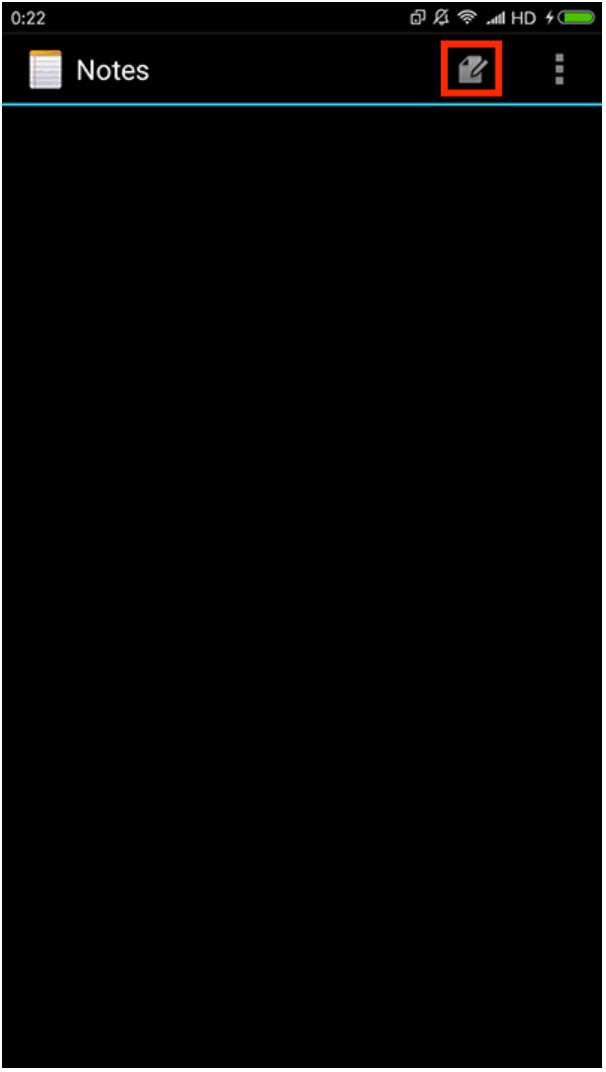
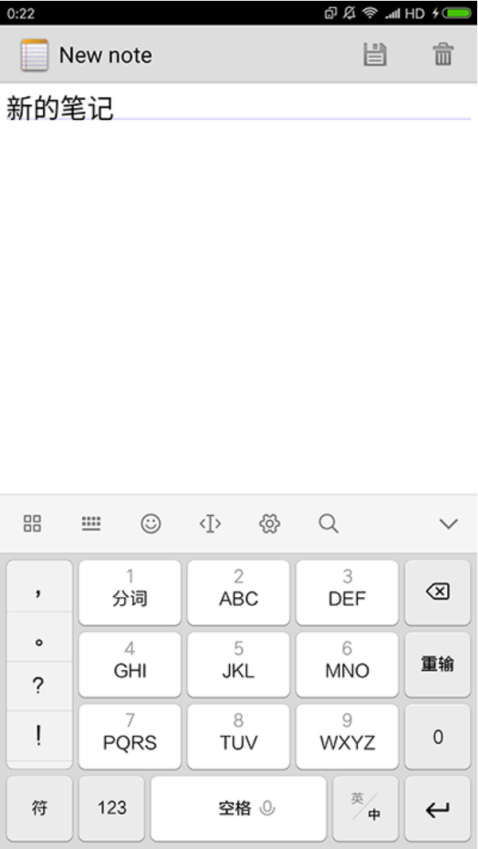
**NotePad-Android应用的介绍文档**

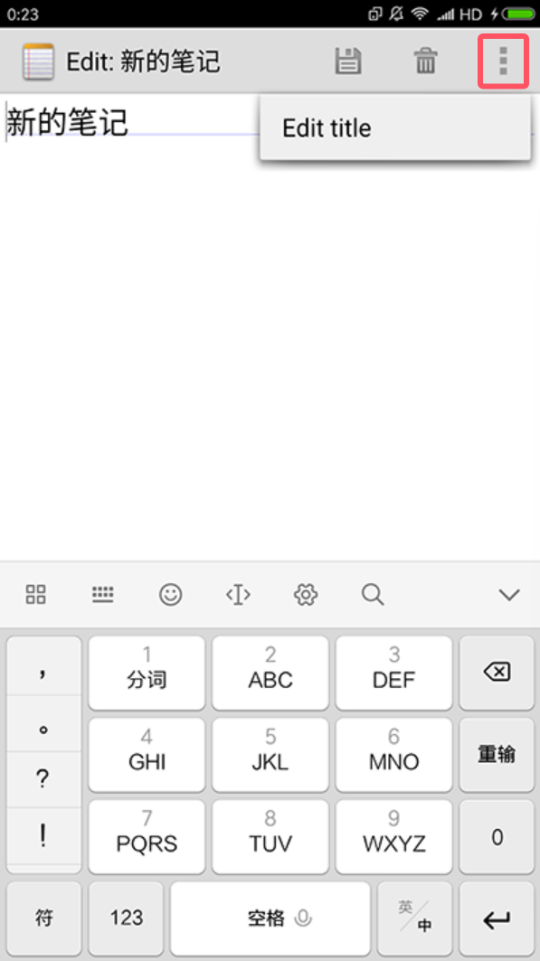
1. **初始应用的功能**
2. 新建笔记和编辑笔记
3. 在主界面点击红色矩形所示按钮，新建笔记并进入编辑界面



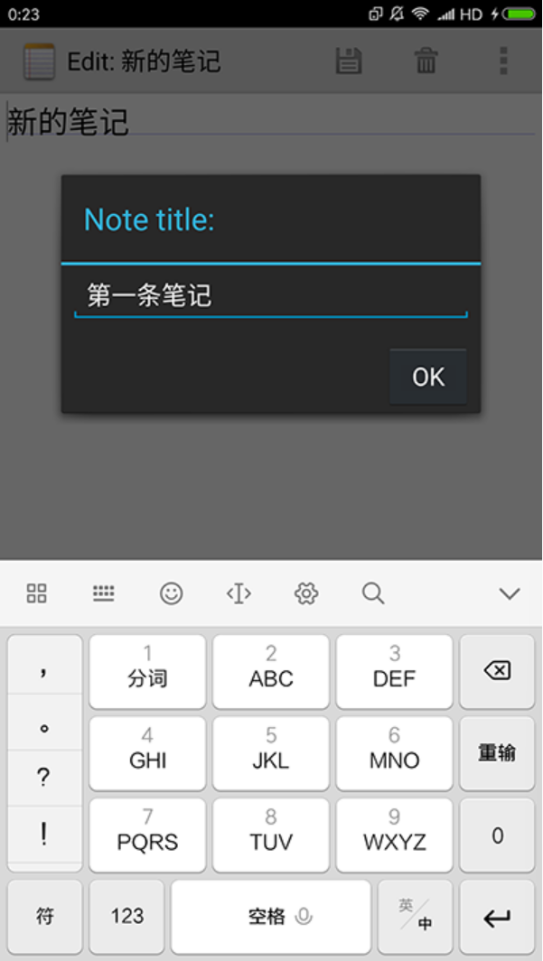
1. 进入笔记编辑界面后，可进行笔记编辑



1. 编辑标题
2. 在笔记编辑界面中点击菜单，显示的菜单条目中有“Edit title”



1. 点击“Edit title”，可编辑笔记标题



1. 笔记列表

在进行笔记的新建和编辑后，在主界面中呈现笔记列表。在初始应用中背景颜色为黑色，字体颜色为白色，且笔记列表中的每个条目都只显示笔记标题



1. **拓展基本功能**
2. 笔记条目增加时间戳显示
3. 功能要求：每个新建笔记都会保存新建时间并显示；在修改笔记后更新为修改时间
4. 实现思路和技术实现

