ChatGPT

Arcana Phase 12.3: Smart Agents & Follow-Up

We introduce a proactive "Agents" system so Arcana can remind users about tasks, approvals or delays without being pushy. Arcana will send a **daily digest email** summarizing each user's pending tasks and unread proofs, display **in-app nudges** (toast banners) for urgent items, and apply **escalation rules** on overdue work. All rules are user-definable "if-this-then-that" triggers. For example, *"if a shared proof is unviewed after 3 days, then nudge the owner."* Well-designed nudges are *"brief and minimal... gentle interruptions"* that feel natural, not disruptive [1] [2].

- **Daily Digest Emails:** Every morning, send each team member an email listing their due/overdue tasks, awaiting approvals, or unviewed proofs. (HubSpot's CRM similarly offers a "task daily digest" summarizing due tasks [3].)
- **In-App Nudges/Alerts:** Show subtle banners or toast pop-ups in the app when attention is needed. For example, a toast might say: "Your proof *Design_X* is still pending review – Resend link?" with a quick "Resend" button. Since toast notifications are auto-dismissing status messages [4], they provide instant feedback without forcing action.
- **Escalation Rules:** If a task or approval slips past its due date by *X* days (or no activity occurs), the system automatically escalates. This could mean sending a stronger reminder or alerting the manager. In practice, automated escalation ensures *"no task is left unfinished"* by re-notifying or reassigning overdue items [5] [6].

Each feature is configurable via an **Agent Rules** UI. Users (or admins) can add rules with triggers (e.g. `proof_unviewed`, `task_overdue`), delays, actions (`nudge`, `escalate`), and tone (agent persona). The backend logs each trigger in `agent_logs` so we don't repeat alerts. Below we detail the schema, examples, delivery logic, API, and UI.

Database Schema (Backend)

We need two new tables: **agent_rules** (storing each nudge/escalation rule) and **agent_logs** (recording each sent nudge). For example:

```sql
CREATE TABLE agent_rules (
  id SERIAL PRIMARY KEY,
  created_by INTEGER REFERENCES users(id),
  trigger_type VARCHAR(50) NOT NULL,
-- e.g. 'proof_unviewed', 'task_overdue', etc.
  delay_days INTEGER NOT NULL,          -- days to wait before triggering
  action_type VARCHAR(20) NOT NULL,     -- e.g. 'nudge', 'escalate'
  channel VARCHAR(10) NOT NULL,         -- 'email' or 'in_app'
  tone VARCHAR(50),                     -- e.g. 'formal', 'energetic', etc.
  active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE agent_logs (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id),  -- user being nudged/notified
  rule_id INTEGER REFERENCES agent_rules(id),
  related_id INTEGER,                    -- optional FK to related item (task/
proof) for de-duplication
  triggered_at TIMESTAMP,
  delivery_channel VARCHAR(10),         -- 'email' or 'in_app'
  status VARCHAR(20),                   -- e.g. 'sent', 'viewed', 'dismissed'
  created_at TIMESTAMP DEFAULT NOW()
);
```

- **agent_rules**: each row is one "if-this-then-that" rule. The `trigger_type` is the event (proof shared, task due, etc.). `delay_days` is how long to wait. `action_type` is typically "nudge" (gentle reminder) or "escalate" (stronger reminder). `channel` selects email vs in-app. `tone` selects an agent persona (for the message voice). This follows best practice for automated workflows: store clear criteria so we can query them later [5].
- **agent_logs**: whenever a rule fires, we record a log. This lets us show history (see UI below) and prevents duplicate alerts (we can index `(rule_id, related_id)` to ensure we only notify once per item). The `status` tracks if the user saw or dismissed the nudge.

This schema supports flexible automation (adding new triggers or channels later) while logging every reminder for audit.

Example Agent Rules

Here are some sample rules a creative event team might set up:

- **Rule 1:** *If a client proof is unviewed for $\geq$ 3 days, then send an in-app nudge.*
- *Trigger*: `proof_unviewed`, `delay_days = 3`
- *Action*: Nudge (in-app), *Tone*: Friendly/Encouraging.

