Product 1: ChatGPT

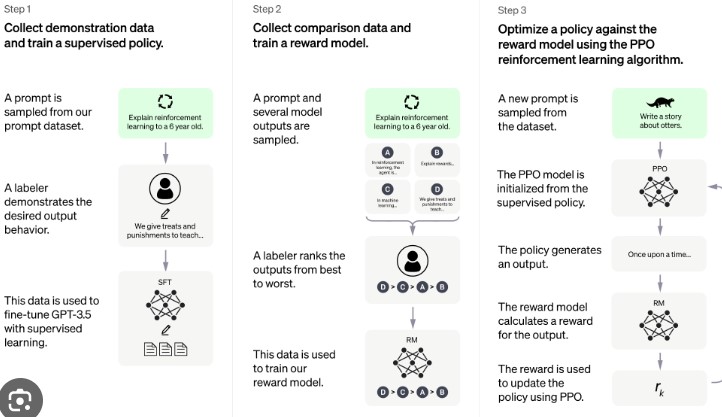
ChatGPT is an existing product developed by OpenAI. It is powered by **large language models** and designed to provide conversational capabilities. The main use case of ChatGPT is to enable users to have interactive and dynamic conversations with an AI-powered chatbot.

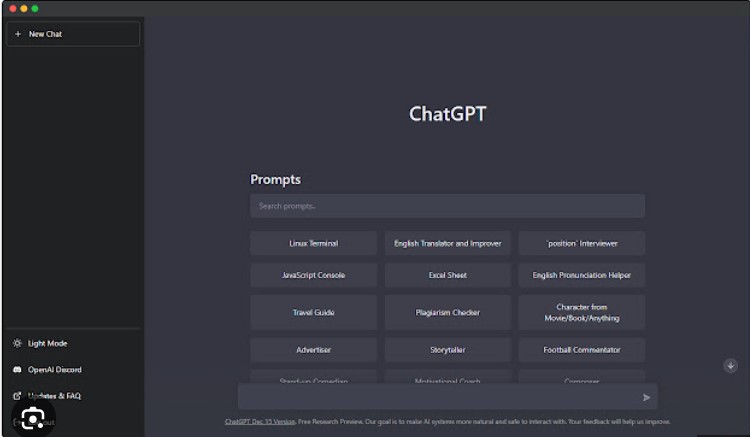
**Users of ChatGPT** can be individuals, businesses, or organizations that want to integrate a chatbot into their platforms or applications. **The task** that users want to perform with ChatGPT is to have natural language conversations with the chatbot and receive relevant and coherent responses.

**The input to ChatGPT** is text-based messages or prompts from the user. These messages can be in the form of questions, statements, or requests. The chatbot processes the input message using its underlying language model and generates an appropriate response based on the context and content of the message.

**The output of ChatGPT** is a text-based response generated by the chatbot. The response aims to provide relevant information or address the user's query or request. The chatbot leverages its language model to generate coherent and context-aware responses that simulate human-like conversation.

ChatGPT, despite being a good product overall, is trained using data readily available in public domain, which runs into the **“garbage in, garbage out”** problem that our Kyndryl project aims to resolve





Product 2: Chatbase

The first existing product that has similar functionality to our product would be Chatbase. Chatbase is a product that provides a chatbot solution using Language Model (LLM) technology. It allows companies to create a chatbot that can answer user queries based on private data, such as PDFs and URLs. Chatbase offers a similar user interface and workflow to our product, making it a relevant comparison.

**The common workflow** for building a chatbot trained with private data through Chatbase is as follows. The user uses the Chatbase UI to upload their private data in the form of **files, text, URLs, etc**. The file will then be ingested, extracted, and processed into trainable data for the chatbot (LLM). Chatbase will then provide the user with a chatbot that has been trained on this data and can answer user queries based on the uploaded information. This is similar to the intended workflow for our product.

