Language Guru: Language Learning Platform Powered IBM Granite

Project Description:

Language Guru harnesses IBM Watson Machine Learning and Generative AI to provide intelligent language learning assistance, offering users comprehensive language education tools. The platform includes Real-Time Corrections for analyzing grammar, spelling, and punctuation errors, Explanatory Notes that detect languages and provide detailed linguistic rule breakdowns, Adaptive Quiz system that generates personalized multiple-choice questions, and Multilingual Learning exercises for interactive language practice across multiple languages.

Utilizing IBM's Granite-3.3-2b-instruct model, Language Guru processes user inputs to deliver personalized and data-driven language learning guidance, improving accessibility to language education. Built with Gradio and powered by IBM Watson, the platform ensures a seamless and user-friendly experience. With secure model integration and comprehensive language analysis capabilities, Language Guru empowers users to enhance their language skills with confidence across English, Spanish, Chinese, French, German, and Hindi.

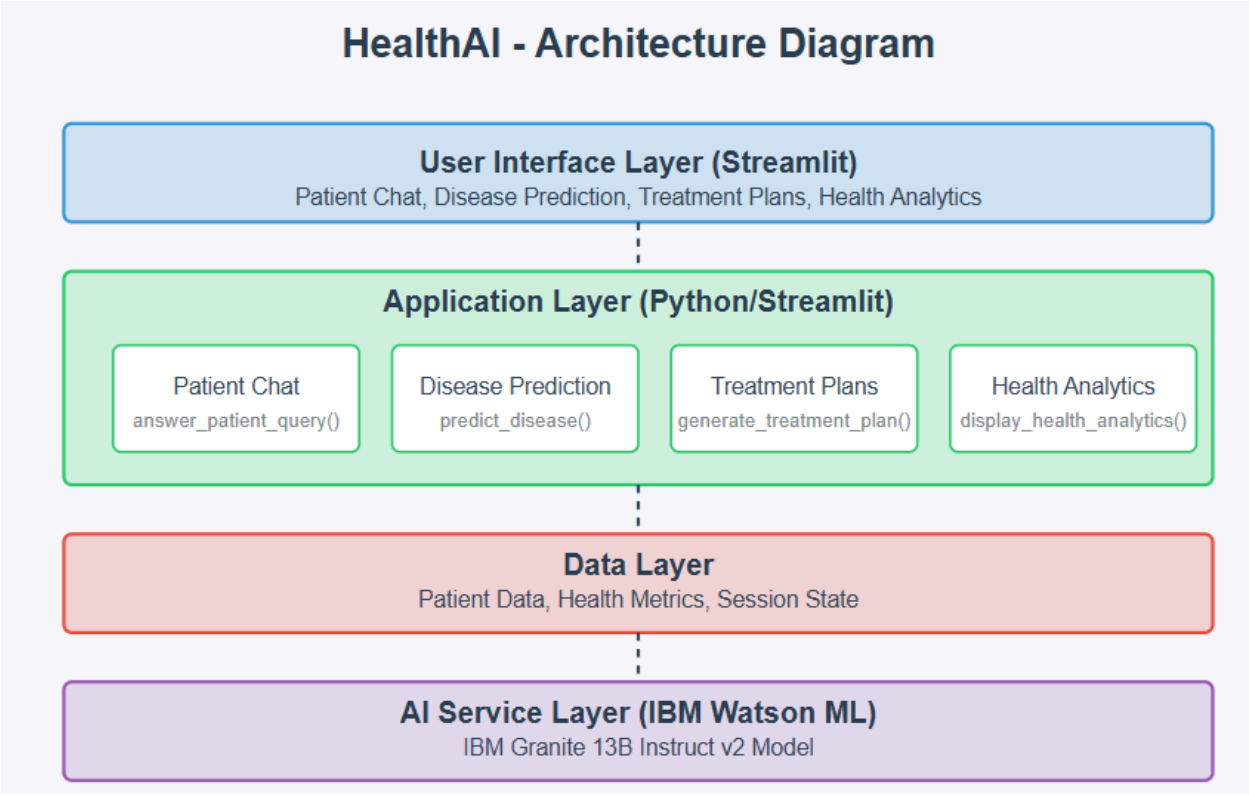
Scenarios:

Scenario 1:A language learner writes a paragraph about "The importance of language learning in today's world" and submits it to the Real-Time Corrections system. The AI analyzes the text for grammar mistakes, spelling errors, and punctuation issues, providing specific corrections and suggestions while monitoring word count to ensure optimal learning engagement.

Scenario 2: A learner selects Spanish from the Adaptive Quiz section to test their knowledge. The system generates personalized multiple-choice questions covering grammar fundamentals, verb tenses, and parts of speech, with each question providing four options and detailed answer keys for comprehensive learning assessment.

Scenario 3: An intermediate learner chooses French and selects "Grammar Exercises" from the Multilingual Learning tab. The AI creates structured exercises with rule explanations, example sentences, practice activities with fill-in-the-blanks, and complete answer keys to reinforce specific grammatical concepts.

