ChatGPT

LifeSync Trust System Developer Specification

This document defines the components, data models, and flows for the LifeSync Trust System, an individual-centric trust network that scales across households, communities, events, and organizations. Each section below describes a major subsystem with its responsibilities, data, and extensibility.

Trust Seal System

- **Trust Seals (and Public Pages):** Every user, household, event, community, or organization is issued a LifeSync *Trust Seal* – a digital identity badge linked to a public web page. For example, a household admin can create a public page containing visitor instructions, delivery protocols, emergency contacts and optional guest Wi-Fi/Bluetooth details [1]. A unique QR code (and matching NFC tag) is auto-generated for each seal [2] [3]. Scanning the QR/tapping NFC opens the seal's public page (hosted online or in-app, cached offline). Different variants can be defined (e.g. "Daily Instructions" or an event-specific page) [1]. Public entities (schools, businesses, community centers) can likewise create verified pages and display the LifeSync seal [4].

- **QR/NFC Integration:** Each seal produces a **Permanent QR code** (static link to the page) and may optionally produce a **Dynamic QR code** for time-bound or temporary use [2]. Permanent codes are printable and usable offline; dynamic codes can auto-expire or be disabled (useful for single-day events). NFC tags carry the same URL payload as the QR, allowing one-tap access for compatible phones [3]. All code scans trigger the same workflow: open the seal's content and join the local sync (see next section).

- **Display and Symbol:** The Trust Seal includes a recognizable visual icon that can be displayed physically (at gates, entrances, building lobbies) and digitally. This symbol signals that the space is part of a verified, coordinated network [5]. Psychologically, it warns ill-intentioned actors that the community is "connected and responsive" and reassures guests that the space is safe [6].

- **Creation and Lifecycle:** A personal Trust Seal is automatically generated when a profile is created, even for guest accounts. New seals (for events, households, or organizations) can be created via the app UI by an authorized administrator. Each seal stores metadata such as owner ID, seal type (individual/household/event), custom instructions text, and access rules. Owners can customize the seal's content at any time (e.g. updating emergency procedures or daily instructions). To revoke a seal, the admin simply disables its code or marks it inactive; the system can then generate a replacement seal (new QR) if needed. All versions of a seal (active or expired) should be logged for audit.

- **Data Responsibility:** The Trust Seal component owns its page data (instructions, contacts, access rules), the seal's identifiers (QR link, NFC ID), and its status (active/inactive). The app backend must store these fields securely and replicate them to peer devices for offline use. Access to modify or view a seal is controlled by the ownership permissions.

- **Extensibility:** The design allows new seal types or hierarchy levels. For example, future extensions could introduce *building-level* or *street-level* seals. The system should allow multiple admins per seal (for multi-family households or co-managed events). New fields (e.g. language

preferences or accessibility notes) can be added to the seal's public page without changing core logic.

Guest and Visitor Syncing

- **Scan and Join Flow:** When a visitor scans a Trust Seal's QR code or taps its NFC tag, the LifeSync app (or web PWA) opens the linked content and automatically attempts to join the local mesh network. Specifically, the visitor's device starts a Bluetooth or Wi-Fi Direct discovery process to connect with a nearby mesh node (for example, the household's router or admin phone) [7]. If no local mesh is available but internet is connected, the app falls back to minimal server sync (using lightweight endpoints on a scheduled basis) [8]. In either case, the visitor gains access to the seal's information (instructions, contacts, etc.) and is logically "added" to the group associated with that space.

- **Guest Portal and Mesh Access:** Upon connecting, the visitor sees a **guest portal** interface that presents a welcome message, the household/community info, and emergency protocols [9]. This portal can include an optional contact form or a "check-in" button to signal arrival. All portal content is cached locally so it is usable offline [7]. At this point, the visitor's device begins participating in the local mesh: it advertises its presence and can send/receive messages to others in range. This makes the guest part of local communications (e.g. group chat or alerts) even without internet.

