# Citizen AI- Intelligent Citizen Engagement Platform

# 1.Introduction

In an era where technology is rapidly reshaping governance and public service delivery, **CitizenAI – Intelligent Citizen Engagement Platform** emerges as a smart solution aimed at bridging the communication gap between citizens and government authorities. This project leverages **Generative AI** and **Natural Language Processing (NLP)** to empower everyday citizens to report civic issues, seek information about government services, and receive AI-generated responses—all in real time.

Built using **Google Colab** and **Gradio**, CitizenAI offers an accessible, interactive, and user-friendly interface that enables users to engage with government systems effortlessly. Whether it is lodging complaints about streetlights, water supply, road maintenance, or accessing details about welfare schemes, the platform ensures that citizens are heard and served efficiently.

The primary goal of this platform is to **increase transparency**, **enhance citizen participation**, and **reduce bureaucratic delays** by providing a smart, AI-driven assistant that interacts in natural language and offers context-aware support tailored to local governance needs.

## 1.1 Project Overview

**CitizenAI – Intelligent Citizen Engagement Platform** is an AI-powered application designed to streamline communication between citizens and local government bodies. Built using **Google Colab** and **Gradio**, the platform enables users to interact with a smart assistant that responds to queries about public services and facilitates the reporting of civic issues such as potholes, malfunctioning streetlights, water shortages, and more.

The solution leverages **Natural Language Processing (NLP)** to understand user queries in plain language and deliver relevant, real-time responses. The main aim is to enhance civic engagement by reducing bureaucracy, promoting transparency, and creating an intelligent feedback loop between the public and administrators.

### 1.2 Purpose

The primary purpose of this project is to:

* **Empower citizens** by giving them a simple and intelligent tool to access government services and register complaints.
* **Promote digital governance** by integrating AI in civic communication systems.
* **Enhance user experience** by replacing complex forms and delayed responses with a conversational interface.
* **Bridge the communication gap** between citizens and government departments through instant, AI-driven interaction.

# 2.Ideation Phase

### Define the Problem Statements

|  |  |
| --- | --- |
| Date | 21 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 2 Marks |

### Customer Problem Statement Template:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.



### Example:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem**  **Statemen**  **t (PS)** | **I am**  **(Customer**  **)** | **I’m trying**  **to** | **But** | **Because** | **Which makes me**  **feel** |
| PS-1 | A working profession  al | Report a path hole in  my way | I don’t know where to  report | Governme  nt sites are confusing and slow | Frustrated and ignored |
| PS-2 | A student | Get informatio  n about  available scholarship  s | The process is too complicate d online | Informatio n is spread across websites | Confused and discouraged |
| PS-3 | A retired senior citizen | Learn about pension schemes | I struggle to use  mobile  apps or  websites | They are not user-  friendly for elders | Disconnecte  d and  anxious |

### Ideation Phase

### Empathize & Discover

|  |  |
| --- | --- |
| Date | 21 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 4 Marks |

### Empathy Map Canvas:

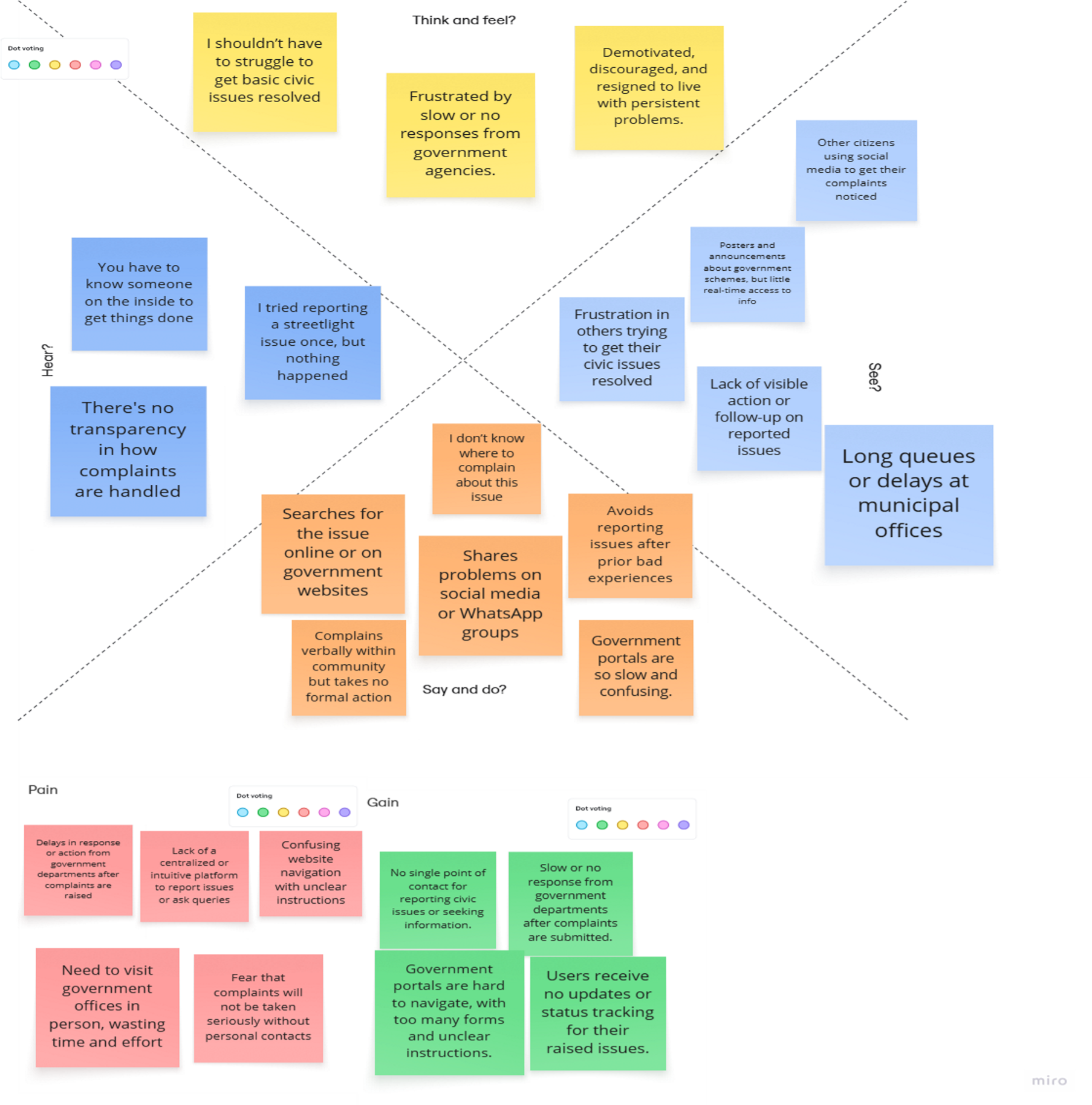
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.

It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.

