İhsan Doğramacı Bilkent University

Fall 2016

CS353 - Database Systems

Term Project Design Report

Online Accommodation System

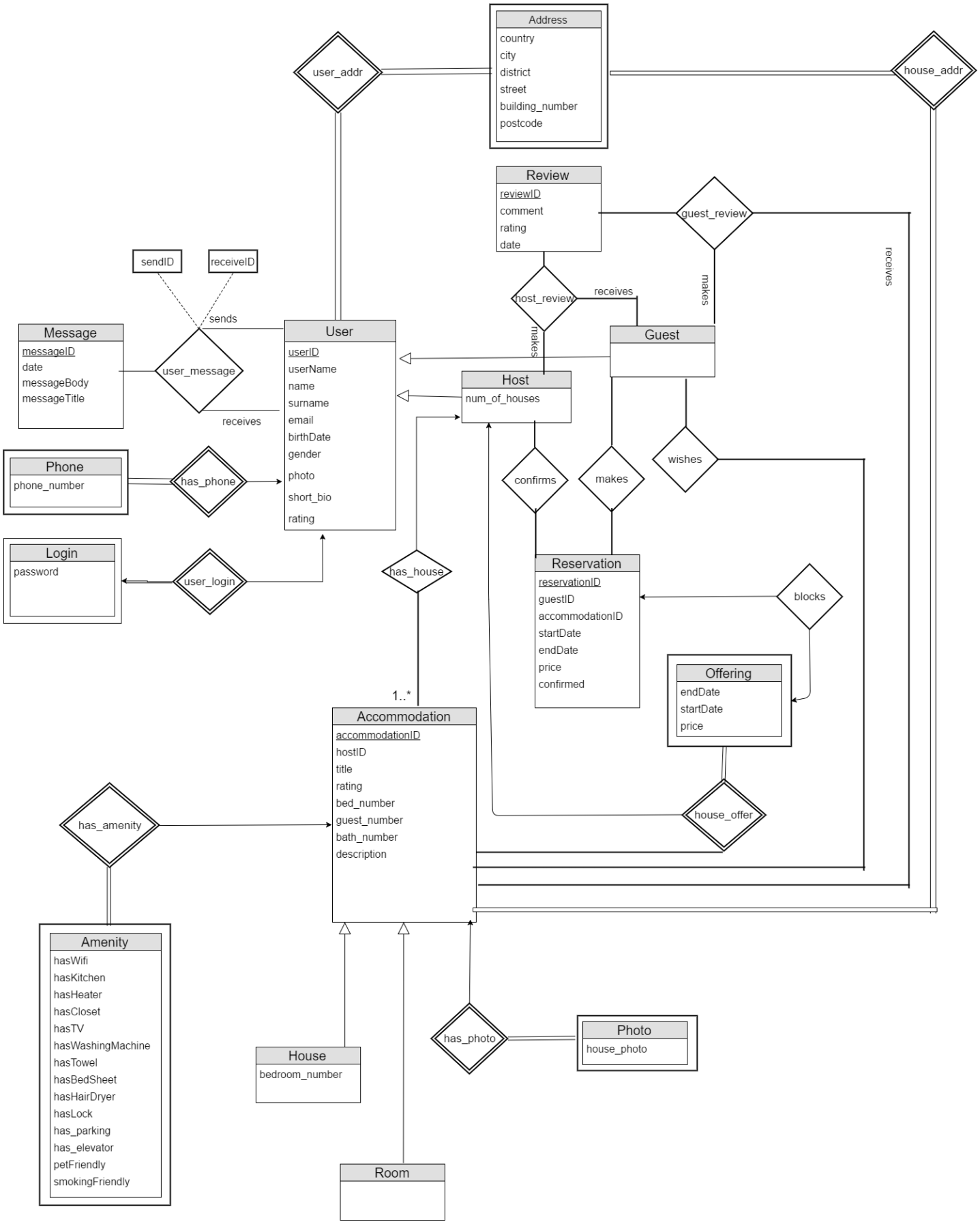
Group 1

Pınar Ezgi Çöl - 21301301
Yonca Yunatcı - 21301970
Selin Fildiş - 21402228
Berk Türk - 21302570

30.11.2016

Contents

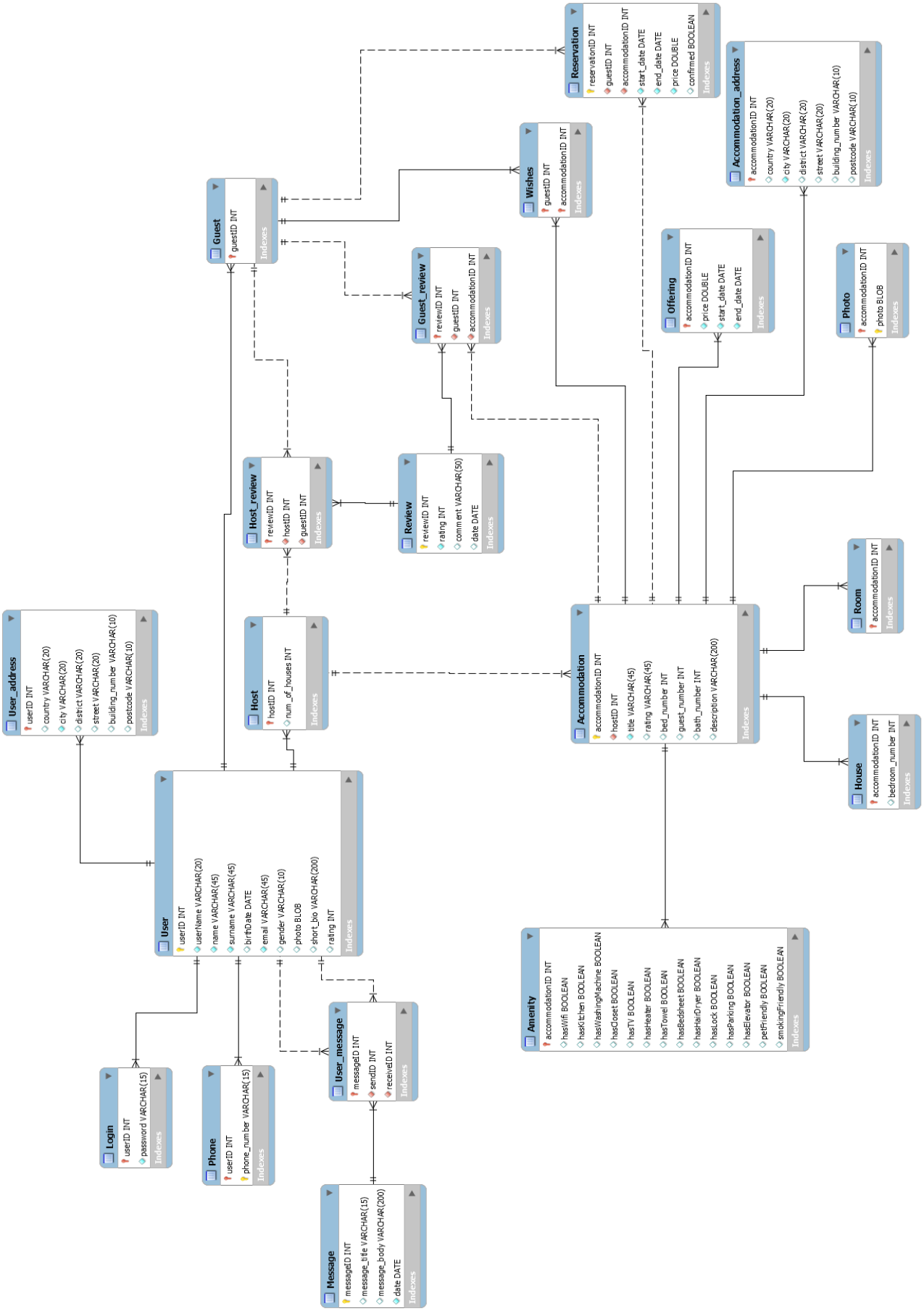
1. ER Diagram


Address
country
city
district
street
building_number
postcode
user_addr
house_addr
Review
reviewID
comment
rating
date
guest_review
receives
makes
host_review
receives
makes
sendID
receiveID
sends
Message
messageID
date
messageBody
messageTitle
user_message
receives
User
userID
userName
name
surname
email
birthDate
gender
photo
short_bio
rating
Guest
Host
num_of_houses
Phone
phone_number
has_phone
confirms
makes
wishes
Login
password
user_login
has_house
Reservation
reservationID
guestID
accommodationID
startDate
endDate
price
confirmed
blocks
Offering
endDate
startDate
price
1..*
Accommodation
accommodationID
hostID
title
rating
bed_number
guest_number
bath_number
description
house_offer
has_amenity
Amenity
hasWifi
hasKitchen
hasHeater
hasCloset
hasTV
hasWashingMachine
hasTowel
hasBedSheet
hasHairDryer
hasLock
has_parking
has_elevator
petFriendly
smokingFriendly
House
bedroom_number
has_photo
Photo
house_photo
Room

1.1 The changes in the ER diagram

- We separate the host and the guest roles. "is a" relationship is used to connect these roles to the user.
- Amenities are separated from Accommodation table.
- House table is updated as Accommodation table. An accommodation could be a house or a room. This distinction is also represented in House and Room tables with "is a" relationship.
- The relation between User and Message is updated. Ternary relationship is used to keep the sender and receiver IDs.
- Login table is created with userID and password.
- Review table is created with reviewID, comment and rating. A review from a host to guest and a review from guest to house are also differentiated and kept in separate tables named Host_review and Guest_review.
- Hosts are given the opportunity to offer their houses for a certain time period. This is kept in Offering table.
- The all reservations made are in the Reservation table with a unique id. If the host of the house confirms the reservation, then the offering for the house for that date interval will be blocked.

