武汉大学计算机学院
本科生课程设计报告

物业管理系统总体设计与实现

专 业 名 称　：计算机类

课 程 名 称　：高级语言程序设计

指 导 教 师　：梁意文　　职称：教授

团 队 成 员 一：常元和（2020302111410）

团 队 成 员 二：汪家伟（2020302111398）

团 队 成 员 三：袁嘉骏（2020302111169）

二〇二一年六月

目录

一、功能设计.....3
1.1 设计背景.....3
1.2 主要功能.....3
1.2.1 管理员功能.....3
1.2.2 用户信息管理.....4
1.3 UML 类图.....4
二、实验环境.....5
2.1 运行环境：QT5.12.11.....5
2.2 编程语言：C++.....5
三、技术细节.....5
3.1 结构设计.....5
3.2 类的设计与实现.....6
3.2.1 概况.....6
3.2.2 类.....6
3.3 UI 界面设计.....8
3.4 账户、密码验证.....8
3.5 页面切换.....9
3.6 信息的修改与存储.....10
3.7 前后端交互（接口）.....10
3.8 数据库.....11
四、开发心得.....12
4.1 小组分工.....12
4.2 收获和体会.....13
4.2.1 软件开发方面.....13
4.2.2 C++语言方面.....13
4.2.3 程序设计方面.....13

一、功能设计

1.1 设计背景

互联网+的生活方式里，信息已成为继劳动力、土地、资本之后的又一大资源，提高利用信息资源的效率是大势所趋。而传统的小区物业管理，工作流程繁杂性、管理复杂、收缴费用与设备维护繁琐、出错率高、效率低下。在此背景下，业主期待操作规范化、服务制度化、责任明确化、管理规范化、流程人性化，能够对自家的物业费用和投诉等情况有更透明、更直观的了解。在前期调研中，我们发现已有团队利用现有的信息基础设施，重点开发和推广应用于各类科技、经济等数据库和网络资源服务系统，取得巨大的社会效益和经济效益。这说明计算机已完全能够胜任物业管理工作，而且更加准确、方便、快捷、高效、清晰、透明，给项目查询和管理带来很大的方便。

在互联网时代，城市社区管理显得尤为重要，仅仅通过简单的物业上门管理显得费时且费力，物业管理系统的构建显得尤为重要，既能实现物业即时地接收到用户的反馈，用户在出现相关状况时也能迅速地联系到物业；本实验旨在通过C++、Qxlsx 数据库、qt 等实现物业管理系统的构建以及管理界面的美化，实现用户和物业管理人员的平台交互。

1.2 主要功能

物业管理系统的主要功能是作为提供物业服务的工具，为物业承包商、用户等提供便利。为此，该系统需要实现以下功能：

1.2.1 管理员功能

在本物业管理系统中，管理员对应社区中的物业承包商，登入管理系统界面后可以修改小区资料，并增加、删除及查询相关信息如社区名称、车位、房型等等。

管理员在管理系统中可对住户的维修以及投诉申请进行相关处理，如实时向户反馈维修或投诉处理进度，并输出至 EXCEL 表格中；并针对住户相应的房

型、车位进行收费，最后对住户的欠费进行排序；同时实现对住户的车位管理。管理员负责及时处理物业投诉、扣除用户物业费，在系统中拥有最高权限，也承担最大责任，统筹管理用户信息。

