CS3354 Software Engineering

Final Project Deliverable 2

Book Shelf Software

Brady Cash, Christian Flores, Emmanuel Samuel, AJ Prosser

1. Delegation of Tasks

Deliverable 1:

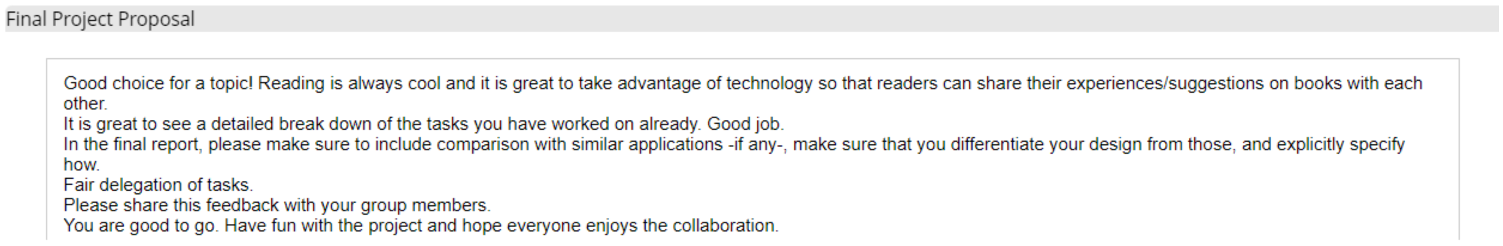
- Setting up Github repository:
    - Task 1.1 - (Everyone)
    - Task 1.2 - Emmanuel
    - Task 1.3 - Emmanuel
    - Task 1.4 - Christian
    - Task 1.5 - AJ
- Diagrams:
    - Use Case diagram- Christian
    - Sequence diagram- AJ and Emmanuel
    - Class diagram- Brady
- Architectural designs:
    - Repository architecture pattern- Brady
- Which software process model and why- Christian
- Functional requirements - AJ
- 6: Non-functional requirements- Christian

Deliverable 2:

- Project Scheduling, Cost, Effort and Pricing Estimation- Christian, AJ, Brady, Emmanuel
- Test plan- Emmanuel
- Comparison with similar designs- AJ
- Conclusion- Christian, AJ, Brady

2. Project Deliverable 1 content:

1.



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. .

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

4. 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. Software Requirements including

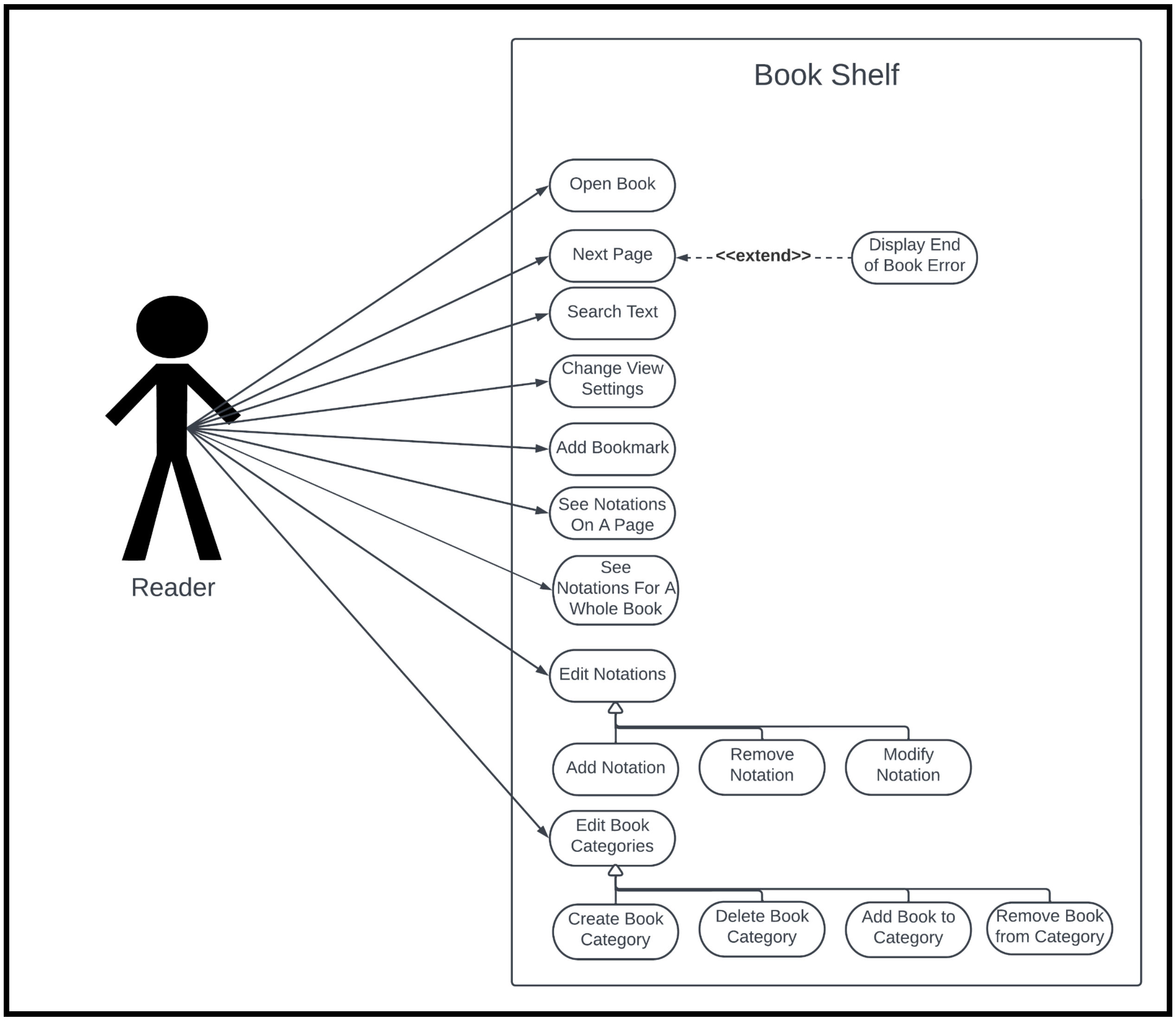
5.a.) Functional requirements.

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.) Non-functional requirements

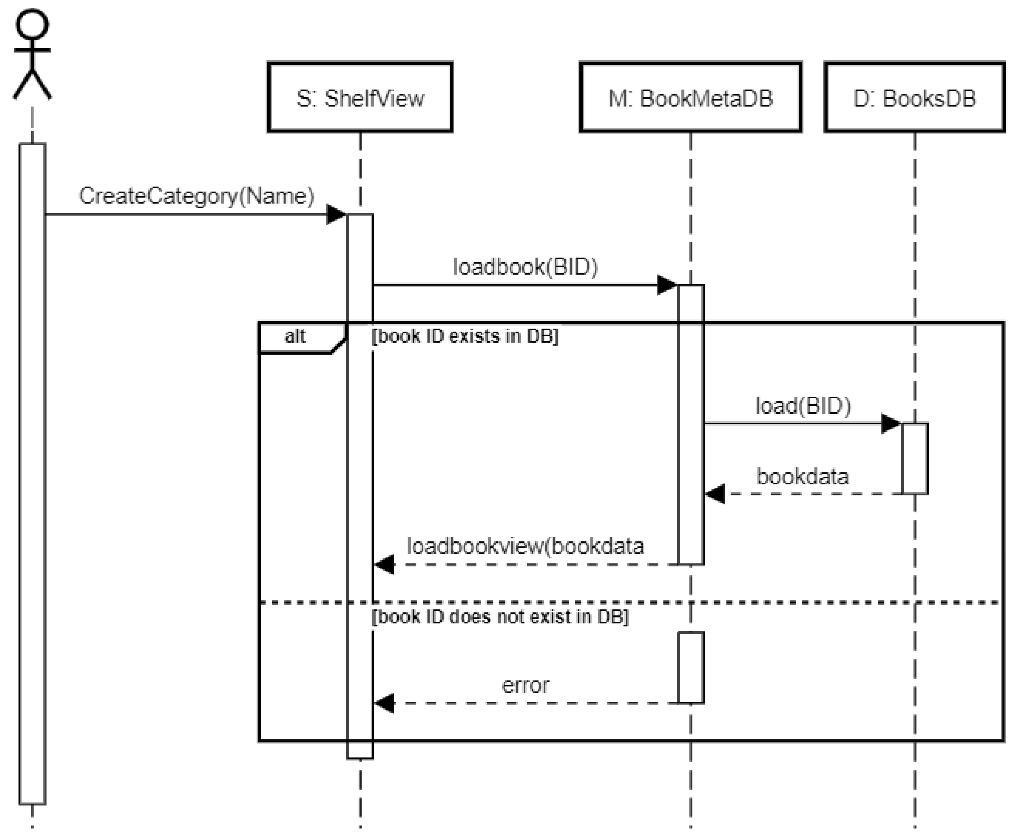
1. **Usability requirement:** The software should support English, Spanish, and Chinese languages.
2. **Performance requirement:** It should not take more than 5 seconds to load a book.
3. **Space requirement:** The software should not be larger than 10 GB.
4. **Dependability requirement:** There should 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. Use case diagram

