Pyro: Your Python Assistant

Welcome to the Pyro project presentation. Pyro is an advanced chatbot designed to assist developers with Python 3.12 documentation queries. Powered by GPT-4 and a Weaviate vector database, Pyro aims to revolutionize how programmers access and utilize Python documentation.

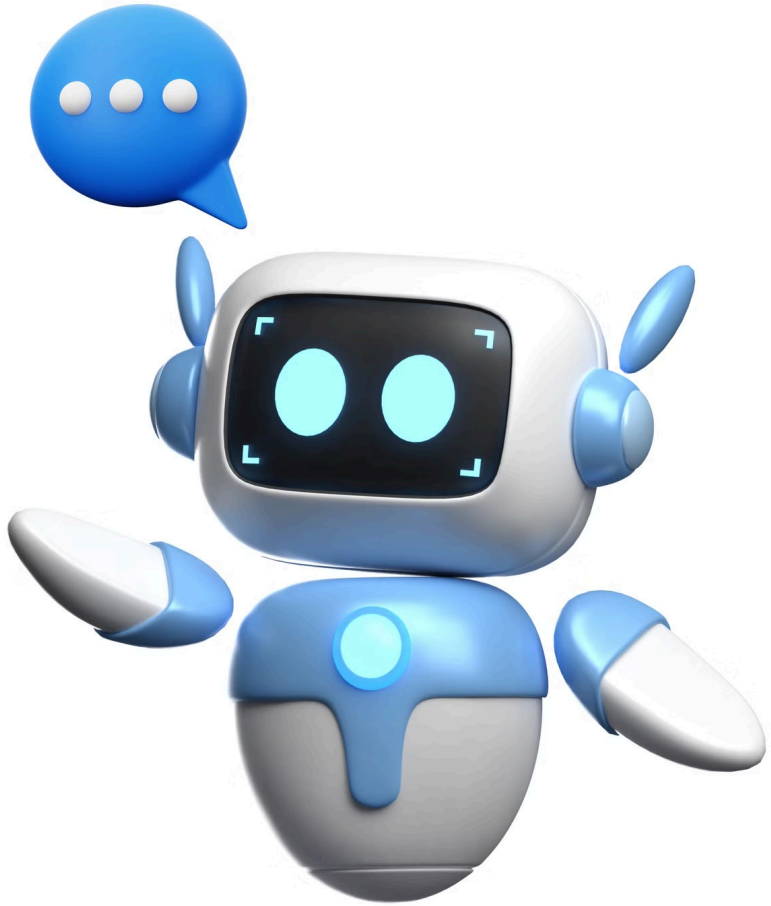
Course: INFO 7375 Prompt Engineering & A.I

Date: July 16, 2024

by JAINAL GOSALIYA

Introduction to Pyro

Pyro is a cutting-edge chatbot that leverages the power of artificial intelligence to provide instant, accurate answers to Python-related questions. It combines the comprehensive knowledge of Python 3.12 documentation with the natural language processing capabilities of GPT-4.



1 AI-Powered

Utilizes GPT-4 for natural language understanding and generation.

2 Comprehensive

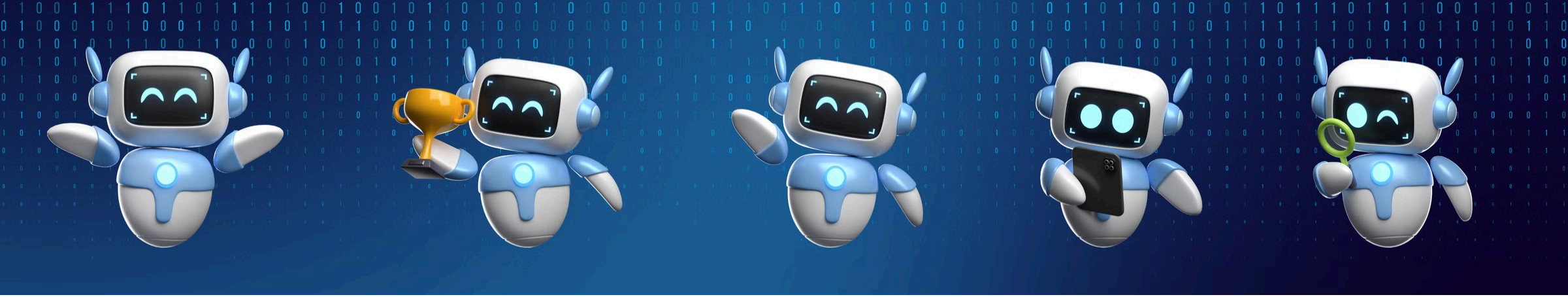
Contains vectorized information from the entire Python 3.12 documentation.

3 Efficient

Provides quick and accurate responses to Python-related queries.

4 User-Friendly

Offers an intuitive interface for seamless interaction.



Chatbot Capabilities

Pyro is designed to assist developers with a wide range of Python-related queries. It can provide explanations, code examples, and best practices for Python programming.

Syntax Assistance

Explains Python syntax and provides usage examples.

Debugging Help

Offers suggestions for common programming errors and bugs.

Library Information

Provides details on Python's standard library and popular third-party packages.

Best Practices

Shares coding best practices and design patterns.

Architecture and Components

Pyro's architecture is built on a robust foundation of cutting-edge technologies. It integrates GPT-4, Weaviate vector database, and a custom RAG pipeline for optimal performance.

