上海交通大学

数据库课程设计报告

设计题目：房屋租赁管理系统

学生姓名：张恒(5123709228)、成城 (5123709126)

系　　别：密西根学院

专　　业：电子计算机工程

指导教师：李 芳

一、开发背景

1.1 背景：

由于 Internet 的迅速普及，在线房屋租赁成为 Internet 技术在企业管理信息系统中的应用和延伸，形成了集计算机，计算机网络、数据库、分布式计算等于一体的信息技术综合体，它打破了时间和地域的界限，使信息交流变得快捷、准确，为建立现代企业管理信息系统提供了充足的条件。企业信息管理系统在此基础上延伸、扩展，使之上下、内外全面贯通。引入 Internet 后，形成了新型的浏览器/服务器（Browser/Server）结构,而传统的客户机/服务器（Client/Server）结构在这方面就远不及 B/S 结构。作为房屋租赁公司的管理者，希望能够对房屋租赁事务管理的整个流程状态，信息资料的情况了如指掌，使其可以做出科学的决策。作为工作人员期望能够避免繁琐的手工操作，甩掉传统的手工记录方式，达到事半功倍的效果。一个能够使其实现管理系统化、规范化、自动化的计算机系统就显得很有必要。利用计算机技术,实现管理系的自动化,规范化就是这个问题最好的解决方法。

1.2 目的：

利用计算机支持企业高效率完成房屋租赁治理的日常事务，是适应现代企业制度要求、推动企业劳动型治理走向科学化、规范化的必要条件；而房屋租赁治理是一项琐碎、复杂而又十分细致的工作，房屋的基本资料，客户资料的治理，房屋租赁治理，收费以及统计表的治理，一般不允许出错，假如实行手工操作，须手工填制大量的表格，这就会耗费工作人员大量的时间和精力,计算机进行房屋租赁工作的治理,不仅能够保证各项信息准确无误、快速输出，同时计算机具有手工治理所无法比拟的优点.例如：检索迅速、查找方便、可靠性高、存储量大、保密性好、寿命长、成本低等。这些优点能够极大地提高物业治理的效率，也是企业的科学化、正规化治理，与世界接轨的重要条件。

开发本系统就是为了解决企业在房屋租赁信息治理中的一些不规范,使房屋租赁信息的治理向着规范化、简单化、有效化的方向发展。

二、功能描述

2.1 系统描述

建立一个房屋租赁管理系统，统一管理中介、租赁者以及房屋信息，以便快速地提供租赁服务。该系统应具有以下功能目标：

1. 该租房公司在全国各地设有分部，每个分部下属多个办公室，每个办公室都有一定数量的房源；

2. 房型按照卧室和卫生间的数量划分，主要有：
   1B1B, 2B1B, 2B2B, 3B2B, 3B3B, 4B2B, 4B4B；
3. 系统需要维护客户表。当一个客户第一次租房时，系统会记录该客户的姓名、性别、电话和邮箱。以客户的电话号码作为标识区分不同的客户；
4. 客户可以在线选择自己想要入住的房型、入住时间和搬出时间。如果此时间段该房型有空房，系统会显示预估房租，客户可以选择预订或是取消。如果客户决定预订房，系统会打印预订确认单。客户还可以选择意向室友。客户如果取消预订，需要提供确认单号码以及日期；
5. 客户入住时需要提供预订时确认单号码或电话号码。客户可以选择是否需要保险，买保险可以选择特定的时间。付房租需要客户提供身份证号码、信用卡信息（包括号码以及信用卡过期日期等相关信息）；
6. 客户搬出时需要记录搬出日期，系统自动根据使用的时间计算房租，客户使用信用卡付款。

系统会维护一个房源表，每年经理会根据房龄（超过 10 年）决定一部分房源淘汰。如果房源被标记"ordered"，说明房可以出售。 然后该房不可以被租用。出售价格是买入价格的一半。