初始应用的笔记列表item只有一个标题，需要再添加一个TextView用来显示时间，布局使用LinearLayout

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@+id/layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
 *<!--原标题TextView-->* <TextView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:id="@android:id/text1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?android:attr/listPreferredItemHeight"  
 android:textAppearance="?android:attr/textAppearanceLarge"  
 android:gravity="center\_vertical"  
 android:paddingLeft="5dip"  
 android:singleLine="true"  
 />  
 *<!--添加显示时间的TextView-->* <TextView  
 android:id="@+id/text1\_time"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textAppearance="?android:attr/textAppearanceSmall"  
 android:paddingLeft="5dip"/>  
</LinearLayout>

PROJECTION变量用来定义Java文件定义的Activity的数据，所以在PROJECTION中加入修改时间作为显示在笔记列表的时间

private static final String[] *PROJECTION* = new String[] {  
 NotePad.Notes.*\_ID*, *// 0* NotePad.Notes.*COLUMN\_NAME\_TITLE*, *// 1* NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*  
};

笔记条目数据通过SimpleCursorAdapter装填，其中用到的dataColumns，viewIDs变量需要添加修改时间

*// The names of the cursor columns to display in the view, initialized to the title column*String[] dataColumns = { NotePad.Notes.*COLUMN\_NAME\_TITLE* , NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE* } ;  
  
*// The view IDs that will display the cursor columns, initialized to the TextView in  
// noteslist\_item.xml*int[] viewIDs = { android.R.id.*text1* , R.id.*text1\_time* };

对显示的时间戳（now变量）进行转换，变成直观的时间格式

在NotePadProvider中的insert方法和NoteEditor中的updateNote方法中对now变量进行转换，前者为创建笔记时产生的时间，后者为修改笔记时产生的时间。转换后用ContentValues的put方法将转换后的dateTime存入数据库

NotePadProvider中的insert方法

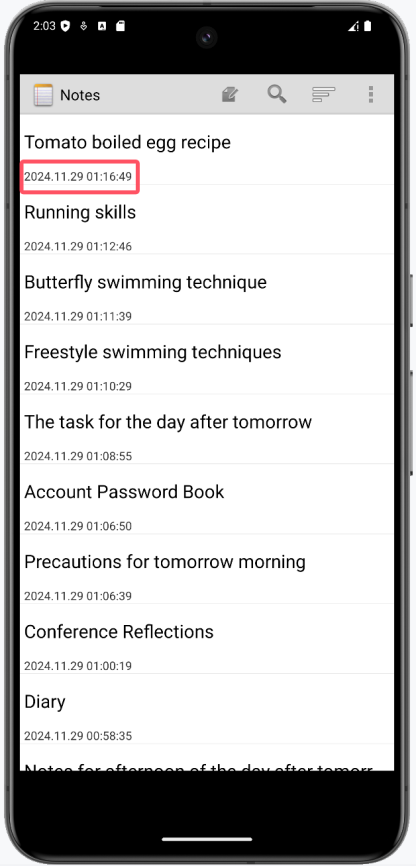
*// Gets the current system time in milliseconds*Long now = Long.*valueOf*(System.*currentTimeMillis*());  
Date date = new Date(now);  
SimpleDateFormat format = new SimpleDateFormat("yyyy.MM.dd HH:mm:ss");  
String dateTime = format.format(date);  
  
*// If the values map doesn't contain the creation date, sets the value to the current time.*if (values.containsKey(NotePad.Notes.*COLUMN\_NAME\_CREATE\_DATE*) == false) {  
 values.put(NotePad.Notes.*COLUMN\_NAME\_CREATE\_DATE*, dateTime);  
}  
  
*// If the values map doesn't contain the modification date, sets the value to the current  
// time.*if (values.containsKey(NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*) == false) {  
 values.put(NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*, dateTime);  
}

NoteEditor中的updateNote方法

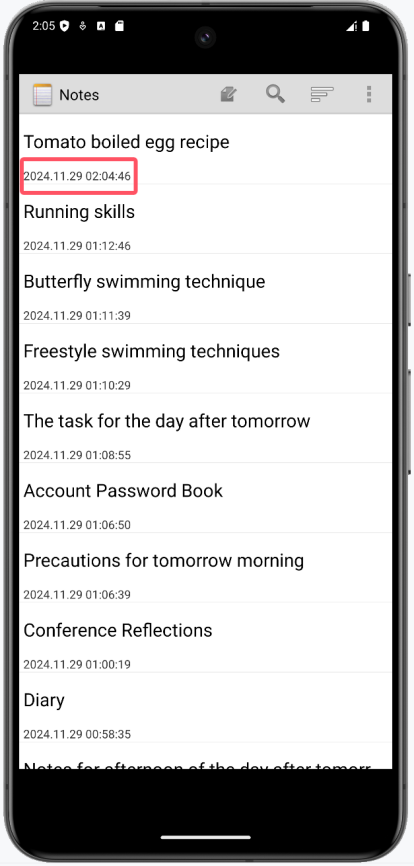
Long now = Long.*valueOf*(System.*currentTimeMillis*());  
Date date = new Date(now);  
SimpleDateFormat format = new SimpleDateFormat("yyyy.MM.dd HH:mm:ss");  
String dateTime = format.format(date);  
values.put(NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*, dateTime);

1. 实现效果界面截图

创建笔记时显示创建时间



修改笔记后显示的时间更新为最新修改的时间



1. 笔记查询功能（按标题查询）
2. 功能要求：点击搜索按钮，进行搜索界面。初始状态的搜索界面不显示笔记条目。在输入搜索内容或回删一部分搜索内容后，系统根据输入内容和笔记的标题进行字符串匹配，刷新符合要求的笔记显示在笔记列表上，后续如果回删搜索内容至为空后，显示所有的笔记
3. 实现思路和技术实现

在应用主界面添加一个搜索按钮。在list\_options\_menu.xml中添加一个搜索的item，使用Android自带的搜索图标，显示状态为总是显示

<item  
 android:id="@+id/menu\_search"  
 android:title="@string/menu\_search"  
 android:icon="@android:drawable/ic\_search\_category\_default"  
 android:showAsAction="always" />

在安卓中有个用于搜索控件：SearchView，可以把SearchView跟ListView相结合，动态地显示搜索结果。首先实现搜索页面，在layout文件夹中新建布局文件note\_search\_list.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <SearchView  
 android:id="@+id/search\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:iconifiedByDefault="false"  
 android:queryHint="输入搜索内容..."  
 android:layout\_alignParentTop="true">  
 </SearchView>  
 <ListView  
 android:id="@android:id/list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 </ListView>  
</LinearLayout>

新建一个NoteSearch类，用于实现搜索的Activity。由于搜索出来的也是笔记列表，所以可以模仿NoteList的activity继承ListActivity。

要动态地显示搜索结果，就要对SearchView文本变化设置监听，NoteSearch除了要继承ListView外还要实现SearchView.OnQueryTextListener接口

public class NoteSearch extends ListActivity implements SearchView.OnQueryTextListener {  
 private static final String[] *PROJECTION* = new String[] {  
 NotePad.Notes.*\_ID*, *// 0* NotePad.Notes.*COLUMN\_NAME\_TITLE*, *// 1  
 //扩展 显示时间 颜色* NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*, *// 2* };  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*note\_search\_list*);  
 Intent intent = getIntent();  
 if (intent.getData() == null) {  
 intent.setData(NotePad.Notes.*CONTENT\_URI*);  
 }  
 SearchView searchview = (SearchView)findViewById(R.id.*search\_view*);  
 *//为查询文本框注册监听器* searchview.setOnQueryTextListener(NoteSearch.this);  
 }  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
 @Override  
 public boolean onQueryTextChange(String newText) {  
 String selection = NotePad.Notes.*COLUMN\_NAME\_TITLE* + " Like ? ";  
 String[] selectionArgs = { "%"+newText+"%" };  
 Cursor cursor = managedQuery(  
 getIntent().getData(), *// Use the default content URI for the provider.  
 PROJECTION*, *// Return the note ID and title for each note. and modifcation date* selection, *// 条件左边* selectionArgs, *// 条件右边* NotePad.Notes.*DEFAULT\_SORT\_ORDER // Use the default sort order.* );  
 String[] dataColumns = { NotePad.Notes.*COLUMN\_NAME\_TITLE* , NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*};  
 int[] viewIDs = { android.R.id.*text1* , R.id.*text1\_time* };  
 SimpleCursorAdapter adapter = new SimpleCursorAdapter(  
 this,  
 R.layout.*noteslist\_item*,  
 cursor,  
 dataColumns,  
 viewIDs  
 );  
 setListAdapter(adapter);  
 return true;  
 }  
 @Override  
 protected void onListItemClick(ListView l, View v, int position, long id) {  
 *// Constructs a new URI from the incoming URI and the row ID* Uri uri = ContentUris.*withAppendedId*(getIntent().getData(), id);  
 *// Gets the action from the incoming Intent* String action = getIntent().getAction();  
 *// Handles requests for note data* if (Intent.*ACTION\_PICK*.equals(action) || Intent.*ACTION\_GET\_CONTENT*.equals(action)) {  
 *// Sets the result to return to the component that called this Activity. The  
 // result contains the new URI* setResult(*RESULT\_OK*, new Intent().setData(uri));  
 } else {  
 *// Sends out an Intent to start an Activity that can handle ACTION\_EDIT. The  
 // Intent's data is the note ID URI. The effect is to call NoteEdit.* startActivity(new Intent(Intent.*ACTION\_EDIT*, uri));  
 }  
 }  
}

