🏫 Case Story

TechEdge University, one of the leading institutes in technology and engineering, has decided to build a digital management system to handle academic data more efficiently.

Currently, all **department and student data** is being tracked manually in spreadsheets.
To modernize, the university has decided to develop a **Java-based application** using **Spring Core and Hibernate (JPA)** for automation and scalability.

Each **Department** (like Computer Science, Mechanical, or Civil) can have multiple **Students** enrolled.
Each **Student** must belong to exactly **one Department**.

Administrators using the system should be able to:

1. **Add new Departments**
2. **Add Students** and **assign them** to Departments
3. **View all Students** belonging to a Department
4. **Update** Department or Student information
5. **Delete** Students or Departments when required
6. **Fetch and manage data efficiently** using Hibernate's **Lazy Loading** and **Second-Level Cache**

Spring Core will handle all **dependency management**, **configuration**, and **service-level components** through annotations like @Component, @Autowired, and @Configuration.
Hibernate (JPA) will handle **data persistence**, **relationships**, and **caching**.

⚙️ System Requirements

Entities:

1. **Department**
    - id
    - name
    - students
2. **Student**
    - id
    - name
    - email
    - department

Relationships:

- **One Department → Many Students**

- **Many Students → One Department**

Behaviors:

- When a **Department** is saved, all its **Students** should automatically be saved
- Students should be fetched
- Frequently accessed entities (Departments and Students) should use **Second-Level Cache (Ehcache)** to reduce database queries.

💻 Functional Requirements

Your task is to implement the following operations using **Spring Core + Hibernate (JPA)**:

Operation	**Description**
addDepartment(Department dept)	Add a new department with basic details
addStudent(Student student)	Add a new student
assignStudentToDepartment(int deptId, Student student)	Assign a student to a specific department
getDepartmentById(int deptId)	Fetch department details (use Second-Level Cache)
viewStudentsByDepartment(int deptId)	List all students belonging to a specific department
updateStudent(int studentId, String newEmail)	Update a student's email or name
deleteStudent(int studentId)	Remove a student from the system
deleteDepartment(int deptId)	Remove a department and its students (Cascade delete)