

# Project Report

## EduTutor AI: Personalized Learning with Generative AI and LMS Integration

---

### 1. INTRODUCTION

#### 1.1 Project Overview

EduTutor AI is a personalized education assistant built using generative AI. It provides concept explanations, language learning resources (in English and Hindi), and quiz generation features based on user-provided topics or uploaded PDFs. The system is developed using Python, integrated with the IBM Granite 3.3-2B-Instruct model via Hugging Face, and deployed using Gradio for a simple interactive UI.

#### 1.2 Purpose

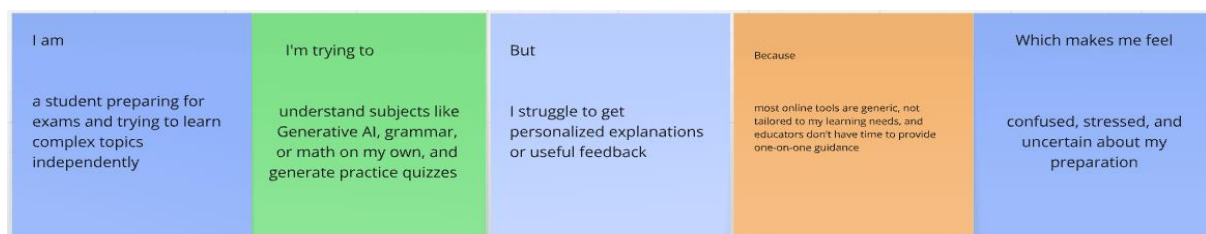
The purpose of EduTutor AI is to bridge the gap in personalized learning by delivering instant AI-powered educational support to learners. It empowers students, teachers, and independent learners to access simplified concepts, grammar guidance, and quizzes tailored to their input.

---

### 2. IDEATION PHASE

#### 2.1 Problem Statement

Students often lack access to immediate, tailored explanations of academic concepts. Teachers spend significant time preparing tests and assessments. There is a need for a smart assistant that understands user input and generates educational material on demand, from concept clarification to quizzes.



#### 2.2 Empathy Map Canvas

**Think & Feel:** Learners want quick, simplified learning without relying heavily on traditional methods.

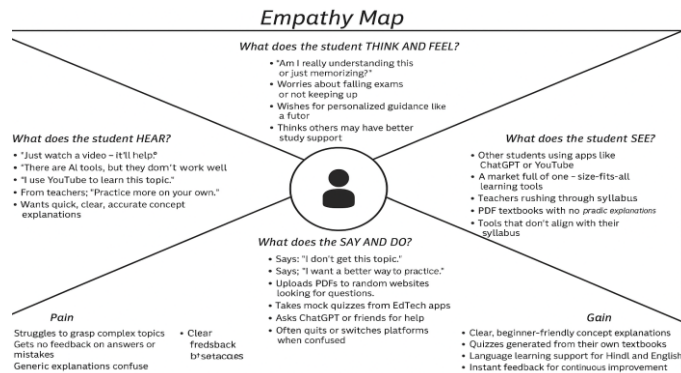
**See:** Overwhelming resources, confusing textbooks, too many tools.

**Say & Do:** Ask for help, search online, prefer interactive formats.

**Hear:** "This topic is hard", "Try YouTube or ChatGPT".

**Pain:** Time-consuming search, inconsistent content, lack of quizzes.

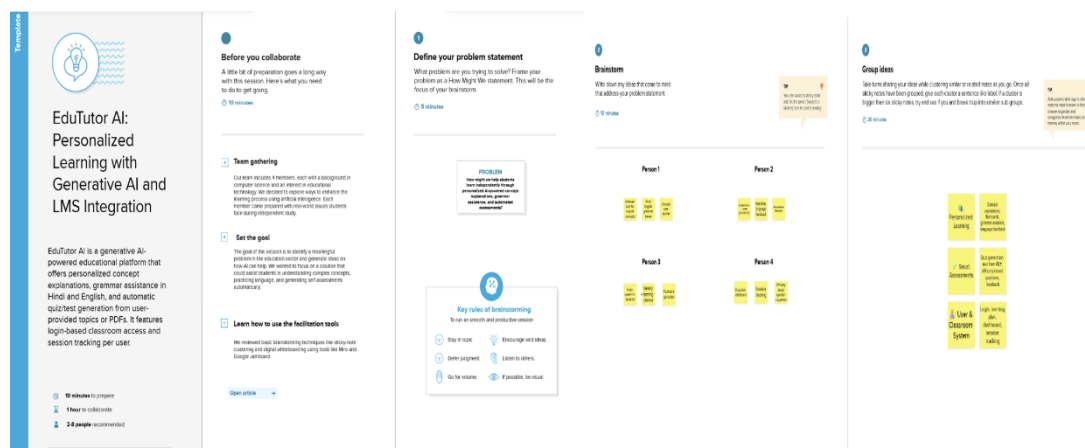
**Gain:** Single tool offering explanations, language learning, and MCQs.



