🎓 Case Study: Library Management System

Author.java

```
package com.example.library.entity;
import jakarta.persistence.*;
@Entity
public class Author {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private long id;

	private String name;
	public Author() {}
	public Author(long id, String name) {
		this.id = id;
		this.name = name;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

}
```

Book.java

```
package com.example.library.entity;
import java.util.Date;
import jakarta.persistence.*;
@Entity
public class Book {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private long id;

	private String title;
```

```
    @Temporal(TemporalType.DATE)
    private Date publishDate;
    @ManyToOne
    @JoinColumn(name = "reader_id")
    private Reader reader;
    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;
    @ManyToOne
    @JoinColumn(name = "author_id")
    private Author author;
        public Book() {}
        public Book(long id, String title, Date publishDate, Reader reader, Category category,
Author author) {
            this.id = id;
            this.title = title;
            this.publishDate = publishDate;
            this.reader = reader;
            this.category = category;
            this.author = author;
        }
        public long getId() {
            return id;
        }
        public void setId(long id) {
            this.id = id;
        }
        public String getTitle() {
            return title;
        }
        public void setTitle(String title) {
            this.title = title;
        }
        public Date getPublishDate() {
            return publishDate;
        }
        public void setPublishDate(Date publishDate) {
            this.publishDate = publishDate;
        }
        public Reader getReader() {
            return reader;
        }
        public void setReader(Reader reader) {
            this.reader = reader;
        }
        public Category getCategory() {
            return category;
        }
        public void setCategory(Category category) {
```

```
		this.category = category;
	}
	public Author getAuthor() {
		return author;
	}
	public void setAuthor(Author author) {
		this.author = author;
	}

}
```

Category.java

```
package com.example.library.entity;
import jakarta.persistence.*;
@Entity
public class Category {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private long id;

	private String name;
	public Category() {}
	public Category(long id, String name) {
		this.id = id;
		this.name = name;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

}
```

Reader.java

```
package com.example.library.entity;
import jakarta.persistence.*;
```

```
@Entity
public class Reader {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private long id;

	private String name;
	private String email;

	public Reader() {}

	public Reader(long id, String name, String email) {
		this.id = id;
		this.name = name;
		this.email = email;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}
```

AuthorRepository.java

```
package com.example.library.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.library.entity.Author;
public interface AuthorRepository extends JpaRepository<Author, Long>{
}
```

BookRepository.java

```
package com.example.library.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.library.entity.Book;
public interface BookRepository extends JpaRepository<Book, Long>{
}
```

CategoryRepository.java

```
package com.example.library.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.library.entity.Category;
public interface CategoryRepository extends JpaRepository<Category, Long>{
}
```

ReaderRepository.java

```
package com.example.library.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.library.entity.Reader;
public interface ReaderRepository extends JpaRepository<Reader, Long>{
}
```

LibraryController.java

```
package com.example.library.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.example.library.entity.*;
import com.example.library.repository.*;
@RestController
@RequestMapping("/api")
public class LibraryController {

  @Autowired private CategoryRepository categoryRepo;
  @Autowired private AuthorRepository authorRepo;
  @Autowired private ReaderRepository readerRepo;
  @Autowired private BookRepository bookRepo;
  // ---------- CATEGORY ----------
  @PostMapping("/categories")
  public Category addCategory(@RequestBody Category category) {
     return categoryRepo.save(category);
  }
  @GetMapping("/categories")
  public List<Category> getAllCategories() {
     return categoryRepo.findAll();
  }
```

```
@PutMapping("/categories/{id}")
public Category updateCategory(@PathVariable Long id, @RequestBody Category category) {
    Category existing = categoryRepo.findById(id).orElseThrow();
    existing.setName(category.getName());
    return categoryRepo.save(existing);
}
@DeleteMapping("/categories/{id}")
public String deleteCategory(@PathVariable Long id) {
    categoryRepo.deleteById(id);
    return "Deleted Successfully";
}
// ----------------- AUTHOR -----------------
@PostMapping("/authors")
public Author addAuthor(@RequestBody Author author) {
    return authorRepo.save(author);
}
@GetMapping("/authors")
public List<Author> getAllAuthors() {
    return authorRepo.findAll();
}
@PutMapping("/authors/{id}")
public Author updateAuthor(@PathVariable Long id, @RequestBody Author author) {
    Author existing = authorRepo.findById(id).orElseThrow();
    existing.setName(author.getName());
    return authorRepo.save(existing);
}
@DeleteMapping("/authors/{id}")
public String deleteAuthor(@PathVariable Long id) {
    authorRepo.deleteById(id);
    return "Deleted Successfully";
}
// ----------------- READER -----------------
@PostMapping("/readers")
public Reader addReader(@RequestBody Reader reader) {
    return readerRepo.save(reader);
}
@GetMapping("/readers")
public List<Reader> getAllReaders() {
    return readerRepo.findAll();
}
@PutMapping("/readers/{id}")
public Reader updateReader(@PathVariable Long id, @RequestBody Reader reader) {
    Reader existing = readerRepo.findById(id).orElseThrow();
    existing.setName(reader.getName());
    existing.setEmail(reader.getEmail());
    return readerRepo.save(existing);
}
@DeleteMapping("/readers/{id}")
public String deleteReader(@PathVariable Long id) {
```

```
        readerRepo.deleteById(id);
        return "Deleted Successfully";
    }
    // ------------------ BOOK ------------------
    @PostMapping("/books")
    public Book addBook(@RequestBody Book book) {
        book.setReader(readerRepo.findById(book.getReader().getId()).orElseThrow());
        book.setCategory(categoryRepo.findById(book.getCategory().getId()).orElseThrow());
        book.setAuthor(authorRepo.findById(book.getAuthor().getId()).orElseThrow());
        return bookRepo.save(book);
    }
    @GetMapping("/books")
    public List<Book> getAllBooks() {
        return bookRepo.findAll();
    }
    @PutMapping("/books/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody Book book) {
        Book existing = bookRepo.findById(id).orElseThrow();
        existing.setTitle(book.getTitle());
        existing.setPublishDate(book.getPublishDate());
        existing.setReader(readerRepo.findById(book.getReader().getId()).orElseThrow());
        existing.setCategory(categoryRepo.findById(book.getCategory().getId()).orElseThrow());
        existing.setAuthor(authorRepo.findById(book.getAuthor().getId()).orElseThrow());
        return bookRepo.save(existing);
    }
    @DeleteMapping("/books/{id}")
    public String deleteBook(@PathVariable Long id) {
        bookRepo.deleteById(id);
        return "Deleted Successfully";
    }
}
```