### Example:

### Example: Citizen AI



### Ideation Phase

### Brainstorm & Idea Prioritization Template

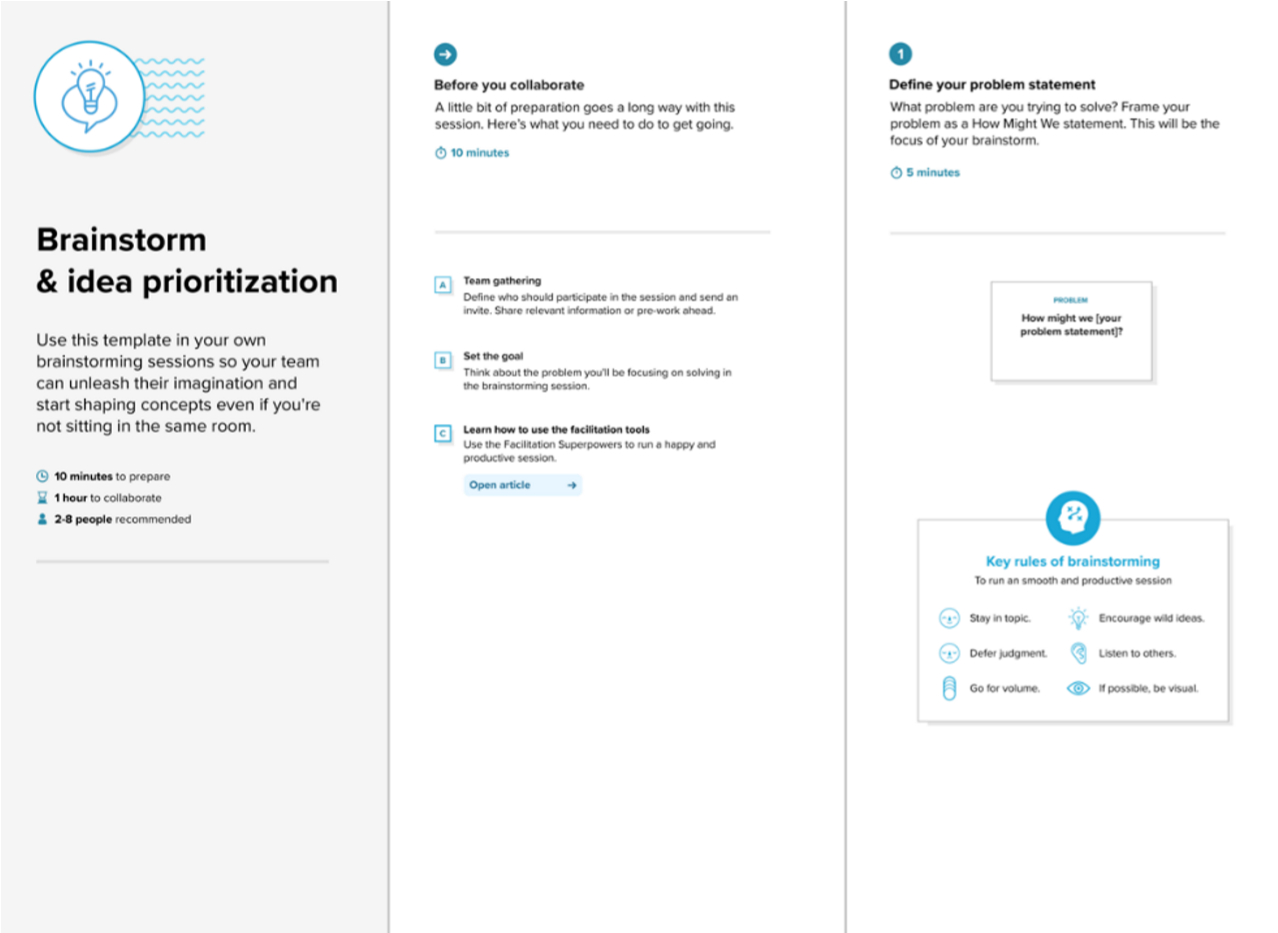
|  |  |
| --- | --- |
| Date | 22 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 4 Marks |

### Brainstorm & Idea Prioritization Template:

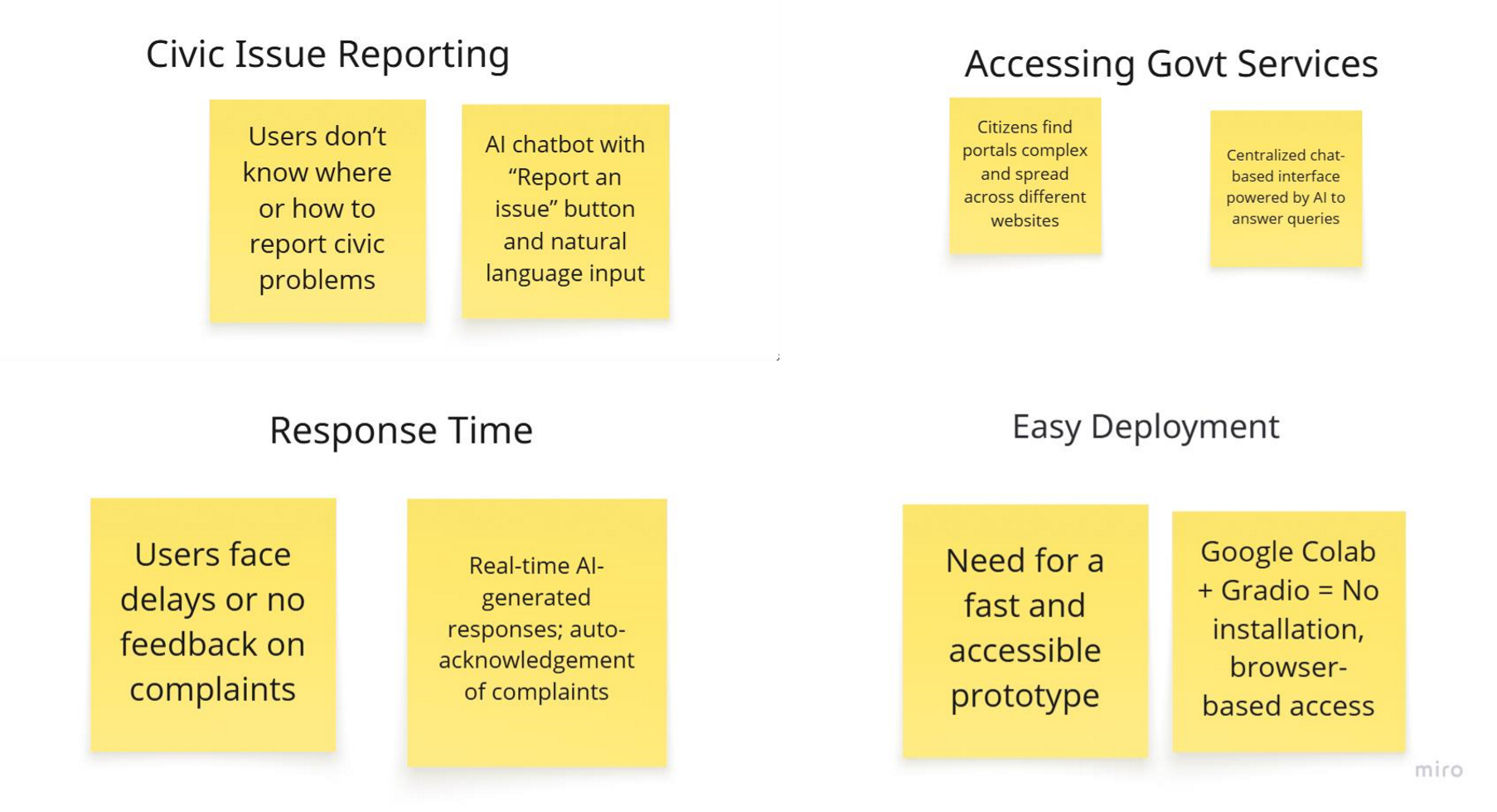
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

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



### Step-2: Brainstorm, Idea Listing and Grouping



### Step-3: Idea Prioritization



|  |  |
| --- | --- |
| Date | 23 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 4 Marks |

# 3.REQUIREMENT ANALYSIS

3.1 CUSTOMER JOURNEY MAP



3.2 SOLUTION REQUIREMENTS

### Project Design Phase-II

### Solution Requirements (Functional & Non-functional) Functional Requirements:

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement**  **(Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Chat-based Citizen Interaction Interface | * Develop a Gradio interface to handle user input and output in chat format * Support free-text natural language queries * Display welcome message, prompts, and fallback responses |
| FR-2 | Issue Reporting & Categorization | * Enable reporting of civic issues like potholes, water problems, streetlight * Automatically categorize the type of complaint using NLP |
| FR-3 | Information Delivery About Government Services | * Allow users to ask questions about schemes, eligibility, and required documents * Provide AI-generated responses based on trained or integrated datasets |
| FR-4 | Response Generation & Confirmation | - Display instant, AI-generated responses in a human-like tone |

### Non-functional Requirements:

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

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-  1 | **Usability** | The platform must have a simple and intuitive user interface (Gradio), ensuring ease of use for all age groups including non-tech-savvy users. |
| NFR-  2 | **Security** | User data (queries, reports) must be handled with privacy and care. Any future storage of data must comply with data protection standards. |
| NFR-  3 | **Reliability** | The system should consistently provide correct, AI-generated responses and not crash during user interaction. |
| NFR-  4 | **Performance** | The chatbot should respond to queries within 2– 3 seconds to maintain a smooth conversational experience. |
| NFR-  5 | **Availability** | The system should be accessible 24/7 via the hosted Gradio interface (as long as the backend is active). |
| NFR-  6 | **Scalability** | The platform should support transition from prototype (Gradio on Colab) to production (FastAPI, cloud deployment) with higher user load. |

### Project Design Phase-II

### 3.3Data Flow Diagram & User Stories

|  |  |
| --- | --- |
| Date | 24 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 4 Marks |

### Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.





