

EduTutor Ai Personalized Learning With Generative-Ai

ProjectDescription:

EduTutor AI is a next-generation personalized language learning platform focused exclusively on helping users master the English language. The system leverages IBM Watson Machine Learning and the powerful Granite-3.3-2b-instruct Generative AI model to deliver intelligent, real-time assistance. One of its core features is the real-time correction engine, which analyzes grammar, spelling, and punctuation mistakes as the user types or speaks. These corrections are accompanied by detailed explanatory notes that break down linguistic rules, allowing learners to understand and retain concepts better.

To ensure personalized growth, EduTutor AI includes an adaptive quiz system that generates multiple-choice questions tailored to the learner's current proficiency level. This dynamic quiz mechanism evolves with the user's progress, promoting continuous improvement. Additionally, the platform offers interactive learning modules that provide hands-on English practice through contextual exercises and real-life examples. Designed using Gradio for seamless user interaction, EduTutor AI offers a clean, accessible, and user-friendly experience across devices.

The integration of IBM Watson ensures robust model handling and secure data processing, making the platform both efficient and trustworthy. EduTutor AI is suitable for a broad audience including students, professionals, and language enthusiasts. It focuses on enhancing reading, writing, and comprehension skills while providing instant feedback. This AI-powered solution transforms the way English is learned, combining cutting-edge technology with educational innovation to create a smarter, faster, and more engaging language learning journey.

Scenario 1: Real-Time Writing Corrections

A learner submits a paragraph on language learning. EduTutor AI analyzes grammar, spelling, and punctuation errors in real time, offers corrections with rule explanations, and tracks word count for optimal engagement.

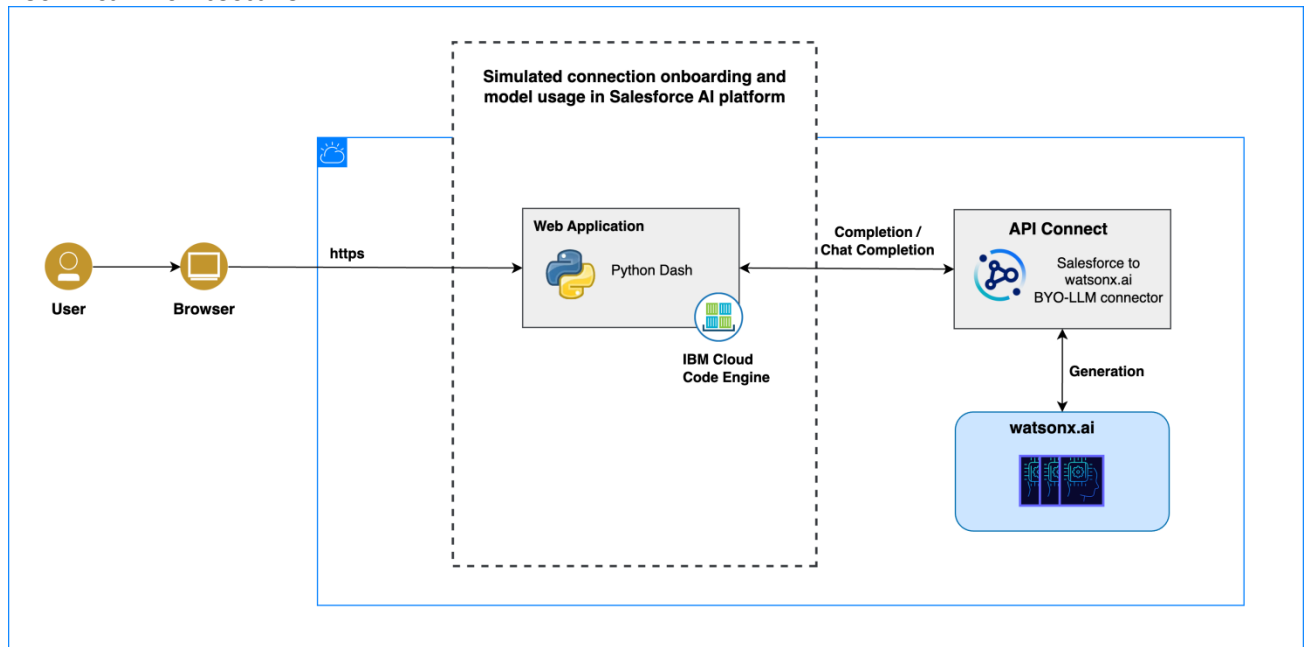
Scenario 2: Adaptive Grammar Quiz

Based on performance, EduTutor AI generates personalized multiple-choice grammar questions covering tenses, articles, and sentence structure, with instant feedback and answer explanations.

