**Software Requirements**

**Specification**

**for**

**BookConnect**

**Prepared by:**

**Usman Amin (22L-6916)**

**Muhammad Mateen (22L-6826)**

**Muhammad Mahad(22L-6747)**

**March 16, 2025**

**Table Of Contents**

**1.** Introduction 3

**1.1.** Purpose 3

**1.2.** Document Conventions 4

**1.3.** Intended Audience and Reading Suggestions 4

**1.4.** Product Scope 5

**2.** Overall Description 6

**2.1.** Product Perspective 6

**2.2.** Product Functions 7

**2.2.1.** User Management 7

**2.2.2.** Book Listings 7

**2.2.3.** Offer Management 8

**2.2.4.** Book Requests 8

**2.2.5.** Messaging System 9

**2.3.** User Characteristics 9

**2.4.** Constraints 10

**3.** External Interface Requirements 11

**3.1.** User Interfaces 11

**3.1.1.** Login and Registration 11

**3.1.2.** Book Listing Page 12

**3.1.3.** Offer Management 12

**3.1.4.** Messaging Interface 13

**3.2.** Hardware Interfaces 13

**3.3.** Software Interfaces 14

**3.4.** Communication Interfaces 15

**4.** System Features 16

**4.1.** Place an Offer for a Book 16

**4.2.** Place an Offer of Cash for a Book 17

**4.3.** Remove Offer 18

**4.4.** Alter or Change an Offer 19

**4.5.** Post a Book for Sale, Exchange, or Both 20

**4.6.** Remove a Book for Sale or Exchange 21

**4.7.** Post a Book Request (Book Needed) 22

**4.8.** Remove a Book Listing 23

**4.9.** Remove a Book Request 24

**4.10.** Notify User 25

**5.** Non-Functional Requirements 26

**5.1.** Performance Requirements 26

**5.2.** Scalability Requirements 27

**5.3.** Security Requirements 28

**5.4.** Usability Requirements 29

**5.5.** Reliability Requirements 30

**6.** Other Requirements 31

**7.** Appendices 32

**7.1.** Appendix A

**7.1.1.** Glossary 32

**7.2.** Appendix B

**7.2.1.** Analysis Models 33

**7.2.2.** Use-Case Descriptions 33

**7.2.3.** Sequence Diagrams 34

**7.2.4.** State Diagrams 36

**7.2.5.** Data Flow Diagrams 38

**7.2.6.** Class Diagram 39

**7. 3.** Appendix C

**7.3.1.** References 39

**1. Introduction**

**1.1 Purpose**

BookConnect is a system designed to facilitate book exchanges, sales, and requests between users. The system allows users to post books for sale or exchange, place offers for books, modify offers, and remove book listings or requests. This document outlines the functional and non-functional requirements of the system.

**1.2 Document Conventions**

1. UML notation is used for diagrams.
2. Functional requirements are prefixed with **FR**
3. Non-functional requirements are prefixed with **NFR**

**1.3 Intended Audience and Reading Suggestions**

This document is intended for developers, testers, project managers, and stakeholders involved in the development and deployment of BookConnect.

**1.4 Product Scope**

BookConnect aims to provide a seamless platform for book transactions through offers and requests. The system will allow users to:

1. List books for sale or exchange.
2. Place and manage offers.
3. Search and browse book listings.
4. Communicate with other users via messaging.

**2. Overall Description**

**2.1 Product Perspective**

BookConnect is an online platform that will be accessible via web and mobile applications. It will store book listings and user interactions in a centralized database.

**2.2 Product Functions**

1. User Management: User registration, login, and profile management.
2. Book Listings: Posting, updating, and removing books for sale or exchange.
3. Offers Management: Placing, modifying, and removing book or cash offers.
4. Book Request: Users can request books and offer an exchange or cash.
5. Messaging: Users can communicate regarding book transactions.

**2.3 User Characteristics**

Users of BookConnect will primarily be book enthusiasts, students, and collectors. Basic computer literacy is required.

**2.4 Constraints**

1. Users must be registered to post books or make offers.
2. Each book post must have a title, description, and price (if applicable).

**3. External Interface Requirements**

**3.1 User Interfaces**

* Login and Registration Page
* Book Listing Page
* Offer Management Page
* Messaging Interface

**3.2 Hardware Interfaces**

* Compatible with standard web browsers and Android/iOS devices.

**3.3 Software Interfaces**

* Database: MySQL or Firebase
* APIs: RESTful APIs for communication between frontend and backend.

**3.4 Communication Interfaces**

* Internal Messaging system
* Email Notifications
* REST API communication
* WebSocket for real-time updates
* Third-party payment gateway

**4. System Features**

**4.1 Place an Offer for a Book**

**4.1.1 Description and Priority:**

This use case allows a registered user to place an offer for a book, either by offering another book in exchange or by offering cash. This is a high-priority function as it is a core feature of the platform.

