EDU TUTOR AI : PERSONALIZED LEARNING WITH GENERATIVE AI AND LMS INTEGRATION

PROJECT DOCUMENT

1.INTRODUCTION

PROJECT TITLE: EDU TUTOR AI: PERSONALIZED LEARNING WITH GENERATIVE AI AND LMS INTEGRATION

TEAM MEMBERS: MANASHA. R

TEAM MEMBERS:DHANASRI. D

TEAM MEMBERS:HARINI. P

TEAM MEMBERS:HARINI. S

2. PROJECT OVERVIEW

=>Purpose:

An AI personalized learning project uses artificial intelligence to create custom educational paths and content for individual students by analyzing their data, preferences, and progress to improve comprehension and engagement. The project involves developing AI algorithms and adaptive platforms that identify learning styles and adjust content and pace accordingly, providing a more effective and tailored educational experience than traditional methods.

=>Features:

•Key features of Edu Tutor AI:

~^ Adaptive Learning:

AI analyzes a student's performance, adjusting the difficulty and sequence of learning materials to match their individual pace and skill level.

~^ Personalized Content & Recommendations:

The system provides tailored exercises, readings, and practice tests based on the student's progress and interests.

~^Intelligent Tutoring & Support:

AI-powered chatbots and virtual assistants offer 24/7 instant support, answering questions and providing explanations as needed.

~^ Real-Time Feedback & Progress Tracking:

AI monitors student performance and delivers immediate, targeted feedback, helping learners identify and correct mistakes quickly.

~^ Data-Driven Insights:

By collecting and analyzing student data, AI systems provide valuable analytics for educators, helping them understand individual strengths and weaknesses and improve learning outcomes.

~^ Engagement & Accessibility Features

Gamification:

AI integrates game-like elements such as leaderboards and rewards to make learning more engaging and motivating.

~^ Multimodal Support:

Platforms can offer content in various formats, including text, audio, and video, catering to diverse learning preferences.

~^ Enhanced Accessibility:

AI can provide customized support for learners with disabilities or unique needs, promoting inclusivity.

~^ Flexible Pacing:

Students can learn at their own speed, with AI adjusting the flow of the curriculum to prevent them from being overwhelmed or rushed.

~^Underlying Technologies Machine Learning Algorithms:

These algorithms power the AI's ability to process learner data and create personalized experiences.

~^Natural Language Processing (NLP):

NLP enables AI systems to understand and respond to student questions in a human-like manner.

~^Predictive Analytics:

AI uses historical data to predict potential performance issues and suggest proactive measures to avoid them.

3.ARCHITECTURE

=>Frontend(Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

=>Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy

Swagger integration.

=>LLM Integration (IBM Watsonx Granite):

Granite LLM models from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate summaries, sustainability tips, and reports.

Vector Search (Pinecone):

Uploaded policy documents are embedded using Sentence Transformers and stored in Pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

=>ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models are used for forecasting and anomaly detection using Scikit-learn. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

4. SETUP INSTRUCTIONS

•Prerequisites:

- o Python 3.9 or later
- o pip and virtual environment tools
- o API keys for IBM Watsonx and Pinecone
- o Internet access to access cloud services

•Installation Process:

- o Clone the repository
- o Install dependencies from requirements.txt
- o Create a .env file and configure credentials
- o Run the backend server using Fast API
- o Launch the frontend via Stream lit
- o Upload data and interact with the modules

5.FOLDER STRUCTURE

The folder structure for Edu Tutor AI personalized learning typically includes:

- o Data: Student information, learning materials, and assessment data.
- o Src: Source code for AI models, learning algorithms, and user interface.
- o Config: Configuration files for settings and parameters.
- o Tests: Automated tests to ensure functionality and performance.
- o Docs: Documentation for developers, administrators, and user.

6. RUNNING THE APPLICATION

To start the project:
➢ Launch the FastAPI server to expose backend endpoints.
➢ Run the Streamlit dashboard to access the web interface.
➢ Navigate through pages via the sidebar.
➢ Upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.
➢ All interactions are real-time and use backend APIs to dynamically update the frontend.

1. Access the Platform:
Navigate to the AI tutor's platform, which is likely a website or a mobile application, from your device.
2. Engage with Content:
Start interacting with the provided educational content and exercises.
3. Allow Adaptation:
Observe how the application adapts to your responses. It will present new materials or modify pacing as it learns about your performance.
4. Utilize Feedback:
Pay attention to the real-time feedback and guidance the AI provides to improve your understanding.

°Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

°Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

7. API DOCUMENTATION

API documentation for Edu Tutor AI personalized learning:

Student Management
- GET /students: Retrieve student profiles
- POST /students: Create new student profile

- PUT /students/{id}: Update student profile
- DELETE /students/{id}: Delete student profile

Learning Plans
- GET /learning-plans: Retrieve learning plans for a student
- POST /learning-plans: Create new learning plan
- PUT /learning-plans/{id}`: Update learning plan
- DELETE /learning-plans/{id}: Delete learning plan

Assessment and Feedback
- POST /assessments: Submit assessment results
- GET /feedback: Retrieve feedback for a student

Content Management
- GET /content: Retrieve learning materials
- POST /content: Upload new learning material
- PUT /content/{id}: Update learning material
- DELETE /content/{id}: Delete learning machine.

•API Documentation Tools

- Swagger
- API Blueprint
- Dox

This outline provides a starting point for documenting the API endpoints, request/response formats, and authentication mechanisms.

8. AUTHENTICATION

•Token-based authentication (JWT or API keys)
•Role-based access (admin, , researcher)
•Username-Password Authentication: Students and educators can log in using their username and password.
•OAuth: Integration with popular platforms (e.g., Google, Facebook) for seamless authentication.
•API Keys: Secure access for developers and integrations.
•Two-Factor Authentication (2FA): Additional security layer via SMS, email, or authenticator apps.
•Single Sign-On (SSO): Centralized authentication for multiple applications.

9.USER INTERFACE

In Edu Tutor AI personalized learning, the user interface (UI) typically includes:

1. Personalized Dashboard
2. Adaptive Learning Paths
3. Interactive Lessons
4. Real-time Feedback
5. Progress Tracking
6. Resource Recommendations
7. User-friendly Navigation

The design prioritizes clarity, speed, and user guidance with help texts.

10.TESTING

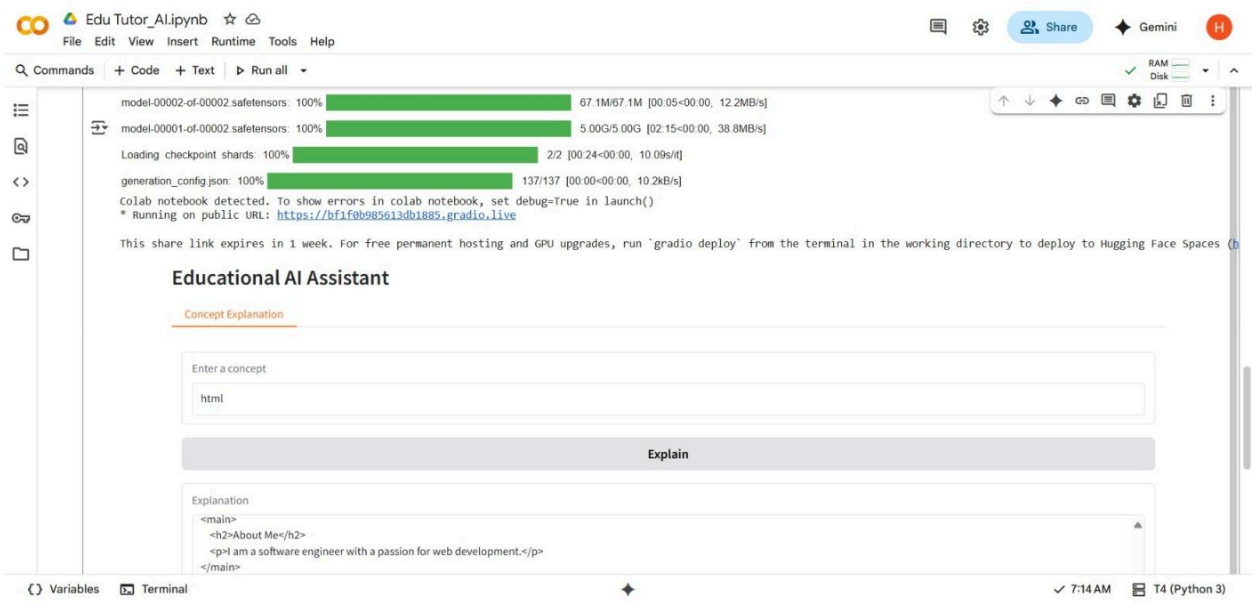
Testing multiple phases of Edu Tutor AI involves:

