THIRUTHANGAL NADAR COLLEGE

Department of Computer application

PROJECT DOCUMENTION

ON

"Citizen AI – Intelligent Citizen Engagement Platform"

Submitted by

Sandhiya T

Lathika V

Divyasree P

Deepika R

Index

BRAINSTORMING &IDEATION

Objective:

Problem Context

Governments often struggle with delayed and inefficient communication with the public. Traditional systems such as phone calls, static websites, or manual helpdesks make it difficult for citizens to get timely and relevant information regarding public services, policies, and civic issues.

Additionally, the government lacks effective tools to understand the overall mood or sentiment of citizens based on their feedback. As a result, citizen trust and engagement with digital governance platforms remain limited.

Purpose of the Project

Citizen AI was developed to solve these challenges by introducing a real-time AI assistant capable of:

- Answering citizen questions with human-like responses using IBM Granite models.
- Analyzing citizen feedback to detect sentiments such as satisfaction or dissatisfaction.
- Presenting actionable insights on a dynamic dashboard.
- Enhancing government responsiveness, transparency, and public trust through AI-powered interactions.

Key Points:

➢ Problem Statement

- Citizens face delays in receiving information or reporting issues.
- Feedback collected by government departments is rarely analyzed in real time.
- There is no central platform to combine public service interaction, sentiment analysis, and policy responsiveness.

➢ Proposed Solution

- **Citizens:** To ask questions, provide feedback, and get quick answers.
- **Government Officials:** To monitor public sentiment and improve services.
- **Policy Makers:** To use data insights for better governance decisions.
- **Developers/Admins:** To manage system operations and improve AI performance.

➢ Target Users

- **City Residents** – To report civic issues easily and get eco-advice.
- **City Administrators –** To monitor feedback, analyze KPIs, and respond to anomalies.
- **Urban Planners –** To summarize long documents and make informed policy decisions.
- **Teachers & Students –** To explore sustainability practices via the Eco Tips Generator.
- **Government Departments –** For utility monitoring and forecasting resource demands

➢ Expected Outcomes

- A responsive, intelligent chatbot available 24/7 for public queries.
- Real-time analysis of citizen feedback to detect trends and issues.
- Dashboards that visualize public mood and interaction frequency.
- Enhanced digital governance through AI, leading to improved public satisfaction and trust.

REQUIREMENT ANALYSIS

Objective:

Functional Requirements:

The Citizen AI platform is designed to offer several key functionalities that enhance citizen engagement and government responsiveness. At the core of the system is a real-time chat assistant that allows users to interact with public service information using natural language. The platform processes user queries and returns intelligent responses using IBM Granite large language models (LLMs). In addition, citizens can submit feedback on government services through a user-friendly web interface. This feedback is stored securely and is automatically analyzed for sentiment—classified as positive, neutral, or negative—to help identify areas of concern or satisfaction. A dynamic dashboard visualizes this sentiment data along with interaction trends and other metrics, enabling decision-makers to understand public opinion in real-time. The system also features a contextual response engine that tracks user interactions and provides personalized replies based on the conversation history, enhancing the overall relevance of the chatbot's responses. All citizen interactions, feedback entries, and sentiment results are persistently stored in a backend database to support analytics and reporting.

Technical Requirements:

The technical implementation of Citizen AI relies on modern, scalable, and secure architecture. The backend is built using Python with the Flask framework, allowing for modular API routing and clean project structure. The frontend is implemented using HTML, CSS, and JavaScript to create responsive user interfaces, including the chatbot, feedback form, and analytics dashboard. AI functionality is powered by IBM Granite, which handles natural language understanding and response generation. The system connects to a relational database such as SQLite or PostgreSQL to store user feedback, sentiment scores, and session logs. Configuration settings, including API keys and database URIs, are securely managed using .env files and a centralized config.py module. To ensure reliability and performance, the application is expected to maintain low response latency (under 3 seconds per query) and support concurrent users through optimized backend routes and efficient frontend rendering. Basic security measures like input validation and environment-based configuration are implemented to protect user data and prevent vulnerabilities. The system also

includes provisions for unit and integration testing to maintain code quality and long-term maintainability.

Key points:

➢ Technical Requirements:

- Backend developed using **Flask (Python)** with RESTful routing.
- Frontend created using **HTML, CSS, and JavaScript** for interactive UI.
- AI integration handled via **IBM Granite API** for NLP and chat responses.
- Database setup using **SQLite or PostgreSQL** for structured data storage.
- Sensitive data and configuration managed through .env and config.py.
- Chat latency kept under **3 seconds** per response for smooth UX.
- Scalable and modular design to support multiple users and future features.
- Input sanitization and secure API key handling for data protection.
- Unit and integration tests included to ensure quality and reliability.

