1. Introduction

1.1 Project Title: LinkedIn GenAI Post Generator

1.2 Overview:

This project is designed to assist LinkedIn influencers in generating new posts that match their previous writing style using Generative AI. By feeding old posts into the system, it analyzes tone, topic, length, and language, then allows the user to generate future posts that align with their unique voice.

1.3 Scope:

Included:

- Analyzing previous LinkedIn posts for metadata (topic, language, length)

- Few-shot prompting to generate new posts with Groq's LLM

- Streamlit-based user interface

Not included:

- Integration with LinkedIn API for direct posting

- Advanced scheduling features or analytics

1.4 Target Audience:

- Developers interested in GenAI applications

- LinkedIn content creators/influencers

- Contributors looking to improve or extend functionality

2. Getting Started

2.1 Prerequisites:

- Python 3.10+

- Git

- Groq API key (free tier available at https://console.groq.com)

- Streamlit

2.2 Installation Guide:

2.2.1 Cloning the Repository:

git clone https://github.com/TanishqThuse/linkedin-style-genai-post-generator.git

cd linkedin-style-genai-post-generator

2.2.2 Environment Setup:

python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

pip install -r requirements.txt

2.2.3 Configuration:

Create a .env file in the root directory:

GROQ\_API\_KEY=your\_groq\_api\_key\_here

2.3 Running the Project:

streamlit run main.py

3. Project Architecture & Tech Stack

3.1 High-Level Architecture:

- Streamlit app UI

- LLM Helper using LangChain and Groq

- Environment and config handled via dotenv

- Few-shot learning enabled by curated past post samples

3.2 Tech Stack:

Frontend: Streamlit

Backend/LLM Layer: Python, LangChain, Groq API

Environment Management: Python venv, dotenv

Other Tools: Git, VS Code, GitHub

4. Core Features and Functionality

4.1 Post Analysis:

Reads influencer's past posts, extracts metadata such as topic, length, and language using simple keyword matching and tagging strategies.

4.2 Post Generation:

Uses few-shot prompting by passing selected past posts and current user inputs to Groq's LLM (e.g., LLaMA 3) for style-matching generation.

4.3 User Control:

Lets the user select topic, language, and length before triggering post generation.

5. Challenges and Solutions

5.1 Challenge: Model Deprecation by Groq

Problem: The model llama-3.2-90b-text-preview was deprecated.

Solution: Replaced with llama3-70b-8192 after reading Groq’s deprecation docs.

Lesson: Always check for model lifecycle updates when using 3rd party AI services.

5.2 Challenge: GitHub Branch Conflicts

Problem: Pushing to main caused non-fast-forward errors due to preexisting remote README.

Solution: Used conflict resolution (--ours strategy) to preserve local README and commit cleanly.

Lesson: Always check remote state before first push or use --force-with-lease cautiously.

6. Testing

6.1 Test Strategy:

Manual testing using sample past posts and UI controls.

6.2 Running Tests:

Currently no automated tests, but can be added via pytest for LLM interface and utils.

6.3 Test Coverage:

N/A

7. Future Enhancements & Roadmap

7.1 Planned Features:

- Integration with LinkedIn API for auto-posting

- Multi-language support

- User login and post history saving

7.2 Areas for Improvement:

- Add automated testing

- Use database for post storage

- Improve few-shot matching algorithm with vector similarity

8. Contribution Guidelines

8.1 How to Contribute:

- Fork the repo

- Create a new branch for your feature or fix

- Submit a pull request with proper description

8.2 Code Style/Conventions:

- Follow PEP8 for Python

- Comment major logic blocks and use descriptive variable names

9. License & Acknowledgements

9.1 License:

MIT License — Commercial use prohibited without permission.

9.2 Acknowledgements:

- Codebasics

- Groq LLMs

- LangChain

- Streamlit

- ChatGPT for collaborative brainstorming and debugging

Prepared by Tanishq Thuse