程序设计训练——大作业报告

项目名称：衔书（Xianshu 或 Tsira News）

技术栈：Java

开发者：经 32-计 38 李垚 2023011628

概述

本项目命名为「衔书」（Xianshu 或 Tsira News），命名灵感来自中国古代青鸟衔文的神话传说，代表应用能够像青鸟一样，将世界各地的讯息带到用户的手中。[1]

基本功能

- 新闻爬取与分类展示
- 新闻浏览（文本、图片、视频、元数据）
- AI 摘要及其本地存储
- 基于关键词、类别和时间的搜索
- 分类编辑
- 历史记录、收藏及其本地存储
- 上拉获取、下拉刷新

Bonus Feature

- 基于缓存机制和单例 `OkHttp` 的高效爬取
- **完善的本地化多账号系统**（登录、注册、记住密码、修改密码、个人中心、一键退出登录，历史记录、收藏、分类编辑等均分账号存储，实现对用户的隔离处理）
- 美观现代的用户界面
- 视频/图片位于同一视图，节省空间且美观
- 富有设计感的应用 logo
- 丝滑的启动页特效
- 基于 `Markwon` 库的 Markdown 格式关于界面
- ……

代码结构

我们只就 `app/build/src/main` 做介绍。

- `assets`：其他资源文件夹
  - `about.md`：关于页面的源文件
  - `NewsDetailsActivity.txt`：这是我原本的 `NewsDetailsActivity`，使用的是 `WebView` 和基于本地文件访问权限的缓存方法。但是由于不符合项目要求含泪返工，这个文件权当留个纪念了。
- `java/com.java.liyao`：项目的主要文件夹
  - `adapter`：适配器
    - `NewsListAdapter`：新闻列表适配器
    - `ImagePagerAdapter`：图片/视频适配器
  - `db`：SQLite 数据库助手
    - `AiSummaryDbHelper`：AI 摘要

        - `CatPrefDbHelper`：列表偏好（分用户）
        - `HistoryDbHelper`：历史记录（分用户）
        - `LikeDbHelper`：收藏列表（分用户）
        - `UserDbHelper`：用户信息
    - `entity`：数据库实体
        - `AiSummaryInfo`
        - `CatPrefInfo`
        - `HistoryInfo`
        - `UserInfo`
    - `AboutActivity`：关于
    - `AccountActivity`：账户信息
    - `AiSummary`：AI 摘要工具类
    - `AiSummaryRetInfo`：AI 摘要信息，将 JSON 返回值转换为摘要
    - `CatTabFragment`：分类标签页布局
    - `EditCategoriesActivity`：编辑分类偏好
    - `HistoryActivity`：历史记录
    - `LikeActivity`：收藏列表
    - `LoginActivity`：登录
    - `MainActivity`：主活动
    - `NewsDetailsActivity`：详情页面
    - `NewsInfo`：新闻信息
    - `OkHttpSingleton`：HTTP 请求器的单例类
    - `RegisterActivity`：注册

    - `SearchActivity`：搜索页面
    - `SearchResultActivity`：搜索结果
    - `WelcomeActivity`：启动页
- `res`：资源文件夹
- `AndroidManifest.xml`

具体实现

新闻爬取

新闻爬取作为实现的关键部分，经过了我多次的优化调整，最终形成了如下图所示的运行方法：

单例类

之所以使用 `OkHttp` 的单例包装类，是为了规避反复创建 HTTP 请求器带来的开销。单例类的代码如下：

```java
package com.java.liyao;

import java.util.concurrent.TimeUnit;
import okhttp3.OkHttpClient;

public class OkHttpSingleton {
    private static OkHttpClient okHttpClient;

    private OkHttpSingleton() {
        // 私有构造函数，防止外部实例化
    }
    public static OkHttpClient getInstance() {
        if (okHttpClient == null) {
            synchronized (OkHttpSingleton.class) {
                if (okHttpClient == null) {
                    okHttpClient = new
OkHttpClient.Builder()
                            .connectTimeout(10,
TimeUnit.SECONDS)
                            .readTimeout(10,
TimeUnit.SECONDS)
                            .writeTimeout(10,
TimeUnit.SECONDS)
                            .build();
                }
            }
        }
        return okHttpClient;
    }
}
```

爬取过程

爬取过程使用 `fetcher` 方法完成。这里展示的并不完全是爬取，还涉及到后续缓存和上拉的部分。

```
 private void fetcher() throws UnsupportedEncodingException
{
    String cachedData = getCache();
    if (cachedData != null && currentPage == 1) {
        processData(cachedData);
        return;
    }
    OkHttpClient okHttpClient =
OkHttpSingleton.getInstance(); // Use the singleton
instance
    // 《新闻》（指四年前）
    String baseUrl =
"https://api2.newsminer.net/svc/news/queryNewsList?
size=15&startDate=2020-07-01&endDate=2024-08-
30&words=&categories=";
    String encodedCatT = Objects.equals(catT, "全部") ? "" :
URLEncoder.encode(catT, StandardCharsets.UTF_8.toString());
    String url = baseUrl + encodedCatT + "&page=" +
currentPage;
    Request request = new Request.Builder()
            .url(url)
            .get()
            .build();
    Call call = okHttpClient.newCall(request);
    call.enqueue(new Callback() {
        @Override
        public void onFailure(@NonNull Call call, @NonNull
IOException e) {
            Log.d("NetworkError", "onFailure: " +
e.toString());
            mHandler.post(() → {
                isLoading = false;
            });
        }

        @Override
        public void onResponse(@NonNull Call call, @NonNull
Response response) throws IOException {
```

```java
            if (response.isSuccessful() && response.body() != null) {
                String data = response.body().string();
                NewsInfo newsInfo = new Gson().fromJson(data, NewsInfo.class);
                if (newsInfo.getData().isEmpty()) {
                    mHandler.post(() -> Toast.makeText(getActivity(), "我们可能需要到更加古老的年代爬取新闻……", Toast.LENGTH_LONG).show());
                } else {
                    mHandler.post(() -> processData(data));
                }
            } else {
                Log.d("NetworkError", "Response not successful or body is null");
                mHandler.post(() -> isLoading = false);
            }
        }
    });
}
```

