

CITIZEN AI INTELLIGENT CITIZEN ENGAGEMENT PLATFORM PROJECT DOCUMENT

1. INTRODUCTION:

- ❖ PROJECT TITLE : Citizen AI Intelligent Citizen Engagement platform
- ❖ Aarthi.R(38C4A4884F90F19A1654E7AA4FAAE28C)
- ❖ Bhuvaneswari. J (347FCOC6BD19EE398B94E3C4092208AD)
- ❖ Gobika.V (8DEE342D4434267A34344652E33F39ED)
- ❖ Nandhini.R (6CD478A3735DC139BE6429869CDE4EBO)

2. PROJECT OVERVIEW:

- ❖ Citizen AI with IBM is designed to create a bridge between citizens and government services using artificial intelligence. The project leverages IBM's AI ecosystem to provide citizens with simplified access to information, quick query resolution, and personalized recommendations. It is aimed at improving transparency, reducing response time, and enhancing trust between the government and citizens.
- ❖ Unlike traditional portals, Citizen AI integrates conversational agents, data analytics, and feedback systems to provide a more

human-like interaction. The goal is to make government services more accessible, especially for people with limited technical skills.

❖ Key features of Citizen AI include:

- ❖ AI-Powered Assistance – Citizens can interact with the system through natural language, asking questions about services, policies, or benefits.
- ❖ Sentiment Analysis – Feedback from citizens is automatically analyzed to measure satisfaction and identify issues.
- ❖ Policy Summarization – Long government documents are condensed into short, easy-to-read summaries.
- ❖ Multilingual Support – Supports multiple Indian and global languages, ensuring inclusivity.
- ❖ Personalized Suggestions – Provides tailored recommendations based on citizen profiles and past interactions.
- ❖ Secure Data Handling – Uses IBM's cloud infrastructure for data encryption and safe storage.
- ❖ Scalability – Designed to handle thousands of requests simultaneously with high availability.

3. ARCHITECTURE:

❖ The architecture of Citizen AI with IBM is divided into three major layers:

❖ 1. User Interaction Layer

- ❖ Web application and mobile app built with Gradio UI.
- ❖ Provides multilingual chat interface.
- ❖ Allows file uploads for policy or form analysis.

❖ 2. Application Logic Layer

❖ Dialog Manager: Handles conversation flow.

❖ Sentiment Analyzer: Processes feedback in real time.

❖ Recommendation Engine: Suggests relevant services.

❖ Authentication Module: Ensures secure citizen login.

❖ 3. Data & AI Layer

❖ IBM Watson NLP for text understanding.

❖ Granite Model Integration for policy summarization.

❖ Database (IBM Cloud Databases) for storing citizen interactions.

❖ Analytics Dashboard for government administrators to view trends.

4. SETUP INSTRUCTIONS:

❖ Prerequisites:

- Python programming knowledge.
- Gradio framework
- IBM granite model accesses(via hugging face)
- Google colab with T4 GPU
- Github account steps:

1. Access the Naan mudhalvan Smart Internz Portal.

2. Choose an IBM Granite model from Hugging face.

3. Run the application in Google colab with required libraries.
4. Upload final project files to Github

5.FOLDER STRUCTURE:

- ❖ app/:Backend logic and integration.
- ❖ ui/: Gradio app interface files.
- ❖ citizen_ai.py: Main application files.
- ❖ model_loader.py: Handles IBM granite model integration.
- ❖ dashboard.py: Visualization of citizen feedback.

6.RUNNING THE APPLICATION:

- 1.Open Google colab and load the project notebook.
- 2.Install dependencies and configure runtime with GPU.
- 3.Run the notebook cells to start the Gradio app.
- 4.Access the provided link to interact with Citizen AI.

7.API DOCUMENTATION:

CITIZEN AI PROVIDES ENDPOINTS FOR:

- ❖ Asking questions about government services.
- ❖ Uploading feedback for sentiment analysis.
- ❖ Viewing summarized policies.
- ❖ Accessing dashboards and reports.

