**移动应用开发大作业**

**题目:Mamba书店的开发与设计**

**院 系:**

**专业班级:**

**学 号:**

**学生姓名:**

**2025年 4 月 28 日**

[**移动应用开发大作业** 1](#_Toc196763375)

[一、Android应用介绍 3](#_Toc196763376)

[1. APP介绍 3](#_Toc196763377)

[2. APP功能 3](#_Toc196763378)

[3. APP亮点 3](#_Toc196763379)

[二、详细设计 3](#_Toc196763380)

[1. 流程图 3](#_Toc196763381)

[2. 数据库设计 4](#_Toc196763382)

[3. 注册、登录模块设计 6](#_Toc196763383)

[（1） 用户登录逻辑 7](#_Toc196763384)

[（2） 用户注册逻辑 7](#_Toc196763385)

[4. 购物车模块设计 9](#_Toc196763386)

[（1） 用户添加到购物车 9](#_Toc196763387)

[（2） 用户在购物车中购买 10](#_Toc196763388)

[（3） 商品从购物车添加到订单 11](#_Toc196763389)

[（4） 用户删除订单 11](#_Toc196763390)

[5. 图书功能模块设计 12](#_Toc196763391)

[（1） 图书数据管理 12](#_Toc196763392)

[（2） 图书列表展示 13](#_Toc196763393)

[（3） 图书详情展示 15](#_Toc196763394)

[（4） 图书适配器 16](#_Toc196763395)

[6. 管理员模块设计 18](#_Toc196763396)

[（1） 图书管理界面 18](#_Toc196763397)

[（2） 添加图书 20](#_Toc196763398)

[（3） 编辑图书 21](#_Toc196763399)

[（4） 删除图书 23](#_Toc196763400)

[（5） 图书列表展示 24](#_Toc196763401)

[三、实际演示 26](#_Toc196763402)

[四、总结及参考文献 29](#_Toc196763403)

# 一、Android应用介绍

## APP介绍

Mamba书店是一个面向用户的图书商城应用，旨在提供一种便捷的线上图书购物体验。它适用于那些希望在线浏览、购买图书，并能够管理个人书籍收藏的用户。无论是学生、爱书之人，还是日常阅读爱好者，都能在该平台找到自己喜欢的书籍并完成购买。

## APP功能

用户功能：

* 用户可以通过注册账号并登录来访问自己的购买历史。
* 用户可以浏览各种图书类别、查看图书详情，帮助用户快速找到感兴趣的图书。
* 用户可以将图书加入购物车，并可以修改购物车中图书的数量或删除图书。
* 用户可以从购物车生成订单并支付。
* 用户可以查看自己的订单，查看订单的详细信息（如购买的图书、数量、价格等），并查看订单的支付状态。

管理员功能：

* 允许管理员输入图书信息并保存到数据库。
* 显示图书列表并提供按钮跳转到添加、编辑和删除图书功能。
* 允许管理员根据图书ID修改图书信息。
* 允许管理员根据ISBN删除图书。
* 展示图书数据，支持分类显示。

