LifeTrack AI: A Local-First, Privacy-Preserving Mobile Health App with An On-Device Wellness Engine

Aryan Dalal[1], Sohan Bhangare[2], Manoranjan Paul[3], Souvik Routh[4], Prof. Deepali Shrikhande[5]
[1,2,3,4]*Department of Information Technology, Vidyalankar Institute of Technology, Wadala (E), Mumbai, India*
[5]*Project Guide, Department of Information Technology, Vidyalankar Institute of Technology, Wadala (E), Mumbai, India*

***Abstract*—Most health apps send sleep records mood notes, journal entries, meals and workouts to a server. That raises privacy concerns stops the app from working without a connection and costs money every time a model is called. This paper presents LifeTrack AI, an Android app built with React Native and Expo that takes a local-first approach. Everything the user records is stored in a SQLite database on the phone. The core features run without any network. LifeTrack AI handles check-ins (mood, sleep, water, steps and a journal) meal and macro tracking, workout planning, mindfulness sessions and weekly reports. On top of these it offers a 100-point Wellness Score, a calorie and macro calculator based on the Mifflin–St Jeor equation a workout-tier classifier and a rule-based coach whose messages are first screened for medical emergencies and eating-disorder distress. Users who want more can switch on an Express server that adds LLM coaching and natural-language meal estimates with a daily limit per user. To keep people coming back the app adds experience points, ranks, streaks that forgive a missed day daily missions and badges with a recovery service that restores workouts cut short by a crash. We describe the architecture, data model, algorithms and tools. Report what has been verified so far: a Jest test suite, for the mindfulness logic and a packet-capture audit that found no outgoing traffic while the on-device AI was running. We have not yet run a user study or any clinical validation. We outline how we plan to do both.**

***Index Terms*—mHealth; Local-First Architecture; On-Device AI; Privacy; Wellness Score; Health Coaching; Gamification; React Native; SQLite.**

I. INTRODUCTION

For people the phone has become the place where people keep track of their health. People note how well they slept, how water they drank, what they ate and which exercises they did and sometimes add a few private thoughts about how the day went. These small entries add up to a picture of a person's habits and wellbeing. Yet most health apps are built as thin front ends for a cloud service: the information goes up is stored else and is analysed there. Three problems follow from this design. Privacy depends on a vendor promises than on how the health app is built the health app stops being useful when the network is down and every model call carries a cost. Few health apps let people see what is happening to their data on their own device.

LifeTrack AI is our attempt to address all three problems and LifeTrack AI is a working Android app than a paper design. Profiles, daily logs, meals and workout data all live in an SQLite database on the device. A transparent rule-based engine calculates the Wellness Score. Suggests a workout tier while a coach answers questions using the users' own logs as context. Before any scoring or generation takes place a set of expressions checks the message for emergency or distress language. An optional Node.js server can add cloud LLM coaching and meal estimation. Lifetrack AI works perfectly well without it. A gamification layer, a crash-recovery service and a shared design system round out the app.

The rest of the paper follows the way LifeTrack AI was built. Section II covers background and earlier work and Section III sets out the gaps that motivated us and states the problem formally. Section IV walks through the nine modules we implemented. Section V compares LifeTrack AI with cloud-based, single-purpose and LLM-only alternatives. Section VI describes the tools we used and how we plan to evaluate the app Section VII reports what has been

verified far and Section VIII looks closely at how the scoring rules behave. We close with limitations and next steps, in Sections IX and X.

II. LITERATURE REVIEW AND TECHNICAL BACKGROUND

Our work touches areas: persuasive technology, gamification, sensor-based mental-health research and more recently, large language models used as coaches [1]– [3]. This section explains where LifeTrack AI sits among them.

A. Privacy in Cloud-Centric mHealth Apps

Studies of health and wellness apps keep showing that personal data leaves the phone in ways people do not expect. One assessment of accredited health apps found privacy weaknesses that had gone unaddressed even though the apps carried an accreditation [4]. Another study followed the network traffic of medicines-related apps. Found a large amount of data being shared with third-party services [5]. Findings like these pushed us towards a rule: by default, nothing leaves the device. Sensitive logs are written to local storage and the one optional network route is kept apart in its own server.

B. Personal Sensing, Mood and Mental Wellbeing

Work on sensing argues that everyday signals such as sleep, activity and self-reported mood say something real about mental health and that they can be used to give people timely feedback [2]. LifeTrack AI is not a tool but it borrows the idea. It records mood, energy, stress, sleep and activity side by side so that users can see in their reports how these signals move together. Mood and journal entrieare among the personal fields we store so they get exactly the same local-only treatment as everything else.

