Java Project
Learning Management System (LMS)

Objective

The objective of the Learning Management System (LMS) is to manage courses, users, enrollments, assessments, and results efficiently. The system aims to facilitate learning by ensuring accurate tracking of user progress, reliable documentation of course content, and timely assessments. This will ultimately support effective learning and ensure users achieve their educational goals.

1. Role

- **Represents**: Different types of users in the LMS, such as Admin and User.
- **Attributes**:
  - `roleId`: A unique identifier for the role.
  - `roleName`: The name of the role (e.g., Admin, User).
- **Role**: Manages the different levels of access and permissions within the LMS. This is crucial for ensuring that users have appropriate access to resources based on their role.

2. User

- **Represents**: An individual who interacts with the LMS.
- **Attributes**:
  - `userId`: A unique identifier for the user.
  - `username`: The username of the user.
  - `passwordHash`: A hashed version of the user's password.
  - `fullName`: The full name of the user.

    - `email`: The email address of the user.
    - `roleId`: The role associated with the user (Admin or User).
- **Role**: Tracks personal information and role assignments for each user. This information is crucial for authentication and authorization within the LMS.

3. Course

- **Represents**: An educational course available within the LMS.
- **Attributes**:
    - `courseId`: A unique identifier for the course.
    - `courseName`: The name of the course.
    - `description`: A detailed description of the course content.
    - `startDate`: The date when the course starts.
    - `endDate`: The date when the course ends.
- **Role**: Manages the details of each course, including its schedule and content. This ensures that users have access to up-to-date educational materials.

4. Enrollment

- **Represents**: The relationship between a user and the courses they are enrolled in.
- **Attributes**:
    - `enrollmentId`: A unique identifier for the enrollment.
    - `userId`: The unique identifier of the user.
    - `courseId`: The unique identifier of the course.
    - `enrollmentDate`: The date on which the user enrolled in the course.
    - `status`: The current status of the enrollment (e.g., Enrolled, Completed).

- **Role**: Tracks which courses a user is enrolled in and their progress. This information is essential for managing course participation and completion.

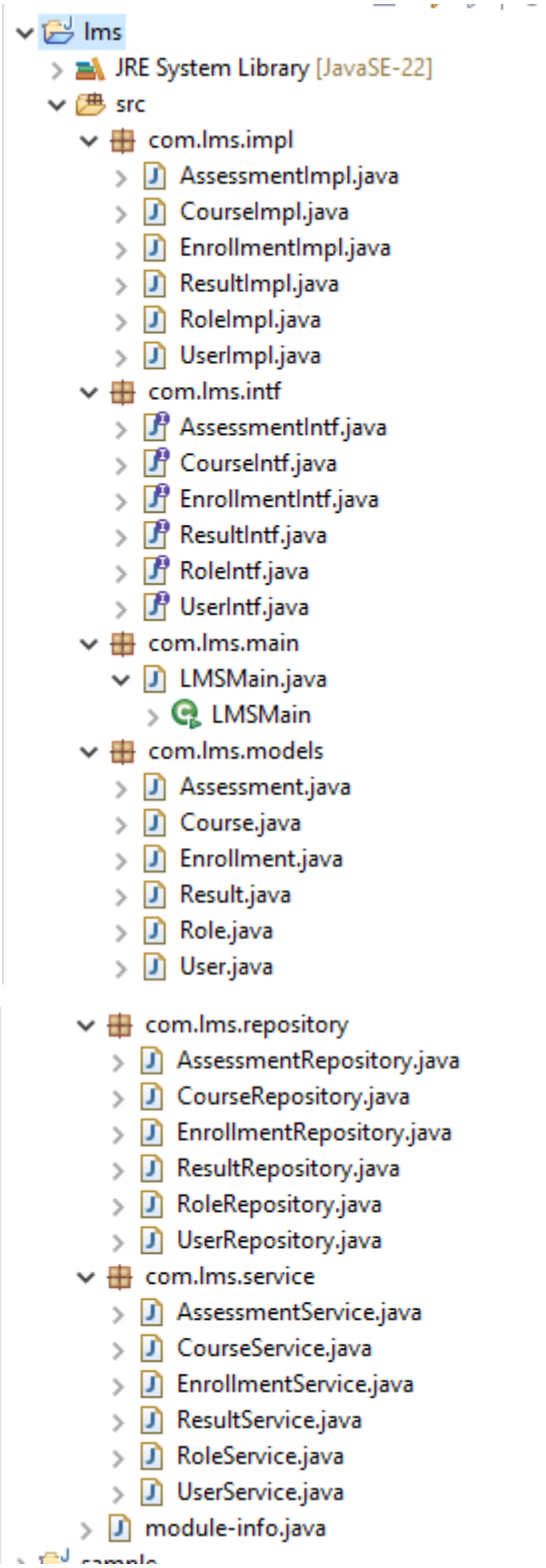
5. Assessment

- **Represents**: An evaluation or test associated with a course.
- **Attributes**:
  - `assessmentId`: A unique identifier for the assessment.
  - `courseId`: The unique identifier of the course associated with the assessment.
  - `assessmentName`: The name of the assessment.
  - `assessmentDate`: The date on which the assessment is to be conducted.
  - `maxScore`: The maximum score that can be achieved in the assessment.
- **Role**: Manages the details of assessments, ensuring that they are properly scheduled and associated with the correct courses.

6. Result

- **Represents**: The outcome of an assessment for a particular user.
- **Attributes**:
  - `resultId`: A unique identifier for the result.
  - `assessmentId`: The unique identifier of the assessment.
  - `userId`: The unique identifier of the user.
  - `score`: The score obtained by the user in the assessment.
- **Role**: Tracks the results of assessments, providing insights into user performance. This information is vital for evaluating the effectiveness of the course and the user's understanding of the material.