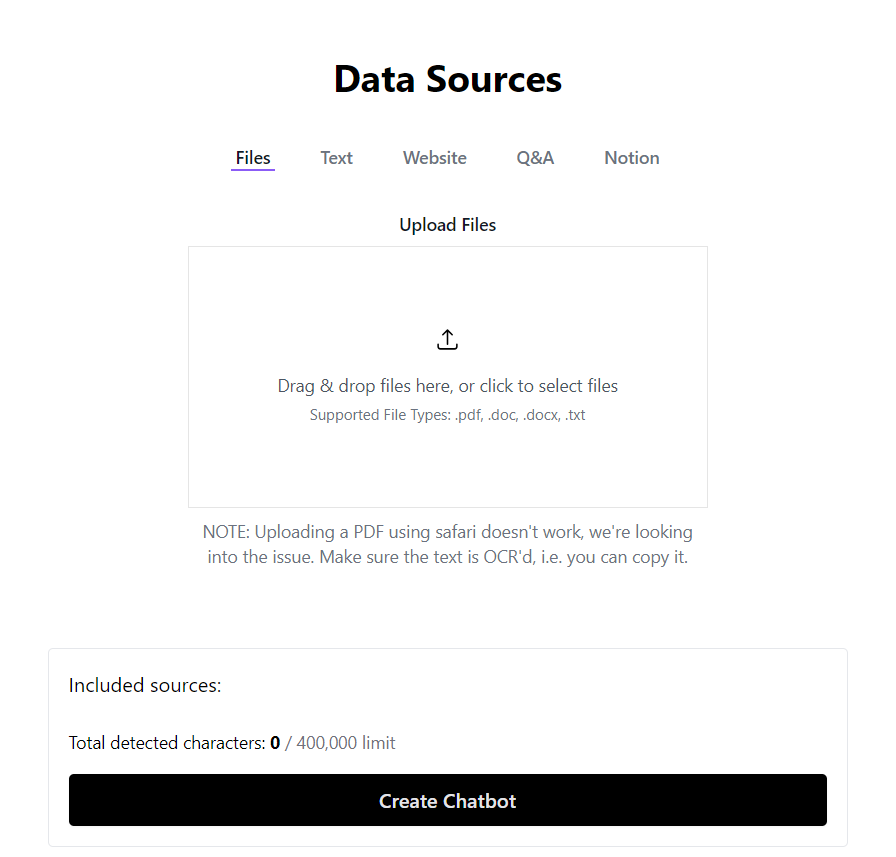


Figure 1: Chatbase Document Upload UI

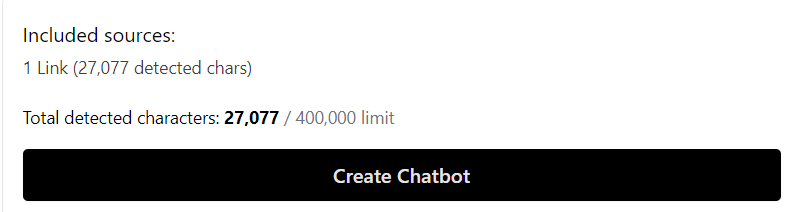


Figure 2: Document Ingested

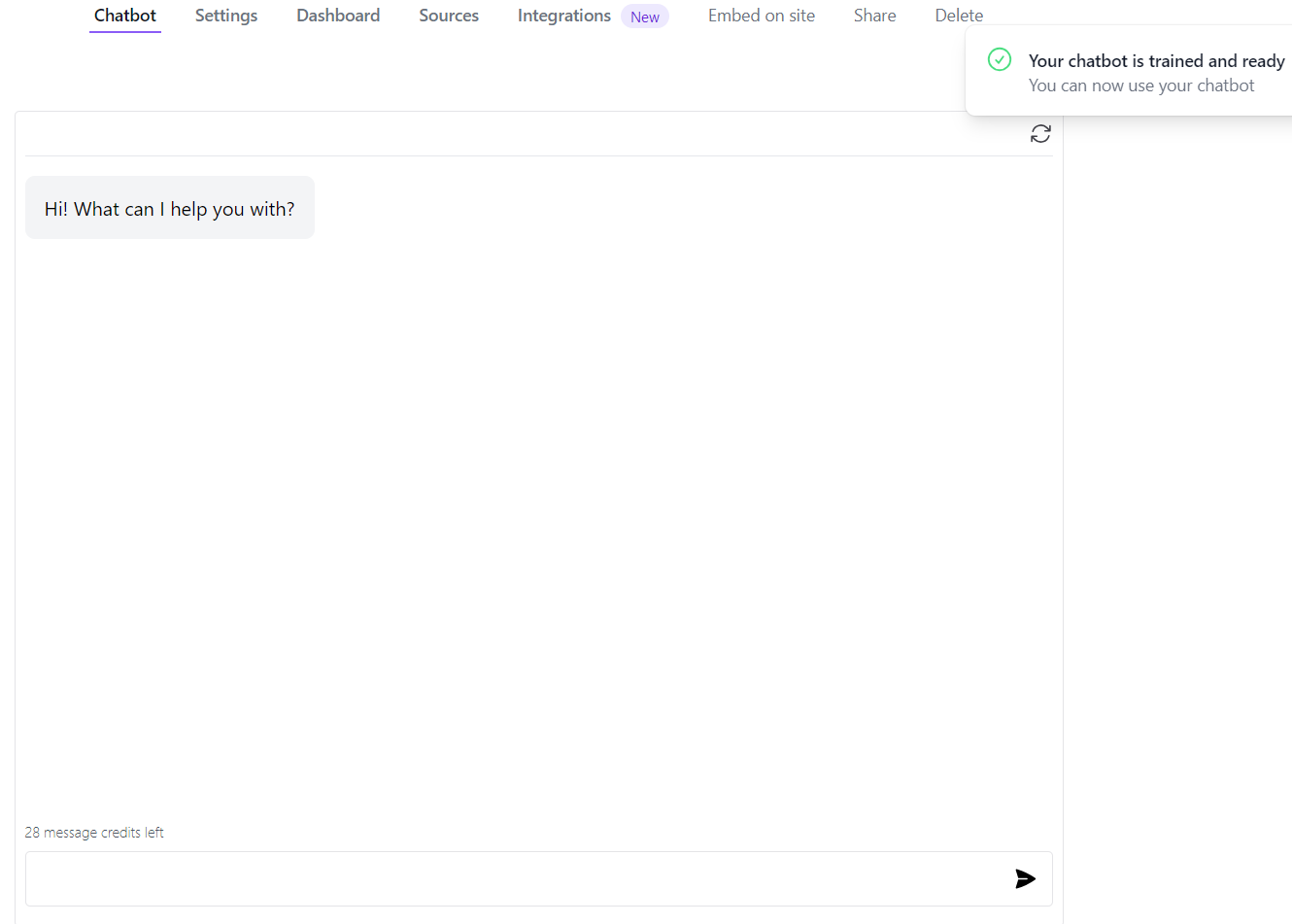


Figure 3: Chatbot ready

The following is an example **use case** of Chatbase. **A new startup** just released its product to the public, and despite providing documentation, they are **overwhelmed with emails and support tickets** from users and developers who **do not read the full documentation**. The company wants to alleviate this burden by **creating a chatbot (LLM) that users can interact with** to find solutions to their problems without overwhelming employees. The user's goal is to reduce the workload of their support team and provide quick and accurate responses to user inquiries.