2. Relational Schema



2.1 User

Relational Model

User(<u>userID</u>,userName, name, surname, birthDate, email, gender, photo, short_bio, rating)

Functional Dependencies

userID -> userName, name, surname, birthDate, email, gender, photo, short_bio, rating

userName -> userID

email -> userID

Keys

Candidate Keys: {(userID),(userName),(email)}
Primary Key: userID
Foreign Key: none

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `userID` INT NOT NULL AUTO_INCREMENT,
  `userName` VARCHAR(20) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `birthDate` DATE NULL,
  `email` VARCHAR(45) NOT NULL,
  `gender` VARCHAR(10) NULL,
  `photo` BLOB NULL,
  `short_bio` VARCHAR(200) NULL,
  `rating` INT NULL,
  PRIMARY KEY (`userID`))
ENGINE = InnoDB;
```

2.2 Host

Relational Model

Host(<u>hostID</u> (FK by User), num_of_houses)

Functional Dependencies

none

Keys

Candidate Keys: {hostID}
Primary Key: hostID
Foreign Key: hostID (by User (userID))

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Host` (
  `hostID` INT NOT NULL,
 `num_of_houses` INT NULL,
  PRIMARY KEY (`hostID`),
  CONSTRAINT `hostID`
    FOREIGN KEY (`hostID`)
    REFERENCES `mydb`.`User` (`userID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.3 Guest

Relational Model

Guest(guestID (FK by User))

Functional Dependencies

none

Keys

Candidate Keys: {guestID}
Primary Key: guestID
Foreign Key: guestID (by User (userID))

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Guest` (
  `guestID` INT NOT NULL,
  PRIMARY KEY (`guestID`),
  CONSTRAINT `guestID`
    FOREIGN KEY (`guestID`)
    REFERENCES `mydb`.`User` (`userID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.4 Accommodation

Relational Model

Accommodation(accommodationID,hostID (FK by Host), title, rating, bed_number, guest_number, bath_number, description)

Functional Dependencies

accommodationID -> hostID, title, rating, bed_number, guest_number, bath_number, description

Keys

Candidate Keys: { accommodationID}
Primary Key: accommodationID
Foreign Key: hostID (by Host)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Accommodation` (
  `accommodationID` INT NOT NULL AUTO_INCREMENT,
  `hostID` INT NOT NULL,
  `title` VARCHAR(45) NOT NULL,
  `rating` VARCHAR(45) NULL,
  `bed_number` INT NULL,
  `guest_number` INT NULL,
  `bath_number` INT NULL,
  `description` VARCHAR(200) NULL,
  PRIMARY KEY (`accommodationID`),
  INDEX `hostID_idx` (`hostID` ASC),
  CONSTRAINT `hostID`
    FOREIGN KEY (`hostID`)
    REFERENCES `mydb`.`Host` (`hostID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.5 House

Relational Model

House(accommodationID (FK by Accommodation), bedroom_number)

Functional Dependencies

accommodationID -> bedroom_number

Keys

Candidate Keys: { accommodationID }
Primary Key: accommodationID
Foreign Key: hostID (by Host)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`House` (
  `accommodationID` INT NOT NULL,
  `bedroom_number` INT NULL,
  PRIMARY KEY (`accommodationID`),
  CONSTRAINT `accommodationID`
   FOREIGN KEY (`accommodationID`)
   REFERENCES `mydb`.`Accommodation` (`accommodationID`)
    ON DELETE CASCADE
   ON UPDATE CASCADE)
 ENGINE = InnoDB;
```

2.6 Room

Relational Model

Room(accommodationID (FK by Accommodation))

Functional Dependencies

none

Keys

Candidate Keys: { accommodationID }
Primary Key: accommodationID
Foreign Key: accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Room` (
  `accommodationID` INT NOT NULL,
  PRIMARY KEY (`accommodationID`),
  CONSTRAINT `accommodationID`
   FOREIGN KEY (`accommodationID`)
    REFERENCES `mydb`.`Accommodation` (`accommodationID`)
    ON DELETE CASCADE
   ON UPDATE CASCADE)
 ENGINE = InnoDB;
```

2.7 Amenity

Relational Model

Amenity(AccommodationID (FK by Accommodation), hasWifi, hasKitchen,
hasWashingMachine, hasCloset, hasTV, hasHeater, hasTowel, hasBedSheet,
hasHairDryer, hasLock, hasParking, hasElevator, petFriendly, smokingFiendly)

Functional Dependencies

AccommodationID -> hasWifi, hasKitchen, hasWashingMachine, hasCloset, hasTV,
hasHeater, hasTowel, hasBedSheet, hasHairDryer, hasLock, hasParking,
hasElevator, petFriendly, smokingFiendly

Keys

Candidate Keys: { accommodationID }
Primary Key: accommodationID
Foreign Key: accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Amenity` (
  `accommodationID` INT NOT NULL,
  `hasWifi` VARCHAR(5) NULL,
  `hasKitchen` VARCHAR(5) NULL,
  `hasWashingMachine` VARCHAR(5) NULL,
  `hasCloset` VARCHAR(5) NULL,
  `hasTV` VARCHAR(5) NULL,
  `hasHeater` VARCHAR(5) NULL,
  `hasTowel` VARCHAR(5) NULL,
  `hasBedsheet` VARCHAR(5) NULL,
  `hasHairDryer` VARCHAR(5) NULL,
  `hasLock` VARCHAR(5) NULL,
  `hasParking` VARCHAR(5) NULL,
  `hasElevator` VARCHAR(5) NULL,
  `petFriendly` VARCHAR(5) NULL,
  `smokingFriendly` VARCHAR(5) NULL,
  PRIMARY KEY (`accommodationID`),
 CONSTRAINT `accommodationID`
  FOREIGN KEY (`accommodationID`)
  REFERENCES `mydb`.`Accommodation` (`accommodationID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.8 Login

Relational Model

Login(userID (FK by User), password)

Functional Dependencies

userID -> password

Keys

Candidate Keys: { userID }
Primary Key: userID
Foreign Key: userID (by User)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Login` (
  `userID` INT NOT NULL,
 `password` VARCHAR(15) NOT NULL,
 PRIMARY KEY (`userID`),
  CONSTRAINT `userID`
  FOREIGN KEY (`userID`)
  REFERENCES `mydb`.`User` (`userID`)
   ON DELETE CASCADE
  ON UPDATE CASCADE)
 ENGINE = InnoDB;
```

2.9 Phone

Relational Model

Phone(userID (FK by User), phone_number)

Functional Dependencies

none

Keys

Candidate Keys: { (userID,phone_number) }
Primary Keys: (userID, phone_number)
Foreign Key: userID (by User)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Phone` (
  `userID` INT NOT NULL,
  `phone_number` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`userID`, `phone_number`),
  CONSTRAINT `userID`
```

```
  FOREIGN KEY (`userID`)
 REFERENCES `mydb`.`User` (`userID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)

ENGINE = InnoDB;
```

2.10 Message

Relational Model

Message(<u>messageID</u>, message_title, message_body, date)

Functional Dependencies

messageID -> message_title, message_body, date

Keys

Candidate Keys: { messageID }
Primary Key: messageID
Foreign Key: none

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Message` (
  `messageID` INT NOT NULL AUTO_INCREMENT,
  `message_title` VARCHAR(15) NULL,
  `message_body` VARCHAR(200) NULL,
  `date` DATE NOT NULL,
 PRIMARY KEY (`messageID`))
ENGINE = InnoDB;
```

2.11 User_message

Relational Model

Message(<u>messageID</u> (FK by Message), sendID (FK by User),receiveID (FK by User))

Functional Dependencies

message -> sendID, receiveID

Keys

Candidate Keys: { messageID }
Primary Key: messageID

Foreign Key: messageID (by Message), sendID (by User (userID)),receiveID (by User (userID))

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`User_message` (
  `messageID` INT NOT NULL,
  `sendID` INT NOT NULL,
  `receiveID` INT NOT NULL,
  PRIMARY KEY (`messageID`),
  INDEX `sendID_idx` (`sendID` ASC),
 INDEX `receiveID_idx` (`receiveID` ASC),
 CONSTRAINT `messageID`
  FOREIGN KEY (`messageID`)
    REFERENCES `mydb`.`Message` (`messageID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `sendID`
    FOREIGN KEY (`sendID`)
   REFERENCES `mydb`.`User` (`userID`)
    ON DELETE CASCADE
   ON UPDATE CASCADE,
  CONSTRAINT `receiveID`
    FOREIGN KEY (`receiveID`)
    REFERENCES `mydb`.`User` (`userID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
 ENGINE = InnoDB;
```

2.12 User_address

Relational Model

user_address(<u>userID</u> (FK by User), country, city, district, street, building_number, postcode)

Functional Dependencies

userID -> country, city, district, street, building_number, postcode

Keys

Candidate Keys: { <u>userID</u> }
Primary Key: <u>userID</u>
Foreign Key: <u>userID</u> (by User)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`User_address` (
        `userID` INT NOT NULL,
        `country` VARCHAR(20) NULL,
        `city` VARCHAR(20) NOT NULL,
        `district` VARCHAR(20) NULL,
        `street` VARCHAR(20) NULL,
        `building_number` VARCHAR(10) NULL,
        `postcode` VARCHAR(10) NULL,
        PRIMARY KEY (`userID`),
        CONSTRAINT `userID`
          FOREIGN KEY (`userID`)
         REFERENCES `mydb`.`User` (`userID`)
          ON DELETE CASCADE
          ON UPDATE CASCADE)
       ENGINE = InnoDB;
```

2.13 Accommodation_address

Relational Model

accommodation_address(accommodationID (FK by Accommodation), country, city, district, street, building_number, postcode)

Functional Dependencies

accommodationID -> country, city, district, street, building_number, postcode

Keys

Candidate Keys: { accommodationID }
Primary Key: accommodationID
Foreign Key: accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Accommodation_address` (
        `accommodationID` INT NOT NULL,
       `country` VARCHAR(20) NULL,
        `city` VARCHAR(20) NOT NULL,
       `district` VARCHAR(20) NULL,
        `street` VARCHAR(20) NULL,
       `building_number` VARCHAR(10) NULL,
        `postcode` VARCHAR(10) NULL,
        PRIMARY KEY (`accommodationID`),
```

```
 CONSTRAINT `accommodationID`
  FOREIGN KEY (`accommodationID`)
  REFERENCES `mydb`.`Accommodation` (`accommodationID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.14 Photo

Relational Model

Photo(accommodationID (FK by Accommodation), photo)

Functional Dependencies

none

Keys

Candidate Keys: { (accommodationID, photo) }
Primary Keys: accommodationID, photo
Foreign Key: accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Photo` (
 `accommodationID` INT NOT NULL,
 `photo` BLOB NOT NULL,
 PRIMARY KEY (`accommodationID`, `photo`),
 CONSTRAINT `accommodationID`
 FOREIGN KEY (`accommodationID`)
  REFERENCES `mydb`.`Accommodation` (`accommodationID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.15 Offering

