SkillPilot Concept Compendium · Volume II

Didactics, Learning-Plan Status, and Course Planning

Understanding-First Learning, Frontier Selection, Learning Velocity, Unified

Backend Plan Status, and Local Course Planning for Teachers

Scope of this compendium

This volume brings together the active concepts governing learning guidance and scheduling on the skill graph: understanding-first learning via the Feynman Technique, frontier selection strategies and Autopilot, the learning-velocity momentum metric, the unified backend-driven learning-plan status shared across Cockpit, chat, and planning, and the browser-local teacher workflows for goal-book scheduling, exam preparation, curriculum-update continuity, and viewing existing learners by SkillPilot ID. The formal graph and completion model is specified in the Skill-Graph Compendium; the earlier Curriculum Time-Axis Exploration remains preserved as background design context.

As of 11 November 2026 · License: Apache-2.0 (technical concept specification; see LICENSING.md).

Binding product clarifications (including 4 November 2026) and stated implementation boundaries apply unchanged.

Contents

1 Feynman Technique for Learning **3**
1.1 Core principle . . . 3
1.2 Steps . . . 3
1.3 Benefits and application tips . . . 3

2 Frontier Goal Selection Strategies and Autopilot **4**
2.1 Frontier goal selection strategies . . . 4
2.2 Autopilot mode . . . 4

3 Learning Velocity **6**
3.1 Purpose and conceptual definition . . . 6
3.2 Interpretation, design principles, and caveats . . . 6

4 Unified Backend-Driven Learning-Plan Status **7**
4.1 Purpose and guiding principles . . . 7
4.2 Domain meaning of the status . . . 7
4.3 Authoritative data foundation . . . 8
4.4 Implicit completion and optional explicit-only counting . . . 9
4.5 Daily and weekly period basis . . . 10
4.6 Authoritative per-subject calculation . . . 10
4.7 Authoritative output format . . . 11
4.8 Backend architecture and data contract . . . 12
4.9 Channel behavior across chat, Cockpit, and planning . . . 12
4.10 Goal selection and focus control . . . 13
4.11 Non-evaluable situations . . . 13
4.12 Scope, verification, and acceptance . . . 13

5 Learning-Goal Book and Local Time Planning for Teachers **15**
5.1 Product goal . . . 15
5.2 Stage 1 capabilities . . . 15
5.3 Security and semantic boundaries . . . 15
5.4 Acceptance criteria . . . 16

6 Exam Material, Task Coverage, and Preparation **17**
6.1 Three distinct statements . . . 17
6.2 Completing preparation . . . 17
6.3 Active and inactive sections . . . 18
6.4 Progress, exam focus, and coach launch . . . 18

7 Reliable Course Plans Across Curriculum Updates **19**
7.1 Goal and current baseline . . . 19
7.2 Plan-author workflow and status states . . . 19
7.3 Platform obligations across updates . . . 20
7.4 Four verifiable rollout steps and code anchors . . . 21

8 Existing-Learner Teacher View by SkillPilot ID **22**
8.1 Product invariants . . . 22
8.2 Local setup, refresh, and plan-handoff flow . . . 22
8.3 Read boundary, encryption, and legacy retirement . . . 23

1 Feynman Technique for Learning

1.1 Core principle

The Feynman Technique, inspired by physicist Richard Feynman's emphasis on deep understanding over rote memorization, is a method to learn and internalize concepts by simplifying and teaching them. As Feynman noted: *"I don't know what's the matter with people: they don't learn by understanding; they learn by some other way—by rote, or something. Their knowledge is so fragile!"*

In SkillPilot, this principle motivates diagnostic dialogue on understanding goals (see Node Types): genuine mastery shows up when a learner can explain and apply an idea independently without leaning on formulaic cues.

1.2 Steps

1. **Choose a concept:** Select the topic or learning goal you want to understand. Write down everything you already know about it.
2. **Teach it simply:** Explain the concept in your own words as if teaching it to someone without prior specialist knowledge. Use plain language, avoid hiding behind jargon, and incorporate concrete examples or analogies.
3. **Identify gaps:** Notice where your explanation falters, becomes convoluted, or relies on unverified assumptions. Those points mark the exact knowledge gaps on the skill graph.
4. **Review and simplify:** Return to the source material or prerequisite goals to close those gaps. Refine your explanation until it is clear, concise, and accurate; repeat as needed.

1.3 Benefits and application tips

- Promotes durable comprehension and transfer rather than fragile recall.
- Reveals superficial familiarity early, before building dependent skills on top of it.
- Applies across disciplines — from physics and mathematics to languages and history.
- Strengthens active articulation in written or spoken coaching sessions: explain aloud or in writing without notes, and iterate until the reasoning stands on its own.

2 Frontier Goal Selection Strategies and Autopilot

This chapter specifies how SkillPilot orders available next goals on the learner's frontier and how the optional **Autopilot** mode transitions between goals.

2.1 Frontier goal selection strategies

Once the backend determines the valid frontier (`stateMachineOptions`) from the skill graph, personal curriculum scope, and mastery state, selection strategies order those candidate goals to match different study habits.

Strategy 1: Random (*Abwechslung*)

- **Concept:** Introduces variety by shuffling the currently reachable frontier goals.
- **Use case:** Suited to learners who want interleaved practice across topics or want to avoid fatigue from staying in a single branch too long.
- **Implementation:** The backend supplies the valid `stateMachineOptions` set; the client shuffles the list deterministically (or on refresh) to present a varied selection without violating `requires` prerequisites.

Strategy 2: Sequential (*Schritt für Schritt*)

- **Concept:** Follows a structured progression along the curriculum tree.
- **Use case:** Suited to learners who prefer completing one topic cluster cleanly before moving to the next branch.
- **Implementation:** Candidate goals are ordered by their hierarchical tree position (tree index / stable goal order), prioritizing sibling goals within the current branch before advancing to subsequent branches.

