GraphQL Lab – Day 2 Homework

Project: Student Management System — Auth, Relations, Pagination, Validation
Time: ~2–3 hours
Starter: continue from your Day-1 in-memory code (`students`, `courses`, `enrollments`)

What you'll add today (at a glance)

- **Auth**: `signup`, `login`, JWT in `Authorization: Bearer …`, `context.user`
- **Guards**: only logged-in users can **create/update/delete** + **enroll/unenroll**
- **Relations**: `Student.courses`, `Course.students` (+ counts)
- **Filtering / sorting / pagination**
- **Validation** & clean deletes

Part 0 — Minimal Auth (required) — 20 pts

Packages:
`npm i jsonwebtoken`
(optional) `npm i bcryptjs` if you want to hash; plain text is acceptable for this lab.

Schema additions

- `type User { id: ID!, email: String! }`
- `type AuthPayload { token: String!, user: User! }`
-
- `type Mutation {`
- `  signup(email: String!, password: String!): AuthPayload!`
- `  login(email: String!, password: String!): AuthPayload!`
- `  # (plus your existing student/course mutations)`
- `}`

Rules

- Email must be unique (case-insensitive) and look like an email.
- Password length ≥ 6.
- `signup` creates a user in **memory** and returns a JWT (`sub` = userId or include `email`).
- `login` verifies credentials and returns a JWT.
- In `ApolloServer` **context**, read `Authorization: Bearer <token>` → `context.user` (or `null` if invalid).
  Tip: decode once per request; don't decode inside each resolver.

Guarding

- The following **require authentication**:
  `addStudent`, `updateStudent`, `deleteStudent`,
  `addCourse`, `updateCourse`, `deleteCourse`,
  `enrollStudent`, `unenrollStudent`.
- If not logged in, throw `new Error("UNAUTHENTICATED")`.

Part 1 — Schema polish & inputs — 15 pts

Add inputs:

- `input StudentUpdateInput { name: String, email: String, age: Int, major: String }`
- `input CourseUpdateInput  { title: String, code: String, credits: Int, instructor: String }`
- `input ListOptions        { limit: Int, offset: Int, sortBy: String, sortOrder: String }`
- `input StudentFilter      { major: String, nameContains: String, emailContains: String, minAge: Int, maxAge: Int }`
- `input CourseFilter       { codePrefix: String, titleContains: String, instructor: String, minCredits: Int, maxCredits: Int }`

Update list queries to accept filters/options:

- `getAllStudents(filter: StudentFilter, options: ListOptions): [Student!]!`
- `getAllCourses(filter: CourseFilter, options: ListOptions): [Course!]!`

Defaults/Caps

- `limit` default **10**, max **50**; `offset` default **0**
- `sortOrder`: `"ASC"` or `"DESC"`
- `sortBy`: one of the entity fields (`"name"`, `"age"`, `"title"`, `"credits"`, …)

Part 2 — Nested resolvers (relations) — 20 pts

Use your in-memory `enrollments` map.

- `Student.courses(parent)` → from `enrollments[parent.id]` return matching courses.
- `Course.students(parent)` → find students whose `enrollments[studentId]` includes `parent.id`.

Computed fields

- `extend type Student { coursesCount: Int! }`
- `extend type Course  { studentsCount: Int! }`

- Return counts from the same `enrollments` data.

Part 3 — Enrollment mutations — 25 pts

Add:

- `enrollStudent(studentId: ID!, courseId: ID!): Student!`

- `unenrollStudent(studentId: ID!, courseId: ID!): Student!`

Rules

- Validate both IDs exist.
- Initialize `enrollments[studentId] = []` if missing.
- If already enrolled:
  - you may **do nothing (idempotent)** or **throw an error** — pick one and mention it in README.
- These two **require auth**.

Part 4 — Update/Delete + constraints — 20 pts

Mutations:

- `updateStudent(id: ID!, input: StudentUpdateInput!): Student!`
- `deleteStudent(id: ID!): Boolean!`
-
- `updateCourse(id: ID!, input: CourseUpdateInput!): Course!`
- `deleteCourse(id: ID!): Boolean!`

On delete

- Student: remove from `students` and delete their `enrollments` entry.
- Course: remove from `courses` and remove that courseId from all `enrollments`.

Validation

- `Student.email` unique (case-insensitive) & email-ish.
- `Student.age` ≥ 16.
- `Course.code` unique (case-insensitive), e.g. `CS101`.

- `Course.credits` in **1..6**.

Part 5 — Filter → Sort → Paginate order — 15 pts

Apply **in this sequence** for both lists:

1. filter
2. sort (`ASC`/`DESC`)
3. offset/limit (respect caps)

If `sortBy` unknown → ignore or default to a safe field.

-