## APP亮点

简洁直观的界面：

* 应用设计简洁，用户能够快速上手，操作流程直观清晰，确保用户在短时间内完成所需操作。

高效的购物车和订单管理：

* 用户可以方便地管理购物车，查看和修改商品数量，生成订单。

管理员便捷的后台管理功能：

* 管理员可以通过后台管理界面对图书进行增删改操作，确保平台的图书库保持最新，订单处理及时。

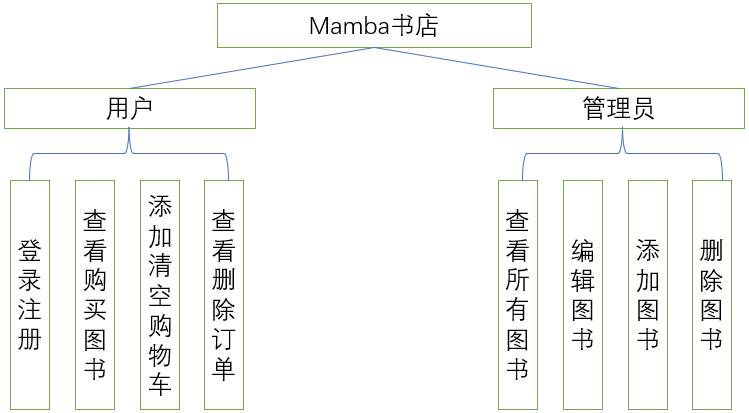
本地数据库支持：

* 本应用使用SQLite作为本地数据库，确保用户数据的持久化存储，支持快速查询和数据更新，保证高效的响应速度。

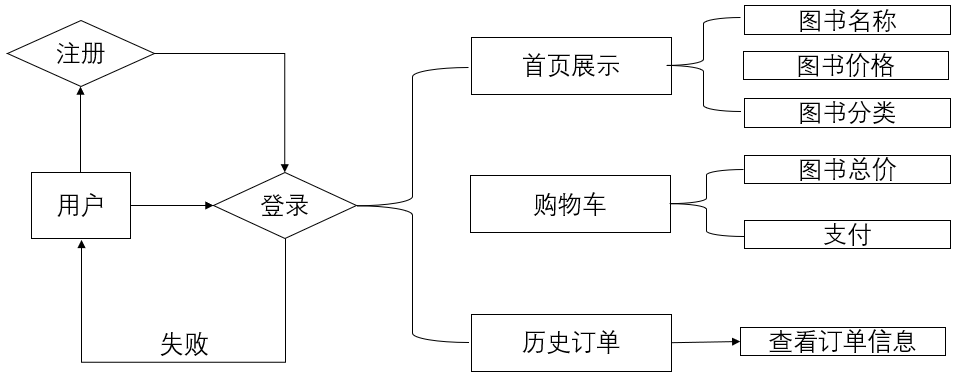
# 二、详细设计

## 流程图

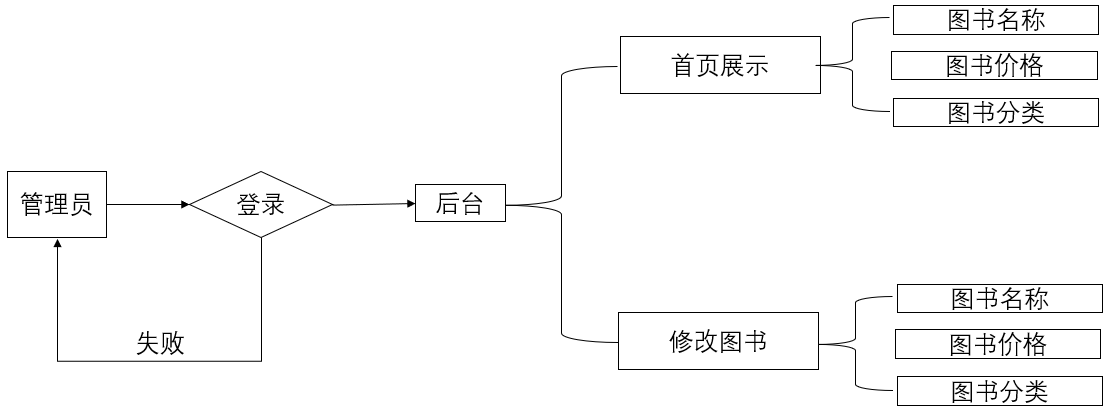
1. 系统功能流程图



1. 用户流程图



1. 管理员流程图



## 数据库设计

account 表（用户账户表）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 说明 |
| id | INTEGER | 是 | 自增主键，用户ID |
| username | VARCHAR(50) |  | 用户名 |
| password | CHAR(32) |  | 用户密码 |

book 表（图书表）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 说明 |
| id | INTEGER | 是 | 自增主键，图书ID |
| category | TINYINT |  | 类别编号 |
| name | VARCHAR(25) |  | 书名 |
| img\_url | VARCHAR(100) |  | 图片URL |
| author | VARCHAR(15) |  | 作者名 |
| isbn | CHAR(13) |  | ISBN编号 |
| description | TEXT |  | 书籍描述 |
| price | DOUBLE |  | 价格 |

cart 表（购物车表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 外键关联 | 说明 |
| id | INTEGER | 是 |  | 自增主键，购物车记录ID |
| account\_id | INTEGER |  | account(id) | 用户ID |
| book\_id | INTEGER |  | book(id) | 图书ID |
| quantity | INTEGER |  |  | 数量 |

book\_order 表（订单表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 外键关联 | 说明 |
| id | INTEGER | 是 |  | 自增主键，订单ID |
| account\_id | INTEGER |  | account(id) | 下单的用户ID |
| order\_time | CHAR(19) |  |  | 下单时间 |

order\_detail 表（订单详情表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 主键 | 外键关联 | 说明 |
| id | INTEGER | 是 |  | 自增主键，订单详情记录ID |
| order\_id | INTEGER |  | book\_order(id) | 所属订单ID |
| book\_id | INTEGER |  | book(id) | 图书ID |
| order\_price | DOUBLE |  |  | 成交时的单价 |
| quantity | INTEGER |  |  | 购买数量 |

public void onCreate(SQLiteDatabase db) {  
 *// 账号表* db.execSQL("CREATE TABLE account" +  
 "(id INTEGER PRIMARY KEY AUTOINCREMENT," +  
 "username VARCHAR(50)," +  
 "password CHAR(32))");  
 *// 书本表* db.execSQL("CREATE TABLE book" +  
 "(id INTEGER PRIMARY KEY AUTOINCREMENT," +  
 "category TINYINT," +  
 "name VARCHAR(25)," +  
 "img\_url VARCHAR(100)," +  
 "author VARCHAR(15)," +  
 "isbn CHAR(13)," +  
 "description TEXT," +  
 "price DOUBLE)");  
 *// 购物车表* db.execSQL("CREATE TABLE cart (" +  
 "id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +  
 "account\_id INTEGER NOT NULL," +  
 "book\_id INTEGER NOT NULL," +  
 "quantity INTEGER NOT NULL," +  
 "CONSTRAINT cart\_book\_id\_book\_id FOREIGN KEY (book\_id) REFERENCES book (id) ON DELETE CASCADE," +  
 "CONSTRAINT cart\_account\_id\_account\_id FOREIGN KEY (account\_id) REFERENCES account (id) ON DELETE CASCADE)");  
 *// 订单表* db.execSQL("CREATE TABLE book\_order (" +  
 "id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +  
 "account\_id INTEGER NOT NULL," +  
 "order\_time CHAR(19) NOT NULL," +  
 "CONSTRAINT order\_account\_id\_account\_id FOREIGN KEY (account\_id) REFERENCES account (id) ON DELETE CASCADE)");  
 *// 订单详情表* db.execSQL("CREATE TABLE order\_detail (" +  
 "id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +  
 "order\_id INTEGER NOT NULL," +  
 "book\_id INTEGER NOT NULL," +  
 "order\_price DOUBLE," +  
 "quantity INTEGER," +  
 "CONSTRAINT order\_detail\_order\_id\_order\_id FOREIGN KEY (order\_id) REFERENCES book\_order (id) ON DELETE CASCADE ON UPDATE NO ACTION," +  
 "CONSTRAINT order\_detail\_book\_id\_book\_id FOREIGN KEY (book\_id) REFERENCES book (id) ON DELETE SET NULL ON UPDATE NO ACTION)");  
 *// 添加初始数据* BookService.*initData*(db);  
}

