“C++ 程序设计与训练”课程大作业

项目报告

项目名称：餐厅服务与管理系统

姓名：赵文亮

学号：2016011452

班级：自 64

日期：2017 年 9 月 22 日

目录

1 系统功能设计 **1**

1.1 总体功能描述 . . . 1

1.2 功能流程描述 . . . 1

1.2.1 登录/注册流程 . . . 1

1.2.2 管理员功能 . . . 1

1.2.3 顾客功能流程 . . . 1

1.2.4 服务员功能 . . . 2

1.2.5 厨师功能 . . . 2

1.2.6 经理功能 . . . 2

2 系统结构设计 **3**

2.1 数据模块 . . . 3

2.1.1 用户类 . . . 3

2.1.2 菜品类 . . . 4

2.1.3 餐桌类 . . . 4

2.1.4 其它数据相关类 . . . 4

2.2 界面模块 . . . 4

2.3 通信模块 . . . 4

3 系统详细设计 **4**

3.1 模式切换设计 . . . 4

3.2 类结构设计 . . . 5

3.2.1 用户类 . . . 5

3.2.2 菜品类 . . . 5

3.2.3 餐桌类 . . . 6

3.2.4 数据库类 . . . 7

3.2.5 消息类 . . . 7

3.2.6 Excel 导出类 . . . 7

3.2.7 Windows 消息发送类 . . . 7

3.2.8 开始界面类 . . . 8

3.2.9 顾客界面类 . . . 8

3.2.10 管理员界面类 . . . 9

3.2.11 服务员界面类 . . . 10

3.2.12 厨师界面类 . . . 11

3.2.13 经理界面类 . . . 11

3.3 界面结构设计 . . . 11

3.3.1 开始界面设计 . . . 11

3.3.2 顾客界面设计 . . . 11

3.3.3 管理员界面设计 . . . 12

3.3.4 厨师界面设计 . . . 13

3.3.5 服务员界面设计 . . . 13

3.3.6 经理界面设计 . . . 13

3.4 关键设计思路 13
3.4.1 容错功能设计 13
3.4.2 消息加密传送 14
3.4.3 多用户实时通信 14
3.4.4 数据处理的封装 14
3.4.5 时钟的绑定 15

4 项目总结 **15**
4.1 项目亮点 15
4.1.1 功能强大，考虑周全 15
4.1.2 界面美观，操作简便 15
4.2 开发难点 15
4.2.1 数据的结构与组织 15
4.2.2 跨进程的通信 15

5 心得体会 **16**
5.1 自学能力 16
5.2 精益求精的精神 16
5.3 调试的收放自如 16

1 系统功能设计

1.1 总体功能描述

本项目实现了一套多用户餐厅服务与管理系统。利用该系统，餐厅的管理员可以方便快捷地管理用户、菜品和餐桌；顾客、服务员、厨师可以实时进行交互并收发消息，共同完成一次就餐流程。经理可以查看服务员与厨师的工作数据，以及菜品的销售数据，并导出为 Excel 数据报表。此外，本项目还实现了多 App 和单 App 两种工作模式，以满足不同的需求[1]。

1.2 功能流程描述

1.2.1 登录/注册流程

程序开始时默认为顾客登录界面。可以通过点击按钮来切换登录和注册状态，通过点击按钮和选择下拉菜单中的用户类别在五种用户间切换（顾客、管理员、厨师、服务员、经理）。

新用户注册时需要输入手机号和密码，并确认密码，两次密码一致时方可注册成功；已有账号的用户通过手机号和密码登录，当密码正确时，可以登录到对应用户类别的界面。

1.2.2 管理员功能

管理员承担着菜单管理、用户管理、餐桌管理三大任务。在所有的用户中拥有最高的权限。

菜单管理 管理员可以管理菜品类别，对类别实施增加、移除、编辑、移动、清空操作，并按类别对菜单实施管理。管理员可以添加、删除、编辑、清空和查找菜品。本项目实现了动态查找，即在搜索框中输入关键字，菜单列表实时显示对应的查询结果。同时，管理员可以在搜索状态下对菜品进行编辑与删除操作，极大地方便了菜品的管理。

用户管理 管理员可以勾选用户类别来对用户进行管理。所有被勾选的类别的用户信息都显示在用户列表中。管理员可以添加、删除、编辑、清空和查找用户。同样，管理员可以动态查找用户并在查找过程中进行编辑或删除。

餐桌管理 管理员可以查看所有餐桌的状态。餐桌状态分为：点菜中、已下单、结账中、已结账、禁用、空闲。前四种状态为顾客占用状态，管理员可以启用或禁用无顾客占用的餐桌。管理员还可以修改餐桌数量[2]。

1.2.3 顾客功能流程

顾客是整个就餐环节的核心，整个流程中，顾客与服务员、厨师之间进行交互，实现消息的收发和状态的更新。整个用餐流程如图 1 所示。

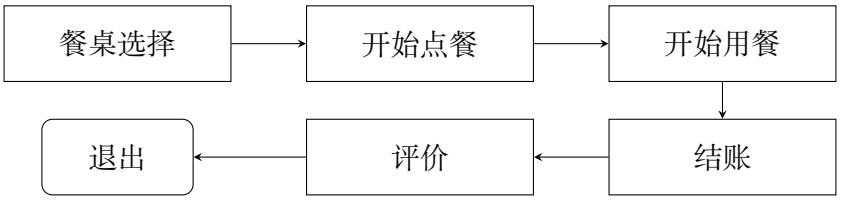

图 1: 顾客用餐流程

[1]详见第 4 页 3.1 节

[2]详见第 7 页 3.2.3 节

餐桌选择　顾客可以实时查看所有餐桌的状态，并选中空闲的餐桌。当顾客选定餐桌并点击提交时，该餐桌状态会变成“点菜中”。同时其它所有处于选桌界面的顾客会立即看到该餐桌状态的变化，服务员的待认领餐桌列表也会更新。

开始点餐　顾客可以看到所有的菜品信息。顾客点击相应的菜品即可将菜品加入菜单右侧的“我的餐盘”，实时查看已点菜品及总金额。顾客可以在餐盘中编辑菜品数量。顾客提交菜单后，已点菜品会立即出现在所有厨师的任务列表中。餐桌状态会更新为“已下单”。

开始用餐　顾客可以查看所有已点菜品的状态。餐品状态分为：等待中、制作中、已完成、已上菜。顾客可以随时给服务员发送消息。用餐结束后，顾客点击“去结帐”按钮可以跳转到结账步骤，餐桌状态更新为“结账中”。

