# Eden Task Module – Focused Development Plan

## Purpose and Scope of the Eden Task Module

The **Eden Task module** (Coordination System) is intended as the central hub for all internal task management and communication. Its purpose is to replace fragmented tools (WhatsApp, email, MS Planner) with **one transparent, trackable workspace** covering interdepartmental tasks, requests, and follow-ups[[1]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). In scope is the full task coordination feature set – from task creation through completion – delivered as a **desktop-first web app**. This module will function as a standalone coordination center for now, laying the foundation for a future integrated ERP. We will **temporarily set aside the broader ERP shell** and any mobile-specific interface. All focus is on making the Task module near-production quality for internal testing, especially by family members acting as pilot users.

**Scope:** The Eden Task module encompasses creating and assigning tasks, tracking their status, facilitating hand-offs between team members, and providing a coordinated view of all ongoing work. It includes the UI, backend, and workflow for task management, along with notifications and basic reporting. We will polish this module in isolation – ensuring it is feature-complete and user-friendly on desktop – before later integrating it into the larger ERP platform.

## Key Features Implemented to Date

The Eden Task module already has a strong functional foundation. Key features developed so far include:

* **Task Creation & Assignment:** Users can create new tasks with details like title, description, priority level, due date, and assign a responsible person. Each task is tied to a project context. Assignment is built-in (tasks store an assignee\_id) so tasks can be owned by a specific team member[[2]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). The creation form includes fields for assignee and other metadata[[3]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L148-L156)[[4]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L160-L168).
* **“Ball-in-Court” Logic (Handoff):** In addition to the static assignee, tasks track who currently has the “ball in court” – i.e. who is **actively responsible for the next action**. Users can hand off the ball to someone else, transferring accountability. This is implemented via a dedicated handoff action: the UI shows the current ball owner and allows choosing a new owner[[5]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L212-L220). When a handoff occurs, the task’s ball\_in\_court field updates and a notification is generated. This ensures clear ownership at every stage and supports workflows where responsibility passes between departments (a core goal of seamless hand-offs)[[6]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L64-L71).
* **Status Tracking & Automation:** Each task carries a status through defined stages (e.g. **To-Do → In Progress → Review → Done**)[[7]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). The system enforces valid status transitions and automates certain updates. For example, if a task has subtasks, the system will automatically mark the parent task “done” when all subtasks are completed, or re-open it if any subtask is reopened[[8]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/services/taskAutoClose.js#L8-L16)[[9]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/services/taskAutoClose.js#L59-L67). This **status change trigger** ensures higher-level tasks reflect the state of their components without manual oversight. Status changes (including completion or reopening) can also trigger notifications and are logged for audit purposes.
