# **Project Report Format**

## **1. INTRODUCTION**

### **1.1 Project Overview**

* **Project Name:** TrafficTelligence
* **Description:** A Jupyter-based traffic analytics system leveraging computer vision and data visualization to detect, analyze, and report vehicle traffic patterns from video feeds.
* **Scope:** Vehicle detection, counting, classification and traffic stats generation using video data sources.

### **1.2 Purpose**

* To equip traffic management authorities and urban planners with automatic analysis of vehicle flows.
* To reduce manual effort and improve decision-making through visualized insights and real-time monitoring.

## **2. IDEATION PHASE**

### **2.1 Problem Statement**

Manual traffic analytics are time-consuming and error-prone. A scalable automated system is needed to analyze vehicle volumes, classify traffic types, and generate metrics efficiently.

### **2.2 Empathy Map Canvas**

| **WHO** | **NEEDS** | **FEELINGS** |
| --- | --- | --- |
| Traffic analysts | accurate, fast vehicle counts and trends | overwhelmed by manual processing |
| City planners | summary metrics & visual reports | lacking timely data for decisions |

### **2.3 Brainstorming**

* Explore computer vision (OpenCV, deep learning).
* Investigate object detection models (e.g., YOLO, SSD).
* Visualize traffic flow across different time windows.
* Provide analytics dashboard or video output overlay.

## **3. REQUIREMENT ANALYSIS**

### **3.1 Customer Journey Map**

1. User provides video input
2. System runs detection per frame
3. Counts and classifies vehicles
4. Generates time-based charts and counts
5. User views results in notebook or exports reports

### **3.2 Solution Requirements**

* Input: video file or camera stream
* Detect vehicle objects per frame
* Count and classify by type
* Aggregate data per minute/hour
* Output charts and summary tables
* Show overlays on video frames

### **3.3 Data Flow Diagram**

css

CopyEdit

[Video Input] → [Frame Extraction] → [Object Detection Model] → [Count & Classify] → [Aggregate Metrics] → [Visualization & Output]

### **3.4 Technology Stack**

* Language: Python (in Jupyter Notebook)
* Libraries: OpenCV, TensorFlow/Keras or PyTorch, matplotlib, seaborn
* Output formats: Inline display, saved screenshots, exported CSV/plots

## **4. PROJECT DESIGN**

### **4.1 Problem–Solution Fit**

Detecting and counting vehicles manually is slow and inconsistent. Using object detection delivers consistency and automation for traffic monitoring purposes.

### **4.2 Proposed Solution**

A Jupyter notebook that:

* Loads video
* Applies pre-trained object detection (e.g., YOLO or TensorFlow model)
* Outputs per-frame counts (cars, trucks, bikes)
* Aggregates metrics
* Visualizes results with plots and overlays

### **4.3 Solution Architecture**

1. **Input Module** – video reader with OpenCV
2. **Detection Module** – pre-trained model interface
3. **Processing Module** – per-frame counting
4. **Aggregation Module** – time-window metrics
5. **Visualization Module** – graphs & frame overlays
6. **Export Module** – save plots and CSV

## **5. PROJECT PLANNING & SCHEDULING**

### **5.1 Project Planning**

| **Task** | **Duration** | **Stats** |
| --- | --- | --- |
| Research & Ideation | 2 days | ✅ |
| Set up Notebook & Code Structure | 1 day | ✅ |
| Develop Detection Pipeline | 3 days | ✅ |
| Build Aggregation Logic | 2 days | ✅ |
| Visualizations & Output | 2 days | ✅ |
| Testing & Validation | 1 day | ✅ |
| Documentation | 1 day | ✅ |

## **6. FUNCTIONAL AND PERFORMANCE TESTING**

### **6.1 Performance Testing**

* Tested runtime on sample video (~10‑minute clip): ~30 seconds per minute of video.
* CPU load peaked at ~80%.
* Accuracy measured by manual sampling: vehicle count precision ~ 90%.

## **7. RESULTS**

### **7.1 Output Screenshots**

*(Include inline screenshots from your notebook)*

* Frame with bounding boxes and counts overlaid
* Time-series plot of vehicle counts per minute
* Summary chart of vehicle type distribution

## **8. ADVANTAGES & DISADVANTAGES**

**Advantages**

* Automated and reproducible counts
* Easy to extend with other object types
* Visual output aides user insights

**Disadvantages**

* Performance depends on model quality
* Playback speed constraints for long videos
* Lighting and occlusion may reduce detection accuracy

## **9. CONCLUSION**

TrafficTelligence successfully automates traffic analytics using computer vision, offering accurate insights without manual processing. It demonstrates strong potential for deployment in traffic monitoring solutions.

## **10. FUTURE SCOPE**

* Real-time streaming camera support
* Deeper vehicle classification (e.g., insurances vs private)
* Heatmap of traffic density spatially
* Web dashboard integration
* Summarize and export reports as PDFs

## **11. APPENDIX**

### **Source Code**

* TrafficTelligence.ipynb – main pipeline
* Traffictelligence\_videdemo.mp4 – demo video output

### **Dataset Link**

* Video samples included in repository demo folder

### 

### 

### **GitHub & Project Demo Link**

* Repository:<https://github.com/VarshiniBonthu/Traffictelligence>
* Demo Video: Traffictelligence\_videdemo.mp4