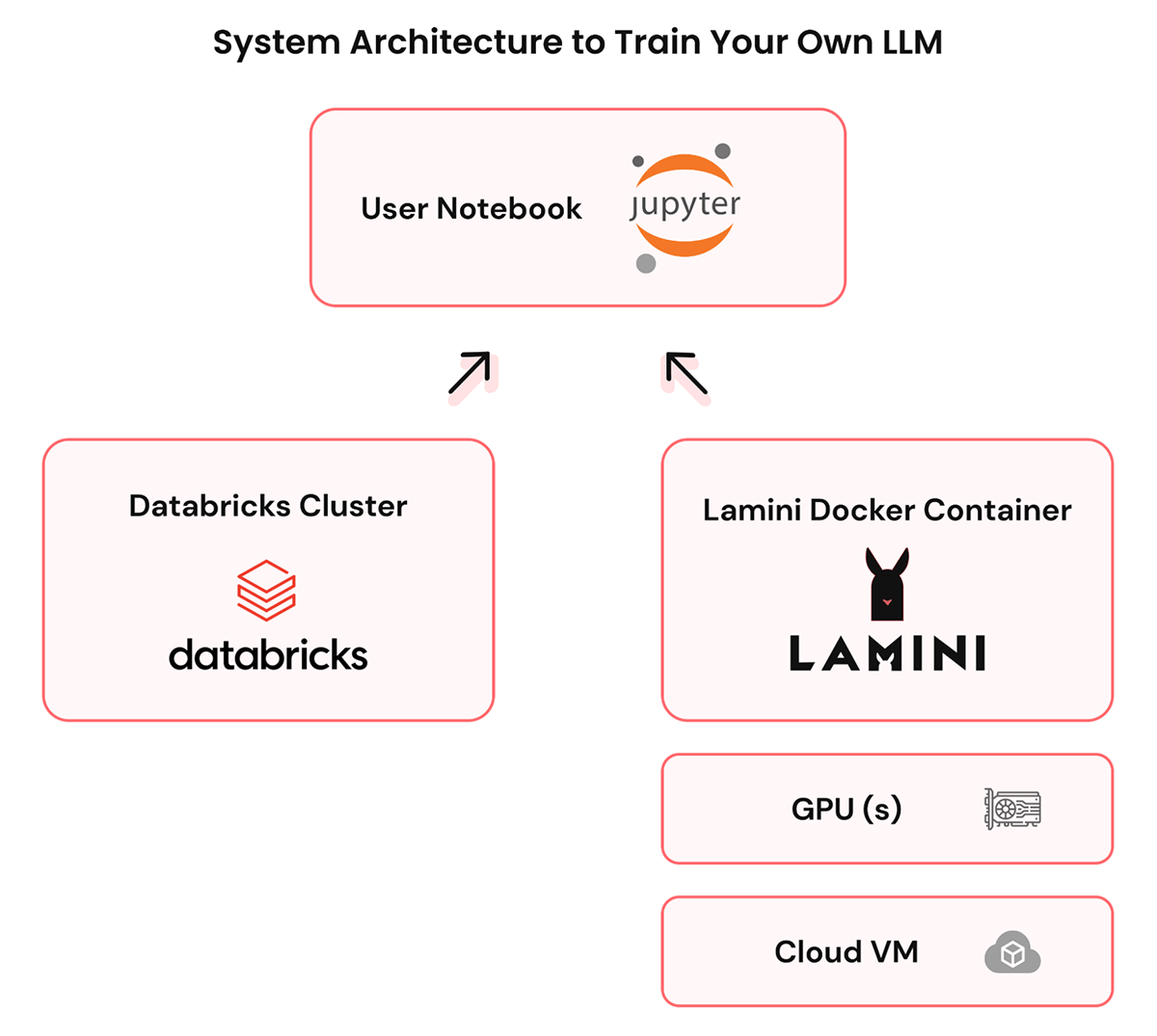
**The tasks** that the user must accomplish are to **gather and upload** their desired private data. The user then simply clicks on **“create a chatbot”** and the Chatbase product will provide them with a chatbot ready to be used. In terms of inputs and outputs, **the input** to Chatbase consists of the user's private data, which can be in the form of **PDFs, URLs, or other relevant formats.** **The output** provided by Chatbase is **a chatbot** that can understand and respond to user queries based on the uploaded data. Users can interact with the chatbot by asking questions or providing prompts, and the chatbot generates appropriate responses based on its training.

It is quite evident that the Chatbase product aligns closely with our product vision, but with a stronger emphasis for **everyday individual users** rather than catered towards enterprise-grade organization. We, however, are really impressed by the **intuitive user interface and ease of use** of Chatbase, and aims to follow those guidelines and perhaps even improve our interface to be even more user-friendly.

