Final Submission Requirements

National Hackathon

AI Powered Battery Fitness Assessment Application

This document specifies every artifact that participating teams must submit for final evaluation. Incomplete submissions will not be evaluated. Read this alongside the Concept Note and Participant Guide (v3.0, July 2026).

Date	July 2026
Reference	Concept Note and Participant Guide: AI Powered Battery Fitness Assessment Application, v3.0, NeGD Team, July 2026
Issued by	National e-Governance Division (NeGD), MeitY, in partnership with MYAS
Audience	All participating hackathon teams

Table of Contents

In Microsoft Word, right-click below and select Update Field to refresh page numbers.

1. Submission Overview

- **Every team must submit a complete submission package** containing all mandatory artifacts listed in this document. Submissions missing any mandatory artifact will not be evaluated.
- **The submission package is divided into 10 categories:** Application, Source Code, Architecture Documents (HLD and LLD), AI Model Documentation, Accuracy Validation Report, Demo Video, Presentation, Data Protection Declaration, User Guide, and Known Issues Log.
- **All artifacts must be submitted as a single compressed archive** (ZIP or TAR.GZ) uploaded to the designated submission portal before the deadline. The archive must follow the folder structure specified in Section 12.
- **Late submissions will not be accepted.** Partial submissions (missing mandatory artifacts) will not be evaluated. Teams are encouraged to submit early and update before the deadline if needed.

2. Application (APK and Installable Build)

No.	Artifact	Specification
2.1	**Android APK (Release Build)**	A signed, release-mode APK file installable on any Android device running Android 10 or later with 3 GB RAM and a standard rear camera. The APK must not require Google Play Services to function. Debug builds will not be accepted.
2.2	**App version and build metadata**	The APK filename must include the team name and version number (e.g., TeamAlpha_BatteryTest_v1.0.apk). The app's About screen must display the version number, build date and team name.
2.3	**Installation instructions**	A README file (plain text or Markdown) with step-by-step installation instructions, including any permissions the app requires and how to grant them. If the app requires any initial setup (calibration, language selection, registration), document the exact steps.
2.4	**Test credentials (if applicable)**	If the app requires login to function, provide at least 2 pre-configured test accounts (one coach, one athlete) with credentials documented in the README. The evaluator must not need to create an account to begin testing.
2.5	**iOS build**	If an iOS version exists, provide a TestFlight link or an IPA file with installation instructions. This is not mandatory.

3. Source Code

The source code is the most important artifact after the working app. The winning solution will be taken to production. Incomplete, obfuscated or unrunnable source code will result in disqualification.

No.	Artifact	Specification
3.1	**Complete source code repository**	The full source code of the application, including all modules, packages and dependencies. Must be provided as a Git repository (ZIP of the .git directory included) so that commit history is visible. Do not submit only the final snapshot; the evaluation committee will review the development history.
3.2	**Build instructions**	A BUILD.md file with exact steps to compile the source code into the submitted APK. Include: required SDK versions (Android SDK, Gradle, NDK if used), required environment variables, build commands, and expected build time. The evaluator must be able to reproduce the APK from source.
3.3	**Dependency manifest**	A complete list of all third-party libraries, frameworks, models and tools used, with version numbers and licence types. This includes: app frameworks (Flutter, React Native, Kotlin, Java), AI/ML libraries (TensorFlow Lite, ONNX Runtime, MediaPipe, Ultralytics), any pre-trained model files, and any other open-source components.
3.4	**AI model files**	All trained or pre-trained model files (TFLite, ONNX, PyTorch, etc.) used by the app, placed in a clearly labelled /models directory. Each model file must be accompanied by a model card (see Section 5). If model files exceed 100 MB, provide a download script that fetches them from a public URL (HuggingFace, GitHub Releases).
3.5	**Backend source (if any)**	If the solution includes a backend server for sync, dashboards or API endpoints, the backend source code must be included with its own build and deployment instructions (Dockerfile preferred). Include database schema and migration scripts.
3.6	**Technical Document**	Code must be reasonably documented with inline comments for non-obvious logic, especially in the AI measurement pipelines. Function and variable names must be descriptive. Dead code and debug logging must be removed from the release build.

4. Architecture Documents

4.1 High-Level Design (HLD)

A document (PDF or DOCX, 8 to 15 pages) covering the system at a component level. Must include:

No.	Section	What to Include
4.1.1	**System context diagram**	A diagram showing the app, the athlete/coach, the sovereign cloud, any managed services, and all external systems (APAAR, Aadhaar, NSRS, Khelo India Portal). Show data flows between each component.