### User Stories

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional**  **Requirement**  **(Epic)** | **USN** | **User Story / Task** | **Acceptance criteria** | **P** |
| Citizen  (Web/Mobile) | Report  Civic Issue | USN-1 | As a citizen, I can report a civic issue by typing in my complaint in natural language. | I can submit an issue and receive a confirmation instantly. |  |
| Citizen  (Web/Mobile) | Ask Govt.  Service Info | USN-2 | As a citizen, I can ask questions about schemes or services. | I get accurate AIgenerated answers for my query. |  |
| Citizen  (Web/Mobile) | Follow-up Questions | USN-3 | As a citizen, I can ask follow-up queries in . | I can continue the conversation without restarting it. |  |
| Citizen  (Web/Mobile) | Language  Support  (Future) | USN-4 | As a citizen, I want to interact in my local language. | I can use Telugu/Hindi for questions and receive responses. | L |
| Citizen  (Web/Mobile) | Feedback Submission | USN-5 | As a user, I want to leave a rating or comment after using the platform. | I see a short feedback prompt at the end of my session. |  |

H

H

M

M

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional**  **Requirement**  **(Epic)** | **USN** | **User Story / Task** | **Acceptance criteria** | **P** |
| Admin | View Complaint Analytics | USN-6 | As an admin, I want to view total complaints by category and area. | I see a dashboard showing visual stats by issue type/location. |  |
| Admin | Export  User  Queries | USN-7 | As an admin, I want to export all queries | As an admin, I want to export all queries submitted by users. |  |
| Support Executive | Respond to  Escalated Issues | USN-8 | As a customer care executive, I want to view and respond to escalated complaints. | I can access issue details and add comments or mark them resolved. |  |
| Support Executive | View Chat History | USN-9 | As a support user, I want to view past userchat history for context. | I can view previous chats linked to a user or issue. |  |

