Educational Organization

Project Documentation format

1. Introduction:

Project Title: Educational Organisation using Servicenow

Team Members:

Gorla Sai Charan : Leader / Organize

Avula Akhileswar : Research Helper

Thalla Indra Sreekar : Design Helper

Bandela Pravallika : Basic Coder

2. Project Overview:

Edu-AI is an AI-powered, full-stack cloud application designed to revolutionize traditional Educational Management Systems (EMS) by embedding intelligent automation across key academic and administrative workflows. Built on IBM Cloud and powered by the IBM Granite foundation model, Edu-AI harnesses advanced Natural Language Processing (NLP) and Generative AI to convert unstructured academic content—such as handwritten notes, raw lesson plans, or administrative policies—into structured educational artifacts like course outlines, quizzes, teaching modules, student progress reports, and automated documentation. The IBM Granite model acts as the cognitive engine throughout the platform, enabling deep semantic understanding and contextual content generation with high precision. By automating syllabus planning, content generation, assessments, student feedback analysis, and real-time virtual assistant support, Edu-AI drastically improves educational delivery, minimizes administrative overhead, and fosters inclusive participation from both educators and students, creating a smarter, AI-enabled educational ecosystem.

Purposes:

- Automate essential phases of academic and administrative workflows using AI.
- Increase educator efficiency by generating lesson plans, test papers, and feedback automatically.
- Translate unstructured academic inputs into structured course modules for clarity and scalability.
- Enhance teaching quality and reduce errors through AI-driven evaluation and content curation.
- Generate human-readable summaries of educational material to aid student learning and onboarding.

- Enable non-technical educators and administrators to contribute using natural language prompts.
- Provide a unified, intelligent academic environment powered by the IBM Granite foundation model.
- Reduce operational costs and teaching preparation time by eliminating repetitive manual tasks.
- Utilize IBM Cloud for secure, scalable, and enterprise-grade deployment in educational institutions.
- Drive innovation in teaching and learning through integration of AI and NLP in academia.

Features:

- Upload PDF documents containing raw, unstructured academic or policy content
- AI-powered classification of content into Educational Management phases (Curriculum Design, Lesson Planning, Delivery, Assessment, Feedback)
- Automatic generation of structured course outlines and weekly plans from uploaded content
- Natural language prompt-to-lesson content or test generation using the IBM Granite model
- AI-based error detection and correction for Python and JavaScript educational code snippets
- Automatic generation of quizzes, assignments, and test cases using unittest or pytest
- Course material summarization with student-friendly explanations for better understanding
- Floating AI chatbot assistant for real-time guidance on course content creation and EMS navigation
- Clean, syntax-highlighted educational code and content display on the frontend
- Fully cloud-based architecture hosted on IBM Cloud for seamless access and uptime
- Modular components — use each capability independently or as part of a complete EMS pipeline
- Smart prompt routing and academic keyword interpretation using LangChain to power the chatbot

3. Architecture:

Frontend:

The Gradio frontend for Edu-AI is designed to offer an intuitive, educator- and student-friendly interface for seamless interaction with the Edu-AI platform. Below is a breakdown of the key components:

1. Interface Structure
   The interface is built using Gradio's Blocks API, enabling a structured and modular layout. It includes:

- A title and subtitle explaining the Edu-AI purpose

- Three tabs:
    - Ask Edu-AI – for general academic or system queries
    - Generate Course Content – for converting inputs into lectures, quizzes, or outlines
    - Student Performance Analysis – for analyzing student submissions or engagement

2. Tab Components
   Each tab is equipped with:

- Input fields (e.g., Textbox, Dropdown) for educators to enter topics, assessments, or student data
- Buttons (e.g., Button) to trigger AI-driven responses
- Output fields (e.g., Textbox, Label, HighlightText) to display the generated lesson content, quizzes, or analysis results

3. Event Handlers
   The interface connects buttons to backend logic through event handlers:

- handle_academic_query: Processes general queries about academic tasks or topics
- handle_course_generation: Generates teaching content, test papers, or course plans
- handle_performance_analysis: Analyzes student answers or engagement metrics

These handlers invoke methods from the EduAI class, which interacts with the IBM Granite model to deliver context-aware responses.

4. Customization
   The interface is enhanced with HTML components (e.g., gr.HTML) to include:

- A stylized header and subtitle describing Edu-AI's functionality
- A footer highlighting: “Powered by IBM Granite Foundation Model & IBM Cloud”

5. Launch Configuration
   The interface uses share=True in iface.launch(), which generates a public, shareable link. This link allows students, teachers, and administrators to use the platform remotely for up to 72 hours.

