Book Brief

Nhat Tran, Dheeraj Gollu, Trey Speidel

Software Design Specification and Requirements Document

Version: (1) **Date: (12/12/2024)**

Table of Contents

1 Introduction

This document describes all data, architectural, interface, and component-level design for the software.

1.1 Goals and Objectives

Book Brief is a web application that allows users to search for books, view book summaries, and discover similar books using AI-powered recommendations.

The objective is to help simplify finding engaging books to read by providing concise summaries and book recommendations.

1.2 Product context

The end product will utilize the Google Books API for providing book data, and the Gemini API for generative responses. The system will provide swift and comprehensive book summaries, offer relevant book recommendations, and reduce time spent researching books.

1.3 Statement of System Scope

The software will retrieve book information from the Google Books API. The software will generate book summaries using the Gemini API, displaying them within a responsive web interface for book search and discovery. It will allow users to explore book details as well as find similar books in a time-efficient way, with real-time information retrieval and generation.

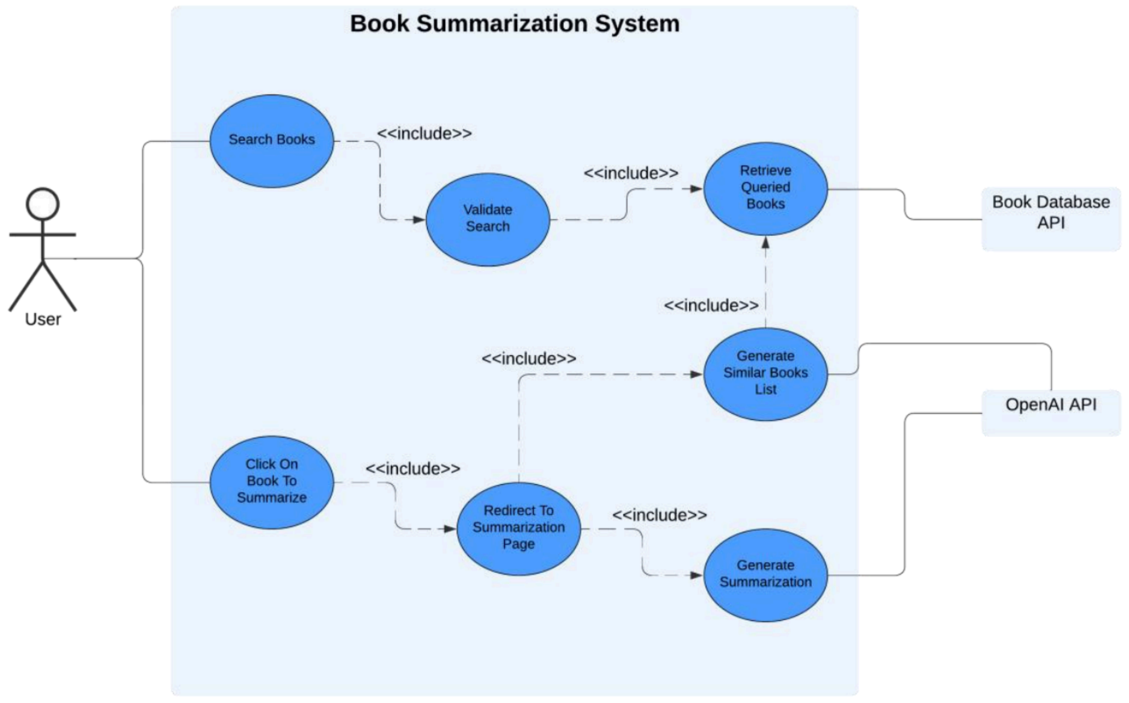


Figure 1: UML Use Case Diagram

2 Architectural design

2.1 System Architecture

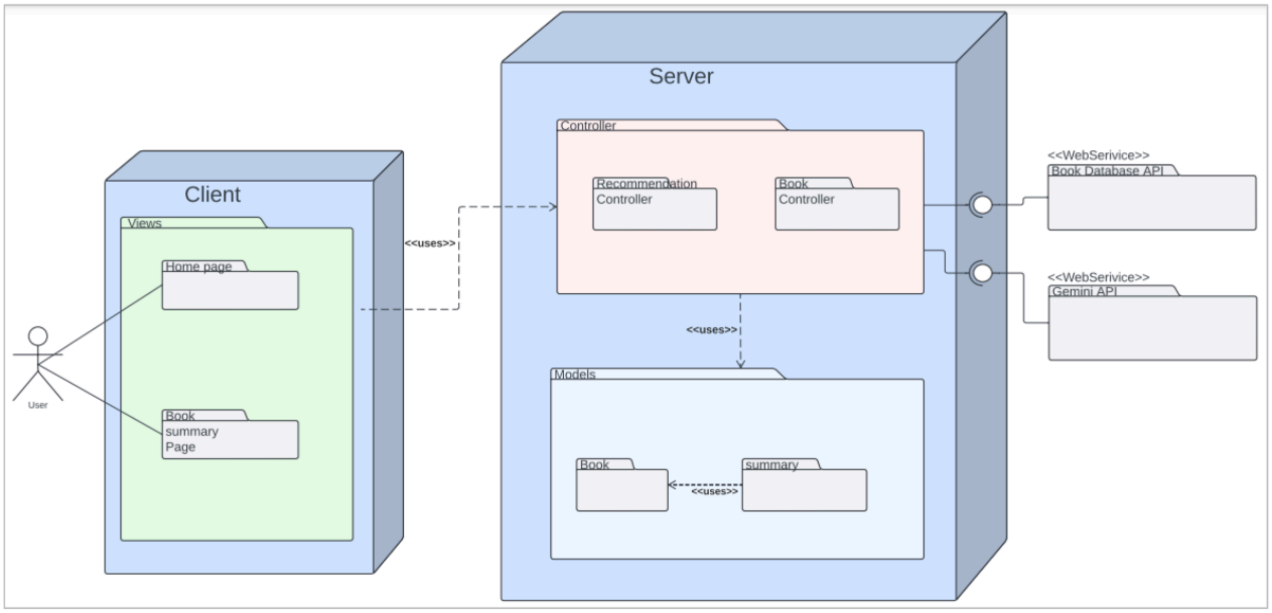


Figure 2: System Architecture Diagram

The system follows a simple modular modern web application architecture. Its front-end is made with React.js, containing the HomePage and SummaryPage modules. The back-end is done with an Express.js server with one endpoint, summarizeBook. The front-end utilizes user input in the search bar, processing it into queries for the Google Books API. The frontend sends the Google Books information down to the backend through its endpoint. The back-end, prompts and handles the summary and recommendation responses from the Gemini API, returning the strings as an HTTP response. The front-end embeds the contents of the HTTP response into the UI, displaying the information to the user.

2.2 Design Rationale

Critical issues for the design rationale of the system include modularity, scalability, performance, and user experience. The system architecture utilizes a separate, modular front and back-end, with pages on the front-end distinct in their responsibilities. It was important to limit the number of needed callbacks between the front-end and back-end, so requests are posted and information is sent down only when necessary. Our architecture also allows for easy scalability, with a services-based approach containing independent components within the high-level modules. To balance complexity with improved performance and a highly intuitive user experience, a single-html-page application style was used with additional focusable windows and information.

3 Key Functionality design

3.1 Book Search

3.1.1 Book Search Use Cases

The user enters a search query in the search bar on the HomePage. The application sends a request to the Google Books API with the user's query. The API returns a list of books matching the search criteria, which is then displayed to the user on the HomePage.