2.2 Autopilot mode

Autopilot reduces navigation friction by automatically advancing the learner to the next eligible frontier goal when the current goal is completed.

Runtime behavior

- **Trigger:** Fires when the current `activeGoal` transitions to the **Mastered** state.
- **Selection:** Picks the **first** goal from the valid frontier after applying the configured selection strategy (*Random* or *Sequential*). When an active learning plan governs progression (see 8.2), plan handoff rules take precedence and generic Autopilot is suppressed.
- **Action:** Invokes `setActiveGoal(nextGoalId)` automatically without requiring manual tree selection.

Configuration and persistence

- **UI:** Configured in the *Mein Lehrplan* (*Personal Curriculum*) settings modal.
- **Persistence:** Stored on the backend `Learner` entity (`learningStrategy`, `autoPilot`, `strictMode`, `showGoalVisualizationsInChat`) so preferences persist across sessions and devices.
- **API:** Updated via `PUT /api/ui/learners/{skillpilotId}/preferences` using a partial JSON payload (`learningStrategy`, `autoPilot`, `strictMode`, `showGoalVisualizationsInChat`); omitted fields keep their persisted values.
- **Strict-mode interaction:** `strictMode` determines whether frontier prerequisites are checked only within the active scope (*optimistic*) or across the complete global prerequisite closure (*strict*; see Context and Views).

3 Learning Velocity

Boundary to Cockpit History and Schedule Status

The current Cockpit chart counts **recorded completion events**, not mutable mastery update timestamps. An event can represent a renewed completion on a later calendar day rather than necessarily a lifetime first transition to mastered, and the event ledger includes completion markers such as orientation without certifying subject-matter mastery. Raw weekly velocity alone must **not** be presented as schedule adherence; authoritative schedule evaluation is governed by 4, and broader calendar-pacing exploration is documented in Curriculum Time Axis and Pacing.

3.1 Purpose and conceptual definition

Learning Velocity is a didactic momentum metric reflecting a learner's **consistency over time** — encouraging steady habits (*"don't break the chain"*) rather than isolated cramming bursts.

Conceptually, Learning Velocity is the count of newly mastered **atomic** goals per calendar week w:

$$\text{velocity}(w) = \left|\{\, g \in A \mid \text{mastered}(g) \wedge \text{achieved_at}(g) \in w \,\}\right|.$$

- **Atomic goals only:** Only leaf goals count (never cluster containers), preventing structural double-counting.
- **Binary goal outcome:** A goal is either mastered or not mastered; there is no partial goal credit. The current storage compatibility threshold `mastery >= 0.9` interprets persisted values and is not a grading scale or permission for partial mastery.
- **Calendar basis:** The present Cockpit implementation ("recorded completions", see Daily Learning Progress) groups events by `Europe/Berlin` calendar weeks, counts the same goal at most once per day, and keeps legacy undated history visible without backfilling it into the chart.

3.2 Interpretation, design principles, and caveats

- **Behavioral indicator, not a grade:** A steady sequence of non-zero weeks signals sustainable learning habits; peaks often reflect sprint phases and gaps highlight interruptions or overload.
- **Fairness, stability, motivation:** Counting only atomic goals across fixed weekly buckets (such as an 8–12 week bar chart alongside recent achievements) keeps the indicator transparent at a glance.
- **No import spikes:** Imports and mutable update timestamps must not fabricate completion spikes; the transactional completion ledger excludes them rather than reconstructing artificial timestamps.
- **Non-comparability across learners and calendars:** Learners have different time budgets, atomic goals vary in effort, context switches (a new subject or phase) can temporarily lower weekly counts, and raw calendar weeks do not model holidays, cancelled lessons, or explicit buffers.
- **Optional future extensions:** Moving-average trend lines, per-subject or per-phase breakdowns, and strictly opt-in inactivity reminders.

4 Unified Backend-Driven Learning-Plan Status

4.1 Purpose and guiding principles

SkillPilot uses a single shared learning-plan status across Cockpit, chat, and learner-facing planning to answer three questions:

What workload is scheduled for the current period? How does cumulative progress compare against the plan? What will the learner work on next?

Both the calculation and the wording live entirely in the backend. Cockpit, chat, and planning surfaces render the supplied statements verbatim — they neither recalculate the status locally nor invent competing assessments.

Core Principle

One domain calculation, one authoritative wording, multiple output channels.

A typical status output reads:

> Mathematics: Daily target reached · 3 learning goals behind Physics: No daily target today · 2 learning goals behind

When transitioning into instruction, the active goal is announced separately:

> **Your active learning goal: Describe power functions with integer exponents**

Simplicity as a design goal The design favors a clean, dependable contract for standard use cases over intricate special-case machinery. **That deliberate scope limit applies to feature breadth, never to the correctness of supported cases:** situations that cannot be evaluated reliably are named explicitly rather than masked behind estimates or heuristics. Existing plan structures, learner state, and sync boundaries are reused; this specification unifies distributed status logic into one shared backend component rather than introducing a new scheduling engine.

Binding Product Clarification of 4 November 2026

Total plan achievement (`I`) counts both explicit and implicit mastery by default. Implicit mastery means demonstrated independent application in suitable closing assessments — not merely skipped preparation. Valid recognition also satisfies later prerequisites and is revoked neither by new downstream consumers nor by switching plans; recognition arises within a valid learning unit and takes effect globally. Daily and weekly period completions (`H`) continue to count only actual explicit completion events.
The learner preference "Count only explicitly achieved learning goals" affects only cumulative plan achievement (`I`). The corrected contract below is not yet confirmed as fully implemented or deployed; the earlier demand-stop implementation and its checks do not prove it. See Assessment-Gated Completion.

4.2 Domain meaning of the status

Period target, cumulative balance, and active goal The status separates three independent statements: the current period's target, the quantitative plan balance, and the active learning goal.

