CAPSTONE PROJECT - 1

CAMPUS SURVEILLANCE AND ANALYSIS SYSTEM

By

Dennis P James, Dijin Dominic, Anuja Alice Thomas

(2348025, 2348027, 2348013)

Under the Guidance of

Dr. Dalvin Vinod Kumar

A capstone project synopsis submitted in partial fulfillment of the requirements for the award of the degree of Master of Science (Data Science) of CHRIST (Deemed to be University)

May – 2024

System Design

for

Campus Surveillance and Analysis System

Prepared by

Dennis P James, Dijin Dominic, Anuja Alice Thomas

SYSTEM DESIGN

1. SOFTWARE ARCHITECTURE

1.1 Overall Architecture

The overall architecture for the surveillance system will be a client-server model where the client interacts with the system through a web application, and the server side handles video processing, storage, alerts, and system maintenance tasks.

1.2 Tiers of the Application

The different tiers of the application include:

i. Presentation tier: a web application for user interaction along with a dashboard to display counts, alerts, and notifications

ii. Application tier

- o Video Processing Service: Preprocesses video feeds.
- o Object Detection Service: Uses YOLOv8 for detecting objects (students, vehicles).
- o Tracking Service: Uses ByteTrack to track detected objects.
- o Data Processing Service: Processes outputs and saves them to the database.
- o Notification Service: Manages real-time alerts and notifications.

iii. Data Tier: consists of the database that stores processed data, counts, student information, and other details.

2. APPLICATION FLOW

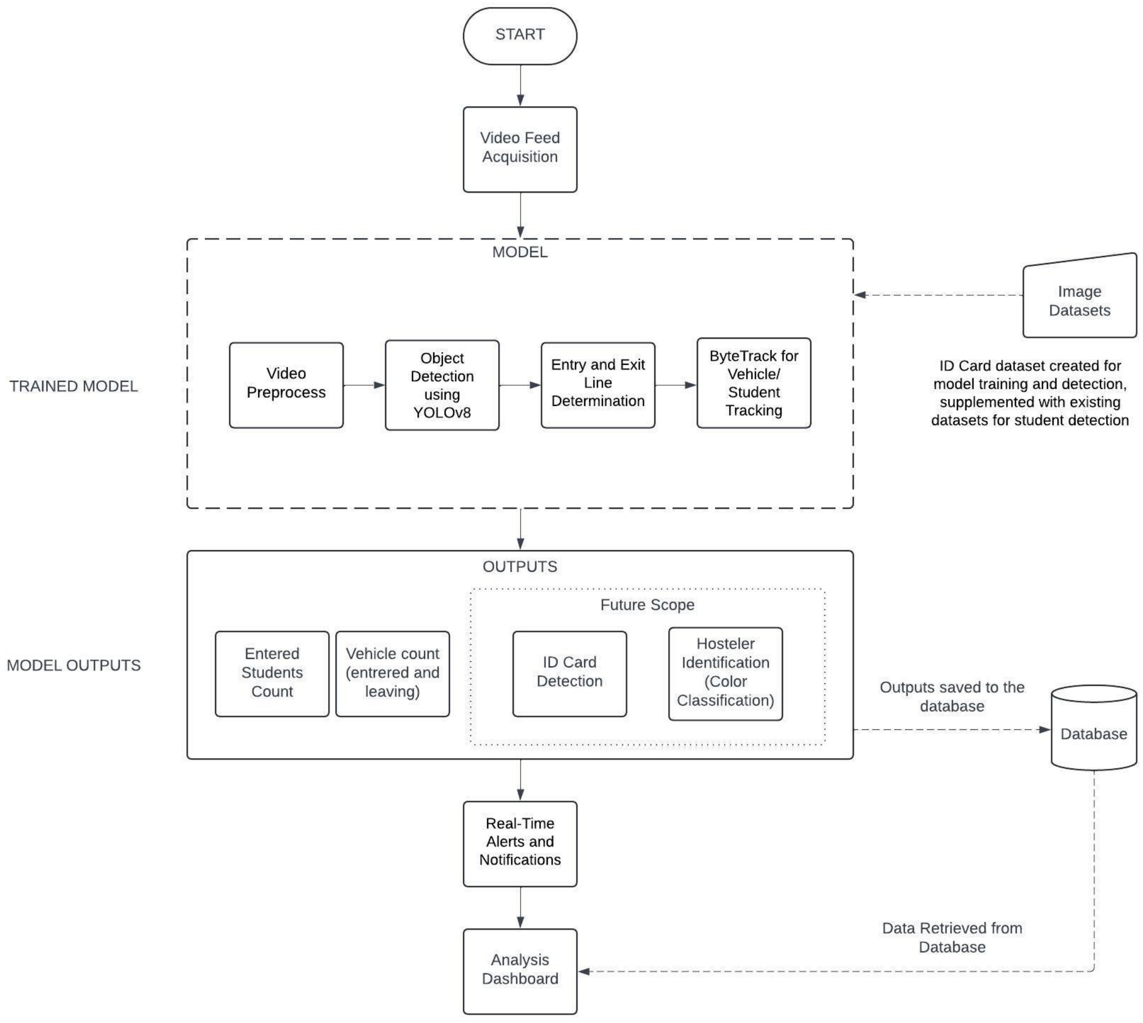
The project consists of three major phases: the video capture and process, the training of the model, and obtaining the outputs and presenting them on the UI.

