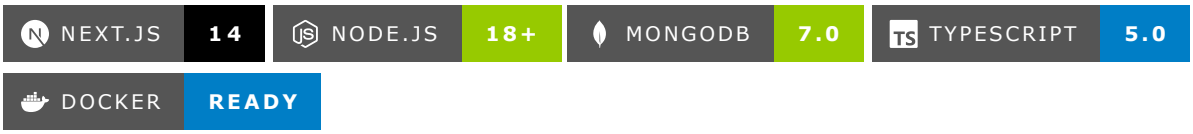
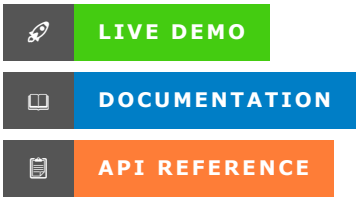
TRINETRA

AI-Powered Urban Community Safety Platform



Empowering communities through intelligent incident reporting and real-time safety insights



Overview

Real

TRINETRA is a cutting-edge, full-stack civic technology platform that revolutionizes urban safety through community engagement. Built with modern technologies and enterprise-grade architecture, it empowers citizens to report, verify, and engage with urban incidents in real-time.

Key Highlights

Modern Architecture

- Next.js 14 with App Router
- Production-ready Node.js backend
- MongoDB with geospatial indexing
- Redis caching & session management
- Docker containerization

Enterprise Security

- JWT authentication with refresh tokens
- Rate limiting & DDoS protection
- Input validation & sanitization
- File upload security
- CORS & security headers

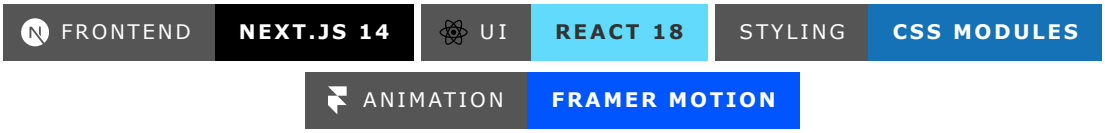
AI-Powered Features

- Real-time incident verification
- Confidence scoring algorithm
- Smart route optimization
- Automated threat assessment
- Community reputation system

📱 User Experience

- Mobile-first responsive design
- Dark/Light theme support
- Real-time notifications
- Interactive maps integration
- Smooth animations & transitions

🎯 Frontend - Next.js Application



📱 Modern Web Application

A **mobile-first**, responsive web application built with Next.js 14 (App Router) providing an intuitive interface for community safety engagement. Features smooth animations, real-time updates, and comprehensive theme support.

```
Mobil
```

✨ Key Features

🎬 Splash Screen

Animated logo with video introduction and smooth transitions

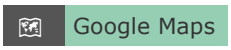
🔐 Authentication

Secure signup/login with JWT integration and session management

📱 Urban Thread Feed

Community incident reports with filtering and reactions

Interactive Maps



Smart routing (fastest/eco/safest routes) with real-time data

Incident Reporting

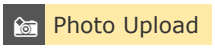


Photo upload, geolocation, and category-based submission

User Profiles

Statistics, contribution history, and reputation tracking

Theme System

Smooth transitions and persistent storage

Responsive Design

Optimized for all device sizes

Performance

SSR, lazy loading, and code splitting

Frontend Technology Stack

Technology	Version	Purpose	Badge
Next.js	14.x	App Router, SSR, Performance	
React	18.x	UI Components, Hooks	
TypeScript	5.x	Type Safety, DX	
Framer Motion	Latest	Animations, Transitions	
Lucide React	Latest	Icons, UI Elements	
CSS Modules	Built-in	Styling, Theming	

Frontend Architecture

ARCHITECTURE MODULAR STRUCTURE CLEAN CODE

```
frontend/
├── app/
│   ├── components/     # Reusable UI components
│   │   ├── ui/            # Base components (Button, Input, Modal)
│   │   ├── forms/         # Form components (Auth, Profile)
│   │   └── layout/        # Layout components (Header, Nav)
│   ├── contexts/       # React Context providers
│   │   ├── AuthContext.tsx    # Authentication state
│   │   └── ThemeContext.tsx   # Theme management
│   ├── hooks/          # Custom React hooks
│   │   ├── useAuth.ts         # Authentication logic
│   │   ├── useLocation.ts     # Geolocation services
│   │   └── useTheme.ts        # Theme switching
│   ├── lib/            # API service layer
│   │   ├── api.ts             # HTTP client
│   │   ├── auth.ts            # Auth utilities
│   │   └── utils.ts           # Helper functions
│   ├── pages/          # Route components
│   │   ├── auth/              # Authentication pages
│   │   ├── dashboard/         # Main dashboard
│   │   └── profile/           # User profile
│   ├── types/          # TypeScript definitions
│   ├── globals.css     # Global styles
│   ├── layout.tsx      # Root layout
│   └── page.tsx        # Entry point
├── Configuration Files
│   ├── next.config.js     # Next.js config & proxy
│   ├── .env.local         # Environment variables
│   ├── tsconfig.json      # TypeScript config
│   ├── tailwind.config.js # Tailwind CSS config
│   └── package.json       # Dependencies & scripts
└── public/
    ├── icons/             # App icons
    ├── images/            # Static images
    └── videos/            # Video assets
```