**4.1.2 Functional Requirements:**

* The system shall allow users to view book details before making an offer.
* The system shall provide options for offering another book or cash.
* The system shall validate and store the offer with the correct linkage.

**4.1.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC1 | |
| Name | Place an Offer for a Book | |
| Summary | The user selects a book and creates an offer by specifying if they are offering another book or cash. | |
| Priority | High | |
| Actors | User | |
| Pre-condition(s) | The user must be registered and logged in. | |
| Post-condition(s) | An offer is added to the system and linked to the respective book post. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | User selects a book from the system | System displays book details |
| 2 | User specifies if they are offering another book or cash. | System provides input fields for book details or cash amount. |
| 3 | User submits the offer. | System records the offer and links it to the respective post and user. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | User selects a book that is no longer available. | System displays an error message stating that the book is no longer listed. |
| 2 | User does not complete the required offer details. | System prompts the user to fill in all required fields. |
| 3 | User decides to cancel the offer before submission. | System discards the entered details and returns to the book page. |

**4.2 Place an Offer of Cash for a Book**

**4.2.1 Description and Priority:**

A user can offer cash for a book listed for sale. This feature is high priority since it facilitates transactions between users.

**4.2.2 Functional Requirements:**

* The system shall allow users to enter and validate cash offers.
* The system shall store and associate the cash offer with the book post.
* The system shall notify the seller about the new offer.

**4.2.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC2 | |
| Name | Place an Offer of Cash for a Book | |
| Summary | Allows a user to offer cash for a book listed in the system. | |
| Priority | High | |
| Actors | User | |
| Pre-condition(s) | The user must be logged in and the book must be listed in the system. | |
| Post-condition(s) | A cash offer is saved in the system. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user selects a book they want to buy.. | The system displays the book details. |
| 2 | The user enters the amount they are willing to offer. | The system records the cash offer. |
| 3 | The system links the offer to the book post. | The book owner is notified of the offer. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user attempts to place a cash offer without logging in. | The system prompts the user to log in. |
| 2 | The user enters an invalid cash amount (e.g., negative value). | The system displays an error message. |

**4.3 Remove an Offer**

**4.3.1 Description and Priority:**

A user can remove an offer they have previously placed. This function is of medium priority.

**4.3.2 Functional Requirements:**

* The system shall provide an interface to view active offers.
* The system shall allow users to remove their offers.
* The system shall remove the offer while maintaining system integrity.

**4.3.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC3 | |
| Name | Allows a user to remove an offer they previously made. | |
| Summary | The user selects a book and creates an offer by specifying if they are offering another book or cash. | |
| Priority | Medium | |
| Actors | User | |
| Pre-condition(s) | The user must have an existing offer. | |
| Post-condition(s) | The offer is no longer available in the system. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user navigates to their existing offers | The system displays a list of offers. |
| 2 | The user selects an offer and chooses the "Remove" option. | The system deletes the offer from the database. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to remove an offer that does not exist. | System displays an error message. |

**4.4 Alter or Change an Offer**

**4.4.1 Description and Priority:**

Users can modify their existing offers, changing the book being offered or the cash amount. This is a medium-priority feature.

**4.4.2 Functional Requirements:**

* The system shall allow users to modify their existing offers.
* The system shall validate the updated details before saving.
* The system shall notify the recipient of the modified offer.

**4.4.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC4 | |
| Name | Alter or Change an Offer | |
| Summary | Allows a user to update their offer by modifying the book being offered or the cash amount. | |
| Priority | Medium | |
| Actors | User | |
| Pre-condition(s) | The user must have an existing offer | |
| Post-condition(s) | The offer details are modified in the system. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user navigates to their offers. | The system displays a list of offers. |
| 2 | The user selects an offer and updates the book being offered or modifies the cash amount | The system updates the offer details accordingly. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to update an offer that has already been accepted by another user. | The system prevents the modification and displays an error message listed. |

**4.5 Post a Book for Sale, Exchange, or Both**

**4.5.1 Description and Priority:**

A user can list a book for sale, exchange, or both. This is a high-priority function.

**4.5.2 Functional Requirements:**

* The system shall allow users to post books for sale, exchange, or both.
* The system shall categorize the book listing correctly.
* The system shall store and display book listings effectively.

**4.5.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC5 | |
| Name | Post a Book for Sale, Exchange, or Both | |
| Summary | Allows a user to list a book for sale, exchange, or both. | |
| Priority | High | |
| Actors | User | |
| Pre-condition(s) | The user must be logged in. | |
| Post-condition(s) | A new book listing is available in the system. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user creates a new post. | The system prompts for book details. |
| 2 | The user specifies whether the book is for sale, exchange, or both. | The system saves the post and categorizes it accordingly. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to post a book without providing necessary details | The system displays an error message and prompts for completion. |

**4.6 Remove a Book for Sale or Exchange**

**4.6.1 Description and Priority:**

A user can remove their active book listing. This is a medium-priority feature.

