Department of Computer Engineering

Bilkent University

# CS 353 Design Report

Prepared by
CS353 - Section 1 - Group 2

Abdullah Al Wali 21402793

Burak Mandıra 21301474

Burak Savlu 21202247

Gökhan Şimsek 21401407

## TABLE OF CONTENTS

## 1. PROJECT DESCRIPTION

In our project we aim to create an online accommodation system similar to that of AirBnB which we will call CnS (Crash and Sleep). The system that we propose will bring together travellers who want to rent a room or a house and the hosts who want to rent their houses or rooms. The system will do this by storing information about users, houses, rooms, offerings and user reviews.

Hosts will be able to make offerings to rent their houses or rooms, review the guests that they have accommodated, accept/refuse guests based on their ranking, withdraw their offers. Likewise, guests will be able to search for houses or rooms, according to the city and the dates between which they will be staying. Guests can also use certain filters that specify the room or house's quality such as number of beds, number of wardrobes, whether it has a private bathroom or not, whether there is a kitchen or not, availability of TV, Wifi, ethernet internet connection, dryer, iron, hangers, washers, free parking. The guests will also be able to see the rank of the host, and make their choice accordingly. After staying in a place, they will be able to review the room or the house alongside the host.

## 2. REVISED E/R MODEL

### 2.1. Changes Made to the Model

- New entity "Reservation" is created with attributes reservation_ID, reserve_start, reserve_end..
- A new relation "decides" between "Host" and newly created "Reservation" is also added. This allows host to confirm/deny the reservations made by the users.
- With the addition of the new entity "Reservation", "reserves" relation becomes "makes", a ternary relation. .
- New weak entity "Amenities" is created with attributes "number_of_bathrooms, wifi, internet, tv, kitchen, dryer, iron, hangers, cable_tv, bathtub, washer, free_parking". These attributes previously belonged to the "Accommodation" entity.

- New weak relation “contains” between “Accommodation” and weak entity “Amenities” is created.
- Added “password” attributes to “Account” entity.
- “Number_of_people” attribute has been added to “Accommodation”.
- “House” now only has “number_of_rooms” attribute. “Number_of_twin_beds” and “number_of_single_beds” attributes are deleted.
- All attributes from “Room” has been deleted. A single “number_of_beds” attribute has been added.
- Deleted the multivalued “address” attribute from “Account”.
- Added new cardinality constraints.
- Made changes to the general outline of the diagram (shapes, sharpened edges ).

## 2.2. Updated E/R Diagram

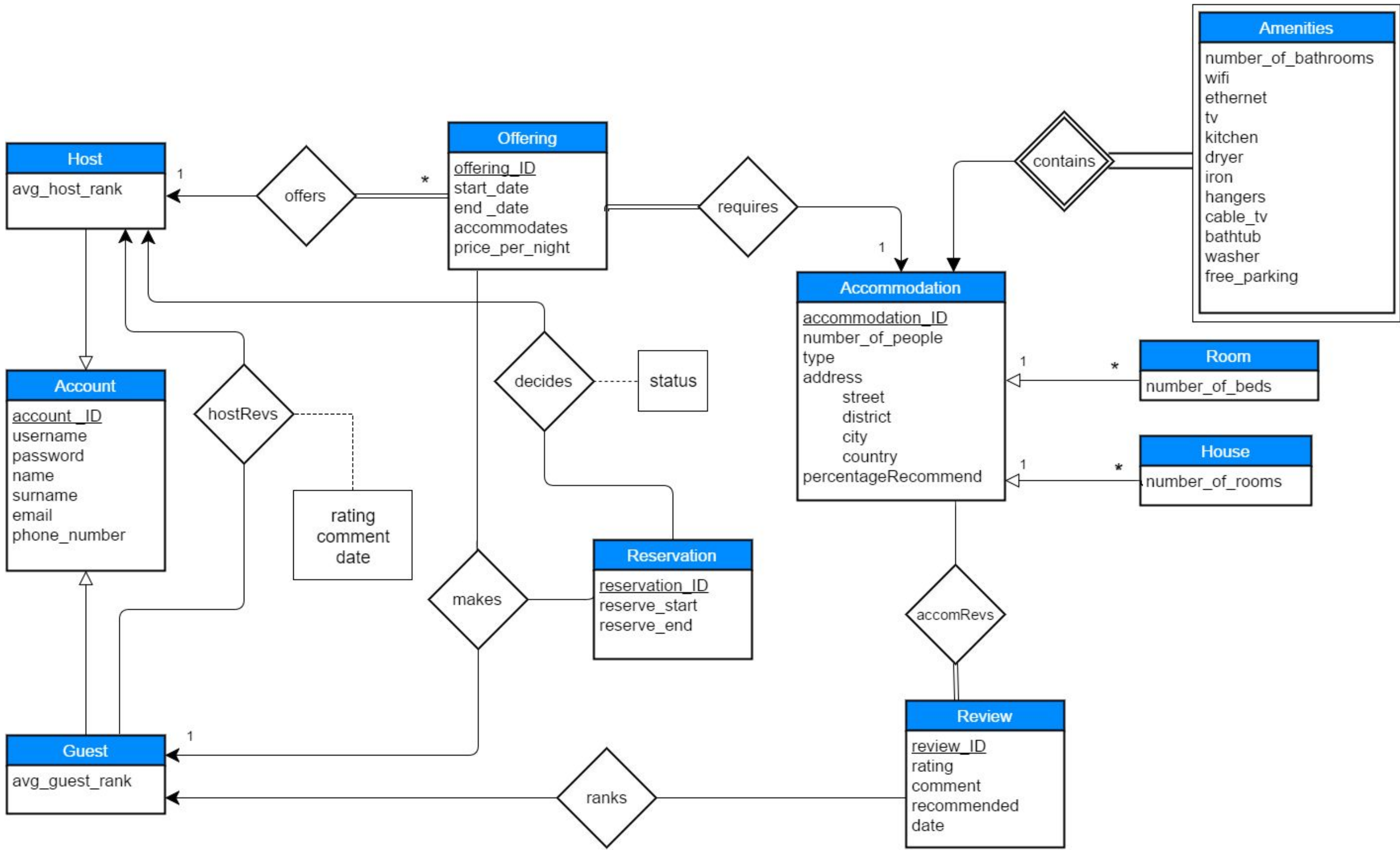


Figure 01: Updated ER Diagram

## 3. Relational Schemas

### 3.1. Account

**Relational Model:**

account(account_ID, password, name, surname, email, phone_number)

**Functional Dependencies:**

account_ID → password, name, surname, email, phone_number

email → account_ID

**Candidate Keys:**

{account_ID}, {email}

**Normal Form:**

3NF

**Table Definition:**

```
CREATE TABLE Account (
        account_ID              numeric(6,0) AUTO_INCREMENT,
        password                varchar(32) NOT NULL,
        name                    varchar(32) ,
        surname                 varchar(32),
        email                   varchar(32) NOT NULL,
        phone_number            character(15),
        PRIMARY KEY (account_ID),
        CHECK (email like ‘_%@__%._%’) ) ENGINE = InnoDB;
```

### 3.2. Host

**Relational Model:**

host(account_ID, avg_host_rank)

**Functional Dependencies:**

account_ID → avg_host_rank

**Candidate Keys:**

{account_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE host (
        account_ID                  numeric(6,0) NOT NULL,
        avg_host_rank               numeric(3,2) ,
        PRIMARY KEY (account_ID),
        FOREIGN KEY (account_ID) REFERENCES account(account_ID)) ENGINE =
InnoDB;
```

## 3.3. Guest

**Relational Model:**

guest(<u>account_ID</u>, avg_guest_rank)

**Functional Dependencies:**

account_ID → avg_guest_rank

**Candidate Keys:**

{account_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE guest (
        account_ID                  numeric(6,0) NOT NULL,
        avg_guest_rank              numeric(3,2),
        PRIMARY KEY (account_ID),
        FOREIGN KEY (account_ID) REFERENCES account(account_ID)) ENGINE =
InnoDB;
```

## 3.4. Offering

**Relational Model:**

offering(<u>offering_ID</u>, start_date, end_date, accommodates, price_per_night)

**Functional Dependencies:**

offering_ID → start_date, end_date, accommodates, price_per_night

**Candidate Keys:**

{offering_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE offering(
        offering_ID                 numeric(8,0) AUTO_INCREMENT,
        start_date                  date,
        end_date                    date,
        accommodates                integer,
        price_per_night             integer,
        PRIMARY KEY (offering_ID)) ENGINE = InnoDB;
```

## 3.5. Offers

**Relational Model:**

offers(<u>account_ID</u>, <u>offering_ID</u>)

**Functional Dependencies:**

No functional dependencies.

**Candidate Keys:**

{account_ID, offering_ID}

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE Account (
        account_ID                  numeric(6,0) NOT NULL,
        offering_ID                 numeric(8,0) NOT NULL,
        PRIMARY KEY (account_ID, offering_ID),
        FOREIGN KEY (account_ID) REFERENCES account(account_ID),
        FOREIGN KEY (offering_ID) REFERENCES offering(offering_ID)) ENGINE =
InnoDB;
```

