

# Requirements Specification

CHURCH LIBRARY

Customer

GLADTIDINGS SOFTWARE LIMITED

SUPPLIER

GROUP C

The delivery comprises

Software and operation for church library

Fredrick Kwaleyela , Davy Mwansa , Paul Mbambo , Praise Mulenga

## Table of Contents

<b>A. Background and Supplier Guide</b>	<b>4</b>
A1. Background and Vision	
A2. Supplier Guide	
A3. Overall Solution and Alternatives	
<b>B. High-Level Demands</b>	<b>8</b>
B1. Flows	
B2. Business Goals	
B3. Early Proof of Concept	
B4. Minimum Requirements & Selection Criteria	
B5. Benefit in Dollar (Net Business Value)	
B6. Selection Criteria: Score Points per Dollar	
<b>C. Tasks to Support</b>	<b>12</b>
Work Area 1: Content Access & Reading	
C1. Browse Church Library Content	
C2. Read Church Material	
C3. Highlight & Annotate Text	
Work Area 2: User Management & Personalization	
C4. Register or Authenticate User	
C5. Manage Reading List or Schedule	
Work Area 3: Content Administration	
C6. Upload New Content	
C7. Edit or Update Existing Content	
Work Area 4: System Monitoring & Configuration	
C8. View and Filter Activity Logs	
Work Area 5: Offline Access Management	
C9. Download Content for Offline Reading	
Work Area 6: Analytics & Insights (Future)	
C10. View Personal Reading Statistics	
Work Area 7: AI-Powered Interactions (Future)	
C11. Ask AI Chatbot a Question	
<b>D. Data to Record</b>	<b>29</b>
D0. Common Fields	
D1. Library Item	
D2. Category	

D3. User	
D5. Review	
<b>E. Other Functional Requirements</b>	<b>32</b>
E1. System Generated Events (Reminders & Notifications)	
E2. Reports (Usage & Analytics)	
E3. Business Rules & Complex Calculations	
E4. Expansion of the System	
<b>F. Integration with External Systems</b>	<b>34</b>
F1. Google & Facebook OAuth	
F2. AI Chatbot API (Future)	
F3. Payment Gateway Integration (Future)	
F4. Cloud Storage Integration (S3 or Equivalent)	
<b>G. Technical IT Architecture</b>	<b>38</b>
G1. Presentation Layer	
G2. Application Layer (Module Responsibilities)	
G3. Domain Layer	
G4. Infrastructure Layer	
G5. Data Access Layer	
G6. External Services	
<b>H. Security</b>	<b>40</b>
H1. Login & Access Rights	
H2. Protection Against Data Loss	
H3. Protection Against Unintended Actions	
H4. Protection Against Threats	
<b>I. Usability and Design</b>	<b>42</b>
I1. Ease-of-Learning & Task Efficiency	
I2. Accessibility & Look-and-Feel (WCAG 2.1 AA)	
<b>J. Other Requirements and Deliverables</b>	<b>48</b>
J1. Other Standards to Obey	
J2. User Training	
J3. Documentation	
J4. Data Conversion	
J5. Installation	
J6. Testing the System	
J7. Phasing Out	

<b>K. The Customer's Deliverables</b>	<b>-</b>
<b>L. Operation, Support, and Maintenance</b>	<b>50</b>
L1. Response Times	
L2. Availability	
L3. Data Storage	
L4. Support	
L5. Maintenance	
<b>M. Traceability Matrix</b>	<b>52</b>

## A. Background and Supplier Guide

### A1. Background and Vision

Currently, the church's library is mostly managed via paper sign-out sheets, spreadsheets, and a basic file-share for digital resources. This manual approach leads to:

- Lost or overdue books with limited tracking
- Time-consuming catalog lookups by staff and members
- Fragmented digital content stored in disparate folders
- No self-service access outside of church hours

The goal is to replace this ad hoc setup with a unified web application that:

1. **Improves Accessibility:** 24/7 online catalog browsing and self-service downloads
2. **Enhances Usability:** Streamlined search, reading, annotation, and scheduling workflows
3. **Supports Staff Efficiency:** Centralized content management, automated notifications, and reporting

Below is a high-level context overview. Double-border boxes and arrows indicate the supplier's delivery scope and integrations to be implemented.

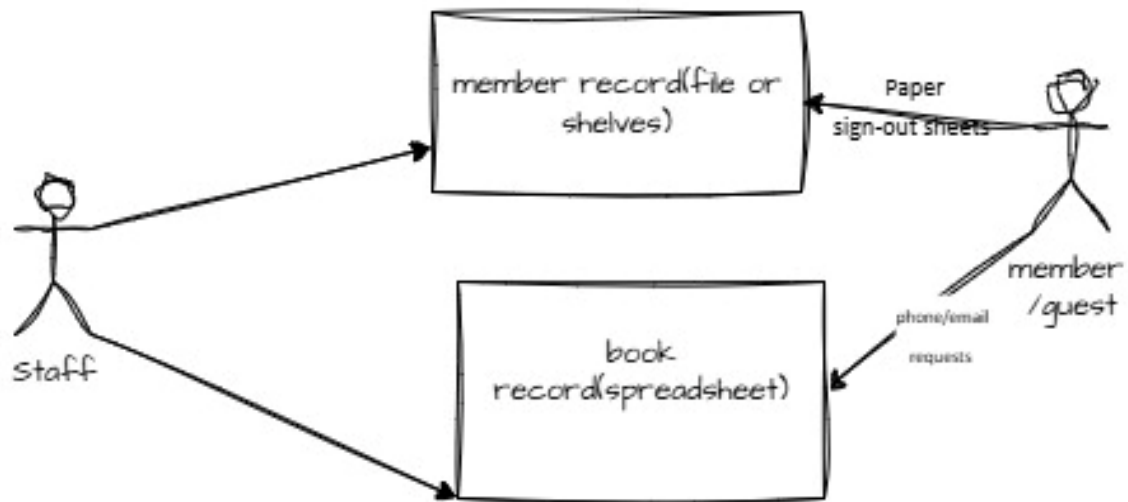
Actors & Systems
• <b>Members / Guests</b> → Paper sign-out sheets & phone/email requests → <b>Library Team</b> • Staff → Spreadsheet & file-share for digital assets → Book Record
Actors & Systems
• <b>Members / Guests</b> → <b>Church Library App</b> ↔ <b>Library Staff</b> → <b>Admin Console</b> → (Manage content, Logs, Reports)

- The **Church Library App** (double-border box) is the supplier's primary deliverable.
- **Integrations to deliver** (double-lined arrows):

This new system will provide a single, cohesive platform for both digital content, enabling both members and staff to complete their tasks efficiently and reliably.

## Current vs visioned system

Current System



Vision for the new system



## A2 Supplier guide

Column 1 shows the customer’s high-level demands (from B2 Business Goals and B1 Flows); Column 2 (in red) is the supplier’s concise proposal for how to meet each; and Column 3 is a shorthand code for tracking and prioritization.

Requirement	Supplier Proposal	Code
<b>BG1. Improve accessibility of church content Make all library materials easily discoverable and usable by members on any device, logged in or guest.</b>	Implement a responsive React Native frontend with full-text Elasticsearch-powered search, offline download support (PDF/EPUB/MP3) and i18n for UI/content metadata.	BG1 P1
<b>BG2. Provide ease of use of system (Usability) Offer an intuitive, streamlined experience so users accomplish common tasks with minimal training.</b>	Design “one-click” workflows in Figma, integrate guided tours, tooltips, and a role-based dashboard using accessible WCAG 2.1 AA components.	BG2 P1
<b>BG3. Efficient support of all user &amp; admin tasks Tasks—from browsing to maintenance—must be quick and reliable without context-switches.</b>	Build a RESTful API (Node.js/Express) with RBAC; React Native UI for U1–U19 flows; server-side pagination & caching; bulk import/export endpoints;	BG3 P1
<b>BG4. Maintainability of the system Platform must be modular, well-documented, and testable for easy updates and extensions.</b>	A layered architecture, document all endpoints via, use version control like git hub to keep track of file changes and collaboration.	BG4 P1
<b>BG5. Low operational costs of the system Minimize hosting, maintenance, and support expenses.</b>	Leverage open-source stack (SQLite, Elasticsearch),	BG5 P1
<b>U-Search: Full-text search &amp; filters Allow filtering by type, language, availability, etc.</b>	Index all content in Elasticsearch; expose faceted search API; integrate React InstantSearch for UI components.	C5-1 P1
<b>U-Annotate: Highlight, notes, bookmarks, Read List and Schedule Enable users to mark and comment on passages.</b>	Use a WebSocket-backed annotation service (e.g. Annotorious) storing metadata in SQLite or Json; UI overlays in the reader component.	C7-2 P2

