AI-Powered Sports Talent Assessment App

Complete Development Roadmap & Architecture

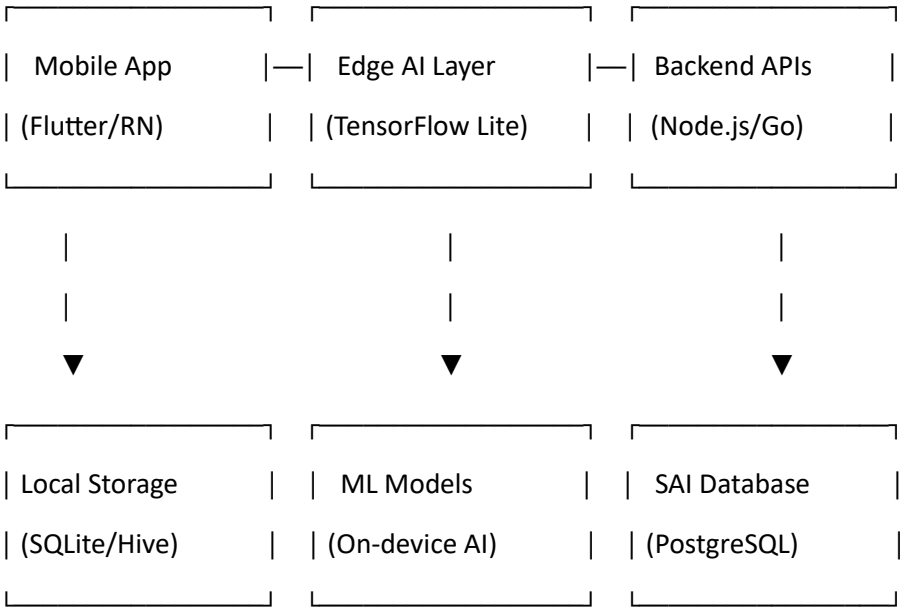
Executive Summary

This document outlines a comprehensive 4-week development plan for creating an AI-powered mobile platform that democratizes sports talent assessment across India, particularly targeting rural and remote areas.

---

1. Project Architecture Overview

1.1 System Architecture



1.2 Technology Stack Comparison

Frontend Mobile App

Technology	Pros	Cons	Recommendation
Flutter	✅ Single codebase, excellent performance, great camera plugins	❌ Larger app size	**RECOMMENDED**
React Native	✅ JavaScript familiarity, good community	❌ Bridge overhead, camera issues	Alternative
Native (Kotlin/Swift)	✅ Best performance, platform-specific features	❌ Dual development	For production scale

Backend Services

Technology	Use Case	Pros	Cons
Node.js + Express	API Gateway, Real-time features	✅ JavaScript ecosystem, fast development	❌ Single-threaded
Go + Gin	High-performance APIs	✅ Excellent concurrency, fast	❌ Learning curve
Python + FastAPI	ML integration	✅ AI/ML libraries, rapid prototyping	❌ Performance limitations

Database Solutions

Database	Use Case	Justification
PostgreSQL	Primary database	ACID compliance, JSON support, scalability
Redis	Caching, sessions	High-performance in-memory storage
SQLite	Local mobile storage	Offline capability, lightweight

AI/ML Framework

Framework	Platform	Best For
TensorFlow Lite	Mobile	On-device inference, model optimization
Core ML	iOS only	iOS-specific optimizations
ONNX Runtime	Cross-platform	Model interoperability

---

2. Detailed File Structure & Architecture

2.1 Mobile App Structure (Flutter)

```
sports_talent_app/
├── lib/
│   ├── main.dart                    # App entry point
│   ├── app/
│   │   ├── app.dart                 # App configuration
│   │   ├── routes.dart              # Navigation routes
│   │   └── themes.dart              # UI themes
│   ├── core/
│   │   ├── constants/
│   │   │   ├── api_constants.dart       # API endpoints
```

```
|   |   |   ├── app_constants.dart        # App-wide constants
|   |   |   └── test_constants.dart       # Sports test parameters
|   |   ├── errors/
|   |   |   ├── exceptions.dart           # Custom exceptions
|   |   |   └── failures.dart             # Error handling
|   |   ├── network/
|   |   |   ├── api_client.dart           # HTTP client wrapper
|   |   |   ├── network_info.dart         # Connectivity checker
|   |   |   └── interceptors.dart         # Request/response interceptors
|   |   └── utils/
|   |       ├── validators.dart           # Input validation
|   |       ├── formatters.dart           # Data formatting
|   |       └── permissions.dart          # Device permissions
|   ├── features/
|   |   ├── authentication/
|   |   |   ├── data/
|   |   |   |   ├── datasources/
|   |   |   |   |   ├── auth_local_datasource.dart
|   |   |   |   |   └── auth_remote_datasource.dart
|   |   |   |   ├── models/
|   |   |   |   |   └── user_model.dart
|   |   |   |   └── repositories/
|   |   |   |       └── auth_repository_impl.dart
|   |   |   ├── domain/
|   |   |   |   ├── entities/
|   |   |   |   |   └── user_entity.dart
|   |   |   |   ├── repositories/
|   |   |   |   |   └── auth_repository.dart
|   |   |   |   └── usecases/
|   |   |   |       ├── login_usecase.dart
```

```
|   |   |   |       ├── register_usecase.dart
|   |   |   |       └── logout_usecase.dart
|   |   |   └── presentation/
|   |   |       ├── bloc/
|   |   |       |   ├── auth_bloc.dart
|   |   |       |   ├── auth_event.dart
|   |   |       |   └── auth_state.dart
|   |   |       ├── pages/
|   |   |       |   ├── login_page.dart
|   |   |       |   ├── register_page.dart
|   |   |       |   └── profile_page.dart
|   |   |       └── widgets/
|   |   |           ├── auth_form.dart
|   |   |           └── social_login_buttons.dart
|   |   ├── video_recording/
|   |   |   ├── data/
|   |   |   |   ├── datasources/
|   |   |   |   |   ├── camera_datasource.dart
|   |   |   |   |   └── video_storage_datasource.dart
|   |   |   |   ├── models/
|   |   |   |   |   ├── video_model.dart
|   |   |   |   |   └── recording_session_model.dart
|   |   |   |   └── repositories/
|   |   |   |       └── video_repository_impl.dart
|   |   |   ├── domain/
|   |   |   |   ├── entities/
|   |   |   |   |   ├── video_entity.dart
|   |   |   |   |   └── test_session_entity.dart
|   |   |   |   ├── repositories/
|   |   |   |   |   └── video_repository.dart
```

```
|   |   |   |   └── usecases/
|   |   |   |       ├── start_recording_usecase.dart
|   |   |   |       ├── stop_recording_usecase.dart
|   |   |   |       └── save_video_usecase.dart
|   |   |   └── presentation/
|   |   |       ├── bloc/
|   |   |       |   ├── video_recording_bloc.dart
|   |   |       |   ├── video_recording_event.dart
|   |   |       |   └── video_recording_state.dart
|   |   |       ├── pages/
|   |   |       |   ├── test_selection_page.dart
|   |   |       |   ├── camera_page.dart
|   |   |       |   └── recording_review_page.dart
|   |   |       └── widgets/
|   |   |           ├── camera_overlay.dart
|   |   |           ├── test_timer.dart
|   |   |           ├── countdown_widget.dart
|   |   |           └── recording_controls.dart
|   |   ├── ai_analysis/
|   |   |   ├── data/
|   |   |   |   ├── datasources/
|   |   |   |   |   ├── ml_model_datasource.dart
|   |   |   |   |   └── analysis_cache_datasource.dart
|   |   |   |   ├── models/
|   |   |   |   |   ├── analysis_result_model.dart
|   |   |   |   |   └── performance_metrics_model.dart
|   |   |   |   └── repositories/
|   |   |   |       └── ai_analysis_repository_impl.dart
|   |   |   ├── domain/
|   |   |   |   ├── entities/
```

```
│   │   │   │   ├── analysis_result_entity.dart
│   │   │   │   └── performance_data_entity.dart
│   │   │   ├── repositories/
│   │   │   │   └── ai_analysis_repository.dart
│   │   │   └── usecases/
│   │   │       ├── analyze_vertical_jump_usecase.dart
│   │   │       ├── count_situps_usecase.dart
│   │   │       ├── analyze_shuttle_run_usecase.dart
│   │   │       ├── detect_cheating_usecase.dart
│   │   │       └── benchmark_performance_usecase.dart
│   │   └── presentation/
│   │       ├── bloc/
│   │       │   ├── ai_analysis_bloc.dart
│   │       │   ├── ai_analysis_event.dart
│   │       │   └── ai_analysis_state.dart
│   │       ├── pages/
│   │       │   ├── analysis_loading_page.dart
│   │       │   ├── results_page.dart
│   │       │   └── performance_dashboard_page.dart
│   │       └── widgets/
│   │           ├── analysis_progress.dart
│   │           ├── performance_chart.dart
│   │           ├── benchmark_comparison.dart
│   │           └── cheat_detection_indicator.dart
│   ├── gamification/
│   │   ├── data/
│   │   │   ├── datasources/
│   │   │   │   ├── achievements_datasource.dart
│   │   │   │   └── leaderboard_datasource.dart
│   │   │   ├── models/
```

```
│   │   │   │   ├── badge_model.dart
│   │   │   │   ├── achievement_model.dart
│   │   │   │   └── leaderboard_model.dart
│   │   │   └── repositories/
│   │   │       └── gamification_repository_impl.dart
│   │   ├── domain/
│   │   │   ├── entities/
│   │   │   │   ├── badge_entity.dart
│   │   │   │   └── user_progress_entity.dart
│   │   │   ├── repositories/
│   │   │   │   └── gamification_repository.dart
│   │   │   └── usecases/
│   │   │       ├── unlock_achievement_usecase.dart
│   │   │       ├── get_leaderboard_usecase.dart
│   │   │       └── update_progress_usecase.dart
│   │   └── presentation/
│   │       ├── bloc/
│   │       │   ├── gamification_bloc.dart
│   │       │   ├── gamification_event.dart
│   │       │   └── gamification_state.dart
│   │       ├── pages/
│   │       │   ├── achievements_page.dart
│   │       │   ├── leaderboard_page.dart
│   │       │   └── progress_page.dart
│   │       └── widgets/
│   │           ├── badge_widget.dart
│   │           ├── progress_bar.dart
│   │           ├── leaderboard_item.dart
│   │           └── achievement_popup.dart
│   └── data_sync/
```

```
|   |       ├── data/
|   |       |   ├── datasources/
|   |       |   |   ├── sync_local_datasource.dart
|   |       |   |   └── sync_remote_datasource.dart
|   |       |   ├── models/
|   |       |   |   ├── sync_queue_model.dart
|   |       |   |   └── upload_status_model.dart
|   |       |   └── repositories/
|   |       |       └── sync_repository_impl.dart
|   |       ├── domain/
|   |       |   ├── entities/
|   |       |   |   └── sync_item_entity.dart
|   |       |   ├── repositories/
|   |       |   |   └── sync_repository.dart
|   |       |   └── usecases/
|   |       |       ├── queue_for_sync_usecase.dart
|   |       |       ├── sync_data_usecase.dart
|   |       |       └── retry_failed_sync_usecase.dart
|   |       └── presentation/
|   |           ├── bloc/
|   |           |   ├── sync_bloc.dart
|   |           |   ├── sync_event.dart
|   |           |   └── sync_state.dart
|   |           └── widgets/
|   |               ├── sync_status_indicator.dart
|   |               └── upload_progress.dart
|   ├── shared/
|   |   ├── widgets/
|   |   |   ├── custom_button.dart
|   |   |   ├── loading_indicator.dart
```

```
|   |   |   ├── error_widget.dart
|   |   |   ├── video_player.dart
|   |   |   └── performance_card.dart
|   |   ├── services/
|   |   |   ├── dependency_injection.dart    # Service locator
|   |   |   ├── local_storage_service.dart   # SQLite/Hive wrapper
|   |   |   ├── notification_service.dart    # Push notifications
|   |   |   ├── biometric_service.dart       # Fingerprint/face recognition
|   |   |   └── ml_service.dart              # TensorFlow Lite wrapper
|   |   └── extensions/
|   |       ├── string_extensions.dart
|   |       ├── datetime_extensions.dart
|   |       └── number_extensions.dart
|   └── models/
|       ├── ai_models/                  # TensorFlow Lite models
|       |   ├── vertical_jump_detector.tflite
|       |   ├── situp_counter.tflite
|       |   ├── shuttle_run_analyzer.tflite
|       |   ├── pose_estimation.tflite
|       |   └── cheat_detection.tflite
|       └── benchmark_data/
|           ├── age_gender_benchmarks.json
|           └── regional_benchmarks.json
├── assets/
|   ├── images/
|   |   ├── logos/
|   |   ├── icons/
|   |   ├── badges/
|   |   └── tutorials/
|   ├── videos/
```

```
|   |   └── tutorial_videos/
|   ├── animations/
|   |   └── lottie_files/
|   └── fonts/
├── test/
|   ├── unit/
|   ├── widget/
|   └── integration/
├── pubspec.yaml                    # Dependencies
└── README.md
```

2.2 Backend API Structure (Node.js)

```
sports-talent-backend/
├── src/
|   ├── app.js                      # Express app setup
|   ├── server.js                   # Server entry point
|   ├── config/
|   |   ├── database.js             # DB configuration
|   |   ├── redis.js                # Redis configuration
|   |   ├── aws.js                  # AWS S3 configuration
|   |   ├── jwt.js                  # JWT configuration
|   |   └── multer.js               # File upload configuration
|   ├── controllers/
|   |   ├── authController.js       # Authentication endpoints
|   |   ├── userController.js       # User management
|   |   ├── videoController.js      # Video upload/processing
|   |   ├── analysisController.js   # AI analysis results
|   |   ├── benchmarkController.js  # Performance benchmarks
|   |   ├── leaderboardController.js # Gamification features
|   |   └── adminController.js      # SAI dashboard
|   ├── middleware/
```

```
|   |   ├── auth.js                  # JWT verification
|   |   ├── validation.js            # Request validation
|   |   ├── rateLimiting.js          # API rate limiting
|   |   ├── errorHandler.js          # Global error handling
|   |   └── fileUpload.js            # File processing
|   ├── models/
|   |   ├── User.js                  # User schema
|   |   ├── Athlete.js               # Athlete profile
|   |   ├── TestSession.js           # Test session data
|   |   ├── VideoSubmission.js       # Video metadata
|   |   ├── AnalysisResult.js        # AI analysis results
|   |   ├── Achievement.js           # Gamification badges
|   |   └── Benchmark.js             # Performance standards
|   ├── routes/
|   |   ├── auth.js                  # Authentication routes
|   |   ├── athletes.js              # Athlete management
|   |   ├── tests.js                 # Sports test routes
|   |   ├── videos.js                # Video handling
|   |   ├── analysis.js              # AI analysis
|   |   ├── gamification.js          # Badges and achievements
|   |   └── admin.js                 # SAI dashboard routes
|   ├── services/
|   |   ├── authService.js           # Authentication logic
|   |   ├── videoProcessingService.js    # Video processing pipeline
|   |   ├── aiAnalysisService.js     # AI model integration
|   |   ├── benchmarkService.js      # Performance comparison
|   |   ├── notificationService.js   # Push notifications
|   |   ├── storageService.js        # File storage (AWS S3)
|   |   ├── encryptionService.js     # Data encryption
|   |   └── auditService.js          # Activity logging
```

```
|   ├── utils/
|   |   ├── validators.js              # Data validation
|   |   ├── helpers.js                 # Utility functions
|   |   ├── constants.js               # Application constants
|   |   ├── logger.js                  # Logging utility
|   |   └── encryption.js              # Security utilities
|   ├── jobs/
|   |   ├── videoProcessingJob.js      # Background video processing
|   |   ├── aiAnalysisJob.js           # Batch AI analysis
|   |   ├── dataCleanupJob.js          # Database maintenance
|   |   └── reportGenerationJob.js     # Periodic reports
|   └── database/
|       ├── migrations/
|       |   ├── 001_create_users_table.js
|       |   ├── 002_create_athletes_table.js
|       |   ├── 003_create_test_sessions_table.js
|       |   ├── 004_create_video_submissions_table.js
|       |   ├── 005_create_analysis_results_table.js
|       |   └── 006_create_benchmarks_table.js
|       ├── seeds/
|       |   ├── benchmark_data.js
|       |   └── sample_users.js
|       └── config.js
├── ai_models/
|   ├── preprocessing/
|   |   ├── video_preprocessor.py      # Video frame extraction
|   |   ├── pose_estimator.py          # Body pose detection
|   |   └── frame_analyzer.py          # Individual frame analysis
|   ├── models/
|   |   ├── vertical_jump/
```

```
|   |   |   ├── jump_detector.py          # Jump height calculation
|   |   |   ├── model.h5                  # Trained model
|   |   |   └── preprocessing.py          # Jump-specific preprocessing
|   |   ├── situp_counter/
|   |   |   ├── rep_counter.py            # Repetition counting
|   |   |   ├── model.h5
|   |   |   └── form_analyzer.py          # Exercise form validation
|   |   ├── shuttle_run/
|   |   |   ├── speed_analyzer.py         # Speed and agility analysis
|   |   |   ├── model.h5
|   |   |   └── distance_calculator.py    # Distance measurement
|   |   └── cheat_detection/
|   |       ├── anomaly_detector.py       # Unusual pattern detection
|   |       ├── model.h5
|   |       └── authenticity_checker.py   # Video manipulation detection
|   ├── training/
|   |   ├── data_preparation.py           # Dataset preparation
|   |   ├── model_training.py             # Training pipeline
|   |   ├── evaluation.py                 # Model evaluation
|   |   └── optimization.py               # Model optimization for mobile
|   └── inference/
|       ├── batch_inference.py            # Server-side batch processing
|       ├── real_time_inference.py        # Real-time analysis
|       └── model_server.py               # Model serving API
├── scripts/
|   ├── setup_database.js                 # Database initialization
|   ├── migrate.js                        # Migration runner
|   ├── seed.js                           # Data seeding
|   ├── backup.js                         # Database backup
|   └── deploy.js                         # Deployment script
```

```
├── tests/
|   ├── unit/
|   ├── integration/
|   └── e2e/
├── docs/
|   ├── api_documentation.md
|   ├── deployment_guide.md
|   └── troubleshooting.md
├── docker/
|   ├── Dockerfile                  # Application container
|   ├── docker-compose.yml          # Multi-service setup
|   └── nginx.conf                  # Load balancer configuration
├── .env.example                    # Environment variables template
├── package.json                    # Dependencies
└── README.md
```