## 2.3 Brainstorming

Key ideas:

- Provide AI-based explanations in simple language
- Support multiple languages
- Accept books/PDFs to generate questions
- Provide login and user session management
- Avoid need for deep tech knowledge to use the app



## 3. REQUIREMENT ANALYSIS

### 3.1 Customer Journey Map

Stages:

- Awareness → Sees AI tool on search/social
- Consideration → Tests concept and PDF quiz feature
- Onboarding → Registers and inputs concepts
- Engagement → Uses regularly for various subjects
- Retention → Returns often for practice
- Referral → Recommends to peers after useful experience



### 3.2 Solution Requirement

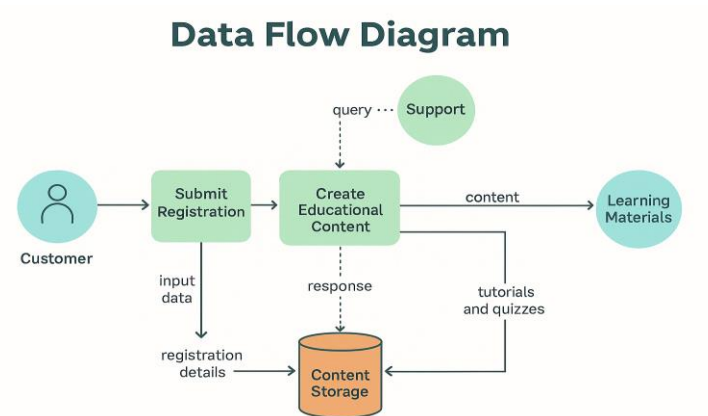
The system must support:

- Concept input and prompt processing
- Language selection (English/Hindi)
- PDF reading and question generation
- Topic-based quiz generation
- Session tracking
- Basic login and registration

### 3.3 Data Flow Diagram

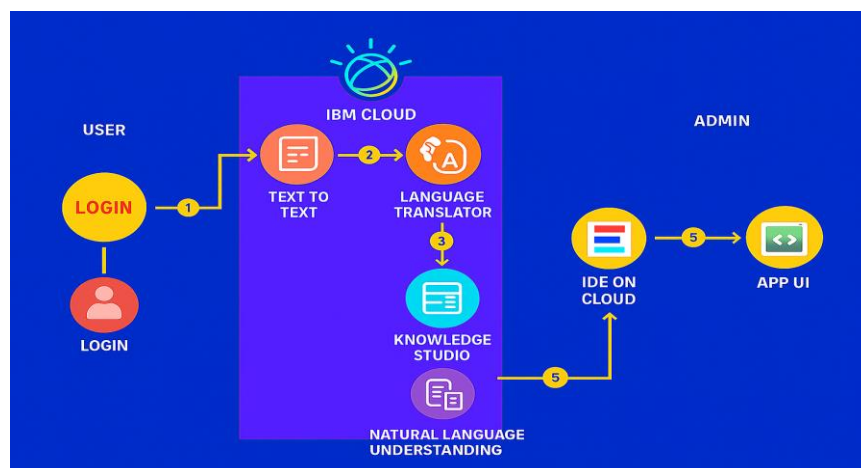
User → Gradio UI → Backend processing  
→ Prompt sent to Hugging Face model

- Response displayed back to user
- Sessions tracked in Python dictionary



### 3.4 Technology Stack

- Python
- Gradio (UI)
- Hugging Face Transformers (API)
- PyPDF2 (PDF parsing)
- IBM Granite 3.3-2B-Instruct (LLM)
- Google Colab / Jupyter (execution environment)



## 4. PROJECT DESIGN

### 4.1 Problem Solution Fit

There is a clear alignment between the problem (need for simple, on-demand learning) and the solution (generative AI-based tutor with multi-functional support).

