CS 353

Database Management Systems

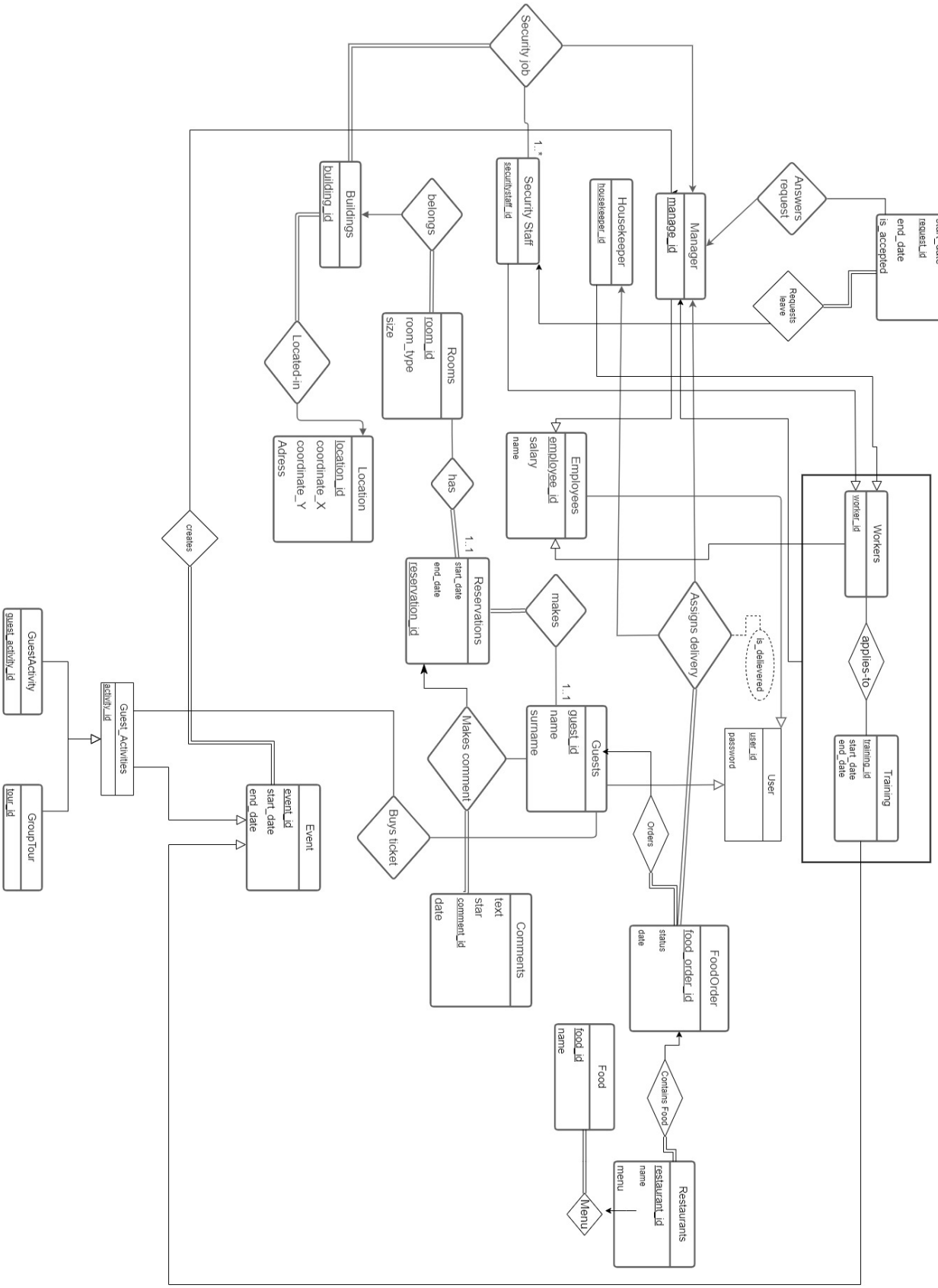
Group 41, Hotel Database Management System

Design Report

April 2, 2021

- Faruk Balcı 21702607
- Uğur Erdem Seyfi 21801744
- Agil Aliyev 21701367
- Alperen Öziş 21703804

1. Revised E/R


LeaveForm
start_date
request_id
end_date
is_accepted
Answers request
Requests leave
Manager
manage_id
Housekeeper
housekeeper_id
Security job
1..*
Security Staff
securitystaff_id
Buildings
building_id
belongs
Rooms
room_id
room_type
size
Located-in
Location
location_id
coordinate_X
coordinate_Y
Adress
has
1..1
Reservations
start_date
end_date
reservation_id
Employees
employee_id
salary
name
Workers
worker_id
applies-to
Training
training_id
start_date
end_date
Assigns delivery
is_delievered
User
user_id
password
makes
1..1
Guests
guest_id
name
surname
Orders
FoodOrder
food_order_id
status
date
Contains Food
Restaurants
restaurant_id
name
menu
Food
food_id
name
Menu
Makes comment
Comments
text
star
comment_id
date
Buys ticket
creates
Event
event_id
start_date
end_date
Guest_Activities
activity_id
GuestActivity
guest_activity_id
GroupTour
tour_id

2. Relations and Table Schemas

2.1 User

Relational Model
User(<u>user_id</u>, password)

Functional Dependencies
user_id → password

Candidate Key
{(user_id)}

Primary Key
(user_id)

Table Definition

```
CREATE TABLE User(
        user_id       CHAR(10) UNIQUE NOT NULL,
        password      VARCHAR(60) NOT NULL,
        PRIMARY KEY(user_id)
);
```

2.2 Guests

Relational Model
Guests(<u>guest_id</u>, name, surname)

Functional Dependencies
guest_id → name, surname

Candidate Key
{(guest_id)}

Primary Key
(guest_id)

Table Definition

```
CREATE TABLE Guests(
        user_id       CHAR(10) UNIQUE NOT NULL,
        name          VARCHAR(60) NOT NULL,
        surname       VARCHAR(60) NOT NULL,
        PRIMARY KEY (user_id)
);
```

2.3 Employees

Relational Model
Employees(employee_id, salary, name)

