第三次作业

一、题目

请基于权限的用户访问控制（RBAC）机制的原理，并在基于监听器模式和事件响应机制下，在事件处理的机制上实现一个 RBAC 的应用实例。其中要求：

1）完整的代码实现；

2）页面操作的显示，与角色和权限控制的实现；

3）对 RBAC 权限控制机制进行分析（例如对页面、对象、功能按钮之间控制的异同）；

扩展：RBAC 是一个模型簇，可以展开对比分析。

二、RBAC 模型簇

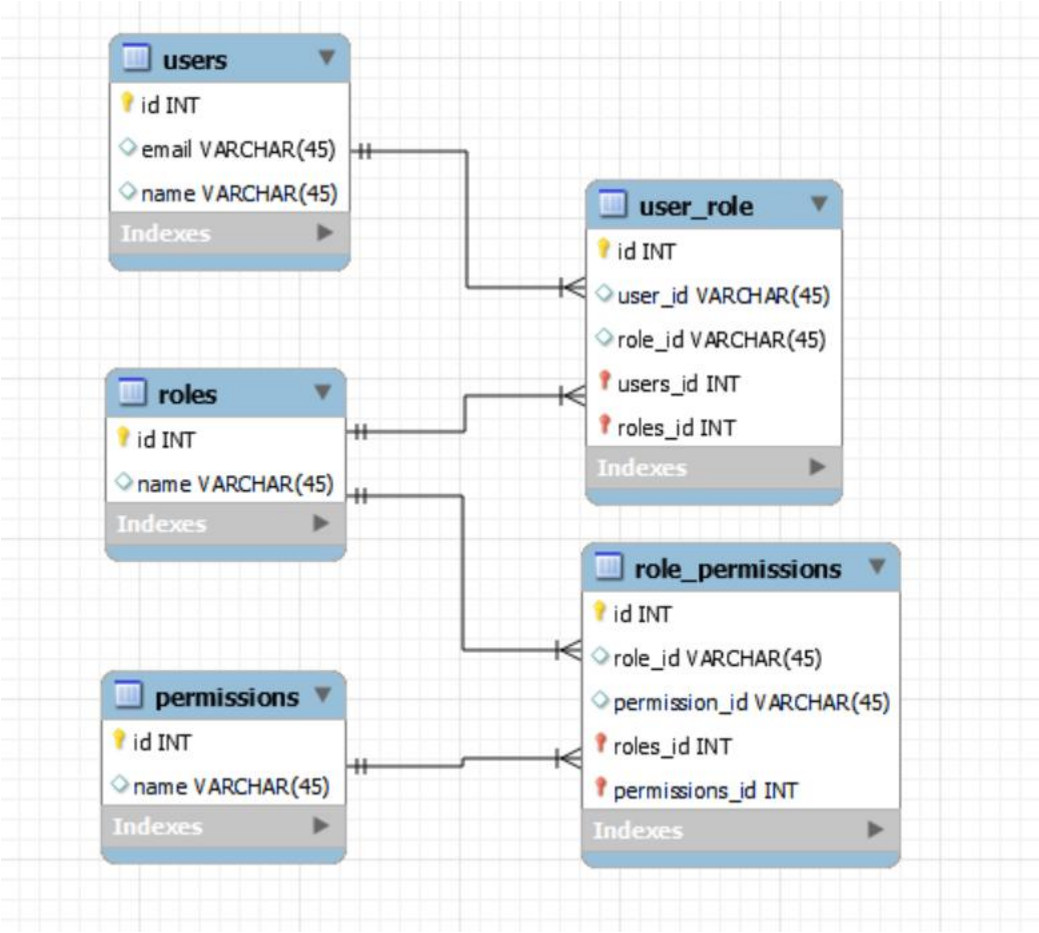
RBAC，即基于角色的访问控制， 是一种常用的权限管理机制，它通过定义不同的角色，每个角色对应特定的权限，从而对用户的访问和操作进行控制。RBAC 的核心思想是用户到角色到权限的映射关系，用户通过角色继承相应的权限，并基于这些权限对系统中的资源进行访问和操作。

其中，用户是系统中的个体，代表实际的操作人员或实体。每个用户在系统中都有一个唯一的标识符，一般是用户 ID。用户可以是员工、管理员、客户等，通常有自己的个人资料和登录凭据，通过这些凭据可以访问系统。

角色是一组权限的集合，代表特定的工作职能或职责。通过创建角色，可以将一组权限分配给多个用户，从而简化权限管理。角色可以根据组织结构和需求进行定义，比如“管理员”、“开发者”、“项目经理”等。

权限是允许用户执行特定操作的能力，比如读取、写入、修改、删除数据等。在 RBAC 中，权限通常与系统资源或操作相关联。权限的定义和管理是确保系统安全的关键。

经典 RBAC 五表设计如下：



用户与角色通过 user_role 表建立关联，一个用户可以拥有多个角色，而一个角色也可以被多个用户分配。

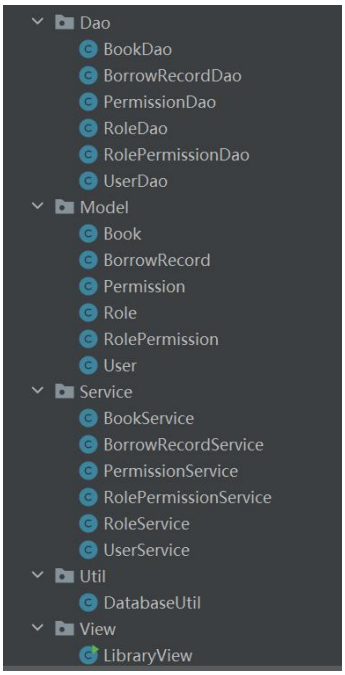
角色与权限通过 role_permissions 表建立关联，一个角色可以有多个权限，而同一个权限可以被多个角色拥有。

用户通过其分配的角色，间接获得这些角色所拥有的权限。因此，权限的分配并不是直接给用户的，而是通过角色进行中介，从而使得权限管理更加简化和可扩展。

三、基于 RBAC 模型设计图书管理系统

基于 RBAC 模型，我们可以设计实现一个图书管理系统。

在我设计的图书管理系统中，角色分为管理员、采购员和用户三种，其中管理员能够查看图书馆的藏书情况、添加图书、删除图书、查看用户的借阅情况;采购员能够查看图书馆的藏书情况、借阅图书、添加图书、删除图书；用户能够查看图书馆的藏书情况、借阅图书。

