****

**Report 1** **：** **Restaurant Automation**

**实现餐厅自动化系统**

**小组成员**

夏瑞

杨帆

吴晨

张天霖

**小组分工：**

夏瑞：负责用户故事的复杂度和优先级判断、部分具体用例图和获取分析数据算法

杨帆：负责部分具体用例图和领域分析

吴晨：负责整体用例图和动态加载时间表算法

张天霖：负责客户需求、工作计划和功能需求

目录

[客户需求描述： 4](#_Toc448345330)

[工作计划： 6](#_Toc448345331)

[术语汇编： 7](#_Toc448345332)

[专业术语： 7](#_Toc448345333)

[非专业术语： 7](#_Toc448345334)

[用户故事： 8](#_Toc448345335)

[复杂度： 8](#_Toc448345336)

[优先级： 9](#_Toc448345337)

[功能需求说明: 11](#_Toc448345338)

[利益相关者 11](#_Toc448345339)

[角色和目标 11](#_Toc448345340)

[用例： 12](#_Toc448345341)

[介绍 12](#_Toc448345342)

[整体用例图： 12](#_Toc448345343)

[具体用例图： 13](#_Toc448345344)

[追溯矩阵 26](#_Toc448345345)

[领域分析: 28](#_Toc448345346)

[领域模型 28](#_Toc448345347)

[1.概念定义 28](#_Toc448345348)

[2.关联定义 29](#_Toc448345349)

[3.领域模型图 30](#_Toc448345350)

[算法： 30](#_Toc448345351)

[获取分析数据 30](#_Toc448345352)

[1.产品的收益 30](#_Toc448345353)

[2.菜品的受欢迎程度 31](#_Toc448345354)

[3.就餐时长 31](#_Toc448345355)

[4.等待订单的时长 31](#_Toc448345356)

[动态加载时间表算法： 33](#_Toc448345357)

# 客户需求描述：

现在社会正步入科技时代。我们的餐厅要紧随时代潮流，准备进行技术改革，以实现餐厅自动化。我们与相关公司达成协议，使用类似苹果平板或安卓平板等新型手持设备来向客户展示我们餐厅的功能和理念。我们希望通过一个能够在设备中运行的应用程序与客户进行信息交互。下面是我们想要解决的问题和欲解决方法：

问题：客户等待用餐和上餐时间很长

在餐饮业，时间就是金钱。如果单位时间内服务客户数量多则销售量多从而利润增多，我们要通过某些解决方法来实现等待时间减少。有以下几个措施。

实现可使用桌子的显示：

顾客进入餐厅，我们会为他们提供就餐号（就餐号可以与顾客手机号进行绑定，用于查询相关服务或信息），当顾客走向平板电脑时需要输入他们的就餐号和用餐人数。此信息将被发送到我们的数据库，它会利用一个算法，合理地解决桌子分配问题。该算法首先按用餐人数多少作为优先考虑对象，它通过检查是否有人数相当并且可使用的桌子。如果没有可使用的桌子，那么下一个考虑的是顾客到来的时间，该算法通过检查数据库，判断谁进入餐厅越早谁的优先级就越高，等等。

在我们的算法结束后，在餐厅的前面显示客户的名称，并显示餐厅的桌子位置分布和他们将可使用的桌子。顾客将确认他们已收到通知并被服务员带到指定位置。此信息也会在服务员的设备中更新，以便让他们知道顾客的位置，如果顾客有很长时间没有交互信息可以根据座位表寻找客户。

当顾客用完餐后，顾客会进行买单。付款完成的过程表明，该桌子需要清洗，这将自动在应用程序通知服务员和清洁工知道哪个桌子需要收拾。当桌子被打扫完的时候，服务员会用一个简单的触摸按钮来让系统知道桌子已经打扫完毕。

实现订单的有序队列：

订单有序队列将是一种将服务生和厨师进行交互并同步显示订单的方式。它将包括一个用来存储顾客订单信息的数据库。当一个服务生提交一组订单时，所有的饭菜都将被放在数据库中队列的末端。厨师和服务员都能看到订单队列。厨师能够在设备上点击一道菜，并选择三种状态的一种：停止，进行，或完成。订单队列中也会显示，自从订单接单后，已经过去了多长时间，服务员可以通过准备时间来估计完成这道菜的时间。当厨师将一道菜的状态设置为完成时，它会向服务员发出一个指示，以便服务员知道它已经做好了，并将指定的食物是放置在服务员的窗口。此外，我们也能够从数据库队列中取出订单，减少数据库中数据量。

问题：解决客户点餐浪费的时间

许多来到餐馆的顾客都是饿着肚子来的，一旦坐下，客户开始一系列点餐用餐行为。作为餐厅的员工，确保服务质量，最大限度地减少所花费的时间是我们的职责。主要问题是时间：服务器需要处理的时间，下订单，下订单给厨师，上餐等时间。浪费了这么多时间，顾客会坐得很久，后到达的顾客就得等更久的时间才能用餐。为了最大限度地提高利润，并不断地希望客户惠顾，我们已经提出了解决方案，加快访问。创建一个功能菜单，有效地实现利润最大化，提高客户满意度。

我们餐厅为顾客提供智能化如苹果手机式的菜单。他们下单后，服务器采取让服务员下单给厨师的方式。我们的餐厅自动化应用程序也可以加快整个过程，最大限度地提高客户的满意度和餐厅利润。随着我们的餐厅自动化应用程序运行，我们有一个顾客自助服务菜单，服务器可以将订单直接发送给厨师，而不必交给服务员。有这个功能的菜单将明显加快了等待时间。一旦厨师完成订单，服务器将通过我们餐厅的自动化应用程序通知。被通知订单完成后，服务器将不会浪费时间检查订单，并可以指定更多的时间倾向于客户，倾向于客户的需求。虽然服务员可以在下订单时提好的建议，但是系统上有个功能可以让服务员不去打扰顾客点餐。所有的一切，将大大减少客户在餐厅用餐的时间，因此，可以让更多的顾客坐下来，最大限度地提高利润。

问题：结账时的时间浪费

想象一下，在支付过程中，有的人想要请客，有的人想AA制，这可能会导致时间上的消耗，在系统中会考虑支付方式，对几个人AA结账或个人结算都给出相应的方案，让顾客直接去选。这样顾客支付的时间的越少，等待的客人更快的来到座位，这样在方便顾客，提高顾客满意度的同时，也提高了我们餐厅的经营效率。还要加快支付的过程，顾客会带着微笑离开餐厅，并且喜欢我们的餐厅。

