Software Architecture Document

Software Name: IGottaBook

Purpose

IGottaBook aims to transform the way readers discover and engage with books. This platform leverages a comprehensive database to provide easy access to a vast selection of books, enhanced by personalized recommendations tailored to each user's preferences and reading history. The software's primary goal is to simplify the book discovery process, making it more engaging and accessible, thus enriching the overall reading experience.

Target Audience

- The target audience for ***IGottaBook*** encompasses a wide range of readers, including avid bibliophiles, casual readers, students, and members of book clubs. The application is designed to cater to anyone with an interest in reading, offering features that accommodate both leisurely browsing and academic research. By providing tools that facilitate easier access to literature and personalized content, ***IGottaBook*** seeks to support and enhance the reading habits of its diverse user base.

Requirements

- **Functional Requirements:**

Req ID	Requirement Description	Rationale
FR01	User Registration and Profile Management	Allows users to create, edit, and manage their profiles securely
FR02	Advanced Book Search	Users can search for books using multiple filters such as title, author, and publication year
FR03	Book Recommendations	Automatically suggest books based on the user's past activities and preferences
FR04	Favorites and Reading Lists	Users can create and manage lists of favorite books and planned reading.
FR05	Review Submission	Users can post reviews for books, which helps in community building and enhances content credibility.

- **Non-Functional Requirements:**

Req ID	Requirement Description	Rationale
NFR01	System Performance	The application should load responses within 2 seconds to maintain user engagement and efficiency, suitable for our book search functionalities
NFR02	Scalability	The system should handle up to 20,000 users at the same time to ensure reliability during high-demand periods
NFR03	Security - Data Encryption	All user data must be encrypted to protect privacy and comply with data protection laws
NFR04	Accessibility Compliance	The application should be accessible according to ADA standards to ensure usability for all users
NFR05	User Interface Responsiveness	The application interface must be intuitive and responsive, ensuring ease of use with minimal learning curve

Scenario Viewpoint

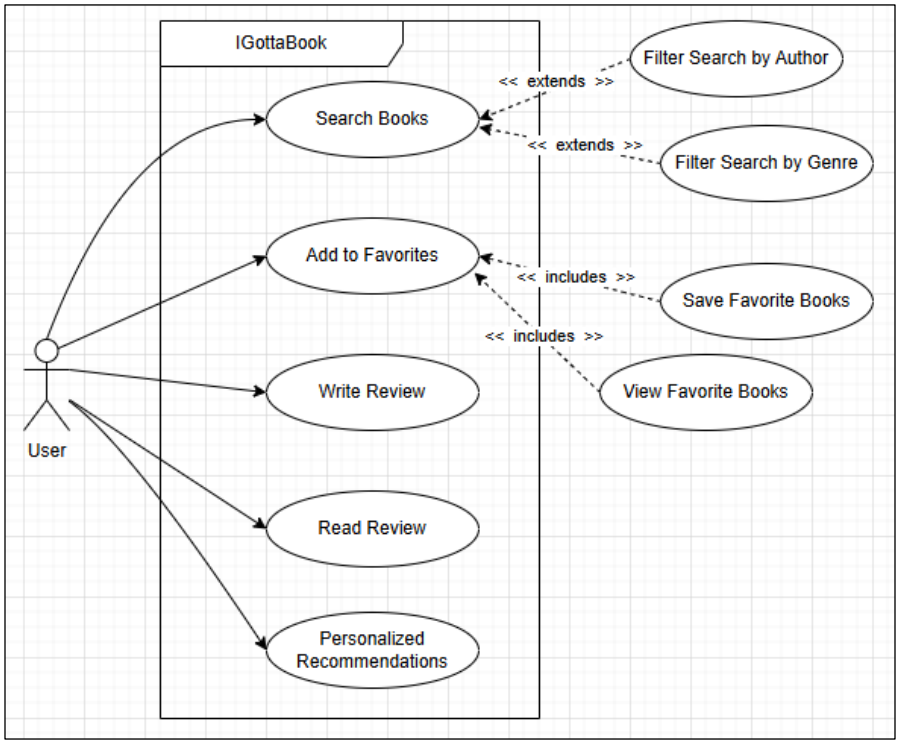

Figure 1 - Use Case Diagram (Scenario View)