Relational Model

Offering(accommodationID (FK by Accommodation), start_date, end_date)

Functional Dependencies

accommodationID -> start_date, end_date

Keys

Candidate Keys: { accommodationID }

Primary Key: accommodationID
Foreign Key: accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Offering` (
        `accommodationID` INT NOT NULL,
        `price` DOUBLE NOT NULL,
        `start_date` DATE NOT NULL,
        `end_date` DATE NOT NULL,
        PRIMARY KEY (`accommodationID`),
        CONSTRAINT `accommodationID`
          FOREIGN KEY (`accommodationID`)
          REFERENCES `mydb`.`Accommodation` (`accommodationID`)
          ON DELETE CASCADE
          ON UPDATE CASCADE)
      ENGINE = InnoDB;
```

2.16 Reservation

Relational Model

Reservation(<u>reservationID</u>, guestID(FK by Guest), accommodationID (FK by Accommodation), start_date, end_date, price, confirmed)

Functional Dependencies

reservationID -> guestID, accommodationID, start_date, end_date, price, confirmed

Keys

Candidate Keys: { reservationID, (guestID, accommodationID, start_date, end_date) }
Primary Key: reservationID
Foreign Key: accommodationID (by Accommodation), guestID(by Guest)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Reservation` (
        `reservationID` INT NOT NULL AUTO_INCREMENT,
        `guestID` INT NOT NULL,
        `accommodationID` INT NOT NULL,
        `start_date` DATE NOT NULL,
        `end_date` DATE NOT NULL,
        `price` DOUBLE NOT NULL,
```

```
  `confirmed` TINYINT(1) NULL,
  PRIMARY KEY (`reservationID`),
  INDEX `guestID_idx` (`guestID` ASC),
  INDEX `accommodationID_idx` (`accommodationID` ASC),
  CONSTRAINT `guestID`
    FOREIGN KEY (`guestID`)
    REFERENCES `mydb`.`Guest` (`guestID`)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
 CONSTRAINT `accommodationID`
  FOREIGN KEY (`accommodationID`)
  REFERENCES `mydb`.`Accommodation` (`accommodationID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
 ENGINE = InnoDB;
```

2.17 Wishes

Relational Model

wishes(accommodationID (FK by Accommodation), guestID (FK by Guest))

Functional Dependencies

none

Keys

Candidate Keys: { (accommodationID, guestID) }
Primary Keys: accommodationID, guestID
Foreign Key: accommodationID (by Accommodation), guestID (by Guest)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Wishes` (
  `guestID` INT NOT NULL,
  `accommodationID` INT NOT NULL,
  PRIMARY KEY (`guestID`, `accommodationID`),
  INDEX `accommodationID_idx` (`accommodationID` ASC),
 CONSTRAINT `guestID`
  FOREIGN KEY (`guestID`)
   REFERENCES `mydb`.`Guest` (`guestID`)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
  CONSTRAINT `accommodationID`
 FOREIGN KEY (`accommodationID`)
```

```
REFERENCES `mydb`.`Accommodation` (`accommodationID`)
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB;
```

2.18 Review

Relational Model

Review(reviewID, rating, comment,date)

Functional Dependencies

reviewID -> rating, comment, date

Keys

Candidate Keys: { reviewID }
Primary Key: reviewID
Foreign Key: none

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Review` (
  `reviewID` INT NOT NULL,
  `rating` INT NOT NULL,
  `comment` VARCHAR(50) NULL,
  `date` DATE NULL,
  PRIMARY KEY (`reviewID`))
ENGINE = InnoDB;
```

2.19 Host_review

Relational Model

host_review(reviewID (FK by Review), hostID (FK by Host), guestID(FK by Guest))

Functional Dependencies

reviewID -> hostID, guestID

Keys

Candidate Keys: { reviewID }
Primary Key: reviewID
Foreign Key: reviewID (by Review), hostID (by Host), guestID(by Guest)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Host_review` (
        `reviewID` INT NOT NULL,
         `hostID` INT NOT NULL,
         `guestID` INT NOT NULL,
        PRIMARY KEY (`reviewID`),
        INDEX `hostID_idx` (`hostID` ASC),
        INDEX `guestID_idx` (`guestID` ASC),
        CONSTRAINT `reviewID`
         FOREIGN KEY (`reviewID`)
        REFERENCES `mydb`.`Review` (`reviewID`)
          ON DELETE CASCADE
          ON UPDATE CASCADE,
         CONSTRAINT `hostID`
           FOREIGN KEY (`hostID`)
           REFERENCES `mydb`.`Host` (`hostID`)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
        CONSTRAINT `guestID`
         FOREIGN KEY (`guestID`)
         REFERENCES `mydb`.`Guest` (`guestID`)
         ON DELETE CASCADE
        ON UPDATE CASCADE)
       ENGINE = InnoDB;
```

2.20 Guest_review

Relational Model

guest_review(<u>reviewID</u> (FK by Review), guestID(FK by Guest),
accommodationID (FK by Accommodation))

Functional Dependencies

reviewID -> guestID, accommodationID

Keys

Candidate Keys: { reviewID }
Primary Key: reviewID
Foreign Key: reviewID (by Review), guestID(by Guest), accommodationID (by Accommodation)

Table Definition