- **Offline-First Sync:** The syncing model is offline-first: devices replicate data peer-to-peer. Approved visitors and residents act as "seed" nodes, relaying messages. For instance, a router hub can run OpenWrt and serve as a local content portal and sync point [8]. Devices opportunistically sync via BLE or Wi-Fi Direct when in proximity. This ensures continuous access: for example, even if cellular/internet is down in an event hall, the local mesh still disseminates updates.

- **Use Cases (joining groups):** In practice, scanning a seal code creates or joins a *Sync Group*. For example, if a taxi rider creates a temporary seal for that cab, co-passengers who scan it become part of a mesh group sharing location and status [10]. Similarly, a community leader can display a meeting's seal; arriving participants scan it, automatically "sign in" to the meeting group, receive attendance tracking links, and access live event information [11]. In corporate buildings, visitors scan the building's LifeSync seal to automatically receive safety protocols and connect to company channels (e.g. security office) [12].

- **Data Responsibility:** The Guest Sync component handles the temporary guest identity and their mesh connection status. It manages session tokens for guests, ensures visitor data (if any) is recorded only for the duration needed, and cleans up after. Devices buffer any collected information (e.g. check-ins) and later sync with the cloud if possible.

- **Extensibility:** New transport layers (e.g. NFC, QR alternatives) can be added by implementing their discovery payload. Additional guest features (like a captive web portal or voice assistant mode) can be integrated without altering core mesh logic.

Governance Flows

- **Hierarchical Approval Chains:** Event requests, resource access, and similar actions follow multi-step approval flows defined by household and community hierarchy. For example, if a child

creates an event request, the system first routes it to their linked parent or guardian [13]. Once the parent approves, if the household has multiple family units, the request then goes to that unit's administrator; after unit approval it moves up to the main household head [14] [15]. Finally, if configured as a community or public event, the request may escalate to community administrators or be broadcast to neighbors (e.g. requiring a vote).

- **Multi-Household Units:** The system supports compound households. As one user notes, "many households have multiple families...each unit will have its own administrator...rental units and extended families" [15]. In practice, the database should represent each household as a collection of units (e.g. 'Unit A', 'Unit B'). Only the main household admin has ultimate authority, but each unit admin can approve actions for members of that unit. Administrators can dynamically add new units (e.g. when renting out a flat) and invite people into them [15]. After invitees accept, admins can create communication groups for each unit or the entire household.

- **Family Tree and Roles:** Each user has defined kinship links (parent, child, spouse) that determine authority. Children must be linked to at least one adult; the app uses this "parent link" to route child requests [13]. The admin interface allows managing family roles (for example, assigning a 'father', 'mother', 'grandparent', etc.). Changes to these roles update the approval logic automatically.

- **Voting and Consensus:** For larger or public events, the system can require community consensus. E.g. one use case suggests a *60% vote-based* rule: "vote based (60%) validation" for events like group trips [16]. In implementation, that could mean an event becomes approved when at least 60% of invited responsible adults grant consent. These parameters should be configurable by the household/community.

- **Implementation & Escalation:** The Ekhaya subsystem governs these flows, notifying each party in turn and logging decisions [17]. Once a request is fully approved at a level, it is posted or announced: for example, a now-approved party can be published on the household's public page or sent to the invited guest list [18]. If any step is denied, the process halts. All intermediate states and timestamps must be recorded so audits can trace who approved or declined.

- **Extensibility:** The workflow engine should allow adding new approval stages or roles. For instance, one could extend it to link in school administrators for school-related events. New conditions (like requiring 2 authorized adults to approve child trips, or adding external agencies) can be plugged in via configuration files or rules scripts.

Emergency Sync Protocols

- **Emergency Activation:** A user can enter an emergency mode ("LifeAlert") at any time. This shares live location and status with designated trusted contacts and emergency personnel. For example, a hiker can enable follow-me tracking, sending periodic GPS updates to family members. Alternatively, pressing a panic button instantly broadcasts a distress alert (with last known location) to the mesh group and to cloud alert services.

- **Geofencing and Check-Ins:** The app supports geofencing: admins can define safe zones (e.g. home area, school campus, event perimeter). If a tracked user crosses a geofence or fails to report in by a scheduled check-in time, an alert is automatically sent to next-of-kin or community responders. In practice, if "someone breaks from the group," the system flags it [19]. For

example, if a child leaves the permitted zone or doesn't check in after school, parents receive an escalation.

