TrafficTelligence: Project Documentation

1. Introduction

- **Project Title:** Traffic Telligence: Advanced Traffic Volume Estimation with Machine Learning
- **Team Members:**
  - Devireddy Poojitha : Frontend Developer & Backend Developer
  - Dronadula Bhavya Sri : Machine Learning Engineer , Project Manager & Tester

2. Project Overview

- **Purpose:**

TrafficTelligence aims to forecast traffic volumes using historical, real-time, and contextual data such as weather and events. It provides predictive analytics for urban traffic management, enabling smarter decisions for commuters and authorities. ·

Features:

- Real-time traffic volume prediction
- Integration with weather and map APIs
- Historical data visualization
- Web interface for user interaction
- Admin dashboard and data export

3. Architecture

- **Frontend:**
  - Built using React.js
  - Components for input forms, dashboards, and data visualization (charts, graphs) ○ Axios for API communication
- **Backend:**
  - Node.js with Express.js
  - RESTful API endpoints for data interaction and prediction requests
  - ML model served via a Python microservice (Flask)
- **Database:**
  - MongoDB for storing user inputs, prediction logs, and analytics data
  - Mongoose ODM for schema definitions and queries

4. Setup Instructions

- **Prerequisites:**
  - Node.js (v18+)
  - MongoDB (local or cloud)
  - Python (v3.10+)
- **Installation:**
  1. Clone the repository: `git clone https://github.com/your-repo/traffic telligence`
  2. Navigate to client and server folders
  3. Install dependencies: `npm install`
  4. Set up `.env` files with API keys and environment configs
  5. Activate Python virtual environment and install Flask requirements

5. Folder Structure

- **Client (React Frontend):**
  - `/components`: UI Components (Header, Dashboard, InputForm)
  - `/pages`: Route-based pages
  - `/services`: API services using Axios
  - `/styles`: CSS modules or Tailwind
- **Server (Node.js Backend):**
  - `/routes`: API route definitions
  - `/controllers`: Business logic for endpoints
  - `/models`: Mongoose schemas
  - `/utils`: Middleware and helper functions

6. Running the Application

- **Frontend:**
- `cd client`
- `npm start`
- **Backend:**
- `cd server`
- `npm start`
- **ML Microservice (Flask):**
- `cd ml-service`
- `python app.py`

7. API Documentation

Endpoint Method Description

/api/predict POST Sends input data and receives prediction

/api/history GET Returns previously logged predictions

/api/export GET Exports data to CSV or PDF format

Example:

```
POST /api/predict
{
 "location": "Hyderabad",
 "date": "2025-07-01",
 "time": "08:00"
}
```

8. Authentication

· **Method:** Token-based Authentication (JWT) ·
Usage:

- Tokens issued on login
- Middleware verifies tokens for protected routes
- Admin and user roles supported

9. User Interface

· Clean, responsive layout using Tailwind CSS ·
Dark/light mode toggle
· Components:

- Input form for predictions
- Output cards and charts
- Admin dashboard with stats and export options