Project Structure:



LMS Implementation:

File Name : AssessmentImpl.java

```
package com.lms.impl;
import com.lms.intf.AssessmentIntf;
import com.lms.models.Assessment;
import java.util.ArrayList;
import java.util.List;
public class AssessmentImpl implements AssessmentIntf {
  private List<Assessment> assessments = new ArrayList<>();
  @Override
  public void addAssessment(Assessment assessment) {
    assessments.add(assessment);
  }
  @Override
  public Assessment getAssessment(int assessmentId) {
    return assessments.stream().filter(a -> a.getId() ==
assessmentId).findFirst().orElse(null);
  }
  @Override
  public List<Assessment> getAllAssessments() {
    return assessments;
  }
  @Override
  public void updateAssessment(Assessment assessment) {
    assessments.replaceAll(a -> a.getId() == assessment.getId() ? assessment : a);
  }
  @Override
  public void deleteAssessment(int assessmentId) {
    assessments.removeIf(a -> a.getId() == assessmentId);
  }
}
```

File Name : CourseImpl.java

```
package com.lms.impl;
import com.lms.intf.CourseIntf;
import com.lms.models.Course;
import java.util.ArrayList;
import java.util.List;
public class CourseImpl implements CourseIntf {
  private List<Course> courses = new ArrayList< >();
  @Override
  public void addCourse(Course course) {
    courses.add(course);
  }
  @Override
  public Course getCourse(int courseId) {
    return courses.stream().filter(c → c.getCourseId() ==
courseId).findFirst().orElse(null);
  }
  @Override
  public List<Course> getAllCourses() {
    return courses;
  }
  @Override
  public void updateCourse(Course course) {
    courses.replaceAll(c → c.getCourseId() == course.getCourseId() ? course : c);
  }
  @Override
  public void deleteCourse(int courseId) {
    courses.removeIf(c → c.getCourseId() == courseId);
  }
}
```

File Name : EnrollmentImpl.java

```
package com.lms.impl;
import com.lms.intf.EnrollmentIntf;
import com.lms.models.Enrollment;
import com.lms.repository.EnrollmentRepository;
import java.util.List;
public class EnrollmentImpl implements EnrollmentIntf {
  private EnrollmentRepository enrollmentRepo = new EnrollmentRepository();
  @Override
  public void addEnrollment(Enrollment enrollment) {
    enrollmentRepo.addEnrollment(enrollment);
  }
  @Override
  public Enrollment getEnrollment(int enrollmentId) {
    return enrollmentRepo.getEnrollment(enrollmentId);
  }
  @Override
  public List<Enrollment> getAllEnrollments() {
    return enrollmentRepo.getAllEnrollments();
  }
  @Override
  public void updateEnrollment(Enrollment enrollment) {
    enrollmentRepo.updateEnrollment(enrollment.getEnrollmentId(), enrollment);
  }
  @Override
  public void deleteEnrollment(int enrollmentId) {
    enrollmentRepo.deleteEnrollment(enrollmentId);
  }
}
```

File Name : ResultImpl.java

```
package com.lms.impl;
import com.lms.intf.ResultIntf;
import com.lms.models.Result;
import com.lms.repository.ResultRepository;
import java.util.List;
public class ResultImpl implements ResultIntf {
  private ResultRepository resultRepo = new ResultRepository();
  @Override
  public void addResult(Result result) {
    resultRepo.addResult(result);
  }
  @Override
  public Result getResult(int resultId) {
    return resultRepo.getResult(resultId);
  }
  @Override
  public List<Result> getAllResults() {
    return resultRepo.getAllResults();
  }
  @Override
  public void updateResult(Result result) {
    resultRepo.updateResult(result.getResultId(), result);
  }
  @Override
  public void deleteResult(int resultId) {
    resultRepo.deleteResult(resultId);
  }
}
```

File Name: RoleImpl.java

```
package com.lms.impl;
import com.lms.intf.RoleIntf;
import com.lms.models.Role;
import java.util.ArrayList;
import java.util.List;
public class RoleImpl implements RoleIntf {
  private List<Role> roles = new ArrayList<>();
  @Override
  public void addRole(Role role) {
    roles.add(role);
  }
  @Override
  public Role getRole(int roleId) {
    return roles.stream().filter(r -> r.getRoleId() == roleId).findFirst().orElse(null);
  }
  @Override
  public List<Role> getAllRoles() {
    return roles;
  }
  @Override
  public void updateRole(Role role) {
    roles.replaceAll(r -> r.getRoleId() == role.getRoleId() ? role : r);
  }
  @Override
  public void deleteRole(int roleId) {
    roles.removeIf(r -> r.getRoleId() == roleId);
  }
}
```

File Name : UserImpl.java

```
package com.lms.impl;
import com.lms.intf.UserIntf;
import com.lms.models.User;
import java.util.ArrayList;
import java.util.List;
public class UserImpl implements UserIntf {
  private List<User> users = new ArrayList< >();
  @Override
  public void addUser(User user) {
    users.add(user);
  }
  @Override
  public User getUser(int userId) {
    return users.stream().filter(u → u.getUserId() == userId).findFirst().orElse(null);
  }
  @Override
  public List<User> getAllUsers() {
    return users;
  }
  @Override
  public void updateUser(User user) {
    users.replaceAll(u → u.getUserId() == user.getUserId() ? user : u);
  }
  @Override
  public void deleteUser(int userId) {
    users.removeIf(u → u.getUserId() == userId);
  }
}
```

LMS Interfaces

File Name : AssessmentIntf.java

```
package com.lms.intf;
import com.lms.models.Assessment;
import java.util.List;
public interface AssessmentIntf {
  void addAssessment(Assessment assessment);
  Assessment getAssessment(int assessmentId);
  List<Assessment> getAllAssessments();
  void updateAssessment(Assessment assessment);
  void deleteAssessment(int assessmentId);
}
```

File Name: CourseIntf.java

```
package com.lms.intf;
import com.lms.models.Course;
import java.util.List;
public interface CourseIntf {
  void addCourse(Course course);
  Course getCourse(int courseId);
  List<Course> getAllCourses();
  void updateCourse(Course course);
  void deleteCourse(int courseId);
}
```

File Name : EnrollmentIntf.java

```
package com.lms.intf;
import com.lms.models.Enrollment;
import java.util.List;
public interface EnrollmentIntf {
  void addEnrollment(Enrollment enrollment);
  Enrollment getEnrollment(int enrollmentId);
  List<Enrollment> getAllEnrollments();
  void updateEnrollment(Enrollment enrollment);
  void deleteEnrollment(int enrollmentId);
}
```

