ChatGPT

Notification System Design for Birdwatching Tours

This spec outlines triggers, channels, frequency, personalization, user preferences, and architecture for notifying users when tours match their bird-list interests. Recommendations are split into **MVP vs future** features.

Notification Triggers

- **New Tour Created (MVP):** When an operator lists a new tour with target species, immediately notify all users whose *chase list* contains *any* of those species (OR logic). This real-time trigger ensures users hear about relevant tours as soon as they're announced. Use an event-driven model: the tour creation event is published and a notification service subscribes and reacts [1]. (E.g. "New tour to see [Species]!")
- **Tour Threshold or Confirmation (MVP):** Since tours require a minimum group to run, trigger a notification when a tour reaches its threshold or is confirmed to go. This provides *social proof* ("Your tour is happening – **12 birders already booked**") and encourages bookings. For example, when the 6th booker signs up on a 6-person tour, email users who *had chase matches* on that tour, as it indicates community interest.
- **Urgency (MVP):** If a tour is nearly full, or close to departure, send an alert to matched users to convey scarcity ("Only 3 spots left – act fast!"). Marketing studies show using scarcity language ("only X left", "act now") in messages boosts conversion [2] [3]. Add such urgency cues if seats are few.
- **Changes & Deals (Future):** Future phases may notify on *price drops* or *date changes* for tours that match a user's interests, as these are material changes. For now, focus on core events (new tours, confirmations).
- **Location Consideration:** Do not filter out tours by distance (birders often travel far). Use location as context only (e.g. "4 hr drive away"), not as an eligibility filter.

Each notification should be **relevant and timely**: send it shortly after the trigger event so the user still remembers their interest [4] [5]. Group similar alerts when possible (e.g. "3 tours match Species X this week" instead of three separate emails) to avoid overload [6] [7].

Delivery Channels

- **Email (MVP):** The primary channel. Use HTML emails (with rich formatting, images of birds, etc.) *and* include a plaintext alternative (multipart MIME) for compatibility [8]. Ensure each email has a clear subject line and "unsubscribe" link [9]. For this older demographic, clear, meaningful subject lines are key (personalized lines can lift open rates by ~26% [10]). Example subject: *"Jane, new tours for your Scarlet Honeyeater list!"*. Include key details (species, dates, urgency) in the body and use clean design.
- **In-App/Web Notification (MVP):** For users logged into the site, show alerts in a notification center. Implement via WebSockets or server-sent events (Supabase Realtime) so new notifications appear in real-time [11]. The in-app UI can mirror email content (title, snippet, link to tour).

- **SMS (Phase 2):** Reserve for high-priority alerts or opt-in users with mobile numbers. SMS must be short (~160 chars) and typically is more urgent. Integrate via a service like Twilio [12]. Be mindful of costs and international delivery (Australia has regional rules). SMS content should be extremely concise (e.g. "Tour to see [Bird] with 2 spots left – details: [link]"). Always require explicit opt-in before SMS.
- **Push Notifications (Future):** If/when a mobile app is built, integrate with APNs/FCM. Push is great for instant re-engagement but requires careful opt-in and permission handling [13]. Initially focus on web/email only.

Channel Priorities: Let users set preferred channels. By default, send an email and in-app alert for each notification. If a user later opts into SMS or (future) push, either send concurrently or use fallback logic: e.g. urgent alerts go via SMS if enabled, otherwise email. Provide user controls for "never send SMS" etc [14] [15].

Frequency & Batching

- **User Controls:** Offer settings (MVP) to let users choose notification frequency: **Instant**, **Daily digest**, **Weekly digest**, or **Off** [16] [15]. Many birders may prefer a daily summary instead of a dozen emails. As a rule of thumb, try not to send more than ~5 notification emails per day to any user [17] (beyond that, switch to digests).
- **Batching/Grouping:** When multiple events match a user, batch them into one message if close in time. For example, if 3 new tours match in one day, combine them into a single "3 new tours" email. [6] [7] This reduces alert fatigue and makes content easier to scan. Also consider grouping by category (e.g. species or region). However, still highlight the count ("2 of your species match"). Critical/urgent messages should bypass batching, while routine ones can wait for the next digest [18].
- **Time-of-Day:** Never send notifications at night. Use the user's home timezone (from their city/state) to infer local time. Enforce quiet hours (e.g. 9pm–7am local) during which only truly urgent messages (e.g. confirmed tour) might be sent; otherwise delay emails until morning [19] [20]. For example, if an event happens at 2am user time, queue the email for 7:00am. Respecting quiet hours is considered best practice to avoid user annoyance [20].
- **Limits and Defaults:** Start users with smart defaults (e.g. immediate alerts for new tours, daily summary for others). Let them dial frequency up or down globally or per-species. Internally track how many emails a user gets; if it exceeds a soft threshold, automatically switch to a summary (as recommended on UX forums [17]).

