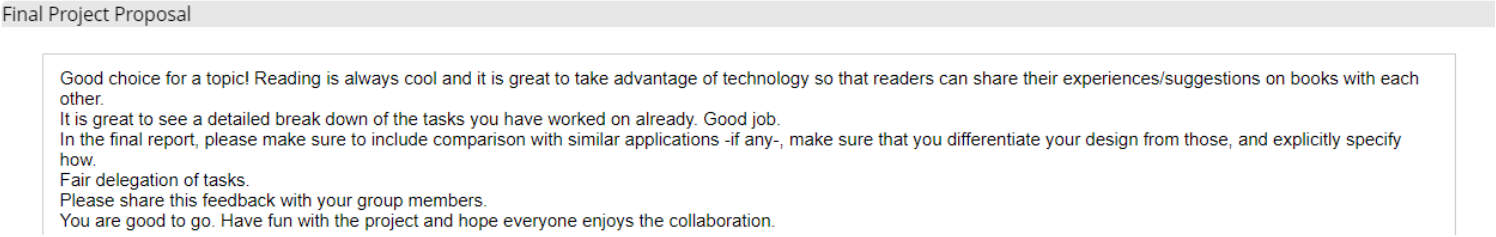
CS3354 Software Engineering
Final Project Deliverable 1

Book Shelf Software

Brady Cash, Christian Flores, Emmanuel Samuel, AJ Prosser

1. [5 POINTS]



Final Project Proposal

Good choice for a topic! Reading is always cool and it is great to take advantage of technology so that readers can share their experiences/suggestions on books with each other.
It is great to see a detailed break down of the tasks you have worked on already. Good job.
In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.
Fair delegation of tasks.
Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

The only request was to add a comparison to similar applications in our final report. We will do this in our final report by adding a comparison to Apple Books and Kindle, which are our two biggest competitors. We will compare and contrast the different capabilities and functions of each.

2. [10 POINTS]

.

Repository URL:

https://github.com/exs210034/3354-Book-Shelf-Software

3. [5 POINTS]

Task Delegation:

- Setting up Github repository:
  - Task 1.1 - (Everyone)
  - Task 1.2 - Emmanuel
  - Task 1.3 - Emmanuel
  - Task 1.4 - Christian
  - Task 1.5 - AJ
  - Task 1.6 -

- Diagrams:
  - Use Case diagram- Christian
  - Sequence diagram- AJ and Emmanuel
  - Class diagram- Brady

- Architectural designs:
  - Repository architecture pattern- Brady

- 4: Which software process model and why- Christian

- 5: Functional requirements - AJ

- 6: Non-functional requirements- Christian

4. [5 POINTS]

Which software process model is employed in the project and why?

We are using a prototyping model for this project because users of the product will be able to experience the product way before we have to release a final product. This means we will be able to gather feedback from the stakeholders in our project, so that we can ensure that our product will be viable and will fulfill the needs of our clients. This also means that our developers will gather immediate feedback from users which ensures that quick changes can be made to better suit our functional and non-functional requirements. Furthermore, since our product is a book shelf, the main functionality, reading the books, can be created from the start, and new features such as bookmarks and annotations can be added in in a consistent time frame, which enables our developers to concentrate on fixing the bugs for one feature at a time instead of having to fix bugs for multiple features at a time.

5. **[15 POINTS]** Software Requirements including

5.a.) [5 POINTS]

Functional requirements.

1. User should be able to load and delete books
2. User should be able to create and maintain categories for books
3. User should be able to read books, including swiping to change pages
4. User should be able to search for words in the text
5. User should be able to add bookmarks
6. User should be able to modify the visual appearance of a book, including a day/night mode, and the ability to zoom in and out
7. User should be able to add, edit, or remove notations, with the additional ability to see a list of all notations on a page and for a whole book.

5.b.) [10 POINTS]

