Project Documentation

CITIZEN AI

1. Introduction

- Project Title: CITIZEN AI
- **Team ID: durgadevi454598@gmail.com**
- **Team Leader:** DURGA DEVI P & durgadevi454598@gmail.com
- Team Members:
  - PREMA J & premaqueenmaker1122@gmail.com
  - BUVANESWARI E & buvana9940@gmail.com
  - NARMATHA M & n41997630@gmail.com

2. Project Overview

Purpose:

Citizen AI is a smart civic engagement platform designed to connect users with public service information and urban safety analytics. It includes both a freelancer-client project system and AI-powered civic interaction tools. It empowers users through project collaboration, real-time communication, and intelligent analysis of city conditions and government services.

Core Features:

- **Project posting and bidding system**
- **Secure user chat interface**
- **Feedback and review mechanism**
- **Admin control dashboard**
- **AI-powered city safety analysis**
- **Virtual government assistant for citizen queries**

3. Architecture

- **Frontend: React.js, Bootstrap, Material UI**
- **Backend: Node.js, Express.js (REST API)**
- **Database: MongoDB (stores users, projects, chats, etc.)**
- **AI System: Python (Transformers, Torch, Gradio)**
- **Execution Environment for AI: Google Colab with T4 GPU support**

4. Setup Instructions

Prerequisites:

- **Node.js**
- **MongoDB**
- **Git**
- **React.js**
- **Express.js**
- **Mongoose**
- **Visual Studio Code**

Installation Steps:

 # Clone the repository

git clone <your_repository_url>

 # Install frontend dependencies

cd client

npm install

 # Install backend dependencies

cd ../server

npm install

5. Folder Structure

Citizen-AI/

│

├── client/ # React frontend

│ ├── components/

│ └── pages/

│

├── server/ # Node.js backend

│ ├── routes/

│ ├── models/

│ └── controllers/

│

├── ai_colab/ # AI-based services (Gradio + Transformers)

│ └── citizen_ai.ipynb

6. Running the Application

Frontend:

cd client

npm start

Backend:

cd server

npm start

Access Application:

7. API Documentation

User Routes:

- **POST /api/user/register – Register new user**
- **POST /api/user/login – Login user**

Project Routes:

- **POST /api/projects/create – Create a new project**
- **GET /api/projects/:id – Get project details**

Application Routes:

- **POST /api/apply – Apply for a posted project**

Chat Routes:

- **POST /api/chat/send – Send a chat message**
- **GET /api/chat/:userId – Fetch chat history with a user**

8. Authentication

- **Method: JSON Web Token (JWT)**
- **Security:**
    - **Password hashing with bcrypt**
    - **Protected routes via Express middleware**
    - **Token verification on each protected endpoint**

9. User Interface (UI) Pages

- **Landing Page**
- **Freelancer Dashboard**
- **Project Details Page**

- **Admin Panel**
- **AI Tools (via embedded iframe or Gradio link)**

10. Testing

- **Method: Manual Testing**
- **Tools Used:**
  - **Postman (API testing)**
  - **Chrome DevTools (Frontend testing)**
  - **MongoDB Compass (DB validation)**

11. Screenshots / Demo

(Insert screenshots or demo links here)

- **Frontend Demo: http://localhost:3000**
- **AI Assistant Demo: *(Google Colab link will be provided after deployment)***

12. Known Issues

- **No socket-based real-time chat (currently uses polling)**
- **AI model inference delay due to cloud-based hosting**
- **Basic UI on mobile devices (needs responsive optimization)**

13. Future Enhancements

- **Add payment integration (Stripe or Razorpay)**
- **Add voice input to AI assistant**
- **Implement multilingual support for citizen interaction**
- **Integrate real-time city data (via public APIs)**
- **Role-based admin system with analytics**

14. Google Colab AI Assistant – Setup

This AI system offers two services:

- **City Analysis Tool**
- **Government Assistant for Citizens**

Instructions to Run in Google Colab:

1. **Open Google Colab**
2. **Set Runtime > Change runtime type > T4 GPU**
3. **Paste and run the following code:**

```
!pip install transformers torch gradio -q


import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM


model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else
torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)


if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token


def generate_response(prompt, max_length=1024):
```

```
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}
    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response


def city_analysis(city_name):
    prompt = f"Provide a detailed analysis of {city_name} including:\n1. Crime Index and safety statistics\n2. Accident rates and traffic safety information\n3. Overall safety assessment\n\nCity: {city_name}\nAnalysis:"
    return generate_response(prompt, max_length=1000)


def citizen_interaction(query):
    prompt = f"As a government assistant, provide accurate and helpful information about the following citizen query related to public services, government policies, or civic issues:\n\nQuery: {query}\nResponse:"
```

```
    return generate_response(prompt, max_length=1000)


with gr.Blocks() as app:
    gr.Markdown("# City Analysis & Citizen Services AI")
    with gr.Tabs():
        with gr.TabItem("City Analysis"):
            with gr.Row():
                with gr.Column():
                    city_input = gr.Textbox(label="Enter City Name",
placeholder="e.g., New York, London, Mumbai...")
                    analyze_btn = gr.Button("Analyze City")
                with gr.Column():
                    city_output = gr.Textbox(label="City Analysis (Crime Index &
Accidents)", lines=15)
            analyze_btn.click(city_analysis, inputs=city_input,
outputs=city_output)


        with gr.TabItem("Citizen Services"):
            with gr.Row():
                with gr.Column():
                    citizen_query = gr.Textbox(label="Your Query",
placeholder="Ask about public services, government policies...")
                    query_btn = gr.Button("Get Information")
                with gr.Column():
                    citizen_output = gr.Textbox(label="Government Response",
lines=15)
```

query_btn.click(citizen_interaction, inputs=citizen_query, outputs=citizen_output)

app.launch(share=True)