`JSON` 数据解析

爬取下的新闻本质是 `JSON` 字符串，并不能直接用于编程。

我们采用广受好评的 Gson 库来处理数据。GsonFormatPlus 插件可以根据 JSON 的格式直接生成对应的类。

```java
package com.java.liyao;

import static org.apache.commons.codec.digest.DigestUtils.sha256Hex;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

@lombok.NoArgsConstructor
@lombok.Data
public class NewsInfo {


 @com.fasterxml.jackson.annotation.JsonProperty("pageSize")
    private String pageSize;
    @com.fasterxml.jackson.annotation.JsonProperty("total")
    private Integer total;
    @com.fasterxml.jackson.annotation.JsonProperty("data")
    private List<DataDTO> data;

 @com.fasterxml.jackson.annotation.JsonProperty("currentPag
e")
    private String currentPage;

    public List<DataDTO> getData() {
        return data;
    }

    public void generateUniqueID() {
        for (DataDTO data : data) {
            data.generateSingleUniqueID();
        }
        // Log.d("UniqueIDSuccessfully", "generateUniqueID:
生成唯一标识符成功! ");
```

```
    }

    @lombok.NoArgsConstructor
    @lombok.Data
    public static class DataDTO implements Serializable {

 @com.fasterxml.jackson.annotation.JsonProperty("image")
        private String image;

 @com.fasterxml.jackson.annotation.JsonProperty("publishTim
e")
        private String publishTime;

 @com.fasterxml.jackson.annotation.JsonProperty("keywords")
        private List<KeywordsDTO> keywords;

 @com.fasterxml.jackson.annotation.JsonProperty("language")
        private String language;

 @com.fasterxml.jackson.annotation.JsonProperty("video")
        private String video;

 @com.fasterxml.jackson.annotation.JsonProperty("title")
        private String title;

 @com.fasterxml.jackson.annotation.JsonProperty("when")
        private List<WhenDTO> when;

 @com.fasterxml.jackson.annotation.JsonProperty("content")
        private String content;

 @com.fasterxml.jackson.annotation.JsonProperty("openRels")
        private String openRels;
 @com.fasterxml.jackson.annotation.JsonProperty("url")
        private String url;

 @com.fasterxml.jackson.annotation.JsonProperty("persons")
        private List<PersonsDTO> persons;

 @com.fasterxml.jackson.annotation.JsonProperty("newsID")
        private String newsID;

 @com.fasterxml.jackson.annotation.JsonProperty("crawlTime"
)
        private String crawlTime;

 @com.fasterxml.jackson.annotation.JsonProperty("organizati
ons")
        private List<OrganizationsDTO> organizations;

 @com.fasterxml.jackson.annotation.JsonProperty("publisher"
)
        private String publisher;

 @com.fasterxml.jackson.annotation.JsonProperty("locations"
)
        private List<LocationsDTO> locations;

 @com.fasterxml.jackson.annotation.JsonProperty("where")
        private List<?> where;

 @com.fasterxml.jackson.annotation.JsonProperty("scholars")
        private List<?> scholars;

 @com.fasterxml.jackson.annotation.JsonProperty("category")
```

```
        private String category;

 @com.fasterxml.jackson.annotation.JsonProperty("who")
        private List<WhoDTO> who;

        public String getVideo() {
            return video;
        }

        public String getUniqueID() {
            return uniqueID;
        }

        private String uniqueID;

        public String getAiSummary() {
            return aiSummary;
        }

        public void setAiSummary(String aiSummary) {
            this.aiSummary = aiSummary;
        }

        private String aiSummary = "";

        private void generateSingleUniqueID() {
            String source = this.title + this.publishTime +
this.publisher;
            this.uniqueID = sha256Hex(source);
        }

        // Getter
        public String getPublisher() {
            return publisher;
        }

        public String getTitle() {
            return title;
        }

        public String getContent() {
            return content;
        }

        public String getPublishTime() {
            return publishTime;
        }

        public String getUrl() {
            return url;
        }

        public List<String> getImage() {
            if (image == null || image.isEmpty()) {
                return Collections.emptyList();
            }

            // 去除字符串两端的大括号和方括号
            String trimmedImage = image.replaceAll("^[\\[\\
{]|[\\]\\}]$", "");

            // 使用正则表达式匹配URL，考虑到URL可能包含逗号，我们使
用非贪婪匹配
            List<String> imageList = new ArrayList<>();
            Pattern pattern = Pattern.compile("
(https?://[^,\"'\\s]+)");
```

```
            Matcher matcher =
pattern.matcher(trimmedImage);

            while (matcher.find()) {
                imageList.add(matcher.group(1).trim());
            }

            return imageList;
        }


        public List<KeywordsDTO> getKeywords() {
            return keywords;
        }

        public List<WhenDTO> getWhen() {
            return when;
        }

        public List<PersonsDTO> getPersons() {
            return persons;
        }

        public List<OrganizationsDTO> getOrganizations() {
            return organizations;
        }

        public List<LocationsDTO> getLocations() {
            return locations;
        }

        public List<WhoDTO> getWho() {
            return who;
        }

        public String getThumbnail() {
            List<String> images = getImage();
            if (!images.isEmpty()) {
                return images.get(0);
            }
            return null;
        }

        @lombok.NoArgsConstructor
        @lombok.Data
        public static class KeywordsDTO implements
Serializable {

 @com.fasterxml.jackson.annotation.JsonProperty("score")
            private Double score;

 @com.fasterxml.jackson.annotation.JsonProperty("word")
            private String word;
        }

        @lombok.NoArgsConstructor
        @lombok.Data
        public static class WhenDTO implements Serializable
{

 @com.fasterxml.jackson.annotation.JsonProperty("score")
            private Double score;

 @com.fasterxml.jackson.annotation.JsonProperty("word")
            private String word;
        }
        @lombok.NoArgsConstructor
        @lombok.Data
        public static class PersonsDTO implements
Serializable {

 @com.fasterxml.jackson.annotation.JsonProperty("count")
            private Double count;

 @com.fasterxml.jackson.annotation.JsonProperty("linkedURL"
)
            private String linkedURL;

 @com.fasterxml.jackson.annotation.JsonProperty("mention")
            private String mention;
        }

        @lombok.NoArgsConstructor
        @lombok.Data
        public static class OrganizationsDTO implements
Serializable {

 @com.fasterxml.jackson.annotation.JsonProperty("count")
            private Double count;

 @com.fasterxml.jackson.annotation.JsonProperty("linkedURL"
)
            private String linkedURL;

 @com.fasterxml.jackson.annotation.JsonProperty("mention")
            private String mention;
        }
        @lombok.NoArgsConstructor
        @lombok.Data
        public static class LocationsDTO implements
Serializable {

 @com.fasterxml.jackson.annotation.JsonProperty("count")
            private Double count;

 @com.fasterxml.jackson.annotation.JsonProperty("linkedURL"
)
            private String linkedURL;

 @com.fasterxml.jackson.annotation.JsonProperty("mention")
            private String mention;
        }

        @lombok.NoArgsConstructor
        @lombok.Data
        public static class WhoDTO implements Serializable
{

 @com.fasterxml.jackson.annotation.JsonProperty("score")
            private Double score;

 @com.fasterxml.jackson.annotation.JsonProperty("word")
            private String word;
        }
    }
}
```

当然，新闻本身的类 `dataDTO` 及其子类需要进行序列化处理，以便在活动之间进行传递。

新闻列表适配器

新闻列表适配器是一个关键的类，涉及到各种对新闻对象的处理，包括设置列表、添加数据、加载图片、历史记录标灰、点击进入详情页等多种功能。

```java
package com.java.liyao.adapter;

import android.annotation.SuppressLint;
import android.content.Context;
// import android.util.Log;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.java.liyao.NewsInfo;
import com.java.liyao.R;
import com.bumptech.glide.annotation.GlideModule;
import com.bumptech.glide.module.AppGlideModule;
import com.java.liyao.db.HistoryDbHelper;
import com.java.liyao.db.LikeDbHelper;
import com.java.liyao.entity.UserInfo;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
```

```
public class NewsListAdapter extends
RecyclerView.Adapter<NewsListAdapter.MyHold> {

    private List<NewsInfo.DataDTO> mDataDTOList = new
ArrayList<>();
    private Context mContext;

    UserInfo userInfo = UserInfo.getUserinfo();
    String eml = userInfo == null ? null :
userInfo.getUser_email();

    public NewsListAdapter(Context context) {
        this.mContext = context;
    }

    public void setListData(List<NewsInfo.DataDTO> list) {
        this.mDataDTOList = list;
        notifyDataSetChanged();
    }

    public boolean isViewed(String unique_id) {
        return
HistoryDbHelper.getInstance(mContext).searchHistory(unique_
id, eml);
    }

    public boolean isLiked(String unique_id) {
        return
LikeDbHelper.getInstance(mContext).searchLike(unique_id,
eml);
    }
    @NonNull
    @Override
    public MyHold onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.n
ew_item, parent, false);
        return new MyHold(view);
    }

    @SuppressLint("CheckResult") // 断无此疏
    @Override
    public void onBindViewHolder(@NonNull MyHold holder,
int position) {
        NewsInfo.DataDTO dataDTO =
mDataDTOList.get(position);

        holder.author_name.setText(dataDTO.getPublisher());
        holder.title.setText(dataDTO.getTitle());
        holder.date.setText(dataDTO.getPublishTime());

        if (isViewed(dataDTO.getUniqueID())) {
            // Log.d("Viewed", "onBindViewHolder: 已浏览");
            // 为标题设置特殊颜色

 holder.title.setTextColor(mContext.getResources().getColor
(R.color.grey));
            // Log.d("AlreadyViewed", "onBindViewHolder: "
+ dataDTO.getTitle() + "已浏览");
        }
        else {
```

```
holder.title.setTextColor(mContext.getResources().getColor(
R.color.black));
        }

        // 使用Glide加载图片，确保了图片URL的正确性
        if (dataDTO.getImage() ≠ null &&
!dataDTO.getImage().isEmpty()) {
            Glide.with(mContext)
                    .load(dataDTO.getThumbnail())
                    .override(300, 300) // 限制图片的尺寸
                    .error(R.drawable.default_holder)
                    .into(holder.thumbnail_pic_s);
        } else {

 holder.thumbnail_pic_s.setImageResource(R.drawable.default
_holder);
        }

        holder.itemView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (onItemClickListener ≠ null) {

 onItemClickListener.onItemClick(dataDTO, position);
                }
            }
        });

        Animation animation =
android.view.animation.AnimationUtils.loadAnimation(mContex
t, R.anim.item_animation_float_up);
```

```
            holder.itemView.startAnimation(animation);
        }

        @Override
        public int getItemCount() {
            return mDataDTOList.size();
        }

        static class MyHold extends RecyclerView.ViewHolder {
            ImageView thumbnail_pic_s;
            TextView title;
            TextView author_name;
            TextView date;

            public MyHold(@NonNull View itemView) {
                super(itemView);
                thumbnail_pic_s =
    itemView.findViewById(R.id.thumbnail_pic_s);
                title = itemView.findViewById(R.id.title);
                author_name =
    itemView.findViewById(R.id.author_name);
                date = itemView.findViewById(R.id.date); // 确保
    这里的ID与布局文件中的ID匹配
            }
        }

        private OnItemClickListener onItemClickListener;

        public void setOnItemClickListener(OnItemClickListener
    onItemClickListener) {
            this.onItemClickListener = onItemClickListener;
        }
        public interface OnItemClickListener {
            void onItemClick(NewsInfo.DataDTO dataDTO, int
    position);
        }

        public void addListData(List<NewsInfo.DataDTO> newData)
    {
            int startPosition = this.mDataDTOList.size();
            this.mDataDTOList.addAll(newData);
            notifyItemRangeInserted(startPosition,
    newData.size());
        }

        public void clearData() {
            this.mDataDTOList.clear();
            notifyDataSetChanged();
        }
    }
```

消息句柄

为了统一处理消息，我们采用一个句柄进行请求。

```
mHandler = new Handler(Looper.getMainLooper()) {
    @Override
    public void handleMessage(@NonNull Message msg) {
        super.handleMessage(msg);
        if (msg.what == 200) {
            String data = (String) msg.obj;
            NewsInfo newsInfo = new Gson().fromJson(data,
NewsInfo.class);
            newsInfo.generateUniqueID();
            if (newsListAdapter != null) {
                if (currentPage == 1) {

 newsListAdapter.setListData(newsInfo.getData());
                } else {

 newsListAdapter.addListData(newsInfo.getData());
                }
                newsListAdapter.notifyDataSetChanged();
            } else {
                Toast.makeText(getActivity(), "获取数据失
败! ", Toast.LENGTH_SHORT).show();
            }
        }
        isLoading = false;
    }
};
```

历史记录标灰

历史记录的具体实现是 SQLite 数据库，我们留到后面讲。对于这个需求，我们在适配器中添加一个检查代码即可：

```
if (isViewed(dataDTO.getUniqueID())) {
    // Log.d("Viewed", "onBindViewHolder: 已浏览");
    // 为标题设置特殊颜色

 holder.title.setTextColor(mContext.getResources().getColor
(R.color.grey));
    // Log.d("AlreadyViewed", "onBindViewHolder: " +
dataDTO.getTitle() + "已浏览");
}
else {

 holder.title.setTextColor(mContext.getResources().getColor
(R.color.black));
}
```