8.AUTHENTICATION:

- Two factor Authentication(2FA):

- ❖ Users verify their identity using OTP send to their phone or E -Mail.
- Role based access:
 - ❖ Citizens: can ask queries and view information.
 - ❖ Officials: Can access detailed reports and analytics.
 - ❖ DataProtection: All credentials are encrypted using IBM key protect.

9.USER INTERFACE:

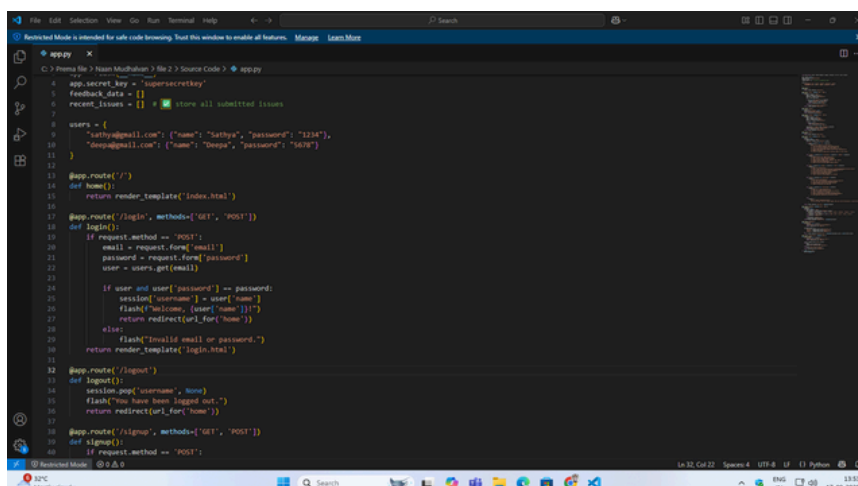
- CITIZEN PORTAL: simple chatbot interface with options for voice and text input.
- DASHBOARD: Graphical in sights for administrators,including charts for feedback sentiment.
- ACCESSIBILITY FEATURES:Adjustable font size,screen reader compatibility,and regional language support.

10.TESTING:

TESTING INCLUDED:

- ❖ Unit testing for AI responses.
- ❖ Manual testing of the Gradio interfaces.
- ❖ Edge case handling with unexpected inputs.

11.SCREENSHOTS:



The screenshot shows a code editor with Python code for a web application. The code includes a Flask app with routes for home, login, and logout. It uses Jinja2 templates for rendering HTML. The code is as follows:

```

1 # Import Flask modules
2 from flask import Flask, request, redirect, url_for, session, flash
3
4 # Create an instance of the Flask object
5 app = Flask(__name__)
6
7 # Secret key for session
8 app.secret_key = "supersecretkey"
9
10 # Feedback data
11 feedback_data = []
12
13 # Recent issues
14 recent_issues = []
15
16 # Users
17 users = {
18     "john@apmail.com": {"name": "John", "password": "1234"},
19     "mike@apmail.com": {"name": "Mike", "password": "5678"}
20 }
21
22 # Home route
23 @app.route("/")
24 def home():
25     return render_template("index.html")
26
27 # Login route
28 @app.route("/login", methods=['GET', 'POST'])
29 def login():
30     if request.method == "POST":
31         email = request.form['email']
32         password = request.form['password']
33         user = users.get(email)
34
35         if user and user['password'] == password:
36             session['username'] = user['name']
37             flash("Welcome, {}!".format(session['username']))
38             return redirect(url_for("home"))
39         else:
40             flash("Invalid email or password.")
41             return render_template("login.html")
42
43 # Logout route
44 @app.route("/logout")
45 def logout():
46     session.pop('username', None)
47     flash("You have been logged out.")
48     return redirect(url_for("home"))
49
50 # Signup route
51 @app.route("/signup", methods=['GET', 'POST'])
52 def signup():
53     if request.method == "POST":
54         email = request.form['email']
55         password = request.form['password']
56         confirm_password = request.form['confirm_password']
57
58         if password == confirm_password:
59             # Add new user
60             users[email] = {"name": request.form['name'], "password": password}
61             flash("User {} has been added.".format(request.form['name']))
62             return redirect(url_for("home"))
63         else:
64             flash("Passwords do not match.")
65             return render_template("signup.html")
66
67 # Feedback route
68 @app.route("/feedback", methods=['POST'])
69 def feedback():
70     feedback_data.append(request.form['feedback'])
71     return redirect(url_for("home"))
72
73 # Recent issues route
74 @app.route("/recent-issues", methods=['GET'])
75 def recent_issues():
76     return render_template("recent-issues.html")
77
78 # Run the app
79 if __name__ == "__main__":
80     app.run(debug=True)

