

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 4](#_Toc468261730)

[2. Relational Schema 5](#_Toc468261731)

[2.1 User 7](#_Toc468261732)

[2.2 Host 7](#_Toc468261733)

[2.3 Guest 8](#_Toc468261734)

[2.4 Accommodation 8](#_Toc468261735)

[2.5 House 9](#_Toc468261736)

[2.6 Room 10](#_Toc468261737)

[2.7 Amenity 10](#_Toc468261738)

[2.8 Login 11](#_Toc468261739)

[2.9 Phone 12](#_Toc468261740)

[2.10 Message 13](#_Toc468261741)

[2.11 User\_message 13](#_Toc468261742)

[2.12 User\_address 14](#_Toc468261743)

[2.13 Accommodation\_address 15](#_Toc468261744)

[2.14 Photo 16](#_Toc468261745)

[2.15 Offering 16](#_Toc468261746)

[2.16 Reservation 17](#_Toc468261747)

[2.17 Wishes 18](#_Toc468261748)

[2.18 Review 19](#_Toc468261749)

[2.19 Host\_review 19](#_Toc468261750)

[2.20 Guest\_review 20](#_Toc468261751)

[3. Functional Components 21](#_Toc468261752)

[3.1 Use Cases 21](#_Toc468261753)

[3.2 Data Structures 23](#_Toc468261754)

[4. User Interface Design and SQLs 24](#_Toc468261755)

[4.1 Entering Screen 24](#_Toc468261756)

[4.2 Login Screen 24](#_Toc468261757)

[4.3 Register Screen 25](#_Toc468261758)

[4.4 Register House Screen 26](#_Toc468261759)

[4.5 Message Send Screen 28](#_Toc468261760)

[4.6 Message Inbox Screen 28](#_Toc468261761)

[4.7 Reservation Screen 30](#_Toc468261762)

[4.8 Manage Houses Screen 31](#_Toc468261763)

[4.9 Advanced Search Screen 32](#_Toc468261764)

[4.10 Reservation Confirmation Screen 34](#_Toc468261765)

[4.11 Show Reservation Screen 35](#_Toc468261766)

[4.12 Accommodation Rating Screen 35](#_Toc468261767)

[4.13 Cancel Reservation Screen 37](#_Toc468261768)

[4.14 Wishlist 37](#_Toc468261769)

[5. Advanced Database Components 38](#_Toc468261770)

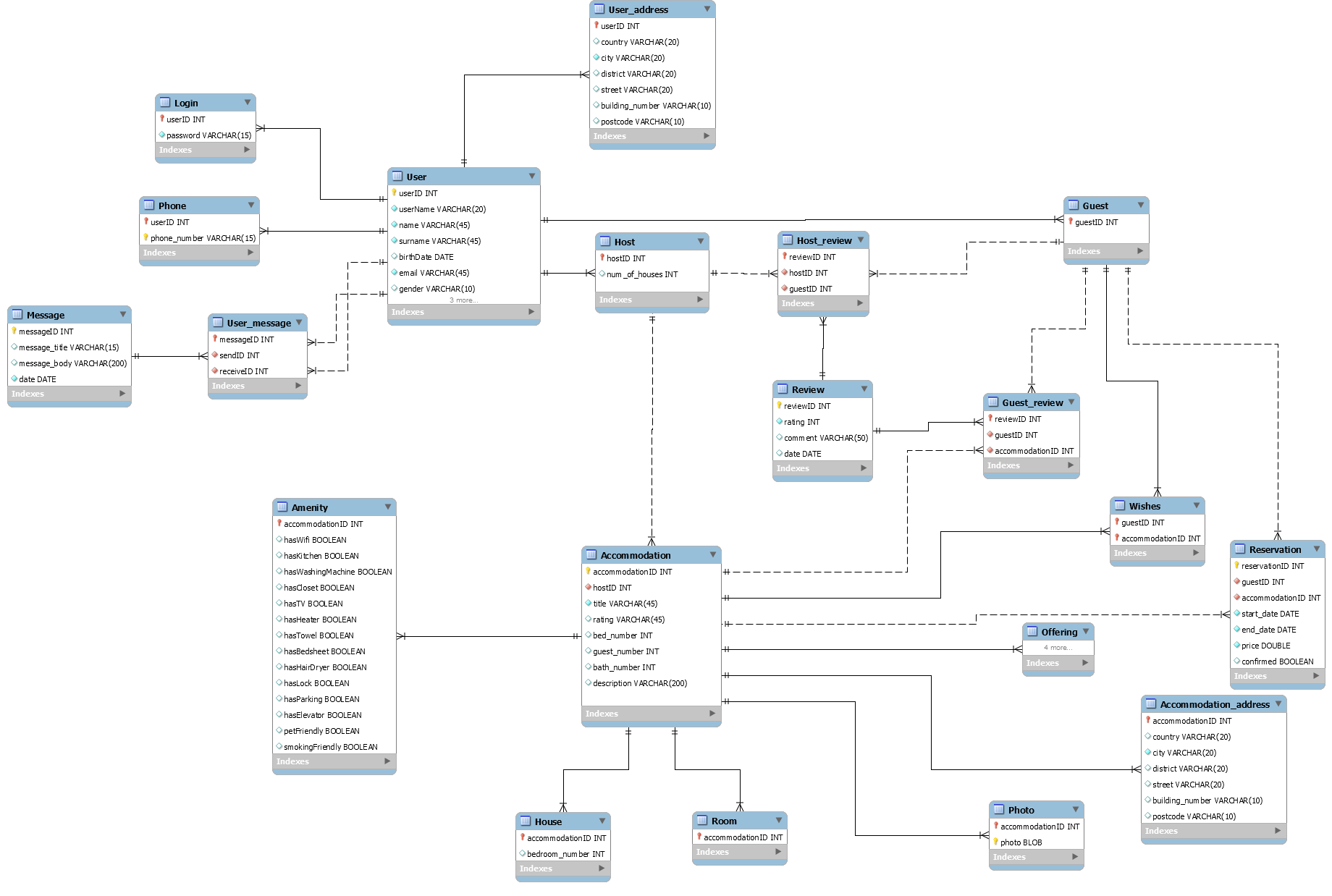
[5.1 Triggers 38](#_Toc468261771)