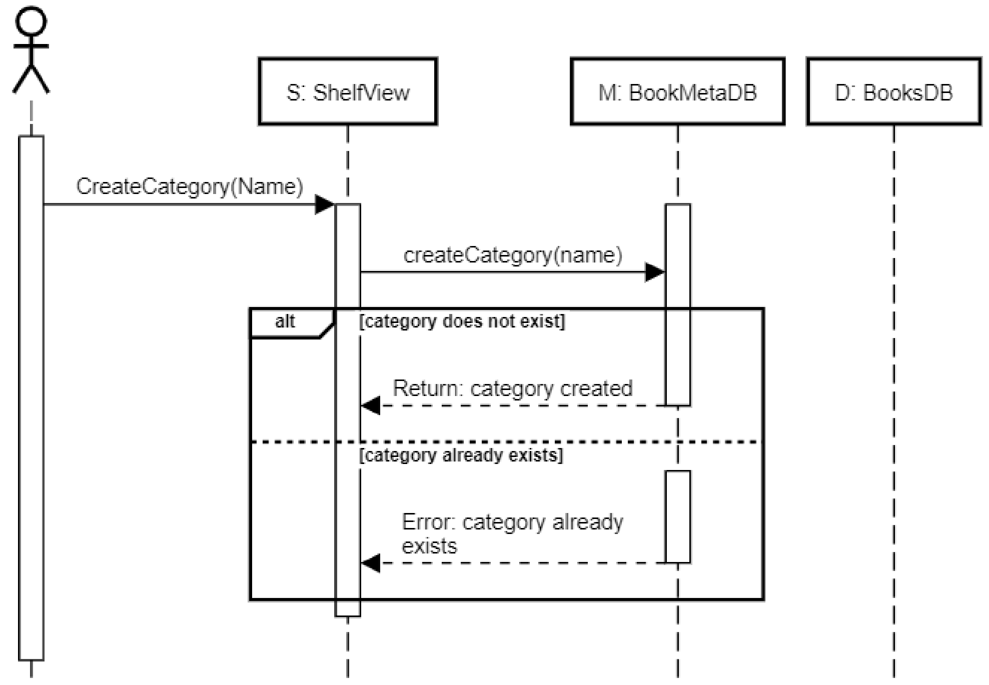


7. Sequence diagram by use case:

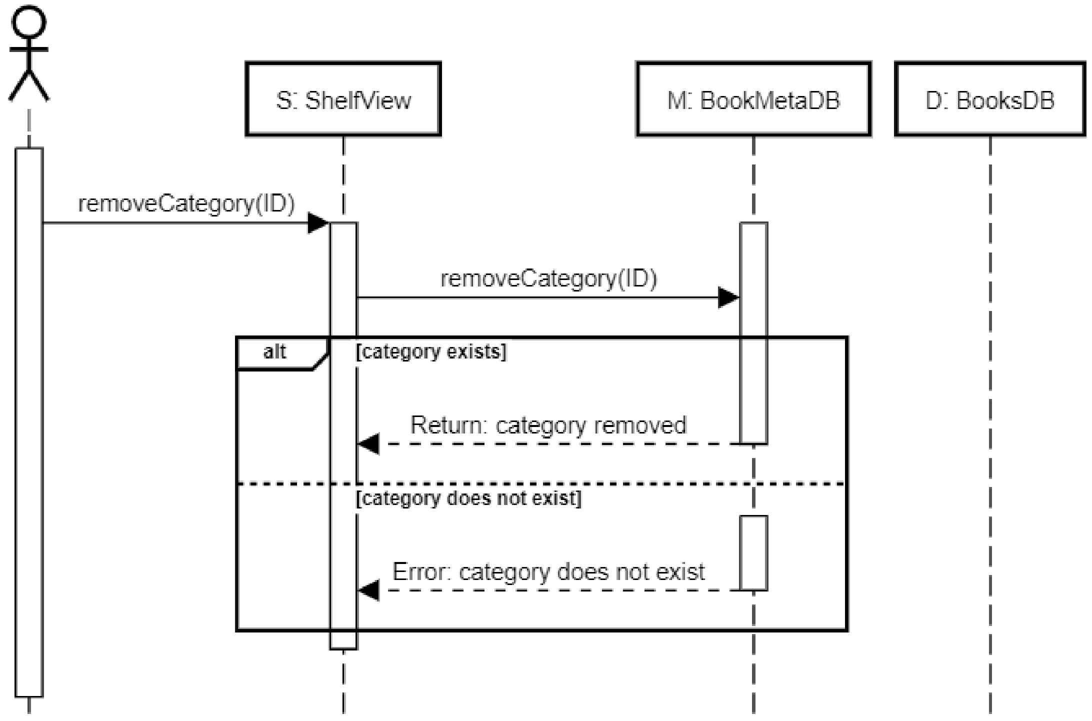
Open Book



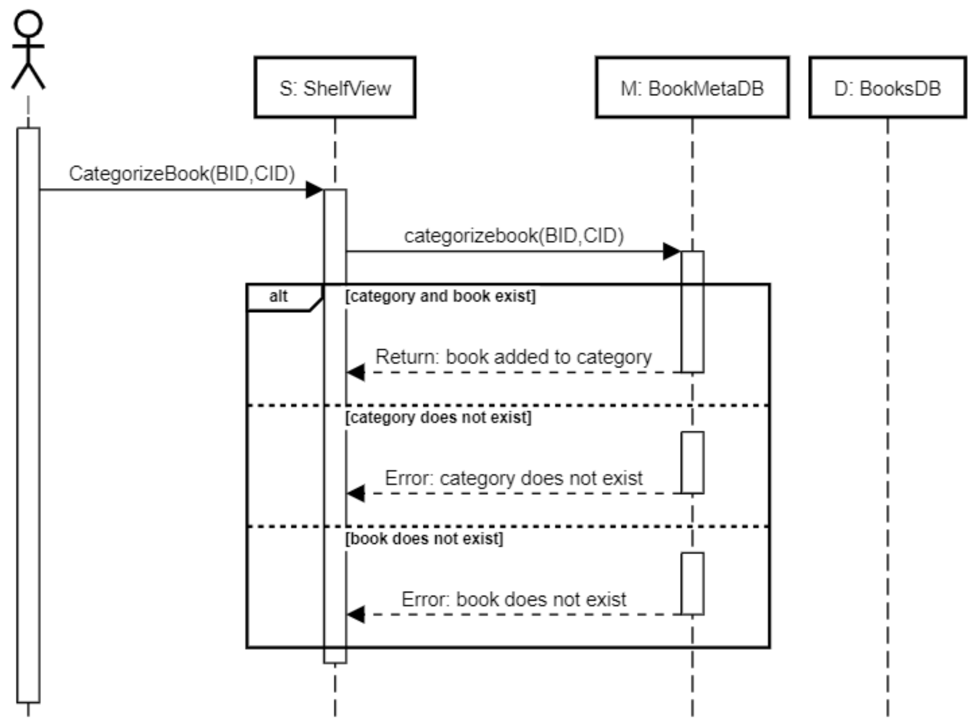
Create Book Category



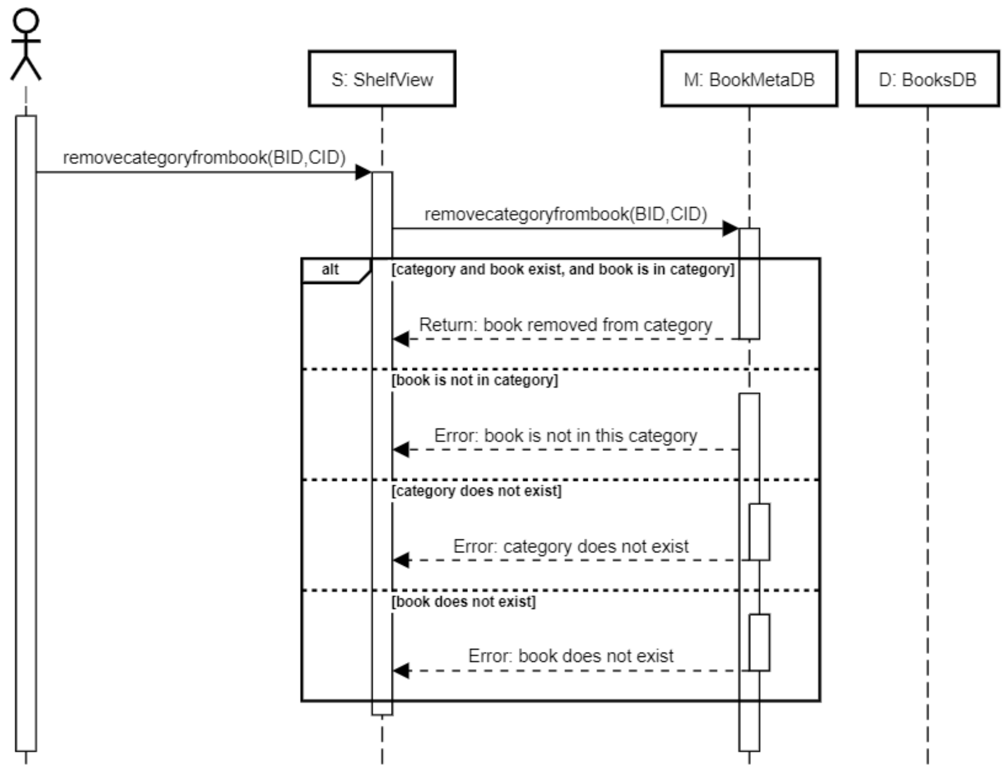
Delete Book Category



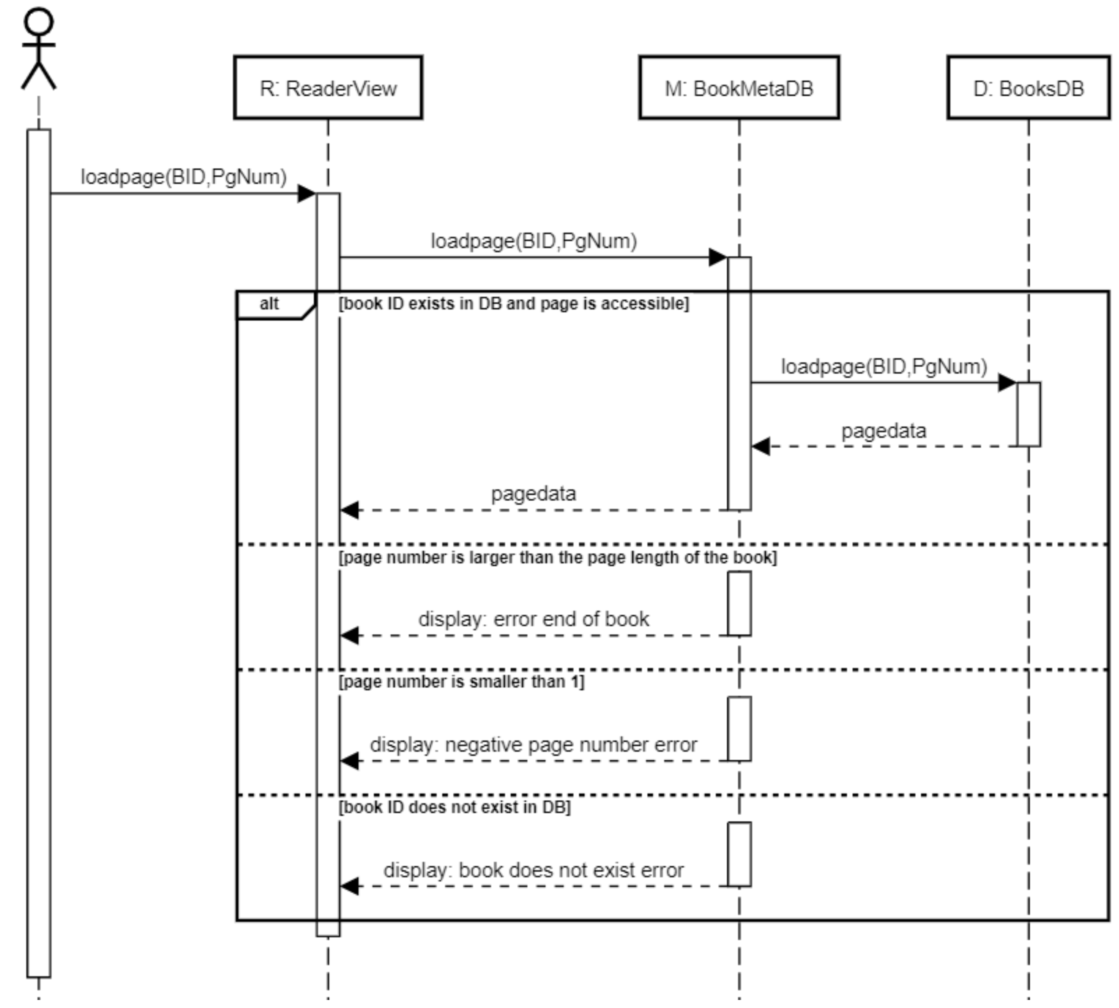
Add book to Category


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

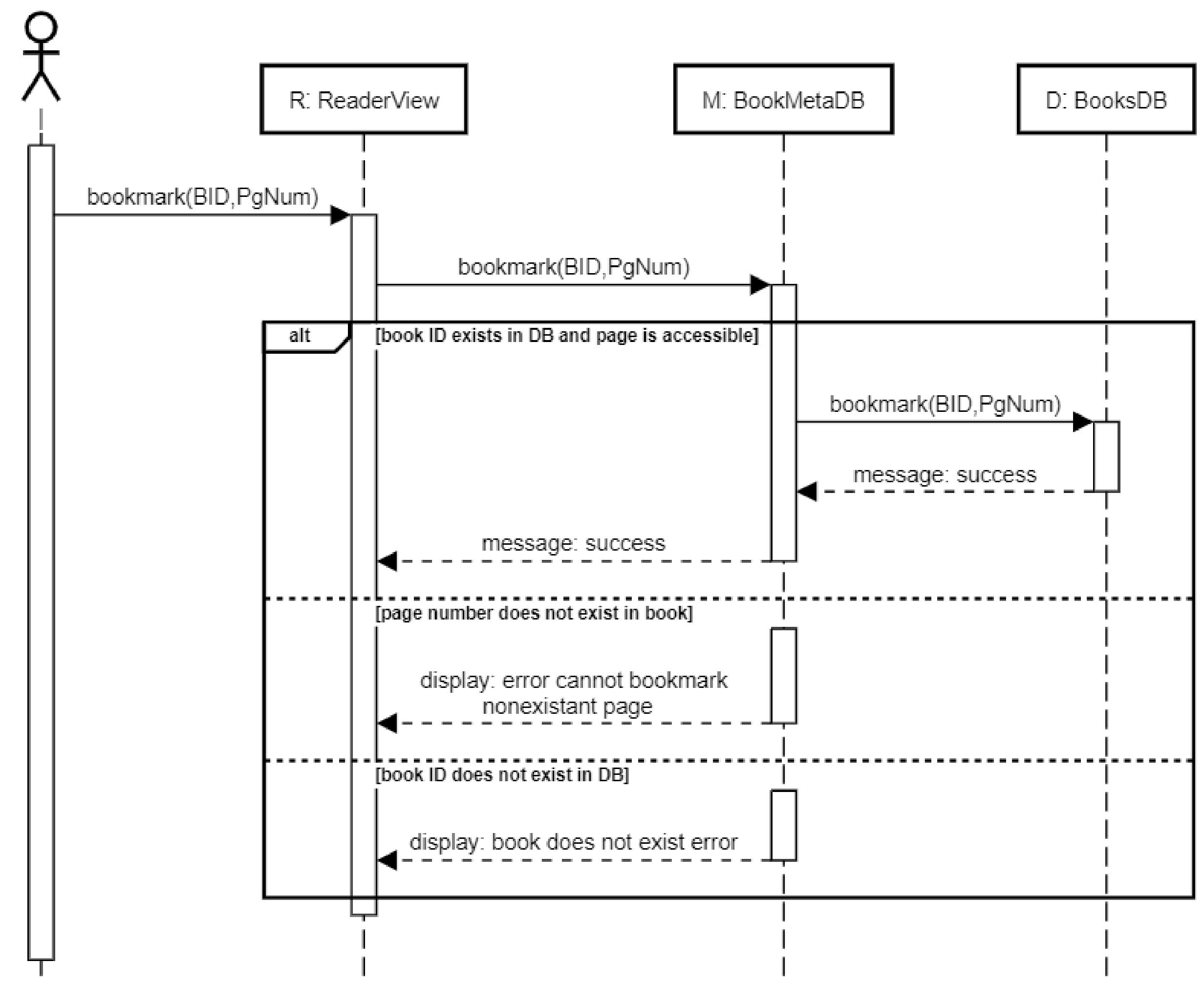
Remove book from Category


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

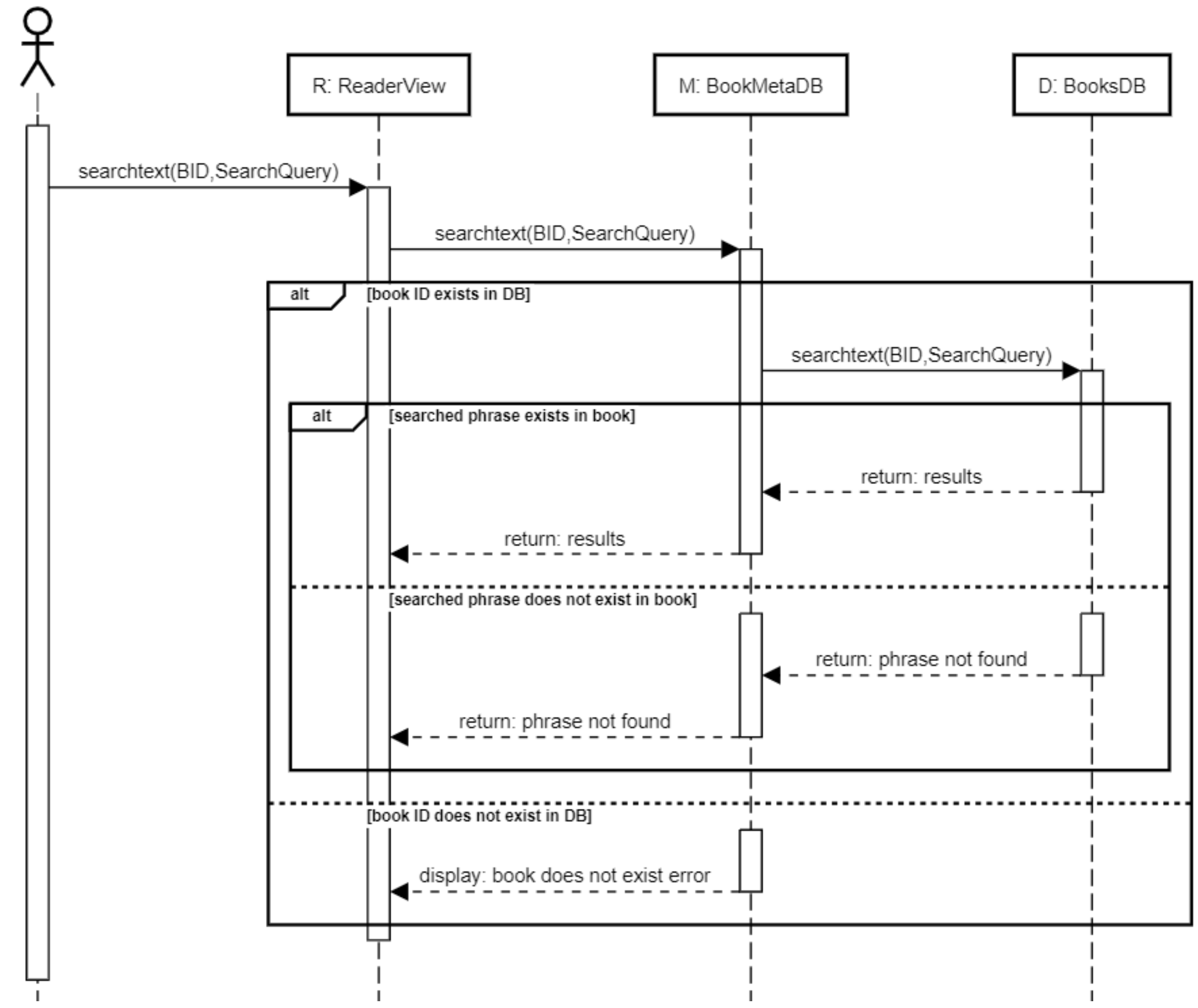
Next Page


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

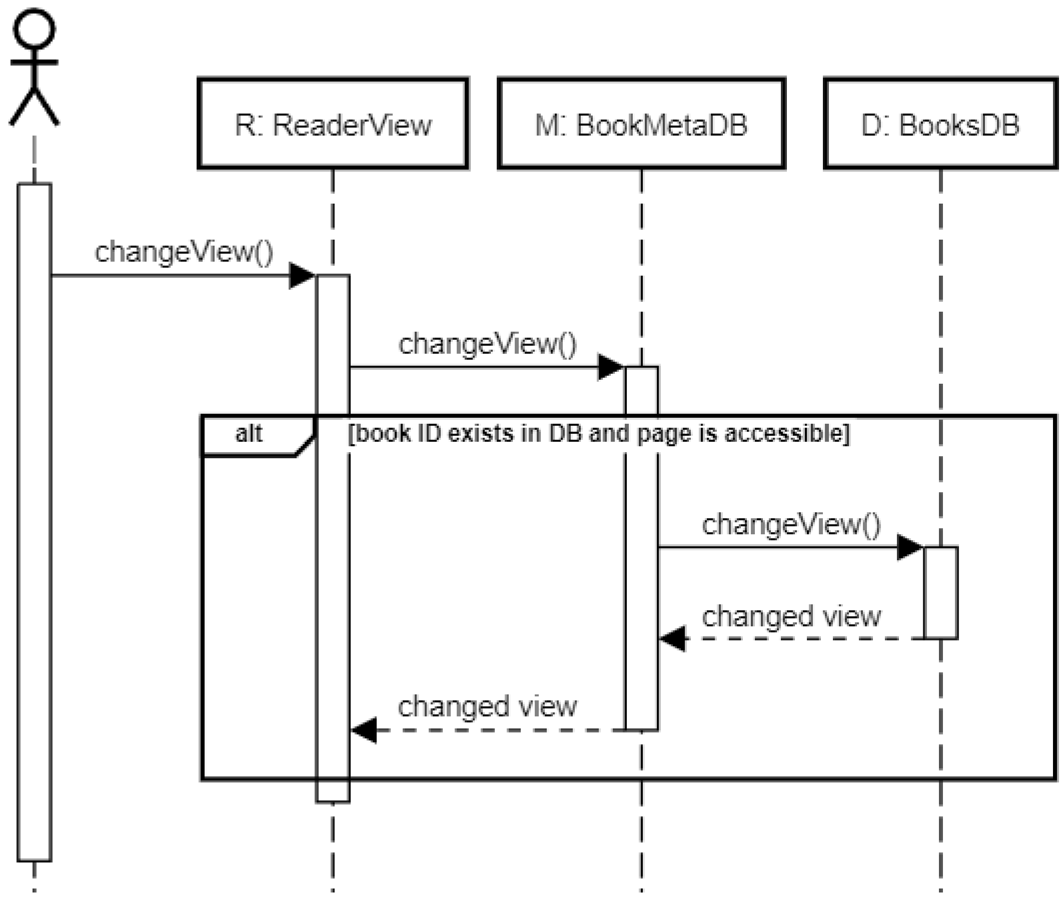
Add Bookmark


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

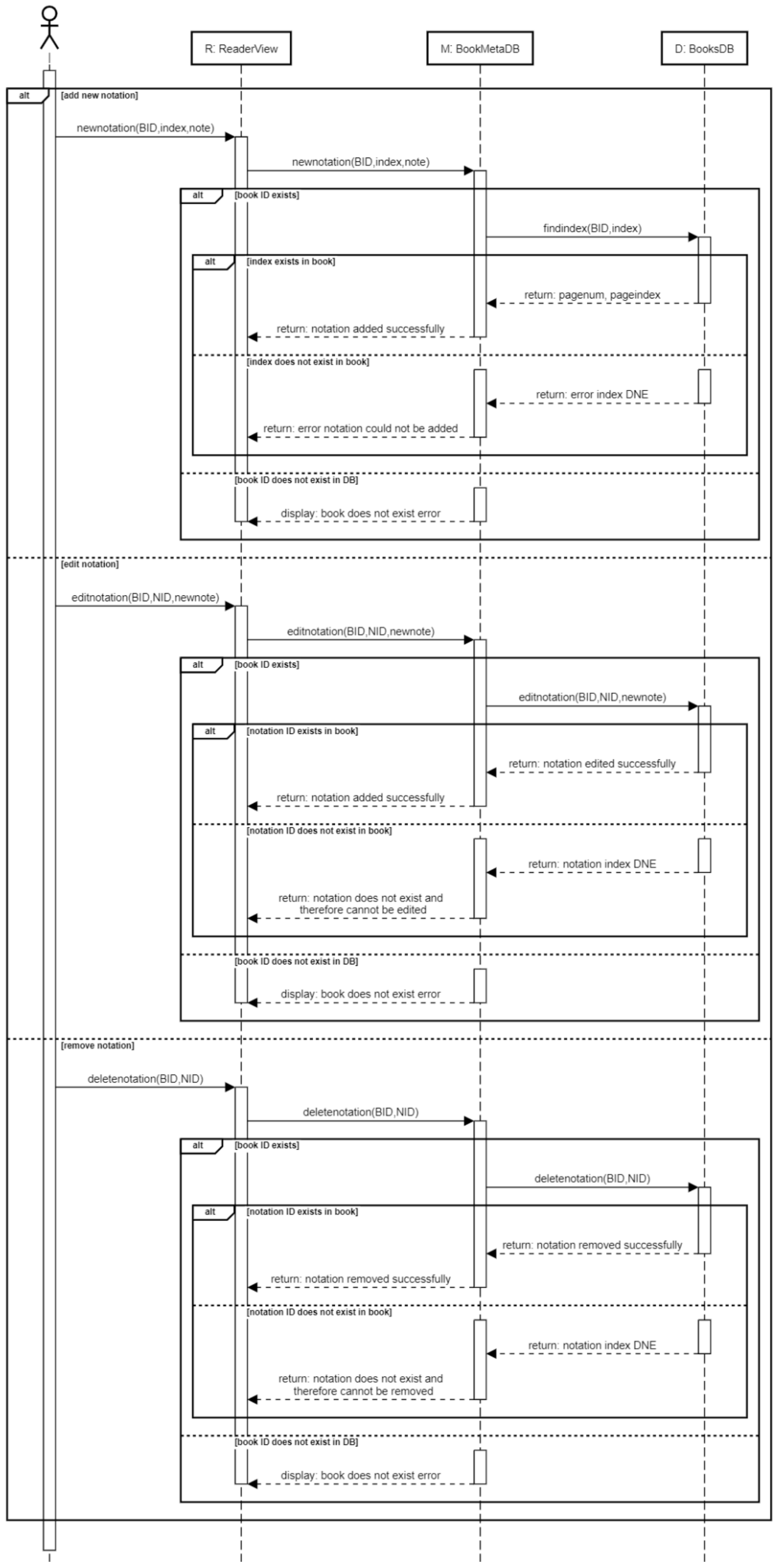
Search Text


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

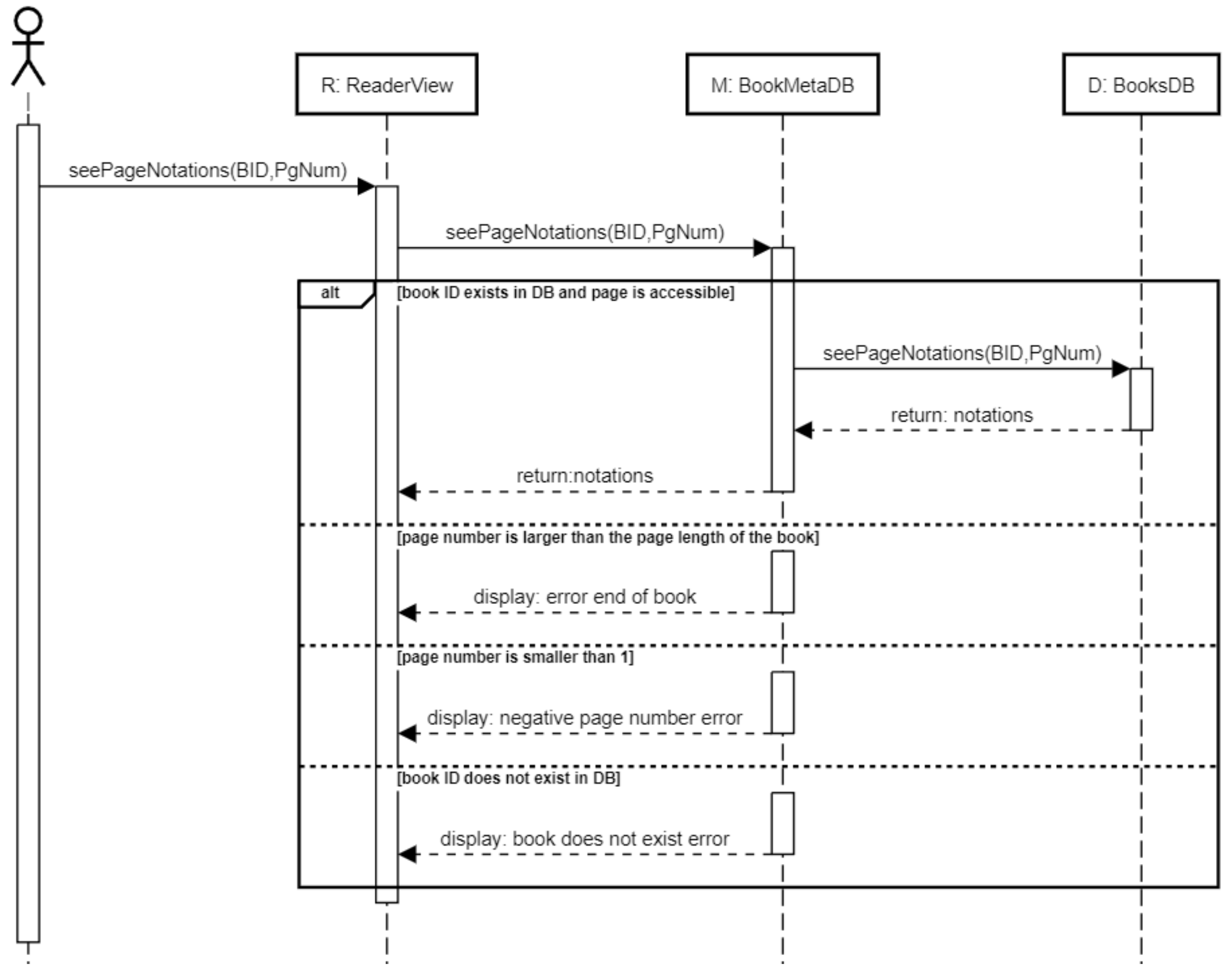
Change View Settings



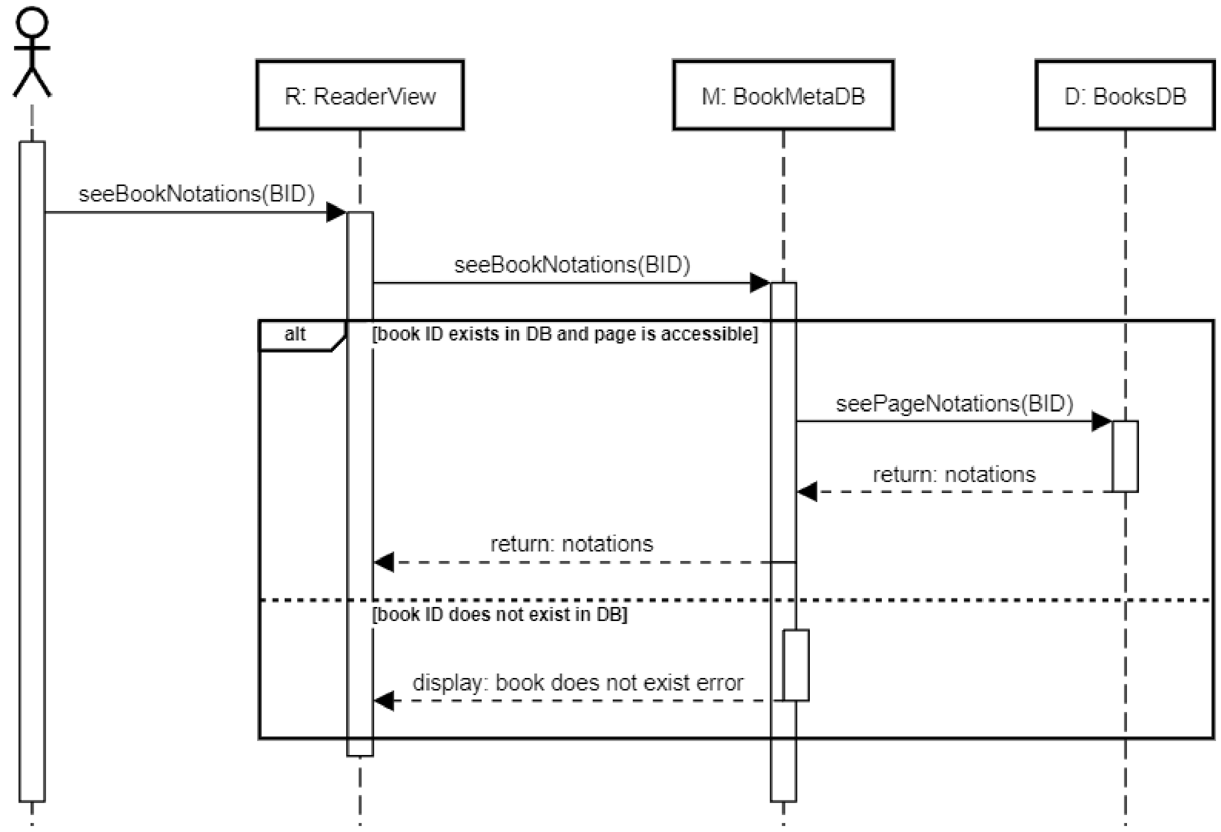
Add, Edit, and Remove notations


