Project Overview

Project: CivicConnect AI - Smart City Assistant

1. Project Overview

CivicConnect AI is an intelligent chatbot designed as a "smart city assistant". It provides a modern and intuitive interface for citizens to report common civic issues and get instant information about city services. The bot is now powered by **Google Gemini API**, bringing real-time AI capabilities to the application.

2. Team Details

- **Team Name:** Thunder Script
- **Team Members:**
  1. Ritesh Raut (2301544)
  2. Mohit Waghmare (2125578)
  3. Kshitij Ramteke (2301601)
  4. Manal Kamble (2301643)
  5. Samiksha Dahule (2301658)
  6. Vedant Telang (2148097)
  7. Tushar Kalmegh (2146960)

- **Author:** Ritesh Raut (Programmer Analyst, Cognizant)
- **Connect with the Author:**
  - GitHub
  - LinkedIn

3. Deep Dive Solution

Use Case:

The core use case is to provide intelligent, context-aware responses and streamlined reporting for common civic issues. The bot now automates the process by generating fictional report IDs (e.g., CR123456), identifying the correct city department based on the issue, and asking for necessary follow-up details like location or address to complete a report.

Solution & Benefits (Features):

The project is a full-stack application with a responsive chat UI and a Python/Flask backend integrated with the Google Gemini API. Key features include:

- **Gemini-Powered AI:** Enables intelligent, context-aware responses.
- **Automated Ticket Generation:** Creates fictional report IDs for user-submitted issues.
- **Department Routing:** Identifies the appropriate city department based on the reported issue.
- **Session Management:** Maintains separate chat histories for different users.
- **Enhanced UX:** Includes a typing indicator, auto-resizing text area, day/night mode, and attachment support.

Innovativeness:

Innovation has evolved from a pure frontend, rule-based approach to a full-stack solution with a live AI backend. The use of the Gemini API allows the bot to handle complex queries, generate unique report IDs, and route issues to specific departments, showcasing a practical application of modern AI to enhance civic communication.

User Experience:

The user experience is significantly enhanced by the new AI backend. The bot's ability to provide context-aware responses and follow-up prompts makes the conversation feel more natural and efficient. The existing intuitive chat UI, responsive design, and user-friendly features remain integral to the positive experience.

Business Opportunity / Market Potential:

As a "smart city assistant" now powered by real AI, CivicConnect AI has much stronger market potential for municipalities. The ability to intelligently manage and route inquiries, in addition to automating responses, can significantly reduce operational costs, increase citizen satisfaction, and make it a highly valuable tool for city administrations.

Ease of Implementation:

The project now has a full-stack setup that requires more steps than the original front-end-only version. It involves setting up a Python virtual environment, installing dependencies from requirements.txt, and configuring a .env file with a Google Gemini API key.

Scalable / Reusable:

The full-stack architecture with Python/Flask and the Google Gemini API is highly scalable and reusable. The backend is now fully functional and designed to handle real-time AI processing, making it a robust foundation that can be adapted for various smart city applications or different AI models.

Financial Feasibility:

The initial development costs leverage open-source technologies (HTML, CSS, JavaScript, Python, Flask), keeping them low. However, the full-stack version with Gemini API integration will have associated costs based on API usage, which should be considered for long-term operational feasibility.

4. MVP (Minimum Viable Product)

The current full-stack MVP provides a live, interactive chatbot experience with core functionalities:

- Real-time AI responses to civic issues powered by the Gemini API.
- Generation of fictional report IDs for user issues.
- Identification of relevant city departments based on user queries.
- Maintenance of conversation history for a better user experience.
- User-friendly features like a dynamic input box, typing indicator, and a theme toggle.

