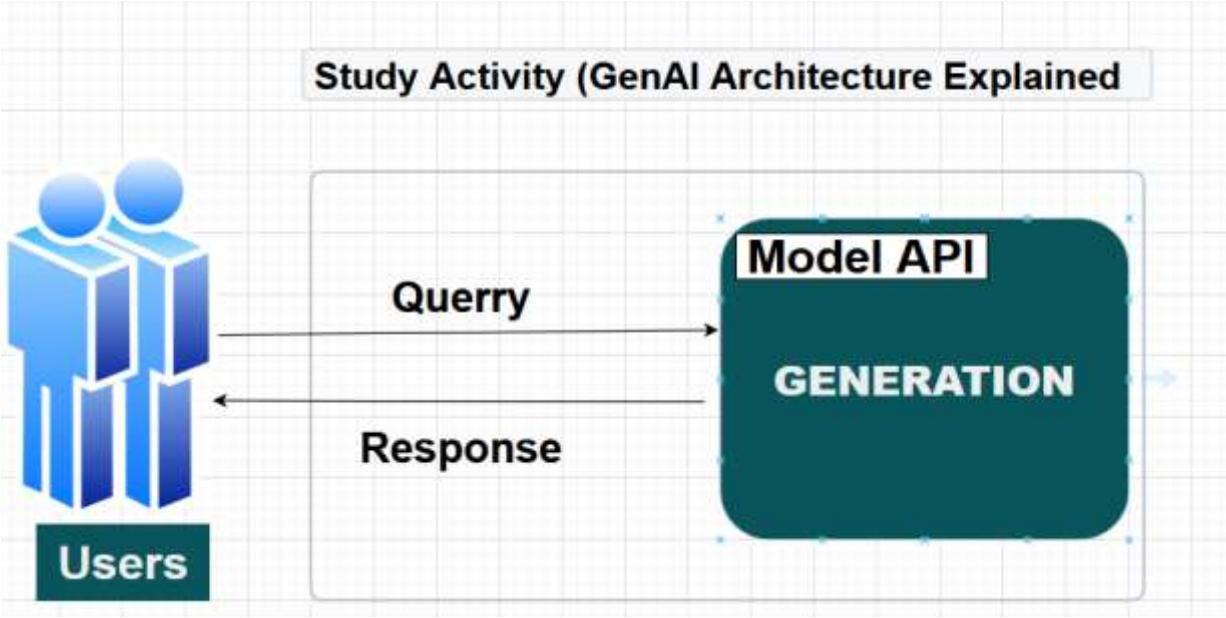
AWS-Based Generative AI Architecture for a Secure Language Learning Application

Business Goal

This is a high-level conceptual architecture diagram that highlights the core features of a Language Learning Application on AWS Cloud. The architecture is designed for non-technical, business-minded stakeholders, providing a clear explanation of the workflow and AWS services involved in the proposed language learning solution.

Step 1: Simple Generative AI Architecture

- Student Interaction: A student submits a query to the AI model through the web or mobile app.
- API Routing: AWS API Gateway securely routes the request to the backend.
- AI Processing: The Generative AI Model (Amazon Bedrock) processes the query and generates a response.



