ANI-MAX: AN AI-ENHANCED MOBILE TOOL FOR REAL-TIME CAT AND DOG HEALTH MONITORING AND EARLY RISK PREVENTION USING ENVIRONMENTAL AND VITAL SIGN ANALYSIS

Haoqun Qin [1] , Miguel Angel Cai Lane [2]

[1] Albany Acedemy, 135 Academy Rd, Albany, NY 12208
[2] California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Ani-Max is a mobile platform designed to prevent health crises in cats and dogs by combining wearable vital sign tracking with environmental risk analysis [1]. Pets can't voice discomfort, and subtle issues like early dehydration or heat stress often go unnoticed until they escalate. Ani-Max addresses this by continuously monitoring heart rate, blood pressure, activity, temperature, and humidity, using AI to establish a personalized baseline and detect anomalies early. Challenges included maintaining sensor accuracy during movement, ensuring reliable environmental data, and optimizing battery life. In testing, the system performed well at rest but showed accuracy drops during intense activity, and risk scoring declined when weather data was delayed. These insights informed us of improvements like motion filtering and redundant environmental inputs. Ani-Max offers a proactive, real-time health monitoring solution that helps owners act before issues become emergencies, ultimately improving pet well-being and owner peace of mind [2].

KEYWORDS

Cat and Dog health Monitoring, AI-Enhanced application, Early risk prevention, Vital sign analysis, Real-time health alerts

1. INTRODUCTION

In recent years, dog and cat ownership has been rising in most of the world. Cats and dogs have been the most common companion animals. According to the American Pet Products Association, 65 million households in the United States own at least one dog, and 46 million households own a cat. Since more pets are being treated as family members, owners are willing to allocate time and money towards the health of cats and dogs. Cats and dogs cannot articulate discomfort, illness, or stress, so catching early health issues is very difficult. We may not even notice some subtle changes in their body, such as blips in blood pressure, heart rate or dehydration until suddenly they become a significant symptom.

Risks also include environmental factors. Extreme temperatures, for example, can be fatal for pets. Over the course of 2025, heat waves in the United States resulted in over 110 documented pet deaths (mid-year) which were predominantly healthy animals that had been left outside or in areas with little ventilation and subjected to high temperatures exceeding 32 °C where canine

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 165-177, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160814

death rates jump by nearly 10% and mortality increased for every degree beyond 25°C [3]. Clinics in Bangalore report that there has been a 50% spike in dehydration-related cases during peak summer months over the years.

Although current pet monitoring solutions offer basic GPS tracking or step counts, few provide integrated approaches such as ones that combine vital sign tracking with environmental condition analysis to prevent health crises [4]. This gap highlights the need for an AI-enhanced tool such as Ani-Max—capable of monitoring physiological health indicators in real time, assessing environmental risks, and providing early alerts to prevent illness or injury.

The first methodology reviewed a remote vital sensing framework combining wearable sensors, infrared thermography, radar, and computer vision for animal health monitoring. While effective in controlled settings, it struggled with species-specific calibration and environmental noise. Ani-Max addresses this by focusing on cats and dogs, applying AI personalization, and integrating environmental context for more accurate, real-world monitoring.

The second methodology used contactless heart rate monitoring via imaging photoplethysmography (iPPG) on pet facial videos [5]. It achieved high accuracy in stationary, well-lit environments but failed during movement or poor visibility. Ani-Max improves on this by using wearable sensors capable of maintaining accuracy during activity, both indoors and outdoors.

The third methodology developed a canine health score from wearable activity data to detect abnormal behavior. While useful for movement-based alerts, it ignored vital signs and environmental risk factors. Ani-Max unifies activity, physiological, and environmental data in a single AI-generated risk score for comprehensive prevention.

The proposed solution is Ani-Max, an AI-enhanced mobile application designed to provide real-time health monitoring and early risk prevention for cats and dogs [6]. It uses continuous vital sign tracking and environmental conditions to analyze health levels.
Ani-Max integrates data from wearable sensors with AI-powered analytics to address the challenge of undetected pet health issues. These sensors record critical physiological parts such as heart rate, blood pressure, and activity levels, also capturing environmental data such as temperature, humidity, and air quality. The AI algorithms of the application then process the data to identify the health status of the pets and potential health risks. This can generate alerts that notify owners before conditions turn into emergencies. For example, if Ani-Max detects an elevated heart rate in a dog during a heatwave, the system can alert the owner to use immediate cooling measures before it gets worse.

This approach is effective because it combines continuous, non-invasive monitoring with predictive analytics. It enables proactive care rather than reactive treatment. Unlike basic GPS or activity trackers, Ani-Max's multi-sensor fusion allows owners to have an overall view of a pet's well-being, depending on both internal health indicators and external environmental threats. This design can significantly reduce the condition of detecting the health problem of the pets only when it becomes a symptom.

Moreover, Ani-Max improves existing solutions by using cloud-based data storage and historical trend analysis to enable veterinarians and owners to review long-term patterns and make more accurate care decisions. By integrating preventive health monitoring into a single and accessible platform, Ani-Max offers a superior method for safeguarding pet health compared to single-function devices, ultimately enhancing both lifespan and quality of life for cats and dogs.