2.3 SAI Admin Dashboard Structure (React)

```
sai-admin-dashboard/
├── public/
├── src/
|   ├── components/
|   |   ├── common/
|   |   |   ├── Header.jsx
|   |   |   ├── Sidebar.jsx
|   |   |   ├── Footer.jsx
|   |   |   └── LoadingSpinner.jsx
|   |   ├── charts/
|   |   |   ├── PerformanceChart.jsx
|   |   |   ├── RegionalDistribution.jsx
|   |   |   └── TalentPipeline.jsx
|   |   ├── athletes/
```

```
|   |   |   ├── AthleteProfile.jsx
|   |   |   ├── AthleteList.jsx
|   |   |   ├── PerformanceAnalysis.jsx
|   |   |   └── VideoReview.jsx
|   |   └── admin/
|   |       ├── UserManagement.jsx
|   |       ├── SystemHealth.jsx
|   |       └── Reports.jsx
|   ├── pages/
|   |   ├── Dashboard.jsx
|   |   ├── Athletes.jsx
|   |   ├── Analytics.jsx
|   |   ├── Settings.jsx
|   |   └── Reports.jsx
|   ├── services/
|   |   ├── api.js
|   |   ├── auth.js
|   |   └── websocket.js
|   ├── utils/
|   |   ├── constants.js
|   |   ├── helpers.js
|   |   └── validators.js
|   ├── styles/
|   |   ├── globals.css
|   |   └── components/
|   ├── App.jsx
|   └── index.js
├── package.json
└── README.md
```

---

3. AI/ML Models Architecture

3.1 Model Specifications

Vertical Jump Detection Model

python

 # Model Architecture

Input: Video frames (224x224x3)

├── CNN Feature Extractor (MobileNetV2 backbone)

├── Temporal Analysis Layer (LSTM)

├── Jump Detection Head

├── Height Calculation Module

└── Output: Jump height in cm, confidence score

 # Key Features:

- Real-time pose estimation
- Ground plane detection
- Jump trajectory analysis
- Height measurement accuracy: ±2cm

Sit-up Counter Model

python

 # Model Architecture

Input: Video frames (224x224x3)

├── Pose Estimation (MediaPipe)

├── Hip/Shoulder Angle Calculator

├── Rep Detection Algorithm

├── Form Validation Module

└── Output: Rep count, form score (0-100)

 # Key Features:

- Angle-based counting
- Form quality assessment
- False positive reduction

- Accuracy: 95%+ in controlled conditions

Shuttle Run Analyzer

python

 # Model Architecture

Input: Video frames (224x224x3)

├── Object Detection (Person tracking)

├── Speed Estimation Module

├── Distance Calculation

├── Agility Metrics Calculator

└── Output: Time, speed, agility score

 # Key Features:

- Multi-person tracking

- Speed profile analysis

- Direction change detection

- Timing accuracy: ±0.1 seconds

Cheat Detection Model

python

 # Model Architecture

Input: Video frames + metadata

├── Video Authenticity Checker

├── Motion Pattern Analyzer

├── Environmental Consistency Checker

├── Anomaly Detection Module

└── Output: Authenticity score (0-1), flags

 # Detection Capabilities:

- Video manipulation (deepfake, editing)

- Unrealistic motion patterns

- Environmental inconsistencies

- Equipment tampering

3.2 Mobile Optimization for YOLOv8 + OpenCV

ONNX Model Conversion Strategy

```
# YOLOv8 to ONNX conversion for mobile deployment
from ultralytics import YOLO
import torch

# Convert YOLOv8 to ONNX
model = YOLO('yolov8n-pose.pt')
model.export(format='onnx', dynamic=True, simplify=True)

# Optimization parameters:
# - Dynamic input shapes for flexible inference
# - Graph simplification for reduced model size
# - FP16 precision for 2x speed improvement
# - Size reduction: 6MB → 3MB (with quantization)
# - Inference speed: 40+ FPS on modern mobile devices
```

Quantization and Pruning Strategy

```
# Post-training quantization for mobile deployment
import onnxruntime as ort
from onnxruntime.quantization import quantize_dynamic, QuantType

def optimize_for_mobile(model_path):
    # Dynamic quantization
    quantized_model = quantize_dynamic(
        model_path,
        model_path.replace('.onnx', '_quantized.onnx'),
        weight_type=QuantType.QUInt8
    )

    # Performance improvements:
    # - Model size: 4x smaller
```

 # - Inference speed: 2-3# AI-Powered Sports Talent Assessment App

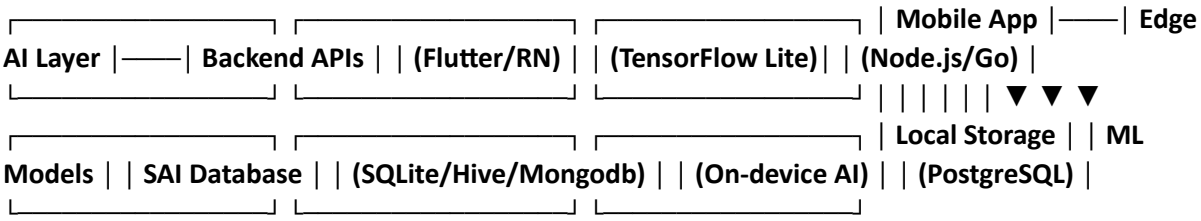
Complete Development Roadmap & Architecture

Executive Summary

This document outlines a comprehensive 4-week development plan for creating an AI-powered mobile platform that democratizes sports talent assessment across India, particularly targeting rural and remote areas.

1. Project Architecture Overview

1.1 System Architecture



1.2 Technology Stack Comparison

Frontend Mobile App

| Technology | Pros | Cons | Recommendation |

|------------|------|------|----------------|

| **Flutter | ✅ Single codebase, excellent performance, great camera plugins | ❌ Larger app size | **RECOMMENDED** |**

| React Native | ✅ JavaScript familiarity, good community | ❌ Bridge overhead, camera issues | Alternative |

| Native (Kotlin/Swift) | ✅ Best performance, platform-specific features | ❌ Dual development | For production scale |

Backend Services

| Technology | Use Case | Pros | Cons |

|------------|----------|------|------|

| **Node.js + Express** | API Gateway, Real-time features | ✅ JavaScript ecosystem, fast development | ❌ Single-threaded |

| **Go + Gin** | High-performance APIs | ✅ Excellent concurrency, fast | ❌ Learning curve |

| Python + FastAPI | ML integration | ✅ AI/ML libraries, rapid prototyping | ❌ Performance limitations |

Database Solutions

| Database | Use Case | Justification |

|----------|----------|---------------|

| **PostgreSQL** | Primary database | ACID compliance, JSON support, scalability |

| **Redis** | Caching, sessions | High-performance in-memory storage |

| **SQLite** | Local mobile storage | Offline capability, lightweight |

AI/ML Framework

| Framework | Platform | Best For |

|-----------|----------|----------|

| **TensorFlow Lite** | Mobile | On-device inference, model optimization |

| **Core ML** | iOS only | iOS-specific optimizations |

| **ONNX Runtime** | Cross-platform | Model interoperability |

---

2. Detailed File Structure & Architecture

2.1 Mobile App Structure (Flutter)

sports_talent_app/ ├── lib/ │ ├── main.dart # App entry point │ ├── app/ │ │ ├── app.dart # App configuration │ │ ├── routes.dart # Navigation routes │ │ └── themes.dart # UI themes │ ├── core/ │ │ ├── constants/ │ │ │ ├── api_constants.dart # API endpoints │ │ │ ├── app_constants.dart # App-wide constants │ │ │ └── test_constants.dart # Sports test parameters │ │ ├── errors/ │ │ │ ├── exceptions.dart # Custom exceptions │ │ │ └── failures.dart # Error handling │ │ ├── network/ │ │ │ ├── api_client.dart # HTTP client wrapper │ │ │ ├── network_info.dart # Connectivity checker │ │ │ └── interceptors.dart # Request/response interceptors │ │ └── utils/ │ │ ├── validators.dart # Input validation │ │ ├── formatters.dart #

Data formatting | | └── permissions.dart # Device permissions | ├── features/ | | ├── authentication/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── auth_local_datasource.dart | | | | | └── auth_remote_datasource.dart | | | | ├── models/ | | | | | └── user_model.dart | | | | └── repositories/ | | | | └── auth_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | └── user_entity.dart | | | | ├── repositories/ | | | | | └── auth_repository.dart | | | | └── usecases/ | | | | ├── login_usecase.dart | | | | ├── register_usecase.dart | | | | └── logout_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── auth_bloc.dart | | | | ├── auth_event.dart | | | | └── auth_state.dart | | | ├── pages/ | | | | ├── login_page.dart | | | | ├── register_page.dart | | | | └── profile_page.dart | | | └── widgets/ | | | ├── auth_form.dart | | | └── social_login_buttons.dart | | ├── video_recording/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── camera_datasource.dart | | | | | └── video_storage_datasource.dart | | | | ├── models/ | | | | | ├── video_model.dart | | | | | └── recording_session_model.dart | | | | └── repositories/ | | | | └── video_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── video_entity.dart | | | | | └── test_session_entity.dart | | | | ├── repositories/ | | | | | └── video_repository.dart | | | | └── usecases/ | | | | ├── start_recording_usecase.dart | | | | ├── stop_recording_usecase.dart | | | | └── save_video_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── video_recording_bloc.dart | | | | ├── video_recording_event.dart | | | | └── video_recording_state.dart | | | ├── pages/ | | | | ├── test_selection_page.dart | | | | ├── camera_page.dart | | | | └── recording_review_page.dart | | | └── widgets/ | | | ├── camera_overlay.dart | | | ├── test_timer.dart | | | ├── countdown_widget.dart | | | └── recording_controls.dart | | ├── ai_analysis/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── ml_model_datasource.dart | | | | | └── analysis_cache_datasource.dart | | | | ├── models/ | | | | | ├── analysis_result_model.dart | | | | | └── performance_metrics_model.dart | | | | └── repositories/ | | | | └── ai_analysis_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── analysis_result_entity.dart | | | | | └── performance_data_entity.dart | | | | ├── repositories/ | | | | | └── ai_analysis_repository.dart | | | | └── usecases/ | | | | ├── analyze_vertical_jump_usecase.dart | | | | ├── count_situps_usecase.dart | | | | ├── analyze_shuttle_run_usecase.dart | | | | ├── detect_cheating_usecase.dart | | | | └── benchmark_performance_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── ai_analysis_bloc.dart | | | | ├── ai_analysis_event.dart | | | | └── ai_analysis_state.dart | | | ├── pages/ | | | | ├── analysis_loading_page.dart | | | | ├── results_page.dart | | | | └── performance_dashboard_page.dart | | | └── widgets/ | | | ├── analysis_progress.dart | | | ├── performance_chart.dart | | | ├── benchmark_comparison.dart | | | └── cheat_detection_indicator.dart | | ├── gamification/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── achievements_datasource.dart | | | | | └── leaderboard_datasource.dart | | | | ├── models/ | | | | | ├── badge_model.dart | | | | | ├── achievement_model.dart | | | | | └── leaderboard_model.dart | | | | └── repositories/ | | | | └── gamification_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── badge_entity.dart | | | | | └── user_progress_entity.dart | | | | ├── repositories/ | | | | | └── gamification_repository.dart | | | | └── usecases/ | | | | ├── unlock_achievement_usecase.dart | | | | ├── get_leaderboard_usecase.dart | | | | └── update_progress_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── gamification_bloc.dart | | | | ├── gamification_event.dart | | | | └── gamification_state.dart | | | ├── pages/ | | | | ├── achievements_page.dart | | | | ├── leaderboard_page.dart | | | | └── progress_page.dart | | | └── widgets/ | | | ├──

badge_widget.dart | | | ├── progress_bar.dart | | | ├── leaderboard_item.dart | | | └── achievement_popup.dart | | └── data_sync/ | | ├── data/ | | | ├── datasources/ | | | | ├── sync_local_datasource.dart | | | | └── sync_remote_datasource.dart | | | ├── models/ | | | | ├── sync_queue_model.dart | | | | └── upload_status_model.dart | | | └── repositories/ | | | └── sync_repository_impl.dart | | ├── domain/ | | | ├── entities/ | | | | └── sync_item_entity.dart | | | ├── repositories/ | | | | └── sync_repository.dart | | | └── usecases/ | | | ├── queue_for_sync_usecase.dart | | | ├── sync_data_usecase.dart | | | └── retry_failed_sync_usecase.dart | | └── presentation/ | | ├── bloc/ | | | ├── sync_bloc.dart | | | ├── sync_event.dart | | | └── sync_state.dart | | └── widgets/ | | ├── sync_status_indicator.dart | | └── upload_progress.dart | ├── shared/ | | ├── widgets/ | | | ├── custom_button.dart | | | ├── loading_indicator.dart | | | ├── error_widget.dart | | | ├── video_player.dart | | | └── performance_card.dart | | ├── services/ | | | ├── dependency_injection.dart # Service locator | | | ├── local_storage_service.dart # SQLite/Hive wrapper | | | ├── notification_service.dart # Push notifications | | | ├── biometric_service.dart # Fingerprint/face recognition | | | └── ml_service.dart # TensorFlow Lite wrapper | | └── extensions/ | | ├── string_extensions.dart | | ├── datetime_extensions.dart | | └── number_extensions.dart | └── models/ | ├── ai_models/ # TensorFlow Lite models | | ├── vertical_jump_detector.tflite | | ├── situp_counter.tflite | | ├── shuttle_run_analyzer.tflite | | ├── pose_estimation.tflite | | └── cheat_detection.tflite | └── benchmark_data/ | ├── age_gender_benchmarks.json | └── regional_benchmarks.json ├── assets/ | ├── images/ | | ├── logos/ | | ├── icons/ | | ├── badges/ | | └── tutorials/ | ├── videos/ | | └── tutorial_videos/ | ├── animations/ | | └── lottie_files/ | └── fonts/ ├── test/ | ├── unit/ | ├── widget/ | └── integration/ ├── pubspec.yaml # Dependencies └── README.md

2.2 Backend API Structure (Node.js)

sports-talent-backend/ ├── src/ | ├── app.js # Express app setup | ├── server.js # Server entry point | ├── config/ | | ├── database.js # DB configuration | | ├── redis.js # Redis configuration | | ├── aws.js # AWS S3 configuration | | ├── jwt.js # JWT configuration | | └── multer.js # File upload configuration | ├── controllers/ | | ├── authController.js # Authentication endpoints | | ├── userController.js # User management | | ├── videoController.js # Video upload/processing | | ├── analysisController.js # AI analysis results | | ├── benchmarkController.js # Performance benchmarks | | ├── leaderboardController.js # Gamification features | | └── adminController.js # SAI dashboard | ├── middleware/ | | ├── auth.js # JWT verification | | ├── validation.js # Request validation | | ├── rateLimiting.js # API rate limiting | | ├── errorHandler.js # Global error handling | | └── fileUpload.js # File processing | ├── models/ | | ├── User.js # User schema | | ├── Athlete.js # Athlete profile | | ├── TestSession.js # Test session data | | ├── VideoSubmission.js # Video metadata | | ├── AnalysisResult.js # AI analysis results | | ├── Achievement.js # Gamification badges | | └── Benchmark.js # Performance standards | ├── routes/ | | ├── auth.js # Authentication routes | | ├── athletes.js # Athlete management | | ├── tests.js # Sports test routes | | ├── videos.js # Video handling | | ├── analysis.js # AI analysis | | ├── gamification.js # Badges and achievements | | └── admin.js # SAI dashboard routes | ├── services/ | | ├── authService.js # Authentication logic | | ├── videoProcessingService.js # Video processing pipeline | | ├── aiAnalysisService.js # AI model integration | | ├── benchmarkService.js # Performance comparison | | ├── notificationService.js # Push notifications

| | ├── storageService.js # File storage (AWS S3) | | ├── encryptionService.js # Data encryption | | └── auditService.js # Activity logging | ├── utils/ | | ├── validators.js # Data validation | | ├── helpers.js # Utility functions | | ├── constants.js # Application constants | | ├── logger.js # Logging utility | | └── encryption.js # Security utilities | ├── jobs/ | | ├── videoProcessingJob.js # Background video processing | | ├── aiAnalysisJob.js # Batch AI analysis | | ├── dataCleanupJob.js # Database maintenance | | └── reportGenerationJob.js # Periodic reports | └── database/ | ├── migrations/ | | ├── 001_create_users_table.js | | ├── 002_create_athletes_table.js | | ├── 003_create_test_sessions_table.js | | ├── 004_create_video_submissions_table.js | | ├── 005_create_analysis_results_table.js | | └── 006_create_benchmarks_table.js | ├── seeds/ | | ├── benchmark_data.js | | └── sample_users.js | └── config.js ├── ai_models/ | ├── preprocessing/ | | ├── video_preprocessor.py # Video frame extraction | | ├── pose_estimator.py # Body pose detection | | └── frame_analyzer.py # Individual frame analysis | ├── models/ | | ├── vertical_jump/ | | | ├── jump_detector.py # Jump height calculation | | | ├── model.h5 # Trained model | | | └── preprocessing.py # Jump-specific preprocessing | | ├── situp_counter/ | | | ├── rep_counter.py # Repetition counting | | | ├── model.h5 | | | └── form_analyzer.py # Exercise form validation | | ├── shuttle_run/ | | | ├── speed_analyzer.py # Speed and agility analysis | | | ├── model.h5 | | | └── distance_calculator.py # Distance measurement | | └── cheat_detection/ | | ├── anomaly_detector.py # Unusual pattern detection | | ├── model.h5 | | └── authenticity_checker.py # Video manipulation detection | ├── training/ | | ├── data_preparation.py # Dataset preparation | | ├── model_training.py # Training pipeline | | ├── evaluation.py # Model evaluation | | └── optimization.py # Model optimization for mobile | └── inference/ | ├── batch_inference.py # Server-side batch processing | ├── real_time_inference.py # Real-time analysis | └── model_server.py # Model serving API ├── scripts/ | ├── setup_database.js # Database initialization | ├── migrate.js # Migration runner | ├── seed.js # Data seeding | ├── backup.js # Database backup | └── deploy.js # Deployment script ├── tests/ | ├── unit/ | ├── integration/ | └── e2e/ ├── docs/ | ├── api_documentation.md | ├── deployment_guide.md | └── troubleshooting.md ├── docker/ | ├── Dockerfile # Application container | ├── docker-compose.yml # Multi-service setup | └── nginx.conf # Load balancer configuration ├── .env.example # Environment variables template ├── package.json # Dependencies └── README.md

