Edu Tutorn: Personalized Learning

# Project Documentation

# 1. Introduction

Project title: Edu Tutorn: Personalized Learning

Team members:

1. Sowmiya S
2. Nithya P
3. Priya Dharshini R

4. Safrin Saniya U

# 2. Project Overview

## Purpose:

Edu Tutorn is an AI-powered personalized learning assistant designed to make education more interactive and adaptive. It helps students understand concepts through natural language explanations and generates quizzes to reinforce learning. By leveraging IBM Granite LLM and a simple Gradio UI, the system provides accessible, on-demand tutoring support for learners of all levels.

## Features:

* Concept Explanation  
  Key Point: Interactive learning support  
  Functionality: Explains complex concepts in simple language with examples.
* Quiz Generator  
  Key Point: Reinforcement through practice  
  Functionality: Creates diverse quiz questions (MCQs, True/False, short answers) with an answer section.
* Conversational Interface  
  Key Point: Natural language interaction  
  Functionality: Learners input queries and receive instant responses in plain English.
* User-friendly Web App  
  Key Point: Accessibility  
  Functionality: Easy-to-use Gradio UI with tabs for explanation and quizzes.

# 3. Architecture

Frontend (Gradio UI): Built with Gradio Blocks and Tabs. Provides a minimal, intuitive interface where learners can switch between 'Concept Explanation' and 'Quiz Generator'.

Backend (Transformers + PyTorch): Uses Hugging Face transformers with AutoModelForCausalLM and AutoTokenizer for natural language generation.

LLM Integration (IBM Granite): Powered by IBM Granite 3.2 Instruct model for explanation and quiz generation tasks.

# 4. Setup Instructions

## Prerequisites:

* Python 3.9 or later
* pip for dependency management
* Internet access (to load IBM Granite model from Hugging Face)
* CUDA-enabled GPU (optional, improves performance)

## Installation:

1. Clone repository.
2. Install dependencies from requirements.txt.
3. Run python edututorai.py.
4. The Gradio app launches with a shareable link.

# 5. Folder Structure

project-root/  
│── edututorai.py # Main application script  
│── requirements.txt # Dependencies  
│── README.md # Documentation

# 6. Running the Application

Launch the app with: python edututorai.py

Open the Gradio link in a browser. Use Concept Explanation Tab or Quiz Generator Tab as required.

# 7. API Documentation

Currently only available through the Gradio UI. Future versions may expose FastAPI endpoints.

# 8. Authentication

Open app runs without authentication for demo purposes.

Future enhancements may include login, role-based access, and teacher/student accounts.

# 9. User Interface

Tabs: Concept Explanation | Quiz Generator

Textbox Inputs: Accepts user queries.

Textbox Outputs: Displays explanations or quiz content.

Buttons: Trigger responses from the AI.

# 10. Testing

* Unit Testing: Prompt functions tested with different inputs.
* Manual Testing: Verified UI flow, input-output consistency.
* Edge Cases: Handled missing inputs, long queries.

# 11. Screenshots

(\*To be added after running the app and capturing UI screenshots.\*)

# 12. Known Issues

* Limited to English language explanations.
* Performance depends on system hardware.
* No persistent storage for generated content.

# 13. Future Enhancements

* Add multi-language support.
* Expose REST API for third-party integration.
* Store and track student progress.
* Add voice-based interaction.
* Gamification features (badges, progress tracking).