Full Stack Development with MERN Project Documentation format

1. Introduction

- **Project Title:** [TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning]
- **Team Members: S.Revathi , S Umme Salma, S Nagendra prasad, v.subramani**
-
- **Project Overview**
- **Purpose: Here is a complete draft of your project documentation based on the structure you provided:**
-
-
- **---**
-
- **□ TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning**
-
- **Team Members:**
- **S. Revathi, S. Umme Salma, S. Nagendra Prasad, V. Subramani**
-
-
- **---**
-
- **□ Project Overview**
-
- **Purpose**
-
- **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.**
-
- **Features**
-
- **Real-time traffic volume estimation using ML models**
-
- **Interactive dashboard for visualizing traffic data**
-
- **Role-based authentication for users and admins**
-
- **API endpoints for feeding camera or sensor data**
-
- **Historical data analytics and visualization**
-
- **Alerts for abnormal traffic patterns**
-
-

-
- ---
-
- ## ⍰ Architecture
-
- **Frontend (React)**
-
- **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**
-
- **Axios used for API calls to backend services**
-
-
- **Backend (Node.js + Express.js)**
-
- **RESTful API built with Express.js**
-
- **Handles authentication, traffic data processing, and ML model integration**
-
- **Implements middleware for error handling and token-based access control**
-
-
- **Database (MongoDB)**
-
- **MongoDB used to store traffic data, user info, and session logs**
-
- **Collections:**
-
- **users: Stores user credentials and roles**
-
- **traffic_data: Logs of vehicle count, timestamps, location metadata**
-
- **alerts: Stores congestion or abnormality reports**
-
-
-
-
- ---
-
- ## ⚙Setup Instructions
-
- **Prerequisites**
-
- **Node.js >= 16.x**
-
- **MongoDB installed locally or cloud-hosted (MongoDB Atlas)**
-
- **Python 3.9+ (for ML model integration via script or API)**
-
-
- **Installation**
-
- **# 1. Clone the repository**

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

-
- **cd client**
- **npm start**
-
- **Ensure MongoDB is running and environment variables are configured properly.**
-
-
- **---**
-
- **⍰ API Documentation**
-
- **Base URL: http://localhost:5000/api**
-
- **Example: POST /auth/login**
-
- **Request:**
- **{**
- **"email": "user@example.com",**
- **"password": "securepass"**
- **}**
-
- **Response:**
- **{**
- **"token": "jwt_token_here",**
- **"user": {**
- **"id": "userId",**
- **"role": "admin"**
- **}**
- **}**
-
-
- **---**
-
- **⍰ Authentication**
-
- **Authentication is handled using JWT tokens**
-
- **Upon login, a token is generated and stored in local storage**
-
- **Routes are protected with middleware that verifies the token**
-
- **Roles (e.g., user, admin) control access to certain resources**
-
-
-
- **---**
-
- **⍰ 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**

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

- **Implement mobile-responsive design**
-
- **Improve prediction accuracy with more training data**
-
-
-
- **---**
-
- **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**
-
- **React Router used for seamless page navigation**
-
- **Axios used for API calls to backend services**
-
- **Backend:** RESTful API built with Express.js
-
- Handles authentication, traffic data processing, and ML model integration
-
- Implements middleware for error handling and token-based access control
-
- **Database:** MongoDB used to store traffic data, user info, and session logs
-
- Collections:
-
- users: Stores user credentials and roles
-
- traffic_data: Logs of vehicle count, timestamps, location metadata
-
- alerts: Stores congestion or abnormality reports

3. Setup Instructions

- **Prerequisites:** Node.js >= 16.x
-
- MongoDB installed locally or cloud-hosted (MongoDB Atlas)

-
- Python 3.9+ (for ML model integration via script or API)
-
- **Installation:** # 1. Clone the repository
- git clone https://github.com/your-username/traffictelligence.git
- cd traffictelligence
-
- # 2. Install dependencies
- cd client
- npm install
-
- cd ../server
- npm install
-
- # 3. Set up environment variables
- # Create a .env file in /server with:
- MONGODB_URI=<your_mongo_uri>
- JWT_SECRET=<your_secret_key>
- PORT=5000
-
-
- ---

4. Folder Structure

- **Client:** client/

```
│
├── public/           # Static assets
├── src/
│   ├── components/       # Reusable UI components
│   ├── pages/            # Views like Dashboard, Login, etc.
│   ├── services/         # API handlers via Axios
│   ├── context/          # Authentication context
│   └── App.js            # Entry point
```

- **Server:** server/

```
│
├── controllers/        # Request handlers
├── models/             # Mongoose schemas
├── routes/             # API route definitions
├── middleware/           # Auth and error middleware
├── services/           # Business logic and ML model integration
├── .env                # Environment variables
└── server.js           # Entry point
```

5.

6. Running the Application

- Start Backend
-
- cd server
- npm start
-
- Start Frontend
-
- cd client
- npm start
-
- Ensure MongoDB is running and environment variables are configured properly.
-
-
- ---

7. **Frontend:** `cd client`
8. `npm start`
   - **Backend:** `cd server`
   - `npm start`

9. API Documentation

- http://localhost:5000/api
-
- Example: POST /auth/login
-
- Request:
- {
-   "email": "user@example.com",
-   "password": "securepass"
- }
-
- Response:
- {
-   "token": "jwt_token_here",
-   "user": {
-     "id": "userId",
-     "role": "admin"
-   }

- }

10. Authentication

- Authentication is handled using JWT tokens
-
- Upon login, a token is generated and stored in local storage
-
- Routes are protected with middleware that verifies the token
-
- Roles (e.g., user, admin) control access to certain resources
-

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

- Testing Strategy
-
- Unit testing with Jest for backend routes and controllers
-
- Component testing with React Testing Library
-
- Manual integration testing for ML model predictions
-

13. Screenshots or Demo

- [Live Demo Link (if hosted)]
- or
- Attach screenshots of:
-
- Dashboard with traffic charts
-
- Login/Register pages
-
- Real-time data input and output
-

14. Known Issues

- ML model performance may drop under low lighting conditions
-
- High latency observed when processing large video inputs
-

- Frontend form validation can be bypassed without backend strict checks
-

15. Future Enhancements

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