Edu Tutor AI: Personalized Learning Generative AI with IBM

**Project Documentation**

**1. Introduction**

**Project title:** Edu Tutor AI: Personalized Learning Generative AI with IBM

**Team members:**

* Team member 1: VIGNESH M
* Team member 2: AKBAR BAIG E
* Team member 3: SARAN P
* Team member 4: SUJAY B
* **2. Project Overview**

**Purpose of EduTutor AI:**

**The purpose of EduTutor AI is to create a personalized learning platform that leverages Generative AI and IBM technologies to enhance the educational experience of students. The system aims to provide adaptive, interactive, and customized learning content based on each learner’s pace, strengths, and weaknesses. By integrating advanced AI models with IBM cloud services and AI tools, EduTutor AI seeks to:**

* **Deliver personalized study plans and recommendations.**

* **Generate adaptive quizzes, explanations, and practice exercises.**

* **Support multiple learning styles (visual, textual, interactive).**

* **Offer real-time feedback and performance tracking.**

* **Enable scalable AI-powered education solutions for institutions and individual learners.**

**Features:**

**⭐ Features of EduTutor AI**

**1. Personalized Learning Paths**

* **Adapts study plans to each learner’s pace, strengths, and weaknesses.**

* **Recommends topics based on previous performance.**

**2. AI-Generated Content**

* **Creates quizzes, flashcards, and practice questions dynamically.**

* **Provides simplified explanations and examples using Generative AI.**

**3. Multi-Modal Learning Support**

* **Text, audio, and visual content for different learning styles.**

* **Interactive lessons with step-by-step problem solving.**

**4. Real-Time Feedback & Analytics**

* **Tracks learner progress and gives instant performance insights.**

* **Offers detailed analytics dashboards for teachers and students.**

**5. Doubt Solving Assistant**

* **AI chatbot to answer student questions anytime.**

* **Provides hints, step-by-step reasoning, and alternate solutions.**

**6. Gamified Learning**

* **Badges, leaderboards, and achievements to boost engagement.**

* **Personalized challenges to make learning fun.**

**7. Seamless IBM Cloud Integration**

* **Scalable backend using IBM Watson, IBM Cloud AI services, or IBM AutoAI.**

* **Ensures secure, reliable, and enterprise-grade performanc**

**8. Collaboration & Sharing**

* **Students can collaborate in study groups.**

* **Teachers can assign AI-generated tasks and monitor performance.**

**3. Architecture**

**🏗 System Architecture of EduTutor AI**

**1. User Layer (Frontend)**

**Students**

**Access personalized lessons, quizzes, progress reports.**

**Interact with AI tutor chatbot for doubt solving.**

**Teachers**

**Monitor student performance analytics.**

**Assign AI-generated content or assessments.**

**---**

**2. Application Layer (Middleware / Backend)**

**Generative AI Engine**

**Creates adaptive quizzes, explanations, and summaries.**

**Uses Large Language Models (LLMs) to support natural conversation.**

**Backend (Python + Transformers):**

* Processes user inputs, generates AI responses using Granite LLM.
* Generates voice output using gTTS.
* Handles model loading and GPU optimization if available.

**LLM Integration (IBM Granite – Hugging Face Model):**

* Model: ibm-granite/granite-3.2-2b-instruct
* Performs natural language understanding and generation.
* Generates disease predictions and treatment plans.

**Audio Generation (gTTS):**

* Converts AI text responses into MP3 audio files.
* Plays directly in Gradio interface.

**Deployment:**

* The app can be deployed locally or shared publicly using app.launch(share=True).

**4. Setup Instructions**

**Prerequisites:**

* Python 3.9 or later
* pip for package installation
* Internet connection (for downloading model and TTS)
* GPU recommended for faster response (optional)

**Installation Process:**

1. Clone the repository.
2. Install dependencies:

pip install gradio torch transformers gtts

1. Run the Gradio app:

python app.py

1. Open the provided local URL or use the public share link.
2. Interact with the Disease Prediction and Treatment Plan modules.

