Overview

This library app allows users to add books (PDFs along with metadata such as author, title, and description) to a digital shelf. The app tracks whether a book has been read or remains unread. The app is built using the MVC (Model-View-Controller) architecture in C#.

1. Design Choices

1.1. MVC Architecture

- **Model: Represents the application's data and the business rules around it, including book information, statuses, and PDF files.**
- **View: The user interface (UI) displays data such as the list of books and their statuses.**
- **Controller: Handles user input, manipulates the model, and updates the view, managing workflows like adding and editing books and updating their status.**

PLEASE REMEMBER TO PROVIDE REASONING FOR WHY YOU HAVE CHOSEN THIS.

1.2. Language and Framework

- **Language: C# for its robustness and compatibility with the .NET framework and MVC architecture.**
- **Framework: ASP.NET MVC for its separation of concerns, making the app easier to manage and scale.**

1.3. File Storage

- **PDF Storage: PDFs are saved as byte arrays in the database, ensuring the files are contained within the database for portability.**

2. Database Structure

2.1. Tables

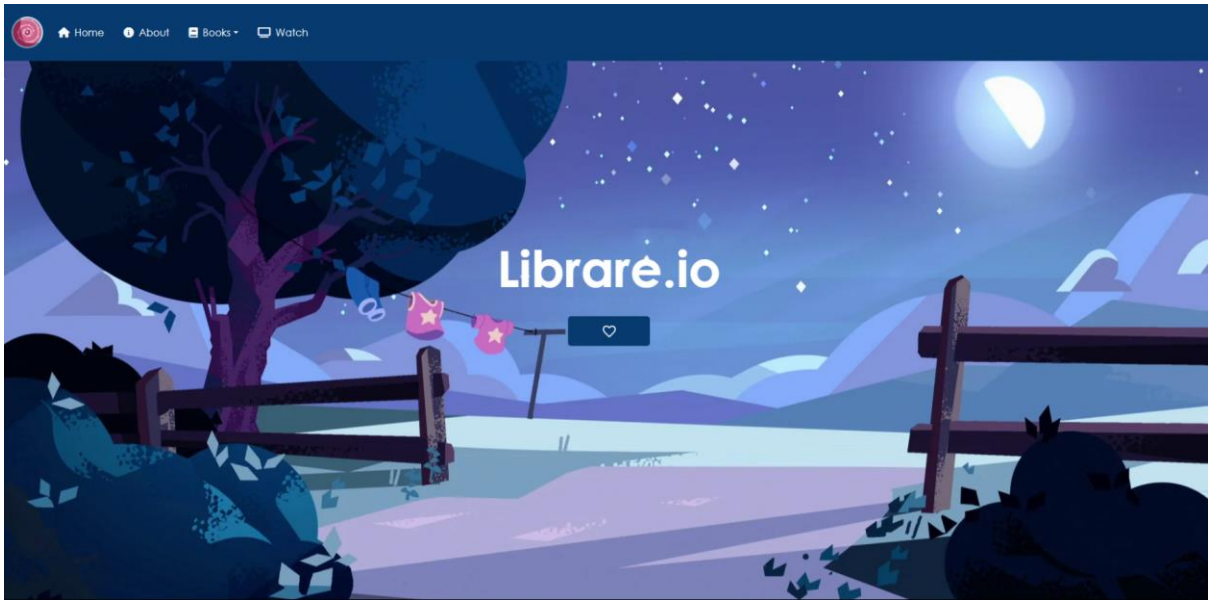
- **Books**
    - **BookID (Primary Key, Auto-Increment): Unique identifier for each book.**
    - **Title (String): The title of the book.**
    - **Author (String): The name of the author.**
    - **Description (Text): A brief description or summary of the book.**
    - **PDFData (Byte Array): The actual PDF file stored as a byte array.**
    - **IsRead (Boolean): A boolean value representing the status of the book (true for "Read," false for "Unread").**
    - **DateAdded (DateTime): The date when the book was added.**

2.2. Relationships

There are no complex relationships in this database, all data is contained within the Books table.

