**Educational AI Assistant**

Personalized Learning with Generative AI

PROJECT DOCUMENTATION

**Team Members:**

1. ABEEB MOHAMMED KASIM.M – Concept Explanation Module
2. GUGAN.D – Quiz Generator and Attender Module
3. GURU BHAVESH.H – Testing & Compiling
4. VISWANATHAN.M– Error Corrections

# Project Overview

Education is transforming through Artificial Intelligence integration. The Educational AI Assistant harnesses Generative AI to provide in-depth topic explanations, dynamic quizzes, immediate feedback, and performance scoring. Utilizing the Granite-3.2 model for content generation, PyTorch and Transformers for inference, and Gradio for the interface, this platform enables personalized learning for students across academic levels, fostering engagement and knowledge retention.

# Objectives

1. To create a personalized learning platform powered by Generative AI.
2. To generate detailed explanations for educational concepts or topics.
3. To provide dynamic multiple-choice quizzes with real-time feedback and scoring.
4. To develop an interactive web interface for seamless user engagement.
5. To ensure clean, structured output from AI-generated content.
6. To deploy the application easily via a web browser or cloud environment.

# System Architecture

The architecture consists of four main modules:

* Concept Explanation Module – Users input topics to receive detailed explanations.
* Quiz Generator and Attender Module – Generates quizzes, accepts answers, and provides feedback with scoring.
* Testing & Compiling – Testing the code and modules and imported liabries & Output.
* Error Corrections-Checking the errors in modules. Resolving errors in coding.

**Workflow:**

User starts the app → Chooses a tab (Explanation or Quiz) → Enters a topic → AI produces content → User engages (views or answers) → Instant feedback/scores shown.

