“C++程序设计与训练”课程大作业

项目报告

项目名称：超市商品管理系统

姓名： 刘柏

学号： 2013011548

班级： 自 36

日期： 2014 年 9 月

目　录

1 系统功能设计

1.1 概述

实现了大作业要求中的**全部基本功能**，还实现了“用户管理系统”、“权限管理系统”、“PDF 导出”等**拓展功能**。

1.2 功能点（表格）

功能类型	实现的功能点	实现方式 **1） 自己编写 C# 代码** **2） 使用 C#标准库** **3） 使用第三方库** **4） 使用 SQL 语句**	备注
初级功能	信息的读写	自己编写 C#代码	所有数据均被全部读到相应的类对象中，在内存变量中完成处理过程，得到最终结果后才存储至数据库或文件。 **没有**在数据库中**直接操作**相应数据。
	信息增加	自己编写 C#代码	点击“操作”——“管理商品”，选择“新建商品（进货）”选项卡。 能根据用户选择的一级类别、二级类别来**动态确定**“信息增加”操作界面。
	信息删除	自己编写 C#代码	点击“操作”——“管理商品”，选择“删除商品”选项卡。
基本功能	状态变更	自己编写 C#代码	点击“操作”——“管理商品”，选择“修改商品信息（上架/销售/退货）”选项卡。 1、 实现了对商品状态信息的修改； 2、 大大拓展了可修改范围，让用户能够修改商品名称、编码等基本信息。
	记录退回原因	自己编写 C#代码	状态为“退回”时会**强制用户输入退货原因**。
	简单查询	自己编写 C#代码	点击“查询”——“单条商品信息查询”。
	数据可存储至文件或数据库	使用 C#标准库(用以连接数据库)	使用 Access 数据库。
	容错特性	自己编写 C#代码	考虑到了每一个非法操作可能会引发的问题。

