KIKONIC

Faith.Online- Development Plan & Module Breakdown

PREPARED BY: IKONIC DEV LLC
PREPARED FOR: MIKE TURNER
August 4, 2025

Faith.online – Project Estimate

Tech Stack Strategy

To deliver a scalable, secure, and maintainable MVP for **Faith.online**, the following stack is recommended:

Layer	Technology	Rationale
Frontend	Next.js + TypeScript	Enables SEO, SSR/SSG for faith pages and feeds; powerful with TypeScript
Backend	NestJS (Node.js)	Modular architecture, scalable services, great with domain-driven design
Database	MongoDB + Mongoose	Schema-flexible for spiritual posts, user data, moderation events
Auth	JWT + RBAC	Secure token-based authentication with dynamic role-based access

Design Thinking and UX Strategy

This phase is foundational — we start by understanding our end users and create interfaces that guide them to spiritual connection, expression, and community.

Goals:

- Ensure **ease of use** for believers, churches, and creators.
- Reduce friction in content creation, browsing, and communication.
- Maintain faith-centric, clean, uplifting visuals.

UX Approach:

- Research faith-based user personas.
- Map user journeys (connecting, posting, creating events, moderation).

- Design for accessibility and inclusiveness across age groups.

UI System:

- Reusable, responsive component library (cards, modals, navs, calendars).
- Faith-themed visual design with customizable elements for branding.
- Modular structure: faith pages, feeds, messaging, moderation tools.

Phase 1: Infrastructure & Architecture Setup

Approach:

- Configure environment variables, code linting, CI/CD hooks (if needed).
- Define consistent folder structure for maintainability.

Authentication & Roles:

- JWT-based login system with refresh tokens.
- Define roles: `believer`, `creator`, `admin`, `superadmin`.
- Middleware-based access control with route guards.

Data Modeling:

- MongoDB schemas designed for extensibility:
    - **Users** – roles, profile, connections.
    - **FaithPages** – org-level profiles for churches, creators.
    - **Posts** – text/image content with tags, reactions.
    - **Reactions, Comments** – amen/bless logic, threaded replies.

    - **Moderation** – flag queues, ban records.
    - **Events** – recurring spiritual activities or group meetings.

Phase 2: Core MVP Modules Development

Faith Directory

- Churches and creators can build **FaithPages**.
- Search and filtering by name, category (e.g., youth, worship).
- Users can **claim pages**, triggering a verification flow.

Rhythm Keeper

- Calendar for personal or group faith events.
- Habit tracker for prayer, study, spiritual routines.
- Shared tools for group/family planning.

Social Feed & Posting

- Believers post testimonies, images, praise reports.
- "Amen" and "Bless" reactions only — encourages positive dialogue.
- Comment threads with moderation and reply nesting.
- Algorithmic or chronological **community feed**.

Connections System

- Believers and orgs can **connect** or follow each other.
- View **My Network** to browse one's active faith community.

Moderation Strategy

- **Manual and auto-flagging** of harmful content.
- Admin **moderation queue** with decision logging.
- AI-powered **word filter** to reduce profanity/politics.

Daily Word & Christian News

- Display a **daily rotating Bible verse** (from API or internal DB).
- Fetch **faith-based news articles** via RSS or headless CMS.

Messaging System

- Real-time 1:1 DMs with spiritually-aligned language and tone.
- WebSocket integration for responsiveness.
- Spam/profanity filters built-in for user safety.

Admin Oversight Tools

- Admindashboard for:
    - Role management.
    - Content moderation (flagged posts, user bans).
    - Metrics: post volume, report stats, growth indicators.

White-Labeling + Personalization

Theme & Branding System

- Allow each FaithPage to customize:

    - Theme colors
    - Fonts
    - Logos
- Save/load themes from MongoDB for dynamic branding output.

Progressive Web App (PWA) Optimization

Mobile First Experience

- Fully responsive design with mobile-optimized UX.
- Installable PWA with offline support via service worker.
- Performance tuning using Lighthouse metrics and caching strategies.

Testing, Rollout, and Iteration Strategy

Testing Scope

- Component-level unit testing (UI & logic).
- API-level integration tests with mocked DB or in-memory stores.
- User testing loops — alpha, beta, post-launch.

Pilot Rollout

- Closed beta with selected churches and faith leaders.
- Collect feedback and behavior analytics.
- Iterate before open public MVP release.

Final Polish & Refinement

- Polish typography, spacing, micro-interactions.
- Load testing, rate limit enforcement.
- Final QA and bug resolution.