Bookshelf Software:

Project Deliverable #1 Report

Team 10 - The Librarians

Github Repository

Software Engineering

CS3354.002

Delegation of Tasks:

- Jayden Boomer:
    - Created the github repository
    - Added all team members to the repository
    - Pgs. 9-17 : Collaborated on sequence diagrams
    - Committed the report to the repository

- Manuel De La Cruz
    - Pg.8 : Create Use Case Diagram

- Andy Do
    - Ensure team project repository URL is included

- Samuel Green
    - Added README file to Github
    - Pgs. 9-17 : Collaborated on sequence diagrams
    - Pg. 19: Created Architectural Design after team decided on best pattern to use

- Sharon Kopuri
    - Pgs. 5-7 : List of software requirements including : Functional & Non functional requirements.

- Eli Mendez
    - Project scope pdf commit on github

- Cassidy Peña
    - Pg.3 : Addressing Feedback
    - Pg.4 : Software Process Model employed
    - Pg.18 : Class Diagram

You can make assumptions, even make up government/country-based rules, requirements to be able to provide one for each

We did not make any assumptions nor make up government / country-based rules or requirements in the project.

Addressing the feedback provided from Course Team Project Proposal:

-List what you are doing / planning to do regarding the feedback provided for your project proposal

The feedback received for our proposal is, “It is unclear what exactly chapter extraction and traversal is from the proposed implementation.”. Regarding this feedback we will make sure to make things more clear on their function in our software.

To make it clear, chapter extraction implies extracting the chapters in a pdf or text file and making a table of content for that file. On the other hand, chapter traversal is the ability to traverse through the chapters of a file using the table of contents generated by the chapter extraction. These two functionalities make the pdf or text file a more smooth and easy user experience.

Describe which software process model is employed in your project and why it was the choice. (Ch 2)

The software process model employed in our project is the Incremental Software model, one of the most common approaches. Meaning we are developing the software incrementally which is the best for our bookshelf software because our software is for students made by students. Our priority is for our software to be the best it can be for our customers / audience. By using the incremental process model we get to have feedback from the users and are able to change it for their liking, incrementally (Sommerville). Finally, this model provides us with a more rapid delivery and deployment of useful software to the customers which is something we are prioritizing.

In the end, the software process model we are employing is the Incremental model because of its easy feedback it provides and its rapid delivery to the users.

List of software requirements including:

a. Functional requirements

1. **Import & Library Sync:**
   The system shall scan a configurable "Books" folder and import **.pdf** and **.txt** files, extracting title/author from metadata or filename, within **10 seconds per 500 files**. Duplicate files (same hash) shall be ignored.
2. **Organize by Categories:**
   The system shall let a user **create/rename/delete** categories and **assign/unassign** books. A book can belong to **≥1 category**. Deleting a category shall **not** delete books.
3. **Text & Metadata Search:**
   The system shall return results for a text query across **book titles/authors** and **full-text** (for .txt and text-selectable .pdf) within **1 second for ≤1,000 books**. Results show **book, match count, first snippet**.
4. **Reader Navigation:**
   While reading, the user can **go to the next/previous page**, **jump to page N**, and **follow table-of-contents links** (if present). All actions complete in **<200 ms** after the file is open.
5. **Bookmarks:**
   The system shall let a user **add, list, rename, delete** bookmarks per book and **jump** to any bookmark. Bookmarks persist across app restarts.
6. **Annotations:**
   The system shall let a user **add, edit, view, and delete** annotations tied to a **page offset + optional text range**. Exporting a book's annotations to **JSON** is supported.
7. **Display Preferences:**
   The system shall support **day/night mode**, **font size adjustment** (txt), and **zoom** (pdf) and persist these **per book**. Changes apply in **<150 ms**.

b. Non-functional requirements (use all non-functional requirement types listed in Figure 4.3 (Ch 4).

1. **Product requirements**
   a. **Performance/Efficiency**
      - Open a **50 MB PDF** to first paint in **≤2.0 s** on target hardware (specify baseline machine).
      - Full-library re-index for **1,000 books / 3 GB** completes in **≤60 s**.
      - CPU stays **<40% avg** during idle library browsing; memory footprint **<500 MB** with 50 books indexed.

Dependability/Reliability

- Crash-free session rate **≥99.5%** over 1,000 sessions.
- Corruption-safe import: if import is interrupted, library state remains consistent (no orphaned records).

Security

- All local data (library DB, annotations, settings) encrypted at rest using **AES-256-GCM** with OS-protected key store.
- No network access is performed without explicit user opt-in (telemetry default **off**).

Organizational requirements

Usability

- First-time task success (import a book, create a category, add a bookmark) for novice users **≥90%** without help in **≤3 minutes** (usability test with n=10).
- Accessibility: all interactive controls keyboard-navigable; contrast ratio **≥4.5:1**; screen-reader labels present.

Environmental (Operating Constraints)

- Supported OS: **Windows 10/11** and **macOS 13+**. App must function offline.

Operational

- Automatic, daily **local backup** of library DB and annotations; restore completes in **≤30 s**.

Development

- Codebase in **TypeScript + Electron** (or your pick—state it) with **≥80% unit test coverage** on core modules; CI runs tests and lints on every push; main branch protected.

External requirements

Regulatory/Legislative

- Honor OS-level data-deletion requests; provide **"Delete all data"** that wipes local store within **≤5 s**.
- Respect copyright: app **does not** bypass DRM or alter protected PDFs.

Ethical

- No dark patterns: import locations are local by default; telemetry is opt-in and clearly disclosed.

Additional product constraints (call out explicitly)

Space

- App install size **≤250 MB**; per-book index overhead **≤1% of file size** (cap 5 MB/book).

Safety/Security (Operational Safety)

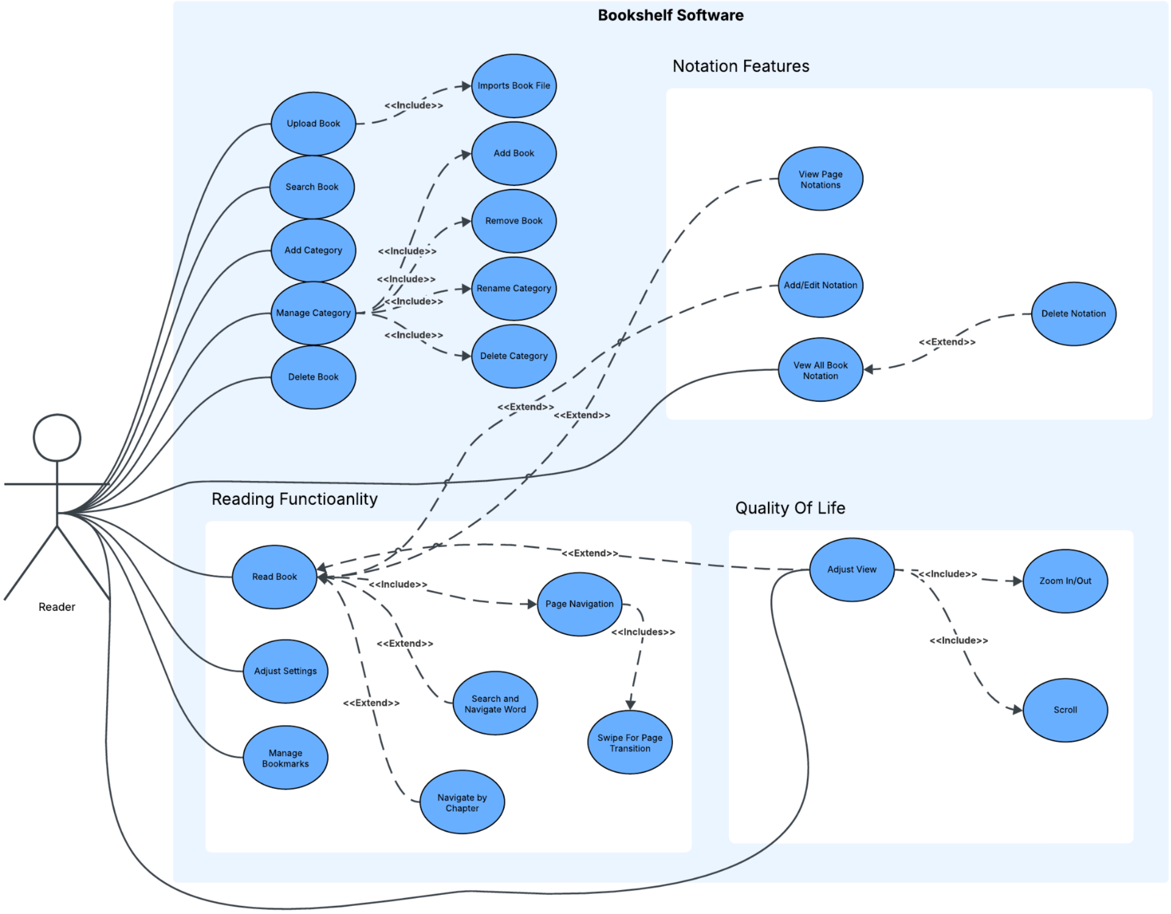
- Never execute embedded file attachments or links without explicit user confirmation; sanitize all file paths to prevent path traversal.