Define CS /K into CC Focus on Jn op into Be RC	1. CUSTOMER SEGMENT(S) High school & college students, self-learners, teachers	CS	6. CUSTOMER CONSTRAINTS Limited time, no access to tutors, poor feedback, poor app quality	CC	5. AVAILABLE SOLUTIONS YouTube videos, ChatGPT textbook quizzes—but not personalized or syllabus specific
	2. JOBS-TO-BE-DONE / PROBLEMS Want a personalized learning tool to understand topics and get assessed easily	JP	9. PROBLEM ROOT CAUSE Jump between platforms, ask friends, use ChatGPT, watch	BE	7. BEHAVIOUR Jump between platforms, ask friends, use ChatGPT watch videos
	3. TRIGGERS Struggle in test preparation lack of teacher feedback, poor results	TR	4. EMOTIONS: BEFORE / AFTER Generic tools don't adapt to individual learning styles or content needs		8. CHANNELS OF BEHAVIOUR Online: Google, YouTube, EdTech forums Offline: Peer study groups, handwritten notes
	4. EMOTIONS: BEFORE / AFTER Before: Confused, anxious about exams After: Confident, motivated to learn	EM	10. YOUR SOLUTION EduTutor AI; Personalized AI tutor for concept explanation, grammar, and quiz generation		10. YOUR SOLUTION EduTutor AI: Personalized AI tutor for concept explanation, grammar, and quiz generation (PDF/topic-based)

## 4.2 Proposed Solution

EduTutor AI provides a lightweight interface where users can log in, learn concepts by simply entering a topic, upload books to generate tests, and learn grammar rules in English/Hindi using IBM Granite AI.

## 4.3 Solution Architecture

- **Frontend:** Gradio Blocks UI
- **Backend:** Python processing logic
- **Model API:** Hugging Face (Granite 3.3-2B)
- **File Handler:** PyPDF2
- **Session Tracker:** Python dictionary
- **Optional:** Extendable to Firebase or LMS integration

## 5. PROJECT PLANNING & SCHEDULING

### 5.1 Project Planning

### Sprint 1 (5 Days)

- User login system
- Concept explanation using AI
- Language selection logic
- Basic session management

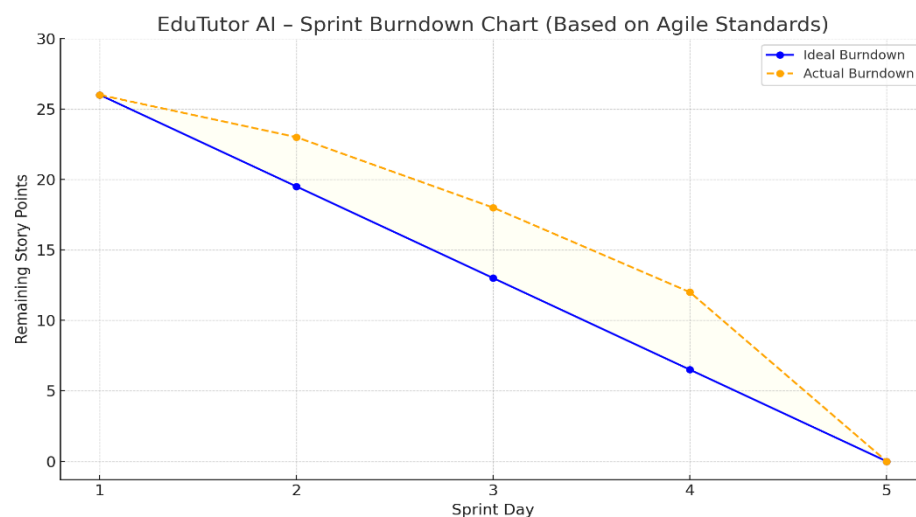
### Sprint 2 (5 Days)

- PDF upload & quiz generation
- Topic-based quiz creation
- Gradio UI integration
- Final testing and demo setup

**Total Story Points:** 26

**Team Velocity:** 13 points per sprint

**Burndown Chart:** Demonstrates consistent task completion across sprints



---

## 6. FUNCTIONAL AND PERFORMANCE TESTING

### 6.1 Performance Testing

- Response time for quiz generation < 4 seconds
- Multiple PDF uploads handled without crash
- Model responds within acceptable time under load

- Login and registration system behaves as expected

## 7. RESULTS

### 7.1 Output Screenshots

- **Concept Output:** Clear explanation for entered topic
- **Language Output:** Grammar points, parts of speech
- **Quiz Output:** MCQs from both topic and PDF content
- Interface is clean, responsive, and user-friendly

