|  |  |
| --- | --- |
| 校徽 | **文字组合横排** |
| **课 程 报 告** |

|  |
| --- |
| XML编程技术 |

|  |  |
| --- | --- |
| 姓名： | 马云骥 |
| 班级： | J软件(嵌入)(专转本)2102 |
| 学号： | 4211153047 |
|  |  |
| 教师： | 余春堂 |

|  |
| --- |
| 2024年5月 |

要 求

开发一个通信录管理系统，具有①用户登录功能②联系人增加、删除、编辑、查找等基本功能。

程序要求：

1. 使用Java语言或其它语言开发均可，界面可以是基于Web的，也可以基于Windows窗口的。
2. 使用XML文件做为唯一数据存储格式。
3. 联系人信息自行设计，可以包括姓名、电话号码、照片等信息。如果包括照片，也只能使用XML格式存储。
4. 功能复杂程度和完善程度自行决定。

报告要求：

1. 按照规定的章节要求撰写课程报告。
2. 正文字号五号，二级标题四号，三级标题小四号。
3. 图文并茂，排版美观。
4. 文档总长度8-10页（包括封面和题目要求）。
5. 纸质版A4纸，正反打印。

# 需求分析

**1.1 背景和动机**

随着现代社会的发展，人们的联系人数量日益增多，包括家庭成员、朋友、同事、客户等。传统的纸质通讯录管理方式已经无法满足现代人的需求，数字化管理成为必然趋势。使用计算机或手机等电子设备管理联系人信息，不仅方便快捷，而且更易于维护和查找。

然而，现有的一些联系人管理应用程序大多依赖于云存储，这可能引发数据隐私和安全问题。此外，某些用户可能更希望使用本地存储方式管理联系人信息。因此，本项目旨在开发一个基于本地存储的联系人管理系统，使用XML文件作为数据存储格式，以确保用户数据的安全和隐私。

**1.2 需求概述**

本项目的目标是开发一个基于Python语言和Tkinter图形界面的联系人管理系统。该系统将具备以下主要功能：

1. **用户注册和登录：**允许用户创建新账号和登录已有账号。
2. **联系人管理：**包括添加、删除、编辑和查找联系人等功能。
3. **数据存储：**使用XML文件存储用户和联系人信息，确保数据的安全性和可维护性。
4. **照片管理：**支持为每个联系人添加照片，并以Base64编码形式存储在XML文件中。

**1.3 功能需求**

**1.3.1 用户管理**

1. **用户注册：**
   1. 用户可以通过输入用户名和密码创建新账号。
   2. 系统需要验证用户名的唯一性，防止重复注册。
2. **用户登录：**
   1. 用户可以通过输入用户名和密码登录系统。
   2. 系统需要验证用户名和密码的正确性。
3. **用户注销：**
   1. 用户可以在登录状态下选择注销当前账号。

**1.3.2 联系人管理**

1. **添加联系人：**
   1. 用户可以为每个联系人添加姓名、电话和照片。
   2. 照片以Base64编码形式存储在XML文件中。
   3. 可以随机生成联系人数据用于测试。
2. **删除联系人：**
   1. 用户可以删除联系人信息，系统需要从XML文件中移除相应的数据。
3. **编辑联系人：**
   1. 用户可以修改已有联系人的信息，包括姓名、电话和照片。
4. **查找联系人：**
   1. 用户可以通过输入姓名或电话关键字查找联系人，系统返回匹配的联系人列表。

**1.3.3 数据存储**

使用XML文件存储数据，用户信息和联系人信息分别存储在两个XML文件中，XML文件结构清晰，易于解析和维护。

**1.4 非功能需求**

1. **易用性：**系统界面友好，操作简便，用户无需复杂的学习即可上手使用。
2. **可靠性：**系统能够稳定运行，确保数据的正确性和完整性。
3. **安全性：**用户数据仅存储在本地，确保隐私和安全。
4. **可维护性：**代码结构清晰，模块化设计，便于后期维护和扩展。

**1.5 用户需求**

**1.5.1 用户分类**

1. **普通用户：**普通用户可以注册和登录系统，管理自己的联系人信息。
2. **管理员用户：**默认的管理员用户（用户名为root，密码为root），可以查看和管理所有用户的联系人信息。

**1.5.2 使用场景**

1. **个人使用：**用户可以使用该系统管理自己联系人信息，包括家庭成员、朋友、同事等。
2. **小型企业使用：**小型企业可以使用该系统管理客户和供应商信息，方便查找和联系。