[2. Constraints 39](#_Toc468261772)

# ER Diagram

er_final.jpg

# Relational Schema



## User

Relational Model

User(userID,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;

## Host

Relational Model

Host(hostID (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;

## 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;

## 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;

## 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;

## 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;

## 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;

## 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;

## 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;

## Message

Relational Model

Message(messageID, 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;

## User\_message

Relational Model

Message(messageID (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;

## User\_address

Relational Model

user\_address(userID (FK by User), country, city, district, street, building\_number, postcode)

Functional Dependencies

userID -> country, city, district, street, building\_number, postcode

Keys

Candidate Keys: { userID }

Primary Key: userID

Foreign Key: userID (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;

## 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;

## 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;

## 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;

## Reservation

Relational Model

Reservation(reservationID, 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;

## 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;

## 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;

## 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;

## Guest\_review

Relational Model

guest\_review(reviewID (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;

# Functional Components

## 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.

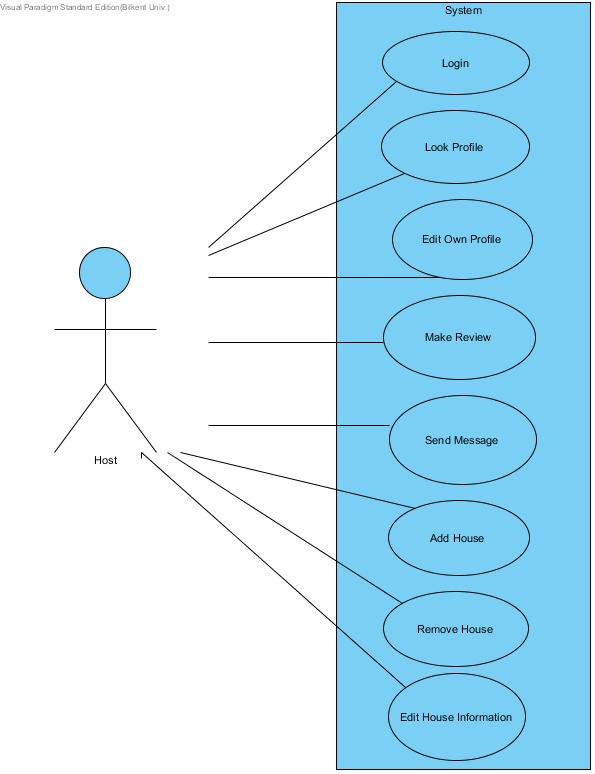
• Can make reviews about his/her customers after the accommodation period ends.

• 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.

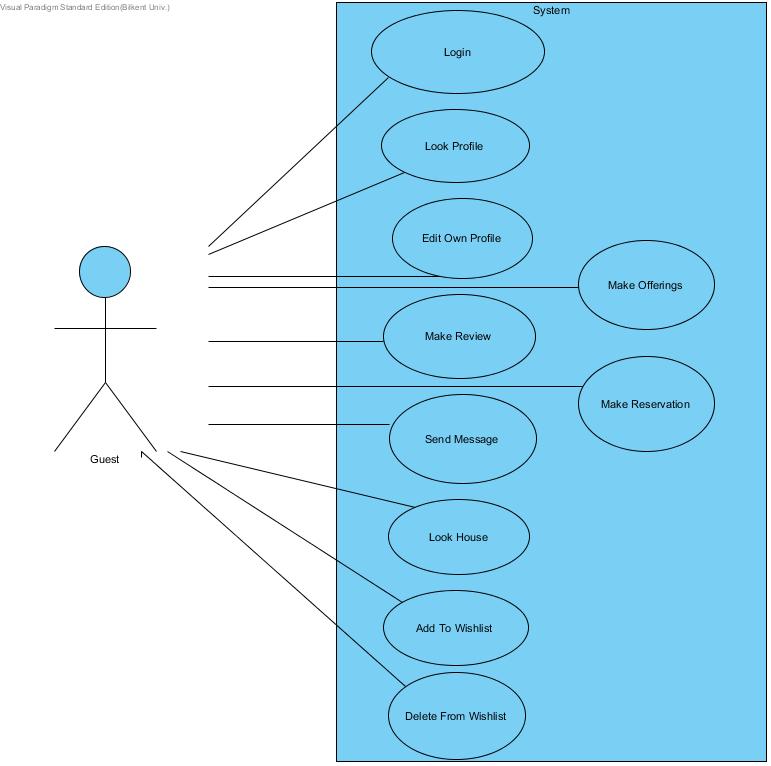
• Can look houses for reservation.

