软件设计综合实验

外卖订餐软件

软件设计文档

项目组成员：

黄雄镖 BillBillBillBill 13331093

郑煦 wrongthings 13331363

许莉 Kulikuly 13331300

徐航 Mr-Leaves 13331296

目录

1 外卖订餐系统需求分析

1.1 外卖订餐系统问题陈述

在日常生活中，外卖订餐与我们的生活及饮食息息相关。我们小组开发的是一个外卖订餐系统，帮助人们解决不想外出就餐的烦恼，提供线上订餐，线下配送的便利。

在外卖订餐系统上，顾客通过浏览多个商家的菜单，选择食品进行订餐。商家通过访问系统编辑自家菜单，处理外卖订餐信息，并进行线下配送。管理员维护顾客与商家的信息。

顾客能够通过绑定手机号码与设置密码在外卖订餐系统上注册。顾客通过登录订餐系统，进行浏览菜单、点餐、查看订单、编辑个人信息等操作。有关商家的简介、食品的价格、评价等会呈现给顾客，作为他们选择食品的参考信息。

顾客选择一家心仪的商家进行点餐，添加或删除食品，选择完毕后提交订单。已提交的订单保存在顾客的订单列表，顾客可以通过查看订单了解订单的状态，包括商家接单与否，配送与否等状态。顾客线下收到外卖后，在系统上选择确认收货，此时订单状态更改为已完成。在此之后，顾客可以对订单中的食品进行评价或者投诉。管理员对顾客的反馈进行管理。

商家在线下向管理员申请并提交相关证明，办理相关手续后，管理员将商家注册在系统上。已在系统上注册的商家能够登录订餐系统，编辑联系电话，地址等基本资料，以及编辑自家的菜单，包括在菜单上添加、删除食品，修改食品价格与图片。

商家收到顾客提交的订单后处理订单，若选择拒绝订单，则将订单状态设置为已关闭；若选择接受订单，则将订单状态设置为已接单，并在配送后将订单状态设置为已配送。订单状态为未处理时，顾客可以取消订单。顾客确认收货后，订单状态更新为已完成。顾客在用餐后可以对订单进行评价，评价会更新至商家评价栏目；也可以对订单进行投诉，投诉将会反馈给管理员。

当顾客对此次外卖订餐有任何疑问时，可以致电商家或管理员向其提出。我们小组开发的外卖订餐系统目前只支持货到付款的支付方式，这在目前的阶段，避免了系统线上支付带来的安全性问题，以及先付款后收货导致的外卖放楼下被故意拿走或拿错现象的出现。

