Bookshelf Software:

Project Deliverable #1 Report

Team 10 - The Librarians

Github Repository

Software Engineering

CS3354.002

Delegation of Tasks:

1. Jayden Boomer:
    - Created the github repository
    - Added all team members to the repository
    - Pgs. 9-16 : Collaborated on sequence diagrams
    - Committed the report to the repository

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

3. Andy Do
    - Ensure team project repository URL is included

4. Samuel Green
    - Added README file to Github
    - Pgs. 9-16 : Collaborated on sequence diagrams
    - Pg. 18: Created Architectural Design after team decided on best pattern to use

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

6. Eli Mendez
    - Project scope pdf commit on github

7. Cassidy Peña
    - Pg.3 : Addressing Feedback
    - Pg.4 : Software Process Model employed
    - Pg.17 : 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.

Functional & Non-Functional Requirements - Assumptions made:

- Single-user, offline desktop app (Windows/macOS), local DB; no cloud
- Scope = **PDF + TXT** only; no OCR/DRM bypass; annotations/bookmarks stored in our Database, not the files.
- One watched the "Books" folder; many-to-many categories that don't delete books.
- Performance targets assume ~**1,000 books/3 GB** on mid-range SSD machines; times are warm-state.
- Core features: import, organize, search, read/nav, bookmarks, annotations, display prefs
- Security: encryption at rest via OS keystore
- Reliability: crash-safe imports; daily local backups.
- Out of scope: cloud sync, multi-device

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:**
   Within **10 seconds per 500 files** the system will scan the "Books" folder and import **.pdf** and **.txt** files,this will bring out the title/author from filename. Duplicate files will be ignored.
2. **Organize by Categories:**
   The system will let the user **create/rename/delete** categories and **assign/unassign** books. A particular book can belong to **≥1 category**. When you decide to delete a category this will **not** delete books.
3. **Text & Metadata Search:**
   Within **1 second for ≤1,000 books** the system shall return results for a text query across **book titles/authors** and **full-text** (for .txt and text-selectable .pdf). The Results would be shown based on the **book, match count, and the first snippet**.
4. **Reader Navigation:**
   When reading, the user can **go to the next/previous page**, **jump to page N**, and **follow the table of contents links**. Note: All the actions are designed to be completed in **<200 ms** after the file is open.
5. **Bookmarks:**
   The system lets the user **add, list, rename, and delete** bookmarks per book and also lets you **jump** to any bookmark. Note: Bookmarks persist across app restarts.
6. **Annotations:**
   The system will let a user **add, edit, view, and delete** annotations tied to a **page offset + optional text range**. Note: Exporting a book's annotations to **JSON** is supported.

7. **Display Preferences:**
   The system will 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

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.
   b. **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).
   c. **Security Requirements**
      - 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**).

2. **Organizational requirements**
   a. **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.
   b. **Environmental (Operating Constraints)**
      - Supported OS: **Windows 10/11** and **macOS 13+**. App must function offline.
   c. **Operational Requirements**
      - Automatic, daily **local backup** of library DB and annotations; restore completes in **≤30 s**.
   d. **Development Requirements**
      - 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.

3. External requirements

a. Regulatory Requirements

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

b. Ethical Requirements

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

c. Additional product constraints (call out explicitly) Space

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

d. Safety/Security (Operational Safety)

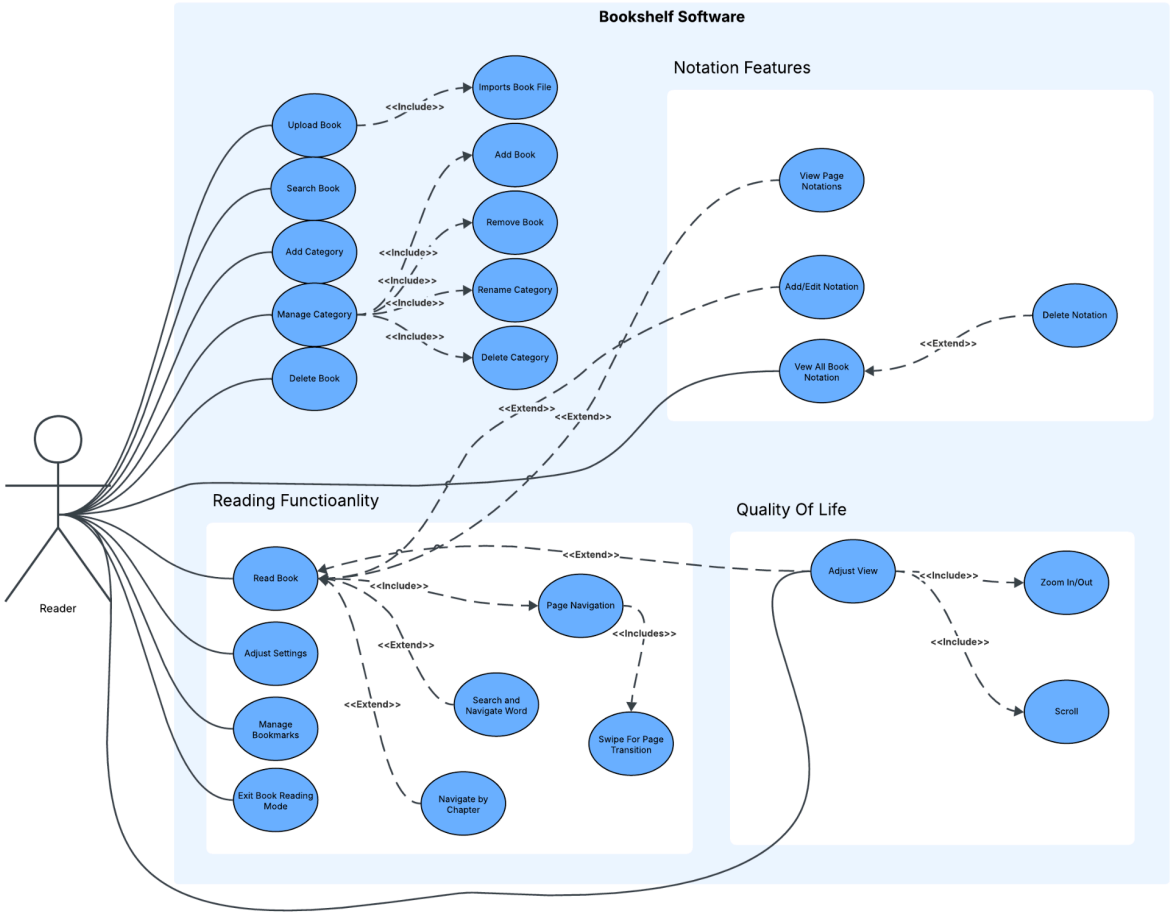
- Never execute embedded file attachments or links without explicit user confirmation; sanitize all file paths to prevent path traversal.