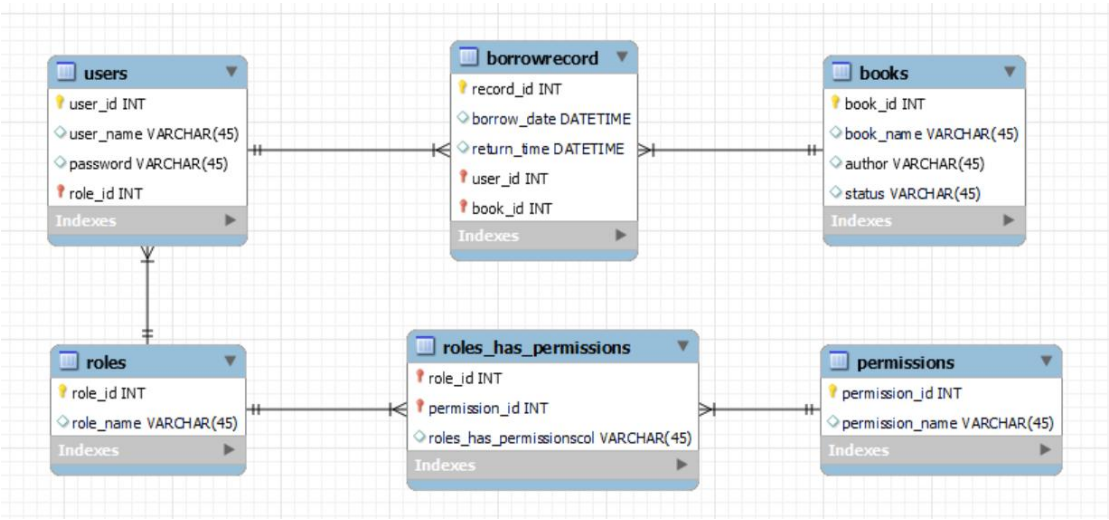


图书管理系统采用了分层架构和 MVC 设计模式，系统被分为五个层次，每一层负责不同的任务，包括视图层（View）、工具层（Util）、业务逻辑层（Service）、数据访问层（DAO），以及模型层（Model）。

在图书管理系统中，视图层（View）通过图形用户界面与用户交互，利用监听器模式和事件响应机制，将用户操作（如登录、注册、借阅图书）与后台逻辑连接起来。当用户点击按钮或输入信息时视图层的监听器捕获这些事件，并将事件信息传递给业务逻辑层

（Service）进行处理。业务逻辑层根据请求的类型执行相应的业务逻辑，例如验证用户登录、添加图书、更新借阅记录等操作。业务逻辑层随后调用数据访问层（DAO），通过数据库查询或更新来获取和修改所需的数据。DAO 层使用工具层（Util）提供的数据库连接工具（如 DatabaseUtil）来管理数据库连接和执行 SQL 操作，将数据持久化或检索回来。处理完数据后，DAO 层将结果返回给业务逻辑层，业务逻辑层再根据业务规则处理这些数据，并将最终结果返回给视图层。视图层根据业务逻辑层返回的结果更新界面（如显示图书列表、借阅情况或提示用户登录成功）。系统中的数据结构由模型层（Model）定义，包括用户、图书和借阅记录等实体类，这些实体类在视图层、业务逻辑层和数据访问层之间传递，确保各层之间的数据一致性和共享，从而实现整个系统的顺畅交互与功能运作。

以下是图书管理系统的数据库设计：



四、图书管理系统的具体实现

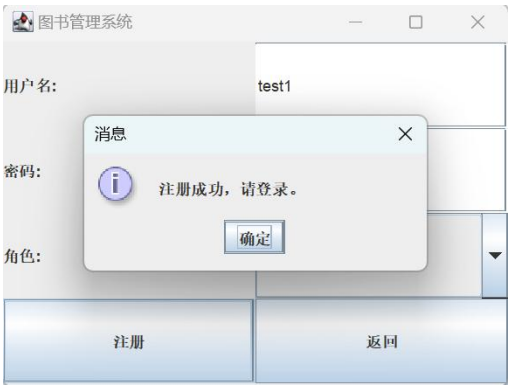
进入系统，自动弹出初始界面：

此时若点击登录则进行登陆界面，点击注册进行注册界面，点击退出则退出程序。以下是登陆界面和注册界面的展示：

用户在注册界面填入用户名、密码即其角色（假设用户的注册都是在管理员的帮助下，因此用户会分配到正确的角色），点击注册后即可完成注册，以下是一个注册示例：

假设有一个新的用户想要借阅图书，他需要完成注册后才能进行后续操作，假设其给自己设置的用户名为 test1，密码为 test111，角色为用户，显示界面如下图所示；

注册成功后，系统会进行对应显示：



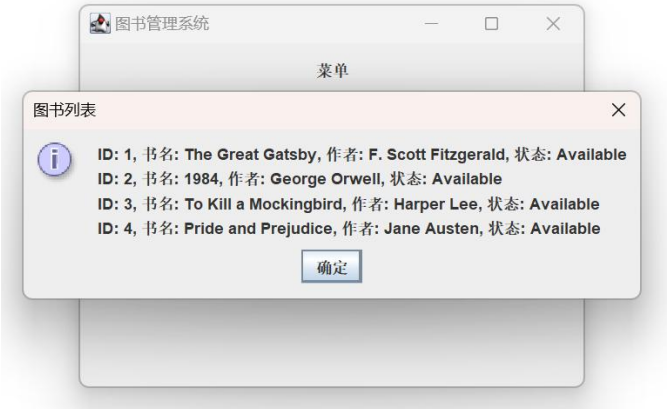
打开数据库，我们发现该用户的信息已经被添加在系统中：

user_id	user_name	password	role_id
1	admin	admin123	1
2	user	user123	2
3	purchaser	purchaser123	3
8	test1	test111	2
NULL	NULL	NULL	NULL