<b>NFR-Performance: Response times</b> <b>Search &lt; 3 s, page loads &lt; 2 s under</b> <b>100 concurrent users.</b>	Benchmark with Locust; deploy Redis caching for hot data; horizontally scale web tier behind a load-balancer.	L1 P1
<b>NFR-Usability: WCAG 2.1 AA</b> <b>compliance Ensure high-contrast,</b> <b>keyboard nav, ARIA labels.</b>	Use React-Aria and ESLint plugin “jsx-a11y”; conduct axe-core automated scans in CI.	I1 P1



## B. High Level Demands

### B1. Flows

Below are the two high-level flows for the Church Library System

#### Notation:

- **Step** = a logical stage in the overall flow
- **Tasks & Subtasks** = the physical, user-visible operations that fulfill that step
- “(opt)” = optional; “(↺)” = repeatable

#### User Flow: Content Discovery & Interaction

Step	Tasks & Subtasks
1. Access App	• Open mobile app (browse UI loads) (U1)
2. Browse or Sign-In	• Browse anonymously (U2)• Sign in / register (email, password, OAuth) (U3)
3. Language Selection (opt)	• Choose Book language (dropdown or profile setting) (U4)
4. Browse Catalog	• Navigate categories: Books, Bible, Leaflets (U5)• View “New Arrivals,” “Staff Picks,” “Most Read” (U6)
5. Read Content (↺)	• Open reader view for chosen item (U7)• Page-turn or scroll through text (U8)
6. Search Specific Content (opt)	• Enter keyword/phrase (title, verse, topic) (U9)• Apply filters (type, date, language) (U10)• Jump to result (U11)
7. Annotate & Save (opt, ↺)	• Highlight passages (U12)• Write/edit personal notes (U13)• Bookmark or “Save for later” (U14)
8. Download Content (opt, ↺)	• Download Content(Make available offline) (U15)
9. Create Reading Schedule (opt)	• Define schedule (dates, reminders) (U16)• Notification reminders (U17)
10. Request for a Bible Study(opt)	• Fill in request form (U18)

<b>11. Logout / Exit (opt)</b>	• Sign out (U19) / Close App (U20)
--------------------------------	------------------------------------

**Notes:**

- Steps 3, 6–9 are optional—users may skip language selection, searching, annotating, downloading, or scheduling.
- Steps 5 and 7 can repeat any number of times during a session (∞).

**Admin Flow: Content & Log Management**

Step	Tasks & Subtasks
<b>1. Admin Sign-In</b>	• Authenticate via secure login (2FA optional) (A1)
<b>2. Dashboard &amp; Logs</b>	• View system activity logs (A2)• Filter/search logs by user, date, action (A3)• Export or archive logs (A4)
<b>3. Content Upload (opt)</b>	• Upload new Books, Bible editions, Leaflets (A5)• Bulk import via CSV/XML (A6)
<b>4. Content Update (opt)</b>	• Edit metadata (title, author, ISBN, language) (A7)• Replace file / update cover image (A8)
<b>5. Content Deletion (opt)</b>	• Soft-delete or purge items (A9)• Manage status (A10)
<b>6. System Configuration (opt)</b>	• Download limits, UI text & languages (A11)• Manage user roles & permissions (A12)
<b>7. Admin Sign-Out</b>	• Securely log out (A13)

**Notes:**

- Steps 3–6 are optional and may be invoked in any order based on daily operations.
- Viewing logs (Step 2) and content management (Steps 3–5) can repeat as needed.

## B2. Business goals

Business Goal	Solution Vision	Related Requirements	Deadline
<b>BG1. Improve accessibility of church content</b>	Make all library materials (books, Bible, leaflets) easily discoverable and usable by every member of the congregation, whether on desktop or mobile, logged in or guest.	<ul style="list-style-type: none"> <li>• Responsive UI supporting desktop, tablet, mobile (WCAG-inspired) (NFR-Usability)</li> <li>• Anonymous browsing of catalog (U2)</li> <li>• Full-text search across all content types (F-Search)</li> <li>• Downloadable for offline access (F-Download)</li> <li>• content metadata (U4, F-Metadata)</li> </ul>	Launch (4 months)
<b>BG 2. Provide ease of use of system (Usability)</b>	Offer an intuitive, streamlined experience so that both members and staff can accomplish common tasks with minimal clicks and training.	<ul style="list-style-type: none"> <li>• Clean, consistent navigation &amp; clear CTAs (NFR-Usability)</li> <li>• “One-click” borrow/read/download workflows (U7, U15)</li> <li>• In-app help/tooltips and searchable FAQ (F-Help)</li> <li>• Bookmarking, highlighting, and note tools with easy access (U12–U14)</li> <li>• Role-based dashboards tuned to user needs (U5, A2, A6)</li> </ul>	Launch (4 months)
<b>BG 3. Efficient support of all user &amp; admin tasks</b>	Ensure every routine task—from searching maintenance and reporting—can be completed quickly and reliably without switching systems.	<ul style="list-style-type: none"> <li>• RBAC with streamlined user and admin flows (U3, A1, A12)</li> <li>• Fast search responses &lt; 3 s, page loads &lt; 5 s (NFR-Performance)</li> <li>• Bulk import/export for book, bible or leaflet and member data (A6, F-Import)</li> <li>• Automated notifications (due reminders, holds ready) (F-Notifications)</li> <li>• In-app logging &amp; audit trails (A2, A3)</li> </ul>	MVP (4 months)
<b>BG4. Maintainability of the system</b>	Build a modular, well-documented platform that can be easily updated, extended, and tested by the development and operations teams.	<ul style="list-style-type: none"> <li>• API-first architecture with clear</li> <li>• Automated unit, integration, and end-to-end tests (NFR-Quality)</li> <li>• Modular codebase and layered architecture or plugin components (NFR-Maintainability)</li> </ul>	Ongoing; review at 4 months post-launch

<b>BG5. Low operational costs of the system</b>	Keep hosting, maintenance, and support expenses minimal by leveraging cloud services, open-source components, and efficient resource usage.	<ul style="list-style-type: none"> <li>• Auto-scaling infrastructure (NFR-Scalability)</li> <li>• Use of open-source libraries and databases (NFR-Cost)</li> <li>• Automated backups and monitoring to reduce manual overhead (A4, NFR-Reliability)</li> <li>• Budget tracking &amp; alerts in admin dashboard (A11)</li> </ul>	Budget cycle Q3 2025
---	---	---	----------------------

### B3. Early proof of concept

Early Proof-of-Concept plan for the highest-risk aspects of the System

<b>Proof ID</b>	<b>Area / What to Prove</b>	<b>Proof of Concept / Test</b>	<b>Requirements Reference</b>	<b>Deadline</b>
<b>B3-1</b>	UI & Core Content Interaction	Develop clickable prototype (HTML or Figma) of home screen → browse/catalog → reader (with highlight, notes, bookmark) → download → schedule flows. Have 5 representative users “think aloud” while stepping through U1–U19 tasks.	U1–U19 (User Flow steps 1–10), NFR-Usability	93 days
<b>B3-2</b>	Usability End-to-End	Conduct moderated usability test (5–8 members & 2 staff) on prototype screens covering search, read, annotate, download, schedule. Measure task success rate, time on task, SUS ≥75.	All I1 usability requirements (clean nav, tooltips, WCAG AA)	93 days

## B4. Stakeholder Map

The following table identifies key stakeholder groups for the Church Library System, outlining their roles, goals, and how the system is designed to meet their needs.