### 3.6. Reservation

**Relational Model:**

reservation(reservation_ID, reserve_start, reserve_end)

**Functional Dependencies:**

reservation_ID → reserve_start, reserve_end

**Candidate Keys:**

{reservation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE reservation (
        reservation_ID              numeric(8,0) AUTO_INCREMENT,
        reserve_start               date,
        reserve_end                 date,
        PRIMARY KEY (reservation_ID)) ENGINE = InnoDB;
```

### 3.7. Makes

**Relational Model:**

makes(reservation_ID, account_ID, offering_ID)

**Functional Dependencies:**

reservation_ID → account_ID, offering_ID, status

**Candidate Keys:**

{reservation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE makes(
        reservation_ID              numeric(8,0) NOT NULL,
        account_ID                  numeric(6,0) NOT NULL,
        offering_ID                 numeric(8,0) NOT NULL,
        PRIMARY KEY(reservation_ID),
        FOREIGN KEY (reservation_ID) REFERENCES reservation(reservation_ID),
        FOREIGN KEY (account_ID) REFERENCES account(account_ID),
        FOREIGN KEY (offering_ID) REFERENCES offering(offering_ID)) ENGINE =
                                                                InnoDB;
```

## 3.8. Decides

**Relational Model:**

decides(<u>reservation_ID</u>, account_ID, status)

**Functional Dependencies:**

reservation_ID → account_ID, status

**Candidate Keys:**

{reservation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE decides(
        reservation_ID              numeric(8,0) NOT NULL,
        account_ID                  numeric(6,0) NOT NULL,
        status                      boolean,
        PRIMARY KEY(reservation_ID),
        FOREIGN KEY (reservation_ID) REFERENCES reservation(reservation_ID),
        FOREIGN KEY (account_ID) REFERENCES account(account_ID)) ENGINE =
                                                                    InnoDB;
```

## 3.9. Accommodation

**Relational Model:**

accommodation(<u>accommodation_ID</u>, number_of_people, type, street, district, city, country, percentageRecommend)

**Functional Dependencies:**

accommodation_ID → number_of_people, type, street, district, city, country, percentageRecommend

**Candidate Keys:**

{accommodation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE accommodation(
    accommodation_ID            numeric(8,0) AUTO_INCREMENT,
    number_of_people            integer,
    type                        boolean,
    street                      varchar(50),
    district                    varchar(30),
    city                        varchar(30),
    country                     varchar(30),
    percentageRecommend         numeric(3,2),
    PRIMARY KEY (accommodation_ID)) ENGINE = InnoDB;
```

## 3.10. Requires

**Relational Model:**

requires(<u>offering_ID,</u> accommodation_ID)

**Functional Dependencies:**

offering_ID → accommodation_ID

**Candidate Keys:**

{offering_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE requires (
        offering_ID                 numeric(8,0) NOT NULL,
        accommodation_ID            numeric(8,0) NOT NULL,
        PRIMARY KEY (offering_ID),
        FOREIGN KEY (offering_ID) REFERENCES offering(offering_ID),
        FOREIGN KEY (accommodation_ID) REFERENCES,
                        accommodation(accommodation_ID)); ENGINE = InnoDB;
```

## 3.11. Amenities

**Relational Model:**

amenities(<u>accommodation_ID</u>, number_of_bathrooms, wifi, ethernet, tv, kitchen, dryer, iron, hangers, cable_tv, bathtub, washer, free_parking)

**Functional Dependencies:**

accommodation_ID → number_of_bathrooms, wifi, ethernet, tv, kitchen, dryer, iron, hangers, cable_tv, bathtub, washer, free_parking

**Candidate Keys:**

{accommodation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE amenities(
        accommodation_ID        numeric(8,0) NOT NULL,
        number_of_bathrooms     integer,
        wifi                    boolean,
        ethernet                boolean,
        tv                      boolean,
        kitchen                 boolean,
        dryer                   boolean,
        iron                    boolean,
        hangers                 boolean,
        cable_tv                boolean,
        bathtub                 boolean,
        washer                  boolean,
        free_parking            boolean,
        PRIMARY KEY (accommodation_ID),
        FOREIGN KEY (accommodation_ID) REFERENCES
                        accommodation(accommodation_ID)) ENGINE = InnoDB;
```

## 3.12. Room

**Relational Model:**

room(accommodation_ID, number_of_beds)

**Functional Dependencies:**

accommodation_ID → number_of_beds

**Candidate Keys:**

{accommodation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE room (
        accommodation_ID            numeric(6,0) NOT NULL,
        number_of_beds              integer,
        PRIMARY KEY (accommodation_ID),
        FOREIGN KEY (accommodation_ID) REFERENCES,
                accommodation(accommodation_ID)) ENGINE = InnoDB;
```

## 3.13. House

**Relational Model:**

house(<u>accommodation_ID</u>, number_of_rooms)

**Functional Dependencies:**

accommodation_ID → number_of_rooms

**Candidate Keys:**

{accommodation_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE house (
        accommodation_ID            numeric(6,0) NOT NULL,
        number_of_rooms             integer,
        PRIMARY KEY (accommodation_ID),
        FOREIGN KEY (accommodation_ID) REFERENCES,
                accommodation(accommodation_ID)) ENGINE = InnoDB;
```

## 3.14. Review

**Relational Model:**

review(<u>review_ID</u>, rating, comment, recommended, date)

**Functional Dependencies:**

review_ID → rating, comment, recommended, date

**Candidate Keys:**

{review_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE review (
        review_ID               numeric(9,0) AUTO_INCREMENT,
        rating                  numeric(1,0) NOT NULL,
        comment                 varchar(300),
        recommended             numeric(4,2),
        date                    date,
        PRIMARY KEY (review_ID)) ENGINE = InnoDB;
```

### 3.15. AccomRevs

**Relational Model:**

accomRevs(review_ID, accommodation_ID)

**Functional Dependencies:**

review_ID → accommodation_ID

**Candidate Keys:**

{review_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE accomRevs (
        review_ID               numeric(9,0) NOT NULL,
        accommodation_ID        numeric(8,0) NOT NULL,
        PRIMARY KEY (review_ID),
        FOREIGN KEY (review_ID) REFERENCES review(review_ID),
          FOREIGN KEY (accommodation_ID) REFERENCES,
                    accommodation(accommodation_ID)) ENGINE = InnoDB;
```

## 3.16. Ranks

**Relational Model:**

ranks(<u>review_ID</u>, account_ID)

**Functional Dependencies:**

review_ID → account_ID

**Candidate Keys:**

{review_ID}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE ranks (
          review_ID                    numeric(9,0) NOT NULL,
          account_ID                   numeric(6,0) NOT NULL,
          PRIMARY KEY (review_ID),
          FOREIGN KEY (review_ID) REFERENCES review(review_ID),
          FOREIGN KEY (account_ID) REFERENCES account(account_ID))
                                                          ENGINE = InnoDB;
```

## 3.17. HostRevs

**Relational Model:**

hostRevs(<u>H.account_ID</u>, <u>G.account_ID</u>, rating, comment, date)

**Functional Dependencies:**

H.account_ID → G.account_ID

**Candidate Keys:**

{(H.account_ID, G.account_ID)}

**Normal Form:**

3NF, BCNF

**Table Definition:**

