《数据库系统》课程设计报告

题目	宾馆客房管理系统		
小组成员信息			
姓名	**学号**	**班级**	**分工**
王天龙	18340168	18级计科6班	系统用户管理模块，前端实现
楚鸿飞	18340030	18级计科1班	信息管理模块，前端实现
林旭生	18340112	18级计科4班	客房预订管理模块，整合报告
彭一铭	18340137	18级计科5班	订单结账管理模块，攥写报告

提交时间：2020 年 1 月 10 日

一. 开发环境与开发工具

- 操作系统：Windows10
- 开发语言：python、HTML、CSS、MySQL
- 开发工具：PyCharm、MySQL workbench
- 浏览器环境：Chrome、Firefox、Safari
- 第三方库：flask

二. 系统需求分析(2 分)

（1）项目简介

本次项目我们小组开发了一套宾馆客房管理系统，从实际应用出发，分析了生活中酒店用户的需求，设计了数据库模型，并创建数据库与配套的前端，所设计的系统具有如下功能模块：

- 系统的用户管理：包括客户的注册、登录，管理员的登录等；
- 客户的信息管理：包括用户基本信息的查询、添加、删除、修改等；
- 客房的信息管理：包括客房基本信息的查询、添加、删除、修改等；
- 客房的预订管理：包括客房预订信息的查询、添加、删除、修改等；
- 客房的结账管理：包括账单信息的查询、添加、删除、修改等。

通过以上功能模块，我们基本实现了以下功能：

- 具有方便的客房预订、登记、结账功能，支持团体预订和团体结账。
- 管理者与顾客可以快速查看酒店内的客房状态，便于管理与预定。
- 提供多种手段查询客人信息。

- 具备一定维护手段，只有酒店管理员登录后才能进行更改房价、房间类型、增减客房等操作。
- 具有完善的结账报表系统。

（2）项目要求

本次项目需要设计并实现一个数据库应用，具体要求包括：

- 有完整的前后端架构，前端为界面，后端为数据库。
- 支持在前端界面对后端数据库进行增删查改操作。

（3）项目动机

在选择期末设计项目的时候，我们小组成员本着中大人的家国情怀精神，决定要设计一个能够真正服务大众，为人民生活带来便利的好应用。想到学校附近的贝岗村里有不少宾馆经常被同学们光顾，生意火爆，而处在如此地段的小宾馆想必很难拥有一个完善的酒店管理系统。怀着心系天下的大爱，我们小组在众多的选题中选择了宾馆客房管理系统，来帮助这些可能买不起大酒店管理系统的小宾馆老板，帮助他人的同时也间接使得同学们使用宾馆更加便捷，可谓利人利己。

我们的应用部署在Web端，使用方便，没有安装门槛，界面简洁明了，便于应用的使用与传播。

（4）系统数据字典

2.4.1 数据结构

- **管理员**数据结构
  - 名字：`admin`
  - 描述：在一个宾馆客房管理系统中，我们的酒店管理员需要能对数据库进行较高权限的操作，因此需要一个名为 `admin` 的关系表格来存储管理员信息。
  - 定义：`admin` = ID + 姓名 + 密码 + 电话 +邮箱
  - 位置：保存到管理员信息表。
- **客户**数据结构
  - 名字：`guest`
  - 描述：为了方便用户在网上直接订房，我们需要名为 `guest` 的表格记录用户信息，想要订房的用户必须首先在网站上注册，在数据库中登记了自己的信息后，才能在线订房。
  - 定义：`guest` = ID + 姓名 + 密码 + 电话 + 邮箱 + 国籍
  - 位置：保存到客户信息表或显示于前端管理员登录下的 `Guests` 页面。
- **客房**数据结构
  - 名字：`room`
  - 描述：一个专门的表格用于记录酒店中的房间信息，以方便管理员管理和用户订房，我们把这个表格叫做 `room` 。
  - 定义：`room` = ID + 类型 + 价格 + 窗户数量 + 是否允许吸烟 + 床数量 + 是否有浴缸

    - 位置：保存到客房信息表或显示于前端的 `Rooms` 页面 。
- **客房预订**数据结构
    - 名字：`booking`
    - 描述：用户订房后，会生成相应的房间预定信息，被保存在 `booking` 这个关系表格中。
    - 定义：`booking` = 预定ID + 房间ID + 用户ID + 开始时间 + 结束时间 + 是否包含餐食 + 入住成人数量 + 入住儿童数量 + 是否已付款
    - 位置：保存到客房预订信息表、显示于前端管理员登陆下的 `Bookings` 页面或前端客户登陆下的 `Orders` 页面。
- **账单**数据结构
    - 名字：`bill`
    - 描述：在用户使用完房间后需要进行结账，账单信息保存在 `bill` 这个关系表格。
    - 定义：`bill` = 账单ID + 用户ID + 开始时间 + 结束时间 + 总价
    - 位置：保存到账单信息表、显示于前端管理员登陆下的 `Bills` 页面或前端客户登录下的 `Bills` 页面。

2.4.2 “管理员”数据结构之数据项

> 以下只展示部分数据项定义，其他数据项定义略。

表2-4-2-1 “g_id”数据项

名字	g_id
别名	客户ID
描述	唯一标识客户在系统中的数字编号
定义	整数型
位置	客户信息表、客房预订信息表、账单信息表

表2-4-2-2 “name”数据项

名字	name
别名	管理员姓名或客户姓名
描述	管理员或客户的基本信息——姓名
定义	字符型
位置	管理员信息表、客户信息表

2.4.3 数据流

> 以“客房预订数据流”来举例描述：

- 数据流名：客房预订数据流；
- 说明：“客房预订”数据结构在系统内的流向；
- 数据流来源：客户预订事务；
- 数据流去向：预订信息处理事务；
- 平均流量：每天几十次；
- 高峰期流量：每天上百次。

2.4.4 数据存储

> 以客房预订信息表数据存储来说明如下：

- 数据存储名：客房预订表；
- 说明：客房预订信息数据，作为原始数据需要保存与备查；
- 编号：`booking_id` 为唯一标识，顺序整数，从 1 开始每次增加 1；
- 输入的数据流：客户预订数据流，来自客户预订操作；
- 输出的数据流：客户结账数据流，用于客户结账记录；
- 数据结构：“客户”、“客房预订”、“账单”

2.4.5 处理过程

> 以“结账”的处理过程说明如下：

处理过程名：结账

- 说明：客户在使用完房间后需要进行结账，要能根据客房的日单价、订房时间来计算出该客户名下的记录所需要支付的价钱；
- 输入：客户预订数据流，客户结账数据流；
- 输出：计算出指定客户的订单所需支付的费用；
- 处理：统计客户名下各个订单所需支付的费用，每个订单需要根据入住时间和退房时间计算住房天数，再由房间ID查询到客房单价进而相乘。

