



“摊易选”地摊管理系统



数据库课程设计报告

----- “摊易选”地摊管理系统

计算机科学与技术学院 计算机科学与技术专业 2019 级 4 班

姓名: 潘煜东 学号: 201800413074

任课教师: 实验教师: 助教:



“摊易选”地摊管理系统

前言	3
一、项目简介	5
二、系统开发平台	6
三、数据库规划	8
3.1 任务陈述	8
3.2 任务目标	9
四、系统定义	10
4.1 系统边界	10
4.2 用户视图	12
五、需求分析	13
5.1 用户需求说明	13
5.1.1 数据需求	13
5.1.2 事务需求	14
5.2 系统需求说明	16
六、数据库逻辑设计	17
6.1 简述	17
6.2 ER 图	18
6.3 数据字典	19
6.3.1 从数据字典中提取出来的实体描述	19
6.3.2 从数据字典中提取出来的逻辑描述	21
6.4 数据库逻辑简要描述	21
6.4 数据库更新	22
七、数据库物理设计	23
7.1 索引	23
7.2 视图	24
7.3 安全机制	25
7.3.1 系统安全	25
7.3.2 数据安全	25
八、应用程序设计	26
8.1 功能模块	26
8.1.1 摊主身份	27
8.1.2 消费者身份	27
8.1.3 管理者身份	28
8.2 界面设计和对应事务设计	29
8.2.1 登录界面	29
8.2.2 注册页面	31
8.2.3 管理者身份主页面	34
8.2.4 管理者身份“摊铺详情”页面	35
8.2.5 管理者身份“我的市场”页面	36
8.2.6 管理者身份“创建市场”页面	38
8.2.7 摊主身份主页面	42
8.2.8 摊主身份“摆摊地点详情”页面	45
8.2.9 摊主身份“我的摊铺”页面	46



“摊易选”地摊管理系统

8.2.10 摊主身份“添加摊铺”页面.....	54
8.2.11 消费者身份主页面.....	57
8.2.12 消费者身份“摊铺详情”页面.....	58
九、动态反馈.....	59
9.1 登录页面.....	59
9.1.1 用户密码错误.....	59
9.1.2 用户名不存在.....	59
9.1.3 用户名为空.....	60
9.1.4 密码为空.....	60
9.2 注册页面.....	61
9.2.1 必填信息遗漏.....	61
9.2.2 用户名重复.....	61
9.2.3 格式不正确.....	62
9.2.4 确认密码与原密码不相符.....	62
9.3 摊主创建摊铺页面.....	62
9.3.1 必填信息缺失.....	62
9.4 管理者创建市场页面.....	64
9.4.1 必填信息缺失.....	64
十、总结.....	65
10.1 系统优点.....	65
10.2 系统不足.....	65
10.3 系统改进.....	65
10.4 经验与收获.....	65
附. 参考文献.....	66



前言

近年来，“摆地摊”、“摊主与城管”一直是社会中热度很高的话题，城管暴力执法、地摊阻碍交通等也一直在坊间被热议。随着国民生活水平、生活质量的迅速提高，占道经营、以“脏乱差”为代名词的地摊在繁华整洁的大城市似乎已经销声匿迹，并且随着越来越多的城市进行创建文明城市等治理行动，影响市容市貌的地摊经济一直生在夹缝中。

但是就在今年，一场突如其来的疫情让整个社会为之停摆。为了防范疫情的扩散，我们国家大部分地区都实行了数月的封闭管理政策，国民经济也为此受到影响。经济的不景气带来了失业率的增加，而地摊经济的优点这时就显得尤为突出。

地摊经济有其天然的三低特质：创业门槛低，没有店铺租金的压力，此外摆地摊没有太高的学历、技能要求，几乎人人都可以摆地摊；失败风险低，由于摆地摊没有店面租金等经济支出，船小好调头，从业者即使失利也能够迅速“满血复活”；商品价格低，地摊由于没有店铺租金等支出，因而商品价格也会较低，能够让居民有更多的购物选择，享受到更多实惠。

正是由于这三低特性，很多的暂时失业在家的民众开始通过摆地摊获取收入，摆地摊成为了保民生、保就业的重要举措，“地摊经济”冲出江湖，并获得政府部门的大力推广。今年六月份在烟台调研的李克强总理说：“地摊经济、小店经济是就业岗位的重要来源，是人间的烟火，和‘高大上’一样，是中国的生机”。此后各地政府放松监管，大力提倡地摊经济、夜市经济。

但是我们也要知道，地摊经济存在的监管难、占道经营影响交通的问题依然多多少少的存在，这一问题主要是因为摊贩不想商铺，他们位置不固定，而监管部门也没有一



“摊易选”地摊管理系统

个渠道获取摊贩们的实时位置，并且也没有渠道为摊贩们登记造册，造成摊贩们成为了“黑户”；此外，对于摊主来说，他们存在的难题是摆摊范围很有限，一般就集中在社区门口、交通主干道路口、固定的夜市等，这一方面因为摊主没有一个渠道获取哪里可以摆摊、哪里不能摆摊、哪里摆摊人流量大等信息，另一方面是因为顾客大多都集中于社区附近、交通主干道附近，因此摊主都希望在这里摆摊，而这又与顾客没有渠道获取想买的东西在哪里的地摊上有、哪个位置摊贩多等信息有关。一句话来说，摊贩没有渠道上传自己的经营信息、顾客没有渠道获取摊贩的经营信息两者紧密结合，只要解决了渠道问题，那么这两个问题就迎刃而解。并且，当我们有一个渠道上传摊贩信息后，监管难的问题也将得到很大程度的解决。

而我想开发的“摊易选”平台就希望打造这样一个渠道，摊主们上传自己的经营范围、实时所处地理位置后，使用该平台的顾客就能够发现这个摊主，这不仅解决了顾客寻摊难的问题，还增加了摊主们地摊的曝光率，能够给摊主带来更多顾客；同样的根据摊主上传的信息，我们能够精确定位每一个摊主，能够解决摊主遇到问题就跑的管理难题，此外通过统计某一区域摊主的数量可以实时监控拥堵情况，从而帮助管理人员解决交通拥堵的问题。综上，通过“摊易选”平台我们解决了摊主、顾客、管理人员存在的诸多问题，可谓是一举三得！

因为该系统能够解决多个痛点问题，这会让摊主主动上传自身信息，从而不需要采取政府主导的强制措施，当摊主们大多愿意上传自身信息时，“摊易选”平台对于三类人群的功能就能充分发挥出来。



一、项目简介

1.1 项目目标用户

“摊易选”地摊管理系统主要面向三类人群，摊主、顾客和管理人员，着力解决三类人群存在的三类重点问题，分别是摊主们存在的寻找摆摊地点难、缺乏宣传渠道的问题；顾客们存在的寻找心仪地摊难、缺乏交流、评价等信息渠道的问题；管理者们存在的监管摊主难的问题。

1.2 项目主要功能简介：

1.2.1 对于摊主：

“摊易选”地摊管理系统主要为摊主提供智能寻找摆摊位置、帮助宣传推荐的服务，系统内会上传每个地区可供摆摊地区的信息和最大荷载量（最多允许的摊铺数目），摊主们使用该平台可以直观看到附近地区可供摆摊的位置和摊铺剩余情况，系统也会为摊主推荐距离摊主位置较近、适合摊主所售卖物品的摆摊位置，从而帮助摊主进行摆摊地点的选择。

当摊主选择好位置后并开始摆摊后，摊主的位置信息和经营信息就会上传系统后台，从而可以该摊主推荐给附近的顾客和与潜在的对摊主所经营物品感兴趣的顾客，从而增加摊主的曝光度、实现摊主与顾客的对接。

此外，系统会开放摊主对摆摊地点的评价功能，摊主们通过查看其他摊主对某一摆摊地点的评价信息，从而选择是否在该地区摆地摊。同样，摊主也可以对自己摆地摊的地方上传自己的点评，反馈自己的意见或感受。实现某种程度上的摊主之间的交流。

1.2.2 对于顾客：

“摊易选”地摊管理系统为顾客提供的推荐具体地摊、商店的功能，当顾客登录该系统时，可以查看附近地区的所有地摊，而且系统也会为顾客推荐附近的较优质地摊。此外，顾客也可以自行搜索想要购买的物品或服务，系统会根据顾客的搜索推荐符合条件的地摊，从而让顾客在逛地摊时精准购物成为现实，解决此前逛地摊“撞大运”式的购物形式。

系统同样会开放摊贩对摊主的评价功能，顾客通过查看其他顾客对摊主评价信息，从而决定是否要在该地摊进行消费，顾客也可以上传自己的评价感受，实现交流、评价、反馈的良好生态。



1.2.3 对于管理者：

“摊易选”地摊管理系统同样服务于管理人员，例如城管等，这可以有效缓解地摊经济监管难的问题。当管理人员登录系统后，系统会推荐当前附近位置的摊主供管理人员管理；此外，如果某位摊主的顾客评价等级信息较低，那么该位摊主也会被优先推荐给管理人员监管。由于系统将摊主的信息记录在后台，当某位摊主在经营期间发生了诸如食品安全问题等，系统可以将此前摊主留在系统内的信息提供给管理人员，确定摊主从而进行相关行为的治理、处罚。凭借此功能，解决了此前地摊经济违法违规成本低、难以处罚的难题。

1.3 项目开发上线形式：

作为数据库课程设计的项目，结合自身实际和技术要求，“摊易选”地摊管理系统将会先开发网页版本，但由于此类应用在移动端使用更加方便快捷，不排除会开发移动端应用，例如 app 或微信小程序等形式。



二、系统开发平台

2.1 开发平台简介

“摊易选”地摊管理系统采用 C-S 架构，使用 Python 中 Django 应用框架的技术路线，采用 MTV 框架模式。

Model 即模型层使用数据库作为持久性储存结构，后台数据库选用 MySQL8.0。这是业界领先的开源数据库，在开源产品中具有仅次于 Apache 服务器的市场占有率。本数据库开放源代码，具有免费使用，比较稳定的特点，适合于小型系统的持久性存储。

View 即视图层是 Django 处理请求的核心代码层，我们大多数的 python 代码均集中于此。视图层对外接受用户请求，对内调度模型层和模板层，统合数据库和前端，最后根据业务逻辑，将处理好的数据与前端结合，返回给用户。视图层是真正的后端。

Template 即模板层，这用于动态渲染 HTML 页面，其包含所需 HTML 页面的静态部分以及一些特殊的模板语法，用于将动态内容插入其中。

2.2 开发语言：Python

用 python 做后端设计开发流程清晰、结构合理，拓展性良好，Python 在数据处理、机器学习等方面表现突出，可按需要进行简单快捷的耦合。

2.3 开发工具：PyCharm 2018.3.3 python 3.6 Django 3.0.4

2.4 数据库：MySQL 8.0

MySQL 是业界领先的开源数据库，在开源产品中具有仅次于 Apache 服务器的市场占有率。本数据库开放源代码，具有免费使用，比较稳定的特点，适合于小型系统的持久性存储。

2.5 操作系统:Microsoft Windows 10



三、数据库规划

3.1 任务陈述

“摊易选”地摊管理系统是地摊经济下解决现有问题的实用性平台，针对地摊经济中的三类人群：摊主、顾客、城市管理人员，“摊易选”系统都能解决其存在的特有的痛点难题，此外在当今经济下行的大环境背景下，地摊经济势必要发挥其保就业、保民生的作用，此前的粗放式、缺乏监管的发展模式是不适应长期发展要求的，面对庞大的地摊信息要有地摊管理系统来规范化地摊经济的发展。

“摊易选”平台针对管理者有摆摊地点推荐、摆摊地点搜索、摆摊地点评价、摊铺信息上传与修改、摊铺信息展示推送等功能；针对购物者有地摊推荐、购买物品（地摊）搜索、地摊（商品）评价等功能；针对城市管理者有摆摊地点区域信息上传与维护、摊主信息审核、摊主信息查看、摊主优先监管推荐、地摊（摊主）评价等功能。与摊主身份和购物者身份相比，城市管理者身份类似于管理员身份模式，拥有最多的权限和功能。城市管理者身份拥有对摊主的审核权、监管权，当摊主产生安全问题时，管理者身份拥有对摊主的否决权等。



“摊易选”地摊管理系统

3.2 任务目标

模块	功能名称	具体描述
登录系统及注册	登录身份管理	用户根据自身需求选择城市管理者、消费者、摊主三种身份之一进行登录。
	信息注册	用户首次登录时需要先注册信息，以摊主身份登录时需要上传个人真实信息，并需要上传摊铺经营信息。以城市管理者身份登录需要上传自身身份证明信息。而对于消费者只需要设置账号密码等基本个人信息即可。
信息推荐	推荐摆摊地点	以摊主身份登录时，系统推荐附近或优质摆摊区域供摊主选择。
	推荐摊铺	以消费者身份登录时，系统推荐附近或优质摊铺供消费者选择。
	推荐监管摊铺	以城市管理者身份登录时，系统推荐附近或优先监管摊铺，优先监管商铺即顾客评分较低或曾出现安全问题的摊铺。
分类搜索	摆摊地点及摆摊物品搜索	以摊主身份登录时，可以搜索自己想摆摊的地点或商品服务，系统会返回符合条件的摆摊地点。
	摊铺及购买物品搜索	以消费者身份登录时，可以搜索自己想消费的摊铺或购买的物品，系统会返回符合条件的摊铺。
	监管摊铺及商品搜索	以城市管理者身份登录，可以搜索要监管的摊铺或者商品，系统会返回符合条件的优先监管摊铺。
	摊铺分类栏	以消费者或城市管理者身份登录系统时，分类栏会按照美食、服装等将类别分别推荐相应的摊铺。推荐的摊铺供消费者选择、供城市管理者监管。
	摆摊地点分类栏	以摊主身份登录系统时，分类栏会按照美食、服装等将类别分别推荐适合对该类商品摆摊的摆摊地点，供摊主选择合适的摆摊地点。
选择摊铺或摆摊地点	摆摊地点信息	显示选择的摆摊地点的基本信息，包括区域位置、摊铺数量限制、摆摊时间限制等。
	摊铺信息	显示选择的摊铺的基本信息。包括位置信息、经营简介、消费者评价等。对于城市管理者身份，还会展示其他城市管理者的检查反馈信息。
评价	摆摊地点评价	以摊主身份登录系统时，在摆摊结束后可以针对该摆摊地点发表自己的评价信息。
	摊铺评价	以消费者身份登录系统时，在购物结束后可以针对该摊铺发表自己的评价信息。
	摊铺检查反馈	以城市管理者身份登录系统时，在对摊铺检查结束后，可以针对该摊铺发表本次检查的反馈信息。



