Notifications Module – Complete Technical, Product, and RAG Documentation (v1)

0. Purpose and Scope

The Notifications module is the unified activity center for ConnecWrk. It collects, categorizes, renders, and routes *all* significant events that require a user's attention — social actions (connection requests), opportunity alerts (jobs / bids / milestones), platform signals (premium prompts), and engagement signals (reactions, comments, etc.).

This doc covers: - UX behavior on **/notifications/** (screenshot reference) - API contracts (read, pagination, mark-as-read, mark-all-read) - Notification data model and type mapping - Real-time surfaces (chat handoff) - State management & interaction flow - Performance and security concerns - Full RAG ingestion spec for this module (namespaces, chunking rules, metadata requirements)

This doc should be ingested into RAG as ground truth for "Notifications", "Activity", "Alerts", "Mark all read", and "I got a notification about X" support questions.

---

1. High-Level UX / Page Anatomy

Route: `/notifications/`

Layout (3-column): 1. **Left Sidebar (User panel)** - Profile card with avatar, name, links ("My Pages"), connection count, profile views, profile completion bar, and quick CTAs (e.g. "Create your business page", "Post your assignment"). - Nav shortcuts: MSME Connect, Freelancer Connect, Assignments, Talent Connect, MSME Jobs, Articles and Trivia. - *Same component used on Timeline & Contacts; refer to Timeline RAG namespace* `timeline.left-sidebar` *for deep contract.*

1. **Center Panel (Notifications list)**
2. Header: "Notifications"
3. List of notification rows in reverse chronological order (newest first)
4. Each row shows:
    - Icon (contextual: e.g. briefcase for job, user icon for connection)
    - Rich text sentence with bold actor names (e.g. `Your contact Tarun Grover has posted a job.`)
    - Inline CTA link (e.g. `Click here to view job details`)
    - Timestamp on far right (e.g. `2mo`)
5. Read state styling: unread = highlighted / bold; read = normal weight.

6. Row click / inline link click will navigate to relevant detail page (job post, profile page, etc.) AND mark notification as read.

7. **Right Sidebar (Engagement widgets)**

8. Premium upsell banner (“Unlock Exclusive Features With Premium”) – opens `/premium-membership/`.

9. "Invite People To Join ConnecWrk" – opens Invite flow.
10. Featured MSMEs block (scrollable list with business name + tagline, plus "View More").
11. Featured Freelancers block (avatar, name, role).
12. *Same component as Timeline Right Sidebar; refer to RAG namespace* `timeline.right-sidebar` *for full data contracts and performance rules.*

---

2. Notification Types (Semantic Model)

Every notification is an event. We normalize them into categories so UI and routing logic can behave predictably.

2.1 Social / Network

- **Connection Request Received**
- Copy sample: `Tarun Grover has sent you a request to connect.`
- CTA: `click here to view profile and Confirm/Decline the request.`
- Action route: user profile page + pending-connection UI (from Contacts module).

- On click: mark-as-read.

- **Connection Request Accepted / New Connection**

- Copy sample: `You are now connected with <Name>`.

- CTA: open contact profile or /contacts/ tab.

- **Profile Viewed** (if implemented)

- Copy sample: `<Name> viewed your profile`.
- CTA: view profile analytics or upgrade to Premium if analytics gated.

2.2 Opportunity / Workstream

- **Job Posted by Contact**
- Copy sample: `Your contact Tarun Grover has posted a job.`
- CTA link: `Click here to view job details.`

- Destination: job detail route from Jobs / Assignments module.

- **Project / Assignment Milestones** *(future / implied by system design)*

- Milestone created / payment released / bid received.

- CTA: go to assignment / project thread.

- **Bid Updates / Proposal Accepted** *(Assignments flow)*

- CTA routes to assignment or bid list.

2.3 Engagement / Content

- **Reaction on Post**
- Someone liked / clapped / supported your timeline post
- CTA routes to that post on `/timeline/` with focus on that card.

- **Comment / Reply on Post**

- CTA routes to the discussion thread within the post.

2.4 Events / Activity Reminders

- "<Speaker> is hosting an event", "Event starts in 1 hour", "You're added as attendee".
- CTA routes to `/events/:eventId`.