e. Portability

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

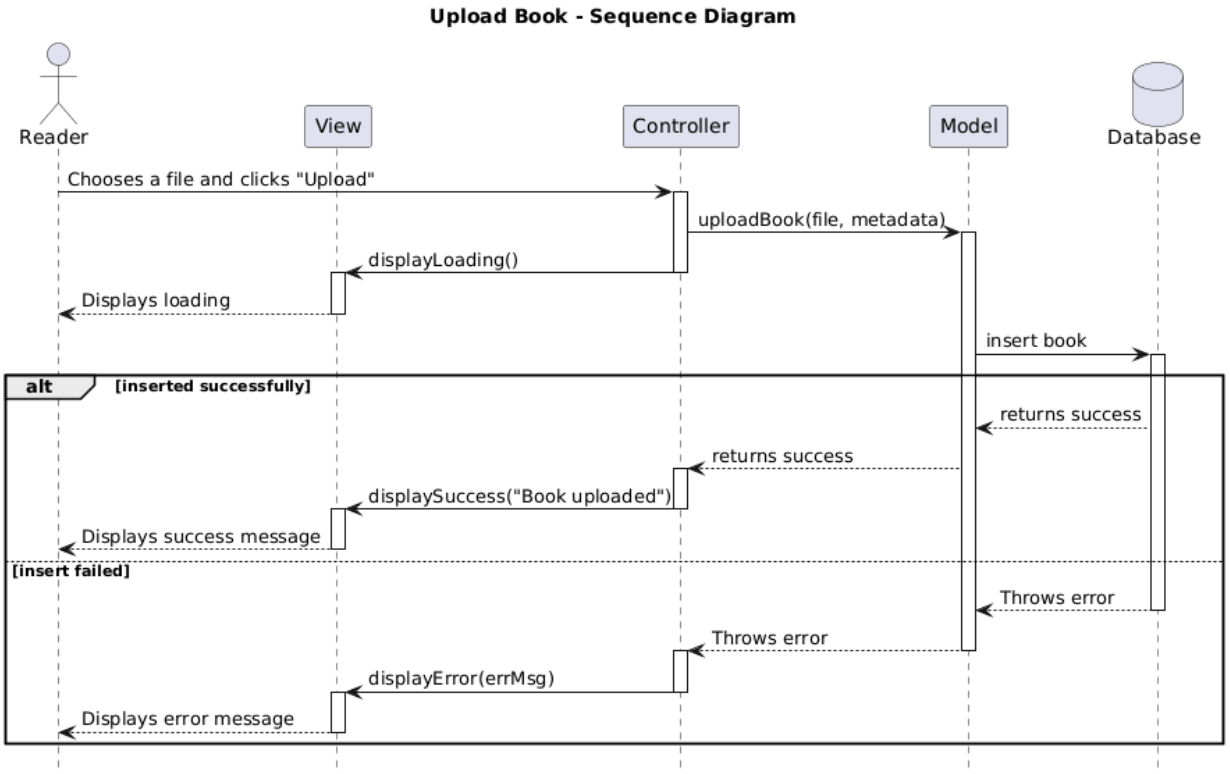
Use case diagram – Provide a use case diagram (Figure 5.5) for your project

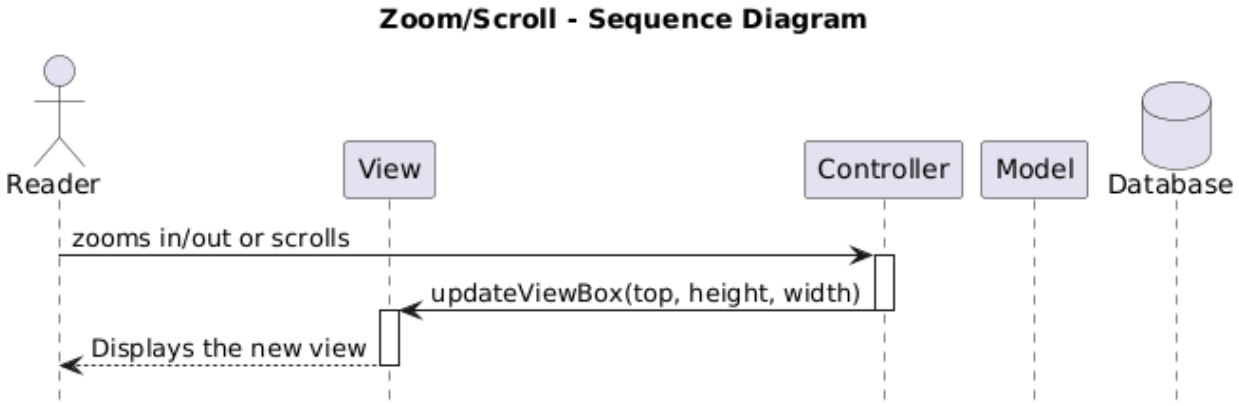
a. Please note that there should be more than one use case depending on the complexity of your project. (Ch 5 and Ch 7)



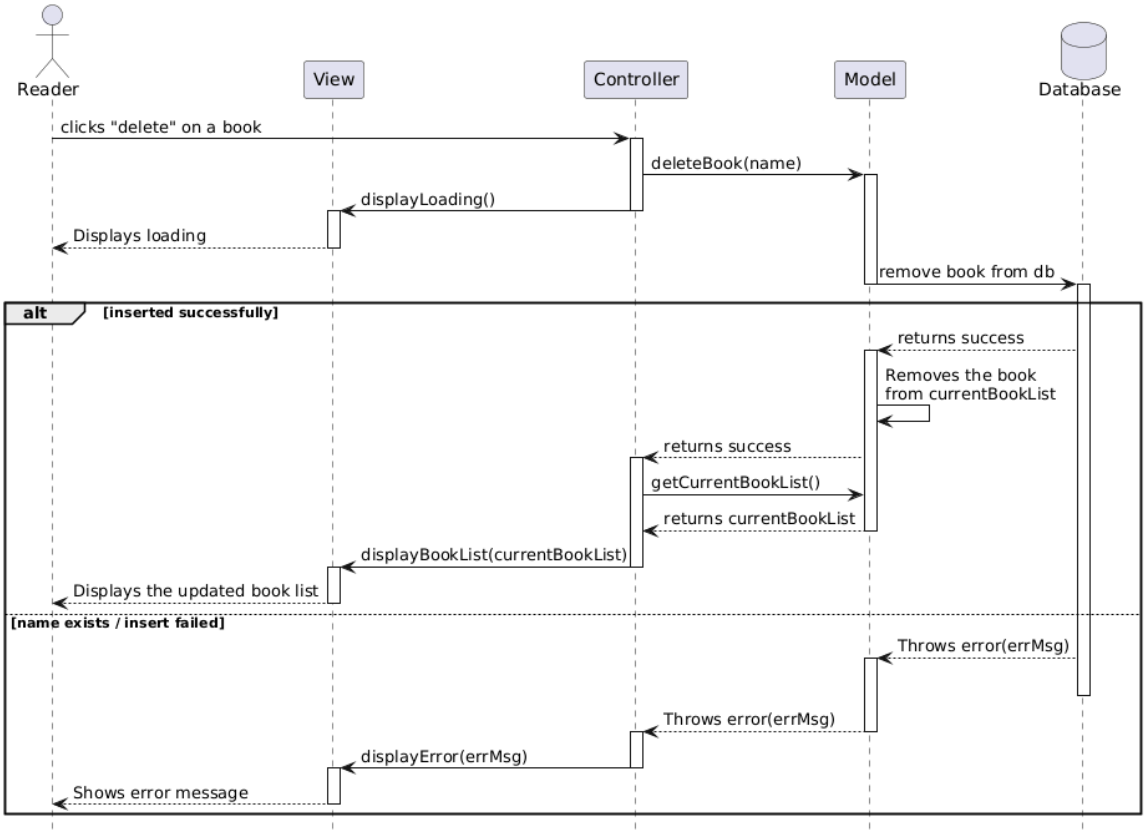
Sequence diagram – Provide sequence diagrams (Figure 5.6 and Figure 5.7) for each use case of your project.

