School of Computer Science and Enginnering

COMP 9900
Information Technology Project
2018, s2

Project Name: ?
Team Name: H2CG

Team Member:

Zhu CAO, z5099326
Yuemian GE, z5111647
Jiahao FENG, z5118650
Harvey Tan, z5147986

Introduction

Existing system

Airbnb and Booking.com, have been widely promoted with the development of such in the tourism industry, becoming well-known third-party business website platform in the recent years. Both of them provide a scalable and maintainable platform for hosts submitting available accommodations. These accommodations are normally used to obtain charge from tourists who book rooms for travelling. Fees from booking normally not only belongs to hosts who provide rooms, but also offer a part of payment to website for web pages modification and maintaining as well as database stability. As these websites are prevailing and convenient, people prefer to search accommodation information online rather than scheduling a consultation with any intermediary institutions.

Problems of Existing system

For Booking.com, it only focus on the resource of hotels and motels, has hardly any resource of apartments and houses. this situation sometimes leads to difficulty on booking of desolate places such as a remote town.

For Airbnb, opposite to Booking.com, almost all the resource in this websites are private accommodations, less hotels or motels making this website similar to a short-term rental system.

Above analysis shows a common problem in both Booking.com and Airbnb system: insufficient resource.

Furthermore, the biggest risk for hosts in Airbnb is that their property will get damaged. While most transactions occur without incident, there are stories of entire houses being trashed by dozens of party-goers when the Airbnb hosts thought they were renting to a quiet suburban family, or an instance when a host came home to find his property had been damaged, items had been stolen and the place was littered with meth pipes.

Solution

According to the problems of existing system, diversified type of accomodations can be submitted to H2CG.

In addition, the host will be notified with the habits and preferences of the person booking the rooms in the host's accommodation. So the host can take the necessary precautions.

Differences

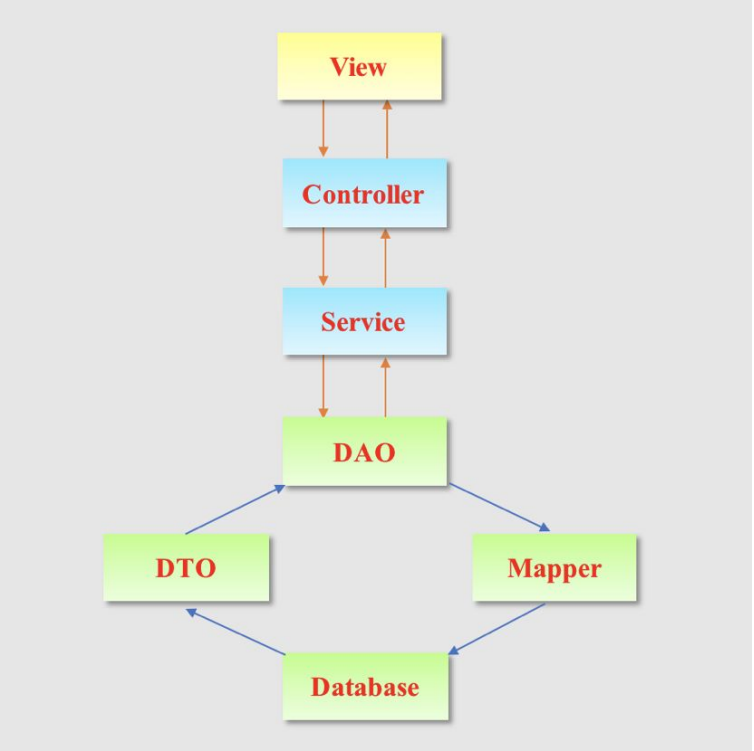
H2CG is catered for people who wants to book single room, shared accommodation. H2CG search provides the cheapest available accommodation by default. Moreover, H2CG accepts source from both private accommodations and public residences. Tourists could book a hotel room for only several nights as well as rent a house for a long-term holiday. As for shared accommodation, customer can see the habits and expectations of tenants, so they have a rough idea of their accommodation mates.

Structure of Report

Background

1. Usage scenario
    1) Users of this system:
    The users of H2CG are mainly divided into two parts: hosts and customers.
    Hosts: The owners of private accommodations such as apartments and houses. And also the owners of public residences such as hotels and motels.
    Customers: mainly are tourists. People travelling to a place for business purpose or for a holiday as well as visiting relatives and friends.
    2) Processing users requests:
    For hosts, after submitting materials and photos about accommodations and details of rooms. These informations will stored in database of this system.
    For customers, after registering personal informations, customers can search available rooms based on searching conditions. After select satisfying rooms, customers could submit habits and comments. All of these information, including room booking and habits as well as comments, will be also stored in database. The database will update automatically and the modified information will shown in the booking list of the web page as well as the advertise page of the website.