H

M

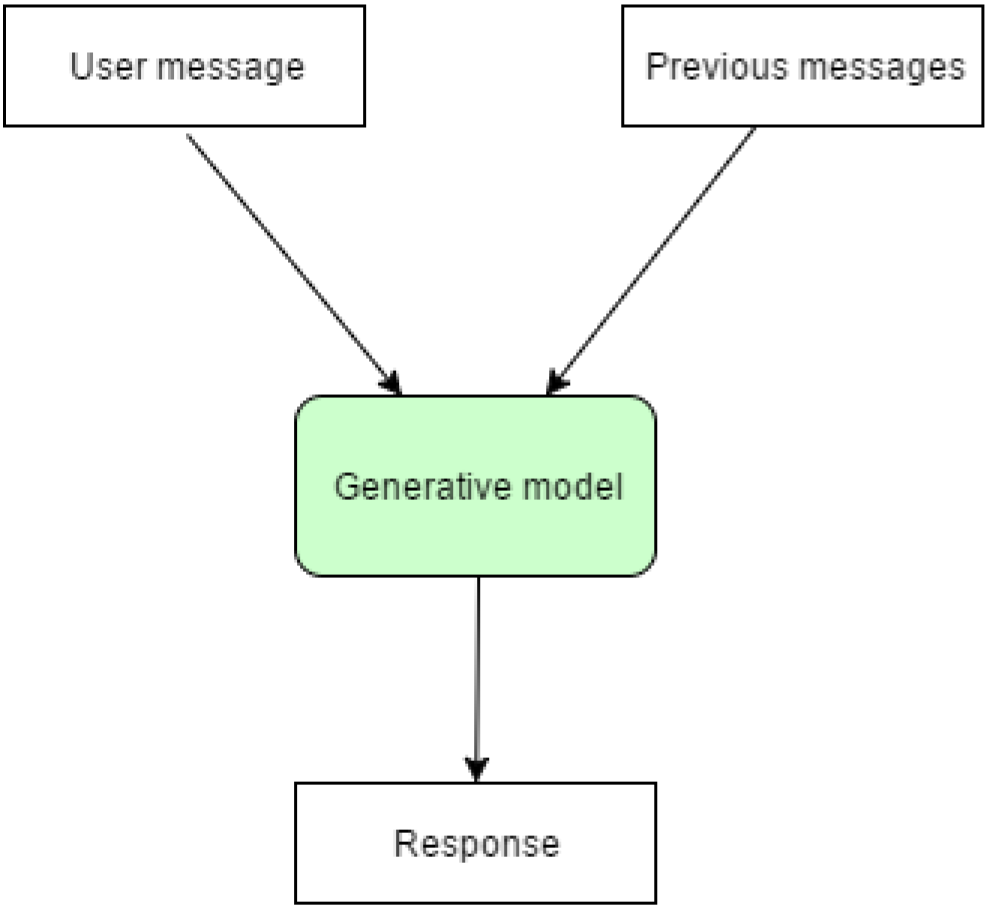
H

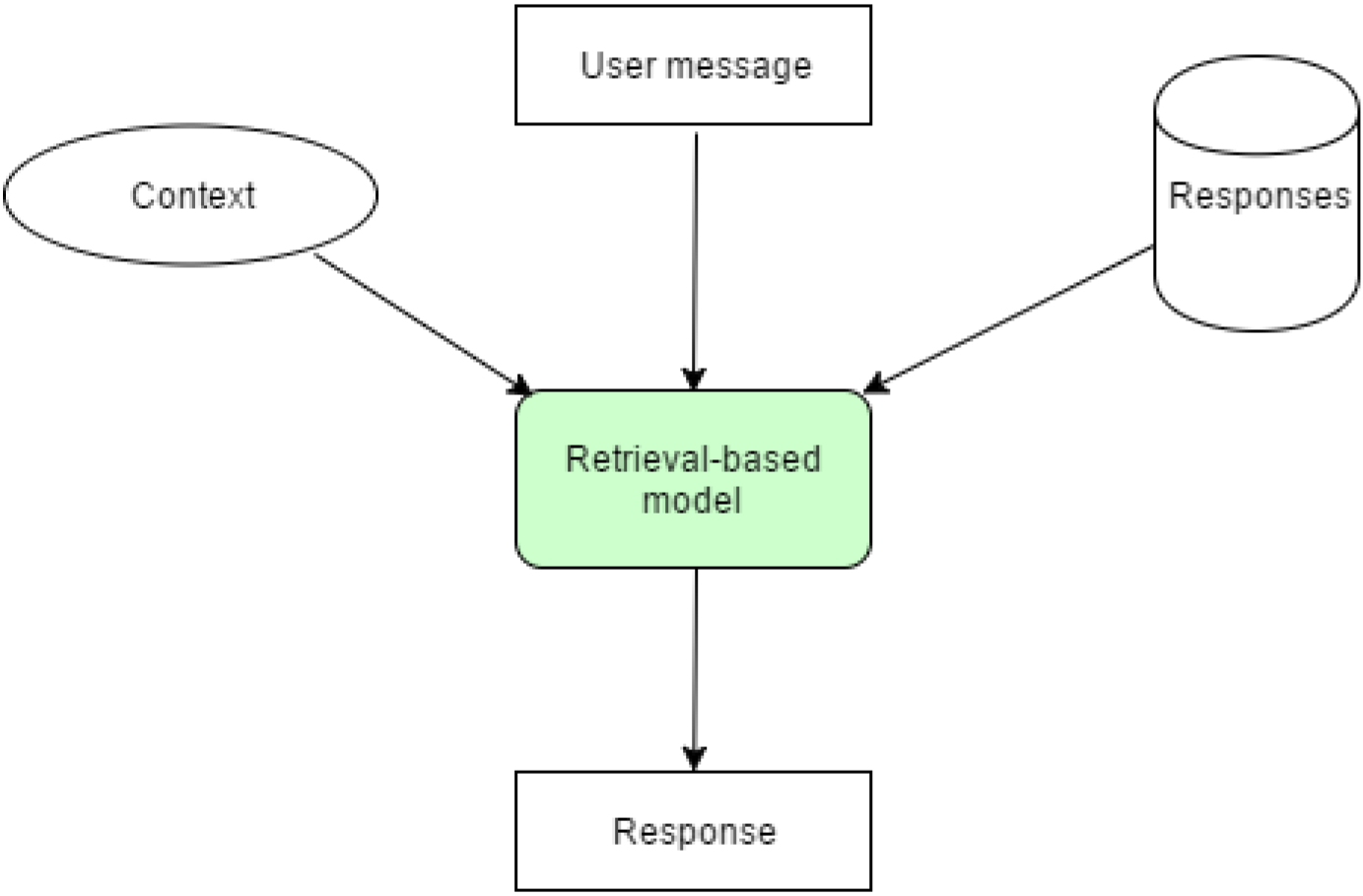
M

### Project Design Phase-II

### 3.4Technology Stack (Architecture & Stack)

|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks | 4 Marks |





### Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1. | User Interface | Chatbot interface for user interaction (issue reporting,  Q&A). | **Gradio**, HTML, CSS. |
| 2. | Application  Logic-1 | Handles routing, query preprocessing, response formatting | **Python**, FastAPI |
| 3. | Application  Logic-2 | AI model interaction and natural language understanding | **IBM Granite 3.3**,  Hugging Face Transformers |
| 4. | Application  Logic-3 | Query classification and response generation logic | Python-based logic & custom intents |
| 5. | Database | Optional storage for FAQs, schemes, issue categories | JSON files / SQLite  (optional) |
| 6. | Cloud Database | For storing user queries, feedback, issue logs (future scope) | IBM Cloudant,  Firebase (optional) |
| 7. | File Storage | Storing logs, screenshots (if uploaded), or static JSON  files | Local Filesystem /  IBM Object Storage  (future) |
| 8. | External API-1 | To fetch area-specific government service information (future) | e-Seva / RTI APIs (future integration) |
| 9. | External API-2 | (Optional) To verify user or connect to citizen services | Aadhar API, etc. |
| 10. | Machine  Learning Model | Understand and respond to user queries in natural language | IBM Granite Model /  Fine-tuned  Transformers |
| 11. | Infrastructure  (Server / Cloud) | Deployed via Google  Colab; can migrate to IBM  Cloud or local host | **Google Colab**, IBM  Cloud, Local/VM,  Docker |

### Table-2: Application Characteristics:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | Frameworks and tools used to build the platform | Gradio (Python),  FastAPI, Hugging  Face Transformers |
| 2. | Security  Implementations | Application-level security  (basic for now, expandable) | Token-based access, HTTPS (when  deployed), basic auth Future: OAuth2, IAM,  OWASP practices |
| 3. | Scalable  Architecture | Modular backend, pluggable AI, potential for microservices and API gateways | Microservices-  friendly: FastAPI + AI Models separated |
| 4. | Availability | Can be deployed to cloud, supports scaling through containerization and serverless platforms | IBM Cloud, Docker,  Cloud Foundry  (optional) |
| 5. | Performance | Optimized for fast inference using lightweight models, Gradio sessions cached | GPU-enabled Colab,  Caching in FastAPI (future), Preloaded responses |

# 4.Project Design Phase

### Project Design Phase

### Project Design Phase Problem – Solution Fit Template

|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | citizen ai – intelligent citizen engagement platform |
| Maximum Marks | 2 Marks |

### Problem – Solution Fit Template:

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer’s problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why **Purpose:**

❑ Solve complex problems in a way that fits the state of your customers.

❑ Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.

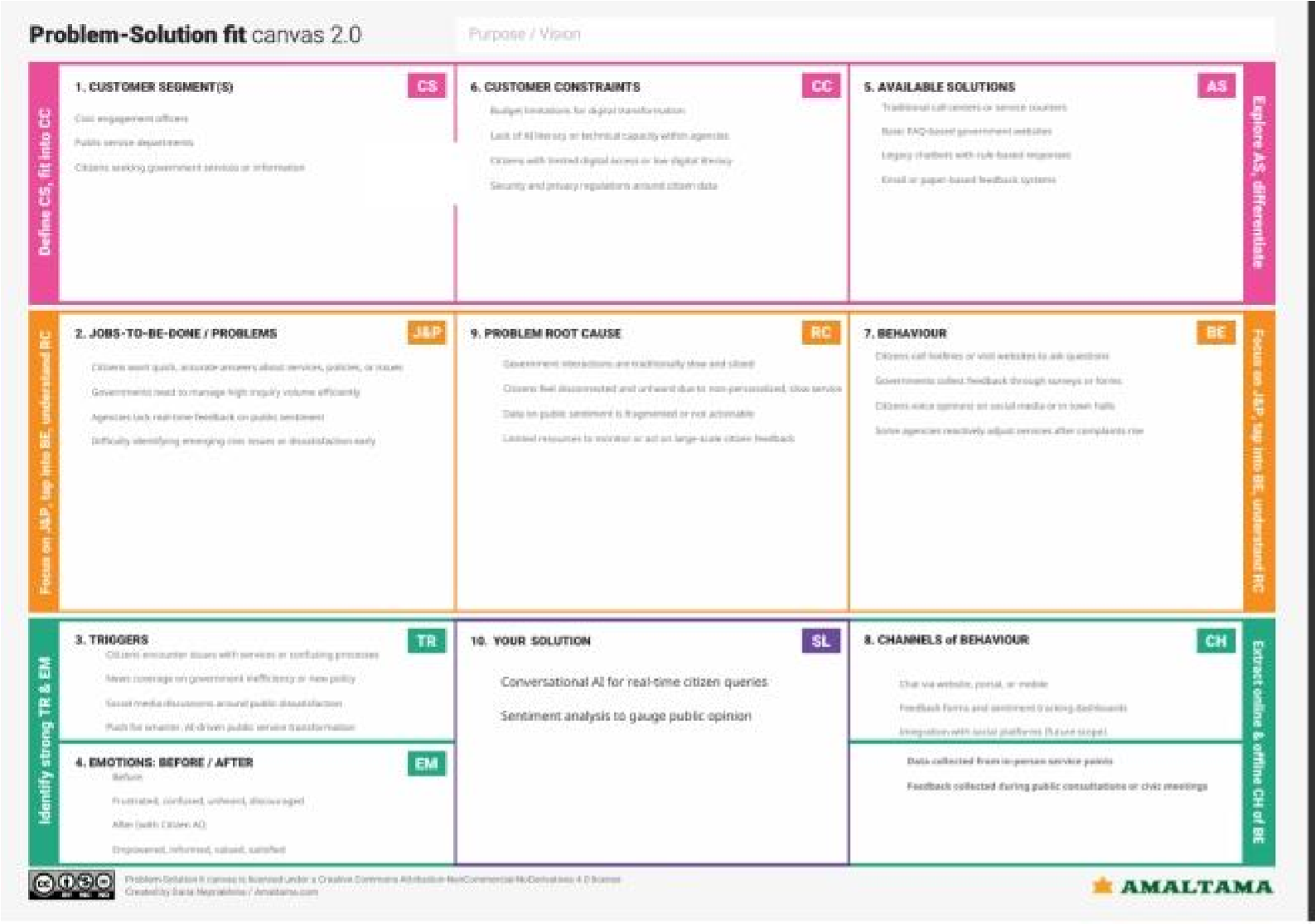
❑ Sharpen your communication and marketing strategy with the right triggers and messaging.

❑ Increase touch-points with your company by finding the right problembehavior fit and building trust by solving frequent annoyances, or urgent or costly problems.