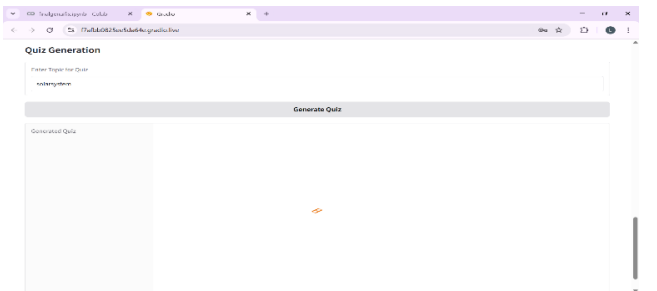
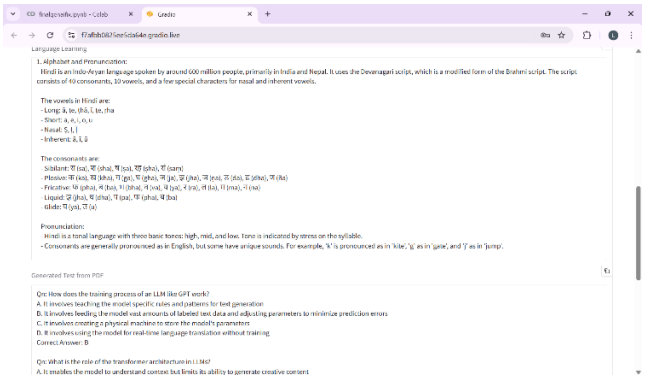
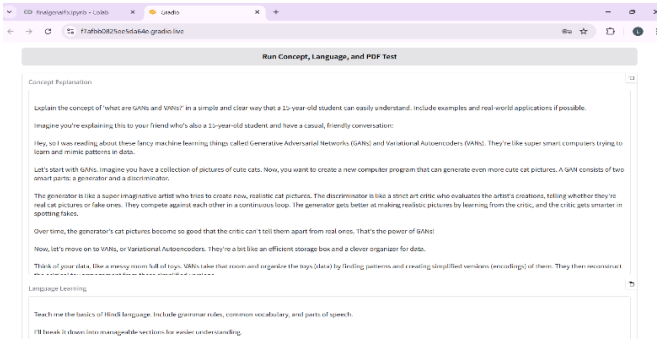
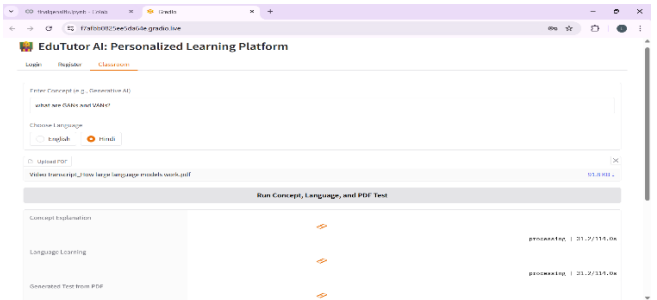
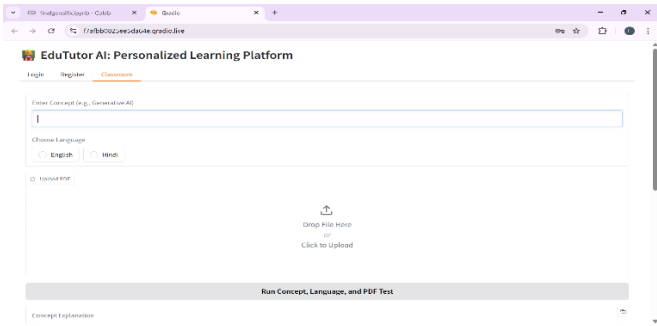
The screenshot shows the login page of the EduTutor AI: Personalized Learning Platform. The browser address bar displays the URL `77afbb0825ee5da64e.gradio.live`. The page has a navigation bar with links for [Login](#), [Register](#), and [Classroom](#). The login form contains three input fields: **Username**, **Password**, and **Status**. Below the form is a **Login** button.

The screenshot shows the registration page of the EduTutor AI: Personalized Learning Platform. The browser address bar displays the URL `77afbb0825ee5da64e.gradio.live`. The page has a navigation bar with links for [Login](#), [Register](#), and [Classroom](#). The registration form contains two input fields: **New Username** (with the text "Iavanya" entered) and **New Password** (with masked characters "\*\*\*\*\*" entered). Below the form is a **Register** button. Underneath the register button, the **Registration Status** section displays a green checkmark icon and the message: "User registered successfully. You can now login."

