A PROJECT REPORT

On

Citizen-AI-Intelligent citizen Engagement platform

By

Uda Mahesh(22FE1A05H1)

Viyyapu Yugandhar( )

Vasumalla Sony(22FE1A05H8)

Under the guidance of

Ganesh M

TABLE OF CONTENTS

ABSTRACT

Citizen AI is an intelligent, web-based platform developed to transform the traditional mechanisms of citizen engagement into a modern, data-driven, and AI-assisted system. Developed using Python (Flask) for the backend, JavaScript and Bootstrap for the frontend, and NLP libraries such as TextBlob for sentiment analysis, the platform creates a seamless interface between the government and its citizens.

The core functionality includes a real-time conversational assistant (chatbot), a structured feedback collection system, and an analytics dashboard that visualizes public sentiment using interactive charts. By enabling citizens to raise queries, provide suggestions, and express concerns, the platform empowers them to participate more actively in governance.

Unlike traditional government websites that offer static information and delayed response mechanisms, Citizen AI supports 24/7 interaction through an intelligent assistant. While the current implementation uses a rule-based engine for FAQs and services, the architecture is designed to be easily upgraded to incorporate advanced language models such as OpenAI's GPT-3.5, IBM Watson Assistant, or IBM Granite. This flexibility ensures that Citizen AI can evolve into a fully AI-driven public communication and governance tool in future iterations.

Furthermore, the feedback collected is automatically classified into sentiment categories — Positive, Neutral, or Negative — using NLP techniques. This allows administrative departments to monitor real-time public opinion and take informed, timely actions. The built-in dashboard provides easy-to-understand visual summaries of citizen engagement, making the platform not only a tool for communication but also for transparency and data-backed decision-making.

Citizen AI has the potential to improve public service delivery, increase trust in digital governance, and serve as a foundational model for smart city and e-governance platforms.

Introduction

1.1 Project Overview

Citizen AI – Intelligent Citizen Engagement Platform is a forward-thinking, AI-ready web application designed to transform how citizens interact with public authorities. In a digital era where efficiency, transparency, and real-time response are critical, Citizen AI leverages open-source technologies to create an intelligent, scalable, and modular solution for government-citizen communication.

Developed using Python's Flask microframework for the backend, HTML/CSS and Bootstrap for responsive UI design, and JavaScript for frontend interactivity, the platform integrates natural language processing using TextBlob for sentiment analysis. It is designed to be lightweight yet powerful, enabling real-time chatbot interaction, automatic sentiment tagging of feedback, and live visualization of public sentiment trends on a secure, offline-capable dashboard.

Key objectives include:

- Real-time Conversational Assistant: A chatbot interface that can understand and reply to common government-related queries like PAN card download, Aadhaar updates, and passport procedures. The system uses a rule-based logic engine with potential for integration with advanced AI models like OpenAI, IBM Watson, or IBM Granite.
- Sentiment Analysis on Citizen Feedback: Feedback collected through the platform is automatically classified as Positive, Negative, or Neutral using NLP, providing actionable insights to administrators.
- Interactive Admin Dashboard: Built with Chart.js, the dashboard offers a real-time pie chart view of sentiment trends, enabling data-driven decision-making for public service improvement.
- Modular Architecture: The entire platform is designed with clear separation of concerns: routing and business logic (Flask), analytics (pandas + JS), database (SQLite), and UI (Bootstrap), allowing for easy customization and expansion.

- Offline and Local Deployment Support: Operates without the need for internet connectivity using SQLite, ensuring accessibility in rural and low-bandwidth regions.

Citizen AI redefines the role of government websites, transforming them from passive information boards to active, responsive platforms that support participatory governance, transparency, and digital inclusion.

1.2 Purpose

The purpose of this project is to:

- Improve public service delivery by enabling citizens to interact with a virtual assistant for faster resolution of common issues.
- Empower government agencies with real-time feedback and sentiment trends to make data-driven decisions.
- Demonstrate the feasibility of lightweight AI tools in civic engagement systems using open-source, cost-effective technologies.
- Lay the groundwork for future AI integrations, such as language support, predictive issue detection, and mobile accessibility.

This document presents:

- A detailed walkthrough of system features: chat, feedback, sentiment analysis, and analytics.
- Explanation of architectural components: Flask routes, database schema, JavaScript integrations, and rule-based AI logic.
- Descriptions of NLP usage, feedback storage, and visualization mechanisms.
- System testing methods, limitations, and scope for enhancement using full AI services.
- Visual illustrations, diagrams, and screenshots from the working prototype.

Ultimately, this report serves as a comprehensive case study and technical guide, illustrating how Citizen AI can enhance government–citizen interaction, build trust, and modernize governance in a scalable and sustainable manner.

IDEATION PHASE

2.1 Problem Statement:

Public interaction with government websites remains outdated, lacking real-time communication, intelligent response systems, and feedback analysis tools —leading to citizen dissatisfaction and delayed administrative action.

Explanation:
Most government portals provide only static content, requiring users to rely on emails, calls, or physical visits to obtain services or raise complaints. Citizens often receive no timely assistance or feedback acknowledgment. Moreover, public feedback, even if collected, is rarely analyzed systematically for sentiment or urgency. This leads to a communication gap between government departments and the public, reducing trust and efficiency in service delivery. Citizen AI solves this by offering a lightweight, AI-ready platform that enables intelligent chatbot interactions, automatic sentiment classification, and live analytics—streamlining citizen engagement and governance.

Customer Problem Statement Template

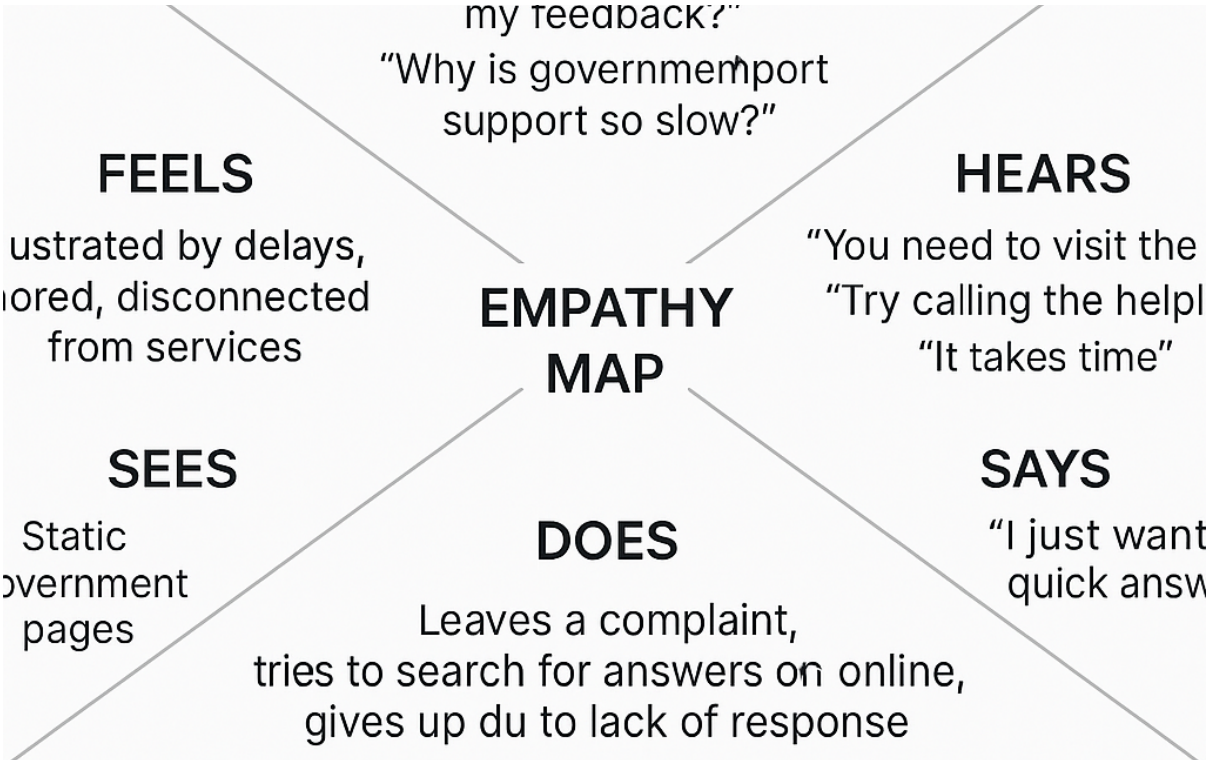
Question	Answer
Who is experiencing the problem?	Citizens attempting to interact with government services, and feedback and service satisfaction.
What is the problem?	Lack of real-time assistance, intelligent response systems, and government communication portals.
When/Where is it happening?	While accessing government websites for services like PAN, A

Why is it important to solve?	Delays in support reduce citizen trust, lead to inefficiency in g timely policy improvements.
How is the problem currently handled?	Through static forms, generic FAQs, delayed manual replies vi analysis.
What makes your solution better?	Citizen AI provides an interactive chatbot, sentiment analysis, driven, AI-powered engagement at low cost.

