Design Documentation

# Functional Design  
  
1. \*\*Book Tracking Feature\*\*: A main component of this application will be a comprehensive book tracking feature. The librarian will be able to maintain an inventory of all the books, marking when books are borrowed or returned. Notifications regarding overdue books would be automated according to the circulation period of each book.   
   
2. \*\*Search Functionality\*\*: Library members will have an interface to search for available books by title or author, allowing them to find books without having to physically search the library. Users can also browse through different categories of books, authors, and publishers.  
  
# Technical Design  
  
1. \*\*Frontend\*\*: The client application would be built using React.js, which is a performant and flexible JavaScript library for building user interfaces. It could further be styled with a CSS framework like Bootstrap for interactive, attractive UI.  
   
2. \*\*Backend\*\*: The server would run on a backend framework like Node.js with Express.js, which can effectively manage API calls, authentication, and other server-side tasks.  
   
3. \*\*Database\*\*: Considering the nature of the data to be stored and retrieved, we will be using a SQL database like PostgreSQL. It’s a robust RDBMS system and offers excellent support for indexing and searching – both of which are essential for this application.  
  
4. \*\*Authentication\*\*: JSON Web Token (JWT) would be used for managing user sessions. This token will carry a payload of user credentials, which is encrypted and only visible to the server.  
  
# System Architecture  
  
1. \*\*Frontend (Client)\*\*: Using React.js, the application will feature a component-based architecture allowing for optimal user experience and UI.  
  
2. \*\*Backend (Server)\*\*: The server is based on Node.js and Express.js. It intercepts client requests and can securely process them. It handles user authentication via JWT, where a token is provided to users upon login. Every subsequent request must include the token for verification.  
  
3. \*\*Database\*\*: PostgreSQL is our database of choice. It will store user information and book records. Borrowing and returning of books will update the status in the database.  
  
4. \*\*Authentication Service\*\*: JWT will be used for authentication. Upon successful login, the server will issue a token to the client. This token will be used in the header for any further requests to ensure proper authorization.  
  
5. \*\*Data Flow\*\*: A client sends a request, which, after passing through the server, gets directed to the necessary database interactions. After the database processing, the data gets sent back to the client via the server.  
  
6. \*\*Admin Management\*\*: Admins can interface with the system through a dedicated Admin dashboard. They can add/ remove books, keep track of borrowed and returned books, receive notifications of overdue books, and more.  
  
As for the PDF, a digital assistant wouldn't be able to generate a PDF directly. However, you could easily copy this text into a document for the platform you'd like (like Word, Google Docs, etc.) and then export/save that document as a PDF.  
  
# Architecture Diagram  
  
Please refer to a software engineer or a system architect to generate a detailed diagram based on the given specifications regarding frontend-client, backend-server, database, and admin management.