**5. Folder Structure**

app.py -Main Gradio app and UI layout

model/ - Optional folder to store downloaded model weights

requirements.txt - Python dependencies

audio/ - Generated TTS audio files (doctor\_voice.mp3, treatment\_voice.mp3)

**6. API Documentation**

*(Note: This Gradio app runs as a local interface; no separate REST API is implemented. Optional future enhancement could add FastAPI backend.)*

* **Inputs:** Textboxes (Explain and Quiz)
* **Outputs:generate the explain the concepts and generate a quiz**

**7. Authentication:**

* **When a user (student/teacher) logs in, they receive a JWT token.**
* **This token must be included in the headers of every API request.**
* **EduTutor AI integrates IBM Watson services (Assistant, NLP, AutoAI).**

# Role-Based Access Control (RBAC)

* Student – Access quizzes, AI tutor, learning path, progress reports.
* Teacher – Create/assign content, view student analytics.
* Admin – Manage users, configure IBM Cloud services.

# Security Features

✔ Password Hashing (bcrypt/argon2 for secure storage). ✔ Token Expiry & Refresh Tokens (to avoid misuse). ✔ HTTPS Encryption (all API calls should use SSL/TLS). ✔ Role-Based Restrictions (teachers can’t impersonate students).

**9. User Interface:**

🎨 User Interface – EduTutor AI

* The EduTutor AI platform has two main roles: Students and Teachers.
* Below are the key UI modules for each:
* 1. Login & Registration

* Simple Sign Up / Sign In page.

* Role selection: Student / Teacher.
* Social login option (Google / IBM ID).

2. Student Dashboard

* Welcome Panel → Greeting + Current Learning Path.

* Progress Tracker → Visual bar or pie chart of completed topics.

* Upcoming Quizzes → AI-generated tests ready to attempt.
* AI Tutor Chatbot → Sidebar chatbot for instant doubt solving.
* Gamification Section → Badges, Leaderboards, Streaks-

3. Learning Module (Student View)

* Topic Page with:

* Video/Lesson Content
* AI-generated flashcards

* Example problems & hints

* Practice Mode → Interactive quizzes with instant feedback.
* Adaptive Mode → Difficulty adjusts automatically.

4. Quiz Interface

* Clean MCQ layout (question + options).
* Timer (optional).
* Hint Button (AI gives guidance without full answer).
* Result Screen with score, explanation, and suggested next topics.

5. Teacher Dashboard

* Class Overview → List of enrolled students.

* Analytics Panel → Graphs showing student progress & weak areas.
* Content Generator → Create or auto-generate quizzes/assignments.
* Student Reports → Downloadable PDF/CSV progress
* 6. Admin Panel (Optional)
* Manage users (add/remove).

* Configure IBM API keys.

* Monitor system usage (active students, quizzes taken).

7. Mobile App (Optional)

* Same functionality in mobile-friendly design.

* Push notifications for quiz reminders & progress updates.

📌 UI Flow (Student Example)

* Login → Dashboard → Select Topic → Study Content → Take Quiz → View Results → Track Progress

* ⚡ Do you want me to sketch a wireframe/mockup (boxes & layout) for this UI so you can use it in your documentation/presentation?Minimalist, tab-based interface with Gradio.
* Tabs: Disease Prediction, Treatment Plans
* Each tab contains:
  + Input fields (textbox, number, dropdown)
  + Action button (Analyze Symptoms / Generate Treatment Plan)
  + Output areas (text response + audio player)
* Clear disclaimer displayed at the top: **For informational purposes only. Always consult healthcare professionals.**

**10. Testing**

**Unit Testing:**

* Great 👍 Let’s add a Testing section for your EduTutor AI project documentation.

* ---
* 🧪 Testing – EduTutor AI
* Testing ensures that EduTutor AI runs reliably, securely, and provides accurate learning outcomes. The testing process includes Unit, Integration, Functional, Performance, and Security testing.

1. Unit Testing

Test individual modules:

User Authentication (JWT token generation/validation).

Quiz Generator (correct question/answer format).

Recommendation Engine (recommends based on history).

Chatbot (Watson Assistant) responses.

Tools:

pytest (Python)

* unittest

2. Integration Testing

Ensure smooth communication between backend, frontend, and IBM Cloud services.

Example tests:

API ↔ Database connection.

IBM Watson NLP ↔ AI Assistant responses.

User Dashboard ↔ Progress Analytics.

**Tools:**

* Postman (API testing)
* Pytest + Requests library

3. Functional Testing

Verify the application meets functional requirements.