# Module Description

1. Concept Explanation Module
   1. User inputs educational topic

○ AI delivers in-depth explanations with examples

○ Output displayed readably

1. Quiz Generator and Attender Module
   1. User enters quiz topic

○ Generates 5 MCQs with A, B, C, D options

○ Submits answers for instant feedback, scoring, and details

1. Backend AI & Interface
   1. Links to Granite-3.2 for responses

○ Manages prompts for explanations/quizzes

○ Handles Gradio state for answers/interactions

1. Integration & Deployment
   1. Incorporates PyTorch/Transformers for inference

○ Deploys via Gradio

○ Ensures module integration

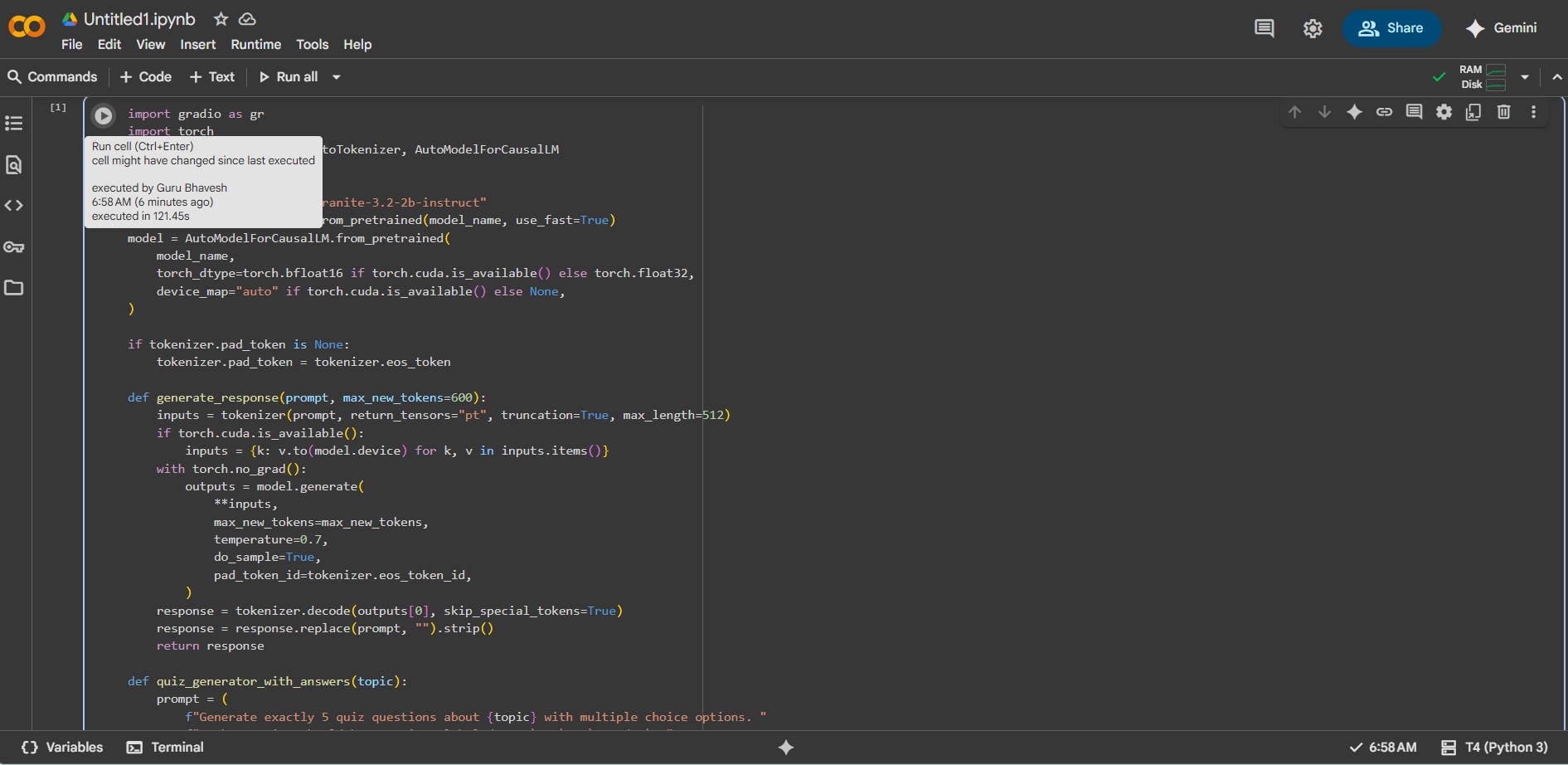
Tools & Technologies Used

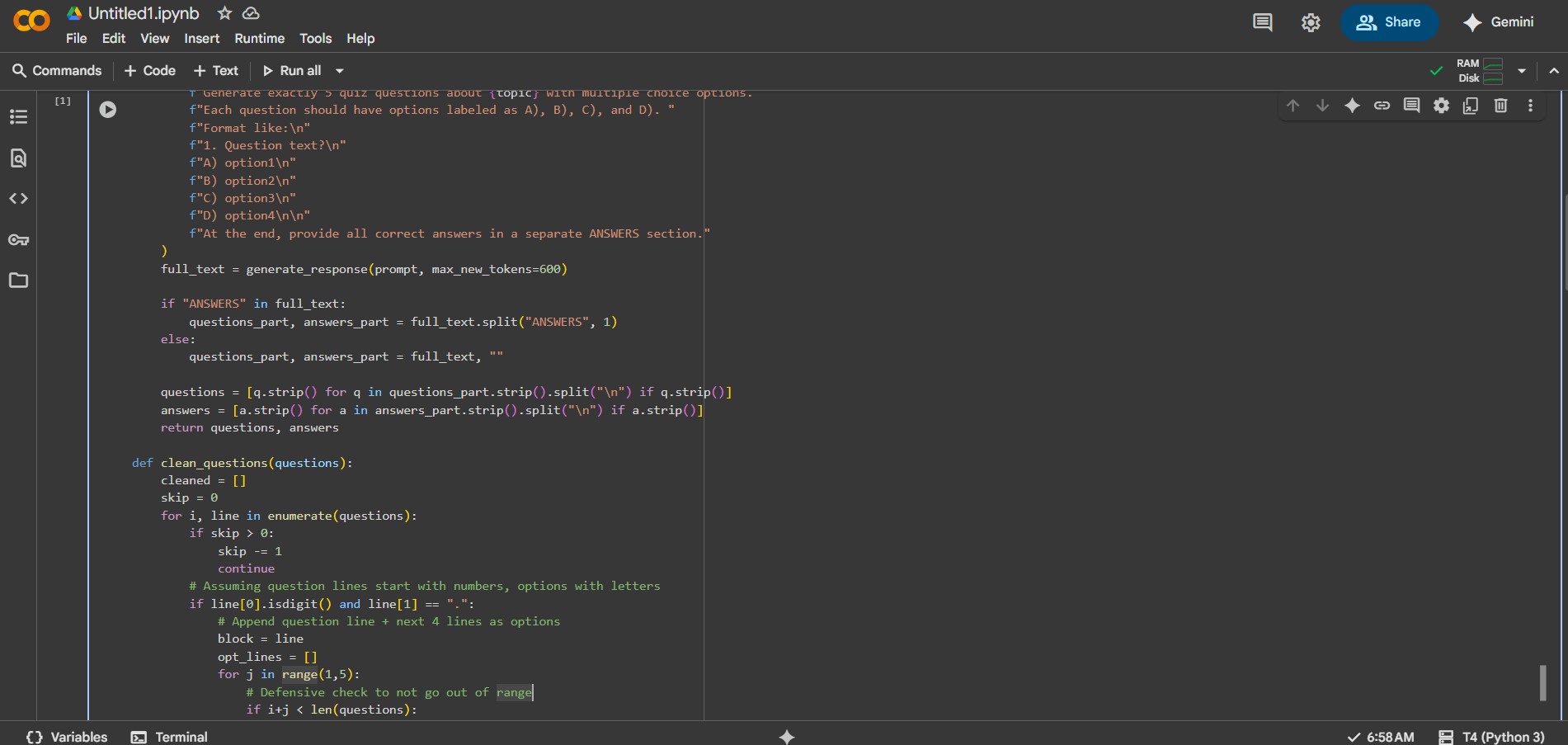
* Python – Programming Language
* Gradio – Web Application Framework
* Granite-3.2 Model – Generative AI for explanations and quizzes
* PyTorch – Model inference and transformations
* Transformers (Hugging Face) – Model loading/interaction
* GitHub – Version Control

