BOOK TRACKER

A MINI-PROJECT BY:

Pranaav kumar.J-230701233
Praveen somasundaram-230701246

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project **"BOOK TRACKER"** is the bonafide work of **"PRANAAV KUMAR.J , PRAVEEN SOMASUNDARAM"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Ms. ASWANA LAL
Asst. Professor
Computer Science and Engineering,
Rajalakshmi Engineering College (Autonomous),
Thandalam,Chennai-602105

SIGNATURE

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Book Tracker Project is a user-friendly, web-based application designed to help readers manage and organize their personal book collections effortlessly. Built using Java for robust backend functionality and MongoDB for scalable, flexible storage, the system offers features such as adding, updating, finding, and deleting books. With its intuitive interface, users can curate and maintain a digital library that reflects their reading journey, allowing them to access detailed book information, such as author, title, and edition, from anywhere and on any device. The project bridges the gap between traditional book collection methods and modern digital convenience, catering to bibliophiles and casual readers alike.

Key modules include user management, which ensures secure access and personalized experiences, and schedule management, which allows users to set reading goals and track their progress effectively. Advanced analytics and reporting features provide insights into reading habits, collection trends, and overall progress, enhancing user engagement and promoting consistent reading habits. With visual dashboards and customizable reports, users can monitor their library growth and make informed decisions about future additions to their collection.

The Book Tracker Project emphasizes scalability, accessibility, and a seamless user experience. By leveraging MongoDB's document-oriented design and Java’s robust integration, the system accommodates dynamic and evolving user needs. This innovative tool not only simplifies book management but also fosters a deeper connection between users and their literary interests, making it an indispensable resource for anyone looking to maintain an organized, accessible, and engaging digital library.

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

Discover the Ultimate Online Book Tracker

Welcome to a seamless way to manage your literary world! This Book Tracker website is a modern, efficient, and user-friendly platform designed to simplify how you organize and store your collection of books. With its intuitive design and robust backend, it acts as your personal online library, giving you a convenient space to maintain your reading journey.

Built with **Java** and powered by **MongoDB**, this platform combines reliable performance with dynamic storage capabilities, ensuring your library evolves as your interests grow.
Whether you're curating a collection of classics, tracking your favorite series, or creating a personal archive, this tool adapts to your needs effortlessly.

Explore the endless possibilities of having your library at your fingertips and embark on a streamlined reading experience like never before.

PURPOSE OF THE PROJECT

The primary purpose of this project is to create an innovative, user-friendly digital platform that redefines how individuals manage their book collections. As reading habits evolve in the digital age, the need for a comprehensive, accessible, and personalized system for organizing books has grown significantly. This Book Tracker addresses that need by providing a seamless online solution for storing and maintaining literary records, helping users keep track of their reading journey in a structured and efficient way.

By leveraging the capabilities of **Java** for robust backend functionality and **MongoDB** for scalable, secure storage, the project aims to deliver a reliable tool capable of handling a diverse range of book collections, from small personal libraries to extensive archives. Beyond simple storage, this tracker is envisioned as a resource that empowers users to interact with their collections in meaningful ways, ensuring they remain engaged with their reading goals and interests.

Ultimately, this project aspires to bridge the gap between traditional book collection methods and modern digital convenience, fostering a deeper appreciation for literature while offering an easy-to-use, tech-driven solution for readers in the 21st century.

ADD A BOOK

The "Add a Book" feature is the heart of this Book Tracker, designed to make organizing your library effortless and enjoyable. With this function, users can seamlessly add books to their collection by providing essential details such as the title, author, and edition. Whether it's a classic masterpiece, the latest bestseller, or a cherished personal favorite, this feature ensures every book finds its rightful place in your digital library. The clean and intuitive interface makes the process quick and hassle-free, while the inclusion of precise details like editions allows you to maintain an accurate and organized record. With just a few clicks, you can watch your library grow, one book at a time, creating a personalized archive that reflects your reading journey.

FIND A BOOK

The "Find a Book" feature transforms the way you navigate your library, offering a fast and efficient way to locate any book in your collection. Whether you're searching for a specific title, an author, or a particular edition, this function allows you to retrieve the exact information you need with ease. Powered by a smart search system, it ensures accurate results even as your collection expands. No more flipping through pages or scrolling endlessly— simply input your query, and the "Find a Book" feature will bring the desired book to your fingertips in seconds. It's like having a personal librarian for your online library, making your reading experience more organized and enjoyable.

UPDATE A BOOK

The "Update a Book" feature adds a layer of flexibility and precision to your digital library, allowing you to keep your collection up to date effortlessly. Whether you've discovered a typo, want to change the edition, or update the author's details, this function ensures that your library always reflects the most accurate information. With a user-friendly interface, modifying any book’s details is a breeze—simply locate the book, make the necessary changes, and save. This feature is perfect for those who value organization and accuracy, giving you full control to maintain your library as a dynamic, ever-evolving resource that adapts to your reading journey.

DELETE A BOOK

The "Delete a Book" feature empowers you to keep your library clean and relevant by effortlessly removing books you no longer need. Whether you’ve finished reading, no longer require the reference, or simply want to declutter your collection, this function provides a quick and hassle-free solution. With just a few clicks, you can locate the book by its title, author, or edition and permanently remove it from your library. This ensures that your collection remains organized and meaningful, reflecting only the books that matter most to you. It's the perfect tool for maintaining a streamlined and personalized online library experience.

AVAILABLE BOOK

The "Available Books" feature serves as your gateway to exploring the full scope of your digital library. With this function, you can view all the books you've added, along with their detailed information such as title, author, and edition, in one organized and visually appealing display. It’s like having a bookshelf at your fingertips, where every book is neatly categorized and easily accessible. Whether you're browsing for your next read or revisiting a favorite title, this feature provides a clear and comprehensive overview of your collection, making your online library both functional and delightful to explore.

SIGNIFICANCE AND BENEFITS

The Book Tracker website, with its rich set of features, offers significant advantages for readers, collectors, and anyone looking to stay organized in their literary pursuits. Its ability to seamlessly manage a personal book collection—by adding, updating, finding, deleting, and viewing books with detailed information—provides users with unparalleled control and convenience.

1. **Enhanced Organization**: With the ability to easily add, update, and organize books, users can maintain a tidy and well-structured library. No more lost titles or confusion about editions—everything is stored systematically, ensuring that books are always accessible when needed.

2. **Time-Saving**: The intuitive "Find a Book" and "Available Books" functions allow users to quickly search through their collection, saving time that would otherwise be spent flipping through physical pages or hunting through disorganized records.

3. **Personalized Experience**: The tracker adapts to each user's unique collection, enabling them to record specific details such as the author and edition of each book. This personalization ensures that the library truly reflects the individual’s reading journey and preferences.

4. **Flexibility and Control**: The "Update a Book" and "Delete a Book" features offer the flexibility to modify or remove entries as needed. This empowers users to maintain an up-to-date, accurate catalog that aligns with their evolving collection, whether they’re adding new titles or removing outdated ones.

5. **Scalability**: As your library grows, the Book Tracker can scale with you. Built on **MongoDB**, it ensures that no matter how large your collection becomes, the platform can handle it efficiently, making it an ideal solution for readers of all types—from casual readers with a small collection to serious bibliophiles with thousands of books.

6. **Improved Reading Experience**: By organizing and tracking books digitally, users can stay engaged with their reading habits. They can track progress, revisit old favorites, and discover gaps in their collection, enhancing the overall reading experience.

7. **Accessible Anywhere**: Being a web-based application, the Book Tracker is accessible from any device, anytime, anywhere. This makes it easy for users to manage their library on the go, whether they're at home, at work, or traveling.

In summary, the Book Tracker not only simplifies the process of managing a book collection but also enhances the overall reading experience by offering a powerful, scalable, and flexible tool. It's an indispensable resource for anyone looking to stay organized, discover new titles, and maintain a well-curated library.

1.2 OBJECTIVE

The objective of the Book Tracker website is to provide the public with an intuitive and efficient platform for managing personal book collections. Users will benefit from the ability to easily add books, including key details like title, author, and edition, allowing them to organize and track their reading journey in a way that suits their individual preferences.

The website offers an easy-to-use search function, enabling users to quickly locate books by title, author, or edition. This eliminates the frustration of sorting through large collections and ensures that users can find their books with minimal effort.

As users' libraries grow, the "Update a Book" feature allows them to maintain accurate and current records, ensuring that every detail is up to date. Additionally, the "Delete a Book" function provides the flexibility to remove outdated or irrelevant entries, keeping the collection clean and organized.

Overall, the Book Tracker website serves the public by offering a personalized, scalable, and accessible solution for book management. It empowers users to maintain a well-organized library, enhances their reading experience, and allows them to stay connected to their literary interests with ease—anytime and anywhere.

I. TO PROVIDE BOOK INFORMATION

The Book Tracker website is designed to help users maintain precise and up-to-date records of their book collections. By ensuring that every book is entered with the correct details, including title, author, and edition, users can confidently rely on the platform to keep their literary information accurate and organized. This feature guarantees that no book is misplaced or misrepresented, providing a trustworthy resource for managing a personal library.

VIEW BOOK WITH AUTHOR NAME

One of the core benefits of the Book Tracker is the ability for users to view their books alongside their respective authors. By storing detailed author information, the platform enables easy identification and retrieval of books based on the author’s name. Whether users are looking for works from a single author or tracking multiple titles by different writers, this feature helps them quickly locate specific books in their collection, ensuring that the author’s details are always accurate and visible.

VIEW BOOK WITH EDITION

In addition to tracking authors, the Book Tracker allows users to view their books by edition. This feature is especially useful for those who collect different editions or formats of the same book, whether it’s a first edition, a special hardcover, or an e-book version. By providing a clear view of each book’s edition, users can ensure that they have the correct version in their library, helping them manage and organize their collection with precision.

