|  |
| --- |
| 软件设计文档（SDD）  项目名称：宫商角徵羽（音频调试与编辑工具）  基于SRS版本：第一次修改 |

目录

[1. 系统概述： 4](#_Toc4080)

[1.1. 系统目标： 4](#_Toc24131)

[1.2. 核心功能： 4](#_Toc14659)

[1.3. 技术特点： 4](#_Toc17287)

[2. 总体设计： 4](#_Toc19298)

[2.1. 系统架构： 4](#_Toc5776)

[2.2. 系统组成： 4](#_Toc18619)

[2.2.1. 用户界面模块： 4](#_Toc841)

[2.2.2. 播放控制模块： 5](#_Toc31711)

[2.2.3. 音频管理模块： 5](#_Toc1805)

[2.2.4. 音频处理模块： 5](#_Toc32159)

[2.2.5. 波形可视化模块： 5](#_Toc11459)

[2.2.6. 数据存储与配置模块： 5](#_Toc5438)

[2.2.7. 第三方库适配模块： 6](#_Toc11981)

[2.3. 用户运行环境需求： 6](#_Toc2832)

[2.3.1. 硬件环境： 6](#_Toc21923)

[2.3.2. 软件依赖环境： 6](#_Toc2370)

[2.3.3. 用户权限要求： 7](#_Toc24591)

[2.4. 数据设计： 7](#_Toc17304)

[2.4.1. 数据存储方式： 7](#_Toc27228)

[2.4.2. 数据表结构设计： 7](#_Toc9828)

[2.4.3. 音频文件存储说明： 8](#_Toc13310)

[2.4.4. 数据一致性与更新机制： 8](#_Toc3597)

[3. 功能设计： 8](#_Toc24442)

[3.1. 音频文件加载与播放： 8](#_Toc21459)

[3.2. 音频合并与剪辑： 9](#_Toc26267)

[3.3. 音频效果处理： 9](#_Toc8633)

[3.4. 波形可视化： 9](#_Toc3378)

[3.5. 用户图形界面与多语言支持： 9](#_Toc20154)

[4. 接口设计： 9](#_Toc12023)

[4.1. 外部接口设计： 9](#_Toc19895)

[4.2. 内部接口设计： 10](#_Toc26942)

[4.2.1. UI控制器→播放控制模块： 10](#_Toc19649)

[4.2.2. 播放控制模块→音频处理模块： 10](#_Toc26533)

[4.2.3. 音频管理模块 ↔数据存储模块： 10](#_Toc20384)

[4.2.4. 波形可视化模块→音频数据模块： 10](#_Toc4840)

[4.3. 人机接口设计： 10](#_Toc10198)

[4.3.1. 界面控制设计： 10](#_Toc7023)

[4.3.2. 多语言界面设计： 11](#_Toc32104)

[4.4. 详细接口列表：（见附录C） 11](#_Toc25956)

[4.5. 错误处理： 11](#_Toc30509)

[5. 性能指标设计： 11](#_Toc19016)

[5.1. 音频文件处理性能： 11](#_Toc18050)

[5.2. 音频播放性能： 11](#_Toc1122)

[5.3. 波形可视化性能： 11](#_Toc14725)

[5.4. 系统资源占用： 11](#_Toc31362)

[5.5. 并发处理能力（局部并发）： 11](#_Toc20555)

[6. 其他设计： 12](#_Toc5860)

[6.1. 安全性设计： 12](#_Toc29870)

[6.2. 可靠性设计： 12](#_Toc27201)

[6.3. 可维护性与扩展性设计： 12](#_Toc12709)

[6.4. 兼容性设计： 12](#_Toc9293)

[6.5. 后台支持设计： 12](#_Toc28232)

[附录： 12](#_Toc30000)

[A. 术语表： 12](#_Toc4578)

[B. 需求/设计跟踪矩阵： 13](#_Toc18363)

[C. 详细接口列表： 14](#_Toc25037)