File Name : ResultIntf.java

```
package com.lms.intf;
import com.lms.models.Result;
import java.util.List;
public interface ResultIntf {
  void addResult(Result result);
  Result getResult(int resultId);
  List<Result> getAllResults();
  void updateResult(Result result);
  void deleteResult(int resultId);
}
```

File Name : RoleIntf.java

```
package com.lms.intf;
import com.lms.models.Role;
import java.util.List;
public interface RoleIntf {
  void addRole(Role role);
  Role getRole(int roleId);
  List<Role> getAllRoles();
  void updateRole(Role role);
  void deleteRole(int roleId);
}
```

File Name : UserIntf.java

```
package com.lms.intf;
import com.lms.models.User;
import java.util.List;
public interface UserIntf {
  void addUser(User user);
  User getUser(int userId);
  List<User> getAllUsers();
  void updateUser(User user);
  void deleteUser(int userId);
}
```

LMS Models:

File Name : Assessment.java

```
package com.lms.models;
public class Assessment {
  private int id;
  private int courseId;
  private String name;
  private String date;
  private int maxScore;
  // Constructor
  public Assessment(int id, int courseId, String name, String date, int maxScore)
{
    this.id = id;
    this.courseId = courseId;
    this.name = name;
    this.date = date;
    this.maxScore = maxScore;
  }
  // Getters and setters
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public int getCourseId() {
    return courseId;
  }
  public void setCourseId(int courseId) {
    this.courseId = courseId;
  }
  public String getName() {
    return name;
```

```
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDate() {
        return date;
    }
    public void setDate(String date) {
        this.date = date;
    }
    public int getMaxScore() {
        return maxScore;
    }
    public void setMaxScore(int maxScore) {
        this.maxScore = maxScore;
    }
    @Override
    public String toString() {
        return "Assessment{" +
                "id=" + id +
                ", courseId=" + courseId +
                ", name='" + name + '\'' +
                ", date='" + date + '\'' +
                ", maxScore=" + maxScore +
                '}';
    }
}
```

File Name : Course.java

```
package com.lms.models;
public class Course {
  private int courseId;
  private String courseName;
  private String description;
  private String startDate;
  private String endDate;
  // Constructor
  public Course(int courseId, String courseName, String description, String
startDate, String endDate) {
    this.courseId = courseId;
    this.courseName = courseName;
    this.description = description;
    this.startDate = startDate;
    this.endDate = endDate;
  }
  // Getters and Setters
  public int getCourseId() {
    return courseId;
  }
  public void setCourseId(int courseId) {
    this.courseId = courseId;
  }
  public String getCourseName() {
    return courseName;
  }
  public void setCourseName(String courseName) {
    this.courseName = courseName;
  }
  public String getDescription() {
    return description;
```

```
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getStartDate() {
        return startDate;
    }
    public void setStartDate(String startDate) {
        this.startDate = startDate;
    }
    public String getEndDate() {
        return endDate;
    }
    public void setEndDate(String endDate) {
        this.endDate = endDate;
    }
    @Override
    public String toString() {
        return "Course{" +
                "courseId=" + courseId +
                ", courseName='" + courseName + '\'' +
                ", description='" + description + '\'' +
                ", startDate='" + startDate + '\'' +
                ", endDate='" + endDate + '\'' +
                '}';
    }
}
```

File Name: Enrollment.java

```
package com.lms.models;
public class Enrollment {
  private int enrollmentId;
  private int userId;
  private int courseId;
  private String status;
  // Constructor
  public Enrollment(int enrollmentId, int userId, int courseId, String status) {
    this.enrollmentId = enrollmentId;
    this.userId = userId;
    this.courseId = courseId;
    this.status = status;
  }
  // Getters and setters
  public int getEnrollmentId() {
    return enrollmentId;
  }
  public void setEnrollmentId(int enrollmentId) {
    this.enrollmentId = enrollmentId;
  }
  public int getUserId() {
    return userId;
  }
  public void setUserId(int userId) {
    this.userId = userId;
  }
  public int getCourseId() {
    return courseId;
  }
  public void setCourseId(int courseId) {
    this.courseId = courseId;
```

```
  }
  public String getStatus() {
    return status;
  }
  public void setStatus(String status) {
    this.status = status;
  }
  @Override
  public String toString() {
    return "Enrollment{" +
        "enrollmentId=" + enrollmentId +
        ", userId=" + userId +
        ", courseId=" + courseId +
        ", status='" + status + '\'' +
        '}';
  }
}
```

File Name : Result.java

```
package com.lms.models;
public class Result {
  private int resultId;
  private int assessmentId;
  private int userId;
  private int score;
  // Constructor
  public Result(int resultId, int assessmentId, int userId, int score) {
    this.resultId = resultId;
    this.assessmentId = assessmentId;
    this.userId = userId;
    this.score = score;
  }
  // Getters and Setters
```

```
  public int getResultId() {
    return resultId;
  }
  public void setResultId(int resultId) {
    this.resultId = resultId;
  }
  public int getAssessmentId() {
    return assessmentId;
  }
  public void setAssessmentId(int assessmentId) {
    this.assessmentId = assessmentId;
  }
  public int getUserId() {
    return userId;
  }
  public void setUserId(int userId) {
    this.userId = userId;
  }
  public int getScore() {
    return score;
  }
  public void setScore(int score) {
    this.score = score;
  }
  @Override
  public String toString() {
    return "Result{" +
        "resultId=" + resultId +
        ", assessmentId=" + assessmentId +
        ", userId=" + userId +
        ", score=" + score +
        '}';
  }
}
```

File Name : Role.java

```java
package com.lms.models;
public class Role {
  private int roleId;
  private String roleName;
  // Constructor
  public Role(int roleId, String roleName) {
    this.roleId = roleId;
    this.roleName = roleName;
  }
  // Getters and Setters
  public int getRoleId() {
    return roleId;
  }
  public void setRoleId(int roleId) {
    this.roleId = roleId;
  }
  public String getRoleName() {
    return roleName;
  }
  public void setRoleName(String roleName) {
    this.roleName = roleName;
  }
  @Override
  public String toString() {
    return "Role{" +
        "roleId=" + roleId +
        ", roleName='" + roleName + '\'' +
        '}';
  }
}
```

