Solution Design Document

1. Project Overview

Project Name: Library Management System

Description: This project aims to create a decentralized library management system using Hyperledger Fabric. The system will manage book inventories, lending processes, and user information securely and transparently. The frontend is built with ReactJS, the chaincode with NodeJS, and the network infrastructure is based on Hyperledger Fabric.

2. System Architecture

Components:

1. **ReactJS Frontend**
   - User Interface for interacting with the library system.
   - Sends API requests to the backend.
2. **Backend (NodeJS)**
   - Express Server hosting RESTful APIs.
   - Uses Hyperledger Fabric SDK for blockchain interactions.
3. **Hyperledger Fabric Network**
   - **Channel:** `librarychannel` for transaction communication.
   - **Organizations:** Org1 and Org2, each with their peers and CAs.
   - **Orderer Service:** Orders transactions and forms blocks.
   - **Chaincode:** Implements library management logic.

3. Detailed Architecture

Frontend (ReactJS)

- **Components:** Login, Dashboard, Add Book, Borrow Book, Return Book, Reduce Copies, Remove Book, Renew Book, Add Review, View Books.
- **API Requests:** Communicate with backend endpoints.

Backend (NodeJS)

- **Express Server:** Manages API endpoints for frontend interactions.
- **Fabric SDK Integration:** Handles chaincode invocation and querying.
- **Endpoints:**
  - `/addBook`
  - `/borrowBook`
  - `/returnBook`

    - `/getBooks`
    - `/reduceCopies`
    - `/removeBook`
    - `/renewBook`
    - `/getAvailableBooks`
    - `/getUserBorrowedBooks`
    - `/getTotalNumBooks`
    - `/addReview`

Hyperledger Fabric Network

- **Channel:** `librarychannel`
- **Organizations:**
    - **Org1:**
        - `peer0.org1.example.com`
        - `peer1.org1.example.com`
        - **CA:** Org1 CA
    - **Org2:**
        - `peer0.org2.example.com`
        - `peer1.org2.example.com`
        - **CA:** Org2 CA
- **Orderer:** `orderer.example.com`
- **Chaincode (NodeJS):** Deployed on Org1 and Org2 peers, managing library functions.

Communication Flow:

1. **Frontend to Backend:** HTTP requests from ReactJS to Express Server.
2. **Backend to Hyperledger Fabric:** Backend sends transactions via Fabric SDK.
3. **Peers to Orderer:** Peers send endorsed transactions to the orderer.
4. **Orderer to Peers:** Ordered blocks are distributed back to peers.

4. Sequence Diagrams

Add Book Sequence:

1. User enters book details in the frontend.
2. Frontend sends a POST request to `/addBook`.
3. Backend processes the request and invokes chaincode.
4. Chaincode adds book details to the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the new book.

Borrow Book Sequence:

1. User selects a book to borrow.
2. Frontend sends a POST request to `/borrowBook`.
3. Backend processes the request and invokes chaincode.
4. Chaincode updates book status in the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the borrowed book status.

Return Book Sequence:

1. User selects a book to return.
2. Frontend sends a POST request to `/returnBook`.
3. Backend processes the request and invokes chaincode.
4. Chaincode updates book status in the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the returned book status.

Reduce Copies Sequence:

1. Admin selects a book to reduce copies.
2. Frontend sends a POST request to `/reduceCopies`.
3. Backend processes the request and invokes chaincode.
4. Chaincode updates the number of copies in the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the updated copies.

Remove Book Sequence:

1. Admin selects a book to remove.
2. Frontend sends a POST request to `/removeBook`.
3. Backend processes the request and invokes chaincode.
4. Chaincode removes book details from the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the removed book.

Renew Book Sequence:

1. User selects a book to renew.
2. Frontend sends a POST request to `/renewBook`.
3. Backend processes the request and invokes chaincode.
4. Chaincode updates the book's due date in the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the renewed book status.

Get Available Books Sequence:

1. User requests available books.
2. Frontend sends a GET request to `/getAvailableBooks`.
3. Backend processes the request and queries chaincode.
4. Chaincode retrieves available books from the ledger.
5. Backend sends the list of available books to the frontend.
6. Frontend displays the available books to the user.

Get User Borrowed Books Sequence:

1. User requests their borrowed books.
2. Frontend sends a GET request to `/getUserBorrowedBooks`.
3. Backend processes the request and queries chaincode.
4. Chaincode retrieves user's borrowed books from the ledger.
5. Backend sends the list of borrowed books to the frontend.
6. Frontend displays the borrowed books to the user.

Get Total Number of Books Sequence:

1. Admin requests total number of books.
2. Frontend sends a GET request to `/getTotalNumBooks`.
3. Backend processes the request and queries chaincode.
4. Chaincode retrieves the total number of books from the ledger.
5. Backend sends the total number of books to the frontend.
6. Frontend displays the total number of books.

Add Review Sequence:

1. User adds a review for a book.
2. Frontend sends a POST request to `/addReview`.
3. Backend processes the request and invokes chaincode.
4. Chaincode adds the review to the ledger.
5. Backend sends a success response to the frontend.
6. Frontend updates the UI with the new review.

5. Data Flow Diagrams

Level 0: Context Diagram

- User interacts with the Library Management System.
- System communicates with Hyperledger Fabric network.

Level 1: Detailed Data Flow

- **Frontend:** Handles user input and displays information.
- **Backend:** Processes requests, interacts with the blockchain.
- **Blockchain:** Maintains and validates transactions.

6. Security Considerations

- **Access Control:** Managed by Hyperledger Fabric's certificate authorities (CAs).
- **Data Integrity:** Ensured by the blockchain's immutable ledger.
- **Confidentiality:** Transactions are private to the `librarychannel`.

7. Implementation Plan

1. **Setup Hyperledger Fabric Network**
   - Configure Org1 and Org2.
   - Create and join `librarychannel`.
   - Deploy chaincode on peers.
2. **Develop Backend (NodeJS)**
   - Set up Express Server.
   - Integrate with Fabric SDK.
   - Implement API endpoints.
3. **Develop Frontend (ReactJS)**
   - Design user interface.
   - Implement frontend logic.
   - Connect to backend APIs.
4. **Testing and Deployment**
   - Unit and integration testing.
   - Deploy on a cloud platform (e.g., AWS, Azure).
   -