1

User Interface

The Streamlit front-end where users input their Python-related queries.

2

GPT-4 Engine

Processes natural language and generates human-like responses.

3

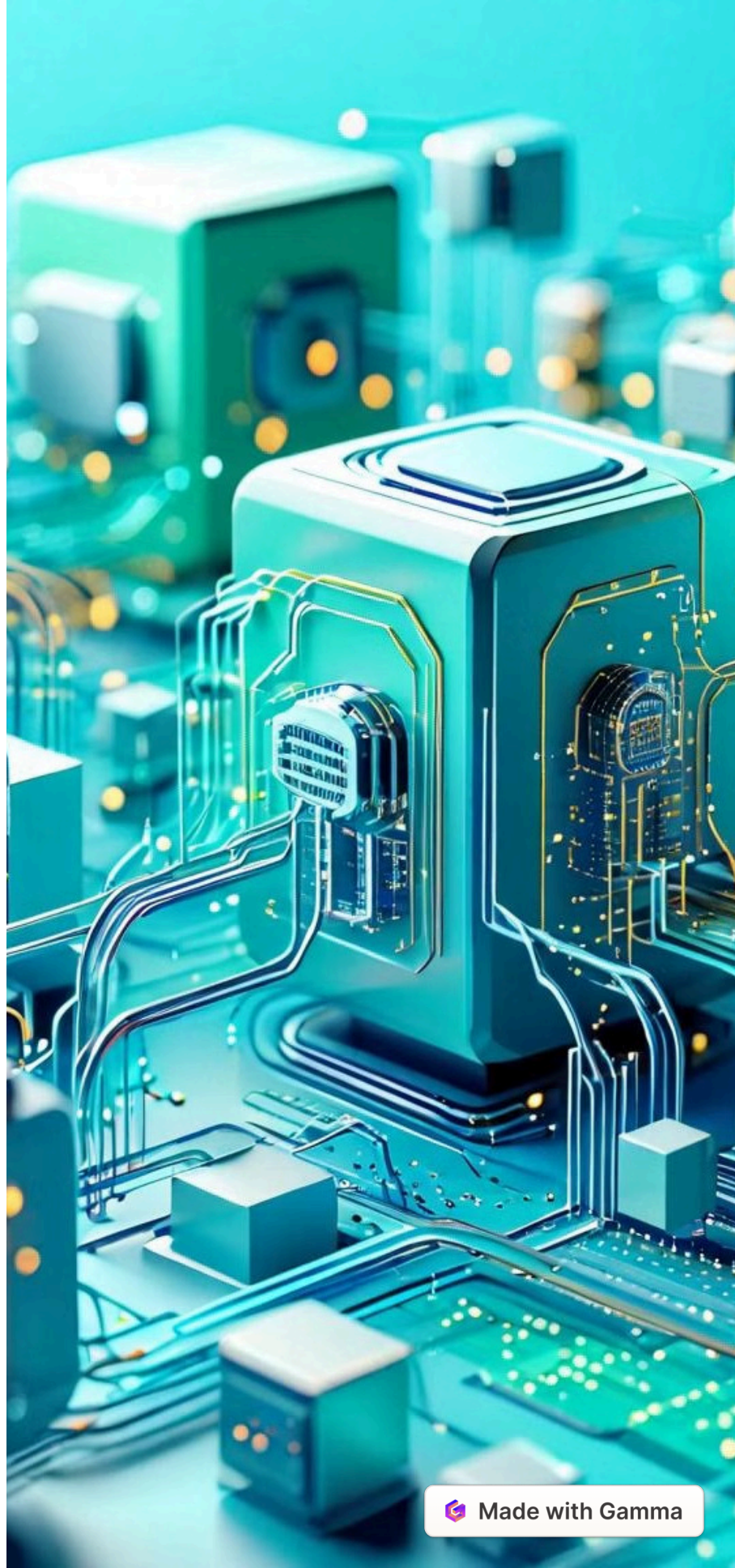
Weaviate Database

Stores and retrieves vectorized Python documentation efficiently.