II. FIND THE BOOK USING AUTHOR

Imagine being able to instantly dive into the complete works of your favorite author with just a click! The "Find the Book Using Author" feature makes this possible, offering readers a seamless and exciting way to discover, track, and explore books by a specific author. Whether you're a passionate fan eager to uncover every novel, short story, or poem by your literary hero, or someone looking to immerse yourself in the rich world of an author's creations, this function is your gateway to a curated literary adventure!

For dedicated readers, scholars, or students, this tool is a game-changer. It allows you to effortlessly gather all works by a particular author, making research and reading easier and faster. You'll never have to waste time searching through endless lists or library shelves again—just type in the author's name, and voilà, everything you need is right at your fingertips!

Whether you're expanding your collection or revisiting the classics of a beloved author, the "Find the Book Using Author" feature brings joy, speed, and organization to your literary journey. It's the ultimate tool for anyone who craves a deeper connection with their favorite writers and seeks to enjoy their works in a whole new way!

III. UPDATE THE BOOK

The "Update the Details of the Book" feature is a powerful tool that gives readers complete control over their library's accuracy and organization. Mistakes happen—whether it's a typo in the author's name, incorrect edition information, or a missing detail like the publication year. This feature allows users to easily correct these inaccuracies, ensuring that their digital library remains a true reflection of their book collection. But its usefulness goes beyond just fixing errors—it provides a wealth of benefits for readers looking to keep their collection in perfect order.

1. Keep Your Library Accurate and Up-to-Date

Readers can update any incorrect information about the books they've added, such as the author's name, edition, or publication date. This ensures that every book in their collection is accurately documented, making the tracker a reliable resource for their literary journey.

2. Track Multiple Editions of a Book

For readers who collect different editions of the same book—whether it's a first edition, limited edition, or a special release—the ability to update edition details becomes essential. By keeping track of these changes, the Book Tracker helps readers differentiate between versions and manage their collection with precision.

3. Reflect Changes in Reading Preferences

Over time, a reader's interests may evolve, or they may want to re-categorize books. The

update feature allows users to change categories, genres, or other details to better reflect their current preferences, making it easier to organize books based on new reading habits.

4. Correct Mistakes from the Data Entry

We all make mistakes while entering details, and sometimes information can be entered incorrectly. Whether it's a missing comma in the author's name or a book's title that's slightly off, this feature gives readers the ability to fix any mistakes without having to delete and re- enter the entire book. This saves time and keeps the collection consistent.

5. Personalize the Book Entry

Sometimes, readers may want to add personal notes or specific details about a book—perhaps a particular edition they own or a unique cover. With the update feature, users can customize book details to include personal information, making their collection even more meaningful.

6. Adapt to New Discoveries

Over time, readers may uncover additional information about a book—whether it's discovering a new edition, an author's biography, or an extra chapter. The update function allows readers to add this newfound information to their entries, ensuring their collection grows along with their knowledge.

7. Maintain an Evolving Digital Library

As books are read, given away, or sold, the ability to update book details allows users to maintain an evolving collection. This ensures that no matter what changes in a reader's collection, the tracker always mirrors what's currently in their library.

Overall, the "Update the Details of the Book" feature empowers readers to keep their library accurate, organized, and personalized, making it an invaluable tool for anyone looking to manage their book collection with precision and ease.

IV. DELETE THE DETAILS OF BOOK

The "Delete the Details of the Book" feature is an essential tool for readers who want to maintain a clean and organized digital library. Sometimes books need to be removed from a collection for various reasons—whether they've been given away, sold, or simply no longer hold personal value. This feature not only helps readers declutter their library but also ensures that their collection remains up-to-date and relevant to their current reading interests. Here are several ways this feature can be used and why it's so useful:

1. Remove Books No Longer Needed

As your library evolves, you may find that certain books no longer fit your reading habits or interests. The ability to delete books ensures that your digital collection stays focused on what matters most to you, whether it's books you plan to reread, reference, or collect.

2. Declutter Your Collection

Over time, as more books are added, your collection may become overwhelming. Deleting books that are no longer part of your reading journey helps you keep a streamlined library. This decluttering makes it easier to locate and enjoy the books that are truly important to you.

3. Correct Duplicate Entries

Sometimes, during data entry, a book may get added more than once, either by mistake or in different formats (hardcover, paperback, e-book). The delete feature allows readers to remove duplicate entries, ensuring that the collection remains accurate and free from redundancy.

4. Remove Unwanted Books After Gifts or Purchases

When given books as gifts or purchasing books on impulse, you may decide later that they're not a good fit for your collection. The "Delete the Details of the Book" feature lets you easily remove those titles, ensuring your library reflects only the books that you truly want to keep.

5. Update Your Collection as You Read

As you progress through your reading journey, certain books may no longer be relevant to your current tastes, or you may have finished a particular series. Deleting books that no longer hold value allows you to make room for new books and keep your collection fresh.

6. Simplify Your Search Process

By deleting books that are no longer part of your collection, the search function becomes more efficient. You won't need to sift through outdated or irrelevant titles when looking for the books that matter most to you, making your library more streamlined and user-friendly.

7. Manage Space for New Additions

As your book collection grows, it's important to make space for new titles. The ability to delete books that are no longer needed or wanted helps you manage your library's size, ensuring that every new book added is purposeful and relevant to your current interests.

8. Maintain an Accurate Record of Gifts or Loans

If you loan a book to someone or decide to give one away, you can delete its entry to keep your collection accurate and up to date. This ensures that your tracker reflects your current possession and prevents any confusion over books you no longer own.

9. Adapt Your Library to Changing Preferences

Over time, a reader's taste in books may change. Books that once seemed interesting may no longer appeal. The delete function allows readers to adapt their collection to reflect evolving preferences, ensuring their library accurately represents their current interests.

10. Easily Remove Out-of-Print or Unwanted Editions

If you come across a book that's out of print or you no longer want to track specific editions, the delete feature lets you remove such entries, ensuring that your collection stays relevant and manageable.

Overall, the "Delete the Details of the Book" feature offers flexibility, organization, and control, making it an invaluable tool for readers who want to keep their digital library up to date, personalized, and free from clutter. Whether you're refining your collection or simply making space for new books, this feature empowers you to shape your library exactly the way you want it.

V. AVAILABLE BOOK FOR READER VIEW

Available Books

The "Available Books" feature is an essential tool that allows readers to view their entire collection at a glance, providing a clear and organized view of all stored books. By displaying essential information such as the book's title, author, and edition, this feature enhances transparency and makes it incredibly easy for users to find and explore the books they own. Here are several ways this feature can be used and why it's so beneficial for readers:

1. Easy Access to Your Entire Collection

The "Available Books" feature gives readers immediate access to their entire library, displaying all the books they've stored in one central location. This eliminates the need to search through different categories or folders, providing a quick overview of everything they own.

2. Efficient Book Discovery

With the ability to view books by title, author, and edition, this feature enables users to efficiently browse and discover specific books in their collection. Whether you're trying to locate a book by a favorite author or searching for a particular edition, it simplifies the process of finding what you need without hassle.

3. Manage Multiple Editions of Books

For avid readers and collectors who enjoy owning multiple editions of the same book—be it hardcover, paperback, or a special edition—the "Available Books" feature lets users see every version clearly. It helps differentiate between different editions, making it easy to manage and track which versions are in their collection.

4. Quickly Identify Missing or Unfinished Books

By listing all the books in one place, users can quickly spot any gaps in their collection. Whether it's books they need to finish reading or titles they still wish to acquire, this feature offers a transparent view of their library, helping them stay on top of their reading goals and

desires.

5. Track Books by Author

With the author information displayed alongside each book, readers can easily track which authors they've collected works from. It's a great way to explore more titles by favorite authors, ensuring that no work from an admired writer is overlooked.

6. Streamline Book Recommendations

For users who enjoy getting book recommendations based on their collection, having an organized view of all available books, including their editions and authors, can serve as a great starting point. This transparency allows readers to evaluate their collection and make informed decisions about which books they might want to add next.

7. Simplify Book Reviews and Sharing

If a reader wants to share their collection or review certain books, having all the book details displayed makes this process seamless. They can quickly pull up a list of books along with their authors and editions, streamlining the process of sharing their thoughts with others or seeking out similar titles.

8. Visualize Your Collection's Growth

As users continue to add books over time, the "Available Books" feature allows them to visually track the growth of their collection. This provides a sense of accomplishment and helps readers see just how much their library has expanded, motivating them to continue growing their collection.

9. Effortless Sorting and Filtering

For users with large collections, this feature makes it easy to sort and filter books by title, author, or edition. Whether you want to organize your collection alphabetically or by genre, this feature offers a fast and efficient way to navigate and manage a vast library.

10. Enhanced Collection Organization

The "Available Books" feature provides a comprehensive, organized structure for readers who value a well-maintained library. Whether it's for personal use, academic reference, or casual browsing, having all your books displayed with their key details ensures that every book is easily accessible and well-organized.

11. Track Books for Future Reading

For readers who use the tracker to plan their reading list, the "Available Books" section allows them to mark and easily track books they still want to read. It helps users create a visual list of upcoming titles, making it easier to prioritize what to read next.

Overall, the "Available Books" feature is a powerful tool that offers transparency, ease of use, and a user-friendly experience. It's perfect for anyone looking to keep their library organized,

find books quickly, track multiple editions, and stay on top of their reading collection.

