
目录

一、项目概述	5
1、项目开发背景	5
2、项目目标	5
二、可行性分析	6
2.1 初步调查	6
2.1.1 客人需求分析	6
2.1.2 雇员需求分析	6
2.1.3 管理层需求分析	7
2.1.4 预订系统需求分析	7
2.1.5 系统整体需求分析	7
2.2 可行性研究	8
2.2.1 系统功能概述	8
2.2.2 系统开发的可行性	9
三、系统分析	10
3.1 用例描述	10
3.1.1 整体用例图	10
3.1.2 客人	10
3.1.3 雇员	14
3.1.4 预订系统	15
3.1.5 管理层	17
3.2 类图	20

3.2.1 客人.....	20
3.2.2 雇员.....	22
3.2.3 预订系统.....	23
3.2.4 管理层	25
3.3 序列图.....	27
3.3.1 客人.....	27
3.3.2 雇员.....	30
3.3.3 管理层	33
3.3.4 预订系统.....	34
四、系统设计	36
4.1 总体设计	36
4.1.1 运行环境.....	36
4.1.2 系统组成结构.....	37
4.2 数据库设计.....	38
4.3 用户界面设计	40
4.3.1 顾客界面.....	40
4.3.2 雇员界面.....	43
4.3.3 管理层界面	45
4.4 后端连接设计	47
4.4.1 顾客.....	47
4.4.2 雇员.....	48
4.4.3 管理层	49

4.4.4 预订系统.....	50
五、系统操作说明	50
5.1 客户操作说明	50
5.2 雇员操作说明	51
5.3 管理层操作说明	51
六、软件测试	51
6.1 测试概要	51
6.1.1 概述.....	51
6.1.2 测试原则.....	52
6.2 测试方法	52
6.3 功能测试	53
6.4 维护	55

一、项目概述

1、项目开发背景

酒店，是提供安全、舒适，令利用者得到短期休息或睡眠空间的商业机构。随着旅游业的飞速发展和社会生活水平的不断提高，人们对酒店的要求也随之提高。因此，开发酒店管理系统能够使酒店内部集中管理和控制，能够快速反映酒店的经营状态以及收益情况，大大降低了管理层和雇员的劳动强度，提高了他们的劳动效率，也节省了客人订酒店的时间，为高质量的酒店服务提供保障。在此背景下，我们开发了 Oasis 酒店管理系统。

2、项目目标

通过本实验，本小组希望能够建成一套具有不断发展能力的完整的酒店管理系统。项目以提高酒店工作人员效率、酒店服务的速度及精度，改善顾客入住体验，减少工作差错为目标，能够实现服务管理的电子化、自动化，提高管理质量，为酒店更好地经营创造条件。主要表现在以下几个方面：

（1）软件将为酒店的经营服务：

该软件具有快速、准确、高效的特点，操作简单，界面友好美观，雇员、管理层能够快速从软件中查看各种信息，提高效率和服务质量。

- ① 能够统计客房客人的到达及离开信息；
- ② 雇员能够查看房间占用情况；
- ③ 雇员为顾客实现预订、修改预订和退订服务；
- ④ 能够统计客房预订信息及酒店的收益信息以供管理层查看。
- ⑤ 系统维护方便可靠，有较高的安全性，满足实用性、先进性的要求。

（2）软件为顾客提供最大方便：

顾客在使用订房服务时，能够按条件查询需要的房间。能够绑定银行卡在以及时付款，可以随时修改预订信息或退房，在此同时，系统会及时按原路退还钱款给顾客。在客人离开时还可以按照客人不同的预订方式为客人打印票据。为顾

客带来极大的便利。

二、可行性分析

2.1 初步调查

2.1.1 客人需求分析

在本系统中，客人能够：

- （1）查看房间。客人在登录软件后，能够查看房间类型、房间价格、能够预订的时间，并能根据要求进行筛选。
- （2）能够进行房间预订。客人在看到心仪的房间后能够及时预订，并且可以选择两种预订方式：预付金预订和常规预订。
- （3）能够及时修改自己的预订信息。包括修改房间类型、预订时间或进行退订。
- （4）能够修改个人信息。如修改绑定银行卡，手机号等等。

2.1.2 雇员需求分析

在本系统中，雇员能够：

- （1）在有客人进行预订或者修改过预订信息后，雇员能够及时确认该信息；
- （2）根据系统中客人的预订信息能够生成两种报表：
 - ① 每日到达报表：即预计在这天到达的客人列表。报表的每一行显示客人姓名、预订类型、预分配的房号数和离开日期，该报表按客人的姓名排序；
 - ② 每日入住报表：即当前还停留在宾馆的客人列表。报表的每一行显示房号、客人姓名和离开日期。如果客人当天离开，姓名前加注星号。该报表按照房号排序。
- （3）能够查看房间状态，为到达的客人分配房间；
- （4）在客人离开时可以根据预订系统中的信息为客人打印相应的票据：该票据

能够反映票据打印日期、客人姓名、房间号、达到日期、离开日期、入住的天数和总金额。对于预付金预订，该票据还能反映提前支付的日期和金额。在客人结账时把票据交给客人。

2.1.3 管理层需求分析

在本系统中，管理层需要：

- (1) 提前一年定好所有房间每晚的基价；
- (2) 能够在任意时间修改酒店房间的基价；
- (3) 能够管理房间信息，查看房间现有状态；
- (4) 能够查看预订系统和雇员生成的所有报表。

2.1.4 预订系统需求分析

- (1) 能够生成未来 30 天每晚已经被预订的房间数的报表和未来 30 天每晚的预计收益和总收益的报表；
- (2) 能够保留所有的更改信息（如客人预订房间、更改预订、支付房费、系统退款，管理层确定及更改基价等）；
- (3) 能够根据客人的预订信息生成票据，以供雇员打印。

2.1.5 系统整体需求分析

(1) 客人方面：在客人选择预订方式后，能够及时计算客人所需付款金额，在客人修改订单或者退订房间时，能够及时计算客人所需补交款项或系统需要退还款项，若为退款则及时退还。

有两种类型的预订：

① 预付金预订。客人在预定时将所有花费一次性付清。预付价格是基价的 75%。

预付金预订需要提前至少 30 天。如果取消预付金预订，将不退款。可以更改预

订（条件是能提供房间），此时将根据预定类型，进行金额的调整，如果金额减少了，将不退款。如果金额增加了，增加金额的金额将在更改时付款。

② 常规预订。对于常规预订，客人在停留的最后一天付款。预订时客人需要提供信用卡号码，如果信用卡没有得到确认，将无法完成预定。如果入住前 3 天内取消预订，需要支付第一天的房费。可以在任何时候进行常规预订，并随时可以更改（需得到确认）。

（2）雇员方面：

① 系统能够及时统计各类信息以供雇员查看，以便生成报表；

② 能够及时反馈顾客的订单信息以供雇员处理和确认；

③ 雇员能够按条件查询客人的预订信息，以便生成和打印票据。

（3）管理层方面：

① 系统能够及时反馈房间各种信息，如：当前房间预订状态、入住状态、房间基价等以供管理层及时查看或修改；

② 系统能够及时反馈雇员和预订系统生成的报表，以供管理员查看；

③ 基价的更改不影响任何现有的预订。

（4）预订系统方面：

① 系统能够统计客人预订信息及到达信息，以便生成报表；

② 系统能够及时计算预计房间收益以反馈给预订系统，使预订系统能够及时生成报表；

③ 系统能够保留所有记录，并且在每个工作日的最后进行文件备份。

2.2 可行性研究

2.2.1 系统功能概述

综上所述，系统应该拥有的功能有：

（1）客人能够预订、修改预订信息、退订，付款和系统及时退款；

（2）雇员能够查看和确认客人的预订信息，能够根据预订信息生成报表，能够

为客人打印票据；

(3) 管理层能够随时修改房间基价，查看房间状态及报表；

(4) 预订系统能够生成报表，保留记录，生成票据，进行文件备份。

2.2.2 系统开发的可行性

(1) 技术可行性：

① 从硬件及开发环境考虑，本软件基于 Windows10 系统，采用了 python 的网络框架 Django：可以快速开发安全和可维护的网站。Django 具有完备性、通用性、安全性、可扩展、可维护性和灵活性的特点。Django 开发简便，界面美观，安全性高，很适合开发一款实用性高的酒店管理系统。

② 后台数据库来说，本软件采用了 MySQL 数据库。MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。MySQL 是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。

MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。因此，本小组决定使用 MySQL 数据库。

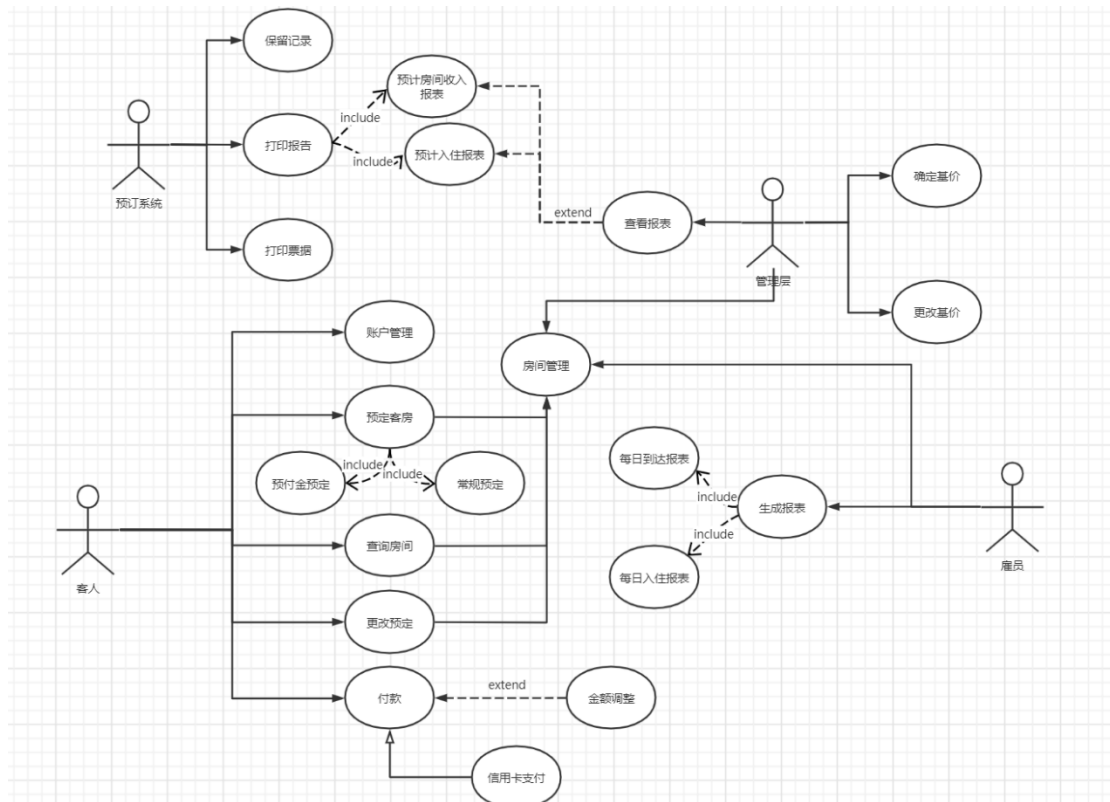
(2) 经济方面可行性：

本系统开发且投入使用后，可以减少工作人员人数，减少了工资发放，提高了工作效率，在一定程度上能够增加酒店的收益。开发起来并不是非常困难，只需对系统进行开发、维护。因此，在经济方面是可行的。

三、系统分析

3.1 用例描述

3.1.1 整体用例图



3.1.2 客人

(1) 账户管理

用例名:	账户管理
主要参与者	客人
描述	客人管理个人账户
前置条件	客人正常进入个人账户
后置条件	若客人更改个人信息，系统更新客人账户信息
主事件流	1.客人进入酒店系统

	2.进入个人账户管理系统 3.更改个人账户信息或进行其他操作 4.系统更新客人的相关操作 5.客人退出账户管理 6.用例结束
扩展流	
补充说明	更改账户信息过程受系统稳定性影响

(2) 预定客房

用例名：	预定客房
主要参与者	客人、预订系统、雇员
描述	客人预订房间，预订后雇员确认相关信息
前置条件	客人正常进入系统，且有可预订的房间
后置条件	如果预订成功，预订系统保存本次预订信息，并更新相关房间的状态
主事件流	1.客人进入预订系统 2.查询预订系统中是否有可预订的房间 3.选择两种预订方式（预付金预订 or 常规预订） 4.付款 5.预订系统保留此次的信息更改 6.雇员对预订信息进行确认 7.用例结束
扩展流	1. 重复预定 如果存在相同的预定(同样的名字、e-mail、入住实践、离开时间)则系统显示原有的预定，并询问客户是否进行新的预定
补充说明	两种预订方式：

	<p>(1) 预付金预订。</p> <p>客人在预订时将所有花费一次性付清。预付价格是基价的75%。</p> <p>(2) 常规预订。</p> <p>对于常规预订，客人在停留的最后一天付款。</p>
--	---

(3) 查询房间

用例名：	查询房间
主要参与者	客人
描述	客人查询房间信息
前置条件	客人正常进入系统
后置条件	若管理层更改房间，系统更新房间信息
主事件流	<p>1.客人进入系统</p> <p>2.查询对应日期的房间信息</p> <p>3.系统保存此次查询记录</p> <p>4.客人退出系统</p> <p>4.用例结束</p>
扩展流	
补充说明	

(4) 更改预定

用例名：	更改预定
主要参与者	客人、预订系统、雇员
描述	客人更改房间预订，预订系统保留更改信息，雇员对更改后的预订信息进行确认(更改预订包括更改房间预订类型或取消预订，客人需要根据自己的需求来选择)
前置条件	客人已经成功预订房间，并且酒店可以提供其他房间
后置条件	如果更改预订成功，系统保存本次的更改信息，更新相关