```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

app.py
C:\Program Files\Naman Mudhavan> file 2 > Source Code > app.py
31 def login():
32     return redirect(url_for('login'))
33
34 @app.route('/signup', methods=['GET', 'POST'])
35 def signup():
36     if request.method == 'POST':
37         name = request.form['name']
38         email = request.form['email']
39         password = request.form['password']
40
41         if email in users:
42             flash("User already exists.")
43             return redirect(url_for('login'))
44
45         users[email] = {'name': name, 'password': password}
46         flash("Signup successful! Please login.")
47         return redirect(url_for('login'))
48     return render_template('signup.html')
49
50 @app.route('/about')
51 def about():
52     return render_template('about.html')
53
54 @app.route('/chat', methods=['GET', 'POST'])
55 def chat():
56     response = None
57     if request.method == 'POST':
58         question = request.form['question'].lower()
59
60         if "voter" in question and "id" in question:
61             response = {
62                 "here's how to apply for a voter ID:\n",
63                 "1. Visit the official portal: http://www.msp.in/a\n",
64                 "2. Click on I want to Register as a New Elector\n",
65                 "3. Fill in your details like name, age, address, etc.\n",
66                 "4. Upload required documents (photo, proof of age, and address)\n",
67                 "5. Submit the form\n",
68                 "6. You will receive an application reference number to track status"
69             }
70
71         elif "adhar" in question and ("correct" in question or "update" in question):
72             response = {
73                 "To update/correct your Adhar details:\n",
74                 "1. Visit: http://am.oido.gov.in/register\n",
75                 "2. Login with your Adhar number and OTP\n",
76                 "3. Choose fields to update (e.g., address, name, DOB)\n",
77                 "4. Upload valid proof documents\n",
78                 "5. Submit and note the URN (Update Request Number)"
79             }
80
81         elif "pan" in question and ("apply" in question or "card" in question):
82             response = {
83                 "To apply for a PAN card:\n",
84                 "1. Visit: http://www.collierservices.nadl.com/pan/en/other/registerContact.html\n",
85                 "2. Select application type as 'New PAN [ Indian Citizen ]'\n",
86                 "3. Fill the form with correct personal details\n",
87                 "4. Upload identity and address proof\n",
88                 "5. Pay the fee and submit the form\n",
89                 "6. You will receive an acknowledgment for tracking"
90             }
91
92         elif "birth" in question and "certificate" in question:
93             response = {
94                 "To get a birth certificate:\n",
95                 "1. Visit your local Municipal or state e-district website\n",
96                 "2. Fill the birth certificate application form\n",
97                 "3. Upload hospital birth slip or other proof\n",
98                 "4. Submit and download once approved"
99             }
100
101         elif "caste" in question and "certificate" in question:
102             response = {
103                 "To apply for a caste certificate:\n",
104                 "1. Visit your state's Meeva or eDistrict portal\n",
105                 "2. Login or register as a citizen\n",
106                 "3. Fill in caste details and upload required proof\n",
107                 "4. Submit the application and note the reference number\n",
108                 "5. Once approved, download the certificate online"
109             }
110
111         else:
112             response = {
113                 "Sorry, I didn't understand that.\n",
114                 "You can ask about: Voter ID, Adhar update, PAN card, Birth Certificate, or Caste Certificate."
115             }
116
117     return render_template('chat.html', response=response)
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