Statement	Must not be inferred from it
Daily or weekly target reached	Nothing remains open overall.
No target for the current period	There is no cumulative backlog.
Quantitatively on track	Every goal scheduled for earlier periods is mastered.
Active learning goal present	The current period target is still unfinished.
Further learning available	Extra study beyond the target was already agreed.

A completed daily target can coincide with a remaining backlog, and an active learning goal can serve catching up, working ahead, or finishing an in-progress unit. Status wording never frames these combinations as contradictory.

Quantitative balance per subject Every progress-relevant planned goal counts as one unit; neither elapsed study time nor difficulty weights enter the balance. Later-scheduled goals that are already mastered count as work ahead and can offset quantitative backlog within the same subject.

Example: three earlier-scheduled goals are still open while five later-scheduled goals are already mastered. Quantitatively the learner is two goals ahead, while the three earlier goals remain substantively open and prioritized during goal selection.

> **"On track" describes quantitative progress volume. It does not certify mastery of every specific topic scheduled up to that date.**

Displayed backlog is a quantity of this net balance, not a raw count of all earlier-scheduled open goal IDs. Detailed plan views may additionally list specific open goals.

No cross-subject offsetting Every subject is evaluated independently: working ahead in Mathematics never offsets a backlog in Physics. There is no cross-subject net score, overall percentage, or combined traffic light. Cross-subject `requires` edges in the skill graph remain unaffected: globally recognized competence satisfies prerequisites in other subjects too, while each subject plan counts solely its own deduplicated goal IDs.

4.3 Authoritative data foundation

Single source of learner outcomes Persisted learner goal outcomes remain the sole individual outcome source. Plan fulfillment is derived from explicit mastery together with globally effective assessment-gated recognition, without writing extra mastery rows or maintaining a second outcome store. Cockpit, goal tree, and teacher planning maintain no separate manual teaching-progress state; former manual teaching coverage and completeness attestations are retired as a progress model (legacy local records remain preserved without producing mastery). Planning sets scope and dates; it neither certifies nor mutates outcomes.

The learner's full personal target set and the plan set G need not coincide: a goal mastered before plan creation belongs to personal achievement without entering G. G is a temporal selection rather than an independent recognition contract — global recognition is computed first and intersected with G afterward. Changing plans or focus never revokes existing recognition. Completed orientation goals count consistently within their scope without claiming subject mastery. Missing or stale outcome reads are surfaced as unavailable, never as zero achieved goals, and status evaluation is strictly read-only.

Baseline-cleaned plan set At plan creation, the baseline records which scheduled goals did not yet have explicit mastery; those goals form the progress-relevant plan set G. Currently recognized implicit goals are not stripped from G at creation time so that enabling "Count only explicitly achieved learning goals" later can still account for missing explicit completions.

Every quantity in the status calculation refers exclusively to this same baseline-cleaned plan set G. Goals explicitly mastered prior to plan creation contribute neither to due targets, nor to cumulative achievement, nor to period completions; goals in G mastered later stay in G and count toward `I` (example: 50 cluster goals with 20 already mastered $\Rightarrow |G| = 30$; scheduling 5 in week 1 yields a weekly target of 5). No separate cutoff timestamp is required.

Stable identity, plan changes, and legacy plans

- **Stable IDs:** Evaluation uses stable subject and goal UUIDs rather than display titles or translations.
- **Deduplication:** Multiple active plans for the same subject merge into one set where each goal ID counts at most once at its earliest valid due date; review/exam blocks add no duplicate due count, and clusters never count on top of their atomic goals.
- **Explicit mutations only:** Scope, dates, or baselines change only through explicit plan edits. Passing a plan's end date does not silently drop open goals from an active plan. Ambiguous legacy baselines are flagged as non-evaluable rather than guessed from current mastery.

4.4 Implicit completion and optional explicit-only counting

Let M denote explicitly mastered goals and W the non-explicit goals globally recognized under Assessment-Gated Completion. By default:

$$M_{\text{eff}} = M \cup W, \qquad I = |G \cap M_{\text{eff}}|.$$

When **"Count only explicitly achieved learning goals"** is enabled, $I = |G \cap M|$ is used instead.

W is reconstructed from fixed authored learning-unit contracts and stored explicit assessment/recall outcomes — not merely from currently displayed or selected curriculum slices. Authored contracts governing existing results remain authoritative so later graph edits never retroactively grant extra competences to past exam results.

Closing assessments of a learning unit must verify independent application of the recognized competences (a bare `requires` edge or lack of further consumers is insufficient). Within the directly governing unit, recognition propagates transitively across mastered intermediate goals; open assessments in later or outer units block neither local recognition nor its global effect.

- **Example:** Suppose Q is the sole closing assessment recognizing A and B, with all three in G. Passing Q yields $I = 3$ by default and $I = 1$ under explicit-only counting; only Q receives an explicit mastery record. A later exam Q_2 requiring A sees A as satisfied without resetting I, whereas ordinary mastery of B alone does not recognize A while Q remains open.
- **Modeling boundaries:** An ordinary goal lacking valid assessment attachment is a modeling defect, not automatic progress. Assessment and memorization goals remain explicit obligations. Missing or unresolvable recognition contracts surface as an evaluation limitation without falling back to legacy demand-stop waivers.

- **Single counting and explicit `H`:** Every goal ID in G counts at most once; explicitly completing an already recognized goal does not increment I a second time, though its genuine explicit completion today may count toward H. Deriving W, switching plans, importing state, or re-rendering never fabricates period events in H.
- **Preference and plan details:** Under *Configuration* → *Learning Settings*, "Count only explicitly achieved learning goals" persists per SkillPilot ID (default: off). Help copy: *"Your overall plan also counts competences whose independent application was demonstrated in assessments. With this setting, only directly completed learning goals count. Your daily or weekly target always counts direct completions only."* Under *Plan Details*, optional `planCredit Text` explains implicit credit (e.g. *"Credited in plan: 3 goals, including 2 via independent application in assessments."*). This specification clarification neither approves the older implementation plan nor claims runtime deployment.

