Paddock Setup Redesign

End-to-end solution proposal for setup, parsing, workspace creation, and live enrichment

Recommended direction: remove the confirmation form, treat the editor as the only recovery mechanism, replace user-facing format classification with internal schedule evidence, and redirect immediately to the course workspace where enrichment streams in live.

Prepared for	Date	Inputs reviewed
Paddock setup redesign	2026-04-20	v2 flow diagram, redesign handoff

UX outcome
No retry loop. The student always lands in the same editor and then in the workspace.

Architecture outcome
formatHint becomes internal schedule evidence; week rows become the setup source of truth.

Delivery outcome
Ship deterministic v1 distribution, feature-flag the flow, and stream enrichment through SSE with polling fallback.

Primary recommendation
Keep the parser cascade and the cached plan. Move all confidence levels into one side-by-side editor, remove domain, professor name, and format hint from student-owned input, add firstClassDate when real dates are missing, and send the student directly to /paddock/[courseId] where enrichment becomes visible in place.

Executive summary

The existing three-tier parser cascade is already strong enough for most syllabi. The main failures are downstream: a brittle formatHint, setup fields that should not be user-owned, a finalize step that can still trigger rebuild logic, and a redirect to a dashboard state that cannot show meaningful progress while enrichment runs.

Decision area	Recommendation	Why it wins	Order
Confirmation step	Remove the classification form	The editor becomes the only recovery mechanism, which eliminates the retry loop and the trust hit from a separate correction screen.	v1
Format signal	Make format evidence server-only	Replace user-facing formatHint with scheduleEvidence plus confidence. The UI should describe what was found, not ask the student to classify the syllabus.	v1
Setup payload	Own only editable setup inputs	Accept courseTitle, termLabel, meetingDays, firstClassDate, and weeks[]. Drop domain, professorName, and formatHint from the client payload.	v1
Calendar synthesis	Add firstClassDate when anchors are missing	Meeting days alone are not enough for a fully topic-based syllabus. One temporal anchor makes schedule generation deterministic.	v1
Redirect target	Go straight to /paddock/[courseId]	The workspace is the first place where the week plan is useful and enrichment can accumulate in place.	v1

Streaming	Use SSE with polling fallback	SSE matches the upload mental model and gives immediate visibility, while polling covers disconnects without backend complexity.	v1
Topic distribution	Use deterministic rules first	Preserve anchors, spread remaining topics evenly, and defer heavier LLM weighting or grouping until the flow is stable.	v1.1

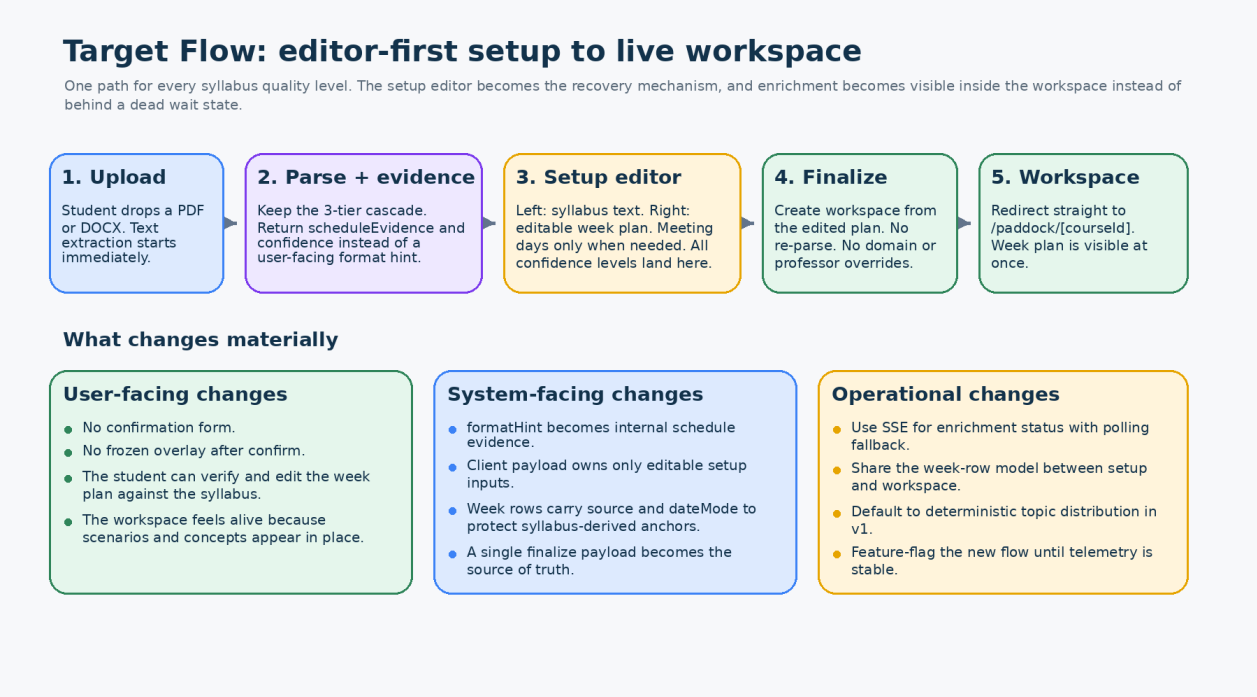
What should remain from the current branch
Keep the upload SSE progress events, the parsed plan cache in paddock_setup_drafts.parsed_plan_json, the enrichment status threading, the multi-course dashboard regression fix, and the interim confirm progress screen until the new editor path is behind a feature flag.