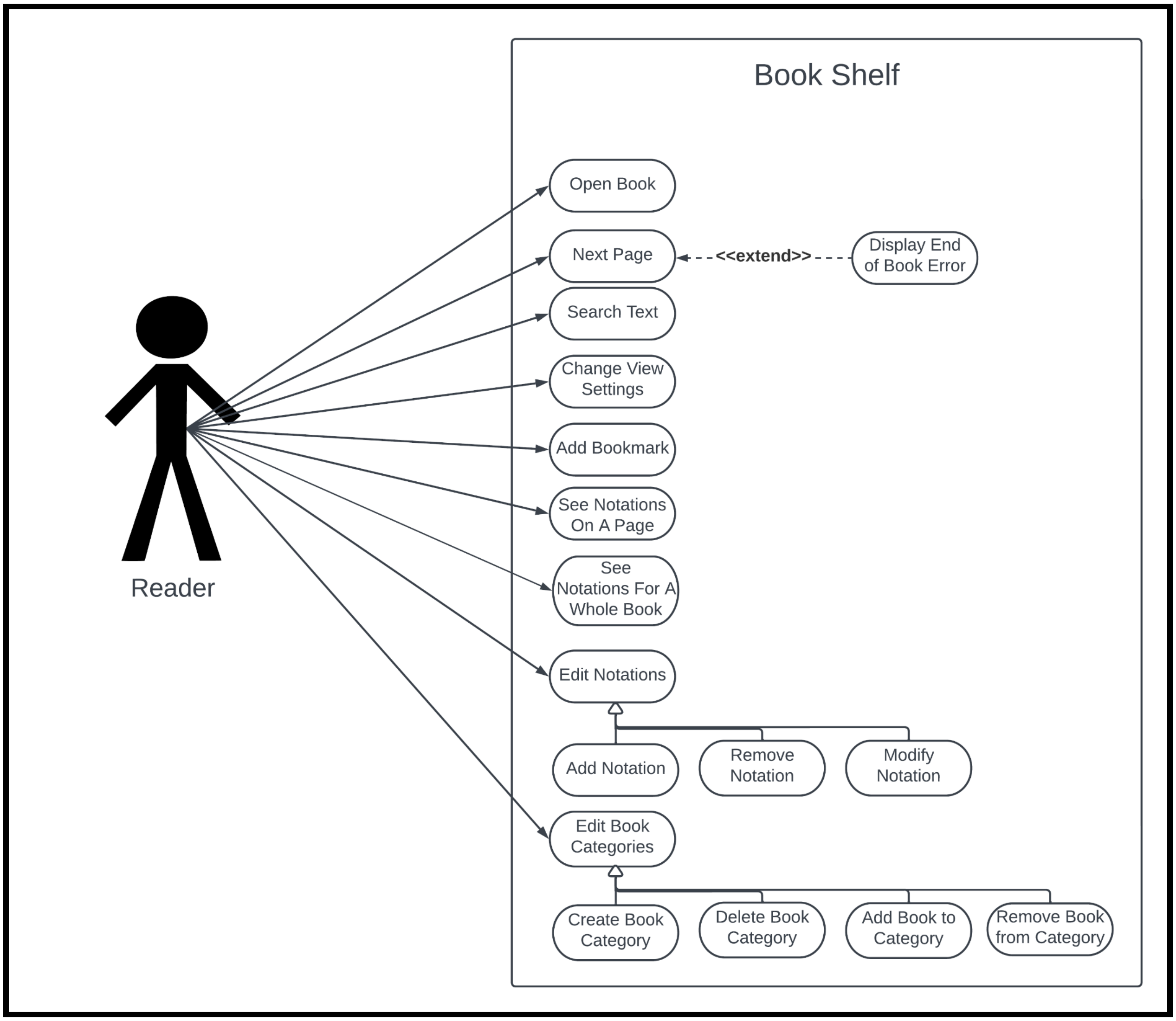
Non-functional requirements

1. **Usability requirement:** The software should support English, Spanish, and Chinese languages.
2. **Performance requirement:** It should not take more than 5 seconds to load a book.
3. **Space requirement:** The software should not be larger than 10 GB.
4. **Dependability requirement:** There should be multiple clouder servers that have backed up information for users to offer redundancy if one of the servers fails.
5. **Security requirement:** It should require the user to set up 2-factor authentication.
6. **Environmental requirement:** The servers that we use to store user data should run purely on renewable electricity.
7. **Operational requirement:** The basic local functionalities of the software should be able to continue to run when internet connection is lost.
8. **Development requirement:** There should be an update to the software once a month to fix any reported bugs or issues.
9. **Regulatory requirement:** The software should require age verification before allowing children to read explicit books. (Assuming there is a regulation for showing explicit material to children)
10. **Ethical requirement:** User data should not be shared with any third-parties without the user’s authorization.

11. **Accounting requirement:** All profits made by the sale of the software should be reported to the IRS. (Assuming the IRS requires us to pay taxes on our companies profits)
12. **Safety/security requirement:** Development of the software should not be offloaded to any companies or people who are based in any of the United State's enemy countries in order to ensure security of our software. (Assuming there is in a law restricting us from doing business with enemy countries for security purposes)

6. [15 POINTS]

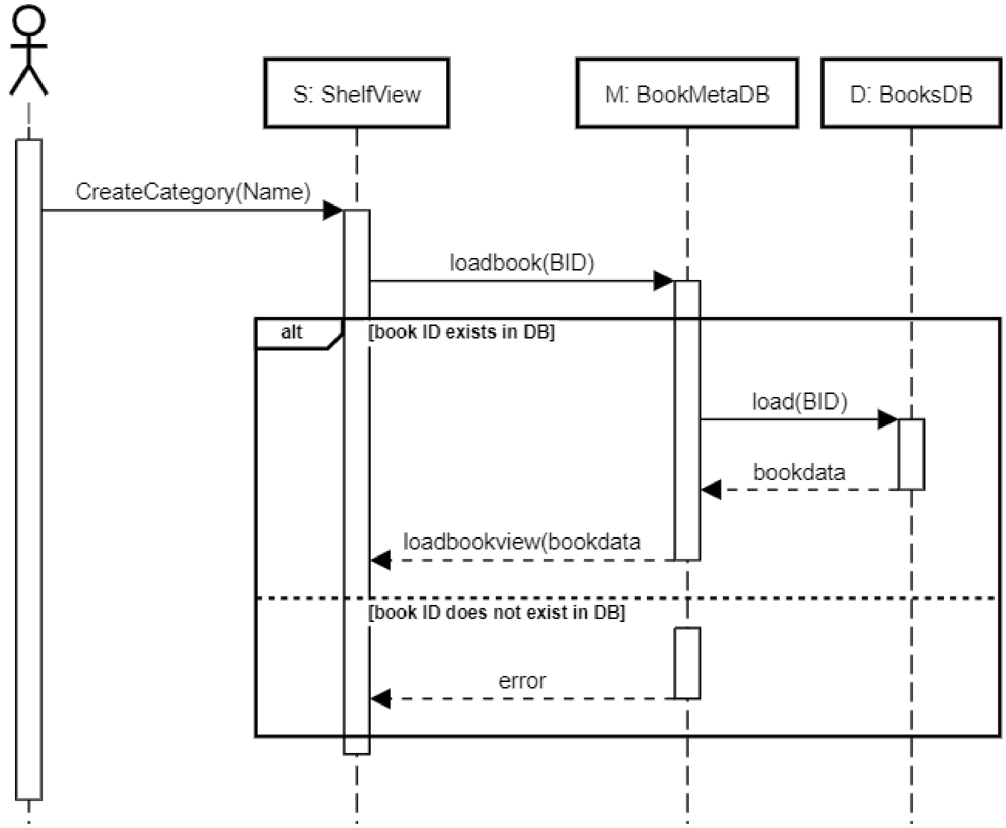
Use case diagram



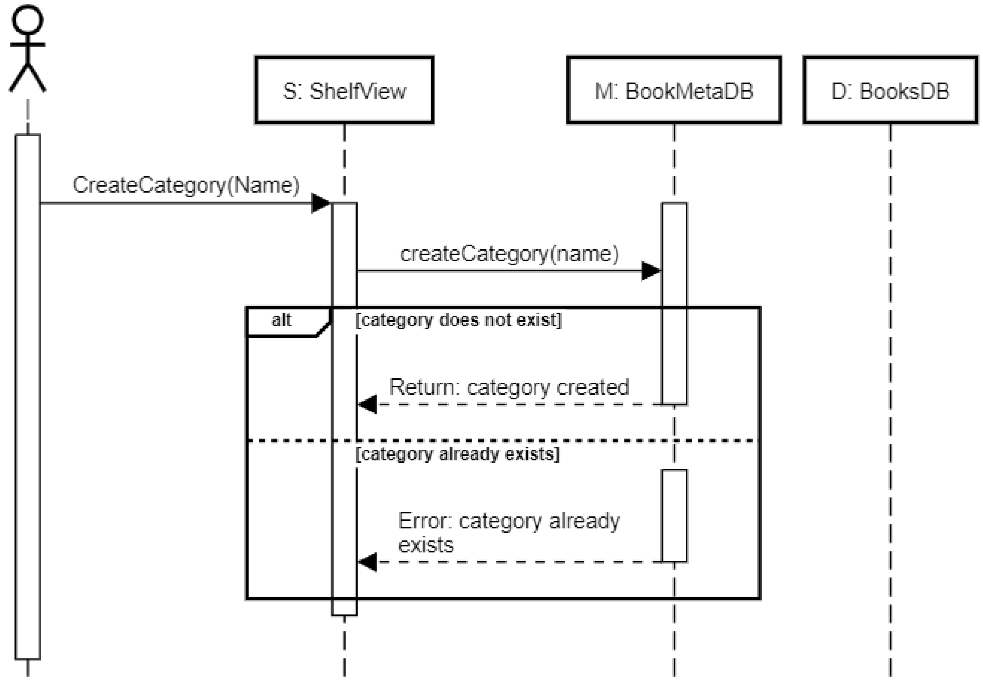
7. [15 POINTS]

Sequence diagram by use case:

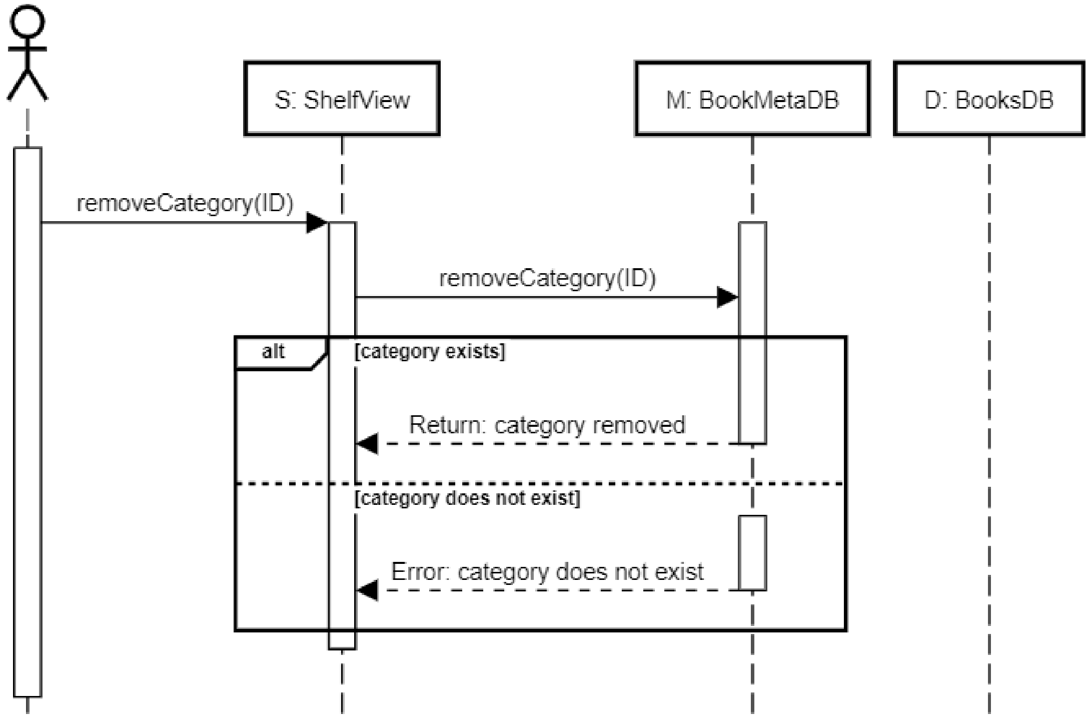
Open Book



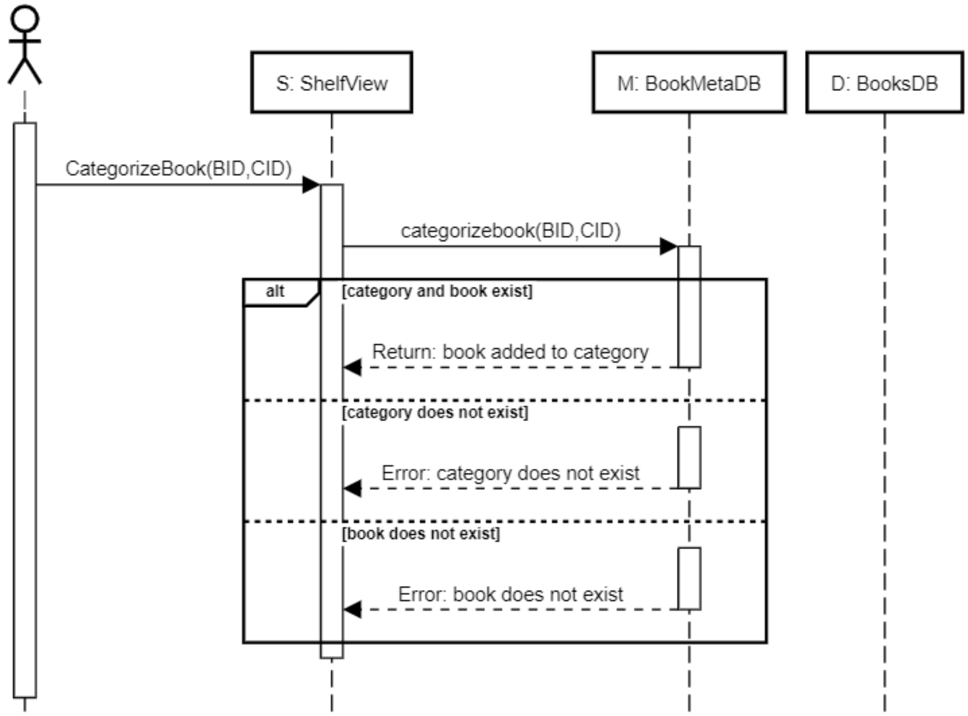
Create Book Category



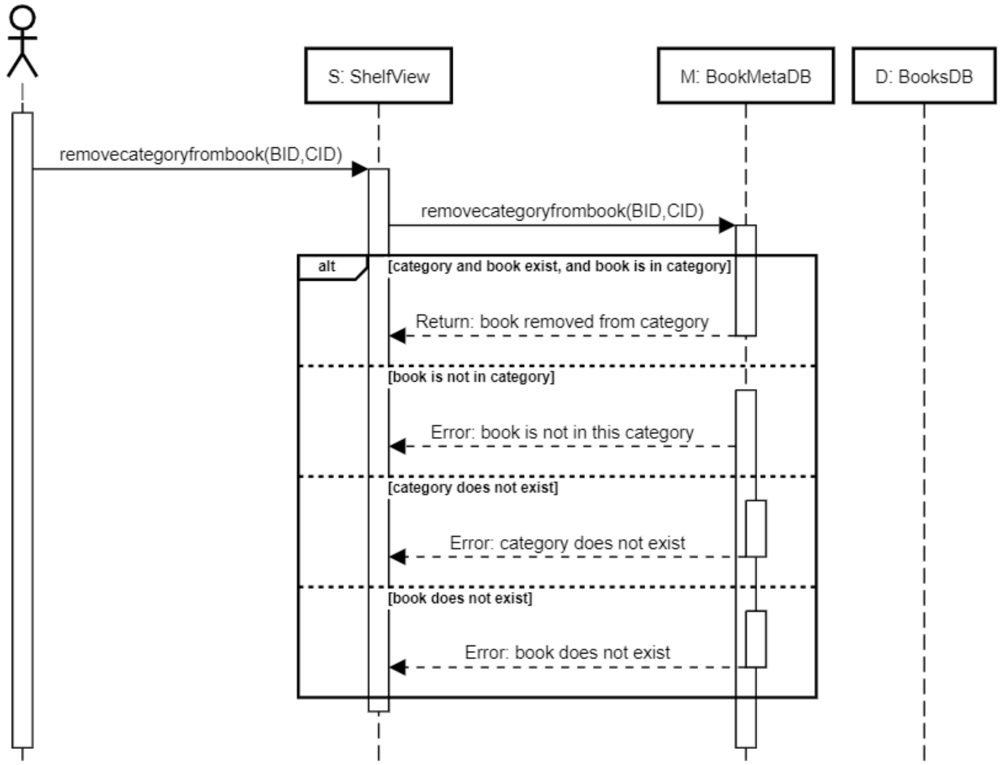
Delete Book Category



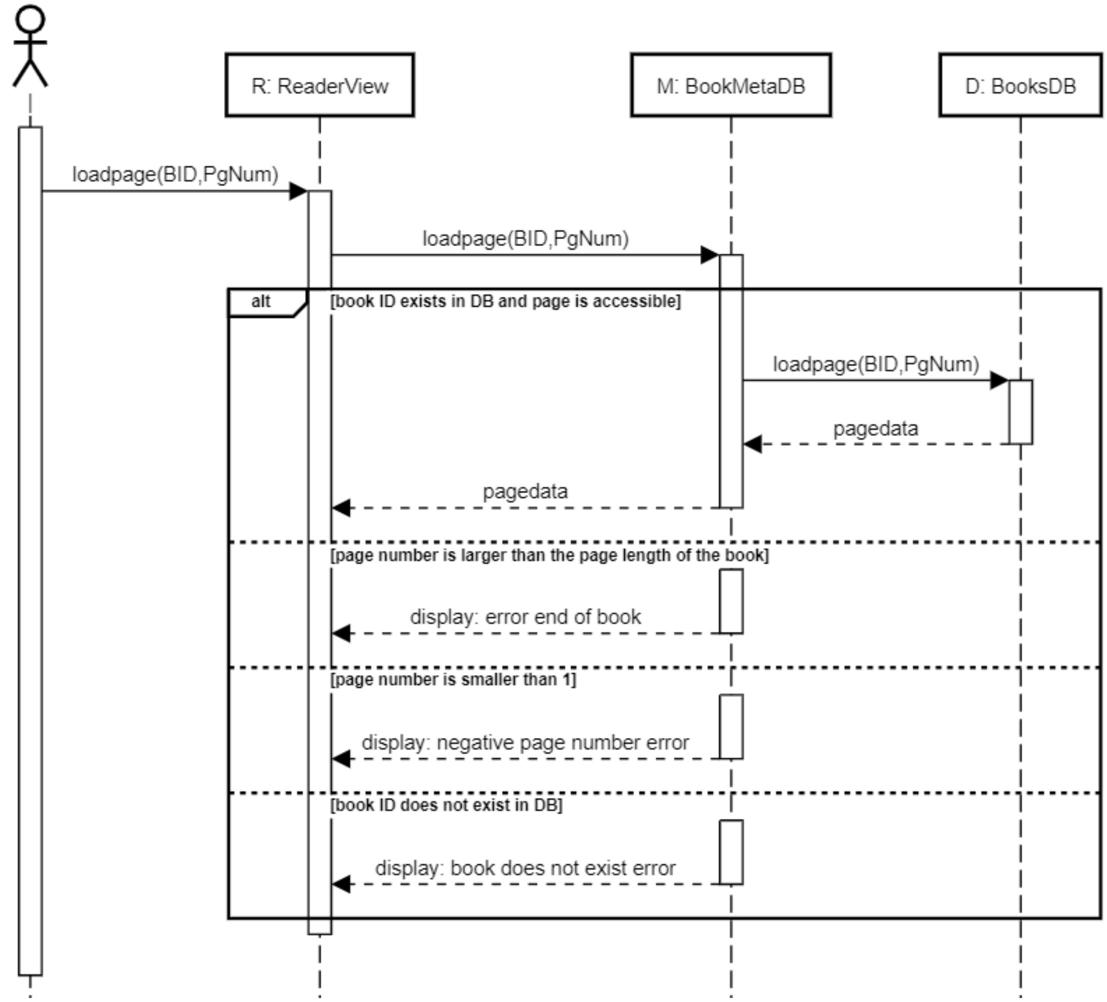
Add book to Category


S: ShelfView
M: BookMetaDB
D: BooksDB
CategorizeBook(BID,CID)
categorizebook(BID,CID)
alt
[category and book exist]
Return: book added to category
[category does not exist]
Error: category does not exist
[book does not exist]