No.	Section	What to Include
4.1.2	**Component architecture**	A diagram and description of the major components inside the app: UI layer, test workflow engine, video capture module, AI inference engine, local database, sync queue, report card generator. Show how they interact.
4.1.3	**AI pipeline overview**	A diagram showing the end-to-end AI pipeline for at least 3 tests: video input, pre-processing, model inference, post-processing, measurement output, confidence score computation. Name the specific models used at each stage.
4.1.4	**Data flow and storage**	How data moves from video capture to local storage to cloud sync. Show the encryption approach, the offline queue design, and how the system handles crash recovery (WAL, journaling, etc.).
4.1.5	**Cloud and sync architecture**	How the app syncs to sovereign infrastructure when connectivity returns. Show resumable upload design, conflict resolution strategy, and the de-identification gateway (if implemented). Indicate what runs on sovereign infra versus managed services.
4.1.6	**Integration design**	How the app would integrate with APAAR, Aadhaar (e-KYC) and NSRS in production. API contracts, data exchange formats, and NSRS ID auto-creation trigger logic. Design-level is acceptable; live integration is not required.
4.1.7	**Security architecture**	Encryption at rest and in transit, role-based access control, device attestation approach, and how video sovereignty is enforced (no video leaves sovereign infrastructure).
4.1.8	**Scalability approach**	How the architecture would scale to 10,000+ concurrent sessions in production. Stateless API design, queue-based video processing, database partitioning strategy.

4.2 Low-Level Design (LLD)

A document (PDF or DOCX, 10 to 20 pages) covering the system at the implementation level. Must include:

No.	Section	What to Include
4.2.1	**Database schema**	Complete entity-relationship diagram with all tables, columns, data types, primary keys, foreign keys and indexes. Cover: athletes, coaches, assessments, test results, video metadata, sync queue. Include schema for both the local SQLite/SQLCipher database and the server-side database (if any).
4.2.2	**API specification**	All REST or GraphQL endpoints for the sync and backend layer, with request/response schemas, authentication method, error codes, and rate limiting. Provide in OpenAPI/Swagger format if possible, or as a structured table.

No.	Section	What to Include
4.2.3	**AI model integration detail**	For each AI-measured test: the exact model file used, input tensor shape and format, pre-processing steps (resize, normalise, colour space), post-processing steps (keypoint extraction, distance computation, state machine logic), and output format. Include code references.
4.2.4	**Video capture pipeline**	Camera initialisation, resolution and frame rate selection logic, frame extraction for AI inference, compression settings, metadata injection, and local storage write path. How the app handles different device capabilities (30 fps versus 120 fps, different resolutions).
4.2.5	**Offline sync queue design**	Queue data structure, retry logic, resumable upload protocol (chunked upload with byte-range support), conflict resolution when the same athlete is assessed on two devices, and how the queue drains on reconnection.
4.2.6	**Report card generation**	Template engine used, data binding approach, PDF rendering library, multilingual text handling (Hindi and English at minimum), and how the report card is generated without internet.
4.2.7	**State machine diagrams**	State diagrams for: the overall battery workflow (registration to report card), the individual test workflow (capture guidance to measurement to result), and the AI measurement logic for sit-up rep counting and shuttle-run turn detection.
4.2.8	**Error handling and edge cases**	How the app handles: mid-test app crash, storage full during recording, camera permission denied, athlete leaves frame during a test, video too dark or too bright, multiple people in frame, and device rotation during recording.

5. AI Model Documentation (Model Cards)

For every AI model used in the solution, provide a model card in the following format. Compile all model cards into a single document (PDF or DOCX).

Field	What to Document
Model name and version	The exact model name, version, and source (e.g., YOLOv8x-Pose, Ultralytics v8.2.0, downloaded from ultralytics.com).
Purpose in the solution	Which test(s) this model is used for and what it does (e.g., pose estimation for Tests 1, 4, 5, 9; object detection for Test 6).
Training data	What dataset the model was trained on (e.g., COCO 2017 Keypoints). If fine-tuned by the team, describe the fine-tuning dataset, size and collection method.
Model architecture and size	Architecture type (e.g., YOLOv8x with CSPDarknet backbone), number of parameters, model file size in MB, and quantisation level (FP32, FP16, INT8).