1.2 外卖订餐系统用例析取

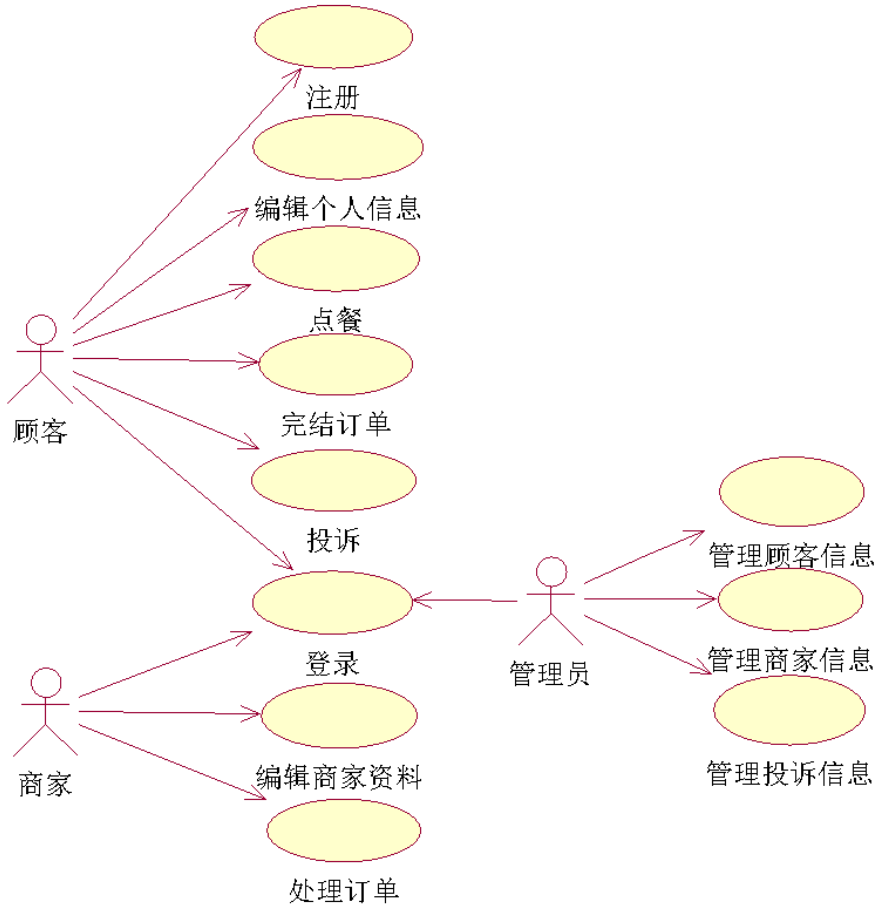

图 1 外卖订餐系统用例析取

1.3 外卖订餐系统用例规约

1.3.1 点餐用例规约

1. 简要说明

本用例允许顾客浏览各个商家的菜单，选择食品。在顾客点餐时，可以看到各个商家提供的菜单信息，添加/删除食品。

2. 参与者

顾客。

3. 事件流

3.1 基本事件流

用例开始于顾客进行点餐。

1）系统要求顾客从系统所提供的商家中选择一个进行点餐

2）一旦顾客选择了一个进行本次点餐的商家后，选择食品子事件流将被执行

3.1.1 选择食品

1）系统向顾客提供所选商家的菜单，并在菜单中每一道食品后面提供“添加”与“删除”按钮

2）顾客选择食品，点击“添加”或“删除”按钮

3）顾客选择食品完毕后，点击“编辑选择完毕”按钮，进入编辑订单子事件流

3.1.2 编辑订单

1）系统向顾客提供已选的食品项目及数量，顾客填写收货人姓名、联系电话、收货地址，并根据需要增减订单中食品项目的数量

2）顾客编辑订单完毕后，进入提交订单子事件流

3.1.3 提交订单

1）系统检验订单满足订单填写完整的条件，包括收货人姓名，联系电话与收货地址均填写完整，并且所选食品不为空

2）如果订单填写完整，系统将订单提交给商家，并且将订单保存在顾客已提交订单列表中

3.2 备选事件流

3.2.1 不满足订单填写完整的条件

如果在提交订单子事件流中，收货人姓名、联系电话与收货地址未填写完整，或者所选食品项目为空，系统将提示一条错误信息，顾客可选择继续编辑订单。

3.2.2 放弃订单

顾客未提交订单前可以放弃已编辑的订单，此时本用例重新开始。

4. 特殊要求

无

5. 前置条件

本用例开始前顾客必须先登录进系统。

6. 后置条件

如果用例成功，订单被提交给商家处理，并且保存在顾客的订单列表中。否则，此订单不在顾客列表中被保存，也不向商家提交。

7. 活动图

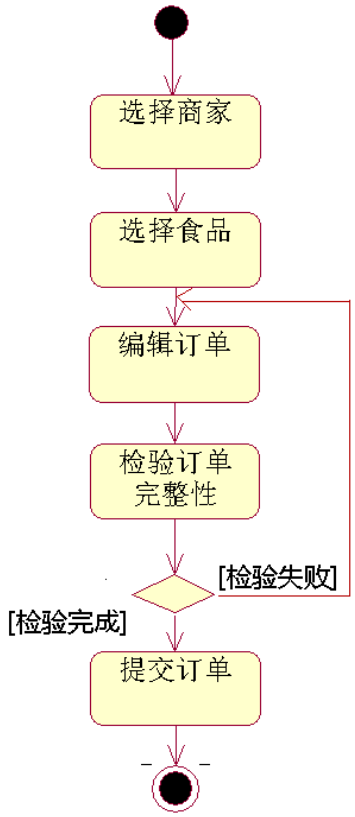

图 2 点餐用例活动图

1.3.2 完结订单用例规约

1. 简要说明