配置完成后，我们点击某一条新闻然后回到主页，只需刷新即可看到效果。

上/下拉

由于我在 XML 中使用了 `SwipeRefreshLayout`，因此下拉刷新变得非常简单。

```
swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
    @Override
    public void onRefresh() {
        currentPage = 1;
        newsListAdapter.clearData();
        clearCache(); // 清除缓存
        try {
            fetcher();
        } catch (UnsupportedEncodingException e) {
            Log.e("EncodingError", "Unsupported Encoding
Exception", e);
        }
        swipeRefreshLayout.setRefreshing(false);
    }
});
```

上拉需要一点简单的技巧。我们先定义 `isLoading` 和 `currentPage` 两个变量来跟踪加载状态和当前页码，然后在 `RecyclerView` 增加一个滚动监听器：

```
// 添加滚动监听器以实现加载更多
newsList.addOnScrollListener(new
RecyclerView.OnScrollListener() {
    @Override
    public void onScrolled(@NonNull RecyclerView
recyclerView, int dx, int dy) {
        super.onScrolled(recyclerView, dx, dy);
        int visibleItemCount =
layoutManager.getChildCount();
        int totalItemCount = layoutManager.getItemCount();
        int firstVisibleItemPosition =
layoutManager.findFirstVisibleItemPosition();

        if (!isLoading && (visibleItemCount +
firstVisibleItemPosition) ⩾ totalItemCount - 5
                && firstVisibleItemPosition ⩾ 0 && dy > 0)
{
            loadMoreItems();
        }
    }
    });
```

这个监听器会计算是否触发加载事件。为了加快加载速度，我们在离底部还有 5 个时就开始加载下一篇。

缓存机制

为了避免大量的重复请求，我使用 `SharedPreferences` 实现了一个简单的缓存机制。该机制会以五分钟为期限缓存请求到的数据，如果 `fetcher` 时缓存没有失效，则直接调取缓存。当然，可以使用下拉强制刷新。

这个缓存机制还不算完善，但是已经可以起到一部分作用了。

历史记录与收藏列表

历史记录与收藏列表的实现基本一致，其实体都是 `HistoryInfo` 类。为了方便地存储各种信息，我们将新闻信息直接打回原形，变成一个 `news_json` 串：

```
package com.java.liyao.entity;

public class HistoryInfo {
    private int history_id;
    private String user_email;
    private String unique_id;
    private String news_json;

    public HistoryInfo(int history_id, String user_email,
String unique_id, String news_json) {
        this.history_id = history_id;
        this.user_email = user_email;
        this.unique_id = unique_id;
        this.news_json = news_json;
    }

    public int getHistory_id() {
        return history_id;
    }

    public void setHistory_id(int history_id) {
        this.history_id = history_id;
    }

    public String getUser_email() {
        return user_email;
    }

    public void setUser_email(String user_email) {
        this.user_email = user_email;
    }

    public String getUnique_id() {
        return unique_id;
    }

    public void setUnique_id(String unique_id) {
        this.unique_id = unique_id;
    }

    public String getNews_json() {
        return news_json;
    }

    public void setNews_json(String news_json) {
        this.news_json = news_json;
    }
}
```

而对于新闻数据的存储，我们采用 SQLite。SQLite 内嵌于 Android 系统，支持 SQL，同时只需要一个文件，非常适合本地化的数据保存。

下面是 `HistoryDbHelper`：

```
package com.java.liyao.db;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import androidx.annotation.Nullable;

import com.java.liyao.entity.HistoryInfo;

import java.util.ArrayList;
import java.util.List;

public class HistoryDbHelper extends SQLiteOpenHelper {
    private static HistoryDbHelper historyDbHelper;
    private static final String DB_NAME = "history.db";
    private static final int DB_VERSION = 1;
    public HistoryDbHelper(@Nullable Context context,
@Nullable String name, @Nullable
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    public synchronized static HistoryDbHelper
getInstance(Context context) {
        if (null == historyDbHelper) {
            historyDbHelper = new HistoryDbHelper(context,
DB_NAME, null, DB_VERSION);
        }
        return historyDbHelper;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // 创建表
        db.execSQL("create table history_table(history_id
integer primary key autoincrement, " +
                "user_email text," +       // 按照我的规定，我
们使用用户邮箱作为用户的唯一标识
                "unique_id text," +
                "news_json text" +
                ")");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase,
int i, int i1) {
        // 先不写
    }
    public int addHistory(String user_email, String
unique_id, String news_json) {
        if (!searchHistory(unique_id, user_email)) {
            SQLiteDatabase db = getWritableDatabase();
            ContentValues values = new ContentValues();

            values.put("user_email", user_email);
            values.put("unique_id", unique_id);
            values.put("news_json", news_json);
            String nullColumnHack = "values(null,?,?,?)";

            int insert = (int) db.insert("history_table",
nullColumnHack, values);
            db.close();
            // Log.d("SuccessfullyAddToHistory", "onCreate:
已添加到历史记录");
            return insert;
        }
        return 0;
    }

    // 这个返回的是 HistoryInfo
    @SuppressLint("Range")
    public List<HistoryInfo> getHistory(String ue) {
        SQLiteDatabase db = getReadableDatabase();
        List<HistoryInfo> list = new ArrayList<>();
        String sql;
        Cursor cursor;
        if (ue == null) {
            sql = "select
history_id,user_email,unique_id,news_json  from
history_table";
            cursor = db.rawQuery(sql, null);
        } else {
            sql = "select
history_id,user_email,unique_id,news_json  from
history_table where user_email=?";
            cursor = db.rawQuery(sql, new String[]{ue});
        }
        while (cursor.moveToNext()) {
            int history_id =
cursor.getInt(cursor.getColumnIndex("history_id"));
            String user_email =
cursor.getString(cursor.getColumnIndex("user_email"));
            String unique_id =
cursor.getString(cursor.getColumnIndex("unique_id"));
            String news_json =
cursor.getString(cursor.getColumnIndex("news_json"));
            list.add(new HistoryInfo(history_id,
user_email, unique_id, news_json));
        }
        cursor.close();
        db.close();
        return list;
    }

    @SuppressLint("Range")
    public boolean searchHistory(String uk, String ue) {
        SQLiteDatabase db = getReadableDatabase();
        String sql;
        Cursor cursor;
        if (ue == null) {
            sql = "select history_id, user_email,
unique_id, news_json from history_table where unique_id=?";
            cursor = db.rawQuery(sql, new String[]{uk});
        } else {
            sql = "select history_id, user_email, unique_id, news_json from history_table where unique_id=? and user_email=?";
            cursor = db.rawQuery(sql, new String[]{uk, ue});
        }
        boolean result = cursor.getCount() > 0;
        cursor.close();
        db.close();
        Log.d("SearchHistory", "searchHistory: unique_id=" + uk + ", user_email=" + ue + ", result=" + result);
        return result;
    }

    // 调试性功能
    public int deleteAllHistory() {
        SQLiteDatabase db = getWritableDatabase();
        int rowsDeleted = db.delete("history_table", null, null);
        db.close();
        return rowsDeleted;
    }
}
```

历史记录本身是侧边栏中的一个元素，因此我们需要在 `MainActivity` 中绑定点击事件：

```
nav_view.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull
MenuItem item) {

    if (item.getItemId() == R.id.nav_history) {
        Intent intent = new Intent(MainActivity.this,
HistoryActivity.class);
        startActivity(intent);
    // ...
```

然后我们仿造首页列表写一个历史记录的布局和活动即可。为了保证离线可访问性，可以修改代码，使得所有历史记录都从数据库加载：

```
// 这个和新闻列表也差不多

package com.java.liyao;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import com.google.gson.Gson;
import com.java.liyao.adapter.NewsListAdapter;
import com.java.liyao.db.HistoryDbHelper;
import com.java.liyao.entity.HistoryInfo;
import com.java.liyao.entity.UserInfo;
import java.util.ArrayList;
import java.util.List;

public class HistoryActivity extends AppCompatActivity {
    private RecyclerView newsList;
    private NewsListAdapter newsListAdapter;
    private List<NewsInfo.DataDTO> newsData;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // EdgeToEdge.enable(this);
        setContentView(R.layout.activity_history);

        newsList = findViewById(R.id.newsList);
        newsListAdapter = new NewsListAdapter(this);
        newsList.setAdapter(newsListAdapter);

        newsData = new ArrayList<>();

        // 获取历史记录
        UserInfo userInfo = UserInfo.getUserinfo();
        String eml = userInfo == null ? null :
userInfo.getUser_email();
        List<HistoryInfo> history =
HistoryDbHelper.getInstance(HistoryActivity.this).getHistor
y(eml);

        Gson gson = new Gson();
        for (int i = 0; i < history.size(); i++) {
```

```
 newsData.add(gson.fromJson(history.get(i).getNews_json(),
NewsInfo.DataDTO.class));
        }

        newsListAdapter.setListData(newsData);

        newsListAdapter.setOnItemClickListener((new
NewsListAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(NewsInfo.DataDTO
dataDTO, int position) {
                Intent intent = new
Intent(HistoryActivity.this, NewsDetailsActivity.class);
                intent.putExtra("dataDTO", dataDTO);
                startActivity(intent);
            }
        }));


 findViewById(R.id.history_toolbar).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //
HistoryDbHelper.getInstance(HistoryActivity.this).deleteAll
History(); // 先都清除一下，方便后续测试
                finish();
            }
        });
    }
}
```

