CivicVoice - Civic Issue Reporting and Status Tracking System

Team Number	Team Number: 026
Team Members	Aviral Tyagi (2025201086), Shivam Patel (2025202030), Vadla Shiva Kishan (2025202036), Varun Modi (2025202040), Harsh Jaiswal (2025204011)
Code Repository	https://github.com/varunmodi18/civic-voice-project.git
Presentation	PPT URL
Demo	URL (Youtube Unlisted URL)

About Project

Municipal analysts currently struggle to process citizen complaints effectively because reports arrive in various unstructured formats, such as emails, social media posts, etc. This lack of standardization makes it difficult to extract critical details, categorize issues accurately, and route them to the correct civic department for timely resolution. CivicVoice acts as a conversational tool that guides citizens to clarify their complaints, automatically transforming unstructured input into standardized, actionable summaries for effective municipal routing.

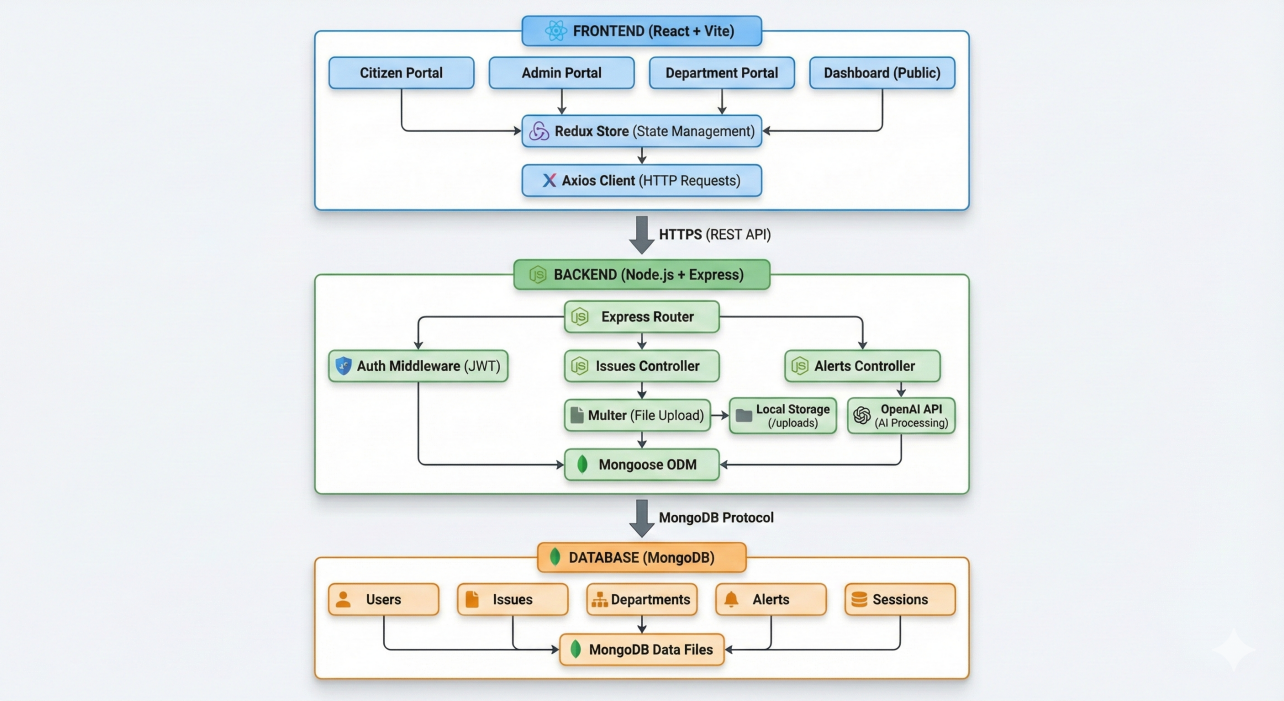

Figure 1: Solution Diagram

The system is designed using a robust **Three-Tier Architecture**, facilitating a clean separation of concerns between the user interface, business logic, and data storage.