本用例允许顾客修改自己已提交的订单的状态，进行确认收货、评价或取消订单等操作。

2. 参与者

顾客。

3. 事件流

3.1 基本事件流

用例开始于顾客对已成功提交的订单进行操作。

1）订单状态共有五种，分别为“未处理”、“已关闭”、“已接单”、“已配送”和“已完成”。顾客可以在相应的订单状态下进行操作，选择取消订单、确认收货、评价或投诉订单。

2）如果订单状态为“未处理”，顾客等待商家接单并配送，等待订单状态更新

3）如果订单状态为“已配送”，顾客可以选择确认收货，则进入确认收货子事件流

4）如果订单状态为“已完成”，顾客可以选择对订单进行评价与否。如果选择“评价”，则进入评价子事件流；如果选择“不评价”，则本用例结束

3.1.1 确认收货

1）订单状态为“已配送”时，顾客点击“确认收货”按钮，确认收到食品

2）系统将订单状态从“已配送”更新为“已完成”

3）系统要求顾客点击所要进行的操作，如果顾客点击“评价”，则进入评价子事件流；如果点击“投诉”，则进入投诉子事件流。

3.1.2 评价

1）顾客填写评价信息，对本次订单中的食品或服务进行评价。

2）顾客填写评价信息完毕后点击“提交”按钮，评价信息出现在相应的商家评价栏目中，本用例结束

3.2 备选事件流

3.2.1 取消订单

如果订单状态为“未处理”，顾客可以选择“取消订单”。如果顾客点击“取消订单”按钮，订单状态更新为“已关闭”并且不会被商家处理，本用例结束。

4. 特殊要求

无

5. 前置条件

本用例开始前顾客须已成功提交订单。

6. 后置条件

如果用例成功，订单状态更改为“已完成”，顾客选择性的进行评价。

7. 活动图

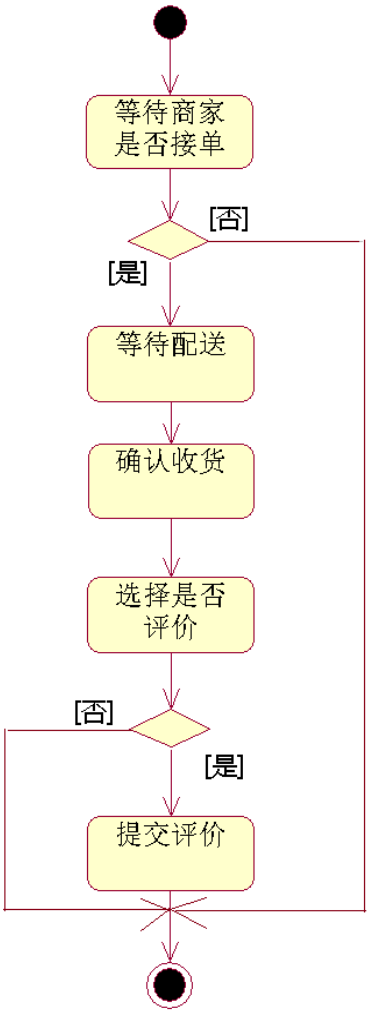

图 3 完结订单用例活动图

1.3.3 处理订单用例规约

1. 简要说明

商家可以查看并处理顾客提交的订单，将订单状态更新为“已接单”、“已关闭”、“已配送”。

2. 参与者

商家

3. 事件流

3.1 基本事件流

用例开始于商家对顾客提交的订单进行处理。

1）系统向商家提供顾客提交的订单，并要求商家选择“接受订单”或“拒绝订单”