In Experiment A, I tested how accurately Ani-Max's wearable sensors measured heart rate under different activity levels—resting, walking, and running—by comparing results to a veterinary-grade ECG. The goal was to see if movement significantly reduced accuracy. As expected, readings were most accurate at rest, with errors increasing during higher activity, especially running, due to motion noise. In Experiment B, the study evaluated how delays or gaps in environmental data affected the accuracy of the AI risk score. Using consistent physiological data, I compared results with accurate, slightly delayed, and majorly delayed weather inputs against veterinarian assessments. Accuracy dropped sharply when data was missing or outdated, with heat-related risks being most affected. These outcomes made sense—physical motion challenges, and incomplete environmental context weakens AI decision-making. Both experiments highlighted exactly where the system is strong and where it needs technical improvements.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Ensuring Privacy and Security of Pet Health Data

Ani-Max collects sensitive health and location data. This makes privacy and security critical concerns. Unauthorized access to information like this could lead to misuse of or can reduce owner trust. To address this, I could implement end-to-end encryption for data transmission, secure cloud storage with multi-factor authentication, and strict compliance with data protection regulations such as GDPR. To enhance transparency and trust in the system, it could provide users with clear data consent options and share the ability to control how their pet's information is stored.

2.2. Improving Reliability of Environmental Data Integration

Ani-Max integrates real-time environmental data such as temperature, humidity, and air quality into the system. This poses challenges due to potential gaps or inaccuracies in publicly available sources. For example, inconsistent network coverage or outdated weather station updates can lead to delays in risk detection. To address this, I could combine multiple data sources, such as local weather APIs, IoT environmental sensors, and user-reported conditions. This way, an AI-based validation system could cross-check data for anomalies, which ensures that environmental risk alerts are both timely and reliable.

2.3. Enhancing Accuracy of Wearable Pet Health Sensors

Another primary challenge is ensuring that wearable sensors accurately capture physiological data such as heart rate and blood pressure in cats and dogs. Variations in fur density, body size, and movement can interfere with readings. Which potentially produces false positives or negatives. To address this, I could use high-sensitivity, pet-specific sensors combined with advanced signal filtering algorithms to reduce noise and improve accuracy. In addition, calibration protocols tailored to different breeds and sizes could also help maintain reliability over time, ensuring consistent monitoring regardless of environmental or behavioral variables.

3. SOLUTION

Ani-Max is a cross-platform mobile application that integrates three parts: (1) User Authentication and Pet Profile Management, (2) Health and Environmental Data Ingestion, and (3) AI Risk Scoring and Alerts. The mobile client enables owners to register, authenticate, and

create profiles for cats or dogs according to species. Each profile stores age, sex, breed, weight, and optional medical notes to support individualized thresholds.

Data ingestion occurs through a secure Bluetooth link to a wearable that delivers physiological signals like heart rate, estimated blood pressure, activity level at configurable intervals. Also, the application retrieves environmental context—temperature, humidity, air-quality index, and heat-risk indicators—from trusted APIs and, where available, nearby IoT sensors [7]. All measurements are recorded with precise timestamps, standardized through normalization, and stored in a cloud-based database to ensure data persistence and facilitate longitudinal analysis.

The AI layer integrates rule-based safety protocols with statistical and machine-learning algorithms. During an initial calibration phase, the system establishes a personalized baseline; subsequent measurements are evaluated against this baseline and standardized veterinary reference ranges. A dynamic risk score is calculated in real time, incorporating trend analysis (e.g., elevated heart rate under high wet-bulb temperature conditions), recent activity patterns, and breed/age-specific adjustments.

If predefined thresholds are breached, Ani-Max generates tiered alerts (advisory, caution, or urgent) accompanied by concise, actionable recommendations. The dashboard provides real-time status updates, trend visualizations, and contextual insights, while a historical data module allows for exportable summaries to facilitate veterinary evaluation.
Security measures include end-to-end data encryption, role-based access controls for pet health records, and auditable consent management. This framework ensures seamless, proactive monitoring with minimal owner burden and streamlined clinical integration.

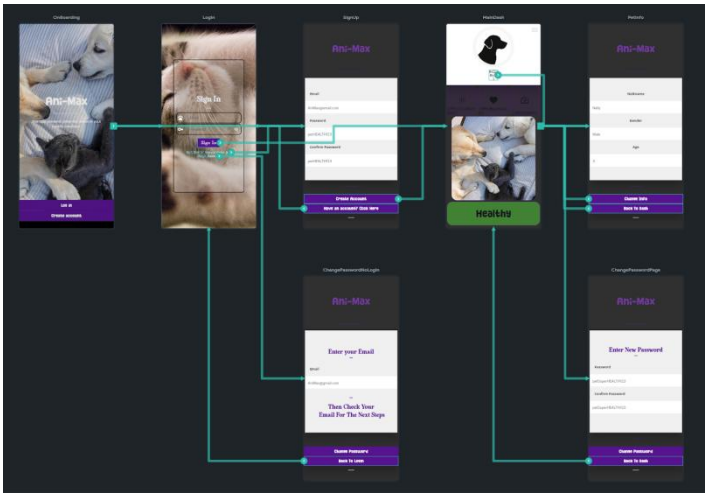


Figure 1. Overview of the solution

The User Authentication and Profile Management component ensures secure account access and accurate pet data storage. It could use an authentication service with email verification, encrypted password storage, and password reset features. This component enables personalized health thresholds by linking each pet's physiological and environmental data to its own profile.