Error: book does not exist

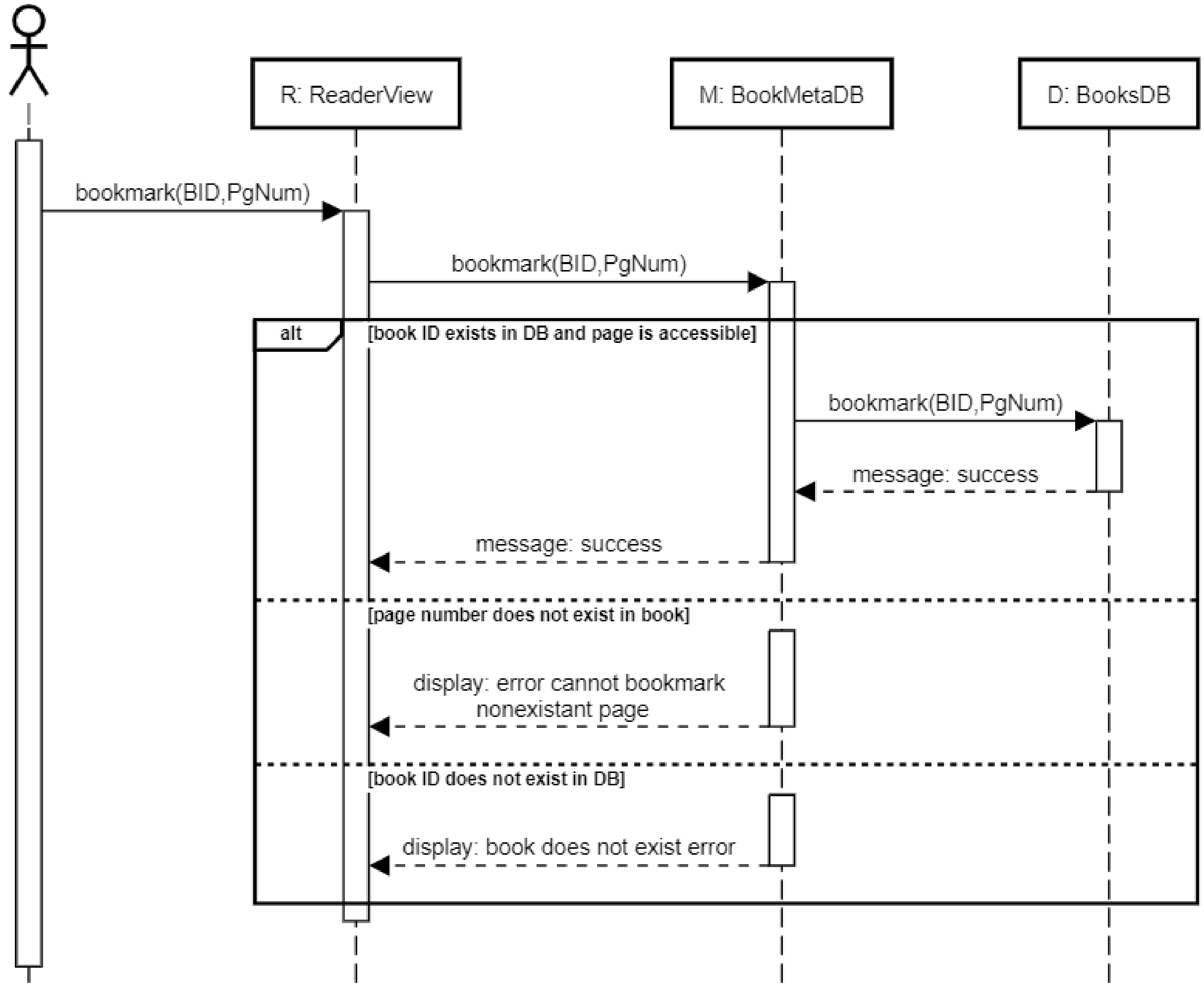
Remove book from Category


S: ShelfView
M: BookMetaDB
D: BooksDB
removecategoryfrombook(BID,CID)
removecategoryfrombook(BID,CID)
alt
[category and book exist, and book is in category]
Return: book removed from category
[book is not in category]
Error: book is not in this category
[category does not exist]
Error: category does not exist
[book does not exist]
Error: book does not exist