4.5 Daily and weekly period basis

Learner configuration persists a per-SkillPilot-ID choice between **1 Day** and **1 Week** (default when unset: **1 Week**; an explicit daily setting is preserved). The reference time zone is **Europe/Berlin** with no custom week-start or timezone selector.

Setting	Current period
1 Day	Current calendar day (`Europe/Berlin`)
1 Week	Current Monday-through-Sunday calendar week (`Europe/Berlin`)

The chosen basis governs status math, Cockpit gauges, chat status, learner planning, and plan-driven goal selection alike. In weekly mode the running week forms a single unit: **elapsed days inside the current week create no intra-week overdue backlog**; any remaining deficit rolls into backlog only at the period boundary. Partial boundary weeks count only actually scheduled goals (never 7× a daily quota). Toggling between daily and weekly mode changes neither mastery nor plan dates, only the evaluation window.

4.6 Authoritative per-subject calculation

Inputs and completion events All inputs are integers over the subject's valid, baseline-cleaned, deduplicated plan set G satisfying $0 \le P \le S \le |G|$ and $0 \le H \le I \le |G|$ (invalid inputs are never silently clamped):

Symbol	Meaning
S	Goals in G scheduled up to and including the **end of the current period** (Sunday night in weekly mode).
P	Goals in G newly due within the current period (scheduled daily or weekly target).
I	Goals in G currently satisfied for the plan under 4.4.
H	Distinct goals in G with a verified **explicit** completion event inside the current period that remain mastered.

Only genuine explicit completion events count toward H (excluding imports, generic timestamps, duplicate submissions, or preference toggles; at most once per goal per period). Mastered state without a reliable event timestamp contributes to I but not H; event-ledger read errors mark evaluation degraded rather than assuming $H = 0$.

Calculation and direction

```
RawBalance                = I - S
RemainingDue              = max(0, -RawBalance)
OpenPeriodTarget          = min(RemainingDue, max(0, P - H))
Backlog                   = RemainingDue - OpenPeriodTarget
Ahead                     = max(0, RawBalance)
CompletedPeriodTarget     = P - OpenPeriodTarget
```

> **Completions in the current period satisfy the current period's target first; any surplus reduces backlog or builds work ahead.**

Prior work ahead can likewise satisfy all or part of P; P is never capped downward when covered ahead of time, but reported as reached. Status direction is `behind` when Backlog > 0, `ahead` when Ahead > 0, and `on_track` otherwise (a negative raw balance can reside entirely inside the still-open current period target).

Reference cases (daily mode; weekly mode analogous)

Scenario	$S/P/I/H$	Output
Start of day, no backlog	`13 / 3 / 10 / 0`	Daily target 0 of 3 · on track
Backlog, one goal done today	`13 / 3 / 9 / 1`	Daily target 1 of 3 · 2 learning goals behind
Daily target met, backlog left	`13 / 3 / 12 / 4`	Daily target reached · 1 learning goal behind
Worked ahead of cumulative due	`13 / 3 / 15 / 6`	Daily target reached · 2 learning goals ahead
Daily target covered by prior work	`13 / 3 / 13 / 0`	Daily target reached · on track
No daily target, backlog present	`10 / 0 / 8 / 0`	No daily target today · 2 learning goals behind
Exam passed today + 1 recognized	`3 / 3 / 2 / 1`	Daily target 2 of 3 · on track
Same 2 credits under larger backlog	`8 / 3 / 2 / 1`	Daily target 1 of 3 · 4 learning goals behind

4.7 Authoritative output format

For every subject with an active plan, the backend emits one line `<Subject>: <PeriodPart> · <BalancePart>` in a deterministic cross-channel order to assemble **`status.text`** (with no LLM summarization or subject omission).

Meaning	German	English
Daily progress ($P > 0$)	`Tagesziel N von P`	`Daily target N of P`
Daily target met	`Tagesziel erreicht`	`Daily target reached`
No daily target ($P = 0$)	`Heute kein Tagesziel`	`No daily target today`
Weekly progress ($P > 0$)	`Wochenziel N von P`	`Weekly target N of P`
Weekly target met	`Wochenziel erreicht`	`Weekly target reached`
No weekly target ($P = 0$)	`Diese Woche kein Wochenziel`	`No weekly target this week`
Backlog	`N Lernziele im Rückstand`	`N learning goals behind`
Ahead	`N Lernziele vorgearbeitet`	`N learning goals ahead`
On track	`im Plan`	`on track`

Singular/plural inflection is handled by the backend; raw balances and "+N extra" suffixes are omitted. "Daily target reached" must never be paraphrased as "Completed P goals today". Output language is passed explicitly (fallback: stored learner language, then German); the AI coach never translates or rewrites `status.text`.

Separate active-goal announcement: `status.text` never embeds the active goal. The backend supplies **"Dein aktives Lernziel: ..." / "Your active learning goal: ..."** separately once at the start of instruction; pure status queries trigger neither task generation nor goal announcements.

4.8 Backend architecture and data contract

A dedicated read-only backend component separates calculation from template rendering without introducing a microservice or generic rules engine. The shared result carries period metadata, overall evaluability and `status.text`, per-subject status lines/directions and Cockpit gauge projections, continuation controls, the separate localized active-goal announcement, and optional `planCreditText`.