1. Frontend Layer (React + Vite)

This layer acts as the client-side interface, hosting specific portals for Citizens, Admins, and Departments, alongside a Public Dashboard.

- **State Management:** Utilizes `Redux Store` for centralized state handling.
- **Network:** Deploys an `Axios Client` to manage HTTPS requests.

2. Backend Layer (Node.js + Express)

The core application logic is hosted here. An `Express Router` directs traffic through `JWT Auth Middleware` to specific controllers.

- **Integrations:** Includes `Multer` for local file uploads and the `OpenAI API` for AI text processing.
- **Data Abstraction:** Uses `Mongoose ODM` for structured database interactions.

3. Database Layer (MongoDB)

The persistent storage layer uses the MongoDB protocol to manage collections such as Users, Issues, Departments, Alerts, and Sessions.

Data Flow: The architecture ensures a seamless data journey:
User Interaction → *State Management* → *REST API Request* → *Authentication* → *Business Logic* → *Database Persistence.*

Persona and User Journey

Mention the stakeholders involved in the project and the journey of each stakeholder in some reasonable diagram view. Provide a few screenshots of the Project. Explain in detail

Software/Hardware

Technology Stack

Frontend Technologies

Technology	Version	Purpose
React	18.2.0	UI library for building interactive user interfaces
Vite	5.1.0	Next-generation frontend build tool
Redux Toolkit	2.2.1	State management
React Router DOM	6.22.3	Client-side routing
Axios	1.6.7	HTTP client for API requests
Leaflet	1.9.4	Interactive maps
React-Leaflet	4.2.1	React components for Leaflet maps
Lucide React	0.263.1	Icon library

Backend Technologies

Technology	Version	Purpose
Node.js	18+ LTS	JavaScript runtime
Express	4.19.2	Web application framework
MongoDB	6.0+	NoSQL database
Mongoose	8.5.1	MongoDB object modeling
OpenAI API	6.9.1	AI-powered text processing
JWT	9.0.2	Authentication tokens
Bcrypt.js	2.4.3	Password hashing
Multer	1.4.5	File upload handling

Hardware Requirements

Minimum Requirements

Component	Specification
Processor	Dual-core 2.0 GHz or equivalent
RAM	4 GB
Storage	2 GB free disk space
Operating System	Windows 10, macOS 10.15+, or Ubuntu 20.04+
Network	Internet connection for API calls

Recommended Requirements

Local Hosting Instructions

Prerequisites

Before setting up CivicVoice locally, ensure the following are installed:

1. **Node.js (v18 or higher)**: Verify installation with `node --version`. Download from `https://nodejs.org/` if not installed.
2. **MongoDB (v6.0 or higher)**: Either install MongoDB Community Server locally from `https://www.mongodb.com/try/do` or create a free MongoDB Atlas cloud account at `https://www.mongodb.com/atlas`.
3. **Git**: Verify with `git --version`. Download from `https://git-scm.com/` if needed.
4. **OpenAI API Key**: Obtain from `https://platform.openai.com/api-keys` for AI-powered text processing features.

Component	Specification
Processor	Quad-core 2.5 GHz or better
RAM	8 GB or more
Storage	5 GB SSD
Operating System	Latest stable version of Windows, macOS, or Linux
Network	Broadband internet connection

Installation Steps

Step 1: Clone the Repository

```
git clone https://github.com/varunmodi18/civicvoice.git
cd civicvoice
```

Step 2: Backend Setup

```
cd backend
npm install
cp .env.example .env
```

Step 3: Configure Environment Variables

Edit the `.env` file with the following configuration:

- `MONGO_URI`: MongoDB connection string (e.g., `mongodb://127.0.0.1:27017/civicvoice` for local or MongoDB Atlas URI)
- `JWT_SECRET`: Strong random string for JWT token generation
- `PORT`: 4000 (default backend port)
- `CLIENT_ORIGINS`: `http://localhost:5173,http://localhost:5174`
- `OPENAI_API_KEY`: Your OpenAI API key

Step 4: Seed the Database

```
npm run seed
```

This creates sample admin, citizen, and department accounts with initial data.