• 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.



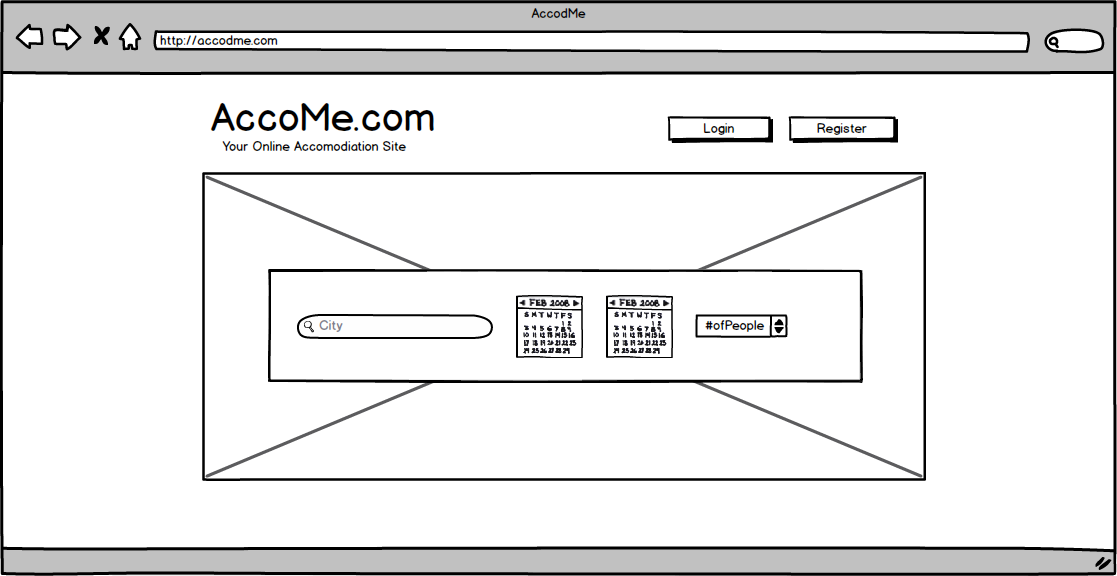
## Data Structures

We use some of SQL's built-in data types such as: int, varchar, date, boolean, blob. The IDs in tables will be auto-incremented with each insert.

# User Interface Design and SQLs

## 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

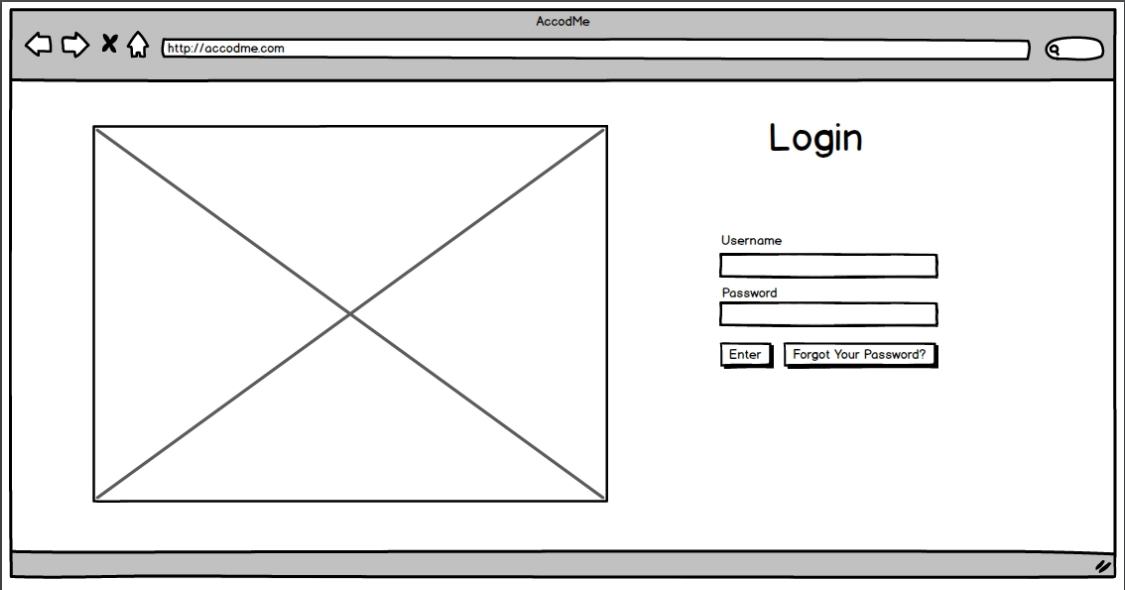
and A.accommodationID = AD. accommodationID

and ($location = AD.country or $location = AD.city

and $guestNumber <= A.guestNumber);