<b>Stakeholder Group</b>	<b>Roles &amp; Responsibilities</b>	<b>Goals &amp; Pain Points</b>	<b>How the System Addresses Their Needs</b>
<b>Church Members / Registered Users</b>	Read, annotate, download, and manage personal reading lists.	Pain points: Lost/overdue books, time-consuming lookups, fragmented digital content, and no 24/7 access. Goals: Easy access to church materials, spiritual growth, and an intuitive reading experience.	Provides 24/7 online access, streamlined search, reading, and annotation tools. The system also includes features for managing reading lists and schedules.
<b>Library Staff / Admins</b>	Manage content (upload, edit, and categorize), monitor system logs, and configure user permissions.	Pain points: Manual, time-consuming processes with paper sheets and spreadsheets. Goals: Efficiently manage content and user data without switching between systems.	Offers a centralized platform for content administration, automated notifications, and reporting. The admin flow is designed to support quick and reliable task completion.
<b>Guests / Anonymous Users</b>	Browse and preview library content without signing in.	Goals: Easily discover church materials.	The system allows for anonymous Browse of the catalog.
<b>IT Support / System Administrators</b>	Monitor system performance, manage configurations, and oversee security.	Goals: Maintain system stability, security, and a well-documented platform that is easy to test and extend.	The system has a dedicated admin portal with restricted access for viewing logs and managing user roles.
<b>Development Team</b>	Build, maintain, and update the application.	Goals: Create a modular, well-documented platform that can be easily updated, extended, and tested.	The system is being built with a layered, API-first architecture, version control, and automated testing.

# C Tasks to support

## Work Area 1: Content Access & Reading

**Purpose:** Enable church members and guests to browse, search, read, and interact with church materials.

**Primary Users:**

- **Registered Users:** Church members or regular readers with accounts.
- **Anonymous Users:** Visitors who can browse and preview content without logging in.

**Environment:**

- Mobile apps (Android/iOS)
- Web front-end
- Varying device sizes and internet availability

### C1. Browse Church Library Content

**Introduction:**

This task allows users to explore available church content based on categories, topics, languages, or popularity. It begins when the user lands on the content dashboard and ends when they exit browsing or select a content item to read.

**Start:**

- User opens app or web page
- Landing page or search panel loads

**End:**

- User exits browsing
- OR selects an item to read

**Frequency:**

- Per user: Multiple times per session
- Total: Hundreds per day across the user base

**Difficult:**

- When filters don't return expected results or categories are overloaded

**Users:**

- Anonymous users
- Registered users

---

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Display all available content	System fetches paginated list from DB
2. Filter by category (e.g., Prophecy, Sabbath Keeping)	User selects filter from dropdown
3. Filter by language (e.g., English, Bemba)	System shows only language-matched items
4. Display “New Arrivals” and “Most Read”	System calculates based on metadata
5. Allow anonymous browsing	System enables guest access with limited interactivity
6p. Problem: Overloaded or ambiguous categories	Admin reviews category design for better UX
7. View brief description or metadata	System shows title, summary, author, etc.
8. Select an item to read → transitions to Task C2	System passes item ID to reader view

## C2. Read Church Material

**Introduction:**

This task enables users to open a selected material and read it using a built-in reader view. It starts when a user clicks on an item and ends when they exit the reader.

**Start:**

- User selects content from browse or search results

**End:**

- User exits reader view or closes the app

**Frequency:**

- Per user: Once per item per session
- System-wide: High daily use

**Difficult:**

- When long texts are hard to scroll or render on smaller screens

**Users:**

- Registered & anonymous users

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Load content into reader view	System fetches full text from DB or file
2. Render pages or continuous scroll	User can swipe/scroll vertically or horizontally
3. Toggle font size	UI provides accessibility settings
4. Adapt document layout for device size	System uses responsive design (HTML/EPUB conversion)
5p. Problem: Original files are only in PDF and hard to read	System converts PDFs to HTML/EPUB for mobile
6. Exit reader	User taps "Back" or closes app

### C3. Highlight & Annotate Text

**Introduction:**

This task allows users to highlight scripture or passages and attach personal notes. It starts when a user selects a portion of text and ends when they save, cancel, or exit the interaction.

**Start:**

- User taps and selects a passage in reader view



**End:**

- Highlight or note is saved
- OR user cancels or exits

**Frequency:**

- Per user: Occasionally, based on reading patterns

**Difficult:**

- On smaller screens or with long passages

**Users:**

- Registered users only (data tied to user profile)

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Select a text passage	User taps-and-holds or drags over text
2. Choose highlight color	UI shows color palette
3. Add/edit personal note	Text input box appears with save option
4. Save highlight & note to profile	Data is stored in user-specific DB table
5. View existing highlights	Reader loads user annotations per content ID
6p. Problem: Confusion between personal and public notes	System clearly marks annotations as private
7. Remove or edit highlights	User accesses “My Annotations” menu

**Requirement Notes (C3):**

- Notes and highlights should sync with user profiles and be stored securely.
- Anonymous users cannot access this feature.
- Each annotation should be tied to a specific content ID and passage location.

## Work Area 2: User Management & Personalization

**Purpose:** Handle registration, login, authentication, and personal content tracking.

**Primary Users:**

- **New Users:** First-time visitors registering manually or via social login.
- **Returning Users:** Regular readers who manage personal reading preferences.

**Environment:**

- Mobile
  - Secure authentication endpoints
- 

### C4. Register or Authenticate User

**Introduction:**

This task enables users to access protected and personalized features by registering or signing into the system. It begins when a user lands on the authentication screen and ends upon successful login or account creation.

**Start:**

- User clicks “Login” or “Register”

**End:**

- User is authenticated and redirected to dashboard

**Frequency:**

- Per user: 1–2 times per session (rarely repeated after session persists)
- System-wide: Moderate, especially on new sessions/devices

**Difficult:**

- When third-party (Google/Facebook) auth fails or internet is unstable

**Users:**

- All app users requiring personalization (notes, highlights, reading lists)

## Subtasks & Variants

Subtask / Variant / Problem	Example Solution
1. Display login/registration form	System loads login UI with option tabs
2. Enter email/password or choose social login	User fills in credentials or taps Google/Facebook button
3a. Authenticate via email/password	System validates user via secure backend
3b. Authenticate via OAuth (Google/Facebook)	OAuth token exchanged and verified
3p. Problem: Third-party login fails due to network issues	System retries and shows fallback option
4. Redirect user to personalized dashboard	Session is initialized; token is stored securely
5. Log out securely	Token/session is destroyed and user is redirected

## C5. Manage Reading List or Schedule

### Introduction:

This task allows users to create, update, or delete a list of materials they plan to read later or on a schedule. It starts when the user enters the reading list interface and ends when the list is saved, modified, or exited.

### Start:

- User navigates to “My Reading List” or “Reading Plan”

### End:

- List is updated or the user exits

### Frequency:

- Varies by user — typically weekly or bi-weekly interaction
- Important for spiritual growth and planning

**Difficult:**

- When managing a long list across devices or syncing issues occur

**Users:**

- Registered users only

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. View current reading list	System fetches list from user profile
2. Add a new book to the list	User taps “Add to Reading List” on book page
3. Remove or reorder items	Drag/drop interface or delete button
4. Set reading schedule/reminders	User inputs timeframes or selects presets
4a. Sync schedule with calendar (future feature)	System uses Google Calendar API or device-native integration
5p. Problem: User loses list after logout	System persists list in secure cloud storage
6. Save changes	System validates and stores updated list

**Requirement Notes (C5):**

- Reading list should sync across devices using user’s unique ID.
- List should be stored in cloud DB with optional offline caching.
- Reading reminders can be integrated with notification services.

## Work Area 3: Content Administration

**Purpose:** Internal staff upload, manage, categorize, and maintain content in the library system.

**Primary Users:**

- **Library Admins:** Full control over content, users, categories, system settings.
- **Content Managers/Curators:** May have limited permissions to upload/edit.

**Environment:**

- Admin Web Portal
- Office setting with stable internet and full-screen interface

## C6. Upload New Content

**Introduction:**

This task allows admins to upload new books, Bible versions, leaflets, or church documents into the library system. It starts when an admin accesses the upload panel and ends when the new item is successfully published.