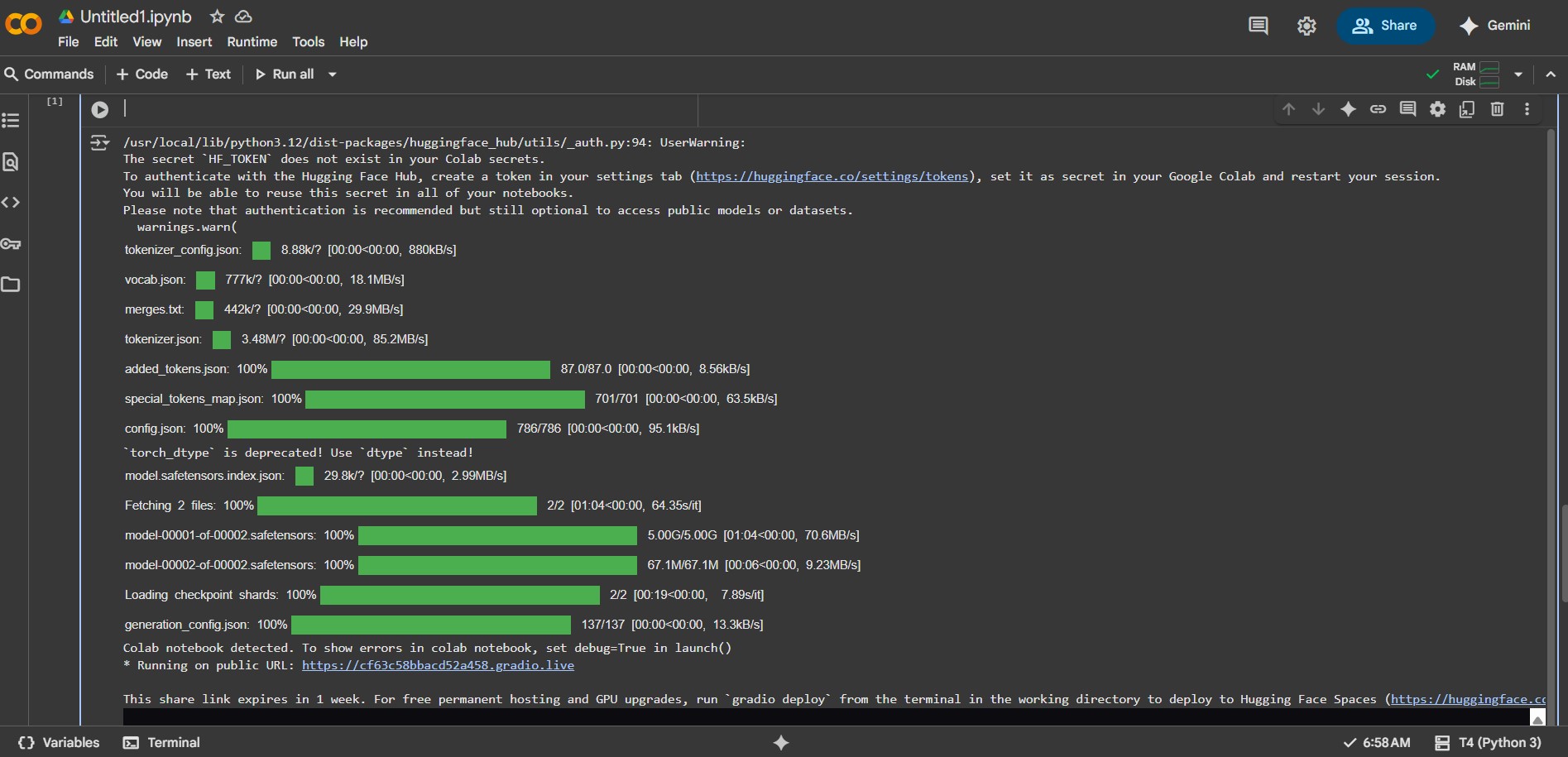
# Implementation Steps

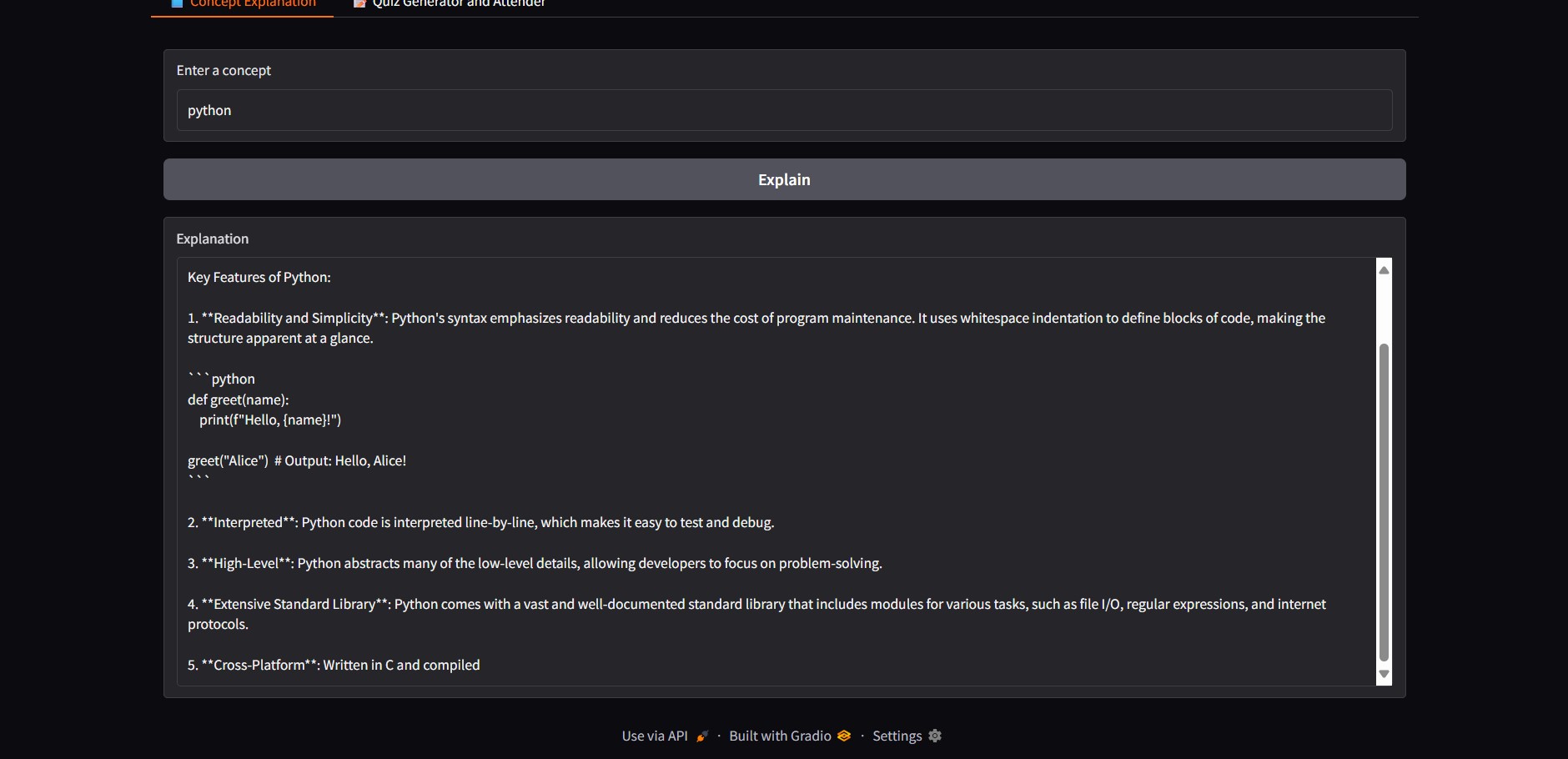
1. Set up environment with Python and libraries.
2. Connect Granite-3.2 via Hugging Face.
3. Build explanation interface for input/display.
4. Develop quiz generator with answer handling.
5. Add feedback/scoring for quizzes.
6. Implement Gradio UI with tabs.
7. Test with samples, refine.
8. Deploy via Gradio.
9. Document and present.