**4.6.2 Functional Requirements:**

* The system shall provide an interface to manage book listings.
* The system shall allow users to remove their book posts.
* The system shall notify affected users about the removal.

**4.6.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC6 | |
| Name | Remove a Book for Sale or Exchange | |
| Summary | Allows a user to remove a book post from the system. | |
| Priority | Medium | |
| Actors | User | |
| Pre-condition(s) | The user must have an active book post. | |
| Post-condition(s) | The book listing is removed. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user navigates to their book listings. | The system displays the list of books posted by the user. |
| 2 | The user selects a book post and removes it. | The system deletes the post but retains offers made by other users. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to remove a book post that has active offers. | The system warns the user and asks for confirmation. |

**4.7 Post a Book Requirement (Book Needed)**

**4.7.1 Description and Priority:**

A user can post a request for a book they need. This is a medium-priority function.

**4.7.2 Functional Requirements:**

* The system shall allow users to request books they need.
* The system shall support multiple exchange options.
* The system shall store and display book requests effectively.

**4.7.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC7 | |
| Name | Post a Book Requirement (Book Needed) | |
| Summary | Allows a user to request a book they need and specify how they will acquire it. | |
| Priority | High | |
| Actors | User | |
| Pre-condition(s) | The user must be logged in. | |
| Post-condition(s) | The book request is available in the system. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user creates a request for a book they need. | The system prompts for request details. |
| 2 | The user specifies if they will offer another book, cash, or both in exchange. | The system saves the request. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to post a request without providing required details. | The system displays an error message. |

**4.8 Remove a Book Listing**

**4.8.1 Description and Priority:**

A user can remove an active book listing. This is a medium-priority function.

**4.8.2 Functional Requirements:**

* The system shall provide an interface to manage listings.
* The system shall allow users to remove book posts.
* The system shall ensure that book removal is properly handled

**4.8.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC8 | |
| Name | Remove a Book Listing | |
| Summary | Allows a user to remove their book listing from the system. | |
| Priority | Medium | |
| Actors | User | |
| Pre-condition(s) | The user must have an active book listing. | |
| Post-condition(s) | The book listing is removed. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user navigates to their listings. | The system displays the user's active listings. |
| 2 | The user selects a listing and removes it. | The system deletes the listing. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to remove a listing that has active offers. | The system prompts the user for confirmation before deletion. |

**4.9 Remove a Book Requirement**

**4.9.1 Description and Priority:**

A user can remove their posted book request. This is a medium-priority function.

**4.9.2 Functional Requirements:**

* The system shall provide an interface to view and manage book requests.
* The system shall allow users to remove their requests.
* The system shall ensure that request removal is properly handled.

**4.9.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC9 | |
| Name | Remove a Book Requirement | |
| Summary | Allows a user to remove their request for a book. | |
| Priority | Medium | |
| Actors | User | |
| Pre-condition(s) | The user must have an active book request. | |
| Post-condition(s) | The book requirement is no longer listed. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user navigates to their book requests. | The system displays the user's book requests. |
| 2 | The user selects a request and removes it. | The system deletes the request from the database. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The user tries to remove a request that has received offers. | The system warns the user and asks for confirmation before deletion. |

**4.10 Notify User**

**4.10.1 Description and Priority:**

The user is notified through a notification when another user places an offer on their book post. This is a High-priority function.

**4.10.2 Functional Requirements:**

* The system shall record all offers placed on book posts, including details such as the offer type (cash or book exchange), the user who placed the offer, and the timestamp.
* The system shall automatically send a notification to the post owner whenever a new offer is placed on their book post, provided that notifications are enabled in their settings.
* The system shall allow users to enable or disable notifications for new offers on their posts through their account settings.

**4.10.3 Use-Case Description Table:**

|  |  |  |
| --- | --- | --- |
| Identifier | UC10 | |
| Name | Notify User | |
| Summary | Notifies a user when another user places an offer on their book post. | |
| Priority | High | |
| Actors | User | |
| Pre-condition(s) | The post must exist in the system, and notifications must be enabled. | |
| Post-condition(s) | The user is notified of the new offer, or no notification is sent if settings disable alerts. | |
| Typical Course of Action | | |
| S# | Actor Action | System Response |
| 1 | Another user places an offer on a book post. | The system records the offer in the database. |
| 2 | The system checks if notifications are enabled for the post owner. | If enabled, the system triggers a notification. |
| 3 | The user receives a notification about the new offer. | The user acknowledges the notification. |
| Alternate Course of Action | | |
| S# | Actor Action | System Response |
| 1 | The post owner has disabled notifications. | The system does not send a notification and logs the offer silently. |

**5. Non-Functional Requirements**

**5.1 Performance Requirements**

* The system shall respond to user requests within 3 seconds for general operations.
* Book search queries shall return results within 5 seconds, even with high database loads.
* The system shall support at least 10,000 concurrent users without performance degradation.
* The system shall efficiently handle at least 100 new book listings per minute.
* The API should process 500 requests per second without failure.