Concerns: Understanding the main functions that IGottaBook provides to users.
Stakeholders: All stakeholders, with a focus on end users.
Modelling techniques: UML Use Case Diagram

- **Search for Books**
  - Users can look up books in the database by typing things like the book's title, author's name, or ISBN. This makes it easy for users to quickly find books they are interested in, improving how they interact with the app.
- **Add to Favorites**
  - Users can save books they like to a special list called 'Favorites'. This feature helps users keep track of books they enjoy and access them quickly later, making their reading experience more personal and organized.
- **Write Reviews**
  - Users can write their thoughts and opinions about the books they have read and share these reviews with others. This feature helps build a community feeling, allows users to express themselves, and helps other readers make better choices about what to read.
- **Read Reviews**
  - Users can read what others have said about books before deciding to read them. This feature is helpful because it lets users learn from others' experiences, helping them decide if a book is right for them.

- **Personalized Recommendations**
  - The app suggests books that users might like based on what they have searched for and read before. This personalized recommendation makes finding new books that match a user's interests much easier and makes the app more enjoyable to use.

The primary user actions in ***IGottaBook*** are to search for books and manage favorite books. When searching, users can apply optional filters like author or genre. Users can also add books to their favorites, write reviews, read reviews, and receive personalized book recommendations. This use case diagram shows these key actions to help make finding and enjoying books easier. It ensures developers focus on what users need most from ***IGottaBook***, which is a smooth and engaging way to access and interact with books. By clearly presenting these user interactions, the diagram helps keep the development aligned with ***IGottaBook*** goal of making reading more accessible and enjoyable for everyone involved.

