代码审查

总体思路

1）先把项目接入 SonarQube，做一次全量+新增代码扫描；
2）再用 CodeQL 针对 Java 做一次本地或 GitHub Actions 扫描。

一、接入 SonarQube

> 前提：本机或局域网有 SonarQube 服务（Community 版即可），并已创建一个项目和 token。

1. **在 `music-server` 里添加 Sonar 插件
   在 `music-server/pom.xml` 的 `<build><plugins>` 里加一段：

```xml
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.10.0.2594</version>
</plugin>
```

2. **在环境变量中配置 Sonar 参数

```shell
set SONAR_HOST_URL=http://localhost:9000
set SONAR_LOGIN=sqp_f1f92e599e8210134f0e69b0c70014022a868ac9
```

3. **在 `music-server` 目录执行扫描

```shell
cd D:\Work\USERPROG\github\music-website-master\music-server
mvn -DskipTests sonar:sonar
```

4. 在 **SonarQube 界面中过滤 “New Code” 或按文件查看**
   重点关注我们新增或修改的类：

   - `UserActionLog.java`
   - `UserActionLogServiceImpl.java`
   - `IpUtils.java`
   - `ConsumerServiceImpl.java`

- `SongController.java`（`/song/play`）
- `SongServiceImpl.java`（`recordSongPlay`）
- `SongPlayRequest.java`

5. 根据规则修复典型问题：
   - 密码加密使用 **MD5+salt**：安全规则会提示弱哈希算法，可以考虑后续迁移到 BCrypt/Argon2（属于原有逻辑，但可以记到技术债里）。
   - 捕获大量具体异常后统一 `throw new RuntimeException(e)`（`SongServiceImpl` 里 Minio 部分）：Sonar 提示异常处理不够细；可以改为统一包装为自定义业务异常并带日志。
   - 魔法字符串：`"REGISTER"`，`"LOGIN"`，`"PLAY_SONG"` 建议后续收敛成枚举 `UserActionType`。
   - `System.out.println` 调试输出：如果有残留，替换为 Logger。

二、使用 CodeQL 扫描 Java 代码

1. GitHub Actions 方式

1. 仓库启用 CodeQL 安全分析（Security → Code scanning → Set up CodeQL）。
2. 选择 Java 模板，生成 `.github/workflows/codeql.yml`。
3. 提交后，每次 push / PR GitHub 会自动：
   - 编译 `music-server`（需要在 workflow 里 `cd music-server` 执行 `mvn -DskipTests compile`）；
   - 运行 CodeQL 分析并在 “Security → Code scanning alerts” 列出问题。

重点查看对我们新增代码的告警，会有：

- 潜在的 **NullPointer**：如 `ConsumerServiceImpl.loginEmailStatus` 中 `consumer1` 可能为 null（你可加一段 `if (consumer1 == null) return R.error("用户不存在");`）。
- 日志注入/信息泄露：如果将用户输入直接拼接为日志/错误，可考虑做转义或统一格式化。
- **SQL** 使用风险：我们使用 MyBatis-Plus 占位符方式，一般不会有注入问题，但 CodeQL 若提示，按建议确认。

2. 本地 CLI 方式

1. 安装 CodeQL CLI，并下载 Java 数据库包（`codeql-java`）。
2. 在仓库根目录执行：

```shell
codeql database create db-music --language=java --command="mvn -DskipTests compile" --source-root=music-server
```

3. 执行查询：

```shell
codeql database analyze db-music codeql/java-queries:Security\*.ql --format=sarifv2.1.0 --output=codeql-results.sarif
```

4. 使用 VS Code + CodeQL 扩展或 GitHub Code Scanning 导入 `sarif` 查看结果。

**三、优先修复项

把 Sonar / CodeQL 跑起来后，优先关注下面几类问题：

- 安全类
  - 登录/重置密码/用户信息接口的 **空指针 + 参数校验**（例如邮箱找不到用户的场景）；
  - 用户行为日志里是否记录了过多敏感数据（避免明文密码、token 等）。
- 可维护性类
  - `UserActionLog` 相关常量抽成枚举；
  - `recordSongPlay` 等方法中字符串格式、异常处理规范化。
- 资源类
  - 确保所有 IO/网络客户端（目前 MinioClient 已交给 Spring 管理）不会在我们新增逻辑中泄露连接。