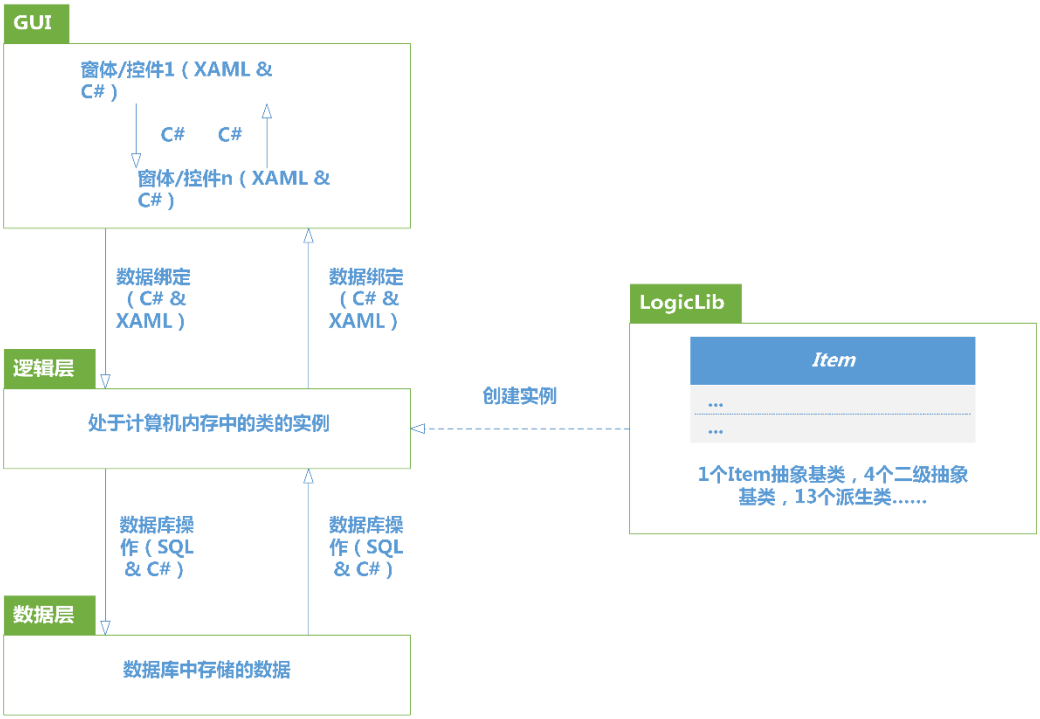
			比如说某项关键信息输入为空、商品编码中掺杂了特殊字符、生成统计图表时未选择需要哪一种图表、**商品状态为“退货”却未输入退货原因**等**数十种**非法操作。 **每次出现非法操作时，都会弹出提示窗口并终止非法操作**。
数据库支持	数据库读写	使用 SQL 语句；	**只使用**了 INSERT、SELECT（未使用 ORDER BY 等）、UPDATE、DELETE 四种**基本 SQL 语句**。 排序操作全部在类的方法中完成。
复杂查询功能	排序	自己编写 C#代码	点击“查询”——“商品数量查询”。 或：点击“查询”——“商品库存/上架时间查询”。 这两个功能中的结果呈现均用到了排序。
	查询指定年、月入库的商品	自己编写 C#代码	点击“查询”——“商品入库时间查询”。
统计功能	统计四个季度的销售情况并制图	使用第三方库（ComponentOne）	单击菜单栏中的“统计”，选择对应统计功能。 由于该第三方库的正版至少要 400 多美元，我们用的是其试用版。 使用时会弹出提示窗口，点提示击“OK”或直接关闭弹窗即可，不影响统计功能的正常使用。
	统计四个大类的商品的销售情况并制图	使用第三方库（ComponentOne）	
	统计销量最大的六种商品并制图	使用第三方库（ComponentOne）	
其它功能	用户登录与验证	自己编写 C#代码；使用 SQL 语句	由于超市管理系统属于商用产品，封闭度较高，故不实行开放注册的机制。取而代之的是管理者在员工管理系统中创建账号并进行授权、分发的模式（参考清华大学信息门户新生密码封的机制）。 **默认管理员工号：0003，密码：121212**
	密码修改	自己编写 C#代码；使用 SQL 语句	点击“系统”——“修改密码”
	新建用户资料	自己编写 C#代码	点击“操作”——“管理员工”，选择“新建员工资料”选项卡。 我们参考了 INFO 的模式，先由管理人员创建好账号、设定初始密码，再将工号和初始密码告知用户，让用户自行修改密码，**故在此处为了方便管理人员确定初始密码，未采用掩码。**
	修改用户资料	自己编写 C#代码	点击“操作”——“管理员工”，选择

			“修改员工资料”选项卡。
	删除用户资料	自己编写 C#代码	点击“操作”——“管理员工”，选择“删除员工资料”选项卡。
	设定用户权限	自己编写 C#代码	点击“操作”——“管理员工”，选择“新建员工资料”选项卡。 或：点击“操作”——“管理员工”，选择“修改员工资料”选项卡。 在里面选择对应的权限即可。
	导出到 PDF	使用第三方库（iText）	点击“查询”——“商品入库时间查询”。 选择年份、月份后单击“查询”按钮即可（未选择时间时会弹出提示并终止非法操作）。 单击“导出到 PDF”并选择路径即可导出 PDF 文件。
	信息栏	自己编写 C#代码	**页面左侧**的信息栏通过显示当前登录用户信息、当前在软件中所处位置、系统时间，**方便用户了解当前系统状态**。

2 系统总体结构

2.1 概要设计

现行的软件设计思路推荐将 UI 层与逻辑层分离开来，这正是 WPF 技术诞生的原因。这次大作业的系统也采用了这种思路。如下是系统的总体结构：



2.1.1 GUI 层

使用 XAML 语言与 C# 制作界面外观，并使用 C# 完成窗口间、控件间的信息交互。

2.1.2 逻辑层

处于内存中的类的实例、结构体，以及实例、结构体的数组或动态数组。它们是排序等算法操作的对象。

2.1.3 系统与数据库的交互

考虑到系统的数据规模与兼容性等问题，我们选择 Access 作为我们的数据库。

考虑到我们的作品有两个比较独立的需要与数据库交互的模块（商品管理系统、员工管理系统），我们设计了两个数据表来储存信息：Item 数据表和 Stuff 数据表；设计了两个类来实现对数据库的操作：ItemAccess 类和 StuffAccess 类。