Test cases:

User login & registration.

Quiz creation, attempt, and result calculation.

Student progress tracking.

Teacher assigning content.

Tools:

Selenium (UI automation).

Robot Framework.

4. Performance Testing

Test application speed and scalability.

Example checks:

Load testing → Can 500 students take a quiz simultaneously?

Response time of AI Tutor chatbot.

Tools:

Apache JMeter

Locust

5. Security Testing

Check for vulnerabilities like SQL injection, XSS, and token misuse.

Ensure encrypted storage of passwords (bcrypt).

Validate role-based access (RBAC) (student can’t access teacher dashboard).

Tools:

OWASP ZAP

IBM Security

6. User Acceptance Testing (UAT)

* **Final testing with real students & teachers.**

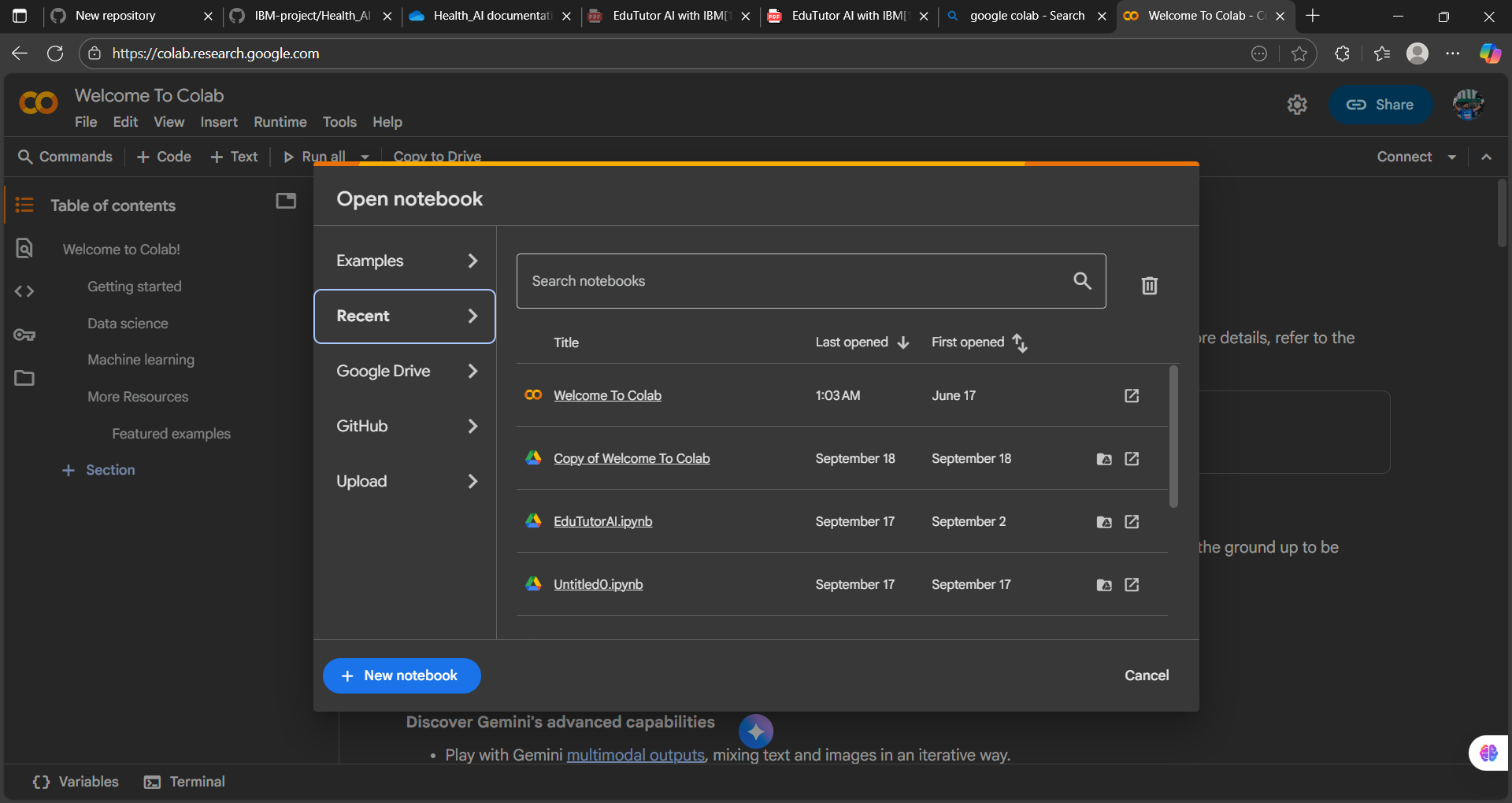
* **Collect feedback on:**
* **Ease of use.**

