Campaign Buddy AI - Project Status Summary

Project Overview

Goal: Build a web app service that uses LLMs to customize email campaigns for different audience segments, initially working with NationBuilder CRM data.

Business Model: SaaS service for political campaigns and advocacy organizations that use NationBuilder.

Core Functionality

1. **Data Processing**: Load NationBuilder database snapshots, extract email history and engagement data
2. **Segmentation**: Group contacts based on email engagement patterns (high engagement, low engagement, new contacts)
3. **AI Generation**: Use LLMs to customize email content for each segment based on historical patterns
4. **Multi-Provider Support**: Support both paid APIs (OpenAI, Claude) and locally-hosted LLMs

Technical Architecture Decisions

Tech Stack (Finalized)

- **Backend**: Python + FastAPI
- **Database**: PostgreSQL (Cloud SQL for production, Docker for local development)
- **LLM Integration**: Multi-provider abstraction (OpenAI API + local models via Ollama)
- **Infrastructure**: Google Cloud Platform (serverless-first with Cloud Run)
- **CI/CD**: GitHub + GitHub Actions
- **Vectorization**: Chroma (local) with migration path to Pinecone (cloud scale)

Key Architecture Principles

- **Environment Agnostic**: Everything works both locally and on Google Cloud
- **Provider Agnostic**: Can switch between LLM providers via configuration
- **Migration-Friendly**: Serverless → dedicated server path without rebuilding
- **Multi-Tenant Ready**: Designed for multiple clients with data isolation

POC Development Plan (5 Phases)

Phase 1: Database Setup & Data Extraction ⏳ CURRENT PHASE

- Set up local PostgreSQL with Docker ✅
- Set up Google Cloud SQL (deferred)
- Load NationBuilder data and extract contacts, segments, email history
- Build data extraction service

Phase 2: Paid LLM Integration

- OpenAI GPT-4 API integration
- Multi-provider abstraction layer
- Email generation for segments
- Usage tracking

Phase 3: Local LLM Integration

- Ollama setup (Llama 2 or Mistral models)
- Extend abstraction to support local models
- Performance comparison with paid APIs

Phase 4: Vector Database Integration

- Chroma setup for embeddings
- RAG (Retrieval-Augmented Generation) implementation
- Historical email pattern retrieval

Phase 5: Context Management

- Segment-specific background context
- Global prompting context
- Context inheritance system

Current Status & Next Steps

✅ Completed Steps

1. **Project Structure**: Created GitHub repo with proper folder structure

```
campaign_buddy_ai/
├── .venv/
├── data/
├── src/
│   ├── __init__.py
│   ├── main.py
│   └── db.py
├── tests/
├── .env
├── requirements.txt
└── docker-compose.yml
```

2. **Dependencies Installed**: FastAPI, asyncpg, SQLAlchemy[asyncio], uvicorn, python-dotenv
3. **Docker Setup**: PostgreSQL 16 running locally in Docker container
   - Container name: `campaign_buddy_postgres`
   - Database: `campaign_buddy_ai`
   - User: `dev_user` / Password: `dev_password`
   - Port: 5432
4. **Database Loaded**: 1GB NationBuilder snapshot successfully restored
   - File: `backup-for-larouchepac20250728-50530-q0c394_`
   - Schema: `nbuild_larouchepac`
   - Status: Data loaded with some foreign key constraint warnings (normal)

🔄 Current Task

Exploring NationBuilder database structure to identify key tables:

- Need to run: `docker exec -it campaign_buddy_postgres psql -U dev_user -d campaign_buddy_ai -c "\dt nbuild_larouchepac.*"`
- Looking for: contacts/people, email campaigns, email interactions, segments/lists

📋 Immediate Next Steps (Phase 1 completion)

1. **Database Exploration**:
   - Identify key tables (people, email_recipients, blasts, lists)
   - Understand schema structure and relationships
   - Document table purposes and key fields
2. **Database Connection Module** (`src/db.py`):

```python
# Async SQLAlchemy connection with environment switching
DATABASE_URL=postgresql+asyncpg://dev_user:dev_password@localhost:5432/campaign_buddy_ai
```

3. **Data Extraction Service** (`src/services/data_extractor.py`):
   - Extract contacts by segments (based on existing NB segments)
   - Extract email history and engagement metrics
   - Prepare data for LLM consumption
4. **Exploration Scripts**:
   - Create data exploration and validation scripts
   - Test extraction functionality

Beta Client Details

- **Rich NationBuilder database**: 1GB with extensive contact and email history
- **Pre-defined segments**: Segments already exist in NB, no need to create them
- **Real data**: Allows for meaningful testing and validation

Environment Configuration

Current .env Setup

```
DATABASE_URL=postgresql+asyncpg://dev_user:dev_password@localhost:5432/campaign_buddy_ai
ENVIRONMENT=development
# OPENAI_API_KEY=your_key_here (for Phase 2)
```

Docker Commands Reference

```bash
```

```
# Start database
docker-compose up -d

# Check status
docker-compose ps

# Access database
docker exec -it campaign_buddy_postgres psql -U dev_user -d campaign_buddy_ai

# Stop database
docker-compose down
```

Key Technical Insights from Setup

1. **Database Format**: NationBuilder exports as PostgreSQL custom dump files (PGDM format)
2. **Schema Structure**: Uses `nbuild_larouchepac` schema prefix
3. **Size Considerations**: 1GB database took ~5-10 minutes to restore
4. **Foreign Key Issues**: Normal to see constraint errors during restore - data still loads correctly

Risk Mitigation Strategies

- **Multi-provider LLM**: Avoid vendor lock-in with abstraction layer
- **Local + Cloud**: Identical environments prevent deployment issues
- **Serverless Start**: Minimize initial costs during validation
- **Migration Path**: Clear path from serverless to dedicated infrastructure

Success Criteria

- **Phase 1**: Successfully extract and segment NationBuilder contacts
- **POC Overall**: Generate meaningfully different emails per segment using both paid and local LLMs
- **Business**: Validate that AI-generated emails are higher quality than generic campaigns

Development Environment

- **OS**: Windows with PowerShell in VSCode
- **Python**: Virtual environment with required packages
- **Database**: PostgreSQL 16 in Docker
- **Version Control**: GitHub with planned CI/CD integration