Use Case ID:	UC-002
Use Case Name:	Search for Books
User Goal:	To search for books.
Scope:	Book Summarization System
Level:	Primary
Primary Actor:	User
Precondition:	The user must know the name of the book they are searching for.
Minimal Guarantee:	An error message stating that books cannot be found.
Success Guarantee:	The user can search for their books.
Trigger:	The user selects the search option.
Success Scenario:	**Step Actions**
	1. The user attempts to search for the book.
	2. Validate the user input.
	3. The system displays all the book suggestions.
Extensions:	**Branching Scenarios**
1A	**Condition:** The user is unable to find the book.
	Step Actions
	1. The system displays a message indicating that no result was found for the book.
	2. The user returns to main step 1.

2A	**Condition:** The user input contains invalid characters
	Step Actions
	1. The system displays an error stating that the query is invalid.
	2. The user returns to main step 1.
3A	**Condition:** The system fails to display
	1. The system displays an error stating that books cannot be displayed.
	2. The user returns to main step 1.

Table 1: Search for Books Use Case

3.1.2 Processing Sequence for Book Search

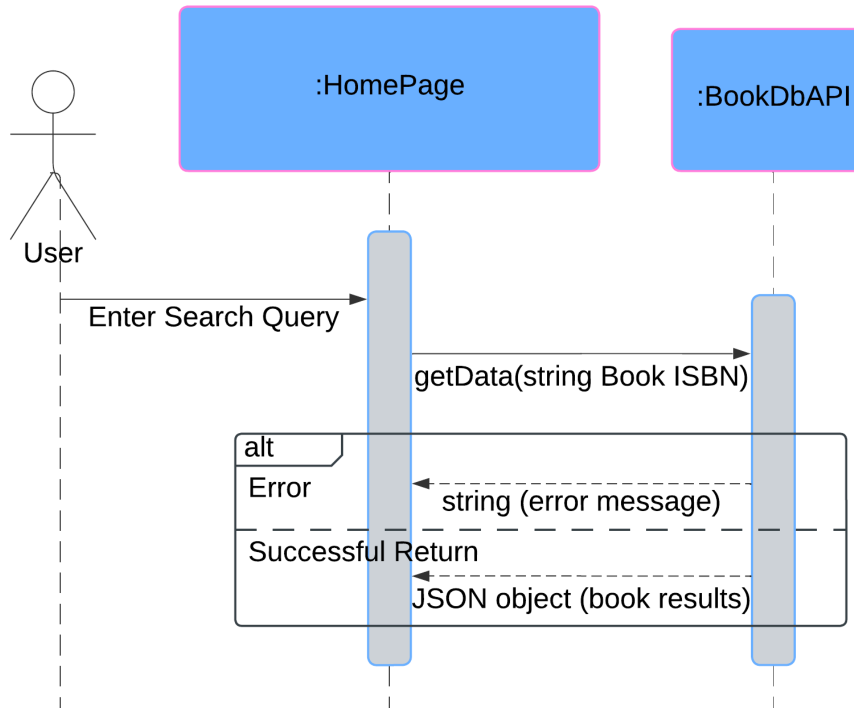


Figure 3: Book Search Sequence Diagram

The user enters a search query in the search bar on the HomePage. The application sends a GET request to the Google Books API with the user's query. The API returns a JSON object list of books matching the search criteria, which is then displayed to the user on the HomePage. The user can then select any of these displayed cards to view individual book information.