File Name : User.java

```java
package com.lms.models;
public class User {
  private int userId;
  private String username;
  private String email;
  private Role role;
  // Constructor
  public User(int userId, String username, String email, Role role) {
    this.userId = userId;
    this.username = username;
    this.email = email;
    this.role = role;
  }
  // Getters and Setters
  public int getUserId() {
    return userId;
  }
  public void setUserId(int userId) {
    this.userId = userId;
  }
  public String getUsername() {
    return username;
  }
  public void setUsername(String username) {
    this.username = username;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
```

```
  public Role getRole() {
    return role;
  }
  public void setRole(Role role) {
    this.role = role;
  }
  @Override
  public String toString() {
    return "User{" +
        "userId=" + userId +
        ", username='" + username + '\'' +
        ", email='" + email + '\'' +
        ", role=" + role +
        '}';
  }
}
```

LMS Repository

File Name: AssessmentRepository.java

```
package com.lms.repository;
import com.lms.models.Assessment;
import java.util.ArrayList;
import java.util.List;
public class AssessmentRepository {
  private List<Assessment> assessments = new ArrayList<>();
  // Add sample data
  public AssessmentRepository() {
    // Adding some sample assessments
    assessments.add(new Assessment(1, 1, "Python Basics Quiz", "2024-11-15",
100));
    assessments.add(new Assessment(2, 2, "JavaScript Project", "2024-12-01",
150));
```

```
    assessments.add(new Assessment(3, 3, "Java Fundamentals Test",
"2024-10-15", 120));
    assessments.add(new Assessment(4, 4, "Ruby on Rails Assignment",
"2024-12-15", 200));
  }
  public void addAssessment(Assessment assessment) {
    assessments.add(assessment);
  }
  public void updateAssessment(int assessmentId, Assessment
updatedAssessment) {
    for (int i = 0; i < assessments.size(); i++) {
      if (assessments.get(i).getId() == assessmentId) {
        assessments.set(i, updatedAssessment);
        break;
      }
    }
  }
  public void deleteAssessment(int assessmentId) {
    assessments.removeIf(assessment → assessment.getId() ==
assessmentId);
  }
  public List<Assessment> getAllAssessments() {
    return new ArrayList<>(assessments);
  }
}
```

File Name : CourseRepository.java

```
package com.lms.repository;
import com.lms.models.Course;
import java.util.ArrayList;
import java.util.List;
public class CourseRepository {
  private List<Course> courses = new ArrayList<>();
  public void addCourse(Course course) {
```

```
    courses.add(course);
  }
  public void updateCourse(int courseId, Course updatedCourse) {
    for (int i = 0; i < courses.size(); i++) {
      if (courses.get(i).getCourseId() == courseId) {
        courses.set(i, updatedCourse);
        break;
      }
    }
  }
  public void deleteCourse(int courseId) {
    courses.removeIf(course -> course.getCourseId() == courseId);
  }
  public List<Course> getAllCourses() {
    return new ArrayList<>(courses);
  }
}
```

File Name : EnrollmentRepository.java

```
package com.lms.repository;
import com.lms.models.Enrollment;
import java.util.ArrayList;
import java.util.List;
public class EnrollmentRepository {
  private List<Enrollment> enrollments = new ArrayList<>();
  public void addEnrollment(Enrollment enrollment) {
    enrollments.add(enrollment);
  }
  public Enrollment getEnrollment(int enrollmentId) {
    return enrollments.stream()
        .filter(e -> e.getEnrollmentId() == enrollmentId)
        .findFirst()
        .orElse(null);
```

```
  }
  public List<Enrollment> getAllEnrollments() {
    return new ArrayList<>(enrollments);
  }
  public void updateEnrollment(int enrollmentId, Enrollment updatedEnrollment)
{
    for (int i = 0; i < enrollments.size(); i++) {
      if (enrollments.get(i).getEnrollmentId() == enrollmentId) {
        enrollments.set(i, updatedEnrollment);
        break;
      }
    }
  }
  public void deleteEnrollment(int enrollmentId) {
    enrollments.removeIf(e -> e.getEnrollmentId() == enrollmentId);
  }
}
```

File Name : ResultRepository.java

```
package com.lms.repository;
import com.lms.models.Result;
import java.util.ArrayList;
import java.util.List;
public class ResultRepository {
  private List<Result> results = new ArrayList<>();
  public void addResult(Result result) {
    results.add(result);
  }
  public Result getResult(int resultId) {
    return results.stream()
        .filter(r -> r.getResultId() == resultId)
        .findFirst()
        .orElse(null);
```

```
  }
  public List<Result> getAllResults() {
    return new ArrayList<>(results);
  }
  public void updateResult(int resultId, Result updatedResult) {
    for (int i = 0; i < results.size(); i++) {
      if (results.get(i).getResultId() == resultId) {
        results.set(i, updatedResult);
        break;
      }
    }
  }
  public void deleteResult(int resultId) {
    results.removeIf(r -> r.getResultId() == resultId);
  }
}
```

File Name : RoleRepository.java

```
package com.lms.repository;
import com.lms.models.Role;
import java.util.ArrayList;
import java.util.List;
public class RoleRepository {
  private List<Role> roles = new ArrayList<>();
  public void addRole(Role role) {
    roles.add(role);
  }
  public void updateRole(int roleId, Role updatedRole) {
    for (int i = 0; i < roles.size(); i++) {
      if (roles.get(i).getRoleId() == roleId) {
        roles.set(i, updatedRole);
        break;
      }
```

```
    }
  }
  public void deleteRole(int roleId) {
    roles.removeIf(role -> role.getRoleId() == roleId);
  }
  public List<Role> getAllRoles() {
    return new ArrayList<>(roles);
  }
}
```