- **Local Mesh Resilience:** LifeSync is designed for network outages. In an emergency, devices form an ad-hoc mesh (Bluetooth/Wi-Fi) to relay messages. As one scenario describes, "even if networks are down… people in an emergency hall can create their own mesh network…using both Bluetooth and WiFi" [20]. This ensures that SOS messages and location pings propagate through the local peer network even without internet or cell service.

- **Group Emergency Sync:** During an active emergency (e.g. a missing child at an event), all group members share an "emergency sync" session. Each participant's location and status is visible to the designated monitors (parents, event staff) via a shared interface [21]. Status updates (like "all accounted for" or incident reports) can be broadcast through the mesh so that everyone stays informed without making phone calls. This enables multiple responders to coordinate (for instance, in a stadium search, separated parents can use the mesh chat to reunite [22]).

- **Escalation Workflow:** If the initial alert is not addressed, the system escalates. For example, if a user's check-in fails twice, it might notify a secondary contact or trigger a neighborhood alert. The exact escalation tree (parent → household head → community admin → 911) should be configurable. Each escalation sends location and context to the next level until someone acknowledges.

- **Data Responsibility:** The Emergency module stores active alert data (location history, timestamps, responder actions) and pushes notifications to devices. It must track who has acknowledged each alert. Privacy rules apply: emergency location sharing is only active while alerts are on; once a situation is cleared, live tracking ends and data is archived or deleted according to policy.

- **Extensibility:** The protocol allows adding new alert channels (SMS, sirens, IoT beacons). It should also integrate with external services (police dispatch, local radio) if available. New sensor inputs (e.g. fall detection) can trigger the same workflow.

Trust Score System

- **Reciprocal Ratings:** LifeSync implements a 10-criterion rating system for peers (e.g. ride-sharing drivers, delivery agents, event hosts). After each verified interaction, both parties rate each other on predefined metrics (such as reliability, safety, communication). These ratings aggregate into each user's public **LifeScore** [23]. Importantly, ratings count only when an interaction is completed through LifeSync (to prevent spam or fake reviews). All ratings are mutual: a rider rates a driver and vice versa, and both scores adjust accordingly.

- **Tiered Levels:** Trust scores map to intuitive tiers. For example, scores 0–50% fall into an **Amber** tier ("needs improvement"), 50–70% into **Growth** ("on track"), 70–90% **Standard**, and 90–100% **Excellent** [24]. To reach each tier, users must not only have a high percentage but also a minimum number of ratings (e.g. 25 ratings to clear the first tier, 50 ratings for the next) [23]. The thresholds are visible on profiles (e.g. progress bars). As one user described: a score below 70% flags the need for improvement, 70–90% meets standards, and above 90% is the goal [24].

- **Community Endorsements:** Separate from personal ratings, entire communities can endorse a person once a consensus is reached. Practically, if ~70% of active households in a community

give positive feedback on an individual's service, that person earns a "Community Endorsed" badge [23]. This is calculated automatically (not manually assigned) and is revoked if scores fall. Notably, community endorsements do *not* directly boost the LifeScore percentage; they act as a qualitative mark of trust from the collective (an additional branding layer) [25] [26].

- **Computation:** The app back-end stores each rating record (with metadata of date, criteria scores, transaction ID). A TrustScore module periodically recalculates the percentage for each user: sum of positive criteria / total possible across all ratings. After every new rating, recalc and update the profile's LifeScore and tier. Endorsements are similarly recomputed whenever community feedback changes. No single admin can override these scores – they depend purely on user feedback as designed [23].

- **Data Responsibility:** The Trust Score component owns rating data and aggregated scores. It must ensure ratings are tied to verified interactions (e.g. a completed ride ID) to prevent falsification. All raw ratings are stored (for audit) but only the aggregated score and tier are exposed publicly.

- **Extensibility:** More rating criteria can be added later (e.g. cleanliness, friendliness). Thresholds (50/70/90) are configurable. The system can also incorporate other signals (like identity verification status) into trust calculations if needed.

