Traffic Telligence: Advanced Traffic Volume Estimation with Machine Learning

---

Team Information

Student Name	Roll Number
Nandanavanam Nikhil (Leader)	22AK1A3061
Shaik Naseer Hussain	22AK1A3058
Praveen Kumar	22AK5A30
Kethineni Nikitha	22AK1A3063

---

1. INTRODUCTION

1.1 Project Overview

"Traffic Telligence" is an advanced machine learning system designed to estimate and forecast traffic volume. By integrating real-time video feeds, weather data, and historical patterns, the system offers accurate traffic predictions, reducing congestion and supporting smart city infrastructure.

1.2 Purpose

The primary objective is to build a predictive and scalable ML-based system to help authorities and commuters make informed traffic decisions. The project provides a low-cost alternative to traditional sensor-based monitoring.

---

2. IDEATION PHASE

2.1 Problem Statement

Urban areas face excessive congestion, leading to lost productivity, emissions, and ineffective traffic management. Current systems lack predictive intelligence and integration with external influencing factors.

2.2 Empathy Map Canvas

- Users: Traffic authorities, city planners, daily commuters, logistics providers
- Pains: Delays, unpredictability, environmental impact
- Needs: Timely predictions, optimized traffic routes, real-time decision making
- Gains: Time and fuel savings, efficient planning, better quality of life

2.3 Brainstorming Highlights

- Utilize structured data from multiple sources (weather, events, holidays)
- Leverage powerful regression models for accurate predictions
- Web-based solution accessible from any device

---

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

Login/Register → Input Parameters → Prediction Generated → Result Displayed

3.2 Solution Requirements

- Dataset: traffic volume.csv
- Models: XGBoost, Random Forest, LightGBM
- Web frontend for user input and result visualization
- OTP-based user authentication

3.3 Data Flow Diagram

1. User submits input
2. Backend Flask API processes data
3. ML model generates prediction
4. Output displayed on UI

3.4 Technology Stack

- Languages: Python, HTML, CSS, JavaScript
- Libraries: Pandas, NumPy, Scikit-learn, XGBoost
- Framework: Flask (API + Web Interface)
- Database: MongoDB (planned)
- Tools: VS Code, GitHub

---

4. PROJECT DESIGN

4.1 Problem-Solution Fit

The solution fits urban mobility needs by transforming static traffic monitoring into proactive prediction using ML.

4.2 Proposed Solution

A multi-module system with:

- Web-based user interface
- Flask backend with ML prediction API
- Model trained and serialized using Pickle/Joblib