2.4.6 数据字典

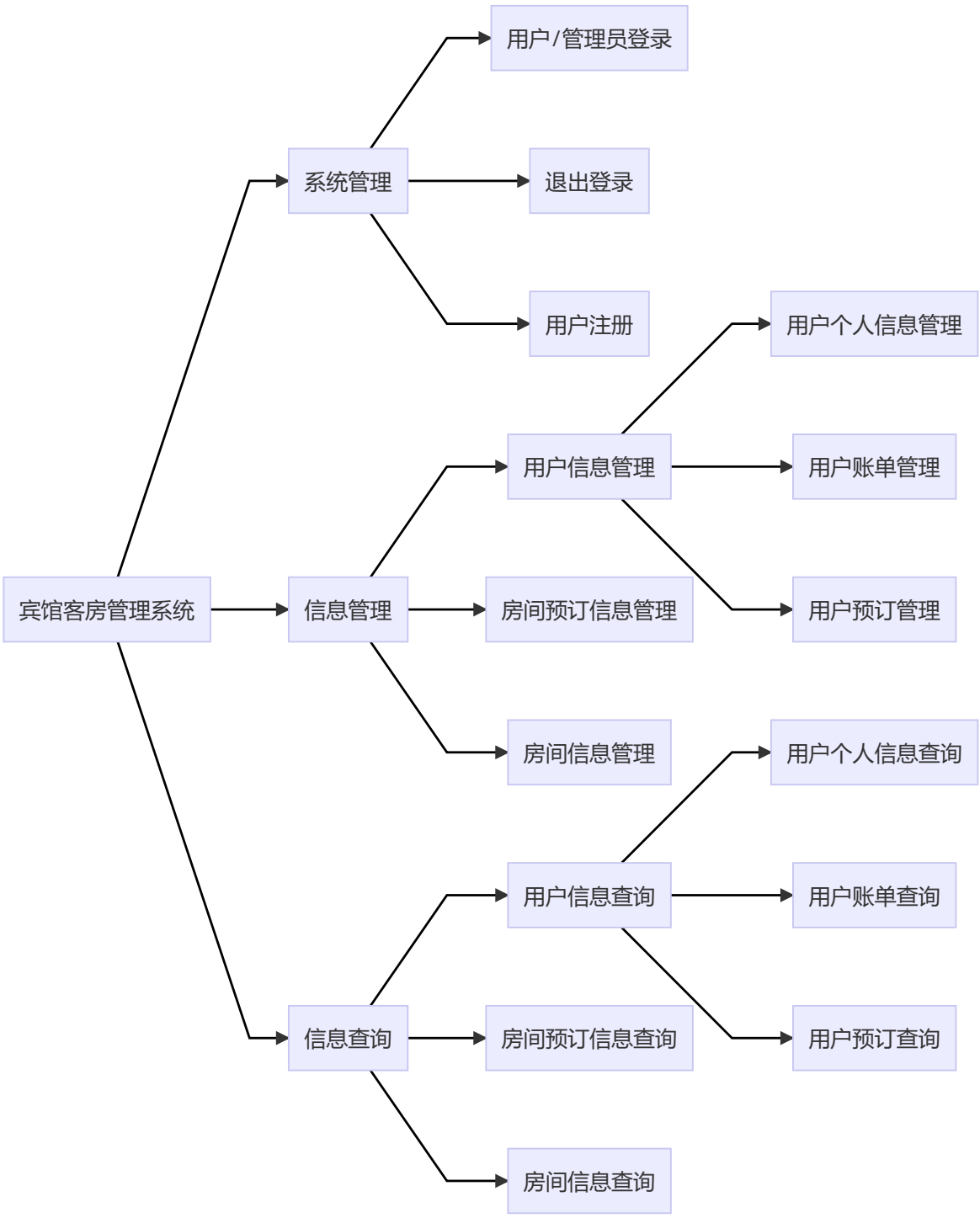
综上所述，我们的数据库系统一共使用了五个关系，分别为 `admin`、`guest`、`room`、`booking`和`bill`，它们的具体内容如下。

<table>
<tr><th>表名</th><th>数据字典表</th><th>数据项描述</th></tr>
<tr><td>admin</td><td>
<table>
<tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr>
<tr><td>a_id</td><td>int(11)</td><td>NO</td><td>PRI</td><td>NULL</td><td>auto_increment</td></tr>
<tr><td>name</td><td>varchar(45)</td><td>NO</td><td>UNI</td><td>NULL</td><td></td></tr>
<tr><td>password</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>phone</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>e_mail</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
</table>
</td><td>管理员ID
管理员姓名
管理员密码
管理员电话
管理员邮箱</td></tr>
<tr><td>guest</td><td>
<table>
<tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr>
<tr><td>g_id</td><td>int(11)</td><td>NO</td><td>PRI</td><td>NULL</td><td>auto_increment</td></tr>
<tr><td>name</td><td>varchar(45)</td><td>NO</td><td>UNI</td><td>NULL</td><td></td></tr>
<tr><td>password</td><td>varchar(256)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>phone</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>e_mail</td><td>varchar(45)</td><td>YES</td><td></td><td>NULL</td><td></td></tr>
<tr><td>country</td><td>varchar(45)</td><td>YES</td><td></td><td>NULL</td><td></td></tr>
</table>
</td><td>用户ID
用户姓名
用户密码
用户电话
用户邮箱
用户国籍</td></tr>
<tr><td>room</td><td>
<table>
<tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr>
<tr><td>r_id</td><td>int(11)</td><td>NO</td><td>PRI</td><td>NULL</td><td>auto_increment</td></tr>
<tr><td>r_type</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>price</td><td>float</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>num_window</td><td>int(11)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>allow_smoke</td><td>varchar(5)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>num_bed</td><td>int(11)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>bathtub</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
</table>
</td><td>房间ID
房间类型
房间价格
窗户数量
是否允许吸烟
床数量
是否有浴缸</td></tr>
<tr><td>booking</td><td>
<table>
<tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr>
<tr><td>booking_id</td><td>int(11)</td><td>NO</td><td>PRI</td><td>NULL</td><td>auto_increment</td></tr>
<tr><td>r_id</td><td>int(11)</td><td>NO</td><td>MUL</td><td>NULL</td><td></td></tr>
<tr><td>g_id</td><td>int(11)</td><td>NO</td><td>MUL</td><td>NULL</td><td></td></tr>
<tr><td>from</td><td>date</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>to</td><td>date</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>meal</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>num_adult</td><td>int(11)</td><td>YES</td><td></td><td>NULL</td><td></td></tr>
<tr><td>num_child</td><td>int(11)</td><td>YES</td><td></td><td>NULL</td><td></td></tr>
<tr><td>is_paid</td><td>varchar(45)</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
</table>
</td><td>预定ID
房间ID
用户ID
开始时间
结束时间
是否包含餐食
入住成人数量
入住儿童数量
是否已付款</td></tr>
<tr><td>bill</td><td>
<table>
<tr><th>Field</th><th>Type</th><th>Null</th><th>Key</th><th>Default</th><th>Extra</th></tr>
<tr><td>bill_id</td><td>int(11)</td><td>NO</td><td>PRI</td><td>NULL</td><td>auto_increment</td></tr>
<tr><td>g_id</td><td>int(11)</td><td>NO</td><td>MUL</td><td>NULL</td><td></td></tr>
<tr><td>from</td><td>date</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>to</td><td>date</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
<tr><td>total_money</td><td>float</td><td>NO</td><td></td><td>NULL</td><td></td></tr>
</table>
</td><td>账单ID
用户ID
开始时间
结束时间
总价</td></tr>
</table>

