**Edu Tutor AI – Personalized Learning**

# 1. Introduction

Project Title: Edu Tutor AI – Personalized Learning

Team Members:  
 • JEGANATHAN S  
 • HARIKRISHNAN S  
 • KALAISELVAM K  
 • LOGESHWARAN R

# 2. Project Overview

## Purpose

The purpose of Edu Tutor AI is to provide a personalized, AI-powered learning assistant that adapts to individual learners’ pace, style, and knowledge level. It enhances education by offering interactive tutoring, smart recommendations, and progress monitoring, ensuring students get the right guidance at the right time.

## Features

• Adaptive Learning Paths: Tailors course content based on student strengths and weaknesses.  
• Conversational Tutor: AI chatbot for instant doubt solving.  
• Assessment Generator: Creates quizzes and practice questions dynamically.  
• Progress Tracking: Monitors student performance with dashboards.  
• Content Summarization: Converts lengthy notes into concise points.  
• Multimodal Input Support: Handles text, PDFs, and images.  
• Recommendation Engine: Suggests resources and strategies.  
• Gamification: Rewards and badges to motivate learners.  
• Teacher Dashboard: Performance analysis for educators.

# 3. Architecture

Frontend (Streamlit/Gradio): Interactive UI with tutor, quizzes, progress graphs.  
Backend (FastAPI/Flask): APIs for quizzes, progress tracking, and adaptive learning.  
LLM Integration (OpenAI/IBM Watsonx Granite): AI tutor, summarization, Q&A.  
ML Modules: Adaptive learning engine, recommendation engine, performance predictor.  
Database (MongoDB/PostgreSQL): Stores student profiles, scores, and logs.

# 4. Setup Instructions

## Prerequisites

• Python 3.9+  
• pip & venv tools  
• API keys (LLM provider)  
• Database setup (MongoDB/PostgreSQL)

## Installation

1. Clone repository  
2. Install dependencies (requirements.txt)  
3. Configure .env with API keys  
4. Run backend server  
5. Launch frontend  
6. Start using Edu Tutor AI

# 5. Folder Structure

app/ – Backend APIs and ML logic  
 └── api/ – API routes (quiz, tutor, progress)  
ui/ – Frontend components  
edu\_dashboard.py – Main dashboard  
tutor\_llm.py – AI tutor chat logic  
quiz\_generator.py – Quiz creation  
progress\_tracker.py – Student analytics  
recommendation\_engine.py – Suggests resources

# 6. Running the Application

1. Start backend server  
2. Run frontend dashboard  
3. Students log in and interact with tutor  
4. Attempt quizzes and track progress  
5. Teachers access dashboards and reports

# 7. API Documentation

• POST /tutor/ask – AI tutor response  
• POST /quiz/generate – Generate quizzes  
• GET /progress/{student\_id} – Fetch progress  
• POST /feedback – Submit feedback

# 8. Authentication

• Token-based (JWT)  
• Role-based access (student, teacher, admin)  
• Planned: OAuth2/SSO