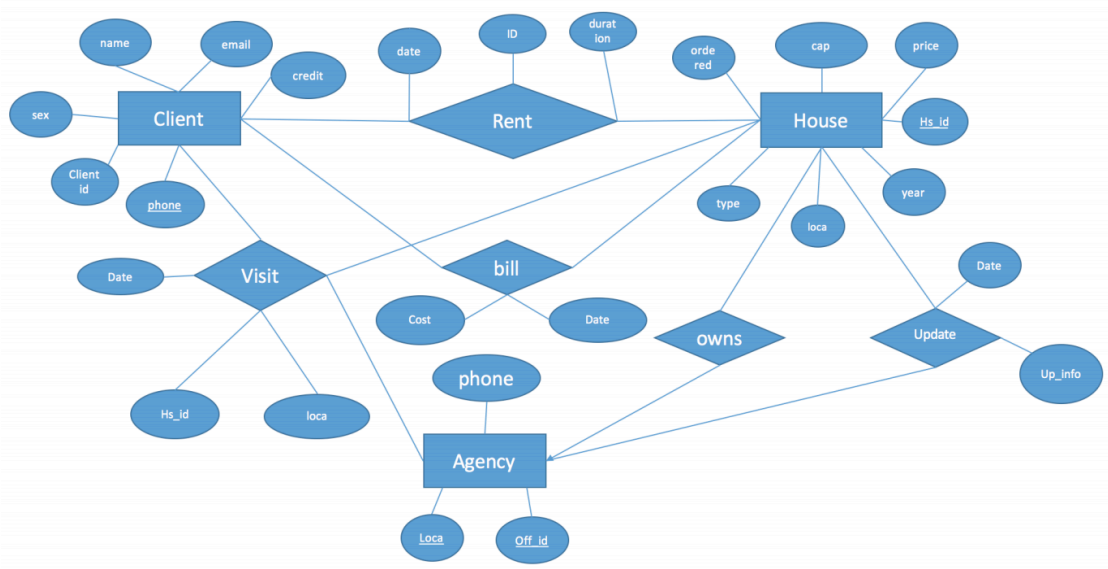
2.2 系统可行性分析

系统现阶段的发展过程中，利用现有人力和物力是完全具备能力开发出来的，作为阶段性产品，日后的发展空间大，实现方法简单容易。本系统采用 web.py 和 mysql 相结合的方法来实现。要求所有数据信息的储存都由数据库来完成，而这些数据信息的调用由 web.py 完成。

目前，房屋租赁现象非常普遍，许多都要求通过中介公司进行管理，并且实际中也有系统的使用。所以本系统的使用市场非常的广阔，易于推广和普及到现实中应用。所以本系统在市场应用上是完全可行的。

2.3 系统分析

1. E/R 模型设计



(1) Agency(locationAgency, idAgency, telephone);表示房屋中介，location 表示地址，idAgency 表示该中介在本地的 id。俩一起作为其主键, telephone 即为中介电话

(2) Client(username, idClient, email, credit_no, password, gender, clientid, create_time); 用户表，credit_no 表示信用卡信息，用来付房租，username 为主键。

(3) House(ordered_status, cap, price, idHouse, year, type, locationHouse); 房屋表，locationHouse, type 和 idHouse 用来做主键，ordered_status 为布尔型，表示是否已经被预订。Year 为房龄，用来表示是否可以出售。

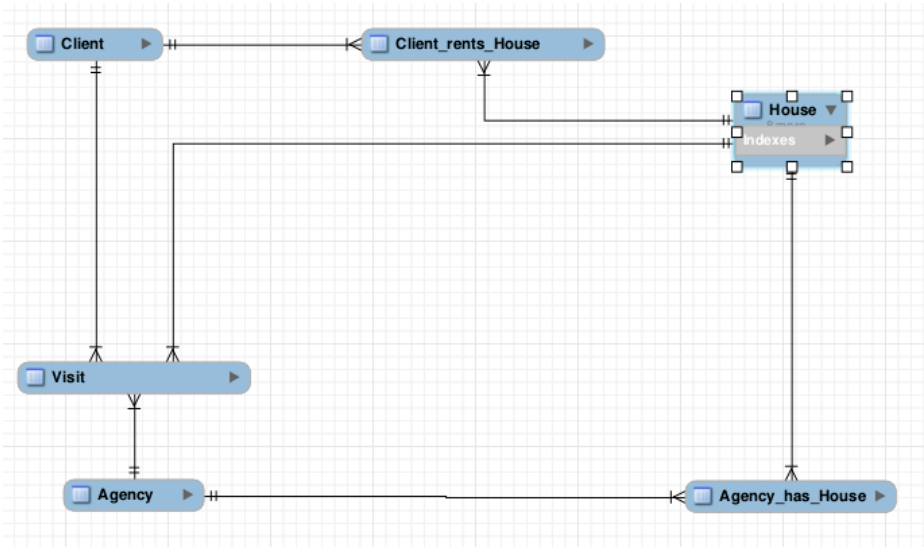
(4) Client_rents_House(idTransaction, duration, startDate, cost); 租赁关系，duration 表示租用时间段。