```
CREATE TABLE IF NOT EXISTS `mydb`.`Guest_review` (
        `reviewID` INT NOT NULL,
```

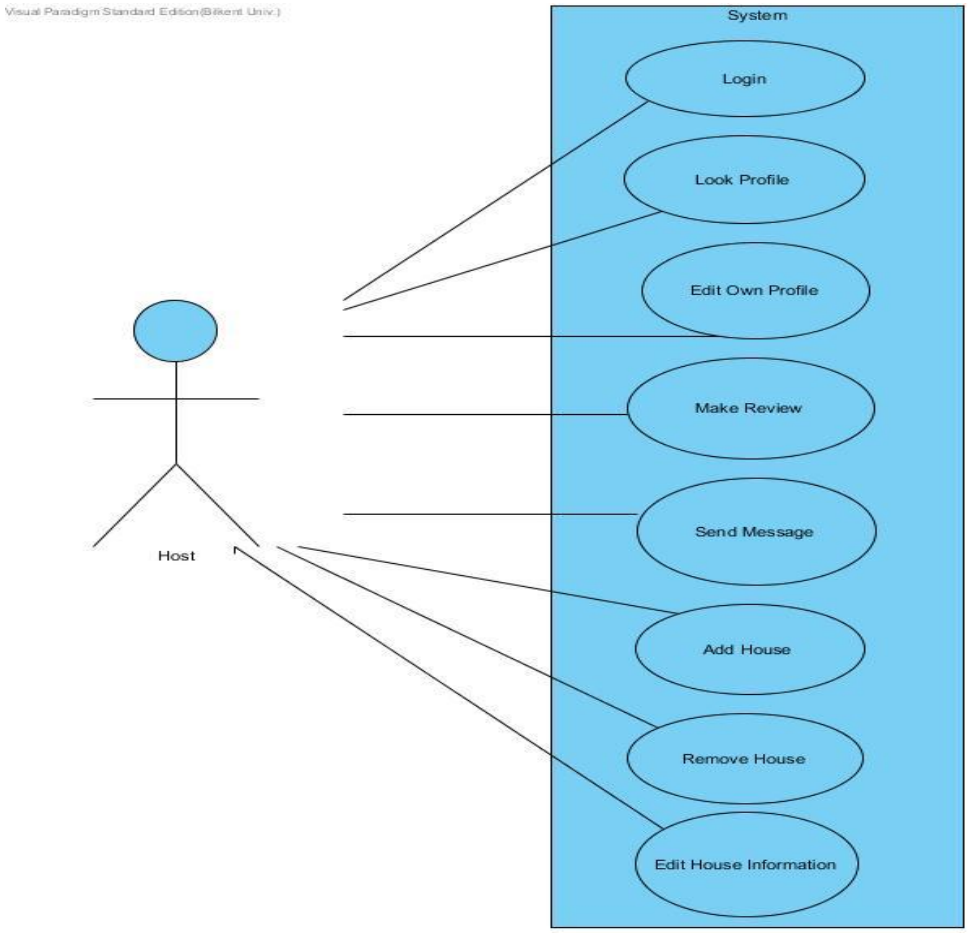
```
`guestID` INT NOT NULL,
 `accommodationID` INT NOT NULL,
PRIMARY KEY (`reviewID`),
  INDEX `guestID_idx` (`guestID` ASC),
INDEX `accommodationID_idx` (`accommodationID` ASC),
CONSTRAINT `reviewID`
  FOREIGN KEY (`reviewID`)
  REFERENCES `mydb`.`Review` (`reviewID`)
 ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT `guestID`
 FOREIGN KEY (`guestID`)
 REFERENCES `mydb`.`Guest` (`guestID`)
 ON DELETE CASCADE
ON UPDATE CASCADE,
  CONSTRAINT `accommodationID`
   FOREIGN KEY (`accommodationID`)
REFERENCES `mydb`.`Accommodation` (`accommodationID`)
  ON DELETE CASCADE
   ON UPDATE CASCADE)
ENGINE = InnoDB;
```

3. Functional Components

3.1 Use Cases

Host:

- Can login to the system with his/her id and password.
- Can look other profile pages.
- Can edit his/her own profile page.
- Can make reviews about his/her customers after the accommodation period ends.
- Can send messages to other users.
- Can add/delete new houses for accommodation.
- Can offer houses for certain dates.
- Can edit houses' information like prices, room numbers, and bed numbers.



Guest

• Can login to the system with is id and password.

• Can look other profile pages.

• Can edit his/her own profile page.

• Can make reviews about his/her hosts and houses' situation after the accommodation period ends.

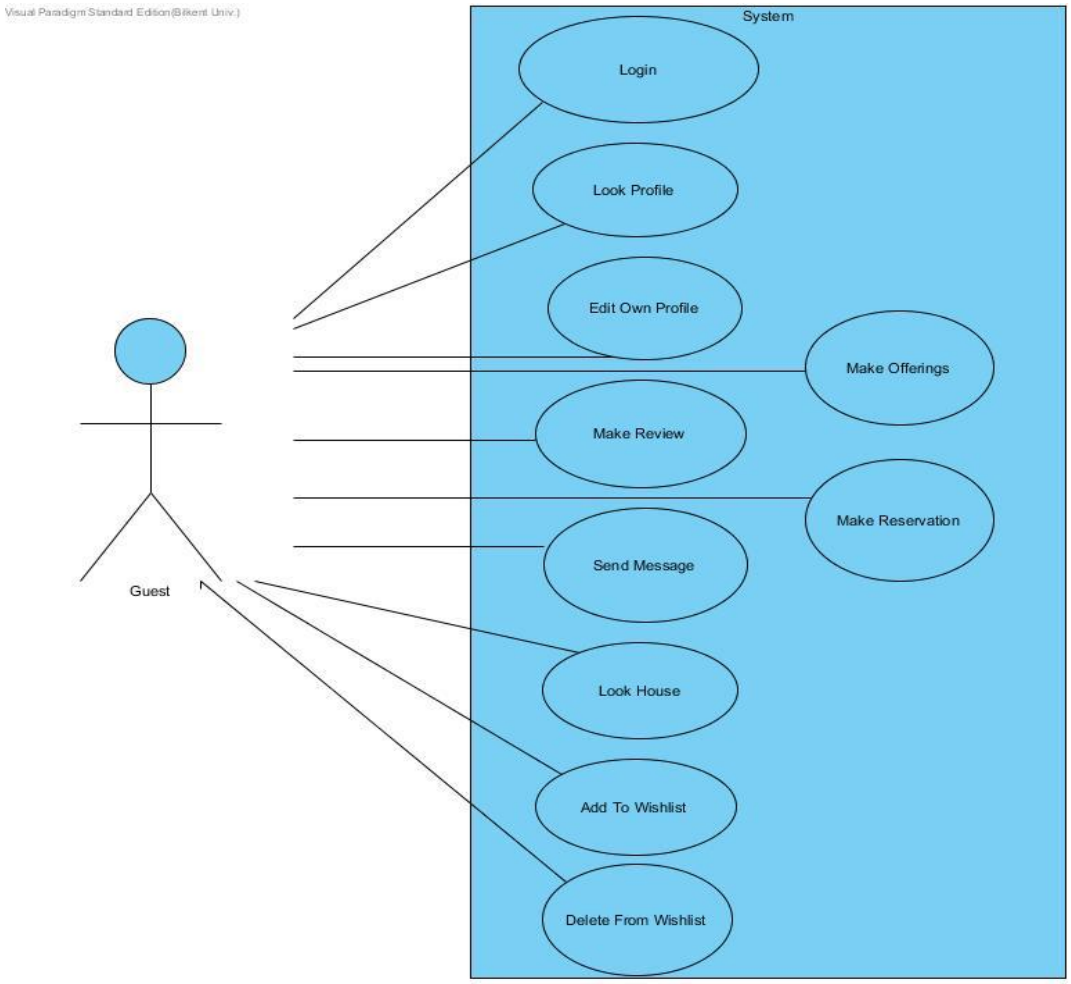
• Can look houses for reservation.

• Can send messages to other users.

• Can add/delete new houses for wish list.

• Can make offerings for houses to rent.

• Can make reservations for desired houses.



3.2 Data Structures

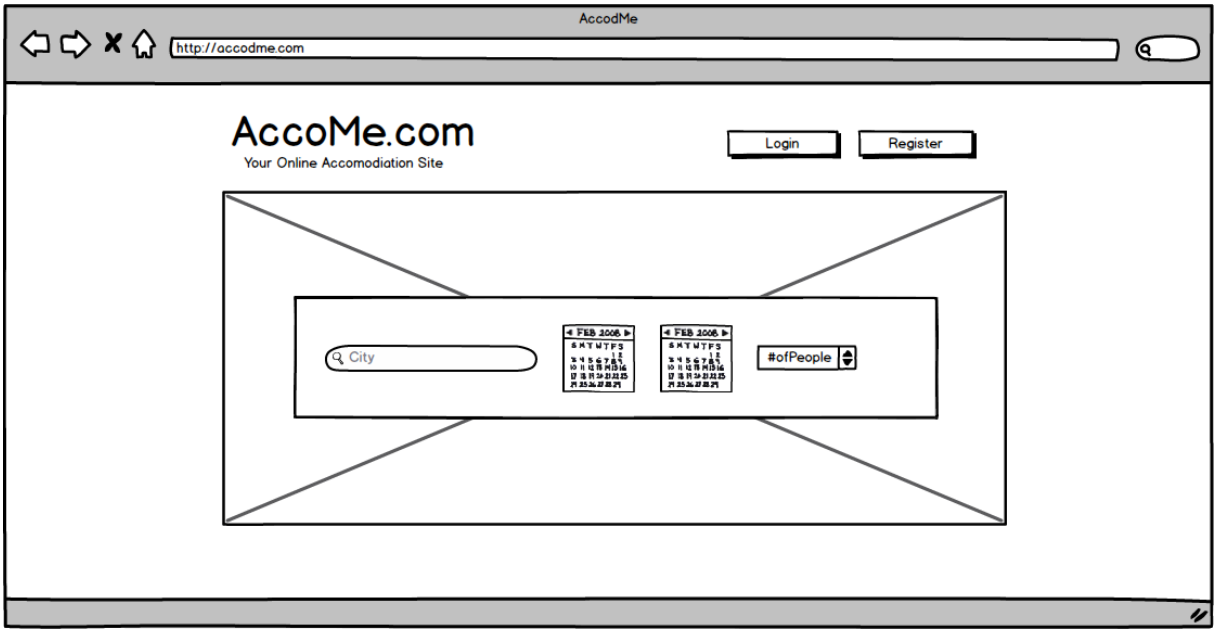
We use some of SQL's built-in data types such as: int, varchar, date, boolean and blob.

The IDs in tables will be auto-incremented with each insert.

4. User Interface Design and SQLs

4.1 Entering Screen

When the user connects to the website, this screen will be the first screen they will encounter. Through this screen, they could do a quick search, login or register to the system.

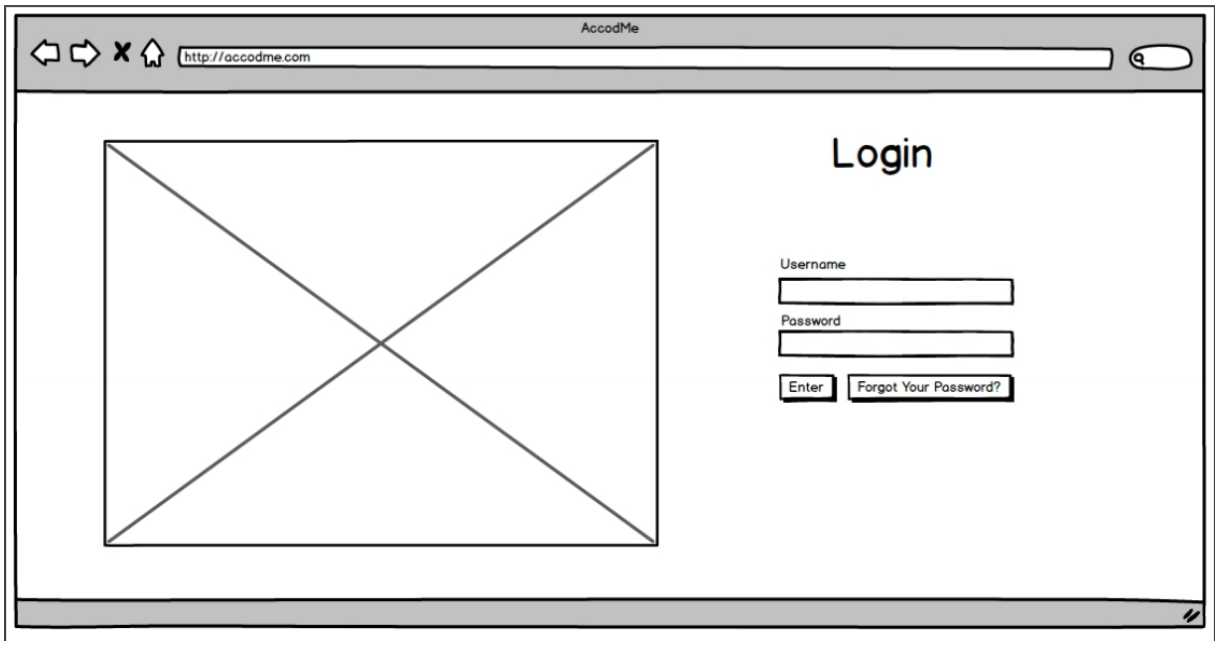


Inputs: $location, $startDate, $endDate, $guestNumber

