**EDU TUTOR AI**

**Introduction**

**Team NM\_ID:NM2025TMID01891)**

**Team Members : V R SANDHYA(TL) (4A7F63F004F2799EFD4DF0672FD38FD)**

**S SAKTHI : (D236451AFAB4A24BA794D8EF98A8B643)**

**P SREE JEGADEESWARI : (514E32A481322E996C14A68BBA312BBF )**

**S TAMILSELVI : (A59091E27F5979EB4A647F002DDE5F93 )**

**V VINOTHINI : (28269151D4B53BA89D341E8FFD827EB4**)

**DOCUMENTATION**

Documentation for Educational AI AssistantTable of

ContentsIntroductionSystem OverviewFeaturesDependencies   
and RequirementsInstallation GuideCode Structure   
OverviewDetailed Code WalkthroughModel SetupTokenizer   
SetupUtility FunctionsConcept Explanation FunctionQuiz   
Generator FunctionGradio InterfaceApplication WorkflowUser   
Interaction FlowError Handling and Edge CasesPerformance   
ConsiderationsDeployment GuidePossible   
EnhancementsSecurity ConsiderationsTesting

StrategyExample Usage Screenshots (Placeholder)Future

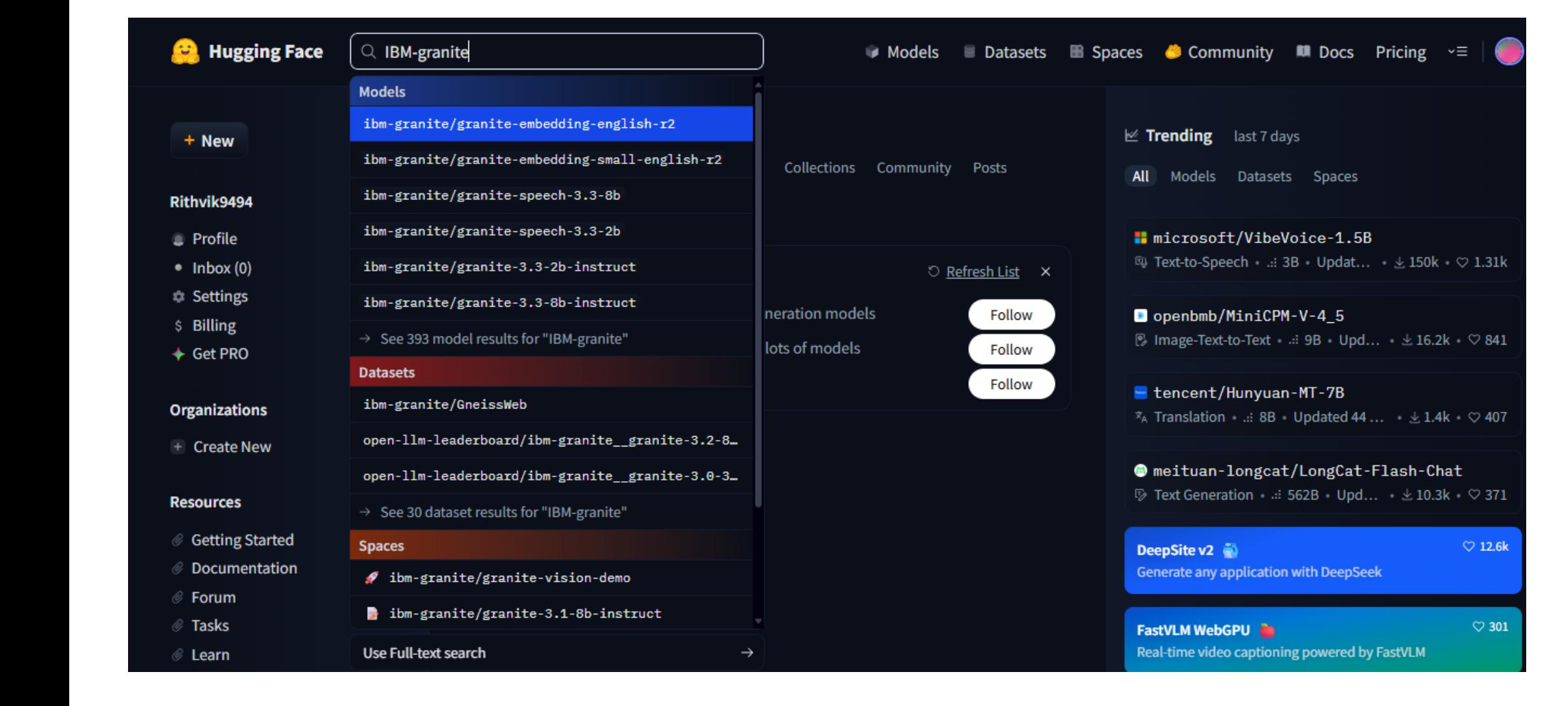
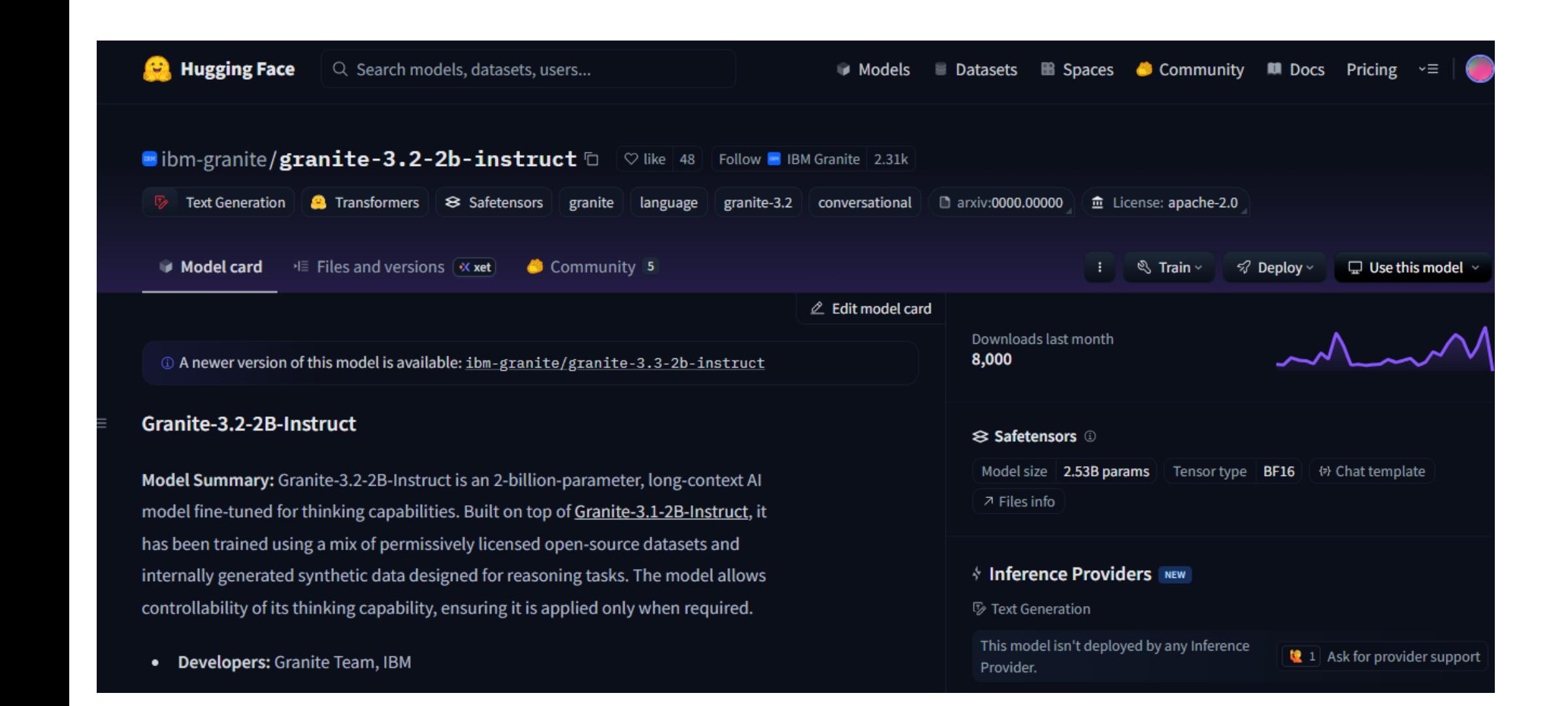
ScopeConclusion1. IntroductionThis project is an AI-powered   
Educational Assistant that leverages large language models