Personalization

- **Content Personalization:** Use the user's name and interests to make messages feel personal. Studies show adding personal details (e.g. name, referenced favorites) increases engagement [10]. For example, the email body could open with "Hi Jane, good news for your Birders List!" or tailor recommendations ("Your Scarlet Honeyeater has been spotted!").
- **Highlight Matches Count:** If a tour covers multiple species on the user's list, mention the number ("This tour covers **2 of your chase species**."). Likewise, if multiple tours match, mention "3 tours match your list today." This dynamic content signals relevance without needing separate emails for each.

- **Social Proof:** Including data like “12 birders already booked” or “9 people are interested” can motivate action via social proof. (Research in marketing shows FOMO cues like this improve conversions.) Frame it positively, e.g. “Join 12 fellow birders on this tour.”
- **Urgency Language:** Use urgent wording in subject lines and CTAs. Campaign Monitor notes that words like “Hurry,” “Act now,” or “Limited seats” create action bias [2] [3]. For instance, a subject line could be “Hurry! Only 3 spots left for 🐦 Scarlet Honeyeater tour.” The email CTA (“Book Now”) should similarly prompt immediate action.
- **Segmented Messaging:** Over time, you can further personalize by region or previous behavior (e.g. “Previously you viewed tours in Tasmania – here’s a new one”). For MVP, focus on species matches; future enhancements might use user location or past bookings to tailor content.

User Preferences & Controls

- **Subscription Management:** In the user profile or settings, allow toggles such as: which channels to receive (email, SMS, etc.), which species or regions to subscribe/unsubscribe, and desired frequency (instant/summary). For example, a user could unsubscribe from “Waterfowl tours” or mute notifications for a species for a time. A straightforward “unsubscribe” link in every email (as required by law) should let them opt out easily [9] [21].
- **Smart Defaults:** When a user signs up, default them to receive **instant notifications** for all species on their list, on all channels (email/in-app). Offer daily digests as a user-selectable option. By default, apply quiet hours (no late-night emails) as mentioned above. These defaults ensure out-of-the-box coverage without configuration, reducing churn.
- **Quiet Hours Setting:** Even though we auto-honor 9pm–7am, consider letting users fine-tune quiet hours (e.g. 10pm–6am) if needed [20]. This adds flexibility.
- **Mute vs. Unsubscribe:** Differentiate “mute for a time” versus permanent unsubscribe. For example, allow “snooze notifications for 1 week” on a species or region. Also support full notification opt-out (with the option to re-subscribe). Always provide a clear “Stop” link in emails [9] [21] to comply with opt-out norms.
- **Granularity:** While per-species toggles are ideal, start with broad controls (global frequency, channel on/off). More granular controls (per-region or per-species) can be added after initial launch.
- **Preference Defaults (MVP):** All users start opted in for new tour alerts by default (since most serious birders want them). Allow them to change this setting at any time.

Technical Architecture

- **Event Pipeline:** Use an event-driven architecture. When a tour is created or updated, emit an event to a message queue (e.g. Redis Stream, RabbitMQ, or Kafka). A Notification Worker service consumes these events. This decouples operators from notifications and allows retry/scale [1] [22]. For example:
- **Producer/API:** A new-tour API (`POST /api/tours`) inserts the tour in the DB and publishes a “TourCreated” event.
- **Message Broker:** Events are enqueued in Redis/Kafka.

- **Workers:** Notification workers pull events, apply business logic (match users, format messages), and enqueue outbound tasks (emails, SMS).

- **Matching Logic:** To find matching users, join the new tour's target species with the users' chase-list table. In SQL: `SELECT user_id FROM user_chase_list WHERE species_id IN (...)`. For bulk efficiency, use an **INSERT … SELECT** into a notification table [23]. For example:

```
INSERT INTO notification_recipient (notification_id, user_id)
  SELECT :notif_id, user_id
  FROM user_chase_list
  WHERE species_id = ANY(:tour_species_list);
```

  This single query avoids fetching all users to the app layer [23]. Index the `user_chase_list(species_id)` column for speed.