Functional Dependencies
employee_id→ salary, name

Candidate Key
{(employee_id)}

Primary Key
(employee_id)

Table Definition

```
CREATE TABLE Employees(
        employee_id        CHAR(10) UNIQUE NOT NULL,
        salary             CHAR(60) NOT NULL,
        name               VARCHAR(60) NOT NULL,
        PRIMARY KEY(employee_id));
```

2.4 Housekeeper

Relational Model
Housekeeper(housekeeper_id)

Functional Dependencies
No dependencies

Candidate Key

{(housekeeper_id)}

Primary Key
(housekeeper_id)

Table Definition

```
CREATE TABLE Housekeeper(
        housekeeper_id CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY (housekeeper_id)
);
```

2.5 SecurityStaff

Relational Model
SecurtiyStaff(securitystaff_id)

Functional Dependencies
No dependencies

Candidate Key
{(securitystaff_id)}

Primary Key
(securitystaff_id)

Table Definition

```
CREATE TABLE Guests(
        securitystaff_id      CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY (securitystaff_id)
);
```

2.6 Manager

Relational Model
Manager(Manager_id)

Functional Dependencies
No dependencies

Candidate Key
{(Manager_id)}

Primary Key
(Manager_id)

Table Definition

```
CREATE TABLE Manager(
        Manager_id          CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(Manager_id)
);
```

2.7 Workers

Relational Model
Workers(worker_id)

Functional Dependencies
No dependencies

Candidate Key
{(worker_id)}

Primary Key
(worker_id)

Table Definition

```
CREATE TABLE Manager(
        worker_id    CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(worker_id)
);
```

2.8 LeaveForm

Relational Model
LeaveForm(start_date, request_id, end_date, is_accepted, manager_id, securitystaff_id)

Functional Dependencies
No dependencies

Candidate Key
{(request_id)}

Primary Key
(request_id)

Table Definition

```
CREATE TABLE LeaveForm(
        request_id CHAR(10),
        start_date DATE,
        end_date DATE,
        is_accepted BOOLEAN,
        manager_id CHAR(10),
        securitystaff_id CHAR(10) NOT NULL,
        PRIMARY KEY(request_id),
        FOREIGN KEY (manager_id) REFERENCES Manager(manager_id),
        FOREIGN KEY (securtystaff_id) REFERENCES
SecurtyStaff(securtystaff_id)
);
```

2.9 Training

Relational Model
Training (training_id, start_Date, end_date)

Functional Dependencies
No functional dependencies

Candidate Key
{(training_id)}

Primary Key
(training_id)

Table Definition

```
CREATE TABLE User(
```

```
        training_id   CHAR(10) UNIQUE NOT NULL,
        start_date    DATE NOT NULL,
        end_date      DATE NOT NULL,
        PRIMARY KEY(training_id )
);
```

2.10 FoodOrder

Relational Model

FoodOrder (food_order_id, , guest_id, food_id, restaraunt_id, assigned_housekeeper_id, date, status)

Functional Dependencies

No functional dependencies

Candidate Key

{(food_order_id)}

Primary Key

(food_order_id)

Table Definition

```
CREATE TABLE FoodOrder(
        food_order_id CHAR(10) NUT NULL,
        food_id CHAR(10) NOT NULL,
        guest_id CHAR(10) NOT NULL,
        restaraunt_id CHAR(10) NOT NULL,
        assigned_housekeeper_id CHAR(10),
        date DATE NOT NULL
        STATUS CHAR(10) NOT NULL
        PRIMARY KEY(food_order_id),
        FOREIGN KEY (guest_id) REFERENCES Guests(guest_id),
        FOREIGN KEY (food_id) REFERENCES Food(food_id)
        FOREIGN KEY (restaraunt_id) REFERENCES Restaraunts(restaraunt_id)
        FOREIGN KEY (assigned_housekeeper_id) REFERENCES
Employee(employee_id)
);
```

2.11 Food

Relational Model

Food(food_id, name, food_order_id, restaurant_id)

Functional Dependencies
food_id→ name, restaurant_id

Candidate Key
{(food_id)}

Primary Key
(food_id)

Table Definition

```
CREATE TABLE User(
	food_id		CHAR(10) UNIQUE NOT NULL,
	name		VARCHAR NOT NULL,
	food_order_id	CHAR(10) NOT NULL,
	PRIMARY KEY(food_id ),
	FOREIGN KEY (food_order_id) REFERENCES
FoodOrder(food_order_id),
	FOREIGN KEY (restaurant_id) REFERENCES
Restaurants(restaurant_id)
);
```

2.12 Restaurants

Relational Model
Restaurants(restaurant_id, name, menu)

Functional Dependencies
restaurant_id → name, menu

Candidate Key
{(restaurant_id)}

Primary Key
(restaurant_id)

Table Definition

```
CREATE TABLE User(
```

```
        restaurant_id        CHAR(10) UNIQUE NOT NULL,
        name                 CHAR(10) UNIQUE NOT NULL,
        menu          VARCHAR NOT NULL,
        PRIMARY KEY(restaurant_id )
);
```

2.13 Comments

Relational Model

Comments(comment_id, text, star, date, guest_id, reservation_id)

Functional Dependencies

comment_id → text, start, date

Candidate Key

{(comment_id)}

Primary Key

(comment_id)

Table Definition

```
CREATE TABLE (
        comment_id CHAR(10) UNIQUE NOT NULL,
        text            CHAR(300) NOT NULL,
        star            INT(10) NOT NULL,
        guest_id        CHAR(10) NOT NULL,
        reservation_id CHAR(10) NOT NULL,
        PRIMARY KEY(comment_id),
        FOREIGN KEY guest_id REFERENCES Guests(guest_id),
        FOREIGN KEY reservation_id REFERENCES
Reservations(reservation_id)
);
```

2.14 Reservations

Relational Model

Reservations(reservation_id, guest_id, start_date, end_date)

Functional Dependencies

No dependencies

Candidate Key
{(reservation_id)}

Primary Key
(reservation_id)

Table Definition

```
CREATE TABLE User(
        reservation_id      CHAR(10) UNIQUE NOT NULL,
        guest_id            CHAR(10) NOT NULL,
        start_date          DATE NOT NULL
        end_date            DATE
        PRIMARY KEY(reservation_id ),
        FOREIGN KEY guest_id REFERENCES Guests(guest_id)
);
```