**5.2 Scalability Requirements**

* The system shall be designed to scale horizontally and vertically based on demand.
* Cloud-based infrastructure should be used to accommodate sudden traffic spikes.
* The database should support partitioning and indexing for efficient searches.
* The system should support expansion to international markets without major modifications.
* Load balancers shall be implemented to distribute requests evenly across servers.

**5.3 Security Requirements**

* All sensitive user data shall be encrypted using AES-256 in storage and transmission.
* User authentication shall be handled using OAuth 2.0 or JWT tokens.
* The system shall automatically log out inactive users after 15 minutes of inactivity.
* Users shall be notified via email or SMS when their account settings are modified.
* The system shall be protected against SQL injection, XSS, and CSRF attacks.

**5.4 Usability Requirements**

* The UI shall follow a minimalist and intuitive design, making navigation easy.
* The platform shall be accessible to people with disabilities, following WCAG 2.1 guidelines.
* Tutorials and tooltips shall be available to help new users understand features.
* Users shall be able to complete common actions (e.g., posting a book) in three clicks or less.
* A mobile-friendly design shall ensure usability on both Android and iOS devices.

**5.5 Reliability Requirements**

* The system shall maintain an uptime of 99.9% annually.
* Data backups shall be performed daily, with a retention period of 30 days.
* If a system failure occurs, automatic failover mechanisms shall ensure continued operation.
* The system shall recover from a crash within 5 minutes.
* The error logging mechanism shall capture and report critical failures automatically.

**6. Other Requirements**

* System should comply with GDPR for data protection.
* A user-friendly UI with a minimal learning curve.