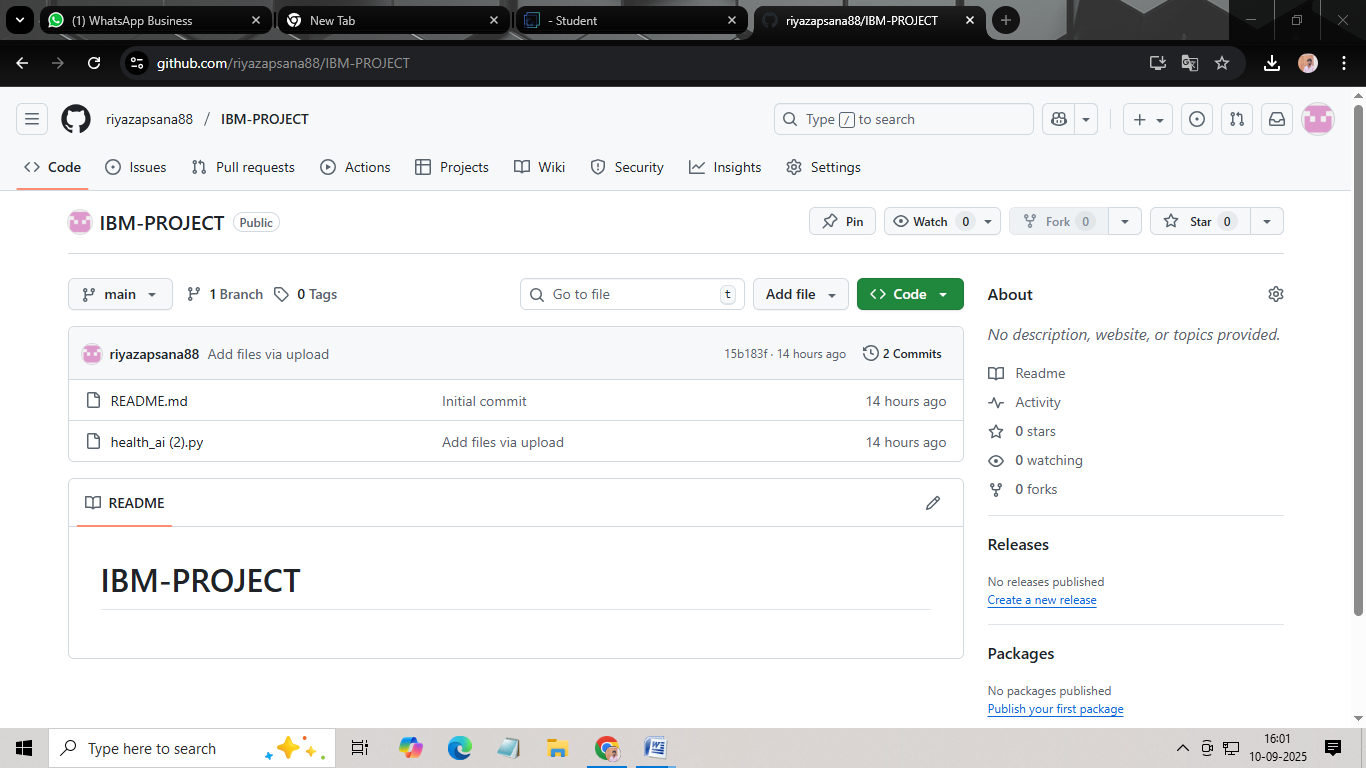
# 9. User Interface

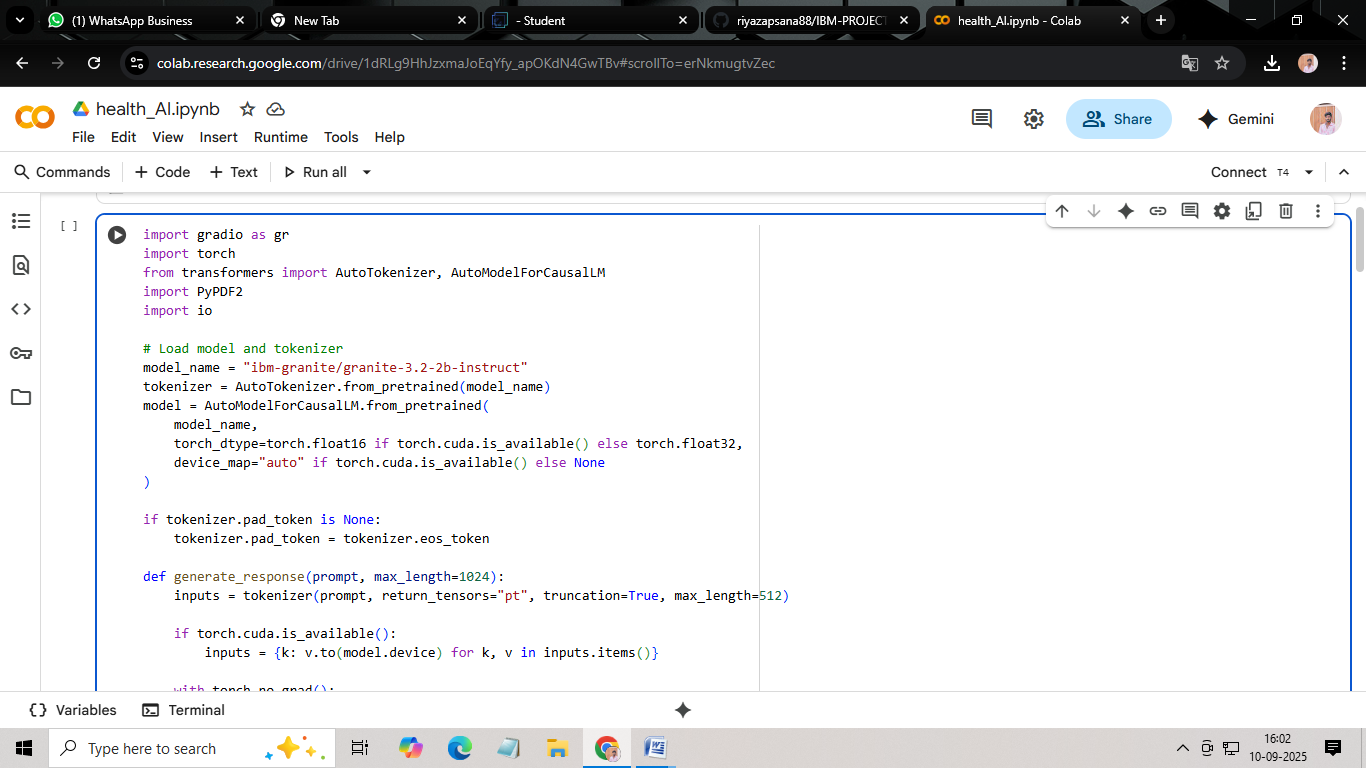
• Student Dashboard: Tutor chat, quizzes, progress graphs  
• Teacher Dashboard: Student analysis tools  
• Gamified interface for engagement

