CS3354 Software Engineering

Final Project Deliverable 2

BookWorm

David Brunner (deb210004@utdallas.edu)

Zachary Taylor (zwt190000@utdallas.edu)

Kaiden Gallardo (kkg200001@utdallas.edu)

Sam Salinas (sas200000@utdallas.edu)

Joseph Egan (jee210001@utdallas.edu)

Project Scope

University Library Web Application

1. **Student Account**
    a. Login to website
    b. Pay fines

2. **Literature Searching**
    a. Search for database
        i. Select subject category
        ii. Select vendor type
        iii. Select if available for user
        iv. Type in name search query
        v. View description
    b. Search for article
        i. Select subject category
        ii. Select if scholarly or non-peer reviewed
        iii. Select if available for user
        iv. Type in title or author query
        v. View citation
        vi. View abstract
        vii. View publishing details
    c. Search for book
        i. Select subject category
        ii. Select if online or print
        iii. Select if available for user
        iv. Type in title or author search query
        v. View citation
        vi. View summary
        vii. View publishing details

3. **Literature Viewing**
    a. Database
        i. Click on external link
    b. Article
        i. Search for word and go to word
        ii. Go to section
        iii. Go to next/previous page
        iv. Change font and size of text
    c. Book
        i. Search for word and go to word
        ii. Go to chapter

iii. Go to next/previous page
iv. Change font and size of text
v. Checkout for in-person pickup if available
vi. Add to personal collection if online and in library catalog

4. **Personal Collection**
    a. Category
        i. Create category
        ii. Delete category
        iii. Search category for book
    b. Book
        i. Add book to category
        ii. Remove book from category

5. **Book Editing**
    a. Notate
        i. Add text note
        ii. Edit text note
        iii. Delete text note
    b. Bookmark book page
        i. Add bookmark
        ii. Edit bookmark
        iii. Delete bookmark
    c. Highlight text passage
        i. Add highlight
        ii. Edit highlight
        iii. Delete highlight

6. **Study Room Reservation**
    a. Search available rooms
        i. Select date
        ii. Select group size
    b. Confirm reservation for a room
    c. Cancel reservation for a room

PROJECT DELIVERABLE 1 CONTENT

1. **Project Draft Description:**

Our team has decided on an interactive library platform as the topic for our project. The application should allow a user to access online content using databases, bookmark resources, upload/download book files from local systems, and annotate their saved documents. Ideally, the program can be used by educational institutions, providing their students with easy access to online resources and note-taking features for their class material.

The primary motivation for choosing this topic was its scope, allowing us to focus on different components such as the database, user interactivity, cloud computing, and the legality of online content. Nevertheless, we are also students so using an online library is familiar to all of us.

We currently do not plan to author a scholarly article after completing the project.

Instructor Feedback:

Good choice for a topic! A comprehensive library management system truly is useful and will promote student and faculty success, or individual engagement into more reading. As a suggestion, please contact UTD Library staff to get more information about the details of the working of the current system, what could be improved, etc. for a better design.

It is great to see a detailed breakdown of the tasks you have considered 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.

2. **GitHub Repository Information:**

   https://github.com/DavidB09/3354-bookworm

3. **Delegation of Tasks:**

   Zachary will take on the role of project manager, keeping the team on schedule and ensuring effective work.

   David will take on the role of secretary, creating/submitting the project documents, setting up any online platforms, and ensuring effective communication.

   Kaiden will mainly focus on researching important topics and supplying resources for the team.

   Sam will focus on the project presentation, creating/uploading the slides and ensuring everyone is prepared.

   Joseph will focus on helping Kaiden with researching topics and designing graphics for the project.

   It is important to note that everyone is expected to contribute to the writing of any documents or code involved with the project. Nevertheless, the previously stated roles may also change as the team progresses throughout the semester, ensuring that we will meet the deadlines for all deliverables.

4. **Software Process Model:**

   To complete BookWorm, the incremental software process will be employed because of the following list of reasons. The diagram in Figure 1 is a graphical example of BookWorm's incremental software process model.

   - BookWorm includes a repository containing access to databases, articles, and literature, meaning to have a selection of increasing size, constant updates to all components will be made as they become available.

- To release a working version of BookWorm as soon as possible, the main features, including managing a student's account, searching for specific databases, articles, and books, and viewing databases, articles, and books, will all be released before completed versions of the editing, personal collection, and study room reservation components are fully published.

- All members of the BookWorm team will be able to fully work on multiple pieces of a single component, as well as previous bug fixes, without having to worry about working in parallel with their other team members due to the linear format of the process.

- All database, article, and literature licensing available will start out small and grow to the needed size depending on the initial impressions of BookWorm. This means that if BookWorm gains a notable, frequent audience, license sizes will increase as new increments are released.

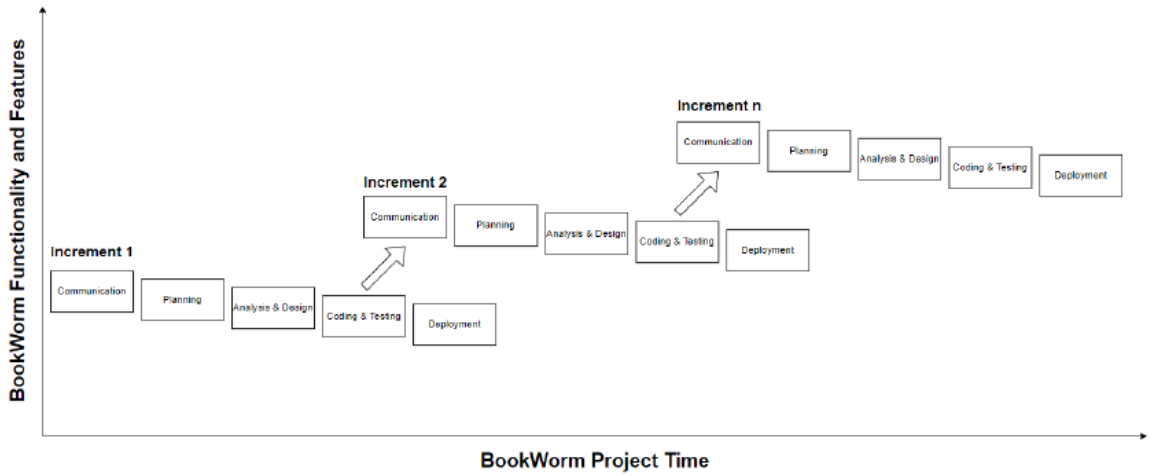


Figure 1: Incremental Software Process Model for BookWorm

5. **Software Requirements:**

Functional Requirements

1. Users

a. Users should be able to log into their university account and pay fines.

b. Authenticated users should be able to search for specific databases, articles, and books by subject category, vendor type, online or physical, availability, title, author, and academic status.

c. Authenticated users should be able to view and navigate databases, articles, and books and see their description, publishing details, citation, and summary.

d. Authenticated users should be able to create a personal collection of books that can be added and removed from personally created categories.

e. Authenticated users should be able to reserve a study room by date and size if available.

f. Authenticated users should be able to edit their copies of books by adding notes, bookmarks, and highlights.

g. Authenticated users should be able to download licensed copies of databases, articles, and books.

Non-Functional Requirements

Usability

- The system should be easily navigable by users so that the number of help requests does not exceed 10 per hour.

Performance

- Each system request should be processed within 4 seconds.

Space

- Memory usage of the site should never exceed 1 gigabyte.

Dependability

- The site should load in 5 seconds when the number of users is less than 1000 and no more than 10 seconds when the number users exceed 5000.

Security

- The user login feature should only allow a maximum of 5 login attempts before locking the user out for 15 minutes.

Environmental

- The repository server should be kept in an area that allows it to stay within the range of 20 to 60 degrees Celsius at all times.

Operational

- The system must maintain a minimum of 20 users per day to be kept in operation. If this is not maintained for a total of 30 days, the system will shut down.

Development

- The system will be programmed in java using object-oriented programming while also using html, CSS, and JavaScript for the website.

Regulatory

- All databases, articles, and books must be checked for their relation to academics in some way shape or form to be kept in the repository.

Ethical

- All databases, articles, and books must be properly licensed in order to prevent copyright infringement and piracy.

