**Full Stack Development with MERN Project Documentation format**

# 1. Introduction

* **Project Title:** [Citizen AI – Intelligent Citizen Engagement Platform]
* **Team Members:** **Team Leader :** L Bhavani

**Team member :** K Ch V S Sidhardha

# 2. Project Overview

* **Purpose:** Citizen AI is designed to enhance citizen engagement by providing an intelligent platform that facilitates communication between citizens and local authorities. The goal is to streamline issue reporting, feedback collection, and information dissemination to improve community services and governance.

**Features:** User registration and authentication

* Issue reporting with categorization and status tracking
* Real-time notifications and updates
* Admin dashboard for managing reports and user queries
* AI-powered suggestions for issue resolution
* Responsive UI for mobile and desktop users
* .

# 3. Architecture

**Frontend:** Built using **React.js** with functional components and hooks

* State management using Context API or Redux (specify your choice)
* React Router for client-side routing
* Responsive design with CSS frameworks like Bootstrap or Material-UI

.

**Backend:** Developed with **Node.js** and **Express.js**

* RESTful API endpoints for CRUD operations
* Middleware for authentication, validation, and error handling
* Integration with external AI services (if any)

.

**Database:** **MongoDB** as the NoSQL database

* Schema design includes collections for Users, Reports, Notifications, and Admins
* Mongoose ODM for schema modeling and interaction
* Indexing for efficient querying

# 4. Setup Instructions

**Prerequisites:** Node.js (version X.X.X or higher)

* MongoDB (local or cloud instance)
* npm or yarn package manager
* Git (for cloning repository)
* **Installation:** <https://github.com/Bhavani209>.
* 2.Navigate to the project directory: cd citizen-ai

3. Install backend dependencies: cd server

npm install

4. Install frontend dependencies: cd ../client

npm install

Set up environment variables:  
 Create a .env file in the server directory with the following variables:

PORT=5000

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_jwt\_secret\_key

# 5. Folder Structure

* **Client:** client/
* ├── public/
* ├── src/
* │ ├── components/ # Reusable UI components
* │ ├── pages/ # Page components
* │ ├── context/ # State management (Context API or Redux)
* │ ├── services/ # API calls
* │ ├── App.js
* │ ├── index.js
* └── package.json
* **Server:** server/
* ├── controllers/ # Request handlers
* ├── models/ # Mongoose schemas
* ├── routes/ # API routes
* ├── middleware/ # Authentication, error handlers
* ├── config/ # DB connection and config files
* ├── utils/ # Utility functions
* ├── server.js # Entry point

└── package.json.

# 6. Running the Application

• Provide commands to start the frontend and backend servers locally.

o **Frontend:** cd client

npm start

The frontend will run on http://localhost:3000

**Backend:** cd server

npm start .

The backend server will run on http://localhost:5000

# 7. API Documentation

* Document all endpoints exposed by the backend.
* Include request methods, parameters, and example responses.

# 8. Authentication

* Uses **JWT (JSON Web Tokens)** for stateless authentication.
* On login, server issues a JWT token which is stored on the client side (localStorage or HTTP-only cookie).
* Protected routes verify the token using middleware before granting access.
* Role-based access control implemented for users and admins.

1. **User Interface** 
   * Provide screenshots or GIFs showcasing different UI features.
2. **Testing**

Unit testing with **Jest** and **React Testing Library** for frontend components.

Backend API testing using **Mocha** and **Chai** or **Supertest**.

Manual testing for UI responsiveness and usability.

1. **Screenshots or Demo** 
   * https://your-demo-link.com/
2. **Known Issues**

Some UI elements may not be fully responsive on smaller devices.

Notification system delay under heavy load.

Occasional token expiration issues requiring re-login.

# 13. Future Enhancements

* Integration of AI chatbot for instant citizen support.
* Mobile app development for better accessibility.
* Advanced analytics dashboard for administrators.
* Multi-language support for wider reach.
* Offline mode for report submission without internet.