收藏功能的实现与之基本一致，只是需要在详情页的底端添加一个按钮并绑定点击事件。

新闻详情

详情页依然是一个比较大的工程，大致可以分为顶部工具栏、媒体加载、元数据加载、AI 摘要、正文、收藏按钮几部分。

顶部工具栏

这个功能在项目中随处可见，具体来说就是一个标题加上返回按钮。我们只在这里说一次实现方法。

工具栏的实现并不难：

```xml
<Toolbar
    android:id="@+id/details_toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/transparent"
    android:navigationIcon="@drawable/comeback"
    android:title="@string/details_title_placeholder">
</Toolbar>
```

然后在活动中设置事件等即可：

```
details_toolbar.setNavigationOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        finish();
    }
});
```

媒体加载

媒体资源包括图片和视频。图片一般是一个字符串化的数组，而视频是字符串。我们首先需要在 `NewsInfo` 中设置合适的处理方法，这里只展示图片处理：

```
public List<String> getImage() {
    if (image == null || image.isEmpty()) {
        return Collections.emptyList();
    }

    // 去除字符串两端的大括号和方括号
    String trimmedImage = image.replaceAll("^[[\\{]|
[]\\}]$", "");

    // 使用正则表达式匹配URL，考虑到URL可能包含逗号，我们使用非贪婪匹
配
    List<String> imageList = new ArrayList<>();
    Pattern pattern = Pattern.compile("
(https?://[^,\"'\\s]+)");
    Matcher matcher = pattern.matcher(trimmedImage);

    while (matcher.find()) {
        imageList.add(matcher.group(1).trim());
    }

    return imageList;
}
```

然后我们写出相应的布局。我将布局写在顶端，使用 `ViewPager2` 实现滚动效果：

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/details_image"
        android:layout_width="match_parent"
        android:layout_height="200dp" />
    <TextView
        android:id="@+id/image_indicator"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@id/details_image"
        android:layout_alignEnd="@id/details_image"
        android:layout_margin="8dp"
        android:background="#80000000"
        android:padding="4dp"
        android:textColor="#FFFFFF"
        android:textSize="14sp" />
</RelativeLayout>
```

然后我们需要适配器。这里是一个相对困难的点，最终的通用适配器如下：

```java
package com.java.liyao.adapter;

import android.content.Context;
import android.net.Uri;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.VideoView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.java.liyao.R;

import java.util.List;

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

    private Context context;
    private List<String> mediaUrls;
    private String videoUrl;

    public ImagePagerAdapter(Context context, List<String>
mediaUrls, String videoUrl) {
        this.context = context;
        this.mediaUrls = mediaUrls;
        this.videoUrl = videoUrl;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(context).inflate(R.layout.image_pager_i
tem, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder
holder, int position) {
        if (position == 0 && videoUrl != null) {
            holder.imageView.setVisibility(View.GONE);
            holder.videoView.setVisibility(View.VISIBLE);

 holder.videoView.setVideoURI(Uri.parse(videoUrl));
            holder.videoView.start();
        } else {
            holder.videoView.setVisibility(View.GONE);
            holder.imageView.setVisibility(View.VISIBLE);
            String imageUrl = mediaUrls.get(position -
(videoUrl != null ? 1 : 0));
            Glide.with(context)
                    .load(imageUrl)
                    .apply(new
RequestOptions().placeholder(R.drawable.default_holder).err
or(R.drawable.error))
                    .into(holder.imageView);
            Log.d("ImageLoader", "Loading image: " +
imageUrl + " Total images: " + mediaUrls.size());
```

```
        }
    }

    @Override
    public int getItemCount() {
        return mediaUrls.size() + (videoUrl ≠ null ? 1 :
0);
    }

    static class ViewHolder extends RecyclerView.ViewHolder
{
        ImageView imageView;
        VideoView videoView;

        ViewHolder(View itemView) {
            super(itemView);
            imageView =
itemView.findViewById(R.id.image_view);
            videoView =
itemView.findViewById(R.id.video_view);
        }
    }
}
```

然后我们在详情页开始加载图片。这里的条件判断有些冗余，但是我决定保留它们。

```
// 初始化ViewPager2
String videoUrl = null;
List<String> imageUrls = dataDTO.getImage();
if (dataDTO.getVideo() ≠ null &&
!dataDTO.getVideo().isEmpty()) {
    videoUrl = dataDTO.getVideo();
}
// videoUrl = "http://vjs.zencdn.net/v/oceans.mp4";

// 此时的 videoURL 要么是 null，要么是一个非空字符串

Log.d("ImageLoader", "onCreate: " + dataDTO.getTitle() +
imageUrls.toString());
if ((imageUrls != null && !imageUrls.isEmpty()) ||
(videoUrl != null)) {
    imagePagerAdapter = new ImagePagerAdapter(this,
imageUrls, videoUrl);
    details_image.setAdapter(imagePagerAdapter);

 details_image.setOrientation(ViewPager2.ORIENTATION_HORIZO
NTAL);

    // 设置指示器
    updateIndicator(1, imagePagerAdapter.getItemCount());

    // 注册页面变化回调
    details_image.registerOnPageChangeCallback(new
ViewPager2.OnPageChangeCallback() {
        @Override
        public void onPageSelected(int position) {
            super.onPageSelected(position);
            updateIndicator(position + 1,
imagePagerAdapter.getItemCount());
        }
    });

    image_indicator.setVisibility(View.VISIBLE);
} else {
    // 如果没有图片、视频，隐藏ViewPager2和指示器
    details_image.setVisibility(View.GONE);
    image_indicator.setVisibility(View.GONE);
}
```

附件中有一个测试视频，证明这个机制能够正常工作。测试用的视频 URL 就是注释中的那个。

元数据加载

元数据加载并不算难，只需要调用现成的 Getter 就可以了。下面是布局文件，采用了现代风格的卡片布局。

```
    <androidx.cardview.widget.CardView
        android:id="@+id/info_card"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        app:cardCornerRadius="8dp"
        app:cardElevation="4dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="16dp">

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:layout_marginTop="8dp">
                <ImageView
                    android:layout_width="24dp"
                    android:layout_height="24dp"

android:src="@drawable/baseline_calendar_today_24" />

                <TextView
                    android:id="@+id/card_date"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="8dp"
                    android:text="时间: 1145-1-41 91:98:00"
                    android:textSize="16sp"

android:textColor="@android:color/darker_gray" />

            </LinearLayout>

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:layout_marginTop="4dp">

                <ImageView
                    android:layout_width="24dp"
                    android:layout_height="24dp"
                    android:src="@drawable/baseline_source_24"
/>

                <TextView
                    android:id="@+id/card_source"
```

```xml
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:text="来源：华清大学新闻办公室"
                android:textSize="16sp"

android:textColor="@android:color/darker_gray" />

        </LinearLayout>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="4dp">

            <ImageView
                android:layout_width="24dp"
                android:layout_height="24dp"
                android:src="@drawable/robot" />

            <TextView
                android:id="@+id/ai_summary"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:text="摘要：点击生成AI摘要"
                android:textSize="16sp"

android:textColor="@android:color/darker_gray" />

        </LinearLayout>
```

```
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

内容加载、收藏

略。

搜索功能

由于用户的搜索需求非常多样，并且我们并没有存储全部爬取到的新闻的打算，因此搜索功能的实现与新闻爬取类似。

布局

我们使用 `ConstraintLayout` 带来更大的灵活性。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/search"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/transparent"
    tools:context=".SearchActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/search_toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@android:color/transparent"
        android:elevation="4dp"
        app:layout_constraintTop_toTopOf="parent"
        app:navigationIcon="@drawable/comeback"
        app:title="@string/search_title"
        app:titleTextColor="@color/black" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/search_keyword_layout"

 style="@style/Widget.MaterialComponents.TextInputLayout.Ou
tlinedBox"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

 app:layout_constraintTop_toBottomOf="@id/search_toolbar">


 <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/search_keyword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/hint_keyword"
            android:inputType="text" />

 </com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:id="@+id/date_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:text="@string/date_label"
        android:textColor="@color/black"
        app:layout_constraintStart_toStartOf="parent"

 app:layout_constraintTop_toBottomOf="@id/search_keyword_la
yout" />

    <com.google.android.material.button.MaterialButton
        android:id="@+id/start_date_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="@string/start_date"
        android:textColor="@android:color/darker_gray"

 app:layout_constraintEnd_toStartOf="@id/end_date_button"
        app:layout_constraintStart_toStartOf="parent"

 app:layout_constraintTop_toBottomOf="@id/date_label" />

    <com.google.android.material.button.MaterialButton
        android:id="@+id/end_date_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="16dp"
        android:text="@string/end_date"
        android:textColor="@android:color/darker_gray"
        app:layout_constraintEnd_toEndOf="parent"

 app:layout_constraintStart_toEndOf="@id/start_date_button"

 app:layout_constraintTop_toBottomOf="@id/date_label" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/search_category_layout"

 style="@style/Widget.MaterialComponents.TextInputLayout.Ou
tlinedBox.ExposedDropdownMenu"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:textColor="@android:color/darker_gray"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

 app:layout_constraintTop_toBottomOf="@id/start_date_button
">

        <AutoCompleteTextView
            android:id="@+id/search_cat"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/category_hint"
            android:inputType="none" />



 </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.button.MaterialButton
        android:id="@+id/search_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:hint="@string/search_button"
        android:text="@string/search_button"
        android:textColor="@color/black"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

触发活动

我们的搜索由首页顶端的搜索栏触发（搜索按钮实际上并没有实际用处）。

```
search.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
SearchActivity.class);
        startActivity(intent);
    }
});

// 搞个怪
search_icon.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "前面的区域，以后再来
探索吧！", Toast.LENGTH_SHORT).show();
    }
});
```