注意：

A.onListItemClick方法是点击NoteList的item跳转到对应笔记编辑界面的方法，NoteList中有这个方法，搜索出来的笔记跳转原理与NoteList中笔记一样，所以可以直接从NoteList中复制过来直接使用。

B.使用PROJECTION，Cursor，adapter方法与时间显示的原理一致

C.动态搜索的实现最主要的部分在onQueryTextChange方法中，在使用这个方法，要先为SearchView注册监听

SearchView searchview = (SearchView)findViewById(R.id.*search\_view*);  
*//为查询文本框注册监听器*searchview.setOnQueryTextListener(NoteSearch.this);

D.onQueryTextChange方法作用是，当SearchView中文本发生变化时，执行其中代码，搜索还有一个重要的部分就是要做到模糊匹配而不是严格匹配，可以使用数据库查询语句中的LIKE和%结合来实现，newText为输入搜索的内容

String[] selectionArgs = { "%"+newText+"%" };

在AndroidManifest.xml注册NoteSearch

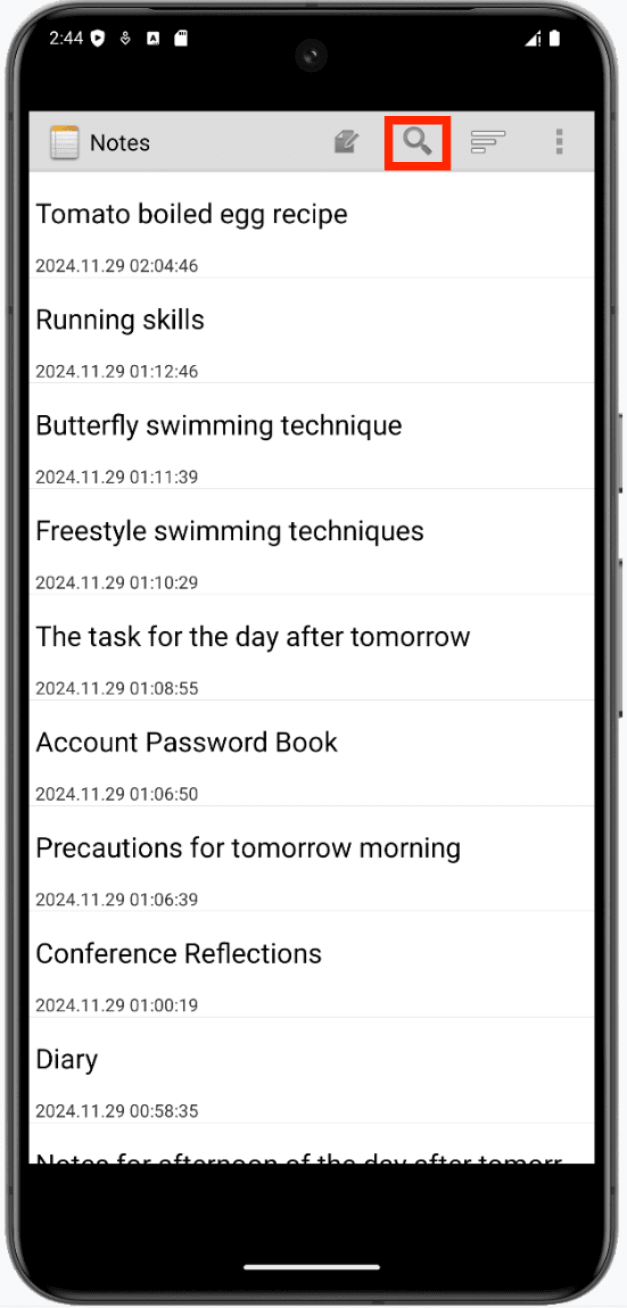
<activity  
 android:name="NoteSearch"  
 android:label="@string/title\_notes\_search"  
 android:theme="@android:style/Theme.Holo.Light">  
</activity>

在NotesList.java中的onOptionsItemSelected方法有一段switch-case语句，实现了点击应用主界面的右上方按钮后的点击交互事件。在这里添加搜索的case语句

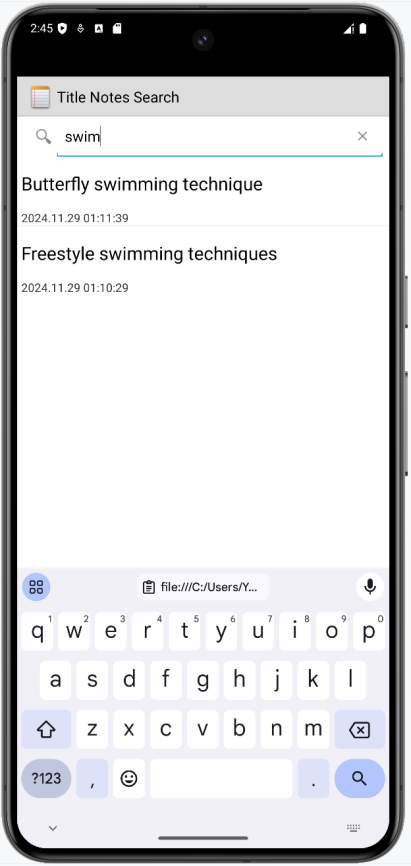
case R.id.*menu\_search*:  
 Intent intent = new Intent();  
 intent.setClass(NotesList.this,NoteSearch.class);  
 NotesList.this.startActivity(intent);  
 return true;

1. 实现效果界面截图

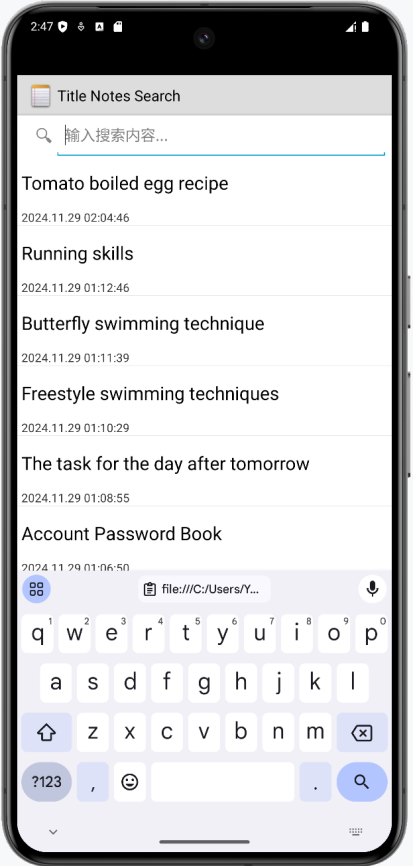
点击搜索按钮进行搜索界面



输入搜索内容，显示符合条件的笔记



3回删搜素内容至空时，显示所有的笔记



1. **拓展附加功能**
2. UI美化
3. 功能要求：在笔记编辑时可以对其切换编辑界面的背景颜色，同时更换笔记列表中的该笔记颜色（和编辑界面背景颜色相同）
4. 实现思路和技术实现