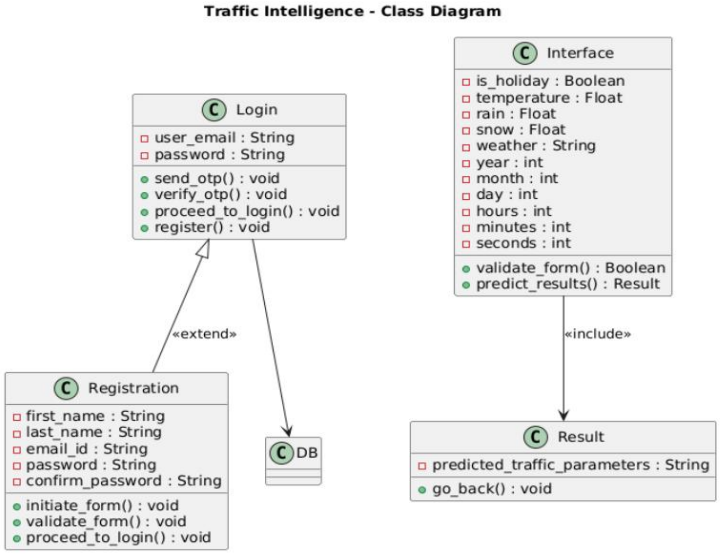
4.3 Solution Architecture

Frontend (HTML Forms) → Flask API → Pre-trained Model → Output to Dashboard

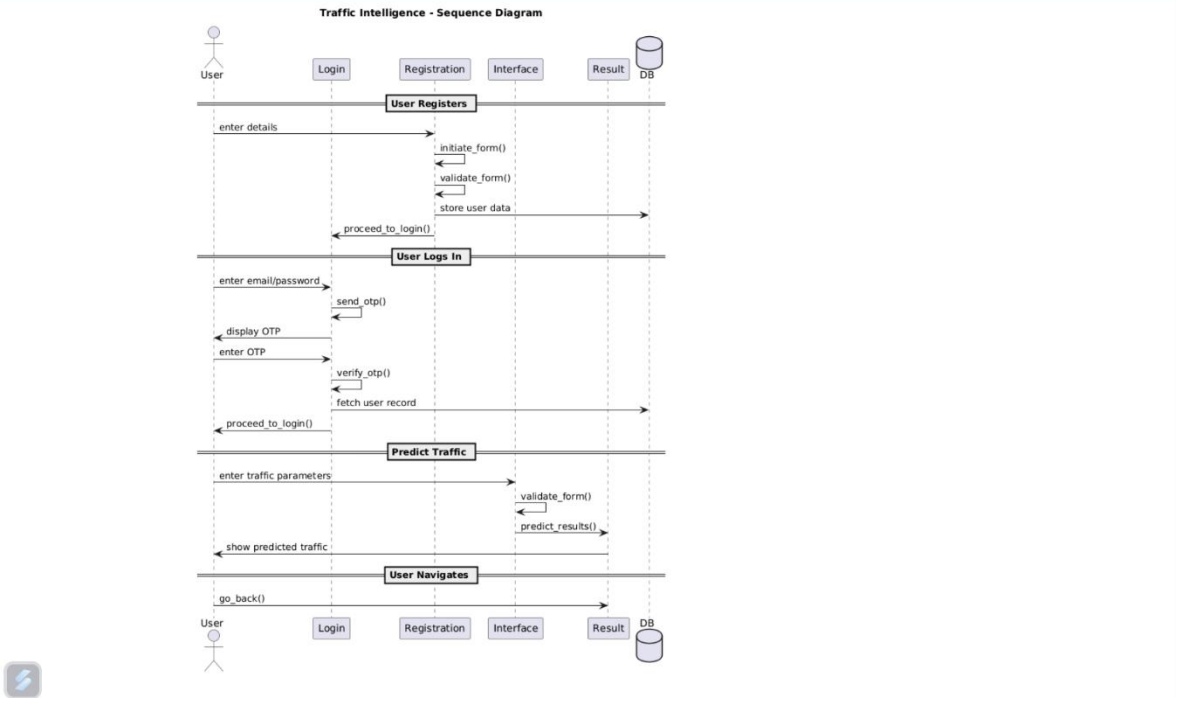
- Templates used: login.html, interface.html, result.html
- Encoders handle categorical inputs for accurate modeling

