“Real-Time Traffic Congestion & Accident Prediction”

A Synopsis submitted in partial fulfillment of the requirements for the award of degree

Of

Bachelor of Technology

In

Department of Computer Science and Engineering

By

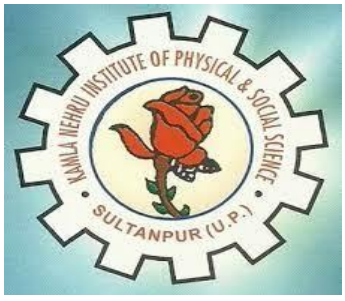
Shubhanshu Mishra

Roll No:2103820100057

Under the supervision of

Dr. Kavita Srivastava (Associate Professor)

(Kamla Nehru Institute of Physical and Social Sciences)



Dr. Abdul Kalam Technical University, Lucknow

April 2025

Abstract

Urbanization has intensified traffic congestion and accident occurrences across cities. The "Real-Time Traffic Congestion & Accident Prediction" system is designed to predict road congestion levels and accident risks using machine learning and real-time data from APIs such as TOMTOM and OpenWeather. With a robust Django-based backend and a React.js frontend, the system provides live traffic visualization, accident risk mapping, and intelligent route suggestions through an interactive dashboard. It leverages RandomForest algorithms for precise forecasting, delivering critical insights to commuters, authorities, and logistics services for proactive traffic management.

Keywords: Real-time Traffic, Accident Prediction, Machine Learning, Django, React.js, TOMTOM API, OpenWeather API, Smart City

1. **Title of the Synopsis :** *"Real-Time Traffic Congestion & Accident Prediction Using Machine Learning"*

2. Introduction

2.1 Background:

Traffic congestion and road accidents disrupt daily life, causing productivity loss, environmental damage, and safety issues. Traditional systems offer little to no predictive insights. This project addresses these limitations through a smart platform that integrates live data, ML models, and interactive maps to forecast and visualize traffic patterns and accident probabilities.

2.2 Importance of Analysis:

Analyzing real-time traffic conditions and combining them with predictive analytics can reduce commute times, optimize route planning, improve emergency response, and enhance overall road safety. Additionally, it enables data-driven decisions for urban development, smart city integration, and climate-conscious transportation strategies.

2.3 Challenges

1. Capturing accurate real-time data from traffic and weather APIs
2. Processing and analyzing vast datasets with low latency
3. Designing scalable machine learning models for live environments
4. Building an intuitive user interface for live interaction and routing
5. Managing system scalability and availability across regions

3. Literature Review

Over the past decade, technological advancements have significantly impacted the transportation sector, yet the integration of predictive analytics and real-time data for traffic and accident management remains limited. Popular applications like **Google Maps** and **Waze** provide **real-time navigation and congestion alerts**, but they primarily operate on crowd-sourced data and lack predictive capabilities driven by machine learning. These platforms also do not integrate weather or historical accident data to determine future risk zones.

Academic studies in the field of traffic prediction generally rely on historical datasets and static simulation models, lacking real-time API integration. Research by institutions including MIT and Stanford has showcased traffic forecasting models using neural networks, but these require high computational power and are rarely implemented in practical, scalable systems.

Enterprise solutions like **IBM Watson IoT for Traffic Management** offer predictive traffic modeling but demand expensive infrastructure and lack public dashboards or open APIs. Moreover, their complexity makes them unsuitable for deployment in small-to-medium cities or educational projects.

Our system bridges these gaps by:

- Using **open and affordable APIs** (TOMTOM and OpenWeather)
- Applying **Random Forest algorithms** for effective and efficient predictions
- Offering a **public-facing, interactive dashboard**
- Deploying the solution in a cloud-native environment (Render), ensuring high availability and ease of access

4. **Problem Statement**

4.1 Need:

The modern transportation network faces mounting pressure from increased vehicle usage, urban sprawl, and climate-sensitive disruptions like floods or fog. These factors have resulted in:

- Increased travel times
- Greater frequency of road accidents
- Reduced emergency response efficiency
- Elevated fuel consumption and emissions

Existing systems are either reactive or informational, rarely offering **forecasting capabilities**. There is a lack of:

- **Intelligent routing** that adapts in real-time
- **Data fusion** from multiple sources (traffic, weather, historical accidents)
- **Machine learning** implementations that are both practical and deployable

A comprehensive solution is needed that delivers **real-time traffic monitoring**, **risk-based predictions**, and **interactive visual interfaces** to serve a variety of stakeholders, from daily commuters to city authorities.