2.2 Empathy Map

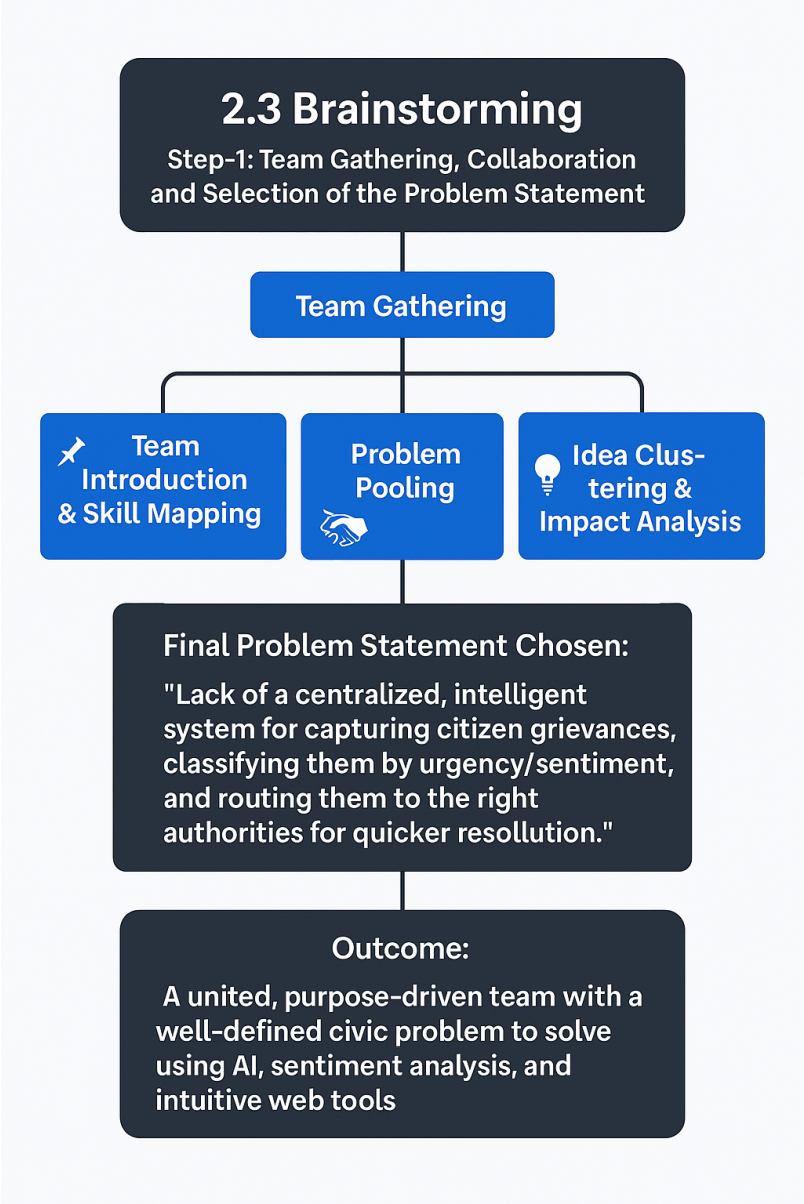
To better understand the user experience and challenges, an empathy map was created for a typical Citizen AI user—such as a local citizen seeking information about government services or an official analyzing public feedback. This visual tool captures what the user thinks, feels, sees, hears, says, and does while interacting with the system. It also highlights their pain points and desired gains.

This empathy map ensures that the Citizen AI platform is designed around the real-world needs and emotions of its end users, enabling more human-centered and effective public service communication.



2.3 Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

Individual Brainstorming

Before coming together as a team, both members explored the problem space individually:

Mahesh's Exploration:

- Noted that **most government websites are static** with no intelligent chatbot or instant support.
- Researched **Flask**, **TextBlob**, and **SQLite** as lightweight tools for rapid development.
- Explored how **rule-based NLP chatbots** can handle basic FAQs without external APIs.
- Reviewed **data privacy** challenges when handling citizen feedback.

Sony's Research:

- Studied **common UI pain points** in government portals: long forms, outdated designs, and poor responsiveness.
- Focused on building an **accessible, mobile-friendly interface** using Bootstrap and JS.
- Created early wireframes for:
    - A **chat interface**
    - A **feedback submission form**
    - A **sentiment analysis dashboard**

✳ Group Collaboration

After the individual phase, Mahesh and Sony met to consolidate ideas, assess feasibility, and finalize the scope:

Key Questions Discussed:

- What are the limitations of current government systems?
- How can we keep the tool usable even in rural or offline settings?
- Should we use rule-based or AI-based answering for the chatbot?

- What technologies ensure fast performance and low cost?

Final Problem Statement:

"Citizens face delays and lack of transparency in their interaction with public services due to static, outdated digital interfaces. There is no intelligent system for answering queries or analyzing feedback."

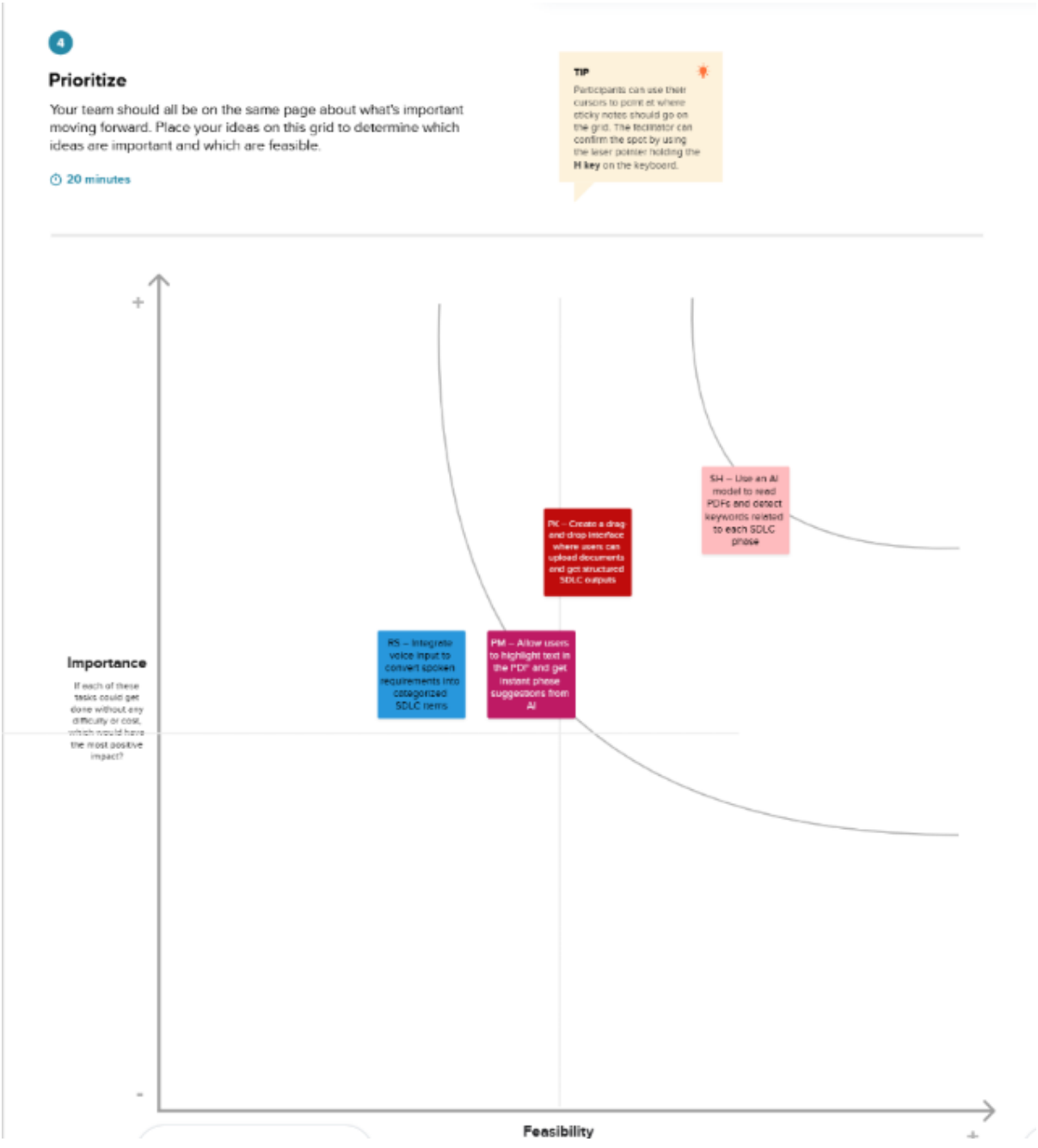
Chosen Solution Approach:

- A web platform using Flask (Backend) + Bootstrap (Frontend).
- Includes a chatbot, feedback system, and admin dashboard.
- Built to work offline-first, with options to plug in IBM Watson/OpenAI in future.

Brainstorming Outcomes:

- Defined user roles: Citizens and Government Staff.
- Listed core modules:
    1. Real-time chatbot
    2. Sentiment analysis via TextBlob
    3. Dashboard with live analytics
- Planned future enhancements:
    1. AI-powered query generation (Granite/OpenAI)
    2. Support for Indian languages
    3. Role-based dashboards
    4. Geo-tagging and policy notifications

Step-3: Idea Prioritization



REQUIREMENT ANALYSIS

3.1 Customer Journey map:

Phase	Steps (What user does)	Interactions (With what or whom)	Goals & Motivations	Positive Moments	Negative Moments	Opportunities
Entice	Hears about Citizen AI through a social media post or college event	Social media, friends, faculty	Curious to try AI-based chatbot for feedback	Learns it works offline and is simple	Unsure if it's trustworthy or official	Include verified badges, trust indicators
Enter	Visits Citizen AI site and sees chatbot and dashboard	Web browser, landing page	Wants to try chat or submit feedback	UI is simple, no login needed	Unsure what to ask chatbot	Provide sample queries or FAQs
Engage	Chats with bot or submits feedback	Chat interface, feedback textbox	Get an answer or vent frustration	Gets an instant response and sentiment tag	Bot can't answer complex or unrelated queries	Integrate OpenAI or Watson for broader coverage
Exit	Views sentiment result or sees dashboard	Dashboard, sentiment chart	Understand what others feel, view trends	Sees pie chart or admin view	No export option for reports	Add CSV/PDF export option for dashboard
Extend	Tells others, or gives more feedback later	WhatsApp, student groups, re-visits site	Help improve public systems through tech	Becomes a regular user or gives idea to staff	No mobile version or notification alert	Build mobile app, add reminder/alert system

3.2 Solution Requirement

The following are the functional requirements of the proposed Citizen AI system:

1. Chatbot Interface
    - Users can interact with a chatbot on the homepage to ask questions related to government services.
    - The chatbot provides instant, rule-based responses based on predefined FAQs.
2. Feedback Submission
    - Citizens can submit feedback using a simple text input form.
    - Submitted feedback is saved in a database for future reference.
3. Sentiment Analysis
    - The system automatically analyzes the sentiment of submitted feedback (Positive, Negative, Neutral).
    - It uses the TextBlob library for basic natural language processing.
4. Dashboard Visualization
    - Admins or officials can view a live dashboard showing sentiment trends via pie charts.
    - Dashboard auto-updates using real-time data from the database.
5. Offline Support
    - All core features work offline (chatbot, feedback, dashboard), relying only on local SQLite and Flask backend.
6. Data Storage
    - Feedback and chat messages are stored in an SQLite database.

- Each entry includes timestamp and role (user or bot).

7. Modular Design
   - The system is built in a modular way to allow future upgrades such as OpenAI/Watson API integration.
   - Files are separated into routes, utilities, templates, and static components.
8. Secure Input Handling
   - Input data is sanitized and validated to avoid crashes and errors.

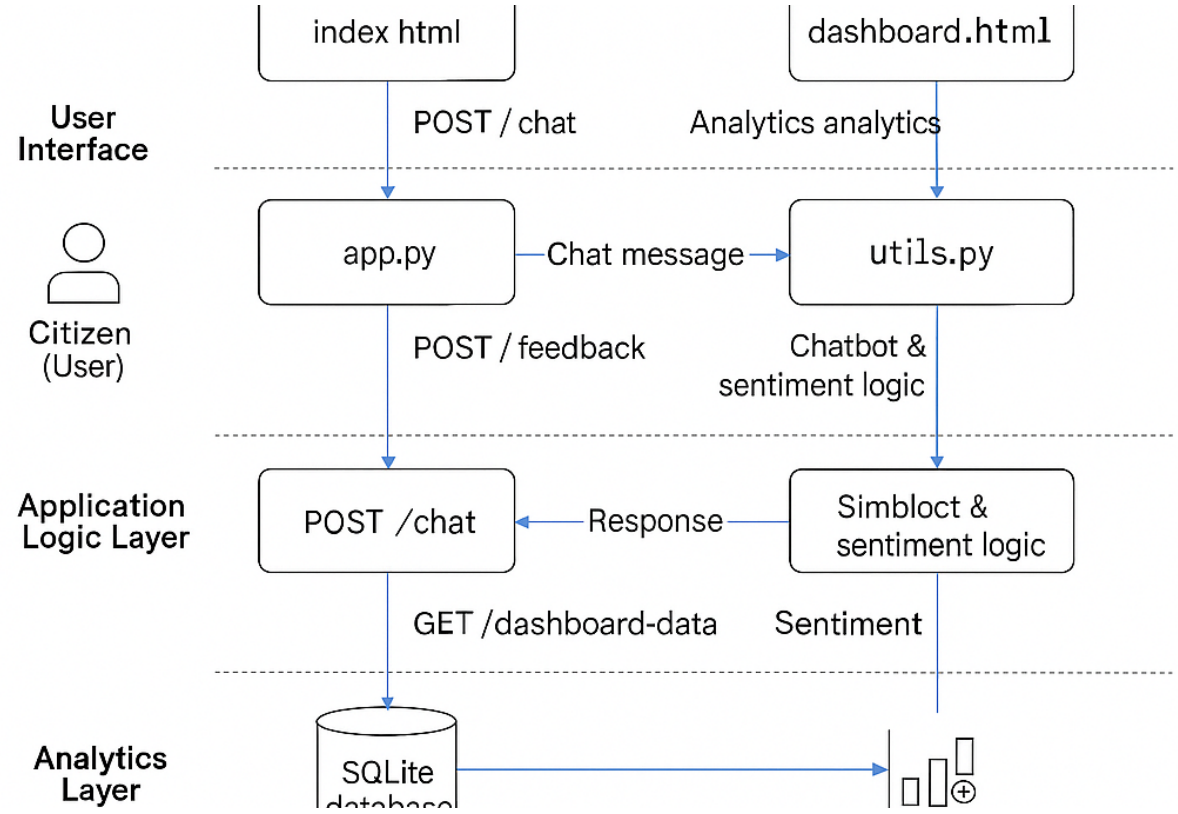
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Clean, responsive user interface using Bootstrap and Flask templates. Easy for citizens and officials to navigate.
NFR-2	Security	Feedback is stored securely in local database. API keys (for future AI models) are managed via environment variables.
NFR-3	Reliability	The system runs consistently on local environments with robust SQLite handling for messages and feedback.
NFR-4	Performance	Fast response to chat queries and sentiment analysis (within milliseconds) even on low-end devices.
NFR-5	Availability	Offline-first architecture allows 24/7 usage without internet unless AI APIs are enabled.
NFR-6	Scalability	Modular backend structure allows easy future upgrades to cloud, multi-language, and multi-user versions.