System Architecture



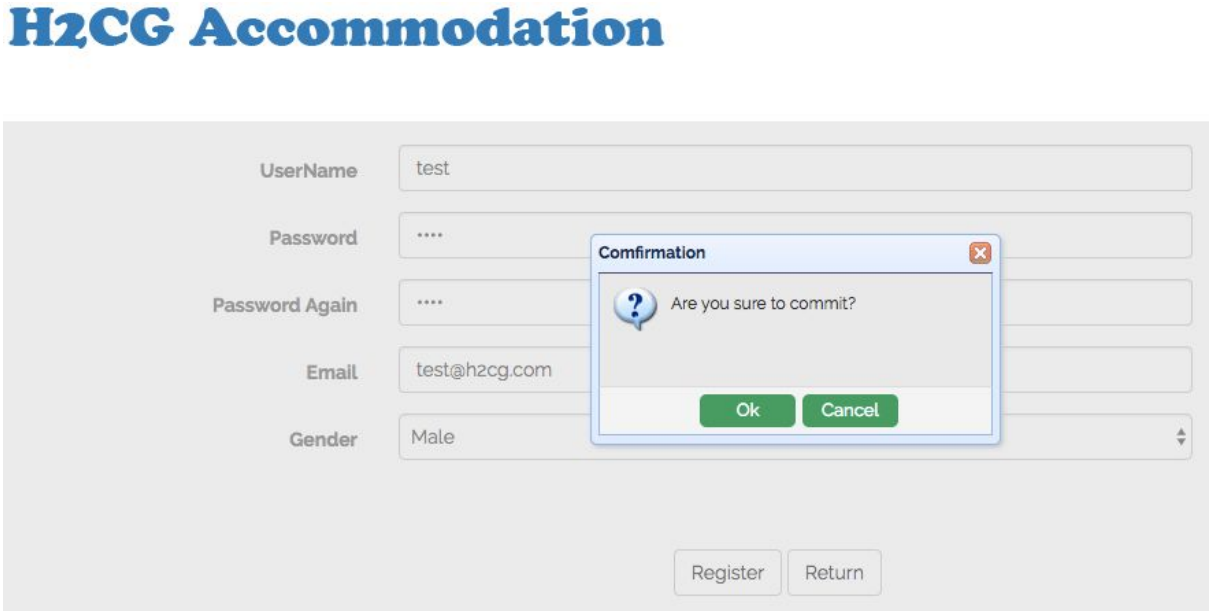
Model-view-controller (MVC) is a modern architectural pattern that divides the project into three interconnected parts. The MVC allows for efficient code reuse and parallel development. This design pattern has lots of benefits once used in a group project. The main components are DAO, DTO, Service, Mapper, Controller, View and Database.

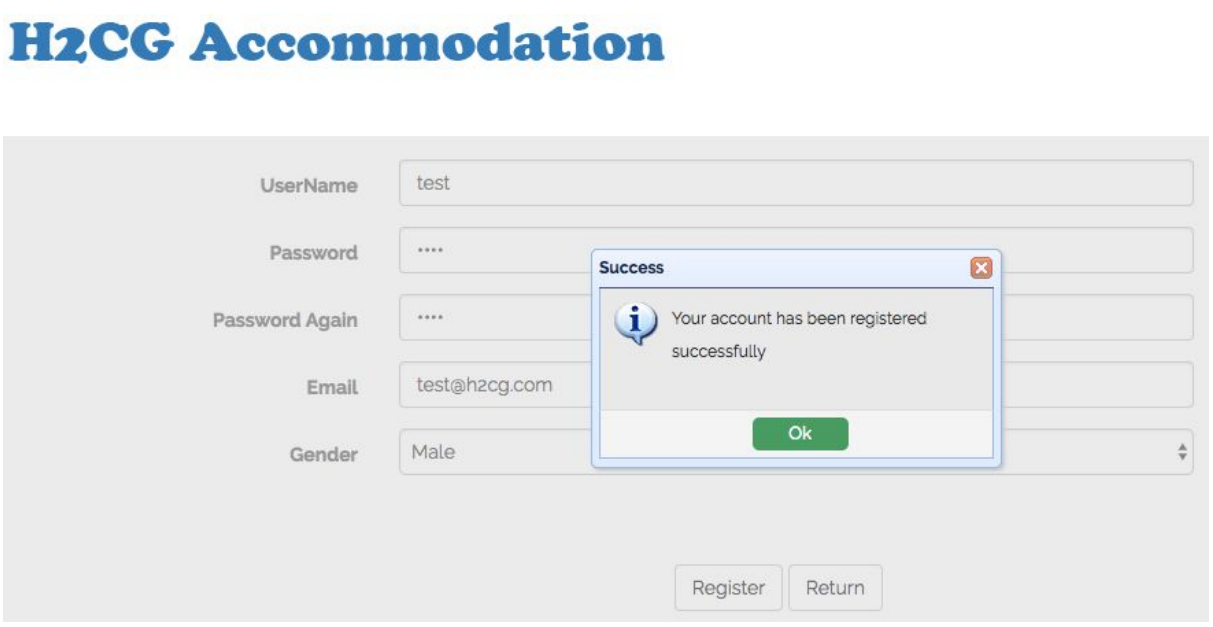
Account Component

1. Register

Required inputs are Username, Password, Email and Gender. Upon receiving inputs, it will INSERT into the database. If INSERT is successful, the system returns a success alert.