**Start:**

- Admin clicks “Upload Content” in backend portal

**End:**

- Content is published and indexed in the library system

**Frequency:**

- Weekly uploads depending on church publishing cadence

**Difficult:**

- When file formats are inconsistent or metadata is incomplete

**Users:**

- Admins
- Content managers

## Subtasks & Variants

Subtask / Variant / Problem	Example Solution
1. Select file to upload (PDF, EPUB, DOCX, etc.)	Admin browses and selects file from device
2. Enter metadata (title, author, date, etc.)	Form includes validations and dropdowns
3. Select category (e.g., Doctrine, History)	Category pulled from predefined or dynamic list
4. Assign language	Admin selects from language dropdown
5. Upload cover image	Image file is added via drag-and-drop or selector
6. Submit and publish item	System processes file and indexes content
6p. Problem: File upload fails due to size or format	System validates file and shows detailed error message
7. Receive confirmation or error feedback	System displays success or error message with details

## Requirement Notes (C6):

- File upload size and format validation must be enforced pre-submission.
- Admins must be able to preview uploaded content before final publishing.
- Cover image should be optional but recommended for UX.

## C7. Edit or Update Existing Content

### Introduction:

This task enables admins to update existing entries in the library — either by changing metadata, replacing content files, or correcting categorization errors. It starts when an item is selected and ends when changes are saved.

### Start:

- Admin searches or navigates to a specific item

**End:**

- Metadata or file is updated and saved

**Frequency:**

- As needed — typically after content audits or correction requests

**Difficult:**

- When data inconsistency or file versioning creates confusion

**Users:**

- Admins

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Search for or browse to content	Admin uses search bar or filters
2. Open metadata editor	System loads form pre-filled with current data
3. Edit fields (title, author, date, etc.)	Admin updates and saves specific fields
4. Replace content file	New version is uploaded, replacing the old one
5. Replace cover image	Updated image is uploaded and previewed
6. Save changes	System validates and applies updates
6p. Problem: Changes not reflected due to caching or sync delays	System clears cache or triggers re-indexing
7. Option to revert to previous version (future enhancement)	Version history or undo capability considered

**Requirement Notes (C7):**

- Edits should be audit-logged for traceability.
- File replacements must preserve item ID and user bookmarks where possible.
- Cover images should have resolution guidelines for quality control.

## Work Area 4: System Monitoring & Configuration

**Purpose:** View logs, adjust system behavior, and manage user permissions and roles.

**Primary Users:**

- **System Administrators**
- **IT Support/Library Supervisor**

**Environment:**

- Admin Portal (Restricted access)
  - Requires audit logging, high-level oversight
- 

## C8. View and Filter Activity Logs

**Introduction:**

This task enables admins to monitor and filter system activity such as user logins, content uploads, and reading activity. It begins when an admin opens the audit log panel and ends after viewing, exporting, or closing the log view.

**Start:**

- Admin accesses “System Logs” section in backend

**End:**

- Admin exits the view or exports results

**Frequency:**

- Periodically (e.g., weekly or monthly) or in response to an issue

**Difficult:**

- When logs are too large or unfiltered



**Users:**

- System administrators
- Library supervisors

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Open system logs dashboard	Admin navigates from backend menu
2. Filter by user, date, or event type	Filters available at top of log view
3. Search by keyword (e.g., “upload”, “delete”)	Text search across logs
4. View details of an individual event	Expandable rows or modal pop-ups
5. Export filtered logs (CSV, PDF)	Export button triggers file download
6p. Problem: Logs load slowly when large	System paginates and compresses data
7. Archive or clear old logs (admin only)	Option to archive by date or size policy

**Requirement Notes (C8):**

- Logs must include timestamp, actor ID, action type, and affected resource.
- Access control: Only users with Admin or Supervisor role should access this module.
- Exported logs must be readable and compatible with compliance reviews

## Work Area 5: Offline Access Management

**Purpose:** Enable offline availability of content, manage downloads and local storage.

**Primary Users:**

- **Mobile App Users** in low-connectivity environments
- **Field Ministry Members** needing access on the go

**Environment:**

- Mobile Devices with SQLite/local storage
- Offline-first design philosophy

---

## C9. Download Content for Offline Reading

### Introduction:

This task allows users to selectively download church materials and store them locally. It begins when a user selects a title for offline use and ends when the file is successfully stored or the user cancels the action.

### Start:

- User taps “Download” on content detail screen

### End:

- File is available offline
- OR download is canceled/failed

### Frequency:

- High in low-bandwidth regions
- Low where connectivity is consistent

### Difficult:

- When files are large or device storage is low

### Users:

- Registered users

### Subtasks & Variants

Subtask / Variant / Problem	Example Solution
1. Tap “Download” on item	System prompts confirmation
2. Verify available device storage	System checks space before download
3. Store file in local SQLite or file storage	Files saved with encrypted path

<b>4. Notify user of success/failure</b>	Toast or modal feedback shown
<b>5. View downloaded content offline</b>	“Offline Library” UI displays locally cached files
<b>6. Manage/delete downloaded items</b>	User selects item > tap delete
<b>7p. Problem: Download interrupted by unstable connection</b>	Retry mechanism + partial download support
<b>8. Pre-filter downloads by language</b>	System only shows items in selected language

## Work Area 6: Analytics & Insights

**Purpose:** Provide individual and global content engagement metrics.

**Primary Users:**

- **Readers:** Want to see personal reading progress
- **Admins:** Need system-wide statistics

**Environment:**

- Reader dashboard, Admin dashboard
  - Connected to backend analytics services
- 

### C10. View Personal Reading Statistics

**Introduction:**

This task allows a user to see their engagement metrics like completed books, reading frequency, bookmarked items, and top-read categories. It begins when a user opens the “My Stats” dashboard and ends when they exit.

**Start:**

- User taps “My Statistics” from profile menu

**End:**

- User exits or finishes reviewing data

**Frequency:**

- Weekly/monthly use by motivated readers

**Difficult:**

- When insights are not visual or easy to interpret

**Users:**

- Registered users only

**Subtasks & Variants**

Subtask / Variant / Problem	Example Solution
1. Load user reading statistics	Backend fetches user activity data
2. Display books read & completed	System shows card/grid list
3. Show highlights/bookmarks count	List or badge count summary
4. Visualize daily/weekly reading trends	Line/bar chart per time window
5p. Problem: Charts are too complex on mobile	Use compact sparkline or toggle chart views
6. Compare personal stats to community average (optional)	Display comparative badge (e.g., “Top 5% Reader”)
7. Navigate to “Most Highlighted” or “In Progress” items	Click-throughs to content

**Requirement Notes (C10):**

- Stats must be calculated from user interaction logs.
- Privacy: All data shown must be visible **only** to the owner.
- Admin stats view is separate and more aggregated (not covered in this task)

## Work Area 7: AI-Powered Interactions (Future)

**Purpose:** Enable users to query church materials via an AI chatbot.

**Primary Users:**

- **Readers** seeking answers to questions about doctrine, policy, etc.
- **Admins** feeding documents and managing bot scope

**Environment:**

- Chatbot interface on web/mobile
  - Requires controlled document embedding
- 

### C11. Ask AI Chatbot a Question

**Introduction:**

This task enables users to query the AI assistant about spiritual matters based solely on church-provided materials. It starts when the user opens the chatbot interface and ends when a meaningful answer is received or the session ends.