## 注册、登录模块设计

用户输入用户名、密码，点击【登录】按钮，LoginActivity 调用 MyAuth.login(username, password)，MyAuth 查询 account表中是否存在匹配的用户名和密码，如果找到匹配则登录成功，进入主界面（MainActivity），如果未找到则提示用户名或密码错误。管理员用户是静态设置为admin，密码admin123，登录成功进入管理员主界面。

### 用户登录逻辑

MyAuth auth = new MyAuth(LoginActivity.this);

if (login) {

// 登录模式

// 1. 检查管理员账号

if ("admin".equals(user) && "admin123".equals(pwd)) {

// 管理员登录，跳转到管理员界面

Intent intent = new Intent(LoginActivity.this, AdminActivity.class);

startActivity(intent);

finish(); // 结束当前登录界面

return;

}

//普通用户登录

switch (auth.authUser(user, pwd)) {

case SUCCESS:

SharedPreferences sp = getSharedPreferences("preferences", MODE\_PRIVATE);

SharedPreferences.Editor editor = sp.edit();

editor.putString("username", user);

editor.apply();

goToMain();

break;

case INVALID\_USERNAME\_OR\_PWD:

Toast.makeText(LoginActivity.this, "用户名或密码错误", Toast.LENGTH\_SHORT).show();

break;

case USER\_EXISTED:

break;

case TOKEN\_TOO\_LONG:

Toast.makeText(LoginActivity.this, "用户名太长", Toast.LENGTH\_SHORT).show();

break;

case UNKNOWN\_ERROR:

Toast.makeText(LoginActivity.this, "出现未知错误", Toast.LENGTH\_SHORT).show();

break;

}

}

### 用户注册逻辑

MyAuth auth = new MyAuth(LoginActivity.this);

switch (auth.addUser(username, password)) {

case SUCCESS:

// 注册成功后保存用户名

SharedPreferences sp = getSharedPreferences("preferences", MODE\_PRIVATE);

SharedPreferences.Editor editor = sp.edit();

editor.putString("username", username);

editor.apply();

// 跳转到主界面

goToMain();

break;

case USER\_EXISTED:

// 提示用户名已存在

Toast.makeText(LoginActivity.this, "用户已存在", Toast.LENGTH\_SHORT).show();

break;

case TOKEN\_TOO\_LONG:

// 提示用户名太长

Toast.makeText(LoginActivity.this, "用户名太长", Toast.LENGTH\_SHORT).show();

break;

case UNKNOWN\_ERROR:

// 提示未知错误

Toast.makeText(LoginActivity.this, "出现未知错误", Toast.LENGTH\_SHORT).show();

break;

}

public AuthResult addUser(String username, String password) {

if (username.length() > 50)

return AuthResult.TOKEN\_TOO\_LONG;

SQLiteDatabase db = this.getWritableDatabase();

Cursor cursor = db.query("account", null, "username=?", new String[]{username}, null, null, null);

// 检查用户名是否已存在

if (cursor.getCount() > 0) {

cursor.close();

db.close();

return AuthResult.USER\_EXISTED;

}

// 插入新用户

ContentValues values = new ContentValues();

values.put("username", username);

values.put("password", md5(password));

long id = db.insert("account", null, values);

cursor.close();

db.close();

return (id > 0) ? AuthResult.SUCCESS : AuthResult.UNKNOWN\_ERROR;

}

public AuthResult authUser(String username, String password) {

if (username.length() > 50)

return AuthResult.TOKEN\_TOO\_LONG;

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query("account", new String[]{"password", "id"}, "username=?", new String[]{username}, null, null, null);

if (cursor.moveToFirst()) {

String storedPassword = cursor.getString(0);

userId = cursor.getInt(1);

cursor.close();

db.close();

// 验证密码

return storedPassword.equals(md5(password)) ? AuthResult.SUCCESS : AuthResult.INVALID\_USERNAME\_OR\_PWD;

}

cursor.close();

db.close();

return AuthResult.INVALID\_USERNAME\_OR\_PWD;

}

## 购物车模块设计

用户可以浏览图书并将图书添加到自己的购物车。该功能允许用户将感兴趣的书本加入购物车，在结算时进行订单管理。购物车功能通过数据库管理每个用户的购物车内容，并允许用户查看、修改（如增加数量、删除项）和最终提交订单。

### 用户添加到购物车

public void onClick(View v) {  
 if (v.getId() == R.id.*btn\_addToCart*) {  
 Cart cart = new Cart(getContext(), MyAuth.*getUserId*());  
 cart.addToCart(bookId, 1);  
 Toast.*makeText*(getContext(), "已加入购物车", Toast.*LENGTH\_SHORT*).show();  
 BottomNavigationView navBar = getActivity().findViewById(R.id.*nav\_view*);  
 navBar.getOrCreateBadge(R.id.*navigation\_cart*).setNumber(cart.getCartCount());  
 }  
}

//添加到购物车public void addToCart(int bookId, int quantity) {  
 SQLiteDatabase db = this.getWritableDatabase();  
 Cursor cursor = db.query("cart", null, "account\_id=? and book\_id=?", new String[]{String.*valueOf*(accountId), String.*valueOf*(bookId)}, null, null,null);  
 if (cursor.getCount() == 0) {  
 ContentValues values = new ContentValues();  
 values.put("account\_id", accountId);  
 values.put("book\_id", bookId);  
 values.put("quantity", quantity);  
 db.insertOrThrow("cart", null, values);  
 } else {  
 cursor.moveToNext();  
 int quantityInCart = cursor.getInt((int) cursor.getColumnIndex("quantity"));  
 ContentValues values = new ContentValues();  
 values.put("quantity", quantityInCart + quantity);  
 db.update("cart", values, "account\_id=? and book\_id=?", new String[]{String.*valueOf*(accountId), String.*valueOf*(bookId)});  
 }  
 cursor.close();  
 db.close();  
}

