应用软件课程设计
设计文档

成员：冯驰、刘梓池、钟睿哲、张峰玮、何梓欣、王奥迪、韩雨虹、姚思悦

1 概述

设计该项目时，我们根据题目和需求，分为前端页面、后端数据库和核心功能实现三个方面进行设计。设计思路及具体实现见后续部分。

2 前端页面设计

2.1 前端独立页面设计

根据附件的页面设计文档，我们在分工后对页面进行了组件设计和功能实现，以下是各页面设计思路。

2.1.1 AI 续写音乐界面

使用 vuetify 的上传音乐文件组件和音乐播放组件，在上传音乐文件后传给后端进行 ai 续写处理。后端处理之后，返回给前端用户进行试听。

如果用户对该音频满意，则可以选择发布音频，在发布时填写音频的标题、详情和图片等，并同意用户隐私协议来进行发布。发布后的音频会出现在用户的已发布列表和广场上。

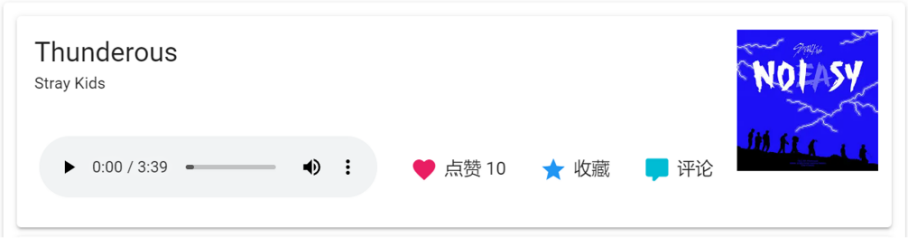
2.1.2 注册登录界面

用户可以使用手机号进行注册，在注册时输入用户名、密码和手机验证码。注册成功后即可用该用户名登录。其中用户名和手机号在数据库中是一对一关系，且用户名不可重复使用。

用户输入用户名和密码后，如果在数据库中匹配，则可成功登陆，跳转到用户首页界面。如果不匹配，则登录失败。如果用户忘记密码，则可在忘记密码界面使用手机验证码进行修改密码。修改成功后可继续登录。

2.1.3 音乐卡片组件

在本项目中，我主要负责音乐卡片组件的编写，在首页、热门音频、个人中心页面中音乐卡片作为其子组件存在，所以一个功能完善的音乐卡片对于我们的音乐论坛来说十分重要。由于前端框架采用的是 vuetify，所以在设计之初选择了 vuetify 官方文档中一个比较合适的 Cards 组件作为基础并根据实际需求进行了修改。



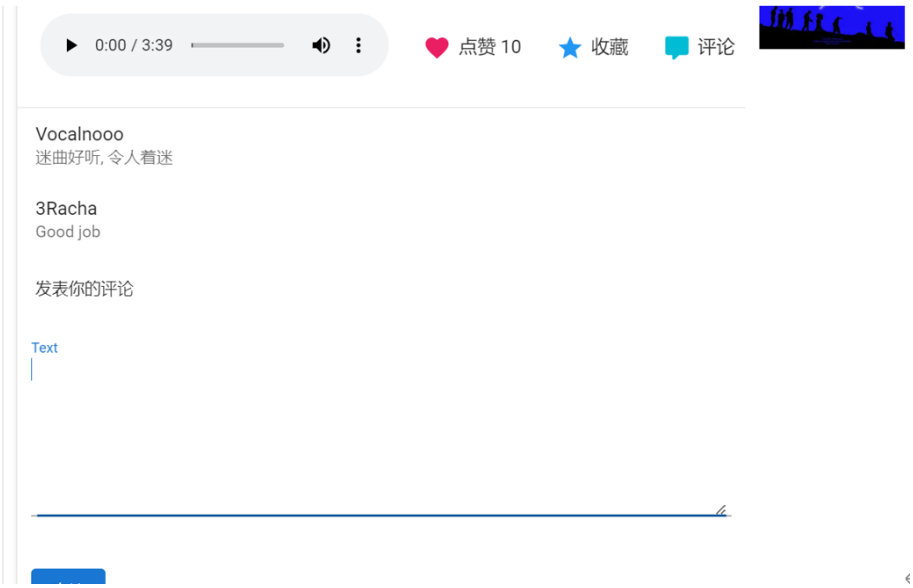
以下面这条音频为例，“Thunderous”是音频发布的标题，可以类比歌曲名称；“Stray Kids”是发布者的昵称，点击播放按钮可以播放本音频，其他用户可以执行的操作有点赞、取消点赞，收藏、取消收藏、评论。