DATABASE

Detailed Report on MongoDB for the Book Tracker App

MongoDB is a NoSQL database that is widely known for its efficiency, flexibility, and scalability. It is particularly effective for handling large volumes of unstructured or semi-structured data, making it an ideal choice for applications such as a Book Tracker App. MongoDB's document-oriented architecture stores data in BSON (Binary JSON) format, which allows for highly flexible and efficient data storage, especially when the structure of the data is subject to change, as is common in a dynamic application like a book tracker.

How Efficient is MongoDB for the Book Tracker App?

1. **Scalability and Performance**
   MongoDB is designed to scale out horizontally, meaning that as the Book Tracker app grows, MongoDB can handle increased data and traffic by simply adding more servers. This horizontal scalability ensures that the database performs efficiently, even as the number of books, users, and transactions increases. MongoDB's use of in-memory processing and its optimized storage engine ensures fast data retrieval and high performance, crucial for an app that involves frequent read and write operations, such as searching, adding, and updating book records.

2. **Real-Time Data Access**
   MongoDB supports high-speed data retrieval, making it ideal for real-time applications like the Book Tracker app. Whether a user is searching for a book by title, author, or edition, MongoDB allows for rapid querying and instant access to the stored data, which enhances the user experience and ensures that users can view their books and make changes without delays.

3. **Flexible Data Structure**
   MongoDB allows for a flexible data model, which is extremely useful in a book tracking system. Books often have varying attributes such as title, author, edition, publisher, genre, publication year, and more. MongoDB's document-based storage lets

us store each book record as a document with different fields, accommodating

variations in book data and providing flexibility as the app evolves. Unlike traditional relational databases, MongoDB does not require predefined schemas, which makes adding new attributes to a book entry (e.g., a field for a special edition or a personal review) straightforward and efficient.

4. **Schema-less Design**
   One of MongoDB's standout features is its schema-less design. This means that each book record can have unique attributes or different structures without disrupting the overall database design. For the Book Tracker app, this flexibility allows users to input books with different levels of detail, whether a user adds just the title and author or goes in-depth with editions, publication history, and genres. This adaptability reduces the need for complex database migrations as the app's features evolve.

How Useful is MongoDB for the Book Tracker App?

1. **Efficient Search and Querying**
   The Book Tracker app will benefit greatly from MongoDB's powerful querying capabilities. MongoDB's support for rich queries allows users to easily search for books based on various attributes (such as title, author, edition, and genre) in an efficient and optimized manner. Additionally, MongoDB supports indexing on frequently searched fields, ensuring that queries like finding a book by its author or edition are executed in the fastest way possible, regardless of how large the collection grows.

2. **Easy Integration with the Web Application**
   MongoDB's integration with Java is seamless, making it ideal for a Java-based Book Tracker app. Java drivers for MongoDB provide a simple API for connecting, querying, and modifying data, making it easy for developers to build and maintain the backend of the Book Tracker app. This easy integration reduces development time and helps ensure smooth operation between the frontend and backend.

3. **Data Redundancy and Replication**
   MongoDB supports replica sets, which provide automatic failover and redundancy. This means that the Book Tracker app will always have a reliable and available data source. Even if one server goes down, MongoDB automatically redirects the traffic to another replica set, ensuring that the app remains operational without any data loss. This feature is essential for maintaining a consistent and dependable user experience, especially as the app gains more users.

4. **Handling Large Data Volumes**
   As the Book Tracker app scales and users begin adding thousands or even millions of books to their collections, MongoDB's ability to handle large data volumes will become even more valuable. MongoDB stores data in collections rather than tables, which allows it to handle large amounts of data efficiently. The database is designed to handle big data with ease, ensuring that the app remains fast and responsive even with growing amounts of data.

5. **Analytics and Data Aggregation**
   MongoDB also provides powerful aggregation capabilities, which will be useful for generating reports or analyzing book collections in the future. For example, if users

want to see how many books they have from a particular author, genre, or publication year, MongoDB can perform real-time aggregation on these datasets, helping the Book Tracker app offer valuable insights and statistics to users.

How Data is Stored in MongoDB

1. **Document-Oriented Storage**
   MongoDB stores data in documents, which are similar to JSON objects. A book in the Book Tracker app can be stored as a document containing all relevant fields, such as:

2. Each book document can contain any number of attributes, making it flexible and adaptable to changes in the data structure. This structure allows the Book Tracker app to add, update, or delete fields as needed without disrupting the overall database schema.

3. **Collections and Databases**
   Books are stored in collections, which are akin to tables in traditional relational databases. MongoDB uses collections to group documents that share similar data. The Book Tracker app would have a "Books" collection where each book record is stored as a separate document. The database would also house other collections for different data types, such as users, categories, and reviews.

4. **Indexing for Faster Retrieval**
   To enhance search performance, MongoDB uses indexes. By creating indexes on fields like author, title, and edition, the Book Tracker app can quickly locate specific books in the collection. Indexing ensures that users can search for books and retrieve them instantly, regardless of the size of the database.

How MongoDB is Useful and Efficient for the Book Tracker App

MongoDB is a natural fit for the Book Tracker app due to its flexibility, scalability, and efficient data management capabilities. It enables the app to handle dynamic data, such as various editions and user-generated content, while maintaining optimal performance for both small and large collections. By leveraging MongoDB's powerful querying, indexing, and replication features, the app ensures a smooth user experience, fast performance, and data security. MongoDB's ability to scale horizontally and handle large amounts of data makes it an ideal choice for the growing needs of the Book Tracker app, providing users with a reliable and responsive platform to track, manage, and explore their book collections.

1.3 MODULES

The Book Tracker website's module-based design ensures a seamless and organized user experience. Each module is tailored to handle specific aspects of the app, providing distinct features and purposes that cater to both functional and user-centric needs. Below is a detailed breakdown of two key modules: **User Management Module** and **Schedule Management Module.**

I. USER MANAGEMENT MODULE

. User Management Module

This module focuses on handling all user-related activities, ensuring secure access, personalized experiences, and streamlined management of user accounts. It provides the foundation for creating an interactive and secure environment for all users.

Features

1. **User Registration**
    a. Allows new users to create an account by providing essential details such as username, email, and password.
    b. Option for additional personalization fields like profile picture or contact preferences.
2. **Login/Logout**
    a. Ensures secure access to the website through a reliable authentication mechanism.
    b. Enables users to log in to access their personalized book collections and features.
    c. Provides an easy-to-use logout feature to ensure data security when users finish their session.
3. **Profile Management**
    a. Lets users view and edit their profiles, including details like username, email, or preferences.
    b. Allows the addition of custom profile settings such as preferred book genres or notification preferences.
    c. Enables password changes or account deletion for user convenience.

PURPOSE

- **Secure User Access**: The User Management Module ensures that every user has a unique account and secure access to their data. This feature protects personal information and prevents unauthorized access.

- **Personalized Experience**: By managing user profiles, the module creates a personalized environment where users can customize their preferences and manage their book collections efficiently.
- **User Accountability**: Tracking user activity (such as adding, updating, or deleting books) becomes easier with unique user profiles, enabling better accountability and data management.

II. SCHEDULE MANAGEMENT MODULE

The Schedule Management Module is designed to help users organize their reading plans and manage their book-related schedules effectively. This feature is especially beneficial for avid readers who wish to track their reading progress or maintain a timeline for book-related tasks.

1. **Add Reading Schedules**
    a. Allows users to set reading goals by specifying a start and end date for each book.
    b. Users can add custom reminders to stay on track with their reading plans.
2. **Track Reading Progress**
    a. Displays the progress of books being read, showing completed chapters or percentage finished.
    b. Provides insights into how well users are adhering to their planned schedule.
3. **Reminder Notifications**
    a. Sends notifications or alerts to remind users about upcoming deadlines for finishing a book or starting a new one.
    b. Allows customization of notification preferences based on user convenience.
4. **Reading History Management**
    a. Keeps a record of books completed, including start and end dates, enabling users to look back on their reading journey.
    b. Provides an option to rate or review books after completing them, enhancing user engagement.

PURPOSE

- **Encourages Consistent Reading Habits**: By setting schedules and reminders, the module motivates users to maintain regular reading habits and achieve their goals.
- **Improves Time Management**: Users can plan their reading time effectively, balancing it with other responsibilities.
- **Enhances User Engagement**: Tracking progress and maintaining a reading history fosters a sense of accomplishment and keeps users engaged with the Book Tracker website.
- **Supports Personal Growth**: This module helps users allocate time for personal development through reading, making it easier to complete books and expand their knowledge.

3. DATA ANALYTICS & REPORTING MODULES

The **Data Analytics & Reporting Module** is a cornerstone feature of the Book Tracker website, aimed at offering insightful analytics and customizable reports to enhance user experience. This module empowers users to gain deeper insights into their reading habits, collection trends, and book management activities. Below is a comprehensive breakdown of the module's structure, features, and purposes.

Features

1. Reading Analytics

- **Track Reading Progress**:
  Provides visual charts and statistics showing how much of a book has been read (e.g., chapters completed, pages covered, or percentage finished).
- **Time Spent on Reading**:
  Logs and analyzes the time users spend reading daily, weekly, or monthly.
- **Reading Speed Calculation**:
  Uses data on pages or chapters completed over time to calculate average reading speed and predict how long it will take to finish a book.

2. Collection Insights

- **Genre and Author Distribution**:
  Analyzes the user's collection to identify trends, such as favorite genres, most-read authors, or common themes in the library.
- **Edition Comparisons**:
  Provides insights into the types of editions users own (e.g., hardcovers vs. paperbacks) and highlights any duplicate editions for better library management.
- **Top Rated Books**:
  Automatically identifies the books with the highest user ratings, helping users understand their preferences better.