File Name : UserRepository.java

```
package com.lms.repository;
import com.lms.models.User;
import java.util.ArrayList;
import java.util.List;
public class UserRepository {
  private List<User> users = new ArrayList<>();
  public void addUser(User user) {
    users.add(user);
  }
  public void updateUser(int userId, User updatedUser) {
    for (int i = 0; i < users.size(); i++) {
      if (users.get(i).getUserId() == userId) {
        users.set(i, updatedUser);
        break;
      }
    }
  }
  public void deleteUser(int userId) {
    users.removeIf(user -> user.getUserId() == userId);
  }
  public List<User> getAllUsers() {
```

```
    return new ArrayList<>(users);
  }
}
```

LMS Service:

File Name: AssessmentService.java

```
package com.lms.service;
import com.lms.models.Assessment;
import com.lms.repository.AssessmentRepository;
import java.util.List;
public class AssessmentService {
  private AssessmentRepository assessmentRepository;
  public AssessmentService() {
    this.assessmentRepository = new AssessmentRepository();
  }
  public void addAssessment(Assessment assessment) {
    assessmentRepository.addAssessment(assessment);
  }
  public void updateAssessment(int assessmentId, Assessment
updatedAssessment) {
    assessmentRepository.updateAssessment(assessmentId,
updatedAssessment);
  }
  public void deleteAssessment(int assessmentId) {
    assessmentRepository.deleteAssessment(assessmentId);
  }
  public List<Assessment> getAllAssessments() {
    return assessmentRepository.getAllAssessments();
  }
}
```

File Name : CourseService.java

```
package com.lms.service;
import com.lms.models.Course;
import com.lms.repository.CourseRepository;
import java.util.List;
public class CourseService {
  private CourseRepository courseRepository;
  public CourseService() {
    this.courseRepository = new CourseRepository();
  }
  public void addCourse(Course course) {
    courseRepository.addCourse(course);
  }
  public void updateCourse(int courseId, Course updatedCourse) {
    courseRepository.updateCourse(courseId, updatedCourse);
  }
  public void deleteCourse(int courseId) {
    courseRepository.deleteCourse(courseId);
  }
  public List<Course> getAllCourses() {
    return courseRepository.getAllCourses();
  }
}
```

FileName : EnrollmentService.java

```
package com.lms.service;
import com.lms.models.Enrollment;
import com.lms.repository.EnrollmentRepository;
import java.util.List;
public class EnrollmentService {
```

```
  private EnrollmentRepository enrollmentRepository;
  public EnrollmentService() {
    this.enrollmentRepository = new EnrollmentRepository();
  }
  public void addEnrollment(Enrollment enrollment) {
    enrollmentRepository.addEnrollment(enrollment);
  }
  public void updateEnrollment(int enrollmentId, Enrollment updatedEnrollment)
{
    enrollmentRepository.updateEnrollment(enrollmentId, updatedEnrollment);
  }
  public void deleteEnrollment(int enrollmentId) {
    enrollmentRepository.deleteEnrollment(enrollmentId);
  }
  public List<Enrollment> getAllEnrollments() {
    return enrollmentRepository.getAllEnrollments();
  }
}
```

FileName : ResultService.java

```
package com.lms.service;
import com.lms.models.Result;
import com.lms.repository.ResultRepository;
import java.util.List;
public class ResultService {
  private ResultRepository resultRepository;
  public ResultService() {
    this.resultRepository = new ResultRepository();
  }
  public void addResult(Result result) {
    resultRepository.addResult(result);
  }
```

```
  public void updateResult(int resultId, Result updatedResult) {
    resultRepository.updateResult(resultId, updatedResult);
  }
  public void deleteResult(int resultId) {
    resultRepository.deleteResult(resultId);
  }
  public List<Result> getAllResults() {
    return resultRepository.getAllResults();
  }
}
```

FileName : RoleService.java

```
package com.lms.service;
import com.lms.models.Role;
import com.lms.repository.RoleRepository;
import java.util.List;
public class RoleService {
  private RoleRepository roleRepository;
  public RoleService() {
    this.roleRepository = new RoleRepository();
  }
  public void addRole(Role role) {
    roleRepository.addRole(role);
  }
  public void updateRole(int roleId, Role updatedRole) {
    roleRepository.updateRole(roleId, updatedRole);
  }
  public void deleteRole(int roleId) {
    roleRepository.deleteRole(roleId);
  }
  public List<Role> getAllRoles() {
    return roleRepository.getAllRoles();
  }}
```

FileName : UserService.java

```
package com.lms.service;
import com.lms.models.User;
import com.lms.repository.UserRepository;
import java.util.List;
public class UserService {
  private UserRepository userRepository;
  public UserService() {
    this.userRepository = new UserRepository();
  }
  public void addUser(User user) {
    userRepository.addUser(user);
  }
  public void updateUser(int userId, User updatedUser) {
    userRepository.updateUser(userId, updatedUser);
  }
  public void deleteUser(int userId) {
    userRepository.deleteUser(userId);
  }
  public List<User> getAllUsers() {
    return userRepository.getAllUsers();
  }
}
```