Success case example: -

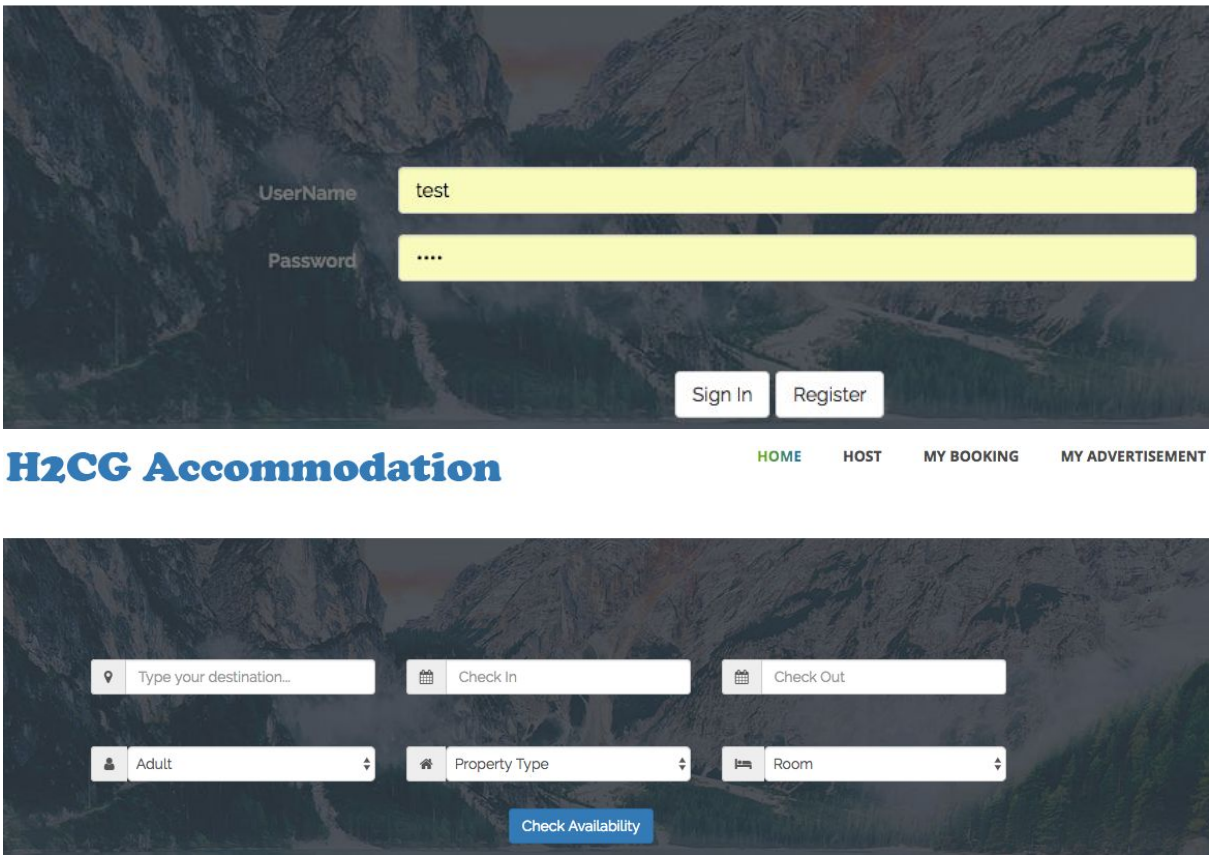




2. Login

Required inputs are Username and password. Upon receiving inputs, it will check credentials. If credentials exist and password matches, the system redirects to the home page.