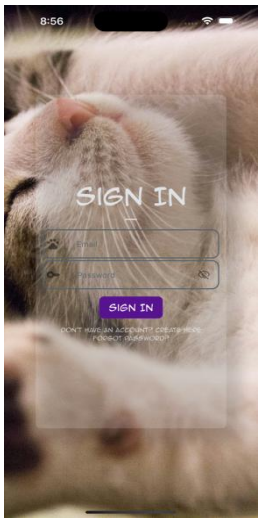


Figure 2. User Authentication Interface for Ani-Max Mobile Application

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
final FirebaseAuth _auth = FirebaseAuth.instance;
final FirebaseFirestore _db = FirebaseFirestore.instance;

/// Creates an account, then creates the user document in Firestore.
Future<UserCredential> signUpWithEmail({
required String email,
required String password,
String? displayName,
}) async {
final cred = await _auth.createUserWithEmailAndPassword(
email: email.trim(),
password: password,
);

final uid = cred.user!.uid;

// Optional: set display name in FirebaseAuth profile
if (displayName != null && displayName.trim().isNotEmpty) {
await cred.user!.updateDisplayName(displayName.trim());
}

// Create user record in Firestore
await _db.collection('users').doc(uid).set({
'email': email.trim(),
'displayName': displayName?.trim(),
'createdAt': FieldValue.serverTimestamp(),
'updatedAt': FieldValue.serverTimestamp(),
}, SetOptions(merge: true));

return cred;
}

/// Adds a pet profile under the authenticated user.
Future<void> createPetProfile({
required String petName,
required String species, // "dog" or "cat"
required String breed,
required double weightKg,
required int ageYears,
String? notes,
}) async {
final user = _auth.currentUser;
if (user == null) {
throw StateError('User not logged in');
}

final petDoc = _db
.collection('users')
.doc(user.uid)
.collection('pets')
.doc(); // auto-ID

await petDoc.set({
'petId': petDoc.id,
'petName': petName.trim(),
'species': species.trim().toLowerCase(),
'breed': breed.trim(),
'weightKg': weightKg,
'ageYears': ageYears,
'notes': notes?.trim(),
'createdAt': FieldValue.serverTimestamp(),
'updatedAt': FieldValue.serverTimestamp(),

// Baseline placeholders (calibration can update these later)
'baseline': {
'restingHeartRateBpm': null,
'restingBpEstimate': null,
'calibratedAt': null,
},
});
}

Future<void> signInWithEmail({
required String email,
required String password,
}) async {
await _auth.signInWithEmailAndPassword(
email: email.trim(),
password: password,
);
}

Future<void> signOut() => _auth.signOut();
}
```

Figure 3. User Registration and Pet Profile Creation Function Implemented with Firebase Authentication

The code shown represents the user registration and pet profile creation process. This function executes when a new user signs up or adds a new pet. First, the system calls the create User With Email And Password() method provided by Firebase Authentication. This method securely registers user credentials and returns a unique user ID (UID) [8]. The UID acts as a primary identifier in the database.

Next, the program creates a structured document in Firestore under the “users” collection. Within this document, pet profile data such as name, species, breed, weight, and age are stored as key-value pairs. These attributes are critical because they later inform the AI model’s personalized thresholds.

Variables defined include uid, email, petName, breed, and weight. The backend stores this information in encrypted cloud storage. The database rules enforce role-based access control, meaning only authenticated users may access their pet’s data. This component runs at the start of the application flow and establishes the foundation for secure and personalized monitoring.

The Data Ingestion component collects physiological data from wearable sensors and environmental data from APIs. It relies on Bluetooth Low Energy (BLE) communication and RESTful API calls [9]. This module standardizes incoming data streams and synchronizes timestamps before storing them in the cloud database.

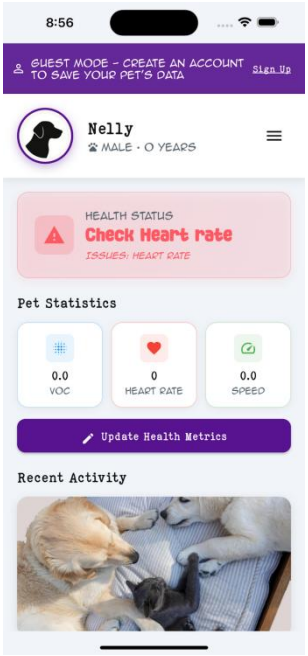
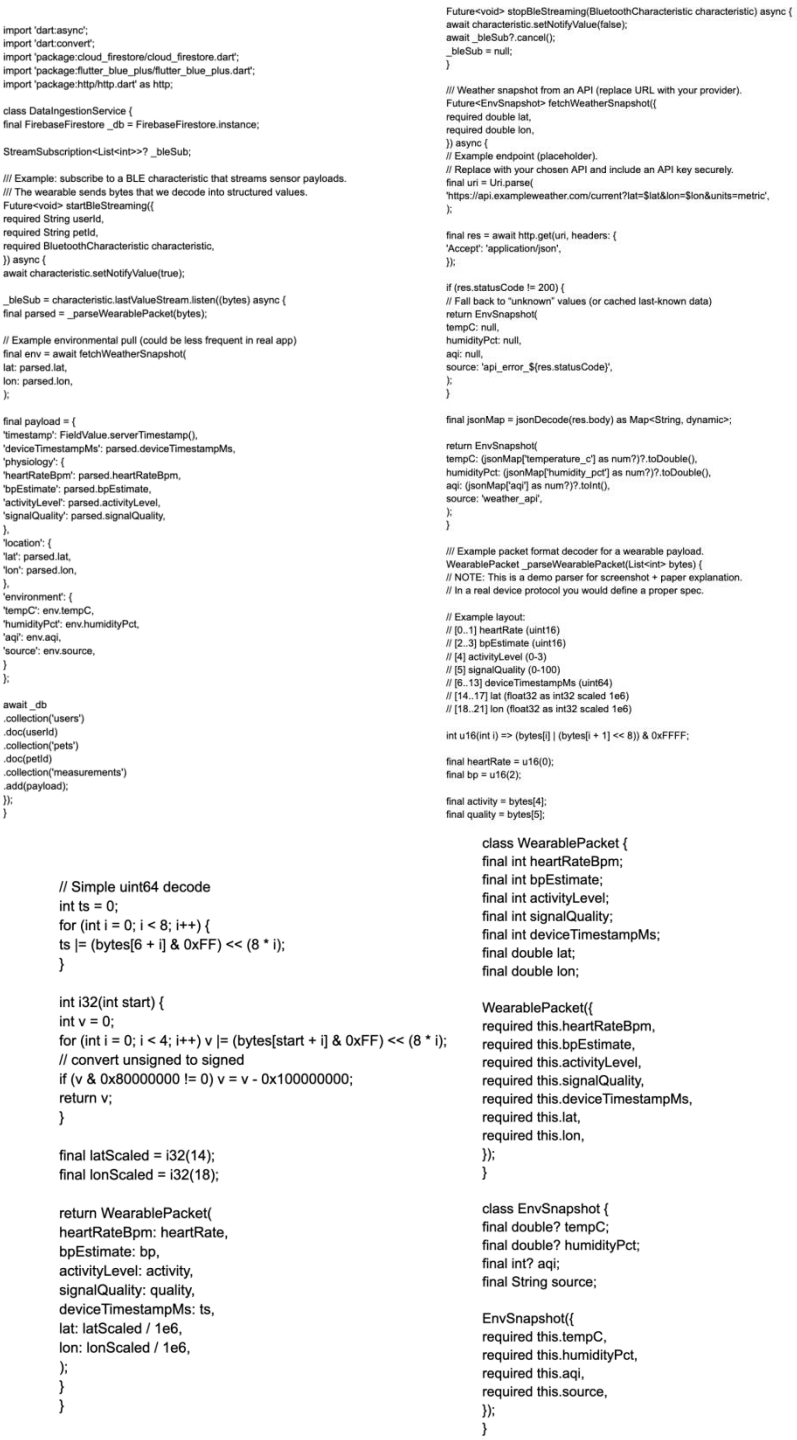


Figure 4. Ani-Max Application Main Dashboard Displaying Pet Health Monitoring Interface