3.1.3 Structural Design for Book Search

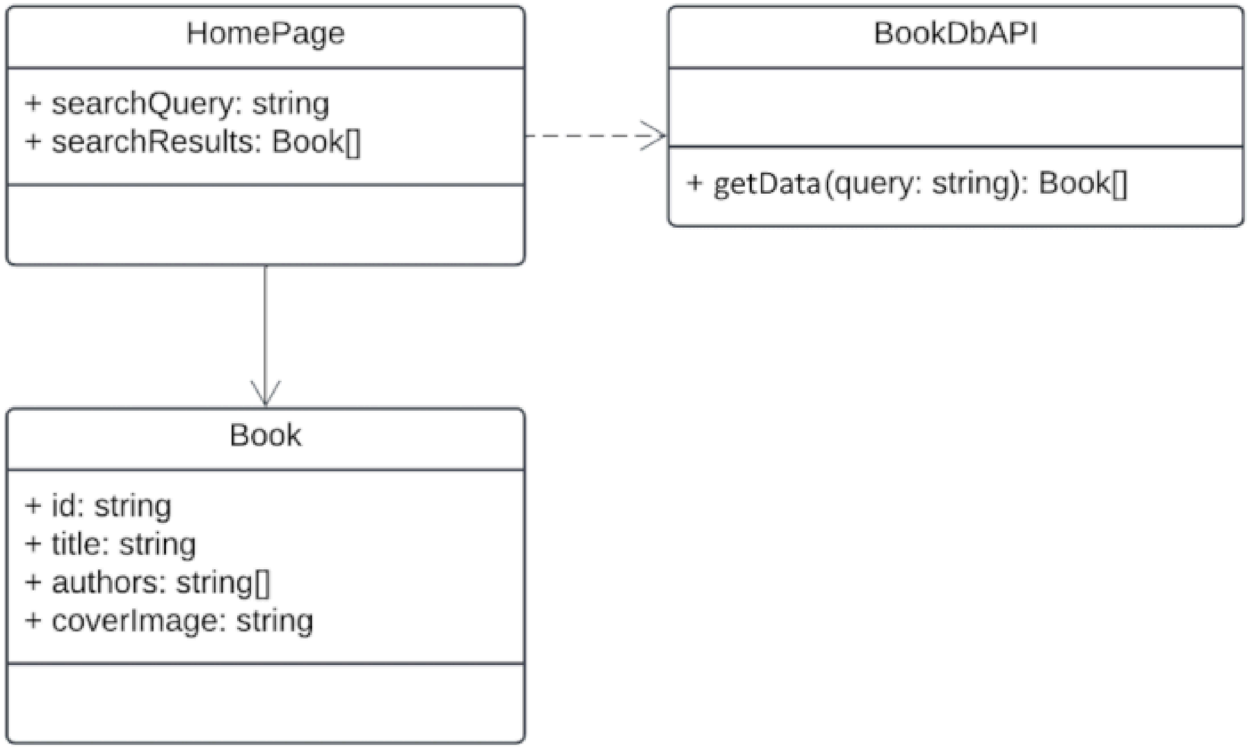


Figure 4: Book Search Structure Diagram

3.1.4 Key Activities

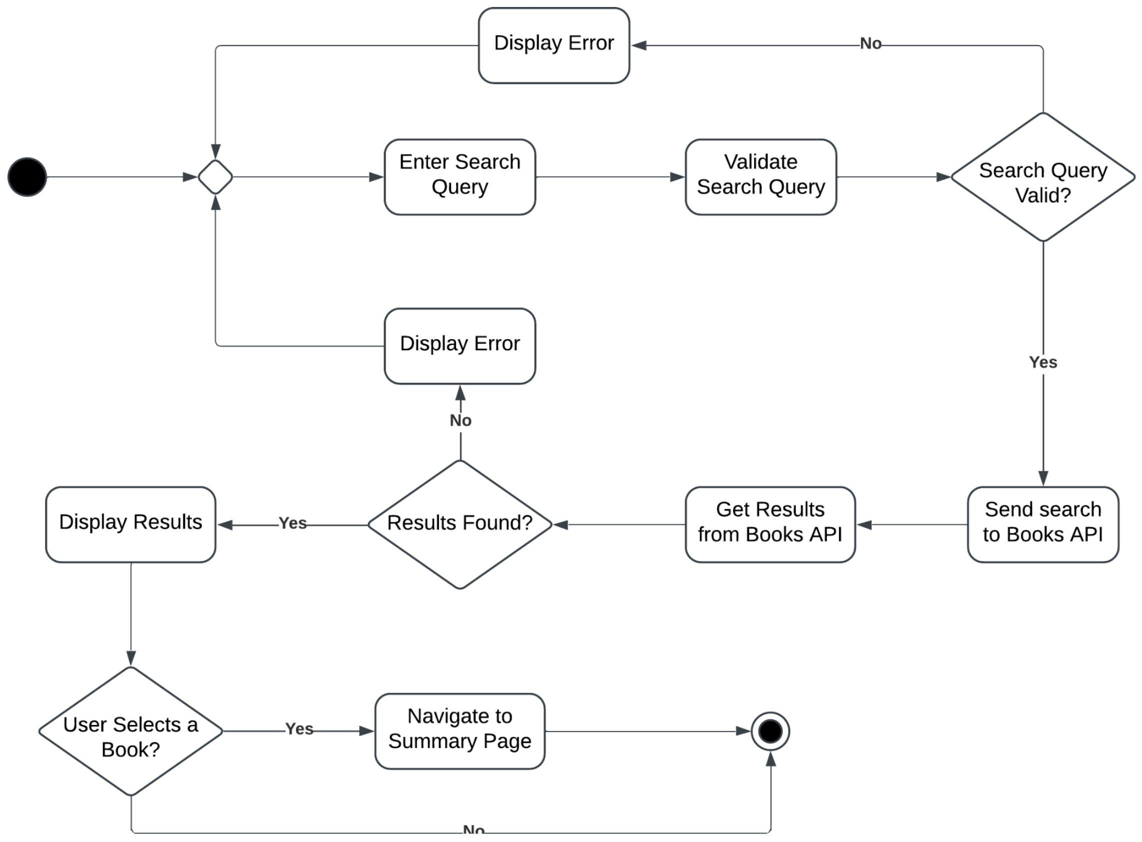


Figure 5: Book Search Activity Diagram

3.1.5 Software Interface to Other Components

The Book Search functionality interacts with the Google Books API to retrieve book information based on user queries.

3.2 Book Summary Generation

3.2.1 Book Summary Generation Use Cases

When a user clicks on a book from the search results or the discovery queue, they are navigated to the SummaryPage. The application sends a POST request to the back-end with the book's title and author. The back-end uses the Gemini AI API to generate a summary of the book, which is then returned to the front-end and displayed on the SummaryPage.

Use Case ID:	UC-001
Use Case Name:	Book Summarization Summary
User Goal:	To summarize a book based on user input.
Scope:	Book Summarization System

Level:	Summary
Primary Actor:	User
Precondition:	The user intends to search for a book to receive its summary.
Minimal Guarantee:	The user is unable to access the system.
Success Guarantee:	Users would be able to view a summary of a book.
Trigger:	The user opens the website.
Success Scenario:	**Step Actions**
	1. The user searches for books.
	2. The user selects a book from the search results.
	3. The user accesses the book summary section.
	4. The user views the book summary and metadata.
	5. The user can return to the previous section, start a new search, or choose a book from the list of five similar titles.
Extensions:	**Branching Scenarios**
1A	**Condition:** The user is unable to find the book.
	Step Actions
	1. The system displays a message indicating that no result was found for the book.
	2. The user returns to main step 1.
2A	**Condition:** The system fails to let the user select the book.
	Step Actions
	1. The system displays an error stating can't select the book.
	2. The user returns to main step 1.
4A	**Condition:** The system fails to generate the summarization.
	Step Actions

	1. The system displays an error stating that it cannot summarize the book.
	2. The user returns to main step 1.
5A	**Condition:** The system fails to let the user go back to the previous section.
	Step Actions
	1. The system displays an error stating it can't return to the previous section.
	2. The user goes back to main step 1.
5B	**Condition:** The system fails to provide five similar book titles.
	Step Actions
	1. The system displays a message stating that similar titles were not generated.
	2. The user goes back to main step 1.

Table 2: Book Summarization Summary Use Case

Use Case ID:	UC-003
Use Case Name:	Generate Summarization
User Goal:	To get a summary of a book
Scope:	Book Summarization System
Level:	Primary
Primary Actor:	User
Precondition:	The user searches for a book to summarize
Minimal Guarantee:	An error message stating that a summary of the book cannot be found.
Success Guarantee:	The user can get a summary of the book they selected.
Trigger:	The user selects a desired book from the suggestions.
Success Scenario:	**Step Actions**

	4. The user selects the desired book.
	5. The system displays book information and summary.
Extensions:	**Branching Scenarios**
1A	**Condition:** The user cannot select their book.
	Step Actions
	3. The system displays an error message indicating that the book cannot be selected.
	4. The user returns to main step 1.
2A	**Condition:** The system fails to generate the summarization.
	Step Actions
	3. The system displays an error stating that it can't summarize the book.
	4. The user returns to main step 1.