Step 2: RAG (Retrieval-Augmented Generation) Pipeline Implementation

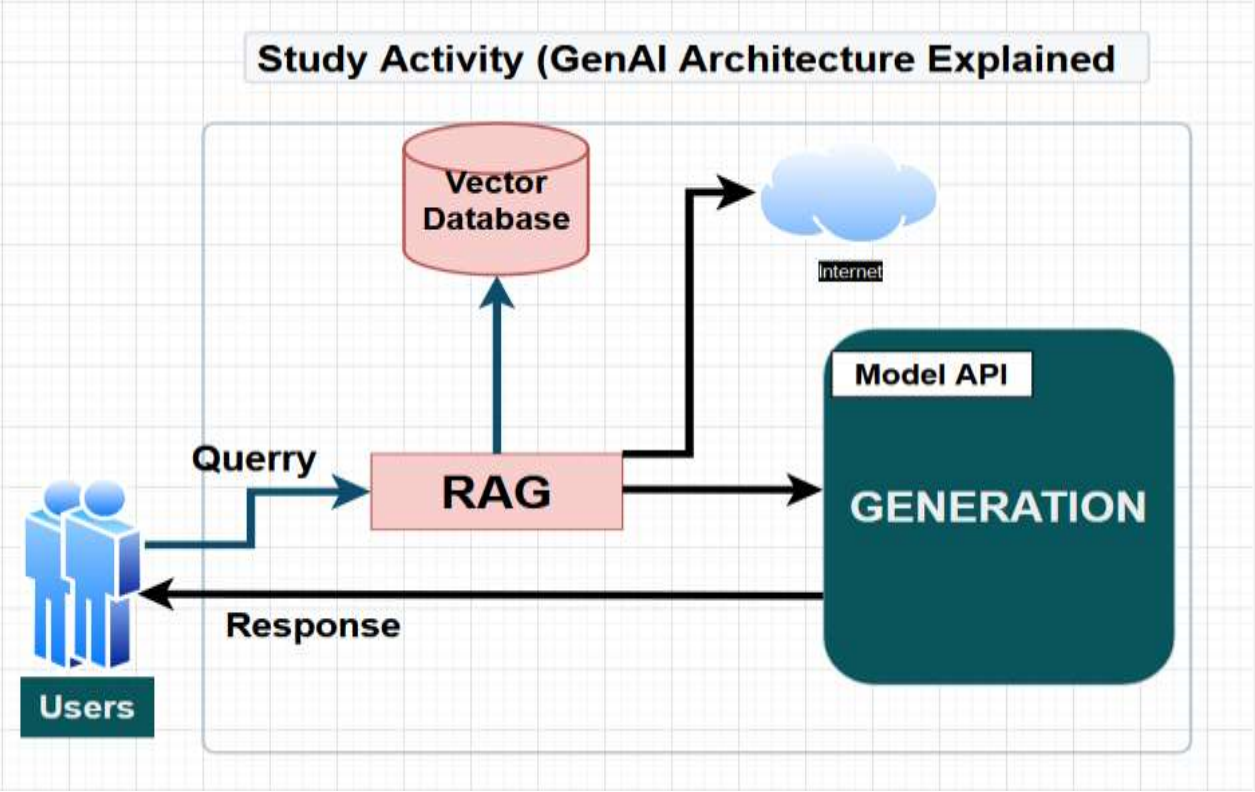
To enhance the application's ability to access up-to-date and specialized information, we have opted to implement RAG.

Why RAG?

RAG allows the AI model to retrieve relevant data from external sources (such as a database or the internet) before generating a response. This approach provides more accurate and current responses, enables customization with customer-specific data and it is cost-effective compared to fine-tuning, as it avoids retraining large models.

How It Works

Student submits a query through the application. If relevant data is needed, the model retrieves it from a vector database (e.g., Amazon OpenSearch Service) for structured information or from the internet via API calls (using AWS Lambda & API Gateway.



