PROJECT REPORT

Online Student Management System Using Spring and Hibernate

1. Introduction

The **Online Student Management System** is a Java-based mini-project developed using **Spring Framework** and **Hibernate ORM**.
It demonstrates how modern enterprise applications can efficiently manage data through **Dependency Injection**, **Object-Relational Mapping (ORM)**, and **Transaction Management**.

This project simplifies administrative tasks such as adding students, assigning courses, updating information, deleting records, and managing fee payments or refunds — all while ensuring data consistency and modular design.

🎯 2. Objectives

- To implement **Dependency Injection (DI)** using Spring's Java-based configuration.
- To perform **CRUD operations** (Create, Read, Update, Delete) using Hibernate ORM.
- To apply **Transaction Management** for secure and atomic fee payment or refund operations.
- To demonstrate **Spring + Hibernate Integration** in a layered architecture.
- To create a simple and modular **student-course management system**.

⚙️ 3. Technologies and Tools Used

Component	Technology
Programming Language	Java (JDK 11 or higher)
Frameworks	Spring (Core, Context), Hibernate ORM
Database	MySQL / H2 Database
Build Tool	Maven
IDE	IntelliJ IDEA / Eclipse
Additional Concepts	Dependency Injection, Transaction Management, DAO Pattern

🧩 4. System Architecture

The system follows a **three-layered architecture**:

1. **Presentation Layer**
   - Handled through a simple console-based interface (can later be extended to web).
   - Accepts input and displays output.
2. **Service Layer**
   - Contains business logic in StudentService.java.
   - Uses Spring's @Service annotation and dependency injection to interact with DAO.
3. **Data Access Layer (DAO)**
   - Implemented in StudentDAO.java using Hibernate.
   - Responsible for CRUD operations and communicating with the database.

🧱 5. Module Description

Module	Description
Student Management	Handles student registration, updates, deletion, and viewing records.
Course Management	Manages available courses and their association with students.
Fee Management	Performs payments, refunds, and ensures data integrity using transaction management.
Spring Configuration (AppConfig)	Defines all beans using Java-based configuration. Integrates Spring and Hibernate seamlessly.

🧮 6. Class Descriptions

App.java

Main entry point of the application.
Initializes the Spring context and retrieves the StudentService bean to perform CRUD and fee operations.

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

StudentService service = context.getBean(StudentService.class);

AppConfig.java

Contains all **Spring Bean configurations** and **Hibernate SessionFactory** setup using Java annotations.
Manages DataSource, SessionFactory, and Transaction Manager beans.

Student.java

Entity class mapped to the students table using Hibernate annotations.
Contains fields such as studentId, name, balance, and course.

Course.java

Represents the course entity.
Mapped to courses table and linked to Student via @ManyToOne relationship.

StudentDAO.java

Implements all **database operations** (save, update, delete, list, search) using Hibernate sessions.

StudentService.java

Contains **business logic** for student management and fee handling.
Uses Spring's @Transactional for transaction management.

🧠 7. Working of the Project

1. Application starts through App.java.
2. Spring initializes all beans defined in AppConfig.java.
3. User performs actions like adding a student or assigning a course.
4. StudentService calls corresponding DAO methods.
5. Hibernate converts object actions into SQL queries and executes them via JDBC.
6. Transaction management ensures rollback in case of any failure.

🗄️ 8. Database Design

Tables

1. students

Column	Type	Description
student_id	INT	Primary Key
name	VARCHAR	Student's name
course_id	INT	Foreign Key (courses table)
balance	DOUBLE	Remaining balance

2. courses

Column	Type	Description
course_id	INT	Primary Key
course_name	VARCHAR	Name of the course
duration	INT	Duration in months

3. payments

Column	Type	Description
payment_id	INT	Primary Key
student_id	INT	Foreign Key
amount	DOUBLE	Payment amount
date	DATE	Payment date

⚡ 9. Key Features

- Full CRUD (Create, Read, Update, Delete) operations.
- Spring Dependency Injection using Java Config (no XML).
- Hibernate ORM for database persistence.
- Transaction Management using @Transactional.
- Modular, reusable, and easily extendable codebase.
- Works with either MySQL or in-memory H2 database.

🧩 10. Sample Output

Application Started...

Student Added Successfully!

Student Details:

Name: Ankit Pareek

Course: Spring and Hibernate

Duration: 6 Months

Balance: 740000

📊 11. Future Enhancements

- Add a **web interface** using Spring MVC or Spring Boot.
- Implement **login and authentication** for admins/students.
- Integrate **email or SMS notifications** for payments.
- Create **reports and analytics** using graphical charts.
- Deploy on a cloud platform such as AWS or Azure.

🧾 12. Conclusion

This project successfully demonstrates how **Spring** and **Hibernate** can work together to build scalable, maintainable, and efficient Java applications.
It showcases key enterprise development principles such as:

- Loose coupling via **Dependency Injection**
- Clean database handling using **Hibernate ORM**
- Reliable data consistency through **Transaction Management**

The system provides a solid foundation for extending to a full-fledged online student management platform.

📚 13. References

1. Spring Framework Documentation – https://spring.io
2. Hibernate ORM Documentation – https://hibernate.org
3. Java SE Developer Guide – https://docs.oracle.com/javase
4. Maven Official Documentation – https://maven.apache.org