2.15 Rooms

Relational Model
Rooms(room_id, room_type, size, building_id, reservation_id)

Functional Dependencies
no dependencies

Candidate Key
{(room_id)}

Primary Key
(room_id)

Table Definition

```
CREATE TABLE Rooms(
        room_id             CHAR(10) UNIQUE NOT NULL,
        room_type           VARCHAR NOT NULL,
        size                CHAR(10) NOT NULL,
        building_id         CHAR(10) NOT NULL,
        reservation_id      CHAR(10)
        PRIMARY KEY(room_id ),
        FOREIGN KEY building_id REFERENCES Buildings(building_id),
```

```
	FOREIGN KEY reservation_id REFERENCES
Reservations(reservation_id)
);
```

2.16 Buildings

Relational Model

Buildings (building_id, location_id)

Functional Dependencies

no dependencies

Candidate Key

{(building_id)}

Primary Key

(building_id)

Table Definition

```
CREATE TABLE User(
	building_id		CHAR(10) UNIQUE NOT NULL,
	PRIMARY KEY(building_id ),
	FOREIGN KEY location_id REFERENCES Locations(location_id)
);
```

2.17 Location

Relational Model

Location (location_id, coordinate_X, coordinate_Y, Adress)

Functional Dependencies

no dependencies

Candidate Key

{(location_id)}

Primary Key

(location_id)

Table Definition

```
CREATE TABLE User(
        location_id          CHAR(10) UNIQUE NOT NULL,
        coordinate_X         CHAR  NOT NULL,
        coordinate_Y         CHAR  NOT NULL,
        Adress               VARCHAR  NOT NULL,
        PRIMARY KEY(location_id )
);
```

2.18 Event

Relational Model

Event (event_id, start_date, end_date, manager)

Functional Dependencies

event_id -> start_date, end_date

Candidate Key

{(event_id)}

Primary Key

(event_id)

Table Definition

```
CREATE TABLE Event(
        event_id      CHAR(10) UNIQUE NOT NULL,
        manager       CHAR(10) NOT NULL,
        start_date    DATE,
        end_date      DATE,
        PRIMARY KEY(event_id ),
        FOREIGN KEY (manager) REFERENCES (Manager(manager_id))
);
```

2.19 Guest_Activities

Relational Model

Guest_Activities(activity_id)

Functional Dependencies

no dependencies

Candidate Key

{(activity_id)}

Primary Key
(activity_id)

Table Definition

```
CREATE TABLE User(
        activity_id             CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(activity_id )
);
```

2.20 GroupTour

Relational Model
GroupTour(tour_id)

Functional Dependencies
no dependencies

Candidate Key
{(tour_id)}

Primary Key
(tour_id)

Table Definition

```
CREATE TABLE GroupTour(
        tour_id         CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(tour_id)
);
```

2.21 Guest_Activity

Relational Model
Guest_Activity(guess_activity_id)

Functional Dependencies
no dependencies

Candidate Key
{(guess_activity_id)}

Primary Key
(guess_activity_id)

Table Definition

```
CREATE TABLE User(
        guess_activity_id   CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(guess_activity_id )
);
```

Answers request

LeaveForm'a Manager foreign keyi ekle - done

Request leave

LeaveForm'a Security Staff foreign keyi ekle ve not null constraint koy - done

2.22 AppliesTo

Relational Model
applies-to(worker_id, training_id)

Functional Dependencies
No dependencies

Candidate Key
{((worker_id, training_id))}

Primary Key
(worker_id, training_id)

Table Definition

```
CREATE TABLE applies-to(
        worker_id     CHAR(10) UNIQUE NOT NULL,
        training_id   CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(worker_id, training_id),
```

```
        FOREIGN KEY (worker_id) references Workers(worker_id),
        FOREIGN KEY (training_id) references Training(training_id)
);
```

2.23 Security-Job

Relational Model
Security-Job(manager_id, security_id, building_id)

Functional Dependencies
No dependencies

Candidate Key
{((manager_id, security_id, building_id))}

Primary Key
(manager_id, security_id, building_id)

Table Definition

```
CREATE TABLE Security-Job(
        manager_id CHAR(10) UNIQUE NOT NULL,
        security_id  CHAR(10) UNIQUE NOT NULL,
        building_id CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(manager_id, security_id, building_id),
        FOREIGN KEY (manager_id) references Manager(manager_id),
        FOREIGN KEY (security_id) references SecurityStaff(security_id),
        FOREIGN KEY (building_id) references Building(building_id)
);
```

2.24 Buys-Ticket

Relational Model
Buys-Ticket(activity_id, guest_id)

Functional Dependencies
No dependencies

Candidate Key
{((activity_id, guest_id))}

Primary Key
(activity_id, guest_id)

Table Definition

```
CREATE TABLE Security-Job(
        activity_id    CHAR(10) UNIQUE NOT NULL,
        guest_id  CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(activity_id, guest_id),
        FOREIGN KEY (activity_id) references Guest_Activities(activity_id),
        FOREIGN KEY (guest_id) references Guests(guest_id)
);
```

2.25 MakesComment

Relational Model
Buys-Ticket(reservation_id, guest_id, comment_id)