LMS Main

File Name : LMSMain.java

```
package com.lms.main;
import com.lms.models.*;
import com.lms.service.*;
import java.util.Scanner;
public class LMSMain {
  public static void main(String[] args) {
    UserService userService = new UserService();
    RoleService roleService = new RoleService();
    CourseService courseService = new CourseService();
    EnrollmentService enrollmentService = new EnrollmentService();
    AssessmentService assessmentService = new AssessmentService();
    ResultService resultService = new ResultService();
    Scanner scanner = new Scanner(System.in);
    int choice = -1;
    while (choice != 0) {
      System.out.println("Learning Management System");
      System.out.println("1. Manage Users");
      System.out.println("2. Manage Roles");
      System.out.println("3. Manage Courses");
      System.out.println("4. Manage Enrollments");
      System.out.println("5. Manage Assessments");
      System.out.println("6. Manage Results");
      System.out.println("0. Exit");
      System.out.print("Enter your choice: ");
      choice = scanner.nextInt();
      scanner.nextLine(); // Consume newline
      switch (choice) {
        case 1:
          manageUsers(scanner, userService);
          break;
```

```
            case 2:
                manageRoles(scanner, roleService);
                break;
            case 3:
                manageCourses(scanner, courseService);
                break;
            case 4:
                manageEnrollments(scanner, enrollmentService);
                break;
            case 5:
                manageAssessments(scanner, assessmentService);
                break;
            case 6:
                manageResults(scanner, resultService);
                break;
            case 0:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
    scanner.close();
}
// User Management
private static void manageUsers(Scanner scanner, UserService userService) {
    int choice = -1;
    while (choice != 0) {
        System.out.println("User Management");
        System.out.println("1. Add User");
        System.out.println("2. Update User");
        System.out.println("3. Delete User");
        System.out.println("4. List Users");
        System.out.println("0. Back to Main Menu");
```

```
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        switch (choice) {
          case 1:
            System.out.print("Enter User ID: ");
            int userId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Username: ");
            String username = scanner.nextLine();
            System.out.print("Enter Email: ");
            String email = scanner.nextLine();
            System.out.print("Enter Role ID: ");
            int roleId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            Role role = new Role(roleId, "Role Name");
            User user = new User(userId, username, email, role);
            userService.addUser(user);
            System.out.println("User added successfully.");
            break;
          case 2:
            System.out.print("Enter User ID to update: ");
            int updateUserId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new Username: ");
            String newUsername = scanner.nextLine();
            System.out.print("Enter new Email: ");
            String newEmail = scanner.nextLine();
            System.out.print("Enter new Role ID: ");
            int newRoleId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            Role newRole = new Role(newRoleId, "New Role Name");
            User updatedUser = new User(updateUserId, newUsername, newEmail,
newRole);
```

```
            userService.updateUser(updateUserId, updatedUser);
            System.out.println("User updated successfully.");
            break;
        case 3:
            System.out.print("Enter User ID to delete: ");
            int deleteUserId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            userService.deleteUser(deleteUserId);
            System.out.println("User deleted successfully.");
            break;
        case 4:
            System.out.println("List of Users:");
            for (User u : userService.getAllUsers()) {
                System.out.println(u);
            }
            break;
        case 0:
            System.out.println("Returning to Main Menu...");
            break;
        default:
            System.out.println("Invalid choice. Try again.");
    }
  }
}
// Role Management
private static void manageRoles(Scanner scanner, RoleService roleService) {
  int choice = -1;
  while (choice != 0) {
    System.out.println("Role Management");
    System.out.println("1. Add Role");
    System.out.println("2. Update Role");
    System.out.println("3. Delete Role");
    System.out.println("4. List Roles");
    System.out.println("0. Back to Main Menu");
```

```
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
scanner.nextLine(); // Consume newline
switch (choice) {
  case 1:
    System.out.print("Enter Role ID: ");
    int roleId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.print("Enter Role Name: ");
    String roleName = scanner.nextLine();
    Role role = new Role(roleId, roleName);
    roleService.addRole(role);
    System.out.println("Role added successfully.");
    break;
  case 2:
    System.out.print("Enter Role ID to update: ");
    int updateRoleId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.print("Enter new Role Name: ");
    String newRoleName = scanner.nextLine();
    Role updatedRole = new Role(updateRoleId, newRoleName);
    roleService.updateRole(updateRoleId, updatedRole);
    System.out.println("Role updated successfully.");
    break;
  case 3:
    System.out.print("Enter Role ID to delete: ");
    int deleteRoleId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    roleService.deleteRole(deleteRoleId);
    System.out.println("Role deleted successfully.");
    break;
  case 4:
    System.out.println("List of Roles:");
    for (Role r : roleService.getAllRoles()) {
```

```
            System.out.println(r);
          }
          break;
        case 0:
          System.out.println("Returning to Main Menu...");
          break;
        default:
          System.out.println("Invalid choice. Try again.");
      }
    }
  }
  // Course Management
  private static void manageCourses(Scanner scanner, CourseService
courseService) {
    int choice = -1;
    while (choice != 0) {
      System.out.println("Course Management");
      System.out.println("1. Add Course");
      System.out.println("2. Update Course");
      System.out.println("3. Delete Course");
      System.out.println("4. List Courses");
      System.out.println("0. Back to Main Menu");
      System.out.print("Enter your choice: ");
      choice = scanner.nextInt();
      scanner.nextLine(); // Consume newline
      switch (choice) {
        case 1:
          System.out.print("Enter Course ID: ");
          int courseId = scanner.nextInt();
          scanner.nextLine(); // Consume newline
          System.out.print("Enter Course Name: ");
          String courseName = scanner.nextLine();
          System.out.print("Enter Description: ");
          String description = scanner.nextLine();
```

```
            System.out.print("Enter Start Date (YYYY-MM-DD): ");
            String startDate = scanner.nextLine();
            System.out.print("Enter End Date (YYYY-MM-DD): ");
            String endDate = scanner.nextLine();
            Course course = new Course(courseId, courseName, description,
startDate, endDate);
            courseService.addCourse(course);
            System.out.println("Course added successfully.");
            break;
        case 2:
            System.out.print("Enter Course ID to update: ");
            int updateCourseId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new Course Name: ");
            String newCourseName = scanner.nextLine();
            System.out.print("Enter new Description: ");
            String newDescription = scanner.nextLine();
            System.out.print("Enter new Start Date (YYYY-MM-DD): ");
            String newStartDate = scanner.nextLine();
            System.out.print("Enter new End Date (YYYY-MM-DD): ");
            String newEndDate = scanner.nextLine();
            Course updatedCourse = new Course(updateCourseId,
newCourseName, newDescription, newStartDate, newEndDate);
            courseService.updateCourse(updateCourseId, updatedCourse);
            System.out.println("Course updated successfully.");
            break;
        case 3:
            System.out.print("Enter Course ID to delete: ");
            int deleteCourseId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            courseService.deleteCourse(deleteCourseId);
            System.out.println("Course deleted successfully.");
            break;