Root-cause diagnosis

Problem	Root cause	User impact
Frozen confirm	The user is still in the wrong surface when creation starts	The confirm screen can only wait; it cannot help the student verify or improve the plan.
Wrong fields	domain, professorName, and formatHint are user-editable	These fields can corrupt canonical matching, deduplication, and trust.
Format mistakes	detectFormatHint() counts raw date or number lines	Administrative dates and numbered policy sections are misclassified as schedule structure.
Plan rebuild risk	Meeting-day corrections can regenerate without a clear row contract	The system can mutate content that the student thinks is fixed.
Dead post-create state	The dashboard cannot express 70-200 seconds of enrichment meaningfully	The student sees a pulsing wait rather than a usable workspace.

Target end-to-end flow

The target experience is one continuous path: upload, parse, edit, create, and watch the course fill in live. Confidence changes how much is prefilled, not which screen the student sees.



Proposed end-to-end flow: editor-first setup followed by a live workspace.

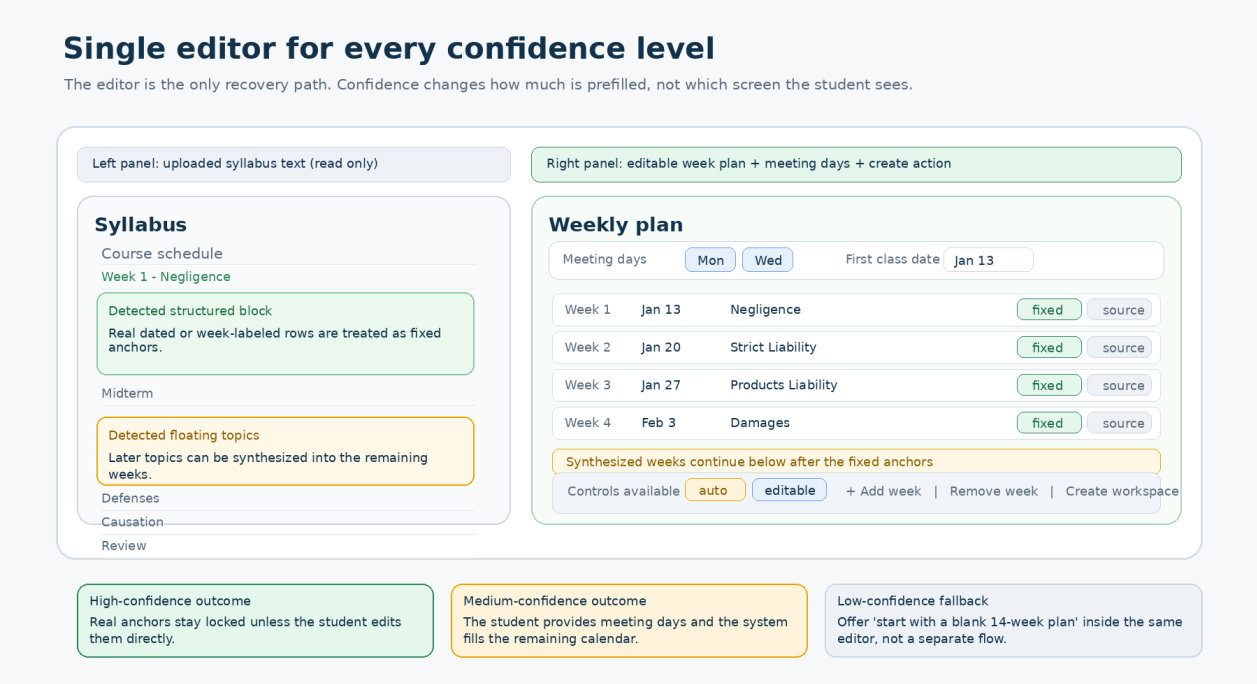
User journey

- The student uploads a syllabus and sees progress during extraction and parsing.
- The server returns a parsed draft plus schedule evidence and a confidence level.
- The student always lands in the same side-by-side editor with syllabus text on the left and the week plan on the right.
- Finalize creates the workspace from the edited plan without asking the student for domain, professor name, or a format correction.
- The student lands in the new course immediately and watches concepts, domain, scenarios, and ready-state signals appear in place.

Design principle
The editor is not a fallback branch; it is the default and only setup surface. That keeps the flow trustworthy for high-confidence parses and recoverable for low-confidence parses.

Editor-first setup experience

Every confidence level lands in the same editor. High-confidence drafts feel almost complete, medium-confidence drafts ask for the minimum missing calendar information, and low-confidence drafts can start from a blank 14-week plan without leaving the surface.



Single-editor setup: syllabus reference on the left, editable week plan on the right.

Confidence	Trigger	Editor behavior	Student ask
High	4+ genuine date-topic pairs or strong week labels	Real anchors are prefilled and treated as fixed unless the student edits them directly.	Usually none
Medium	Topic structure exists but dates are partial or absent	Topics are prefilled, dateMode=auto rows are synthesized, and the calendar fills after meetingDays and firstClassDate are provided.	meetingDays and firstClassDate when needed
Low	Few usable milestones or ambiguous structure	Show the best attempt plus a clear 'start with a blank 14-week plan' action inside the same editor.	Optional heavy editing

Decisions for unresolved UX questions

Question	Recommendation	Reason
Semester length	Do not ask explicitly in v1	Default to 14 weeks, use real anchors when present, and let the student add or remove rows.