Portability

- Library (DB + annotations) export/import to a single **.zip** to move between machines; round-trip fidelity **100%** in QA tests.

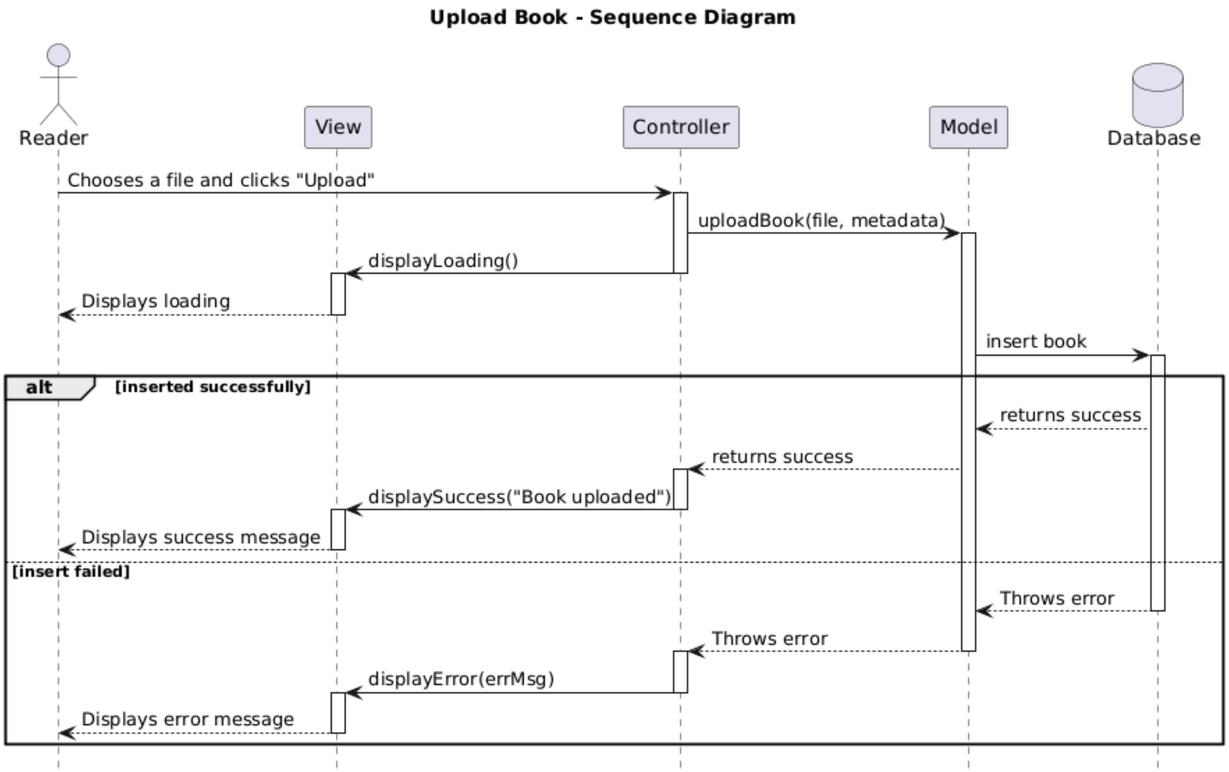
Use case diagram – **Provide a use case diagram (Figure 5.5) for your project**

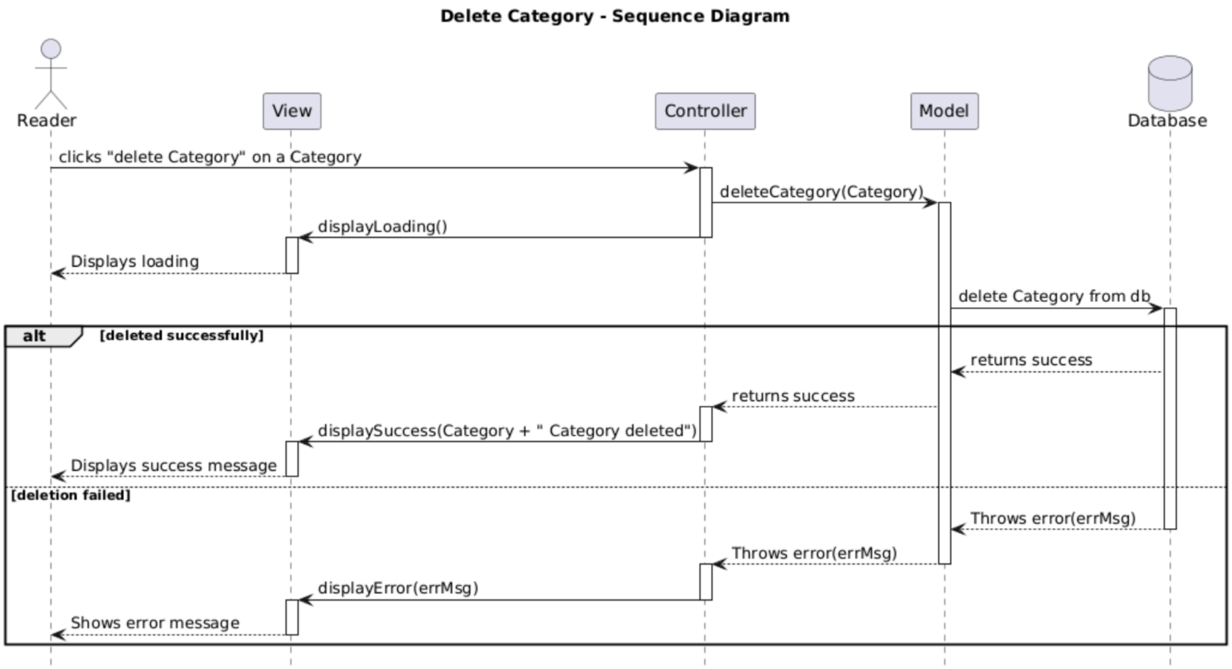
a. Please note that there should be more than one use case depending on the complexity of your project. (Ch 5 and Ch 7)



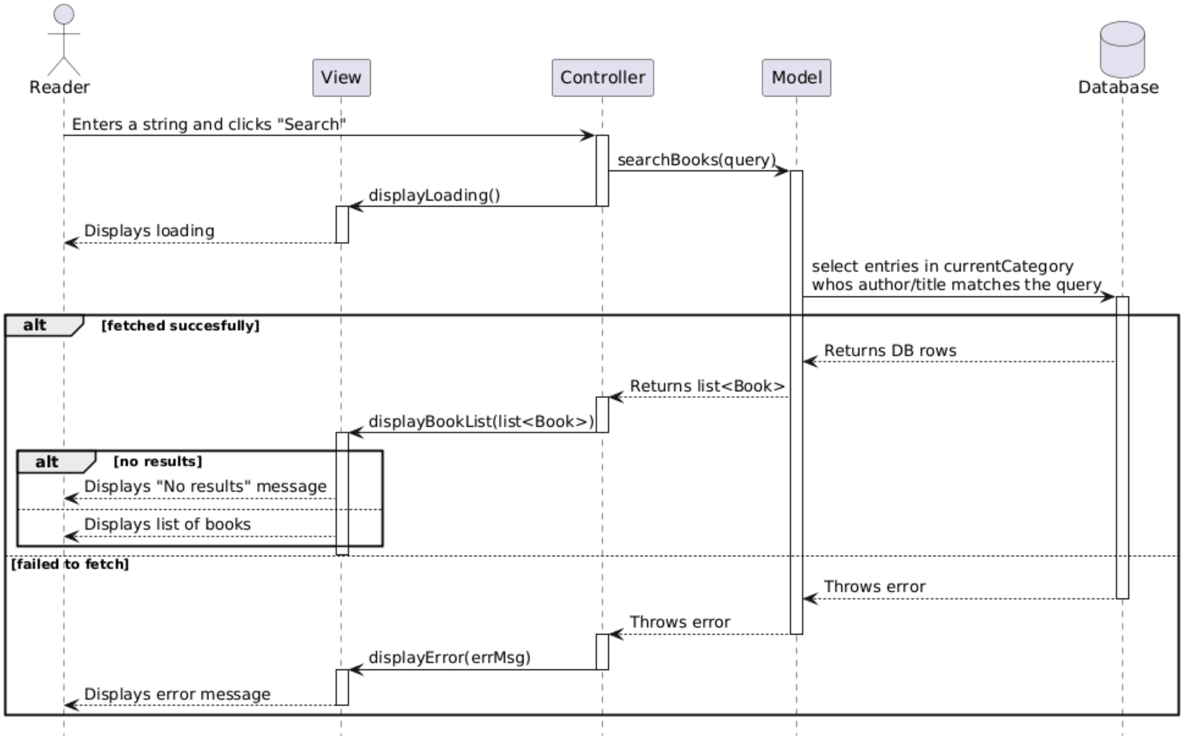
Sequence diagram – Provide sequence diagrams (Figure 5.6 and Figure 5.7) for each use case of your project.

a. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)

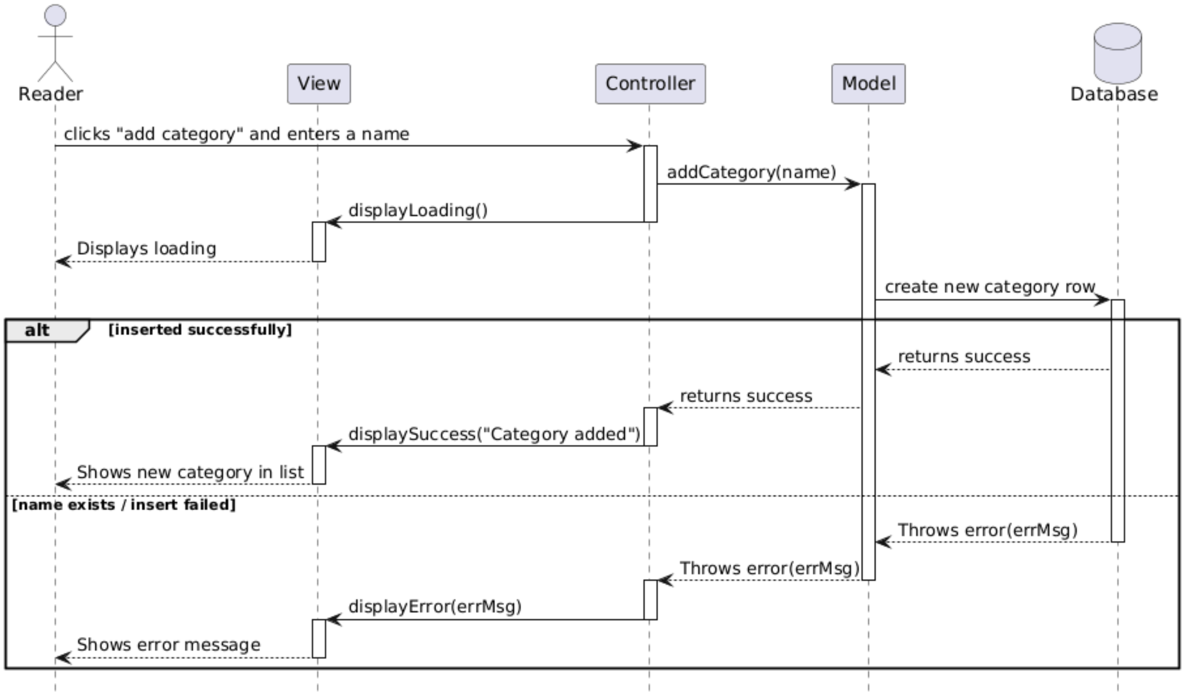




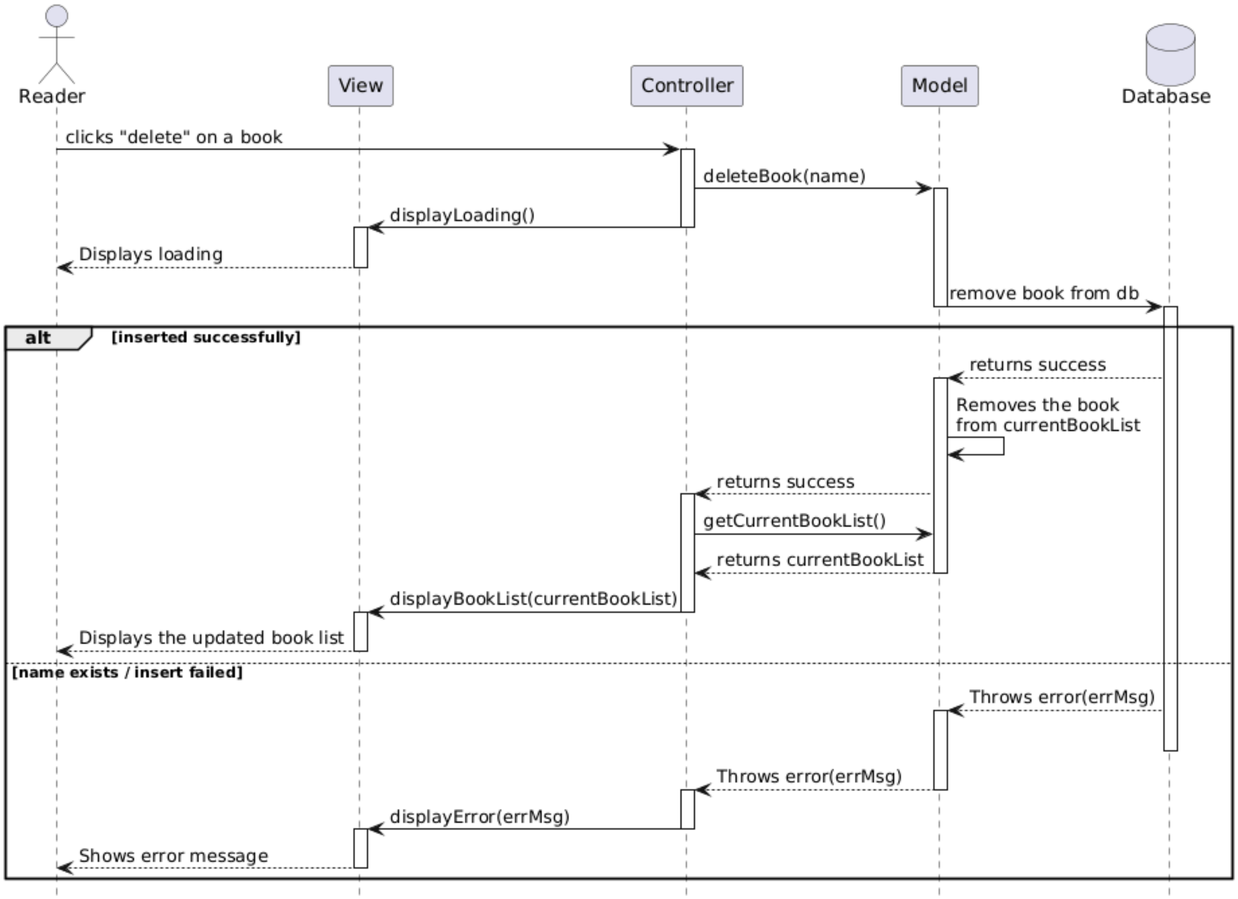
Search Books - Sequence Diagram



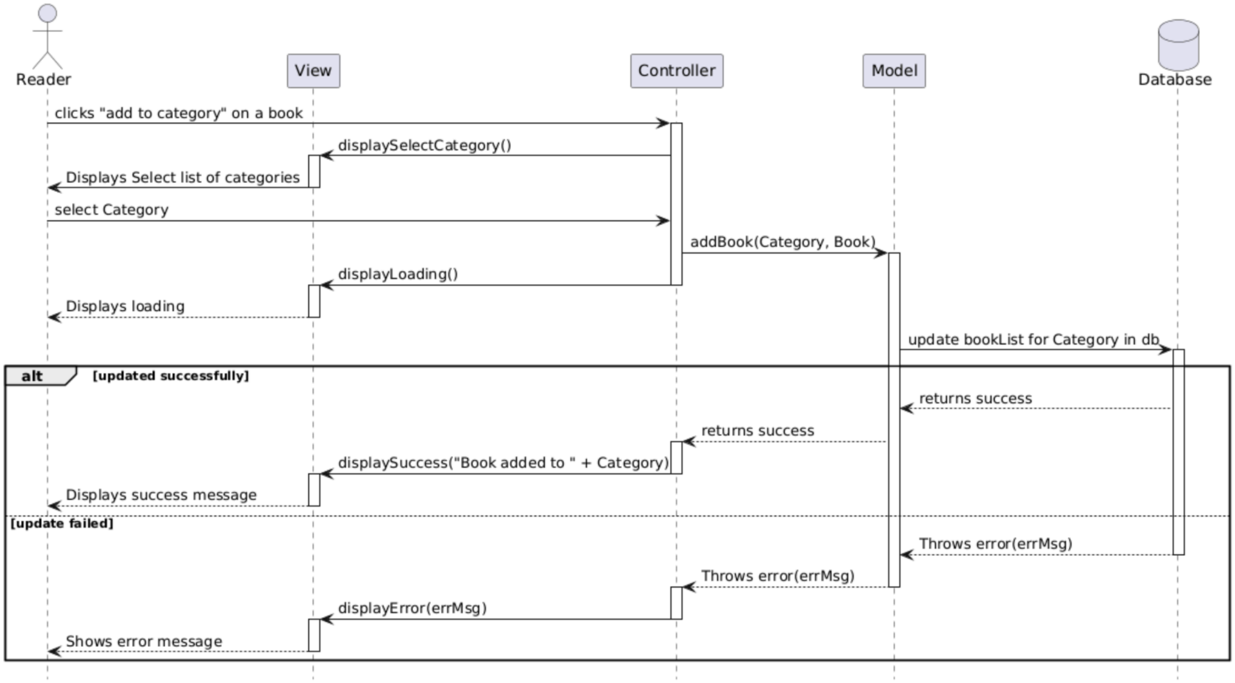
Add Category - Sequence Diagram