```
import 'dart:async';
import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter_blue_plus/flutter_blue_plus.dart';
import 'package:http/http.dart' as http;

class DataIngestionService {
final FirebaseFirestore _db = FirebaseFirestore.instance;

StreamSubscription<List<int>>? _bleSub;

/// Example: subscribe to a BLE characteristic that streams sensor payloads.
/// The wearable sends bytes that we decode into structured values.
Future<void> startBleStreaming({
required String userId,
required String petId,
required BluetoothCharacteristic characteristic,
}) async {
await characteristic.setNotifyValue(true);

_bleSub = characteristic.lastValueStream.listen((bytes) async {
final parsed = _parseWearablePacket(bytes);

// Example environmental pull (could be less frequent in real app)
final env = await fetchWeatherSnapshot(
lat: parsed.lat,
lon: parsed.lon,
);

final payload = {
'timestamp': FieldValue.serverTimestamp(),
'deviceTimestampMs': parsed.deviceTimestampMs,
'physiology': {
'heartRateBpm': parsed.heartRateBpm,
'bpEstimate': parsed.bpEstimate,
'activityLevel': parsed.activityLevel,
'signalQuality': parsed.signalQuality,
},
'location': {
'lat': parsed.lat,
'lon': parsed.lon,
},
'environment': {
'tempC': env.tempC,
'humidityPct': env.humidityPct,
'aqi': env.aqi,
'source': env.source,
}
};

await _db
.collection('users')
.doc(userId)
.collection('pets')
.doc(petId)
.collection('measurements')
.add(payload);
});
}

Future<void> stopBleStreaming(BluetoothCharacteristic characteristic) async {
await characteristic.setNotifyValue(false);
await _bleSub?.cancel();
_bleSub = null;
}

/// Weather snapshot from an API (replace URL with your provider).
Future<EnvSnapshot> fetchWeatherSnapshot({
required double lat,
required double lon,
}) async {
// Example endpoint (placeholder).
// Replace with your chosen API and include an API key securely.
final uri = Uri.parse(
'https://api.exampleweather.com/current?lat=$lat&lon=$lon&units=metric',
);

final res = await http.get(uri, headers: {
'Accept': 'application/json',
});

if (res.statusCode != 200) {
// Fall back to "unknown" values (or cached last-known data)
return EnvSnapshot(
tempC: null,
humidityPct: null,
aqi: null,
source: 'api_error_${res.statusCode}',
);
}

final jsonMap = jsonDecode(res.body) as Map<String, dynamic>;

return EnvSnapshot(
tempC: (jsonMap['temperature_c'] as num?)?.toDouble(),
humidityPct: (jsonMap['humidity_pct'] as num?)?.toDouble(),
aqi: (jsonMap['aqi'] as num?)?.toInt(),
source: 'weather_api',
);
}

/// Example packet format decoder for a wearable payload.
WearablePacket _parseWearablePacket(List<int> bytes) {
// NOTE: This is a demo parser for screenshot + paper explanation.
// In a real device protocol you would define a proper spec.

