ChatGPT

Evaluation of "UX Redesign: Task-Based Architecture (Approach C)" Plan for Groupwheel

1. Assessment of Proposed UX Principles

One Task Per Screen: The plan embraces a *focused screen* model where each page answers a single question (e.g. Dashboard = “Which activity?”, Setup = “What groups do I have?”) [1]. This principle is sound for reducing cognitive load, but its execution must be scrutinized. In practice, some proposed screens still bundle multiple subtasks. For example, the new **Setup** page is meant to handle roster management, group configuration, **and** preferences all in one place [2] [3]. While conceptually these subtasks all contribute to “setting up groups,” cramming them together risks violating the very one-task-per-screen ideal. Careful UI design (e.g. sections or progressive disclosure) is needed so the Setup page remains straightforward despite its breadth. Overall, “One Task Per Screen” should improve clarity, but the implementation must ensure each screen’s scope truly feels singular to the user.

Linear Paths, Not Webs: The redesign favors a linear, wizard-like navigation flow with clear forward/back progression and minimal branching [4]. This addresses prior issues of teachers getting lost in a “maze” of options by guiding them step by step. The benefit is a more guided experience – especially for new users – with one primary action per screen and a clear “escape route” (cancel/back) [4]. However, a strictly linear flow can introduce *rigidity*. Experienced or returning teachers might want to jump directly to a specific task (e.g. editing groups or adjusting settings) without stepping through a sequence. The plan partially addresses this by letting returning users bypass the wizard (going from the dashboard directly into Workspace or Setup) [5]. Still, eliminating the ability to navigate freely (the original design had more interlinked pages) might frustrate power users. The linear path principle should be balanced with some on-demand shortcuts (for example, a direct link to the Setup page for an activity) so that the UI doesn’t become *too* constrained. As long as the linear flow is default but not mandatory in all cases, this principle will improve UX by reducing confusion.

Teacher Vocabulary Only: The plan explicitly bans internal jargon in favor of terms teachers use daily [6]. For instance, what the code/domain calls a “Program” or “Scenario” will be labeled as an **“Activity”** or **“Groups”** in the UI [6]. This is a positive change – aligning language with users’ mental models improves usability. It means UI labels and navigation will be clearer (e.g. a teacher sees “Activities” and “Roster” instead of unfamiliar terms like Program/Pool). One caveat is ensuring **consistency**: existing text and documentation must all be updated. The codebase shows many domain references (e.g. `Pool`, `Program`, `Scenario` classes) [7] [8], but the UI should abstract those away. Another risk is if some “teacher-friendly” terms might be ambiguous. For example, calling everything an “Activity” is broad – the UI should contextually clarify that it means a grouping project (perhaps by examples or icons). As long as the terminology is applied uniformly and the old jargon purged from the interface, this principle will greatly reduce user confusion.

Smart Defaults Over Configuration: The plan prioritizes automatic “sensible” behavior and hides advanced settings [9]. In practice, this means the grouping algorithm runs automatically with default parameters and **power features are hidden until needed** [9]. The intent is to accommodate impatient teachers – the majority should get good results without tweaking options. This aligns with known usage: the MVP currently only provides one grouping algorithm run by default [10], and indeed teachers currently get a single auto-generated grouping unless they manually regenerate [11]. The upside is a faster path to results (e.g. the wizard will auto-run the algorithm at the end and jump straight to the workspace [12]). The **risk**, however, is oversimplification: in edge cases where the default doesn't fit (perhaps a teacher has specific grouping constraints or preferences), the user might feel disempowered. Completely hiding configuration means advanced users may not even realize certain capabilities exist. The plan does acknowledge this trade-off by keeping advanced features *available* but tucked away (e.g. algorithm settings "collapsible, advanced" on the Setup page [13]). It will be critical to implement those escape hatches properly: for instance, a teacher who *wants* to fine-tune groupings or import detailed preferences can still find those options (perhaps under an expandable "Advanced Settings" panel). If smart defaults truly cover ~90% of cases as assumed [9], this principle will make the UI cleaner for most users – but it's important to validate that assumption with real usage data and provide a discoverable path for the other ~10%.

State is Obvious: The plan emphasizes clear state indicators so teachers always know *where they are, what they can do, and how to get back* [14]. This addresses past UX failures where users finished the wizard and then felt lost about what to do next [15]. Concretely, the redesign promises that at any given moment the interface will communicate the current step or mode and the next action. For example, the new dashboard activity cards will show a status ("○ Editing" vs "● Published") and present a clear primary action (e.g. a big "Edit Groups" button if the activity is in progress, or "View" if it's published) [16]. In the Workspace, the presence of an obvious "Show to Class" button signals the next step of presenting [17]. These cues should mitigate confusion. One area to watch is the **"smart redirect"** behavior for the Activity Hub route: the plan proposes that visiting an activity's base URL will automatically send the user to either Setup or Workspace depending on whether groups have been generated [18]. Technically convenient, this could momentarily confuse a user ("Why did clicking my activity take me somewhere else?"). The alternative mentioned – a minimal hub page showing the status with explicit buttons – might actually make state more obvious at the cost of one extra click. Overall, the spirit of "State is Obvious" is well-founded: the UI should use labels, headings, and navigation flow to continually orient the teacher. Implementation details like the redirect vs. hub page will determine how well this principle is realized. It may be safer to include at least a visual indicator during any redirect (so the user understands *why* they landed on Setup or Workspace). As long as each screen has clear titles and the navigation highlights where you are (e.g. a breadcrumb or consistent header naming), this principle will greatly improve user confidence in using the app.

2. Inconsistencies and Risky Assumptions in the Plan

Route Restructure (14 routes to 9): The consolidation of routes is logical but warrants scrutiny for completeness. The plan lists current vs. new routes [19] [20], eliminating several pages entirely. Notably, the **removed routes** include some that provided standalone advanced features [21]:

- `/groups/templates` – eliminated in favor of handling group templates inline on the Setup page [21]. This assumes all template functionality can be folded smoothly into the new Setup UI. If template management (creating, editing templates for group structures) was complex, the Setup page could become overloaded. It's risky if the plan doesn't allocate adequate UI for templates; simply "inline in setup" may understate the work needed. Any inconsistency here (like missing features that the old templates page had) would frustrate users who rely on saved templates.

- `/groups/[id]/candidates` – removed, with the idea that **candidate group options** will be shown inline in the workspace [22]. This is forward-looking; currently the app does not have a multi-option gallery (the MVP "teacher gets one option" gap is acknowledged [11]). The plan assumes that eliminating the separate candidates page now is fine because the feature will reappear integrated into the regenerate workflow. However, until the *Phase 2 Candidate Gallery* is implemented, teachers effectively lose the ability to see alternate groupings side-by-side. The plan's assumption is that auto-picking the "best" grouping is enough [23], which is a bit **risky** – "best" is defined by algorithmic metrics (like highest percentage of top choices) that may not always align with a teacher's judgment. If multiple algorithms or seeds were considered to pick that best option, that's not clarified and would need new implementation. Otherwise, "best" just means "the one default algorithm result," offering no real choice. In short, removing the candidates route is acceptable only if the promised inline candidates (or at least easy regeneration) truly materialize to cover that need.

- `/groups/[id]/students/[studentId]` – a per-student detail view, being **removed entirely** as "rarely used" [24]. This is an assumption that could backfire if teachers or admins ever needed to drill down into an individual student's info or preferences in the context of a grouping. If indeed usage data shows it was rarely touched, removing it simplifies the app. But consider consistency: if a teacher clicks on a student name expecting more info (a natural action), now nothing happens or it's not allowed. The plan doesn't mention any alternative way to view a student's profile/preferences in-group. This removal might be okay for now, but it's a *risky assumption* that no one will want that detail. It may warrant either confirming it via user research or planning a lightweight alternative (e.g. a hover tooltip with student's details or showing their preference satisfaction in-line).

- `/groups/[id]/sessions` – removed, with "sessions" (the history of published group sets) folded into the Activity Hub or Setup [24]. The idea is to present past published groupings in the new Setup page's "History" section [25]. This makes sense to reduce separate pages, but it assumes the new HistorySection will fully replace the old sessions page. The plan should ensure *nothing is lost*: e.g., can teachers still view or export previous grouping sessions? Will the Setup page clearly indicate how to see past group assignments? If the implementation of this is unclear, there's a risk that teachers won't realize history exists (if it's too hidden) or that it might not handle all data (e.g. compliance with data retention – an open question in docs [26]).

