Slackbot Content Pipeline — AI Engineer Assignment

Author: Ahtesham Shaikh

1. Project Overview

This project implements a Python-based Slackbot Content Pipeline that accepts keyword input, processes it through a content pipeline, and produces an automated PDF report. The solution is a local, modular demonstration of the core pipeline: input → preprocessing → grouping → outline & idea generation → report generation. The implementation focuses on clarity, modularity, and testability so it can be extended to production-grade features.

2. Objectives

• Accept keyword input (CSV or pasted list) and produce cleaned keywords.
• Group related keywords into clusters for targeted content generation.
• Generate an outline and a post idea for each cluster.
• Produce a downloadable PDF report summarizing raw/cleaned keywords, clusters, outlines, and post ideas.
• Provide a Slack-friendly interface (slash command / endpoint) for real-time operation.

3. High-level Architecture

The system is organized into modular components:
• Flask API: exposes the /slack/events endpoint and receives simulated Slack events.
• Slack Bot Layer: Slack Bolt patterns are used; for local testing the app runs in mock mode.
• Content Generator Module: generates outlines and PDF reports using ReportLab.
• Test Client: test_request.py simulates Slack POSTs for validation.
• Output Storage: generated PDFs are saved to the outputs/ folder with timestamps.

4. Core Modules & Files

main.py — launches the Flask app.

slack_bot.py — endpoint handling and Slack Bolt initialization (mock mode for local testing).

content_generator.py — creates the structured PDF report using ReportLab. Produces title, summary, bullets and timestamp.

pdf_generator.py — helper utilities for PDF composition (if present).

test_request.py — simulates Slack events for testing the pipeline.

README.md — setup and run instructions.

5. Implementation Notes

• Keyword Handling: The demo currently processes direct text input and produces a programmatic summary. A CSV upload and explicit cleaning (lowercasing, trimming, deduplication) can be added quickly via a small extension.

• Grouping: Semantic grouping is currently simulated. For production, I recommend SentenceTransformers embeddings + KMeans or pgvector on Supabase for vector search.

• Outline Generation: The current implementation uses a scripted approach (programmatic outlines). This can be upgraded to an LLM call (OpenAI/Hugging Face) or search-driven extraction (SerpAPI) where free tiers are available.

• Slack Integration: The code follows Slack Bolt patterns; to enable real Slack operations, set SLACK_BOT_TOKEN and SLACK_SIGNING_SECRET, update the app's request URL, and install to a workspace.

• Report Delivery: The PDF is saved locally in outputs/. To auto-deliver, implement Slack files.upload using the bot token or integrate SendGrid (free tier) for email delivery.

6. Security & Operational Considerations

• Do not commit secrets to the repo; use environment variables for tokens and API keys.
• Validate CSV uploads and limit file sizes to avoid resource exhaustion.
• When using external APIs, consider rate limits and caching via Upstash Redis.

7. Future Work & Extensions

• Add CSV upload endpoint and UI for keyword entry.
• Implement semantic clustering with SentenceTransformers + pgvector or Pinecone.
• Integrate SERP extraction for outline creation or use a free search API.
• Add Slack file upload and SendGrid email delivery for automated report distribution.
• Implement persistent storage (Supabase) and /history command for past runs.

Generated by: Ahtesham Shaikh — 2025-10-13