	房间的状态
主事件流	<p>客人进入预订系统</p> <p>选择要更改预订房间类型或是取消预订</p> <p>若选择更改预订房间类型：</p> <ol style="list-style-type: none"> （1）查询预订系统中是否有其他可预订的房间 （2）更改房间 （3）系统查询客户的预订类型，确定更改费用 （4）客人付款 （5）预订系统保留此次更改信息 （6）雇员对更改信息进行确认 （7）修改成功 <p>4.若选择取消预订：</p> <ol style="list-style-type: none"> （1）取消订单 （2）系统查询预订方式，确定取消费用 （3）系统退款 （4）预订系统保存本次的取消信息，更新房间状态 （5）雇员进行信息确认 （6）取消成功 <p>5.用例结束</p>
扩展流	取消预定
补充说明	<p>预付金预订。</p> <p>客人更改预订房间类型，此时将根据预订类型，进行金额的调整，如果金额减少了，将不退款。如果金额增加了，增加金额的金额将在更改时付款。</p> <p>如果取消预付金预订，将不退款。</p> <p>常规预订。</p> <ol style="list-style-type: none"> （1）对于常规预订，客人在停留的最后一天付款，随时可以更改（需得到确认）。 （2）入住前 3 天内取消预订，需要支付第一天的房费。

(5) 付款

用例名：	付款
主要参与者	客人、预订系统
描述	客人支付订单相关费用，预订系统保留并更新相关信息
前置条件	客人完成预订房间，更改房间或取消订单等需要支付相关费用的操作
后置条件	支付成功后，系统更新客人的订单状态，预订系统保留付款记录
主事件流	1.客人进入付款阶段 2.信用卡付款 3.预订系统保留信息 4.用例结束
扩展流	信用卡支付
补充说明	

3.1.3 雇员

(1) 房间管理

用例名：	房间管理
主要参与者	雇员
描述	雇员对房间进行管理
前置条件	系统正常运行
后置条件	雇员成功确认客人每次的更改预订或客人的预订信息
主事件流	1、雇员成功登入系统 2、有新的预订信息或有客人更改预定 3、雇员进行信息确认
扩展流	a.系统在任意时刻失败：（雇员无法登录系统或确认失败）

	1、雇员重启系统 2、登录
补充说明	

(2) 生成报表

用例名：	生成报表
主要参与者	雇员
描述	雇员根据每日的预订信息生成每日到达报表和每日入住报表
前置条件	雇员正常进入系统且当日有达到客人及正在入住客人
后置条件	雇员成功生成两种报表
主事件流	1、雇员正常进入系统 2、系统显示预计当天到达的客人，生成每日到达报表 3、雇员根据单签还停留在宾馆的客人，生成每日入住报表
扩展流	a.系统在任意时刻失败：（雇员无法登入系统或系统无法显示数据/数据有误） 1、雇员重启系统 2、登录
补充说明	

3.1.4 预订系统

(1) 保留记录

用例名：	保留记录
主要参与者	预订系统、客人、管理层
描述	预订系统需要保留所有的更改信息，并且反映在软件里（如客人预订房间、更改预订、支付房费、系统退款，管

	理层确定及更改基价等)
前置条件	房间信息有改变
后置条件	软件将最后的数据写入磁盘
主事件流	<ol style="list-style-type: none"> 1. 有新客人进行预订或有客人更改了预订信息 2. 有管理层人员对房间基价进行了调整 3. 预订系统接受更改信息的反馈 4. 预订系统更新更改后的信息并将最后的数据写入磁盘 5. 在每个工作日的最后生成所有文件的备份拷贝，并存储在另外的地点。
扩展流	
补充说明	

(2) 打印报告

用例名:	打印报告
主要参与者	预订系统
描述	预订系统打印预计房间收入的报表和预计入住的报表
前置条件	系统正常
后置条件	预订系统成功打印报表
主事件流	<ol style="list-style-type: none"> 1. 预订系统连接到系统磁盘的数据 2. 根据数据中的房间预订的数据打印出预计入住的报表；根据数据中 30 天内预订数量与房间价格打印出预计房间收入报表
扩展流	
补充说明	

(3) 打印票据

用例名:	打印票据
主要参与者	预订系统、雇员
描述	预订系统根据客人的预订信息打印出日期、姓名、房间号、房费等
前置条件	有客人进行预订房间且预订系统正常
后置条件	成功打印票据
主事件流	<ol style="list-style-type: none">1. 预订系统连接到软件后台的客人预订信息数据2. 客人的预订是常规预订, 对信息中的预订日期, 客人姓名, 房间号, 到达日期, 离开日期, 入住天数和总金额进行打印; 客人的预订是预付金预订, 对信息中的预订日期, 客人姓名, 房间号, 到达日期, 离开日期, 入住天数和总金额, 提前支付的日期和金额进行打印3. 雇员在客人结帐时把票据交给客人。
扩展流	
补充说明	

3.1.5 管理层

(1) 查看报表

用例名:	查看报表
主要参与者	管理层
描述	管理层进入系统, 查看所需的三种报表
前置条件	预订系统生成预计入住和预计房间收入的报表, 雇员生成的每日到达的和每日入住的报表。

后置条件	管理层了解报表，方便更改基价
主事件流	1、系统正常运行 2、管理层能成功登录系统 3、预订系统打印好报表 4、管理层查看报表
扩展流	
补充说明	

(2) 确定基价

用例名：	确定基价
主要参与者	管理层、预订系统
描述	管理层确定基价，预订系统保留基价信息。
前置条件	管理层进入系统
后置条件	确定基价后可进行预付金预订，预付价格是基价的 75%。
主事件流	1、系统正常运行 2、管理层能成功登录系统 3、管理层确定不同种房间的基价 4、预定系统保留相关信息 5、系统显示房间基价 6、用例结束
扩展流	
补充说明	管理层至少提前一年就确定了所有 45 个房间的基价

(3) 房间管理

用例名:	房间管理
主要参与者	管理层
描述	管理层管理房间的现有状态，为客人预订分配房间
前置条件	房间充足
后置条件	确定房间类型，便于预订系统显示可预订房间类型，以及雇员为当日到店的客人分配相应类型的房间号。
主事件流	<ol style="list-style-type: none">1. 管理层进入系统2. 查询供酒店预订的房间3. 各类型房间状态更新4. 用例结束
扩展流	可用房间已满，无法继续分配供应客人居住的房间
补充说明	房间管理包括客人，酒店工作人员的房间管理。

(4) 更改基价

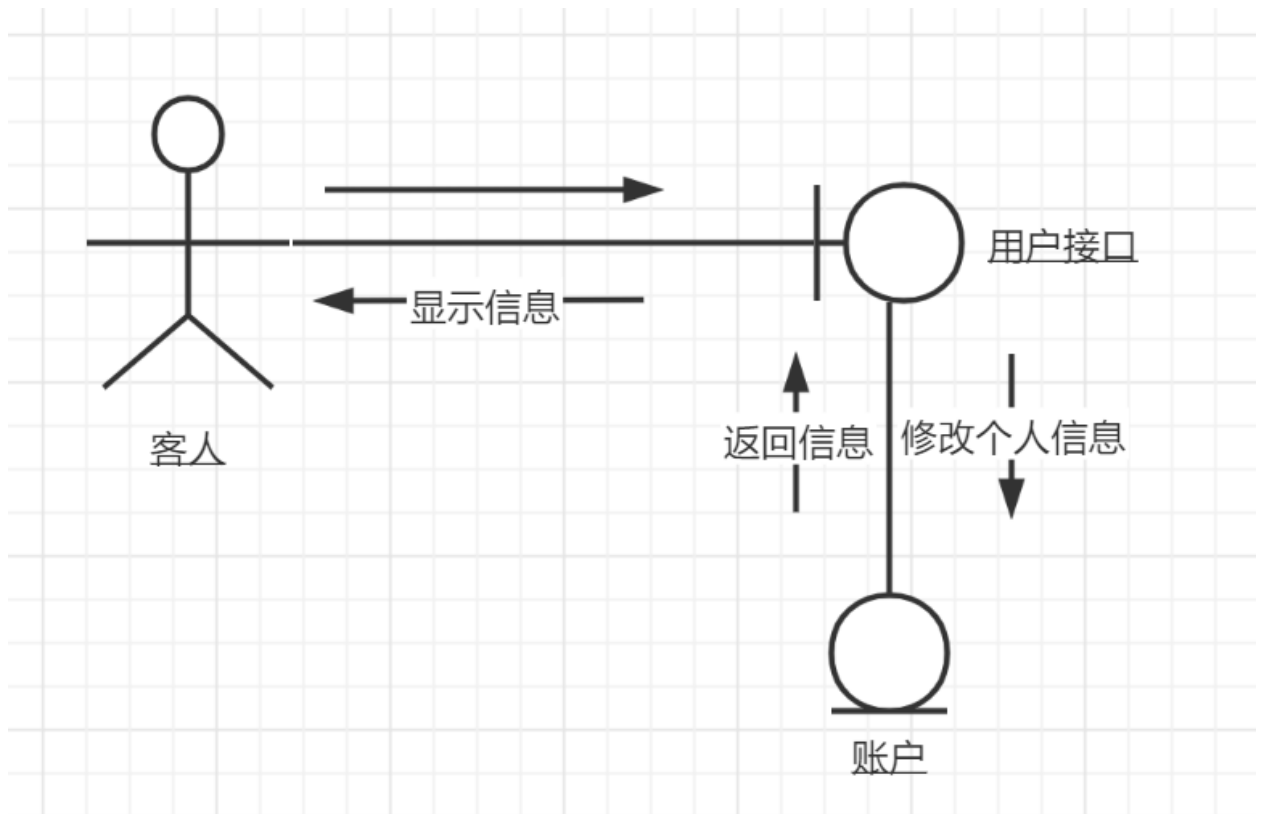
用例名:	更改基价
主要参与者	管理层、预订系统
描述	管理层更改基价
前置条件	管理员已经确定过基价
后置条件	更改基价之后的预订价格相应更改
主事件流	<ol style="list-style-type: none">1. 管理层进入系统2. 更改基价3. 预订系统保留此次更改信息4. 系统更新房间基价5. 用例结束

