# Task 2 Report

**Title:** Player Re-Identification in a Single Camera Feed

**Intern:** Shankhni Marandi

**Company:** Liat.ai (Stealth Mode Internship)

**Submission Format:** Jupyter Notebook + Report

**Date:** June 24, 2025

## Objective

The objective of Task 2 is to implement a **player re-identification system** that maintains **consistent player IDs** in a single camera video feed. The goal is to:

- Detect players using the provided YOLOv11 model.
- Assign unique IDs to each player in the first few seconds.
- Ensure that players retain the same ID throughout the video, even after temporarily going out of the frame and reappearing.

## Tools and Technologies Used

- Python 3.12 (Jupyter Notebook)
- YOLOv5 (custom-trained yolov11.pt weights)
- Deep SORT Realtime tracker
- OpenCV for frame handling and video output
- Matplotlib and ipywidgets for interactive visualizations
- Torch and torchvision for model handling
- Ultralytics library for loading and running YOLO models

## Methodology

The process began by loading the 15-second input video (15sec_input_720p.mp4) and reading it frame by frame. Each frame was passed through the YOLOv11 model to detect players. Detections were filtered to include only the "player" class. The resulting bounding boxes were passed into the Deep SORT tracker to assign and maintain unique player IDs.
The annotated frames were displayed using a slider-based interactive viewer, enabling visual inspection of tracking consistency frame by frame. The processed frames were also saved into an output video (output_video.mp4) for submission.

**Results and Observations**

The tracking system successfully assigned and maintained unique IDs for players across the video. Even when players went out of frame and reappeared, Deep SORT retained their identities. A slider widget allowed interactive browsing through the frames, while a bar graph showed the number of players detected per frame.
The final output video contained bounding boxes labeled with consistent ID: x tags across all frames. Detection accuracy was high, with occasional flickers only when players overlapped heavily or partially occluded each other.

**Challenges Faced**

The YOLOv11 model provided was not compatible with the latest PyTorch Hub interface, so it had to be loaded using the Ultralytics library instead. SORT algorithm did not function reliably and was replaced with Deep SORT due to its enhanced tracking logic. Inference was relatively slow on CPU but sufficient for this short video.

Installing dependencies like deep_sort_realtime, ipywidgets, and ensuring compatibility with Python 3.12 required some adjustments during setup.

**Improvements and Additions**

The solution includes an interactive visualization interface using ipywidgets, allowing frame-by-frame inspection of player tracking with X and Y axis context. A player count bar chart helps validate the consistency of detection across frames. The tracker settings (like max_age and nn_budget) were tuned for better accuracy on this clip.

**Output Summary**

- Jupyter Notebook: Task2_Player_ReID_SingleFeed.ipynb
- Output Video: output_video.mp4 with bounding boxes and IDs
- Interactive slider to view each frame
- Consistent tracking results using Deep SORT
- Bar chart showing players per frame