SMART CITY MANAGEMENT PROJECT

Smart City Lucknow - IoT Dashboard & Management System

A comprehensive smart city management platform built with React and Node.js that provides real-time monitoring and management of various urban systems including traffic, energy, waste management, air quality, and emergency services.

❖ Table of Contents

❖ Overview

Smart City Lucknow is a modern web application that simulates a smart city management system. It provides real-time monitoring of various urban infrastructure components through IoT sensors and offers an intuitive dashboard for citizens and administrators to interact with city services.

This comprehensive documentation covers all aspects of your Smart City project, including:

1. **Complete Overview** - What the project does and its capabilities
2. **Detailed Features** - All functionality broken down by user type
3. **Technology Stack** - All dependencies with versions
4. **Installation Guide** - Step-by-step setup instructions
5. **Configuration** - Environment variables and settings
6. **Project Structure** - Detailed folder and file organization
7. **API Documentation** - All endpoints and their purposes
8. **Usage Instructions** - How to run and use the application
9. **Development Guidelines** - How to contribute and extend
10. **Deployment Options** - Production deployment strategies

The documentation is structured to help new users understand the project quickly and get it running on their systems. It includes all the important dependencies, setup requirements, and usage instructions needed for a complete understanding of the Smart City

➢ Key Capabilities:

- **Real-time Data Monitoring**: Live updates from IoT sensors
- **Interactive Dashboard**: Comprehensive overview of city metrics
- **AI-Powered Chatbot**: Intelligent assistant for citizen queries
- **Admin Panel**: Complete system management interface
- **Responsive Design**: Works seamlessly across all devices
- **Caching System**: Optimized performance with Redis caching

❖ Features

➢ Dashboard Features (Protected)

- **Traffic Monitoring**: Real-time traffic intensity and status
- **Energy Management**: Power consumption tracking by sector
- **Waste Management**: Bin level monitoring across city zones
- **Air Quality**: AQI monitoring and pollution alerts
- **Emergency Services**: Incident reporting and management

➢ AI Assistant

- **Smart Chatbot**: Context-aware responses about city services
- **Multi-category Support**: Traffic, air quality, energy, waste, emergency, tourism

- **Real-time Suggestions**: Dynamic help recommendations

➢ Admin Features

- **User Management**: Complete user administration
- **Incident Management**: Emergency response coordination
- **System Monitoring**: Cache statistics and performance metrics
- **Data Analytics**: Comprehensive reporting and insights

❖ Technology Stack

➢ Backend

- **Node.js** (v18+) - Runtime environment
- **Express.js** (v5.1.0) - Web framework
- **MongoDB** (v8.18.0) - Database with Mongoose ODM
- **Socket.IO** (v4.0.0) - Real-time communication
- **JWT** (v9.0.2) - Authentication
- **Redis** (v4.7.1) - Caching layer
- **Node-cron** (v4.2.1) - Scheduled tasks
- **bcryptjs** (v3.0.2) - Password hashing

➢ Frontend

- **React** (v19.1.1) - UI framework
- **Vite** (v7.1.2) - Build tool and dev server
- **React Router DOM** (v7.8.2) - Client-side routing
- **Tailwind CSS** (v4.1.13) - Styling framework
- **DaisyUI** (v5.1.10) - Component library
- **Framer Motion** (v12.23.12) - Animations
- **React Leaflet** (v5.0.0) - Interactive maps
- **Recharts** (v3.1.2) - Data visualization
- **Socket.IO Client** (v4.0.0) - Real-time updates

➢ Development Tools

- **ESLint** - Code linting
- **Prettier** - Code formatting
- **Nodemon** - Development server auto-restart

❖ Prerequisites

Before you begin, ensure you have the following installed:

- **Node.js** (v18.0.0 or higher)
- **npm** (v8.0.0 or higher)

- **MongoDB** (v5.0 or higher)
- **Redis** (v6.0 or higher) - Optional but recommended for caching

➢ System Requirements

- **RAM**: Minimum 4GB, Recommended 8GB
- **Storage**: At least 2GB free space
- **OS**: Windows 10/11, macOS 10.15+, or Linux Ubuntu 18.04+

❖ Installation

➢ Clone the Repository

```
git clone <repository-url>
cd "SMART CITY - 2"
```

➢ Backend Setup

```
# Navigate to backend directory
cd backend
# Install dependencies
npm install
# Create environment file
cp .env.example .env
# Edit .env with your configuration (see Configuration section)
```