结账　顾客可以查看详细的账单信息。顾客可以选择微信支付或现金支付。顾客点击支付完成后，会自动给服务员发送“结账”消息。服务员解决该消息后，餐桌状态变为“已结账”，同时顾客界面自动跳转到评价界面。

评价　顾客可以为每道菜品打分，为服务该餐桌的服务员打分。点击提交后，评分结果会写入每道菜品对应的厨师和该餐桌服务员的工作数据中。程序退出，餐桌状态变为“空闲”。

1.2.4　服务员功能

服务员的任务是处理顾客发送的信息，协助顾客完成结账，接收厨师发来的菜品就绪的信息以便上菜。界面分为认领餐桌、顾客服务、我的餐桌三个部分。

认领餐桌　所有处于“点菜中”、“已下单”、“结账中”状态且无服务员认领的的餐桌为待认领状态的餐桌。这些餐桌都会出现在服务员的待认领餐桌列表中。

顾客服务　服务员认领的所有餐桌的顾客发送的消息会实时显示在该服务员的消息列表中。同时，当这些餐桌的菜品已完成时，服务员也会受到菜品就绪的消息。服务员处理结束后可点击“已解决”按钮，该消息自动从列表中清除。特别地，如果服务员处理了结账消息，且对应餐桌处于“结账中”状态，该餐桌的状态会变成“已结账”，并且该餐桌的顾客会自动进入到评价环节；如果服务员处理了上菜消息，对应菜品的状态会变为“已上菜”。

我的餐桌　可实时查看该服务员认领的餐桌列表。

1.2.5　厨师功能

厨师可查看所有未完成的菜品，包括未被认领的菜品和被该厨师认领但未完成的菜品。每当有顾客下单时，所有厨师的菜品列表都会更新。新出现的菜品状态均为“等待中”。厨师点击按钮认领某个菜品后，该菜品状态变为“制作中”，并从其它厨师的列表中删除。当厨师点击“已完成”按钮后，对应菜品状态变为“已完成”，并从该厨师的列表中删除，同时向服务员发送菜品就绪的消息。

1.2.6　经理功能

经理可以查看所有厨师和服务员的编号、手机号码和工作情况，以及餐厅的销售数据，并将所有数据导出为 Excel 数据报表。

厨师的工作情况包括基本的工作数据：做菜数量、做菜总时间、顾客评分，以及由此计算而来的做菜平均时间、平均得分；服务员的工作情况包括基本的工作数据：服务总次数、服务总时间、服务桌数、顾客评分，以及由此计算而来的平均服务时间、平均得分。经理可以通过这些数据来综合考查厨师和服务员的工作量、工作质量和工作效率。餐厅销售数据包括每种菜品的基本信息（类别、名称、单价）和销量（售出总数、销售额）以及该餐厅的营业额。

2 系统结构设计

2.1 数据模块

整个系统的数据主要分为三种：用户数据、菜品数据、餐桌数据。每种数据都需要有相关的类进行管理。本项目使用 SQLite 数据库实现数据的持久化，在多 App 模式下，数据库还承担着多用户之间共享数据的任务。

2.1.1 用户类

用户类型一共分为五种。定义用户类别为 0-4 的整数，从小到大依次为：顾客、管理员、厨师、服务员、经理。users.h 中定义了静态的 QString 数组，存储每种类型的用户对应的名称。不同类型的用户需要包含的数据不尽相同，如表 1 所示。

用户类型	数据
顾客	编号、手机号、密码、餐桌号
管理员	编号、手机号、密码
厨师	编号、手机号、密码、做菜数量、总分数、总时间
服务员	编号、手机号、密码、服务次数、服务桌数、总分数、总时间
经理	编号、手机号、密码

表 1: 每种类型用户的数据

用户的基本信息为编号、手机号、密码。定义 User 类为一般的用户类，包含用户的基本信息。管理员和经理可直接定义为 User 类的对象。顾客类（Customer）、厨师类（Cook）、服务员类（Waiter）分别从 User 类继承，并添加各自独有的数据成员。此外，User 类中还应该添加标识用户类别的数据成员。所有用户类的 UML 图如图 2 所示，其中省略了部分成员函数。

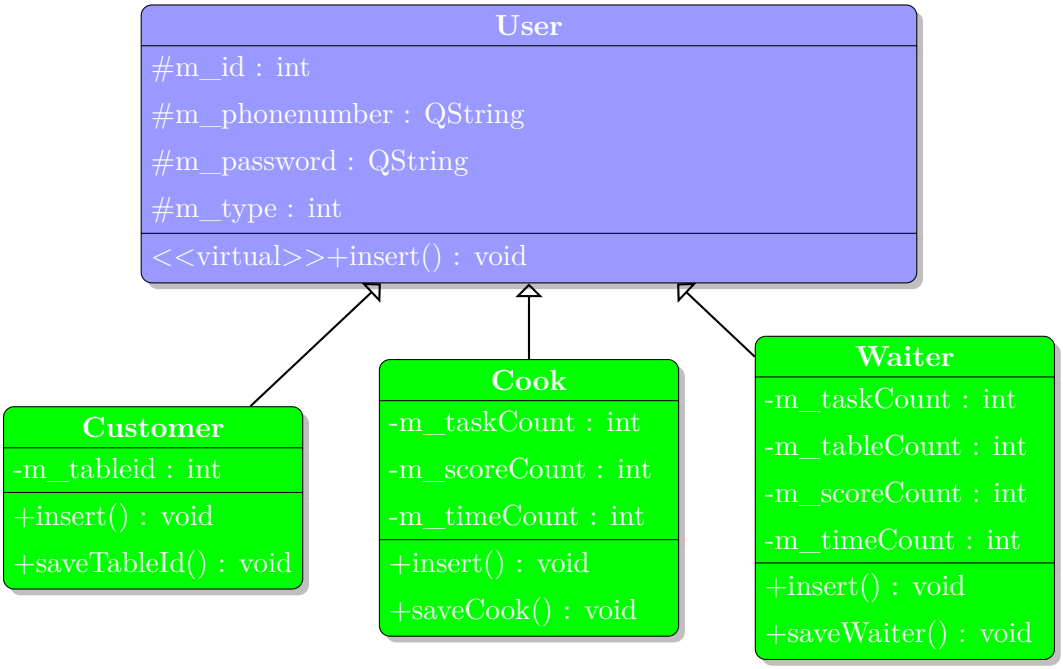

图 2: 用户类 UML 图

本程序使用群体类 Users 来方便用户的统一管理。Users 类中具有私有成员 m_Users，类型为

QVector<User*>, 是包含多个 User 指针的动态数组。此处使用指针存储是为了便于使用多态技术, 简化函数的调用。Users 类中实现了管理用户对象的常用方法。