* **Notifications System:** The module includes a notifications engine to keep users informed of important events. Users receive **auto-alerts for new tasks assigned to them, status changes, comments added, and ball hand-offs**, among other events[[10]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). A notification bell icon in the UI shows unread notification count in real-time[[11]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L94-L101). Users can view a dropdown list of recent notifications grouped by type (e.g. ball hand-offs, comments, status changes) and mark them as read[[6]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L64-L71)[[12]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L94-L102). The backend queues notifications for each relevant user and even has placeholders for daily summary emails (e.g. an automated summary of tasks “in your court”, due soon, or overdue)[[13]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/notifications.js#L170-L178)[[14]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/notifications.js#L140-L148). While the email/text delivery isn’t fully implemented (currently just console logs), the framework for a robust notification system is in place.
* **Comments & Internal Notes:** Each task has an **internal comment thread** for collaborators to discuss or add notes. Team members can post comments on a task (e.g. to clarify requirements or provide updates). This is reflected in the UI as a comments section showing each comment’s author and timestamp[[15]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L270-L278). The system notifies relevant users when a new comment is added (so that, for instance, the task requester or current assignee knows there’s new input on the task)[[16]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L38-L46)[[17]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L72-L80). These comments serve as an **audit trail** and internal notes log, ensuring decisions and communications about the task are preserved[[18]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0).
* **File Attachments:** Users can attach files and media to tasks, supporting richer collaboration. The backend supports uploading attachments (images, PDFs, voice notes, etc.) and storing metadata like filename and file type[[19]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). In the UI, an **Attachments panel** on each task allows viewing and adding attachments[[20]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L310-L319). Voice notes are also supported: the data model has fields for a voice recording URL and its transcript, aligning with the goal of allowing voice inputs with automatic transcription in the future[[21]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L28-L33). All attachments and uploads are tied to the task record for traceability (and audit logs record these events). This feature transforms the task module into a **media-rich communication tool**, where drawings, photos from site, or voice instructions can live alongside task details.
* **Dashboard and Task Overview:** A central **Dashboard** view gives a high-level overview of task metrics and recent activity. The dashboard displays summary cards (total tasks, tasks overdue, tasks in progress) and charts, such as tasks by status and tasks by assignee[[22]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/DashboardPage.jsx#L38-L46). It also includes a feed of recent activity (e.g. new tasks or comments) to provide a snapshot of what’s happening across the system. This helps managers and users see the **big picture** at a glance – how many tasks are open, who has most on their plate, and if any deadlines are at risk. There is also an “All Tasks” view (and project-specific task lists) with filtering logic available to slice tasks by project, status, owner, etc., ensuring that users can find and organize tasks as needed.
* **Permissions & Roles:** The system has a role-based access control foundation. Roles like **Admin, Manager, User, Guest** are defined, with appropriate read/write permissions on tasks[[23]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). For example, certain users may only see tasks from their department or those assigned to them, whereas managers/admins have broader visibility. This ensures sensitive tasks or projects can be restricted, while general coordination tasks are transparent. While the full role logic is in early stages (and for testing, all family users might be given broad access), the underlying structure (permission checks in the backend routes, and UI adaptation for guest vs authenticated user) is ready. This will allow the Task module to be used by internal staff and potentially external collaborators with controlled access levels.

