Luckypunt Dev Workflow Summary (Updated)

Terminology

- ProjectA: Main app (live), GitHub branch 'main', connected to Supabase DB A
- ProjectB: Dev version, on branch 'dev', connected to Supabase DB B (clone of DB A)

Workflow Phases

Phase 1 - Active Development & Big DB Changes

1. Create a new GitHub branch 'dev' from 'main'
2. In Supabase, go to ProjectA → Database → Backups → Restore to New Project
3. Name the new project (e.g., luckypunt_dev)
4. Supabase clones the full DB before initializing system schemas (no restore conflicts)
5. Update .env in the dev app with new keys (URL, anon key, service role key, connection string)
6. Test and develop on dev app using cloned DB

Phase 2 - Stable App & Minor Features

- Use feature branches off 'main'
- Use Supabase migrations to keep DB in sync
- Only merge and deploy when tested

.env Setup for Dev App

```
SUPABASE_URL=https://<new-project-ref>.supabase.co
SUPABASE_ANON_KEY=<new-anon-key>
SUPABASE_SERVICE_ROLE_KEY=<new-service-role-key>
DATABASE_URL=postgresql://postgres:<project-ref>:<password>@<host>.supabase.com:5432/postgres
```

Important Notes

- Do NOT use pg_dump/pg_restore to clone Supabase DBs (causes schema ownership issues)
- Supabase system schemas (auth, storage) cannot be overwritten via manual restore
- Use the Dashboard 'Restore to New Project' for reliable, full cloning

This guide replaces the prior pg_restore-based workflow. Use this for cloning and environment setup.