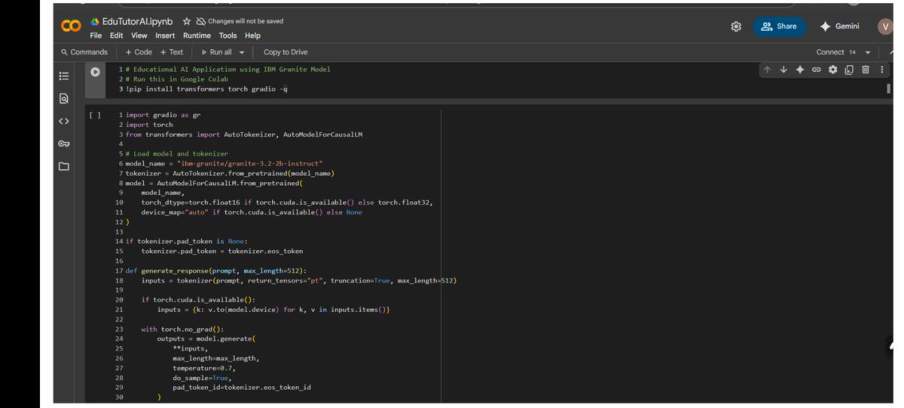
(LLMs) to explain academic concepts and generate quizzes

interactively using a web-based UI powered by Gradio. The   
primary model used is IBM’s Granite family of open-weight   
LLMs.The application’s key purposes are:Enable   
students/learners to understand complex concepts   
interactively.Provide quizzes for self-assessment.Serve as a   
lightweight educational tool with real-time AI assistance.2.

System OverviewThe system integrates:Hugging Face   
Transformers: To load and run the pretrained large language   
model.PyTorch: To run inference on CPU or GPU for faster   
generation.Gradio: To create a user-friendly, shareable web   
interface.Users can type in concepts or topics, and the model   
will return detailed explanations or custom quizzes.3.

FeaturesConcept Explanation: Detailed breakdown of any   
concept with real-world examples.Quiz Generator: Creates 5

diverse questions with different formats (MCQ, True/False,   
short answer).User-friendly Interface: Powered by Gradio Tabs



for easy navigation.Cross-device Support: Shareable Gradio   
links allow users to access the app in a browser.GPU

Utilization:

If CUDA is available, the model runs efficiently on GPU.4.   
Dependencies and RequirementsProgramming Language:

Python 3.8+Libraries Used:torch (PyTorch)transformers   
(Hugging Face LLM toolkit)gradio (Web interface)Hardware

Requirements:CPU-only mode: Basic usage possible with

slower generation.GPU (Recommended): CUDA-enabled device

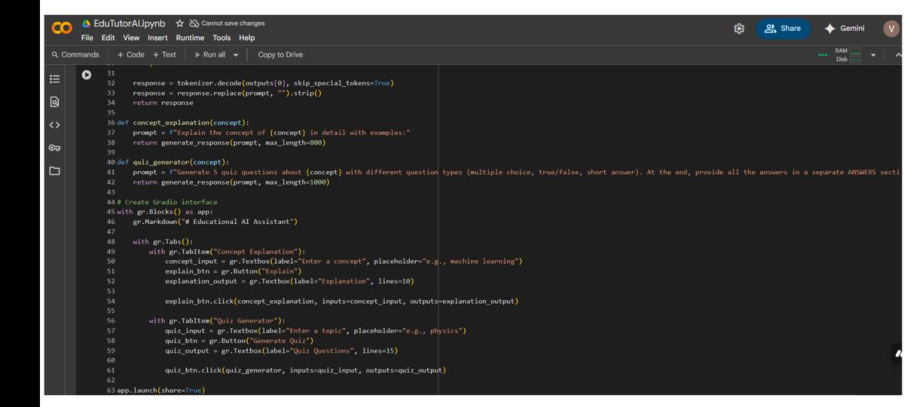
for better performance.5. Installation Guide# Create a virtual

environment

python -m venv edai\_env

source edai\_env/bin/activate # Linux/Mac

edai\_envScriptsactivate # Windows



# Install dependencies

pip install torch

pip install transformers

pip install gradio6. Code Structure OverviewModel and   
Tokenizer Loading: Defines which LLM is being used.Utility   
Functions: Handles prompt construction and response   
generation.Concept Explanation Module: Returns AI-  
generated explanation of topics.Quiz Generator Module:   
Returns 5 quiz questions + answers.Gradio UI Setup: Creates   
tabs with input fields and interactive buttons.7. Detailed Code   
WalkthroughModel Setupmodel\_name = "ibm-granite/granite-