Table 3: Generate Summarization Use Case

3.2.2 Processing sequence for Book Summary Generation

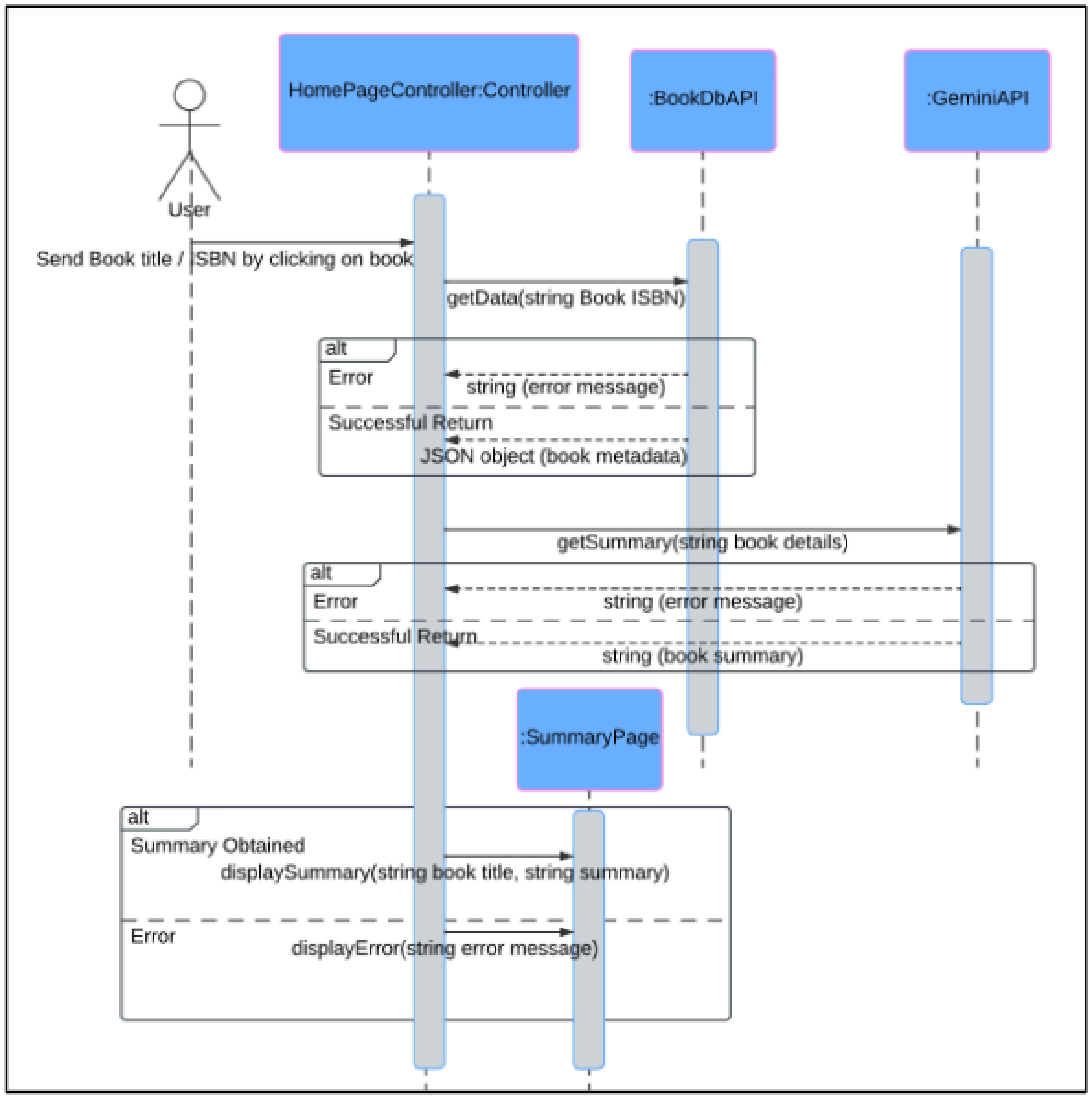

HomePageController:Controller
:BookDbAPI
:GeminiAPI
User
Send Book title / ISBN by clicking on book
getData(string Book ISBN)
alt
Error
string (error message)
Successful Return
JSON object (book metadata)
getSummary(string book details)
alt
Error
string (error message)
Successful Return
string (book summary)
:SummaryPage
alt
Summary Obtained
displaySummary(string book title, string summary)
Error
displayError(string error message)

Figure 6: Book Summary Generation Sequence Diagram

The user clicks on a book from the search results or the discovery queue on the HomePage. The application navigates to the SummaryPage, passing the book's title, author, and cover image as state. The application sends a POST request to the back-end's /summarizeBook endpoint with the book's title and author. The backend uses the Google Generative AI API to generate a summary of the book. The back-end returns the generated summary as a JSON response to the front-end. The application displays the book's summary on the SummaryPage.

3.2.3 Structural Design for Book Summary Generation

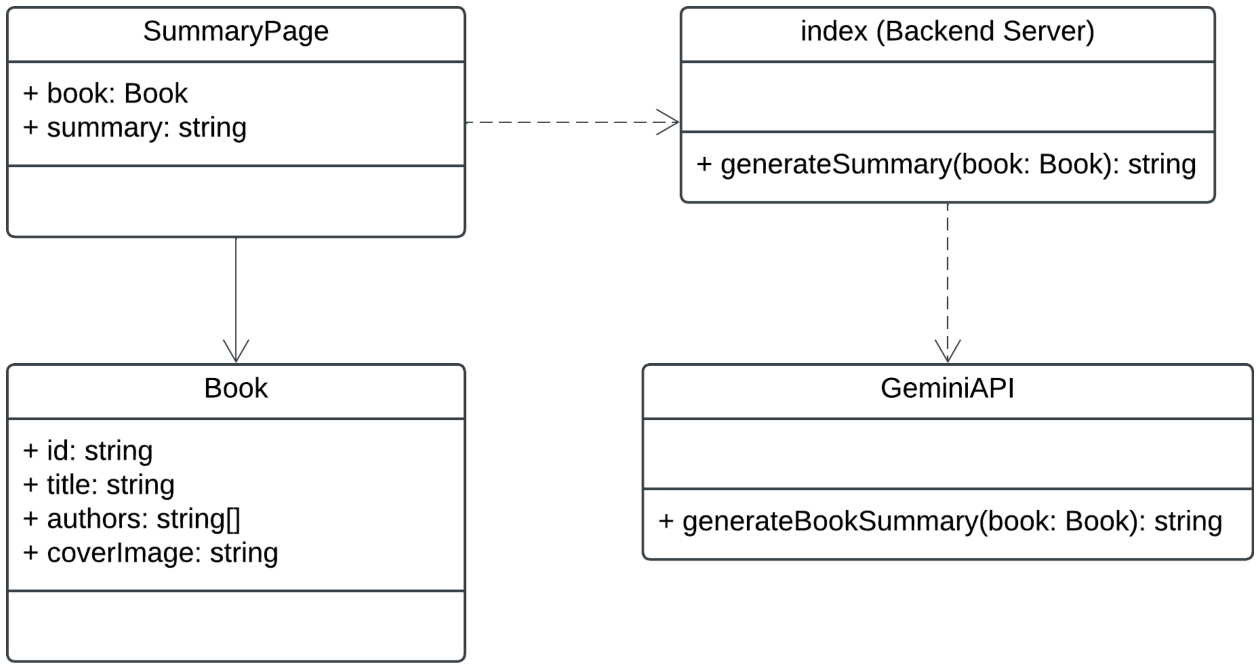


Figure 7: Book Summary Generation Class Diagram

3.2.4 Key Activities

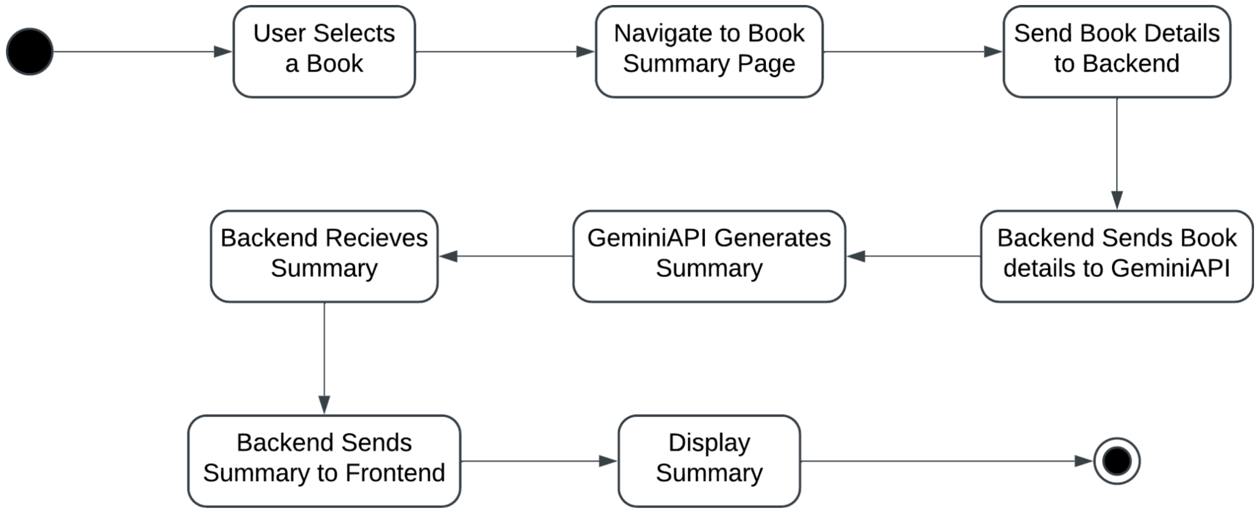

User Selects a Book
Navigate to Book Summary Page
Send Book Details to Backend
Backend Sends Book details to GeminiAPI
GeminiAPI Generates Summary
Backend Recieves Summary
Backend Sends Summary to Frontend
Display Summary

Figure 8: Book Summary Generation Activity Diagram

3.2.5 Software Interface to other components

The Book Summary Generation functionality interacts with the backend's /summarizeBook endpoint and the Google Generative AI API to generate book summaries.