(6) Visit(dateVisit, locationHouse)看房关系，

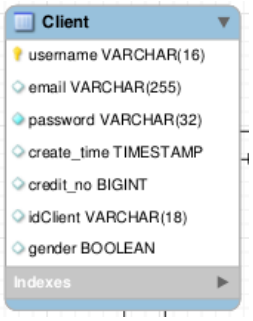
(7) Agency_has_House()；表示某房属于中介。

2. MySQL 实现

使用图型化工具 MySQL Workbench 生成 E/R 关系图如下图所示

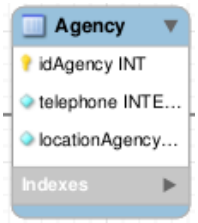


Client 表

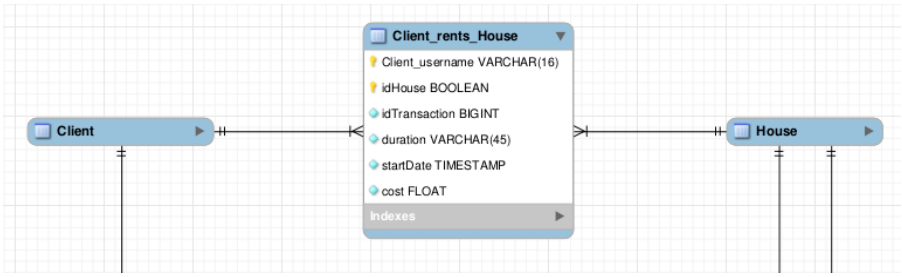


House 表

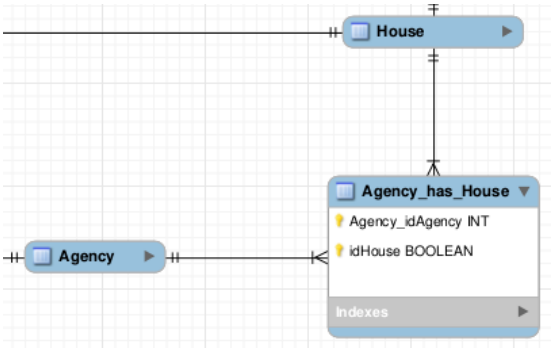
Agency 表



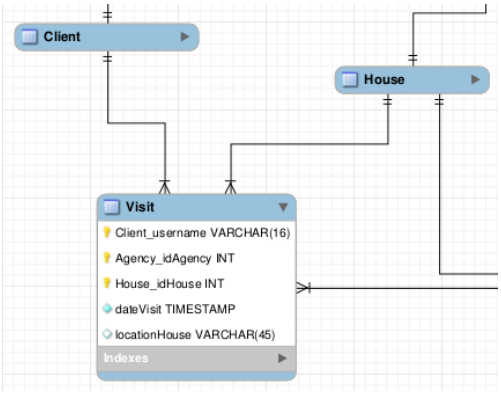
Client 和 House 之前存在租用关系



Agency 和 House 之间存在拥有关系



Visit 代表看房事件，由 Agency 和 Client 共同安排



三、逻辑模型设计和优化

E-R 图中所需的几个表的设计：

```
mysql> desc Agency;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| idAgency        | int(11)     | NO   | PRI | NULL    |       |
| telephone       | varchar(11) | NO   |     | NULL    |       |
| locationAgency  | varchar(45) | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

表 3..1：中介信息表

```
mysql> desc Client;
+--------------+-------------+------+-----+-------------------+-------+
| Field        | Type        | Null | Key | Default           | Extra |
+--------------+-------------+------+-----+-------------------+-------+
| username     | varchar(16) | NO   | PRI | NULL              |       |
| email        | varchar(255)| YES  |     | NULL              |       |
| password     | varchar(32) | NO   |     | NULL              |       |
| created_time | timestamp   | NO   |     | CURRENT_TIMESTAMP |       |
| credit_no    | bigint(20)  | YES  |     | NULL              |       |
| idClient     | int(11)     | YES  |     | NULL              |       |
| gender       | char(1)     | YES  |     | NULL              |       |
+--------------+-------------+------+-----+-------------------+-------+
7 rows in set (0.00 sec)
```

表 3.2 ：用户信息表

```
mysql> desc House;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| idHouse        | int(11)     | NO   | PRI | NULL    |       |
| ordered_status | tinyint(1)  | NO   |     | NULL    |       |
| price          | float       | NO   |     | NULL    |       |
| type           | varchar(45) | YES  |     | NULL    |       |
| year           | int(11)     | YES  |     | NULL    |       |
| locationHouse  | varchar(45) | NO   |     | NULL    |       |
| cap            | varchar(45) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