Step 3: Implementing Security Guardrails

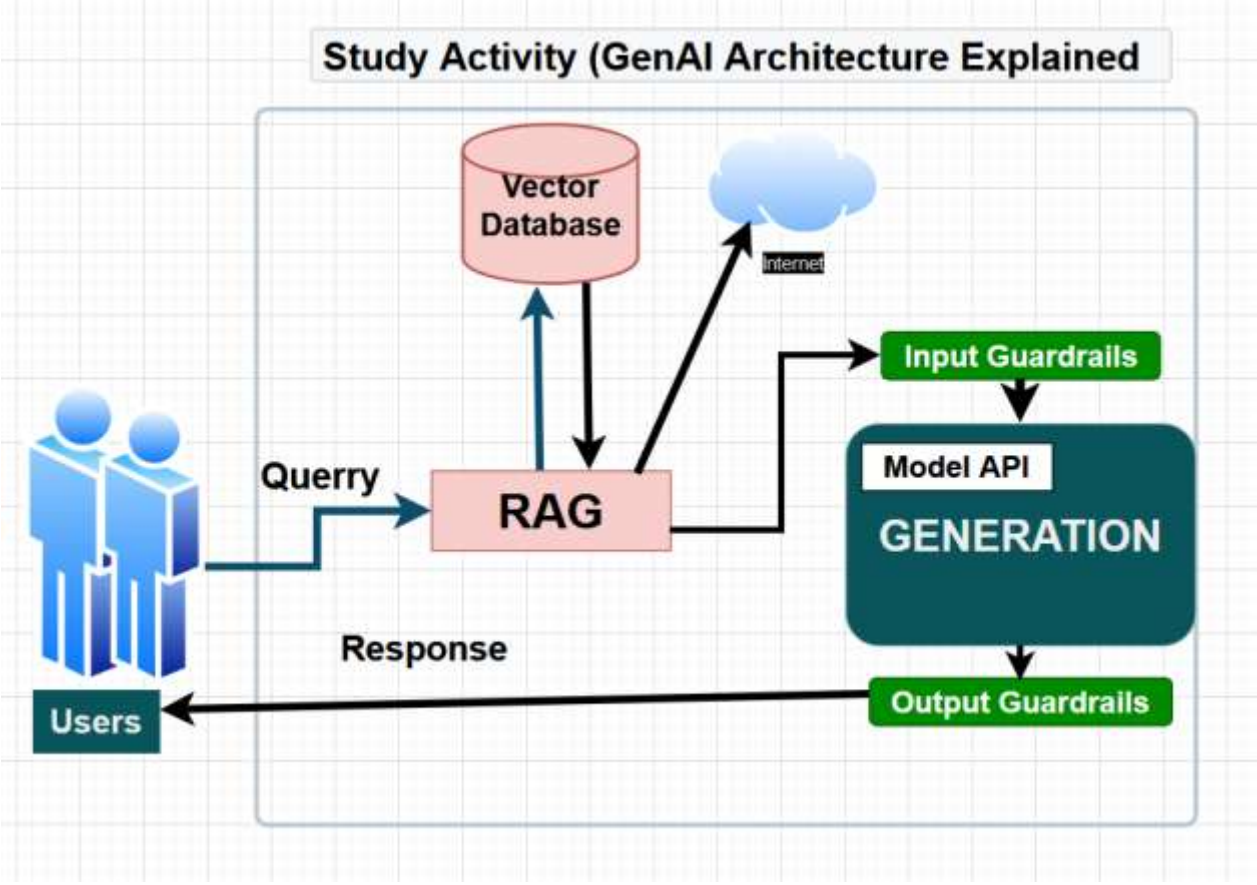
To ensure a safe and secure language learning experience, I have integrated multiple security guardrails at both the input and output stages.

i. Input Guardrails block harmful or inappropriate user inputs

- AWS WAF will prevent malicious queries, such as SQL injection
- Amazon GuardDuty will detect and alert on potential security threats, including unauthorized access or anomaly detection.

ii. Amazon Bedrock – Processes retrieved context (via RAG) and generates a personalized response.

iii. Output Guardrails – Amazon Comprehend will scan and filter the generated response to ensure no toxic, biased, or non-compliant content is sent out and that responses align with educational guidelines.



Step 4: Prompt Caching with Amazon ElastiCache (Redis)

To enhance efficiency and performance, we have integrated a prompt cache into the architecture using Amazon ElastiCache (Redis).

This will store frequently used querries and responses

Reduces redundant LLM processing , improving speed and cost efficiency.