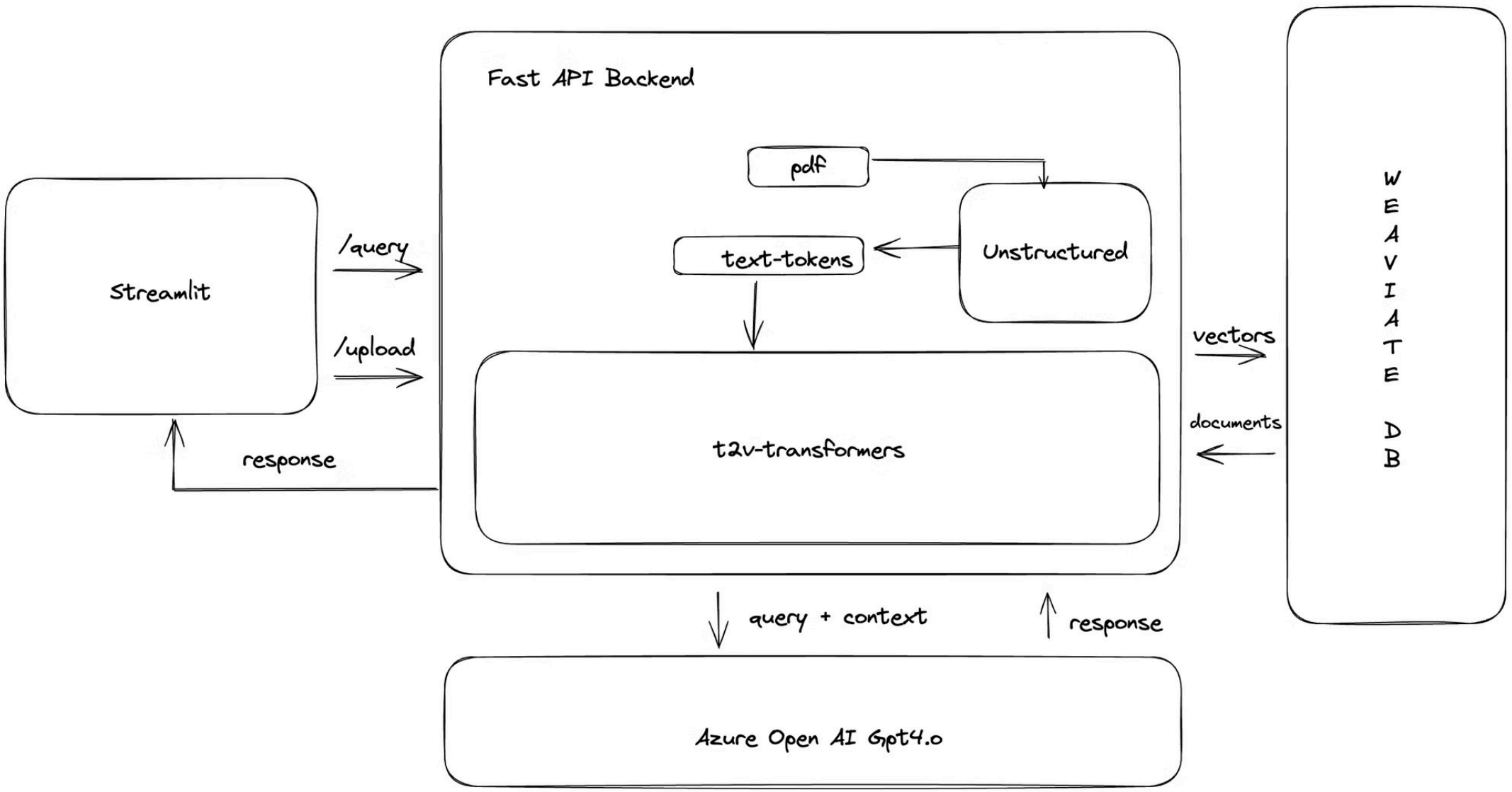
4

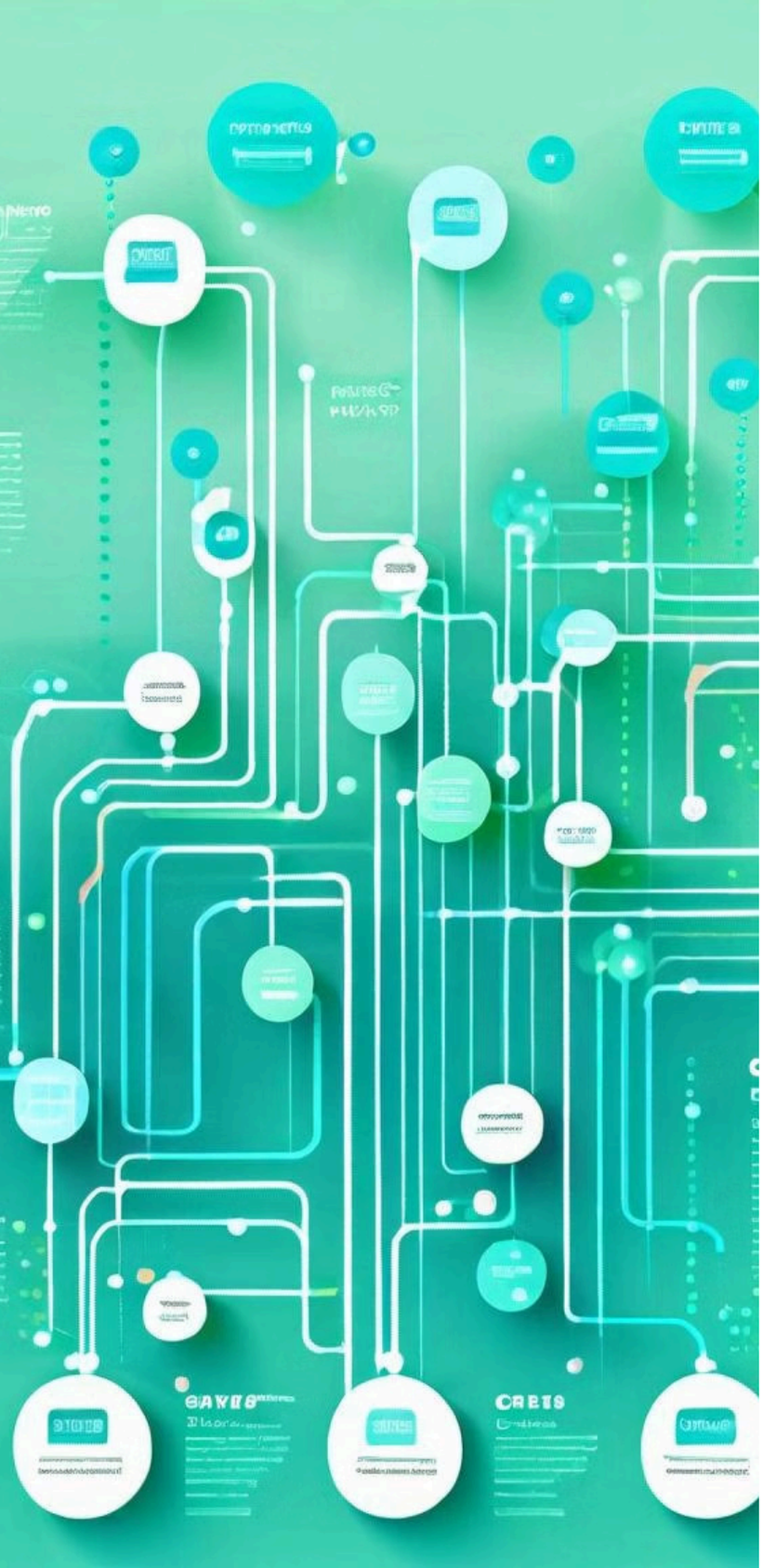
RAG Pipeline

Enhances responses by augmenting them with relevant retrieved information. The file/upload api takes pdfs as input and then clean text is extracted, tokenized & vectors are generated using **transformers** models.



Architecture Diagram





Data Collection and Preprocessing

The foundation of Pyro's knowledge is built on comprehensive Python 3.12 documentation. This data undergoes rigorous preprocessing to ensure optimal performance.

1

Documentation Scraping

Automated tools extract content from official Python 3.12 documentation sources.

2

Text Cleaning

Raw text is cleaned, removing irrelevant information and formatting artifacts.

3

Tokenization

Clean text is tokenized into meaningful units for further processing.

4

Vectorization

Tokenized content is converted into high-dimensional vectors for efficient storage and retrieval.

Retrieval-Augmented Generation (RAG) Pipeline

Pyro employs a sophisticated RAG pipeline to enhance its responses. This system combines the power of retrieval-based and generative AI approaches.

