# 移动互联网技术及应用

# 大作业报告

## 题目：NFC 打卡系统的设计与实现

## 类型：应用系统设计实现


| 姓名 | 班级 | 学号 |
| --- | --- | --- |
| 冯涛 | 2021211303 | 2021211023 |


**2024.5.31**

# 目录



# 1 相关技术

## 1.1 NFC（近场通信）技术

(1) 技术介绍：NFC（Near Field Communication）是一种基于无线射频技术的近距离通信技术，允许两个设备之间在非接触的情况下进行通信。它可以在手机、智能卡、标签等设备上实现，并且在支付、数据传输、身份验证、门禁控制等领域有广泛应用。NFC 工作在 13.56MHz 频率，通信距离一般为几厘米，确保通信的安全性和隐私性。 NFC 的三种工作模式包括读卡器模式、卡模式和点对点模式，分别允许设备作为主动器、被动器或者两者之间的等价通信

(2) 实际用例：

[1] 检测 NFC 标签：通过 NfcAdapter 类检测并读取 NFC 标签。

[2] 前台调度系统：使用 PendingIntent 和 IntentFilter 设置前台调度系统，以确保应用在检测到 NFC 标签时能够立即响应。

[3] 处理 NFC 标签：在 onNewIntent 方法中处理新的 Intent，读取 NFC 标签的数据并进行相应的处理。

## 1.2 SQLite 数据库

(1) 技术介绍：SQLite 是一个开源的嵌入式关系数据库，实现了自给自足的、无服务器的、配置无需的、事务性的 SQL 数据库引擎。它是一个零配置的数据库，这意味着与其他数据库系统不同，比如 MySQL、PostgreSQL 等，SQLite 不需要在

系统中设置和管理一个单独的服务。这也使得 SQLite 是一种非常轻量级的数据库解决方案，非常适合小型项目、嵌入式数据库或者测试环境中。

(2) 实际用例：

[1] SQLiteOpenHelper：通过扩展 SQLiteOpenHelper 类创建和管理数据库。

[2] SQL 语句：使用 SQL 语句创建表（如 CardInfo 和 Attendance 表）、插入数据、查询和删除记录。

[3] ContentValues：用于存储一组键值对，便于将数据插入到数据库表中。

# 2 系统功能需求

## 2.1 系统功能描述

(1) 录入模式：未录入系统的 NFC 卡片会通过此功能录入系统，将识别到的 NFC 卡片的标签 ID，将 ID 与对应的信息（班级-学号-姓名）写入本地数据库中，完成对未知卡片的录入。点击清除卡片录入信息，可以删除之前已经录入信息的卡片，将其还原为未知卡片，不再记录相关信息。

(2) 打卡模式：进入打卡模式后，输入打卡主题，即可进行一次关于该主题的打卡，当前打卡信息会在下方显示，能够查看到实时打卡情况，对打卡情况会保存到本地数据库，用于后续查看历史记录。

(3) 历史记录：在历史记录中，一共有 4 种搜索模式。首先是直接点击搜索按钮，将会显示已存在的所有打卡信息（班级-学号-姓名-打卡主题-打卡时间-打卡次数），也可以通过学号、打卡主题、打卡日期来搜索符合某一个具体条件的打卡信息。点击清除历史记录可以清除本地存储的打卡信息。

## 2.2 需求分析

(1) 背景：NFC（近场通信）技术因其便捷性和安全性，在考勤管理系统中得到了广泛应用。本应用旨在利用 NFC 技术实现学生考勤的自动化管理，通过识别 NFC 卡片，实现信息录入、考勤记录和历史查询等功能。

(2) 目标：开发一个基于 Android 平台的 NFC 打卡应用，能够识别学生的 NFC 卡片，记录打卡信息，并提供打卡历史查询功能。系统应具备高效、准确、易用和安全的特点。