3.2-2b-instruct"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(

model\_name,

torch\_dtype=torch.float16 if torch.cuda.is\_available() else

torch.float32,

device\_map="auto" if torch.cuda.is\_available() else None

)Loads IBM Granite model for text generation.Chooses float16

for efficiency when GPU is available.Automatically assigns   
device placement if CUDA is available.Tokenizer Setupif   
tokenizer.pad\_token is None:

tokenizer.pad\_token = tokenizer.eos\_tokenEnsures   
tokenizer has a valid pad\_token.Required because Hugging   
Face models sometimes lack padding tokens.Utility Function:

Response Generatordef generate\_response(prompt,   
max\_length=512):

inputs = tokenizer(prompt, return\_tensors="pt",   
truncation=True, max\_length=512)

if torch.cuda.is\_available():

inputs = {k: v.to(model.device) for k, v in inputs.items()}

with torch.no\_grad():

outputs = model.generate(

\*\*inputs,

max\_length=max\_length,

temperature=0.7,

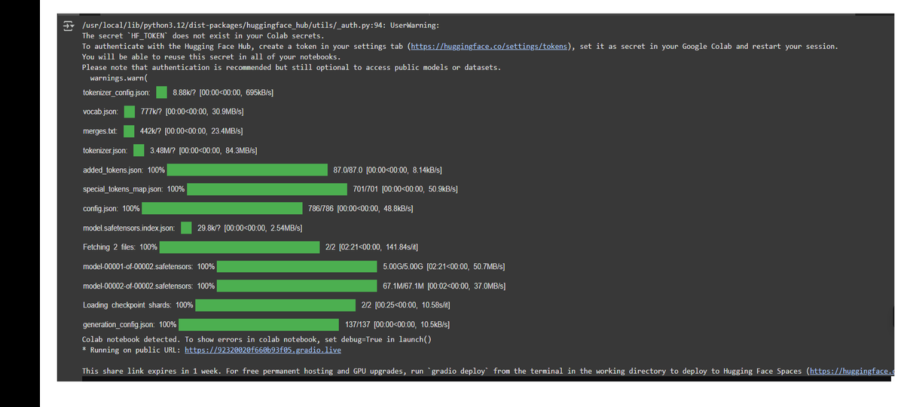
do\_sample=True,

pad\_token\_id=tokenizer.eos\_token\_id

)

response = tokenizer.decode(outputs[0],   
skip\_special\_tokens=True)

response = response.replace(prompt, "").strip()



return responseHandles prompt creation and processing

through the LLM.temperature=0.7 ensures balanced

creativity.Strips prompt from final response.Concept   
Explanationdef concept\_explanation(concept):

prompt = f"Explain the concept of {concept} in detail with

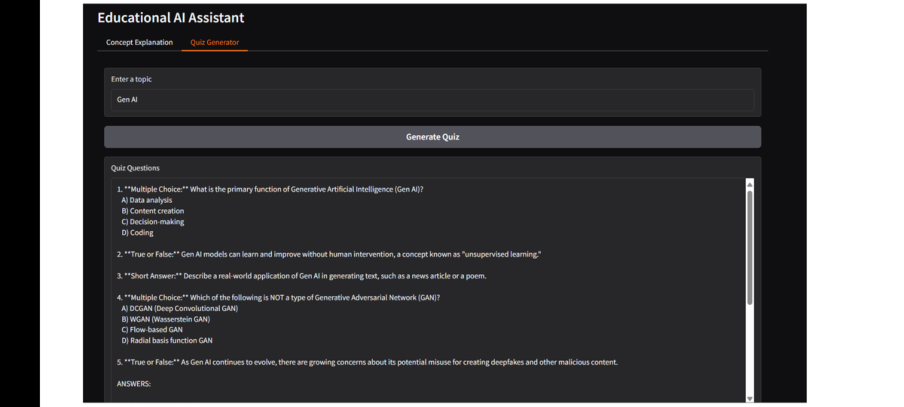
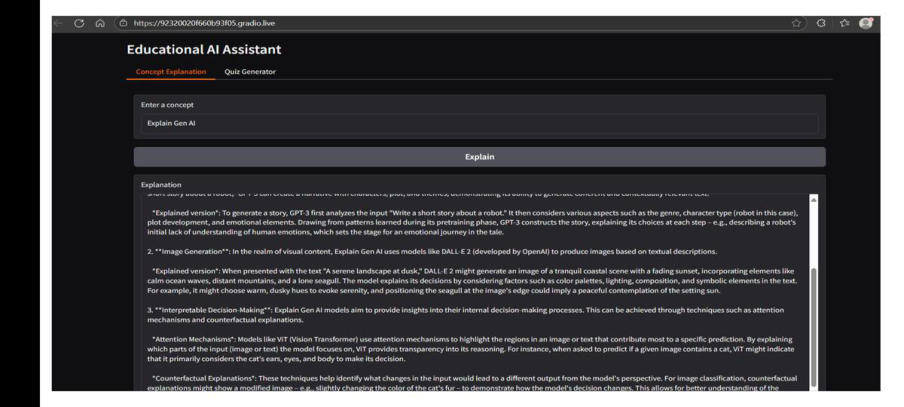
examples:"