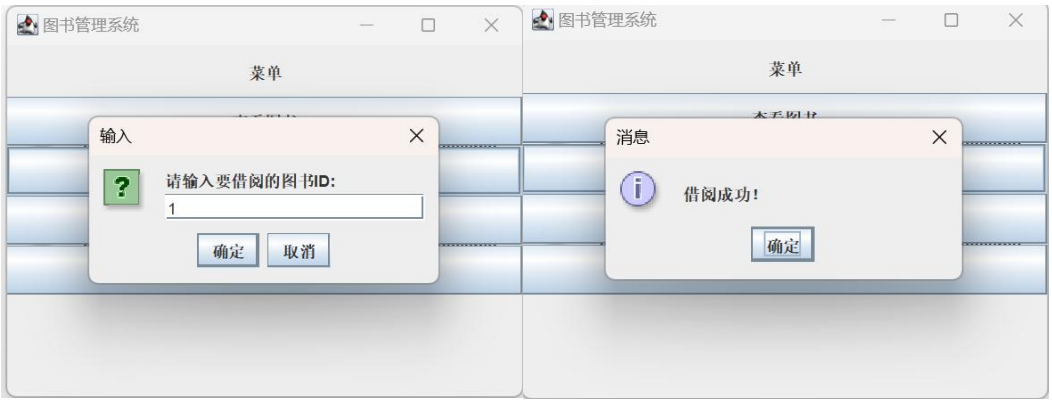
接着我们尝试使用刚刚注册的账号进行登陆操作：

登录成功后，系统将进入菜单页面，由于登陆账号的角色为用户，则该账号登录后仅有“查看图书”、“借阅图书”、“归还图书”的权限，具体显示如下：

点击“查看图书”按钮，即可查看所有图书的基本情况（包括书名、作者、借阅情况）：



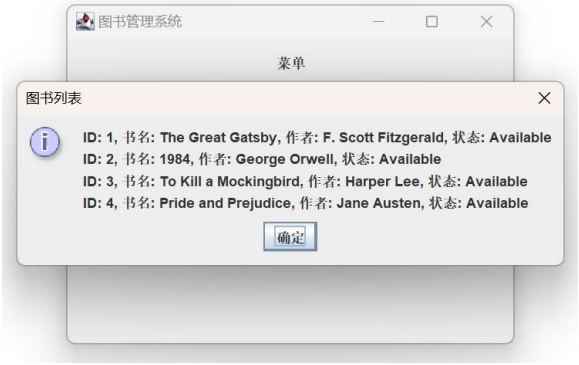
初始状态下，所有图书都是没有被借阅的，所以状态都是“Available”，接着点击“确定”按钮回到菜单，并点击“借阅图书”按钮，尝试借阅 ID 为 1 的图书：



借阅成功并返回菜单后，再次点击“查看图书”按钮，发现 ID 为 1 的图书状态改为“Borrowed”，说明借阅成功：

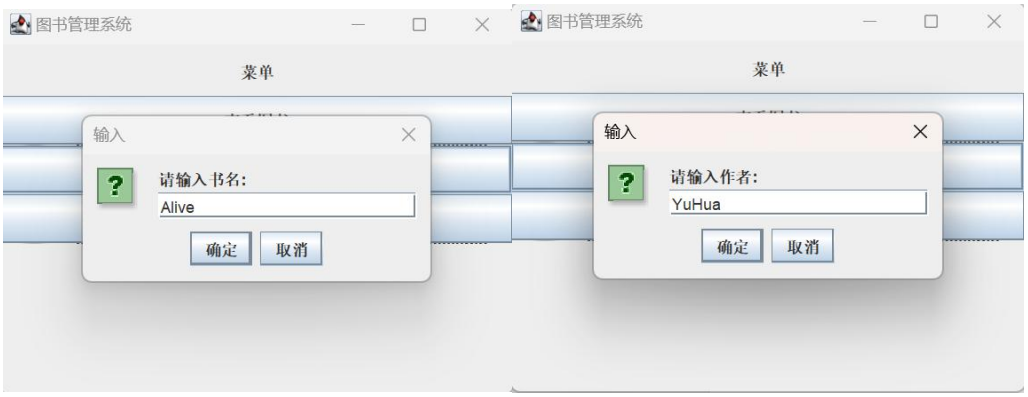
接着尝试归还图书，若归还没有借阅的图书则系统会进行提示，归还该用户借阅的图书则会显示归还成功：

