Football Training App - FastAPI Backend Documentation (Phase 1)

Overview

This project implements the backend for the Football Training App. It processes training videos to analyze juggling (ball tracking and kick validation) and detect PSV jerseys using a trained YOLOv8 model. The backend also tracks performance metrics (such as correct kicks, incorrect kicks, and longest streak) and stores session data in Firebase for later retrieval.

Project Flow

The following steps outline the complete flow of the project from video upload to session data retrieval:

1. **Video Upload:**
    - A client (or later, a Flutter app) uploads a training video through the `/analyze_exercise` API endpoint.
2. **Video Processing:**
    - The backend saves the uploaded video temporarily and processes it frame-by-frame.
    - **Ball Tracking:**
      Each frame is passed through the `detect_ball()` function to detect the ball using color segmentation (HSV filtering) and validate if the ball reaches the required knee height.
    - **Pose Detection:**
      The `detect_foot_kick()` function (using MediaPipe) checks that only the feet are used for kicking.
    - **Performance Metrics:**
      The `StreakTracker` class is used to count correct and incorrect kicks, and to maintain the longest streak of valid kicks.
    - **Error Logging:**
      The `ErrorHandler` class logs errors (e.g., when no ball is detected or when non-foot movements occur).
    - **Jersey Detection:**
      Each frame is also processed by the YOLOv8 model (via the `detect_jersey()` function) to detect the presence of a PSV jersey. Confidence scores from each frame are collected and averaged.
3. **Session Data Compilation & Storage:**
    - After processing all frames, the backend compiles session data, including:
        - Correct and incorrect kicks
        - Longest valid kick streak

    - Errors encountered
    - Averaged jersey detection confidence (categorized as High, Average, Low, or Not Detected)
  - This session data is stored in Firebase Firestore using the `save_session()` function.
4. **Response & Data Retrieval:**
   - The `/analyze_exercise` endpoint returns the session data as a JSON response.
   - The `/session_results/{user_id}` endpoint allows retrieval of past session data from Firebase for a given user.
5. **Future Integration:**
   - This backend will later integrate with a Flutter frontend that will allow users to upload videos and view their training results in a user-friendly interface.

Current File Structure

```
football_project_phase1_api/
├── dataset/                                 # Dataset and model-related files
│   ├── images/                              # Contains training/validation
images
│   ├── labels/                              # YOLO annotation files for images
│   ├── psv_dataset.yaml                     # Dataset configuration for YOLO
training
│   ├── runs/                                # YOLO training outputs (logs,
results, weights, etc.)
│   └── yolov8n.pt                           # Pre-trained YOLOv8 model file
(used for inference)
├── football-b34a3-firebase-adminsdk-fbsvc-fa34f1117a.json  # Firebase service
account credentials
├── main.py                                  # FastAPI main entry point
├── modules/                                 # Modular backend components
│   ├── ball_tracking.py                     # Detects the ball & validates
kicks
│   ├── pose_tracking.py                     # Ensures that only feet are used
for kicking
│   ├── jersey_detection.py                  # Uses YOLOv8 for PSV jersey
detection
│   ├── streak_tracker.py                    # Tracks correct/incorrect kicks
and longest streak
│   ├── error_handler.py                     # Detects and logs errors during
kick detection
│   └── firebase_db.py                       # Stores/retrieves session data
from Firebase
├── your_own_firebase_credentials.json       # (Optional) Additional Firebase
credentials file
├── requirements.txt                         # Python dependencies for the
project
├── venv/                                    # Virtual environment (not tracked
by Git)
└── __pycache__/                             # Cached Python bytecode
(automatically generated)
```

Module Details

1. ball_tracking.py

- **Functions:**
  - `detect_ball(frame, frame_height)`: Uses HSV filtering to detect the ball in the frame and returns a kick status ("Valid Kick ", "Invalid Kick ", or "No Kick Detected") based on whether the ball reaches knee height.
  - `process_video(file_bytes)`: Processes an entire video file by reading frames, applying `detect_ball()`, and returning a list of kick statuses.

2. pose_tracking.py

- **Function:**
  - `detect_foot_kick(frame)`: Utilizes MediaPipe to process pose landmarks and determines whether the player is using only their feet to kick. Returns "Valid Foot Kick " or "Invalid Kick (Non-Foot Movement)".

3. jersey_detection.py

- **Function:**
  - `detect_jersey(image_bytes)`: Loads the YOLOv8 model (`best.pt`), processes an image frame, and returns a JSON with a jersey detection status and confidence score.

4. streak_tracker.py

- **Class:**
  - `StreakTracker`: Maintains counters for correct kicks, incorrect kicks, and the longest streak. Methods update these counters based on each frame’s kick validation.

5. error_handler.py

- **Class:**
  - `ErrorHandler`: Logs errors based on the output from ball tracking and pose tracking. Provides a method to return a list of detected errors.

6. firebase_db.py

- **Functions:**
  - `save_session(user_id, session_data)`: Saves session data (including kick metrics, errors, and jersey detection confidence) to Firebase Firestore.
  - `get_session(user_id)`: Retrieves stored session data for a given user from Firebase Firestore.

FastAPI Main Application (main.py)

- **Endpoints:**

    - `POST /analyze_exercise`: Accepts a training video (multipart/form-data) along with a user ID, processes the video frame-by-frame (applying ball tracking, pose tracking, and jersey detection), updates performance metrics, stores session data in Firebase, and returns the session results as JSON.
    - `GET /session_results/{user_id}`: Retrieves stored session data for the specified user from Firebase.
- **Integration Details in main.py:**
    - Video frames are processed in a loop where:
        - Ball detection and pose detection determine the kick validity.
        - The streak tracker and error handler are updated accordingly.
        - Each frame is encoded and passed to the YOLO-based jersey detection, with confidence scores collected for averaging.
    - After processing, an average jersey confidence is computed and categorized (High, Average, Low, Not Detected).
    - The session data (including a Firebase timestamp) is saved and returned as a JSON response.

Dependencies (requirements.txt)

A typical `requirements.txt` file might include:

```
fastapi
uvicorn
opencv-python
numpy
mediapipe
ultralytics
torch
firebase-admin
```

Conclusion

This documentation covers:

- **The overall project flow** from video upload to session data storage.
- **Detailed descriptions** of each module and their roles.
- **File structure** and how each component contributes to the backend functionality.
- **How the FastAPI endpoints interact** with the modules to perform juggling analysis and PSV jersey detection.

This backend is fully set up for processing training videos, detecting key metrics, and storing results in Firebase, and is ready for future integration with a Flutter frontend.