2.3 SAI Admin Dashboard Structure (React)

sai-admin-dashboard/ ├── public/ ├── src/ | ├── components/ | | ├── common/ | | | ├── Header.jsx | | | ├── Sidebar.jsx | | | ├── Footer.jsx | | | └── LoadingSpinner.jsx | | ├── charts/ | | | ├── PerformanceChart.jsx | | | ├── RegionalDistribution.jsx | | | └── TalentPipeline.jsx | | ├── athletes/ | | | ├── AthleteProfile.jsx | | | ├── AthleteList.jsx | | | ├── PerformanceAnalysis.jsx | | | └── VideoReview.jsx | | └── admin/ | | ├── UserManagement.jsx | | ├── SystemHealth.jsx | | └── Reports.jsx | ├── pages/ | | ├── Dashboard.jsx | | ├── Athletes.jsx | | ├── Analytics.jsx | | ├── Settings.jsx | | └── Reports.jsx | ├── services/ | | ├── api.js | | ├── auth.js | | └── websocket.js | ├── utils/ | | ├── constants.js | | ├── helpers.js | | └── validators.js | ├── styles/ | | ├── globals.css | | └── components/ | ├── App.jsx | └── index.js ├── package.json └── README.md

---

3. AI/ML Models Architecture

3.1 Model Specifications

Vertical Jump Detection Model

```python
# Model Architecture
Input: Video frames (224x224x3)
 ├── CNN Feature Extractor (MobileNetV2 backbone)
 ├── Temporal Analysis Layer (LSTM)
 ├── Jump Detection Head
 ├── Height Calculation Module
 └── Output: Jump height in cm, confidence score

# Key Features:
- Real-time pose estimation
- Ground plane detection
- Jump trajectory analysis
- Height measurement accuracy: ±2cm
Sit-up Counter Model
# Model Architecture
Input: Video frames (224x224x3)
 ├── Pose Estimation (MediaPipe)
 ├── Hip/Shoulder Angle Calculator
 ├── Rep Detection Algorithm
 ├── Form Validation Module
 └── Output: Rep count, form score (0-100)
```

 # Key Features:

- Angle-based counting

- Form quality assessment

- False positive reduction

- Accuracy: 95%+ in controlled conditions

Shuttle Run Analyzer

 # Model Architecture

Input: Video frames (224x224x3)

├── Object Detection (Person tracking)

├── Speed Estimation Module

├── Distance Calculation

├── Agility Metrics Calculator

└── Output: Time, speed, agility score

 # Key Features:

- Multi-person tracking

- Speed profile analysis

- Direction change detection

- Timing accuracy: ±0.1 seconds

Cheat Detection Model

 # Model Architecture

Input: Video frames + metadata

├── Video Authenticity Checker

├── Motion Pattern Analyzer

├── Environmental Consistency Checker

├── Anomaly Detection Module

└── Output: Authenticity score (0-1), flags

 # Detection Capabilities:

- Video manipulation (deepfake, editing)

- Unrealistic motion patterns

- Environmental inconsistencies

- Equipment tampering

3.2 Model Optimization for Mobile

Quantization Strategy

```
# Post-training quantization
model_int8 = tf.lite.TFLiteConverter.from_keras_model(model)
model_int8.optimizations = [tf.lite.Optimize.DEFAULT]
model_int8.target_spec.supported_types = [tf.lite.constants.INT8]

# Size reduction: 4x smaller
# Inference speed: 2-3x faster
# Accuracy loss: <5%
```

Model Pruning

```
# Structured pruning for mobile deployment
import tensorflow_model_optimization as tfmot

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude
pruned_model = prune_low_magnitude(model,
                                   pruning_schedule=tfmot.sparsity.keras.PolynomialDecay(
                                     initial_sparsity=0.50,
                                     final_sparsity=0.90,
                                     begin_step=0,
                                     end_step=1000))
```

---

4. 4-Week Development Timeline

Week 1: Foundation & Core Setup

Days 1-2: Project Setup & Architecture

- **[ ] Set up development environment**
- **[ ] Initialize Flutter project with clean architecture**
- **[ ] Set up backend Node.js project structure**
- **[ ] Configure database schemas (PostgreSQL)**

- **[ ] Set up Redis for caching**
- **[ ] Configure CI/CD pipeline (GitHub Actions)**

Days 3-4: Authentication & User Management

- **[ ] Implement user registration/login (mobile + backend)**
- **[ ] JWT token implementation**
- **[ ] Basic user profile management**
- **[ ] Database user tables and relationships**
- **[ ] Input validation and error handling**

Days 5-7: Video Recording Infrastructure

- **[ ] Camera integration (Flutter)**
- **[ ] Video recording functionality**
- **[ ] Local video storage**
- **[ ] Basic video player**
- **[ ] File upload preparation**
- **[ ] Video compression implementation**

Week 2: AI/ML Integration & Core Features

Days 8-10: AI Models Development

- **[ ] Set up TensorFlow/PyTorch training environment**
- **[ ] Collect and prepare training datasets**
- **[ ] Train vertical jump detection model**
- **[ ] Train sit-up counter model**
- **[ ] Train shuttle run analyzer**
- **[ ] Convert models to TensorFlow Lite**

Days 11-12: Mobile AI Integration

- **[ ] Integrate TensorFlow Lite into Flutter**
- **[ ] Implement on-device inference pipeline**
- **[ ] Pose estimation integration (MediaPipe)**
- **[ ] Real-time video analysis**
- **[ ] Performance optimization for low-end devices**

Days 13-14: Sports Test Implementation

- **[ ] Implement test selection UI**

- **[ ] Create test-specific recording interfaces**
- **[ ] Add countdown timers and instructions**
- **[ ] Implement basic analysis results display**
- **[ ] Test validation and error handling**

Week 3: Advanced Features & Backend

Days 15-16: Cheat Detection & Validation

- **[ ] Implement cheat detection algorithms**
- **[ ] Video authenticity verification**
- **[ ] Motion pattern analysis**
- **[ ] Environmental consistency checks**
- **[ ] Integration with main analysis pipeline**

Days 17-18: Backend API Development

- **[ ] Complete REST API endpoints**
- **[ ] Video upload and processing pipeline**
- **[ ] Background job processing (Bull Queue)**
- **[ ] Data synchronization logic**
- **[ ] Performance benchmarking system**

Days 19-21: Gamification & User Experience

- **[ ] Achievement system implementation**
- **[ ] Badge unlocking mechanism**
- **[ ] Leaderboard functionality**
- **[ ] Progress tracking**
- **[ ] Push notifications setup**
- **[ ] UI/UX polishing**

Week 4: Integration, Testing & Deployment

Days 22-24: SAI Admin Dashboard

- **[ ] React admin dashboard setup**
- **[ ] Athlete data visualization**
- **[ ] Performance analytics charts**
- **[ ] Video review interface**
- **[ ] Report generation features**

- [ ] Real-time data updates (WebSocket)

Days 25-26: Comprehensive Testing & Quality Assurance

- [ ] Unit testing for YOLOv8 + OpenCV integration
- [ ] Integration testing for MongoDB/MySQL data flows
- [ ] Performance testing on various mobile devices
- [ ] AI model accuracy validation with test datasets
- [ ] Video processing pipeline stress testing
- [ ] Offline functionality testing
- [ ] Security testing for data transmission
- [ ] User acceptance testing with beta users

Days 27-28: Deployment & Production Setup

- [ ] Set up production MongoDB cluster / MySQL with replication
- [ ] Configure AWS S3 for video storage with CDN
- [ ] Deploy backend APIs with load balancing
- [ ] Set up Redis cluster for caching and sessions
- [ ] Configure CI/CD pipeline for automated deployments
- [ ] Set up monitoring and logging (ELK stack)
- [ ] Prepare app store submission materials
- [ ] Final documentation and handover preparation

---

5. Detailed Technology Integration Guide

5.1 Local Storage Options Comparison

SQLite Integration

```
// pubspec.yaml dependencies
dependencies:
  sqflite: ^2.3.0
  path: ^1.8.3

// Database helper class
class DatabaseHelper {
  static final DatabaseHelper instance = DatabaseHelper._init();
```

```
static Database? _database;

DatabaseHelper._init();

Future<Database> get database async {
  if (_database != null) return _database!;
  _database = await _initDB('sports_talent.db');
  return _database!;
}

Future<Database> _initDB(String filePath) async {
  final dbPath = await getDatabasesPath();
  final path = join(dbPath, filePath);

  return await openDatabase(
    path,
    version: 1,
    onCreate: _createDB,
  );
}

Future _createDB(Database db, int version) async {
  await db.execute('''
    CREATE TABLE athletes (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
      age INTEGER NOT NULL,
      gender TEXT NOT NULL,
      phone TEXT UNIQUE,
      created_at TEXT NOT NULL
    )
```

```
    ''');

    await db.execute('''
      CREATE TABLE test_sessions (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        athlete_id INTEGER NOT NULL,
        test_type TEXT NOT NULL,
        video_path TEXT NOT NULL,
        analysis_result TEXT,
        score REAL,
        created_at TEXT NOT NULL,
        synced INTEGER DEFAULT 0,
        FOREIGN KEY (athlete_id) REFERENCES athletes (id)
      )
    ''');

    await db.execute('''
      CREATE TABLE sync_queue (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        table_name TEXT NOT NULL,
        record_id INTEGER NOT NULL,
        action TEXT NOT NULL,
        data TEXT NOT NULL,
        created_at TEXT NOT NULL,
        attempts INTEGER DEFAULT 0
      )
    ''');
  }
}

// Usage in data layer
```

```
class LocalAthleteDataSource {
  final DatabaseHelper _dbHelper = DatabaseHelper.instance;

  Future<int> insertAthlete(AthleteModel athlete) async {
    final db = await _dbHelper.database;
    return await db.insert('athletes', athlete.toMap());
  }

  Future<List<AthleteModel>> getAllAthletes() async {
    final db = await _dbHelper.database;
    final result = await db.query('athletes');
    return result.map((json) => AthleteModel.fromMap(json)).toList();
  }

  Future<void> markForSync(String tableName, int recordId, String action, Map<String, dynamic>
data) async {
    final db = await _dbHelper.database;
    await db.insert('sync_queue', {
      'table_name': tableName,
      'record_id': recordId,
      'action': action,
      'data': jsonEncode(data),
      'created_at': DateTime.now().toIso8601String(),
    });
  }
}
MongoDB Realm Integration (Alternative)
// pubspec.yaml dependencies
dependencies:
  realm: ^1.6.0
```

```
// Realm model definitions
import 'package:realm/realm.dart';

part 'models.g.dart';

@RealmModel()
class _Athlete {
  @PrimaryKey()
  late ObjectId id;

  late String name;
  late int age;
  late String gender;
  late String? phone;
  late DateTime createdAt;
  late bool synced = false;
}

@RealmModel()
class _TestSession {
  @PrimaryKey()
  late ObjectId id;

  late _Athlete? athlete;
  late String testType;
  late String videoPath;
  late String? analysisResult;
  late double? score;
  late DateTime createdAt;
  late bool synced = false;
}
```

```
// Realm service implementation
class RealmService {
  static Realm? _realm;
  static Realm get realm {
    _realm ??= Realm(Configuration.local([
      Athlete.schema,
      TestSession.schema,
    ]));
    return _realm!;
  }

  // CRUD operations
  static void saveAthlete(Athlete athlete) {
    realm.write(() {
      realm.add(athlete);
    });
  }

  static List<Athlete> getAllAthletes() {
    return realm.all<Athlete>().toList();
  }

  static List<TestSession> getUnsyncedSessions() {
    return realm.all<TestSession>().where((session) => !session.synced).toList();
  }

  static void markAsSynced(TestSession session) {
    realm.write(() {
      session.synced = true;
    });
```

```
 }
}
```

5.2 YOLOv8 + OpenCV Integration Details

Android Native Implementation

```
// build.gradle (Module: app)
android {
    ndkVersion "25.1.8937393"

    defaultConfig {
        ndk {
            abiFilters 'arm64-v8a', 'armeabi-v7a'
        }
    }
}

dependencies {
    implementation 'org.opencv:opencv-android:4.8.0'
    implementation 'ai.onnxruntime:onnxruntime-android:1.16.0'
}

// YOLOv8Detector.kt
import ai.onnxruntime.*
import org.opencv.android.OpenCVLoaderCallback
import org.opencv.core.*
import org.opencv.imgproc.Imgproc
import org.opencv.dnn.Dnn

class YOLOv8Detector(private val context: Context, private val modelPath: String) {
    private lateinit var ortSession: OrtSession
    private lateinit var ortEnvironment: OrtEnvironment
```

```
fun initialize(): Boolean {
    return try {
        ortEnvironment = OrtEnvironment.getEnvironment()
        val modelBytes = context.assets.open(modelPath).readBytes()
        ortSession = ortEnvironment.createSession(modelBytes)
        true
    } catch (e: Exception) {
        Log.e("YOLOv8Detector", "Failed to initialize: ${e.message}")
        false
    }
}

fun detectPose(inputMat: Mat): List<PoseKeypoint> {
    // Preprocess image
    val blob = preprocessImage(inputMat)

    // Run inference
    val inputName = ortSession.inputNames.iterator().next()
    val inputTensor = OnnxTensor.createTensor(ortEnvironment, blob)
    val results = ortSession.run(mapOf(inputName to inputTensor))

    // Post-process results
    val outputTensor = results[0].value as Array<Array<FloatArray>>
    return postprocessPose(outputTensor, inputMat.width(), inputMat.height())
}

private fun preprocessImage(inputMat: Mat): FloatArray {
    // Resize to model input size (640x640 for YOLOv8)
    val resizedMat = Mat()
    Imgproc.resize(inputMat, resizedMat, Size(640.0, 640.0))
```

```
        // Convert BGR to RGB
        val rgbMat = Mat()
        Imgproc.cvtColor(resizedMat, rgbMat, Imgproc.COLOR_BGR2RGB)

        // Normalize to [0, 1] and convert to CHW format
        val normalizedArray = FloatArray(3 * 640 * 640)
        val rgbData = ByteArray(rgbMat.total().toInt() * rgbMat.elemSize().toInt())
        rgbMat.get(0, 0, rgbData)

        for (i in rgbData.indices) {
            val pixelValue = rgbData[i].toInt() and 0xFF
            val channel = i % 3
            val pixelIndex = i / 3
            normalizedArray[channel * 640 * 640 + pixelIndex] = pixelValue / 255.0f
        }

        return normalizedArray
    }

    private fun postprocessPose(output: Array<Array<FloatArray>>, originalWidth: Int,
originalHeight: Int): List<PoseKeypoint> {
        val keypoints = mutableListOf<PoseKeypoint>()
        val scaleX = originalWidth / 640.0f
        val scaleY = originalHeight / 640.0f

        // Extract keypoints (17 points for COCO pose)
        for (i in 0 until 17) {
            val x = output[0][0][i * 3] * scaleX
            val y = output[0][0][i * 3 + 1] * scaleY
            val confidence = output[0][0][i * 3 + 2]
```

```
            if (confidence > 0.5) {  // Confidence threshold
                keypoints.add(PoseKeypoint(i, x, y, confidence))
            }
        }

        return keypoints
    }
}

// OpenCV-based analysis functions
class OpenCVAnalyzer {
    companion object {
        fun calculateJumpHeight(keypoints: List<PoseKeypoint>, groundLevel: Float): Float {
            // Find hip keypoint (keypoint index 11 in COCO format)
            val hipKeypoint = keypoints.find { it.index == 11 }
            return hipKeypoint?.let { hip ->
                maxOf(0f, groundLevel - hip.y) * PIXEL_TO_CM_RATIO
            } ?: 0f
        }

        fun countSitups(poseSequence: List<List<PoseKeypoint>>): Int {
            var repCount = 0
            var inDownPosition = false

            for (pose in poseSequence) {
                val shoulder = pose.find { it.index == 5 } // Left shoulder
                val hip = pose.find { it.index == 11 }     // Left hip
                val knee = pose.find { it.index == 13 }    // Left knee

                if (shoulder != null && hip != null && knee != null) {
                    val angle = calculateAngle(shoulder, hip, knee)
```

```
                // Detect situp phases
                if (angle < 90 && !inDownPosition) {
                    inDownPosition = true
                } else if (angle > 130 && inDownPosition) {
                    repCount++
                    inDownPosition = false
                }
            }
        }

        return repCount
    }

    private fun calculateAngle(p1: PoseKeypoint, p2: PoseKeypoint, p3: PoseKeypoint): Double {
        val v1 = Point(p1.x - p2.x, p1.y - p2.y)
        val v2 = Point(p3.x - p2.x, p3.y - p2.y)

        val dotProduct = v1.x * v2.x + v1.y * v2.y
        val magnitude1 = sqrt(v1.x * v1.x + v1.y * v1.y)
        val magnitude2 = sqrt(v2.x * v2.x + v2.y * v2.y)

        return acos(dotProduct / (magnitude1 * magnitude2)) * 180.0 / PI
    }

    const val PIXEL_TO_CM_RATIO = 0.1f // Calibration constant
  }
}
```

iOS Native Implementation (Swift)

```
// YOLOv8Detector.swift
import Foundation
```

```
import CoreML
import Vision
import OpenCV

class YOLOv8Detector {
    private var model: VNCoreMLModel?

    init(modelName: String) {
        setupModel(modelName: modelName)
    }

    private func setupModel(modelName: String) {
        guard let modelURL = Bundle.main.url(forResource: modelName, withExtension: "mlmodel")
else {
            print("Failed to find model file")
            return
        }

        do {
            let mlModel = try MLModel(contentsOf: modelURL)
            model = try VNCoreMLModel(for: mlModel)
        } catch {
            print("Failed to load Core ML model: \(error)")
        }
    }

    func detectPose(in image: UIImage, completion: @escaping ([PoseKeypoint]) -> Void) {
        guard let model = model else {
            completion([])
            return
        }
```

```
        let request = VNCoreMLRequest(model: model) { request, error in
            guard let results = request.results as? [VNRecognizedObjectObservation] else {
                completion([])
                return
            }

            let keypoints = self.extractKeypoints(from: results)
            completion(keypoints)
        }

        guard let cgImage = image.cgImage else {
            completion([])
            return
        }

        let handler = VNImageRequestHandler(cgImage: cgImage, options: [:])
        do {
            try handler.perform([request])
        } catch {
            print("Failed to perform pose detection: \(error)")
            completion([])
        }
    }

    private func extractKeypoints(from observations: [VNRecognizedObjectObservation]) ->
[PoseKeypoint] {
        // Process Core ML results to extract pose keypoints
        var keypoints: [PoseKeypoint] = []

        for observation in observations {
```

```
        // Extract keypoint coordinates and confidence scores
        // Implementation depends on your specific model output format
      }

      return keypoints
    }
}
```

