Edu Tutor

Project documentation

1.Introduction

- Project Title : Edu Tutor AI: Personalized Learning
- Team leader : Pavithra R
- Team member : Hemalatha R
- Team member : Madhavi G
- Team member : Kirthika M

2.Project Overview

•Purpose:

The purpose of the EduTutor AI project is to use **IBM Granite models** to provide personalized learning support. It helps students by generating clear **concept explanations**, creating **quizzes** for practice, and offering **interactive learning** through a simple Gradio interface. Running on **Google Colab with GPU support**, the system is easy to access and deploy, while also encouraging students to explore modern AI tools and version control with GitHub.

•Features:

EduTutor AI offers key features that make learning simple and engaging. It provides **AI-generated concept explanations** to help students understand topics better and **automatically creates quizzes** for self-assessment. The system uses a **Gradio interface** for interactive learning and runs efficiently on **Google Colab with GPU support**. By integrating **IBM Granite models** from Hugging Face and using **GitHub for version control**, the project ensures scalability, reliability, and collaboration.

3.Architecture

The architecture of EduTutor AI is designed to ensure smooth learning experiences with simple integration. It begins with the **Naan Mudhalvan Smart Interz portal**, which provides access to project resources. The **IBM Granite models** from Hugging Face form the AI core, delivering concept explanations and quizzes. The application is executed in **Google Colab with T4 GPU support**, where dependencies are installed and the model runs efficiently. Finally, the project files and code are uploaded to **GitHub** for version control and collaboration.

4.Setup Instructions

To set up EduTutor AI, first log in to the **Naan Mudhalvan Smart Interz portal** and access the project workspace. Then, select an **IBM Granite model** such as `granite-3.2-2b-instruct` from Hugging Face. Open a notebook in **Google Colab**, enable the **T4 GPU**, and install the required libraries. Run the provided code to launch the application, and finally download the project file to upload and manage it in **GitHub**.

5.Folder Structure

ϖnotebooks/
Jupyter notebooks used in Google Colab for running and testing EduTutor AI

ϖsrc/
Core source code

- data_preprocessing/ (scripts for input handling and formatting)
- models/ (scripts for loading and running IBM Granite models)
- app/ (Gradio application code)
- utils/ (helper functions and utilities)

ϖmodels/
Downloaded and cached IBM Granite models (e.g., `granite-3.2-2b-instruct`)

ϖoutput/
Gradio application outputs, results, and logs

ϖconfigs/
Configuration files (e.g., model setup, runtime settings)

ϖ**requirements.txt**
List of Python dependencies required for EduTutor AI

ϖ**README.md**
Project overview and setup instructions

.envϖ
Environment variables such as Hugging Face API keys

6.Running the Application

To run the EduTutor AI application, first open the **Google Colab notebook** and make sure the required dependencies are installed. Begin by setting the runtime to **T4 GPU** for faster execution, then run the installation command for libraries such as Transformers, Torch, and Gradio. Once the setup is complete, execute the provided EduTutor AI code, which will automatically download the selected **IBM Granite model** from Hugging Face. After initialization, the system launches a **Gradio web application link** in the Colab output. By clicking the link, users can access the EduTutor AI interface in a new browser tab, where they can input queries, generate quizzes, and receive AI-powered explanations. The project code can also be exported as a `.py` file and uploaded to **GitHub** for version control and future use.

7.API Documentation

The API documentation for EduTutor AI provides details on how users and developers can interact with the system's functionalities. It describes how to send requests and receive responses for tasks such as **submitting learning queries**, generating **concept explanations**, or creating **quizzes**. The documentation specifies the required input formats, usually in JSON, and outlines the structure of the responses, which may include explanations, quiz questions, and answer options.

Authentication can be handled through **Hugging Face API keys**, which must be included to access the IBM Granite models securely. The documentation also covers **error handling**, listing common HTTP status codes and messages (e.g., `200 OK` for successful requests or `400 Bad Request` for input errors). Developers can integrate EduTutor AI's services into other platforms by following these API guidelines. In practice, the Colab notebook exposes the Gradio app as an interactive interface, while the backend functions can be adapted into REST APIs for broader use.

8.Authentication

Authentication in EduTutor AI ensures that only authorized users can access the application and its AI services. Since the system relies on **IBM Granite models hosted on Hugging Face**, users must provide a valid **Hugging Face API key** to download and run the models securely. This key acts as a credential that verifies the user’s identity and usage rights. The API key should be stored in a secure **.env file** or as an environment variable within Google Colab to prevent unauthorized access.

For version-controlled projects on **GitHub**, authentication mechanisms like personal access tokens may be required to push or pull updates securely. Additionally, when the EduTutor AI application is extended to a production environment, **role-based access control (RBAC)** can be implemented to restrict functionalities based on user roles (e.g., student, teacher, or admin). Secure transmission protocols such as **HTTPS** must also be used to protect sensitive data during communication. Overall, authentication mechanisms safeguard the integrity of EduTutor AI, ensuring that resources and model services are used responsibly.