Public/Private Visibility Controls

- **Field-Level Privacy:** Every profile, household page, and event page has granular visibility settings. Owners can choose which fields to make public on their Trust Seal page. Possible fields include personal contact info, guest Wi-Fi password, emergency contacts, detailed instructions, and live GPS availability. By default, only minimal info (like emergency contacts and generic instructions) is shown on public pages [1] [27]. Owners can toggle each field on/off; hidden fields are never transmitted to visitors.

- **Event Visibility:** When creating an event or community announcement, the creator selects public or private. A public event is posted to the community's public page and anyone can join/sync; a private event bypasses the page and instead sends unique invite links or QR codes to specified guests [28]. For instance, a child's approved birthday party might be marked private so that only invited families receive the sync code, whereas a street cleanup event could be public on the neighborhood page.

- **GPS Sharing Control:** Users can separately enable or disable live location sharing. For example, a parent may choose to share a child's location only during an active sync session (like during a trip) and not otherwise. The app respects this flag: if GPS sharing is off, no location data is shared at all (even in emergencies). If on, only designated trusted contacts see the real-time map feed.

- **Reciprocity on Sync:** When two users connect via LifeSync, each one can see only the data the other has agreed to share. For example, if Alice scans Bob's seal, Alice will see Bob's public profile fields but not any of Bob's private notes or contacts. Similarly, Bob will not automatically see Alice's details unless Alice has made them public. This two-way mutual-consent model ensures "you only see what I share" – no unilateral access. (All shared fields follow the settings described above.)

- **Data Responsibility:** The system enforces these visibility rules at the data layer. The Seal/Profile service stores privacy flags for each field. When serving a page or mesh session, only allowed fields are included in the payload. All other fields are filtered out. If a field's visibility is changed, the system must update stored visitor data accordingly (e.g. stop broadcasting a field's value).

- **Extensibility:** New data categories (e.g. medical info, allergy alerts) can be added with corresponding privacy toggles. New visibility scopes (friends-only, household-only) could also be implemented by extending the privacy model.

Contact Role Management

- **Role Assignment:** When adding a contact (either by importing address book or scanning a personal QR code), the app prompts the user to assign one or more roles (kinship type, household member, emergency authority, service provider, etc.). These roles are used in governance and sharing logic. For example, tagging someone as "Emergency Contact" automatically grants them permission to receive alert notifications; marking someone as "Child" flags them as needing guardian approval [13].

- **Household Units:** As noted, a household can consist of multiple units (nuclear families, tenants, guests) [15]. Each unit can have its own contact list. The owner can create named units (e.g. "Unit A – Smith Family", "Unit B – Apartment 1") and add people into each. Invited contacts then synchronize with the household according to their unit membership. For instance, a tenant might be invited with limited sync rights (no parental authority) as opposed to an actual family member.

- **Family Tree UI:** The LifeSync app provides an interactive family/household tree view. When contacts are added or roles changed, the tree updates in real time. This visualization helps admins see, for instance, who the parents and children are, and who is in each sub-unit. Changes to roles (like setting a new guardian) automatically propagate through the system's logic [15].

- **Access Rights:** Each role carries default permissions. For example, a contact marked as "Parent" can approve events or see a child's details; an "Emergency Responder" can view safety protocols; a "Neighbor" might only see community events. These mappings should be configurable. Removing a contact or lowering their role should immediately revoke any access they had (e.g. closing their mesh session, removing them from chat groups).

- **Sync Reciprocity:** In every peer-to-peer sync (QR code exchange), both parties only learn what the other has shared. The system never exposes more than the intersection of each user's shared data. This applies even to contacts: when synchronizing two users, only the attributes they mutually agreed to expose are exchanged [29]. Any contact lists or private notes remain local unless explicitly shared.

- **Data Responsibility:** The Contacts module stores all imported contacts, their assigned roles, and their relationship links (e.g. parent-of, child-of). It provides APIs for querying who is in a given household/unit. Role changes must update downstream modules (governance, visibility, trust) as needed.