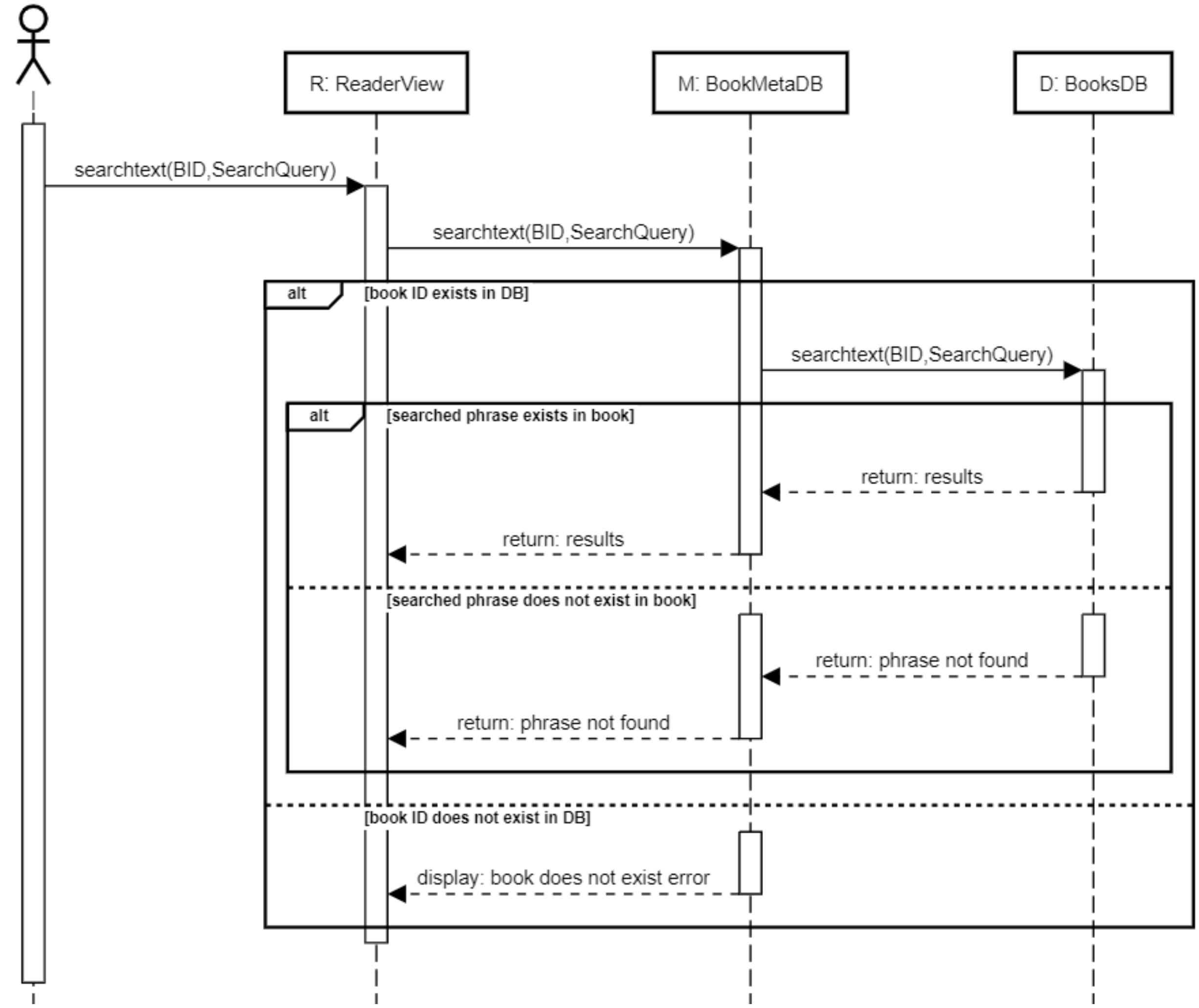
Next Page


R: ReaderView
M: BookMetaDB
D: BooksDB
loadpage(BID,PgNum)
loadpage(BID,PgNum)
alt
[book ID exists in DB and page is accessible]
loadpage(BID,PgNum)
pagedata
pagedata
[page number is larger than the page length of the book]
display: error end of book
[page number is smaller than 1]
display: negative page number error
[book ID does not exist in DB]
display: book does not exist error