4.2 Gaps in Existing Research

Despite the abundance of data, most available traffic management tools are not predictive. Key research and industrial gaps include:

- Minimal integration of **weather effects on accident probability**
- Lack of **real-time, dynamic visualization** tools
- Poor support for **alert mechanisms or route-based notifications**
- No focus on **user-friendly deployment** for mobile and web platforms
- Absence of **modular, scalable architectures** that allow city-wide or regional expansion

This project not only addresses these gaps but also introduces **institutional deployability**, ensuring educational bodies, municipalities, and smart city initiatives can benefit without needing complex hardware infrastructure.

5. Objectives of the Study

- To **collect real-time data** from traffic and weather APIs (TOMTOM, OpenWeather)
- To **build and train machine learning models** (Random Forest Regressor and Classifier) capable of predicting:
    - Traffic congestion levels
    - Speed patterns
    - Accident-prone zones based on historical and real-time factors
- To develop a **modular Django REST backend** to serve predictions and store logs
- To create an **interactive frontend using React.js and Google Maps API**
- To **deploy the application using Render**, ensuring fault-tolerance and automatic scaling

6. Scope of the Study

Institutional Scope:

- **Government bodies and traffic departments** can use this platform for live monitoring and to implement proactive controls (like signal timings or congestion tax areas).
- **Educational institutions** can use this as a practical model for smart city applications, ML use cases, and cross-disciplinary learning.
- **General commuters and fleet operators** can use the dashboard to find better routes and avoid accident-prone zones.

Student Scope:

- Integration with **mobile applications** using React Native.
- Real-time **voice-based alerts**
- Smart signal control using **IoT sensor feedback loops.**
- Integration with **autonomous vehicle systems** for city-wide AI-based navigation.

Technical Scope:

This system encompasses an end-to-end intelligent traffic solution covering:

- **Frontend**: Built with React.js, incorporating Google Maps for live traffic overlays, interactive dashboards, and route suggestions

- **Backend**: Django and Django REST Framework handle user requests, API consumption, ML prediction routing, and secure access management
- **Machine Learning Models**:
    - RandomForestRegressor: Predicts congestion level based on traffic flow, speed, and time
    - RandomForestClassifier: Predicts accident probability based on geolocation, weather, and time
- **APIs Used**:
    - **TOMTOM**: For live road conditions and congestion updates
    - **OpenWeather**: For weather factors such as fog, rain, temperature, and visibility
- **Database**: MySQL for real-time storage and PostgreSQL for structured querying of geospatial data.

7. Research Methodology

The research methodology adopted for this project follows a **structured, iterative, and data-driven approach**, ensuring the system is built on robust foundations in both software engineering and applied data science. The methodology spans requirement analysis, data handling, model training, integration, deployment, and validation.

7.1 Design

The architecture follows a **modular and scalable full-stack design**, segmented into the following layers:

- **Presentation Layer**: The frontend, developed using React.js, is designed to deliver a responsive, user-friendly dashboard with live traffic visualizations, alerts, and predictive overlays using Google Maps and TOMTOM SDK.

- **Application Layer**: The Django backend exposes REST APIs that process user requests, communicate with external APIs, and invoke machine learning models for prediction.

- **Data Processing Layer**: Data retrieved from APIs is cleaned and transformed using Pandas and NumPy, ensuring structured inputs for model training.

- **Machine Learning Layer**: Trained Random Forest models are loaded via joblib and triggered for real-time predictions based on new incoming data.

- **Deployment Layer**: Hosted on Render with autoscaling and continuous deployment integrated through GitHub actions.

7.2 Data Collection

The project gathers data from multiple, reliable sources:

- **TOMTOM API**: Supplies real-time road congestion, traffic flow, and route speed data.
- **OpenWeather API**: Provides current weather conditions including temperature, humidity, visibility, precipitation, and wind patterns.
- **Historical Data Sets**: Accident logs, traffic reports, and road safety data obtained from public datasets (e.g., Kaggle, Indian Government Open Data Portal).

