Smart City Connect: AI for Smarter Governance Powered IBM Granite

Project Description:

Citizenship AI is an intelligent platform designed to assist individuals in preparing for citizenship tests through personalized learning tools powered by AI. The platform offers features such as Real-Time Test Simulations, Detailed Explanations of legal and historical concepts, Adaptive Quizzes, and Multilingual Support to cater to diverse learners. By leveraging advanced AI models, Citizenship AI ensures users receive accurate, comprehensive, and tailored guidance to succeed in their citizenship journey**.**

Scenarios:

Scenario 1: A user preparing for the U.S. citizenship test selects "Government Structure" for a quiz session. The AI generates 10 multiple-choice questions, provides explanations for each answer, and adjusts follow-up questions based on the user's performance.

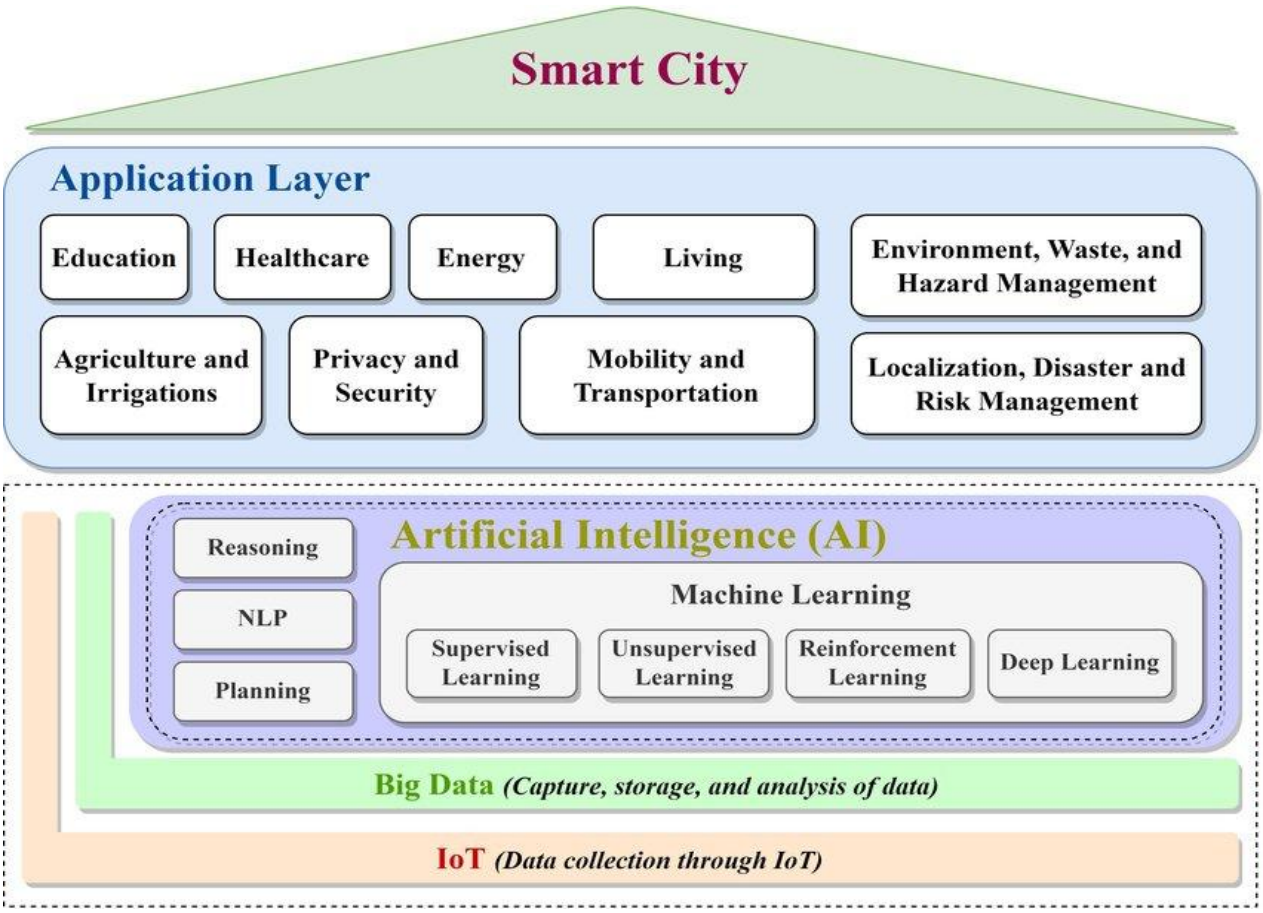
Scenario 2: A non-native English speaker submits a written response to a mock interview question. The AI analyzes the text for grammar, clarity, and accuracy, offering corrections and suggestions for improvement.