3.3 Data Flow Diagram

A **Data Flow Diagram (DFD)** is a visual tool used to represent how information moves through a system. For the **Citizen AI** project, the DFD illustrates how citizen queries, feedback, and system responses flow across various modules. This helps stakeholders understand how data is captured, processed, stored, and analyzed within the application.



4.2 Citizen AI – Data Flow Steps

1. User (Citizen)

- **Role**: Initiates interaction with the system.
- **Action**: Sends a message via the chatbot or submits feedback.
- **Input**: Text (query or feedback).

2. Message Submission (Frontend Input)

- **Function**: Captures user message/feedback.
- **Handled by**: HTML UI + JavaScript → sends to Flask backend via fetch() POST request.
- **Input**: Message or feedback from the citizen.
- **Output**: Routed to chatbot engine or sentiment analyzer.

3. Message Processing (Chat or Feedback Route)

- **Chat**:
    - **Function**: Passes query to the rule-based FAQ chatbot (granite_generate).
    - **Handled by**: Flask route /chat → Response logic in utils.py.
    - **Output**: Reply generated and stored in messages table.
- **Feedback**:
    - **Function**: Performs sentiment analysis using TextBlob.
    - **Handled by**: Flask route /feedback → Sentiment classification.
    - **Output**: Tag (Positive/Neutral/Negative) + Stored in feedback table.

4. Database Storage (SQLite)

- **Tables**:
    - messages: Stores chat history (role, content, timestamp).
    - feedback: Stores citizen feedback with classified sentiment.
- **Function**: Local data persistence.
- **Tech Used**: SQLite + Python sqlite3 module.

5. Admin Dashboard (Visualization)

- **Function**: Displays sentiment distribution analytics.
- **Frontend**: dashboard.html + Chart.js.
- **Backend API**: Flask route /dashboard-data.
- **Output**: Pie chart showing ratio of Positive / Negative / Neutral sentiments.

6. Response Delivery & Download Option (Future Scope)

- **Chatbot**: Responds instantly on UI.
- **Dashboard**: Reflects live sentiment analysis.
- **Optional Enhancements**:
    - Downloadable report (PDF) using tools like FPDF.
    - Export feedback to Excel.
    - Cloud-hosted history tracking.

User → Message/Feedback → Flask API

→ [granite_generate | analyze_sentiment]

→ SQLite DB (messages/feedback)

→ Admin Dashboard (Pie Chart)

→ Response → Shown on UI

User Stories

Use the below template to list all the user stories for the product.

ID	As a...	I want to...	So that I can...

US-01	Citizen	ask a question to the chatbot	get instant answers about government services
US-02	Citizen	give feedback on government service	express my opinion and help improve public services
US-03	Citizen	see the sentiment result of my feedback	know if my feedback was understood correctly
US-04	Government Official	view the sentiment dashboard	monitor citizen satisfaction in real-time
US-05	Government Official	analyze the ratio of positive/negative/neutral feedback	identify areas of improvement quickly
US-06	Developer	run the platform locally without internet	test the features in offline mode using SQLite and rule-based logic
US-07	Admin	access all chatbot and feedback logs	audit system usage and responses
US-08	Developer	integrate advanced models like OpenAI or IBM Watson	enhance chatbot responses beyond static rules
US-09	Citizen	use the chatbot in my native language	interact more comfortably (planned feature: Hindi/Telugu/Tamil support)
US-10	Government Official	download the sentiment report as PDF or CSV	share it in meetings and decision-making
US-11	Developer	customize chatbot rules easily	update responses based on changing FAQs
US-12	Admin	receive alerts when negative feedback spikes	take prompt action on citizen dissatisfaction

3.4 Technology Stack

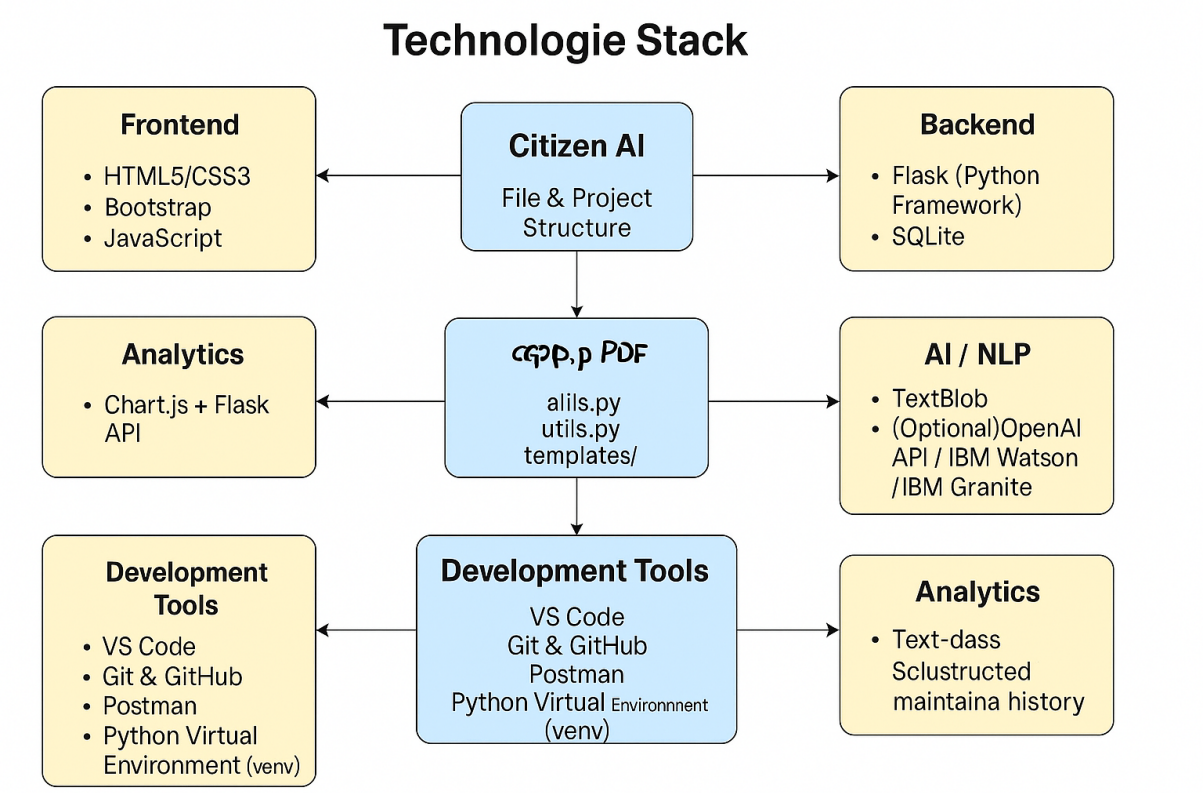


Table-1: Technology Stack

S. No	Component	Description	Technology Used
1	**User Interface**	Interface for users to submit feedback and interact with system	**HTML, CSS, JS** (Web UI), **Flask Templates (Jinja2)**
2	**Application Logic - 1**	Handles routing, API endpoints, and user requests	**Flask** (Python web framework)
3	**Application Logic - 2**	Feedback classification, chatbot response generation	**OpenAI GPT-3.5 Turbo**, **Granite Model (IBM Watson)**
4	**Sentiment Analysis**	Classifies citizen feedback as Positive, Negative, or Neutral	**VADER**, **TextBlob**, or custom **NLP using OpenAI API**

