Technical Report

1. Project Overview

This project, developed for the Influence OS AI Intern Assessment, is a full-stack web application named InfluenceAI. Its primary objective is to function as an autonomous AI agent for LinkedIn personal branding. The application is designed to understand a user's professional context, generate engaging and relevant content, and automate the process of sharing that content on LinkedIn, thereby enhancing a user's professional presence and brand.

Core Functionalities Implemented:

- **Secure User Authentication:** Full integration with the LinkedIn API using the OAuth 2.0 protocol for secure, passwordless user login.
- **User Profile Analysis:** Ability to fetch and display a logged-in user's basic profile information (name, profile picture) from the LinkedIn API.
- **AI-Powered Content Generation:** Utilizes Google's Gemini Large Language Model (LLM) to create high-quality, professional LinkedIn posts based on user-defined roles and topics.
- **Content Management & History:** All generated posts are saved to and retrieved from a persistent PostgreSQL database, providing users with a history of their content.
- **Automated Posting:** Allows users to share their generated content directly to their LinkedIn feed with a single click.
- **Live Deployment:** The entire application is deployed on the cloud, with a live, publicly accessible URL for the frontend.

2. Architecture Overview

The application is built on a modern, decoupled full-stack architecture, ensuring scalability and maintainability.

- **Frontend:** A dynamic single-page application built with **Next.js** and **React**. It is responsible for the user interface, state management, and all client-side interactions. The frontend is deployed globally on **Vercel**.
- **Backend:** A robust REST API built with **Python** and the **FastAPI** framework. It handles business logic, AI integration, database operations, and secure communication with the LinkedIn API. The backend is deployed on **Render** as a web service.
- **Database:** A **PostgreSQL** database, fully managed and hosted by Render. It is connected to the backend via Render's private network, ensuring secure and low-latency communication.
- **External Services:**
    - **Google AI Platform:** Provides the Gemini LLM for content generation.

    - **LinkedIn API:** Used for user authentication and content sharing.

Data Flow Diagram: `User -> Vercel Frontend -> Render Backend (FastAPI) -> (PostgreSQL DB | Google Gemini API | LinkedIn API)`

3. AI Model Choices & Implementation

- **Model:** The application uses the **`gemini-1.5-flash-latest`** model from the Google Gemini family. This model was chosen for its excellent balance of performance, quality, and its availability within a generous free tier, making it ideal for a development project.
- **Orchestration:** As required by the project brief, **LangChain** was used for AI orchestration. Specifically, the `langchain-google-genai` library was used to interact with the model. The LangChain Expression Language (LCEL) was implemented (`prompt | llm | parser`) to create a clean, efficient, and modern processing chain for generating content.

4. Key Implementation Decisions

- **Backend Framework: FastAPI** was selected for its high performance, asynchronous capabilities, and automatic generation of interactive API documentation (`/docs`), which was invaluable for testing and development.
- **Frontend Framework: Next.js** was chosen as it is the industry standard for production-grade React applications, offering features like server-side rendering and a seamless deployment experience on Vercel.
- **Authentication:** The full **OAuth 2.0 Authorization Code Flow** was implemented. This is the most secure method for third-party authentication as it ensures the application never handles or stores user passwords. Access tokens are managed on the client side to make authenticated API calls.
- **Deployment:** A dual-platform cloud deployment was chosen for optimal performance and ease of use.
    - **Render** was selected for the backend and database due to its seamless GitHub integration, automatic deployments, and managed PostgreSQL service with a generous free tier.
    - **Vercel** was chosen for the frontend for its best-in-class support for Next.js, global CDN, and automatic deployments on every `git push`.
- **CORS & Environment Management:** A key challenge was managing communication between the frontend, backend, and external APIs across different environments (local vs. production). This was solved by implementing a robust CORS policy in FastAPI and managing all secret keys and URLs through environment variables, which were configured both locally and in the respective deployment services (Render and Vercel).