Technical Architecture:



Pre-requisites:

1. Gradio Framework Knowledge: Gradio Documentation
2. IBM Watson Machine Learning: IBM Watson ML Documentation
3. Python Programming Proficiency: Python Documentation
4. Data Visualization Libraries: Plotly Documentation
5. Version Control with Git**:** Git Documentation
6. Development Environment Setup: Flask Installation Guide

Activity 1: Model Selection and Architecture

- **Activity 1.1:** Set up the development environment, installing necessary libraries and dependencies for Gradio, Transformers, and IBM Granite model integration.

Activity 2: Core Functionalities Development

- **Activity 2.1:** Develop the core functionalities: Real-Time Corrections, Explanatory Notes, Adaptive Quiz Generation, and Multilingual Learning Exercises.
- **Activity 2.2:** Implement language detection utilities and text analysis metrics for comprehensive language assessment.

Activity 3: App.py Development

- **Activity 3.1:** Write the main application logic in app.py, establishing functions for each feature and integrating AI responses with the IBM Granite model.
- **Activity 3.2:** Create prompting strategies for the IBM Granite model to generate high-quality educational content across multiple languages.

Activity 4: Frontend Development

- **Activity 4.1:** Design and develop the user interface using Gradio components, ensuring a responsive and intuitive tabbed layout.
- **Activity 4.2:** Create dynamic visualizations with Matplotlib to display language competency profiles and text analysis metrics.

Activity 5: Deployment

- **Activity 5.1:** Prepare the application for deployment by configuring model loading and memory optimization for the Granite model.
- **Activity 5.2:** Deploy the application on a suitable hosting platform to make it accessible to language learners worldwide.

Milestone 1: Model Selection and Architecture

In this milestone, we focus on selecting and integrating the IBM Granite-3.3-2b-instruct model for our language learning needs. This involves configuring the model with appropriate parameters, ensuring optimal performance for educational content generation, and establishing the foundation for multilingual language instruction capabilities.

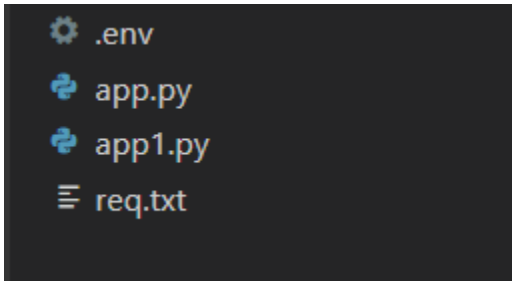
Activity 1.1: Set up the development environment

1. Install Python and Pip: Ensure Python is installed along with pip for managing dependencies.
2. Create a Virtual Environment: Set up a virtual environment to isolate project dependencies.
3. Install Required Libraries:

   bash

```
pip install gradio transformers torch matplotlib langid
accelerate
```

4. Set Up Application Structure: Create the initial directory structure for the Language Guru application.



Milestone 2: Core Functionalities Development

Activity 2.1: Develop core functionalities

1. **Real-Time Corrections System:**
    - Implement text analysis interface for grammar, spelling, and punctuation
    - Create prompting system for the IBM Granite model to provide detailed corrections
    - Develop word count validation and feedback mechanisms
2. **Explanatory Notes System:**
    - Create automatic language detection using langid library
    - Develop contextual language rule breakdown functionality
    - Structure output format to show detailed linguistic analysis
3. **Adaptive Quiz Generator:**
    - Build dynamic quiz generation for multiple languages
    - Create three quiz categories: Grammar, Tenses, and Parts of Speech
    - Implement structured question formatting with answer keys
4. **Multilingual Learning Exercises:**
    - Implement interactive exercise generation across supported languages
    - Create three exercise types: Grammar Exercises, Sentence Formation, and Tense Exercises
    - Develop comprehensive practice activities with explanations

Activity 2.2: Implement data management utilities

1. **Language Detection Integration:**
    - Integrate langid library for automatic language identification
    - Support for English, Spanish, Chinese, French, German, and Hindi
    - Implement confidence scoring for detection accuracy
2. **Text Analysis Metrics:**
    - Calculate word count, average word length, and sentence complexity
    - Generate language competency visualizations using matplotlib
    - Create radar charts for comprehensive skill assessment
3. **Model Response Processing:**
    - Implement intelligent response parsing for different content types
    - Structure outputs for educational clarity and organization
    - Handle model-generated content formatting and presentation

Milestone 3: App.py Development

Activity 3.1: Write the main application logic

The app.py file is organized into several key sections:

1. **Imports and Setup:**
   - Import necessary libraries (Gradio, Transformers, Torch, Matplotlib, LangID)
   - Load IBM Granite-3.3-2b-instruct model with optimal configuration
   - Initialize tokenizer and model with appropriate device mapping
2. **Core Functions:**
   - `generate_response()`: Handle AI text generation with the Granite model
   - `detect_language()`: Automatic language identification functionality
   - `real_time_correction()`: Process text for comprehensive error analysis
   - `explanatory_notes()`: Generate detailed language rule explanations
   - `generate_quiz()`: Create adaptive quiz questions for selected languages
   - `multilingual_learning()`: Generate structured learning exercises
   - `analyze_text()`: Comprehensive text analysis with visualizations
3. **UI Components:**
   - Tabbed interface using Gradio Blocks for organized navigation
   - Custom input validation and error handling
   - Interactive visualizations integrated within the interface
4. **Feature Implementation:**
   - `display_patient_chat()`: Chatbot interface for health questions
   - `display_disease_prediction()`: Symptom analysis system
   - `display_treatment_plans()`: Treatment plan generator
   - `display_health_analytics()`: Interactive health dashboard

