VitalGuard: Constraint-Based Ethical Design for Ultra-Lightweight Offline AI in Humanitarian Contexts

Gyu-min Jeon (Morgan J.)
M-Corp Ethical AI, Hanbat National University, Republic of Korea
contact@mcorpai.org

February 2026

Abstract

Humanitarian AI often assumes connectivity, centralized data, and institutional capacity that are absent in refugee camps, conflict zones, and high-surveillance environments, where privacy and liability risks can outweigh potential benefits. We present VitalGuard, an ultra-lightweight offline AI artefact constrained to 54 kB and implemented in vanilla HTML and JavaScript with zero external dependencies. VitalGuard is designed to be auditable and non-extractive: it performs on-device, session-bounded processing without collecting, storing, or transmitting personal data. Rather than optimizing for benchmark performance, we propose a constraint-based ethical design approach in which minimalism operates as an accountability mechanism. We derive ten implementable design principles from GDPR, UNCRC, and CRPD, and map each principle to concrete architectural constraints. We then formalize key properties using a description-length proxy for auditability, a privacy-by-non-collection threat model, and computational complexity bounds for real-time operation on decade-old smartphones. Preliminary validation includes a multi-week design review by a European government development agency and exploratory academic discussions with humanitarian and disability-innovation researchers. We present illustrative use cases for refugee identity matching and basic health screening, and we candidly discuss technical limits, deployment barriers, and misuse risks, including surveillance appropriation. VitalGuard is framed as a research pathway toward verifiable, governance-ready offline AI in vulnerable settings.

Keywords: Offline AI, humanitarian technology, ethical AI, ultra-lightweight systems, TinyML, privacy-by-design, refugee self-reliance, constraint-based design, GDPR compliance, surveillance resistance

1. Introduction

The deployment of artificial intelligence in humanitarian contexts has expanded substantially in recent years, with applications ranging from satellite-based displacement monitoring to natural language processing for crisis communication [1, 2]. International organisations, nongovernmental organisations (NGOs), and academic institutions have invested considerable resources in exploring how machine learning might improve service delivery to vulnerable populations, including refugees, internally displaced persons, and communities affected by conflict or natural disaster [3, 4]. Yet a persistent and largely unresolved tension underlies this expansion: the very populations most in need of technological support are those least served by the dominant paradigm of cloud-dependent, data-intensive AI [5].

This tension is not merely technical. It is structural. Cloud-based AI systems require reliable internet connectivity, centralised data storage, and ongoing computational resources—conditions that are routinely absent in refugee camps, conflict zones, and regions subject to digital surveillance [6, 7]. More critically, the data collection practices inherent to most contemporary AI systems create legal and ethical liabilities that many humanitarian organisations cannot accept. Under the General Data Protection Regulation (GDPR), even well-intentioned data processing may expose organisations to substantial financial penalties and reputational damage [8]. In contexts where the populations served are already subject to surveillance, persecution, or exploitation, the risks are not abstract but immediate and potentially life-threatening [9, 10].

The consequence is a paradox: the communities with the greatest need for AI-assisted support are precisely those excluded from its benefits by the architectural assumptions of mainstream AI development. This exclusion is not a failure of computing power or algorithmic sophistication. It is a failure of design philosophy. The prevailing approach optimises for performance on standard benchmarks while treating deployment constraints—connectivity, privacy, institutional liability—as afterthoughts to be addressed through adaptation rather than foundational design [11, 12].

This paper proposes an alternative framing. Rather than beginning with the question, “What can AI do?” and subsequently attempting to retrofit solutions for constrained environments, we begin with the question, “What may AI ethically be permitted to do in the most vulnerable settings, and under what institutional conditions?” This inversion of the standard design logic motivates VitalGuard, an ultra-lightweight offline AI system constrained to 54 kilobytes, implemented entirely in vanilla HTML and JavaScript, requiring no external libraries, no server connections, and no data transmission of any kind.

VitalGuard is not an attempt to compete with large-scale AI systems on performance metrics. It is a deliberate exercise in constraint-based ethical design: the proposition that extreme minimalism—in file size, in computational requirements, in data handling—can itself function as a form of accountability [13]. A system that collects no data cannot leak data. A system that requires no network connection cannot be intercepted. A system that fits within 54 kilobytes can be inspected in its entirety by a single reviewer in a reasonable timeframe. These are not merely engineering conveniences; they are ethical properties that address the specific concerns of humanitarian organisations considering technology adoption [14, 15].

The contributions of this paper are as follows. First, we articulate ten principles of ethical AI design derived from international legal frameworks (GDPR, UNCRC, CRPD) and grounded in the operational realities of humanitarian deployment. Second, we present the architecture of VitalGuard, demonstrating how each design principle translates into specific technical decisions. Third, we provide operational definitions and mathematical framing for auditability, privacy exposure, and runtime constraints, using description-length proxies, a privacy-by-non-collection threat model, and computational complexity analysis. Fourth, we report preliminary validation including a multi-week design review by a European government agency and early-stage academic discussions with the Institute of Development Studies (IDS) and UCL Global Disability Innovation Hub. Fifth, and perhaps most importantly, we present a candid and detailed assessment of limitations, field barriers, and ethical risks—including the possibility that the system could be misused for surveillance purposes—as we believe that honest engagement with failure modes is a prerequisite for responsible innovation in this domain [16, 17].

It is worth noting what this paper does not claim. We do not claim that VitalGuard outperforms existing AI systems on standard benchmarks—it does not, and by design cannot. We do not claim that the system is ready for deployment—field validation has not yet occurred. We do not claim that constraint-based design is universally superior to performance-optimised approaches—in many contexts, it is not. What we do claim is that for a specific and important class of deployment scenarios—those characterised by surveillance risk, infrastructure absence, and institutional liability sensitivity—the design philosophy presented here may offer a pathway that the dominant paradigm cannot. Whether this claim withstands empirical scrutiny is a question for future work.

We also wish to acknowledge a structural tension in this work. The system is designed by a researcher who is not part of the displacement-affected communities for potential deployment among displaced populations. This power asymmetry cannot be resolved by technical means alone, and we address it explicitly in our discussion of ethical risks (Section 6). The aspiration toward community ownership and participatory design is genuine but presently unrealised; we treat it as a research objective rather than an achieved outcome.

The remainder of this paper is organised as follows. Section 2 reviews related work across ten domains relevant to ultra-lightweight humanitarian AI. Section 3 presents the design principles and their implementation. Section 4 provides mathematical formalisation. Section 5 describes the system architecture and preliminary validation. Section 6 addresses limitations, risks, and ethical concerns. Section 7 concludes with reflections on future directions.

2. Related Work

The design space addressed by VitalGuard sits at the intersection of several research traditions that have, until recently, developed largely in isolation. This section reviews ten domains whose convergence frames the contribution of the present work. Rather than treating these as independent literatures, we attempt to trace the logical connections between them, as it is precisely their intersection—ultra-lightweight computation, ethical governance, and humanitarian deployment—that defines the gap this paper seeks to address.

2.1 TinyML and Edge AI

The TinyML paradigm has demonstrated that meaningful machine learning inference is feasible on microcontrollers with as little as 256 kilobytes of memory [18, 19]. Warden and Situnayake (2022) provide a comprehensive treatment of deployment strategies for resource-constrained devices, while Banbury et al. (2021) establish benchmarking standards through MLPerf Tiny [20, 21]. More recently, Lin et al. (2024) have shown that on-device training—not merely inference—can be achieved within severe memory budgets [22]. However, most TinyML research focuses on sensor data and embedded systems rather than humanitarian applications, and the ethical dimensions of deployment in vulnerable contexts remain underexplored.

2.2 AI Ethics and Governance Frameworks

The ethical AI literature has expanded considerably, with Jobin et al. (2019) identifying over 84 guidelines across national and institutional frameworks [23]. Floridi et al. (2022) provide a philosophical foundation for AI governance that emphasises transparency and accountability [24]. The European Union’s AI Act (2024) establishes regulatory requirements that directly affect humanitarian technology deployment [25]. Critically, these frameworks increasingly recognise that ethical AI is not merely about avoiding harm but about ensuring that technological benefits reach excluded populations. Yet a gap persists between high-level principles and operational implementation, particularly in low-resource settings where compliance mechanisms may be unavailable [26].