C. Behaviour Change, Gamification and Habits

Persuasive-technology research such as Foggs behaviour model holds that a behaviour happens when motivation, ability and a prompt come together at the moment [1]. Game elements such as points, ranks, badges and streaks are a way of supplying some of that motivation [7]. A literature review of studies found that gamification often helps but that the results depend heavily on context and on the use involved for that reason we treat these elements as a support for a check-in habit that should take two minutes a day and not as the main reason to use the app.

D. Rule-Based Engines versus LLM Coaching

Large language models can write personal coaching text [3] but using one directly inside a health app comes with costs. Responses are delayed each request has a price the app needs an internet connection. There is an added privacy risk because a third-party API handles the user's context. Rule-based engines sit at the end of the scale. They are predictable, fast and easy to audit. They struggle with natural language. We use both. A deterministic engine handles scoring, target calculation, safety screening and coaching on the phone. The optional LLM path is used for open-ended questions and in the current version the LLM never takes part in scoring or in any safety decision.

E. Nutrition and Energy Models

Daily energy targets use the Mifflin–St Jeor equation, which estimates resting energy use from weight, height, age and sex [8]. Total energy use comes from multiplying that figure by an activity factor. Our protein target of 1.6–2.2 g per kilogram of body weight matches evidence that adults who do resistance training benefit from intakes of about this size [9]. The sleep thresholds follow the 7–9-hour recommendation for adults [10]. The activity bands are our rough cut-offs, inspired by the World Health Organizations advice to stay active and sit less [11].

F. Summary

In short, we wanted to move from a pile of trackers to one local-first platform in which scoring, coaching and gamification share the same data. We are upfront about what's not in yet. Federated or on-device learning [12] a validated score and encryption, at rest are all future work.

III. RESEARCH GAPS AND PROBLEM FORMULATION

Based on the literature [2] [4]– [6] and on our use of the app during development we discovered five problems that typical wellness apps rarely solve together.

1) Data residency. By default, biometrics, journal entries and meal logs go to the cloud. Privacy then depends on what a company writes in its policy not

on how the app's built. Anyone who wants to track their health must accept that their data could be exposed.

2) Latency. When the app's intelligence lives in the cloud scoring and coaching do not work offline. Every use requires a network wait even though the daily routine should take about two minutes.
3) Fragmentation. Mood, sleep, water intake, food, workouts and mindfulness are usually spread across apps. As a result, no single score or coach can look at all of them. Give a full picture.
4) Safety handling. A chat-style coach may treat a message about chest pain, self-harm or an eating disorder, like any question. A strict safety check must run before anything
5) Resilience. Good habits do not last without feedback. If the phone crashes in the middle of a workout the app can lose the user's data.

To be more exact we describe a user's situation on day d as a tuple Ud = {P, Ld, Nd Wd}. (1)
Here P is the profile, which includes age, sex, height, weight, activity level and goal. Ld is the check-in data, made up of mood sleep hours, water in litres, steps, energy and stress. Nd represents the nutrition total, including calories, protein, carbohydrates, fats and fibre. Wd is the list of workouts completed in the seven days. The Wellness Score is calculated as:
$S(Ud) = 100 - \Sigma k\ \delta k(Ud)$
This means the Wellness Score starts at 100 and subtracts penalty points based on thresholds for each factor: sleep, hydration, mood, steps, calories and protein. These penalties are defined in Section IV-E.The coach receives a message q and the context Ud then returns a reply R(q, Ud). There's one rule: if the message q matches a safety pattern the scoring is. The reply is a referral. This is explained in Section IV-F.The privacy rule is straightforward. No part of Ud is ever sent off the device. The, on-device pipeline keeps everything. Only if the user enables the server route will any data be transmitted. Even then only the specific information needed for the request is sent. Nothing more.

IV. PROPOSED SYSTEM ARCHITECTURE AND MODULE DESIGN

LifeTrack AI is organized into five layers: presentation, domain services, on-device intelligence, data access and persistence (Fig. 1). LifeTrack AI runs on React Native 0.81.5 with Expo SDK 54 [13] [14]. LifeTrack AI contains nine modules, which I describe in Sections IV-A to IV-I: app start-up and recovery local storage, the check-in and dashboard nutrition the Wellness Score engine the AI coach, workouts and gamification the optional server and the overall pipeline.

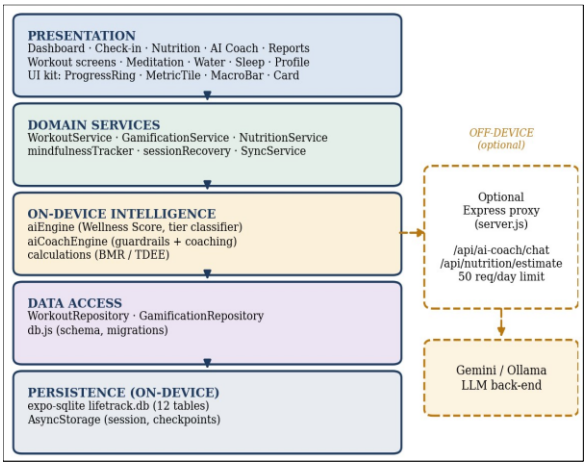

Fig. 1. Five-tier local-first architecture of LifeTrack AI. Solid components run on the device; the dashed proxy and LLM back end is optional.