Physical Viewpoint

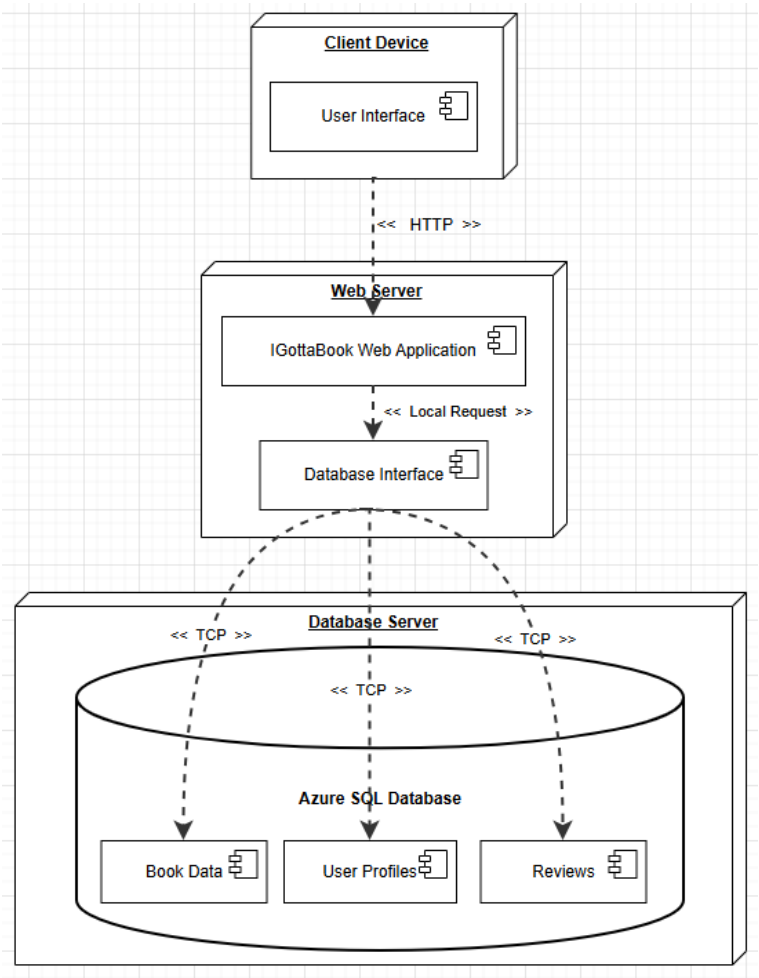

Figure 2 - Deployment Diagram (Physical View)

Concerns: Mapping of software to hardware, communication protocols, and modules related to data management and user interactions.
Stakeholders: Software architect, software developers
Modelling techniques: UML Deployment Diagram

- **User Interface (Client Device)**
  - The User Interface is what people use to interact with ***IGottaBook*** on their personal devices like computers or smartphones. It's designed to be simple and user-friendly, allowing users to easily search for books, read and write reviews, and manage their favorite lists. When a user performs any action, such as searching for a book, this interface sends the details to the web application to handle the request, making sure users have a seamless and responsive experience.
- **IGottaBook Web Application (Web Server)**
  - This is the main software component that runs on a web server. It takes care of all the operations behind the scenes. When the User Interface sends a request, like a book search, this application processes it by figuring out what information is needed and then asking the Database Interface to

fetch this information from the database. It ensures that all user requests are handled efficiently, maintaining the performance and reliability of ***IGottaBook***.

- **Database Interface (Web Server)**
  - Positioned on the web server, the Database Interface acts as a middleman between the web application and the actual database. It receives data requests from the web application, such as retrieving a user's favorite books or fetching new book recommendations, and translates these into actions the database can perform. This component is crucial for organizing and securing the data exchange, ensuring data accuracy and timely access.
- **Azure SQL Database (Database Server)**
  - Stored on a specialized database server, this powerful database holds all the data ***IGottaBook*** needs to function. It includes detailed records of books, user profiles, and reviews. The Database Interface communicates with it to retrieve or update data as users interact with the application, supporting a vast amount of information and multiple user requests simultaneously. This ensures that ***IGottaBook*** can serve many users at once without slowing down, keeping the application fast and reliable.

The ***IGottaBook*** web application connects to the Azure SQL Database over a secure TCP connection to store and retrieve book, user, and review data. Hosted on Microsoft Azure's cloud infrastructure, the database provides scalable and reliable storage. The user interface, accessed on client devices, communicates with the ***IGottaBook*** Web Application on the web server over HTTP. This setup allows users to search for books, add favorites, write and read reviews, and receive personalized recommendations. The UML Deployment Diagram for ***IGottaBook*** is essential because it visually shows how the software components are set up across different devices and servers, and how they interact. It illustrates how requests from users travel through the system and how data is managed, helping developers and engineers understand the architecture and their roles in maintaining it. This clarity is crucial for ensuring the application runs smoothly and efficiently, providing a good experience to users. It also supports future growth, making it easier to scale ***IGottaBook*** as more users join or more data is added.