3.3 Similar Books Recommendation

3.3.1 Similar Books Recommendation Use Cases

After generating a book summary, the back-end also returns a list of similar books in JSON format. The front-end displays these similar books as cards on the SummaryPage, allowing users to click on them to view their summaries.

Use Case ID:	UC-004
Use Case Name:	Generate Similar Books List
User Goal:	Suggest all the related books for the user.
Scope:	Book Summarization System
Level:	Sub-Function
Primary Actor:	User
Precondition:	The user searched for a book to summarize.
Minimal Guarantee:	A message indicating that no related books were found.
Success Guarantee:	The system displays book suggestions for the user.
Trigger:	The user selects a book from the book suggestion.
Success Scenario:	**Step Actions**
	1. The user selects a book to summarize.
	2. The system displays five related books based on the selected book's metadata.
Extensions:	**Branching Scenarios**
1A	**Condition:** The system is not able to summarize the book.
	Step Actions
	1. The system displays a message indicating that a summary cannot be generated at the moment and asks the user to try again.
	2. The user returns to main step 1.
2A	**Condition:** The system fails to display book suggestions for the user.

	Step Actions
	1. The system displays an error stating that similar suggested books cannot be displayed.
	2. The user returns to main step 1.

Table 4: Generate Similar Books List Use Case

3.3.2 Processing Sequence for Similar Books Recommendation

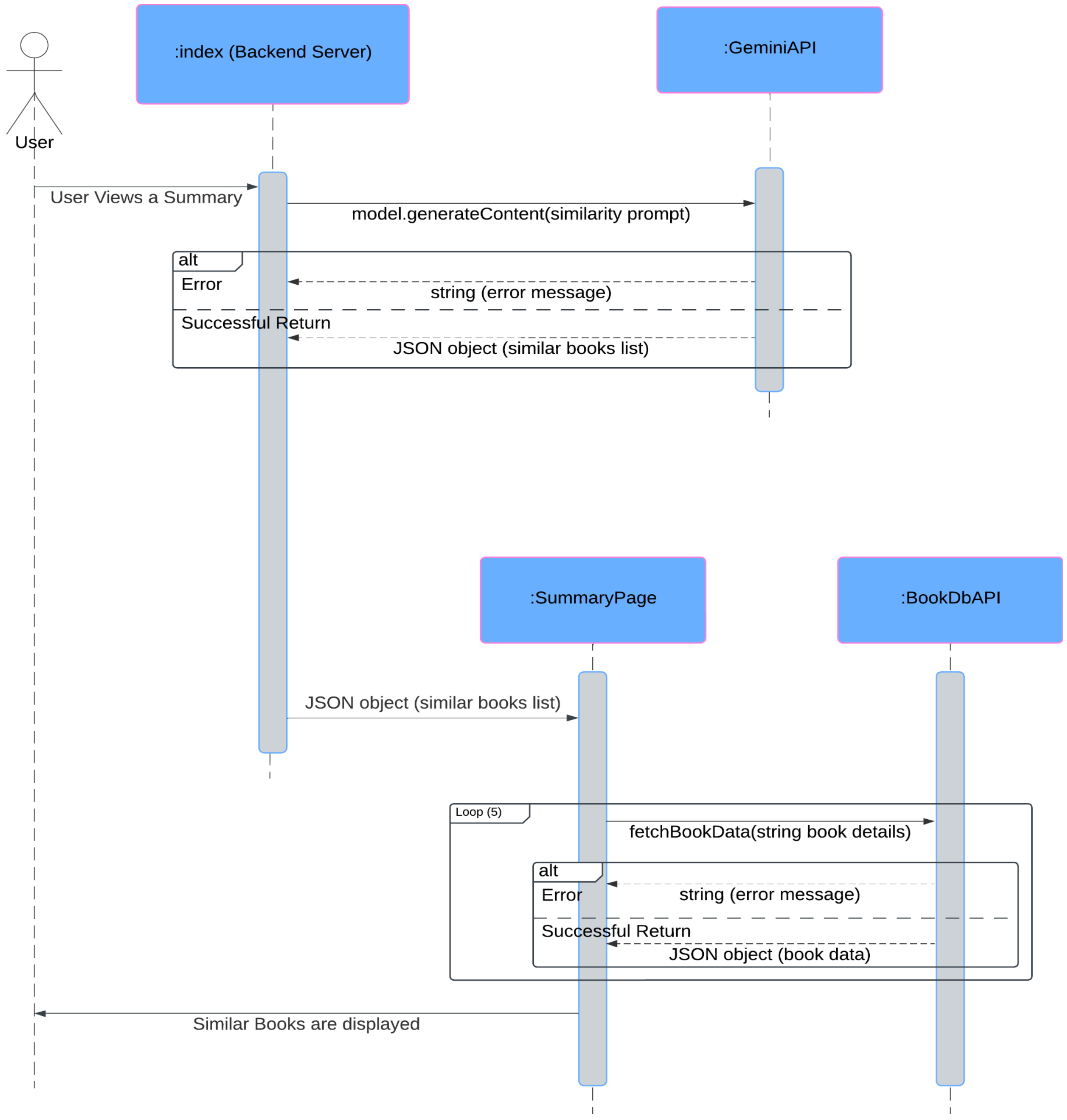


Figure 9: Similar Books Recommendation Sequence Diagram

The back-end generates a list of similar books along with the book summary using the Google Generative AI API. The back-end includes the list of similar books in the JSON response sent to the front-end. The front-end retrieves the book information for each similar book using the Google Books API. The application displays the similar books on the SummaryPage. When a user clicks on a similar book, the application navigates to the SummaryPage for that book, repeating the Book Summary Generation process.