# 3 系统设计与实现

## 3.1 总体设计

(1) 应用采用分层架构，主要包括以下层次：

[1] 表示层（UI 层）：负责用户界面的显示和用户交互。

[2] 业务逻辑层（Service 层）：处理应用的业务逻辑，包括打卡记录的处理和查询。

[3] 数据访问层（Data 层）：负责与数据库的交互，执行数据的存储、读取、更新和删除操作。

(2) 主要模块：

应用分为几个主要模块，各模块之间相互协作，实现完整的功能。主要模块包括：

[1] NFC 识别模块

[2] 信息录入模块

[3] 考勤打卡模块

[4] 历史查询模块

[5] 数据管理模块

3.2 各层模块设计

(1) NFC 识别模块：

[1] 功能：识别 NFC 卡片，获取卡片 ID。

[2] 主要类和方法：

- NfcAdapter: 用于初始化和管理 NFC 功能。
- PendingIntent: 用于设置前台调度系统。
- IntentFilter: 过滤特定类型的 NFC 标签。
- onNewIntent: 处理新检测到的 NFC 标签 Intent。
- bytesToHex: 将卡片 ID 字节数组转换为十六进制字符串。

(2) 信息录入模块：

[1] 录入学生信息（班级、学生 ID、姓名）并与 NFC 卡片关联。

[2] 主要类和方法：

- InputActivity: 提供用户界面和逻辑处理。
- DatabaseHelper: 管理数据库连接和操作。
- addCardInfo: 将卡片信息插入数据库的 CardInfo 表。

(3) 考勤打卡模块

[1] 功能：记录学生的打卡信息，包括打卡时间和主题。

[2] 主要类和方法：

- RecordActivity: 提供打卡界面和逻辑处理。
- DatabaseHelper: 管理数据库连接和操作。
- recordAttendance: 将打卡记录插入数据库的 Attendance 表。

(4) 历史查询模块

[1] 功能：查询和显示历史打卡记录，支持按条件（学生 ID、主题、日期）筛选。

[2] 主要类和方法：

- HistoryActivity: 提供查询界面和逻辑处理
- DatabaseHelper: 管理数据库连接和操作
- searchAttendanceRecords: 从数据库中查询打卡记录并显示在界面上

(5) 数据管理模块

[1] 功能：管理和维护数据库中的数据，支持数据的清除和重置

[2] 主要类和方法：

- DatabaseHelper: 管理数据库连接和操作。
- clearHistoryRecords: 清除历史打卡记录。
- clearCardInfo: 清除卡片信息。

3.3 数据库设计：应用使用 SQLite 数据库存储数据，主要包括两个表：

(1) CardInfo 表：

- CardID（TEXT）：主键，NFC 卡片 ID。

- Class（TEXT）：班级名称。
- StudentID（TEXT）：学生 ID。
- Name（TEXT）：学生姓名。

(2) Attendance 表：

- ID（INTEGER）：主键，自增。
- CardID（TEXT）：外键，关联 CardInfo 表。
- Class（TEXT）：班级名称。
- StudentID（TEXT）：学生 ID。
- Name（TEXT）：学生姓名。
- Timestamp（TEXT）：打卡时间。
- Theme（TEXT）：打卡主题。
- Count（INTEGER）：打卡次数。

3.4 用户界面设计

(1) 主界面（MainActivity）：提供导航按钮，分别进入信息录入、考勤打卡和历史查询界面，如图 3.4-1。

(2) 信息录入界面（InputActivity）：输入班级、学生 ID、姓名，并通过 NFC 卡片录入卡片信息，如图 3.4-2。

(3) 考勤打卡界面（RecordActivity）：输入打卡主题，并通过 NFC 卡片记录打卡信息，如图 3.4-3。

(4) 历史查询界面（HistoryActivity）：输入查询条件，查看历史打卡记录，如图 3.4-4。


图 3.4-1


图 3.4-2


图 3.4-3


图 3.4-4

## 3.5 关键代码解释

(1) 这些代码段为每个按钮设置了点击事件监听器，点击后分别启动 InputActivity、RecordActivity 和 HistoryActivity，实现了主界面的导航功能。

```
Button inputButton = findViewById(R.id.input_button);
Button recordButton = findViewById(R.id.record_button);
Button historyButton = findViewById(R.id.history_button);

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

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

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

```

(2) NFC 初始化和调度：NfcAdapter、PendingIntent 和 IntentFilter 用于初始化 NFC 功能，并设置前台调度系统，确保应用在检测到 NFC 标签时能够及时响应。

