Football Training App – Flutter & FastAPI Integration Guide

Overview

This guide describes how the FastAPI backend for the Football Training App will be integrated with a Flutter frontend. The backend processes training videos to analyze juggling (ball tracking and kick validation) and detect PSV jerseys using a trained YOLOv8 model. It also tracks performance metrics (correct/incorrect kicks, longest streak, errors) and stores session data in Firebase. The Flutter app will serve as the user interface for video uploads and displaying results.

Project Flow

1. **Video Upload:**
    - The user (via the Flutter app) selects and uploads a training video through the app.
    - The video file is sent to the FastAPI backend using a POST request to the `/analyze_exercise` endpoint along with a user ID.
2. **Video Processing (Backend):**
    - **Temporary Storage:** The FastAPI backend temporarily saves the uploaded video.
    - **Frame-by-Frame Analysis:**
        - **Ball Tracking:** Each frame is processed by `detect_ball()` (in `modules/ball_tracking.py`) to detect the ball using HSV filtering and determine if the ball reaches the required knee height (valid kick).
        - **Pose Detection:** `detect_foot_kick()` (in `modules/pose_tracking.py`) ensures that only feet are used for kicking.
        - **Streak & Error Tracking:** The `StreakTracker` (in `modules/streak_tracker.py`) updates counts for correct and incorrect kicks and tracks the longest valid streak. The `ErrorHandler` (in `modules/error_handler.py`) logs errors such as missing ball or non-foot movement.
        - **Jersey Detection:** Each frame is also encoded and passed to `detect_jersey()` (in `modules/jersey_detection.py`), which uses the YOLOv8 model (loaded from `best.pt`) to detect a PSV jersey. Confidence scores from each frame are collected and averaged.
    - **Session Data Compilation:** After processing all frames, the backend compiles:
        - Kick metrics (correct, incorrect, longest streak)
        - Logged errors
        - Averaged jersey detection confidence (categorized as High, Average, Low, or Not Detected)

- **Firebase Storage:** The compiled session data is saved to Firebase Firestore using functions from `firebase_db.py`.

3. **Response & Data Retrieval:**

    - The backend returns the session data as a JSON response.
    - The Flutter app can also retrieve past session data using the `/session_results/{user_id}` endpoint.

4. **Flutter Frontend Interaction (Future Integration):**

    - The Flutter app uses packages like `http` and `file_picker` to interact with the backend.
    - It will allow users to upload videos and then display the analyzed results (kick statistics, jersey detection confidence, error logs) in an intuitive UI.
    - The integration will use the backend endpoints to fetch and display real-time training feedback.

Current Backend 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)
├── your-firebase-credentials.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
│   ├── 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   #Additional Firebase credentials file
├── requirements.txt                     # Python dependencies for the project
├── venv/                              # Virtual environment (not tracked by Git)
└── __pycache__/             # Cached Python bytecode (automatically generated)
```

Key Backend Components

1. ball_tracking.py

- **Function:** `detect_ball(frame, frame_height)`
  Uses HSV color filtering to detect the ball and checks if it meets the height requirement to count as a valid kick.

2. pose_tracking.py

- **Function:** `detect_foot_kick(frame)`
  Utilizes MediaPipe to ensure that only the feet are involved in kicking, flagging non-foot movements.

3. jersey_detection.py

- **Function:** `detect_jersey(image_bytes)`
  Loads the YOLOv8 model (`best.pt`) and processes each frame to detect a PSV jersey, returning a confidence score.

4. streak_tracker.py

- **Class:** `StreakTracker`
  Maintains counters for correct kicks, incorrect kicks, and tracks the longest streak of valid kicks.

5. error_handler.py

- **Class:** `ErrorHandler`
  Logs errors (e.g., no ball detected, incorrect body part usage) based on the analysis from ball and pose tracking.

6. firebase_db.py

- **Functions:** `save_session(user_id, session_data)` and `get_session(user_id)`
  Interact with Firebase Firestore to store and retrieve session data.

7. main.py

- **Endpoints:**
  - `POST /analyze_exercise`: Processes the uploaded video frame-by-frame (applying ball tracking, pose detection, and jersey detection), updates performance metrics, saves the session data to Firebase, and returns the results as JSON.
  - `GET /session_results/{user_id}`: Retrieves stored session data for the specified user.
- **Integration:**
  In the frame processing loop, after ball and pose detection, each frame is passed through `detect_jersey()`, and the jersey confidence values are averaged to give an overall detection result.

Flutter Integration (Future Steps)

- **HTTP Communication:**
  The Flutter app will use the `http` package to send video files to the backend and retrieve JSON responses.

- **File Upload:**
  The app will employ the `file_picker` package to let users select and upload training videos.
- **Displaying Results:**
  After receiving the analysis from the backend, the Flutter UI will display the number of correct kicks, incorrect kicks, longest streak, any detected errors, and the jersey detection confidence.
- **Session Management:**
  Users will be able to view their historical training sessions retrieved from Firebase via the backend.

Conclusion

This integration guide explains how the FastAPI backend and Flutter frontend will work together:

- **Backend Processing:** Analyzes training videos to evaluate juggling performance and detect PSV jerseys, tracks performance metrics, logs errors, and stores session data in Firebase.
- **Frontend Integration:** A Flutter app will upload training videos, receive analysis results from the FastAPI backend, and display them in a user-friendly interface.