**1.6 项目开发环境**

1. **开发语言：**Python
2. **图形界面库：**Tkinter
3. **其它第三方库：**
   1. Faker：用于测试时生成随机的联系人姓名和电话信息。
   2. Requests：处理网络请求，获取随机的联系人头像。
   3. Pillow：对头像图片进行剪裁、统一分辨率处理。
4. **数据存储：**XML文件
5. **开发工具：**Pycharm或其他Python开发环境
6. **操作系统：**支持Windows

**1.7 项目目标**

通过本项目的开发，旨在实现一个高效、便捷、安全的联系人管理系统，满足用户日常管理联系人信息的需求。系统应具有良好的用户体验和可靠性，能够在不同平台上稳定运行。通过该项目，开发者可以掌握Python编程、Tkinter图形界面设计以及XML数据存储和处理的相关知识和技能。

# 第二章 总体设计

**2.1 系统架构**

系统采用MVC（模型-视图-控制器）架构：

1. **模型（Model）：**负责数据存储和管理，包括用户和联系人信息的存储和操作。
2. **视图（View）：**负责用户界面的显示和交互。
3. **控制器（Controller）：**负责处理用户输入，并更新模型和视图。

**2.2 模块划分**

1. **用户管理模块：**负责用户的注册、登录、注销等功能。
2. **联系人管理模块：**负责联系人信息的添加、删除、编辑和查找等功能。
3. **数据存储模块：**负责将用户和联系人信息存储到XML文件中。
4. **界面模块：**使用Tkinter实现图形用户界面，提供用户交互功能。

**2.3 数据存储设计**

**users.xml** 文件结构：

<users>

   <user>

       <username>root</username>

       <password>root</password>

       <contacts/>

   </user>

   <user>

       <username>user1</username>

       <password>password1</password>

       <contacts>

           <contact id="contact1"/>

           <contact id="contact2"/>

       </contacts>

   </user>

</users>

**contacts.xml** 文件结构：

<contacts>

   <contact id="contact1">

       <name>John Doe</name>

       <phone>1234567890</phone>

       <photo>base64\_encoded\_photo</photo>

   </contact>

   <contact id="contact2">

       <name>Jane Smith</name>

       <phone>0987654321</phone>

       <photo>base64\_encoded\_photo</photo>

   </contact>

</contacts>

# 第三章 详细设计与编码实现

**3.1 用户管理模块**

**3.1.1 用户注册**

**功能：**添加新用户信息到users.xml文件

**实现：**（完整代码篇幅过长，这里仅说明关键部分，下同）

def add\_user(self, username, password):

   new\_user = ET.SubElement(self.tree.getroot(), "user")

   ET.SubElement(new\_user, "username").text = username

   ET.SubElement(new\_user, "password").text = password

   ET.SubElement(new\_user, "contacts")

   self.save\_xml()

**3.1.2 用户登录**

**功能：**验证用户名和密码

**实现：**

def validate\_user(self, username, password):

   for user in self.tree.findall("user"):

       if user.find("username").text == username and user.find("password").text == password:

           return True

   return False

**3.2 查找联系人**

**3.2.1 添加联系人**

**功能：**添加联系人信息到contacts.xml文件，并关联到当前用户

**实现：**

def add\_contact(self, username, name, phone, photo=None):

   contact\_id = str(uuid.uuid4())

   new\_contact = ET.SubElement(self.tree.getroot(), "contact", id=contact\_id)

   ET.SubElement(new\_contact, "name").text = name

   ET.SubElement(new\_contact, "phone").text = phone

   if photo:

       encoded\_photo = encode\_photo\_to\_base64(photo)

       ET.SubElement(new\_contact, "photo").text = encoded\_photo

   self.save\_xml()

   self.add\_contact\_to\_user(username, contact\_id)

**3.2.2 删除联系人**

**功能：**从contacts.xml文件中删除联系人信息，并更新用户的联系人列表

**实现：**

def delete\_contact(self, contact\_id):

   contact = self.tree.find(f"contact[@id='{contact\_id}']")

   self.tree.getroot().remove(contact)

   self.save\_xml()

   self.remove\_contact\_from\_user(contact\_id)

**3.2.3 编辑联系人**

**功能：**更新联系人信息

**实现：**

def update\_contact(self, contact\_id, name, phone, photo=None):

   contact = self.tree.find(f"contact[@id='{contact\_id}']")

   contact.find("name").text = name

   contact.find("phone").text = phone

   if photo:

       encoded\_photo = encode\_photo\_to\_base64(photo)

       contact.find("photo").text = encoded\_photo

   self.save\_xml()