四、系统定义

4.1 系统边界

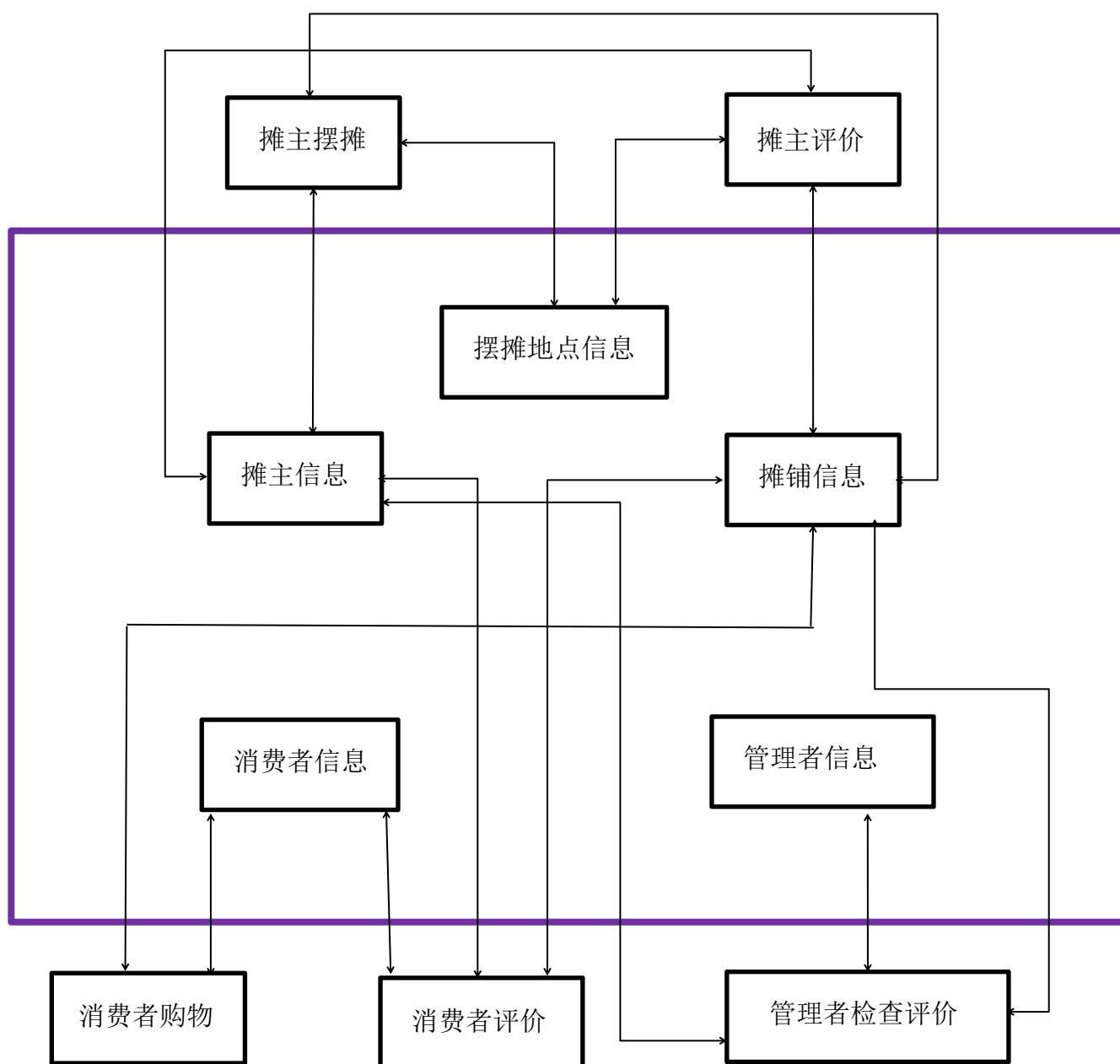
系统边界描述数据库系统和企业信息系统的其他部分的接口，是信息系统内部构成元素与外部有联系实体之间的信息关系的描述与分割。它并不需要在它们之间划一条物理边界，而只需要弄清它们之间信息输入与输出的分割。

本数据库系统共包括摆摊地点信息、摊主信息、摆摊地点信息、消费者信息、城市管理者信息等几个模块，该信息系统的其他部分包括摊主选择摆摊地点摆摊、摊主评价摆摊地点、消费者选择摊铺购物、消费者对摊铺评价、城市管理者对摊铺即摊主进行检查评价等。

数据库系统的内部构成元素与外部其他部分之间的信息关系如下图所示：



“摊易选”地摊管理系统





4.2 用户视图

城市管理者用户视图

(1) 摆摊地点信息管理：

上传、查看、修改、维护摆摊地点信息，包括允许摆摊的地理位置、最大允许摆摊数量、允许经营时间、允许经营类别等。

(2) 监管摆摊地点：

对摆摊地点（区域），监控此处的摊铺数量等实时信息，对于摊主数量较多的摆摊地点适当增加监管力度或动态调整摊铺数量。

(3) 摊主信息管理：

对摊主上传到系统中的个人实名身份信息、经营信息进行审核并决定是否通过，当摊铺发生经营问题时及时通过信息确定摊主，从而协助处理问题。

(4) 监管摊铺并进行评价反馈：

针对系统推荐的优先监管摊铺，要及时进行检查与反馈，同时对附近地区的摊铺也要进行日常的监管。监管完成要在系统内对摊铺进行及时准确的评价，给顾客的消费行为提供指引。

摊主用户视图

(1) 摊铺和个人信息上传与管理：

摊主在使用系统时，需要首先上传自身的真实身份信息和摊铺经营信息，得到管理员审核通过后才能获得进一步功能的权限。此外还可以对个人信息和摊铺经营信息进行日常修改维护。

(2) 摆摊地点推荐：

系统根据摊主上传的个人经营信息和位置信息，推荐附近且符合要求的摆摊地点供摊主选择。

(3) 摆摊地点搜索：

摊主可以根据自身需要通过搜索得到潜在的摆摊地点，例如摊主可以根据理想摆摊地点的位置、经营类别进行搜索，系统会返回符合条件的搜索结果供摊主进行选择。



(4) 摆摊开始与结束:

当摊主选择好摆摊地点并开始摆摊后,通过点击开始摆摊按钮可将自身经营信息、地理位置信息上传后台数据库,系统就可以将该地摊推荐给相应的潜在消费者。当摊主结束摆摊时,同样可以点击相应按钮,系统会终止对该摊铺的推荐和展示。

(5) 摆摊地点评价:

摊主可以针对自己所在的摆摊地点进行相应的评价与反馈。

消费者用户视图

(1) 摊铺推荐:

系统根据购物者的位置信息等推荐附近的、购物者可能感兴趣的摊铺供购物者进行选择。

(2) 摊铺、商品搜索:

购物者可以搜索自己想要购买的商品或者摊铺名称,系统会将符合条件的摊铺推荐给购物者。

(3) 摊铺评价:

购物者在消费后可以针对摊铺进行自己的评价与反馈。

五、需求分析

5.1 用户需求说明

5.1.1 数据需求

其中需求数据为:

1、摊主基本信息记录:

摊主编号、摊主姓名、摊主身份证号、摊主手机号、摊主性别、摊主头像、登录密码

2、摊铺基本信息记录:



“摊易选”地摊管理系统

摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经营状态

3、消费者基本信息记录：

消费者编号、消费者手机号、消费者用户名、消费者头像、登录密码

4、城市管理者基本信息记录：

管理者编号、管理者身份、管理者姓名、管理者身份证号、管理者手机号、管理者头像、登录密码

5、摆摊地点基本信息记录：

摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点图片、摆摊地点经营状态

5.1.2 事务需求

1、数据录入：

(1) 摊主注册：

摊主手机号、登录密码

(2) 录入摊主的基本信息：

摊主编号、摊主姓名、摊主身份证号、摊主性别、摊主头像

(3) 消费者注册：

消费者手机号、消费者用户名、登录密码

(4) 录入消费者信息：

消费者编号、消费者头像

(5) 城市管理者注册：

管理者手机号、登录密码

(6) 录入城市管理者信息：

管理者编号、管理者身份、管理者姓名、管理者身份证号、管理者头像

(7) 录入摊铺基本信息：

摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经



营状态

(8) 录入摆摊地点信息：

摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点图片、摆摊地点经营状态

2、实体联系：

(1) 摊主选择摆摊地点摆摊：

摊主编号、摊铺编号、摆摊地点编号、实时位置信息

(2) 摊主评价摆摊地点：

摊主编号、摊铺编号、摆摊地点编号、评价内容（图片、文字）

(3) 消费者评价摊铺：

消费者编号、摊铺编号、评分、评价内容（图片、文字）

(4) 城市管理者检查反馈摊铺：

管理者编号、摊铺编号、检查评分、检查反馈内容（图片、文字）

3、数据更新/删除：

(1) 摊主信息的更新

(2) 消费者信息的更新

(3) 城市管理者信息的更新

(4) 摆摊地点信息的更新/ 删除

(5) 摊铺信息的更新/ 删除

4、数据查看：

城市管理者用户视图下：

(1) 按照摊主编号列出摊主的摊主编号、摊主姓名、摊主身份证号、摊主手机号、摊主性别

(2) 按照摆摊地点编号列出摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点经营状态

(3) 按照摆摊地点评分列出摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点经营状态

(4) 按照摊铺评分列出摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、



摊铺图片、摊铺经营状态

- (5) 按照检查评分列出摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经营状态

城市管理者领导者用户视图下：

- (1) 按照管理者编号列出管理者编号、管理者身份、管理者姓名、管理者身份证号、管理者手机号、管理者头像

摊主用户视图下：

- (1) 按照距离列出摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点经营状态
- (2) 按照评分列出摆摊地点编号、摆摊地点名称、摆摊地点位置范围、最多允许摊铺数量、适合经营类别、摆摊地点经营状态

消费者用户视图下：

- (1) 按照距离列出摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经营状态
- (2) 按照评分列出摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经营状态
- (3) 按照检查评分列出摊主编号、摊铺编号、摊铺名称、摊铺经营类别、摊铺简介、摊铺图片、摊铺经营状态

5.2 系统需求说明

“摊易选”地摊管理平台需要有较强的数据处理功能，如果现在某一地区推行，在理论上应该容纳上万人的数据资料，并且在搜索方面要有较快的响应速度，并能够处理多方面组合搜索数据请求。平台能够有效的处理各种异常，应具有良好的健壮性。

5.2.1 初始数据库大小（试验阶段）

- (1) 大约有数百个摆摊地点，数十个城市管理者、上千个摊主及相应的摊铺，上万个消费者。

5.2.2 网络和共享需求

- (1) 必须能够支持至少上百名用户同时访问，需要考虑这么大数量并发访问的许可需



求。

5.2.3 性能

高峰期：每天的下午、晚上

- (1) 单个记录查询时间少于 1 秒，高峰期少于 5 秒
- (2) 多个记录查询时间少于 5 秒，高峰期少于 10 秒
- (3) 更新/保存记录时间少于 1 秒，高峰期少于 5 秒

5.2.4 安全性

- (1) 数据库必须有口令保护
- (2) 每个用户分配特定的用户视图所应有的访问权限
- (3) 用户只能在适合他们完成工作需要的窗口中看到需要的数据

5.2.5 备份和恢复

每天 24 点备份

5.2.6 用户界面

菜单驱动，联机帮助

5.2.7 法律问题

对用户信息管理，遵守法律

六、数据库逻辑设计

6.1 简述：

“摊易选”地摊管理平台主要服务对象和主要功能设计在前面已经提过，在数据库中我们应该建立消费者、摊主、管理者、摆摊地点、摊位、评论六个实体集和摊位属于摊主、摊位在摆摊地点营业、摊主评价摆摊地点、消费者评价摊位、管理者评价摊位五个联系集。

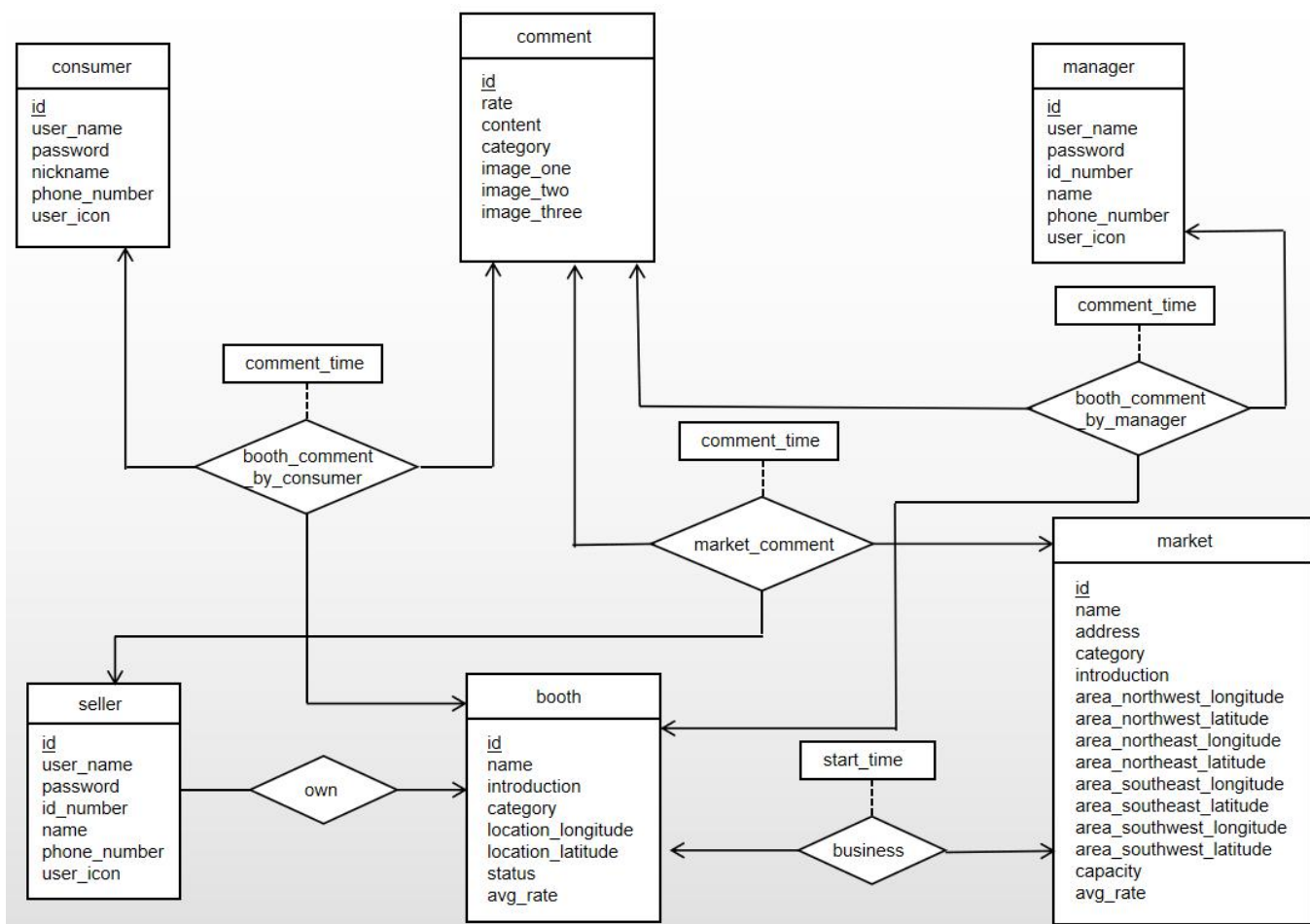
由于摊位属于摊主这个关系是一个一对多的关系，即一个摊主可以拥有多个摊位，此外没有其他描述性属性，因此我们可以不单独建立联系集，而直接在摊位实体集 booth 中增加一个属性 seller_id，将其作为外键，主表是实体集 seller，这样就可以表示某个摊位属于某个摊主，另外建立了外键约束，当主表 seller 发生变化时，从表 booth