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

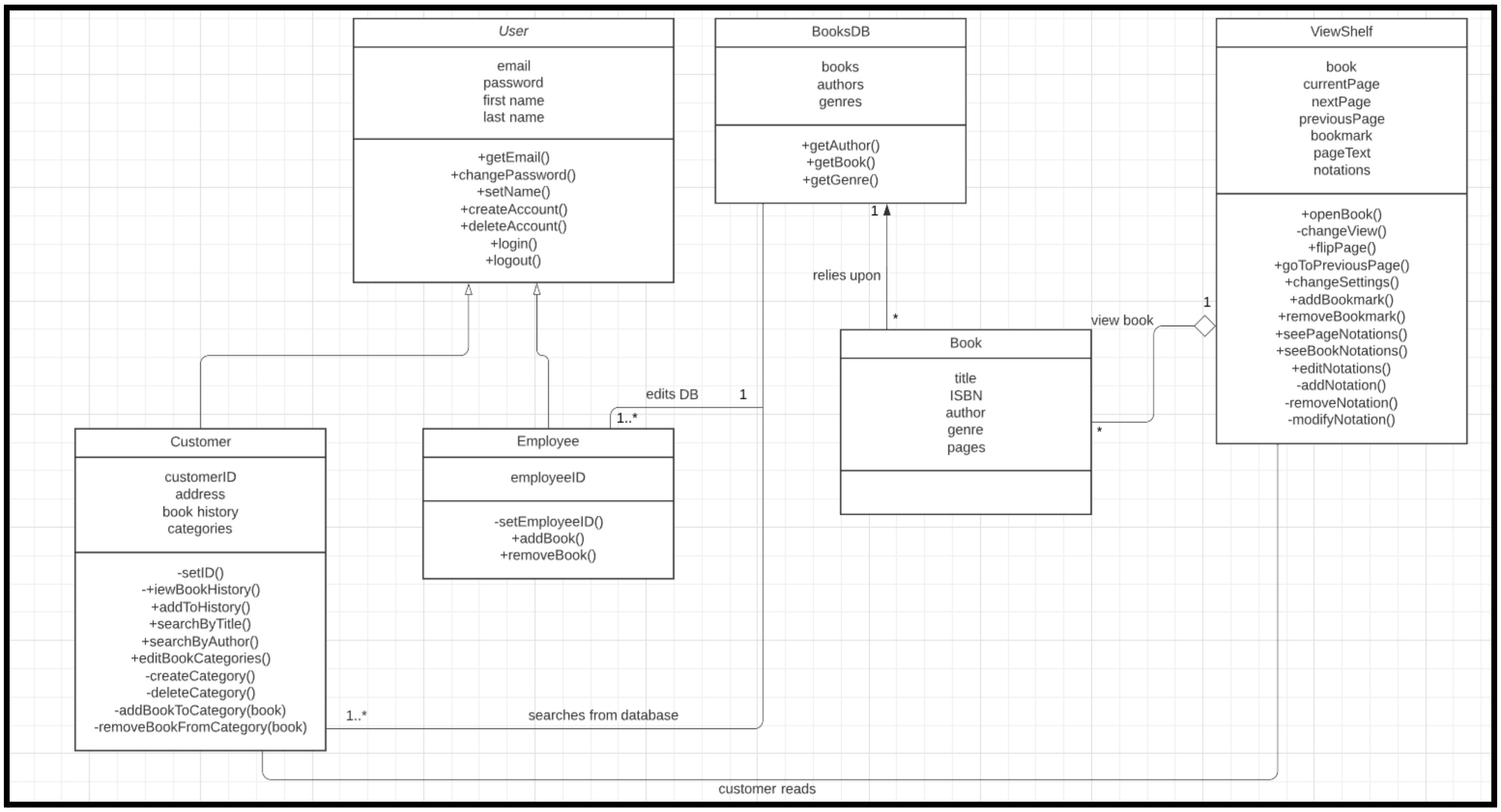
See notations on a page


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

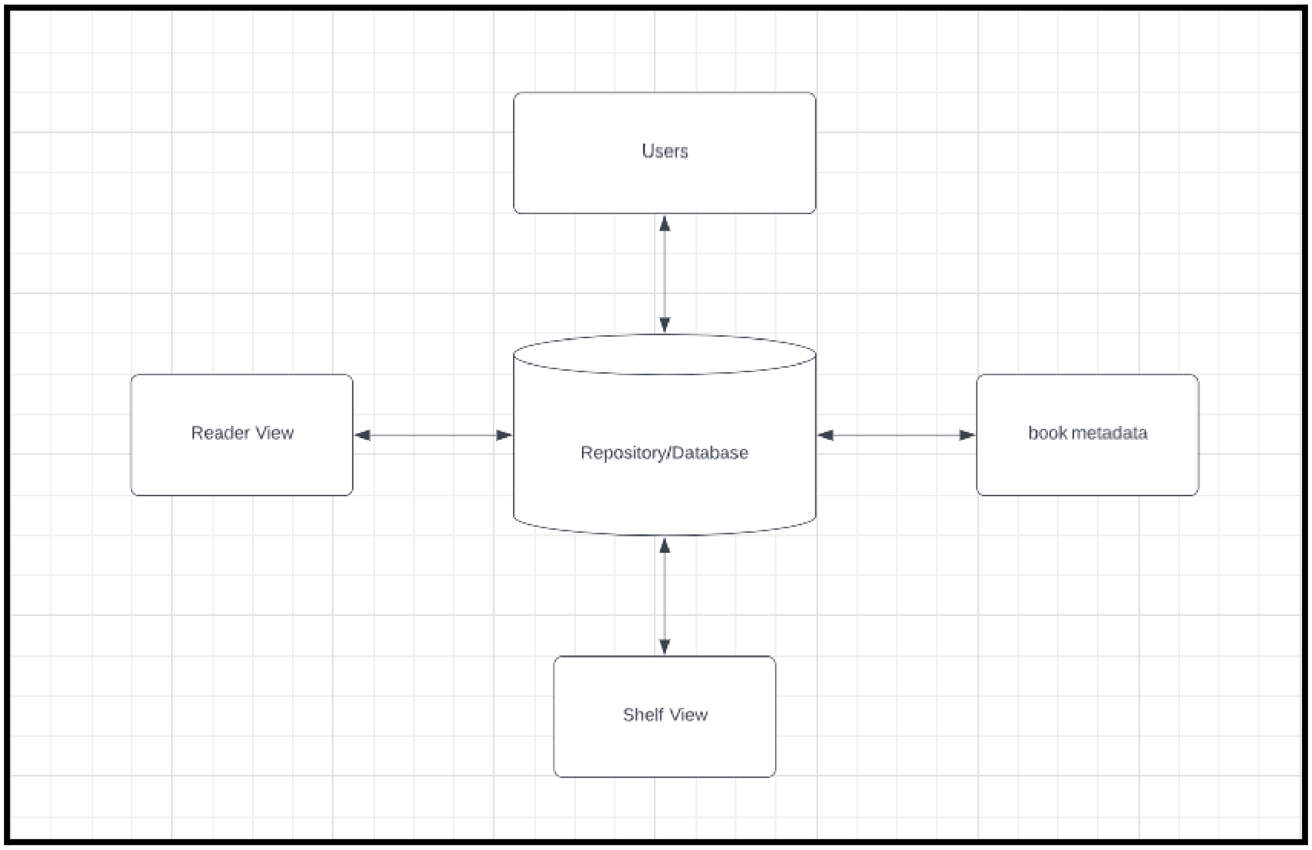


8. Class diagram



9. Architectural design

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



END OF DELIVERABLE 1

3. Project Scheduling, Cost, Effort/Pricing Estimation, Project duration/staff

3.1. Project Scheduling

The initial startup of our project will begin on May 1 2023, and it will end on May 8 2023. This is because I have estimated using the function point algorithm that it will take less than 1 week to program our bookshelves functional capabilities, as seen in the next question. Thus, we will spend a couple extra days sorting out bugs, which is why I gave it a week for the initial release. The number of workers per day is estimated to be 8 hours a day per person only on weekdays, and we would have 7 employees.