2.3 Differential Privacy

Differential privacy (DP), formalised by Dwork and Roth (2014), provides a rigorous guarantee that the inclusion or exclusion of a single individual's data does not substantially change the distribution of a released output. DP has been adopted in large-scale consumer systems and extended to federated and distributed settings. In humanitarian contexts, DP is often discussed as a way to reduce the ethical and legal risk of analytics on sensitive populations, because its assurance is mathematical rather than institutional.

VitalGuard does not claim to implement (epsilon, delta)-differential privacy in its present form. The system performs local, session-bounded processing and does not publish aggregate statistics or maintain a learning loop across individuals. Instead, VitalGuard reduces privacy exposure by avoiding the collection, storage, and transmission of personal data. We include DP here as a comparative baseline and as a potential extension should future versions require population-level learning or reporting.

2.4 Kolmogorov Complexity and Information Theory

Kolmogorov complexity provides a formal lens on description length: the complexity of an object can be characterised as the length of the shortest program that generates it on a universal machine. While the exact complexity of real software is uncomputable and depends on the reference machine, the concept offers an intuition for why a small, self-contained artefact can be audited more reliably than a sprawling system. In this work, we treat the 54 kB constraint not as a proof about Kolmogorov complexity, but as a practical auditable boundary: a size at which the full behaviour of the system can be inspected end-to-end by a single reviewer within a bounded time window. Information theory therefore supports the argument for minimalism as accountability rather than a claim of formal optimality.

2.5 Humanitarian AI and Development Technology

The application of AI in humanitarian settings has been examined by multiple scholars and institutions. The OCHA Centre for Humanitarian Data (2023) documents both opportunities and failures in humanitarian data systems [34]. Vinuesa et al. (2020) map AI's potential contributions to the Sustainable Development Goals [35], while Madianou (2019) introduces the concept of "technocolonialism" to describe power asymmetries in humanitarian technology transfer [36].

More recently, field reports from UNHCR and ICRC highlight the gap between AI's theoretical potential and its actual deployment, noting that institutional risk aversion, data protection concerns, and infrastructure limitations remain the primary barriers to adoption [37, 38].

2.6 Explainable AI (XAI)

The demand for explainability in AI systems has produced a substantial literature, from Ribeiro et al.'s LIME (2016) to more recent work on concept-based explanations [39, 40]. In humanitarian settings, explainability takes on additional urgency: decisions affecting refugee status, health triage, or resource allocation must be auditable not only by technical reviewers but by affected communities and their advocates [41]. VitalGuard's vanilla JavaScript implementation represents an extreme form of explainability—the entire codebase can be read and understood without specialised tools—though we acknowledge that code transparency alone does not guarantee interpretability for non-technical stakeholders.

2.7 Classical Algorithms and Lightweight Computation

Prior to the deep learning era, substantial work established that classical algorithms—Bayesian classifiers, decision trees, probabilistic matching—can achieve reliable performance on well-defined tasks with minimal computational overhead [42, 43]. Cormen et al. (2022) provide canonical complexity analyses that inform our selection of algorithms constrained to $O(n \log n)$ time and $O(n)$ space [44]. This classical foundation is not nostalgic; it is pragmatic. In settings where every kilobyte matters, algorithmic elegance is not a luxury but a necessity.

2.8 Digital Sovereignty and Surveillance Resistance

The concept of digital sovereignty has gained prominence in policy discussions, particularly in the European context [45]. For vulnerable populations, digital sovereignty is not an abstract governance principle but a survival concern: the ability to control one's own data may be the difference between safety and persecution [9, 46]. VitalGuard's fully offline architecture is designed to address this concern at the most fundamental level—by ensuring that no data ever leaves the device on which it is processed.

2.9 International Legal Frameworks

Three international frameworks are particularly relevant to the present work. The GDPR (2016/2018) establishes stringent data protection requirements that apply regardless of the data subject's economic status or geographic location [8]. The UNCRC provides special protections for children's data and wellbeing, which are critical in refugee contexts where minors constitute a disproportionate share of displaced populations [47]. The CRPD mandates accessibility and non-discrimination in technology design, a requirement that intersects with VitalGuard's commitment to operating on low-specification devices [48].

2.10 Offline-First Architecture

The offline-first design paradigm, though primarily associated with progressive web applications in the commercial sector, has direct applicability to humanitarian deployment [49, 50]. Recent work on local-first software principles emphasises that systems should function without network connectivity as the default, rather than the exception [51]. VitalGuard extends this principle to its logical conclusion: the system is not offline-capable but offline-only, with no network functionality whatsoever. This architectural choice eliminates an entire category of security and privacy risks, though it introduces constraints on model updating and synchronisation that are discussed in Section 6.

2.11 Synthesis: The Gap at the Intersection

The ten domains reviewed above share a common observation, though each articulates it from a different vantage point: existing approaches to AI in humanitarian settings are structurally misaligned with the contexts in which they are deployed. TinyML research demonstrates that lightweight computation is feasible but does not address humanitarian ethics. AI ethics frameworks articulate principles but struggle with operational implementation in low-resource settings. Differential privacy provides mathematical rigour but has been applied primarily in commercial contexts. Humanitarian technology research documents failures and power asymmetries but often lacks concrete technical proposals for alternative architectures.

The gap that VitalGuard seeks to address sits precisely at the intersection of these domains: the absence of a system that simultaneously satisfies the computational constraints identified by TinyML research, the ethical requirements articulated by AI governance frameworks, the privacy guarantees formalised by differential privacy theory, the deployment realities documented by humanitarian technology scholars, and the legal obligations imposed by international regulatory frameworks. This is not a claim that existing work is insufficient—it is an observation that the convergence of these requirements has not yet been explored as a unified design problem.

3. Design Principles and Implementation

The design of VitalGuard is governed by ten principles derived from the operational requirements of humanitarian deployment and the legal frameworks discussed in Section 2. These principles are not aspirational; each one corresponds to a concrete architectural constraint that shapes the system's implementation. We present them here with their technical operationalisation, followed by mathematical formalisation of key system properties.

3.1 Ten Principles of Ethical AI for Humanitarian Contexts

Principle 1: Minimise hallucinations. Incorrect outputs in sensitive domains—child protection, healthcare, disaster response—can have immediate and severe consequences. VitalGuard addresses this by restricting itself to classical probabilistic methods (Bayesian classifiers, rulebased systems) rather than generative models, and by enforcing mandatory human review thresholds: any output with confidence below 0.8 triggers automatic deferral to human judgement.

Principle 2: Ensure transparency. The system is implemented entirely in vanilla HTML and JavaScript. The complete source code is readable by any developer with basic web programming knowledge, without requiring specialised machine learning expertise. This represents a form of radical transparency—the entire system is auditable by inspection—though we acknowledge that code-level transparency does not automatically translate to decision-level interpretability for non-technical users.

Principle 3: Guarantee accessibility in low-income settings. Open-source code and architecture are essential to earning trust in humanitarian contexts. The system requires no proprietary software, no licensing fees, and no specialised hardware beyond a basic smartphone or laptop.

Principle 4: Eliminate data exploitation. VitalGuard operates without any data collection or centralised server infrastructure. All processing occurs locally on the user's device. No telemetry, analytics, or usage data is transmitted. This serverless design eliminates the legal risks associated with data processing agreements and cross-border data transfers under GDPR.

Principle 5: Operate in low-resource environments. The system is designed to run on refurbished smartphones and devices powered by portable solar panels. The 54 kB constraint ensures that the system loads instantaneously even on 2G connections (for initial deployment) and runs efficiently on hardware with minimal processing power and memory.

Principle 6: Remove legal liability. Even minimal exposure to litigation can make institutional adoption infeasible in high-risk contexts. VitalGuard is therefore designed to minimise organisational exposure by ensuring that personal data is not collected, persistently stored, or transmitted, and by keeping processing ephemeral and user-controlled within a single session. The system does not claim differential privacy unless an explicit noise mechanism is added; instead, Section 4 provides a privacy-by-non-collection argument and a threat model that clarifies what is and is not protected.

Principle 7: Be free of charge and lightweight. Free access enables adoption in communities with no income. The absence of licensing costs also ensures that international institutions bear no financial liability for deployment.