“摊易选” 地摊管理系统

的数据会发生相应的变化，例如当 seller 实体集某个记录被删除，那么属于该摊主的摊位肯定不存在，此时由于外键约束，从表中属于被删除摊主的摊位会自动被删除，这就保证了数据的正确性。

6.2 ER 图：



该 ER 图包括消费者 consumer、摊主 seller、管理者 manager、摆摊地点 market、摊位 booth、评论 comment 六个实体和摊位属于摊主 own、摊位在摆摊地点营业 business、摊主评价摆摊地点 market_comment、消费者评价摊位 booth_comment_by_consumer、管理者评价摊位 booth_comment_by_manager 六个联系集。

此外，在后续的进展中，再次添加存储所有静态图片地址的实体集 pic_urls，该实体集将存储整个项目所有的静态图片地址，因此该实体集应与其他各实体集例如 market、booth 等有联系，我们不单独设置联系集，而在 pic_urls 实体集中添加属性具体的实体集属性 category 与 id。具体见 6.5。

在项目中我们需要使用 session 机制来暂时存储信息，信息同样需要存储到数据库中。Django 框架自动生成数据表 django_session。



6.3 数据字典：

6.3.1 从数据字典中提取出来的实体描述：

消费者实体表 consumer：

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int (11)		10	NO	PRIMARY	FALSE
user_name	用户名	varchar (64)	64		NO	user_name	FALSE
password	密码	varchar (128)	128		NO		
nickname	昵称	varchar (64)	64		NO		
phone_number	电话	varchar (32)	32		YES		
user_icon	头像	varchar (64)	64		NO		

摊主实体表 seller：

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int (11)		10	NO	PRIMARY	FALSE
user_name	用户名	varchar (64)	64		NO	user_name	FALSE
password	密码	varchar (64)	64		NO		
id_number	身份证号	varchar (32)	32		NO	id_number	FALSE
name	真实姓名	varchar (128)	128		NO		
phone_number	电话号码	varchar (32)	32		YES		
user_icon	头像	varchar (64)	64		NO		

管理者实体表 manager：

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int (11)		10	NO	PRIMARY	FALSE
user_name	用户名	varchar (64)	64		NO	user_name	FALSE
password	密码	varchar (64)	64		NO		
id_number	身份证号	varchar (32)	32		NO	id_number	FALSE
name	真实姓名	varchar (128)	128		NO		
phone_number	电话号码	varchar (32)	32		YES		
user_icon	头像	varchar (64)	64		NO		



“摊易选” 地摊管理系统

摊位实体表 booth:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int(11)		10	NO	PRIMARY	FALSE
seller_id	外键	int(11)		10	NO	外码seller(id)	TRUE
name	名称	varchar(128)	128		NO		
introduction	简介	varchar(256)	256		YES		
category	分类	varchar(32)	32		NO		
location_longitude	摆摊时位置经度	float		12	YES		
location_latitude	摆摊时位置纬度	float		12	YES		
status	是否营业	int(11)		10	NO		
avg_rate	平均评分 (计算列)	float		12	YES		

摆摊地点实体表 market:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int(11)		10	NO	PRIMARY	FALSE
name	名称	varchar(128)	128		NO		
address	地址	varchar(128)	128		NO		
introduction	简介	varchar(256)	256		YES		
area_northwest_longitude	西北经度	float		12	NO		
area_northwest_latitude	西北纬度	float		12	NO		
area_northeast_longitude	东北经度	float		12	NO		
area_northeast_latitude	东北纬度	float		12	NO		
area_southeast_longitude	东南经度	float		12	NO		
area_southeast_latitude	东南纬度	float		12	NO		
area_southwest_longitude	西南经度	float		12	NO		
area_southwest_latitude	西南纬度	float		12	NO		
capacity	容纳摊位数量	int(11)		10	NO		
avg_rate	平均评分 (计算列)	float		12	YES		

评论实体表 comment:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int(11)		10	NO	PRIMARY	FALSE
rate	评分	float		12	NO		
content	评价内容	varchar(512)	512		NO		
image_one	评价图片链接	varchar(64)	64		YES		
image_two	评价图片链接	varchar(64)	64		YES		
image_three	评价图片链接	varchar(64)	64		YES		



“摊易选”地摊管理系统

6.3.2 从数据字典中提取出来的联系描述:

摊位在摆摊地点营业联系集 business:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
booth_id	摊位 id	int(11)	10	0	NO	PRIMARY	FALSE
market_id	市场 id	int(11)	10	0	NO	PRIMARY	FALSE
start_time	开始营业时间	datetime			NO		

摊主评价摆摊地点联系集 market_comment:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
seller_id	摊主 id	int(11)	10	0	NO	PRIMARY	FALSE
market_id	市场 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_id	评价 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_time	评价时间	datetime			NO		

消费者评价摊位联系集 booth_comment_by_consumer:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
consumer_id	消费者 id	int(11)	10	0	NO	PRIMARY	FALSE
booth_id	摊位 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_id	评价 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_time	评价时间	datetime			NO		

管理者评价摊位联系集 booth_comment_by_manager:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
manager_id	管理者 id	int(11)	10	0	NO	PRIMARY	FALSE
booth_id	摊位 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_id	评价 id	int(11)	10	0	NO	PRIMARY	FALSE
comment_time	评价时间	datetime			NO		

四种联系集中, manager_id、booth_id、consumer_id、seller_id、market_id、comment_id 在各自的联系集中既是联合主码, 又是外键, 这样当主表数据发生改变时, 从表会自动改变数据, 从而保证数据正确性。

6.4 数据库逻辑简要描述:

ER 图中存在五个联系集, 但是数据库中我们将联系集 own (摊位属于摊主) 转化为在摊位实体集 booth 中添加一个参考实体集 seller 的外键 seller_id, 主要原因是该联系集 own 没有描述性属性, 因而选择不建立实体集。但是对于另外四个联系集, 我们都根据关联的实体集的主码和描述性属性建立联系集, 其中关联实体集的主码作为联系集的联合主码, 与此同时, 每个属性都作为参考关联实体集的外键, 这样当关联实体集



“摊易选”地摊管理系统

数据发生变化时,联系集也会自动做出必要的改变,从而保证数据完整性。例如对于摊主评价摆摊地点的联系集 `market_comment`,如果摊主被注销,那么该摊主的评价也需要删除,当我们在联系集 `market_comment` 建立相关的外码依赖时,联系集中与被删除摊主相关的记录就会被删除,那么我们就无法根据该联系集查找到属于该被删除摊主的评价,从而达到自动删除评价的目的。

登录注册时,我们需要根据身份操作消费者 `consumer`、摊主 `seller`、管理者 `manager` 三个数据表之一。

对于摊主,还可以建立、修改或删除属于自己的摊位,这会操作数据表 `booth`;此外,摊主可以选择摊位和摆摊地点进行摆摊,这会操作联系集 `business`;并且同时更新 `booth` 表中的是否营业标志 `status` 和经纬度属性 `location-longitude` 和 `location-latitude`。当摊主对摆摊地点评价时,会对数据表 `market_comment` 进行操作。

对于管理者,可以建立市场,自己建立的市场只有自己可以修改信息或者删除。在建立、修改、删除市场时都会操作数据表 `market`。此外,管理者可以选择摊铺进行监管、查看摊铺基本信息,这时会操作数据表 `booth`,这里的操作主要是 `select` 操作。此外管理者可以对摊铺进行评价,此时会操作数据表 `booth_comment_by_manager`。

对于消费者,可以选择摊铺进行查看摊铺信息等操作,这时会操作数据表 `booth`,当消费者对摊位进行评价时,会分别操作数据表 `booth_comment_by_consumer`。

6.5 数据库更新:

在开发过程中,发现了数据库此前设计上出现的一些问题。

(1) 首先是我们的项目需要存储一些图片,包括摊主上传的摊铺图片、管理人员上传的摆摊地点图片以及评论中的图片,我们可以将这些图片都存储在项目的特定静态文件夹中,而将图片的相对路径存储在数据库中。我们设置数据表 `pic_urls`,其中包括 `id`、`category` 和 `url` 三个字段,具体类型定义如下:

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
<code>id</code>	所属类别 <code>id</code>	<code>int(11)</code>	10	0	NO	PRIMARY	FALSE
<code>category</code>	图片所属类别	<code>varchar(64)</code>	64	0	NO	PRIMARY	FALSE
<code>url</code>	图片路径	<code>varchar(128)</code>	128	0	NO	PRIMARY	FALSE

其中 `category` 表示该图片所属类别,例如如果是摊铺图片,那么 `category` 应为“`booth`”,如果是摆摊地点图片,那么 `category` 应为“`market`”;`id` 表示该图片所属的主体在主体表中的 `id` 值,例如如果某图片属于 `booth` 表中 `id` 为 5 的摊铺所有,那么在 `pic_urls` 表中,`category` 应记录为“`booth`”,`id` 应记录为 5。这样通过 `category` 和 `id`,我们就将确定该图片的“所有者”。`url` 记录的是图片的相对路径。

此外,一个“所有者”可以拥有多张图片,所以 `id`、`category`、`url` 三者联合作为数据表的主键。

(2) 在此前的摆摊地点表 `market` 中,我们存储了平面中四个地点的经纬度,我们可以改变为只存储西北角的经纬度,在存储东西方向的距离 `east_dis` 和南北方向的距离



“摊易选”地摊管理系统

south_dis。此外，我们只存储了表示摆摊地点摊位容量的 capacity，我们再次添加另一个字段 current_capacity，表示此时摆摊地点的摊位数量，这样 capacity - current_capacity 的结果就是摆摊地点的摊位剩余量，我们在向摊主展示边摊地点是只展示摊位剩余量大于 0 的摆摊地点即可。最后，我们要有数据表示摆摊地点的允许经营范围和联系方式，再增加两个字段 category 和 phone_number 即可。

此外，在实现过程中，我们需要对摆摊地点 market 进行创建、修改或删除，那么执行这些行为的主体一定管理员身份 manager，那么我们的设计是管理员能管理所有摆摊地点还是管理员只能管理部分摆摊地点呢？显然后者更合理，我们可以仿照摊主 seller 对摊铺 booth 的行为，设置管理者只能管理其创建的摆摊地点，例如编辑该摆摊地点或删除该摆摊地点。那么我们需要为摆摊地点 market 和管理者 manager 建立联系，我们可以使用联系集，也可以在 market 表中添加字段 manager_id，我们选用后者。

market 表的具体类型定义如下：

列名	注释	数据类型	字符长度	数据精度	能否为空	键名称	能否重复
id	唯一标识	int(11)		10	NO	PRIMARY	FALSE
name	名称	varchar(128)	128		NO		
address	地址	varchar(128)	128		NO		
introduction	简介	varchar(256)	256		YES		
area_northwest_longitude	西北经度	float		12	NO		
area_northwest_latitude	西北纬度	float		12	NO		
east_dis	东西距离	float		12	NO		
south_dis	南北距离	float		12	NO		
capacity	容纳摊位数量	int(11)		10	NO		
current_capacity	摊位占有量	int(11)		10	NO		
category	允许经营种类	varchar(64)	64		NO		
phone_number	联系方式	varchar(32)	32		NO		
avg_rate	平均评分 (计算列)	float		12	YES		

七、数据库物理设计

7.1 索引

数据库中的索引：

表名	主键	外键
consumer	id	



“摊易选” 地摊管理系统

seller	id	
manager	id	
booth	id	seller_id
market	id	
comment	id	
pic_urls	id	
business	booth_id, market_id	booth_id, market_id
market_comment	seller_id, market_id, comment_id	seller_id, market_id, comment_id
booth_comment_by_consumer	consumer_id, market_id, comment_id	consumer_id, market_id, comment_id
booth_comment_by_manager	manager_id, market_id, comment_id	manager_id, market_id, comment_id

在表示摊铺的表 **booth** 中，我们设置了外键 **seller_id**，即表示摊铺被摊主所有，当摊主的信息注销时，由于外键约束，**booth** 表中对应的记录也会被自动删除。

在记录图片路径即其对应主体的表 **pic_urls** 中，我们没有建立外键约束，因为不同身份的主体，其 **id** 有可能相同。当我们需要查找某主体的图片时，我们需要根据该主体所属的类别（**booth**、**market**）**category** 和在其类别表（**booth**、**market**）中的 **id** 来找到图片的路径，从而将图片显示。在上传图片时也按此上传图片的信息。

由于没有设置外键，当我们删除某条摊铺、摆摊地点的信息时，我们应该将其所有的图片在静态文件夹和数据库中同时删除，因此首先通过查找 **pic_urls** 表得到图片的相对路径，将其在静态文件夹中删除，再将 **pic_urls** 表中的记录删除。

market_comment 表示摊主对摆摊地点的评论、**booth_comment_by_consumer** 表示消费者对摊铺的评论、**booth_comment_by_manager** 表示管理者对摊铺的评论，这三个数据表均表示关系，表示评论提出者、评论接受者、评论三者之间的对应关系，并建立外键约束，当三者有一方的记录被删除时，对应的联系也会删除。例如当摊主信息被注销时，那么在 **market_comment** 表中关于该摊主的记录会被自动删除，但是在此之前我们需要根据 **market_comment** 找到该摊主的所有评论，到 **comment** 表中将具体的评论删除。