**Appendix A: Glossary**

GDPR: General data protection regulation

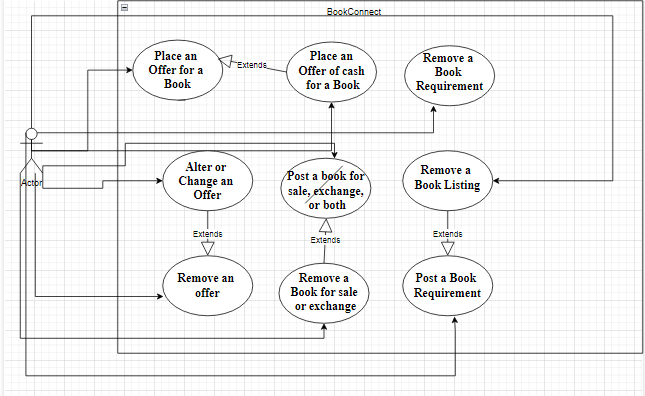
UI: User Interface

REST: Representational State Transfer

API: Application Programming Interface

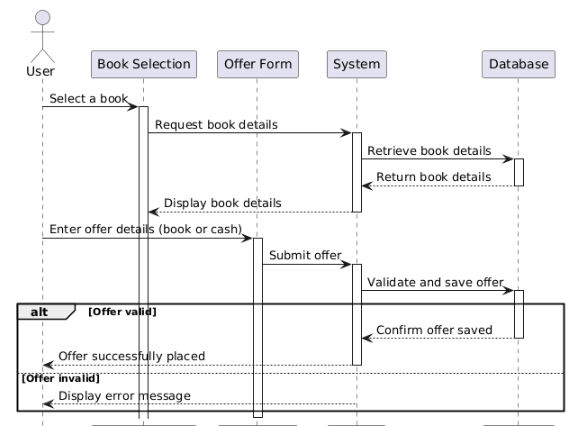
**Appendix B: Analysis Models**

**Use-Case Description:**

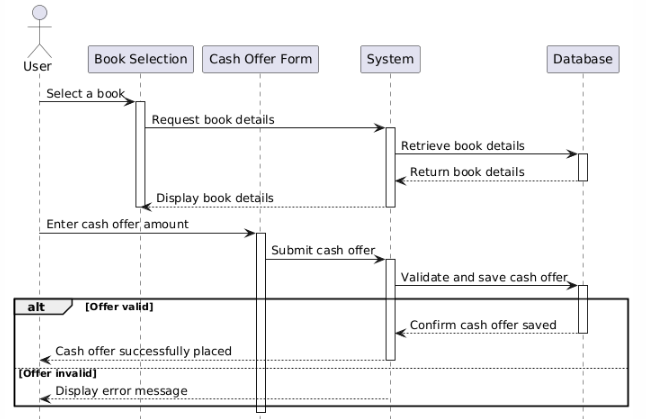
****

**Sequence Diagrams:**

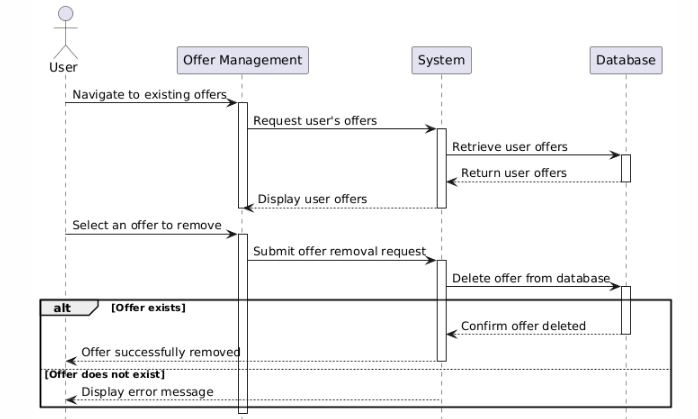
1. **Place an Offer for A Book**

****

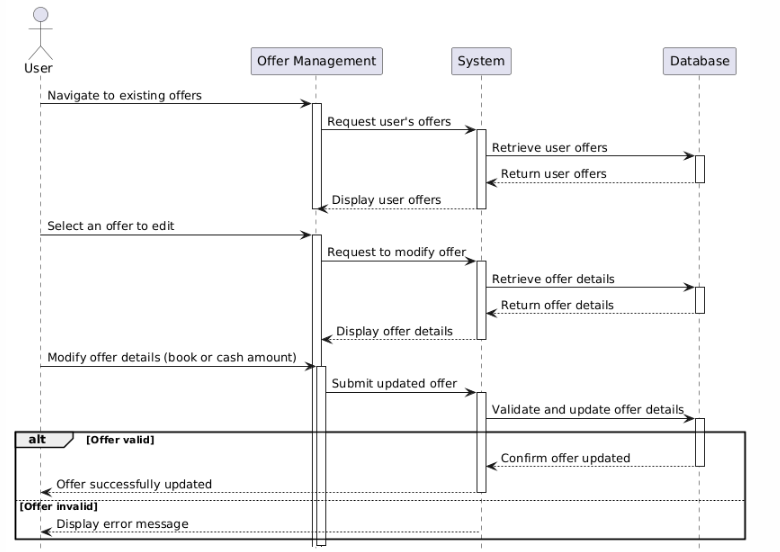
1. **Place an Offer of Cash for a Book**