Principle 8: Emphasise simplicity and standardisation. Life-saving AI requires reliable statistics and probabilistic analysis rather than complex reasoning that increases hallucination risk. VitalGuard uses well-understood classical algorithms whose behaviour is predictable and whose error modes are well-characterised in the literature.

Principle 9: Be non-specialist friendly. Most NGO personnel are not programmers. The codebase is designed to be straightforward, extensively commented, and easy to adapt in the field. Every function includes explanatory documentation accessible to readers with basic technical literacy.

Principle 10: Collect no data and allow simple deletion. Users must be able to erase the AI and all associated data at any time. The system stores no persistent data; closing the browser window eliminates all traces of usage. This ease of deletion means freedom from responsibility for deploying organisations and safety for users in surveillance-heavy environments.

3.2 Architectural Implementation

The system architecture follows a four-layer design, with each layer constrained by the principles described above:

Input Layer. Accepts text-based inputs (names, dates, symptom checklists) through a minimal HTML interface. Input validation occurs locally with no data persistence. Image and sensor data, where required, are processed as simplified metadata rather than raw binary, reducing both computational overhead and privacy exposure.

Processing Layer. Contains the core inference engine, implemented in vanilla JavaScript. Algorithms include Naïve Bayes classifiers for risk stratification, Levenshtein distance matching for identity verification, and weighted scoring models for triage decisions. All algorithms operate within O(n log n) time complexity and O(n) space complexity bounds.

Output Layer. Presents results through the HTML interface with explicit confidence indicators. Outputs below the 0.8 confidence threshold are flagged with mandatory human review warnings. All output is ephemeral—no results are persisted beyond the active browser session.

Security Layer. Restricts all sensitive processing to volatile in-memory state and clears state on session termination. The implementation forbids network calls and persistent storage by design, reducing remote attack surface and limiting long-lived data exposure. The small codebase (54 kB) supports complete security audit by a single reviewer within a bounded time window.

The relationship between these four layers and the ten design principles is not one-to-one but many-to-many: each principle constrains multiple layers, and each layer embodies multiple principles. For example, Principle 4 (eliminate data exploitation) constrains the Input Layer (no persistent storage), the Processing Layer (no telemetry), and the Security Layer (session-based encryption). Similarly, Principle 2 (ensure transparency) applies across all layers through the choice of vanilla JavaScript, which renders the entire processing pipeline readable without decompilation, obfuscation, or specialised tooling.

A critical architectural decision is the constraint-based feature inclusion protocol. For any proposed feature F, inclusion requires satisfying a conjunctive condition: F is necessary for the target humanitarian use case AND F can be implemented within the 54 kB boundary AND F does not violate any of the ten design principles. This triple-filter approach deliberately privileges exclusion over inclusion. The result is a system that does fewer things than a conventional AI application but does them within a fully verifiable ethical and legal framework. We regard this as a feature, not a limitation—though we acknowledge that reasonable observers may disagree. The fallback mechanism deserves particular attention. When the system's confidence in a prediction falls below the 0.8 threshold, control is not merely flagged but actively transferred to a human decision-maker. This threshold was selected based on established literature on human-AI collaboration, which suggests that below this level, the marginal benefit of algorithmic assistance is outweighed by the risk of overreliance on uncertain predictions [16]. The threshold is configurable by deploying organisations, recognising that different operational contexts may justify different calibrations. However, the existence of a mandatory fallback is not configurable—it is a structural requirement of the system's ethical architecture.

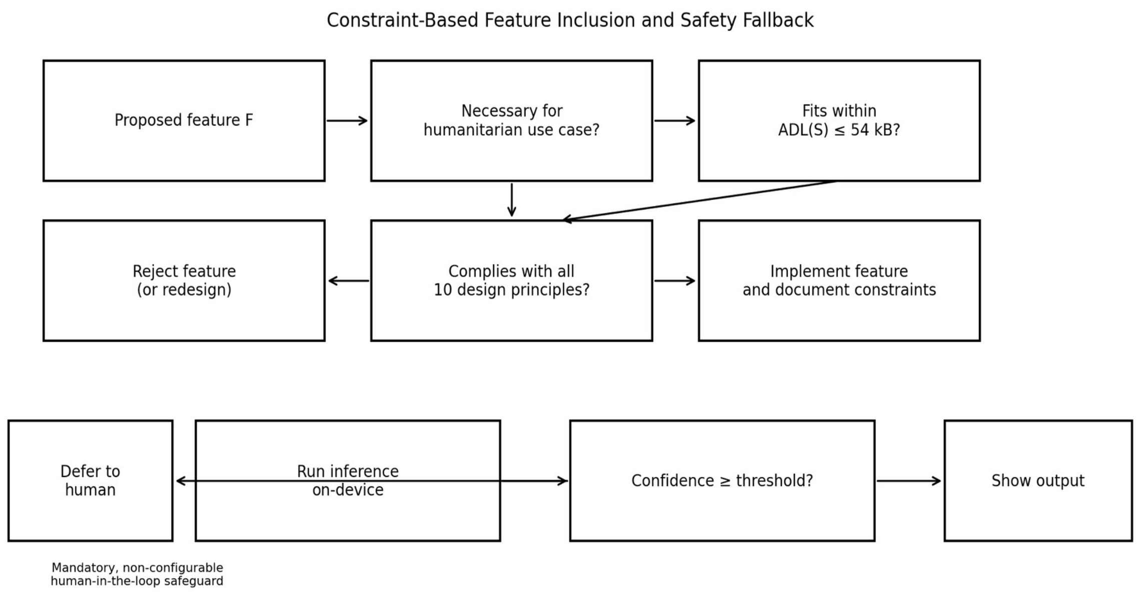
Constraint-Based Feature Inclusion and Safety Fallback
Proposed feature F
Necessary for humanitarian use case?
Fits within ADL(S) ≤ 54 kB?
Reject feature (or redesign)
Complies with all 10 design principles?
Implement feature and document constraints
Defer to human
Run inference on-device
Confidence ≥ threshold?
Show output
Mandatory, non-configurable human-in-the-loop safeguard

Figure 3. Constraint-based feature inclusion protocol and mandatory safety fallback (conceptual).

4. Mathematical Formalisation

This section provides formal mathematical characterisation of three key properties of the VitalGuard system: size constraint justification through Kolmogorov complexity, privacy guarantees through differential privacy, and computational efficiency through complexity analysis.

4.1 Kolmogorov Complexity and the 54 kB Boundary

Let $K(x)$ denote the Kolmogorov complexity of a binary string x, defined as the length of the shortest program p that produces x on a universal Turing machine U: $K(x) = \min\{ |p| : U(p) = x \}$ (1)

Kolmogorov complexity is uncomputable in general and depends on the choice of U. Consequently, we do not claim to measure K(S) for the VitalGuard artefact S. Instead, we introduce an auditable description-length proxy, ADL(S), defined as the literal encoded size of the distributed system (including HTML, JavaScript, and CSS). The design constraint ADL(S) <= 54 kB provides a practical upper bound on description length and serves as an engineering limit chosen to support end-to-end human audit.

The ethical rationale is operational: a system that can be fully read, executed, and reasoned about within a bounded time window is harder to hide behind opacity. Following the intuition of software comprehension metrics (e.g., Halstead-style token counts), a 54 kB vanilla JavaScript artefact typically corresponds to a few thousand meaningful tokens. This places the full codebase within a range that a skilled reviewer can realistically inspect in a working day, including the inspection of network calls, storage operations, and algorithmic pathways.