Development Viewpoint

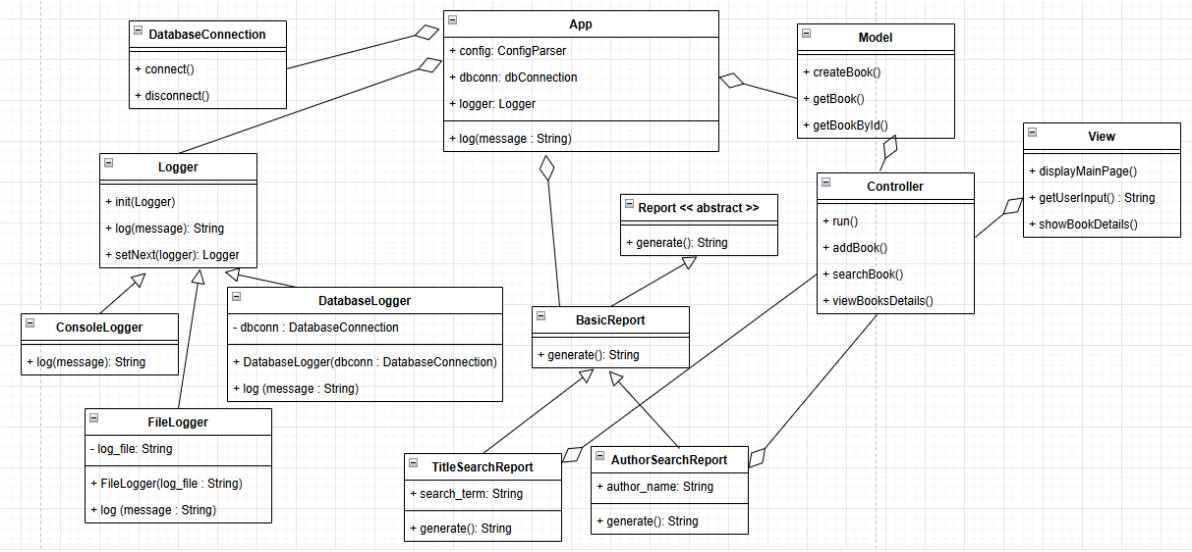

Figure 3 - Class Diagram (Development View)

Concerns: Organization of software modules, efficient handling of user interactions, and robust data management.

Stakeholders: Software developers, project managers

Modelling techniques: UML Class Diagram

- **App (Singleton Pattern)**
  - The App class acts as the central singleton of ***IGottaBook***, handling essential configuration, database connections, and logging setup. It ensures that all resources are consistently managed throughout the application, allowing only one instance to exist. This singleton approach reduces redundant configurations and helps maintain a controlled application state.
- **Logger (Chain of Responsibility Pattern)**
  - Logging is managed by the Logger class, which implements the Chain of Responsibility pattern. Subclasses such as *ConsoleLogger*, *FileLogger*, and *DatabaseLogger* extend the Logger class to handle specific logging tasks, each taking on logging responsibilities as needed (console, file, or database logging). Each subclass can pass log messages to the next in the chain using the *setNext(logger)* method, which allows for flexible and extensible logging management.
- **Report (Decorator Pattern)**
  - The Report class serves as an abstract base for generating customizable reports, and it works as the core of the Decorator Pattern in the ***IGottaBook***
  - application. *BasicReport* provides a standard implementation, while decorators like *TitleSearchReport* and *AuthorSearchReport* extend this base to allow reports to be filtered by title or author. This design enables users to create specific reports based on their needs, adding flexibility and reuse to the reporting functionality.
- **Model**
  - The Model class is dedicated to data management, encapsulating CRUD operations for books, reviews, and user information. It provides methods such as *createBook()*, *getBooks()*, and *getBookById()*, which

allow the Controller to interact with the database in a structured, consistent way. This encapsulation maintains data integrity and abstracts away database details, making data management more modular and accessible.

- **View**
  - The View class is responsible for rendering the user interface elements and capturing user input. Methods such as *displayMainPage()* and *showBookDetails()* help present information, while *getUserInput()* gathers data or commands from users. Operating without state, the View efficiently supports UI updates and adjustments, enhancing flexibility and user experience.
- **Controller (MVC Pattern)**
  - The Controller class plays a critical role in bridging the Model and View classes, aligning with the Model-View-Controller (MVC) pattern. It processes user actions, interacts with the Model to retrieve or modify data, and updates the View accordingly. Key methods in Controller include *run()*, *addBook()*, *searchBooks()*, and *viewBookDetails()*, ensuring that user interactions are effectively managed. By coordinating between Model and View, the Controller maintains a clear separation of concerns, keeping each component modular and focused on its responsibilities.