音乐卡片的代码文件是 muscard.vue，html 部分由一个 v-for 指令的 v-card 组成，v-for 指令把数组的选项列表渲染，用到了 item in items 语法，v-card 是整体的结构，下图展示了部分功能的 html 代码，涉及的函数有点击播放按钮的 playmusic(item)，点赞和取消点赞的 toggle_like(item)。

```
<v-card-subtitle v-text="item.artist"></v-card-subtitle>
<v-card-actions>
  <v-col cols=auto>
    <div class="audio_con">
    <audio ref='audio' src="../assets/demo.mp3" controls loop class="myaudio" @click="playmusic(item)"> </audio>
    </div>
  </v-col>
  <v-col cols=auto>
    <v-btn icon color="pink" @click="toggle_like(item)">
      <v-icon>mdi-heart</v-icon>
    </v-btn>
     <v-text>点赞 {{item.likecnt}}</v-text>
  </v-col>
```

除此之外，为了美观和简洁，评论功能采用 Expansion panels 即扩展面板，只有点击评论图标才会展开其面板。其他人发表的评论也是使用了 v-for 指令进行数组列表渲染。

```
      <v-expand-transition>
        <div v-show="item.comment">
          <v-divider></v-divider>
          <!--其他人发表的评论-->
          <v-vol v-for="(cmt, j) in item.comments" :key="j">
          <v-list-item two-line>
          <v-list-item-content>
          <v-list-item-title>{{cmt.id}}</v-list-item-title>
          <v-list-item-subtitle>{{cmt.content}}</v-list-item-subtitle>
          </v-list-item-content>
          </v-list-item>
          </v-vol>
          <v-card-text>
            发表你的评论
          </v-card-text>
```



热门音频页面和首页的整体框架一致，只需要分别在 html 代码部分加入 <muscard> </muscard>，在 script 代码部分加入 componets 组件即可调用 muscard 子组件，降低了代码的冗余度和可读性。

```
<div style="margin:auto">
<muscard> </muscard>
</div>
```

```
import muscard from './muscard.vue'
export default {

  components:{
    'muscard':muscard
  },
```

2.1.4 个人信息设置页面

首先是对整体页面的渲染，该部分与首页设计边框保持一致：上边栏选用相同的图片，左边栏统一设计为显示头像和用户昵称，且可以通过选项跳转到不同的页面。

接下来是对个人信息提交部分的卡片设计：

(1) 账号信息卡片：设置了三个输入栏，分别为用户昵称、邮箱和个人简介，其中昵称和个人简介规定了字数，邮箱规定了格式。

(2) 个人信息卡片：设置了一个选项组件和一个输入组件，对用户性别进行选择，对用户生日设计了可以通过日历表进行选择填写。

(3) 访问权限卡片：设置了一个选项组件，用来规定用户的作品权限。

(4) 提交部分：设置了点击”submit” 按钮是否可以提交，点击”clear” 按钮是否可以清空全部界面。

最后是代码实现，在写前端时使用了 vuetify 语言，按照以上设计添加了不同的组件，并使用

驼峰式实现了与后端的 api 接口交互，成功获取到数据库中存储的信息，并且可以将填写的数据上传到数据库中。

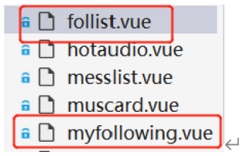
2.1.5 个人中心

个人中心界面框架与广场页一致，同时直接使用音频卡片子组件，显示“我”发布的音频。导航栏可以跳转“我的关注”、“新消息”、“我的评论”等界面。

2.1.6 我的关注

我的关注界面显示关注的用户，在 myfollwing 中使用 follist 列表组件。列表组件易于在集合中识别特定项目，为组织一组文本和图像提供一致的样式。

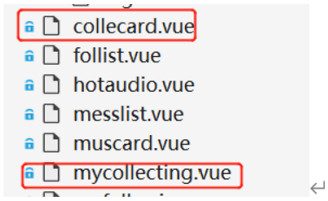
使用简单头像列表，使用 v-list-item-icon, v-list-item-title 和 v-list-item-avatar。




```
<v-list>
    <v-virtual-scroll
                    :items="items"
                    :item-height="610"
                    height="700">
        <template v-slot:default="{ item }">
            <v-list-item v-for="item in items"
                         :key="item.title">
                <v-list-item-icon>
                    <v-icon v-if="item.icon"
                            color="pink">
                        mdi-star
                    </v-icon>
                </v-list-item-icon>
                <v-list-item-content>
                    <v-list-item-title class="text-h7" v-text="item.title"></v-list-item-title>
                </v-list-item-content>
                <v-list-item-avatar>
                    <v-img :src="item.avatar"></v-img>
                </v-list-item-avatar>
            </v-list-item>
            </template>
    </v-virtual-scroll>
</v-list>
```