Overall, the Eden Task module’s functionality is mostly in place – one can create tasks, assign and reassign responsibility, comment, attach files, change status, and get notified of updates. These capabilities make it a **central coordination hub** for the company’s operations, aligning with the original vision of unified task tracking[[24]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0). The focus now shifts to refining the user experience and interface to make these features intuitive and polished for everyday use.

## Reported UI/UX Issues and Pain Points

During initial demos and testing, several UI/UX issues have been identified. These are areas where the current interface or workflow is confusing, visually unpolished, or lacking in functionality. Addressing these is critical to achieve a **Google Workspace-level polish**. Key issues reported include:

* **Visible “Dev Mode” Banner:** In the current build, a developer utility banner is present at the top of the app. It displays a 🔧 **Dev Mode** label with the current user email and allows switching users for testing[[25]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/DevAuthSwitcher.jsx#L40-L48). While useful for development, this bright yellow banner is **distracting and confusing for end users**. Family testers remarked that it made the app feel clearly “in development” and were unsure if they should interact with it. **Resolution:** This DevAuthSwitcher should be disabled or hidden in any user testing deployment. We’ll ensure that when running in a test/production mode, regular users see no dev banners or technical controls. The interface should feel like a polished product, not a development sandbox.
* **“Create Task” Button Not Obvious:** Early feedback noted that users didn’t immediately know how to create a new task. Originally, the UI required navigating into a project to find a “New Task” or had the create action somewhat buried. This has been partially addressed by adding a prominent **“+ Create Task”** button on the Dashboard and project pages[[26]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/ProjectDetail.jsx#L33-L37). However, we must verify that this action is highly visible on all relevant screens. **Resolution:** Keep a consistent primary action button (styled prominently, e.g. a floating action button or top-right button) for creating a task, available from the dashboard and project views. Possibly include a tooltip or callout for first-time users (e.g. “Click here to add a new task”). During testing, we’ll confirm whether users can easily find how to add tasks. If any confusion remains, we may add a global “Create Task” in the header or a persistent “+” icon.
* **Layout and Visual Design Polish:** Testers described the UI as functional but lacking in visual appeal and coherence. Some noted it feels “like a developer interface” rather than a slick modern app. Specific issues include: inconsistent spacing and alignment, minimal styling on elements, and lack of visual hierarchy. For example, the task handoff form currently asks for a raw user UUID input[[27]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L218-L226) – a clear placeholder for a proper user picker UI. This kind of rough edge signals that the UI isn’t final. Additionally, lists and cards are very plain (default white backgrounds with basic text), and the use of system font/unstyled components makes the app look less polished. **Resolution:** Embrace a more refined **Material Design aesthetic** throughout. We’ll introduce consistent padding/margin using an 8px grid, apply typography styles (e.g. headers vs body text) uniformly, and use colors and surfaces from a Material palette. Key interactive elements (buttons, inputs) should get a styling overhaul – using proper hover states, focus indicators, and shadows/ripples where appropriate, akin to Google Workspace apps. We have already included Material-like design tokens (e.g. CSS variables like --md-surface and --md-divider) in the code[[28]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L8-L15), but need to apply them comprehensively. A cohesive stylesheet or design system will be established so that the app feels consistent and modern. We’ll also replace any technical jargon or placeholder controls with user-friendly components. For instance, the ball handoff input will become a dropdown menu of users with names (and maybe avatars), so testers can select a person by name instead of typing an ID. Overall, the UI should be **clean, intuitive, and visually on par with a production app** after this polishing pass.
* **Navigation and Layout Flow:** Some users found the navigation structure a bit confusing initially. The app currently has a top header with “Projects” and “Reports” links[[29]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L20-L25), as well as a tab menu for Projects, Dashboard, All Tasks, etc. This redundancy could confuse users (e.g. two different ways to view projects). Also, when viewing tasks for a project, the context wasn’t always obvious (the earlier design had a back button labeled “← Back” inside the task list card[[30]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L106-L114)). **Resolution:** Simplify and clarify the navigation. We will likely stick to one primary navigation scheme. For example, use a left sidebar or a top menu bar for main sections (Projects, Dashboard, Reports), and within a project, clearly show the project name and provide in-page controls (like the create task button). The header will consistently show the app name/logo and the notifications icon, while the section navigation can be consolidated. Additionally, we should label pages and use breadcrumbs if necessary so testers always know “where” they are (e.g. “Projects / Project Alpha / Task 123”). The goal is a **smooth interaction flow**: testers can easily go from the dashboard overview into a project, see its tasks, click a task to view details, and come back out. We’ll eliminate any extraneous steps or duplicate navigation elements that might confuse this flow.
* **Miscellaneous UX Quirks:** A few other minor issues came up that we will address:
* The **“Dev Mode” status badge** in the header (“Coordination • Alpha”) is fine as a version indicator, but combined with the dev banner it reinforced the unfinished feel. With the dev banner gone, the “Alpha” badge can stay for internal testing but we might style it subtly (it’s currently a stark black badge[[31]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L16-L24)).
* Some forms lack validation or feedback – e.g. if creating a task fails, users only get an alert dialog. We should show inline error messages in a friendly way.
* The app lacks any tour or help. Given family testers may not be tech-savvy, a brief on-boarding blurb on the dashboard (“Welcome to Eden Task – here’s how to get started: create a task, assign it, use comments…”) could be helpful. Even a Notion page or PDF guide could accompany the test.
* No easy way to see “my tasks” vs “all tasks”. We have an All Tasks view, but we might highlight tasks that are “in your court” for each user. For example, a filter or section for “Tasks waiting on you” would leverage the ball-in-court data usefully. This wasn’t explicitly requested by testers but aligns with the central hub concept and will likely improve UX for accountability.