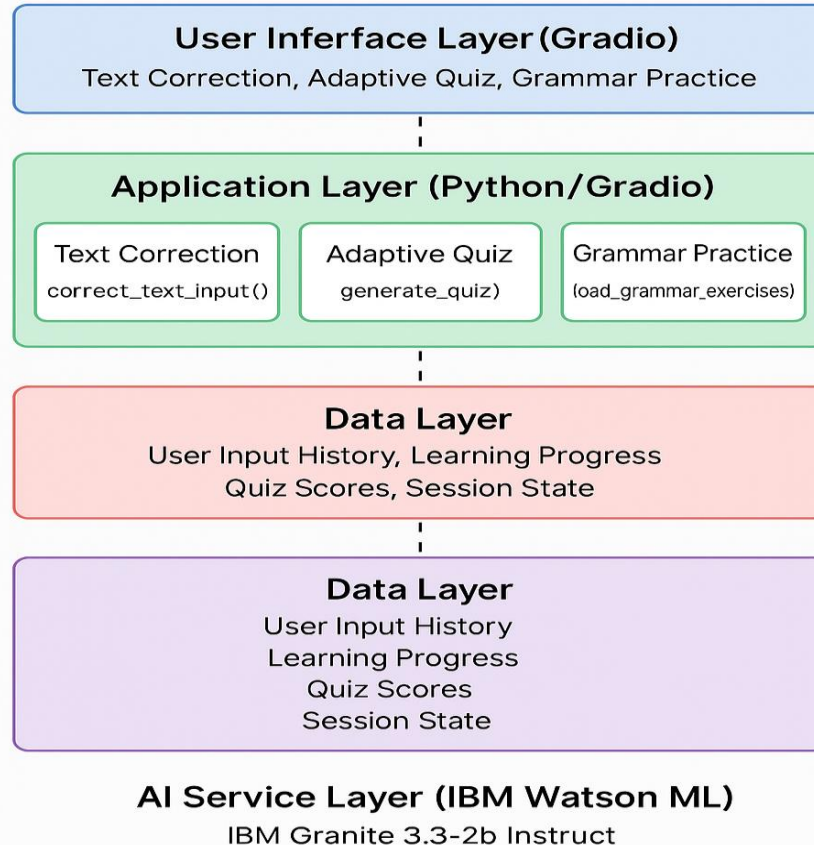
Scenario 3: Interactive Grammar Practice

An intermediate learner selects Grammar Practice. The platform delivers structured exercises—fill-in-the-blanks, rule-based examples, and real-time validation—targeting core grammar concepts with full answer keys.

TechnicalArchitecture:



EduTutor AI – Architecture Diagram



Development Activities (Mapped to Architecture)

Activity 1: Model Selection and Architecture (AI Service Layer + Application Layer)

- **Activity 1.1:** Set up the development environment, install libraries (Gradio, Transformers, IBM Granite), and configure model integration in the AI Service Layer.

Activity 2: Core Functionalities Development (Application Layer)

- **Activity 2.1:** Develop core logic for `correct_text_input()`, `generate_quiz()`, and `load_grammar_exercises()` functions to support real-time corrections, notes, quizzes, and grammar exercises.
- **Activity 2.2:** Implement language detection and analysis metrics for feedback logic, tied closely to the Data Layer and application logic.

Activity 3: `app.py` Development (Application Layer + AI Service Layer)

- **Activity 3.1:** Define functions in `app.py` to handle logic for user inputs, AI responses, and calling IBM Granite model endpoints.
- **Activity 3.2:** Develop prompt engineering strategies for the Granite model to ensure quality, educational output from the AI Service Layer.