此时再次点击“查看图书”按钮，发现 ID 为 1 的图书的状态转换为“Available”，说明归还图书操作成功：



退出登录，注册一个角色为“采购员”的账号，并进行登录，其应有“查看图书”、“添加图书”的权限：

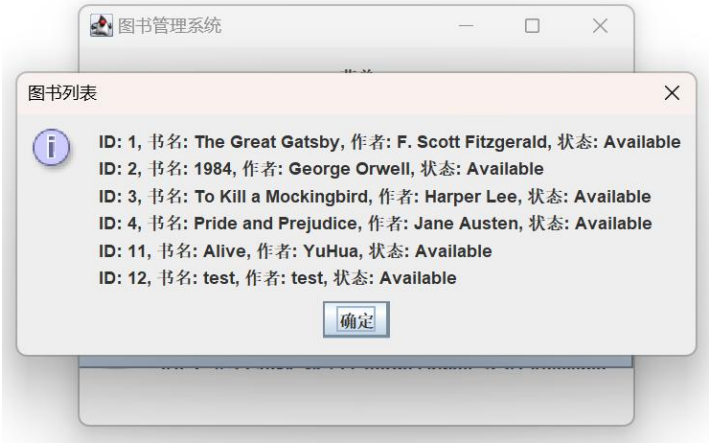
点击“添加图书”按钮，尝试添加图书“Alive”，作者为“YuHua”：



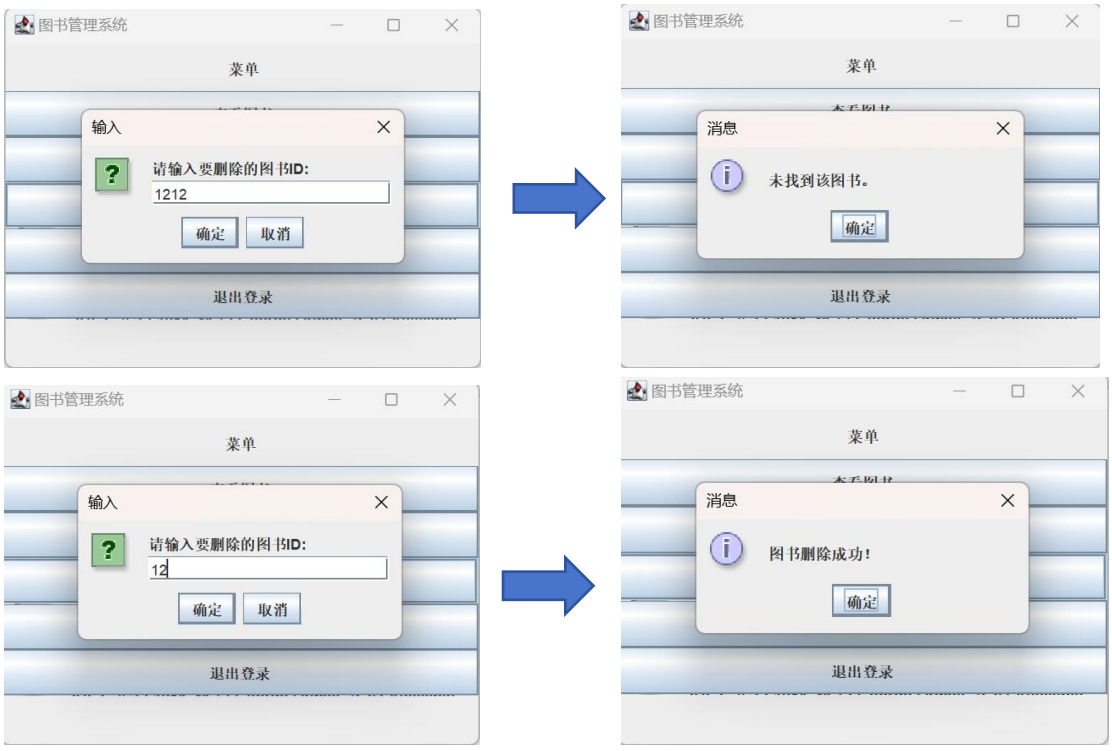
点击“确认”按钮，系统会显示添加成功，接着点击“查看图书”按钮，发现“Alive”的基本信息已被录入系统，图书添加成功：

退出登录，注册一个角色为“管理员”的账号，登录该账号，其应有“查看图书”、“添加图书”、“删除图书”、“查看借阅情况”的权限：

先准备一个书名和作者均为“test”的图书作为测试删除图书功能的示例：



通过“查看图书”可知其 ID 为 12，接着点击“删除图书”按钮，输入图书 ID 即可删除对应图书，若输入的 ID 不存在则会进行提示，否则成功删除该 ID 号的图书：



返回菜单后再次点击“查看图书”按钮，发现 ID 为 12 的图书已经不在数据库中，说明图书删除成功：

返回到菜单，先借阅 ID 为 2 的图书作为对照，后点击“查看借阅情况”按钮，即可查询到所有借阅情况（包括借阅的用户、图书书名、借阅时间和是否归还及归还时间）：

以上即为本人实现的图书管理系统的全部功能。

附：图书管理系统的 java 代码：

1. Dao 层

1.1 BookDao：

```
package src.Dao;

import src.Model.Book;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class BookDao {

    public List<Book> getAllBooks() {
        List<Book> books = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM books";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                Book book = new Book();
                book.setBookId(rs.getInt("book_id"));
                book.setTitle(rs.getString("title"));
                book.setAuthor(rs.getString("author"));
                book.setStatus(rs.getString("status"));
                books.add(book);
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return books;
    }

    public void addBook(Book book) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "INSERT INTO books (title, author, status) VALUES (?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, book.getTitle());
            stmt.setString(2, book.getAuthor());
            stmt.setString(3, book.getStatus());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateBook(Book book) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "UPDATE books SET title = ?, author = ?, status = ? WHERE book_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, book.getTitle());
            stmt.setString(2, book.getAuthor());
            stmt.setString(3, book.getStatus());
            stmt.setInt(4, book.getBookId());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deleteBook(int bookId) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "DELETE FROM books WHERE book_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, bookId);
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

1.2 BorrowRecordDao：

```
package src.Dao;

import src.Model.BorrowRecord;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class BorrowRecordDao {

    public List<BorrowRecord> getAllBorrowRecords() {
        List<BorrowRecord> records = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM borrow_records";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                BorrowRecord record = new BorrowRecord();
                record.setRecordId(rs.getInt("record_id"));
                record.setUserId(rs.getInt("user_id"));
                record.setBookId(rs.getInt("book_id"));
                record.setBorrowDate(rs.getDate("borrow_date"));
                record.setReturnDate(rs.getDate("return_date"));
                records.add(record);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return records;
    }

    public void addBorrowRecord(BorrowRecord record) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "INSERT INTO borrow_records (user_id, book_id, borrow_date) VALUES (?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, record.getUserId());
            stmt.setInt(2, record.getBookId());
            // 使用 Timestamp 来包含具体时间
            stmt.setTimestamp(3, new java.sql.Timestamp(record.getBorrowDate().getTime()));
```

```
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateBorrowRecord(BorrowRecord record) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "UPDATE borrow_records SET return_date = ? WHERE record_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            // 使用 Timestamp 来包含具体时间
            stmt.setTimestamp(1, new java.sql.Timestamp(record.getReturnDate().getTime()));
            stmt.setInt(2, record.getRecordId());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deleteBorrowRecord(int recordId) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "DELETE FROM borrow_records WHERE record_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, recordId);
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

1.3 PermissionDao：

```
package src.Dao;

import src.Model.Permission;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class PermissionDao {
```

```
public List<Permission> getAllPermissions() {
    List<Permission> permissions = new ArrayList<>();
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "SELECT * FROM permissions";
        PreparedStatement stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            Permission permission = new Permission();
            permission.setPermissionId(rs.getInt("permission_id"));
            permission.setPermissionName(rs.getString("permission_name"));
            permissions.add(permission);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return permissions;
}

public void addPermission(Permission permission) {
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "INSERT INTO permissions (permission_name) VALUES (?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, permission.getPermissionName());
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void updatePermission(Permission permission) {
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "UPDATE permissions SET permission_name = ? WHERE permission_id = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, permission.getPermissionName());
        stmt.setInt(2, permission.getPermissionId());
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void deletePermission(int permissionId) {
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "DELETE FROM permissions WHERE permission_id = ?";
```

```
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, permissionId);
                stmt.executeUpdate();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

1.4 RoleDao：