Success case example: -

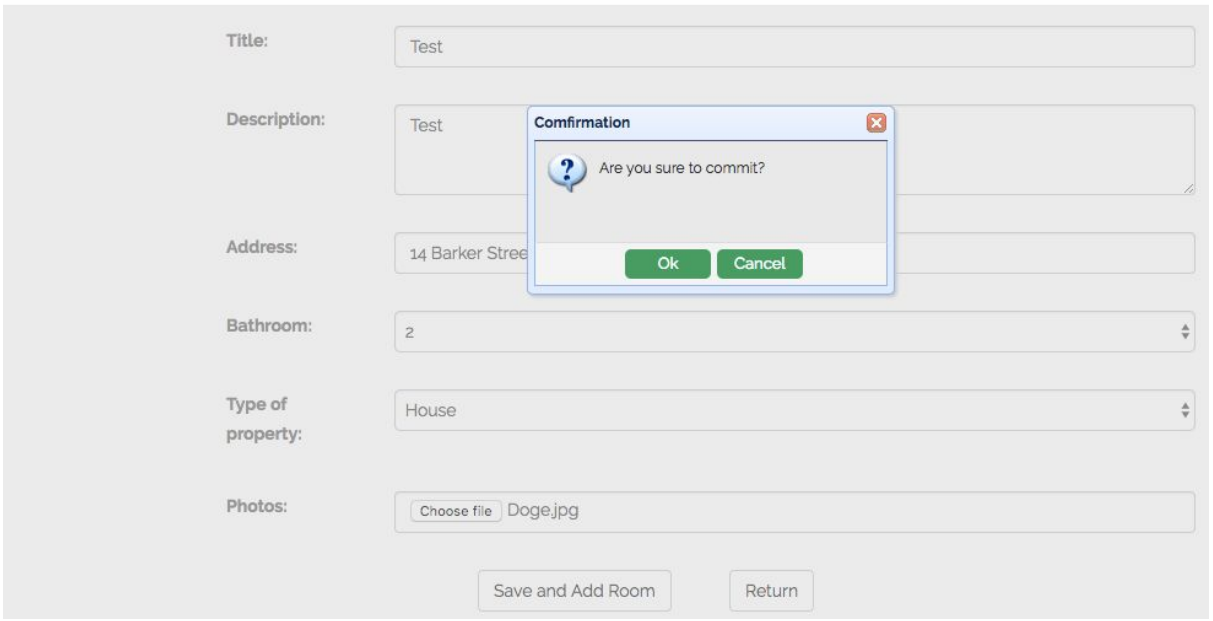


Click here to review the source code

Host Component

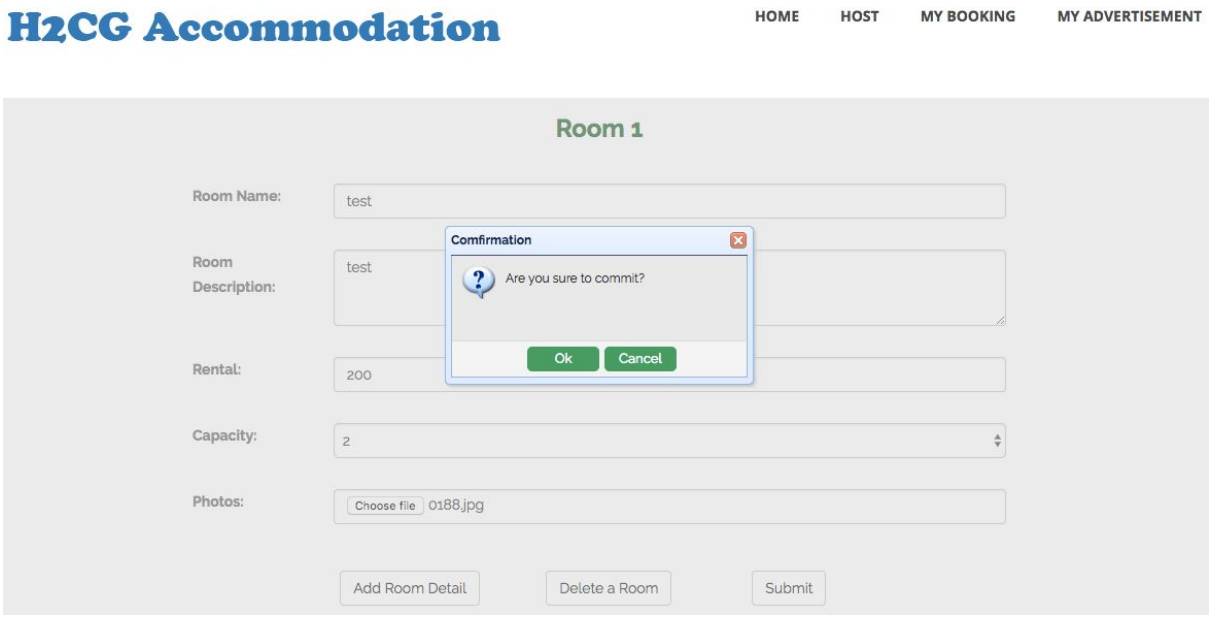
1. Add Advertisement

Required inputs are Title, Description, Address, Bathroom, Type of Property and Photos. Upon receiving input in the Address field, the system integrated google map api to autocomplete the address. This ensures the address is valid. If all the fields are filled, upon clicking the “save and add room” button, the system will redirect to add room page after successfully INSERT into the database.



2. Add Room(s)

Required inputs are Room Name, Room Description, Rental, Capacity and Photos.
Upon receiving inputs, the system will allow users to add more room when the "Add Room Detail" button is clicked.

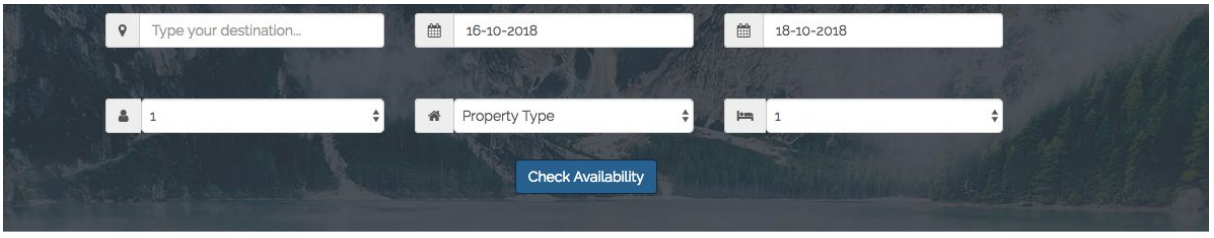


The submit button will redirect to the user's personal Advertisement management page.

[Click here to review the source code]

Search Component

The required inputs are Check In Date, Check Out Date, number of Adults and number of rooms. The optional inputs are destination (Suburb only) and Type of Property. Upon receiving required inputs, the system will return all the results which satisfy the user request, the system SELECT the data from the database (advertisement which has the lowest rental rate by default).



test

12 Barker Street, Kingsford NSW, Australia

test

House	1	2	1
Property Type	Bedrooms	Bathrooms	Capacity

Upon clicking the advertisement, the system will redirect to the advertisement that was clicked.

test

Rating:0.0/5

12 Barker Street, Kingsford NSW, Australia

test

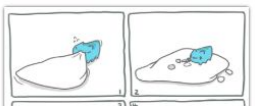
House	1	2	1
Property Type	Bedrooms	Bathrooms	Maximum number of people

16-10-2018 18-10-2018 Check Available Rooms

ROOM: TEST $200

test

BOOK NOW



[Click here to review the source code]

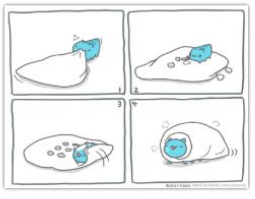
Book Component

Upon clicking the “BOOK NOW” button, the optional inputs are Habits and Expectation.