表 3.3：房屋信息表

```
mysql> desc Visit;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Client_username | varchar(16) | NO   |     | NULL    |       |
| Agency_idAgency | int(11)     | NO   |     | NULL    |       |
| House_idHouse   | int(11)     | NO   |     | NULL    |       |
| dateVisit       | varchar(20) | NO   |     | NULL    |       |
| locationHouse   | varchar(45) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

表 3.4：看房信息表

四、应用程序设计

由于房屋租赁管理系统涉及信息的录入，更新和查询等等操作，需要直观性强一目了然的界面设计，故此我们选择了 web.py 作为程序设计语言，具体界面和介绍如下：

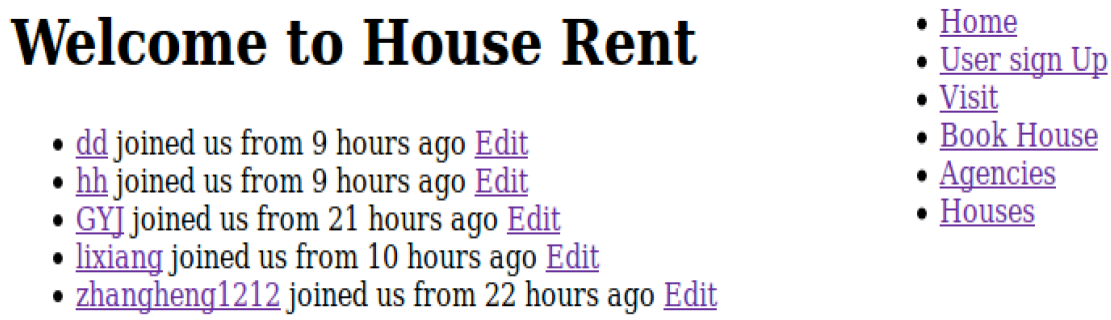


图 4.1：房屋租赁管理系统主界面：

左边信息页显示了所有已经录入信息的用户名字，点击名字可以显示用户信息。

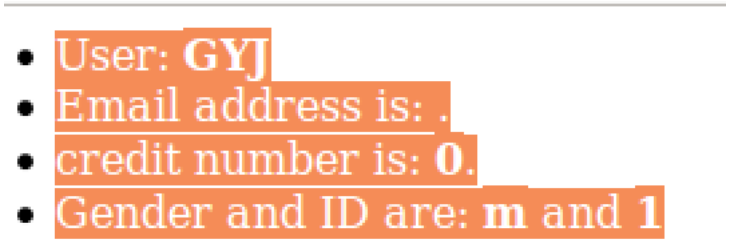


图 4.2　用户信息界面：

点击“Edit”，可以进入用户信息编辑页：

图 4.3　用户信息修改界面

点击“Delete Client”可以删除该用户。修改信息填入后，点击“Submit”可以修改该用户信息。

点击“User Sign Up”：

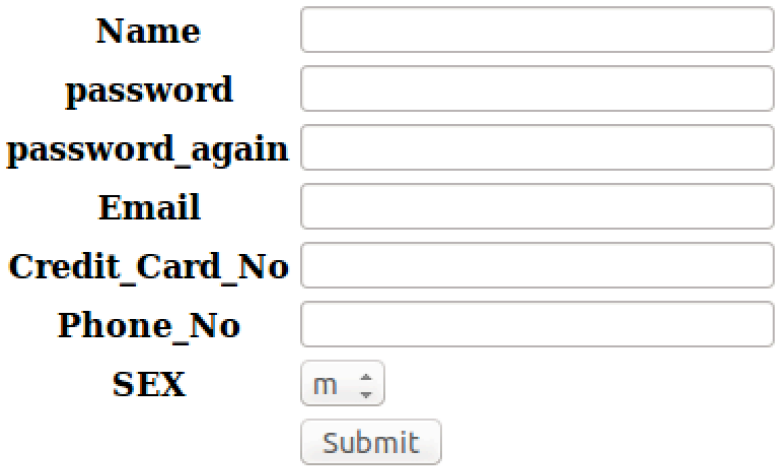


图 4.4　房屋信息录入

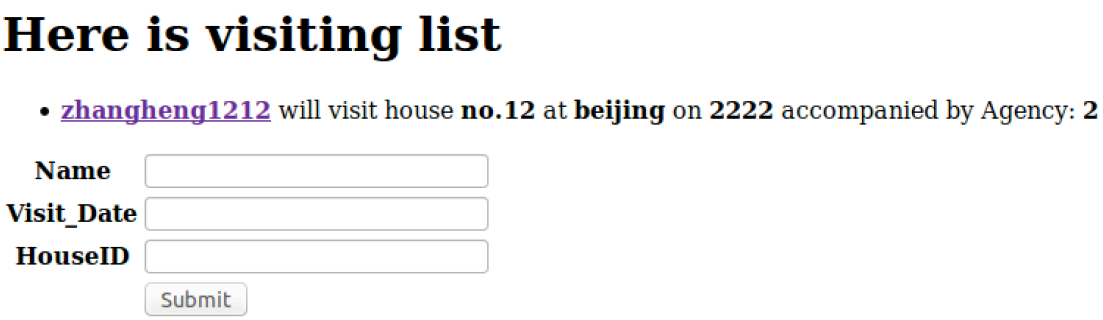


图 4.5　参观信息显示及录入

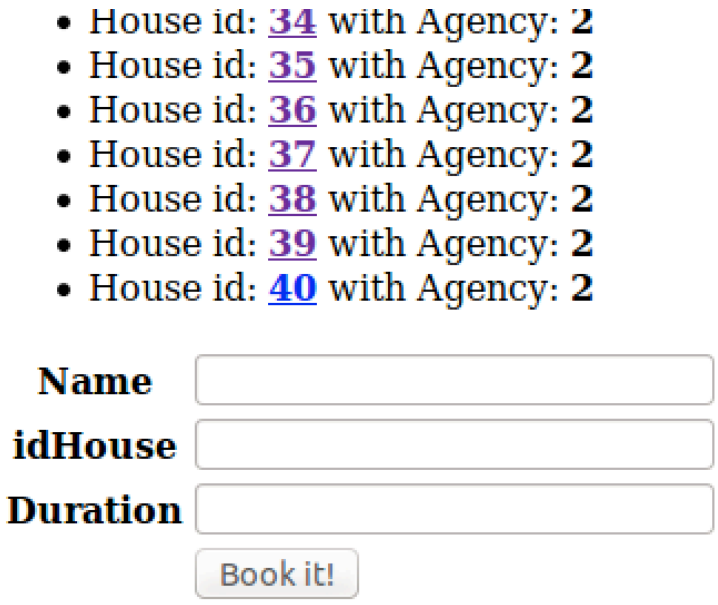


图 4.6　房屋预定

点击“House id”可以显示房屋状态以及是否被预订：

图 4.7　房屋预订状态

点击“status”可以擦开订单的具体信息：

图 4.8　房屋订单信息

点击主页面的 Agencies 可以显示中介信息页：

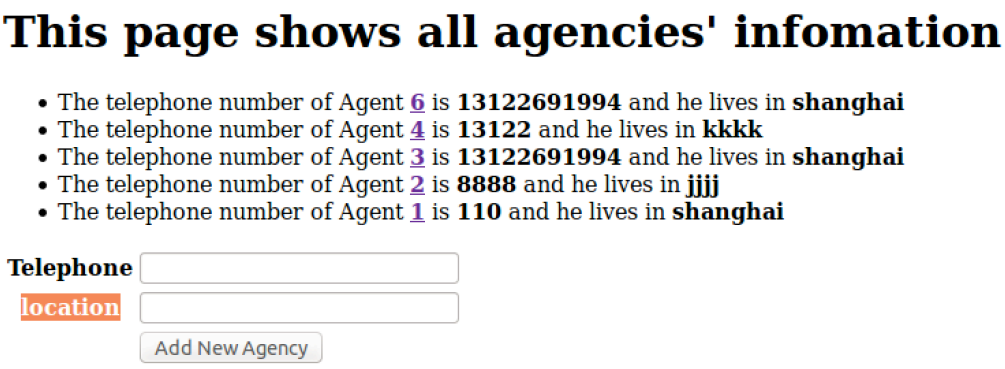


图 4.9　中介信息图

点击“Agency ID”可以查看中介拥有的房屋信息

图 4.10　中介房屋信息图

点击“delete”可以删除该中介和该房屋的代理关系。点击“Delete Agency”可以删除该中介。

五、课程设计心得体会

这次的程序完全是我自己借助数据库和 web.py 教材以及从图书馆所借的一本参考书完成的。看到自己把所出现的错误一个个地改出来，心里真的是很欣慰。我体会到了自己完成一次作业的成就感，更重要的是我体会到了认真对待一件事并好好完成的愉悦，这将对我以后产生很大帮助。

通过这一次课程设计，我把在数据库原理中的知识应用到实践，这让我明白，理论永远是理论，要是没有实践，理论永远是一纸空文。我们要从实践中才能学到更多的东西。而实践又要以理论为基础，要是数据结构知识不扎实，做出的程序也是漏洞百出。所以，我们要将理论和实践结合起来，把我们在课堂上学到的东西运用在实际中，这样才能提高我们自身的能力。

虽然这次课程设计小组每个人做的题目都不一样，但是在任务中，大家互帮互助,团结一心，使得各自最终的程序能圆满运行。我体会到了团队的重要性，互帮互助的好处。

在接到任务时，其实心里有点恐惧，感觉自己没有能力做出来，就想着反正可以上网去搜，也就不怎么着急。后来想到如果总是依赖互联网，而不去努力，自己永远不可能有提高，于是开始自己试着写。当程序终于可以运行时，那一刻感觉好自豪，感觉自己所有的努力都没有白费。更让我重新理解了那就名言“当你只有一个目标时，整个世界都会给你让路”。

在课程设计过程中，我们意识到，团队精神是很重要的，组员一起分工合作，遇到困难一起讨论解决，集思广益，才能使程序更加完善。

在学习 web.py 的时候，我清楚的明白了学习需要细心和耐心，特别是设计比较复杂的界面程序时，而在真正设计的时候还是会粗心。程序设计的过程中，还有很多是由于粗心而导致的错误，也有很多是对以前的知识遗忘了，这就告诫自

己以后做什么事都要认真仔细，不要大大咧咧。即使再有能力，不认真，不付出努力，所谓的能力也终究只能是空有一身本领。

六、主要界面代码

python 代码：

main.py：