```
package src.Dao;

import src.Model.Role;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class RoleDao {

    public List<Role> getAllRoles() {
        List<Role> roles = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM roles";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                Role role = new Role();
                role.setRoleId(rs.getInt("role_id"));
                role.setRoleName(rs.getString("role_name"));
                roles.add(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return roles;
    }

    public void addRole(Role role) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "INSERT INTO roles (role_name) VALUES (?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
```

```
            stmt.setString(1, role.getRoleName());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateRole(Role role) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "UPDATE roles SET role_name = ? WHERE role_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, role.getRoleName());
            stmt.setInt(2, role.getRoleId());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deleteRole(int roleId) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "DELETE FROM roles WHERE role_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, roleId);
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

1.5 RolePermissionDao：

```
package src.Dao;

import src.Model.RolePermission;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class RolePermissionDao {
```

```
// 获取指定角色的所有权限
public List<RolePermission> getPermissionsByRoleId(int roleId) {
    List<RolePermission> rolePermissions = new ArrayList<>();
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "SELECT * FROM roles_has_permissions WHERE role_id = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, roleId);
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            RolePermission rolePermission = new RolePermission();
            rolePermission.setRoleId(rs.getInt("role_id"));
            rolePermission.setPermissionId(rs.getInt("permission_id"));
            rolePermissions.add(rolePermission);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return rolePermissions;
}

// 为角色添加权限
public void addRolePermission(int roleId, int permissionId) {
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "INSERT INTO roles_has_permissions (role_id, permission_id) VALUES (?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, roleId);
        stmt.setInt(2, permissionId);
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// 移除角色的权限
public void deleteRolePermission(int roleId, int permissionId) {
    try (Connection conn = DatabaseUtil.getConnection()) {
        String sql = "DELETE FROM roles_has_permissions WHERE role_id = ? AND permission_id = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, roleId);
        stmt.setInt(2, permissionId);
        stmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

```
    }

    // 获取所有角色和权限的关联关系
    public List<RolePermission> getAllRolePermissions() {
        List<RolePermission> rolePermissions = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM roles_has_permissions";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                RolePermission rolePermission = new RolePermission();
                rolePermission.setRoleId(rs.getInt("role_id"));
                rolePermission.setPermissionId(rs.getInt("permission_id"));
                rolePermissions.add(rolePermission);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rolePermissions;
    }
}
```

1.6 UserDao：

```
package src.Dao;

import src.Model.User;
import src.Util.DatabaseUtil;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class UserDao {

    public List<User> getAllUsers() {
        List<User> users = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM users";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                User user = new User();
                user.setUserId(rs.getInt("user_id"));
```

```
                user.setUserName(rs.getString("user_name"));
                user.setPassword(rs.getString("password"));
                user.setRoleId(rs.getInt("role_id"));
                users.add(user);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return users;
    }

    public void addUser(User user) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "INSERT INTO users (user_name, password, role_id) VALUES (?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, user.getUserName());
            stmt.setString(2, user.getPassword());
            stmt.setInt(3, user.getRoleId());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void updateUser(User user) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "UPDATE users SET user_name = ?, password = ?, role_id = ? WHERE user_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, user.getUserName());
            stmt.setString(2, user.getPassword());
            stmt.setInt(3, user.getRoleId());
            stmt.setInt(4, user.getUserId());
            stmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deleteUser(int userId) {
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "DELETE FROM users WHERE user_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, userId);
            stmt.executeUpdate();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public User getUserByCredentials(String userName, String password) {
        User user = null;
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM users WHERE user_name = ? AND password = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, userName);
            stmt.setString(2, password);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                user = new User();
                user.setUserId(rs.getInt("user_id"));
                user.setUserName(rs.getString("user_name"));
                user.setPassword(rs.getString("password"));
                user.setRoleId(rs.getInt("role_id"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return user;
    }

    public User getUserById(int userId) {
        User user = null;
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM users WHERE user_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                user = new User();
                user.setUserId(rs.getInt("user_id"));
                user.setUserName(rs.getString("user_name"));
                user.setPassword(rs.getString("password"));
                user.setRoleId(rs.getInt("role_id"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return user;
```

```
        }

}
```

2. Model 层

2.1 Book

```
package src.Model;

public class Book {
        private int bookId;
        private String title;
        private String author;
        private String status;

        // Getters and Setters
        public int getBookId() {
                return bookId;
        }

        public void setBookId(int bookId) {
                this.bookId = bookId;
        }

        public String getTitle() {
                return title;
        }

        public void setTitle(String title) {
                this.title = title;
        }

        public String getAuthor() {
                return author;
        }

        public void setAuthor(String author) {
                this.author = author;
        }

        public String getStatus() {
                return status;
        }

        public void setStatus(String status) {
```

```
        this.status = status;
    }
}
```

2.2 BorrowRecord

```
package src.Model;

import java.util.Date;

public class BorrowRecord {
    private int recordId;
    private int userId;
    private int bookId;
    private Date borrowDate;
    private Date returnDate;

    // Getters and Setters
    public int getRecordId() {
        return recordId;
    }

    public void setRecordId(int recordId) {
        this.recordId = recordId;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getBookId() {
        return bookId;
    }

    public void setBookId(int bookId) {
        this.bookId = bookId;
    }

    public Date getBorrowDate() {
        return borrowDate;
    }
```

```
    public void setBorrowDate(Date borrowDate) {
        this.borrowDate = borrowDate;
    }

    public Date getReturnDate() {
        return returnDate;
    }

    public void setReturnDate(Date returnDate) {
        this.returnDate = returnDate;
    }
}
```

2.3 Permission

```
package src.Model;

public class Permission {
    private int permissionId;
    private String permissionName;