Quick Start Guide

SETUP EASY TIME 5 MINUTES

Prerequisites

```
# Required versions
Node.js >= 18.0.0
npm >= 9.0.0
```

⚡ Installation

```
# 1️⃣ Clone the repository
git clone https://github.com/your-org/trinetra.git
cd trinetra/frontend

# 2️⃣ Install dependencies
npm install

# 3️⃣ Setup environment
cp .env.example .env.local
# Edit .env.local with your configuration

# 4️⃣ Start development server
npm run dev
```

🌐 Available Scripts

Command	Description	Usage
`npm run dev`	Development server	Hot reload on `localhost:3000`
`npm run build`	Production build	Optimized build in `.next/`
`npm run start`	Production server	Serve built application
`npm run lint`	Code linting	ESLint + Prettier
`npm run type-check`	TypeScript check	Validate types

⚡ Backend - Enterprise Node.js API

🏗️ Production-Ready API Server

A **enterprise-grade** RESTful API server built with Node.js and Express, featuring comprehensive security, performance optimization, and intelligent data processing. Designed for high-traffic production environments with advanced monitoring and scaling capabilities.

✨ Core Features

🔐 Security & Authentication

- JWT with refresh token rotation
- Rate limiting & DDoS protection
- Input validation & sanitization
- File upload security
- IP filtering & CORS
- Security headers & CSP

🚀 Performance & Scalability

- Redis caching & session storage
- Database connection pooling
- Compression & response optimization
- Clustering support
- Memory monitoring
- Graceful shutdown

🤖 AI & Intelligence

- Real-time incident verification
- Confidence scoring algorithm
- SerpAPI integration
- Automated threat assessment
- Smart route optimization
- Community reputation system

📊 Monitoring & Logging

- Structured logging with Winston
- Performance metrics
- Health check endpoints
- Error tracking & reporting
- Request correlation IDs
- Prometheus integration

🛠️ Backend Technology Stack

Technology	Version	Purpose	Badge
Node.js	18+	Runtime Environment	
Express.js	4.x	Web Framework	

Technology	Version	Purpose	Badge
MongoDB	7.0+	Database	MongoDB 7.0
Redis	7.2+	Caching & Sessions	Redis 7.2
JWT	Latest	Authentication	JWT Auth
Winston	Latest	Logging	Winston Logging
Joi	Latest	Validation	Joi Validation
PM2	Latest	Process Management	PM2 Process Manager

External Integrations

Service	Purpose	Status
SerpAPI	Real-time data verification	Status Active
Google Maps	Geolocation & routing	Status Active
Google Vision	Image analysis	Status Active
Prometheus	Metrics collection	Status Active
Grafana	Monitoring dashboard	Status Active

Backend Architecture

ARCHITECTURE ENTERPRISE STRUCTURE PRODUCTION READY

```
backend/
├── controllers/          # Business logic & request handlers
│   ├── authController.js       # Authentication & JWT
│   ├── contributeController.js # Incident reporting
│   ├── threadsController.js    # Community feed
│   ├── profileController.js    # User management
│   ├── routesController.js     # Route planning
│   └── commentsController.js   # Comments system
├── models/               # MongoDB schemas
│   ├── User.js                 # User model + geolocation
│   ├── Post.js                 # Incident reports + TTL
│   └── Comment.js              # Community comments
├── routes/               # API endpoint definitions
├── middleware/           # Security & validation
│   ├── auth.js                 # JWT authentication
```

```
│   ├── security.js             # 🛡️ Security middleware
│   ├── performance.js          # ⚡ Performance optimization
│   └── errorHandler.js         # 🚨 Global error handling
├── 🔧 utils/                 # Helper functions
│   ├── confidence.js           # 🤖 AI verification engine
│   └── storage.js              # 📁 File handling utilities
├── 📊 config/                # Configuration
│   ├── database.js             # 🗄️ MongoDB connection
│   └── logger.js               # 📝 Winston logging
├── ☑️ validation/            # Input validation
│   └── schemas.js              # 📋 Joi validation schemas
├── 🐳 Docker Files
│   ├── Dockerfile              # 📦 Production container
│   ├── docker-compose.yml      # 🔧 Multi-service setup
│   └── .dockerignore           # 🚫 Docker ignore rules
├── 🚀 Deployment
│   ├── ecosystem.config.js     # 🔄 PM2 configuration
│   ├── nginx/                  # 🌐 Reverse proxy config
│   └── scripts/                # 🛠️ Deployment scripts
├── 📁 uploads/               # Temporary file storage
├── 📋 logs/                  # Application logs
├── 🌍 .env.example           # Environment template
└── 🎯 server.js              # Application entry point
```