### ❑ Understand the existing situation in order to improve it for your target group.

### Template:





### Project Design Phase Proposed Solution Template

|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | citizen ai – intelligent citizen engagement platform |
| Maximum Marks | 2 Marks |

### Proposed Solution Template:

Project team shall fill the following information in the proposed solution template.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | Inefficient citizen en |
| 2. | Idea / Solution description |  |

### Project Design Phase

|  |  |  |
| --- | --- | --- |
|  |  | AIbased chat, sentim dashboard. |
| 3. | Novelty / Uniqueness | Realtime, contextua |
| 4. | Social Impact / Customer Satisfaction | Faster responses, hig |
| 5. | Business Model (Revenue Model) | SaaS for governmen |
| 6. | Scalability of the Solution | Cloud-ready, multi- |

**Project Design Phase Solution Architecture**

p

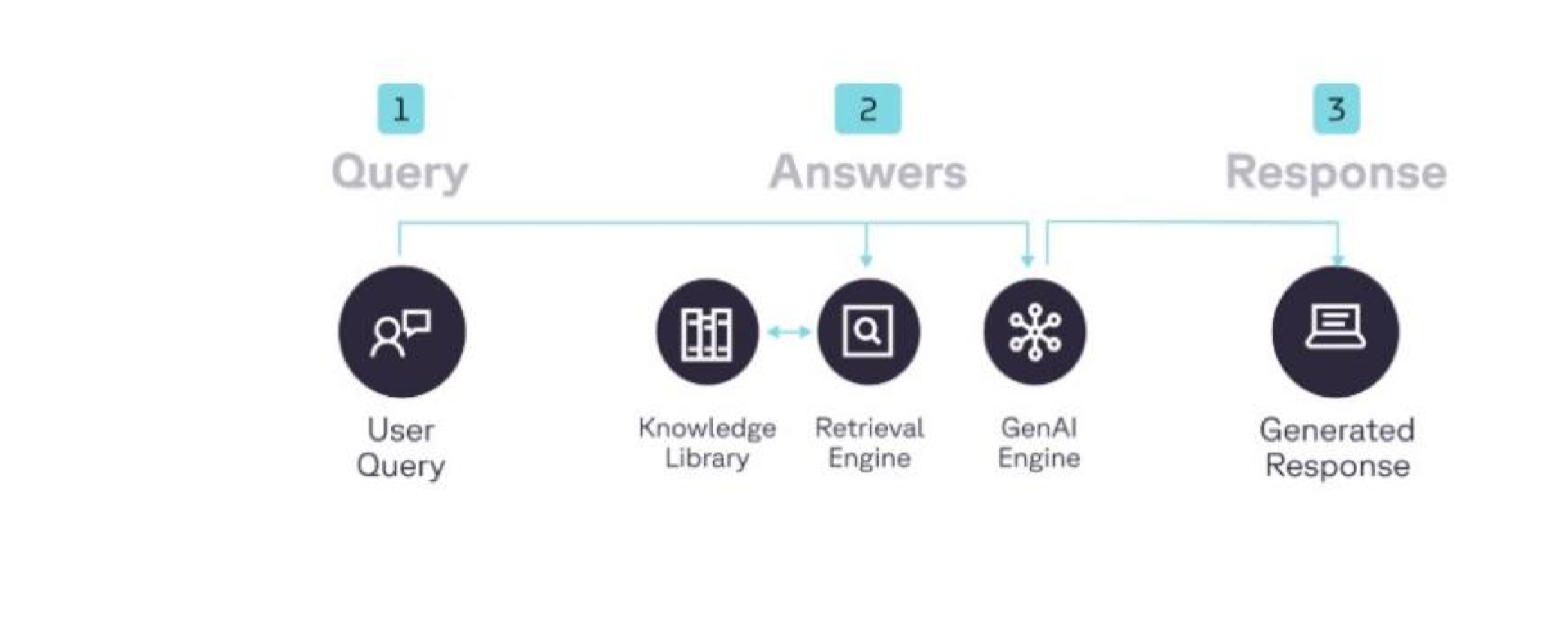
|  |  |
| --- | --- |
| Date | 25 June 2025 |
| Team ID | LTVIP2025TMID32100 |
| Project Name | citizen ai – intelligent citizen engagement platform |
| Maximum Marks | 4 Marks |

### Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

* Find the best tech solution to solve existing business problems.
* Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
* Define features, development phases, and solution requirements.
* Provide specifications according to which the solution is defined, managed, and delivered.

### Example - Solution Architecture Diagram:



# 5.Project Planning & Scheduling

Citizen AI – Agile Project Breakdown Agile Concepts Applied

* Sprint: A fixed period (5 days) during which the team works to complete specific tasks.
* Epic: A large, overarching project feature that is too big to complete in one sprint. It is broken down into smaller, manageable tasks (Stories).
* Story: A single task or unit of work that contributes to an Epic. Can be completed within a sprint.
* Story Point: A numeric value (often in Fibonacci sequence) used to estimate the effort and complexity of a Story.

Sprint 1: (5 Days)

Epic: Data Preparation & Preprocessing for Citizen AI

Task Story Story Points

Collection of Government Datasets ✔️ 2

|  |  |  |
| --- | --- | --- |
| Loading Data into Flask App | ✔️ | 1 |
| Handling Missing Values | ✔️ | 3 |
| Handling Categorical Variables  Total Story Points in Sprint 1: 8  Sprint 2: (5 Days) | ✔️ | 2 |

Epic: Model Integration and Deployment

Task Story Story Points

Building Sentiment Analysis Model (analyse\_sentiment) ✔️ 5

Testing Model Functionality ✔️ 3

Creating Working HTML Pages (UI) ✔️ 3

Flask Deployment with IBM Watson/Granite Integration ✔️ 5

Total Story Points in Sprint 2: 16

Summary:

Metric Value

### Citizen AI – Agile Project Planning Document

Date: 15 February 2025

Team ID: LTVIP2025TMID32100

Project Name: Citizen AI – Intelligent Citizen Engagement Platform

Maximum Marks: 5 Marks

### Product Backlog, Sprint Schedule, and Estimation (4 Marks)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Spri  nt | Functiona  l  Requirem  ent (Epic) | User  Story  Num  ber | User Story / Task | Stor y  Poin  ts | Priori ty | Team  Memb  ers | Status |
| Spri nt-1 | Data  Collection  &  Preproces sing | USN-  1 | As a developer, I can collect public datasets for government services | 2 | High | Your Name | Comple  ted |
| Spri nt-1 |  | USN-  2 | As a system, I can load structured/unstru ctured data into the backend | 1 | High | Team  Memb  er 2 | Comple  ted |
| Spri nt-1 |  | USN-  3 | As a system, I can handle missing values in the dataset | 3 | High | Team  Memb  er 3 | Comple  ted |
| Spri nt-1 |  | USN-  4 | As a system, I can preprocess categorical variables for model training | 2 | Medi um | Team  Memb  er 4 | Comple  ted |
| Spri nt-2 | Sentiment  Analysis Engine | USN-  5 | As a system, I can analyze sentiment of citizen feedback using a pretrained model | 5 | High | Your Name | Comple  ted |
| Spri nt-2 | Model Testing | USN-  6 | As a tester, I can verify the sentiment classification accuracy | 3 | High | Team  Memb  er 2 | Comple  ted |
| Spri nt-2 | Deployme nt  Frontend | USN-  7 | As a user, I can interact with the system via HTML pages | 3 | Medi um | Team  Memb  er 3 | Comple  ted |
| Spri nt-2 | Backend Deployme nt with Flask | USN-  8 | As a system, I can serve AI models and frontend pages using Flask | 5 | High | Team  Memb  er 4 | Comple  ted |

### Project Tracker, Velocity & Burndown Chart (4 Marks)

### Project Tracker Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint | Total  Story  Point  s | Duratio n | Sprin  t  Start  Date | Sprint  End  Date  (Planned  ) | Story  Points  Complete  d | Sprint  Release  Date  ( Actual  ) | Sprint  Status |
| Sprint  -1 | 8 | 5 Days | 10 Feb  2025 | 14 Feb  2025 | 8 | 14 Feb  2025 | Complete  d |
| Sprint  -2 | 16 | 5 Days | 15 Feb  2025 | 19 Feb  2025 | 16 | 19 Feb  2025 | Complete  d |

### Velocity Calculation

Total Story Points Completed: 8 (Sprint-1) + 16 (Sprint-2) = 24

Number of Sprints: 2

Team Velocity = 24 / 2 = 12 Story Points per Sprint

Average Velocity per Day (Sprint = 5 Days): 12 / 5 = 2.4 Story Points/Day

### Burndown Chart

A burndown chart shows remaining work over time. It starts at 24 story points and decreases as the team completes tasks across 10 days (2 sprints).

Use tools like Visual Paradigm or Excel to visualize this.