4.4 UML Diagrams

Class Diagram



Sequence Diagram

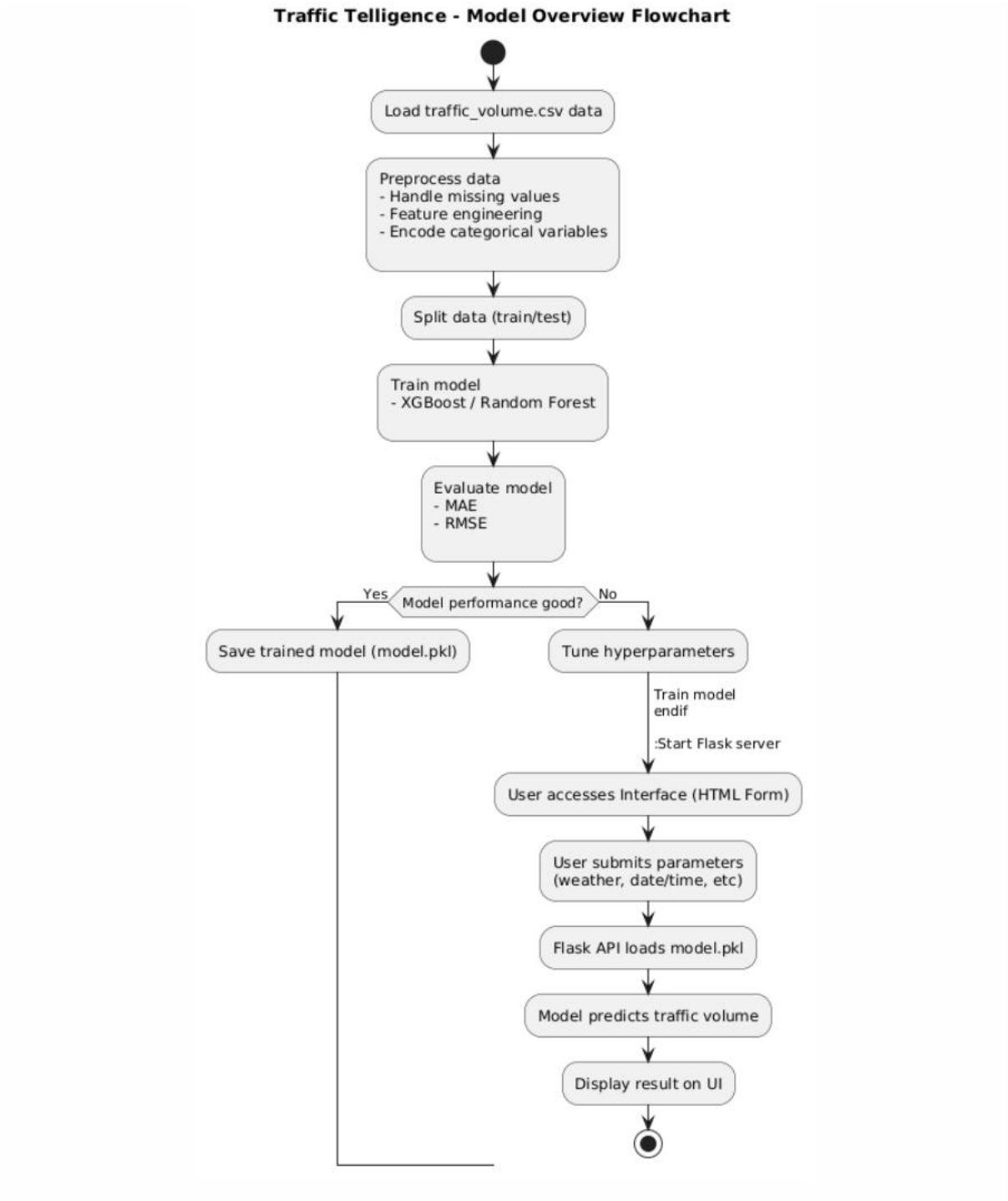


4.5 Model Overview

Our prediction engine was built on historical traffic data, enriched with external features like weather and holidays. We implemented XGBoost and Random Forest Regressors, chosen for their speed and ability to handle large feature sets.

Key Highlights:

- Features: hour, day_of_week, is_holiday, temperature, precipitation
- Preprocessing: One-hot encoding, null handling
- Evaluation: MAE, RMSE for performance validation
- Final model serialized with Pickle


Traffic Telligence - Model Overview Flowchart
Load traffic_volume.csv data
Preprocess data
- Handle missing values
- Feature engineering
- Encode categorical variables
Split data (train/test)
Train model
- XGBoost / Random Forest
Evaluate model
- MAE
- RMSE
Model performance good?
Yes
No
Save trained model (model.pkl)
Tune hyperparameters
Train model
endif
:Start Flask server
User accesses Interface (HTML Form)
User submits parameters
(weather, date/time, etc)
Flask API loads model.pkl
Model predicts traffic volume
Display result on UI

5. PROJECT PLANNING & SCHEDULING

5.1 Agile Sprint Schedule

Sprint	Tasks
1	Data collection and cleaning
2	Model training and tuning
3	Flask backend setup
4	Frontend design
5	Testing & bug fixing
6	Final documentation & deploy

---

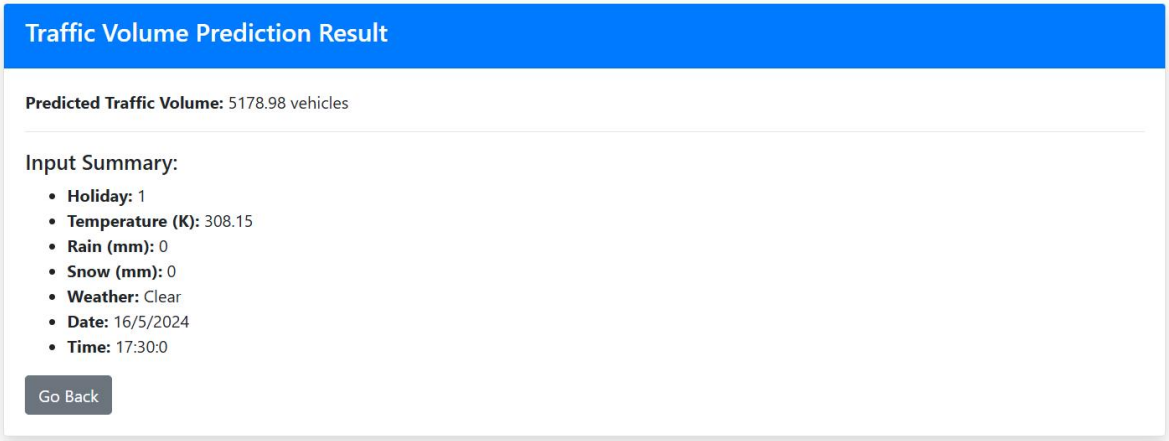
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Testing Overview