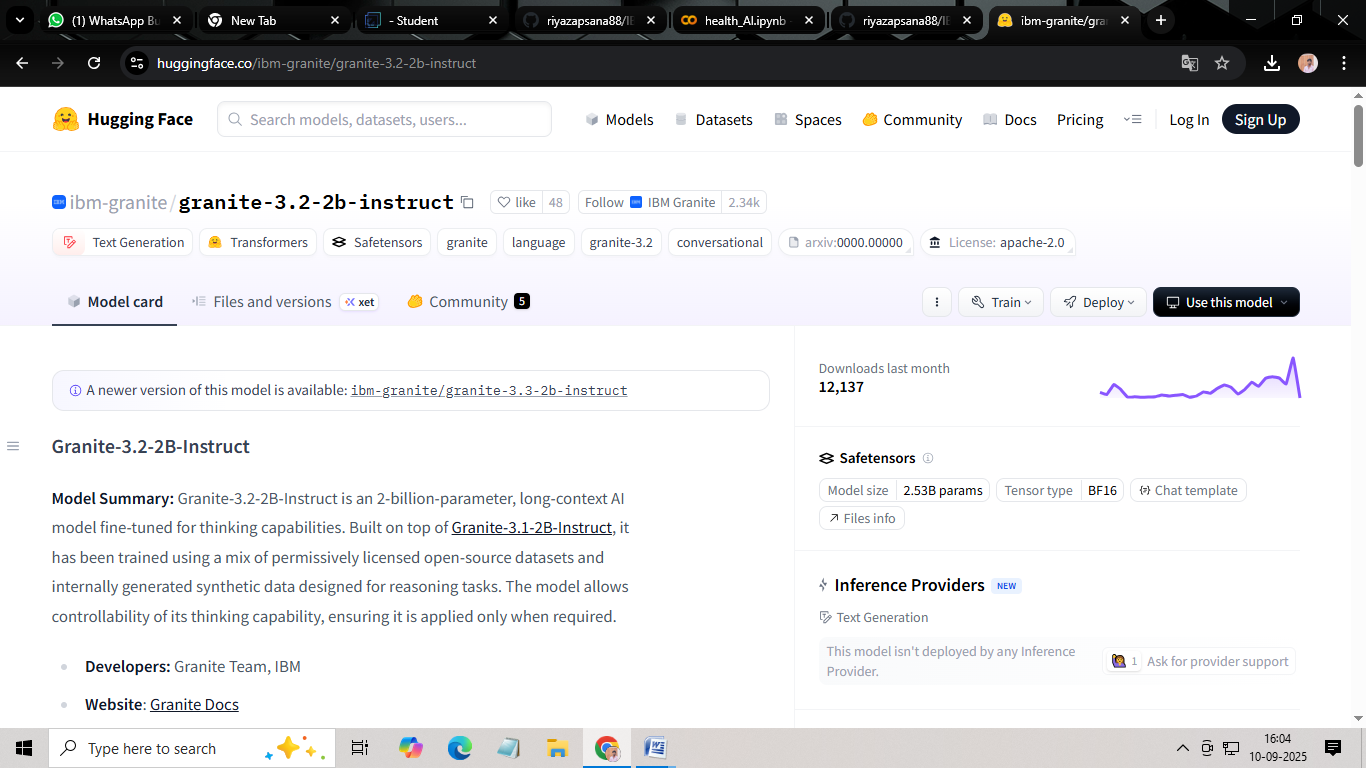
# 10. Testing

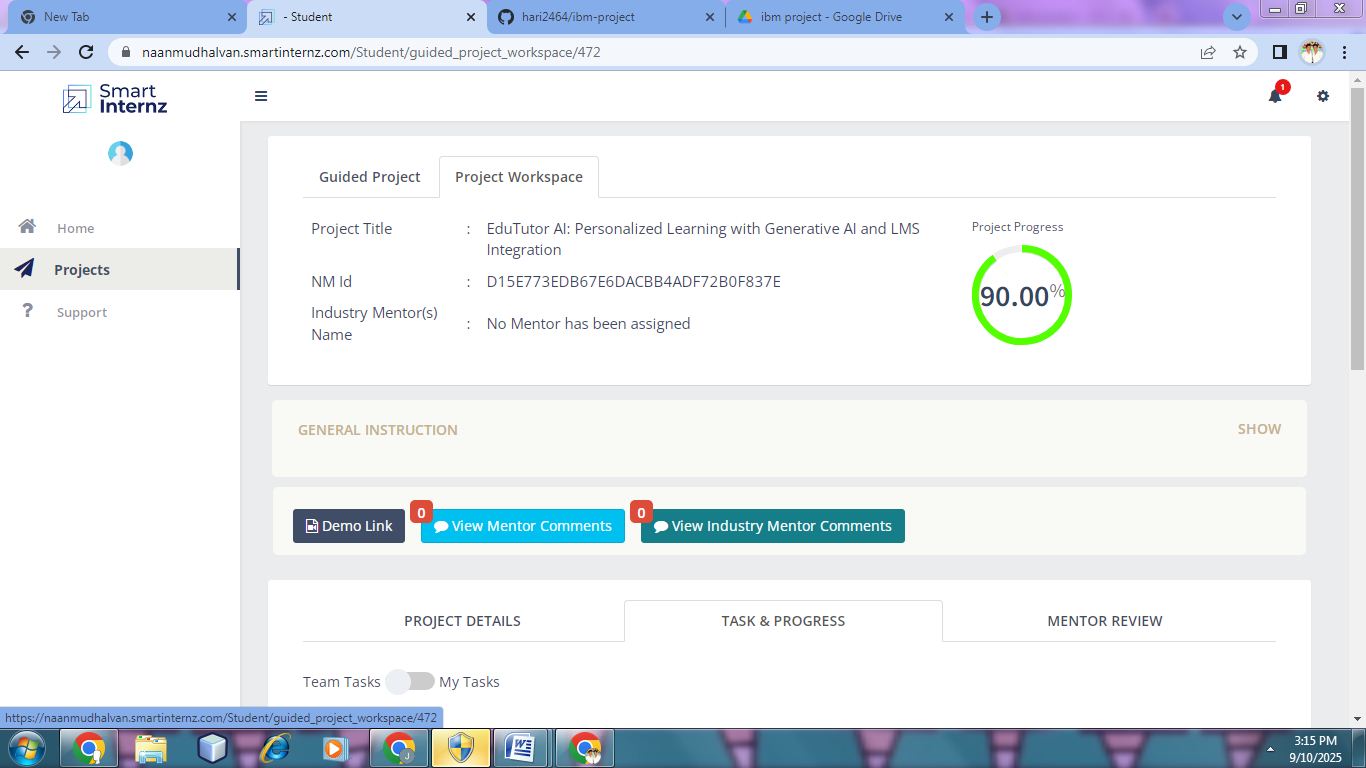
• Unit Testing: Quiz generation, recommendation engine  
• API Testing: Swagger/Postman  
• Manual Testing: Tutor chat responses, adaptive difficulty  
• Edge Cases: Wrong inputs, large files, API errors

