系统建模报告

项目名称：数字音乐合成器系统

目 录

一、需求规格说明 2

1系统概述 2

1.1编写目的 2

1.2背景 2

1.3目的 3

1.4 产品描述 3

2对功能的基本描述 4

2.1前台 4

2.2后台 4

2.3顾客模块 4

3系统功能需求分析 4

3.1功能性需求 4

3.2非功能性需求 5

4用例模型 6

4.1参与者 6

4.2 用例 7

4.3用例图 7

二、类图 8

1类 8

2 UML类图 13

三、顺序图 16

四、构件图 16

5

# 一、需求规格说明

## 1系统概述

## 1.1编写目的

本需求文档的目的是详细描述数字音乐合成器网页系统的功能需求、设计目标和技术实现路径，为开发团队提供明确的开发指导和依据。

## 1.2背景

随着音乐创作逐步走向数字化，传统的合成器和音乐创作软件面临较高的学习门槛和硬件要求。本系统旨在提供一个在线的数字音乐合成器工具，使得用户无需安装额外的硬件和软件即可进行音乐创作和音效设计，降低使用门槛，提升创作效率。

## 1.3目标

本系统的目的是为用户提供一个实时音频合成和处理的平台，支持音轨编辑、音效设计、节奏控制等功能，旨在满足创作音乐和音效的需求。

## 1.4产品描述

该系统包含以下主要功能模块：

* **全局控制模块（顶部控制区域）**：用户可以通过控制bpm和拍号来调整整体音乐的节奏感，也可以剪切音块，实时播放音轨和音效，完成作品后导出音频文件。
* **音轨编辑模块（左侧控制区域）**：支持多轨音频合成，用户可以自由添加、删除、编辑音轨。
* **音块编辑模块（中心控制区域）**：支持多音块合成，用户可以自由添加、删除、编辑音块。
* **音符设计模块（隐藏控制区域）**：提供音调、音量、节拍数设置功能。

# 2对功能的基本描述

2.1 **全局控制模块**

全局控制模块允许用户调节整体音乐的节奏和拍号。用户可以实时调整 **BPM**（每分钟节拍数）和 **拍号**，快速适应不同风格的创作需求。此外，用户可以拖动鼠标改变播放位置，剪切音块。用户还可以通过播放按钮启动或暂停音乐创作，实时监听音轨和音效的合成效果。完成作品后，系统支持将音乐导出为常见音频格式WAV，方便分享和保存。

2.2 **音轨编辑模块**

音轨编辑模块是系统的核心功能之一，支持多轨音频的合成与编辑。用户可以自由添加、删除、编辑音轨，进行多轨混音操作。每个音轨都可以独立调节音量、静音、独奏等参数，帮助用户精确控制每个音轨的表现。

2.3 **音块编辑模块**

音块编辑模块允许用户通过音块的组合与编辑来创建音乐片段。音块是基于音符设计的模块化单位，用户可以自由创建、删除、复制和调整音块的时长、音量等参数。

**2.4** 音符设计模块

音符设计模块提供音调、音量和节拍数选择。用户可以在此模块内设计和编辑音符的具体细节并将它添加到音块中，如音符的音高、音量、节拍数等。这个模块的设计使得创作过程更加灵活和自由。

# 3系统功能需求分析

## 3.1功能性需求

# **（1）全局控制模块**

全局控制模块是系统的核心控制单元，允许用户调节整体音乐的节奏和拍号。通过该模块，用户可以实时调整BPM（每分钟节拍数）和拍号，适应不同风格的音乐创作需求。同时，用户可以拖动时间轴控制播放位置，剪切音轨，并通过播放按钮启动或暂停音乐创作。完成创作后，系统支持将音乐导出为常见音频格式（如WAV）进行保存或分享。

