# Library Management System - Code Review & Error Analysis Report

## Project Overview
- **Frontend:** React + Vite + TailwindCSS
- **Backend:** Spring Boot + MySQL
- **Database:** MySQL (Hibernate ORM)
- **API Calls:** Axios (Frontend)

---

## Backend (Spring Boot) Analysis

###  Positive Aspects
 **Proper Layered Architecture** (Controller  Service  Repository).
 **Uses `JpaRepository` for Database Access.**
 **Proper API Endpoints (`GET`, `POST`, `PUT`, `DELETE`).**

###  Issues & Fixes

#### 1 Hardcoded Database Credentials (Security Risk)
- **Issue:** `spring.datasource.username=root`, `spring.datasource.password=root`
- **Fix:** Use environment variables.
  ```properties
  spring.datasource.username=${DB_USER}
  spring.datasource.password=${DB_PASS}
  ```

#### 2 No Exception Handling for Missing Records
- **Issue:** If a book or member is missing, the API returns `null` instead of a proper error.
- **Fix:** Use `ResponseEntity` to handle errors.
  ```java
  @GetMapping("/{book_isbn}")
  public ResponseEntity<?> getBookByIsbn(@PathVariable("book_isbn") String bookIsbn) {
      return bookRepository.findById(bookIsbn)
          .map(ResponseEntity::ok)
          .orElse(ResponseEntity.status(HttpStatus.NOT_FOUND).body("Book not found"));
  }
  ```

#### 3 Passwords Stored as Plain Text
- **Issue:** Passwords are directly stored in the database.
- **Fix:** Use **BCrypt password hashing**.
  ```java
  @Autowired
  ```

  private PasswordEncoder passwordEncoder;

  public Member addMember(Member member) {
      member.setPassword(passwordEncoder.encode(member.getPassword()));
      return memberRepository.save(member);
  }

#### 4 No `@Transactional` for Update & Delete Operations
- **Issue:** No transactional protection for data consistency.
- **Fix:** Add `@Transactional`.
  ```java
  @Transactional
  public void deleteBook(String bookIsbn) {
      bookRepository.deleteById(bookIsbn);
  }
  ```

#### 5 No Duplicate Check Before Adding Books or Members
- **Issue:** The system allows duplicate entries.
- **Fix:** Add `existsById()` method.
  ```java
  boolean existsById(String bookIsbn);
  ```

---

## Frontend (React) Analysis

###  Positive Aspects
 Uses **React Functional Components** (`useState`, `useEffect`).
 API Integration with **Axios (`axios.get()`, `axios.post()`).**
 UI uses **TailwindCSS** for styling.

###  Issues & Fixes

#### 1 Hardcoded API URL (`http://localhost:8080`)
- **Issue:** The API URL is hardcoded, which breaks in production.
- **Fix:** Use an **`.env` file**.
  ```env
  REACT_APP_API_URL=http://localhost:8080
  ```
  ```javascript
  const API_URL = process.env.REACT_APP_API_URL;
  axios.get(`${API_URL}/api/books`);
  ```

#### 2 No Error Handling for API Failures
- **Issue:** If the backend is down, the UI does not show any error.
- **Fix:** Display an error message in the UI.
  ```javascript
  const [error, setError] = useState(null);

  const fetchBooks = async () => {
      try {
          const response = await axios.get("http://localhost:8080/api/books");
          setBooks(response.data);
          setError(null);
      } catch (error) {
          console.error("Error fetching books:", error);
          setError("Failed to load books. Please try again.");
      }
  };
  ```

#### 3 Unused TypeScript Dependency (`typescript` Installed but Not Used)
- **Issue:** The project has `typescript`, but all files are `.jsx`.
- **Fix:**
  - If **not using TypeScript**, remove it:
    ```sh
    npm uninstall typescript
    ```
  - If using TypeScript, rename files (`App.jsx`  `App.tsx`).

#### 4 Missing `dotenv` for Environment Variables
- **Issue:** No `.env` support in `vite.config.js`.
- **Fix:** Install `dotenv`.
  ```sh
  npm install dotenv
  ```
  - Import it in `vite.config.js`:
    ```javascript
    import dotenv from "dotenv";
    dotenv.config();
    ```

#### 5 Inconsistent Book ISBN Key in `BooksTable.jsx`
- **Issue:** Backend uses `bookIsbn`, but frontend refers to `id`.
- **Fix:** Update `BooksTable.jsx`.
  ```javascript
  ```

```
  <td className="border px-4 py-2">{book.isbn}</td>
```

---

## Final Recommendations
 **Fix backend security risks (DB credentials, password hashing).**
 **Improve error handling (API responses, frontend UI errors).**
 **Remove unused dependencies (`typescript` if not used).**
 **Use `.env` for API URLs to support production deployment.**
 **Ensure database constraints (unique emails, book ISBNs).**

**Let me know if you need help implementing these fixes! **