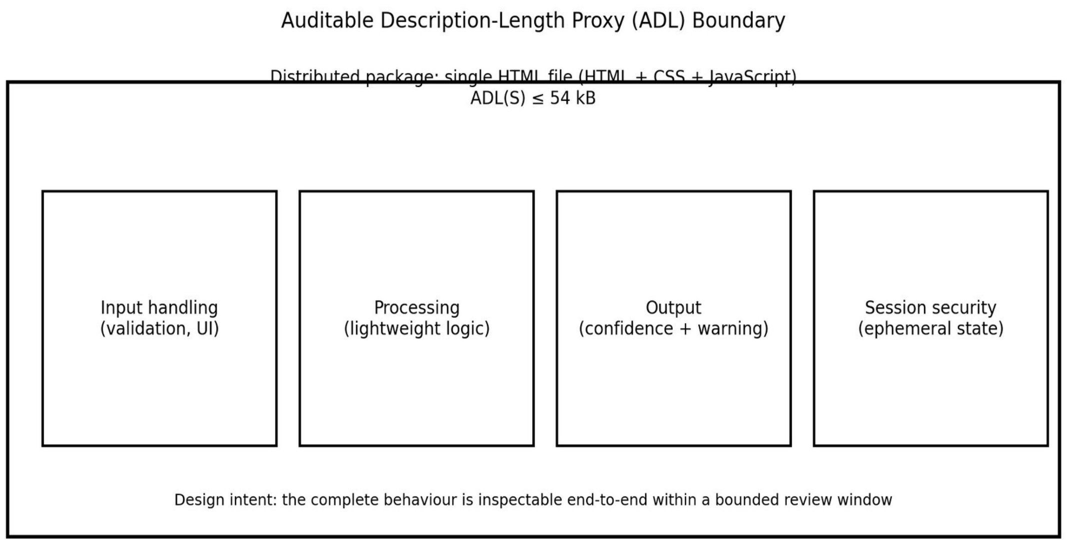
Auditable Description-Length Proxy (ADL) Boundary
Distributed package: single HTML file (HTML + CSS + JavaScript)
ADL(S) ≤ 54 kB
Input handling (validation, UI)
Processing (lightweight logic)
Output (confidence + warning)
Session security (ephemeral state)
Design intent: the complete behaviour is inspectable end-to-end within a bounded review window

Figure 2. The 54 kB auditable description-length boundary as a practical review envelope (conceptual).

We also define an information density ratio rho as: rho(S) = H(functionality) / ADL(S) (2) where H(functionality) is an informal measure of the diversity of functional outputs supported by the system. VitalGuard seeks a high rho by restricting functionality to compact classical algorithms and deterministic rules rather than parameter-heavy neural architectures. The purpose of this section is therefore not to claim theoretical optimality, but to justify a reviewable size boundary as a first-class ethical constraint.

4.2 Privacy-by-Non-Collection and Session Ephemerality

Differential privacy (DP) provides a noise-based statistical guarantee that limits how much information about any one individual can be inferred from released outputs. A randomised mechanism M satisfies (epsilon, delta)-differential privacy if, for all datasets D1 and D2 that differ in at most one element, and for all measurable sets S: Pr[M(D1) in S] <= exp(epsilon) * Pr[M(D2) in S] + delta (3)

VitalGuard does not claim to implement (epsilon, delta)-differential privacy in its current form, because it does not add an explicit noise mechanism and it does not operate a population-level learning or reporting loop. Instead, VitalGuard reduces privacy exposure through a different design axis: it aims to avoid the creation of long-lived data custody by prohibiting persistent collection and by keeping processing local to the user device.

We formalise this stance as two implementable constraints. First, the no-persistent-retention constraint: session data are represented as in-memory variables only and are not written to persistent stores such as localStorage, IndexedDB, cookies, or file-system APIs. Let Mem(t) denote volatile memory state during the session and Persist denote the set of all persistent storage locations accessible to the runtime. Then the intended invariant is: For all t in [t_start, t_end], Write(Mem(t), Persist) = false. (4) Second, the no-exfiltration constraint: the runtime contains no outbound network calls (for example fetch, XMLHttpRequest, WebSocket, or beacon APIs). Let NetCalls denote the set of outbound network invocations. Then: NetCalls = emptyset. (5)

Under a threat model where the device and browser environment are not already compromised, these constraints substantially reduce the risk of remote mass-surveillance, because there is no server-side repository to seize, leak, or subpoena. However, these are not absolute privacy guarantees: an attacker with local device access, malware-level compromise, or screen-capture capability can still observe user inputs and outputs. We therefore treat privacy as a bounded, threat-model-dependent property rather than an unconditional proof.

For future extensions that require aggregated reporting or model updates, DP remains relevant. In that case, local DP or federated DP could be integrated as an explicit mechanism, and only then would DP parameters (epsilon, delta) become meaningful claims for VitalGuard.

4.3 Computational Complexity Bounds and Performance Targets

VitalGuard relies on classical algorithms with transparent computational complexity. Two representative components are (i) naive Bayes classification for lightweight decision support and (ii) Levenshtein-distance matching for identity string comparison. For naive Bayes classification with N samples and F features, training complexity is O(NF) and inference is O(F). For Levenshtein distance between strings of length m and n, standard dynamic programming yields O(mn) time and O(mn) space, with well-known optimisations to O(min(m,n)) space when needed. These bounds provide predictability on low-resource devices. Rather than asserting a universal runtime such as "100 ms on baseline hardware" without a reproducible measurement context, we define performance as a design target: for typical humanitarian inputs (short text fields, small feature vectors, and limited vocabulary), the system is designed to remain interactive on commodity low-end smartphones. Appendix A provides a concrete benchmarking protocol that fixes device class, browser engine, input sizes, and measurement method so that future evaluations can report defensible numbers. Finally, we discuss GDPR exposure. The GDPR definition of "processing" is broad and includes operations performed on personal data even when the data never leaves the device. VitalGuard therefore does not "avoid processing entirely"; rather, it avoids persistent collection and organisational custody. The design goal is to minimise institutional liability by ensuring that the deploying organisation does not receive, store, or transmit personal data through VitalGuard, and by making the processing path fully inspectable.

4.4 Threat Model and Auditable Security Invariants

Because VitalGuard is intended for adversarial environments, security claims must be explicit about assumptions. We consider two primary adversary classes.

A remote adversary can observe network traffic and may control or coerce infrastructure operators, but does not have direct control of the user device. In this setting, the no-exfiltration constraint (Section 4.2) is the key protective measure.

A local adversary can obtain temporary or permanent access to the user device, including through malware, device seizure, or coercion. In this setting, VitalGuard cannot guarantee confidentiality of user inputs or outputs; the system is not a secure enclave and is not designed to resist device-level compromise.

From these assumptions, we propose four auditable invariants that can be checked directly in the source code. Invariant I1: no third-party code. The distributed artefact includes no external libraries, CDNs, remote scripts, or dynamic code loading. Invariant I2: no outbound network calls. The code contains no calls to fetch, XMLHttpRequest, WebSocket, sendBeacon, or equivalent APIs. Invariant I3: no persistent writes. The code contains no writes to localStorage, IndexedDB, cookies, service worker caches, file system APIs, or other persistent storage mechanisms. Invariant I4: deterministic algorithmic core. The core decision logic is inspectable and does not depend on opaque model weights or remote inference. These invariants do not guarantee perfect safety, but they provide a pragmatic foundation for independent verification, which is often the limiting factor in humanitarian deployment decisions.

5. System Description and Preliminary Validation

5.1 Architecture Overview

VitalGuard is implemented as a single HTML file containing embedded CSS styling, JavaScript inference logic, and a minimal user interface. The system is structured as four modular components—input handling, algorithmic processing, output presentation, and session security—each of which can be independently inspected and verified. The modular design permits field adaptation: individual components can be modified or replaced without affecting system integrity, provided the interfaces between modules are preserved. The constraint-based decision framework governs feature inclusion through a two-stage filter: (1) Is this functionality necessary for the humanitarian use case? (2) Can it be implemented within the 54 kB boundary while maintaining all ten design principles? Features that pass both filters are implemented; those that fail either are excluded, regardless of their potential utility. This approach inverts the standard feature-maximisation logic of commercial software development, prioritising ethical constraint satisfaction over functional breadth. A fallback mechanism ensures safety in all operational scenarios: when AI confidence falls below the defined threshold (0.8), the system immediately defers to human judgement with an explicit notification. This is not a graceful degradation feature but a core safety mechanism—the system is designed to recognise the boundaries of its own competence.