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

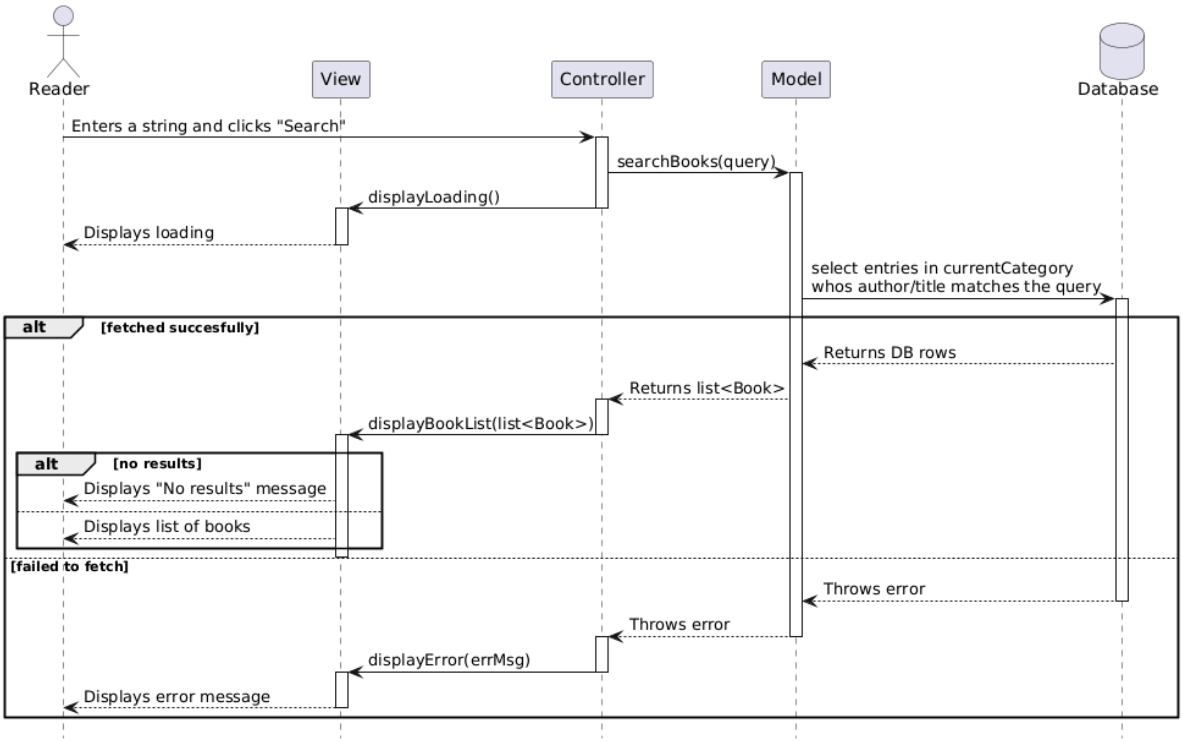




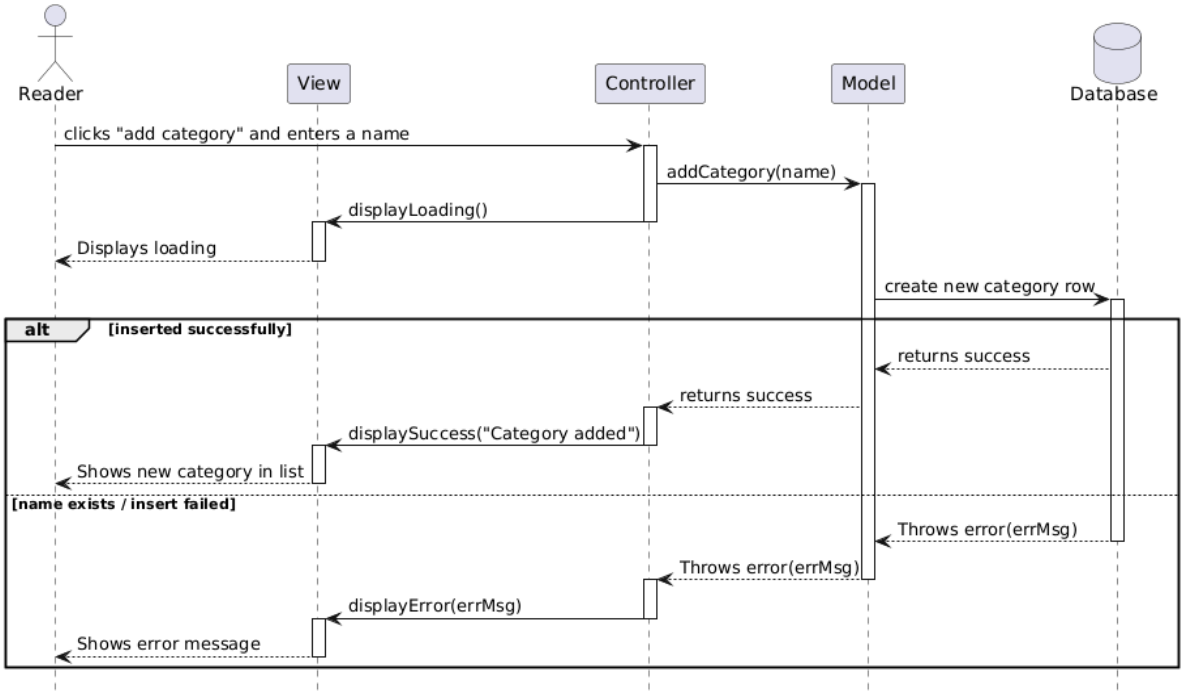
Delete Book - Sequence Diagram



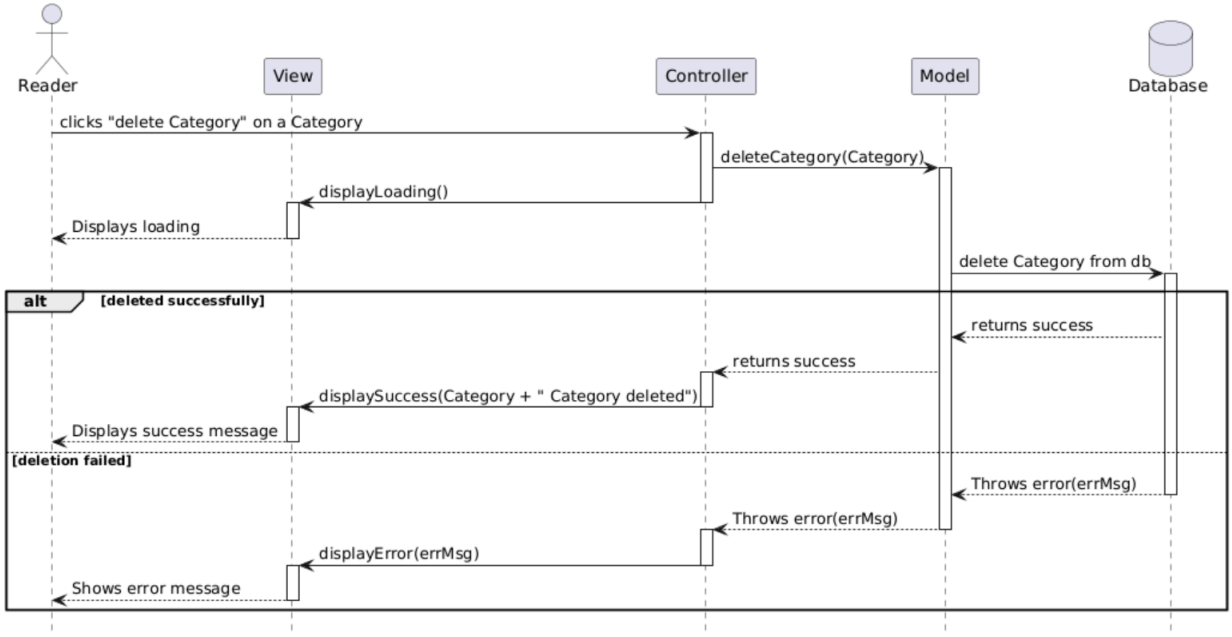
Search Books - Sequence Diagram



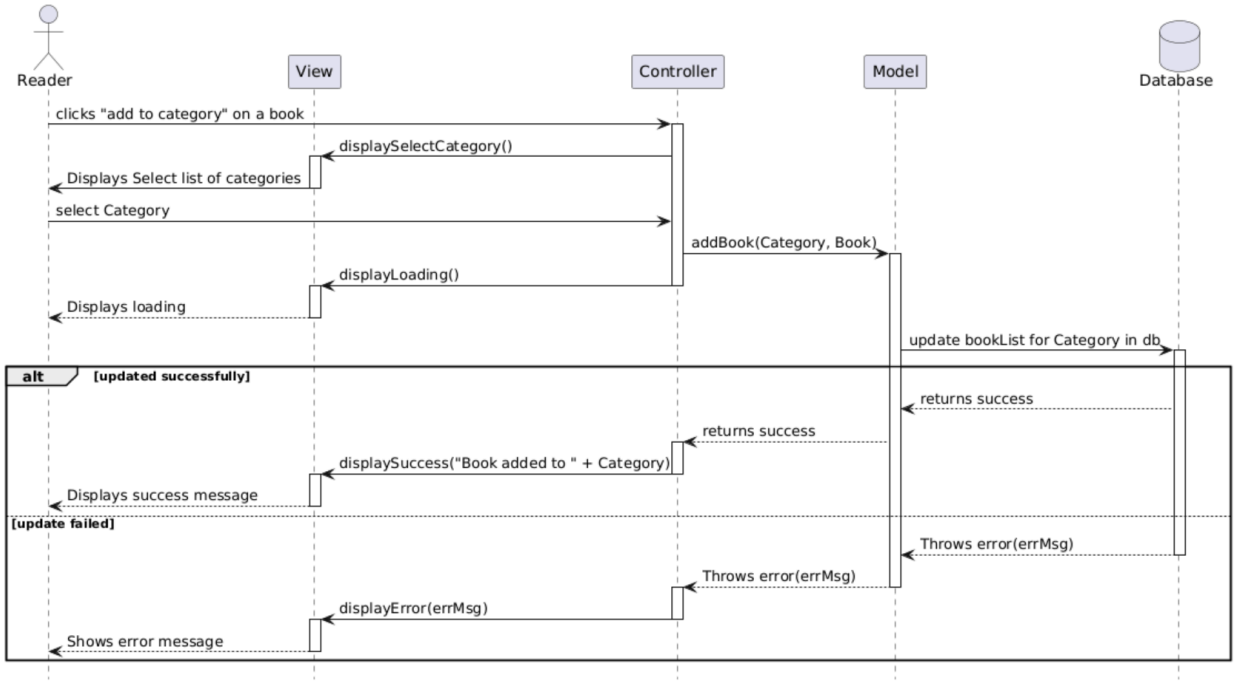