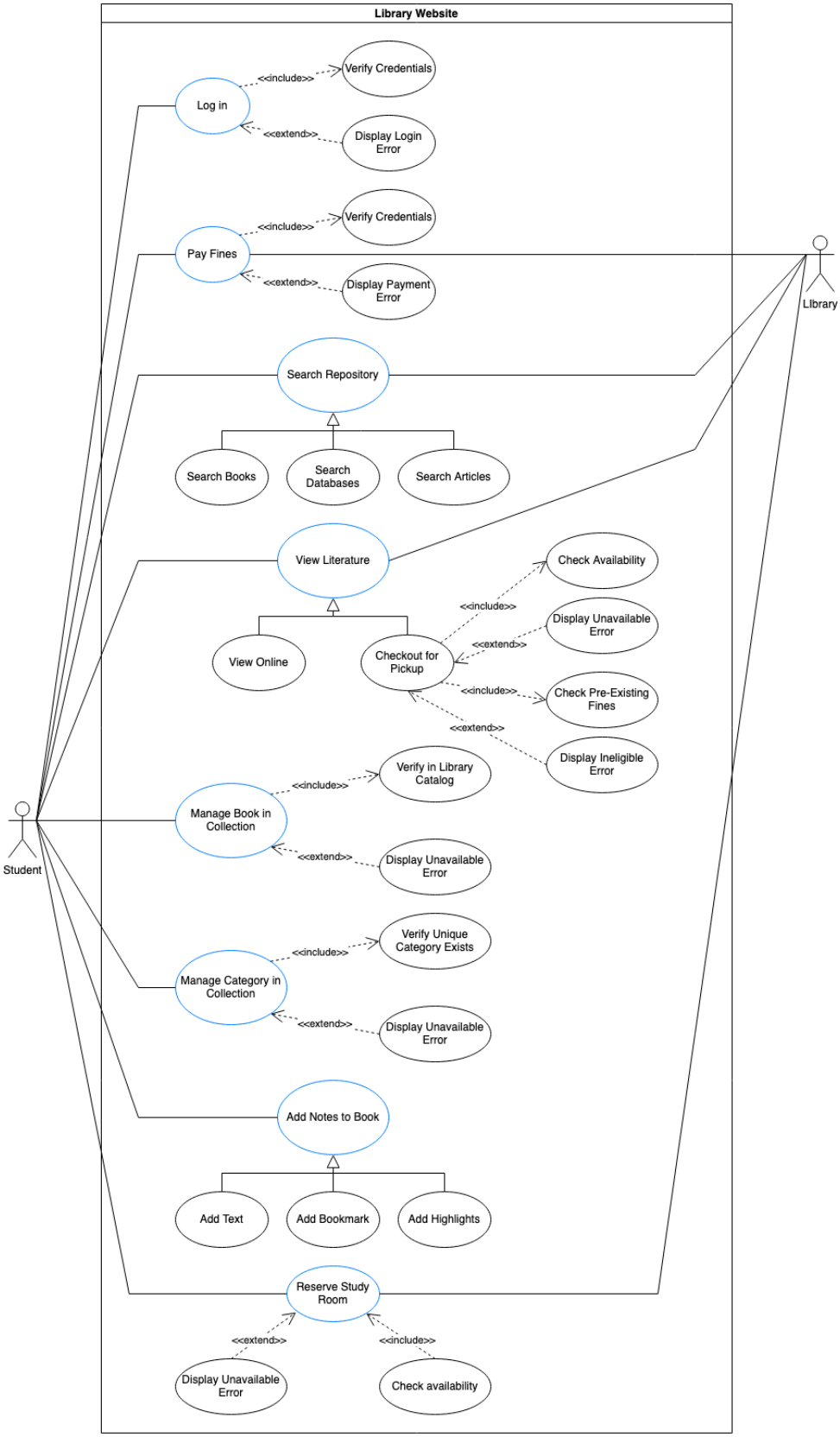
Accounting

- The upkeep of all systems must not exceed $100,000 a year, including the cost of licenses, webhosting, and employment.

Safety/Security

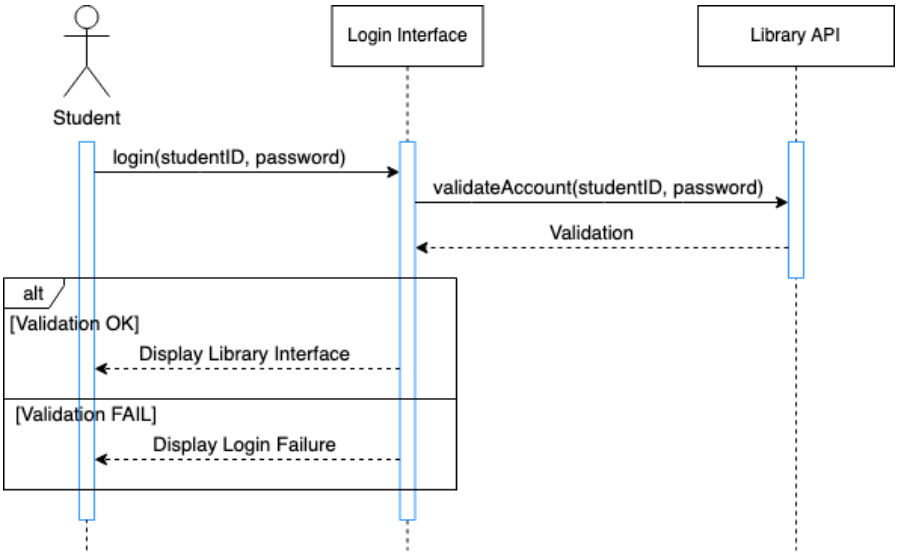
- All databases, articles, and books must be scanned for malware and viruses by an expert beforehand to ensure that users are not exposed to harm.

6. Use Case Diagram:

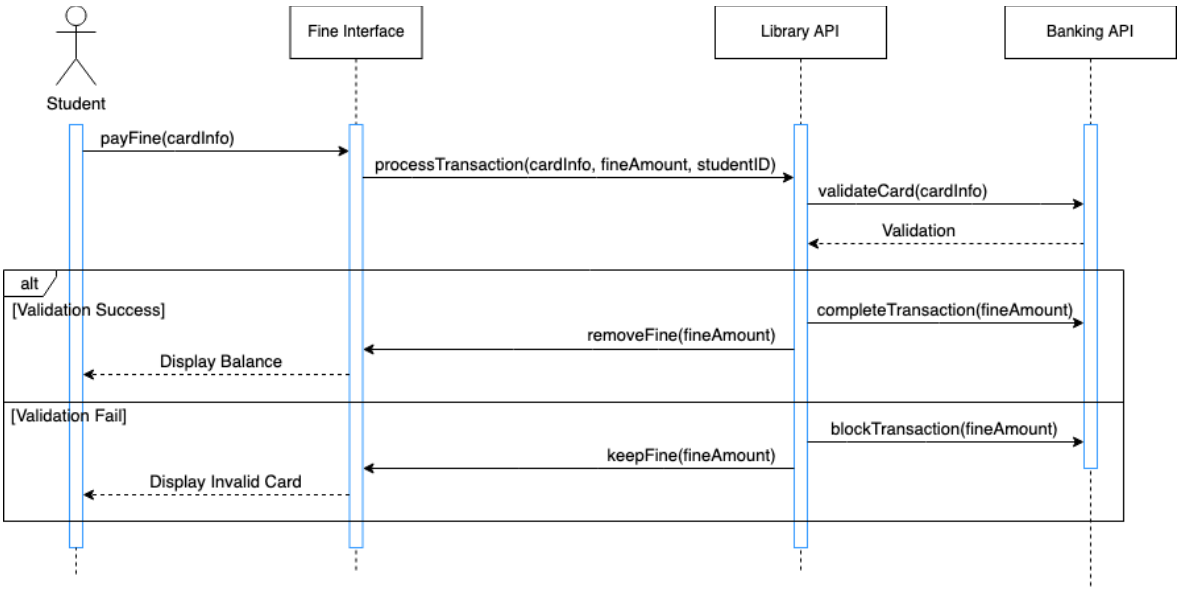


7. Sequence Diagrams:

User Login:

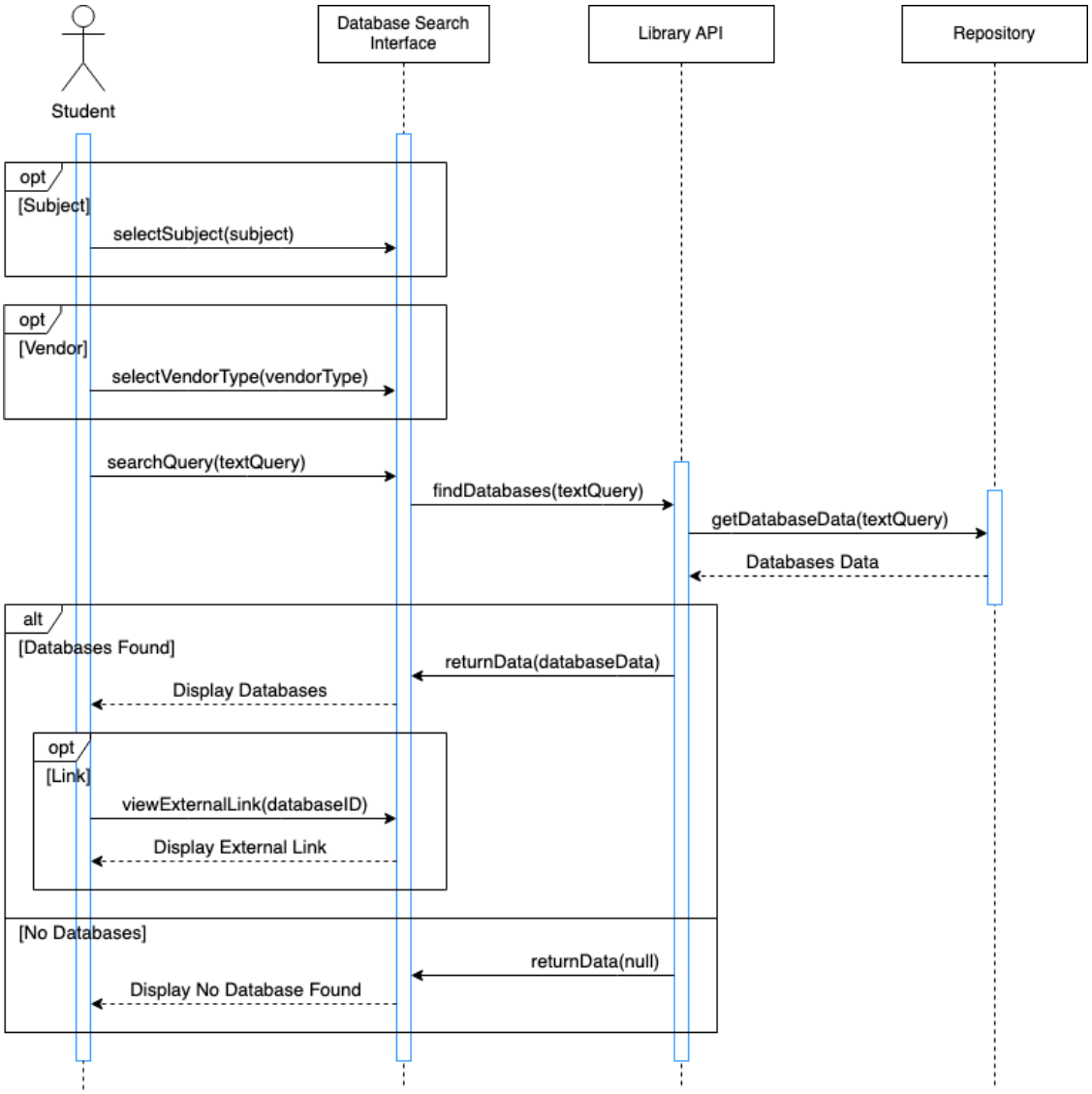


User Pay Fine:

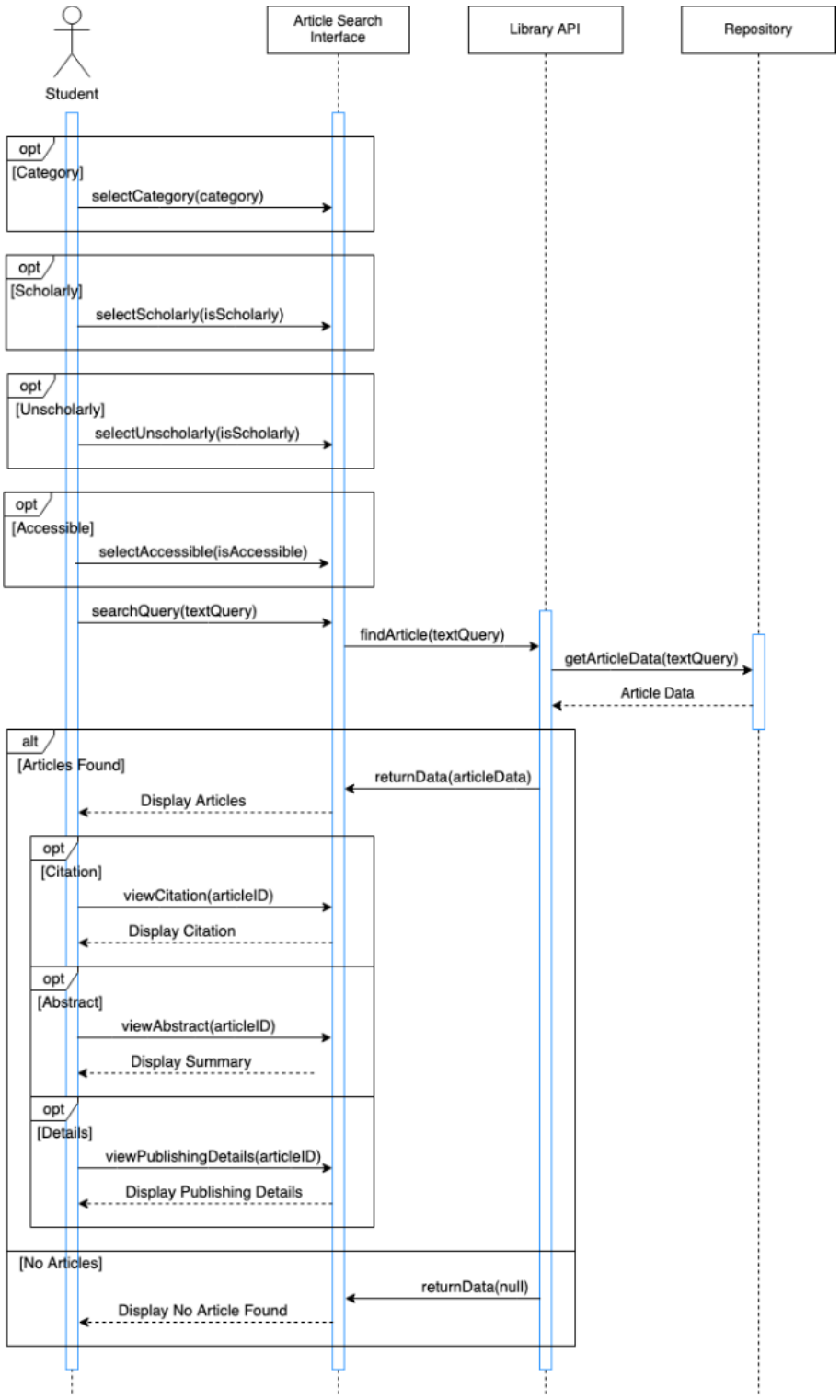


User Search:

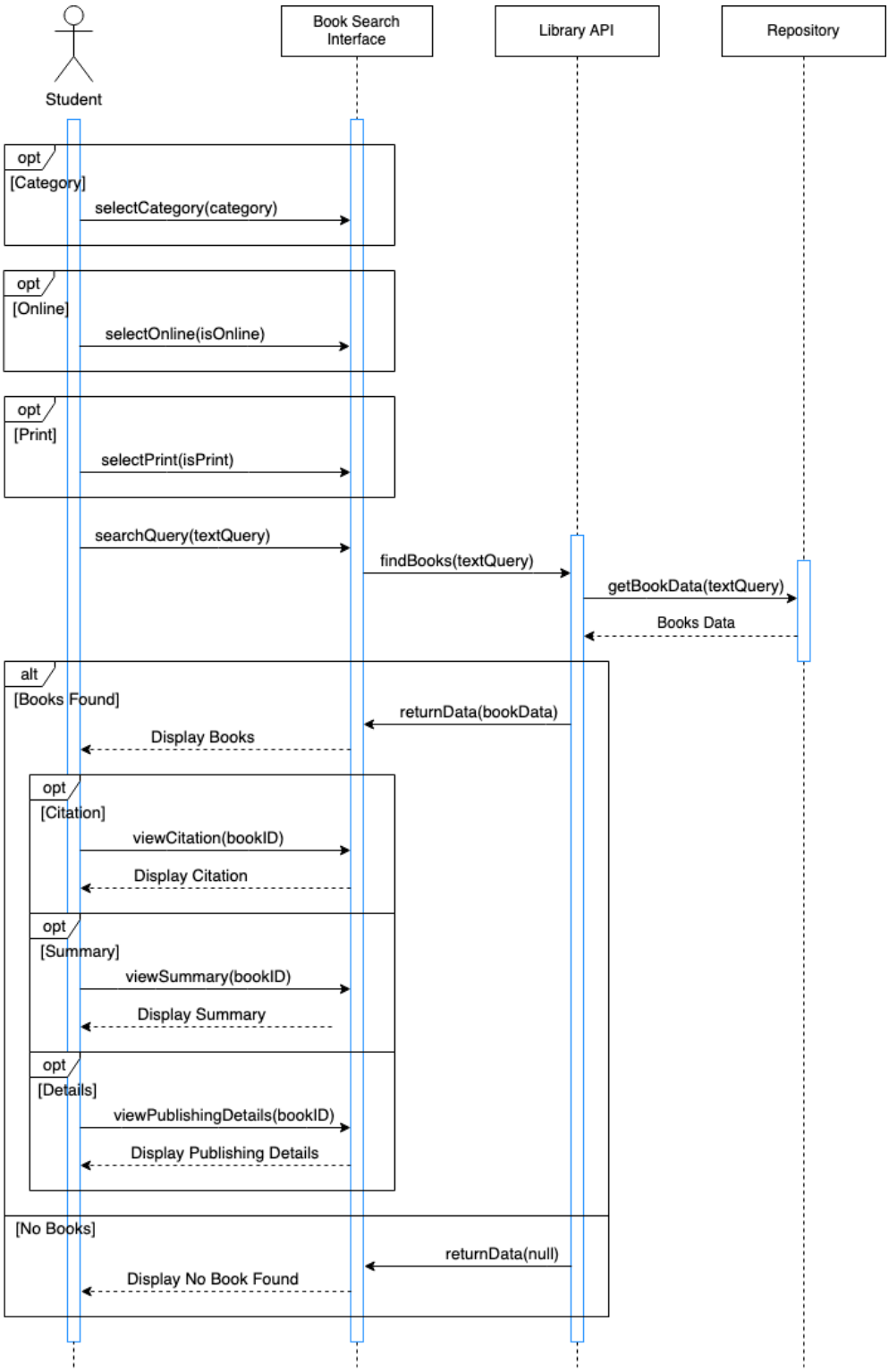
Database Search:



Article Search:

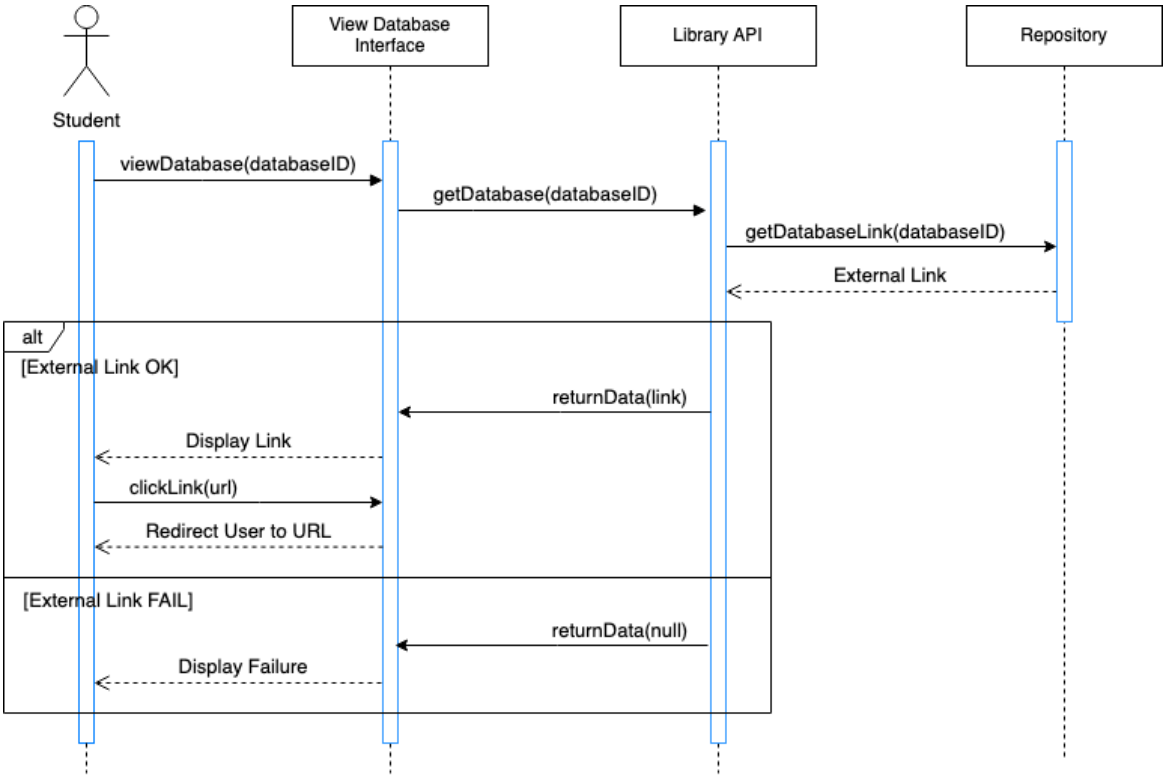


Book Search:

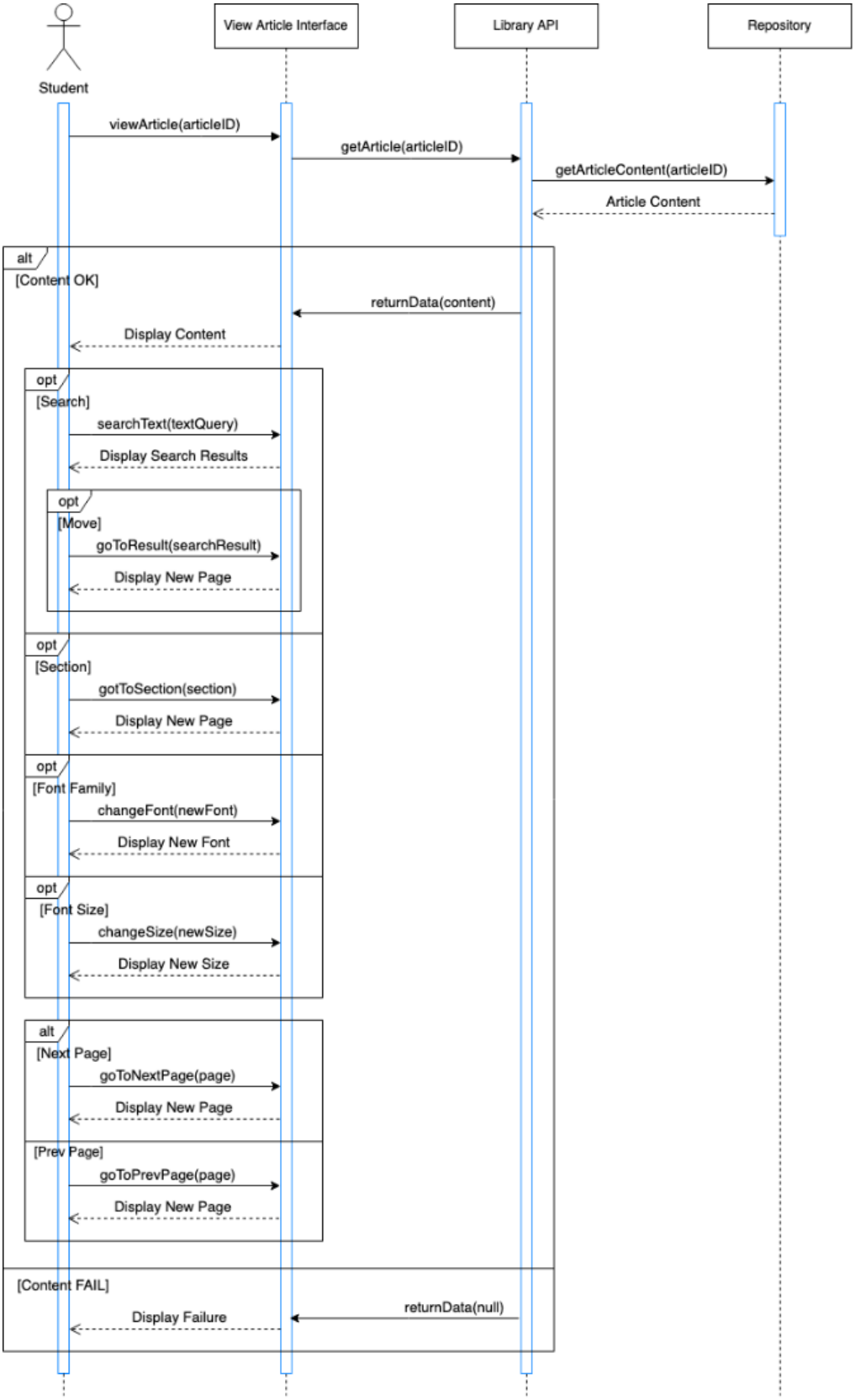

Student
Book Search Interface
Library API
Repository
opt
[Category]
selectCategory(category)
opt
[Online]
selectOnline(isOnline)
opt
[Print]
selectPrint(isPrint)
searchQuery(textQuery)
findBooks(textQuery)
getBookData(textQuery)
Books Data
alt
[Books Found]
returnData(bookData)
Display Books
opt
[Citation]
viewCitation(bookID)
Display Citation
opt
[Summary]
viewSummary(bookID)
Display Summary
opt
[Details]
viewPublishingDetails(bookID)
Display Publishing Details
[No Books]
returnData(null)
Display No Book Found

User View:

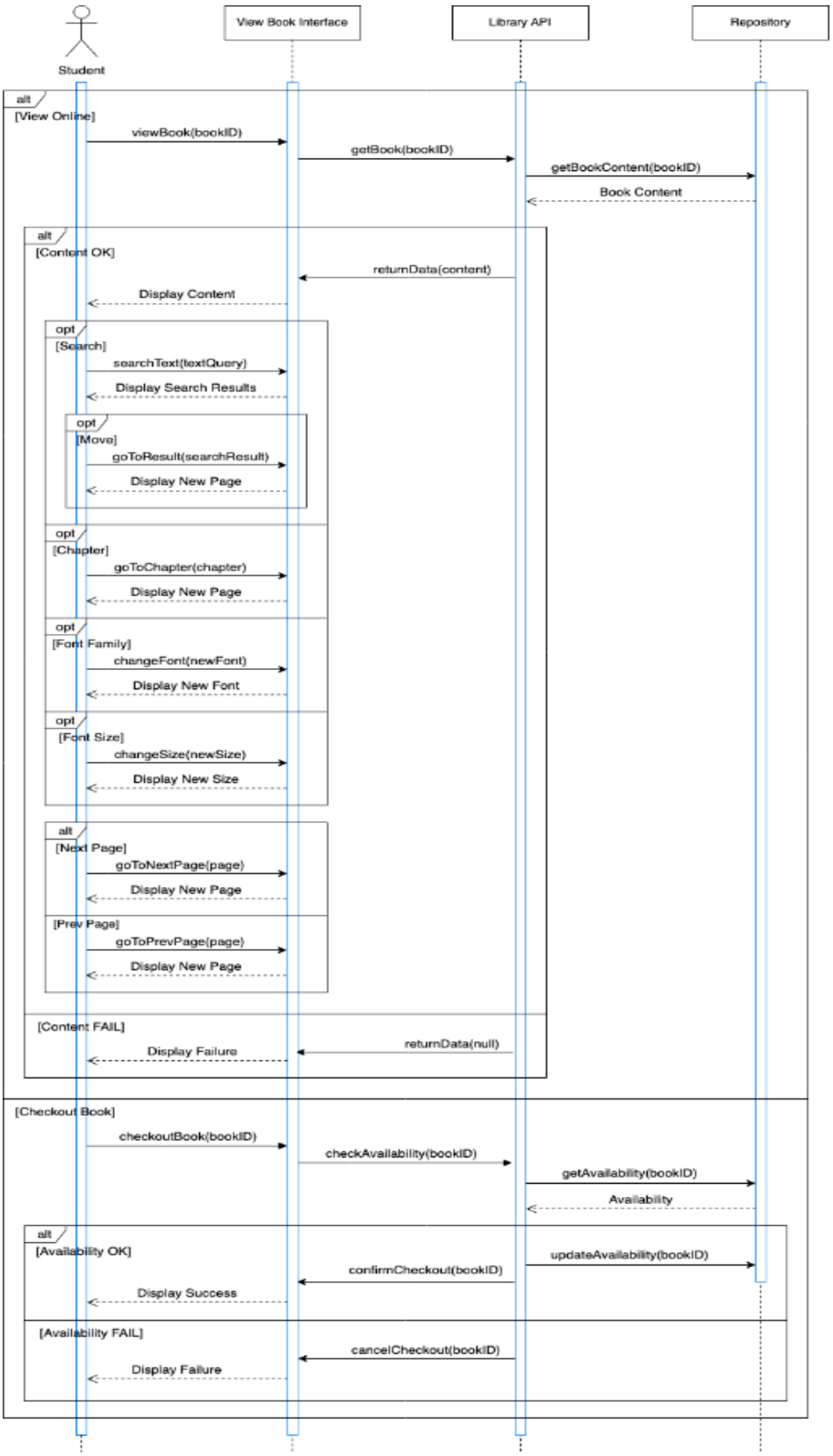
Database View:



Article View:

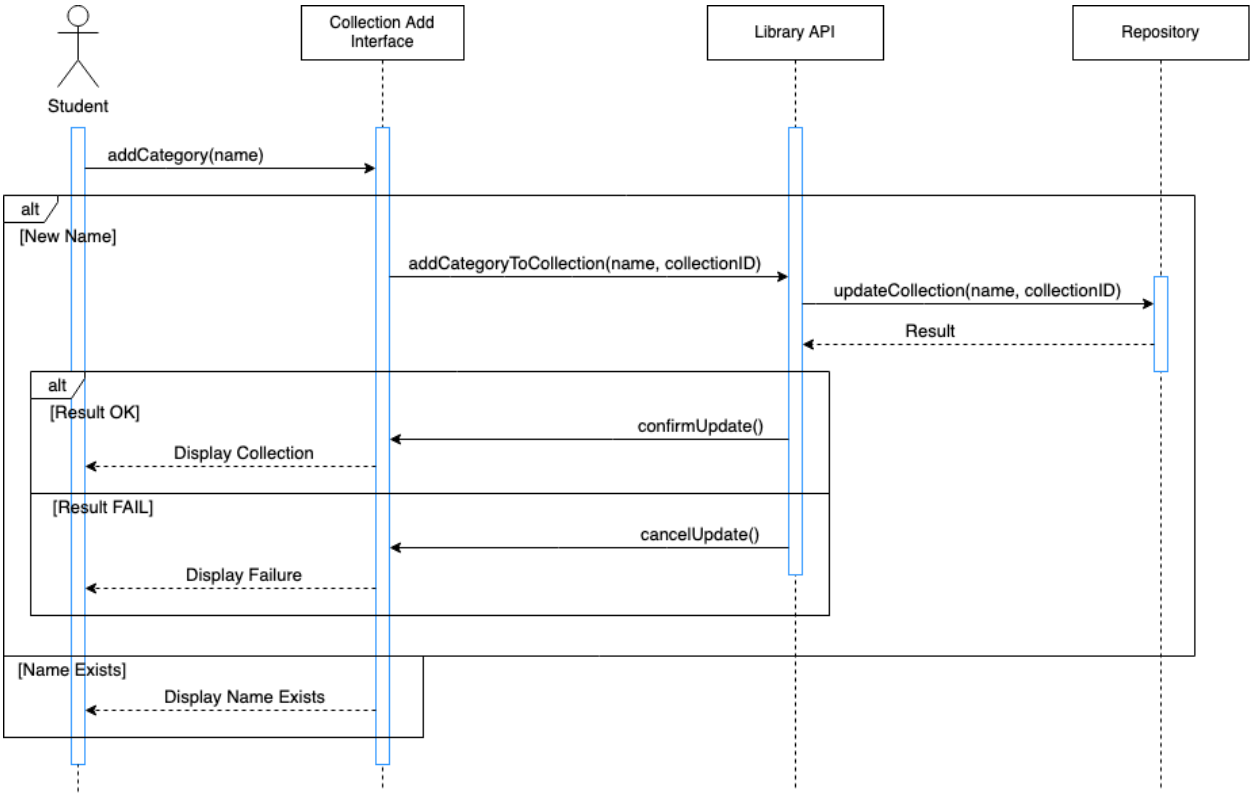


Book View:

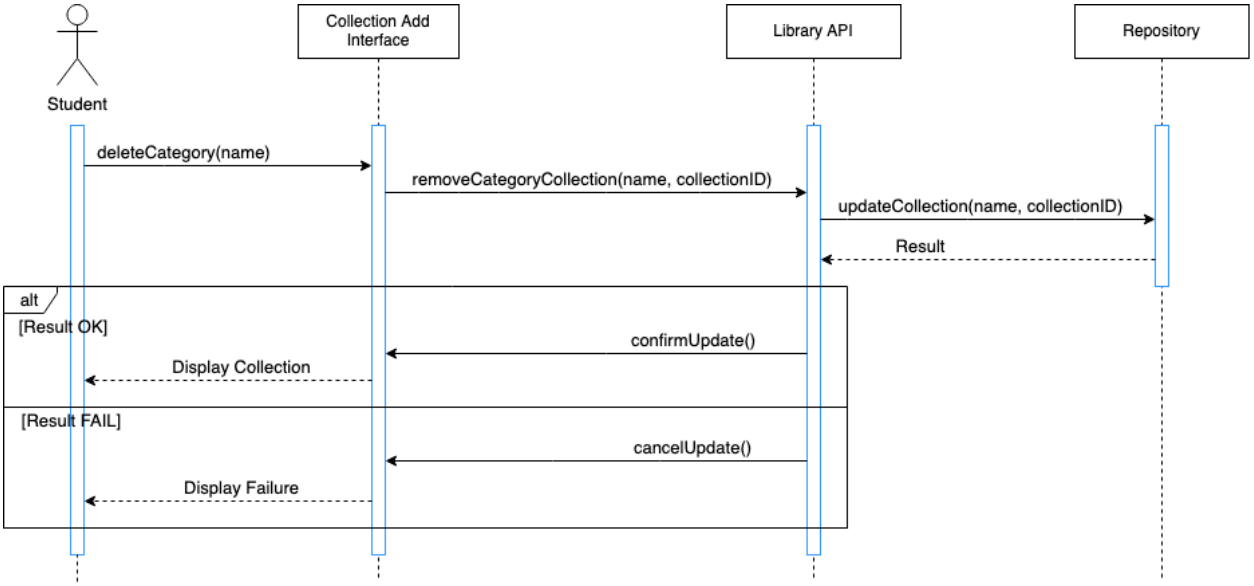

Student
View Book Interface
Library API
Repository
alt
[View Online]
viewBook(bookID)
getBook(bookID)
getBookContent(bookID)
Book Content
alt
[Content OK]
returnData(content)
Display Content
opt
[Search]
searchText(textQuery)
Display Search Results
opt
[Move]
goToResult(searchResult)
Display New Page
opt
[Chapter]
goToChapter(chapter)
Display New Page
opt
[Font Family]
changeFont(newFont)
Display New Font
opt
[Font Size]
changeSize(newSize)
Display New Size
alt
[Next Page]
goToNextPage(page)
Display New Page
[Prev Page]
goToPrevPage(page)
Display New Page
[Content FAIL]
returnData(null)
Display Failure
[Checkout Book]
checkoutBook(bookID)
checkAvailability(bookID)
getAvailability(bookID)
Availability
alt
[Availability OK]
updateAvailability(bookID)
confirmCheckout(bookID)
Display Success
[Availability FAIL]
cancelCheckout(bookID)
Display Failure

Category Management:

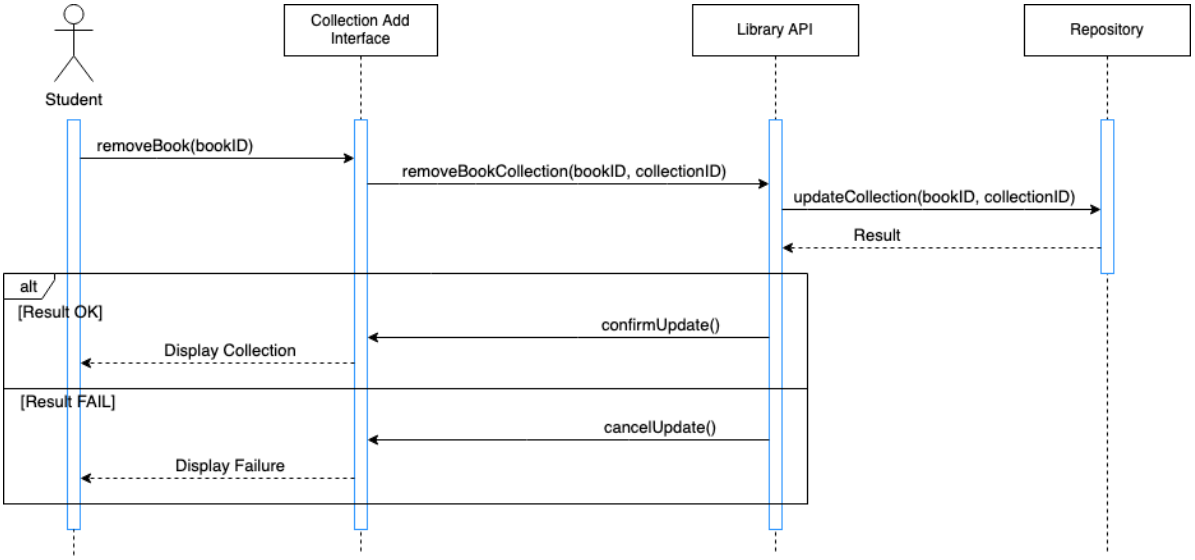
Add Category:



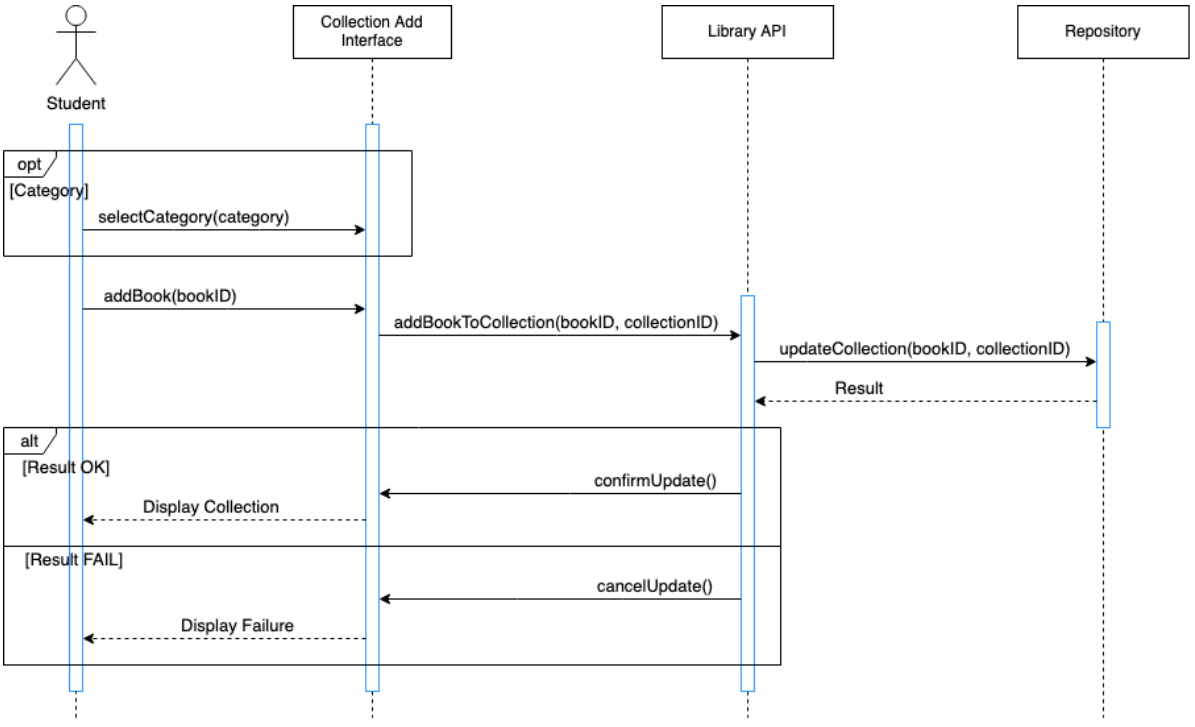
Delete Category:



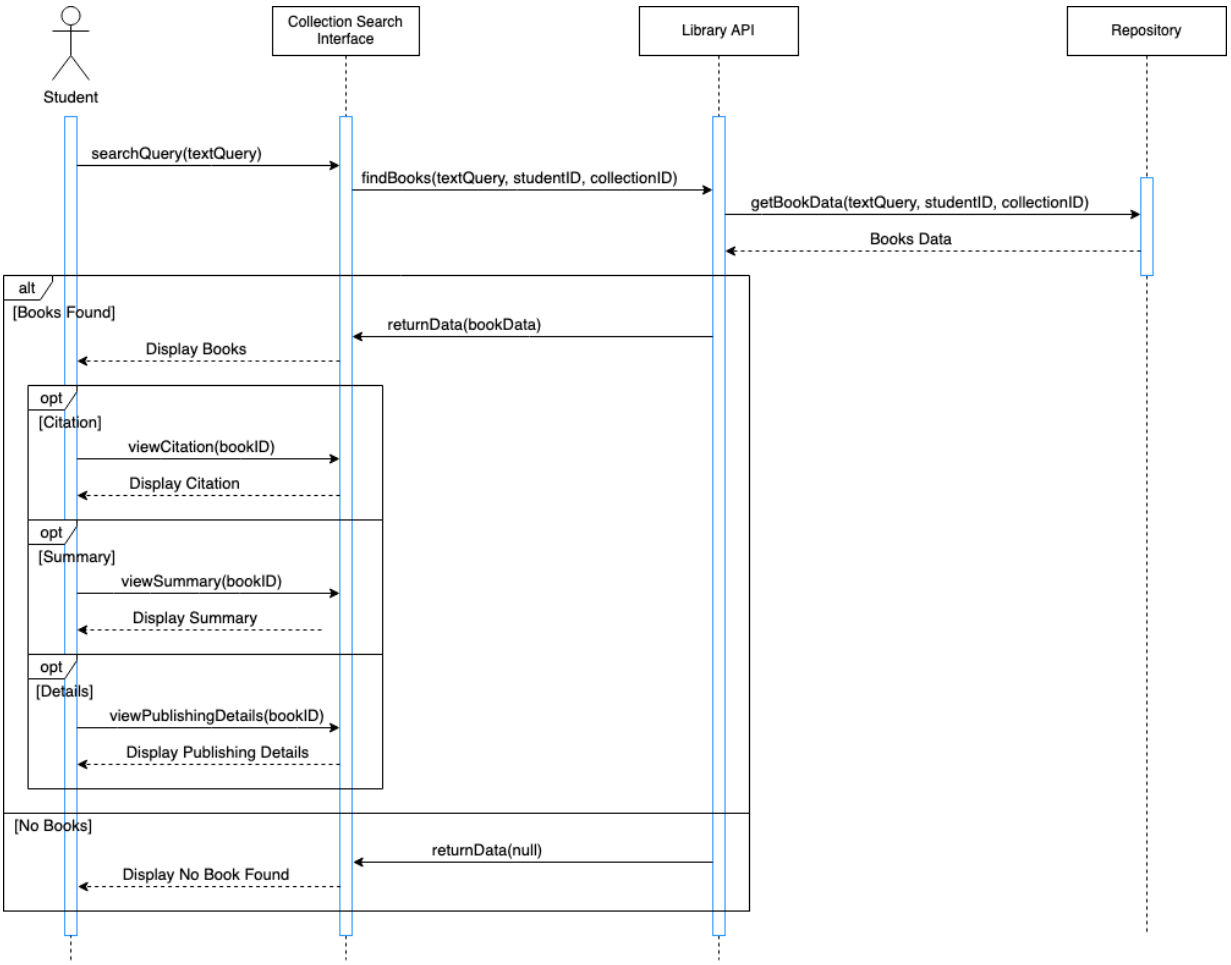
Remove Book from Collection:



