CS 457 FS2016

Software Architecture Requirements and Design

Cloud Library Project Report

By

Latha Muddu

Swarna Kosaraju

Sai Jyothi Gudibandi

Haritha Ichapurapu

Contents

[**Introduction** 3](#_Toc457045499)

[**Architecture Diagram**……….……………………………………………………………………………………………………………………….4](#_Toc457045500)

[**Architectural Style**](#_Toc457045506)…………………………………………………………………………………………………………………………………….6

[**UML Diagrams**](#_Toc457045510)…………………………………………………………………………………………………………………………………………..7

[**Use Case Diagram**](#_Toc457045511)………………………………………………..……………………………………………………………………………7

[**Class Diagram**](#_Toc457045512)…………………………………………………………..……………………………………………………………………….8

[**Activity Diagram**](#_Toc457045512)……………………………………………………………………………………………………………………………10

[**Sequence Diagram**](#_Toc457045512)……………………………………………………………………………………………………………………….…11

**1. Suggest Books**……………………………………………………………………………………………………………………..…11

**2. Request Books**………………………………………………………………………………………………………………….…...12

**3. Upload Books**…………………………….………………………………………………………………………………………..…13

[**Design Pattern**](#_Toc457045520)…………………………………………………………………………………………………………………………………………14

**Team Contribution**………………………………………………………………………………………………………………………………..…15

**Conclusion**……………………………………………………………………………………………………………………………………………….15

[**References**](#_Toc457045523)……………………………………………………………………………………………………………………………………………….15

# **Introduction**

## **Use Case:**

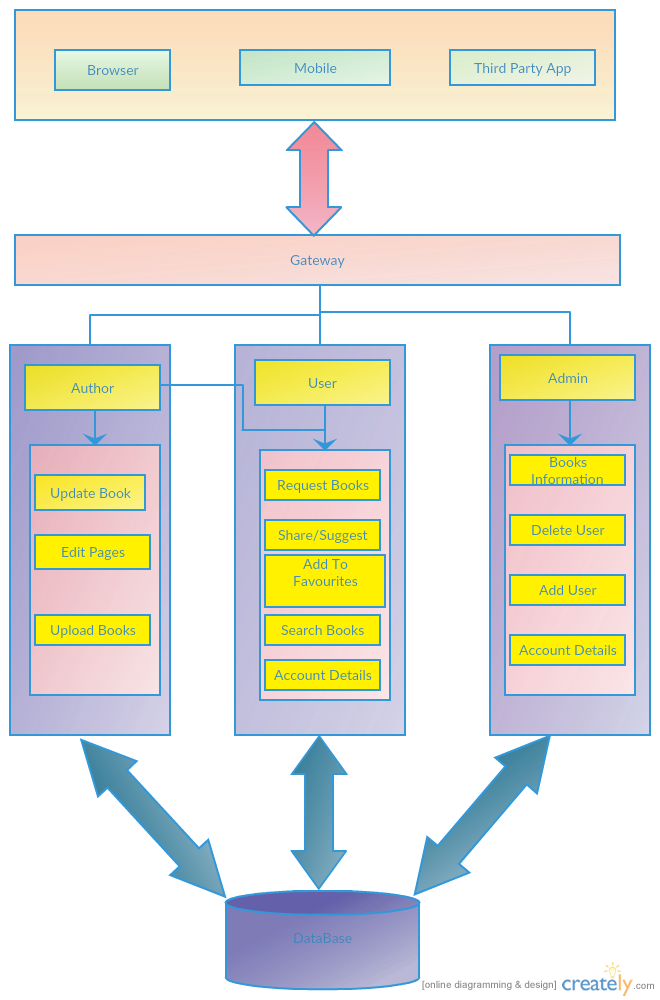
The application we designed is the cloud library. This application allows the users to upload the books, edit pages and share the books to other users. We can also save some books to the favorite books list and suggest them to other users.