A. App Start-Up, Navigation, and Crash Recovery

The main component, App.js launches the app in a fixed sequence. It keeps the splash screen on display while loading the Inter fonts. Then it runs initDatabase() to create the tables and apply any pending migrations. It checks AsyncStorage for a saved session to determine whether to show the Login screen or the Main screen. After that it calls check And Recover Orphaned Session () to look for workouts that were started but never finished. Once all these steps are complete does the splash screen disappear.
Navigation is managed using React Navigation 7. A root stack displays either the sign-in flow—comprising Login and Register—or the Main route. The Main route includes a bar with five tabs: Dashboard, Check-in AI Coach, Nutrition and Reports. Other screens like the water and sleep trackers workout screens, mood journal, meditation and profile are accessed from the dashboard or the tabs. When a workout is in progress the sessionRecovery service saves a checkpoint to AsyncStorage at intervals. If the app is closed unexpectedly the time it starts it detects the checkpoint

saves the workout to the database with the time recorded so far and removes the checkpoint. If the workout finishes properly the checkpoint is simply deleted.

B. Local-First Storage

All persistent data is stored in a SQLite file called lifetrack.db. This file is accessed through expo-sqlite inside the app's folder [16]. The database schema consists of 12 tables. Foreign keys are set to delete a user's logs when the user is removed. The ALTER TABLE commands used for migrations can be run times without causing errors. Compound indexes like idx_mindfulness_user_date help keep date-based queries fast. Table I summarizes the six tables. We use a repository pattern with classes like WorkoutRepository and GamificationRepository to keep SQL code separate, from the screens. A multi-set workout is saved in a transaction to ensure it is either fully stored or not stored at all.

TABLE I. CORE RELATIONAL ENTITIES OF THE ON-DEVICE DATABASE

Table	Key Fields	Purpose
users	id, email, gender, age, height, weight, activity_level, goal, target_calories/protein/carbs/fats/fiber	Authentication, biological profile and personalised nutrition targets
daily_logs	user_id, date (unique pair), mood, journal, sleep_hours, water_intake, steps, workout_done, meditation_done, energy_level, stress_level	One upserted check-in row per user per day
nutrition_logs	meal_name, calories, protein, carbs, fats, fiber, meal_time, source	Meal records; source is manual or ai_estimate
workout_sessions	day_name, title, duration_seconds, total_volume, xp_earned, notes	Completed training sessions and earned XP
workout_set_logs	session_id, exercise_name, set_number, weight_kg, reps, is_completed	Per-set weight and repetition records (cascade on delete)
mindfulness_sessions	start/end timestamp, duration_seconds, source, mood	Meditation, breathwork and manual sessions

I noticed that besides these six the schema also contains tables, for gamification state (user_gamification) schedules (workout_schedule_days) and planned exercises (workout_exercises). The app relies on the Android sandbox to keep apps from reading the database, which is a nice security measure. The app does not encrypt the database, which we list as work so we plan to add encryption later.

C. Daily Check-in and Dashboard

On the Check-in screen the user picks a mood (good neutral or bad). Sliders then set sleep from 0 to 14 hours and water from 0 to 5 liters. The user can also add steps toggle the workout and meditation switches rate energy and stress and write a journal note. Saving is an upsert (INSERT... ON CONFLICT (user_id, date) DO UPDATE) so editing the day overwrites the entry and never creates a duplicate. The Dashboard reads todays log todays calories and the current streak. It shows the Wellness Score in a progress ring, four tiles for sleep, water, steps and calories an AI tip card and a few quick-action buttons.

These screens are built from a set of simple components: PrimaryButton (with haptic feedback) Card and GradientCard, MetricTile, MacroBar, ProgressRing, Chip, Skeleton and SectionHeader. They receive everything through props. Contain no business logic. A theme file (theme.ts). Themecontext provide dark and light palettes with colors for sleep, water, steps and calories. The app opens in mode by default.