➢ Functional Requirements:

- Enable real-time chatbot interaction using IBM Granite.
- Accept citizen queries and provide human-like responses.
- Allow feedback submission through a web form interface.
- Automatically analyze sentiment (Positive, Neutral, Negative) from feedback.
- Display sentiment insights and user trends on a dynamic dashboard.
- Provide personalized and context-aware replies based on session history.
- Store all user queries, feedback, and sentiment results in a secure database.

➢ Constraints & Challenges:

- **Granite API Token Limitations:**
  - May limit length or frequency of messages; prompt optimization required.
- **Latency in Real-Time Communication:**
  - Long response times may degrade user experience; handled with async support.
- **Sentiment Classification Accuracy:**
  - Simple models may misclassify neutral/negative tones.
- **Data Privacy & Consent:**
  - Citizen feedback may include sensitive content; requires secure storage.
- **User Diversity:**
  - Citizens with varying levels of digital literacy require intuitive UI.
- **Deployment & Infrastructure:**
  - Hosting costs, cloud configuration (optional), and domain management may be required for demo readiness.

PROJECT DESIGN

Objective:

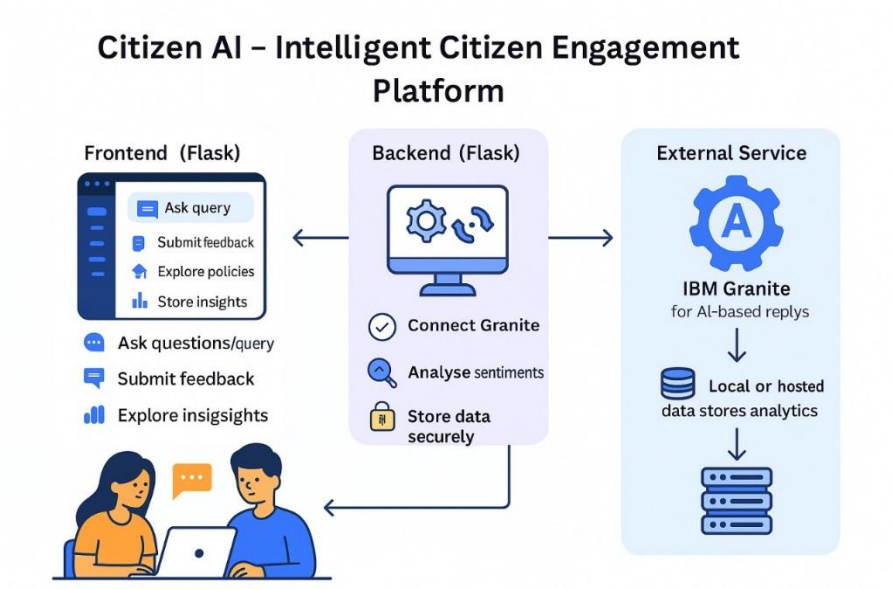
The objective of the Project Design phase is to define the overall structure, flow, and technical organization of the Citizen AI platform. It includes the layout of system components, the interaction between modules, and the architecture that enables real-time chatbot responses, sentiment analysis, feedback handling, and analytics dashboard visualization. This design ensures that the system remains scalable, modular, and maintainable, while delivering a smooth experience to both citizens and administrators

Key points:

The Citizen AI platform follows a modular Flask architecture with distinct components for routes, models, templates, and static files, ensuring clean structure and maintainability. User queries are handled via chat_routes.py, processed by granite_interface.py using IBM Granite, and returned in real-time through chat.js. Feedback is collected via feedback_routes.py, analyzed by sentiment_analysis.py, and stored in the database. Admins view insights through dashboard_routes.py, which fetches and visualizes sentiment trends. Contextual responses are enabled by session tracking. Configuration is securely managed through .env and config.py, supporting smooth AI-driven workflows

.

➢ System Architecture Diagram:



➢ User flow:

- A **citizen** opens the platform and interacts with the chatbot by typing a query.
- The message is sent to the backend, where IBM Granite processes it and returns a response.
- The citizen may also submit feedback using a form, which is analyzed for sentiment and stored in the database.
- An **admin** logs into the dashboard to monitor overall sentiment trends and feedback analytics.
- The system tracks session history, enabling context-aware, more personalized responses for returning or active users.