搜索活动

搜索活动实际上就是简单的处理信息输入和传递的过程。一个比较关键的点是制造一个日历组件，便于用户选择日期范围。

```java
package com.java.liyao;

import android.annotation.SuppressLint;
import android.app.DatePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.button.MaterialButton;
import
com.google.android.material.textfield.TextInputEditText;

import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;

public class SearchActivity extends AppCompatActivity {
    private androidx.appcompat.widget.Toolbar
search_toolbar;
```

```java
    private TextInputEditText search_keyword;
    private MaterialButton start_date_button;
    private MaterialButton end_date_button;
    private AutoCompleteTextView search_cat;
    private MaterialButton search_button;

    private static final List<String> allCats =
Arrays.asList(new String[]{"全部", "娱乐", "军事", "教育", "文
化", "健康", "财经", "体育", "汽车", "科技", "社会"});

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search);

        // 初始化控件
        search_toolbar = findViewById(R.id.search_toolbar);
        setSupportActionBar(search_toolbar);
        search_keyword = findViewById(R.id.search_keyword);
        start_date_button =
findViewById(R.id.start_date_button);
        end_date_button =
findViewById(R.id.end_date_button);
        search_cat = findViewById(R.id.search_cat);
        search_button = findViewById(R.id.search_button);

        // 设置 Toolbar
        search_toolbar.setNavigationOnClickListener(v ->
finish());

        // 初始化日期按钮文本
        updateDateButtonText();
        // 关键词
        search_keyword.setOnFocusChangeListener((v,
hasFocus) → {
            if (hasFocus) {
                search_keyword.setHint("");
            } else {
                search_keyword.setHint("请输入关键词");
            }
        });

        // 设置日期按钮
        start_date_button.setOnClickListener(v →
showDatePickerDialog(true));
        end_date_button.setOnClickListener(v →
showDatePickerDialog(false));

        // 设置搜索类别
        ArrayAdapter<String> adapter = new ArrayAdapter<>
(this, android.R.layout.simple_dropdown_item_1line,
allCats);
        search_cat.setAdapter(adapter);

        search_button.setOnClickListener(v → {
            String keyword =
search_keyword.getText().toString().trim();
            String startDate =
start_date_button.getText().toString().replace(getString(R.
string.start_date) + ": ", "").trim();
            String endDate =
end_date_button.getText().toString().replace(getString(R.st
ring.end_date) + ": ", "").trim();
            String category =
search_cat.getText().toString().trim();
```

```
            // 执行搜索逻辑
            Toast.makeText(this, "搜索关键词: " + keyword +
", 开始日期: " + startDate + ", 结束日期: " + endDate + ", 类
别: " + category, Toast.LENGTH_LONG).show();
            // TODO
            Intent intent = new Intent(SearchActivity.this,
SearchResultActivity.class);
            intent.putExtra("KEYWORD", keyword);
            intent.putExtra("START_DATE", startDate);
            intent.putExtra("END_DATE", endDate);
            intent.putExtra("CATEGORY", category);
            startActivity(intent);
            // 先注释下，测试一下能不能传入正确格式再开活动
        });
    }

    @SuppressLint("SetTextI18n")
    private void updateDateButtonText() {

 start_date_button.setText(getString(R.string.start_date) +
": 请选择日期");

 end_date_button.setText(getString(R.string.end_date) + ":
请选择日期");
    }

    private void showDatePickerDialog(final boolean
isStart) {
        Calendar calendar = Calendar.getInstance();
        DatePickerDialog datePickerDialog = new
DatePickerDialog(this,
                (view, year, month, dayOfMonth) → {
                    Calendar selectedDate =
Calendar.getInstance();
                    selectedDate.set(year, month,
dayOfMonth);

                    SimpleDateFormat sdf = new
SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
                    String date =
sdf.format(selectedDate.getTime());

                    if (isStart) {

 start_date_button.setText(getString(R.string.start_date) +
": " + date);
                    } else {

 end_date_button.setText(getString(R.string.end_date) + ":
" + date);
                    }
                },
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH));

        // 设置按钮文本

 datePickerDialog.setButton(DatePickerDialog.BUTTON_POSITIV
E, "确定", datePickerDialog);

 datePickerDialog.setButton(DatePickerDialog.BUTTON_NEGATIV
E, "取消", datePickerDialog);

        datePickerDialog.show();
        }
    }
```

结果活动

搜索结果的展示实际上与首页分类列表相似，为了节省篇幅，这里不做展示。

AI 摘要

在 AI 摘要的实现中我遇到了很大的困难，具体来说就是一直报出空指针异常，但是顺着调用栈往上爬却看到的是令人莫名其妙的库代码。最终，在无数次的尝试后，我发现了两个关键问题。

其一，我们不能在 UI 中直接进行网络请求，需要开一个新的线程。

其二，使用 SDK 方法很难获取到正确的结果，因此我选择了文档中的 HTTP 方法。具体来说，就是构造一个 HTTP 请求头。同时还需要使用适当的替换方法去掉控制字符，不然会发生 error。

这是我最终的方法：

```java
public static String aiSummaryInvoke(String content) {
        // List<ChatMessage> messages = new ArrayList<>();
        final String finalPrompt = BASIC_PROMPT + content;
        OkHttpClient client = new OkHttpClient();

        MediaType mediaType =
MediaType.parse("application/json");

        String json_content = "{\n" +
                "    \"model\": \"glm-4\",\n" +
                "    \"messages\": [\n" +
                "        {\n" +
                "            \"role\": \"user\",\n" +
                "            \"content\": \"" + finalPrompt
+ "\"\n" +
                "        }\n" +
                "    ]\n" +
                "}";

        json_content = json_content.replace("\n",
"").replace("\t", "").replace("\r", "");
        RequestBody body = RequestBody.create(mediaType,
json_content);

        Log.d("AiSummary", "aiSummaryInvoke: " +
json_content);

        Request request = new Request.Builder()

.url("https://open.bigmodel.cn/api/paas/v4/chat/completions
")
                .post(body)
                .addHeader("Authorization", "Bearer " +
API_KEY)
                .addHeader("Content-Type",
"application/json")
                .build();

        try {
            Response response =
client.newCall(request).execute();
            String s = response.body().string();
            AiSummaryRetInfo.ChoicesDTO.MessageDTO
messageDTO = new Gson().fromJson(s,
AiSummaryRetInfo.class).getChoices().get(0).getMessage();
            s = messageDTO.getContent();
//              Log.d("AiSummary", s);
            return s;
        } catch (IOException e) {
            e.printStackTrace();
        }
        String ret = "failed to get ai summary";
        return ret;
    }
```

其中的 `finalPrompt` 为 `Please summarize the main content of the following news in concise and accurate simplified Chinese. Ignore any irrelevant words and provide the summary directly without any additional information.\n` 加上新闻内容。

最终我成功返回了 `JSON` 结果。使用 `Gson` 对其进行解析，并获取 AI 摘要文本即可：

```
// NewsDetailsActivity
ai_summary.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String uk = dataDTO.getUniqueID();
        boolean isSummarized =
AiSummaryDbHelper.getInstance(NewsDetailsActivity.this).sea
rchSummary(uk);
        if (!isSummarized) {
            ai_summary.setText("摘要：AI摘要生成中...");
            new Thread(() -> {
                String aiSummary =
AiSummary.aiSummaryInvoke(dataDTO.getContent());
                setAi_summary_DTO(aiSummary);
            }).start();
        } else {
            ai_summary.setText("摘要: " +
AiSummaryDbHelper.getInstance(NewsDetailsActivity.this).get
Summary(uk).getAiSummary());
        }
    }
});
```

而对于 AI 摘要的本地存储，我依然使用 SQLite。新建 AI 摘要实体类：

```
package com.java.liyao.entity;

public class AiSummaryInfo {
    private int summaryId;
    private String uniqueId;
    private String aiSummary;

    public AiSummaryInfo(int summaryId, String uniqueId,
String aiSummary) {
        this.summaryId = summaryId;
        this.uniqueId = uniqueId;
        this.aiSummary = aiSummary;
    }

    public int getSummaryId() {
        return summaryId;
    }
    public void setSummaryId(int summaryId) {
        this.summaryId = summaryId;
    }

    public String getUniqueId() {
        return uniqueId;
    }

    public void setUniqueId(String uniqueId) {
        this.uniqueId = uniqueId;
    }

    public String getAiSummary() {
        return aiSummary;
    }

    public void setAiSummary(String aiSummary) {
        this.aiSummary = aiSummary;
    }
}
```

然后按照新闻唯一标识符 -AI 摘要文本构造数据库助手即可。

```
// 每日数据库（1/114514）

package com.java.liyao.db;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

import com.java.liyao.entity.AiSummaryInfo;

public class AiSummaryDbHelper extends SQLiteOpenHelper {
    private static AiSummaryDbHelper sHelper;
    private static final String DB_NAME = "summary.db";
    private static final int VERSION = 1;

    public AiSummaryDbHelper(@Nullable Context context,
@Nullable String name, @Nullable
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    public synchronized static AiSummaryDbHelper
getInstance(Context context) {
        if (null == sHelper) {
            sHelper = new AiSummaryDbHelper(context,
DB_NAME, null, VERSION);
        }
        return sHelper;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //创建user_table表
        db.execSQL("create table summary_table(summary_id
integer primary key autoincrement, " +
                "unique_id TEXT," +
                "ai_summary TEXT" +
                ")");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase,
int i, int i1) {
        // 先不写
    }

    public int addSummary(String unique_id, String
ai_summary) {
        if (!searchSummary(unique_id)) {
            SQLiteDatabase db = getWritableDatabase();
            ContentValues values = new ContentValues();

            values.put("unique_id", unique_id);
            values.put("ai_summary", ai_summary);

            int insert = (int) db.insert("summary_table",
null, values);
            db.close();
            return insert;
        }
        return 0;
    }

    @SuppressLint("Range")
    public boolean searchSummary(String uk) {
        SQLiteDatabase db = getReadableDatabase();
        String sql;
        Cursor cursor;
        sql = "select summary_id, unique_id, ai_summary
from summary_table where unique_id=?";
```

```
        cursor = db.rawQuery(sql, new String[]{uk});
        boolean result = cursor.getCount() > 0;
        cursor.close();
        db.close();
        return result;
    }

    @SuppressLint("Range")
    public AiSummaryInfo getSummary(String uniqueId) {
        SQLiteDatabase db = this.getReadableDatabase();
        AiSummaryInfo summaryInfo = null;
        String sql = "SELECT summary_id, unique_id,
ai_summary FROM summary_table WHERE unique_id=?";
        Cursor cursor = db.rawQuery(sql, new String[]
{uniqueId});

        if (cursor.moveToFirst()) {
            int summaryId =
cursor.getInt(cursor.getColumnIndex("summary_id"));
            String unique_id =
cursor.getString(cursor.getColumnIndex("unique_id"));
            String aiSummary =
cursor.getString(cursor.getColumnIndex("ai_summary"));
            summaryInfo = new AiSummaryInfo(0, uniqueId,
"");
            summaryInfo.setSummaryId(summaryId);
            summaryInfo.setUniqueId(unique_id);
            summaryInfo.setAiSummary(aiSummary);
        }
        cursor.close();
        db.close();
        return summaryInfo;
    }
```

```
}
```