```
CREATE TABLE hostRevs (
        H.account_ID, G.account_ID,
        rating          numeric(2,1) NOT NULL,
        comment         varchar(300),
        date            datetime,
        PRIMARY KEY (H.account_ID, G.account_ID),
        FOREIGN KEY (H.account_ID) REFERENCES Host(account_ID),
        FOREIGN KEY (G.account_ID) REFERENCES Guest(account_ID)),
                                                        ENGINE = InnoDB;
```

# 4. FUNCTIONAL COMPONENTS

## 4.1. USE CASES/SCENARIOS

CnS has two users, the host and the guest, for which the use cases and scenarios are described as follows:

Hosts

- Hosts should be able to make an offering that specifies the qualities of the place they are willing to rent.
- Hosts should be able to add/remove offerings.
- Hosts should be able to change the qualities of the place they are offering.
- Hosts should be able to accept/decline the guests that want to make a reservation.
- Hosts should be able to view the ranks of the guests and view the reviews about them.
- Hosts should be able to rank the guests that they accommodated.

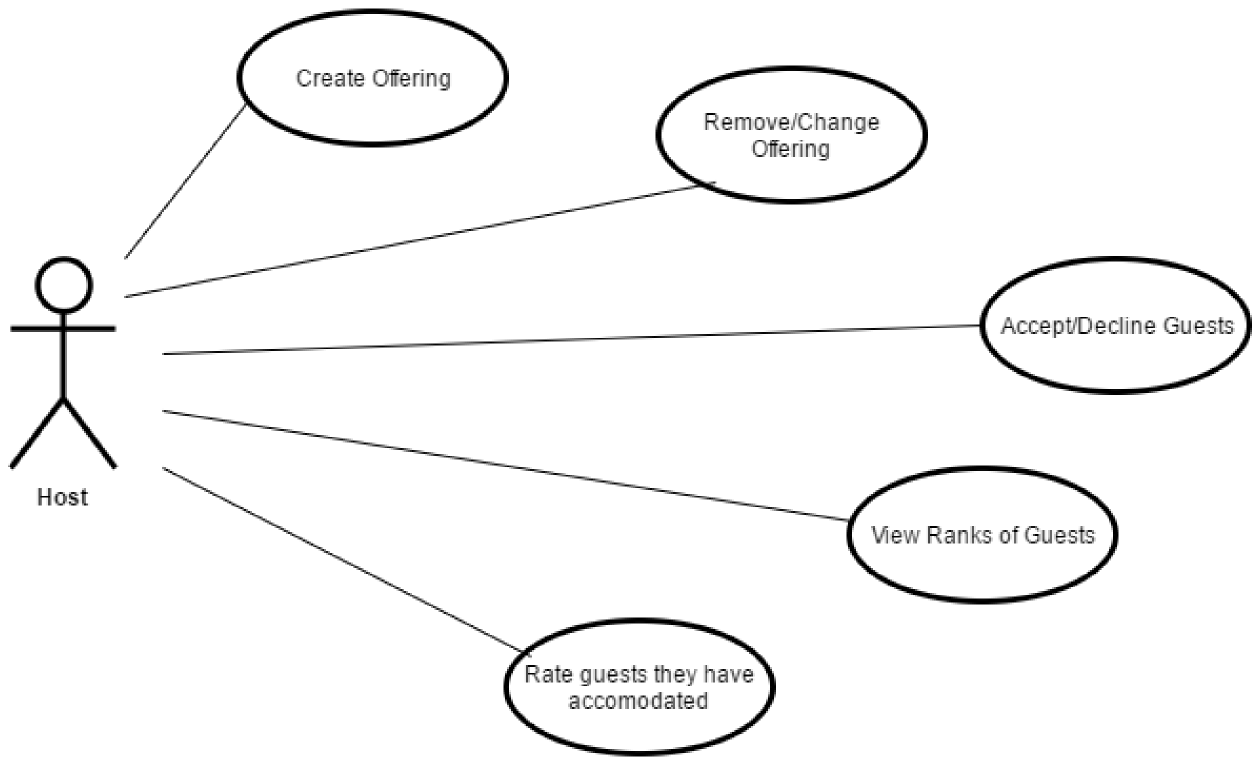


Figure 02: Host Use Case Diagram

### Guests

- Guests should be able to search for accommodation through the offerings made by the hosts, specifying the date and the cities.
- Guests should be able to use certain filters during their search, such as the type of the accommodation, e.g. a room or a house, wifi, ethernet connection, TV, availability of a kitchen, availability of items like dryer, iron, hangers, washer, and free parking.
- Guests should be able to view the rank of a certain host.
- Guests should be able to view the reviews of a certain place.
- Guests should be able to view the status of their reservations (rejection or approval).
- Guests should be able to make reservations.