D. Nutrition Tracking and Targets

When a user registers or edits their profile calculations.js works out the metabolic rate (BMR) with the Mifflin–St Jeor equation,

$$BMR = 10W + 6.25H - 5A + s \quad (3)$$

where W is weight in kg H is height in cm A is age in years. S is a constant that depends on sex (+5 for men

−161 for women) [8]. That value is multiplied by an activity factor to give daily energy expenditure (TDEE) which is then adjusted for the user's goal: maintain, cut with a deficit or bulk with a surplus. NutritionService splits the calories across macronutrients setting protein between 1.6 and 2.2 g/kg [9] and saves the targets in the user's table.

The Nutrition screen groups meals into breakfast, lunch, dinner and snacks adds up the totals and shows four MacroBars (protein, carbohydrates, fats and fibre) against the targets. An "Estimate with AI" button sends a plain-text meal description to the server (Section IV-H). The returned numbers are stored with source = 'ai_estimate'. It always stays clear which values were typed in and which were estimated.

E. Composite Wellness Score Engine

The engine in aiEngine.js starts every day at a score of 100. Takes away points for each of six factors listed in Table II. The thresholds are based on published guidelines, for sleep [10] and physical activity [11]. They have not been fine-tuned on clinical data. The penalties are simply added together. Each one has a fixed value. The largest possible total deduction is 95 points. The final score always stays between and 100 (Fig. 2, Left).

TABLE II. WELLNESS SCORE PENALTY RULES

Factor	Condition	Deduction	Severe sleep deficit
Sleep	< 6.0 h	-20	Moderate sleep debt
Sleep	6.0 ≤ h < 7.0	-10	Likely dehydration
Hydration	< 1.5 L	-15	Sub-optimal fluid intake
Hydration	1.5 ≤ L < 2.5	-8	Low affect
Mood	Bad	-15	Baseline affect
Mood	Neutral	-5	Sedentary day
Steps	< 3,000	-15	Below activity target
Steps	3,000 ≤ n < 7,000	-8	Severe deficit
Calories	< 1,200 kcal	-20	Sub-optimal intake
Calories	1,200 ≤ c < 1,800 kcal	-10	Low protein intake
Protein	< 50 g	-10	Severe sleep deficit

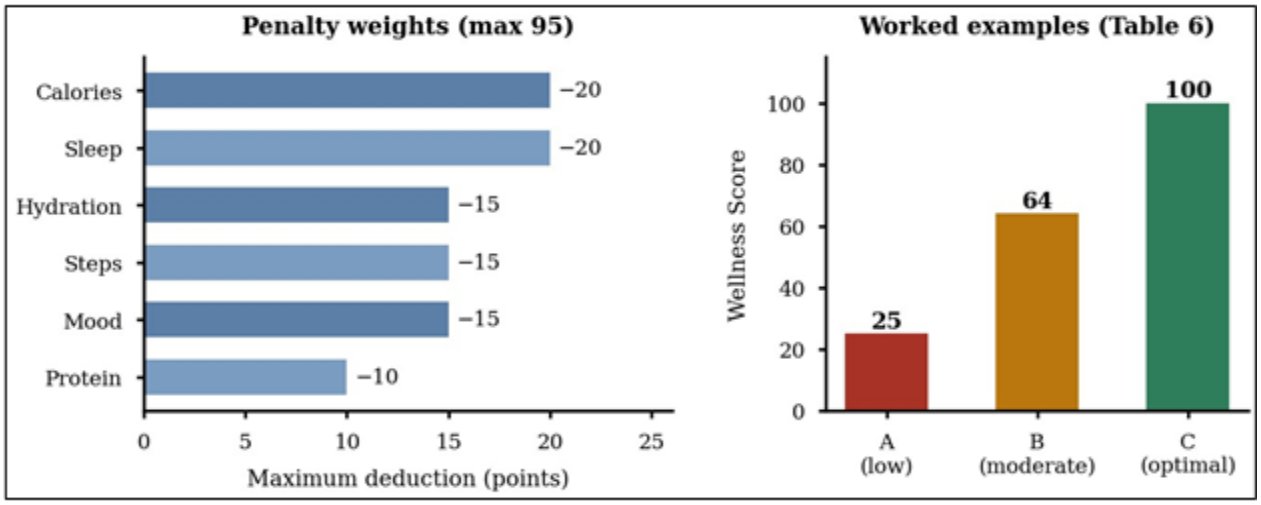

Fig. 2. Maximum deduction per Wellness Score factor (left) and scores of the three worked examples in Section VIII (right).

A second smaller classifier checks how workouts the user did in the last seven days (Fwk) and picks a training tier. If the user did than three sessions, they get the Beginner Foundation tier. That's a full-body program, three days a week. If they did three or four sessions, they get the Intermediate Conditioning tier. That's a /lower split done over four days. If they did five or more sessions, they get the Advanced tier. That's a push/pull/legs program, six days a week. In

the Workout Planner each tier is shown next, to an "AI Rebalance" button.

