Work Report: Player Matching AI

1. Objective

The goal of this project is to develop an interactive, scalable, and modular **web application** to:

- Detect football players from two different video sources (broadcast and tacticam).
- Extract visual features using a CNN model.
- Match detected players across views using cosine similarity and the Hungarian algorithm.
- Visualize detections and player mappings clearly.
- Export results as JSON and optionally video for further analysis or model supervision.

---

2. System Architecture

a. Detection Pipeline

- Uses **YOLOv8** for bounding box detection.
- Filters detections to class "person" with confidence > threshold.
- Extracts player crops and computes features using pretrained **ResNet18**.

b. Feature Extraction

- Uses torchvision.models.resnet18(pretrained=True) for high-dimensional feature vectors.
- Applies image transformations (ToPIL, Resize, ToTensor) before inference.

c. Player Matching

- Computes **cosine similarity** between all player features across both views.
- Uses **Hungarian Algorithm** (scipy.optimize.linear_sum_assignment) to obtain optimal matching.

---

3. UI

Layout:

- Clear headers, descriptions.
- Multiple file upload areas and configuration inputs.

Inputs:

- YOLOv8 model (.pt)
- Broadcast video (.mp4)
- Tacticam video (.mp4)
- Start frame, End frame, and Frame stride for selective processing

Outputs:

- Downloadable player_mapping.json

- Per-frame side-by-side visualization of bounding boxes
- Interactive slider for frame navigation
- One-click button to generate a comparison video (broadcast vs tacticam)

---

4. Key Modules

Module	Purpose
utils/features.py	Extracts CNN feature embeddings for player crops
utils/detection.py	YOLOv8 detection logic with configurable frame sampling
utils/matching.py	Computes cosine similarity & matches players
utils/visualization.py	Draws bounding boxes and generates side-by-side comparison video
config.py	Stores global constants (e.g., detection threshold)

---

6. Output Artifacts

- player_mapping.json — JSON file containing mapping like:

```
{
 "tacticam_player_0": "broadcast_player_2",
 "tacticam_player_1": "broadcast_player_0"
}
```

- comparison_video.mp4 — Side-by-side visualization of detections in both views.
-

---

9. Summary

The **Player Matcher** app, a modular, scalable, and interactive tool that enables:

- Efficient multi-view player identity matching
- Clear UI for non-technical users
- Batch processing for long videos
- Visual validation of model accuracy

It is ready for integration into a larger football analysis pipeline, dataset curation tool, or match review system.