3. Activity Reporting

- **Book Addition and Deletion History**:
  Keeps a log of when books were added or removed, providing a timeline of collection changes.
- **Update History**:
  Tracks changes made to book details, such as corrections in author names or editions, ensuring transparency and accountability.
- **Reading Schedule Reports**:
  Summarizes how well users adhere to their schedules, including missed or completed deadlines.

4. Customizable Reports

- **Dynamic Filters**:
  Enables users to generate reports based on specific parameters such as genre, author, reading progress, or time frames.
- **Exportable Reports**:
  Allows users to export their reports in various formats (PDF, CSV, etc.) for sharing or personal record-keeping.
- **Comparison Reports**:
  Provides side-by-side comparisons of different periods, such as reading habits over two months or changes in the collection over a year.

5. Visual Dashboards

- **Interactive Graphs and Charts**:
  Displays key metrics in visually appealing formats, such as bar charts, pie charts, and line graphs.
- **Real-Time Updates**:
  Reflects changes in reading habits or book collections instantly, providing users with up-to-date insights.

- **Goal Tracking Widgets**:
  Includes visual widgets for tracking reading goals, such as the number of books to read per month.

Purpose

1. Enhancing User Engagement

By providing detailed insights into reading habits and library trends, the Data Analytics & Reporting Module motivates users to stay engaged with their collections. Users can set new goals, explore trends, and find new ways to enjoy their books based on the insights provided.

2. Encouraging Self-Improvement

With metrics like reading speed and progress, users can evaluate their reading habits and work towards improvement. Whether it's increasing reading speed or diversifying genres, the analytics help users focus on personal growth.

3. Supporting Data-Driven Decisions

For users with large collections, this module helps them make informed decisions, such as which genres to focus on, which authors to explore further, or which editions to prioritize. The transparency in collection management ensures users maintain a well-organized and meaningful library.

4. Providing Transparency

The module's activity tracking features provide users with a clear overview of their book management activities. This is especially useful for understanding how their library has evolved over time or for identifying any discrepancies in book details.

5. Goal Tracking and Motivation

Reading schedule analytics and goal tracking keep users motivated to stay on course. With visual feedback and reminders, users are encouraged to complete their reading goals, fostering a sense of accomplishment.

6. Facilitating Sharing and Collaboration

Exportable reports allow users to share their reading progress and collection insights with friends, book clubs, or on social platforms. This fosters a community experience and helps users discuss their reading journeys with others.

Benefits of the Data Analytics & Reporting Module

For Readers

1. **Personalized Insights**: Helps readers understand their preferences and habits, allowing them to optimize their reading experience.
2. **Efficient Library Management**: Provides clarity on which books to prioritize, read, or remove, making collection management effortless.
3. **Motivation to Read More**: Visual progress tracking and goal dashboards encourage users to achieve their reading targets.

For Collectors

1. **Detailed Inventory**: Offers collectors a comprehensive view of their books, highlighting editions, genres, and authors for better organization.
2. **Duplicate Management**: Identifies duplicate books or editions, enabling collectors to declutter their collections efficiently.

For Enthusiasts Sharing Data

1. **Showcase Collections**: The ability to generate and share visually appealing reports allows users to showcase their libraries or reading progress with others.
2. **Data-Driven Recommendations**: The module's insights can be used to explore similar books or authors based on user preferences.

SOFTWARE DESCRIPTION

NETBEANS

NetBeans is an open-source integrated development environment (IDE) primarily used for developing applications in Java, although it also supports other programming languages such as PHP, C/C++, HTML5, and JavaScript. Developed initially by Sun Microsystems, NetBeans is now managed and distributed by the Apache Software Foundation as part of the Apache NetBeans project.

KEY FEATURES OF NETBEANS

CROSS-PLATFORM COMPATIBILITY

NetBeans runs on multiple platforms, including Windows, macOS, Linux, and Solaris, providing a consistent development environment regardless of the operating system.

MULTI-LANGUAGE SUPPORT

While its core strength lies in Java, NetBeans supports a variety of other programming languages through plug-ins and extensions.

MODULAR ARCHITECTURE

The IDE is built around a modular architecture, where developers can install or remove modules to tailor the IDE to their specific needs.

2.1 LANGUAGES

JAVA SWING

Java Swing is a part of Java's Standard Library, providing a rich set of graphical user interface (GUI) components for building desktop applications. It is part of the Java Foundation Classes (JFC) and offers a lightweight, platform-independent toolkit for creating windows, buttons, menus, text fields, and other elements of a user interface. Introduced in the Java 2 platform, Swing has become a

popular choice for Java developers aiming to build user-friendly and interactive desktop applications.

KEY FEATURES OF JAVA SWING

PLATFORM INDEPENDENCE:

Like Java, Swing applications are platform-independent and can run on any system with a Java Runtime Environment (JRE).

LIGHTWEIGHT COMPONENTS:

Unlike AWT (Abstract Window Toolkit), Swing components do not rely heavily on the native GUI components of the operating system, making them more flexible and customizable.

RICH SET OF COMPONENTS:

Swing provides a comprehensive collection of components, including labels, buttons, tables, lists, trees, sliders, and more.

PLUGGABLE LOOK AND FEEL (PLAF):

Developers can change the appearance of Swing components to match a specific theme or mimic the look of a native OS GUI.

EVENT-DRIVEN PROGRAMMING:

Swing supports event handling, allowing developers to respond to user actions such as button clicks, mouse movements, and key presses.

CUSTOMIZABILITY:

Developers can create custom components or extend existing ones to fit specific application requirements.

MVC ARCHITECTURE:

Swing follows the Model-View-Controller (MVC) design pattern, separating the representation of data from the user interface, which improves modularity and reusability.

CORE SWING COMPONENTS

TOP-LEVEL CONTAINERS:

- **JFrame:** Represents the main application window.
- **JDialog:** Used for pop-up dialogs.
- **JApplet:** Supports applets that can run in web browsers.

BASIC CONTROLS:

- **JButton:** Represents a clickable button.
- **JTextField:** Allows single-line text input.
- **JCheckBox:** Represents a checkbox for multiple selections.

ADVANCED COMPONENTS:

- **JTable:** Displays tabular data.
- **JTree:** Represents hierarchical data.
- **JTabbedPane:** Provides a tabbed navigation interface.

CONTAINERS:

- **JPanel:** A generic container for grouping other components.
- **JSplitPane:** Divides space into two resizable areas.

APPLICATIONS OF JAVA SWING

- Desktop productivity tools (e.g., text editors, spreadsheets).
- Data visualization applications.
- Simple games.
- Custom GUI applications for specific industries, such as banking and education.

ADVANTAGES OF JAVA SWING

1. Cross-Platform Compatibility:
2. Flexibility and Extensibility:
3. Ease of Development:
4. Backward Compatibility:

3. REQUIREMENTS AND ANALYSIS

The success of a book tracker system hinges on a thorough understanding of the requirements and careful analysis of the operational and technical needs. This phase ensures the development of a robust system that addresses both user expectations and organizational goals while maintaining scalability and efficiency. Below, the requirements and analysis are detailed, covering functional and non-functional aspects as well as user and system-level needs.