* **导出音频**：当创作完成后，用户可以将作品导出为WAV或MP3等常见格式，方便保存或分享。
* **播放位置控制**：通过拖动播放头或点击时间轴，用户可以精确控制播放位置，支持跳转到小节或音符级别。
* **音块创建功能：**用**户可以在本音轨没有音块的部分绘制音块。**
* **音块剪切功能**：用户可以在现时间段支持剪切操作。
* **实时控制BPM**：提供一个滑块或输入框，允许用户调整BPM值，通常范围为60-200 BPM。每次调整都会实时影响到音轨播放节奏。
* **实时控制拍号**：用户可以通过下拉菜单或输入框调整拍号，支持多种常见拍号（如4/4、3/4、6/8等）。
* **播放控制**：包含回到开头、播放、停止、循环播放等操作，用户可以控制音乐的播放状态。

## 3.2非功能性需求

# **（1）性能需求**

* **响应时间**：
  + 音频播放时，用户点击播放按钮后，音频在100毫秒内开始播放。
  + 用户点击添加或编辑音轨时，界面在200毫秒内完成响应。
  + 在音轨编辑和音效调整时，系统保证低延迟，确保音频实时反映用户的操作（如调整音量、切换音效）。
* **吞吐量**：
  + 系统能够同时处理100条音轨，每条音轨至少包含10分钟的音频数据，并在编辑时保持流畅。
* **音频渲染速度**：
  + 在导出音频时，系统以至少1:1的速率完成音频的渲染和导出（即导出10分钟的音频在10分钟内完成）。

### **（2） 可用性需求**

* **系统可用性**：
  + 系统达到99.9%的可用性（每年不可用时间不超过8.76小时），保证大部分时间内用户能够无障碍使用平台进行创作。
* **故障恢复能力**：
  + 系统出现故障时能够在3秒内恢复，并向用户提供清晰的错误提示或恢复选项。
* **容错性**：

在音轨编辑过程中出现轻微错误或资源冲突时，系统通过自动保存或版本控制功能避免丢失用户操作，保证创作的连续性。

### **（3） 可维护性需求**

* **代码可读性与结构**：
  + 代码遵循模块化设计，确保每个功能模块（如音轨编辑、音效处理、导出功能）相互独立，便于未来的修改与扩展。
  + 注释简洁明了，并有详细的文档支持，开发者可以快速理解系统架构和功能。

### **（4） 兼容性需求**

* **平台兼容性**：
  + 系统支持主流操作系统（Windows、macOS、Linux）的桌面浏览器，同时支持iOS和Android的移动端浏览器。
* **浏览器兼容性**：
  + 系统支持主流浏览器（Chrome、Firefox、Safari、Edge），确保不同浏览器用户获得一致的使用体验。

### **（5） 资源消耗需求**

* **内存使用**：
  + 音轨和音效的加载、编辑减少内存消耗，确保系统在处理大量音轨或音频素材时不出现卡顿。
* **CPU 使用**：
  + 音频处理（如实时播放、效果渲染）优化系统资源使用，避免过度占用CPU，影响其他操作。