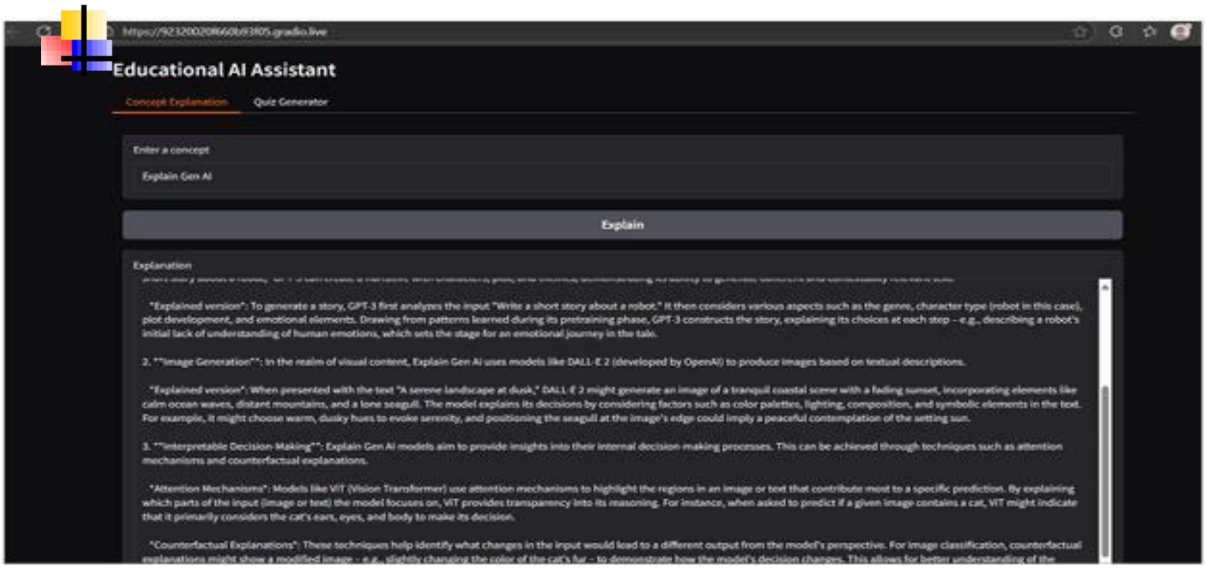
9.User Interface

The **Edu Tutor AI** user interface is designed to provide students with a personalized and interactive learning experience. The dashboard greets the student, shows their progress, and recommends lessons based on their learning history. A dedicated AI Tutor panel allows users to ask questions and receive step-by-step guidance with hints and explanations. Lesson pages display content such as videos, text, and exercises, while quizzes provide instant feedback to track understanding. The interface includes a clear header with navigation and profile options, and a footer with help and contact links. Overall, the design is clean, responsive, and accessible, ensuring an engaging and efficient learning environment.

10.Testing

EduTutor AI is a personalized learning application developed using IBM Granite models from Hugging Face. The project focuses on creating simple yet effective educational tools such as concept explainers and quiz generators. It is implemented in Google Colab for easy setup and reliable performance, with Gradio used for building an interactive interface. To begin, learners first explore the Naan Mudhalvan Smart Interz Portal, where they can enroll in the project and access resources and guided instructions. Next, an IBM Granite model is selected from Hugging Face; in this project, the lightweight and efficient **granite-3.2-2b-instruct** model is chosen. The application is then built in Google Colab by setting up the runtime with a T4 GPU, installing necessary libraries such as transformers, torch, and gradio, and running the provided code to launch the application. Once executed, the model downloads automatically and the Gradio app link becomes available for real-time interaction. Finally, the completed project is uploaded to GitHub by creating a repository, downloading the Colab code as a .py file, and committing it to the repository for sharing and version control. Through this workflow, EduTutor AI demonstrates the integration of IBM's Granite models with user-friendly tools, enabling a practical, hands-on approach to personalized learning.

OUTPUT



Educational AI Assistant

Enter a concept

Explain Gen AI

Explanation

"Explained version": To generate a story, GPT-3 first analyzes the input "Write a short story about a robot." It then considers various aspects such as the genre, character type (robot in this case), plot development, and emotional elements. Drawing from patterns learned during its pretraining phase, GPT-3 constructs the story, explaining its choices at each step – e.g., describing a robot's initial lack of understanding of human emotions, which sets the stage for an emotional journey in the tale.

2. **Image Generation**: In the realm of visual content, Explain Gen AI uses models like DALL-E 2 (developed by OpenAI) to produce images based on textual descriptions.

"Explained version": When presented with the text "A serene landscape at dusk," DALL-E 2 might generate an image of a tranquil coastal scene with a fading sunset, incorporating elements like calm ocean waves, distant mountains, and a lone seagull. The model explains its decisions by considering factors such as color palettes, lighting, composition, and symbolic elements in the text. For example, it might choose warm, dusky hues to evoke serenity, and positioning the seagull at the image's edge could imply a peaceful contemplation of the setting sun.

3. **Interpretable Decision-Making**: Explain Gen AI models aim to provide insights into their internal decision-making processes. This can be achieved through techniques such as attention mechanisms and counterfactual explanations.

"Attention Mechanisms": Models like ViT (Vision Transformer) use attention mechanisms to highlight the regions in an image or text that contribute most to a specific prediction. By explaining which parts of the input (image or text) the model focuses on, ViT provides transparency into its reasoning. For instance, when asked to predict if a given image contains a cat, ViT might indicate that it primarily considers the cat's ears, eyes, and body to make its decision.

"Counterfactual Explanations": These techniques help identify what changes in the input would lead to a different output from the model's perspective. For image classification, counterfactual explanations might show a modified image – e.g., slightly changing the color of the cat's fur – to demonstrate how the model's decision changes. This allows for better understanding of the

Conclusion

In conclusion, EduTutor AI demonstrates how IBM Granite models can be effectively integrated with Google Colab and Gradio to build simple yet impactful personalized learning tools. By combining AI-driven explanations and quizzes with an easy deployment workflow, the project highlights the potential of generative AI in education while ensuring accessibility and scalability through GitHub sharing.

THANK YOU