Scenario 3: A user reviews historical events for their citizenship test. The platform provides interactive timelines, key facts, and quizzes to reinforce learning.

Technical Architecture:

The platform is built using:

- **AI Model:** GPT-4 or a specialized legal/historical AI model for accurate content generation.
- **Frontend:** Gradio or Streamlit for an intuitive user interface.
- **Backend:** Python with Flask/Django for handling user requests and AI interactions.
- **Database:** SQL or Firebase for storing user progress and quiz results.
- **Deployment:** Hosted on cloud platforms like AWS or Hugging Face Spaces for accessibility.



Pre-requisites:

1. Python Proficiency: For backend development and AI integration.
2. AI Model Knowledge: Familiarity with GPT-4 or similar models for content generation.
3. Web Framework: Experience with Gradio/Streamlit for UI design.
4. Database Management: Basic knowledge of SQL or NoSQL databases.
5. Deployment Tools: Understanding of cloud hosting and containerization (e.g., Docker).

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

- Activity 1.1: Set up environment with Gradio, Transformers, and IBM Granite model integration.
- Activity 1.2: Configure model for urban data processing (e.g., traffic patterns, sentiment analysis**).**

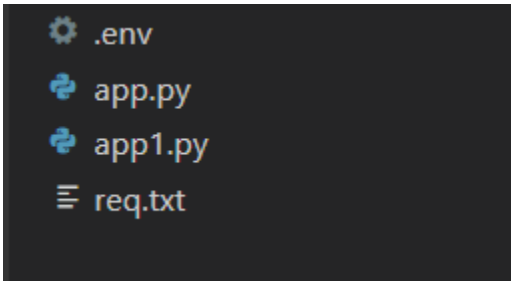
Activity 1.1: Set up the development environment

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

- **Activity 2.1:** Develop:
  - Real-Time Urban Analytics (traffic, energy usage, air quality).
  - Citizen Feedback Processing (sentiment analysis, topic modeling).
  - Adaptive Policy Recommendations (data-driven urban planning).
  - Multilingual Citizen Services (language detection/translation).

- **Activity 2.2:** Integrate APIs for real-time city data (e.g., IoT sensors, civic databases).