#### 4用例模型

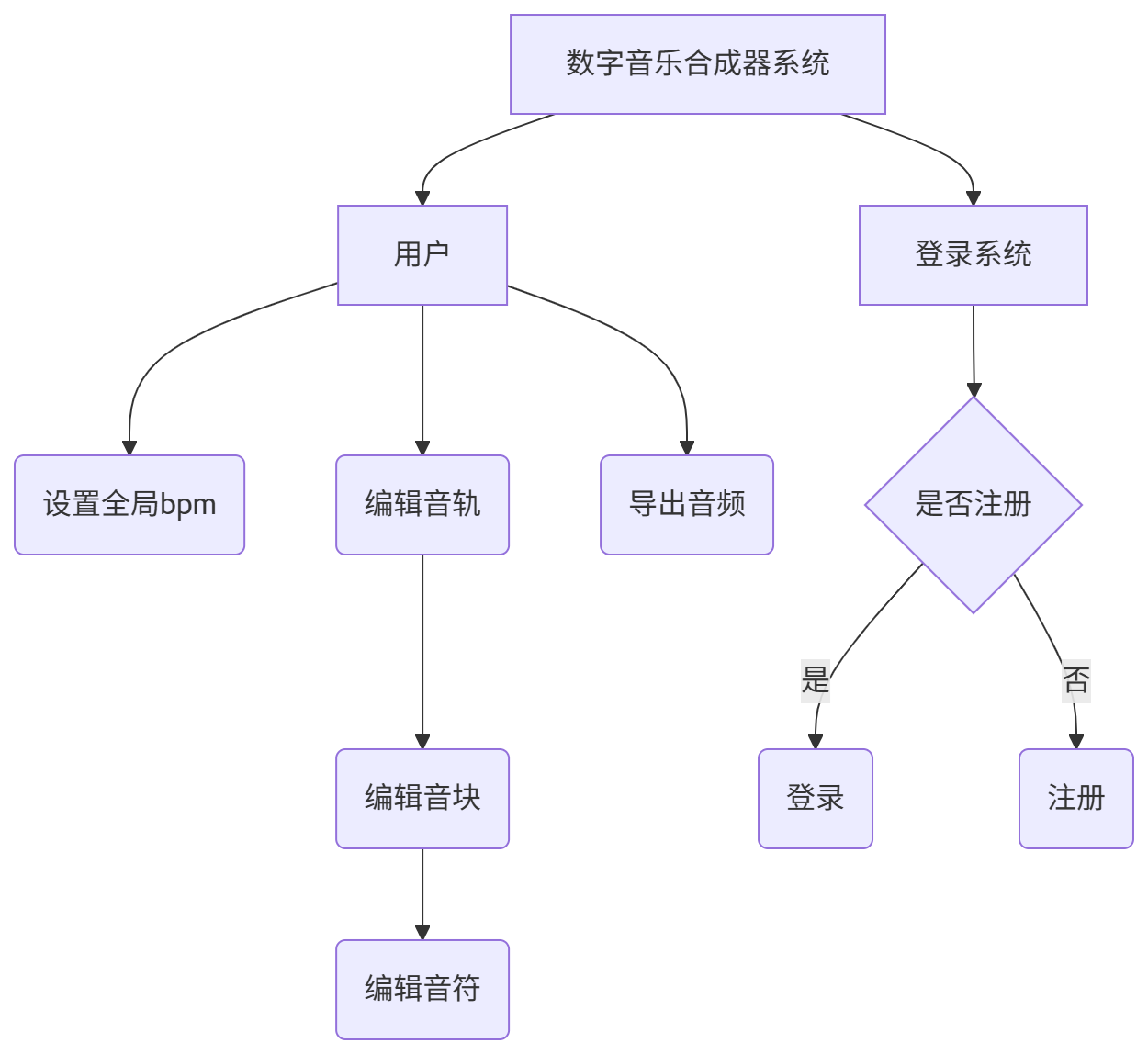
## 4.1 参与者

* **用户**：可以编辑音轨、音块，编辑音符等。
* **登录系统**：用于完成账号登录。

## 4.2 用例

* **用户**：
  + 设置全局bpm
  + 浏览、编辑音轨
  + 浏览、编辑音块
  + 浏览、编辑音符
  + 导出音频
* **登录系统**：
  + 注册账号
  + 登录账号

## 4.3用例图



# 二、类图

#### 1类

#### (1) 类：UserInterface（用户界面）

#### 方法：

* **display\_track\_list(self)**  
  显示音轨列表。
* **display\_note\_grid(self)**  
  显示音符网格。
* **display\_playback\_controls(self)**  
  显示播放控制面板。
* **display\_file\_operations(self)**  
  显示文件操作界面。
* **handle\_user\_input(self)**  
  处理用户输入，监听并响应用户操作。

**(2) 类DigitalSynthesizer（数字音乐合成器）**

#### 属性：

* **tracks** (List[Track]): 存储音轨对象的列表。
* **waveform** (np.ndarray): 当前生成的音频波形。
* **bpm** (int): 每分钟节拍数，影响音轨的节奏。
* **sample\_rate** (int): 采样率（Hz），控制音频的质量。
* **audio\_engine** (None): 目前未使用的音频引擎，可能用于音频播放等功能。
* **volume** (float): 合成器音量控制，取值范围 0.0 到 1.0。