F. On-Device AI Coach and Safety Guardrails

Every message follows the path set in aiCoachEngine.js (Figure 3). First the text is checked against two groups of expressions. EMERGENCY_PATTERNS include events like chest pain, stroke passing out and heavy bleeding, as well as statements about self-harm and suicide. EATING_DISTRESS_PATTERNS include words that show problems, with eating. If either group matches, the process of scoring and creating a response is skipped. The user gets a recommendation to get help from an emergency service. A warning message stays on the screen.

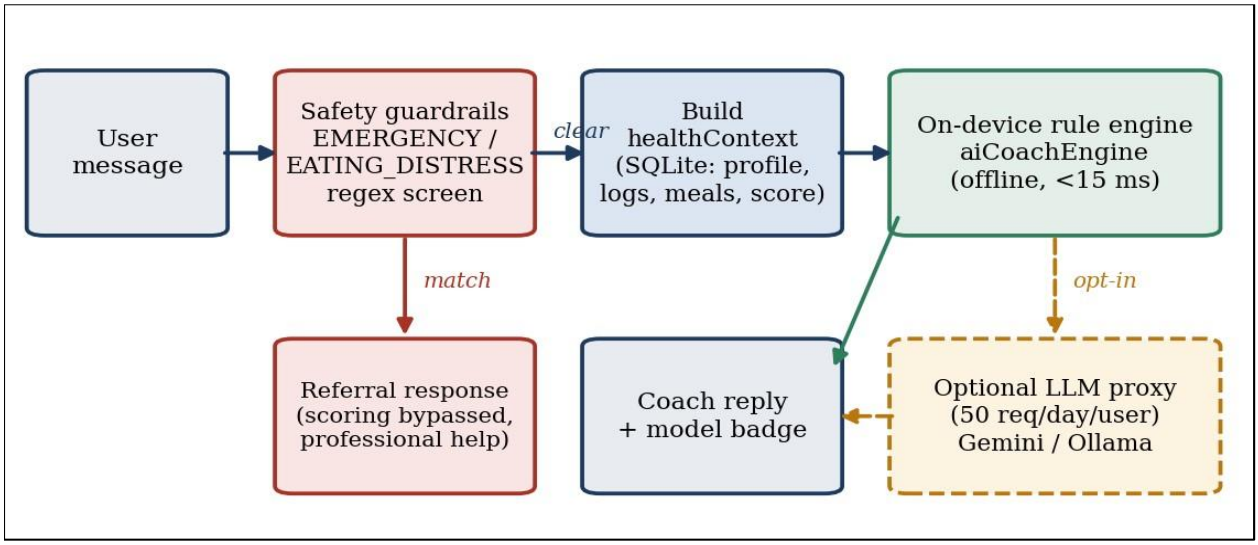

Fig. 3. Message routing in the AI coach: safety screen first, then context assembly and the on-device engine, with an optional LLM proxy.

If nothing fits the coach creates a context object using the users, daily_logs, nutrition_logs and fitness tables (profile todays log, meals and current score) and the rule engine gives an answer based on those details. The user is always shown which path led to the response either "On-Device Rule Engine" or the cloud model. The engine works in, then 15 milliseconds and does not need an internet connection so the Dashboard can update its AI tip every time it starts.

G. Workouts, Gamification, and Mindfulness

WorkoutService manages a workout from beginning to end. StartWorkoutSession fetches the exercises for the selected day. LogSetCompletion records the weight and repetitions. Starts the rest timer.

FinishWorkoutSession calculates the volume $V = \Sigma$ weight_kg × reps (4) over all completed sets awards 100 XP plus a bonus and sets daily_logs.workout_done to 1. The history and statistics screens show session volume estimated one-rep-max trends, personal records and muscle-group split.

Gamification Service turns XP into one of four ranks: Novice (0–499) Warrior (500–1,499) Titan (1,500–2,999) and Legend (3,000 and above). Gamification Service also checks running totals to unlock milestone badges such as "First Workout" and "Centurion". Daily missions, such as reaching a step goal or meditating for ten minutes add XP well. These are the game elements described by Deterding et al. [7]. We think of them as a support and not the main reason to use the app [6].

The mindfulness Tracker service accepts sessions between 10 seconds and 4 hours long. The mindfulness Tracker service splits a session that runs past midnight into two records. The

mindfulness Tracker service adds up the minutes for each day. Tracks streaks with a one-day grace rule so that a single missed day does not end the streak. The Meditation screen has a countdown timer, a Breathwork Exercise module for box (4-4-4-4) and 4-7-8 breathing with animated guidance, a way to record a session after the fact (ManualMindfulnessModal) and a mood question, at the end.

