** in the system. Every borrowing has details of the **start date** and **end date** that the product will be borrowed. Every borrowing also has the details of the **book ID**, **borrower** **ID**, and the **librarian ID** that approves the borrowing. Every borrowing also has a **status**, which can be pending, approved, returned late, and returned on time.

When the student wants to borrow a book, they can see the book details first, and then **request** **borrowing**. The user must input the start and end date of the borrowing, and after that the application will validate if the book is already borrowed during the date range. If there is already a borrowing that has been approved for that duration, the application will display a message that the book will not be available during the date range, other than that the application will insert the borrowing details to the database with status “Pending”.

The librarian can **approve the pending requests** by viewing the pending requests, selecting a request that they want to approve, and approve the borrowing request. The application will validate if the book is already borrowed during the date range. If there is already a borrowing that has been approved for that duration, the application will display an error message which says that the borrowing request could not be approved, other than that the application will update the borrowing status to “Approved”.

The librarian can **process the book return** by viewing the borrowing request and clicking finish borrowing. The application will update the borrowing status to “Returned late” or “Returned on time” according to the end date and returned date. If the borrowing is overdue, then the application will calculate the late return fine and send the billing to the student’s email address.

* **Book Management**

The book data that is stored in the database are the **book ID**, **title**, **author**, and **description**. To help students in finding the books in the application, users will be able to **search book** by using a **keyword**. The application will fetch the related data and display it. If there are no related books, the application will display an error message.

Students can also **add books to their favourite book list** by selecting a book, and then the application will display the book details, and the student can click the add to favourites button. The application will insert the data to the database and display a success message. The students can also **search for books** while **adding the books to their favourite book list**.

\*\*\*

You are asked as a System Analyst to make the **Use Case Diagram** and **Activity Diagram** for the case above.