**3.2.4 查找联系人**

**功能：**根据姓名或电话查找联系人

**实现：**

def search\_contacts(self, username, keyword):

   user = self.find\_user(username)

   contact\_ids = user.find("contacts").text.strip().split()

   contacts = [self.tree.find(f"contact[@id='{cid}']") for cid in contact\_ids]

   results = [contact for contact in contacts if keyword.lower() in contact.find("name").text.lower() or keyword.lower() in contact.find("phone").text.lower()]

   return results

**3.3 界面模块**

使用Tkinter实现图形用户界面，提供用户交互功能。通过调用上述模块中的功能，实现界面的各种功能。

# 第四章 测试

**4.1 测试环境**

**操作系统：**Windows 11

**Python 版本：**3.12

**必要的 Python 库：**tkinter、pillow、xml.etree.ElementTree、uuid、base64、io、requests

**4.2 测试用例**

针对各个功能模块，设计详细测试用例，以覆盖系统的主要功能和可能出现的异常情况。

**4.2.1 用户注册和登录**

**测试用例1：成功注册新用户**

前提条件：系统启动，进入登录注册界面。

操作步骤：输入唯一的用户名和密码，点击注册按钮。

预期结果：提示注册成功，用户信息存储在users.xml文件中。

**测试用例2：注册重复用户名**

前提条件：系统启动，进入登录注册界面。

操作步骤：输入已存在的用户名和任意密码，点击注册按钮。

预期结果：提示用户名已存在，注册失败。

**测试用例3：成功登录**

前提条件：系统启动，进入登录注册界面。

操作步骤：输入已注册的用户名和正确的密码，点击登录按钮。

预期结果：登录成功，进入主界面。

**测试用例4：登录失败（用户名错误）**

前提条件：系统启动，进入登录注册界面。

操作步骤：输入不存在的用户名和任意密码，点击登录按钮。

预期结果：提示用户名或密码错误，登录失败。

**测试用例5：登录失败（密码错误）**

前提条件：系统启动，进入登录注册界面。

