Student Course Registration System

Functional Requirements

Purpose: console-based student course registration system using **DynamoDB Local**. Courses are preloaded. Students can sign up, log in, view courses, enroll/drop, and use waitlists. All key actions are logged.

Actors

- **STUDENT** — signs up, logs in, views courses, enrolls/drops, views profile and waitlist, opts out of waitlist.
- **SYSTEM** — seeds courses at startup, enforces business rules, writes audit logs.

(No ADMIN interface (courses are preloaded in DB). No web/UI — console only.)

Functional Requirements (FR)

FR-01 — Student Sign Up (Mandatory)

Description: A user can create a student account by providing `studentId`, `name`, `email`, and `password`.
Behavior / rules:

- `studentId` and `email` must be unique (system checks).
- Password is **hashed** before storing (bcrypt or similar).
  Why: necessary to identify students and secure credentials.

Acceptance criteria

- Given unique `studentId` and `email` → when registering → then `Students` table has record with hashed password and empty enroll/waitlist lists.

FR-02 — Student Login (Mandatory)

Description: A registered student can log in using `studentId` + `password`.
Behavior / rules:

- Verify password against stored hash.
- On success return an in-memory **session token** (UUID) with expiry (default 30 minutes).
- All protected operations must require a valid token.

Acceptance criteria

- Given correct credentials → login returns token and StudentLogs shows `LOGIN` entry.

FR-03 — View Profile (Mandatory)

Description: A logged-in student can view their profile: `studentId`, `name`, `email`, and list of applied courses with statuses.
Statuses shown:

- `ENROLLED` — currently enrolled (active).
- `WAITLISTED` — currently in a course waitlist + show position (1-based).
- `DROPPED/COMPLETED` — past courses (endDate < today or student dropped).

Acceptance criteria

- Given student with enrollments/waitlists → profile shows lists and positions correctly.

FR-04 — Preloaded Courses (Mandatory)

Description: Courses are **preloaded** into the `Courses` table at project startup (seed). No admin UI required.
Notes: seeding script or startup code will create required course items if table empty.

Acceptance criteria

- On startup with empty `Courses` table → system inserts sample courses (C101, C102, …) with required attributes.

FR-05 — View Courses (Mandatory)

Description: Students can list all courses. Each course listing shows:

- `courseId`, `courseName`
- `currentEnrolledCount`
- `remainingSeats = maxSeats - currentEnrolledCount`
- `startDate`, `endDate`
- `latestEnrollmentBy`

Acceptance criteria

- `listCourses()` returns courses with computed `remainingSeats` and the above attributes.

FR-06 — Enroll (Mandatory)

Description: Students may attempt to enroll in a course. Enrollment succeeds only when business rules pass; otherwise the student is waitlisted if possible.

Preconditions (all must hold to directly enroll):

1. Current date ≤ `latestEnrollmentBy`.
2. `currentEnrolledCount < maxSeats`.
3. Student has fewer than 5 active enrollments (active = enrolled and endDate >= today).

On success:

- Atomically increment `currentEnrolledCount` and append student into `enrolledIds` (use DynamoDB `UpdateItem` with `ConditionExpression`).
- Add `courseId` to student's `enrolledCourseIds`.
- Append `ENROLL` log to `StudentLogs`.

When course is full:

- If student has < 3 waitlists and not already on waitlist → append to `waitlistIds` and student.waitlistedCourseIds; write `WAITLIST_JOIN` log.
- If student already on 3 waitlists → operation fails with explanatory message.

Duplicates:

- Prevent enrolling/waitlisting twice for same course.

Acceptance criteria

- Enroll success increments `currentEnrolledCount`, student shows in `enrolledCourseIds`, log recorded.
- If full and student allowed → placed in `waitlistIds` with position reported.

FR-07 — Drop (Mandatory)

Description: A student may drop a course they are enrolled in or remove themselves from a waitlist.

Rules:

- Student can only drop if current date ≤ `endDate`.
- If dropping while enrolled:
    - Decrement `currentEnrolledCount` and remove student from `enrolledIds`.
    - Remove `courseId` from student.enrolledCourseIds.

    - Append `DROP` log.
    - Trigger **waitlist promotion** (see FR-08) — sequential processing.
- If removing from waitlist:
    - Remove student from `waitlistIds` and student.waitlistedCourseIds.
    - Append `WAITLIST_OPT_OUT` log.