2）如果商家选择“拒绝订单”，此时进入拒绝订单子事件流

3）如果商家选择“接受订单”，此时进入接受订单子事件流

3.1.1 拒绝订单

1）订单状态从“未处理”更新为“已关闭”并反馈给用户，商家不再对此订单进行更多操作，本用例结束

3.1.2 接受订单

1）订单状态从“未处理”更新为“已接单”，并反馈给用户

2）商家准备食品，配送后将订单状态更新为“已配送”，本用例结束

3.2 备选事件流

3.2.1 商家收到顾客线下取消订单请求

如果商家收到顾客取消订单的请求，商家根据自身情况同意或拒绝。若同意，则商家将订单状态更新为“已关闭”；如果商家不同意，则订单状态不发生改变，本用例继续。

4. 特殊需求

无

5. 前置条件

本用例开始前商家收到顾客的订单。

6. 后置条件

商家处理订单完毕。

7. 活动图

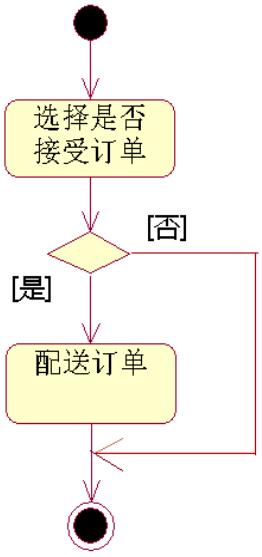

图 4 处理订单用例活动图

1.3.4 外卖订餐系统补充规约

1. 目标

本文档的目的是定义订餐系统的需求。该补充规约部分补充说明了在用例规约中未列出的系统需求，即除了系统功能性需求以外的非功能性需求，如兼容性、可靠性等。

2. 兼容性

系统 Web 界面与 chrome、IE、Firefox 等多种浏览器兼容。

3. 可靠性

本系统提供 24 小时不间断服务。

4. 性能

1） 系统使用多进程+多协程的模型，大大提高并发性。

2） 系统经过压力测试，支持两百并发访问 Web 页面。

5. 安全性

1）系统必须防止用户信息外泄或被非法篡改。

2）系统用户分为顾客、商家及管理员三种角色，每种角色的具有相应权限，系统必须保证用户权限不能超出对应权限范围。

6. 设计约束

暂无

1.4 术语表

本部分内容包括与本系统开发相关的关键概念定义。

1. 外卖订餐系统

本次我们小组的开发任务。本系统是顾客进行线上外卖订餐的系统，为商家提供编辑菜单、处理订单等功能，为顾客提供点餐、完结订单、编辑个人信息等功能，为管理员提供维护商家信息、维护顾客信息等功能。下文所称本系统均指外卖订餐系统。

2. 顾客

指在本系统上点餐的用户。

3. 商家

从事商业活动的个人或组织，在本系统上提供为顾客提供食品。

4. 菜单

商家在本系统中编辑后，显示在商户首页的可供用户选择的食品列表。

5. 订单

顾客所要购买的商品凭据，包括商品种类、数量、单价和总价，以及顾客的联系方式等信息。

6. 配送

商家为顾客制作食品后，按照顾客填写的信息，将食品送到指定地点。

2 外卖订餐系统架构设计

2.1 外卖订餐系统架构描述

本外卖订餐系统采用基于 C/S 的架构，代码的组织方式为 MVC 三层结构，其中的三个层次分别为视图层（View）、控制器层（Controller）和模型层（Model）。代码整体采取前后端分离的方式，前端负责视图层和控制器层，后端负责模型层，客户端作为 SPA 实现，前后端通讯使用 REST 的方式。