- `/algorithms` – dropped, to be possibly moved to static help/docs [27]. This implies any UI for selecting or learning about algorithms is gone from the app's main flow. It's probably fine (teachers likely don't tweak algorithms often in the current MVP), but it's based on the assumption that algorithm configuration is not needed in-app. If some advanced user wanted to choose a different algorithm or see how the algorithm works, they now cannot do so directly – a potential point of frustration. The plan's mitigation is to document it externally, but that relies on users seeking out documentation. This removal is only safe if indeed >90% of use-cases never required touching algorithm settings, which might be true given MVP only had one default algorithm [28].

- `/groups/new/sheets` – removed, with Google Sheets roster import integrated into the main wizard flow [29]. This is mostly a UI simplification (embedding the sheet connect option in the "Students" step). It should be fine if executed – the risk is low as long as that functionality isn't lost. The plan specifically lists "keep paste, simplify sheet [import]" as a task for the new StepStudents [30], indicating awareness of this need.

One inconsistency to note: the current app uses a `/scenarios/[id]/student-view` route for the read-only presentation mode [31], but the plan doesn't explicitly list `/scenarios/...` in current or new routes. Instead, it introduces `/activities/[id]/present` as the new presentation route [32]. The omission of the old `/scenarios` route in the "Current Routes (14)" list suggests a minor oversight – the plan should ensure that the old **student view route** is also redirected or handled. It's a small detail, but important for completeness: any existing link or internal reference to the student-view should be mapped to the new `/present` page to avoid broken navigation.

Wizard Simplification (3-step wizard): The plan's new wizard flow drops from 4–5 steps to 3 [33], which is a bold change. The removed steps are *Name*, *Select Roster*, and *Preferences*, with their functions either automated or merged:

- **Removing the "Name Activity" step**: In the current wizard, the final step prompted teachers to name the activity and review details [34] [35]. The new plan cuts this, instead auto-generating the activity name "from the first group or roster" [23]. This is potentially inconsistent with user expectations – naming something is a personal touch and helps identification later. The plan assumes an algorithmic name (perhaps using the roster name or first group name) will be "good enough." This is *risky* if the auto-name is too generic ("Untitled Activity" or "New Groups") or confusing. For example, if a teacher pastes a roster of "Mrs. Smith's Class" and doesn't explicitly name the activity, will the system call it "Mrs. Smith's Class Activity"? Or if the teacher defines one group name (say "Art Club"), will it name the whole activity "Art Club"? There's a chance of weird or non-unique default names. The plan doesn't mention how a teacher can **rename** an activity later – presumably they could on the dashboard or in an Activity settings, but no UI for renaming is described. This requires additional implementation (e.g. making the activity title editable on the dashboard card or the Workspace header) to avoid a situation where many activities end up with system-generated names that teachers can't change easily. A safer alternative might be to allow editing the name during the "Review" step (instead of fully removing it, maybe pre-fill a suggested name that the user can override). If truly removing it, the design should clearly show the generated name somewhere and allow later edits, otherwise this simplification could backfire in usability.

- **Combining or eliminating roster selection (returning user flow)**: The current system has an initial "Select or import roster" step for returning teachers (hence 5 steps for them, including a roster reuse step) [36]. The plan's "inline choice" approach [12] suggests that the new Step 1 (Students) will handle both cases: either paste a new roster or pick an existing one. It's crucial that this is done, because simply removing the select roster step without alternative would be a regression – teachers who have an existing roster (Pool) must be able to reuse it easily. The plan does not detail the UI for this, representing a **needed clarification**. An assumption might be that StepStudents will show a dropdown like "Use an existing roster?" if any are available, before or instead of the paste field. If that isn't implemented, returning teachers might be forced to re-import the same student list, which is inefficient and risks duplicates. So, while the plan simplifies the wizard path, it *must not drop functionality*: the design should incorporate roster reuse seamlessly (e.g. a simple toggle or list of saved rosters on the "Students" step). This is an example of an inconsistency where the plan emphasizes streamlining but needs more detail to ensure no user need (like roster reuse) is lost.

- **Removing the Preferences step**: The original wizard had an optional step to import student group requests/preferences via CSV [37], which is a power feature (helping the algorithm honor students' choices). The plan removes this as a dedicated step, which is a significant change. The assumption is that preferences input can be handled later “inline with groups” (38) or on the Setup page as an optional section (25). This is potentially risky and inconsistent with how some teachers might expect to set up their activity. If a teacher comes prepared with preference data (e.g. a form of who wants which club), the new wizard doesn’t ask for it at all – it jumps straight to generating groups without those inputs. The teacher would then have to realize *after the fact* that they should go to the Setup page’s Preferences section, import the data, and regenerate the groups. That workflow is not explained in the plan’s described flows, and could lead to confusion (“I had student choices – why wasn’t I asked for them? Now how do I incorporate them?”). The plan is assuming that simplifying the initial flow (fewer steps) is worth deferring this advanced task. It’s a trade-off: many teachers may not have preference data initially, so skipping it speeds things up, but for those who do, it adds a roundabout path. **Inconsistency** arises if the UI doesn’t somehow hint at this option. Perhaps the plan expects teachers to only worry about preferences after seeing the initial grouping (progressive disclosure). This needs careful implementation: the Setup page’s “Preferences (optional)” section (25) must be easy to find and possibly a call-to-action like “Have student requests? Import them here to improve group matches.” Without that guidance, the removal of the wizard step could be a usability pitfall. In summary, dropping the Preferences step is daring the user to have a simpler first experience, but it assumes teachers either don’t need it upfront or will somehow know to handle it later – a somewhat **risky assumption** that should be validated.

- **Auto-running the algorithm and no candidate selection:** The wizard’s final step (Review) in the new design doesn’t let the user pick from multiple groupings; it will automatically generate one solution and proceed (23). The plan explicitly notes “no candidate gallery (auto-pick best option, can regenerate in workspace)” (38). The assumption is that the system can choose a “best” grouping (likely by maximizing the satisfaction metrics like top-choice percentage) without user input. This raises a question: how is “best” determined? Currently only one algorithm run exists (39), so presumably “best” just means that one result. If the plan intends to quietly run several algorithms and pick the top result, that’s new behavior requiring implementation. Regardless, not giving any choice in the wizard could be fine for novices (who might not know how to choose anyway), but it might frustrate users who *do* understand the trade-offs and want to see alternatives. They do allow manual regeneration later, but that’s less discoverable than an upfront gallery. An inconsistency to watch: the plan describes this as temporary – it positions the candidate gallery as an optional feature for later (40). So effectively they assume that temporarily removing choice is okay because it will come back in Phase 2 or 3 as an inline gallery. This is a gamble: if Phase 2 delays, users are stuck with one-shot groupings for now. The plan’s success criteria (“wizard steps reduced from 4-5 to 3” as a quantitative win (33)) need to be balanced against user satisfaction when they inevitably ask “What if I don’t like the grouping? Could I have seen other options?”. In the interim, the regenerate button in the Workspace must be very visible and perhaps guided (maybe a tooltip like “Not happy? Click Regenerate for a different grouping”).

In summary, the wizard simplification rests on the assumption that fewer steps = happier teachers. It’s generally true that a shorter onboarding is less daunting, but the removed steps had their purposes. The plan will need to **carefully design Step 2 (Groups)** to cover both group naming/sizing and maybe a hint of preferences, and **design the post-wizard experience** (Setup/Workspace) to cover what was skipped. Any inconsistency in covering those tasks elsewhere could leave gaps in the user experience.

Removal/Demotion of Power Features: A core theme is pushing power-user features out of the main flow and into secondary UIs or hiding them behind toggles. The plan calls for removing or downplaying features like detailed analytics, history management, layout switching, and export options:

- The **analytics panel** (which presumably showed detailed stats and charts about the grouping) is slated for removal, replaced by a simple inline summary [41] [42]. This simplifies the workspace visually, but it assumes teachers don't need the deeper analysis readily. If some teachers valued those analytics (e.g. a breakdown of how many got their 1st, 2nd choice, etc. in detail), making it non-expandable could be a loss. The plan bets that a compact summary ("72% got top choice" shown in the toolbar as in the mockup [43]) is sufficient for decision-making. Likely this is fine for most, but it's worth noting that any desire for more analytics would now require, say, exporting data or future features – a trade-off in capability.