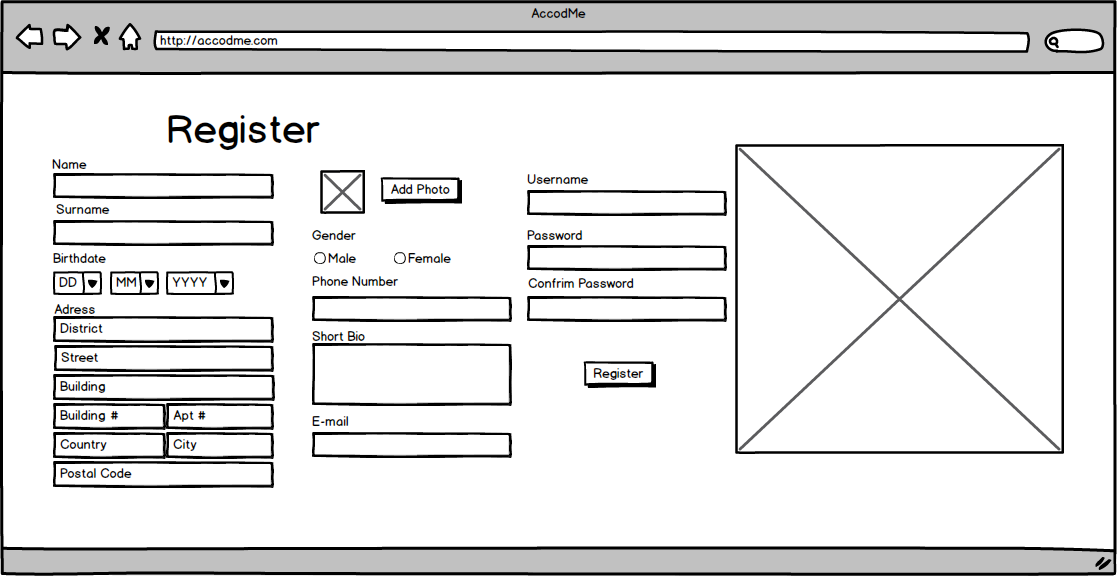
## Login Screen

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.



## 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

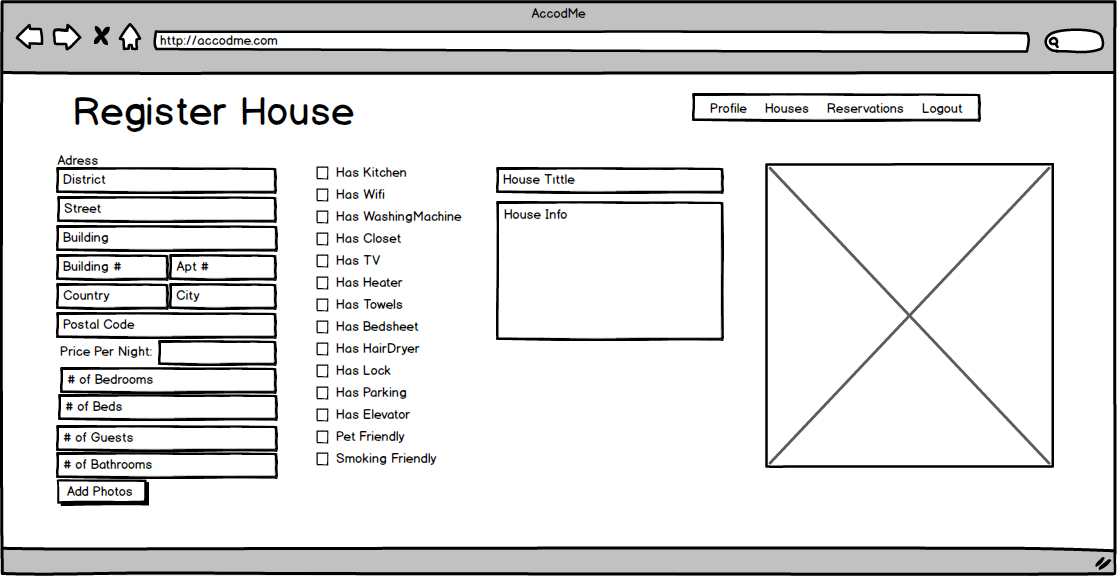
((SELECT userID

FROM User U

WHERE U.userName = $userName), $country, $city, $district, $street, $buildingNumber, $postcode);