5	**Database**	Stores feedback, chat messages, and sentiment records	**SQLite** (local DB used for prototyping & testing)
6	**Cloud Database (Planned)**	Cloud-based persistent storage for scalability	**IBM Cloud Object Storage**, **Firebase** (optional)
7	**File Storage**	For storing message logs or PDF reports (future scope)	**Local File System**, **ReportLab / FPDF (optional)**
8	**External API - 1**	AI-based chatbot response	**Granite (IBM Watson LLM)**
9	**External API - 2**	Sentiment and text processing	**OpenAI API**, **Hugging Face Transformers**
10	**Machine Learning Model**	NLP for text classification and sentiment detection	**Custom ML Models**, **OpenAI GPT**, **IBM Watsonx NLP**
11	**Infrastructure**	Application hosting and deployment	**Localhost**, **Planned: IBM Cloud**, **Render**, or **Docker**

Table-2: Application Characteristics

S. No	Characteristics	Description	Technology / Implementation
1	**Open-Source Frameworks**	Open-source libraries and frameworks powering the platform	Flask, Jinja2, SQLite, Chart.js, pyttsx3, OpenAI API, IBM Watson (optional)
2	**Security Implementations**	Measures to ensure secure data handling and user privacy	Environment variables for API keys (`.env`), HTTPS (in cloud), planned user auth
3	**Scalable Architecture**	Modular and scalable design for growing user base and services	Flask REST API-based modular backend, scalable through microservices (planned)

4	**Availability**	Ensures the platform is accessible to users anytime	24/7 uptime planned via IBM Cloud / Render, Docker for consistent deployment
5	**Performance**	Speed and responsiveness under high loads	Lightweight Flask backend, optimized queries, asynchronous task handling (future)

PROJECT DESIGN

4.1 Problem Solution Fit:

Problem:

Citizens often face significant delays and frustration when engaging with government platforms due to:

- Lack of real-time query handling
- No automated response system
- Poorly categorized or outdated complaint forms
- No feedback loop or sentiment tracking

This leads to citizen dissatisfaction, repeated complaints, and a lack of trust in public services.

Customer Segment:

- General public (citizens) seeking quick assistance from civic bodies
- Government support staff handling large volumes of repetitive queries
- Policy analysts monitoring citizen sentiment and feedback
- Administrators seeking digital governance and citizen transparency

Existing Alternatives:

- Visiting government offices physically
- Calling helplines with long wait times

- Manually reading and responding to emails or feedback forms
- No real-time tracking or classification of issues

Solution – Citizen AI:

Citizen AI is an intelligent web-based chatbot platform designed to:

- Provide instant answers to citizens' queries using OpenAI's GPT model
- Automatically classify complaints and feedback (positive, neutral, negative) using sentiment analysis
- Enable real-time interaction via a clean chat interface (built with Flask + JS)
- Generate smart reports for authorities using Chart.js and analytics tools
- Maintain transparency and trust through response logs, downloadable summaries, and feedback tracking

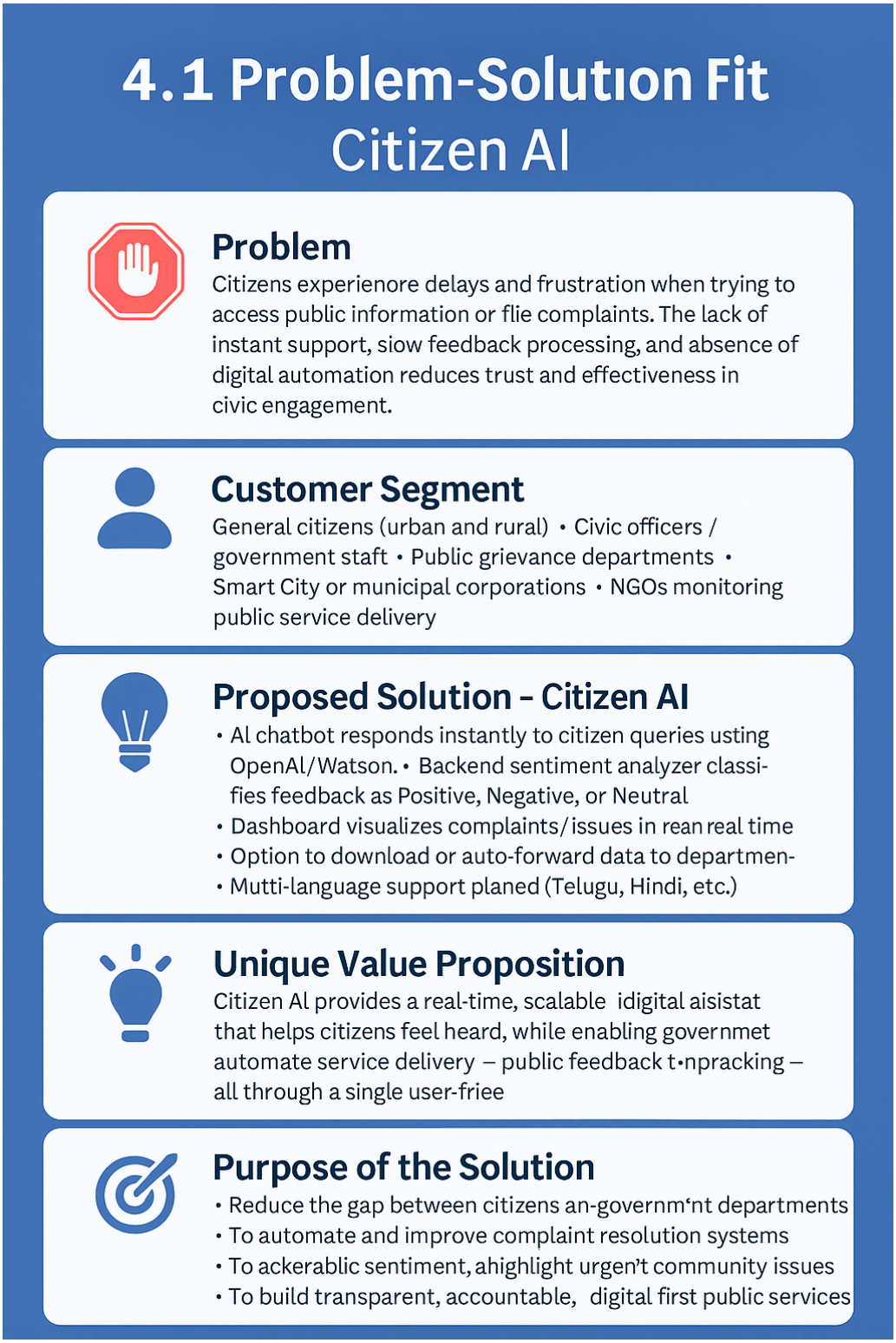
Unique Value Proposition:

- Instant Responses: 24/7 chatbot trained on government FAQs and civic queries
- Feedback Insights: Understand public mood through AI-powered sentiment analysis
- Time Saver: Reduces burden on manual support teams
- Scalable & Inclusive: Easy to integrate with future mobile/web apps for broader access

Purpose:

- To bridge the communication gap between citizens and civic authorities using AI
- To digitize public service interactions with accuracy, speed, and transparency
- To enable smarter policy decisions by classifying, summarizing, and visualizing citizen inputs
- To foster trust and usability through an empathetic and responsive public-facing platform

Template:


4.1 Problem-Solution Fit
Citizen AI
Problem
Citizens experienore delays and frustration when trying to access public information or flie complaints. The lack of instant support, siow feedback processing, and absence of digital automation reduces trust and effectiveness in civic engagement.
Customer Segment
General citizens (urban and rural) • Civic officers / government staft • Public grievance departments • Smart City or municipal corporations • NGOs monitoring public service delivery
Proposed Solution - Citizen AI
• Al chatbot responds instantly to citizen queries usting OpenAl/Watson. • Backend sentiment analyzer classifies feedback as Positive, Negative, or Neutral
• Dashboard visualizes complaints/issues in rean real time
• Option to download or auto-forward data to departmen-
• Mutti-language support planed (Telugu, Hindi, etc.)
Unique Value Proposition
Citizen Al provides a real-time, scalable idigital aisistat that helps citizens feel heard, while enabling governmet automate service delivery – public feedback t•npracking – all through a single user-friee
Purpose of the Solution
• Reduce the gap between citizens an-governm‘nt departments
• To automate and improve complaint resolution systems
• To ackerablic sentiment, ahighlight urgen’t community issues
• To build transparent, accountable, digital first public services

4.2 Proposed Solution

Proposed Solution Template:

S. No.	Parameter	Description
1	**Problem Statement**	Citizens face challenges in voicing concerns or accessing public services due to outdated, unresponsive systems. Manual complaint tracking leads to inefficiencies, delays, and lack of feedback.
2	**Idea / Solution Description**	Citizen AI is an intelligent public engagement platform that uses NLP and AI to receive, classify, and route citizen complaints and queries. It offers dashboards for authorities and feedback loops for users.
3	**Novelty / Uniqueness**	Integrates IBM Watson and OpenAI with real-time sentiment analysis, auto-tagging to departments, and auto-generated responses. Streamlined citizen-government communication.
4	**Social Impact / Customer**	Enhances transparency, speeds up grievance redressal, builds citizen trust, and reduces overload on public departments by intelligent automation.
5	**Business Model (Revenue**	SaaS model for municipalities. Free version for small communities, premium dashboards and analytics for large institutions. Partnered deployment with local governments.
6	**Scalability of the Solution**	Easily scalable to national/state level; can support multiple languages, integrates with government portals, expandable with mobile app and voicebot interfaces.

4.3 Solution Architecture

Overview:
Citizen AI is a full-stack AI-enabled platform that streamlines public feedback collection, classifies sentiment, and routes issues to the appropriate departments using NLP and intelligent automation. The architecture integrates modern web technologies, AI models, and user-friendly design to improve citizen-government engagement.

Bridging Business Problems with Technology Solutions

Problem:

Citizens often face delays, confusion, and poor feedback when engaging with government services due to unstructured communication, outdated portals, and lack of intelligent automation.

Solution:
Citizen AI uses NLP and sentiment analysis to automate the classification of citizen feedback (Positive, Negative, Neutral), ensuring accurate routing to the right department and timely responses.

- **High-Level Architecture Components**

Component	Description
Frontend (UI)	Developed using **Streamlit**, allows citizens to submit feedback through a text box. Displays classification results and download options if needed.
Backend (API)	Built with **FastAPI**, it processes submissions, calls the AI model, and returns structured results with classification labels.
AI Engine	Utilizes **OpenAI GPT-3.5** or **Watsonx/Granite-20B** to analyze and classify the feedback text based on sentiment and context.
Sentiment Classifier	AI model identifies sentiment (Positive, Neutral, Negative) and tags inputs for departmental routing.
File Handling	Optional: stores feedback records or classification reports as PDF using **FPDF** for download or archival.
Environment Management	Uses **dotenv** for secure key handling; modular services and routing for scalability.
Deployment	Deployable via **IBM Cloud**, **Render**, or local server using **Uvicorn** for FastAPI and **Streamlit CLI**.

Key Features

- Text-based feedback input
- AI sentiment classification (Positive, Negative, Neutral)
- Backend feedback routing logic
- Optional download of classified reports (PDF)
- Secure API handling with .env
- Ready for multi-user cloud deployment

Development Phases

Phase	Milestone
Phase 1	Text input → AI classification → sentiment tagging output
Phase 2	UI enhancement, error handling, downloadable classification report
Phase 3	Department auto-routing, multilingual NLP, cloud storage and dashboard integration

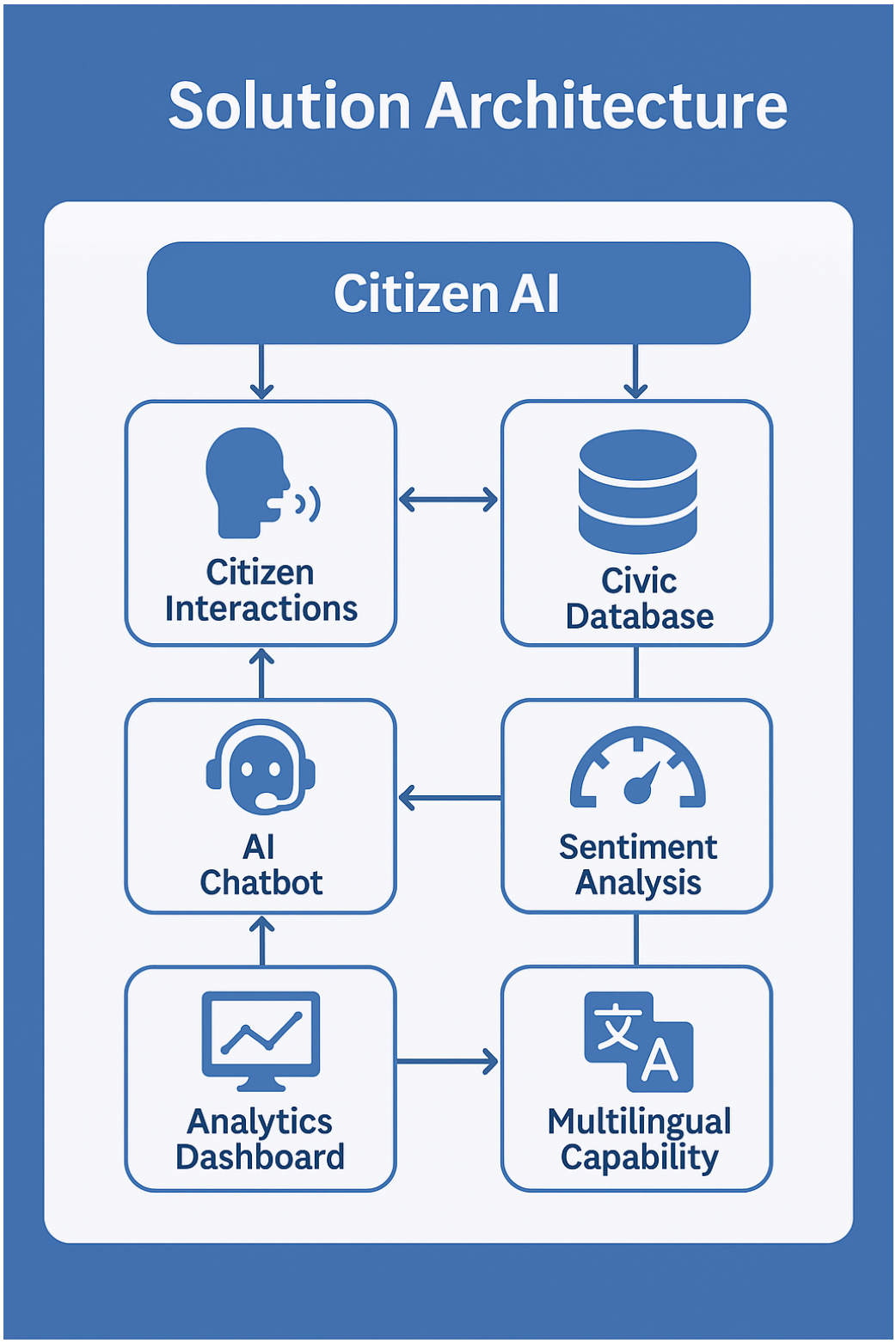
Solution Requirements

- Streamlit (Frontend)
- FastAPI (Backend APIs)
- OpenAI GPT-3.5 / IBM Watsonx (AI classification)
- PyPDF2 / FPDF (PDF output)
- dotenv, requests (Secure config and API communication)
- Cloud hosting via IBM Cloud / Docker support