H. Optional LLM Server

For users who choose to enable cloud features a Node.js Express server runs on port 5000. Provides three endpoints. GET /api/health returns the status of the Node.js Express server. POST /api/ai-coach/chat forwards chat requests to a hosted or local language model such as Gemini or Ollama. POST /api/nutrition/estimate takes a plain-text description of food and returns estimated values for calories, protein, carbohydrates and fats.

The chat route limits each user to 50 requests per day. The limit is tracked in memory with a map that stores a user ID as the key and a (date count) pair, as the value and the count resets when the date changes. We deliberately kept this Node.js Express server from the main pipeline. It receives what a specific request needs and no scoring or safety decision depends on it. If the Node.js Express server cannot be reached the app carries on working as normal.

I. Pipeline Summary

Algorithm 1 brings together the modules, from Sections IV-A to IV-H starting from start-up moving through scoring and ending with the handling of a coaching message.

ALGORITHM 1: LOCAL-FIRST DAILY WELLNESS AND COACHING PIPELINE

Input: User u, date d, optional message q
Output: Wellness Score S, coach reply R
Const: K = {sleep, water, mood, steps, kcal, prot}
1: initDatabase(); $u \leftarrow$ checkAuth()
2: RecoverOrphanedSession(u) ▷ IV-A
3: $L \leftarrow$ SelectDailyLog(u, d)
4: $N \leftarrow$ AggregateNutrition(u, d) ▷ IV-D
5: $S \leftarrow 100$
6: for each factor k in K do
7: $S \leftarrow S -$ Penalty$_k(L, N)$ ▷ Table II
8: $tier \leftarrow$ ClassifyTier(F_{wk}) ▷ IV-E
9: if q is given then ▷ IV-F
10: if Match(EMERGENCY ∪ DISTRESS, q) then
11: return S, Referral(q)
12: $ctx \leftarrow$ BuildHealthContext(u, L, N, S)
13: $R \leftarrow$ LocalCoach(q, ctx)
14: if cloud and RateOK(u) then
15: $R \leftarrow$ ProxyCoach(q, ctx)
16: return S, R

V. SYSTEMATIC COMPARATIVE ANALYSIS

I look at Table III. It compares LifeTrack AI with three kinds of existing approach: mHealth apps, single-purpose trackers and LLM-only coaching assistants. The columns for these baselines show trends, within each category. These columns are not measurements of any product.

TABLE III. HOW LIFETRACK AI DIFFERS FROM CLOUD-CENTRIC, SINGLE-PURPOSE AND LLMONLY APPROACHES

Dimension	Cloud-Centric mHealth	Single-Purpose Tracker	LLM-Only Coach	LifeTrack AI
Data residency	Records typically stored on vendor servers	Often local but siloed	Prompts and context sent to a third-party API	All logs in on-device SQLite; cloud path optional
Offline operation	Limited	Usually available	Not available	Full, including scoring and coach
Coaching intelligence	Server-side models	Charts only, little advice	Generative; grounded only if context is supplied	Rule engine grounded in live logs; optional LLM

Safety handling	Varies by vendor	Not applicable	Model-dependent	Regex guardrails run before scoring or generation
Scope of tracking	Often one or two domains	Single domain	Conversation only	Mood, sleep, water, nutrition, workouts, mindfulness
Engagement mechanics	Varies	Minimal	None built in	XP, ranks, forgiving streaks, missions, badges
Failure resilience	Server-dependent	Varies	Network-dependent	Orphaned-session recovery; idempotent migrations

VI. IMPLEMENTATION SPECIFICATIONS AND EVALUATION FRAMEWORK

A. Technical Stack

Table IV lists the technologies used in the current system. Features we only plan to add (Section X) are not included.

TABLE IV. TECHNOLOGY STACK OF THE IMPLEMENTED SYSTEM

Component	Technology Used
Runtime	React Native 0.81.5 with Expo SDK 54; Android 14 / standalone APK
Navigation	React Navigation 7 (stack and bottom tabs)
Local persistence	expo-sqlite (lifetrack.db); @react-native-async-storage/async-storage
UI and graphics	Lucide icons; react-native-svg progress rings; react-native-chart-kit line and bar charts; expo-haptics; Inter font family
On-device intelligence	aiEngine.js (Wellness Score, tier classifier); aiCoachEngine.js (guardrails, coaching); calculations.js (BMR, TDEE, macros)
Services and data access	WorkoutService, GamificationService, NutritionService, mindfulnessTracker, sessionRecovery, SyncService; WorkoutRepository, GamificationRepository
Optional back end	Node.js and Express (port 5000); in-memory rate limiter; Gemini 3.7 Flash or Ollama
Testing	Jest unit tests; Wireshark packet capture for the traffic audit

B. Evaluation Scope

We look at the system from two angles. The mindfulness service, which limits how long a session can last splits sessions that cross midnight. Allows a streak to continue even if a day is missed is tested by a Jest suite, in tests/mindfulnessTracker.test.js. The local-first claim is examined by running a packet-capture audit while the on-device AI is working. Section VII shows the results of both tests. The scoring, nutrition and workout logic is described in full in Sections IV-D to IV-G so Section VIII instead shows these ideas through step-by-step examples.

