**Smart Trafficking enforcement:**

**An Automated Fine generation system**

**Overview**

* **Project Title:** n Full Stack AI-driven traffic monitoring system that detects multiple objects, vehicle speed, helmet usage, rider count, number plate, and opposite lane violations to enhance road safety and enforcement and generates fine accordingly.
* **Team Name:** Detox
* **Institution:** Kongu Engineering College
* **Team Members:**
  + Akshay J – *Computer Vision Developer, Deployment*
  + Krishnakumar N – *Frontend Developer*
  + Dinesh K - *Research and Development, Dataset Cleaning and Preparation*
  + Kavin Pradheep S T - *Solution Architect, UI/UX Designer*
  + Dhivya Prakash S S -*Backend Developer*
* **Date:** 24-09-2024

**2. Problem Statement**

* **Business Challenge:** Many traffic management systems lack effective, automated tools to monitor helmet usage, count riders, recognize license plates, detect opposite lane violations, and measure vehicle speed, leading to insufficient enforcement and road safety challenges.
* **Target Audience:** Traffic authorities, law enforcement agencies, and transportation departments aiming to enhance regulatory enforcement, improve road safety, and streamline traffic management using advanced deep learning and computer vision technologies.

**3. Background**

* **Context:** The transportation and traffic management industry are increasingly focused on improving road safety and compliance. Despite advancements, enforcing regulations related to helmet usage and vehicle identification remains a significant challenge due to manual monitoring and the limitations of current technologies.
* **Existing Solutions:** Current traffic management systems rely heavily on manual inspections and basic camera systems for helmet usage, rider counting, and license plate recognition. These systems often face issues such as low accuracy in varied conditions, slow processing times, and difficulty in integrating data for real-time decision-making.
* **Why This Product:** This product leverages advanced deep learning and computer vision technologies to automate the detection of helmet usage, count the number of riders, and recognize license plates. By addressing the limitations of existing solutions, it provides a more reliable and efficient method for monitoring and enforcing road safety regulations, ultimately improving compliance and safety outcomes.

**4. Product Concept**

* **Vision:** To enhance traffic safety and regulatory compliance by providing an advanced, automated system for detecting helmet usage, counting riders, and recognizing license plates using cutting-edge deep learning and computer vision technologies.
* **Key Features:**

1. **Automated Helmet Detection:** Accurately identifies whether riders are wearing helmets by detecting the presence of a helmet in the image. YOLOv5 is fast, accurate, and well-suited for real-time object detection tasks.
2. **Rider Counting:** Efficiently counts the number of riders on motorcycles by detecting multiple objects (riders) in a single frame. YOLOv5 can be used here due to its real-time detection capabilities, or Faster R-CNN for more accurate detection in complex images.
3. **License Plate Recognition:** Captures and recognizes license plates for vehicle identification and tracking. YOLO is used to detect the license plate's location, and then OpenALPR or Tesseract OCR is applied to extract and recognize the plate characters.
4. **Speed Detection:** Measure vehicle speed to detect over speeding violations.
5. **Opposite Lane Penalty Detection:** Detect vehicles violating lane regulations.
6. **High-Resolution Image Analysis:** Utilizes high-resolution imaging for improved accuracy in diverse lighting and environmental conditions.
7. **Scalable Integration:** Designed to integrate seamlessly with existing traffic management and enforcement systems.

* **Unique Value:** The product stands out by combining advanced deep learning algorithms with sophisticated computer vision techniques to offer precise and automated monitoring of helmet usage, rider counting, and license plate recognition. Unlike existing solutions, it provides a comprehensive and accurate approach to enhancing traffic safety and regulatory compliance, without relying on real-time data.

**5. Business Value**

* **Benefits:**

1. Enhanced road safety through better helmet enforcement and rider counting.
2. Improved regulatory compliance with accurate license plate recognition.
3. Reduced operational costs with automated monitoring.