- Guests should be able to rank and review their accommodation, with the review consisting of a rating, an optional description, optional pros and cons fields, and a “Would you Recommend this offering to a Friend” question.

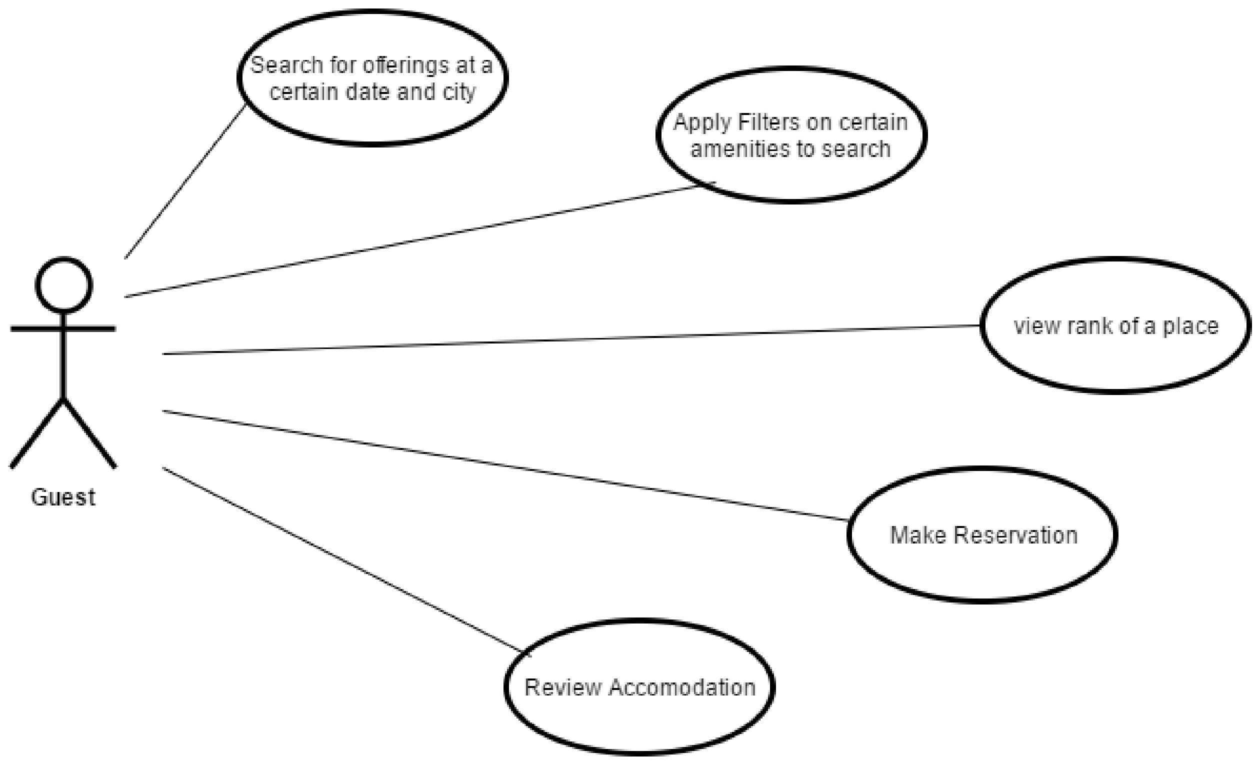

Search for offerings at a certain date and city
Apply Filters on certain amenities to search
view rank of a place
Make Reservation
Review Accomodation
Guest

Figure 03: Guest Use Case Diagram

## 4.2. ALGORITHMS

### Accommodation Ranking

Users review accommodation by providing a rating over 5 and answering a "would you recommend this place to a friend?" question with yes/no.

When looking for a new accommodation the users should be able to view the ranking of this accommodation consisting of the average of all ratings and the percentage of recommendations.

In order to compute the ranking of an accommodation, the Review table will be filtered to only include rankings related to the desired accommodation, and then the average of all rating values will be computed. Then the number of recommendations will be used along with the total number of reviews to calculate the percentage of recommendation of this place.

### 4.3. DATA STRUCTURES

## 5. USER INTERFACE DESIGN/SQL STATEMENTS

### 5.1. Sign up

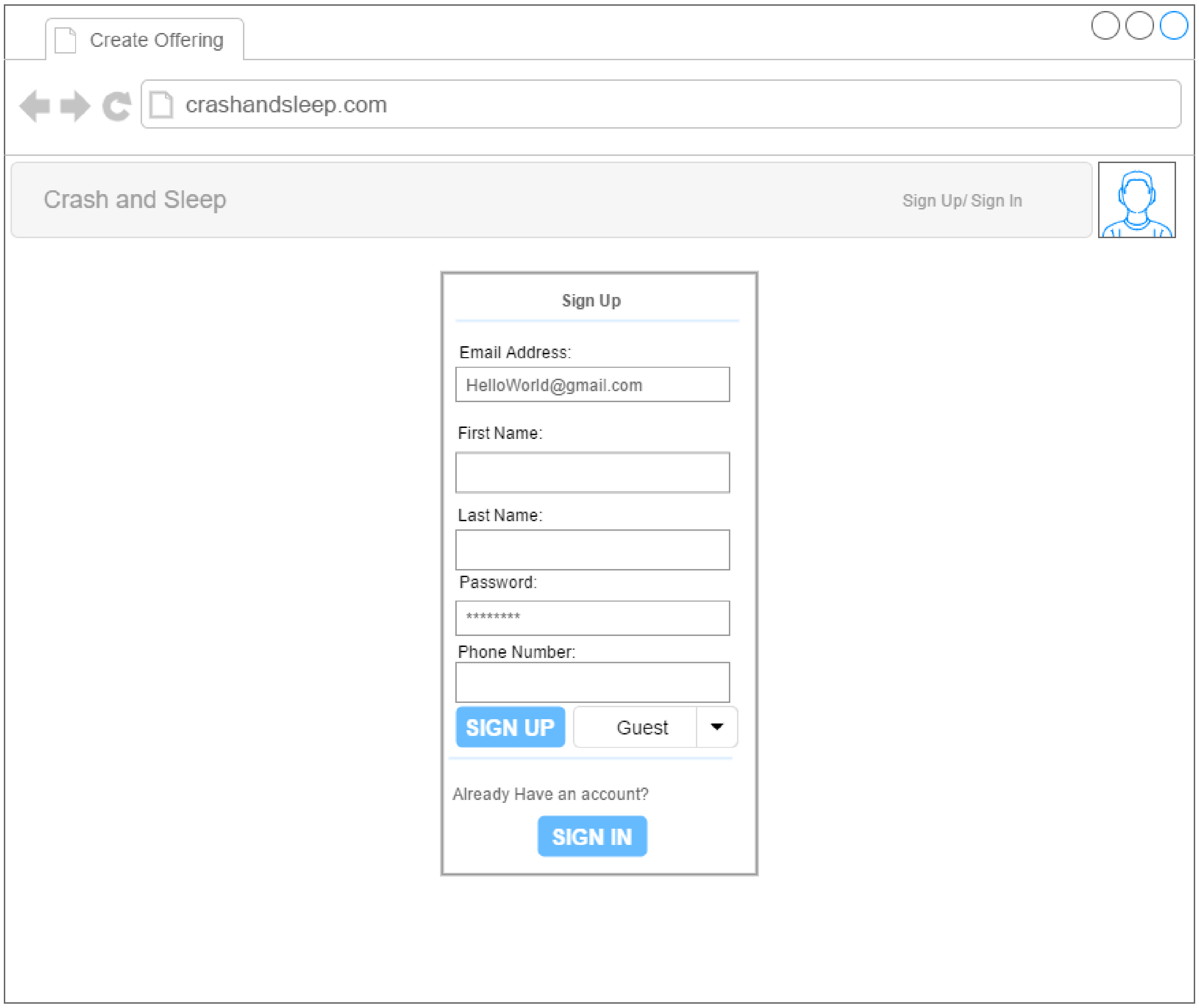


Figure 04: Sign up Page

**Inputs:** @email, @firstName, @lastName, @password, @number, @type

**Process:** User creates a guest or host account using email, first name, last name, and password. A new account is added to the database.

**SQL Statements:**

INSERT INTO Account VALUES ( 0, @email, @password, @firstName, @lastName, @number);

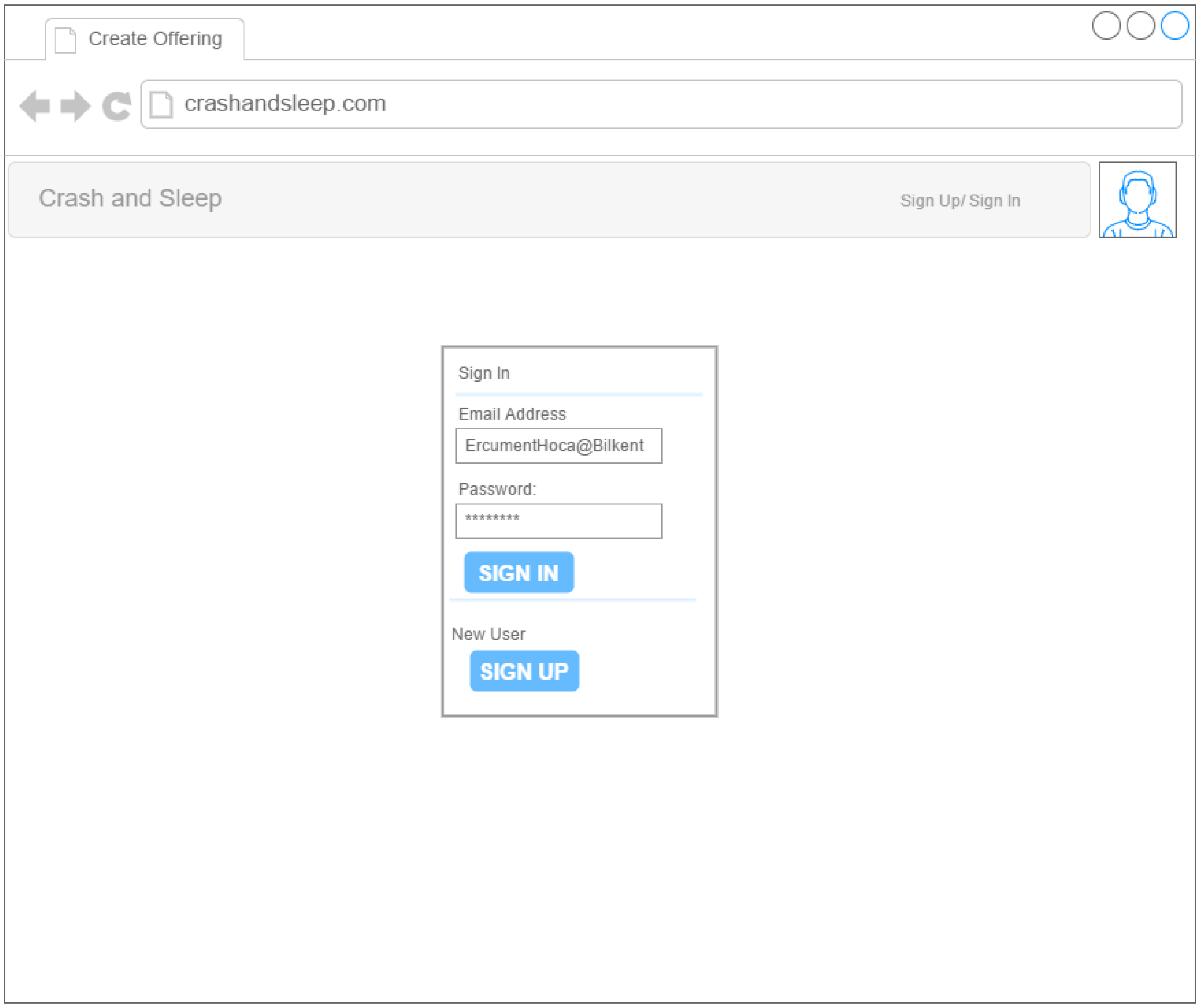
*If @type = Guest:*

INSERT INTO Guest VALUES ( account_ID, 0);

*If @type = Host:*

INSERT INTO Host VALUES (account_ID, 0);

### 5.2. Sign in



Email Address
ErcumentHoca@Bilkent
Password:
********

Figure 05: Sign in Page

**Inputs:** @email, @password

**Process:** User enters his email and password and logs in.

**SQL Statements:**