2.1.2 菜品类

菜品类 Dish 可以用来管理菜品的数据：菜品编号、类别、名称、价格、描述、数量、状态、桌号、厨师编号。菜品类的对象可以是菜单中的某一道菜品, 也可以是顾客所点的菜品。对于菜单中的菜品，桌号定义为 0。对于顾客点的菜品，桌号为顾客所选桌号，通过菜品的对象可以得到对应的桌号, 进而得到对应的餐桌；也可以通过厨师编号查找到认领该菜品任务的厨师。为了方便菜品的统一管理, 本程序使用群体类 Dishes 来管理菜品，Dishes 中具有私有成员 m_Dishes, 类型为 QVector<Dish>。Dishes 类中实现了管理菜品对象的常用方法。

2.1.3 餐桌类

餐桌类 DiningTable 用来管理餐桌数据：餐桌编号、顾客消息、服务员编号、餐桌状态。可以通过服务员编号找到认领该餐桌的服务员。使用群体类 DiningTables 实现了餐桌的统一管理。DiningTables 类中的私有成员 m_tables 是 QVector<DiningTable> 类型的动态数组。

2.1.4 其它数据相关类

- 消息类 Message：用来封装顾客消息，包含餐桌号和消息内容两个数据。服务员界面会使用消息类管理消息。
- 数据库类 DataBase：封装了 SQLite 数据库的常用操作，简化了程序中对数据库的使用。
- Excel 导出类 ExportExcel：封装了对 Excel 的基本操作，用于在经理界面导出数据报表。

2.2 界面模块

本项目的界面部分主要由六个类构成，即开始界面和五种类型用户的界面。开始界面可以通过选择用户类别来进行不同类别用户的登录和注册。

2.3 通信模块

本项目实现了多 App 和单 App 两种模式的多用户系统。在多用户之间交互尤其是多 App 模式下，信息的同步是至关重要的。本项目采用 Windows 消息实现了多进程的通信。Windows 消息发送类（WMSender）封装了发送消息的基本操作，并具有用来解码消息的静态方法。

3 系统详细设计

3.1 模式切换设计

本项目具有两种操作模式。在多 App 操作模式下，用户登录之后即进入对应的界面。不同的用户需要打开多个 App 登录, 通过数据库实现必要的数据共享; 在单 App 操作模式下，用户登录之后登录界面不会关闭，从而可以在原登录界面继续用不同身份登录。单 App 操作模式开启时，提取数据库内的数据到内存；运行期间不使用数据库，所有数据通过静态变量管理；程序关闭时在将内存中的当前数据保存到数据库。两种模式通过开始界面的隐藏按钮实现切换[3]。两种模式运行期间提取和保存数据

[3]详见第 11 页 3.3.1 节和第 8 页 3.2.8 节

的方式不同。单 App 模式下，从静态变量提取数据，保存到静态变量中；多 App 模式则从数据库提取，保存到数据库中。所有数据模块的提取和保存操作都进行了模式判断。

3.2 类结构设计

3.2.1 用户类

User 包含编号、手机号、密码、用户类型四个保护数据成员，提供了基本的查看和修改信息的公有接口。User 类中具有虚函数 insert，用于将该对象的信息保存在数据库（或静态变量）中。User 类提供了静态函数 getPointer 来根据类别返回相应的用户指针，以便通过多态的技术将新用户保存。

Customer 继承自 User 类，增加了数据成员 m_tableid，并提供了相应的公有接口来修改和查看餐桌号。公有函数 saveTableId 将当前 m_tableid 的值保存到数据库（或静态变量）中。重写了虚函数 insert 专门用来保存顾客信息。

Cook 继承自 User 类，增加了数据成员（详见图 2）保存厨师的工作数据，并提供了相应的公有接口来对它们进行修改和查看。公有函数 saveCook 用来保存厨师的工作数据。重写了虚函数 insert 专门用来保存厨师的所有信息。

Waiter 继承自 User 类，增加了数据成员（详见图 2）保存服务员的工作数据，并提供了相应的公有接口对它们进行修改和查看。公有函数 saveWaiter 用来保存服务员的工作数据。重写了虚函数 insert 专门用来保存服务员的所有信息。

Users 用户的群体类，内部的私有成员 m_Users（类型为 QVector<User*>）保存了用户的指针，便于实现多态。如果该对象由单一类别的用户构成，则可以用私有成员 m_type 来保存类别信息。

用户的信息按类别存在同一个数据库的不同数据表中。在单 App 模式下，Users 类的静态成员 UsersData 用于在运行期间存储用户信息。

Users 实现了用于用户管理的所有方法，其中主要的方法有：

- Users(int type)：调用私有函数 void extractUsersVector(int type)，将该类别的所有用户以指针的形式提取到 m_Users 中。
- void extractUsersVector(int type)：私有方法。根据 type 的值，从相应的数据库（或静态变量）中提取数据，并通过动态内存分配以指针的形式添加到 m_Users 中。
- void save()：保存所有的用户。先将该用户类别对应的数据库（或静态变量）清空，再通过 m_Users 中的每一个指针调用 insert 函数，由于已经实现了虚函数，用户信息会正确地保存到相应的数据库（或静态变量）中。
- User* userValue(const QString &phonenumber)：遍历 m_Users 查找该手机号对应的用户并返回其指针。
- Users search(const QString &key)：查找用户。定义查找结果为一个 Users 对象（使用默认构造函数），遍历 m_Users 中的所有用户，在手机号、密码中查找关键字 key。如果找到，则添加该用户到查找结果中。
- 通过操作 m_Users 数组，实现了所有基本操作，如 add（增加新用户）、remove（删除用户）、clear（清空用户）、edit（编辑用户信息）、indexOf（查找用户对应的下标）等。

此外，还需要实现复制构造函数并重载赋值运算符实现深复制；在析构函数、remove 和 clear 中要释放动态分配的内存资源。

3.2.2 菜品类

Dish 存储了菜品编号、类别、名称、价格、描述、状态、数量、桌号、厨师编号，并提供了公

有接口对上述属性进行修改和查看。

当 Dish 对象表示菜单中的一道菜时，桌号定义为 0，状态、厨师编号为缺省值，数量表示该菜品的售出总数。可通过 saveCount 将菜单中菜品的数量信息保存到数据库（或静态变量）。

当 Dish 对象表示顾客所点的菜品时，桌号为顾客的桌号，状态为菜品当前状态。有厨师认领该菜品时，保存厨师编号。数量表示顾客所点该菜品的份数。可通过公有方法 void save(char type) 来保存相关信息。其中 type 用来标识保存类型：type 为‘s’代表保存菜品状态，‘c’代表保存厨师编号。

