Design Documentation

\*\*Functional Design:\*\*  
  
1. \*\*Book Categorization:\*\*  
The system will provide a user-friendly dashboard for library managers. This dashboard will show a database of all books currently in the library. Managers can categorize books by multiple factors including genre, author, publication date, book length, etc. Searches can be performed based on these categories, allowing managers to organize and keep track of their collections easily.  
  
2. \*\*Online Reservation:\*\*  
For library members, an online book reservation feature will be provided. Members can browse books in the library's inventory and reserve books currently checked out by other members. They will receive an automatic notification when the book is available. They can then come to the library to borrow the book or have it reserved until they visit.  
  
\*\*Technical Design:\*\*  
  
The system will consist of a front-end developed using React.js, a backend developed with Django, and PostgreSQL for the database.  
  
1. \*\*Frontend (Client):\*\*  
The front-end will be developed in React.js to provide a smooth and interactive user interface. It will consume APIs provided by the Django backend to render data dynamically.  
  
2. \*\*Backend (Server):\*\*  
The backend will be developed using Django. Django's capabilities, like inbuilt admin panel, object-relational mapper (ORM), and user authentication features, make it a perfect choice for this system.  
  
3. \*\*Database:\*\*  
The system will use a PostgreSQL database to store all data, including books, categories, library members, and reservations. Postgres is known for its performance, scalability and ability to handle complex queries, making it a suitable choice for this project.  
  
4. \*\*Authentication Service:\*\*  
Authentication and authorization will be handled by Django's inbuilt user authentication system, combined with OAuth for token-based authentication.   
  
\*\*Architecture Diagram:\*\*  
  
1. Users (library managers or members) interact with the system using the web interface built with React.js.  
2. React.js sends requests to the Django backend, which fetches or manipulates data from the PostgreSQL database.  
3. For secured endpoints (e.g., managing books, managing reservations), Django checks whether the request has a valid authentication token.  
4. If the token is valid, Django performs the requested operation. If not, it responds with an error message.  
5. All responses from Django are sent back to the React.js app which then updates the user interface accordingly.  
  
\*\*Admin Management:\*\*  
  
The Django's inbuilt admin panel will provide an interface for system admins to manage all parts of the system. This includes:  
  
- Adding, editing and deleting books from the library.  
- Categorizing and keeping track of books.  
- Managing and authenticating library members.  
- Viewing and managing reservations.  
- Responding to member queries and complaints.   
  
\*\*Final Note:\*\*  
To create a PDF version of this specification document, you can use a markdown-to-PDF converter with this markdown text. For diagrams, you can use diagramming software like Lucidchart or Microsoft Visio to create a detailed system architecture map, based on the systems and interactions described in the document.   
  
You asked me to provide a fancy PDF format response plan but as a text-based AI, I cannot provide PDF responses. My suggestion above should guide you to create a PDF document from my response.