[1] onResume 和 onPause 方法：在活动生命周期中启用和禁用前台调度系统。

[2] onNewIntent 方法：处理新的 NFC 标签 Intent。

[3] addCardInfo 方法：将卡片信息保存到数据库。

```
nfcAdapter = NfcAdapter.getDefaultAdapter(this);
pendingIntent = PendingIntent.getActivity(this, 0, new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

intentFiltersArray = new IntentFilter[] {
        new IntentFilter(NfcAdapter.ACTION_TECH_DISCOVERED)
};
techListsArray = new String[][] {
        new String[] { NfcA.class.getName() },
        new String[] { NfcB.class.getName() },
        new String[] { IsoDep.class.getName() },
        new String[] { NfcF.class.getName() },
        new String[] { NfcV.class.getName() },
        new String[] { Ndef.class.getName() },
        new String[] { NdefFormatable.class.getName() }
};

@Override
protected void onResume() {
    super.onResume();
    if (nfcAdapter != null) {
        nfcAdapter.enableForegroundDispatch(this, pendingIntent, intentFiltersArray, techListsArray);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (nfcAdapter != null) {
        nfcAdapter.disableForegroundDispatch(this);
    }
}

@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
    if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(intent.getAction())) {
        Log.d("InputActivity", "NFC Tag Detected");
    }
}

private void addCardInfo(String cardID, String className, String studentID, String name) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("CardID", cardID);
    values.put("Class", className);
    values.put("StudentID", studentID);
    values.put("Name", name);
    db.insert("CardInfo", null, values);
    db.close();
}
```

(3) NFC 初始化和调度：类似 InputActivity 的 NFC 初始化和前台调度设置。

[1] onNewIntent 方法：检测到 NFC 标签时，如果输入的主题不为空，则调用 recordAttendance 方法记录打卡信息。

[2] recordAttendance 方法：根据卡片 ID 从数据库中查找学生信息，并将打卡记录保存到数据库。

[3] getCountForTheme 方法：获取当前主题的打卡次数。

[4] bytesToHex 方法：将字节数组转换为十六进制字符串，便于处理卡片 ID。

```
nfcAdapter = NfcAdapter.getDefaultAdapter(this);
pendingIntent = PendingIntent.getActivity(this, 0, new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

intentFiltersArray = new IntentFilter[] {
        new IntentFilter(NfcAdapter.ACTION_TECH_DISCOVERED)
};
techListsArray = new String[][] {
        new String[] { NfcA.class.getName() },
        new String[] { NfcB.class.getName() },
        new String[] { IsoDep.class.getName() },
        new String[] { NfcF.class.getName() },
        new String[] { NfcV.class.getName() },
        new String[] { Ndef.class.getName() },
        new String[] { NdefFormatable.class.getName() }
};

@Override
protected void onResume() {
    super.onResume();
    if (nfcAdapter != null) {
        nfcAdapter.enableForegroundDispatch(this, pendingIntent, intentFiltersArray, techListsArray);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (nfcAdapter != null) {
        nfcAdapter.disableForegroundDispatch(this);
    }
}

@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
    if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(intent.getAction())) {
        Log.d("RecordActivity", "NFC Tag Detected");
        if (editTextTheme.getText().toString().isEmpty()) {
            Toast.makeText(this, "请输入打卡主题", Toast.LENGTH_SHORT).show();
        } else {
            recordAttendance(bytesToHex(tag.getId()), editTextTheme.getText().toString());
        }
    }
}

private void recordAttendance(String cardID, String theme) {
```

```
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM CardInfo WHERE CardID = ?", new String[]{cardID});
    if (cursor.moveToFirst()) {
        String className = cursor.getString(cursor.getColumnIndex("Class"));
        String studentID = cursor.getString(cursor.getColumnIndex("StudentID"));
        String name = cursor.getString(cursor.getColumnIndex("Name"));
        String timestamp = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault()).format(new Date());

        int count = getCountForTheme(db, theme) + 1;

        ContentValues values = new ContentValues();
        values.put("CardID", cardID);
        values.put("Class", className);
        values.put("StudentID", studentID);
        values.put("Name", name);
        values.put("Timestamp", timestamp);
        values.put("Theme", theme);
        values.put("Count", count);
        db.insert("Attendance", null, values);

        recordList.add(new AttendanceRecord(className, studentID, name, timestamp, theme, count));
        recordAdapter.notifyDataSetChanged();

        Toast.makeText(this, "打卡成功：" + name, Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "未知卡片", Toast.LENGTH_SHORT).show();
    }
    cursor.close();
    db.close();
}

private int getCountForTheme(SQLiteDatabase db, String theme) {
    Cursor cursor = db.rawQuery("SELECT COUNT(*) FROM Attendance WHERE Theme = ?", new String[]{theme});
    int count = 0;
    if (cursor.moveToFirst()) {
        count = cursor.getInt(0);
    }
    cursor.close();
    return count;
}

private String bytesToHex(byte[] bytes) {
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}
```

