Here is the full, comprehensive report for your project. This document is structured to be easily copied into Google Docs for your team's future reference.

Project Blueprint: Multi-Tenant Church Management Platform

(Cross-Platform: Web, Android, & iOS)

1. Executive Summary

This document outlines the strategic plan and technology stack for building a multi-tenant, cross-platform application designed to serve multiple churches from a single codebase. The primary goal is to provide a comprehensive management solution for church administrators and a rich engagement hub for members.

The technical architecture is specifically chosen to be "vibe code" friendly (rapid development) and extremely cost-efficient, leveraging a 100% serverless stack built primarily on Cloudflare's free-tier services. The frontend will be developed once using React Native (Expo) to deploy native applications for iOS, Android, and a fully-featured web app.

2. Target Audience

The platform serves four distinct user groups across two main categories:

- **Platform "Managers" (The Customer):**
  - **Church Admins:** Staff responsible for their specific church's operational data. They manage the church's brand (logo, details) and user permissions.
  - **Pastors & Ministry Leaders:** The primary content creators. They manage spiritual content, events, announcements, and member engagement.
- **Platform "Consumers" (The End-User):**
  - **Church Members:** The general congregation. They use the app to stay informed, connect, RSVP for events, access resources, and give.
  - **New Visitors:** Prospective members using the app to learn about the church, find service times, and see upcoming events.

3. Role-Based Access Control (RBAC)

To ensure strict data separation and proper permissions (multi-tenancy), the system will use four distinct roles.

Role	Scope	Key Permissions
Super Admin (You)	**Platform-Wide**	• Manages the creation and deactivation of *Church* accounts. • Accesses platform-wide analytics and billing. • Cannot see or edit the member data of any individual church.
Church Admin	**Single Church**	• **Full control over their church.** • Manages church profile: logo, name, address, service times. • Manages users: Can assign/revoke Pastor & Admin roles. • All permissions of a Pastor.
Pastor (Content Manager)	**Single Church**	• **Manages content and engagement.** • Creates/Edits:

		Announcements, Events, Bible Studies, Sermons. • Manages: Prayer Wall (moderation), Event RSVPs. • Manages members (add/edit/remove). • *Cannot* change the main church logo or details.
User (Member)	**Single Church**	• **Consumes and interacts.** • Views all content (events, sermons, announcements). • Interacts: RSVPs to events, submits prayer requests, gives online. • Manages their own personal profile and privacy settings.

4. Comprehensive Feature List

Module 1: Platform & Church Administration

- **Multi-Tenant System:** A single Super Admin dashboard to onboard new churches.
- **Church Profile Management (Admin):** Each Church Admin can edit their church's:
    - Church Name & Logo
    - Address, Phone, Email
    - Service Times
    - Welcome Message & Social Media Links

Module 2: Member & Group Management

- **Member Directory (Admin/Pastor):** A searchable directory of all members.
- **User Management (Admin):** Ability to add members manually, approve new sign-ups, and assign roles.
- **Small Groups:** Create sub-groups (e.g., "Youth Ministry," "Choir," "Welcome Team") and assign members to them for targeted communication.

Module 3: Content & Engagement

- **Announcements:**
  - (Pastor/Admin) Create, schedule, and publish announcements.
  - (User) Receive announcements via a "feed" and as **push notifications**.
- **Events Calendar:**
  - (Pastor/Admin) Create single or recurring events with details (image, date, time, location, map link).
  - (User) View events in a calendar or list view.
  - (User) RSVP for events.
  - (Pastor/Admin) View and manage the attendee list.
- **Prayer Notes (Prayer Wall):**
  - (User) Submit prayer requests (with an option for anonymity).
  - (All) View the prayer wall and click an "I prayed for this" button.
  - (Pastor/Admin) Ability to moderate and remove submissions.
- **Bible Study / Resources:**
  - (Pastor/Admin) A section to create study series, write devotionals, and upload resources (PDFs, notes).

Module 4: Value-Add Modules

- **Sermon Manager:**
  - (Pastor/Admin) Upload audio files (MP3) or link to video (YouTube/Vimeo).
  - (User) Browse, listen to, or watch past sermons.
- **Online Giving:**
  - Integration with a payment processor (e.g., Stripe) for one-time and recurring donations.
- **Volunteer Management:**
  - (Pastor/Admin) Post volunteer needs for specific events or roles.
  - (User) Sign up to volunteer for opportunities.