问题：高效的管理利润表

管理者希望看到他们的利润和损失，每当他们需要看利润表时。应用程序会生成一个当前的利润表和损益表。管理者通过检查财务问题和库存利润等问题来实现管理餐厅的预算和财务状态。这将使管理者可以为餐厅做更好的营销决策。

问题：菜单的添加和改变

作为应用程序的一部分，可以让管理者方便地添加和删除菜单上的菜品。我们希望管理者能看到更改后菜单的样式。

问题：员工出勤

管理者为了发现出勤问题，需要观察他们的工作时间表和每天出勤的情况。因此需要有一个员工门户的界面来进行相关操作。员工可以登录到平板电脑上的应用程序来查看日历并显示应出勤信息，此功能也可以为员工提供几个选项，为他们轮班进行更换等。

# 工作计划：

该小组有四名成员，夏瑞、杨帆、吴晨和张天霖。已经分成两队来最大限度地提高生产力。夏瑞和杨帆，吴晨和张天霖。每一队将有具体的任务，在未来的几周内，他们将做出自己的贡献。

夏瑞和杨帆：

1将在未来的几周时间里，完成登录界面（经理、清洁人员的登录界面），完成最主要功能------点餐功能。服务器为服务员和客户提供点餐的后台逻辑支持，实现从菜品展示、菜品选择\修改、菜品确认下单的一系列操作流程，并通过与app端的界面对接，使得这些功能能够真正展现在用户的面前，以简洁、高效的操作流程，完成点餐的功能。

2 尝试完成座位安排的后台逻辑，当用户向该系统请求座位的时候，系统按照一定的座位安排和排队等待算法，为用户分配座位或告知用户等待，若等待用户安排了座位后，系统将向用户推送消息，告知分配的座位信息。

3. 尝试完成支付逻辑。当客户发出支付请求时，系统会提供支付信息，客户根据信息完成支付。

4 厨师交由安卓端。

吴晨和张天霖：

安卓端页面布局设计：在页面设计时需要考虑后台逻辑，但充分的考虑只是为了最大可能地避免后期与服务器交互的逻辑冲突。在设计布局时完全抛弃后台逻辑，以不同角色的执行功能为出发点进行设计。

欢迎页面->选择“客户使用”按钮或者“工作人员使用”按钮。

工作人员使用，跳转到登录页面。

客户使用，跳转到客户页面。

1）登录页面：适用对象为：服务员、清洁工、厨师

以上几类角色共用一个登录页面，希望在数据库的user表中附带存储一项role的属性，便于在登录后及时提取转向不同角色的页面。

在有富余时间的情况下，欢迎页面、一台手机登录一次后就无需再重复登录的记住登录功能可以直接上线。

2）桌子的呈现方式：如果仅从UI的角度去考虑，理想化的情况是构建图形化的餐厅桌子情况，可以方便为客户、服务员、清洁工同时提供查看功能。后台建表应该维护桌子编号、桌子容量（人数）、桌子状态（空闲、占用、代打扫等）等属性信息

3）客户页面：2个输入框：名字和用餐人数的多少。一个按钮：确定。

返回指引信息。一个按钮：入座 。 点击后，进入下一页面

2个按钮：点餐、呼叫服务员

4）服务员页面

5）厨师页面

# 术语汇编：

## 专业术语：

数据库：按照数据结构来组织、存储和管理数据的仓库，用来存储菜品、菜单项等。

员工门户：员工可以访问的时间表，列出他们的应出勤时间和改变后的情况。

餐厅自动化：使用一个内部餐厅管理应用程序，并在餐厅内进行主要业务。

图形用户界面：允许通过使用图片和文本和用户进行很好通信的界面

订单队列：一个优先级队列，显示每一个订单的进展情况。

## 非专业术语：

清洁工：清理桌子，把脏盘子放在洗碗机，收拾桌子。

厨师：负责创建和规划菜单，进行食品准备，监督服务员上菜。

服务员：为客户下单，上菜，结账。

顾客：一个人或一方访问该餐厅吃饭，并可以点菜和地方保留从一个智能手机或平板电脑应用程序。

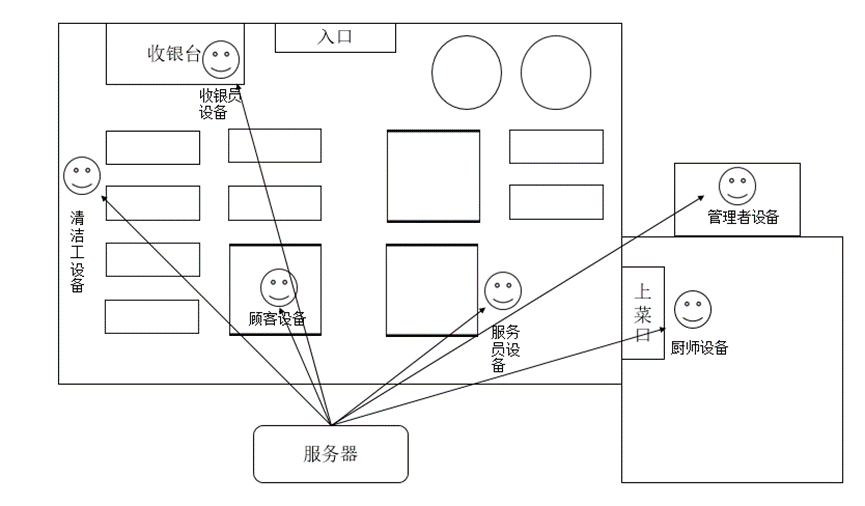
经理：负责库存管理、员工排程、薪资及客户满意度。

菜单：一个客户可以从中选择的菜品列表。

顾客满意：食品准备、客户服务和客户体验的衡量，超过顾客期望

利润：收入减去费用

# 用户故事：



## 复杂度：

为每个用户故事，我们为其评分3-7分。分值是粗略的估计需要多少精力来实现每个用户故事，根据团队的技术和每个用户故事的复杂度进行评判。

3分---可以作为其他需求的子需求，在实现其他需求的同时，就可自然完成

4分---正常需求、容易实现、界面简单

5分---正常需求、实现起来较为自然，不需要过多的逻辑、界面设计也为正常难度

6分---正常需求、实现需要比较多的逻辑、模块之间的调用情况多，界面设计有一定难度

7分---“炫酷”需求、需要复杂的逻辑处理或数据分析、界面制作复杂。

## 优先级：

ST-X-#: 高优先级 (最可能实现)

ST-X-#: 中优先级 (尽可能实现)

ST-X-#: 低优先级 (不太可能实现)