数据表设计附图如下：

Item

字段名称	数据类型
ID	自动编号
FirstCategory	短文本
SecondCategory	短文本
Code	短文本
ItemName	短文本
Condition	数字
BuyingPrice	数字
SellingPrice	数字
Quarter	数字
Remark	短文本
FirstSubClassA	短文本
FirstSubClassB	短文本
SecondSubClassA	短文本
SecondSubClassB	短文本
ManipulateTime	短文本

Stuff

字段名称	数据类型
ID	自动编号
StuffNumber	短文本
StuffName	短文本
StuffPosition	短文本
StuffPassword	短文本

2.1.4 LogicLib

解决方案中的一个类库项目，包括 1 个为所有商品提供抽象基类的 Item 类，4 个分别为电器、百货、食品、书籍提供抽象基类的二级父类，以及 13 个最派生类。此外还有 1 个员工类、2 个枚举。

2.2 小组分工

小组成员姓名	小组成员班级	小组成员学号	小组成员分工
刘柏	自 36	2013011548	**前端：** 1、 设计客户端整体功能结构； 2、 选择界面配色方案。 **逻辑层：** 1、 实现系统容错功能，为每一处数据输入接口添加容错性检查功能； 2、 设计并实现了 ItemAccess 类，用于操作 Item 数据库； 3、 设计并实现了 StuffAccess 类，用于操作 Stuff 数据库； 4、 完成“复杂功能查询”中的排序； 5、 完成“操作”模块中从前端获取用户输入的数据并存入对象、在数据库中存储或更新或删除数据、在逻辑层处理数据、在前端显示数据的功能； 6、 完成“查询”模块中从数据库读取数据、在逻辑层的类中处理数据、在前端显示数据的功能； 7、 完成“统计”模块中从数据库读取数据、在逻辑层的类中处理数据、在前端显示数据的功能；； 8、 完成用户登录验证的功能； **后端：** 1、 建立 Market 数据库，设计并创建 Item 和 Stuff 数据表； 2、 导入原始数据并增添操作时间等新属性； 3、 建立系统与数据库的连接，将数据库的操作进行封

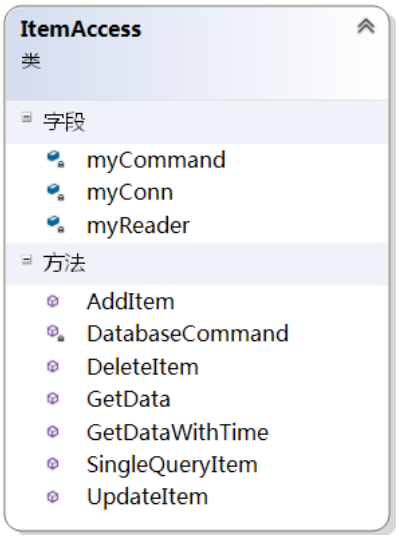
			装，实现系统和数据库的相对独立。 **其它：** 1、编写用户手册。
张为先	自 36	2013011542	**前端：** 1、 完成 GUI 的绘制与交互逻辑； 2、 完成窗后或控件之间的数据传递； **逻辑层：** 1、 将各种商品抽象成类，建立了 1 个 Item 抽象基类、4 个一级分类、13 个二级分类的继承体系； 2、 为每个类添加字段、属性、方法（即建立了系统结构图中的 LogicLib 项目）； 3、 通过数据绑定（Binding）实现 GUI 与内存中对象的信息交互； 4、 利用第三方控件（ComponentOne）实现绘制统计图的功能； 5、 利用第三方程序集实现导出 PDF 文件的功能； 6、 设计实现了操作权限系统； **后端：** 1、 实现修改密码功能； **其它：** 1、 提出总体架构设计； 2、 使用 Installshield 完成程序的打包、发布。

3 本人工作内容

3.1 设计并实现操作数据库的类

3.1.1 概述

为了实现对数据库的操作，我设计了两个类：ItemAccess 和 StuffAccess。前者用于操作数据库中的 Item 数据表，后者用于操作数据库中的 Stuff 数据表。由于二者的设计理念和实现具有高度相似性，接下来仅以 ItemAccess 类为例。