2.5 System / Platform

- Birthday reminders for contacts
- Policy / security notices (password updated, session activity)
- Premium nudges ("Your trial ends soon") – may deeplink into `/premium-membership/`

---

3. Data Model / Shape of a Notification Object

> NOTE: This is reconstructed contract-level shape we expect from `/api/setting/notificationslist` based on current UI behavior and downstream usage.

```
{
  "notificationId": "noti_12345",          // unique id, string
  "type": "job_posted",                    // semantic category key
  "actorUserId": "user_789",               // user who triggered it (optional
for system notices)
  "actorName": "Tarun Grover",             // display name
  "actorAvatar": "tarun.jpg",              // optional avatar/img
  "copyHeadline": "Your contact Tarun Grover has posted a job.",
  "copyCtaText": "Click here to view job details.",
  "ctaRoute": "/jobs/job-detail?jobId=abc123", // where to go when clicking
  "createdAt": "2025-08-01T12:30:00Z",     // source timestamp
  "timeAgo": "2mo",                        // UI-friendly time label
  "readYN": "N",                           // "Y" or "N" for read-state
  "context": {
    "jobId": "abc123",                    // domain-specific context (job,
bid, post, etc.)
    "postId": null,
    "connectionRequestId": null,
    "eventId": null
  }
}
```

Fields we **must** persist for RAG quality: - `type` (to enable intent classification downstream) - `copyHeadline` + `copyCtaText` (exact phrasing matters for user Q&A like "why did I get this alert?") - `ctaRoute` (so assistant can describe where they'll land) - `readYN` (so assistant can answer "why is this still bold?" / "how to mark read?") - `timeAgo` (for UX parity)

Security note: `actorUserId` is PII. We NEVER surface `actorUserId` or raw `actorAvatar` directly to other users in assistant responses unless user already has access in UI context.

---

4. Core Backend Routes / Contracts

4.1 Fetch Notifications (Paginated)

Endpoint: `GET /api/setting/notificationslist`

Auth: Bearer token **required**.

Query Params: - `page`: integer, required. (1-indexed) - `pageSize`: integer, optional (defaults ~20)

Response (representative):

```
{
  "success": true,
  "data": [ { ...NotificationObject }, { ... } ],
  "currentPage": 1,
  "totalPage": 10,
  "hasMore": true
}
```

Error cases: - `401 Unauthorized` → missing/invalid token - `500 Internal Server Error` → fetch failure

Frontend behavior: - On mount: load `page=1` - Infinite scroll: use IntersectionObserver at list bottom to request `page+1` when visible - De-dupe by `notificationId` on append - Stop when `hasMore === false`

4.2 Mark Single Notification as Read

Endpoint: `GET /api/setting/notificationReadUpdate?notificationId={id}`

Purpose: mark 1 notification as read.

Auth: Bearer token required.

Side effects: - Backend flips `readYN` to "Y" - UI immediately updates local state (optimistic update) - Optional: call lazily when user - clicks CTA link - opens profile/job/etc. from that row

Error cases: - `400 Bad Request`: missing `notificationId` - `401 Unauthorized`: invalid token - `500 Internal Server Error`: DB update fail

> If mark-read fails, we do *not* block navigation. We just retry marking read silently in background.

4.3 Mark ALL Notifications as Read (bulk)

Endpoint: `POST /api/setting/notificationReadUpdateAll`

Auth: Bearer token required.

Body (expected):

```
{
  "markAll": true
}
```

Response (representative):

```
{
  "success": true,
  "message": "All notifications marked as read"
}
```

Usage: - Triggered by "Mark all as read" control (if surfaced in header / kebab menu) - Optimistically set every `readYN` to "Y" in local state

Error cases: - `401 Unauthorized` - `500 Internal Server Error`

4.4 Chat / Messaging Hand-off (Realtime)

Some notifications represent direct outreach that implies "you should talk to this person now" – e.g. connection accepted, project bid, milestone dispute.

For those, the UI may deep link into chat.

Realtime transport: Firebase Firestore - Collection path: `chats/{chatId}/messages` - We subscribe via Firestore listeners for: - new incoming message - last message preview / unread state

Security: Firestore security rules enforce that only participants can read/write that chatId.

In RAG: We NEVER expose Firestore path names or chatId values unless the asking user already sees that chat thread in UI.

5. UI / Frontend Behavior Details

5.1 Notification Row Rendering

For each notification: - LEFT: context icon - contact request → user icon / handshake icon - job posted → briefcase / work icon - system → info icon / bell icon - BODY: - Bold actor name ("Tarun Grover") - Plain body copy - Smaller subtext line for CTA (`Click here to view job details.`) - RIGHT: - `timeAgo` (e.g. `2mo`, `3mo`, `5h`, `1d`) - Right-aligned, muted gray

Unread styling: - row background slightly tinted OR bold text on headline line. - When clicked → optimistic `readYN = Y`.