三. 功能需求分析

（1）MySQL实现的宾馆客房管理系统功能需求

酒店管理系统的功能模块并不复杂，主要实现了对用/客户信息、房间预定信息、房间信息、账单信息等的管理，系统功能模块图如下。



(2) 基于 web 的网上宾馆客房预订系统功能需求

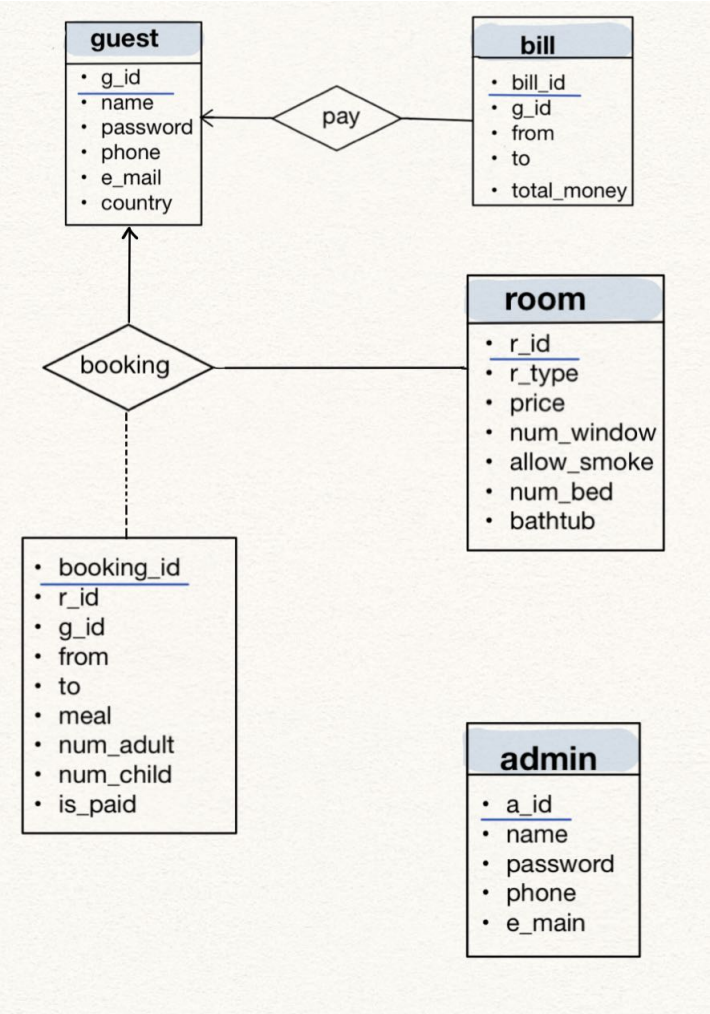
- 能实现网上用户的注册于登录，登录用户的管理；
- 能方便查询客房信息，方便订房选择；
- 能实现基本的客房预订功能，包括房型选择、日期预订、人数登记等；
- 能完成订单支付，并产生相应账单记录；
- 用户事后能查询自己的历史订单及明细数据；
- 具有商务网站的基本功能，包括系统简介、用户注册、关于我们等；

- Web 网页系统运行稳定、可靠，操作简单、方便。

四. 系统设计

（1）数据概念结构设计（系统ER图）

> 本项目设计的ER图如下图所示



（2）数据库关系模式设计

- 注意到`bill`实体集和`booking`联系集中都包含`from`和`to`属性，但是由于预订房间的居住时间与实际居住时间不同，比如客人可能会在预订时间的最后一天再续住两天，所以我们在`bill`实体集和`booking`联系集中都建立了`from`和`to`属性，这并不造成数据冗余，也不会出现更新异常的问题
- 一个客人存储在酒店的数据库里，没有什么特殊原因是不会被删除的，所以这就意味着会出现一个客人有多笔支付订单和预订订单的现象，故我们将`guest`与`bill`设计为一对多的关系；同样的，一个客人可能需要团体预订，所以会出现一个客人预订了多个房间，故我们将`guest`与`room`设计为一对多的关系。

（3）数据库物理结构设计

- 刚开始我们是想把`guest`和`admin`两个实体集合二为一的，但是这势必需要添加一个属性判断这个登陆者是否为管理员身份，而这不仅造成了数据冗余因为客人完全没有必要拥有这个属性，而且客人的数量可能会十分庞大所以搜索管理员的耗时会增加，于是我们想到了以下两种解决方法：
  - 使用索引：在整个实体集中建立属于管理员的索引。虽然这样能够解决搜索管理员的耗时问题，但是这却会降低更新表的速度，如对表进行`INSERT`、`UPDATE`和`DELETE`。因为更新表时，MySQL不仅要保存数据，还要保存一下索引文件。
  - 建立一个只属于管理员的实体集：我们小组觉得这不仅完美解决了上述问题，而且还便于后期维护，所以我们采用了建立一个只属于管理员的实体集的方法
- 因为`guest`和`admin`的权限不同，所以我们设计`guest`只有对`booking`和`bill`的插入或删除操作权限；而`admin`则还有对`room`的插入、删除、更新及`bill`的访问和使用数据的操作权限

五. 系统功能的实现

（1）主要功能模块的实现过程

5.1.1 前端页面设计

前端页面设计方面，我们主要是改造现成的模板，或者借助前端代码生成网站实现。由于本次实验重点不在于前端，所以在这里就不一一介绍。具体可以查看GitHub 。

5.1.2 后端开发

这里主要介绍python如何于mysql语句进行连接，然后对数据库进行操作。对于如何实现web应用的具体逻辑就不一一介绍，具体可以查看GitHub 。

- 连接mysql数据库

  在python里连接mysql数据库，主要是依靠`flask_mysqldb`库来实现。在flask的配置里添加以下信息

```
from flask_mysqldb import MySQL
app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'  # 数据库地址
app.config['MYSQL_USER'] = 'root'  # 用户名
app.config['MYSQL_PASSWORD'] = '1234'  # 密码
app.config['MYSQL_DB'] = 'hms'  # 使用的数据库
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'
mysql = MySQL(app)
```

  连接数据后，就可以在python里执行mysql，进行插入、删除、查询等操作

```
cur = mysql.connection.cursor()  # 开启一个连接
cur.execute('select * from room')  # 执行select命令
rooms = cur.fetchall()  # 取查询结果
cur.close()  # 关闭连接
```

在python中要执行mysql语句，要想使用`cur=mysql.connection.cursor()`开启一个连接；然后使用`cur.execute()`执行mysql语句；如果要获取查询的结果，可以用`cur.fetchall()`或`cur.fetchone()`函数，分别是取回所有查询到的记录和取回第一条记录；如果对数据库进行了修改，修改后要使用`mysql.conntion.commit()`来立即提交；最后不要忘了使用`cur.close()`来关闭连接。

- 插入数据

  mysql里支持批量插入，可以在一条`insert`语句里插入多条记录，从而避免程序和数据库建立多次连接，从而增加服务器负荷。批量插入命令如下

```
INSERT INTO

[表名]([列名],[列名])

 VALUES

([列值],[列值])),

([列值],[列值])),

([列值],[列值]));
```

只要把要插入的值接在`VALUES`后面即可，在python里插入数据主要用`add`函数实现。

```
# table: 字符串，插入的表名；values: 字典，key: 变量名，value: 变量值(string或数值)；num，是插入的数量
def add(table, values, num=1):
    cmd = 'insert into `{}` ('.format(table)
    insert_values = '('
    for v in values:
        if values[v] != '':
            cmd += '`{}`,'.format(v)
            insert_values += '"{}",'.format(values[v])
    cmd = cmd.strip(',')
    cmd += ') values '
    insert_values = insert_values.strip(',')
    insert_values += ')'
    for i in range(num):
        cmd += insert_values + ','
    cmd = cmd.strip(',')
    # print(cmd)  输出为insert into
    cur = mysql.connection.cursor()
    try:
        cur.execute(cmd)
```

```
        mysql.connection.commit()  # 对数据库进行修改后，要用commit进行提
交
    except Exception as e:
        print('insert error: {}'.format(e))
    cur.close()  # 关闭连接
```

在用户添加订单、管理员添加房间等操作时，都是用到了这个函数。

- 更新数据

  mysql语句中，对某个表里的记录更新如下

```
UPDATE table_name SET field1=new-value1, field2=new-value2 [WHERE
Clause]
```

在python里对数据库更新使用`update_value`函数实现，具体定义如下

```
# table: 字符串，插入的表名；values: 字典，key: 变量名，value: 变量值
(string或数值)；condition: 字符串，where语句的选择条件
def update_value(table, values, condition='true'):
    cmd = 'update `{}` set '.format(table)
    for v in values:
        cmd += '`{}`="{}",'.format(v, values[v])
    cmd = cmd.strip(',')
    cmd += ' where ({})'.format(condition)
    cur = mysql.connection.cursor()
    try:
        cur.execute(cmd)
        mysql.connection.commit()
    except Exception as e:
        print('update error: {}'.format(e))
    cur.close()
```