给NotesList换个主题，把暗黑风换成白亮风。在AndroidManifest.xml中NotesList的Activity中修改

<activity android:name="NotesList" android:label="@string/title\_notes\_list" android:theme="@android:style/Theme.Holo.Light">

为了实现笔记的背景颜色能够永久保存，在数据库中添加一个颜色的字段。在NotePad类中的Notes类中声明*COLUMN\_NAME\_BACK\_COLOR*变量，并使用int数据类型定义好五种不同的颜色变量

public static final String *COLUMN\_NAME\_BACK\_COLOR* = "color";  
public static final int *DEFAULT\_COLOR* = 0; *//白色*public static final int *YELLOW\_COLOR* = 1; *//黄色*public static final int *BLUE\_COLOR* = 2; *//蓝色*public static final int *GREEN\_COLOR* = 3; *//绿色*public static final int *RED\_COLOR* = 4; *//红色*

在创建数据库表处（DatabaseHelper类中的onCreate方法）添加颜色字段

public void onCreate(SQLiteDatabase db) {  
 db.execSQL("CREATE TABLE " + NotePad.Notes.*TABLE\_NAME* + " ("  
 + NotePad.Notes.*\_ID* + " INTEGER PRIMARY KEY,"  
 + NotePad.Notes.*COLUMN\_NAME\_TITLE* + " TEXT,"  
 + NotePad.Notes.*COLUMN\_NAME\_NOTE* + " TEXT,"  
 + NotePad.Notes.*COLUMN\_NAME\_CREATE\_DATE* + " INTEGER,"  
 + NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE* + " INTEGER,"  
 + NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR* + " INTEGER"  
 + ");");  
}

在NotePadProvider类中对新增的颜色字段做相应处理，

在static{}中添加

*sNotesProjectionMap*.put(  
 NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*,  
 NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*);

在insert方法中添加

if (values.containsKey(NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*) == false) {  
 values.put(NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*, NotePad.Notes.*DEFAULT\_COLOR*);  
}

实现新建笔记时，背景颜色默认为白色

原有的SimpleCursorAdapter无法满足填充背景颜色的功能，所以自定义一个CursorAdapter类继承SimpleCursorAdapter，既能完成cursor读取的数据库内容填充到item，又能将颜色填充。将其命名为MyCursorAdapter。

将颜色填充到ListView，可以使用SimpleCursorAdapter中的bindView方法来实现

public class MyCursorAdapter extends SimpleCursorAdapter {  
 public MyCursorAdapter(Context context, int layout, Cursor c, String[] from, int[] to) {  
 super(context, layout, c, from, to);  
 }  
  
 @Override  
 public void bindView(View view, Context context, Cursor cursor) {  
 super.bindView(view, context, cursor);  
  
 int x = cursor.getInt(cursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*));  
  
 switch (x) {  
 case NotePad.Notes.*DEFAULT\_COLOR*:  
 view.setBackgroundColor(Color.*rgb*(255, 255, 255));  
 break;  
 case NotePad.Notes.*YELLOW\_COLOR*:  
 view.setBackgroundColor(Color.*rgb*(247, 216, 133));  
 break;  
 case NotePad.Notes.*BLUE\_COLOR*:  
 view.setBackgroundColor(Color.*rgb*(165, 202, 237));  
 break;  
 case NotePad.Notes.*GREEN\_COLOR*:  
 view.setBackgroundColor(Color.*rgb*(161, 214, 174));  
 break;  
 case NotePad.Notes.*RED\_COLOR*:  
 view.setBackgroundColor(Color.*rgb*(244, 149, 133));  
 break;  
 default:  
 view.setBackgroundColor(Color.*rgb*(255, 255, 255));  
 break;  
 }  
 }  
}

将NotesList类中的PROJECTION添加颜色变量

private static final String[] *PROJECTION* = new String[] {  
 NotePad.Notes.*\_ID*, *// 0* NotePad.Notes.*COLUMN\_NAME\_TITLE*, *// 1* NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*,  
 NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*,  
};

并把该类中使用的SimpleCursorAdapter改成MyCursorAdapter

MyCursorAdapter adapter = new MyCursorAdapter(  
 this, *// The Context for the ListView* R.layout.*noteslist\_item*, *// Points to the XML for a list item* cursor, *// The cursor to get items from* dataColumns,  
 viewIDs  
);

此时运行会发现系统报错：2024-11-28 10:21:11.138 23752-23752 SQLiteLog com.example.android.notepad E (1) no such column: color in "SELECT \_id, title, modified, color FROM notes ORDER BY modified DESC"

使用数据库版本升级的办法，将NotePadProvider类中的*DATABASE\_VERSION变量由2改成3*

private static final int *DATABASE\_VERSION* = 3;

编辑笔记的Activity在NoteEditor类中实现。为了实现编辑笔记时的背景色更换，在NoteEditor类中的PROJECTION变量中添加颜色

private static final String[] *PROJECTION* =  
 new String[] {  
 NotePad.Notes.*\_ID*,  
 NotePad.Notes.*COLUMN\_NAME\_TITLE*,  
 NotePad.Notes.*COLUMN\_NAME\_NOTE*,  
 NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*};

在NoteEditor类中有onResume()方法，onResume()方法在正常启动时会被调用，一般是onStart()后会执行onResume()，在Acitivity从Pause状态转化到Active状态也会被调用。利用这个特点，将从数据库读取颜色并设置编辑界面背景色的操作放入其中，这样除了从笔记列表点进来时可以被执行到，跳到改变颜色的Activity（接下来会提到），改变后也会被执行到

在NoteEditor类中的onResume()方法添加读取颜色数据的操作

int x = mCursor.getInt(mCursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*));switch (x){  
 case NotePad.Notes.*DEFAULT\_COLOR*:  
 mText.setBackgroundColor(Color.*rgb*(255, 255, 255));  
 break;  
 case NotePad.Notes.*YELLOW\_COLOR*:  
 mText.setBackgroundColor(Color.*rgb*(247, 216, 133));  
 break;  
 case NotePad.Notes.*BLUE\_COLOR*:  
 mText.setBackgroundColor(Color.*rgb*(165, 202, 237));  
 break;  
 case NotePad.Notes.*GREEN\_COLOR*:  
 mText.setBackgroundColor(Color.*rgb*(161, 214, 174));  
 break;  
 case NotePad.Notes.*RED\_COLOR*:  
 mText.setBackgroundColor(Color.*rgb*(244, 149, 133));  
 break;  
 default:  
 mText.setBackgroundColor(Color.*rgb*(255, 255, 255));  
 break;  
 }

在编辑界面的菜单xml文件（editor\_options\_menu.xml）中添加更改背景颜色的item，图标为Android自带，显示状态为总是显示

<item android:id="@+id/menu\_color"  
 android:title="@string/menu\_color"  
 android:icon="@android:drawable/ic\_menu\_preferences"  
 android:showAsAction="always"/>

点击更改背景颜色的item图标，要显示一个对话框界面用于让用户选择颜色。所以新建布局note\_color.xml，垂直线性布局放置5个ImageButton

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <ImageButton  
 android:id="@+id/color\_white"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 android:background="@color/white"  
 android:onClick="white"/>  
 <ImageButton  
 android:id="@+id/color\_yellow"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 android:background="@color/yellow"  
 android:onClick="yellow"/>  
 <ImageButton  
 android:id="@+id/color\_blue"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 android:background="@color/blue"  
 android:onClick="blue"/>  
 <ImageButton  
 android:id="@+id/color\_green"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 android:background="@color/green"  
 android:onClick="green"/>  
 <ImageButton  
 android:id="@+id/color\_red"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 android:background="@color/red"  
 android:onClick="red"/>  
</LinearLayout>