作为管理者：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-M-1 | 我可以根据收入和用户的反馈来修改菜单，这样我们能提供更灵活的菜品 | 6 |
| ST-M-2 | 我可以添加和雇佣一个职员 | 4 |
| ST-M-3 | 我可以简单的提取餐厅的年月周的利润和损失 | 4 |
| ST-M-4 | 我可以添加花费并修改雇员的薪水 | 4 |
| ST-M-5 | 我可以修改商品价格 | 4 |
| ST-M-6 | 我可以查看库存的有效期和数量 | 4 |
| ST-M-7 | 我可以根据员工排名赋予或夺回权利（升职降职） | 6 |
| ST-M-8 | 我可以查看员工的出勤情况 | 5 |
| ST-M-9 | 餐馆座位可调整（方便manager对餐馆进行硬件升级） | 7 |

作为厨师：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-Ch-1 | 我可以查看订单队列 | 5 |
| ST-Ch-2 | 我可以标记订单准备还是完成状态 | 5 |
| ST-Ch-3 | 我可以通过平板和服务员沟通 | 6 |
| ST-Ch-4 | 如果库存有限，我可以修改菜单 | 4 |
| ST-Ch-5 | 我可以修改更新供应库存告诉经理缺少什么配料 | 6 |

作为清洁工：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-B-1 | 我可以查看那张桌子是干净的 | 4 |
| ST-B-2 | 我可以标记脏的桌子并清洁 | 5 |

作为服务员：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-W-1 | 我可以快速的输入顾客的订单并提交给厨师 | 5 |
| ST-W-2 | 我可以看到顾客的账单并输入支付的信息 | 5 |
| ST-W-3 | 我可以使用优惠券或折扣 | 7 |
| ST-W-4 | 我可以标记现在脏的桌子让人清理 | 5 |
| ST-W-5 | 我可以将有人坐下的桌子改变状态 | 5 |
| ST-W-6 | 我可以根据用户需求添加特殊的指示 | 4 |

作为顾客：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-C-1 | 我可以通过智能机或平板点菜 | 5 |
| ST-C-2 | 我可以一键查看座位情况 | 6 |
| ST-C-3 | 我可以远程订座位 | 5 |
| ST-C-4 | 我可以提前让厨师做菜，告诉厨房何时制作订单 | 7 |
| ST-C-5 | 我可以输入团队的就餐号让平板告诉我们的桌子在哪里 | 6 |

作为员工：

|  |  |  |
| --- | --- | --- |
| 标记 | 用户故事 | 评分 |
| ST-E-1 | 我可以查看工作时间（轮班） | 5 |
| ST-E-2 | 我可以上班签到（登录退出） | 5 |

# 功能需求说明:

## 利益相关者

本项目包含以下几种利益相关者，分别是服务员、厨师、管理者、清洁工。顾客不是直接的系统利益相关者，但是他们的操作会影响整个系统的结果。另一组利益相关者包括所有参与开发和管理系统的人。利益来自于工作的设计和实施解决方案，来使系统完善。

角色和目标  
清洁工  
角色-负责餐厅的清洁工作的员工。  
目标-管理的桌子的状态，清洁桌子，在应用程序标记脏桌子。

厨师  
角色-负责厨师做饭和让服务员上菜。  
目标-管理仓库成分清单，改变订单的顺序队列，设置订单的状态，通知服务员。

客户  
角色-餐厅用餐者。  
目标-点餐（可以让厨师提前做），付账。

经理  
管理整个餐厅的员工。  
目标-管理员工和日程表，跟踪库存，跟踪利润/损失。

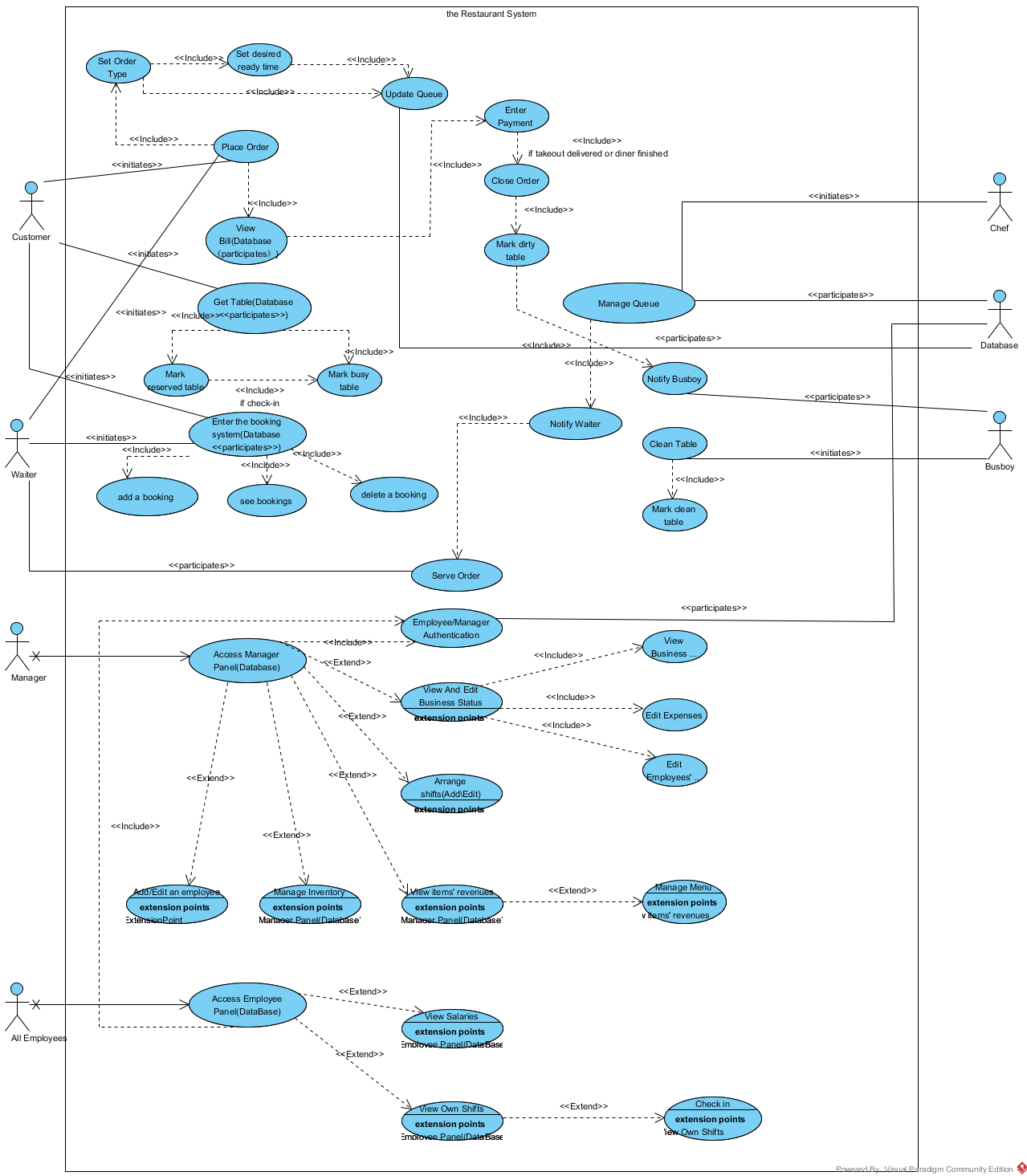
服务员  
角色-与客户互动的员工。  
目标-为客户下单，跟踪订单的顺序，并在客户中收取费用。