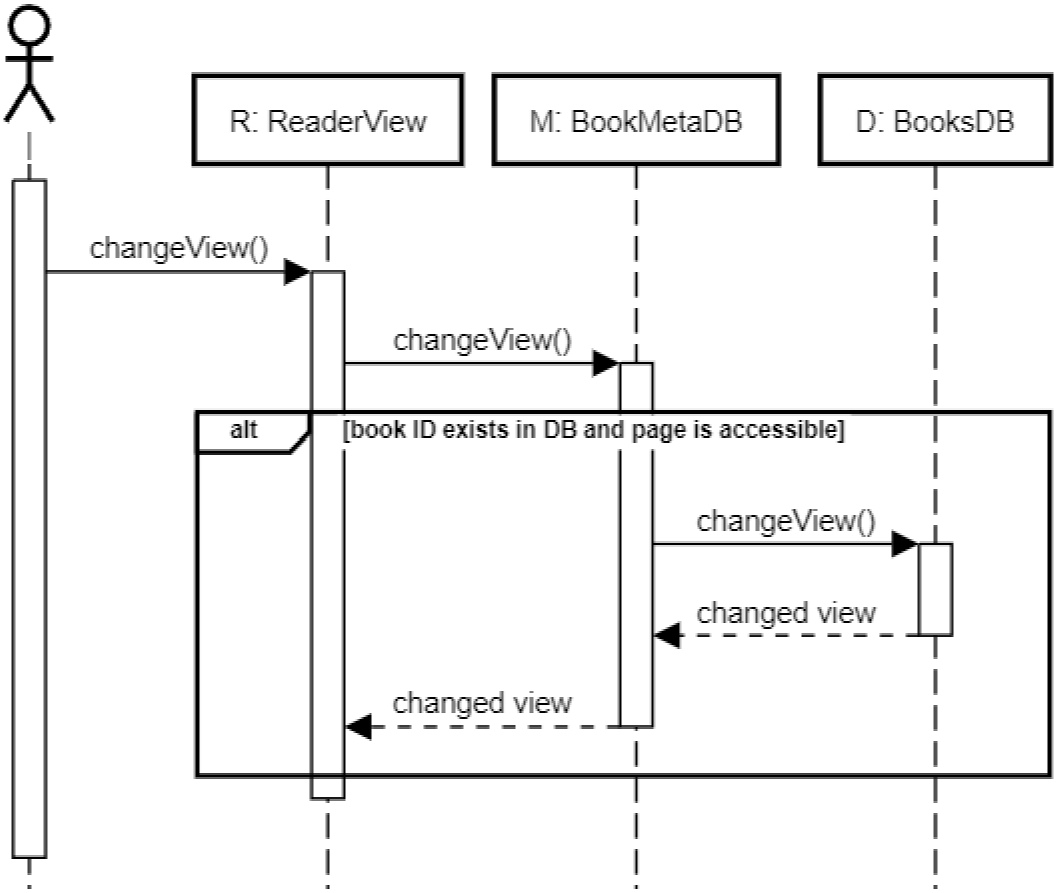
Add Bookmark


R: ReaderView
M: BookMetaDB
D: BooksDB
bookmark(BID,PgNum)
bookmark(BID,PgNum)
alt
[book ID exists in DB and page is accessible]
bookmark(BID,PgNum)
message: success
message: success
[page number does not exist in book]
display: error cannot bookmark nonexistant page
[book ID does not exist in DB]
display: book does not exist error

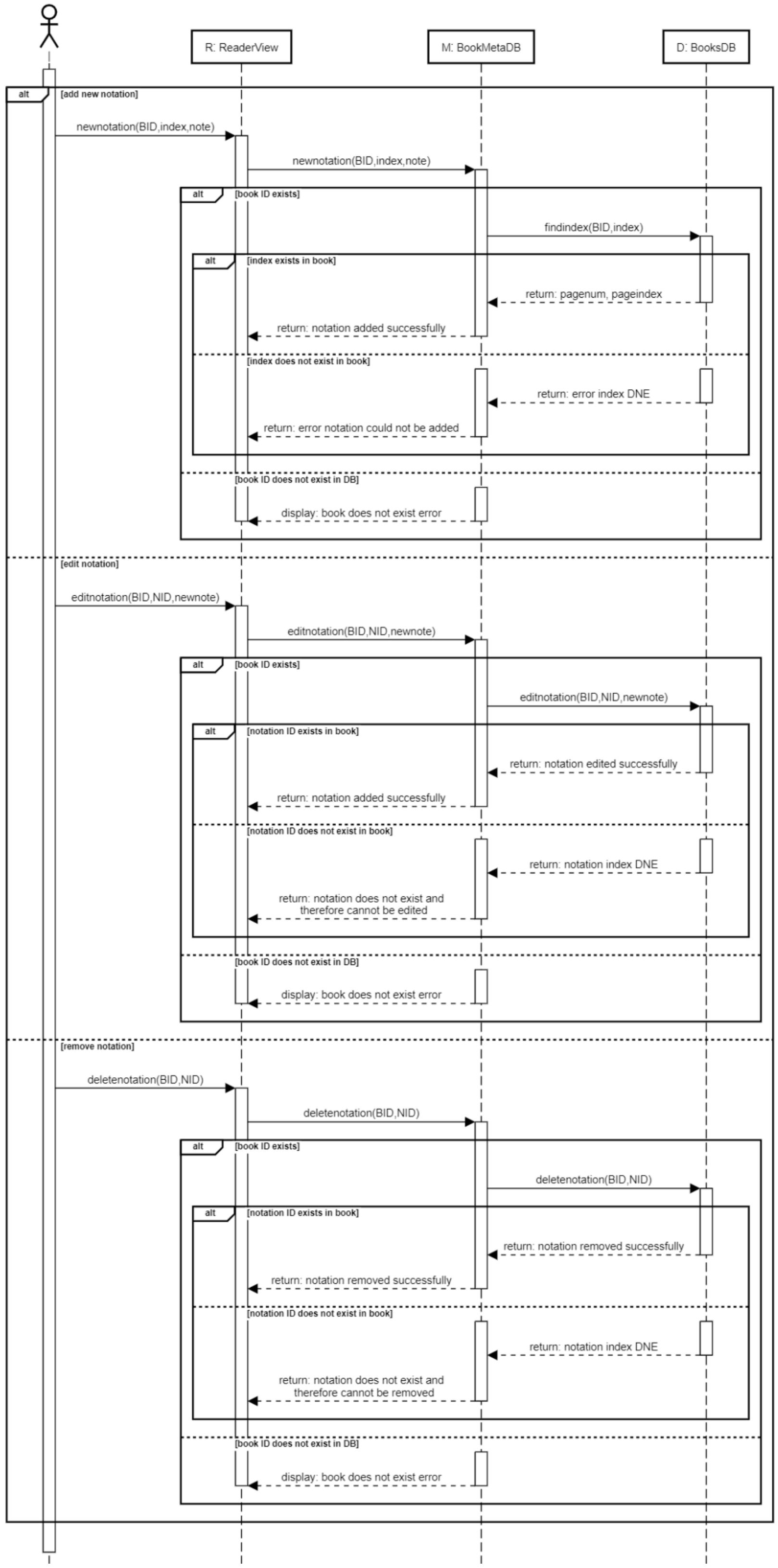
Search Text


R: ReaderView
M: BookMetaDB
D: BooksDB
searchtext(BID,SearchQuery)
searchtext(BID,SearchQuery)
alt
[book ID exists in DB]
searchtext(BID,SearchQuery)
alt
[searched phrase exists in book]
return: results
return: results
[searched phrase does not exist in book]
return: phrase not found
return: phrase not found
[book ID does not exist in DB]
display: book does not exist error