在新建note\_color.xml之前创建好color.xml文件，定义好几种颜色

*<?*xml version="1.0" encoding="utf-8"*?>*<resources>  
 <color name="black">#000000</color>  
 <color name="yellow">#FFFF00</color>  
 <color name="blue">#0000FF</color>  
 <color name="green">#008000</color>  
 <color name="red">#FF0000</color>  
 <color name="white">#FFFFFF</color>  
</resources>

创建NoteColor的Acitvity，用来选择颜色

public class NoteColor extends Activity {  
 private Cursor mCursor;  
 private Uri mUri;  
 private int color;  
 private static final int *COLUMN\_INDEX\_TITLE* = 1;  
 private static final String[] *PROJECTION* = new String[] {  
 NotePad.Notes.*\_ID*, *// 0* NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*,  
 };  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*note\_color*);  
 *//从NoteEditor传入的uri* mUri = getIntent().getData();  
 mCursor = managedQuery(  
 mUri, *// The URI for the note that is to be retrieved.  
 PROJECTION*, *// The columns to retrieve* null, *// No selection criteria are used, so no where columns are needed.* null, *// No where columns are used, so no where values are needed.* null *// No sort order is needed.* );  
 }  
 @Override  
 protected void onResume(){  
 *//执行顺序在onCreate之后* if (mCursor != null) {  
 mCursor.moveToFirst();  
 color = mCursor.getInt(*COLUMN\_INDEX\_TITLE*);  
 }  
 super.onResume();  
 }  
 @Override  
 protected void onPause() {  
 *//执行顺序在finish()之后，将选择的颜色存入数据库* super.onPause();  
 ContentValues values = new ContentValues();  
 values.put(NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR*, color);  
 getContentResolver().update(mUri, values, null, null);  
 }  
 public void white(View view){  
 color = NotePad.Notes.*DEFAULT\_COLOR*;  
 finish();  
 }  
 public void yellow(View view){  
 color = NotePad.Notes.*YELLOW\_COLOR*;  
 finish();  
 }  
 public void blue(View view){  
 color = NotePad.Notes.*BLUE\_COLOR*;  
 finish();  
 }  
 public void green(View view){  
 color = NotePad.Notes.*GREEN\_COLOR*;  
 finish();  
 }  
 public void red(View view){  
 color = NotePad.Notes.*RED\_COLOR*;  
 finish();  
 }  
  
}

在AndroidManifest.xml中新增NoteColor的Acitvity，并把该活动的主题定义为对话框样式

<activity android:name="NoteColor"  
 android:theme="@android:style/Theme.Holo.Light.Dialog"  
 android:label="ChangeColor"  
 android:windowSoftInputMode="stateVisible"/>

在NoteEditor中添加函数changeColor()，跳转改变颜色的activity，将uri信息传到新的activity

private final void changeColor() {  
 Intent intent = new Intent(null,mUri);  
 intent.setClass(NoteEditor.this,NoteColor.class);  
 NoteEditor.this.startActivity(intent);  
}

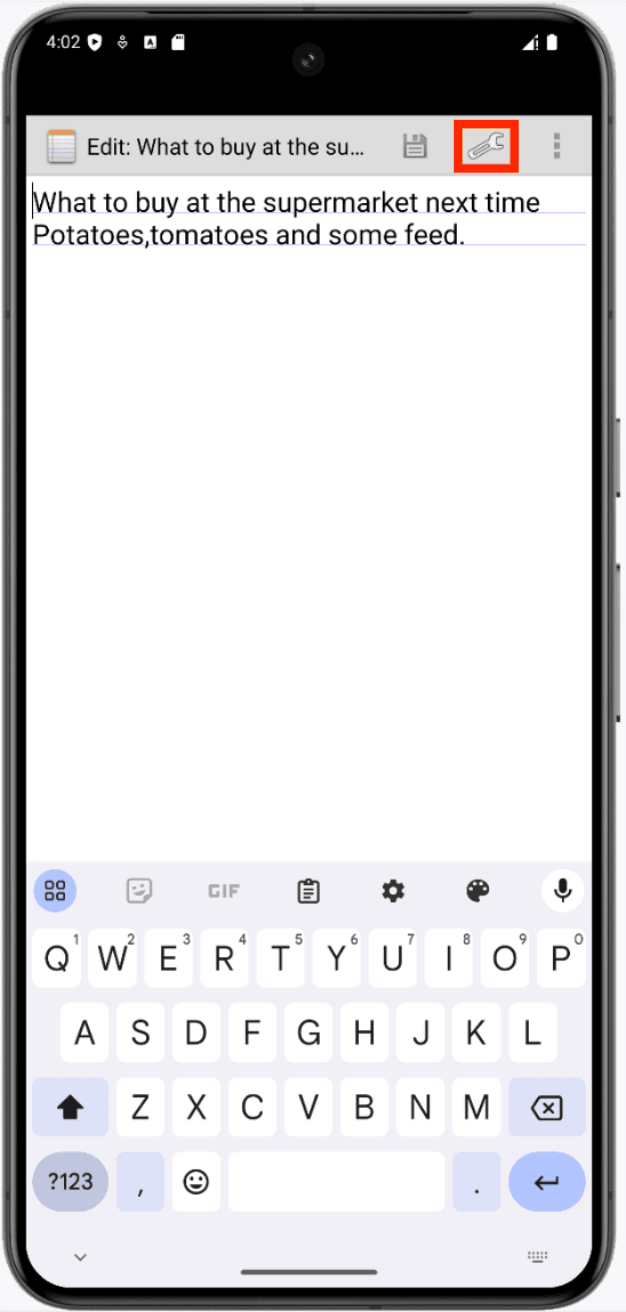
在NoteEditor.java中的onOptionsItemSelected方法有一段switch-case语句，实现了点击编辑界面的右上方按钮后的点击交互事件。在这里添加修改背景颜色的case语句

case R.id.*menu\_color*:  
 changeColor();  
 break;

1. 实现效果界面截图

在笔记编辑时对其切换编辑界面的背景颜色

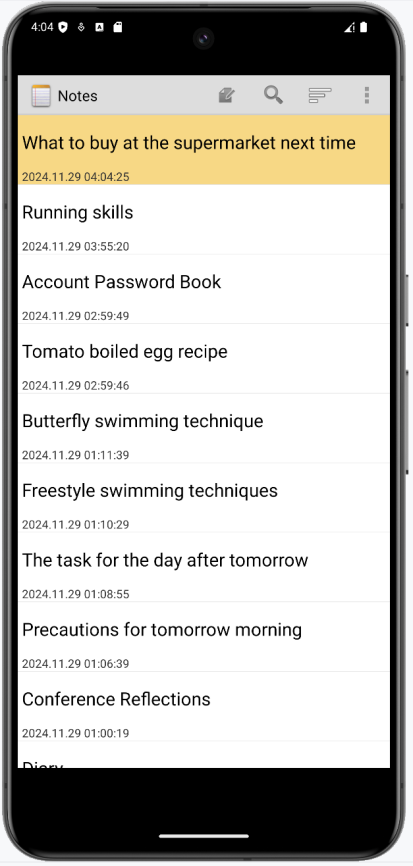
A.点击笔记进入编辑界面，在编辑界面中点击右上方的改变背景颜色按钮



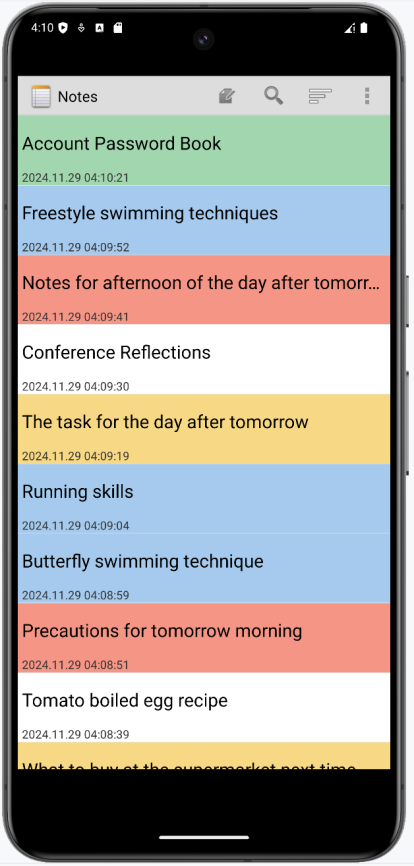
B.选择背景颜色



C.修改背景颜色后的编辑页面和返回应用主页面后的笔记列表



更换背景颜色后的笔记列表



1. 笔记导出
2. 功能要求：在笔记编辑界面点击右上角的菜单按钮，显示导出文件选项，点击后显示对话框，用户输入导入文件的名称后点击确定即可将指定笔记保存到本地
3. 实现思路和技术实现