3.1.2 关注封装的设计理念

在设计之初，我就遇到了一个问题：每一种操作，不论是插入数据，还是查询、更新和删除数据，都需要连接数据库并打开。假如说在每一种操作数据库的成员函数里都加上连接数据库的语句，会增加不少代码量，且一个一个修改起来也颇为麻烦。

所以我的设想是应该有一个专门的函数来执行连接数据库的指令。与此同时，我又注意到，对数据表中数据的操作都是由类似于 myCommand = new OleDbCommand(commandString, myConn); 的语句执行的，不同的功能实现只需要改变相应的 commandString 即可。

这让我萌发了一个想法：将直接操作数据库的部分全部封装在 DatabaseCommand 函数里，每次调用这个函数时，都会**打开数据库，并且根据传入的 commandString 参数执行相关操作**。这样子**其它函数只需要负责生成相应的 commandString 并调用 DatabaseCommand 函数**即可，大大减少了代码量。

3.1.3 DatabaseCommand 方法的设计

该函数属于特色函数，但其原理非常简单：先打开指定的数据库，再将 commandString 作为参数初始化 OleDbCommand 类的一个对象即可完成操作。与此同时，还利用了课上所学的异常处理的方法，当出现异常时会弹出相关信息并终止非法操作。

值得注意的是，考虑到**该函数**不会在类体外被直接调用，其**被设置成 private 成员**，这样就**相对彻底地防止了数据库在外部被破坏**。

3.1.4 其它方法的设计

DatabaseCommand 方法之外，还有 AddItem，UpdateItem 等方法，其作用为根据传入的

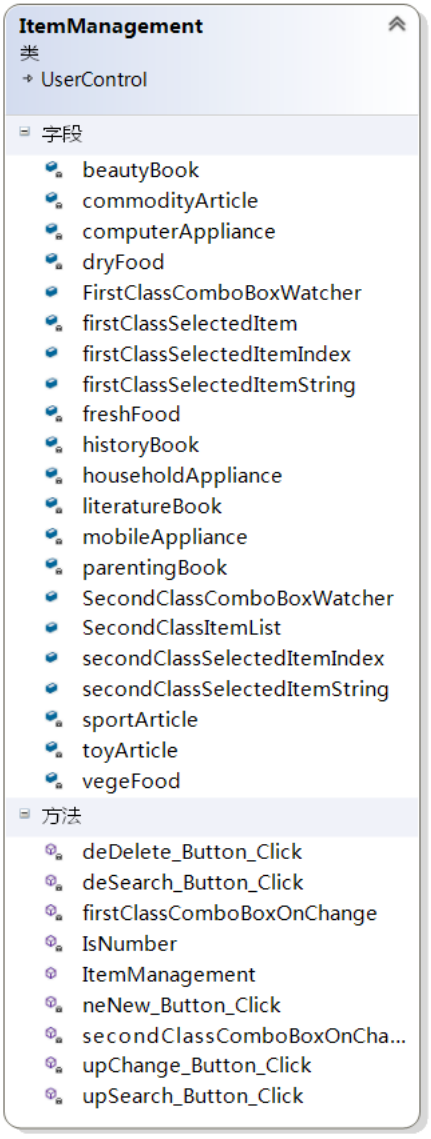
参数生成 commandString，调用 DatabaseCommand 函数获得需要的数据，之后返回。

3.2 实现前端控件与数据交互的类

3.2.1 概述

数据的处理和显示的流程图大致如下图所示，逻辑层的类起到了实现前端控件与数据交互的作用，其重要性不言而喻。

我实现了两个类：ItemManagement 和 StuffManagement。前者用于实现 Item 类数据与控件的交互，后者用于 Stuff 类与控件的交互。由于二者的设计理念和实现具有高度相似性，接下来仅以 ItemManagement 类为例。



3.2.2 获取前端控件中的数据

张为先实现了双向数据绑定，因此，控件中的数据与类的属性实现了同步变化，即在控件中改变了的数据能够实时反映在类的属性中。因此，为了获得前端控件中的数据，只需**调用对应的商品类中的方法**即可。