Functional Dependencies
No dependencies

Candidate Key
{((activity_id, guest_id))}

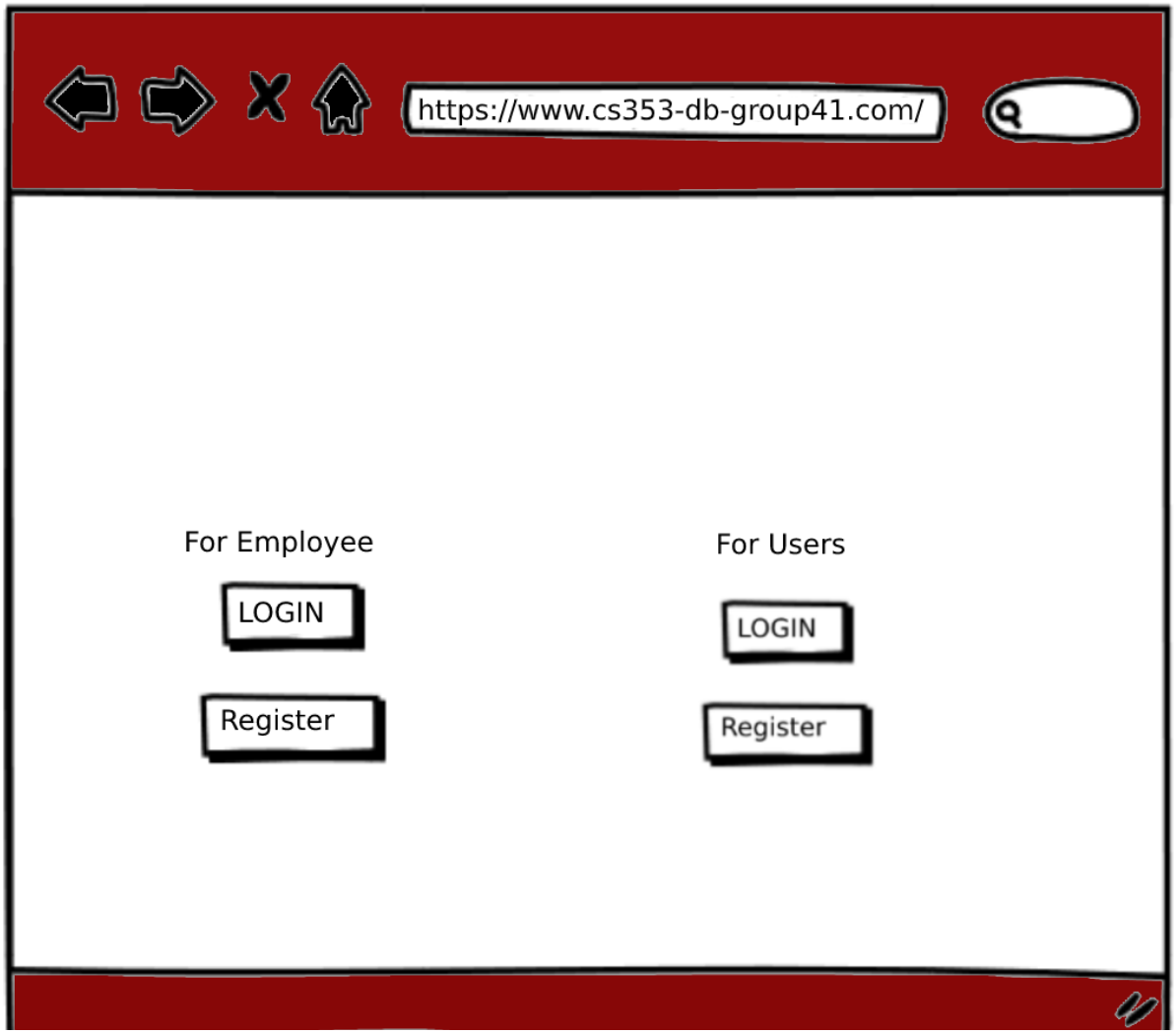
Primary Key
(activity_id, guest_id)

Table Definition

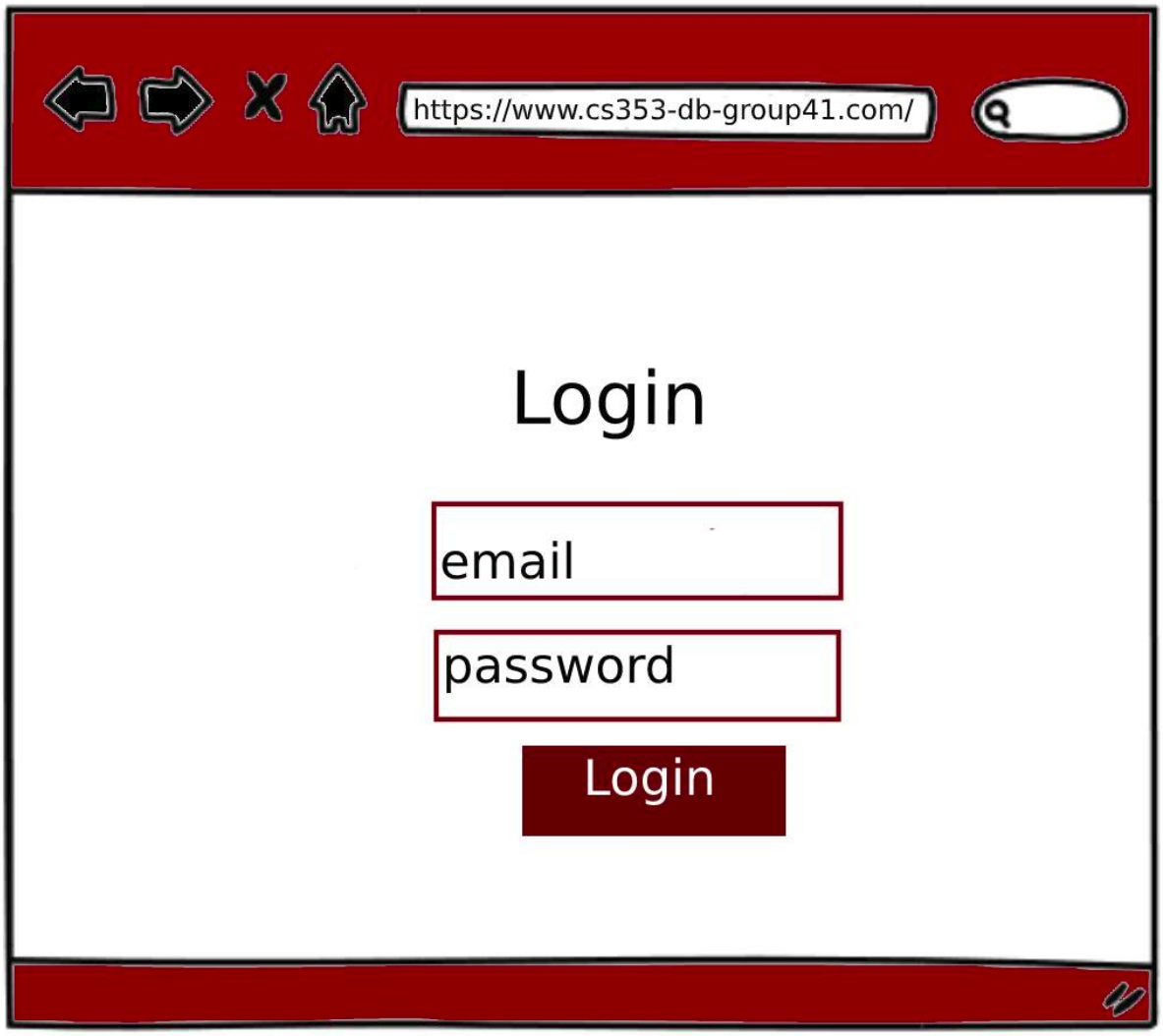
```
CREATE TABLE Security-Job(
        activity_id    CHAR(10) UNIQUE NOT NULL,
        guest_id  CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY(activity_id, guest_id),
        FOREIGN KEY (activity_id) references Guest_Activities(activity_id),
        FOREIGN KEY (guest_id) references Guests(guest_id)
);
```