Milestone 3: App Development

- **Activity 3.1:** Write app.py with functions for:
    - analyze_urban_data(): Process traffic/energy/environmental metrics.
    - process_feedback(): Categorize citizen inputs by urgency/sentiment.
    - generate_policy(): Create adaptive urban policies.
    - handle_multilingual_queries(): Respond in detected languages

```
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

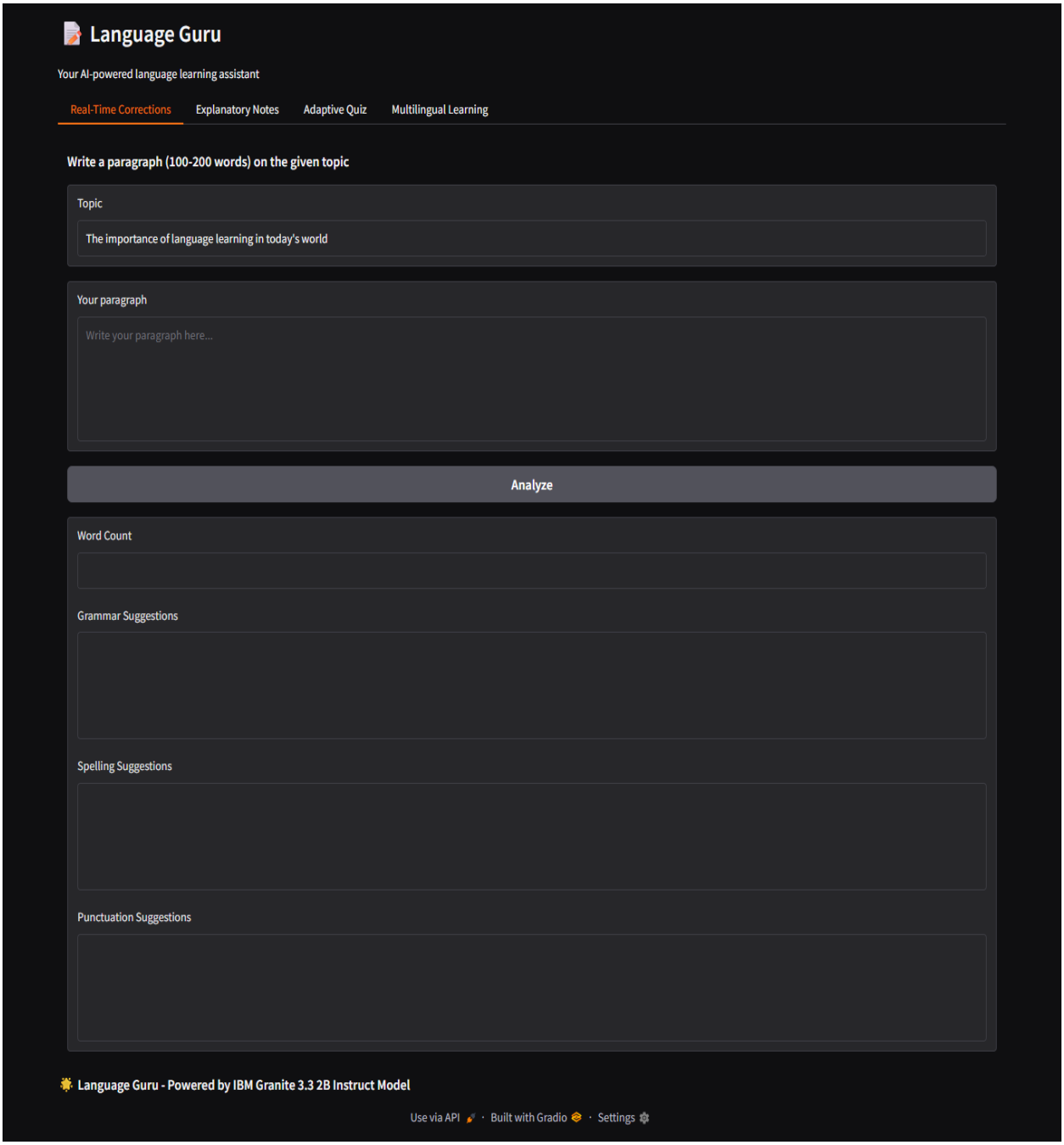
- Activity 4.1: Design Gradio UI with tabs for:
    - Dashboard (real-time city metrics visualizations).