```
SELECT A.photo, O.price, A.accommodationID, A.bed_number, A.guest_number, A.rating
FROM Accommodation A, House H, Room R, Accommodation_address AD, Offering O
WHERE (A.accommodationID = R. accommodationID or A.accommodationID =
        H.accommodationID)
        and A.accommodationID = O.accommodationID
        and $startDate >= O.startDate
        and $endDate <= O.endDate
        and A.accommodationID = AD. accommodationID
         and  ($location = AD.country or $location = AD.city
        and $guestNumber <= A.guestNumber);
```

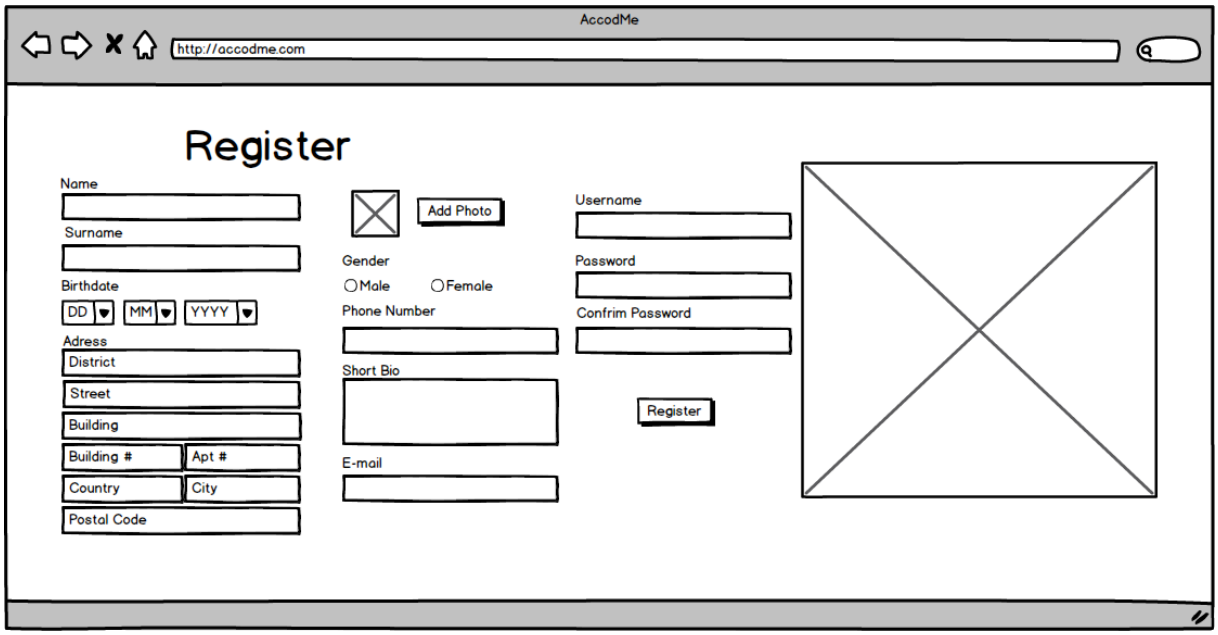
4.2 Login Screen

When the user clicks login, there will be a screen to enable the user to enter the system via their registered information. The user may log in via entering their credentials or may click the “forgot password” button to retrieve their password.



4.3 Register Screen

The register screen enables the user to register to the system. The user has to enter their name, surname, birthdate, address, gender, phone number, bio, email, username, password and photo to register to the system.



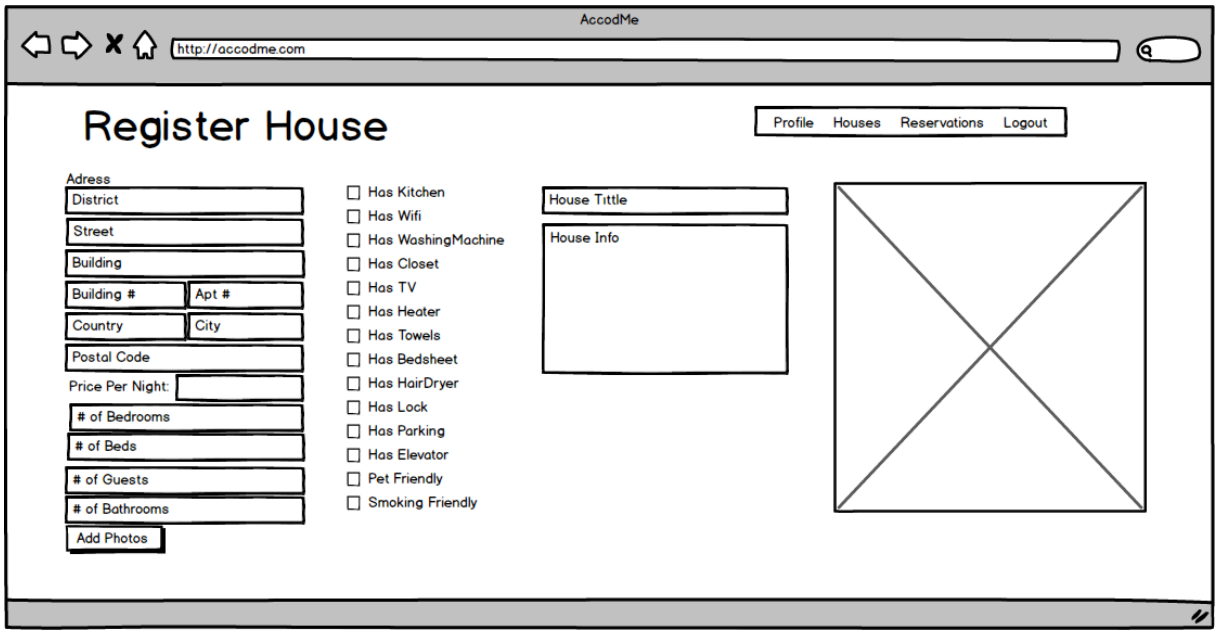
Inputs: $name, $surname, $birthDate, $phoneNumber, $bio, $email, $userName, $password, $photo, $country, $city, $district, $street, $buildingNumber, $postcode

INSERT INTO User (userName, name, surname, email, birthDate, gender, photo, short_bio)
SELECT ($userName, $name, $surname, $email, $birthDate, $gender, $photo, $bio)

```
FROM dual
WHERE (NOT EXISTS(SELECT *
                  FROM User U
                  WHERE U.userName = $userName))
and (NOT EXISTS(SELECT *
                FROM User U
                WHERE U.email = $email));



INSERT INTO Login(userID, password) VALUES
((SELECT userID
FROM User U
WHERE U.userName = $userName), $password);

INSERT INTO Phone(userID, phone_number) VALUES
((SELECT userID
FROM User U
WHERE U.userName = $ username), $phoneNumber);

INSERT INTO user_address(userID, country, city, district, street, building_number, postcode)
VALUES
((SELECT userID
  FROM User U
  WHERE U.userName = $userName), $country, $city, $district, $street, $buildingNumber,
$postcode);
```

4.4 Register House Screen

If the user wants to register a house under his/her name, then he will have to fill out this form for the house details.



Inputs: $userID, $title, $description, $country, $city, $district, $street, $buildingNumber, $postcode, $numOfBeds, $maxNumOfGuests, $numOfBathrooms, $hasKitchen, $hasWifi, $hasWashingMachine, $hasCloset, $hasTV, $hasHeater, $hasTowels, $hasBedSheet, $hasHairDryer, $hasLock, $hasParking, $hasElevator, $petFriendly, $smokingFriendly, $photo

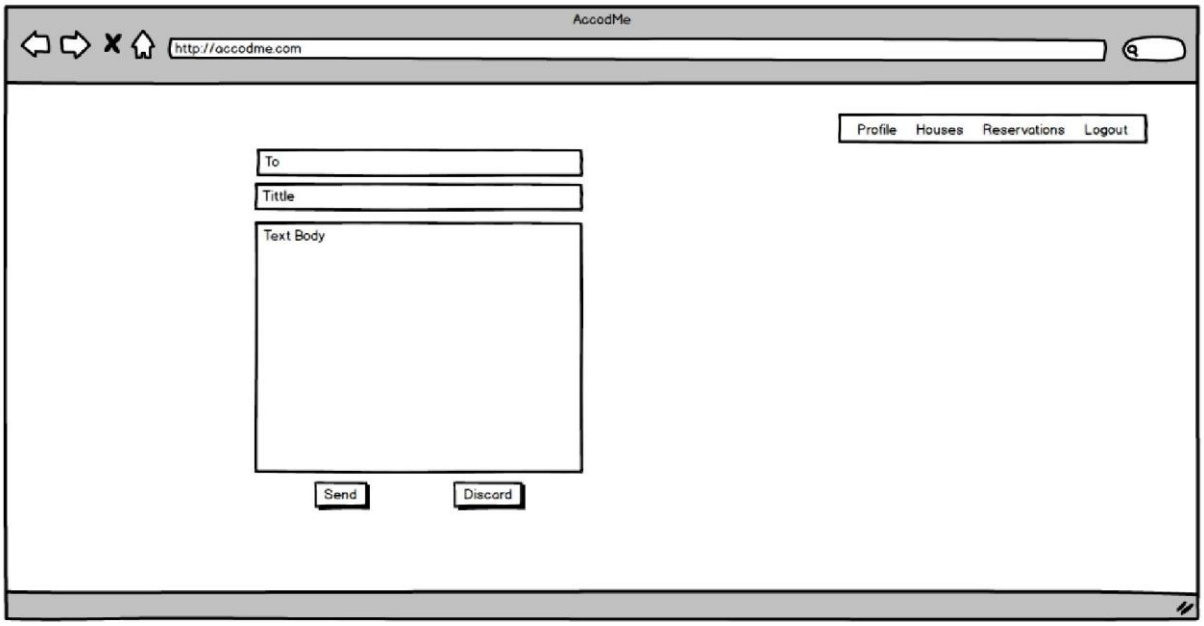
```
INSERT INTO Accommodation (hostID, title, bed_number, guest_number, bath_number,
                description)
VALUES ($userID, $title, $numOfBeds, $maxNumOfGuests, $birthDate, $numOfBathrooms,
                $description);
```

```
INSERT INTO Accommodation_address(accommodationID,country, city, district, street,
building_number, postcode)
VALUES((SELECT max(accommodationID)
                FROM Accommodation A
                GROUP BY accommodationID), $country, $city, $district, $street, $buildingNumber,
$postcode);
```