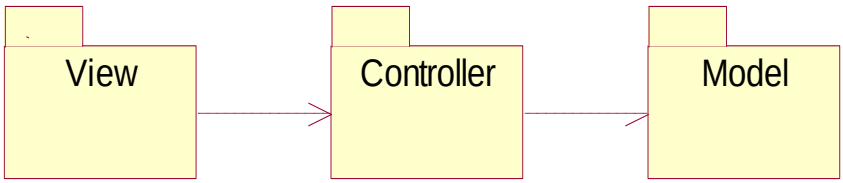

图 5 外卖订餐系统架构图

2.1.1 视图层（View）

视图层主要负责页面的渲染，系统与用户的交互。客户端使用 HTML/CSS/JavaScript 实现，其中用到了 Foundation 和 Vue.js 框架。Vue.js 在前端页面提供了路由功能，通过调用服务端提供的 RESTful API 进行通讯。

2.1.2 控制器层（Controller）

控制器层主要采用 Django REST framework 实现，是实现系统业务逻辑的核心，该层只需实现相应的 RESTful API。该层主要接收来自客户端的请求，经过处理后由 REST 接口提供数据给表现层。其中功能包括注册、登陆、获取商家列表、获取食品列表、下单等。

2.1.3 模型层（Model）

模型层实现本系统实体对象的数据库访问，提供了数据库的连接，对象关系模型以及数据的持久化服务，使用 Django 提供的 model 来以 ORM 的方式将对象与数据库表关联，数据库的借口被封装在 ORM 机制内部，避免了 SQL 注入攻击。该层定义了 Customer, Seller, Food, Order 等模型。所使用的数据库有 Redis 与 SQLite，Redis 作为缓存数据库，SQLite 作为数据主数据库。

3 外卖订餐软件设计技术

技术选型

我们小组目前完成的外卖订餐软件是使用前后端分离技术开发的 WEB 应用。前端是一个用 vue.js 实现的单页应用，响应速度非常快，仅通过 API 与后端 API 服务器进行交互，后端 API 服务器使用了 Django REST framework，并使用 Nginx 作为反向代理服务器，使用 Gunicorn 作为 WSGI HTTP 服务器，利用 Gevent 作为 Gunicorn 的 worker，使用多进程+多协程的模型大大提高了并发性。

附压力测试图：

使用两百并发访问前端页面，处理数量一秒 2670。

```
bill@bill-pc ~ ab -c 200 -n 10000 http://takeout.com/business/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking takeout.com (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        nginx/1.9.10
Server Hostname:        takeout.com
Server Port:            80

Document Path:          /business/
Document Length:        626 bytes

Concurrency Level:      200
Time taken for tests:   3.745 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      9220000 bytes
HTML transferred:       6260000 bytes
Requests per second:    2670.40 [#/sec] (mean)
Time per request:       74.895 [ms] (mean)
Time per request:       0.374 [ms] (mean, across all concurrent requests)
Transfer rate:          2404.40 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0  14.1      0     998
Processing:    42   74  24.2     70     195
Waiting:       42   74  24.2     70     195
Total:         42   74  27.9     70    1056
```

图 6 压力测试图

使用一百并发访问 API 服务器较为繁重的 store，处理数量一秒 138。

```
✗ bill@bill-pc ~ ab -c 100 -n 1000 http://takeout.com/api/business/store
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking takeout.com (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        nginx/1.9.10
Server Hostname:        takeout.com
Server Port:            80

Document Path:          /api/business/store
Document Length:        1534 bytes

Concurrency Level:      100
Time taken for tests:   7.228 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      1755000 bytes
HTML transferred:       1534000 bytes
Requests per second:    138.35 [#/sec] (mean)
Time per request:       722.800 [ms] (mean)
Time per request:       7.228 [ms] (mean, across all concurrent requests)
Transfer rate:          237.11 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   3.4      0      12
Processing:    30  679 546.3    538    3110
Waiting:       30  679 546.3    538    3110
Total:         31  680 546.9    538    3110
```

图 7 压力测试图

3.1 Object-Oriented Programming

3.1.1 技术说明

面向对象编程的概念在整个项目中都有体现，后端使用的 Python 语言中一切皆为对象，在后台与数据库交互上，我们使用了 ORM（对象关系映射）技术，实现面向对象编程语言里不同类型系统的数据之间的转换，提供的不只是描述不同对象间关系的一个简单而直接的方式，也提高了灵活性。