In summary, the current UI/UX, while functionally complete, needs a thorough **polish and usability sweep**. Removing obvious dev artifacts, improving visual design consistency, and addressing the points above will significantly enhance the user experience. The aim is that when family testers use the updated module, it feels **intuitive, professional, and ready for serious use**, not a prototype.

## Immediate Next Steps for Refinement

To transform the Eden Task module into a test-ready, impressive desktop app, we will execute a focused set of improvements. These steps target both the **user interface polish** and any remaining feature gaps or bugs. The immediate action plan is as follows:

* **UI/UX Design Overhaul:** Apply a comprehensive visual update to align with Material Design and Google Workspace standards. This includes using a neutral light background with card-style containers, consistent font styles, and proper use of color for emphasis. We will introduce a primary accent color for actionable items (e.g. the “Create Task” button could be a standout color matching company branding). All buttons will get consistent styling (padding, rounded corners, hover effect) and forms will use clearer layouts and labels. Minor components like icons will be added for clarity (e.g. use a 🏷️ icon for tags, a 📎 for attachments, etc., or Material Icon equivalents). Overall spacing will be standardized so that screens don’t feel cluttered or misaligned. We’ll leverage the existing CSS variable theme tokens to ensure colors (surface, divider, warning, etc.) are used appropriately for light/dark elements[[28]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L8-L15). By the end, the app’s look-and-feel should evoke a polished, cohesive product.
* **Improve Interaction Flows:** Streamline how users interact with tasks:
* Implement a **User Picker for Handoff** – replacing the UUID text input with a dropdown of users (showing names/emails). This will make handing off a task a quick, error-free action for testers. We might also allow adding a short note during handoff (the backend supports a note field for ball handoff) so context can be given when passing the task[[32]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L56-L64).
* Enable **inline edits** for certain fields. Currently, status changes are done via buttons, which is fine, but editing a task’s title/description or assignee after creation might require an “Edit” mode or turning those fields into editable inputs. We will add an “Edit Task” option in the task detail view so users can update the task info without going through an obscure flow. This could be a button that toggles input fields for title/description or opens an edit dialog.
* **Confirmation and Undo**: Add confirmation prompts for destructive actions like deleting a task (already partially in place with a JS confirm) but style it nicely (a modal dialog). Also consider an “Undo” snackbar after quick actions like status changes or deleting a comment, to improve user confidence when testing.
* **Consistent Navigation & Breadcrumbs:** As noted, unify the navigation. We will likely remove the redundant tab bar and rely on the header links (or vice versa). Every page (Dashboard, Projects list, Project detail, Task detail) will have a clear title and a back or home navigation element so users never feel lost. For example, on the Task Detail page (or drawer), we can show the task title and under it a link back to the project or list.
* **Side Panel for Task Details:** To create a more dynamic desktop experience, consider keeping the task list visible while viewing a task’s details. Instead of always navigating to a separate page, we can use a **slide-out detail drawer** (as implemented in the code with TaskDetailDrawer component) for viewing and editing a task, overlaying on the right side of the list. This interaction (common in Gmail or Trello) allows users to flip through tasks without losing context of the list. We’ll test this approach – it can make the app feel faster and more like a “hub”. If it complicates navigation for testers, we will at least ensure the separate Task Detail page is easily accessible and has a Back button.
* **Responsive (Desktop) Layout:** Ensure the layout works well on various desktop/laptop screen sizes. We don’t target phones now, but it should handle, say, a smaller laptop or a large monitor gracefully. Using the max-width container (1200px) keeps content centered on big screens[[33]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L10-L18). On smaller screens (like small laptops), the design should still display all information without breaking (we will test down to around 1024px width). Scroll bars and overflow should be handled so that modals or panels don’t get cut off. While not full responsive design, this guarantees our family testers (who may use different computers or an iPad in desktop mode) have a smooth view.
* **Feature Completeness & Enhancements:** Close any functional gaps so testers can exercise all critical features:
* Finish the **Attachments UI**: ensure that users can upload and download attachments easily. We’ll test file uploads (the multi-step upload flow) with typical files. We might add an inline file preview or at least an icon indicating file type for each attachment in the list. If possible, enable image thumbnails or a quick way to open attachments in a new tab. Also, handle errors (e.g. file too large) gracefully.
* **Filtering and Search:** Expose filtering options in the UI for task lists. For example, on the “All Tasks” view, allow filtering by status (dropdown: show me only open or only done tasks), by assignee, or by project. The backend already supports query filters for tasks[[34]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L92-L101), so it’s mainly a matter of adding UI controls. Similarly, implement a basic search bar that can search task titles or tags. This might help testers with many tasks to find specific ones and will demonstrate the “central hub” capability (finding any task globally).
* **Project/Task Metadata Visibility:** Some task fields like tags, department, origin (how the task was created) exist but might not be fully shown. We will surface tags clearly (perhaps as chips on the task detail – which is partly done[[35]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L128-L137)) and possibly allow adding tags in the UI. Department or project association is largely inherent by where the task is created, but on a task detail we can display the project name and department responsible. Ensuring that metadata is visible will help testers contextualize tasks (e.g. “Oh this task is part of Project X and in the Coordination department”).
* **Notifications Workflow:** Verify that notifications are working end-to-end in the UI. When a tester assigns a task to another tester, does the assignee see a new notification in their bell icon? When a comment is added, do relevant people see it? We will fix any bugs in the notification badge count or dropdown. If feasible in our test environment, we might integrate an email or at least simulate one (for example, have the system send a summary email to an address we can check). However, email/SMS is optional for the family test since the in-app notifications should suffice. We will also ensure the **“Mark all as read”** and refresh functions on the notification panel work as intended[[36]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L126-L134). The goal is to make notifications a reliable way for users to stay updated without constantly refreshing the page.
* **Permissions & Roles Testing:** For the family test, we may keep everyone as an “Admin” or “Manager” to simplify things (full access). But we should still test that the role logic doesn’t inadvertently hide something. For example, if a user is marked as a Guest, do they see their limited tasks? We’ll double-check that permission middleware on the server doesn’t block normal actions in unexpected ways during testing. Since the focus is not on security right now but on UX, we might run the test in a “friendly” mode where roles are not restrictive, then later enforce them for real use.
* **Quality and Polish:** In addition to features, take care of general quality items:
* Fix any known **bugs or performance issues** (e.g. if the task list sometimes fails to load or the dashboard charts have errors, address those). Ensure the app runs smoothly without console errors or crashes during normal use.
* Conduct a content review: all text in the UI (labels, messages, alerts) should be clear and free of typos. Use consistent terminology (e.g. if we use “To-Do” vs “Open” consistently – currently both appear in code for status[[37]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L20-L28), we should settle on one naming scheme visible to users). Also, ensure the term “Ball in Court” is explained or intuitive – we might change the label to “Current Owner” or have a tooltip explaining “This is the person currently responsible to act.”
* Remove any **leftover developer elements** besides the dev banner. For instance, if the console logs are visible or any debug info on screen (like raw IDs), strip those out for the test build. The presence of an “Alpha” tag is acceptable as a version label, but nothing more that makes the user feel like they are using an unfinished system.
* Prepare some **sample data**: populate the system with a few projects and tasks so that when family testers log in, they see an example of how it looks (rather than an empty screen). For example, create a sample project with a few tasks (some open, some done) to illustrate the dashboard charts and allow them to play around before adding their own. This will make the first impression better and testing more efficient.

By executing the steps above, we aim to elevate the Eden Task module from a working prototype to a **near-production, polished application**. The emphasis is on **usability, clarity, and completeness**. Each feature will be revisited with a coat of polish so that internal testers can use the system without frustration or confusion, and so the module makes a positive impression as a core tool for coordination.