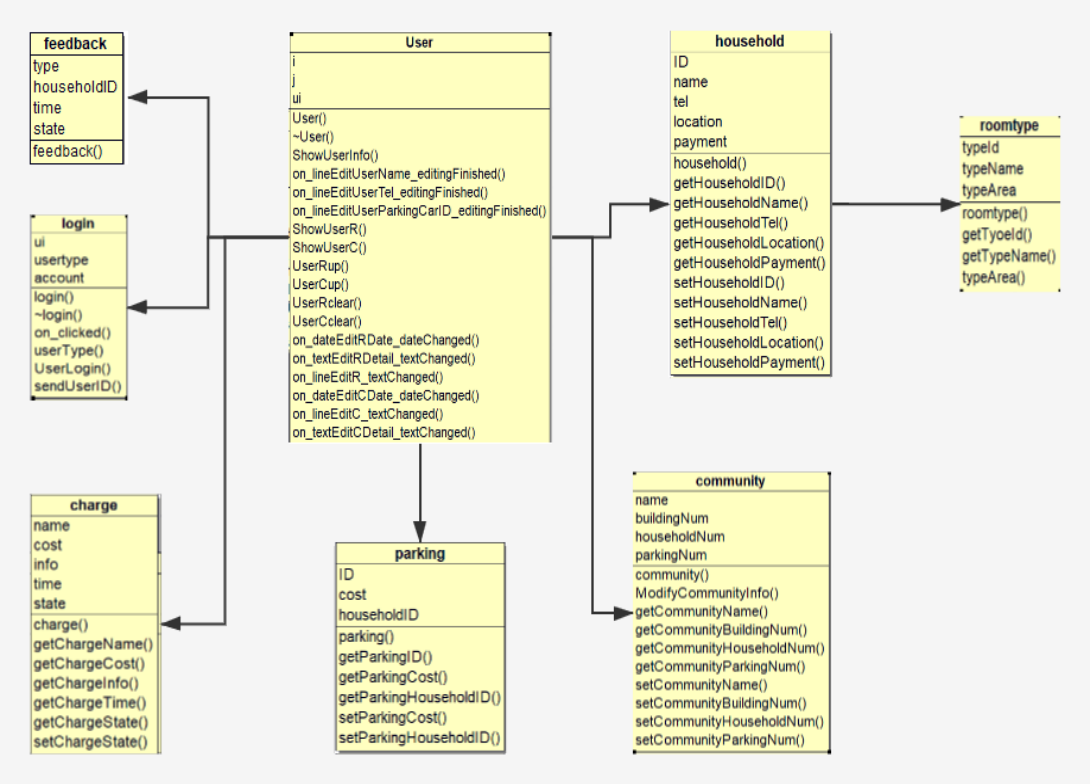
管理员对相应小区资料、住户资料可进行管理，并最终实现相应信息的输出。

1.2.2 用户信息管理

在权限范围内，用户可对自己的信息进行相应修改，但一些权限外的信息只有管理员身份能进行修改；住户在登入用户系统后更改自己的相关个人信息，如门牌号和联系电话等等，方便物业及时联系用户和上门维修。

故障保修的常见业务可直接填写系统中的报修清单反馈给物业，并根据解决程度给予评价。住户可以在用户界面添加或删除报修和相应投诉，在报修后物业方面会接收到住户申请，并处理信息，住户可在系统方面删除错误的报修信息，也可以在物业维修完成后对服务进行评价。