在管理员更新房间、用户修改订单等情况下，都是用该函数实现。

- 查询数据

  查询数据，使用的是`select`语句。

  以查询当天的空闲房间为例

```
@app.route('/rooms')
def rooms_list():
    cur = mysql.connection.cursor()
    cur.execute('select * from room')  # 查询到所有房间
    rooms = cur.fetchall()
    num = len(rooms)
    today = datetime.date.today()  # 获取当天日期
    for i in range(num):  # 遍历查到的房间，判断每一房间是否在当天空闲
        result = cur.execute(
            'select * from booking where r_id="{}" and `from`<="{}" and
`to`>"{}"'.format(rooms[i]['r_id'], today, today))  # 判断该房间是否空闲
        if result > 0:
```

```
            rooms[i]['available'] = False
        else:
            rooms[i]['available'] = True
    cur.close()
    return render_template('rooms.html', rooms=rooms)  # 返回给前端
```

前端发起请求后，就会执行这个函数，返回酒店所有房间，以及房间状态。

- 删除数据

  删除数据库里的数据，使用的是`delete`语句

  以删除一个房间为例

```
@app.route('/delete_room/<int:r_id>', methods=['POST'])
@is_admin_logged_in  # 要求要有管理员权限
def delete_room(r_id):  # 指定要删除房间的房间号
    cur = mysql.connection.cursor()
    cur.execute('delete from room where r_id="{}"'.format(r_id))
    mysql.connection.commit()
    cur.close()
    return redirect('/rooms')  # 返回前端
```

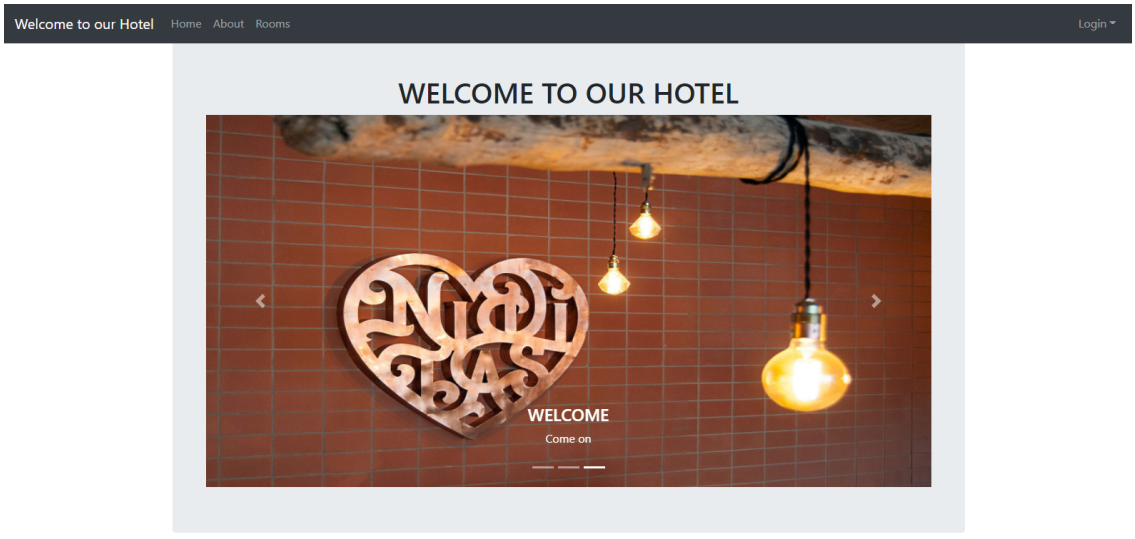
5.1.3 部署

我们的数据库管理系统已经部署到云服务器上，可以通过http://121.36.251.61/访问。由于云服务器是免费使用的，所以在2021/02/02前可以访问到。部署的方案是，使用nginx做反向代理，uwsgi做web服务器，具体部署过程就不一一介绍。