* **Market Potential:** Significant growth in the global traffic management and public safety technology market.
* **Revenue Model:**

1. Licensing and subscription fees.
2. Custom integration and consultation services.

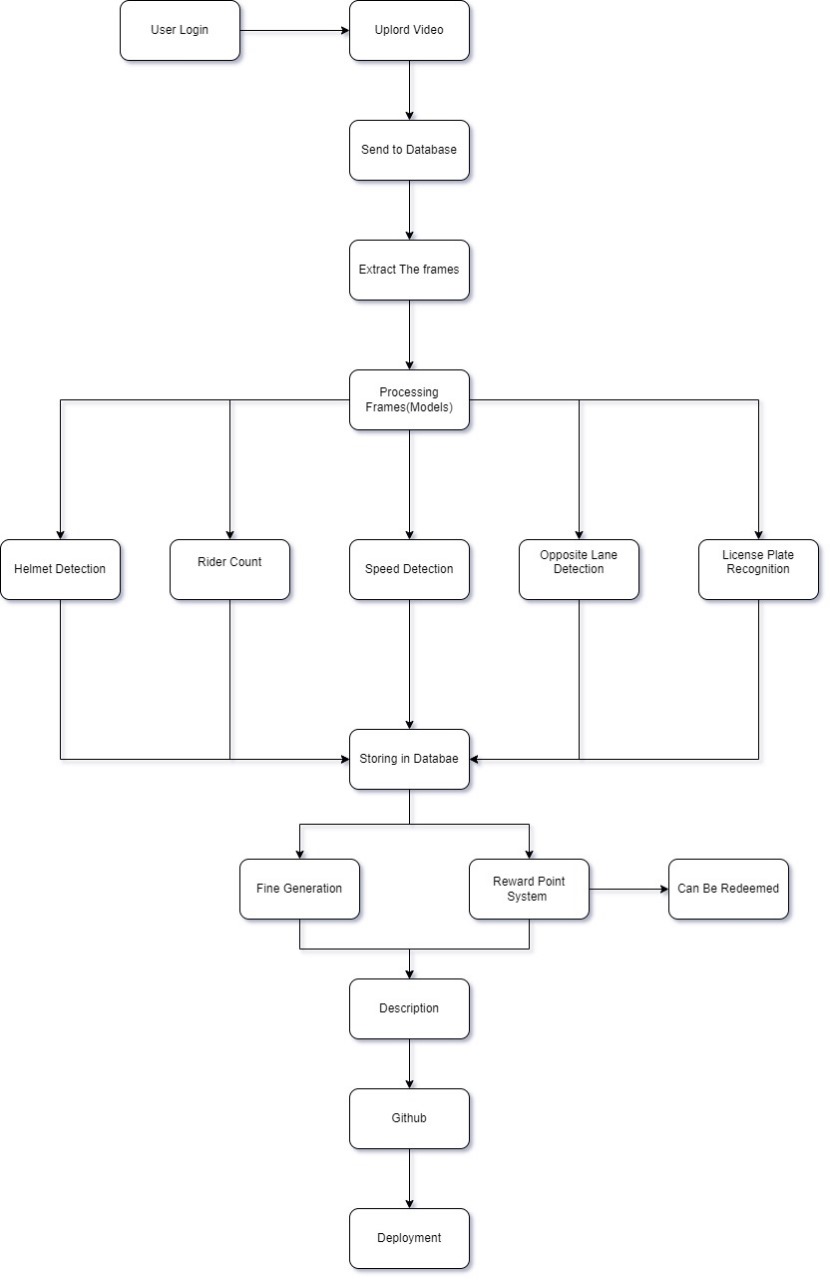
**6. Cost Overview**

* **Development Costs:** The project leverages open-source technologies like React, Flask, YOLO, OpenCV, and Tesseract OCR, keeping technology expenses low. With an internal development team handling both frontend and backend, as well as computer vision tasks, labour costs are minimized.
* **Operational Costs:** Ongoing costs include server hosting and occasional updates. Cloud hosting services like AWS or DigitalOcean will be used for API requests, keeping monthly costs affordable.
* **Cost-Benefit Analysis:** The initial development and operational costs are relatively low due to open-source tools and internal resources. This cost is outweighed by the potential applications in traffic management, safety enforcement, and insurance fraud prevention, each generating substantial business value. As the solution can serve multiple clients, the investment is likely to yield significant returns.