在编辑界面的菜单xml文件（editor\_options\_menu.xml）中添加导出文件的item

<item android:id="@+id/menu\_output"  
 android:title="@string/menu\_output" />

编辑点击导出文件选项后显示的对话框。新建布局output\_text.xml，垂直线性布局放置EditText和Button

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:paddingLeft="6dip"  
 android:paddingRight="6dip"  
 android:paddingBottom="3dip">  
 <EditText android:id="@+id/output\_name"  
 android:maxLines="1"  
 android:layout\_marginTop="2dp"  
 android:layout\_marginBottom="15dp"  
 android:layout\_width="wrap\_content"  
 android:ems="25"  
 android:layout\_height="wrap\_content"  
 android:autoText="true"  
 android:capitalize="sentences"  
 android:scrollHorizontally="true" />  
 <Button android:id="@+id/output\_ok"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:text="@string/output\_ok"  
 android:onClick="OutputOk" />  
</LinearLayout>

3创建导出文件的活动OutputText，用来实现导出文件

public class OutputText extends Activity {  
 *//要使用的数据库中笔记的信息* private static final String[] *PROJECTION* = new String[] {  
 NotePad.Notes.*\_ID*, *// 0* NotePad.Notes.*COLUMN\_NAME\_TITLE*, *// 1* NotePad.Notes.*COLUMN\_NAME\_NOTE*, *// 2* NotePad.Notes.*COLUMN\_NAME\_CREATE\_DATE*, *// 3* NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*, *// 4* };  
 *//读取出的值放入这些变量* private String TITLE;  
 private String NOTE;  
 private String CREATE\_DATE;  
 private String MODIFICATION\_DATE;  
 *//读取该笔记信息* private Cursor mCursor;  
 *//导出文件的名字* private EditText mName;  
 *//NoteEditor传入的uri，用于从数据库查出该笔记* private Uri mUri;  
 *//关于返回与保存按钮的一个特殊标记，返回的话不执行导出，点击按钮才导出* private boolean flag = false;  
 private static final int *COLUMN\_INDEX\_TITLE* = 1;  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*output\_text*);  
 mUri = getIntent().getData();  
 mCursor = managedQuery(  
 mUri, *// The URI for the note that is to be retrieved.  
 PROJECTION*, *// The columns to retrieve* null, *// No selection criteria are used, so no where columns are needed.* null, *// No where columns are used, so no where values are needed.* null *// No sort order is needed.* );  
 mName = (EditText) findViewById(R.id.*output\_name*);  
 }  
 @Override  
 protected void onResume(){  
 super.onResume();  
 if (mCursor != null) {  
 *// The Cursor was just retrieved, so its index is set to one record \*before\* the first  
 // record retrieved. This moves it to the first record.* mCursor.moveToFirst();  
 *//编辑框默认的文件名为标题，可自行更改* mName.setText(mCursor.getString(*COLUMN\_INDEX\_TITLE*));  
 }  
 }  
 @Override  
 protected void onPause() {  
 super.onPause();  
 if (mCursor != null) {  
 *//从mCursor读取对应值* TITLE = mCursor.getString(mCursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_TITLE*));  
 NOTE = mCursor.getString(mCursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_NOTE*));  
 CREATE\_DATE = mCursor.getString(mCursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_CREATE\_DATE*));  
 MODIFICATION\_DATE = mCursor.getString(mCursor.getColumnIndex(NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE*));  
 *//flag在点击导出按钮时会设置为true，执行写文件* if (flag == true) {  
 write();  
 }  
 flag = false;  
 }  
 }  
 public void OutputOk(View v){  
 flag = true;  
 finish();  
 }  
 private void write()  
 {  
 try  
 {  
 *// 如果手机插入了SD卡，而且应用程序具有访问SD的权限* if (Environment.*getExternalStorageState*().equals(  
 Environment.*MEDIA\_MOUNTED*)) {  
 *// 获取SD卡的目录* File sdCardDir = Environment.*getExternalStorageDirectory*();  
 *//创建文件目录* File targetFile = new File(sdCardDir.getCanonicalPath() + "/" + mName.getText() + ".txt");  
 *//写文件* PrintWriter ps = new PrintWriter(new OutputStreamWriter(new FileOutputStream(targetFile), "UTF-8"));  
 ps.println(TITLE);  
 ps.println(NOTE);  
 ps.println("创建时间：" + CREATE\_DATE);  
 ps.println("最后一次修改时间：" + MODIFICATION\_DATE);  
 ps.close();  
 Toast.*makeText*(this, "保存成功,保存位置：" + sdCardDir.getCanonicalPath() + "/" + mName.getText() + ".txt", Toast.*LENGTH\_LONG*).show();  
 }  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

在AndroidManifest.xml中新增OutputText的Acitvity，并把该活动的主题定义为对话框样式

<activity android:name="OutputText"  
 android:label="@string/output\_name"  
 android:theme="@android:style/Theme.Holo.Dialog"  
 android:windowSoftInputMode="stateVisible">  
</activity>

在AndroidManifest.xml中新增对SD卡进行创建、删除文件和向SD卡写入数据的权限，注意和application同级，否则程序报错无法运行

*<!-- 在SD卡中创建与删除文件权限 -->*<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"  
 tools:ignore="ProtectedPermissions,WrongManifestParent" />  
*<!-- 向SD卡写入数据权限 -->*<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"  
 tools:ignore="WrongManifestParent" />

在NoteEditor中添加函数outputNote()，跳转导出笔记的activity，将uri信息传到新的activity

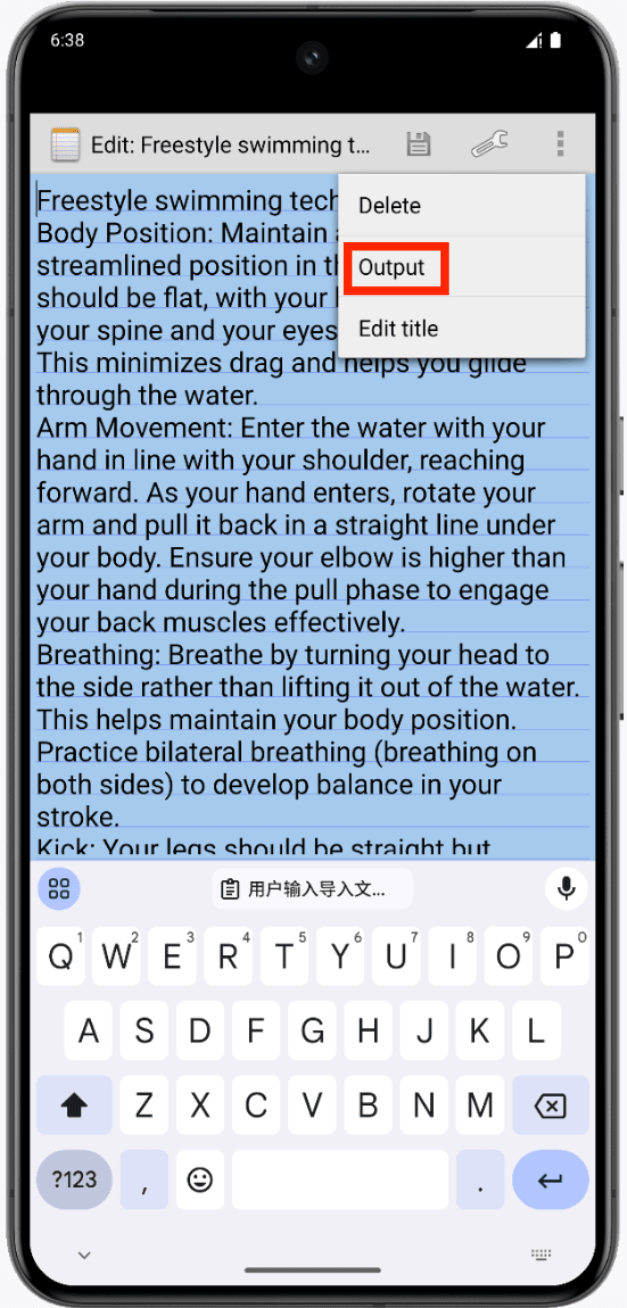
private final void outputNote() {  
 Intent intent = new Intent(null,mUri);  
 intent.setClass(NoteEditor.this,OutputText.class);  
 NoteEditor.this.startActivity(intent);  
}