Manual setup	Do not create a separate mode yet	A blank 14-week option inside the editor is enough for the first release.
Shared components	Share the core week-row editor model only	Setup and workspace wrappers have different responsibilities and should not be the same page component.

Parsing, evidence, and detection

The parser cascade stays. What changes is the contract around it: the system should return schedule evidence and confidence to drive the editor, not a user-facing format classification dropdown.

Internal schedule evidence object

```
type ScheduleEvidence = {
  scheduleStructure: 'dated' | 'numbered' | 'topic_based' | 'hybrid' | 'unclear';
  confidence: 'high' | 'medium' | 'low';
  dateTopicPairs: number;
  weekLabels: number;
  topicalHeadings: number;
  adminDateLines: number;
  structuredBoundaryIndex?: number;
};
```

Detection changes

Area	Current rule	Proposed rule	Outcome
Dated detection	Count raw date lines; threshold 5	Count only genuine date-topic pairs after filtering administrative lines; threshold 4	Short real schedules are detected, while scattered due dates stop misclassifying the

			syllabus.
Numbered detection	Count any numbered lines	Filter out non-topic headers such as grading, attendance, materials, and prerequisites before counting	Policy outlines stop looking like weekly structure.
User-facing signal	Expose formatHint as a correction field	Expose plain-language evidence such as 'We mapped 11 class meetings' or 'Add meeting days to place these topics on the calendar'	The UI stays truthful even when confidence is medium.
Partial temporal handling	No explicit evidence boundary	Return structuredBoundaryIndex and preserve the structured portion as fixed	The editor can mark 'from syllabus' versus 'synthesized' honestly.

Why formatHint should disappear from the UI
The student does not care whether the system labels the syllabus 'dated' or 'topic-based'. The student cares whether the mapped plan looks right. Evidence-based copy preserves trust while still giving the system the structure signal it needs.

Week model and calendar synthesis

The setup editor and the finalize endpoint need a shared row model that distinguishes syllabus-derived anchors from synthesized dates. That is the key to safe regeneration when meeting days change.

Recommended row contract

```
type WeekRow = {
  id: string;
  weekIndex: number;
  title: string;
  date: string | null;
  source: 'syllabus' | 'synthesized' | 'manual';
  dateMode: 'fixed' | 'auto';
};

type SetupFinalizePayload = {
  draftId: string;
```

```
  courseTitle?: string;
  termLabel?: string;
  meetingDays?: string[];
  firstClassDate?: string;
  weeks: WeekRow[];
};
```

Field	Purpose	Why it matters
source	Tracks whether a row came from the syllabus, was synthesized, or was added manually.	The editor can show honest provenance and the backend can treat anchored rows differently.
dateMode	fixed rows stay stable unless the student edits them; auto rows can be recalculated.	Meeting-day changes do not rewrite syllabus-derived anchors.
firstClassDate	Required only when the syllabus lacks real calendar anchors.	Meeting days alone are insufficient to synthesize a reliable calendar.