7.2 视图

在数据库查询中，有时我们需要多表联合查询，有时我们需要提取表的部分信息，这里为了方便我们可以建立视图，视图是一种虚拟存在的表，但是并不实际存在于数据库中，数据库中只存放了视图的定义，我们在使用视图时根据视图的定义可以动态生成



数据。当我们不再使用视图时，也可以将其删除，这样就会在数据库中删除该视图的定义。

7.3 安全机制

7.3.1 系统安全

摊易选地摊管理系统提供了充足的异常处理机制，在很多的查询、更改等失误中，使用 ajax 技术，服务器端根据可能出现的错误信息返回给浏览器端，浏览器端通过动态的展示报错信息引导用户进行更改等，这样就防止出现浏览器端、服务器端数据不一致、报错信息不明显导致用户体验差的问题。

7.3.2 数据安全

登录时设置密码，确保数据库不被随意更改，保证数据的安全性。

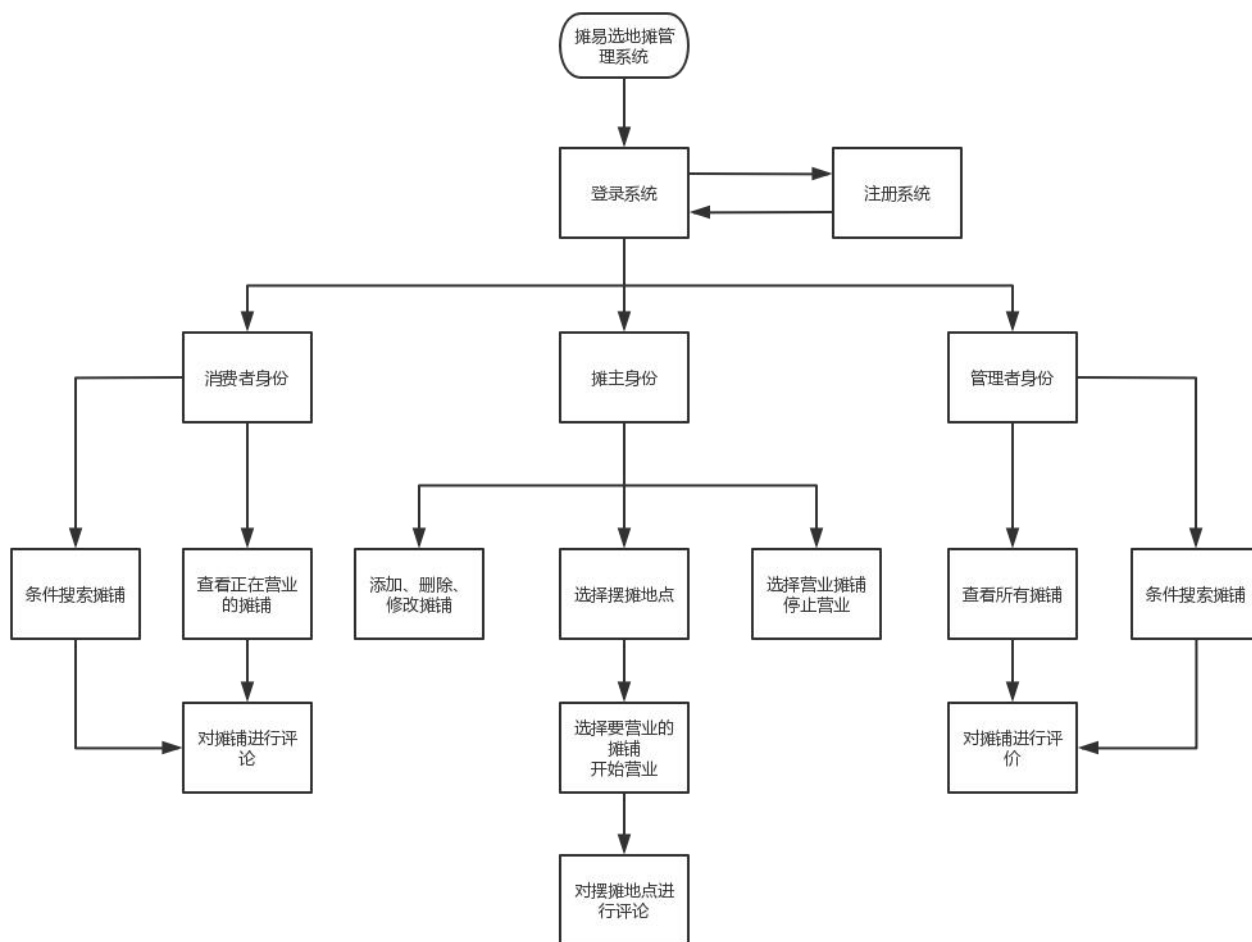
摊易选地摊管理系统提供三种身份入口，分别是摊主、消费者和管理者，三种身份得到的服务不相同，例如只有消费者可以创建摊铺、删除摊铺，并且可以设置摊铺为营业状态或歇业状态；而只有管理者可以创建、删除摆摊地点；对于消费者，其只能查看所有营业状态的摊铺，不能查看摆摊地点的信息。三种身份对应着不同的权限，不同的操作对象，保证了数据的准确性和安全性。

在本系统中，几乎所有更改数据表的操作都需要通过事务实现，并且有联系的表的数据更新，都在服务端设计好逻辑和对应的 sql 语句，避免出现差错，这些都很好的保护了关系型数据库的强一致性，保证了数据安全。



八、应用程序设计

摊易选地摊管理系统的主要功能如下图所示：



8.1 功能模块

摊易选地摊管理系统共有三种身份，即摊主、消费者和管理者。对于摊主身份提供：摊主选择摆摊地点、摊主评价摆摊地点、摊主创建摊铺、摊主删除摊铺、摊主选择摊铺开始营业、摊主选择摊铺停止营业、摊主查看修改个人信息等功能；对于消费者身份提供：消费者选择摊铺、消费者评价摊铺、消费者查看修改个人信息等功能；对于管理者身份提供：管理者选择待监管摊铺、管理者评价摊铺、管理者创建摆摊地点、管理者查看修改摆摊地点、管理者删除摆摊地点等功能。



8.1.1 摊主身份:

- 摊主创建摊铺:

- (1) 输入摊铺名称
- (2) 选择摊铺经营种类（至少选择一个种类）
- (3) 输入摊铺描述
- (4) 为摊铺选择图片并添加
- (5) 检测摊铺名称是否与该摊主已存在摊铺重名，如果重名，需要更改名称
- (6) 完成摊铺的创建

- 摊主删除摊铺:

- (1) 进入我的摊铺页面
- (2) 若要删除摊铺处于营业状态，需先停止营业
- (3) 处于歇业状态的摊铺，可点击删除摊铺按钮
- (4) 完成摊铺的删除

- 摊主选择摆摊地点和摊铺开始营业:

- (1) 选择摆摊类别（美食、服装、果蔬、日用、娱乐），默认为全部选择。
- (2) 可根据需要搜索摆摊地点的名称和地点。
- (3) 可根据需要选择排序依据（距离、评分）
- (4) 选择摆摊地点，进入摆摊地点详情页面
- (5) 点击“我要摆摊”按钮，进入“我的摊铺”页面
- (6) 选择处于歇业状态的摊铺，点击“开始营业”
- (7) 完成选择摆摊地点和摊铺并开始营业的功能

- 摊主评价摆摊地点:

- (1) 进入我的摊铺页面
- (2) 对处于营业状态的摊铺可以点击摆摊地点评价按钮
- (3) 进入摆摊地点评价页面
- (4) 发表具体评价，可上传评价图片
- (5) 完成对摆摊地点的评价

- 摊主查看修改个人信息:

- (1) 进入个人信息页面
- (2) 修改需要修改的个人信息
- (3) 提交后即完成对个人信息的修改

8.1.2 消费者身份:

- 消费者选择摊铺:

- (1) 选择摊铺经营类别（美食、服装、果蔬、日用、娱乐），默认为全部选择。
- (2) 可根据需要搜索摊位的名称和地点
- (3) 可根据需要选择排序依据（距离、评分）



- (4) 选择摊铺，进入摊铺详情页面
- (5) 完成摊铺选择的功能

- 消费者评价摊铺：

- (1) 进入摊铺详情页面
- (2) 点击“我要评价”按钮，进入摊铺评价页面
- (3) 发表具体评价，可上传评价图片
- (4) 完成对摊铺的评价

- 消费者查看修改个人信息：

- (1) 进入个人信息页面
- (2) 修改需要修改的个人信息
- (3) 提交后即完成对个人信息的修改

8.1.3 管理者身份：

- 管理者选择待监管摊铺：

- (1) 选择摊铺经营类别（美食、服装、果蔬、日用、娱乐），默认为全部选择。
- (2) 可根据需要搜索摊位的名称和地点
- (3) 可根据需要选择排序依据（距离、评分）
- (4) 选择摊铺，进入摊铺详情页面
- (5) 完成待监管摊铺选择的功能

- 管理者评价摊铺：

- (1) 进入待监管摊铺详情页面
- (2) 点击“我要评价”按钮，进入摊铺评价页面
- (3) 发表具体评价，可上传评价图片
- (4) 完成对摊铺的评价

- 管理者创建摆摊地点：

- (1) 输入摆摊地点名称
- (2) 选择摆摊地点允许经营类别（至少选择一个）
- (3) 输入摆摊地点地点
- (4) 输入摆摊地点最大摊位容量
- (5) 输入摆摊地点简介
- (6) 输入摆摊地点联系方式、地址
- (7) 检查摆摊地点名称是否与已存在摆摊地点名称相同
- (8) 完成摆摊地点的创建

- 管理者查看、修改摆摊地点：

- (1) 选择摆摊地点经营类别（美食、服装、果蔬、日用、娱乐），默认为全部选择。
- (2) 可根据需要搜索摆摊地点的名称和地点。
- (3) 可根据需要选择排序依据（距离、评分）



- (4) 进入摆摊地点详情页面
- (5) 点击修改信息进入摆摊地点修改页面
- (6) 完成相关信息的修改并提交
- (7) 完成摆摊地点查看、修改功能

• 管理者删除摆摊地点:

- (1) 查看摆摊地点
- (2) 点击摆摊地点进入详情页面
- (3) 点击“删除摆摊地点”按钮
- (4) 再次确认
- (5) 完成对摆摊地点的删除功能

8.2 界面设计和对应事务设计

8.2.1.1 登录界面

用户在登录界面选择登陆身份、输入用户名和密码并经过系统验证后，可以进入各自身份的主页进行相关操作。这里使用 AJAX 技术，在与后台验证后将错误信息实时更新于前端页面，引导用户解决问题。

The image shows a login interface on a light orange background. The interface is a white card with a blue header labeled '登录' (Login). Below the header, there are three radio buttons for user roles: '我是消费者' (I am a consumer), '我是摊主' (I am a stall owner), and '我是管理者' (I am a manager). The '我是消费者' option is selected. Below the radio buttons are two input fields: '请输入账号' (Please enter account) and '请输入密码' (Please enter password). Below the input fields is a checkbox labeled '记住我' (Remember me). Below the checkbox is a blue button labeled '登录' (Login). At the bottom of the card, there is a link '忘记密码?' (Forgot password?) and a link '还没有账户? 立即注册' (No account? Register now).



8.2.1.2 登录事务设计

登录页面逻辑

```
def login(request):
    if request.session.get('is_login', None): # 如果已经登陆过
        identity = request.session.get('identity')
        return redirect('{}/index/'.format(identity))
    if request.method == 'POST': # 如果准备登陆
        # 前端拿来的数据
        identity = request.POST.get('identity')
        user_name = request.POST.get('user_name')
        password = request.POST.get('password')
        response = {'identity': identity} # 记录返回的数据
        # 从数据库中拿到用户 id 和密码
        name = db_login(identity, user_name, password) # 返回登录的昵称
        if name is not None:
            if name[1] is True: # 如果成功登录
                # 在 session 中保存信息
                request.session['is_login'] = True
                request.session['identity'] = identity
                request.session['user_name'] = user_name
                if identity == "consumer":
                    request.session['nickname'] = name[0]
            else:
                request.session['name'] = name[0]
                response['success'] = 'true'
                # return redirect('/index/') # 重定向给 index
        else: # 如果登录不成功, 说明密码错误
            response['success'] = 'false'
            response['user_name_exists'] = 'true'
        else: # 如果昵称为空, 说明不存在该用户名
            response['success'] = 'false'
            response['user_name_exists'] = 'false'

    return JsonResponse(response) # 最后都返回 json 数据

# 如果首次进来, 需要第一次展示登录页面
return render(request, "registerLogin/login.html")
```




```
def db_login(identity, user_name, password):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU', password='tanyixuan1904',
                            database='tanyixuan', charset='utf8', port=3306)

    cursor = conn.cursor()
    # 如果身份是消费者
    if identity == "consumer":
        sql = "select nickname from {table_name} where user_name = '{user_name}';" \
            .format(table_name=identity, user_name=user_name)
        # 如果身份是摊主或者管理者
    else:
        sql = "select name from {table_name} where user_name = '{user_name}';" \
            .format(table_name=identity, user_name=user_name)

    cursor.execute(sql)
    result = cursor.fetchone() # 结果
    if result is not None: # 如果存在这个user_name
        if identity == "consumer":
            sql = "select nickname from {table_name} where user_name = '{user_name}' and password = " \
                "'{password}';".format(table_name=identity, user_name=user_name, password=password)
        else:
            sql = "select name from {table_name} where user_name = '{user_name}' and password = " \
                "'{password}';".format(table_name=identity, user_name=user_name, password=password)
        cursor.execute(sql)
        login_name = cursor.fetchone()
        if login_name is not None: # 如果密码正确, 那么存在名字
            result += (True,)
        else: # 密码不正确, 不存在名字
            result += (False,)

    cursor.close() # 关闭
    conn.close()
    # 不存在该用户, 返回空 都返回result
    return result
```

8.2.2.1 注册界面

用户在注册界面选择摊主、消费者、管理员三种身份之一进行注册, 并且对于三种身份注册所需信息也不同, 摊主身份和管理员身份均需要进行实名注册。注册成功后自动跳转至登录页面进行登录。

消费者身份:



“摊易选”地摊管理系统

欢迎！

您可以选择消费者、摊主、管理者三种身份之一进行注册。

这里是摊易选网站，选择摊主、管理者身份注册时需要进行实名注册，谢谢配合！

f t @ G+ @ v

注册您的账户

请选择您想要注册的身份

☒ 我是消费者 ☐ 我是摊主 ☐ 我是管理者

用户账号

昵称

联系方式

请输入密码

确认密码

点击注册

摊主与管理者身份注册信息相同：

欢迎！

您可以选择消费者、摊主、管理者三种身份之一进行注册。

这里是摊易选网站，选择摊主、管理者身份注册时需要进行实名注册，谢谢配合！

f t @ G+ @ v

注册您的账户

请选择您想要注册的身份

☐ 我是消费者 ☒ 我是摊主 ☐ 我是管理者

用户账号

真实姓名

身份证号

联系方式

请输入密码

确认密码

点击注册

8.2.2.2 注册事务设计

注册页面逻辑

```
def register(request):  
    if request.method == 'POST': # 如果准备注册  
        # 前端拿来的数据  
        request_values = {}  
        request_values['identity'] = request.POST.get('identity')  
        request_values['user_name'] = request.POST.get('user_name')
```



“摊易选” 地摊管理系统

```
request_values['password'] = request.POST.get('password')
request_values['phone_number'] = request.POST.get('phone_number')
request_values['user_icon'] = request.POST.get('user_icon')

# 如果是消费者
if request_values['identity'] == "consumer":
    request_values['nickname'] = request.POST.get('nickname')
else:
    request_values['name'] = request.POST.get("name")
    request_values['id_number'] = request.POST.get("id_number")
# 从数据库中拿到结果, 字典保存
response = db_register(request_values)
return JsonResponse(response)
else:
    return render(request, "registerLogin/register.html")

def db_register(request_values):
    response = {'identity': request_values['identity']} # 用于返回数据
    conn = MySQLdb.connect(host='localhost', user='tanyixuanl', password='tanyixuanl904',
                           database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
    # 身份是消费者
    if request_values['identity'] == "consumer":
        sql = "select id from consumer where user_name = \"{}\";".format(user_name=request_values['user_name'])
        cursor.execute(sql)
        user_id = cursor.fetchone()

        # 如果存在id, 说明数据库中有了相同的用户名 注册失败
        if user_id is not None:
            response['success'] = "false"
            response['user_name_exists'] = "true"
        # 不存在id, 说明可以注册
        else:
            sql = "INSERT INTO consumer (user_name, password, phone_number, nickname, user_icon) VALUES \"(\", \"{}\", \"{}\", \"{}\", \"{}\", \"{}\";\".format(user_name=request_values['user_name'], password=request_values['password'],
                                                                    phone_number=request_values['phone_number'], nickname=request_values['nickname'],
                                                                    user_icon=request_values['user_icon'])
            cursor.execute(sql)
            conn.commit() # 插入的 不要忘记commit
            cursor.close()
            conn.close()
            response['success'] = "true"

# 身份是卖家或者管理者
```



身份是卖家或者管理者

```
else:
```

查看是否有相同的用户名

```
sql = "select id from {table_name} where user_name = \"{user_name}\";"
.format(table_name=request_values['identity'], user_name=request_values['user_name'])
```

```
print(sql)
```

```
cursor.execute(sql)
```

```
user_id = cursor.fetchone()
```

```
print (user_id)
```

存在id, 说明有重复的用户名

```
if user_id is not None:
```

```
response['success'] = "false"
```

```
response['user_name_exists'] = "true"
```

```
else:
```

```
response['success'] = 'true'
```

```
response['user_name_exists'] = 'false'
```

```
# 查看是否有相同的id_number
```

```
sql = "select id from {table_name} where id_number = \'{id_number}\';".format(
    table_name=request_values['identity'], id_number=request_values['id_number'])
```

```
cursor.execute(sql)
```

```
user_id = cursor.fetchone()
```

```
if user_id is not None:
```

```
response['success'] = "false"
```

```
response['id_number_exists'] = "true"
```

```
else:
```

```
response['id_number_exists'] = 'false'
```

```
if response['success'] == "true": # 如果可以注册
```

```
sql = "INSERT INTO {table_name} (user_name, password, name, id_number, phone_number, user_icon) VALUES " \
      "(\\"{user_name}\\", \\"{password}\\", \\"{name}\\", \\"{id_number}\\", \\"{phone_number}\\", \\"{user_icon}\\");"

.format(table_name=request_values['identity'], user_name=request_values['user_name'],
password=request_values['password'], name=request_values['name'], id_number=request_values['id_number'],
phone_number=request_values['phone_number'], user_icon=request_values['user_icon'])
```

```
cursor.execute(sql)
```

```
conn.commit() # 插入的 不要忘记commit
```

```
cursor.close()
```

```
conn.close()
```

```
return response # 最后都要返回结果
```

8.2.3 管理者身份主页面

当我们选择管理者身份登录并输入正确的账号密码之后，我们进入管理者身份主页面。管理者身份主页面主要展示可供监管的摊铺，我们可以选择经营类别来选择特定的摊铺，我们还可以通过选择排序依据来更改摊铺推荐标准，例如按照距离排序和按照评分排序，两种排序方式均可自定义升序或者降序。在上方搜索框我们可以输入想要搜索的摊铺名称或者地址来得到特定的摊铺，支持模糊搜索。

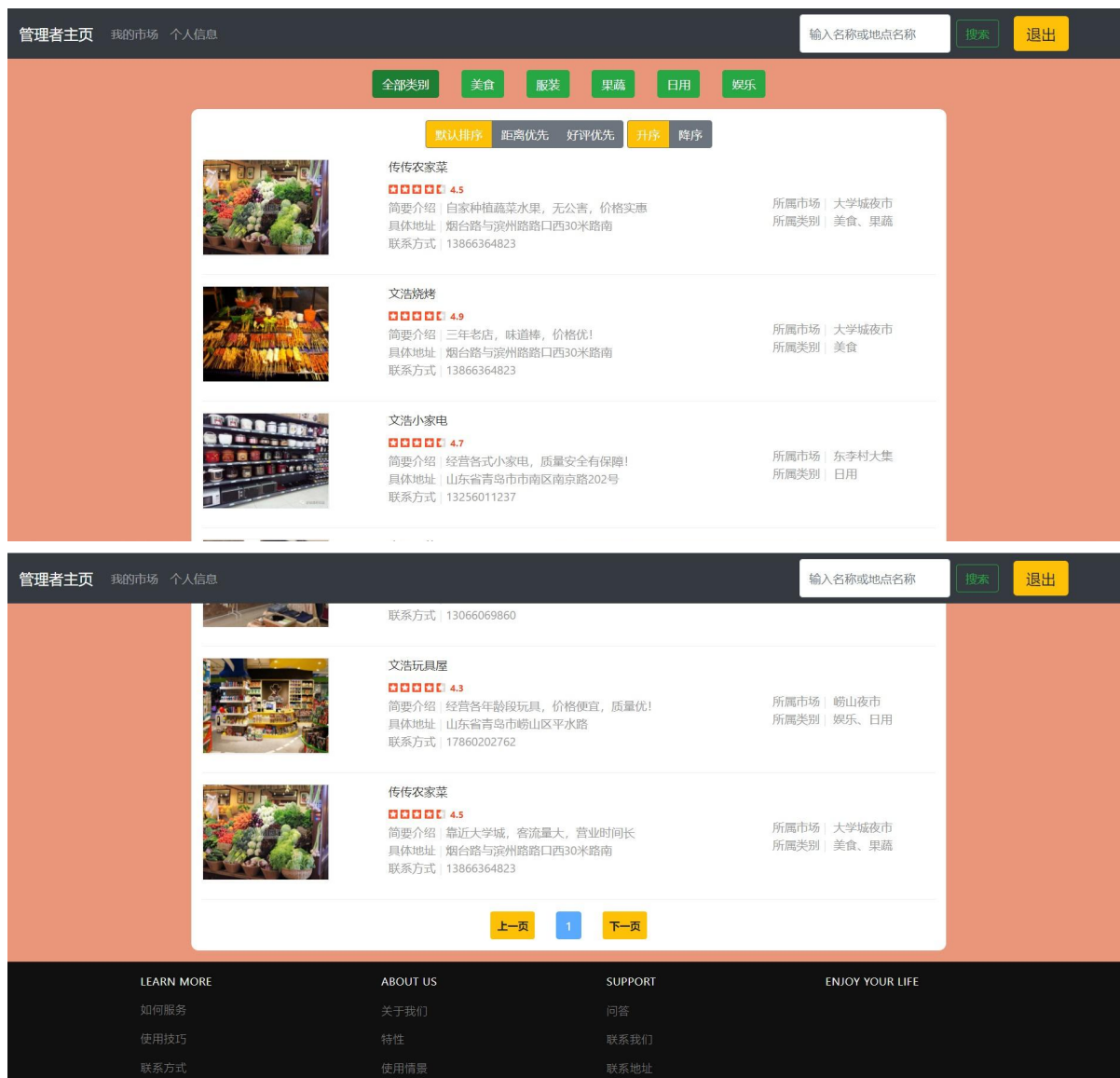
此外我们可以点击下一页、上一页加载更多的摊铺。



“摊易选”地摊管理系统

上方导航栏可以通过点击相应按钮进入“我的市场”页面、“个人信息”页面，此外通过点击退出按钮可以退出登录，并重新跳转至登录页面。

下方页脚可以通过点击友情链接进入相应的服务辅助网站。



8.2.4 管理者身份“摊铺详情”页面

当管理者在主页中点击摊铺时，会进入该摊铺的“摊铺详情”页面，展示该摊铺的所有详细信息。信息展示框顶部搭配有轮播图，可展示该摊铺上传的全部图片。

管理者可点击“我要监管”按钮，后台会将该管理者与摊铺设置对应关系，管理者即可对该摊铺进行检查等日常监管。此外，管理者还可以根据后台提示选择进入百度地图等网站，以便提供导航服务。




“摊易选” 地摊管理系统

摊铺详情

返回首页

个人信息

退出



我要监管!

摊铺名称:

文浩烧烤

所在市场 (摆摊地点):

大学城夜市

地址:

烟台路与滨州路路口西30米路南

简介:

三年老店, 味道棒, 价格优!

经营类别:

美食

联系方式:

13866364823

摊铺详情

返回首页

个人信息

退出

我要监管!

摊铺名称:

文浩烧烤

所在市场 (摆摊地点):

大学城夜市

地址:

烟台路与滨州路路口西30米路南

简介:

三年老店, 味道棒, 价格优!

经营类别:

美食

联系方式:

13866364823

8.2.5.1 管理者身份“我的市场”页面

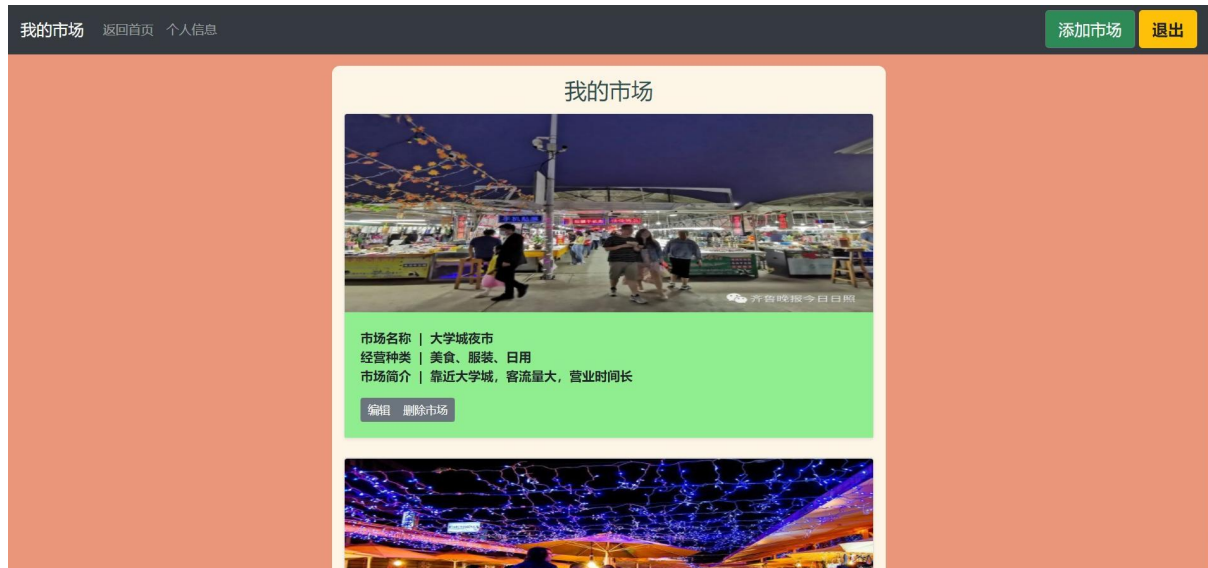
当管理者点击导航栏“我的市场”按钮时,即可进入“我的市场”页面。该页面展示该管理者创建的所有摆摊地点(市场),只有创建市场的管理者才可以维护该市场的信息。例如管理者可以删除已存在市场,或者修改已存在市场的信息。

对于管理者创建的每个市场,我们可以点击每个市场对应的“删除市场”按钮,将该市场删除。但是删除功能只有在该市场没有摊铺营业的状态下才可以完成,因此我们需要等待市场的所有摊铺停止营业。此外,我们可以点击编辑按钮进入编辑页面,可以修改所创建市场的基本信息。

在导航栏中有“添加市场”按钮,点击可进入“添加市场”页面。



“摊易选” 地摊管理系统



8.2.5.2 管理者身份获取我的市场事务

展示我的市场（摆摊地点）页面

```
def my_market(request):
    if not request.session.get('is_login', None):
        # 如果没有登录, 跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "manager":
        target_index = "{identity}/index/".format(identity=identity)
        return redirect(target_index)

    # 如果登录, 跳转到 my_market 页面
    user_name = request.session.get('user_name', None) # 用户名
    market_list = db_get_my_market(user_name)
    return render(request, 'manager/my-market.html', {"market_list":
market_list})

def db_get_my_market(user_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuan',
password='tanyixuan1904',
                        database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()

    # 再找未营业的
    sql = "select id market_id, name market_name, category, introduction, "
```



```
\
    "(select url from pic_urls where category=\"market\" and
id=market_id limit 0, 1) pic_url " \
    "from market " \
    "where manager_id=" \
    "(select id from manager where
user_name=\"{user_name}\");".format(user_name=user_name)
cursor.execute(sql)
market_list = dictfetchall(cursor)

# 关闭数据库连接
cursor.close()
conn.close()
return market_list
```

8.2.6.1 管理者身份“创建市场”页面


管理者在“创建市场”页面可以添加新的市场（摆摊地点），其中管理者必须输入市场名称、选择市场经营范围、输入市场简介、市场容量、市场地址、联系方式，可选择上传图片，并且可以上传多张图片。此外，管理者可以根据地图，在地图上选择地点，并且选择地点后可以自动填充市场地址，并且可以将经纬度信息传递给后台存储，从而可以为之后的定位提供必要信息。

所有信息都填写完毕后，可以点击提交按钮，即可创建新的市场，并自动跳转至我的市场页面。



“摊易选” 地摊管理系统

添加市场 返回首页 我的市场 个人信息 退出



市场地址

请输入您的市场地址!