Activity 4: Frontend Development (User Interface Layer)

- **Activity 4.1:** Design a responsive tab-based UI using Gradio (Text Correction, Quiz, Grammar Practice).
- **Activity 4.2:** Use Matplotlib or Plotly to visualize user learning progress and text analysis insights in the dashboard.

Activity 5: Deployment (Full Stack Integration)

- **Activity 5.1:** Optimize memory usage and model loading for the IBM Granite model in the backend.
- **Activity 5.2:** Deploy the entire application on a suitable cloud/hosting platform, ensuring the UI connects smoothly to backend AI services.

Milestone 1: Model Selection and Architecture

In this milestone, the focus is on selecting and integrating the **IBM Granite-3.3-2b-instruct** model to support AI-powered English language learning. This involves configuring the model for optimal educational content generation and laying the groundwork for advanced language processing capabilities.

Activity 1.1: Set Up the Development Environment

To begin development, follow this procedure to properly configure your environment:

Step 1: Install Python and pip

Ensure that Python (version 3.8 or higher) and pip are installed.

```
bash
CopyEdit
python --version
pip --version
```

If not installed, download and install from:

<https://www.python.org/downloads/>

Step 2: Create a Virtual Environment

Set up a virtual environment to manage and isolate project dependencies:

```
bash
CopyEdit
python -m venv edututor_env
```

Activate the environment:

- **Windows:**

```
bash
CopyEdit
edututor_env\Scripts\activate
```

- **macOS/Linux:**

```
bash
CopyEdit
source edututor_env/bin/activate
```

Step 3: Install Required Libraries

Install all necessary Python libraries using pip:

```
bash
CopyEdit
pip install gradio transformers torch matplotlib langid accelerate flask
```

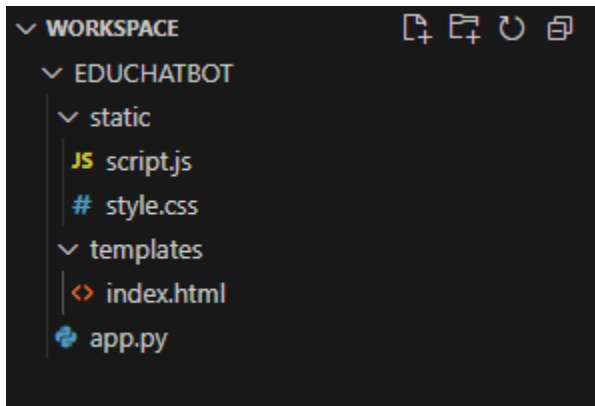
These libraries support:

- Gradio – for frontend interface
- Transformers – to load and use IBM Granite or Hugging Face models
- Torch – for model computation
- Matplotlib – for data visualization
- LangID – for language detection
- Accelerate – for model optimization and deployment
- Flask – for backend application handling

Step 4: Set Up Application Structure

Create the initial project directory structure:

```
csharp
CopyEdit
EduTutor_Chatbot_FINAL/
├── app.py                # Main backend logic
├── templates/
│   └── index.html        # Frontend layout
├── static/
│   ├── style.css         # Styling
│   └── script.js         # Frontend interactivity\
```



This structure aligns with the overall architecture:

- `app.py` will integrate the Granite model and core logic
- `templates/index.html` and the `static/` folder support the Gradio-based user interface

Milestone2:CoreFunctionalitiesDevelopment

Activity2.1:Developcorefunctionalities

1. Real-TimeCorrectionsSystem:

- Implementtextanalysisinterfaceforgrammar,spelling,andpunctuation
- CreatepromptingsystemfortheIBMGranitemodeltoprovidedetailedcorrections
- Developwordcountvalidationandfeedbackmechanisms

2. ExplanatoryNotesSystem:

- Createautomaticlanguagedetectionusinglangidlibrary
- Developcontextuallanguagerulebreakdownfunctionality
- Structureoutputformattoshowdetailedlinguisticanalysis

3. AdaptiveQuizGenerator:

- Bulddynamicquizgenerationformultiplelanguages
- Createthreequizcategories:Grammar,Tenses,andPartsofSpeech
- Implementstructuredquestionformattingwithanswerkeys

4. MultilingualLearningExercises:

- Implementinteractiveexercisegenerationacrosssupportedlanguages
- Createthreeexercisetypes:GrammarExercises,SentenceFormation,andTense Exercises
- Developcomprehensivepracticeactivitieswithexplanations