ROOM: TEST $200

test

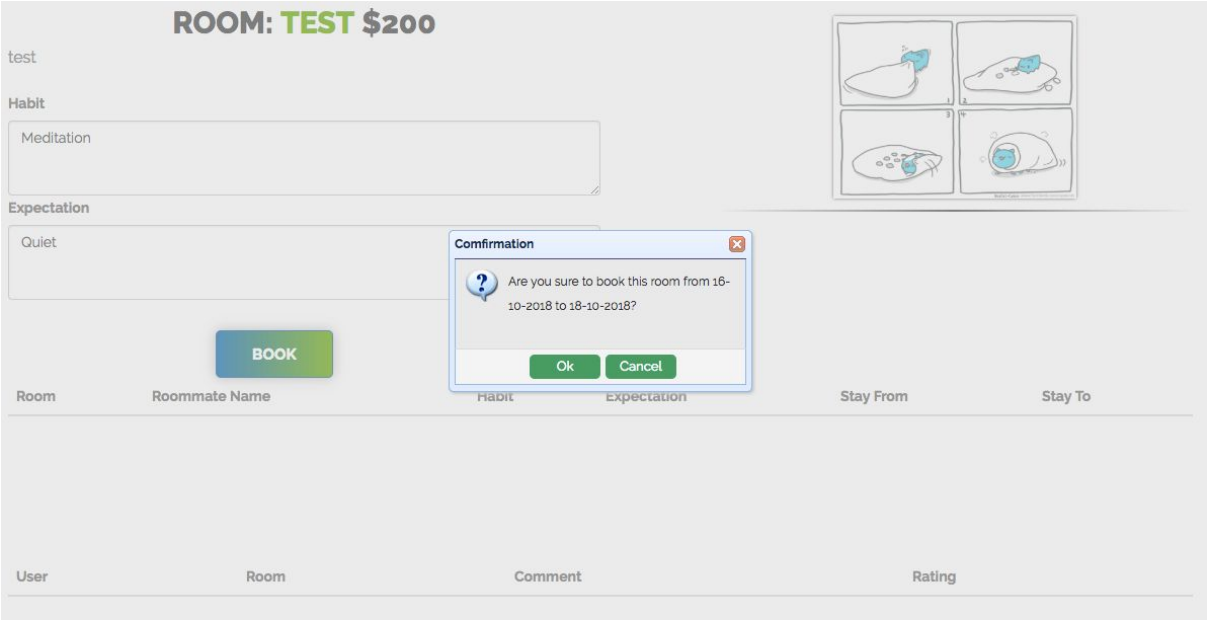
BOOK NOW



Room	Roommate Name	Habit	Expectation	Stay From	Stay To

User	Room	Comment	Rating

Upon clicking “BOOK” button, the system INSERT into the database.



After clicking ok, the page will refresh and the system will SELECT based on the advertisement id, then booking details is displayed.

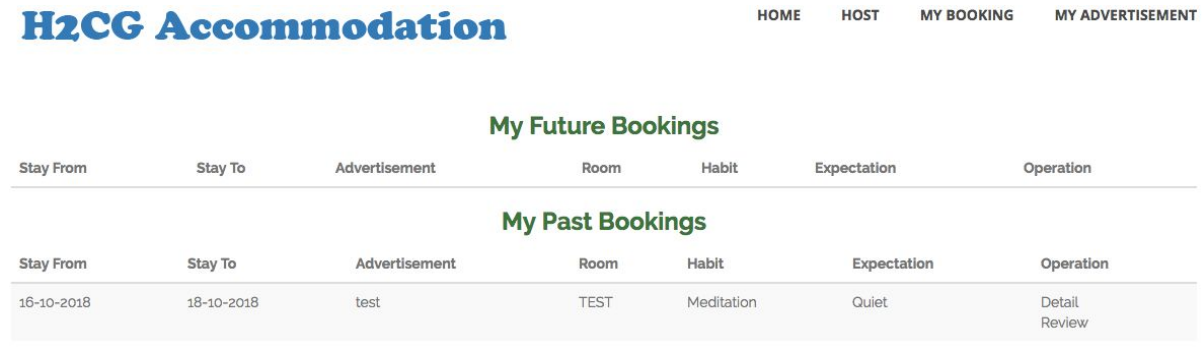
Room	Roommate Name	Habit	Expection	Stay From	Stay To
TEST	test	Meditation	Quiet	16-10-2018	18-10-2018

User	Room	Comment	Rating

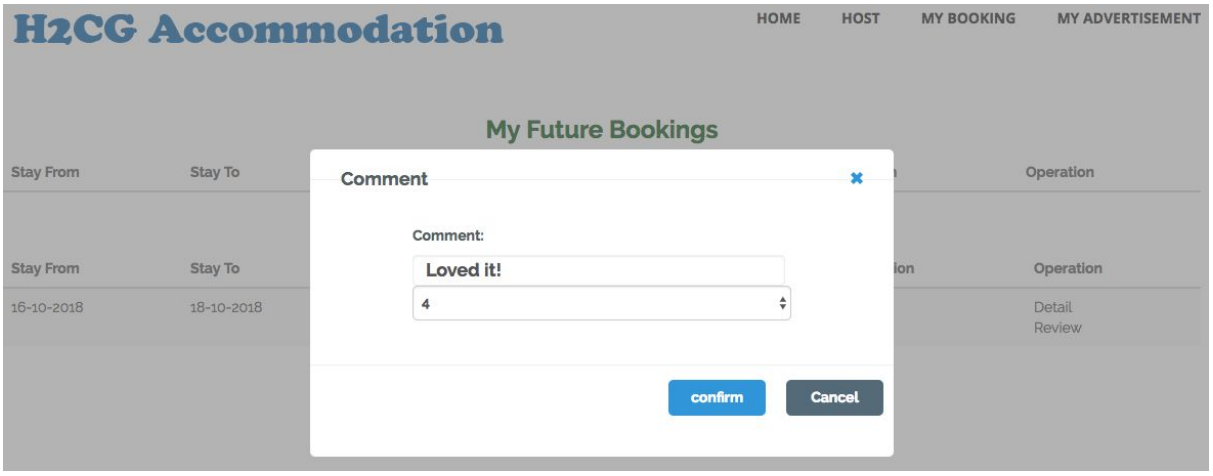
[Click here to review the source code]