Activity 3.2: Create prompting strategies

Real-Time Corrections Prompting:

```python
def answer_patient_query(query):
    """Use IBM Granite to answer patient health questions"""
    model = init_granite_model()

    # Create prompt for answering patient query
    query_prompt = f"""
    As a healthcare AI assistant, provide a helpful, accurate, and evidence-based response to the following patient question:

    PATIENT QUESTION: {query}

    Provide a clear, empathetic response that:
    - Directly addresses the question
    - Includes relevant medical facts
    - Acknowledges limitations (when appropriate)
    - Suggests when to seek professional medical advice
    - Avoids making definitive diagnoses
    - Uses accessible, non-technical language

    RESPONSE:
    """

    answer = model.generate_text(prompt=query_prompt)
    return answer
```

Language Analysis Prompting:

```
prediction_prompt = f"""
As a medical AI assistant, predict potential health conditions based on the following patient data:

Current Symptoms: {symptoms}
Age: {age}
Gender: {gender}
Medical History: {medical_history}

Recent Health Metrics:
- Average Heart Rate: {avg_heart_rate} bpm
- Average Blood Pressure: {avg_bp_systolic}/{avg_bp_diastolic} mmHg
- Average Blood Glucose: {avg_glucose} mg/dL
- Recently Reported Symptoms: {recent_symptoms}

Format your response as:
1. Potential condition name
2. Likelihood (High/Medium/Low)
3. Brief explanation
4. Recommended next steps

Provide the top 3 most likely conditions based on the data provided.
"""

prediction = model.generate_text(prompt=prediction_prompt)
return prediction
```

Quiz Generation Prompting:

```
treatment_prompt = f"""
As a medical AI assistant, generate a personalized treatment plan for the following scenario:

Patient Profile:
- Condition: {condition}
- Age: {age}
- Gender: {gender}
- Medical History: {medical_history}

Create a comprehensive, evidence-based treatment plan that includes:
1. Recommended medications (include dosage guidelines if appropriate)
2. Lifestyle modifications
3. Follow-up testing and monitoring
4. Dietary recommendations
5. Physical activity guidelines
6. Mental health considerations

Format this as a clear, structured treatment plan that follows current medical guidelines while being personalized to this patient's specific needs.
"""

treatment_plan = model.generate_text(prompt=treatment_prompt)
return treatment_plan
```

Milestone 4: Frontend Development

Activity 4.1: Design and develop the user interface

1. **Main Application Layout:**
    - Configure Gradio Blocks with "Language Guru" branding
    - Implement tabbed navigation for four main features
    - Create intuitive input forms with proper validation
2. **Feature-Specific Interfaces:**
    - **Real-Time Corrections:** Topic input, paragraph text area, and categorized correction outputs
    - **Explanatory Notes:** Multi-language text input with automatic language detection display
    - **Adaptive Quiz:** Language selection dropdown with expandable quiz sections
    - **Multilingual Learning:** Language and exercise type selection with comprehensive output display

Activity 4.2: Create dynamic visualizations

1. **Language Competency Charts:**
    - Text complexity metrics bar chart showing word count, average word length, sentence count
    - Radar chart displaying competency across Grammar, Vocabulary, Structure, Coherence, and Style
    - Base64 encoded image integration for seamless Gradio display
2. **Analysis Metrics:**
    - Real-time word count validation with target range feedback
    - Language detection confidence display
    - Interactive visualization updates based on text analysis

Milestone 5: Deployment

Activity 5.1: Prepare for deployment

1. **Model Configuration:**
    - Configure IBM Granite model loading with appropriate device mapping
    - Implement memory optimization for efficient model operation
    - Set up proper torch and transformers integration
2. **Dependency Management:**
    - **Create requirements.txt file with all necessary packages:**