3.3.3 Structural Design for Similar Books Recommendation

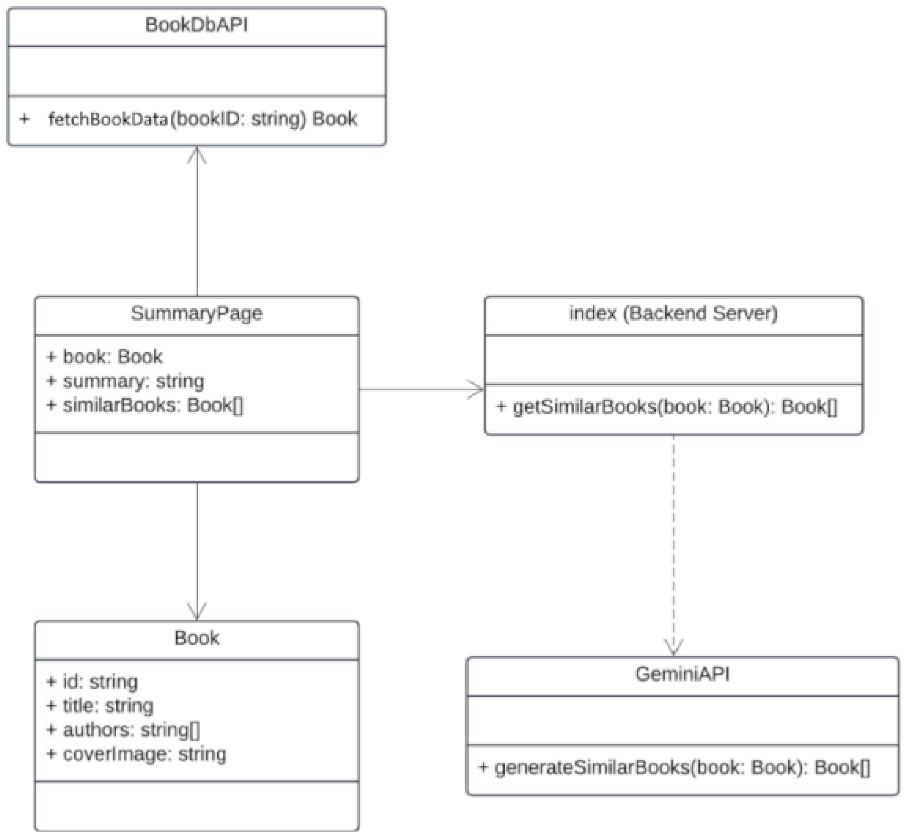


Figure 10: Similar Books Recommendation Class Diagram

3.3.4 Key Activities

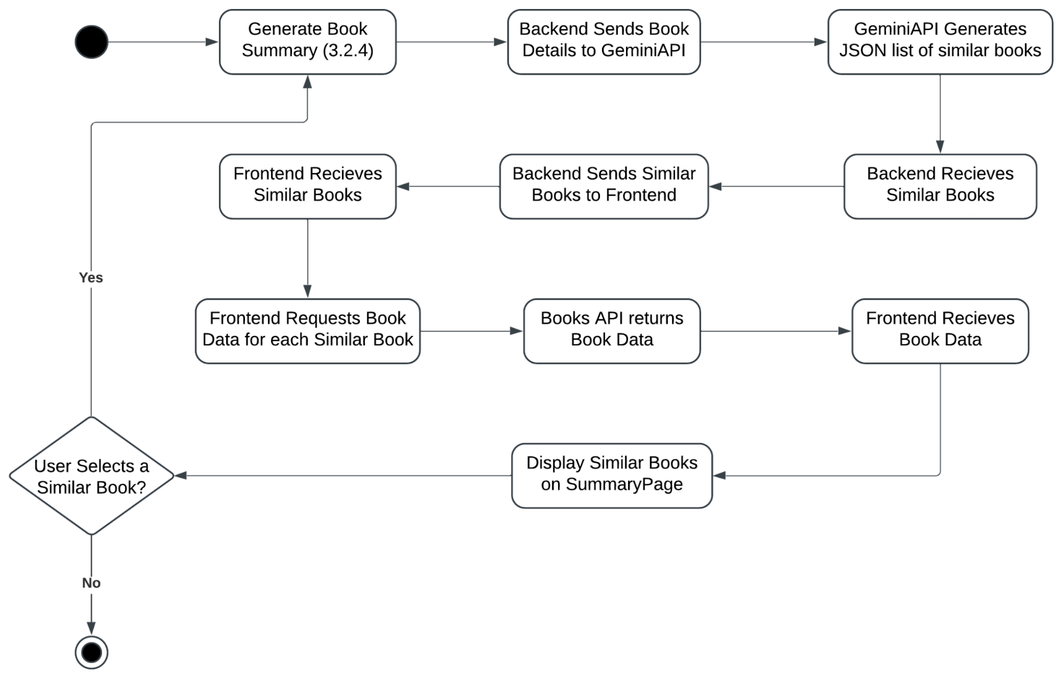

Generate Book Summary (3.2.4)
Backend Sends Book Details to GeminiAPI
GeminiAPI Generates JSON list of similar books
Backend Recieves Similar Books
Backend Sends Similar Books to Frontend
Frontend Recieves Similar Books
Frontend Requests Book Data for each Similar Book
Books API returns Book Data
Frontend Recieves Book Data
Display Similar Books on SummaryPage
User Selects a Similar Book?
Yes
No

Figure 11: Similar Books Recommendation Activity Diagram

3.3.5 Software Interface to Other Components

The Similar Books Recommendation component interacts with the Google Books API to retrieve book information for the recommended similar books.

4 User interface design

4.1 Interface design rules

BookBrief uses a modern, intuitive, and visually appealing design. Design principles adhered to include consistency, clarity, and responsiveness. UI components like input fields, buttons, colors, and fonts maintain a consistent style throughout the application. For clarity, the layout and navigation are very front and center to the user, with no deeply navigable menus to get lost in. The UI is also designed to respond to user inputs quickly and visually, giving the application a highly interactive look and feel.

4.2 Description of the user interface

The Book Brief application consists of two main pages: the HomePage and the SummaryPage. Each page is designed to provide a specific set of functionalities and information to the user.

4.2.1 Home Page

The HomePage serves as the landing page of the Book Brief application. It provides users with a search functionality to find books and displays a discovery queue of randomly selected books.

4.2.1.1 Screen Images

Shown here is the home page of the web application.