We came up with an object oriented architecture style so that the each task of the application is an individual reusable and self-sufficient objects, each containing data and behavior related to it. Each process is considered as an individual reusable object. We designed our application based on open-close principle. So at any point of time we could easily extend the functionalities of the application without changing or modifying the code. We used data base to save the information of the users registered and details of the books availability. Our application allows the users to view the books availability and give access to other basic operation only if the user is registered. We have three main components in the architecture: Author, User and Admin. Each of them have different functionalities and different access levels.

We designed the class diagram based on the requirement discussed above. We had 5 main classes in our class diagram: Author, user, admin, library and database. We also designed the activity and sequence diagrams for this application. We had chosen 3 use cases for the sequence diagram: search books, request books, upload books. Each of them is discussed in detailed in the later part of the document.

We designed our application based on Abstract Factory design pattern. Our use case requires some functionalities to be abstracted. So we decided that abstract factory design pattern best suits our application. In our use case the author, admin and the user functionalities are hidden.

# **Architecture Diagram**



# **Description**

**User:** Upon entering the credentials, User contains the basic user information like Name, id and account information

**Author**: Only the authors of the books have access to this. On entering the credentials, the author could view his basic information.

**Admin:** Contains all users’ information and have access to add/delete user, Account information, Books information as in when the books come in.

**Upload:** The author can upload books.

**Update Book:** The author can edit pages in the book and can update the same.

**Request Book:** The user can request book which is not available currently.

**Edit Pages:** An Edit page is the one where only the authors have access to it. If any modifications are required in the book the author can edit the pages and update it.

**Search Books:** The users and the authors can search the books required, book it online and take the book for free of cost for some days as suggested by the librarians. We can search a book based on title, author name, book id, language.

**Share/Suggest Books:** The users and the authors can suggest or share a book to his/her friends.

**Add to Favourites:** The users can add few books to favourite list and view the availability of those books later.

**Account Details:** The account details consist of user information like any fines in his account, check in and check out of the book, notification of availability of a book that is previously added to favourites that is not available.

**Database:** The Database consists of the user data, Authors Data, Librarian Data and the Books data.

# **Architectural Style:**

An architectural style is building of structure based on various features. With time these architectures styles keep changing. We have chosen Object oriented architectural style for our project. The reason behind is that it reduces the maintenance cost, real world modelling is possible, code reusability, flexibility and reliable. In object oriented architectural style everything is considered as an object. In this architecture the tasks are divided for an application. Each task could be reused and contains the data and behaviour of objects related to it. Object oriented architecture is applied to structure based approach. The key principles of object oriented architecture include:

* **Abstraction**– This feature allows for reduce complexity of operation by generalizing them based on the characteristics.
* **Inheritance**– Each object can use the functionalities of other objects i.e. they can override the behaviour.
* **Decoupling –** By defining an abstract interface the objects can be decoupled in such a way that the consumer can understand.
* **Encapsulation –** The internal details of the objects can be hidden. This ensures security.
* **Composition –** Objects can be integrated with other objects and can hide that object details.
* **Polymorphism –** The functionalities can be reused in several objects.

Object Oriented Architectural Style

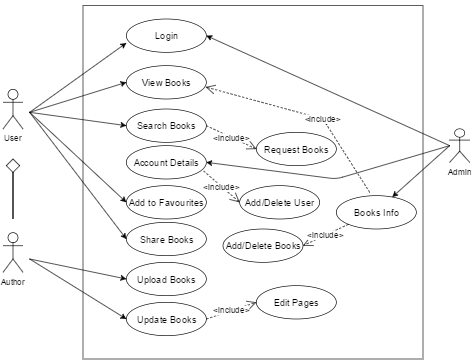
* **Modularity –**  The degree at which the system components are separated or combined
* **Re-usability –** Usage of existing functionalities
* **Open-Close principle –**Open to extend the functionalities but close to modify them i.e., the source could need not be changed but can extend the functionalities.