```
""" Main page using webpy 0.3 """
import web
import model

### Url mappings

urls = (
    '/', 'Index',
    '/view/(\d+)', 'View',
    '/signup', 'Signup',
    '/delete/(\d+)', 'Delete',
    '/edit/(\d+)', 'Edit',
    '/visit', 'Visit',
    '/editvisitor/(.+)/(\d+)', 'EditVisitor',
    '/deletevisitor/(.+)/(\d+)', 'DeleteVisitor',
    '/bookhouse', 'Book',
    '/houseid/(\d+)', 'BookEdit',
    '/bookhouse/succeed', 'Succeed',
    '/bookhouse/fail', 'Fail',
    '/orderinfo/(\d+)', 'OrderInfo',
    '/noorder', 'NoOrder',
    '/deleteorder/(\d+)', 'DeleteOrder',
    '/agencies', 'Agencies',
    '/agency/(\d+)', 'Agency',
    '/delete/agency/(\d+)', 'DeleteAgency',
    '/houses', 'House',
    '/delete/aghashouse/(\d+)', 'Delaghashouse',
    '/edithouse/(\d+)', 'Edithouse',
    '/deletehouse/(\d+)', 'DeleteHouse',
)



### Templates
t_globals = {
    'datestr': web.datestr
```

```
}
render = web.template.render('/home/henry/database/webpy/cs357proj/templates',
base='base', globals=t_globals)

class House:
    form = web.form.Form(
        web.form.Textbox('price'),
        web.form.Textbox('type'),
        web.form.Textbox('year'),
        web.form.Textbox('locationHouse'),
        web.form.Textbox('capacity'),
        web.form.Button('Add New House'),
        validators = [web.form.Validator("location is same as capacity", lambda i:
i.locationHouse != i.capacity)]
        )
    def GET(self):
        form=self.form()
        houses = model.get_Allhouses()
        return render.allhouses(houses, form)
    def POST(self):
        form=self.form()
        houses = model.get_Allhouses()
        if not form.validates():
            return render.allhouses(houses,form)
        model.new_house(form.d.price,          form.d.type,          form.d.year,
form.d.locationHouse, form.d.capacity)
        raise web.seeother('/houses')
class Edithouse:
    def GET(self, hsid):
        form=House.form()
        house=model.get_house(hsid)
        form.fill(house)
        return render.editindihouse(house, form)
    def POST(self, hsid):
        form=House.form()
        house=model.get_house(hsid)
        if not form.validates():
            return render.editindihouse(house, form)
        model.update_house(hsid,    form.d.price,    form.d.type,    form.d.year,
form.d.locationHouse, form.d.capacity)
        raise web.seeother('/houses')

class DeleteHouse:
    def POST(self, hsid):
```

```
        model.del_house(hsid)
        raise web.seeother('/houses')
class Edit:

    def GET(self, id):
        client = model.get_client(int(id))
        form = Signup.form()
        form.fill(client)
        return render.edit(client, form)

    def POST(self, id):
        form = Signup.form()
        client = model.get_client(int(id))
        if not form.validates():
            return render.edit(client, form)
        model.update_client(int(id), form.d.Name, form.d.password, form.d.Email,
form.d.Credit_Card_No, form.d.Phone_No, form.d.SEX)
        raise web.seeother('/')