    // Getters and Setters
    public int getPermissionId() {
        return permissionId;
    }

    public void setPermissionId(int permissionId) {
        this.permissionId = permissionId;
    }

    public String getPermissionName() {
        return permissionName;
    }

    public void setPermissionName(String permissionName) {
        this.permissionName = permissionName;
    }
}
```

2.4 Role

```
package src.Model;

public class Role {
    private int roleId;
```

```
    private String roleName;

    // Getters and Setters
    public int getRoleId() {
        return roleId;
    }

    public void setRoleId(int roleId) {
        this.roleId = roleId;
    }

    public String getRoleName() {
        return roleName;
    }

    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }
}
```

2.5 RolePermission

```
package src.Model;

public class RolePermission {
    private int roleId;
    private int permissionId;

    // Getters and Setters
    public int getRoleId() {
        return roleId;
    }

    public void setRoleId(int roleId) {
        this.roleId = roleId;
    }

    public int getPermissionId() {
        return permissionId;
    }

    public void setPermissionId(int permissionId) {
        this.permissionId = permissionId;
    }
}
```

2.6 User

```
package src.Model;

public class User {
    private int userId;
    private String userName;
    private String password;
    private int roleId;

    // Getters and Setters
    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public int getRoleId() {
        return roleId;
    }

    public void setRoleId(int roleId) {
        this.roleId = roleId;
    }
}
```

3. Service 层

3.1 BookService

```
package src.Service;

import src.Dao.BookDao;
import src.Model.Book;
import java.util.List;

public class BookService {
    private BookDao bookDao = new BookDao();

    public List<Book> getAllBooks() {
        return bookDao.getAllBooks();
    }

    public void addBook(Book book) {
        bookDao.addBook(book);
    }

    public void updateBook(Book book) {
        bookDao.updateBook(book);
    }

    public void deleteBook(int bookId) {
        bookDao.deleteBook(bookId);
    }

    public Book getBookById(int bookId) {
        List<Book> books = bookDao.getAllBooks();
        for (Book book : books) {
            if (book.getBookId() == bookId) {
                return book;
            }
        }
        return null;
    }
}
```

3.2 BorrowRecordService

```
package src.Service;

import src.Dao.BorrowRecordDao;
import src.Model.BorrowRecord;
import src.Util.DatabaseUtil;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class BorrowRecordService {
    private BorrowRecordDao borrowRecordDao = new BorrowRecordDao();

    public List<BorrowRecord> getAllBorrowRecords() {
        return borrowRecordDao.getAllBorrowRecords();
    }

    public void addBorrowRecord(BorrowRecord record) {
        borrowRecordDao.addBorrowRecord(record);
    }

    public void updateBorrowRecord(BorrowRecord record) {
        borrowRecordDao.updateBorrowRecord(record);
    }

    public void deleteBorrowRecord(int recordId) {
        borrowRecordDao.deleteBorrowRecord(recordId);
    }

    public List<BorrowRecord> getBorrowRecordsByUserId(int userId) {
        List<BorrowRecord> records = new ArrayList<>();
        try (Connection conn = DatabaseUtil.getConnection()) {
            String sql = "SELECT * FROM borrow_records WHERE user_id = ? AND return_date IS NULL";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, userId);
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                BorrowRecord record = new BorrowRecord();
                record.setRecordId(rs.getInt("record_id"));
                record.setUserId(rs.getInt("user_id"));
                record.setBookId(rs.getInt("book_id"));
                record.setBorrowDate(rs.getDate("borrow_date"));
                record.setReturnDate(rs.getDate("return_date"));
                records.add(record);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            return records;
        }
}
```

3.3 PermissionService

```
package src.Service;

import src.Dao.PermissionDao;
import src.Model.Permission;
import java.util.List;

public class PermissionService {
        private PermissionDao permissionDao = new PermissionDao();

        public List<Permission> getAllPermissions() {
            return permissionDao.getAllPermissions();
        }

        public void addPermission(Permission permission) {
            permissionDao.addPermission(permission);
        }

        public void updatePermission(Permission permission) {
            permissionDao.updatePermission(permission);
        }

        public void deletePermission(int permissionId) {
            permissionDao.deletePermission(permissionId);
        }
}
```

3.4 RolePermissionService

```
package src.Service;

import src.Dao.RolePermissionDao;
import src.Model.RolePermission;
import java.util.List;

public class RolePermissionService {
        private RolePermissionDao rolePermissionDao = new RolePermissionDao();

        // 根据角色 ID 获取所有权限
        public List<RolePermission> getPermissionsByRoleId(int roleId) {
            return rolePermissionDao.getPermissionsByRoleId(roleId);
```

```
    }

    // 为角色添加权限
    public void addPermissionToRole(int roleId, int permissionId) {
        rolePermissionDao.addRolePermission(roleId, permissionId);
    }

    // 从角色中移除权限
    public void removePermissionFromRole(int roleId, int permissionId) {
        rolePermissionDao.deleteRolePermission(roleId, permissionId);
    }

    // 获取所有角色和权限的关联关系
    public List<RolePermission> getAllRolePermissions() {
        return rolePermissionDao.getAllRolePermissions();
    }
}
```

3.5 RoleService

```
package src.Service;

import src.Dao.RoleDao;
import src.Dao.RolePermissionDao;
import src.Model.Role;
import src.Model.RolePermission;
import java.util.List;

public class RoleService {
    private RoleDao roleDao = new RoleDao();
    private RolePermissionDao rolePermissionDao = new RolePermissionDao();

    public List<Role> getAllRoles() {
        return roleDao.getAllRoles();
    }

    public void addRole(Role role) {
        roleDao.addRole(role);
    }

    public void updateRole(Role role) {
        roleDao.updateRole(role);
    }

    public void deleteRole(int roleId) {
```

```
        roleDao.deleteRole(roleId);
    }

    public void addPermissionToRole(int roleId, int permissionId) {
        rolePermissionDao.addRolePermission(roleId, permissionId);
    }

    public void removePermissionFromRole(int roleId, int permissionId) {
        rolePermissionDao.deleteRolePermission(roleId, permissionId);
    }

    public List<RolePermission> getPermissionsByRoleId(int roleId) {
        return rolePermissionDao.getPermissionsByRoleId(roleId);
    }
}
```

3.6 UserService