Tradeoffs:

* Objects need to identify the other objects they want to interact with and also should evaluate what if an object changes.
* Not all problems can be broken down into clearly defined objects.
* Strong coupling between super classes and sub classes.

# **UML Diagrams**

## **Use Case Diagram**



### **Description**

**User Functionalities:**

Firstly, the user enters the credentials. If entered credentials are right the user logs in to his page else will be asked to enter correct credentials. The user can view the books available with the librarian, search for required book by entering the book title or he could search by author. The user can view his account details by clicking the account details tab. If the user liked a particular book he could add that to his favourite book list and view the books at later point of time.

**Author Functionalities:**

The author first enters valid credentials and logs in to his page.by clicking on updated pages, he could search for a book to be updated and update the pages. After the author has updated the pages it will be automatically updated in the database. The author can upload a new book to the database.

## **Class Diagram**

Class diagram is used for describing structure and behaviour in the use cases, provides a conceptual model of the system in terms of entities and their relationships, used for requirement capture, end-user interaction. Detailed class diagrams are used for developers. A class diagram depicts classes and their interrelationships.

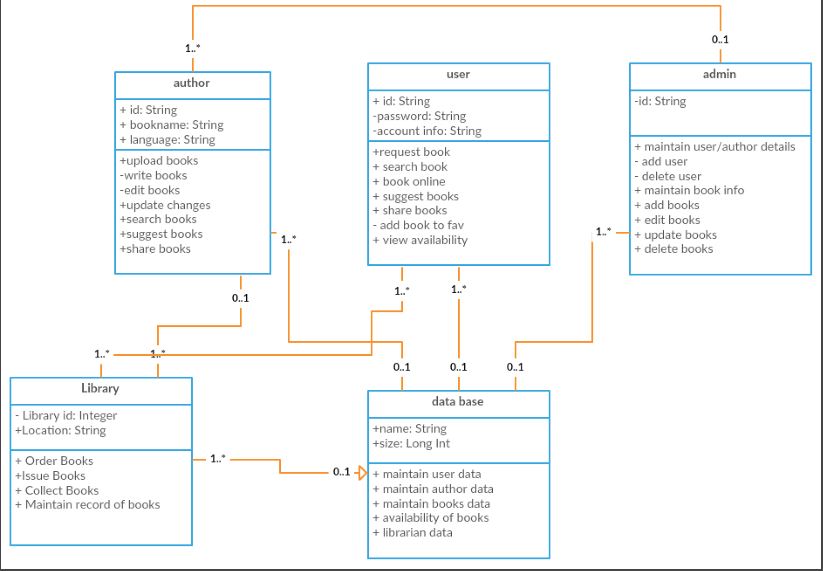
There are 4 classes

• Author

• User

• Admin

• Database



### **Description**

• **Author**

 Attributes:Author id, Author Name, Language.

Operations: Author uploads books, Writes books; makes any changes to the books uploaded (edit books). He also performs operations such as searching books, suggesting books, share books with other users.

• **User**

Attributes: User id, password, Account information.

Operations:User searches for books, shares books with his friends on cloud, requests for books that are not available in the cloud, book any books online, suggest books for other users, add books to favourites, and view availability of books in the cloud.

• **Admin**

Attributes: User information.

Operations: Maintain details of books, users, and authors. Admin also has the power to add users and delete users, adding books, deleting books and updating books is also done by the admin.

• **Database**

Attributes: Database id, Size.

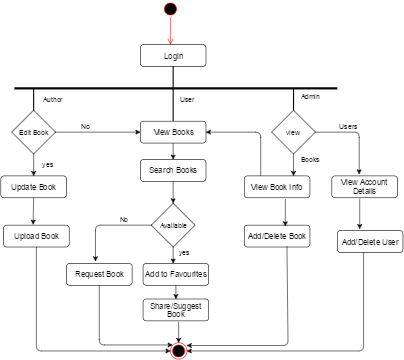
Operations: Database maintains the user data, maintains author data, availability of books is also maintained and updated in the database, maintains details of the libraries and books available in particular libraries.