- **Database Schema:** (PostgreSQL via Supabase) Key tables:

- **users** (id, email, home_location, timezone, preferences_id, etc.)
- **species** (id, name, other taxonomy data).
- **user_chase_list** (user_id, species_id) – a join table listing user interests.
- **tours** (id, operator_id, date, price, location_id, threshold, seats_available, etc.)
- **tour_target_species** (tour_id, species_id) – since tours can have 1–5 target species.
- **notifications** (id, tour_id, triggered_at, content, type, etc.) – a core notification event (for all recipients).
- **notification_recipient** (notification_id, user_id, channel, sent_at, read_at) – one row per user-targeted notification (for inbox and tracking). This follows the model of many systems [24].
- **user_preferences** (id, user_id, email_enabled, sms_enabled, push_enabled, frequency, quiet_start, quiet_end, etc.) – storing user settings. Example schema from industry includes flags for enabled channels and quiet hours [25].
- **locations/regions** (for tours and user regions, if needed later).

Ensure referential integrity (foreign keys) and indexes (e.g. on `notification_recipient(user_id)`, `user_chase_list(species_id)`).

- **Queues & Workers:** Use at least two queues: a **Priority/Real-Time queue** (for urgent alerts) and a **Bulk/Batch queue** (for digests and lower-priority sends) [26]. For example, tours nearing full or confirmed might go into the priority queue so emails/SMS go out immediately. Non-urgent digests can be batched by region or time bucket. The workers should respect rate-limits (e.g. email per-second) and automatically back off on failures. Use a background job or cron (Supabase Edge function or Vercel Cron) to trigger scheduled digests (e.g. compile daily summaries each evening).

- **Delivery Services:** Integrate with external services: SMTP (e.g. Resend) or SendGrid for email [9]; Twilio for SMS [12]. For in-app, rely on Supabase's real-time or polling APIs. Each send should be logged (status, timestamp, provider message ID) in `notification_recipient` for auditing.

- **API Endpoints (example):**

- `POST /api/tours` – Operator creates a tour (with target species, date, price, etc.). Triggers the matching process.

- `POST /api/notifications` – Internal endpoint to enqueue a notification event (could be used by other systems).
- `GET /api/users/{userId}/notifications` – Fetch a user's notifications (in-app inbox). Support pagination and mark-as-read.
- `GET|PUT /api/users/{userId}/preferences` – Retrieve/update channel/frequency settings.
- `GET|POST /api/users/{userId}/chase-list` – List or update the user's species list. (Alternatively, manage chase list via the UI; API calls to support it.)
- `POST /api/notifications/send` – (Worker endpoint) Accepts a payload and delivers it to the chosen channel(s) (email/SMS).

These follow REST conventions and can be built as Next.js API routes. Security and auth (via Supabase Auth) are assumed for user-specific endpoints.

Queue Design & Scaling

- **Asynchronous Processing:** Always write to a queue rather than sending directly in the user request flow [27]. This decouples spikes from downstream systems. For instance, on tour creation, immediately return success to the operator and do all notification work in background.
- **Message Broker:** A hosted queue (Redis Streams, RabbitMQ, or AWS SQS) can handle bursts. For scalability, a log-based system like Kafka is also an option, but for MVP a simpler queue suffices. The queue should support retry and dead-letter for failed sends (e.g. if Twilio is down).
- **Prioritization:** Tag notifications by priority. E.g. "urgent" tours (threshold met, low seats) go to a high-priority queue consumed by a fast worker pool. Daily digests can go through a separate batch worker. This matches the common **priority/standard/bulk queue** pattern [26].
- **Backoff and Deduplication:** If a user has already been notified of a tour, avoid resending the same alert. For example, if a tour's date changes and you re-trigger, check a `notification_recipient` log to avoid duplicate emails. Use unique constraints or idempotency keys to prevent double-sends.
- **Monitoring & Alerting:** Instrument metrics: queue length, processing lag, and error rates. Alert if backlog grows beyond a minute or if delivery failure rates spike [28]. This ensures we scale consumers up or investigate issues before users go without notifications.

