AI Virtual Coach Mobile Application

Product Requirements Document (PRD)

---

1. Overview

1.1 Product Vision

The AI Virtual Coach application is a research-driven mobile application that enables users to perform a single exercise session in front of their smartphone camera and receive automated, AI-based feedback on exercise form and performance. The system leverages pose-based deep learning models for exercise recognition and form assessment, providing reflective post-exercise feedback without storing raw video data.

The application is designed as a **proof-of-concept and research demo** aligned with a bachelor thesis and academic publication, while remaining architecturally extensible toward future full on-device inference and real-world deployment.

1.2 Goals

- Demonstrate an end-to-end working AI-powered mobile application
- Validate lightweight exercise recognition and assessment models
- Provide interpretable, research-grade exercise feedback
- Preserve user privacy by avoiding raw video storage
- Enable rapid experimentation and future scalability

1.3 Non-Goals (v1)

- Real-time corrective feedback during exercise
- Multi-exercise workout sessions
- App Store / production-level deployment
- Social features, gamification, or personalization
- Full offline operation

---

2. Target Users

Primary User

- Researcher / student demonstrator (controlled demo environment)

Secondary Users (Future)

- Fitness enthusiasts seeking automated form feedback
- Rehabilitation or coaching scenarios

---

3. Supported Platforms

Mobile

- Cross-platform mobile application (Flutter)
- Single-device demo focus (developer-owned device)

Backend

- Python-based backend running locally on developer laptop
- Internet connectivity required

---

4. High-Level System Architecture

4.1 Architecture Overview

The system follows a **hybrid architecture**: - Mobile device handles data capture and pose extraction - Backend handles all AI inference and feedback generation

This separation minimizes mobile complexity while maximizing research flexibility.

```
Mobile App (Flutter)
  └─ Camera
  └─ MediaPipe Pose (live)
  └─ Pose normalization
  └─ Session buffering
        ↓ JSON
Python Backend
  └─ Pose cleaning
  └─ Exercise recognition
  └─ Rep segmentation
  └─ Assessment models
  └─ Score aggregation
  └─ Feedback agent
        ↓ Response
Mobile App (Text Feedback UI)
```

---

5. Core User Flow (v1)

1. User selects:
2. Exercise type (from predefined list)
3. Camera view (front or side)
4. User positions phone and starts recording
5. App extracts pose in real time (no video storage)
6. User completes exercise and taps **Finish**
7. App sends pose sequence to backend
8. Backend processes data and returns feedback
9. App displays text-based feedback and scores

---

6. Functional Requirements

6.1 Mobile Application Requirements

6.1.1 Camera & Pose Extraction

- Capture video frames using device camera
- Run MediaPipe Pose live on each frame
- Extract 2D keypoints with confidence scores
- Timestamp each pose frame

6.1.2 Data Handling

- Normalize pose keypoints on-device
- Buffer pose frames for the full session
- Discard raw video immediately after pose extraction
- Support session lengths of ~30–60 seconds

6.1.3 Networking

- Send pose sequence as a single JSON payload at session end
- Receive structured feedback response from backend

6.1.4 UI & UX

- Simple, reliable UI optimized for demo usage
- Exercise and view selection screen
- Recording state indicator
- Text-only feedback screen displaying:
- Overall assessment summary
- Five aspect scores (0–10)

6.2 Backend Requirements (Python)

6.2.1 Pose Preprocessing

- Validate incoming pose data
- Remove low-confidence frames
- Handle missing or noisy joints

6.2.2 Exercise Recognition

- Input: pose sequence
- Output: predicted exercise label + confidence score
- Purpose: validation and sanity check

6.2.3 Rep Segmentation

- Heuristic, signal-based segmentation
- Use joint angles / key joint trajectories
- Over-segmentation allowed
- Drop unstable or low-quality reps

6.2.4 Assessment Models

- One assessment model per exercise per view
- Input: rep-level normalized pose windows
- Output: five aspect scores per rep (0–10)

6.2.5 Aggregation

- Aggregate rep-level scores to session-level scores
- Compute mean (and optionally variance) per aspect

6.2.6 Feedback Agent

- Interpret aggregated scores using rule-based logic
- Generate reflective, post-exercise feedback
- Optional LLM usage for natural language generation

---

7. Data Contracts

7.1 Mobile → Backend

- Exercise ID (selected by user)
- View type (front / side)
- Pose sequence:
- Frame timestamp
- Normalized keypoints (x, y)

- Confidence per keypoint

7.2 Backend → Mobile

- Exercise recognition confidence
- Session-level scores (5 aspects)
- Textual feedback summary
- Optional warnings (e.g., low confidence input)

---

8. Model & AI Constraints

- All models trained on normalized 2D pose data
- TensorFlow-based models
- Rep-level assessment models
- No raw RGB video input
- Models must run efficiently on CPU

---

9. Performance & Quality Requirements

- End-to-end processing time: < 5 seconds post-session
- Mobile app remains responsive during recording
- No video stored locally or transmitted
- Robust to minor pose noise and frame drops

---

10. Privacy & Security

- No raw video storage
- No personally identifiable information stored
- Pose data transmitted only for inference
- Local backend only (v1)

---

11. Extensibility & Future Work

11.1 On-Device Inference

- Exercise recognition and assessment models convertible to TFLite
- Clear abstraction boundaries allow future migration

11.2 Feature Expansion

- Real-time corrective feedback
- Multi-exercise sessions

- Visual pose playback
- Performance analytics

12. Risks & Mitigations

Risk	Mitigation
Pose noise	Confidence filtering & smoothing
Rep segmentation errors	Over-segmentation + filtering
Mobile instability	Thin client architecture
Time constraints	Controlled demo scope

13. Success Criteria

- Fully working end-to-end demo
- Accurate exercise recognition and assessment
- Clear, interpretable feedback
- Stable mobile experience
- Positive evaluation by academic supervisor

14. Milestones (Suggested)

1. Backend inference pipeline complete
2. Rep segmentation validated
3. Mobile pose capture functional
4. Mobile–backend integration
5. Demo polish and testing

Document Status: Final – v1 (Graduation Project MVP)