# **🚀 DevOps & Deployment Plan**

**Project Name:** Defeah Bible Study  
 **Version:** 1.0  
 **Date:** July 2025

## **1. 🔁 CI/CD Strategy**

| **Stage** | **Tool** | **Purpose** |
| --- | --- | --- |
| Code Hosting | GitHub | Main repo for Flutter + Edge functions |
| CI/CD for Flutter | GitHub Actions + Flutter SDK | Automate build, test, lint for Android/iOS/Web |
| Edge Function Deploy | Supabase CLI or GitHub Actions | On commit to main, auto-deploy edge function |
| App Store Deployments | Manual (initially) | Google Play & Apple TestFlight via Flutter build |

✅ **CI Jobs**:

* flutter analyze
* flutter test
* supabase functions deploy --project defeah-app
* Auto-deploy web build to Firebase/Supabase if applicable

**📢 CI Notifications:**

* Configure Slack or email alerts for failed builds, test failures, and deployment issues.
* Optional: Enable GitHub Checks for visible CI status per PR.

## **2. 🧪 Environments**

| **Environment** | **Purpose** | **Infra** |
| --- | --- | --- |
| Development | Local + Edge testing | Supabase local emulator + Firebase Emulator |
| Staging | Internal QA (optional) | Supabase staging project + TestFlight / Internal APK |
| Production | Public app users | Supabase (live DB + API), OpenAI/Claude, Firebase Hosting |

* Use Supabase project-level environments for DB and API key separation.
* .env and GitHub Secrets should be environment-specific.

## **3. 📊 Monitoring & Logging**

| **Layer** | **Tool** | **What it Monitors** |
| --- | --- | --- |
| Flutter (Mobile/Web) | Crashlytics / Sentry | App crashes, performance, user-impacting errors |
| Supabase Edge | Supabase Logs | API errors, usage stats, function failures |
| API Rate Limits | Supabase RLS or Cloudflare | Abuse prevention & throttling |
| LLM Costs | OpenAI/Anthropic billing | Monitor token usage, flag cost overages |

✅ **Error Escalation:**

* Critical crash/errors: Notify dev team via Slack/Email.
* Set up budget alerts for LLM usage thresholds.

## **4. 📦 Infrastructure Deployment Flow**

Dev pushes → GitHub main

↓

GitHub Actions

→ Run tests

→ Deploy Edge Function to Supabase

→ (Optional) Deploy web build to Firebase

↓

Mobile builds → Local/CI → Upload to Google Play/TestFlight

**🔜 Future Automation:**

* Migrate manual mobile build flow to GitHub Actions or Codemagic.
* Use fastlane for version bumping and upload.

## **5. 🔧 Secrets & API Keys**

Use GitHub Secrets or Supabase Secrets CLI to store:

* OPENAI\_API\_KEY (backend only)
* FIREBASE\_PROJECT\_ID
* SUPABASE\_ANON\_KEY / SUPABASE\_SERVICE\_ROLE\_KEY

🔐 **Best Practices**:

* Never commit secrets to Git.
* Use flutter\_dotenv for local frontend key access.
* Rotate keys every 90 days.

## **6. 💵 Estimated Cost Breakdown**

| **Service** | **Plan** | **Est. Monthly Cost** |
| --- | --- | --- |
| Supabase | Free/$25 | DB + Auth + Logs + Edge functions |
| OpenAI GPT-3.5/Claude Haiku | Pay-as-you-go | $10–15 for ~1,000+ study queries |
| Firebase Auth/Hosting | Free tier | Supports 10K users + static hosting |
| GitHub Actions | Free tier | Includes 2,000 CI minutes per month |
| Crashlytics | Free | Built-in with Firebase |

## **🧾 Compliance Notes (for Razorpay / Google Pay Integration)**

* ✅ **PCI-DSS Scope**: Razorpay handles all PCI compliance.
* ❌ App does **not** store or process card details directly.
* 🔐 Use **Razorpay Checkout** for secure donation flow (₹100 default).
* 📜 Update privacy policy to list Razorpay/Google Pay as processors.
* 🌐 Consider fallback message/UI if Razorpay fails or is blocked regionally.