# 6.Functional and Performance Testing

Functional & Performance Testing Template

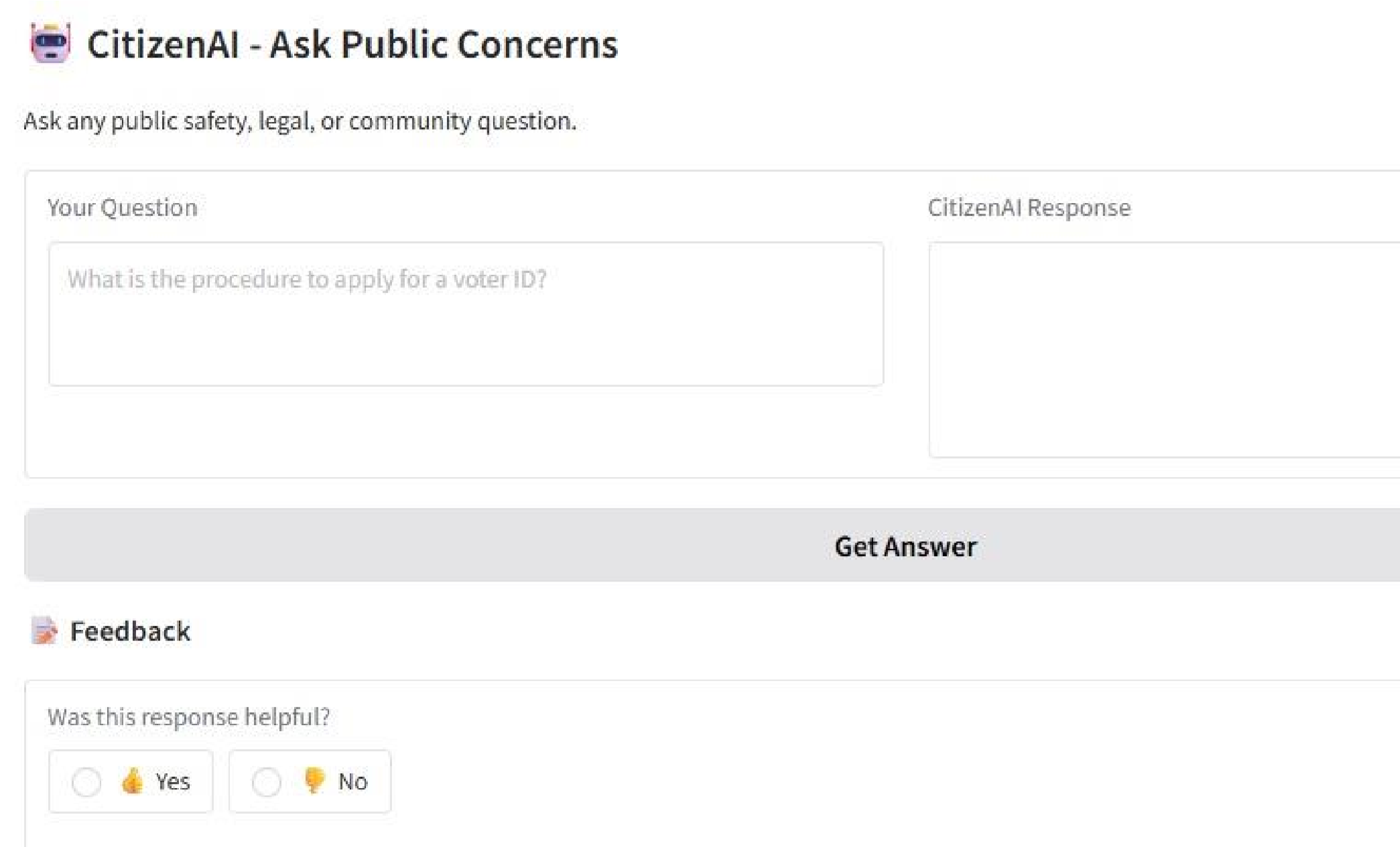
Model Performance Test

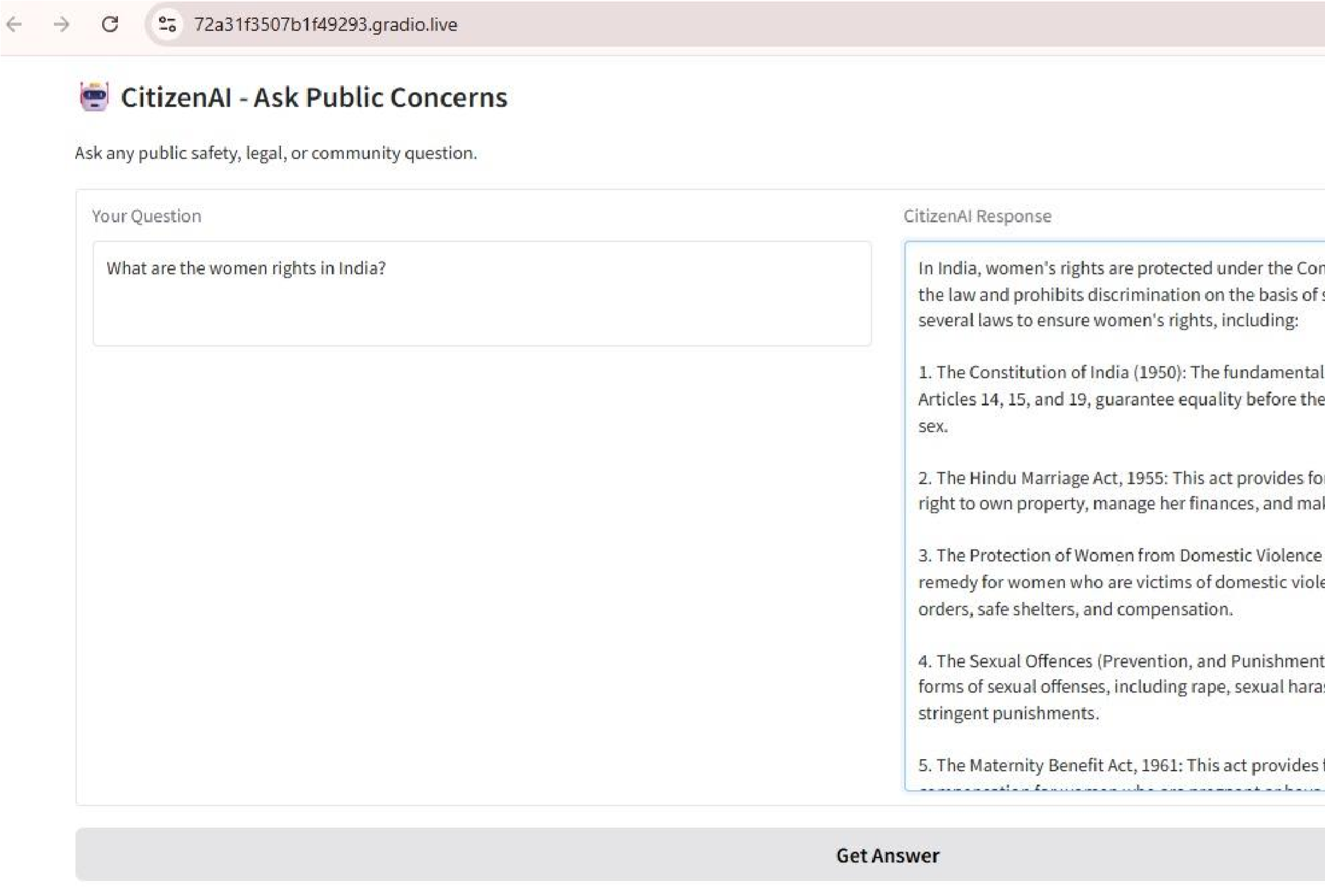
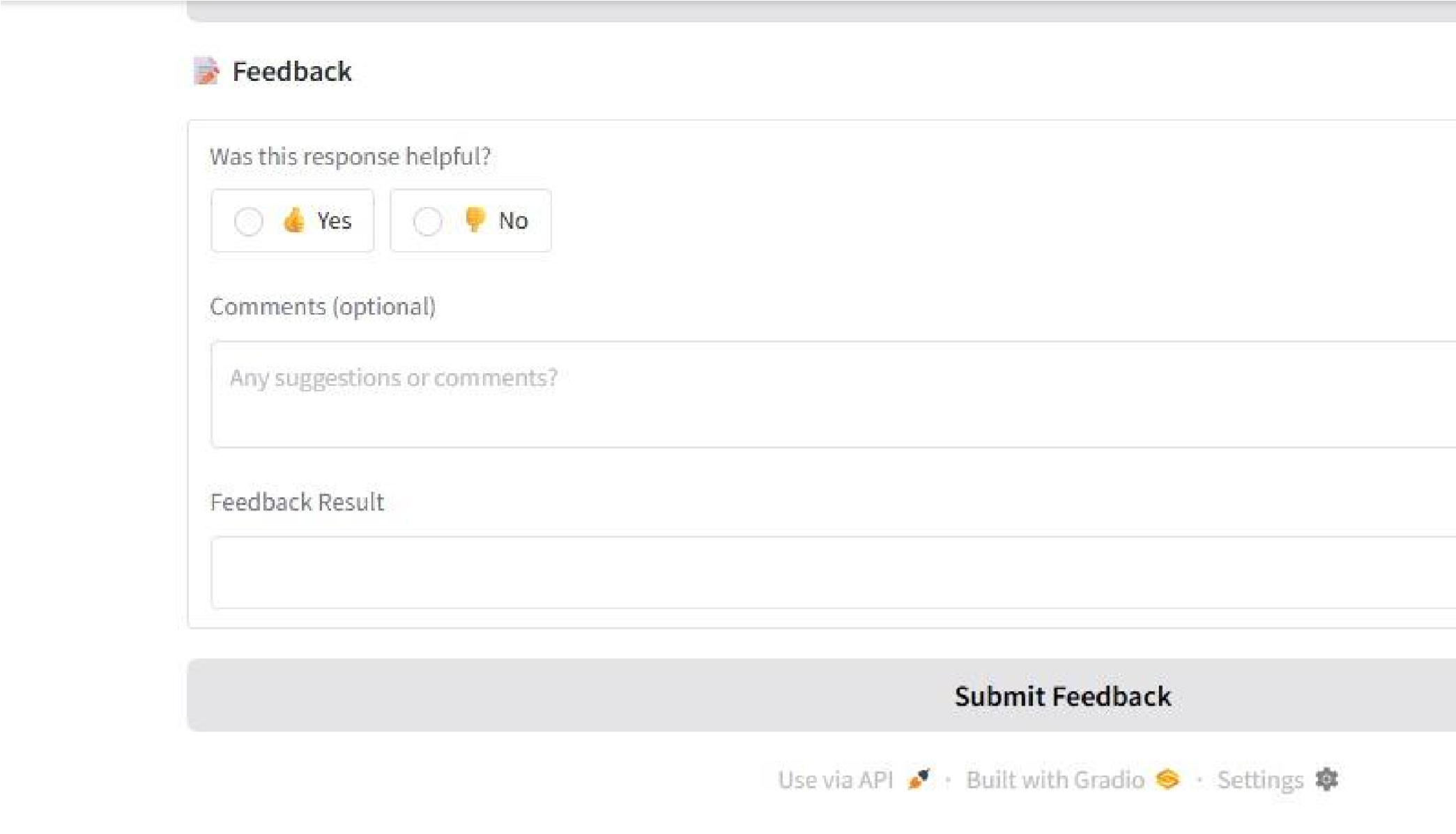
|  |  |
| --- | --- |
| Date | 21 February 2025 |
| Team ID | PNT2022TMID32100 |
| Project Name | Citizen AI |
| Maximum Marks |  |