Review/ Rating component

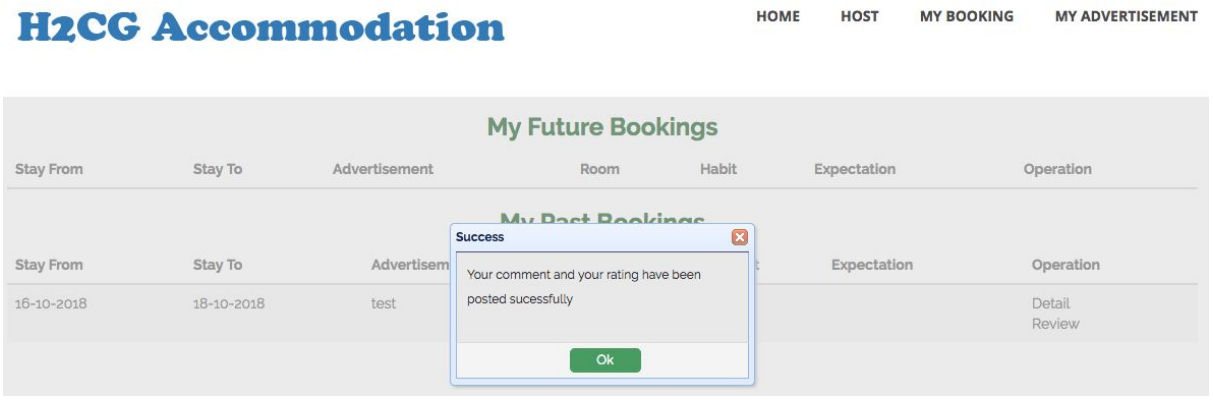
Upon clicking “MY BOOKING”, the system will display 2 section, “My Future Booking” and “My Past Booking”. The system SELECT the relevant data from the database.



Upon clicking “Review” at your past bookings under the section of Operation, user is able to leave feedback on the user’s experience living in the accommodation.



Upon confirming the review, a success alert will pop up.



To check if the review is successfully INSERTed, user can click on “Detail” under Operation to view the advertisement.

Room	Roommate Name	Habit	Expection	Stay From	Stay To
TEST	test	Meditation	Quiet	16-10-2018	18-10-2018

User	Room	Comment	Rating
test	TEST	Loved it!	4

[Click here to review the source code]

Conclusion

Tackle problems mentioned in component:

One of the problems dealt with during the whole project development is UI design. UI design is the most easy component but also a time-consuming part of the project. UI design cannot be finished at the beginning, because of every time a new function developed, the UI design need to be updated in order to show this new function. In addition, the format of the html is hard to adjust, the layout adjustment including color and shape of patterns costs an immense amount of time since there are a range of conditions need to be considered in design, for example the interests of customers, which styles of website customers prefer.

Another problem dealt with during the whole project development as well is database completion. there are two matters in database completion. One is the collection of data, this system is not used to usage in the reality, it is more likely an accomplish of an idea. Therefore, the datas in the database are not the actual informations, establish a unrealistic information in the database always cause problems. For example, when searching pictures on the internet, copyright of the pictures need to be considered carefully. To fix this problem, filter is essential, pictures with copyright have to be filtered out. Another matter is the Key Field of the database. The Key Field of the database need to be considered seriously, the incorrect of database design may cause serious problem. Key Field decide the tables, further decide the whole database. At the beginning, a wrong Key Field cause a incorrect selection of information, which cause a lot of time on debug the program. After figure out the problem in database, we check each Key in each table carefully, finally prevent the further mistake caused by Key.

Improvements

1. Integrating other databases to provide users to get the cheapest deal in accommodation
2. Integrating google route api to show the distance between scenic view, places with the accommodation.
3. Personalised recommendations for users. It will be based on factors that includes them filling up additional questions which are optional mainly focused on the style of accommodations that they fancy and also based on past bookings.
4. Recommendations of places around the accommodation. Displays personalised recommendations based on the time of day, displaying breakfast/ brunch places in the morning/afternoon, dinner places in the evening etc.

Appendix

Used Techniques

1. **IDE:** Eclipse
2. **Diary management:** BitBucket
3. **Scrum:** Trello
4. **Programming language:** Java 1.7, HTML, CSS
5. **Framework structure:** Spring, MyBatis
6. **Database:** MySQL

Installation manual

1. Install eclipse
2. Install tomcat
https://www.programmergate.com/step-by-step-guide-for-installing-tomcat-on-windows/
3. Install maven
http://www.vogella.com/tutorials/EclipseMaven/article.html#copyright-and-license
4. Import exist maven project

Once all of above have been installed successfully, click file->Import->Existing Maven Project to import our project.

Source Code Structure

1. Account Component

   1) The frontend design
   userLoginView.jsp
   userRegister.jsp
   These two jsp files are designed to display the login page and register page.
   The Backend design

   `function addUser()`
   This javascript function is used to add a new user. If success, it will redirect to the userLoginView page.

   2) The backend design
   UserController.java_`userLogin`

```
public String userLogin(@ModelAttribute UsersDTO userDto,
Model model, HttpSession session, @RequestParam(value =
"backUrl", required = false) String url)
```

This function first check whether the current session is available. If there exists a user, it redirects to search page. Otherwise it redirects to the login page.

UserService.java_`loginCheck`
UserDAO.java_`selectByNameAndPassword`

```
public UsersDTO loginCheck(UsersDTO UsersDTO);
UsersDTO selectByNameAndPassword(@Param("userName")String
userName, @Param("pwd")String pwd);
```

The first function receives values from `selectByNameAndPassword` in `userDAO.java` and return this value.

UserMapper.xml

```
<select id="selectByNameAndPassword"
resultMap="BaseResultMap" parameterType="String">
```

This is a simple SQL. It chooses all values from table USERS according to user_name and password. Then it returns value to `selectByNameAndPassword` in `userDAO.java`.