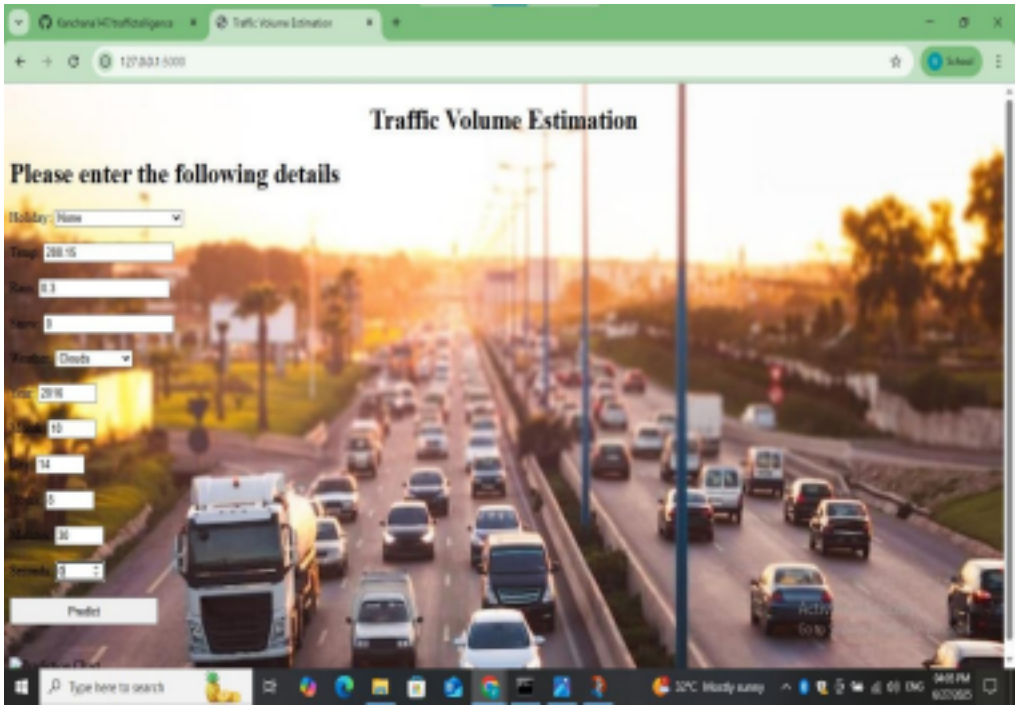
10. Testing

· **Tools Used:**

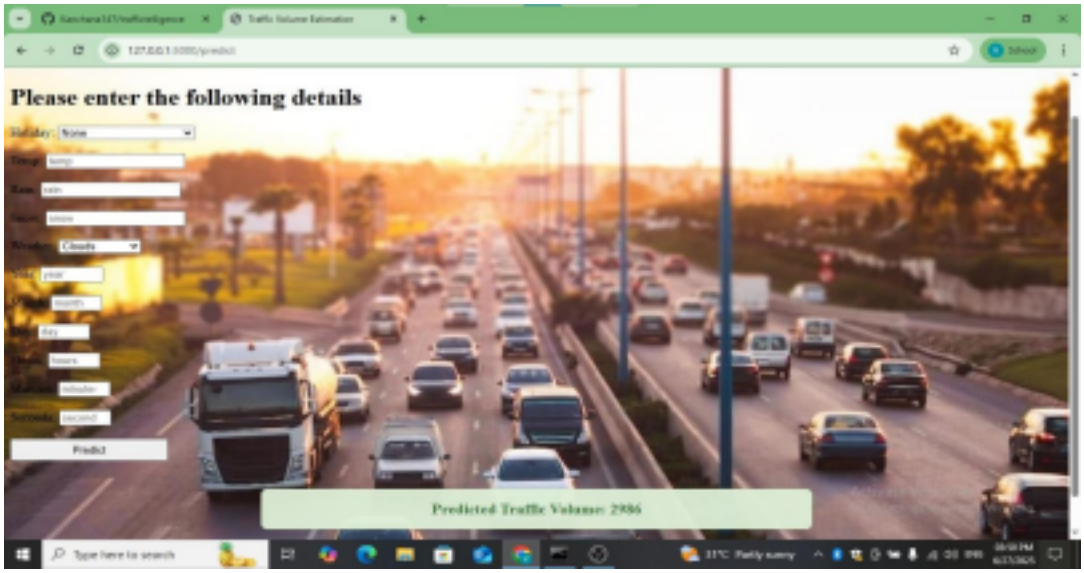
- Jest for React unit testing
- Postman for backend API testing
- Pytest for ML model evaluation

· **Test Cases:**
    - Input validation
    - API response codes
    - Model accuracy and prediction time

11. Screenshots or Demo ·



·



·

· **Live Demo:**
https://drive.google.com/file/d/1YySItyup8PbvFxbefa28fKEDzI1cnw5i/vi

ew?usp=drive_link

12. Known Issues

- Occasional lag on large dataset imports
- Limited dataset coverage in rural regions
- Requires retraining for seasonal data changes

13. Future Enhancements

- Integrate mobile app (React Native)
- Add real-time traffic camera feed analysis
- Smart signal automation via IoT integration
- Role-based access and analytics comparison features