1. **系统概述：**
   1. **系统目标：**

提供一款基于JavaFX的跨平台音频编辑工具，支持音频加载、播放、效果处理、合并剪辑及波形可视化功能，满足SRS中定义的功能与性能需求。

* 1. **核心功能：**
* 支持WAV/MP3/FLAC/OGG格式的音频文件加载与播放
* 实时音效效果处理（音量、音调、回声、混响）
* 多文件合并与时间轴剪辑
* 动态音频波形可视化
* 多语言自适应图形界面
  1. **技术特点：**
* 跨平台：基于Java 11 + JavaFX 17构建
* 高性能：采用TarsosDSP音频处理库优化算法
* 模块化：分层架构设计（用户界面层/逻辑层/数据层）

1. **总体设计：**

**本章节描述系统的总体架构、模块划分与职责界定，确保系统设计具备良好的结构性、可扩展性与维护性**

* 1. **系统架构：**

本系统采用经典的三层架构设计，分为以下：

**用户界面层（JavaFX）**

**↓**

**业务逻辑层（音频控制器、音效处理器）**

**↓**

**数据管理层（文件存储、SQLite数据库、第三方音频库）**

各层之间通过明确的接口进行通信，实现低耦合、高内聚的模块划分，利于功能扩展和后期维护。

* 1. **系统组成：**
     1. **用户界面模块：**

-职责说明：

* 提供可视化操作界面，供用户选择音频文件、设置音效参数、控制播放流程
* 支持中英文界面切换，通过ResourceBundle动态切换语言资源

-实现要点：

* 基于JavaFX实现，采用MVC设计模式
* 使用SceneBuilder构建界面布局，FXML+控制器绑定逻辑
* 响应用户操作并将指令发送至逻辑层
  + 1. **播放控制模块：**

-职责说明：

* 管理播放、暂停、继续播放、停止等功能
* 控制音频数据流的启动与终止，维护当前播放状态

-实现要点：

* 使用Java Sound API的SourceDataLine实现音频输出
* 定义PlayerState状态枚举：STOPPED，PLAYING，PAUSED
* 状态切换驱动界面按钮状态的变化（UI同步）
  + 1. **音频管理模块：**

-职责说明：

* 管理音频文件的加载、保存、合并、剪辑
* 提供文件缓存机制优化性能

-实现要点：

* 支持格式：WAV,MP3,OGG
* 使用JavaFX FileChooser进行文件选择，文件信息存储到SQLite
* 合并剪辑通过操作音频字节流实现，保持原音质
  + 1. **音频处理模块：**

-职责说明：

* 实现音量调节、音调调整、混响、回声等常见音频效果
* 支持实时处理与链式效果结合

-实现要点：

* 基于TarsosDSP处理库实现核心算法
* 每种效果独立封装为单独类（如VolumeEffectProcessor）
* GUI控件变化通过监听器实时更新处理参数
  + 1. **波形可视化模块：**

-职责说明：

* 将音频数据渲染为波形图，支持缩放与选取操作

-实现要点：

* 使用JavaFX Canvas绘制波形图
* 音频样本数据由后端木块抽样并压缩处理
* 波形与播放进度同步显示
  + 1. **数据存储与配置模块：**

-职责说明：

* 存储配置信息、用户偏好（如最近打开文件、语言设置）
* 存储音频元数据（时长、声道、采样率）

-实现要点：

* 使用SQLite存储结构化数据
* 应用启动时加载配置，关闭时自动保存
  + 1. **第三方库适配模块：**

-职责说明：

* 封装第三方库的使用接口，提升代码解耦性

-实现要点：

* JLayer：处理MP3解码
* TarsosDSP：实现音效与音频分析
* 定义统一接口层，例如AudioEffectAdapter封装多种效果实现

以上模块共同构成了系统的核心组成，确保各部分职责清晰，便于团队协作与后期维护

* 1. **用户运行环境需求：**
     1. **硬件环境：**

-最低配置（保证基础功能运行）：

* 操作系统：Windows 8.1 / macOS 10.13+ / Linux（64位内核≥4.15）
* 处理器：双核CPU（Intel i3-5代或AMD同级别）
* 内存：2GB RAM
* 存储：100MB可用空间（不含音频文件存储）
* 显卡：支持OpenGL 2.0的集成显卡（如Intel HD 4000）