Deterministic v1 distribution

```
totalMeetings = semesterWeeks * meetingsPerWeek

preserve all fixed rows
remainingSlots = totalMeetings - fixedRows.length

if remainingTopics roughly match remainingSlots:
  spread topics 1:1 and reserve leftover slots for review
else if remainingTopics < remainingSlots:
  spread topics evenly and assign extra slots to heavier topics using simple heuristics
else:
  group adjacent related topics deterministically and let the editor remain the safety net
```

Scope boundary for v1
Do not make the first release depend heavily on an LLM for topic weighting or grouping. Use deterministic spreading and lightweight heuristics first, then add LLM assistance later for medium-confidence topic-based drafts.

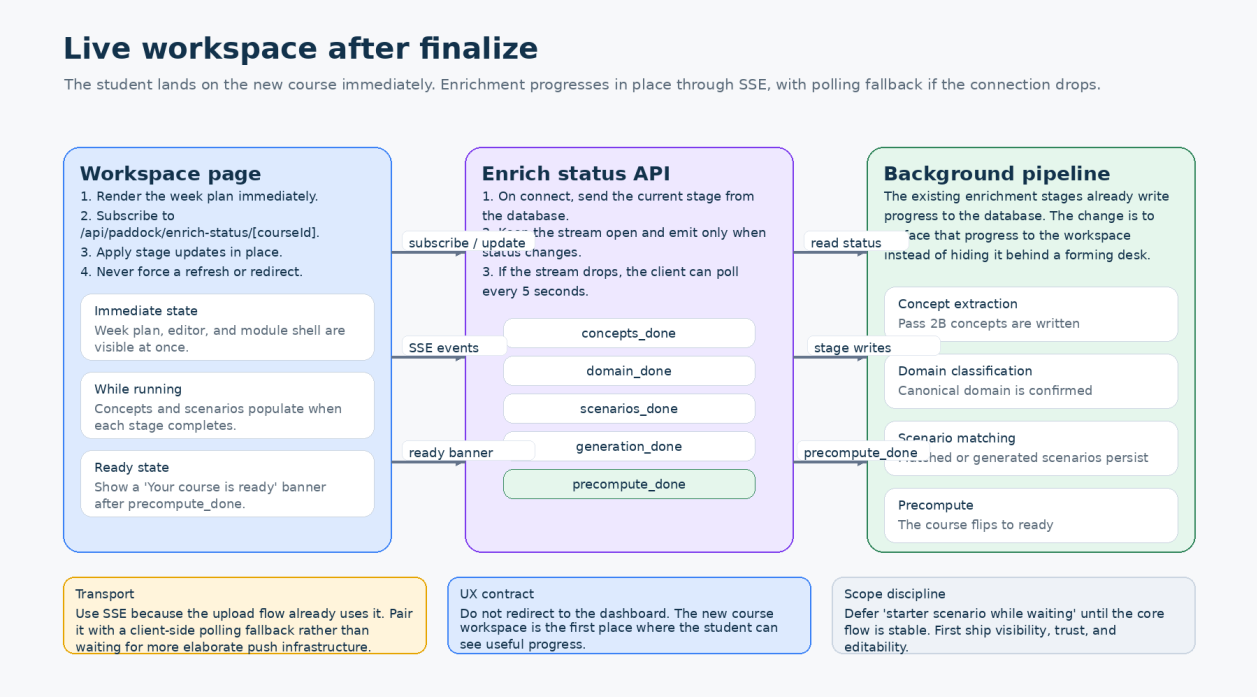
Partially temporal syllabi

For hybrid syllabi, preserve the structured part exactly as extracted, compute the remaining semester slots, and synthesize only the floating topics after the boundary. The editor should visually mark this

boundary so the student knows which rows came directly from the syllabus and which rows were inferred.

Workspace creation and live enrichment

After finalize, the student should never return to a holding dashboard. The new course workspace is the primary destination because it already contains the week plan and can accumulate enrichment in place.



The workspace subscribes to enrichment progress and updates in place.