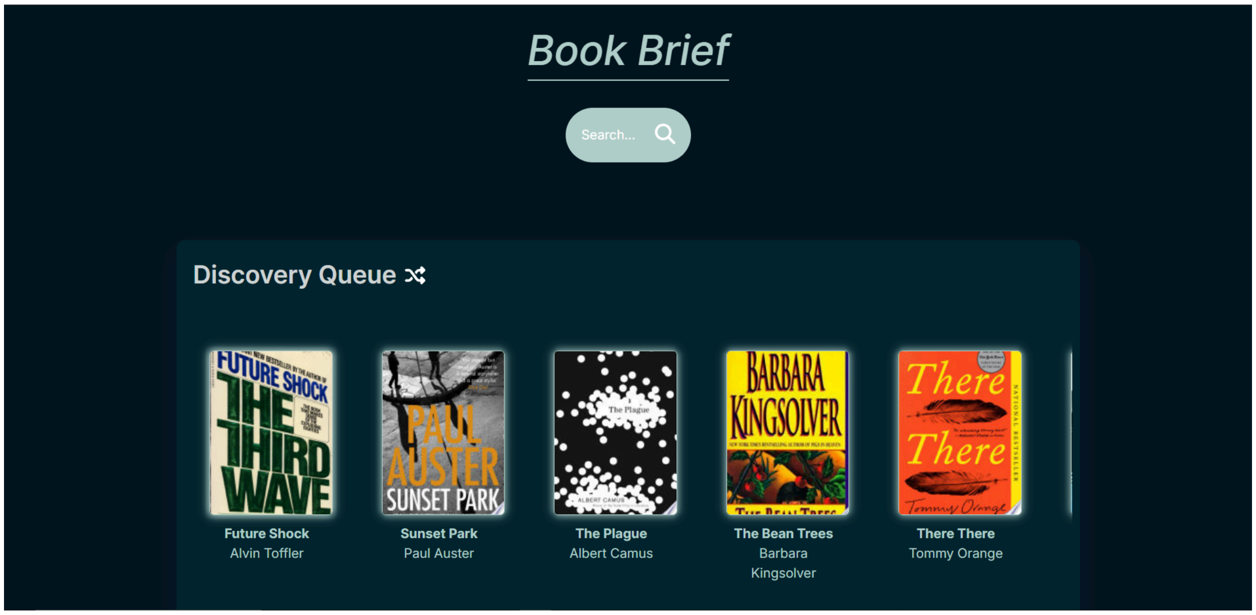


Figure 12.1 - Home Page without a search query

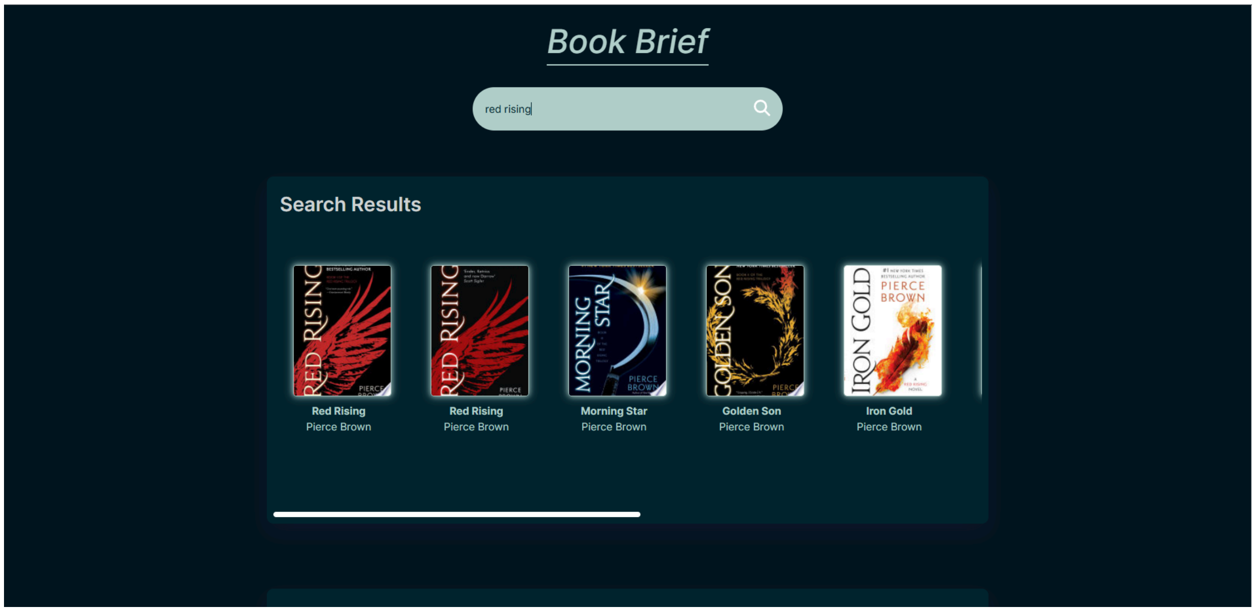


Figure 12.2.1 - Home Page with a search query

4.2.1.2 Objects and Actions

The main objects visible on the HomePage are the search bar, book cards, and discovery queue. The search bar is located at the top-middle of the HomePage below the title, as the first thing users see. Users can execute the search by hitting the Enter key or hitting the search icon button. Book cards are used to display compact book information in a consistent format. Book cards each contain the book cover image, title, and author information. Book cards have a consistent width and height to enable a uniform layout.

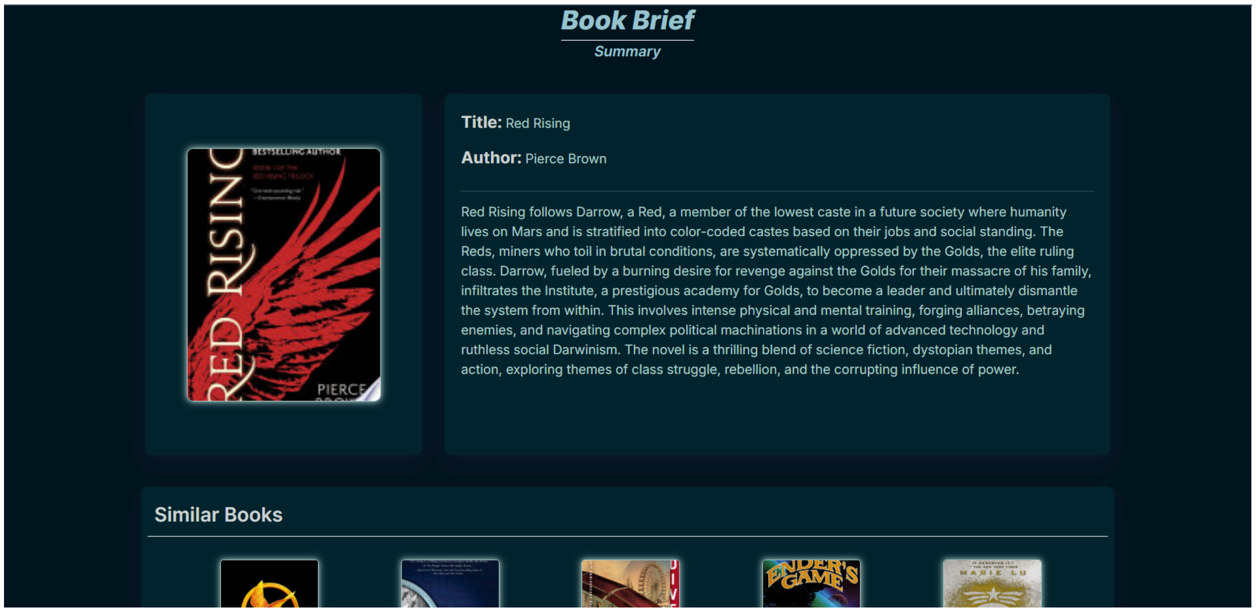
Clicking on a book card navigates the user to the SummaryPage for that specific book. The discovery queue section displays a selection of randomly chosen books for users to explore. It utilizes a horizontal scrolling layout to accommodate a larger number of book cards.

4.2.2 Summary Page

The SummaryPage displays detailed information about a selected book, including its summary and a list of similar books.

4.2.2.1 Screen Images

Shown here is the summary page of the web application.



Title: Red Rising

Author: Pierce Brown

Red Rising follows Darrow, a Red, a member of the lowest caste in a future society where humanity lives on Mars and is stratified into color-coded castes based on their jobs and social standing. The Reds, miners who toil in brutal conditions, are systematically oppressed by the Golds, the elite ruling class. Darrow, fueled by a burning desire for revenge against the Golds for their massacre of his family, infiltrates the Institute, a prestigious academy for Golds, to become a leader and ultimately dismantle the system from within. This involves intense physical and mental training, forging alliances, betraying enemies, and navigating complex political machinations in a world of advanced technology and ruthless social Darwinism. The novel is a thrilling blend of science fiction, dystopian themes, and action, exploring themes of class struggle, rebellion, and the corrupting influence of power.

Figure 13 - Summary Page

4.2.2.2 Objects and Actions

The SummaryPage displays the book cover image, title, and author information as text boxes at the top of the page. Below that information, the page contains the Gemini-generated summary of the book. At the bottom of the page, it includes a list of similar books to the one currently being viewed. Similar books are displayed using the same book card format as on the HomePage. Users can horizontally scroll through the list of similar books to explore related titles. Clicking on a similar book card navigates the user to the SummaryPage for that specific book. A "Back to Home" button is provided on the SummaryPage to allow users to go back to the HomePage. Clicking the "Back to Home" button takes the user back to the main search and discovery interface.

5 Restrictions, limitations, and constraints

- Must work with modern web browsers: Chrome, Firefox, Safari, etc.

- The tool must be lightweight to operate in a browser environment.
- Book data being available or accurate is highly dependent on the Google Books API's coverage.
- An internet connection is required to access the application and for it to function properly.
- The application currently supports only the English language.

6 User Characteristics

6.1 Key Users

Key users will use Book Brief to obtain book information and summaries from a generative AI model. Moreover, these users can utilize the search bar to search for books themselves with certain tags or keywords such as genre, fiction, and author.