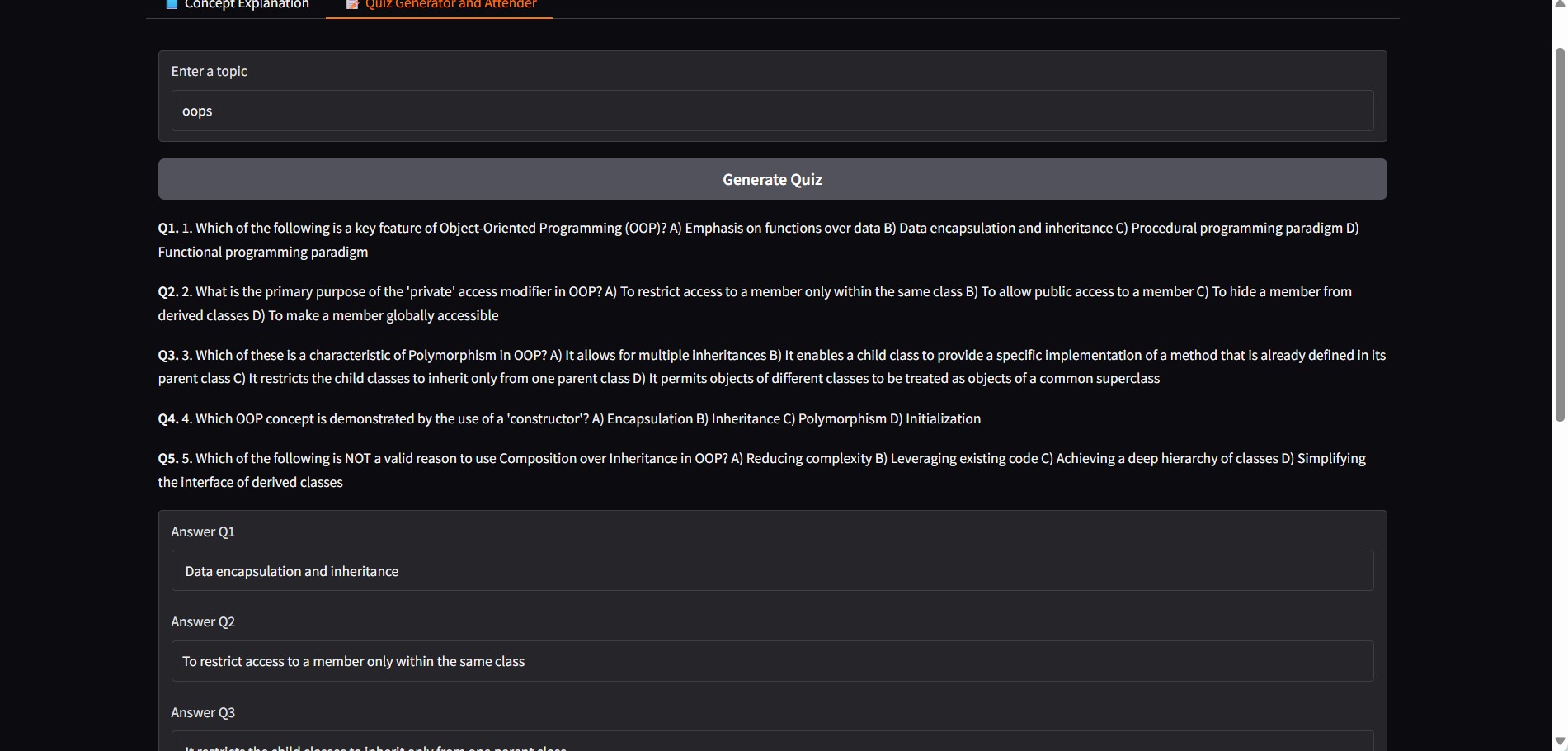
Results & Screenshots:

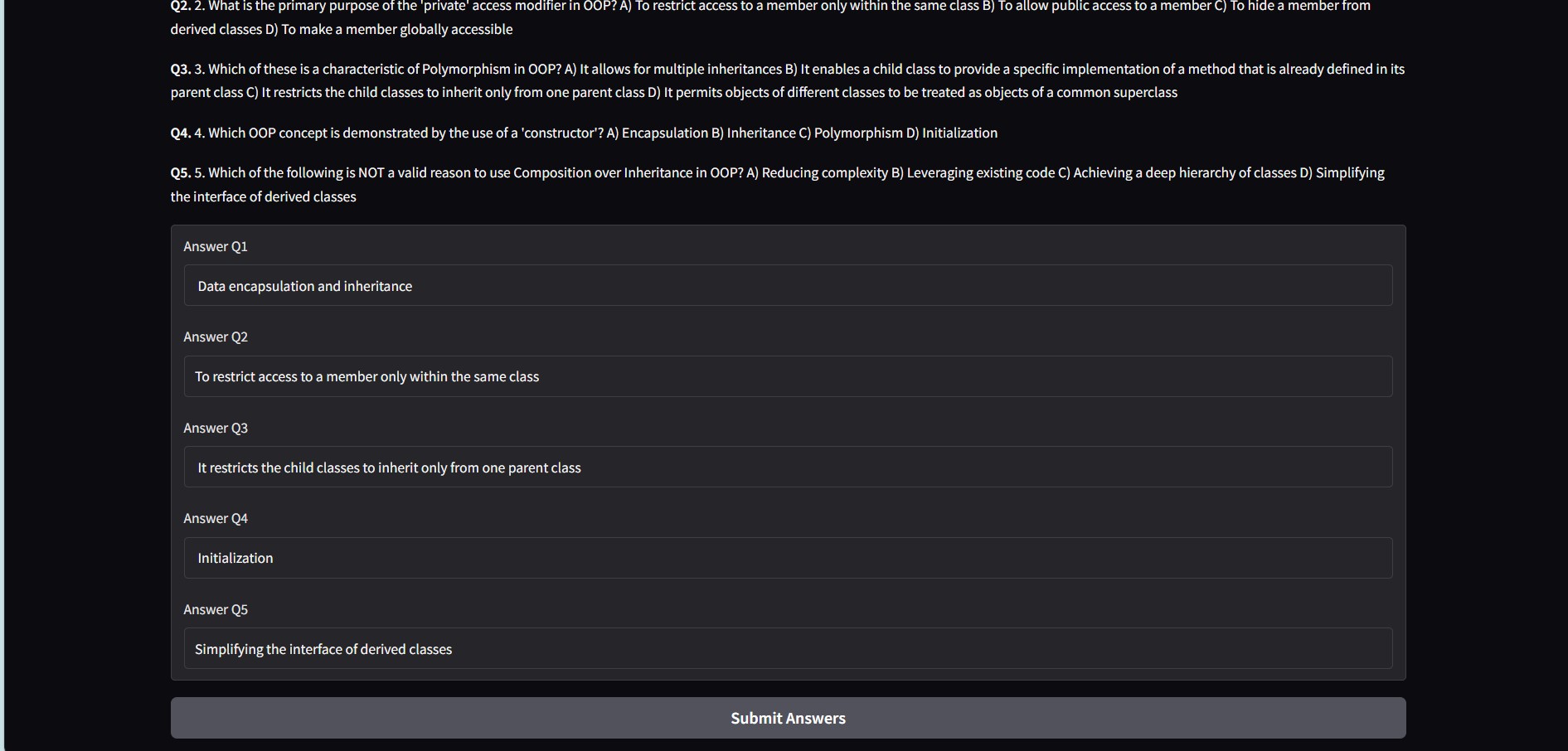












The application successfully:

* Generates explanations automatically.
* Creates formatted quizzes with evaluations.