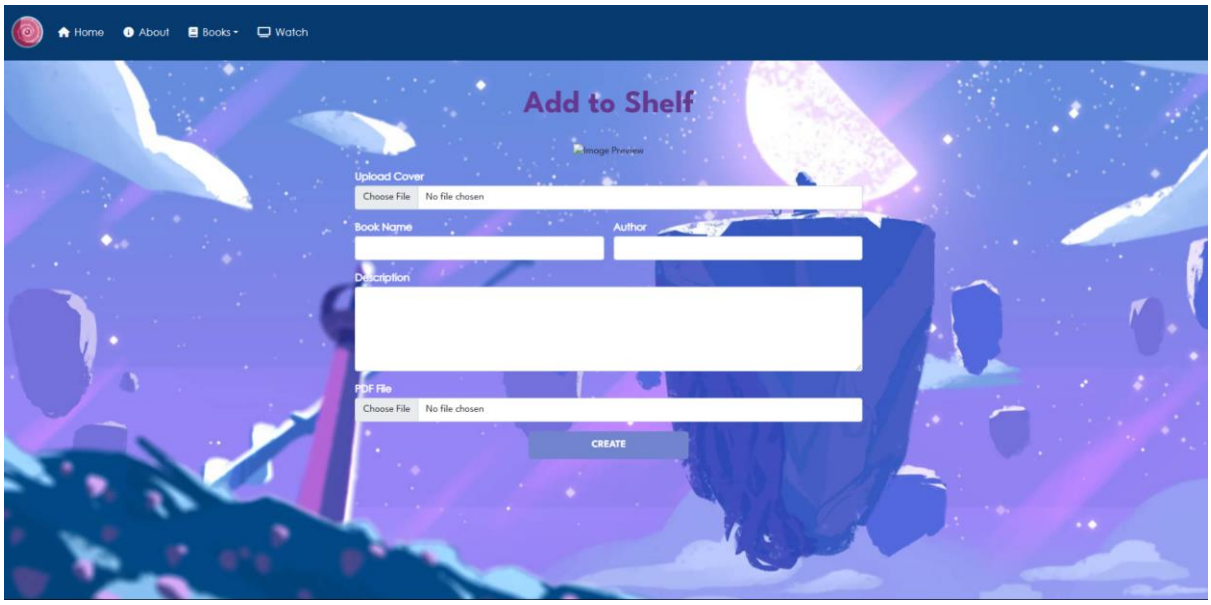
3. GUI Layout

3.1. Main View (Book Shelf)



- **Book List: Displays all books, their titles, authors, and current status (Read/Unread).**
- **Filters: Dropdown or toggle to filter books by status (Read/Unread).**
- **Add New Book Button: Opens a form to add a new book.**