4. PROGRAM CODE

ARENA FOR CODING: NETBEANS

TRAIN ADDER JAVA PROGRAM

```
import com.mongodb.MongoClientSettings;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

/**
 *
 * @author harsh
 */
public class train_adder extends javax.swing.JFrame {

    /**
     * Creates new form train_adder
     */
    public train_adder() {
        initComponents();
    }
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel2 = new javax.swing.JLabel();
    jLabel1 = new javax.swing.JLabel();
    jTextField1 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jTextField3 = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    jTextField4 = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    jTextField5 = new javax.swing.JTextField();
    jLabel7 = new javax.swing.JLabel();
    jTextField6 = new javax.swing.JTextField();
    jLabel8 = new javax.swing.JLabel();
    jTextField7 = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();

    jLabel2.setText("jLabel2");

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE); j
```

```
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license

   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
   */

package com.mycompany.sample;

import java.io.File;

   /**
   *
   * @author jevan
   */

public class FileModel {

      private static FileModel instance;

      private File selectedFile;

      // Private constructor to enforce singleton pattern
      private FileModel() {}

      public static FileModel getInstance() {
         if (instance == null) {
            instance = new FileModel();
         }

         return instance;

      }


      public File getSelectedFile() {

         return selectedFile;
      }

      public void setSelectedFile(File selectedFile) {
         this.selectedFile = selectedFile;

      }

   }
   /*
   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
   license
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
 template
 */
package com.mycompany.sample;

 /**
 *
 * @author jevan
 */
import com.mongodb.MongoClientSettings;
import com.mongodb.MongoClientURI; import
com.mongodb.client.MongoClient; import
com.mongodb.client.MongoClients; import
com.mongodb.client.MongoCollection; import
com.mongodb.client.MongoDatabase; import
org.bson.Document;
import javax.swing.JOptionPane; import
java.util.Arrays;

public class NewJFrame extends javax.swing.JFrame {

    /**
     * Creates new form NewJFrame
     */
    public NewJFrame() {

        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jTextField2 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("NAME");
```

```
        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel2.setText("CREATE PASSWORD");

        jButton1.setText("CREATE USER");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("BACK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG,
false)
                            .addComponent(jTextField1)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addGap(182, 182, 182)
                            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
176, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addComponent(jTextField2)
                        .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(0, 0, Short.MAX_VALUE))
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(60, 60, 60)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG,
false)
                .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(0, 0, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
           jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(jPanel1Layout.createSequentialGroup()
              .addContainerGap()
              .addComponent(jLabel1)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
           .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
              .addGap(18, 18, 18)
              .addComponent(jLabel2)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
           .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
              .addGap(18, 18, 18)
           .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addComponent(jLabel3)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addComponent(jButton2)
              .addContainerGap(10, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
           layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(layout.createSequentialGroup()
              .addGap(99, 99, 99)
           .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 253,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addContainerGap(126, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
           layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(layout.createSequentialGroup()
              .addGap(35, 35, 35)
```

```
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(58, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String uri = "Connection string for mongodb";

        // Create a new MongoClient using the connection string
        try (MongoClient mongoClient = MongoClients.create(uri)) {

            // Connect to the "PRACTICE" database
            MongoDatabase database = mongoClient.getDatabase("PRACTICE");

            // Get the "example" collection
            MongoCollection<Document> collection = database.getCollection("example");

            // Create documents to insert
            Document doc1 = new Document("name", jTextField1.getText())
                    .append("password",jTextField2.getText());

            // Insert multiple documents at once
            collection.insertOne(doc1);

            System.out.println("Documents inserted successfully");
            JOptionPane.showMessageDialog(this,"created user successfully");

        }
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        add_book x = new add_book();
            x.setVisible(true);
            this.dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
```

```
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewJFrame.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NewJFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
```

```
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
}

/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
*/

package com.mycompany.sample;

/**
*
* @author jevan
*/
public class Sample {

    public static void main(String[] args) {
        add_book a = new add_book();
        a.setVisible(true);
    }
}


/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
*/
package com.mycompany.sample;

/**
*
* @author jevan
*/
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import javax.swing.JOptionPane;
import org.bson.Document;

public class add_book extends javax.swing.JFrame {

    /**
     * Creates new form add_book
     */
    public add_book() {
```

```
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel3 = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();

        jLabel3.setText("jLabel3");

        jButton2.setText("jButton2");

        jButton3.setText("jButton3");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jButton1.setText("LOGIN");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel1.setText("USERNAME:");

        jLabel2.setText("PASSWORD : ");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG,
false)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 98,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField1)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 76,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
191, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(49, 49, 49)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(32, 32, 32)
                .addComponent(jButton1)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(93, 93, 93)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(104, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
```

```
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(47, 47, 47)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(80, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String uri = "Connecting string mongodb";  // Replace with your actual MongoDB URI

        // MongoDB connection and operation
        try (MongoClient mongoClient = MongoClients.create(uri)) {
            // Connect to the "PRACTICE" database
            MongoDatabase database = mongoClient.getDatabase("PRACTICE");

            // Get the "users" collection
            MongoCollection<Document> collection = database.getCollection("example");

            // Query the database for a document with the matching username and password
            Document query = new Document("name",
  jTextField1.getText()).append("password", jTextField2.getText());
            Document user = collection.find(query).first();
            Document admin = new Document("name", "admin").append("password", "admin");

            if (admin.equals(query)){
                NewJFrame x = new NewJFrame();
                x.setVisible(true);
                this.dispose();
            }
            else if (user != null) {
                // User found, credentials are correct
                dash x = new dash();
                x.setVisible(true);
                this.dispose();
            } else {
                // User not found, credentials are incorrect
                JOptionPane.showMessageDialog(this,"give the correct password");
            }
        }
    }

    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(add_book.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(add_book.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(add_book.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(add_book.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new add_book().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
}



/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
*/
package com.mycompany.sample;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import javax.swing.JOptionPane;
import org.bson.Document;

/**
*
* @author jevan
*/
public class addbook extends javax.swing.JFrame {

    /**
     * Creates new form addbook
     */
    public addbook() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel4 = new javax.swing.JLabel();
        buttonGroup1 = new javax.swing.ButtonGroup();
        jScrollBar1 = new javax.swing.JScrollBar();
```

```
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();

        jLabel4.setText("jLabel4");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jButton1.setText("ADD THE BOOK");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel2.setText("NAME OF THE BOOK :");

        jLabel5.setText("EDITION");

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("ADD BOOK");

        jLabel3.setText("AUTHOR OF THE BOOK :");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG,
false)
                                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
170, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jTextField1)
```

```
                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
159, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField2)
                        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE,
135, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE, 247,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(73, 73, 73)
                            .addComponent(jButton1))))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(49, 49, 49)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 159,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(12, 12, 12)
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel3)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel5)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton1)
                .addContainerGap())
        );

        jButton2.setText("BACK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            jButton2ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(65, 65, 65)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addGap(145, 145, 145)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
111, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(76, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(22, 22, 22)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton2)
            .addContainerGap(13, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // TODO add your handling code here:
    String uri = "connecting string mongodb";

    // Create a new MongoClient using the connection string
    try (MongoClient mongoClient = MongoClients.create(uri)) {

        // Connect to the "PRACTICE" database
        MongoDatabase database = mongoClient.getDatabase("PRACTICE");

        // Get the "example" collection
```

```
            MongoCollection<Document> collection = database.getCollection("BOOK");

            // Create documents to insert
            Document doc1 = new Document("name", jTextField1.getText().toLowerCase())
                    .append("author",jTextField2.getText().toLowerCase())
                    .append("edition", jTextField3.getText().toLowerCase());

            // Insert multiple documents at once
            collection.insertOne(doc1);
            jTextField1.setText(null);
            jTextField2.setText(null);
            jTextField3.setText(null);
            System.out.println("Documents inserted successfully");
            JOptionPane.showMessageDialog(this,"inserted the book successfully");

        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(addbook.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(addbook.class.getName()).log(java.util.logging.Level.SE
```

```
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(addbook.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(addbook.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new addbook().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollBar jScrollBar1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    // End of variables declaration
}



/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
*/
package com.mycompany.sample;

/**
*
```

```
 * @author jevan
 */
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
public class authorfind extends javax.swing.JFrame {

    /**
     * Creates new form authorfind
     */
    public authorfind() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("FIND THE BOOK");

        jLabel2.setText("AUTHOR NAME :");

        jButton1.setText("FIND BOOK");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
```

```
        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null},
                {null, null},
                {null, null},
                {null, null}
            },
            new String [] {
                "BOOK", "EDITION"
            }
        ));
        jScrollPane1.setViewportView(jTable1);

        jButton2.setText("BACK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
135, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequentialGroup()
                            .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 178,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(jButton1))
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 339,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(136, 136, 136)
                        .addComponent(jButton2)))
```

```
                .addContainerGap(31, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1))
                .addGap(31, 31, 31)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 151,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton2)
                .addContainerGap(8, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String URI = "Connecting string mongodb";
         DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.setRowCount(0); // Clear existing data
        String authorName = jTextField1.getText();
        try (MongoClient mongoClient = MongoClients.create(URI)) {
            MongoDatabase database = mongoClient.getDatabase("PRACTICE");
            MongoCollection<Document> collection = database.getCollection("BOOK");

            // Define the filter based on author name
            Document filter = new Document("author", authorName);

            try (MongoCursor<Document> cursor = collection.find(filter).iterator()) {
                boolean found = false;
                while (cursor.hasNext()) {
                    found = true;
                    Document doc = cursor.next();
                    String name = doc.getString("name");
                    String edition = doc.getString("edition");
```

```
                model.addRow(new Object[]{name, edition});
            }

            if (!found) {
            JOptionPane.showMessageDialog(this, "No books found for the author: " +
authorName);
            } else {
            JOptionPane.showMessageDialog(this, "Books by " + authorName + " loaded
successfully!");
            }
          }
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        this.dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
          }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(authorfind.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(authorfind.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(authorfind.class.getName()).log(java.util.logging.Level.S
```

```
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(authorfind.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new authorfind().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}



/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
*/
package com.mycompany.sample;

/**
*
* @author jevan
*/
public class dash extends javax.swing.JFrame {

    /**
     * Creates new form dash
     */
    public dash() {
        initComponents();
    }
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jPanel3 = new javax.swing.JPanel();
    jPanel2 = new javax.swing.JPanel();
    jButton2 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jButton6 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton5 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jButton2.setText("ADD BOOK");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    jButton4.setText("DELETE BOOK");
    jButton4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton4ActionPerformed(evt);
        }
    });

    jButton6.setText("AVAILABLE BOOK");
    jButton6.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton6ActionPerformed(evt);
        }
    });

    jButton3.setText("UPDATE BOOK");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });
```

```
        jButton5.setText("FIND BY AUTHOR");
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                    .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG,
false)
                        .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
107, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE,
124, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jButton5)))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(68, 68, 68)
                        .addComponent(jButton6)))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL INE)
                    .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)
```

```
.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL INE)
                .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
            .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("DASHBOARD");

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addGap(85, 85, 85)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 88,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
```

```
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(23, 23, 23)
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(22, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(18, 18, 18)
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(30, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(50, 50, 50)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(89, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(42, 42, 42)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(86, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        addbook x = new addbook();
        x.setVisible(true);
    }
```

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        updatedelete q = new updatedelete();
        q.setVisible(true);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        delete v = new delete();
        v.setVisible(true);
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        authorfind w = new authorfind();
        w.setVisible(true);
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        showbook r = new showbook();
        r.setVisible(true);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(dash.class.getName()).log(java.util.logging.Level.SEVER E,
null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(dash.class.getName()).log(java.util.logging.Level.SEVER E,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(dash.class.getName()).log(java.util.logging.Level.SEVER E,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(dash.class.getName()).log(java.util.logging.Level.SEVER E,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new dash().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    // End of variables declaration
}


/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
*/
package com.mycompany.sample;

/**
*
* @author jevan
*/
import com.mongodb.client.MongoClient; import
com.mongodb.client.MongoClients;
```

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

import javax.swing.*;
public class delete extends javax.swing.JFrame {

    /**
     * Creates new form delete
     */
    public delete() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("NAME :");

        jButton1.setText("DELETE");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel2.setText("DELETE BOOK");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
```

```
NG)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI NG)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
72, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 149,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(21, 21, 21)
                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
91, javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(35, 35, 35)
                        .addComponent(jButton1)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)
                .addGap(18, 18, 18)
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap())
        );

        jButton2.setText("BACK");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
```