➢ Frontend Setup

```
# Navigate to frontend directory (from project root)
cd frontend
# Install dependencies
npm install
```

➢ Database Setup

```
# Start MongoDB service
# Windows: Start MongoDB service from Services
# macOS: brew services start mongodb-community
# Linux: sudo systemctl start mongod
# Create database (optional - will be created automatically)
# Connect to MongoDB and create 'smartcity' database
```

➢ Redis Setup (Optional)

```
# Windows: Download and install Redis from official website
# macOS: brew install redis && brew services start redis
# Linux: sudo apt-get install redis-server && sudo systemctl start redis
```

❖ Configuration

➢ Backend Environment Variables

Create a `.env` file in the backend directory:

```
# Database
MONGO_URI=mongodb://localhost:27017/smartcity
# JWT Configuration
JWT_SECRET=your_super_secret_jwt_key_here_make_it_long_and_secure
# Server Configuration
PORT=5000
NODE_ENV=development
# Redis Configuration (Optional)
REDIS_URL=redis://localhost:6379
# CORS Configuration
FRONTEND_URL=http://localhost:5173
```

➢ Frontend Configuration

The frontend automatically connects to the backend API. Ensure the backend is running on the configured port.

➢ Project Structure

```
SMART CITY - 2/
├── backend/ # Backend application
│   ├── config/
│   │   └── db.js # Database configuration
│   ├── controllers/ # Route controllers
│   │   ├── authController.js # Authentication logic
│   │   ├── chatbotController.js # AI chatbot logic
│   │   ├── complaintController.js # Complaint management
│   │   ├── incidentController.js # Emergency incidents
│   │   └── trafficController.js # Traffic data management
│   ├── middleware/
│   │   └── authMiddleware.js # JWT authentication middleware
│   ├── models/ # MongoDB schemas
│   │   ├── user.js # User model
│   │   ├── traffic.js # Traffic data model
│   │   ├── energy.js # Energy consumption model
│   │   ├── waste.js # Waste management model
│   │   ├── Air.js # Air quality model
│   │   ├── Incident.js # Emergency incidents model
│   │   ├── Complaint.js # Citizen complaints model
│   │   └── Alert.js # System alerts model
│   ├── routes/ # API routes
│   │   ├── admin/ # Admin-specific routes
│   │   ├── authRoutes.js # Authentication routes
│   │   ├── trafficRoutes.js # Traffic API endpoints
│   │   ├── energyRoutes.js # Energy API endpoints
│   │   ├── wasteRoutes.js # Waste management API
│   │   ├── airRoutes.js # Air quality API
│   │   ├── incidentRoutes.js # Emergency API
│   │   ├── complaintRoutes.js # Complaints API
│   │   └── chatbotRoutes.js # AI chatbot API
│   ├── services/
│   │   └── cacheService.js # Redis caching service
│   ├── utils/
```

```
│   │   └── cacheUtils.js # Cache utility functions
│   ├── server.js # Main server file
│   ├── start-backend.bat # Windows startup script
│   └── package.json # Backend dependencies
├── frontend/ # Frontend application
│   ├── public/
│   │   ├── images/ # Static images
│   │   └── vite.svg # Vite logo
│   ├── src/
│   │   ├── api/ # API configuration
│   │   │   └── axios.js # Axios HTTP client setup
│   │   ├── assets/ # Static assets
│   │   │   └── pages/ # Page components
│   │   ├── components/ # Reusable components
│   │   │   ├── Navbar.jsx # Navigation component
│   │   │   ├── Sidebar.jsx # Dashboard sidebar
│   │   │   ├── Chatbot.jsx # AI assistant component
│   │   │   ├── EnhancedChatbot.jsx # Advanced chatbot
│   │   │   ├── ProtectedRoute.jsx # Route protection
│   │   │   └── CacheManager.jsx # Cache management UI
│   │   ├── context/
│   │   │   └── AuthContext.jsx # Authentication context
│   │   ├── layouts/
│   │   │   └── DashboardLayout.jsx # Dashboard layout
│   │   ├── pages/ # Page components
│   │   │   ├── admin/ # Admin pages
│   │   │   ├── Home.jsx # Landing page
│   │   │   ├── Dashboard.jsx # Main dashboard
│   │   │   ├── Login.jsx # Login page
│   │   │   ├── Register.jsx # Registration page
│   │   │   ├── Traffic.jsx # Traffic monitoring
│   │   │   ├── Energy.jsx # Energy management
│   │   │   ├── Waste.jsx # Waste monitoring
│   │   │   ├── Air.jsx # Air quality monitoring
│   │   │   ├── Emergency.jsx # Emergency services
│   │   │   ├── FamousPlaces.jsx # Tourist attractions
│   │   │   └── Contact.jsx # Contact information
│   │   ├── services/ # Frontend services
│   │   │   ├── api.js # API service layer
│   │   │   ├── cachedApi.js # Cached API calls
│   │   │   ├── cacheService.js # Frontend caching
│   │   │   └── socket.js # Socket.IO client
│   │   ├── utils/
│   │   │   └── auth.js # Authentication utilities
│   │   ├── App.jsx # Main app component
│   │   ├── App.css # Global styles
│   │   ├── index.css # Base styles
│   │   └── main.jsx # Application entry point
│   ├── talwind.config.js # Tailwind CSS configuration
│   ├── vite.config.js # Vite configuration
│   └── package.json # Frontend dependencies
└── README.md # This documentation
```