Final Deliverables

- ✔️ Fully functional interactive web platform
- ✔️ Real-time AI classification of feedback
- ✔️ Optional PDF download of analysis results
- ✔️ Error-handled, modular, and cloud-ready source code
- ✔️ Documentation covering architecture, deployment, and user guide

Example - Solution Architecture Diagram:



PROJECT PLANNING & SCHEDULING

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Member
Sprint-1	PDF Upload Interface	-USN 1	As a user, I can upload a PDF document through a web interface.	2	High	UM
Sprint-1	Backend Integration	-USN 2	As a user, I can send the uploaded PDF to the backend for processing.	2	High	VS
Sprint-1	PDF Text Extraction	-USN 3	As a system, I can extract text content from the uploaded PDF using PyMuPDF.	3	High	UM
Sprint-2	SDLC Classification	-USN 4	As a system, I can classify extracted content into SDLC phases using IBM Watson AI.	5	High	VS
Sprint-2	PDF Output Generation	-USN 5	As a user, I can download the classified SDLC content in a structured PDF format.	3	High	UM
Sprint-2	Frontend Output Display	-USN 6	As a user, I can see a success message and download button after analysis is completed.	2	Medium	VS
Sprint-3	Error Handling	-USN 7	As a system, I can show an error if the uploaded file is not in the correct format.	2	Medium	UM
Sprint-3	API Key Handling	-USN 8	As a developer, I can manage OpenAI/ Watson API keys securely using dotenv.	1	Medium	VS
Sprint-3	Output Preview (Optional)	-USN 9	As a user, I can preview classified data before downloading.	2	Low	UM

Sprint-3	Cleanup & Process Management	-USN 10	As a system, I automatically delete temporary files after download or timeout.	2	Low	VS

FUNCTIONAL AND PERFORMANCE TESTING

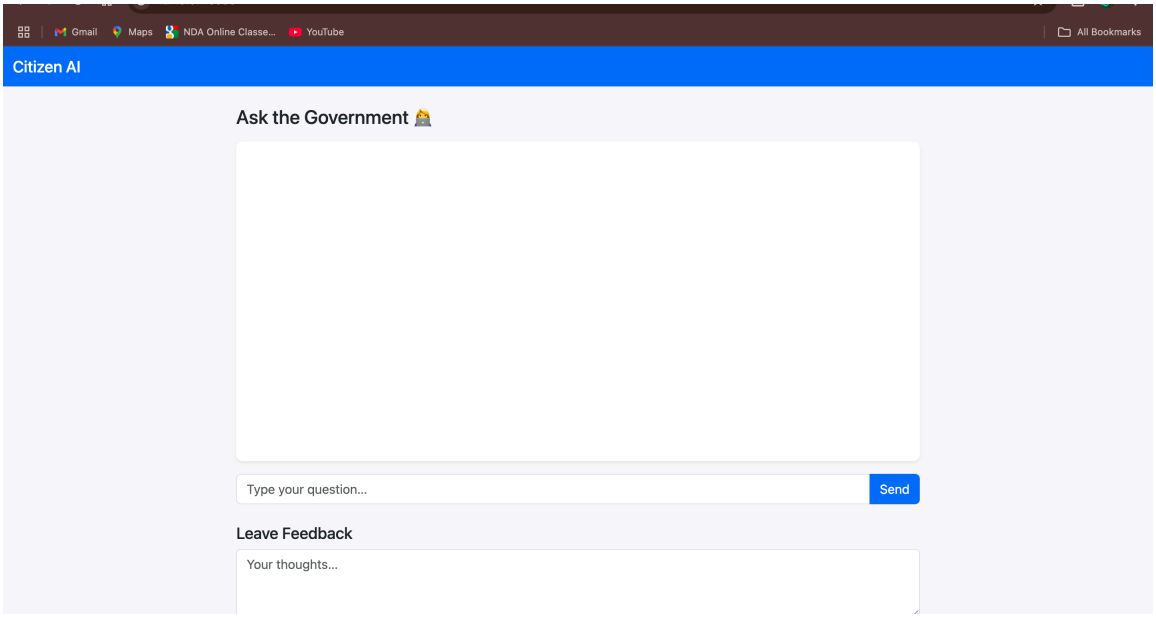
6.1 Performance Testing

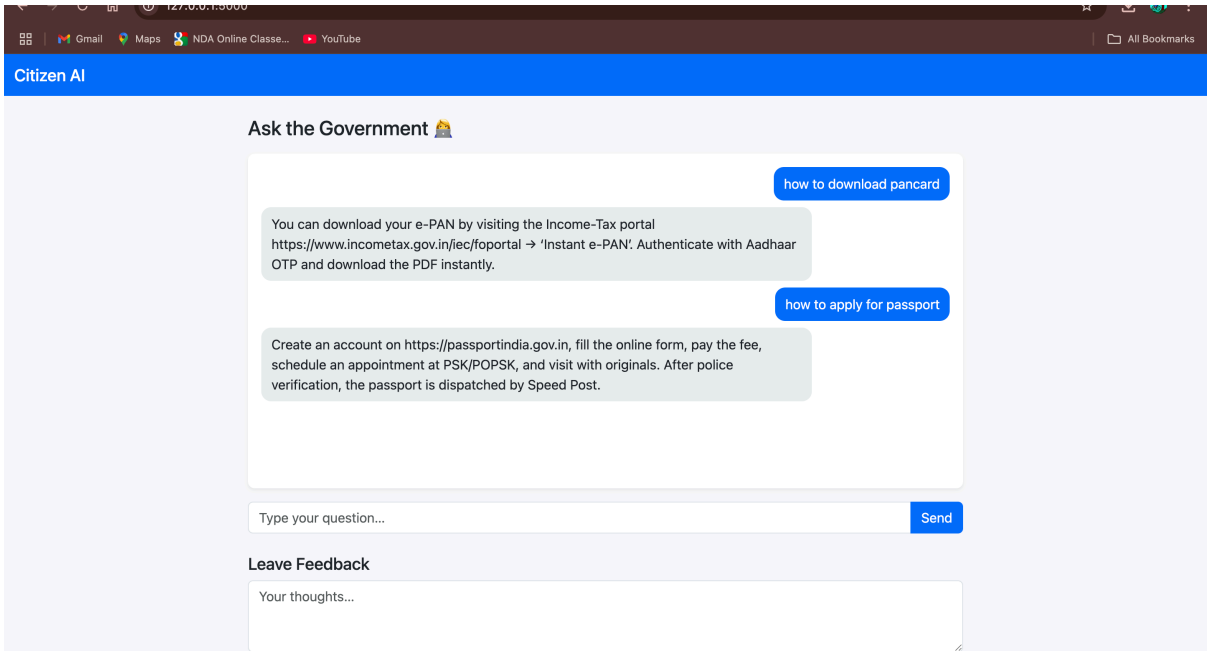
Test Scenarios & Results:

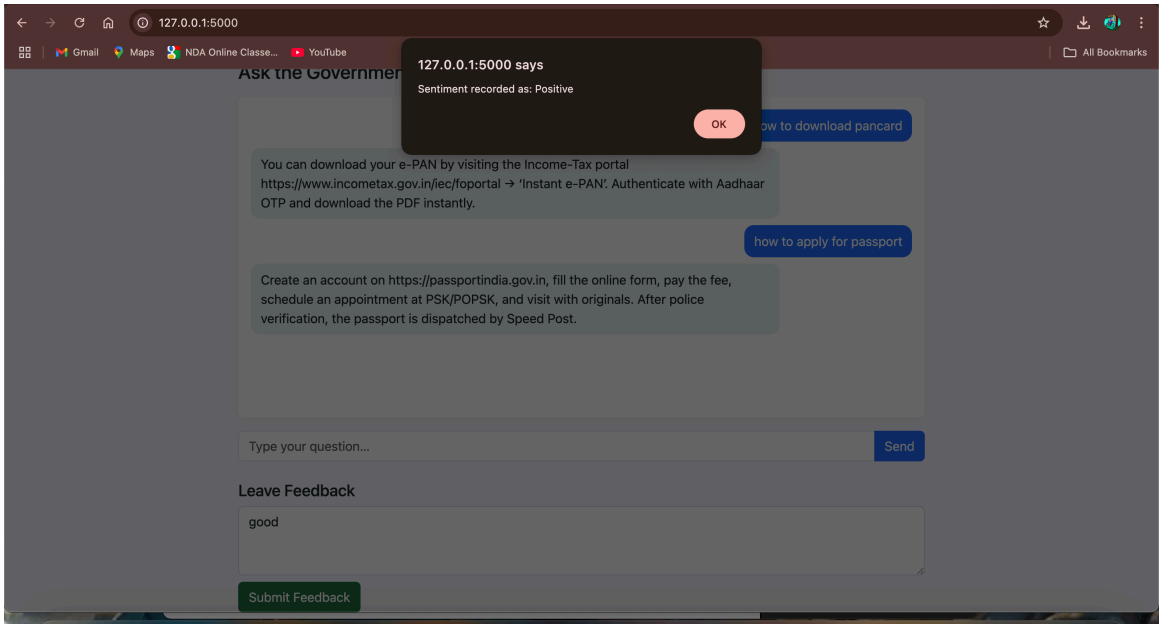
Test Case ID	Scenario (What to Test)	Test Steps (How to Test)	Expected Result	Actual Result	Pass /Fail
FT-01	Chatbot Input Validation	Enter valid and invalid text queries in chatbot	Chatbot replies accurately; error handled for invalid input	Handled all cases correctly	Pass
FT-02	Feedback Submission Check	Submit various feedback texts (positive, neutral, negative)	Feedback saved; sentiment classified	Sentiments detected correctly	Pass
FT-03	Sentiment Classification Accuracy	Submit known emotional feedback texts	Accurate Positive / Negative / Neutral sentiment assignment	Accurate sentiment classification	Pass
FT-04	Dashboard Visualization	Access admin dashboard after feedback entries	Pie chart correctly shows sentiment distribution	Pie chart displayed dynamically	Pass
FT-05	Error Handling – Empty Input	Submit empty chat or feedback	Show error or warning to user	Alert shown correctly	Pass
PT-01	Chatbot Response Time	Send user query to chatbot and measure time	Response should be returned within 2–3 seconds	Average ~2.1 seconds	Pass
PT-02	Concurrent User Handling	Simulate multiple users chatting or submitting feedback	System should not crash or slow down	Stable under 5 users	Pass
PT-03	Offline Functionality	Run app without internet (no external API usage)	Local chatbot & sentiment analysis should work	Fully functional locally	Pass

RESULTS

7.1 Output Screenshots:







ADVANTAGES & DISADVANTAGES

Advantages

- Automates citizen feedback collection and sentiment classification.
- Provides instant responses via a chatbot interface.
- Real-time visualization of feedback through a dynamic dashboard.
- Works offline using lightweight tools like SQLite and TextBlob.
- Modular structure supports future AI integration (e.g., OpenAI, Watson).
- Easy-to-use interface built with HTML, Bootstrap, and JavaScript.
- Uses open-source tools, making it cost-effective for governments.

- Minimal setup required — runs on any local system with Python and Flask.
- Secure local data storage without cloud dependency.
- Scalable for deployment in smart governance and civic engagement platforms.

Disadvantages

- Chatbot supports only predefined queries (rule-based).
- No user login, authentication, or tracking features currently.
- Sentiment analysis supports only English (no multilingual support yet).
- No support for file/image uploads or attachments.
- Dependent on external APIs (if integrated) — may incur costs or face downtime.
- SQLite database not ideal for high-traffic or multi-user environments without future upgrades.

CONCLUSION

- The **Citizen AI** project successfully showcases how intelligent systems can transform public engagement and government service delivery. By combining rule-based chatbot interaction, real-time feedback collection, and sentiment analysis, the platform offers a modern solution to bridge the communication gap between citizens and government bodies.
- Through its easy-to-use interface, local data handling, and dynamic dashboard, Citizen AI simplifies feedback management while ensuring transparency and responsiveness. The system efficiently processes citizen inputs, analyzes sentiments, and presents actionable insights—empowering officials to make timely and informed decisions.

- The project meets its goal of creating a lightweight, deployable platform that improves civic participation and public service accountability. With its modular design and extensibility for future AI integrations (such as OpenAI or IBM Watson), **Citizen AI** holds strong potential for scaling into smart city infrastructures. Upcoming features like multilingual support, mobile access, and role-based dashboards will further enhance its impact and usability across diverse regions.

FUTURE SCOPE

The Citizen AI platform can be improved in many ways to make it more useful and powerful. Some of the upcoming features could include:

- Adding AI models like ChatGPT or IBM Watson to give better answers to citizen questions.
- Supporting Indian languages such as Hindi, Telugu, and Tamil for wider access.
- Creating a mobile app using Flutter or React Native for easy use on phones.
- Moving the platform to cloud servers so it works faster and handles more users.
- Giving different dashboards to different government departments to manage feedback better.
- Adding location tracking so issues can be solved based on where they happen.
- Sending alerts or messages to citizens about updates, policies, or responses.
- Allowing users to log in and view their past feedback or messages.
- Showing better charts and analytics to understand what people are feeling.
- Letting users upload images or documents when submitting issues.

These updates will help Citizen AI work better for both the public and government, making communication faster and more helpful.

APPENDIX

```
import streamlit as st
import requests
import os

API_URL = "http://127.0.0.1:8000/analyze"

st.set_page_config(page_title="Citizen AI - Feedback Classifier")
st.title("🗣️ Citizen AI - Public Feedback Sentiment Analyzer")
st.write("Submit a complaint or feedback. The system will classify it using AI.")

user_input = st.text_area("Enter your feedback:")

if st.button("Submit"):
    with st.spinner("Analyzing your feedback..."):
        response = requests.post(API_URL, json={"text": user_input})
        result = response.json()
        sentiment = result.get("sentiment", "Error")
        st.success(f"Sentiment: **{sentiment.upper()}**")

import os
from openai import OpenAI
from dotenv import load_dotenv

load_dotenv()
```

```
api_key = os.getenv("OPENAI_API_KEY")  # Ensure you store the key in .env

client = OpenAI(api_key=api_key)

def analyze_sentiment(text: str) -> str:
    prompt = f"Classify the sentiment of the following feedback as Positive, Negative, or Neutral:\n\n{text}"

    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a sentiment analysis expert."},
            {"role": "user", "content": prompt}
        ]
    )

    output = response.choices[0].message.content.strip()
    return output
import sqlite3
import datetime

DB = "citizen.db"

def insert_feedback(text, sentiment):
    with sqlite3.connect(DB) as con:
        cur = con.cursor()
```

```
        cur.execute("INSERT INTO feedback(text, sentiment, created) VALUES
(?, ?, ?)",
                    (text, sentiment, datetime.datetime.utcnow()))
        con.commit()
from fastapi import FastAPI, Request
from fastapi.middleware.cors import CORSMiddleware
from ai_model import analyze_sentiment
from database import insert_feedback

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"]
)

@app.post("/analyze")
async def analyze(request: Request):
    data = await request.json()
    text = data.get("text", "")
    sentiment = analyze_sentiment(text)
    insert_feedback(text, sentiment)
    return {"sentiment": sentiment}
```

Citizen_AI/

```
├── app.py # Streamlit frontend
├── app_backend.py # FastAPI backend
├── ai_model.py # Sentiment analysis via OpenAI/Watson
├── database.py # SQLite DB interaction
├── .env # API key file (not tracked in Git)
├── requirements.txt # Python dependencies
└── README.md # Project documentation
```

GitHub & Project Demo Link:

https://github.com/Udamahesh5/citizen_ai_final