3. UI Design and corresponding SQL statements

3.1. Homepage.



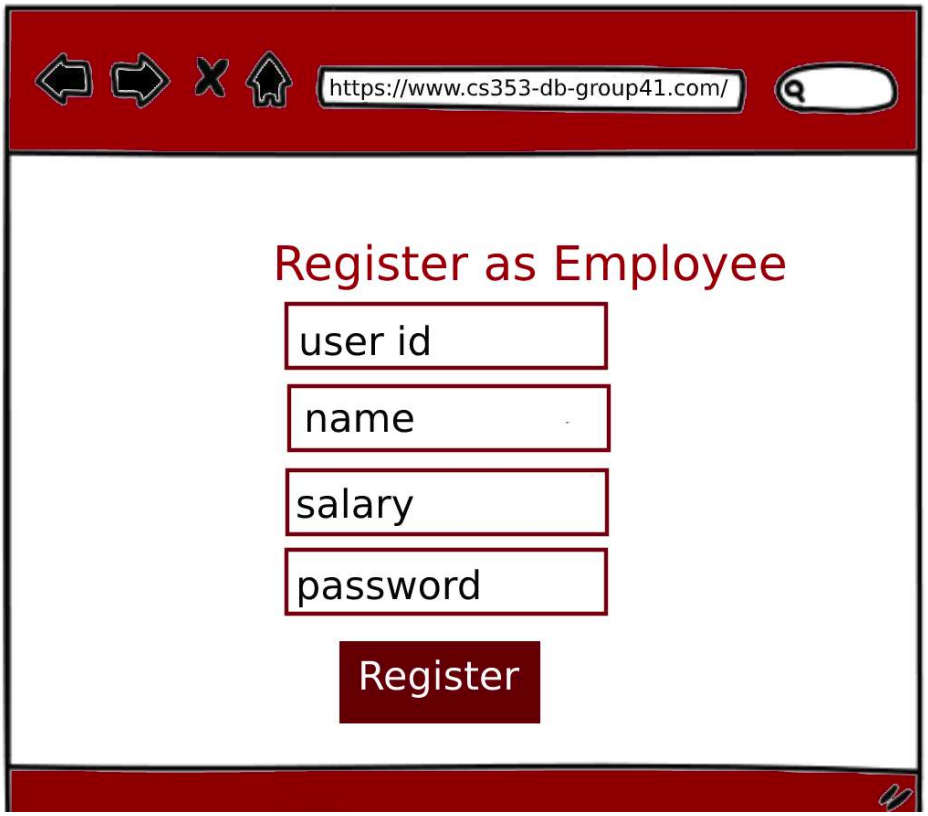
3.2 Login Page



Login with email and password:

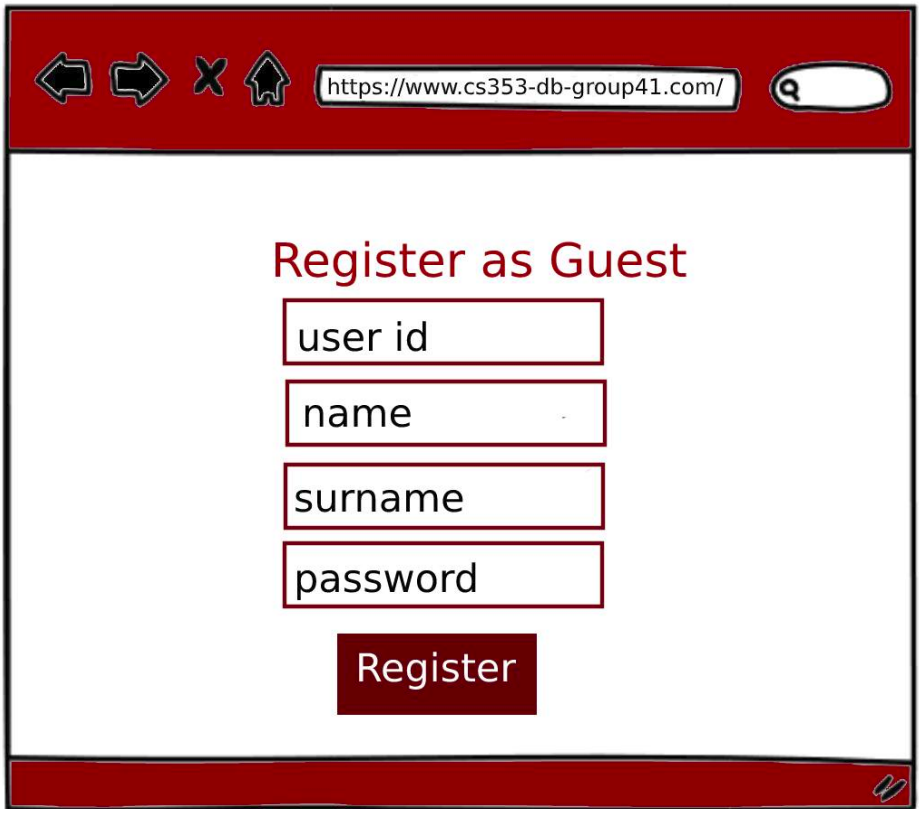
```
SELECT *
FROM User
WHERE User.user_id = @email and user.password = @password
```

