# **Full Stack Development with**

# **MERN Project Documentation**

## format

### 1. Introduction

0	<b>Project Title:</b> [TrafficTelligence: Advanced Traffic Volume Estimation with
	Machine Learning] Trans Marchana S Barrathi S Hyanna Salma S Nagardya masad
0	Team Members: S.Revathi , S Umme Salma, S Nagendra prasad, v.subramani
0	Project Overview
0	Purpose: Here is a complete draft of your project documentation based on the structure you provided:
0	Joseph Provinces
0	
0	<b></b>
0	☑ TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning
0	
0	Team Members:
0	S. Revathi, S. Umme Salma, S. Nagendra Prasad, V. Subramani
0	
0	
0	
0	? Project Overview
0	Purpose
0	
0	TrafficTelligence is a smart traffic monitoring and estimation system designed to improve urban traffic management. Leveraging machine learning models, the system analyzes real-time traffic feeds and historical data to estimate vehicle volume, identify congestion patterns, and support predictive traffic planning.
0	Factoria
0	Features
0	Real-time traffic volume estimation using ML models
0	Interactive dashboard for visualizing traffic data
0	Role-based authentication for users and admins
0	API endpoints for feeding camera or sensor data
0	Historical data analytics and visualization
0	Alerts for abnormal traffic patterns

```
0
0
o 2 Architecture
   Frontend (React)
0
   Developed using React.js with functional components and hooks
   UI components designed with Material-UI (MUI) for a responsive and modern
   look
   React Router used for seamless page navigation
0
   Axios used for API calls to backend services
0
0
0
   Backend (Node.js + Express.js)
   RESTful API built with Express.js
0
   Handles authentication, traffic data processing, and ML model integration
0
0
   Implements middleware for error handling and token-based access control
0
0
   Database (MongoDB)
0
   MongoDB used to store traffic data, user info, and session logs
0
   Collections:
0
   users: Stores user credentials and roles
0
   traffic_data: Logs of vehicle count, timestamps, location metadata
0
0
   alerts: Stores congestion or abnormality reports
0
0
0
0
0
   Setup Instructions
0
0
   Prerequisites
0
   Node.js  >= 16.x 
0
   MongoDB installed locally or cloud-hosted (MongoDB Atlas)
   Python 3.9+ (for ML model integration via script or API)
0
   Installation
0
```

# 1. Clone the repository

```
git clone https://github.com/your-username/traffictelligence.git
   cd traffictelligence
0
0
   # 2. Install dependencies
0
   cd client
0
   npm install
0
   cd ../server
0
   npm install
0
0
   # 3. Set up environment variables
0
   # Create a .env file in /server with:
   MONGODB_URI=<your_mongo_uri>
0
   JWT_SECRET=<your_secret_key>
0
   PORT=5000
0
0
0
0
0
   2 Folder Structure
0
0
   Client (React Frontend)
0
0
   client/
0
0
                          # Static assets
        public/
0
        - src/
0
           - components/
                              # Reusable UI components
0
            pages/
                           # Views like Dashboard, Login, etc.
0
            - services/
                           # API handlers via Axios
0
           - context/
                           # Authentication context
0
           - App.js
                           # Entry point
0
0
   Server (Node.js Backend)
0
0
   server/
0
0
                           # Request handlers
        - controllers/
0
                           # Mongoose schemas
        - models/
0
                          # API route definitions
        - routes/
0
                             # Auth and error middleware
        - middleware/
0
                          # Business logic and ML model integration
        - services/
0
                         # Environment variables
        .env
0
                          # Entry point
        - server.js
0
0
0
0
0
   Running the Application
0
0
   Start Backend
0
0
   cd server
0
   npm start
0
0
```

**Start Frontend** 

```
cd client
0
o npm start
   Ensure MongoDB is running and environment variables are configured
   properly.
0
0
0
0
   2 API Documentation
0
0
   Base URL: http://localhost:5000/api
0
0
   Example: POST /auth/login
0
0
   Request:
0
0
    "email": "user@example.com",
0
    "password": "securepass"
0
0
0
   Response:
0
0
    "token": "jwt_token_here",
0
    "user": {
0
     "id": "userId",
0
     "role": "admin"
0
0
0
0
0
0
   2 Authentication
0
   Authentication is handled using JWT tokens
0
0
   Upon login, a token is generated and stored in local storage
0
   Routes are protected with middleware that verifies the token
0
0
   Roles (e.g., user, admin) control access to certain resources
0
0
0
0
0
0
   2 User Interface
0
   Screenshots / GIFs (Insert your media here)
0
   Dashboard View: Traffic volume graphs and live camera feed integration
0
0
   Login Page: Secure login with form validation
0
```