• **Library**

Attributes: Location, Id

Operations: Library orders books, issues books and collects books from users, maintaining record of books in the database.

## **Activity Diagram**



### **Description**

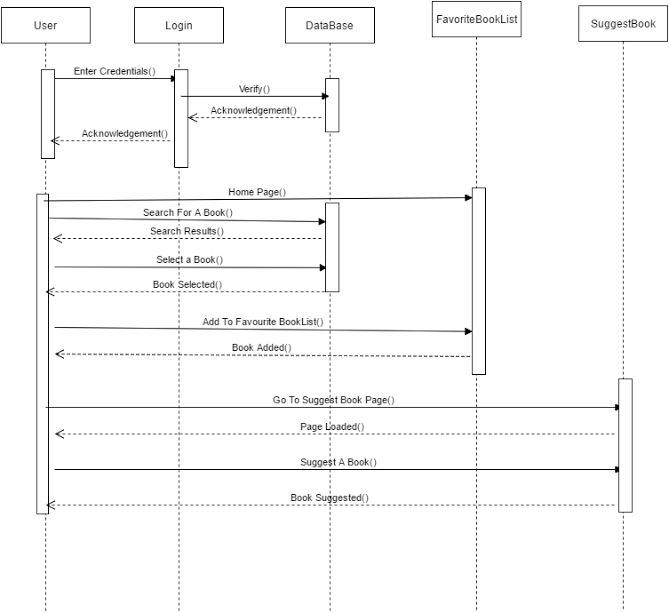
The author, user or the admin can login to their page by entering the valid credentials. The user can view the books available or search for the required book. If the book is available, he could request for issue else he could request for the book. He could suggest or share a book with his friends. And also he can add books to favorites for future reference.

The admin maintains all users account details either he can add/delete user, maintains all books information where he can also have access to add/delete books. As an user he can access books in library.

The author writes books and uploads them into cloud. He can also edit books and update them. He also can access all the books in library as a user.

## **Sequence Diagrams**

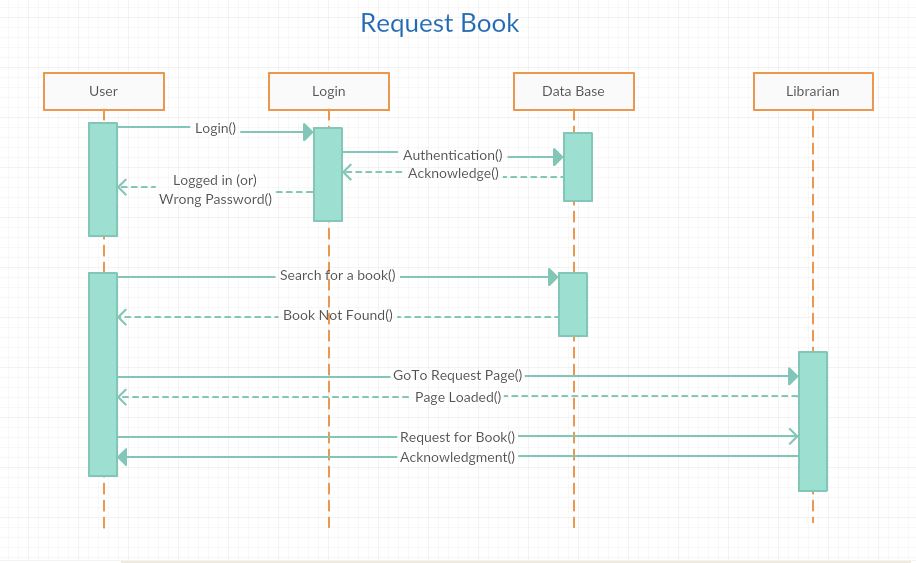
### **1. Suggest Books**



### **Description**

The user enters his credentials. The credentials are validated with the database and the user enters the home page if the credentials are correct else a message will be displayed saying that the entered credentials are wrong and enter correct credentials. The user searches for a particular book by entering the book title or the author name. The search will be done with the books available in the database and the search results are produced. Now, the user can add the selected book to favorite book list and view them later. The user can suggest a book to his friends.