## 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);

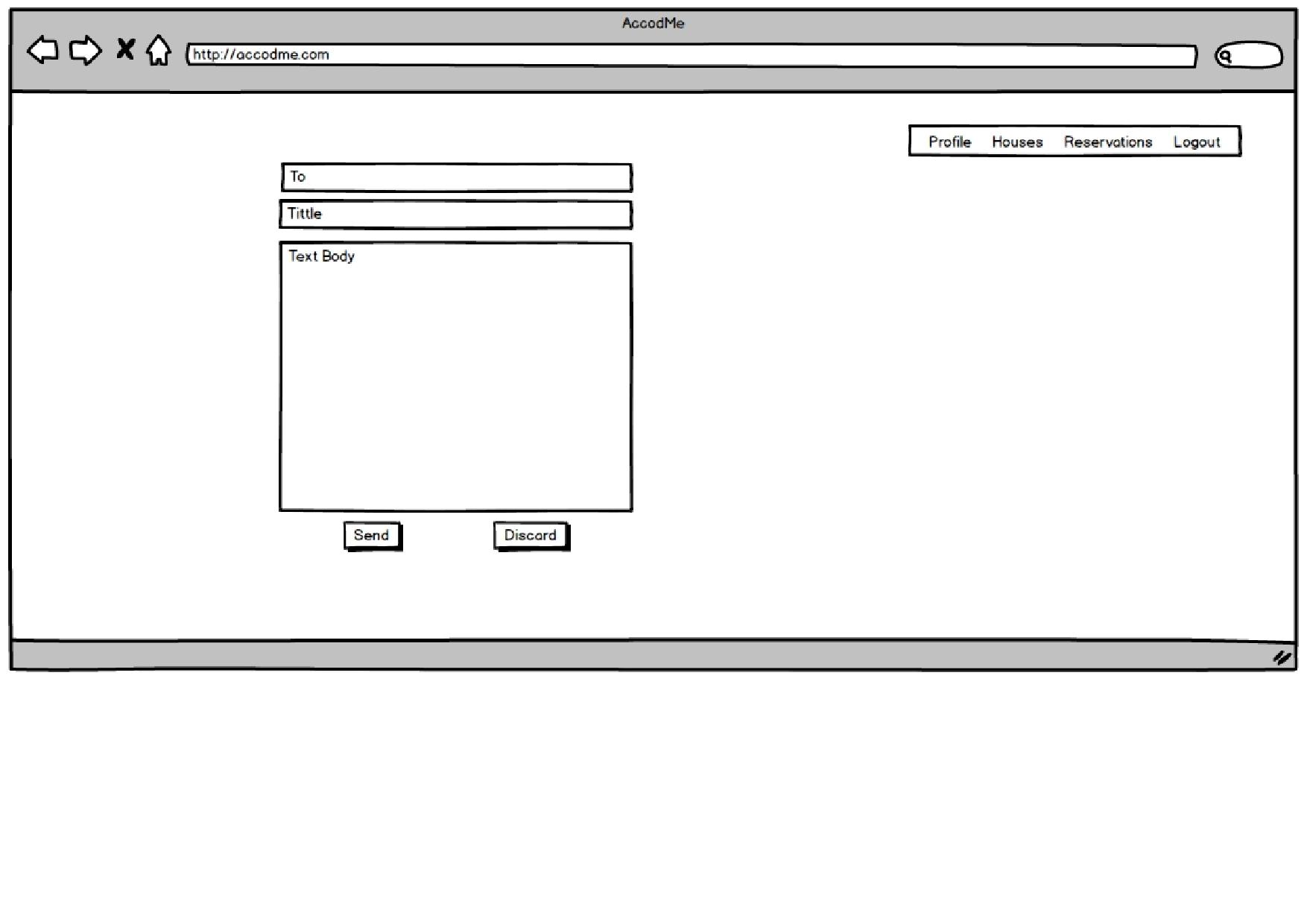
INSERT INTO Amenity(accommodationID, hasWifi, hasKitchen, hasWashingMachine, hasCloset, hasTV, hasHeater, hasTowels, hasBedSheet, hasHairDryer, hasLock, hasParking, hasElevator, petFriendly, smokingFriendly)

VALUES((SELECT max(accommodationID)

FROM Accommodation A

GROUP BY accommodationID), $hasKitchen, $hasWifi, $hasWashingMachine, $hasCloset, $hasTV, $hasHeater, $hasTowels, $hasBedSheet, $hasHairDryer, $hasLock, $hasParking, $hasElevator, $petFriendly, $smokingFriendly);

