Below is a comprehensive README.md or project documentation file. It compiles the architecture, the GitHub Action "Keep-Alive" strategy, and the offline license verification logic we discussed.

You can copy this into a file named **BACKEND_DOCS.md** in your project folder.

Desktop App Backend & Licensing System (PySide6 + Supabase)

This document outlines the architecture for a tiered licensing system using **Supabase** as the backend, **GitHub Actions** for database maintenance, and **Local JWT Validation** for offline access.

---

1. System Architecture

The app uses a "Phoned-Home" model. It stays offline-capable by using signed leases but requires an internet connection once every 30 days to refresh the license and check for updates.

Components:

- **Database (Supabase):** Stores license keys, hardware IDs, and update metadata.
- **Storage (GitHub):** Hosts the actual .exe / .zip update installers (Free & Unlimited).
- **Client (PySide6):** Verifies license keys and performs background update checks.

---

2. Database Schema (Supabase SQL)

Run the following in your Supabase SQL Editor to set up the necessary tables:

SQL

```
-- 1. Licenses Table
CREATE TABLE licenses (
```

```sql
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  license_key TEXT UNIQUE NOT NULL,
  tier TEXT CHECK (tier IN ('free_trial', 'standard', 'ultimate', 'developer')),
  hwid TEXT, -- Bound to the user's hardware
  last_activated TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  is_active BOOLEAN DEFAULT TRUE
);

-- 2. Updates Table
CREATE TABLE app_updates (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  target_tier TEXT, -- Which tier gets this update
  version_number TEXT NOT NULL,
  download_url TEXT NOT NULL,
  is_critical BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

---

3. GitHub Action: "Keep-Alive" (Free Tier Fix)

To prevent Supabase from pausing your database after 7 days of inactivity, create .github/workflows/keep_alive.yml:

YAML

```yaml
name: Supabase Keep-Alive
on:
  schedule:
    - cron: '0 0 * * *' # Every day at midnight
  workflow_dispatch:

jobs:
  ping:
    runs-on: ubuntu-latest
    steps:
      - name: Heartbeat Query
        run: |
    curl -X GET "${{ secrets.SUPABASE_URL }}/rest/v1/licenses?select=count" \
    -H "apikey: ${{ secrets.SUPABASE_KEY }}" \
    -H "Authorization: Bearer ${{ secrets.SUPABASE_KEY }}"
```

Note: Add SUPABASE_URL and SUPABASE_KEY to your GitHub Repo Secrets.

---

4. Offline Logic (JWT Leasing)

To allow the app to work without internet, the app generates a **Local Lease File** upon successful online activation.

State	Action
First Launch	Requires Internet. User enters License Key. App binds Key to Hardware ID.
Subsequent Launch (Online)	App checks Supabase. If valid, updates local license.dat with a new 30-day expiry.
Subsequent Launch (Offline)	App decodes license.dat. If Signature is valid and Expiry > Today, app opens.

---

5. Tiered Update Strategy

Updates are filtered by the tier column in the app_updates table.

1. **Check:** App sends current_version and tier to Supabase.
2. **Filter:** Supabase returns the latest download_url where target_tier == user_tier.
3. **Download:** PySide6 uses requests or urllib to download the installer from GitHub in a background thread (QThread).
4. **Install:** App prompts user: *"A new version for the Ultimate Tier is ready. Restart to install?"*

---

6. Implementation Checklist

- [ ] Create Supabase Project.

- [ ] Apply SQL Schema.
- [ ] Set up GitHub "Keep-Alive" Action.
- [ ] Implement machineid retrieval in Python.
- [ ] Build the PySide6 "Activation" window.
- [ ] Create the background Update Checker thread.

Would you like me to generate the specific Python class that handles the background update checking in PySide6?