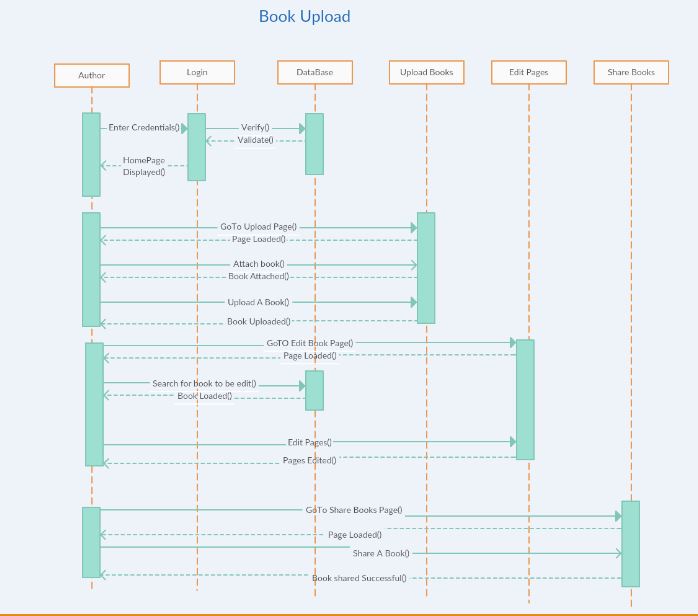
### **2. Request Book**



### **Description**

The user enters the credentials and these credentials entered are verified with the database. After entering valid credentials, the user can view the home page. Now, the user can search for a book and if the book is not available in the database, the user can request the librarian for the book.

### **3. Upload a Book**



### **Description**

The Author enters the credentials and it will be validated with the database. The user can view the home page if the credentials entered are correct. The Author can go to the upload book tab and attach the book that he wants to upload. He could upload the book by clicking on upload tab. Now, the author can edit the pages in the book if required. He needs to go to Edit Pages tab and search for the book that he wants to update. After selecting the book, he can edit the pages and the same will be updated in the database. The same book can be shared to his friends if needed by going to share books page.

# **Design Pattern**

## **Abstract factory Pattern**

The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes. This pattern separates the details of implementation of a set of objects from their general usage and relies on object composition, as object creation is implemented in methods exposed in the factory interface.

In this use case we abstract the pattern to only user, author and admin and the inner methods are hidden. Each user has their own functionalities which cannot be accessed by other as per the credentials that are entered. For example, the user has access to few functions, author has access to few functions and admin looks after few functions. Hence all the users are grouped as per their access permissions.

# **Team Contribution**

Initially we all worked on the design patterns, did a lot of research on it and finally came to a conclusion of which design pattern is apt for our project. We believed that Abstract Factory pattern best suits our application. We also studied about the architectural styles, made discussions on it and finalized that Object oriented design pattern is best for this application. Later, we worked on class diagrams, sequence diagrams, activity and uml diagrams and came up with the best. We all worked individually on each and every aspect of the project and then discussed on it later came to a conclusion of which idea is best for our application. The material given was also very helpful.

# **Conclusion**

Thus a use friendly application is designed to make things easy and better. The registered user can have the facility to check for the availability of the required book share it with the friends and can request for the book that is not available. The author has different set of privileges. He can edit the pages of the book, upload a new book to the users.

# **References**

https://msdn.microsoft.com/en-us/library/ee658117.aspx

<http://www.encyclopedia.com/doc/1O11-objectorientedarchitectur.html>

<http://www.dba-oracle.com/t_object_oriented_approach.htm>

<https://en.wikipedia.org/wiki/Architectural_style>

<https://en.wikipedia.org/wiki/Open/closed_principle>

<http://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/object-oriented-architecture>