return generate\_response(prompt, max\_length=800)Quiz   
Generatordef quiz\_generator(concept):

prompt = f"Generate 5 quiz questions about {concept} with   
different question types (multiple choice, true/false, short   
answer). At the end, provide all the answers in a separate

ANSWERS section."

return generate\_response(prompt, max\_length=1000)Gradio   
UIwith gr.Blocks() as app:



gr.Markdown("# Educational AI Assistant")

with gr.Tabs():   
 with gr.TabItem("Concept Explanation"):   
 concept\_input = gr.Textbox(label="Enter a concept",   
placeholder="e.g., machine learning")

explain\_btn = gr.Button("Explain")

explanation\_output = gr.Textbox(label="Explanation",

lines=10)

explain\_btn.click(concept\_explanation,   
inputs=concept\_input, outputs=explanation\_output)

with gr.TabItem("Quiz Generator"):

quiz\_input = gr.Textbox(label="Enter a topic",   
placeholder="e.g., physics")

quiz\_btn = gr.Button("Generate Quiz")

quiz\_output = gr.Textbox(label="Quiz Questions",

lines=15)

quiz\_btn.click(quiz\_generator, inputs=quiz\_input,   
outputs=quiz\_output)

app.launch(share=True)Simple two-tab layout.One for

explanations, one for quizzes.The share=True parameter   
generates a public Gradio link.8. Application WorkflowUser   
enters a concept/topic.Backend sends request to model with   
custom prompt.Model generates a textual response.Response   
displayed in Gradio interface.9. User Interaction FlowTab 1

(Concept Explanation): Input concept →Click "Explain" →  
Read detailed answer.Tab 2 (Quiz Generator): Input topic →

Click "Generate Quiz" →Get 5 questions + answers.10. Error   
Handling and Edge CasesIf user inputs an empty string →  
Returns minimal or blank response.If GPU not available →  
Runs smoothly on CPU with slower speed.If concept is too   
broad →Model may generate generic results.11. Performance

ConsiderationsGPU usage boosts performance.Batch   
processing could improve scalability.Current setup is user-  
friendly but not optimized for multi-user loads.12. Deployment   
GuideCan be hosted using Gradio share link.For production:   
Deploy on Hugging Face Spaces or Streamlit Cloud.For   
enterprise: Containerize with Docker and serve behind API.13.

Possible EnhancementsAdd speech-to-text input for   
learners.Include PDF export feature for quizzes.Store user   
quiz history in a backend database.Multi-language explanation   
support.14. Security ConsiderationsSince prompts are user-  
inputted, guard against malicious usage.Ensure prompt   
injection attacks are minimized.When deployed publicly, add   
query filters.15. Testing StrategyUnit Tests: For   
generate\_response, concept\_explanation,   
quiz\_generator.Integration Tests: Verify end-to-end model   
responses in the UI.User Acceptance Testing: Gather feedback   
from educators and students.16. Example Usage   
Screenshots(Placeholder – include screenshots of Gradio UI

once running.)17. Future ScopeAdaptive learning path

generation.Integration with classroom dashboards.Expand   
question variety with difficulty levels.18. ConclusionThis

Educational AI Assistant demonstrates how LLMs can

enhance learning experiences. With concept explanations and   
quiz generation, it supports both teaching and self-study.