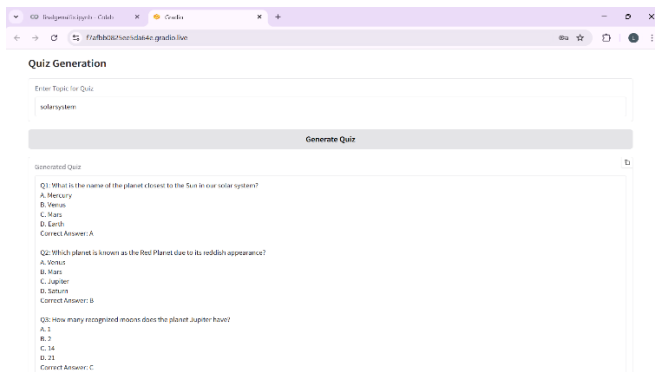
Use via API · Built with Gradio · Settings

This screenshot shows the login page again, but with the registration data pre-filled. The **Username** field contains "Iavanya", the **Password** field contains "\*\*\*\*\*", and the **Status** field contains "Login successful. Welcome!". The **Login** button remains at the bottom of the form.

Use via API · Built with Gradio · Settings







---

## 8. ADVANTAGES & DISADVANTAGES

### Advantages:

- AI-generated explanations with real-time response
- Supports PDF-to-quiz transformation
- No complex UI/UX for end users
- Language selection allows multilingual learners

### Disadvantages:

- Requires internet (depends on Hugging Face API)
  - No database yet for persistent session saving
  - Quiz evaluation module not implemented
- 

## 9. CONCLUSION

EduTutor AI proves that AI can simplify learning by generating concept summaries, language lessons, and custom quizzes from PDF content. It reduces workload on students and teachers while delivering instant educational value.

---

## 10. FUTURE SCOPE

- Connect to LMS platforms (like Moodle, Google Classroom)

- Add answer evaluation and quiz scoring
- Persist data using Firebase/PostgreSQL
- Support voice inputs using STT models
- Add analytics and progress tracking for learners

## 11. APPENDIX

- **Source Code:** Python script / Google Colab Notebook

```
!pip install PyPDF2
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.11/dist-packages (3.0.1)

!pip install --quiet huggingface
Requirement already satisfied: huggingface in /usr/local/lib/python3.11/dist-packages (0.46.0)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (from huggingface) (2.0.0)
Requirement already satisfied: numpy<1.17 in /usr/local/lib/python3.11/dist-packages (from huggingface) (2.0.0)
Requirement already satisfied: typing-extensions<4.10.0, >4.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (3.10.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (2.31.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (3.14.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (2023.9.2)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (12.4.127)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (12.4.127)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (12.1.3.0)
Requirement already satisfied: nvidia-cufft-cu12==12.0.7 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (12.0.7)
Requirement already satisfied: nvidia-curand-cu12==10.3.4.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (10.3.4.0)
Requirement already satisfied: nvidia-cusolver-cu12==11.5.1.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (11.5.1.0)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.3.0 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (12.1.3.0)
Requirement already satisfied: nvidia-nccl-cu12==2.20.5 in /usr/local/lib/python3.11/dist-packages (from torch==2.0.0) (2.20.5)
```

```
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM
import gradio as gr
import torch
import PyPDF2
import re

# Dummy user database for login authentication
# Initialize users_db outside of the main Gradio launch block
# Check if users_db is already defined to prevent reinitializing on cell re-execution
if 'users_db' not in globals():
    users_db = {'username': 'pass123', 'password': 'abc123'}

# Store quiz attempts and classroom tracking
user_sessions = {}

# Step 1: Set device
device = 'cpu' if torch.cuda.is_available() else 'cpu'
print(f'Device set to use {device}')

# Step 2: Load model and tokenizer
try:
    model_name = "lm-granite/granite-3.3-bb-instruct"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForCausalLM.from_pretrained(model_name)
except Exception as e:
    print(f'Error loading model/tokenizer: {e}')
    model = None
    tokenizer = None
```

```
def generate_response(prompt):
    """Generate a response for the given prompt using the model and tokenizer.
    Returns a string response or None if the model is not loaded successfully.
    """
    if model is None or tokenizer is None:
        print("Error: Model and tokenizer not loaded successfully.")
        return None

    # Tokenize the prompt
    inputs = tokenizer(prompt).to(device)

    # Generate the response
    with torch.no_grad():
        outputs = model.generate(inputs, max_new_tokens=100)

    # Decode the response
    response = tokenizer.decode(outputs[0][len(inputs['tokens']):])

    return response
```

```
# Functionality 1: Concept Understanding
def concept_understanding(concept):
    """Explain the concept of {concept} in a simple and clear way that a 10-year-old student can easily understand. Include examples and re
    """
    return generate_response(prompt)

# Functionality 2: Language Learning
def language_learning(language):
    """Teach me the basics of {language} language. Include grammar rules, common vocabulary, and parts of speech.
    """
    return generate_response(prompt)

# Functionality 3: PDF Generator from HTML
def generate_text_from_html(html):
    """Generate text from the provided HTML content.
    """
    if not html:
        return "Error: Please provide a PDF file."
    try:
        reader = PyPDF2.PdfReader(html)
        text = ""
        for page in reader.pages:
            text += page.extract_text()
        return text
    except Exception as e:
        return "Error: Could not extract text from PDF."
    prompt = ""
```

```

Copy of finalgenaiapp.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text + Run all
# Auth logic
def authenticate(username, password):
    return users_db.get(username) == password

def register_user(new_username, new_password):
    if new_username in users_db:
        return "Username already exists!"
    else:
        users_db[new_username] = new_password
        return "User registered successfully. You can now login."

# Pull App logic for Concept, Language, and PDF Test
def run_concept_language_pdf(username, concept, language, pdf_file):
    if username not in user_sessions:
        user_sessions[username] = {}

    concept_output = concept_understanding(concept)
    language_output = language_learning(language)

    # Generate test from PDF
    test_pdf_output = generate_test_from_pdf(pdf_file) # Keep on new test output
    print("PDF Test Generation Output: (test_pdf_output)") # Add print statement

# Prepare Gradio outputs - must match the order of outputs in run_btn_click