3.2.3 将数据显示在前端控件上

将数据显示在前端控件的机制更为简单：在界面的 xaml 文件对应的 xaml.cs 文件中为相应的控件的属性赋值即可，例如：upBuyingPrice_TextBox.Text = queryResult[6];。

3.3 实现容错功能

3.3.1 概述

在我们设计的过程中，容错特性始终具有极为重要的地位。

我们考虑到了每一个非法操作可能会引发的问题。比如说某项关键信息输入为空、商品编码中掺杂了特殊字符、生成统计图表时未选择需要哪一种图表、商品状态为“退货”却未输入退货原因等**数十种非法操作**。

每次出现非法操作时，都会弹出提示窗口并终止非法操作。

3.3.2 非法操作的分类

我们将非法操作的类型分为了四类：

输入的信息为空（比如说在新建商品信息时用户漏添加商品编号、商品名称等关键信息，或在执行查询操作时忘记选择查询条件）；

输入的关键信息与数据库中已有数据重复（比如说输入的商品的唯一标识符——商品编码在库中已经存在）；

输入信息不合法（比如说在应该输入数字的文本框中输入了特殊字符）；

查询结果为空（假如说继续运行的话一方面占用程序资源，另一方面容易产生“空引用”等问题）。

3.3.3 终止操作

从源代码就可看出，在每一处有输入操作的地方，均有非法操作的检测接口。每当检测

到非法操作时，就会弹出提示窗口，并利用 return 语句结束当前函数的操作。

3.4 设计算法

3.4.1 概述

其实就本项目而言，并不需要用到比较复杂的算法，可能难点主要集中在类的设计、实现与管理上。因此，整个项目中，涉及到算法的问题不是太多，我所完成的比较主要的算法都位于“复杂查询”的功能模块中，主要是“统计同名商品的数目”和“索引法排序”两种算法。

3.4.2 统计同名商品的数目

在实现“商品数量查询”功能时，我们遇到了一个问题：在数据库里面，不同编码的商品对应一条数据记录，同一名称的商品可能有很多条记录。因此，仅仅读取数据库中的数据，是无法确定同名商品的数量的，必须在逻辑层里面进行处理。

为此，我先获取原始数据并存到数组中（记为“原始数组”），之后遍历每一项元素。若原始数组之前尚未出现过具有同商品名称的元素，则将该元素的信息保存到一个新数组（记为“处理数组”）中；若原始数组之前已经出现过具有同商品名称的元素， 则将处理数组中该项商品名称对应的“数量”属性加一。这样处理过后，得到的就是包含有同名商品数量的信息的数据。

3.4.3 索引法排序

我们从数据库中读取数据后，将数据存储在了数组之中。数组元素的信息量比较大，假如说通过改变数组元素存储顺序的方式来排序，就会牵涉到对元素地址的操作，工作量较大。况且，我们所要的效果是数据能以有序的方式呈现，只要输出效果达到要求即可，不须改变数组元素实际的顺序。

为此，联想到在 C 语言课程上了解到的排序算法，我决定采用索引法。核心就是保持数组元素存储顺序不变，获得最大（或最小）的前 N 项元素的下标，并将这些下标值存储到一个新的数组中。

这个算法的实现并不复杂：新创建一个 orderArray 数组，为第 i 项赋初值 i，然后在传统的排序方法中，将下标由[j]改为[orderArray[i]]，并将原来交换元素值的语句更改为交换 orderArray 对应元素。

4 项目总结

4.1 分工明确的重要性

4.1.1 前期分工

在设计项目的时候，我们将项目大致分为三个模块：前端（图形界面）、逻辑层、后端（数据库）。

经过分析，我们发现，前端和后端的独立性比较强，而逻辑层与二者关系比较密切。因此，在项目的初期阶段，为了提高开发效率，我们决定由张为先开发前端，由我开发后端，与此同时，两人需要分别编写自己负责的模块在逻辑层的接口，方便日后整合上去。

事实证明，这种分工方式是十分有效的。在初期阶段我们的工作互不干扰，双方都可以自由地设计自己负责的模块，而当需要整合时，只需要在逻辑层使用接口即可。

4.1.2 后期分工

当项目进展到了后期阶段，主体的框架都已搭建得比较完善，此时对不同模块的开发已经很难实现完全的分离，我们的工作呈现出交叉的特性。

为此，我们改变了分工的思路：双方都可以对整个项目进行修改，不局限于自己原本负责的模块。例如：我在后期对界面中按钮的配色作了调整、张为先在后期添加了修改密码的功能。