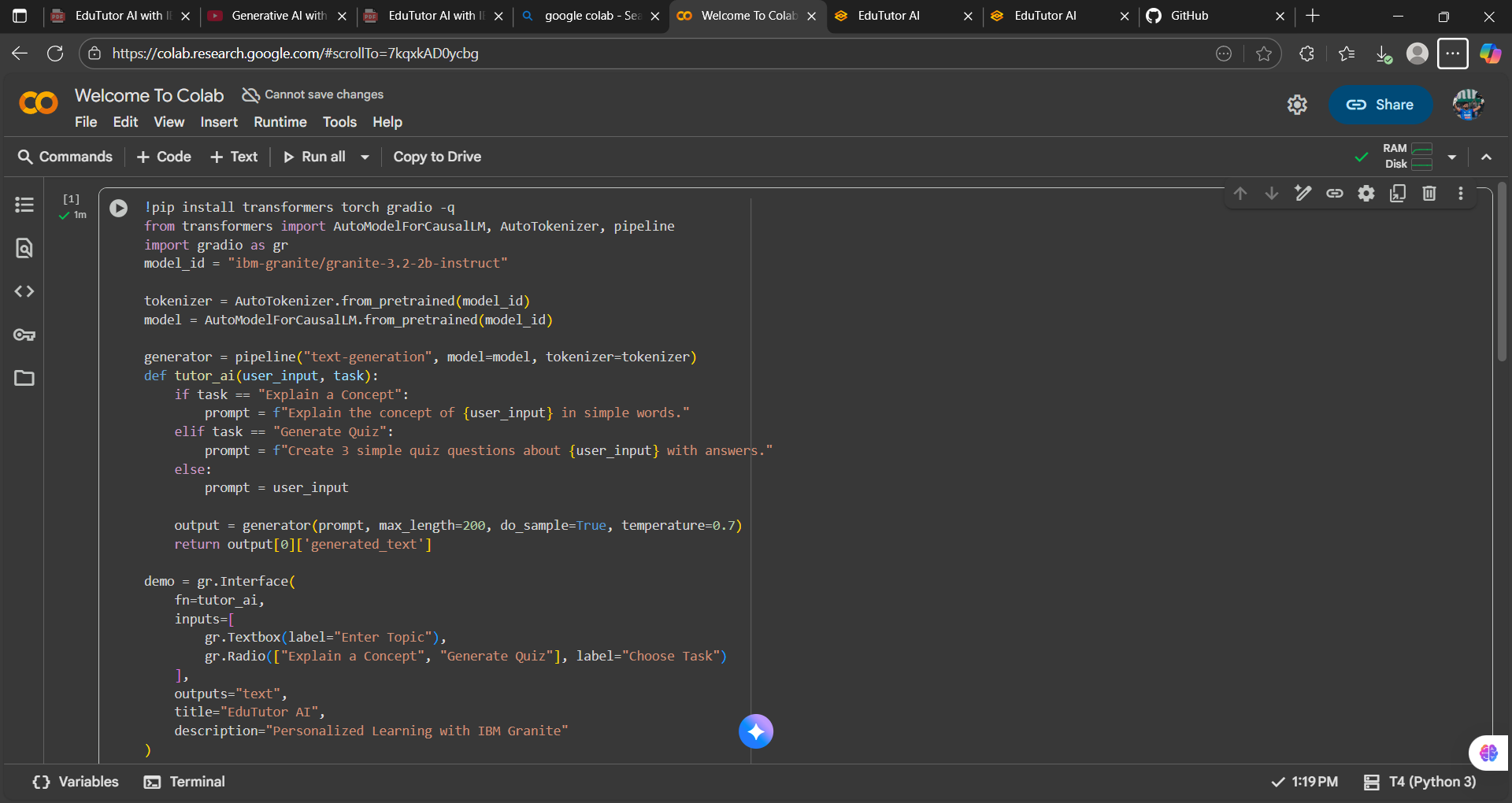
* **Accuracy of AI-generated quizzes**.