🚀 Backend Setup

```
# Install dependencies
npm install

# Configure environment
cp .env.example .env
# Edit .env with your database URL and API keys

# Start server
node server.js

# Access API
http://localhost:8080
```

🔧 Environment Configuration

Frontend (.env.local)

```
# API Configuration
NEXT_PUBLIC_API_URL=http://localhost:8080
NEXT_PUBLIC_APP_NAME=UrbanThreads
NEXT_PUBLIC_APP_VERSION=1.0.0
```

```
# API Keys
NEXT_PUBLIC_SERPAPI_KEY=your_serpapi_key
NEXT_PUBLIC_GOOGLE_MAPS_API_KEY=your_maps_key

# Feature Flags
NEXT_PUBLIC_ENABLE_LOCATION_SERVICES=true
NEXT_PUBLIC_ENABLE_PHOTO_UPLOAD=true
NEXT_PUBLIC_DEBUG_MODE=true
```

Backend (.env)

```
# Server Configuration
PORT=8080
NODE_ENV=development

# Database
MONGODB_URI=mongodb://127.0.0.1:27017/TRINETRA

# Authentication
JWT_SECRET=your_secure_jwt_secret

# API Keys
SERPAPI_KEY=your_serpapi_key

# File Upload
MAX_FILE_SIZE=5000000
UPLOAD_PATH=./uploads
```

⚠ **Security Note**: Never commit `.env` files with production secrets to version control!

API Integration

The frontend and backend communicate through a proxy configuration in `next.config.js`:

```
async rewrites() {
  return [
    {
      source: '/api/:path*',
      destination: 'http://localhost:8080/api/:path*'
    }
  ]
}
```

Key API Endpoints

- `POST /api/auth/register` - User registration
- `POST /api/auth/login` - User authentication
- `GET /api/threads` - Get community incidents with location filtering
- `POST /api/contribute` - Submit incident reports
- `GET /api/profile/me` - Get user profile and statistics
- `POST /api/routes` - Calculate optimal routes

🚀 Deployment

Frontend Deployment

Recommended: Vercel

```
# Connect your GitHub repository to Vercel
# Configure environment variables in Vercel dashboard
# Automatic deployments on push to main branch
```

Alternative: Traditional Hosting

```
npm run build
npm start
```

Backend Deployment

Options: Heroku, AWS, DigitalOcean, Railway

```
# For Heroku
git push heroku main

# For Docker
docker build -t trinetra-backend .
docker run -p 8080:8080 trinetra-backend
```

Production Checklist

- [ ] Update API URLs in environment variables
- [ ] Configure HTTPS for both frontend and backend
- [ ] Set up MongoDB Atlas for production database
- [ ] Configure CORS for production domains
- [ ] Set up monitoring and logging
- [ ] Configure rate limiting for production load

Security Features

- **JWT Authentication**: Secure token-based authentication
- **Rate Limiting**: API protection against abuse
- **Input Validation**: Comprehensive request sanitization
- **File Upload Security**: Safe handling of user uploads
- **CORS Configuration**: Proper cross-origin resource sharing
- **Environment Variables**: Secure configuration management

Testing

Frontend Testing

```
# Run tests
npm test

# Run tests in watch mode
npm run test:watch

# Test build
npm run build
```

Backend Testing

```
# Test API endpoints
npm test

# Test with specific environment
NODE_ENV=test npm test
```

Contributing

We welcome contributions! Please follow these steps:

1. **Fork** the repository
2. **Create** a feature branch (`git checkout -b feature/amazing-feature`)
3. **Follow** existing code style and patterns
4. **Add** tests for new functionality
5. **Commit** your changes (`git commit -m 'Add amazing feature'`)
6. **Push** to the branch (`git push origin feature/amazing-feature`)
7. **Open** a Pull Request

Development Guidelines

- Write clear, descriptive commit messages
- Add comments for complex logic
- Follow TypeScript best practices
- Test thoroughly before submitting
- Update documentation as needed

🐛 Troubleshooting

Common Issues

Frontend not connecting to backend:

- Verify backend is running on port 8080
- Check `next.config.js` proxy configuration
- Confirm API URL in `.env.local`

Geolocation not working:

- Ensure HTTPS in production
- Check browser permissions
- Verify location services are enabled

Image uploads failing:

- Check file size limits
- Verify upload directory permissions
- Confirm multer configuration

📄 License

This project is licensed under the **MIT License** - see the LICENSE file for details.

🙏 Acknowledgments

- Next.js team for the amazing framework
- MongoDB team for the database platform
- SerpAPI for real-time data verification
- Google Maps for location services
- The open-source community for inspiration

📞 Support

- 📧 **Email**: support@trinetra.com
- 🐛 **Issues**: GitHub Issues
- 📖 **Documentation**: Wiki
- 💬 **Discussions**: GitHub Discussions