Cockpit gauge projections (`periodGauge` and `balanceGauge`)

- **First dial (`periodGauge`, titled "Today" or "This Week"):** supplies `completed` (satisfied period quota in goals from G), `target` (P), and `needlePosition = completed / target` when $P > 0$, or `completed = 0`, `target = 0`, `needlePosition = null` when $P = 0$ (`null` if non-evaluable).
- **Second dial (`balanceGauge`, titled "Overall"):** uses `net = Ahead - Backlog` (**not** $I - S$, so open current-period work alone never displays as behind). Typical workload M = typicalAmount is the median of all **positive** deduplicated period targets across the subject's schedule (even sample sizes round the midpoint average up to the next integer; zero-target periods are excluded; if no positive period exists, `balanceGauge = null`). With scaleLimit = $2M$, `needlePosition = min(1, net / scaleLimit)` for net ≥ 0 and `needlePosition = max(-1, net / (scaleLimit + 1))` for net < 0; `severeBehind` holds iff net $< -2M$, and `strongAhead` iff net $\geq 2M$.

Reduced chat projection and consistency Chat receives only the rendered status and continuation controls — neither legacy counters (`dueToday`, `completedToday`, `openToday`, `openOverdue`, `extraCompletedToday`) nor `periodGauge`/`balanceGauge`. Every evaluation reads plan, mastery, completion ledger, preferences, and the resolved Personal Curriculum closure from one consistent snapshot respecting period boundaries.

4.9 Channel behavior across chat, Cockpit, and planning

- **Chat:** Emits `status.text` verbatim at session start/resume, on status questions, and after status-relevant updates at most once per reply, without adding competing summaries. Upon mastering a goal, the didactic order stays: feedback on completed goal → updated plan status (when applicable) → announcement and teaching of next goal. On external AI hosts, verbatim rendering remains an integration check item.

- **Cockpit:** Renders two equal-height needle dials per subject in plan mode (`periodGauge` blue/gray "x of y goals"; `balanceGauge` red/gray/green with the backend balance text directly beneath the needle, legible on narrow viewports, with no duplicate Settings or Continue button inside the Today panel). Missing period targets, missing `typicalAmount`, and degraded data are distinguished cleanly.
- **Planning and details:** Displays plan dates, actual sections/deadlines/buffers (including scheduled buffers with 0 remaining weekdays), and `planCreditText` inside *Plan Details*. Legacy parallel counters ("Total to date…", "7-day pace") are retired; teaching coverage and exam prerequisites remain separate concepts.

4.10 Goal selection and focus control

In weekly mode, eligible goals from the entire current week are selectable right from Monday. The backend prioritizes reachable open goals from earlier periods ahead of current-period goals while still allowing work ahead; graph prerequisites and Personal Curriculum scope remain authoritative, and an in-progress valid goal is not displaced merely because the calendar period rolled over.

Authorized continuation of an accepted plan may set temporary focus to the required plan section without mutating the Personal Curriculum. Pure status queries, pauses, active exam sessions, or unresolved subject switches trigger no focus change; reaching a period target never locks out voluntary study, nor does backlog auto-start unrequested extra work.

4.11 Non-evaluable situations

Situation	Behavior
No learning plan configured	Explicit notice without inventing a zero balance
Subject fully evaluable	Normal subject status line
Subject degraded by an invalid partial plan	Explicit warning (e.g. *"1 subject plan not evaluable (Physics)."*), never a fake full balance
Other subjects remain evaluable	Their valid status lines stay visible
Only an older snapshot available	Timestamp/staleness stated explicitly

Missing or corrupt inputs are never rendered as `0 of 0`, "target reached", or "on track"; ability to continue learning remains decoupled from schedule evaluability.

4.12 Scope, verification, and acceptance

Rollout replaces distributed frontend calculations, legacy counter fields, and conflicting coach instructions across schemas, payloads, server instructions, and tests together rather than layering another parallel calculation on top.

Verification area	Core requirements
Baseline	Pre-mastered goals stay excluded from S, P, I, H; plan goals mastered later remain in G.
Math & monotonicity	All reference cases match; $0 \leq \text{completed} \leq P$; backlog and ahead never negative or simultaneously positive; valid completions never worsen status.
Events & calendar	No double counting; proper handling of imports, missing timestamps, mastery revocations, read errors, day/week rollovers, and basis switches.
Subjects & contracts	Per-subject deduplication, zero cross-subject offsetting, fixed DE/EN templates, separate active-goal line, reduced chat projection.
Control & parity	Weekly goals reachable Monday morning; status/pause queries read-only; identical state produces identical statements in Cockpit, chat, and planning.

Acceptance criterion: The backend calculates and phrases; the Cockpit displays; the coach relays the status and teaches.

5 Learning-Goal Book and Local Time Planning for Teachers

Status and Scope Boundary

Approved by the Product Owner on 5 September 2026 for planning and implementation; Stage 1 is implemented locally under **Teachers** → **Class** → **Planning** → **Edit Plan**. This workflow neither triggers a curriculum QA rerun nor constitutes a production deployment. Core domain terms and the strict separation between scheduling and stored learner outcomes follow Curriculum Time Axis and Pacing, while keeping class records strictly browser-local rather than introducing server-side class management.

5.1 Product goal

Teachers can plan a manageable instructional unit directly from the learning-goal book governing their course: inspect and pick suitable goals, assign dates, and immediately see which goals remain unscheduled and which prerequisites lie outside or chronologically after the unit.

Stage 1 connects existing course planning with the learning-goal book without introducing a second schedule model or centralized class storage. Existing plan sections, local revisions, date checks, and explicit handoff into learner-owned schedules are reused. Subsequent exam-preparation support (6) adds exam-material vs. task-coverage comparison, a preparation-completion preview, and individually toggleable plan sections.

5.2 Stage 1 capabilities

1. **Course-projected goal selection:** Within the active course context, the teacher selects learning goals using their descriptions and book layout. The authoritative course projection bounds selectable goals; a publicly visible book goal is not automatically part of a specific course, and subject, state, stage, track, and course profile are never re-guessed.
2. **Explicit draft adoption:** Selected goals populate the local section draft while start date, end date, and optional custom title remain teacher choices. Saving updates only the browser-local plan; only confirmed Cockpit activation updates the learner's personal schedule.
3. **Accessible timeline:** A compact timeline visualizes learning sections, buffers, and milestones alongside full text/date form inputs so neither mouse interaction nor drag-and-drop is required.
4. **Visible planning basis and prerequisite hints:** Scheduled and unscheduled goals within the planning basis are highlighted together with concrete prerequisite hints distinguishing out-of-plan goals from out-of-order dates, without claiming individual learner mastery.
5. **Non-destructive source navigation:** Teachers can jump from any scheduled goal to its book entry and primary curriculum sources without losing unsaved draft edits.