- **History/undo vs. session history:** The plan removes the visible history dropdown or selector in the UI [41] [42]. Currently, there is a session-scoped undo/redo stack implemented for editing changes [44], and possibly a concept of scenario history for multiple publish sessions. The HistorySelector component likely allowed switching between saved scenarios or previous states, but since MVP only had one scenario per activity, it might have been anticipating future versioning. By removing it, the design assumes that *persisting multiple versions isn't needed right now*. This matches the domain model (only one "Scenario" per Program in MVP) [8]. Instead, they plan to log published sessions in a History panel on Setup [25] for reference. The risk here is if a teacher expected to undo beyond the current session (like "go back to last week's grouping"), they now have to use the history in Setup which likely is read-only (the plan doesn't say you can revert to a past session, just view it). Also, by hiding undo/redo in a small toolbar (they keep undo/redo buttons but remove overt history UI [41]), they must ensure that the session-bound undo still works intuitively and that users know its limits (once you leave the page or finish a session, you can't undo further – which is already the case [45] [44], but should be clear). This demotion is probably acceptable – it cleans up clutter – but it assumes no near-term need for persistent history. If that assumption is wrong, reintroducing a user-friendly way to fork or restore prior groupings might be necessary.

- **Preferences as an "advanced" feature:** As discussed, preferences input is now optional and hidden under a collapsible section on Setup [25]. This demotion assumes that only power users will seek it out. It also implies that the main flow (automatic grouping) might ignore subtle constraints that some teachers would want. It's a trade-off between simplicity and control. The risk is less about technical feasibility (the code already supports importing preferences [37], it's just moving the UI to a different place) and more about discovery: Will a teacher who *does* have preference data know to click "Preferences (optional)" on the Setup page? It might require an in-app hint or onboarding note to be safe.

- **Sessions/Multiple group sets:** As noted, moving session history out of the main header into the Setup page reduces clutter for most users (who may not even run multiple sessions). But a teacher running iterative group assignments over a semester might find it harder to notice where those past sessions went. The removal of the "Sessions" button from the header [41] is predicated on the assumption that viewing past sessions is not a frequent need during daily use. If it is needed, the teacher now has to navigate to Setup > History. This is likely fine (history is reference info, not a primary action), but it introduces a context switch that wasn't there before. A possible inconsistency:

the plan's Activity Hub idea originally mentioned showing sessions on the hub [24], but since they lean toward auto-redirect, they must ensure the Setup page's History fulfills the role.

- **Layout toggle removed**: The current UI apparently allowed toggling how groups are displayed (perhaps a list vs grid view, or grouping by different criteria). The plan chooses one "good default" layout and removes the toggle entirely [41] [46]. This is a classic simplification – one less control to worry about – but it assumes the chosen layout works for everyone. If, for example, some teachers preferred a list because they have many groups (the grid might require scrolling) or vice versa, they lose that choice. The design team likely picked whichever format testing showed was clearer. It's a reasonable simplification, but it's an irreversible one – reintroducing a toggle later would be extra effort if complaints arise. The code will need a small cleanup (removing the toggle component and associated state).

- **Export and other power controls hidden in a menu:** The plan moves the export function (and presumably any similar seldom-used actions) into a "… menu" rather than a visible button [41] [46]. This is fine as long as teachers know where to find it when they need it. Exporting (to CSV, Google Classroom, etc.) is often a later-stage action, not something you do constantly, so demoting it from the main toolbar is justified. Just ensure the menu is clearly accessible (maybe labeled or with a clear icon). A potential risk is if that menu also hides something like "Delete activity" or other settings – important actions should still be somewhat visible. But given export was previously prominent and is now tucked away, that's an okay trade-off to reduce distraction.

In general, these removals and demotions are consistent with the *90/10 rule* – design for the 90% common use, hide the 10% advanced. The **risky assumption** is getting the percentages right. If any of these features were actually used by, say, 30-40% of users regularly, hiding them will cause pushback. The plan does rely on some subjective judgment of "rarely used" (e.g. student detail route) or "clutter" (analytics panel). It would be wise to verify usage data if available. Technically, none of this is undoable – the code can keep these features behind flags or conditional UI. But the plan clearly intends to remove the code entirely for cleanliness (files like `AnalyticsPanel.svelte` and `HistorySelector.svelte` are slated for deletion [42]). That means fully committing to the simplification. Any inconsistency between what the designers consider "power features" and what users actually consider important could be problematic. For example, if some teachers heavily used the friend preference import to ensure fair groups, they might be baffled to find that step missing. Mitigation can include strong communication (release notes or in-app guide: "We streamlined the creation process; you can still import preferences later via the Setup page"). Without that, the assumptions behind these removals might introduce learning curves or disappointment for some users.

Reliance on Smart Defaults: The plan heavily leans on the idea that the system will do the right thing automatically for most users – effectively treating the algorithm as an "auto-pilot" and the UI as a simple on-ramp [9]. Some assumptions and their risks:

- The algorithm will run automatically at the end of the wizard (no "Generate" button) and produce a decent grouping [12]. This assumes the algorithm is fast and reliable. If it's slow on large rosters, auto-running could create a lag before the Workspace loads. If it sometimes fails or gives clearly imbalanced groups, the user's first impression could be negative. Previously, having an explicit "Generate" step might have prepared the user for a wait or a result; now it's implied. Technically, the team should ensure a loading indicator or some feedback when auto-generation is happening, so the app doesn't appear to hang.

- **“Smart redirect” on activity open** is another automated behavior: sending teachers to Setup or Workspace based on whether groups exist [18]. This default is smart in theory (it guesses the teacher’s intent), but it could confuse if the guess is wrong. For instance, consider an activity that *has* groups (draft or published): the auto-redirect will always take the teacher to the Workspace. What if the teacher actually wanted to review the roster or preferences (i.e. go to Setup) first? The system doesn’t currently know that intent. So the teacher will have to manually navigate to Setup from there. Not a huge problem, but it’s the kind of minor friction that comes from the system making decisions. The plan’s alternative of a hub page with buttons would put the decision back in the user’s hands at the cost of one more click [18]. The chosen approach optimizes for the common case (most likely the teacher wants to edit groups if they exist), but the assumption that this redirect logic covers all needs is something to watch. It might be wise to allow quick navigation between Workspace/ Setup within the activity UI to compensate (e.g. a small tab or menu to switch, since the hub is gone).

- The “smart defaults” philosophy extends to **group generation settings**. For example, when a teacher doesn’t manually specify group names or counts (choosing “auto split” mode), the system likely picks a number of groups or group size automatically. The StepGroupsUnified code in the current system already tried to pick a reasonable default number of groups (like based on class size and a heuristic) [47]. The plan keeps this behavior. The risky assumption is that these auto-decisions will satisfy teachers. If a teacher disagrees (e.g. “I actually wanted 5 smaller groups, not 4 big groups, but I didn’t specify so the system guessed wrong”), they might have to redo some steps. The plan suggests the new StepGroups will let the teacher either name groups or let the system split – so presumably the teacher still has an opportunity to influence group count if they care [48]. As long as that UI is clear (perhaps a choice “Number of groups vs. group size” or a slider), the smart default can be overridden. If not, then the default might not always match the teacher’s expectation, which could be frustrating if they don’t know how to change it.

- Another reliance is that hiding complexity won’t hide capability entirely. For instance, algorithm configuration is gone from the UI, but what if a certain scenario truly needs a different approach (say a purely random assignment vs. balanced)? The plan is effectively saying: trust the system’s balanced algorithm, and if you want something else (random), you might have to manipulate inputs or wait for future features. That is acceptable for an MVP so long as the default algorithm performs robustly across scenarios.

In summary, the plan’s assumptions about smart defaults are generally in line with modern UX trends (reduce user decisions, do sensible things automatically). The risk comes if those defaults fail even a minority of use cases without an obvious remedy. The good news is the plan does not eliminate the *possibility* of doing things differently – it just makes them hidden. As evaluators, we should ensure the implementation includes cues or paths for when the defaults aren’t good enough. For example, the plan mentions advanced algorithm settings collapsible on Setup [49] – if that includes things like “Prefer smaller groups” or toggling algorithm modes, that could address edge cases. If it’s left unimplemented, then the app might be inflexible.

Inconsistencies to call out: the plan’s **qualitative success criteria** includes “Power features exist but don’t clutter main flow” [50]. This assumes those features indeed exist in some form after the redesign. It’s a reminder that the implementation must not accidentally drop a feature without relocating it. Any gap between what was removed and what was reintroduced elsewhere is an inconsistency that could bite. For example, if the team removes the Preferences step but then *doesn’t* finish building the PreferencesSection on Setup in time, the feature is effectively gone – that would be a critical failure. Therefore, each removal should be matched with either an alternative solution or a conscious decision that it's truly not needed. The plan mostly does this mapping (as listed in the removed routes mapping [21]), but a few need clarifications (roster reuse, student detail, etc., as discussed).

