### 餐厅点餐系统需求分析说明书

目录

**[1. 引言](#_Toc19551_WPSOffice_Level1)** **[1](#_Toc19551_WPSOffice_Level1)**

[1.1 编写目的](#_Toc31886_WPSOffice_Level2) [2](#_Toc31886_WPSOffice_Level2)

[1.2背景说明](#_Toc4576_WPSOffice_Level2) [2](#_Toc4576_WPSOffice_Level2)

[1.3术语定义](#_Toc15962_WPSOffice_Level2) [2](#_Toc15962_WPSOffice_Level2)

[1.4参考资料](#_Toc28561_WPSOffice_Level2) [2](#_Toc28561_WPSOffice_Level2)

**[2. 概述](#_Toc31886_WPSOffice_Level1)** **[2](#_Toc31886_WPSOffice_Level1)**

[2.1任务概述](#_Toc1067_WPSOffice_Level2) [2](#_Toc1067_WPSOffice_Level2)

[2.2 功能概述](#_Toc10509_WPSOffice_Level2) [3](#_Toc10509_WPSOffice_Level2)

1. **引言** 
   1. 编写目的

在软件开发前期，通过跟客户沟通，获得一份双方都认同的需求说明文档，针对用户提 出的修改要求，及时处理，更新需求文档。确定需求后才能进行相应的开发。

明确系统的基本需求，旨在整个开发过程中，指导和帮助编写代码分析解决系统的问题， 需求分析文档在软件开发过程中是最重要的一部分，可以在前期避免系统的一些问题，需求 分析的质量决定着系统的质量，贯穿着整个开发的过程。

1.2背景说明

在现代社会城市化的大背景下，城市的规模和人数不断增长，同时也带动了城市经济全 面化的发展，俗话说：民以食为天。餐饮业是一个永远不会衰败的行业，但餐馆由于受到空 间大小的影响，盈利率几乎不会再提高，想要增加更多的盈利就必须加快服务的效率，同时 带动消费效率的提高。这时，我们就会考虑一个方便迅速快捷的服务方式来改变这种情况， 提高竞争力。

1.3术语定义

数据结构：链表（内核链表），结构体，数组，队列，数据库

程序：进程线程

通信：TCP协议，套接字

1.4参考资料

《软件工程》、《数据结构》、《unix网络编程》、《c语言程序设计》、《软件需求分析》

1. **概述**

2.1任务概述

当前餐厅存在的问题：当前餐厅中的大多数环节都是有服务员跑动来完成的，这样会造成餐厅使用的劳动力过多，比较费时费力，而运营效率低下。导致效率低下的原因主要有：

1. 服务员跑动太多浪费时间。例如，服务员将订单送到厨房部的过程中消耗过多时间，从而不能及时接待新进入的顾客，从而影响顾客就餐，影响顾客的回头率。

2. 菜谱是之前打印好的，不能及时的更新菜谱，顾客看到的不是最新的菜谱。

3. 信息的传递都是人工，费时、费力，效率低下。

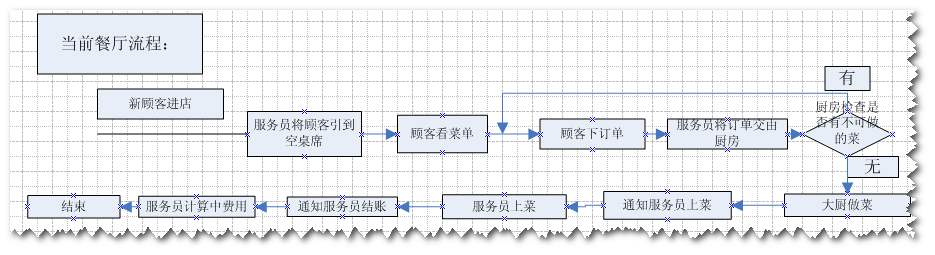


图 1 用例图

解决方法：

1. 将传统的人工转为使用餐厅点餐系统，系统可以为顾客提供以下服务：座位的选择、浏览最新菜谱、自助点菜 ( 下单 ) 、进行催单、查询订单的排队情况以及结账申请。