操作步骤：输入已注册的用户名和错误的密码，点击登录按钮。

预期结果：提示用户名或密码错误，登录失败。

**4.2.2 联系人管理**

**测试用例6：成功添加联系人**

前提条件：用户已登录，进入联系人管理界面。

操作步骤：输入联系人姓名和电话，选择照片，点击保存按钮。

预期结果：提示联系人添加成功，联系人信息存储在contacts.xml文件中。

**测试用例7：成功删除联系人**

前提条件：用户已登录，至少有一个联系人。

操作步骤：选择一个联系人，点击删除按钮。

预期结果：提示联系人删除成功，联系人信息从contacts.xml文件中移除。

**测试用例8：成功编辑联系人**

前提条件：用户已登录，至少有一个联系人。

操作步骤：选择一个联系人，修改联系人信息，点击保存按钮。

预期结果：提示联系人信息更新成功，contacts.xml文件中相应信息被修改。

**测试用例9：成功查找联系人（按姓名）**

前提条件：用户已登录，至少有一个联系人。

操作步骤：输入联系人姓名关键字，点击查找按钮。

预期结果：显示匹配的联系人列表。

**测试用例10：成功查找联系人（按电话）**

前提条件：用户已登录，至少有一个联系人。

操作步骤：输入联系人电话关键字，点击查找按钮。

预期结果：显示匹配的联系人列表。

**4.2.3 界面测试**

**测试用例11：界面显示正常**

前提条件：系统启动。

操作步骤：检查登录界面、主界面和联系人管理界面各控件的布局和显示效果。

预期结果：界面布局合理，各控件显示正常，无重叠或遮挡。

**测试用例12：不同分辨率和缩放设置下的界面显示**

前提条件：系统启动，调整屏幕分辨率和缩放设置。

操作步骤：重复测试用例11。

预期结果：界面在不同分辨率和缩放设置下显示正常。

**4.2.4 性能测试**

**测试用例13：大量联系人数据的处理**

前提条件：用户已登录，系统中有大量联系人数据

操作步骤：执行联系人查询、添加、删除、编辑等操作

预期结果：系统响应及时，操作流畅，无明显卡顿或崩溃

**4.3 测试结果分析**

通过执行上述测试用例，对系统的功能、界面和性能进行了全面的验证，并记录了测试结果和发现的问题。

**4.3.1 功能测试结果**

用户注册、登录、注销等功能正常，验证用户名唯一性和密码正确性。联系人添加、删除、编辑和查找功能正常，数据存储和更新正确。

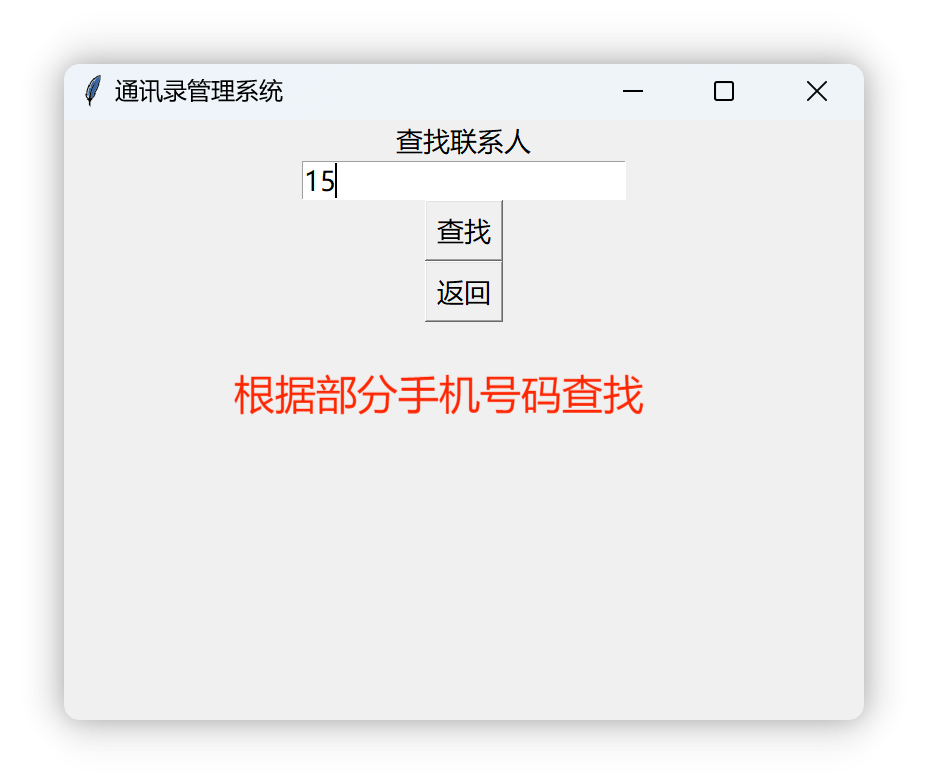
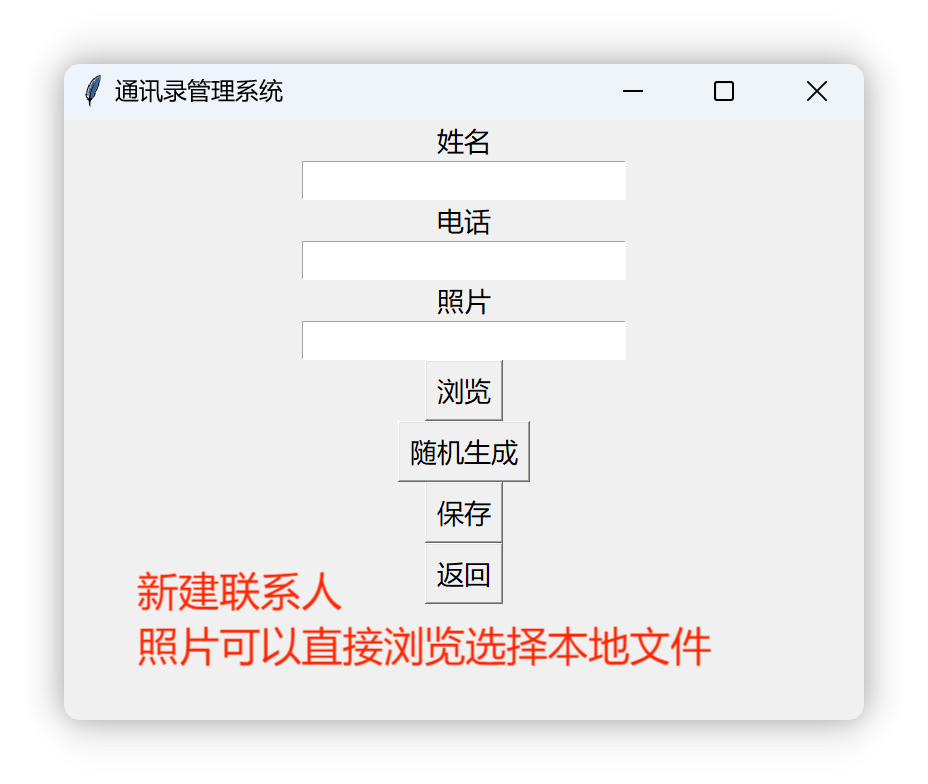
**4.3.2 界面测试结果**