2. Host Component

1) The frontend design
hostAdv.jsp

```
<input type="file" class="form-control" id="photo"
name="photo" placeholder="Please upload the photos"
onchange="uploadImage()"/>
<a href="javascript:validateAndCommit()" class="btn
btn-default">Save and Add Room</a>
```

This jsp file is used to display the page of hosting part, including title, description, address, selecting bathroom and update photos.

`function uploadImage()`
This javascript function is used to upload images. If it's not a "jpg" or "png" type, it will alert a warning.

`function validateAndCommit()`
This javascript function can make sure that all details are entered correctly. Then it posts the userId and address to `isAddrExist` function in `AdvController.java`.

`function addPost()`
If it cannot find a photo, it posts values to `hostAdvertisment` function in `AdvController.java`. Otherwise it posts values to `hostAdvertismentWithPhoto` function in `AdvController.java`.

```
function initAutocomplete()
function fillInAddress()
function geolocate()
```

The first function is to create the autocomplete object, restricting the search to geographical location types. The second one is to get the place details from the autocomplete object. The third function is used to bias the autocomplete object to the user's geographical location, as supplied by the browser's 'navigator.geolocation' object.

2) The backend design

AdvController.java_`isAddrExist`

```
public Map<String, Object>
isAddrExist(@ModelAttribute("advDto") AdvertiseDTO advDto)
```

This function gets values from `advService.checkAddrExist` and if it checks that the address exists, the result is “yes” and it uses `setTitleJson` to set correspond values. Finally, the json object is called by javascript in `hostAdv.jsp`.

AdvController.java_`hostAdvertisment`

AdvController.java_`hostAdvertismentWithPhoto`

```
public Map<String, Object> postAdvertisment
public Map<String, Object> hostAdvertismentWithPhoto
```

These two functions are similar. First they check whether a user have logged in. Then both of them get values from `advService.addAdv`, if it exists then it uses `setTitleJson` to set correspond values. Finally, the json object is called by javascript in `hostAdv.jsp`. The only difference is that the second function will save photos before the function `setTitleJson`.

AdvService.java

```
public AdvertiseDTO checkAddrExist
public int addAdv
```

It returns the value of function `insertSelective` in `advertiseDao`.
It returns the values of function `selectByAddr` in `advertiseDao`.

AdvMapper.xml

```
<insert id="insertSelective"
parameterType="com.h2cg.accommodation.dto.AdvertiseDTO"
keyProperty="id" useGeneratedKeys="true" >
```

This SQL insert all values into the `ADVERTISE` table.

```
<select id="selectByAddr" resultMap="BaseResultMap"
parameterType="com.h2cg.accommodation.dto.AdvertiseDTO">
```

This SQL chooses all details from the `ADVERTISE` table and returns the result to the `checkAddrExist` in `AdvService.java`.

3. Search Component

1) The frontend design

index.jsp

This jsp file is used to display the page of searching part, including destination, check in time, check out time, the property type and so on.

```
function validateAndCommit()
function searchAdv()
function scrollToLocation()
```

The first function is used to check whether all details are validated. If it is validated, it will call `searchAdv` function. In `searchAdv` function, it gets values from the `searchAdv` function in `AdvController.java` then displays the result at the bottom of the same page. The third function is scrolling down to the location of last “div” in the class “son-panel”.

2) The backend design

AdvController.java_`searchAdv`

```
public Map<String, Object> searchAdv(HttpServletRequest
request, @ModelAttribute("bookDto") BookDTO bookDto, Model
model)
```

This function gets result from `selectAdv` in `advService.java`. If the result is not null, then it uses `setListJson` to set correspond values and returns the jason to `index.jsp`.

AdvService.java_`selectAdv`

```
public List<AdvertiseDTO> selectAdv(BookDTO
bookDto,HttpServletRequest request)
```

It returns the value of function `selectAdv` in `advertiseDao`.

AdvMapper.xml_`selectAdv`

```
<select id="selectAdv" resultMap="BaseResultMap"
parameterType="com.h2cg.accommodation.dto.BookDTO">
```

This SQL choose all details from the table composed by Book, Room and Advertise.Here we use `LEFT JOIN` because all possible choices should be based on the table Advertise. And the `stay_begin` or `stay_end` should be in certain area. The result will be sent back to `selectAdv` in `AdvDAO.java`.

4. Book Component

1) The frontend design

advDetail.jsp

This jsp file realized the following functions. First, it displays the detailed information of the accommodation of all possible rooms that you can order. Then it displays the booking page. Following this part, this page displays the detailed information of the room that you have ordered along with the comments and ratings.

```
function validateAndCommit()
function searchRoom()
```

`function bookRoom(roomId)`

The first function is designed to redirect to the `toAdvDetail` function in `AdvController.java`. The second function works as searching rooms according to conditions entered by users. It receives values from function `searchRoom` in `AdvController.java` and then displays the values in the table. The third function is designed to book a room according to `roomId` and it posts values to `bookRoom` function in `advController.java`. If a user books a room successfully, it raises an alert.

2) The backend design

AdvController.java_`toAdvDetail`

```
public String toUserProfile(@ModelAttribute AdvertiseDTO
advDto, @ModelAttribute("bookDto") BookDTO bookDto,
@ModelAttribute RoomDTO roomDto, Model model,
HttpServletRequest request)
```

This function receives values from `selectAdvById` in `AdvService.java`, `selectHousemate` in `bookService.java` and `selectByAdvId` in `reviewService.java`. Finally it returns the values to the `advDetail.jsp`.