数据库（参与特殊角色）  
角色-存储系统所需信息的对象。  
目标-储存和修改库存，员工，利润，损失等信息。

# 用例：

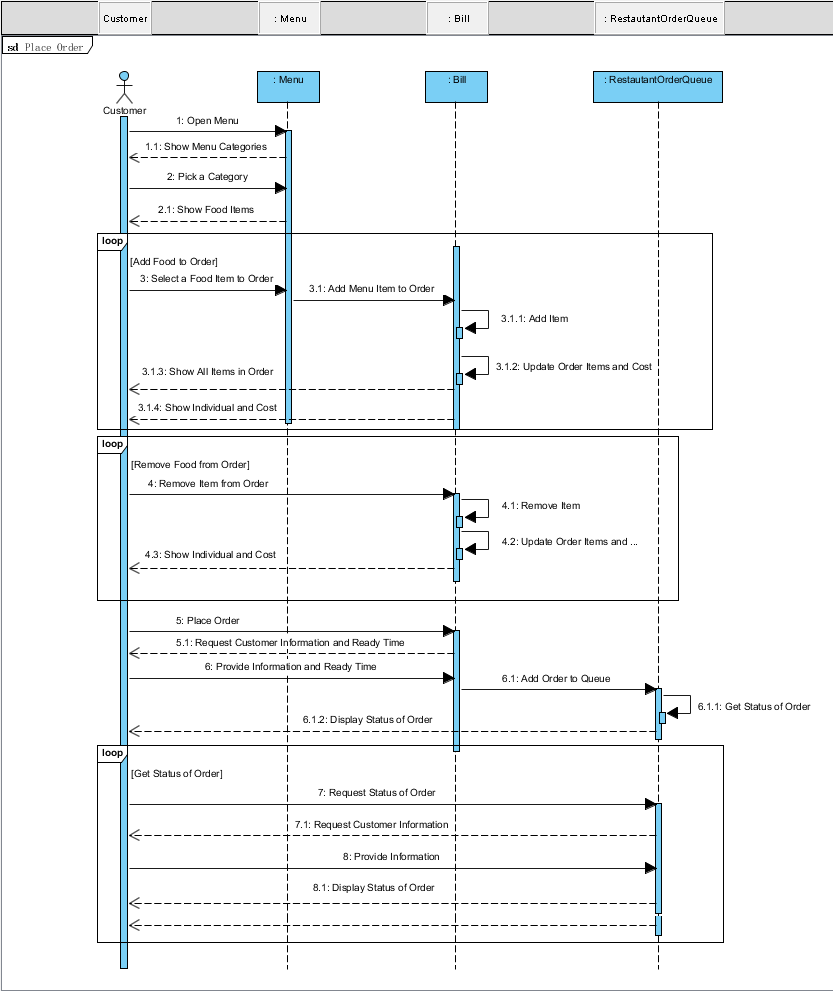
介绍：本小组的用户故事将应用在我们的用例上

## 整体用例图：



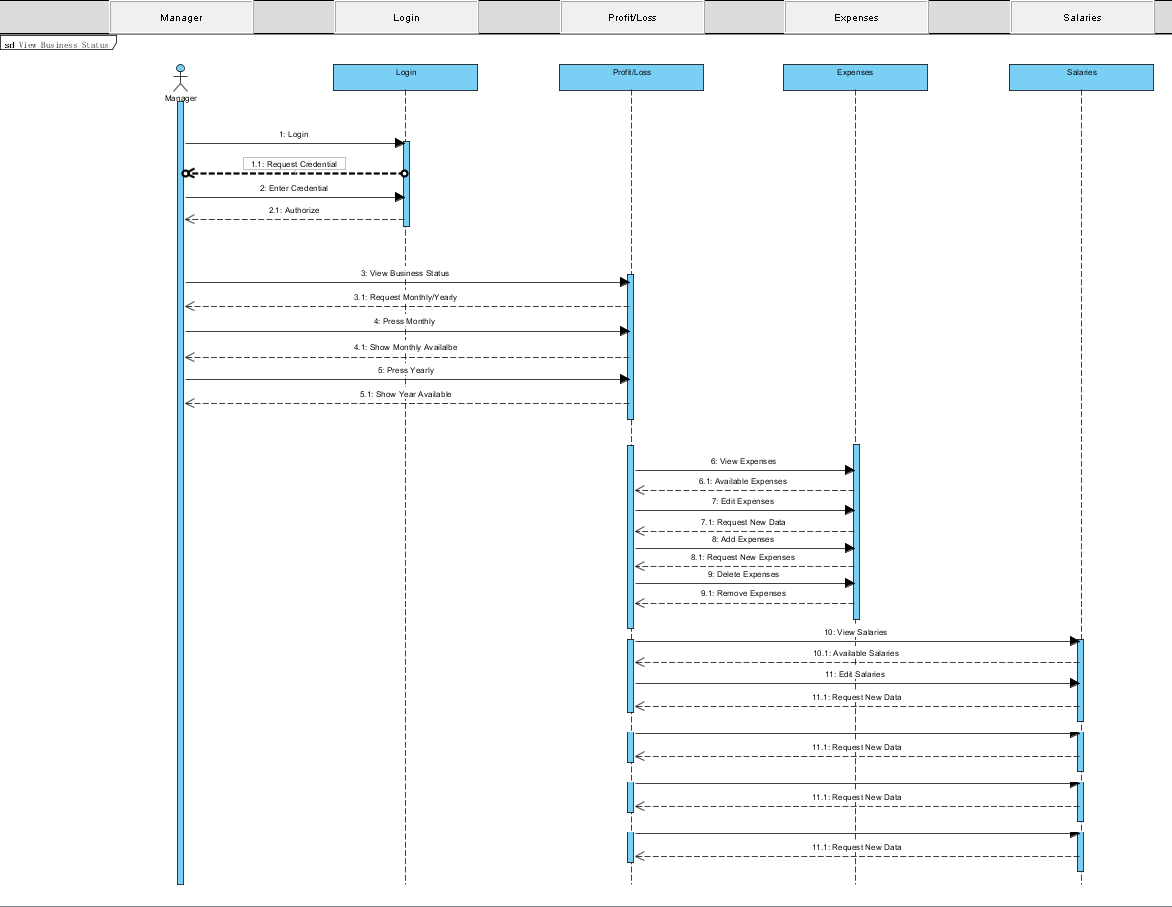
## 具体用例图：

用例1 顾客订餐



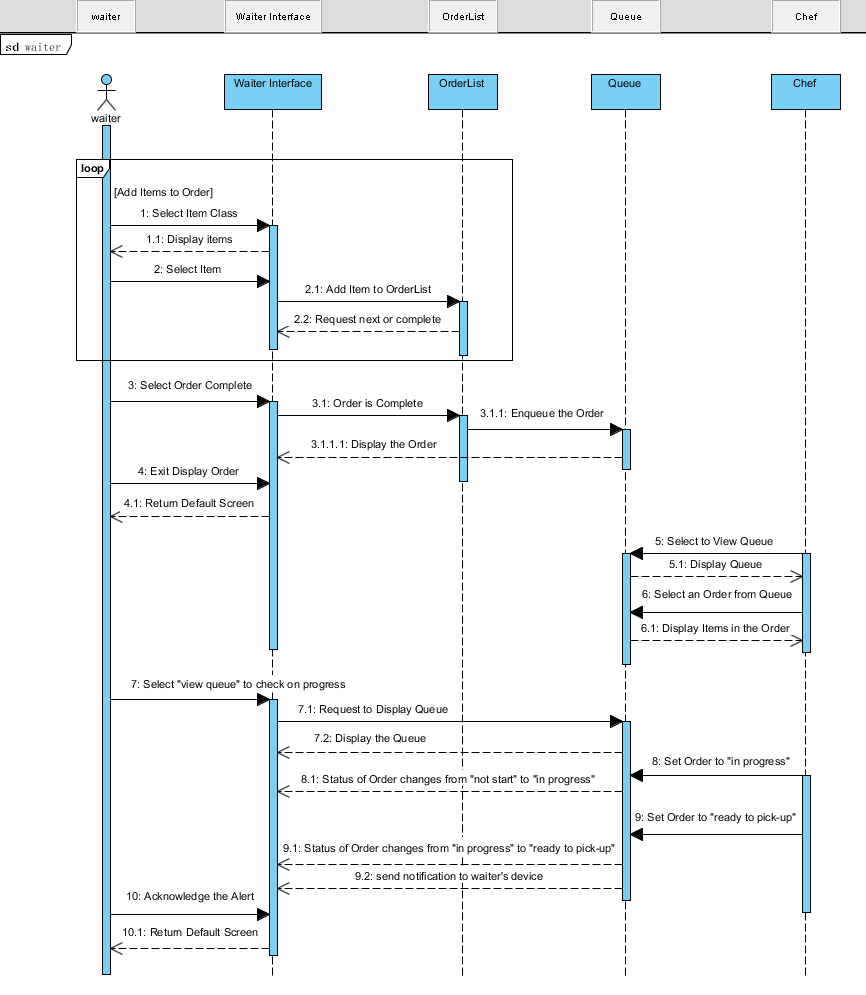
|  |  |
| --- | --- |
| 用例1 顾客订餐 |  |
| 主要角色 | 顾客 |
| 角色目标 | 用餐厅提供设备的程序在餐厅进行点餐 |
| 参与角色 | 服务员 |
| 前后状态 | 在平板显示菜品的详细情况，然后顾客进行点餐 |
| 具体事件详情 | 1.打开菜单  1.1展示菜单类型列表  2.选择一种类型  2.1展示食物详情  3.选择菜单的一种食物  3.1增加菜单食物到订单（显示订单的所有食物和特点及费用）  4.