Product 3: Databrick’s Lamini

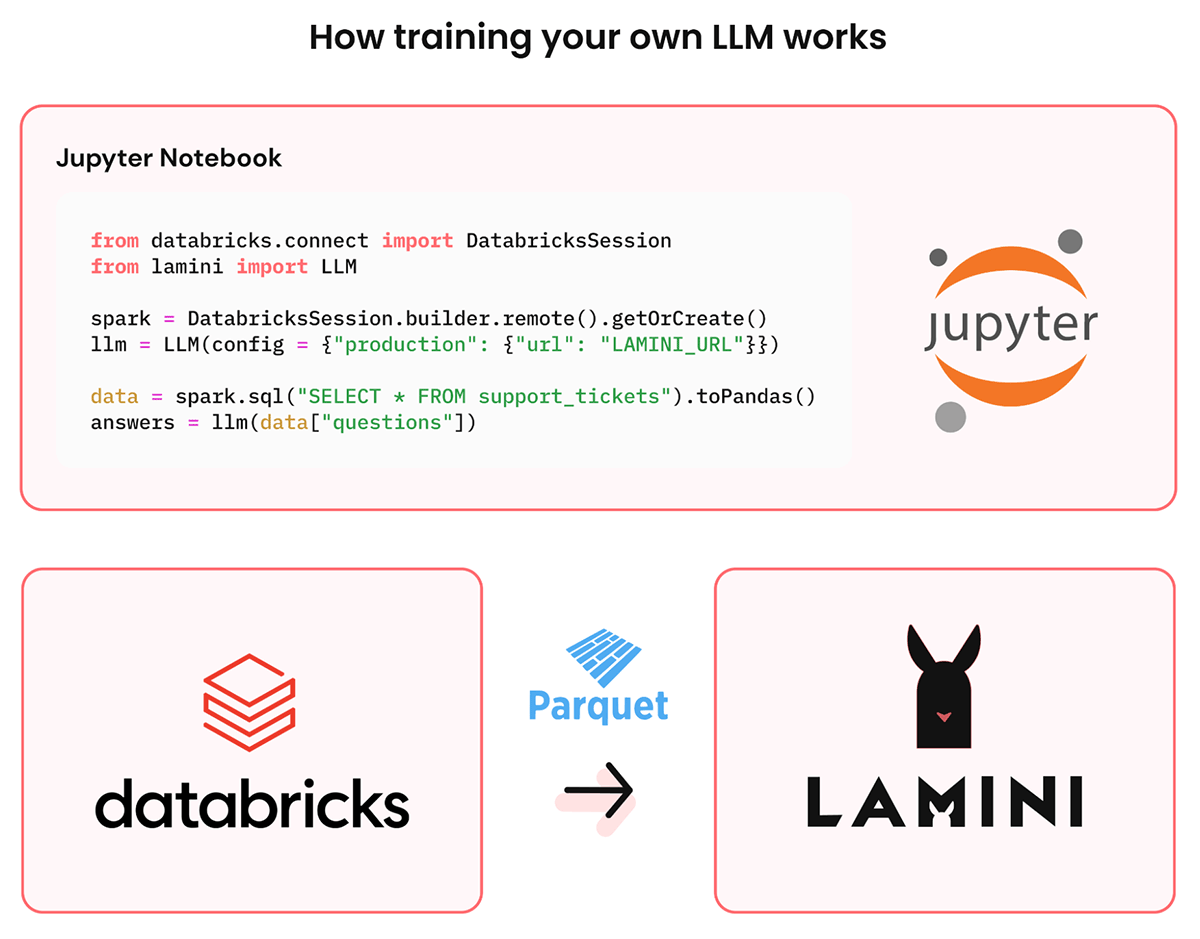
(source: https://www.databricks.com/blog/guest-post-using-lamini-train-your-own-llm-your-databricks-data)

The existing product that shares similarities with our product is Lamini, a library developed by Databricks. Lamini is specifically designed as an **high-code, enterprise-grade solution**, enabling users to create their own Language Model (LLM) trained on private data host. It distinguishes itself by focusing on data security and preventing data leaks through training LLMs exclusively on data stored and maintained on the Databricks data platform. While Lamini offers LLM functionality like our product, it stands out with unique features that cater to the needs of large enterprises and organizations, while losing out on ease of implementation.



Private LLM System Architecture

**A sample workflow** of creating a custom LLM with private data using Lamini is as follows. An enterprise developer or development team will install the library on and connect it to the company’s Databricks instance, more specifically a cluster or a virtual private cloud (VPC). The user would then configure their code to retrieve data from their instance to be used as training data for their LLM. If the data is not already available in the Databricks instance, the user must upload their private data there. Once that data is available, the user can then use it to train a custom LLM for their appropriate needs. All of these steps are done **within the Databricks platform or ecosystems**, eliminating the risk of sensitive data being leaked to the public.