### 用户在购物车中购买

public void onClick(View v) {  
 if (v.getId() == R.id.*btn\_order*) {  
 if (cartItemList.size() > 0) {  
 new AlertDialog.Builder(getContext())  
 .setTitle(R.string.*app\_name*)  
 .setMessage("是否确定购买？")  
 .setPositiveButton("确定", (dialog, which) -> {  
 OrderService orderService = new OrderService(getContext(), MyAuth.*getUserId*());  
 orderService.createOrder(cart.getCart());  
 cart.clearCart();  
 updateTotalPriceView();  
 BottomNavigationView navBar = getActivity().findViewById(R.id.*nav\_view*);  
 navBar.removeBadge(R.id.*navigation\_cart*);  
 recyclerView.getAdapter().notifyItemRangeRemoved(0, cartItemList.size());  
 Toast.*makeText*(getContext(), "购买成功", Toast.*LENGTH\_SHORT*).show();  
 NavController navController = Navigation.*findNavController*(v);  
 navController.navigate(R.id.*action\_navigation\_cart\_to\_navigation\_order*);  
 })  
 .setNegativeButton("取消", (dialog, which) -> {}).create()  
 .show();  
 } else {  
 Toast.*makeText*(getContext(), "购物车为空", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
}

//清空购物车public void clearCart() {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete("cart", "account\_id=?", new String[]{String.*valueOf*(accountId)});  
 db.close();  
}

### 商品从购物车添加到订单

public void addOrder(Order order) {  
 SQLiteDatabase db = getWritableDatabase();  
 ContentValues values = new ContentValues();  
 values.put("account\_id", order.getAccountId());  
 values.put("order\_time", order.getOrderTime().format(Order.*dateTimeFormatter*));  
 long orderId = db.insertOrThrow("book\_order", null, values);  
 for (OrderDetail orderDetail : order.getOrderDetails()) {  
 ContentValues orderDetailValues = new ContentValues();  
 orderDetailValues.put("order\_id", orderId);  
 orderDetailValues.put("book\_id", orderDetail.getBookId());  
 orderDetailValues.put("order\_price", orderDetail.getOrderPrice());  
 orderDetailValues.put("quantity", orderDetail.getQuantity());  
 db.insertOrThrow("order\_detail", null, orderDetailValues);  
 }  
 db.close();  
}

### 用户删除订单

public void onClick(View v) {  
 if (v.getId() == R.id.*btn\_del\_order*) {  
 new AlertDialog.Builder(getContext())  
 .setTitle(R.string.*app\_name*)  
 .setMessage("是否确定删除此订单？")  
 .setPositiveButton("确定", (dialog, which) -> {  
 orderService.deleteOrder(orderId);  
 Toast.*makeText*(getContext(), "删除成功", Toast.*LENGTH\_SHORT*).show();  
 OrderFragment.*getInstance*().refresh(position); *// 通知列表更新* Navigation.*findNavController*(v).navigateUp(); *// 返回列表页面* })  
 .setNegativeButton("取消", (dialog, which) -> {}).create()  
 .show();  
 }  
}

//用户删除订单

public void deleteOrder(int orderId) {  
 SQLiteDatabase db = getWritableDatabase();  
 db.delete("order\_detail", "order\_id=?", new String[]{String.*valueOf*(orderId)});  
 db.delete("book\_order", "id=?", new String[]{String.*valueOf*(orderId)});  
 db.close();  
}

## 图书功能模块设计

### 图书数据管理

负责从数据库中获取图书数据，并提供默认数据初始化和图书增删改操作。

public class BookService extends MyDBHelper {

// 默认图书数据

private static final List<Book> defaultData = new ArrayList<Book>() {{

add(new Book(1, Book.Category.COMPUTER, "第一行代码", "image\_url", "郭霖", "9787115524836", "本书介绍了Android开发的知识...", 65.9));

}};

// 获取所有图书

public List<Book> getList() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query("book", null, null, null, null, null, null);

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

while (cursor.moveToNext()) {

result.add(deserializer(cursor)); // 将数据库数据转换为Book对象

}

cursor.close();

db.close();

return result;

}

// 获取单本图书的详细信息

public Book getBook(int id) {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = db.query("book", null, "id=?", new String[]{String.valueOf(id)}, null, null, null);

cursor.moveToNext();

Book book = deserializer(cursor); // 反序列化

cursor.close();

db.close();

return book;

}

// 反序列化数据库中的数据为Book对象

private static Book deserializer(Cursor cursor) {

Book book = new Book();

book.setId(cursor.getInt(cursor.getColumnIndex("id")));

book.setName(cursor.getString(cursor.getColumnIndex("name")));

book.setCategory(Book.Category.values()[cursor.getInt(cursor.getColumnIndex("category"))]);

book.setImgUrl(cursor.getString(cursor.getColumnIndex("img\_url")));

book.setAuthor(cursor.getString(cursor.getColumnIndex("author")));

book.setISBN(cursor.getString(cursor.getColumnIndex("isbn")));

book.setDescription(cursor.getString(cursor.getColumnIndex("description")));

book.setPrice(cursor.getDouble(cursor.getColumnIndex("price")));

return book;