5.3 Security and semantic boundaries

- **Browser-local privacy boundary:** Class names, memberships, and legacy local notes remain exclusively in browser storage; public goal-book fetches never transmit class or learner identifiers.
- **No silent schedule rewrites:** Book or curriculum updates never silently rewrite plans or dates; incompatible goals surface as explicit conflicts (see 7).
- **Single source of outcomes:** Stored learner goal results remain the sole source of achieved goals. Former manual teaching coverage and completeness attestations are retired as a progress workflow (legacy local entries remain intact without generating mastery).

- **Read-only progress and immutable baseline:** Per-learner subject progress is read-only from the backend's personal target scope and stored results (counting completed orientation goals in scope without claiming subject mastery, and reporting missing/stale reads as unavailable rather than zero). Plan set G stays fixed once captured, and learner plan status follows 4.
- **No fabricated hour budgets:** Raw atomic goal counts do not measure teaching hours; no pseudo-precise automatic hour budgets are synthesized from node counts, and missing plan evaluations render as unknown rather than "all unscheduled".
- **No AI or calendar coupling in Stage 1:** Neither new AI calls nor external calendar services or publications are added; any future AI schedule proposal would have to expose editable assumptions on time, prior knowledge, practice, and buffers. Historical OpenAI review artifacts and published packages remain untouched; local checks do not constitute deployment.

UI implementation resides in `CoursePlanLearningBook.tsx` and `CoursePlanTimeline.tsx` wired into `CoursePlanPilotView.tsx`; `testTrainerCoursePlanUi.ts` covers selection bounds, privacy, draft protection, keyboard focus, timeline proportions, and narrow viewports.

5.4 Acceptance criteria

- A realistic Mathematics or Physics unit can be created inside an existing course, populated from the learning-goal book, dated, saved, and reopened; goals outside the course projection cannot be added.
- Switching views, cancelling, and navigating back never lose saved plans or unconfirmed edits without warning; timeline, date inputs, and goal pickers work via keyboard and on narrow screens.
- Stored goal completions match across goal tree, Cockpit, and planning for identical learner/subject/scope; progress reads stay strictly read-only, and model/UI regressions, TypeScript, and build checks pass.

6 Exam Material, Task Coverage, and Preparation

Course planning links an exam milestone to explicitly selected topics and task packages (without priority weights). The milestone defines the exam date and chosen learning scope; only dated learning sections create weekly quotas and schedule balances. Entry path: **Course Planning → Open Course → Planning**.

6.1 Three distinct statements

Area	Meaning
Exam material	Explicitly selected topics and their visible `contains` subtrees.
Task coverage	Explicit `examData.coveredGoalIds` of the selected exam task packages.
Preparation	Still-open learning goals and required foundations to be worked through before the exam date.

The coverage view surfaces both directions: selected topics lacking an assigned task and extra topics required by chosen tasks. Available task packages count only once selected; assessment goals are counted separately from topic goals, and canonical goals are deduplicated.

A `requires` edge establishes preparation demand rather than tested performance, and metadata mappings are presented as *recorded topic mappings* rather than fresh quality certifications. Unavailable tasks are flagged rather than scheduled; when `coveredGoalIds` references a canonical cluster expanded into visible goals by the course view, it summarizes only those already visible goals without pulling hidden goals into the course.

For kinematics under **Physics → Phase E → Exercises**, two selectable task packages (*Vertical Throw: Signs, Apex, and Return* and *Horizontal Launch: Two Motions, One Shared Time*) each provide six sub-tasks (36 credit points) with full DE/EN solutions and rubrics covering equations of motion, diagrams, and vector-free reasoning (without claiming hands-on lab skill).

6.2 Completing preparation

On a saved active exam milestone, **Complete Preparation** opens a read-only preview (using the stored learner planning baseline, or treating all required goals as open when no learner state is linked). It distinguishes already mastered, scheduled-in-time, and still-unscheduled goals alongside extra foundations and open mappings across a window ending before the exam date with at least one weekday.

By default only the explicit exam selection is scheduled; extra task topics and prerequisite foundations are opt-in toggles. Out-of-course prerequisites are listed separately and never silently added to course targets, while the backend frontier continues enforcing prerequisites during learning.

Adopt Preparation creates an active multi-root canonical learning section without mutating exam material or duplicating goals already scheduled on time. Pulling goals forward from later sections lists every affected section and requires explicit confirmation (emptied sections remain preserved as inactive). The immutable learner planning baseline is preserved; failures keep the preview open with a clear error, and Cockpit handoff remains an explicit confirmation step.

6.3 Active and inactive sections

Every learning section, buffer, and milestone can be toggled active or inactive individually (omitted `active` defaults to active for existing records). Inactive items remain visible, editable, and reactivatable, but are excluded prior to goal expansion and deduplication: they generate neither due targets nor backlog, protect no buffer days, and never act as the next deadline.

Deactivating mutates neither stored mastery nor personal course targets nor a manually active goal; reactivating retains dates and evaluates against current mastery. A wholly inactive schedule may also be explicitly handed off to the Cockpit to pause a previously active plan cleanly.

6.4 Progress, exam focus, and coach launch

Exam material and dated scheduling remain distinct sets: an active exam's scope contains the exact visible atomic goals of its selected topics and task packages (retaining already mastered goals in the denominator; task coverage and prerequisites do not expand it). When dated preparation is still unset, the shared backend status reports exam progress alongside **Schedule open**, and the Cockpit renders overall exam progress and its expandable goal list without inventing weekly quotas or balance dials (whereas a zero-quota week inside an active dated schedule keeps its balance gauge).