➢ UI/UX Consideration:

The Citizen AI platform is designed with a clean, responsive, and user-friendly interface to ensure ease of access for users of all digital literacy levels. The chatbot interface provides real-time feedback using smooth interactions and toast notifications to confirm actions. The layout uses Bootstrap for responsive design, ensuring compatibility across devices such as desktops, tablets, and mobile phones.

High-contrast colors and readable font sizes are used to support accessibility, and input fields are clearly labeled for better usability. The dashboard presents key metrics through intuitive charts and minimal clutter, allowing administrators to quickly understand public sentiment and engagement. The overall design prioritizes simplicity, clarity, and functionality to enhance user engagement and trust.

PROJECT PLANNING

Objective:

The objective of the Project Planning phase is to structure the development of the Citizen AI platform into manageable tasks and timeframes using an Agile methodology. By organizing work into focused sprints and distributing responsibilities across the team, the project ensures steady progress, timely feedback, and successful completion of each module. This structured plan includes sprint goals, task allocation, and milestone tracking.

Key Points:

- **Sprint Planning:**
  - **Sprint 1: Project Setup**
    - Initialize Flask project with app.py, config.py, .env, and install dependencies.
  - **Sprint 2: Chatbot Development**
    - Build real-time chatbot with chat_routes.py, granite_interface.py, and frontend (chat.html, chat.js).
  - **Sprint 3: Feedback & Sentiment**
    - Create feedback form, classify sentiment using sentiment_analysis.py, and store data in DB.
  - **Sprint 4: Dashboard Integration**
    - Develop admin dashboard to visualize sentiment and engagement trends.
  - **Sprint 5: Contextual Chat**
    - Implement session-based contextual replies using helpers.py.
  - **Sprint 6: Database Setup**
    - Define schemas in models.py, initialize DB with init_db.py.
  - **Sprint 7: Testing & QA**
    - Perform unit/integration testing and optimize performance.
  - **Sprint 8: Deployment**
    - Prepare deployment configs, add demo data, finalize UI and documentation.

Task Allocation:

Members	Tasks
Sandhiya T	• Granite integration • chatbot logic • UI components, testing • final review
Lathika V	• Feedback module • database setup • .env configuration • deployment support
Divya sree P	• Sentiment analysis • Flask routing • frontend UI • unit testing
Deepika R	• Dashboard implementation • admin analytics • database model definitions

TimeLine and Milestones:

Date	Milestone
Week-1	● Flask setup ● Granite API integration ● chatbot module
Week-2	● Feedback form ● sentiment analysis ● database connectivity
Week 3	● Dashboard implementation ● context tracking logic
Week 4	● Final testing, bug fixes, UI polish, documentation, and deployment

PROJECT DEVELOPMENT

Objective:

The objective of the Project Development phase is to implement, integrate, and test all modules of the Citizen AI platform, transforming design into a fully functional system. This includes building the chatbot, sentiment analysis module, feedback system, dashboard, and database integration, ensuring they work together seamlessly.

Key Points:

➢ Technology Stack Used:

- **Backend Framework:** Python with Flask (for modular API routing)
- **Frontend Technologies:** HTML, CSS, JavaScript (with Bootstrap)
- **AI Integration:** IBM Granite LLM (for chatbot and contextual responses)
- **Sentiment Analysis:** Custom Python module using NLP techniques
- **Database:** SQLite (development) / PostgreSQL (production-ready)
- **Configuration & Secrets:** .env file and config.py for environment-based setup
- **Version Control:** Git and GitHub
- **Deployment (Optional):** Docker, cloud hosting options

➢ Development Process:

The development of Citizen AI followed a structured and modular approach. The project was divided into independent yet connected components, ensuring flexibility and easier debugging. Each major feature was implemented and tested step by step, starting from backend setup to AI integration and frontend user experience. Below are the key stages of the development process:

Flask Project Initialization:

- Created the basic Flask structure with app.py, config.py, and .env for secure settings.
- Installed necessary dependencies including Flask, IBM Granite SDK, and SQLAlchemy.

Chatbot Module Implementation:

- Developed chat_routes.py to handle incoming chat messages.
- Integrated IBM Granite API through granite_interface.py to fetch AI responses.
- Designed chat.html and chat.js for real-time user interaction on the frontend.