Query Processing

User input is analyzed and converted into a text vector for information retrieval.

Relevant Information Retrieval

The Weaviate database is queried to fetch the most relevant documentation based on these vectors snippets.

Response Generation

GPT-4 generates a response, incorporating the retrieved information and maintaining context.

Performance Evaluation

Pyro's performance is rigorously evaluated using various metrics to ensure high-quality responses. These metrics help identify areas for improvement and track progress.

Metric	Description	Target
Accuracy	Correctness of responses	95%
Response Time	Time to generate an answer	<2 seconds
Relevance	Pertinence to the query	90%

Improving Performance Metrics

Continuous improvement is key to Pyro's success. Various methods are employed to enhance its performance and user experience.

Fine-tuning GPT-4

Regular model updates with Python-specific data to improve response accuracy and relevance.

Expanding Knowledge Base

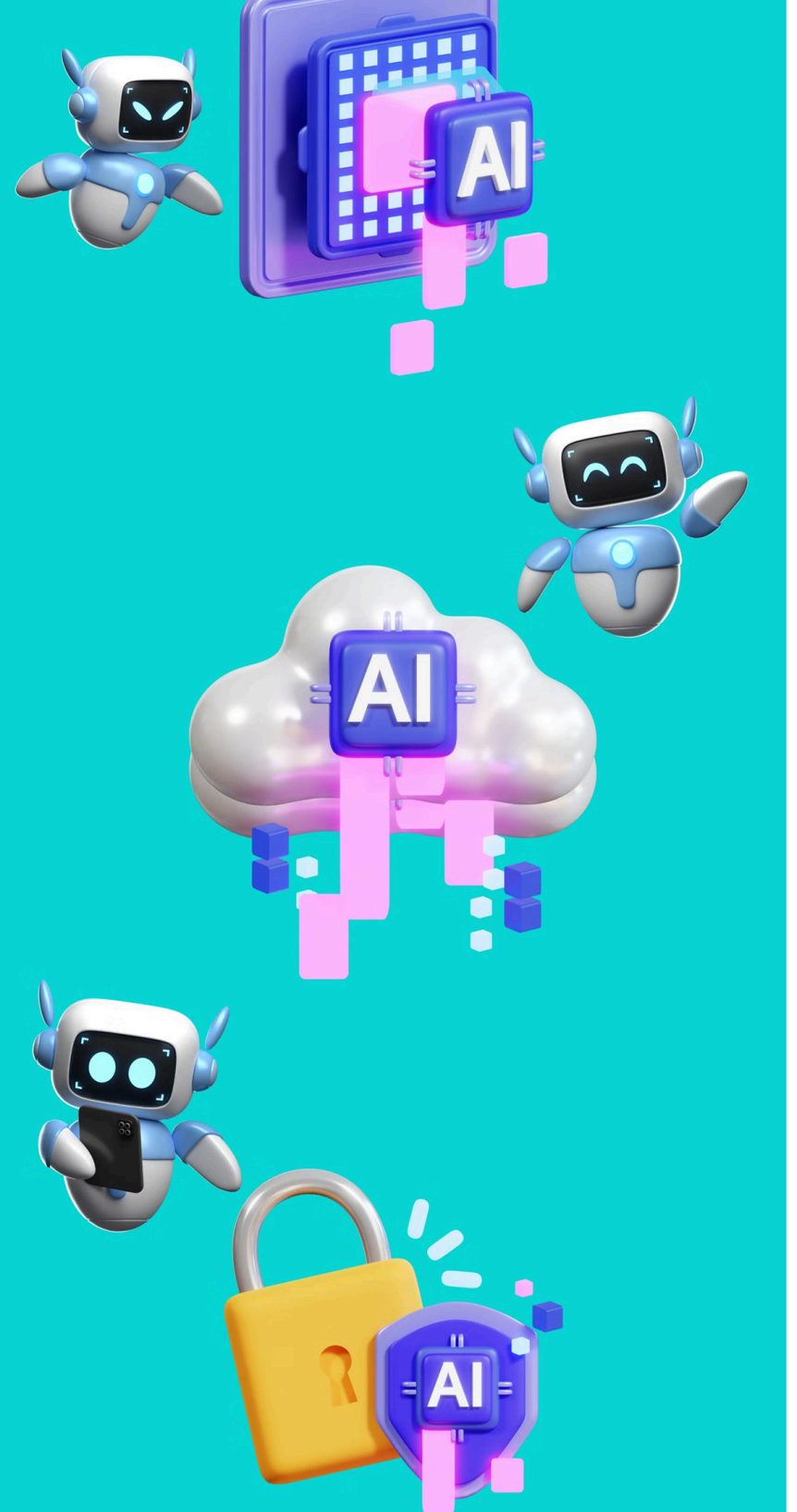
Continuously updating the vector database with the latest Python documentation and community resources.

