**数字音乐合成器项目需求文档**

**1 引言**

**本需求文档旨在明确数字音乐合成器软件的功能、性能及接口需求，为项目的开发、测试和维护提供依据。该系统主要面向音乐创作初学者和音效设计爱好者，提供一个直观、可交互的音频合成环境。用户能够通过本软件自由编辑音色、实时试听音效，实现基础音乐创作和声音探索。**

**2 总体描述**

**2.1 产品概述**

**数字音乐合成器是一款基于Web的加/减法音频合成器软件，集成了加法合成与减法合成功能。支持波形生成、谐波控制、ADSR包络编辑、滤波器调节、多音轨组合、LFO调制等先进音频合成技术。具备用户友好图形界面，支持实时波形可视化及多音轨混音，适合音乐教育、声音实验及轻量级音乐创作。**

**2.2 用户特点**

**基础音乐创作者：使用预设或自定义音色进行简单音乐创作。**

**声音实验爱好者：通过调节参数探索多样音效。**

**学习者：了解音频合成原理和技术。**

**目标用户为具备一定音乐基础或编程背景的学生、声音研究者及独立音乐人，界面简洁直观，功能配置灵活，支持参数实时调整。**

**2.3 运行环境**

**客户端：现代浏览器（Chrome、Firefox、Edge、Safari）**

**服务器端：Python 3.9+，使用FastAPI或Flask框架，部署于支持WSGI的服务器**

**音频处理库：NumPy、SciPy进行波形计算，前端使用Web Audio API进行音频渲染**

**2.4 设计约束**

**支持至少3种振荡器波形类型（正弦波、方波、锯齿波）**

**参数调整实时更新音频输出**

**模块化架构，方便后续拓展**

**3 功能需求**

**3.1 音色生成与编辑**

**多种振荡器波形生成：支持预设音色（钢琴、小提琴等），支持自定义加法合成通过叠加多个正弦波创建复杂音色。**

**谐波控制：调整各谐波振幅比例，动态增减谐波数量，模拟不同乐器音色特性。**

**滤波器调节（减法合成）：提供低通、高通、带通等滤波器，实时调节截止频率和共振。**

**ADSR包络编辑：图形化控制音量的起音、衰减、持续、释音阶段，模拟真实乐器动态响应。**

**LFO调制：低频振荡器调制音高、音量、滤波器频率，产生颤音、震音、哇音效果，支持调节频率、振幅和波形。**

**3.2 音轨管理与混音**

**多音轨组合：分层叠加多个音色，丰富声音层次，提供独立音轨控制。**

**音轨编辑：独立调整音轨音量和声像（Panning），支持分轨播放控制。**

**3.3 实时反馈与导出**

**实时波形可视化：动态显示当前和多音轨叠加音频波形。**

**音频播放：支持实时播放当前项目或单个音轨，支持播放、暂停、循环控制。**

**波形文件导出：支持导出为WAV文件，方便外部使用和分享。**

**4 非功能需求**

**4.1 性能需求**

**实时响应：参数调整应即时反映音效变化，响应延迟不超过100毫秒。**

**低延迟音频播放，保证流畅体验。**

**合理资源利用，避免高CPU和内存占用。**

**4.2 可用性需求**

**直观简洁的图形用户界面，降低学习成本。**

**关键功能易访问，支持拖拽、滑块等交互方式。**

**提供清晰错误提示和操作指导。**

**4.3 兼容性需求**

**浏览器兼容性：支持Chrome、Firefox、Edge、Safari等主流现代浏览器。**

**操作系统兼容性：支持Windows、macOS、Linux等平台。**

**4.4 可维护性与扩展性**

**模块化设计，代码结构清晰，便于维护与拓展。**

**清晰接口规范，方便模块间协作。**

**完善注释和文档，便于团队成员理解。**

**易于单元测试和集成测试。**

**4.5 安全性需求**

**确保用户音色和项目数据保存及加载过程数据完整无损。**

**若涉及账户管理，确保数据隐私和访问控制安全。**

**5 外部接口需求**

**5.1 用户接口（UI）**

**音色编辑区：显示与调节各振荡器参数**

**轨道管理区：添加/删除/复制音轨**

**可视化窗口：显示实时音频波形**

**全局控制区：统一播放/暂停/导出**

**5.2 软件接口**

**前后端通过 HTTP 接口通信**

**所有接口返回 JSON 格式，含状态码与提示信息**

**示例接口：**

**POST /synthesize：发送音轨参数，返回合成音频**

**GET /preset：获取预设音色包络配置**

**5.3 硬件接口**

**无特殊硬件要求，使用浏览器默认音频输出通道**

**6 技术选型**

**项目类型：基于Web应用**

**后端开发语言：Python**

**Web框架：FastAPI 或 Flask**

**音频处理库：NumPy、SciPy**

**前端音频渲染：Web Audio API**

**版本管理：Git + GitHub**

**7 开发方法与流程**

**采用敏捷开发方法，Sprint周期两周。**

**定期Sprint会议，沟通进展和问题。**

**Sprint包含需求确认、设计实现、测试反馈、演示发布。**

**使用GitHub Project看板进行任务管理和进度跟踪。**

**每个Sprint交付可运行增量功能。**

**8 面向对象思想实践**

**8.1 面向对象分析（OOA）**

**识别核心对象：振荡器（Oscillator）、包络生成器（Envelope）、滤波器（Filter）、音轨（Track）、低频振荡器（LFO）等。**

**建立对象关系：音轨聚合多个振荡器，包络控制振荡器振幅等。**

**定义对象行为：振荡器生成波形，滤波器处理频率等。**

**8.2 面向对象设计（OOD）**

**分层架构：表现层（UI）、业务逻辑层（合成引擎）、数据层（音频缓存）。**

**设计模式：工厂模式创建不同波形振荡器，观察者模式实现参数实时更新。**

**定义类接口：Synthesizer接口包含generate()、apply\_filter()等方法。**

**8.3 面向对象编程（OOP）**

**使用Python类实现关键组件，如ADSREnvelope、LowPassFilter。**

**继承体系：基础振荡器类派生出SineOscillator、SquareOscillator等。**

**8.4 面向对象测试（OOT）**

**单元测试验证核心类（验证振荡器波形生成、波形时间转化等）的功能正确性。**

**模拟对象测试：使用Mock滤波器验证合成器交互**

**继承关系测试：验证不同振荡器子类(正弦波、方波等)的多态行为**

**9 预期成果**

**功能完善的数字音乐加/减法合成器Web应用**

**完整GitHub代码仓库，含源代码、详细说明文档和使用指南**

**项目团队报告，包含分工、实现细节、挑战及解决方案**

**项目演示视频，展示软件功能和使用流程**

**10 术语表**

**ADSR：Attack（起音）、Decay（衰减）、Sustain（持续）、Release（释音）**

**LFO：Low Frequency Oscillator，低频振荡器，用于调制音高、音量等参数**

**加法合成：叠加多个正弦波创建复杂音色的方法**

**减法合成：通过滤波器削减或衰减频率成分塑造音色的方法**

**谐波：基频整数倍频率，决定音色特性**

**声像（Panning）：声音在立体声场中的左右位置分布**

**Sprint：敏捷开发周期，通常为1-4周**