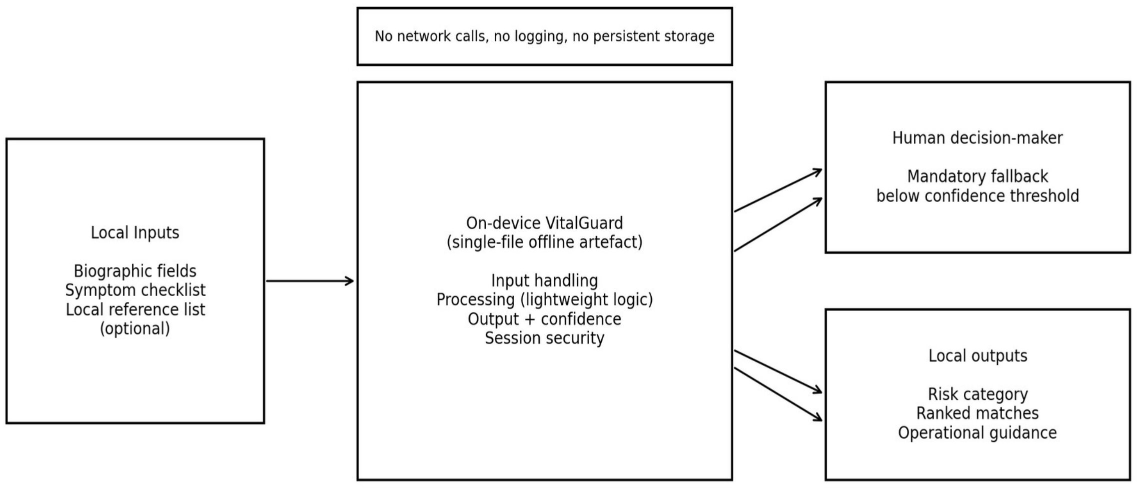
No network calls, no logging, no persistent storage
Local Inputs
Biographic fields
Symptom checklist
Local reference list (optional)
On-device VitalGuard (single-file offline artefact)
Input handling
Processing (lightweight logic)
Output + confidence
Session security
Human decision-maker
Mandatory fallback below confidence threshold
Local outputs
Risk category
Ranked matches
Operational guidance

Figure 1. VitalGuard high-level architecture: offline, on-device processing with mandatory human fallback and no network dependence.

5.2 Preliminary Design Review: European Government Agency Review

In 2025, VitalGuard underwent a three-week design review conducted by a European government development agency. The review examined (i) technical feasibility (whether the 54 kB constraint, offline operation, and dependency-free build were genuine and verifiable), (ii) ethical alignment (whether the stated design principles were reflected in the implementation), and (iii) policy compatibility (whether an offline-first, non-custodial approach could plausibly fit within public sector development programming).

The outcome requires careful characterisation. The review affirmed that core design claims such as offline operation and the absence of server-side data custody were technically checkable within the scope of the assessment. It also raised practical questions about deployment pathways, governance, and field validation, noting that formal public-sector adoption depends on institutional mandate, budgeting, and political timing beyond the scope of an engineering review.

For this paper, the value of the review is limited and specific. It does not constitute endorsement, certification, or approval. Its relevance is that an external public institution considered the design serious enough to review and did not identify immediate disqualifying contradictions between the stated ethical framing and the observable system constraints. We report it as an input to credibility, not as an argument for authority.

5.3 Academic Engagement

At the time of writing, exploratory academic discussions are underway with researchers at two institutions. At the Institute of Development Studies (IDS), University of Sussex, initial conversations with researchers working on AI ethics in development contexts and refugee selfreliance have explored potential alignment between VitalGuard's design philosophy and IDS's research agenda. At the UCL Global Disability Innovation (GDI) Hub, discussions have centred on the compatibility between VitalGuard's constraint-based approach and existing ethics-bydesign frameworks for humanitarian technology. These discussions are at an early stage, and we report them here with appropriate caution. No formal collaboration agreements have been established. The outcome of these conversations may range from productive research partnerships to a determination that the approach requires fundamental revision, or that collaboration is not appropriate at this time. We include this information to provide context for the system's current developmental trajectory, not to claim institutional validation that has not yet been granted.

5.4 Illustrative Use Cases

Use Case 1: Refugee Identity Matching. The system accepts basic biographical information (name, date of birth, language spoken) and performs local normalisation followed by approximate string matching against a locally stored reference database. The matching algorithm employs Levenshtein distance with phonetic normalisation to accommodate transliteration variations common in refugee documentation. Output consists of ranked candidate matches with explicit confidence scores. Where confidence falls below 0.8, the system displays a prominent warning requiring human adjudication. This use case demonstrates that meaningful identity verification functionality can operate within the 54 kB constraint, though with the important caveat that the system supplements rather than replaces existing UNHCR identification processes. Use Case 2: Basic Health Screening. The system presents a structured symptom checklist and applies a Naïve Bayes classifier to produce a three-level risk categorisation (Low, Medium, High) with a recommendation for specialist referral where indicated. This is explicitly a screening tool, not a diagnostic instrument. The distinction is both technical and legal: screening identifies individuals who may benefit from further assessment, while diagnosis assigns a specific medical condition. VitalGuard performs only the former, thereby avoiding the legal liability associated with medical device classification in most jurisdictions. These use cases are illustrative rather than evaluated. They demonstrate that the 54 kB constraint admits functionality of potential humanitarian value, but they do not constitute evidence of real-world effectiveness. Field validation—with actual refugee populations, in actual camp conditions, under actual operational pressures—is a prerequisite for any claim of utility, and such validation has not yet occurred. We present these cases to make the system's intended application concrete, while emphasising that the distance between technical demonstration and field deployment is substantial and should not be underestimated. A further consideration concerns the relationship between VitalGuard and existing humanitarian technology infrastructure. The system is not intended to replace established tools such as UNHCR's PRIMES registration system or WHO's DHIS2 health information platform. Rather, it is designed to operate in the gaps—contexts where connectivity is absent, where existing systems cannot function, or where populations are excluded from institutional services entirely. The complementary rather than competitive positioning is important: VitalGuard may be most valuable precisely where no other digital tool is available, and least relevant where established infrastructure is functioning normally. It is also worth acknowledging the role of the European government design review in the context of the broader international development ecosystem. In this ecosystem, institutional decision-making is characterised by path dependence, risk aversion, and peer referencing. The fact that a European government agency engaged in substantive review of the system's design—examining technical claims, ethical architecture, and policy alignment over a three-week period—does not constitute an endorsement. However, it does represent the passage of an implicit credibility threshold: the technology was deemed sufficiently serious to warrant institutional attention. In an ecosystem where the primary barrier to adoption is often not technical inadequacy but institutional uncertainty, this signal has practical significance, though it should not be overstated.

5.5 Reproducibility and Validation Checklist

VitalGuard is intentionally designed to be inspectable without specialised tooling. A minimal reproducibility package should allow an independent reviewer to verify the core constraints claimed in this paper.

First, size verification. The distributed artefact should be measured as the literal sum of the offline files (HTML, JavaScript, CSS) required to run the system without a network connection. Reporting should clarify whether the measurement is raw file size, compressed archive size, or packaged application size, as these numbers can differ materially.

Second, offline verification. The system should be launched with network connectivity disabled (for example, airplane mode or a controlled firewall rule) to confirm that all functionality described as "offline" remains available and that no requests are attempted.

Third, no-exfiltration verification. The source code can be audited by searching for outbound network APIs such as fetch, XMLHttpRequest, WebSocket, and sendBeacon, and by observing the browser network panel to confirm no outbound requests are made during typical usage.

Fourth, no-persistent-retention verification. The source code can be audited for writes to persistent storage APIs including localStorage, sessionStorage, IndexedDB, cookies, service worker caches, and file system APIs. Reviewers can also monitor storage panels during runtime to confirm that sensitive inputs do not persist beyond the session.

Fifth, algorithmic transparency. For each decision-support feature, the paper should specify the exact algorithmic pathway and parameter choices so that outputs can be reproduced deterministically under the same inputs.

Finally, performance reporting should be treated as empirical rather than asserted. Appendix A defines a timing protocol and representative input sizes. At the time of writing, field deployment remains pending, and claims should be read as design constraints and intended verification steps rather than proof of real-world impact.

6. Limitations, Risks, and Ethical Considerations

This section constitutes what we regard as the most important part of the paper. A system designed for deployment in humanitarian contexts—where decisions affect the safety and wellbeing of vulnerable populations—must be assessed not by what it can do, but by what it cannot do and what might go wrong. We organise this assessment into four categories: technical limitations, field deployment barriers, ethical risks, and future research directions.