Dishes 菜品的群体类，用内部的私有成员 m_Dishes（类型为 QVector<Dish>）来存放菜品信息，整数 m_table 表示菜品所属餐桌编号（为 0 表示为菜单中的菜品）。

菜单的数据库分为类别部分和菜品部分，类别部分包括一张数据表，里面写入了当前菜品的类别信息；菜品部分中，每种类别的菜品分别保存在以类别名称命名的数据表。用户所点菜品储存到以餐桌编号命名的数据表中。单 App 模式下，运行期间菜品信息由静态成员存储：categoryList（类型为 QStringList）用来存放类别信息，MenuDataInCategory（类型为 QHash<QString, Dish>）用来存放菜单菜品信息，Hash 表以类别名称为键；DishesData （类型为 QVector<Dishes>）用于存储每个餐桌所点菜品。

主要的方法包括：

- Dishes(int table)：将 table 的值赋给 m_table ，并调用私有方法 extractDishesVector 提取餐桌编号为 table 的菜品到 m_Dishes。若 table 为 0，则提取菜单中的菜品。
- void extractDishesVector()：私有方法。如果 m_table 为 0，则从数据库（或静态变量）中提取菜单信息到 m_Dishes 中，否则提取对应餐桌的所点菜品信息到 m_Dishes 中。
- void merge() 和 void split()：当本对象表示某位顾客所点的全部菜品，在顾客的点餐和结账界面时，需要将相同菜品合并。但在厨师认领做菜任务、顾客查看菜品制作状态时，应该把多份相同的菜品分开展示。为此，merge 和 split 方法将菜品合并或拆分。merge 遍历全部菜品，将重复的菜品合并，重数计入该 Dish 对象的 m_Count 中；split 将所有 m_Count 不为 1 的菜品拆分成 m_Count 份菜品。
- Dishes& operator=(QHash<QString ,Dishes> dishesInCategory)：重载了赋值运算符，便于将以类别存储的菜品信息直接转换为 Dishes 类型储存。先将本对象清空，再遍历静态变量 categoryList 依次获取类别信息，并通过 dishesInCategory 获得对应的 Dishes，添加到本对象中。
- void save()：根据桌号保存菜品信息。如果桌号为 0，代表保存到菜单。
- QHash<QString, Dishes> categoryHash()：将 Dishes 中的菜品按类别存储到 Hash 表中，便于管理员按类别管理菜品。
- Dishes extractUnfinishedDishes(int cookid)：供厨师界面调用的静态方法。遍历所有餐桌的所有菜品，将处于“等待中”的菜品和处于“制作中”且厨师编号为 cookid 的菜品提取并返回。
- Dishes search(const QString &key)：查找菜品。定义查找结果为一个 Dishes 对象（使用默认构造函数），遍历 m_Dishes 中的所有菜品，在名称、价格、描述中查找关键字 key。如果找到，则添加该菜品到查找结果中。
- 通过操作 m_Dishes 数组，实现了所有常用操作。如 add（增加菜品）、remove（删除菜品）、clear（清空菜品）、edit（编辑菜品信息）、indexOf（查找菜品对应的下标）。

3.2.3 餐桌类

DingingTable 储存了餐桌编号、用户信息、餐桌状态、服务员编号。提供了公有接口来对这些数据进行修改和访问。void save(char type) 用于将餐桌信息保存到数据库（或静态变量）中。type

用于标识保存类型：type 为‘s’表示保存餐桌状态，‘m’表示保存餐桌信息，‘w’表示保存服务员编号。

DiningTables 餐桌的群体类，方便餐桌的统一管理。QVector<DiningTable> 类型的私有成员 m_tables 用来所有的餐桌信息在数据库中保存在一个数据表中。单 App 模式运行期间，餐桌信息保存在静态变量 TablesData 中。主要的成员函数包括：

- DiningTables(bool extract)：构造函数，extract 用来标识是否要从数据库（或静态变量）中提取餐桌信息。若提取，则调用 extractDiningTables，否则创建空的餐桌列表。
- void extractDiningTables()：从数据库（或静态变量）中提取所有的餐桌信息到 m_tables 中。
- setTableNumber(int count, bool compulsory)：用来设置餐桌的数量。count 为目标数量，compulsory 用来标识是否强制设置。在强制模式下，本对象中的餐桌数量置为 count，返回 true；再非强制模式下，如果餐桌数量需要减少，且丢弃的餐桌中有顾客正在用餐，则放弃更改，返回 false。
- void save()：保存所有餐桌的信息到数据库（或静态变量）中。

此外，还有两个供服务员界面调用的静态方法：

- static DiningTables extractWaitingTables()：用来提取所有待认领的餐桌。遍历所有的餐桌，寻找餐桌状态为“点菜中”或“已下单”或“结账中”，并且服务员编号为-1（代表无人认领）的餐桌，将这些餐桌组合成 DiningTables 对象返回。
- static DiningTables extractServingTables(int waiterid)：提取编号为 waiterId 的服务员已认领的餐桌。遍历所有的餐桌，若服务员编号为 waiterid，则将该餐桌添加到返回结果中。

3.2.4 数据库类

DataBase 封装了数据库的基本操作。私有数据成员有数据库名称、数据表名称、创建表格所需参数、主键和一个 QSqlDatabase 的对象。主要的成员函数有：

- DataBase(QString dbname, QString tbname, QString parameters, QString primaryKey= “id”)：根据传入参数给数据成员赋值。
- DataBase(int type, QString primaryKey= “id”)：专门用于连接用户数据库, type 用来标识用户类别。
- 通过封装对 SQL 语句的执行，实现了 insert（插入数据）、update（更新数据）、select（选择某一列的所有数据）、clear（清空表）、count（返回行数）等常用操作。

3.2.5 消息类

消息类包括私有数据成员消息内容和餐桌编号，以及有对它们进行访问的公有接口，

3.2.6 Excel 导出类

封装了 Excel 的常用操作，包括插入工作表、插入单元格、设置当前工作表、合并单元格、保存文件、退出。通过 QAxObject 和 Excel VBA 实现。

3.2.7 Windows 消息发送类

封装了发送 Windows 消息的操作。通过静态成员变量保存本程序中所有界面的句柄，并按界面种类保存为 Hash 表。在必要的时可以调用 updateReceivers 来更新接收者界面句柄信息。该函数通过

列举当前所有顶层窗口，获得其标题，并保存具有相应界面标题的窗口的句柄来实现。公有方法 send 封装了消息的发送，静态方法 msgProcess 封装了消息的接收和处理。