5.2 Navigation / Click Zones

- We allow inline deep links inside the body copy (`click here` etc.).
- We can also make the whole row clickable (implementation detail) – but if we do, the click must route to the most relevant destination:
- Job posted → job detail page
- Connection request → that user's profile (with Accept / Decline)
- Project update → project milestone view, etc.

5.3 Empty State

When no notifications: - Show message like `You're all caught up!` - Hide infinite-scroll sentinel - Still render right sidebar for discovery and premium upsell

5.4 Error / Loading States

- While loading: skeleton rows OR spinner at top of list section
- On fetch error:
- Show inline error toast `Couldn't load notifications`
- Retry on scroll OR manual refresh
- Still render cached notifications if present

6. State Management / Frontend Logic

6.1 Local State Shape (conceptual React-ish)

```
const [notifState, setNotifState] = useState({
  items: [],           // array<NotificationObject>
  page: 1,
  hasMore: true,
  loading: false,
  error: null,
});
```

6.2 Infinite Scroll

- Use an `IntersectionObserver` on a sentinel div rendered after the list.
- When sentinel enters viewport AND `hasMore` AND not `loading` → fetch next page.
- Append items, dedupe by `notificationId`.
- Stop observing when `hasMore` becomes false.

6.3 Optimistic Read Updates

When user interacts with a row: 1. Immediately set that row's `readYN` to `"Y"` in local state. 2. Fire `notificationReadUpdate` in background. 3. Ignore non-200 failure unless 401 (token invalid).

6.4 Bulk Read

When "Mark all as read": 1. Flip all `readYN` to `"Y"` in local state. 2. Fire bulk API. 3. On failure, optionally toast but do not revert UI.

---

7. Security, Privacy, Compliance

7.1 Authentication

- ALL calls to `/api/setting/notificationslist`, `/api/setting/notificationReadUpdate`, `/api/setting/notificationReadUpdateAll` require a valid Bearer token.
- The Bearer token is read from `localStorage` just like the rest of the app.

7.2 Authorization

- Backend MUST validate that `notificationId` belongs to the requesting user before updating read state.
- A user must NEVER be able to mark someone else's notification as read or fetch someone else's notifications.

7.3 PII Exposure Rules

- We show `actorName` (like `Tarun Grover`) because that is already visible to the user in their in-app UI.
- We **never** surface:
- personal emails
- phone numbers
- internal IDs (actorUserId, jobId, etc.) directly in assistant responses unless the user already has that exact ID on-screen or in previous turn AND it's required for an allowed workflow.

7.4 Notification Retention / Audit

- Read status is persisted.
- Timeline of events (e.g. "2mo", "3mo") implies historical retention; exact retention policy is backend-defined.
- For RAG answers, assistant MUST avoid promising retention windows unless documented.

7.5 Rate Limiting / Abuse Prevention

- Fetch list is paginated → implicit throttling.
- `notificationReadUpdate` could be spammed; backend should throttle repeated calls for same notificationId.
- Firestore chat access is governed by Firestore security rules for real-time messaging.

---

8. Performance / UX Notes

- **Pagination Limit:** ~20 rows per page.
- **IntersectionObserver:** avoids explicit "Load more" button.
- **Optimistic UI:** user gets immediate visual feedback when marking read.
- **Shared Sidebars:** Left and Right sidebars are re-used components from Timeline. Reuse reduces bundle size and cognitive switching.
- **Minimal Re-renders:** dedupe logic prevents duplicate notifications causing extra layout thrash.

Accessibility: - Each notification row is logically a clickable region with ARIA role `link`. - Timestamps use short relative labels ("2mo", "3m", "1d"), which are screen-reader friendly.

---

9. RAG Ingestion Strategy for Notifications

This section defines how this doc and related chunks get ingested into the Retrieval-Augmented Generation pipeline so assistants can answer: - "Why did I get this notification?" - "How do I mark them as read?" - "What does 'Tarun Grover has posted a job' mean?" - "Where do notifications live in the UI?"

9.1 RAG Namespaces (Notifications)

We store different slices of knowledge in distinct namespaces to improve retrieval precision.

`notifications.product` - High-level behavior, UX, purpose of /notifications/ - User-facing descriptions ("what does this mean", "how do I use it") - When assistant answers onboarding / helpdesk style questions, it should mostly retrieve from here.

`notifications.api` - All backend routes, params, response shapes, auth requirements, error codes - Used when engineers / QA / support ask "which endpoint marks a notification read?"

`notifications.ui-components` - Left Sidebar summary, Notification Row rendering rules, Right Sidebar widgets, Premium upsell, pagination UX, read-state styling, click routing rules, empty state UX. - Used when designers/front-end/dev QA ask "how should unread look" / "what happens on click".