- *E.g.* " Your proof *DesignConcept-42* is still pending client review. Want me to resend it?"

- **Rule 2:** *If a task is overdue by 1 day with no activity, then escalate via email.*

- *Trigger*: `task_overdue`, `delay_days = 1`
- *Action*: Escalate (email manager), *Tone*: Formal.

- *E.g.* "Reminder: Task *Venue_booking* was due yesterday and hasn't been updated. Please review and reschedule."

- **Rule 3:** *If no feedback on a draft event plan after 5 days, send daily digest email.*

- (This could be implemented as a custom rule or part of the nightly digest job.)

- **Rule 4:** *If a project has 0 recent updates for 7 days, nudge the project owner.*

- *Trigger*: `project_idle`, `delay_days = 7`
- *Action*: Nudge (in-app), *Tone*: The Producer (energetic).
- *E.g.* "⚡ It's been a week on Project *SummerGala*. Ready to take the next step?"

These examples illustrate the "if [event]+days, then [reminder]" pattern. Teams can customize triggers (e.g. new trigger types like `approval_pending`) and actions (e.g. adding SMS or Slack as channels later).

Delivery Logic (Scheduling & Escalation)

A background worker (cron job or queue worker) runs daily (e.g. at 6:00 AM) to evaluate all **active** agent rules and send alerts. The flow is:

1. **Fetch Active Rules:** Query `agent_rules` where `active = true`.
2. **Check Conditions:** For each rule, find items matching the trigger:
3. *Proof unviewed:* select proofs with `shared_date <= NOW() - INTERVAL '3 days'` and no `viewed_at`.
4. *Task overdue:* select tasks where `due_date <= NOW() - INTERVAL '1 day'` and `status != completed`.
5. Etc.
6. **Prevent Duplicates:** For each matching item, check `agent_logs` to see if we already alerted on this (use `rule_id` + item ID). If not, proceed.
7. **Create Log & Send Alert:** Insert a row in `agent_logs` (with status=`sent`). Then deliver the reminder:
8. If **email**, queue an email with a template (pulling data from the rule's tone, task details, etc.).
9. If **in_app**, push a notification via WebSocket or record it so it appears in the user's next page load.
10. **Handle Escalation:** Some rules may have `action_type = escalate`. In that case, notify the manager or assign higher priority.

This automated workflow fits known practices: by "setting up rules that send reminders or escalate tasks" after a set time, we ensure "no task is left unfinished" [5]. Overdue items automatically bubble up, reducing manual chasing.

API Endpoints

To support the frontend, we expose RESTful endpoints for managing rules and logs. For example:

- `GET  /api/agent_rules` – List all agent rules (for the settings page).
- `POST /api/agent_rules` – Create a new rule (body: trigger_type, delay_days, action_type, channel, tone, active).
- `GET  /api/agent_rules/{id}` – Get details of a rule.
- `PATCH/PUT /api/agent_rules/{id}` – Update a rule (e.g. change delay or deactivate).

- `DELETE /api/agent_rules/{id}` – Remove a rule.

- `GET  /api/agent_logs` – List recent alerts sent (optionally filter by user or date).

- `GET  /api/agent_logs/{id}` – Get a log entry (for details on a specific nudge).
- `GET  /api/notifications` – (For in-app view) fetch current unread in-app notifications for the user.
- `POST /api/notifications/{id}/dismiss` – Mark an in-app notification as seen/dismissed.

Each endpoint returns JSON. (For example, `GET /api/agent_rules` returns an array of rules like `{id, trigger_type, delay_days, action_type, channel, tone, active}`.) These APIs let the frontend list, create, and update rules, and display the agent log/history. (In practice you'd also add auth checks and validation.)

Frontend UI Design

In-App Notifications (Nudges & Banners)

In the application's UI, we add a notification component for agent messages. We will use **toast notifications** or banners that appear unobtrusively. In UX design, toasts are "auto-dismissing status messages… used to convey information without disrupting user workflows" [4]. A banner is a persistent message at the top (useful for critical alerts), whereas a toast appears briefly and then fades [7]. Arcana's friendly reminders should mainly use toasts to avoid blocking the user.