```
package src.Service;

import src.Dao.UserDao;
import src.Model.User;
import java.util.List;

public class UserService {
    private UserDao userDao = new UserDao();

    public List<User> getAllUsers() {
        return userDao.getAllUsers();
    }

    public boolean login(String userName, String password) {
        User user = getUserByCredentials(userName, password);
        return user != null;
    }

    public void registerUser(User user) {
        userDao.addUser(user);
    }

    public void updateUser(User user) {
        userDao.updateUser(user);
    }

    public void deleteUser(int userId) {
```

```
        userDao.deleteUser(userId);
    }

    public User getUserByCredentials(String userName, String password) {
        return userDao.getUserByCredentials(userName, password);
    }

    public User getUserById(int userId) {
        return userDao.getUserById(userId);
    }

}
```

4. Util 层

4.1 DatabaseUtil：

```
package src.Util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseUtil {
    // 数据库连接配置
    private static final String URL = "jdbc:mysql://localhost:3306/library_db";
    private static final String USER = "root"; // 数据库用户名
    private static final String PASSWORD = "wangxinghao130"; // 数据库密码

    // 加载数据库驱动（只需加载一次）
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            throw new RuntimeException("数据库驱动加载失败！");
        }
    }

    // 获取数据库连接
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    // 关闭数据库连接
    public static void closeConnection(Connection connection) {
```

```
            if (connection != null) {
                try {
                    connection.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
}
```

5. View 层

5.1 LibraryView：

```
package src.View;

import src.Service.UserService;
import src.Service.BookService;
import src.Service.BorrowRecordService;
import src.Model.User;
import src.Model.Book;
import src.Model.BorrowRecord;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class LibraryView {
    private JFrame frame;
    private UserService userService = new UserService();
    private BookService bookService = new BookService();
    private BorrowRecordService borrowRecordService = new BorrowRecordService();
    private User currentUser; //  保存当前登录用户

    public LibraryView() {
        initialize();
    }

    private void initialize() {
        frame = new JFrame("图书管理系统");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setLayout(new BorderLayout());

        showMainMenu();
    }
```

```
private void showMainMenu() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(4, 1));

    JLabel welcomeLabel = new JLabel("欢迎使用图书管理系统", SwingConstants.CENTER);
    panel.add(welcomeLabel);

    JButton loginButton = new JButton("登录");
    JButton registerButton = new JButton("注册");
    JButton exitButton = new JButton("退出");

    panel.add(loginButton);
    panel.add(registerButton);
    panel.add(exitButton);

    frame.getContentPane().removeAll();
    frame.getContentPane().add(panel, BorderLayout.CENTER);
    frame.revalidate();
    frame.repaint();

    loginButton.addActionListener(e -> showLoginScreen());
    registerButton.addActionListener(e -> showRegisterScreen());
    exitButton.addActionListener(e -> System.exit(0));
}

private void showLoginScreen() {
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(3, 2));

    JLabel userLabel = new JLabel("用户名:");
    JLabel passLabel = new JLabel("密码:");
    JTextField userField = new JTextField();
    JPasswordField passField = new JPasswordField();

    panel.add(userLabel);
    panel.add(userField);
    panel.add(passLabel);
    panel.add(passField);

    JButton loginButton = new JButton("登录");
    JButton backButton = new JButton("返回");

    panel.add(loginButton);
```

```
        panel.add(backButton);

        frame.getContentPane().removeAll();
        frame.getContentPane().add(panel, BorderLayout.CENTER);
        frame.revalidate();
        frame.repaint();

        loginButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());
            User user = userService.getUserByCredentials(username, password);
            if (user != null) {
                JOptionPane.showMessageDialog(frame, "登录成功！");
                currentUser = user;
                showUserMenu();
            } else {
                JOptionPane.showMessageDialog(frame, "用户名或密码错误，请重试。");
            }
        });

        backButton.addActionListener(e -> showMainMenu());
    }

    private void showRegisterScreen() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4, 2));

        JLabel userLabel = new JLabel("用户名:");
        JLabel passLabel = new JLabel("密码:");
        JTextField userField = new JTextField();
        JPasswordField passField = new JPasswordField();

        JLabel roleLabel = new JLabel("角色:");
        JComboBox<String> roleComboBox = new JComboBox<>(new String[]{"Admin", "User", "Purchaser"});

        panel.add(userLabel);
        panel.add(userField);
        panel.add(passLabel);
        panel.add(passField);
        panel.add(roleLabel);
        panel.add(roleComboBox);

        JButton registerButton = new JButton("注册");
        JButton backButton = new JButton("返回");
```

```
        panel.add(registerButton);
        panel.add(backButton);

        frame.getContentPane().removeAll();
        frame.getContentPane().add(panel, BorderLayout.CENTER);
        frame.revalidate();
        frame.repaint();

        registerButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());
            String role = (String) roleComboBox.getSelectedItem();
            int roleId = switch (role) {
                case "Admin" -> 1;
                case "User" -> 2;
                case "Purchaser" -> 3;
                default -> 2;
            };

            User newUser = new User();
            newUser.setUserName(username);
            newUser.setPassword(password);
            newUser.setRoleId(roleId);
            userService.registerUser(newUser);
            JOptionPane.showMessageDialog(frame, "注册成功，请登录。");
            showMainMenu();
        });

        backButton.addActionListener(e -> showMainMenu());
    }

    private void showUserMenu() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(7, 1)); // 增加一个行来容纳新的按钮

        JLabel menuLabel = new JLabel("菜单", SwingConstants.CENTER);
        panel.add(menuLabel);

        JButton viewBooksButton = new JButton("查看图书");
        panel.add(viewBooksButton);

        JButton actionButton1 = new JButton();
        JButton actionButton2 = new JButton();
```

```
JButton viewBorrowRecordsButton = new JButton("查看借阅情况");
JButton logoutButton = new JButton("退出登录");