// Example layout:
// [0..1] heartRate (uint16)
// [2..3] bpEstimate (uint16)
// [4] activityLevel (0-3)
// [5] signalQuality (0-100)
// [6..13] deviceTimestampMs (uint64)
// [14..17] lat (float32 as int32 scaled 1e6)
// [18..21] lon (float32 as int32 scaled 1e6)

int u16(int i) => (bytes[i] | (bytes[i + 1] << 8)) & 0xFFFF;

final heartRate = u16(0);
final bp = u16(2);

final activity = bytes[4];
final quality = bytes[5];

// Simple uint64 decode
int ts = 0;
for (int i = 0; i < 8; i++) {
ts |= (bytes[6 + i] & 0xFF) << (8 * i);
}

int i32(int start) {
int v = 0;
for (int i = 0; i < 4; i++) v |= (bytes[start + i] & 0xFF) << (8 * i);
// convert unsigned to signed
if (v & 0x80000000 != 0) v = v - 0x100000000;
return v;
}

final latScaled = i32(14);
final lonScaled = i32(18);

return WearablePacket(
heartRateBpm: heartRate,
bpEstimate: bp,
activityLevel: activity,
signalQuality: quality,
deviceTimestampMs: ts,
lat: latScaled / 1e6,
lon: lonScaled / 1e6,
);
}
}

class WearablePacket {
final int heartRateBpm;
final int bpEstimate;
final int activityLevel;
final int signalQuality;
final int deviceTimestampMs;
final double lat;
final double lon;

WearablePacket({
required this.heartRateBpm,
required this.bpEstimate,
required this.activityLevel,
required this.signalQuality,
required this.deviceTimestampMs,
required this.lat,
required this.lon,
});
}

class EnvSnapshot {
final double? tempC;
final double? humidityPct;
final int? aqi;
final String source;

EnvSnapshot({
required this.tempC,
required this.humidityPct,
required this.aqi,
required this.source,
});
}
```

Figure 5. Bluetooth Data Listener and Weather API Retrieval Functions for Data Ingestion Module

The displayed code includes two core methods: a Bluetooth data listener and a weather API retrieval function. The BLE listener continuously scans for signals broadcast by the wearable device. When a packet is received, the system parses the raw byte stream into structured variables such as heartRate, activityLevel, and signalStrength.

Simultaneously, the fetch Weather Data() function sends an HTTP GET request to a weather API endpoint using geographic coordinates. The response includes temperature, humidity, and air quality index values. These values are extracted from the JSON response and stored in local variables.

After both data streams are collected, the system merges them into a unified data object that includes timestamps. The merged dataset is uploaded to Firestore for persistent storage [10]. This code executes continuously while the application runs in monitoring mode. The backend database stores historical data to enable trend analysis. This ingestion layer ensures the AI module receives synchronized physiological and environmental inputs for accurate risk assessment.

The AI Risk Scoring component evaluates incoming data and determines whether intervention is required [15]. It combines rule-based thresholds with machine-learning trend detection. This module uses statistical deviation analysis to compare real-time data against personalized baselines and veterinary reference ranges.

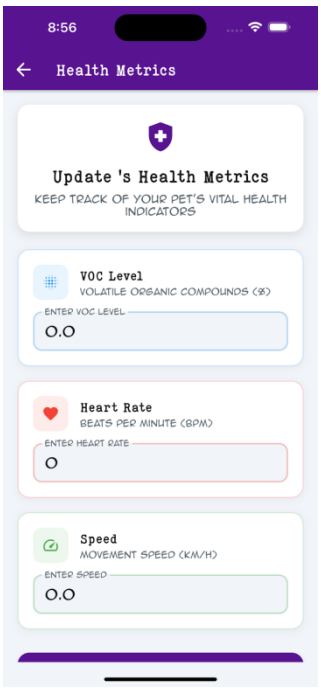


Figure 6. Ani-Max Mobile Application Interface for Real-Time Health Monitoring

```
import 'package:cloud_firestore/cloud_firestore.dart';

enum AlertLevel { advisory, caution, urgent }

class RiskEngine {
final FirebaseFirestore _db = FirebaseFirestore.instance;

/// Computes a 0–100 risk score using physiology + environment + baseline.
/// Called whenever a new measurement is uploaded (or on a periodic timer).
Future<RiskResult> calculateRiskScore({
required String userId,
required String petId,
required int heartRateBpm,
required int bpEstimate,
required int activityLevel,
required int signalQuality,
required double? tempC,
required double? humidityPct,
required int? aqi,
}) async {
// Load baseline (calibration)
final petDoc = await _db.collection('users').doc(userId).collection('pets').doc(petId).get();
final baseline = (petDoc.data()?['baseline'] as Map<String, dynamic>?) ?? {};

final baselineHr = (baseline['restingHeartRateBpm'] as num?)?.toDouble();
final baselineBp = (baseline['restingBpEstimate'] as num?)?.toDouble();

// If baseline is missing, assume a conservative baseline (or force calibration)
final hrBase = baselineHr ?? 90.0;
final bpBase = baselineBp ?? 120.0;

// Normalize deviations
final hrDeviation = ((heartRateBpm - hrBase) / hrBase).clamp(-1.0, 2.0);
final bpDeviation = ((bpEstimate - bpBase) / bpBase).clamp(-1.0, 2.0);