-推荐配置：（流畅运行所有功能）：

* 操作系统：Windows 10 / macOS 12+ / Linux（64位内核≥5.4）
* 处理器：四核CPU（Intel i5-8代或AMD Ryzen 5同级别）
* 内存：8GB RAM（支持大文件分块加载与实时音效处理）
* 存储：SSD硬盘，1GB可用空间（提升文件读写速度）
* 显卡：独立显卡（如NVIDIA GTX 1050或AMD RX 560）

-配置设计依据：

* 内存：最低2GB满足基础UI和轻量音频处理（JavaFX内存占用约300MB），推荐8GB应对大文件分块加载（如50MB+音频）和复杂音效计算
* CPU：多核优化音效算法（如混响、回声）的并行计算，推荐配置确保低延迟（≤500ms）
* 图形需求：JavaFX动态波形渲染依赖GPU加速，独立显卡可提升可视化流畅度
* 系统兼容性：Java 11与JavaFX 17要求64位系统，旧版系统（如Win7）可能需额外配置运行时库
  + 1. **软件依赖环境：**
* JRE/JDK：Java SE 11或以上，必须已安装
* JavaFX：JavaFX 17 Runtime，由于图形界面渲染
* 音频驱动：支持PCM输出的音频设备驱动，确保播放功能正常
* 可选依赖：SQLite、TarsosDSP、JLayer，程序已内置，用户无需安装
  + 1. **用户权限要求：**
* **系统需具备文件读写权限（用于音频导入与保存）**
* **无需管理员权限即可安装和运行**
  1. **数据设计：**

本系统采用“结构化+非结构化数据分离”设计思路，保证音频处理的灵活性与系统运行的高效性

* + 1. **数据存储方式：**
* 使用SQLite 3 本地嵌入式数据库保存音频元信息与用户配置，无需用户额外安装数据库服务器
* 音频文件本体以本地文件方式存储，支持WAV、MP3、OGG等格式
* 波形图缓存及处理结果（如剪辑、合并后的中间文件）以临时文件存储在指定路径下
  + 1. **数据表结构设计：**

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 说明 |
| id | INTEGER | 主键，自增 |
| file\_name | TEXT | 文件名称 |
| file\_path | TEXT | 文件路径 |
| time\_long | DOUBLE | 音频时长 |
| track | INTEGER | 声道数 |
| create\_time | DATETIME | 创建时间 |
| modify\_time | DATETIME | 最近修改时间 |