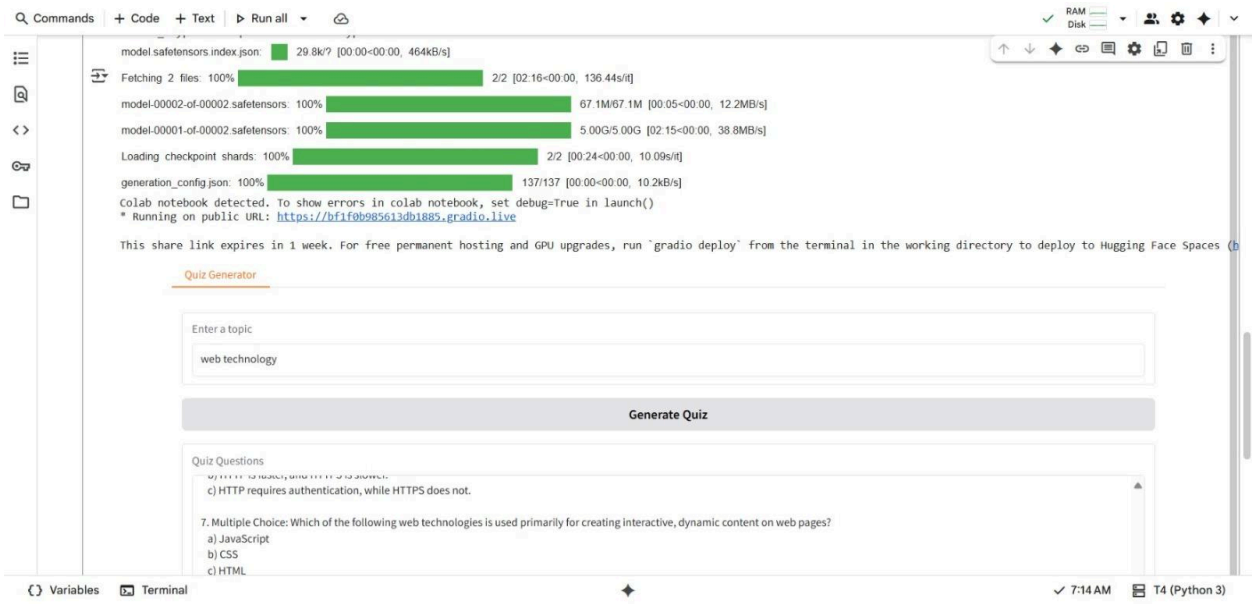
1. Unit Testing: Verifying individual components (e.g., AI models, algorithms) function correctly.
2. Integration Testing: Ensuring seamless interaction between components and systems.
3. Functional Testing: Validating the platform's features and functionality meet requirements.
4. User Acceptance Testing (UAT): Confirming the platform meets user expectations and needs.
5. Performance Testing: Evaluating the platform's scalability, speed, and responsiveness.
6. Security Testing: Identifying vulnerabilities and ensuring data protection.

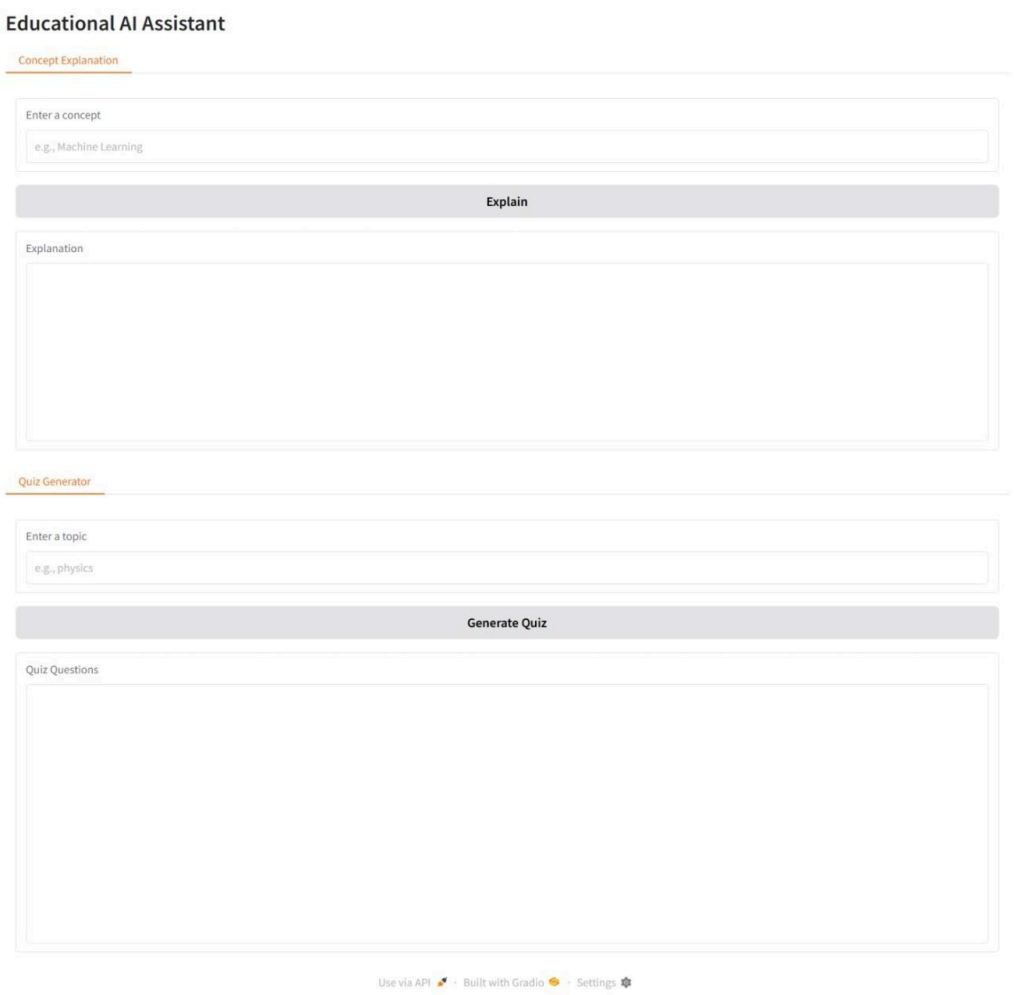
These testing phases help ensure Edu Tutor AI's reliability, effectiveness, and security.