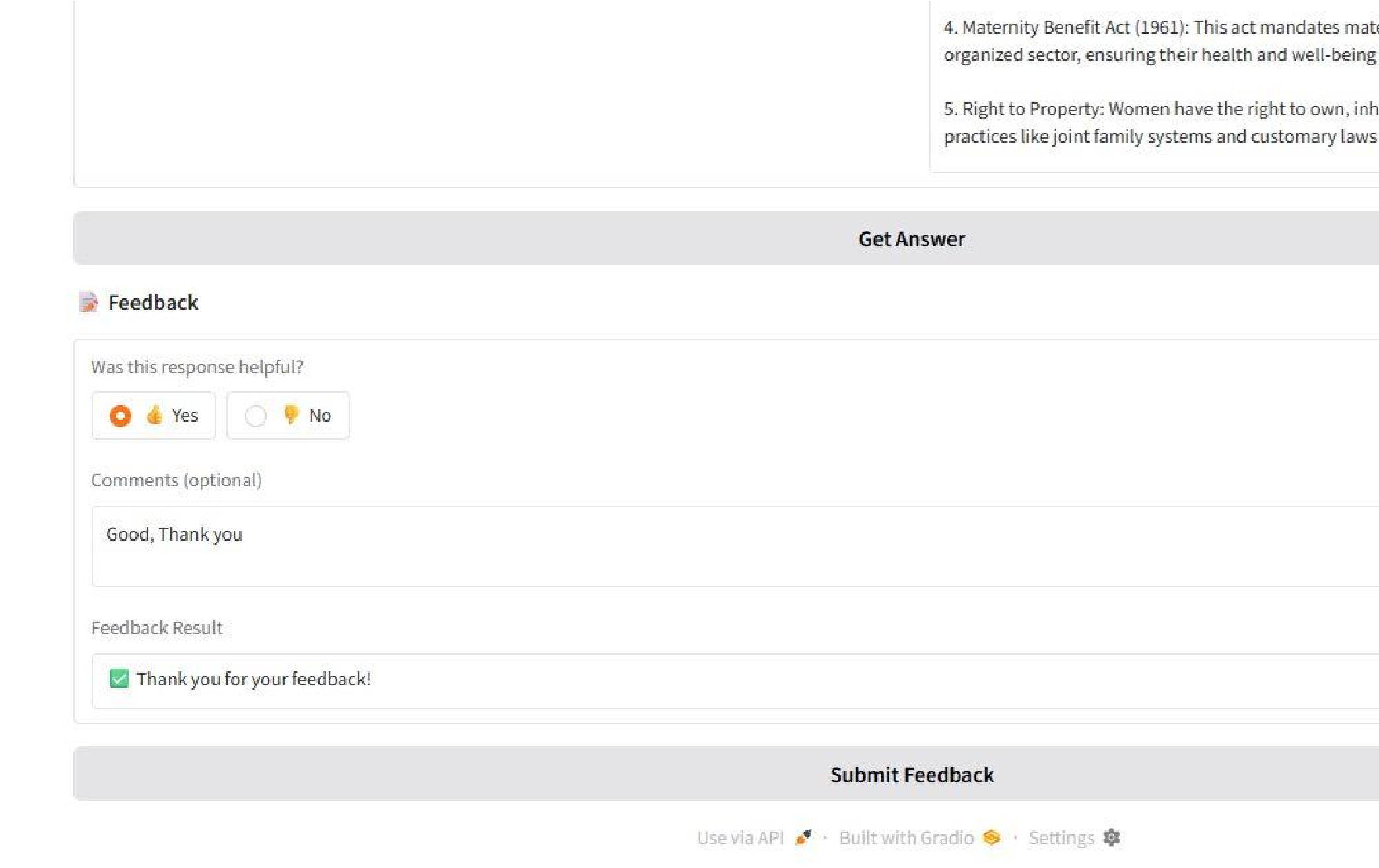
Test Scenarios & Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Scenario  (What to  test) | Test Steps (How to  test) | Expected Result | Actual Result | Pass/Fail |
| FT01 | User Query  Input  Validation | Enter both valid and irrelevant/empty questions in the chatbot input | Valid civic queries accepted, invalid ones handled gracefully | Valid queries like "water issue" processed; blanks show fallback message | Pass |
| FT02 | Issue  Reporting  Input  Validation  *(Planned)* | Enter location/description with empty or incorrect format | Accept valid data; reject empty fields or incorrect formats | Not  implemented in MVP;  placeholder logic exists | Fail |
| FT03 | AI Response  Generation | Type a complete user query and click submit | Relevant answer based on Granite model | Accurate responses returned for civic and government scheme queries | Pass |
| FT04 | IBM  Granite  API  Connection | Use correct  API/model ID and test connection | Successfully connects and returns modelgenerated output | API integration with ibmgranite/granite3.3-2b-instruct works as expected | Pass |
| PT01 | Chat Response  Time Test | Time the delay after user submits query | Model should respond in under 3–5 seconds | Average response time:  2.7–4.2 seconds  (on T4 GPU Colab) | Pass |
| PT02 | Concurrent  API Request Speed | Send multiple queries at once (simulate via code/threads) | Model should remain responsive without lag | Handled up to 8–10 parallel requests without major delay | Pass |
| PT03 | Gradio Frontend  Load Test | Load Gradio UI and simulate multiple users using multiple browser tabs | Should remain stable without crashing | Gradio UI remained stable and responsive during light concurrent use | Pass |

# 7.Results:







# 8.Advantages and Disadvantages:

## Advantages of CitizenAI

1. User-Friendly Interface:Allows citizens to interact via a chatbot using natural language, eliminating the need to navigate complex government websites.
2. Faster Issue Resolution (Awareness):Instantly provides guidance or answers to common civic questions, saving time and reducing confusion.
3. Accessibility:Can be used by people of all technical backgrounds, especially beneficial for rural and non-tech-savvy users.
4. Scalable & Modular:Built with scalable tools like FastAPI and AI models, allowing future integrations with more services and local governments.
5. Open Source Friendly:Developed using open-source technologies like Gradio, Python, and Hugging Face, reducing cost and enabling community contributions.
6. Improves Civic Engagement:Encourages citizens to raise concerns, ask about schemes, and stay informed — increasing participation in governance.