User Feedback Loop

Incorporating user feedback to identify and address areas of improvement in responses.

Optimizing RAG Pipeline

Refining the retrieval and generation processes for faster and more accurate responses.



Deployment and Integration

Pyro is designed for seamless deployment and integration into various development environments. The goal is to make Python documentation easily accessible to developers.

1

Deployment Workflow

Streamline the deployment process with automated scripts that handle code packaging, infrastructure provisioning, and app deployment to production.

2

Cloud Platform Integration

Leverage cloud platform services like AWS, Azure, or Google Cloud for scalable infrastructure, managed databases, and DevOps tooling.

3

User Feedback Loop

Implement a continuous feedback system to gather user insights, monitor performance, and quickly address any issues or feature requests.

Future Work

Extending Knowledge Base

Continuously expanding the database of more language documentations, open-source libraries, and community resources to provide even more comprehensive support for developers.

Multimodal Interaction

Integrating voice and visual interfaces to enable seamless, hands-free interactions, allowing developers to access information more efficiently.

Integration with IDEs

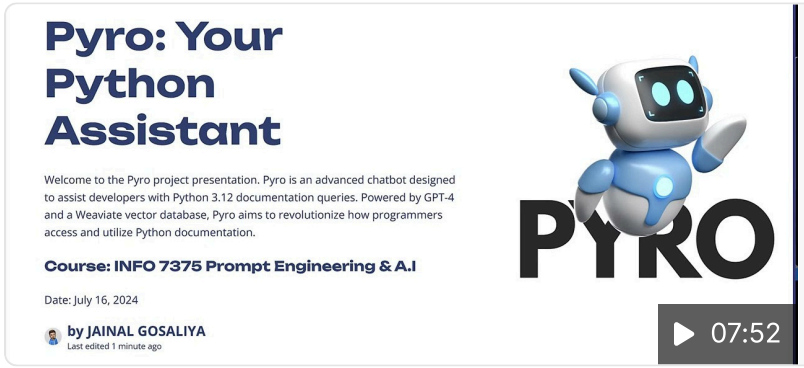
Seamlessly integrating Pyro within popular Python IDEs, such as PyCharm and Visual Studio Code, to provide in-context documentation and assistance.