```
SELECT account_ID

FROM  Accounts

WHERE email = @email AND password = @password;
```

### 5.3. Accept/Refuse Hosts

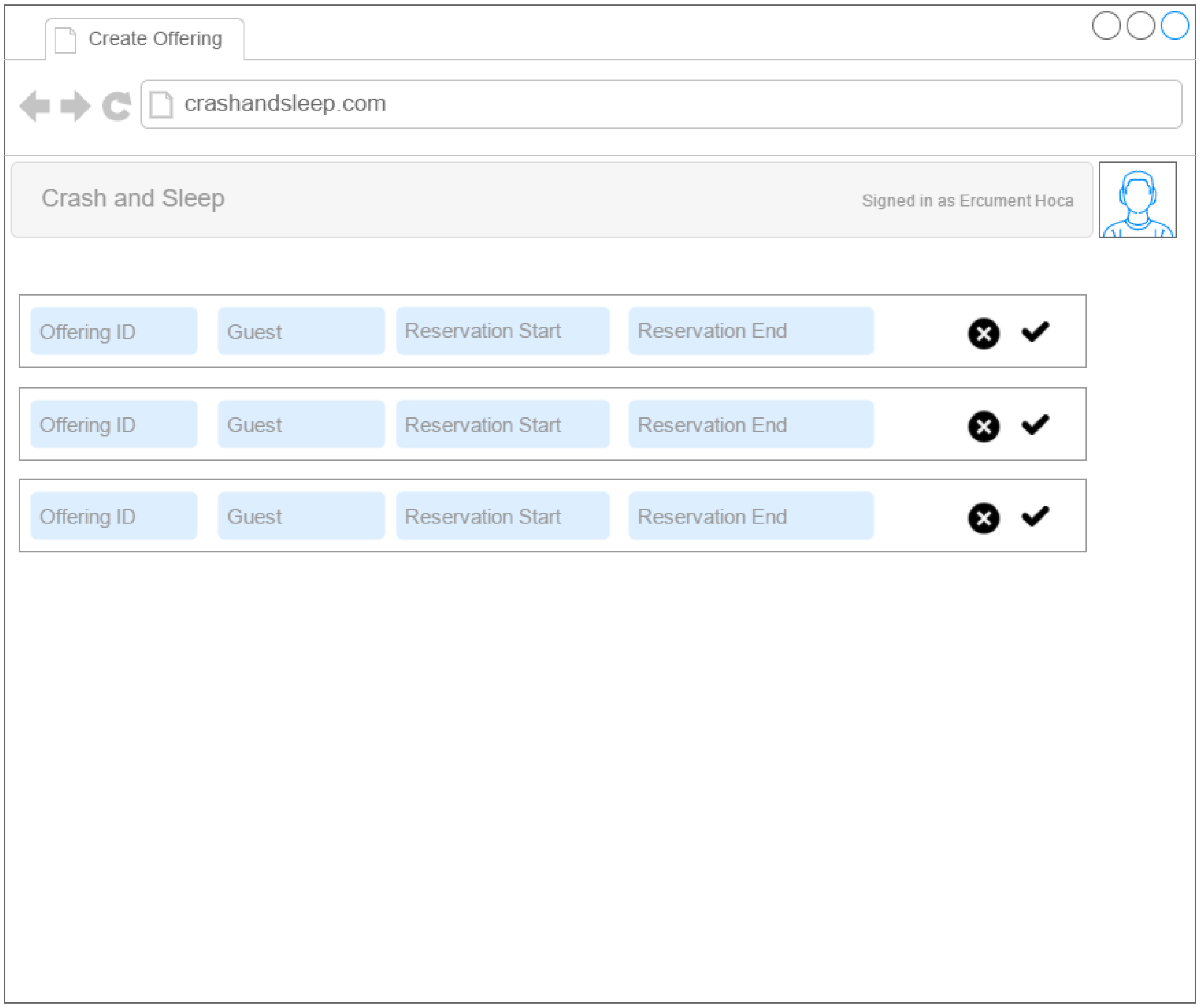


Figure 06: Accept/Refuse Hosts Page

**Inputs:** @accept (True or False according to what the user clicks)

**Process:** Host sees a list of reservation requests. He either accepts or accepts each request.

**SQL Statements:**

SELECT offeringID, firstName, lastName, reservation_start, reservation_end

FROM Offers join ( (makes natural join Account) natural join Reservation) on Offers.offeringID = Makes.offeringID

WHERE Offers,AccountID = @currentUserID;

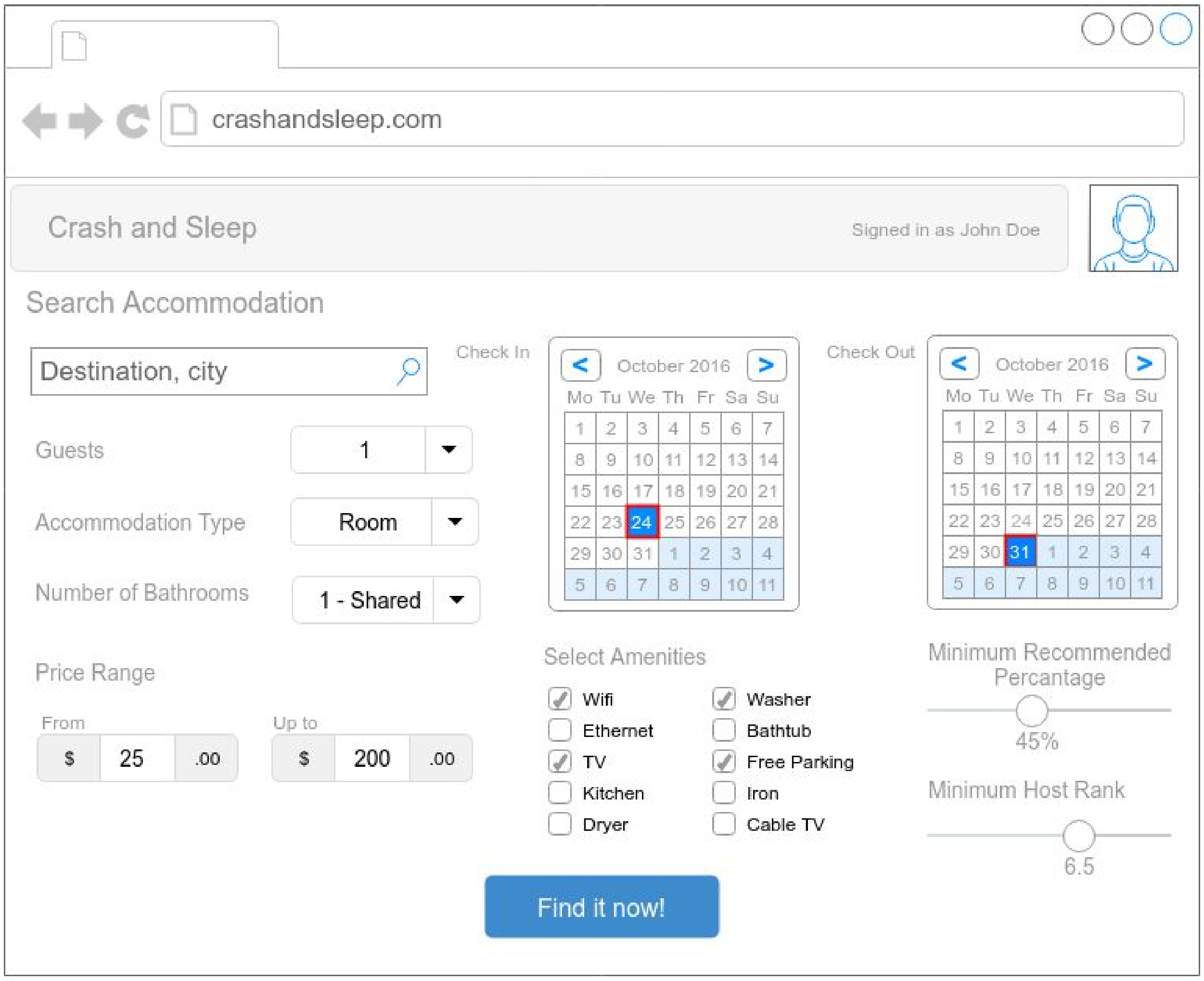
*If offer accepted:*

INSERT INTO Decides VALUES( @selectedReservationID, @currentUserID, True);

*If offer Declined:*

INSERT INTO Decides VALUES( @selectedReservationID, @currentUserID, False);

## 5.4. Search Accommodation


crashandsleep.com

Crash and Sleep — Signed in as John Doe

Search Accommodation

Destination, city

Guests: 1

Accommodation Type: Room

Number of Bathrooms: 1 - Shared

Price Range — From: $ 25 .00 — Up to: $ 200 .00

Check In: October 2016, 24

Check Out: October 2016, 31

Select Amenities:
- [✓] Wifi
- [ ] Ethernet
- [✓] TV
- [ ] Kitchen
- [ ] Dryer
- [✓] Washer
- [ ] Bathtub
- [✓] Free Parking
- [ ] Iron
- [ ] Cable TV

Minimum Recommended Percantage: 45%

Minimum Host Rank: 6.5

Figure 07: Search Accommodation Page

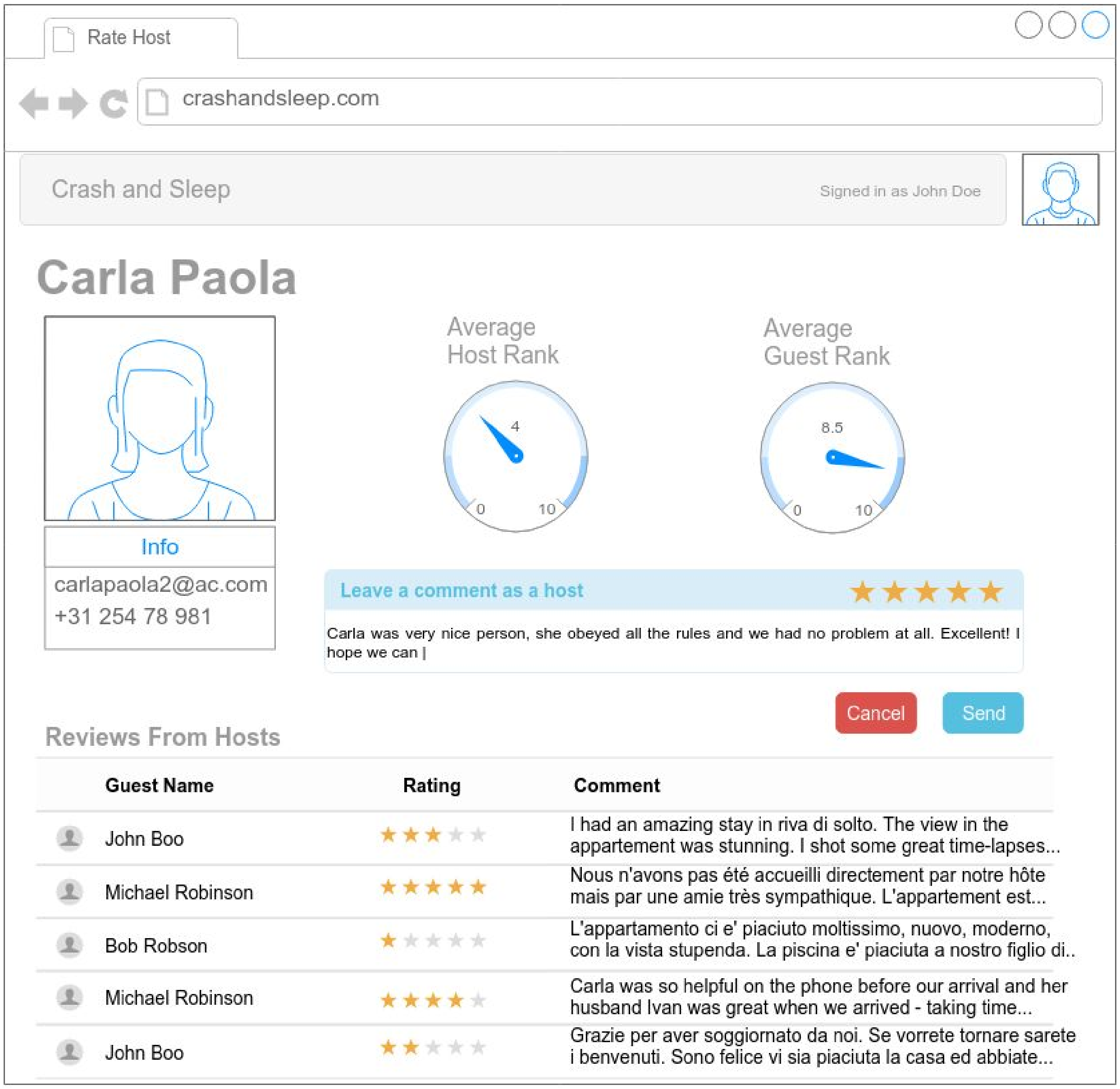
**Inputs:** @city, @district, @check_in, @check_out, @num_of_guest, @type, @num_of_baths, @price_from, @price_up, @wifi, @ethernet, @tv, @kitchen, @dryer, @washer, @bathtub, @free_parking, @iron, @cable_tv, @min_recomm_percentage, @min_host_rank

**Process:** User wants to find an accommodation according to chosen requirements.

**SQL Statements:**