#### 方法：

* **\_\_init\_\_(self, bpm: int = 120, sample\_rate: int = 44100, volume: float = 1.0)**  
  初始化合成器的基本参数，包括节拍、采样率和音量。
* **add\_track\_by\_property(self, timbre: str, pitch\_range: str, bpm: int, sample\_rate: int, volume: float) -> Track**  
  根据属性创建并添加一条音轨，返回新创建的音轨实例。
* **add\_track(self, track: Track) -> Track**  
  添加已有的音轨对象，返回添加后的音轨。
* **remove\_track(self, track\_id: int) -> bool**  
  根据音轨 ID 删除音轨，删除成功返回 True，否则返回 False。
* **set\_bpm(self, bpm: int) -> None**  
  设置合成器的节拍速度。
* **generate\_waveform(self) -> np.ndarray**  
  根据所有音轨生成合成波形，返回合成后的音频数据。
* **play\_for\_preview(self) -> None**  
  播放生成的音频波形进行预览。
* **save\_to\_file(self, filename: str) -> None**  
  将当前音频波形保存为 WAV 格式的文件。
* **open\_file(self, filename: str) -> bool**  
  打开指定的音频文件（未实现）。

#### 辅助函数：

* **play\_instance()**  
  创建一个 DigitalSynthesizer 实例，添加多个音轨并播放，作为示例。

#### (3) 类AudioEngine（音频引擎）

#### 属性：

* **sample\_rate** (int): 音频采样率，默认为 44100 Hz。

#### 方法：

* **\_\_init\_\_(self)**  
  构造函数，初始化采样率为 44100。
* **play\_audio(self, data: np.ndarray) -> None**  
  播放传入的音频数据。该方法目前没有实现具体的播放逻辑。
* **stop\_playback(self) -> None**  
  停止音频播放。该方法目前没有实现具体的停止逻辑。

### **(4) 类Track （音轨类）**

#### 属性：

* **bpm** (int): 每分钟节拍数 (Beats Per Minute)，控制音轨的节奏。
* **sample\_rate** (int): 音频采样率，单位为 Hz，默认 44100。
* **timbre** (str): 音轨的音色类型，如 "piano", "violin" 等。
* **track\_id** (int): 音轨的唯一标识符。
* **track\_name** (str): 音轨名称，根据 track\_id 动态生成。
* **pitch\_range** (str): 音轨的音高范围（如 "C3-B5"）。
* **volume** (float): 音轨的音量（范围从 0 到 1）。
* **lowest\_note** (Note | None): 音轨中的最低音符。
* **highest\_note** (Note | None): 音轨中的最高音符。
* **note\_blocks** (list): 存储音轨中的 NoteBlock 对象的列表。
* **waveform** (np.ndarray | None): 音轨最终的波形。

#### 方法：

* **\_\_init\_\_(self, timbre, pitch\_range, bpm, sample\_rate, volume, track\_id)**  
  构造函数，用于初始化音轨对象的各项属性。
* **add\_note\_block(self, note\_names, beat\_times, volume=None, start\_beat=None, block\_id=-1) -> bool**  
  向音轨中添加一个新的 NoteBlock。音符可以有名称、节拍、音量、起始节拍等参数。如果 volume 或 start\_beat 为 None，则使用默认值。
* **remove\_note\_block(self, block\_id) -> bool**  
  删除指定编号的音块。删除成功返回 True，失败则返回 False。
* **generate\_waveform(self) -> np.ndarray**  
  生成音轨的合成波形，将每个音块的波形拼接起来并调整音量。
* **calculate\_pitch\_range(self) -> str**  
  计算并返回音轨的音高范围。若音轨没有音符，则返回 "No notes in track"。
* **change\_timbre(self, target\_timbre) -> bool**  
  更改音轨的音色。若成功更改音色，返回 True，否则返回 False。

