LearnSphere

1) Overview

LearnSphere is a console-based e-learning platform where users can browse, enroll in, and review courses. The app persists data in a relational database. It demonstrates a clean domain model (Course, Category, User) with enrollment and review features and exposes core operations through a small service layer.

Scope covered

- CRUD for core entities
- Users can register, search courses, enroll, and review (0-10)
- Admin view of users with their enrollments
- Average rating shown on search and on demand

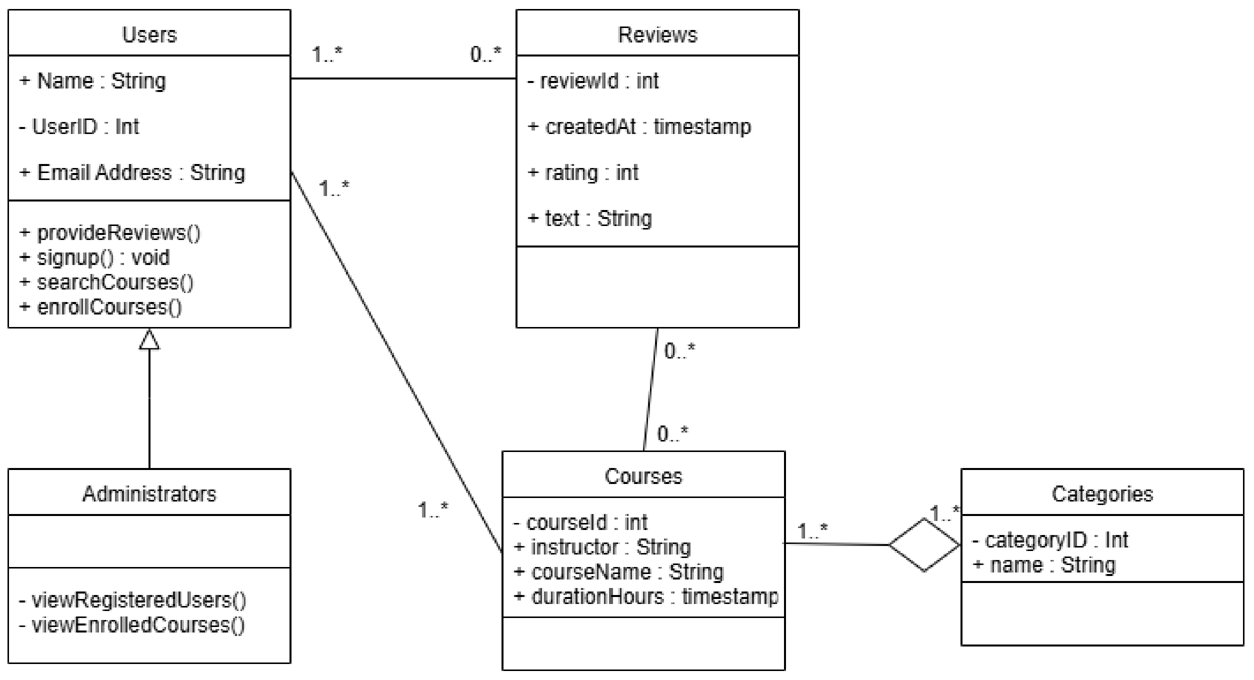
Key assumptions

- One review per (user, course); updates overwrite the previous review
- Enrollment is unique per (user, course)
- Categories are predefined (but CRUD is available)
- Average rating is the mean of all stored integer ratings (0–10) for a course