2. 将信息的传递由传统的人工方式改为由信号传递，菜谱用电子设备显示可以及时的更新最新的菜谱显示给顾客。将顾客与餐厅之间的信息传递智能化，顾客的要求可以直接传达到餐厅的各部，使服务员由足够多的时间去处理别的事情。

2.2 功能概述

本系统的主要功能分为：餐厅前台端系统（客户端）、服务器后台端系统（服务端）。

以下分别对各个端的系统功能做一个功能概述。

* 餐厅前台端系统：

1. 从服务端获取最新的菜谱；

2. 在本地系统中进行点单，并将订单传送到服务器后台系统；

3. 催单，向服务器后台端系统发送信息，要求系统尽快处理本订单；

4. 结账，向服务器后台端系统发送结账请求。

* 服务器后台端系统：

1. 更新本地系统中的菜谱信息；

2. 接收订单，并保存订单信息；

3. 将订单传送到厨房端系统；

4. 对催单请求进行处理；

5. 对订单进行结账处理，并将信息反馈到餐厅前台端系统；

6. 管理当前系统中等待厨房处理的订单队列；

7. 处理订单信息队列；

8. 将已处理的订单进行处理（写入数据库），以便日后做账；

1. **数据流程图**

3.1 数据字典

数据字典用来描述系统所要处理的数据的全部信息。

客户端和服务端通信的数据统一为json格式，服务端解析json格式到对应的结构体。

订单

描述：用来描述某一餐桌的菜品订单信息

别名：无

组成：账单编号 + 菜品编号 + 菜品名 + 菜品价格 + 菜品数量 + 菜品总价 + 确认订单时间 + 确认完成时间 + 订单标志位

入座信息

描述：用来描述某个客户的入座请求信息

别名：

组成： json 格式字符串，包含信息：用户编号 + 用户名 + 餐桌号

点单信息

描述：用来描述某一餐桌所点的菜的信息

别名：无

组成： json 格式字符串，包含信息：订单类型 + 菜品编号 + 所点数量

菜谱信息

描述：用来描述餐厅的最新菜谱数据

别名：无

组成： json 格式字符串，包含信息：菜品编号+菜品名 + 价格 + 库存 + 类型

催单信息

描述：用来描述一个订单的催单处理请求

别名：无

组成： json 格式字符串，包含信息：订单处理类型

结账信息

描述：用来描述一个订单的结账处理

别名：无

组成： json 格式字符串，包含信息：菜品名 + 单价 + 总价

菜品信息

描述：用来描述一道菜的相关信息

别名：菜品

组成：菜名编号 + 菜名 + 单价

菜谱表

描述：用来描述系统中所有菜品信息

别名：菜谱

组成：由菜品信息构成的一个表

餐桌表

描述：用来描述餐厅当前所有餐桌的使用信息

别名：无

组成：餐桌信息

餐桌信息

描述：用来描述餐厅当前某一餐桌的使用信息

别名：无

组成：餐桌编号 + 餐桌占用信息

等待配菜员处理的 json 格式订单队列

描述：用来描述由配菜员处理的 json 格式订单队列

别名：订单队列

组成： json 格式订单 + 桌号

3.2 数据结构

数据结构用于实现在计算机中的存储和处理，有别于数据字典。

用户表

描述：用来描述就餐会员用户信息

别名：无

组成描述：用户 ID+ 用户名 + 桌号 + 订单状态 + 账单状态 + 就餐时间

菜谱表

描述：用来描述系统中所有菜品信息

别名：无

组成描述：库存表

账单表

描述：用来描述系统中所有消费记录

