EnCoach LMS — Assignment Workflow

Audience: Entity admins managing students, teachers, classrooms, sections, and batches.
Scope: Single-entity workflow. Cross-entity isolation is enforced by the API and is documented at the end.
Goal: Get every student into the right batch (course × section × term) with the right teachers, in the fewest clicks.

1. The mental model

EnCoach uses **four canonical entities** that build on each other. Read this once — it makes every screen self-explanatory afterwards.

Entity	What it represents	Owns
Classroom	A homeroom: a physical room repurposed as a class group (e.g. "Grade 7-A")	Roster of students, homeroom teachers, the courses studied here
Course	A subject taught (e.g. "Physics 102")	Course sections (A/B/C…), syllabus, learning objectives
Course Section	A timetabled instance of a course (e.g. "Physics 102 — Section A")	Optional capacity, sequence, branch
Batch	The intersection of a classroom × course × section × term	The actual enrolled students and the teachers who teach this slice

Rule of thumb: the **classroom is the anchor**. You set up the classroom once (roster + homeroom teachers), then you bind one or more (course, section, term) tuples to it. Each binding becomes a *batch* which automatically inherits the roster and homeroom teachers. Per-batch overrides are available when you need them.

```
Entity (school / academy)
  ■■■ Classroom (homeroom)              ← step 1 & 2
        ■■■ Roster (students)           ← step 1
        ■■■ Homeroom teachers           ← step 2
        ■■■ Courses                     ← step 3 (assign a course → cascade)
              ■■■ Batch (course × section × term)   ← step 4 (per-batch tuning)
                    ■■■ Enrolled students   (= roster, overridable)
                    ■■■ Teachers            (= homeroom teachers, overridable)
```

2. The 4-step workflow

The Classrooms page is the single entry point. Each tab is a numbered step. The workflow guide at the top of the detail pane shows green check marks as you complete each step.

Step 1 — Roster

Where: Classrooms → pick a classroom → tab **1. Roster**

Action: Click *Manage Roster* and pick the students who belong to this homeroom.

Why it matters: The roster cascades into every batch you create later. Set this once; downstream steps populate themselves.

API: `POST /api/classrooms/{id}/students` with `{ student_ids: [...], mode: "set" | "add" }`

Step 2 — Faculty

Where: tab **2. Faculty**

Action: Click *Manage Faculty* and pick the homeroom teachers.

Why it matters: When you assign a course in step 3, these teachers are automatically attached to the new batch. You can override per batch later.

API: `POST /api/classrooms/{id}/teachers` with `{ teacher_ids: [...], mode: "set" | "add" }`

Step 3 — Courses (cascade)

Where: tab **3. Courses**

Action: Click *Assign Course*, then:

1. Pick a course from the dropdown.
2. (Optional) Pick a section. Sections are loaded dynamically from `/api/courses/{course_id}/sections`. Leave on *Auto* to use the whole course.
3. (Optional) Enter a term key (e.g. `2026-T1`).
4. Click **Assign + Cascade**.

What the server does in one transaction:

- Creates a canonical batch `(classroom × course × section × term)` (or reuses an existing one).
- Auto-enrolls every student from the classroom roster into the batch.
- Attaches the homeroom teachers to the batch.

API: `POST /api/classrooms/{id}/assign-course` with `{ course_id, section_id?, term_key?, teacher_ids? }`. Cross-entity course/section/teacher IDs are rejected with HTTP 403.

Step 4 — Batches (per-batch tuning)

Where: tab **4. Batches**

Action: Each batch row has a **Manage** button. The dialog has two tabs:

4a. Students

Pre-checked with the batch's actual enrollments. Use this when you need a *subset* of the homeroom in this batch (e.g. half the homeroom is in section A, the other half is in section B). One click on *reset to classroom roster* re-syncs back to the cascade.

APIs:

- `POST /api/batches/{id}/students` — add students
- `POST /api/batches/{id}/students/remove` — remove students

4b. Teachers

Pre-checked with the batch's current teachers. Use this when sections are taught by different teachers, or when a substitute takes over for one batch. One click on *reset to homeroom teachers* re-syncs.

APIs:

- `GET /api/batches/{id}/teachers`
- `POST /api/batches/{id}/teachers` with `{ teacher_ids: [...], mode: "set" | "add" }`
- `POST /api/batches/{id}/teachers/remove`

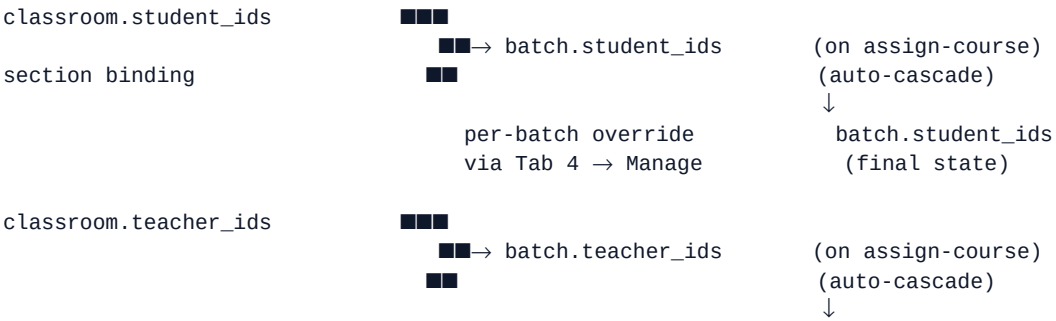
3. Worked example

Goal: enroll 24 Grade-7 students into Physics 102 (sections A and B), with Ms. Salem teaching A and Mr. Khaled teaching B.