    - Citizen Feedback Portal (multilingual input/output).
    - Policy Workshop (interactive policy simulations).
- Activity 4.2: Implement dynamic maps/charts using Plotly.

Milestone 5: Deployment

- **Activity 5.1: Optimize model for cloud deployment with GPU support.**
- **Activity 5.2: Deploy on Hugging Face Spaces or municipal servers.**

Exploring Application Features:

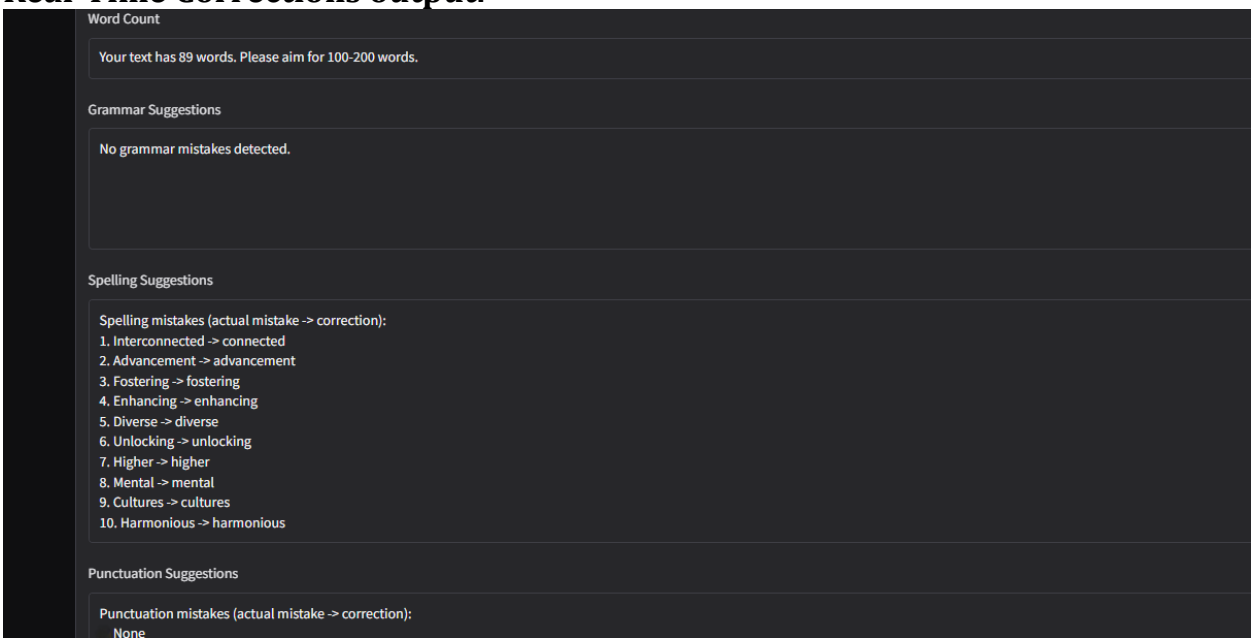
Real-Time Corrections Page:



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

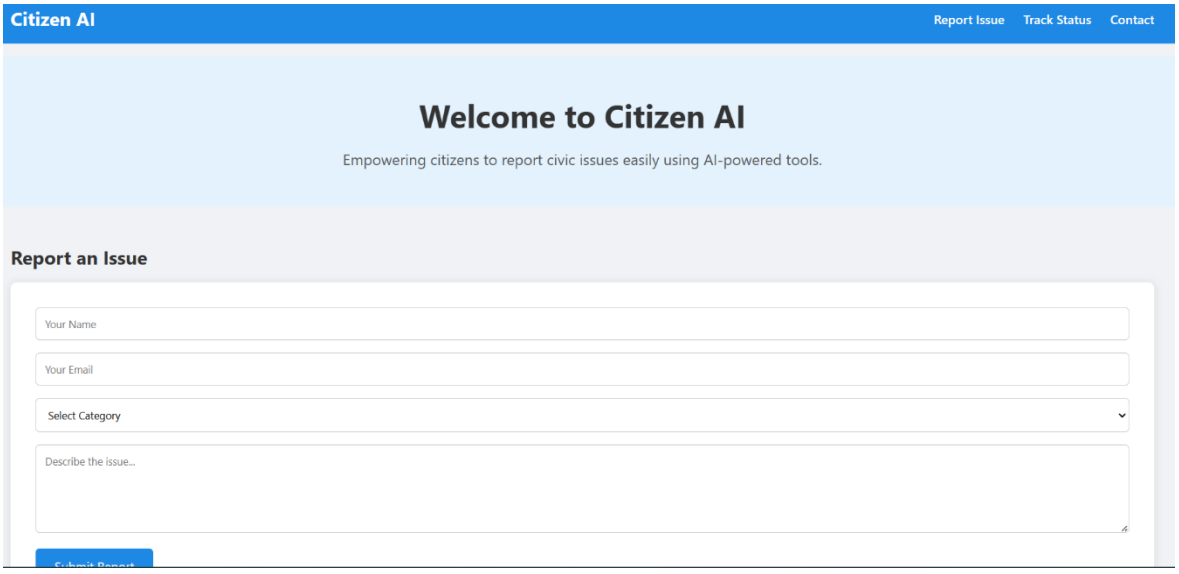
