Architectural Document for BudgetMate

1)Overview This document outlines the architectural design for the Library Management System (LMS), focusing on book cataloging, user management, and transaction tracking. It provides a high-level view of the system components, data flow, and integrations necessary for efficient library operations.

2)System Architecture The Library Management System follows a modular client-server architecture, ensuring scalability and maintainability.

2.1 High-Level Architecture

- **Frontend:** React.js for UI.

● **Backend:** Node.js with Express for API development

- **Database:** MongoDB for storing books, users, and transactions.
- **OCR Service:** Tesseract OCR for receipt scanning.
- **Authentication:** JWT based authentication.

Components and Modules

User Interface (UI) Layer

- **Dashboard**: Displays book availability, user transactions, and notifications
- **Book Search & Filter**: Users can search and filter books by category, author, or availability
- **User Management**: Registration, login, and profile management for students and librarians
- **Transaction History**: Displays borrowed and returned books

Business Logic Layer

- **Book Borrowing & Returning Logic**: Tracks due dates and availability
- **Fine Calculation**: Automatic fine computation for overdue books
- **Reservation System**: Allows users to reserve unavailable books
- **Report Generation**: Generates reports on book usage, fines, and user activity

Data Layer

⌄ **MongoDB Collections**:

- **Users**: Stores user profiles, roles, and authentication details
- **Books**: Stores book details, categories, and availability status
- **Transactions**: Logs book borrowing, returning, and fines
- **Reservations**: Tracks book reservations and priority queues

Third-Party Integrations

- **Payment Gateway (Optional)**: For fine payments
- **Email Service**: Sends overdue notifications and reservation alerts
- **Barcode Scanner API**: Facilitates quick book check-in/check-out

Data Flow

1. User logs in and accesses the library dashboard.
2. User searches for a book and either borrows it or reserves it if unavailable.
3. If borrowed, the system updates the book's status and logs the transaction.
4. If reserved, the system adds the user to the reservation queue.
5. On return, the system updates availability and calculates any fines if overdue.
6. The admin dashboard provides reports on book usage and user activity.

Security Considerations

- **Authentication**: JWT-based user authentication.
- **Encryption**: AES encryption for sensitive user and transaction data.
- **Secure API Calls**: HTTPS for all API communications.
- **Role-Based Access**: Admins, librarians, and users have different access levels

Performance & Scalability

- **Load Balancing**: Nginx or AWS Load Balancer for handling traffic.
- **Caching**: Redis for optimizing frequently accessed data.
- **Scalability**: Horizontal scaling of backend services based on user demand.

Assumptions

- Users have internet access for web-based services.
- Library staff will update book details and availability in the system.
- Users will follow borrowing policies to maintain system efficiency.

RISHI GUPTA (RA2211003010843)

HARDIK BARAK (RA2211003010859)

BARATH VK (RA2211003010853)