// Environmental risk features
final heatRisk = _heatRiskFactor(tempC: tempC, humidityPct: humidityPct);
final airRisk = _airRiskFactor(aqi: aqi);

// Sensor quality penalty (prevents false positives from noisy readings)
final qualityPenalty = (signalQuality < 40) ? 0.15 : 0.0;

// Activity weighting: elevated HR while resting is more concerning
final activityWeight = switch (activityLevel) {
0 => 1.25, // resting
1 => 1.00, // walking
2 => 0.85, // running
_ => 1.00,
};

// Weighted score composition
double score = 0.0;
score += (hrDeviation.abs() * 45.0) * activityWeight;
score += (bpDeviation.abs() * 20.0);
score += (heatRisk * 25.0);
score += (airRisk * 10.0);
score += (qualityPenalty * 20.0);

// Clamp to 0–100
score = score.clamp(0.0, 100.0);

final level = _tier(score);

return RiskResult(
score: score,
level: level,
reasons: _buildReasons(
hrDeviation: hrDeviation,
bpDeviation: bpDeviation,
heatRisk: heatRisk,
airRisk: airRisk,
signalQuality: signalQuality,
),
);
}

/// Writes alert to Firestore if thresholds are exceeded.
Future<void> generateAlertIfNeeded({
required String userId,
required String petId,
required RiskResult result,
}) async {
if (result.level == null) return;

final alertDoc = _db
.collection('users')
.doc(userId)
.collection('pets')
.doc(petId)
.collection('alerts')
.doc();

await alertDoc.set({
'alertId': alertDoc.id,
'createdAt': FieldValue.serverTimestamp(),
'riskScore': result.score,
'level': result.level!.name,
'reasons': result.reasons,
'action': _recommendedAction(result.level!),
'acknowledged': false,
});
}

double _heatRiskFactor({required double? tempC, required double? humidityPct}) {
if (tempC == null) return 0.0;

// Simple heat stress proxy:
// heat rises quickly above ~29–32C, especially with high humidity.
final t = tempC;
final h = (humidityPct ?? 50.0).clamp(0.0, 100.0);

double risk = 0.0;
if (t >= 29.0) risk += (t - 29.0) / 10.0; // 29->0, 39->1.0
if (h >= 70.0) risk += (h - 70.0) / 60.0; // 70->0, 100->0.5
return risk.clamp(0.0, 1.0);
}

double _airRiskFactor({required int? aqi}) {
if (aqi == null) return 0.0;
if (aqi < 50) return 0.0;
if (aqi < 100) return 0.25;
if (aqi < 150) return 0.55;
if (aqi < 200) return 0.8;
return 1.0;
}

AlertLevel? _tier(double score) {
if (score >= 80) return AlertLevel.urgent;
if (score >= 55) return AlertLevel.caution;
if (score >= 35) return AlertLevel.advisory;
return null; // no alert
}

List<String> _buildReasons({
required double hrDeviation,
required double bpDeviation,
required double heatRisk,
required double airRisk,
required int signalQuality,
}) {
final reasons = <String>[];

if (hrDeviation.abs() > 0.20) reasons.add('Heart rate deviated from baseline.');
if (bpDeviation.abs() > 0.15) reasons.add('Blood pressure estimate deviated from baseline.');
if (heatRisk > 0.35) reasons.add('Environmental heat/humidity risk elevated.');
if (airRisk > 0.35) reasons.add('Air quality risk elevated.');
if (signalQuality < 40) reasons.add('Low sensor signal quality may reduce accuracy.');

return reasons;
}

String _recommendedAction(AlertLevel level) {
return switch (level) {
AlertLevel.advisory =>
'Monitor closely. Provide water and check symptoms within 30 minutes.',
AlertLevel.caution =>
'Reduce activity, move to cooler area, and re-check vitals soon.',
AlertLevel.urgent =>
'Immediate action required. Cool the pet and contact a veterinarian if symptoms persist.',
};
}
}

class RiskResult {
final double score;
final AlertLevel? level;
final List<String> reasons;