Activity2.2:Implementdatamanagementutilities

1. LanguageDetectionIntegration:

- Integratelangidlibraryforautomaticlanguageidentification
- SupportforEnglish,Spanish,Chinese,French,German,andHindi
- Implementconfidencescoringfordetectionaccuracy

2. TextAnalysisMetrics:

- Calculatewordcount,averagewordlength,andsentencecomplexity
- Generatelanguagecompetencyvisualizationsusingmatplotlib
- Create radar charts for comprehensive skill assessment

3. ModelResponseProcessing:

- Implementintelligentresponseparsingfordifferentcontenttypes
- Structureoutputsforeducationalclarityandorganization
- Handlemodel-generatedcontentformattingandpresentation

Milestone 3: App.py Development

This milestone involves writing the core backend logic of the EduTutor AI application within the `app.py` file. It defines the structure, model interaction, functional logic, and user interface integration using Gradio.

Activity 3.1: Write the Main Application Logic

The `app.py` file is structured into four primary components as described below:

1. Imports and Setup

- Import essential libraries:
 - Gradio – for UI building
 - Transformers, Torch, Accelerate – for AI model handling
 - Matplotlib, LangID – for visualization and language detection
- Load the **IBM Granite-3.3-2b-instruct model** with optimized parameters
- Initialize the tokenizer and model, including device mapping (`cuda/cpu`) for efficient inference

2. Core Functions

- `generate_response(query):`
Generates AI-based answers using IBM Granite for grammar/vocabulary-related questions
- `detect_language(text):`
Detects the language of the input using `langid`, alerts users if input is not in English
- `real_time_correction(text):`
Analyzes user input for grammar, punctuation, and spelling mistakes and suggests corrections
- `explanatory_notes(text):`
Provides explanations for each correction and highlights grammatical rules in a user-friendly way
- `generate_quiz(topic, level):`
Creates customized multiple-choice quizzes based on grammar topics and difficulty level
- `multilingual_learning(level, focus_area):`
Generates structured exercises for learners based on level and language focus
- `analyze_text(text):`
Performs text complexity analysis and visualizes readability, grammar issues, and vocabulary diversity using Matplotlib

3. UI Components

- Create a tabbed interface using **Gradio Blocks**:
 - Tabs for **Real-Time Correction**, **Explanations**, **Adaptive Quiz**, **Learning Plan**, **Text Analysis**
- Each tab contains relevant inputs and outputs, styled for a clean learning experience
- Input validation and custom error handling included for robust usability
- Matplotlib visualizations (e.g., bar graphs, pie charts) integrated in the Text Analysis tab

4. Feature Implementation

Instead of HealthAI features, these EduTutor AI-specific features are implemented:

- `display_query_helper()`:
Chat-style interface for grammar or vocabulary questions powered by IBM Granite
- `display_quiz_generator()`:
Interface to choose topics and difficulty, and take an adaptive quiz
- `display_correction_tool()`:
Input box to receive real-time grammar corrections and rule explanations
- `display_learning_dashboard()`:
Dashboard to visualize language progress, error types, complexity scores, and recommendations

Activity3.2:Create prompting strategies

Real-Time Corrections Prompting:

```
def create_grammar_plan(level, focus_area):
    """Create a grammar improvement plan for a learner"""
    model = init_granite_model()

    prompt = f"""
You are an AI English tutor. Create a weekly grammar improvement plan for a {level} learner focused on: {focus_area}.

Include:
- Daily topics
- Practice suggestions
- Common mistakes
- Tips and motivation

Format it as a clear weekly plan.

RESPONSE:
"""
    return model.generate_text(prompt=prompt)
```

Language Analysis Prompting:

```
def answer_learning_query(query):
    """Use IBM Granite to answer English language learning questions"""
    model = init_granite_model()

    prompt = f"""
You are an English language tutor AI. Provide a helpful, clear, and student-friendly answer to the following question:

QUESTION: {query}

Your response should:
- Explain the concept simply
- Include examples if applicable
- Avoid technical jargon
- Encourage learning with positivity

RESPONSE:
"""
    return model.generate_text(prompt=prompt)
```