5.3 Database Schema Designs

MongoDB Schema (Mongoose)

```
// models/mongodb/User.js
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  _id: {
    type: mongoose.Schema.Types.ObjectId,
    auto: true
  },
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    unique: true,
    lowercase: true
  },
  phone: {
    type: String,
    required: true,
```

```
  unique: true
},
dateOfBirth: {
  type: Date,
  required: true
},
gender: {
  type: String,
  enum: ['male', 'female', 'other'],
  required: true
},
location: {
  state: String,
  district: String,
  pincode: String,
  coordinates: {
    type: [Number], // [longitude, latitude]
    index: '2dsphere'
  }
},
profile: {
  avatar: String,
  bio: String,
  height: Number, // in cm
  weight: Number, // in kg
  sports: [String],
  experience: String
},
achievements: [{
  badgeId: {
    type: mongoose.Schema.Types.ObjectId,
```

```
    ref: 'Achievement'
  },
  unlockedAt: {
    type: Date,
    default: Date.now
  }
}],
stats: {
  totalTests: { type: Number, default: 0 },
  bestScores: {
    verticalJump: Number,
    situps: Number,
    shuttleRun: Number,
    enduranceRun: Number
  },
  overallRank: Number
},
isVerified: {
  type: Boolean,
  default: false
},
verificationToken: String,
passwordHash: String,
refreshTokens: [String],
lastLogin: Date,
deviceInfo: [{
  deviceId: String,
  platform: String,
  appVersion: String,
  lastSeen: Date
}]
```

```
}, {
  timestamps: true,
  collection: 'users'
});

// Indexes for performance
userSchema.index({ email: 1 });
userSchema.index({ phone: 1 });
userSchema.index({ 'location.coordinates': '2dsphere' });
userSchema.index({ 'stats.overallRank': 1 });
userSchema.index({ createdAt: -1 });

module.exports = mongoose.model('User', userSchema);

// models/mongodb/TestSession.js
const testSessionSchema = new mongoose.Schema({
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  testType: {
    type: String,
    enum: ['vertical_jump', 'situps', 'shuttle_run', 'endurance_run', 'flexibility'],
    required: true
  },
  videoSubmission: {
    originalPath: String,
    processedPath: String,
    thumbnailPath: String,
    duration: Number,
```

```
  fileSize: Number,
  resolution: String,
  uploadedAt: Date,
  storageProvider: {
    type: String,
    enum: ['aws_s3', 'gcp_storage', 'local'],
    default: 'aws_s3'
  },
  metadata: {
    fps: Number,
    codec: String,
    bitrate: Number
  }
},
aiAnalysis: {
  status: {
    type: String,
    enum: ['pending', 'processing', 'completed', 'failed'],
    default: 'pending'
  },
  processedAt: Date,
  modelVersion: String,
  results: {
    primaryMetric: Number, // Main score (jump height, reps, time, etc.)
    secondaryMetrics: {
      form: Number,
      consistency: Number,
      technique: Number
    },
    keypoints: [[Number]], // Pose keypoints for each frame
    motionData: {
```

```
      speed: [Number],
      acceleration: [Number],
      trajectory: [[Number]]
    },
    confidence: Number,
    flags: [String] // Any issues detected
  },
  cheatDetection: {
    authenticity: Number,
    manipulationFlags: [String],
    environmentalConsistency: Number,
    overallTrust: Number
  },
  processingTime: Number // in milliseconds
},
manualReview: {
  required: Boolean,
  reviewedBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Admin'
  },
  reviewedAt: Date,
  comments: String,
  adjustedScore: Number,
  approved: Boolean
},
benchmarkComparison: {
  ageGroup: String,
  genderGroup: String,
  percentile: Number,
  nationalAverage: Number,
```

```
    regionalAverage: Number,
    improvement: Number // compared to previous attempts
  },
  sessionMetadata: {
    deviceInfo: {
      model: String,
      os: String,
      appVersion: String
    },
    location: {
      coordinates: [Number],
      address: String
    },
    weather: {
      temperature: Number,
      humidity: Number,
      conditions: String
    },
    testConditions: {
      indoor: Boolean,
      lighting: String,
      surface: String
    }
  }
}, {
  timestamps: true,
  collection: 'test_sessions'
});

// Compound indexes for common queries
testSessionSchema.index({ userId: 1, testType: 1, createdAt: -1 });
```

```
testSessionSchema.index({ 'aiAnalysis.status': 1, createdAt: 1 });
testSessionSchema.index({ 'benchmarkComparison.percentile': -1 });

module.exports = mongoose.model('TestSession', testSessionSchema);
```

MySQL Schema (Alternative)

```
-- MySQL database schema
CREATE DATABASE sports_talent_assessment;
USE sports_talent_assessment;

-- Users table
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    phone VARCHAR(15) NOT NULL UNIQUE,
    date_of_birth DATE NOT NULL,
    gender ENUM('male', 'female', 'other') NOT NULL,

    -- Location data
    state VARCHAR(100),
    district VARCHAR(100),
    pincode VARCHAR(10),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),

    -- Profile information
    avatar_url VARCHAR(500),
    bio TEXT,
    height_cm INT,
    weight_kg DECIMAL(5,2),
    sports JSON,
```

```
    experience ENUM('beginner', 'intermediate', 'advanced'),

    -- Authentication
    password_hash VARCHAR(255) NOT NULL,
    email_verified BOOLEAN DEFAULT FALSE,
    phone_verified BOOLEAN DEFAULT FALSE,
    verification_token VARCHAR(100),

    -- Timestamps
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    last_login TIMESTAMP,

    -- Indexes
    INDEX idx_email (email),
    INDEX idx_phone (phone),
    INDEX idx_### 3.2 Mobile Optimization for YOLOv8 + OpenCV

#### ONNX Model Conversion Strategy
```python
# YOLOv8 to ONNX conversion for mobile deployment
from ultralytics import YOLO
import torch

# Convert YOLOv8 to ONNX
model = YOLO('yolov8n-pose.pt')
model.export(format='onnx', dynamic=True, simplify=True)

# Optimization parameters:
# - Dynamic input shapes for flexible inference
# - Graph simplification for reduced model size
```
```

```
# - FP16 precision for 2x speed improvement
# - Size reduction: 6MB → 3MB (with quantization)
# - Inference speed: 40+ FPS on modern mobile devices
```

Quantization and Pruning Strategy

```
# Post-training quantization for mobile deployment
import onnxruntime as ort
from onnxruntime.quantization import quantize_dynamic, QuantType

def optimize_for_mobile(model_path):
    # Dynamic quantization
    quantized_model = quantize_dynamic(
        model_path,
        model_path.replace('.onnx', '_quantized.onnx'),
        weight_type=QuantType.QUInt8
    )

    # Performance improvements:
    # - Model size: 4x smaller
    # - Inference speed: 2-3# AI-Powered Sports Talent Assessment App
```

Complete Development Roadmap & Architecture

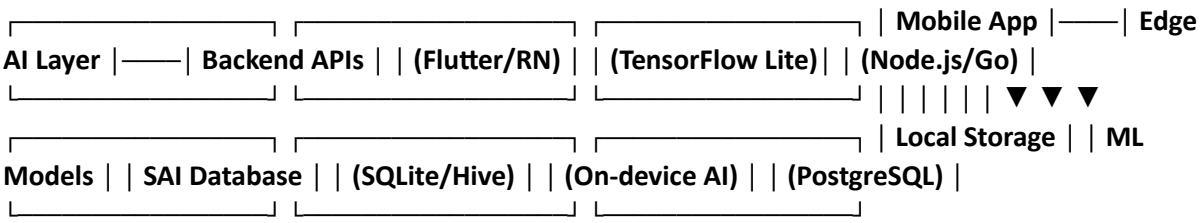
Executive Summary

This document outlines a comprehensive 4-week development plan for creating an AI-powered mobile platform that democratizes sports talent assessment across India, particularly targeting rural and remote areas.

---

1. Project Architecture Overview

1.1 System Architecture



1.2 Technology Stack Comparison

Frontend Mobile App

| Technology | Pros | Cons | Recommendation |

|---|---|---|---|

| **Flutter** | ✅ Single codebase, excellent performance, great camera plugins | ❌ Larger app size | **RECOMMENDED** |

| React Native | ✅ JavaScript familiarity, good community | ❌ Bridge overhead, camera issues | Alternative |

| Native (Kotlin/Swift) | ✅ Best performance, platform-specific features | ❌ Dual development | For production scale |

Backend Services

| Technology | Use Case | Pros | Cons |

|---|---|---|---|

| **Node.js + Express** | API Gateway, Real-time features | ✅ JavaScript ecosystem, fast development | ❌ Single-threaded |

| **Go + Gin** | High-performance APIs | ✅ Excellent concurrency, fast | ❌ Learning curve |

| Python + FastAPI | ML integration | ✅ AI/ML libraries, rapid prototyping | ❌ Performance limitations |

Database Solutions

| Database | Use Case | Justification |

|---|---|---|

| **PostgreSQL** | Primary database | ACID compliance, JSON support, scalability |

| **Redis** | Caching, sessions | High-performance in-memory storage |

| **SQLite** | Local mobile storage | Offline capability, lightweight |

AI/ML Framework

| Framework | Platform | Best For |

|-----------|----------|----------|

| **TensorFlow Lite** | Mobile | On-device inference, model optimization |

| **Core ML** | iOS only | iOS-specific optimizations |

| **ONNX Runtime** | Cross-platform | Model interoperability |

---

2. Detailed File Structure & Architecture

2.1 Mobile App Structure (Flutter)

sports_talent_app/ ├── lib/ │ ├── main.dart # App entry point │ ├── app/ │ │ ├── app.dart # App configuration │ │ ├── routes.dart # Navigation routes │ │ └── themes.dart # UI themes │ ├── core/ │ │ ├── constants/ │ │ │ ├── api_constants.dart # API endpoints │ │ │ ├── app_constants.dart # App-wide constants │ │ │ └── test_constants.dart # Sports test parameters │ │ ├── errors/ │ │ │ ├── exceptions.dart # Custom exceptions │ │ │ └── failures.dart # Error handling │ │ ├── network/ │ │ │ ├── api_client.dart # HTTP client wrapper │ │ │ ├── network_info.dart # Connectivity checker │ │ │ └── interceptors.dart # Request/response interceptors │ │ └── utils/ │ │ ├── validators.dart # Input validation │ │ ├── formatters.dart # Data formatting │ │ └── permissions.dart # Device permissions │ ├── features/ │ │ ├── authentication/ │ │ │ ├── data/ │ │ │ │ ├── datasources/ │ │ │ │ │ ├── auth_local_datasource.dart │ │ │ │ │ └── auth_remote_datasource.dart │ │ │ │ ├── models/ │ │ │ │ │ └── user_model.dart │ │ │ │ └── repositories/ │ │ │ │ └── auth_repository_impl.dart │ │ │ ├── domain/ │ │ │ │ ├── entities/ │ │ │ │ │ └── user_entity.dart │ │ │ │ ├── repositories/ │ │ │ │ │ └── auth_repository.dart │ │ │ │ └── usecases/ │ │ │ │ ├── login_usecase.dart │ │ │ │ ├── register_usecase.dart │ │ │ │ └── logout_usecase.dart │ │ │ └── presentation/ │ │ │ ├── bloc/ │ │ │ │ ├── auth_bloc.dart │ │ │ │ ├── auth_event.dart │ │ │ │ └── auth_state.dart │ │ │ ├── pages/ │ │ │ │ ├── login_page.dart │ │ │ │ ├── register_page.dart │ │ │ │ └── profile_page.dart │ │ │ └── widgets/ │ │ │ ├── auth_form.dart │ │ │ └── social_login_buttons.dart │ │ ├── video_recording/ │ │ │ ├── data/ │ │ │ │ ├── datasources/ │ │ │ │ │ ├── camera_datasource.dart │ │ │ │ │ └── video_storage_datasource.dart │ │ │ │ ├── models/ │ │ │ │ │ ├── video_model.dart │ │ │ │ │ └── recording_session_model.dart │ │ │ │ └── repositories/ │ │ │ │ └── video_repository_impl.dart │ │ │ ├── domain/ │ │ │ │ ├── entities/ │ │ │ │ │ ├── video_entity.dart │ │ │ │ │ └── test_session_entity.dart │ │ │ │ ├── repositories/ │ │ │ │ │ └── video_repository.dart │ │ │ │ └── usecases/ │ │ │ │ ├── start_recording_usecase.dart │ │ │ │ ├── stop_recording_usecase.dart │ │ │ │ └── save_video_usecase.dart │ │ │ └── presentation/ │ │ │ ├── bloc/ │ │ │ │ ├── video_recording_bloc.dart │ │ │ │ ├── video_recording_event.dart │ │ │ │ └──

video_recording_state.dart | | | ├── pages/ | | | | ├── test_selection_page.dart | | | | ├── camera_page.dart | | | | └── recording_review_page.dart | | | └── widgets/ | | | ├── camera_overlay.dart | | | ├── test_timer.dart | | | ├── countdown_widget.dart | | | └── recording_controls.dart | | ├── ai_analysis/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── ml_model_datasource.dart | | | | | └── analysis_cache_datasource.dart | | | | ├── models/ | | | | | ├── analysis_result_model.dart | | | | | └── performance_metrics_model.dart | | | | └── repositories/ | | | | └── ai_analysis_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── analysis_result_entity.dart | | | | | └── performance_data_entity.dart | | | | ├── repositories/ | | | | | └── ai_analysis_repository.dart | | | | └── usecases/ | | | | ├── analyze_vertical_jump_usecase.dart | | | | ├── count_situps_usecase.dart | | | | ├── analyze_shuttle_run_usecase.dart | | | | ├── detect_cheating_usecase.dart | | | | └── benchmark_performance_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── ai_analysis_bloc.dart | | | | ├── ai_analysis_event.dart | | | | └── ai_analysis_state.dart | | | ├── pages/ | | | | ├── analysis_loading_page.dart | | | | ├── results_page.dart | | | | └── performance_dashboard_page.dart | | | └── widgets/ | | | ├── analysis_progress.dart | | | ├── performance_chart.dart | | | ├── benchmark_comparison.dart | | | └── cheat_detection_indicator.dart | | ├── gamification/ | | | ├── data/ | | | | ├── datasources/ | | | | | ├── achievements_datasource.dart | | | | | └── leaderboard_datasource.dart | | | | ├── models/ | | | | | ├── badge_model.dart | | | | | ├── achievement_model.dart | | | | | └── leaderboard_model.dart | | | | └── repositories/ | | | | └── gamification_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── badge_entity.dart | | | | | └── user_progress_entity.dart | | | | ├── repositories/ | | | | | └── gamification_repository.dart | | | | └── usecases/ | | | | ├── unlock_achievement_usecase.dart | | | | ├── get_leaderboard_usecase.dart | | | | └── update_progress_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── gamification_bloc.dart | | | | ├── gamification_event.dart | | | | └── gamification_state.dart | | | ├── pages/ | | | | ├── achievements_page.dart | | | | ├── leaderboard_page.dart | | | | └── progress_page.dart | | | └── widgets/ | | | ├── badge_widget.dart | | | ├── progress_bar.dart | | | ├── leaderboard_item.dart | | | └── achievement_popup.dart | | └── data_sync/ | | ├── data/ | | | ├── datasources/ | | | | ├── sync_local_datasource.dart | | | | └── sync_remote_datasource.dart | | | ├── models/ | | | | ├── sync_queue_model.dart | | | | └── upload_status_model.dart | | | └── repositories/ | | | └── sync_repository_impl.dart | | ├── domain/ | | | ├── entities/ | | | | └── sync_item_entity.dart | | | ├── repositories/ | | | | └── sync_repository.dart | | | └── usecases/ | | | ├── queue_for_sync_usecase.dart | | | ├── sync_data_usecase.dart | | | └── retry_failed_sync_usecase.dart | | └── presentation/ | | ├── bloc/ | | | ├── sync_bloc.dart | | | ├── sync_event.dart | | | └── sync_state.dart | | └── widgets/ | | ├── sync_status_indicator.dart | | └── upload_progress.dart | ├── shared/ | | ├── widgets/ | | | ├── custom_button.dart | | | ├── loading_indicator.dart | | | ├── error_widget.dart | | | ├── video_player.dart | | | └── performance_card.dart | | ├── services/ | | | ├── dependency_injection.dart # Service locator | | | ├── local_storage_service.dart # SQLite/Hive wrapper | | | ├── notification_service.dart # Push notifications | | | ├── biometric_service.dart # Fingerprint/face recognition | | | └── ml_service.dart # TensorFlow Lite wrapper | | └── extensions/ | | ├── string_extensions.dart | | ├── datetime_extensions.dart | | └── number_extensions.dart | └── models/ | ├── ai_models/ # TensorFlow Lite models | | ├── vertical_jump_detector.tflite | | ├── situp_counter.tflite | | ├── shuttle_run_analyzer.tflite | |