**Start:**

- User taps “Ask the Library” or similar entry point

**End:**

- AI responds or user exits the session

**Frequency:**

- On-demand; expected spike during Bible study or group preparation

**Difficult:**

- When user input is vague or documents lack depth

**Users:**

- Registered users

**Subtasks & Variants**

<b>Subtask / Variant / Problem</b>	<b>Example Solution</b>
<b>1. Open chatbot interface</b>	Chat window slides in or loads
<b>2. Enter a question</b>	User types or speaks input
<b>3. System queries AI model scoped to church docs</b>	Embedding/querying powered by vector search (e.g., LangChain)
<b>4. Display AI response with cited sources</b>	AI highlights answer and document references
<b>5p. Problem: AI hallucinates or references unapproved content</b>	Restrict model context to uploaded PDFs only
<b>6. Refine or follow-up the question</b>	Context maintained in ongoing session
<b>7. Exit or clear chat history</b>	Data cleared from device/session

## D. Data to Record

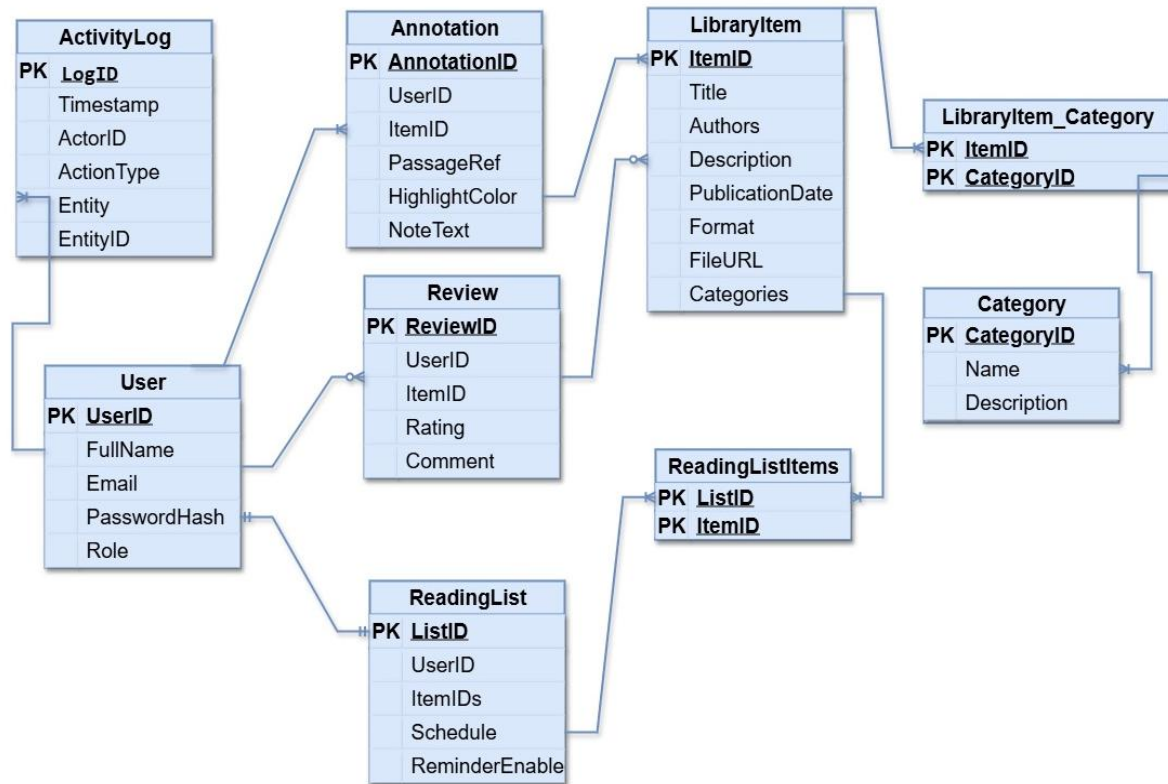
To effectively support library operations, content administration, user interactions, and analytics, the system must persist and manage structured data across multiple entities. This section defines the data model, including all core tables, their attributes, relationships, and usage context. All data can be created, updated, and queried through the functionalities outlined in Chapter C.

### Data Model (E/R)

#### Entities and Relationships:

- **LibraryItem** – Stores metadata and files for physical/digital library content.
- **Category** – Hierarchical tags used to classify items for browsing and filtering.
- **User** – Represents registered library members and administrative roles.
- **Review** – User-generated ratings and feedback on library items.
- **Annotation** – Highlights and notes added by users to specific content.
- **ReadingList** – Personal schedules or lists of materials for future reading.
- **ActivityLog** – System-level audit trail for traceability.

## ERD Diagram:





## D0. Common Fields

Field	Description	Example Solution
<b>ChangeTime</b>	Timestamp of record creation or modification	Auto-filled by system
<b>ChangedBy</b>	Actor (User/System) that triggered the change	User ID or "System"

## D1. LibraryItem

**Represents physical/digital materials in the library.**

**Examples:** "Holy Bible (NIV)", "Early Church History Audiobook"

**Data Source:** Entered by librarians during cataloging, with metadata auto-fetched from ISBN

**Data Use:** Browsing, searching, personalized recommendations

**Data Volume:** ~10,000 records initially, ~500 annually

Field	Description	Example Solution
<b>ItemID</b>	Unique ID (ISBN/UUID)	Auto-generated
<b>Title</b>	Full title (max 200 chars)	Text
<b>Authors</b>	List of contributors	Array of names
<b>Description</b>	Summary or abstract (max 500 chars)	Text
<b>PublicationDate</b>	Year of publication	YYYY format
<b>Format</b>	Type (eBook, PDF, Audiobook, etc.)	Predefined list
<b>FileURL</b>	Link to digital asset (if applicable)	Cloud storage path
<b>Categories</b>	Linked categories (D2 references)	Array of CategoryIDs

## D2. Category

**Hierarchical tags used for thematic classification.**

**Examples:** "Sermons", "Children's Resources", "Theology > Apologetics"

**Data Source:** Defined by admins or imported from known taxonomies

**Data Use:** Filtering, browsing, targeted recommendations

**Data Volume:** ~50 categories

Field	Description	Example Solution
<b>CategoryID</b>	Unique identifier	Auto-increment
<b>Name</b>	Category label (max 50 chars)	Text
<b>Description</b>	Explanation of scope (max 200 chars)	Text

### D3. User

**Registered users of the library system.**

**Examples:** Congregation members, guests, staff

**Data Source:** Self-registration or admin creation

**Data Use:** Authentication, personalization, audit logging

**Data Volume:** ~2,000 users initially; 5–10 added daily

Field	Description	Example Solution
<b>UserID</b>	Unique identifier	UUID or Email
<b>FullName</b>	Full name (max 100 chars)	Text
<b>Email</b>	Contact for notifications	Validated format
<b>PasswordHash</b>	Encrypted credential	Bcrypt or Argon2
<b>Role</b>	Access level (Member, Librarian, Admin)	Predefined roles

## D4. Annotation

**Highlights and notes added by users on specific passages.**

**Data Source:** Submitted by users during reading sessions

**Data Use:** Personal study, synchronized across sessions

**Data Volume:** High volume per user (sync with ReadingList)

Field	Description	Example Solution
<b>AnnotationID</b>	Unique annotation reference	Auto-increment
<b>UserID</b>	Owner of annotation (refers D3)	Foreign Key
<b>ItemID</b>	Item being annotated (refers D1)	Foreign Key
<b>PassageRef</b>	Text location identifier (page, verse, etc.)	Positional reference
<b>HighlightColor</b>	Optional color tag	Hex code
<b>NoteText</b>	Optional user comment (max 300 chars)	Text
<b>CreatedAt</b>	Timestamp	Auto-filled

## D5. Review

**Ratings and feedback tied to content items.**

**Examples:** “Very insightful book”, “Narration was clear”

**Data Source:** Entered through the review interface

**Data Use:** Popularity ranking, community feedback

**Data Volume:** ~5,000 reviews initially; 10–20 new/day

Field	Description	Example Solution
<b>ReviewID</b>	Unique review ID	Auto-increment
<b>UserID</b>	Reviewer (links to D3)	Foreign Key
<b>ItemID</b>	Reviewed content (links to D1)	Foreign Key
<b>Rating</b>	1–5 stars	Integer
<b>Comment</b>	Optional feedback (max 300 chars)	Text
<b>ReviewDate</b>	Date submitted	Auto-filled

## D6. ReadingList

**User-managed reading plans and queues.**

**Data Source:** Created and updated by users

**Data Use:** Scheduling, reminders, and personalized navigation

**Data Volume:** Variable per user

Field	Description	Example Solution
ListID	Unique list reference	Auto-increment
UserID	Owner of list	Foreign Key
ItemIDs	List of items to read	Array of ItemIDs
Schedule	Optional planned dates	ISO date range
ReminderEnabled	Whether notification is active	Boolean

## D7. ActivityLog

**Audit trail for key system operations.**

**Examples:** "User uploaded new item", "Item metadata edited"

**Data Source:** System events

**Data Use:** Monitoring, rollback, compliance

**Data Volume:** High, depends on operations frequency

Field	Description	Example Solution
LogID	Unique log entry	Auto-increment
Timestamp	When the event occurred	System-generated
ActorID	User/system that triggered the action	UserID or "System"
ActionType	What action was performed	Enum (CREATE, DELETE, etc.)
Entity	Target table (User, Item, Review, etc.)	Table name
EntityID	Reference ID of affected record	Primary Key reference
Details	JSON payload of what changed	Structured JSON

## E. Other functional requirements

This section captures functional requirements that go beyond the core user tasks such as login, search, or reading. These requirements define how the system should behave in specific scenarios, support automation, analytics, and align with administrative rules that help enhance user experience and also allow for system scalability.

