



# QuizGame 测验系统开发文档



## 1. 项目概述

QuizGame是一个基于Python和Tkinter开发的测验系统，用于进行在线测验。系统支持多种题型（选择题、填空题、判断题），具有计时、计分、错题收藏等功能，并提供音效反馈和答题分析功能。



## 2. 系统架构



### 2.1 技术栈

类别	技术选型
编程语言	Python 3.6+
GUI框架	Tkinter
数据处理	openpyxl
音频处理	pygame
图像处理	Pillow (PIL)
文件格式	Excel(.xlsx)、文本文件(.txt)、音频文件(.wav)



### 2.2 核心模块




- QuizGame.py：主程序文件，包含GUI界面和主要业务逻辑
- Exam.py：数据处理模块，包含题目和学生信息管理
- 数据文件：
  - Quiz.xlsx：题目库
  - 名单.xlsx：学生信息
  - Record.txt：最高分记录
  - 小测验游戏说明.txt：游戏说明文档
  - sounds/：音效文件目录









## 3. 功能模块说明



### 3.1 用户认证模块

-  支持学号和姓名登录
-  学号必须为10位数字
-  姓名必须与名单完全匹配





## 3.2 测验管理模块

-  支持三种题型：选择题、填空题、判断题
-  每种题型随机抽取5道题目
-  题目选项随机排序
-  支持计时功能
-  实时显示得分
-  题型正确率统计

## 3.3 答题功能

-  选择题：单选模式
-  填空题：文本输入
-  判断题：对/错选择
-  答题后即时反馈
-  支持题目收藏功能
-  答题音效反馈

## 3.4 收藏功能

-  可收藏任意题目
-  支持移除收藏
-  提供收藏集查看功能
-  支持收藏集动态更新

## 3.5 成绩管理

-  实时计算得分
-  记录最高分
-  保存答题记录
-  题型正确率分析

## 3.6 音效系统

-  支持自定义音效
-  自动生成默认音效
-  音效音量控制
-  音效文件管理

## 3.7 音乐播放模块

- 🎵 支持音乐播放
- 🔄 自动循环播放
- 🔊 音量控制
- 📁 音乐文件管理

## 📁 4. 数据文件格式

---

### 📊 4.1 Quiz.xlsx

包含以下工作表：

- 📄 测验信息：测验名称、时长
- 📄 选择题：题目、选项、答案、解析
- 📄 填空题：题目、答案、解析
- ✅ 判断题：题目、答案、解析

### 👥 4.2 名单.xlsx

- 📄 学号（10位数字）
- 👤 姓名

### 🔊 4.3 音效文件

- 📁 sounds/correct.wav ：答对音效
- 📁 sounds/wrong.wav ：答错音效

## 📖 5. 使用说明

---






### 💻 5.1 运行环境要求

- 🐍 Python 3.x
- 📦 依赖包：
  - tkinter
  - openpyxl
  - pygame
  - numpy
  - Pillow

### 🚀 5.2 启动方式







1. 直接运行 QuizGame.exe
2. 或通过Python运行 QuizGame.py

### 📋 5.3 操作流程

1.  输入学号和姓名登录
2.  按顺序完成各类题型
3.  可随时收藏题目
4.  完成后查看成绩
5.  查看答题分析

## 6. 注意事项

---

-  确保所有数据文件在正确位置
-  学号必须为10位数字
-  答题时注意剩余时间
-  可随时收藏题目以便复习
-  确保音效文件完整性
-  关注答题统计分析




## 7. 更新日志

---

### v1.1.0

-  新增音效反馈功能
-  新增答题分析功能
-  优化收藏功能
-  修复已知问题

### v1.2.0

-  新增音乐播放功能
-  优化音乐播放体验
-  修复已知问题

## 8. 音乐播放模块说明

---

### 8.1 音乐控制面板

- 位置：所有页面底部
- 组件：
  - 播放/暂停按钮
  - 导入音乐按钮
  - 音量调节滑块
- 实现方法：`create_music_controls()`

### 8.2 音乐播放功能





- 导入音乐： `import_music()`
  - 支持格式：mp3, wav
  - 自动循环播放
  - 异常处理机制
- 播放控制： `play_music()` , `pause_music()`
  - 播放/暂停切换
  - 状态同步
  - 界面更新
- 音量控制： `change_volume()`
  - 实时调节
  - 范围：0-1
  - 持久化存储