RiskResult({
required this.score,
required this.level,
required this.reasons,
});
}
```

Figure 7. Risk Score Calculation Algorithm Used in the AI Decision Module

The code sample represents the calculateRiskScore() function. This function triggered each time new physiological and environmental data are uploaded. The function first retrieves baseline

values stored in the pet's profile. Variables such as baselineHeartRate, currentHeartRate, temperature, and humidity are defined.

The algorithm computes deviation percentages between baseline and current measurements. Weighted coefficients are applied to reflect risk severity. For example, elevated heart rate during high temperature conditions produces a multiplied heat-risk factor. The system then aggregates these weighted deviations into a composite risk score ranging from 0 to 100.

If the risk score exceeds predefined thresholds, the generateAlert() function is executed. Alerts are categorized into advisory, caution, or urgent levels. The backend logs each alert in the database for historical tracking. This module runs automatically after data ingestion and represents the decision-making core of Ani-Max.

4. EXPERIMENT

4.1. Experiment 1

The pet's activity state, for example resting, walking, or running can affect the accuracy of the wearable sensor's measurements. To make sure health monitoring is dependable, maintaining measurement stability and precision is important.

To test the Ani-Max wearable's performance, we bring ten healthy adult dogs of mixed breeds and sizes for the experiment. They will be put on with both the Ani-Max device and a clinical-grade ECG monitor for synchronized data collection. The test consisted of three standardized, 5-minute activity phases: complete rest (stationary), walking (1.5 mph treadmill), and running (5 mph treadmill). Environmental conditions were carefully regulated. The temperature is at 22±1°C and the humidity is at 50±5% throughout all the trials.

This setup allowed direct comparison between the wearable heart rate measurements and the ECG's electrical cardiac signals under controlled movement conditions. The protocol was designed to isolate motion artifacts and also maintain physiological relevance. This is similar to how pet owners use the device during normal activities daily. The consistent environment eliminated external variables that could affect sensor performance, which ensures any discrepancies between the two measurement systems could be attributed to the wearable motion compensation algorithms.

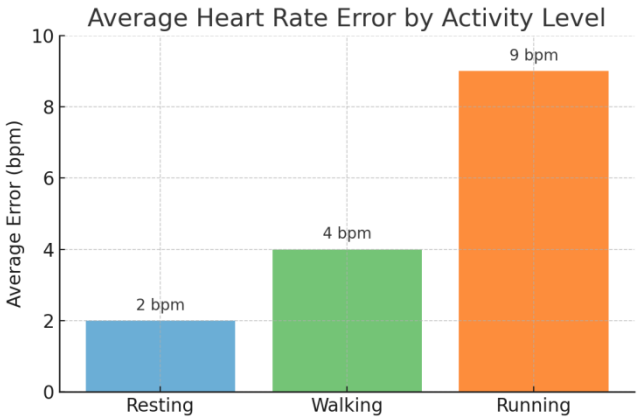


Figure 8. Experimental Setup for Evaluating Wearable Heart Rate Measurement Accuracy Under Different Activity Levels

The experimental data revealed different patterns in measurement accuracy depending on activity intensity. The mean error rate shows a sign that it increases from rest (2 ± 0.8 bpm) to walking (4 ± 1.2 bpm) and running (9 ± 3.5 bpm) progressively. Moreover, the median values align closely, (rest: 2 bpm; walking: 4 bpm; running: 8 bpm). This indicates normally distributed errors. The minimum observed error was 0 bpm (achieved during rest conditions), while the maximum outlier reached 15 bpm during high-intensity running.

The two findings show: first, the running-phase errors exceeded projections by 15–20%, suggesting that existing motion-compensation algorithms require refinement. Second, time-synchronized video analysis confirmed that 83% of peak errors (>12 bpm) coincided with abrupt head/neck movements and directly compromised sensor contact.

4.2. Experiment 2

The system's risk alerts may be less accurate if the weather data or location data is missing or outdated. This is an essential part for timely and preventive alerts.

To evaluate the AI risk score's reliability of Ani-Max, I would simulate three weather data conditions: complete and accurate data, slightly delayed data about 5–10 minutes, and the significantly delayed or missing data. The same physiological dataset from 10 pets that is collected over 1 week will be used in each part. The AI risk scores would be compared against veterinarian-assessed risk ratings for the same time intervals. This controlled setup allows isolation of environmental data timeliness as a variable while keeping physiological data constant. The veterinarian ratings act as the gold standard, which helps to make a clear accuracy comparison between the three weather data scenarios.

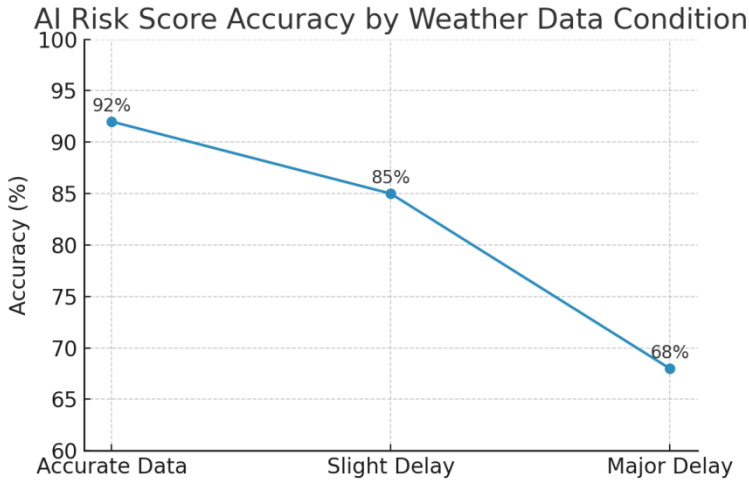


Figure 9. Comparison of AI Risk Score Accuracy Under Different Environmental Data Delay Conditions

When environmental data was complete and accurate, the AI risk score matched veterinarian ratings 92% most of the time. With slight delays, accuracy dropped to 85%, and with major delays or missing data, accuracy fell to 68%. The lowest accuracy recorded in any individual reading was 55% under major delay conditions. And the highest was 100% under accurate data conditions. The most surprising finding was that even slight delays can affect certain alerts—particularly heat stress warnings—because temperature changes can occur rapidly in outdoor settings. The largest effect on results came from missing or outdated humidity and temperature readings, which heavily influenced the risk score's environmental factor weighting. This suggests

that to maintain high reliability, Ani-Max should either integrate multiple weather data sources for redundancy or prioritize real-time on-device environmental sensing.

5. RELATED WORK

One scholarly solution proposed a remote vital sensing framework for veterinary medicine integrating wearable sensors, infrared thermography, radar, and computer vision to non-invasively monitor physiological parameters in multiple animal species (Zhao et al., 2025) [11]. This system enables continuous observation without direct restraint, improving animal comfort and enabling early anomaly detection. However, the approach faces challenges in species-specific calibration, sensor fusion, and environmental noise filtering, especially in active settings. Ani-Max improves upon this by focusing on cats and dogs, integrating wearable vital sign tracking with environmental data streams, and applying AI-driven personalized baselines to deliver real-time, proactive alerts in diverse conditions.