├── pose_estimation.tflite │ │ └── cheat_detection.tflite │ └── benchmark_data/ │ ├── age_gender_benchmarks.json │ └── regional_benchmarks.json ├── assets/ │ ├── images/ │ │ ├── logos/ │ │ ├── icons/ │ │ ├── badges/ │ │ └── tutorials/ │ ├── videos/ │ │ └── tutorial_videos/ │ ├── animations/ │ │ └── lottie_files/ │ └── fonts/ ├── test/ │ ├── unit/ │ ├── widget/ │ └── integration/ ├── pubspec.yaml # Dependencies └── README.md

2.2 Backend API Structure (Node.js)

sports-talent-backend/ ├── src/ │ ├── app.js # Express app setup │ ├── server.js # Server entry point │ ├── config/ │ │ ├── database.js # DB configuration │ │ ├── redis.js # Redis configuration │ │ ├── aws.js # AWS S3 configuration │ │ ├── jwt.js # JWT configuration │ │ └── multer.js # File upload configuration │ ├── controllers/ │ │ ├── authController.js # Authentication endpoints │ │ ├── userController.js # User management │ │ ├── videoController.js # Video upload/processing │ │ ├── analysisController.js # AI analysis results │ │ ├── benchmarkController.js # Performance benchmarks │ │ ├── leaderboardController.js # Gamification features │ │ └── adminController.js # SAI dashboard │ ├── middleware/ │ │ ├── auth.js # JWT verification │ │ ├── validation.js # Request validation │ │ ├── rateLimiting.js # API rate limiting │ │ ├── errorHandler.js # Global error handling │ │ └── fileUpload.js # File processing │ ├── models/ │ │ ├── User.js # User schema │ │ ├── Athlete.js # Athlete profile │ │ ├── TestSession.js # Test session data │ │ ├── VideoSubmission.js # Video metadata │ │ ├── AnalysisResult.js # AI analysis results │ │ ├── Achievement.js # Gamification badges │ │ └── Benchmark.js # Performance standards │ ├── routes/ │ │ ├── auth.js # Authentication routes │ │ ├── athletes.js # Athlete management │ │ ├── tests.js # Sports test routes │ │ ├── videos.js # Video handling │ │ ├── analysis.js # AI analysis │ │ ├── gamification.js # Badges and achievements │ │ └── admin.js # SAI dashboard routes │ ├── services/ │ │ ├── authService.js # Authentication logic │ │ ├── videoProcessingService.js # Video processing pipeline │ │ ├── aiAnalysisService.js # AI model integration │ │ ├── benchmarkService.js # Performance comparison │ │ ├── notificationService.js # Push notifications │ │ ├── storageService.js # File storage (AWS S3) │ │ ├── encryptionService.js # Data encryption │ │ └── auditService.js # Activity logging │ ├── utils/ │ │ ├── validators.js # Data validation │ │ ├── helpers.js # Utility functions │ │ ├── constants.js # Application constants │ │ ├── logger.js # Logging utility │ │ └── encryption.js # Security utilities │ ├── jobs/ │ │ ├── videoProcessingJob.js # Background video processing │ │ ├── aiAnalysisJob.js # Batch AI analysis │ │ ├── dataCleanupJob.js # Database maintenance │ │ └── reportGenerationJob.js # Periodic reports │ └── database/ │ ├── migrations/ │ │ ├── 001_create_users_table.js │ │ ├── 002_create_athletes_table.js │ │ ├── 003_create_test_sessions_table.js │ │ ├── 004_create_video_submissions_table.js │ │ ├── 005_create_analysis_results_table.js │ │ └── 006_create_benchmarks_table.js │ ├── seeds/ │ │ ├── benchmark_data.js │ │ └── sample_users.js │ └── config.js ├── ai_models/ │ ├── preprocessing/ │ │ ├── video_preprocessor.py # Video frame extraction │ │ ├── pose_estimator.py # Body pose detection │ │ └── frame_analyzer.py # Individual frame analysis │ ├── models/ │ │ ├── vertical_jump/ │ │ │ ├── jump_detector.py # Jump height calculation │ │ │ ├── model.h5 # Trained model │ │ │ └── preprocessing.py # Jump-specific preprocessing │ │ ├── situp_counter/ │ │ │ ├── rep_counter.py # Repetition counting │ │ │ ├── model.h5 │ │ │ └── form_analyzer.py # Exercise form validation │ │ ├── shuttle_run/ │ │ │ ├── speed_analyzer.py # Speed and agility analysis │ │ │ ├── model.h5 │ │ │ └── distance_calculator.py # Distance measurement │ │ └── cheat_detection/ │ │ ├──

anomaly_detector.py # Unusual pattern detection | | ├── model.h5 | | └── authenticity_checker.py # Video manipulation detection | ├── training/ | | ├── data_preparation.py # Dataset preparation | | ├── model_training.py # Training pipeline | | ├── evaluation.py # Model evaluation | | └── optimization.py # Model optimization for mobile | └── inference/ | ├── batch_inference.py # Server-side batch processing | ├── real_time_inference.py # Real-time analysis | └── model_server.py # Model serving API ├── scripts/ | ├── setup_database.js # Database initialization | ├── migrate.js # Migration runner | ├── seed.js # Data seeding | ├── backup.js # Database backup | └── deploy.js # Deployment script ├── tests/ | ├── unit/ | ├── integration/ | └── e2e/ ├── docs/ | ├── api_documentation.md | ├── deployment_guide.md | └── troubleshooting.md ├── docker/ | ├── Dockerfile # Application container | ├── docker-compose.yml # Multi-service setup | └── nginx.conf # Load balancer configuration ├── .env.example # Environment variables template ├── package.json # Dependencies └── README.md

2.3 SAI Admin Dashboard Structure (React)

sai-admin-dashboard/ ├── public/ ├── src/ | ├── components/ | | ├── common/ | | | ├── Header.jsx | | | ├── Sidebar.jsx | | | ├── Footer.jsx | | | └── LoadingSpinner.jsx | | ├── charts/ | | | ├── PerformanceChart.jsx | | | ├── RegionalDistribution.jsx | | | └── TalentPipeline.jsx | | ├── athletes/ | | | ├── AthleteProfile.jsx | | | ├── AthleteList.jsx | | | ├── PerformanceAnalysis.jsx | | | └── VideoReview.jsx | | └── admin/ | | ├── UserManagement.jsx | | ├── SystemHealth.jsx | | └── Reports.jsx | ├── pages/ | | ├── Dashboard.jsx | | ├── Athletes.jsx | | ├── Analytics.jsx | | ├── Settings.jsx | | └── Reports.jsx | ├── services/ | | ├── api.js | | ├── auth.js | | └── websocket.js | ├── utils/ | | ├── constants.js | | ├── helpers.js | | └── validators.js | ├── styles/ | | ├── globals.css | | └── components/ | ├── App.jsx | └── index.js ├── package.json └── README.md

---

3. AI/ML Models Architecture

3.1 Model Specifications

Vertical Jump Detection Model

```python

# Model Architecture

Input: Video frames (224x224x3)

├── CNN Feature Extractor (MobileNetV2 backbone)
```

├── Temporal Analysis Layer (LSTM)

├── Jump Detection Head

├── Height Calculation Module

└── Output: Jump height in cm, confidence score

 # Key Features:

- Real-time pose estimation

- Ground plane detection

- Jump trajectory analysis

- Height measurement accuracy: ±2cm

Sit-up Counter Model

 # Model Architecture

Input: Video frames (224x224x3)

├── Pose Estimation (MediaPipe)

├── Hip/Shoulder Angle Calculator

├── Rep Detection Algorithm

├── Form Validation Module

└── Output: Rep count, form score (0-100)

 # Key Features:

- Angle-based counting

- Form quality assessment

- False positive reduction

- Accuracy: 95%+ in controlled conditions

Shuttle Run Analyzer

 # Model Architecture

Input: Video frames (224x224x3)

├── Object Detection (Person tracking)

├── Speed Estimation Module

├── Distance Calculation

├── Agility Metrics Calculator

└── Output: Time, speed, agility score

 # Key Features:

- Multi-person tracking

- Speed profile analysis

- Direction change detection

- Timing accuracy: ±0.1 seconds

Cheat Detection Model

 # Model Architecture

Input: Video frames + metadata

├── Video Authenticity Checker

├── Motion Pattern Analyzer

├── Environmental Consistency Checker

├── Anomaly Detection Module

└── Output: Authenticity score (0-1), flags

 # Detection Capabilities:

- Video manipulation (deepfake, editing)

- Unrealistic motion patterns

- Environmental inconsistencies

- Equipment tampering

3.2 Model Optimization for Mobile

Quantization Strategy

 # Post-training quantization

model_int8 = tf.lite.TFLiteConverter.from_keras_model(model)

model_int8.optimizations = [tf.lite.Optimize.DEFAULT]

model_int8.target_spec.supported_types = [tf.lite.constants.INT8]

 # Size reduction: 4x smaller

 # Inference speed: 2-3x faster

 # Accuracy loss: <5%

Model Pruning

 # Structured pruning for mobile deployment

import tensorflow_model_optimization as tfmot

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

pruned_model = prune_low_magnitude(model,

pruning_schedule=tfmot.sparsity.keras.PolynomialDecay(

initial_sparsity=0.50,

final_sparsity=0.90,

begin_step=0,

end_step=1000))

---

4. 4-Week Development Timeline

Week 1: Foundation & Core Setup

Days 1-2: Project Setup & Architecture

- **[ ] Set up development environment**
- **[ ] Initialize Flutter project with clean architecture**
- **[ ] Set up backend Node.js project structure**
- **[ ] Configure database schemas (PostgreSQL)**
- **[ ] Set up Redis for caching**
- **[ ] Configure CI/CD pipeline (GitHub Actions)**

Days 3-4: Authentication & User Management

- **[ ] Implement user registration/login (mobile + backend)**
- **[ ] JWT token implementation**
- **[ ] Basic user profile management**
- **[ ] Database user tables and relationships**
- **[ ] Input validation and error handling**

Days 5-7: Video Recording Infrastructure

- **[ ] Camera integration (Flutter)**
- **[ ] Video recording functionality**

- **[ ] Local video storage**
- **[ ] Basic video player**
- **[ ] File upload preparation**
- **[ ] Video compression implementation**

Week 2: AI/ML Integration & Core Features

Days 8-10: AI Models Development

- **[ ] Set up TensorFlow/PyTorch training environment**
- **[ ] Collect and prepare training datasets**
- **[ ] Train vertical jump detection model**
- **[ ] Train sit-up counter model**
- **[ ] Train shuttle run analyzer**
- **[ ] Convert models to TensorFlow Lite**

Days 11-12: Mobile AI Integration

- **[ ] Integrate TensorFlow Lite into Flutter**
- **[ ] Implement on-device inference pipeline**
- **[ ] Pose estimation integration (MediaPipe)**
- **[ ] Real-time video analysis**
- **[ ] Performance optimization for low-end devices**

Days 13-14: Sports Test Implementation

- **[ ] Implement test selection UI**
- **[ ] Create test-specific recording interfaces**
- **[ ] Add countdown timers and instructions**
- **[ ] Implement basic analysis results display**
- **[ ] Test validation and error handling**

Week 3: Advanced Features & Backend

Days 15-16: Cheat Detection & Validation

- **[ ] Implement cheat detection algorithms**
- **[ ] Video authenticity verification**
- **[ ] Motion pattern analysis**
- **[ ] Environmental consistency checks**
- **[ ] Integration with main analysis pipeline**

Days 17-18: Backend API Development

- **[ ] Complete REST API endpoints**
- **[ ] Video upload and processing pipeline**
- **[ ] Background job processing (Bull Queue)**
- **[ ] Data synchronization logic**
- **[ ] Performance benchmarking system**

Days 19-21: Gamification & User Experience

- **[ ] Achievement system implementation**
- **[ ] Badge unlocking mechanism**
- **[ ] Leaderboard functionality**
- **[ ] Progress tracking**
- **[ ] Push notifications setup**
- **[ ] UI/UX polishing**

Week 4: Integration, Testing & Deployment

Days 22-24: SAI Admin Dashboard

- **[ ] React admin dashboard setup**
- **[ ] Athlete data visualization**
- **[ ] Performance analytics charts**
- **[ ] Video review interface**
- **[ ] Report generation features**
- **[ ] Real-time data updates (WebSocket)**

Days 25-26: Comprehensive Testing & Quality Assurance

- **[ ] Unit testing for YOLOv8 + OpenCV integration**
- **[ ] Integration testing for MongoDB/MySQL data flows**
- **[ ] Performance testing on various mobile devices**
- **[ ] AI model accuracy validation with test datasets**
- **[ ] Video processing pipeline stress testing**
- **[ ] Offline functionality testing**
- **[ ] Security testing for data transmission**
- **[ ] User acceptance testing with beta users**

Days 27-28: Deployment & Production Setup

- [ ] Set up production MongoDB cluster / MySQL with replication
- [ ] Configure AWS S3 for video storage with CDN
- [ ] Deploy backend APIs with load balancing
- [ ] Set up Redis cluster for caching and sessions
- [ ] Configure CI/CD pipeline for automated deployments
- [ ] Set up monitoring and logging (ELK stack)
- [ ] Prepare app store submission materials
- [ ] Final documentation and handover preparation

---

5. Detailed Technology Integration Guide

5.1 Local Storage Options Comparison

SQLite Integration

```
// pubspec.yaml dependencies
dependencies:
  sqflite: ^2.3.0
  path: ^1.8.3

// Database helper class
class DatabaseHelper {
  static final DatabaseHelper instance = DatabaseHelper._init();
  static Database? _database;

  DatabaseHelper._init();

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDB('sports_talent.db');
    return _database!;
  }

  Future<Database> _initDB(String filePath) async {
```

```
  final dbPath = await getDatabasesPath();
  final path = join(dbPath, filePath);

  return await openDatabase(
    path,
    version: 1,
    onCreate: _createDB,
  );
}

Future _createDB(Database db, int version) async {
  await db.execute('''
    CREATE TABLE athletes (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      name TEXT NOT NULL,
      age INTEGER NOT NULL,
      gender TEXT NOT NULL,
      phone TEXT UNIQUE,
      created_at TEXT NOT NULL
    )
  ''');

  await db.execute('''
    CREATE TABLE test_sessions (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      athlete_id INTEGER NOT NULL,
      test_type TEXT NOT NULL,
      video_path TEXT NOT NULL,
      analysis_result TEXT,
      score REAL,
      created_at TEXT NOT NULL,
```

```
      synced INTEGER DEFAULT 0,
      FOREIGN KEY (athlete_id) REFERENCES athletes (id)
    )
  ''');

  await db.execute('''
    CREATE TABLE sync_queue (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      table_name TEXT NOT NULL,
      record_id INTEGER NOT NULL,
      action TEXT NOT NULL,
      data TEXT NOT NULL,
      created_at TEXT NOT NULL,
      attempts INTEGER DEFAULT 0
    )
  ''');
 }
}

// Usage in data layer
class LocalAthleteDataSource {
 final DatabaseHelper _dbHelper = DatabaseHelper.instance;

 Future<int> insertAthlete(AthleteModel athlete) async {
  final db = await _dbHelper.database;
  return await db.insert('athletes', athlete.toMap());
 }

 Future<List<AthleteModel>> getAllAthletes() async {
  final db = await _dbHelper.database;
  final result = await db.query('athletes');
```

```
    return result.map((json) => AthleteModel.fromMap(json)).toList();
  }

  Future<void> markForSync(String tableName, int recordId, String action, Map<String, dynamic>
data) async {
    final db = await _dbHelper.database;
    await db.insert('sync_queue', {
      'table_name': tableName,
      'record_id': recordId,
      'action': action,
      'data': jsonEncode(data),
      'created_at': DateTime.now().toIso8601String(),
    });
  }
}
MongoDB Realm Integration (Alternative)
// pubspec.yaml dependencies
dependencies:
  realm: ^1.6.0

// Realm model definitions
import 'package:realm/realm.dart';

part 'models.g.dart';

@RealmModel()
class _Athlete {
  @PrimaryKey()
  late ObjectId id;

  late String name;
```

```
  late int age;
  late String gender;
  late String? phone;
  late DateTime createdAt;
  late bool synced = false;
}

@RealmModel()
class _TestSession {
  @PrimaryKey()
  late ObjectId id;

  late _Athlete? athlete;
  late String testType;
  late String videoPath;
  late String? analysisResult;
  late double? score;
  late DateTime createdAt;
  late bool synced = false;
}

// Realm service implementation
class RealmService {
  static Realm? _realm;
  static Realm get realm {
    _realm ??= Realm(Configuration.local([
      Athlete.schema,
      TestSession.schema,
    ]));
    return _realm!;
  }
```

```
  // CRUD operations
  static void saveAthlete(Athlete athlete) {
    realm.write(() {
      realm.add(athlete);
    });
  }

  static List<Athlete> getAllAthletes() {
    return realm.all<Athlete>().toList();
  }

  static List<TestSession> getUnsyncedSessions() {
    return realm.all<TestSession>().where((session) => !session.synced).toList();
  }

  static void markAsSynced(TestSession session) {
    realm.write(() {
      session.synced = true;
    });
  }
}
```

