Project Proposal: Scalable Data Infrastructure for Pawzz

Animal Care Directory Role: Data Analyst Intern Candidate

Date: February 8, 2026

Candidate: Uday Dokania

1. Executive Summary

This proposal outlines a robust, automated, and auditable data pipeline designed to transform fragmented animal care information into India's first verified pet-care ecosystem. By utilizing a combination of Python-driven automation and SQL-based relational mapping, we can reduce founder dependency while maintaining a "Practo-standard" of data integrity.

2. System Design & Workflow (Q1)

2.1 Proposed Tool Stack

- **Primary Database:** PostgreSQL to handle relational complexity between different service categories (e.g., linking one NGO to multiple shelter locations).
- **Automation & Cleaning:** Python (Pandas, Scrapy, and Regex) for web scraping, standardizing messy inputs, and finding duplicates.
- **Workflow Management:** Airtable or a custom CRM to track the lifecycle of each listing and manage intern assignments.
- **Analytics:** Power BI to monitor team throughput, data quality scores, and city-wise coverage.

2.2 Data Pipeline Stages

To ensure every listing is accurate and auditable, data will move through five distinct stages:

1. **Submitted:** Raw inputs sourced via scrapers, community forms, or intern research.
2. **Needs Fix:** Automated scripts flag entries with missing mandatory fields, invalid phone formats, or inconsistent data.
3. **Ready for Verification:** Complete entries assigned to the verification team for manual/telephonic checks.
4. **Verified:** Proof (geotagged photos, licenses, or call recordings) is attached and approved.
5. **Published:** The listing is live and accessible to the public directory.

2.3 Handling Unstructured Inputs

To manage "messy" data like WhatsApp screenshots or inconsistent addresses, the system will utilize:

- **Regex Standardization:** Python-based Regular Expressions to force all phone numbers and addresses into a universal format.
- **OCR Integration:** Optical Character Recognition (OCR) to extract text from clinical signs, business cards, or handwritten notes provided via screenshots.

3. Verification & Integrity (Q2)

3.1 Verification Checklist: Veterinary Clinics

To be marked as "Verified," a listing must meet the following proof standards:

- **Professional Proof:** A valid Veterinary Council registration number or official license scan.
- **Physical Proof:** A timestamped, geotagged photo of the clinic entrance or storefront.
- **Operational Proof:** A recorded tele-verification call confirming current active hours and available emergency services.

3.2 Red Flag Detection

Unreliable listings will be identified using the following system-level "Red Flags":

- **NAP Inconsistency:** Different names, addresses, or phone numbers listed for the same entity across different platforms (e.g., Google vs. Instagram).
- **Media Duplication:** The use of stock internet imagery instead of original on-site photos.
- **Address Conflicts:** Multiple unrelated businesses registered at the same residential address or a lack of physical habituation.

3.3 Audit Trail & Escalation

To ensure accountability, every listing will include a digital audit trail:

- **Editor Logging:** A permanent record of who modified the data, what was changed, and when.
- **Proof Reference:** A direct link to the specific proof (call recording or photo) used for the verification status.
- **Escalation Rules:** Listings that fail physical verification twice are Rejected. Listings with verified locations but missing licenses are Published with Warning until the document is provided.

Technical Annex: Implementation Logic

To ensure the directory is scalable and reduces founder dependency, the following technical components will be integrated into the workflow.

A. Database Architecture (SQL Schema)

A relational structure in PostgreSQL ensures data integrity and allows for complex auditing. Below is the proposed schema for managing directory listings and their audit history.

SQL

```
-- Main Table for Directory Listings
CREATE TABLE Directory_Listings (
    listing_id SERIAL PRIMARY KEY,
    category VARCHAR(50) NOT NULL, -- Vet, NGO, Shelter, etc.
    business_name VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) UNIQUE, -- Primary key for duplicate prevention
    latitude DECIMAL(9,6),
    longitude DECIMAL(9,6),
    verification_status VARCHAR(20) DEFAULT 'Submitted',
    proof_url TEXT, -- Link to geotagged photo or license
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Audit Trail Table to track changes
CREATE TABLE Audit_Logs (
    log_id SERIAL PRIMARY KEY,
    listing_id INT REFERENCES Directory_Listings(listing_id),
```

```
    editor_id INT, -- Intern ID
    old_value TEXT,
    new_value TEXT,
    change_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    verification_proof_attached BOOLEAN DEFAULT FALSE
);
```

B. Data Cleaning Automation (Python Script)

To handle messy "Unstructured Inputs" (like inconsistent phone formats from WhatsApp or web scraping), the following Python logic will be used for standardization.

Python

```
import re

def standardize_phone(phone_str):
    # Remove all non-numeric characters
    clean_digits = re.sub(r'\D', '', phone_str)

    # Standardize to 10 digits and add prefix
    if len(clean_digits) >= 10:
        return f"+91{clean_digits[-10:]}"
    return None # Flag for manual 'Needs Fix' stage

# Example usage for automated cleaning
raw_inputs = ["098765-43210", "+91 9988776655", "91234 56789"]
cleaned_data = [standardize_phone(num) for num in raw_inputs]
print(cleaned_data)
# Output: ['+919876543210', '+919988776655', '+919123456789']
```

C. Data Pipeline Visualization

The movement of data through the system is governed by strict status transitions to ensure no listing is published without verification.

1. **Submitted:** Initial entry into the Directory_Listings table.
2. **Needs Fix:** Automated SQL triggers or Python scripts flag missing proof_url or invalid phone_number.
3. **Ready:** Mandatory fields are validated; assigned to an intern for a verification call.
4. **Verified:** Intern uploads proof; status updated in Audit_Logs.
5. **Published:** Listing becomes visible in the public directory.