class Agencies:
    form = web.form.Form(
        web.form.Textbox('Telephone'),
        web.form.Textbox('location'),
        web.form.Button('Add New Agency'),
        validators = [web.form.Validator("telephone.", lambda i: i.location !=
i.Telephone)]
    )
    def GET(self):
        form=self.form()
        agencies = model.get_agencies()
        return render.agencies(agencies, form)
    def POST(self):
        form = self.form()
        agencies = model.get_agencies()
        if not form.validates():
            return render.agencies(agencies,form)
        model.new_agency(form.d.Telephone, form.d.location)
        raise web.seeother('/agencies')

class Agency:
    form = web.form.Form(
        web.form.Textbox('Telephone'),
        web.form.Textbox('location'),
        web.form.Textbox('Choose_House'),
```

```
        web.form.Button('Edit'),
        validators = [web.form.Validator("telephone.", lambda i: i.location !=
i.Telephone)]
    )
    def GET(self, agid):
        agency = model.get_agency(agid)
        agencyhashouse = model.get_aghashouse(agid)
        print agencyhashouse
        form = self.form()
        form.fill(agency)
        return render.showAgency(agency, form, agencyhashouse)
    def POST(self, agid):
        form = self.form()
        agencyhashouse = model.get_aghashouse(agid)
        agency = model.get_agency(agid)
        if not form.validates():
            raise web.seeother('/agencies')
        if model.hsoccupied(form.d.Choose_House)==0:
            model.update_agency(agid, form.d.Telephone, form.d.location)
            model.agencywantshouse(agid, form.d.Choose_House)
        raise web.seeother('/agencies')


class Index:

    def GET(self):
        clients = model.get_clients()
        return render.index(clients)

class View:
    def GET(self, id):
        client = model.get_client(id)
        return render.view(client)

class Succeed:
    def GET(self):
        return render.succeed()
class Fail:
    def GET(self):
        return render.fail()
class NoOrder:
    def GET(self):
        return render.noorder()
class Signup:
```

```
    form = web.form.Form(
        web.form.Textbox('Name'),
        web.form.Password('password'),
        web.form.Password('password_again'),
        web.form.Textbox('Email'),
        web.form.Textbox('Credit_Card_No'),
        web.form.Textbox('Phone_No'),
        web.form.Dropdown('SEX', ['m', 'f']),
        web.form.Button('Submit'),
        validators = [web.form.Validator("Passwords didn't match.", lambda i:
i.password == i.password_again)]
    )

    def GET(self):
        form = self.form()
        return render.signup(form)

    def POST(self):
        form = self.form()
        if not form.validates():
            return render.signup(form)
        if model.exist(form.d.Name) == 0:
            model.new_client(form.d.Name,     form.d.password,     form.d.Email,
form.d.Credit_Card_No, form.d.Phone_No, form.d.SEX)
        raise web.seeother('/')

class Delete:
    def POST(self, id):
        model.del_client(int(id))
        raise web.seeother('/')

class DeleteAgency:
    def POST(self, id):
        model.del_agency(id)
        raise web.seeother('/agencies')

class DeleteVisitor:
    def POST(self, name, date):
        a = model.del_visitor(name, date)
        raise web.seeother('/visit')

class Delaghashouse:
    def GET(self, idhs):
```

```
        model.del_aghashouse(idhs)
        raise web.seeother('/agencies')

class DeleteOrder:

    def POST(self, hsid):
        model.order_delete(hsid)
        raise web.seeother('/bookhouse')

class EditVisitor:
    def GET(self, name, date):
        visitor = model.get_visitor(name)
        form = Visit.form()
        form.fill(visitor)
        return render.editvisitor(visitor, form)
    def POST(self, name, date):
        form = Visit.form()
        client = model.get_visitor(name)
        if not form.validates():
            return render.editvisitor(client, form)
        if model.exist(form.d.Name)==1:
            ag=model.get_agencyID(form.d.HouseID)
            print ag
            hs=model.get_house(form.d.HouseID)
            print hs
            model.del_visitor(name,date)
            model.new_visit(form.d.Name,    form.d.Visit_Date,    form.d.HouseID,
hs.locationHouse, ag.Agency_idAgency)
        raise web.seeother('/visit')

class BookEdit:
    def GET(self, hsID):
        form = Book.form()
        house = model.get_house(hsID)
        form.fill(house)
        return render.editHouse(house, form)
    def POST(self, hsID):
        form = Visit.form()
        house = model.get_house(hsID)
        if not form.validates():
            return render.editHouse(house, form)

class OrderInfo:
    def GET(self, hsID):
```

```
        form = Book.form()
        order = model.get_order(hsID)
        form.fill(order)
        if order!=None:
            return render.orderinfo(order, form)
        else:
            raise web.seeother('/noorder')
    def POST(self, hsID):
        form = Book.form()
        if not form.validates():
            return render.house(form)
        if model.existHs(form.d.idHouse)==1 and model.exist(form.d.Name)==1 and
model.hs_booked(form.d.Name, form.d.idHouse)==1:
            model.order_update(form.d.Name, form.d.idHouse, form.d.Duration)
            raise web.seeother('/bookhouse/succeed')
        else:
            raise web.seeother('/bookhouse/fail')

class Visit:
    form = web.form.Form(
        web.form.Textbox('Name'),
        web.form.Textbox('Visit_Date'),
        web.form.Textbox('HouseID'),
        web.form.Button('Submit'),
    )

    def GET(self):
        form = self.form()
        visitors = model.get_visitors()
        return render.visit(form, visitors)
    def POST(self):
        form = self.form()
        if not form.validates():
            return render.visit(form)
        if model.exist(form.d.Name)==1 and model.existHs(form.d.HouseID):
            ag=model.get_agencyID(form.d.HouseID)
            hs=model.get_house(form.d.HouseID)
            model.new_visit(form.d.Name,    form.d.Visit_Date,    form.d.HouseID,
hs.locationHouse, ag.Agency_idAgency)
        raise web.seeother('/visit')
class Book:
    form = web.form.Form(
        web.form.Textbox('Name'),
        web.form.Textbox('idHouse'),
```