3.1.2 具体描述

整个项目

3.2 Aspect-Oriented Programming

3.2.1 技术说明

AOP 主要实现的目的是针对业务处理过程中的切面进行提取，它所面对的是处理过程中的某个步骤或阶段，以获得逻辑过程中各部分之间低耦合性的隔离效果。在项目中，对于请求使用中间件进行了预处理，如处理 Token 验证，JSON 和 QueryString 的转换等，避免了大量的代码冗余情况。

3.2.2 具体描述

中间件模块：takeout/middlewares.py

3.3 Service Oriented Architecture

3.3.1 技术说明

在后端编程中，按照模块对代码进行划分，并针对抽出公用代码，降低了项目整体的耦合度，服务之间的关系最小化，提高了内聚性。

在前端编程中，采用组件式开发方式，大大地提高了代码的复用程度。

3.3.2 具体描述

整个项目

3.4 Design Patterns

3.4.1 技术说明

项目中使用了部分设计模式进行开发，如抽象工厂模式等。

3.4.2 具体描述

抽象工厂模式：系统中分为三类用户，而三类用户行为上都有相同的地方，如注册、登录等操作，因此我们使用了一个名为 UserBase 的 Model 作为基类，Admin，Seller，User 都继承于该类。

对应模块及行数：

lib/models/userbase.py:5

admin/models/admin.py:5

bussiness/models/seller.py:6

customer/models/customer.py:7

4 外卖订餐软件模块划分

我们小组将外卖订餐系统划分为三个功能逻辑上相对独立的子系统模块，分别为商家管理模块，顾客管理模块，管理员管理模块。

4.1 商家管理模块设计

描述：商家管理模块

视图层（View）包括的类有：

About App Detail Home Login Navbar Order Register

控制器层（Controller）包括的类有：

SellerList SellerDetail StoreList StoreDetail FoodList FoodDetail

模型层（Model）包括的类有：

Food Seller Store Review FoodReview OrderReview

4.2 顾客管理模块设计

描述：顾客管理模块

视图层（View）包括的类有：

About App Comment Detail Home Login Navbar Orbitimages Order Orderdetail Register Sortbar

控制器层（Controller）包括的类有：

CustomerList CustomerDetail DeliveryInformationList DeliveryInformationDetail ComplaintList ComplaintDetail OrderList OrderDetail

模型层（Model）包括的类有：

Customer DeliveryInformation Order OrderFood Complaint Review FoodReview OrderReview

4.3 管理员管理模块设计

描述：管理员管理模块

视图层（View）包括的类有：

About Addbussiness App Complain Home Login Navbar Register

控制器层（Controller）包括的类有：

AdminDetail AdminList

模型层（Model）包括的类有：

Admin

5 外卖订餐系统部署设计

5.1 外卖订餐系统部署图

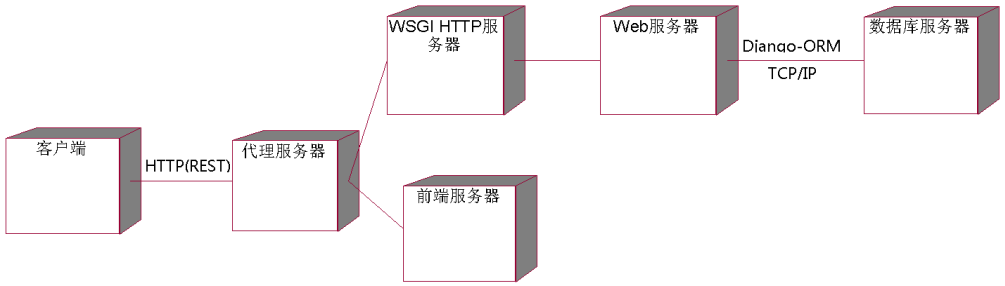

图 8 外卖订餐系统部署图

注：本项目同时为我们小组成员在系统分析与设计课程中设计完成的项目