Quiz Generation Prompting:

```
def generate_quiz(topic, level):  
    """Generate multiple-choice grammar quiz based on topic and difficulty"""  
    model = init_granite_model()  
  
    prompt = f"""  
Generate a 5-question English grammar quiz on the topic: {topic}.  
Difficulty level: {level}  
  
Each question should have:  
- One correct answer  
- Three distractors  
- Explanation of the correct answer  
  
Format:  
Q1. ...  
A. ...  
B. ...  
C. ...  
D. ...  
Answer: ...  
Explanation: ...  
"""  
    return model.generate_text(prompt=prompt)
```

Milestone 4: Frontend Development

This milestone involves building the interactive and user-friendly frontend for EduTutor AI using Gradio. The UI facilitates seamless interaction with the backend functionalities such as real-time corrections, adaptive quizzes, rule explanations, and performance analytics.

Activity 4.1: Design and Develop the User Interface

1. Main Application Layout

- Build the UI using `Gradio Blocks` with customized "**EduTutor AI**" branding (logo/title).
- Structure the layout using **Tabbed Navigation** with four main feature tabs:
 - Real-Time Correction
 - Explanatory Notes
 - Adaptive Quiz
 - Multilingual Learning
- Implement intuitive input forms with:
 - Responsive layouts
 - Field validations (e.g., non-empty, character limits)
 - Feedback for invalid or missing input

2. Feature-Specific Interfaces

- **Real-Time Corrections Tab**
 - Input for selecting a grammar topic
 - Paragraph textarea for user input
 - Output sections categorized into:
 - Grammar corrections
 - Punctuation suggestions
 - Spelling fixes
- **Explanatory Notes Tab**
 - Text input with **automatic language detection**
 - Output of rule-based explanations for grammar errors
 - Color-coded highlight of text issues
- **Adaptive Quiz Tab**
 - Language dropdown (currently English only, expandable)
 - Difficulty level selector
 - Dynamic quiz rendering with MCQs
 - Immediate feedback for each question
- **Multilingual Learning Tab**
 - Language selection dropdown
 - Exercise type selector (fill-in-the-blanks, error spotting, sentence correction)
 - Generated structured learning content with instructions and answer keys

Activity 4.2: Create Dynamic Visualizations

1. Language Competency Charts

- Display visual insights using **Matplotlib** and `base64` encoding to embed images in Gradio
- Bar Chart for:
 - Word Count

- Sentence Count
 - Average Word Length
- Radar Chart representing language competency levels in:
 - Grammar
 - Vocabulary
 - Structure
 - Coherence
 - Style

2. Analysis Metrics

- **Real-Time Word Count** and live feedback:
 - Highlights if input is too short or too long
 - Suggests optimal range (e.g., 100–200 words)
- **Language Detection Confidence:**
 - Displays detected language and confidence score using `langid`
- **Interactive Visual Feedback:**
 - Visualization elements update in real-time based on text input
 - Improves engagement and understanding of user progress

Milestone 5: Deployment

This milestone covers the final steps to prepare, test, and deploy the EduTutor AI application. It ensures the model is optimized, the application runs efficiently, and the platform is made accessible to users either locally or via cloud platforms like Hugging Face Spaces.

Activity 5.1: Prepare for Deployment

1. Model Configuration

- **Granite Model Loading:**
 - Configure loading of IBM Granite-3.3-2b-instruct with device mapping using `torch.device('cuda' if torch.cuda.is_available() else 'cpu')`
 - This ensures compatibility across machines with and without GPU
- **Memory Optimization:**
 - Use `torch.no_grad()` for inference-only operations to save memory
 - Avoid reloading the model on every API call
- **Transformers Integration:**
 - Integrate Hugging Face `transformers` library correctly with the model and tokenizer
 - Use `AutoTokenizer` and `AutoModelForCausalLM` or `AutoModelForSeq2SeqLM` as appropriate

2. Dependency Management

- **Create a `requirements.txt` File** containing all necessary packages:

```
txt
CopyEdit
gradio
transformers
torch
matplotlib
numpy
langid
accelerate
flask
```

- This file ensures consistent environment setup during local testing or cloud deployment