****

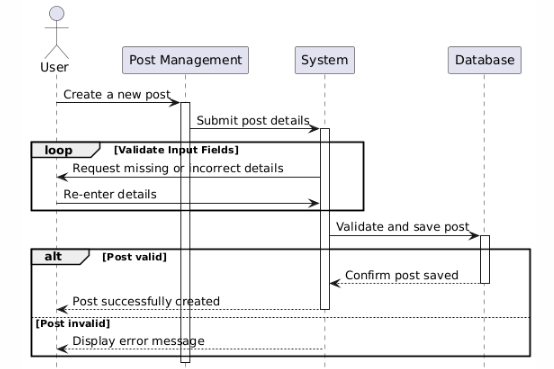
1. **Remove an Offer**

****

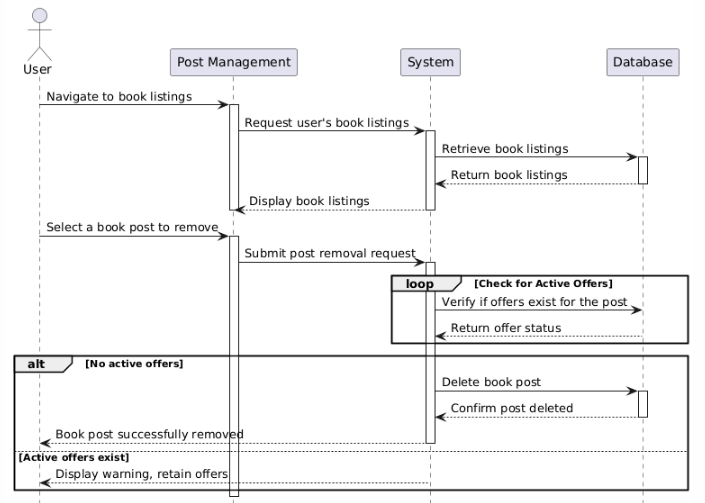
1. **Alter or Change an Offer**

****

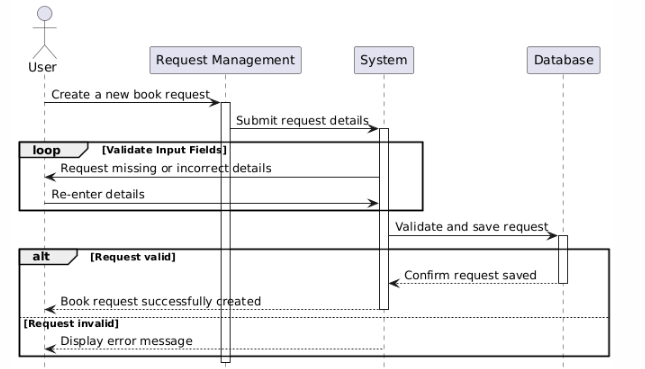
1. **Post a Book for Sale, Exchange, or Both**

****

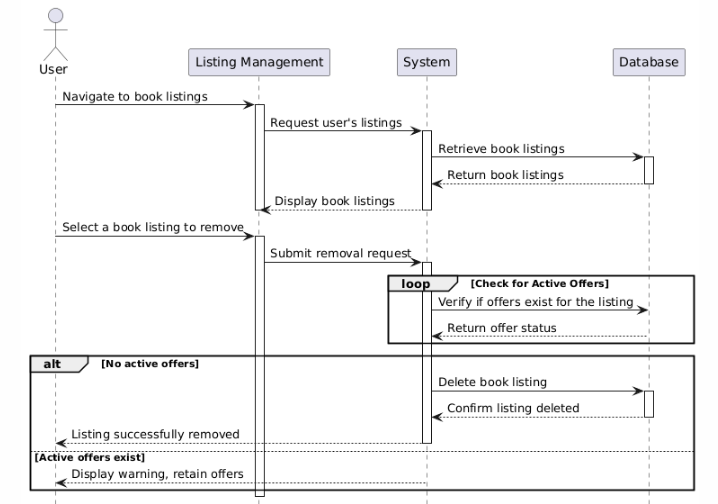
1. **Remove a Book for Sale or Exchange**

****

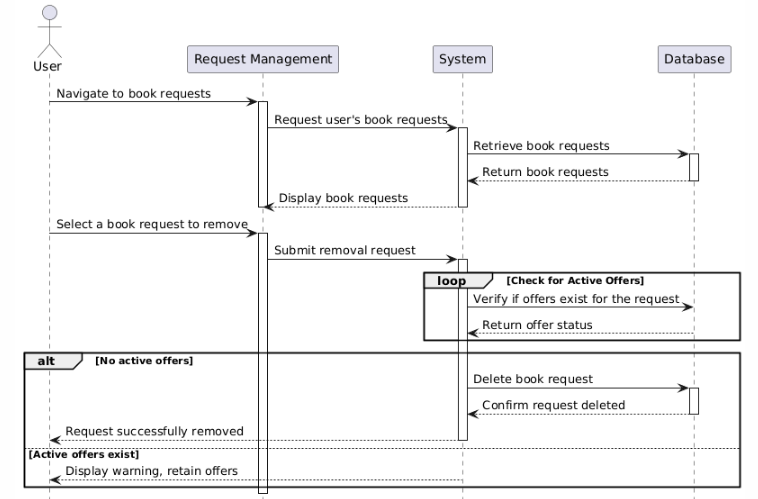
1. **Post a Book Requirement (Book Needed)**