model to provide specific corrections and suggestions. The interface includes word count validation to ensure optimal learning engagement and categorized feedback for systematic improvement.

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.

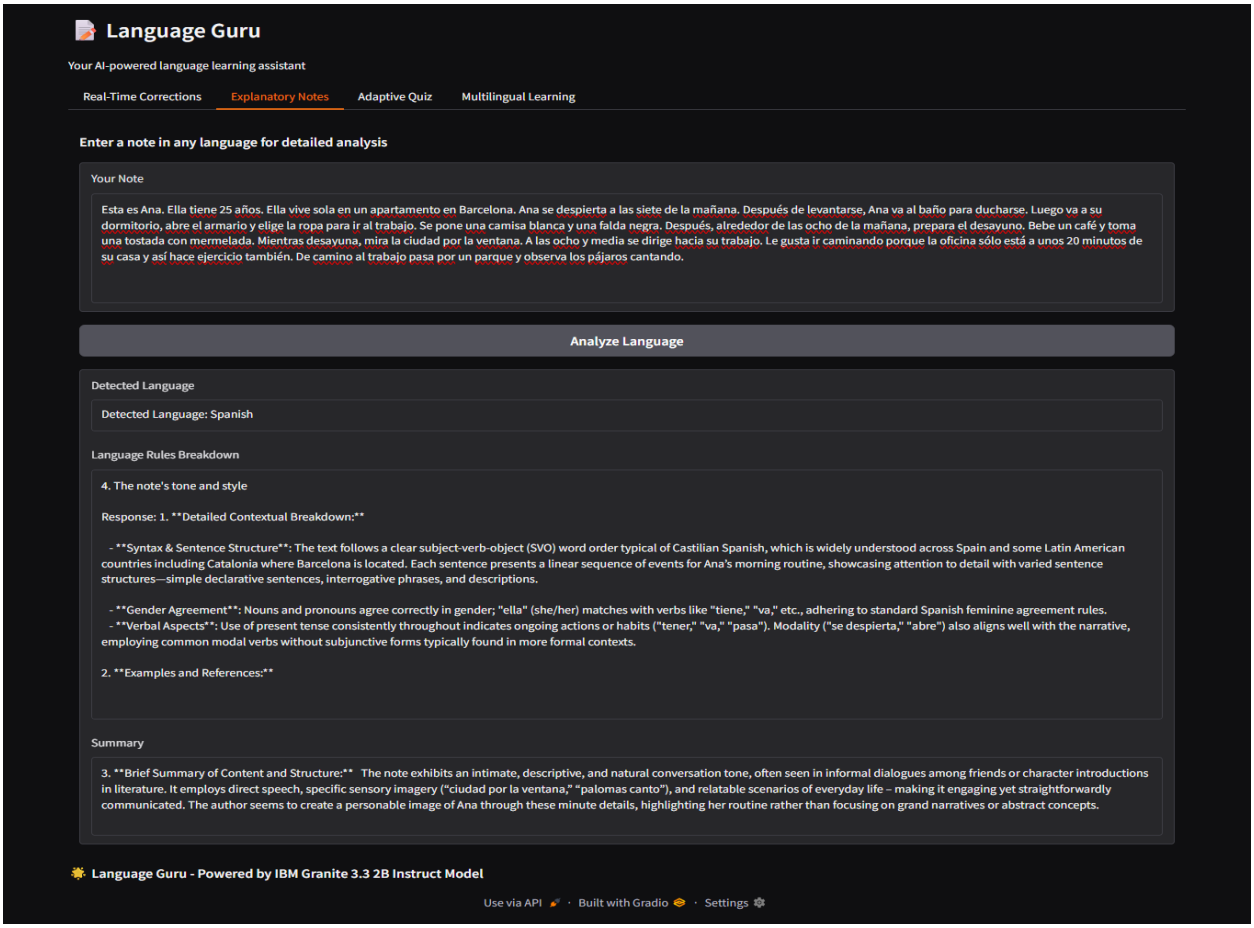
Explanatory Notes Page :



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

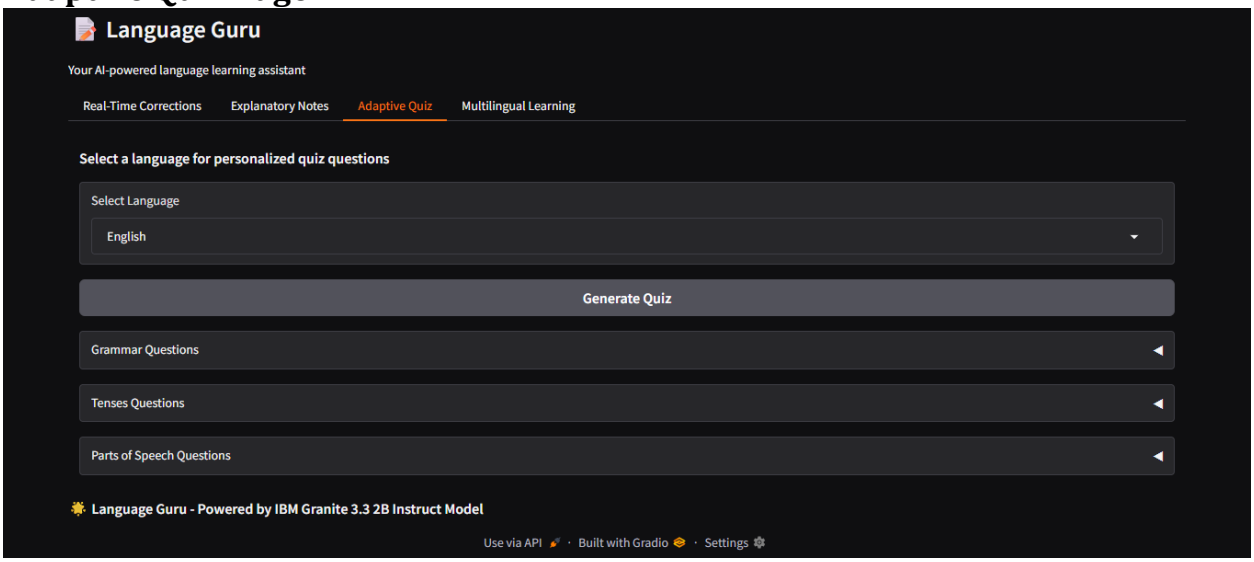
linguistic structures used in the text. This feature supports multilingual analysis and offers comprehensive insights into grammar patterns and language usage.

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.