5.2 YOLOv8 + OpenCV Integration Details

Android Native Implementation

```
// build.gradle (Module: app)
android {
    ndkVersion "25.1.8937393"

    defaultConfig {
        ndk {
            abiFilters 'arm64-v8a', 'armeabi-v7a'
```

```
        }
    }
}

dependencies {
    implementation 'org.opencv:opencv-android:4.8.0'
    implementation 'ai.onnxruntime:onnxruntime-android:1.16.0'
}

// YOLOv8Detector.kt
import ai.onnxruntime.*
import org.opencv.android.OpenCVLoaderCallback
import org.opencv.core.*
import org.opencv.imgproc.Imgproc
import org.opencv.dnn.Dnn

class YOLOv8Detector(private val context: Context, private val modelPath: String) {
    private lateinit var ortSession: OrtSession
    private lateinit var ortEnvironment: OrtEnvironment

    fun initialize(): Boolean {
        return try {
            ortEnvironment = OrtEnvironment.getEnvironment()
            val modelBytes = context.assets.open(modelPath).readBytes()
            ortSession = ortEnvironment.createSession(modelBytes)
            true
        } catch (e: Exception) {
            Log.e("YOLOv8Detector", "Failed to initialize: ${e.message}")
            false
        }
    }
```

```
fun detectPose(inputMat: Mat): List<PoseKeypoint> {
    // Preprocess image
    val blob = preprocessImage(inputMat)

    // Run inference
    val inputName = ortSession.inputNames.iterator().next()
    val inputTensor = OnnxTensor.createTensor(ortEnvironment, blob)
    val results = ortSession.run(mapOf(inputName to inputTensor))

    // Post-process results
    val outputTensor = results[0].value as Array<Array<FloatArray>>
    return postprocessPose(outputTensor, inputMat.width(), inputMat.height())
}

private fun preprocessImage(inputMat: Mat): FloatArray {
    // Resize to model input size (640x640 for YOLOv8)
    val resizedMat = Mat()
    Imgproc.resize(inputMat, resizedMat, Size(640.0, 640.0))

    // Convert BGR to RGB
    val rgbMat = Mat()
    Imgproc.cvtColor(resizedMat, rgbMat, Imgproc.COLOR_BGR2RGB)

    // Normalize to [0, 1] and convert to CHW format
    val normalizedArray = FloatArray(3 * 640 * 640)
    val rgbData = ByteArray(rgbMat.total().toInt() * rgbMat.elemSize().toInt())
    rgbMat.get(0, 0, rgbData)

    for (i in rgbData.indices) {
        val pixelValue = rgbData[i].toInt() and 0xFF
```

```
            val channel = i % 3
            val pixelIndex = i / 3
            normalizedArray[channel * 640 * 640 + pixelIndex] = pixelValue / 255.0f
        }

        return normalizedArray
    }

    private fun postprocessPose(output: Array<Array<FloatArray>>, originalWidth: Int,
originalHeight: Int): List<PoseKeypoint> {
        val keypoints = mutableListOf<PoseKeypoint>()
        val scaleX = originalWidth / 640.0f
        val scaleY = originalHeight / 640.0f

        // Extract keypoints (17 points for COCO pose)
        for (i in 0 until 17) {
            val x = output[0][0][i * 3] * scaleX
            val y = output[0][0][i * 3 + 1] * scaleY
            val confidence = output[0][0][i * 3 + 2]

            if (confidence > 0.5) {  // Confidence threshold
                keypoints.add(PoseKeypoint(i, x, y, confidence))
            }
        }

        return keypoints
    }
}

// OpenCV-based analysis functions
class OpenCVAnalyzer {
```

```
companion object {
    fun calculateJumpHeight(keypoints: List<PoseKeypoint>, groundLevel: Float): Float {
        // Find hip keypoint (keypoint index 11 in COCO format)
        val hipKeypoint = keypoints.find { it.index == 11 }
        return hipKeypoint?.let { hip ->
            maxOf(0f, groundLevel - hip.y) * PIXEL_TO_CM_RATIO
        } ?: 0f
    }

    fun countSitups(poseSequence: List<List<PoseKeypoint>>): Int {
        var repCount = 0
        var inDownPosition = false

        for (pose in poseSequence) {
            val shoulder = pose.find { it.index == 5 } // Left shoulder
            val hip = pose.find { it.index == 11 }     // Left hip
            val knee = pose.find { it.index == 13 }    // Left knee

            if (shoulder != null && hip != null && knee != null) {
                val angle = calculateAngle(shoulder, hip, knee)

                // Detect situp phases
                if (angle < 90 && !inDownPosition) {
                    inDownPosition = true
                } else if (angle > 130 && inDownPosition) {
                    repCount++
                    inDownPosition = false
                }
            }
        }
```

```
        return repCount
    }

    private fun calculateAngle(p1: PoseKeypoint, p2: PoseKeypoint, p3: PoseKeypoint): Double {
        val v1 = Point(p1.x - p2.x, p1.y - p2.y)
        val v2 = Point(p3.x - p2.x, p3.y - p2.y)

        val dotProduct = v1.x * v2.x + v1.y * v2.y
        val magnitude1 = sqrt(v1.x * v1.x + v1.y * v1.y)
        val magnitude2 = sqrt(v2.x * v2.x + v2.y * v2.y)

        return acos(dotProduct / (magnitude1 * magnitude2)) * 180.0 / PI
    }

    const val PIXEL_TO_CM_RATIO = 0.1f // Calibration constant
  }
}
```

iOS Native Implementation (Swift)

```
// YOLOv8Detector.swift
import Foundation
import CoreML
import Vision
import OpenCV

class YOLOv8Detector {
  private var model: VNCoreMLModel?

  init(modelName: String) {
    setupModel(modelName: modelName)
  }
```

```
    private func setupModel(modelName: String) {
        guard let modelURL = Bundle.main.url(forResource: modelName, withExtension: "mlmodel")
else {
            print("Failed to find model file")
            return
        }

        do {
            let mlModel = try MLModel(contentsOf: modelURL)
            model = try VNCoreMLModel(for: mlModel)
        } catch {
            print("Failed to load Core ML model: \(error)")
        }
    }

    func detectPose(in image: UIImage, completion: @escaping ([PoseKeypoint]) -> Void) {
        guard let model = model else {
            completion([])
            return
        }

        let request = VNCoreMLRequest(model: model) { request, error in
            guard let results = request.results as? [VNRecognizedObjectObservation] else {
                completion([])
                return
            }

            let keypoints = self.extractKeypoints(from: results)
            completion(keypoints)
        }
```

```
        guard let cgImage = image.cgImage else {
            completion([])
            return
        }

        let handler = VNImageRequestHandler(cgImage: cgImage, options: [:])
        do {
            try handler.perform([request])
        } catch {
            print("Failed to perform pose detection: \(error)")
            completion([])
        }
    }

    private func extractKeypoints(from observations: [VNRecognizedObjectObservation]) ->
[PoseKeypoint] {
        // Process Core ML results to extract pose keypoints
        var keypoints: [PoseKeypoint] = []

        for observation in observations {
            // Extract keypoint coordinates and confidence scores
            // Implementation depends on your specific model output format
        }

        return keypoints
    }
}
```

5.3 Database Schema Designs

MongoDB Schema (Mongoose)

```
// models/mongodb/User.js
const mongoose = require('mongoose');
```

```
const userSchema = new mongoose.Schema({
  _id: {
    type: mongoose.Schema.Types.ObjectId,
    auto: true
  },
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    unique: true,
    lowercase: true
  },
  phone: {
    type: String,
    required: true,
    unique: true
  },
  dateOfBirth: {
    type: Date,
    required: true
  },
  gender: {
    type: String,
    enum: ['male', 'female', 'other'],
    required: true
  },
```

```
location: {
  state: String,
  district: String,
  pincode: String,
  coordinates: {
    type: [Number], // [longitude, latitude]
    index: '2dsphere'
  }
},
profile: {
  avatar: String,
  bio: String,
  height: Number, // in cm
  weight: Number, // in kg
  sports: [String],
  experience: String
},
achievements: [{
  badgeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Achievement'
  },
  unlockedAt: {
    type: Date,
    default: Date.now
  }
}],
stats: {
  totalTests: { type: Number, default: 0 },
  bestScores: {
    verticalJump: Number,
```

```
    situps: Number,
    shuttleRun: Number,
    enduranceRun: Number
   },
   overallRank: Number
  },
  isVerified: {
   type: Boolean,
   default: false
  },
  verificationToken: String,
  passwordHash: String,
  refreshTokens: [String],
  lastLogin: Date,
  deviceInfo: [{
   deviceId: String,
   platform: String,
   appVersion: String,
   lastSeen: Date
  }]
}, {
  timestamps: true,
  collection: 'users'
});

// Indexes for performance
userSchema.index({ email: 1 });
userSchema.index({ phone: 1 });
userSchema.index({ 'location.coordinates': '2dsphere' });
userSchema.index({ 'stats.overallRank': 1 });
userSchema.index({ createdAt: -1 });
```

```
module.exports = mongoose.model('User', userSchema);


// models/mongodb/TestSession.js
const testSessionSchema = new mongoose.Schema({
  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  testType: {
    type: String,
    enum: ['vertical_jump', 'situps', 'shuttle_run', 'endurance_run', 'flexibility'],
    required: true
  },
  videoSubmission: {
    originalPath: String,
    processedPath: String,
    thumbnailPath: String,
    duration: Number,
    fileSize: Number,
    resolution: String,
    uploadedAt: Date,
    storageProvider: {
      type: String,
      enum: ['aws_s3', 'gcp_storage', 'local'],
      default: 'aws_s3'
    },
    metadata: {
      fps: Number,
      codec: String,
```

```
    bitrate: Number
  }
 },
 aiAnalysis: {
  status: {
   type: String,
   enum: ['pending', 'processing', 'completed', 'failed'],
   default: 'pending'
  },
  processedAt: Date,
  modelVersion: String,
  results: {
   primaryMetric: Number, // Main score (jump height, reps, time, etc.)
   secondaryMetrics: {
    form: Number,
    consistency: Number,
    technique: Number
   },
   keypoints: [[Number]], // Pose keypoints for each frame
   motionData: {
    speed: [Number],
    acceleration: [Number],
    trajectory: [[Number]]
   },
   confidence: Number,
   flags: [String] // Any issues detected
  },
  cheatDetection: {
   authenticity: Number,
   manipulationFlags: [String],
   environmentalConsistency: Number,
```

```
    overallTrust: Number
  },
  processingTime: Number // in milliseconds
},
manualReview: {
  required: Boolean,
  reviewedBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Admin'
  },
  reviewedAt: Date,
  comments: String,
  adjustedScore: Number,
  approved: Boolean
},
benchmarkComparison: {
  ageGroup: String,
  genderGroup: String,
  percentile: Number,
  nationalAverage: Number,
  regionalAverage: Number,
  improvement: Number // compared to previous attempts
},
sessionMetadata: {
  deviceInfo: {
    model: String,
    os: String,
    appVersion: String
  },
  location: {
    coordinates: [Number],
```

```
    address: String
  },
  weather: {
    temperature: Number,
    humidity: Number,
    conditions: String
  },
  testConditions: {
    indoor: Boolean,
    lighting: String,
    surface: String
  }
 }
}, {
 timestamps: true,
 collection: 'test_sessions'
});

// Compound indexes for common queries
testSessionSchema.index({ userId: 1, testType: 1, createdAt: -1 });
testSessionSchema.index({ 'aiAnalysis.status': 1, createdAt: 1 });
testSessionSchema.index({ 'benchmarkComparison.percentile': -1 });

module.exports = mongoose.model('TestSession', testSessionSchema);
```

MySQL Schema (Alternative)

```
-- MySQL database schema
CREATE DATABASE sports_talent_assessment;
USE sports_talent_assessment;

-- Users table
CREATE TABLE users (
```

```
id BIGINT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
email VARCHAR(255) NOT NULL UNIQUE,
phone VARCHAR(15) NOT NULL UNIQUE,
date_of_birth DATE NOT NULL,
gender ENUM('male', 'female', 'other') NOT NULL,

-- Location data
state VARCHAR(100),
district VARCHAR(100),
pincode VARCHAR(10),
latitude DECIMAL(10, 8),
longitude DECIMAL(11, 8),

-- Profile information
avatar_url VARCHAR(500),
bio TEXT,
height_cm INT,
weight_kg DECIMAL(5,2),
sports JSON,
experience ENUM('beginner', 'intermediate', 'advanced'),

-- Authentication
password_hash VARCHAR(255) NOT NULL,
email_verified BOOLEAN DEFAULT FALSE,
phone_verified BOOLEAN DEFAULT FALSE,
verification_token VARCHAR(100),

-- Timestamps
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
```

```
    last_login TIMESTAMP,

    -- Indexes
    INDEX idx_email (email),
    INDEX idx_phone (phone),
    INDEX idx_### 3.2 Mobile Optimization for YOLOv8 + OpenCV

#### ONNX Model Conversion Strategy
```python
# YOLOv8 to ONNX conversion for mobile deployment
from ultralytics import YOLO
import torch

# Convert YOLOv8 to ONNX
model = YOLO('yolov8n-pose.pt')
model.export(format='onnx', dynamic=True, simplify=True)

# Optimization parameters:
# - Dynamic input shapes for flexible inference
# - Graph simplification for reduced model size
# - FP16 precision for 2x speed improvement
# - Size reduction: 6MB → 3MB (with quantization)
# - Inference speed: 40+ FPS on modern mobile devices
Quantization and Pruning Strategy
# Post-training quantization for mobile deployment
import onnxruntime as ort
from onnxruntime.quantization import quantize_dynamic, QuantType

def optimize_for_mobile(model_path):
    # Dynamic quantization
    quantized_model = quantize_dynamic(
```
```

```
    model_path,
    model_path.replace('.onnx', '_quantized.onnx'),
    weight_type=QuantType.QUInt8
  )

  # Performance improvements:
  # - Model size: 4x smaller
  # - Inference speed: 2-3# AI-Powered Sports Talent Assessment App
```

Complete Development Roadmap & Architecture

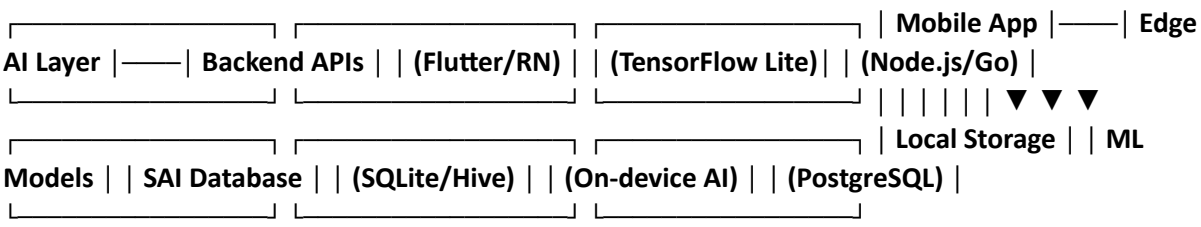
Executive Summary

This document outlines a comprehensive 4-week development plan for creating an AI-powered mobile platform that democratizes sports talent assessment across India, particularly targeting rural and remote areas.

---

1. Project Architecture Overview

1.1 System Architecture



1.2 Technology Stack Comparison

Frontend Mobile App

Technology	Pros	Cons	Recommendation
Flutter	✅ Single codebase, excellent performance, great camera plugins	❌ Larger app size	**RECOMMENDED**

| React Native | ✅ JavaScript familiarity, good community | ❌ Bridge overhead, camera issues | Alternative |

| Native (Kotlin/Swift) | ✅ Best performance, platform-specific features | ❌ Dual development | For production scale |

Backend Services

| Technology | Use Case | Pros | Cons |

|---|---|---|---|

| **Node.js + Express** | API Gateway, Real-time features | ✅ JavaScript ecosystem, fast development | ❌ Single-threaded |

| **Go + Gin** | High-performance APIs | ✅ Excellent concurrency, fast | ❌ Learning curve |

| Python + FastAPI | ML integration | ✅ AI/ML libraries, rapid prototyping | ❌ Performance limitations |

Database Solutions

| Database | Use Case | Justification |

|---|---|---|

| **PostgreSQL** | Primary database | ACID compliance, JSON support, scalability |