```
        web.form.Textbox('Duration'),
        web.form.Button('Book it!'),
    )
    def GET(self):
        form = self.form()
        houses = model.get_houses()
        return render.house(form, houses)
    def POST(self):
        form = self.form()
        if not form.validates():
            return render.house(form)
        if model.existHs(form.d.idHouse)==1 and model.exist(form.d.Name)==1 and
model.hs_booked(form.d.Name, form.d.idHouse)==0:
            houseinfo = model.get_house(form.d.idHouse)
            cost = houseinfo.price * int(form.d.Duration)
            model.house_book(form.d.Name, form.d.idHouse, form.d.Duration, cost)
            raise web.seeother('/bookhouse/succeed')
        else:
            raise web.seeother('/bookhouse/fail')

app = web.application(urls, globals())

if __name__ == '__main__':
    app.run()
```

model.py:

```
import web, datetime

db = web.database(dbn='mysql', db='mydb_rent', user='root', pw='123456')
def existHs(hsid):
    try:
        db.select('House', where="idHouse=$hsid", vars=locals())[0]
        return 1
    except IndexError:
        return 0
def exist(name):
    try:
        db.select('Client', where="username=$name", vars=locals())[0]
        return 1
    except IndexError:
        return 0
```

```
def hsoccupied(hsid):
    try:
        db.select('Agency_has_House', where='idHouse=$hsid', vars=locals())[0]
        return 1
    except Exception:
        return 0
def new_agency(tele, location):
    AgencyID = 0
    agencies = get_agencies()
    for agency in agencies:
        if agency.idAgency >= AgencyID:
            AgencyID = agency.idAgency +1
    db.insert('Agency',                                    idAgency=AgencyID,
telephone=tele,locationAgency=location)

def get_agencies():
    return db.select('Agency', order='idAgency DESC')
def update_agency(agid, tel, loca):
    db.update('Agency',          where='idAgency=$agid',          telephone=tel,
locationAgency=loca, vars=locals())
def get_agency(agid):
    try:
        return db.select('Agency', where='idAgency=$agid', vars=locals())[0]
    except IndexError:
        return None

def get_agencyID(houseID):
    return        db.select('Agency_has_House',        where="idHouse=$houseID",
vars=locals())[0]
def get_aghashouse(agid):
    return     db.select('Agency_has_House',     where='Agency_idAgency=$agid',
vars=locals())

def del_agency(agid):
    try:
        db.delete('Agency', where="idAgency=$agid", vars=locals())
        return 1
    except Exception:
        return 0
def del_aghashouse(hsid):
    try:
        db.delete('Agency_has_House', where="idHouse=$hsid", vars=locals())
    except Exception:
        return None
```

```
def new_client(Name, Password, Email, Credit_Card_No, Phone, Gender):
    clients = get_clients()
    Clientid = 0
    for client in clients:
        if client.idClient >= Clientid:
            Clientid = client.idClient + 1
    db.insert('Client', username=Name, email=Email, password=Password,
        created_time=datetime.datetime.utcnow(), credit_no=Credit_Card_No,
        gender=Gender, idClient=Clientid)
def del_client(id):
    db.delete('Client', where="idClient=$id", vars=locals())

def update_client(id, Name, Password, Email, Credit_Card_No, Phone, Gender):
    del_client(id)
    db.insert('Client', username=Name, email=Email, password=Password,
        created_time=datetime.datetime.utcnow(), credit_no=Credit_Card_No,
        gender=Gender, idClient=id)
def get_clients():
    return db.select('Client', order='idClient DESC')

def get_client(id):
    try:
        return db.select('Client', where='idClient=$id', vars=locals())[0]
    except IndexError:
        return None

def visit_exsit(name):
    try:
        db.select('Visit', where="Client_username=$name", vars=locals())[0]
        return 1
    except IndexError:
        return 0

def new_visit(name, date, houseID, loca, agid):
    db.insert('Visit',        Client_username=name,        Agency_idAgency=agid,
House_idHouse=houseID, dateVisit=date, locationHouse=loca)

def get_visitor(username):
    try:
        return       db.select('Visit',       where='Client_username=$username',
vars=locals())[0]
    except IndexError:
```