This Development View is crucial for understanding how ***IGottaBook***'s components are organized, interact, and are enhanced by design patterns. The diagram provides developers and architects with a clear structure, showing the responsibilities and relationships of each component. By following established patterns like Singleton, Chain of Responsibility, Decorator, and MVC, the architecture ensures modularity, flexibility, and ease of maintenance. This clarity is essential for scaling and extending ***IGottaBook*** as it grows in functionality and user base.

Design Pattern Summary

Pattern	Role	Classes
Singleton	Manages centralized application state, including configurations, database connection, and logger	App
Chain of Responsibility	Handles logging across multiple outputs, such as console, file, and database	Logger, ConsoleLogger, FileLogger, DatabaseLogger
Decorator	Allows customization of report generation by adding search filters like title and author	Report, BasicReport, TitleSearchReport, AuthorSearchReport
Model-View-Controller (MVC)	Manages user interactions by separating application logic, data management, and UI rendering	Model, View, Controller

Logical Viewpoint

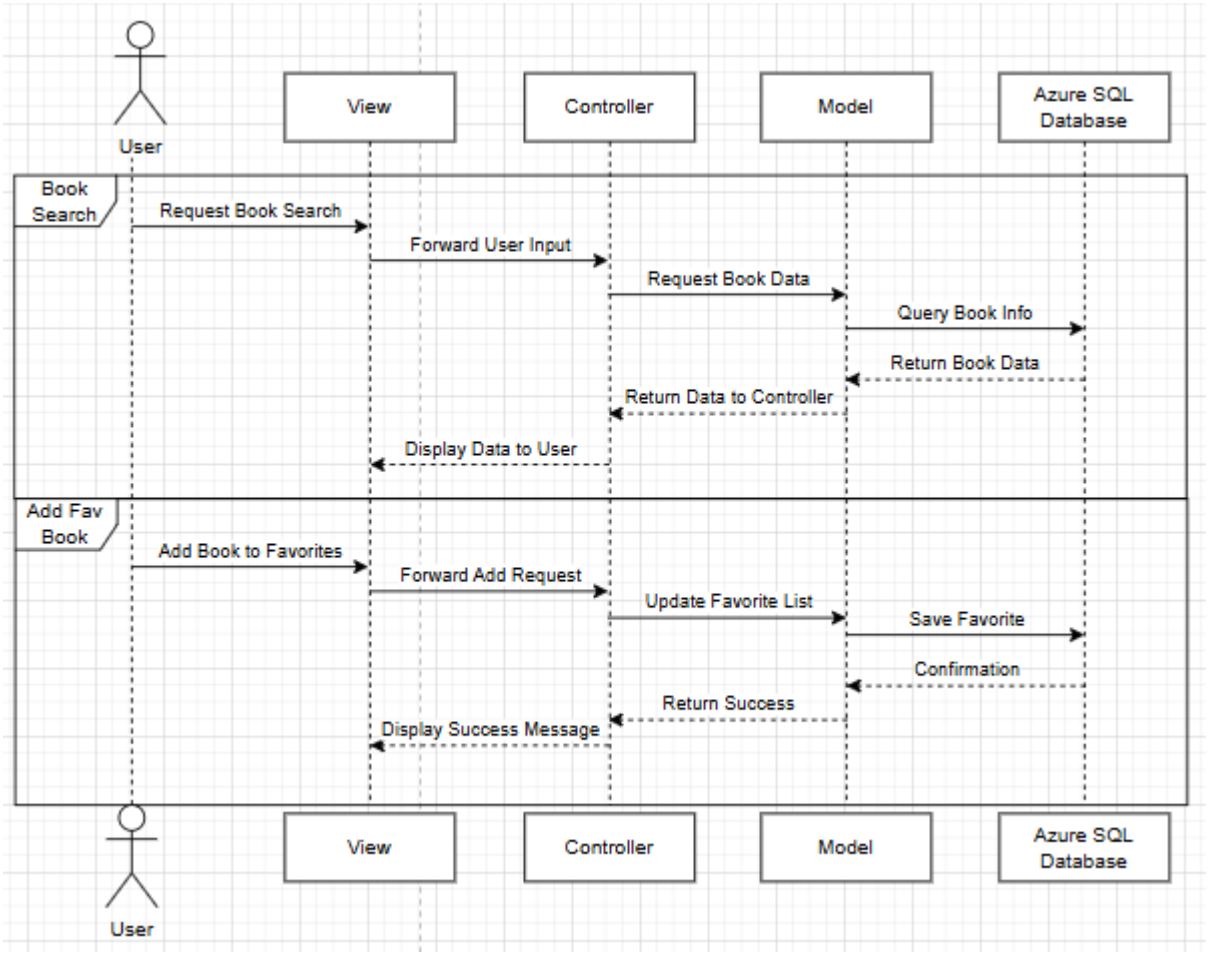

Figure 4 - Sequence Diagram (Logical View)

Concerns: User interaction flow, efficient data retrieval, and clear module communication
Stakeholders: Software developers, project managers
Modelling techniques: UML Sequence Diagram

- **Search for Book**
  - The user initiates a book search through the User Interface. The Controller handles this request by calling the Search Service, which retrieves book data through the Database Interface. This data is fetched from the Azure SQL Database and returned up the chain, ultimately displaying the results to the user. This interaction highlights how ***IGottaBook*** efficiently retrieves and presents search data in response to user queries.