用户系统

用户系统可谓是我项目中最大的 bonus。我使用 SQLite，实现了登录、注册等全套的功能。

`UserInfo` 实体

简而言之，首先我们需要一个 `UserInfo` 实体，代表用户的信息，包含用户 ID（自动生成）、邮箱（全局唯一）、用户名和密码等。由于每次只能有一个用户登录，因此依然使用我们的单例模式。

```
package com.java.liyao.entity;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class UserInfo {
    private int user_id;
    private String nickname;
    private String user_email;
    private String password;

    public static UserInfo sUserInfo; // 没登录的时候就是 null

    public UserInfo(int user_id, String nickname, String
user_email, String password) {
        this.user_id = user_id;
        this.nickname = nickname;
        this.user_email = user_email;
        this.password = password;
    }

    public static UserInfo getUserinfo() {
        return sUserInfo;
    }

    public static void setUserinfo(UserInfo userinfo) {
        sUserInfo = userinfo;
    }

    public int getUser_id() {
        return user_id;
    }

    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }

    public String getNickname() {
        return nickname;
    }

    public void setNickname(String nickname) {
        this.nickname = nickname;
    }

    public String getUser_email() {
        return user_email;
    }

    public void setUser_email(String user_email) {
        this.user_email = user_email;
    }

    public String getPassword() {
        return password;
    }

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

    public void logOut() {
        sUserInfo = null;
        return;
    }
}
```

`UserDbHelper`

下一步需要设计用户数据库助手。这个工作几乎毫无难度，只是在测试时，由于前后数据库的表格式不兼容发生了错误，最终通过删除原有数据表并更新解决了问题。

登录和注册活动

登录和注册本质是就是对数据库的查询/写入。篇幅所限，我们只展示注册页面的布局和活动。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```xml
        android:id="@+id/register"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".RegisterActivity">

        <androidx.appcompat.widget.Toolbar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:navigationIcon="@drawable/comeback"
            android:id="@+id/backToLogin"
            android:layout_marginTop="10dp"
            >

        </androidx.appcompat.widget.Toolbar>

        <ImageView
            android:id="@+id/logoImageView"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="40dp"
            android:background="@drawable/circle_shape"
            android:clipToOutline="true"
            android:contentDescription="@string/app_name"
            android:scaleType="centerCrop"
            android:src="@mipmap/ic_launcher_round"
            tools:targetApi="31" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="20dp"
            android:gravity="center"
            android:text="@string/register_title"
            android:textColor="#333"
            android:textSize="20sp"
            android:textStyle="bold" />

        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="20dp"
            android:orientation="vertical">

            <androidx.appcompat.widget.LinearLayoutCompat
                android:layout_width="match_parent"
                android:layout_height="50dp"
                android:background="@drawable/login_et_bg">

                <ImageView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_vertical"
                    android:layout_marginStart="12dp"
                    android:src="@drawable/nickname">

                </ImageView>

                <EditText
                    android:layout_width="wrap_content"
                    android:layout_height="match_parent"
                    android:layout_marginStart="12dp"
                    android:autofillHints=""

     android:background="@android:color/transparent"
```

```
                android:gravity="center_vertical"

    android:hint="@string/please_input_nickname"
                android:inputType="text"
                android:textColorHint="@color/grey"
                android:textSize="15sp"
                android:id="@+id/registerEtrNickname">

            </EditText>
        </androidx.appcompat.widget.LinearLayoutCompat>


        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_marginTop="20dp"
            android:background="@drawable/login_et_bg">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginStart="12dp"

    android:contentDescription="@string/app_name"
                android:src="@drawable/email">

            </ImageView>

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_marginStart="12dp"
                android:autofillHints=""

    android:background="@android:color/transparent"
                android:gravity="center_vertical"
                android:hint="@string/please_input_email"
                android:inputType="textEmailAddress"
                android:textColorHint="@color/grey"
                android:textSize="15sp"
                android:id="@+id/registerEtrEmail">

            </EditText>
        </androidx.appcompat.widget.LinearLayoutCompat>

        <androidx.appcompat.widget.LinearLayoutCompat
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_marginTop="20dp"
            android:background="@drawable/login_et_bg">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginStart="12dp"

    android:contentDescription="@string/app_name"
                android:src="@drawable/password">

            </ImageView>

            <EditText
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
```

```
                    android:layout_marginStart="12dp"
                    android:autofillHints=""

    android:background="@android:color/transparent"
                    android:gravity="center_vertical"

    android:hint="@string/please_input_password"
                    android:inputType="textPassword"
                    android:textColorHint="@color/grey"
                    android:textSize="15sp"
                    android:id="@+id/registerEtrPassword"
                    >

                </EditText>
            </androidx.appcompat.widget.LinearLayoutCompat>


            <Button
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="30dp"
                android:background="@drawable/login_btn_bg"
                android:text="@string/lets_register"
                android:textColor="#000000"
                android:textSize="20sp"
                android:id="@+id/registerBtn">

            </Button>

        </androidx.appcompat.widget.LinearLayoutCompat>
    </androidx.appcompat.widget.LinearLayoutCompat>
```

```java
package com.java.liyao;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.java.liyao.db.CatPrefDbHelper;
import com.java.liyao.db.UserDbHelper;

import java.util.Arrays;
import java.util.List;

public class RegisterActivity extends AppCompatActivity {

    private EditText registerEtrNickname;
    private EditText registerEtrEmail;
    private EditText registerEtrPassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_register);
        registerEtrNickname =
findViewById(R.id.registerEtrNickname);
        registerEtrEmail =
findViewById(R.id.registerEtrEmail);
        registerEtrPassword =
findViewById(R.id.registerEtrPassword);


 findViewById(R.id.backToLogin).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // 血的教训：不要去跳转，不然会出现多个Activity
                finish();
            }
        });


 findViewById(R.id.registerBtn).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String nickname =
registerEtrNickname.getText().toString();
                String email =
registerEtrEmail.getText().toString();
                String password =
registerEtrPassword.getText().toString();

                if (TextUtils.isEmpty(nickname) ||
TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
                    Toast.makeText(RegisterActivity.this,
"信息缺失！", Toast.LENGTH_SHORT).show();
```

```
                    } else {
                        long ret =
UserDbHelper.getInstance(RegisterActivity.this).register(ni
ckname, email, password);
                        if (ret > 0) {
                            List<String> allCats =
Arrays.asList(new String[]{"全部", "娱乐", "军事", "教育", "文
化", "健康", "财经", "体育", "汽车", "科技", "社会"});

 CatPrefDbHelper.getInstance(RegisterActivity.this).addCatP
ref(email, allCats.toString());

 Toast.makeText(RegisterActivity.this, "注册成功！",
Toast.LENGTH_SHORT).show();
                            finish();
                        }
                    }
                }
            });
        }
}
```

账号信息

账号信息的展示主要分为两部分，侧边栏顶部的信息和个人中心。侧边栏顶部的默认样式与应用的黑白色调不是很搭配，因此进行一些样式微调。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
```

```xml
    android:layout_height="@dimen/nav_header_height"
    android:background="@android:color/transparent"
    android:gravity="bottom"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

 android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <ImageView
        android:id="@+id/avatar"
        android:layout_width="80dp"
        android:layout_height="80dp"

 android:contentDescription="@string/nav_header_desc"

 android:paddingTop="@dimen/nav_header_vertical_spacing"
        app:srcCompat="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/tv_nickname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"

 android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:text="@string/nav_header_title"

 android:textAppearance="@style/TextAppearance.AppCompat.Bo
dy1"
        android:textColor="@color/black"/>
    <TextView
        android:id="@+id/tv_user_email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:layout_marginStart="10dp"
        android:text="@string/nav_header_subtitle" />
</LinearLayout>
```

然后在主活动中添加相应代码。为了实现登录后的实时更新，我们需要在 `onResume` 中添加相关语句。

```
tv_nickname.setText(userInfo.getNickname());
tv_user_email.setText(userInfo.getUser_email());
tv_user_email.setVisibility(View.VISIBLE);
// 未登录时同理
```

个人中心

个人中心依然是侧边栏的一部分，需要绑定点击事件。绑定后写一个样式文件即可。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/account"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```xml
    android:orientation="vertical"
    tools:context=".AccountActivity">

    <Toolbar
        android:id="@+id/account_toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/transparent"
        android:navigationIcon="@drawable/comeback"
        android:title="@string/account_title">