#	Step	What you do	Result
1	Create classroom	Classrooms → New Classroom → "Grade 7"	Empty classroom shell.
2	Roster	Tab 1 → Manage Roster → pick all 24 students	Classroom now has 24 students.
3	Homeroom teachers	Tab 2 → Manage Faculty → add Ms. Salem and Mr. Khaled	Both are default teachers.
4	Assign Section A	Tab 3 → Assign Course → Physics 102 → Section A → 2026-T1 → *Assign + Cascade*	Batch *"Grade 7 — Physics 102 — A"* is created with **24 students** and **2 teachers**.
5	Assign Section B	Tab 3 → Assign Course → Physics 102 → Section B → 2026-T1 → *Assign + Cascade*	Batch *"Grade 7 — Physics 102 — B"* is created with **24 students** and **2 teachers**.
6	Split the roster	Tab 4 → on Section A's batch click *Manage* → tab Students → uncheck students 13–24 → Save	Section A has 12 students.
7	Split the roster	Tab 4 → on Section B's batch click *Manage* → tab Students → uncheck students 1–12 → Save	Section B has 12 students.
8	Pin teacher per section	Tab 4 → on Section A's batch click *Manage* → tab Teachers → keep only Ms. Salem → Save	Section A taught by Ms. Salem.
9	Pin teacher per section	Tab 4 → on Section B's batch click *Manage* → tab Teachers → keep only Mr. Khaled → Save	Section B taught by Mr. Khaled.

Total: 9 clicks beyond the picker dialogs. Steps 6–9 only happen when you need per-section overrides; if every section has the whole homeroom and the same teachers, you stop at step 5.

4. Where assignments come from (the cascade chain)



```
                                  per-batch override          batch.teacher_ids
                                  via Tab 4 → Manage           (final state)
```

- The cascade only runs **on** `assign-course` . After that, the batch and the classroom roster diverge independently. Adding a new student to the classroom **after** a batch exists does *not* enroll them automatically — re-run the assignment or use Tab 4 → Manage.
- `mode: "add"` on roster/faculty mutations unions; `mode: "set"` replaces. The UI uses `set` everywhere, so what you see is what you get.

---

5. Entity isolation guarantees

Every endpoint above is wrapped in the same multi-tenant guard:

1. The user's `entity_ids` defines their scope. A user with one or more entities is **always** entity-scoped, even if they hold Odoo system groups.
2. Foreign-key IDs sent by the client (course, section, classroom, student, teacher) are validated to belong to the **target entity**. Cross-entity IDs return **HTTP 403**.
3. Model-level `@api.constrains` rejects direct ORM writes that mix entities — so even shell scripts can't poison the data.

This is verified by the `smoke_entity_isolation.py` script, which asserts every cross-entity write attempt returns 403 and every cross-entity read leaks zero records.

---

6. Reference: endpoints used

Method	Path	Purpose
POST	`/api/classrooms/{id}/students`	Set/add classroom roster
DELETE	`/api/classrooms/{id}/students`	Remove from roster
POST	`/api/classrooms/{id}/teachers`	Set/add homeroom teachers
GET	`/api/courses/{id}/sections`	List sections for a course
POST	`/api/courses/{id}/sections/generate-defaults`	Generate A/B/C sections
POST	`/api/classrooms/{id}/assign-course`	Bind course → create batch + cascade
DELETE	`/api/classrooms/{id}/courses/{course_id}`	Detach course (keeps existing batch)
GET	`/api/batches/{id}/students`	List batch enrollments
POST	`/api/batches/{id}/students`	Add students to a batch
POST	`/api/batches/{id}/students/remove`	Remove students from a batch
GET	`/api/batches/{id}/teachers`	List batch teachers
POST	`/api/batches/{id}/teachers`	Set/add batch teachers
POST	`/api/batches/{id}/teachers/remove`	Remove batch teachers

All endpoints require a Bearer token from `POST /api/login` .

7. Smoke tests shipped with the platform

Script	Verifies
`smoke_assignment_workflow.py`	The full 4-step flow against a live backend: classroom → roster → teachers → assign-course → per-batch override. Asserts the cascade and the override are persisted server-side.
`smoke_entity_isolation.py`	An entity-A admin cannot read or modify entity-B records via any of the endpoints above.
`seed_dynamic_sections_demo_via_api.py`	Idempotently seeds a demo classroom + course-with-sections + batches using only the public API.

Run from the repo root with `.conda-envs/odoo19/bin/python <script>.py` .

Last updated: April 2026. Backend module: `encoach_lms_api` *. Frontend page:* `frontend/src/pages/ClassroomsPage.tsx` *.*