

## THE INDIE QUILL – ARCHITECTURE UPDATE SUMMARY

This document records the precise architectural updates made for both production systems:

The Indie Quill Bookstore and The Indie Quill Collective.

---

### 1. BOOKSTORE – ARCHITECTURE CHANGES

---

- Production hosting moved from Replit Deployments to a dedicated Render Web Service.
- Connected backend to Neon Postgres using DATABASE\_URL stored in Render secrets.
- Synced schema fully using Drizzle ORM:
  - `npx drizzle-kit push --config=drizzle.config.ts`
- Verified Neon DB is authoritative for production (schema reports “No changes detected”).
- Backend and frontend now run in a single Node Web Service on Render.
- Implemented stable health endpoints (`/health`).
- Deployment pipeline standardized:  
Replit (development) → GitHub → Render (manual deploy)
- Confirmed Bookstore production API operational:
  - `/api/books`
  - `/api/authors`
- Verified author visibility and initial book ingestion from Neon.

---

### 2. COLLECTIVE – ARCHITECTURE CHANGES

---

- Migrated Collective backend from Replit to Render Web Service.
- Session system fixed using:
  - `connect-pg-simple`
  - Neon-backed session table
  - `createTableIfMissing: true`
- Successfully created and synced the "session" table in Neon.
- Resolved Express routing issues:

- Root route no longer overrides SPA rendering
- Error middleware repaired
- Entire NPO workflow restored:
  - User registration
  - Application creation
  - Admin dashboard (Applications, Users, Stats)
- Frontend hosted as its own Render Node Web Service (not static).
- Updated required environment variables:
  - DATABASE\_URL
  - SESSION\_SECRET
  - VITE\_API\_BASE\_URL
- Resolved all “Not authenticated” and 401/403 issues.
- Neon schema synced using Drizzle; all tables now exist and match Replit dev.

---

### 3. SUMMARY

---

Both systems now operate under a unified, stable architecture:

- Replit → Development only
- GitHub → Source of truth
- Render → Production hosting (Web Services)
- Neon → Production Postgres
- Drizzle → Schema management and migrations
- Unified deploy workflow across both apps

This document should be stored in each repository so the infrastructure is clearly understood by future developers or automated assistants.