3. Do the Proposed Flows Cover All User Needs?

The plan outlines four main flows: **First-Time Teacher**, **Returning Teacher**, **Editing Changes**, and **Presenting to Class** [51] [5] [52] [53]. We need to compare these with the current repository's functionality to see if all use cases are accounted for.

- **Flow 1: First-Time Teacher – "Landing → Create Activity → Wizard (3 steps) → Workspace → Present"** [51] **.** This covers the journey of a new teacher creating their first grouping activity from scratch and then showing it to the class. It addresses core needs: importing a roster, configuring groups (at a basic level), and getting groups generated, then presenting them. The **current repo** supports all these tasks: roster import (via CSV paste or Sheets) [54], group generation with a default algorithm [28], and a student view for presentation [31]. The new flow simplifies how these are done (fewer steps as discussed). One thing to verify is **roster import options** – the plan's Step 1 for the wizard should let a first-timer paste a roster or connect Google Sheets easily, which it indicates it will [30]. The flow omits explicitly any step for preferences or fine-tuning groups, which as noted could be needed by some first-time users if they have that data. However, since this is "first-time," we can assume those users likely try the simplest case first (the plan assumes they won't immediately use advanced options). Another need is **naming the activity**, which the flow omits (auto-naming instead). A new user might not think to name it anyway, so auto-naming helps them past that hurdle – but as mentioned, the ability to rename later should be available or they might end up with a default name they don't like. After the wizard, the flow takes them to **Workspace (editing)** then **Present**. The current repo allowed editing the groups via drag-and-drop and undo/redo on the "Activity detail" page [55] – this is preserved and made the central workspace. Presenting uses the student view route, which is now the Present page. So for a first-time teacher, it looks like all fundamental needs (create groups from a list of students, adjust if needed, and display to class) are covered in Approach C.

The question is: are there any needs a first-time user might have that this flow doesn't mention? One possibility: **printing or exporting the groups** for offline use. The plan does mention export being moved to a "..." menu [41], so presumably the functionality remains. A first-time teacher after presenting might want to print the groups or share them digitally – that should still be possible (the plan's references and domain docs note print/Google Classroom integration as required [56]). Though not explicitly in the flow description, it's likely covered by the existence of the export option. Another need: what if the teacher made a mistake in the roster (like omitted a student)? In the current system, they could go back or edit the roster possibly. In the new design, after wizard the teacher would go to Setup page to add the missing student (roster management) [13] and regenerate. The flow doesn't describe this scenario, but the architecture does allow it. It will rely on the teacher discovering the Setup page if needed, which isn't in the linear flow. Perhaps the UI will have hints (maybe an "Activities" back arrow that leads to the activity hub or directly to Setup). This is a slight gap in the described flow – error correction or iteration isn't spelled out for first-timers, but presumably the Setup page and regenerate enable it.

- **Flow 2: Returning Teacher – "Dashboard → [Activity Card] → Workspace (or Setup if needs changes)"** [5] **.** This flow addresses users who have used the system before and have existing activities. The dashboard now lists "Your Activities" (instead of the old /groups list) and each card gives status and an action [16] . The plan meets returning users' needs to quickly jump back into a project. If an activity is already in progress or has draft groups, the card's primary action is "Edit Groups" (taking them to the Workspace) [16] . If it's published or finalized, it might show "View" (likely leading to Present mode) [57] – though the plan text suggests "View," they might mean view as in present or a read-only state. In either case, a returning teacher can get where they need.

The conditional "(or Setup if needs changes)" implies that if something is incomplete, they'd go to Setup. It aligns with the smart redirect logic [18] : an activity with no groups yet would send them to Setup by default. One scenario: a returning teacher might start a new activity (for a new class or new semester). The flow for that would be similar to first-time (click "+ New Activity" on dashboard, do wizard). The plan doesn't explicitly list "Create another new activity" in flows, but it's presumably the same wizard flow again. That should cover that need.

Another common need for returning users: **managing or updating an existing activity's roster or preferences over time.** The plan introduces the Setup page specifically to handle configuration changes for an activity (like adding students mid-semester, adjusting group counts, importing new preference data, etc.) [13] [25] . So if a teacher returns and some conditions changed (new students joined the class, etc.), they can go to Setup, update, and regenerate. The flow says the card might lead directly to Workspace unless "needs changes." It's not clear how the system will decide that – possibly if the activity is currently published or if the teacher explicitly navigates to Setup. The flow could be interpreted that most returning sessions go straight to editing, and only if something is amiss do they go to setup. It might rely on the teacher's own judgment to visit Setup if they want to make changes to config. The plan's alternative hub page (if implemented) could allow the teacher to choose to go to Setup instead of workspace. If it's auto-redirect, a teacher might have to navigate to Setup manually (maybe via the "Activities" list or a small link). This is a slight usability consideration: returning users might not always want to land in editing mode. For example, if a teacher knows they have to add 5 new students this semester *before* editing groups, an automatic jump to Workspace means they'll then have to click to Setup anyway. It's not a dealbreaker – just a nuance that the linear logic might not predict perfectly.

Aside from that, returning users' needs like seeing all their past activities at a glance (the dashboard provides that) and knowing the status of each (provided by the status indicators on cards [58]) are addressed. The plan even simplifies terminology on the dashboard (calling them Activities instead of "Groups" which was the old route label). One detail: if the user had many activities, does the dashboard paginate or group them? The mockup shows two cards side by side [59] , implying a grid of cards. That likely is fine.

Also, the plan acknowledges *published vs editing* states, which covers the need to distinguish between something currently being worked on and something completed. They provide "● Published" vs "○ Editing" status icons on cards [60] . This is good for returning users to orient themselves.

In terms of current repo functionality: currently, after finishing the wizard, the scenario (groups) is stored in IndexedDB and listed on the /groups page as an activity [61]. So the dashboard concept exists and is being refined. The new flows cover the same listing but with improved info and actions, so nothing is lost there.

- **Flow 3: Teacher Making Changes – "Workspace → (drag students) → (auto-saves) → Present when ready"** [52] **.** This flow corresponds to the iterative editing process. Based on the repo, once a scenario is created, teachers can manually drag-and-drop students between groups to fine-tune the result [55]. Undo/redo is available for that session [55]. The plan's Workspace design is a "pure editing interface" with drag-drop, and it explicitly mentions that changes auto-save continuously [52]. This aligns with current behavior (auto-save with debounce to local storage) [62]. So the basic need of adjusting groups after initial generation is well covered.

Are all editing-related needs handled? Teachers might want to perform actions like swapping two students, moving many students at once, or clearing a group. Drag-and-drop covers single moves; hopefully, the UI still allows multi-select or other shortcuts if needed (not mentioned, so likely not a new addition – but the MVP might not have multi-select anyway). The removal of some toolbar clutter (like a dedicated "Students" list button) means that to add a brand new student, one can't do it from the Workspace directly now. In the current UI, was that possible? Perhaps the "Students" header action navigated to the students sub-page or opened a list of all students. Now, a teacher in the Workspace who realizes they need to add someone must navigate to Setup. This is a slight inconvenience but arguably not a common frequent need during grouping edits – adding students tends to be a setup task. As long as the teacher knows to go to Setup (perhaps via the main Activities page or a link), it's okay.

The flow ends with "Present when ready," implying that after making changes, the teacher will hit the "Show to Class" button to go into Presentation mode [17]. One user need here is **publishing** the final groups (notifying students or marking the grouping as final). The plan distinguishes editing vs published states, but it doesn't describe an explicit "Publish" action in the flow steps. However, it does mention in the principles "Continuous autosave with explicit publishing" [63], meaning they intend a clear action to publish to students. In the Workspace design, the "Show to Class" might just be a presentation view, not the actual publish (like a projector mode). It's possible that publishing is automatic when you present, or separate. Actually, the plan's references to "Sessions" (published history) suggest there is a concept of publishing beyond just showing on screen. Perhaps "Show to Class" implicitly publishes or the act of hitting "Done Presenting" might prompt to publish changes. The plan isn't explicit here, which is an area needing clarification. In the domain, a Scenario has a status like DRAFT or ADOPTED [64]; publishing might mark it as adopted. If currently the system automatically considered the scenario active once created, the plan might introduce a manual publish step. Since "Publish to Students" was emphasized as a separate deliberate action in the UX Strategy [65], the new design likely has a prominent Publish button somewhere (maybe on Workspace toolbar or on the Setup page). But in the flows, they skipped mentioning it, possibly focusing only on the UI flows. We should ensure the final design covers that – teachers need to know if the groups are just in-progress or if they have been finalized for the class. If "Show to Class" is merely a presentation and not record-keeping, there should be a separate publish action (which could tie into sending notifications or locking the grouping).