3.2.8 开始界面类

开始界面类 SignWidget 用整数 m_type 来标识当前用户类别，用布尔值 m_state 来标识界面状态，false 代表登录界面，true 代表注册界面。静态成员 useDataBase(bool) 用来标识当前的模式，true 表示多 App 模式，false 表示单 App 模式。私有成员 copyRightClick(int) 用来记录版权声明按钮被点击的次数。当点击次数为 5 时，切换模式，并将其置零。切换到单 App 模式时，会从数据库中一次性提取所有信息到静态变量中。Users 数组 m_Users 用于保存所有用户信息。主要的函数有：

- SignWidget(int type)：构造函数，type 用来标识用户类别，默认值为 0（顾客）。将 m_state 初始化为 false，即登录界面；将 copyRightClick 初始化为 0；提取顾客信息到 m_Users[0]，并调用 setUp 进行界面初始化。
- bool checkValid()：检查输入信息是否有效。登录界面时，检查手机号格式、该用户是否已经注册；注册界面时，检查手机号格式，该用户是否已经注册、密码长度、两次密码是否一致。如果信息无效，则禁用登录或注册按钮。
- void onChangeType()：当前用户类别变化时触发。修改 m_type 的值到当前值，并进行界面的相应修改（详见第 12 页图 3）。如果该类别用户的信息未提取，则提取信息到 m_Users 中。
- void changeSignState()：切换登录和注册。修改 m_state 的值，并修改当前界面。
- void onSign()：注册或登录。由于之前调用过 checkValid 函数，故只有信息符合要求时才可点击登录或注册按钮。注册状态下给出注册成功消息，并保存新用户信息；登录状态下，通过 m_Users 及当前类别调用 userValue（见 3.2.1）获取用户指针，并得到密码。密码正确则打开相应用户的界面。顾客、厨师、服务员登录时，登录界面会将对应的指针传入相应的用户界面。

3.2.9 顾客界面类

主界面利用标签页组织五个用餐步骤。主要的方法有：

- CustomerWidget(Customer *customer)：构造函数。首先检查菜单是否为空，如果为空，则显示提示信息，顾客不能进行任何操作；如果不为空，则由顾客的指针获取对应的餐桌号，并查询状态。根据对应的状态确认当前所处的环节。
- void setFixedIndex(int index)：用于固定标签页，限制顾客按照步骤完成用餐流程。该函数过滤了标签页改变的信号。
- DiningTable getTable()：顾客选桌后可通过此函数获取对应的餐桌。

选桌界面类 主要有的方法有：

- void tableUpdate()：提取餐桌数据并显示，顾客可以查看当前所有餐桌的状态，系统默认选择第一个空闲的餐桌（如果存在）。如果当前没有可用餐桌，则给出提示，禁用提交按钮。
- void onTableChanged()：选定餐桌号发生变化时触发，改变提示信息。
- void onSubmitTable()：提交餐桌时触发。保存餐桌编号至顾客信息，主界面保存餐桌编号，将状态置为“点菜中”，然后跳转到点餐界面。

点餐界面类 用 Dishes 对象 m_Menu 和 m_Tray 分别表示菜单和“我的餐盘”。主要的方法有：

- void setUp()：提取菜品信息并初始化菜单和“我的餐盘”。

- void onTrayChange()：当所选菜品发生改变时触发。根据菜单中的选择菜品更新托盘内容。如果餐盘内容为空，则禁用提交按钮。
- void onCountEdit()：用户编辑菜品数量时触发。修改的结果保存在 m_Tray 中对应菜品的 m_count 成员中。
- void onSubmitDishes()：提交菜品时触发。首先更新菜单中每道菜品被点的次数，并调用 saveCount 进行保存。将每道菜品的状态都置为“等待中”。接着调用 split 方法[4] 将 m_Tray 中的菜品拆分，再调用 m_Tray.save() 保存菜品数据，并将餐桌状态更新为“已下单”。

用餐界面类 用户可以实时查看菜品的进度，并给服务员发送消息。主要的方法有：

- void update()：提取本餐桌的菜品信息，更新菜品状态。
- void onSend()：：通过主界面的 getTale 获取到餐桌，再调用 DiningTable 类的方法设置并保存信息。

支付界面类 顾客可以查看详细的账单，并选择支付方式进行支付。主要的方法有：

- void SetUp()：界面初始化。获取对应餐桌，将餐桌状态置为“结账中”。提取该餐桌的所有菜品信息，并使用 merge 方法[5]进行整合后，展示在列表中。
- void onWechat()，void onCash()：微信支付和现金支付。点击支付完成后，会给服务员发送“结账”消息，等待服务员处理。
- void CheckPayState()：检查餐桌状态，若为“已结账”，则跳转到评分界面。

评分界面类 顾客打分结束后提交评分，调用 onSubmit() 方法，将每道菜品的评分写入对应厨师的工作数据中，服务员的服务桌数加 1，并和评分一同写入工作数据中

3.2.10 管理员界面类

管理员界面分为三个子界面，用标签页来组织。包括菜单管理，用户管理和餐桌管理。

菜单管理类 菜单管理包括菜品类别管理和菜品管理。使用 Dishes::categoryList 来保存菜品类别信息；用 m_DishesInCategory（类型为 QHash<QString, Dishes>）来分类别存储菜品；m_Dishes 用来保存全部菜品；m_chosenDishes 用于保存选中类别的菜品；m_searchDishes 用来保存菜品的搜索结果。主要的方法有：

- void onCategoryManage()：管理菜品类别，弹出对话框，显示出当前全部的类别信息，和所有的管理按钮。
- void onCategoryEdit()：编辑菜品类别并更新类别列表。修改类别名称和该类别中所有菜品的类别信息。如果改后名称和另一个已有类别的名称相同，则询问是否合并。若合并则将两个类别的菜品归到一起；若不合并则修改失败。
- void onCategoryAdd()：添加一个新的菜品类别，并更新类别列表。可以在类别管理、添加菜品、编辑菜品时调用。
- void onCategoryRemove()：从 Dishes::categoryList 移除类别，并清除该类别中的所有菜品。
- void onCategoryMove()：将一个类别的全部菜品移动到其它类别中。并修改原类别所有菜品的类别信息。
- void onCategoryClear()：清空所有类别和其中的菜品。

[4]详见第 6 页 3.2.2 节

[5]详见第 6 页 3.2.2 节

- void categoryUpdate(bool initialize)：更新菜品类别列表。initialize 用于标识是否为初次更新。初次更新默认所有类别都被选中；之后的每次更新各种类别的选中状态保持与更新前一致，新增的类别默认为选中状态。
- checkMenuValid(bool editing, const Dish &dish)：有效性检查，在编辑和添加菜品时有不同的判断机制[6]。
- void menuUpdate()：更新菜单数据。判断哪些类别被选中，将这些菜品添加到 m_chosenDishes 中，并在列表中显示。
- 通过调用 Dishes 类的相关方法，实现菜单管理的增（onMenuAdd）、删（onMenuDelete）、改（onMenuEdit）、查（onMenuSearch）、清空（onMenuClear）。