5. UI/UX Strategy

The design philosophy will be **clean, simple, and accessible**. The primary goal is ease of use for a non-technical audience of all ages.

- **Core Principle:** "Write Once, Run Everywhere." The UI will be built using **React Native + Expo**, which compiles the same JavaScript/TypeScript code into:
  1. A native iOS App
  2. A native Android App
  3. A progressive web app (PWA) for browsers
- **Key Screens & User Flow:**
  1. **Onboarding:** A simple flow for the user to:
     - Find and select their church from a searchable list.
     - Log in or sign up for *that specific church's* community.
  2. **Main Dashboard (Bottom Tab Navigation):** A simple, 5-tab navigation bar for users.
     - **Home:** Dashboard with the next event, latest announcement, and featured sermon.
     - **Events:** The full event calendar and list.
     - **Sermons/Study:** The media and resource library.
     - **Prayer:** The prayer wall.
     - **More/Profile:** Access to their profile, settings, giving, and the "Admin" section (if applicable).
  3. **Admin Panel:** A separate, simple interface (accessible from the "More" tab for Admins/Pastors) with clear buttons: "Create Event," "Post Announcement," "Manage Members," etc.
- **Key UI Elements:**
  - **Cards:** The primary component for displaying events, announcements, and sermons.
  - **Clear CTAs (Call to Action):** Large, obvious buttons for "RSVP," "Give Now," "Submit Prayer."
  - **Forms:** Simple, single-column forms for all inputs.

6. Recommended Technology Stack (Cost-Efficient & Serverless)

This stack is designed to operate almost entirely on Cloudflare's generous free tier, minimizing server costs.

Component	**Technology**	**Rationale (Why)**
Frontend (Cross-Platform)	**React Native (with Expo)**	**The "Vibe Code" Core.** You write one JavaScript/TypeScript codebase. Expo builds it for native iOS, native Android, and Web. This is the fastest way to support all 3 platforms.
Frontend Hosting (Web)	**Cloudflare Pages**	**Cost: $0.** Your React Native *web app* is deployed here. It's a free, lightning-fast global CDN with automatic CI/CD (deploys when you push to GitHub).
Backend API	**Cloudflare Workers**	**Cost: $0 (Free Tier).** This is your server. It's serverless, meaning you don't pay for idle time. The free tier (100,000 requests/day) is more than enough to start. Use the **Hono** framework for fast API building.
Database	**Cloudflare D1**	**Cost: $0 (Free Tier).** A serverless SQL database. The free tier includes *billions* of row reads and *millions* of writes per month. It's perfect for structured data like users, events, and announcements.

File/Media Storage	**Cloudflare R2**	**Cost: $0 (Free Tier).** This is the biggest money-saver. R2 is an object store (like AWS S3) for logos, sermon audio (MP3s), and PDFs. It has **$0 egress fees**, meaning you are not charged when users view/download files. 10GB of storage is free.
Authentication	**Lucia Auth** or **@hono/auth-js**	**Cost: $0.** These are lightweight, open-source auth libraries that run inside your Cloudflare Worker. You avoid paying for expensive third-party services like Auth0 by managing your own user sessions in your D1 database.
Push Notifications	**OneSignal**	**Cost: $0 (Free Tier).** The one external service. OneSignal has a powerful free tier and handles the complexity of sending push notifications to iOS and Android devices. Your Cloudflare Worker will call the OneSignal API when a Pastor posts an announcement.

7. Multi-Tenancy Architecture Model

To keep church data separate, we will use a **Database-level isolation** model:

- **Church Table:** A top-level table that stores each church's info (ID, name, logo_url).
- **Discriminator Column:** Every other table in the D1 database (e.g., Users, Events, Announcements) will have a church_id column.
- **API Logic:** Every request that comes into the Cloudflare Worker API will be authenticated. The user's session will identify which church_id they belong to. All database queries will be filtered by this church_id.
    - **Example Query:** SELECT * FROM Events WHERE church_id = ? AND event_date > ?
    - This simple logic, enforced at the API (Worker) level, ensures that Church A can *never* see data from Church B.