1.3 UML 类图



二、实验环境

2.1 运行环境：QT5.12.11

2.2 编程语言：C++

三、技术细节

3.1 结构设计

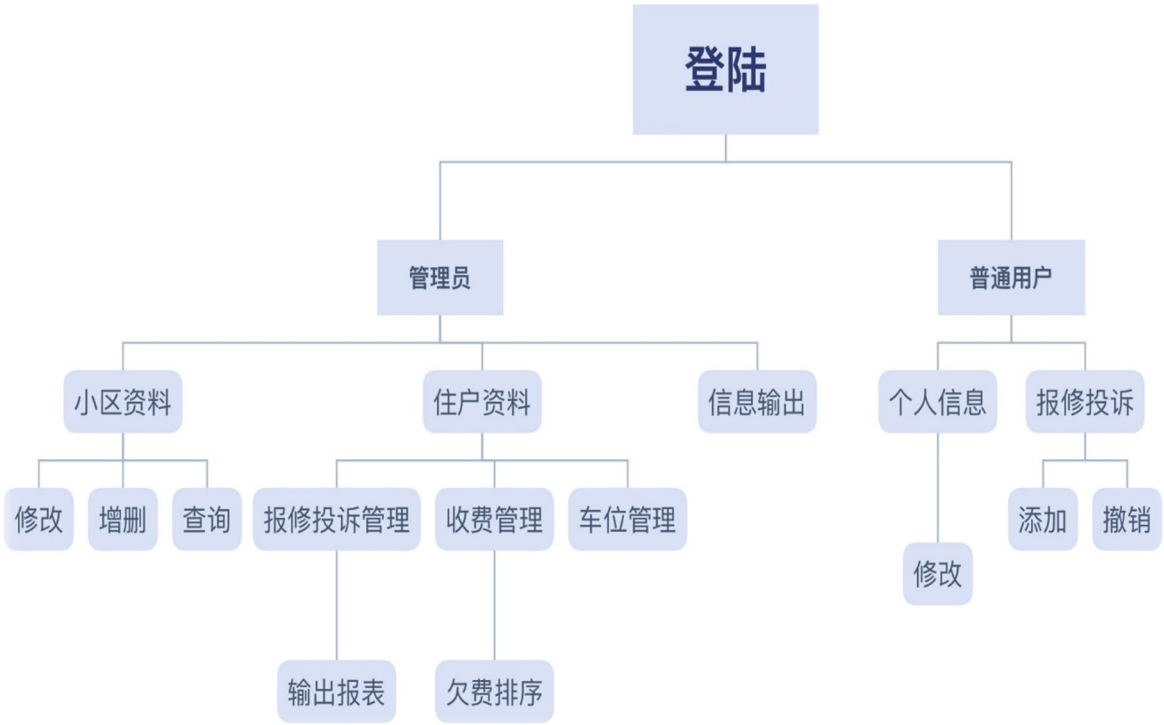


图 3.1 结构设计图

根据上述功能，系统主要分为三个界面：

1. 登录界面

2. 管理员界面

3. 普通用户界面

这样设计的好处是各个模块之间只用提供简单的接口，可以降低系统各部分的耦合性，进而使多人协作开发更加便捷，便于分工。并且由于各个部分只用调用其他部分的接口，替换和增加新的功能更加简便，程序因而具有扩展性。

3.2 类的设计与实现

3.2.1 概况

类的设计基本参照 3.1 中所述的结构图进行，各个主要的类对应的 ui 文件通过 connect 函数将按钮和各个页面联系起来，实现页面之间的切换。

3.2.2 类

login 类：在用户选择两个不同按钮时使用 usertype 函数判断账户类型，判断账户类型后再判断用户输入的账户和密码，若账户不存在，则弹出弹窗“账户不存在”；若密码错误，则弹出弹窗提示“密码错误”；登入之后根据判断的用户类型分别进入住户系统界面或者物业管理人员界面。

user 类：user 类与相应的 ui 文件及头文件组成了用户界面，用户界面各个按钮与相应功能相联系，如 on_lineEditUserName_editingFinished 函数实现了用户在用户界面进行文本修改，该函数对应的是住户姓名修改，其他几个相似的函数分别对应了修改联系电话，停放车辆的车牌号以及报修投诉的内容等。

community 类：community 类通过相关的 set 及 get 函数，实现了诸如设置和获取社区名称，设置和获取车位号码等功能，并且通过 modifyCommunityInfo 函数来修改相关信息，存储至数据表中，当用户触发输出报表按钮时，相关函数会将数据库六个中对应的 sheet 输出打印。