| **Redis** | Caching, sessions | High-performance in-memory storage |

| **SQLite** | Local mobile storage | Offline capability, lightweight |

AI/ML Framework

| Framework | Platform | Best For |

|---|---|---|

| **TensorFlow Lite** | Mobile | On-device inference, model optimization |

| **Core ML** | iOS only | iOS-specific optimizations |

| **ONNX Runtime** | Cross-platform | Model interoperability |

---

2. Detailed File Structure & Architecture

2.1 Mobile App Structure (Flutter)

```
sports_talent_app/
├── lib/
│   ├── main.dart # App entry point
│   ├── app/
│   │   ├── app.dart # App configuration
│   │   ├── routes.dart # Navigation routes
│   │   └── themes.dart # UI themes
│   ├── core/
│   │   ├── constants/
│   │   │   ├── api_constants.dart # API endpoints
│   │   │   ├── app_constants.dart # App-wide constants
│   │   │   └── test_constants.dart # Sports test parameters
│   │   ├── errors/
│   │   │   ├── exceptions.dart # Custom exceptions
│   │   │   └── failures.dart # Error handling
│   │   ├── network/
│   │   │   ├── api_client.dart # HTTP client wrapper
│   │   │   ├── network_info.dart # Connectivity checker
│   │   │   └── interceptors.dart # Request/response interceptors
│   │   └── utils/
│   │       ├── validators.dart # Input validation
│   │       ├── formatters.dart # Data formatting
│   │       └── permissions.dart # Device permissions
│   ├── features/
│   │   ├── authentication/
│   │   │   ├── data/
│   │   │   │   ├── datasources/
│   │   │   │   │   ├── auth_local_datasource.dart
│   │   │   │   │   └── auth_remote_datasource.dart
│   │   │   │   ├── models/
│   │   │   │   │   └── user_model.dart
│   │   │   │   └── repositories/
│   │   │   │       └── auth_repository_impl.dart
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   └── user_entity.dart
│   │   │   │   ├── repositories/
│   │   │   │   │   └── auth_repository.dart
│   │   │   │   └── usecases/
│   │   │   │       ├── login_usecase.dart
│   │   │   │       ├── register_usecase.dart
│   │   │   │       └── logout_usecase.dart
│   │   │   └── presentation/
│   │   │       ├── bloc/
│   │   │       │   ├── auth_bloc.dart
│   │   │       │   ├── auth_event.dart
│   │   │       │   └── auth_state.dart
│   │   │       ├── pages/
│   │   │       │   ├── login_page.dart
│   │   │       │   ├── register_page.dart
│   │   │       │   └── profile_page.dart
│   │   │       └── widgets/
│   │   │           ├── auth_form.dart
│   │   │           └── social_login_buttons.dart
│   │   ├── video_recording/
│   │   │   ├── data/
│   │   │   │   ├── datasources/
│   │   │   │   │   ├── camera_datasource.dart
│   │   │   │   │   └── video_storage_datasource.dart
│   │   │   │   ├── models/
│   │   │   │   │   ├── video_model.dart
│   │   │   │   │   └── recording_session_model.dart
│   │   │   │   └── repositories/
│   │   │   │       └── video_repository_impl.dart
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── video_entity.dart
│   │   │   │   │   └── test_session_entity.dart
│   │   │   │   ├── repositories/
│   │   │   │   │   └── video_repository.dart
│   │   │   │   └── usecases/
│   │   │   │       ├── start_recording_usecase.dart
│   │   │   │       ├── stop_recording_usecase.dart
│   │   │   │       └── save_video_usecase.dart
│   │   │   └── presentation/
│   │   │       ├── bloc/
│   │   │       │   ├── video_recording_bloc.dart
│   │   │       │   ├── video_recording_event.dart
│   │   │       │   └── video_recording_state.dart
│   │   │       ├── pages/
│   │   │       │   ├── test_selection_page.dart
│   │   │       │   ├── camera_page.dart
│   │   │       │   └── recording_review_page.dart
│   │   │       └── widgets/
│   │   │           ├── camera_overlay.dart
│   │   │           ├── test_timer.dart
│   │   │           ├── countdown_widget.dart
│   │   │           └── recording_controls.dart
│   │   ├── ai_analysis/
│   │   │   ├── data/
│   │   │   │   ├── datasources/
│   │   │   │   │   ├── ml_model_datasource.dart
│   │   │   │   │   └── analysis_cache_datasource.dart
│   │   │   │   ├── models/
│   │   │   │   │   ├── analysis_result_model.dart
│   │   │   │   │   └── performance_metrics_model.dart
│   │   │   │   └── repositories/
│   │   │   │       └── ai_analysis_repository_impl.dart
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── analysis_result_entity.dart
│   │   │   │   │   └── performance_data_entity.dart
│   │   │   │   ├── repositories/
│   │   │   │   │   └── ai_analysis_repository.dart
│   │   │   │   └── usecases/
│   │   │   │       ├── analyze_vertical_jump_usecase.dart
│   │   │   │       ├── count_situps_usecase.dart
│   │   │   │       ├── analyze_shuttle_run_usecase.dart
│   │   │   │       ├── detect_cheating_usecase.dart
│   │   │   │       └── benchmark_performance_usecase.dart
│   │   │   └── presentation/
│   │   │       ├── bloc/
│   │   │       │   ├── ai_analysis_bloc.dart
│   │   │       │   ├── ai_analysis_event.dart
│   │   │       │   └── ai_analysis_state.dart
│   │   │       ├── pages/
│   │   │       │   ├── analysis_loading_page.dart
│   │   │       │   ├── results_page.dart
│   │   │       │   └── performance_dashboard_page.dart
│   │   │       └── widgets/
│   │   │           ├── analysis_progress.dart
│   │   │           ├── performance_chart.dart
│   │   │           ├── benchmark_comparison.dart
│   │   │           └── cheat_detection_indicator.dart
│   │   ├── gamification/
│   │   │   ├── data/
│   │   │   │   ├── datasources/
│   │   │   │   │   ├── achievements_datasource.dart
│   │   │   │   │   └── leaderboard_datasource.dart
│   │   │   │   ├──
```

models/ | | | | | ├── badge_model.dart | | | | | ├── achievement_model.dart | | | | | └── leaderboard_model.dart | | | | └── repositories/ | | | | └── gamification_repository_impl.dart | | | ├── domain/ | | | | ├── entities/ | | | | | ├── badge_entity.dart | | | | | └── user_progress_entity.dart | | | | ├── repositories/ | | | | | └── gamification_repository.dart | | | | └── usecases/ | | | | ├── unlock_achievement_usecase.dart | | | | ├── get_leaderboard_usecase.dart | | | | └── update_progress_usecase.dart | | | └── presentation/ | | | ├── bloc/ | | | | ├── gamification_bloc.dart | | | | ├── gamification_event.dart | | | | └── gamification_state.dart | | | ├── pages/ | | | | ├── achievements_page.dart | | | | ├── leaderboard_page.dart | | | | └── progress_page.dart | | | └── widgets/ | | | ├── badge_widget.dart | | | ├── progress_bar.dart | | | ├── leaderboard_item.dart | | | └── achievement_popup.dart | | └── data_sync/ | | ├── data/ | | | ├── datasources/ | | | | ├── sync_local_datasource.dart | | | | └── sync_remote_datasource.dart | | | ├── models/ | | | | ├── sync_queue_model.dart | | | | └── upload_status_model.dart | | | └── repositories/ | | | └── sync_repository_impl.dart | | ├── domain/ | | | ├── entities/ | | | | └── sync_item_entity.dart | | | ├── repositories/ | | | | └── sync_repository.dart | | | └── usecases/ | | | ├── queue_for_sync_usecase.dart | | | ├── sync_data_usecase.dart | | | └── retry_failed_sync_usecase.dart | | └── presentation/ | | ├── bloc/ | | | ├── sync_bloc.dart | | | ├── sync_event.dart | | | └── sync_state.dart | | └── widgets/ | | ├── sync_status_indicator.dart | | └── upload_progress.dart | ├── shared/ | | ├── widgets/ | | | ├── custom_button.dart | | | ├── loading_indicator.dart | | | ├── error_widget.dart | | | ├── video_player.dart | | | └── performance_card.dart | | ├── services/ | | | ├── dependency_injection.dart # Service locator | | | ├── local_storage_service.dart # SQLite/Hive wrapper | | | ├── notification_service.dart # Push notifications | | | ├── biometric_service.dart # Fingerprint/face recognition | | | └── ml_service.dart # TensorFlow Lite wrapper | | └── extensions/ | | ├── string_extensions.dart | | ├── datetime_extensions.dart | | └── number_extensions.dart | └── models/ | ├── ai_models/ # TensorFlow Lite models | | ├── vertical_jump_detector.tflite | | ├── situp_counter.tflite | | ├── shuttle_run_analyzer.tflite | | ├── pose_estimation.tflite | | └── cheat_detection.tflite | └── benchmark_data/ | ├── age_gender_benchmarks.json | └── regional_benchmarks.json ├── assets/ | ├── images/ | | ├── logos/ | | ├── icons/ | | ├── badges/ | | └── tutorials/ | ├── videos/ | | └── tutorial_videos/ | ├── animations/ | | └── lottie_files/ | └── fonts/ ├── test/ | ├── unit/ | ├── widget/ | └── integration/ ├── pubspec.yaml # Dependencies └── README.md

2.2 Backend API Structure (Node.js)

sports-talent-backend/ ├── src/ | ├── app.js # Express app setup | ├── server.js # Server entry point | ├── config/ | | ├── database.js # DB configuration | | ├── redis.js # Redis configuration | | ├── aws.js # AWS S3 configuration | | ├── jwt.js # JWT configuration | | └── multer.js # File upload configuration | ├── controllers/ | | ├── authController.js # Authentication endpoints | | ├── userController.js # User management | | ├── videoController.js # Video upload/processing | | ├── analysisController.js # AI analysis results | | ├── benchmarkController.js # Performance benchmarks | | ├── leaderboardController.js # Gamification features | | └── adminController.js # SAI dashboard | ├── middleware/ | | ├── auth.js # JWT verification | | ├── validation.js # Request validation | | ├── rateLimiting.js # API rate limiting | | ├── errorHandler.js # Global error handling | | └── fileUpload.js # File processing | ├── models/ | | ├── User.js # User

schema | | ├── Athlete.js # Athlete profile | | ├── TestSession.js # Test session data | | ├── VideoSubmission.js # Video metadata | | ├── AnalysisResult.js # AI analysis results | | ├── Achievement.js # Gamification badges | | └── Benchmark.js # Performance standards | ├── routes/ | | ├── auth.js # Authentication routes | | ├── athletes.js # Athlete management | | ├── tests.js # Sports test routes | | ├── videos.js # Video handling | | ├── analysis.js # AI analysis | | ├── gamification.js # Badges and achievements | | └── admin.js # SAI dashboard routes | ├── services/ | | ├── authService.js # Authentication logic | | ├── videoProcessingService.js # Video processing pipeline | | ├── aiAnalysisService.js # AI model integration | | ├── benchmarkService.js # Performance comparison | | ├── notificationService.js # Push notifications | | ├── storageService.js # File storage (AWS S3) | | ├── encryptionService.js # Data encryption | | └── auditService.js # Activity logging | ├── utils/ | | ├── validators.js # Data validation | | ├── helpers.js # Utility functions | | ├── constants.js # Application constants | | ├── logger.js # Logging utility | | └── encryption.js # Security utilities | ├── jobs/ | | ├── videoProcessingJob.js # Background video processing | | ├── aiAnalysisJob.js # Batch AI analysis | | ├── dataCleanupJob.js # Database maintenance | | └── reportGenerationJob.js # Periodic reports | └── database/ | ├── migrations/ | | ├── 001_create_users_table.js | | ├── 002_create_athletes_table.js | | ├── 003_create_test_sessions_table.js | | ├── 004_create_video_submissions_table.js | | ├── 005_create_analysis_results_table.js | | └── 006_create_benchmarks_table.js | ├── seeds/ | | ├── benchmark_data.js | | └── sample_users.js | └── config.js ├── ai_models/ | ├── preprocessing/ | | ├── video_preprocessor.py # Video frame extraction | | ├── pose_estimator.py # Body pose detection | | └── frame_analyzer.py # Individual frame analysis | ├── models/ | | ├── vertical_jump/ | | | ├── jump_detector.py # Jump height calculation | | | ├── model.h5 # Trained model | | | └── preprocessing.py # Jump-specific preprocessing | | ├── situp_counter/ | | | ├── rep_counter.py # Repetition counting | | | ├── model.h5 | | | └── form_analyzer.py # Exercise form validation | | ├── shuttle_run/ | | | ├── speed_analyzer.py # Speed and agility analysis | | | ├── model.h5 | | | └── distance_calculator.py # Distance measurement | | └── cheat_detection/ | | ├── anomaly_detector.py # Unusual pattern detection | | ├── model.h5 | | └── authenticity_checker.py # Video manipulation detection | ├── training/ | | ├── data_preparation.py # Dataset preparation | | ├── model_training.py # Training pipeline | | ├── evaluation.py # Model evaluation | | └── optimization.py # Model optimization for mobile | └── inference/ | ├── batch_inference.py # Server-side batch processing | ├── real_time_inference.py # Real-time analysis | └── model_server.py # Model serving API ├── scripts/ | ├── setup_database.js # Database initialization | ├── migrate.js # Migration runner | ├── seed.js # Data seeding | ├── backup.js # Database backup | └── deploy.js # Deployment script ├── tests/ | ├── unit/ | ├── integration/ | └── e2e/ ├── docs/ | ├── api_documentation.md | ├── deployment_guide.md | └── troubleshooting.md ├── docker/ | ├── Dockerfile # Application container | ├── docker-compose.yml # Multi-service setup | └── nginx.conf # Load balancer configuration ├── .env.example # Environment variables template ├── package.json # Dependencies └── README.md

2.3 SAI Admin Dashboard Structure (React)

sai-admin-dashboard/ ├── public/ ├── src/ | ├── components/ | | ├── common/ | | | ├── Header.jsx | | | ├── Sidebar.jsx | | | ├── Footer.jsx | | | └── LoadingSpinner.jsx | | ├── charts/

| | | ├── PerformanceChart.jsx | | | ├── RegionalDistribution.jsx | | | └── TalentPipeline.jsx | | ├── athletes/ | | | ├── AthleteProfile.jsx | | | ├── AthleteList.jsx | | | ├── PerformanceAnalysis.jsx | | | └── VideoReview.jsx | | └── admin/ | | ├── UserManagement.jsx | | ├── SystemHealth.jsx | | └── Reports.jsx | ├── pages/ | | ├── Dashboard.jsx | | ├── Athletes.jsx | | ├── Analytics.jsx | | ├── Settings.jsx | | └── Reports.jsx | ├── services/ | | ├── api.js | | ├── auth.js | | └── websocket.js | ├── utils/ | | ├── constants.js | | ├── helpers.js | | └── validators.js | ├── styles/ | | ├── globals.css | | └── components/ | ├── App.jsx | └── index.js ├── package.json └── README.md

---

3. AI/ML Models Architecture

3.1 Model Specifications

Vertical Jump Detection Model

```python

 # Model Architecture

Input: Video frames (224x224x3)

├── CNN Feature Extractor (MobileNetV2 backbone)

├── Temporal Analysis Layer (LSTM)

├── Jump Detection Head

├── Height Calculation Module

└── Output: Jump height in cm, confidence score

 # Key Features:

- Real-time pose estimation

- Ground plane detection

- Jump trajectory analysis

- Height measurement accuracy: ±2cm

Sit-up Counter Model

 # Model Architecture

Input: Video frames (224x224x3)

├── Pose Estimation (MediaPipe)

├── Hip/Shoulder Angle Calculator

├── Rep Detection Algorithm

├── Form Validation Module

└── Output: Rep count, form score (0-100)

 # Key Features:

- Angle-based counting

- Form quality assessment

- False positive reduction

- Accuracy: 95%+ in controlled conditions

Shuttle Run Analyzer

 # Model Architecture

Input: Video frames (224x224x3)

├── Object Detection (Person tracking)

├── Speed Estimation Module

├── Distance Calculation

├── Agility Metrics Calculator

└── Output: Time, speed, agility score

 # Key Features:

- Multi-person tracking

- Speed profile analysis

- Direction change detection

- Timing accuracy: ±0.1 seconds

Cheat Detection Model

 # Model Architecture

Input: Video frames + metadata

├── Video Authenticity Checker

├── Motion Pattern Analyzer

├── Environmental Consistency Checker

├── Anomaly Detection Module

└── Output: Authenticity score (0-1), flags

 # Detection Capabilities:

- Video manipulation (deepfake, editing)

- Unrealistic motion patterns

- Environmental inconsistencies

- Equipment tampering

3.2 Model Optimization for Mobile

Quantization Strategy

 # Post-training quantization

model_int8 = tf.lite.TFLiteConverter.from_keras_model(model)

model_int8.optimizations = [tf.lite.Optimize.DEFAULT]

model_int8.target_spec.supported_types = [tf.lite.constants.INT8]

 # Size reduction: 4x smaller

 # Inference speed: 2-3x faster

 # Accuracy loss: <5%

Model Pruning

 # Structured pruning for mobile deployment

import tensorflow_model_optimization as tfmot

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

pruned_model = prune_low_magnitude(model,

pruning_schedule=tfmot.sparsity.keras.PolynomialDecay(

initial_sparsity=0.50,

final_sparsity=0.90,

begin_step=0,

end_step=1000))

---

4. 4-Week Development Timeline

Week 1: Foundation & Core Setup

Days 1-2: Project Setup & Architecture

- **[ ] Set up development environment**
- **[ ] Initialize Flutter project with clean architecture**
- **[ ] Set up backend Node.js project structure**
- **[ ] Configure database schemas (PostgreSQL)**
- **[ ] Set up Redis for caching**
- **[ ] Configure CI/CD pipeline (GitHub Actions)**

Days 3-4: Authentication & User Management

- **[ ] Implement user registration/login (mobile + backend)**
- **[ ] JWT token implementation**
- **[ ] Basic user profile management**
- **[ ] Database user tables and relationships**
- **[ ] Input validation and error handling**

Days 5-7: Video Recording Infrastructure

- **[ ] Camera integration (Flutter)**
- **[ ] Video recording functionality**
- **[ ] Local video storage**
- **[ ] Basic video player**
- **[ ] File upload preparation**
- **[ ] Video compression implementation**

Week 2: AI/ML Integration & Core Features

Days 8-10: AI Models Development

- **[ ] Set up TensorFlow/PyTorch training environment**
- **[ ] Collect and prepare training datasets**
- **[ ] Train vertical jump detection model**
- **[ ] Train sit-up counter model**
- **[ ] Train shuttle run analyzer**
- **[ ] Convert models to TensorFlow Lite**

Days 11-12: Mobile AI Integration

- **[ ] Integrate TensorFlow Lite into Flutter**
- **[ ] Implement on-device inference pipeline**
- **[ ] Pose estimation integration (MediaPipe)**
- **[ ] Real-time video analysis**
- **[ ] Performance optimization for low-end devices**

Days 13-14: Sports Test Implementation

- **[ ] Implement test selection UI**
- **[ ] Create test-specific recording interfaces**
- **[ ] Add countdown timers and instructions**
- **[ ] Implement basic analysis results display**
- **[ ] Test validation and error handling**

Week 3: Advanced Features & Backend

Days 15-16: Cheat Detection & Validation

- **[ ] Implement cheat detection algorithms**
- **[ ] Video authenticity verification**
- **[ ] Motion pattern analysis**
- **[ ] Environmental consistency checks**
- **[ ] Integration with main analysis pipeline**

Days 17-18: Backend API Development

- **[ ] Complete REST API endpoints**
- **[ ] Video upload and processing pipeline**
- **[ ] Background job processing (Bull Queue)**
- **[ ] Data synchronization logic**
- **[ ] Performance benchmarking system**

Days 19-21: Gamification & User Experience

- **[ ] Achievement system implementation**
- **[ ] Badge unlocking mechanism**
- **[ ] Leaderboard functionality**
- **[ ] Progress tracking**
- **[ ] Push notifications setup**
- **[ ] UI/UX polishing**

Week 4: Integration, Testing & Deployment

Days 22-24: SAI Admin Dashboard

- **[ ] React admin dashboard setup**
- **[ ] Athlete data visualization**
- **[ ] Performance analytics charts**
- **[ ] Video review interface**
- **[ ] Report generation features**
- **[ ] Real-time data updates (WebSocket)**

-- Indexes INDEX idx_email (email), INDEX idx_phone (phone), INDEX idx_location (latitude, longitude), INDEX idx_created (created_at));

-- Test sessions table CREATE TABLE test_sessions (id BIGINT PRIMARY KEY AUTO_INCREMENT, user_id BIGINT NOT NULL, test_type ENUM('vertical_jump', 'situps', 'shuttle_run', 'endurance_run', 'flexibility') NOT NULL,

-- Video information

video_original_path VARCHAR(500),

video_processed_path VARCHAR(500),

video_thumbnail_path VARCHAR(500),

video_duration INT, -- in seconds

video_file_size BIGINT, -- in bytes

video_resolution VARCHAR(20),

video_fps DECIMAL(5,2),

video_codec VARCHAR(50),

video_bitrate INT,

video_uploaded_at TIMESTAMP,

-- AI Analysis results

analysis_status ENUM('pending', 'processing', 'completed', 'failed') DEFAULT 'pending',

analysis_processed_at TIMESTAMP,

analysis_model_version VARCHAR(50),

analysis_primary_score DECIMAL(10,3),

analysis_form_score DECIMAL(5,2),

analysis_consistency_score DECIMAL(5,2),

analysis_technique_score DECIMAL(5,2),

analysis_confidence DECIMAL(5,4),

analysis_keypoints JSON,

analysis_motion_data JSON,

analysis_flags JSON,

analysis_processing_time INT, -- in milliseconds

-- Cheat detection

cheat_authenticity_score DECIMAL(5,4),

cheat_manipulation_flags JSON,

cheat_environmental_score DECIMAL(5,4),

cheat_overall_trust DECIMAL(5,4),

-- Manual review

manual_review_required BOOLEAN DEFAULT FALSE,

manual_reviewed_by BIGINT,

manual_reviewed_at TIMESTAMP,

manual_comments TEXT,

manual_adjusted_score DECIMAL(10,3),

manual_approved BOOLEAN,

-- Benchmark comparison

benchmark_age_group VARCHAR(20),

benchmark_gender_group VARCHAR(10),

benchmark_percentile DECIMAL(5,2),

benchmark_national_avg DECIMAL(10,3),

benchmark_regional_avg DECIMAL(10,3),

benchmark_improvement DECIMAL(10,3),

-- Session metadata

device_model VARCHAR(100),

device_os VARCHAR(50),

app_version VARCHAR(20),

session_latitude DECIMAL(10, 8),

session_longitude DECIMAL(11, 8),

session_address VARCHAR(200),

weather_temperature DECIMAL(4,1),

weather_humidity DECIMAL(5,2),

weather_conditions VARCHAR(50),

test_indoor BOOLEAN,

test_lighting VARCHAR(50),

test_surface VARCHAR(50),

-- Timestamps

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

-- Foreign keys

FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (manual_reviewed_by) REFERENCES users(id),

-- Indexes

INDEX idx_user_test_date (user_id, test_type, created_at),

INDEX idx_analysis_status (analysis_status, created_at),

INDEX idx_percentile (benchmark_percentile DESC),

INDEX idx_review_required (manual_review_required, created_at)

);

-- Achievements and badges CREATE TABLE achievements (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100) NOT NULL, description TEXT, badge_icon_url VARCHAR(500), badge_color VARCHAR(7), category ENUM('performance', 'consistency', 'improvement', 'participation', 'special') NOT NULL, criteria JSON NOT NULL, points INT DEFAULT 0, rarity ENUM('common', 'uncommon', 'rare', 'epic', 'legendary') DEFAULT 'common', is_active BOOLEAN DEFAULT TRUE, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

-- User achievements junction table CREATE TABLE user_achievements (id BIGINT PRIMARY KEY AUTO_INCREMENT, user_id BIGINT NOT NULL, achievement_id INT NOT NULL, unlocked_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, progress_data JSON,

FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (achievement_id) REFERENCES achievements(id) ON DELETE CASCADE,

UNIQUE KEY unique_user_achievement (user_id, achievement_id),

INDEX idx_user_unlocked (user_id, unlocked_at DESC)

);

-- Leaderboards CREATE TABLE leaderboards (id BIGINT PRIMARY KEY AUTO_INCREMENT, user_id BIGINT NOT NULL, test_type ENUM('vertical_jump', 'situps', 'shuttle_run', 'endurance_run', 'flexibility') NOT NULL, score DECIMAL(10,3) NOT NULL, session_id BIGINT NOT NULL,

-- Ranking categories

overall_rank INT,

age_group_rank INT,

gender_rank INT,

regional_rank INT,

-- Metadata

age_group VARCHAR(20),

gender VARCHAR(10),

region VARCHAR(100),

-- Timestamps

achieved_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,

FOREIGN KEY (session_id) REFERENCES test_sessions(id) ON DELETE CASCADE,

-- Indexes for efficient ranking queries

INDEX idx_overall_rank (test_type, overall_rank),

INDEX idx_age_gender_rank (test_type, age_group, gender, score DESC),

INDEX idx_regional_rank (test_type, region, score DESC),

INDEX idx_user_scores (user_id, test_type, score DESC)

);

-- Performance benchmarks CREATE TABLE benchmarks (id INT PRIMARY KEY AUTO_INCREMENT, test_type ENUM('vertical_jump', 'situps', 'shuttle_run', 'endurance_run', 'flexibility') NOT NULL, age_group VARCHAR(20) NOT NULL, gender ENUM('male', 'female', 'combined') NOT NULL, region VARCHAR(100),

-- Statistical data

sample_size INT NOT NULL,

mean_score DECIMAL(10,3) NOT NULL,

median_score DECIMAL(10,3) NOT NULL,

std_deviation DECIMAL(10,3) NOT NULL,

percentile_10 DECIMAL(10,3),

percentile_25 DECIMAL(10,3),

percentile_50 DECIMAL(10,3),

percentile_75 DECIMAL(10,3),

percentile_90 DECIMAL(10,3),

percentile_95 DECIMAL(10,3),

percentile_99 DECIMAL(10,3),

-- Data validity

last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

data_source VARCHAR(100),

is_official BOOLEAN DEFAULT FALSE,

UNIQUE KEY unique_benchmark (test_type, age_group, gender, region),

INDEX idx_lookup (test_type, age_group, gender)

);

-- Sync queue for offline support CREATE TABLE sync_queue (id BIGINT PRIMARY KEY AUTO_INCREMENT, user_id BIGINT NOT NULL, table_name VARCHAR(50) NOT NULL, record_id BIGINT NOT NULL, action ENUM('INSERT', 'UPDATE', 'DELETE') NOT NULL, data JSON NOT NULL, priority INT DEFAULT 1, attempts INT DEFAULT 0, max_attempts INT DEFAULT 3, last_attempt_at TIMESTAMP, error_message TEXT, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