After the video capture, the training of the model has four major steps:

i. Preprocessing the captured video
ii. Frame-wise analysis for object detection
iii. Entry/ exit line determination
iv. Tracking of the exit and entry counts based on the directions.

The required details are collected and stored in the database as needed.

The following flowchart represents a basic flowchart of the significant processes in the project:


START
Video Feed Acquisition
MODEL
TRAINED MODEL
Video Preprocess
Object Detection using YOLOv8
Entry and Exit Line Determination
ByteTrack for Vehicle/ Student Tracking
Image Datasets
ID Card dataset created for model training and detection, supplemented with existing datasets for student detection
OUTPUTS
MODEL OUTPUTS
Entered Students Count
Vehicle count (entrered and leaving)
Future Scope
ID Card Detection
Hosteler Identification (Color Classification)
Outputs saved to the database
Database
Real-Time Alerts and Notifications
Analysis Dashboard
Data Retrieved from Database

3. MODEL/ ALGORITHMS

3.1 YOLOv8

The project mainly uses the YOLOv8, a cutting-edge deep learning model for real-time object detection in computer vision tasks. Its sophisticated architecture and innovative algorithms have transformed the object detection field, allowing for precise and efficient identification of objects in real-time situations.

3.2 Byte Track

ByteTrack is an algorithm for tracking multiple objects in videos, aiming to improve accuracy and reliability. It stands out in preserving the identity of objects even when they are obstructed and in intricate surroundings. ByteTrack utilizes sophisticated association methods to match identified objects across frames accurately. Created for instantaneous execution, it is appropriate for tasks such as monitoring, self-driving cars, and automation in the industry.

3.3 Working of YOLOv8 and Byte Track

The system starts by **capturing live video feeds** from cameras installed around the parking area. The Video Processing Service, utilizing OpenCV, preprocesses these feeds by **splitting them into individual frames**. Each frame is then processed by the YOLOv8 model, explicitly trained for vehicle detection. The YOLOv8 model analyses each frame and identifies vehicles, outputting **bounding boxes** and labels indicating the presence and positions of the car.

The Supervision library then manages these detection results, integrating YOLOv8 with Byte Track to streamline the detection and tracking pipeline. The Supervision library coordinates the data flow, ensuring that the vehicle detection results from YOLOv8 are efficiently passed on to Byte Track. **Byte Track** then takes over, using the detection data to track the vehicles across consecutive frames. It assigns unique IDs to each detected vehicle and maintains them as the cars move through the frames, ensuring consistent tracking.

Since vehicles enter and leave through the same gate, the system sets up a single boundary line at the gate to monitor vehicle movement. The Data Processing Service analyses the tracked vehicle data to determine whether a vehicle crosses the boundary line in an entry or exit direction. This is achieved by monitoring each vehicle's trajectory and movement patterns as it interacts with the boundary line. Byte Track facilitates this by continuously updating each vehicle's position and movement vector based on its unique ID. By comparing the vehicle's positions across frames, the system can infer the direction of movement.