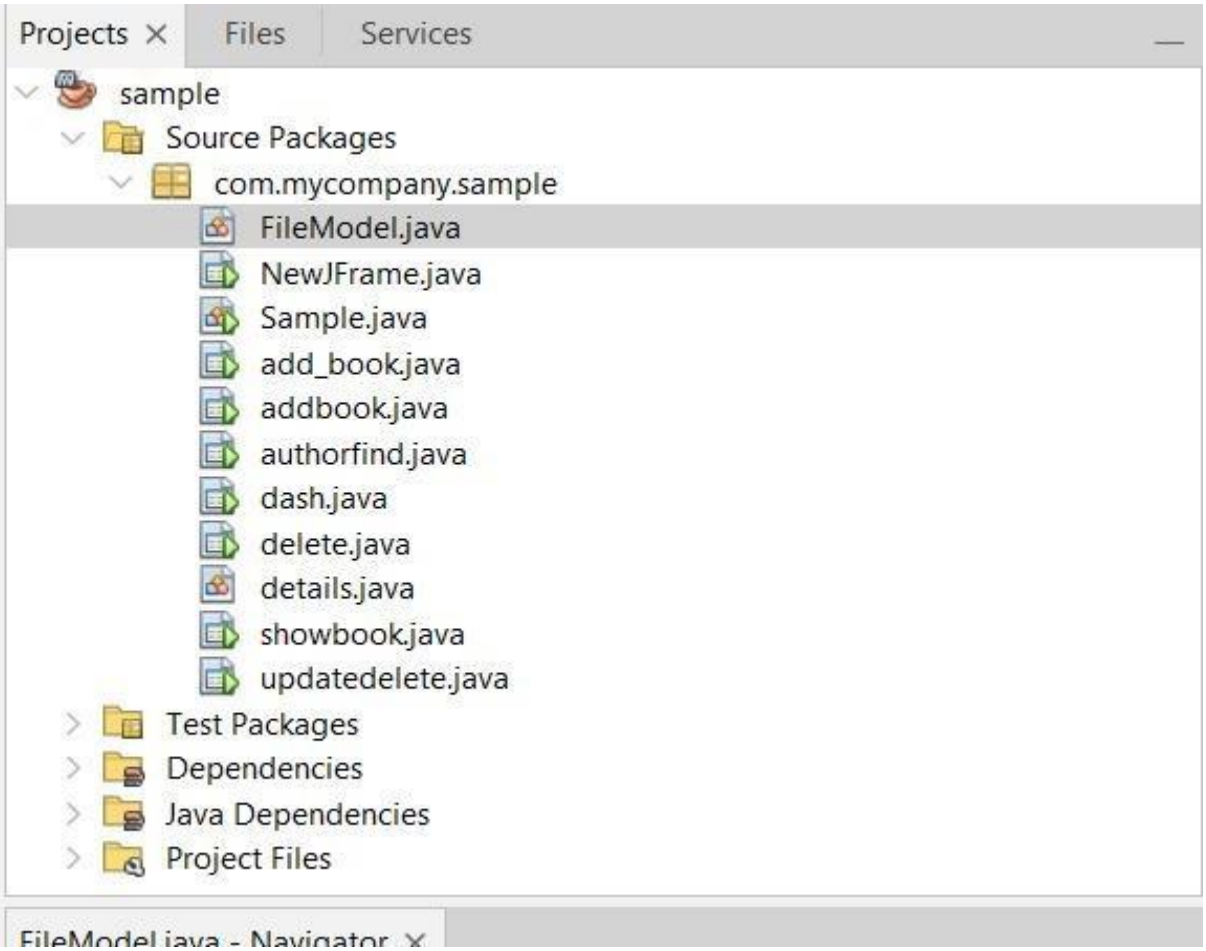
```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(97, 97, 97)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addGap(133, 133, 133)
                    .addComponent(jButton2)))
            .addContainerGap(142, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
           layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(layout.createSequentialGroup()
              .addGap(48, 48, 48)
           .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addComponent(jButton2)
              .addContainerGap(86, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String URI = "connection string mongodb";;
        String bookName = jTextField1.getText().trim();

if (bookName.isEmpty()) {
    JOptionPane.showMessageDialog(this, "Please enter a book name to delete.");
  }

  try (MongoClient mongoClient = MongoClients.create(URI)) {
    MongoDatabase database = mongoClient.getDatabase("PRACTICE");
    MongoCollection<Document> collection = database.getCollection("BOOK");

    // Define filter based on book name
    Document filter = new Document("name", bookName);

    // Check if the book exists
    Document book = collection.find(filter).first();
    if (book == null) {
        JOptionPane.showMessageDialog(this, "Book not found.");
    }

    // If book exists, proceed with deletion
    var deleteResult = collection.deleteOne(filter);
```

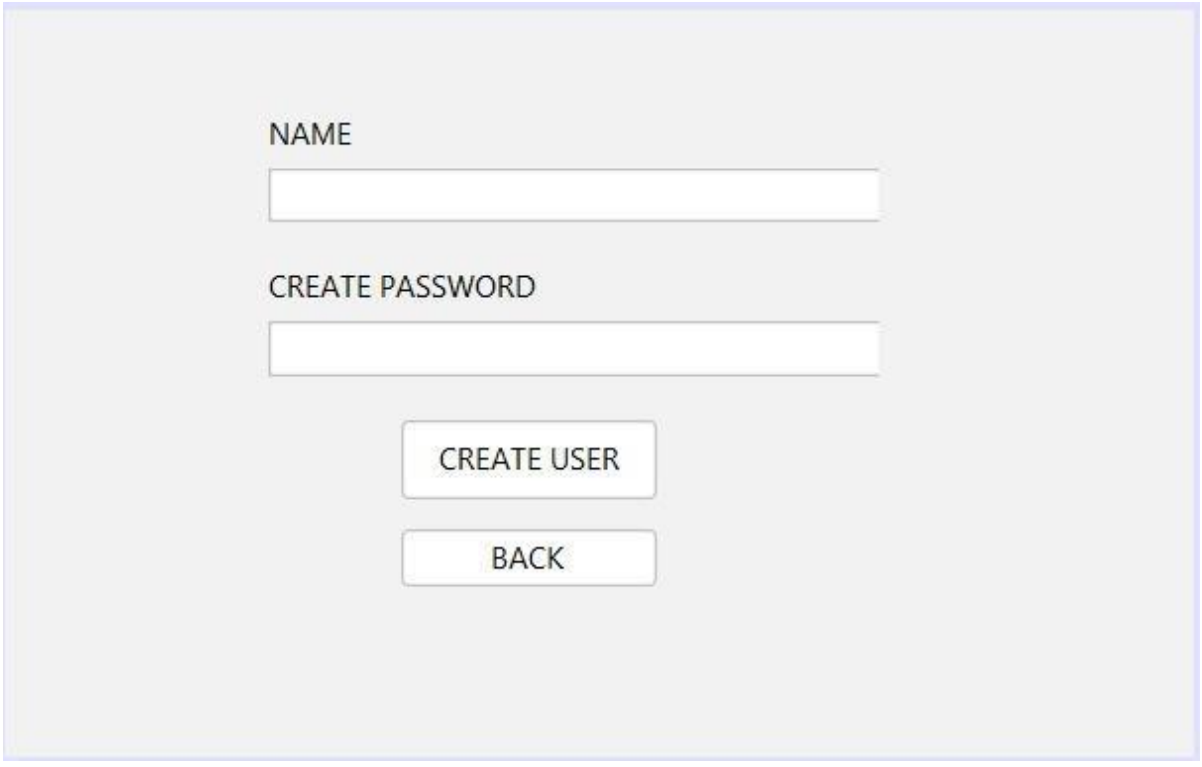
```
        if (deleteResult.getDeletedCount() > 0) {
            JOptionPane.showMessageDialog(this, "Book deleted successfully!");
        } else {
            JOptionPane.showMessageDialog(this, "Failed to delete the book.");
        }
    }
        }

        private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
            // TODO add your handling code here:
            this.dispose();
        }

        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            /* Set the Nimbus look and feel */
            //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
            /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
             * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
             */
            try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                    if ("Nimbus".equals(info.getName())) {
                        javax.swing.UIManager.setLookAndFeel(info.getClassName());
                        break;
                    }
                }
            } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(delete.class.getName()).log(java.util.logging.Level.SEVE RE,
null, ex);
            } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(delete.class.getName()).log(java.util.logging.Level.SEVE RE,
null, ex);
            } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(delete.class.getName()).log(java.util.logging.Level.SEVE RE,
null, ex);
            } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(delete.class.getName()).log(java.util.logging.Level.SEVE RE,
null, ex);
```