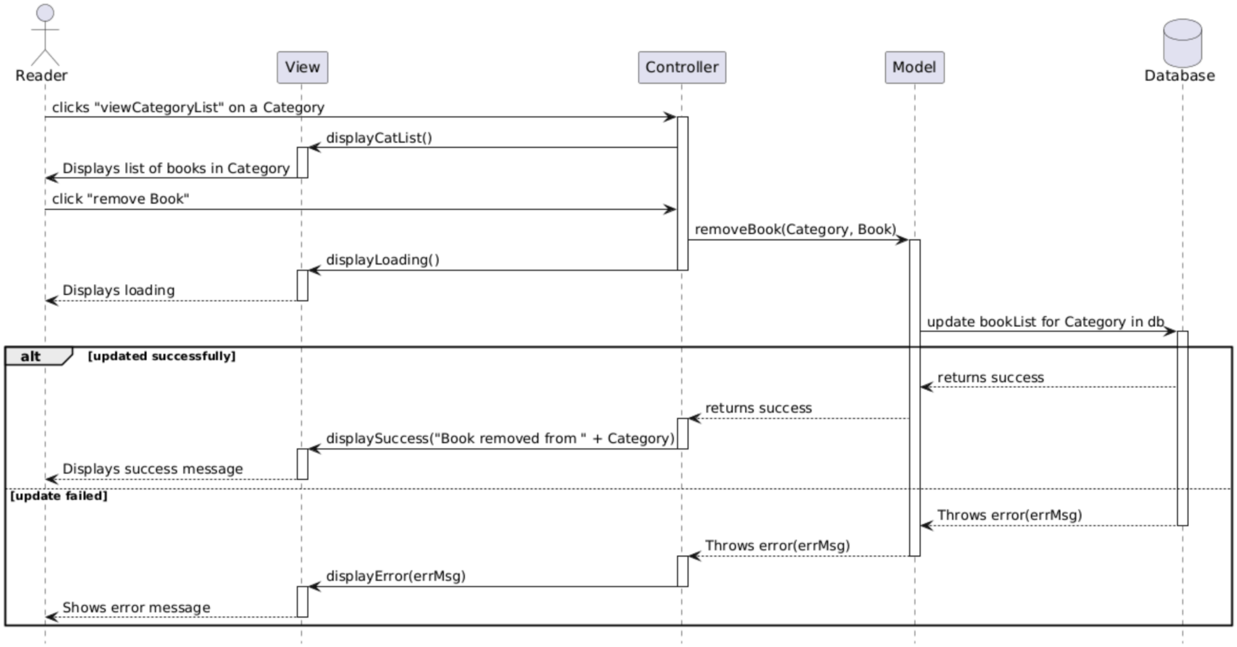
Delete Book - Sequence Diagram



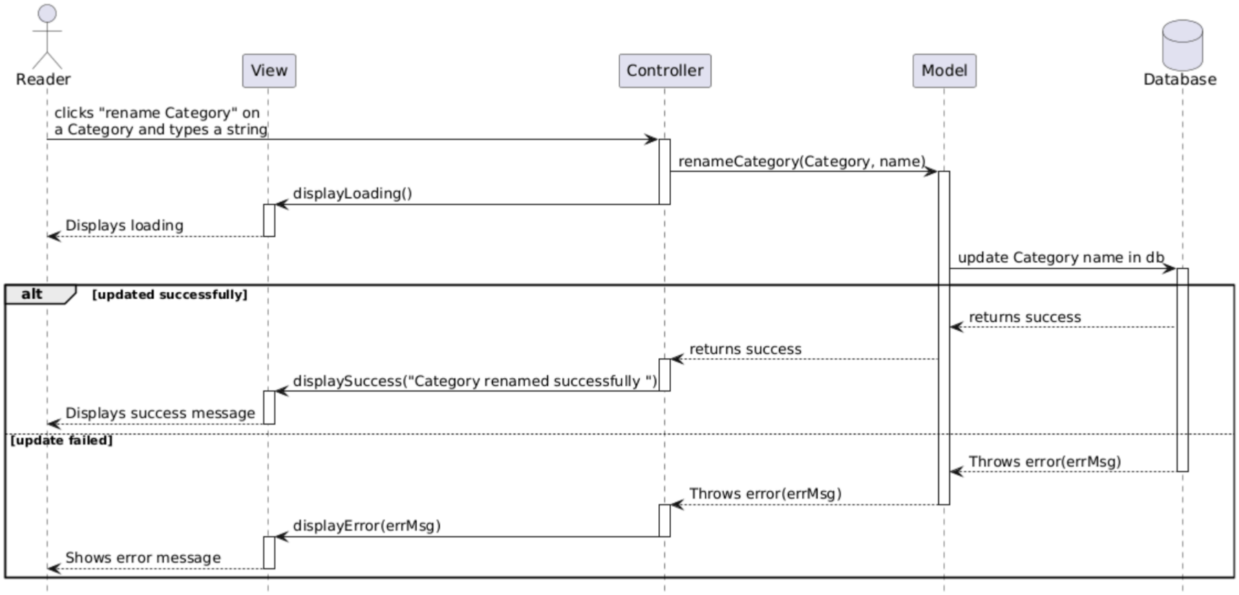
Add Book to Category - Sequence Diagram



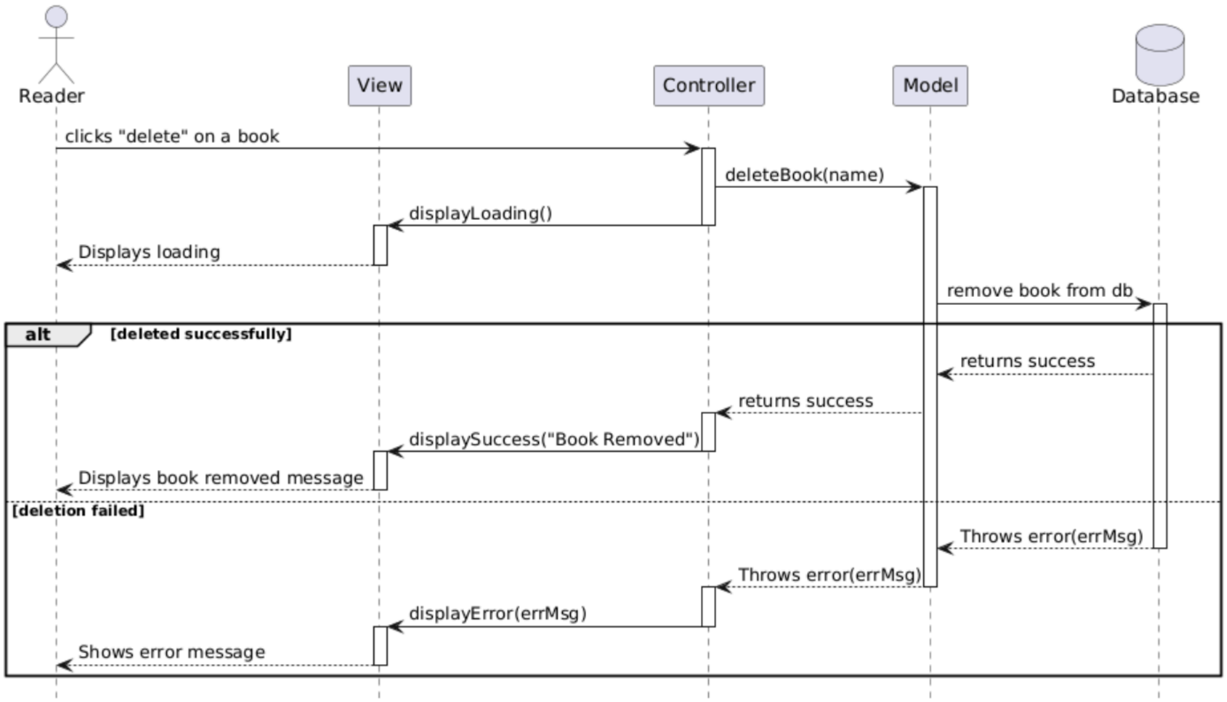
Remove Book from Category - Sequence Diagram