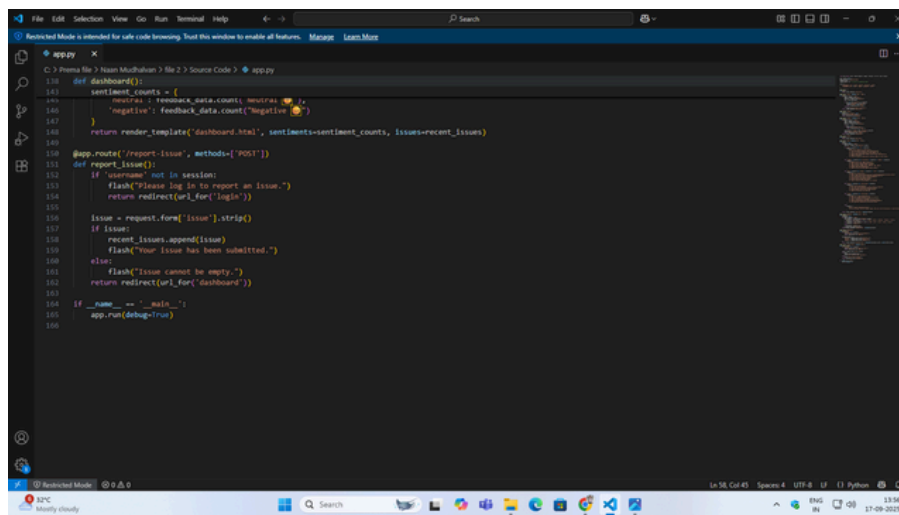
app.py
C:\Program Files\Naman Mudhavan> file 2 > Source Code > app.py
55 def about():
56     return render_template('about.html')
57
58 @app.route('/chat', methods=['GET', 'POST'])
59 def chat():
60     response = None
61     if request.method == 'POST':
62         question = request.form['question'].lower()
63
64         if "voter" in question and "id" in question:
65             response = {
66                 "here's how to apply for a voter ID:\n",
67                 "1. Visit the official portal: http://www.msp.in/a\n",
68                 "2. Click on I want to Register as a New Elector\n",
69                 "3. Fill in your details like name, age, address, etc.\n",
70                 "4. Upload required documents (photo, proof of age, and address)\n",
71                 "5. Submit the form\n",
72                 "6. You will receive an application reference number to track status"
73             }
74
75         elif "adhar" in question and ("correct" in question or "update" in question):
76             response = {
77                 "To update/correct your Adhar details:\n",
78                 "1. Visit: http://am.oido.gov.in/register\n",
79                 "2. Login with your Adhar number and OTP\n",
80                 "3. Choose fields to update (e.g., address, name, DOB)\n",
81                 "4. Upload valid proof documents\n",
82                 "5. Submit and note the URN (Update Request Number)"
83             }
84
85         elif "pan" in question and ("apply" in question or "card" in question):
86             response = {
87                 "To apply for a PAN card:\n",
88                 "1. Visit: http://www.collierservices.nadl.com/pan/en/other/registerContact.html\n",
89                 "2. Select application type as 'New PAN [ Indian Citizen ]'\n",
90                 "3. Fill the form with correct personal details\n",
91                 "4. Upload identity and address proof\n",
92                 "5. Pay the fee and submit the form\n",
93                 "6. You will receive an acknowledgment for tracking"
94             }
95
96         elif "birth" in question and "certificate" in question:
97             response = {
98                 "To get a birth certificate:\n",
99                 "1. Visit your local Municipal or state e-district website\n",
100                 "2. Fill the birth certificate application form\n",
101                 "3. Upload hospital birth slip or other proof\n",
102                 "4. Submit and download once approved"
103             }
104
105         elif "caste" in question and "certificate" in question:
106             response = {
107                 "To apply for a caste certificate:\n",
108                 "1. Visit your state's Meeva or eDistrict portal\n",
109                 "2. Login or register as a citizen\n",
110                 "3. Fill in caste details and upload required proof\n",
111                 "4. Submit the application and note the reference number\n",
112                 "5. Once approved, download the certificate online"
113             }
114
115         else:
116             response = {
117                 "Sorry, I didn't understand that.\n",
118                 "You can ask about: Voter ID, Adhar update, PAN card, Birth Certificate, or Caste Certificate."
119             }
120
121     return render_template('chat.html', response=response)
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