```
SELECT H.name, H.surname, H.avg_host_rank, O.accommodates, O.price_per_night,
          Acc.city, Acc.district, Acc.street, Acc.type, Acc.percentageRecommend,
          Ame.number_of_bathrooms, Ame.wifi, Ame.ethernet, Ame.tv,
          Ame.kitchen, Ame.dryer, Ame.iron, Ame.cable_tv, Ame.bathtub,
          Ame.washer, Ame.free_parking
FROM Host H, Offers Of, Offering O, Requires R, Accommodation Acc, Amenities Ame
WHERE O.offering_ID = Of.offering_ID
          AND Of.account_ID = H.account_ID AND O.offering_ID = R.offering_ID
          AND R.accommodation_ID = Acc.accommodation_ID
          AND Acc.accommodation_ID = Ame.accommodation_ID
          AND Acc.city = @city AND Acc.district = @district
          AND O.start_date <= @check_in AND O.end_date >= @check_out
          AND O.accommodates >= @num_of_guest AND Acc.type = @type
          AND Ame.number_of_bathrooms = @num_of_baths
          AND O.price_per_night BETWEEN @price_from AND @price_up
          AND Ame.wifi = @wifi AND Ame.ethernet = @ethernet
          AND Ame.tv = @tv AND Ame.kitchen = @kitchen AND Ame.dryer = @dryer
          AND Ame.washer = @washer AND Ame.bathtub = @bathtub
          AND Ame.free_parking = @free_parking AND Ame.iron = @iron
          AND Ame.cable_tv = @cable_tv
          AND Acc.percentageRecommend >= @min_recomm_percentage
          AND H.avg_host_rank >= @min_host_rank
```

## 5.5. Host Ranks Guest



carlapaola2@ac.com
+31 254 78 981

Carla was very nice person, she obeyed all the rules and we had no problem at all. Excellent! I hope we can |

Reviews From Hosts

| Guest Name | Rating | Comment |
|---|---|---|
| John Boo | ★★★☆☆ | I had an amazing stay in riva di solto. The view in the appartement was stunning. I shot some great time-lapses... |
| Michael Robinson | ★★★★★ | Nous n'avons pas été accueilli directement par notre hôte mais par une amie très sympathique. L'appartement est... |
| Bob Robson | ★☆☆☆☆ | L'appartamento ci e' piaciuto moltissimo, nuovo, moderno, con la vista stupenda. La piscina e' piaciuta a nostro figlio di.. |
| Michael Robinson | ★★★★☆ | Carla was so helpful on the phone before our arrival and her husband Ivan was great when we arrived - taking time... |
| John Boo | ★★☆☆☆ | Grazie per aver soggiornato da noi. Se vorrete tornare sarete i benvenuti. Sono felice vi sia piaciuta la casa ed abbiate... |

Figure 08: Host ranks his/her guest

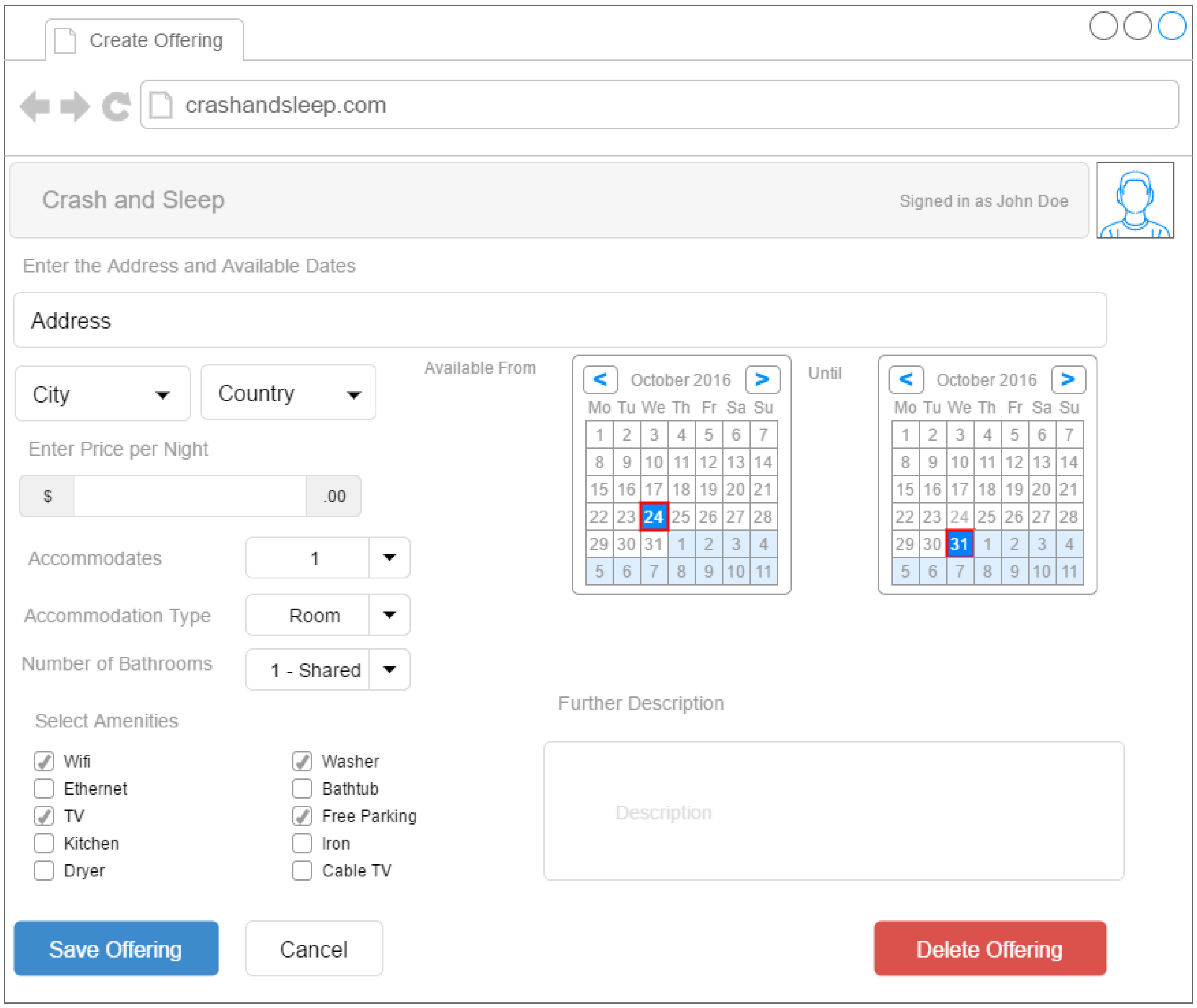
**Inputs:** @guest_rating, @comment, @date, @H.account_ID, @G.account_ID

**Process:** Host leaves a comment about his/her guest by clicking a “leave a comment” part on the guest’s page after guest departs from the house.

**SQL Statements:**