Backend:

This version of Edu-AI is designed to run on Google Colab, using Colab's cloud compute backend. Here’s how it integrates:

1. Library Installation
   The setup begins with installing essential libraries including transformers, torch, gradio, accelerate, and bitsandbytes via !pip install. This approach aligns with best practices for running AI models in Colab notebooks.

2. Model Loading and Inference
   The EduAI class loads pre-trained models like:

- "ibm-granite/granite-3.3-2b-instruct" for academic content generation
- "microsoft/DialoGPT-medium" for general educational chat responses

These models are managed via the transformers library, with Colab's infrastructure handling heavy lifting for inference and generation tasks.

3. Gradio Interface
   A Gradio interface wraps around the model's functionality, enabling an interactive and visual layer for educators and students to use AI capabilities without needing coding experience.

4. Public Link Generation
   When launched with share=True, Gradio provides a temporary public URL that remains active for 72 hours. This makes it easy for faculty, students, or administrators to access Edu-AI from any device without any installations.

4. Setup Instructions

Prerequisites:

Before starting the Edu-AI project in Google Colab, ensure the following are available:

• Google Account to access and run Google Colab notebooks

• IBM Cloud Account with access to Watsonx.ai and the Granite 3.2 Instruct model

• IBM Cloud API Key with access rights to Watsonx foundation model

• Colab-compatible Python environment (Colab default: Python 3.10+)

• Required Python packages (listed below)

♦ Installation & Configuration Steps in Google Colab

You can run the full project pipeline inside a Google Colab notebook. Here's a step-by-**step guide:**

1. Set Up Required Python Packages
   Install required packages in your Colab environment:

python

CopyEdit

!pip install transformers torch gradio accelerate bitsandbytes PyPDF2

5. Folder Structure

Since EDU-AI was developed entirely within a Google Colab notebook, the project does not follow a conventional file/folder structure (e.g., separate /client and /server directories). Instead, the entire system is organized into modular notebook cells, each representing logical components of the educational automation pipeline.

¬ Notebook-Based Architecture Overview
Instead of folders, the Colab notebook is divided into the following logical sections (cell groups):

1. PDF Upload & Content Extraction
   • Uses PyMuPDF to read academic PDF content (e.g., syllabus, lesson plans, policy documents)
   • Extracts and preprocesses raw educational material

2. IBM Granite Model Integration
   • Authenticates with Watsonx.ai using the ibm-watson-machine-learning SDK
   • Calls Granite v3.2 Instruct for various tasks:
   o Classifying academic content into EMS phases
   o Generating educational content
   o Auto-correcting code written by students
   o Creating test cases and assessments
   o Summarizing content for learning and documentation

3. Classification & Course Outline Generation
   • Splits extracted content into instructional units
   • Prompts Granite to classify each section
   • Optionally formats output into structured weekly plans or course outlines

4. Content Processing Modules
   • Separate cells for:
   o Lesson content generation from academic prompts
   o Auto-correction of Python/JavaScript code
   o Test and quiz generation
   o Educational content summarization

5. Chatbot Assistant (Optional)
   • Implements a simple chatbot interface (text input/output)
   • Routes user queries related to the Educational Management System to Granite with tailored prompts

6. Output Display and Export
   • Displays generated lessons, quizzes, summaries, and feedback directly in Colab
   • Optionally saves outputs to Google Drive or allows them to be downloaded as files

6. Running the Application

Frontend : Frontend Server (Gradio):
Since the frontend is built using Gradio, it's not a traditional frontend framework like React or Angular that would use npm start. Instead, the frontend server is launched using the launch() method in the Gradio code:

python

CopyEdit

iface.launch(share=True)

To run the frontend server, you would execute the Python script containing this code.

Backend :
The backend server is also built using Python, leveraging libraries like transformers and torch. The backend logic is contained within the EduAI class and its methods.
To run the backend server, you would execute the same Python script that launches the Gradio frontend. The backend logic is tightly coupled with the frontend in this implementation.

If you were to separate the frontend and backend into distinct projects (e.g., a React frontend and a Python backend), the commands might look like this:

Backend Server:
Assuming a Python backend with a Flask or FastAPI server:

bash

CopyEdit

cd backend

python app.py

7. API Documentation:

Edu-AI runs entirely within a Google Colab notebook and directly interacts with IBM Watsonx's Granite v3.2 Instruct model using the IBM Watson Machine Learning Python SDK. It does not expose any public REST API endpoints, but it internally follows a modular, function-based structure for various educational tasks.

