can you create another assignment for books crud app. the same principles as the previous but with small changes (not big). also make it slightly harder.

**ChatGPT said:**

ChatGPT

Here’s an assignment for a Books CRUD app that follows the same principles as the previous one, but with small changes and a slightly increased difficulty.

**Project Description:**

You will create a command-line Java application to manage a database of "Books." This exercise will involve creating, reading, updating, and deleting book records from a MySQL database using JDBC with DriverManager. This version adds some validation and error-handling complexity compared to the previous exercise.

**Instructions:**

**1. Project Setup:**

* **Estimated Time**: 30-45 minutes
* **Tools**: Java, MySQL, Maven

**Steps**:

1. Install MySQL and ensure it's running.
2. Create a new Maven project.
3. Add the MySQL JDBC driver dependency in pom.xml:

xml

Copy code

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.31</version> <!-- Use the latest stable version -->

</dependency>

**2. Database Setup:**

* **Estimated Time**: 20-30 minutes

**Steps**:

1. Create a MySQL database named library\_db.

sql

Copy code

CREATE DATABASE library\_db;

1. Create a books table with fields: id, title, author, genre, published\_year, and isbn. Add a unique constraint on isbn to ensure no duplicate ISBNs.

sql

Copy code

CREATE TABLE books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(100),

genre VARCHAR(50),

published\_year INT,

isbn VARCHAR(20) UNIQUE

);

**3. Basic CRUD Operations (with Validation and Error Handling):**

* **Estimated Time**: 4-5 hours

You will implement the following CRUD operations with some input validation:

1. **Create**: Add a new book record, ensuring that the isbn is unique and published\_year is a valid year (>= 1450 and <= current year).
2. **Read**: Retrieve and display all book records.
3. **Update**: Update a book’s details, ensuring that any change in isbn doesn’t conflict with existing records.
4. **Delete**: Delete a book record by ID.

For each operation, write methods that interact with the MySQL database using JDBC and DriverManager. Additionally, add proper exception handling and user-friendly error messages.

**4. Code Implementation:**

**Steps**:

1. **Database Connection**: Create a utility class (DatabaseConnection) to manage the connection to MySQL using DriverManager (same as the previous task).
2. **Book Class**: Create a Book class to represent the book entity:

java

Copy code

public class Book {

private int id;

private String title;

private String author;

private String genre;

private int publishedYear;

private String isbn;

// Getters, Setters, and Constructor

}

1. **CRUD Methods**: Implement the CRUD operations in a BookRepository class.

**Create a Book with Validation**:

java

Copy code

public void addBook(Book book) {

if (book.getPublishedYear() < 1450 || book.getPublishedYear() > Year.now().getValue()) {

System.out.println("Error: Invalid published year.");

return;

}

String sql = "INSERT INTO books (title, author, genre, published\_year, isbn) VALUES (?, ?, ?, ?, ?)";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, book.getTitle());

stmt.setString(2, book.getAuthor());

stmt.setString(3, book.getGenre());

stmt.setInt(4, book.getPublishedYear());

stmt.setString(5, book.getIsbn());

stmt.executeUpdate();

} catch (SQLException e) {

if (e.getErrorCode() == 1062) {

System.out.println("Error: Duplicate ISBN.");

} else {

e.printStackTrace();

}

}

}

**Read Books**:

java

Copy code

public List<Book> getAllBooks() {

List<Book> books = new ArrayList<>();

String sql = "SELECT \* FROM books";

try (Connection conn = DatabaseConnection.getConnection();

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

Book book = new Book(rs.getInt("id"), rs.getString("title"),

rs.getString("author"), rs.getString("genre"),

rs.getInt("published\_year"), rs.getString("isbn"));

books.add(book);

}

} catch (SQLException e) {

e.printStackTrace();

}

return books;

}

**Update a Book with ISBN Check**:

java

Copy code

public void updateBook(Book book) {

String sql = "UPDATE books SET title = ?, author = ?, genre = ?, published\_year = ?, isbn = ? WHERE id = ?";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, book.getTitle());

stmt.setString(2, book.getAuthor());

stmt.setString(3, book.getGenre());

stmt.setInt(4, book.getPublishedYear());

stmt.setString(5, book.getIsbn());

stmt.setInt(6, book.getId());

stmt.executeUpdate();

} catch (SQLException e) {

if (e.getErrorCode() == 1062) {

System.out.println("Error: Duplicate ISBN.");

} else {

e.printStackTrace();

}

}

}

**Delete a Book**:

java

Copy code

public void deleteBook(int id) {

String sql = "DELETE FROM books WHERE id = ?";

try (Connection conn = DatabaseConnection.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, id);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

**5. Command-Line Interface (With Input Validation):**

* **Estimated Time**: 45-60 minutes

**Steps**:

1. Implement a command-line interface with basic options like "1. Add Book," "2. View Books," "3. Update Book," and "4. Delete Book."
2. Add input validation for isbn and published\_year to ensure they follow proper formats and values.

**6. Exception Handling and User Feedback:**

* **Estimated Time**: 45 minutes

Implement better error handling in the application:

* Catch SQL exceptions and show user-friendly messages for common errors (e.g., duplicate ISBN, invalid book details).
* Ensure that the system doesn't crash on bad input and prompts the user to correct their input.

**Deliverables:**

* Fully functional Books CRUD Java application.
* Maven pom.xml file with the MySQL JDBC driver dependency.
* SQL file with the database schema.
* A basic command-line interface.
* Input validation and exception handling.

**Extra Challenge (Optional):**

* Implement search functionality that allows users to search for books by title or author.
* Allow the user to update only specific fields (without requiring the entire record to be updated).

This version of the CRUD application adds more difficulty by incorporating ISBN validation, input constraints, and better error handling, giving you a more robust task while still focusing primarily on database interactions.