Add Category - Sequence Diagram



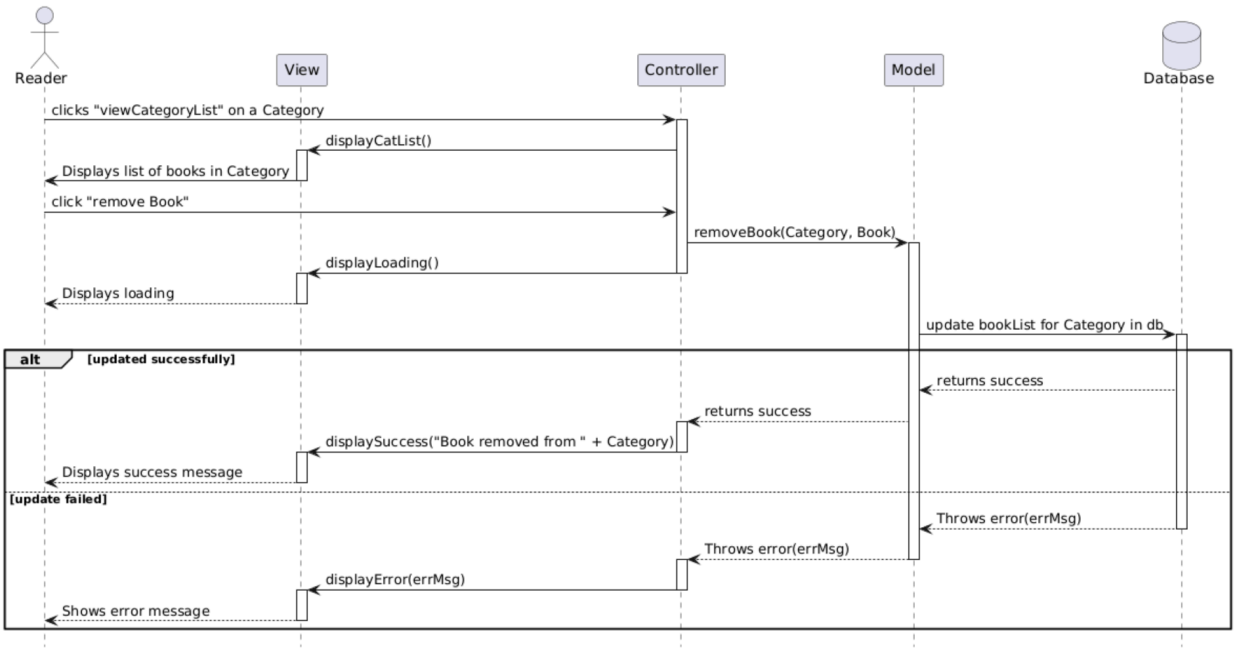
Delete Category - Sequence Diagram



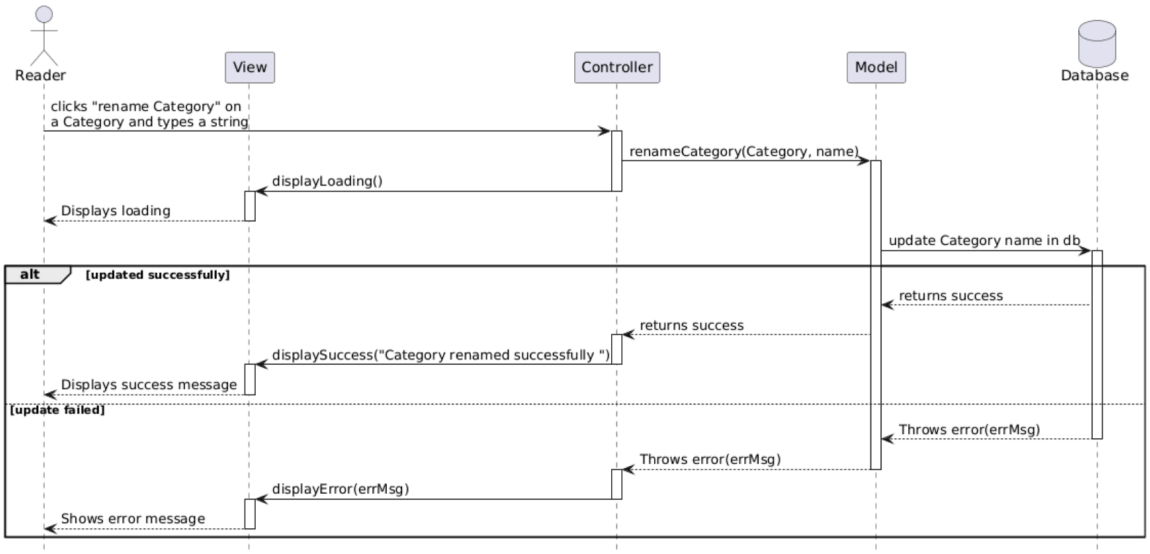
Add Book to Category - Sequence Diagram



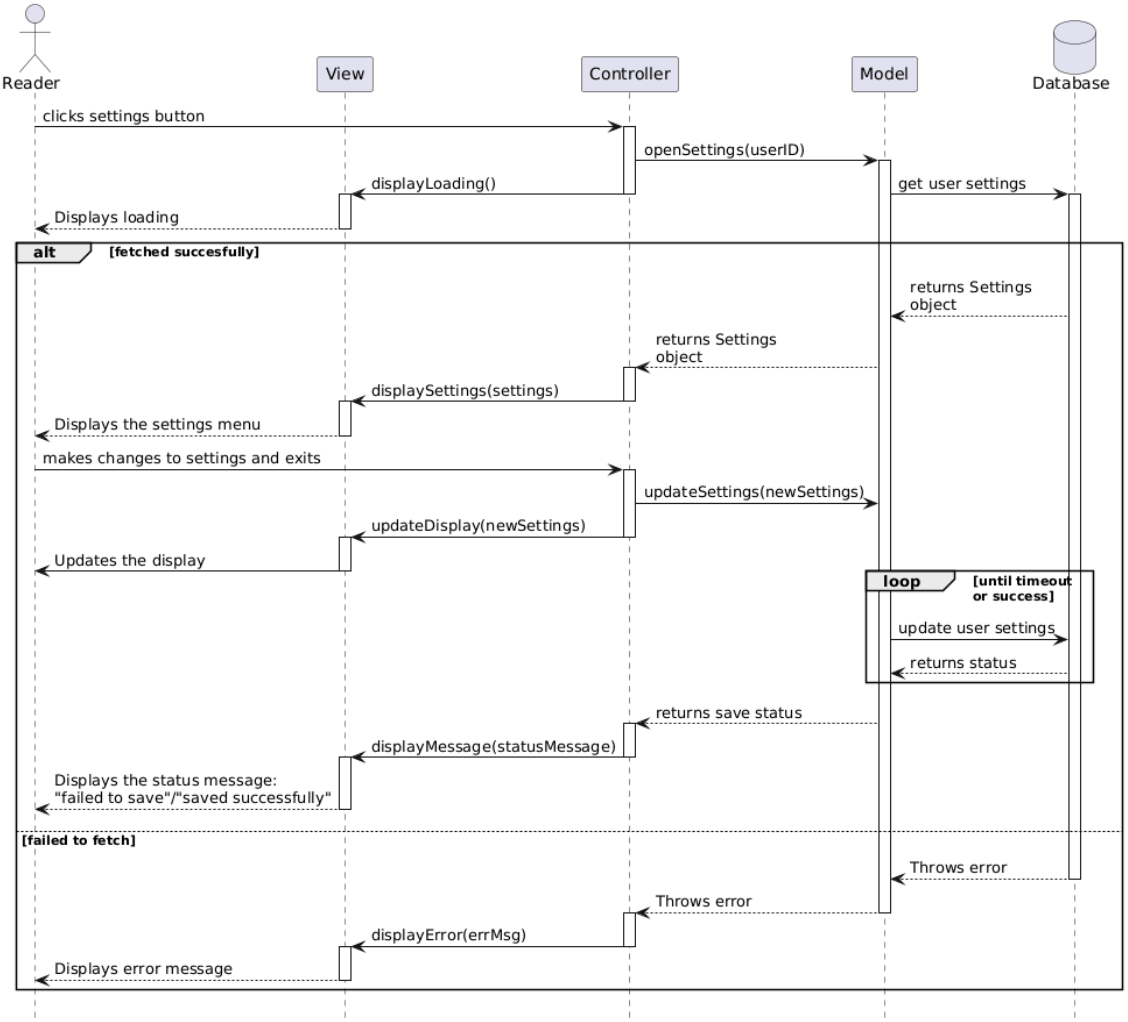
Remove Book from Category - Sequence Diagram