Feedback & Sentiment Analysis:

- Created a feedback form and handled submissions via feedback_routes.py.
- Analyzed sentiment using the analyse_sentiment() function in sentiment_analysis.py.
- Stored feedback and sentiment data into the database using SQLAlchemy models.

Dashboard Development:

- Built dashboard_routes.py to serve sentiment and feedback metrics.
- Designed dashboard.html with embedded charts for real-time analytics using JavaScript.
- Displayed sentiment counts and citizen engagement trends for admin users.

Contextual Response Enhancement:

- Implemented session tracking to maintain chat context.
- Used helpers.py to store previous interactions and send enriched prompts to Granite for more relevant responses.

Database Integration:

- Defined all database schemas (User, Feedback, Sentiment) in models.py.
- Created init_db.py for easy initialization and reset during testing.
- Connected the application to SQLite/PostgreSQL for persistent data handling.

Testing and Final Integration:

- Performed module-wise testing followed by full integration.
- Addressed sync issues between frontend and backend.
- Ensured end-to-end functionality from chat input to dashboard visualization.

➢ Challenges & Fixes:

- **API Integration Issues:**
  Initial connection to IBM Granite API failed due to incorrect headers and key usage. Fixed by properly loading environment variables and validating API structure.
- **Slow Chat Response:**
  LLM responses were delayed due to large prompts. Solved by optimizing the prompt format and reducing input token length.
- **Frontend and Backend Sync:**
  Chat UI failed to reflect real-time responses. Resolved by properly handling async calls and using consistent JSON structures.
- **Sentiment Misclassification:**
  Feedback was inaccurately tagged. Improved model accuracy by refining classification logic and preprocessing input text.
- **UI Freezing & Form Reloads:**
  Long LLM calls caused UI lag. Handled with asynchronous JS and form handling improvements.

- **Database Connection Errors:**
  Faced schema mismatches during early testing. Resolved by standardizing model definitions and resetting the database using init_db.py.

FUNCTIONAL & PERFOMANCE TESTING

Objective:

The primary objective of the Functional and Performance Testing phase is to ensure that all features of the Citizen AI platform work as intended and that the system performs reliably under expected user conditions. This phase involved testing each module individually and then in combination, checking for accuracy, responsiveness, and stability. The goal was to identify and resolve any bugs, inconsistencies, or performance bottlenecks before deployment.

Key Points:

➢ Test Cases Executed:

- **Chatbot:**
  Verified real-time responses from IBM Granite for various citizen queries.
- **Feedback Submission:**
  Checked form handling, data storage, and confirmation messages.
- **Sentiment Analysis:**
  Tested classification as Positive, Neutral, or Negative for diverse feedback inputs.
- **Dashboard:**
  Validated data visualization, sentiment graphs, and trend accuracy.
- **Contextual Chat:**
  Ensured conversation history is maintained and used for relevant replies.
- **Database:**
  Confirmed data integrity, proper schema usage, and reliable storage.
- **Performance:**
  Measured response times (<3s), smooth UI experience, and multi-user handling.

➢ Bug Fixes & Improvements:

Several issues were identified and fixed during testing:

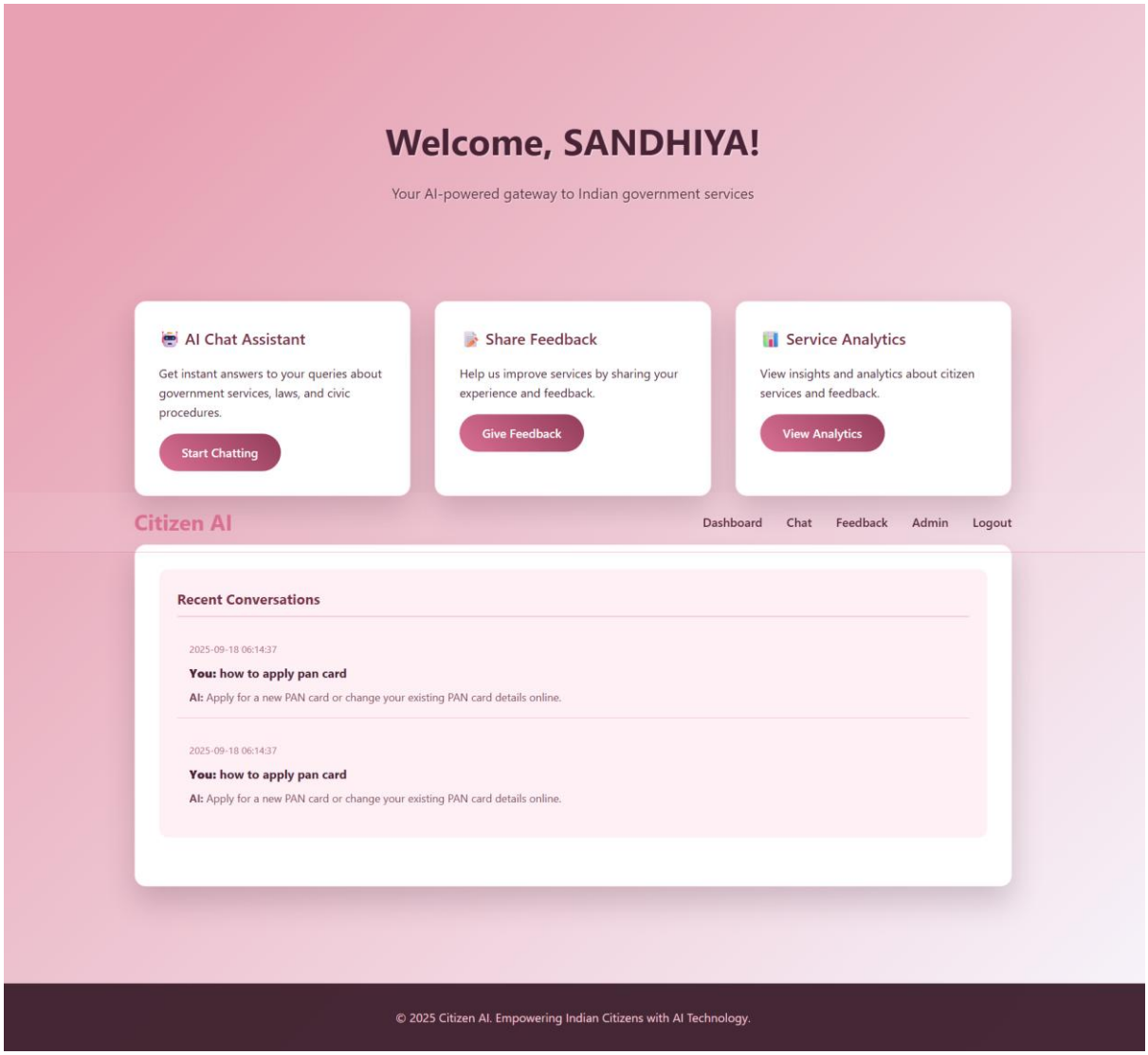
- **Granite API Delay:**
  Resolved by reducing prompt size and optimizing token usage.
- **Incorrect Sentiment Labels:**
  Refined logic in analyse_sentiment() to improve accuracy.
- **Frontend-Backend Mismatch:**
  Standardized API responses to maintain consistent data formats.
- **Chart Rendering Errors:**
  Fixed parsing logic for feedback timestamps and values.
- **UI Freezing on Long Responses:**
  Implemented asynchronous handling and loading indicators.

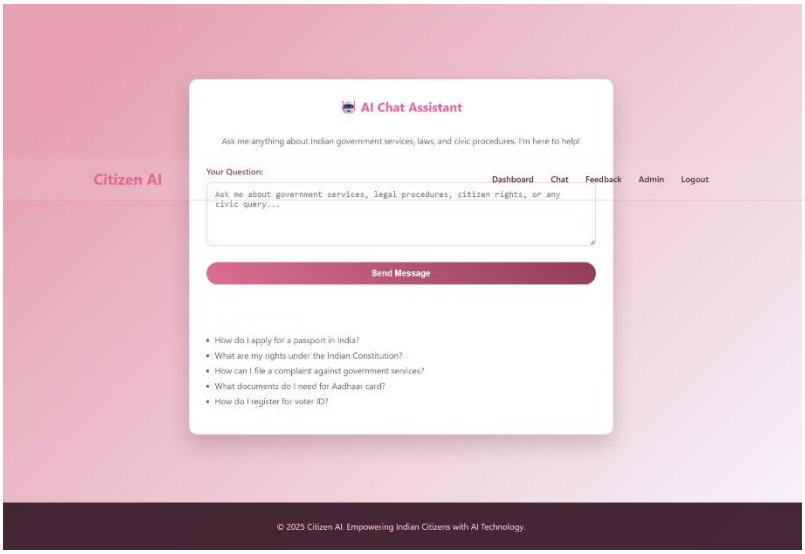
➢ Final Validation:

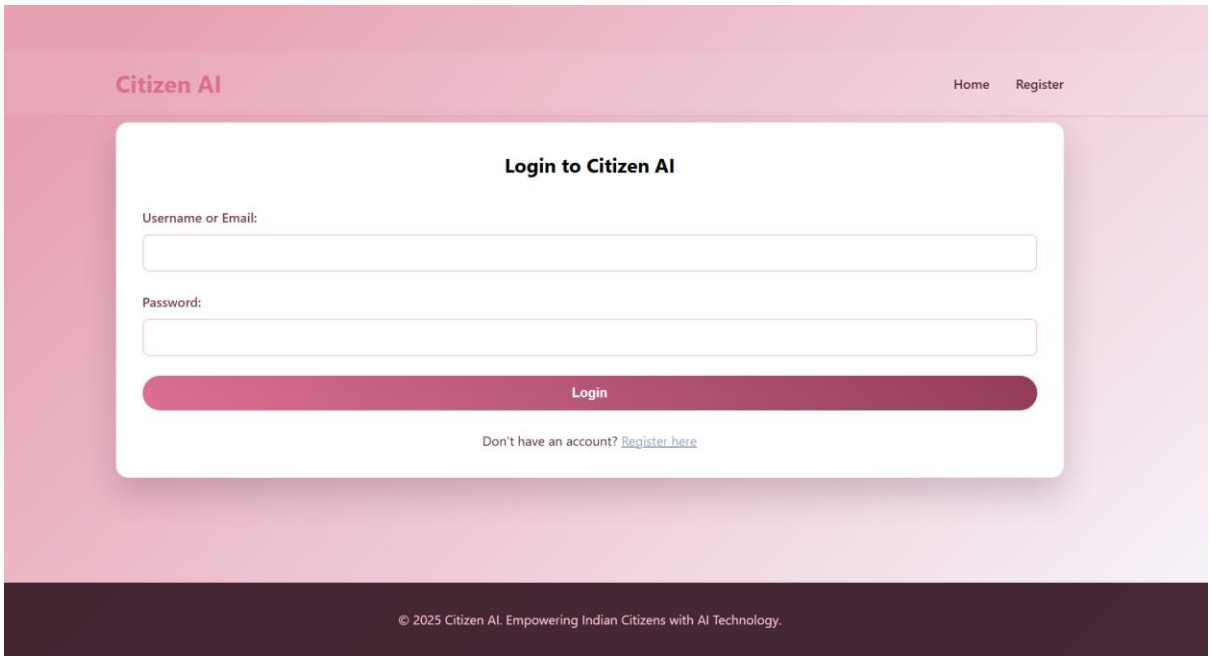
After thorough testing, the Citizen AI platform was confirmed to:

- Handle real-time citizen interaction effectively.
- Process and store feedback accurately.
- Display sentiment trends and analytics correctly.
- Respond quickly and scale to multiple users.
- Meet the functional requirements defined in the planning phase.

➢ Deployment:


Welcome, SANDHIYA!
Your AI-powered gateway to Indian government services
AI Chat Assistant
Get instant answers to your queries about government services, laws, and civic procedures.
Start Chatting
Share Feedback
Help us improve services by sharing your experience and feedback.
Give Feedback
Service Analytics
View insights and analytics about citizen services and feedback.
View Analytics
Citizen AI
Dashboard
Chat
Feedback
Admin
Logout
Recent Conversations
2025-09-18 06:14:37
You: how to apply pan card
AI: Apply for a new PAN card or change your existing PAN card details online.
2025-09-18 06:14:37
You: how to apply pan card
AI: Apply for a new PAN card or change your existing PAN card details online.
© 2025 Citizen AI. Empowering Indian Citizens with AI Technology.


Citizen AI
Dashboard
Chat
Feedback
Admin
Logout
AI Chat Assistant
Ask me anything about Indian government services, laws, and civic procedures. I'm here to help!
Your Question:
Ask me about government services, legal procedures, citizen rights, or any civic query...
Send Message
How do I apply for a passport in India?
What are my rights under the Indian Constitution?
How can I file a complaint against government services?
What documents do I need for Aadhaar card?
How do I register for voter ID?
© 2025 Citizen AI. Empowering Indian Citizens with AI Technology.


Citizen AI
Home
Register
Login to Citizen AI
Username or Email:
Password:
Login
Don't have an account? Register here
© 2025 Citizen AI. Empowering Indian Citizens with AI Technology.