```
        return None
def get_visitors():
    return db.select('Visit', order='House_idHouse DESC')

def update_visitor(oldname, newname, olddate, newdate, loca, houseID, agid):
    del_visitor(oldname, olddate)
    new_visit(newname, newdate, houseID, loca, agid)

def del_visitor(name, date):
    db.delete('Visit',   where="Client_username=$name   and   dateVisit=$date",
vars=locals())



def get_house(keyID):
    return db.select('House', where='idHouse=$keyID', vars=locals())[0]
def get_houses():
    return db.select('Agency_has_House', order='idHouse')

def house_book(name, hsid, length, cost):
    db.update('House', where='idHouse=$hsid', ordered_status=1, vars=locals())
    print length
    db.insert('Client_rents_House',     Client_username=name,     idHouse=hsid,
duration=length, startDate=datetime.datetime.utcnow(), cost=cost)

def hs_booked(name, hsid):
    try:
        db.select('Client_rents_House',     where="Client_username=$name     AND
idHouse=$hsid", vars=locals())[0]
        return 1
    except IndexError:
        return 0

def get_order(houseID):
    try:
        return     db.select('Client_rents_House',     where="idHouse=$houseID",
vars=locals())[0]
    except IndexError:
        return None
def order_delete(hsid):
    db.update('House', where='idHouse=$hsid', ordered_status=0, vars=locals())
    db.delete('Client_rents_House', where="idHouse=$hsid", vars=locals())

def order_update(name, hsid, length):
    order_delete(hsid)
```

```
    house_book(name, hsid, length)

def agencywantshouse(agid, hsid):
    try:
        db.insert('Agency_has_House', Agency_idAgency = agid, idHouse = hsid)
    except Exception:
        return None

def get_Allhouses():
    return db.select('House', order='idHouse')
def new_house(price, type, year, loca, capacity):
    houses = get_Allhouses()
    hsid = 0
    for house in houses:
        if house.idHouse >= hsid:
            hsid = house.idHouse+1
    db.insert('House', idHouse=hsid, price=price, ordered_status = 0, type=type,
year=year,locationHouse=loca, cap = capacity)
def update_house(houseid, price, type, year, loca, capacity):
    db.update('House', where="idHouse=$houseid", price=price, type=type, year
=year, locationHouse=loca, cap = capacity, vars=locals())
def del_house(hsid):
    try:
        db.delete('House', where="idHouse=$hsid", vars=locals())
    except Exception:
        return None
```

SQL:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `mydb_rent` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `mydb_rent` ;

-- -----------------------------------------------------
-- Table `mydb_rent`.`Client`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`Client` ;
CREATE TABLE IF NOT EXISTS `mydb_rent`.`Client` (
  `username` VARCHAR(16) NOT NULL,
  `email` VARCHAR(255) NULL,
  `password` VARCHAR(32) NOT NULL,
  `created_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
  `credit_no` BIGINT NULL,
  `idClient` INT,
  `gender` CHAR NULL,
  PRIMARY KEY (`username`));


-- -----------------------------------------------------
-- Table `mydb_rent`.`House`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`House`;
CREATE TABLE IF NOT EXISTS `mydb_rent`.`House` (
  idHouse INT NOT NULL,
  `ordered_status` TINYINT(1) NOT NULL,
  `price` FLOAT NOT NULL,
  `type` VARCHAR(45) NULL,
  `year` INT NULL,
  `locationHouse` VARCHAR(45) NOT NULL,
  `cap` VARCHAR(45) NULL,
  PRIMARY KEY (idHouse))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb_rent`.`Agency`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`Agency`;
CREATE TABLE IF NOT EXISTS `mydb_rent`.`Agency` (
  idAgency INT NOT NULL,
  `telephone` VARCHAR(11) NOT NULL,
  `locationAgency` VARCHAR(45) NOT NULL,
  PRIMARY KEY (idAgency))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb_rent`.`Agency_has_House`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`Agency_has_House`;
CREATE TABLE IF NOT EXISTS `mydb_rent`.`Agency_has_House` (
  Agency_idAgency INT NOT NULL,
  idHouse INT NOT NULL,
  PRIMARY KEY (Agency_idAgency, idHouse),
  INDEX `fk_Agency_has_House_House1_idx` (idHouse ASC),
  INDEX `fk_Agency_has_House_Agency1_idx` (Agency_idAgency ASC),
```

```
  CONSTRAINT `fk_Agency_has_House_Agency1`
    FOREIGN KEY (Agency_idAgency)
    REFERENCES `mydb_rent`.`Agency` (idAgency)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Agency_has_House_House1`
    FOREIGN KEY (idHouse)
    REFERENCES `mydb_rent`.`House` (`idHouse`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = dec8;


-- -----------------------------------------------------
-- Table `mydb_rent`.`Client_rents_House`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`Client_rents_House`;
CREATE TABLE IF NOT EXISTS `mydb_rent`.`Client_rents_House` (
  `Client_username` VARCHAR(16) NOT NULL,
  `idHouse` INT NOT NULL,
  `duration` VARCHAR(45) NULL,
  `startDate` TIMESTAMP NULL,
  `cost` FLOAT NULL,
  PRIMARY KEY (`Client_username`, `idHouse`),
  INDEX `fk_Client_has_House_House1_idx` (`idHouse` ASC),
  INDEX `fk_Client_has_House_Client1_idx` (`Client_username` ASC),
  CONSTRAINT `fk_Client_has_House_Client1`
    FOREIGN KEY (`Client_username`)
    REFERENCES `mydb_rent`.`Client` (`username`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Client_has_House_House1`
    FOREIGN KEY (`idHouse`)
    REFERENCES `mydb_rent`.`House` (`idHouse`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);


-- -----------------------------------------------------
-- Table `mydb_rent`.`Visit`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb_rent`.`Visit` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb_rent`.`Visit` (
  `Client_username` VARCHAR(16) NOT NULL,
  `Agency_idAgency` INT NOT NULL,
  `House_idHouse` INT NOT NULL,
  `dateVisit` VARCHAR(20) NOT NULL,
  `locationHouse` VARCHAR(45) NULL);

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```