```
INSERT INTO Amenity(accommodationID, hasWifi, hasKitchen, hasWashingMachine,
hasCloset, hasTV, hasHeater, hasTowels, hasBedSheet, hasHairDryer, hasLock, hasParking,
hasElevator, petFriendly, smokingFriendly)
VALUES((SELECT max(accommodationID)
                FROM Accommodation A
                GROUP BY accommodationID), $hasKitchen, $hasWifi, $hasWashingMachine,
$hasCloset, $hasTV, $hasHeater, $hasTowels, $hasBedSheet, $hasHairDryer, $hasLock,
$hasParking, $hasElevator, $petFriendly, $smokingFriendly);
```

4.5 Message Send Screen

If the user wants to send a message, then this send page will open up. The user needs to enter the receiver, the title and the message body. Then he/she can send or discard the message at will.



Inputs: $userName, $To, $title, $textBody, $date

BEGIN ATOMIC

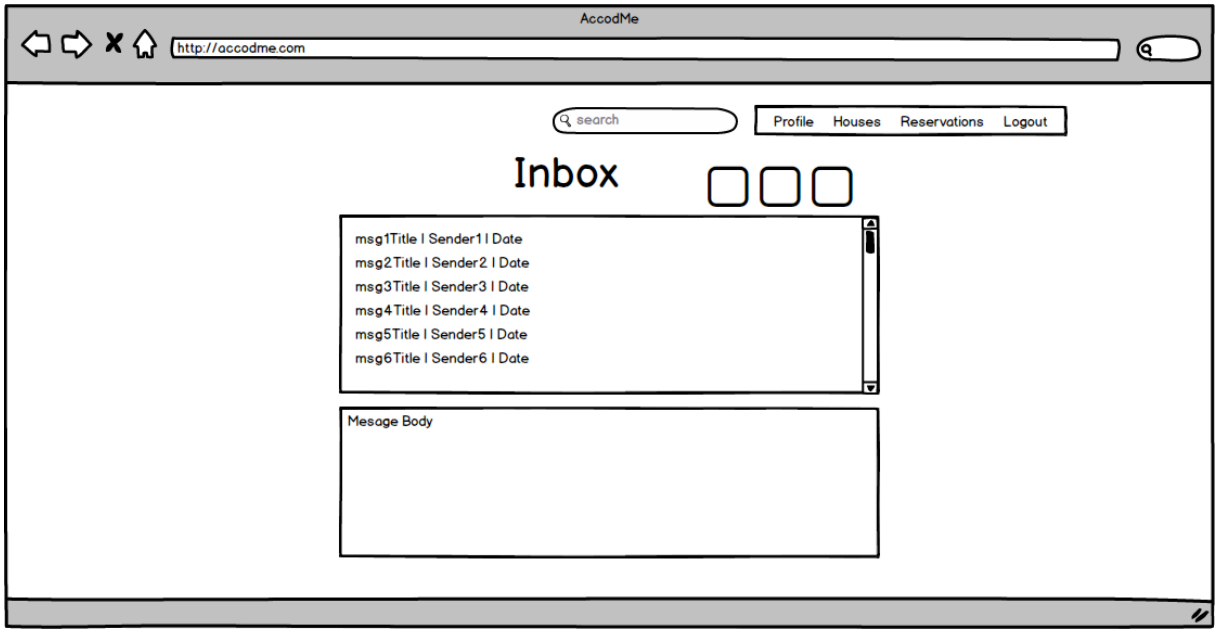
INSERT INTO Message(message_title, message_body, date)
VALUES($title, $textBody, $date);

INSERT INTO User_message((SELECT max(messageID)
FROM Message
GROUP BY messageID),
(SELECT userID
FROM User
WHERE User.userName = $userName),
(SELECT userID
FROM User
WHERE User.userName = $To));

END;

4.6 Message Inbox Screen

The user can view their inbox and see their messages. Then, he/she can reply to the message or delete it.



Inputs: $userID

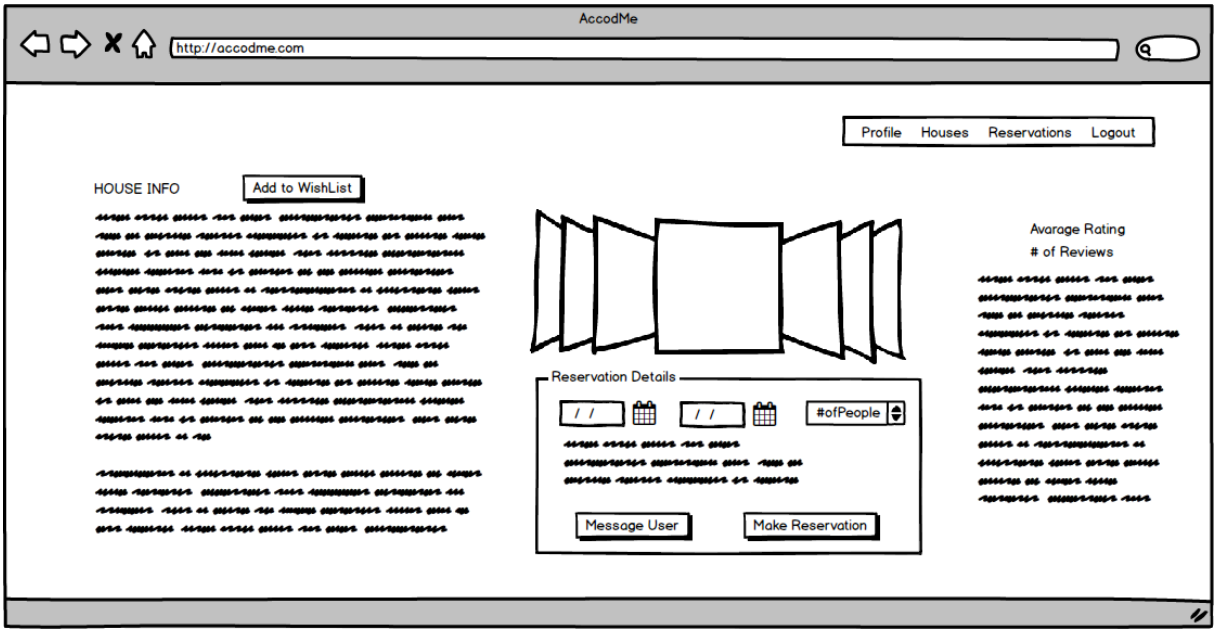
SELECT M.message_title, U.sendID, M.date
FROM Message M, User_message U
WHERE U.receiveID = $userID and M.messageID = U.messageID

Inputs: $messageID

SELECT M.message_body
FROM Message M
WHERE M.messageID = $messageID

4.7 Reservation Screen

The user can view the houses and their ratings and add them to a wishlist, message the host, and make a reservation in this screen.



Inputs: $accommodationID
SELECT *
FROM Accommodation A
WHERE A.accommodationID = $accommodationID;

Inputs: $guestID, $accommodationID
INSERT INTO Wishes(guestID, accommodationID)
VALUES ($guestID, $accommodationID);

Inputs: $startDate, $endDate, $numOfPeople, $accommodationID, $userID
INSERT INTO Reservation(guestID, accommodationID, startDate, endDate, price)
VALUES($userID, $accommodationID, $startDate, $endDate,
(SELECT O.price * ($endDate-$startDate)
FROM Offering O
WHERE O.accommodationID = $accommodationID and $startDate>= O.start_date
and $endDate <= O.endDate));

Inputs: $accommodationID

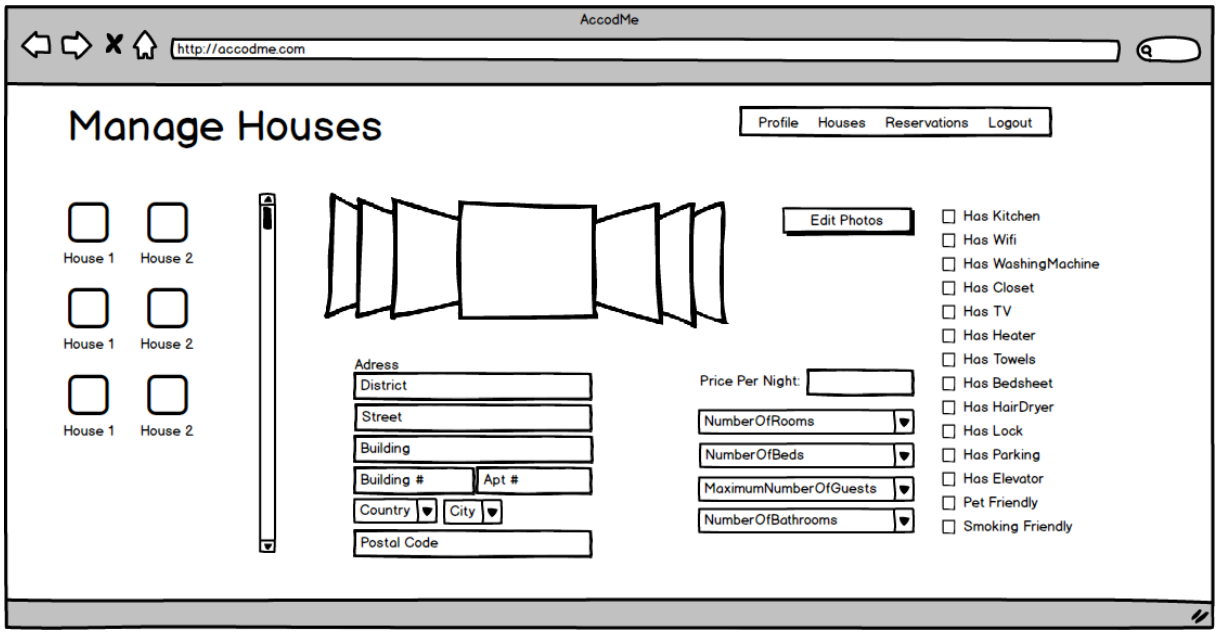
SELECT rating
FROM Accommodation A
WHERE A.accommodationID = $accommodationID

Inputs: $accommodationID
SELECT count(G.reviewID)
FROM Guest_review G
WHERE G.accommodationID = $accommodationID
GROUP BY G.accommodationID

Inputs: $accommodationID
SELECT User.name, Review.date, Review.comment
FROM User, Guest_review, Review, Guest
WHERE Guest_review.accommodationID = $accommodationID
and Guest_review.reviewID = Review.reviewID
and User.userID = Guest.guestID
and Guest_review.guestID = Guest.guestID;

4.8 Manage Houses Screen

The hosts can change the information about their houses in this screen.



Inputs: $userID

SELECT A.title

FROM Accommodation A, Host H
WHERE A.hostID = H.hostID and H.hostID = $userID

Inputs: $userID, $accommodationID, $title, $description, $country, $city, $district, $street, $buildingNumber, $postcode, $numOfBeds, $maxNumOfGuests, $numOfBathrooms, $hasKitchen, $hasWifi, $hasWashingMachine, $hasCloset, $hasTV, $hasHeater, $hasTowels, $hasBedSheet, $hasHairDryer, $hasLock, $hasParking, $hasElevator, $petFriendly, $smokingFriendly, $photo

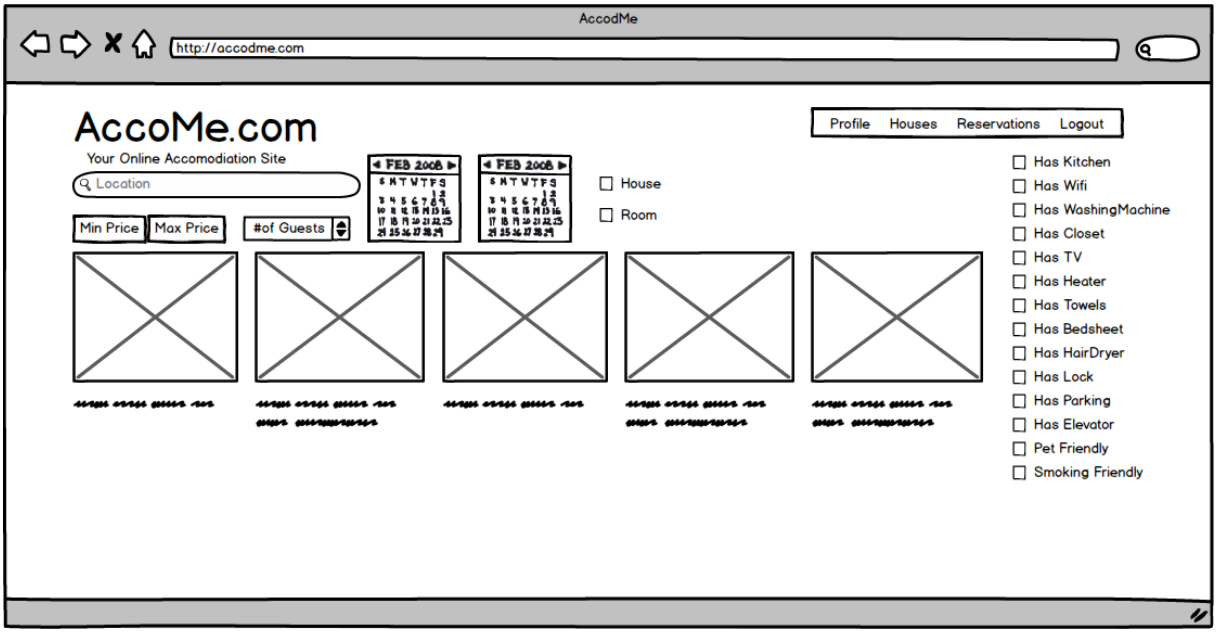
UPDATE Accommodation
SET title = $title, bed_number = $numOfBeds, guest_number = $maxNumOfGuests, bath_number = $numOfBathrooms
WHERE accommodationID = $accommodationID;

UPDATE Accommodation_address
SET country = $country, city = $city, district = $district, street = $street, building_number = $buildingNumber, postcode = $postcode)
WHERE accommodationID = $accommodationID;

UPDATE Amenity
SET hasWifi = $hasWifi, hasKitchen = $hasKitchen, hasWashingMachine = $hasWashingMachine, hasCloset = $hasCloset, hasTV = $hasTV, hasHeater = $hasHeater, hasTowels = $hasTowels, hasBedSheet = $hasBedSheet, hasHairDryer = $hasHairDryer, hasLock = $hasLock, hasParking= $hasParking, hasElevator = $hasElevator, petFriendly = $petFriendly, smokingFriendly = $smokingFriendly
WHERE accommodationID = $accommodationID;

4.9 Advanced Search Screen

In the advanced search screen, the user will search houses in regards of his/her specific requirements. This screen will filter out the houses according to these advanced search requirements.



Inputs: $location, $startDate, $endDate, $guestNumber, $House, $minPrice, $maxPrice

SELECT A.photo, O.price, A.accommodationID, A.bed_number, A.guest_number, A.rating
FROM Accommodation A, House H, Accommodation_address AD, Offering O
WHERE (A.accommodationID = H.accommodationID)
and A.accommodationID = O.accommodationID
and $startDate >= O.startDate
and $endDate <= O.endDate
and A.accommodationID = AD. accommodationID
and ($location = AD.country or $location = AD.city
and $guestNumber <= A.guestNumber
and O.price <=$maxPrice
and O.price >= $minPrice);

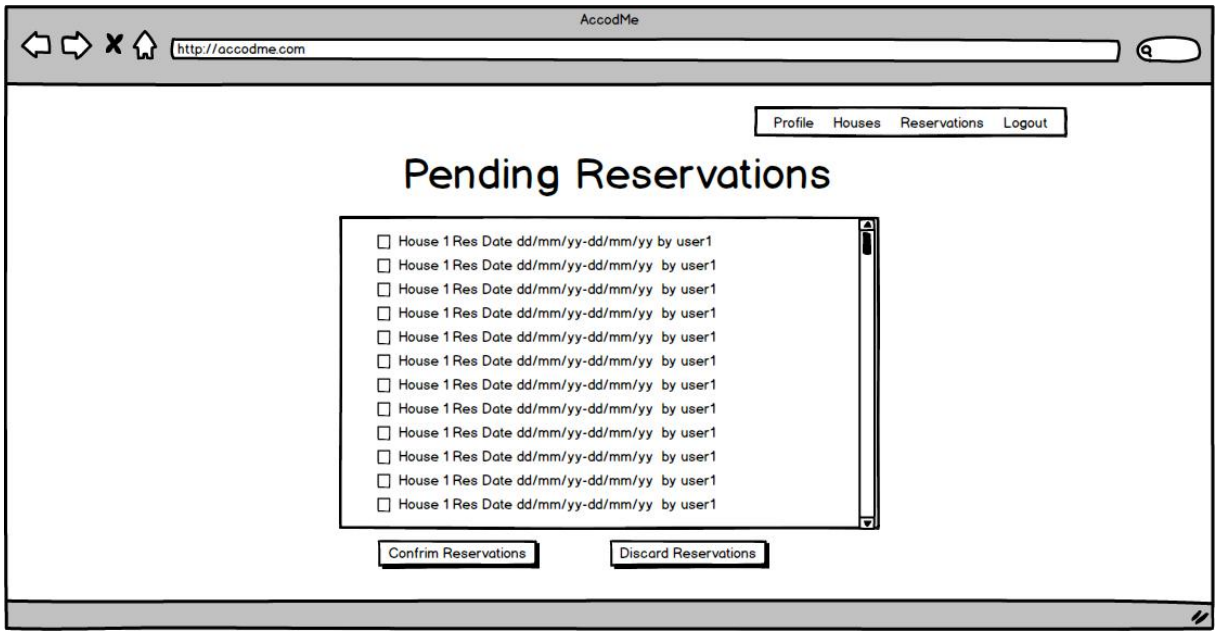
Inputs: $location, $startDate, $endDate, $guestNumber, $Room, $minPrice, $maxPrice, $numOfGuests

SELECT A.photo, O.price, A.accommodationID, A.bed_number, A.guest_number, A.rating
FROM Accommodation A, Room R, Accommodation_address AD, Offering O
WHERE (A.accommodationID = R.accommodationID)
and A.accommodationID = O.accommodationID
and $startDate >= O.startDate
and $endDate <= O.endDate
and A.accommodationID = AD. accommodationID
and ($location = AD.country or $location = AD.city

and $guestNumber <= A.guestNumber
and O.price <=$maxPrice
and O.price >= $minPrice);
);

4.10 Reservation Confirmation Screen

The host has to approve the reservations of the users that will stay in their house. To enable that, the user will have the opportunity to view these requests and approve it.

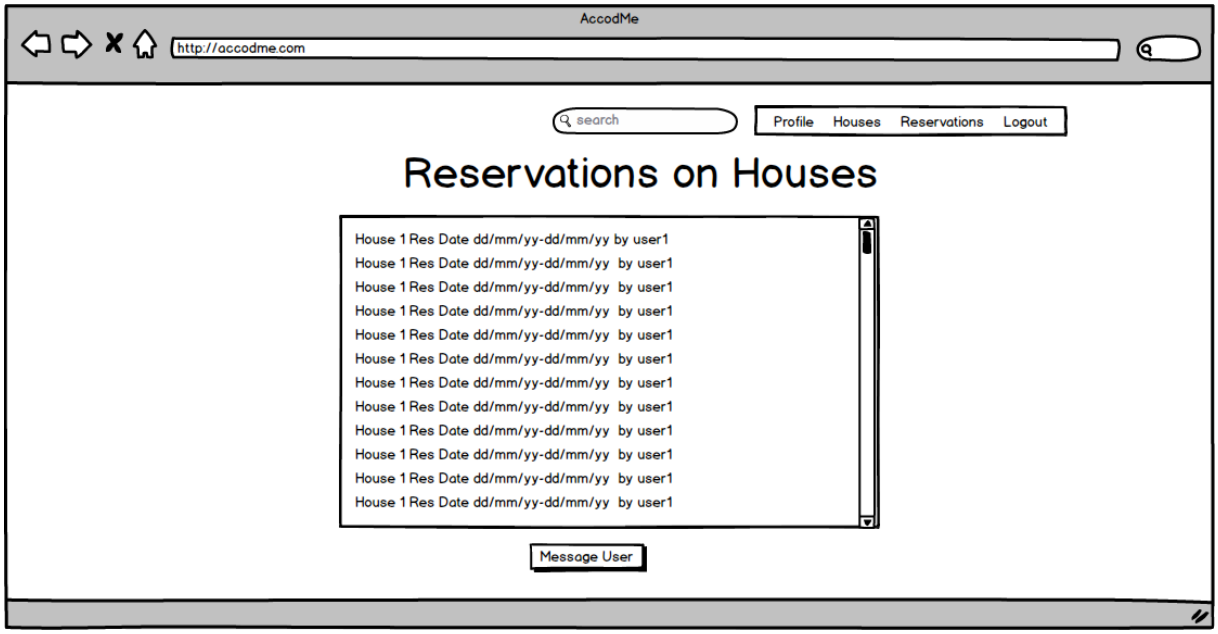


Inputs: $userID
SELECT A.title, R.startDate, R.endDate, U.userName
FROM Reservation R, User U, Guest G
WHERE R.accommodationID = A.accommodationID
and A.hostID = $userID
and G.guestID = R.guestID
and G.guestID = U.userID;

Inputs: $confirm, $reservationID
UPDATE Reservation
SET confirm = $confirm
WHERE reservationID = $reservationID

4.11 Show Reservation Screen

In this screen the host can view the reservations that he has approved and message the user that has reserved the house.

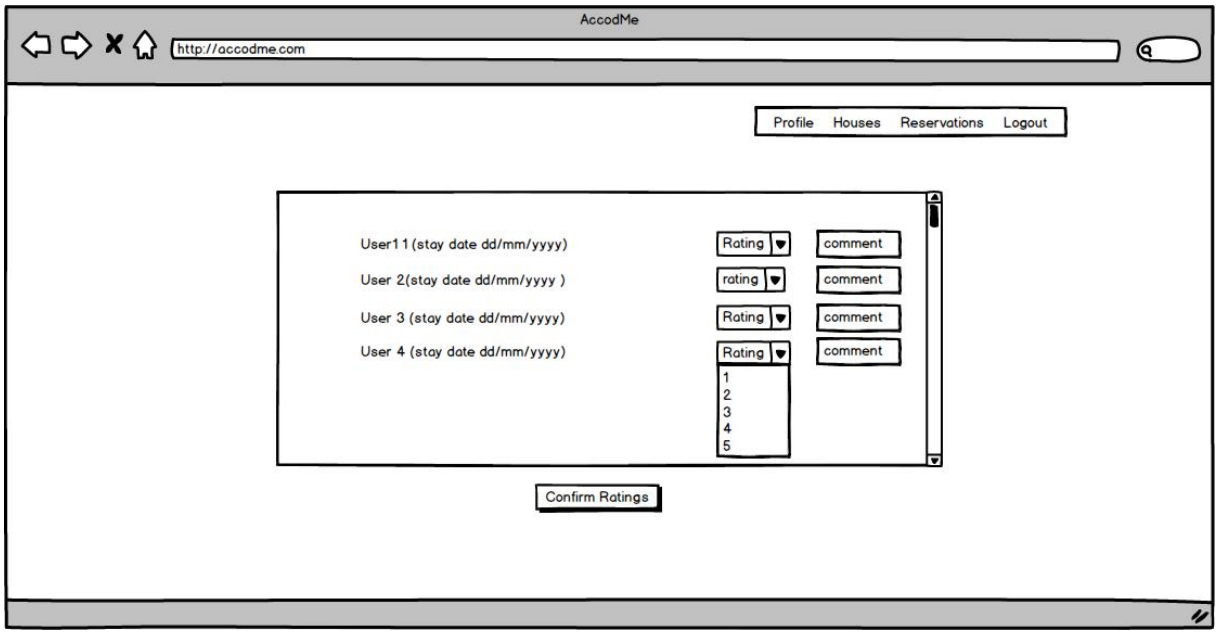


Inputs: $userID