用户管理类 用户管理同样需要选择用户类别，和菜单管理不同的是，用户的类别已经确定为五种，故不需要添加和类别管理相关的功能。和菜品管理类似，具有 usersUpdate 和 checkUserValid 来更新用户列表和检查用户信息有效性。通过调用 Users 类的相关方法，实现了用户管理的增（onUsersAdd）、删（onUsersDelete）、改（onUsersEdit）、查（onUsersSearch）、清空（onUsresClear）。

餐桌管理 管理员可以对“空闲”或“禁用”的餐桌修改状态，而不能修改有顾客占用的餐桌。管理员可以设置餐桌数目。首先尝试非强制修改，调用 DiningTables 的 setTableNumber 方法[7]，设置 compulsory 为 false。如果失败，则询问是否强制修改。若强制修改设置 compulsory 为 true，重新调用 setTableNumber 方法。

3.2.11 服务员界面类

服务员界面类分为三个部分：认领餐桌、顾客服务、我的餐桌。

认领餐桌类 利用 DiningTables 对象 m_waitingTables 来管理所有未认领餐桌。主要的方法有：

- void update()：更新待认领餐桌列表，通过调用 DiningTables 类的静态成员函数 extractWaitingTables 来提取待认领的餐桌并保存到 m_waitingTables 中。
- void onClaim()：认领餐桌，将自己的编号写入餐桌的服务员编号中。

顾客服务类 利用 DiningTables 对象 m_servingTables 来管理该服务员认领的所有餐桌。利用 QVector<Message> m_messages 来管理所有的消息。利用 m_time（类型为 QHash<QString, QTime>）来记录每条消息处理的时间。主要的方法有：

- void update()：更新消息列表。利用 DiningTables::extractServingTables 来获取该服务员认领的所有餐桌，再提取每个餐桌的消息并保存为 Message 类对象。若 m_messages 中已经存在该消息对象则忽略。出现新消息时，将该消息绑定时钟并在消息列表中将每条消息显示出来。特别地，如果该消息是由厨师发送的菜品就绪的消息，则通过解码[8]，用红色显示出来。
- void onSolve()：解决某条消息。通过消息对象得到餐桌号和消息内容，重置该餐桌的消息内容为空。如果是结账消息且顾客处于结账状态，解决后将餐桌状态改为“已结账”；如果解决了上菜消息，则更改对应的菜品状态为“已上菜”。解决消息后，利用为该消息绑定的时钟计算服务时间，加到该服务员的服务时间中，并使服务次数加 1，写入工作数据中。

我的餐桌类 用于显示该服务员认领的所有餐桌。其中的 update 方法利用 DiningTables::extractServingTable
来获取该服务员认领的所有餐桌并保存到私有成员 DiningTables 对象 m_servingTables 中，再通过列表的形式显示出来。

[6]详见第 13 页 3.4.1 节

[7]详见第 7 页 3.2.3 节

[8]详见第 14 页 3.4.2 节

3.2.12 厨师界面类

利用 Dishes m_unfinishedDishes 来管理所有的未完成菜品。利用 m_time（类型为 QHash<QString, QTime> ）记录每道菜品完成的时间。主要的方法有：

- void update()：通过调用静态方法 Dishes::extractUnfinishedDishes 来获取所有的未完成菜品，并显示在未完成菜品列表中。
- void onClicked()：认领或完成任务。如果当前按钮为“认领任务”，则将该菜品状态改为“制作中”，厨师编号改为该厨师的编号。同时为该菜品绑定时钟；如果当前按钮为“已完成”，则将该菜品状态改为已完成，并将菜品信息加密[9]，写入对应餐桌的消息中利用为该菜品绑定的时钟计算时间，加到该厨师的做菜总时间中，并使做菜数量加 1，写入工作数据中。

3.2.13 经理界面类

经理界面包含三个子界面，分别为 CookTable，WaiterTable，DishTable，用来显示厨师工作数据、服务员工作数据、菜品销售数据。三个界面类由表格界面类 Table 继承。onExport() 方法通过使用 ExportExcel 类来将三个数据表导出为 Excel 形式。

Table 定义了纯虚函数 virtual QVector<QStringList> getContent() 和 virtual QStringList getHeaderLabel()。用来获取表格内容和表头内容。保护方法 setUp() 完成了界面的布局和表格的写入。

CookTable，WaiterTable，DishTable 均集成自 Table 类，实现了各自的 getHeaderLabel 和 getContent 方法来获取不同类型的数据。

3.3 界面结构设计

3.3.1 开始界面设计

餐厅服务系统应该以顾客为中心，故系统的初始界面应该为顾客登录界面。且对于顾客来说，界面应该美观整洁，不应该出现很多供其它类别用户使用的按钮。基于以上考虑，设计开始界面如图 3 所示。程序的初始界面为图 3a，上面的“顾客”按钮点击之后会变成一个下拉菜单，此时用户类别变成管理员。通过下拉菜单可以切换用户类别。当用户类别切换回“顾客”时，界面又回到图 3a 的状态。点击“新用户？点击注册”按钮可以切换到注册界面，如图 3c 所示。

在界面的最下面的版权声明，其实是一个隐藏的按钮。只要点击它五次，就可以在多 App 模式和单 App 模式下切换。多 App 模式下一个登录界面只负责一个用户的登录，登录成功后登录界面自动关闭；单 App 模式下登录成功后开始界面仍然存在以便其他用户登录或注册。

3.3.2 顾客界面设计

顾客界面由五个子界面构成，用标签页的形式组织不同的子界面。

餐桌选择 界面主体为一个表格，显示当前所有的餐桌的编号以及状态。每个餐桌前面都有一个单选按钮，用于选择餐桌。表格下方的标签用来提示顾客所选餐桌的编号。点击提交按钮进入开始点餐界面。

开始点餐 界面主体为两个表格，左侧表格用来展示菜单，右侧表格用来显示“我的餐盘”。顾客点击菜品前面的复选框，就可以将菜品加入“我的餐盘”，双击餐盘中的某一道菜品，弹出修改数量的对话框。点击提交按钮提交餐盘中的菜品，进入开始用餐界面。

[9]详见第 14 页 3.4.2 节

(a) 顾客登录界面　　(b) 管理员登录界面　　(c) 顾客注册界面

图 3: 开始界面的切换

开始用餐　界面主体为一个表格，用来实时显示所点的所有菜品和对应的状态。表格下方的输入框用来输入消息。点击发送之后，输入框中的消息回发送给认领该餐桌的服务员。点击结账按钮进入结账界面。