Field	What to Document
Input specification	Input tensor shape, colour space, normalisation method, and any required pre-processing.
Output specification	Output tensor shape, what each output represents (keypoints, bounding boxes, class scores), and any required post-processing.
Runtime and framework	Inference framework (TFLite, ONNX Runtime, MediaPipe), hardware acceleration (GPU, NPU, CPU), and whether the model runs in real time or post-capture.
Performance on target device	Inference time in milliseconds on the target device (name the device model and chipset). FPS achievable for video-based tests.
Known limitations	Known failure modes: lighting conditions, clothing occlusion, camera angle sensitivity, frame rate dependency, distance range, and any body-type or demographic limitations observed.
Licence	The licence under which the model is distributed (AGPL, Apache 2.0, MIT, proprietary, etc.) and any usage restrictions.

6. Accuracy Validation Report

> *This is where honesty matters most. A small but rigorous validation is worth far more than inflated numbers. The evaluation committee will scrutinise methodology, not just results.*

No.	Section	What to Include
6.1	**Validation methodology**	For each AI-measured test: the ground truth method used (stadiometer, measuring tape, calibrated stopwatch, manual counting), the number of subjects tested, their age and gender distribution, and the environmental conditions during testing.
6.2	**Device matrix**	List every device model tested on, with chipset, RAM, camera frame rate, and Android version. Report accuracy results separately for each device.
6.3	**Environmental conditions**	Describe the lighting, surface, background and weather conditions during validation. Categorise as: outdoor direct sunlight, outdoor overcast, indoor well-lit, indoor poorly-lit.
6.4	**Results per test**	For each AI-measured test: mean error, standard deviation, min error, max error, and the number of measurements in the sample. Present as a table.
6.5	**Failure cases**	Describe at least 3 cases where the AI measurement failed or was significantly inaccurate. Explain why (e.g., occlusion, wrong camera angle, low frame rate, poor lighting). This demonstrates understanding of the system's limitations.
6.6	**Dataset disclosure**	If any international or synthetic dataset was used for training or validation, disclose it here: name, source, size, demographics, and

No.	Section	What to Include
		how it differs from Indian conditions.
6.7	**Honest assessment**	A candid paragraph stating: which accuracy benchmarks you believe your system can meet, which it cannot yet meet, and what would be needed to close the gap (more data, higher frame rate, better calibration, etc.).

7. Demo Video

No.	Artifact	Specification
7.1	**Full demo video**	A single continuous screen recording (8 to 12 minutes) showing the complete workflow: app launch, athlete registration, at least 4 tests demonstrated end-to-end (protocol guidance, video recording, AI measurement, result display), report card generation, and PDF export. Must be recorded on a real Android device, not an emulator.
7.2	**Offline demonstration**	Within the demo video, clearly show the device being switched to aeroplane mode, at least 2 tests completed fully offline, and the report card generated without internet. Then show connectivity restored and sync initiated.
7.3	**Voiceover or captions**	The video must include either a spoken narration explaining what is being shown at each step, or on-screen captions and annotations. Silent screen recordings without explanation will not be accepted.
7.4	**Format and quality**	MP4 format, 1080p minimum, maximum file size 500 MB. If the file exceeds 500 MB, provide a link to a download location (Google Drive, OneDrive or similar, not YouTube). The video must be downloadable, not streaming-only.

8. Presentation Deck

No.	Artifact	Specification
8.1	**Presentation file**	A slide deck (PDF or PPTX, 12 to 20 slides) for the live demo session. Must be self-contained (no external links required to view).
8.2	**Required slide structure**	Cover the following in order: (1) Team introduction, (2) Problem understanding, (3) Solution overview, (4) AI approach per test (with diagrams), (5) Architecture (HLD summary), (6) Live demo plan, (7) Accuracy results, (8) Limitations and what you would improve, (9) Production readiness assessment, (10) Q and A.
8.3	**Time allocation**	Teams will have 20 minutes total: 8 minutes for presentation, 7

No.	Artifact	Specification
		minutes for live demo on a real device, and 5 minutes for evaluator questions. Plan accordingly.

9. Data Protection Compliance Declaration

This is a mandatory legal document. Unsigned or missing declarations will result in disqualification.

No.	Artifact	Specification
9.1	**Team declaration**	A signed declaration (PDF with signatures of all team members) stating: (a) no athlete video was uploaded to any commercial cloud service during development or testing, (b) all video data is stored exclusively on hardware under the team's direct physical control, (c) informed consent was obtained from all subjects recorded during validation, (d) for any subjects under 18, written parental/guardian consent was obtained, (e) all data will be deleted or handed over to the organisers within 30 days of the hackathon conclusion.
9.2	**Consent forms (anonymised)**	Anonymised copies (names redacted) of the consent forms used for any video footage collected during prototyping and validation. If no footage was collected, state so explicitly.
9.3	**Data inventory**	A table listing: number of subjects recorded, age range, whether any were minors, storage location of the footage, and planned deletion date.