6.1 Technical Limitations

VitalGuard’s 54 kB constraint, while ethically motivated, imposes genuine and non-trivial functional restrictions. First, the system cannot perform complex medical diagnosis. The Naïve Bayes classifier used for health screening is suitable for triage—identifying individuals who should be referred for professional assessment—but lacks the sophistication required for differential diagnosis. Any attempt to extend the system into diagnostic territory would be both technically inadequate and ethically irresponsible, as it would expose deploying organisations to medical liability without providing commensurate clinical benefit. Second, real-time image processing is beyond the system’s capacity. The 54 kB constraint precludes the inclusion of image decoding libraries; where visual information is relevant, the system can process only pre-extracted metadata (e.g., EXIF data, image dimensions) rather than pixel-level content. This limitation is significant for applications such as document verification, where image analysis might otherwise improve accuracy. Third, the system includes no generative AI capabilities. This is a deliberate exclusion rather than a technical shortcoming: generative models introduce hallucination risks that are unacceptable in humanitarian decision-making contexts. A system that fabricates plausible but false information about a refugee’s identity or medical condition could cause direct and measurable harm. Fourth, model updates require manual file replacement. The system has no mechanism for over-the-air updates, versioning, or automatic patching. This is a direct consequence of the offline-only architecture. While it eliminates the security risks associated with update mechanisms (a common attack vector), it creates a maintenance burden: field staff must physically deliver updated files, introducing delay and logistical complexity.

6.2 Field Deployment Barriers

Technology adoption in humanitarian settings is constrained less by technical capability than by institutional, social, and political factors. Three barriers merit specific attention. Institutional legal conservatism presents the most formidable obstacle. Even when a technology demonstrably works and is demonstrably safe, humanitarian organisations may decline to adopt it because the perceived risk of being the first adopter outweighs the expected benefit. This is a rational response to an asymmetric incentive structure: the institutional cost of a technology failure (reputational damage, regulatory sanction, litigation) far exceeds the institutional reward of a technology success (incremental improvement in service delivery). Addressing this barrier requires not better technology but better risk-sharing mechanisms—for example, pilot programmes in which legal liability is jointly designed between the technology provider and the deploying institution. Staff technology acceptance is a second, related challenge. Many humanitarian field workers operate in environments where digital technology is unreliable or has previously been associated with surveillance and control. Building trust in a new technological tool requires sustained engagement, training, and—critically—evidence that the tool has been co-designed with input from the communities it is intended to serve. VitalGuard’s deployment strategy must include participatory design processes rather than simply delivering a finished product. Data localisation and contextual adaptation present a third barrier. Algorithms trained or configured for one refugee population may perform poorly when applied to another. Linguistic patterns, naming conventions, health profiles, and cultural contexts vary substantially across displacement situations. VitalGuard’s modular architecture is designed to facilitate local adaptation, but the adaptation process itself requires technical capacity that may not be available in all deployment settings.

6.3 Ethical Risks and Potential for Misuse

We identify four categories of ethical risk that must be confronted directly. Surveillance appropriation. Although VitalGuard is designed as a surveillance-resistant technology, the environment in which it is deployed may not share this property. A camp administrator, government agency, or armed group could potentially repurpose the system’s identity matching functionality for monitoring or control purposes. The technical architecture resists this—no data is transmitted, no logs are kept—but the social context of deployment can undermine technical safeguards. Mitigation requires mandatory institutional ethical review prior to any deployment, with veto authority granted to community representatives. Developer bias. Even a system implemented in transparent vanilla JavaScript embodies the biases of its designers. Algorithm selection, feature weighting, threshold calibration, and interface design all reflect the worldview and assumptions of the development team. If that team lacks diversity—in geography, culture, gender, disability status, or lived experience of displacement—the system will inevitably encode blind spots that may disadvantage specific populations. Open-source publication is a necessary but insufficient mitigation; active community oversight and ongoing audit by affected populations are essential. The illusion of technological self-reliance. There is a risk that the simplicity and offline capability of VitalGuard may encourage the misconception that technology alone can solve humanitarian challenges. A 54 kB file does not constitute self-reliance. Effective deployment requires training infrastructure, maintenance capacity, legal support, ethical oversight, and—most fundamentally—institutional commitment to the welfare of the populations served. Technology is a component of a support ecosystem, not a substitute for one. Asymmetric design authority. VitalGuard is currently authored outside the displacement-affected communities it aims to serve. The risk of imposing externally designed systems on communities without meaningful agency in the design process is real and cannot be fully mitigated by good intentions alone. The project’s long-term trajectory must include a transfer of design authority through partnerships with local technical actors and community governance structures. This transfer is aspirational at present and represents a research challenge as much as a technical one.

A further ethical consideration concerns the epistemological assumptions embedded in the system’s design. The ten principles presented in Section 3, while grounded in international legal frameworks, reflect a particular normative tradition—one rooted in Western liberal conceptions of privacy, autonomy, and individual rights. In some deployment contexts, community-level decision-making, collective data governance, or alternative conceptions of privacy may be more appropriate. The system’s framework should therefore be understood as a starting point for dialogue with affected communities rather than a universally applicable template. Future iterations must incorporate mechanisms for community-level ethical review that go beyond Western institutional ethics protocols.

Finally, we note the paradox of transparency itself. VitalGuard’s small codebase is fully auditable, which we present as a feature. Yet auditability presumes auditors: individuals with both the technical competence and the institutional authority to conduct meaningful review. In many humanitarian settings, such auditors may not be available, and the theoretical transparency of the system may not translate into practical accountability. Addressing this gap requires investment in local technical capacity—an investment that goes well beyond the scope of any single technology project.

6.4 Future Research Directions

Four directions for future work emerge from the analysis above. First, a controlled pilot deployment of 3–6 months in an actual refugee setting, conducted in partnership with academic institutions and humanitarian organisations, is essential to test the system’s assumptions against field reality. Second, the development of a community self-reliance model—a structured pathway for transferring technical ownership from external developers to local communities—requires interdisciplinary research combining technology transfer theory, participatory design, and development studies. Third, the system’s applicability beyond refugee contexts—to human rights activists, journalists, persons with disabilities, and other populations operating under surveillance—warrants systematic exploration. Fourth, policy engagement with international organisations (United Nations agencies, the International Committee of the Red Cross, the World Health Organisation) is needed to assess whether and how constraint-based AI design might be incorporated into institutional technology adoption frameworks.

6.5 Operational Security and Deployment Guidance

VitalGuard should be treated as decision support rather than a substitute for medical, legal, or protection expertise. In particular, outputs that relate to identity matching, health triage, or risk scoring can create harm if interpreted as authoritative.

From an operational security perspective, the strongest protection offered by VitalGuard is the absence of server-side custody. This reduces mass-surveillance and bulk compromise risk, but it does not protect against device seizure, malware, coercion, or screen capture. Deployments in high-risk settings should therefore assume that local compromise is possible and should avoid entering data that would be catastrophic if exposed.

Where organisations distribute VitalGuard, the distribution channel itself becomes part of the threat model. Offline distribution should use integrity-checked media (for example, signed packages or checksums) to reduce the risk of tampered builds. The codebase should remain readable and unminified to support inspection by local partners.

Finally, governance should be explicit about responsibility. If an organisation deploys VitalGuard, it should publish a short, plain-language statement describing what the system does, what it does not do, and what data is and is not retained. In humanitarian contexts, such clarity is often as important as technical correctness.

7. Conclusion

This paper has presented VitalGuard, an ultra-lightweight offline AI system constrained to 54 kB, and has argued that constraint-based ethical design is a viable and necessary orientation for humanitarian AI. In contexts characterised by surveillance risk, resource scarcity, and acute legal liability, the driving question should not be "What can this system do?" but "What may it ethically be permitted to do, and under what institutional conditions?"

The ten design principles articulated here are not individually novel; many align with existing AI ethics guidelines. The distinctive contribution is the attempt to operationalise them simultaneously within a single auditable artefact, and to show how ethical commitments can be translated into enforceable architectural constraints. We motivate the 54 kB boundary as an auditable description-length proxy, and we characterise privacy primarily through non-collection, non-retention, and no-exfiltration constraints under an explicit threat model. Where stronger privacy guarantees are required for future extensions, formal mechanisms such as differential privacy remain relevant, but they are not claimed without explicit implementation.

