**Solution Architecture – SmartSDLC Project**

Date: 27 June 2025

Team ID: LTVIP2025TMID22707

Project Name: SmartSDLC: AI-Enhanced Software Development Lifecycle

**SmartSDLC Architecture Overview**

* **Frontend & Hosting**
  + Uses **Gradio** to build the interactive UI (text input, exercise selectors, charts), running within Google Colab for rapid prototyping and GPU access .
  + Public access enabled via **ngrok** (e.g., using pyngrok in Colab notebooks)
* **Language Detection**
  + Utilizes **langid** to detect the input language and output a confidence score, ensuring feedback aligns with the learner’s chosen language.
* **Model Inference Layer**
  + Leverages **IBM Granite 3.3‑2B‑Instruct** through Hugging Face Transformers, PyTorch, and Accelerate.
  + The 2 B model supports instruction-following, multilingual reasoning, and long-context capabilities (128K tokens)
  + Optional **quantized GGUF variants** for reduced memory usage and faster inference.
* **Caching**
  + Implements lightweight **in-memory caching** (e.g., lru\_cache) to optimize performance and help achieve sub-2-second response times.
* **Analysis & Visualization**
  + Uses **Matplotlib** or **Plotly** to compute and render metrics (word count, sentence complexity) and generate bar and radar charts for learner feedback.
* **API & Tunneling**
  + **ngrok/pyngrok** tunnels the Gradio interface.
  + Optionally wraps the app with **FastAPI** to provide REST endpoints alongside the Gradio UI for further integration.
* **Production-Ready Layer (Optional)**
  + Employs **Docker + FastAPI + Redis** to containerize the app, support scalable caching, and handle concurrent accesses beyond Colab environments.
* **Logging & Monitoring**
  + Built-in **Python logging** tracks inference latency and exceptions.
  + Optionally integrates **Prometheus/Grafana** for detailed performance monitoring in production setups.

**Summary**

This architecture enables a rapid MVP via Colab + Gradio + ngrok, ensures efficient real-time corrections with caching and optional quantization, offers rich visual feedback, and supports a smooth transition to production with containerization, scalable APIs, and monitoring tools. Let me know if you'd like a PlantUML diagram or code scaffold to build this out!