```
INSERT INTO hostRevs VALUES (  @H.account_ID, @G.account_ID, @guest_rating,
                                    @comment, @date);
DELIMITER $$
CREATE TRIGGER update_rank AFTER INSERT ON hostRevs
        FOR EACH ROW BEGIN
                UPDATE Guest
                        SET avg_guest_rank =
                                ((SELECT sum(rating) total
                                FROM hostRevs HR
                                GROUP BY G.account_ID
                                  HAVING HR.G.account_ID = @G.account_ID
                                  WHERE Guest.account_ID = @G.account_ID)
                                + @guest_rating) /
                                (SELECT count(1)
                                 FROM hostRevs HR
                                 GROUP BY G.account_ID
                                 HAVING G.account_ID = @G.account_ID)
                        WHERE account_ID = @G.account_ID;
        END $$
DELIMITER ;
```

## 5.6. Make Offering



Accommodates: 1

Accommodation Type: Room

Number of Bathrooms: 1 - Shared

Available From: October 2016, 24 selected

Until: October 2016, 31 selected

Select Amenities

- [✓] Wifi
- [ ] Ethernet
- [✓] TV
- [ ] Kitchen
- [ ] Dryer
- [✓] Washer
- [ ] Bathtub
- [✓] Free Parking
- [ ] Iron
- [ ] Cable TV

Figure 09: Creating an offering

**Inputs:** @city, @country, @address, @start_date, @end_date, @accommodates, @type, @price_per_night, @num_of_baths, @wifi, @ethernet, @tv, @kitchen, @dryer, @washer, @bathtub, @free_parking, @iron, @cable_tv

**Process:** User creates an offering and an accommodation with the respective qualities.

**SQL Statements:**

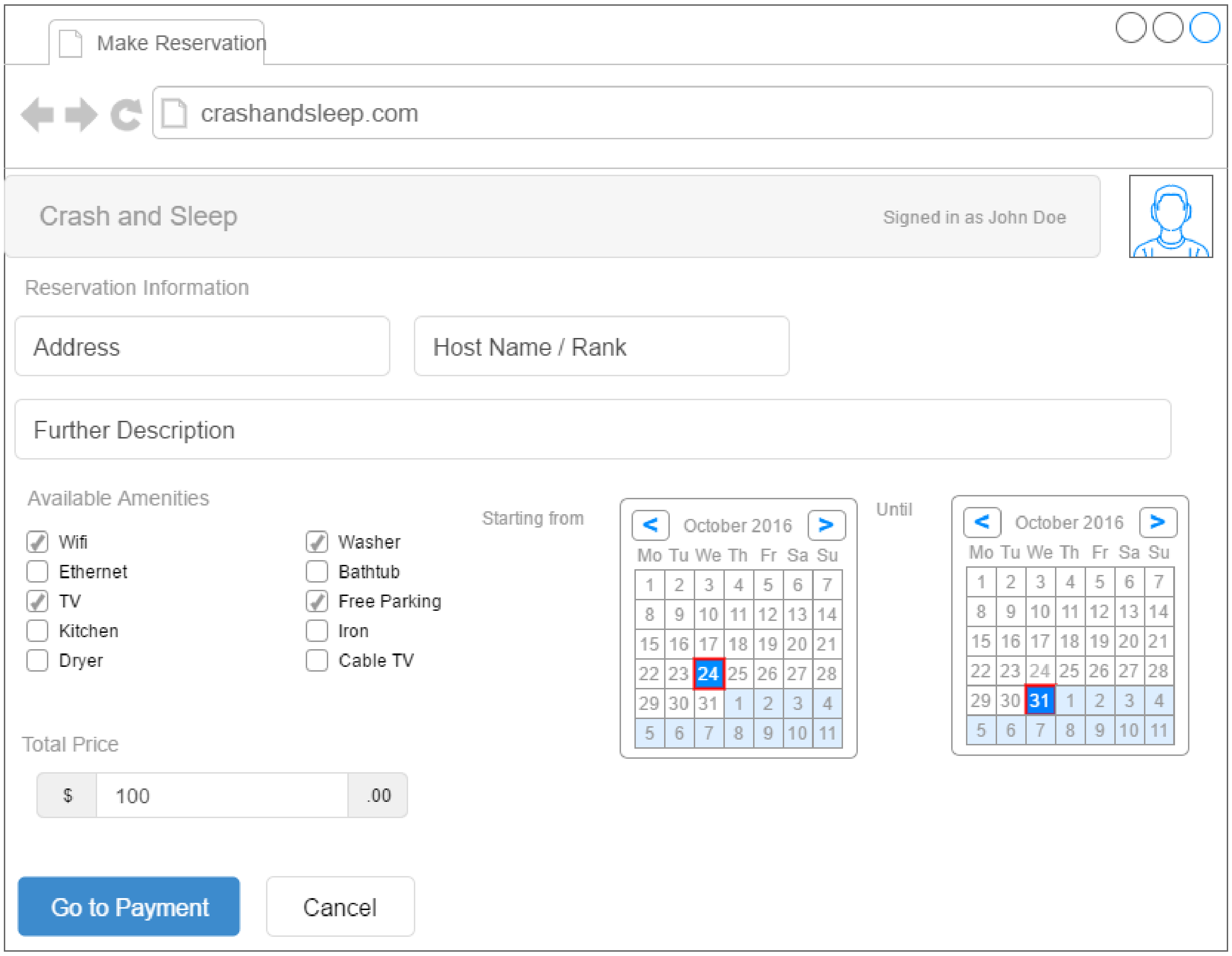
```sql
INSERT INTO Offering VALUES (offering_ID, @start_date, @end_date,
@accommodates, @price_per_night);

INSERT INTO Accommodation VALUES (accommodation_ID, @number_of_people,
@type, @address, @city, @country, 0);

INSERT INTO requires (offering_ID, accommodation_ID);