7.3 Data Processing

The raw datasets undergo multiple transformations:

- **Data Cleaning**: Handling missing values, duplicate removal, standardization of timestamp formats.
- **Feature Engineering**: Derived features such as day of the week, peak hour index, congestion gradient, and visibility range are added.
- **Normalization**: Continuous variables like speed, temperature, and precipitation are normalized to ensure uniform model performance.

7.4 Deployment

To ensure continuous availability and real-time responsiveness, the system is deployed using:

- **Render**: Fully managed cloud platform used for both backend (Django) and frontend (React) hosting with autoscaling and high availability.
- **Google Maps Platform**: For rendering traffic overlays, routes, and hotspot visuals.
- **GitHub Actions**: CI/CD pipelines automatically build and deploy the project on code commits.
- **Firebase / Browser Push API**: For sending real-time traffic and accident alerts (future enhancement).
- **Docker (Optional)**: Containerizes backend for modular deployment.

8. Expected Outcomes

The implementation of this project is expected to deliver both **practical and strategic outcomes** that enhance user experience, enable data-driven decision-making, and support smart city initiatives.

8.1 Live & Predictive Traffic Visualization Platform

The web-based platform will allow users to:

- View **real-time congestion levels** on a map interface
- Identify **accident-prone zones** using dynamic heatmaps
- Receive **route-based predictions** based on time, location, and traffic patterns

The platform will empower users to make smarter travel decisions, ultimately reducing commute times and stress levels.

8.2 Real-Time Alerts and Smart Routing

Users will receive **push notifications and route recommendations** based on predicted congestion and weather-related delays. This includes:

- Alternative low-traffic routes
- Time-based travel suggestions
- Weather-induced hazard alerts (e.g., fog, rain)

Such insights improve safety, comfort, and fuel efficiency.

8.3 Public and Institutional Utility

- **For Traffic Authorities**: Predictive dashboards for congestion hotspots, optimized traffic signal management, and accident forecasting.
- **For Emergency Services**: Fastest route recommendations for ambulances and fire trucks.
- **For Logistics & Fleet Management**: Reduced delays and improved ETA (estimated time of arrival) calculations for delivery services.

These applications contribute directly to more responsive urban governance and enhanced emergency services.

8.4 Academic & Research Benefits

- Serves as a **case study** for machine learning integration in urban infrastructure
- Demonstrates **real-world application of REST APIs, ML models, and frontend frameworks**
- Encourages interdisciplinary exploration (CSE, Civil Engineering, Urban Planning)

8.5 Environmental and Economic Impact

- **Reduced fuel consumption** by optimizing routes
- **Lower carbon emissions** due to less idling and better flow
- **Time saved per trip**, reducing productivity losses caused by traffic jams

9. Work Plan

Week	Tasks	Description	Expected Outcome
Week 1	Requirement Gathering	Define APIs, ML algorithms, data sources	Project Roadmap
Week 2	UI/UX Design	Create wireframes and map interfaces	Visual blueprints
Week 3	Backend Setup	Configure Django API and database	Operational backend
Week 4	ML Model Training	Train congestion and accident predictors	Tested ML models
Week 5	API Integration	Connect APIs (TOMTOM, OpenWeather)	Live data flow
Week 6	Frontend Development	Build dashboard, map, notifications	Working UI
Week 7	Testing	Perform functional and performance testing	Optimized system
Week 8	Deployment	Final deployment and documentation	Production-ready platform

10. References

1. **TOMTOM API – https://developer.tomtom.com**
2. **OpenWeather API – https://openweathermap.org/api**
3. **Django REST Framework – https://www.django-rest-framework.org**
4. **Scikit-learn – https://scikit-learn.org**
5. **Pandas – https://pandas.pydata.org**
6. **Render Deployment – https://render.com**
7. **React.js Docs – https://reactjs.org**
8. **Google Maps API – https://developers.google.com/maps**
9. **NumPy – https://numpy.org**
10. **IEEE Smart City Research – https://ieeexplore.ieee.org**