    </Toolbar>

    <ImageView
        android:id="@+id/logoImageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="100dp"
        android:background="@drawable/circle_shape"
        android:clipToOutline="true"
        android:scaleType="centerCrop"
        android:src="@mipmap/ic_launcher_round"
        tools:targetApi="31" />

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/account_email"
        android:layout_marginStart="40dp"
        android:layout_marginTop="40dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="20dp">
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginStart="12dp"
            android:src="@drawable/email">

        </ImageView>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:text="占位"
            android:id="@+id/account_email_text"
            android:textSize="15sp">

        </TextView>

    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:id="@+id/account_password"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="40dp">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```xml
                android:layout_gravity="center_vertical"
                android:layout_marginStart="12dp"
                android:src="@drawable/password">

            </ImageView>

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="15dp"

     android:text="@string/account_password_placeholder"
                android:id="@+id/account_password_text"
                android:textSize="15sp">

            </TextView>

        </androidx.appcompat.widget.LinearLayoutCompat>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="60dp"
            android:text="@string/change_password_title"
            android:id="@+id/account_change_password"
            android:textColor="@color/black">

        </Button>

    </androidx.appcompat.widget.LinearLayoutCompat>
```

个人中心的重点就是修改密码，具体请看下一小节。

修改密码

修改密码的本质是对数据库的修改。当然，根据一般逻辑，我们需要输入原密码，才能修改新密码。

修改密码的选项被我放在个人中心中，只需要为「修改密码」按钮绑定点击事件即可。为了不创造出太多个活动页面，我选择了 `AlertDialog` 作为展示方式，这样可以创建一个美观的弹窗。

```
account_change_password.setOnClickListener(v → {
            // 跳转到修改密码的活动
            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            LayoutInflater inflater =
this.getLayoutInflater();
            View dialogView =
inflater.inflate(R.layout.dialog_change_password, null);
            builder.setView(dialogView);

            EditText changePasswordEtrOld =
dialogView.findViewById(R.id.changePasswordLlOld).findViewB
yId(R.id.changePasswordEtrOld);
            EditText changePasswordEtrNew =
dialogView.findViewById(R.id.changePasswordLlNew).findViewB
yId(R.id.changePasswordEtrNew);
            EditText changePasswordEtrNewAgain =
dialogView.findViewById(R.id.changePasswordLlNewAgain).find
ViewById(R.id.changePasswordEtrNewAgain);
            Button changePasswordBtnChange =
dialogView.findViewById(R.id.changePasswordBtnChange);

            AlertDialog dialog = builder.create();
            changePasswordBtnChange.setOnClickListener(v1 -
> {
                // 修改密码
                String oldPassword =
changePasswordEtrOld.getText().toString();
                String newPassword =
changePasswordEtrNew.getText().toString();
                String newPasswordAgain =
changePasswordEtrNewAgain.getText().toString();
                String eml = userInfo.getUser_email();
                if (Objects.equals(userInfo.getPassword(),
oldPassword)) {
                    if (Objects.equals(newPassword,
newPasswordAgain)) {
                        userInfo.setPassword(newPassword);
                        // 修改数据库

 UserDbHelper.getInstance(this).changePassword(eml,
newPassword);
                        Toast.makeText(this, "密码修改成功! 请
重新登录", Toast.LENGTH_SHORT).show();
                        dialog.dismiss();
                        // 跳转到登录活动
                        Intent intent = new Intent(this,
LoginActivity.class);
                        startActivity(intent);
                    } else {
                        // 修改失败
                        Toast.makeText(this, "两次输入的密码不
一致! ", Toast.LENGTH_SHORT).show();
                    }
                } else {
                    // 修改失败
                    Toast.makeText(this, "原密码错误！", 
Toast.LENGTH_SHORT).show();
                }
                dialog.dismiss();
            });

            dialog.show();
        });
```

记住密码

如果用户每次登录都要重新输入邮箱和密码，那无疑是很不人性化的。这里我们选择使用 `SharedPreferences` 作为轻量级的持久化数据存储方案。

在登录的布局文件中新增一个复选框：

```
<CheckBox
    android:id="@+id/rememberMe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/remember_me_text">
</CheckBox>
```

然后为其绑定一个点击事件：

```
rememberMe.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton
compoundButton, boolean b) {
        isLogin = b;
    }
});
```

然后创建 `SharedPreferences` ：

```
protected void onCreate(Bundle savedInstanceState) {
        // ...

        sharedPreferences =
getSharedPreferences("userInfo", MODE_PRIVATE);

        // ...

        boolean is_login =
sharedPreferences.getBoolean("isLogin", false);
        if (is_login) {

 loginEtrEmail.setText(sharedPreferences.getString("email",
""));

 loginEtrPassword.setText(sharedPreferences.getString("pass
word", ""));
            rememberMe.setChecked(true);
        }

        // ...

        findViewById(R.id.loginBtn).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email =
loginEtrEmail.getText().toString();
                String password =
loginEtrPassword.getText().toString();
                if (TextUtils.isEmpty(email) ||
TextUtils.isEmpty(password)) {
                    Toast.makeText(LoginActivity.this, "信息
缺失！", Toast.LENGTH_SHORT).show();
                } else {
                    UserInfo userInfo =
UserDbHelper.getInstance(LoginActivity.this).login(email);
                if (null != userInfo &&
userInfo.getPassword().equals(password) {
                    UserInfo.setUserinfo(userInfo);

                    SharedPreferences.Editor editor =
sharedPreferences.edit();
                    editor.putString("email", email);
                    editor.putString("password", password);
                    editor.putBoolean("isLogin", isLogin);
                    editor.apply();
                }

        // ...
    }
});
```

这样就可以实现记住用户名和密码的功能了。

一键登出

一键登出的按钮位于侧边栏，因此绑定点击事件，并调用准备好的方法即可。

```
else if (item.getItemId() == R.id.nav_logout) {
    userInfo.logOut();
    Toast.makeText(MainActivity.this, "已退出登录！", 
Toast.LENGTH_SHORT).show();
    onResume();
}
```

分类修改

由于我早就打算做账号系统了，因此在分类偏好方面也要分账号存储。为了不破坏原有的结构，我们构造一个用户邮箱-分类偏好数据库。这个数据库与其他数据库相比有一个关键的问题：**数据库只能存储字符串，但是分类偏好本身是列表**。所以我们需要将其转换为字符串。

```
package com.java.liyao.entity;

import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

public class CatPrefInfo {
    private String userEmail;
    private List<String> catPref;
    private int catPrefId;
```

```java
    public int getCatPrefId() {
        return catPrefId;
    }

    public void setCatPrefId(int catPrefId) {
        this.catPrefId = catPrefId;
    }

    public CatPrefInfo(int catPrefId, String userEmail,
List<String> catPref) {
        this.userEmail = userEmail;
        this.catPref = catPref;
        this.catPrefId = catPrefId;
    }

    public CatPrefInfo(String userEmail, List<String>
catPref) {
        this.userEmail = userEmail;
        this.catPref = catPref;
    }

    public String getUserEmail() {
        return userEmail;
    }

    public void setUserEmail(String userEmail) {
        this.userEmail = userEmail;
    }

    public List<String> getCatPref() {
        return catPref;
    }
    public void setCatPref(List<String> catPref) {
        this.catPref = catPref;
    }

    public static List<String> stringToList(String
listString) {
        List<String> list = new ArrayList<>();
        if (listString != null && !listString.isEmpty()) {
            listString = listString.substring(1,
listString.length() - 1); // 移除首尾的中括号
            StringTokenizer tokenizer = new
StringTokenizer(listString, ", "); // 使用逗号和空格作为分隔符
            while (tokenizer.hasMoreTokens()) {
                list.add(tokenizer.nextToken());
            }
        }
        return list;
    }
}
```

```
package com.java.liyao.db;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;
import com.java.liyao.entity.CatPrefInfo;

import java.util.ArrayList;
import java.util.List;

public class CatPrefDbHelper extends SQLiteOpenHelper {
    private static CatPrefDbHelper sHelper;
    private static final String DB_NAME = "catpref.db";
    private static final int VERSION = 1;

    public CatPrefDbHelper(@Nullable Context context,
@Nullable String name, @Nullable
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    public synchronized static CatPrefDbHelper
getInstance(Context context) {
        if (null == sHelper) {
            sHelper = new CatPrefDbHelper(context, DB_NAME,
null, VERSION);
        }
        return sHelper;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //创建user_table表
        db.execSQL("create table catpref_table(catpref_id
integer primary key autoincrement, " +
                "user_email text," +       //  按照我的规定，我
们使用用户邮箱作为用户的唯一标识
                "cat_pref text" +
                ")");
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase,
int i, int i1) {
        // 先不写
    }

    public int addCatPref(String user_email, String
cat_pref) {
        if (!searchCatPref(user_email)) {
            SQLiteDatabase db = getWritableDatabase();
            ContentValues values = new ContentValues();

            values.put("user_email", user_email);
            values.put("cat_pref", cat_pref);

            int insert = (int) db.insert("catpref_table",
null, values);
            db.close();
            return insert;
        }
        return 0;
    }

    @SuppressLint("Range")
    public List<CatPrefInfo> getCatPref(String ue) {
        SQLiteDatabase db = getReadableDatabase();
        List<CatPrefInfo> list = new ArrayList<>();
        String sql;
        Cursor cursor;
        if (ue == null) {
            sql = "select catpref_id,user_email,cat_pref
from catpref_table";
            cursor = db.rawQuery(sql, null);
        } else {
            sql = "select catpref_id,user_email,cat_pref
from catpref_table where user_email=?";
            cursor = db.rawQuery(sql, new String[]{ue});
        }
        while (cursor.moveToNext()) {
            int catpref_id =
cursor.getInt(cursor.getColumnIndex("catpref_id")); //
Corrected column name
            String user_email =
cursor.getString(cursor.getColumnIndex("user_email"));
            String cat_pref =
cursor.getString(cursor.getColumnIndex("cat_pref"));
            List<String> cat_prefList =
CatPrefInfo.stringToList(cat_pref);
            list.add(new CatPrefInfo(catpref_id,
user_email, cat_prefList));
        }
        cursor.close();
        db.close();
        return list;
    }