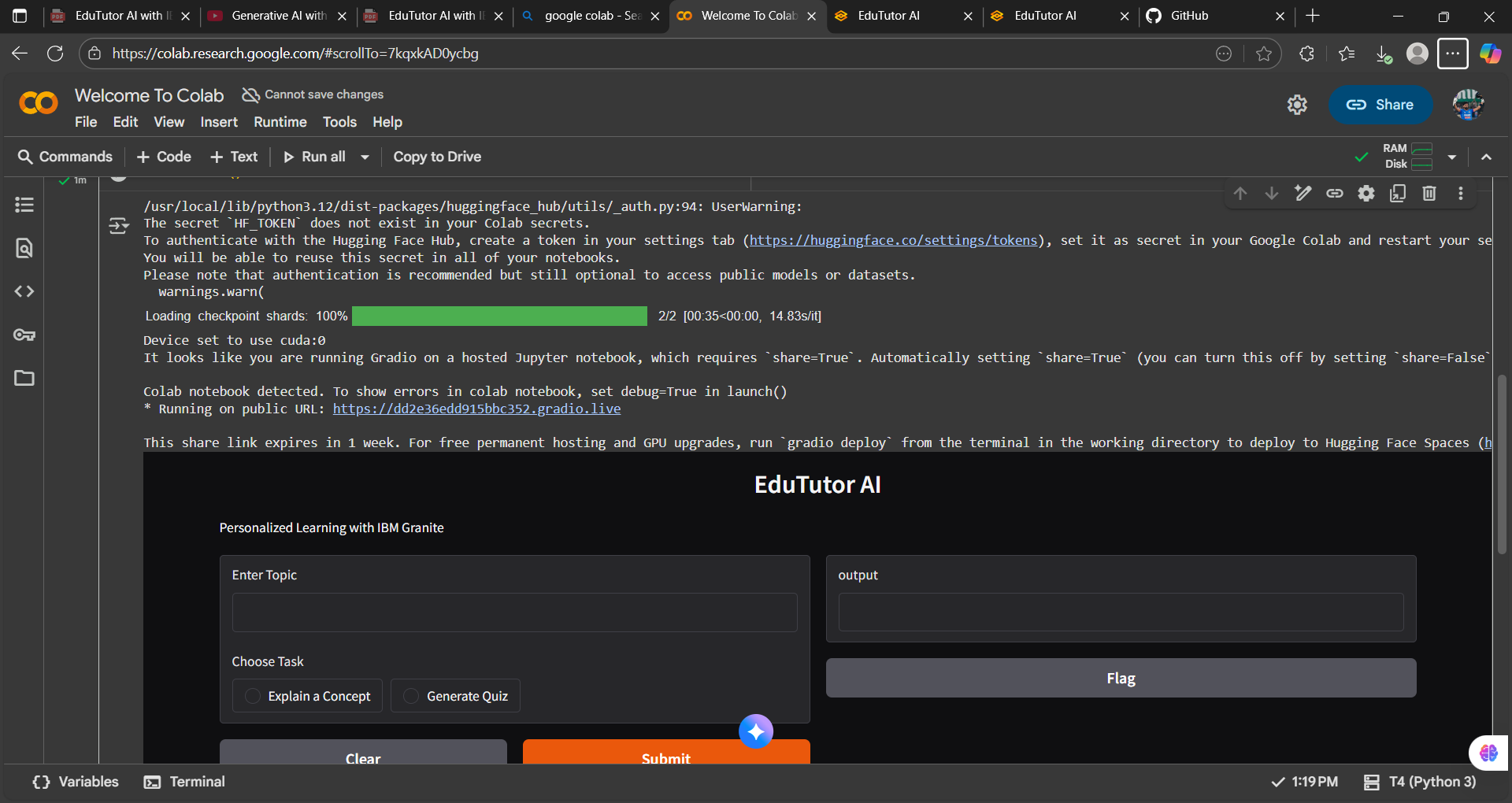
Unit Tests → Integration Tests → Functional Tests → Performance & Security Tests → UAT → Deployment