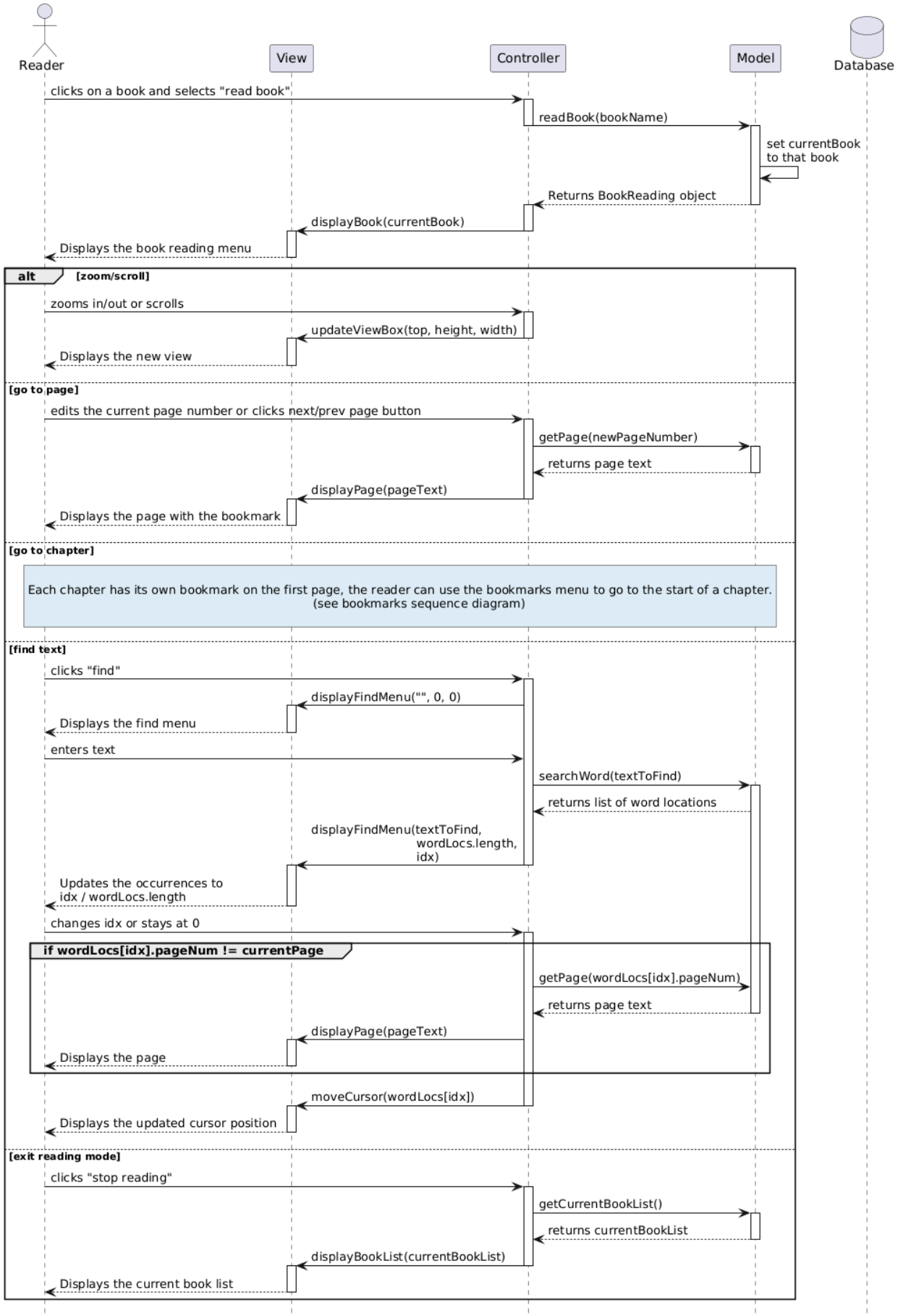
Rename Category - Sequence Diagram

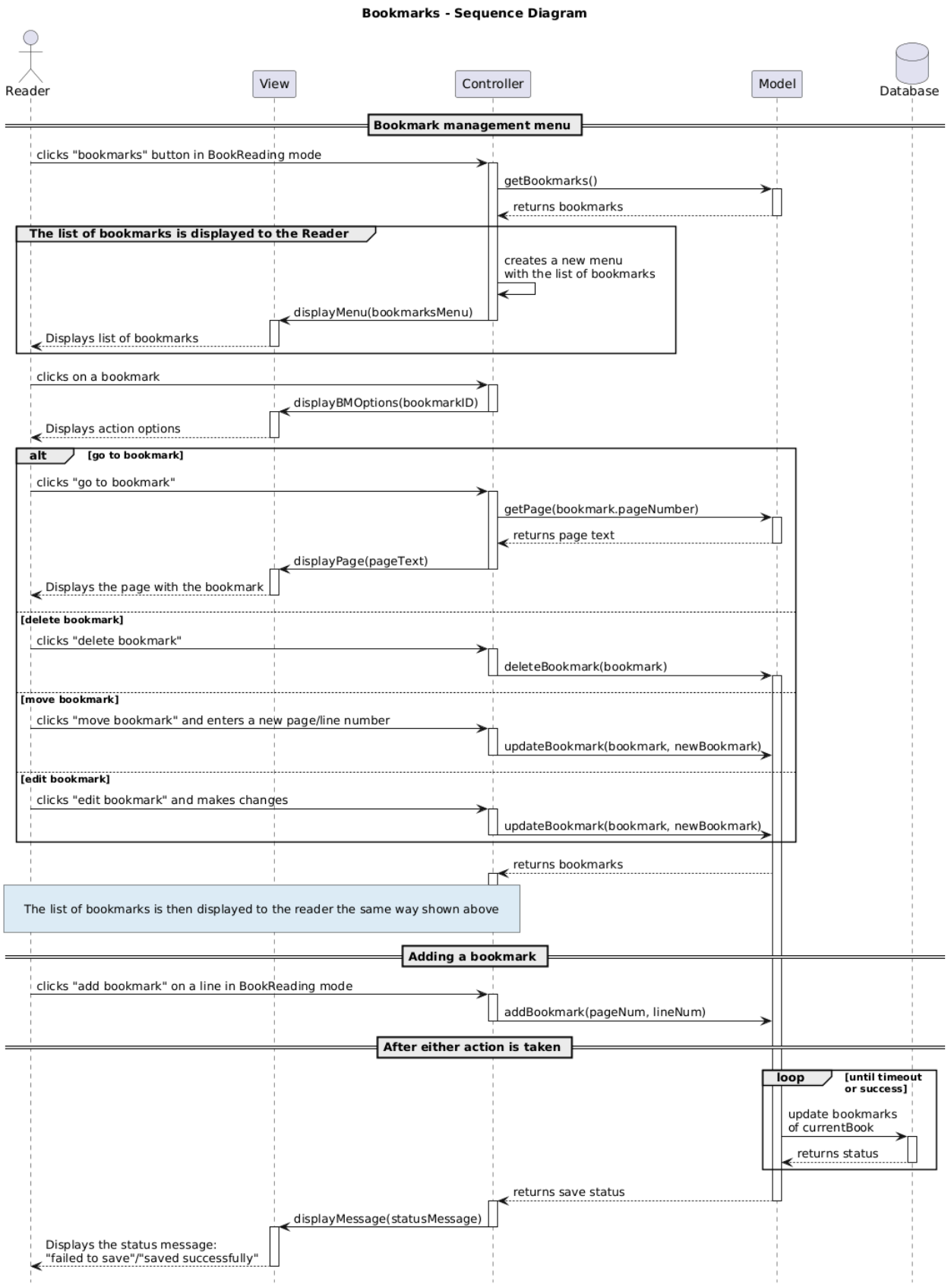


Adjust Settings - Sequence Diagram

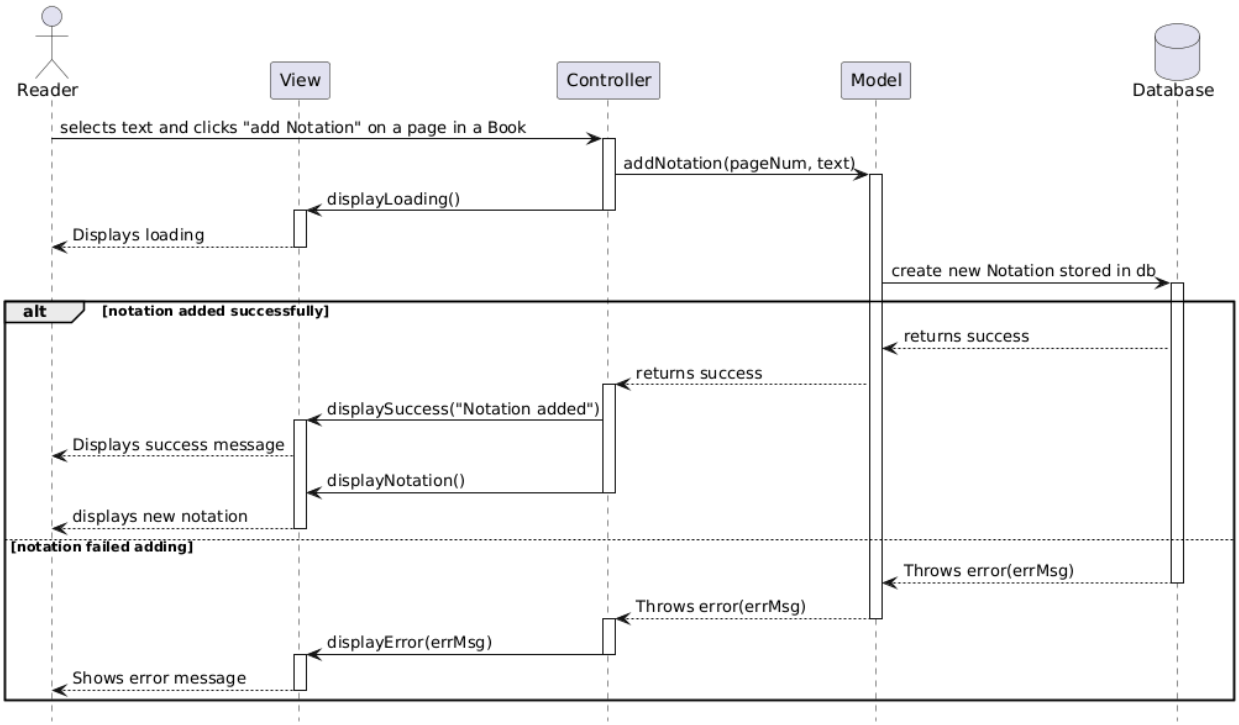


Read Book - Sequence Diagram

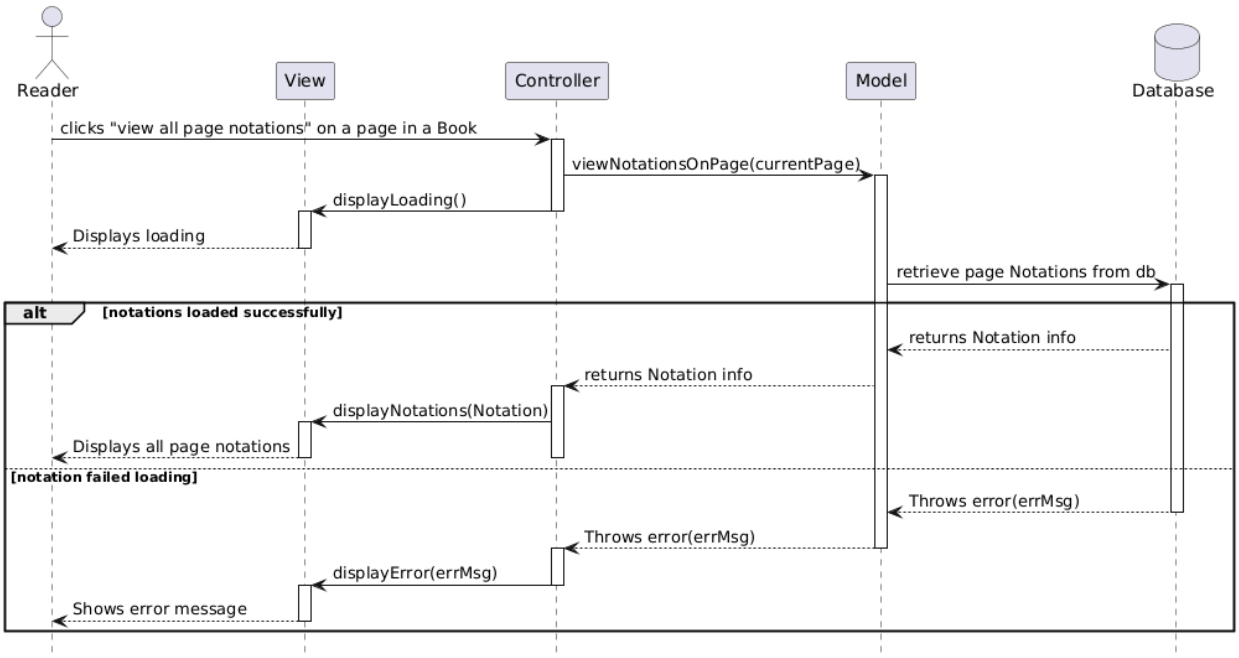



Bookmarks - Sequence Diagram
Reader
View
Controller
Model
Database
Bookmark management menu
clicks "bookmarks" button in BookReading mode
getBookmarks()
returns bookmarks
The list of bookmarks is displayed to the Reader
creates a new menu
with the list of bookmarks
displayMenu(bookmarksMenu)
Displays 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
Adding a bookmark
clicks "add bookmark" on a line in BookReading mode
addBookmark(pageNum, lineNum)
After either action is taken
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"