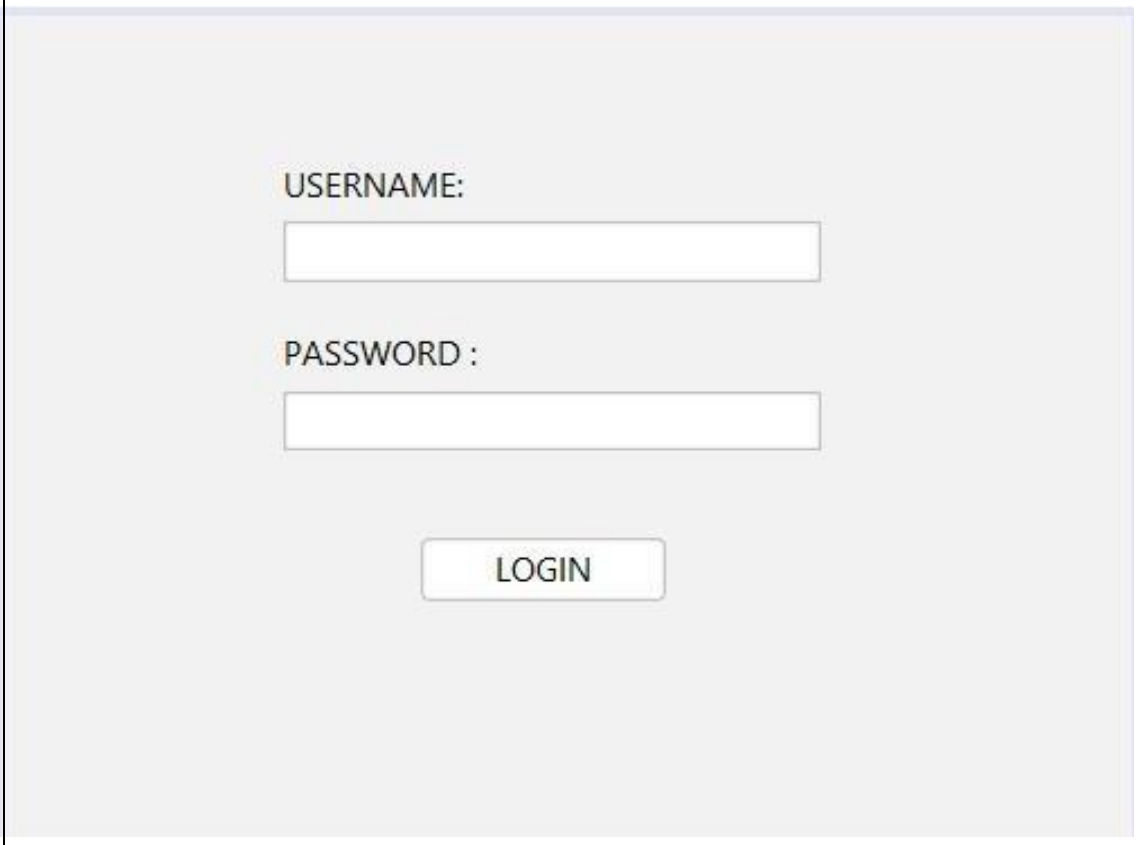
```
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new delete().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}



/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
*/
package com.mycompany.sample;

import java.awt.Color; import
java.awt.Component; import
java.awt.Font;  import
java.awt.Image;
import java.awt.image.BufferedImage;
import javax.swing.BorderFactory; import
javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel; import
javax.swing.JPanel;

/**
*
* @author jevan
*/
public class details {
```

```
    public static void showDetailView(String title, String description, BufferedImage image) {
    JFrame detailFrame = new JFrame("Detail View");
    detailFrame.setSize(400, 400);
    detailFrame.setLocationRelativeTo(null);

    JPanel detailPanel = new JPanel();
    detailPanel.setLayout(new BoxLayout(detailPanel, BoxLayout.Y_AXIS));
    detailPanel.setBackground(Color.WHITE);
    detailPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

    JLabel titleLabel = new JLabel(title);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    detailPanel.add(titleLabel);

  ImageIcon imageIcon = new ImageIcon(image.getScaledInstance(200, 150,
Image.SCALE_SMOOTH));
    JLabel imageLabel = new JLabel(imageIcon);
    imageLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    detailPanel.add(imageLabel);

  JLabel descriptionLabel = new JLabel("<html><p style='width:300px;'>" + description +
"</p></html>");
    descriptionLabel.setFont(new Font("Arial", Font.PLAIN, 14));
    descriptionLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    detailPanel.add(descriptionLabel);

    JButton closeButton = new JButton("Close");
    closeButton.setAlignmentX(Component.CENTER_ALIGNMENT);
    closeButton.addActionListener(e -> detailFrame.dispose());
    detailPanel.add(Box.createVerticalStrut(10));
    detailPanel.add(closeButton);

    detailFrame.add(detailPanel);
    detailFrame.setVisible(true);
  }
  }
```

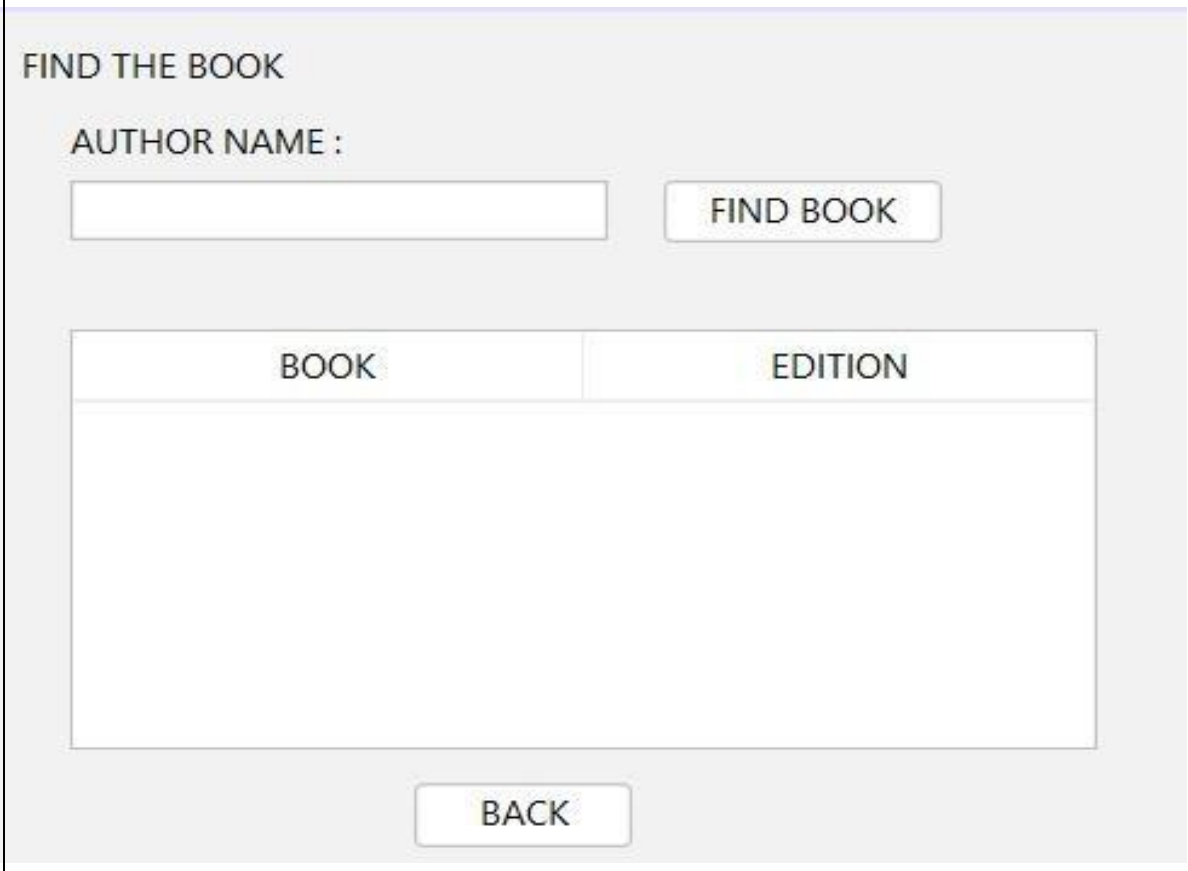
PROJECT DESIGN :