}

// 初始化图书数据

public void initData(SQLiteDatabase db) {

for (Book book : defaultData) {

ContentValues values = new ContentValues();

values.put("name", book.getName());

values.put("category", book.getCategory().ordinal());

values.put("img\_url", book.getImgUrl());

values.put("author", book.getAuthor());

values.put("isbn", book.getISBN());

values.put("description", book.getDescription());

values.put("price", book.getPrice());

db.insert("book", null, values); // 向数据库中插入图书数据

}

}

}

### 图书列表展示

使用RecyclerView展示图书列表，通过适配器绑定数据并响应点击事件，点击图书跳转到详情页面。

public class BookFragment extends Fragment implements MyBookRecyclerViewAdapter.OnItemClickListener {

private RecyclerView recyclerView;

private GridLayoutManager gridLayoutManager;

private MyBookRecyclerViewAdapter adapter;

private ArrayList<Object> bookListWithHeader;

private BookService bookService;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

bookService = new BookService(getContext());

List<Book> books = bookService.getList(); // 获取图书数据

// 分类图书

HashMap<Book.Category, List<Book>> categorizedBooks = categorizeBooks(books);

bookListWithHeader = new ArrayList<>();

for (Map.Entry<Book.Category, List<Book>> entry : categorizedBooks.entrySet()) {

bookListWithHeader.add(entry.getKey()); // 添加分类标题

bookListWithHeader.addAll(entry.getValue()); // 添加该分类下的图书

}

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_book\_item\_list, container, false);

// 设置RecyclerView

recyclerView = view.findViewById(R.id.book\_list);

gridLayoutManager = new GridLayoutManager(getContext(), 2);

recyclerView.setLayoutManager(gridLayoutManager);

adapter = new MyBookRecyclerViewAdapter(bookListWithHeader);

adapter.setOnItemClickListener(this);

recyclerView.setAdapter(adapter);

return view;

}

@Override

public void onItemClick(View v) {

// 获取点击项并跳转到详情页

int position = recyclerView.getChildAdapterPosition(v);

Book book = (Book) bookListWithHeader.get(position);

Bundle bundle = new Bundle();

bundle.putInt("id", book.getId()); // 将图书ID传递到详情页

NavController navController = Navigation.findNavController(v);

navController.navigate(R.id.action\_navigation\_booklist\_to\_bookDetailFragment, bundle);

}

// 按照类别对图书进行分类

private HashMap<Book.Category, List<Book>> categorizeBooks(List<Book> books) {

HashMap<Book.Category, List<Book>> map = new HashMap<>();

for (Book book : books) {

map.putIfAbsent(book.getCategory(), new ArrayList<>());

map.get(book.getCategory()).add(book);

}

return map;

}

}

### 图书详情展示

通过图书ID获取图书的详细信息并展示。

public class BookDetailFragment extends Fragment implements View.OnClickListener {

private int bookId;

private TextView nameView, authorView, isbnView, descView;

private NWImageView imageView;

private BookService bookService;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

assert getArguments() != null;

bookId = getArguments().getInt("id");

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_book\_detail, container, false);

nameView = view.findViewById(R.id.text\_name2);

authorView = view.findViewById(R.id.text\_author2);

isbnView = view.findViewById(R.id.text\_isbn);

descView = view.findViewById(R.id.text\_desc);

imageView = view.findViewById(R.id.detail\_img);

FloatingActionButton addToCart = view.findViewById(R.id.btn\_addToCart);

addToCart.setOnClickListener(this);

// 获取图书并展示

bookService = new BookService(getContext());

Book book = bookService.getBook(bookId);

nameView.setText(book.getName());

authorView.setText(book.getAuthor());

isbnView.setText(String.format("ISBN: %s", book.getISBN()));

descView.setText(book.getDescription());

imageView.setImageURL(book.getImgUrl());

return view;

}

@Override

public void onClick(View v) {

if (v.getId() == R.id.btn\_addToCart) {

// 添加图书到购物车

Cart cart = new Cart(getContext(), MyAuth.getUserId());

cart.addToCart(bookId, 1);

Toast.makeText(getContext(), "已加入购物车", Toast.LENGTH\_SHORT).show();

}

}

}

### 图书适配器

将图书数据绑定到RecyclerView，并响应图书项的点击事件。

public class MyBookRecyclerViewAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

private enum ViewType {

CONTENT, // 图书项

HEADER // 分类标题

}

// 点击事件监听器接口

public interface OnItemClickListener {

void onItemClick(View view);

}

private ArrayList<Object> bookList; // 图书列表（包含图书和分类标题）

private OnItemClickListener mItemClickListener; // 点击事件监听器

public MyBookRecyclerViewAdapter(ArrayList<Object> items) {

bookList = items;

}

@NonNull

@Override

public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view;

if (viewType == ViewType.CONTENT.ordinal()) {

// 创建图书项的视图

view = LayoutInflater.from(parent.getContext()).inflate(R.layout.book\_fragment\_item, parent, false);

return new ContentViewHolder(view); // 返回图书项ViewHolder

} else {

// 创建分类标题视图

view = LayoutInflater.from(parent.getContext()).inflate(R.layout.list\_header, parent, false);

return new HeaderViewHolder(view); // 返回分类标题ViewHolder

}

}

// 绑定数据到视图

@Override

public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {

if (holder.getItemViewType() == ViewType.CONTENT.ordinal()) {

// 绑定图书数据

Book item = (Book) bookList.get(position);

ContentViewHolder contentHolder = (ContentViewHolder) holder;

contentHolder.itemView.setTag(item.getId()); // 设置图书ID作为标签

contentHolder.nameView.setText(item.getName()); // 设置图书名称

contentHolder.authorView.setText(item.getAuthor()); // 设置作者

contentHolder.priceView.setText(String.format("%s元", item.getPrice())); // 设置价格

contentHolder.imgView.setImageURL(item.getImgUrl()); // 设置封面图

// 设置点击事件

contentHolder.itemView.setOnClickListener(v -> mItemClickListener.onItemClick(v));