        case 4:
```

```
            System.out.println("List of Courses:");
            for (Course c : courseService.getAllCourses()) {
              System.out.println(c);
            }
            break;
          case 0:
            System.out.println("Returning to Main Menu...");
            break;
          default:
            System.out.println("Invalid choice. Try again.");
        }
      }
    }
    // Enrollment Management
    private static void manageEnrollments(Scanner scanner, EnrollmentService
enrollmentService) {
      int choice = -1;
      while (choice != 0) {
        System.out.println("Enrollment Management");
        System.out.println("1. Add Enrollment");
        System.out.println("2. Update Enrollment");
        System.out.println("3. Delete Enrollment");
        System.out.println("4. List Enrollments");
        System.out.println("0. Back to Main Menu");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        switch (choice) {
          case 1:
            System.out.print("Enter Enrollment ID: ");
            int enrollmentId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter User ID: ");
            int userId = scanner.nextInt();
```

```
            System.out.print("Enter Course ID: ");
            int courseId = scanner.nextInt();
            System.out.print("Enter Enrollment Date (YYYY-MM-DD): ");
            String enrollmentDate = scanner.nextLine();
            Enrollment enrollment = new Enrollment(enrollmentId, userId, courseId,
enrollmentDate);
            enrollmentService.addEnrollment(enrollment);
            System.out.println("Enrollment added successfully.");
            break;
        case 2:
            System.out.print("Enter Enrollment ID to update: ");
            int updateEnrollmentId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new User ID: ");
            int newUserId = scanner.nextInt();
            System.out.print("Enter new Course ID: ");
            int newCourseId = scanner.nextInt();
            System.out.print("Enter new Enrollment Date (YYYY-MM-DD): ");
            String newEnrollmentDate = scanner.nextLine();
            Enrollment updatedEnrollment = new Enrollment(updateEnrollmentId,
newUserId, newCourseId, newEnrollmentDate);
            enrollmentService.updateEnrollment(updateEnrollmentId,
updatedEnrollment);
            System.out.println("Enrollment updated successfully.");
            break;
        case 3:
            System.out.print("Enter Enrollment ID to delete: ");
            int deleteEnrollmentId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            enrollmentService.deleteEnrollment(deleteEnrollmentId);
            System.out.println("Enrollment deleted successfully.");
            break;
        case 4:
            System.out.println("List of Enrollments:");
```

```
            for (Enrollment e : enrollmentService.getAllEnrollments()) {
              System.out.println(e);
            }
            break;
          case 0:
            System.out.println("Returning to Main Menu...");
            break;
          default:
            System.out.println("Invalid choice. Try again.");
        }
      }
    }
    // Assessment Management
    private static void manageAssessments(Scanner scanner, AssessmentService
assessmentService) {
      int choice = -1;
      while (choice != 0) {
        System.out.println("Assessment Management");
        System.out.println("1. Add Assessment");
        System.out.println("2. Update Assessment");
        System.out.println("3. Delete Assessment");
        System.out.println("4. List Assessments");
        System.out.println("0. Back to Main Menu");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        switch (choice) {
          case 1:
            System.out.print("Enter Assessment ID: ");
            int assessmentId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Course ID: ");
            int courseId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
```

```
            System.out.print("Enter Assessment Name: ");
            String assessmentName = scanner.nextLine();
            System.out.print("Enter Assessment Date (YYYY-MM-DD): ");
            String assessmentDate = scanner.nextLine();
            System.out.print("Enter Max Score: ");
            int maxScore = scanner.nextInt();
            Assessment assessment = new Assessment(assessmentId, courseId,
assessmentName, assessmentDate, maxScore);
            assessmentService.addAssessment(assessment);
            System.out.println("Assessment added successfully.");
            break;
        case 2:
            System.out.print("Enter Assessment ID to update: ");
            int updateAssessmentId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new Course ID: ");
            int newCourseId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new Assessment Name: ");
            String newAssessmentName = scanner.nextLine();
            System.out.print("Enter new Assessment Date (YYYY-MM-DD): ");
            String newAssessmentDate = scanner.nextLine();
            System.out.print("Enter new Max Score: ");
            int newMaxScore = scanner.nextInt();
            Assessment updatedAssessment = new
Assessment(updateAssessmentId, newCourseId, newAssessmentName,
newAssessmentDate, newMaxScore);
            assessmentService.updateAssessment(updateAssessmentId,
updatedAssessment);
            System.out.println("Assessment updated successfully.");
            break;
        case 3:
            System.out.print("Enter Assessment ID to delete: ");
            int deleteAssessmentId = scanner.nextInt();
```

```
            scanner.nextLine(); // Consume newline
            assessmentService.deleteAssessment(deleteAssessmentId);
            System.out.println("Assessment deleted successfully.");
            break;
          case 4:
            System.out.println("List of Assessments:");
            for (Assessment a : assessmentService.getAllAssessments()) {
              System.out.println(a);
            }
            break;
          case 0:
            System.out.println("Returning to Main Menu...");
            break;
          default:
            System.out.println("Invalid choice. Try again.");
        }
      }
    }
    // Result Management
    private static void manageResults(Scanner scanner, ResultService
resultService) {
      int choice = -1;
      while (choice != 0) {
        System.out.println("Result Management");
        System.out.println("1. Add Result");
        System.out.println("2. Update Result");
        System.out.println("3. Delete Result");
        System.out.println("4. List Results");
        System.out.println("0. Back to Main Menu");
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        switch (choice) {
          case 1:
```

```
            System.out.print("Enter Result ID: ");
            int resultId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Assessment ID: ");
            int assessmentId = scanner.nextInt();
            System.out.print("Enter User ID: ");
            int userId = scanner.nextInt();
            System.out.print("Enter Score: ");
            int score = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            Result result = new Result(resultId, assessmentId, userId, score);
            resultService.addResult(result);
            System.out.println("Result added successfully.");
            break;
        case 2:
            System.out.print("Enter Result ID to update: ");
            int updateResultId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter new Assessment ID: ");
            int newAssessmentId = scanner.nextInt();
            System.out.print("Enter new User ID: ");
            int newUserId = scanner.nextInt();
            System.out.print("Enter new Score: ");
            int newScore = scanner.nextInt();
            Result updatedResult = new Result(updateResultId, newAssessmentId,
newUserId, newScore);
            resultService.updateResult(updateResultId, updatedResult);
            System.out.println("Result updated successfully.");
            break;
        case 3:
            System.out.print("Enter Result ID to delete: ");
            int deleteResultId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            resultService.deleteResult(deleteResultId);
```

```
            System.out.println("Result deleted successfully.");
            break;
          case 4:
            System.out.println("List of Results:");
            for (Result r : resultService.getAllResults()) {
              System.out.println(r);
            }
            break;
          case 0:
            System.out.println("Returning to Main Menu...");
            break;
          default:
            System.out.println("Invalid choice. Try again.");
        }
      }
    }
}
```