2.1.7 我的收藏

可以看到我收藏的音乐。在 mycollecting.vue 中使用 collecard 组件，在卡片中使用栅格令布局更有条理。

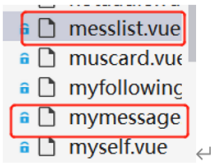



```
<v-container fluid>
    <v-row dense>
        <v-col v-for="card in cards"
               :key="card.title"
               :cols="card.flex">
            <v-card>
                <v-img :src="card.src"
                       class="white--text align-end"
                       gradient="to bottom, rgba(0,0,0,.1), rgba(0,0,0,.5)"
                       height="200px">
                    <v-card-title v-text="card.title"></v-card-title>
                </v-img>

                <v-card-actions>
                    <v-spacer></v-spacer>
                    <v-col cols=auto>
                        <v-btn @click="item.star=!item.star" icon color="blue">

                            <v-icon>mdi-star</v-icon>
                        </v-btn>
                    </v-col>
                    <v-col cols=auto>
                        <v-btn @click="item.comment=!item.comment" icon color="grey">
                            <v-icon>mdi-share-variant</v-icon>
                        </v-btn>
                        </v-col>
                </v-card-actions>
            </v-card>
        </v-col>
```

2.1.8 新收藏

新通知界面显示收到的通知、消息，包括最近的通知和之前收到的通知，在 mymessage 中使用 messlist 列表组件。

列表组件将 v-list-item-avatar 和 v-list-item-icon 合并为一个单行列表。




```
<v-list subheader>
    <v-subheader>最近的通知</v-subheader>
        <v-list-item v-for="chat in recent"
                     :key="chat.title">
            <v-list-item-avatar>
                <v-img :alt="`${chat.title} avatar`"
                       :src="chat.avatar"></v-img>
            </v-list-item-avatar>

            <v-list-item-content>
                <v-list-item-title v-text="chat.title"></v-list-item-title>
            </v-list-item-content>

            <v-list-item-icon>
                <v-icon :color="chat.active ? 'deep-purple accent-4' : 'grey'">
                    mdi-message-outline
                </v-icon>
            </v-list-item-icon>
        </v-list-item>
</v-list>

        <v-divider></v-divider>

        <v-list subheader>
            <v-subheader>之前的通知</v-subheader>

            <v-list-item v-for="chat in previous"
                         :key="chat.title">
                <v-list-item-avatar>
                    <v-img :alt="`${chat.title} avatar`"
                           :src="chat.avatar"></v-img>
                </v-list-item-avatar>

                <v-list-item-content>
                    <v-list-item-title v-text="chat.title"></v-list-item-title>
                </v-list-item-content>
            </v-list-item>
```

2.1.9 创作中心页面

该页面设计文档单独附在文件最后。

2.2 核心功能实现

使用 LSTM 生成音乐，给一段音乐的开头作为 context/condition，通过 step by step 的方式，生成指定长度的音乐片段。

技术框架设计：基于 Python，选择 PyTorch 作为 Deep Learning 的框架，ffmpeg 和 scipy 进行音频处理，numpy 进行傅里叶变换。

2.2.1 原理简述

(1) 音频预处理：对音频进行降采样，降低训练开销，在音频质量和训练开销之间进行折中；以音频采样率为时间窗，对音频进行傅里叶变换，得到频域分量，作为网络的输入特征量；

(2) 网络构造：使用 LSTM 作为 predictor，![LSTM-structure](fig/LSTM-structure.png) 相比于 RNN 引入了多个门控单元，更好捕捉长期记忆，缓解了梯度衰减和梯度爆炸的问题；引入

Dropout 防止过拟合；

(3) 训练：以输入的音频作为 condition/context，进行接下来指定长度的音频的预测；使用 MSE 进行约束，作为损失函数；使用 Adam 优化器，迭代更新网络参数；

(4) 推理：以输入的音频作为 condition/context，进行接下来指定长度的音频的预测；使用 step-by-step 的预测方式进行迭代，直到生成的音频长度达到指定长度；

(5) 转为可播放音频 (.wav): 对网络的输出量进行傅里叶逆变换，得到可播放的音频；

2.2.2 代码结构

‘dataset‘：数据结构，训练时依靠该迭代器取数据

‘model‘：网络结构

‘utils‘：实用工具，如音频处理代码

‘main.py‘：训练网络

‘inference.py‘：测试/推理