## 9. 数据结构



---

### 9.1 Quiz.xlsx



包含以下工作表：

-  测验信息：测验名称、时长
-  选择题：题目、选项、答案、解析
-  填空题：题目、答案、解析
-  判断题：题目、答案、解析

### 9.2 名单.xlsx

-  学号（10位数字）
-  姓名

### 9.3 音效文件

-  `sounds/correct.wav` ：答对音效
-  `sounds/wrong.wav` ：答错音效

### 9.4 音乐相关

```
python
1  # 音乐播放状态
2  {
3      'current_music': str, # 当前音乐文件路径
4      'music_playing': bool, # 播放状态
5      'volume': float,      # 音量大小(0-1)
6  }
```



## 10. 异常处理

---



### 10.1 音乐播放异常

python

```
1  try:
2      pygame.mixer.music.load(file_path)
3      pygame.mixer.music.play(-1)
4  except Exception as e:
5      messagebox.showerror("错误", f"无法加载音乐文件: {str(e)}")
```



### 10.2 其他异常处理

// ... existing code ...



## 11. 性能优化

---



### 11.1 音乐播放优化

1. 使用多线程播放
2. 异步加载音乐文件
3. 内存管理优化
4. 音量调节性能优化



### 11.2 其他优化

// ... existing code ...



## 12. 扩展性设计

---



### 12.1 音乐模块扩展

1. 支持更多音频格式
2. 播放列表功能
3. 音频效果处理
4. 自定义音效



### 12.2 其他扩展

// ... existing code ...



## 13. 测试建议

---



### 13.1 音乐功能测试

- 1. 音乐文件加载测试
- 2. 播放控制测试
- 3. 音量调节测试
- 4. 多线程稳定性测试
- 5. 内存占用测试

## 🎵 13.2 其他测试

// ... existing code ...

## 🎵 14. 部署说明

---

### 🎵 14.1 环境要求

- Python 3.6+
- pygame
- Pillow
- numpy

### 🎵 14.2 安装步骤

- 1. 安装Python环境
- 2. 安装依赖库：

```
bash
```

```
1 | pip install pygame
2 | pip install Pillow
3 | pip install numpy
```

### 🎵 14.3 运行说明

- 1. 确保所有依赖库已安装
- 2. 运行主程序：

```
bash
```

```
1 | python QuizGame.py
```

## 🎵 15. 维护建议

---

### 🎵 15.1 音乐模块维护

- 1. 定期检查音频文件完整性
- 2. 监控内存使用情况
- 3. 更新音频处理库

#### 4. 优化播放性能

## 🎵 15.2 其他维护

// ... existing code ...

## 游戏模式实现

---

### 模式设计

QuizGame 实现了三种游戏模式，通过不同的时间限制来提供不同的游戏体验：

#### 1. 简单模式

- 无时间限制
- 适合初学者
- 代码中通过 `game_mode == "simple"` 判断

#### 2. 中等模式

- 10秒时间限制
- 提供适度挑战
- 代码中通过 `game_mode == "medium"` 判断

#### 3. 困难模式

- 5秒时间限制
- 提供高难度挑战
- 代码中通过 `game_mode == "hard"` 判断

### 核心实现

#### 1. 模式配置

python

```
1 self.mode_times = {
2     "simple": 20, # 简单模式20秒（实际不显示计时器）
3     "medium": 10, # 中等模式10秒
4     "hard": 5 # 困难模式5秒
5 }
```

#### 2. 计时器实现

python

```
1 def question_countdown(self):
2     """每题倒计时"""
3     if self.game_mode != "simple" and self.question_time_left > 0:
4         self.question_timer_label.config(text=f"剩余时间: {self.question_time_left}s")
5         self.question_time_left -= 1
6         self.question_timer_id = self.root.after(1000, self.question_countdown)
```



```
7         elif self.game_mode != "simple" and self.question_time_left <= 0:
8             # 时间到，显示超时提示并进入下一题
9             self.question_timer_label.config(text="时间到！", fg="red")
10            messagebox.showinfo("超时", "超时啦！")
11            self.next_question()
```

## 关键功能

### 1. 模式选择

- 在登录界面通过单选按钮实现
- 使用 `StringVar` 存储选择结果
- 登录时保存到 `self.game_mode`

### 2. 计时器显示

- 仅在非简单模式下显示
- 使用 `tk.Label` 显示倒计时
- 每秒更新一次显示

### 3. 超时处理

- 时间到时显示提示框
- 自动进入下一题
- 取消当前计时器

### 4. 计时器管理

- 在切换题目时取消计时器
- 在进入新题目时重置计时器
- 在退出答题时清理计时器

## 注意事项

1. 计时器使用 `root.after()` 实现，需要注意及时取消
  2. 在切换模式或退出时需要清理计时器资源
  3. 计时器显示需要考虑主题切换的影响
  4. 超时处理需要考虑用户体验，提供清晰的提示
-