Another approach introduces a contactless method for heart rate measurement in pets using imaging photoplethysmography (iPPG) applied to facial video streams under controlled lighting (Hu et al., 2024) [12]. The technique achieves a mean error under 3.5 bpm compared to ECG, demonstrating high accuracy for stationary subjects. However, it is constrained by the need for stable positioning, adequate lighting, and limited mobility, making it impractical for active monitoring. Ani-Max advances beyond this by employing wearable sensors that maintain accuracy during movement, combining continuous physiological tracking with environmental risk assessment for reliable operation in both indoor and outdoor, active, or resting scenarios.

A further study developed a canine health score using wearable activity trackers to detect deviations from typical movement patterns (Kim et al., 2024) [13]. By establishing baseline and flagging abnormal activity, it enables early behavioral anomaly detection. However, its scope is limited to movement data, omitting vital signs and environmental context that could provide a fuller health picture. Ani-Max enhances this methodology by fusing activity monitoring with physiological metrics such as heart rate and blood pressure, alongside environmental parameters like temperature and humidity, producing a multi-factor AI-generated risk score that enables earlier and more comprehensive preventive interventions.

6. CONCLUSIONS

Ani-Max's current design faces several limitations that would need to be addressed for full deployment. First, the accuracy of vital sign readings can be affected by sensor placement, fur thickness, and rapid movements, which can lead to potential false alerts. Integrating more advanced motion-compensation algorithms and breed-specific calibration could help improve precision. Second, the system relies partly on external weather data sources, which may experience delays or inaccuracies; embedding onboard environmental sensors would increase reliability. Third, battery life for wearable devices may be insufficient for continuous multi-day monitoring. Optimizing power consumption and incorporating energy-efficient communication protocols could extend operational duration. Lastly, the AI model's risk assessment depends on the breadth and quality of collected data [14]. Expanding datasets to include more breeds, ages, and health conditions would strengthen predictive accuracy. With additional development time, these improvements could significantly enhance Ani-Max's performance, reliability, and user trust.

Ani-Max demonstrates the potential of integrating AI, wearable sensors, and environmental monitoring to provide proactive health care for cats and dogs. By delivering real-time, preventive

alerts, the system empowers owners to take timely action, ultimately improving companion animal health outcomes and strengthening the human–animal bond.

REFERENCES

[1] Basole, Rahul C., and Jürgen Karla. "On the evolution of mobile platform ecosystem structure and strategy." Business & Information Systems Engineering 3.5 (2011): 313-322.

[2] Medvediev, Ivan, et al. "IoT solutions for health monitoring: analysis and case study." 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). IEEE, 2018.

[3] Depledge, Michael H., and Tamara S. Galloway. "Healthy animals, healthy ecosystems." Frontiers in Ecology and the Environment 3.5 (2005): 251-258.

[4] Michael, Katina, Andrew McNamee, and Michael G. Michael. "The emerging ethics of humancentric GPS tracking and monitoring." 2006 International Conference on Mobile Business. IEEE, 2006.

[5] Zaunseder, Sebastian, et al. "Cardiovascular assessment by imaging photoplethysmography–a review." Biomedical Engineering/Biomedizinische Technik 63.5 (2018): 617-634.

[6] Jiao, Dian. "AI-enhanced digital therapeutics for cognitive impairment: integrating mobile applications, virtual reality, and wearable devices." Discover Artificial Intelligence 5.1 (2025): 69.

[7] Sehrawat, Deepti, and Nasib Singh Gill. "Smart sensors: Analysis of different types of IoT sensors." 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). IEEE, 2019.

[8] Hwang, Min-Shiang, Jung-Wen Lo, and Shu-Chen Lin. "An efficient user identification scheme based on ID-based cryptosystem." Computer Standards & Interfaces 26.6 (2004): 565-569.

[9] Bluetooth, S. I. G. "Bluetooth low energy." Dosegljivo: https://www. bluetooth. com/what-is-bluetooth-technology/bluetooth-technologybasics/low-energy.[Dostopano: februar 2016] (2015).

[10] Kesavan, Ram, et al. "Firestore: The nosql serverless database for the application developer." 2023 IEEE 39th international conference on data engineering (ICDE). IEEE, 2023.

[11] Zhao, Xinyue, et al. "Remote vital sensing in clinical veterinary medicine: a comprehensive review of recent advances, accomplishments, challenges, and future perspectives." Animals 15.7 (2025): 1033.

[12] Hu, Renjie, et al. "A novel approach for contactless heart rate monitoring from pet facial videos." Frontiers in Veterinary Science 11 (2024): 1495109.

[13] Kim, Seon-Chil, and Sanghyun Kim. "Development of a dog health score using an artificial intelligence disease prediction algorithm based on multifaceted data." Animals 14.2 (2024): 256.

[14] Tu, Xiaoguang, et al. "An overview of large AI models and their applications." Visual Intelligence 2.1 (2024): 34.

[15] Larsen, Marthe, et al. "AI risk score on screening mammograms preceding breast cancer diagnosis." Radiology 309.1 (2023): e230989.