8. Authentication:

♦ Hugging Face API Key Authentication
Edu-AI uses the Hugging Face-hosted IBM Granite v3.2 Instruct model, accessed through the Hugging Face Inference API. Authentication is handled via a personal API key provided by Hugging Face.
This key allows authorized access to large language models hosted on the Hugging Face platform without managing a complex user login system.

• How It Works in the Project:
You authenticate by passing the API key as a Bearer token in the HTTP request headers:

python

CopyEdit

```
import requests

API_URL = "https://api-inference.huggingface.co/models/ibm/granite-3b-instruct"

headers = {

   "Authorization": f"Bearer YOUR_HUGGINGFACE_API_KEY"

}

response = requests.post(API_URL, headers=headers, json={"inputs": prompt})
```

9. User Interface:

Citizen - Ai is designed to run in an interactive **Google Colab notebook**, where each module functions as a logically separated UI block. Users interact with the system by uploading files, entering text, and viewing AI-generated outputs directly within the notebook interface.

Edu-AI

AI-powered Educational Management Platform

Course Generator | Performance Analysis

Enter your query

How can I generate a Python quiz?

Get AI Response

Edu-AI Response

Query: How can I generate a Python quiz? Here's a simplified process:

1. "Enter the Topic": Begin by typing the topic yu'd like the quiz to cover (e.g. Python Loops, Functions, Data Types)

2. "Select Format": Choose the quiz format. Ed-AI supports:

- Multiple-Choice Questions (MCQs)
- Fill-in-the-blanks
- True/False

Citizen AI

AI-powered Intelligent Educationl Platform

Ask Citizen AI | Academic Institutes | Career Advice

Enteryour city name

Pune

Get Academic Institutes

Academic Institute Information

You are an AI education assistant. Provide a list of promineent accademic institute (universities, colleges, polytechn|a forthe cityl or the : Pume.

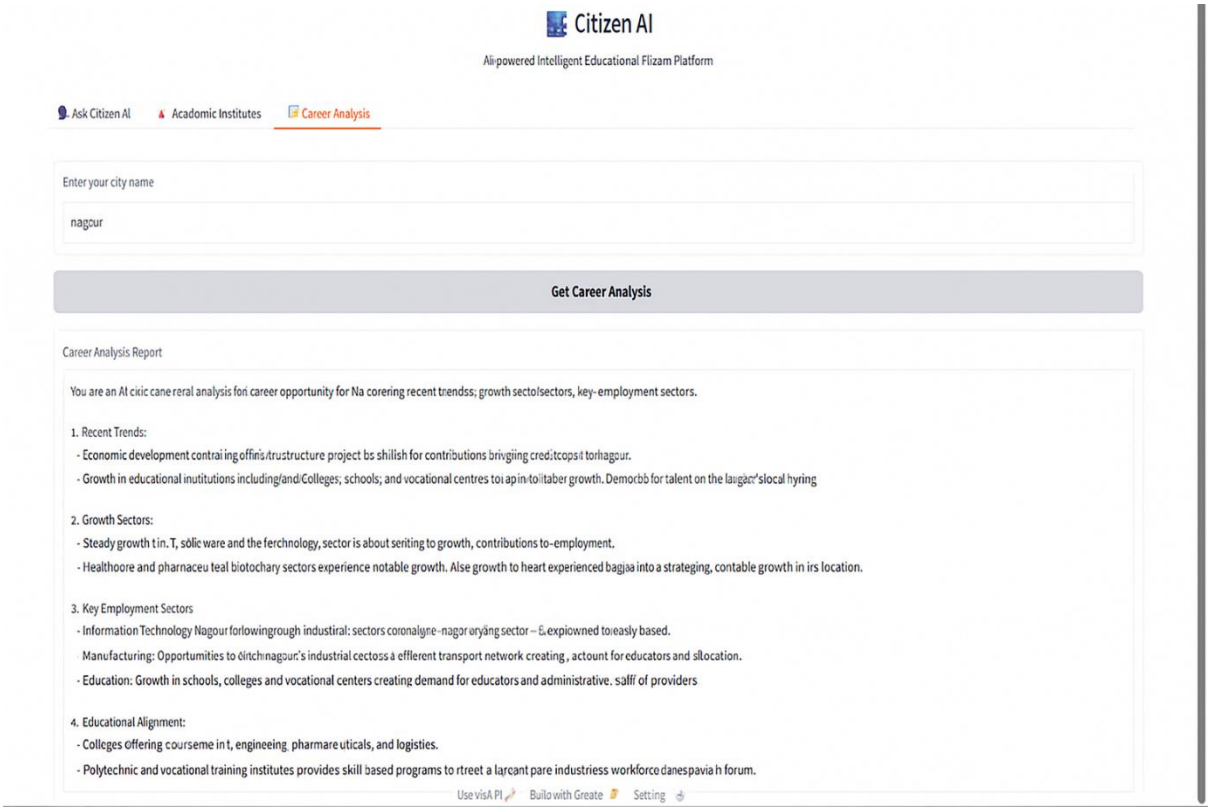
Here Lttle list:

1. Pune University
2. College of Engineering Pune
3. Symblogis Institute of Techhology
4. Mehasoaficical Institute of Technology

Please check their websites for precise innc iriomation as Institutes have multiple *carnpup1.s* ***and*** *offer. fiversa.* cirtasspecializations.

Powered by IBM Granite AII CitizenAI Platform

Use via API | Built with Gradie | Settings



10. Testing:

- Testing Strategy

  Edu-AI is tested through manual testing and functional validation using the Gradio interface. Each module is designed as a user-facing interactive block, allowing rapid iteration and debugging.

What Was Tested:

• PDF Parsing: Verified extraction of academic content from diverse PDF formats using PyPDF2.

• Model Output Validation:

o Course generation prompts yield structured, relevant instructional material.

o Quiz and test creation prompts generate correct and formatted assessments.

• Fallback Model Handling: Ensured system remains operational with a secondary model (DialoGPT-medium) when the IBM Granite model fails to load.

¬ Tools Used:

• Google Colab runtime

• Gradio's live UI for visual verification

• Print/log statements for model loading/debug fallback

11. Screenshots or Demo:
Demo Link:
https://drive.google.com/file/d/1wZ2bkjZ081VKW55Oay_t56EHIBeCBlra/view?usp=drivesdk

12. Known Issues:

Here are some known bugs or limitations that may affect users or developers:

• Model Load Time:
The IBM Granite 3.3-2B Instruct model is large and can take considerable time to load in Colab, especially on free-tier runtimes.

• Memory Constraints in Google Colab:
Large models may cause memory overflows or slow performance on limited resources.

• Fallback Model Simplicity:
The fallback model (DialoGPT-medium) is significantly less capable and may produce generic or unrelated outputs.

• PDF Parsing Limitations:
Some academic PDFs, especially scanned documents or those with unusual formatting, may not be parsed accurately by PyPDF2.

• Response Variability:
Generative AI outputs may vary between runs and may include irrelevant or incomplete educational content.

• No Persistent Storage:
There's no backend database or file-saving mechanism, so all generated lessons, tests, or code are lost when the session ends unless manually saved.

• No Authentication:
The app does not currently restrict access or protect the Hugging Face API key, which can be a security risk in shared classroom or institutional environments.

13. Future Enhancements:

To improve usability, scalability, and feature richness, the following enhancements are recommended:

Functional Enhancements:
• Add More Modules:
Incorporate additional tabs for:
o Bug Fixing in Student Code

o Test Case and Quiz Generation
o Lesson Summarization
o Assignment Feedback and Suggestions

• Custom Model Selector:
Let educators or admins choose between models (e.g., IBM Granite, GPT-4, Mistral) for different educational tasks.

• Improve Prompt Engineering:
Use structured prompt templates and dropdowns to tailor AI responses more precisely for lesson planning or grading.

Application & UI Improvements:
• Persistent Storage:
Add integration with cloud storage or databases (e.g., Firebase or MongoDB) to save user sessions, assignments, and generated content.

• Export Functionality:
Allow users to download generated material (lesson plans, code, quizzes) in .txt, .pdf, or .py formats.

• Code Execution Cell:
Let users test AI-generated code (e.g., for Python programming classes) directly within the app using secure sandboxing.

• Enhanced Error Handling:
Provide clearer error messages, especially during model loading, API failures, or input parsing.

Security Enhancements:
• API Key Protection:
Store the Hugging Face API key in environment variables or through a proxy server to prevent exposure in the notebook.

• User Authentication (Optional):
Add user login support if the platform is shared publicly or used across institutions for session and data management.

Deployment Options:
• Dockerize the App:
Containerize Edu-AI with Docker for easy deployment on IBM Cloud or school-owned infrastructure.

• Host on a Web Server:
Convert the Colab-based app to a fully hosted web application using Flask + Gradio, and deploy on IBM Cloud, Hugging Face Spaces, or internal servers.