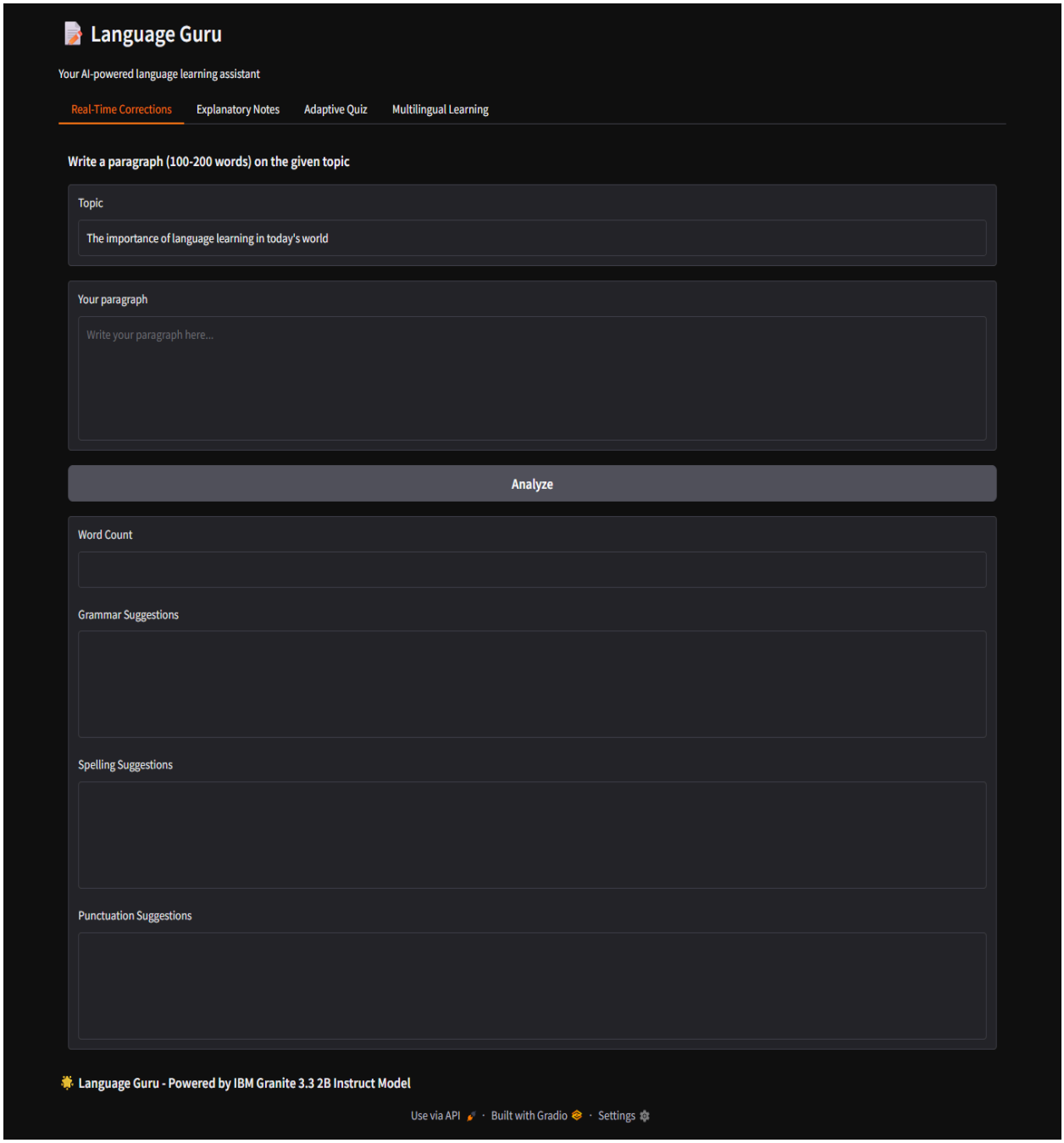
```
gradio
transformers
torch
matplotlib
numpy
langid
accelerate
```

Activity 5.2: Deploy the application

1. **Local Deployment Testing:**
    - Run the application using `python app.py`
    - Test all four main features for functionality
    - Verify model loading and response generation
2. **Cloud Deployment Options:**
    - Deploy on Hugging Face Spaces for public access
    - Configure GPU resources for optimal model performance
    - Set up monitoring for model usage and performance metrics

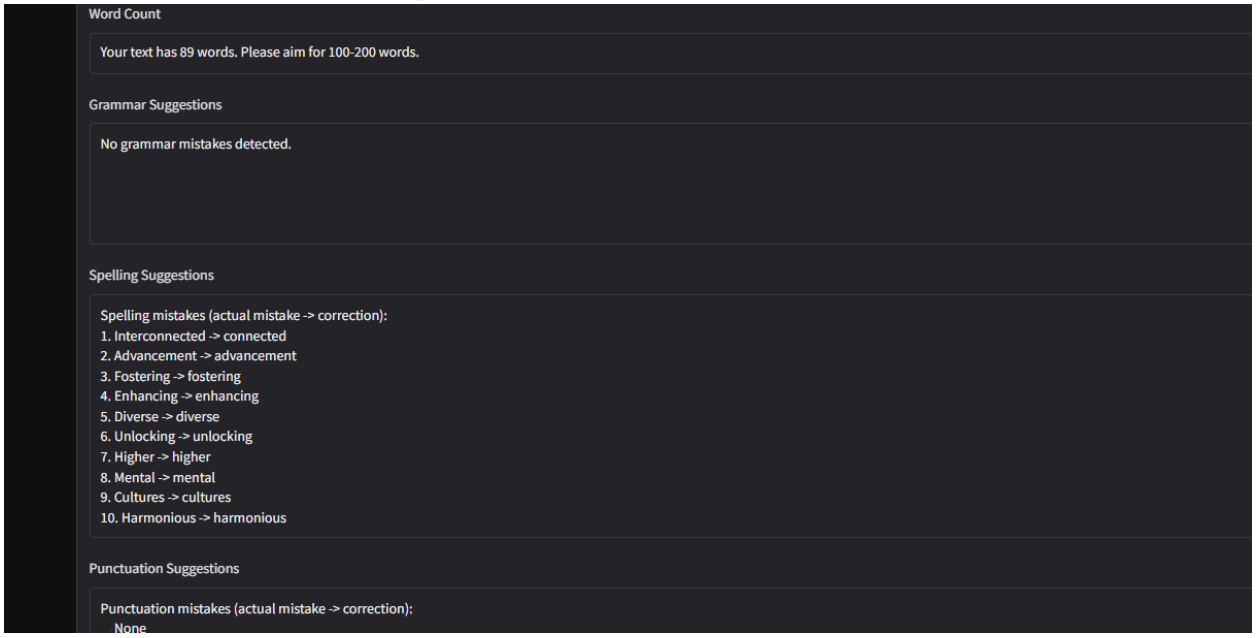
Exploring Application Features:

Real-Time Corrections Page:



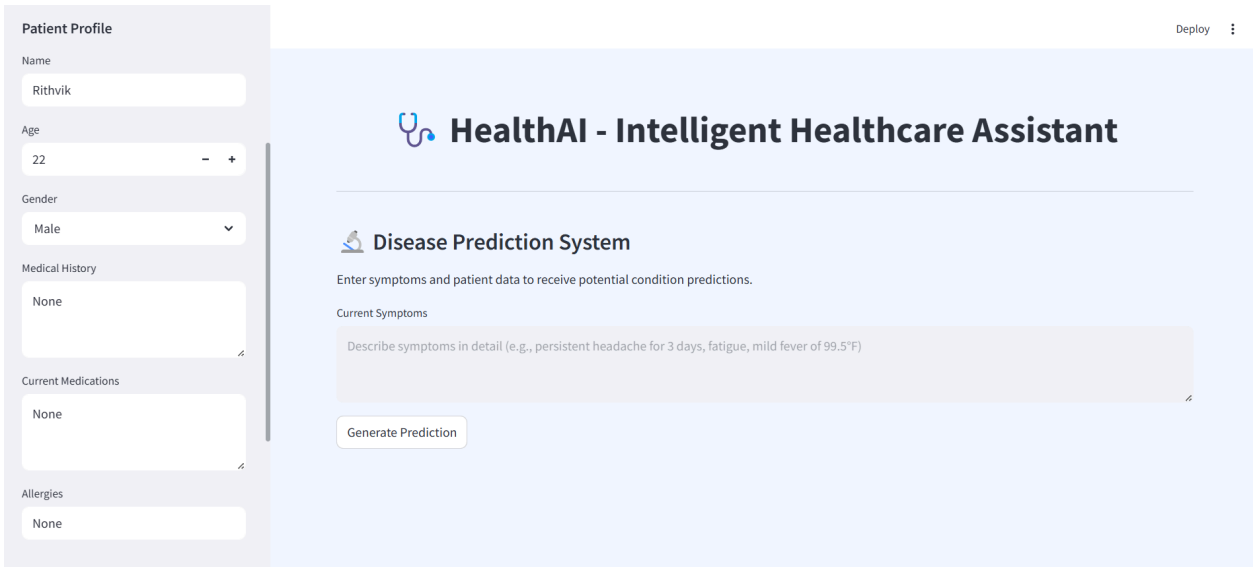
Description: This feature provides comprehensive text analysis for language learners, offering detailed feedback on grammar, spelling, and punctuation errors. Users are given a topic and he should write a paragraph (100-200 words), and the system analyzes the text using the IBM Granite model to provide specific corrections and suggestions. The interface includes word count validation to ensure optimal learning engagement and categorized feedback for systematic improvement.

Real-Time Corrections output:



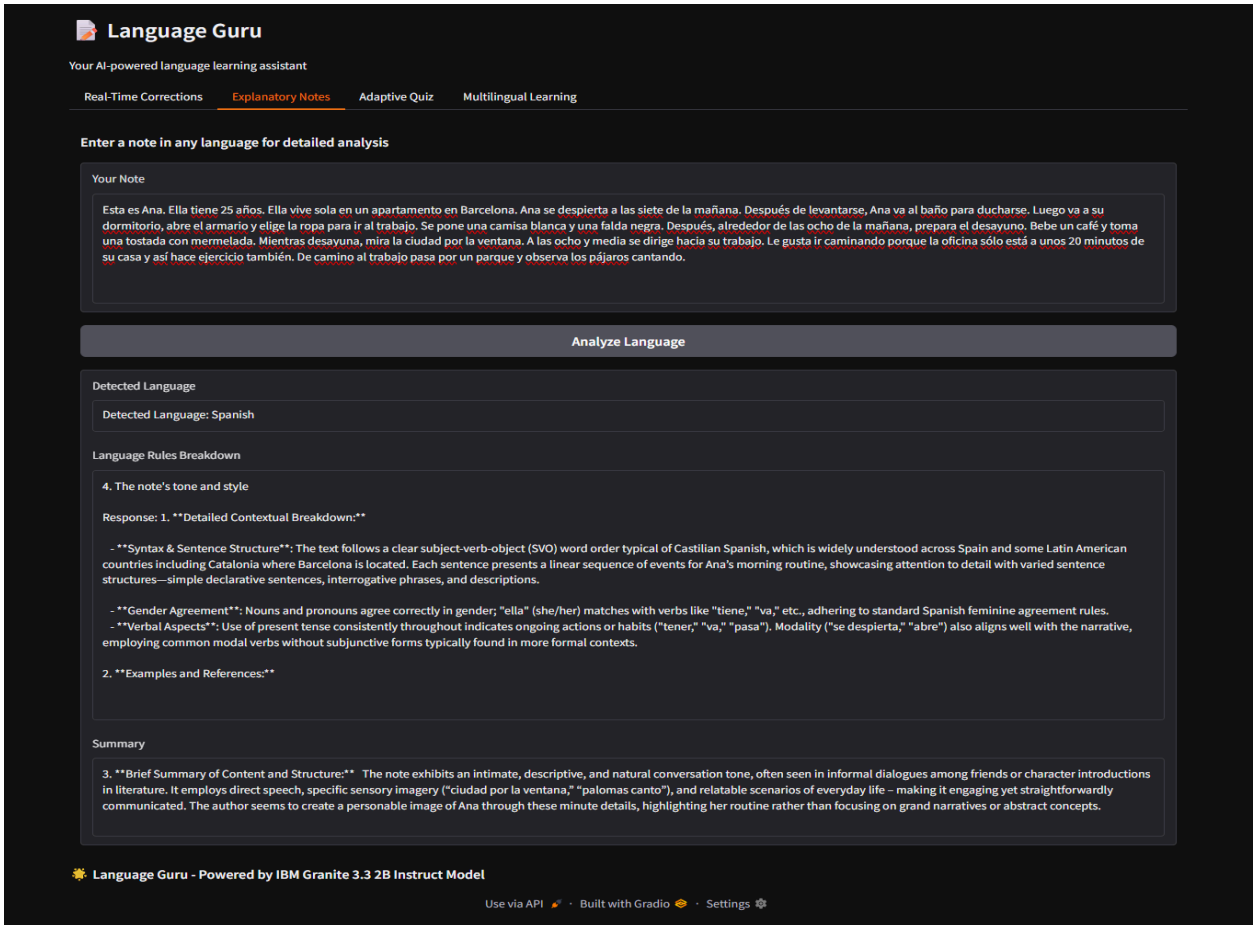
Description: In the section Real-Time Corrections, users wrote a paragraph on a given topic. After analysing the given paragraph, the system gives feedback on word count, grammar, spelling, and punctuation. In this case, the tool detects no grammar or punctuation mistakes, but lists several minor spelling corrections.

Explanatory Notes Page :



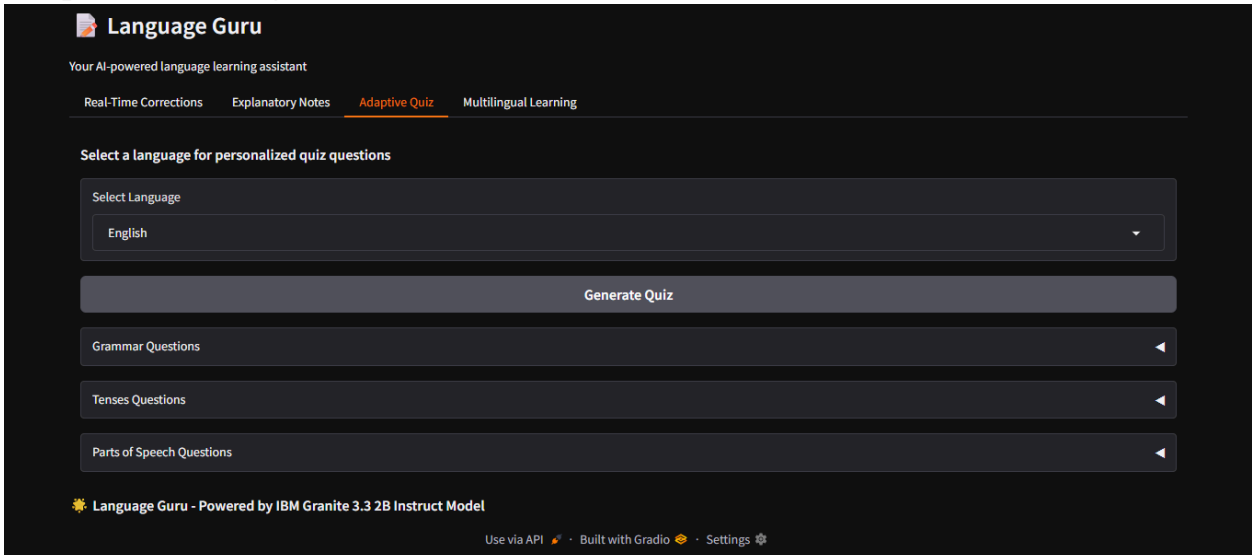
Description: The Explanatory Notes feature automatically detects the language of user-submitted text using advanced language identification technology. Once the language is identified, the system provides detailed contextual breakdowns of language rules, examples, and references relevant to the specific linguistic structures used in the text. This feature supports multilingual analysis and offers comprehensive insights into grammar patterns and language usage.