Step 5: Start Backend Server

```
npm run dev
```

The backend server will start at `http://localhost:4000`.

Step 6: Frontend Setup

Open a new terminal window:

```
cd frontend
npm install
```

Step 7: Start Frontend Development Server

```
npm run dev
```

The frontend application will be accessible at `http://localhost:5173`.

Default Login Credentials

Troubleshooting

- **MongoDB Connection Issues**: Ensure MongoDB service is running using `sudo systemctl start mongod` (Linux/macOS) or `net start MongoDB` as Administrator (Windows).
- **Port Already in Use**: Kill processes using `npx kill-port 4000` for backend or `npx kill-port 5173` for frontend.
- **OpenAI API Errors**: Verify API key validity, check sufficient credits, and ensure proper permissions.
- **CORS Errors**: Verify `CLIENT_ORIGINS` in `.env` matches frontend URL and clear browser cache.

Role	Email	Password
Admin	admin@civicvoice.local	Admin@123
Citizen 1	citizen1@civicvoice.local	Citizen@123
Citizen 2	citizen2@civicvoice.local	Citizen@123
Citizen 3	citizen3@civicvoice.local	Citizen@123
Roads Department	roads@civicvoice.local	Dept@123
Water Department	water@civicvoice.local	Dept@123
Power Department	power@civicvoice.local	Dept@123

Contribution

This project was developed collaboratively by all team members with distributed responsibilities across frontend, backend, database design, AI integration, and documentation. Each member contributed to multiple aspects of the system to ensure comprehensive understanding and implementation.

Aviral Tyagi (2025201086) focused on frontend development, implementing React components for the citizen complaint interface and interactive map integration using Leaflet. Contributions include UI/UX design, Redux state management setup, and responsive layout implementation.

Shivam Patel (2025202030) worked on backend API development, designing and implementing RESTful endpoints using Express.js. Key contributions include authentication middleware with JWT, complaint routing logic, and integration with MongoDB for data persistence.

Vadla Shiva Kishan (2025202036) contributed to database schema design using Mongoose, developed the seeding scripts for initial data population, and implemented department management features. Additional work includes database optimization and query performance enhancement.

Varun Modi (2025202040) led the OpenAI API integration for AI-powered text processing and complaint summarization. Contributions include conversational interface development, prompt engineering for complaint standardization, and coordinating overall system architecture and GitHub repository management.

Harsh Jaiswal (2025204011) focused on admin dashboard development, user role management, and complaint tracking features. Additional contributions include file upload functionality using Multer, testing and debugging, and comprehensive project documentation.

The project repository contains detailed commit history showing incremental development, code reviews, and collaborative problem-solving. All team members participated in testing, bug fixes, and documentation throughout the development lifecycle. The distribution of commits reflects balanced contributions with approximately 15-25 commits per member, demonstrating active and consistent participation from all team members.

References

1. React - A JavaScript library for building user interfaces.
   https://reactjs.org/
2. Vite - Next Generation Frontend Tooling.
   https://vitejs.dev/
3. Redux Toolkit - The official, opinionated, batteries-included toolset for efficient Redux development.
   https://redux-toolkit.js.org/
4. React Router DOM - Declarative routing for React web applications.
   https://reactrouter.com/en/main
5. Axios - Promise based HTTP client for the browser and Node.js.
   https://axios-http.com/
6. Leaflet - An open-source JavaScript library for mobile-friendly interactive maps.
   https://leafletjs.com/

7. React-Leaflet - React components for Leaflet maps.
   https://react-leaflet.js.org/
8. Node.js - JavaScript runtime built on Chrome's V8 JavaScript engine.
   https://nodejs.org/
9. Express - Fast, unopinionated, minimalist web framework for Node.js.
   https://expressjs.com/
10. MongoDB - The most popular NoSQL database.
    https://www.mongodb.com/
11. Mongoose - Elegant MongoDB object modeling for Node.js.
    https://mongoosejs.com/
12. OpenAI API - Access to OpenAI's powerful AI models.
    https://platform.openai.com/
13. JSON Web Tokens (JWT) - An open standard for securely transmitting information between parties.
    https://jwt.io/