All considered, the editing flow covers the act of modifying groups and proceeding to present, which is the core iterative cycle teachers will do. Advanced editing needs like undo (still present), and "Regenerate" if they want to try a different automatic distribution (still in the toolbar as a dropdown [66]), are intact. The plan even simplifies regenerate by putting algorithm options under a dropdown on the toolbar [66] –

presumably in the future that's where multiple candidate algorithms can appear. So iterative human+algorithm interplay seems supported: the teacher can drag manually or click regenerate for a new suggestion, repeatedly, which covers a broad range of editing strategies.

- **Flow 4: Presenting to Class – "Workspace → 'Show to Class' → Present mode → 'Done' → Back to Workspace"** [53] **.** This flow addresses the need to actually use the group results in the classroom setting. The present mode, as described, is a clean, read-only view where all groups are displayed for projection, including a way for a student to search their name to find their group [67] [68] . This corresponds to the current *student-view* feature which was already implemented (read-only view with search) [55] . The plan doesn't remove this capability; it actually integrates it as a first-class "Present" route. So that need is fully covered, arguably improved by making it part of the main flow (the button "Show to Class" is right in the header) [69] – no need to navigate to a separate URL manually as perhaps before.

After hitting "Done Presenting," it returns to the Workspace [67] . A teacher's need here is to perhaps make further adjustments after showing (maybe a student complains and they decide to swap on the fly). The flow supports that by returning to editing seamlessly. Good.

One user need around presentation could be printing a hard copy for posting or for records. The plan's Present mode description is all digital, but printing might just be handled by browser print styles or the export function. The original system required a print/export feature (noted as required in design questions) [56] . As long as that still exists (maybe via the "… menu" on workspace or an export on the present page), it's fine.

Another need: after presenting, maybe **publishing** the results officially so students can be notified or view them later. If "Done Presenting" just goes back, the teacher might wonder "did this save? Are students informed?" Since there's no student portal yet (it's a planned feature) [70] , publishing might simply mean the teacher saying "these are final." The plan's Phase 1 in UX Strategy did mention adding a "Publish to Students" concept [65] . It's possible that by Phase 4 (Setup Page creation), the "History (published sessions)" section indicates that when you hit Publish, an entry goes there [25] . The plan however doesn't detail the UI for publish in this Approach C doc – a slight oversight. It might assume publishing happens either automatically at some point or via a button not described. We should highlight that: to fully cover user needs, if notifying students or marking final groupings is needed, the UI should include a publish step (even if it's just conceptual given no auth in MVP, maybe it's just for teacher's own tracking).

Summing up the flows vs current functionality, **most user needs are addressed**: creating, editing, viewing groups, adjusting inputs, iterating, and presenting. The plan actually covers continuous management better by introducing the Setup page for configuration changes (something the original wizard paradigm made clunky). The continuous usage (semester-long updates) is implicitly supported by the teacher returning to Setup/Workspace as needed, which aligns with real workflows [71] [72] .

What might *not* be fully covered or could be edge needs?

- **Advanced grouping scenarios:** For example, if a teacher wanted to run multiple distinct groupings for the same set of students (say one set of groups for a project in October and a different set in November), the current MVP didn't fully support multiple scenarios per activity (it would probably suggest making a new activity or reusing the same and losing the old grouping). The plan doesn't explicitly introduce multi-scenario support aside from the History log of published sessions. So a teacher need like "keep past groupings and start a new one with the same roster" is partially covered by duplicating the activity or using the History as reference. Approach C doesn't add a direct "duplicate activity" or "new session" button (beyond listing history). Teachers might handle this by creating a new Activity each time if they want parallel scenarios. That's fine, but it's something to clarify in guidance – perhaps each semester or each major regrouping is a separate Activity entry.

- **Collaboration or sharing:** Not really in scope here – multi-teacher collaboration is a future concern [73]. The flows assume a single teacher user, which is consistent with current functionality (no user accounts or sharing yet).

- **Deleting or archiving an activity:** The plan doesn't mention it, but presumably the dashboard should allow deleting an old activity. Current status shows activities can be archived (domain has `status: 'ACTIVE'|'ARCHIVED'` for a Program) [74] [75]. The plan's cleanup might expect an archive or delete in the "... menu" on an activity card or such. If not implemented, the user might accumulate clutter. It's not discussed, so that's an area of additional implementation if not already present.

- **Edge preference cases:** If a teacher had "never pair these two students" type of rule, the MVP doesn't have a UI for that yet [11] and Approach C doesn't introduce it either (they rely on manual adjustment). So that need remains unmet, but it's known as a planned future feature outside this UX overhaul.

To check if flows cover repo functionality: According to the STATUS.md, every MVP feature is accounted for in the new flows, just relocated: - Wizard (create groups) – still there but shorter [76]. - CSV roster import – still in StepStudents [54]. - Preference import – moved from wizard to Setup (still supported, just not in initial flow) [77]. - Group configuration (specific vs auto) – still in the new StepGroups (the new StepGroupsSimple likely retains both modes) [77]. - Balanced algorithm – still the default grouping method [28]. - Basic analytics – still present (though panel removed, summary remains) [28] [78]. - Drag-drop editing – unchanged in core, just now on the primary workspace page [55]. - Undo/redo – still in the toolbar (we see [Undo]/[Redo] in the mockup) [43]. - Read-only student view – now the Present page (functionality remains same) [31]. - Roster reuse – needs to be integrated into new wizard Step 1 (the plan implies it will) [36]. - Auto-save – continues (no save button introduced, explicitly kept) [63]. - Activity dashboard – improved and renamed (still listing activities) [79]. - Browser persistence (IndexedDB) – likely unaffected by UI changes, still store scenarios locally [79].

Additionally, **Known Gaps** from status: - "Teacher gets one option (no candidate gallery)" [11] – Approach C initially still has this gap but plans to fix it in Phase 2 with inline options. So they're aware and covering it in near-term phases. - "No conflict rules UI" – still none in new flows, but not a regression (just not implemented yet). - "No authentication" – unchanged, not directly a UX flow issue for now (just means each teacher's data is local). - "Mobile untested" – the new design should in theory be simpler on mobile (less modals, more single pages), but the plan admits Phase 3 (split-pane) might be challenging on mobile [80]. As it stands in Approach C initial phases, nothing explicitly addresses mobile, but since target is laptop for now, it's not a user need being tackled yet.

In conclusion, the proposed flows do map to all primary user tasks provided by the current system, often streamlining them. The only caveats are: - Some advanced or less common tasks are moved out of the main line (but still possible, e.g. import preferences after creation, reuse roster via new UI, view history in setup). - A few implicit actions like publishing final results are not detailed and should be clarified in implementation so that user's end goals (like informing students or marking completion) are met. - The flows assume ideal behavior; error handling (like invalid CSV, etc.) isn't mentioned but presumably will be handled similarly to current (with warnings as before [54]).

Overall, the flows seem to cover the user needs identified in the current functionality, with improvements for orientation and continuous use. Any missing steps (like roster reuse or renaming activities) are likely just not explicitly written in the plan, but they need to be addressed when building to avoid regressions.

4. Potential Usability and Technical Risks

Implementing this ambitious UX overhaul introduces several potential risks:

Oversimplified Workflows: While simplicity is the goal, there's a danger of **overshooting and oversimplifying**. One example is the 3-step wizard. By eliminating steps like naming and preferences entirely, the initial workflow becomes super streamlined – but possibly *too* bare-bones for some scenarios. Usability testing is needed to ensure new users don't feel something is missing. If a teacher expects to provide certain info during setup and the system never asks, that could cause doubt ("Did I miss a step?"). The plan's success criteria tout reduction in steps and controls as a victory in itself [33]. However, fewer steps isn't inherently better if those steps were carrying important context. For instance, not asking for an activity name might confuse a teacher when they later see an auto-generated name on the dashboard they don't recognize. Similarly, not explicitly walking through preferences might mean some teachers never discover that feature at all (since it's not in the flow, and only tucked away later).

There's also a risk that combining multiple setup tasks on one page (roster, group config, prefs, history) – while technically one "screen" – could actually introduce complexity in another form (scanning and toggling sections). If not carefully designed, the teacher might be unsure what to do first on that page. The principle of one task per screen needs reinforcing via UI cues (section headers like "1. Choose roster → 2. Configure groups → (optional) 3. Import preferences"). If those are absent, the user might perceive the page as a lot of forms.