While minimalistic in design, it offers strong potential for   
extension to a full-fledged learning system.Would you like me

to expand this into a polished PDF report (with headers,   
images, and proper formatting) so you can directly use it as a

15–20 page document?

BenefitTailored MaterialsReal-Time FeedbackProgress   
TrackingFlexible ExportCustomizationCollaborationEdutorAI   
ExampleQuizzes/flashcards from student input

• Auto grading, solution hints  
• Analytics dashboard  
• Save/share as PDF, PPT, CSV  
• Edit questions/edit flashcards  
• Group study and sharing  
• How It WorksAI processes student work for relevanceAI   
 assesses and explains answers instantlyMonitors   
 performance and adapts contentAny device, any format for   
 personalized reviewUser fine-tunes resources for   
 preferencesBuild, review, and learn   
 interactivelyBenefitEdutorAI ExampleHow It   
 WorksTailored MaterialsQuizzes/flashcards from student   
 input  
• AI processes student work for relevanceReal-Time   
 FeedbackAuto grading, solution hints  
• AI assesses and explains answers instantlyProgress   
 TrackingAnalytics dashboard  
• Monitors performance and adapts contentFlexible   
 ExportSave/share as PDF, PPT, CSV  
• Any device, any format for personalized   
 reviewCustomizationEdit questions/edit flashcards User   
 fine-tunes resources for preferencesCollaborationGroup   
 study and sharing  
• Build, review, and learn interactivelyEdutorAI enhances   
 personalized learning experiences by dynamically   
 generating study materials and assessments tailored to

each learner’s unique needs, preferences, and real-time   
 performance  
• Its AI-powered features help students, teachers, and   
 parents create, modify, and share customized quizzes,   
 flashcards, and worksheet content, ensuring learning is   
 more relevant and engaging  
• Key Ways EdutorAI Personalizes LearningAdaptive   
 Content GenerationEdutorAI analyzes student-supplied   
 text, images, or PDFs to create personalized learning   
 resources, such as quizzes and flashcards, focusing on topics   
 where learners need improvement or more practice  
• The platform uses performance analytics to assess progress   
 and tailors subsequent material based on strengths and   
 gaps, allowing for individualized practice and review  
• Real-Time Feedback and Progress TrackingAfter students   
 interact with quizzes or submit answers, EdutorAI offers   
 instant evaluations and actionable feedback, making   
 learning adaptive and iterative  
• Students and educators can monitor advancement, identify   
 difficult concepts, and adjust their study strategies   
 accordingly

• Flexible Modes of DeliveryLearning resources can be   
 exported in multiple formats—PDFs, text, CSV, PPT—for   
 convenient study, review, and collaboration  
• The system supports both self-paced learning and classroom   
 use, adapting to individual and group learning contexts  
• Customizable and Editable ResourcesUsers can modify the   
 AI-generated questions, quizzes, and flashcards, refining   
 them to match specific learning goals, levels, or interests

• Teachers can adapt content for differentiated instruction,   
 while students can create material suited to their learning

pace

• Collaboration and Peer EngagementEdutorAI’s sharing   
 features promote collaborative learning by allowing   
 classmates, teachers, and parents to exchange materials,   
 feedback, and study progress  
• This interaction helps learners benefit from tailored group   
 activities and peer-to-peer support  
• Benefits TableBenefitEdutorAI ExampleHow It   
 WorksTailored MaterialsQuizzes/flashcards from student   
 input  
• AI processes student work for relevanceReal-Time   
 FeedbackAuto grading, solution hints  
• AI assesses and explains answers instantlyProgress   
 TrackingAnalytics dashboard  
• Monitors performance and adapts contentAny device, any   
 format for personalized reviewFlexible ExportSave/share as   
 PDF, PPT, CSV  
• CustomizationEdit questions/edit flashcards

• User fine-tunes resources for

preferencesCollaborationGroup study and sharing  
• Build, review, and learn interactivelyEdutorAI’s adaptive,   
 interactive tools ensure that every learner receives the right   
 material at the right time, maximizing engagement,   
 retention, and success

What is Editor AI?

Editor AI is a tool that uses artificial intelligence to help users   
write, edit, and improve text (like documents, emails, or   
articles) automatically or with smart suggestions.