别名：无

组成描述：账单编号 + 菜品信息 + 消费总价 + 订单时间

餐桌表

描述：用来描述餐厅当前所有餐桌的使用信息

别名：无

组成描述：由餐桌信息构成的一个顺序表

订单

描述：用来描述某一餐桌的用餐信息

别名：无

组成描述：账单编号 + 菜品编号 + 菜品名 + 菜品价格 + 菜品数量 + 菜品总价 + 确认订单时间 + 确认完成时间 + 订单标志位

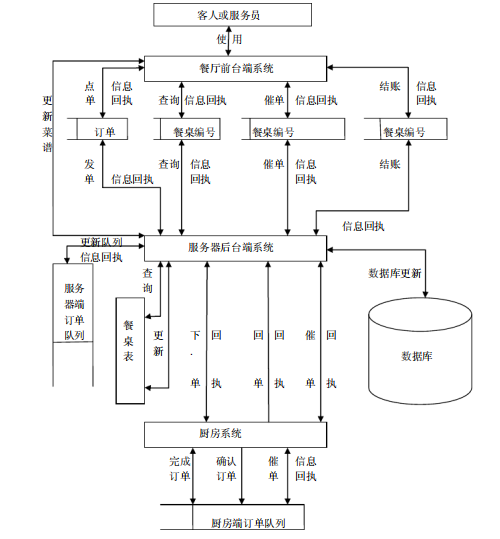
订单队列

描述：用来描述由厨房前台端系统处理的订单信息的队列

别名：无

组成描述：由订单构成的一个队列

3.3数据流程图



1. **系统要求**

4.1 性能要求

系统必须有很高的运作效率，用户填写的订单在输入到系统后，系统必须快速及时地做出响应，迅速地处理有关信息，显示出所有必须的信心并打印出各项清单；由于需要存储大量用户的信息，所以系统必须有足够大的存储容量；系统必须能够承受各种不确定因素以保护存储的信息的安全，并能够准确记录各种内部事件；系统可靠性也是系统最为重要的性能要求之一，要求在意外因素之下，系统仍然能够保护系统和存储的信息的完整。

4.2 体验要求

接口是用户与系统进行通信的主要通道，因此，接口除了完备的功能外，还必须简洁明了，方便用户的使用，这样才能提高餐厅的工作效率，继而为餐厅节省了不必要浪费的时间。

必须使用户能准确快速地进行信息输入，从而使得用户与系统之间的通信时间尽可能地缩短。

系统必须能准确快速地打印出各类清单。

1. **意外处理**

|  |  |
| --- | --- |
| 出错信息 | 错误处理 |
| 入座不成功 | 显示可选桌号 |
| 提交订单错误 | 提示检查订单格式内容 |
| 结账金额错误 | 检查是否有菜品没上 |
|  |  |

1. **可行性要求**

6.1 技术可行性

本系统是基于 linux 基础上研发的，目前采用 linux 和 sqlite3 作为后端，安卓为前端的技术已经很成熟，并且 linux 和 sqlite3 相关的技术支持很多，所以在技术可行性方面不存在问题。

在前端，采用安卓操作系统带来了更加稳定和易用的技术，并且安卓具有节省资源和成本的优势。

6.2 经济可行性

基建投资：

需要一个 PC 终端，即一台联想商务电脑， windowsXP 或 win7 操作系统，一台惠普商用打印机，还要一个数据库管理系统，多个小型终端（可以是平板电脑），总计 5 万左右。

其它一次性支出：

|  |  |
| --- | --- |
| 项目 | 价格（万元） |
| 可行性分析以及需求研究 | 1 |
| 开发设计 | 2 |
| 测试基准 | 1 |
| 转化原有的数据库 | 0.5 |
| 总价 | 4.5 |
|  |  |

经常性支出（五年）：

PC 终端和各个客户端的维护，网络的维护，再加上一些其他的不可预知的费用，大概要 5.5 万左右。

支出总计： 5 + 4.5 + 5 .5 = 15 万元

6.3 法律可行性

本系统采用的均是开源资源，所以除了在硬件上有成本支出外，在软件方面上不存在法律风险。

1. **结论意见**

经过上面的可行性研究分析，该项目在操作性、技术可行性、经济可行性、法律可行性上军满足要求。因此，开发此系统的构想是可行的，可以着手实施 。