11.SCREEN SHOTS

=>Output:

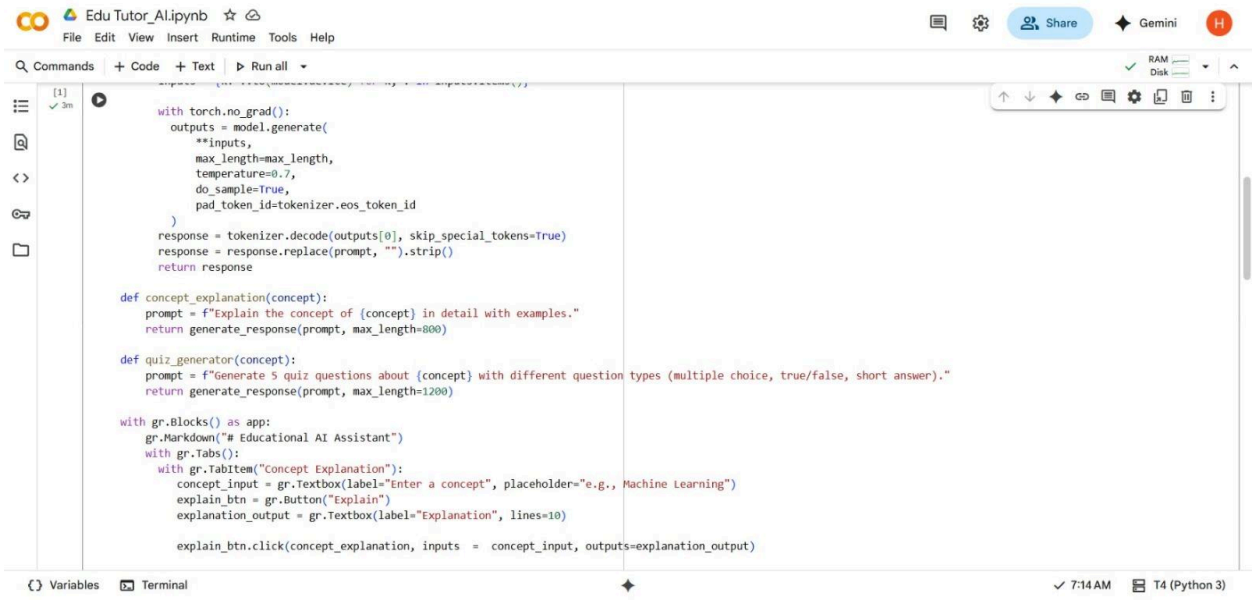


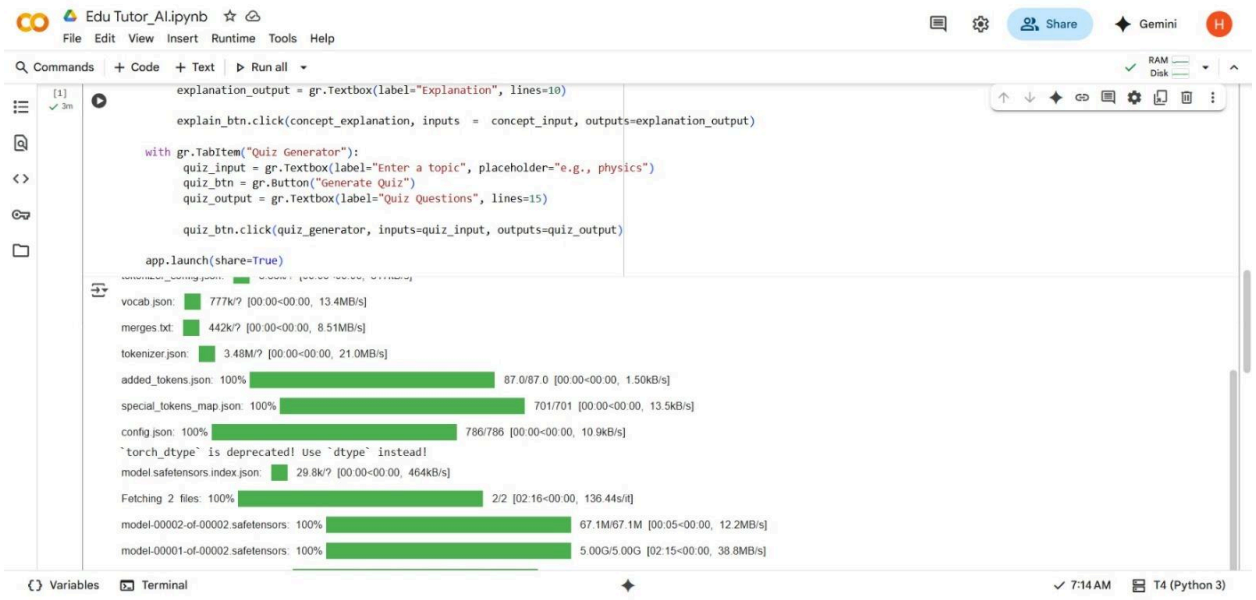

Commands
+ Code
+ Text
Run all
RAM
Disk
model.safetensors.index.json: 29.8k/? [00:00<00:00, 464kB/s]
Fetching 2 files: 100% 2/2 [02:16<00:00, 136.44s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:05<00:00, 12.2MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:15<00:00, 38.8MB/s]
Loading checkpoint shards: 100% 2/2 [00:24<00:00, 10.09s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 10.2kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://bf1f0b985613db1885.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (h
Quiz Generator
Enter a topic
web technology
Generate Quiz
Quiz Questions
c) HTTP requires authentication, while HTTPS does not.
7. Multiple Choice: Which of the following web technologies is used primarily for creating interactive, dynamic content on web pages?
a) JavaScript
b) CSS
c) HTML
Variables
Terminal
7:14 AM
T4 (Python 3)


Educational AI Assistant
Concept Explanation
Enter a concept
e.g., Machine Learning
Explain
Explanation
Quiz Generator
Enter a topic
e.g., physics
Generate Quiz
Quiz Questions
Use via API
Built with Gradio
Settings

=>Coding:


Edu Tutor_AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[1]
3m
Eductional AI Application using IBM Granite Model
Run this in Google Colab
!pip install transformers torch gradio -q
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
model_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
)
if tokenizer.pad_token is None:
tokenizer.pad_token = tokenizer.eos_token
def 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
Variables
Terminal
7:14 AM
T4 (Python 3)


Edu Tutor_AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM
Disk
[1]
3m
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 response
def concept_explanation(concept):
prompt = f"Explain the concept of {concept} in detail with examples."
return generate_response(prompt, max_length=800)
def quiz_generator(concept):
prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple choice, true/false, short answer)."
return generate_response(prompt, max_length=1200)
with 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)
Variables
Terminal
7:14 AM
T4 (Python 3)



12.KNOWN ISSUES

Key issues in AI personalized learning include data privacy and security concerns, potential for algorithmic bias and misinformation in AI outputs, high implementation costs and infrastructure needs, the lack of human connection and empathy compared to human teachers, the need for adequate teacher training, and the risk of students over-relying on AI rather than developing critical thinking skills.

13.FUTURE ENHANCEMENT

In Edu Tutor AI, "future enhancement" refers to planned improvements or upgrades that will be made to the AI-powered personalized learning platform. These enhancements aim to:

1. Improve learning outcomes
2. Increase student engagement
3. Enhance teacher support
4. Expand AI-driven features
5. Refine user experience

"Future enhanced" refers to planned improvements or upgrades that will be made to a system, product, or technology in the future.