结账　界面主体为一个表格，顾客可以查看完整的账单信息。顾客可以点击“微信支付”或“现金支付”进行支付。支付完成后，当服务员处理完用户结账的消息时，界面自动跳转到评分环节。

评价　界面主体为一个表格，里面有每道菜品的信息。顾客可以为菜品打分；表格下方是服务员的打分框，顾客可以为服务员打分。点击提交后，界面关闭，并显示告别信息。

3.3.3　管理员界面设计

管理员界面包括菜单管理、用户管理和餐桌管理三个子界面。

菜单管理　界面主要由类别管理下拉菜单、菜单列表、四个管理按钮、搜索框构成。

类别管理框下拉后出现所有的类别，管理员可以勾选前面的复选框进行管理。类别列表的最后一行是一个类别管理按钮，点击之后出现一个对话框，显示所有类别的详细信息，以及增加、编辑、移除、移动、清空按钮，用于管理类别。添加类别时，弹出对话框，请求输入类别名称；编辑类别时，弹出对话框，对类别进行修改。点击移除时，直接删除所选类别；点击移动时，弹出对话框，要求指定原类别与目标类别；点击清空直接清空所有的类别。

菜单列表中显示了所选类别菜品的详细信息。管理员可以通过右侧的四个管理按钮：添加、删除、编辑、清空来管理菜品。添加和编辑菜品时，会弹出相应的对话框，可以添加或编辑菜品的所有信息：类别、名称、价格、描述。对话框还提供了类别的添加按钮，即可以在添加和编辑菜品期间新建类别。点击删除按钮会删除菜单中选中的菜品，点击清空按钮会弹出提示消息，询问是否清空。搜索框可以实现动态搜索，即当搜索框中内容变化时，菜单马上显示搜索结果。搜索成功后右侧的提示标签会显示“搜索成功”的信息，否则显示搜索失败。在搜索的过程中，管理员也可以执行编辑或删除操作。

用户管理　界面主要由类别选择组合框，用户列表，四个管理按钮，搜索框构成。管理员可以在类别选择组合框中选择需要查看的类别，结果显示在用户列表中。管理按钮和搜索框的操作方法和菜单管理类似。

餐桌管理　界面主要由一个表格构成，里面显示了所有餐桌的编号和状态。管理员可以通过餐桌前面的复选框修改无顾客占用的餐桌的状态（空闲/禁用）。表格上方是修改餐桌数量按钮，点击后弹

出对话框要求输入餐桌数量，如果修改失败，弹出对话框询问是否强制修改[10]。

3.3.4 厨师界面设计

厨师界面只有一个表格，每一行包括菜品名称和按钮。菜品无人认领时，按钮文字为“认领任务”。点击按钮，按钮文字变为“已完成”。点击“已完成”，该菜品从列表移除。

3.3.5 服务员界面设计

服务员界面由认领餐桌、顾客服务和我的餐桌三个子界面构成。

认领餐桌 由一个表格构成，显示所有未认领的餐桌，每个餐桌右侧有一个认领按钮，点击后即可认领该餐桌。

顾客服务 由一个表格构成，显示所有该服务员认领的餐桌的消息，每条消息右侧有一个解决按钮，点击之后该消息从消息列表中删除。

我的餐桌 由一个列表构成，显示该服务员认领的所有餐桌。

3.3.6 经理界面设计

经理界面由厨师工作数据、服务员工作数据、菜品销售数据三个子界面构成。每个子界面中由显示相应信息的表格，和一个导出 Excel 按钮。点击按钮会弹出对话框选择保存路径，确认会开始导出。导出结束后询问是否打开 Excel，点击打开即可查看导出结果。

3.4 关键设计思路

3.4.1 容错功能设计

由于本项目与用户之间的交互较多，所以要充分设计容错功能。本项目的容错功能主要通过以下手段实现。

检查函数 即当用户输入数据时，实时调用检查函数检测内容是否合法。开始界面类中的函数 checkValid 用来检查登录或注册信息的合法性。登录模式下，手机号未注册、手机号格式不正确会给出错误信息；注册模式下，手机号已注册、手机号格式不正确、密码长度不符合要求、两次密码不一致会给出错误信息。

在管理员管理菜单时，需要进行添加或编辑。函数 checkMenuValid 实现了检查的功能。在添加菜品时，菜品名称已存在、菜品名称为空、菜品价格为空会给出错误信息。但是在编辑菜品时，如果名称输入框中的菜品名称已存在，需要判断此时的菜品名称是不是修改前的菜品名称。如果是则说明没有对名称进行改动，此时不应报错。所以函数 checkMenuValid 具有参数 editing 来标识是否处于编辑状态，而参数 Dish 用来指明正在修改的菜品。用户管理中的 checkUserValid 函数也是这个道理。

正则表达式 当对输入格式有要求时，采用正则表达式严格控制。例如手机号输入框中只能输入首位是 1，且长度不超过 11 位的内容；管理员对菜品管理时，菜品的价格只能输入整数或最多两位小数。

禁用按钮 判断输入非法后，采用禁用相关按钮的方式限制。例如当用户没有点餐时，提交餐盘的按钮会被禁用。

[10]详见第 7 页 3.2.3 节

3.4.2 消息加密传送

厨师可以给服务员发送菜品就绪的消息，顾客也可以给服务员发送消息。所以服务员必须要区分消息是来自厨师还是顾客。顾客可能点了很多份同样名字的菜品，某道菜品就绪后服务员接到消息，点击“已解决”，必须实现对应的菜品状态更改为“已上菜”。所以必须将菜品的编号信息写入消息，以便服务员确定具体是哪道菜品就绪，而不是只靠菜品名称查找。

基于以上两点考虑，厨师发送消息时需要对消息进行加密，消息格式为“name␣␣id␣”，其中 name 为菜品名称，id 为菜品编号。注意该消息的 name 和 id 之间有两个空格，末尾有一个空格。顾客在发送消息时，会自动调用函数将其消息首尾的空格删除，于是厨师和顾客的消息就可以通过末尾字符是否为空格区分。服务员接收到消息后，通过查找“␣␣”即可定位菜品名称和菜品编号，进而正确处理上菜消息。

3.4.3 多用户实时通信

多用户系统的复杂之处在于多个用户的数据共享与信息同步。尤其是在一整套用餐流程中，顾客、服务员和厨师之间以及同种类别用户之间有着复杂的交互关系（如表 2所示）。在多 App 模式和单 App 模式下数据共享分别由数据库和静态变量实现。两种模式下的信息同步都通过 Windows 消息完成。