- **View Book Details**
  - When the user selects a book from the search results, the UI requests detailed book information from the Controller. The Controller uses the Database Interface to fetch the necessary details from the database, ensuring that users can access in-depth information about a selected book. This step emphasizes the system's logical handling of data retrieval and display for specific items.
- **Add to Favorites**
  - Users can add a book to their favorites, starting with a request from the UI. The Controller forwards

this request to the Favorites Service, which processes the addition through the Database Interface and stores the favorite book in the Azure SQL Database. The flow ends with a confirmation to the user, showcasing ***IGottaBook***'s ability to manage user preferences and save them securely.

The Logical Viewpoint, shown through the Sequence Diagram, is essential for understanding how ***IGottaBook***'s components interact during runtime. It highlights how different parts of the system, like the User Interface, Controller, Service layers, and Database Interface, work together step-by-step to respond to user requests. By illustrating the data and control flow across these interactions, the diagram makes it clear how the application handles tasks such as searching for books, viewing details, and adding favorites. This logical flow helps stakeholders understand how data moves through the system, ensuring that responses are accurate and consistent. The Sequence Diagram also shows the runtime structure, helping developers see how each user action is processed in an organized way, creating a smooth and reliable experience. This view is crucial for planning and maintaining ***IGottaBook***'s internal logic, ensuring that each component contributes to a user-friendly and dependable application.