roomtype 类：roomtype 类实现了管理员设置户主的房型，并可在查询信息页面查到住户相关的房型信息，我们在该系统中一共设置了两套房型，值得注意的是

不同房型的大小不同，我们是根据平方米来设置房型收费价格的，所以不同房型的价格同样也是不同的，这为 charge 类收取相关房型的费用做准备。

图 3.2.2.1 房型 A

图 3.2.2.2 房型 B

parking 类：parking 类我们将其关联了住户的房号，住户的车辆的车牌号，在住户拥有车辆的前提下，物业管理人员可以针对用户的车位进行按月收费，为了计算方便，我们在这设置了车位每月一元，方便在输出报表后检查功能是否完善。

charge 类：charge 类是针对 parking 类以及 roomtype 类进行运行，为了收费方便，我们设置了在每月的特定时间进行收费，很明显收费的是住户的房费和车位费，在 get 函数获取到住户所需的收费信息后，系统会自动从住户的账户扣费，管理人员可根据待缴费用进行排序，显示住户的待缴排行。

feedback 类：feedback 类主要的函数是 feedback 函数，我们设置 feedback 函数的目的是为了当住户在用户界面向物业管理员发出报修信息后，物业管理人员可根据当前维修进度及时设置反馈状态，分别是已受理，处理中，已完成；在已完成后反馈给住户时，住户可根据自己对维修服务的满意程度对物业方面给出评价。

3.3 UI 界面设计

借助 QtDesigner 设计前端界面，选中相应控件后，点击工具栏上快速布局工具进行布局，然后通过编辑器对窗口及控件进行各种调整、设置、添加资源（例如：图片）、动作。还可以直接编辑 Qt 引以为豪的信号槽（signal 和 slot），QtDesigner 使得我们在程序设计中更快的开发出程序界面，避免了用纯代码来写一个界面的繁琐，同时 PyQt 支持界面与逻辑分离，大大降低了难度。最后使用 DOS 命令将 QtDesigner 设计出来的默认 ui 文件（包含类 css 布局设计）编译成 py 文件。

图 3.3.1 小区信息界面

3.4 账户、密码验证

为了保障物业管理系统的安全性，我们对管理员和用户分别设置了不同种类的账户，当用户在输入账户和密码之后，系统将会通过 on_clicked 函数判断账户是否存在，账户不存在则会弹出“Wrong! Please try again.”那么如果账户

正确，系统将会开始验证密码，管理员密码为固定的“admin”，这里直接采用了字符串验证；但是不同用户的账户和密码肯定是不同的，考虑到这一点，我们采用了遍历的方法来验证用户的密码。具体实现如下：

```
bool login::UserLogin()
{
    for (int i=0;i<7;i++)
        if (account[i][0] == ui.username->text() && account[i][1] == ui.password->text())
        {
            UserID = ui.username->text();
            return true;
        }
    return false;
}
```

UserLogin 函数将在 on_clicked 函数中被调用，用以判断用户的账户密码是否正确，从而实现账户密码验证。

3.5 页面切换

页面切换的核心是 connect 函数，该函数是实现某个按钮被点击时，根据其相应的所指页面实现切换。当然，要实现登入界面到管理员或者用户系统的切换，首先要实现的就是判断使用者是管理员还是用户，这里我们用到了 useType 函数，可以在用户点击不同的按钮登录时判断用户的类型，具体实现如下：

```
void login::userType()
{
    if (ui.radioButtonHousehold->isChecked())
    {
        usertype = false;
    }
    if (ui.radioButtonManager->isChecked())
    {
        usertype = true;
    }
}
```

该函数在 connect 函数中被调用，方便实现相关页面的切换。

登入界面与管理员界面或用户界面的切换：

前文已经说到 connect 函数是实现页面切换所必要的，connect 函数的具体实现如下。

```
ui.setupUi(this);
connect(ui.pushButtonLogin, SIGNAL(clicked()), this, SLOT(on_clicked()));
connect(ui.pushButtonCancel, SIGNAL(clicked()), this, SLOT(close()));
connect(ui.radioButtonManager, SIGNAL(toggled(bool)), this, SLOT(userType()));
connect(ui.radioButtonHousehold, SIGNAL(toggled(bool)), this, SLOT(userType()));
ui.radioButtonManager->setChecked(false);
ui.radioButtonHousehold->setChecked(false);
```