} else {

// 绑定分类标题

String header = ((Book.Category) bookList.get(position)).getName();

HeaderViewHolder headerHolder = (HeaderViewHolder) holder;

headerHolder.headerTextView.setText(header); // 设置分类标题

}

}

@Override

public int getItemCount() {

return bookList.size();

}

// 判断当前单元格的类型：图书项或分类标题

@Override

public int getItemViewType(int position) {

return bookList.get(position) instanceof Book ? ViewType.CONTENT.ordinal() : ViewType.HEADER.ordinal();

}

// 图书项的ViewHolder

public static class ContentViewHolder extends RecyclerView.ViewHolder {

public final TextView nameView, authorView, priceView;

public final NWImageView imgView;

public ContentViewHolder(View itemView) {

super(itemView);

nameView = itemView.findViewById(R.id.text\_name); // 图书名称

authorView = itemView.findViewById(R.id.text\_author); // 图书作者

imgView = itemView.findViewById(R.id.list\_image); // 图书封面

priceView = itemView.findViewById(R.id.text\_price); // 图书价格

}

}

// 分类标题的ViewHolder

public static class HeaderViewHolder extends RecyclerView.ViewHolder {

public final TextView headerTextView;

public HeaderViewHolder(View itemView) {

super(itemView);

headerTextView = itemView.findViewById(R.id.text\_list\_header); // 分类标题

}

}

// 设置点击事件监听器

public void setOnItemClickListener(OnItemClickListener mItemClickListener) {

this.mItemClickListener = mItemClickListener;

}

}

## 管理员模块设计

### 图书管理界面

显示图书列表并提供按钮跳转到添加、编辑和删除图书功能。

public class AdminActivity extends AppCompatActivity {

private Button btnAddBook, btnEditBook, btnDeleteBook;

private RecyclerView recyclerView;

private MyBookRecyclerViewAdapter adapter;

private BookService bookService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_admin);

bookService = new BookService(this);

btnAddBook = findViewById(R.id.btnAddBook);

btnEditBook = findViewById(R.id.btnEditBook);

btnDeleteBook = findViewById(R.id.btnDeleteBook);

recyclerView = findViewById(R.id.recyclerViewBooks);

// 点击按钮跳转到添加图书界面

btnAddBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, AddBookActivity.class);

startActivity(intent);

});

// 点击按钮跳转到编辑图书界面

btnEditBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, EditBookActivity.class);

startActivity(intent);

});

// 点击按钮跳转到删除图书界面

btnDeleteBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, DeleteBookActivity.class);

startActivity(intent);

});

recyclerView.setLayoutManager(new LinearLayoutManager(this));

refreshBookList(); // 加载图书列表

}

@Override

protected void onResume() {

super.onResume();

refreshBookList(); // 回来时刷新图书列表

}

private void refreshBookList() {

List<Book> bookList = bookService.getList();

ArrayList<Object> mixedList = new ArrayList<>();

// 根据图书分类将图书分组并插入分类标题

for (Book.Category category : Book.Category.values()) {

mixedList.add(category); // 添加分类标题

for (Book b : bookList) {

if (b.getCategory() == category) {

mixedList.add(b); // 添加图书

}

}

}

adapter = new MyBookRecyclerViewAdapter(mixedList);

adapter.setOnItemClickListener(view -> {

int id = (int) view.getTag(); // 获取图书ID

// 跳转到编辑页面并传递图书ID

Intent intent = new Intent(AdminActivity.this, EditBookActivity.class);

intent.putExtra("book\_id", id);

startActivity(intent);

});

recyclerView.setAdapter(adapter);

}

}

### 添加图书

允许管理员输入图书信息并保存到数据库。

public class AddBookActivity extends AppCompatActivity {

private EditText etName, etAuthor, etPrice, etCategory, etISBN, etImgUrl, etDescription;

private Button btnSave;

private BookService bookService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_book);

bookService = new BookService(this);

etName = findViewById(R.id.etName);

etAuthor = findViewById(R.id.etAuthor);

etPrice = findViewById(R.id.etPrice);

etCategory = findViewById(R.id.etCategory);

etISBN = findViewById(R.id.etISBN);

etImgUrl = findViewById(R.id.etImgUrl);

etDescription = findViewById(R.id.etDescription);

btnSave = findViewById(R.id.btnSave);

btnSave.setOnClickListener(v -> {

String name = etName.getText().toString();

String author = etAuthor.getText().toString();

String imgUrl = etImgUrl.getText().toString();

String isbn = etISBN.getText().toString();

String description = etDescription.getText().toString();

int category = Integer.parseInt(etCategory.getText().toString()); // 获取图书分类

double price = Double.parseDouble(etPrice.getText().toString());

// 创建图书对象

Book newBook = new Book();

newBook.setName(name);

newBook.setAuthor(author);

newBook.setImgUrl(imgUrl);

newBook.setISBN(isbn);

newBook.setDescription(description);

newBook.setCategory(Book.Category.values()[category]); // 设置图书类别

newBook.setPrice(price);

// 添加图书到数据库

bookService.addBook(newBook);

Toast.makeText(AddBookActivity.this, "添加成功！", Toast.LENGTH\_SHORT).show();

finish(); // 返回

});