Explanatory Notes Output:



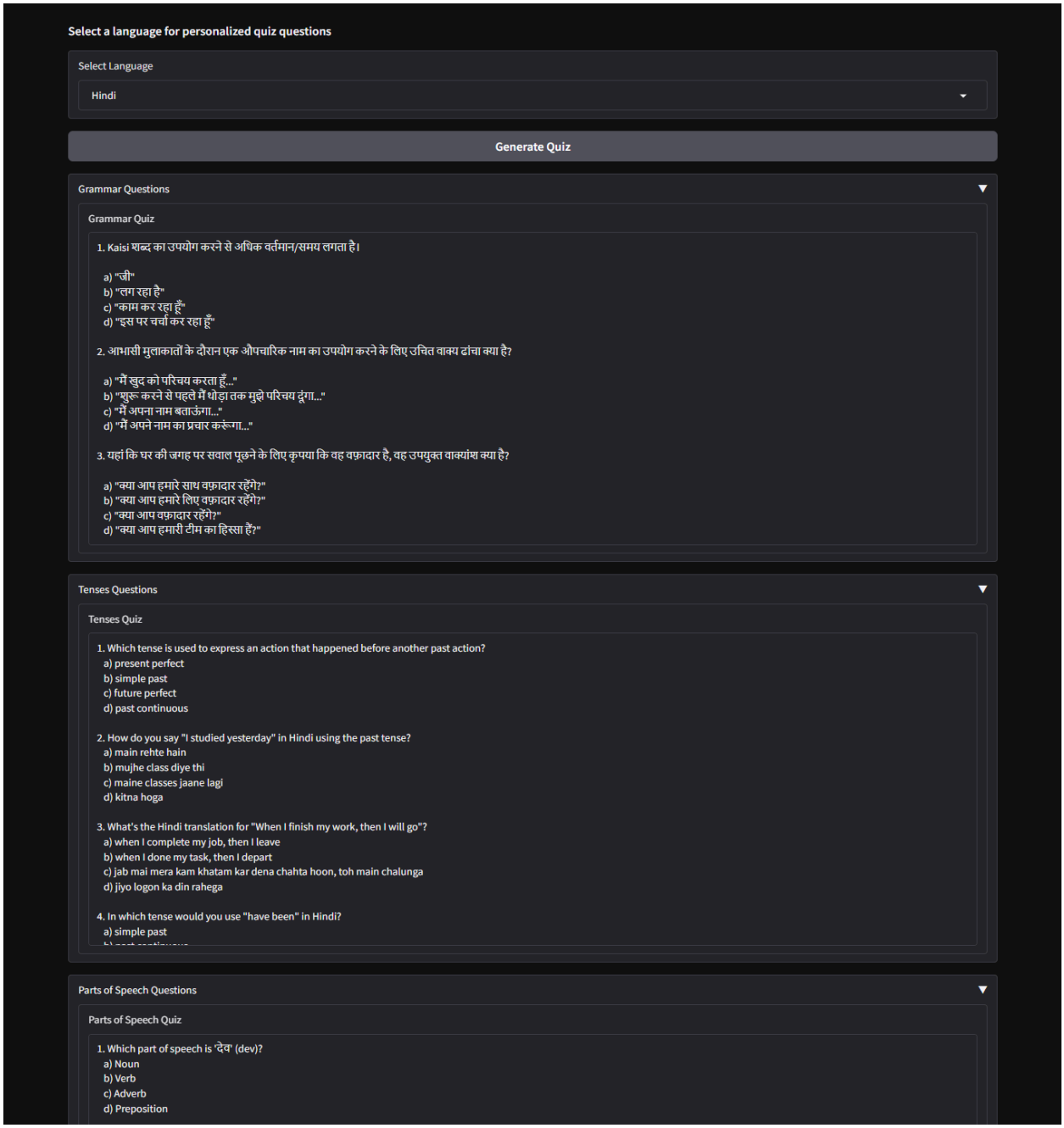
Description: A Spanish paragraph describing a character named Ana and her morning routine is entered for analysis. After clicking "Analyze Language", the app detects the language as Spanish and provides a detailed language breakdown. It includes insights into syntax and sentence structure, gender agreement, and verbal aspects, noting the use of the present tense and subject-verb-object word order.

Adaptive Quiz Page:



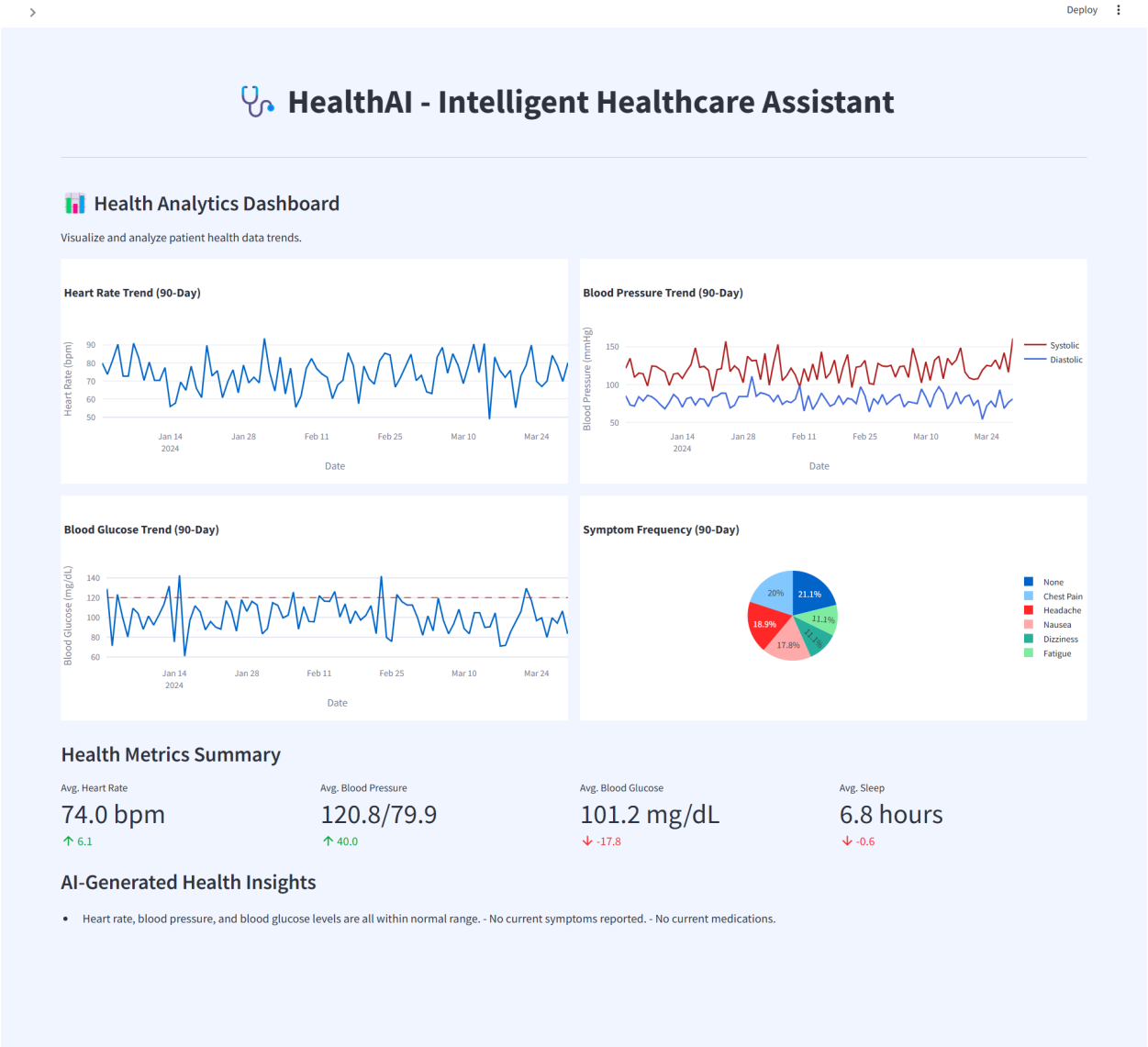
Description: This interactive quiz system generates personalized multiple-choice questions across six supported languages (English, Spanish, Chinese, French, German, Hindi). The quiz is organized into three categories: Grammar Questions, Tenses Questions, and Parts of Speech Questions. Each category contains 10 carefully crafted questions with four options each, complete with answer keys for self-assessment and learning reinforcement.

Adaptive Quiz output:



Description: The user has selected Hindi as the language for personalized quiz generation. The interface presents questions across three categories: Grammar, Tenses, and Parts of Speech. Grammar questions focus on sentence structure and correct usage in Hindi, while Tenses questions test the user's understanding of past, present, and future tense forms through translation-based exercises. The Parts of Speech section includes questions that require identifying nouns, verbs, and adjectives in given Hindi sentences, with correct answers provided.

Multilingual Learning Page:



Description: The Multilingual Learning section offers structured educational exercises tailored to specific languages and learning objectives. Users can select from six supported languages and choose between Grammar Exercises, Sentence Formation, and Tense Exercises. Each exercise type provides comprehensive learning materials including rule explanations, example sentences, practice activities, and complete answer keys for effective language skill development.

Multilingual Learning output:

Description:

Conclusion:

Utilizing IBM Watson Machine Learning capabilities, the application ensures accurate language analysis, comprehensive error detection, intelligent quiz generation, and structured learning exercises across six major languages. The systematic development process—spanning model integration, core feature implementation, frontend development with Gradio, and visualization enhancement—led to the creation of an interactive, educationally-focused platform.

Built with Gradio framework, Language Guru facilitates seamless interaction with AI-powered language tools and provides dynamic visualizations of learning progress through competency charts and text analysis metrics. This project highlights how targeted AI models and a well-structured educational framework can enhance language learning accessibility and effectiveness.

APPENDIX :

Source Code:[hosted on GitHub]

GitHub/Project Demo : https://github.com/Johnbi5/HealthAI-Intelligent-Healthcare-Assistant-Using-IBM-Granite