3.3 Register Page



Register as Employee:
INSERT INTO Employees(employee_id, salary, name)
Values(@user_id, @salary, @name)

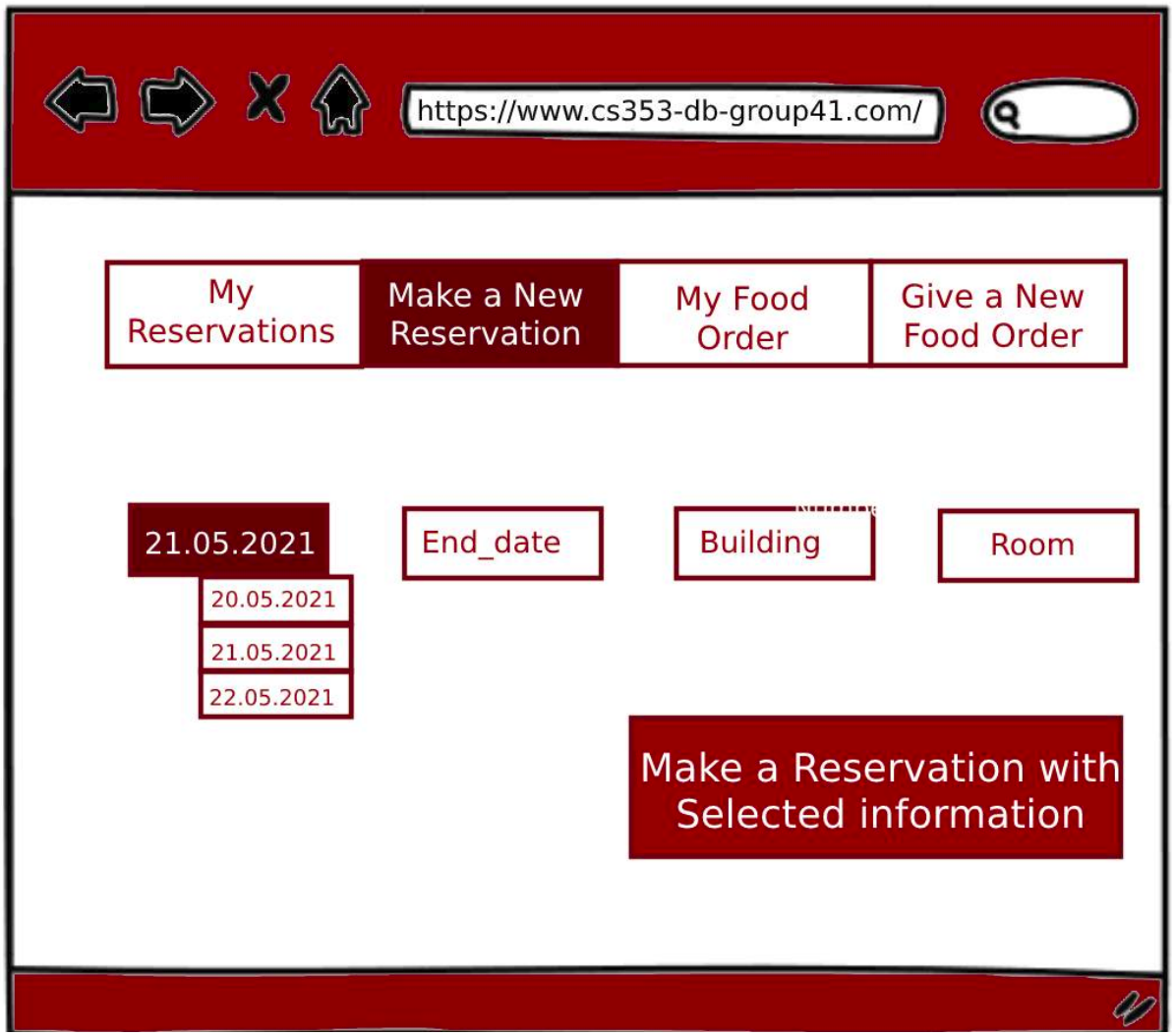
INSERT INTO User(@user_id, @password)
Values(@user_id, @password)



Register as Guest:
INSERT INTO Guest(guest_id, name, surname)
Values(@user_id, @name, @surname)

INSERT INTO User(@user_id, @password)
Values(@user_id, @password)