Loss of Advanced Control / Power-User Frustration: By demoting advanced features, the plan risks alienating some experienced users or those who *do* need those features. For example, a teacher with very specific grouping requirements may find the lack of an upfront preferences step or algorithm settings irritating – they have to go hunting for those options. If power features are *too hidden*, they may be effectively undiscoverable, which equates to lost functionality. The plan's philosophy is correct that new users want sane defaults, but consider providing *some* route for advanced usage. Perhaps an "Advanced Setup" mode toggle, or even documentation that explains "If you want to do X, go to Setup page and expand Preferences/Advanced." Otherwise, there's a risk that a minority of users will feel the tool is oversimplified or not suitable for complex cases, even if under the hood it still can do those things.

Technically, removing these features means less code to maintain in the short run, but if user feedback demands their return, it could be costly to reintroduce. For instance, if after release many ask "How do I see alternative groupings?" the team might have to prioritize building the candidate gallery sooner or at least provide a quick fix (like an experimental feature flag to show multiple candidates). The plan did schedule Candidate Gallery as next, so that risk is being mitigated proactively.

Ambiguous Redirects and Navigation Flow: The “smart redirect” for `/activities/[id]` is potentially confusing as noted. From a technical standpoint, implementing it means the app’s routing logic must check the state of the activity (does it have groups?) on load and then navigate. In SvelteKit (if that’s the framework given the `+page.svelte` structure), this likely involves a `load` function that fetches the scenario and then issues a redirect. A risk is handling edge cases: what if the data isn’t loaded yet when deciding? Or what if a user somehow navigates quickly and triggers multiple redirects? These are probably minor. The bigger issue is user perception: hitting a URL and ending up somewhere else without an explanation. If I bookmark an activity link, I might always end up on Workspace (once I’ve generated groups). That’s fine if that’s what I want, but if I expected a landing page, I may think the link is broken or changed. Given that the app is moving away from the old structure, maybe no user bookmarks exist (the plan even says no need for old route redirects because presumably it’s a new release) [40]. But still, within the app, a teacher might not understand the distinction between the hub and workspace and wonder “why do I go here now?” To mitigate this, the UI could briefly flash a message (“Redirecting to Workspace…”) or, as suggested earlier, perhaps not automatically redirect at all but present a small choice. This is a UX nuance – either approach can work, but the chosen path should be tested for comprehension.

Another navigation risk is that by removing some routes, the ways to reach certain functions become fewer. For instance, previously there was a route dedicated to editing sessions or viewing templates. Now there isn’t – so if a teacher doesn’t go through the one remaining path (Setup page), they won’t find those. It puts more pressure on the new pages to be well signposted. If the Setup page is hidden behind a redirect (like you normally skip it unless you need it), how does a teacher reach it on their own? Possibly from the dashboard (maybe an overflow menu on the activity card: “Setup”), or by knowing the URL. This needs to be clarified in the UI, otherwise the linear flow might *overconstrain* navigation.

Technical Complexity and Risks in Implementation: The plan outlines a multi-phase implementation [81] [82], which helps break it down. However, each phase has its challenges:

- *Phase 1 (Route restructure)*: Creating new routes and adding redirects must be done carefully to not break functionality. Initially, the new pages like `/activities/new` might just mirror the old `/groups/new` wizard until Phase 2 redesigns it. During the transition, there’s risk of bugs where a redirect loop could occur or an old component still references a now-removed route. The plan to update all internal links in Phase 5 [83] is crucial – missing any link could cause a runtime navigation error. Also, because data is stored in IndexedDB keyed likely by activity IDs, the route change shouldn’t affect it – but any code that assumed a certain URL structure might need updates. For example, if any deep link or context assumed “/groups/[id]” for something, it must be revised.

- *Phase 2 (Wizard simplification)*: This involves major surgery on the wizard components. Removing several Svelte components (StepName, StepPreferences, etc.) [84] and creating a new StepGroupsSimple means a lot of testing and QA. There’s risk of new bugs (e.g. combining group naming and preference could introduce issues if not done right). And any logic in the removed steps needs to move somewhere: e.g. StepPreferences did parsing and validation of CSV preferences [37] – that logic should likely move to the PreferencesSection of Setup. If the team simply deletes it in Phase 2 and waits to add PreferencesSection in Phase 4, then between those phases the app temporarily loses preference import entirely. That’s a big risk if the deployment happens mid-phase.

They might avoid deploying until all phases are done, or need feature flags to hide incomplete features. The plan does mention each phase should deliver working software and you can pause at a phase boundary [85]. To honor that, Phase 2 deliverable should include at least a minimal way to import preferences (or explicitly decide to temporarily drop it). If they skip it until Phase 4, any user needing preferences in that interim would be blocked. So it's safer to implement the PreferencesSection earlier or not remove StepPreferences until the new path exists. Such sequencing issues are a technical project risk.

- *Phase 3 (Workspace cleanup)*: Removing UI elements like the AnalyticsPanel and HistorySelector means ensuring those functions are either not critical or have alternatives already in place. From a code perspective, ripping those out could have side effects. For example, if the AnalyticsPanel also housed some computation logic or triggered updates, the devs must ensure the inline summary still gets computed. If the summary relies on existing analytics code, that's fine, but if the panel was doing it, they need to move it elsewhere. Likewise, HistorySelector removal means ensuring the undo/redo still works properly and the code that might have managed multiple scenarios is adjusted. These are manageable but must be tested (e.g. after these removals, test that undo/redo still functions and that publishing a new session still logs properly to the history).

- *Phase 4 (Setup page creation)*: This phase is essentially building a new, fairly complex page from scratch with multiple subcomponents [13] [86]. Technical risks here include integrating with existing domain logic (e.g. adding students to a roster/pool, which likely already exists in the repository's domain layer, but now needing a UI to do it). Also, this page will need to coordinate data across sections (for example, if you add a student in Roster section, that changes the student count which might affect group config or algorithm outcomes). Ensuring reactive updates and not introducing state bugs is key. The plan calls for algorithm options collapsible – depending on the complexity, that could be trivial (maybe just a placeholder for now since only one algorithm exists). Templates inline means the Setup page must list available group templates (if the teacher has any pre-saved configurations). That requires pulling data (maybe templates are stored in IndexedDB or generated from history). If templates were previously managed on a separate page, moving them into a section might be straightforward or not, but it's new UI that must be intuitive (e.g. a dropdown "Choose a template for groups"). A risk is making sure the teacher understands how templates differ from current groups – not to confuse them with the actual groups of the activity. This might need UI cues like "Apply a saved group structure to this activity."

- *Data migration / backward compatibility:* The plan explicitly states that removing routes doesn't require supporting old bookmarks [40]. This implies either there aren't external users yet or it's acceptable to break URLs. If there are any existing users with saved links (to e.g. a student view or directly to /groups/[id]/edit), they'll get a 404 unless redirect rules are in place. The plan does say they will add redirects from old to new in Phase 1 [83]. So technically they will likely map `/groups` to `/activities`, `/groups/new` to `/activities/new`, etc. But interestingly, they said no redirects needed for old bookmarks (perhaps they assume all usage is via the app UI which they are updating in lockstep) [40]. I would still implement basic redirects to be safe, because even during development or testing someone might hit an old URL. Not doing so is an avoidable risk.

Usability of the New Design: Beyond the removal of features, the new design itself must prove intuitive. Potential pitfalls: - The **Setup vs Workspace** distinction might confuse some users initially. They might wonder why there are two separate pages for an activity and when to use each. The plan's approach is to hide that complexity (auto-redirect or have clear primary actions). If someone ends up on Setup page after creation (in case no groups yet) and then generates groups, they go to Workspace and maybe never return to Setup. Later, if they need to import new preferences, they have to know to go back to Setup. Without an explicit menu, how will they navigate? Possibly via the Activities list or a link. If that's not obvious, they might try to find a preferences option in the Workspace (which won't exist). So, a UX risk is that the concept of "two modes: setup vs editing" might not be crystal clear. Some subtle indicator or help text might be needed (e.g. on Workspace page, a label or link like "Need to change roster or settings? Go to Setup"). The plan's smart hub page (if they did minimal page instead) could also solve it by offering a choice when re-entering an activity.

- The **auto-naming** of activities, as mentioned, could lead to confusion if multiple "Untitled" or similar names appear on the dashboard. Technically, if the code uses roster name for activity, and if a teacher always pastes rosters without a name, what do they use? Maybe the first few characters of the student list? That is not ideal. This is a usability issue because the dashboard might show cryptic names. Additional work is likely needed: perhaps use a timestamp or prompt the user at some point to name it (maybe on the Dashboard itself via inline edit). If that's not handled, it could reduce the usefulness of the dashboard (imagine two "New Activity" entries – which is which?).