```
SELECT A.title, R.startDate, R.endDate, U.userName
FROM Reservation R, User U, Guest G
WHERE R.accommodationID = A.accommodationID
        and A.hostID = $userID
        and G.guestID = R.guestID
        and G.guestID = U.userID;
```

4.12 Accommodation Rating Screen

The hosts will rate the users via this screen. As the hosts have the opportunity to decline a reservation they might decide upon the user's rating.



Inputs: $userID, $currentDate

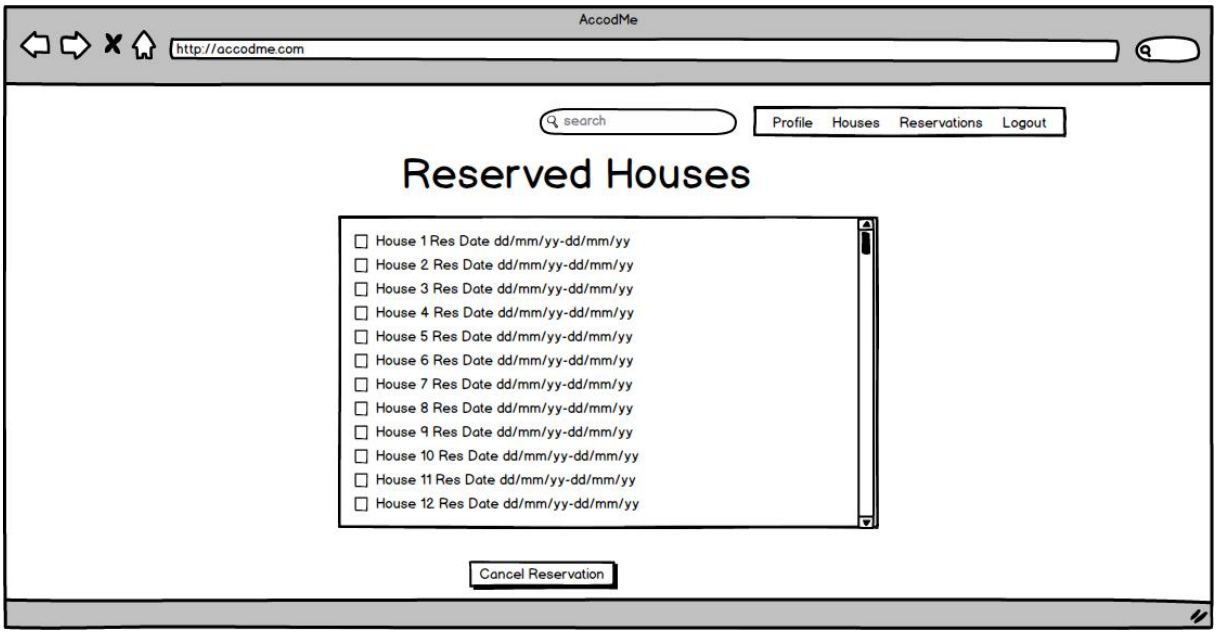
```
SELECT A.title, R.startDate, R.endDate, AD.city
FROM Reservation R, User U, Guest G, Accommodation_address AD
WHERE R.accommodationID = A.accommodationID
        and A.hostID = $userID
        and G.guestID = R.guestID
        and G.guestID = U.userID
        AD.accommodationID = A.accmomodationID
        and R.endDate <= $currentDate;
```

Inputs: $userID, $accommodationID, $rating, $comment