// 根据用户角色显示不同的操作
switch (currentUser.getRoleId()) {
    case 1: // Admin
        actionButton1.setText("添加图书");
        actionButton2.setText("删除图书");
        panel.add(actionButton1);
        panel.add(actionButton2);
        panel.add(viewBorrowRecordsButton);
        break;
    case 2: // User
        actionButton1.setText("借阅图书");
        actionButton2.setText("归还图书");
        panel.add(actionButton1);
        panel.add(actionButton2);
        break;
    case 3: // Purchaser
        actionButton1.setText("添加图书");
        panel.add(actionButton1);
        break;
    default:
        JOptionPane.showMessageDialog(frame, "无效的角色。");
        return;
}

panel.add(logoutButton);

frame.getContentPane().removeAll();
frame.getContentPane().add(panel, BorderLayout.CENTER);
frame.revalidate();
frame.repaint();

viewBooksButton.addActionListener(e -> showBookList());

actionButton1.addActionListener(e -> {
    if (currentUser.getRoleId() == 1 || currentUser.getRoleId() == 3) {
        addBook();
    } else if (currentUser.getRoleId() == 2) {
        borrowBook();
    }
});
```

```
        actionButton2.addActionListener(e -> {
            if (currentUser.getRoleId() == 1) {
                deleteBook();
            } else if (currentUser.getRoleId() == 2) {
                returnBook();
            }
        });

        viewBorrowRecordsButton.addActionListener(e -> {
            if (currentUser.getRoleId() == 1) {
                showBorrowRecords();
            }
        });

        logoutButton.addActionListener(e -> {
            currentUser = null;
            showMainMenu();
        });
    }

    private void showBorrowRecords() {
        List<BorrowRecord> records = borrowRecordService.getAllBorrowRecords();
        StringBuilder recordList = new StringBuilder();
        for (BorrowRecord record : records) {
            User user = userService.getUserById(record.getUserId());
            Book book = bookService.getBookById(record.getBookId());
            recordList.append("用户: ").append(user.getUserName())
                    .append(", 图书: ").append(book.getTitle())
                    .append(", 借阅时间: ").append(record.getBorrowDate())
                    .append(", 归还时间: ")
                    .append(record.getReturnDate() != null ? record.getReturnDate() : "未归还")
                    .append("\n");
        }
        JOptionPane.showMessageDialog(frame, recordList.toString(), "借阅情况",
JOptionPane.INFORMATION_MESSAGE);
    }

    private void showBookList() {
        List<Book> books = bookService.getAllBooks();
        StringBuilder bookList = new StringBuilder();
        for (Book book : books) {
            bookList.append("ID: ").append(book.getBookId())
                    .append(", 书名: ").append(book.getTitle())
                    .append(", 作者: ").append(book.getAuthor())
```

```
                .append(", 状态: ").append(book.getStatus()).append("\n");
        }
        JOptionPane.showMessageDialog(frame, bookList.toString(), "图书列表",
JOptionPane.INFORMATION_MESSAGE);
    }

    private void addBook() {
        String title = JOptionPane.showInputDialog(frame, "请输入书名:");
        // 检查是否取消输入或输入为空
        if (title == null || title.trim().isEmpty()) {
            return; // 取消操作，不显示异常
        }

        String author = JOptionPane.showInputDialog(frame, "请输入作者:");
        // 检查是否取消输入或输入为空
        if (author == null || author.trim().isEmpty()) {
            return; // 取消操作，不显示异常
        }

        Book newBook = new Book();
        newBook.setTitle(title.trim());
        newBook.setAuthor(author.trim());
        newBook.setStatus("Available");

        bookService.addBook(newBook); // 调用服务层方法来添加图书
        JOptionPane.showMessageDialog(frame, "图书添加成功！");
    }

    private void deleteBook() {
        String bookIdStr = JOptionPane.showInputDialog(frame, "请输入要删除的图书 ID:");
        // 检查是否取消输入或输入为空
        if (bookIdStr == null || bookIdStr.trim().isEmpty()) {
            return; // 取消操作，不显示异常
        }

        try {
            int bookId = Integer.parseInt(bookIdStr);
            Book book = bookService.getBookById(bookId);
            if (book != null) {
                bookService.deleteBook(bookId); // 调用服务层方法来删除图书
                JOptionPane.showMessageDialog(frame, "图书删除成功！");
            } else {
                JOptionPane.showMessageDialog(frame, "未找到该图书。");
            }
```

```
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(frame, "请输入有效的图书 ID。");
        }
    }

    private void borrowBook() {
        String bookIdStr = JOptionPane.showInputDialog(frame, "请输入要借阅的图书 ID:");
        try {
            int bookId = Integer.parseInt(bookIdStr);
            Book book = bookService.getBookById(bookId);
            if (book != null && "Available".equals(book.getStatus())) {
                BorrowRecord record = new BorrowRecord();
                record.setUserId(currentUser.getUserId());
                record.setBookId(bookId);
                record.setBorrowDate(new java.util.Date());
                borrowRecordService.addBorrowRecord(record); // 添加借阅记录
                book.setStatus("Borrowed");
                bookService.updateBook(book); // 更新图书状态
                JOptionPane.showMessageDialog(frame, "借阅成功！");
            } else {
                JOptionPane.showMessageDialog(frame, "该图书不可借阅或不存在。");
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(frame, "请输入有效的图书 ID。");
        }
    }

    private void returnBook() {
        String bookIdStr = JOptionPane.showInputDialog(frame, "请输入要归还的图书 ID:");
        try {
            int bookId = Integer.parseInt(bookIdStr);
            List<BorrowRecord> records =
borrowRecordService.getBorrowRecordsByUserId(currentUser.getUserId());
            for (BorrowRecord record : records) {
                if (record.getBookId() == bookId && record.getReturnDate() == null) {
                    record.setReturnDate(new java.util.Date());
                    borrowRecordService.updateBorrowRecord(record); // 更新借阅记录
                    Book book = bookService.getBookById(bookId);
                    if (book != null) {
                        book.setStatus("Available");
                        bookService.updateBook(book); // 更新图书状态
                    }
                    JOptionPane.showMessageDialog(frame, "归还成功！");
                    return;
```

```
                    }
                }
                JOptionPane.showMessageDialog(frame, "未找到该借阅记录，请检查图书 ID。");
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(frame, "请输入有效的图书 ID。");
            }
        }

        public static void main(String[] args) {
            EventQueue.invokeLater(() -> {
                LibraryView window = new LibraryView();
                window.frame.setVisible(true);
            });
        }
}
```