10. User Guide

No.	Artifact	Specification
10.1	**Quick start guide**	A 2 to 3 page guide (PDF, with screenshots) that a PE teacher with no technical background could follow to: install the app, register an athlete, complete one test, and view the result. Written in simple language, Hindi and English.
10.2	**Test-by-test recording instructions**	For each of the 10 tests: recommended camera placement (distance, height, angle), what should be visible in the frame, lighting tips, and common mistakes to avoid. Include at least one diagram or annotated screenshot per test.
10.3	**Troubleshooting guide**	A short FAQ covering: what to do if the AI measurement seems wrong, what to do if the app crashes mid-test, how to re-record a test, how to switch between Hindi and English, and how to export the report card.

11. Known Issues and Limitations Log

A structured document (PDF or Markdown) listing every known issue, limitation and gap in the current solution. This demonstrates maturity and self-awareness, and will be evaluated positively.

Column	Description
Issue ID	A sequential identifier (e.g., KI-001, KI-002).
Category	One of: AI Accuracy, App Stability, UI/UX, Offline, Performance, Security, Scalability, Data Protection.
Description	Clear description of the issue or limitation.
Impact	How it affects the user experience or measurement quality. Rate as High, Medium or Low.
Workaround (if any)	Any temporary mitigation available to the user.
Resolution plan	What would be needed to fix this for production (more data, better model, hardware requirement, etc.).

12. Submission Folder Structure

> *Your submission archive must follow this exact folder structure. Evaluators will look for artifacts in these locations. Missing folders or misplaced files may cause artifacts to be overlooked.*

The submission must be a single archive named TeamName_Submission.zip containing:

- **/README.md** — top-level overview, team name, members, and quick links to each section below
- **/app/** — APK file, installation instructions, test credentials
- **/source/** — complete source code repository (with .git history)
- **/source/BUILD.md** — build instructions to reproduce the APK from source
- **/source/DEPENDENCIES.md** — complete dependency manifest with versions and licences
- **/models/** — all AI model files (or download scripts if files exceed 100 MB)
- **/docs/HLD.pdf** — High-Level Design document
- **/docs/LLD.pdf** — Low-Level Design document
- **/docs/Model_Cards.pdf** — AI model documentation for every model used
- **/docs/Validation_Report.pdf** — accuracy validation report with methodology and results
- **/docs/User_Guide.pdf** — quick start guide and test-by-test recording instructions
- **/docs/Known_Issues.md** — known issues and limitations log
- **/docs/API_Spec.yaml (or .json)** — API specification for sync and backend endpoints (if applicable)
- **/docs/DB_Schema.sql (or .png)** — database schema (SQL dump or ER diagram image)
- **/demo/** — demo video file (or link file if hosted externally)
- **/presentation/** — slide deck (PDF or PPTX)

- **/compliance/** — signed data protection declaration, anonymised consent forms, data inventory
- **/backend/** — backend source code, Dockerfile, deployment instructions (if applicable)

13. Final Submission Checklist

Artifact	Mandatory?	Section Reference
Android APK (signed release build)	Yes	Section 2.1
Installation instructions (README)	Yes	Section 2.3
Test credentials (if login required)	Yes	Section 2.4
Complete source code (with Git history)	Yes	Section 3.1
Build instructions (BUILD.md)	Yes	Section 3.2
Dependency manifest (DEPENDENCIES.md)	Yes	Section 3.3
AI model files (or download script)	Yes	Section 3.4
High-Level Design document (HLD)	Yes	Section 4.1
Low-Level Design document (LLD)	Yes	Section 4.2
AI model cards (one per model)	Yes	Section 5
Accuracy validation report	Yes	Section 6
Demo video (8-12 min, with narration)	Yes	Section 7
Presentation deck (12-20 slides)	Yes	Section 8
Data protection declaration (signed)	Yes	Section 9.1
Consent forms (anonymised)	Yes	Section 9.2
Data inventory	Yes	Section 9.3
User guide (quick start + per-test)	Yes	Section 10
Known issues and limitations log	Yes	Section 11
Backend source and Dockerfile	Yes	Section 3.5
API specification (OpenAPI/Swagger)	Yes	Section 4.2.2
Database schema	Yes	Section 4.2.1
iOS build (TestFlight or IPA)	Yes	Section 2.5

Submit everything. The winning solution goes to production. Every missing artifact is a question the production team will have to answer without you.