移出订单的食物（显示更新的所有食物和特点及费用）  5.下订单  5.1询问客户信息和上菜时间  6.提供客户信息和上菜时间  6.1增加订单到队列  6.2显示订单状态  7.请求订单状态  7.1请求顾客信息  8.提供信息  8.1展示订单状态 |

用例2 查看业务状态



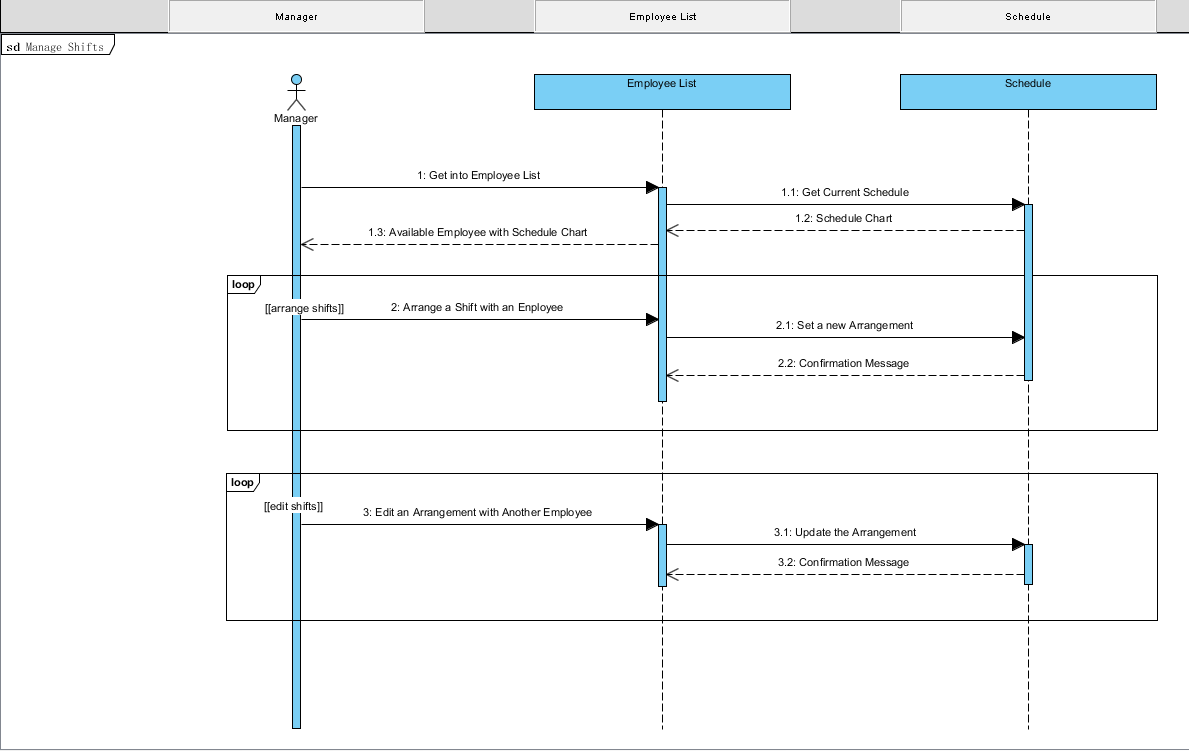
|  |  |
| --- | --- |
| 用例2 查看业务状态 |  |
| 主要角色 | 管理者 |
| 角色目标 | 查看经营的利润和损失，查看员工的薪水 |
| 参与角色 | 无 |
| 前后状态 | 管理者登入程序，允许管理者查看员工状态信息等 |
| 具体事件详情 | 1.登录  2.进入程序  3.查看业务状态  3.1请求月份、年份  4.输入月份  4.1显示月份信息  5.输入年份  5.1显示年份信息  6.查看费用  7.进入费用  8.增加费用  9.删除费用  10.查看薪水  10.1可选择项  11.进入薪水  11.1获取所需数据 |