Key user's knowledge of books is limited to the extent that they prefer to read books but do not have knowledge of all the kinds of books available. These users generally have trouble finding interesting books, or they are not aware of certain genres of books that may interest them. Therefore, subject matter experience would be rated as a journeyman.

The user's experience with relevant technology would be rated as master. Similar to the search bar in Amazon, Book Brief will return intuitive search results. Moreover, the similar books provided by the Gemini API will be interactive and clickable as cards. The used technology is standard in database search engines and online stores. Therefore, key users will have mastery of the technology.

6.2 Secondary Users

Secondary users will use Book Brief to browse the books available from the Google Books API. These users would be composed of an older demographic. Alternatively, this is indicative of people who would rather browse through the books themselves instead of following the Gemini similar books. Secondary users are likely to have more knowledge of books if they prefer to browse the catalog themselves. On the other hand, users could be less knowledgeable due to having read/remembered fewer books.

6.3 Unimportant Users

People who are not interested in reading books or are not interested in using technology to determine which books to read. Unimportant users are uninterested in reading books, which means their reading knowledge is limited. Technological skills could be anywhere from novice to master. It is unknown, or difficult to predict, how much technological experience an individual who reads a few books might have. This group could also include people who have intellectual disabilities or people who dropped out of school and are not interested in pursuing education. Alternatively, people who are simply not interested in books.

7 Assumptions and Dependencies

7.1 Assumptions

The following assumptions help set common expectations and provide context for decision-making.

- Users must know English to use the system, as the system will operate strictly in English.
- Users must desire to improve their knowledge of books, due to the nature of the system.
- Users must have moderate experience using web applications, as the system will appear and behave similarly to other common online services.
- Users must be capable of using an internet browser, as that is how the service is accessed.
- Users must have access to a standard internet browser for the same reason.

7.2 Dependencies

The technical dependencies include external libraries or frameworks which system components will use to function.

- Node.js
- Npm
- Express
- Axios
- Google Books API
- Google Gemini API

8 Specific Requirements

8.1 System Functional Requirements

UF-1: Users should be able to search for specific books they are interested in.

- SF-1.1: The system must provide a search bar that allows users to input keywords, titles, or author names to locate specific books.
- SF-1.2: The system must process search queries and display relevant results.

UF-2: Users can explore random books to discover new titles and gain more information for that book.

- SF-2.1: The system must include a feature to randomly suggest books when prompted by the user.
- SF-2.2: Each random book suggestion must include basic details, such as the title and author.

UF-3: Users can click on a book to generate a detailed summary.

- SF-3.1: The system must integrate with Gemini AI to generate detailed summaries for selected books.
- SF-3.2: The system should process and provide a detailed summary of the book for the user.

UF-4: Users can view a list of similar books related to the one they selected.

- SF-4.1: The system must analyze the selected book's genre and author to generate a list of similar books.
- SF-4.2: The system must retrieve and display the list of similar books based on the user's request.

UF-5: Users can navigate between pages.

- SF-5.1: The system must implement navigation that supports seamless transitions between pages without excessive load times or disruptions.

8.2 Software System Attributes

8.2.1 Usability

UNF-1: Users should experience minimal delays when interacting with the system, ensuring smooth and efficient usage.

- SNF-1.1: The system must process user interactions, such as clicks and searches, with a response time of under 1 second.

UNF-3: No more than 5 similar books should be generated and displayed.

- SNF-3.1: The recommendation algorithm must execute within 5 seconds to ensure results are displayed promptly and do not exceed the defined limit.

8.2.2 Performance

UNF-2: The system should generate summaries for users within a short time frame to ensure a smooth user experience.

- SNF-2.1: The system must generate book summaries within 5 seconds to ensure that users do not experience delays.

9 Appendices

9.1 Packaging and installation issues

1.) Locate prerequisites: Node.js, npm
2.) Acquire API keys: Google Books, Gemini
3.) Clone the project git repository
4.) Install dependencies
5.) Set up environment variables, containing API keys
6.) Start the server using npm start
7.) Access the application within the browser at the appropriate address

9.2 User Manual

Using the system to learn about particular books:

1.) Access the application within the browser at the appropriate address.

2.) Click on a card within the Discovery Queue, or enter a search query and press “enter”.
3.) This will return another horizontal queue of search results, which can also be clicked to open the summary page.
4.) From the summary page, a summary can be read as well as can the cover and title be seen.
5.) Below the summary on the summary page, similar books can be clicked on to bring up more summaries.

9.3 Open Issues

- Allowing for user-generated content including reviews to provide more insights
- Allowing for advanced search and search filtering
- Support for multiple alternative languages
- Partnering with bookstores to allow for purchasing books online

9.4 Lessons Learned

9.4.1 Design Patterns

One pattern used is Container and Presentation. The Container and presentation pattern is a pattern that aims to separate the presentation logic from the business logic in React, thereby making it modular, testable, and one that follows the separations of concern principle. When a user is navigated to the summary page for a book, a list of 5 similar books is displayed in a specific format where each book is displayed with the book cover image, title, and authors. The book information will be retrieved from an API. This pattern helps separate the code for retrieving book data from the book database API and displaying the information to the user. Consequently, errors from API calls and component rendering on the webpage can be easily identified. Therefore, testing is more manageable.

Another pattern used is Controlled Components. When a user searches for a book, the query will be used by an API to retrieve up to 20 book results, which will be displayed to the user. Here, the input field’s (search bar) value must be readily accessible and checked before sending the query to the LLM API. The Controlled Component pattern ensures predictability and synchronized state management. As a result of ensuring that there is only a single source that manages a component's properties, this pattern allows for a more predictable and controlled flow of data in the application.

9.4.3 Team Communications

Team communications were conducted over a Discord server and in person. An improvement that could be made is transitioning to a standard SMS group chat for enhanced communication.

9.4.4 Task Allocations

Tasks were allocated based on team member knowledge and availability. For improvement, it could be helpful to stick more rigorously to the planned distribution of tasks with less improvisation.

9.4.5 Desirable Changes

Trey: I would like to improve the discovery queue through a more advanced selection method, not just random selection. Additionally, adding the capability to post user-generated anonymous comments and reviews would significantly improve the quality of insights the application provides.

Dheeraj: If given the opportunity to work on the project for another month, I would be interested in vastly expanding and improving the search functionalities. This would include advanced filtering and a more structured display of results.

Nhat: I'd improve the user interface and overall user experience. A sleeker design, faster navigation, and clearer layout would ensure users can access features effortlessly. Additionally, I'd focus on optimizing performance to make the application more responsive and user-friendly, potentially opening up more support for mobile devices.

9.4.6 Challenges Faced

Trey: System implementation is probably the most difficult task. In my opinion this is because it involves translating abstract concepts into tangible code and features, and it can reveal infeasibility hidden by abstraction. Debugging, testing, and integrating different components while handling unintended behavior and unforeseen problems makes for a very dynamic portion of the project.

Dheeraj: In my opinion, requirements specification is the hardest task because it involves conceptualizing the foundation of the entire project. It is a unique challenge to take a vague idea and transform it into specific requirements, and it is very easy to improperly draft your requirements such that the desired functionality is not actually reached. Further, understanding user needs and desires is a major part of this step, which can be very unclear.

Nhat: System design is the most difficult task. This is because it is when the most important, core determinations are made. It requires accurately assessing needed design patterns and how they apply to your specific use case. Choosing the wrong pattern, or not applying a pattern correctly can lead to many much bigger problems down the road, and for this reason, it is simultaneously the most difficult and the most important.