---

🧩Main Components

**1. User Interface (UI)**

Text editor where users type or paste content

Buttons for grammar check, rewrite, summarize, etc.

**2. Input Processing**

Converts user text into a format the AI can understand

Detects language, tone, and writing style

**3. AI Engine (Core Brain)**

Uses Natural Language Processing (NLP)

Performs tasks like:

Grammar & spell check

Rewriting sentences

Summarizing text   
Improving clarity or tone

**4. Knowledge Base (Optional)**

Domain-specific rules or style guides

Custom vocabulary (e.g., brand terms)

**5. Output Generator**

Returns edited or improved text   
May offer suggestions with explanations

**6. Feedback Loop (Learning)**

Tracks user choices (accepted or rejected edits)   
Improves AI suggestions over time (if learning is enabled)

🔄How It Works – Step by Step

1. ✍User types or pastes text

2. 🧾System analyzes text (grammar, tone, structure)

3. 🤖AI processes and suggests improvements

4. ✅User accepts, edits, or ignores suggestions

5. 🔁(Optional) AI learns from feedback to improve

future results

🔑Extra Key Features of EduTutorAI

24/7 Virtual Tutor

Acts like a personal AI tutor available anytime.

Students can ask doubts instantly without waiting for

teachers.

**Gamified Learning**

Adds badges, leaderboards, and rewards to keep students

motivated.

Makes learning fun and engaging like a game.

**Adaptive Assessments**

Difficulty of questions changes based on student performance.

Provides customized question banks for weak areas.

**Content Generation for Teachers**

Teachers can use EduTutorAI to generate lesson plans,   
worksheets, and quizzes in minutes.

Saves time and effort in preparing study material.

**Data-Driven Insights**

Generates performance analytics and progress reports.

Helps teachers and parents monitor learning outcomes.

**Multilingual Support**

Supports regional languages (Tamil, Hindi, Telugu, etc.) along

with English.

Encourages inclusive education for rural and urban learners.

**AI-Powered Revision Tools**

Creates flashcards, mind maps, summaries automatically.   
Suggests revision tests before exams.

**Seamless Integration**   
Can connect with Google Classroom, MS Teams, and other   
LMS platforms.

Compatible with mobile, tablet, and PC devices.

**Offline Learning Support**

Provides downloadable notes, PDFs, and recorded lectures.   
Helps students with low internet access.

**⚙Dependencies and Requirements of EduTutorAI**

**1. Hardware Requirements**

Minimum

Processor: Dual Core CPU (Intel i3 / AMD equivalent)

RAM: 4 GB

Storage: 20 GB free space   
Internet: 2–5 Mbps connection

Recommended

Processor: Quad Core CPU (Intel i5 / AMD Ryzen 5 or above)

RAM: 8–16 GB (for smooth AI model execution)

Storage: 50 GB SSD (for faster access to AI datasets & models)   
GPU: NVIDIA T4 / RTX (for running Generative AI models in   
Colab or locally)

Internet: Stable 10+ Mbps broadband

**2. Software Requirements**   
Operating System   
Windows 10/11, Linux (Ubuntu 20.04+), or macOS

Programming Languages

Python 3.9+ (primary language)   
JavaScript (for frontend integration if needed)   
Libraries / Frameworks (Python)   
transformers (Hugging Face – AI models)   
torch (PyTorch – AI/ML framework)   
gradio (for interactive AI apps)   
scikit-learn (machine learning utilities)   
numpy, pandas (data handling)   
matplotlib / seaborn (data visualization)

Web / App Tools

Flask / FastAPI (for backend services)

HTML, CSS, JavaScript (for UI/UX)   
Tailwind / Bootstrap (optional styling)

**3. AI Dependencies**   
Model Requirements   
Pre-trained LLMs like IBM Granite 3.2, LLaMA, GPT, or

Falcon

Fine-tuned models for question answering, summarization,

and translation

Cloud / Runtime Environment

Google Colab (GPU T4 runtime for training & testing)

Optionally: AWS, Azure, or IBM Cloud for scalability

Datasets Needed

Educational datasets (NCERT, CBSE, AI-generated content)

Custom training sets for Tamil + English bilingual education

4. Platform Dependencies   
LMS Integration   
Moodle, Google Classroom, or custom LMS support

Database

MySQL / PostgreSQL (for storing user data, scores, progress)