Process Viewpoint

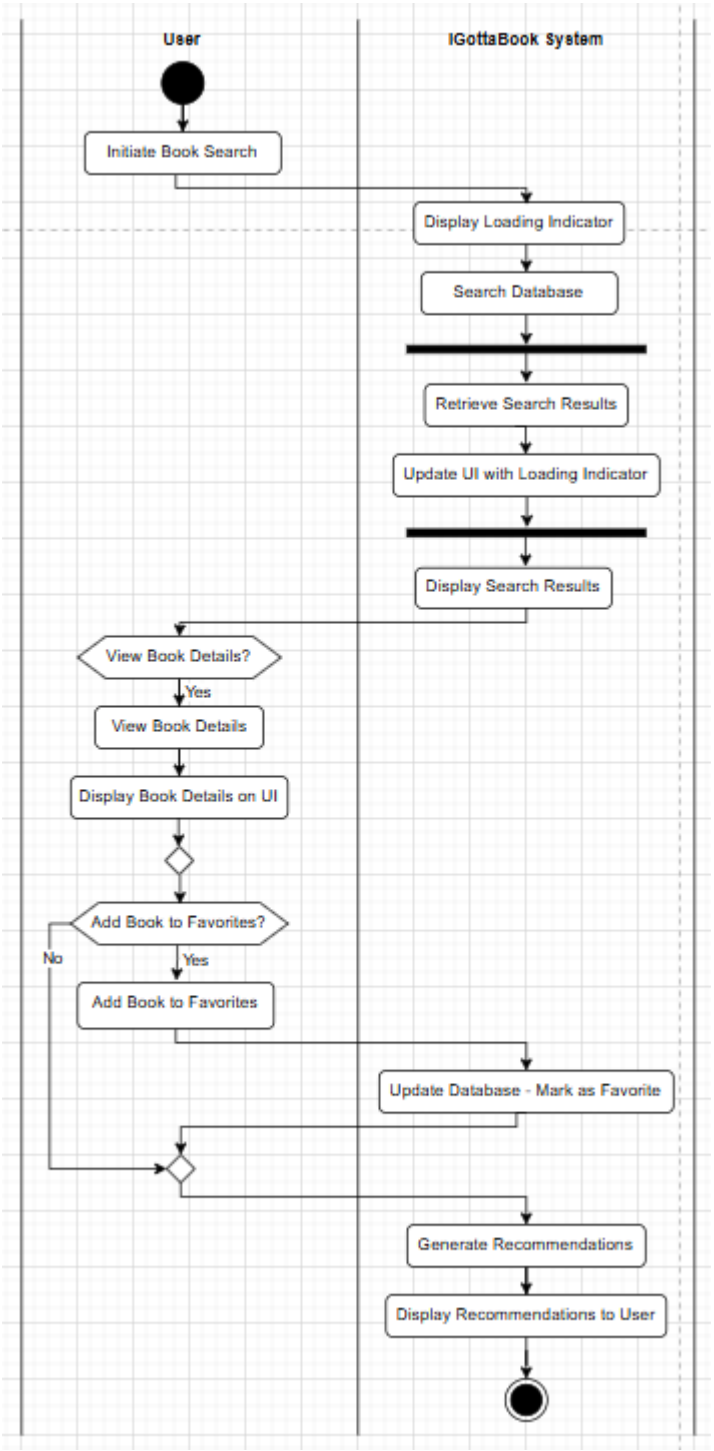

Figure 5- Activity Diagram (Process View)

Concerns: Concurrency, task synchronization, parallel processing, and workflow management

Stakeholders: Software developers, system architects, project managers

Modelling techniques: UML Activity Diagram

- **User Initiates Book Search**
  - When the user starts a book search by typing a query, ***IGottaBook*** searches the database for relevant

books and shows a loading indicator on the screen to keep the user informed.

- **Display Search Results**
  - After finding results, ***IGottaBook*** displays the matching books. The user can view more details about a book or add it to their favorites list. If added to favorites, the database is updated so the user can easily find the book again.
- **View Book Details**
  - If the user selects a specific book, details are fetched from the database and displayed on the UI.
- **Add Book to Favorites**
  - The user can add the selected book to their favorites list. This action updates the database, marking the book as a favorite for the user.
- **Generate Book Recommendations**
  - Once the user has marked favorites, ***IGottaBook***'s recommendation engine suggests similar books based on their preferences. These recommendations are displayed, helping the user explore new books they might like.
- **End Process**
  - The activity flow completes after recommendations are displayed, and the user can continue exploring more features.

The Activity Diagram will illustrate the flow of activities when a user performs key actions in IGottaBook, including searching for a book, adding it to favorites, and viewing recommendations. This diagram provides insights into task coordination and decision points in the application, showing how concurrent tasks are managed and synchronized.