}

}

### 编辑图书

允许管理员根据图书ID修改图书信息。

public class EditBookActivity extends AppCompatActivity {

private EditText etId, etName, etAuthor, etPrice, etCategory, etISBN, etImgUrl, etDescription;

private Button btnUpdate;

private BookService bookService;

private int bookId;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_book);

bookService = new BookService(this);

etId = findViewById(R.id.etId);

etName = findViewById(R.id.etName);

etAuthor = findViewById(R.id.etAuthor);

etPrice = findViewById(R.id.etPrice);

etCategory = findViewById(R.id.etCategory);

etISBN = findViewById(R.id.etISBN);

etImgUrl = findViewById(R.id.etImgUrl);

etDescription = findViewById(R.id.etDescription);

btnUpdate = findViewById(R.id.btnUpdate);

// 获取图书ID并加载图书数据

bookId = getIntent().getIntExtra("book\_id", -1);

if (bookId != -1) {

loadBookData(bookId);

}

btnUpdate.setOnClickListener(v -> saveChanges());

}

private void loadBookData(int id) {

Book book = bookService.getBook(id);

if (book != null) {

etId.setText(String.valueOf(book.getId()));

etId.setEnabled(false); // 禁止修改ID

etName.setText(book.getName());

etAuthor.setText(book.getAuthor());

etCategory.setText(String.valueOf(book.getCategory().ordinal()));

etISBN.setText(book.getISBN());

etImgUrl.setText(book.getImgUrl());

etDescription.setText(book.getDescription());

etPrice.setText(String.valueOf(book.getPrice()));

}

}

private void saveChanges() {

// 获取表单数据并更新图书信息

String name = etName.getText().toString();

String author = etAuthor.getText().toString();

String imgUrl = etImgUrl.getText().toString();

String isbn = etISBN.getText().toString();

String description = etDescription.getText().toString();

int category = Integer.parseInt(etCategory.getText().toString());

double price = Double.parseDouble(etPrice.getText().toString());

Book book = new Book();

book.setId(bookId); // 保持原ID

book.setName(name);

book.setAuthor(author);

book.setImgUrl(imgUrl);

book.setISBN(isbn);

book.setDescription(description);

book.setCategory(Book.Category.values()[category]);

book.setPrice(price);

bookService.updateBook(book); // 更新图书数据

Toast.makeText(EditBookActivity.this, "修改成功！", Toast.LENGTH\_SHORT).show();

finish();

}

}

### 删除图书

允许管理员根据ISBN删除图书。

public class DeleteBookActivity extends AppCompatActivity {

private EditText etISBN;

private Button btnDelete;

private BookService bookService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_delete\_book);

bookService = new BookService(this);

etISBN = findViewById(R.id.etISBN);

btnDelete = findViewById(R.id.btnDelete);

btnDelete.setOnClickListener(v -> {

String isbn = etISBN.getText().toString().trim();

if (isbn.isEmpty()) {

Toast.makeText(DeleteBookActivity.this, "请输入ISBN", Toast.LENGTH\_SHORT).show();

return;

}

// 删除图书

bookService.deleteBookByISBN(isbn);

Toast.makeText(DeleteBookActivity.this, "删除成功！", Toast.LENGTH\_SHORT).show();

finish();

});

}

}

### 图书列表展示

展示图书数据，支持分类显示。

public class AdminActivity extends AppCompatActivity {

private Button btnAddBook, btnEditBook, btnDeleteBook;

private RecyclerView recyclerView;

private MyBookRecyclerViewAdapter adapter;

private BookService bookService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_admin);

// 初始化服务

bookService = new BookService(this);

// 获取按钮

btnAddBook = findViewById(R.id.btnAddBook);

btnEditBook = findViewById(R.id.btnEditBook);

btnDeleteBook = findViewById(R.id.btnDeleteBook);

recyclerView = findViewById(R.id.recyclerViewBooks);

// 添加图书按钮点击事件，跳转到添加图书页面

btnAddBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, AddBookActivity.class);

startActivity(intent);

});

// 编辑图书按钮点击事件，跳转到编辑图书页面

btnEditBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, EditBookActivity.class);

startActivity(intent);

});

// 删除图书按钮点击事件，跳转到删除图书页面

btnDeleteBook.setOnClickListener(v -> {

Intent intent = new Intent(AdminActivity.this, DeleteBookActivity.class);

startActivity(intent);

});

// 设置RecyclerView布局管理器

recyclerView.setLayoutManager(new LinearLayoutManager(this));

// 刷新图书列表

refreshBookList();

}

@Override

protected void onResume() {

super.onResume();

refreshBookList(); // 回到AdminActivity时刷新图书列表

}

private void refreshBookList() {

List<Book> bookList = bookService.getList(); // 获取所有图书数据

ArrayList<Object> mixedList = new ArrayList<>();

// 根据图书的类别对图书进行分组

for (Book.Category category : Book.Category.values()) {

mixedList.add(category); // 添加分类标题

for (Book b : bookList) {

if (b.getCategory() == category) {

mixedList.add(b); // 添加该分类下的图书

}

}

}

// 设置适配器并绑定数据

adapter = new MyBookRecyclerViewAdapter(mixedList);

adapter.setOnItemClickListener(view -> {

int id = (int) view.getTag(); // 获取图书ID

// 跳转到编辑页面并传递图书ID

Intent intent = new Intent(AdminActivity.this, EditBookActivity.class);

intent.putExtra("book\_id", id);

startActivity(intent);

});

// 设置RecyclerView的适配器

recyclerView.setAdapter(adapter);

