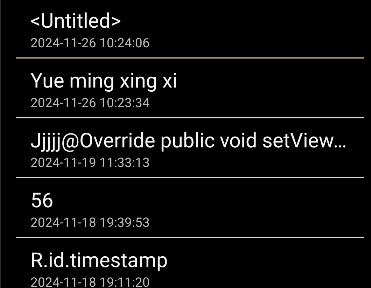
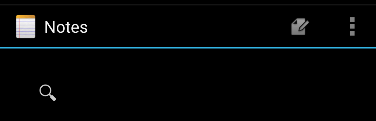
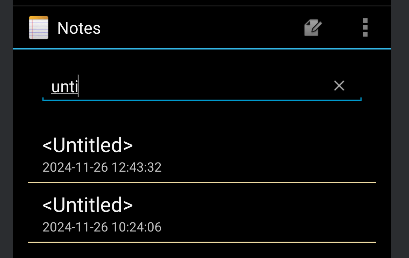
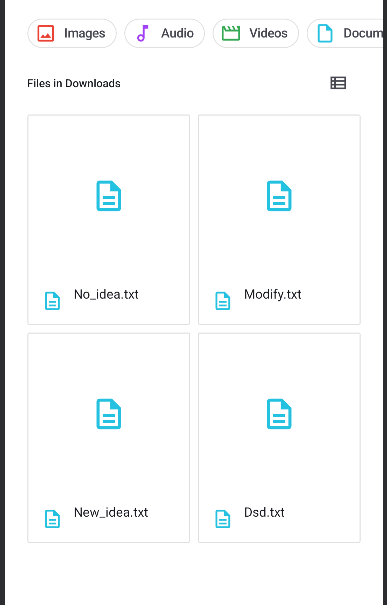
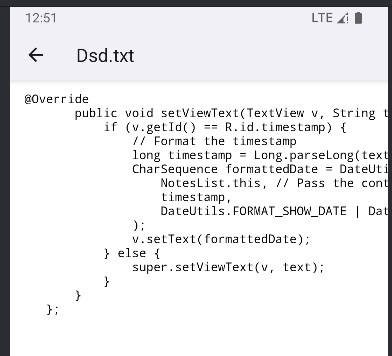
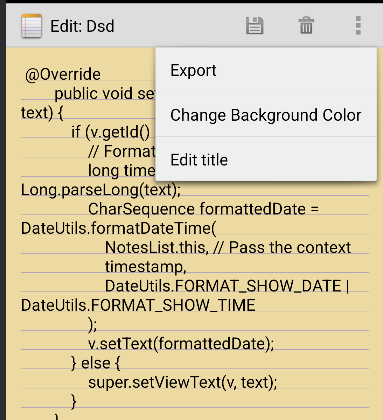
# 期中项目README  
  
## 时间戳功能  
  
### 效果展示：  
  
  
  
### 实现思路：  
  
1. NotePadProvider.java 中的时间戳插入和更新  
  
```  
@Override  
public Uri insert(Uri uri, ContentValues initialValues) {  
 // Validates the incoming URI. Only the full provider URI is allowed for inserts.  
 if (sUriMatcher.match(uri) != NOTES) {  
 throw new IllegalArgumentException("Unknown URI " + uri);  
 }  
  
 // A map to hold the new record's values.  
 ContentValues values;  
  
 // If the incoming values map is not null, uses it for the new values.  
 if (initialValues != null) {  
 values = new ContentValues(initialValues);  
  
 } else {  
 // Otherwise, create a new value map  
 values = new ContentValues();  
 }  
  
 // Gets the current system time in milliseconds  
 Long now = Long.valueOf(System.currentTimeMillis());  
  
 // 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, now);  
 }  
  
 // 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, now);  
 }  
  
 // 如果值映射中不包含标题，设置默认标题。  
 if (values.containsKey(NotePad.Notes.COLUMN\_NAME\_TITLE) == false) {  
 Resources r = Resources.getSystem();  
 values.put(NotePad.Notes.COLUMN\_NAME\_TITLE, r.getString(android.R.string.untitled));  
 }  
  
 // 如果值映射中不包含笔记内容，设置为空字符串。  
 if (values.containsKey(NotePad.Notes.COLUMN\_NAME\_NOTE) == false) {  
 values.put(NotePad.Notes.COLUMN\_NAME\_NOTE, "");  
 }  
  
 // 打开数据库对象的“写”模式。  
 SQLiteDatabase db = mOpenHelper.getWritableDatabase();  
  
 // 执行插入操作并返回新笔记的行ID。  
 long rowId = db.insert(  
 NotePad.Notes.TABLE\_NAME, // 插入的表名  
 NotePad.Notes.COLUMN\_NAME\_NOTE, // 如果值映射为空，SQLite会将此列值设为null  
 values // 包含列名和值的映射  
 );  
  
 // 如果插入成功，行ID存在。  
 if (rowId > 0) {  
 // 创建带有笔记ID模式的新URI。  
 Uri noteUri = ContentUris.withAppendedId(NotePad.Notes.CONTENT\_ID\_URI\_BASE, rowId);  
  
 // 通知观察者数据已更改。  
 getContext().getContentResolver().notifyChange(noteUri, null);  
 return noteUri;  
 }  
  
 // 如果插入失败，抛出异常。  
 throw new SQLException("Failed to insert row into " + uri);  
}  
```  
  