One shared backend resolver determines scope across Cockpit, plan start, and coach (picking the earliest active exam date, breaking ties stably by section ID, and never merging separate exams). Activating or starting the exam plan sets the full atomic exam scope as focus while `activeGoal` marks only the current step within it.

In standard (*optimistic*) mode, effective prerequisites apply within the exam focus (starting kinematics at *"Investigate motion using diagrams"*), without marking external foundations mastered or adding them to focus; only explicit strict mode blocks on open external prerequisites. Without active exam material, voluntary study across the Personal Curriculum remains unrestricted.

Start Learning jumps from the Cockpit straight to provider selection and explicit new-session launch after re-verifying the stored SkillPilot ID and curriculum server-side. This workflow writes no mastery records and infers no competence from topic metadata.

7 Reliable Course Plans Across Curriculum Updates

Status and Scope

As of 8 September 2026 — Product Owner requested UX and architecture concept; automated transition rules and structured diagnostic payloads beyond the initial local hotfix remain future implementation work. *Plan author* refers to the teacher or mentor defining learning sections and dates. This specification implies neither automatic deployment nor changes to published coach contracts.

7.1 Goal and current baseline

A teacher who schedules "Upper-Secondary Physics through late September" must be able to keep using that schedule after editorial refinements or semantically equivalent goal revisions without manual repair steps. Teacher intervention belongs only where the chosen competence scope, a deadline, or a didactically relevant prerequisite genuinely changes. Primary protection sits **before** curriculum publication: routine updates must never silently break a substantively valid active schedule.

- **Local teacher draft:** A local fix already prevents a single retired atomic ID in the historical subject-wide baseline from locking every section in the draft.
- **Copied learner schedule:** The independent personal schedule already ignores subject-wide fingerprint churn, yet retired copied atom IDs or altered prerequisite ordering can still lock a subject plan, and split goals require reviewed transition mappings (title similarity never suffices).
- **Diagnostic and communication boundary:** Subject cards must separate local draft state from the active learner schedule, and the backend must stop collapsing distinct conflicts into generic `stale` / `personal-curriculum-changed` claims without proof. Because classes live exclusively in browser storage, no server push channel can notify an offline teacher ahead of the learner's next visit.

7.2 Plan-author workflow and status states

1. **Planning:** Pick competence scope and dates; *"Make available for learners"* copies only after confirmed preview. Compatibility promises appear in UI copy only once backed by the technical guarantees below.
2. **Compatible update:** The schedule continues running with no modal, red badge, or re-activation step (optional history note: *"Goal descriptions updated. Your schedule remains active."*).
3. **Optional competence addition:** The active plan keeps running (e.g. *"Physics active · 1 section has an update suggestion"*).
4. **Targeted review and decision:** Opening the notice lands directly on the affected section with keyboard focus, exact reason, learner impact, and explicit choices (e.g. *"Keep current scope"* vs. *"Include added topic"*); cancelling leaves the valid schedule untouched.

Every subject card answers **"What applies to the learner right now?"** first and displays **"Your draft"** secondarily.

Situation	Plan-author badge	Behavior
Active plan valid	`Active for learner`	Learning proceeds; no action needed.
Local draft edited	`Active for learner · Draft modified`	Prior schedule governs until confirmed handoff.
Optional curriculum expansion	`Active · Update suggestion`	Inspectable impact; no blocking dialog.
Genuine conflict	`Decision needed: Section …`	Exact cause, affected goals, and resolution (e.g. verified LK → GK profile change).
Technical fetch failure	`Status currently unavailable · last checked …`	Retry option; never prompts schedule edits.
Plan mode intentionally off	`Plan learning paused`	Explained as deliberate state, not an error.

Red styling is reserved for verified execution blockers; version bumps, changed atom counts, or network failures do not prove a domain conflict.

7.3 Platform obligations across updates

Separate planning intent from derived execution: Competence scope, validity, and dates are durable; expanded atomic goal lists are versioned projections, and recomputation neither rewrites historical baselines nor fabricates mastery.

Change category	Required platform handling
Editorial wording/display on same goal	Keep compatible with zero plan action.
Change outside scheduled scope	Zero impact on this schedule.
Rename / internal cluster restructuring	Resolve via stable IDs or reviewed successor links (never title heuristics).
Equivalent split of an atomic goal	Require reviewed predecessor → successor mapping covering task scheduling, evidence semantics, and progress counts (no artificial backlog spike, no automatic 100% mastery carry-over).
New competence goal in same cluster	Surface as optional suggestion; keep previously committed scope runnable.
New prerequisite edge	Check valid ordering inside existing dates first; escalate only real scope/date changes.
Learner Personal Curriculum change	Report separately as a learner-scope change with targeted reconciliation.

Before release, a dedicated update gate diffs old vs. new skill graphs for removed anchors, splits, and prerequisite changes, replays representative schedules, and blocks unresolved breaks (paired with privacy-preserving read-only impact checks on stored personal schedules and post-release smoke tests). For learners, partial conflicts keep unaffected valid subjects runnable (e.g. *"Physics is being checked. You can continue with Mathematics today."*) without unlocking unverified dependent tasks or turning learners into error messengers.

7.4 Four verifiable rollout steps and code anchors

1. **Shared diagnostic contract** replacing generic `stale` with structured cause, subject/section, impact, check timestamp, and action shared across Trainer, preview, and Cockpit.
2. **Visible UX states** on subject cards and course overviews after walking through three reference scenarios with the Product Owner (editorial update, equivalent atom split, genuine course-profile change).
3. **Reviewed successor mappings and CI update gate** across legacy and current schedule formats (the step that actually eliminates unexpected update breaks).
4. **End-to-end learner-then-teacher acceptance** backed by regressions, update gate, and smoke checks.