**ER图：**

-AudioFile音频文件表：

* 存储用户加载的音频文件基础元数据，包括文件路径、名称、时长、声道数、采样率以及时间戳
* 所有音频播放与处理操作均基于该表中数据进行管理

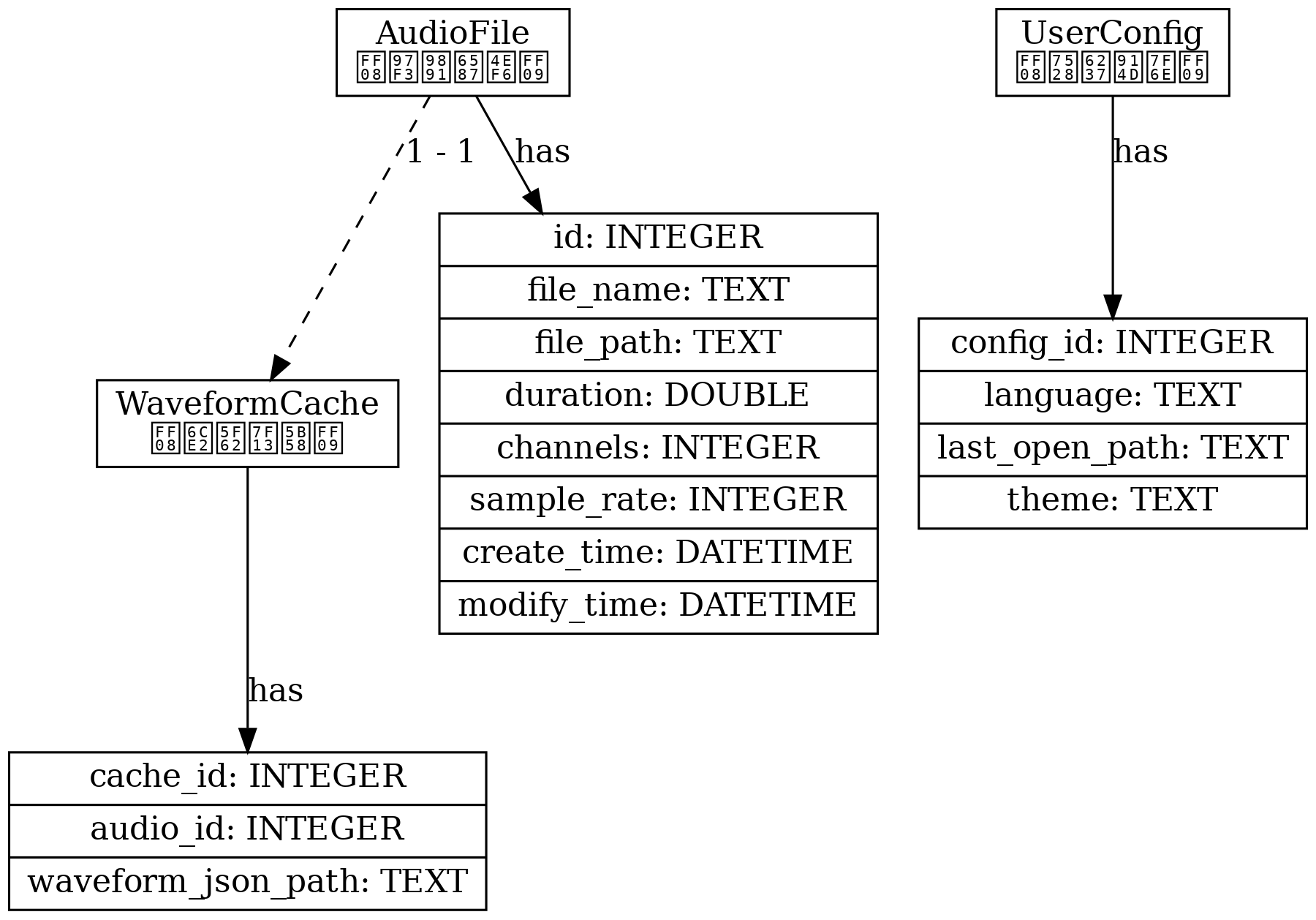
-WaveformCache波形缓存表：

* 记录与音频对应的波形数据缓存文件路径，优化波形图渲染性能
* 采用一对一关联方式，即每个音频文件对应一个波形缓存项

-UserConfig用户配置表：

* 存储用户偏好设置，如语言、最近打开路径等，提升系统的可用性与个性化体验

各数据表使用SQLite数据库进行管理，具备轻量、可移植、无需安装等优点，复合桌面应用的部署要求。该设计实现了结构化数据与非结构化文件（音频数据本体）的分离，便于后续扩展功能模块如收藏管理、标签分类等

****

* + 1. **音频文件存储说明：**
* 原始上传文件保存在‘/audio\_data/’路径
* 剪辑、合并等操作生成的新文件保存在‘/audio\_output’路径
* 若启用波形缓存功能，将生成‘.json’文件保存采样点数据，加速可视化处理
  + 1. **数据一致性与更新机制：**
* 每次加载/处理文件后将更新数据库对应记录
* 删除音频文件时同步删除数据库记录和本地文件
* 应用关闭时保存用户最近操作记录与界面配置