### E1. System generated events (Reminders & notifications)

System Generated Events: Automatic actions such as notifications, reminders, and alerts based on user behavior or system thresholds (e.g., inactivity or storage limits).

Requirement	Example solutions
<b>E1-1. Remind user to finish a book if unread after certain number of days.</b>	Admin configures the number of days System sends push/email reminders.
<b>E1-2. Notify users when new materials in preferred language/topic is available.</b>	Notifications sent when admin uploads.

### E2. Reports (Usage & analytics)

Reports (Usage & Analytics): Requirements for generating visual or downloadable reports on user activity, material popularity, and administrative oversight.

Requirement	Example solutions
<b>E2-1. Per-user reading statistics.</b>	Dashboard chart; CSV export.
<b>E2-2. Most-read materials overall/per language.</b>	Top 10 report by read-count; PDF & on-screen.

<b>E2-3. Ad-hoc data exploration.</b>	Built-in report builder; export filters.
---------------------------------------	--

### E3. Business rules & complex calculations

Business Rules & Complex Calculations: Logical constraints and calculations that govern system behavior such as progress tracking, access controls, and search result filtering based on user preferences.

Requirement	Example solutions
<b>E3-1. Enforce language and category filter on search results.</b>	
<b>E3-2. Calculate reading progress percentage.</b>	Compute $(\text{pages\_read}/\text{total\_pages}) * 100$ .
<b>E3-3. Limit highlight count per user/material to Y.</b>	Configurable limit; reject excess.

### E4. Expansion of the system

Expansion of the System: Future-facing features that anticipate integration with AI or enhanced interactivity beyond the initial deployment.

Requirement	Example solutions
<b>E4-1. Future AI Q&amp;A integration.</b>	API endpoints for AI service.

## F. Integration with External Systems

This section outlines the necessary integrations between the Church Library System and various external applications or services. Each integration is defined by its purpose, the data exchanged, and any relevant technical considerations or constraints. These integrations are crucial for extending the functionality of the Church Library System and ensuring its seamless operation within the church's existing digital ecosystem.

### F.1. Google and Facebook OAuth

- **Purpose (Problem Solved):**
  - Requiring users to create a new, separate account with a unique password for the Church Library System can be a barrier to adoption and increase password fatigue.
  - Users prefer the convenience and familiarity of signing in with existing social media or email accounts.
- **Data Exchange Details:**
  - **What data is exchanged:** User authentication tokens, basic profile information (i.e., email, user ID, name) upon successful authentication.
  - **Direction of flow:** Unidirectional (authentication from Google/Facebook to Church Library System).
  - **Frequency of exchange:** On-demand (each time a user attempts to sign in via OAuth).
  - **Trigger of exchange:** User selecting "Sign in with Google" or "Sign in with Facebook" option.
- **Interface Specifications/Constraints:**

- Standard OAuth 2.0 protocols will be used for secure authentication.
- Requires registration of the Church Library System application with Google and Facebook developer platforms.

## F.2. AI Chatbot API (Future Integration)

- **Purpose (Problem Solved):**
  - Provides an interactive, AI-powered way to ask questions and receive answers directly from the content within the church library.
  - Manually searching for answers to specific doctrinal or historical questions across multiple documents can be time-consuming for users.
- **Data Exchange Details:**
  - **What data is exchanged:** User queries (text), system context (relevant passages from uploaded church materials), AI-generated responses.
  - **Direction of flow:** Bidirectional (user query to Gemini API, AI response back to user).
  - **Frequency of exchange:** On-demand (whenever a user interacts with the chatbot).
  - **Trigger of exchange:** User initiates a query to the AI chatbot.
- **Interface Specifications/Constraints:**
  - Requires the use of Google Gemini or OpenAI API.
  - The AI chatbot must be strictly confined to the knowledge base of the uploaded church materials to prevent hallucination.
  - Requires uploading of PDF documents and policies into the AI for querying.



### F.3. Payment Gateway Integration (Future Feature)

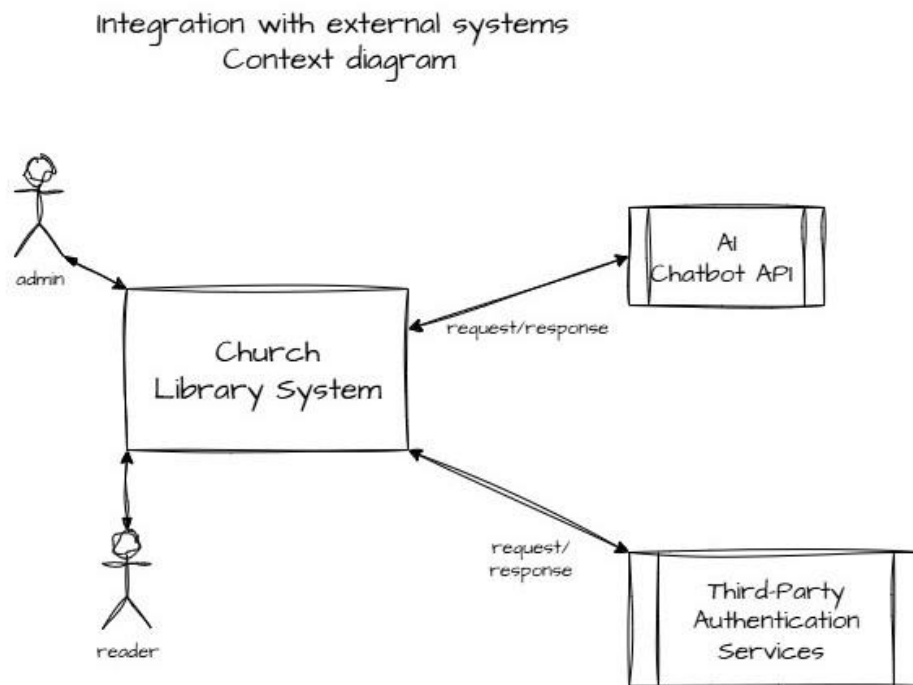
- **Purpose (Problem Solved):**
  - Enables monetization of premium digital library materials.
  - Facilitates secure online payments for users who wish to purchase or access restricted content (e.g., exclusive books, digital archives).
- **Data Exchange Details:**
  - **What data is exchanged:** Payment transaction data (amount, user ID, content ID), payment confirmation tokens, and error/failure responses.
  - **Direction of flow:** Bidirectional (Church Library System sends payment request; gateway returns success/failure).
  - **Frequency of exchange:** On-demand (whenever a user initiates a purchase).
  - **Trigger of exchange:** User clicks "Buy" or "Unlock" on a paid item.
- **Interface Specifications/Constraints:**
  - Integration via secure APIs (e.g., Stripe, PayPal) following PCI-DSS compliance standards.
  - Payment transactions must be verified using secure tokens and receipts.
  - Backend must support payment callbacks and status synchronization.

### F.4. Cloud Storage Integration (Amazon S3 or Equivalent)

- **Purpose (Problem Solved):**
  - Ensures reliable, scalable, and secure storage of digital content (books, leaflets, images) outside of the core application server.
  - Reduces server load and improves performance by offloading large static assets.
- **Data Exchange Details:**
  - **What data is exchanged:** Uploaded files (PDFs, EPUBs, images), metadata (e.g., file path, MIME type), signed URL requests for downloads.
  - **Direction of flow:** Bidirectional (upload from admin to S3; download to user devices).