```
BEGIN ATOMIC

INSERT INTO Review(rating, comment)
VALUES($rating, $comment);

INSERT INTO guestReview( reviewID, guestID, accommodationID)
VALUES((SELECT max(reviewID)
            FROM Review
            GROUP BY reviewID), $userID, $accommodationID);

END
```

4.13 Cancel Reservation Screen

The users can view their reservation and cancel it via this screen.

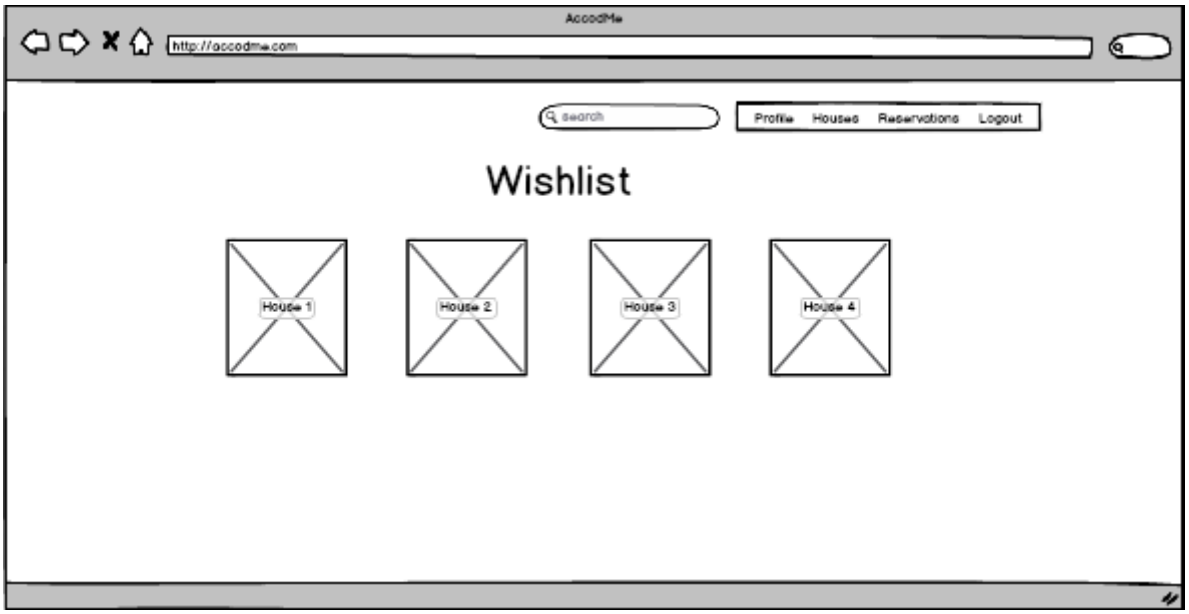


Inputs: $userID
SELECT A.title, R.startDate, R.endDate, U.userName
FROM Reservation R, User U, Guest G
WHERE R.accommodationID = A.accommodationID
and G.guestID = $userID
and G.guestID = R.guestID;

Inputs: $reservationID
DELETE FROM Reservation
WHERE reservationID = $reservationID

4.14 Wishlist

The user can view their wishlisted houses via this menu.




```
select Accodommodation.accommodationID, Accommodation.title
from Accommodation, Photo
where Accommodation.accomodationID = (Select Wishes.accommodationID
                                      from Wishes, User
                                      where User.userID = Wishes.userID) AND
                                Photo.accommodationID = Accommodation.accomodationID
```

5. Advanced Database Components

5.1 Triggers

- As the host enters (inserts) new houses into the system, the attribute 'number of house' should be incremented:

```
create trigger num_of_houses _update after insert on Accommodation
referencing new row as nrow
for each row
begin atomic
    update Host
    set num_of_houses = num_of_houses + 1
    where Host.hostID = nrow.hostID;
end;
```

- If a host gives a review about his/her guest, the rating of that user should be updated:

```
create trigger userrating_update after insert on Host_review
referencing new row as nrow
for each row
when exists (select  R.rating
                    from Review R
                    where nrow.reviewID =R.reviewID )
begin atomic
   update User
   set rating =
       (select avg(rating)
        from Review R, Host_review H
         where R.reviewID=H.reviewID
        group by H.guestID)
   where User.userID = nrow.guestID;
end
```

- If a guest gives a review about the house that he/she has stayed, the rating of that accommodation should be updated:

```
create trigger hostrating_update after insert on Guest_review
referencing new row as nrow
for each row
when exists (select  R.rating
                    from Review R
                    where nrow.reviewID =R.reviewID )
begin atomic
   update Accommodation
   set rating =
       (select avg(rating)
        from Review R, Guest_review G
         where R.reviewID=G.reviewID
        group by G.accommodationID)
   where Accommodation.accommodationID = nrow.accommodationID;
end;
```

5.2 Constraints

• Search for houses does not require a login to the system. However, users have to log in to the system otherwise they cannot make any reservation or add a house.

• Usernames and emails are unique.

• Every house can only have one reservation at the same date.

• A reservation can only be made if the accommodation is on the offerings in the specified dates.

• The reservation should be confirmed by the host of the house.

• Guests cannot edit hosts' houses.

• For every house, there must be a host.

• Hosts cannot edit guests' profile, and guests cannot edit hosts' profile.

• Guests can only add an existing house to their wish list or make a reservation.