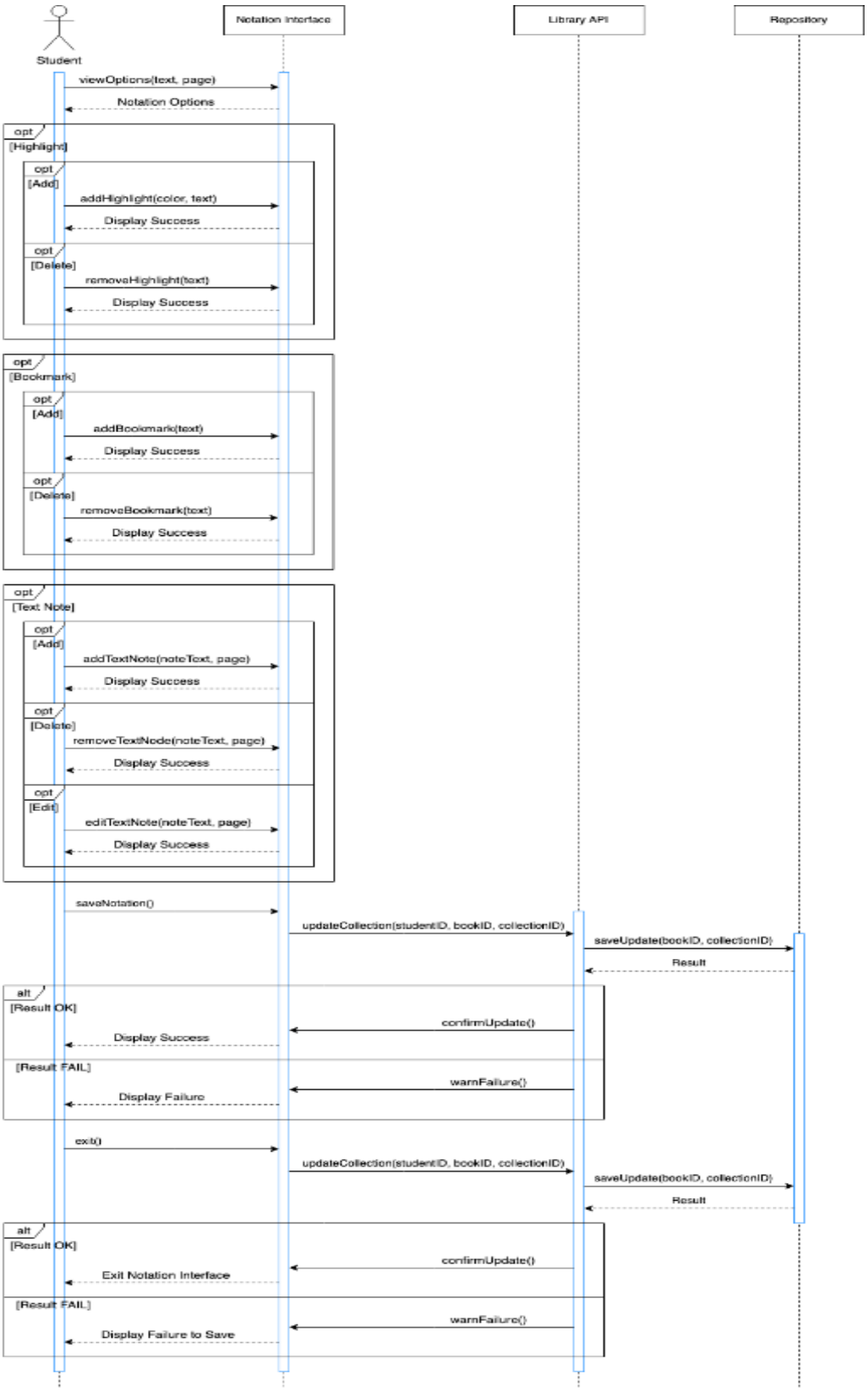
Add Book to Collection:



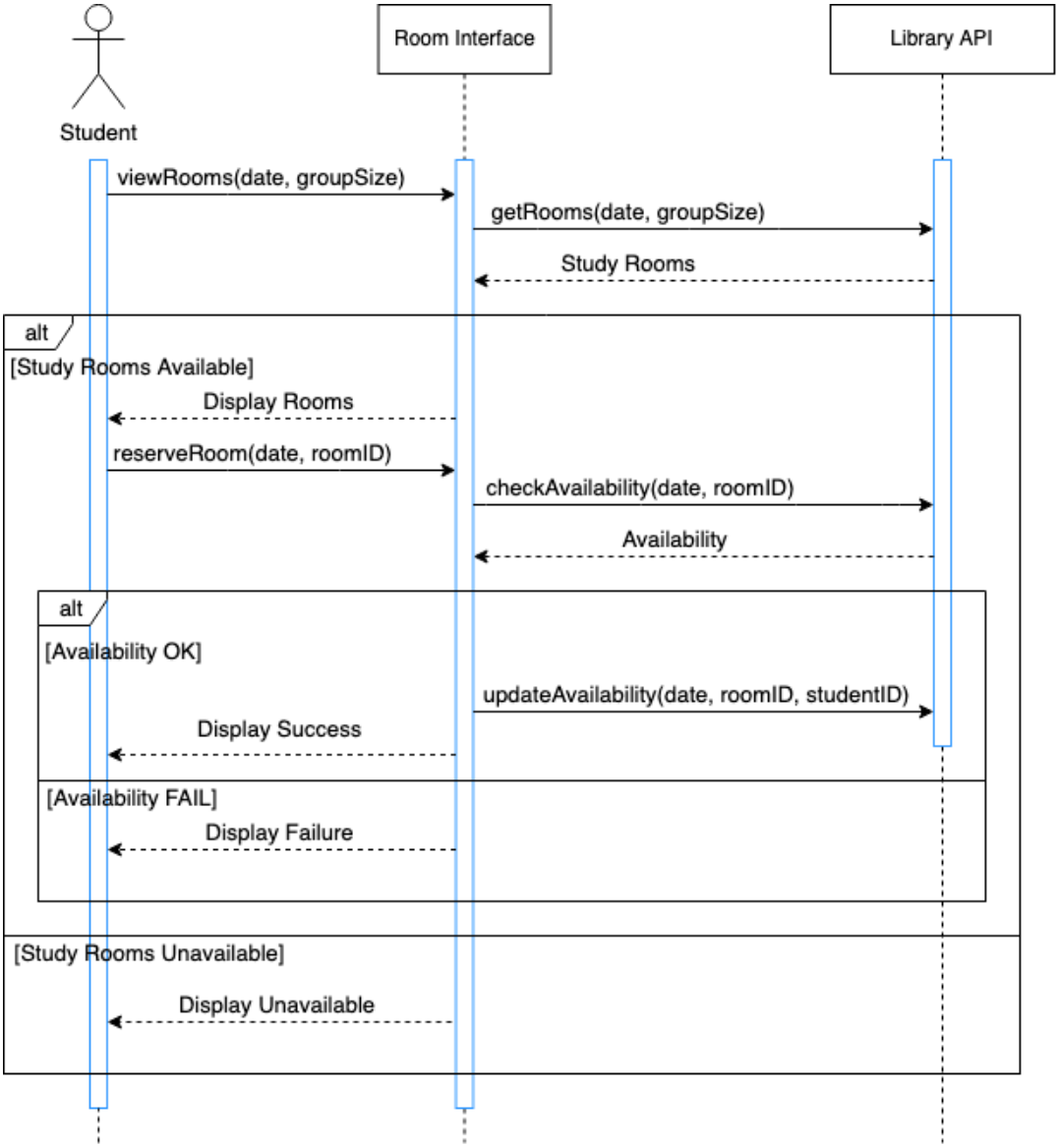
Search Collection



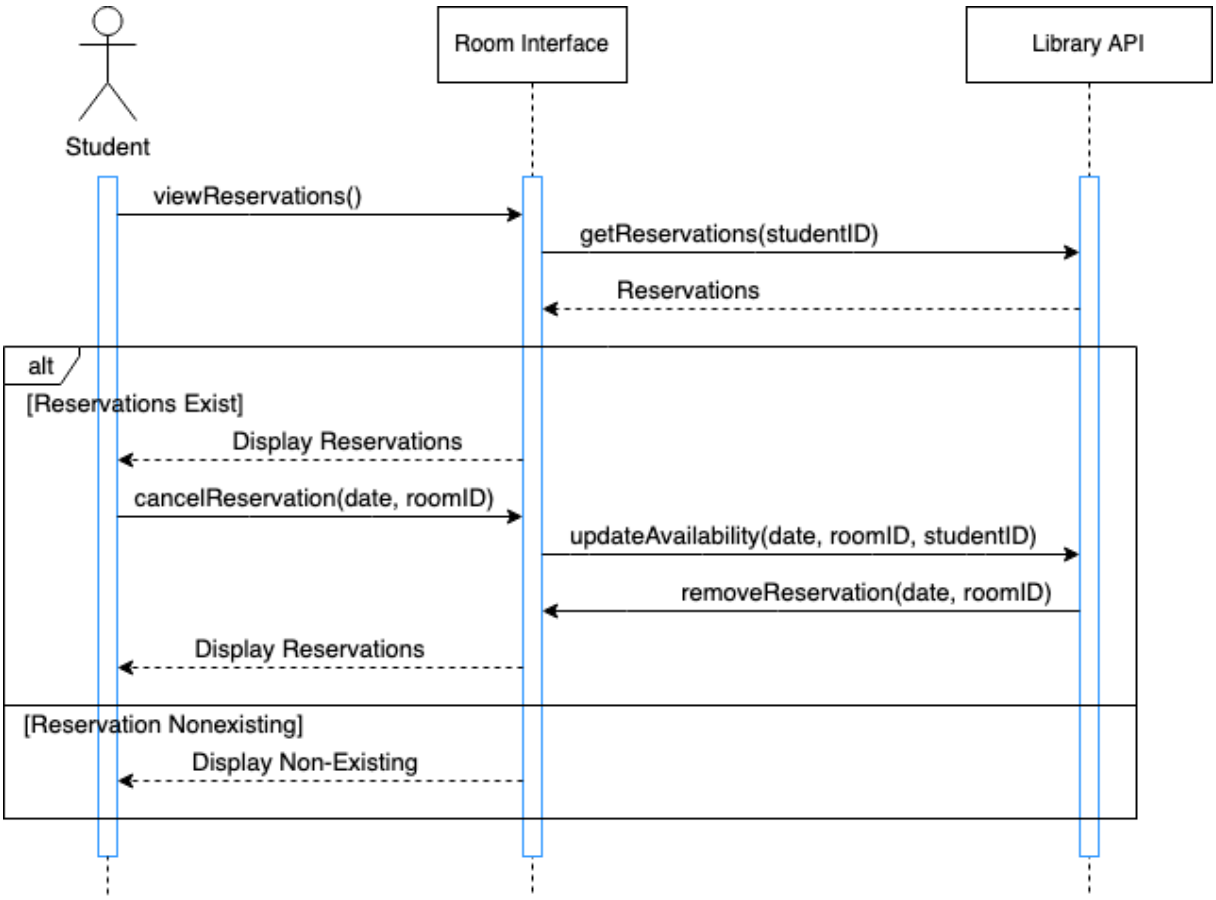
Book Notating:



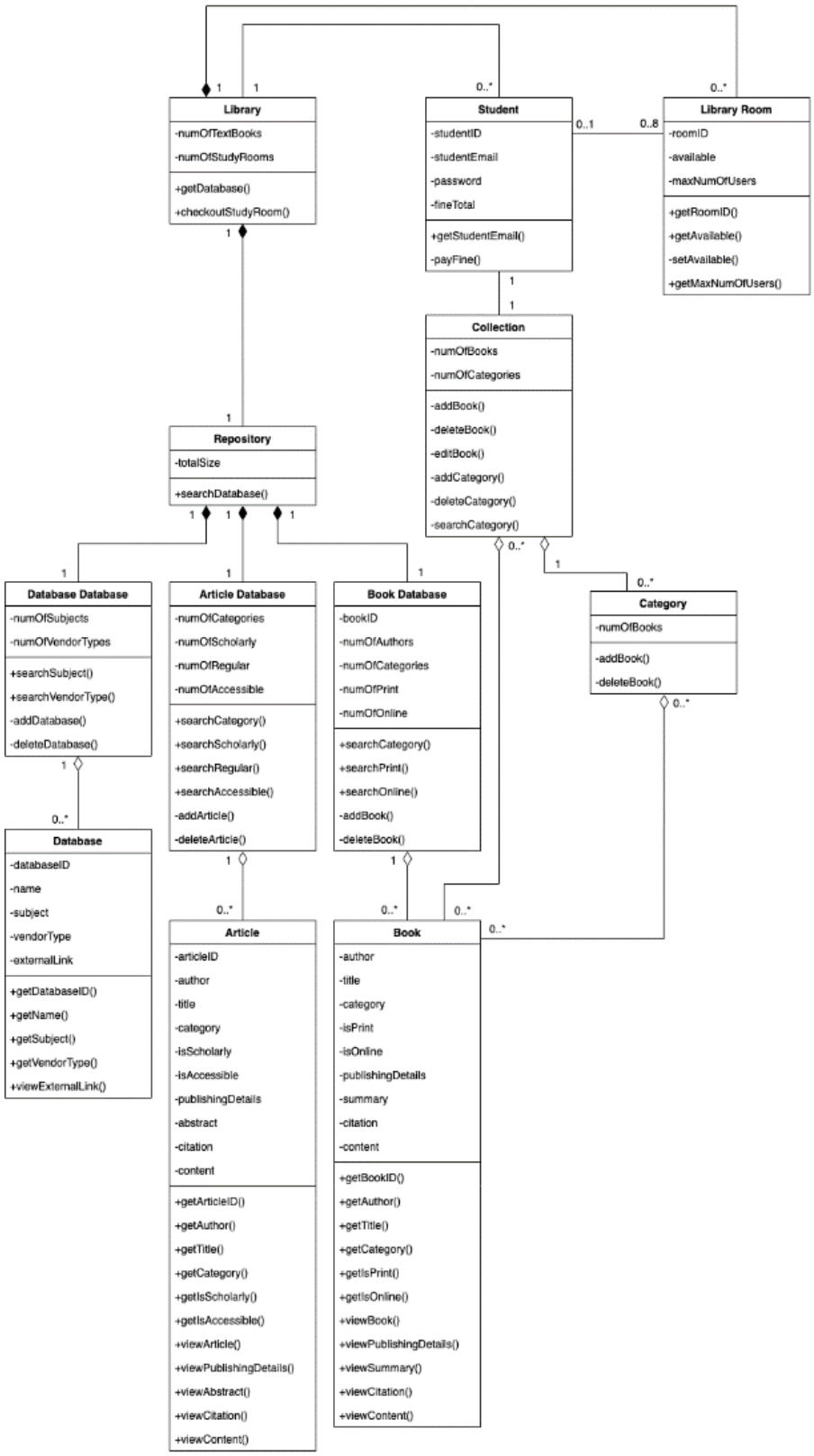
Reserve Study Room:



Cancel Study Room Reservation:

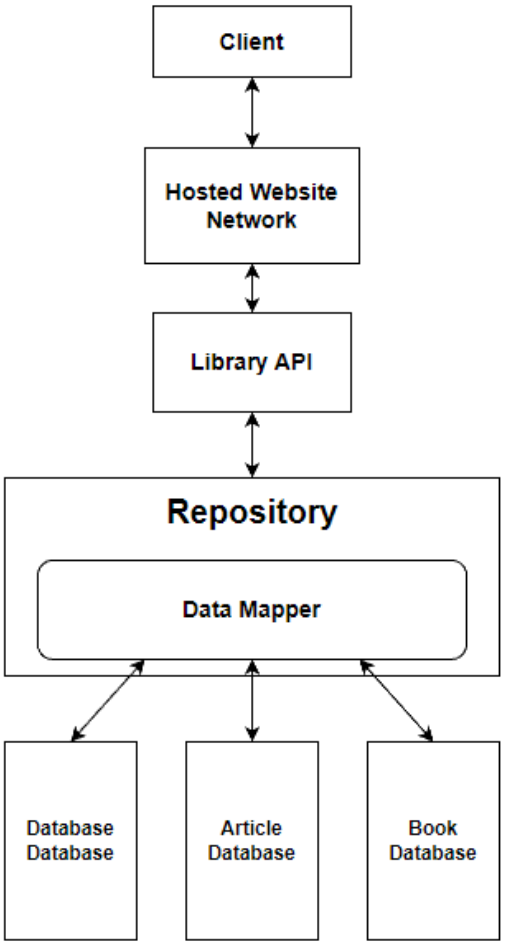


8. Class Diagram:



9. Architectural Design:

Repository Architecture Pattern:



PROJECT DELIVERABLE 2 CONTENT

10. Project Scheduling, Cost, Effort, and Pricing Estimation

a. Project Scheduling

The estimated duration was calculated using the Function Point method, using the values from the table seen in the next section.

Estimated Duration: 2 weeks

Expected Duration: 3 weeks

Start Date: 5/08/2023

End Date: 5/29/2023

Important Factors:

- The development team is inexperienced; hence the expected duration is longer than the estimated duration.
- Because the development team is inexperienced, the duration of the project has been extended to account for learning the systems and tools that the team will be working with, as well as the bugs and fixes that will be encountered along the way.
- Development time will only account for work done on the weekdays, and not weekends.
- Each developer will be required to work for 8 hours a day on each weekday.

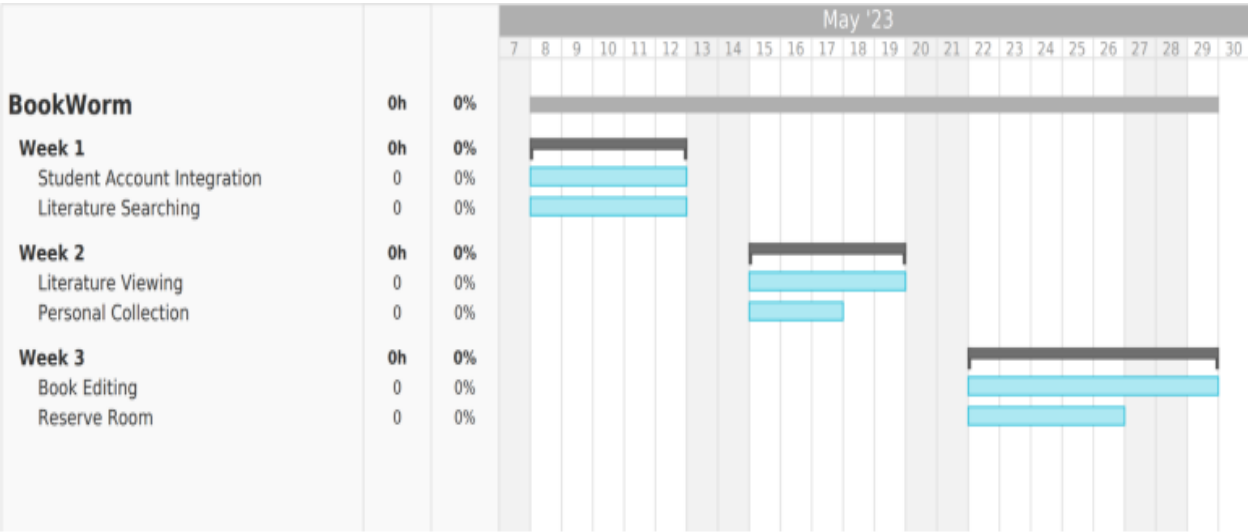
Estimated Implementation Duration:

- Student Account Integration (3 to 5 days)
- Literature Searching (3 to 5 days)
- Literature Viewing (3 to 5 days)
- Personal Collection (2 to 3 days)
- Book Editing (5 to 6 days)
- Student Room Reservation (3 to 5 days)

Overall Project Schedule & Scheduling Notes:

- Students must be able to search for the literature before they are able to view it or add it to their collection. Those items must be completed sequentially.
- Students must be able to view the literature before they can edit it. Those items must be completed sequentially.