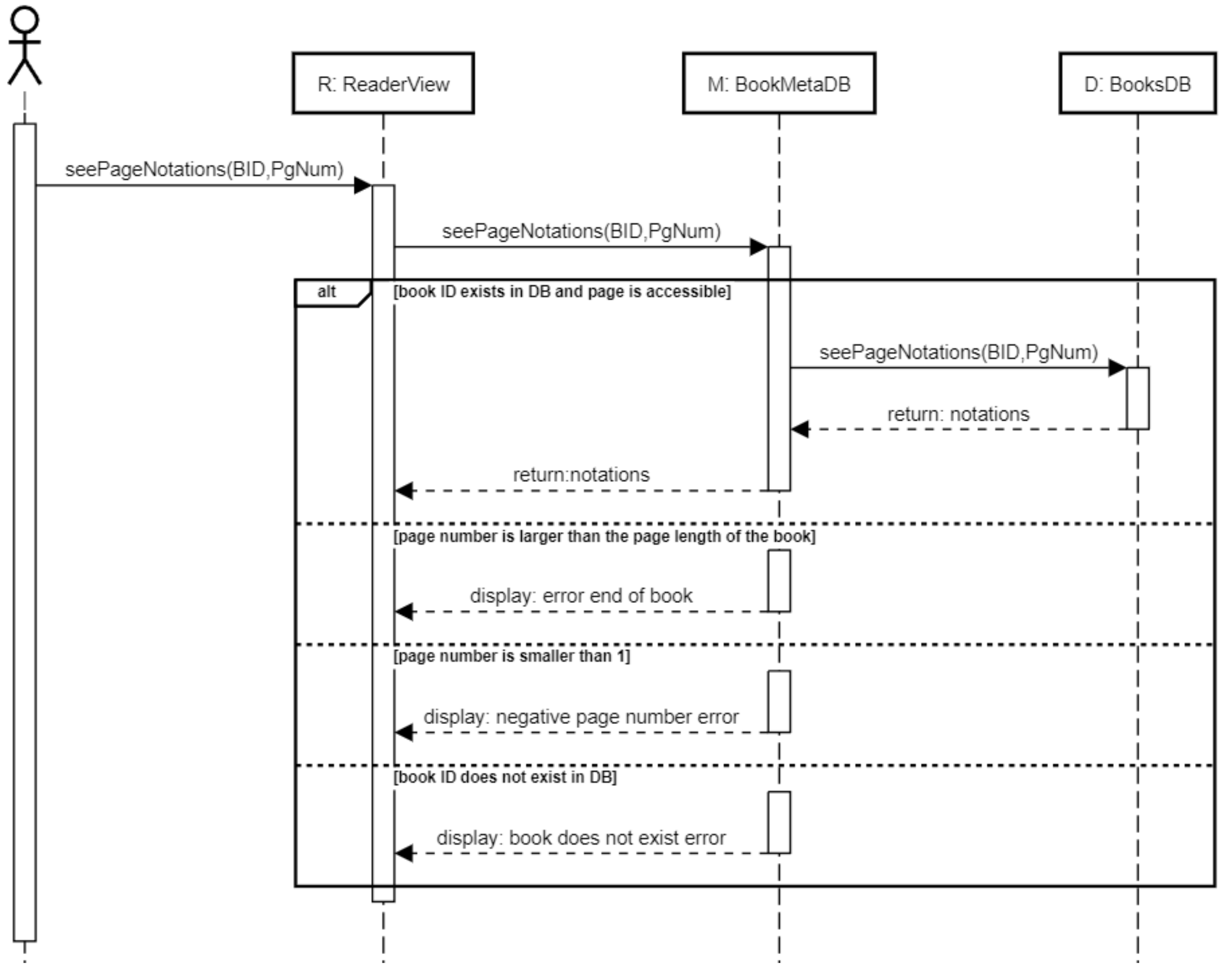
Change View Settings



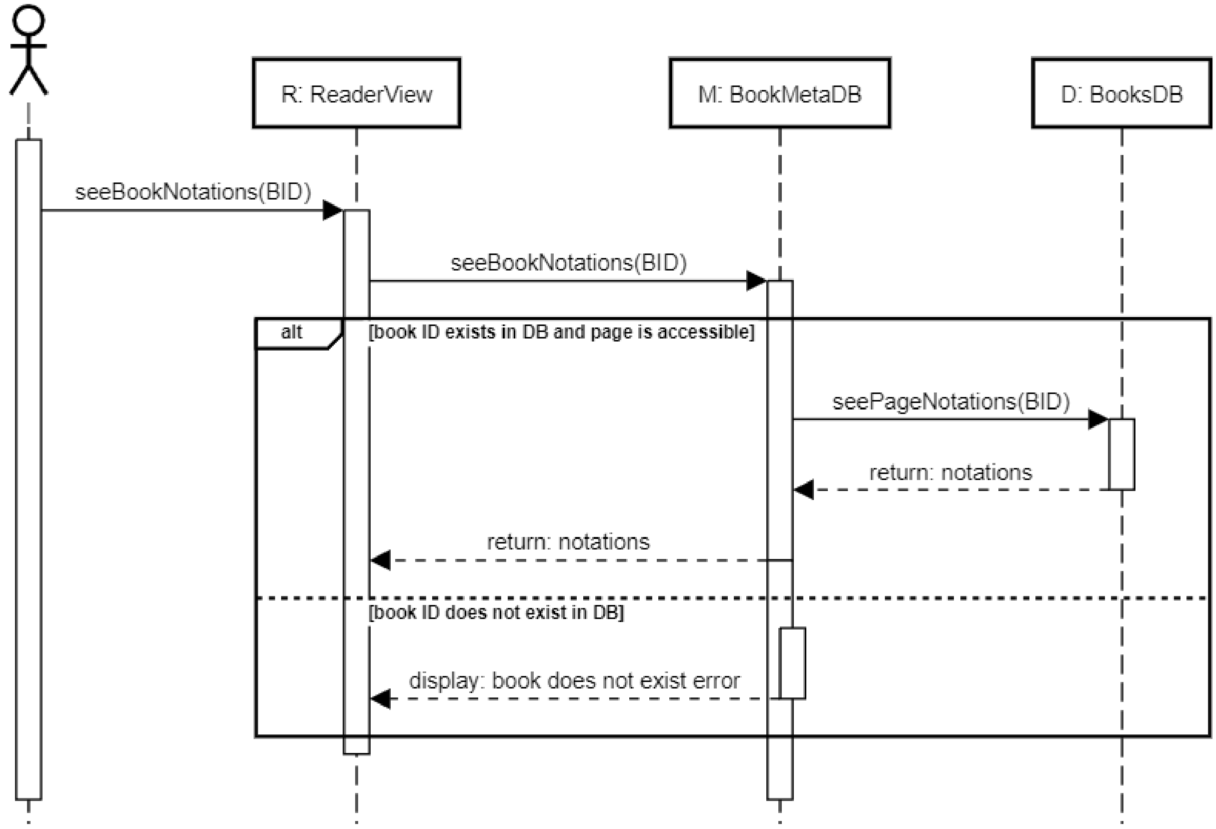
Add, Edit, and Remove notations


R: ReaderView
M: BookMetaDB
D: BooksDB
alt
[add new notation]
newnotation(BID,index,note)
newnotation(BID,index,note)
alt
[book ID exists]
findindex(BID,index)
alt
[index exists in book]
return: pagenum, pageindex
return: notation added successfully
[index does not exist in book]
return: error index DNE
return: error notation could not be added
[book ID does not exist in DB]
display: book does not exist error
[edit notation]
editnotation(BID,NID,newnote)
editnotation(BID,NID,newnote)
alt
[book ID exists]
editnotation(BID,NID,newnote)
alt
[notation ID exists in book]
return: notation edited successfully
return: notation added successfully
[notation ID does not exist in book]
return: notation index DNE
return: notation does not exist and therefore cannot be edited
[book ID does not exist in DB]
display: book does not exist error
[remove notation]
deletenotation(BID,NID)
deletenotation(BID,NID)
alt
[book ID exists]
deletenotation(BID,NID)
alt
[notation ID exists in book]
return: notation removed successfully
return: notation removed successfully
[notation ID does not exist in book]
return: notation index DNE
return: notation does not exist and therefore cannot be removed
[book ID does not exist in DB]
display: book does not exist error