# Known Issues

During development and testing of the Educational AI Assistant, several potential issues were identified based on the technologies used. These are derived from common reports in the Hugging Face community, Stack Overflow discussions, and official documentation for Granite-3.2 and Gradio integrations. While the project functions as intended in controlled environments like Google Colab with T4 GPU, users may encounter the following:

# Device Mismatch Errors in Gradio with

**Transformers/PyTorch**: When running the app, especially on mixed CPU/GPU setups, a common error occurs where tensors are placed on incompatible devices (e.g., model on GPU but inputs on CPU). This leads to runtime errors like "RuntimeError: Expected all tensors to be on the same device." Solution: Explicitly move inputs and model to the same device (e.g., using to(device) in PyTorch) before inference. This issue is frequently reported in Gradio demos integrating Transformers models.

# Model Loading and Memory Constraints: The

Granite-3.2-2B-Instruct model (2 billion parameters) requires significant RAM/GPU memory (at least 4-6 GB VRAM recommended). On low-resource machines or without GPU acceleration, loading via -----------------------------------------🡪

AutoModelForCausalLM.from\_pretrained() may fail or cause out-of-memory errors. In Google Colab, ensure T4 GPU is selected; for local runs, use quantization (e.g., via torch\_dtype=torch.bfloat16). Inherited from earlier Granite versions, the model may produce inaccurate or biased responses in edge cases, requiring safety testing.

1. **Pipeline Compatibility with Gradio**: For certain tasks like object detection or custom pipelines, gr.Interface.from\_pipeline() may raise "Unsupported pipeline type" errors if the Transformers pipeline isn't directly compatible (e.g., for non-standard tasks). In this project, since we use custom functions for quiz generation

and explanation, this is avoided by defining manual predict functions. However, if extending to other pipelines, verify compatibility.

1. **Generation Quality and Prompt Sensitivity**: As an instruct-tuned model, Granite-3.2 performs well on reasoning tasks but may generate inconsistent or hallucinated content for complex educational topics, especially without refined prompts. Users report occasional degraded performance in multilingual or highly specialized domains. Mitigation: Use structured prompts with <think></think> tags for better reasoning, as recommended in IBM documentation.
2. **Deployment and Sharing Limitations**: When sharing via Gradio's public links (e.g., in Colab), sessions may timeout after inactivity, and high-traffic use can exceed free tier limits on Hugging Face Spaces. Additionally, without user authentication, progress isn't saved across sessions. For production, deploy on Hugging Face Spaces or IBM Cloud, but monitor for API rate limits if integrating external services.

These issues do not prevent core functionality but highlight areas for optimization. The model inherits ethical limitations from prior versions, such as potential biases in outputs, and should be used with safeguards like Granite Guardian for risk detection. Future enhancements can address these through fine-tuning or additional safety layers.

**Future Scope**

Educational AI Assistant showcases Generative AI's role in education by offering personalized explanations, quizzes, and feedback. It meets objectives through integrated generation and interactive UI.

Conclusion

The **Educational AI Assistant** project demonstrates the powerful potential of combining advanced AI language models with interactive web interfaces to enhance learning experiences. By leveraging the Granite-3.2 model in a Gradio application, the assistant effectively supports users with detailed concept explanations and dynamically generated quizzes tailored to chosen topics. This fosters active learning and provides immediate feedback, essential for student engagement and mastery.

This project highlights the practicality of AI in education — offering scalable, personalized educational assistance without the constraints of human resource limitations. It shows how carefully designed prompts and interface logic can create clear, structured educational content from AI-generated text.

Future developments can further improve interactivity and accessibility, expanding to diverse question formats, adaptive learning paths, and multimodal content delivery. Overall, the project affirms that

AI-powered educational tools can augment traditional methods, helping learners achieve their goals more efficiently and confidently.