## Testing Plan with Family Users and UX Evaluation

Once the refinements are in place, we will conduct a structured internal testing phase using family members as trial users. The objectives are to verify the module’s functionality in a real-world simulation and to gather qualitative feedback on user experience. Below is the plan for rolling out testing and capturing insights:

1. **Test Setup:** Deploy the Eden Task module to a stable testing environment (e.g. a cloud URL or local server accessible on the home network) that the family can access on their desktops/laptops. We will create user accounts for each family tester, covering a range of roles for realism – e.g. one user as “Manager” (with ability to see all tasks), others as “User” (sees tasks assigned to or created by them, if we simulate that permission), and maybe one as a limited “Guest” to simulate an external collaborator view. However, as noted, we might keep roles relatively open during testing to avoid confusion, and focus on UI behavior.
2. **Onboarding Briefing:** Before they start, provide a short guide or introduction. This can be a one-page document or a quick in-person walkthrough explaining the basics: how to create a task, how to assign it, what “ball in court” means, how to comment, and how to mark tasks done. Emphasize that this is an alpha test, and encourage them to speak aloud any confusion or expectations as they use the app (if we are observing) or to jot down notes. If family members are non-technical, use analogies (e.g. “creating a task is like sending someone a to-do, and ball in court is like handing them the baton to act next”).
3. **Testing Scenarios:** Have the testers perform common workflows:
4. **Task Creation & Assignment:** User A creates a new task (e.g. “Plan family vacation itinerary”) and assigns it to User B. Verify they find the Create Task button easily and can fill out the form without issues. Observe if they set a due date or any tags and whether anything in the form confuses them.
5. **Task Notification:** After assignment, User B (the assignee) should notice a notification or see the task appear in their view. Ensure User B checks the Notifications bell and can read the task details. This will test whether notifications are noticeable enough. We might simulate an email notification as well and ask if they saw an email (if that feature is active).
6. **Ball Handoff (Reassignment):** Have User B, instead of completing the task, use the **handoff** function to delegate the task to User C (another family member). Check that User B can identify the handoff control and select User C easily (thanks to the new dropdown, presumably). User C should get a notification of the handoff. This scenario will reveal if the “ball in court” concept is understandable to them. We should ask afterwards: “Did you understand what handing off the ball meant?” and see if they interpret it correctly. If not, their feedback might suggest renaming or better explaining it in the UI.
7. **Commenting & Collaboration:** Have one user add a comment to the task (e.g. “I’ve done part of this, please check.”). Then another user replies with another comment. Ensure they can easily find the comment box and that comments update in real-time (or via refresh). See if they expect an email or notification for comments and ensure the system provides one. This will test the interaction of the comment UI – we’ll watch if they struggle with the text input or don’t notice the comment section.
8. **Status Updates:** Direct a user to change the status of the task (e.g. mark it “In Progress” then “Done”). Confirm the status buttons or dropdown are intuitive. For example, if the task is moved to “Done,” does the UI clearly reflect it (maybe with a green check or strikethrough)? If a task is done, see if the testers look for it in a “completed” section or if they expect it to disappear. Their behavior can inform if we need to clarify how completed tasks are shown (maybe they stay in list but labeled done, etc.). If subtasks are in use (we might not push that heavily for family testing), we could test adding a couple subtasks and completing them to see the auto-close of the parent task.
9. **Dashboard Use:** Ask the “Manager” user to interpret the Dashboard. Do they understand the charts (tasks by status, by assignee) and do those numbers make sense relative to what they did? This will check if our summary data is correct and if the visualization is user-friendly. If the charts or terms confuse them, we’ll note that for refinement.
10. **Attachments:** If feasible, have a user attach a file (maybe a photo or PDF) to a task and have another user try to view/download it. This tests the file upload flow and whether the attachment is easily accessible. It will also reveal if any UI feedback (like showing upload progress or file size limits) is needed.
11. **Collecting Feedback:** After these tasks, gather the family for a debrief. Encourage them to share what they found easy or hard:
12. **Clarity:** Did they understand what to do at each step? Were any labels or buttons unclear? (For example, do they know what “All Tasks” means vs “Projects”, or did anyone click “Reports” and wonder what it does.)
13. **Visual Appeal:** What did they think of the look and feel? Do they have any comments like “it looks professional” or “it felt a bit plain/old-fashioned”? Family users often will be honest if something looked off.
14. **Navigation:** Could they navigate between projects, dashboard, and tasks without help? Or did they get “lost” at any point?
15. **Missing Features:** Did they expect to be able to do something that the app didn’t allow? For instance, perhaps someone tried to drag-and-drop to attach a file (and we need to support that), or someone wanted to sort tasks by due date, etc. These observations can prioritize future enhancements.
16. **General Satisfaction:** Would they find this tool useful for coordinating real tasks? This is more subjective, but their impression matters. If they suggest “I would just WhatsApp instead” then we know we have to make using the app more compelling. On the other hand, if they say “This is neat, I can see all our stuff in one place,” that’s a good sign.
17. **Iterate on Feedback:** Based on the testing, we will refine further as needed. Perhaps a family member finds the term “Tasks in Court” odd – we might change the terminology to “Pending with X” in the UI. Or if they all missed noticing the notification bell, we might add a more aggressive indicator (like flashing or a sound, though sound might be too much). We should be prepared to make small adjustments quickly and possibly run a second round of tests. Family testers can also be asked to continue using the module over a few days for any real small project (even a non-work one, like planning an event) to see how it holds up over time and repeated use.