#### 测试函数：play\_little\_star()

* 该函数临时测试一个简单的旋律“**小星星**”。创建一个音轨并按顺序添加音符，最后生成音轨波形并播放。

### **(5) 类NoteBlock （音块类）**

#### 属性：

* **bpm** (int): 每分钟节拍数 (Beats Per Minute)，控制音块的节奏。
* **sample\_rate** (int): 音频采样率，单位为 Hz，默认 44100。
* **timbre** (str): 音块的音色类型，如 "piano"、"violin" 等。
* **block\_id** (int): 音块的唯一标识符。
* **note\_names** (list[str]): 存储音符名称的列表，例如 ["C4", "E4", "G4"]。
* **beat\_times** (list[float]): 每个音符持续的节拍数。
* **start\_beat** (list[float]): 每个音符的起始节拍位置。
* **volume** (list[float]): 每个音符的音量（0.0 ~ 1.0）。
* **notes** (list[Note]): 存储音符对象 Note 的列表。
* **waveform** (np.ndarray | None): 该音块的最终音频波形数据。

#### 方法：

* **\_\_init\_\_(self, timbre, bpm, sample\_rate, note\_names=None, beat\_times=None, start\_beat=None, volume=None, block\_id=0)**  
  构造函数，用于初始化音块对象。根据传入的音符名称、节拍、音量等信息创建多个 Note 对象并存储在 notes 中。
* **generate\_waveform(self) -> np.ndarray**  
  生成音块的合成音频波形。会根据每个音符的起始节拍、持续时间和音量生成音频波形，并将多个音符的波形合成在一起。
* **show\_time\_domain(self)**  
  绘制音块的时域波形图，展示音频的幅度与时间的关系。
* **add\_note(self, note\_name, beat\_time, volume, start\_beat=0) -> bool**  
  向音块中添加一个音符。根据提供的音符名称、节拍、音量和起始节拍等参数，创建 Note 对象并将其添加到音块中。
* **remove\_note(self, note\_id) -> bool**  
  删除指定编号的音符，并更新音符列表及音符的编号。

#### 测试函数：\_\_main\_\_

* **play\_little\_star()**:  
  该函数创建一个音块并向其添加一系列音符，最终合成波形并绘制波形图。此外，还会使用 sounddevice 播放音轨生成的音频。

### **(6) 类Note（音符类）**

#### 属性：

* **note\_id** (int): 音符的唯一标识符。
* **note\_name** (str): 音符的名称（如 "C4", "D#5", "rest"）。
* **beat\_time** (float): 音符的持续时间，以节拍数为单位。
* **volume** (float): 音符的音量，范围从 0 到 1。
* **timbre** (str): 音符的音色（如 "piano", "violin", 等）。
* **duration** (float): 音符的持续时间，单位秒（基于 beat\_time 和 bpm 计算）。
* **sample\_rate** (int): 音频采样率（Hz）。
* **waveform** (np.ndarray | None): 音符的音频波形。

#### 方法：

* **\_\_init\_\_(self, timbre, bpm, sample\_rate, note\_name, beat\_time, volume, note\_id)**
  + 构造函数，用于初始化音符的各种属性，音符名称、节拍、音色等。
* **note\_midi()**
  + 将音符名称转换为 MIDI 音高编号，便于音符比较。
  + **返回值**: MIDI 编号（int）。
  + **异常**: 如果是 rest 或音符格式非法，抛出 ValueError。
* **\_\_lt\_\_(self, other)**、**\_\_gt\_\_(self, other)**、**\_\_eq\_\_(self, other)**
  + 比较当前音符与其他音符的音高（不支持与休止符比较）。
* **note2freq(note)**
  + 将音符名称转换为频率（Hz）。
  + **返回值**: 音符对应的频率（float）。
  + **异常**: 如果音符格式无效，抛出 ValueError。
* **generate\_waveform()**
  + 生成该音符对应的音频波形。
  + **返回值**: 音符的波形数据（np.ndarray）。
  + **异常**: 如果音色不支持，抛出 ValueError。
* **play\_for\_preview()**
  + 播放音符的音频波形，用于预览。
* **show\_time\_and\_freq\_domain()**
  + 显示音符的时域和频域图像，帮助分析音符的波形和频谱。