app.py
C:\Program Files\Naman Mudhavan> file 2 > Source Code > app.py
61     if request.method == 'POST':
62         elif "pan" in question and ("apply" in question or "card" in question):
63             response = {
64                 "To apply for a PAN card:\n",
65                 "1. Visit: http://www.collierservices.nadl.com/pan/en/other/registerContact.html\n",
66                 "2. Select application type as 'New PAN [ Indian Citizen ]'\n",
67                 "3. Fill the form with correct personal details\n",
68                 "4. Upload identity and address proof\n",
69                 "5. Pay the fee and submit the form\n",
70                 "6. You will receive an acknowledgment for tracking"
71             }
72
73         elif "birth" in question and "certificate" in question:
74             response = {
75                 "To get a birth certificate:\n",
76                 "1. Visit your local Municipal or state e-district website\n",
77                 "2. Fill the birth certificate application form\n",
78                 "3. Upload hospital birth slip or other proof\n",
79                 "4. Submit and download once approved"
80             }
81
82         elif "caste" in question and "certificate" in question:
83             response = {
84                 "To apply for a caste certificate:\n",
85                 "1. Visit your state's Meeva or eDistrict portal\n",
86                 "2. Login or register as a citizen\n",
87                 "3. Fill in caste details and upload required proof\n",
88                 "4. Submit the application and note the reference number\n",
89                 "5. Once approved, download the certificate online"
90             }
91
92         else:
93             response = {
94                 "Sorry, I didn't understand that.\n",
95                 "You can ask about: Voter ID, Adhar update, PAN card, Birth Certificate, or Caste Certificate."
96             }
97
98     return render_template('chat.html', response=response)
```

```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

app.py
C:\Program Files\Naman Mudhavan> file 2 > Source Code > app.py
107         else:
108             response = {
109                 "Sorry, I didn't understand that.\n",
110                 "You can ask about: Voter ID, Adhar update, PAN card, Birth Certificate, or Caste Certificate."
111             }
112
113     return render_template('chat.html', response=response)
114
115 @app.route('/feedback', methods=['GET', 'POST'])
116 def feedback():
117     sentiment = None
118     if request.method == 'POST':
119         user_feedback = request.form['feedback'].lower()
120
121         if any(word in user_feedback for word in ["good", "great", "awesome", "helpful", "love"]):
122             sentiment = "Positive 🟢"
123         elif any(word in user_feedback for word in ["bad", "poor", "worst", "hate", "slow"]):
124             sentiment = "Negative 🔴"
125         else:
126             sentiment = "Neutral ⚪"
127
128     feedback_data.append(sentiment)
129     return render_template('feedback.html', sentiment=sentiment)
130
131 @app.route('/dashboard')
132 def dashboard():
133     if 'username' not in session:
134         flash("Please log in to view the dashboard.")
135         return redirect(url_for('login'))
136
137     sentiment_counts = {
138         'positive': feedback_data.count("Positive 🟢"),
139         'neutral': feedback_data.count("Neutral ⚪"),
140         'negative': feedback_data.count("Negative 🔴")
141     }
142     return render_template('dashboard.html', sentiments=sentiment_counts, issues=recent_issues)
143
144 @app.route('/report-issue', methods=['POST'])
145 def report_issue():
146     pass
```

A screenshot of a Jupyter Notebook interface. The code defines a 'dashboard()' function that calculates sentiment counts from 'review_data' and 'feedback_data', and returns a template with these counts and a list of recent issues. A route for '/report-issue' is defined, which checks if the user is logged in, logs the report, and updates the 'recent_issues' list. The notebook is running on a Google Colab environment, as indicated by the 'Restricted Mode' warning and the 'Colab' logo in the bottom right corner.

12.KNOWN ISSUES:

- ❖ Limited scope due to demo environment.
- ❖ Requires internet for colab runtime and Hugging face model.

13.FUTURE ENHANCEMENTS:

- ❖ Integrate advanced analytics for deeper insights.
- ❖ Expand support for multiple languages. Deploy on
- ❖ cloud platforms for real world scalability.

14.DEMO VIDEO LINK

https://drive.google.com/file/d/1nhH5fxH1OIdQUaRqgcV4_Q-8Ws6Uf2UC/view?usp=drivesdk