}

}

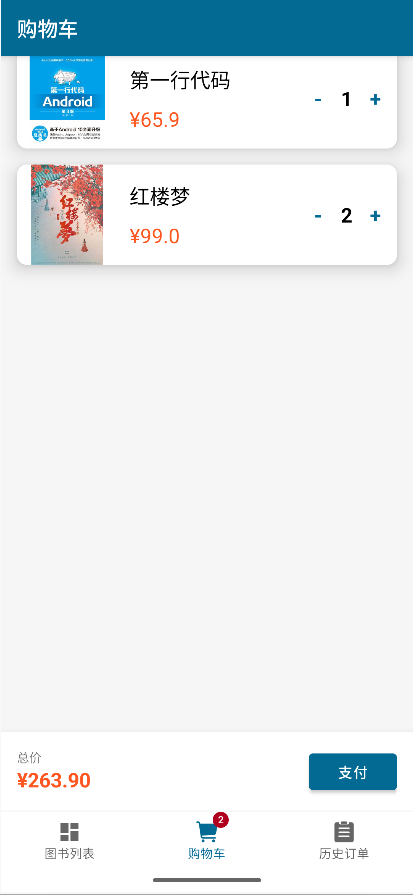
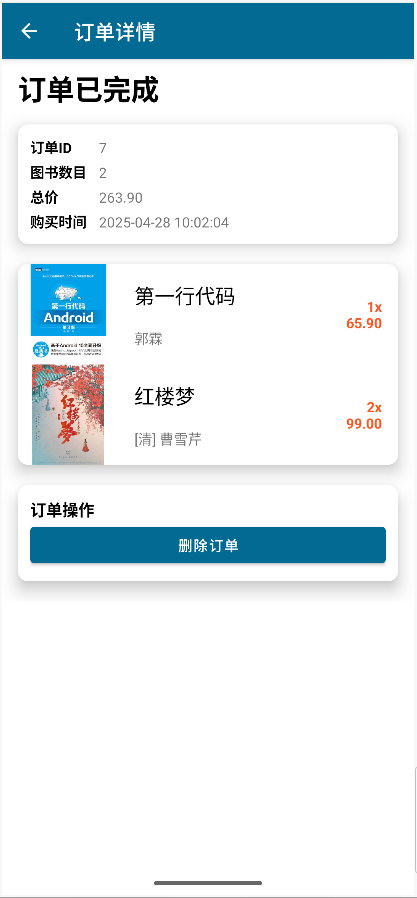
三、实际演示

用户登录、注册：用户注册后会自动登录并跳转到主页



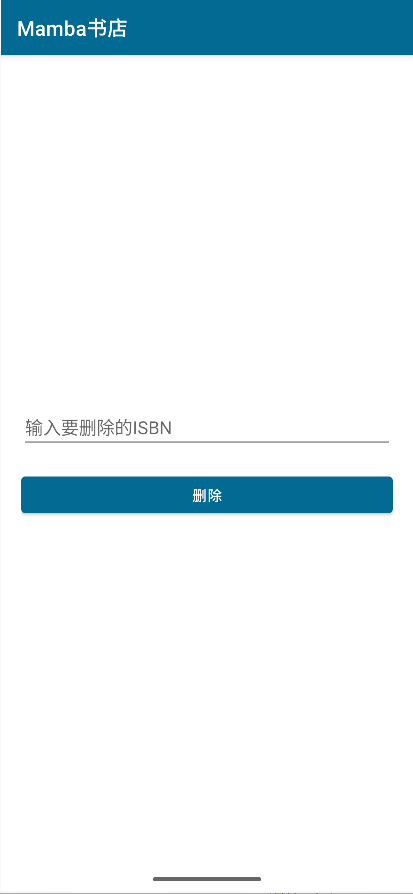
用户主界面：用户可选择图书并添加到购物车，付款后会自动清空购物车，付款信息在历史订单中



管理员界面：可选择图书并修改该图书信息（内容自动填充），也可根据书的ID来修改图书信息；可以添加、删除图书，主页可滑动查看所有书籍。



四、总结及参考文献

这学期学习了Android开发，通过这次大作业让我受益匪浅，之前一直都是写老师布置的实验，都是局部功能，现在要实现一个完整的应用，第一时间不知道如何下手，但好在经过不懈努力开发完成了。

在这次图书管理应用的开发过程中，我通过实际动手编程，进一步理解了Android开发的核心概念，并掌握了如何设计和实现一个完整的Android应用。这次作业不仅让我熟悉了Android 应用开发，还让我在解决实际问题时，深入思考了如何提高应用的性能、可用性和用户体验。在开发过程中我也遇到很多问题，得益于现在AI的发展，那些问题都得到了解决。我目前觉得项目还是过于简略，没有完整的支付系统、评论区、个人信息关联、账号注销、管理员页面过于简陋、管理员无法操作用户数据、没有实际的发货功能等，在未来我会完善这些功能，希望这个APP能真正的可以被使用。

参考文献

[1] 郭霖. \*第一行代码：Android\* (第2版). 北京: 人民邮电出版社, 2016.

[2] Gemeaux-Y, "Android studio—SQlite数据库注册登录" CSDN. [Online]. Available: https://blog.csdn.net/genijmni/article/details/106626300. [Accessed: Apr. 21, 2025].

[3] Teacher.Hu, "Android SQlite数据库使用详解" CSDN. [Online]. Available: https://blog.csdn.net/huweiliyi/article/details/105461725. [Accessed: Apr. 23, 2025].

[4] 编程乐学, "基于Android Studio如何实现 购物商城 案例（简单易上手）" CSDN. [Online]. Available: https://blog.csdn.net/qq\_29823791/article/details/135014530. [Accessed: Apr. 23, 2025].

[5] 郭晟玮, "【Android教程】适配器 Adapter" CSDN. [Online]. Available: https://blog.csdn.net/u014316335/article/details/137771000. [Accessed: Apr. 23, 2025].

[6] jun\_tong, "购物车实现单选、多选、全选和一些fragment的布局" CSDN. [Online]. Available: https://blog.csdn.net/jun\_tong/article/details/80751616. [Accessed: Apr. 23, 2025].