您还未输入市场地址!

联系方式

请输入您的联系方式!

您还未输入联系方式!

未上传图片将使用默认图片! Browse

提交!

地图选择地点后，市场地址输入框会被自动填充：

地图选点



市场地址

山东省青岛市即墨区泰安街48号

8.2.6.2 管理者身份创建市场事务

```
def create_market(request):  
    if not request.session.get('is_login', None):  
        # 如果没有登录，跳转到登录页面  
        return redirect('/login/')  
    identity = request.session.get('identity', None) # 身份  
    if identity != "manager":  
        target_index = "{identity}/index/".format(identity=identity)  
        return redirect(target_index)  
  
    # 如果是 post 请求  
    if request.method == "POST":  
        user_name = request.session.get('user_name', None)  
        market_name = request.POST.get("market-name") # 摊铺名称  
        market_categorys = request.POST.getlist("market-category")
```




“摊易选” 地摊管理系统

```
market_category = "" # 摊铺经营类别
for i in range(len(market_categorys)):
    if i == 0:
        market_category += market_categorys[i]
    else:
        market_category += ('、' + market_categorys[i])
market_introduction = request.POST.get("market-introduction") # 经营简介
market_address = request.POST.get("market-address")
market_phone_number = request.POST.get("market-phone-number")
market_capacity = request.POST.get("market-capacity")
longitude = request.POST.get("longitude")
latitude = request.POST.get("latitude")

# 将这些信息存入数据库
result = db_create_market(user_name, market_name, market_category,
                           market_introduction, market_address, market_capacity,
                           market_phone_number, longitude, latitude)
# 如果摊铺创建成功
if result['success'] == 'true':
    market_pics = request.FILES.getlist("market-pics", None)

    # 图片的路径
    pic_urls = []
    for pic_file in market_pics:
        pic_urls.append('/images/market/' + market_name +
str(market_pics.index(pic_file)) + '.jpg')

    # 将摊铺的图片都存储到根目录/static/images/market
    if market_pics is not None:
        for pic_file in market_pics:
            with open('static' + pic_urls[market_pics.index(pic_file)], 'wb') as
market_pic_file:
                for chunk in pic_file.chunks():
                    market_pic_file.write(chunk)

    # 将图片路径存到数据表 pic_urls 中
    db_create_pic_url(result['market_id'], 'market', pic_urls)
    return redirect('/manager/index/')

# 如果不是 post 请求
return render(request, "manager/create-market.html")
```



“摊易选” 地摊管理系统

```
def db_create_market(user_name, market_name, market_category, market_introduction,
market_address, market_capacity,
                        market_phone_number, longitude, latitude):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                            database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
    """
    sql = "insert into market" \
        "(seller_id, name, introduction, category, location_longitude,
location_latitude, status)" \
        "values " \
        "((select id from seller where user_name=\"{user_name}\"), \"{market_name}\",
\"{market_introduction}\", " \
        "\"{market_category}\", 0, 0, 0);".format(user_name=user_name,
market_name=market_name,
                                                market_category=market_category,
market_introduction=market_introduction)
    """
    sql = "insert into market" \
        "(name, address, introduction, area_northwest_longitude,
area_northwest_latitude, east_dis, south_dis, " \
        "capacity, category, phone_number, current_capacity, manager_id, mark)" \
        "values " \
        "(\">{market_name}\", \">{address}\", \">{intro}\", \">{longitude}\",
\"{latitude}\", \">{east_dis}\", " \
        "\"{south_dis}\", \"{cap}\", \">{category}\", \">{phone_number}\",
\"{current_cap}\", " \
        "(select id from manager where user_name=\"{user_name}\"), " \
        "\"{mark}\");".format(market_name=market_name,
                                address=market_address,
                                intro=market_introduction,
                                longitude=longitude,
                                latitude=latitude,
                                east_dis=1,
                                south_dis=1,
                                cap=market_capacity,
                                category=market_category,
                                phone_number=market_phone_number,
                                current_cap=market_capacity,
                                user_name=user_name,
                                mark=0)
```



```
print(sql)
try:
    cursor.execute(sql)
    # 成功了就提交
    conn.commit()
    print("Debug: 测试是否成功插入数据")
    result = {"success": "true"}
    # 还要查询 market_id
    sql = "select @@identity;"
    cursor.execute(sql)
    result['market_id'] = cursor.fetchone()[0]
except:
    # 回退
    conn.rollback()
    result = {"success": "false", "market_name_repeat": "false"}

# 关闭数据库连接
cursor.close()
conn.close()
return result
```

8.2.7.1 摊主身份主页面

当我们选择摊主身份登录并输入正确的账号密码之后，我们进入摊主身份主页面。摊主身份主页面主要展示可供选择的摆摊地点，我们可以选择经营类别来选择特定的摆摊地点，我们还可以通过选择排序依据来更改摆摊地点推荐标准，例如按照距离排序和按照评分排序，两种排序方式均可自定义升序或者降序。在上方搜索框我们可以输入想要搜索的摆摊地点名称或者地址来得到特定的摆摊地点，支持模糊搜索。

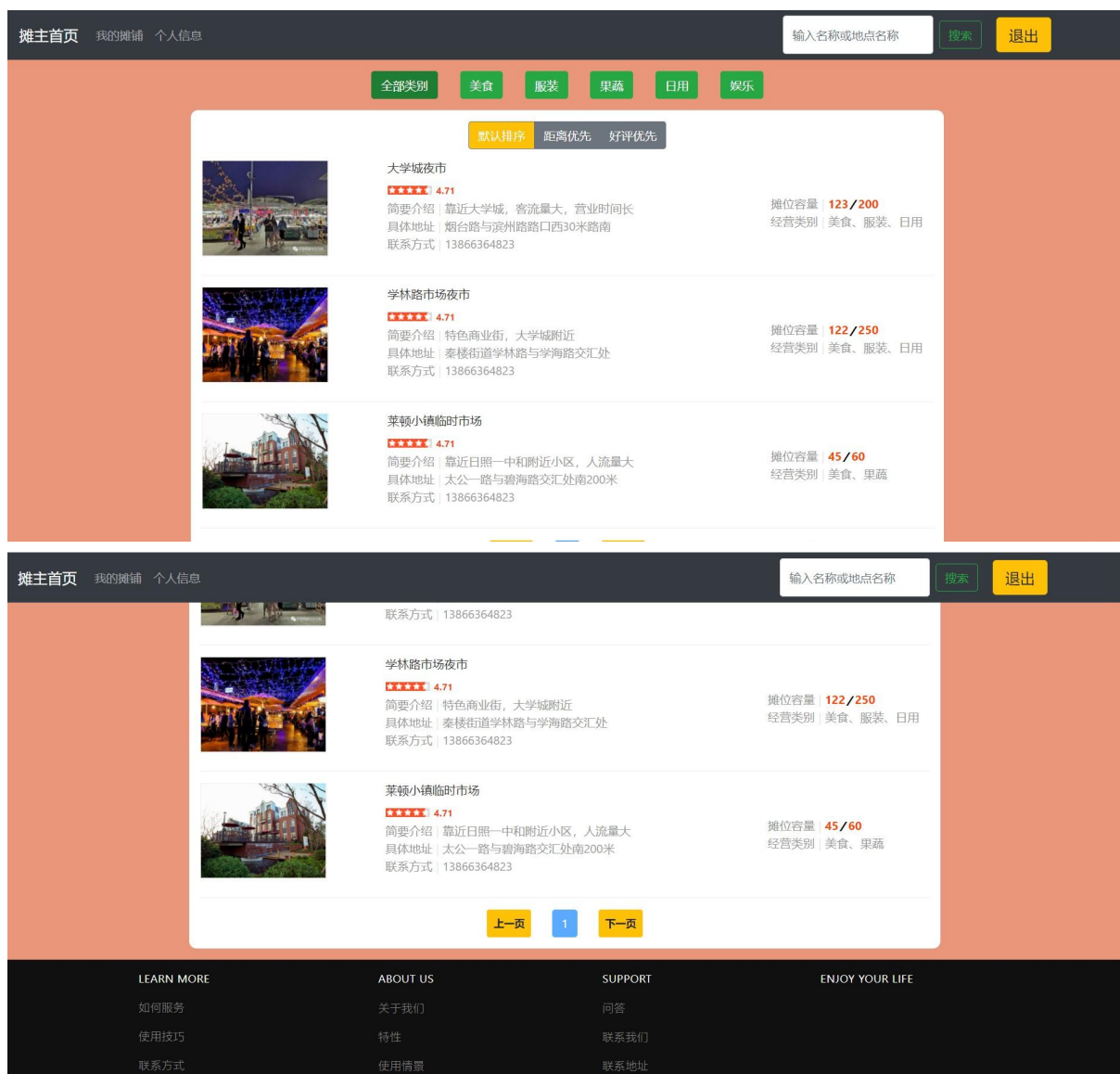
此外我们可以点击下一页、上一页加载更多的摆摊地点。

上方导航栏可以通过点击相应按钮进入“我的摊铺”页面、“个人信息”页面，此外通过点击退出按钮可以退出登录，并重新跳转至登录页面。

下方页脚可以通过点击友情链接进入相应的服务辅助网站。



“摊易选” 地摊管理系统



8.2.7.2 摊主身份获取摆摊地点列表事务

```
def get_market(request):  
    pageNo = request.POST.get("pageNo")  
    pageSize = request.POST.get("pageSize")  
    sortItem = request.POST.get("sortItem")  
    category = request.POST.get("category")  
    searchItem = request.POST.get("searchItem")  
    sortBasis = request.POST.get("sortBasis")  
  
    # 距离排序, 接收经纬度  
    if sortItem != "" and sortItem != "mark":  
        sortItem = request.POST.getlist("sortItem")
```



“摊易选” 地摊管理系统

```
market_list = db_get_market(pageNo, pageSize, sortItem, category, sortBasis,
searchItem)
```

```
response = {} # 用于返回结果
if len(market_list) == 0:
    response["success"] = "false"
else:
    response["success"] = "true"
    response["marketList"] = market_list
return JsonResponse(response, safe=False)
```

```
def db_get_market(pageNo, pageSize, sortItem, category, sortBasis, searchItem):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
```

```
database='tanyixuan', charset='utf8', port=3306)
```

```
cursor = conn.cursor()
```

```
pageBegin = (int(pageNo)-1)*int(pageSize) # 起始项 为了分页
```

```
category_radio = '%' + category + '%' # 为了得到特定类别的摆摊地点
```

```
search_item = '%' + searchItem + '%' # 为了得到名称中符合条件的摆摊地点
```

```
# 按照评分
```

```
if sortItem == "mark":
    sql = "select id, name, address, introduction, current_capacity, capacity,
category, phone_number, mark, " \
        "(select url from pic_urls where category = \"market\" and id = market.id limit
0, 1) pic_url from market " \
        "where current_capacity > 0 and category like \"{radio}\" and " \
        "(name like \"{search_item}\" or address like \"{search_item}\") order by mark
{sortBasis} limit " \
        "{pageBegin}, {pageSize};" \
        .format(search_item=search_item, radio=category_radio, pageBegin=pageBegin,
        pageSize=pageSize, sortBasis=sortBasis)
```

```
# 按照默认排序
```

```
elif sortItem == "":
    sql = "select id, name, address, introduction, current_capacity, capacity,
category, phone_number, mark, " \
        "(select url from pic_urls where category = \"market\" and id = market.id limit
0, 1) pic_url from market " \
        "where current_capacity > 0 and category like \"{radio}\" and " \
        "(name like \"{search_item}\" or address like \"{search_item}\") limit
{pageBegin}, {pageSize};" \
```



“摊易选” 地摊管理系统

```
.format(search_item=search_item, radio=category_radio, pageBegin=pageBegin,
pageSize=pageSize)
# 按照地理位置
else:
    sql = "select id, name, address, introduction, current_capacity, capacity,
category, phone_number, mark, " \
        "(select url from pic_urls where category = \"market\" and id = market.id
limit 0, 1) pic_url from market " \
        "where current_capacity > 0 and category like \"{radio}\" and " \
        "(name like \"{search_item}\" or address like \"{search_item}\") order by" \
        "power({latitude} - area_northwest_latitude, 2) + power({longitude} -
area_northwest_longitude, 2) " \
        "{sortBasis} limit {pageBegin}, {pageSize};" \
    .format(search_item=search_item, radio=category_radio,
            latitude=sortItem[0], longitude=sortItem[1], pageBegin=pageBegin,
            pageSize=pageSize, sortBasis=sortBasis)

    cursor.execute(sql)
    print(sql)
    market_list = dictfetchall(cursor)
    cursor.close()
    conn.close()
    return market_list
```

8.2.8 摊主身份“摆摊地点详情”页面

当摊主在主页中点击摆摊地点时，会进入该摆摊地点的“摆摊地点详情”页面，展示该摆摊地点的所有详细信息。信息展示框顶部搭配有轮播图，可展示该摊铺上传的全部图片。

当点击“我要摆摊”按钮时，系统会记住该摆摊地点的信息，跳转到摊铺展示页面，在该页面可以选择处于歇业状态的摊铺开始营业，从而实现选择歇业摊铺在选定摆摊地点营业的功能。





“摊易选” 地摊管理系统

摆摊地点详情 返回首页 个人信息 退出

我要摆摊!