扩展流	管理层可以在一天的任何时候更改基价
补充说明	基价的更改不影响任何现有的预订。

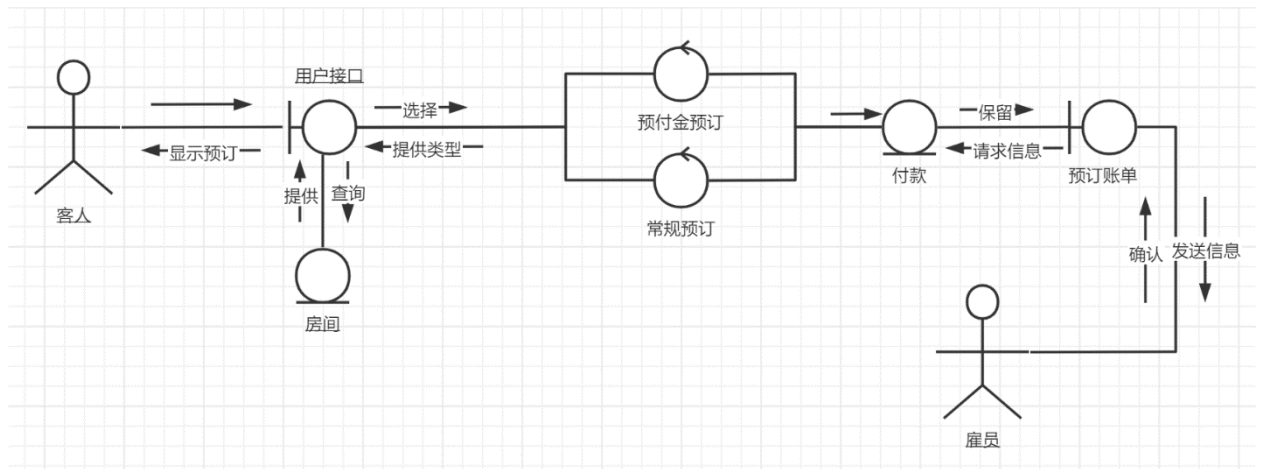
3.2 类图

3.2.1 客人

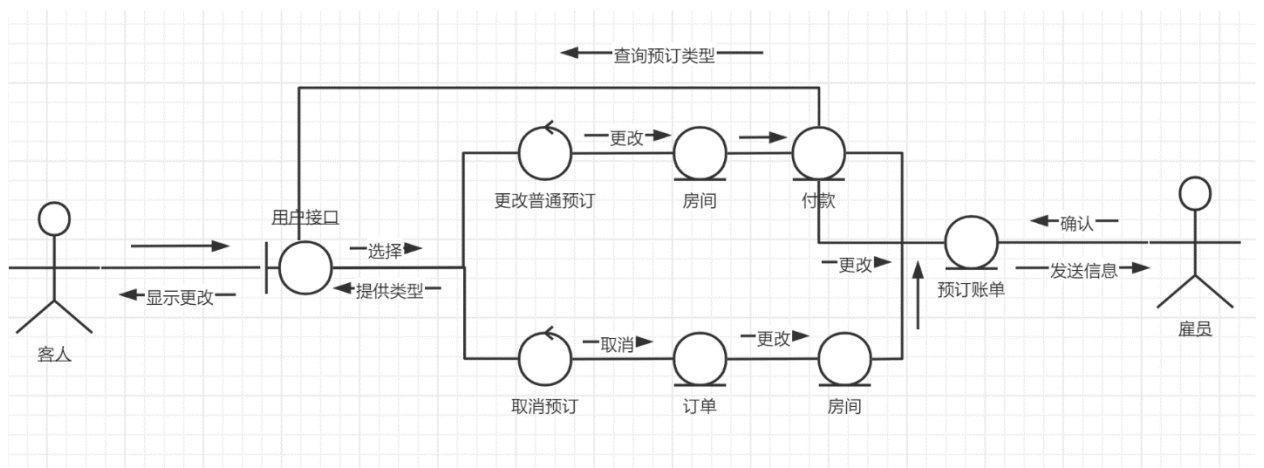
(1) 更改账户信息



(2) 预订客房

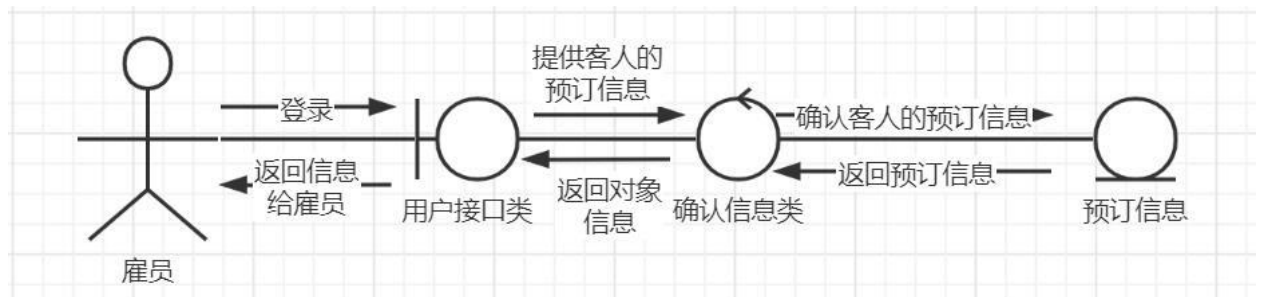


(3) 更改预订

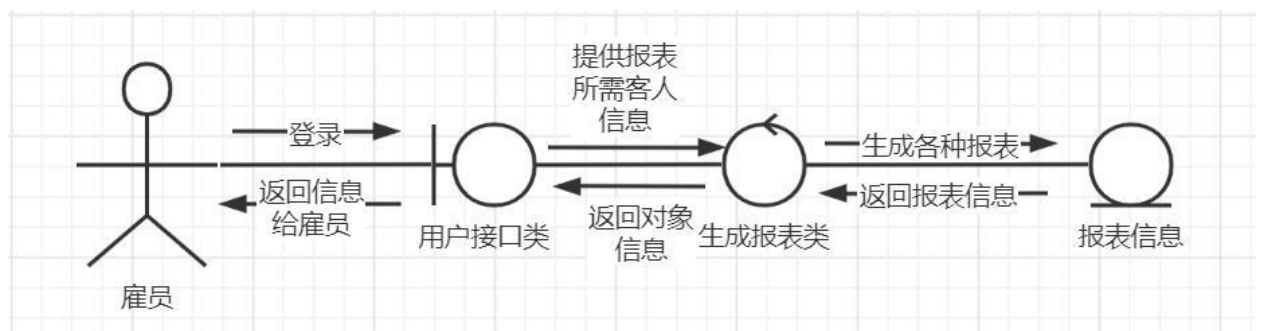


3.2.2 雇员

(1) 确认预订信息

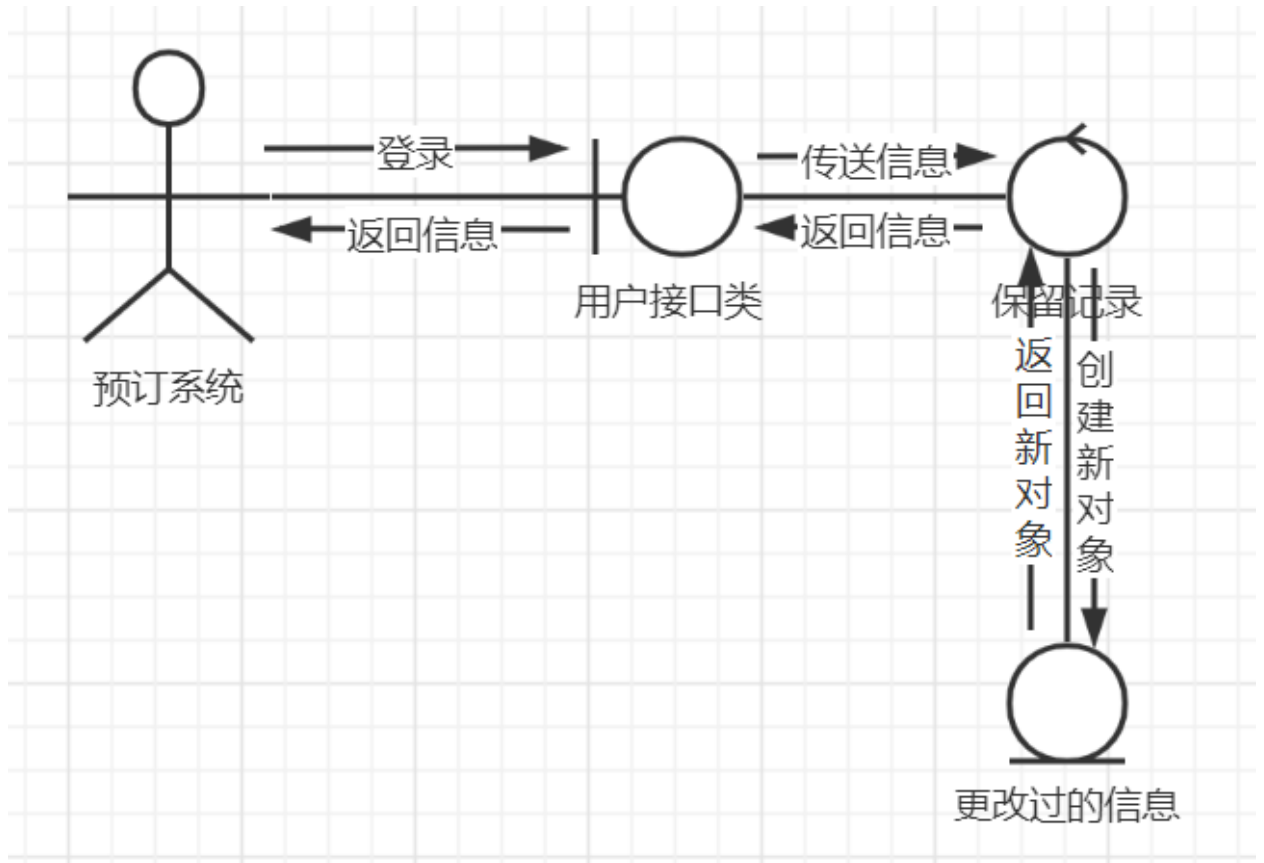


(2) 生成报表

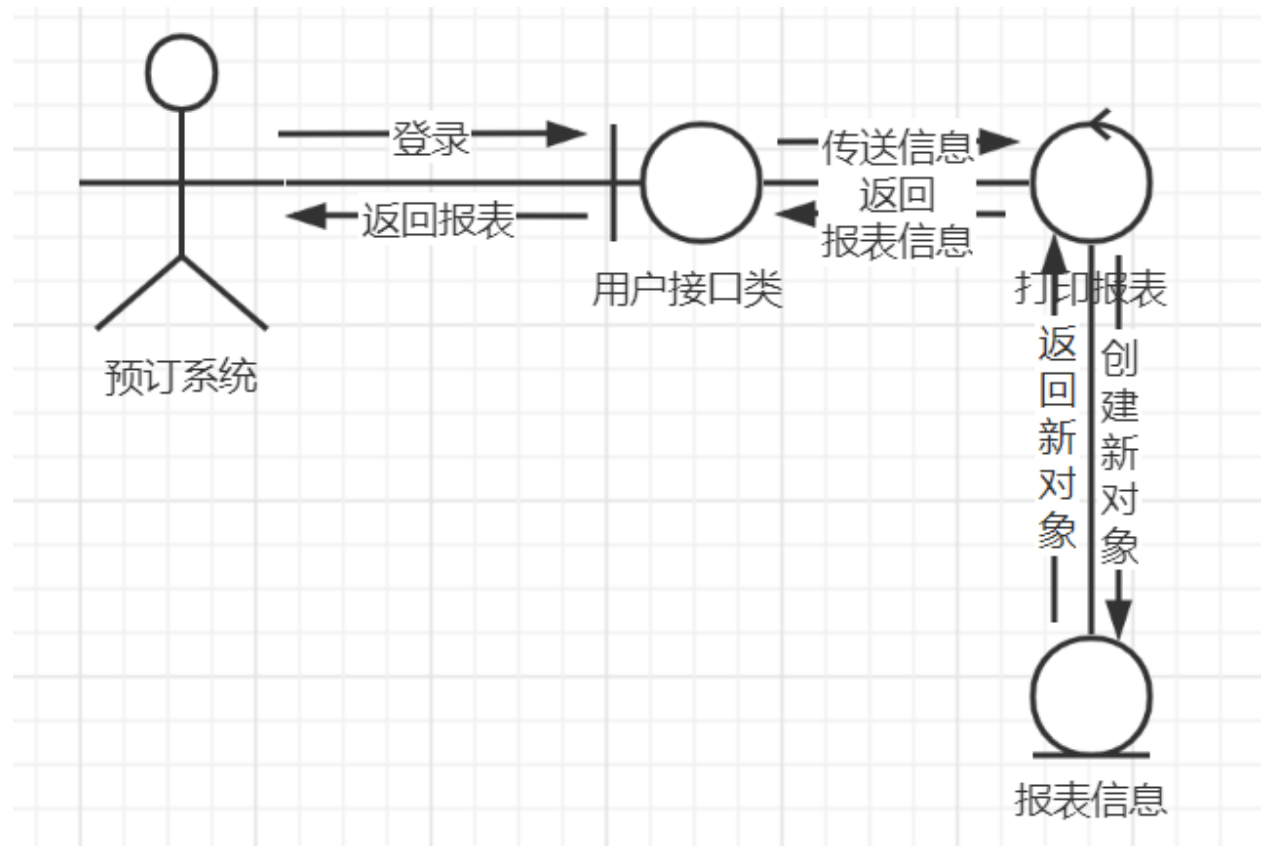


3.2.3 预订系统

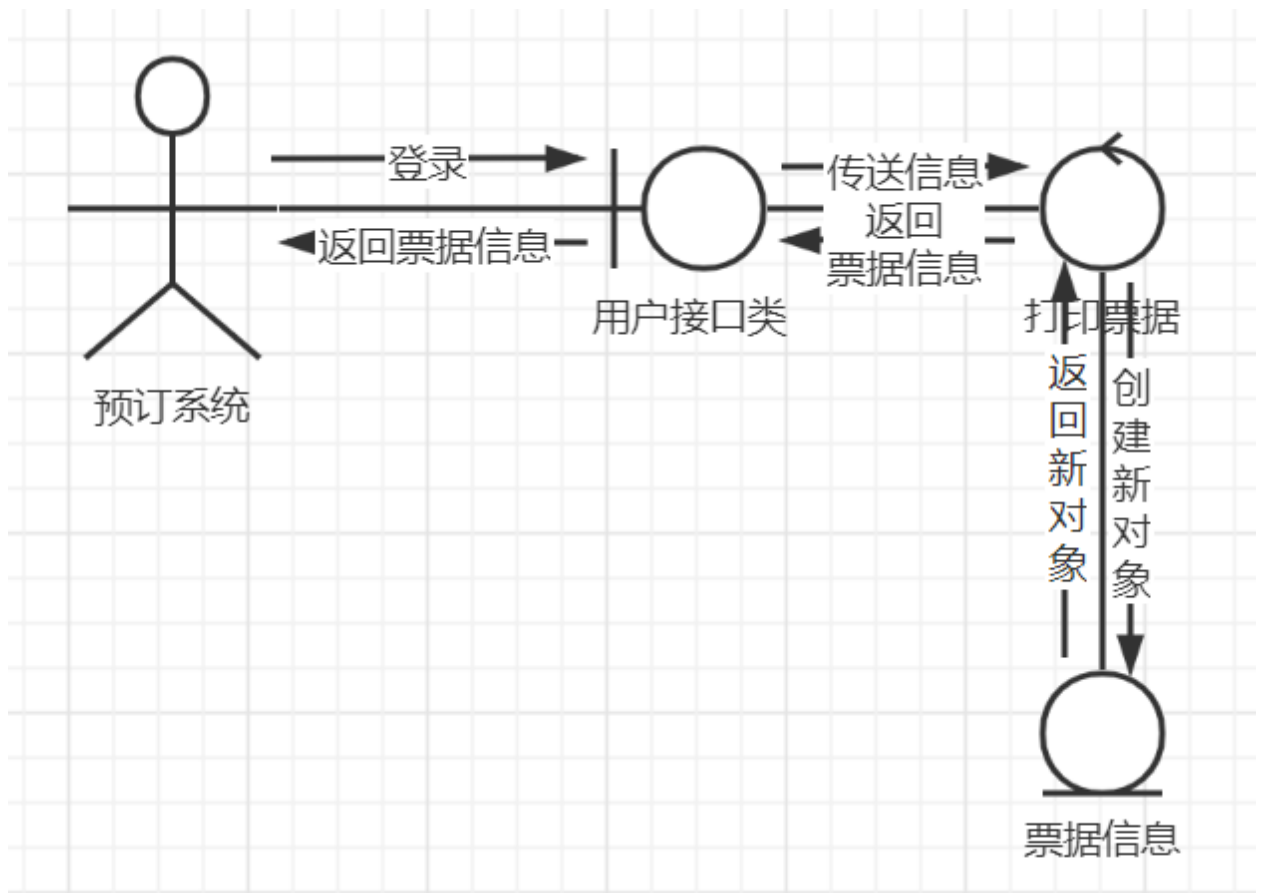
(1) 保留记录



(2) 打印报表

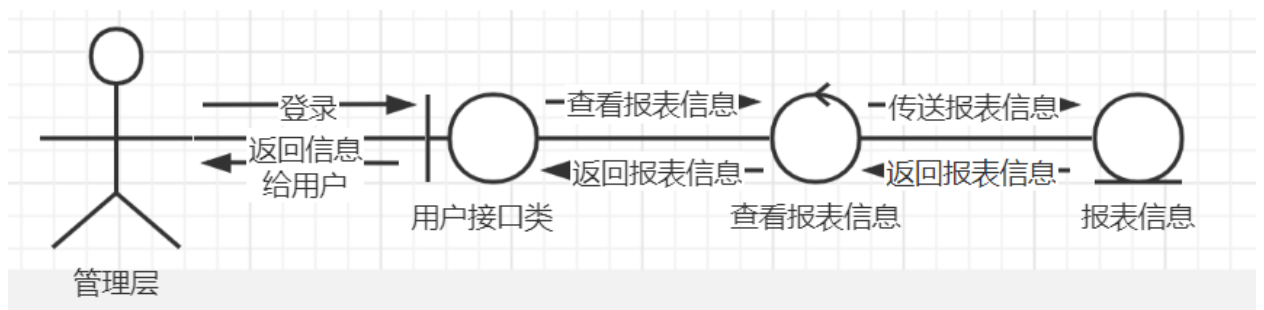