3.2. Cost, Effort and Pricing Estimation. - Function Point (FP)

	Function Category	Count	Complexity			Count x Complexity
			Simple	Avg	Cmplx	
1	# of User Input	1 (Notation) + 1 (Categories + 1 (Search Text) + 1 (Adding Bookmark) + 1 (View settings) = 5 inputs	3	4	6	4 * 5 = 20
2	# of User Output	1 (Displaying Text W/ Correct View Settings) + 1 (Displaying a Notation) = 2	4	5	7	2 * 5 = 10
3	# of User Queries	2 (Displaying Notations per book/per page) + 1 (Text Search Results) + 1 (Displaying books within a category) = 4	3	4	6	4 * 4 = 16
4	# of Data Files and Relational Tables	1 (Type of book file), 1 (Table to categorize all the books), 1 (Relational table to connect the first and second) = 3	7	10	15	Complexity of Relational table is simple, the other two are average 3 * 9 = 27
5	# of External Interfaces	1 (Touchscreen)	5	7	10	1 * 7 = 7
					GFP	80

Calculate adjustment:

PC1 = 1
PC2 = 1
PC3 = 0
PC4 = 1
PC5 = 4
PC6 = 1
PC7 = 0
PC8 = 0
PC9 = 2
PC10 = 1
PC11 = 1
PC12 = 3
PC13 = 3
PC14 = 5

Sum(PC) = 20
PCA = 0.65 + .01(20) = 0.85

Function Point = GFP x PCA
Function Point = 80 * 0.85
Function Point = 68
Estimated effort = FP / productivity = 68 / 60 = 1.13333
D = 1.13333 / 7 = .162 weeks

It is difficult to calculate the real-world pricing from a function point number without prior experience. However, the slides give an example of an experienced programmer doing 60 function points per week.

3.3. Estimated cost of hardware products

Server and Storage - Since the average storage size of a book is only around 2 MB [5], with 4 TB of storage we could hold roughly 2 million books in our repository. Building a small business server with no storage would cost around $2000 [6] given the going rates of motherboards and processors. Add in four 1TB SSDs for $200 each, and the final price for our server will only be $2800. Note that we are building our own server and not using cloud services.