摆摊地点名称: 大学城夜市

摊位总容量: 200 摊位剩余容量: 123

地址: 烟台路与滨州路路口西30米路南

简介: 靠近大学城, 客流量大, 营业时间长

允许经营类别: 美食、服装、日用

联系方式: 13866364823

8.2.9.1 摊主身份“我的摊铺”页面

当摊主点击导航栏“我的摊铺”按钮时,即可进入“我的摊铺”页面。该页面展示该摊主创建的所有摊铺,这些摊铺又分为营业状态摊铺和歇业状态摊铺,摊主可以点击营业状态摊铺的“停止营业”按钮,该摊铺会转为歇业状态。同样我们可以点击歇业状态摊铺的开始营业按钮,该摊铺会转为营业状态。需要注意的是,营业时必须要有对应的摆摊地点,所以只有当我们是“摆摊地点详情”页面通过“我要摆摊”按钮跳转到“我的摊铺”页面时,我们才能开始营业,否则会自动跳转进入摆摊地点展示页面,即主页面。

对于处于歇业状态的摊铺,我们可以点击每个摊铺对应的“删除摊铺”按钮,将该摊铺删除。而对处于营业状态的摊铺,我们需要先停止营业,才能将该摊铺删除。

在导航栏中有“添加摊铺”按钮,点击可进入“添加摊铺”页面。

我的摊铺 返回首页 个人信息 添加摊铺 退出

营业摊铺




摊铺名称 | 传传农家菜
摆摊地点 | 学林路市场夜市
摆摊位置 | 泰康街道学林路与学海路交汇处
停止营业 编辑 营业状态

歇业摊铺



摊铺名称 | 传传眼镜店
经营种类 | 日用
摊铺简介 | 眼镜价格低, 质量优, 不坑人! 欢迎选购!
开始营业 编辑 删除摊铺 歇业状态





8.2.9.2 摊主身份获取“我的摊铺”事务

展示我的摊位页面

```
def my_booth(request):
    if not request.session.get('is_login', None):
        # 如果没有登录, 跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "seller":
        target_index = "{identity}/index/".format(identity=identity)
        return redirect(target_index)

    # 如果登录, 跳转到 my_booth 页面
    request.session["market_id"] = 0 # 要摆摊的地点设置为 0, 这里只是单纯展示我的摊位
    user_name = request.session.get('user_name', None) # 用户名
    my_booth_list = db_get_my_booth(user_name)
    return render(request, 'seller/my-booth.html', {"market_id": 0, "my_booth_list":
my_booth_list})
```

```
def db_get_my_booth(user_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                        database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()

    # 先找正在营业的
    sql = "select " \
        "booth.id booth_id, " \
        "booth.name booth_name, " \
        "market.id market_id, " \
        "market.name market_name, " \
        "market.address market_address, " \
        "(select url from pic_urls where category=\"booth\" and id = booth_id limit 0,
1) pic_url " \
        "from " \
        "((booth join business on booth.id=business.booth_id) join market on
market.id=business.market_id) " \
        "where " \
        "booth.seller_id = (select id from seller where user_name=\"{user_name}\")
order by business.start_time" \
        .format(user_name=user_name)
```



```
cursor.execute(sql)
my_booth_list = {"open": dictfetchall(cursor)}

# 再找未营业的
sql = "select id booth_id, name booth_name, category, introduction, " \
      "(select url from pic_urls where category=\"booth\" and id=booth_id limit 0, 1) \
pic_url " \
      "from booth " \
      "where status=0 and seller_id=" \
      "(select id from seller where \
user_name=\"{user_name}\");".format(user_name=user_name)
cursor.execute(sql)
my_booth_list["close"] = dictfetchall(cursor)

# 关闭数据库连接
cursor.close()
conn.close()
return my_booth_list
```

8.2.9.3 摊主身份开始营业事务

开始营业

```
def open_booth(request):
    if not request.session.get('is_login', None):
        # 如果没有登录, 跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "seller":
        target_index = "{identity}/index/".format(identity=identity)
        return redirect(target_index)

    # 如果是 post 请求
    if request.method == "POST":
        market_id = request.POST.get("market_id")
        booth_name = request.POST.get("booth_name")
        user_name = request.session.get("user_name", None)
        result = db_open_booth(market_id, booth_name, user_name)
        # 返回结果
        return JsonResponse(result)
    # 否则返回主页
    else:
        return redirect("/seller/index/")
```



```
def db_open_booth(market_id, booth_name, user_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                        database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
    sql = "insert into business(booth_id, market_id, start_time) " \
    "values " \
    "((select id from booth where name=\"{booth_name}\" and seller_id = " \
    "(select id from seller where user_name=\"{user_name}\")), " \
    "{market_id}, now());".format(booth_name=booth_name, user_name=user_name,
market_id=market_id)

    try:
        cursor.execute(sql)
        # 还要修改 booth 中的 status
        sql = "update booth set status = 1 " \
        "where name=\"{booth_name}\" and seller_id = " \
        "(select id from seller where user_name=\"{user_name}\");" \
        .format(booth_name=booth_name, user_name=user_name)
    try:
        cursor.execute(sql)
        # 将摆摊地点的可容纳摊位数量-1
        sql = "update market set current_capacity = current_capacity - 1 " \
        "where id={market_id};".format(market_id=market_id)
    try:
        cursor.execute(sql)
        # 成功了就提交
        conn.commit()
        result = {"success": "true"}
    except:
        # 回退
        conn.rollback()
        result = {"success": "false"}
    except:
        # 回退
        conn.rollback()
        result = {"success": "false"}
    except:
        # 回退
        conn.rollback()
```



```
result = {"success": "false"}

# 关闭数据库连接
cursor.close()
conn.close()
return result
```

8.2.9.4 摊主身份停止营业事务

停止营业

```
def close_booth(request):
    if not request.session.get('is_login', None):
        # 如果没有登录, 跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "seller":
        target_index = "{identity}/index/".format(identity=identity)
        return redirect(target_index)

    # 如果是 post 请求
    if request.method == "POST":
        market_id = request.POST.get("market_id")
        booth_name = request.POST.get("booth_name")
        user_name = request.session.get("user_name", None)
        result = db_close_booth(market_id, booth_name, user_name)
        # 返回结果
        return JsonResponse(result)
    # 否则返回主页
    else:
        redirect("/seller/index/")

def db_close_booth(market_id, booth_name, user_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
        password='tanyixuan1904',
        database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()

    sql = "delete from business " \
        "where market_id = {market_id} and booth_id = " \
        "(select id from booth where name='{booth_name}' and seller_id = " \
        "(select id from seller where user_name='{user_name}'))";"
```



```
.format(market_id=market_id, booth_name=booth_name, user_name=user_name)

try:
    cursor.execute(sql)
    # 还要修改 booth 中的 status
    sql = "update booth set status = 0 " \
          "where name=\"{booth_name}\" and seller_id = " \
          "(select id from seller where user_name=\"{user_name}\");" \
          .format(booth_name=booth_name, user_name=user_name)
    try:
        cursor.execute(sql)
        # 将摆摊地点的可容纳摊位数量+1
        sql = "update market set current_capacity = current_capacity + 1 " \
              "where id={market_id};".format(market_id=market_id)
        try:
            cursor.execute(sql)
            # 成功了就提交
            conn.commit()
            result = {"success": "true"}
        except:
            # 回退
            conn.rollback()
            result = {"success": "false"}
    except:
        # 回退
        conn.rollback()
        result = {"success": "false"}
except:
    # 回退
    conn.rollback()
    result = {"success": "false"}
# 关闭数据库连接
cursor.close()
conn.close()
return result
```

8.2.9.5 摊主身份删除摊铺事务

这里需要注意，当摊主决定删除某个摊铺时，我们要一并删除静态文件中存储的该摊铺的图片。由于我们使用了一个实体集 `pic_urls` 来存储所有图片的路径，所以我们要首先根据要删除摊铺的 `id` 在数据表 `pic_urls` 中找到属于该摊铺的所有图片路径，这由函数 `db_delete_pic_url` 实现。再调用相关函数将得到的图片路径全部从静态文件夹中删除。



删除摊铺

```
def delete_booth(request):
    if not request.session.get('is_login', None):
        # 如果没有登录, 跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "seller":
        target_index = "{identity}/index/".format(identity=identity)
        return redirect(target_index)

    # 如果是 post 请求
    if request.method == "POST":
        booth_name = request.POST.get("booth_name")
        user_name = request.session.get("user_name", None)
        # 返回删除的 booth 的 id, 为了方便删除 pic_url
        result = db_delete_booth(booth_name, user_name)
        if result["success"] == "true":
            # 删除 pic_urls 中的图片链接 并返回这些链接, 为了在静态文件夹中把他们删除
            pic_urls = db_delete_pic_url(result["booth_id"], "booth")
            # 在静态文件夹中删除
            for pic_url in pic_urls:
                os.remove('static' + pic_url[0])
            # 返回结果
            return JsonResponse(result)

        # 否则返回主页
    else:
        return redirect("/seller/index/")

def db_delete_booth(booth_name, user_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
        password='tanyixuan1904',
        database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
    sql = "select id from booth where name=\"{booth_name}\" and seller_id=" \
        "(select id from seller where "
    user_name="{user_name}");".format(booth_name=booth_name, user_name=user_name)
    try:
        cursor.execute(sql)
        booth_id = cursor.fetchone() # 是一个元组形式
        if booth_id is None:
            result = {"success": "false"}
        else:
            booth_id = booth_id[0]
```



“摊易选” 地摊管理系统

```
result = {"booth_id": booth_id}
sql = "delete from booth where id={booth_id};".format(booth_id=booth_id)
try:
    cursor.execute(sql)
    # 提交
    conn.commit()
    result["success"] = "true"
except:
    # 回退
    conn.rollback()
    result["success"] = "false"
except:
    # 回退
    conn.rollback()
    result = {"success": "false"}

# 关闭数据库连接
cursor.close()
conn.close()
return result

def db_delete_pic_url(id, category):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                                database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()

    # 返回路径
    sql = "select url from pic_urls where category=\"{category}\" and
id={id};".format(category=category, id=id)
    cursor.execute(sql)
    pic_urls = cursor.fetchall()

    # 删除他们
    sql = "delete from pic_urls where category=\"{category}\" and
id={id};".format(category=category, id=id)
    try:
        cursor.execute(sql)
        # 成功了就提交
        conn.commit()
    except:
        # 回退
```




```
conn.rollback()

# 关闭数据库连接
cursor.close()
conn.close()

# 返回路径，元组类型，在静态文件夹中把他们删除
return pic_urls
```

8.2.10.1 摊主身份“添加摊铺”页面

摊主在“添加摊铺”页面可以添加新的摊铺，其中摊主必须输入摊铺名称、选择摊铺经营范围、输入摊铺简介，可选择上传图片，并且可以上传多张图片。所有信息都填写完毕后，可以点击提交按钮，即可创建新的摊铺，并自动跳转至我的摊铺页面。

8.2.10.2 摊主身份“添加摊铺”事务

需要注意，当摊主创建摊铺时，我们要将其上传的图片全部存入静态文件夹，并将文件路径存入数据表 `pic_urls` 中，这里我们通过函数 `db_create_pic_url` 实现。

此外，我们的逻辑是同一摊主不允许创建同名的摊铺，这里我们通过函数 `db_check_booth_repeat` 来实现。

```
# 创建摊铺
def create_booth(request):
    if not request.session.get('is_login', None):
        # 如果没有登录，跳转到登录页面
        return redirect('/login/')
    identity = request.session.get('identity', None) # 身份
    if identity != "seller":
```



“摊易选” 地摊管理系统

```
target_index = "{identity}/index/".format(identity=identity)
return redirect(target_index)

# 如果是 post 请求
if request.method == "POST":
    user_name = request.session.get('user_name', None)
    booth_name = request.POST.get("booth-name") # 摊铺名称
    booth_categorys = request.POST.getlist("booth-category")
    booth_category = "" # 摊铺经营类别
    for i in range(len(booth_categorys)):
        if i == 0:
            booth_category += booth_categorys[i]
        else:
            booth_category += (',' + booth_categorys[i])
    booth_introduction = request.POST.get("booth-introduction") # 经营简介
    # 将这些信息存入数据库
    result = db_create_booth(user_name, booth_name, booth_category,
booth_introduction)
    # 如果摊铺创建成功
    if result['success'] == 'true':
        booth_pics = request.FILES.getlist("booth-pics", None)

        # 图片的路径
        pic_urls = []
        for pic_file in booth_pics:
            pic_urls.append('/images/booth/' + booth_name +
str(booth_pics.index(pic_file)) + '.jpg')

        # 将摊铺的图片都存储到根目录/static/images/booth
        if booth_pics is not None:
            for pic_file in booth_pics:
                with open('static' + pic_urls[booth_pics.index(pic_file)], 'wb') as
booth_pic_file:
                    for chunk in pic_file.chunks():
                        booth_pic_file.write(chunk)

        # 将图片路径存到数据表 pic_urls 中
        db_create_pic_url(result['booth_id'], 'booth', pic_urls)
    return redirect('/seller/my-booth/')

# 如果不是 post 请求
return render(request, "seller/create-booth.html")
```



```
def db_create_booth(user_name, booth_name, booth_category, booth_introduction):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                            database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
    sql = "insert into booth" \
        "(seller_id, name, introduction, category, location_longitude,
location_latitude, status)" \
        "values" \
        "((select id from seller where user_name=\"{user_name}\"), \"{booth_name}\",
\"{booth_introduction}\", " \
        "\"{booth_category}\", 0, 0, 0);".format(user_name=user_name,
booth_name=booth_name,
                                                booth_category=booth_category,
booth_introduction=booth_introduction)
    try:
        cursor.execute(sql)
        # 成功了就提交
        conn.commit()
        result = {"success": "true"}
        # 还要查询 booth_id
        sql = "select @@identity;"
        cursor.execute(sql)
        result['booth_id'] = cursor.fetchone()[0]
    except:
        # 回退
        conn.rollback()
        result = {"success": "false", "booth_name_repeat": "false"}

    # 关闭数据库连接
    cursor.close()
    conn.close()
    return result

def db_check_booth_repeat(user_name, booth_name):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                            database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()
```



```
# 先看看有没有重复的摊铺名称
sql = "select id from booth where name=\"{booth_name}\" and seller_id=" \
      "(select id from seller where
user_name=\"{user_name}\");".format(booth_name=booth_name, user_name=user_name)
cursor.execute(sql)
print(cursor.fetchone())
if cursor.fetchone() is None:
    result = {"success": "true"}
else:
    result = {"success": "false"}
return result

# id 是身份的 id, category 是类别, 字符串类型
def db_create_pic_url(id, category, pic_urls):
    conn = MySQLdb.connect(host='localhost', user='tanyixuanU',
password='tanyixuan1904',
                           database='tanyixuan', charset='utf8', port=3306)
    cursor = conn.cursor()

    for url in pic_urls:
        sql = "insert into pic_urls (id, category, url) values ({id}, \"{category}\",
        \"{url}\");" \
        .format(id=id, category=category, url=url)
        try:
            cursor.execute(sql)
            # 成功了就提交
            conn.commit()
        except:
            # 回退
            conn.rollback()
    # 关闭数据库连接
    cursor.close()
    conn.close()
```

8.2.11 消费者身份主页面

当我们选择消费者身份登录并输入正确的账号密码之后,我们进入消费者身份主页面。消费者身份主页面主要展示可以消费的摊铺,我们可以选择经营类别来选择特定的摊铺,我们还可以通过选择排序依据来更改摊铺推荐标准,例如按照距离排序和按照评分排序,两种排序方式均可自定义升序或者降序。在上方搜索框我们可以输入想要搜索的摊铺名称或者地址来得到特定的摊铺,支持模糊搜索。

