Got it ✅ You want a **fun yet practical assignment problem statement** that requires learners to apply **OOPs concepts, Exception Handling, JPA, and Hibernate ORM**. Here’s one that combines a relatable theme with enough complexity to test all those skills:

**🎬 Assignment Problem Statement: Movie Booking & Review System**

**Background**

You have been hired to build the backend for an **online movie booking and review platform** called *CineVerse*. The platform allows users to browse movies, book tickets, write reviews, and manage their accounts.

The system must be implemented in **Java** using:

* **OOPs concepts** (abstraction, inheritance, encapsulation, polymorphism, interfaces)
* **Exception Handling** (custom exceptions for business rules)
* **JPA and Hibernate ORM** (for persistence with relational DB mapping)

**Core Requirements & Use Cases**

**1. User Management**

* Users can register with **name, email, password, role (USER / ADMIN)**.
* Admins can add/update/remove movies.
* Users can update their profile.

**Classes & Concepts:**

* User (Base Class) → Subclasses: Admin, Customer.
* Use **Encapsulation** for sensitive fields like password.
* Use **Polymorphism** for different role behaviors (e.g., only Admin can add movies).

**Exception Handling:**

* UserAlreadyExistsException if someone registers with an existing email.
* UnauthorizedAccessException if a non-admin tries to add a movie.

**2. Movie Management**

* Movies have **title, genre, duration, releaseDate, rating, and showTimes**.
* A movie can have **multiple showtimes** (One-to-Many mapping).
* Admins can add new movies and update existing ones.

**Classes & Concepts:**

* Movie entity
* ShowTime entity (One Movie → Many ShowTimes, bidirectional)
* Use **Lombok** for boilerplate code (getters/setters, constructors).

**Exception Handling:**

* MovieNotFoundException if searched movie doesn’t exist.
* InvalidShowTimeException if admin tries to add past dates.

**3. Booking Tickets**

* A customer can book tickets for a showtime.
* Each booking has **bookingId, customer, movie, showTime, seatCount, and status**.
* Once booked, seats reduce from available capacity.

**Classes & Concepts:**

* Booking entity with ManyToOne mapping (Many Bookings → One User).
* Implement **Abstraction**: an IBookable interface with method bookTickets().
* Concrete class TicketBookingService implements the interface.

**Exception Handling:**

* SeatsUnavailableException if user tries to book more seats than available.
* BookingNotFoundException if booking ID does not exist.

**4. Movie Reviews & Ratings**

* Customers can add reviews for movies.
* Review contains **reviewId, user, movie, comment, rating (1-5)**.
* A movie’s overall rating = average of all reviews.

**Classes & Concepts:**

* Review entity (Many Reviews → One Movie).
* **Polymorphism:** Different display of reviews (Admin sees flagged reviews too).

**Exception Handling:**

* InvalidRatingException if user gives rating outside 1–5.
* DuplicateReviewException if user tries to review the same movie twice.

**5. Persistence with JPA & Hibernate**

* Entities must be annotated with @Entity, @Table, @OneToMany, @ManyToOne, etc.
* Use EntityManager or Spring Data JPA Repository for CRUD operations.
* Relationships to implement:
  + **One-to-Many** → Movie → ShowTimes
  + **One-to-Many** → Movie → Reviews
  + **Many-to-One** → Booking → User

**Bonus Features (Optional for Advanced Learners)**

* Implement **search filters** (by genre, release year, rating).
* Implement **cancellation of bookings** with refund rules.
* Add **logging** using SLF4J/Logback.
* Implement **pagination** for reviews.

**Deliverables**

1. Java Classes implementing the above features with **OOPs concepts**.
2. **Custom Exception Classes** with proper handling.
3. **JPA + Hibernate mappings** for persistence.
4. Test cases (JUnit) for critical scenarios (e.g., seat booking, review submission).

👉 This problem statement is **fun (movies & reviews)**, **realistic (resembles BookMyShow/IMDb)**, and **rich enough** to cover **OOPs + Exception Handling + JPA/Hibernate** in one go.