Acceptance criteria

- After drop, `currentEnrolledCount` decreased and one eligible waitlisted student (if any) is auto-enrolled and logs recorded.

FR-08 — Waitlist Management (Mandatory)

Description: The system handles waitlists with FIFO semantics and promotion logic.

Rules:

- Waitlist is FIFO (`waitlistIds` list on Course).
- A student can be on at most **3** course waitlists.
- **Auto-promotion**: when a seat opens, iterate waitlist in order and attempt to enroll the first eligible student:
    - If candidate has < 5 active enrollments → attempt atomic enroll (as FR-06).
    - If candidate has 5 active enrollments → **skip** them (do not remove) and check the next candidate. The skipped candidate **remains in place**.
    - On successful auto-enroll → remove candidate from waitlist, update their student record, and append `AUTO_ENROLL` log. Promote only one candidate per single seat opening (per operation).

Opt-out:

- Student may remove themselves from any course waitlist (see FR-07).

Acceptance criteria

- Waitlist promotion enrolls the first eligible candidate while preserving positions of skipped ineligible candidates.

FR-09 — Audit Logging (Mandatory)

Description: All key actions are recorded in `StudentLogs` table.

Actions logged:

- `SIGNUP`, `LOGIN`, `ENROLL`, `DROP`, `WAITLIST_JOIN`, `WAITLIST_OPT_OUT`, `AUTO_ENROLL`

Log attributes (per entry):

- `logId` (PK; UUID or timestamp string)
- `studentId`
- `action` (string)
- `courseId` (nullable)
- `timestamp` (epoch ms or ISO-8601)

Acceptance criteria

- Each action above produces a corresponding log row in `StudentLogs` with correct fields.

FR-10 — Prevent Duplicates & Enforce Limits (Mandatory)

Description: System enforces:

- No duplicate enroll or duplicate waitlist entries for same course.
- Max **5** active enrollments per student.
- Max **3** waitlists per student.

Acceptance criteria

- Attempting to re-enroll or re-waitlist fails with clear message; limits enforced.

FR-11 — Session & Minimal Auth (Mandatory)

Description: Login issues an in-memory session token (UUID) with expiry (default 30 minutes). Protected operations require valid token.

Acceptance criteria

- Token is returned on login; protected endpoints in CLI require token; expired token forces re-login.

FR-12 — Tests (Mandatory)

Description: Provide at least **5 JUnit tests** covering core flows:

- signup/login (auth)
- enroll when seat available
- enroll → waitlist when full
- drop → promotion from waitlist
- duplicate prevention & limits

Acceptance criteria

- Tests pass locally using DynamoDB Local or mocked DAOs.

Constraints (System constraints)

- All persistent storage must be **DynamoDB Local** (no SQL).
- Console-only UI.

- Preloaded courses (no admin interface).
- Passwords must be hashed.
- Atomic/conditional updates for course seat allocation (use ConditionExpression).

Non-Functional Requirements

- **NFR-01 Usability:** Console messages must be clear and informative (success/failure reasons).
- **NFR-02 Maintainability:** Follow layered architecture (models → dao → services → app). Keep business logic in services.
- **NFR-03 Security:** Passwords hashed; session tokens random and expiry enforced.
- **NFR-04 Logging:** SLF4J console logs for developers; audit logs in DB (append-only).
- **NFR-05 Testability:** Unit tests (JUnit 5) and optional light integration tests against DynamoDB Local.
- **NFR-06 Performance (demo-level):** Local operations should complete quickly; use conditional updates to avoid overbooking.
- **NFR-07 No external network dependency:** All operations should work offline with DynamoDB Local.

Acceptance Criteria

- FR-01: Student row exists with `passwordHash` not equal to raw password.
- FR-02: Login returns token, StudentLogs contains `LOGIN`.
- FR-05: `listCourses()` returns `remainingSeats` = `maxSeats` - `currentEnrolledCount`.

- FR-06: Enroll success updates Course (atomic update) and Student lists; waitlist append when full.
- FR-07/08: Drop leads to `AUTO_ENROLL` for eligible candidates and correct log entries.
- FR-09: Logs exist in StudentLogs with correct `action` and `timestamp`.
- FR-10: Attempt to enroll when already enrolled yields error; attempt to join >3 waitlists fails.
- FR-11: Expired token blocks protected operations.