## 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

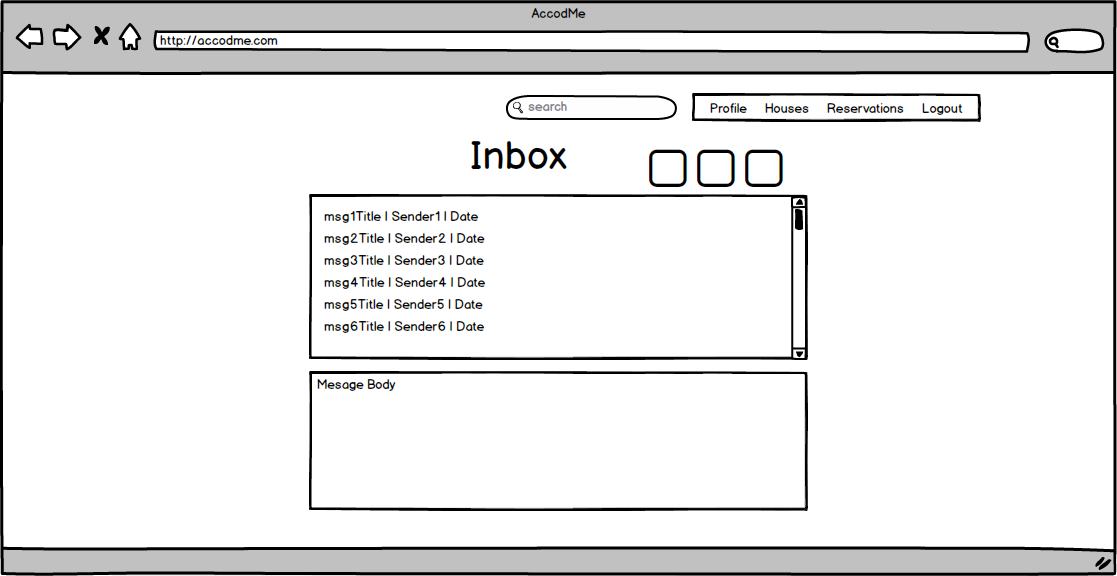
FROM User

WHERE User.userName = $To));

END;

## 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

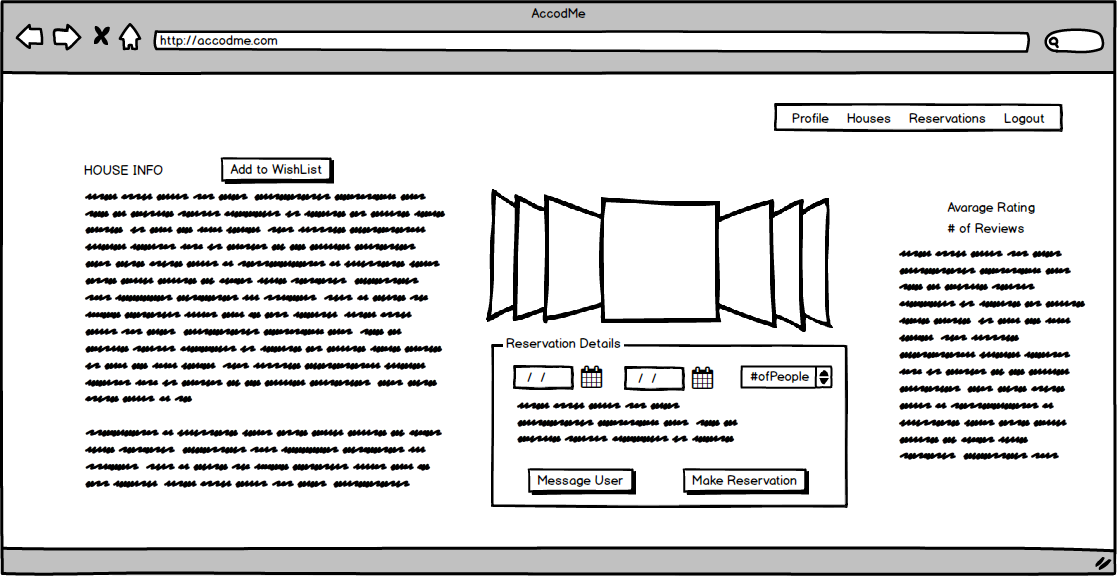
Inputs: $messageID

SELECT M.message\_body

FROM Message M

WHERE M.messageID = $messageID

## 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

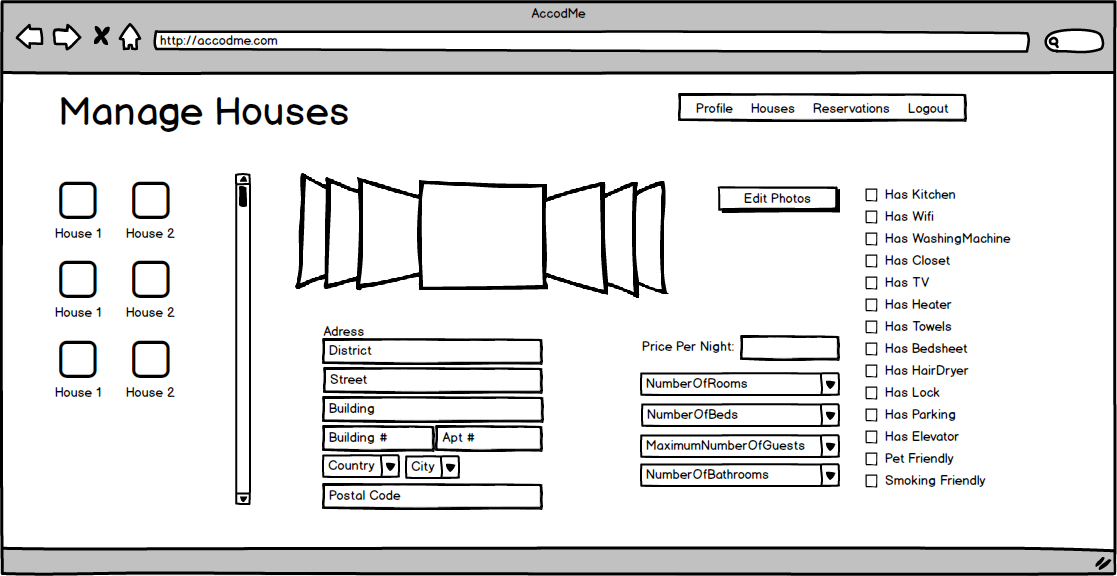
WHERE Guest\_review.accommodationID = $accommodationID

and Guest\_review.reviewID = Review.reviewID

and User.userID = Guest.guestID

and Guest\_review.guestID = Guest.guestID;