(4) searchAttendanceRecords 方法：根据用户输入的查询条件从数据库中查找打卡记录，并更新显示在 RecyclerView 中。clearHistoryRecords 方法：清空数据库中的打卡记录，并更新显示。

```
searchButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        searchAttendanceRecords();
    }
});

clearHistoryButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clearHistoryRecords();
    }
});

private void searchAttendanceRecords() {
    String studentId = searchStudentId.getText().toString();
    String theme = searchTheme.getText().toString();
    String date = searchDate.getText().toString();
    SQLiteDatabase db = dbHelper.getReadableDatabase();
    StringBuilder queryBuilder = new StringBuilder("SELECT * FROM Attendance WHERE 1=1");
    List<String> args = new ArrayList<>();

    if (!studentId.isEmpty()) {
        queryBuilder.append(" AND StudentID = ?");
        args.add(studentId);
    }
    if (!theme.isEmpty()) {
        queryBuilder.append(" AND Theme = ?");
        args.add(theme);
    }
    if (!date.isEmpty()) {
        queryBuilder.append(" AND DATE(Timestamp) = DATE(?)");
        args.add(date);
    }

    Cursor cursor = db.rawQuery(queryBuilder.toString(), args.toArray(new String[0]));
    recordList.clear();
    int totalCount = 0;

    while (cursor.moveToNext()) {
        String className = cursor.getString(cursor.getColumnIndex("Class"));
        String studentID = cursor.getString(cursor.getColumnIndex("StudentID"));
        String studentName = cursor.getString(cursor.getColumnIndex("Name"));
        String timestamp = cursor.getString(cursor.getColumnIndex("Timestamp"));
        String themeResult = cursor.getString(cursor.getColumnIndex("Theme"));
        int count = cursor.getInt(cursor.getColumnIndex("Count"));

        AttendanceRecord record = new AttendanceRecord(className, studentID, studentName, timestamp, themeResult, count);
        recordList.add(record);
        totalCount++;
    }
    cursor.close();
    recordAdapter.notifyDataSetChanged();

    if (!studentId.isEmpty()) {
        totalAttendanceCount.setVisibility(View.VISIBLE);
        totalAttendanceCount.setText("总打卡次数: " + totalCount);
    } else {
        totalAttendanceCount.setVisibility(View.GONE);
    }
}

private void clearHistoryRecords() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    db.delete("Attendance", null, null);
    recordList.clear();
    recordAdapter.notifyDataSetChanged();
    totalAttendanceCount.setVisibility(View.GONE);
    Toast.makeText(this, "历史记录已清除", Toast.LENGTH_SHORT).show();
}
```

(5) onCreate 方法：创建两个表 CardInfo 和 Attendance，分别用于存储卡片信息和打卡记录。onUpgrade 方法：在数据库版本升级时删除旧表，并重新创建新表。

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE CardInfo (CardID TEXT PRIMARY KEY, Class TEXT, StudentID TEXT, Name TEXT)");
    db.execSQL("CREATE TABLE Attendance (ID INTEGER PRIMARY KEY AUTOINCREMENT, CardID TEXT, Class TEXT, StudentID TEXT, Name TEXT, Timestamp TEXT, Theme TEXT, Count INTEGER)");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS CardInfo");
    db.execSQL("DROP TABLE IF EXISTS Attendance");
    onCreate(db);
}

```

(6) 数据模型类：AttendanceRecord 用于表示打卡记录的数据模型，包括班级、学生ID、姓名、打卡时间、主题和打卡次数.