Version Control

Git / GitHub (for project collaboration)

5. User Requirements

Students need only:

A mobile/PC with internet

A registered account (EduTutorAI portal / app login)

Teachers/Admins need:

Access to dashboard for monitoring progress & uploading

resources

✅In short:

Hardware →Laptop/PC with GPU preferred

Software →Python + AI libraries + LMS integration tools

AI →Hugging Face / IBM Granite models, Colab runtime

Platform →LMS + Database + Cloud hosting

**⚙Installation of EduTutorAI**

1. Prerequisites

Before installation, ensure the following are ready:   
A system with Python 3.9+ installed

Stable internet connection

Access to Google Colab / VS Code / Jupyter Notebook

GitHub account (for cloning repositories)

2. Installation Steps

🔹Step 1: Clone / Download the Project   
Open Google Colab or your terminal.

Run the following command:   
git clone https://github.com/your-repo/edututorai.git

cd edututorai

🔹Step 2: Install Python Dependencies   
Install required Python libraries using pip:   
pip install torch torchvision torchaudio   
pip install transformers   
pip install gradio   
pip install pandas numpy matplotlib scikit-learn

(You can also create a requirements.txt and install everything

at once)   
pip install -r requirements.txt

🔹Step 3: Setup Runtime (Google Colab / Local)

If using Google Colab:

Go to Runtime →Change Runtime Type →Select GPU (T4

preferred)

If using Local system:

Ensure your GPU drivers + CUDA Toolkit are installed (for

NVIDIA GPUs).

🔹Step 4: Configure Model

Load a pre-trained AI model (example: IBM Granite or   
Hugging Face model):

from transformers import AutoTokenizer,

AutoModelForCausalLM

import torch

model\_name = "ibm-granite/granite-3.2-2b-instruct" #

Example

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(

model\_name,

torch\_dtype=torch.float16 if torch.cuda.is\_available() else

torch.float32,

device\_map="auto"

)

🔹Step 5: Run EduTutorAI Interface

Launch the Gradio app for interactive use:   
import gradio as gr

def tutor\_ai(input\_text):   
 inputs = tokenizer(input\_text,   
return\_tensors="pt").to(model.device)   
 outputs = model.generate(\*\*inputs, max\_length=200)   
 return tokenizer.decode(outputs[0],   
skip\_special\_tokens=True)

gr.Interface(fn=tutor\_ai, inputs="text", outputs="text",

title="EduTutorAI").launch()

👉This will open a web-based interface where users can chat

with EduTutorAI.

🔹Step 6: Database & LMS Integration (Optional)

Setup MySQL/PostgreSQL database for user progress   
tracking.

Connect EduTutorAI with Moodle / Google Classroom API if   
required.

3. Verification of Installation

Run a sample query:

Input: "Explain Merge Sort in simple terms"   
Output: AI-generated explanation   
If the response is correct, EduTutorAI is successfully installed

🎉

✅With these steps, EduTutorAI can be installed and run

either on Colab (easiest) or local system with GPU.

Conclusion

EdutorAIdemonstrateshowGenerativeAIcantransformeducationby   
providingapersonalized,interactive,andscalablelearningexperience.By

integratingwithLearningManagementSystems(LMS)andleveraging

advancedAImodels,theplatformadaptstotheuniqueneeds,pace,and   
learningstyleofeachstudent.

Throughfeaturessuchaspersonalizedprompts,automatedassistance,   
multilingualsupport,andreal-timeguidance,EdutorAIbridgesthegap   
betweentraditionalclassroomteachingandmodernAI-powereddigital   
learning.Itnotonlyenhancesacademicoutcomesbutalsobuildsstudents’  
criticalthinking,creativity,andproblem-solvingskills.

Forteachersandinstitutions,EdutorAIreducesworkload,improvesstudent   
engagement,andenablesdata-drivendecision-makingtotrackprogress   
effectively.Withitsuser-friendlydesignandrobustAIintegration,theproject   
holdspotentialtobecomeascalableEdTechsolutionthatempowersboth

learnersandeducators.

Inessence,EdutorAIisnotjustatoolbutasteptowardthefutureofsmart,   
inclusive,andadaptiveeducation,aligningwithglobaltrendsinAI-driven   
personalizedlearning.