`notifications.security-privacy` - Auth required, PII rules, who can see what, what we never expose. - Used when compliance / CS asks "can user A see user B's notifications?" or "can we show phone numbers in notifications?".

`notifications.rag-routing` - Meta-guidance telling the assistant which namespace to consult for each user intent category. (This is NOT shown to end users directly.)

All five namespaces MUST be ingested. Do **not** cross-mix their content during ingestion; keep boundaries clean.

9.2 Chunking Rules

We break this doc into small, semantically coherent chunks (200–400 words per chunk). Each chunk will become a retrievable unit.

Examples of good chunks: - Chunk A: "What is the Notifications page and what types of alerts will I see?" (namespace: `notifications.product`) - Chunk B: "GET /api/setting/notificationslist contract" (namespace: `notifications.api`, endpoint: `notificationslist`) - Chunk C: "How marking read works, optimistic UI, bulk mark" (namespace: `notifications.product`, `notifications.api`) - Chunk D: "Security & PII rules" (namespace: `notifications.security-privacy`) - Chunk E: "Infinite scroll + hasMore + IntersectionObserver" (namespace: `notifications.ui-components`)

We DO NOT create one 5,000-word chunk.

We DO create many focused chunks labeled with correct metadata (below).

9.3 Required Metadata Per Chunk (must accompany every chunk in the vector store)

```
{
  "namespace":
"notifications.api",          // one of the namespaces defined above
  "component": "NotificationList",          // UI component OR logical
subsystem
  "feature_area": "mark-as-read",           // short slug of the feature
covered by this chunk
  "endpoint": "/api/setting/
notificationslist", // API path if relevant, else "N/A"
  "requires_auth": true,                     // boolean, does the described
feature need Bearer token / login
  "pii_sensitivity": "high",                // "none" | "low" | "high" based
on exposure risk
  "version": "v1",                          // increment when behavior
changes
  "page_url": "/notifications/",           // main UX surface this applies to
  "component_type": "page"                  // "page" | "api" | "ui" |
"security" | "flow" | "faq"
}
```

Rules: - `requires_auth` should be `true` for anything describing private per-user data, even if endpoint is GET. - `pii_sensitivity` is `high` if actor names or connection info is discussed. - `page_url` MUST include `/notifications/` for anything about this page. - `version` must bump if we change pagination size, event categories, read rules, etc.

9.4 Assistant Retrieval Policy (how assistant should use these chunks at runtime)

- If the user asks "Why did I get this alert that <Name> posted a job?":
- Retrieve from `notifications.product` and `notifications.ui-components` chunks that describe job-post notifications and row formatting.
- Include explanation that it's triggered because a contact posted a job.

- DO NOT expose internal IDs or any data not already visible to that user.

- If the user asks "How do I mark all as read?":

- Retrieve from `notifications.api` (bulk mark read) + `notifications.product` (UX behavior), mention `/api/setting/notificationReadUpdateAll` *conceptually*, but don't expose raw endpoint unless user is clearly a developer/support.

- If the user asks "Can other users see my notifications?":

- Retrieve from `notifications.security-privacy` and answer with the rule: notifications are private and gated by auth.

- If the user asks dev questions like "what params does /api/setting/notificationslist take?":

- It's safe to answer using `notifications.api` chunks, because that's internal developer context.

9.5 Redaction Rules for Assistant

When generating an answer for an end user (not engineer / admin / support role): - The assistant MUST NOT: - Leak internal IDs (`notificationId`, `actorUserId`), chatId, or Firestore paths - Promise retention windows or data deletion timelines unless explicitly documented in `notifications.security-privacy` - Infer private details about the actor beyond what the UI already shows (ex: "Tarun Grover is a hiring manager at …" unless that was clearly visible in the notification row or job post itself)

When answering for an internal dev / support persona (explicitly asked, e.g. "what's the endpoint?" or "what are the params?"): - It's OK to share endpoint shapes, payload, example JSON, etc.

Assistant must always follow these restrictions even if user asks directly.

---

10. FAQ Chunks (RAG-ready examples)

Below are canonical Q→A snippets. Each of these should be ingested as standalone chunks (component_type: "faq") so RAG can quickly answer "what does this mean" questions.

10.1 "What is the Notifications page and what can I do there?"

The Notifications page (`/notifications/`) is where you see everything that needs your attention across ConnecWrk — for example: connection requests, jobs posted by your contacts, people reacting to your posts, or upcoming events. Each item tells you what happened, who triggered it, when it happened (like `2mo`), and gives you a direct link (like `Click here`) to view or respond. You can open a notification to see details (for example, the job post or the sender's profile), and that will automatically mark it as read.