3.4 Guest: Make Reservations



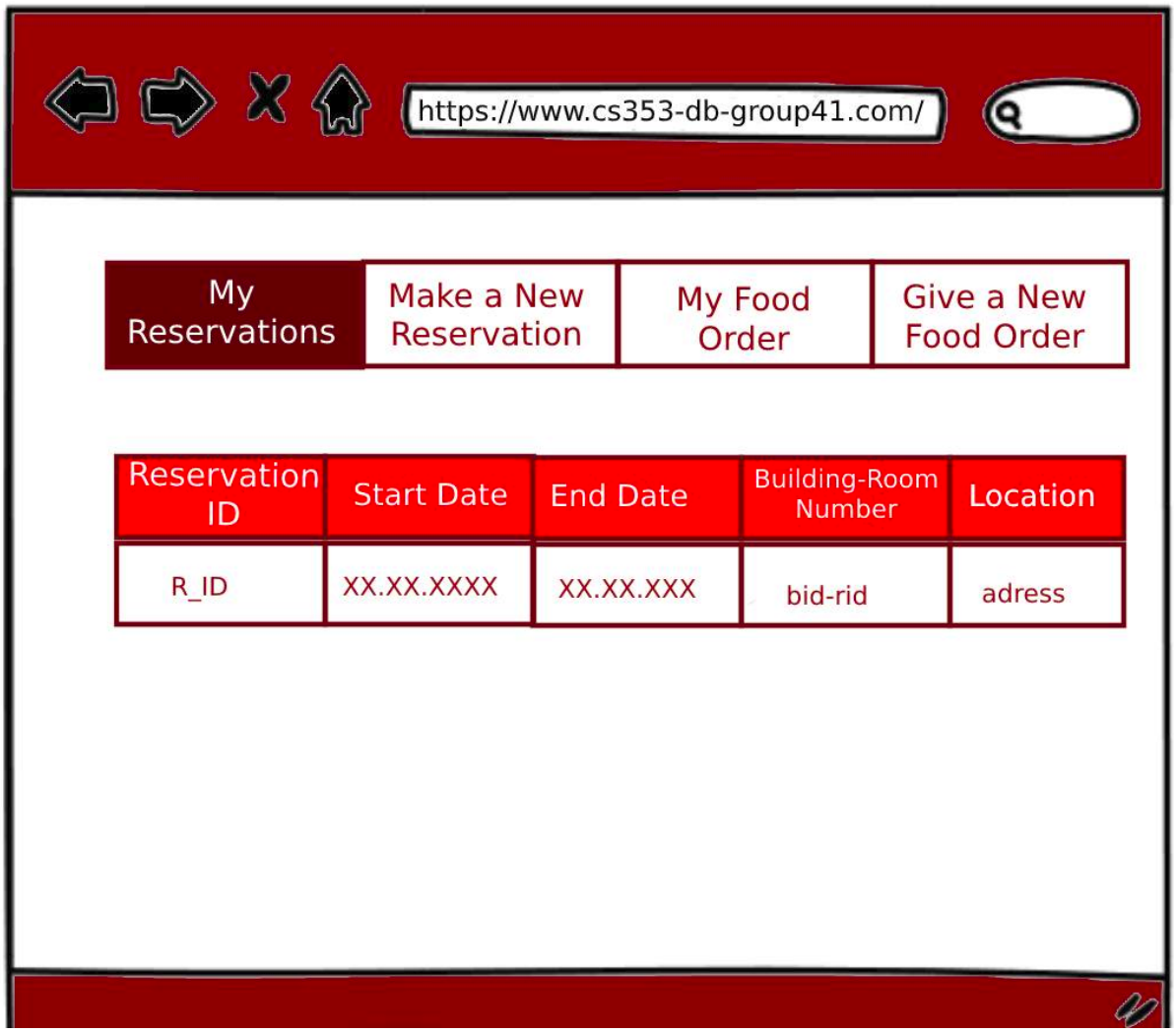
When browsing Buildings:
SELECT B.id
FROM Buildings B

When browsing Rooms:
SELECTt R.room_id
FROM Rooms R, Buildings B
WHERE R.building_id = B.id
AND B.id = @selected_building_id

When “Make reservation” button clicked:
note: @reservation_id will be generated by program.

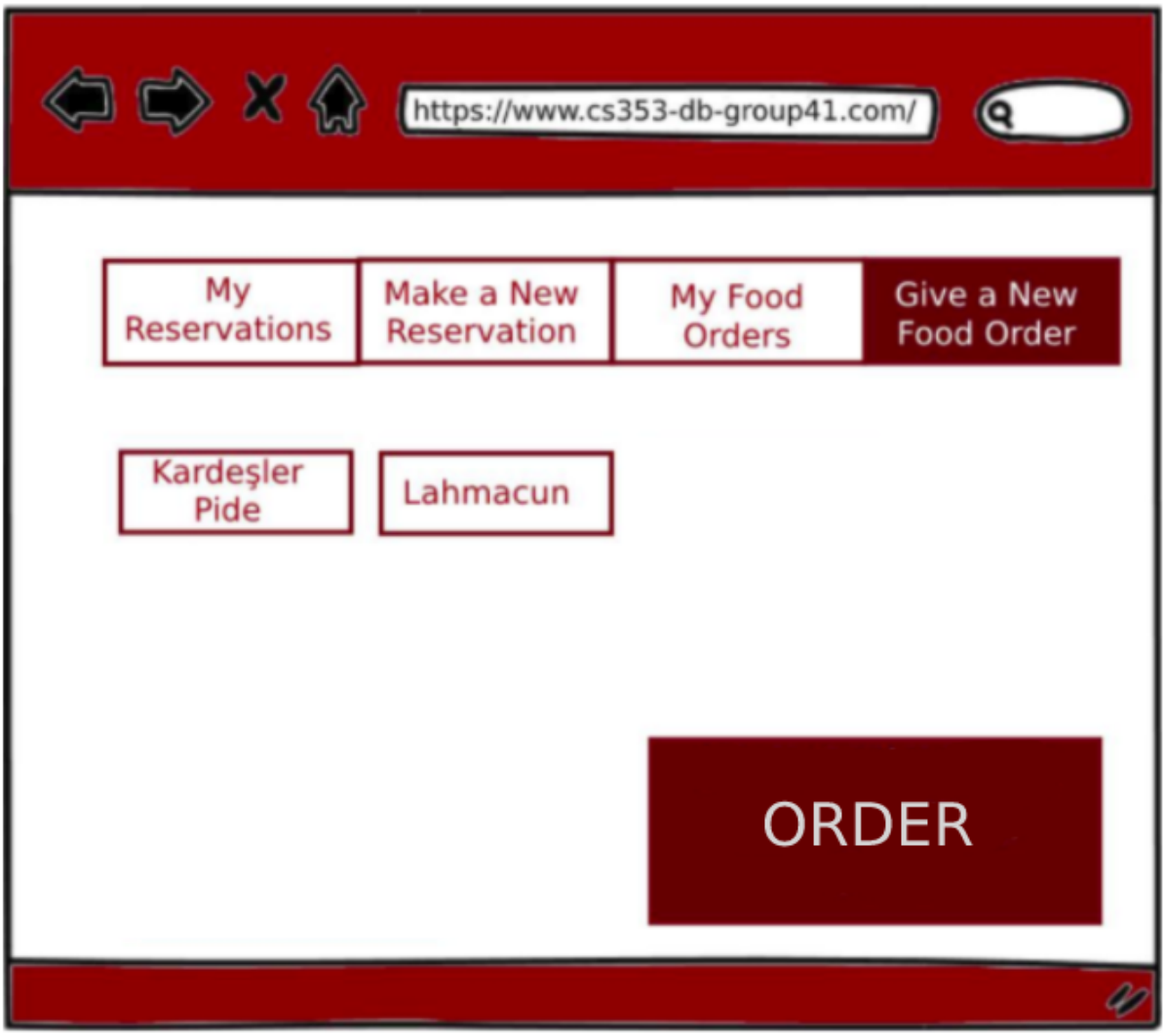
```
INSERT INTO Reservations(start_date, end_date,reservation_id)
Values(@start_date, @end_date, @reservation_id)

UPDATE Rooms
SET reservation_id = @reservation_id
WHERE room_id = @room_id
```