****

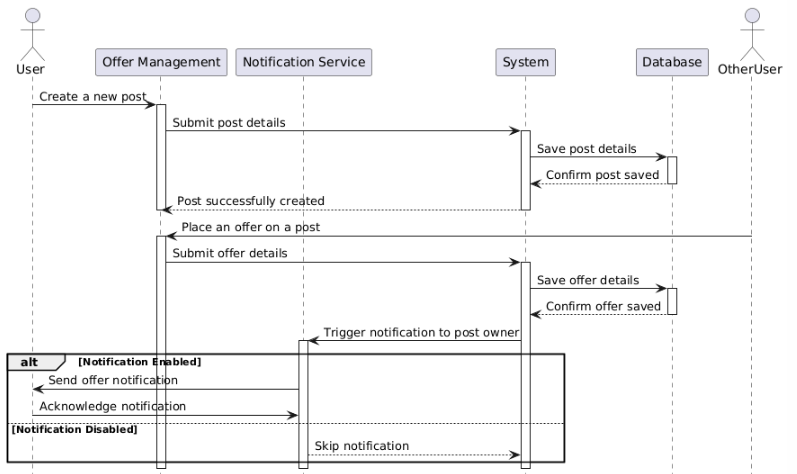
1. **Remove a Book Listing**

****

1. **Remove a Book Requirement**

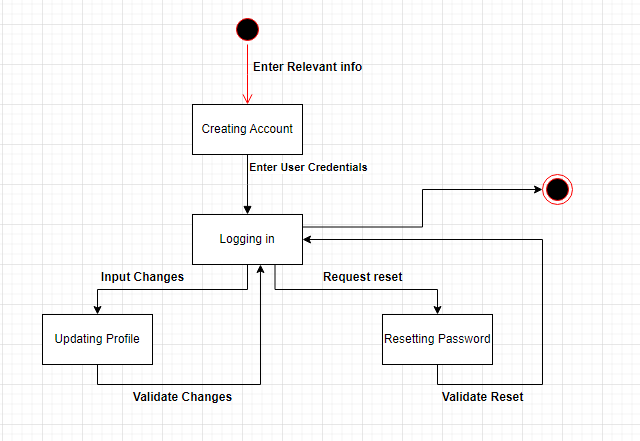
****

1. **Notify User**

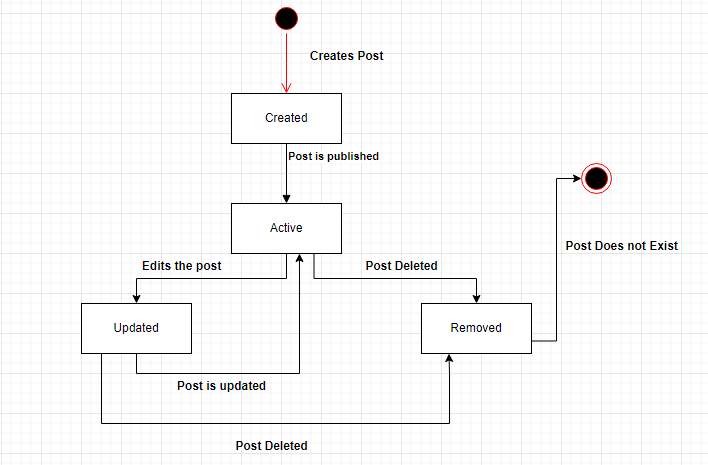
****

**State Diagrams:**

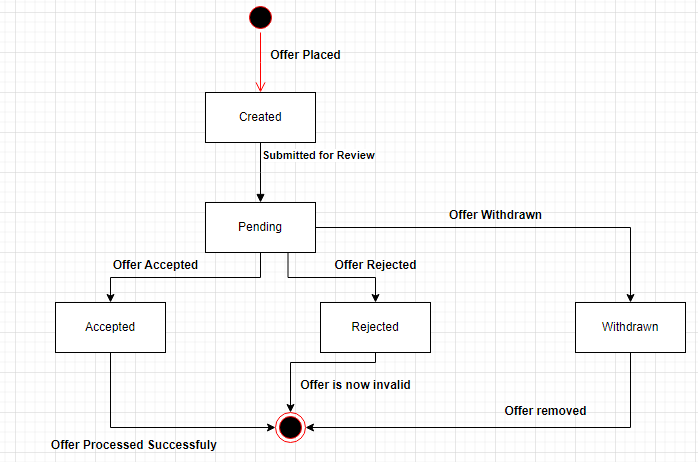
1. **User Registration**

****

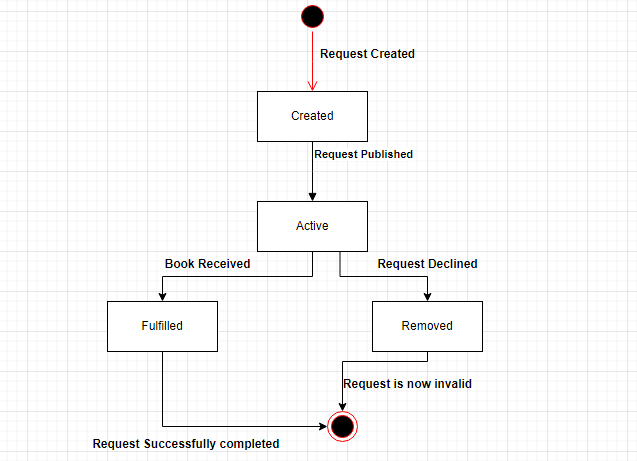
1. **Post Management**

****

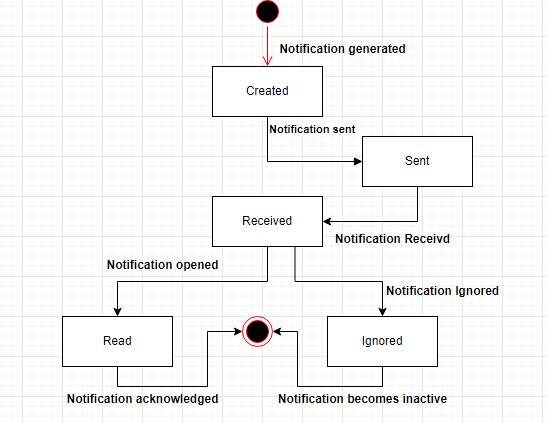
1. **Offer Management**

****

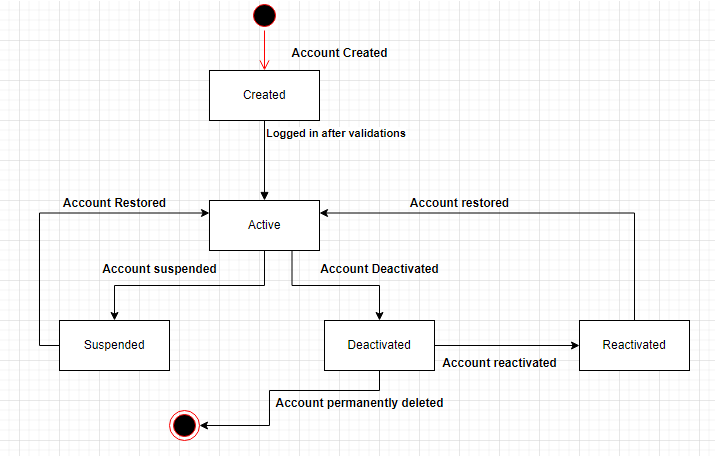
1. **Book Request Management**

****

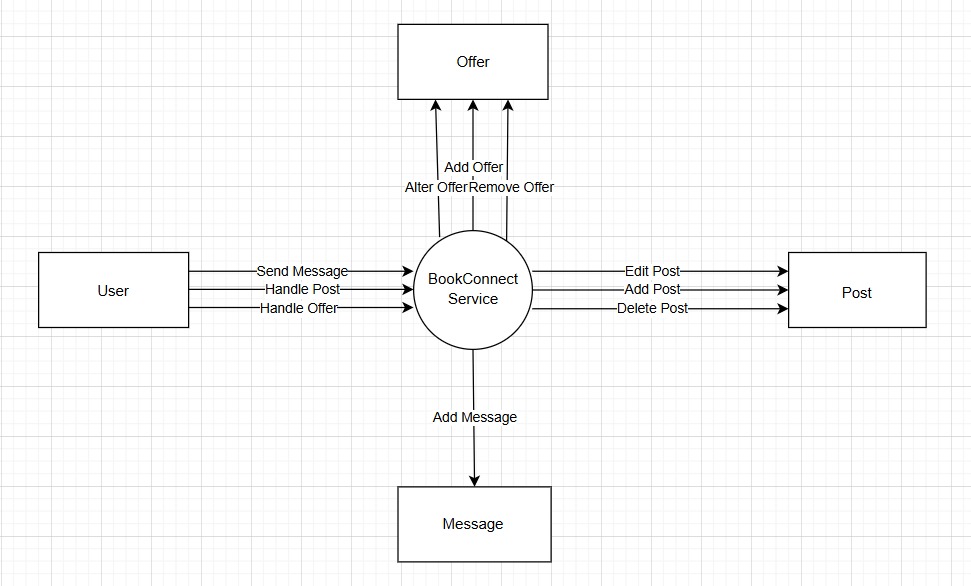
1. **Notification Management**