Metrics & Optimization

- **Delivery Metrics:** Track basic KPIs for each channel: email delivery rate (bounces), open rate, click-through rate (CTR, e.g. link to book the tour), and conversion rate (users who actually book after clicking). High open/CTR indicates relevance; low rates suggest message/content tweak. For email, include tracking pixels/links to measure opens and clicks. For in-app alerts, log which are viewed or clicked.
- **Conversion Tracking:** Tie notifications to bookings by using unique links or tags. E.g. include a query param or campaign ID in the tour link. Then measure what percentage of notified users end up booking (or at least viewing) the tour. This tells you which notifications drive revenue versus those that are just noise.
- **Engagement Monitoring:** Watch for signs of fatigue: unsubscribe rates or user disabling alerts. If many users unsubscribe from a species or channel, that's a signal to adjust (maybe frequency was too high) [29] [13]. Also monitor opt-in rates for new channels (e.g. SMS opt-ins).
- **A/B Testing:** Continuously test variations in subject lines, email copy, send times, etc. For instance, test including the user's name or number of matching species in the subject (personalized lines can

boost opens ~26% [10]). Test sending at 9am vs 3pm local time to see which gets more engagement. The large demographic (45–65) tends to open emails (~25% open rate [30]), so iterative testing can further improve those numbers.

- **Content Analytics:** Evaluate which content elements perform best. For example, try adding an emoji or an image and see if CTR improves. Or experiment with including social-proof snippets vs. not. Use the data to refine the template (keeping in mind older users prefer clarity over clutter).
- **Batch vs Real-Time Review:** Compare the performance of instant vs. daily-digest notifications. For each, track open/conversion and iterate. If instant emails to a user are rarely opened, consider moving that user to digest mode. As a guideline, campaigns sending **more than ~5 updates/day** should switch to digests [17].
- **Reporting Dashboard:** Build internal dashboards (e.g. in Supabase) to display these metrics and trends over time. Automate alerts if a campaign's CTR drops or unsubscribe rate spikes, so you can respond quickly.

Sources: We draw on industry best practices and studies of notification systems and email marketing [1] [29] [10] [30] (see citations above). These inform both user-facing design (relevance, timing, personalization) and backend architecture (event-driven flow, queues, database schema). All recommendations are prioritized by **(MVP)** or **(Future)** based on implementation complexity.

---

[1] [22] [23] [24] Building a Scalable Notification System: From Database Design to Push Notifications | by aceiny | Medium
https://medium.com/@a_zeraibi/notification-system-on-scale-2ac248df8b83

[2] [3] How to Create a Sense of Urgency for Conversions in Email Right Now | Campaign Monitor
https://www.campaignmonitor.com/blog/email-marketing/how-to-create-sense-urgency-in-emails/

[4] [5] Trigger Emails: Best Practices and Examples | Lead Genera
https://leadgenera.com/knowledge-hub/email-marketing/trigger-emails-best-practices-and-examples/

[6] interaction design - Are there guidelines on the timing and frequency of notifications? - User Experience Stack Exchange
https://ux.stackexchange.com/questions/123963/are-there-guidelines-on-the-timing-and-frequency-of-notifications

[7] [15] [18] [20] [21] [29] Batching & Digest - SuprSend, Notification infrastructure for Product teams
https://docs.suprsend.com/docs/best-practices-for-batching-digest

[8] Plain Text vs. HTML Emails: Why Marketers Should Send Both Formats
https://glockapps.com/blog/plain-text-vs-html-emails-why-marketers-should-send-both-formats/

[9] [11] [12] [19] [25] [26] [27] Notification System Design: Architecture & Best Practices
https://www.magicbell.com/blog/notification-system-design

[10] An Expert Guide to Email Personalization [2024]
https://www.ama.org/marketing-news/email-personalization-strategies/

[13] A Guide To Push Notification Best Practices | Braze
https://www.braze.com/resources/articles/push-notifications-best-practices

[14] [28] Design a Notification System: A Complete System Design Guide | by Madhur Banger | Dec, 2025 | Medium
https://medium.com/@bangermadhur/design-a-notification-system-a-complete-system-design-guide-3b20d49298de

16 17 What's the best approach: real time email notifications or digest emails at regular intervals? - User Experience Stack Exchange
https://ux.stackexchange.com/questions/62246/whats-the-best-approach-real-time-email-notifications-or-digest-emails-at-regu

30 Email Open Rates by Age Group: What Marketers Should Know
https://neilpatel.com/marketing-stats/email-open-rates-by-age-group/