```
public class AttendanceRecord {
    private String className;
    private String studentId;
    private String studentName;
    private String timestamp;
    private String theme;
    private int count;

    public AttendanceRecord(String className, String studentId, String studentName, String timestamp, String theme, int count) {
        this.className = className;
        this.studentId = studentId;
        this.studentName = studentName;
        this.timestamp = timestamp;
        this.theme = theme;
        this.count = count;
    }

    public String getClassName() {
        return className;
    }

    public String getStudentId() {
        return studentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public String getTimestamp() {
        return timestamp;
    }

    public String getTheme() {
        return theme;
    }

    public int getCount() {
        return count;
    }
}

```

(7) 数据绑定：onBindViewHolder 方法将 AttendanceRecord 数据绑定到视图组件，显示打卡记录的详细信息。

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    AttendanceRecord record = records.get(position);
    holder.className.setText("班级：" + record.getClassName());
    holder.studentId.setText("学号：" + record.getStudentId());
    holder.studentName.setText("姓名：" + record.getStudentName());
    holder.timestamp.setText("时间：" + record.getTimestamp());
    holder.theme.setText("主题：" + record.getTheme());
    holder.count.setText("次数：" + String.valueOf(record.getCount()));
}

```

# 4 系统可能的扩展

4.1 云端数据存储：将本地数据库的数据转移到服务器上存储，使得在使用者更换设备或者误删 app 后能够通过账号得到历史数据，避免数据丢失造成损失。

4.2 学生端远程添加卡片信息：新加入班级的学生可以通过服务器将自己的 NFC 卡片与个人信息进行绑定并上传到服务器，更加方便快捷，避免需要使用老师的手机进行信息添加。

4.3 对历史数据的展示更加优化，分类更加多样和精确，例如：给出某位学生一学期的总打卡次数，某周的打卡次数等等，对其打卡情况进行更加合适的展示，便于管理者掌握打卡情况

4.4 加入打卡提醒功能，当打卡开始时，使用 Android 的 AlarmManager 实现定时提醒。结合推送服务（如 Firebase Cloud Messaging）发送通知。

4.5 定位打卡：结合 GPS 定位功能，确保打卡是在指定地点进行的，防止作弊。

# 5 总结体会

在设计和开发 NFC 打卡系统的过程中，结合了多种移动互联网技术，实现了高效、便捷的考勤管理功能。以下是一些关键技术和我的体会：

(1) NFC（近场通信）技术是一种短距离无线通信技术，适用于身份认证、移动支付和数据交换等场景。在本系统中，利用 NFC 技术实现学生的快速打卡，提高了考勤管理的效率。NFC 技术的集成相对简单，但需要注意设备的兼容性和用户的使用习惯。在实际开发中，通过前台调度系统实现了对 NFC 标签的实时响应，提高了用户体验。

(2) SQLite 是一种轻量级的关系型数据库，适用于移动应用的数据存储。在本系统中，使用 SQLite 存储学生信息和打卡记录，实现了数据的持久化管理。SQLite 使用方便、性能良好，但在处理大量数据时需要注意优化查询和数据存储结构。同时，为了确保数据安全，考虑在未来引入加密和备份机制。

(3) Android 是一个基于 Linux 内核的开源操作系统，广泛应用于移动设备。本系统基于 Android 平台开发，通过 Activity、Intent、Service 等组件实现了各项功能。Android 平台提供了丰富的开发工具和 API，支持快速开发和迭代。但需要注意不同设备和版本的兼容性问题，尤其是在使用 NFC 等特定硬件功能时。

(4) 用户界面是用户与系统交互的桥梁。通过 RecyclerView、EditText、Button 等 UI 组件，构建了简洁友好的用户界面。界面设计不仅要美观，还要注重用户体验和操作的便利性。在实际开发中，采用了 Material Design 规范，使界面更加一致和直观。

(5) 通过 NFC 打卡系统的开发，我深入理解了移动互联网技术的特点和应用场景。移动开发不仅需要扎实的技术基础，还需要关注用户体验和数据安全。未来，随着技术的不断发展，移动互联网将继续改变人们的生活方式，带来更多创新和机遇。