****

1. **User Account Management**

****

**Level 0 Data Flow Diagram:**

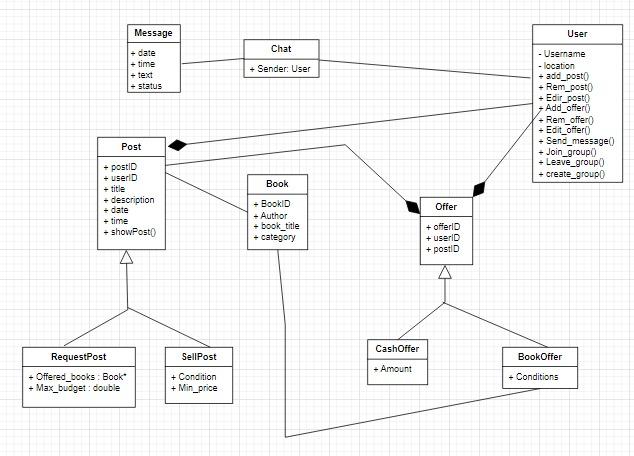
****

**Level 1 Data Flow Diagram:**

**A diagram of a service

AI-generated content may be incorrect.**

**Class Diagram:**

****

**Appendix C:**

**References**

1. <https://app.diagrams.net/?src=about>
2. <https://www.visual-paradigm.com/features/uml-tool/>
3. <https://www.lucidchart.com/pages/examples/uml_diagram_tool>
4. <https://www.visual-paradigm.com/solution/freeumltool/>
5. https://www.geeksforgeeks.org/top-7-uml-diagram-tools-that-you-can-consider/
6. GCR Slides