在NoteEditor.java中的onOptionsItemSelected方法中的一段switch-case语句中添加导出文件的case语句

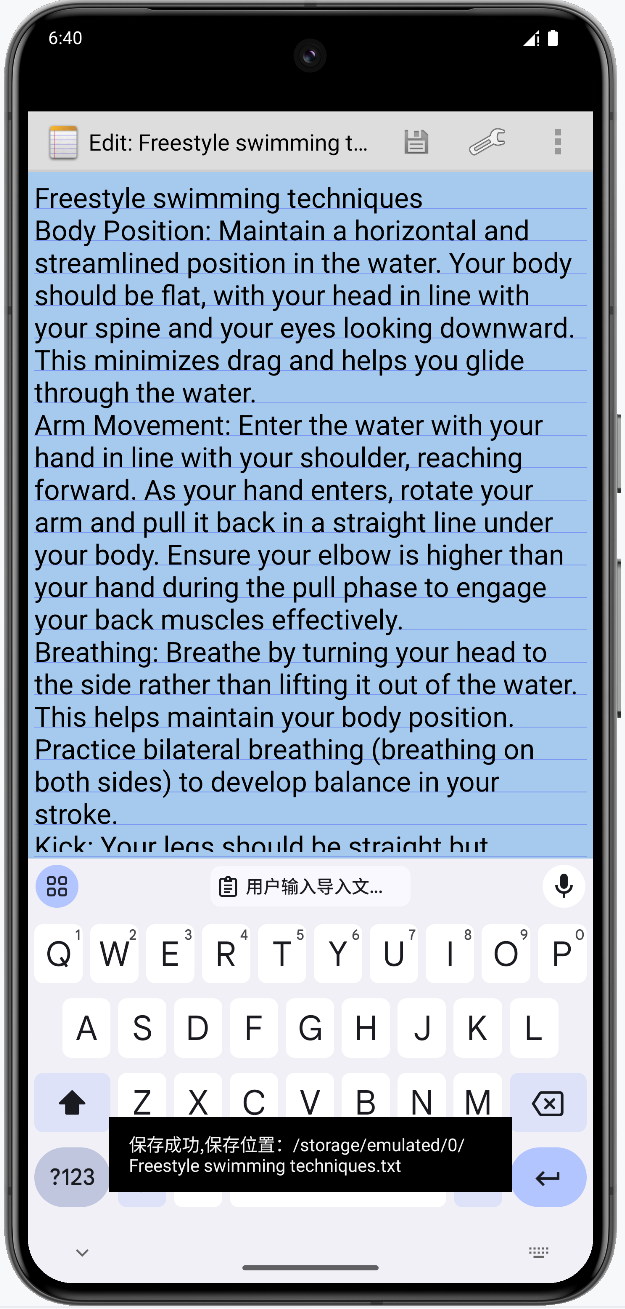
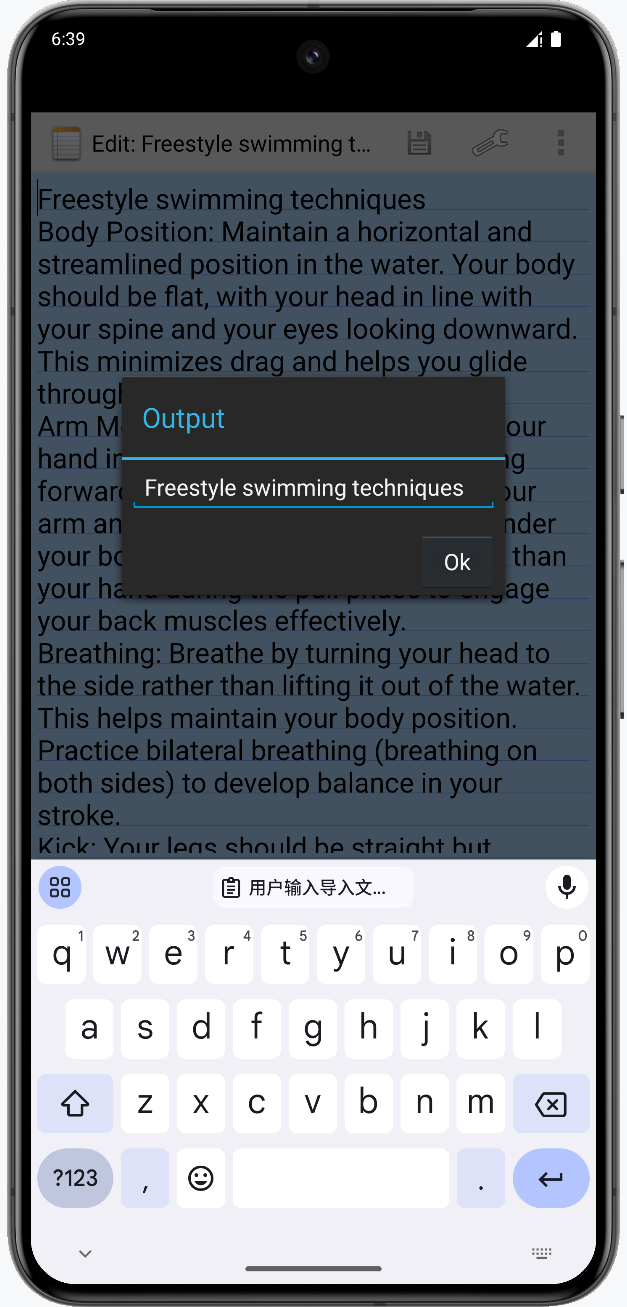
case R.id.*menu\_output*:  
 outputNote();  
 break;

1. 实现效果界面截图

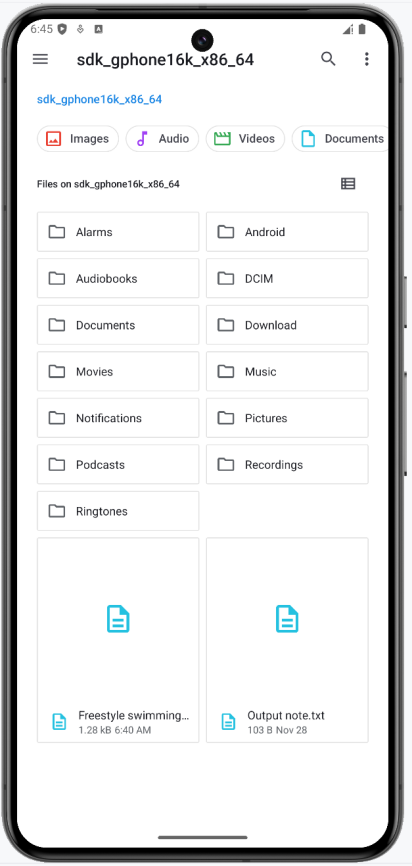
在笔记编辑界面点击右上角的菜单按钮，显示导出文件选项，点击后显示对话框



用户在对话框中输入导入文件的名称后点击OK，指定笔记被保存到本地



打开模拟器的文件管理器查看笔记是否被保存到本地



1. 笔记排序
2. 功能要求：在应用主界面点击排序按钮，便会显示三种排序方式：创建时间、修改时间、颜色。点击一种排序方式后，笔记列表依据选中的排序方式进行排序
3. 实现思路和技术实现