用例3 管理订餐队列



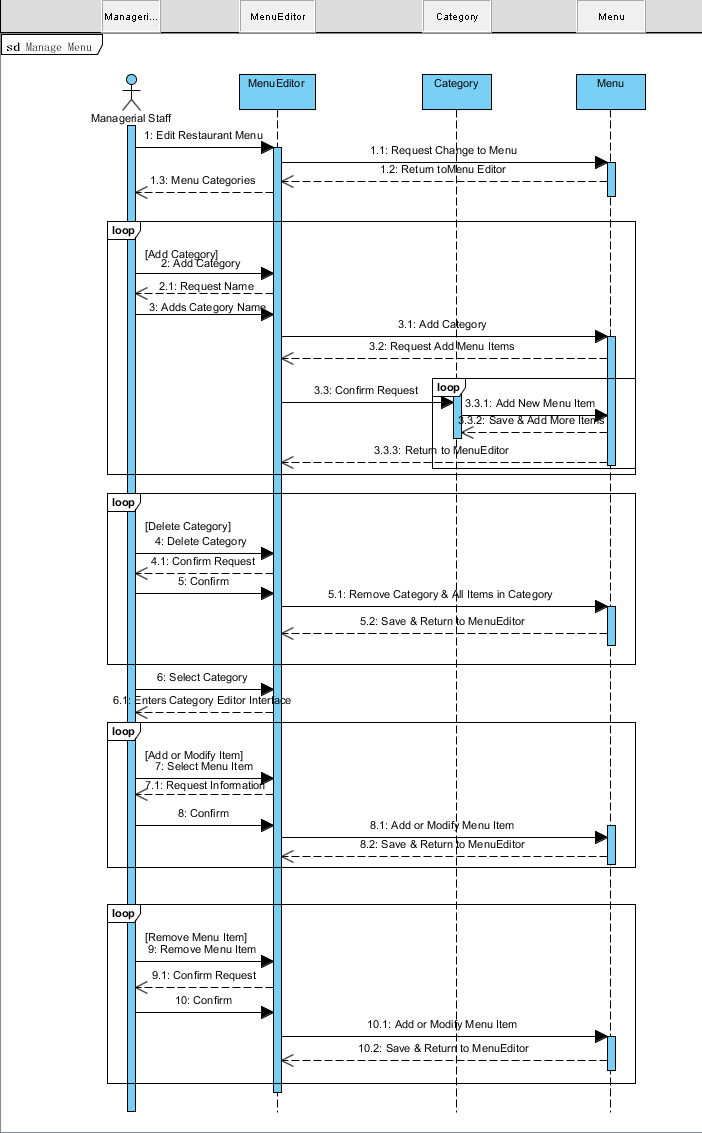
|  |  |
| --- | --- |
| 用例3 管理订餐队列 |  |
| 主要角色 | 服务员 |
| 角色目标 | 增加订单到订单队列 |
| 参与角色 | 厨师 |
| 前后状态 | 服务员用程序将顾客下单后的订单放到厨师的订单队列，厨师可以进行菜品状态的更换 |
| 具体事件详情 | 1.选择菜品  2.选择菜品  2.1增加到队列  2.2进行下次的请求  3.选择订单  3.1订单完成  3.1.1排列订单  4.退出显示订单  5.选择展示队列  6.选择展示订单  7.选择展示队列来检验  7.1请求展示队列  7.2显示队列  8.定义订单的状态是“未做”  9. 定义订单的状态“准备做”  9.1 定义订单的状态从“未做”到“准备做”  9.2和服务员设备进行传输  10.进行通知 |

用例4 管理工作调度



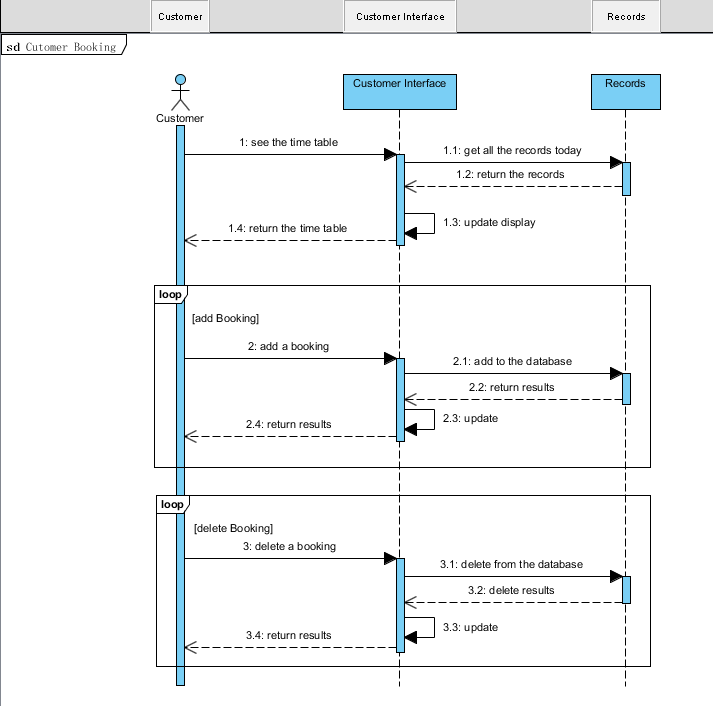
|  |  |
| --- | --- |
| 用例4 管理工作调度 |  |
| 主要角色 | 管理者 |
| 角色目标 | 查看员工出勤及更改相关信息 |
| 参与角色 | 员工 |
| 前后状态 | 登录程序后进行相关操作 |
| 具体事件详情 | 1.进入员工列表  1.1获得当前工作安排  1.2返回安排表  1.3获得可选择的员工工作表  2.进行员工安排的改变  2.1设置新的上班次序  2.2确认信息  3.进入其他员工安排  3.1更新信息  3.1确认信息 |