The preliminary validation reported here—an external public-sector design review and early-stage academic engagement—should be interpreted narrowly. It suggests that the design framing is coherent and that core constraints are technically inspectable, but it does not substitute for field trials or impact evaluation. Real-world usefulness will depend on participatory deployment, governance arrangements, distribution channels, and the safety practices of organisations and communities.

Ultimately, VitalGuard is not a finished solution. It is an exploration pathway that treats minimalism as accountability and seeks a pragmatic middle ground between high-capability cloud AI and low-trust toolchains that expose vulnerable populations to extraction. We hope the work prompts further research into auditable, offline-first, community-sovereign AI systems that can be responsibly deployed where the stakes are highest.

Acknowledgements

I thank the humanitarian and disability-innovation researchers who provided early-stage, informal feedback on the problem framing and on the risks of technology deployment in displacement contexts. Any remaining errors, omissions, or overstatements are solely my responsibility.

Data and Code Availability

VitalGuard is intended as an auditable software artefact. Public project documentation and offline demos are hosted at https://mcorpai.org/. A public repository containing the project site source and accompanying materials is available at https://github.com/henrymorgan10/mcorpai-org. The specific 54 kB release bundle referenced in this paper is distributed as a single-file offline artefact; a tamper-evident checksum and archival copy can be provided upon request to support independent review.

Because this paper does not report empirical user data or field trials, no personal data or participant datasets are associated with this release.

If future versions incorporate quantitative evaluations or field deployments, the project will provide a data-governance plan covering informed consent, minimisation, retention, access control, incident response, and community oversight, consistent with GDPR-aligned humanitarian data responsibility guidance.

References

[1] Imran, M., Castillo, C., Diaz, F., & Vieweg, S. (2015). Processing social media messages in mass emergency: A survey. ACM Computing Surveys, 47(4), Article 67. https://doi.org/10.1145/2771588

[2] Meier, P. (2015). Digital Humanitarians: How Big Data Is Changing the Face of Humanitarian Response. Routledge.

[3] UNHCR. (2024). "Innovation and the use of technology in forced displacement contexts." UNHCR Innovation Report.

[4] Beduschi, A. (2022). "International refugee law, international human rights law, and artificial intelligence." International Journal of Refugee Law, 34(1), 5-24.

[5] Madianou, M. (2021). "Nonhuman humanitarianism: When AI for good turns out to be harmful." Information, Communication & Society, 24(6), 850-868.

[6] Jacobsen, K. L. (2023). "The politics of humanitarian technology." Security Dialogue, 54(2), 158-175.

[7] Latonero, M., & Kift, P. (2022). "On digital passages and borders: Refugees and the new infrastructure for movement and control." Social Media + Society, 8(1), 1-11.

[8] European Parliament. (2016). "General Data Protection Regulation (GDPR)." Regulation (EU) 2016/679. Official Journal of the European Union.

[9] Access Now. (2021). "Iris scanning of refugees is disproportionate and dangerous — What’s happening behind IrisGuard’s closed doors?" Access Now (press release). https://www.accessnow.org/press-release/irisguard-refugees-jordan/

[10] International Committee of the Red Cross (ICRC) & Privacy International. (2018). "The Humanitarian Metadata Problem: Doing No Harm in the Digital Era." Joint report. https://privacyinternational.org/report/2509/humanitarian-metadata-problem-doing-no-harm-digital-era

[11] Tomašev, N., Cornebise, J., Hutter, F., et al. (2020). "AI for social good: Unlocking the opportunity for positive impact." Nature Communications, 11(1), 2468.

[12] Whittlestone, J., Nyrup, R., Alexandrova, A., & Cave, S. (2022). "The role and limits of principles in AI ethics." AI and Ethics, 2(1), 1-12.

[13] Zuboff, S. (2019). "The age of surveillance capitalism." Public Affairs Press.

[14] Eubanks, V. (2021). "Automating inequality: How high-tech tools profile, police, and punish the poor." St. Martin's Press.

[15] Benjamin, R. (2023). "Race after technology: Abolitionist tools for the new Jim Code." Polity Press.

[16] Raji, I. D., Smart, A., White, R. N., et al. (2020). "Closing the AI accountability gap." Proceedings of ACM FAccT, 33-44.

[17] Selbst, A. D., Boyd, D., Friedler, S. A., et al. (2019). "Fairness and abstraction in sociotechnical systems." Proceedings of ACM FAccT, 59-68.

[18] Warden, P., & Situnayake, D. (2022). "TinyML: Machine learning with TensorFlow Lite on Arduino and ultra- low-power microcontrollers." O'Reilly Media, 2nd edition.

[19] David, R., Duke, J., Jain, A., et al. (2021). "TensorFlow Lite Micro: Embedded machine learning for TinyML systems." Proceedings of MLSys.

[20] Banbury, C. R., Reddi, V. J., Lam, M., et al. (2021). "Benchmarking TinyML systems: Challenges and direction." arXiv preprint arXiv:2003.04821v3.

[21] Reddi, V. J., Cheng, C., Kanter, D., et al. (2022). "MLPerf Mobile inference benchmark." Proceedings of MLSys.

[22] Lin, J., Zhu, L., Chen, W. M., et al. (2024). "On-device training under 256KB memory." Proceedings of NeurIPS.

[23] Jobin, A., Ienca, M., & Vayena, E. (2019). "The global landscape of AI ethics guidelines." Nature Machine Intelligence, 1(9), 389-399.

[24] Floridi, L., Cowls, J., Beltrametti, M., et al. (2022). "AI4People: An ethical framework for a good AI society." Minds and Machines, 28, 689-707.

[25] Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, 2024.

[26] Mittelstadt, B. (2019). "Principles alone cannot guarantee ethical AI." Nature Machine Intelligence, 1(11), 501- 507.

[27] Dwork, C., & Roth, A. (2014). "The algorithmic foundations of differential privacy." Foundations and Trends in Theoretical Computer Science, 9(3-4), 211-407.

[28] Apple Inc. (2023). "Differential privacy technical overview." Apple Machine Learning Research.

[29] Kairouz, P., McMahan, H. B., Avent, B., et al. (2021). "Advances and open problems in federated learning." Foundations and Trends in Machine Learning, 14(1-2), 1-210.

[30] Abadi, M., Chu, A., Goodfellow, I., et al. (2016). "Deep learning with differential privacy." Proceedings of ACM CCS, 308-318.

[31] Kolmogorov, A. N. (1965). "Three approaches to the quantitative definition of information." Problems of Information Transmission, 1(1), 1-7.

[32] Solomonoff, R. J. (1964). "A formal theory of inductive inference." Information and Control, 7(1-2), 1-22, 224- 254.

[33] Li, M., & Vitányi, P. (2019). "An introduction to Kolmogorov complexity and its applications." Springer, 4th edition.

[34] OCHA Centre for Humanitarian Data. (2023). "The state of open humanitarian data." OCHA Annual Report.

[35] Vinuesa, R., Azizpour, H., Leite, I., et al. (2020). "The role of artificial intelligence in achieving the Sustainable Development Goals." Nature Communications, 11(1), 233.

[36] Madianou, M. (2019). "Technocolonialism: Digital innovation and data practices in the humanitarian response to refugee crises." Social Media + Society, 5(3), 1-13.

[37] UNHCR Innovation Service. (2024). "Technology in forced displacement: A responsible approach." UNHCR Report.

[38] ICRC. (2023). "Artificial intelligence and machine learning in armed conflict: A human-centred approach." ICRC Position Paper.

[39] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?: Explaining the predictions of any classifier." Proceedings of ACM KDD, 1135-1144.

[40] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., et al. (2020). "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges." Information Fusion, 58, 82-115.

[41] Gilpin, L. H., Bau, D., Yuan, B. Z., et al. (2023). "Explaining explanations: An overview of interpretability of machine learning." Proceedings of IEEE DSAA, 80-89.

[42] Bishop, C. M. (2006). "Pattern recognition and machine learning." Springer.

[43] Murphy, K. P. (2022). "Probabilistic machine learning: An introduction." MIT Press.

[44] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). "Introduction to algorithms." MIT Press, 4th edition.