(3) 打印票据

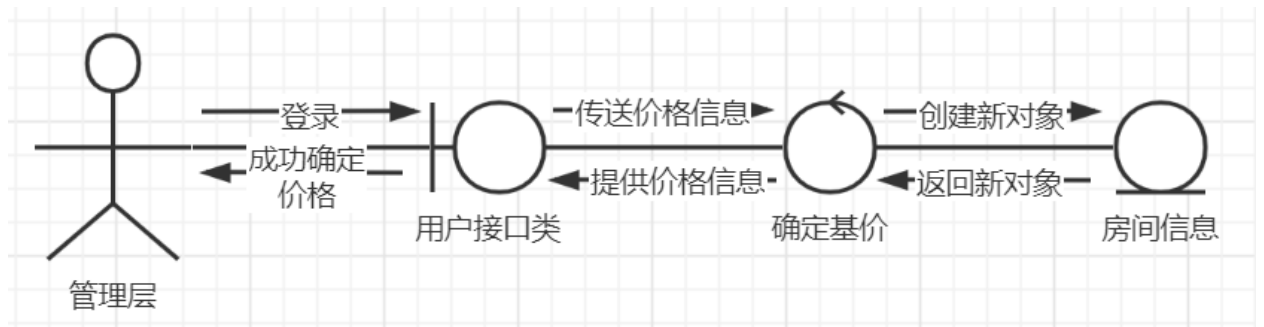


3.2.4 管理层

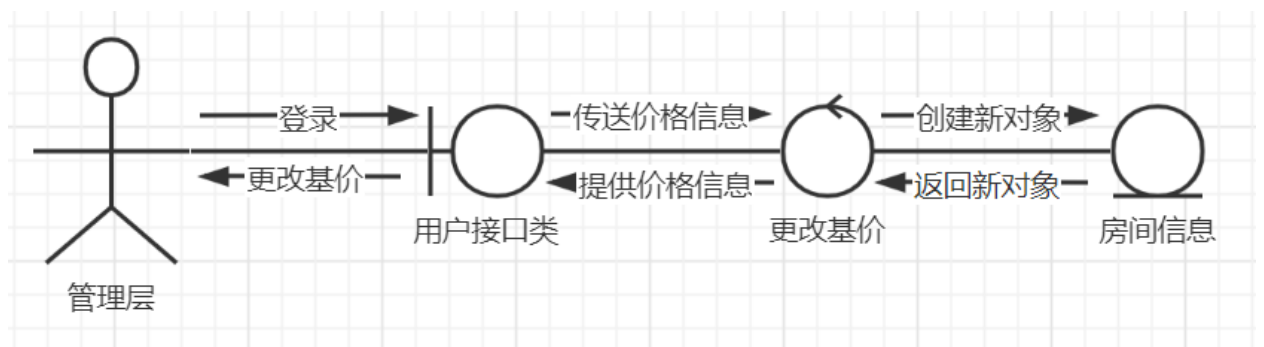
(1) 查看报表



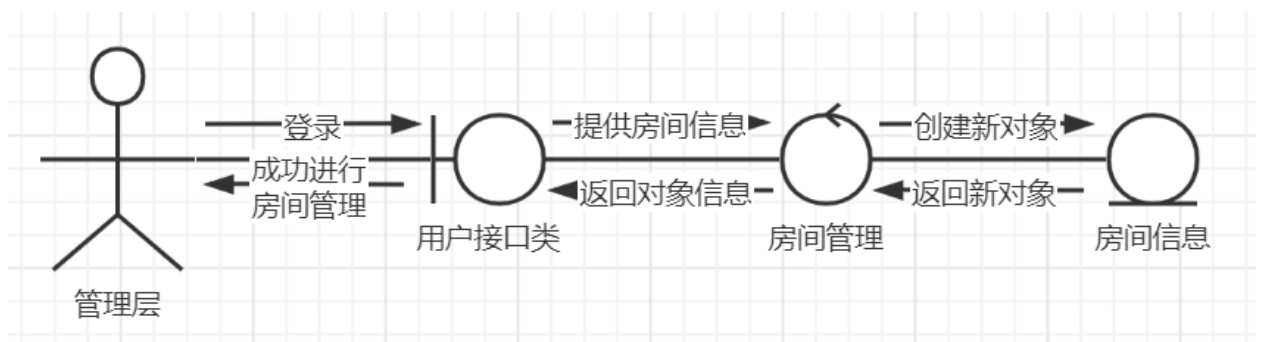
(2) 确定基价



(3) 更改基价



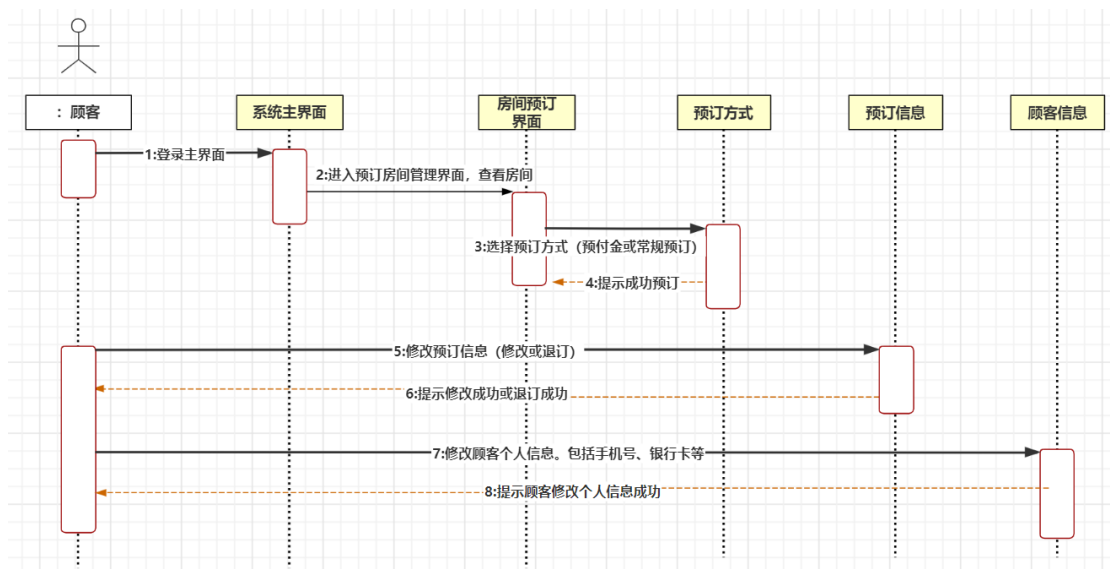
(4) 房间管理



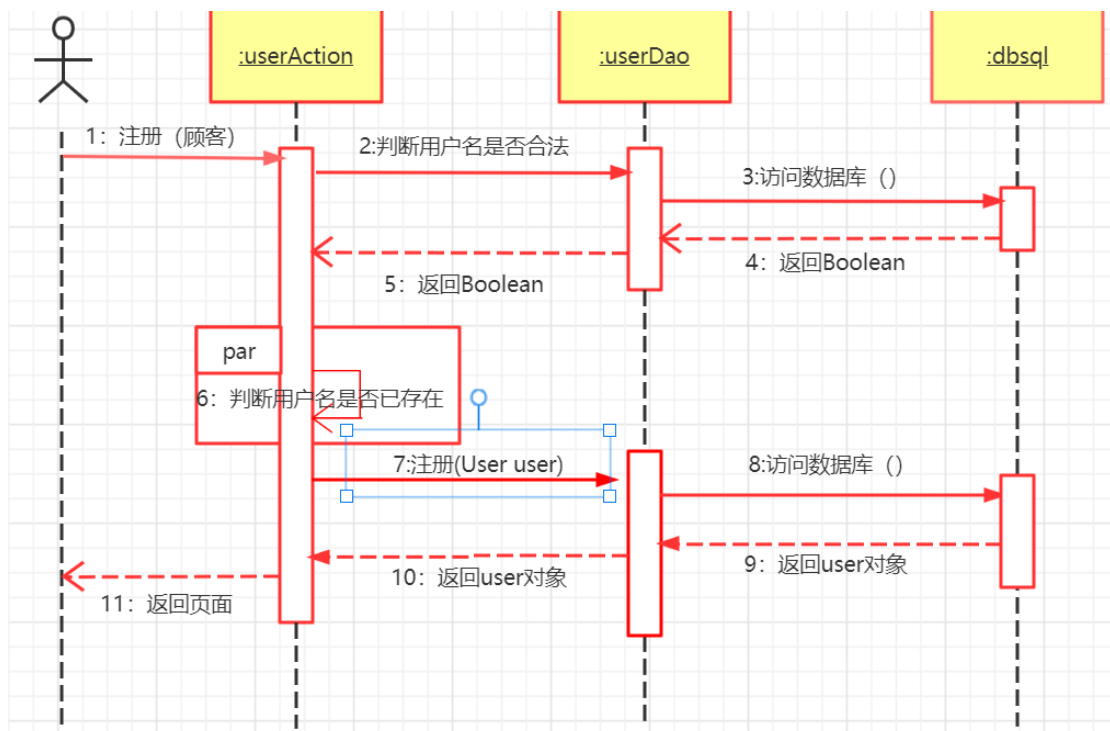
3.3 序列图

3.3.1 客人

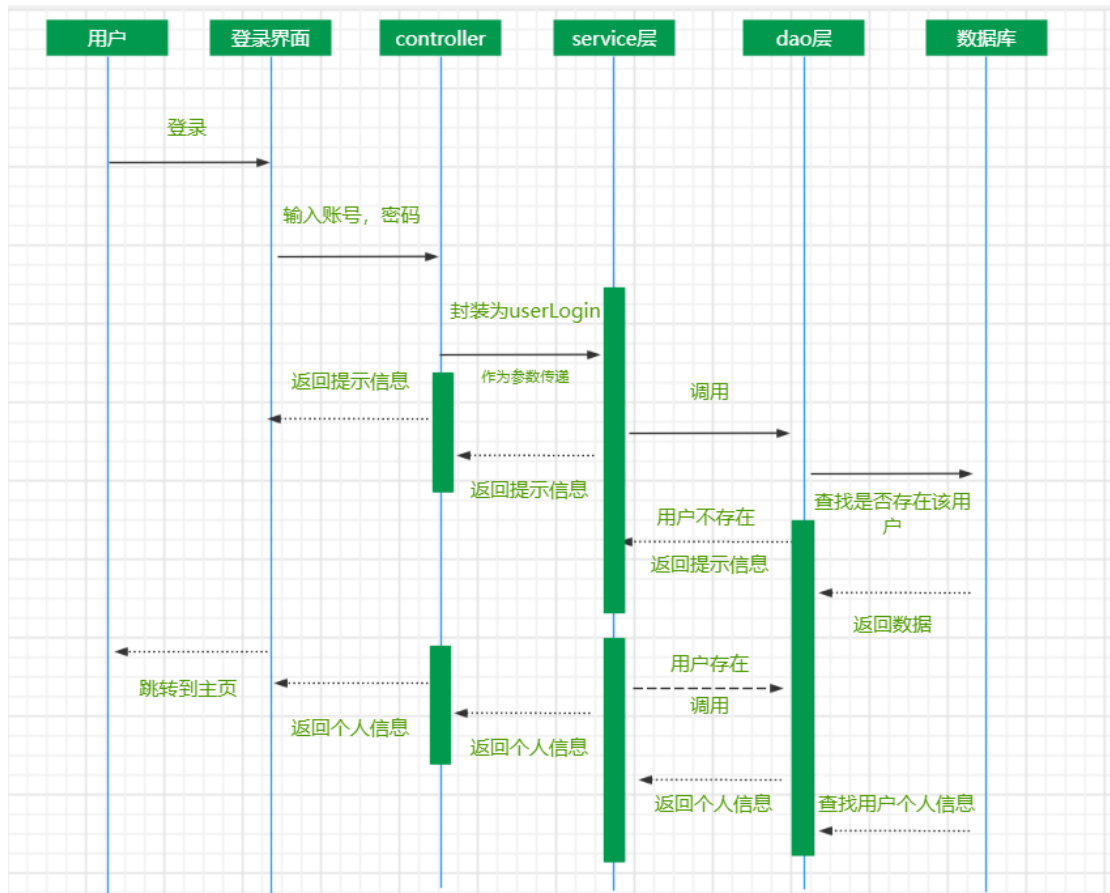
(1) 总体时序图：



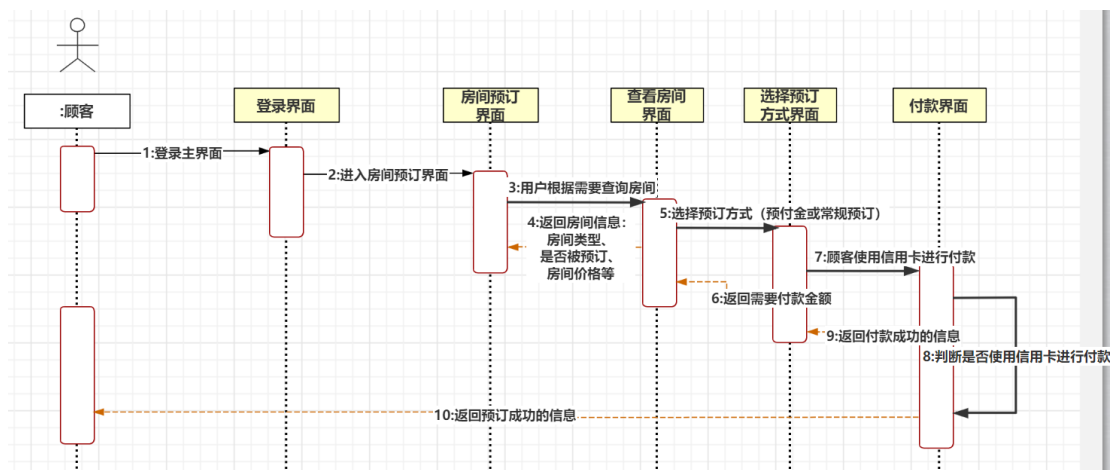
(2) 顾客注册时序图：



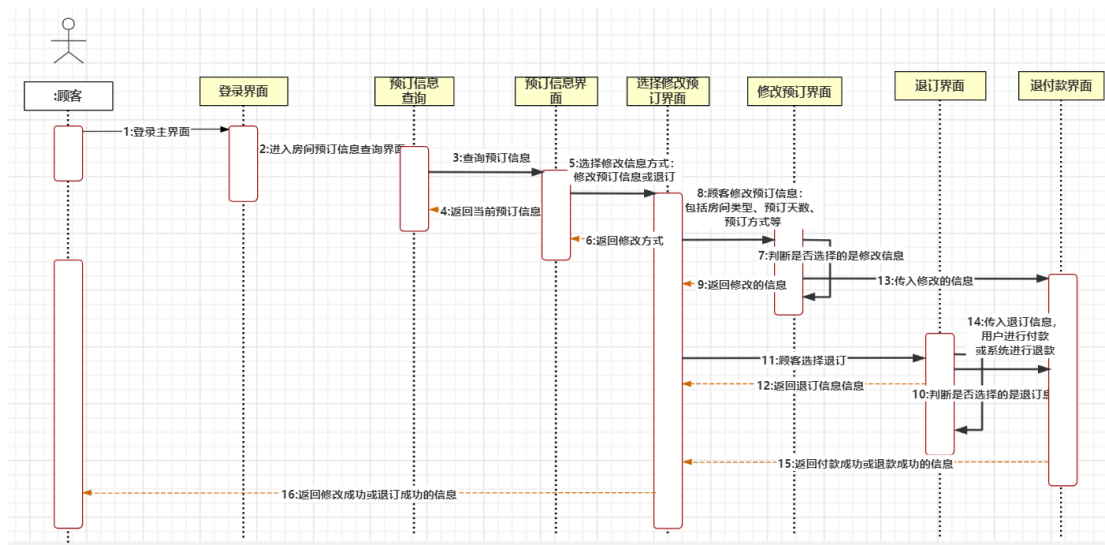
(3) 顾客登录时序图:



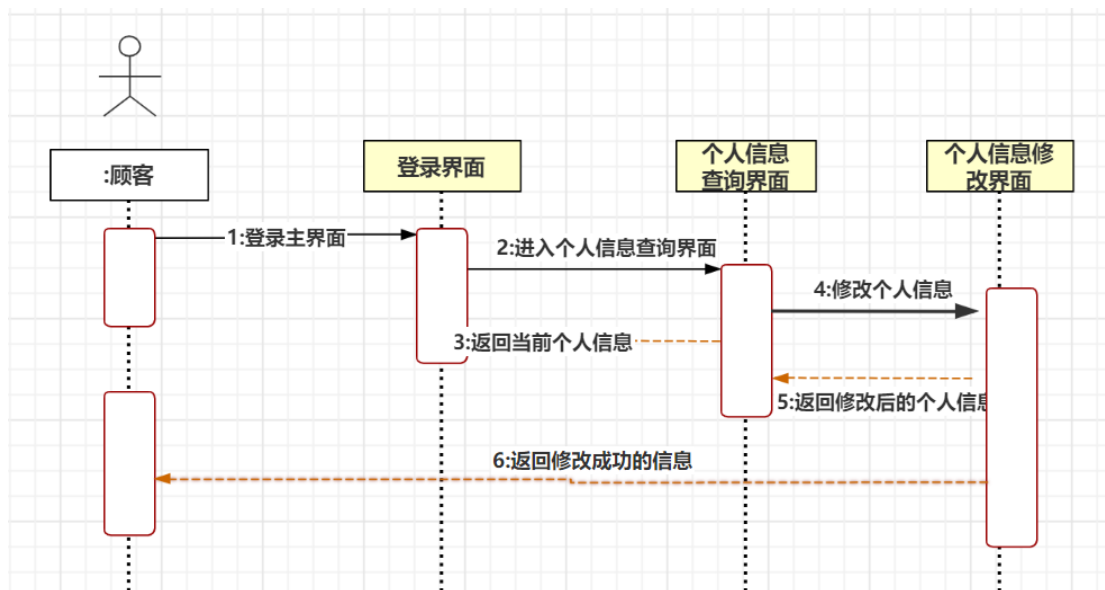
(3) 顾客预订房间时序图:



(4) 顾客修改预订信息时序图：

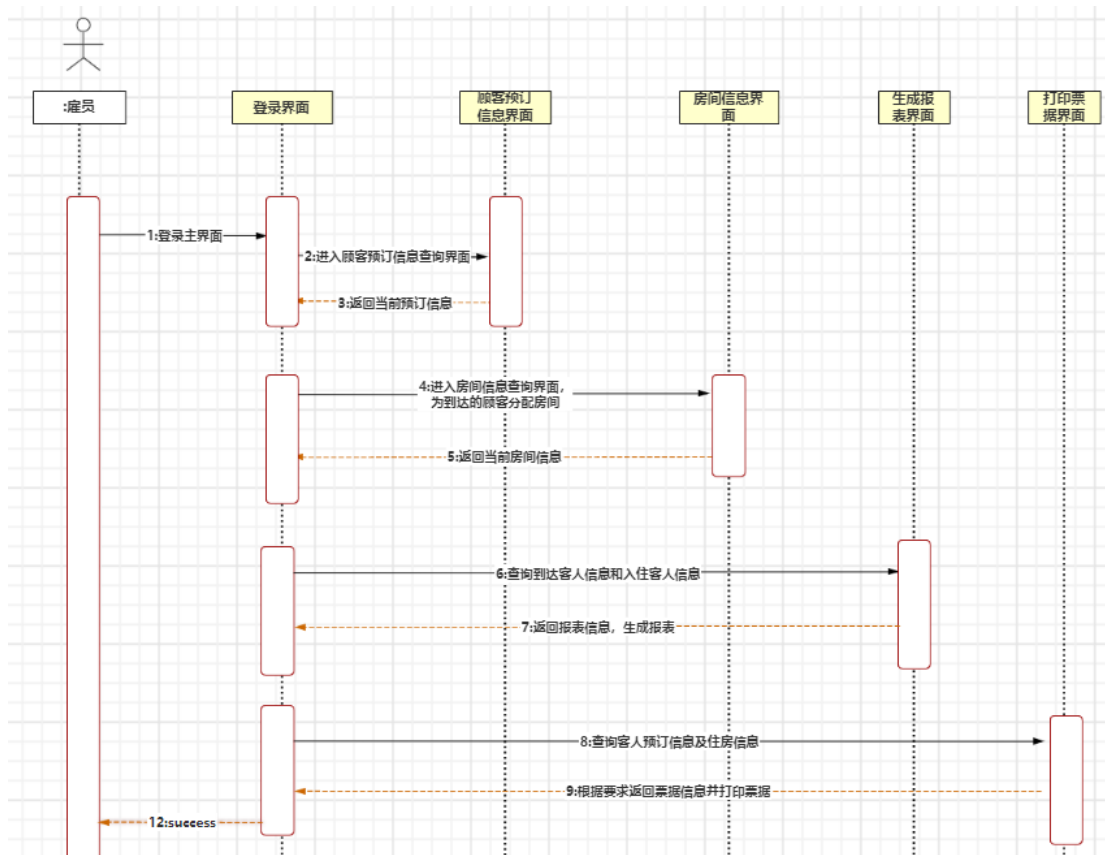


(5) 顾客修改个人信息时序图：

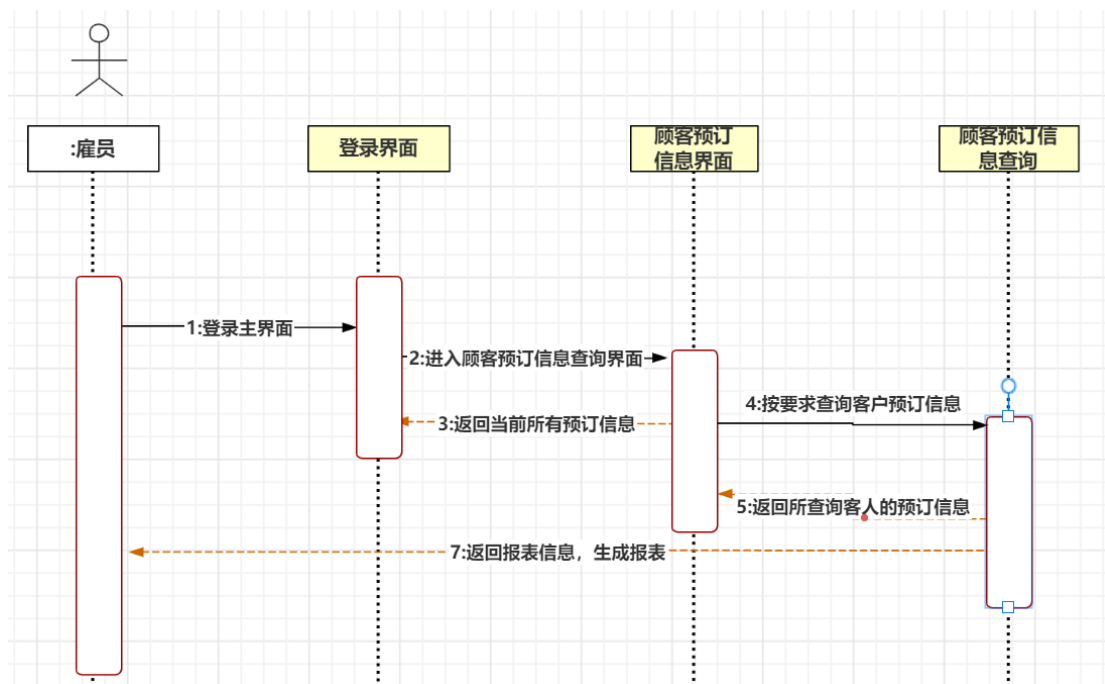


3.3.2 雇员

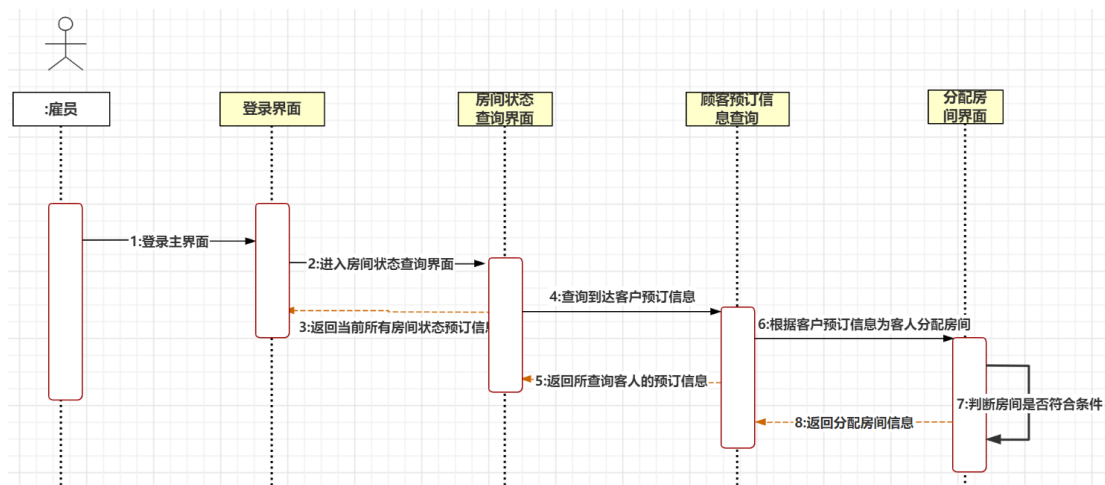
(1) 总体时序图：



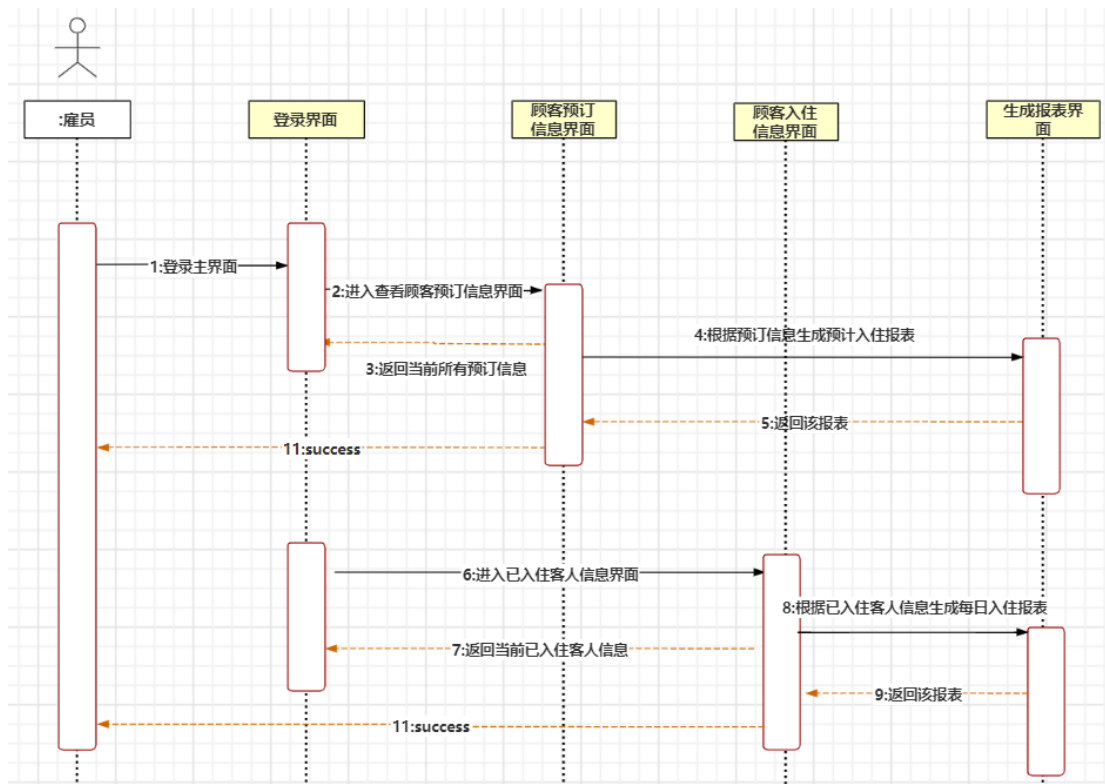
(2) 查询顾客信息时序图：



(3) 为到达顾客分配房间时序图：

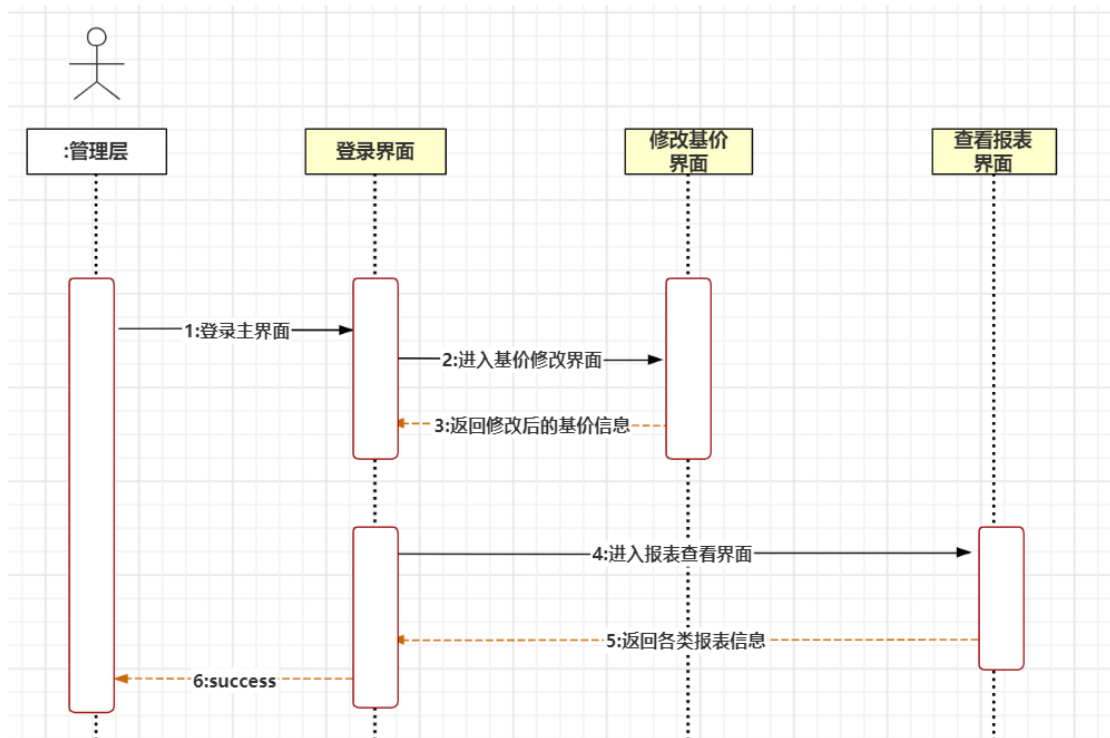


(4) 生成报表时序图：

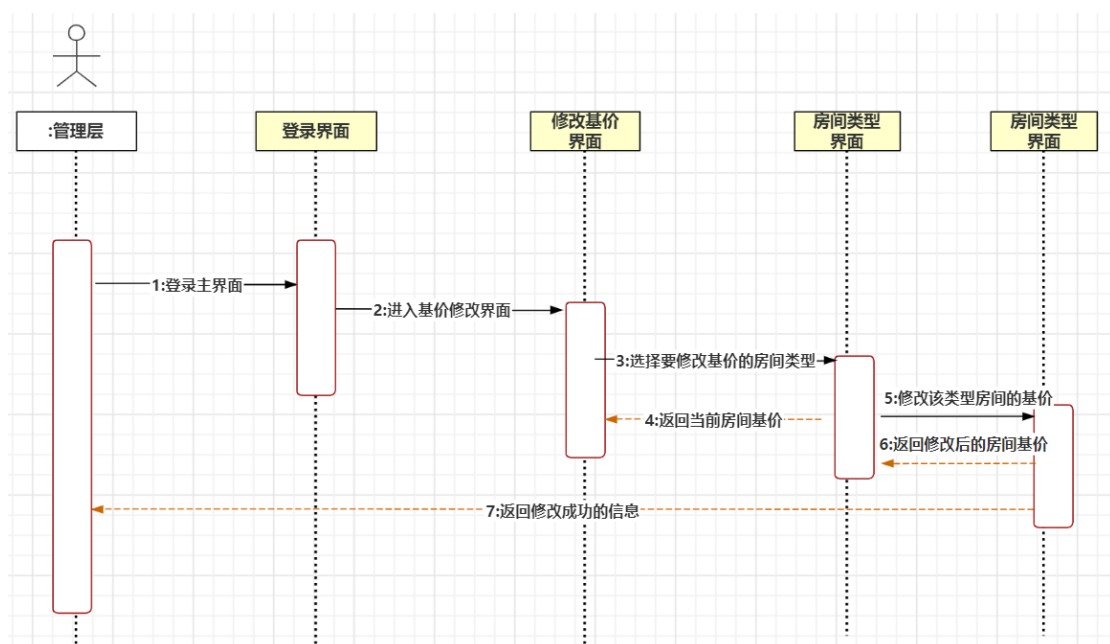


3.3.3 管理层

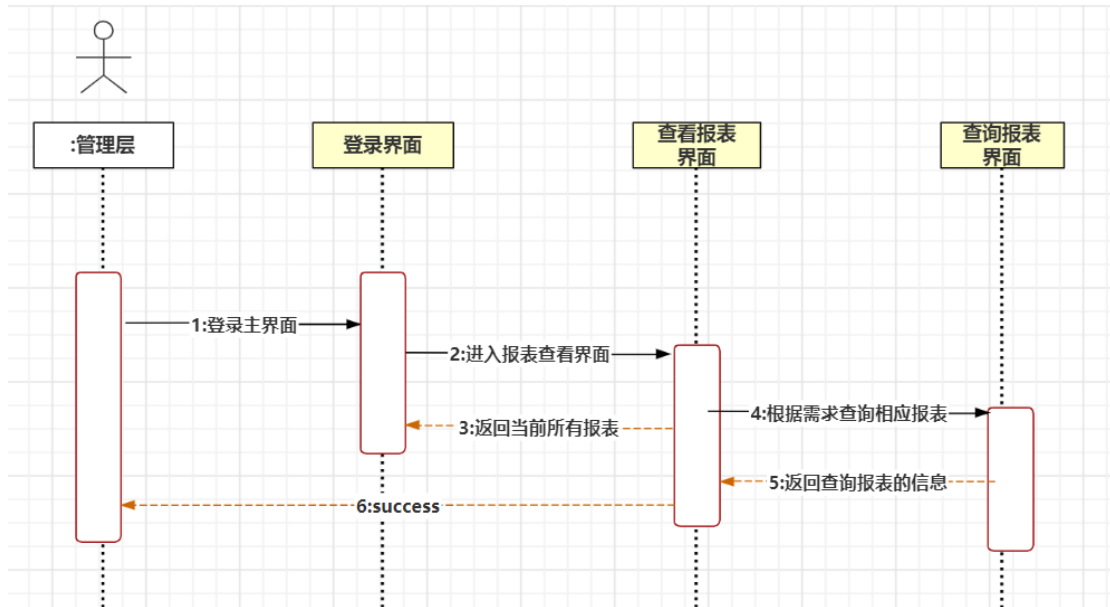
(1) 总体时序图：



(2) 修改基价时序图：

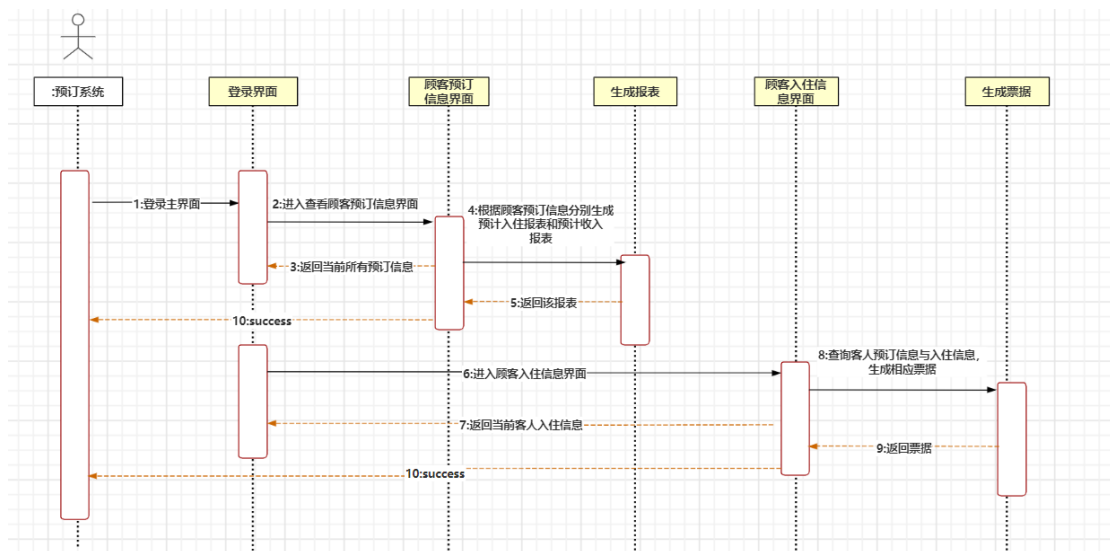


(3) 查看报表时序图:

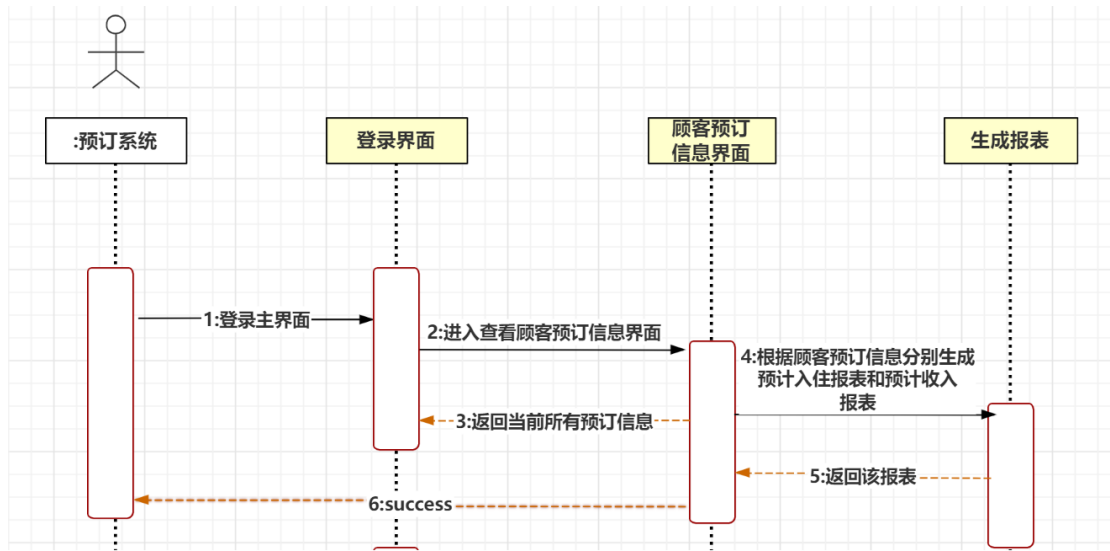


3.3.4 预订系统

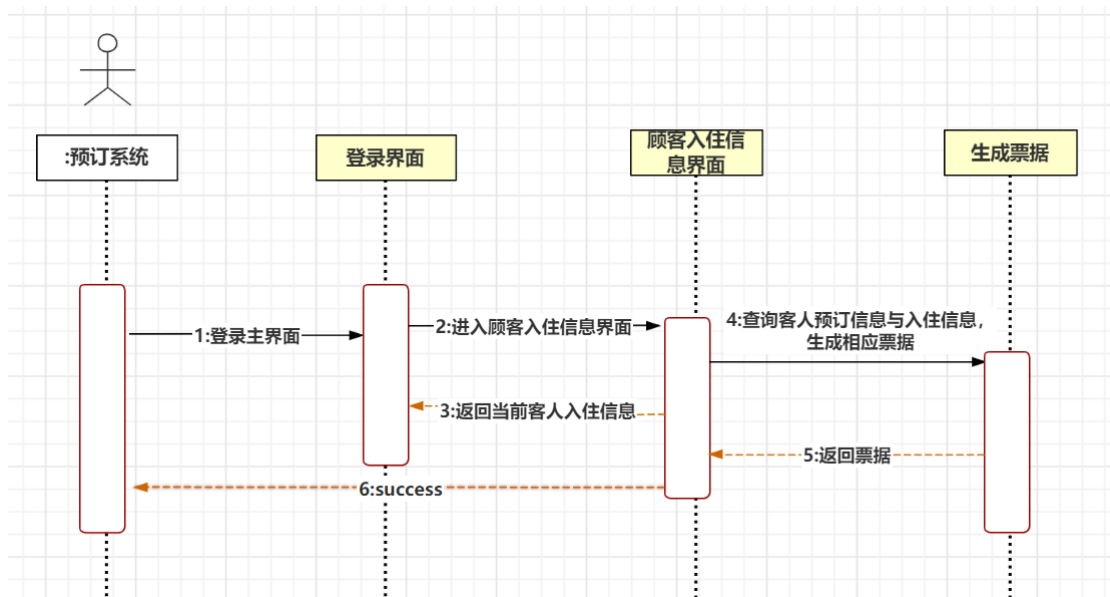
(1) 总体时序图:



(2) 生成报表时序图:



(3) 生成票据时序图:



四、系统设计

4.1 总体设计

4.1.1 运行环境

本项目基于 Windows10 系统，用 python 的 Django 网络架构和 MySQL 数据库共同开发。Python 版本：3.X；MySQL 版本：8.X。

4.1.2 系统组成结构



4.2 数据库设计

在 Django 项目建好后，在 setting.py 中设置好 mysql 连接参数：

```
1. DATABASES = {  
2.     'default': {  
3.         'ENGINE': 'django.db.backends.mysql',  
4.         'NAME': 'book_2', # 要连接的数据库，连接前需要创建好  
5.         'USER': 'root', # 连接数据库的用户名  
6.         'PASSWORD': '123', # 连接数据库的密码  
7.         'HOST': '127.0.0.1', # 连接主机，默认本级  
8.         'PORT': 3306 # 端口 默认 3306  
9.     }  
10. }
```

在 models.py 文件中根据自己需求，填写数据表的结构，在本实验中，数据库的表设计如下：

```
1. class sign(models.Model):  
2.     sno = models.AutoField(primary_key=True)  
3.     spassword = models.CharField(max_length=15)  
4.     sbz = models.IntegerField()  
5. class rooms(models.Model):  
6.     rno = models.AutoField(primary_key=True) # 房间号  
7.     rtype = models.CharField(max_length=10) # 房间类  
           型, 大床房/ 双床房/ 商务房等...  
8.     rprice = models.IntegerField()  
9.     rin = models.BooleanField() # 是否入住  
10. class bookrecord(models.Model):  
11.     bdate = models.DateTimeField()  
12.     btype = models.IntegerField() # 预订类型  
13.     broomtype = models.CharField(max_length=10)  
           # 预订的房间类型  
14.     bdays = models.IntegerField(default=1) # 预订天数  
15.     bcno = models.ForeignKey('sign', on_delete=models.CASCADE)  
           # 预订客户编号  
16.     iscancel = models.IntegerField(default=0) # 取消记  
           录, 默认为0  
17. class checkedrecord(models.Model):
```

```

18.     crno = models.IntegerField()
19.     cdate = models.DateTimeField()
20.     ccno = models.IntegerField()
21.     cdays = models.IntegerField(default=1)
22.     cpaid = models.IntegerField(default=0)          #是否结
    账
23. class moneyrecord(models.Model):
24.     mtype = models.IntegerField()                  #收入来自3
    中类型: 1-预付金 2-常规 3-常规取消预订(3天内支付第一天)
25.     mrno = models.IntegerField()                  #收入来自哪
    个房间
26.     mrtype = models.CharField(max_length=10)
27.     money = models.IntegerField()                  #单次收入记
    录
28.     mdate = models.DateTimeField()                 #收入来自哪
    天
29.     mdays = models.IntegerField()                  #收入的天数
    (根据收入类型1、2-预订天数, 3-默认为1)