Event	UI effect	Why it matters
concepts_done	Concepts and module-level learning structures appear in the workspace.	First meaningful content improvement after the initial plan.
domain_done	The system shows the confirmed domain internally for matching and generation.	No student override is needed.
scenarios_done	Matched or generated scenarios populate the course experience.	The waiting period becomes visibly productive.
generation_done	Longer-tail generation assets are attached when needed.	Novel domains continue to improve without blocking initial use.

precompute_done	Show 'Your course is ready' and flip remaining ready-state UI.	The student gets a clear end state without refreshing.

Transport recommendation

Use SSE for the primary transport because the upload flow already uses it and the mental model fits. Implement a polling fallback in the client so a dropped stream degrades gracefully instead of leaving the page stale.

Non-goals for the first release
Do not wait for pg NOTIFY, a perfect push infrastructure, or a starter scenario while the course forms. The first release should solve trust, visibility, and editability.

Implementation plan

The cleanest delivery path is contract-first. Update the parser evidence and finalize payload first, then build the editor, then switch the post-create destination and streaming behavior behind a feature flag.

Phase	Major work	Done when
Phase 1 - contracts	Replace user-facing formatHint with scheduleEvidence; update finalize schema; add WeekRow and firstClassDate.	New payloads compile end to end without the old fields.
Phase 2 - setup UI	Replace CourseClassificationConfirm with SyllabusEditor and WeekPlanEditor; wire confidence messaging and blank-plan fallback.	A student can create a workspace without touching the old confirm form.
Phase 3 - workspace streaming	Add /api/paddock/enrich-status/[courseId]; subscribe from [courseId]/+page.svelte; show live enrichment states.	The course workspace visibly improves after creation.
Phase 4 - hardening	Add telemetry, feature flag, rollback path, and regression tests for fixed vs auto rows.	The new flow is safe to expand.

Area	Key changes	Files
Frontend	Replace confirm phase in PaddockQuickSetup.svelte; add SyllabusEditor.svelte and WeekPlanEditor.svelte; subscribe to live status from [courseId]/+page.svelte.	apps/web-svelte/src/lib/components/paddock/* apps/web-svelte/src/routes/paddock/[courseId]/+page.svelte
Backend	Fix detectFormatHint() behavior or replace it with evidence generation; update setup/confirm endpoint; add enrich-status SSE endpoint; add topic-distribution.ts.	apps/web-svelte/src/lib/server/paddock/syllabus-metadata.ts apps/web-svelte/src/routes/api/paddock/setup/confirm/+server.ts apps/web-svelte/src/routes/api/paddock/enrich-status/[courseId]/+server.ts
Shared model	Define WeekRow, SetupFinalizePayload, and evidence types so setup and workspace agree on row ownership.	routes/paddock/syllabus-api.ts or shared type module

Rollback path
Keep the old confirmation component behind a feature flag until telemetry shows that the new editor path is stable. Delete the old component only after the new finalize payload, redirect, and stream behavior are proven.

Acceptance criteria and telemetry

Acceptance check	Expected behavior
Setup contract	The client can no longer submit domain, professorName, or formatHint to the finalize endpoint.
Plan stability	Fixed rows never move when meetingDays changes unless the student edits those rows directly.
Topic-based fallback	A syllabus with no real anchors can be finalized with

	meetingDays, firstClassDate, and an editable set of rows.
Redirect	After creation, the user lands on /paddock/[courseId], not the dashboard.
Visibility	Enrichment stages become visible in the workspace within one status cycle of the database update.
Recovery	Low-confidence drafts can start from a blank 14-week plan inside the same editor.

Metric	Why instrument it
setup_editor_shown	How often the student reaches the editor by confidence band
setup_editor_edits_count	How much manual correction is needed before finalize
finalize_to_workspace_ms	Whether creation feels immediate
enrich_stage_duration_ms	Where the background pipeline spends time
sse_disconnect_rate	How often polling fallback is needed
fixed_row_mutation_count	Regression detector for anchor stability

Resolved open questions

Question	Answer	Rationale
Ask for semester length?	No	Default to 14 weeks unless real anchors imply otherwise. Let the student add or remove rows.
Build separate manual mode?	No	Keep one editor and provide a blank-plan action inside it.

Share the full page component?	No	Share the row editor model and validation, not the whole page wrapper.
SSE or polling?	Both	SSE first for UX; polling fallback for resilience.
Starter scenario while waiting?	Later	Not needed for the first release.

Bottom line
Make setup editor-first, evidence-driven, and workspace-first. Demote format detection from a user-facing decision to an internal signal, and let the edited week plan become the source of truth that creation and enrichment build on.

Appendix: referenced inputs

Reviewed source materials: paddock-setup-flow-v2.html and paddock-setup-redesign-handoff.md.