C. Planned Evaluation Metrics

Beyond what has been checked we want to test the app in four more areas.

1) Speed. We want to see how long it takes for the Wellness Score and the rule-engine response to appear. We also want to measure how long the app takes to start up including database setup. This will be tested on a range of Android phones.
2) Privacy. We will do repeated packet-capture checks on every screen. We will test with and, without the server. We need to make sure that only the data the user chose to send actually leaves the device.
3) Engagement. We will use a System Usability Scale questionnaire. We will also measure how a checking takes. We will track participation over seven and thirty days with a group of users.
4) Score validity. We need to check how the Wellness Score matches expert opinion. We also need to check how well it matches proven self-report measures of wellbeing. Without this we cannot claim that the thresholds are set correctly.

VII. EVALUATION STATUS

Section IV describes the app that now runs fully on an Android 14 target. The app lets a user register receive

calculated nutrition targets submit check-ins log meals, plan and run workouts, complete meditation and breathing session's view reports and chat with the on-device coach. When the optional server is active the app can also perform cloud estimation and coaching. Table V lists the five verification scenarios recorded for the project and the app passed each of them.

TABLE V. VERIFICATION SCENARIOS AND OUTCOMES

Scenario	Method	Assertion	Outcome
Duration minimum	isValidSessionDuration(5)	Rejects sessions under 10 seconds	PASS
Duration maximum	isValidSessionDuration(15000)	Rejects sessions longer than 4 hours	PASS
Midnight partition	partitionSessionByDay(start, end)	Splits an overnight session into two daily logs	PASS
Streak grace day	calculateStreakWithForgiveness(date s)	Keeps a streak alive when one day is missed	PASS
Zero-leakage audit	Wireshark packet capture	No data transmitted during on-device AI runs	PASS

These results back two points: the mindfulness rules act as expected at their limits and the, on-device AI shows no network traffic during the tests we ran. These results do not reveal how simple the app feels to use how long people keep using it or what the Wellness Score actually means from a perspective. We have not conducted a user study. We have not examined how fast the app works on various phones. Because of the missing data we explain what the system does and how we intend to verify it. We have decided not to provide any predicted outcomes.

VIII. SCORE SENSITIVITY AND WORKED EXAMPLES

Because the Wellness Score is a calculation that uses six inputs, I can examine exactly how the Wellness Score behaves. Table VI applies the rules, from Table II to three days and I worked these out by hand from the thresholds so the numbers do not come from real users.

TABLE VI. WORKED WELLNESS SCORE COMPUTATION FOR THREE HYPOTHETICAL DAYS (ILLUSTRATIVE, NOT MEASURED)

Factor	A (low)	B (moderate)	C (optimal)
Sleep	5.5 h → −20	6.5 h → −10	8.0 h → 0
Hydration	1.2 L → −15	2.0 L → −8	3.0 L → 0
Mood	Neutral → −5	Good → 0	Great → 0
Steps	2,500 → −15	5,000 → −8	9,000 → 0
Calories	1,500 kcal → −10	1,700 kcal → −10	2,100 kcal → 0
Protein	45 g → −10	60 g → 0	120 g → 0
Wellness Score	25	64	100

A few things stand out. Calories each take away 20 points if not met. Nutrition overall. Which includes calories and protein. Makes up 30 of the 95 penalty points. That means even if someone sleeps well drinks water and walks a lot, they can still lose 30 points just from not eating enough. Mood is the part that's based on how you feel and it can change the score by up to 15 points. So, one bad mood day alone won't bring the score down by itself.

The score never goes to zero. That means it can't tell the difference between a day and a day with no data. Unless we handle missing data in a clear way. That last point is something we want to test not something we've already proven. Now the daily logs table sets all number columns to zero by default. So, if someone doesn't enter anything for a day and we score it without checking the app would still count penalties for sleep, water and steps. Even though those were never recorded.

We think people will trust the score more if the app treats "no data" as different, from "zero value." The usability study we plan to run described in Section VI-C is meant to check if that's true.