用例5 管理菜单



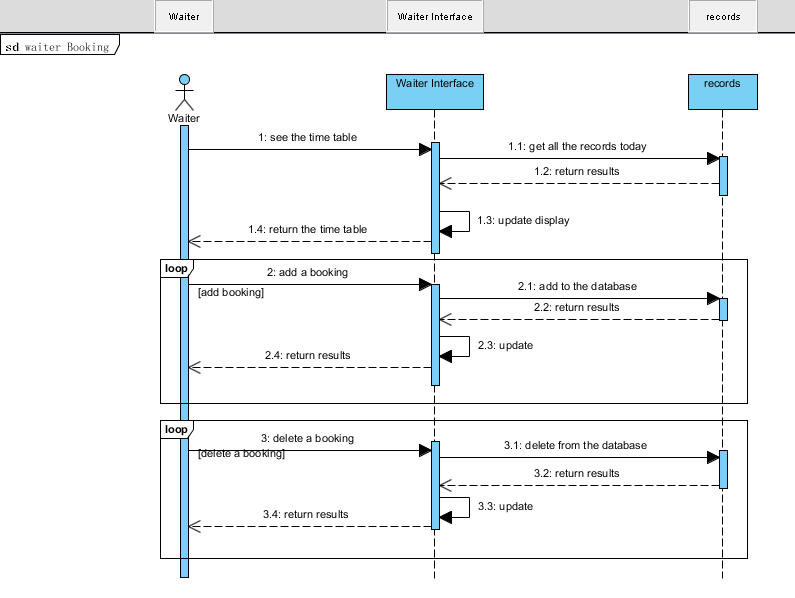
|  |  |
| --- | --- |
| 用例5 管理菜单 |  |
| 主要角色 | 管理者 厨师 |
| 角色目标 | 根据不同原因来改变菜单 |
| 参与角色 | 无 |
| 前后状态 | 进入程序的修改菜单选项来更新菜单 |
| 具体事件详情 | 1.进入餐厅菜单  1.1请求更改菜单  1.2进入菜单更改  1.3显示菜单类别  2.增加菜单类别  3.增加类别名字  3.1增加类别  3.2请求增加菜品  3.3确认请求  3.3.1更改增加菜品  3.3.2保存并增加  3.3.3返回菜单项  4.删除类别  5.确认删除  5.1删除菜品  5.2保存并返回菜单  6.选择菜单菜品  7.请求信息  8.确认  8.1增加或修改菜单菜品  8.2保存并返回  9删除菜品  10.确认  10.1删除菜品  10.2保存并返回 |

用例6 顾客预订座位



|  |  |
| --- | --- |
| 用例6 顾客预订座位 |  |
| 主要角色 | 顾客 |
| 角色目标 | 顾客进行座位的预订 |
| 参与角色 | 无 |
| 前后状态 | 顾客进入界面根据时间表预订座位 |
| 具体事件详情 | 1.看时间表  1.1获得今天的记录  1.2返回数据  1.3更新显示  1.4返回时间表  2.增加一个预订  2.1增加到数据库  2.2返回结果  2.3更新  2.4返回结果  3.删除一个预订  3.1从数据库删除  3.2删除结果  3.3更新  3.4返回结果 |

用例7 服务员预订座位



|  |  |
| --- | --- |
| 用例7 服务员预订座位 |  |
| 主要角色 | 服务员 |
| 角色目标 | 服务员进行座位的预订 |
| 参与角色 | 无 |
| 前后状态 | 服务员进入界面根据时间表预订座位 |
| 具体事件详情 | 1.看时间表  1.1获得今天的记录  1.2返回数据  1.3更新显示  1.4返回时间表  2.增加一个预订  2.1增加到数据库  2.2返回结果  2.3更新  2.4返回结果  3.删除一个预订  3.1从数据库删除  3.2删除结果  3.3更新  3.4返回结果 |

## 追溯矩阵

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 用户故事 | UC-1 | UC-2 | UC-3 | UC-4 | UC-5 | UC-6 | UC-7 |
| ST-M-1 |  |  |  |  |  |  |  |
| ST-M-2 |  |  |  |  |  |  |  |
| ST-M-3 |  |  |  |  |  |  |  |
| ST-M-4 |  |  |  |  |  |  |  |
| ST-M-5 |  |  |  |  |  |  |  |
| ST-M-6 |  |  |  |  |  |  |  |
| ST-M-7 |  |  |  |  |  |  |  |
| ST-M-8 |  |  |  |  |  |  |  |
| ST-M-9 |  |  |  |  |  |  |  |
| ST-Ch-1 |  |  |  |  |  |  |  |
| ST-Ch-2 |  |  |  |  |  |  |  |
| ST-Ch-3 |  |  |  |  |  |  |  |
| ST-Ch-4 |  |  |  |  |  |  |  |
| ST-Ch-5 |  |  |  |  |  |  |  |
| ST-B-1 |  |  |  |  |  |  |  |
| ST-B-2 |  |  |  |  |  |  |  |
| ST-W-1 |  |  |  |  |  |  |  |
| ST-W-2 |  |  |  |  |  |  |  |
| ST-W-3 |  |  |  |  |  |  |  |
| ST-W-4 |  |  |  |  |  |  |  |
| ST-W-5 |  |  |  |  |  |  |  |
| ST-W-6 |  |  |  |  |  |  |  |
| ST-C-1 |  |  |  |  |  |  |  |
| ST-C-2 |  |  |  |  |  |  |  |
| ST-C-3 |  |  |  |  |  |  |  |
| ST-C-4 |  |  |  |  |  |  |  |
| ST-C-5 |  |  |  |  |  |  |  |
| ST-E-1 |  |  |  |  |  |  |  |
| ST-E-2 |  |  |  |  |  |  |  |