[45] Floridi, L. (2020). "The fight for digital sovereignty." Philosophy & Technology, 33, 369-378.

[46] Couldry, N., & Mejias, U. A. (2019). "The costs of connection: How data is colonizing human life and appropriating it for capitalism." Stanford University Press.

[47] United Nations. (1989). "Convention on the Rights of the Child (UNCRC)." Treaty Series, 1577, 3.

[48] United Nations. (2006). "Convention on the Rights of Persons with Disabilities (CRPD)." Treaty Series, 2515, 3.

[49] Sauble, D. (2015). "Offline First Web Development: Design and implement a robust offline app experience using Sencha Touch and PouchDB." Packt Publishing. ISBN 978-1-78588-566-2.

[50] Kleppmann, M., Wiggins, A., van Hardenberg, P., & McGranaghan, M. (2019). "Local-first software: You own your data, in spite of the cloud." Proceedings of ACM Onward!, 154-178.

[51] Linklater, G., Marais, C., Herbert, A., & Irwin, B. (2018). Offline-First Design for Fault Tolerant Applications. SATNAC 2018 Conference Paper. (Full text available via Rhodes University repository/ResearchGate).

[52] Halstead, M. H. (1977). "Elements of software science." Elsevier.

[53] McCabe, T. J. (1976). "A complexity measure." IEEE Transactions on Software Engineering, SE-2(4), 308-320.

Appendix A. Benchmarking and Measurement Protocol

This appendix specifies a minimal protocol for reproducible timing and memory measurements of VitalGuard. The goal is to prevent performance claims from becoming marketing statements and to enable independent verification.

Device classes. Report results on at least two classes: (i) a low-end Android device with 2–4 GB RAM and a mid-range ARM CPU, and (ii) a commodity laptop with an x86 CPU. Report operating system version and browser engine (Chromium, WebKit, or Gecko) with exact version.

Offline condition. Confirm network is disabled during measurement. Record whether airplane mode, firewall rules, or an isolated LAN was used.

Input sizes. For Levenshtein matching, report timings for name strings of length 10, 20, 40, and 80 characters. For naive Bayes, report training sizes N = 50, 200, 1,000 and feature dimensions F = 10, 50, 200. These ranges cover the intended humanitarian "small data" regime.

Measurement method. Use the browser high-resolution timer (performance.now()) for wall-clock timing. Run each case for 100 iterations after a warm-up phase of 20 iterations. Report median, 95th percentile, and maximum.

Memory reporting. Where possible, report peak heap usage using browser performance tools or OS-level process monitoring. State limitations if the platform does not expose reliable memory measures.

Reporting template. Each reported metric should include device, OS, browser, input size parameters, and whether code was minified. If optimisation changes are made (for example space-optimised Levenshtein), the implementation variant must be identified explicitly.

Appendix B. Algorithmic Specifications and Pseudocode

This appendix summarises the core algorithms used by VitalGuard to support independent re-implementation.

Naive Bayes (multinomial) for lightweight classification. Inputs are a feature vector x = (x1,...,xF) and a set of class labels C. Training estimates class priors P(c) and likelihoods P(fi|c) with Laplace smoothing. Inference selects argmax_c log P(c) + sum_i x_i log P(fi|c).

Levenshtein distance for approximate string matching. The standard dynamic programming matrix D of size (m+1) x (n+1) is initialised with D[i,0]=i and D[0,j]=j. Recurrence: D[i,j]=min(D[i-1,j]+1, D[i,j-1]+1, D[i-1,j-1]+cost), where cost is 0 if characters match and 1 otherwise.

Simple rule-based scoring. Where VitalGuard uses threshold rules (for example symptom triage), the exact thresholds and feature definitions should be included in a configuration block within the source code so that local partners can review and adjust them.

```
Pseudocode 1. Levenshtein distance.
function levenshtein(a, b):
    m = len(a); n = len(b)
    create array D of size (m+1) x (n+1)
    for i from 0 to m: D[i,0] = i
    for j from 0 to n: D[0,j] = j
    for i from 1 to m:
        for j from 1 to n:
            cost = 0 if a[i-1] == b[j-1] else 1
            D[i,j] = min(D[i-1,j] + 1,
                         D[i,j-1] + 1,
                         D[i-1,j-1] + cost)
    return D[m,n]
Pseudocode 2. Multinomial naive Bayes inference.
function predict_nb(x, priors, likelihoods):
    best_c = None; best_score = -infinity
    for each class c:
        score = log(priors[c])
        for each feature i:
            score += x[i] * log(likelihoods[c][i])
        if score > best_score: best_score = score; best_c = c
    return best_c
```

Appendix C. Governance and Deployment Checklist

VitalGuard is designed to minimise harm through technical constraints, but deployment risk cannot be eliminated by engineering alone. This appendix provides a governance checklist suitable for NGOs, local partners, and public institutions.

Clarify purpose and boundaries. Publish a plain-language statement of intended use, non-intended use, and known limitations. Avoid framing the system as medical diagnosis, legal adjudication, or identity proof.

Consent and agency. Ensure that users understand that the tool runs locally, what inputs are required, and that they may refuse use without penalty. In humanitarian settings, consent is often compromised by power asymmetries; this must be acknowledged.

Safeguard against coercion. Establish procedures for responding to requests from authorities or armed groups that seek to use the tool for surveillance or profiling. The organisation should have a documented refusal policy.

Integrity of distribution. Distribute verifiable builds, maintain checksums, and document the provenance of any modified versions. Avoid bundling with third-party analytics or ad frameworks.

Local adaptation. If local partners adjust thresholds or rules, require documentation of the change, rationale, and a basic harm review. Transparency is more important than optimisation.

Accountability. Assign a responsible owner for incident response, feedback intake, and periodic review. Even a "no data" tool can cause harm through decision influence.

Appendix D. Extended Principle-to-Constraint Mapping

This appendix expands the mapping between ethical principles and concrete constraints to support reviewers who prefer narrative reasoning over checklists. The mapping is written to be auditable: each principle is paired with one or more observable implementation constraints and with an explicit failure mode.

Principle 1, protect dignity and agency, is enforced by keeping the system local, optional, and non-mandatory. Failure mode: a deployment that makes tool usage a prerequisite for aid access converts decision support into coercion.

Principle 2, function in infrastructure absence, is enforced by eliminating runtime dependencies on networks, remote APIs, and third-party libraries. Failure mode: a "mostly offline" design that silently fails when connectivity drops can create dangerous false confidence.

Principle 3, resist surveillance by design, is enforced by no-exfiltration constraints and by refusing server-side logging. Failure mode: adding analytics, remote debugging, or silent telemetry—even for benign monitoring—creates a data exhaust that can be repurposed.

Principle 4, be understandable and contestable, is enforced by the auditable boundary and deterministic algorithmic core. Failure mode: integrating opaque model weights or remote inference undermines the ability of local partners to challenge outputs.

Principle 5, minimise dependency and lock-in, is enforced by plain web standards (HTML and JavaScript) and by the absence of proprietary SDKs. Failure mode: depending on a vendor runtime can transfer governance power away from the community.

Principle 6, minimise legal liability, is enforced by avoiding organisational custody of personal data and by keeping processing session-bounded. Failure mode: storing identifiers locally for convenience can still create meaningful harm if devices are seized.

Principle 7, fairness under scarcity, is enforced by restricting features to inputs that are realistically available and by avoiding proxy variables that encode protected characteristics. Failure mode: using convenience proxies (for example location, device identifiers, or language) can reproduce discriminatory patterns.

Principle 8, fail safely, is enforced by conservative defaults and by presenting outputs as suggestions with uncertainty. Failure mode: presenting scores without context encourages over-trust.

Principle 9, enable local adaptation, is enforced by keeping thresholds and rule parameters explicit in code and by encouraging documentation of changes. Failure mode: hidden parameters and minified code prevent local governance.

Principle 10, be auditable by one person, is enforced by the 54 kB constraint, readable formatting, and removal of external dependencies. Failure mode: code growth beyond the audit boundary recreates the opacity of large systems.

Across these principles, the central argument is that ethical claims must be testable. VitalGuard treats size, offline operation, and the absence of exfiltration and persistence as verifiable constraints, so that disagreement does not depend on trusting the author’s intent.