Based on the direction of movement, the system updates the vehicle counts accordingly, incrementing the count for entries and decrementing it for exits. The updated vehicle counts and associated data are stored in the database.

The Notification Service sends real-time alerts and updates about parking availability to the front end. The front end, a web application built with frameworks like React or Angular, displays a real-time dashboard showing available parking spots and notifications. This comprehensive flow managed and streamlined by the Supervision library, ensures accurate vehicle counting and provides timely information to users about parking availability on campus, even when vehicles enter and leave through the same gate.

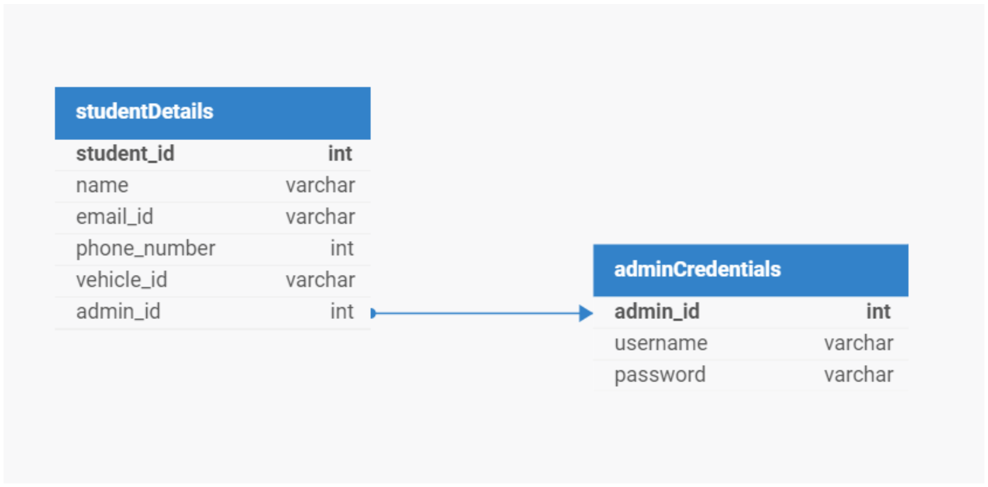
4. DATABASE DESIGN

The requirement for the database is minimal for the project. The following are the required tables within the database:

i. Student Details

ii. Admin Details

The table schemas are as follows:



studentDetails	
student_id	**int**
name	varchar
email_id	varchar
phone_number	int
vehicle_id	varchar
admin_id	int

adminCredentials	
admin_id	**int**
username	varchar
password	varchar

<u>adminCredentials</u>

This table has the username and corresponding passwords confidential only for the management and higher authorities to access the recorded data.

- Primary key: admin_id

<u>studentDetails</u>

It has all the details of students registered for the parking lot alerts.

- Primary Key: student_id
- Foreign Key admin_id refers adminCredentials table

5. INTERFACE DESIGN

5.1 Interface

A basic interface design for your campus surveillance project focusing on parking slot analysis and allocation:

1. Login Screen

- Username (Text field)
- Password (Password field)
- Login Button

2. Dashboard

- Header
  - Project Title (e.g., "Campus Surveillance & Parking Management")
  - Logout Button
- **Navigation Menu (Sidebar):**
  - Home (Can contain the company information and services)
  - Parking Slot Dashboard (Prompt for Login)

3. Parking Slot Dashboard

- **Live Feed Section:**
    - Display live video feed from cameras monitoring the parking area.
    - Overlay with YOLO model detections showing available and occupied slots.
- **Summary Section:**
    - Total Parking Slots: X
    - Available Slots: Y
    - Occupied Slots: Z

5.2 Design Considerations

- Responsive Design: Ensure the interface is accessible on desktops, tablets, and smartphones.
- Real-time Updates: Use WebSockets or similar technology to update real-time slot availability and vehicle detection.
- User-friendly Interface: Keep the design intuitive with clear labels and tooltips for each functionality.

5.3 Technology Stack

- **Frontend**: HTML, CSS, JavaScript (React.js or Vue.js)
- **Backend:** Python (Flask or Django)
- **Database:** PostgreSQL or MySQL
- **Computer Vision:** OpenCV, YOLO