**11. Screenshots**

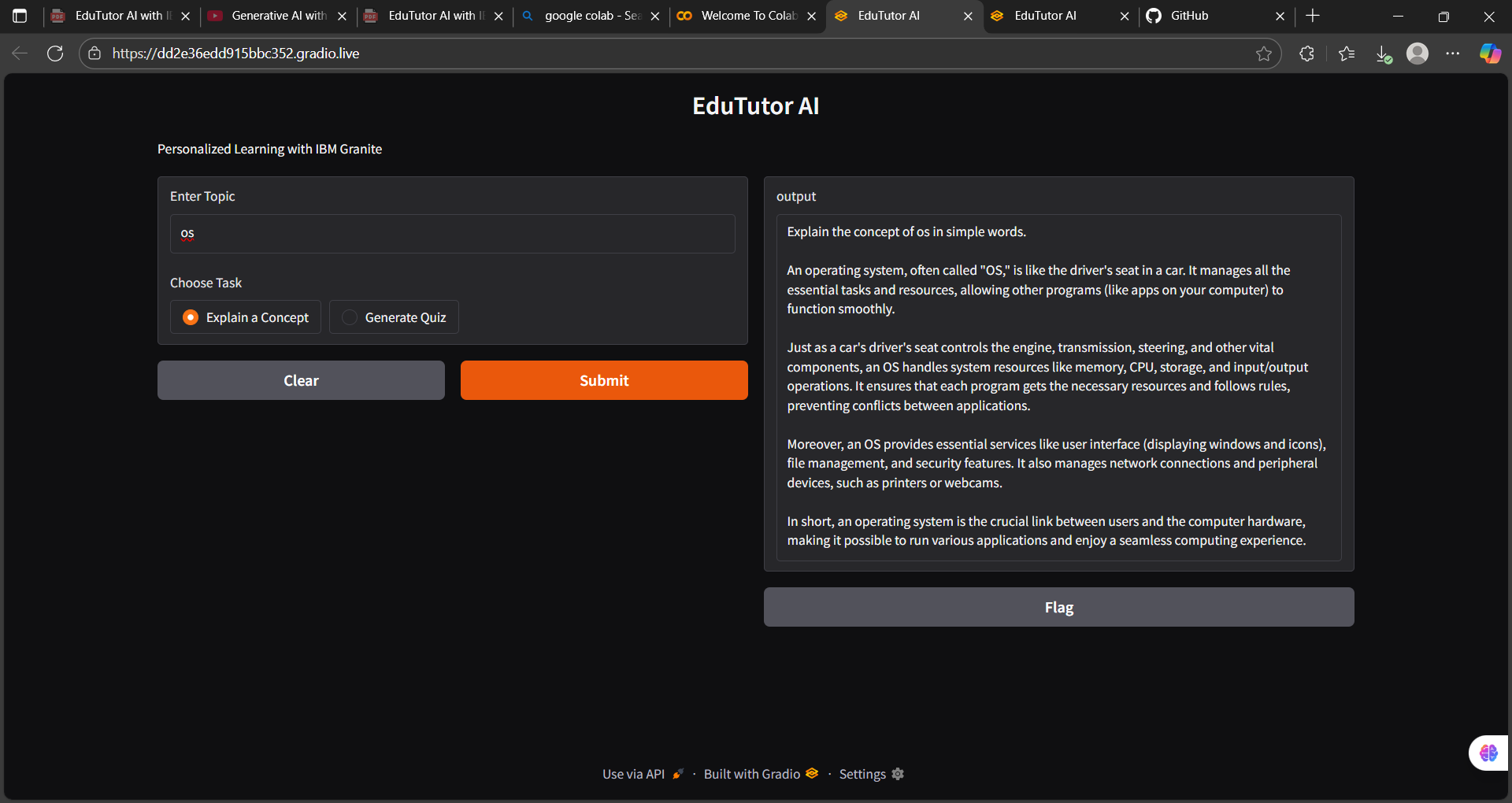
**TOOLS:**

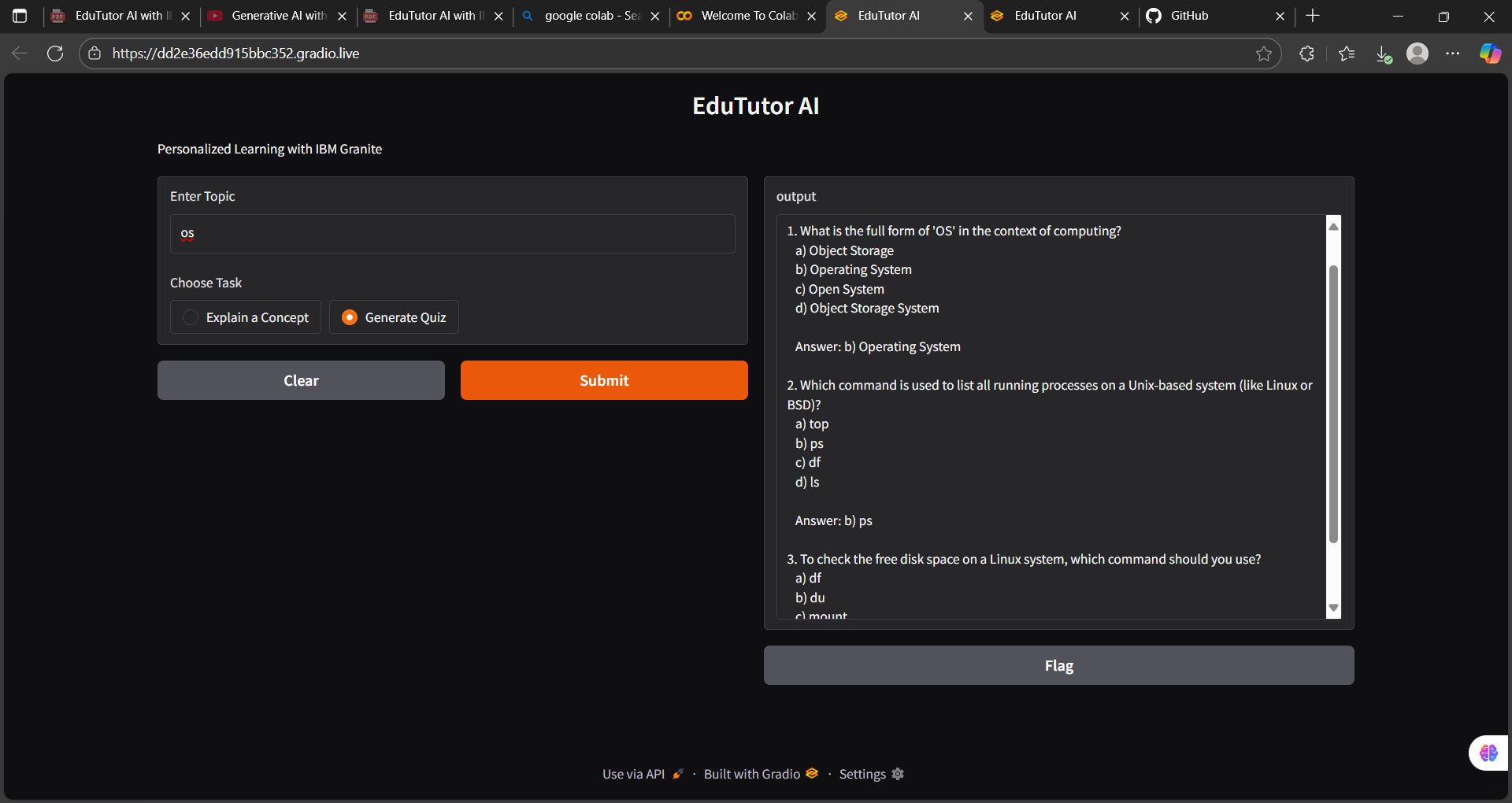


**CODING:**

**OUTPUT LINK:**

**FINAL OUTPUT:**





**GITHUB SCREENSHOT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**12. Known Issues**

* Large prompts may exceed model max token limit.
* Some complex medical queries may be inaccurate.
* Voice generation may fail if internet connection is unstable.
* Model requires significant memory; GPU recommended.

**13. Future Enhancements**

* Convert Gradio app into REST API with FastAPI backend.
* Add user authentication and history tracking.
* Integrate with cloud TTS services for better voice quality.
* Add additional health modules (e.g., medication reminders, symptom checker).
* Mobile-friendly web interface.