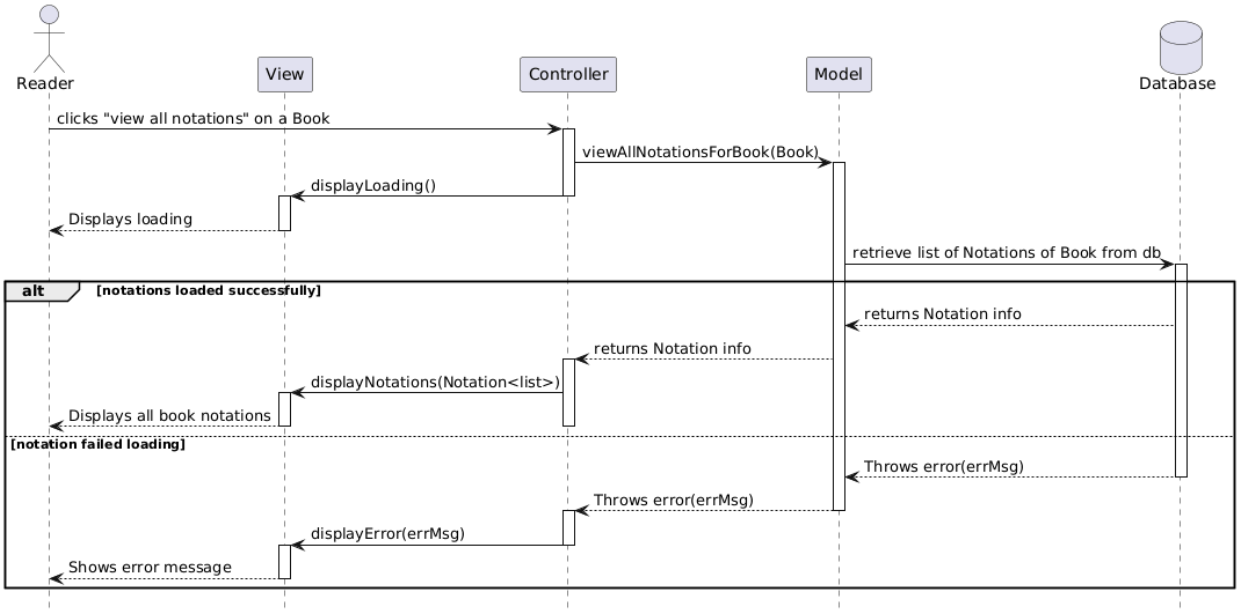
Add Notation - Sequence Diagram



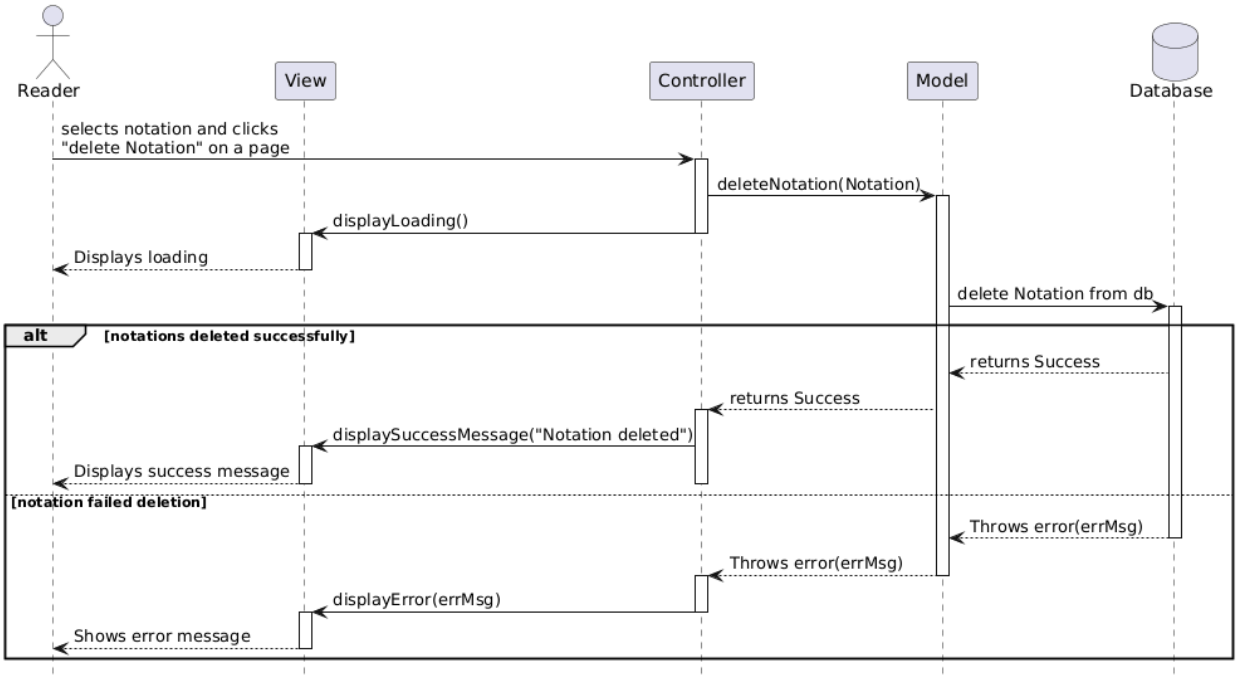
View Page Notations - Sequence Diagram



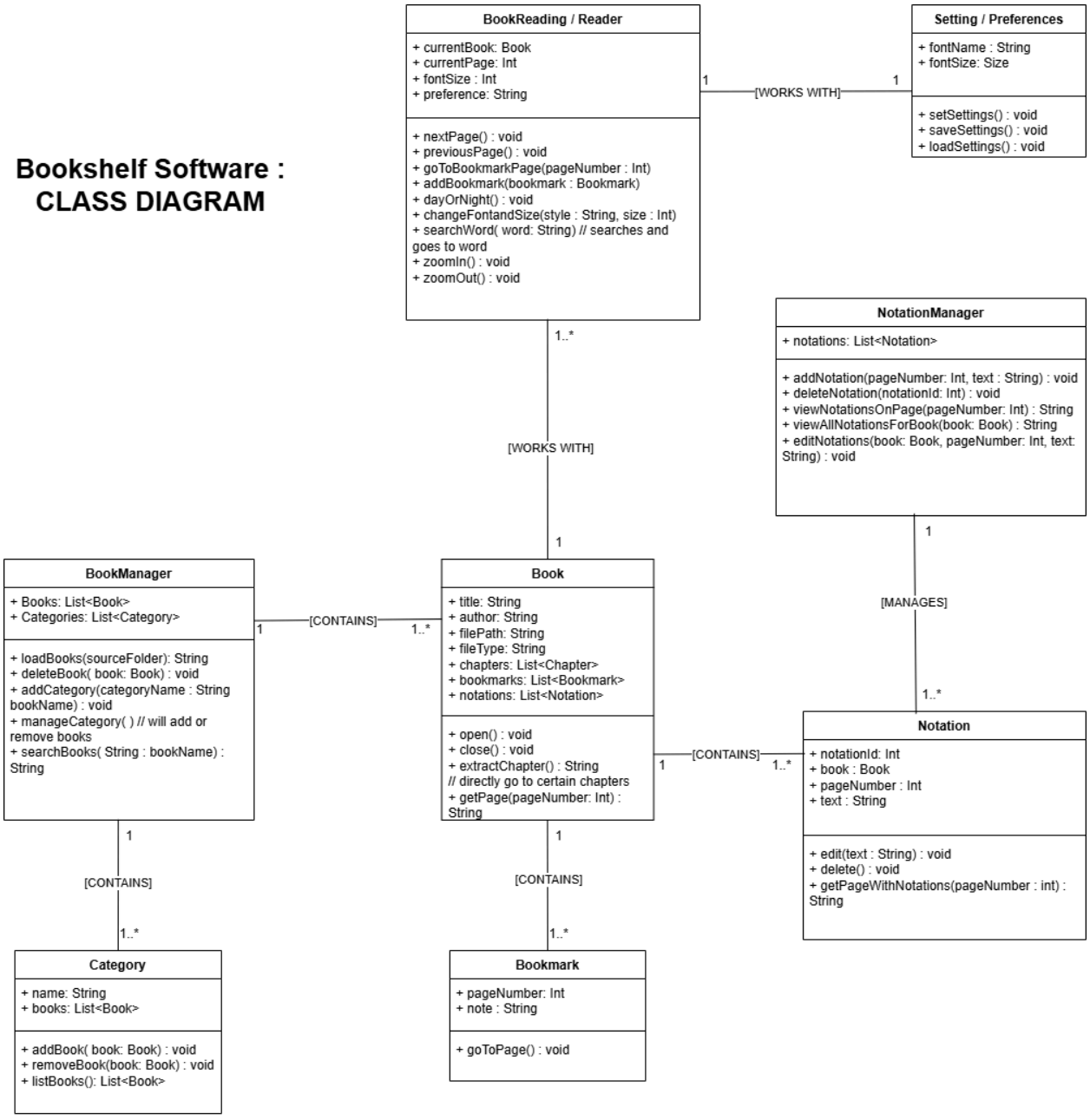
View all Book Notations - Sequence Diagram



Delete Notation - Sequence Diagram