5. Future Scope

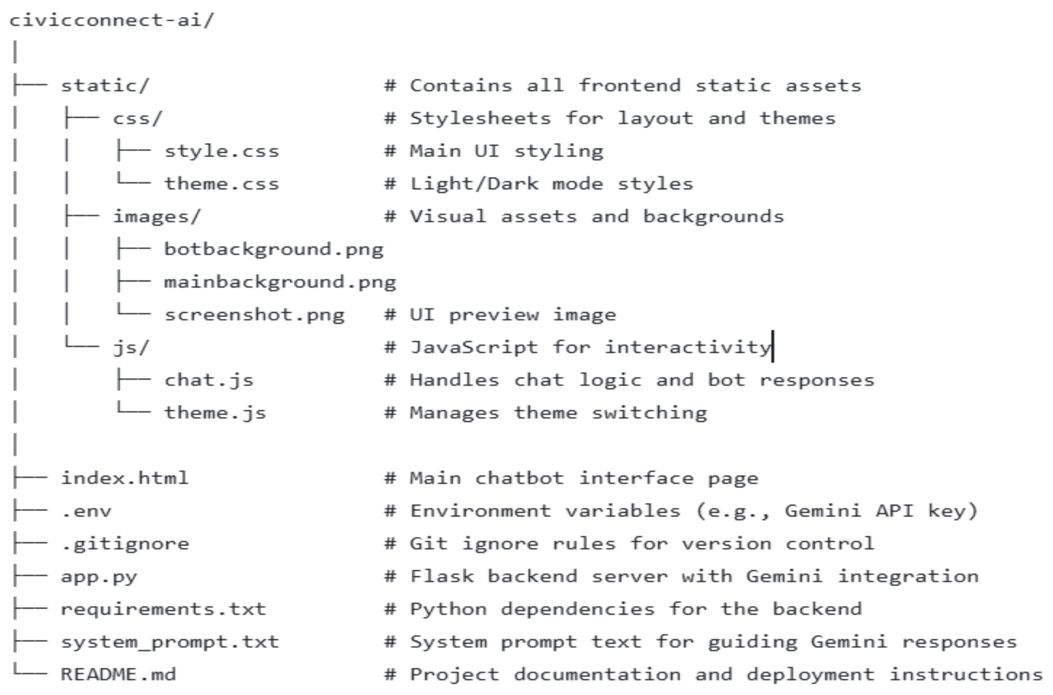
The future scope is to build upon the current full-stack foundation. Potential enhancements include:

- Integrating the bot with real city databases for real-time information retrieval (e.g., live traffic updates, permit status).
- Implementing a more sophisticated natural language understanding (NLU) model to handle a wider range of nuanced and complex user queries.
- Developing an administrative dashboard for city officials to monitor and manage reported issues.
- Integrating with external APIs for location services to pinpoint the exact addresses of reported issues.

6. Tech Stack

Layer	Technologies Used
Frontend	HTML5, CSS3, JavaScript (ES6+), Bootstrap
Backend	Python, Flask, Google Gemini API, python-dotenv, Pillow
Deployment	Azure App Service

7. File Structure



```
civicconnect-ai/
│
├── static/                  # Contains all frontend static assets
│   ├── css/                 # Stylesheets for layout and themes
│   │   ├── style.css        # Main UI styling
│   │   └── theme.css        # Light/Dark mode styles
│   ├── images/              # Visual assets and backgrounds
│   │   ├── botbackground.png
│   │   ├── mainbackground.png
│   │   └── screenshot.png   # UI preview image
│   └── js/                  # JavaScript for interactivity
│       ├── chat.js          # Handles chat logic and bot responses
│       └── theme.js         # Manages theme switching
│
├── index.html               # Main chatbot interface page
├── .env                     # Environment variables (e.g., Gemini API key)
├── .gitignore               # Git ignore rules for version control
├── app.py                   # Flask backend server with Gemini integration
├── requirements.txt         # Python dependencies for the backend
├── system_prompt.txt        # System prompt text for guiding Gemini responses
└── README.md                # Project documentation and deployment instructions
```

8. Resources & Links

- **GitHub Personal Repository:** https://github.com/Riteshraut0116/CivicConnect-AI-Bot/tree/azure_final_deployment

- **Azure Devops Repository:** https://dev.azure.com/Vibects12/2301544/_git/2301544?version=GBmaster

- **Live Demo (UI Example Only):** https://civicconnectbot.netlify.app/

- **Live Demo (Azure App Service Deployment):** https://cts-vibeappce3814-3.azurewebsites.net/

9. Actual Live Demo Steps (Full Stack with Gemini AI Backend)

This process will run the application locally on your machine, demonstrating the live interaction with the Gemini API.

Prerequisites:

- Python 3.7+
- A Google Gemini API Key

Steps:

1. **Clone the repository:** Open your terminal or command prompt and clone the project.
   - git clone <GitHub repository URL>
2. **Navigate to the project directory:**
   - cd CivicConnect-AI-Bot
3. **Checkout the correct branch:**
   - git checkout azure_final_deployment
4. **Set up the environment:**
   - Create and activate a virtual environment: python -m venv venv and source venv/bin/activate (or venv\Scripts\activate for Windows).

5. **Install dependencies:**
   - pip install -r requirements.txt
6. **Add your Gemini API key:**
   - Create a .env file in the civicconnect-ai directory and add your key in the format: GEMINI_API_KEY="your_api_key_here"
7. **Run the Flask server:**
   - python app.py
8. **Access the application:**
   - Open your web browser and go to http://localhost:5000 to see the chatbot in action.