AdvController.java_`searchRoom`

```
public Map<String, Object> searchRoom(HttpServletRequest
request, @ModelAttribute("bookDto") BookDTO bookDto, Model
model)
```

This function receives values from `selectAvailableRoom` in `roomService.java`. If the number of available rooms is null, the result is failure and return to the json object. The jason object is called by javascript in `advDetail.jsp`.

AdvController.java_`bookRoom`

```
public Map<String, Object> bookRoom(@ModelAttribute BookDTO
bookDto, HttpSession session)
```

This function receives values from the function of `insertBooking` in `bookService.java`. If the value is bigger than 0, which means booking a room successfully and use `setTitleJson` to set values in `advDetail.jsp`. Finally it returns the json object.

BookService.java

```
public List<BookDTO> selectHousemate(BookDTO bookDto)
```

It receives values from `selectHousemate` in `bookDao.java` and the function `selectHousemate` in `bookDao.java` receives result from the SQL in `BookMapper.xml`.

BookMapper.xml

```
<select id="selectHousemate" resultMap="BaseResultMap"
parameterType="com.h2cg.accommodation.dto.BookDTO">
```

This SQL choose booking details from the table composed by Book, Room and Advertise.Here we use `LEFT JOIN` because all possible choices should be based on the table Book. And the booked `stay_end` should be bigger than `stayBegin`, the booked `stay_begin` should be smaller than `stayEnd`.

RoomService.java

```
public List<RoomDTO> selectAvailableRoom(BookDTO
DTO,HttpServletRequest request)
```

It receives values from `selectAvailableRoom` in `RoomDao.java` and the function `selectAvailableRoom` receives result from the SQL in `RoomMapper.xml`.

RoomMapper.xml

```
<select id="selectAvailableRoom" resultMap="BaseResultMap"
parameterType="com.h2cg.accommodation.dto.BookDTO">
```

This SQL choose all details from the table composed by Book and Room table. Here we use `LEFT JOIN` because all possible choices should be based on the table Book. And the `stay_begin` should be bigger than current data. All lists are ordered by `room_id`. The result will be sent to the roomList in `selectAvailableRoom`.

5. Review Component

1) The frontend design

bookingList.jsp

This part is used to create a table with several headings: Stay From, Stay to, Advertisement, Room and so on. Then it displays the values in the object `bookingListBeforeCheckIn` using a `c:forEach` loop. The value in `bookingListBeforeCheckIn` comes from the function `toBookingManagement` in `AdvController.java`.

```
<td><a
href="./toAdvDetail?id=${booking1.room.adv.id}">Detail</a>
<br><a
href="javascript:cancelBooking(${booking1.id})">Cancel</a></td
>
```

This part is to create two hyperlinks, the `Detail` is redirected to the `toAdvDetail` page with certain values and the `Cancel` is redirected to the javascript function `cancelBooking`.

```
function cancelBooking(id)
```

This javascript function works for canceling the order that the user hasn't check in. It first uses `$.messager.confirm` to raise a confirmation. Then it posts the current booking.id to the function `cancelBooking` in `AdvController.java`. If it deletes the room successfully, it will raise an alert and refresh current page.

```
function rating(id)
```

This javascript function works for adding comment on the room already checked in. It

2) The backend design

AdvController.java_`toBookingManagement`

```
public String toBookingManagement(@ModelAttribute
AdvertiseDTO advDto, Model model, HttpSession session)
```

This is the booking management function in controller layer. First it checks whether the current session is available. If not, it turns to the login page. Then it receives the DTO list from `selectBookingBeforeCheckIn` in `BookService.java` and uses `model.addAttribute` to post these data to the variable `${bookingListBeforeCheckIn}` mentioned before in `bookingList.jsp`.

AdvController.java_`cancelBooking`

```
public Map<String, Object>
cancelBooking(@RequestParam(required = false) Integer id)
throws IOException
```

This function gets values from `bookService.deleteBooking` and if it deletes a line successfully, the result is greater than 0 and it uses `setTitleJson` to set correspond values. Finally, the json object is called by javascript in `booingList.jsp`.

BookService.java_`selectBookingBeforeCheckIn`

```
public List<BookDTO> selectBookingBeforeCheckIn(Integer
userId, Date today)
```

This function receives values from `bookDao.selectBookingBeforeCheckIn` and uses `DateUtils.getDateString` to transform the `Timestamp` into `String`. Then the result is called by the function `toBookingManagement` in `AdvController.java`.

BookDAO.java

```
List<BookDTO> selectBookingBeforeCheckIn
List<BookDTO> selectBookingAfterCheckIn
int deleteBooking(Integer id);
```

The `BookDAO` defined three functions mentioned before and they are implemented in `BookMapper.xml`.

BookMapper.xml

```
<select id="selectBookingBeforeCheckIn"
resultMap="BaseResultMap"
parameterType="com.h2cg.accommodation.dto.BookDTO">
```

This SQL choose all details from the table composed by Book, Room and Advertise.Here we use `LEFT JOIN` because all possible choices should be based on the table Book. And the `stay_begin` should be bigger than current data. All lists are ordered by `stay_begin`.

References

https://en.wikipedia.org/wiki/Airbnb
https://en.wikipedia.org/wiki/Booking.com
https://www.booking.com
https://www.airbnb.com.au/
https://mvnrepository.com/artifact/org.springframework/spring-context
http://camel.apache.org/mybatis.html
https://www.w3schools.com/html/
https://www.w3schools.com/css/default.asp
https://www.w3schools.com/sql/sql_ref_mysql.asp
https://www.w3schools.com/js/
https://www.learnjavaonline.org/
https://www.websitebuilderexpert.com/how-to-build-a-website/
https://www.careeranna.com/articles/report-writing-format-sample-report/