See notations on a page

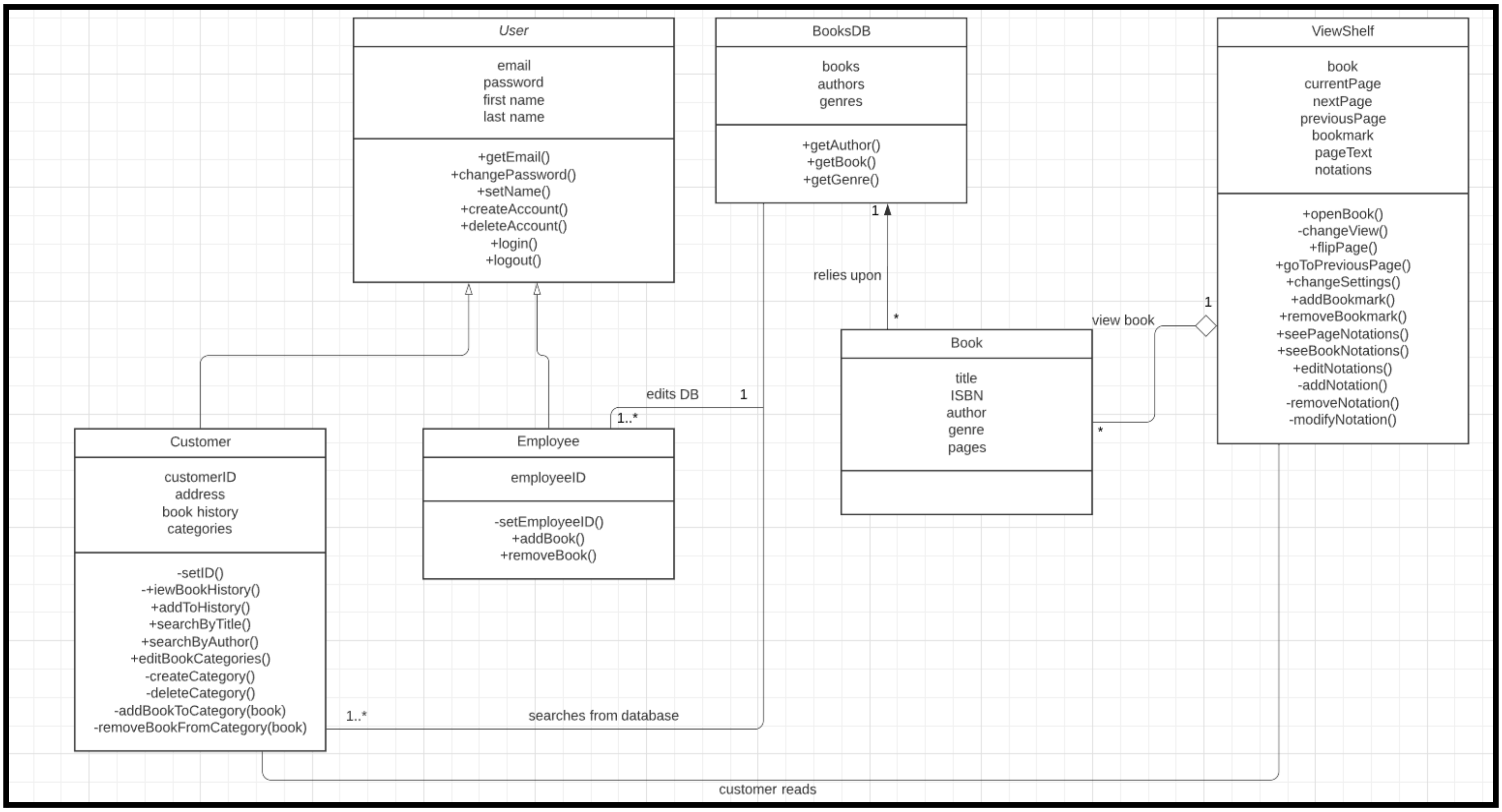

R: ReaderView
M: BookMetaDB
D: BooksDB
seePageNotations(BID,PgNum)
seePageNotations(BID,PgNum)
alt
[book ID exists in DB and page is accessible]
seePageNotations(BID,PgNum)
return: notations
return:notations
[page number is larger than the page length of the book]
display: error end of book
[page number is smaller than 1]
display: negative page number error
[book ID does not exist in DB]
display: book does not exist error

See notations for a whole book



8. [15 POINTS]

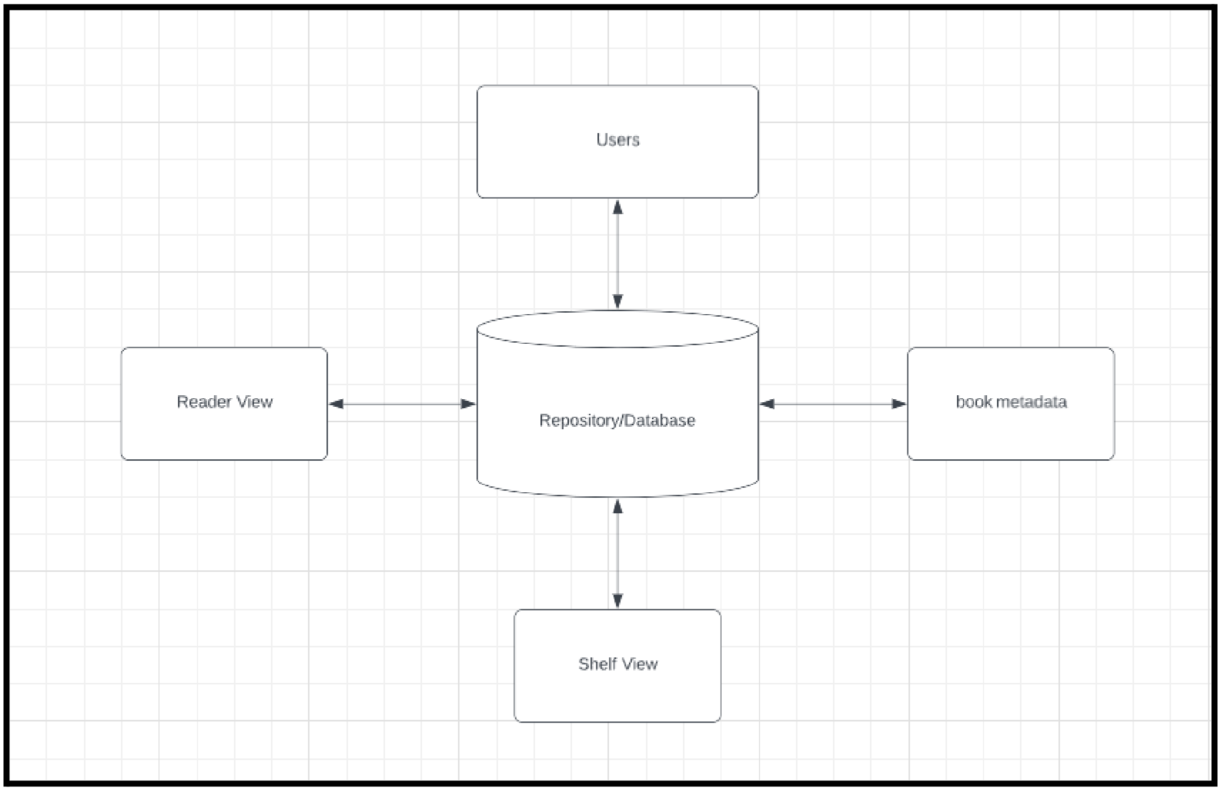
Class diagram



9. [15 POINTS]

Architectural design

9.3. Repository architecture pattern (similar to Figure 6.11) (Our Choice)


Users
Reader View
Repository/Database
book metadata
Shelf View