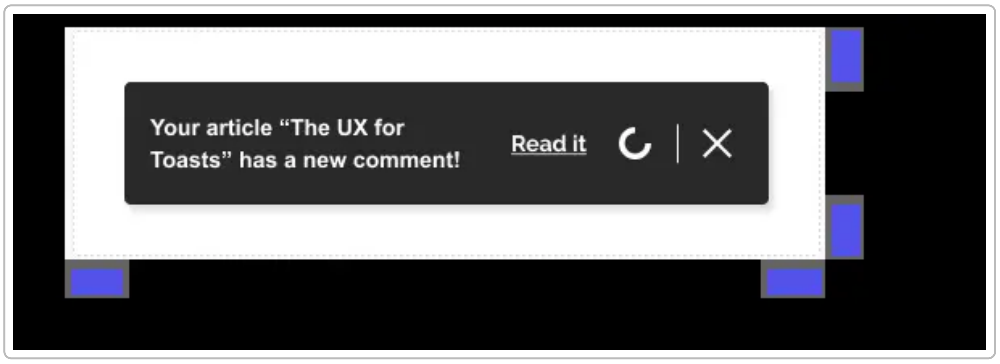

Figure: Example in-app toast notification (nudge). A typical toast might say "Your proof *Design123* is pending approval – Resend?" with a quick action button. As design guidelines note, toasts must remain "brief and minimal… gentle interruptions" [1]. They slide in, display a one-line message and optional buttons, then auto-dismiss after a few seconds (or when the user closes them). This keeps the flow seamless.

Colors and icons can convey urgency. For example, an **error/urgent** toast could be red with a "Retry" or "Resolve" button, while a **success/info** toast is green. Design rules suggest that "negative" (error) toasts get higher priority (not auto-dismiss) and may include explicit actions, whereas "positive" toasts can disappear automatically [8].

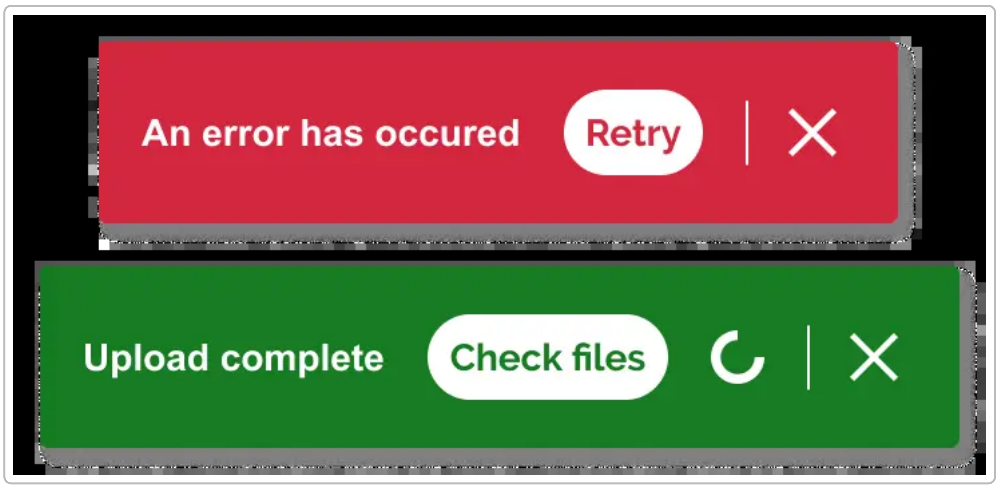

Figure: Toast variants (error vs success notifications). In this example, the red toast "An error has occurred – Retry" is modal in color, while the green "Upload complete – Check files" indicates success. Using these visual variants helps users instantly grasp the message's context. In our system, each rule's `tone` setting will select appropriate wording (formal, casual, etc.) and color.