经过实践，这一做法在后期阶段是比较合适的，我们可以比较自由地修改自己发现的小问题而不必交由对方，同时，我们也接触到了对方所涉及的技术领域，开拓了视野，锻炼了能力。

4.2 用户体验的改良

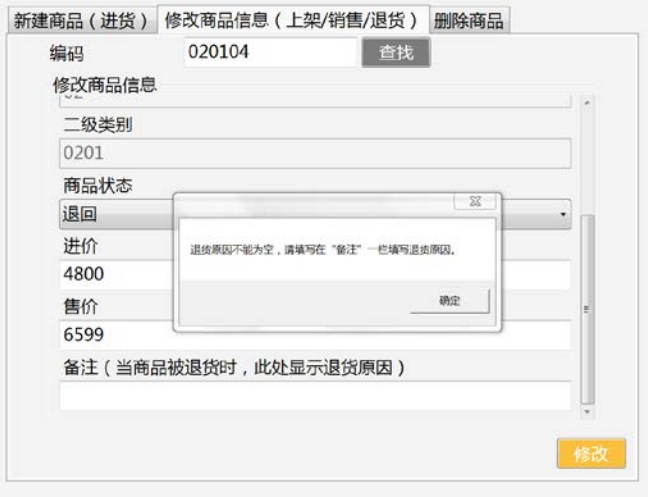
4.2.1 容错性

由于本系统涉及到较多的用户信息输入的操作，而且加入有非法操作很可能导致程序崩溃，因此改善系统的容错性对提升用户体验而言至关重要。

在设计的过程中，我们体会到了设计容错性的复杂程度。首先就是要界定什么类型的操作属于非法操作，我们原先只考虑了用户输入为空这一种情况，后来发现，商品编码不唯一（在单条查询时会出错）、价格的输入带有特殊字符（无法输入到不符合数据库中规定的格式）等情况同样需要考虑。

当检测到非法操作后，怎样终止非法操作也是一个问题。我们在尝试了很多方法后，考虑到所有操作都是在类的方法中完成的，最终决定采用 return 语句，在检测到异常后终止函数的运行。