#### 解释：  
  
- 获取当前时间： Long now = Long.valueOf(System.currentTimeMillis());  
- 获取当前系统时间的毫秒值。  
- 设置创建时间： values.put(NotePad.Notes.COLUMN\_NAME\_CREATE\_DATE, now);  
- 如果插入的值映射中没有创建时间，则设置为当前时间。  
- 设置修改时间： values.put(NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE, now);  
- 如果插入的值映射中没有修改时间，则设置为当前时间。  
  
2. NoteEditor.java 中的时间戳更新  
  
```  
private final void updateNote(String text, String title) {  
 // 设置一个包含要更新的值的映射。  
 ContentValues values = new ContentValues();  
 values.put(NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE, System.currentTimeMillis());  
  
 // 如果提供了新的标题，则更新标题。  
 if (title != null) {  
 values.put(NotePad.Notes.COLUMN\_NAME\_TITLE, title);  
 }  
  
 // 更新笔记内容。  
 values.put(NotePad.Notes.COLUMN\_NAME\_NOTE, text);  
  
 // 更新数据库中的记录。  
 int count = getContentResolver().update(mUri, values, null, null);  
  
 // 如果更新成功，通知观察者数据已更改。  
 if (count > 0) {  
 mCursor.requery();  
 mOriginalContent = text;  
 }  
}  
```  
  
#### 解释：  
  
- 设置修改时间： values.put(NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE, System.currentTimeMillis());  
- 在每次更新笔记时，设置修改时间为当前时间。  
- 更新数据库记录： getContentResolver().update(mUri, values, null, null);  
- 使用 ContentResolver 更新数据库中的记录，确保修改时间被更新。  
  
3. NotesList.java 中的时间戳显示  
  
```  
SimpleCursorAdapter adapter = new SimpleCursorAdapter(  
 this,  
 R.layout.noteslist\_item,  
 cursor,  
 new String[]{NotePad.Notes.COLUMN\_NAME\_TITLE, NotePad.Notes.COLUMN\_NAME\_MODIFICATION\_DATE},  
 new int[]{android.R.id.text1, R.id.timestamp},  
 0  
) {  
 @Override  
 public void setViewText(TextView v, String text) {  
 if (v.getId() == R.id.timestamp) {  
 long timestamp = Long.parseLong(text);  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault());  
 sdf.setTimeZone(TimeZone.getTimeZone("Asia/Shanghai"));  
 String formattedDate = sdf.format(new Date(timestamp));  
 v.setText(formattedDate);  
 } else {  
 super.setViewText(v, text);  
 }  
 }  
};  
```  
  
#### 解释：  
  
- 解析时间戳： long timestamp = Long.parseLong(text);  
- 将从数据库中获取的时间戳字符串转换为长整型。  
- 格式化时间戳： SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault());  
- 使用 SimpleDateFormat 将时间戳格式化为可读的日期时间字符串。  
- 设置时间戳文本： v.setText(formattedDate);  
- 将格式化后的日期时间字符串设置到 TextView 中。  
  
## 笔记查询功能  
  
### 效果展示  
  
  
  
### 实现思路  
  
1. NotesList.java 中的 performSearch 方法  
  
```  
private void performSearch(String query) {  
 if (query.isEmpty()) {  
 // 恢复到初始状态，显示所有笔记  
 Cursor cursor = managedQuery(  
 NotePad.Notes.CONTENT\_URI,  
 PROJECTION,  
 null,  
 null,  
 NotePad.Notes.DEFAULT\_SORT\_ORDER  
 );  
 SimpleCursorAdapter adapter = (SimpleCursorAdapter) getListAdapter();  
 adapter.changeCursor(cursor);  
 } else {  
 String selection = NotePad.Notes.COLUMN\_NAME\_TITLE + " LIKE ? OR " + NotePad.Notes.COLUMN\_NAME\_NOTE + " LIKE ?";  
 String[] selectionArgs = {"%" + query + "%", "%" + query + "%"};  
 Cursor cursor = managedQuery(  
 NotePad.Notes.CONTENT\_URI, // 直接使用 NotePad.Notes.CONTENT\_URI 而不是 Uri.withAppendedPath  
 PROJECTION,  
 selection,  
 selectionArgs,  
 NotePad.Notes.DEFAULT\_SORT\_ORDER  
 );  
  
 // 添加日志输出，检查查询结果  
 if (cursor == null || cursor.getCount() == 0) {  
 Log.d(TAG, "No notes found for query: " + query);  
 } else {  
 Log.d(TAG, "Found " + cursor.getCount() + " notes for query: " + query);  
 }  
  
 SimpleCursorAdapter adapter = (SimpleCursorAdapter) getListAdapter();  
 adapter.changeCursor(cursor);  
 }  
}  
```  
  