```
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
INDEX idx_sync_priority (priority DESC, created_at),
INDEX idx_sync_status (attempts, max_attempts, created_at)
);
```

5.4 Performance Optimization Strategies

Mobile App Optimization

```dart
// Performance optimization for Flutter app
class PerformanceOptimizations {
  // Lazy loading for heavy computations
  static Widget buildOptimizedVideoPlayer(String videoPath) {
    return FutureBuilder<VideoPlayerController>(
      future: _initializeVideoController(videoPath),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const CircularProgressIndicator();
        }
        if (snapshot.hasData) {
          return VideoPlayer(snapshot.data!);
        }
        return const Text('Error loading video');
      },
    );
  }

  // Memory management for video processing
  static Future<void> processVideoWithMemoryManagement(String videoPath) async {
    const int maxFramesInMemory = 30; // Process in chunks
    final frames = <VideoFrame>[];
```

```
  await for (final frame in VideoProcessor.extractFrames(videoPath)) {
    frames.add(frame);

    if (frames.length >= maxFramesInMemory) {
      // Process current chunk
      await _processFrameChunk(frames);

      // Clear memory
      frames.clear();

      // Force garbage collection if needed
      if (await DeviceMemory.isLowMemory()) {
        await Future.delayed(Duration(milliseconds: 100));
      }
    }
  }

  // Process remaining frames
  if (frames.isNotEmpty) {
    await _processFrameChunk(frames);
  }
}

// Background processing for AI analysis
static void analyzeVideoInBackground(String videoPath, String testType) {
  compute(_runAIAnalysis, {
    'videoPath': videoPath,
    'testType': testType,
  });
}
```

```
static Map<String, dynamic> _runAIAnalysis(Map<String, String> params) {
  // Heavy AI processing runs in isolate
  final analyzer = AIAnalysisEngine();
  return analyzer.analyze(params['videoPath']!, params['testType']!);
}

// Efficient image loading and caching
static Widget buildOptimizedImage(String imageUrl) {
  return CachedNetworkImage(
    imageUrl: imageUrl,
    memCacheHeight: 200, // Limit memory usage
    memCacheWidth: 200,
    placeholder: (context, url) => Shimmer.fromColors(
      baseColor: Colors.grey[300]!,
      highlightColor: Colors.grey[100]!,
      child: Container(
        width: 200,
        height: 200,
        color: Colors.white,
      ),
    ),
    errorWidget: (context, url, error) => Icon(Icons.error),
  );
}

// Battery optimization
static void optimizeForBatteryLife() {
  // Reduce frame rate for non-critical operations
  WidgetsBinding.instance.addPostFrameCallback((_) {
    if (BatteryLevel.isLow()) {
```

```
      AIAnalysisService.setLowPowerMode(true);
      VideoRecording.reduceQuality();
    }
  });
 }
}

// Device capability detection
class DeviceCapabilities {
 static bool isHighEndDevice() {
  final deviceInfo = DeviceInfoPlugin();
  // Check RAM, CPU, GPU capabilities
  return _checkHardwareSpecs();
 }

 static Map<String, dynamic> getOptimalSettings() {
  return {
   'videoResolution': isHighEndDevice() ? '1080p' : '720p',
   'aiModelComplexity': isHighEndDevice() ? 'high' : 'medium',
   'maxConcurrentOperations': isHighEndDevice() ? 4 : 2,
   'cacheSize': isHighEndDevice() ? 100 : 50, // MB
  };
 }
}
```

5.5 Security Implementation

Data Encryption and Security

```
// Security service for sensitive data
class SecurityService {
 static const String _keyAlias = 'sports_talent_key';

 // Encrypt sensitive data before local storage
```

```
static Future<String> encryptData(String data) async {
  final key = await _getOrCreateKey();
  final encryptedData = await AESEncryption.encrypt(data, key);
  return base64Encode(encryptedData);
}

static Future<String> decryptData(String encryptedData) async {
  final key = await _getOrCreateKey();
  final decodedData = base64Decode(encryptedData);
  return await AESEncryption.decrypt(decodedData, key);
}

// Secure video file handling
static Future<String> secureVideoUpload(String videoPath) async {
  // Add watermark with user ID and timestamp
  final watermarkedVideo = await VideoProcessor.addWatermark(
    videoPath,
    watermark: _generateSecurityWatermark(),
  );

  // Calculate file hash for integrity
  final fileHash = await FileUtils.calculateSHA256(watermarkedVideo);

  // Upload with metadata
  return await CloudStorage.uploadSecure(watermarkedVideo, {
    'hash': fileHash,
    'uploadedAt': DateTime.now().toIso8601String(),
    'deviceId': await DeviceInfo.getDeviceId(),
  });
}
```

```
// Biometric authentication
static Future<bool> authenticateUser() async {
  final localAuth = LocalAuthentication();

  try {
    final isAvailable = await localAuth.canCheckBiometrics;
    if (!isAvailable) return false;

    return await localAuth.authenticate(
      localizedReason: 'Please authenticate to access sports assessment',
      options: AuthenticationOptions(
        biometricOnly: true,
        stickyAuth: true,
      ),
    );
  } catch (e) {
    return false;
  }
}

// API request signing
static Map<String, String> signRequest(Map<String, dynamic> data) {
  final timestamp = DateTime.now().millisecondsSinceEpoch.toString();
  final nonce = _generateNonce();
  final signature = _generateHMAC(data, timestamp, nonce);

  return {
    'X-Timestamp': timestamp,
    'X-Nonce': nonce,
    'X-Signature': signature,
  };
```

```
  }
}
```

5.6 Testing Strategy

Comprehensive Testing Plan

```
// Unit tests for AI analysis
void main() {
  group('YOLOv8 Analysis Tests', () {
    late YOLOv8Service yoloService;

    setUp(() {
      yoloService = YOLOv8Service();
    });

    testWidgets('Vertical jump detection accuracy', (tester) async {
      final testVideo = 'assets/test_videos/vertical_jump_sample.mp4';
      final result = await yoloService.analyzeVerticalJump(testVideo);

      expect(result.jumpHeight, greaterThan(0));
      expect(result.confidence, greaterThan(0.8));
      expect(result.keypoints, hasLength(17)); // COCO pose format
    });

    testWidgets('Situp counting accuracy', (tester) async {
      final testVideo = 'assets/test_videos/situps_10_reps.mp4';
      final result = await yoloService.analyzeSitups(testVideo);

      expect(result.repCount, equals(10));
      expect(result.formScore, greaterThan(70));
      expect(result.averageConfidence, greaterThan(0.85));
    });
```

```
  testWidgets('Cheat detection sensitivity', (tester) async {
    final fakeVideo = 'assets/test_videos/manipulated_jump.mp4';
    final result = await yoloService.detectCheating(fakeVideo);

    expect(result.authenticityScore, lessThan(0.5));
    expect(result.flags, contains('motion_inconsistency'));
  });
});

group('Database Integration Tests', () {
  testWidgets('MongoDB connection and CRUD', (tester) async {
    final db = MongoDBService();
    await db.connect();

    // Test user creation
    final user = AthleteModel(
      name: 'Test Athlete',
      email: 'test@example.com',
      age: 25,
    );

    final userId = await db.createUser(user);
    expect(userId, isNotNull);

    // Test session creation
    final session = TestSessionModel(
      userId: userId,
      testType: TestType.verticalJump,
      videoPath: '/path/to/test/video.mp4',
    );
```

```
    final sessionId = await db.createSession(session);
    expect(sessionId, isNotNull);

    await db.disconnect();
  });
});

group('Performance Tests', () {
  testWidgets('AI processing time within limits', (tester) async {
    final stopwatch = Stopwatch()..start();

    final result = await YOLOv8Service.quickAnalysis('test_video.mp4');

    stopwatch.stop();
    expect(stopwatch.elapsedMilliseconds, lessThan(5000)); // 5 second limit
  });

  testWidgets('Memory usage under control', (tester) async {
    final initialMemory = await DeviceMemory.getCurrentUsage();

    // Process multiple videos
    for (int i = 0; i < 5; i++) {
      await YOLOv8Service.analyzeVideo('test_video_$i.mp4');
    }

    final finalMemory = await DeviceMemory.getCurrentUsage();
    final memoryIncrease = finalMemory - initialMemory;

    expect(memoryIncrease, lessThan(50)); // Less than 50MB increase
  });
});
```

```
}

// Integration tests
void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('End-to-End Tests', () {
    testWidgets('Complete video analysis workflow', (tester) async {
      await tester.pumpWidget(SportsAssessmentApp());

      // Login
      await tester.tap(find.byKey(Key('login_button')));
      await tester.pumpAndSettle();

      // Record video
      await tester.tap(find.byKey(Key('vertical_jump_test')));
      await tester.pumpAndSettle();

      // Simulate video recording
      await tester.tap(find.byKey(Key('record_button')));
      await Future.delayed(Duration(seconds: 10)); // Simulate recording
      await tester.tap(find.byKey(Key('stop_button')));
      await tester.pumpAndSettle();

      // Wait for analysis
      await tester.pumpAndSettle(Duration(seconds: 30));

      // Verify results displayed
      expect(find.byKey(Key('analysis_results')), findsOneWidget);
      expect(find.textContaining('Jump Height:'), findsOneWidget);
    });
```

```
  });
}
```

---

6. Deployment and DevOps Strategy

6.1 CI/CD Pipeline Configuration

```
# .github/workflows/flutter-ci.yml
name: Flutter CI/CD

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
        with:
          flutter-version: '3.16.0'

      - name: Install dependencies
        run: flutter pub get

      - name: Run analyzer
        run: flutter analyze

      - name: Run tests
        run: flutter test --coverage
```

```
    - name: Upload coverage
      uses: codecov/codecov-action@v3
      with:
        file: coverage/lcov.info

  build-android:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - uses: actions/setup-java@v3
        with:
          distribution: 'zulu'
          java-version: '11'

      - name: Setup Android SDK
        uses: android-actions/setup-android@v2

      - name: Build APK
        run: flutter build apk --release

      - name: Upload APK
        uses: actions/upload-artifact@v3
        with:
          name: app-release.apk
          path: build/app/outputs/flutter-apk/app-release.apk

  build-ios:
    needs: test
```

```
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - name: Build iOS
        run: flutter build ios --release --no-codesign

# .github/workflows/backend-ci.yml
name: Backend CI/CD

on:
  push:
    paths: ['backend/**']

jobs:
  test:
    runs-on: ubuntu-latest
    services:
      mongodb:
        image: mongo:5.0
        ports:
          - 27017:27017
      redis:
        image: redis:7.0
        ports:
          - 6379:6379

    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
```

```
        node-version: '18'

    - name: Install dependencies
      working-directory: ./backend
      run: npm ci

    - name: Run tests
      working-directory: ./backend
      run: npm test
      env:
        MONGODB_URI: mongodb://localhost:27017/test
        REDIS_URL: redis://localhost:6379

  deploy:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
    - name: Deploy to production
      run: |
        # Deployment commands here
        echo "Deploying to production..."
```

6.2 Docker Configuration

```
# backend/Dockerfile
FROM node:18-alpine

# Install system dependencies for AI processing
RUN apk add --no-cache \
    python3 \
    py3-pip \
    opencv-dev \
```

```
    opencv-python \
    ffmpeg

WORKDIR /app

# Copy package files
COPY package*.json ./
RUN npm ci --only=production

# Copy AI models
COPY ai_models/ ./ai_models/
RUN pip3 install ultralytics opencv-python onnxruntime

# Copy application code
COPY . .

EXPOSE 3000

# Health check
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:3000/health || exit 1

CMD ["npm", "start"]

# docker-compose.yml
version: '3.8'

services:
  backend:
    build: ./backend
    ports:
```

```
    - "3000:3000"
    environment:
      - NODE_ENV=production
      - MONGODB_URI=mongodb://mongodb:27017/sports_talent
      - REDIS_URL=redis://redis:6379
    depends_on:
      - mongodb
      - redis
    volumes:
      - ./uploads:/app/uploads
      - ./ai_models:/app/ai_models

  mongodb:
    image: mongo:5.0
    restart: always
    volumes:
      - mongodb_data:/data/db
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_PASSWORD}

  redis:
    image: redis:7.0-alpine
    restart: always
    command: redis-server --appendonly yes
    volumes:
      - redis_data:/data

  nginx:
    image: nginx:alpine
    ports:
```

```
    - "80:80"
    - "443:443"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
    - ./ssl:/etc/nginx/ssl
  depends_on:
    - backend

volumes:
  mongodb_data:
  redis_data:
```