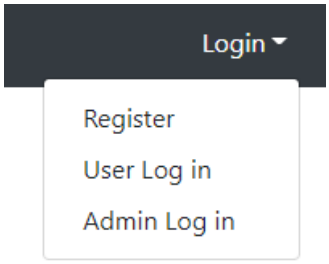
（2）运行界面展示

5.2.1 用户运行界面

- 主页



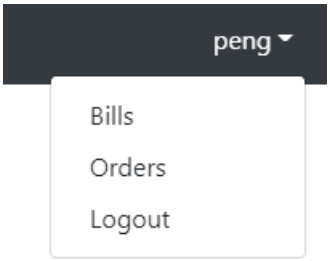
> 进入网页后，点击右上角下拉框，可以选择注册或登录。



注册新用户，注册完成后自动跳转到登录界面。

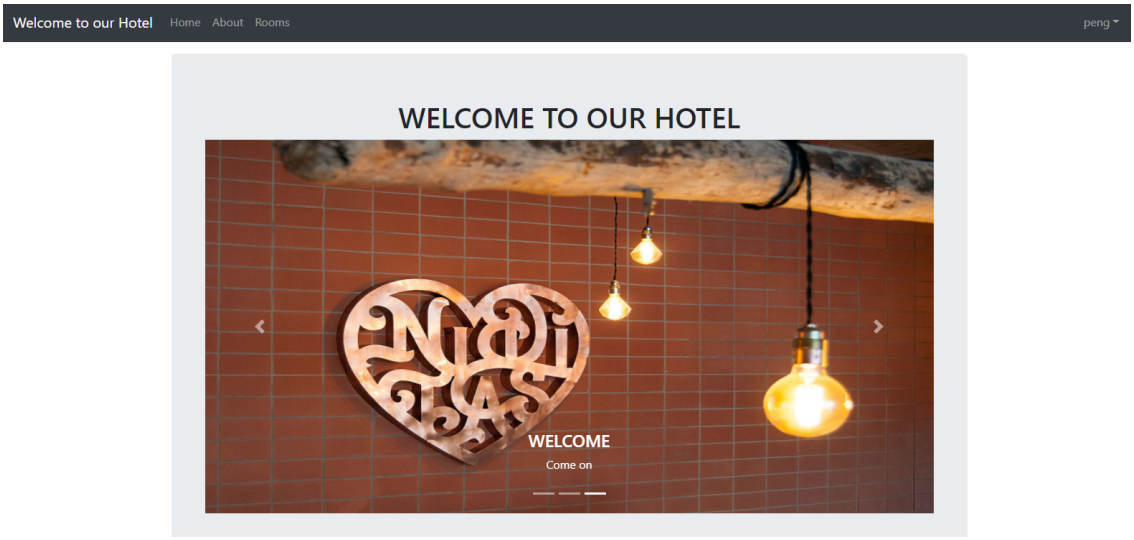
- 用户登录

> 登陆后返回主页，此时右上角下拉框显示用户名，点击用户名，可以查看当前的账单和预定，也可以退出登录。



- Home

> 点击左上角的Home，返回主页。

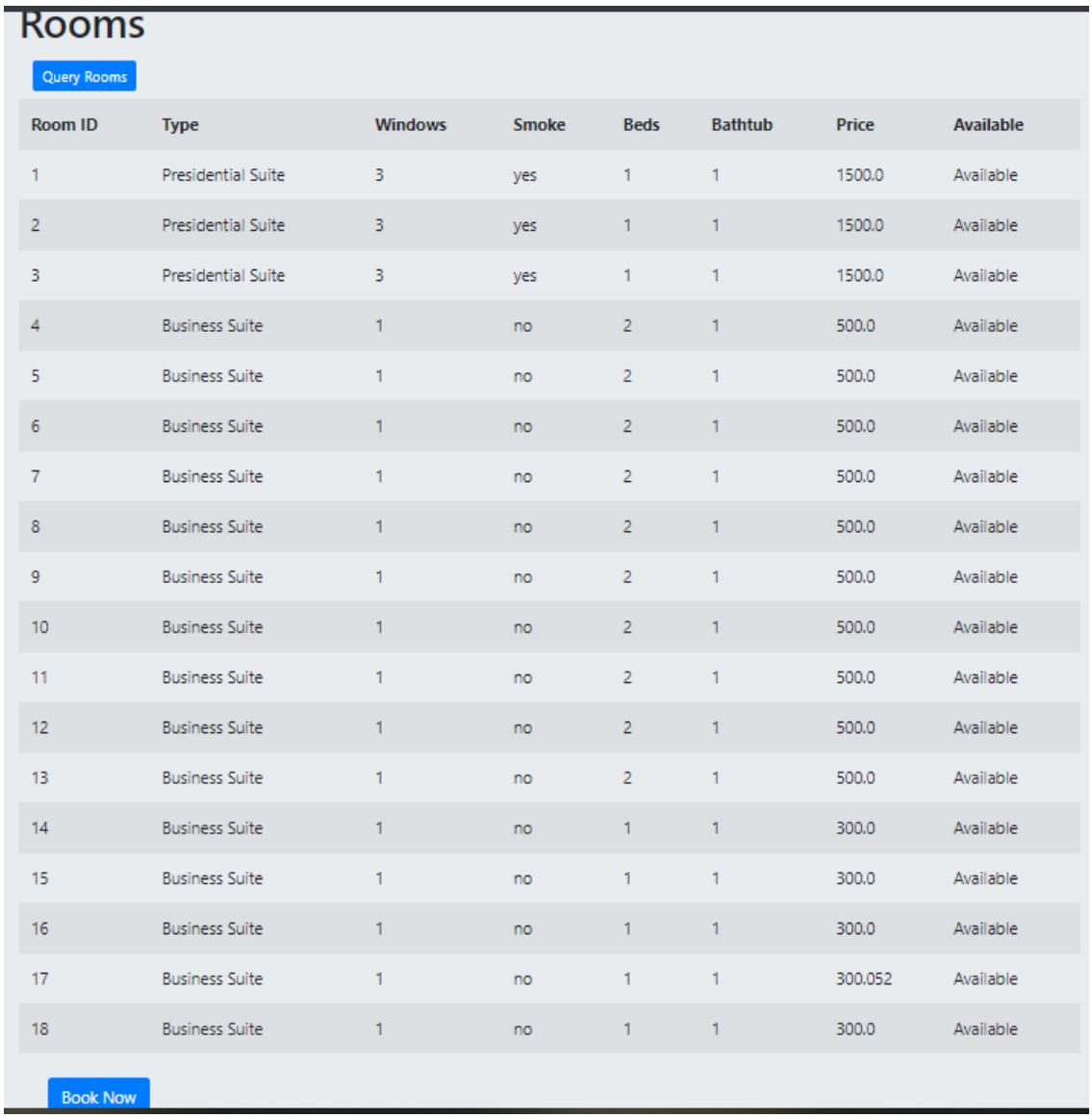


- About

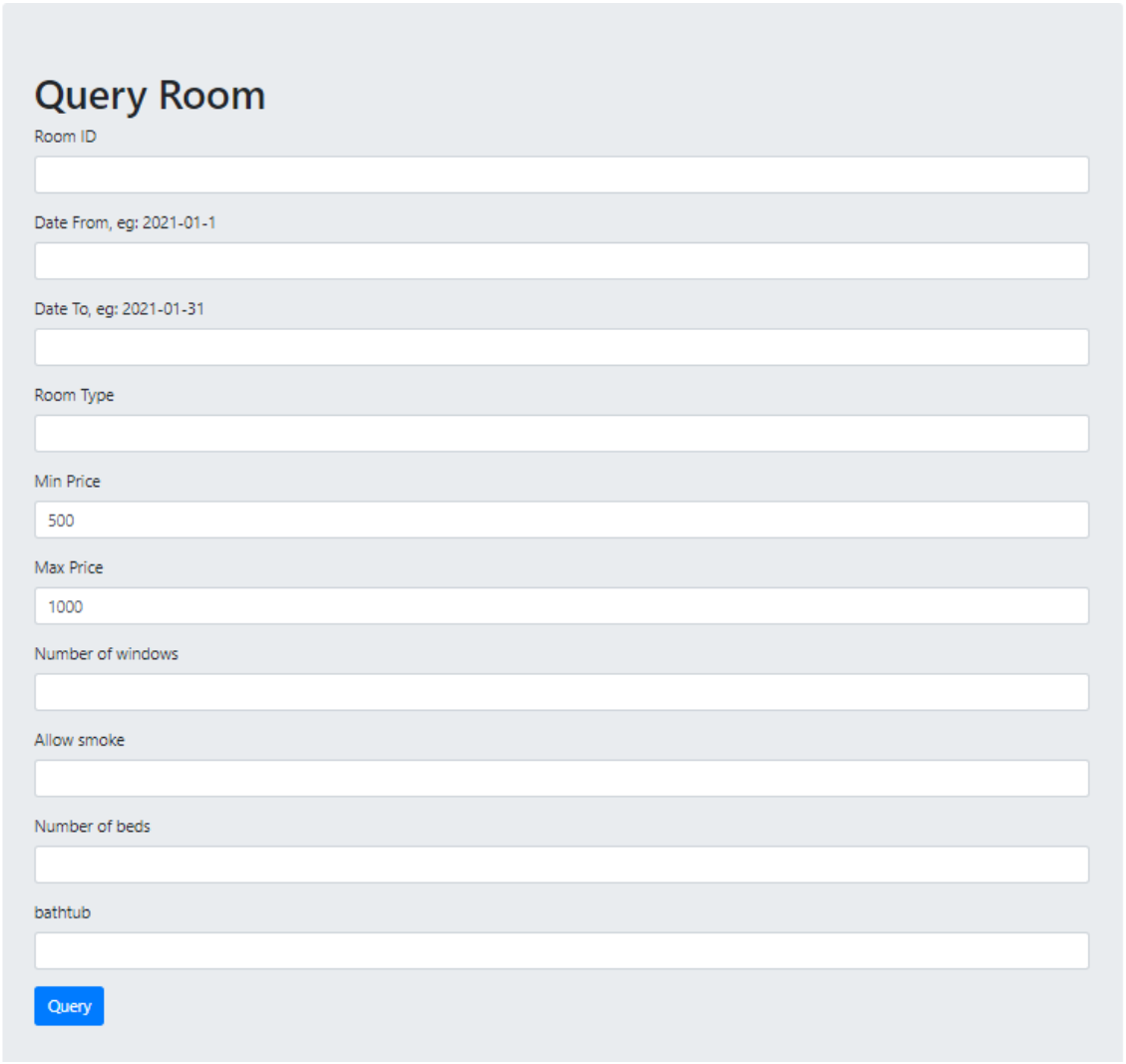
> 点击左上角的About，查看酒店相关信息。

- Rooms

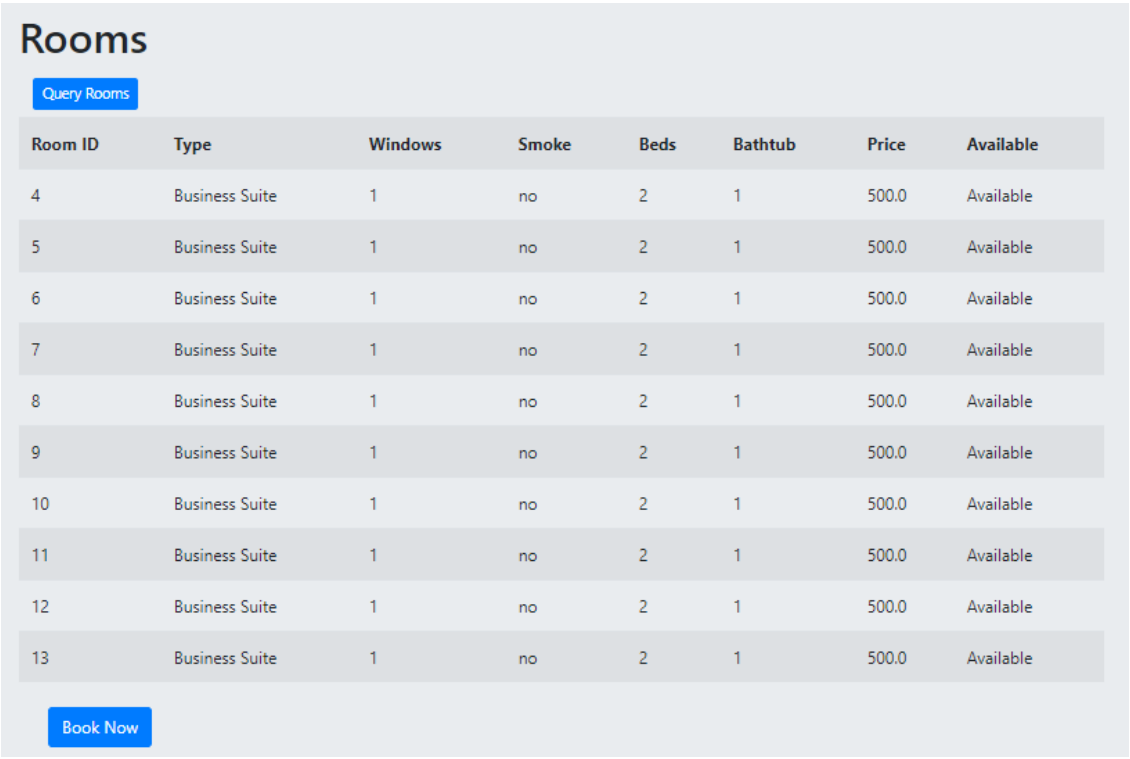
> 点击左上角的Rooms，查看房间信息与预定房间。



> 点击Query Rooms，可以根据个人要求筛选查询房间。

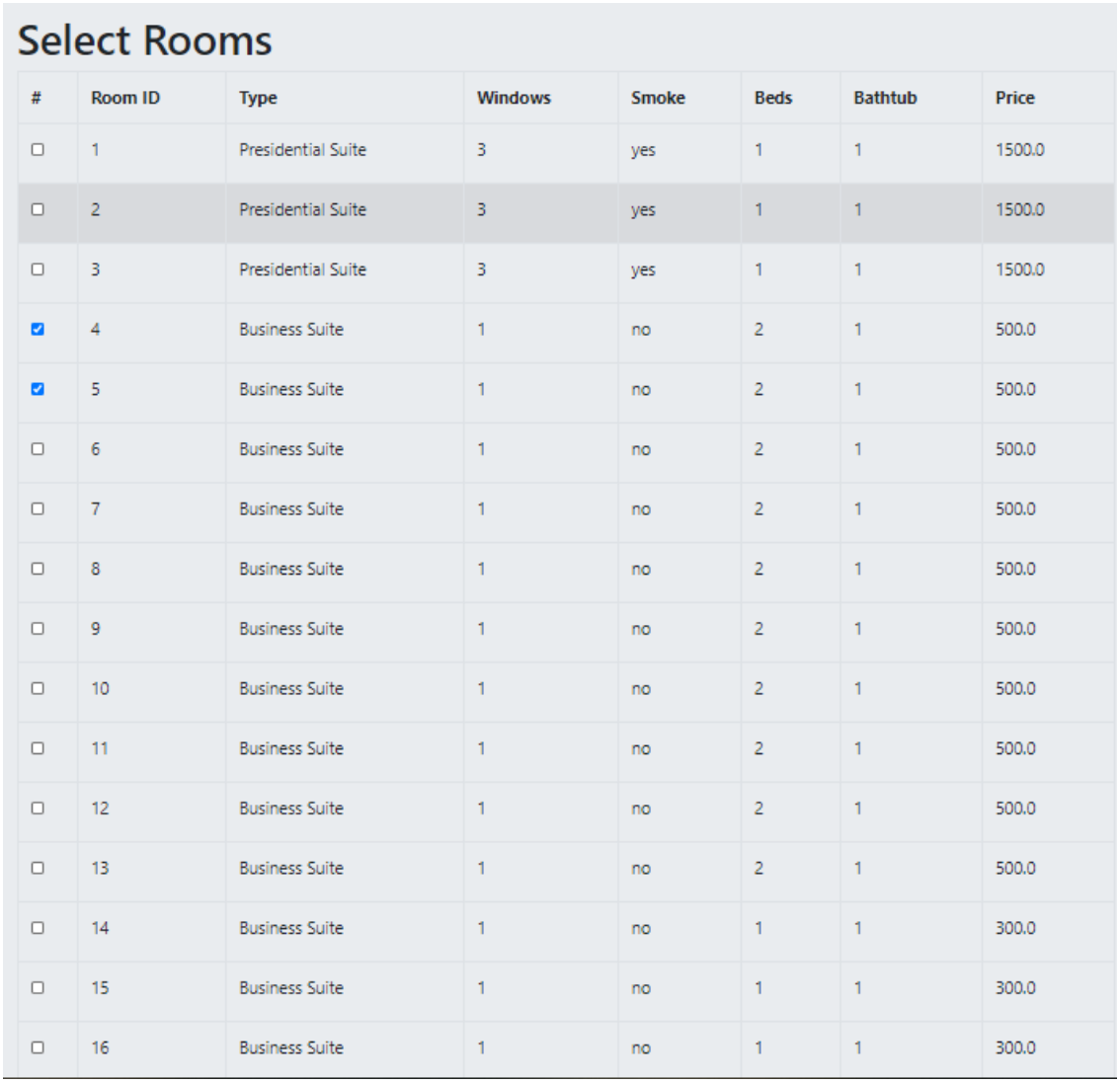


如上图希望查看价格为500-1000的房间，查询结果如下。



点击Book Now，填写相关信息，然后选择要预定的房间，即可对房间完成预定。

> 注意到在选择房间时可以对多个房间进行勾选，便于用户进行团体预定操作。



- Orders

> 点击右上角用户名，在下拉框中选择Orders，即可查看当前的订单。若想要取消订单，只需勾选要取消的订单并点击Cancel the Order即可。

Order for Guest: peng

#	Booking ID	Room ID	Check in	Check out	meal	num_adult	num_child	price	days	cost	is_paid
☐	5	4	2021-01-06	2021-01-07	no	2	0	500.0	1	500.0	no
☐	6	5	2021-01-06	2021-01-07	no	2	0	500.0	1	500.0	no

Total Charges: 1000.0

Cancel the Order

- Bills

> 点击右上角用户名，在下拉框中选择Bills，即可查看需要支付的订单，当Orders中的订单到期（根据真实世界中的时间）时，会被自动转移到Bills中，可以看到由于现在还没到预定的生效时间（Check in Date），因此Bills中仍然为空。