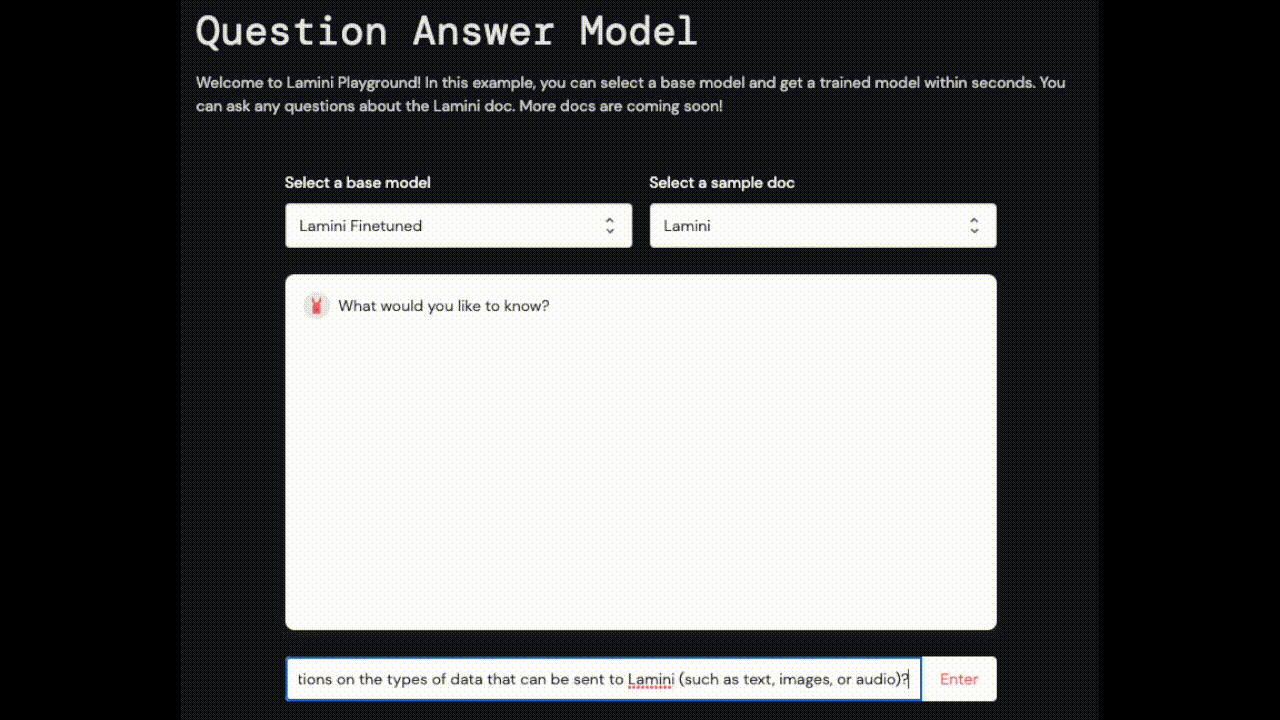
Training a LLM with Lamini

A good sample **use case** for Lamini is when a top tech company or a trading firm desires to build a customized LLM trained on their internal engineering documents. The objective is to enhance the development speed and efficiency of their developers. By maintaining full control over their LLM and avoiding reliance on costly and inconsistent external solutions, they can address their specific requirements. Hiring additional machine learning engineers for building custom LLM models is not only expensive but also unnecessary. Using Lamini would then be the most cost effective solution for the organization.

**The user base** of Databricks's Lamini primarily consists of tech or financial companies. These users possess a technical background and seek to improve their development efficiency through custom LLM models trained on their own private data. Privacy and control over their data are crucial considerations for these users.

**The main task** for the user is to set up a custom LLM using Databricks's Lamini. If their data is not already available on the Databricks platform, they must first securely upload it. Once the data is accessible, they install and connect Lamini, which handles all the necessary setup and infrastructure for the LLM. The engineers can then train the custom LLM with their data, tailoring it to their specific use case.

Regarding **inputs** and outputs, Lamini requires the user's private data on the Databricks platform, which may include internal engineering documents, code repositories, or other relevant information. **The output** of Lamini is a customized LLM capable of understanding and generating human-like text based on the user's specific data. Users can interact with the LLM by providing questions or prompts, and the LLM generates appropriate responses based on its training. The output can be in the form of text displayed to users in an application, API responses, or any other format suitable for the user's specific use case.



Lamini Output UI