在提升系统容错性的进程中，我们对人机交互的理念有了更深刻的理解不仅训练了逻辑思维与编程能力，更对人机交互的机制有了更深刻的理解。



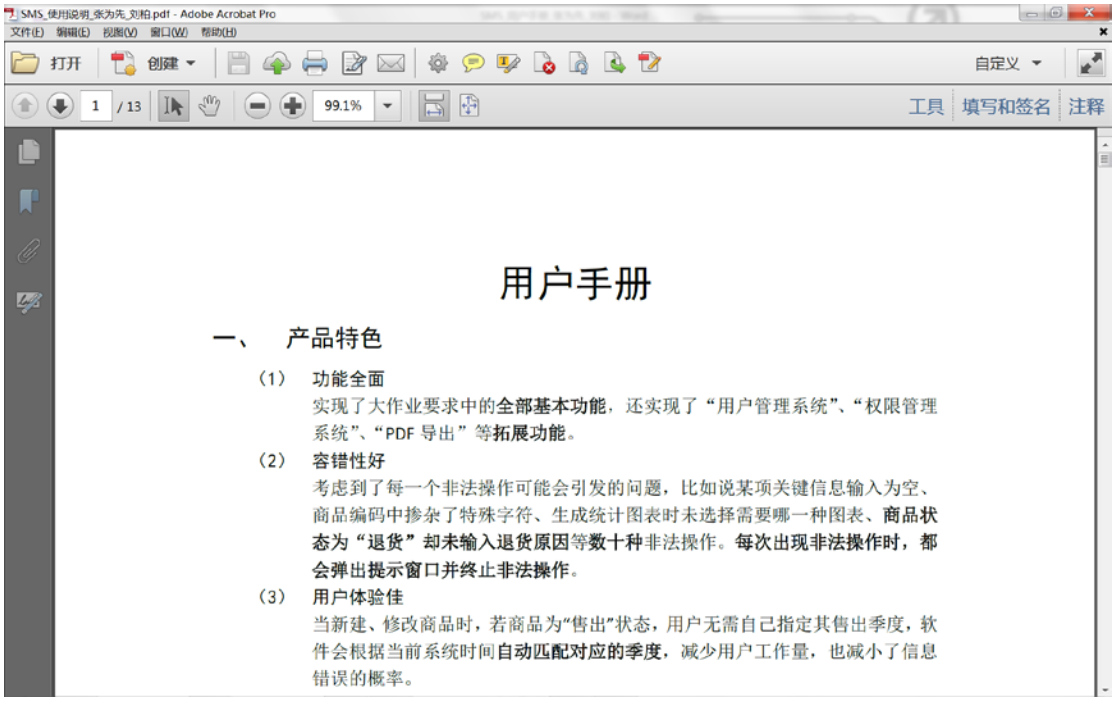
4.2.2 操作信息的显示

在进行设计的过程中我们发现了一个问题：由于一些功能的实现界面比较相似，很多用户进入到相应界面后往往不能很方便地获知当前究竟处在哪个功能区。为此，我们在窗口左侧特别增设了信息栏，显示当前用户资料、用户位置、系统时间等资料。制作好之后再使用软件感觉方便了许多。

4.2.3 用户操作的指引

身为开发人员，在开发软件的过程中，我们觉得软件是非常易用的。但在请其他同学作为用户进行测评时，还是会发现不少同学对软件的操作不甚熟练。

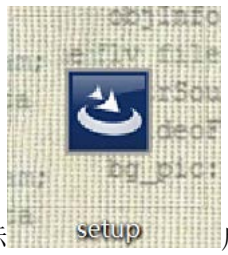
为此，我们编写了用户手册来帮助用户使用本软件。


SMS_使用说明_余为先_刘柏.pdf - Adobe Acrobat Pro
文件(F) 编辑(E) 视图(V) 窗口(W) 帮助(H)
打开
创建
自定义
1 / 13
99.1%
工具
填写和签名
注释
用户手册
一、 产品特色
（1） 功能全面
实现了大作业要求中的全部基本功能，还实现了“用户管理系统”、“权限管理系统”、“PDF 导出”等拓展功能。
（2） 容错性好
考虑到了每一个非法操作可能会引发的问题，比如说某项关键信息输入为空、商品编码中掺杂了特殊字符、生成统计图表时未选择需要哪一种图表、商品状态为“退货”却未输入退货原因等数十种非法操作。每次出现非法操作时，都会弹出提示窗口并终止非法操作。
（3） 用户体验佳
当新建、修改商品时，若商品为“售出”状态，用户无需自己指定其售出季度，软件会根据当前系统时间自动匹配对应的季度，减少用户工作量，也减小了信息错误的概率。

5 相关问题的说明

5.1 安装与登录

5.1.1 系统安装

将压缩包解压，点击该图标 

后，按照提示进行操作即可。

注：

1. *安装要求 .Net Framework 4.5（若无，请联系我们：18810917624，我们会将.Net 安装包发过去）*
2. *安装完成后会在桌面创建快捷方式*
3. *可在控制面板中卸载本程序*

5.2.1 登录系统

由于超市管理系统属于商用产品，封闭度较高，故不实行开放注册的机制。取而代之的是管理者在员工管理系统中创建账号并进行授权、分发的模式。

默认管理员工号：0003，密码：121212

5.2 关于图表控件的弹窗问题

由于图表控件的正版至少要 400 多美元，我们用的是其试用版，会弹出如下的窗口，点击“OK”或直接关闭弹窗即可，不影响统计功能的正常使用。


About C1.WPF.C1Chart
ComponentOne
a division of GrapeCity
C1.WPF.C1Chart
Evaluation version.
This dialog will not be shown if you rebuild your application using a
Purchase
Activate
Register
Online Resources:
Download Center
Contact Us
Web Store
Technical Support
Resellers
Version History
Submit Incident
This product is part of the ComponentOne Studio Enterprise.
Copyright (c) 2001-2014 GrapeCity, Inc.