```

根据代码生成以下数据库表：（以收入记录表为例）

	id	mtype	mrno	money	mdate	mdays	mrtype
▶	1	2	103	9995	2021-05-05 00:00:00.000000	5	商务房
	2	1	102	3750	2022-08-09 00:00:00.000000	5	双床房
	3	2	102	5000	2022-08-09 00:00:00.000000	5	双床房
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL

接着在 pycharm 中的 Terminal 窗口处，输入以下代码即可：

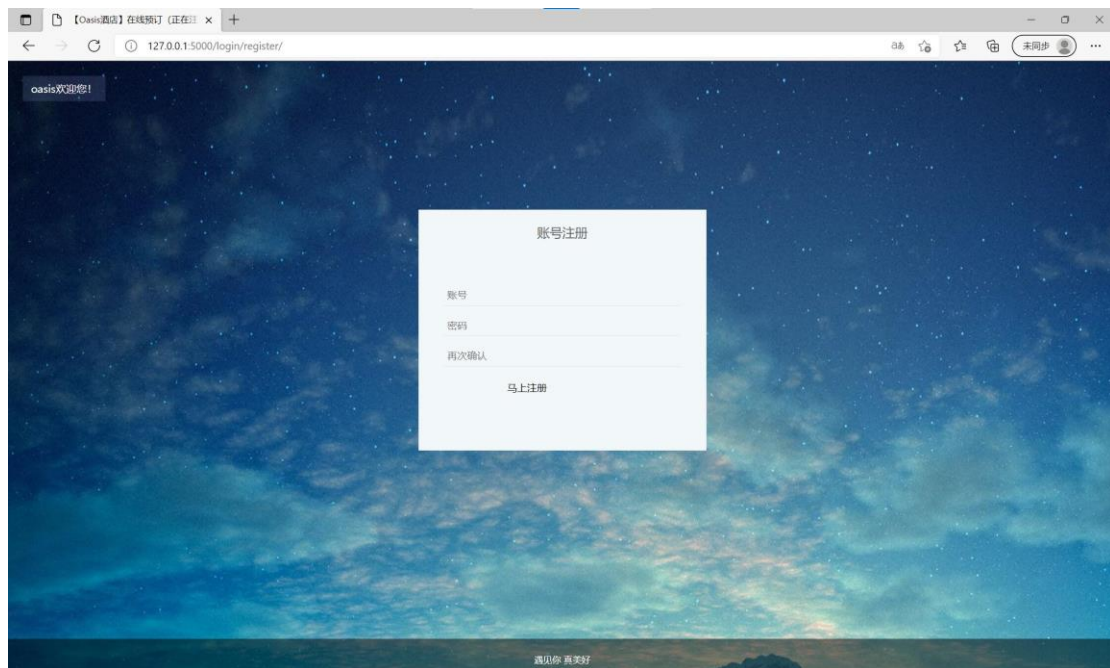
```
1. python manage.py makemigrations
```

注：Django 无法建立数据库，需要我们先建立数据库后，再用以上方法建立数据表。

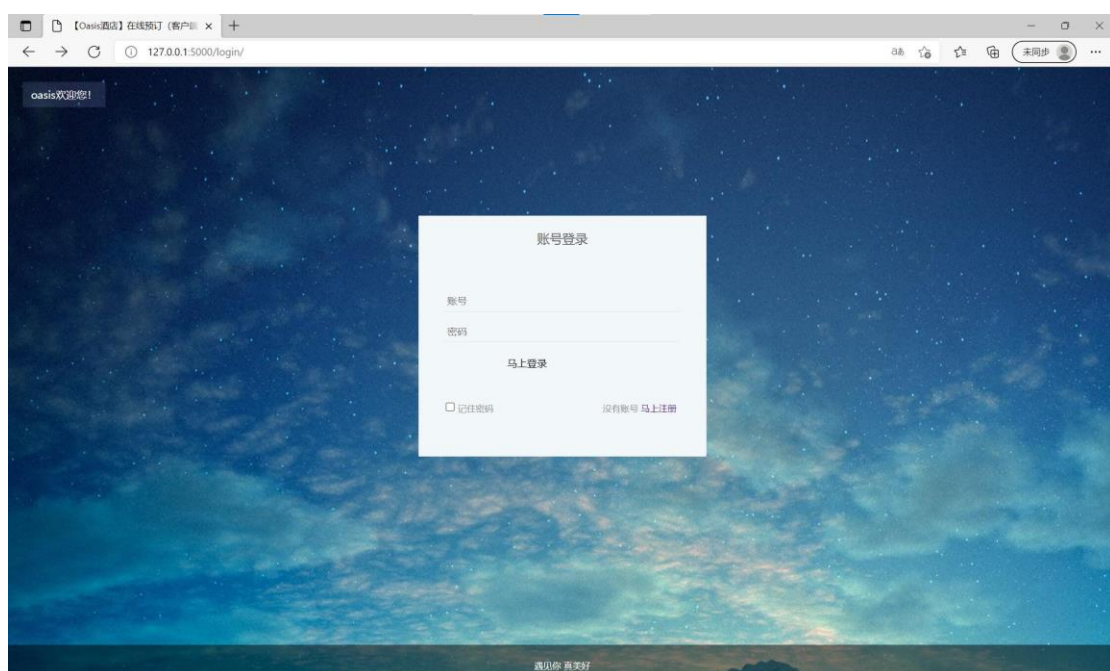
4.3 用户界面设计

4.3.1 顾客界面

(1) 注册界面



(2) 登录界面



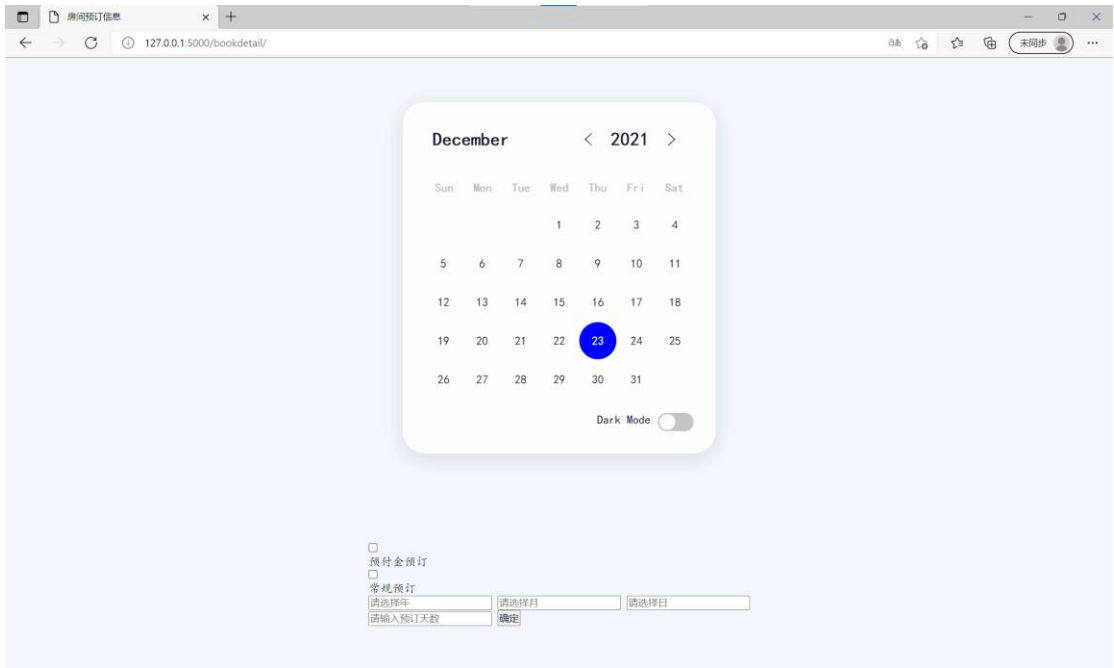
(3) 客户登录主界面



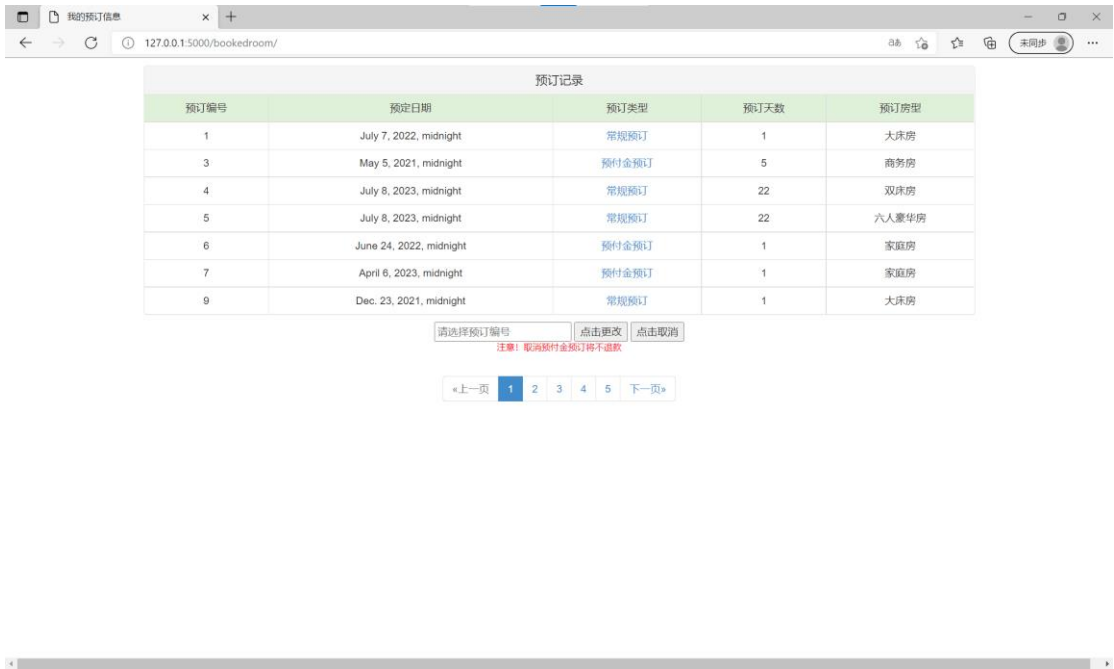
(4) 查看房间界面



(5) 预订房间界面



(6) 修改预订界面



4.3.2 雇员界面

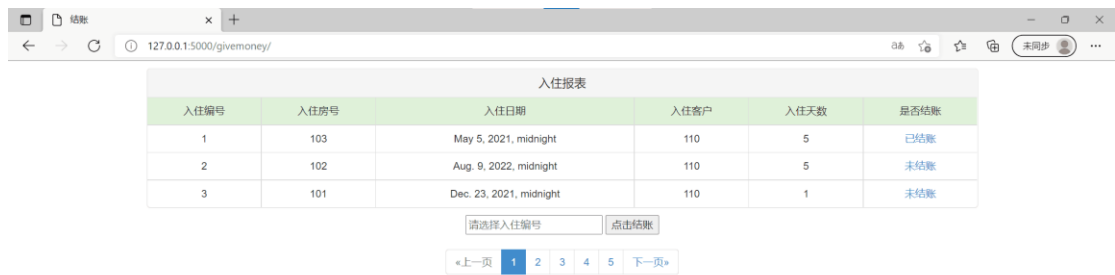
(1) 雇员主界面



(2) 雇员查看预订界面和分配房间界面



(3) 雇员结账界面



入住房号

入住编号	入住房号	入住日期	入住客户	入住天数	是否结账
1	103	May 5, 2021, midnight	110	5	已结账
2	102	Aug. 9, 2022, midnight	110	5	未结账
3	101	Dec. 23, 2021, midnight	110	1	未结账

请选择入住编号 点击结账

«上一页» 1 2 3 4 5 下一页»

(4) 雇员生成报表界面



查看报表

每日到达报表

预订编号	预订客户	预定日期	预订类型	预订天数	预订房型
9	sign object (110)	Dec. 23, 2021, midnight	常规预订	1	大床房

«上一页» 1 2 3 4 5 下一页»

打印每日到达报表 打印每日入住报表

每日入住报表

房间号	客户	入住天数
-----	----	------

«上一页» 1 2 3 4 5 下一页»

