**Project Documentation**

**Citizen AI – Intelligent Citizen Engagement Platform**

**1. Introduction**

* **Project Title:** Citizen AI – Intelligent Citizen Engagement Platform
* **Team Members:**
  + **Team Leader:** Kandavalli Ashok
  + **Team Member:** Jannu Eeswari
  + **Team Member:** Jakkula Kavya
  + **Team Member:** Jakkula Sai Venkata Sathwik

**2. Project Overview**

* **Purpose:**  
  Citizen AI is built to streamline how citizens interact with government services by providing instant AI-powered answers to their queries using a chatbot.
* **Features:**
  + Real-time conversational chatbot
  + Context-aware responses using IBM Granite 3.3–2B Instruct
  + Simple Gradio web interface
  + Deployable on cloud or local machine

**3. Architecture**

* **Frontend:**  
  Implemented using **Gradio**, a Python-based UI library for building fast and interactive web interfaces.
* **Backend:**  
  Built with **Python**, using the transformers library to load and interact with the IBM Granite LLM.
* **Database:**  
  No database is used in the current version as the model is stateless. Future versions may integrate with MongoDB for feedback and logs.

**4. Setup Instructions**

* **Prerequisites:**
  + Python 3.9+
  + pip (Python package manager)
  + Hugging Face account (for access to the IBM Granite model)
* **Installation Steps:**

git clone https://github.com/your-username/citizen-ai-chatbot.git

cd citizen-ai-chatbot

pip install -r requirements.txt

Replace "HF\_TOKEN" in the script with your Hugging Face token.

Or load the model once and save it locally to avoid using token each time

**5. Folder Structure**

citizen-ai-chatbot/

│

├── granite\_chatbot.py # Main code (Gradio + HuggingFace pipeline)

├── /content/granite-model # Local model files after saving

├── README.md # Project overview and usage

└── requirements.txt # Required Python packages

**6. Running the Application**

* **Frontend&BackendCombined:**  
  Since Gradio wraps both in one file, run:

python granite\_chatbot.py

**7. API Documentation**

No separate REST API is exposed. The app uses a Gradio interface for input and response:

* **Input:** Text prompt (e.g., "How to apply for a driving license?")
* **Output:** Generated AI response based on IBM Granite LLM

**8. Authentication**

* No authentication is required in the current build.
* Future enhancements may include:
  + Admin login to monitor questions
  + OAuth or JWT token-based access for citizens

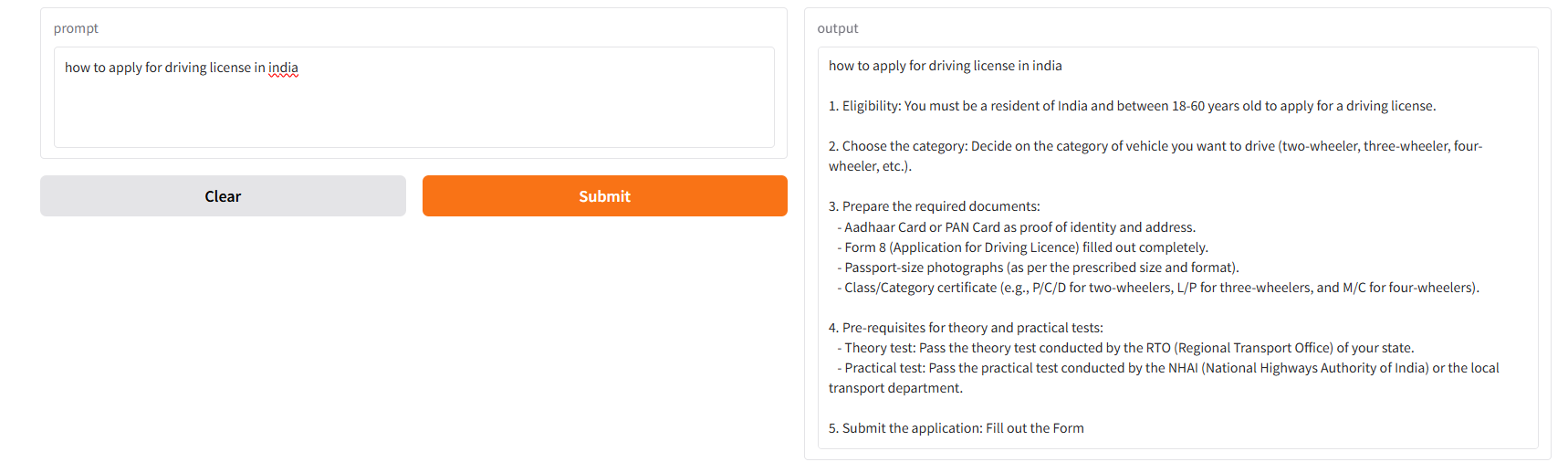
**9. User Interface**

* Simple textbox-based interface built with Gradio.
* Prompt area + response display
* Hosted as a one-page app with title and description

**10. Testing**

* **Manual Testing:**
  + Verified multiple prompts for response accuracy and latency.
  + Load tested with multiple queries in a row.
* **Tools Used:**
  + Python logging
  + Response validation against expected structure

**11. Screenshot**



**12. Known Issues**

* Requires internet for model download (first-time load).
* No live connection with official government databases or APIs.
* Cannot verify accuracy for very region-specific queries.

**13. Future Enhancements**

* Add multilingual support (Telugu, Hindi, etc.)
* Integrate with government APIs for live info
* Sentiment analysis on citizen feedback
* Dashboard for administrators
* Voice-based interface for accessibility