图中分别有四行 connect 函数的相关代码，第一行实现的是登录按钮，它与第三行和第四行的代码紧密联系，当点击登录按钮时，系统就会自动判断账户的类型，密码的正确与否，在系统中分别对应着 on_clicked 函数和 userType 函数；第二行代码实现的是关闭页面，当用户点击退出时系统会结束运行并退出。值得注意的是我们没有将用户界面和管理员界面之间实现切换，因为这在实际中是很少出现管理员和用户的切换的，所以在系统中我们没有将其考虑在内。

3.6 信息的修改与存储

在相应页面的信息的修改我们通过 write 函数来实现，而对应地我们运用了 save 函数来将其保存至 info 表单中，不同的信息我们分配了不同的 sheet，总共对应了 6 个不同的 Excel 的 sheet；

相应的 wirte 函数和 save 函数修改与保存实现如下（此处的 wirte 是指写入表单中）

以小区信息为例：

```
Document doc("Data.xlsx");
doc.selectSheet("小区信息");
doc.write(2,1,ui.lineEditCommunityName->text());
doc.write(2,2,ui.lineEditCommunityLocation->text());
doc.save();
```

此处首先选择了小区信息的相应的 sheet，其次通过 wirte 函数来修改表单中相应的信息，最后 save 函数确保信息被存储进 sheet 中，实现相关信息的修改和保存。其他类型的信息修改和保存的实现都与之大致相似。

3.7 前后端交互（接口）

接口描述了类的行为和功能，而不需要完成类的特定实现。C++的接口是使用抽象类来实现的，抽象类与数据抽象互不混淆，数据抽象是一个把实现细节与

相关的数据分离开的概念。

```
connect(ui.buttonEditCommunityInfo, SIGNAL(clicked()), this, SLOT(EditCommunityInfo()));
connect(ui.buttonSaveCommunityInfo, SIGNAL(clicked()), this, SLOT(SaveCommunityInfo()));
connect(ui.buttonEditHouseholdInfo,SIGNAL(clicked()),this,SLOT(EditHouseholdInfo()));
connect(ui.buttonSaveHouseholdInfo,SIGNAL(clicked()),this,SLOT(SaveHouseholdInfo()));
connect(ui.buttonOutHouseholdInfo,SIGNAL(clicked()),this,SLOT(PrintHouseholdInfo()));
connect(ui.buttonChargeDo, SIGNAL(clicked()), this, SLOT(DoCharge()));
connect(ui.buttonChargeCancel, SIGNAL(clicked()), this, SLOT(CancelCharge()));
connect(ui.buttonEditParkingInfo,SIGNAL(clicked()),this,SLOT(EditParkingInfo()));
connect(ui.buttonSaveParkingInfo,SIGNAL(clicked()),this,SLOT(SaveParkingInfo()));
connect(ui.pushButtonDoing,SIGNAL(clicked()),this,SLOT(ChangeFeedbackInfo()));
connect(ui.pushButtonDone,SIGNAL(clicked()),this,SLOT(FeedbackInfoDone()));
connect(ui.pushButtonDoing2,SIGNAL(clicked()),this,SLOT(ChangeFeedbackInfo2()));
connect(ui.pushButtonDone2,SIGNAL(clicked()),this,SLOT(FeedbackInfoDone2()));
connect(ui.pushButtonSwitchRoomType,SIGNAL(clicked()),this,SLOT(SwitchRoomType()));
connect(ui.buttonDeleteHousehold,SIGNAL(clicked()),this,SLOT(DeleteHousehold()));
connect(ui.tableWidgetHouseholdInfo->horizontalHeader(),SIGNAL(sectionClicked(int)),this,SLOT(slot_sortByColumn(int)));
connect(ui.buttonOutputAll,SIGNAL(clicked()),this,SLOT(OutputAll()));
//读取excel文件
```