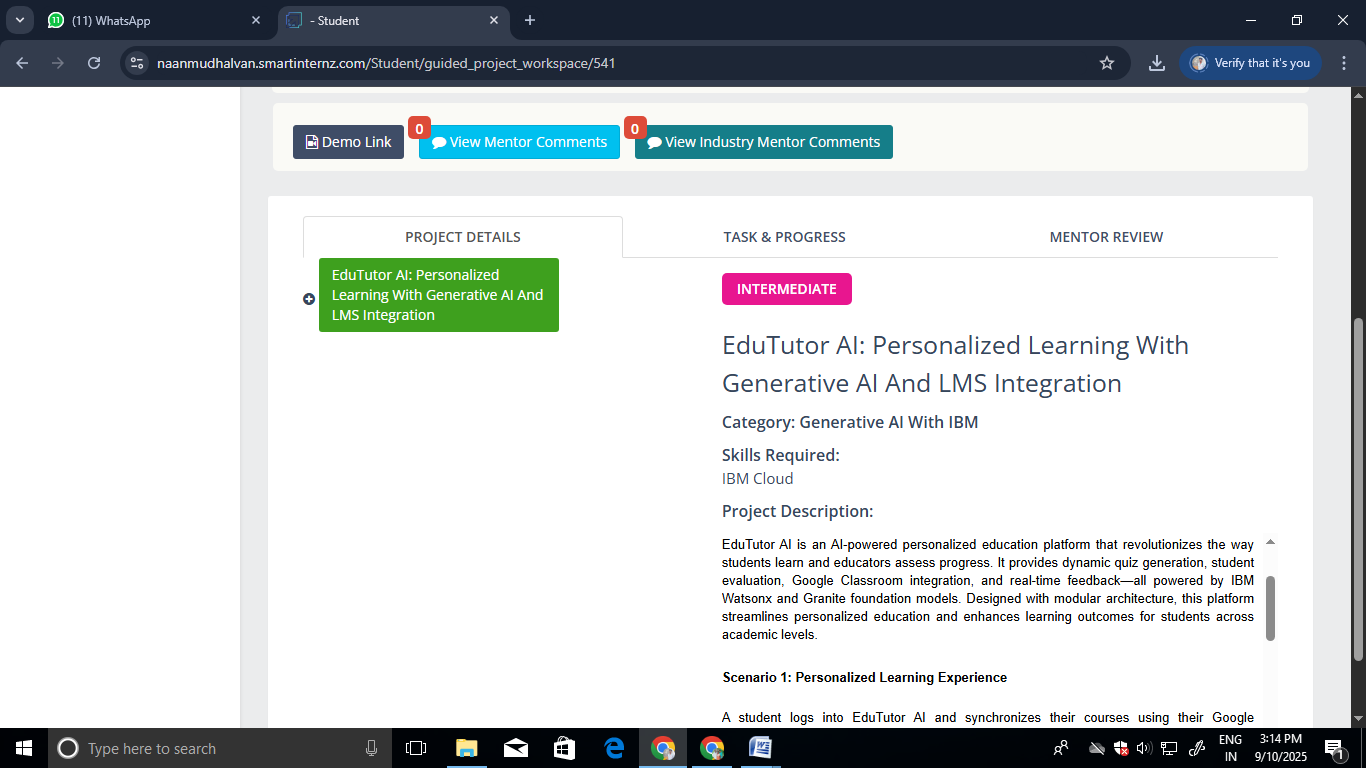
# 11. Screenshots











# 12. Known Issues

• Dependency on external LLM APIs  
• Limited offline functionality  
• Scalability for large classes

# 13. Future Enhancements

• Speech-to-text and voice tutor  
• Offline mode  
• Multi-language support  
• LMS integration (Moodle, Google Classroom)  
• Enhanced gamification features