Throughout the testing, our aim is to evaluate **UX comfort and intuitiveness**. We want new users to successfully use the Eden Task module with minimal instruction. Any point where a tester stumbled or was confused is golden feedback to improve the design. By the end of the family testing phase, we should have high confidence that the Task module is user-friendly and robust, ready for broader internal deployment.

## Out-of-Scope Areas (to be Deferred)

To maintain focus, certain aspects will intentionally **NOT** be addressed in this phase. These include:

* **Mobile Responsiveness:** We will not be optimizing the Eden Task module for mobile or small-screen use at this time. The current interface is designed for desktop browsers. Making a great desktop experience is the priority; mobile layouts (responsive CSS, touch optimizations, etc.) will be tackled in a later phase. Family testers are expected to use laptops/desktops, so any mobile access would be incidental. We acknowledge the importance of mobile in the long run, but for now it’s out of scope.
* **Broader ERP Integration:** The Task module will operate in isolation without integrating with other ERP modules (Procurement, Accounting, etc.). For this phase, we won’t attempt single-sign-on across modules or a unified navigation shell that switches between modules. The **ERP shell and multi-module architecture** work is paused. We also won’t link tasks to external data (for instance, a task won’t yet automatically generate a QuickBooks entry or connect to an estimate record). The focus is purely on the task coordination features. Integration points (like creating tasks from emails or linking tasks to project management schedules) will be noted for future development but not implemented now.
* **Advanced Automation & AI Features:** Any AI-assisted functionality or complex automation (Phase 2/3 items) will be deferred. That means features like automated task scheduling, predictive reminders, or the reward/point system for performance are not in the current plan. Our daily summary notifications are still basic (no actual email sending yet). We won’t work on AI-driven insights or recommendation engines until the core user experience is solid. This also includes the “CFO intelligence engine” or other analytics mentioned in vision documents – those remain future work.
* **Performance Tracking and Leaderboards:** Although the vision included rewarding on-time completion and point scoring[[38]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0), we will not implement the scoring or leaderboard system in this phase. We’ll collect the data (task completion times, etc.) but not surface any gamification just yet. It’s more important that users simply use the system; the motivational features can be added once usage is established.
* **External Integrations (Email, Voice Input)**: We have placeholders for creating tasks via email and voice (voice attachments with transcription)[[19]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0), but we won’t fully implement email ingestion or live voice dictation now. For testing, we might simulate an emailed task by manually using the “origin” field or just have testers manually input tasks. Real integration with an email inbox or a voice-to-text service will be considered later. The UI may show a disabled button or info text like “(Voice input coming soon)” to communicate the future feature, but clicking it won’t do much in this version.
* **Non-Core Modules:** Any features outside of Task coordination, such as the **Reports** section beyond basic task metrics, will be deprioritized. There is a “Reports” tab in the UI, but we will not expand that right now, except perhaps a simple placeholder page. Similarly, user management, authentication improvements, or other ERP settings will remain basic (we’ll use what’s already there, e.g. a simple login, without adding complexity like OAuth or detailed user profile pages).