# 领域分析:

## 领域模型

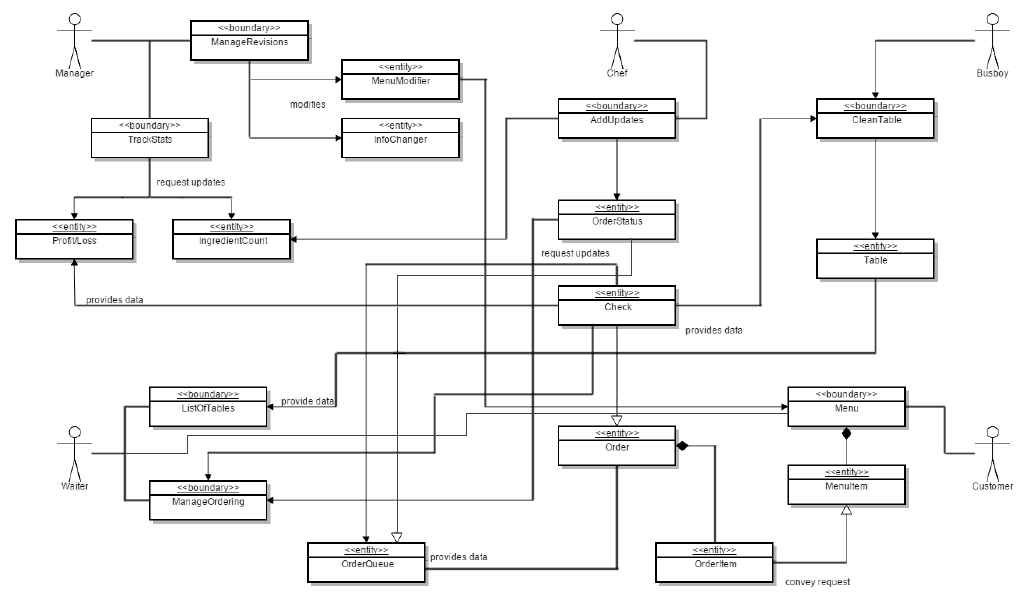
### 1.概念定义

|  |  |  |
| --- | --- | --- |
| 责任描述 | 类型 | 概念 |
| 查看日程表，看收益损失，调动员工工作安排 | D | Manager |
| 系统知道所有花费、收益和损失 | K | Profit/loss |
| 可编辑费用 | D | Manager |
| 顾客进入菜单进行点餐 | D | OrderItem |
| 知道顾客的订单 | K | OrderQueue |
| 系统知道和展示订单状态 | K | Orderstatus |
| 改变员工信息 | D | Manager |
| 更变菜单 | D | MenuModifier |
| 点餐 | D | Waiter/customer |
| 系统知道配料的数量 | K | IngredientCount |
| 系统增加到队列 | D | Check |
| 预定座位 | D | Waiter/customer |
| 系统排列订单状态 | K | Orderstatus |
| 厨师更改订单状态 | D | Orderstatus |
| 系统统计账单 | K | Check |
| 系统收账单并更改桌子状态 | D | Table |
| 服务员更改桌子状态 | D | Table |
| 系统知道员工日程表 | K | Schedule |

### 2.关联定义

|  |  |  |
| --- | --- | --- |
| 概念相关 | 关联描述 | 关联名 |
| Manager ↔ MenuModifer | 允许管理者更改菜单 | Modifies |
| Manager ↔ Profit/Loss | 允许管理者查看利润损失 | Provides Data |
| Manager ↔ IngredientCount | 允许管理者查看配料数量 | Requests Updates |
| Customer ↔ Waiter | 顾客提供订单请求向服务员 | Conveys Requests |
| Waiter ↔ Chef | 服务员提供订单给厨师 | Conveys Requests |
| Customer ↔ OrderItem | 顾客点餐 | Conveys Requests/Requests Save |
| Check ↔ OrderQueue | 把订单放入队列 | Provide queue |
| Waiter ↔ OrderStatus | 服务员更新订单状态 | Requests Updates |
| Chef ↔ OrderStatus | 厨师改变订单状态 | Provides Data |
| Waiter ↔ Check | 服务员订座 | Conveys Requests/Provides Data |
| Waiter ↔ Customer | 服务员请求支付 | Provides Data |
| Waiter ↔ Tables | 服务员更新桌子状态 | Provides Data |

### 3.领域模型图



# 算法：

## 获取分析数据

### 1.产品的收益

Sellmap[productId,sells]

for(All orderitems in the db)

{

//pick up the productId,

//find the entry in sellmap

sells++

}

Show(sellmap)

### 2.菜品的受欢迎程度

SortMapBysells(sellMap);

//put top10 and bottom 10 items

### 3.就餐时长

时长：（1）30分钟以下 （2）30分钟-60分钟 （3）60分钟-90分钟 （4）90分钟-120分钟 （5）120分钟以上 对应period[5]

for (All orders in period)

{

time1=place\_order\_timestamp;

time2=pay\_bill\_timestamp;

duration=time-time1;

Switch(duration){

//put this order into its group

}

}

ShowAsChart(period);

### 4.等待订单的时长

for(All orders need to be calculate)

{

waitTime=0;

lastDish=null//Because we have not provide food to customer before their first dish comes.Maybe the restaurant offer some food for

customer on free,in order to decrease the waiting time.In that case,the lastDish can be initialized by a not-null object.

Current =0 //The relative time to the order-start timestamp

For(every orderItem in the order Order By serveTime Asc)

{

currentDish = orderItem.getDish;

if(lastDish.eatTime<orderItem.serveTime){

//last dish may have been eaten up.while the next dish has not been served.the customer may wait.

waitTime=orderItem.serveTime-current-lastDish.eatTime

}//End if

Else{

//the customer may eating the food while the next dish is coming.No wait time.

}

lastDish = currentDish;

current = orderItem.serveTime

}//End for

}//End for

//calculate the average waitTime and show the waitTime distribute Chart

## 动态加载时间表算法：

private LinearLayout getTimeTableView(List<TimeTableModel> model) {

LinearLayout mTimeTableView = new LinearLayout(getContext());

mTimeTableView.setOrientation(VERTICAL);

int modesize = model.size();

if (modesize <= 0) {

mTimeTableView.addView(addStartView(MAXNUM));

} else

for (int i = 0; i < modesize; i++) {

if (i == 0) {

mTimeTableView.addView(addStartView(model.get(0)

.getStartnum()));

mTimeTableView.addView(getMode(model.get(0)));

} else if (model.get(i).getStartnum()

- model.get(i - 1).getStartnum() > 0) {

mTimeTableView.addView(addStartView(model.get(i)

.getStartnum() - model.get(i - 1).getStartnum()-(model.get(i-1).getEndnum()-model.get(i-1).getStartnum())));//减去持续的时间 modifed by WuChen 3.24

mTimeTableView.addView(getMode(model.get(i)));

}

if (i + 1 == modesize) {

mTimeTableView.addView(addStartView(MAXNUM

- model.get(i).getEndnum()));

}

}

return mTimeTableView;

}