图 3.7.1 pmis.cpp 文件中的接口

```
connect(ui.pushButtonLogin, SIGNAL(clicked()), this, SLOT(on_clicked()));
connect(ui.pushButtonCancel, SIGNAL(clicked()), this, SLOT(close()));
connect(ui.radioButtonManager, SIGNAL(toggled(bool)), this, SLOT(userType()));
connect(ui.radioButtonHousehold, SIGNAL(toggled(bool)), this, SLOT(userType()));
```

图 3.7.2 login.cpp 文件中的接口

```
connect(ui->pushButtonRup,SIGNAL(clicked()),this,SLOT(UserRup()));
connect(ui->pushButtonCup,SIGNAL(clicked()),this,SLOT(UserCup()));
connect(ui->pushButtonRclear,SIGNAL(clicked()),this,SLOT(UserRclear()));
connect(ui->pushButtonCclear,SIGNAL(clicked()),this,SLOT(UserCclear()));
```

图 3.7.3 user.cpp 文件中的接口

将动作、函数与对应的按钮名称连接起来，实现前后端交互设计。

3.8 数据库

读取数据库文件：

```
QXlsx::Document xlsx;
Document doc("Data.xlsx");
```

数据读取（以物业收费功能为例）：

```
doc.selectSheet("收费信息");
QVariant ChargeNum = doc.read(1,2);
ui.dateEditChargeTime->setDisplayFormat("yyyy.MM.dd");
ui.dateEditChargeTime->setDate(QDateTime::currentDateTime().date());
ui.dateEditChargeTime->setCalendarPopup(true);
ShowChargeInfo();
```

数据更改（以小区信息功能为例）：

```
QXlsx::Document xlsx;
Document doc("Data.xlsx");
doc.selectSheet("小区信息");
doc.write(2,1,ui.lineEditCommunityName->text());
doc.write(2,2,ui.lineEditCommunityLocation->text());
doc.save();
ui.lineEditCommunityName->setFrame(false);
ui.lineEditCommunityLocation->setFrame(false);
ui.lineEditCommunityName->setReadOnly(true);
ui.lineEditCommunityLocation->setReadOnly(true);
```

利用行列式表格高效的存储数据，excel 可以对数据进行排序、筛选、分类汇总、数据验证、条件格式等操作，同时可以导出 excel 文件，生动展示数据。

QXlsx 是一个可以读写 Excel 文件的库。它不需要 Microsoft Excel，可以在 Qt5 支持的任何平台上使用。它具有的功能：创新新的 xlsx 文件、从.xlsx 文件中提取数据、编辑现有的.xlsx 文件。

四、开发心得

4.1 小组分工

常元和：设计系统功能；QXlsx 数据库编写；部分后台逻辑编写（管理员界面）

汪家伟：需求分析、系统总体结构设计；Qt 前端编写；选题报告、实验报告撰写

袁嘉骏：系统的调试、优化；部分后台逻辑编写（登陆界面、用户界面）；实验报告撰写

4.2 收获和体会

4.2.1 软件开发方面

1、熟悉使用 Git 进行版本控制的一般流程；

2、学会用 Github 进行协作开发，为以后的项目开发积累经验；

3、初步了解并实现前后端交互设计，界面可视化的初步尝试；

4、项目完成度较高，深入整个软件开发的流程。

4.2.2 C++语言方面

1、练习了 C++的基本语法和一般设计规范；

2、提供了标准化接口、通过抽象类描述类的行为和功能；

3、尝试连接 Qt 提供的开放库 QXlsx 数据库，使用 excel 文件处理信息；

4.2.3 程序设计方面

1、从需求出发设计程序功能，观照系统的总体功能设计；

2、初步了解、实现一些设计模式；

3、认识到了面向对象范式的好处和不足；

4、认识到了调试、优化程序的重要性。