界面布局合理，控件显示正常。在不同分辨率和缩放设置下，界面显示无异常。

**4.3.3 性能测试结果**

系统在处理大量联系人数据时，响应及时，操作流畅，无明显卡顿。联系人查询速度较快，能够在短时间内返回匹配结果。获取随机联系人的头像时有概率会发生卡顿。

**4.4 截图展示**

****

**4.5 问题和解决方案**

在测试过程中，我发现了一些问题，并提出了相应的解决方案以确保系统的稳定性和用户体验。

**问题1：**在高分辨率和高缩放设置下，界面显示模糊。

**解决方案：**通过设置Tkinter的DPI感知，使界面在高DPI显示器上显示清晰。

**问题2：**联系人列表过多时，界面无法滚动查看全部联系人。

**解决方案：**在联系人列表中添加滚动条，并确保界面支持调整大小。

**问题3：**原先处理联系人的ID直接使用数字编号，导致多次增删操作后ID冲突。

**解决方案：**将联系人ID从数字编号改为UUID，确保每个联系人ID的唯一性。

**问题4：**请求随机头像时可能会卡顿。

**解决方案：**为了尽量减少卡顿，确保良好的网络状况，必要时可以使用网络代理。

**4.6 测试总结**

通过系统的全面测试，验证了联系人管理系统的各项功能、界面显示效果和性能，确保了系统的稳定性和用户体验。测试过程中发现的问题得到了及时解决，为系统的进一步优化和改进提供了宝贵的参考。在实际使用中，该系统能够满足用户对联系人管理的需求，具备良好的用户体验。

# 第五章 总结

在本次实验中，我设计并实现了一个基于Python和Tkinter的联系人管理系统，通过这个过程，我深入了解了GUI应用的设计、XML文件的操作以及用户认证机制的实现。这不仅加深了我对Python编程的认识，也锻炼了我的软件开发和问题解决技能。

首先，系统的实现让我体会到了Tkinter库的强大功能，特别是在构建用户界面和处理用户交互方面。我学会了如何使用Tkinter来创建窗口、布局管理器以及各种控件。通过实际的界面设计和事件处理，我对Tkinter的使用有了更深的理解，并且能够灵活运用其功能来实现复杂的GUI应用。

其次，XML文件的读写操作在本实验中扮演了关键角色。我通过使用Python的xml.etree.ElementTree模块，成功实现了用户数据和联系人数据的存储和管理。这一过程不仅加深了我对XML数据格式的理解，也提高了我在数据持久化和数据解析方面的技能。

此外，实验中我还实现了用户注册和登录功能，设计了简单的用户认证机制，并引入了UUID来保证每个联系人的唯一标识。这使我对用户管理和权限控制有了更深入的认识，并且通过UUID的应用，解决了联系人ID冲突的问题，确保了数据的一致性和完整性。

在处理联系人照片时，我使用了Pillow库对图像进行处理，并将其编码为Base64字符串存储在XML文件中。这让我掌握了基本的图像处理和编码技术，并能够在实际项目中灵活应用。

在整个实验过程中，我还遇到了一些问题，并通过不断调整和优化代码，逐步解决了这些问题。例如，在高分辨率显示器上界面显示模糊的问题，通过设置Tkinter的DPI感知解决了；在联系人列表过多时界面无法滚动的问题，通过添加滚动条并支持界面调整大小解决了等等。

当然，本程序还存在很多问题及改进方向。界面设计还可以更加美观和友好，增加更多高级功能，如联系人分组、导入导出联系人等。为了方便测试，我在处理用户信息时直接明文存储了密码，这有很大的安全隐患，更好的办法就是更改为存储密码的哈希值，修改密码比对的逻辑。

总的来说，这次实验不仅增强了我的技术技能，也激发了我继续探索更多复杂项目的兴趣。我期待将这些经验应用于未来的项目，进一步优化应用性能和用户体验。这次实验是对我的编程能力的一次全面提升，同时也是对理论知识的实际应用，为我今后的学习和工作提供了宝贵的经验和指导。