Technical Document

Backend Technologies

1. **Framework: NestJS**
    - **Description**: A progressive Node.js framework built for building scalable and efficient server-side applications.
    - **Use In**: The core server-side logic, API routing, and managing middleware/services.
2. **Database: PostgreSQL (with TypeORM)**
    - **Description**: PostgreSQL is a relational database, and TypeORM is an ORM that helps interact with the database using TypeScript/JavaScript.
    - **Use In**: Storing user data, books, highlights, and other relational data.
3. **Authentication: JWT (with optional OAuth2 using Passport)**
    - **Description**: JWT (JSON Web Tokens) for stateless user authentication and optional OAuth2 for third-party login (e.g., Google).
    - **Use In**: Authorize users and handle login/logout functionality, token-based authentication for protected routes.
4. **API Documentation: Swagger**
    - **Description**: A tool to automatically generate API documentation and enable testing of API endpoints.
    - **Use In**: Provides interactive documentation for developers to test and understand API functionality.
5. **Logging: Winston**
    - **Description**: A versatile logging library that supports different log levels and outputs (console, file, etc.).
    - **Use In**: Capture and monitor application logs (errors, info, warnings) to aid in debugging and performance tracking.
6. **Cloud Hosting: Heroku**
    - **Description**: A cloud platform for deploying and hosting applications with support for PostgreSQL.
    - **Use In**: Hosting the backend API and PostgreSQL database in a cloud environment.

Frontend Technologies

1. **Framework: React with TypeScript**
    - **Description**: A JavaScript library for building user interfaces with component-based architecture, combined with TypeScript for type safety.
    - **Use In**: Build the frontend user interface with reusable components, ensure type safety during development.
2. **State Management: Redux Toolkit**
    - **Description**: A simplified version of Redux for managing global application state.
    - **Use In**: Manage global state (e.g., user authentication, book and highlight data) and handle asynchronous actions.
3. **Routing: React Router**
    - **Description**: A routing library for managing navigation in single-page applications.
    - **Use In**: Enable navigation between different pages in the application (e.g., dashboard, book details, profile).
4. **UI Library: Material-UI and Tailwind CSS**
    - **Description**: Material-UI provides pre-built UI components, and Tailwind CSS is a utility-first CSS framework.
    - **Use In**: Material-UI for pre-built components (e.g., buttons, modals) and Tailwind for rapid utility-based styling.
5. **HTTP Client: Axios**
    - **Description**: A promise-based HTTP client for making API requests.
    - **Use In**: Handle API requests to the backend (e.g., fetching book data, posting highlights).
6. **Form Handling: React Hook Form + Yup for Validation**
    - **Description**: React Hook Form simplifies form state management, and Yup is used for schema-based validation.
    - **Use In**: Manage form submissions (e.g., login, book creation) and validate form data.
7. **Styling: Styled Components and Tailwind CSS**
    - **Description**: Styled Components is a CSS-in-JS library, and Tailwind CSS is a utility-first CSS framework.
    - **Use In**: Tailwind for utility-first styling across the app and Styled Components for dynamic, component-level styling.
8. **No Testing Framework**
    - **Description**: No testing framework has been selected for this iteration.
    - **Use In**: Testing may be added in future versions as needed.
9. **Linting & Formatting: ESLint + Prettier**
    - **Description**: ESLint identifies potential code issues, while Prettier ensures consistent code formatting.
    - **Use In**: Maintain code quality, consistency, and readability across the team.

10. **Environment Variables: dotenv**
    - **Description**: dotenv manages environment variables by loading them from a `.env` file.
    - **Use In**: Store sensitive data like API keys, database URLs, and JWT secrets securely and load them into the application.