Demo Time

🤖 Let's see Pyro in action 🚀

Click below to watch the Demo

YouTube Link



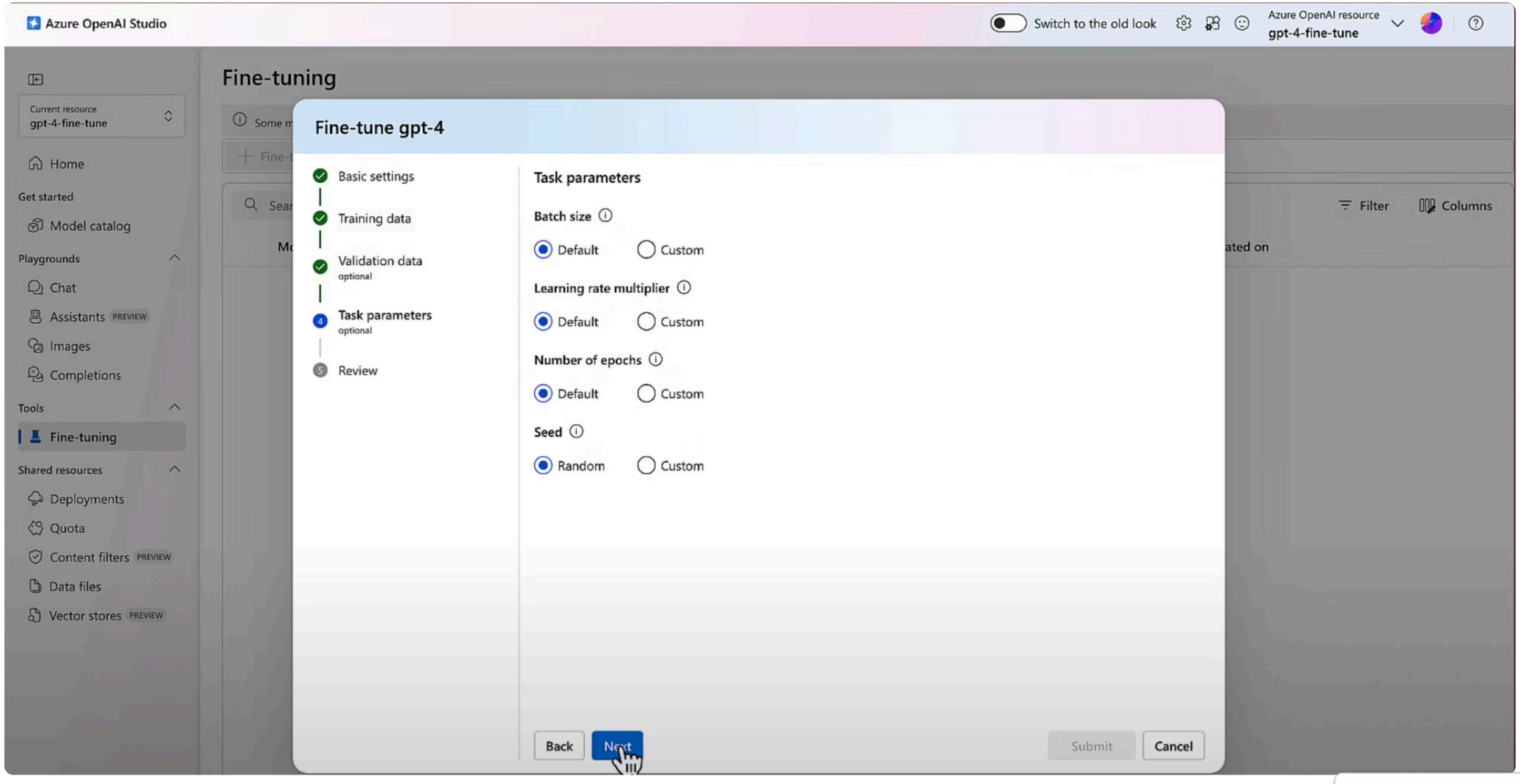
YouTube

Pyro-bot Presentation

Pyro-bot final Presentation with full information on internals, architecture, scope etc

Fine Tuning

Fine tuning on Azure OpenAI studio. This provides a more visual, straightforward and intuitive way to fine-tune our private gpt 4.o model. Resulting in faster, secure and scalable deployments on personal Azure Cloud.

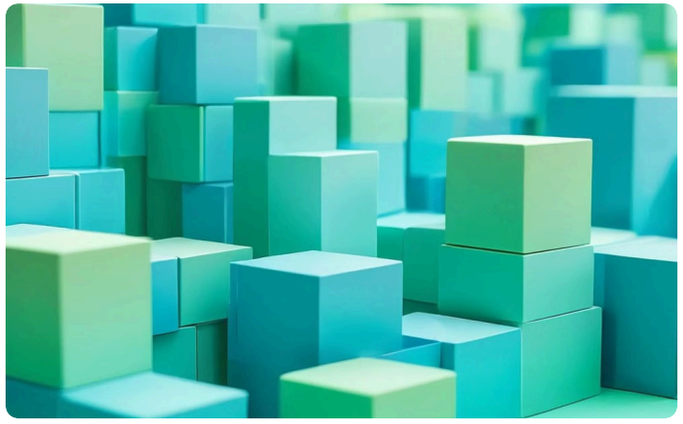


Finetuning Metrics

Lets see some metrics, methods and results for fine-tuning the GPT 4.o model for Pyro

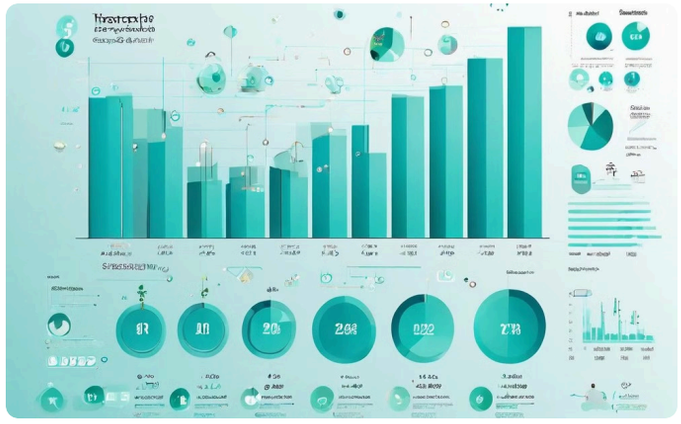
Training Set

The training set included 100 carefully crafted system prompts designed to teach the Pyro model to handle complex Python questions. These prompts were developed using metrics obtained before fine-tuning, which had previously yielded unsatisfactory results



Validation

The validation set consisted of 20 system prompts, specifically curated to assess Pyro's performance on challenging Python questions. These prompts were chosen based on insights from initial evaluations, which highlighted key areas needing improvement.