    @SuppressLint("Range")
    public boolean searchCatPref(String ue) {
        SQLiteDatabase db = getReadableDatabase();
        String sql;
        Cursor cursor;
        if (ue == null) {
            // If user_email is null, modify the query to
not include user_email in the WHERE clause
            sql = "select catpref_id, user_email, cat_pref
from catpref_table";
            cursor = db.rawQuery(sql, null);
        } else {
            // If user_email is not null, include it in the
WHERE clause
            sql = "select catpref_id, user_email, cat_pref
from catpref_table where user_email=?";
            cursor = db.rawQuery(sql, new String[]{ue});
        }
        boolean result = cursor.getCount() > 0;
        cursor.close();
        db.close();
        return result;
    }

    // 还需要一个修改列表的函数，这里我们假定邮箱都是已经存在的
    public int updateCatPref(String user_email,
List<String> cat_pref) {
        SQLiteDatabase db = getWritableDatabase();
        ContentValues values = new ContentValues();
        String cat_pref_string = cat_pref.toString();
        values.put("cat_pref", cat_pref_string);
        boolean isUnlogged = user_email == null;
        String whereClause = isUnlogged ? null :
"user_email=?";
        String[] whereArgs = isUnlogged ? null : new
String[]{user_email};
        int update = db.update("catpref_table", values,
whereClause, whereArgs);
        db.close();
```

```
            return update;
        }

        // 再写一个获取分类偏好的方法
        public List<String> getCatPrefList(String user_email) {
            List<CatPrefInfo> catPref = getCatPref(user_email);
            if (catPref.isEmpty()) {
                return null;
            }
            return catPref.get(0).getCatPref();
        }

        // 这方法是临时的，就是给我调试用的，后面会删掉
        public int resetAllUserPreferences(List<String>
    defaultCatPrefs) {
            SQLiteDatabase db = getWritableDatabase();
            ContentValues values = new ContentValues();
            String defaultCatPrefString =
    defaultCatPrefs.toString();
            values.put("cat_pref", defaultCatPrefString);

            // 更新所有用户的偏好列表为默认列表
            int updateCount = db.update("catpref_table",
    values, null, null);
            db.close();
            return updateCount;
        }
    }
```

如上文所言，在每次注册新用户时，我们都为其创建一个默认分类列表（全部）。

对于列表的修改，我们依然将其放在侧边栏中并绑定点击事件。对于分类的展示，我们选择 `GridLayout` 和修改后的 `Widget. MaterialComponents. Button. OutlinedButton` 。

```
<!—— themes.xml ——>
<style name="CategoryButton"
parent="Widget.MaterialComponents.Button.OutlinedButton">
    <item name="android:layout_width">0dp</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_columnWeight">1</item>
    <item name="android:layout_margin">10dp</item>
    <item name="android:textColor">@color/black</item>
    <item
name="strokeColor">@android:color/darker_gray</item>
    <item name="strokeWidth">1.5dp</item>
</style>

<Button
    android:id="@+id/btn_category1"
    style="@style/CategoryButton"
    android:text="全部" />
```

而修改的效果，我们通过字体颜色的变化指定。

在列表修改的具体活动中，我们首先需要获取当前用户的偏好列表。未登录用户也有自己的列表，邮箱为 “null” （注意不是 `null` ）。

```
UserInfo userInfo = UserInfo.getUserinfo();
        String eml = userInfo != null ?
userInfo.getUser_email() : "null";
```

```
        List<String> tmpCatList =
CatPrefDbHelper.getInstance(this).getCatPrefList(eml);

        // Log.d("UserCatPreference", "onCreate: " +
tmpCatList.toString());

        for (Button btn : category_btns) {
            String catName = btn.getText().toString();
            if (tmpCatList.contains(catName)) {

 btn.setTextColor(getResources().getColor(R.color.black));
            } else {

 btn.setTextColor(getResources().getColor(R.color.grey));
            }
        }
```

同时，我们需要为每一个按钮都绑定一个点击事件。在选中状态下，点击会删除；反之会被加入。为了保证列表顺序的一致性，我们定义两个工具方法：

```
public static void addAndSort(List<String> tmpCatList,
String newCategory) {
    if (!tmpCatList.contains(newCategory)) {
        tmpCatList.add(newCategory);
        // 根据correctOrderList的顺序对tmpCatList进行排序
        Collections.sort(tmpCatList, new Comparator<String>
() {
            @Override
            public int compare(String o1, String o2) {
                int index1 = allCats.indexOf(o1);
                int index2 = allCats.indexOf(o2);
                return Integer.compare(index1, index2);
            }
        });
    }
}

public static void removeAndSort(List<String> tmpCatList,
String newCategory) {
    if (tmpCatList.contains(newCategory)) {
        tmpCatList.remove(newCategory);
        // 根据correctOrderList的顺序对tmpCatList进行排序
        Collections.sort(tmpCatList, new Comparator<String>
() {
            @Override
            public int compare(String o1, String o2) {
                int index1 = allCats.indexOf(o1);
                int index2 = allCats.indexOf(o2);
                return Integer.compare(index1, index2);
            }
        });
    }
}
```

下面是具体的实现：

```
for (Button btn : category_btns) {
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String catName = btn.getText().toString();
            boolean isBlack = btn.getCurrentTextColor() ==
getResources().getColor(R.color.black);
```

```
            // boolean isLoggedIn = userInfo ≠ null;

            if (isBlack) {

 btn.setTextColor(getResources().getColor(R.color.grey));
                removeAndSort(tmpCatList, catName);

 CatPrefDbHelper.getInstance(EditCategoriesActivity.this).u
pdateCatPref(eml, tmpCatList);

            } else {

 btn.setTextColor(getResources().getColor(R.color.black));
                addAndSort(tmpCatList, catName);

 CatPrefDbHelper.getInstance(EditCategoriesActivity.this).u
pdateCatPref(eml, tmpCatList);
            }
        }
    });
}
```

为了保证用户回到首屏后分类立即刷新，我们需要在 `MainActivity` 的 `onResume` 方法中加入读取机制：

```
cats =
CatPrefDbHelper.getInstance(this).getCatPrefList(userInfo.g
etUser_email());
```

关于页面

关于页面依然是侧边栏中的一个实体，只需要如上面一样绑定点击事件即可。我的 `about. md` 是使用 Markdown 书写的，因此我需要一个 Markdown 解析库。这里我选择 `Markwon` 。

```java
package com.java.liyao;

import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toolbar;

import androidx.appcompat.app.AppCompatActivity;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import io.noties.markwon.Markwon;
// 导入删除线的扩展
import
io.noties.markwon.ext.strikethrough.StrikethroughPlugin;

public class AboutActivity extends AppCompatActivity {
    private TextView about_text;
    private Toolbar about_toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // EdgeToEdge.enable(this);
        setContentView(R.layout.activity_about);
        about_text = findViewById(R.id.about_text);
        about_toolbar = findViewById(R.id.about_toolbar);

        about_toolbar.setNavigationOnClickListener(v →
finish());

        Markwon markwon = Markwon.builder(this)
                .usePlugin(new StrikethroughPlugin()) // 使
用扩展
                .build();

        // 从assets目录下
        String markdown = readMarkdownFromAssets();
        markwon.setMarkdown(about_text, markdown);
    }

    private String readMarkdownFromAssets() {
        StringBuilder content = new StringBuilder();
        try {
            InputStream is = getAssets().open("about.md");
            BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
            String line;
            while ((line = reader.readLine()) ≠ null) {
                content.append(line).append("\n");
            }
            reader.close();
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return content.toString();
    }
}
```

然后就可以得到一个比较美观的关于页面。

后期优化

后期的优化主要包括这几方面:

- 调整详情页样式，将元数据和 AI 摘要制成美观的卡片（我是卡片化布局的爱好者）。
- 规范了一些命名（当然还有很多不规范的地方）。

结语

这次大作业确实是前所未有的体验，毕竟这是我第一次开发具有 GUI 的应用程序。

平心而论，我对 Java 的掌握并不出色，线程异步网络请求这些东西只是知其然而不知其所以然，面对 Android Studio 庞杂的文件结构也只能不停地翻 CheatSheet。做作业的过程无疑也是艰辛的：作业的前几天，我几乎都在抓狂中度过，计划纸扔了一张又一张（我保留着用纸写计划的习惯）。面对无数接踵而至的难题，身为 I 人的我只能自我寻找解决方法。从看着什么都没有的用户界面直龇牙，到开始细致地考虑美工和设计；从什么都需要翻 CheatSheet、查教程、问 AI，到能为别人解惑答疑；从「做出来就行」到「做个还行的就行」到「必须得做个好的」，感觉自己已经可以算一个 Java Android 开发者了（当然，实际上还差得远）。

如果要说对这门课程有什么知识之外的感悟，应该就是重温了我在写前端时的两条：不完全懂不代表不能写，开发者都不是全都弄懂了才上手的（甚至是只弄懂了很少一部分就可以上手了）；以及 MVP（最小可用产品）原则。

> We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.

完成比完美更重要。先把能用的搓出来，再去慢慢改良。要是要修改到完美才登台亮相，那演出早已经结束了。

感谢许老师精彩细致的课程讲解，感谢助教学长学姐的悉心指导，感谢每一位 Java 程序设计训练的同学。我们下一个起点再会。

---

1. Tsira（青若）是一个固定词头。↩