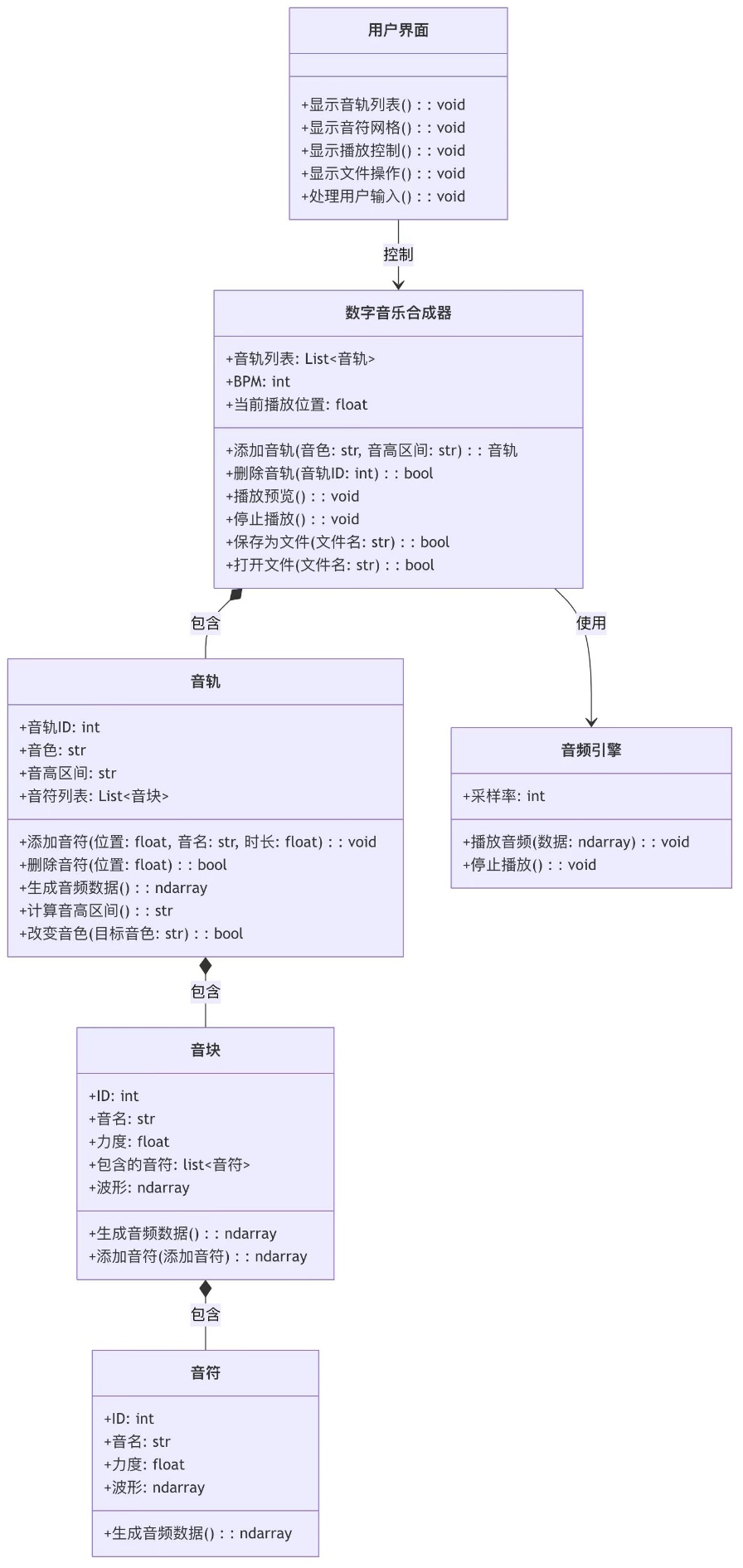
#### 测试函数：

* **play\_for\_preview()** 和 **show\_time\_and\_freq\_domain()** 配合使用，可以用来预览音符的波形和展示其时域/频域图像。