Output:

Users(Add,List)

```
Learning Management System
1. Manage Users
2. Manage Roles
3. Manage Courses
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 1
User Management
1. Add User
2. Update User
3. Delete User
4. List Users
0. Back to Main Menu
Enter your choice: 1
Enter User ID: 1
Enter Username: dhanapal
Enter Email: dhanapal.m@payoda.com
Enter Role ID: 1
User added successfully.
User Management
1. Add User
2. Update User
3. Delete User
4. List Users
0. Back to Main Menu
Enter your choice: 4
List of Users:
User{userId=1, username='dhanapal', email='dhanapal.m@payoda.com', role=Role{roleId=1, roleName='Role Name'}}
User Management
1. Add User
2. Update User
3. Delete User
4. List Users
0. Back to Main Menu
Enter your choice:
```

Roles(Add,List)

```
Learning Management System
1. Manage Users
2. Manage Roles
3. Manage Courses
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 2
Role Management
1. Add Role
2. Update Role
3. Delete Role
4. List Roles
0. Back to Main Menu
Enter your choice: 1
Enter Role ID: 1
Enter Role Name: Admin
Role added successfully.
Role Management
1. Add Role
2. Update Role
3. Delete Role
4. List Roles
0. Back to Main Menu
Enter your choice: 4
List of Roles:
Role{roleId=1, roleName='Admin'}
Role Management
1. Add Role
2. Update Role
3. Delete Role
4. List Roles
0. Back to Main Menu
Enter your choice:
```

Course(Add,List)

```
Learning Management System
1. Manage Users
2. Manage Roles
3. Manage Courses
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 3
Course Management
1. Add Course
2. Update Course
3. Delete Course
4. List Courses
0. Back to Main Menu
Enter your choice: 1
Enter Course ID: 1
Enter Course Name: Java
Enter Description: Java basics
Enter Start Date (YYYY-MM-DD): 2024-08-11
Enter End Date (YYYY-MM-DD): 2023-08-21
Course added successfully.
Course Management
1. Add Course
2. Update Course
3. Delete Course
4. List Courses
0. Back to Main Menu
Enter your choice: 4
List of Courses:
Course{courseId=1, courseName='Java', description='Java basics', startDate='2024-08-11', endDate='2023-08-21'}
Course Management
1. Add Course
2. Update Course
3. Delete Course
4. List Courses
0. Back to Main Menu
Enter your choice:
```

Enrollments(Add,List)

```
Learning Management System
1. Manage Users
2. Manage Roles
3. Manage Courses
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 4
Enrollment Management
1. Add Enrollment
2. Update Enrollment
3. Delete Enrollment
4. List Enrollments
0. Back to Main Menu
Enter your choice: 1
Enter Enrollment ID: 1
Enter User ID: 1
Enter Course ID: 1
Enter Enrollment Date (YYYY-MM-DD): Enrollment added successfully.
Enrollment Management
1. Add Enrollment
2. Update Enrollment
3. Delete Enrollment
4. List Enrollments
0. Back to Main Menu
Enter your choice: 4
List of Enrollments:
Enrollment{enrollmentId=1, userId=1, courseId=1, status=''}
Enrollment Management
1. Add Enrollment
2. Update Enrollment
3. Delete Enrollment
4. List Enrollments
0. Back to Main Menu
Enter your choice:
```

Assessment(Add,List)

```
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 5
Assessment Management
1. Add Assessment
2. Update Assessment
3. Delete Assessment
4. List Assessments
0. Back to Main Menu
Enter your choice: 1
Enter Assessment ID: 1
Enter Course ID: 1
Enter Assessment Name: Test1
Enter Assessment Date (YYYY-MM-DD): 2024-08-08
Enter Max Score: 100
Assessment added successfully.
Assessment Management
1. Add Assessment
2. Update Assessment
3. Delete Assessment
4. List Assessments
0. Back to Main Menu
Enter your choice: 4
List of Assessments:
Assessment{id=1, courseId=1, name='Python Basics Quiz', date='2024-11-15', maxScore=100}
Assessment{id=2, courseId=2, name='JavaScript Project', date='2024-12-01', maxScore=150}
Assessment{id=3, courseId=3, name='Java Fundamentals Test', date='2024-10-15', maxScore=120}
Assessment{id=4, courseId=4, name='Ruby on Rails Assignment', date='2024-12-15', maxScore=200}
Assessment{id=1, courseId=1, name='Test1', date='2024-08-08', maxScore=100}
Assessment Management
1. Add Assessment
2. Update Assessment
3. Delete Assessment
4. List Assessments
0. Back to Main Menu
Enter your choice:
```

Result(Add,List)

```
Learning Management System
1. Manage Users
2. Manage Roles
3. Manage Courses
4. Manage Enrollments
5. Manage Assessments
6. Manage Results
0. Exit
Enter your choice: 6
Result Management
1. Add Result
2. Update Result
3. Delete Result
4. List Results
0. Back to Main Menu
Enter your choice: 1
Enter Result ID: 1
Enter Assessment ID: 1
Enter User ID: 1
Enter Score: 75
Result added successfully.
Result Management
1. Add Result
2. Update Result
3. Delete Result
4. List Results
0. Back to Main Menu
Enter your choice: 4
List of Results:
Result{resultId=1, assessmentId=1, userId=1, score=75}
Result Management
1. Add Result
2. Update Result
3. Delete Result
4. List Results
0. Back to Main Menu
Enter your choice:
```