**Advanced License Plate Recognition System: Requirements Document**

Büşra Yıldırım – 1031510030

Havva Nur Aslan – 1031510003

**1. Project Overview**

**Project Name:** Advanced License Plate Recognition System

**Objective:** The primary objective of this project is to develop an AI-powered system capable of automatically detecting vehicles and license plates within images and videos. This system aims to provide foundational solutions for applications such as traffic monitoring and parking management by offering accurate object identification and counting through a user-friendly web interface.

**2. Technology Stack**

The system leverages a focused set of modern technologies to ensure efficient performance and a responsive user experience:

* **Core Framework:** Python 3.8+
* **Object Detection:** YOLOv8x/PyTorch (utilizing a custom-trained model for optimal vehicle and license plate detection)
* **Image Processing:** OpenCV
* **Web Framework:** Flask (for building the web interface)
* **Frontend:** HTML5, CSS3 (for delivering a dynamic user experience)
* **Deployment:** Google Colab (for model training and experimentation)

**3. Functional Requirements**

This section details the core functionalities the system must deliver.

**3.1 Media Input & Management**

* **Image Upload:** The system shall allow users to upload image files in **JPG** and **PNG** formats.
  + **FR-3.1.1:** The maximum allowable file size for image uploads shall be **10MB**.
* **Video Upload:** The system shall allow users to upload video files in **MP4** format.
  + **FR-3.1.2:** The maximum allowable file size for video uploads shall be **100MB**.

**3.2 Detection Capabilities**

* **Vehicle Identification:** The system shall accurately identify and categorize various vehicle types, such as **cars**, **trucks**, and **motorcycles**, within uploaded media.
* **License Plate Detection:** The system shall detect and provide bounding box coordinates for **license plates** within the uploaded media.
* **Object Counting:** The system shall provide a count of detected objects per category (e.g., number of cars, number of license plates) based on the analysis.

**3.3 Processing Workflow**

* **Analysis Trigger:** A dedicated "**Detect**" button shall initiate the analysis process upon user request.
* **Progress Indication:** The system shall display a clear progress indicator to inform the user about the ongoing processing status.
* **Results Overlay:** The system shall overlay the detection results (bounding boxes and labels) directly onto the original uploaded image or video frame.

**4. Non-Functional Requirements**

This section outlines the quality attributes and constraints that the system must adhere to.

**4.1 Performance**

* **Processing Speed:** The system shall process images within a reasonable timeframe (e.g., under 5 seconds for a typical 5MB image). Video processing time will depend on video length and resolution.

**4.2 Usability**

* **Intuitive User Interface:** The web interface shall be user-friendly, easy to navigate, and visually appealing for straightforward media uploads and result viewing.
* **Error Handling:** The system shall provide clear and informative error messages to the user in case of issues like invalid file formats or oversized uploads.

**4.3 Scalability**

* **Model Training:** The model training process is conducted in Google Colab, allowing for scalable compute resources during development.
* **Web Application:** The Flask web application is designed for simplicity, suitable for single-user or low-concurrent usage.

**4.4 Maintainability**

* **Code Quality:** The codebase shall be well-structured, modular, and adhere to standard Python coding practices.
* **Testability:** Key system components, especially the detection model and image processing logic, shall be designed for easy testing.