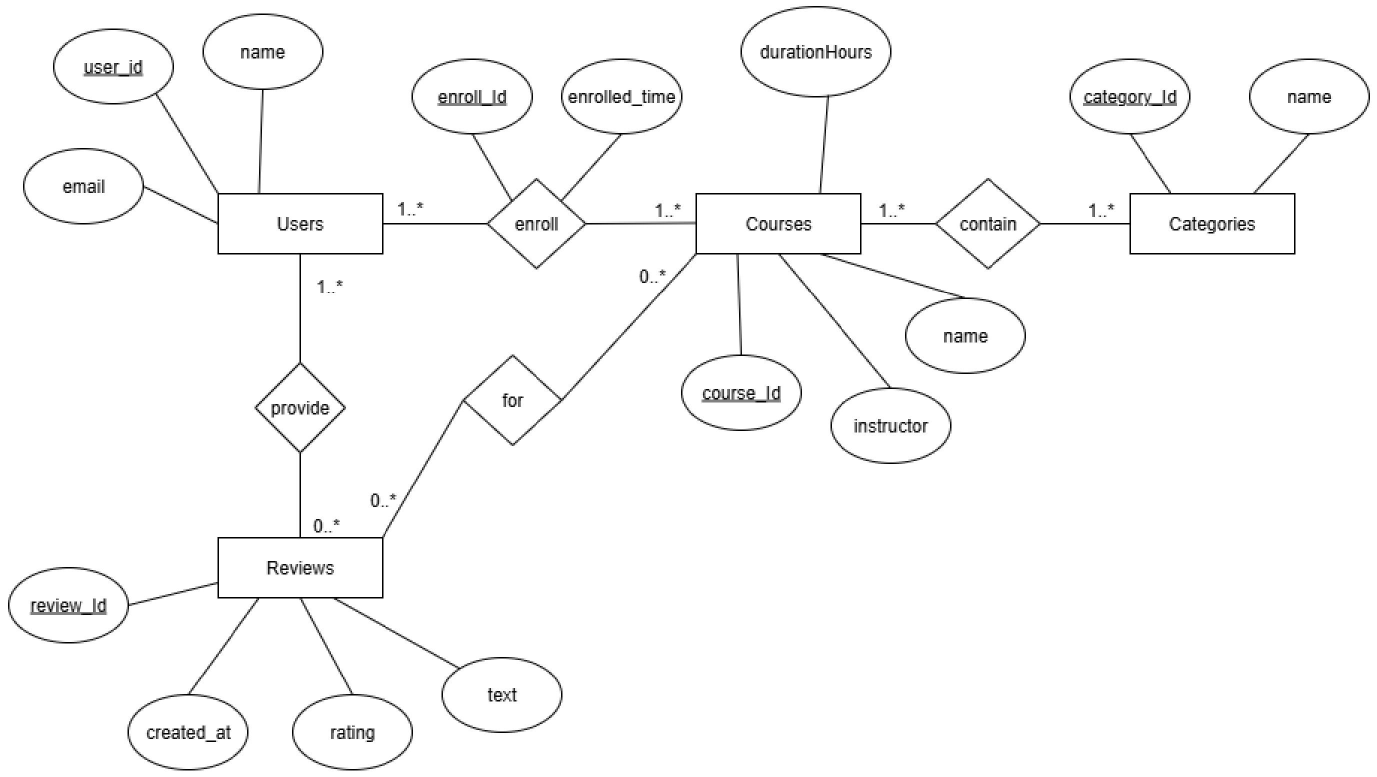
2) Architecture & Tech

- **Java 8**, **Hibernate 5.6**, **MySQL**, **Maven**
- **Package layout**
    - com.learnsphere.entity – JPA entities
    - com.learnsphere.service – transactional service methods (app logic)
    - com.learnsphere.util – HibernateUtil (SessionFactory bootstrap)
    - com.learnsphere.app – console UI (App)
- **Patterns**: simple service layer with a helper to wrap each operation in a Hibernate session and transaction

3) Class Diagram



4) ER Diagram


user_id
name
email
Users
enroll_Id
enrolled_time
enroll
durationHours
Courses
course_Id
instructor
name
contain
category_Id
name
Categories
provide
for
Reviews
review_Id
created_at
rating
text
1..*
1..*
1..*
1..*
1..*
0..*
0..*
0..*

5) Mapping and Constraints

Course

- ManyToMany categories with JoinTable (name="course_category")
- Basic fields: name, instructor, durationHours

Category

- Column (unique = true) for name

User

- Column (unique = true) for email

Enrollment

- ManyToOne with user, course
- Table-level UniqueConstraint (columnNames={"user_id", "course_id"})

Reviews

- ManyToOne user, course
- Table-level UniqueConstraint (columnNames={"user_id", "course_id"})
- rating validated 0-10

6) Core Logic

- **Transaction wrapper (all database ops go through it)**

```
private <T> T tx(Function<Session, T> work) {
  try (Session s = HibernateUtil.getSessionFactory().openSession()) {
    Transaction t = s.beginTransaction();
    T out = work.apply(s);
    t.commit();
    return out;
  }}
```

  - Ensures consistent transaction boundaries
  - Keeps the Session short-lived, per operation

- **Add course (idempotent categories, atomic create)**
  - For each category name, lookup (lowercased); create if absent
  - Create Course, attach categories, persist in one transaction

- **Register user**
  - Validate email uniqueness; persist User(name, email)

- **Enroll the user in the course**
  - Check the existence of User and Course
  - Insert Enrollment (user, course, enrolled_time)
  - Database-level unique constraint prevents duplicates; service translates violations to a friendly message

➢ **Add/Update review**

- Enforce rating ∈ [0,10]
- If a review exists for (user, course), update rating; otherwise, insert new
- Unique constraint guarantees single review per pair

➢ **Search courses (by name/instructor/category) + avg rating**

- Keyword search (case-insensitive) over Course.name, Course.instructor, and joined Category.name
- For each hit: compute average rating with a single aggregate query

```
// AVG rating
Double avg = (Double) s.createQuery(
  "select avg(r.rating) from Review r where r.course.id = :cid"
).setParameter("cid", courseId).uniqueResult();
```

➢ **Admin: Users with enrollments**

- Load users and left join their enrollments

7) What the console app shows

1. Seed (first run): Categories (etc: Programming/Design/Business), Courses ("Java Fundamentals", "Creative Marketing"), Test user
2. Menu with options: Add Category/Course, Register, Enroll, Search, Review, Avg per course, Admin view
3. Typical flow: register → list/search → enroll → review → check average → admin listing

9) Non-functional and data integrity

- Persistence: durable storage in RDBMS
- Integrity: database constraints (uniques + Foreign Keys) enforce business rules
- Portability
- Simplicity: single JAR module; no app server required

10) Testing checklist

- Register duplicate email → rejected
- Enrolling the same user in the same course twice → rejected (unique)
- Review outside 0-10 → rejected
- Update review (same user+course) → average changes accordingly
- Search by category keyword returns expected courses and displays the latest average

11) Future enhancements

- Pagination and sorting for search results
- Instructor entity; multi-instructor per course
- Soft deletes and audit timestamps
- Authentication and roles (admin vs user)