Bill for Guest: peng

Booking ID	Room ID	Price	Check In Date	Check Out Date	Days	Cost

Total Charges: 0

Pay

> 要对订单进行支付，只需点击Pay，就可以看到老板的微信收款码，紧跟时代潮流，使用方便快捷。

5.2.2 管理员运行界面

- 添加管理员

> 在服务器上运行一下`add_admin.py`脚本，按照提示输入用户名、电话号码、邮箱、密码。

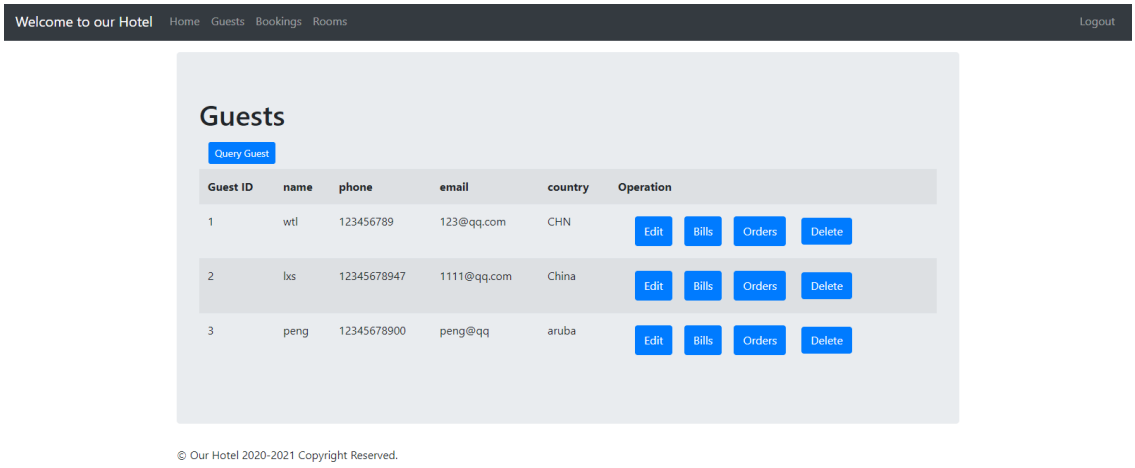
```
(env) wtl@hecs:~/database$ python add_admin.py
User Name: admin101
Phone: 123456
E-mail: 123@mail.com
Password:
Confirm:
```

- 管理员登录

> 在主页左上角点击Login，选择Admin login in，即可进行管理员登录。管理员登陆后页面上方的导航栏会发生变化，没有了About这一项，而多了Guests和Bookings这两项。

