E-Learning Platform

Made by Marsel Yuldashev.   
Supervised by AL-Magsoosi Husam Kareem Farhan.

# Description

This platform connects teachers and learners through online courses. Teachers can build courses with lessons, submit them for approval, and share knowledge. Learners browse courses, track their progress, and mark lessons as completed.

Administrators ensure quality by reviewing submissions. Everyone gets a tailored experience: teachers manage content, learners focus on their goals, and admins maintain standards.

It’s designed to make learning organized, accessible, and engaging—whether you’re sharing expertise or gaining new skills.

# Technologies used

## Backend

* **Spring Boot**: Powers the server-side logic, handling HTTP requests, user authentication, and connecting all components.
* **Spring Security**: Manages login/logout, Google OAuth2 integration, and role-based permissions (learner, creator, admin).
* **Spring Data JPA**: Simplifies database operations (CRUD) for courses, lessons, and users.

## Database

* **Hibernate**: Translates Java objects (like Course, Lesson) into database tables and relationships.
* **MySQL**: Stores all data (users, courses, enrolments, completions).

## Frontend

* **Thymeleaf**: Generates dynamic HTML pages (e.g., displaying courses, progress badges) using server-side data.
* **Bootstrap 5**: Provides responsive layouts, styling, and components (cards, grids, buttons).
* **Font Awesome**: Adds icons (user profile, search) for visual clarity.

## Integrations

* **Google OAuth2**: Lets users log in securely with their Google accounts.
* **YouTube Embed API**: Converts video URLs into playable lessons using iframes.

## Search & Queries

* **JPQL (Java Persistence Query Language)**: Powers case-insensitive course searches.

## Tools

* **Maven**: Manages project dependencies.

# UML diagram

бичвэр, шугам, дэлгэцийн зураг-г агуулсан зураг

AI-generated content may be incorrect.

**Complete lessons** – Learners mark lessons as finished to track progress.  
**View course completion percentage** – Displays how much of a course a learner has completed.  
**Enroll in courses** – Learners sign up for courses they wish to take.  
**View enrolled courses and their completion status** – Learners check their enrolled courses and progress.  
**Observe enrolled learners in a course** – Instructors/admins see a list of learners in a specific course.  
**Search courses** – Users find courses using keywords, categories, or filters.  
**Create lesson/course** – Instructors or creators design new lessons or courses.  
**Approve lesson/course** – Admins review and publish submitted content.  
**Edit course** – Modify existing course content (e.g., updates, corrections).  
**Observe created courses and their approval status** – Creators monitor their courses’ publication status.  
**Delete course** – Admins/creators remove a course from the platform.

# Class diagram

бичвэр, тоо, дэлгэцийн зураг, шугам-г агуулсан зураг

AI-generated content may be incorrect.

**Learner** is a user who can enroll in courses, complete lessons, and track their progress.  
**Course** is a structured collection of lessons created by a CourseCreator and can be approved by an Administrator.  
**Lesson** is an individual unit within a course that learners can complete.  
**Administrator** is a user who approves courses and lessons to ensure quality control.  
**CourseCreator** is a user who creates and manages courses for learners.  
**LearnerRepository** provides methods to fetch and manage learner data from the database.  
**CourseRepository** handles retrieval and storage of course-related data.  
**LessonRepository** manages access to lesson data, including filtering by course or approval status.  
**AdministratorRepository** allows fetching Administrator data by email.  
**CourseCreatorRepository** enables retrieval of CourseCreator data and their created courses.  
**LearnerController** handles web requests related to learner activities like course enrollment and lesson completion.  
**CourseController** manages operations related to courses such as search, enrollment, approval, and completion.  
**RegistrationController** deals with user registration and profile setup.  
**SecurityConfig** defines the application’s authentication and access control rules.  
**GlobalUserRoleAdvice** injects user role information into responses for consistent UI behavior.  
**GlobalUnapprovedCourseLessonAdvice** provides logic to process or flag unapproved courses and lessons.  
**ELearningApplication** is the main class that starts and runs the Spring Boot application.

# Database diagram

бичвэр, дэлгэцийн зураг, тоо, шугам-г агуулсан зураг

AI-generated content may be incorrect.

**1.**CourseCreator

* **Purpose**: Represents users who create courses.
* **Fields**:
  + id (Long): Unique identifier.
  + name (String): Name of the creator.
  + email (String): Contact email.
  + courses (List<Course>): Courses created by this user.

**2.**Administrator

* **Purpose**: Manages platform content and approvals.
* **Fields**:
  + id (Long): Unique identifier.
  + name (String): Admin’s name.
  + email (String): Contact email.
  + approvedCourses (List<Course>): Courses approved by this admin.

**3.**Course

* **Purpose**: Represents a course on the platform.
* **Fields**:
  + id (Long): Unique identifier.
  + title (String): Course title.
  + description (String): Course overview.
  + imageUrl (String): Thumbnail/image link.
  + isRejected (boolean): True if the course is rejected during review.
  + approvedBy (Administrator): Admin who approved the course.
  + courseCreator (CourseCreator): Creator of the course.
  + lessons (List<Lesson>): Lessons included in the course.
  + learners (List<Learner>): Learners enrolled in the course.
  + completors (List<Learner>): Learners who completed the course.