- **Frequency of exchange:** High — during all upload and download operations.
  - **Trigger of exchange:** Admin uploads a file or user requests to read/download an item.
- **Interface Specifications/Constraints:**
  - Uses AWS S3 REST API or equivalent cloud storage service.
  - File upload/download operations should use pre-signed URLs to ensure security.
  - Storage bucket access must be restricted and encrypted; files should follow a clear naming/versioning convention.

## Context Diagram



## G. Technical IT Architecture

The system must meet performance, scalability, and reliability requirements under the constraints below. Choose **one** alternative based on your operational model.

## Key Modules & Responsibilities

### Presentation Layer

- **Mobile App (React Native)**
  - Browse, read, annotate, download, AI chat
- **Admin Web Portal (React)**
  - Upload/manage content, logs, user roles

### Application Layer

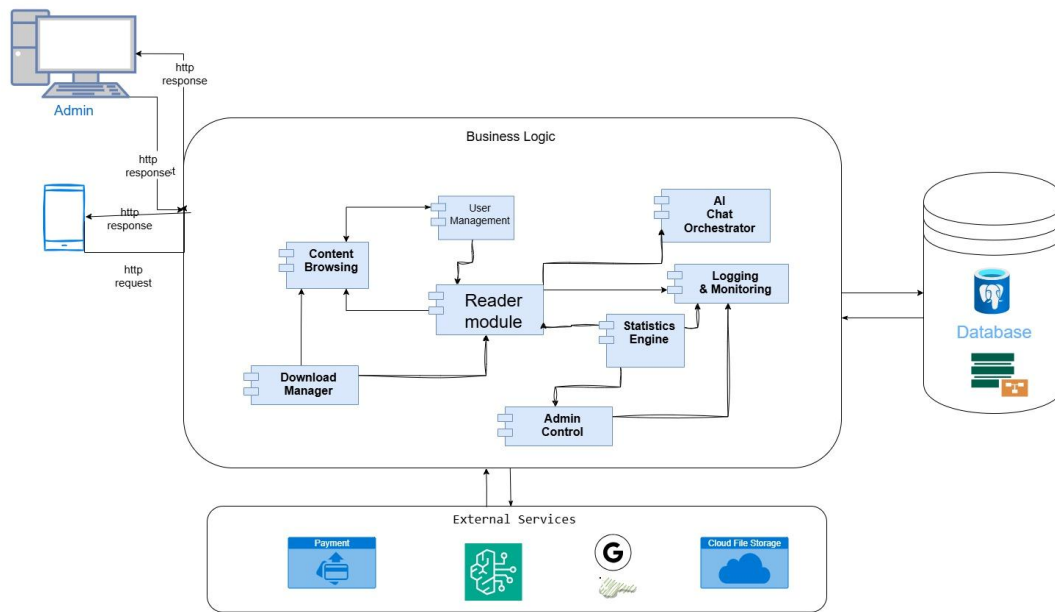
Module	Responsibilities
User Management	Register/login, Google OAuth, session handling
Content Browsing	Category browsing, filtering, search suggestions
Reader Module	Text rendering, highlighting, annotations, bookmarking
Download Manager	Offline caching
Admin Control	Upload, edit, delete content, manage languages/categories
Statistics Engine	Reading logs, personal dashboards
AI Chat Orchestrator	Embeds context, queries AI, returns response with citations
Logging & Monitoring	Tracks user/admin activity, exports system logs

### External Services Integration

Service	Layer	Purpose
Google OAuth	Infrastructure Layer	Sign-in via Google, token exchange
Cloud Storage	Infrastructure Layer	Upload/download PDFs, covers, metadata

<b>AI API</b>	Infrastructure Layer	Answering doctrinal queries based on document embedding
<b>Payment (future)</b>	Infrastructure Layer	Paid content, checkout flow, user entitlements

## Architecture Diagram



## H. Security

Security requirements ensure that only authorized users can access system functionality, and that personal and sensitive content is protected against unauthorized access, loss, or corruption. These measures maintain the integrity and confidentiality of digital library operations, especially

since user activity, highlights, and reading materials may include sensitive or proprietary information.

## H1. Login and access rights for users

Authentication mechanisms (e.g., email/password or Google sign-in), session control, and role-based access enforcement for admins, librarians, and members.

Subtask	Example solutions
H1-1. Email/password authentication.	Login form; password hashing and encryption
H1-2. Optional sign up via Google.	Authentication integration.
H1-3. Session timeout after X period.	"Keep me logged in" option available
H1-4. Enforce password strength.	Client/server validation.

## H2. Protection against data loss

Regular and on-demand backup requirements to safeguard against accidental or malicious data loss.

Requirement	Example solutions
H2-1. monthly automated backup.	
H2-2. On-demand backup trigger.	Admin portal "Backup now" button.

### H3. Protection against unintended actions

Confirmation prompts and soft-delete options that help prevent irreversible or unintended user behavior.

Requirement	Example solutions
H3-1. Confirm delete actions.	Modal confirmation; two-step.
H3-2. Soft-delete from reading list with recycle bin.	Can be restored within 30 day.

### H4. Protection against threats

Technical defenses such as login throttling, enforced HTTPS, and potential future expansion to firewall rules and monitoring.

Requirement	Example solutions
H4-1. Rate-limit login attempts.	IP throttling; captcha.
H4-2. Enforce HTTPS.	

## Part I Usability and design

The Usability & Design requirements (Section I) for the Church Library System tailored for a React-JS frontend.

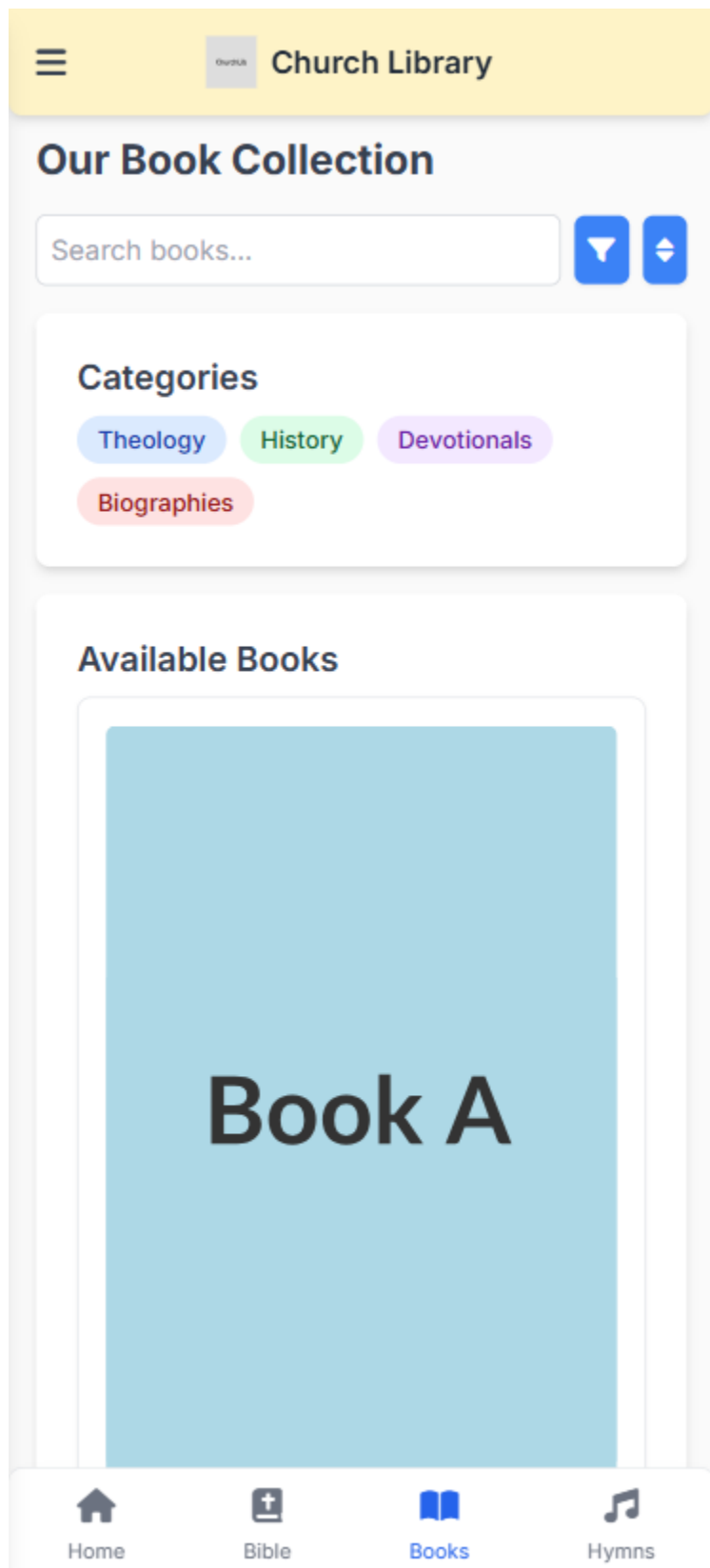
## I1. Ease-of-Learning and Task Efficiency