Activity 5.2: Deploy the Application

1. Local Deployment Testing

- Run the app locally using:

```
bash
CopyEdit
python app.py
```

- Test each of the four core EduTutor AI features:
 - Real-Time Grammar Corrections
 - Explanatory Notes
 - Adaptive Quiz Generator

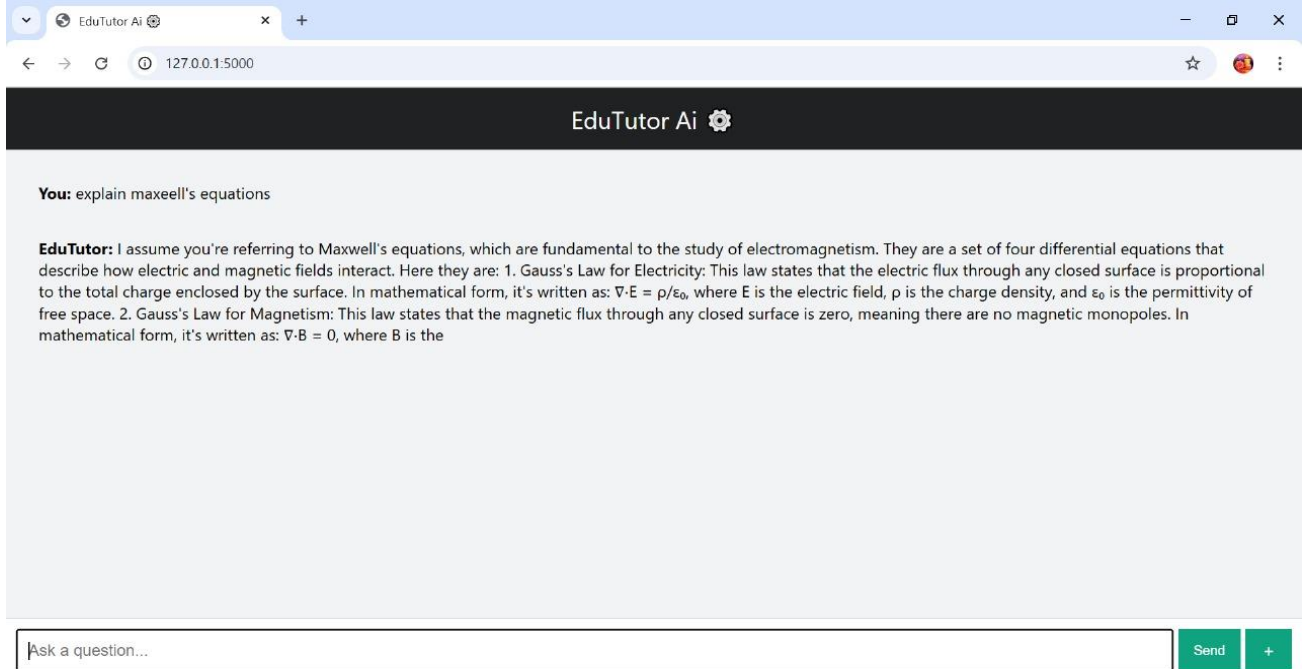
- Multilingual Grammar Exercises
- Confirm model loads correctly and responses are generated for:
 - User questions
 - Text correction
 - Quiz logic
 - Rule explanations

2. Cloud Deployment Options

- **Deploy on Hugging Face Spaces:**
 - Push the application (app.py, requirements.txt, and Gradio interface) to a GitHub repo
 - Connect it to Hugging Face under **Gradio SDK**
- **GPU Configuration:**
 - Enable GPU support for optimal model inference (recommended for Granite model)
- **Monitoring:**
 - Use Hugging Face's built-in logs to:
 - Track model usage
 - Measure response time
 - Monitor crash reports or deployment failures