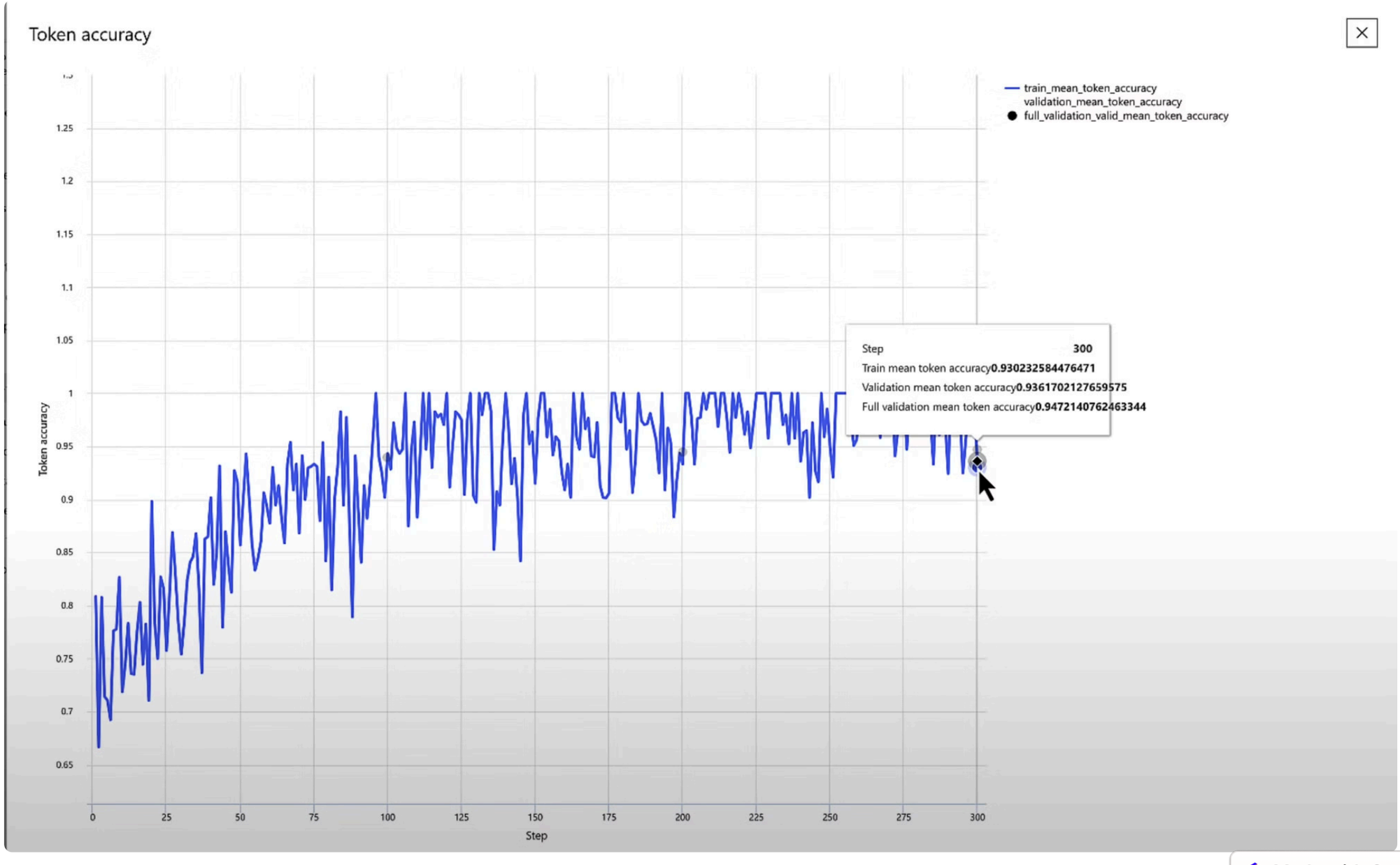
Results

After a finetuning cycle of over 2 hours and over **35,000 training token** usage