Metadata:

```
{
  "namespace": "notifications.product",
  "component": "NotificationsPage",
  "feature_area": "overview",
  "endpoint": "N/A",
  "requires_auth": true,
  "pii_sensitivity": "high",
  "version": "v1",
  "page_url": "/notifications/",
  "component_type": "faq"
}
```

10.2 "Why did I get a notification that my contact posted a job?"

You got that notification because someone you're connected with created a new job or assignment. We surface these so you don't miss paid opportunities from your own network. When you click the link in that notification, you'll go straight to the job details so you can review it or apply. Only you can see your own notification feed; it's not public.

Metadata:

```
{
  "namespace": "notifications.product",
  "component": "JobPostNotification",
  "feature_area": "job-opportunity",
  "endpoint": "/api/setting/notificationslist",
  "requires_auth": true,
  "pii_sensitivity": "high",
  "version": "v1",
  "page_url": "/notifications/",
  "component_type": "faq"
}
```

10.3 "How do I mark notifications as read or clear them?"

When you click a notification or follow its link (like `Click here to view job details`), it's marked as read for you automatically. You may also get an option to mark everything as read at once. That bulk action updates all items so they're no longer highlighted as unread.

Metadata:

```json
{
  "namespace": "notifications.product",
  "component": "MarkRead",
  "feature_area": "read-state",
  "endpoint": "/api/setting/notificationReadUpdateAll",
  "requires_auth": true,
  "pii_sensitivity": "high",
  "version": "v1",
  "page_url": "/notifications/",
  "component_type": "faq"
}
```

10.4 Dev / Support: "What API do we call to get notifications?"

Use `GET /api/setting/notificationslist?page={page}&pageSize={pageSize}` with a valid Bearer token. The response includes `data` (array of notifications), plus pagination fields like `currentPage`, `totalPage`, and `hasMore`. Each notification object contains `notificationId`, text describing what happened, a destination link (for navigation), and a `readYN` flag.

Metadata:

```json
{
  "namespace": "notifications.api",
  "component": "NotificationsListAPI",
  "feature_area": "fetch-list",
  "endpoint": "/api/setting/notificationslist",
  "requires_auth": true,
  "pii_sensitivity": "high",
  "version": "v1",
  "page_url": "/notifications/",
  "component_type": "faq"
}
```

10.5 "Can other people see my notifications?"

No. Your notifications are private to you. The system requires your login token to fetch them, and the server checks that each notification actually belongs to you. Other users cannot view your feed.

Metadata:

```json
{
  "namespace": "notifications.security-privacy",
  "component": "Privacy",
  "feature_area": "visibility",
  "endpoint": "/api/setting/notificationslist",
  "requires_auth": true,
  "pii_sensitivity": "high",
  "version": "v1",
  "page_url": "/notifications/",
  "component_type": "faq"
}
```

---

11. Implementation TODO / Future Evolutions (Version Control Notes)

These notes matter because RAG answers MUST reflect versioned behavior.

- **Unread badge / counter:** We may expose a global unread count in the top nav bell icon near `Home | Contacts | Messages | Notifications`. If this ships, bump `version` to `v2` for `notifications.product` and `notifications.api` chunks and document the new endpoint (likely something like `/api/setting/notificationUnreadCount`).

- **Actionable inline buttons:** Instead of "Click here", we may embed inline buttons: `View Job`, `Accept`, `Decline`. That adds POST routes for accept/decline directly from Notifications. When released, create new chunks in `notifications.ui-components` with `version: v2` and DO NOT overwrite old chunks — we preserve backward-compatible history.

- **Real-time push:** We may move from poll/paginate to WebSocket or push for live notifications. That introduces a new transport (WebSocket channel or Firebase listener). When shipped, add `notifications.realtime` namespace, mark `requires_auth=true`, and document security model.

Until those changes are live, this document (v1) is the source of truth for `/notifications/`.

TL;DR for RAG Consumers

- `/notifications/` = personal inbox of activity
- Core API: `GET /api/setting/notificationslist`
- Mark read: `GET /api/setting/notificationReadUpdate` (single) or `POST /api/setting/notificationReadUpdateAll` (bulk)
- You only see YOUR notifications; auth required
- Each notification shows what happened, who triggered it, when, and where to go next
- RAG ingestion MUST split this doc into 200–400 word chunks with the metadata schema above, under namespaces:
- `notifications.product`
- `notifications.api`
- `notifications.ui-components`
- `notifications.security-privacy`
- `notifications.rag-routing`
- Assistant answers MUST respect PII redaction + auth visibility rules.

This is the canonical Notifications module spec for ConnecWrk v1.