3.5 Guest: Show Reservations




```
SELECT R.reservation_id, R.start_date, R.end_date, B.building_id,
Rooms.room_id, L.address
FROM Reservations as R, Buildings as B, Rooms, Locations as L
WHERE R.reservation_id = Rooms.reservation_id AND Rooms.building_id =
B.building_id AND L.location_id = B.location_id
```

3.6 Guest: Give Food Order

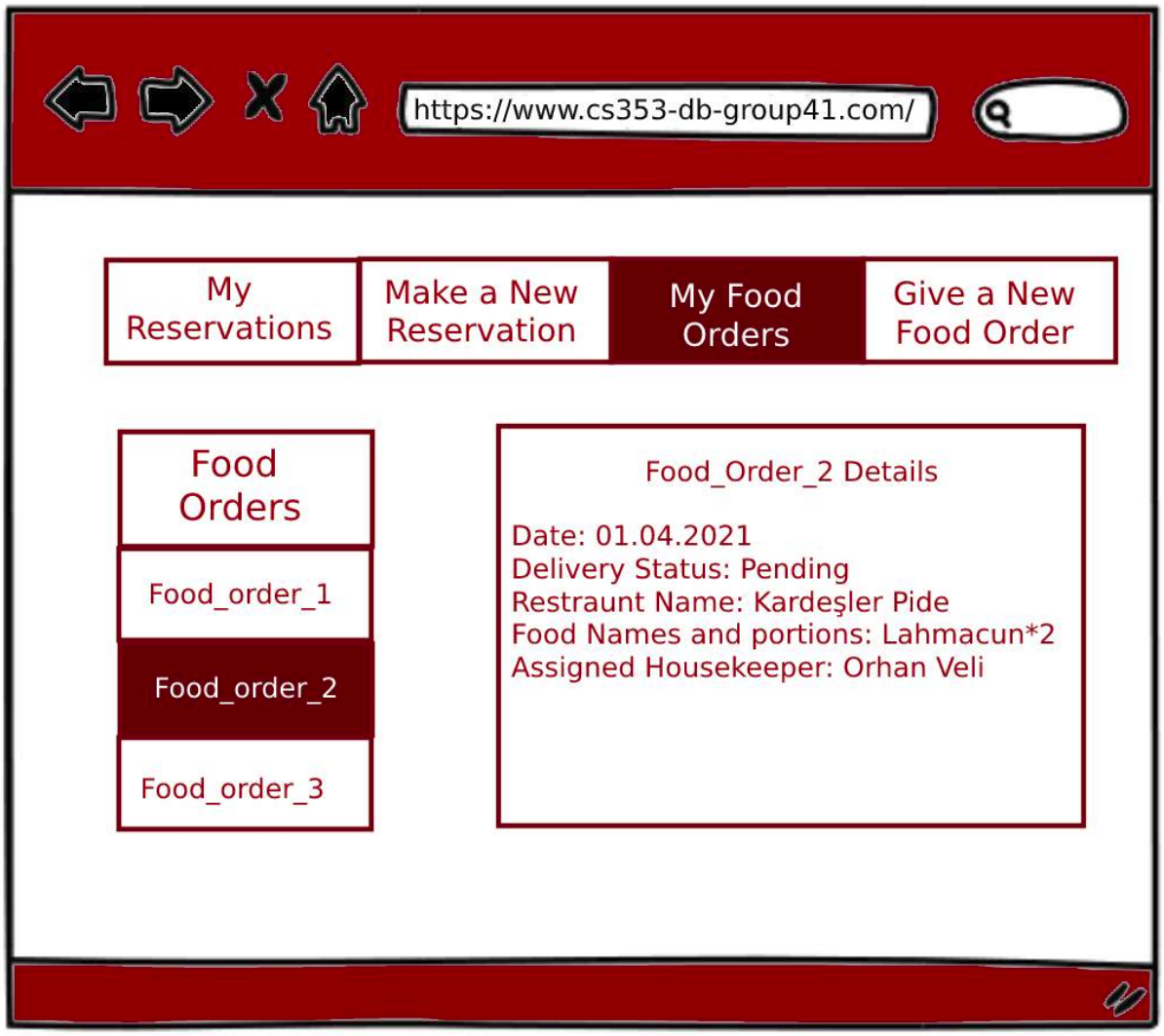


When browsing restaurants:
SELECT name
FROM restaurants

When browsing foods:
SELECT F.name
FROM Food as F, Restaurants as R
WHERE F.restaurant_id = (SELECT restaurant_id FROM Restaurants
WHERE name = @restaurant_name)

When order button clicked:
note: @food_order_id will be generated by program.
INSERT INTO FoodOrder (food_order_id, , guest_id, food_id, restaraunt_id, assigned_housekeeper_id, date, status)
VALUES @food_order_id @guest_id,
(Select food_id FROM Food as F, Restaurants as R WHERE F.restaurant_id = R.restaurant_id AND @foodname = F.name AND R.name = @restaurant_name),
(SELECT restaraunt_id FROM Restaurants WHERE r_name = @restaraunt_name),
NULL, @current_date, “order_given”

3.7 Guest: Show Food Orders



Listing Food Orders:
SELECT O.food_order_id
FROM FoodOrders O
Where O.guest_id = @guest_id

Listing details of a food order:

```
SELECT O.food_order_id, O.date, O.status, R.restaurant_name, F.fname,
E.employee_id
FROM FoodOrders O, Restaraunts R, Food F, Employee E
WHERE O.assigned_housekeeper_id = E.employee_id
AND O.restaraunt_id = R.restaraunt_id
AND O.food_id = F.food_id
AND O.food_order_id = @selected_order_id
```