INSERT INTO amenities (accommodation_ID, @wifi, @ethernet, @tv, @kitchen,
@dryer, @washer, @bathtub, @free_parking, @iron, @cable_tv)
```

## 5.7. Make Reservation



- [✓] Wifi
- [ ] Ethernet
- [✓] TV
- [ ] Kitchen
- [ ] Dryer
- [✓] Washer
- [ ] Bathtub
- [✓] Free Parking
- [ ] Iron
- [ ] Cable TV

Starting from: October 2016, 24

Until: October 2016, 31

Total Price: $ 100 .00

Figure 10: Making a reservation

**Inputs:** @reserve_start, @reserve_end, @accommodation_ID, @account_ID

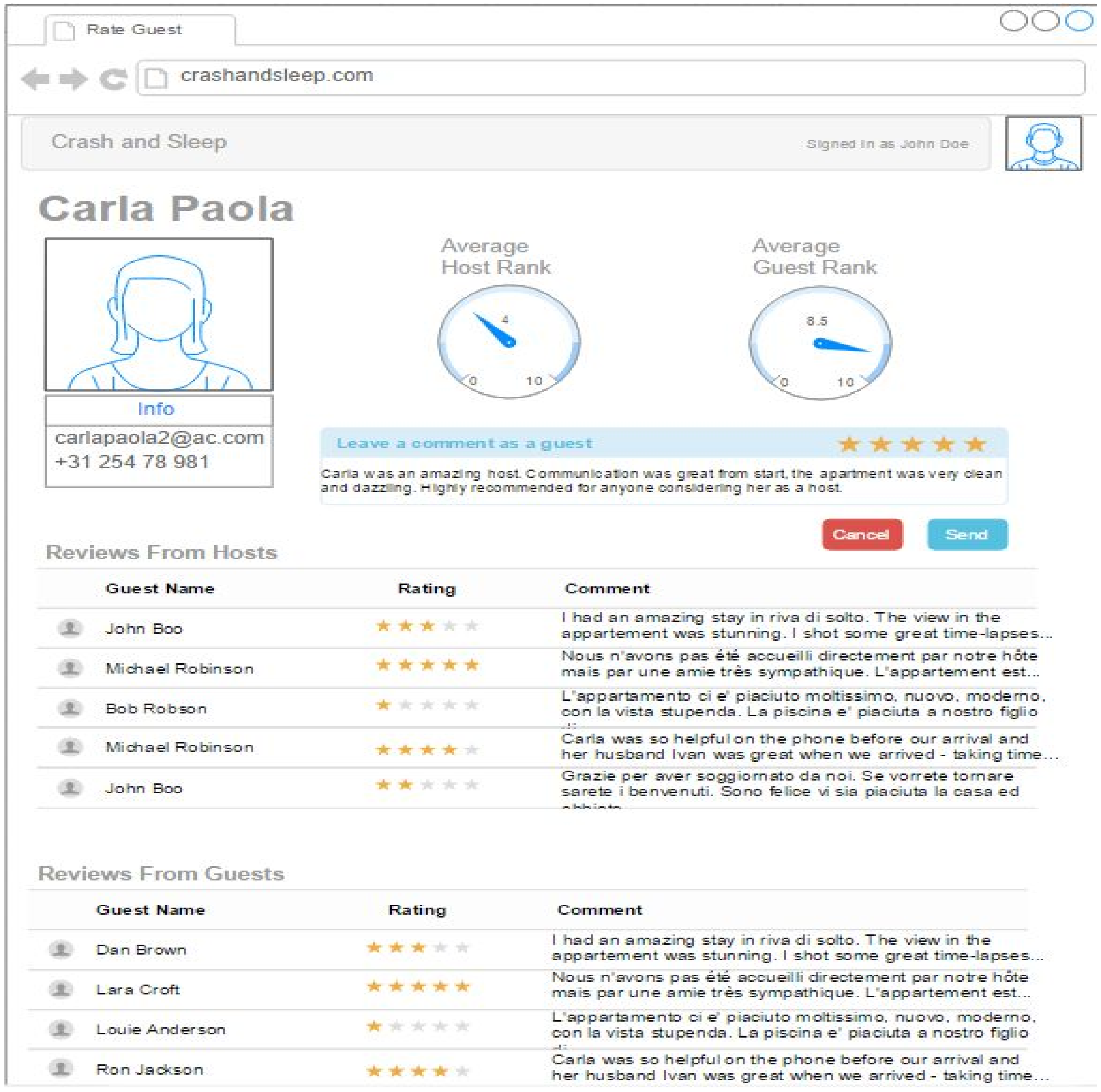
**Process:** The user views the final information about the place they want to stay in, and choose the dates between which they will stay.

**SQL Statements:**

INSERT INTO Reservation VALUES (reservation_ID, @reserve_start, @reserve_end);

INSERT INTO Makes VALUES (reservation_ID, @account_ID, @accommodation_ID)

## 5.8. Guest Ranks Host



Carla Paola

carlapaola2@ac.com
+31 254 78 981

Carla was an amazing host. Communication was great from start, the apartment was very clean and dazzling. Highly recommended for anyone considering her as a host.

Reviews From Hosts

| Guest Name | Rating | Comment |
|---|---|---|
| John Boo | ★★★☆☆ | I had an amazing stay in riva di solto. The view in the appartement was stunning. I shot some great time-lapses... |
| Michael Robinson | ★★★★★ | Nous n'avons pas été accueilli directement par notre hôte mais par une amie très sympathique. L'appartement est... |
| Bob Robson | ★☆☆☆☆ | L'appartamento ci e' piaciuto moltissimo, nuovo, moderno, con la vista stupenda. La piscina e' piaciuta a nostro figlio |
| Michael Robinson | ★★★★☆ | Carla was so helpful on the phone before our arrival and her husband Ivan was great when we arrived - taking time... |
| John Boo | ★★☆☆☆ | Grazie per aver soggiornato da noi. Se vorrete tornare sarete i benvenuti. Sono felice vi sia piaciuta la casa ed |

Reviews From Guests

| Guest Name | Rating | Comment |
|---|---|---|
| Dan Brown | ★★★☆☆ | I had an amazing stay in riva di solto. The view in the appartement was stunning. I shot some great time-lapses... |
| Lara Croft | ★★★★★ | Nous n'avons pas été accueilli directement par notre hôte mais par une amie très sympathique. L'appartement est... |
| Louie Anderson | ★☆☆☆☆ | L'appartamento ci e' piaciuto moltissimo, nuovo, moderno, con la vista stupenda. La piscina e' piaciuta a nostro figlio |
| Ron Jackson | ★★★★☆ | Carla was so helpful on the phone before our arrival and her husband Ivan was great when we arrived - taking time... |

Figure 11: Guest ranks his/her host

**Inputs:** @host_rating, @comment, @date, @host_id, @guest_id, @recommended, @review_id

**Process:** Guest leaves a comment about his/her host by clicking a “leave a comment” part on the host’s page after guest departs from the house.

**SQL Statements:**

```
SELECT LAST A.accommodation_ID as accom
FROM Host H, Offering O, Accommodation A, offers T, requires R, Guest G, makes M
WHERE H.account_ID = T.account_ID
        AND T.offering_ID = O.offering_ID
        AND O.offering_ID = R.offering_ID
        AND R.accomodation_ID = A.accommodation_ID
        AND H.account_ID = @host_id
        AND G.account_ID = @guest_id
        AND G.account_ID = M.account_ID
        AND M.offering_ID = O.offering_ID
INSERT INTO review VALUES (  @review_id, @guest_rating,
                                        @comment, @recommended, @date);
INSERT INTO ranks VALUES (  @review_id, @guest_ID);
INSERT INTO accomRevs VALUES (  @review_id, accom);


DELIMITER $$
CREATE TRIGGER update_review AFTER INSERT ON review
        FOR EACH ROW BEGIN
                UPDATE Host
                        SET avg_host_rank =
                    ((SELECT sum(rating) total

                    FROM Review R, Accommodation A, Offering O, Host H,
              Guest G, offers T, requires RQ, accomRevs AR, ranks GR

                    WHERE H.account_ID = T.account_ID

                          AND T.offering_ID = O.offering_ID

                          AND O.offering_ID = R.offering_ID

                          AND R.accomodation_ID = A.accommodation_ID

                          AND A.accommodation_ID =
                 AR.accommodation_ID

                          AND AR.review_ID = R.review_ID

                          AND G.account_ID = @guest_ID

                          AND H.account_ID = @host_id

                          AND G.account_ID = GR.account_ID

                          AND GR.review_ID = R.review_ID

                    GROUP BY H.account_ID

                     HAVING H.account_ID = @host_ID

                    + @host_rating) /

                    (SELECT count(1)

                     FROM Review R, Accommodation A, Offering O, Host H,
      Guest G, offers T, requires RQ, accomRevs AR, ranks GR

                    WHERE H.account_ID = T.account_ID

                          AND T.offering_ID = O.offering_ID
                    AND O.offering_ID = R.offering_ID

                    AND R.accomodation_ID = A.accommodation_ID

                    AND A.accommodation_ID =
            AR.accommodation_ID

                    AND AR.review_ID = R.review_ID

                    AND G.account_ID = @guest_ID

                    AND H.account_ID = @host_id

                    AND G.account_ID = GR.account_ID

                    AND GR.review_ID = R.review_ID

                GROUP BY H.account_ID

                 HAVING H.account_ID = @host_ID

    END $$

DELIMITER ;
```

# 6. ADVANCED DATABASE COMPONENTS

## 6.1. VIEWS

### 6.1.1. Offerings View For All Users

```
CREATE VIEW Offering_Info AS

        SELECT start_date, end_date, number_of_people, type, address

        FROM Offering O, Requires R, Accommodation A

        WHERE O.offering_ID = R.offering_ID AND
A.Accommodation_ID = R.Accommodation_ID
```

## 6.2. STORED PROCEDURES

- A procedure will be used to notify users when they have been reviewed.
- A procedure will be used to notify hosts when one of their offerings is requested by a guest.
- A procedure will be used to notify guests when one of their reservations is accepted.

## 6.3. TRIGGERS

- When a review is added to Reviews table, the corresponding host's rank is updated.
- When a review is added to hostRev table, the corresponding guest's rank is updated.

## 6.4. CONSTRAINTS

- The system can not be used without log-in.
- A guest can not reserve an offering with a past end-date.
- A guest can not review their host until the end-date of their stay.
- A host can not review their guest until the end-date of guest's stay.
- A user can not see other users' passwords.
- End-date of an offering can not be before start-date.