1. **功能设计：**

**本章节基于需求规格，详细说明各功能模块的设计思路、实现方式及模块间的交互关系，保证功能实现具有一致性和完整性**

* 1. **音频文件加载与播放（对应模块：2.2.2.音频播放模块、2.2.3.音频管理模块）：**

**-功能描述：**

* 用户可通过界面加载本地音频文件并进行播放，支持播放、暂停、停止控制

**-实现方法：**

* 使用JavaFX 的 FileChooser 实现文件选择，支持 WAV、MP3、OGG 格式
* 文件信息（路径、时长、声道数）存入 SQLite 数据库
* 播放功能使用 SourceDataLine 实现音频输出，配合状态机控制播放行为
  1. **音频合并与剪辑（对应模块：2.2.3.音频管理模块）：**

**-功能描述：**

* 支持将多个音频文件拼接为一个完整音频，也可以从一个音频中剪辑出片段

**-实现方法：**

* 合并功能读取多个音频文件流并顺序写入新文件
* 剪辑功能根据用户设定的时间范围提取子音频流并输出为新文件
* 所有新生成的文件信息同步写入数据库
  1. **音频效果处理（对应模块：2.2.4.音频处理模块）：**

**-功能描述：**

* 用户可对当前播放音频应用音量、音调、混响、回声等常见音效，并实时预览效果

**-实现方法：**

* 每种效果封装为一个 AudioEffectProcessor 类，继承统一接口
* 使用 TarsosDSP 实现实时音频流处理
* GUI 控件（滑块、选择框）绑定效果参数，支持链式组合效果
  1. **波形可视化（对应模块：2.2.5.波形可视化模块）：**

**-功能描述：**

* 实时显示音频播放的波形图，帮助用户直观了解音频结构

**-实现方法：**

* 使用 JavaFX Canvas 绘制波形
* 后端模块对音频流进行采样、归一化与压缩处理
* 播放进度同步更新画布绘图区域
  1. **用户图形界面与多语言支持（对应模块：2.2.1.用户界面模块）：**

**-功能描述：**

* 提供直观、统一的界面操作入口，支持中文与英文界面切换

**-实现方法：**

* 局域JavaFX FXML定义UI结构，使用ResourceBundle管理多语言资源
* 控制器根据用户偏好切换文本显示
* 所有功能入口（如按钮、菜单）由UI控制层统一管理

1. **接口设计：**

**本章节规范系统对外接口、内部模块接口与用户交互接口的设计标准，确保模块间通信清晰、前后端协作顺畅**

* 1. **外部接口设计：**
     1. **硬件接口：**
* 音频输出设备：通过操作系统调用声卡接口，使用SourceDataLine 播放 PCM 流数据
* 显卡接口：JavaFX渲染波形图，需显卡支持OpenGL2.0及以上标准
  + 1. **软件接口：**
* Java Sound API：负责音频流控制播放
* JavaFX：由于图形用户界面构建与渲染
* TarsosDSP：实现音效处理（如回声、混响、音量调节）
* JLayer：实现MP3文件解码功能
  + 1. **数据库接口：**
* SQLite 3 本地数据库：用于存储音频文件元数据与用户配置信息，标准SQL语句读写操作
  + 1. **通信接口：**
* 本项目为本地应用，前端通过HTTP本地请求与后端通信（如/api/audio/play），采用RESTful API规范，无需外部服务器支持
  1. **内部接口设计：**
     1. **UI控制器→播放控制模块：**
* 调用方法：startPlayback(audioId)、pausePlayback()、resumePlayback()、stopPlayback()
  + 1. **播放控制模块→音频处理模块：**
* 调用方法：applyEffect(effectType, params)（应用音量、音调、回声等效果）
  + 1. **音频管理模块 ↔数据存储模块：**
* 加载音频时读取数据库，保存操作时更新数据库记录
  + 1. **波形可视化模块→音频数据模块：**
* 调用采样接口，获取简化版波形数据用于绘图
  1. **人机接口设计：**
     1. **界面控制设计：**
* 文件选择器：用于加载本地音频文件
* 播放控制区：播放、暂停、继续、停止按钮
* 效果调整区：滑块调整音量、音调参数
* 波形显示区：绘制音频波形，支持缩放与拖动
* 状态提示区：实时显示播放状态与异常提示
  + 1. **多语言界面设计：**
* 支持中文、英文动态切换，使用 ResourceBundle 实现界面文字本地化管理
  1. **详细接口列表：（见附录C）**
  2. **错误处理：**

|  |  |
| --- | --- |
| 错误码 | 含义 |
| 400 | 请求参数错误 |
| 404 | 资源未找到（如音频ID无效） |
| 500 | 服务器内部错误 |