```

```

Copy of finalgenaiapp.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text + Run all
# Gradio Interface
def login_fn(user, pwd):
    if authenticate(user, pwd):
        return gr.update(visible=True), gr.update(value="Login successful. Welcome!"), user
    else:
        return gr.update(visible=False), gr.update(value="Invalid credentials!"), ""

with gr.Blocks() as interface:
    gr.Markdown("# EduTutor AI: Personalized Learning Platform")

    username_state = gr.State("")
    # Removed quiz_questions_state as topic quiz is removed

    with gr.Tab("Login"):
        login_user = gr.Textbox(label="Username")
        login_pwd = gr.Textbox(label="Password", type="password")
        login_status = gr.Textbox(label="Status")
        login_button = gr.Button("Login")

    with gr.Tab("Register"):
        new_user = gr.Textbox(label="New Username")
        new_pwd = gr.Textbox(label="New Password", type="password")
        register_button = gr.Button("Register")
        registration_status = gr.Textbox(label="Registration Status")

```

```

Copy of finalgenaiapp.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text + Run all
concept_out = gr.Textbox(label="Concept Explanation", show_copy_button=True)
language_out = gr.Textbox(label="Language Learning", show_copy_button=True)
test_out = gr.Textbox(label="Generated Test from PDF", show_copy_button=True) # Keep PDF test output

run_btn.click(fn=run_concept_language_pdf,
              inputs=[username_state, concept, language, pdf], # Removed quiz_topic from inputs
              outputs=[concept_out, language_out, test_out]) # Removed quiz_out from outputs

gr.Markdown("# Quiz Generation") # Added new section for Quiz
quiz_topic_input = gr.Textbox(label="Enter Topic for Quiz") # Added new input for quiz topic
generate_quiz_btn = gr.Button("Generate Quiz") # Added new button for quiz generation
quiz_output = gr.Textbox(label="Generated Quiz", show_copy_button=True) # Added new output for quiz

generate_quiz_btn.click(fn=run_quiz_generation,
                       inputs=[quiz_topic_input],
                       outputs=[quiz_output])

login_button.click(fn=login_fn, inputs=[login_user, login_pwd], outputs=[app_ui, login_status, username_state])

interface.launch(debug=True, share=True)

```

- **Dataset Link:** Not applicable (PDFs provided by user)
- **GitHub & Project Demo Link:**

GitHub link:

<https://github.com/Lavanya-Sripathi/EduTutorAI>

Demo video link:

<https://drive.google.com/file/d/1uRPM64xr8anuqj13qsYrjYKMUWteauO/view?usp=drivesdk>