## Manage Houses Screen

The hosts can change 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;

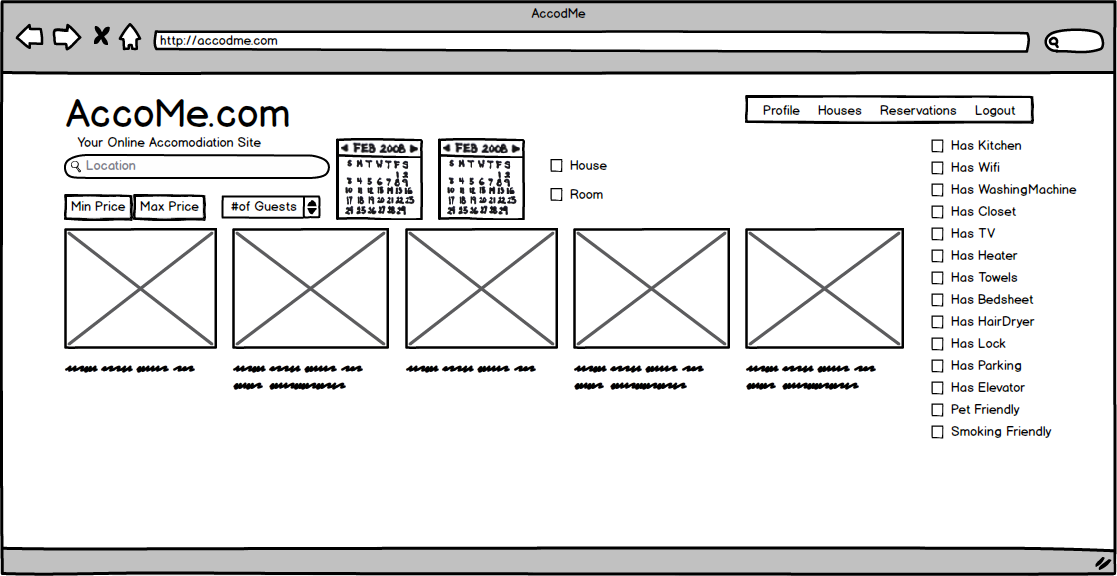
UPDATE Amenity

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;

## 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

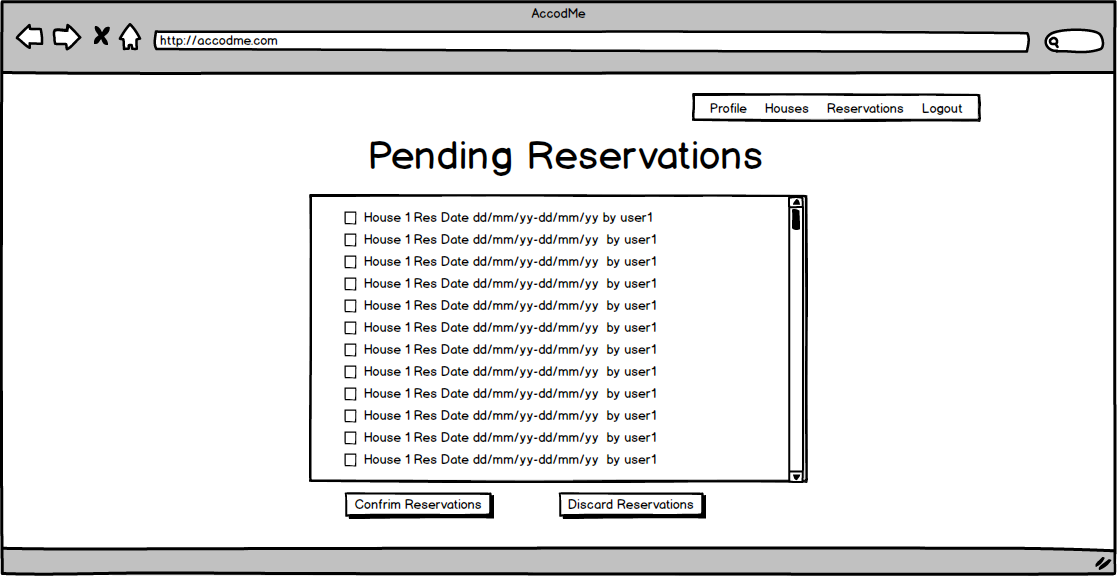
and $guestNumber <= A.guestNumber

and O.price <=$maxPrice

and O.price >= $minPrice);

);