❖ API Documentation

➢ Authentication Endpoints

```
POST /api/auth/register # User registration
POST /api/auth/login # User login
POST /api/auth/forgot # Password reset request
```

➢ Data Monitoring Endpoints

```
GET /api/traffic # Get traffic data
GET /api/energy # Get energy consumption data
GET /api/waste # Get waste management data
```

```
GET /api/air # Get air quality data
GET /api/incidents # Get emergency incidents
```

➢ Admin Endpoints

```
GET /api/admin/users # Get all users
POST /api/admin/users # Create new user
PUT /api/admin/users/:id # Update user
DELETE /api/admin/users/:id # Delete user
```

➢ Chatbot Endpoints

```
POST /api/chatbot/chat # Send message to chatbot
GET /api/chatbot/suggestions # Get conversation suggestion
```

➢ Cache Management

```
GET /api/cache/stats # Get cache statistics
POST /api/cache/clear # Clear all cache
```

❖ Usage

➢ Starting the Application

Option 1: Manual Start

```
# Terminal 1 - Start Backend
cd backend
npm start
# Terminal 2 - Start Frontend
cd frontend
npm run dev
```

Option 2: Windows Batch Script

```
# Double-click start-backend.bat in the backend folder
# Then start frontend manually
cd frontend
npm run dev
```

➢ Accessing the Application

- **Frontend**: http://localhost:5173
- **Backend API**: http://localhost:5000
- **Admin Panel**: http://localhost:5173/admin

➢ Default Credentials

- **Admin User**: admin@smartcity.com / admin123
- **Regular User**: user@smartcity.com / user123

❖ Development

➢ Development Scripts

Backend

```
npm start # Start production server
npm run dev # Start development server with nodemon
```

Frontend

```
npm run dev # Start development server
npm run build # Build for production
npm run preview # Preview production build
npm run lint # Run ESLint
```

➢ Code Style

- **ESLint**: Configured for React and Node.js
- **Prettier**: Automatic code formatting
- **Conventional Commits**: Follow semantic commit messages

➢ Adding New Features

1. Create feature branch from `main`
2. Implement changes with tests
3. Update documentation
4. Submit pull request

❖ Deployment

➢ Production Environment Variables

```
NODE_ENV=production
MONGO_URI=mongodb://your-production-db-url
JWT_SECRET=your-production-jwt-secret
PORT=5000
REDIS_URL=redis://your-redis-url
```

➢ Build for Production

```
# Backend
cd backend
npm install --production
# Frontend
cd frontend
npm run build
```

➢ Deployment Options

- **Heroku**: Easy deployment with buildpacks

- **AWS**: EC2 with RDS and ElastiCache
- **DigitalOcean**: Droplet with managed databases
- **Docker**: Containerized deployment

❖ Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes ('git commit -m 'Add amazing feature')
4. `Push to the branch` (git push origin feature/amazing-feature`)
5. Open a Pull Request

➢ Development Guidelines

- Follow existing code style
- Add tests for new features
- Update documentation
- Ensure all tests pass

❖ License

This project is licensed under the MASAI School - see the LICENSE file for details.

❖ Support

For support and questions:

- Create an issue in the repository
- Contact the development team
- Check the documentation

❖ Future Enhancements

- **Mobile App**: React Native application
- **Advanced Analytics**: Machine learning predictions
- **IoT Integration**: Real sensor data integration
- **Multi-language Support**: Internationalization
- **Advanced Security**: OAuth2, 2FA
- **Performance Monitoring**: APM integration

NAME :- BHANU PRATAP SINGH
STUDENT CODE : IITG24061512

Generated by 1.14.0