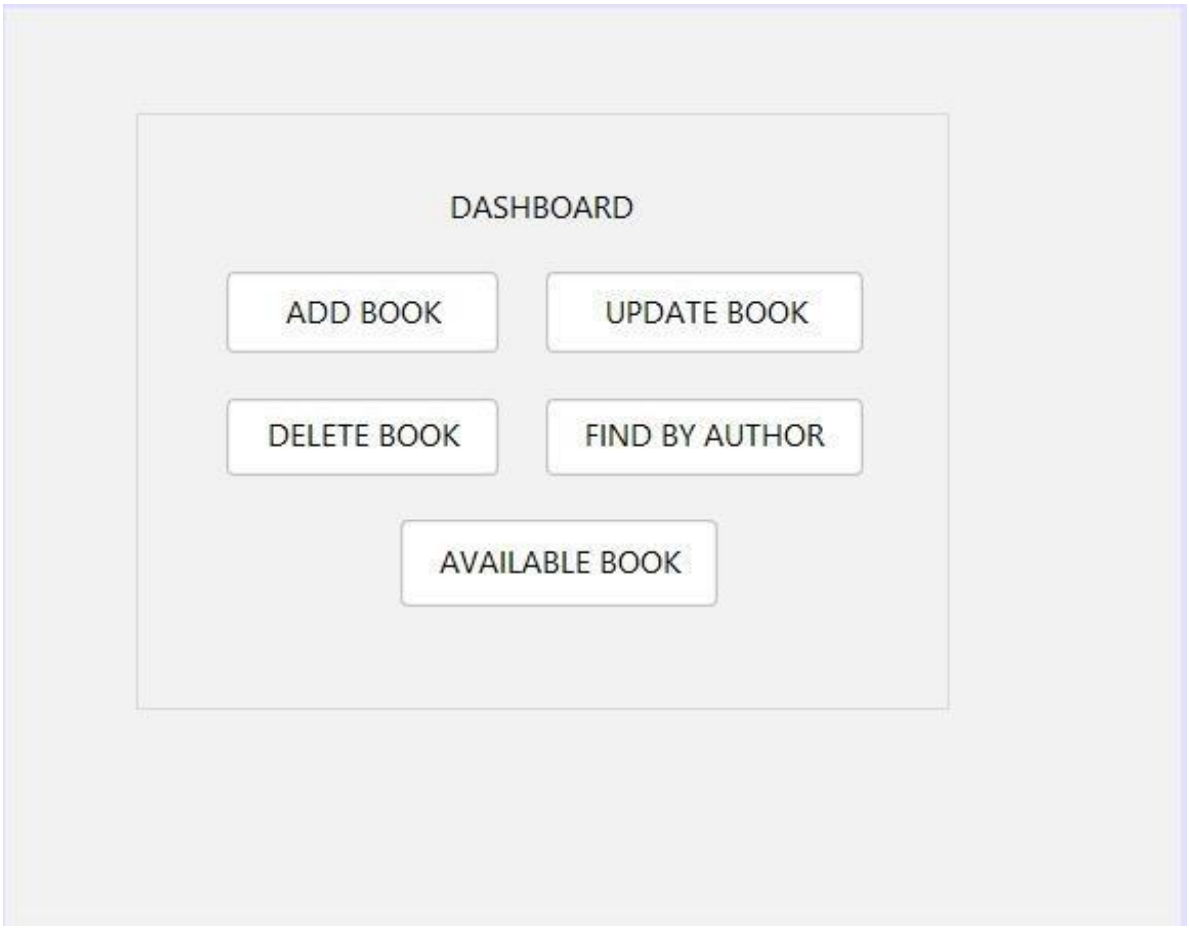


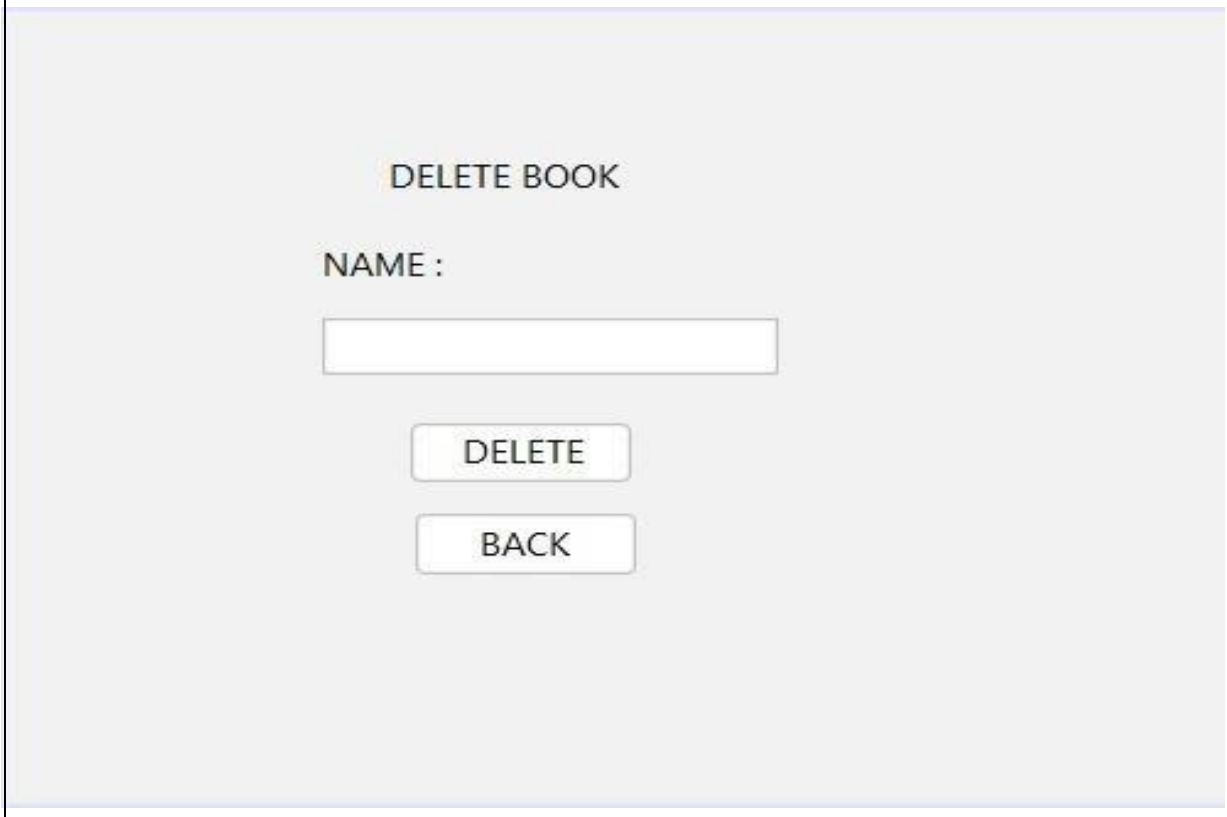


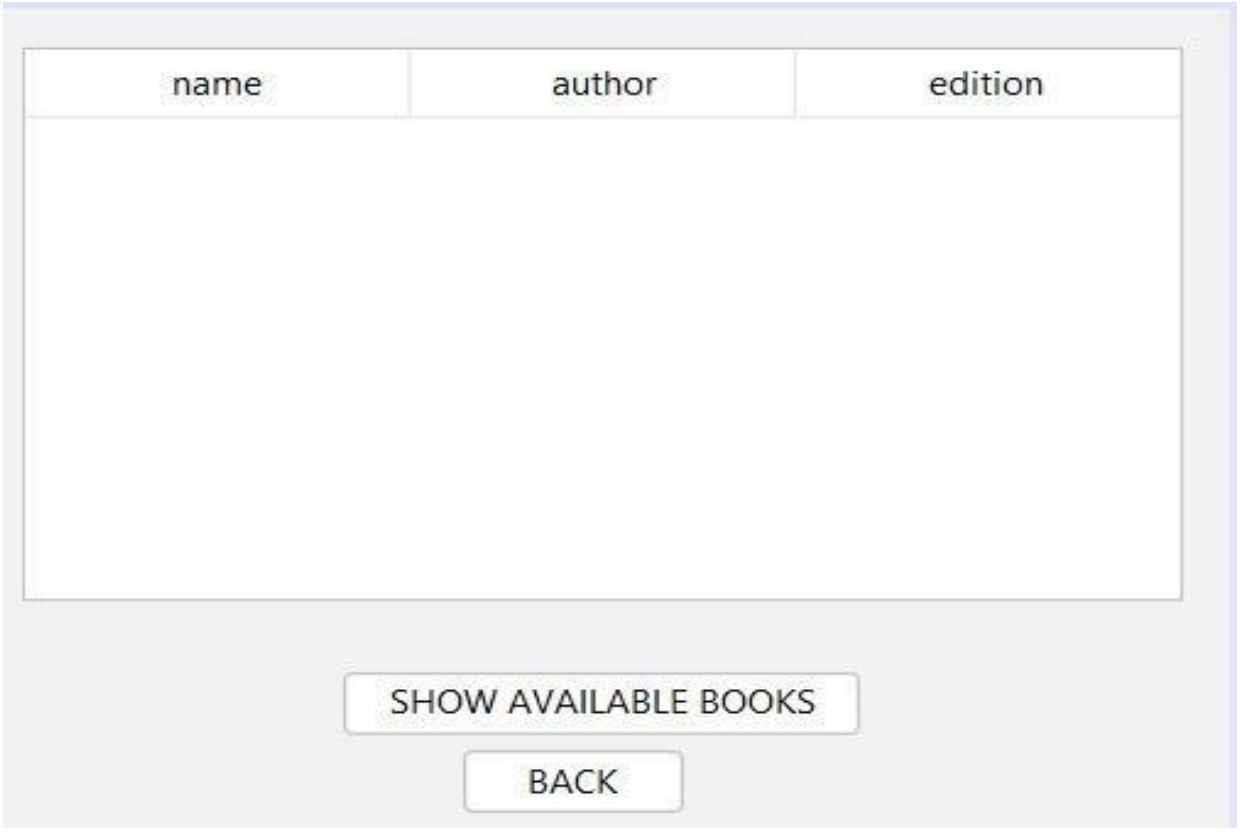

USERNAME:
PASSWORD :
LOGIN


ADD BOOK
NAME OF THE BOOK :
AUTHOR OF THE BOOK :
EDITION
ADD THE BOOK
BACK


FIND THE BOOK
AUTHOR NAME :
FIND BOOK
BOOK
EDITION
BACK


DASHBOARD
ADD BOOK
UPDATE BOOK
DELETE BOOK
FIND BY AUTHOR
AVAILABLE BOOK


DELETE BOOK
NAME :
DELETE
BACK


name
author
edition
SHOW AVAILABLE BOOKS
BACK


UPDATE THE BOOK
NAME OF THE BOOK :
AUTHOR :
EDITION :
UPDATE
BACK

6.CONCLUSION

The Book Tracker Project successfully addresses the challenges of managing and organizing personal book collections in the digital age. By leveraging the power of Java for backend development and MongoDB for flexible, scalable storage, the system offers a comprehensive solution that combines efficiency, accessibility, and user-friendliness. Its robust features, including the ability to add, update, find, and delete books, empower users to create and maintain a digital library tailored to their preferences and needs.

With modules like user management, reading schedules, and advanced data analytics, the project promotes consistent reading habits, enhances user engagement, and provides actionable insights into personal reading trends. Additionally, the inclusion of real-time access and seamless integration ensures that users can manage their collections effortlessly, whether at home or on the go.

In conclusion, the Book Tracker Project bridges the gap between traditional book-keeping methods and modern technological advancements, fostering a deeper connection between users and their literary pursuits. It is a scalable, reliable, and innovative tool that simplifies book management, making it an essential resource for readers and collectors in the 21st century.