## 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

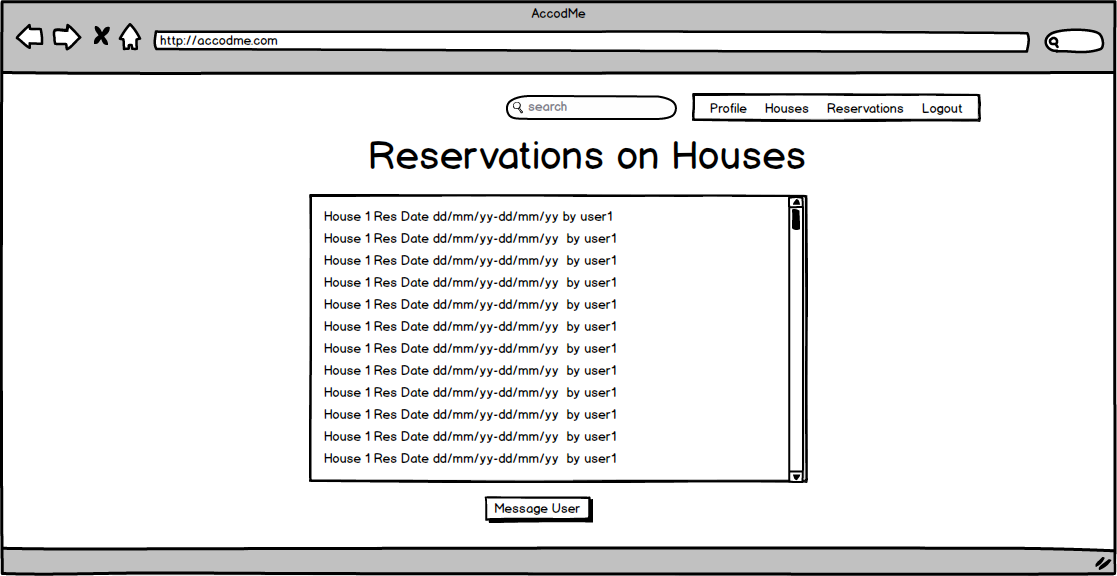
UPDATE Reservation

SET confirm = $confirm

WHERE reservationID = $reservationID

## 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

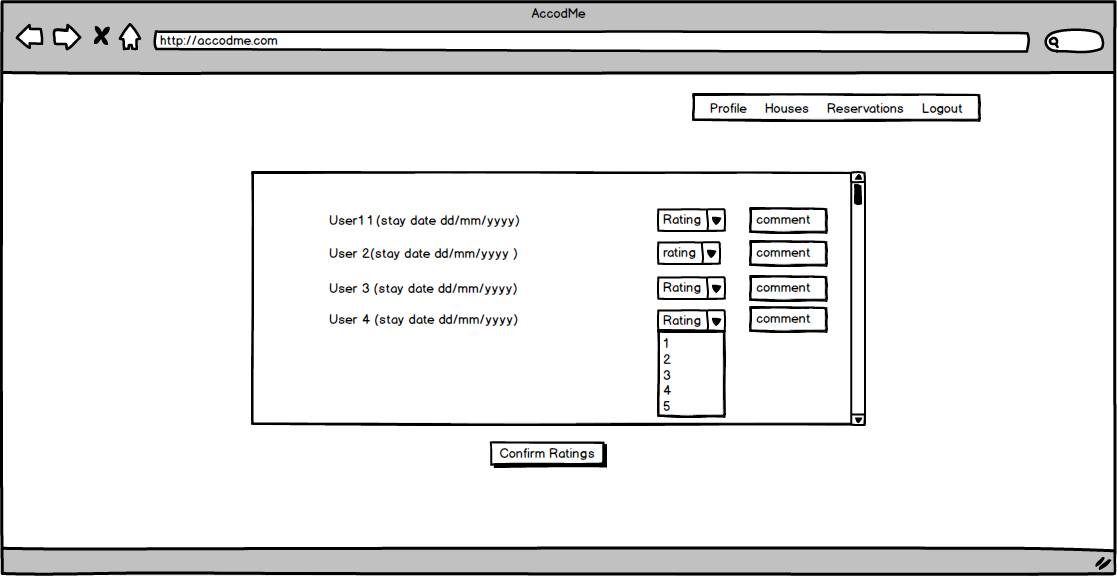
and A.hostID = $userID

and G.guestID = R.guestID

and G.guestID = U.userID;

## 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

## 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;

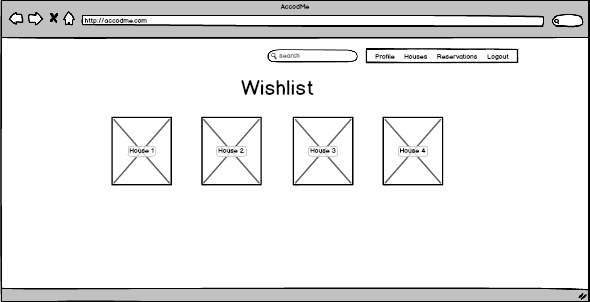
Inputs: $reservationID

DELETE FROM Reservation

WHERE reservationID = $reservationID

## Wishlist

The user can view their wishlisted houses via this menu.



select \*

from Accommodation

where Accommodation.id = (Select Wishes. accommodationID from Wishes, User where User.userID = Wishes.userID

# Advanced Database Components

## 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;**

## 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.