- Server cost: $2800

3.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)
Integrated Development Environment (IDE) - We will be using VS Code for its flexibility to work with different frameworks and tools.

- Visual Studio Code: Free of cost

Cross-platform Framework - We will be using React Native because of its cross-platform abilities and functionality with Android and iOS devices. Additionally, we will use the React Native for Web library for desktop users.

- React Native: Free of cost
- React Native for Web: Free of cost

Backend Framework - For our app, we plan on using Java with the Spring boot framework for the backend. This combination allows for easy data access and extensibility.

- Java: Free of cost
- Spring Boot: Free of cost

Database Software - For our app we plan on using MariaDB for our database software, MariaDB operates under the GPL (GNU General Public License) license, therefore it wouldn't cost us anything unless we breached the license.

- MariaDB: Free of cost

Version Control System: We will be using Git, and will be hosting our repositories on GitHub. We will be using the Free Plan on GitHub because of the small team size.

- Git: Free of cost
- GitHub: Free of cost

Continuous Integration/Continuous Deployment (CI/CD) - We will be using GitHub actions to automate the build and deployment processes as 2,000 minutes is included in our Free Plan on GitHub.

- GitHub Actions: Free of cost

Project Management and Collaboration Tools - We will be using Trello to communicate with team members and organize/assign tasks. We will not need extensive plans due to small team size and simplicity of project.

- Trello: Free of cost

Design and Prototyping Tools - We will be using Figma to design the app's user interface, along with developing prototypes for the interface. Due to the short development period and small team size, the starter plan is satisfactory for our needs.