- Account Integration and Room reservation can be completed in parallel with any other task, so those tasks are flexible.



b. Cost, Effort, and Pricing Estimation

The team decided to use function point for our algorithm estimation technique.

For the parameter numbers, we decided to go through the sequence diagrams and count all the different input options along with the corresponding output within each view or system component. As a result, the user input and output counts are high since the library website is quite dependent on user interaction. Nevertheless, most of those functionalities are simple to implement.

We also counted the number of user queries as each time the user needs to look for a particular item within the library database, as well as their own collection. These lookups have an average complexity of implementation.

Since the data files and relational tables are unique databases and directories, we decided to only include the user collection database. However, we counted the books, articles, and database directories along with the central repository for our external interfaces. Their implementation complexities are mid-tier and difficult since they require efficient traversing whenever a user query occurs.

For each processing complexity, we figured that the data aspects and online data access are very important to our program. Nevertheless, the internal processing to make database searching efficient and the overall user experience of using the site is also essential to the overall usability. All the other complexities are average, with installation-related functionality having no importance since the program is utilized online.

	Function Category	Count	Simple	Average	Complex	
1	User Input	55	3	4	6	55 * 3 = 165
2	User Output	49	4	5	7	49 * 4 = 196
3	User Queries	4	3	4	6	4 * 4 = 16
4	Data Files and Relational Tables	1	7	10	15	1 * 10 = 10
5	External Interfaces	5	5	7	10	5 * 10 = 50

Gross Function Point = 437

PC1: 5

PC2: 4

PC3: 3

PC4: 2

PC5: 3

PC6: 5

PC7: 3

PC8: 3

PC9: 2

PC10: 5

PC11: 1

PC12: 0

PC13: 0

PC14: 5

PCA: 0.65 + 0.01(5 + 4 + 3 + 2 + 3 + 5 + 3 + 3 + 2 + 5 + 1 + 5) = 1.06

Function Point: 437 * 1.06 = 463.22 FP

Estimated Effort: 463.22 / 60 = 7.7 ≈ 8 person-weeks

Duration: 8 / 5 = 1.6 ≈ 2 weeks

c. Estimated Cost of Hardware Products

The only hardware cost that we will require is 5 desktops, resulting in an upfront cost of around $2,000.

d. Estimated Cost of Software Products

The main software cost for the library program is the database, namely the software that contains all of the books, articles, and database links along with their semantic information. Using an average of 50 TB storage along with 25 TB download data per month, the total cost will be around $10,000/month [1].

The second largest expense would be the server that hosts the website. Since speed and availability are very important factors in the success of the application, we decided to select the expensive option and rent a dedicated server for approximately $100/month [2]. The larger storage space and RAM is worth the extra cost, providing students with quicker access with less downtime [2].

Nevertheless, for developing and maintaining the code, our team also chose to invest in the GitHub Enterprise plan. This costs around $21/month per user, resulting in around $84/month for our team during the development phase, and $42/month during the development phase [3].

e. Estimated Cost of Personnel

We decided to employ a team of four software developers and one team manager for the development of the library software. After release, two software developers will continue to maintain the program and incrementally add new features as time progresses.

Overall, reflecting the average rates across the United States, we will pay our software developers around $52/hour and the manager will receive $65/hour [4] [5]. Thus, our development phase will cost around $24,570 assuming the project takes 2 weeks, and everyone works 45 hours per week.

11. Test Plan

a. Pay Fine Module

The test plan created consists of code for the “Pay Fine” module of BookWorm which can be found in the above pages. This module is specifically designed to follow the sequence diagram that can be found in the sequence diagram section of the deliverable.

The following code is written in Java and tested using Junit.

```
class Student {
    double fineAmount;

    public Student() {
        fineAmount = 0;
    }

    public void increaseFine(double amount) {
        if (amount > 0)
            fineAmount += amount;
    }

    public double payFine(double amount) {
        if (amount > fineAmount)
            return -1;

        return fineAmount -= amount;
    }
}
```

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class StudentTest {
    @Test
    public void testFinePayment() {
        Student student1 = new Student();
        Student student2 = new Student();
        Student student3 = new Student();

        student1.increaseFine(13.5);
        student2.increaseFine(5);
        student3.increaseFine(2.75);

        assertEquals("Testing smaller payment", 7.25, student1.payFine(6.25), 0.0001);
        assertEquals("Testing larger payment", -1, student2.payFine(45), 0.0001);
        assertEquals("Testing equal payment", 0, student3.payFine(2.75), 0.0001);
    }
}
```

12. Comparison with Similar Designs

The Library Management Systems space seems to be dominated by a couple key players, primarily Alma by Ex Libris, BiblioCommons, and Polaris by innovate. While BiblioCommons and Polaris are focused on the space of public library management and Alma is designed for academic library management, both of them offer interesting comparisons to our project.

BiblioCommons is the closest example to the project we are developing. It features a simple system to manage resources and books so that users of the library can search through the catalog. This is similar to our project in the simple design of the program. BiblioCommons is also highly adaptable to communicating with different library systems, similar to our project. One of the special features of BiblioCommons is the social tagging system and algorithmic suggestions of books [6]. This turns the BiblioCommons system into more of a social media site where books can be recommended to users and the recommendations are influenced by many different factors. This type of design is definitely outside the scope of our project.

Polaris is another big player in the space. It is more focused towards large libraries with greater needs for interconnectivity and strongly managed resources. While Polaris was originally developed by a smaller team, it has since been acquired and integrated in part into the suite of systems by Ex Libris [7]. Polaris doesn't have many standout features, primarily relying on its large market share to keep influence in the library space. It is similar to our design in the way that it manages resources and is designed to communicate with other systems, but it is vastly larger in scope.

The dominant Integrated Library System is the suite of solutions offered by Ex Libris, with their flagship Library Management System being called Alma. Ex Libris' experience in the domain of library software grants them keen insight into the happenings in the industry. Their Alma platform is designed with the entire Ex Libris suite in mind, meaning that the software system is meant to be integrated with the rest of their offerings. The integration of Alma with many other Ex Libris systems means that the software is operating at a much larger scale than our planned project [8]. While the systems are similar in essence, the differences become apparent when the needs of both systems are considered. Our project is simply focused on implementing the base features of a library management system, like collecting and tagging resources and providing access to users.

Alma is designed to accomplish much more, including specific features that would be desired by an academic library, like interfacing with the entire world of academic publishing [8]. Our project does offer support and connections to journal platforms, however Alma goes beyond our scope in implementing the ability to discover academic resources.

Based on the comparisons between similar products in the space, our project maintains the basic functionality of the needs of a library but without implementing the more advanced functionality of a fully featured library management system like those examined. While it would be interesting to see how the space can evolve and how our project could grow into the niche filled by other products, this is beyond the scope of our project.

13. Conclusion

Overall, BookWorm is a project designed to be a university library web application with search and viewing features for articles, books, and databases, as well as student login, personal collection, book editing, and study room reservation features. At the start of BookWorm, it was decided that there would be notation features for articles and databases as well, however, after a conversation with the team during the design phase, it was removed from the plan. Along with the removal of this feature, the downloading and uploading of local book files was also removed due to complications with regulating the licensing state of the locally acquired literature.

Throughout the process of creating BookWorm, all team members kept their original task delegations and completed them within the required project scope timeframe. BookWorm's incremental software process model allows the team to pursue further implementation of removed features in the future without having to delay the first commit of the software. After thoroughly creating the functional and non-functional requirements of BookWorm mentioned above, all use cases and sequence diagrams were made by the delegated team members without issue. The repository architecture pattern chosen for BookWorm was decided based on the parameters of the project not allowing for a combination between two architecture patterns, which would've been MVC and repository if allowed.

The cost, effort, and pricing estimation of the software was researched meticulously, and ending up having the team apply function point analysis based on the sequence diagrams made, as well as having hardware and software cost estimates based on multiple read examples that led us to estimate a total cost of around $10,000 monthly for both hardware and software combined. The labor costs were also found using the average current salaries for software engineers.

Lastly, the test plan for the fine paying feature of BookWorm was created, and the Junit testing class followed suit in order to make sure the implementation worked correctly. As stated throughout, the BookWorm project team worked diligently in the beginning of the project phase in order to not have to deviate from any design choices made, thus completing the software known as BookWorm.

References

[1] Backblaze. (n.d.). *Cloud Storage Pricing*. Retrieved April 18, 2023, from https://www.backblaze.com/b2/cloud-storage-pricing.html

[2] Simonson, J., & Main, K. (2023, February 28). *Website hosting cost guide 2023*. Forbes. Retrieved April 18, 2023, from https://www.forbes.com/advisor/business/website-hosting-cost/

[3] GitHub (n.d.). *Pricing*. Retrieved April 18, 2023, from https://github.com/pricing

[4] Flexible (n.d.). *Software Developer Hourly Rate Guide*. Retrieved April 18, 2023, from https://flexiple.com/software/hourly-rate/

[5] Zippia. (n.d.). *Software Development Manager Salary*. Retrieved April 18, 2023, from https://www.zippia.com/software-development-manager-jobs/salary/

[6] “BiblioCommons Emerges.” Oder, Norman. Library Journal. July 19, 2008. https://web.archive.org/web/20160701194854/http://lj.libraryjournal.com/2008/07/industry-news/BiblioCommons-emerges-revolutionary-social-discovery-system-for-libraries/

[7] “Innovative Interfaces Acquires Polaris Library Systems.” Breeding, Marshall. American Library Association. https://www.ala.org/tools/article/ala-techsource/innovative-interfaces-acquires-polaris-library-systems

[8] “2022 Library Systems Report.” Breeding, Marshall. American Libraries Magazine. https://americanlibrariesmagazine.org/2022/05/02/2022-library-systems-report/