Class diagram – Provide a class diagram of your project.



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)

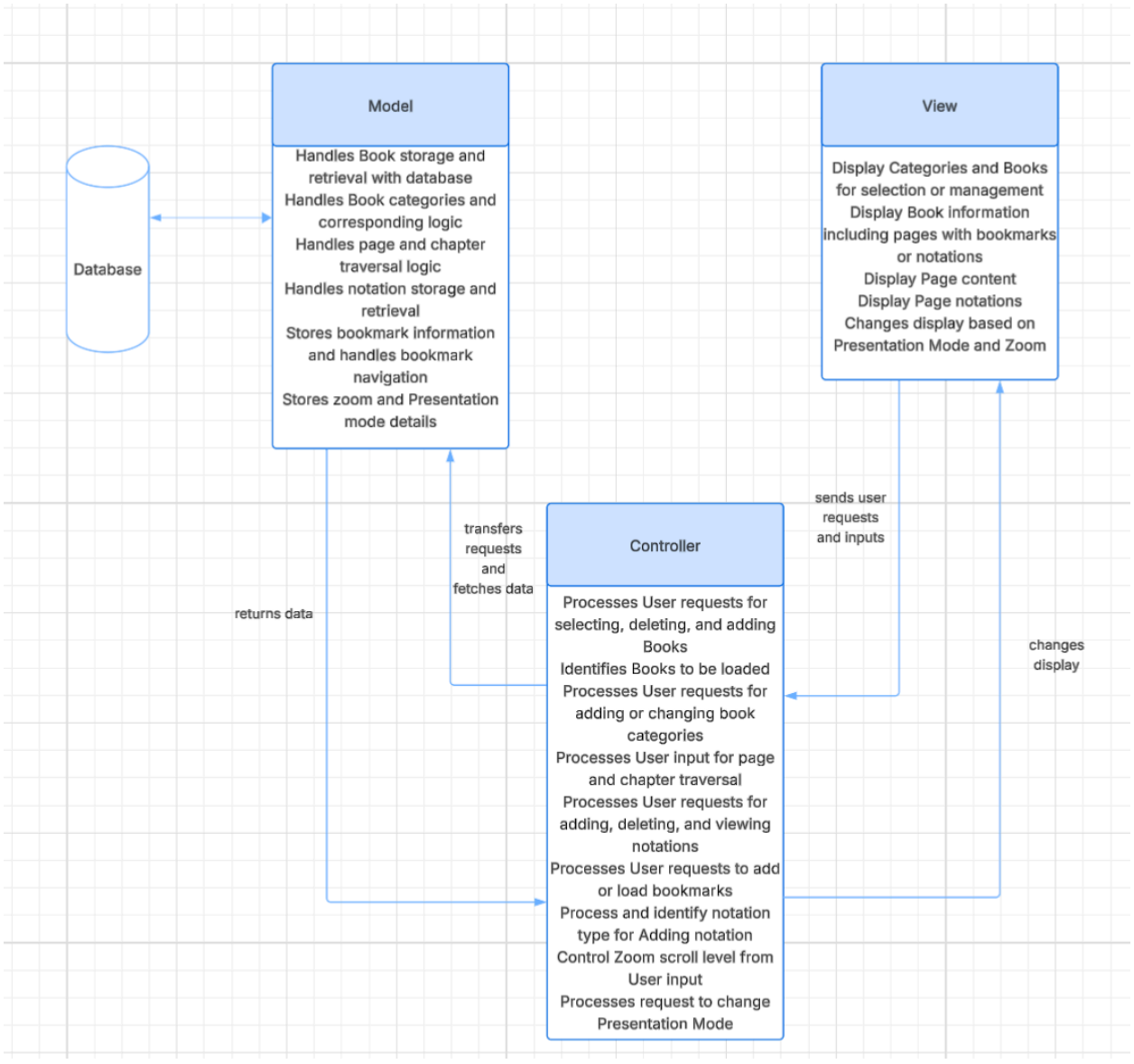
i. Model-View-Controller (MVC) pattern (Figure 6.6)

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)



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.