**错误返回示例：**

**{**

**“code”: 400,**

**“message”: “音频文件ID无效”**

**}**

1. **性能指标设计：**

**本章节定义系统运行时需满足的性能指标，确保用户体验的流畅性与稳定性**

* 1. **音频文件处理性能：**
* 加载速度：加载一个100MB的音频文件，时间不超过5s
* 剪辑处理：剪辑操作响应时间不超过2s
* 合并处理：合并操作在300MB以内文件时，处理时间不超过8s
  1. **音频播放性能：**
* 播放延迟：从用户点击播放到实际音频开始输出的延迟不超过500ms
* 播放稳定性：播放过程中不应出现音频卡顿或中断，允许误差小于1%
  1. **波形可视化性能：**
* 波形渲染速度：在加载完成后的1s内完成波形初次绘制
* 波形交互响应：缩放、拖动波形视图时响应延迟小于100ms
  1. **系统资源占用：**
* CPU占用率：播放或编辑音频时，CPU占用率不高于70%
* 内存占用：系统总内存占用不超过500MB（包括缓存和波形数据）
  1. **并发处理能力（局部并发）：**
* 支持同时打开并管理最多5个音频文件进行基本操作（如播放、剪辑、效果预览），互不影响

1. **其他设计：**

**本章节描述系统在安全性、可靠性、可维护性、扩展性、兼容性方面的设计考量**

* 1. **安全性设计：**
* 文件类型校验：只允许导入WAV、MP3、OGG格式的音频文件，防止恶意文件攻击
* 大小限制：上传音频文件最大限制为500MB，防止异常超大音频导致系统崩溃
* 异常处理：所有模块均配置异常捕获与日志记录，保证即使发生错误也能正常降级
* 资源释放：播放、处理完成后及时释放文件流、线程资源，避免资源泄露
  1. **可靠性设计：**
* 错误提示机制：统一的错误提示弹窗，详细说明问题原因及解决办法
* 断点续处理：对于播放、剪辑、合并等操作，若中断可允许用户重新发起，自动重置状态
* 测试覆盖：关键模块（如播放控制器、音效处理器）均覆盖单元测试，覆盖率不低于80%
  1. **可维护性与扩展性设计：**
* 模块化结构：系统按照功能划分为独立模块，便于后续功能扩展（如新增音效种类、导出格式）
* 接口标准化：所有模块之间通过定义清晰的接口进行交互，减少模块之间的耦合
* 插件设计预留：音效处理部分采用接口抽象，可支持未来第三方音效插件接入
  1. **兼容性设计：**
* 跨平台支持：Windows（8.1及以上）、macOS（10.13及以上）、主流Linux桌面系统
* 显示适配：界面自适应常见分辨率（1366x768及以上），高DPI屏幕友好
* JDK版本兼容：支持Java SE 11至Java SE 21
  1. **后台支持设计：**
* 本地后台线程：音频播放、波形渲染、文件加载均在单独线程中运行，提升界面响应性
* 后台缓存机制：大型音频文件按需分片加载，避免一次性占用过多内存
* 后台数据同步：音频文件元数据、用户配置在后台自动保存到SQLite数据库

**附录：**

1. **术语表：**

|  |  |
| --- | --- |
| **术语** | **含义** |
| PCM | Pulse-Code Modulation，脉冲编码调制，音频数据编码格式之一 |
| WAV | 无损音频文件格式，常用于高质量音频存储 |
| MP3 | 有损压缩音频格式，广泛应用于流媒体播放 |
| OGG | 开源自由音频压缩格式，支持高效压缩与高质量播放 |
| FFT | Fast Fourier Transform，快速傅里叶变换，常用于音频频谱分析 |
| UI | User Interface，用户界面 |
| API | Application Programming Interface，应用程序接口 |
| SQLite | 轻量级嵌入式数据库，用于本地数据存储 |
| JavaFX | 用于构建桌面应用程序GUI的Java图形框架 |
| TarsosDSP | 音频信号处理库，支持实时音效处理 |