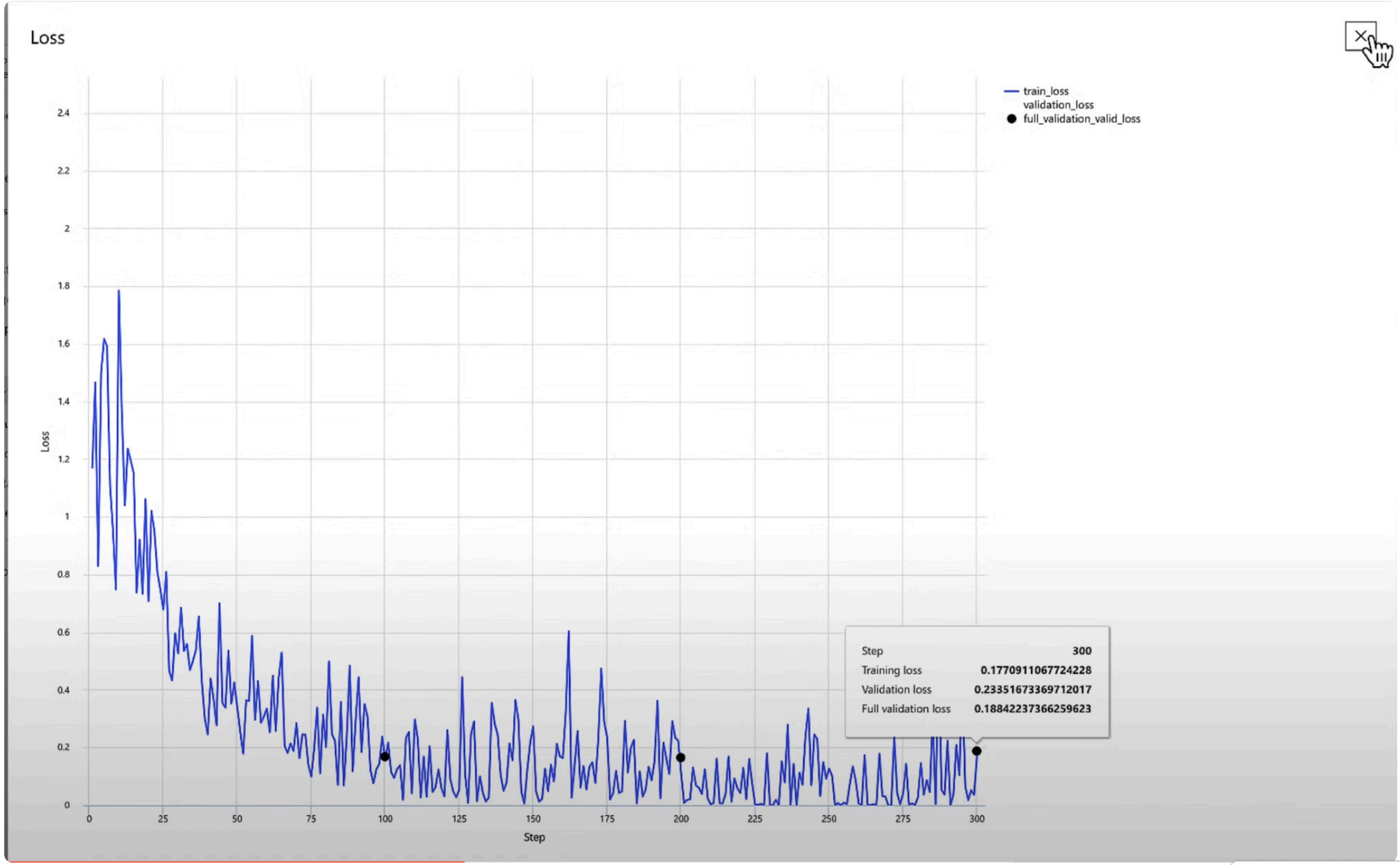
The results are perfect:-

- Full Validation Loss: **18%**
- Full Model Accuracy: **94%**
- Number of epochs used: **300**

Token Accuracy



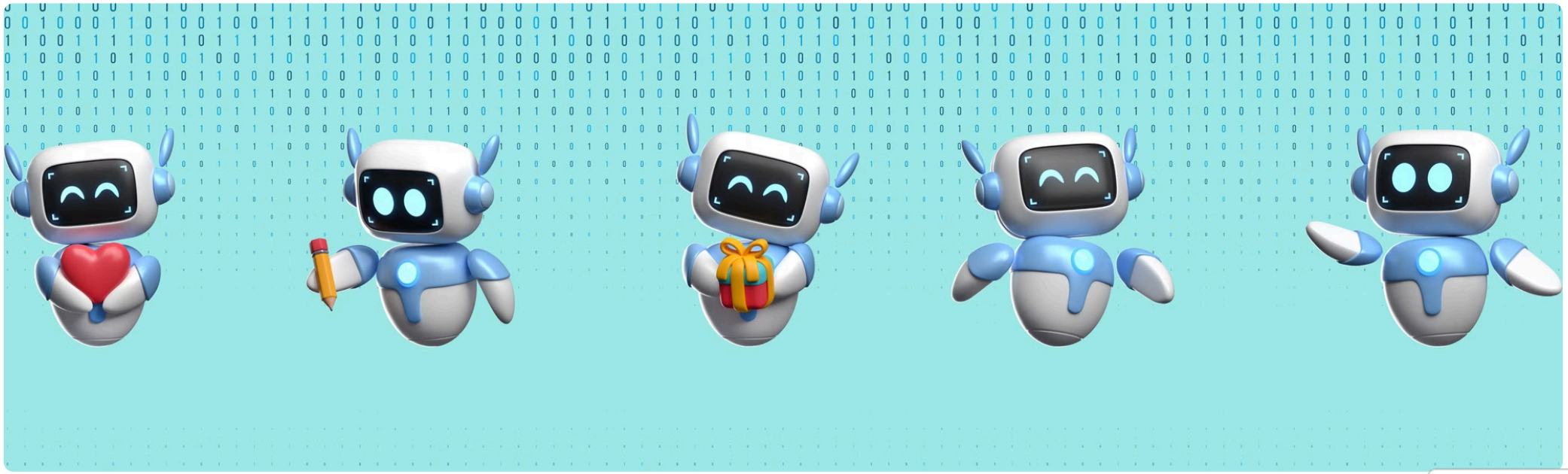
Validation Loss



Conclusion

Pyro has proven to be a valuable Python documentation assistant, providing accurate and timely information to developers. Key takeaways include the importance of continuous performance improvement, seamless deployment, and a roadmap focused on expanding capabilities.

Moving forward, Pyro aims to enhance user experience through features like multi-language support, code generation, and virtual reality integration. As Pyro continues to evolve, it will remain committed to its mission of making Python documentation more accessible and user-friendly for developers of all skill levels.



Q&A: Audience Feedback and Discussion

Gather Insights

Get valuable insights and feedback on areas of improvement and things to improve

Follow up

Committed to addressing any unanswered questions or action items after the presentation.