Alert Page: Visual list of recent alerts and notifications

0	
0	
0	
0	<b>?</b> Testing
0	
0	Testing Strategy
0	Unit testing with Jest for backend routes and controllers
0	Component testing with React Testing Library
0	Manual integration testing for ML model predictions
0	The state of the s
0	
0	
0	<del></del>
0	Screenshots or Demo
0	
0	> [Live Demo Link (if hosted)]
0	or
0	Attach screenshots of:
0	Dashboard with traffic charts
0	Dashboard with traffic charts
0	Login/Register pages
0	
0	Real-time data input and output
0	
0	
0	
0	
0	
0	☑ Known Issues
0	E Known Issues
0	ML model performance may drop under low lighting conditions
0	
0	High latency observed when processing large video inputs
0	
0	Frontend form validation can be bypassed without backend strict checks
0	
0	
0	
0	
0	2 Future Enhancements
0	Integrate real-time video feed processing via OpenCV
0	
0	Deploy model to cloud using TensorFlow.js or Flask microservice
0	Add admin panel for user management

```
Implement mobile-responsive design
      Improve prediction accuracy with more training data
   0
   0
   0
      Let me know if you'd like this in Markdown, PDF, or as a GitHub
       README.md file!
      Features: Highlight key features and functionalities.
2. Architecture
      Frontend: Developed using React.js with functional components and hooks
      UI components designed with Material-UI (MUI) for a responsive and modern
       look
   0
       React Router used for seamless page navigation
       Axios used for API calls to backend services
      Backend: RESTful API built with Express.js
   0
       Handles authentication, traffic data processing, and ML model integration
       Implements middleware for error handling and token-based access control
   0
       Database: MongoDB used to store traffic data, user info, and session logs
   0
      Collections:
   0
      users: Stores user credentials and roles
      traffic_data: Logs of vehicle count, timestamps, location metadata
   0
      alerts: Stores congestion or abnormality reports
3. Setup Instructions
      Prerequisites: Node.js >= 16.x
```

MongoDB installed locally or cloud-hosted (MongoDB Atlas)

```
0
      Installation: # 1. Clone the repository
   0
       git clone https://github.com/your-username/traffictelligence.git
   0
       cd traffictelligence
   0
      # 2. Install dependencies
   0
      cd client
   0
      npm install
   0
   0
      cd ../server
   0
      npm install
   0
   0
      # 3. Set up environment variables
   0
      # Create a .env file in /server with:
   0
      MONGODB_URI=<your_mongo_uri>
      JWT_SECRET=<your_secret_key>
      PORT=5000
   0
   0
   0
   0
4. Folder Structure
       Client: client/
         — public/
                              # Static assets
           — src/
             — components/
                                  # Reusable UI components
                               # Views like Dashboard, Login, etc.
               – pages/
             --- services/
                                # API handlers via Axios
              — context/
                               # Authentication context
                               # Entry point
               – App.js
   0
       Server: server/
   0
   0
            controllers/
                               # Request handlers
   0
            - models/
                               # Mongoose schemas
   0
                              # API route definitions
            routes/
   0
            - middleware/
                                 # Auth and error middleware
   0
                               # Business logic and ML model integration
            services/
   0
                             # Environment variables
            - .env
   0
           - server.js
                              # Entry point
```

Python 3.9+ (for ML model integration via script or API)

0

0

0

"role": "admin"

## 6. Running the Application

```
Start Backend
   0
      cd server
      npm start
      Start Frontend
   0
      cd client
      npm start
      Ensure MongoDB is running and environment variables are configured properly.
   0
   0
7. Frontend: cd client
8. npm start
          o Backend: cd server
          o npm start
9. API Documentation
   o http://localhost:5000/api
      Example: POST /auth/login
   0
      Request:
   0
   0
        "email": "user@example.com",
   0
        "password": "securepass"
   0
   0
   0
      Response:
   0
   0
        "token": "jwt_token_here",
   0
        "user": {
   0
         "id": "userId",
```

0 }

#### 10. Authentication

o Authentication is handled using JWT tokens

• Upon login, a token is generated and stored in local storage

o Routes are protected with middleware that verifies the token

Roles (e.g., user, admin) control access to certain resources

0

#### 11. User Interface

Screenshots / GIFs (Insert your media here)

Dashboard View: Traffic volume graphs and live camera feed integration

Login Page: Secure login with form validation

Alert Page: Visual list of recent alerts and notifications

Ni

### 12. Testing

o Testing Strategy

0

• Unit testing with Jest for backend routes and controllers

0

Component testing with React Testing Library

0

Manual integration testing for ML model predictions

0

#### 13. Screenshots or Demo

• [Live Demo Link (if hosted)]

 $\circ$  or

Attach screenshots of:

0

Dashboard with traffic charts

0

Login/Register pages

0

• Real-time data input and output

0

#### 14. Known Issues

o ML model performance may drop under low lighting conditions

0

• High latency observed when processing large video inputs

15. Future Enhancements			
0	Integrate real-time video feed processing via OpenCV		
0			
0	Deploy model to cloud using TensorFlow.js or Flask microservice		
0			
0	Add admin panel for user management		
0			
0	Implement mobile-responsive design		
0			
0	Improve prediction accuracy with more training data		
0			
0			
0			
0			

Frontend form validation can be bypassed without backend strict checks