接收方操作 / 发送方操作 / 发送方	顾客	服务员	厨师
顾客	更新餐桌列表 提交餐桌	更新消息列表 发送消息	更新菜品列表 提交菜品
服务员	更新菜品状态、跳转界面 处理上菜和结账消息	更新餐桌列表 认领餐桌	\
厨师	更新菜品状态 认领或完成任务	更新消息列表 发送菜品就绪消息	更新任务列表 认领任务

表 2: 用餐流程用户交互关系

本项目基于 Windows 消息实现了多用户之间的实时通信。Windows 消息发送类 WMSender 封装了 Windows 发送消息的操作。其中的 updateReceiver 函数遍历所有的顶层窗口，将窗口标题为六种界面的标题的窗口的句柄分类别保存下来。每种界面的窗体句柄存在数组中，不同界面的句柄数据按界面名称存储为 Hash 表。每次发送消息时指定接收者类型，则发送函数会遍历该类型的所有窗体句柄，发送对应的消息。发送的消息内容有三种：“update”，“new”，“close”。“update” 用于提示接收者更新数据；“new” 用于提示所有用户有新用户来访，需要调用 updateReceiver 更新窗体句柄；“close” 用于提示接收者关闭窗口。静态函数 msgProcess 用于处理消息。只需在发送者里加入 WMSender 对象，在接收者里通过重写 nativeEvent 调用 msgProcess，即可实现消息的收发。

3.4.4 数据处理的封装

本项目中的数据相关类主要有用户类、菜品类、餐桌类。定义了三种群体类，并将常用操作封装起来，便于用户界面（尤其是管理员界面）的使用。同时，这些类中将保存数据与提取数据操作的封装极大地方便了单 App 和多 App 两种操作模式的切换。

3.4.5 时钟的绑定

为了计算服务员和厨师的工作时间，需要对每个任务绑定时钟。时钟由 QHash<QString, QTime> 类型的 Hash 表来管理。对于服务员，将“餐桌编号，消息内容”作为键；对于厨师，将“餐桌编号，菜品编号”作为键。服务员每新收到一个消息，就绑定一个 QTime 对象记录时间，解决后通过当前时间与收到消息的时间差来作为该消息的处理时长。同理，利用厨师完成菜品和认领菜品的时间差作为厨师的做菜时间。进而可以评定厨师和服务员的工作效率。

4 项目总结

4.1 项目亮点

4.1.1 功能强大，考虑周全

本项目实现了多用户的系统，能够做到不同用户之间的实时通信，并具有单 App 和多 App 两种模式，满足不同的需求。各个界面的逻辑、功能都考虑周详，力求完整、合理、便捷。例如，在管理员对菜品和用户的管理时，实现了动态搜索，即搜索结果随输入框中内容变化而变化。实现了搜索过程中的编辑与删除，更加方便了信息的管理。在菜品类别和用户类别的选择中，实现了“全选”框的三态效果。经理界面下，导出 Excel 的设计方便了经理对餐厅的管理。

4.1.2 界面美观，操作简便

为了美化界面，提升用户体验，本项目使用了 QSS 进行渲染。此外，为了实现不同系统下的风格统一和进一步美化，去除了系统自带的边框和标题栏，并添加了自定义的标题栏。为此，需要重写窗口的 mousePressEvent 实现鼠标拖动效果。同时利用 QPainter 绘制了界面的圆角和阴影等效果。

本项目连接了大量的信号槽用于方便用户操作。例如，在登录界面，用户输入手机号之后只需按下回车，即可切换到密码输入框，再按下回车即可点击登录按钮。

4.2 开发难点

4.2.1 数据的结构与组织

本项目中涉及到很多种类的数据，如何合理有效地组织这些数据是开发过程中的重要问题。通过分析不同数据之间的联系，经过多次的尝试和修改，现有的结构才被确定。

4.2.2 跨进程的通信

为了使程序功能更符合实际，项目作者从一开始就决定要开发出可以多个 App 互相交互的程序。但是这并不容易，尤其是信息的同步是一大难题。项目的第一个版本采用了数据库通信的方式，即一方写入数据库，另一方不断扫描数据库，当发现有数据变化时做出相应操作。但是毫无疑问这样的处理手段比较笨拙，而且效率很低。

项目的第二个版本采用了 TCP/IP 进行通信。将服务器端和客户端的收发消息的操作封装起来，实现了多用户的通信。然而这个版本的缺点是，在开启程序之前需要先打开服务器，否则不能建立连接。这不利于对用户容错性的实现。

项目的第三个版本，也就是此版本，采用了 Windows 消息[11]进行通信，完美地解决了之前版本的问题，使得跨进程的通信变得简单、优雅。

[11]感谢董老师提供的思路

5 心得体会

通过本项目的开发，笔者有以下几点收获和体会。

5.1 自学能力

本次项目的开发的大部分知识都需要通过自己查找资料来学习。例如数据库、Qt、Windows API 等。这些资料大部分来源于网络，而网络上的知识又浩如烟海。所以，有效地查阅并筛选资料，并将其合理地应用到自己的项目中，是一种让人受益终生的能力。

5.2 精益求精的精神

由于本项目在暑假就布置下来，笔者在暑假期间自学了 C++ 和 Qt，并完成了项目要求的功能。但是笔者并没有就此止步，而是不断钻研新的技术，例如关于跨进程的通信技术。对于界面的美化，笔者的界面部分用纯代码编写。虽然这样的任务量很大，但是这更加有利于对界面的精细调整和控制。

5.3 调试的收放自如

在程序的调试过程中，有些时候是可以通过不断分析逻辑、设置断点跟踪等手段找出错误。但在另一些情况下，问题出在其它地方，而这种问题不是可以通过不断调试来解决的。尤其是在从网络上学习到新知识时，一些基本的配置没有完成，程序便无法实现目的。例如，笔者为了实现经理界面能够将服务员和厨师的工作数据以及菜品的销售数据导入到 Excel 中，自学了利用 QAxObject 和 Excel VBA 操作 Excel 的方法。但是，当笔者测试网上的代码时，总是不能实现。在这种情况下，笔者没有过度纠结这个问题，而是先暂时搁置。很久之后，笔者决定再次着手解决这个问题。笔者在搜索的过程中无意中看到了 <qt_windows.h> 头文件，突发奇想，引用了这个头文件，并根据报错信息尝试添加了一行代码，居然用了很短的时间就导出成功了。通过这个过程，笔者明白了程序失败的原因可能是多方面的，有些时候单单凭努力是难以解决。反而如果做到收放自如，暂时搁置，说不定过一段时间后就有了新的灵感。