Agent Settings Page

We add a **"Smart Agents"** or **"Follow-Up Rules"** page in the Arcana UI. This page lists all configured agent rules in a table. Key elements:

- A table/grid of rules: columns for *Trigger Type*, *Delay*, *Action*, *Channel*, *Tone*, and an *Active* toggle. Each row shows the rule details and whether it's enabled.
- **Create/Edit Rule**: A form (modal or panel) lets the user set up a new rule. Fields include:
- **Trigger Type:** dropdown of options (e.g. "Proof Unviewed", "Task Overdue", "Approval Pending", etc.).
- **Delay (days):** numeric input.
- **Action Type:** radio or dropdown (`Nudge` vs `Escalate`).
- **Channel:** checkbox or dropdown (`Email`, `In-App`).
- **Agent Tone:** dropdown of persona names (e.g. Formal, Energetic, Playful).
- **Active:** on/off toggle.
- **Buttons:** Save or Cancel. (Existing rules can be edited or deleted, and active/inactive toggled.)

Common UI patterns (inspired by email filter or workflow builders) will make this intuitive. For example, using dropdowns and date pickers as in Apple Mail rules or HubSpot workflows. A preview or help text can show what the final message might look like with the chosen tone.

Agent Log / History View

We also provide a **Log** or **History** view so users (and managers) can see past nudges and escalations. This might be a tab or page showing a list of `agent_logs` entries:

- Columns: Date, User (who was notified), Rule Name/ID, Delivery Channel, Status (Sent / Viewed / Dismissed), and optional Link to the related item (task, proof, etc.).
- Filters: by date range, user, or rule.

- "Seen" indicator: e.g. a different row color if the user clicked/dismissed the notification.

This transparency helps track that reminders were sent. For example, if a task stalled, you can see that Agent *Producer* sent two nudges and one alert, so you know the follow-ups happened.

Agent Personas and Prebuilt Agents (Optional)

For a friendly touch, we can define **Agent Personas** – each with a name, style, and example phrasing. Users can assign a persona (via the `tone` field) to each rule so messages stay on-brand. For example:

- **The Archivist** (Formal, Precise) – uses concise, professional language. *"Dear Team, Task* Venue Setup *is now overdue. Please address this promptly."* Useful for administrative reminders or approvals.
- **The Producer** (Energetic, Encouraging) – upbeat and lively. *" Heads up!* Catering Order *is almost due. Let's wrap it up!"* Good for creative deadlines and positive nudges.
- **The Maestro** (Warm, Organized) – supportive and friendly. *"Good morning! You have 3 items waiting today. Let me know if you need any help."* Suitable for daily summaries or gentle reminders.

We could ship a few prebuilt agent profiles, each with suggested default rules. For instance, a "New Project Onboarding" agent that welcomes new tasks, or a "Design Lead" agent that reminds reviewers. The user can then tweak these rules or create their own, much like choosing a persona for a chatbot. (In chatbot design literature, matching the agent's tone to its role builds trust and engagement [9].)

By configuring these personas and rules, Arcana becomes a proactive assistant: it *remembers* to nudge on your behalf. The combination of daily emails, in-app toasts, and escalations will keep creative teams on track without feeling nagged, aligning with UX best practices for subtle "in-app nudges" [10] [1].

---

[1] [8] UX Files - The UI of notification toasts - Benoit Rajalu - Frontend Engineer and design systems specialist
https://benrajalu.net/articles/ui-of-notification-toasts

[2] [10] The Complete Guide To In-App Nudges 2025: Building Seamless User Experiences
https://nudgenow.com/blogs/in-app-nudges

[3] Set up task reminders and daily task digest emails
https://knowledge.hubspot.com/tasks/task-reminders-and-daily-digest

[4] [7] What is a toast notification? Best practices for UX - LogRocket Blog
https://blog.logrocket.com/ux-design/toast-notifications/

[5] [6] What is Escalation Rules? Everything You Need to Know
https://www.cloudsylla.com/post/what-is-escalation-rules

[9] Chatbots: Persona (Part 3) — Tone of Voice | by HuggyMonkey | Medium
https://medium.com/@HuggyMonkey/chatbots-persona-part-3-tone-of-voice-11d15166e1c0