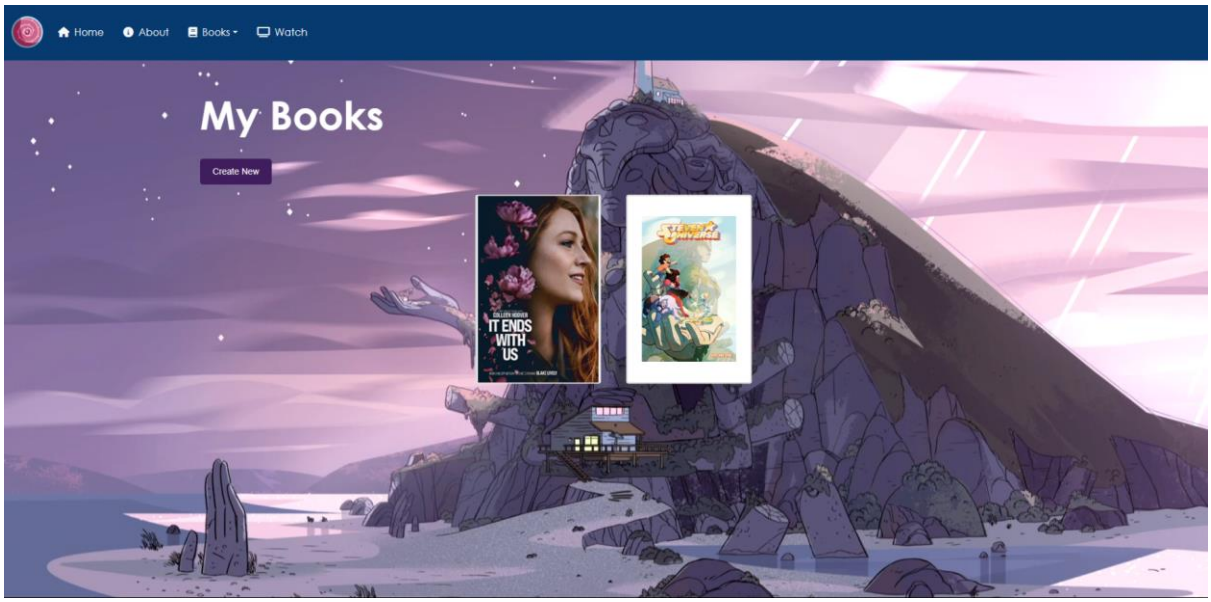
3.2. Add/Edit Book View

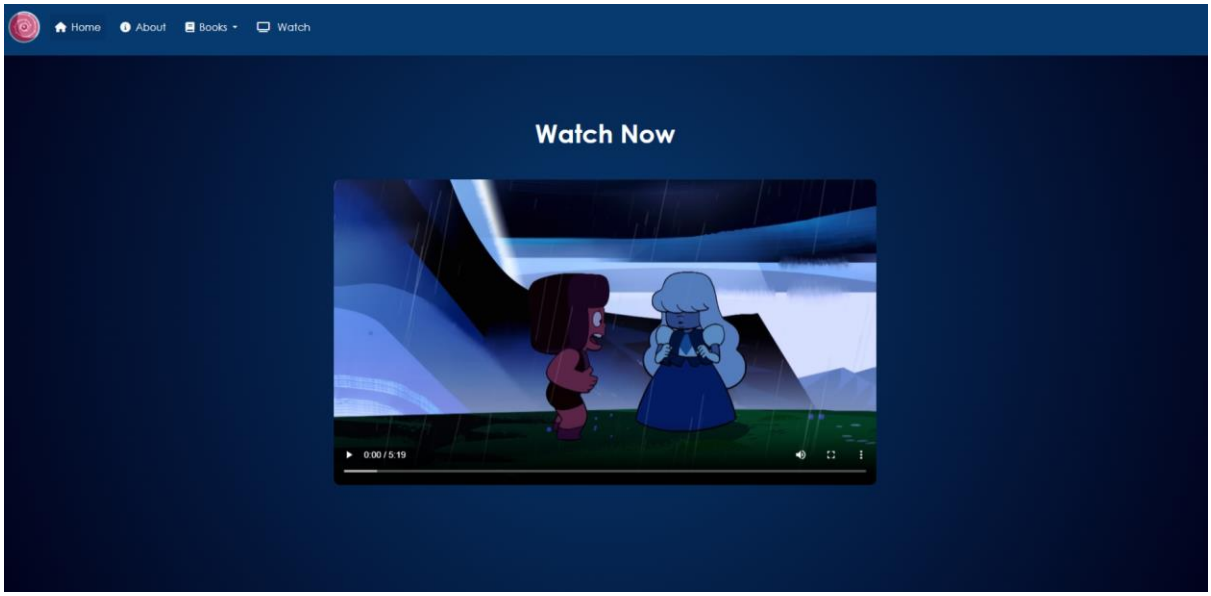


- **Title Field: Input field for the book's title.**
- **Author Field: Input field for the author's name.**

- **Description Field: Text area for a brief description.**
- **PDF Upload: File upload control for the book's PDF.**
- **Status Selector: Checkbox or toggle to set the status as "Read" (checked) or "Unread" (unchecked).**
- **Save/Cancel Buttons: Buttons to save or cancel.**

3.3. Book Details View





- **Title and Author Display: Displays the book's title and author.**
- **Description Display: Shows the book's description.**
- **PDF Viewer: Embedded PDF viewer for reading within the app (optional feature).**
- **Status Display/Toggle: Displays the current status with an option to toggle between "Read" and "Unread."**

4. Assumptions

1. **User Behavior: It is assumed that users will correctly upload PDF files that are well-formed and not corrupted (or that they will only upload PDFs, thus enforcing the restriction of only PDF files).**
2. **Data Integrity: The app assumes that all input data (e.g., book title, author, description) will be provided in a proper format and will be validated before being stored in the database.**
3. **Storage: The database will be able to handle the storage of potentially large byte arrays (PDF files) without performance issues.**
4. **Security: Basic security measures are in place to prevent unauthorized access to the database and uploaded files.**
5. **Environment: The app is assumed to be hosted in a controlled environment with sufficient resources for file storage and database operations.**

5. Constraints

1. **File Size: There may be a practical limit on the size of PDF files that can be uploaded due to database storage constraints or application settings.**
2. **Database Performance: Storing large numbers of PDFs as byte arrays in the database could impact performance, especially if the database grows significantly in size.**
3. **Scalability: The current design assumes a moderate number of users and books; significant scaling may require architectural changes.**
4. **Security Considerations: As PDFs are stored in the database, there is a constraint to ensure that the database is properly secured against unauthorized access and that sensitive data is encrypted if necessary.**
5. **Compatibility: The application is designed for environments compatible with C# and the .NET framework, limiting its deployment to such environments.**

6. Conclusion

This design ensures a clean separation of concerns, making the app scalable and easy to maintain. The decision to store PDFs as byte arrays within the database centralizes data, enhancing portability but may introduce constraints related to storage and performance. The user interface is designed for simplicity, making it easy for users to manage their digital library.