- Figma: Free of cost

3.5. Estimated cost of personnel

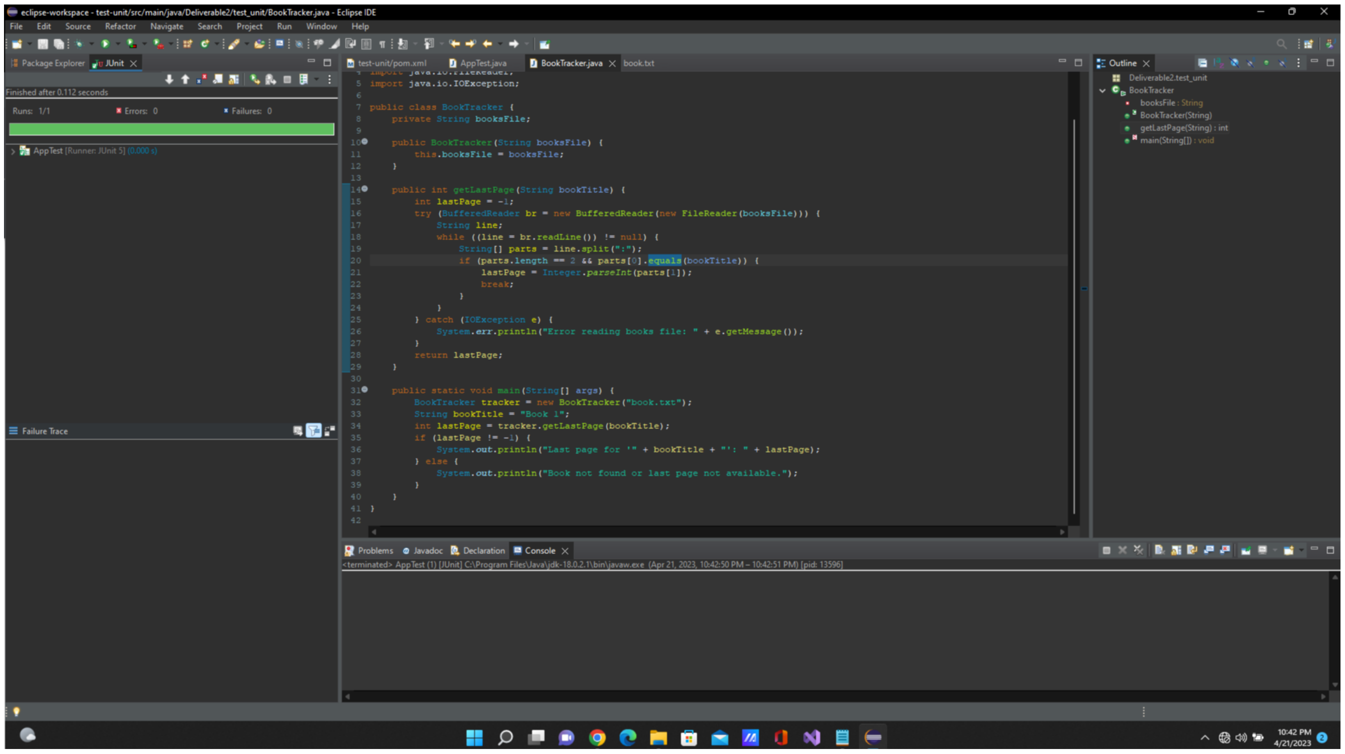
Per levels.fyi, the median software engineer salary in Dallas, Texas is **$124,000** and the median software engineering manager salary in Dallas, Texas is **$210,095.** . I estimate that we could build and maintain our app with a team of 7 engineers and 1 manager which would lead to a cost of personnel of **$1,078,095**

4. [10 POINTS] A test plan for your software:

Among the many functionalities required for this software, one of them is ensuring that the application retrieves the correct page number from a database. The software needs to be able to show the user the page that they left off from the last time they read the book. This data will be concurrently updated in the backend using MariaDB.

However, due to limitations of the project and access to the database by the grader, we developed a simple method that demonstrates how this would work. The class "BookTracker" accepts the title of the book, and looks up the title in the database (in this project we just used a .txt file) and returns the associated page number. Code is in the deliverable file.

Screenshot:



Finished after 0.112 seconds

Runs: 1/1 Errors: 0 Failures: 0

AppTest [Runner: JUnit 5] (0.000 s)

```java
import java.io.IOException;

public class BookTracker {
    private String booksFile;

    public BookTracker(String booksFile) {
        this.booksFile = booksFile;
    }

    public int getLastPage(String bookTitle) {
        int lastPage = -1;
        try (BufferedReader br = new BufferedReader(new FileReader(booksFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(":");
                if (parts.length == 2 && parts[0].equals(bookTitle)) {
                    lastPage = Integer.parseInt(parts[1]);
                    break;
                }
            }
        } catch (IOException e) {
            System.err.println("Error reading books file: " + e.getMessage());
        }
        return lastPage;
    }

    public static void main(String[] args) {
        BookTracker tracker = new BookTracker("book.txt");
        String bookTitle = "Book 1";
        int lastPage = tracker.getLastPage(bookTitle);
        if (lastPage != -1) {
            System.out.println("Last page for '" + bookTitle + "': " + lastPage);
        } else {
            System.out.println("Book not found or last page not available.");
        }
    }
}
```

```
<terminated> AppTest (1) [JUnit] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Apr 21, 2023, 10:42:50 PM – 10:42:51 PM) [pid: 13596]
```

5. Comparison of your work with similar designs

Our software is functionally an e-book reader, so we will compare it to other popular ebook readers. *Calibre* is one of the most popular pieces of software which allows the user to read and manage books. *Calibre* is open source, and has had many years in development so it is only natural that it is more feature-robust than our implementation [1]. *Calibre* supports categorization, and many other metadata tools, but is, however, largely focused on organizing the user's library for use with a physical e-book reader. *Calibre* does have a reader within the software which supports many of our goals as well, but is not compatible with iOS or Android, which makes it different than our software which is designed to run on touchscreens. *Calibre* is a software which is free to download, which is cheaper than our software.

Another software is the operating system of the Amazon Kindle [2]. The Kindle is another very popular e-book reader, and it offers many of the same features as our application. However, the Kindle software is tied to the proprietary hardware of a kindle, and cannot be run on another operating system, like a tablet. Additionally, the Kindle is designed with a higher screen latency and lower refresh rate to prolong the battery life of the device, which makes leaving notes inconvenient. Additionally, the kindle is designed with a monochromatic display, so it is not a viable option for many books with colored illustrations. The Kindle retails starting at $99, but versions which allow drawn notations and (presumably) higher refresh rates are much pricier.

The final piece of software we will compare is Apple Books, the native book reader that comes pre-installed on iOS devices [3]. This software is very powerful, and offers many powerful features, such as bookmarking, keeping track of the currently read page, and drawn notations. However, it does not seem to have an equivalent to text notations, and therefore no way to log them. Additionally, the metadata tagging leaves a lot to be desired. Categories exist, but editing information past the displayed book name is generally not available. This software is restricted to iOS devices, and cannot run on Android systems. This software comes free with iOS devices, so if the user already owns an iOS device such as an iPhone or iPad, it is free, but if the user does not, using the software would cost the price of a device.

6. Conclusion

Overall, we were able to successfully plan out the creation of a book shelf software. We adequately demonstrated the cost and the amount of effort it would take to make the entire system. However, we had to cut a lot of features in order to make the project more reasonable for the four of us to plan out. Since we originally had 60+ features it wouldn't have been reasonable to offload that much work on our sequence diagram creators. However, since we were planning on implementing a prototyping software process model, as time went on we could implement all the features we originally planned on in different releases of our product, so it was alright to remove some features on our initial release, so we could deliver a working product to our stakeholders in time.

7. References:

[1] K. Goyal, “About Calibre,” *Calibre E-book management*. [Online]. Available: https://calibre-ebook.com/about. [Accessed: 20-Apr-2023].

[2] “Kindle,” *Amazon*. [Online]. Available: https://www.amazon.com/b/?node=6669702011&tag=googhydr-20&hvadid=453973924243&hvpos=&hvnetw=g&hvrand=6608668888517392614&hvpone=&hvptwo=&hvqmt=e&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9026839&hvtargid=kwd-294877324766&ref=pd_sl_6itck04ygw_e. [Accessed: 20-Apr-2023].

[3] “Apple Books,” *Apple*. [Online]. Available: https://www.apple.com/apple-books/. [Accessed: 20-Apr-2023].

[4] “Software engineer salary in Dallas, TX,” *Levels.fyi*. [Online]. Available: https://www.levels.fyi/t/software-engineer/locations/greater-dallas-area. [Accessed: 20-Apr-2023].

[5]“Average size of a kindle book,” Elite Authors, 16-Feb-2023. [Online]. Available: https://eliteauthors.com/blog/the-average-size-of-a-kindle-book/. [Accessed: 21-Apr-2023].

[6]“How much does a server cost for a large business?,” Abacus, 17-Dec-2021. [Online]. Available: https://goabacus.com/how-much-server-cost-large-business/. [Accessed: 21-Apr-2023].