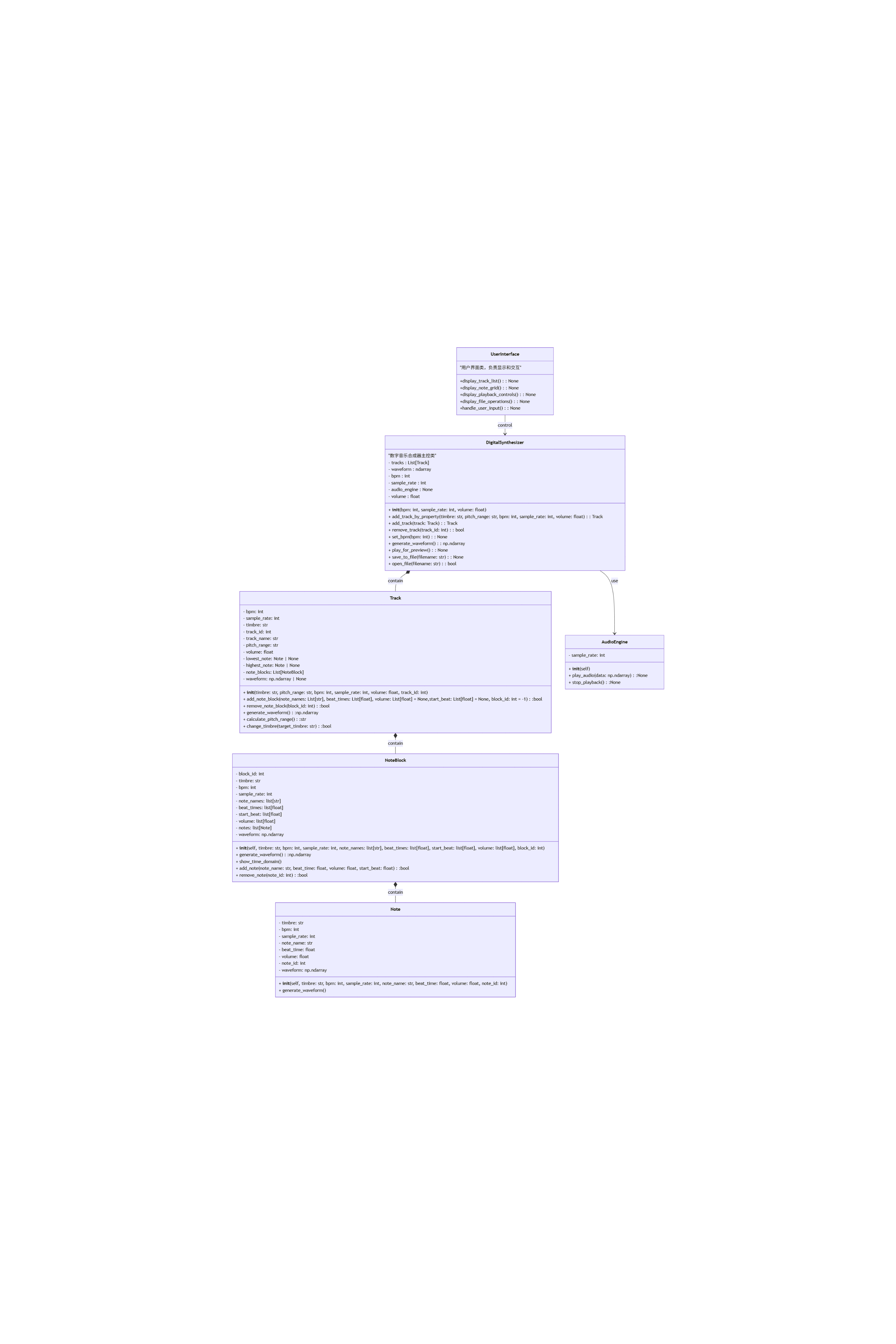
# 

## 2 UML类图