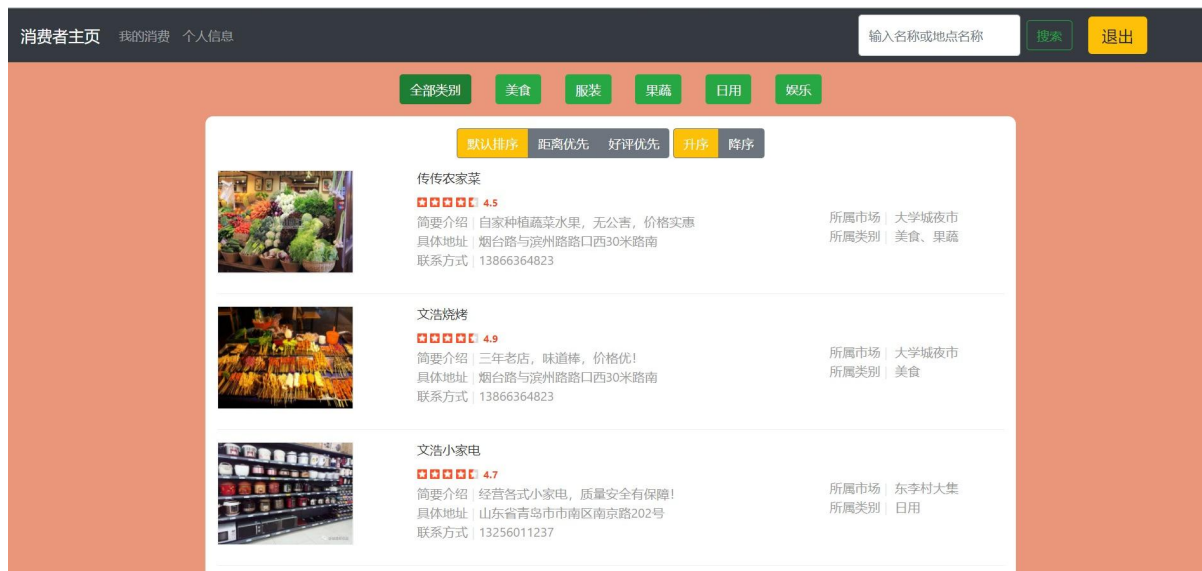


“摊易选”地摊管理系统

此外我们可以点击下一页、上一页加载更多的摊铺。

上方导航栏可以通过点击相应按钮进入“我的消费”、“个人信息”页面，此外通过点击退出按钮可以退出登录，并重新跳转至登录页面。

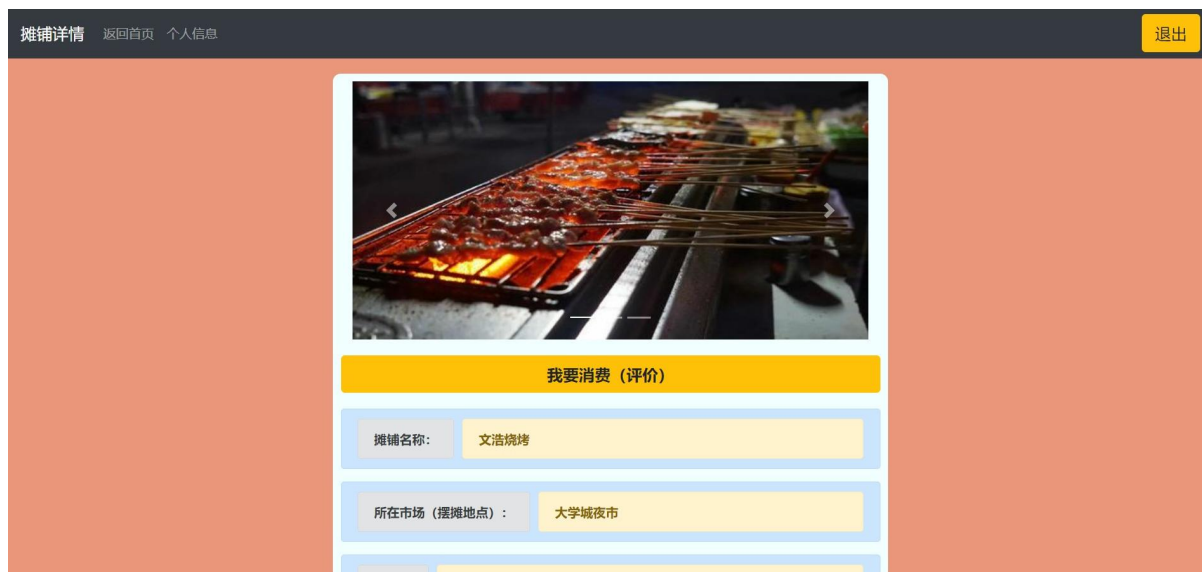
下方页脚可以通过点击友情链接进入相应的服务辅助网站。



8.2.12 消费者身份“摊铺详情”页面

当消费者在主页中点击摊铺时，会进入该摊铺的“摊铺详情”页面，展示该摊铺的所有详细信息。信息展示框顶部搭配有轮播图，可展示该摊铺上传的全部图片。

消费者可点击“我要消费（点评）”按钮，后台会跳转进入美团、微信支付、支付宝等线上下单网站进行进一步的消费，或者跳转进入评价页面，此外，消费者也可以根据后台提示选择跳转进入百度地图等网站提供导航服务。





九、动态反馈

9.1 登录页面

用户在登录界面输入用户名并输入密码，点击登录按钮后，后台根据用户名和密码的匹配情况，会判断用户犯了什么样的错误，从而引导用户进行正确的操作。

9.1.1 用户密码错误

当用户输入的密码错误时，密码输入框后会动态返回“密码错误！”的字样。



9.1.2 用户名不存在

当用户输入的用户名并不存在时，用户名输入框后会动态返回“账号不存在！”的字样。





“摊易选”地摊管理系统

9.1.3 用户名为空

当用户未输入用户名时，用户名输入框后会动态返回“账号不能为空！”的字样，提醒用户输入用户名（账号）。

The screenshot shows a login interface with a white header containing the title '登录' (Login). Below the header is a blue background area. At the top of this area are three radio buttons for user roles: '我是消费者' (I am a consumer), '我是摊主' (I am a stall owner), and '我是管理者' (I am a manager). Below these are two input fields. The first field is for the username, with the placeholder text '请输入账号' (Please enter account) and a red error message '账号不能为空!' (Account cannot be empty!) to its right. The second field is for the password, with the placeholder text '请输入密码' (Please enter password) and a red error message '密码不能为空!' (Password cannot be empty!) to its right. Below the password field is a checkbox labeled '记住我' (Remember me). At the bottom of the blue area is a large orange button labeled '登录' (Login). Below the blue area is a white footer containing the text '忘记密码?' (Forgot password?) and '还没有账户? 立即注册' (No account? Register now).

9.1.4 密码为空

当用户未输入密码时，密码输入框后会动态返回“密码不能为空！”的字样，提醒用户输入密码。

The screenshot shows the same login interface as the previous one, but with the username field filled with the text 'eecspan'. The password field now has the red error message '密码不能为空!' (Password cannot be empty!) to its right. The '记住我' (Remember me) checkbox is still present. The orange '登录' (Login) button and the footer text remain the same.



9.2 注册页面

用户在注册界面输入各项信息，点击注册按钮后，后台根据用户注册身份检查用户信息的完整度情况，会判断用户是否有信息遗漏或者信息输入错误，从而引导用户进行正确的操作。

9.2.1 必填信息遗漏

当用户有必填信息遗漏时，会在第一个遗漏信息输入框后提示，并且中止向后台提交信息。

注册您的账户

请选择您想要注册的身份

☒ 我是消费者 ☐ 我是摊主 ☐ 我是管理者

用户账号

昵称

请填写此字段。

联系方式

请输入密码

确认密码

点击注册

9.2.2 用户名重复

当用户设置用户名时，如果该用户名已存在，那么系统会在用户名输入框后提示。

注册您的账户

请选择您想要注册的身份

☒ 我是消费者 ☐ 我是摊主 ☐ 我是管理者

eecspan

用户名已存在!

eecspan

15662699817

...

...

点击注册



9.2.3 格式不正确

当用户输入联系方式或者身份证号时，如果格式不正确，系统会在相应输入框后进行提示。

注册您的账户

请选择您想要注册的身份

☒ 我是消费者 ☐ 我是摊主 ☐ 我是管理者

电话格式不正确！

点击注册

9.2.4 确认密码与原密码不相符

当用户设置密码并确认密码时，如果后者与前者不相符，那么系统会提示。

注册您的账户

请选择您想要注册的身份

☒ 我是消费者 ☐ 我是摊主 ☐ 我是管理者

确认密码错误！

点击注册

9.3 摊主创建摊铺页面

9.3.1 必填信息缺失

当摊主在创建摊铺时，如有必填信息缺失，那么系统会给予相应提示。

此外，根据输入框的颜色，可以提示用户哪一些是必填项，哪一些是选填项。可以



“摊易选”地摊管理系统

看到上传照片是绿色输入框，说明为选填项。其余为红色输入框，说明为必填项。

我的摊铺 个人信息

摊铺名称

请输入您的摊铺名称!

☐ 美食
☐ 服装
☐ 果蔬
☐ 日用
☐ 娱乐
您还未选择类别!

摊铺简介

请输入您的摊铺简介!

您还未输入摊铺简介!

未上传图片将使用默认图片!

摊铺名称

☐ 美食
☐ 果蔬
☐ 日用
☐ 娱乐
您还未选择类别!

摊铺简介

请输入您的摊铺简介!

您还未输入摊铺简介!

未上传图片将使用默认图片!

摊铺名称

☐ 美食
☐ 服装
☐ 果蔬
☐ 日用
☒ 娱乐

摊铺简介

请输入您的摊铺简介!

您还未输入摊铺简介!

未上传图片将使用默认图片!



9.4 管理者创建市场页面

9.4.1 必填信息缺失

当管理者在创建市场时，如有必填信息缺失，那么系统会给予相应提示。

此外，根据输入框的颜色，可以提示用户哪一些是必填项，哪一些是选填项。可以看到上传照片是绿色输入框，说明为选填项。其余为红色输入框，说明为必填项。

The screenshot displays a web form for creating a market. The form is divided into several sections, each with a title and a corresponding input field. The input fields are color-coded: red for required fields and green for optional fields. The sections and their fields are as follows:

- 市场名称** (Market Name): A red input field with a red border and a red error message "请输入您的市场名称!" (Please enter your market name!). A tooltip "请填写此字段。" (Please fill in this field.) points to the field.
- 市场简介** (Market Introduction): A red input field with a red border and a red error message "请输入您的市场简介!" (Please enter your market introduction!). A tooltip "请填写此字段。" (Please fill in this field.) points to the field.
- 市场容量** (Market Capacity): A red input field with a red border and a red error message "请输入您的市场容量!" (Please enter your market capacity!). A tooltip "请填写此字段。" (Please fill in this field.) points to the field.
- 市场地址** (Market Address): A red input field with a red border and a red error message "请输入您的市场地址!" (Please enter your market address!). A tooltip "请填写此字段。" (Please fill in this field.) points to the field.
- 联系方式** (Contact Information): A red input field with a red border and a red error message "请输入您的联系方式!" (Please enter your contact information!). A tooltip "请填写此字段。" (Please fill in this field.) points to the field.
- 上传图片** (Upload Image): A green input field with a green border and a green error message "未上传图片将使用默认图片!" (No image uploaded, default image will be used!). A "Browse" button is next to the field.

At the bottom of the form, there is a blue "提交!" (Submit!) button.



十、总结

10.1 系统优点

- 1、系统设计较全面、设置了三种身份，管理者、摊主和消费者，系统解决了摊主无法寻找摆摊地点、消费者无法寻找摊铺和管理者无法定位摊主的难题。
- 2、界面较美观，软件提示反馈详细，操作简单，能够较好的满足需求。
- 3、数据库设计内容具体详细，条理清晰，关系明确，通过设置联系集等消除了部分冗余，是数据的查询等更加高效。
- 4、信息提示系统细致完善，对于用户可能发生的错误操作，给予错误信息提示。
- 5、对异常进行了处理，在用户操作不规范时给出错误信息。

10.2 系统不足

- 1、未检测系统的并发执行效率，未进行充分的压力检测，因此系统可能存在一些并发执行的问题。
- 2、数据库的设计还是存在一些冗余，还可以进行进一步的优化。

10.3 系统改进

- 1、可以对系统的一些功能进行进一步的完善，例如设置更加完善的评价机制，更齐全的服务，例如实时导航、对于管理者提供实时人流量预测，从而在拥挤的地方给予及时的预警提醒。
- 2、

10.4 经验与收获

经过这为期一个月的数据库课程设计，让我真正入门了 web 开发，第一次接触到了后端，打消了我此前的恐惧感。另外，在实践中发现实践与理论的差距，许多理论的想法在实践中会发生各种此前未预料到的问题，实践出真知！

在这次系统的设计和实现过程中，我更加深刻的了解了一个成熟的系统的开发流程，也意识到了前期准备的重要性。并且，第一次直观感受到一个完善、合理、高效的数据库设计对一个项目是多么重要。此外，通过设计项目，更加深刻地了解了 c/s 架构的实现流程和特点，学习到了 django 框架，接触并尝试了 ajax，为之后的开发奠定了基础。

当然，本次课程设计我制作的网站还存在一些缺陷的，由于时间较短，所以我的数据库设计上一些细节还需要优化。并且“摊易选”地摊管理系统是一个较为复杂的系统，登录身份多，提供的个性化服务也较多。并且我做的网站并未投入使用，所以在一些功



“摊易选”地摊管理系统

能上可能存在漏洞，但是出现的问题可以在后期需要的时候进行扩展。

整个系统开发历时将近一个月，从自己选题、到根据数据、事务特点设计数据库，并且从 0 开始自学 `django` 框架，自写前端页面和后端逻辑，一步一步都是遇到不会的东西去学，之后将其应用到项目中。虽然最后的项目还略显简陋，很多地方也存在问题，但是看见自己的网站能够运行，并且自己设计的复杂逻辑也能够无数遍排 `bug` 后正确执行，这其中的欣喜和成就感是很充足的。这短时间的体验让我体验到一点点课业学习之外的工程，为我未来的发展奠定了基础。

附. 参考文献

- [1] Abraham Silberschatz, Henry F. Korth, 《数据库系统概念》，机械工业出版社，2008。
- [2] 刘江，刘江的博客教程，<https://www.liujiangblog.com/course/django/>。