**4.**Learner

* **Purpose**: Represents users enrolled in courses.
* **Fields**:
  + id (Long): Unique identifier.
  + name (String): Learner’s name.
  + email (String): Contact email.
  + enrolledCourses (List<Course>): Courses the learner is enrolled in.
  + completedLessons (List<Lesson>): Lessons marked as completed.
  + completedCourses (List<Course>): Courses fully completed.

**5.**Lesson

* **Purpose**: Represents a lesson within a course.
* **Fields**:
  + id (Long): Unique identifier.
  + title (String): Lesson title.
  + description (String): Lesson content summary.
  + imageUrl (String): Thumbnail/image link.
  + videoUrl (String): Link to the lesson video.
  + isRejected (boolean): True if the lesson is rejected during review.
  + approvedBy (Administrator): Admin who approved the lesson.
  + course (Course): Parent course of the lesson.
  + completors (List<Learner>): Learners who completed this lesson.

**Key Relationships**:

* **CourseCreator → Course**: A creator can create multiple courses (courses list).
* **Administrator → Course**: Admins approve courses (approvedCourses list).
* **Course ↔ Lesson**: A course contains multiple lessons (lessons list).
* **Learner ↔ Course**: Learners enroll in courses (enrolledCourses) and track completions (completedCourses).
* **Learner ↔ Lesson**: Learners track completed lessons (completedLessons).

# Future development

**1. User Experience (UX) Enhancements**

**a. Gamification**

* **Badges & Achievements**: Create custom badges (e.g., "Fast Learner" for finishing a lesson quickly, "Perfect Streak" for completing 5 lessons in a row).
* **Points System**: Award points for completing lessons, writing reviews, or helping others in forums. Let learners redeem points for perks (e.g., unlock premium content).
* **Leaderboards**: Display top learners per course or globally. Add filters (e.g., monthly rankings).
* **Progress Milestones**: Celebrate milestones (e.g., "50% Course Complete!" with confetti animations).

**b. Discussion Forums**

* **Course-Specific Threads**: Let learners ask questions or share notes directly within a course.
* **Upvote System**: Allow users to upvote helpful answers, with the best answers pinned at the top.
* **Instructor Participation**: Enable course creators to moderate forums and answer questions.
* **Notifications**: Alert users when their post receives a reply or upvote.

**c. Progress Visualization**

* **Interactive Dashboards**: Use libraries like **Chart.js** or **D3.js** to show progress as bar graphs, pie charts, or timelines.
* **Skill Maps**: Visualize skills gained from courses (e.g., "Python: Intermediate" after completing a Python course).
* **Personalized Insights**: Generate weekly/monthly progress reports (e.g., "You spent 8 hours learning this week!").

**d. Course Ratings & Reviews**

* **Star Ratings**: Let learners rate courses from 1 to 5 stars.
* **Detailed Reviews**: Include categories like "Content Quality," "Instructor Support," or "Difficulty."
* **Sort & Filter**: Allow users to sort courses by rating, popularity, or difficulty.
* **Instructor Responses**: Let course creators reply to reviews to address feedback.

**e. Personalization**

* **Learning Paths**: Recommend course sequences based on a learner’s goals (e.g., "Become a Web Developer: Start with HTML → CSS → JavaScript").
* **Dark Mode**: Add a toggle for a dark theme to reduce eye strain.

**2. Technical Enhancements**

**a. Mobile App Development**

* **Cross-Platform Tools**: Use **Flutter** or **React Native** to build iOS/Android apps.
* **Key Features**:
  + **Push Notifications**: Remind users about deadlines or new content (use **Firebase Cloud Messaging**).
  + **Synced Progress**: Ensure course progress is updated across devices in real-time.
  + **Offline Mode**: Cache lessons and quizzes for offline access (store data locally with **SQLite** or **Hive**).

**b. Search Improvements**

* **Advanced Filters**: Let users filter by duration (e.g., "Under 2 hours"), difficulty (Beginner/Advanced), or language.
* **Autocomplete**: Suggest course titles or topics as users type (use **Elasticsearch** or **Algolia**).
* **Search History**: Display recently searched terms for quick access.
* **Tag System**: Allow creators to add tags to courses (e.g., "#Python", "#WebDev") for better discovery.

**c. Notifications System**

* **Types of Alerts**:
  + Course reminders (e.g., "Your ‘Python Basics’ course is 50% incomplete!").
  + New lesson alerts (e.g., "A new lesson in ‘React 101’ is available!").
  + Forum updates (e.g., "Your question received 3 new answers!").
* **Customization**: Let users choose which notifications to receive (e.g., mute forum alerts).
* **Scheduling**: Avoid sending notifications during nighttime (use timezone detection).

**d. Offline Access**

* **Download Manager**: Allow learners to download entire courses or individual lessons.
  + Show download progress and storage usage.
* **Sync Mechanism**: Automatically update progress when the user reconnects to the internet.
* **Security**: Encrypt downloaded content to prevent unauthorized sharing.

**e. Performance Optimization**

* **Lazy Loading**: Load course images/videos only when the user scrolls to them (improves page speed).
* **Caching**: Cache frequently accessed data (e.g., course lists) using **Redis** or browser localStorage.
* **Code Splitting**: Break large frontend bundles into smaller chunks (e.g., with **Webpack**).