By explicitly deferring these areas, we ensure all efforts stay directed at the **task management experience**. The rationale is that a well-received, highly usable Task module will build confidence and provide a strong base onto which those other features can later be added. Once the Task module is polished and tested internally, we can then return to broader ERP integration and mobile support with lessons learned and a clearer UI direction.

By following this plan, we will deliver a **desktop-first Eden Task module** that is polished to near-production quality. It will showcase high attention to UI/UX (comparable to modern productivity tools) and comprehensive task coordination features (creation, assignment, ball-in-court tracking, notifications, comments, attachments, etc.). The module will serve as a **central coordination hub** for EDEN, ready for broader internal adoption after the successful family pilot. Crucially, we focus on making this one module excellent, while temporarily setting aside ancillary concerns. This concentrated approach will result in a testable, impressive application that can anchor the next stages of our ERP development[[24]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0).

[[1]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[2]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[7]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[10]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[18]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[19]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[23]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[24]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) [[38]](https://www.notion.so/2861f729313880c593d8f58f57f1efa0) EDEN Coordination System

<https://www.notion.so/2861f729313880c593d8f58f57f1efa0>

[[3]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L148-L156) [[4]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L160-L168) [[30]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx#L106-L114) TaskList.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskList.jsx>

[[5]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L212-L220) [[15]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L270-L278) [[20]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L310-L319) [[27]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L218-L226) [[35]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx#L128-L137) TaskDetailDrawer.tsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/TaskDetailDrawer.tsx>

[[6]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L64-L71) [[11]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L94-L101) [[12]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L94-L102) [[36]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx#L126-L134) NotificationsBell.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/NotificationsBell.jsx>

[[8]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/services/taskAutoClose.js#L8-L16) [[9]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/services/taskAutoClose.js#L59-L67) taskAutoClose.js

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/services/taskAutoClose.js>

[[13]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/notifications.js#L170-L178) [[14]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/notifications.js#L140-L148) notifications.js

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/notifications.js>

[[16]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L38-L46) [[17]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L72-L80) [[21]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L28-L33) [[32]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L56-L64) [[34]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L92-L101) [[37]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js#L20-L28) tasks.js

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/routes/tasks.js>

[[22]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/DashboardPage.jsx#L38-L46) DashboardPage.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/DashboardPage.jsx>

[[25]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/DevAuthSwitcher.jsx#L40-L48) DevAuthSwitcher.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/45411c8264f6eb4d65c9adf0f38373646504c5c7/apps/coordination_ui/src/components/DevAuthSwitcher.jsx>

[[26]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/ProjectDetail.jsx#L33-L37) ProjectDetail.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/pages/ProjectDetail.jsx>

[[28]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L8-L15) [[29]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L20-L25) [[31]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L16-L24) [[33]](https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx#L10-L18) EdenHeader.jsx

<https://github.com/edenplumbinginc-svg/Eden_erp/blob/1b5b050dd6cf1061c8d8f1ba77cf92fe545efa32/apps/coordination_ui/src/components/EdenHeader.jsx>