- **Reliance on teacher to intuit hidden functions:** e.g. that clicking the three-dot menu will reveal export/print. If the icon isn't obvious or there's no label, some may miss it. Minor issue, but important for teachers who might not be tech-savvy. They might literally ask "How do I print this?" if the button is tucked away. So ensuring the menu is visible and perhaps documented in a tooltip is important.

Technical performance risks: Not explicitly mentioned, but worth considering: by consolidating into larger pages (especially Setup which shows roster lists, etc.), the initial load might be heavier. If the roster is large (say 200 students) and preferences matrix is big, loading all that on one Setup page could be slow or laggy in the UI. The original wizard split things across steps partly to chunk work. Now one page might handle multiple data sets. The developers should lazy-load sections (e.g. don't render the Preferences section unless expanded) to mitigate this. Also, by using Svelte stores or contexts, they need to ensure the data flow from Setup to Workspace is smooth (e.g. after generating, Workspace has the groups; if they go back to Setup, it should reflect updated rosters etc.). Keeping state in sync can get tricky – but likely they will treat the activity's data as source of truth, which should be fine.

Testing and Quality: The repository had E2E tests covering the full wizard flow [87]. Those will all break once the flow changes. They need to rewrite tests for the new 3-step wizard and new routes. There's a risk that not all edge cases are caught after refactor. Investing in updating automated tests is crucial to catch things like "importing a malformed CSV shows a warning" – that logic should still work in the new Setup preferences section, for example.

Finally, consider **transition for existing users** (if any): A teacher who used the old version might be surprised by the new UI. There's a change management aspect. Even if the user base is small, a quick tour or pop-up "What's new" could help them not get lost. For instance, if someone was used to clicking "Edit" and now there is no separate edit route because it's always inline, they might initially think something's missing. However, since the plan is likely part of a major revamp, perhaps users expect big changes.

In summary, the main risks lie in *hiding complexity* (possible discoverability issues) and *ensuring new navigation logic is clear*. Technically, the plan is feasible but requires careful sequencing to not temporarily break functionality. If executed phase-wise with thorough testing, these risks can be mitigated. The key is not to assume that "simpler is always better" – sometimes removing a feature can remove a safety net or a capability some relied on. Being ready to adjust (maybe bring back a small hint or option if needed) will be important once real teachers interact with this redesign.

5. Trade-Offs in the Proposal and Suggestions

The Approach C plan makes several deliberate trade-offs between simplicity and flexibility. Let's justify those decisions and consider if there are better alternatives or mitigations:

Trade-off: Fewer routes & pages vs. Dedicated feature screens.
By cutting the route count from 14 to 9 and merging functionalities, the team clearly prioritized *simpler navigation* over isolated workflows. The justification is that teachers will have a more streamlined experience – no more jumping to a separate "Edit mode" page or a "Templates" page; everything is either done in the main flow or on the Setup workspace [88] [21]. This reduces context-switching, which is good. The downside is potential UI bloat on the remaining pages and possibly burying some features.

Suggestion: Introduce gentle guidance within those consolidated pages. For example, on the new **Setup page**, include context-sensitive help text or wizards for advanced tasks. When a teacher expands the Preferences section, a one-line note like "Import student group requests (optional, for fine-tuning the algorithm)" can remind them of the feature's purpose, since it no longer has a whole step to explain itself. Similarly, the History section could note "Here you can review past published group sets." These micro-explanations can compensate for the loss of dedicated pages that might have had more instructions or space.

Another suggestion is to consider keeping a lightweight **Activity Hub page** instead of an immediate redirect. The plan mentioned an alternative where `/activities/[id]` shows minimal info with buttons [18]. Even if you auto-redirect by default, you might implement that page and use it if a user navigates intentionally (say via a menu). It could serve as a central "settings" page. This would address navigation flexibility without re-introducing complexity: advanced users could access it to jump between Setup/Workspace/Present quickly, whereas novices following the main path get redirected automatically. Essentially, it'd be a hidden option for those who need it. If doing so, make sure it's not confusing – it could simply be the *same* Setup page but with an overview at top. This is an optional enhancement to help orient users who, for example, click on an Activity name expecting a summary.

Trade-off: 3-Step Wizard speed vs. completeness.
The proposal sacrifices some completeness in the initial wizard (no explicit naming, preferences, candidate choices) to get teachers to results faster [38]. The logic is that a teacher can create their first activity in under 5 minutes, with "no 'what do I do next?' moments" [50]. This is a valid goal since teacher drop-off in a long setup could be high. The removal of friction (like not forcing them to come up with a name or find a CSV of preferences mid-flow) likely makes the first-time experience smoother. The trade-off is that some teachers might have wanted to do those things.

Suggestion: Provide **post-creation prompts or cues** as needed. For instance, after the wizard auto-generates groups and lands the user in the Workspace, a small banner could say: "Groups generated! You can now adjust them or add student preferences for finer results." and maybe a button "Add Preferences" that jumps to Setup's preference section. This way, you still don't bog down the initial creation with extra steps, but you catch those who have more to do. It's like offering an on-ramp to advanced features only if the user is ready. Another idea is to allow an **"Advanced" toggle in the wizard**: perhaps on the first step or somewhere, a little link that says "Need to set advanced options (like student choices)? You can do that later in Setup." This at least communicates that the capability isn't gone, just postponed.

For the **naming issue**, an alternative approach short of a full step could be: auto-generate a name but show it on the dashboard immediately with an edit icon. If the teacher doesn't like "Activity 1" or "New Clubs – Jan 2026" (whatever default), they can click and rename right on the dashboard card. This would be a relatively simple addition and would resolve potential confusion. It's a trade-off (the plan tried to avoid asking for a name at all), but giving the option unobtrusively later is a fair compromise. The code would need to support updating an activity name (which likely it does, since StepName was doing it previously). This is an example of preserving flexibility without reintroducing a whole step.

Trade-off: Hiding power features vs. interface clarity.
The plan chooses to hide or remove many advanced controls (templates, algorithm settings, detailed analytics) for a cleaner UI. This trade-off favors new or casual users at the expense of power users. The justification given is that power features *"don't clutter main flow"* but still "exist" [50]. So, the expectation is that advanced users can find them if needed, just not front-and-center.

Suggestion: Implement a clear **"Advanced Options" affordance** where appropriate. For example, on the Setup page, the Algorithm settings could be behind an expandable panel labeled in plain words, maybe "Advanced Settings (Optional)". On the Workspace, since most extras were removed, consider a keyboard shortcut or a profile setting that advanced users can use to re-enable some power tools (for instance, a debug key to show more metrics or candidate comparisons). This doesn't have to be advertised in the UI for everyone, but documentation or a help center could mention it for those who care. Alternatively, since teachers are not likely to read docs thoroughly, perhaps a **power-user mode toggle** in a settings menu (if any exists in future). This is more speculative – currently no user account settings exist to store such preferences, given no authentication. So maybe not feasible now. In lieu of that, just ensure that the "hidden" features are not *too hidden*. Collapsible sections should have clear headings and maybe a brief description, so that a teacher exploring the interface realizes "oh, I can also import preferences or tweak algorithm here if I want". The plan does intend to label those sections (Roster, Groups, Preferences, History on Setup) [13], which is good.

Trade-off: Immediately removing old components vs. phasing out gradually.
From an implementation perspective, the plan is quite aggressive about deleting old code (routes and components) by the end [82]. This is great for maintainability if the new approach works flawlessly, but it leaves little fallback. One could consider keeping some of the old flows accessible for a transition period (even if hidden), just in case something in the new flow fails. However, given the context (likely an internal or beta stage product), it might be acceptable to do a hard cut-over.

Suggestion: If the user base is live and depending on the tool, consider rolling out these changes in a feature-flagged way. For example, release the new routes but keep the old ones working via redirects for a version; maybe allow a power user to access the old Preferences step through a direct link if absolutely necessary (not publicly advertised). This could cover any gaps if, say, the new Setup page's preference import has a bug at first – the old StepPreferences could still be accessible via some fallback. Once confidence is high, then remove it. This strategy reduces risk, though it means maintaining old code a bit longer. The plan's timeline (3-4 weeks for full implementation) [89] is short, suggesting they intend to revamp quickly and likely test thoroughly rather than maintain two systems. So this suggestion might not be needed if internal testing is robust.

