# **AI-enabled Text Summarizer App**

**Production Deployment Strategy**

For the current demo, due to resource constraints, I used the LLaMA 3.2 1B model. For production-level deployment, we can select more powerful models based on benchmarks that evaluate performance on text summarization tasks.

In a production environment, model selection should be guided by the specific task and requirements. We can define the necessary benchmarks, evaluate and rank candidate models accordingly, and then proceed with deployment.

For example the benchmarks that can be considered for text summarization are :

* CNN/DailyMail  
  This benchmark is widely used for testing abstractive summarization. It pairs news articles with multi-sentence summaries, challenging models to extract key information from long-form text. The dataset is particularly suited for evaluating LLMs on tasks where comprehensive summaries are required.
* Gigaword  
  Gigaword focuses on extreme summarization, where the goal is to generate concise, one-line summaries or headlines from input texts. It is an excellent benchmark for testing brevity and informativeness in summarization models.
* XSum (Extreme Summarization)  
  XSum is designed for single-document summarization and emphasizes creating highly concise summaries. It includes BBC news articles and professionally written single-sentence summaries, making it a tough benchmark for abstractive summarization.
* PubMed  
  This dataset targets summarization in the biomedical domain. It features over 200,000 scientific papers with abstracts, testing LLMs' ability to handle domain-specific terminology and dense content. It is particularly valuable for assessing models in research and healthcare applications.
* SAMSum  
  SAMSum is a unique benchmark for summarizing conversational data. It contains over 16,000 human-written chat summaries, evaluating a model's ability to process informal dialogues and fragmented language into coherent summaries.
* GovReport  
  GovReport focuses on summarizing government reports, providing a dataset of highly formal and structured texts. It tests models on their ability to summarize dense, policy-oriented documents with accuracy and coherence.

**Journey after model selection :**

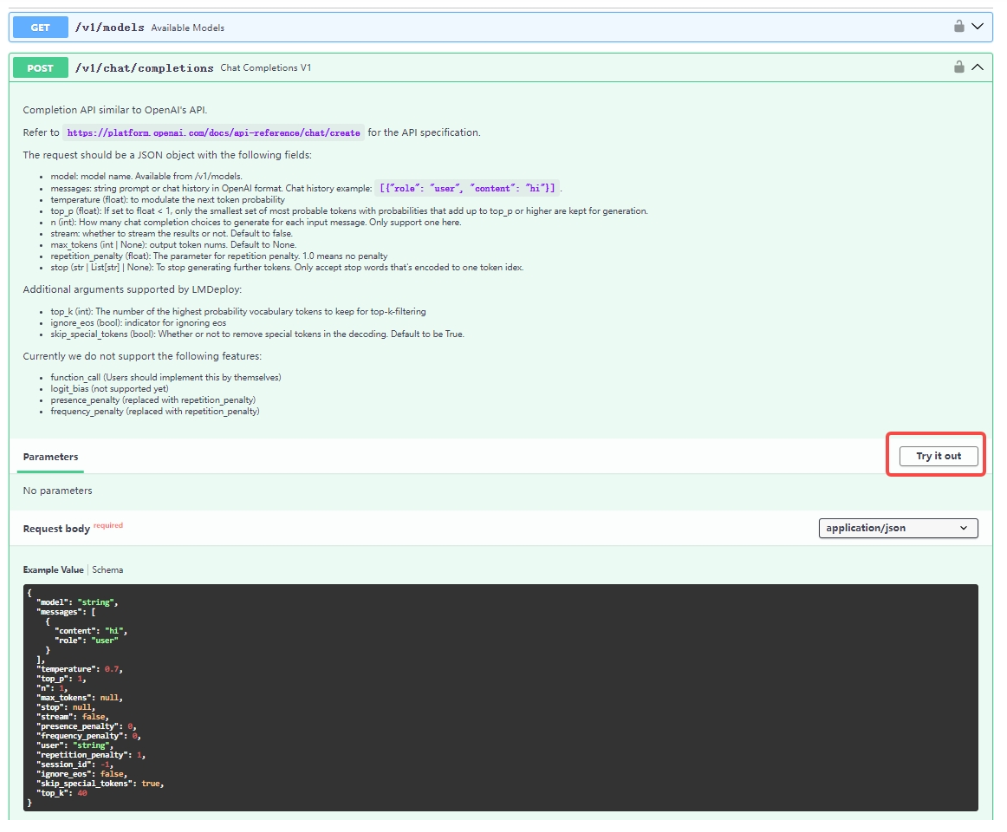
**(High level view of production deployment process)**

**Open source model Journey :**

If the model is an open source model, We must check the feasibility and availability of the model in frameworks like A high-throughput and memory-efficient inference and serving engine for LLMs, Some of them are vLLM, LMDeploy, SGLang, TensorRT, LitServe.

1. Select the model based on the use case and benchmarks and the type of the model (Open source or proprietary).
2. We need to derive the Server computation based on the model, based on the inference hits (Live Traffic hits) and launch an ec2 instance.
3. Set up required [NVIDIA and CUDA](https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=20.04&target_type=deb_local) toolkit for GPU Access and usage
4. Experiment on different serving engines for the selected model and finalize the serving engine.
5. Deploy the model and create a FAST API using a serving engine like vLLM, LMDeploy.
6. Use the created FAST API Endpoint for the integration of the model as a micro service. I recommend using [LMDeploy](https://lmdeploy.readthedocs.io/en/latest/llm/api_server.html) as its TurboMind is an inference engine that supports high throughput inference for conversational LLMs. It’s based on NVIDIA’s [FasterTransformer](https://github.com/NVIDIA/FasterTransformer).

The sample [endpoint](https://lmdeploy.readthedocs.io/en/latest/llm/api_server.html) of the created fast api via LMDeploy :



**Proprietary model Journey :**

If the model is a proprietary, we must access the model endpoint from the respective org/ model provider console page or cloud providers like AWS Bedrock, GCP Vertex AI, Azure AI Studio.

We can then use the endpoint and infer the model and integrate model into any service and prompt it.