在应用主界面的菜单xml文件（list\_options\_menu.xml）中添加排序图标的item

<item  
 android:id="@+id/menu\_sort"  
 android:title="@string/menu\_sort"  
 android:icon="@android:drawable/ic\_menu\_sort\_by\_size"  
 android:showAsAction="always" >  
 <menu>  
 <item  
 android:id="@+id/menu\_sort1"  
 android:title="@string/menu\_sort1"/>  
 <item  
 android:id="@+id/menu\_sort2"  
 android:title="@string/menu\_sort2"/>  
 <item  
 android:id="@+id/menu\_sort3"  
 android:title="@string/menu\_sort3"/>  
 </menu>  
</item>

在NotesList.java中的onOptionsItemSelected方法有一段switch-case语句，实现了点击应用主界面的右上方按钮后的点击交互事件。在这里添加搜索的case语句

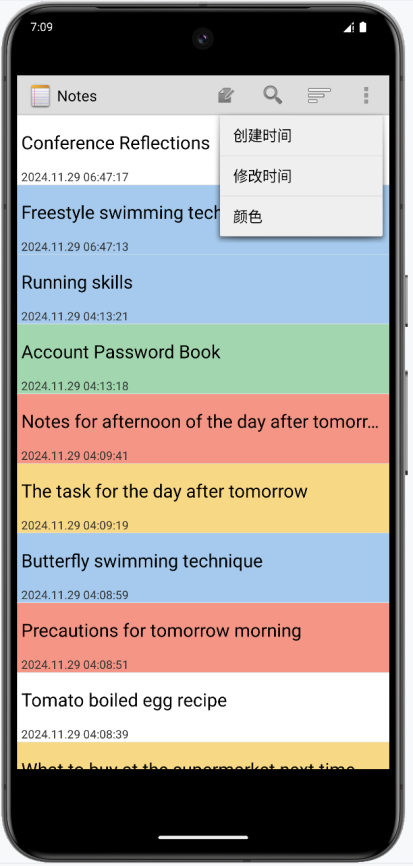
*//创建时间排序*case R.id.*menu\_sort1*:  
 cursor = managedQuery(  
 getIntent().getData(),  
 *PROJECTION*,  
 null,  
 null,  
 NotePad.Notes.*\_ID* );  
 adapter = new MyCursorAdapter(  
 this,  
 R.layout.*noteslist\_item*,  
 cursor,  
 dataColumns,  
 viewIDs  
 );  
 setListAdapter(adapter);  
 return true;  
*//修改时间排序*case R.id.*menu\_sort2*:  
 cursor = managedQuery(  
 getIntent().getData(),  
 *PROJECTION*,  
 null,  
 null,  
 NotePad.Notes.*DEFAULT\_SORT\_ORDER* );  
 adapter = new MyCursorAdapter(  
 this,  
 R.layout.*noteslist\_item*,  
 cursor,  
 dataColumns,  
 viewIDs  
 );  
 setListAdapter(adapter);  
 return true;  
*//颜色排序*case R.id.*menu\_sort3*:  
 cursor = managedQuery(  
 getIntent().getData(),  
 *PROJECTION*,  
 null,  
 null,  
 NotePad.Notes.*COLUMN\_NAME\_BACK\_COLOR* );  
 adapter = new MyCursorAdapter(  
 this,  
 R.layout.*noteslist\_item*,  
 cursor,  
 dataColumns,  
 viewIDs  
 );  
 setListAdapter(adapter);  
 return true;

由于排序会多次使用到cursor，adapter，所以将adapter,cursor,dataColumns,viewIDs定义在onOptionsItemSelected函数外，NotesList类内

private MyCursorAdapter adapter;  
private Cursor cursor;  
private String[] dataColumns = { NotePad.Notes.*COLUMN\_NAME\_TITLE* , NotePad.Notes.*COLUMN\_NAME\_MODIFICATION\_DATE* } ;  
private int[] viewIDs = { android.R.id.*text1* , R.id.*text1\_time* };

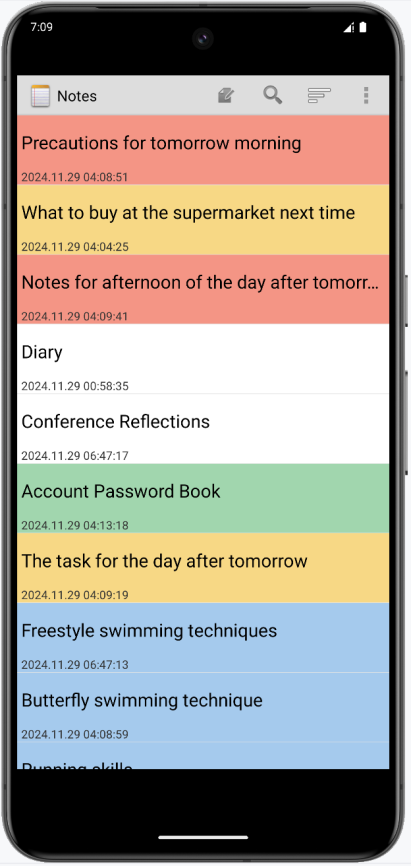
1. 实现效果界面截图

1在应用主界面点击排序按钮，便会显示三种排序方式

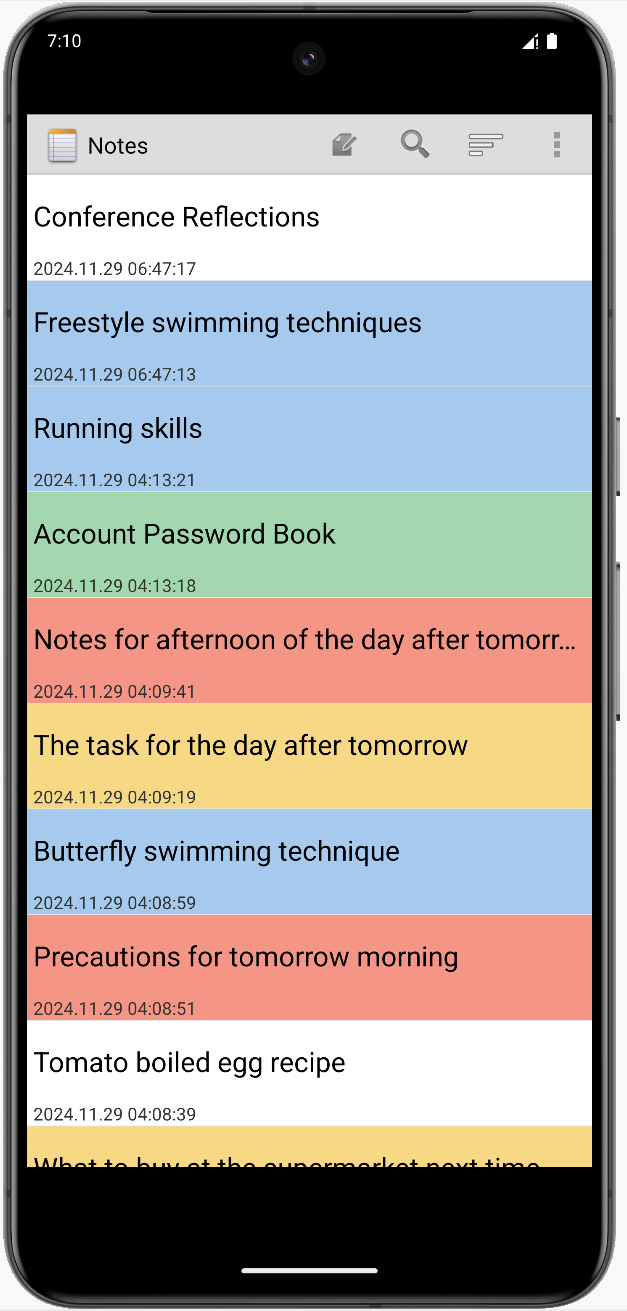


2点击一种排序方式后，笔记列表依据选中的排序方式进行排序

A.按照创建时间排序



B.按照修改时间排序



C.按照颜色排序

