**SDD HEADERS**

1. **INTRODUCTION**
   1. **Purpose:**
   2. **Scope:**
   3. **Overview:**
2. **High Level Design Overview:**

**2.1) System Architecture**

**2.2) Database Design**

**2.3) Application Component**

**1. \_\_init\_\_**

* **Purpose:** Initializes the DetectPLate object. Sets up dictionaries for tracking timestamps and positions, a set for logged IDs, initializes the OCR system, and connects to the SQLite database.
* **Key Components:**
  + self.trk\_pt & self.trk\_pp: Track previous timestamps and positions for each object.
  + self.logged\_ids: Tracks already processed IDs to avoid redundancy.
  + self.ocr: Configures the PaddleOCR system for optical character recognition.
  + Database connection established using connect\_to\_db.

**2. connect\_to\_db**

* **Purpose:** Establishes a connection to an SQLite database and ensures the detection\_data table exists.
* **Key Operations:**
  + Connects to number\_plates.db.
  + Creates the detection\_data table if it doesn't exist.
* **Returns:** A database connection object.

**3. perform\_ocr**

* **Purpose:** Extracts text from an image using OCR.
* **Input:** A cropped image in numpy.ndarray format.
* **Output:** Extracted text from the image as a string.
* **Key Validations:**
  + Ensures the input is not None.
  + Validates the input is a NumPy array.
* **OCR Implementation:** Uses PaddleOCR.

**4. save\_to\_database**

* **Purpose:** Saves detection details to the SQLite database.
* **Inputs:**
  + date: Detection date.
  + time: Detection time.
  + track\_id: Unique ID of the tracked object.
  + number\_plate: Extracted text (license plate).
* **Key Operations:**
  + Executes an SQL INSERT query to store the data.
  + Commits the transaction to ensure the data is saved.

**5. perform\_detection**

* **Purpose:** Detects and tracks objects in a video frame, annotates them, extracts license plate text using OCR, and stores the information in the database.
* **Steps:**
  1. Annotator Initialization: Sets up the Annotator for adding bounding boxes and labels.
  2. Track Extraction: Updates tracked objects' data (self.boxes, self.track\_ids, and self.clss).
  3. **OCR Processing:**
     + Crops the image region defined by the bounding box.
     + Extracts text using perform\_ocr.
  4. **Save Data:** If the track\_id is new and OCR text is not empty:
     + Logs details in the SQLite database.
     + Sends data to a remote server via POST.
  5. Frame Display: Annotates the frame with tracking data and OCR results.
* **Returns:** The annotated frame.

**6. Main Script Logic**

1. **Video Capture Initialization:**
   * Reads the video file passed as a command-line argument.
   * Checks if the video file can be opened.
2. **Object Detection and Tracking:**
   * Initializes the DetectPLate object with a specified YOLO model (best\_20.pt).
   * Processes every third frame of the video to optimize performance.
   * Passes each frame to perform\_detection for object tracking, OCR, and annotation.
3. **Display Results:**
   * Shows the processed frames in a window.
   * Allows the user to quit by pressing q.
4. **Cleanup:**
   * Releases the video capture object and closes the display window.

**7. Argument Parsing**

* Purpose: Allows the user to specify the path to the video file as a command-line argument.