**7. User Interface (UI)**

* **Design Summary:** The UI is designed for simplicity and functionality, allowing users to upload video inputs, view real-time results, and download the processed video. The goal is to provide an intuitive, user-friendly interface that clearly displays detection results such as helmet usage, rider count, and license plate recognition, all with bounding boxes overlaid on the video. Users can interact with the system through minimal steps and receive clear feedback.
* **Key Screens:**
  + **Input Screen (Upload Video)**:
    - Users can upload a video file through this screen. There is a prominent "Upload Video" button that allows the user to select a file from their system. After uploading, users can hit "Process Video" to send the video to the backend for analysis.
    - **Key Features**: File upload functionality, clear instructions, "Process Video" button for initiating analysis.
  + **Processing Screen**:
    - While the backend processes the video, a progress indicator is shown. This screen displays a loading bar or spinner to inform users that the video is being analysed.
    - **Key Features**: Processing status, estimated time remaining, or visual progress bar.
  + **Results Screen (Processed Video Display)**:
    - After the video is processed, it returns to the user with bounding boxes indicating helmet detection, rider count, and the recognized license plate. The processed video is displayed alongside the detailed results.
    - **Key Features**:
      * Processed video playback with overlaid bounding boxes around detected helmets, riders, and license plates.
      * Display of detection results (e.g., "Helmet Detected", "Rider Count: 2", "License Plate: ABC-1234").
      * Option to download the processed video or review the details of the detection.

**User Flow:**



**8. Technical Overview**

* **Components:**
  + **Frontend Interface**: The user-facing web interface where users can upload video files, initiate processing, and view results. It includes the video display with bounding boxes for helmet detection, rider counting, and license plate recognition.
  + **Backend API**: A Flask-based API responsible for receiving video inputs, processing them using deep learning models, and returning the processed video with detection results.
  + **Deep Learning Models**: The core AI engine performing helmet detection (YOLO), rider counting (OpenCV or Faster RCNN), and license plate recognition (OCR using Tesseract).
  + **Video Processing Module**: Handles video input frames, processes each frame, and returns annotated frames with detection results.
  + **Storage/Database**: Stores processed video files and historical detection data for analysis or future reference.
* **Technologies:**
  + **React**: Used for building the frontend user interface, allowing users to upload video files and view real-time results.
  + **Flask**: The backend framework for handling API requests, video processing, and communication between the frontend and AI models.
  + **YOLO**: Employed for real-time helmet detection in the video frames.
  + **Roboflow**: Roboflow is a platform that simplifies building, training, and deploying computer vision models. It provides tools for dataset management, annotation, and data augmentation, supporting formats like YOLO and COCO.
  + **OpenCV/Faster RCNN**: Used for counting the number of riders on a bike.
  + **Tesseract OCR**: Optical Character Recognition used for extracting license plate numbers from video frames.
  + **AWS/DigitalOcean**: For cloud hosting and storage, enabling efficient video processing and serving results to the frontend.
* **Integration:** The frontend React interface interacts with the Flask backend through API calls. The user uploads the video via the UI, which is sent to the backend for processing. The Flask backend utilizes deep learning models (YOLO, OpenCV, Tesseract) to detect helmets, count riders, and recognize license plates. Once processed, the video with bounding boxes and results is sent back to the frontend for display. All data is handled and processed in real-time, ensuring a seamless interaction between the components.

**9. Conclusion**

* **Summary:** This full-stack solution automates the detection of helmet usage, rider counting, speed detection, and license plate recognition using deep learning. The system is intuitive and designed for traffic enforcement and road safety, integrating easily with existing infrastructure.
* **Next Steps:**
  + Conducting performance testing to evaluate the detection models' accuracy and processing speed.
  + Optimizing the user interface and backend to handle larger video files and improve response time.
  + Finalizing the integration between the frontend and backend to ensure seamless real-time interactions.
  + Deploying the system on a cloud platform like AWS or DigitalOcean for wider access and testing scalability.