Relevant implementation paths: `app/src/utils/localTeacherCoursePlan.ts`, `app/src/components/CoursePlanPilotView.tsx`, `app/src/views/TrainerView.tsx`, `app/src/utils/teacherLearningPlanActivation.ts`, `app/src/components/TrainerLearningPlanActivation.tsx`, `backend/src/main/java/com/skillpilot/backend/service/LearnerService.java`, `LearnerLearningPlanService.java`, and `app/src/utils/learnerLearningPlanCopy.ts` (see also 8, 5, and Review Freeze).

8 Existing-Learner Teacher View by SkillPilot ID

Status and Credential Boundary

Active first-party WebGUI workflow (`VITE_EXISTING_LEARNER_LINKING_ENABLED`; disabled in package-consumer builds). A teacher adds an existing learner to a strictly browser-local class card by entering that learner's permanent SkillPilot ID — creating no server-side classes, teacher accounts, or membership records. **Identity limitation:** A permanent Skill Pilot ID is not a publicly shareable read-only identifier; it is the bearer secret granting full learner-state access. Apart from the explicit, confirmed **"Make available in cockpit"** plan-copy action, the Trainer UI uses it solely via read operations, while the credential itself remains a full learner key.

8.1 Product invariants

- **Sole learner ownership:** Personal Curriculum Level 2, focus, planned goals, mastery, and all other learner state remain owned exclusively by the learner. The teacher's browser stores only the local class name, local learner alias, permanent SkillPilot ID, and a sanitized Level-2 snapshot (`selected`, `filterId`, `durationModel`, `stage`).
- **Separate subject views:** Mathematics, Physics, and other selected subjects act as switchable local views of the same record, each bound to its own skill landscape and cached scope filter.
- **Functional read-only UI and explicit plan copy:** Every Trainer control that mutates learner state remains disabled for existing-learner classes. Only after explicit confirmation does **"Make available in cockpit"** copy the schedule label, dated blocks, and validated atomic goal IDs into the learner-owned personal subject schedule — omitting class names, aliases, coverage, mastery, baselines, and teacher history, with zero automatic sync.
- **Immutable planning baseline vs. live progress:** When first scheduling a scope, a dedicated `no-store` read captures all learner-facing atomic Level-2 targets and their open subset independently of Level-3 focus (storing only goal IDs, counts, and timestamp — no SkillPilot ID or numeric per-goal mastery, and omitted from plan exports). Live progress is read separately from the backend's current personal target set and stored outcomes (matching Cockpit/tree counts, including in-scope orientation completions without claiming subject mastery, and surfacing missing/stale reads as unavailable rather than zero).
- **Retirement of manual teaching coverage:** Manual teaching coverage and completeness attestations are retired as a progress workflow (no second `Unterrichts-IST`; legacy browser-local records stay intact without producing mastery).
- **Zero backend relationship records:** No teacher, class, invitation, or membership record is persisted server-side; deleting a local class removes only that browser entry while the permanent SkillPilot ID stays valid elsewhere. Password-encrypted class exports carry the permanent ID and must be handled as bearer-secret containers; former server-linked supervision cards are never migrated.

8.2 Local setup, refresh, and plan-handoff flow

1. Under **+ New class → Add an existing SkillPilot ID locally**, the teacher enters class name, local learner name/alias, and permanent SkillPilot ID.
2. The browser fetches `GET /api/ui/learners/{skillpilotId}` (`cache: no-store`), verifies the exact ID and a readable `personalCurriculum` with at least one selected subject, and creates a local `ClassSession`.

3. Opening the class refreshes profile and mastery via ordinary learner routes; scheduling additionally reads `GET /api/ui/learners/{skillpilotId}/planning-scope` (`cache: no-store`) to obtain the complete Level-2 target set independently of Level-3 focus (read-only; numeric storage threshold is a compatibility detail, not a grading scale). Editing and saving the class replaces the cached snapshot only after a successful refresh.
4. **"Make available in cockpit"** expands each block against the personalized subject graph and prompts for replacement confirmation. The server revalidates atom IDs and block focus, admits newly added IDs only while open, lets IDs already in the personal schedule remain for continuity, removes chronological overlap, and enforces an expected revision so stale local drafts never silently overwrite newer schedules — mutating neither focus, active goal, nor mastery.

The copied schedule stays learner-controlled: plan mode defaults to off, its first goal starts deliberately, and enabling plan mode suppresses generic Autopilot (2.2). After a confirmed completion, automatic handoff applies only when the completed goal belongs to at least one currently valid stored plan: a unique eligible candidate from a plan containing that goal takes precedence, falling back to a unique eligible candidate across all valid plans only if none exists there. Ties, invalid/missing plans, or absent due goals fail closed (keeping current focus without selecting a new active goal).

8.3 Read boundary, encryption, and legacy retirement

- **Frontend projection vs. bearer authority:** Requests hit standard learner endpoints (including the global mastery map); switching subjects clears navigation state and never cross-renders Mathematics state in Physics or vice versa, while enforceable read-only isolation in future features would require a distinct scoped credential. The teacher device and unlocked browser profile sit inside the trust boundary.
- **Browser storage and AES-256-GCM exports:** Active class records sit unencrypted in the local browser profile; downloaded class files are encrypted client-side via PBKDF2-SHA-256 + AES-256-GCM using a teacher-chosen passphrase never sent to SkillPilot (protecting exported files at rest only; legacy plaintext files load only as flagged migration input, and former server-linked files are rejected).
- **Removal of server-linked supervision cards:** On first Trainer load, the browser purges entries marked `linked-supervision`, `linkedSupervision`, or `teacher-membership` along with their local credentials and course-plan keys, while a forward-only database migration drops legacy workspace, course, and membership tables. Learner sessions, MCP/OAuth tools, provider launch flows, and submitted MCP Apps UI artifacts remain untouched.