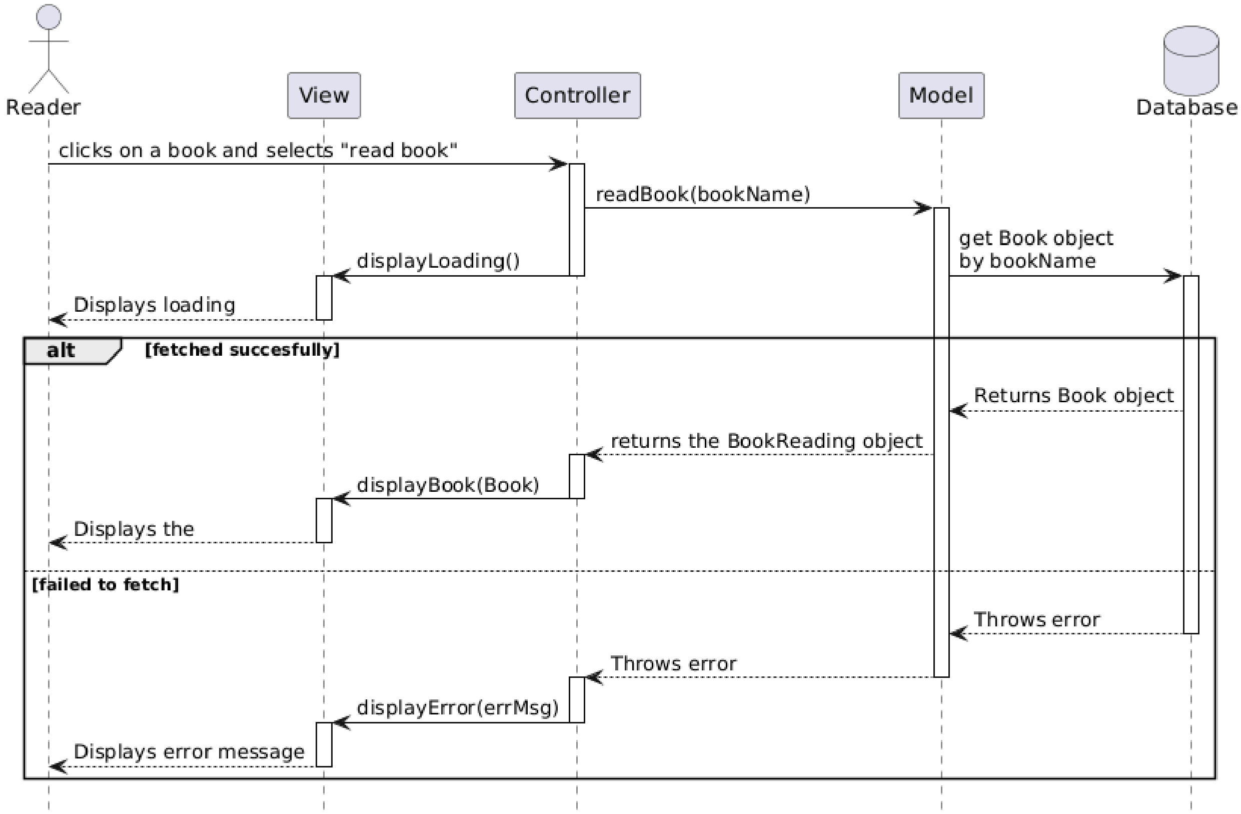
Rename Category - Sequence Diagram



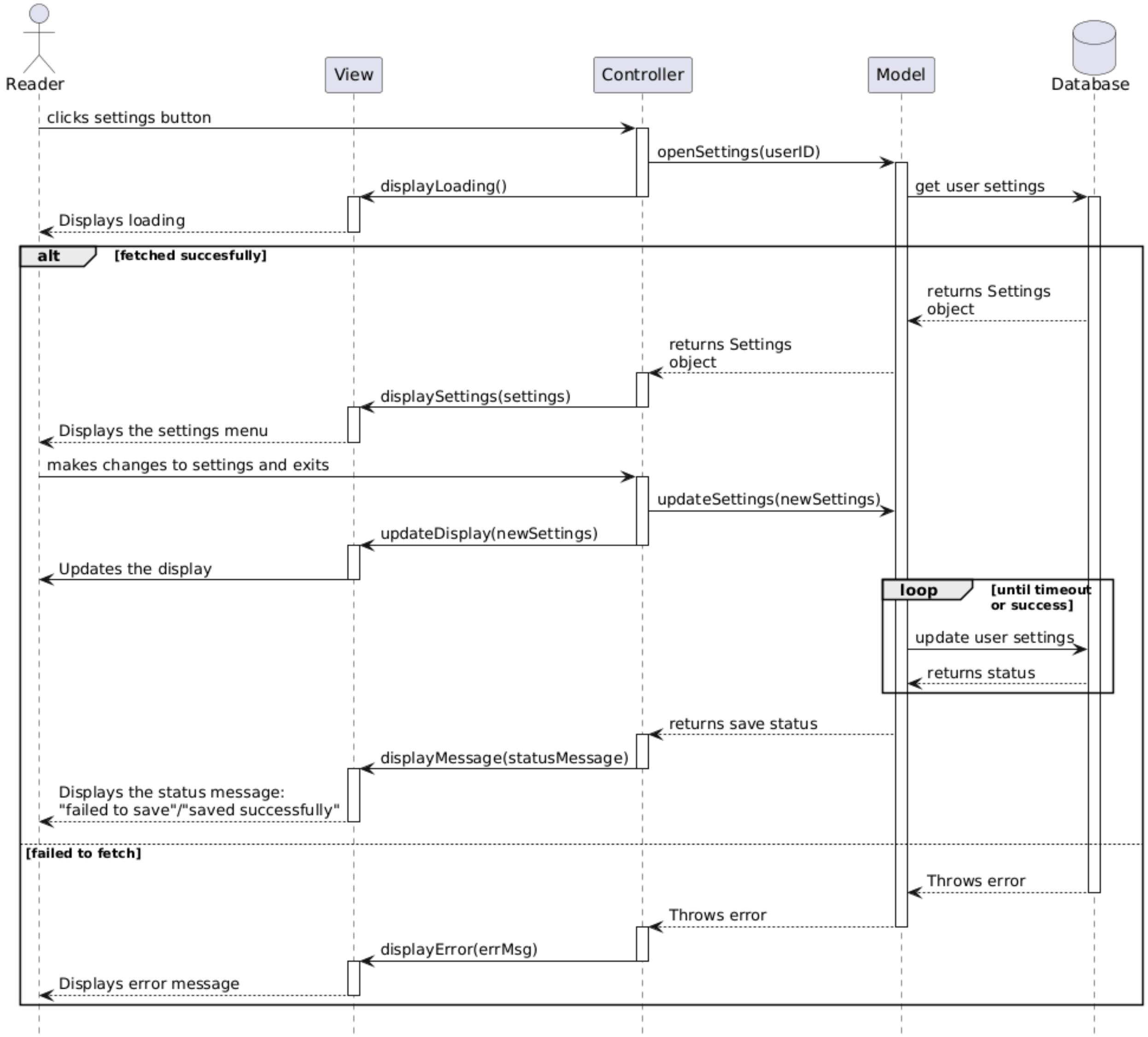
Delete Book - Sequence Diagram



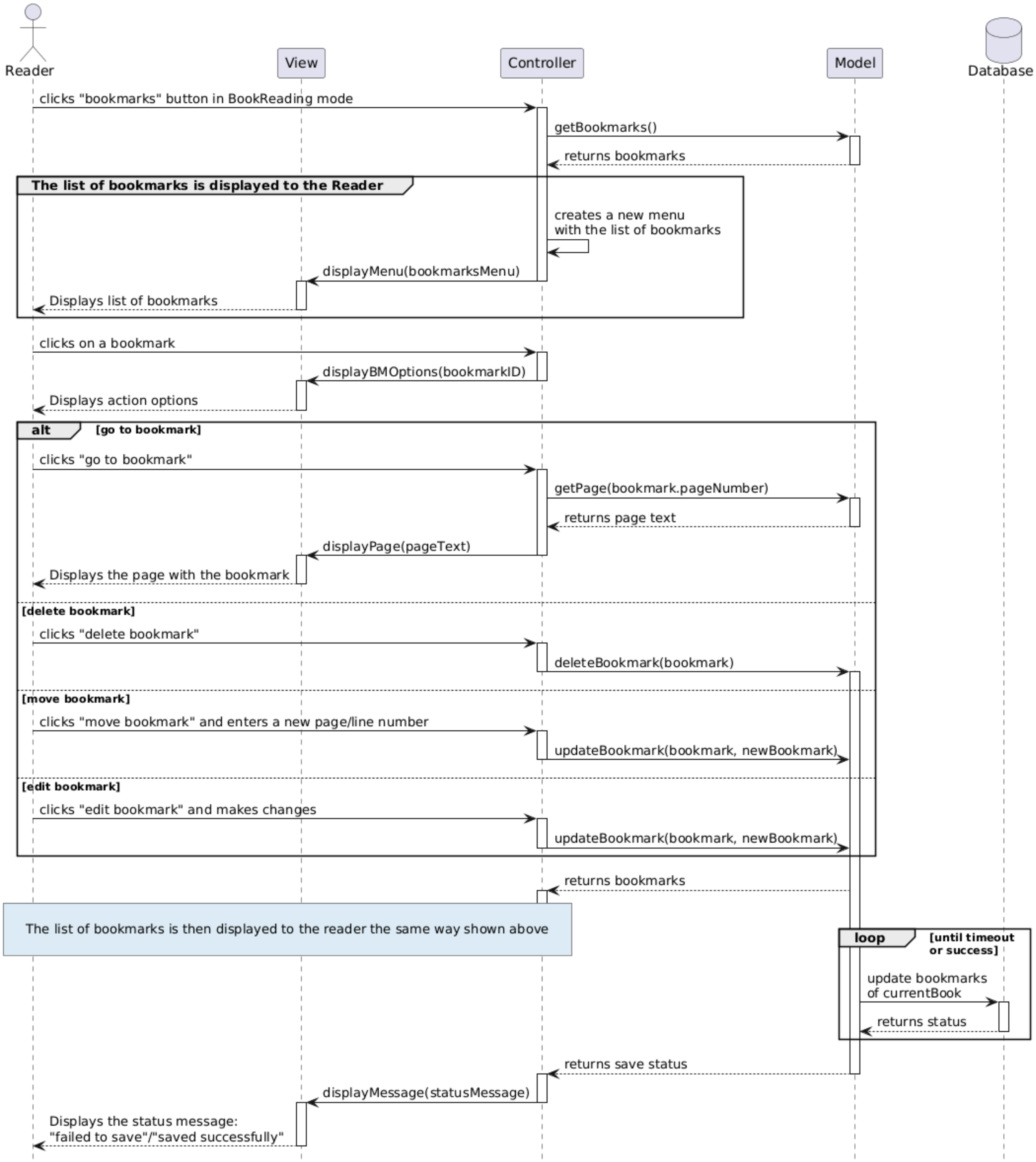
Read Book - Sequence Diagram



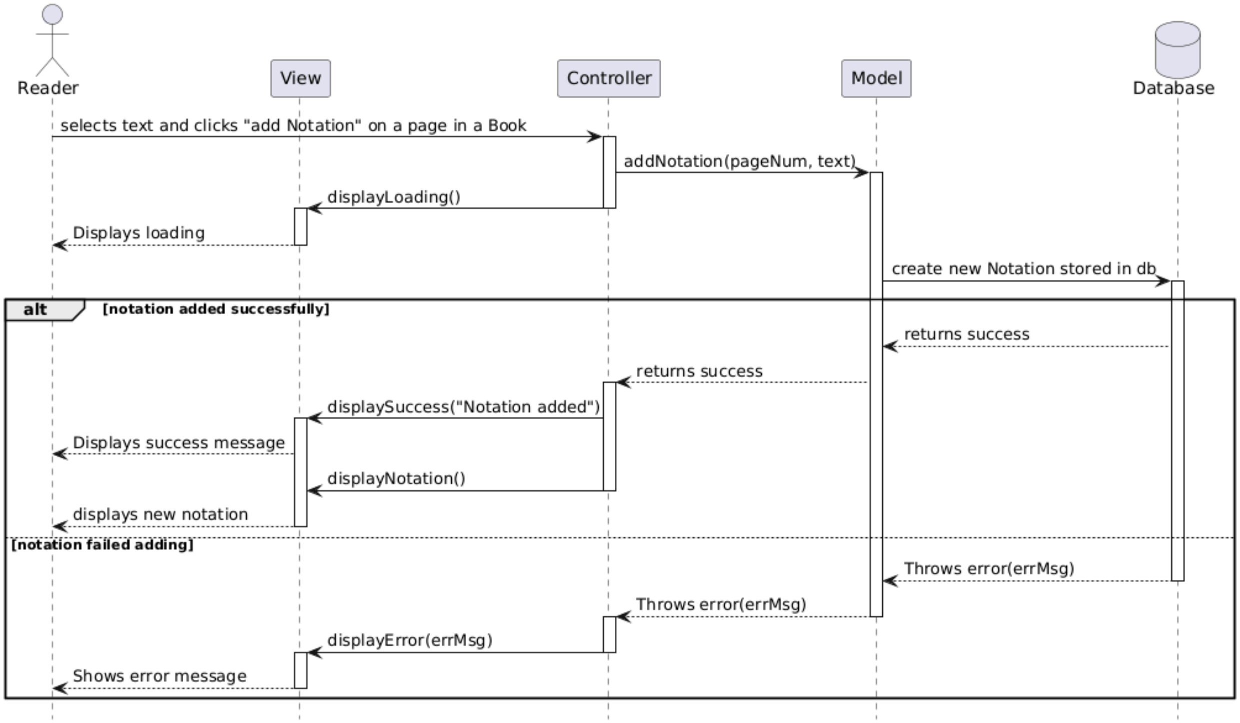
Adjust Settings - Sequence Diagram



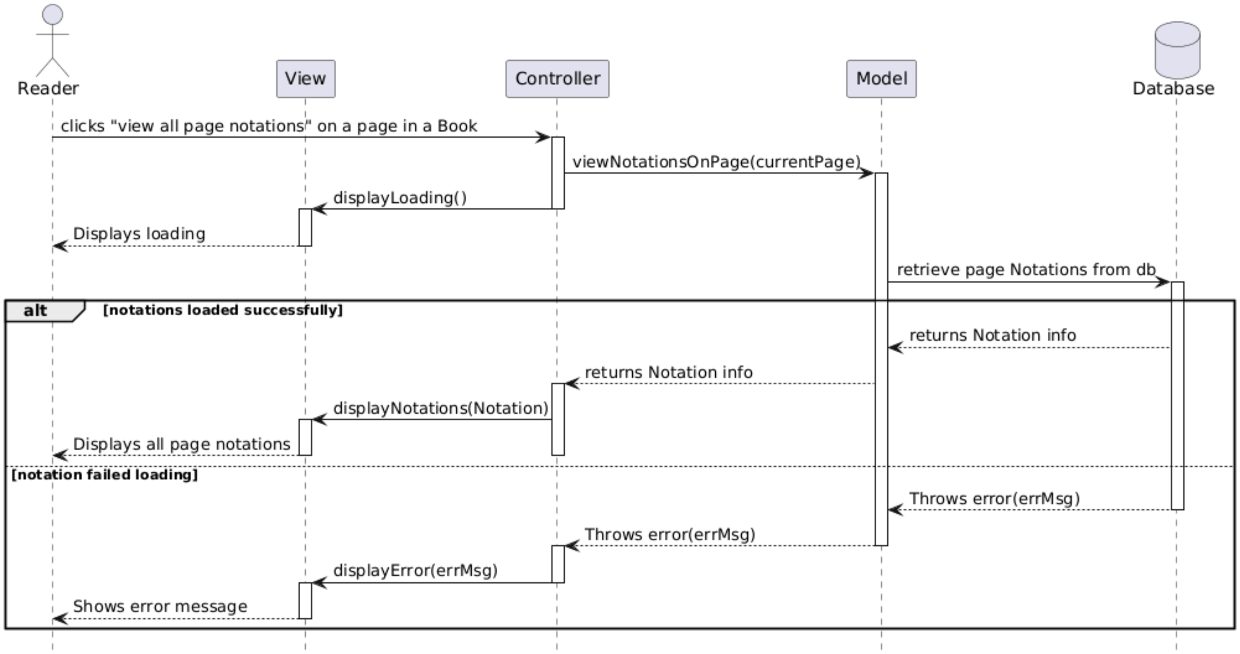
Bookmarks - Sequence Diagram


Reader
View
Controller
Model
Database
clicks "bookmarks" button in BookReading mode
getBookmarks()
returns bookmarks
The list of bookmarks is displayed to the Reader
creates a new menu
with the list of bookmarks
displayMenu(bookmarksMenu)
Displays list of bookmarks
clicks on a bookmark
displayBMOptions(bookmarkID)
Displays action options
alt
[go to bookmark]
clicks "go to bookmark"
getPage(bookmark.pageNumber)
returns page text
displayPage(pageText)
Displays the page with the bookmark
[delete bookmark]
clicks "delete bookmark"
deleteBookmark(bookmark)
[move bookmark]
clicks "move bookmark" and enters a new page/line number
updateBookmark(bookmark, newBookmark)
[edit bookmark]
clicks "edit bookmark" and makes changes
updateBookmark(bookmark, newBookmark)
returns bookmarks
The list of bookmarks is then displayed to the reader the same way shown above
loop
[until timeout or success]
update bookmarks
of currentBook
returns status
returns save status
displayMessage(statusMessage)
Displays the status message:
"failed to save"/"saved successfully"