Requirement (I1-X)	Example Solution	Code
<b>I1-1. Critical usability problems</b> Any <i>critical</i> usability problem (observed in $\geq 2$ users) must be logged, treated as a system error, and fixed according to defect-priority rules.	Conduct early think-aloud tests on Figma/React mockups of key flows (browse, search, read, annotate). Route each critical issue into the support ticket system (e.g. Jira) for triage and repair before development continues.	I1-1 P1
<b>I1-2. Early usability testing</b> UI must be usability-tested and all critical problems found must be resolved before coding begins.	Assemble 5 representative users; run moderated tests on React mockups covering U1–U19 flows; iterate designs until zero critical issues remain.	I1-2 P1
<b>I1-3. Post-training competence</b> After a 1-hour super-user demo, $\geq 90$ % of ordinary users must complete all Chapter C tasks (browse, search, annotate, download) with $\leq 2$ critical issues in testing.	Deliver a 60-min interactive tutorial and verify via follow-up sessions that users can perform U1–U19 tasks; cap at 2 critical usability problems per 10 users.	I1-3 P2
<b>I1-4. Error-message clarity</b> During usability tests, when shown any error message, $\geq 85$ % of users must correctly interpret its meaning and next step.	Curate all reader and system errors; display them to users in tests; collect explanations and refine text. Use React-Intl for contextual messaging.	I1-4 P2
<b>I1-6. Super-user trainability</b> A designated super-user (staff librarian) must be able to master all admin functions (A1–A13) and train others within 2 days.	Provide a 2-day instructor-led workshop and interactive “Admin Guide” built in Storybook; track completion with quizzes.	I1-6 P2
<b>I1-7. Frequent-user efficiency</b> After 1 week of regular use, a member must be able to search, highlight, bookmark, and download 5 items in $\leq 5$ minutes.		I1-7 P2





## Mock up screens





## Welcome Home!



### Featured Content

This section will display new arrivals, popular books, and staff picks. Imagine a carousel of book covers here.

**Book 1**

Book Title 1

**Book 2**

Book Title 2

**Book 3**

Book Title 3



Home



Bible



Books



Hymns



Church

Church Library

## Hymns & Songs



### Popular Hymns

Amazing Grace



How Great Thou Art



It Is Well With My Soul



### Song Categories

Worship

Praise

Children's Songs



Home



Bible



Books



Hymns



Church

## The Holy Bible

### Read Scripture

This section will allow users to select books, chapters, and verses to read. It will also include search functionality specific to the Bible text.

Genesis



Chapter 1



Bible text will appear here...

### Bible Study Tools

- [Commentaries](#)
- [Concordance](#)



Home



Bible



Books



Hymns

## J. Other Requirements and Deliverables

### J1. Documentation

Requirements	Example Solutions	Code
<b>1. Before system delivery, course material must be made available for super users to use in onboarding others.</b>	Delivered in editable PowerPoint and PDF format.	J1-1
<b>2. A month after delivery, user-facing documentation for mobile and admin workflows must be provided.</b>	Includes annotated screenshots and short how-to guides.	J1-2
<b>3. Basic IT documentation for server setup and content management must be provided at delivery.</b>	Includes environment variables, database schema, and backup strategy.	J1-3
<b>4. All documentation must be in electronic form and editable by the customer.</b>	Delivered via shared folder (PDF + editable docs).	J1-4

### J2. Installation

Requirements	Example Solutions	Code
<b>1. The supplier must install the backend, frontend, and database on the chosen cloud environment.</b>	Target stack: Node.js, PostgreSQL, and AWS S3.	J2-1
<b>2. The supplier must deploy the initial mobile app build to Play Store</b>		J2-2

### J3. Testing the System

Requirements	Example Solutions	Code
<b>1. The customer wants to review supplier's test coverage prior to delivery.</b>	Supplier provides test case index and test plan document.	J3-1
<b>2. Regression testing must be performed after major updates.</b>		J3-2
<b>3. Customer will conduct final acceptance testing with a test build and real content.</b>	Supplier provides a UAT environment with seeded data.	J3-3

## L. Operation, support, and maintenance

This section outlines non-functional requirements related to the system's runtime behavior, performance expectations, support structure, and long-term maintainability. These are critical to ensuring the system remains reliable, responsive, and sustainable over time, especially when serving a large church community with varied digital literacy levels.

### L1. Response times

The system's performance benchmarks under typical usage conditions, including page loads, content streaming, and UI responsiveness.

Task	Response time
Search query	$\leq 3$ s for 95%
Material load	$\leq 20$ s for 98%
Reading mode scroll behavior	$\leq 0.2$ s for 98%

### L2. Availability

Required uptime guarantees during business hours and off-hours to ensure uninterrupted access to resources.

Period	Availability
Weekdays 24/7	$\geq 99.5\%$

### L3. Data storage

Retention policies for uploaded materials and mechanisms to archive or restore content as needed.

Requirement	Example solutions
Active materials retained 5 years.	DB & object store retention policies.



Archive older materials	
-------------------------	--

## L4. Support

Levels of support responsiveness for user queries or system issues, and communication channels available

Requirement	Example solutions
L4-1. First-level support response $\leq 4h$ .	
L4-2. Second-level support $\leq 24h$ .	Escalation procedure.

## M.Requirement Traceability Matrix

Req ID	Short Description	BG1	BG2	BG3	BG4	BG5
C1	Browse Church Library Content	X	X	X		
C2	Read Church Material	X	X	X		
C3	Highlight & Annotate Text		X	X		
C4	Register/Authenticate User	X		X		
C5	Manage Reading List or Schedule		X	X		
C6	Upload New Content			X	X	
C7	Edit/Update Existing Content			X	X	
C8	View and Filter Activity Logs			X	X	
C9	Download Content for Offline Reading	X	X	X		
C10	View Personal Reading Statistics		X			
C11	Ask AI Chatbot a Question	X	X	X		
E1-1	Remind user to finish a book after X days		X	X		
E2-1	Per-user reading statistics report		X	X		
E2-2	Most-read materials overall		X	X		
E3-2	Calculate reading progress percentage		X	X		
E4-1	Future AI Q&A integration ready	X	X	X	X	
F1	Google & Facebook OAuth	X		X		
F2	AI Chatbot API	X	X	X		
F3	Payment Gateway Integration			X		
F4	Cloud Storage Integration	X		X	X	X
H1-1	Email/password & Google sign-in	X		X		
H2-1	Monthly automated backup			X		

<b>H3-1</b>	Confirm delete actions		X	X		
<b>H4-1</b>	Rate-limit login attempts			X		
<b>I1-2</b>	Early usability testing on mockups		X		X	
<b>I1-4</b>	Error message clarity $\geq 85$ % interpretability		X		X	
<b>I2</b>	WCAG 2.1 AA compliance	X	X			
<b>J1-1</b>	Data protection & privacy best practices	X		X		
<b>J2-1</b>	Train 10 super users		X			
<b>J3-2</b>	User-facing documentation delivered 1 mo post-go-live		X			
<b>J5-1</b>	Install on chosen cloud environment			X	X	
<b>J6-2</b>	Regression testing in CI/CD			X	X	
<b>L1</b>	Response times: Search $\leq 3$ s, pages $\leq 2$ s			X		
<b>L2</b>	$\geq 99.5\%$ availability 24/7			X		
<b>L4-1</b>	1st-level support $\leq 4$ h			X		X