Trade-off: Automatic decisions vs. user control.
We see this in auto-run algorithm, auto-redirect, auto-naming, etc. The benefit is fewer decisions for the user, the cost is potential misalignment with user intent. These are generally justifiable because teachers typically prefer not to fiddle with controls – they want it done. The team's user research likely supports these changes (the plan hints that *"teachers have no patience to learn new tools"* [90]). So erring on the side of automation is likely the correct call for the target audience.

One thing to justify: **No candidate gallery initially**. The trade-off was "reduce complexity now vs. provide optimal choice." The plan justifies it by saying the algorithm will pick a good option and teachers can always regenerate if unhappy [23] . This is reasonable given the timeline – a gallery is a complex UI to build, and providing five options might overwhelm a non-analytical user. The intermediate plan is to make the gallery a secondary feature later (which power users might appreciate) [40] . A suggestion around this: maybe incorporate a simple text or icon indicator in the Workspace like "Try another grouping" near the Regenerate button to hint that multiple options exist conceptually. It's a minor UX cue that could make up for the lack of an obvious multi-option view. The plan already has regenerate, so perhaps labeling it clearly ("Regenerate: Try a different grouping") would serve as that hint.

Trade-off: Removing "Edit mode" page vs. potential multi-user in future.
This is forward-looking: by embedding editing in the main page, they solved orientation issues [15] . It's a good trade – teachers stay in one context. It does raise a future consideration: if real-time collaboration (multiple teachers editing) is added, having everything on one page might cause more complex sync issues or conflicts. But the plan explicitly had a principle of "Multiplayer-ready architecture" [73] , so presumably the data layer was designed for that. They can stick with one-page and use operational transforms or similar in future. So that trade-off (no separate edit page) is fine and likely a permanent improvement.

Areas needing more work or clarification for viability:

- **Renaming Activities:** As mentioned, after removing StepName, the implementation needs to provide a way to change the activity name (even if it's just auto-named initially). This isn't described in the plan. In the repository, an Activity (Program) has a name field [7] . We should make sure the UI exposes it at some point. Perhaps an inline edit on the activity's Workspace header (the mockup shows "Spring Clubs" as a header in Workspace [91] – maybe clicking that could edit it, just as one can rename a document in many apps by clicking the title). That would be a neat solution that doesn't add an extra screen. If not that, then on the dashboard card's … menu "Rename". This needs design, but it's relatively small effort for significant user benefit. I would suggest it as a follow-up to ensure the auto-name isn't final.

- **Roster Reuse in New Wizard:** We've pointed this out, but to reiterate: make sure the new Step 1 has UI for choosing an existing roster. Implementation-wise, one could populate a dropdown with saved Pools from IndexedDB (the domain model supports multiple Pools) [92] . The current

StepSelectRoster code likely can be repurposed or merged. This detail will make the difference between returning users loving the new wizard ("I can still use my saved class list easily") or being annoyed ("Why do I have to paste the roster again? I had it saved before!"). So clarifying this in the plan or design is important for viability.

- **Publish Workflow:** As noted, clarify how an editing teacher publishes the final groups to mark a session complete. The plan hints at it (History of published sessions), but doesn't say what the teacher clicks to trigger a publish. Possibly it's an explicit "Publish to students" button on Workspace (maybe replaced by "Show to Class" in this context). The code currently might treat the scenario as implicitly active, but since they track history, they likely have a notion of finalizing. For the plan to be viable, the UI must surface that in a non-confusing way. Perhaps when clicking "Done Presenting," it could prompt "Do you want to mark these groups as published/final?" or automatically mark them. I would suggest the team make this explicit in the UI (like a distinct Publish button if possible), as it's a critical action (one that presumably notifies students in a fully built system). It aligns with the principle of explicit publishing [65]. If they skip implementing a clear publish action, the teacher might be unsure if just dragging students around actually took effect for the class or not.

- **Student Detail Access:** Even though /students/[id] route is gone, consider adding a minimal way to get info on a student if needed (e.g. maybe on hover or click in the Workspace show their full name, any preference data like "Requested: Art Club" if available). This would replace the need for a whole page. It's a "nice-to-have" that could save a teacher time if a student asks "Did you see my request? What did I put?" – the teacher could quickly check without digging out the CSV. The plan doesn't mention it, but it would enhance the "State is obvious" principle by making individual student's state in the grouping more obvious (like maybe highlight if they got their top choice, etc., which could be done via a small icon). These refinements are beyond the core plan but are alternatives to having had a separate route.

Justifying the trade-offs: The plan's rationale largely stands up: teachers benefit from a task-focused design with fewer distractions. The data from user research (implied by statements like *"teachers have no patience"* [90]) supports reducing complexity. Each trade-off seems to err on the side of simplicity and teacher-centered language/flow, which is likely correct for an education context where many users may not be tech experts and use the tool under time pressure. The removal of features like friend-of-friend preferences (not mentioned here but the STATUS said "Removed friend-based preferences, request-only mode" [93]) also indicates they're focusing on core use cases after learning what teachers actually used.

Alternative approaches considered: It's worth noting that the title says "Approach C" – implying there were Approach A and B proposals. We don't have those details, but Approach C likely was chosen because it balanced extremes (maybe Approach A was too minimal, Approach B too complex?). The chosen plan tries to make the UX *task-based*, which conceptually is strong. It frames everything around the teacher's tasks, not the system's internal features. This is a justified design direction because earlier the app was feature-based (Wizard vs Edit vs Analyze pages) and that mismatched the real workflow [94]. Approach C addresses that mismatch by making one continuous workflow. In that sense, the trade-offs made serve the larger goal of aligning the app with how teachers actually work over time (import -> group -> refine -> present, repeatedly).

One final suggestion: **communication**. When rolling out this redesign, accompany it with clear release notes or tooltips highlighting what's new and why. For example, if a teacher was used to a Preferences step, a note could say “We’ve simplified the setup – you can now add student preferences later if you want, rather than up front.” This manages expectations and educates users about the new way to do old tasks. A small guided tour on first launch of the new version could mitigate initial confusion and help justify the changes from the user’s perspective (“Oh, they made it simpler for me, and I can still do X if I need to.”).

In conclusion, the plan’s trade-offs skew towards simplicity and clarity, which is appropriate given the target audience and the problems identified (orientation loss, false completion, etc. ⑮). To make this plan fully viable, the development team should ensure: (1) no critical functionality is unintentionally lost (covering all “removed” features with new solutions), (2) the new flows are clearly communicated to users, and (3) any risky assumptions (like teachers not needing a certain feature often) are validated with actual usage once it’s live, with a willingness to tweak if needed. So far, Approach C looks well-thought-out to address the prior UX pain points, and with the above clarifications and minor additions, it should lead to a significantly improved user experience for Groupwheel.

Sources:

- Groupwheel UX Redesign Plan (Approach C) (95) ㉑ ㊳
- Groupwheel Domain & Status Documents ⑧ (96)
- Groupwheel UX Strategy Notes ⑮ (63)

---

① ② ③ ④ ⑤ ⑥ ⑨ ⑫ ⑬ ⑭ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉙ ㉚ ㉜ ㉝ ㊳ ㊵ ㊶ ㊷ ㊸ ㊻ ㊽ ㊾ ㊿ (51) (52) (53) (57) (58) (59) (60) (66) (67) (68) (69) (78) (81) (82) (83) (84) (85) (86) (88) (89) (90) (91) (95)
PLAN.md
file://file_000000003ef871f8875f6168d94497ac

⑦ ⑧ (64) (74) (75) (92) domain_model.md
https://github.com/andysmith26/groupwheel/blob/e99e0b23339aec9aacc5553aae2f526ec9b164fa/docs/domain_model.md

⑩ ⑪ ㉘ ㉛ ㊱ ㊴ ㊹ (54) (55) (61) (62) (70) (76) (77) (79) (87) (93) (96) STATUS.md
https://github.com/andysmith26/groupwheel/blob/e99e0b23339aec9aacc5553aae2f526ec9b164fa/docs/STATUS.md

⑮ ㊺ (56) (63) (65) (71) (72) (73) (80) (94) UX_STRATEGY.md
https://github.com/andysmith26/groupwheel/blob/e99e0b23339aec9aacc5553aae2f526ec9b164fa/docs/UX_STRATEGY.md

㉞ ㉟ ㊼ StepName.svelte
https://github.com/andysmith26/groupwheel/blob/e99e0b23339aec9aacc5553aae2f526ec9b164fa/src/lib/components/wizard/StepName.svelte

㊲ StepPreferences.svelte
https://github.com/andysmith26/groupwheel/blob/e99e0b23339aec9aacc5553aae2f526ec9b164fa/src/lib/components/wizard/StepPreferences.svelte