- **Extensibility:** New role types can be added (e.g. “Coach”, “Group Organizer”) along with custom access rules. The UI can allow admins to define role groups (e.g. an “Emergency Team” composed of several contacts).

Device Tiering and Accessibility

- **Capability Detection:** On first launch, LifeSync measures the device’s hardware and OS features (RAM, storage, Android version, presence of BLE/Wi-Fi Direct, GPS) [30]. It then assigns a device **Tier** (Lite, Intermediate, Full) and suggests it to the user (with override option). The choice of tier determines feature availability and resource usage limits.

- **Tier Definitions:** - *Lite Tier* targets Android Go and low-end phones: minimal UI, basic sync (no multi-hop mesh), and PWA mode for limited devices (or even feature phones via browser). Heavy features (like background mesh scanning, video calls, or large attachments) are disabled. The app uses lightweight layouts and aggressive caching.

- *Intermediate Tier* is for mid-range devices: enables local mesh communication (BLE/Wi-Fi Direct 1–2 hops) and offline content, but may limit simultaneous connections or background jobs.

- *Full Tier* is for high-end devices: all features enabled, including multi-hop mesh, high-resolution media, and continuous background sync.

- **Progressive Enhancement:** The app hides or disables features that the device cannot support. For example, on a browser-only device, mesh chat options are hidden and the user is offered a read-only PWA experience [31]. Users are informed of any disabled features (e.g. “Offline mesh chat not available on this device”). Clear prompts explain how to switch tiers if needed (e.g. “Tap here to switch to Lite mode for better battery life”).

- **Performance Safeguards:** Special optimizations are applied for Lite/Android Go devices: use minimal bitmap sizes, downsample images, avoid heavy animations, and batch network operations [32]. The app pauses nonessential tasks on low battery or low memory [32]. It also periodically cleans up old cached data to free space. Users may be prompted to clear storage or switch to Lite mode if their device is constrained.

- **Accessibility:** The interface uses large touch targets and an offline help section [32]. There is a low-literacy mode (with simplified text and voice prompts) for those who need it. All images and icons have alt-text, and color contrast meets accessibility standards. Map elements and emergency notices are viewable by screen readers.

- **Offline Capability:** Regardless of tier, core safety features work without internet. All seal pages, contact info, and approved workflows are cached on-device. Any data generated offline (check-ins, messages) is queued and synced when connectivity returns [31].

- **Extensibility:** The tier system can accommodate new categories (for example, a “Ultra Lite” text-only mode for feature phones). Additional device capabilities (like NFC payment) can be detected and integrated by extending the capability check routine.

Naming and Branding

To ensure a cohesive user experience, all features will use LifeSync-branded names. Proposed conventions:

- **LifeSeal (Trust Seal):** The official LifeSync emblem/badge placed on pages and QR codes, signaling a secure, trusted space.
- **LifeKey:** The user's personal digital identity token. (Could also refer to QR codes or credentials that "unlock" connections.)
- **LifeSync Page:** The public profile page for a user/household/event (formerly "public page").
- **SyncCode:** The QR/NFC code associated with a LifeSeal, used for scanning and syncing.
- **LifeAlert:** Emergency mode (panic button, SOS channel) and the associated workflows.
- **LifeGuest Portal:** The local offline-access interface that visitors connect to via a seal.
- **LifeScore:** The trust rating displayed on a user's public profile.
- **LifeContact:** The contact role management area (family tree interface).
- **LifeChat / LifeCall:** Names for the offline messaging or calling feature (if needed).

Each name begins with "Life" (the LifeSync brand) and intuitively describes its function (e.g. LifeAlert for emergencies). Consistent color schemes and icons (e.g. a shield for LifeSeal, a key symbol for LifeKey) will reinforce these terms in the UI.

Sources: LifeSync's design is informed by the detailed specifications and example flows in the LifeSync Foundational Thoughts document [27] [1] [5] [20] [13] [23] [24]. These sources describe the envisioned QR/NFC logic, mesh connectivity, governance approval paths, emergency protocols, and trust scoring that this specification formalizes.

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] Foundational Thoughts.txt
file://file-MzeoGY7TmBZNnYn5SZokf5