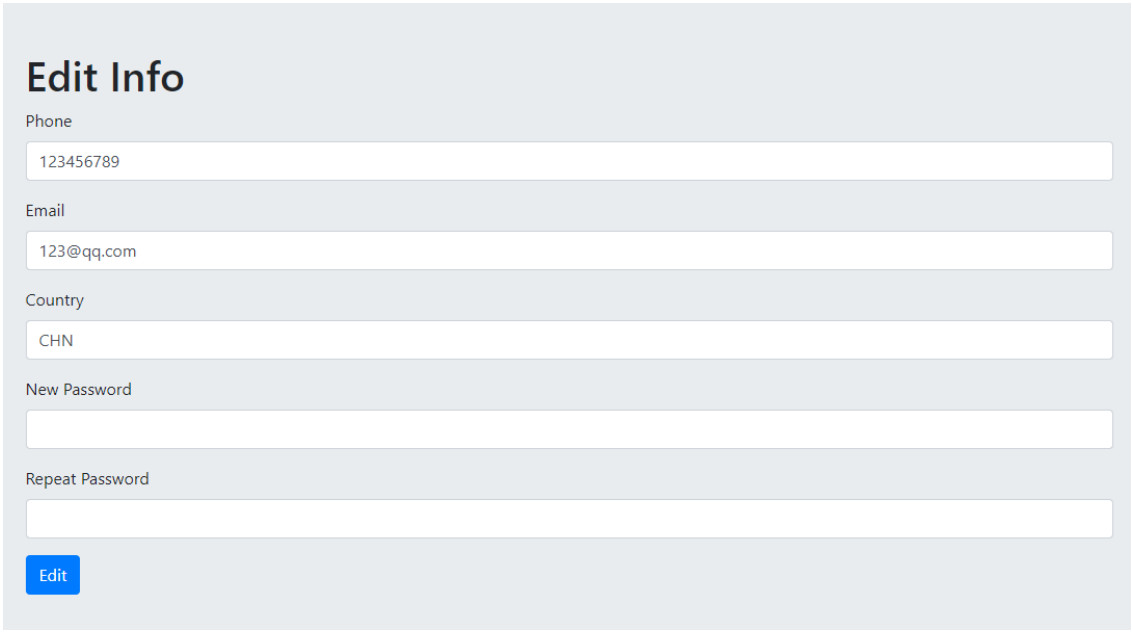
- 查看客户信息

> 点击导航栏中的Guests，进入查看客户信息的界面。

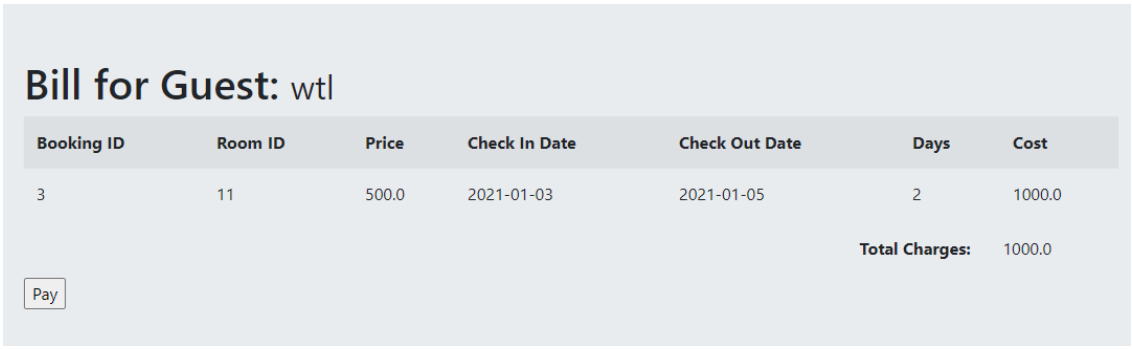


> 点击Query Guest，可以根据条件对客户进行筛选

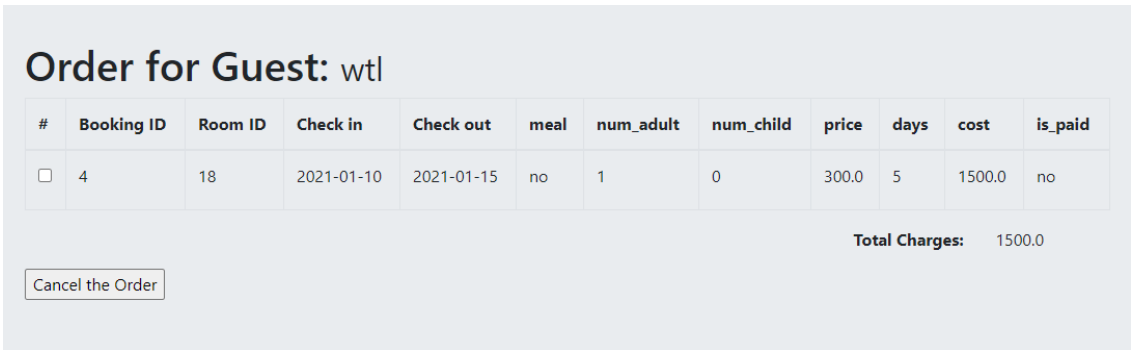
> 点击Edit，可以对客户信息进行修改。



> 点击Bills，查看客户待付的账单。



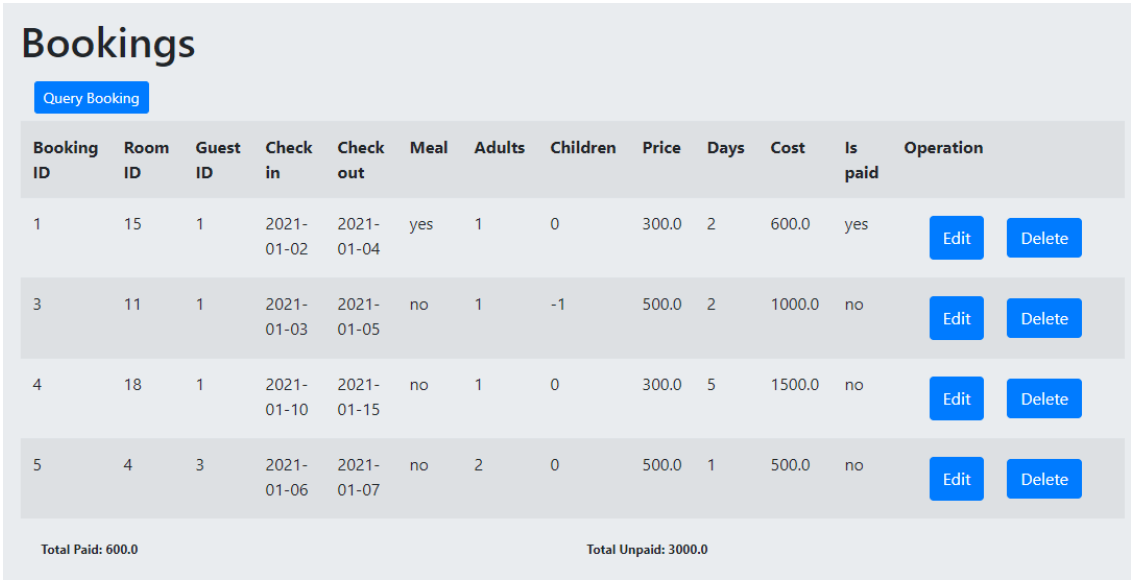
> 点击Orders，查看客户的预定。



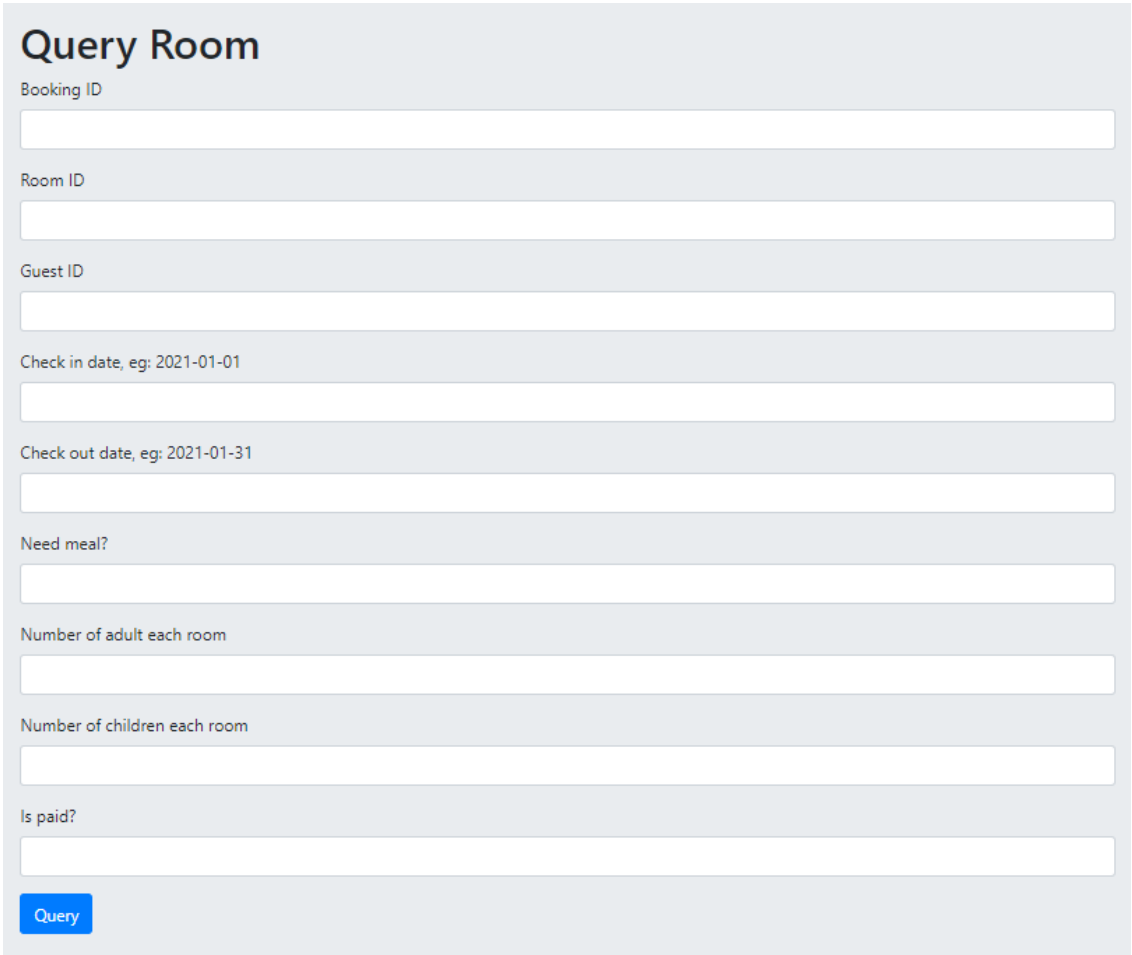
> 点击Delete，可以删除客户信息。

- 查看所有预定信息

> 点击导航栏的Bookings，可以查看所有的预定信息，包括以前的和还未生效的，并且可以同时查看支付信息。



> 点击Query Booking，可根据条件筛选订房信息。



> 点击Edit，可以对订房信息进行修改。

> 点击Delete，可以删除订房信息。

可以看到，我们的数据库针对普通用户与管理员开放了不同权限的操作，方便管理的同时也增加了数据库的安全性，对重要数据的删改操作只有管理员才能进行。

六. 总结

本课程设计中用到的《数据库系统》理论课概念与知识。

- **需求分析**：需求分析是数据库设计的起点，直接影响后续阶段的设计和数据库系统能否被合理使用。分四个步骤：确定数据库范围、分析数据应用过程、收集与分析数据和编写需求分析报告。需求分析的目标是了解与分析用户的信息及应用处理的要求，并将结果按一定格式整理形成需求分析报告。
- **数据库关系模式设计**：作为整个系统设计的中心，需要根据需求分析考虑各个功能的实现，进而设计合理的关系以完整实现整个系统。
- **数据约束**：数据的安全保密性，针对不同类型数据的操作权限；数据完整性，指数据正确性的约束和验证准则，及一致性保护的要求；响应时间，主要指某些特点应用要求的数据存取时间限制；数据恢复，主要指转储及恢复的时机与范围等。
- **概念及逻辑结构设计**：概念设计结构是在需求分析中产生的需求分析报告基础上按照特定的方法设计满足应用需求的用户信息结构，通常称为概念模型。常用方法有实体分析法（自顶向下）和属性综合法（自底向上）。
- **物理结构设计**：物理设计指对于一个给定的数据库逻辑结构，研究并构造物理结构的过程，主要是确定数据库在存储设备上的存储结构及存储方法， 因 DBMS 不同可能包括建立索引和聚集等待。
- **实体-联系模型（E-R模型）**：在课程设计的过程中，我们构建了要实现数据库的实体-联系模型，并明确每一个实体集、联系集中的主码、外码等约束，确定每个属性的变量类型，以及它们的取值范围，这些工作让我们的设计思路更加清晰，对后续数据库的代码实现有很大的帮助。
- **数据定义语言、数据操纵语言**：灵活运用数据定义语言对数据库需要的关系等进行创建，利用数据操纵语言在表中增加、删除、查询、修改数据库中的数据，实现数据库应用的基本功

能。