IX. LIMITATIONS

We want everyone to understand where the current system ends. The thresholds and values used in the Wellness Score were chosen by hand based on published guidance and not on a dataset of people's wellbeing. The Wellness Score is therefore a signal, not something that has been calibrated or tested for medical use and the app does not offer medical advice. The safety checks are made up of expressions for English. The safety checks can miss ways of saying the same thing, as well as phrases in other languages and the safety checks can also fire on words that are not a problem. The safety checks are a line of protection but not a substitute for help from a real professional. The database relies on the Android sandbox. Is not encrypted by the app. Sign-in is done on the device. If a phone is lost or stolen the current design offers no protection. The optional server keeps its rate limits in memory. The rate limits reset whenever the server restarts and do not work across multiple copies. Finally, the testing in Section VII is limited. The testing in Section VII consists of a set of tests for the mindfulness module and a packet audit. There are no studies with users no tests across devices and no data on long-term use. What we present here is the structure of the system and the plan, for checking it not results.

X. CONCLUSION AND FUTURE WORK

We introduced LifeTrack AI, a health app that keeps all records stored locally in an on-device SQLite database. With this local storage it still offers scoring, coaching and engagement features. The app includes check-ins, nutrition tracking, workout logging and mindfulness sessions. It features a 100-point Wellness Score uses the Mifflin–St Jeor equation to calculate calorie targets and classifies workouts into tiers based on intensity. A rule-based coach operates behind safety filters to ensure guidance stays safe and appropriate. There's also a game- layer to keep users engaged. The app has built-in crash recovery and an optional server that limits requests for cloud-based LLM features. Unlike health apps that rely on the cloud LifeTrack AI works fully offline—no internet connection is needed for any of its core features. The next steps naturally follow from the limitations. In the term we will run a pilot user study to evaluate usability and how well people stay engaged with the app. We will test performance speed across a range of devices handle missing data clearly in the scoring system and expand safety patterns to include more phrasings and languages. Later we plan to add encrypted storage to protect data better adjust scoring weights using validated wellbeing instruments and allow data import from wearables and health platforms. We also want to explore on-device or federated learning to personalize advice without sending raw user data to a server [12]. Because the app deals with personal data any move toward cloud-based features must be reviewed against data protection laws, like Indias Digital Personal Data Protection Act, 2023 [17]. In the term LifeTrack AI could offer user-controlled data exports and a human-reviewed library of coaching content that works alongside the rule-based engine.

REFERENCES

[1] B. J. Fogg, "A behavior model for persuasive design," in *Proc. 4th Int. Conf. Persuasive Technology*, 2009, Art. no. 40.

[2] D. C. Mohr, M. Zhang, and S. M. Schueller, "Personal sensing: Understanding mental health using ubiquitous sensors and machine learning," Annual Review of Clinical Psychology, vol. 13, pp. 23–47, 2017.

[3] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023.

[4] K. Huckvale, J. T. Prieto, M. Tilney, P.-J. Benghozi, and J. Car, "Unaddressed privacy risks in accredited health and wellness apps: A cross-sectional systematic assessment," BMC Medicine, vol. 13, Art. no. 214, 2015.

[5] Q. Grundy, K. Chiu, F. Held, A. Continella, L. Bero, and R. Holz, "Data sharing practices of medicines related apps and the mobile ecosystem: Traffic, content, and network analysis," BMJ, vol. 364, Art. no. l920, 2019.

[6] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? A literature review of empirical studies on gamification," in Proc. 47th Hawaii Int. Conf. Syst. Sci., 2014, pp. 3025–3034.

[7] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining 'gamification'," in Proc. 15th Int. Academic MindTrek Conf., 2011, pp. 9–15.

[8] M. D. Mifflin, S. T. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty, and Y. O. Koh, "A new predictive equation for resting energy expenditure in healthy individuals," The American Journal of Clinical Nutrition, vol. 51, no. 2, pp. 241–247, 1990.

[9] R. W. Morton et al., "A systematic review, meta-analysis and meta-regression of the effect of protein supplementation on resistance training-induced gains in muscle mass and strength in healthy adults," British Journal of Sports Medicine, vol. 52, no. 6, pp. 376–384, 2018.

[10] M. Hirshkowitz et al., "National Sleep Foundation's sleep time duration recommendations: Methodology and results summary," Sleep Health, vol. 1, no. 1, pp. 40–43, 2015.

[11] World Health Organization, WHO Guidelines on Physical Activity and Sedentary Behaviour. Geneva, Switzerland: WHO, 2020.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS), PMLR, vol. 54, 2017, pp. 1273–1282.

[13] Meta Platforms, "React Native Documentation," 2025.

[14] Expo, "Expo SDK Documentation," 2025. (Cross-platform React Native toolchain, including the expo-sqlite module.)

[15] Google, "Material Design: Accessibility and touch target guidelines," 2024.

[16] SQLite Consortium, "SQLite Documentation," 2025. (Embedded relational database engine.)

[17] Ministry of Electronics and Information Technology, Government of India, "The Digital Personal Data Protection Act, 2023," 2023.