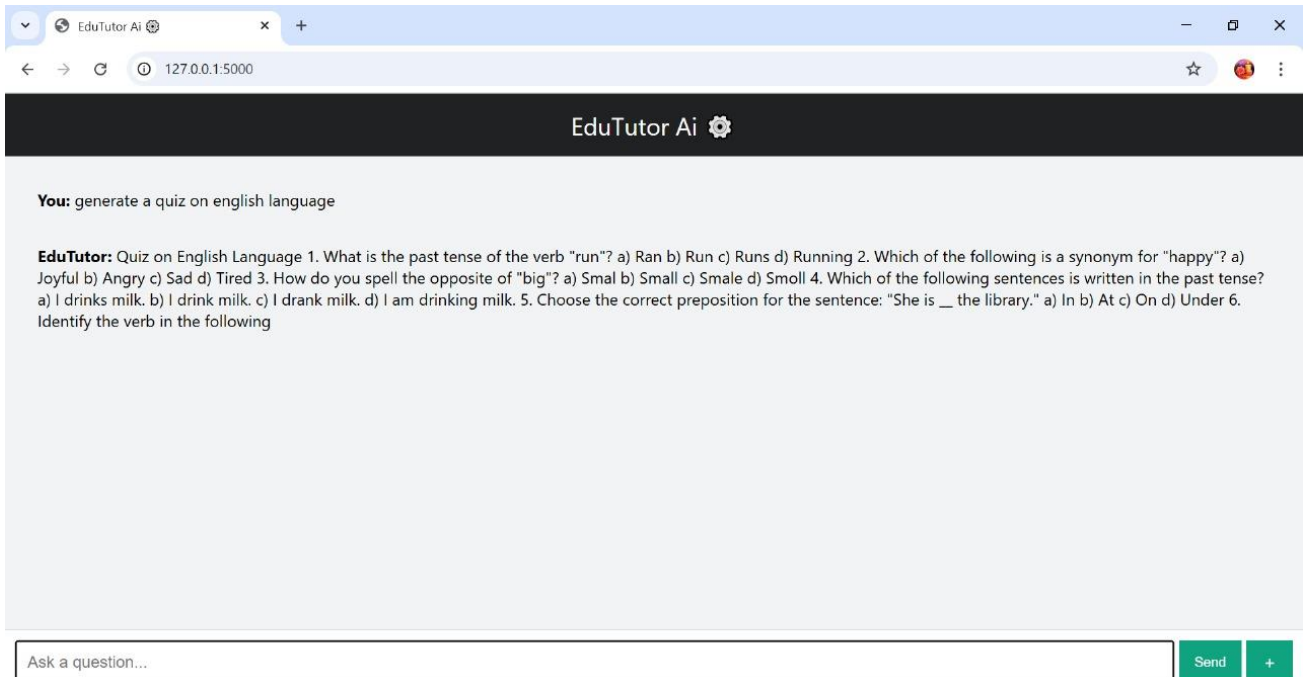
Exploring Application Features:.

Explanatory Notes Output:



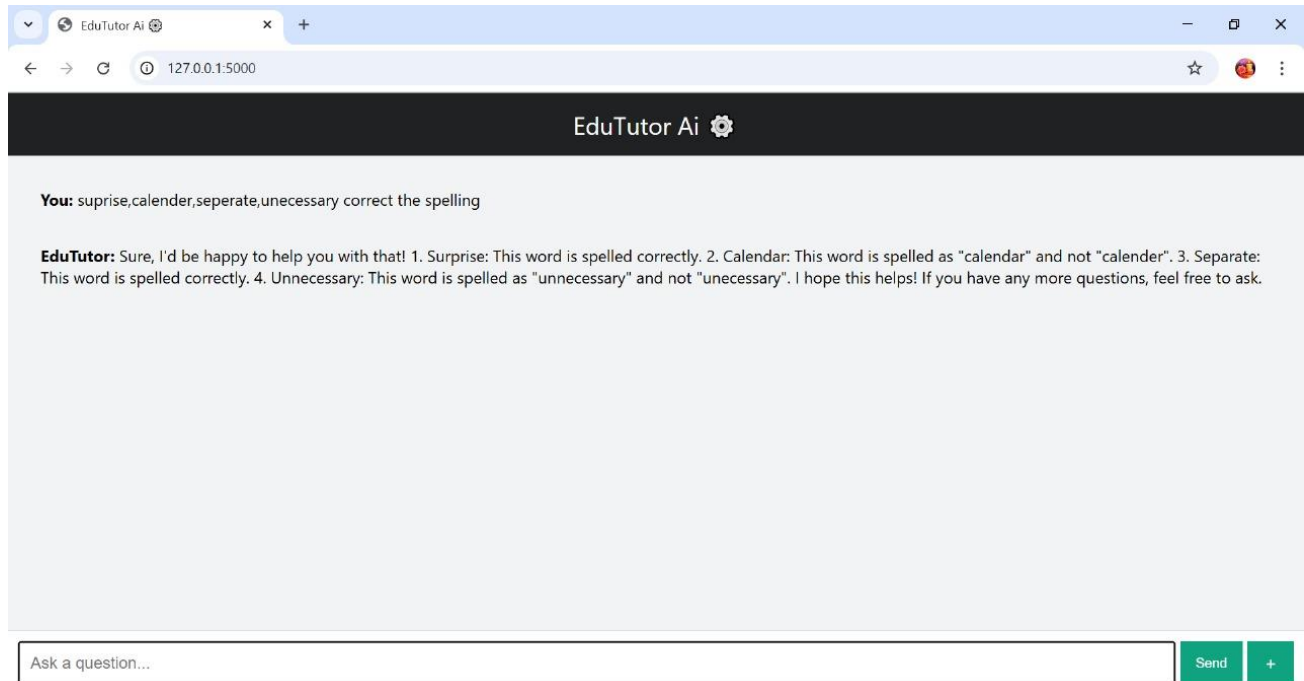
The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "EduTutor Ai". The main content area displays a user prompt: "You: explain maxeell's equations". Below this, the AI response reads: "EduTutor: I assume you're referring to Maxwell's equations, which are fundamental to the study of electromagnetism. They are a set of four differential equations that describe how electric and magnetic fields interact. Here they are: 1. Gauss's Law for Electricity: This law states that the electric flux through any closed surface is proportional to the total charge enclosed by the surface. In mathematical form, it's written as: $\nabla \cdot \mathbf{E} = \rho / \epsilon_0$, where \mathbf{E} is the electric field, ρ is the charge density, and ϵ_0 is the permittivity of free space. 2. Gauss's Law for Magnetism: This law states that the magnetic flux through any closed surface is zero, meaning there are no magnetic monopoles. In mathematical form, it's written as: $\nabla \cdot \mathbf{B} = 0$, where \mathbf{B} is the magnetic field." At the bottom, there is a text input field with the placeholder "Ask a question..." and two buttons labeled "Send" and "+".

Adaptive Quiz Page:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "EduTutor Ai". The main content area displays a user prompt: "You: generate a quiz on english language". Below this, the AI response reads: "EduTutor: Quiz on English Language 1. What is the past tense of the verb 'run'? a) Ran b) Run c) Runs d) Running 2. Which of the following is a synonym for 'happy'? a) Joyful b) Angry c) Sad d) Tired 3. How do you spell the opposite of 'big'? a) Smal b) Small c) Smale d) Smoll 4. Which of the following sentences is written in the past tense? a) I drinks milk. b) I drink milk. c) I drank milk. d) I am drinking milk. 5. Choose the correct preposition for the sentence: 'She is _ the library.' a) In b) At c) On d) Under 6. Identify the verb in the following". At the bottom, there is a text input field with the placeholder "Ask a question..." and two buttons labeled "Send" and "+".

Real-Time Corrections output:



Conclusion

The EduTutor AI application leverages the powerful capabilities of **IBM Watson Machine Learning**, specifically the **Granite-3.3-2b-instruct** model, to deliver accurate language analysis, real-time error correction, intelligent quiz generation, and structured grammar exercises for English learners. The project was developed through a structured and phased approach covering model integration, core feature implementation, frontend development using **Gradio**, and dynamic visualization enhancements.

By utilizing the **Gradio framework**, EduTutor AI offers a seamless and interactive user experience, enabling learners to engage with AI-powered educational tools intuitively. Features such as real-time grammar correction, adaptive quizzes, and explanatory rule-based feedback ensure personalized and impactful language learning.

The platform also provides dynamic visualizations that track language progress through text analysis metrics and competency charts, improving learner engagement and self-assessment. This project demonstrates how a carefully selected AI model, combined with an educationally structured interface, can significantly improve the **accessibility, interactivity, and effectiveness** of language learning tools for diverse users.