(5) 雇员打印票据界面

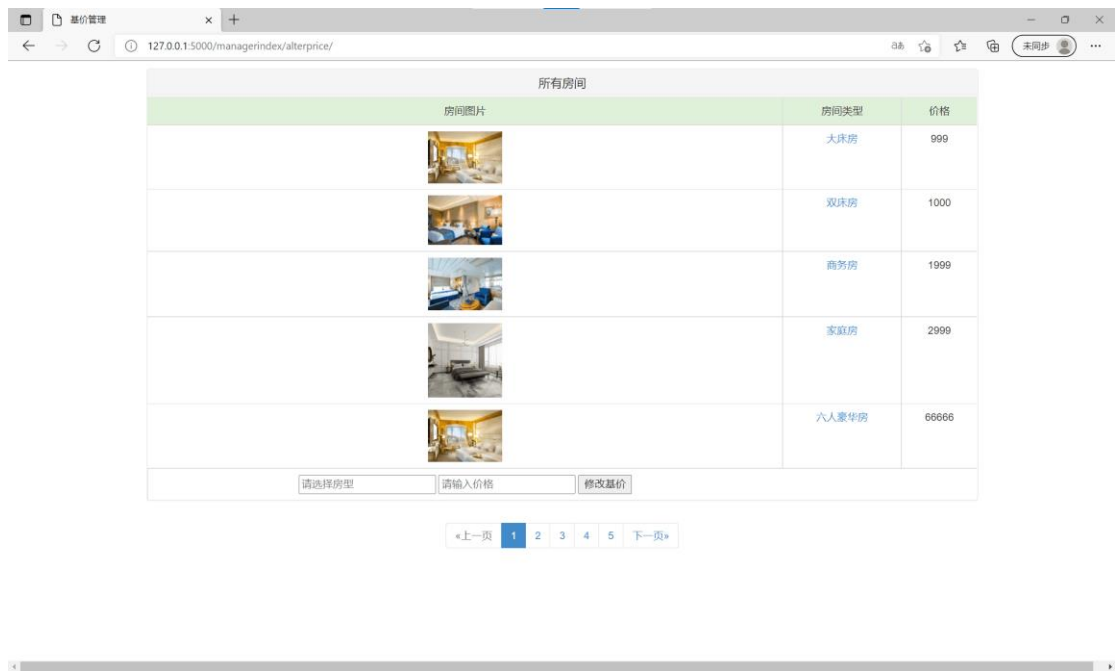


4.3.3 管理层界面

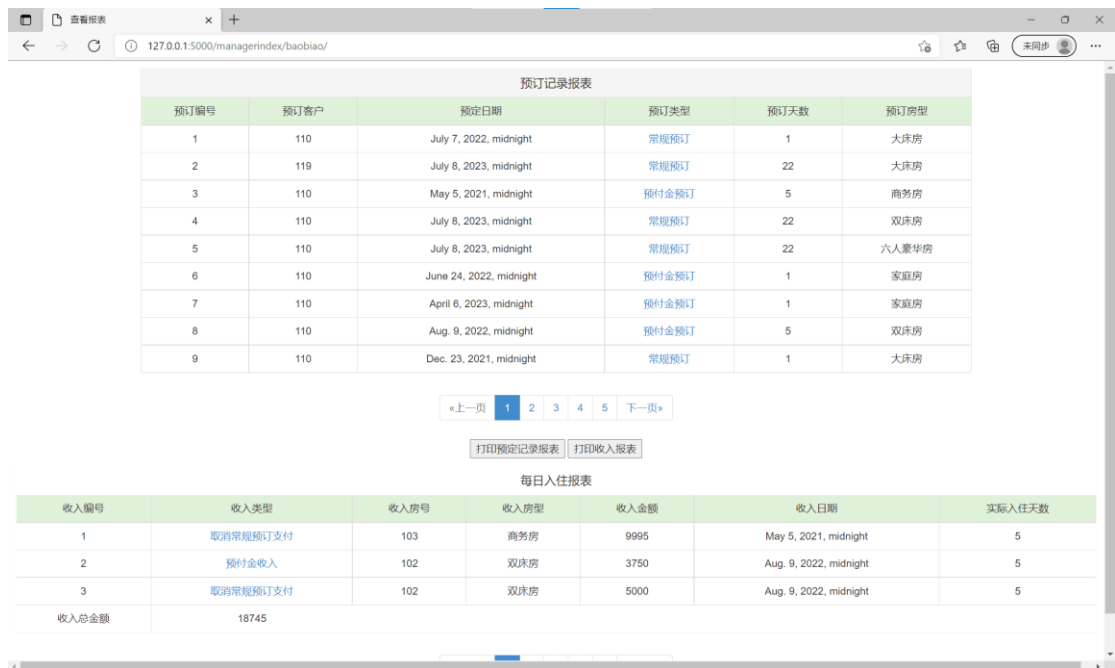
(1) 管理层主界面



(2) 修改基价界面



(3) 查看报表界面



(4) 打印报表界面



4.4 后端连接设计

4.4.1 顾客

以顾客更改预订信息为例：

```
1. def displaybooked(request):
2.
3.     if request.POST:
4.         alterroom[0] = request.POST.get('fanghao')
5.         print(alterroom)
6.         if alterroom[0] != '':
7.             if 'genggai' in request.POST:
8.                 return redirect('/alterdetail')
9.             elif 'quxiao' in request.POST:
10.                quxiaorow = models.bookrecord.objects.get(id=int(alterroom[0]))
11.                quxiaorow2 = models.bookrecord.objects.filter(id=int(alterroom[0]))
12.                # 预付金预订不退款
13.                if quxiaorow.btype == 1:
14.                    quxiaorow2.update(iscancel=1)
```

```

15.         return HttpResponse('<script>alert("取消
    预付金预订成功，无退款！
    ");location.href="/bookedroom";</script>')
16.         elif quxiaorow.btype == 2:
17.             yudingtime = quxiaorow.bdate
18.             if (yudingtime.replace(tzinfo=None)-
                datetime.datetime.now()).days > 3 :
19.                 quxiaorow2.update(iscancel=1)
20.                 return HttpResponse('<script>alert(
    "取消常规预订成功！");location.href="/bookedroom";</script>')
21.             else:
22.                 #先根据房型查价格
23.                 mtemp = models.rooms.objects.get(rn
                o=quxiaorow.bcno_id)
24.
25.                 money=mtemp.rprice
26.                 #生成付款记录
27.                 nowtime = datetime.datetime.now()
28.                 models.moneyrecord.objects.create(m
                type=3, mrno=quxiaorow.bcno_id, money=money,
29.                                                     m
                date=nowtime, mdays=quxiaorow.bdays,
30.                                                     m
                rtype=quxiaorow.broomtype)
31.                 return HttpResponse('<script>alert(
    "请支付第一天的房费！ "+str(money)+'
    元);location.href="/bookedroom";</script>')
32.
33.     allrooms = models.bookrecord.objects.filter(bcno_id=int
    (userno[0]),iscancel=0)
34.     return render(request, 'bookedroom.html', {'rooms':allr
    ooms})

```

4.4.2 雇员

以雇员为顾客分配房间为例，数据连接函数如下：

```

1. def allocate(request):
2.     if request.POST:
3.         allocateroom[0] = request.POST.get('fanghao')
            #得到前端发来的房号
4.         ydbh[0] = request.POST.get('bianhao')
5.         if allocateroom[0] != '' and ydbh[0] != '':

```



```

6.         allo1 = models.rooms.objects.get(rno=int(allocateroom[0]))
7.         allo2 = models.rooms.objects.filter(rno=int(allocateroom[0]))
8.         print(allocateroom[0],ydbh[0])
9.         yd = models.bookrecord.objects.get(id=int(ydbh[0]))
10.        if allo1.rin == 1:
11.            return HttpResponse('<script>alert("分配失败! 该房间已被分配");location.href="/allbooked";</script>')
12.        elif yd.broomtype != allo1.rtype:
13.            return HttpResponse('<script>alert("分配失败! 与用户预订房型不一样!");location.href="/allbooked";</script>')
14.        else:
15.            #房间入住
16.            allo2.update(rin=1)
17.            #生成入住记录
18.            addrecord = models.checkedrecord.objects.create(crno=allo1.rno, cdate=yd.bdate, ccno=yd.bcnno_id, cdays=yd.bdays)
19.            #更新收入记录中的房号
20.            temp = models.moneyrecord.objects.filter(mtype=1,mdate=yd.bdate,mdays=yd.bdays)
21.            temp.update(mrno=allo1.rno)
22.            return HttpResponse('<script>alert("分配成功! ");location.href="/allbooked";</script>')
23.
24.    allrooms = models.bookrecord.objects.filter(iscancel=0)
25.    allfangjians = models.rooms.objects.all()
26.    return render(request, 'allbooked.html', {'rooms':allrooms,'fangjians':allfangjians})

```

4.4.3 管理层

以管理层修改基价为例，数据连接函数如下：

```

1. def pricemanage(request):
2.     if request.POST:
3.         nowprice = request.POST.get('nowprice')
4.         nowtype = request.POST.get('fangxing')
5.         if nowprice != '':

```

```

6.         roomsrow = models.rooms.objects.filter(rtype=nowtype)
7.         roomsrow.update(rprice=int(nowprice))
8.         return HttpResponse('<script>alert("修改基价成功!");location.href="/alterprice";</script>')
9.
10.        room1 = models.rooms.objects.get(rtype='大床房')
11.        room2 = models.rooms.objects.get(rtype='双床房')
12.        room3 = models.rooms.objects.get(rtype='商务房')
13.        room4 = models.rooms.objects.get(rtype='家庭房')
14.        room5 = models.rooms.objects.get(rtype='六人豪华房')
15.
16.        return render(request, 'alterprice.html', {'room1':room1, 'room2':room2, 'room3':room3, 'room4':room4, 'room5':room5})

```

4.4.4 预订系统

以载入报表数据为例，用如下函数进行数据载入：

```

1. def myform(request):
2.     yuding = models.bookrecord.objects.all()
3.     shouru = models.moneyrecord.objects.all()
4.     allprice = models.moneyrecord.objects.aggregate(Sum('money'))
5.     return render(request, 'baobiao.html', {'rooms': yuding, 'fangjians':shouru, 'allmoney':allprice})

```

五、系统操作说明

5.1 客户操作说明

- ① 在首次登录时，用户需要先注册账号。要注意账号一定为数字，否则将注册失败；
- ② 用户登录后可以进行房间的查看和预订；
- ③ 预订需要填写预订日期和入住天数，可以选择预付金预订和常规预订两种方式。预付金预订的预付价格是基价的 75%，需提前一个月进行预订。
- ④ 可以修改预订信息，但使用预付金预订时如果金额减少了，将不退款。如果金额增加了，增加金额的金额将在更改时付款。
- ⑤ 在不需要时可以退订，但如果取消预付金预订，将不退款。

5.2 雇员操作说明

雇员可以为客人分配房间，但查询时需要输入正确的房间类型。雇员需要在客人离开时打印票据交给客人。

5.3 管理层操作说明

可以随时修改基价，且基价的修改不影响现有的预订。查看和打印报表都有相应的按钮。

六、软件测试

6.1 测试概要

6.1.1 概述

软件测试是伴随着软件的产生而产生的。早期的软件开发过程中软件规模都很小、复杂程度低，软件开发的过程混乱无序、相当随意，测试的含义比较狭窄，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对测试的投入极少，测试介入也晚，常常是等到形成代码，产品已经基本完成时才进行测试。到了上世纪 80 年代初期，软件和 IT 行业进入了大发展，软件趋向大型化、高复杂度，软件的质量越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计了各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征。人们还将“质量”的概念融入其中，软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且将测试作为软件质量保证(SQA)的主要职能，包含软件质量评价的内容，Bill Hetzel 在《软件测试完全指南》(Complete Guide of Software Testing)一书中指出：“测试是以评价一个程序或者系统属性为目标的任何一种活动。测试是对软件质量的度量。”这个定义至今仍

被引用。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。

软件测试已有了行业标准(IEEE/ANSI)，1983 年 IEEE 提出的软件工程术语中给软件测试下的定义是：“使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别”。

这个定义明确指出:软件测试的目的是为了检验软件系统是否满足需求。它再也不是一个一次性的,而且只是开发后期的活动,而是与整个开发流程融合成一体。软件测试已成为一个专业,需要运用专门的方法和手段,需要专门人才和专家来承担。

6.1.2 测试原则

对计算机软件进行测试前,首先需遵循软件测试原则,即不完全原则的遵守。不完全原则即为若测试不完全、测试过程中涉及免疫性原则的部分较多,可对软件测试起到一定帮助。因软件测试因此类因素具有一定程度的免疫性,测试人员能够完成的测试内容与其免疫性成正比,若想使软件测试更为流畅、测试效果更为有效,首先需遵循此类原则,将此类原则贯穿整个开发流程,不断进行测试,而并非一次性全程测试。

6.2 测试方法

软件测试的方法有:① 静态测试方法;② 动态测试;③ 黑盒测试;④ 白盒测试。

软件测试的策略有:① 单元测试;② 集成测试。

在本次实验中,本小组选择的测试方法为:黑盒测试。主要测试系统的每个功能能否正常运作。

黑盒测试,顾名思义即为将软件测试环境模拟为不可见的“黑盒”。通过数据输入观察数据输出,检查软件内部功能是否正常。测试展开时,数据输入软件中,等待数据输出。数据输出时若与预计数据一致,则证明该软件通过测试,若数据与预计数据有出入,即便出入较小亦证明软件程序内部出现问题,需尽快解

决。

6.3 功能测试

功能模块		合法等价类		非法等价类		测试结果
		输入数据	预期输出	输入数据	预期输出	
注册模块	录入	账号：19040118	注册成功	账号：马茜晔	注册失败，账号必须为数字	成功
		密码：123456		密码：123456		
		确认密码：123456		确认密码：123456		
	录入	账号：19040118	注册成功	账号：19040118	注册失败，两次密码输入应该相同	成功
		密码：123456		密码：123456		
		确认密码：123456		确认密码：12345678		
	录入	账号：19040118	注册成功	账号：19040118	注册失败，密码不能为空	成功
		密码：123456		密码：空		
		确认密码：123456		确认密码：空		
顾客登录模块	录入	账号：19040118	登录成功	账号：19040118	登录失败，密码不能为空	成功
		密码：123456		密码：空		
	录入	账号：19040118	登录成功	账号：空或不是数字	登录失败，需要输入正确账号	成功
		密码：123456		密码：123456		
顾客订房模块	录入	大床房、2021.12.23、入住天数：2天、预付金预订	预订成功，请付款	三人房、2021.12.23、入住天数：2、预付金预订	预订失败，没有该房间类型	成功
	修改预订	用户名：张三、预订方	修改成功，请	用户名：张三、预订方	修改失败，修改	成功

顾客修改预订模块		式：预付金预订	及时缴费	式：预付金预订	数据不能为空	
		原入住天数：2		原入住天数：2		
		新入住天数：3		新入住天数：空		
顾客修改个人信息模块	修改信息	用户名：张三	修改成功	用户名：张三	修改失败，新密码不能为空	成功
		原密码：12345		原密码：12345		
		新密码：123456		新密码：空		
雇员分配房间模块	检索	房间类型：大床房	大床房房间为：101-109	房间类型：三人房	无该房间类型	成功
			未入住房间号为：101、102、103，可为客人分配			
管理层修改基价	修改基价	房间类型：大床房	修改成功	房间类型：空	修改错误，房间类型不能为空	成功
		原基价：999/晚		原基价：999/晚		
		新基价：900/晚		新基价：900/晚		
		房间类型：大床房	修改成功	房间类型：大床房	修改错误，新基价不能为空	成功
		原基价：999/晚		原基价：999/晚		
		新基价：900/晚		新基价：空		

6.4 维护

本系统主要采取三种方式维护：

（1）改正性维护

在软件初期使用，由于某种原因，软件存在一些错误或缺陷。需要由软件开发小组按用户要求对系统进行改正性维护。

（2）适应性维护

当系统长时间使用后，随着用户的增加，系统在某些方面的速度可能不如刚开始的时候，所以针对这个问题我们要做出适当维护。

（3）完善性维护

在系统使用几年后，该应用系统可能不适应业务的发展。用户对软件提出了新的需求和性能要求。