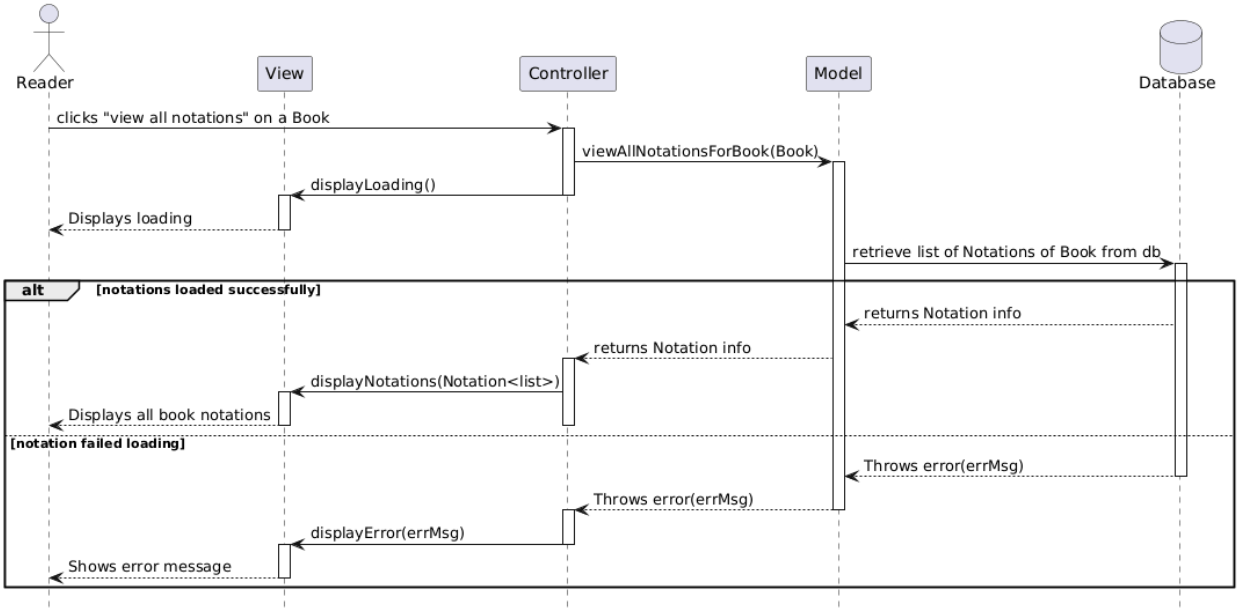
Add Notation - Sequence Diagram



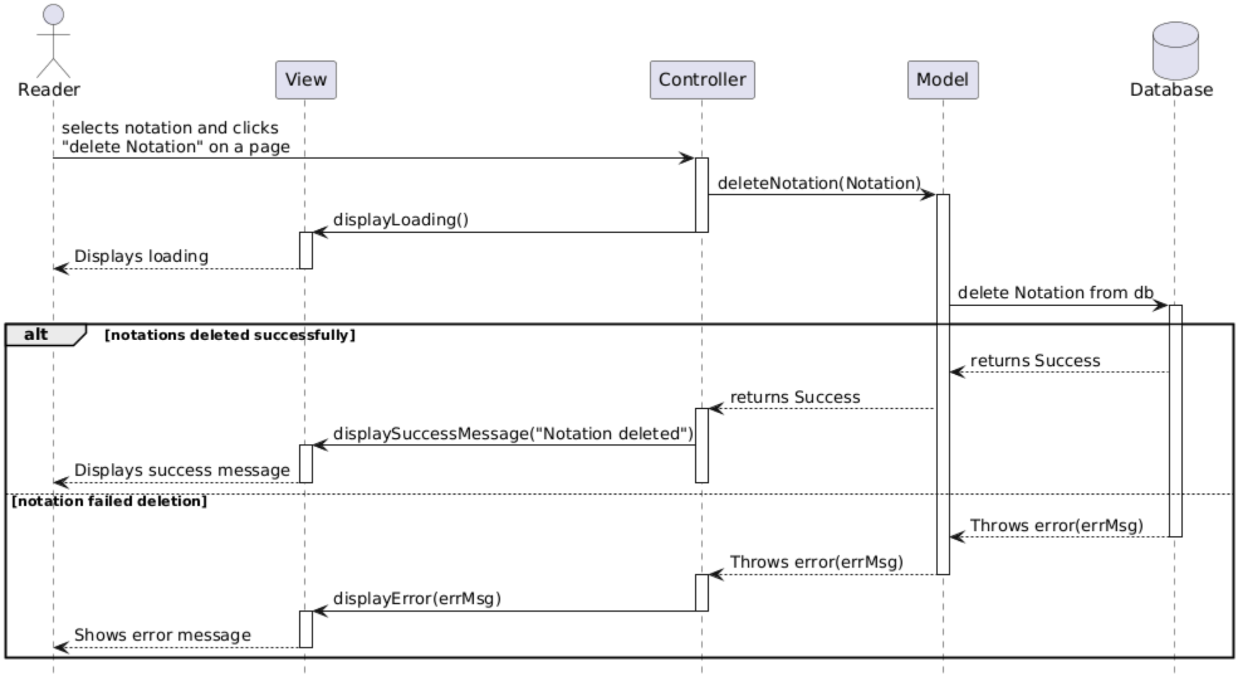
View Page Notations - Sequence Diagram


Reader
View
Controller
Model
Database
clicks "view all page notations" on a page in a Book
viewNotationsOnPage(currentPage)
displayLoading()
Displays loading
retrieve page Notations from db
alt
[notations loaded successfully]
returns Notation info
returns Notation info
displayNotations(Notation)
Displays all page notations
[notation failed loading]
Throws error(errMsg)
Throws error(errMsg)
displayError(errMsg)
Shows error message

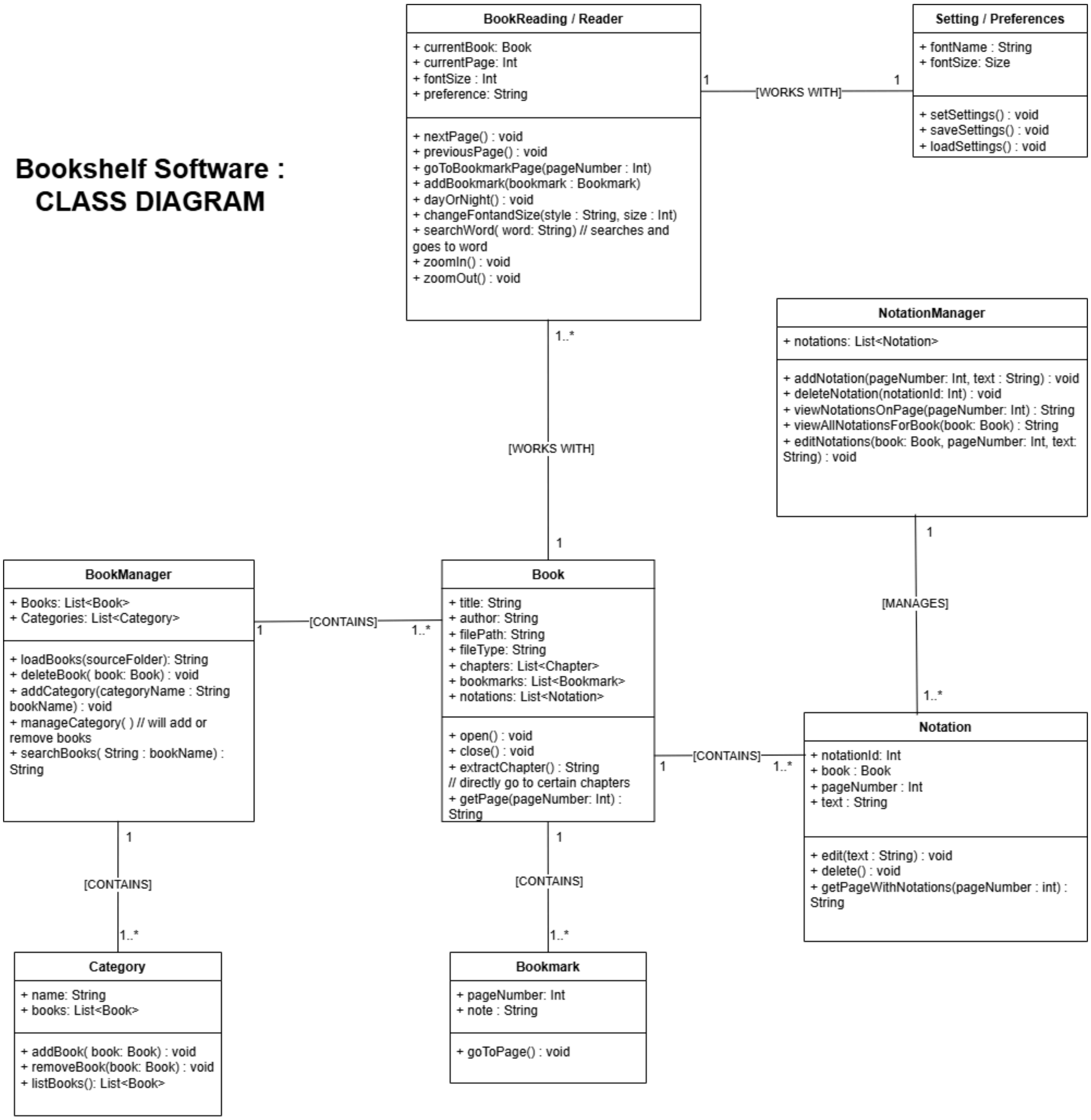
View all Book Notations - Sequence Diagram


