Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

---

11. Screenshots

(To be added after UI implementation)

---

12. Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

---

13. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention

Enhanced mobile application for wider adoption

Do you want me to format this into a Word/PDF file (like the IBM sample) with proper headings and styling?Audio transcript

EXIT COURSE

Question
10/10

A healthcare provider needs a summary of patient records to highlight recurring health concerns over the past year, such as seasonal illnesses and chronic conditions.

What should the prompt specify to ensure IBM Granite generates a useful summary?

Summarize patient records to highlight recurring health concerns and patterns.

Generate a detailed report on patient visits, including trends for each month.

Provide a list of the most prescribed medications along with usage patterns.

Classify symptoms into categories such as respiratory, digestive, and musculoskeletal conditions.Audio transcript

EXIT COURSE

Question
10/10
A healthcare provider needs a summary of patient records to highlight recurring health concerns over the past year, such as seasonal illnesses and chronic conditions.

What should the prompt specify to ensure IBM Granite generates a useful summary?

Summarize patient records to highlight recurring health concerns and patterns.

Generate a detailed report on patient visits, including trends for each month.

Provide a list of the most prescribed medications along with usage patterns.

Classify symptoms into categories such as respiratory, digestive, and musculoskeletal conditions.Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

---

11. Screenshots

(To be added after UI implementation)

---

12. Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

---

13. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention

Enhanced mobile application for wider adoption

Do you want me to format this into a Word/PDF file (like the IBM sample) with proper headings and styling?Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

---

11. Screenshots

(To be added after UI implementation)

---

12. Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

---

13. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention

Enhanced mobile application for wider adoption

Do you want me to format this into a Word/PDF file (like the IBM sample) with proper headings and styling?Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive

guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

---

11. Screenshots

(To be added after UI implementation)

---

12. Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

---

13. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention

Enhanced mobile application for wider adoption

Do you want me to format this into a Word/PDF file (like the IBM sample) with proper headings and styling?Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

---

11. Screenshots

(To be added after UI implementation)

---

12. Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

---

13. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention

Enhanced mobile application for wider adoption

Do you want me to format this into a Word/PDF file (like the IBM sample) with proper headings and styling?Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform

Team Leader: Sandhiya. S

Team Members:

Vanisri. M

Satheeswari. R

Lakshmi. R

Leelavathi. G

---

2. Project Overview

Purpose:
The purpose of Citizen AI is to enhance the way citizens interact with government bodies and community services. By leveraging artificial intelligence, natural language processing, and real-time analytics, the platform enables seamless communication, transparency, and efficiency. It empowers citizens to access information, provide feedback, and receive guidance on civic issues, while supporting officials with insights and data-driven decision-making.

Features:

Conversational Interface – Citizens can interact using natural language for queries, complaints, and service requests.

Policy Summarization – Government circulars and announcements are simplified into concise, easy-to-understand summaries.

Citizen Feedback Loop – Collects and analyzes public feedback to improve service delivery.

Real-Time Notifications – Keeps citizens updated on policies, events, and emergency alerts.

Service Request Tracking – Allows citizens to register complaints/issues and track resolution progress.

Community Insights Dashboard – Helps officials monitor public sentiment, feedback trends, and engagement statistics.

Multilingual Support – Ensures inclusivity by supporting multiple regional languages.

---

3. Architecture

Frontend (Streamlit/React): Interactive UI for citizens and officials with dashboards, feedback forms, and chat assistant.

Backend (FastAPI): Handles API requests, feedback storage, complaint tracking, and AI-based processing.

AI/LLM Integration (Watsonx/Other LLMs): Provides summarization, chatbot responses, and feedback analysis.

Database (MongoDB/PostgreSQL): Stores user queries, complaints, and system logs.

Analytics Module: Tracks citizen engagement, feedback trends, and generates actionable insights.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

FastAPI, Streamlit, and necessary Python libraries

API keys for LLM integration

Database setup (MongoDB/PostgreSQL)

Installation:

1. Clone the repository

2. Install dependencies via requirements.txt

3. Configure .env file with API keys and DB credentials

4. Run backend server with FastAPI

5. Launch frontend dashboard with Streamlit

6. Start interacting with the platform

---

5. Folder Structure

app/ – Backend logic and APIs
app/api/ – Routes for chat, feedback, and complaints
ui/ – Frontend components and citizen dashboard
document_summarizer.py – Policy summarization logic
feedback_analyzer.py – Analyzes citizen feedback
report_generator.py – Generates citizen engagement reports

---

6. Running the Application

Start the FastAPI backend server

Launch the Streamlit dashboard

Citizens interact via chat, forms, and complaint tracking

Officials access insights through the admin dashboard

---

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response

POST /submit-feedback – Collects and stores citizen feedback

POST /submit-complaint – Registers citizen complaints/issues

GET /track-complaint/{id} – Tracks status of submitted complaints

GET /policy-summary – Returns summarized government policies

---

8. Authentication

Role-based access control (Citizen / Official / Admin)

Token-based authentication (JWT)

Secure credential management

---

9. User Interface

Citizen Portal: Complaint registration, service requests, notifications, and chat assistance

Official Dashboard: Insights, analytics, and policy dissemination tools

Multilingual Support: Easy accessibility for all users

---

10. Testing

Unit Testing: Core modules like chatbot, feedback analyzer

API Testing: Endpoints validated with Swagger/Postman

User Testing: Ensuring accessibility and ease of use

Edge Case Handling: Invalid queries, large feedback sets

1.1Known Issues

Initial version lacks offline support

Limited to text input (voice support planned in future)

12. Future Enhancements

Voice-enabled citizen assistant

Integration with government service portals (payment, e-services)

AI-driven predictive analysis for civic issue prevention.

13. Screenshot