1. **需求/设计跟踪矩阵：**

**本章节建立《宫商角徵羽软件需求规格说明书》（SRS）与《宫商角徵羽软件设计说明书》(SDD)之间的对应关系，实现需求到设计的正向与反向跟踪，确保系统设计完整覆盖需求**

|  |  |  |
| --- | --- | --- |
| **正向跟踪（SRS→SDD）** | | |
| **SRS需求编号** | **需求描述** | **对应设计模块** |
| SRS-0010 | 支持加载和播放音频文件 | 3.1 音频文件加载与播放；2.5 音频管理模块、2.3 播放控制模块 |
| SRS-0020 | 提供音频效果处理功能 | 3.2 音频效果处理；2.4 音频处理模块 |
| SRS-0030 | 支持音频合并与剪辑 | 3.3 音频合并与剪辑；2.5 音频管理模块 |
| SRS-0040 | 提供音频波形可视化功能 | 3.4 波形可视化；2.6 波形可视化模块 |
| SRS-0050 | 提供友好的图形用户界面 | 3.5 用户界面与多语言支持；2.3 用户界面模块 |

|  |  |  |
| --- | --- | --- |
| **反向追踪（SDD→SRS）** | | |
| **设计模块** | **设计描述** | **对应需求编号** |
| 2.3 用户界面模块 | 提供图形界面与多语言切换功能 | SRS-0050 |
| 2.4 音频处理模块 | 实现音效处理功能，如音量调节、音调调整 | SRS-0020 |
| 2.5 音频管理模块 | 支持音频文件的加载、保存、剪辑、合并 | SRS-0010、SRS-0030 |
| 2.6 波形可视化模块 | 绘制音频波形并同步播放进度 | SRS-0040 |
| 2.7 数据存储与配置模块 | 存储音频元数据与用户偏好设置 | 支持整体功能需求（辅助模块） |

1. **详细接口列表：**

|  |  |
| --- | --- |
| **音频文件管理接口** | |
| 接口名称 | 加载音频文件 |
| 接口地址 | /api/audio/load |
| 请求方法 | POST |
| Content-Type | Multipart/form-data |
| 请求参数 | File（音频文件，必填） |
| 返回示例 | {“code”:200, “message”:”加载成功”, “data”:{“audiold”:”123456”}} |
| 错误示例 | {“code”:400, “message”:”文件格式不支持” |
| 接口名称 | 获取音频文件列表 |
| 接口地址 | /api/audio/list |
| 请求方法 | GET |
| 返回示例 | {“code”:200, “message”:”获得成功”, “data”:[...]} |
| 错误示例 | {“code”:500, “message”:”服务器内部错误”} |
| **音频播放控制接口** | |
| 接口名称 | 播放音频 |
| 接口地址 | /api/audio/play |
| 请求方法 | POST |
| Content-Type | Application/json |
| 请求参数 | Audiold（String，必填） |
| 返回示例 | { "code": 200, "message": "播放成功" } |
| 错误示例 | { "code": 404, "message": "音频文件未找到" } |
| **音频效果处理接口** | |
| 接口名称 | 调整音量 |
| 接口地址 | /api/audio/effect/volume |
| 请求方法 | POST |
| 请求参数 | audioId（String，必填），volume（float，范围0.0-1.0） |
| **音频合并与剪辑接口** | |
| 接口名称 | 合并音频文件 |
| 接口地址 | /api/audio/merge |
| 请求方法 | POST |
| 请求参数 | audioIds（String数组，必填） |
| 接口名称 | 剪辑音频文件 |
| 接口地址 | /api/audio/clip |
| 请求方法 | POST |
| 请求参数 | audioId（String，必填），startTime（秒），endTime（秒） |
| **音频波形可视化接口** | |
| 接口名称 | 获取音频波形数据 |
| 接口地址 | /api/audio/waveform |
| 请求方法 | GET |
| 请求参数 | audioId（String，必填） |