Reader
View
Controller
Model
Database
clicks "view all notations" on a Book
viewAllNotationsForBook(Book)
displayLoading()
Displays loading
retrieve list of Notations of Book from db
alt
[notations loaded successfully]
returns Notation info
returns Notation info
displayNotations(Notation<list>)
Displays all book notations
[notation failed loading]
Throws error(errMsg)
Throws error(errMsg)
displayError(errMsg)
Shows error message

Delete Notation - Sequence Diagram



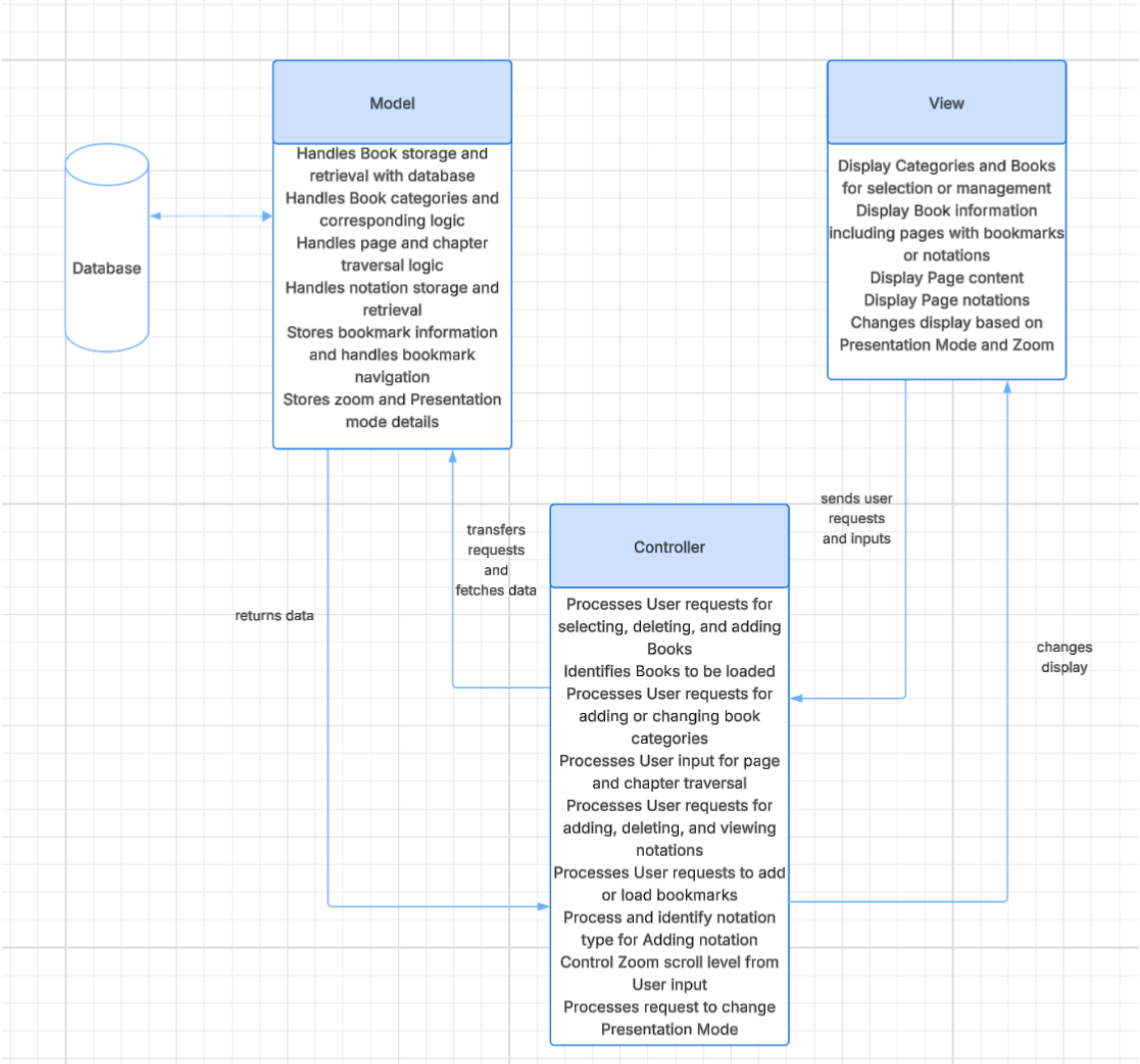
Class diagram – Provide a class diagram of your project.


Bookshelf Software :
CLASS DIAGRAM
BookReading / Reader
+ currentBook: Book
+ currentPage: Int
+ fontSize : Int
+ preference: String
+ nextPage() : void
+ previousPage() : void
+ goToBookmarkPage(pageNumber : Int)
+ addBookmark(bookmark : Bookmark)
+ dayOrNight() : void
+ changeFontandSize(style : String, size : Int)
+ searchWord(word: String) // searches and goes to word
+ zoomIn() : void
+ zoomOut() : void
1
[WORKS WITH]
1
Setting / Preferences
+ fontName : String
+ fontSize: Size
+ setSettings() : void
+ saveSettings() : void
+ loadSettings() : void
1..*
[WORKS WITH]
1
NotationManager
+ notations: List<Notation>
+ addNotation(pageNumber: Int, text : String) : void
+ deleteNotation(notationId: Int) : void
+ viewNotationsOnPage(pageNumber: Int) : String
+ viewAllNotationsForBook(book: Book) : String
+ editNotations(book: Book, pageNumber: Int, text: String) : void
1
[MANAGES]
1..*
BookManager
+ Books: List<Book>
+ Categories: List<Category>
+ loadBooks(sourceFolder): String
+ deleteBook(book: Book) : void
+ addCategory(categoryName : String bookName) : void
+ manageCategory( ) // will add or remove books
+ searchBooks(String : bookName) : String
1
[CONTAINS]
1..*
Book
+ title: String
+ author: String
+ filePath: String
+ fileType: String
+ chapters: List<Chapter>
+ bookmarks: List<Bookmark>
+ notations: List<Notation>
+ open() : void
+ close() : void
+ extractChapter() : String
// directly go to certain chapters
+ getPage(pageNumber: Int) : String
1
[CONTAINS]
1..*
Notation
+ notationId: Int
+ book : Book
+ pageNumber : Int
+ text : String
+ edit(text : String) : void
+ delete() : void
+ getPageWithNotations(pageNumber : int) : String
1
[CONTAINS]
1..*
Category
+ name: String
+ books: List<Book>
+ addBook(book: Book) : void
+ removeBook(book: Book) : void
+ listBooks(): List<Book>
1
[CONTAINS]
1..*
Bookmark
+ pageNumber: Int
+ note : String
+ goToPage() : void

Architectural design – Provide an architectural design of your project.

a. Based on the characteristics of your project, choose, and apply only one appropriate architectural pattern from the following list: (Ch 6)

i. Model-View-Controller (MVC) pattern (Figure 6.6)
ii. Layered architecture pattern (Figure 6.9)
iii. Repository architecture pattern (Figure 6.11)
iv. Client-server architecture pattern (Figure 6.13)
v. Pipe and filter architecture pattern (Figure 6.15)



Model

Handles Book storage and retrieval with database
Handles Book categories and corresponding logic
Handles page and chapter traversal logic
Handles notation storage and retrieval
Stores bookmark information and handles bookmark navigation
Stores zoom and Presentation mode details

View

Display Categories and Books for selection or management
Display Book information including pages with bookmarks or notations
Display Page content
Display Page notations
Changes display based on Presentation Mode and Zoom

Controller

Processes User requests for selecting, deleting, and adding Books
Identifies Books to be loaded
Processes User requests for adding or changing book categories
Processes User input for page and chapter traversal
Processes User requests for adding, deleting, and viewing notations
Processes User requests to add or load bookmarks
Process and identify notation type for Adding notation
Control Zoom scroll level from User input
Processes request to change Presentation Mode

Database
returns data
transfers requests and fetches data
sends user requests and inputs
changes display

Work Cited (References)

GeeksforGeeks. “MVC Design Pattern.” *GeeksforGeeks*, 18 Aug. 2017, www.geeksforgeeks.org/system-design/mvc-design-pattern/.

GeeksforGeeks. “MVC Architecture System Design.” *GeeksforGeeks*, July 2024, www.geeksforgeeks.org/system-design/mvc-architecture-system-design/.

GeeksforGeeks. “Class Diagram | Unified Modeling Language (UML).” *GeeksforGeeks*, 30 Aug. 2018, www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/.

“Examples of class diagrams for softwares” prompt. *ChatGPT,* 23 Mar.version, OpenAI, 4 Oct. 2023, chat.openai.com/chat

Sommerville, Ian. *Software Engineering*. 10th ed., Boston, Pearson Education Limited, 2016.