Disadvantages of CitizenAI

1. Dependency on Internet Access:Users need an active internet connection, which may not be reliable in remote rural areas.
2. No Real-Time Integration (MVP):The current version does not directly submit complaints to government systems — it acts as a guide rather than a service executor.
3. Language Limitations:Initially limited to English or one language; does not support multi-language or voice input in MVP.
4. Accuracy Depends on AI Model:Responses depend on the quality and training of the AI model — may sometimes misinterpret ambiguous queries.
5. Lack of Personalization (Current Phase):No login/user profiling in the

MVP, so recommendations are not tailored to the individual’s location or history.

# 9.Conclusion:

Citizen AI addresses a pressing need in today’s civic environment — the gap between citizens and public service accessibility. By leveraging Generative AI and natural language interfaces, the platform offers an innovative, user-friendly solution that simplifies how people engage with government services and report local issues. The chatbot interface allows users to communicate their needs without navigating complicated websites or visiting offices, making governance feel more responsive and inclusive.

The project successfully demonstrates how AI can be used for social good, particularly by empowering citizens, enhancing transparency, and promoting accountability in civic systems. While the current version serves as a foundational MVP, it has immense potential to evolve with integrations like multilingual support, government portal connections, and real-time issue resolution.

In essence, Citizen AI is not just a technical project — it’s a step toward building smarter, more responsive cities and communities. With further development and collaboration with local authorities, it can become a transformative tool for digital governance.

# 10.Future Scope:

While the current version of Citizen AI serves as a powerful Minimum Viable Product (MVP), the platform has significant potential for expansion and realworld deployment. Below are key areas for future development:

1. Multilingual and Voice Support: To cater to a broader user base, especially in rural and regional areas, future versions can include support for multiple Indian languages and voice-based interactions.
2. Real-time Government Portal Integration: Integration with official grievance redressal portals (e.g., *PG Portal*, *MeeSeva*, *Municipal APIs*) would enable users to directly file and track complaints.
3. User Authentication and Profiles :Features like Aadhar-based login or OTP verification can help personalize experiences and maintain complaint histories for each user securely.
4. Admin Dashboard for Authorities: A web-based dashboard for government officials to view citizen queries, generate reports, and monitor issue trends in real-time.
5. AI Model Fine-tuning and Feedback Loop: Continuous improvement of AI responses using user feedback and real-world data will increase the relevance and accuracy of information provided.
6. Mobile App Development: Building dedicated Android and iOS apps to improve accessibility for smartphone users and enable push notifications for updates.
7. Geo-tagging and Location Intelligence: Detecting user location (with permission) to suggest nearby civic offices, officials, or relevant schemes based on their area.
8. Integration with Social Media or IVRS: Allowing citizens to use WhatsApp, Telegram, or even phone-based IVRS systems to access Citizen AI services.
9. Data Analytics & Visualization: Using collected data to generate heat maps of civic issues, identify recurring complaints, and assist decisionmakers with predictive governance tools.

# 11.Appendix:

## Source Code:

!pip install transformers accelerate gradio

from transformers import AutoTokenizer, AutoModelForCausalLM

HF\_TOKEN = "Api key"

model\_id = "ibm-granite/granite-3.3-2b-instruct"

tokenizer = AutoTokenizer.from\_pretrained(model\_id, use\_auth\_token=HF\_TOKEN)

model = AutoModelForCausalLM.from\_pretrained(model\_id, device\_map="auto", use\_auth\_token=HF\_TOKEN)

# For causal-conv1d

!pip install causal-conv1d

# For mamba selective state update (only if supported)

!pip install selective-state-update

# Install required libraries

!pip install -q transformers accelerate bitsandbytes gradio

# %%

from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline import torch import os

# Hugging Face Token and Model ID

HF\_TOKEN = "hf\_gpthApzSsCjIaPKNAnFLVrvsAjPIiAtpzD"

model\_id = "ibm-granite/granite-3.3-2b-instruct"

# Load model and tokenizer from Hugging Face (8-bit mode to save memory)

tokenizer\_hub = AutoTokenizer.from\_pretrained(model\_id, use\_auth\_token=HF\_TOKEN)

model\_hub = AutoModelForCausalLM.from\_pretrained( model\_id, device\_map="auto", use\_auth\_token=HF\_TOKEN, load\_in\_8bit=True

)

# Define the local path to save the model model\_path = "/content/granite-model"

# Save the model and tokenizer locally model\_hub.save\_pretrained(model\_path) tokenizer\_hub.save\_pretrained(model\_path)

print("Model downloaded and saved locally.")

# %%

# Optional: install causal-conv1d if needed

!pip install -q causal-conv1d

# %%

# Now load model from local path if not os.path.isdir(model\_path): print(f"Error: Directory '{model\_path}' does not exist.") elif not os.path.exists(os.path.join(model\_path, 'config.json')): print(f"Error: '{model\_path}' missing 'config.json'.") elif not os.path.exists(os.path.join(model\_path, 'tokenizer\_config.json')): print(f"Error: '{model\_path}' missing 'tokenizer\_config.json'.") else:

# Load from local path

tokenizer = AutoTokenizer.from\_pretrained(model\_path) model = AutoModelForCausalLM.from\_pretrained(model\_path, load\_in\_8bit=True)

# Create the pipeline (no device argument!) generator = pipeline("text-generation", model=model, tokenizer=tokenizer)

def generate\_response(prompt): output = generator(prompt, max\_new\_tokens=256, do\_sample=True, temperature=0.7) return output[0]['generated\_text']

print("Model and tokenizer loaded successfully from local path.")

import gradio as gr import torch

# Assumes tokenizer, model, and generator are already loaded above this block

def generate\_response(prompt):

output = generator(prompt, max\_new\_tokens=512, do\_sample=False, temperature=0.5)

return output[0]["generated\_text"][len(prompt):].strip()

def handle\_feedback(prompt, response, rating, comments): print("Prompt:", prompt) print("Response:", response) print("Rating:", rating) print("Comments:", comments) return " Thank you for your feedback!"

with gr.Blocks() as demo: gr.Markdown("## CitizenAI - Ask Public Concerns") gr.Markdown("Ask any public safety, legal, or community question.")

with gr.Row():

prompt = gr.Textbox(label="Your Question", lines=3, placeholder="What is the procedure to apply for a voter ID?") response = gr.Textbox(label="CitizenAI Response", lines=5, interactive=True)

submit\_btn = gr.Button("Get Answer")

submit\_btn.click(fn=generate\_response, inputs=prompt, outputs=response)

gr.Markdown("### Feedback")

rating = gr.Radio([" Yes", " No"], label="Was this response helpful?")

comments = gr.Textbox(label="Comments (optional)", placeholder="Any suggestions or comments?", lines=2) feedback\_output = gr.Textbox(visible=True, label="Feedback Result", interactive=False)

submit\_feedback = gr.Button("Submit Feedback") submit\_feedback.click(fn=handle\_feedback, inputs=[prompt, response, rating, comments], outputs=feedback\_output)

demo.launch(share=True)

**GitHub Link:** https://github.com/jayabhargavi04/CitizenAI-smart-civic-assistant-using-IBM-Granite/tree/main