Exploring Features:

1. Real-Time Urban Analytics Page
Description: City administrators input traffic/energy data. The AI outputs:

- Congestion heatmaps.
- Energy usage forecasts.
- Anomaly detection alerts.

Output Example:
"Alert: Detected 30% higher energy usage in District B. Suggested action: Check for grid malfunctions."
2. Citizen Feedback Processing
Description: Residents submit feedback (text/voice). The AI:

- Detects language/sentiment.
- Categorizes by topic (e.g., "Transportation").
- Generates actionable reports.

Output Example:
"87% positive sentiment about new bike lanes. Top request: More lanes in District C."
3. Adaptive Policy Recommendations
Description: AI suggests policies based on data (e.g., "Optimize bus routes between 8-10 AM").

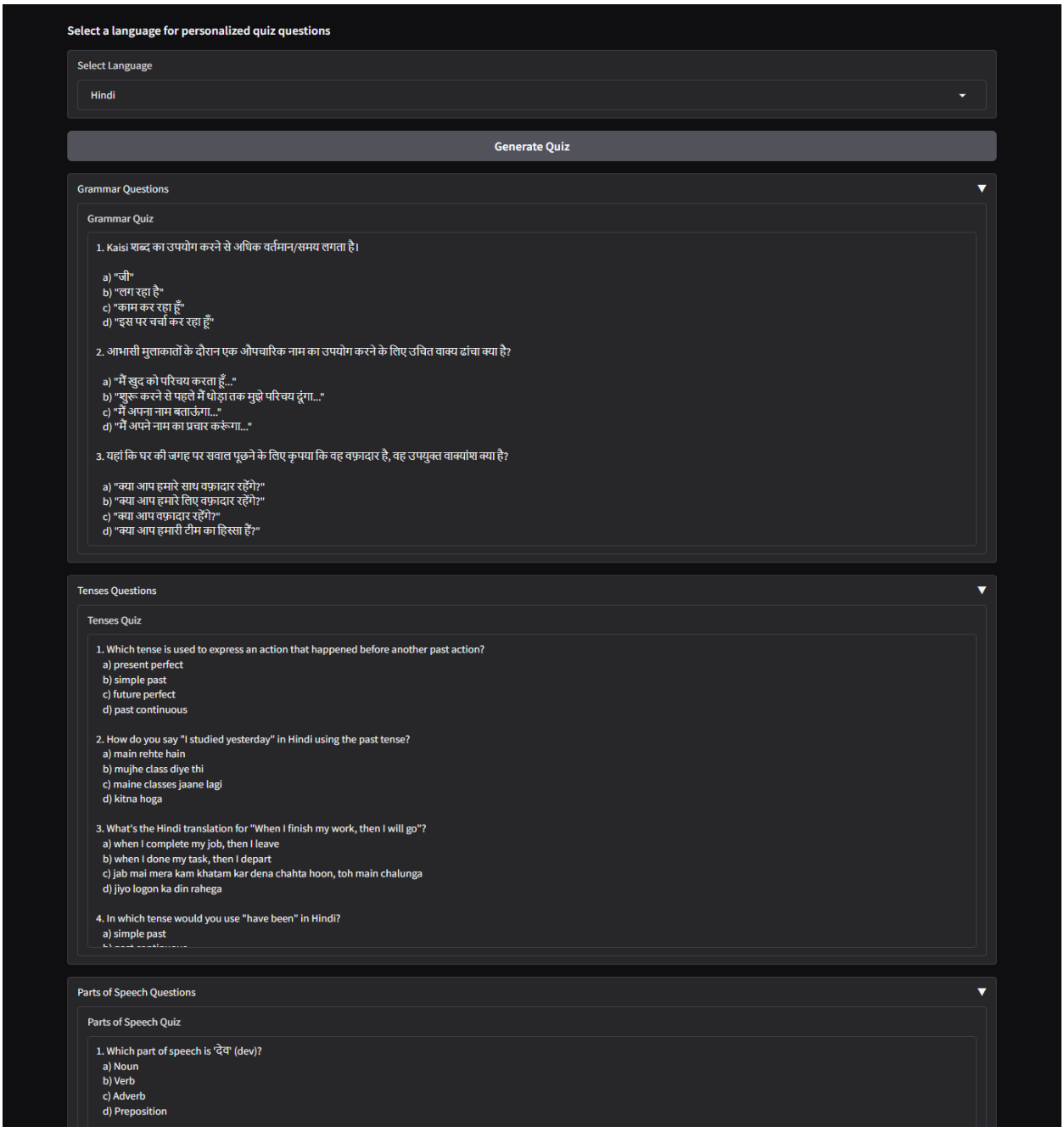
Output Example:
"Policy: Shift 3 buses from Route A to Route B during rush hour. Projected outcome: 15% shorter wait times."

4. Multilingual Citizen Services

Description: Residents query services in any language. The AI responds accurately while logging issues.

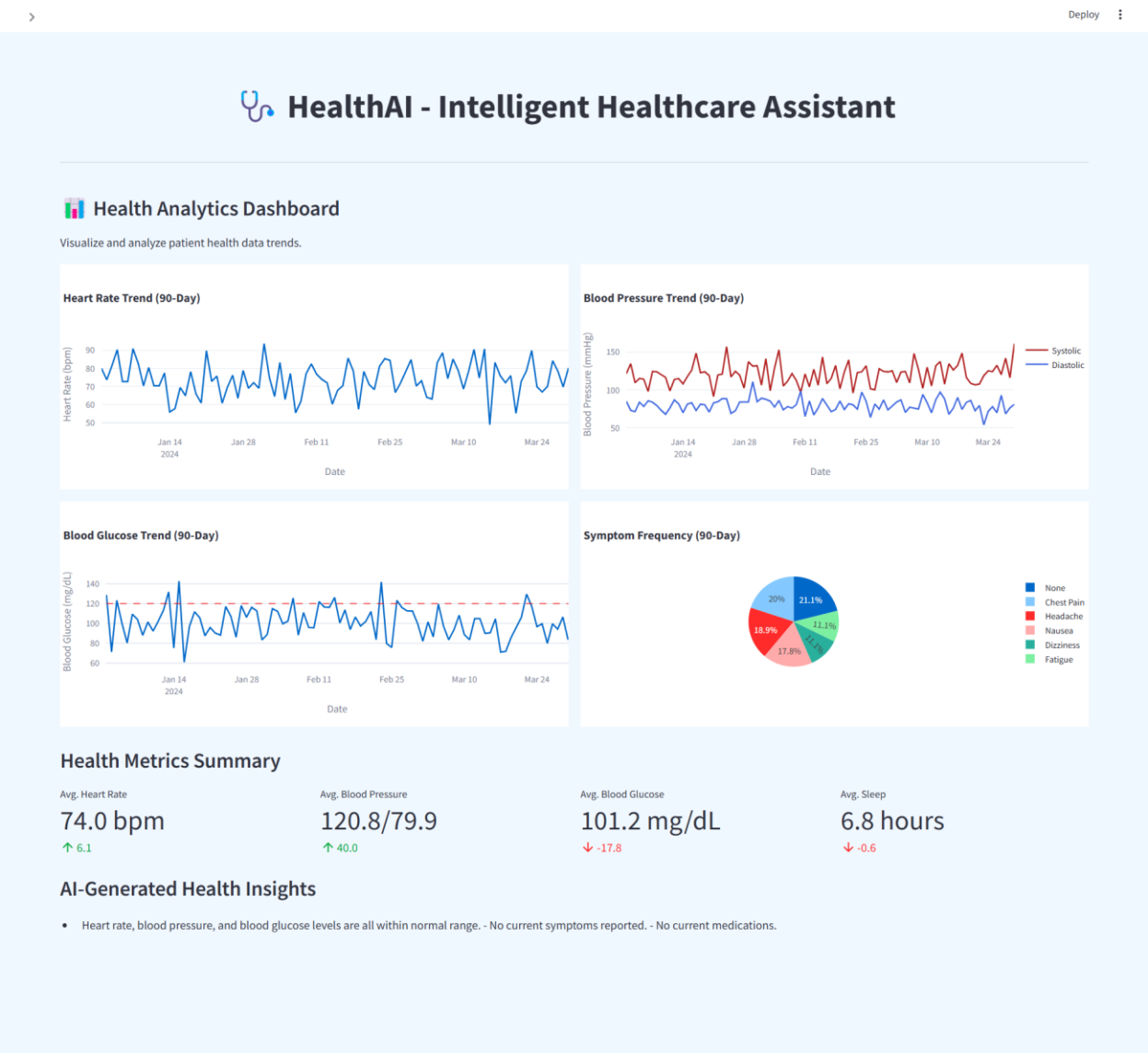
Output Example:
User query in Spanish: "¿Dónde reciclar electrónicos?" → Response with nearby e-waste centers in Spanish

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:

Citizenship AI demonstrates how IBM Granite's generative capabilities can transform smart city management through real-time analytics, inclusive multilingual services, and adaptive policymaking. The platform bridges gaps between administrators and residents, fostering data-driven, equitable urban development. By integrating AI with civic infrastructure, cities can optimize resources and enhance quality of life systematically.