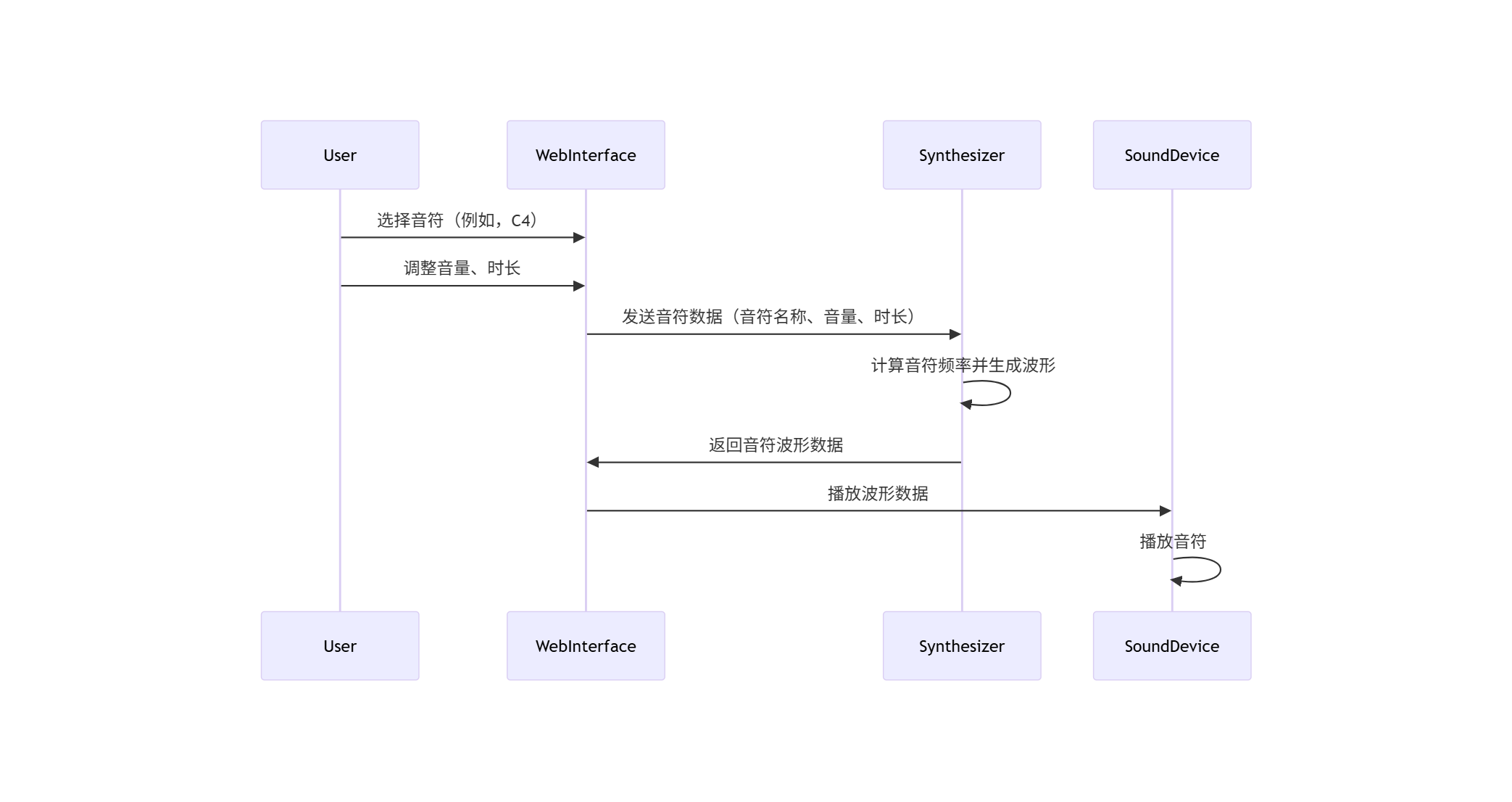
粗略：



详细：



三、序列图



四、构件图