#### 解释：  
  
- *功能：* : 这个方法用于执行笔记的搜索操作。它根据用户输入的查询字符串 query 来过滤笔记列表。  
  
- *逻辑：* : 如果查询字符串为空，则恢复到初始状态，显示所有笔记。  
 : 否则，构建一个 SQL 查询语句，使用 LIKE 关键字来匹配笔记的标题或内容。  
 : 执行查询并更新 SimpleCursorAdapter 以显示新的查询结果。  
 : 添加日志输出，用于调试和检查查询结果。  
  
2. NotePadProvider.java 中的 query 方法  
  
```  
@Override  
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {  
 SQLiteQueryBuilder qb = new SQLiteQueryBuilder();  
 qb.setTables(NotePad.Notes.TABLE\_NAME);  
  
 switch (sUriMatcher.match(uri)) {  
 case NOTES:  
 qb.setProjectionMap(sNotesProjectionMap);  
 break;  
 case NOTE\_ID:  
 qb.setProjectionMap(sNotesProjectionMap);  
 qb.appendWhere(NotePad.Notes.\_ID + "=" + uri.getPathSegments().get(NotePad.Notes.NOTE\_ID\_PATH\_POSITION));  
 break;  
 case LIVE\_FOLDER\_NOTES:  
 qb.setProjectionMap(sLiveFolderProjectionMap);  
 break;  
 case NOTES\_SEARCH:  
 qb.setProjectionMap(sNotesProjectionMap);  
 if (selection != null && !selection.isEmpty()) {  
 qb.appendWhere(selection);  
 }  
 break;  
 default:  
 throw new IllegalArgumentException("Unknown URI " + uri);  
 }  
  
 String orderBy;  
 if (TextUtils.isEmpty(sortOrder)) {  
 orderBy = NotePad.Notes.DEFAULT\_SORT\_ORDER;  
 } else {  
 orderBy = sortOrder;  
 }  
  
 SQLiteDatabase db = mOpenHelper.getReadableDatabase();  
 Cursor c = qb.query(db, projection, selection, selectionArgs, null, null, orderBy);  
 c.setNotificationUri(getContext().getContentResolver(), uri);  
 return c;  
}  
```  
  
#### 解释：  
  
- *功能：这个方法是 NotePadProvider 中的核心方法，用于处理来自客户端的查询请求。*- *逻辑：* : 使用 SQLiteQueryBuilder 构建查询。  
 : 根据传入的 URI 匹配不同的模式（如 NOTES、NOTE\_ID、LIVE\_FOLDER\_NOTES 和 NOTES\_SEARCH）来设置不同的查询条件。  
 : 对于 NOTES\_SEARCH 模式，如果传入的 selection 不为空，则将其附加到查询条件中。  
 : 设置默认的排序顺序，如果客户端没有指定排序顺序。  
 : 执行查询并返回 Cursor 对象。  
 : 设置通知 URI，以便在数据变化时通知客户端  
  
## 导出笔记功能  
  
### 效果展示  
  
  
  
### 实现思路  
1. 在 NoteEditor.java 文件中，导出笔记的功能主要集中在 exportNote() 方法中  
```  
private void exportNote() {  
 String title = mCursor.getString(mCursor.getColumnIndex(NotePad.Notes.COLUMN\_NAME\_TITLE));  
 String content = mText.getText().toString();  
  
 // Create a file name  
 String fileName = title.replaceAll("[^a-zA-Z0-9]", "\_") + ".txt";  
  
 // Get the desktop directory  
 File dir = new File(Environment.getExternalStorageDirectory(), "Download"); // 修改这里  
 if (!dir.exists()) {  
 dir.mkdirs();  
 }  
  
 // Create a file  
 File file = new File(dir, fileName);  
  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  
 if (checkSelfPermission(Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) {  
 requestPermissions(new String[]{Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, REQUEST\_WRITE\_STORAGE);  
 return;  
 }  
 }  
 try {  
 // Write the note content to the file  
 FileOutputStream fos = new FileOutputStream(file);  
 fos.write(content.getBytes());  
 fos.close();  
  
 // Show a toast message  
 Toast.makeText(this, "Note exported to " + file.getAbsolutePath(), Toast.LENGTH\_LONG).show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 Toast.makeText(this, "Failed to export note", Toast.LENGTH\_SHORT).show();  
 }  
}  
```  
  
#### 解释：  
  
- title：从 mCursor 中获取当前笔记的标题。  
- content：从 mText（即 EditText 控件）中获取当前笔记的内容。  
- 使用正则表达式将标题中的非字母数字字符替换为下划线，并添加 .txt 扩展名，生成文件名  
- 获取外部存储的下载目录。 如果目录不存在，则创建目录  
- 如果设备的 API 级别大于等于 23（Marshmallow），检查是否已授予写入外部存储的权限。 如果没有权限，请求权限并返回。  
- 使用 FileOutputStream 将笔记内容写入文件。 写入成功后，显示一条 Toast 消息，告知用户文件已导出的路径。 如果写入过程中发生 IOException，捕获异常并显示失败的 Toast 消息。