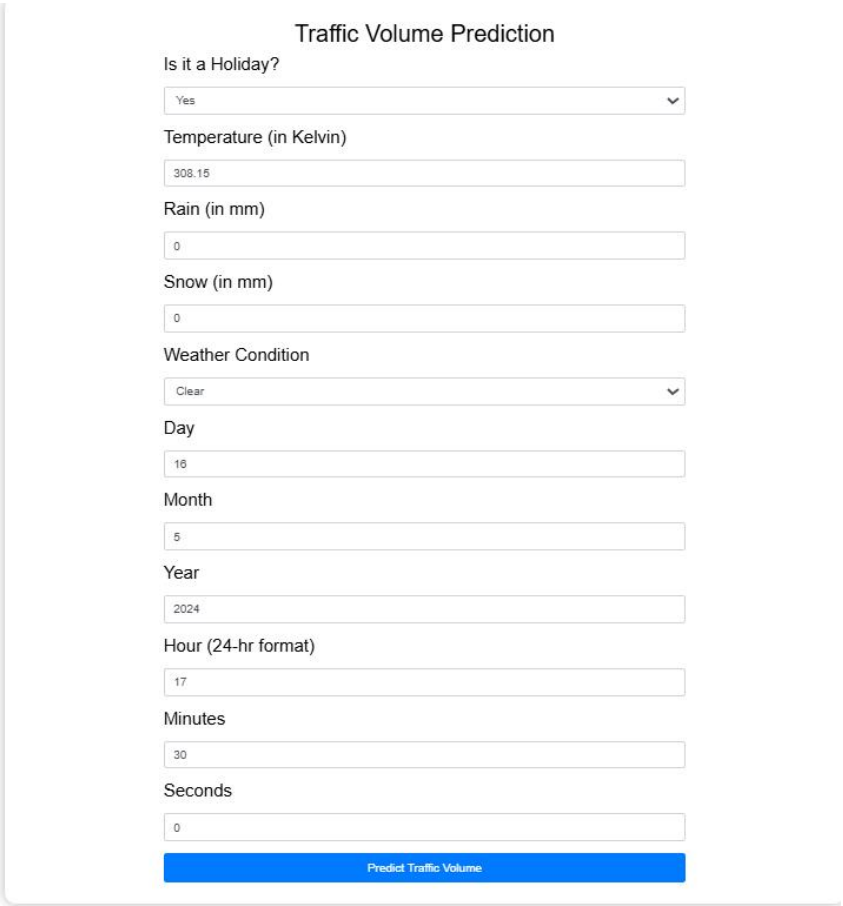
- Verified consistent results across local and deployed models
- Tested performance on edge cases and realistic data
- Fixed logic discrepancy between local script and Flask API
- Integrated OTP system for login security

6.2 Output Screens

Prediction Result Display:



Input Form Interface:



7. RESULTS

- Inputs like weather, time, holiday status accepted via form
- Accurate predictions shown on result dashboard
- Summary of user inputs displayed with each output
- Interface validated across multiple test scenarios

---

8. ADVANTAGES & DISADVANTAGES

Advantages

- ML-powered predictions improve accuracy
- Cost-effective alternative to hardware sensors
- Scalable and extendable system
- Easy to use for both authorities and citizens

Disadvantages

- Accuracy depends on data quality
- Requires regular model updates with new trends

---

9. CONCLUSION

The project successfully demonstrates how machine learning can be applied to traffic forecasting. Its flexible design, interactive interface, and robust logic meet the intended objectives and make it suitable for real-world implementation.

---

10. FUTURE SCOPE

- Real-time data integration (live cameras, GPS feeds)
- Deploy system on mobile platform
- Enhance prediction engine with deep learning (CNNs, RNNs)
- Cloud integration and visualization dashboard

---

11. APPENDIX

- GitHub & Project Demo Link: https://github.com/Nikhil-23-btech/APSCHE.git
- Source Code & Dataset: Included in the repository
- Diagrams and Screenshots: See Design and Testing sections above

---

This report reflects the team's unified effort in conceptualizing, designing, developing, and testing a solution that supports future-ready traffic estimation aligned with the vision of smart urban development.